Five steps for building consistent dynamic process models and their implementation in the computer tool modeller

Five Steps for Building

Consistent Dynamic Process Models

and their Implementation in the Computer Tool Modeller

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Westerweele, Mathieu R.

Five steps for building consistent dynamic process models : and their implementation in the computer tool Modeller / by Mathieu R. Westerweele.-
Eindhoven : Technische Universiteit Eindhoven, 2003.
Proefschrift. - ISBN 90-386-2964-8
NUR 913
Trefwoorden: procesregeling en dynamica / chemische technologie; fysisch-chemische simulatie en modellering / dynamische simulatie; methodologie / wiskundige modellen / computerapplicatie; algoritme
Subject headings: process control and dynamics / chemical engineering; physicochemical simulation and modelling / dynamic simulation; methodology / mathematical models / computer application; algorithm

Omslag: Govert Sleegers
Druk: Universiteitsdrukkerij TU Eindhoven, The Netherlands

Five Steps for Building

Consistent Dynamic Process Models

and their Implementation in the Computer Tool Modeller

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Eindhoven,
op gezag van de Rector Magnificus,
prof.dr. R.A. van Santen, voor een commissie
aangewezen door het College voor Promoties
in het openbaar te verdedigen op
maandag 14 april 2003 om 16.00 uur

door

Mathieu Ruurd Westerweele

geboren te Oostburg

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.Dipl-Ing. H.A. Preisig
en
prof.dr.ir. A.C.P.M. Backx

THERE IS NO TIME LIKE THE PRESENT

Mark Twain said, "I have been through some terrible
things in my life, some of which actually happened".
These words hold great wisdom;
the wisdom of knowing that there is no sense worrying
about the past or fearing the future.
Your past will not change,
and your future concerns may never come.

Many people spend a great deal of time and energy
thinking about past problems and future concerns,
while the present is passing them by.
Remember the here and now is a gift,
which is why they call it the 'present'.
Treat it like a present,
enjoy it to the fullest and make the most of it.

Don't let your life pass you by,
focus on the present and enjoy today!

- Glen Hopkins

Contents

Summary

The main goal of this research project was the development of a systematic modelling method for the design of first principles based (i.e. based on physical insight), dynamic process models for macroscopic physical, chemical and/or biological processes. The modelling of such processes is one of the most important tasks of a process engineer, for these models are used on a large scale for all kinds of engineering activities, such as process control, opimisation, simulation, process design and fundamental research. The construction of these models is, in general, seen as a difficult and very time consuming task and is preferably handed over to "modelling experts". This does not have to be the case if a clear, stepwise method is adhered to.

The modelling methodology is based on the hierarchical decomposition of processes into thermodynamic systems and consists of five steps: **(i)** Subdivide the process into smaller parts, that can be easily (mathematically) described individually. In this step, the so-called *Physical Topology* is constructed, using only two basic building blocks, namely *Systems*, which represent capacities that are able to store the fundamental extensive quantities (such as component mass, energy and momentum), and *Connections*, which describe the transfer of these fundamental extensive quantities through the common boundaries of the interconnected systems. **(ii)** Construct the *Species Topology*. The Species Topology describes the distribution of all involved chemical and/or biological species as well as all their transformation into other species in the various parts of the process. **(iii)** For each relevant extensive quantity (such as, for example, component mass or energy) of each system write the corresponding balance equations. **(iv)** Augment the model equations with definitions and equations of state and choose the transfer laws and kinetic laws that express the flow and production rates (present in the balance equations). Also define possible constraining assumptions (i.e. "simplifications" that may lead to high-index DAEs (Differential Algebraic Equations)). The resulting mathematical problems can always be resolved by applying a model reduction. **(v)** Add the dynamic behaviour of the information processing units, such as controllers.

These steps for building a model do not have to be done strictly in this

sequence - at least not for the overall model. It is left to the model designer when the details are being specified in each part of the model.

This five step procedure of building dynamic process models always results in a differential algebraic system with an index of one. The model can be used for solving certain problems related to the process or it can be further modified by applying mathematical manipulations, such as linearisation or model reduction.

An other important goal of this project was the implementation of the modelling methodology into a new computer tool, called "The Modeller". This tool is designed to effectively assist a model designer with the construction of consistent process models and to (significantly) reduce the needed time and overall effort. With the modelling tool, the process models can be built using two main operations, namely refining existing systems (the *top-down* approach) or grouping systems (the *bottom-up* approach). These two operations can be used interchangeably and all performed actions can be undone (multiple undo/redo mechanism). The software allows to store, retrieve, import or export models (or parts of models) at any stage of the model definition. This allows for a safe mechanism of model reuse.

The computer tool is implemented with BlackBox Component Builder 1.4 (Component Pascal), which is a component oriented programming language comparable to Java.

Voorwoord

"*Het duurt even, maar dan heb je ook wat!*", zegt men wel eens. Dat gaat ook zeker op voor dit werk: het heeft iets langer geduurd dan gebruikelijk om dit promotiewerk af te ronden, maar het resultaat is een stuk werk waar ik trots op ben en waar ook anderen iets aan zullen hebben. Tijdens mijn onderzoek is een methodiek ontwikkeld waarmee men (veel) sneller dynamische proces modellen kan opstellen, die bovendien "automatisch" worden gedocumenteerd door het gebruik van deze methode. De ontwikkelde methode is reeds gedeeltelijk geïmplementeerd in het computerprogramma "MODELLER", waarmee het effect van de methode zeer goed kan worden gedemonstreerd.

Ik heb de afgelopen 5 jaar veel geleerd en daar ben ik vele mensen dankbaar voor. Allereerst natuurlijk een woord van dank aan mijn dagelijkse begeleider en promotor professor Heinz Preisig. Heinz, ik vond het erg plezierig om met je samen te werken. Je was altijd - nou ja, meestal - open voor discussies en hebt me veel geleerd op velerlei gebieden. Ik hoop dan ook het contact niet te verliezen nadat deze promotie is afgerond, zodat er nog veel goede dingen uit een blijvende samenwerking kunnen voortvloeien. Ook mijn copromotor Ton Backx, de leden van de kerncommissie, Okko Bosgra en Wolfgang Marquardt, alsook de overige leden van de promotiecommissie, Costas Pantelides, Piet Kerkhof, Jos Keurentjes en Johan Wijers, wil ik van harte bedanken voor hun steun, interesse, constructieve discussies en feedback.

In de eerste jaren van mijn promotieonderzoek was Martin Weiss mijn directe begeleider. Ik heb van hem veel geleerd (vooral op wiskundig gebied) en vind het jammer dat hij er in de latere fase van mijn onderzoek niet meer bij was. Martin, bedankt voor je steun en interessante discussies. Ook de studenten, Steven Grijseels en Reinout Dekkers, die mij in de eerste fase van mijn onderzoek hebben geholpen met het uitzoeken van bepaalde zaken, wil ik hartelijk bedanken voor hun inzet en bijdrage.

Naast mijn bijdrage op wetenschappelijk gebied heb ik ook op educatief gebied een steentje proberen bijdragen. Ik wil Henk Leegwater hartelijk bedanken voor de ruime mogelijkheden mezelf op dit gebied te ontwikkelen. Ik

heb van deze periode ontzettend veel geleerd en heb daarin ook zeer veel plezier gehad. Ook de student-assistenten, M'hamed, Leo, Frans, Hans-Jan, Hans en Alex, die hebben geholpen het practicum op poten te zetten en te houden, wil ik hartelijk bedanken voor hun enorme inzet.

Voor de fijne werksfeer die rond mij heerste in de eerste fase van mijn onderzoek wil ik graag mijn ex-collega aio's Annelies, Georgo, Gerwald, Patrick, Roel en Uwe, alsook de overige medewerkers van onze ex-vakgroep, Tiny, Tanja, Thea, Frank, Stefan, Ivonne, Henk, Martin en Heinz, bedanken.

In de tweede fase van mijn onderzoek was ik gehuisvest bij de regeltechniek vakgroep van de faculteit electrotechniek. Ik wil deze gehele vakgroep en met name Paul van de Bosch hartelijk bedanken voor hun bereidheid onze groep op te nemen in de hunne alsook voor de plezierige werksfeer tijdens de afronding van mijn promotiewerk.

Govert Sleegers wil ik bedanken voor het ontwerpen van de omslag van dit proefschrift. Goof, ondanks dat je waarschijnlijk geen bal van de inhoud van dit proefschrift begrijpt (of wilt begrijpen), vind ik toch dat je er erg goed in geslaagd bent een beeld hiervan te schetsen op de omslag. Hartelijk dank voor het fraaie ontwerp.

Voor het wekelijkse vertier in de vorm van squash, pool, spelletjes dan wel muziek wil ik graag Xander, Jip, Bart, Martin, Arjan, Paul, Emiel, Patrick, Ignace, Jan, Edwin, Luc, Ruben, Rogier en de leden van Quadrivium en Bigband de Mooche bedanken alsook alle anderen die me tijdens deze brainstormsessie niet te binnen zijn geschoten.

Mijn ouders en schoonouders alsook de rest van mijn familie wil ik hartelijk bedanken voor hun steun, geduld en interesse. Pa en ma, hartelijk bedankt mij de mogelijkheid te bieden zover te kunnen komen. Al mijn aangetrouwde "broers" en "zussen", Frank, Kurt, Mark, John, Annemiek en Bianca, maar in het bijzonder mijn eigen broer en zus, Michel en Esther, wil ik hartelijk bedanken voor hun interesse, steun en vele goede hints.

Het laatste bedankje gaat natuurlijk richting de belangrijkste mensen in mijn leven: mijn eigen gezinnetje. Lisanne, Kevin en Nathalie (en zelfs ook ons volgende kindje, die pas over enkele maanden het daglicht voor het eerst zal zien), bedankt voor de vele plezierige momenten en de nodige inspiratie die jullie me hebben bezorgd tijdens mijn promotieperiode. Jullie hebben mijn leven heel wat meer inhoud en kleur gegeven. Mijn belangrijkste steun en toeverlaat was en is natuurlijk mijn vrouw Miranda. Zonder haar geduld, steun en liefde zou ik zeker niet tot zo'n mooi eindresultaat zijn gekomen. Miranda, bedankt voor alles. Ik hoop dat we nog vele jaren gelukkig kunnen verder leven met ons gezin.

Mathieu.

Chapter 1

Introduction

A chemical engineer is often asked to describe the dynamic and/or static behaviour of a physical-chemical-biological (PCB) process (or a set of these processes, which constitute a plant), because information about this behaviour is needed for operations like analysis, control, design, simulation, optimisation or process operation. In order to analyse the behaviour of such a process, the engineer often needs a mathematical representation of the physical and chemical and/or biological phenomena taking place in it.

The representation of a PCB process in form of a mathematical model is the key to many chemical engineering problems (e.g. (Cellier 1991) (Marquardt 1994*a*)(Ogunnaike 1994)(Hangos and Cameron 2001)). Modelling a chemical process requires the use of all the basic principles of chemical engineering science, such as thermodynamics, kinetics, transport phenomena, etc. It should therefore be approached with care and thoughtfulness (Stephanopoulos 1984).

1.1 Motivation and Goals

A (mathematical) model of a process is usually a system of mathematical equations, whose solutions reflect certain quantitative aspects (dynamic or static behaviour) of the process to be modelled (Aris 1978) (Denn 1986)(Ogunnaike 1994). The development of such a mathematical process model is initiated by mapping a process into a mathematical object. The main objective of a mathematical model is to describe some behavioural aspects of the process under investigation.

The modelling activity should not be considered separately but as an integrated part of a problem solving activity. Preisig (Preisig 1991*b*), (Preisig 1991*a*) analysed and decomposed the overall task of problem solving into the following set of subtasks:

- **(Primary) Modelling**. The first step in the process of obtaining a

process model is the mapping of the "real-world" into a mathematical object, called the primary model. In doing so, one may take different views and accordingly apply different theories, which naturally will result in different models. Within this first step, assumptions are made about the principle nature of the process (such as time scales of hydraulics and reactions, fundamental states, etc.).

- **Model manipulation.** The model can be simplified by applying mathematical manipulations, such as:
  - Model reduction
  - Linearisation
  - Transformation to alternative representations of the model
  - re-arrangement of the mathematical problem equations
- **Problem specification**. Certain variables are instantiated (i.e. defined as known), such that they are available later during solution time and such that the number of equations equals the number of unknown variables.
- **Analysis of the mathematical model**. The analysis of the mathematical problem is done in connection with the specification of the model. On the simple level, a degree of freedom analysis may be done, which for large scale systems is by no means trivial. On the higher level, one may look into things as index of the differential algebraic system.
- **Solution of the mathematical model**. General purpose mathematical packages, such as differential algebraic equation solvers, large scale simulators, linear algebra packages, etc., are used to solve particular problems.
- **Analysis of the results**. The analysis of the results must focus on a verification of the results by comparing them with known pieces of information. This may be experimental data or just experience.

The outcome of the performance of each task must fulfil a set of specifications and requirements, otherwise the design is iterated by looping back to any previous task. This implies that modelling is a recursive and iterative process (and that includes not only the modelling, but everything that is associated with the use of the model). Rarely does one in the first attempt obtain a proper model for the problem under investigation. Usually, an adequate model is constructed progressively through a loop comprising a series of tasks of model development and model validation.

More often than not, the time spent on collecting the information necessary to properly define an adequate model is much greater than the time spent by a simulator program in finding a solution. Most publications and textbooks present the model equations without a description of how the model equations have been developed. Hence, to learn dynamic model development, novice modellers must study examples in textbooks, the work of more skilled modellers, or use trial and error (Moe 1995).

During the last decades there tends to be an increasing demand for models of higher complexity, which makes the model construction even more time consuming and error-prone (MacKenzie and Ponton 1991)(Marquardt 1991)(Marquardt 1992). Moreover, there are many different ways to model a process (mostly depending on the application for which the model is to be used): different time scales, different levels of detail, different assumptions, different interpretations of (different parts of) the process, etc. Thus a vast number of different models can be generated for the same process. All this calls for a systematisation of the modelling process, comprising of an appropriate, well-structured modelling methodology for the efficient development of adequate, sound and consistent process models. Modelling tools building on such a systematic approach support teamwork, re-use of models, provide complete and consistent documentation and, not at least, improve process understanding and provide a foundation for the education of process technology (Bieszczad 2000)(Moe 1995)(Preisig *et al.* 1991).

This work presents the concepts and design of Modeller, a computer-aided modelling tool built on a structured modelling methodology, which aims to effectively assist in the development of process models and helps and directs a modeller through the different steps of this methodology. It was not a project objective to introduce new elements or a new modelling language. The objective of this project is to provide a systematic model design method that meets all the mentioned requirements and turns the art of modelling into the science of model design.

The focus of this work is primarily on modelling and not on problem solving. Most of the currently available modelling languages and simulation packages focus on model manipulation, specification, analysis and/or solution and more or less leave out the modelling part. In general it is assumed that the mathematical model of the process under investigation is known or easy to assemble. The development of process models, however, is slow, error prone and consequently a costly operation in terms of time and money (Sargent 1983)(Asbjørnsen *et al.* 1989)(Marquardt 1992).

Modelling is an acquired skill, and the average user finds it a difficult task (Sargent 1989)(Evans 1990). A modeller may inadvertently incorporate modelling errors during the mathematical formulation of a physical phenomenon.

Formulation errors, algebraic manipulation errors, writing and typographical errors are very common when a model is being implemented in a computing environment. Thus any procedure which would allow to do some of the needed modelling operations automatically would eliminate a lot of simple, low-level (and hard to detect) errors (Preisig 1991*c*).

Modeller is a computer-aided modelling tool which is designed to assist a model designer to map a process into a mathematical model, using a systematic modelling methodology. The task solved by Modeller is purely the construction and manipulation of the structure and definition of process models. The tool is not designed to solve specific problems concerning any model, as such programs (Modelica, gProms, ASCEND, Dymola, SpeedUp, DASSL, DASOLV, DAEPACK, etc.) are already available. The sole task of Modeller is to provide an interface allowing to construct and edit symbolic process models. The output of Modeller is a first-principles based (i.e. based on physical insight (Asbjørnsen *et al.* 1989)) mathematical model with a minimal realisation, which is easily transformed to serve as an input to existing modelling languages and/or simulation packages, such as gProms, ABACUSS, ASCEND, Modelica, Matlab.

1.2 Application Dependency of Models

This work is concerned with the mathematical modelling of macroscopic physical and/or chemical processes as they appear in general in chemical or biological plants. A mathematical model of a process usually consists of a set of equations, which describe the dynamic and/or static behaviour of this process. There are many ways to generate these equations and there are many different ways to describe the same process, which will usually result in different models. The approach a modeller takes when constructing a model for a process depends on:

- *the application for which the model is to be used.* Different models are used for different purposes. For example, a model which is used for the control of a process shall be different from a model which is used for the design or analysis of that same process;

- *the amount of accuracy that has to be employed.* This is of course partially depending on the application of the model and on the time-scale in which the process has to be modelled. In general, a model which needs to describe a process on a small time-scale demands more details and accuracy then the model of the same process which describes the process over a larger time-scale;

- *the view and knowledge of the modeller on the process.* Different people have different backgrounds and different knowledge and will therefore often approach the same problem in different ways, which can eventually lead to different models of the same process.

1.3 Structured Modelling Methodology

A modelling methodology can be defined as an algorithmic procedure intended to lead from specific knowledge of physical and topological nature of a process to a mathematical model of that process (Weiss 2000). The modelling methodology we use (Preisig 1994*b*) is based on the hierarchical decomposition of processes into networks of *elementary systems* and *physical connections*. Elementary systems are regarded as thermodynamic simple systems and represent (lumped) capacities able to store extensive quantities (such as component mass, energy and momentum). The connections have no capacity and represent the transfer of extensive quantities between these systems[1]. The construction of a process model with our methodology consists of the following steps:

1. Break the process down into elementary systems that exchange extensive quantities through physical connections. The resulting network represents the *physical topology.*

2. Describe the distribution of all involved chemical and/or biological species as well as all reactions in the various parts of the process. This represents the *species topology.*

3. For each elementary system and each fundamental extensive quantity (the collection of which is the fundamental state) that characterises the system write the corresponding balance equation. The result has the following form:

$$\frac{d}{dt}\underline{\mathbf{x}} = \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}}$$

   In this (simplified!) form, the balance equations run over all the fundamental extensive quantities ($\underline{\mathbf{x}}$) of all defined systems, over all the defined connections ($\underline{\mathbf{z}}$) and over all the defined reactions ($\underline{\mathbf{r}}$) of the physical topology. The matrices $\underline{\underline{\mathbf{A}}}$ and $\underline{\underline{\mathbf{B}}}$ are completely defined by the model designers definition of the physical and species topology of the process.

[1] A similar basic classification, splitting up the process in *devices* and *connections*, was develloped by Marquardt (e.g.: (Marquardt 1994*b*)).

4. Add algebraic equations to the model definition:

   - Augment the model equations with definitions and equations of state:

     $$\underline{\mathbf{y}}_{\Sigma} = \underline{\mathbf{f}}(\underline{\mathbf{x}}_{\Sigma})$$

     For each system in the physical topology a mapping must exist, which maps the primary state (i.e. the fundamental state $\underline{\mathbf{x}}_{\Sigma}$) into a secondary state ($\underline{\mathbf{y}}_{\Sigma}$).
   - Choose the transfer laws and kinetic laws that express the flow and production rates (present in the balance equations). Also define possible constraining assumptions (i.e. "simplifications" that may lead to high-index DAEs (Westerweele and Preisig 2000), (Moe 1995)), such as order of magnitude assumptions.

     $$\begin{array}{ll} \text{Transfer laws:} & \underline{\mathbf{z}}_c = \underline{\mathbf{g}}_1(\underline{\mathbf{y}}_{or}, \underline{\mathbf{y}}_{tar}) \text{ or } \underline{\mathbf{0}} = g_2(\underline{\mathbf{y}}_{or}, \underline{\mathbf{y}}_{tar}) \\ \text{Production laws:} & \mathbf{r}_{i,\Sigma} = \underline{\mathbf{h}}_1(\underline{\mathbf{y}}_{\Sigma}) \text{ or } \mathbf{0} = \underline{\mathbf{h}}_2(\underline{\mathbf{y}}_{\Sigma}) \end{array}$$

     For each connection either the transfer law must be given, which defines the flow ($\underline{\mathbf{z}}_c$) through the connection (usually as a function of secondary state variables of the two interconnected systems), or a constraint must be given. For each reaction either the production law must be given, which defines the reaction rate ($\mathbf{r}_{i,\Sigma}$) of the reaction (usually as a function of secondary state variables of the system the reaction takes place in), or a constraint must be given. When a constraint is given for a connection or a reaction, two or more fundamental state variables are linked together, either directly or indirectly. This means that not all state variables are independent and that some kind of index reduction method shall have to be applied. A detailed discussion on the problems and their solutions, when introducing assumptions, is given in chapter 6.

5. Add the dynamic behaviour of the information processing units, such as controllers.

These steps for building a model do not have to be done strictly in this sequence - at least not for the overall model. It is left to the model designer when the details are being specified in each part of the model.

This five step procedure of building dynamic process models always results in a differential algebraic system with an index of one. The model can be used for solving certain problems related to the process or it can be further modified by applying mathematical manipulations, such as linearisation or model reduction.

1.4 Overview of Related Work

This section gives a summary on several existing software packages and computer languages for dynamic process modelling. It is not the idea of this section to give a thorough review of all the different packages and languages that have been developed over the last few decades, but rather to give an idea on what kind of different types of tools have been developed.

There exist many software tools that are designed to facilitate certain aspects of the modelling process. Most of these tools are designed to solve sets of mathematical equations, which do not necessarily have to be related to a physical process. For dynamic process modelling these tools can be divided into three groups:

- **General purpose mathematical solvers**: These are very general mathematical problem solvers that are not especially designed to solve specific problems that are related to physical processes. These environments accept many kinds of mathematical problems. When used for modelling, a model designer must define all variables and equations that are of interest himself and also has to be very aware of the correct computational causality of the entered equations. It may cost a model designer a lot of work (doing substitutions, transformations and other manipulations) to generat a code that is solvable. The solvers can make no statements about the physical feasibility of the problem. A modeller should therefore have a good background in the subject and have an ability to translate the physical phenomena and structure of a process into a mathematical notation (Himmelblau and Bischoff 1968).

  Examples of general purpose mathematical solvers (involving symbolic and numerical tools) are: Matlab (*MATLAB user's guide* 1992), Maple (Blachman and Mossinghoff 1998), Mathematica (Wolfram 1991).

  Examples of more specific differential algebraic system solvers are: DASSL (Petzold 1983), LSODI (Hindmarsh 1983).

- **Generic physical process modelling languages**: These languages support a structured and declarative representation of process models. They may be viewed more as passive data structures representing structured sets of model equations rather than as modelling knowledge comprising data and inference methods. Hence, by construction, modelling languages do not provide guidance for the modeller (Marquardt 1994*b*).

  Implementations of these languages can be used for modelling of large, complex and heterogeneous physical systems. They are, in general, suited for multi-domain modelling, for example, physical process oriented applications, applications involving mechanical, electrical, hydraulic, and

control subsystems, robotics, automotive and aerospace applications, etc. The tools allow the modeller to declare (very) large systems of (differential, algebraic and discrete) equations in symbolic form, which, in general, do not have to be prepared manually in order to fit the solver. The computer takes primary responsibility for selecting and implementing the appropriate symbolic and numerical methods to determine the results. Some of the implementations (e.g. ABACUSS II and Dymola) are even capable of integrating systems with high-index differential and algebraic equations.

By automatically determining the solution to the user-specified equations, these tools are a significant aid to equation-oriented modelling. They significantly reduce the amount of coding needed to develop new models or modify existing ones and are capable of solving very large systems of equations. However, the solution capabilities cannot be fully exploited unless the correct, consistent generation and maintanance of these complex sets of equations can be ensured (Bieszczad 2000). The mathematical modelling tools cannot provide assistence on modelling itself because they focus on equation solving methods and are not explicitly linked to physical concepts.

Bieszczad rightfully wrote: "Furthermore, although the symbolic form of the equations facilitates reuse, their applicability may be uncertain because assumptions used in deriving these equations are not explicitly maintained. Thus, it is the responsibility of the modeler to provide comments explaining assumptions used as the basis of model equations, to analyse the consistency and logic of these assumptions, and to correctly derive the equations. These tasks are neither assisted nor enforced by any of the equation-based modelling tools. This can especially lead to difficulty in model editing and analysis. For example, to avoid redundant modeling it would be ideal if one continuously evolving model could be used over the lifetime of a project. Over time, such a model may grow to hundreds or thousands of equations while modifications are made by several different modelers. However, whenever an assumption is changed or added, the modeller must analyse the set of existing model equations (which may or may not be consistently commented), determine the modifications necessary throughout the system of equations, then implement and document these changes. As a result, this task may quickly become unwieldy as a model grows in complexity." (Bieszczad 2000)

Examples of modelling languages and their implementations are: Dymola (Modelica) (Mod 2000)(Elmqvist and Mattsson 1997), Omola, gProms (Barton 1992)(Barton and Pantelides 1993), ABACUSS II (Freehery and Barton 1996), ASCEND (Piela 1989)(Piela and Westerberg 1993),

YAHSMT (Thevenon and Flaus 2000), SimGen (Dormoy and Furic 2000), SpeedUp (now distributed by AspenTech) (Pantalides 1998), Chi (Fábián 1999).

- **Automated model derivation programs for physical/chemical processes**:

  The basic idea of these programs is to build equations from first principles (i.e. physical insight). In general terms, these programs have the same goal as The Modeller and are closely related to this work: (partially) automating and thereby speeding up the construction of dynamic process models for physical/chemical processes. A few general comments on observations, differences and shortcomings of the other tools and projects that I have read about are in order:

  - They all have some kind of automatic balance equation generation, but a clear link to the algebraic equations that also need to be present in order to fully describe the process is not made.
  - It is only allowed to construct "true" index-1 dynamic models. Assumptions that lead to higher index process models are not supported.
  - It is, in general, not clear what kind of assumptions a modeller can make and what the implications of the assumptions are.
  - Steady-state assumptions on parts of the dynamic model can clear up certain modelling difficulties. Such assumptions cannot be made by the existing modelling programs.
  - Mostly, the aim of the projects was to build an automated model derivation program (which was, in general, not limited to building only dynamic process models). The emphasis was often on the implementation and therefore the modelling approaches that lay behind these programs, in general, do not reach a high level of abstraction (vector and matrix algebra are almost never used, but can certainly make modelling algorithms more transparent). This often makes the implementation (and the discussion) more complex.

  Examples of automated model derivation programs are: Model.la (Bieszczad 2000), ModDev (Jensen 1998), ModKit (Lohmann 1998), PROFIT (Telnes 1992), KBMoSS (Vázquez-Román *et al.* 1996).

1.5 Scope of this Project

The scope of this work is limited to dynamic process models for physical, chemical and/or biological processes that are based on physical insight. The

processes are interpreted networks of interconnected lumped systems. Distributed parameter systems are not considered, since this type of systems can be approximated by a large number of interconnected lumped systems[2].

This work concerns models for physical, chemical and/or biological processes and since for such processes momentum and electric current are usually of minor importance, this work is concentrated on mass and energy. The discussed framework is, though, of a more general nature, such that the now existing program can readily be extended to accomodate any other fundamental extensive quantity, such as momentum and electrical charge.

In my opinion, the majority of dynamic models is build for the purpose of solving initial value problems (i.e. simulations) and only a small part for other purposes, such as dynamic design, optimal control and parameter estimation. Although models, as derived with our methodology can, in principle, be used for any purpose (simulation, design, optimal control, parameter estimation), the main emphasis of this work is on the construction of models (consisting of differential algebraic equations) that are suitable for the solution of initial value problems.

In short, the main questions that I try to answer in this thesis are the following:

1. Can I 'prove' that modelling of dynamic processes is more of a science than an art, such that it can be used by a much larger group of engineers than is the case today? I.e. is there more structure in the modelling process than is generally thought?

2. Is it possible to (seriously) speed up the whole modelling process?

3. To what extent can the modelling process be captured in a computer program and under which conditions?

1.6 Outline and Intention of the Thesis

This thesis describes a systematic modelling approach for the development of first principles based, dynamic process models for macroscopic chemical, physical and/or biological processes. It thesis consists of three main parts.

The first part, Modelling Methodology, describes the systematic modelling approach that was partially developed during my research period. In

[2]This is just one way of approximating distributed systems. There are also other ways, but these fall beyond the scope of this thesis.

chapter 3 the basic building blocks for constructing physical topologies are introduced and an explanation on how to construct physical topologies is given. Chapter 4 introduces the so-called species topology and some species related concepts such as permeability, directionality and reactions. In chapter 5 the information of the physical and species topologies is combined to generate the balances of relevant fundamental extensive quantities. Also, the algebraic relations a model designer has to supply, in order to complete the model, are discussed. The last chapter of the first part, chapter 6, discusses the implications of introducing order of magnitude assumptions into the model and gives some solutions to the thereby imposed problems.

The second part, Implementation, runs parallel to the first part and discusses the implementation of the topologies into the computertool the Modeller. Chapter 8 describes the construction and manipulation of physical topologies. Chapter 9 discusses the construction and manipulation of species topologies. The implementation and handling of equations and variables is discussed in chapter 10.

In the last part, Examples and Conclusion, a number of examples are worked out with the intention to show the reader that first principle modelling does not have to be difficult - something that is generally assumed - but can actually be fairly simple if our structural approach is used. The final chapter, chapter 12, contains the main conclusions, summarises the main contributions, and gives some suggestions for further research.

The context of this project is modelling of physical/chemical/biological processes, including any type of production facility (chemicals, food, biotech) and natural process. The project aimes at systematising modelling of the above mentioned type of processes with the objective to implement the resulting methodologies into a computing environment. This will primarily increase the efficiency of model building, maintenance and model derivations.

The concepts presented in this thesis are applicable to a large group of "modellers". Therefore, an important personal goal was to make this thesis as "readable" as possible, such that a large percentage of the people who read this book can actually understand the presented material and even apply it, if necessary. This means that I have tried to explain all the concepts in plain English, giving simple supporting examples and minimising the needed mathematics and formalisms. A basic understanding of differential mathematics and vector and matrix algebra is required though.

Part I

Modelling Methodology

Chapter 2

Introduction to Modelling

Process systems engineering spans the range between process design, operations and control. Whilst experiments are essential, modelling is one of the most important activities in process engineering, since it provides insight into the behaviour of the process(es) being studied. Modelling is a complex activity, as the information of appropriate and consistent models requires help from experts and often results in a knowledge-intensive, time consuming and error-prone task (Vázquez-Román *et al.* 1996)[1].

2.1 Three Principal Problems

The first step in identifying the various characteristic steps of process systems engineering problem solving is to identify a minimal set of generic problems that are being solved. Most problems in this area have three major components, which are :

- **Model** :: A realisation of the modelled system simulating the behaviour of the modelled system.
- **Data** :: Instantiated variables of the model. May be parameters that were defined or time records of input or output data obtained from an actual plant, marketing data etc.
- **Criterion** :: An objective function that provides a measure and which, for example, is optimised giving meaning to the term *best* in a set of choices.

[1]The material presented in this chapter was partially taken from an internal report (Preisig 1994*a*).

The particular nature of the problem then depends on which of these components is known and which is to be identified. Each type of problem is associated with a name which changes from discipline to discipline. The choice of the names listed below was motivated by the relative spread of the respective term in the communities. The following principle problems are defined (Figure 2.1):

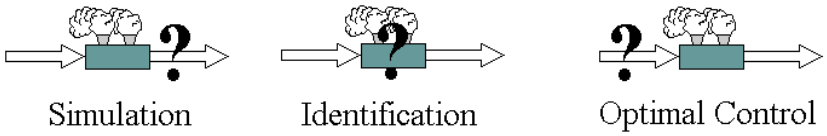

Figure 2.1: *The three base problems.*

- **Simulation**: Given model, given input data and given parameters find the output of the process.

- **Identification**: Given model structure, some times several structures, given process input and output data, and a given criterion, find the best structure and the parameters for the parameterised model, where "best" is measured with the criterion.

- **Optimal Control**: Given a plant model, a criterion associated with process input and process output, and the process characteristics, find the best input subject to the criterion.

The definition of **simulation** is straightforward. The task of **identification** though is not, in that it includes finding structure as well as parameters of a model. This in turn implies that many tasks match this description, such as process design and synthesis, controller design and synthesis, parameter identification, system identification, controller tuning and others fit this definition. The definition of the **optimal control** task is also wider than one usually would project. Namely process scheduling and planning are part of this definition as well as the design of a shut-down sequence in a sequential control problem, to mention a few non-traditional members of this class.

In all three classes a model is involved. In this discussion parameterised input/output mapping are used because they are usually the type of model capturing the behaviour of a given system in the most condensed form. Though in each case it is solved for a different set of its components. In the case of the simulation, the outputs are being computed, in the case of the identification task, the best parameters are found and in the case of optimal control, the best input record is being determined.

In order to solve a problem, the model must be supplemented with a set of definitions, which, in combination with the model, define a mathematical problem that can be solved by a particular mathematical method. These definitions are instantiations of variables that assign known quantities to variables or functions of know quantities, where the functions may be arbitrarily nested.

On this highest level, process engineering problem solving has four principle components :

1. Formulation of a model;
2. Problem specification;
3. Problem solution method;
4. Problem analysis.

The following sections discuss the problem of the model formulation in more details.

2.2 The Modelling Process

Models take a central position in all process engineering tasks as they replace the process for the analysis. They represent an abstraction of the process, though not a complete reproduction. Models make it possible to study the behaviour of a process within the domain of common characteristics of the model and the modelled process without affecting the original process. It is thus the common part, the homomorphism or the analogy between the process and model and sometimes also the homomorphism between the different relations (== theories) mapping the process into different models that are of interest. The mapping of the process into a model does not only depend on the chosen theory, but also on the conditions under which the process is being viewed. The mapped characteristics vary thus not only with the applied theory but also with the conditions.

Different tasks focus on different characteristics and require different levels of information about these characteristics. For example control would usually be achievable with very simple dynamic models, whilst the design of a reactor often requires very detailed information about this particular part of the process (Aris 1978). The result is not a single, all-encompassing model but a whole family of models. In fact, there is no such thing as a unique and complete model, certainly not from the philosophical point of view nor from a practical, as it simply reflects the unavoidable inability to accumulate complete knowledge about the complex behaviour of a real-world system. More practically and pragmatically a process is viewed as the representation of the

essential aspects of a system, namely as an object, which represents the process in a form *useable for the defined task*. Caution is advised, though, as the term *essential* is subjective and may vary a great deal with people and application.

The term **multi-faceted modelling** has been coined reflecting the fact that one deals in general with a whole family of models rather than with a single model (Zeigler 1984). Whilst certainly the above motivation is mainly responsible for the multi-faceted approach, solution methods also have use for a family of models as they can benefit from increasing the level of details in the model as the solution converges.

An integrated environment must support multi-faceted modelling, that is mapping of the process into various process models, each reflecting different characteristics or the same characteristics though with different degree of sophistication and consequent information contents. In order to identify components of the suggested environment, the process of deriving process models is the subject of the next section.

2.3 Formulation of the Model

Establishing a model, as it is used in a particular application, involves a number of operations, which can be broken down into a number of principle steps. Analysing the operations on each individual step then reveals the communalities and differences. The object being mapped is referred to as the plant which is here used as a synonym for physical-chemical-biological system, an entity which transforms matter by the means of a chemical or biological process in a physical environment.

Alternative models can be generated in one of two ways. The **first approach** is to use different theories to map the plant into a mathematical object thus obtain different mathematical models using different theories. The **second possibility** is to manipulate one of the models deriving new models. Again, an integrated environment should allow for both mechanisms.

Any manipulation will in general reduce, or at the best maintain, the information contents of the model with respect to structural complexity. The manipulation can only result in transformed models or simplified models.

2.3.1 Primary Modelling Operation

The very first step in the process of obtaining a model is the mapping of the real-world prototype, the plant, into a mathematical object, here called the primary model. This process is non-trivial because it involves structuring of the process into components and the application of a mapping theory for each component. Since the theories are context dependent, the structuring is tightly

coupled to the theory chosen. The process of breaking the plant down to basic thermodynamic systems determines largely the level of details included in the model. It is consequently also one of the main factors for determining the accuracy of the description the model provides.

Chapter 3 gives an idea on how a process can be broken down into smaller parts using only two basic building blocks, namely systems and connections. The resulting interconnected network of capacities is called the *physical topology*.

In chapter 4 the so-called *species topology* is introduced. This species topology is superimposed on the physical topology and defines which species and what reactions are present in each part of the physical topology.

Structuring is one factor determining the complexity of the model. Another factor comes to play when choosing descriptions for the various mechanisms such as extensive quantity transfer and conversion. For example in a distributed system, mass exchange with an adjacent phase may be modelled using a boundary layer theory or alternatively a surface renewal theory, to mention just two of many alternatives. The person modelling the process thus will have to make a choice. Since different theories result in different models, this implies making a choice between different models for the particular component. Structuring and use of theories will always imply *intrinsic simplifying assumptions*. For example modelling a heat transfer stream between the contents of the jacket and the contents of a stirred tank may be modelled using an overall heat transfer model, that is a pseudo-steady state assumption about the separating wall and the thermal transfer resistant of the fluids is made. Though, if one is interested in the dynamics of the wall, a simple lumping of the wall improves the description or if this is still not sufficient one may choose to describe the heat transfer using a series of distributed coupled system or a series of coupled lumped systems.

Assuming for the moment that the structuring is not causing any problems and assuming that a theory or theories are available for each of the components, the mapping into a mathematical object associated with a chosen view and an associated theory is formal.

The presumption is made that no other intrinsic assumptions are being made at this point, that is, the theory is applied in its purest form. Specifically, the conservation principles, which describe the basic behaviour of plants, are applied in their most basic form. Particularly the energy balance is formulated in terms of total energy and not in any derived state functions.

Chapter 5 discusses the mathematical description of process models (the *equation topology*) and deals with the last three steps of the modelling method (outlined in the first chapter of this thesis), since all of these steps involve equations. This chapter is devoted to "proper model building", so assumptions that implicate modelling difficulties are kept clear of.

2.3.2 Assumptions about the Nature of the Process

Once a mathematical model for the process components has been established, usually the next operation is to implement a set of assumptions which eliminate complete terms or parts thereof in the equations of the primary model. These are assumptions about the principle nature of the process such as no reaction, no net kinetic or potential energy gain or loss, no accumulation of kinetic or potential energy. Whilst not complete, this is a very typical set of assumptions applied on this level in the modelling process. The assumptions are of the form of definitions, that is variables, representing the terms to be simplified are instantiated. For example, the production term in the component mass balances might be set to zero.

Additional assumptions that simplify the process model may be introduced at any point in time. A very common simplification is the introduction of a pseudo-steady state assumption for a part of the process which essentially zeroes out accumulation terms and transmutes a differential equation into algebraic relation between variables. These can then be used to simplify the model by substitution and algebraic manipulation.

The final chapter of this part, chapter 6, discusses which alterations have to be made to the equation topology when certain assumptions are being made.

Chapter 3

Hierarchical Organisation: The Physical Topology

In order to get a mathematical description of a process, a modeller usually has to break the process down into smaller parts for convenience reasons. A process is thus assumed to consist of a set of subprocesses. These subprocesses may in turn be divided into smaller processes, and so on, until the process consists of a number of subprocesses, which each are small enough to be handled individually. In the modelling method that we employ this means that a process is divided into a network of interconnected volume elements. Each volume element of such a network consists of a single phase that is uniformly distributed and hence displays uniform properties over its volume. The network of volume elements describes, so-to-speak, the physical structure of the process and shall be referred to as the *physical topology* of that process.

The physical topology contains, once established, the maximum of information about the dynamic phenomena captured in the model. Any modification on the topology changes the dynamic information contents.

3.1 Building Blocks

The physical topology is built from only two principle components, namely *systems* and *connections*. The systems (volume elements) represent capacitive elements able to store extensive quantities (such as mass, energy and momentum) and are therefore associated with mass. The connections describe the exchange of extensive quantities between two such capacities. Hence, systems and connections appear in an alternating sequence in a process representation.

As mentioned before, a process can be decomposed into a set of interactive thermodynamic systems based on classical axiomatic thermodynamics.

A thermodynamic system is defined (Tisza 1961) as a finite region in space specified by a set of physically quantifiable variables. Such a system consists of a *phase*. The systems concept, though, does not require uniform conditions within this phase. So, in order to define a consistent and generic primitive modelling component - for the sake of basic and fundamental modelling - we have to slightly adjust this definition.

One can distinguish between two "types" of systems. The first are lumped systems, which have a homogeneous or uniform phase and consequently have uniform intensive properties (such as concentration, temperature and density). The others are distributed systems, which do not exhibit uniform conditions across their phase: the intensive properties are gradually changing with the defined spatial coordinates. These two types can be mapped into one by defining only one primitive simple system that may be of finite or differentially small volume. This results in the following definition:

DEFINITION 3.1.1 *A* **simple system** *(Σ) is a body of finite or differentially small volume consisting of a single phase or a pseudo-phase*

This definition implies that a capacity of any nature is present in the system and that the state of the system is uniform within the spatial domain it occupies. In the definition the system is called a simple system because it relates to a single uniform phase or a *pseudo-phase* within the system. The phase concept is hereby extended by allowing for a *pseudo-phase*. This is done in order to support the granularity concept. A pseudo-phase is a spatial domain with spatially averaged properties and is associated with the granularity of the model. On a finer scale, the phase concept applies usually well, but as one increases the granularity, i.e. larger spatial domains are represented as systems, the systems spatial domain may not exhibit uniform physical properties any more. However, for the purpose of the model, one may still want to view this domain as a uniform system. For this reason the pseudo-phase concept is introduced.

A system represents a capacity and is separated from its neigbouring systems by means of an imaginary wall of no mass and therefore of no capacity. This *boundary*, which encapsulates the total mass of the system, is a theoretical artifact. Although these boundaries often replace physical walls in a model, they are certainly not the same. Mapping a physical wall into a boundary of a thermodynamic system implies that the capacity effect of the wall is neglected and that it is mapped into an abstract thermodynamic wall. If one wishes to model a wall as a capacitive component, this wall should be represented as a system, and thus be associated with a capacity.

Systems are interacting through their common boundaries. Classical ther-

modynamics defines three classes of systems based on the nature of these interactions, namely isolated, closed and open systems. A thermodynamically isolated system has no interaction with its environment. The interaction of a closed system with its environment is limited to the transfer of heat only, without any exchange of matter. A thermodynamically open system can interact with its environment through its boundary transferring both energy and mass.

The interactions between systems are represented by the second type of building block used to construct the physical topology, namely the *connections*. A *connection* represents an interaction between two capacitive elements c.q. systems. It describes the exchange of extensive quantities across the boundary separating two adjacent systems. Therefore, a connection cannot exist in isolation. In other words, a connection cannot be established prior to the definition of two distinct systems as origin and target of the connection. The implication of this paradigm is that all plant models are closed. No open ends exist. Any source of material, energy, or any extensive quantity for that matter must be modelled and the same applies for any sink.

A connection represents a common boundary segment of the connected systems. An arbitrary number of connections may exist between two systems through any of the common boundary elements. The connections, at this stage, do not specify yet the specific details on how and how much is being communicated through them, only the nature of the connections is defined. They can either be mass, heat or work connections.

DEFINITION 3.1.2 *A* **connection** *is a directed communication path for the exchange of a specific extensive quantity between two systems through a boundary element, separating the two systems*

The definition given above implies that connections are directed. This means that the flow of the extensive quantity through the connection is measured relative to a reference co-ordinate system. This co-ordinate system is defined by the origin and the target of the connection (this is graphically depicted as an arrow with a head and a tail).

The actual flow is caused by a difference in the states of the two systems, such as heat flow by a difference in the temperature and convective flow by a difference in the pressure. By convention, the sign of the flow is considered to be positive when the transferred quantity flows from the origin to the target.

Example 3.1: ***A General Process.***

Figure 3.1a represents a general open process, embedded in its environment. The process divided into 3 parts, each of which can

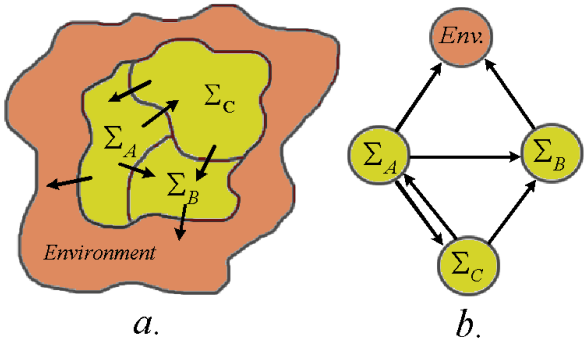

Figure 3.1: a. *A general open process and its interaction with its environment,* b. *Possible physical topology of this process.*

be modelled as simple thermodynamic systems. Each arrow in this drawing represents the transfer of a specific extensive quantity across the boundary of two adjacent systems. The process can be modelled as a closed process if the environment is considered as a simple system. The interactions between the systems and the environment can then be modelled in the same way as the interactions between the individual systems.

Figure 3.1b shows a physical topology of the process. As you can see, the physical topology is built from (lumped) capacities, shown as circles, and directed connections, shown as arrows. Each connection has an origin and a target system. The origin and the target define the reference co-ordinate system for the direction of the flow of extensive quantity between them. So, if the transferred quantity flows from the target to the origin, this will correspond to a negative flow from the origin to the target.

□

Example 3.2 shows a possible division of a process into smaller parts, which results in a possible physical topology of that process. The physical topology consists only of systems and connections between those systems.

Example 3.2: ***A Possible Physical Topology***

Figure 3.2a shows a schematic representation of a reactor that could be part of a larger plant. The reactor is divided in four parts by three partitions and the contents of the reactor is heated via the surface of the reactor. The actual contents of the reactor and the reaction(s) taking place in it are of no concern at this first stage

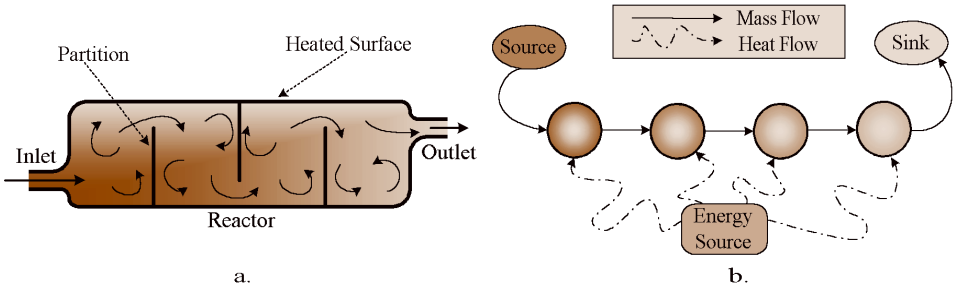

Figure 3.2: a. *Schematic representation of an example reactor, b. Possible physical topology for this reactor.*

of the model construction. Presently, only a physical topology for this reactor shall be constructed. This topology will form the basis for the rest of the modelling process. A possible division of the reactor into simple systems is shown in figure 3.2b. The circles and the square represent capacitive volume elements (simple lumped systems), the arrows represent paths for the exchange of extensive quantities (connections).

The reactants are fed into the reactor at the top end, represented by the first lump. A source element was introduced here in order to model the inflow. It is important to realise that if the reactor would be modelled within a larger network, the source might be replaced by another physical element. The source system has to be introduced here, because we have to construct closed process models with our modelling methodology. This means that mass must come from somewhere; it cannot just appear. Then the fluid passes from one lump to another, until it leaves the reactor at the tail end. The products are collected in a sink system, again because mass cannot just disappear. The heat that is added to each lump is represented by a connection from an energy source, which describes the energy transfer.

Obviously this physical topology would result in a relatively simple model of the reactor. The model maps the four parts of the reactor into four individual lumped systems or volume elements. It is implied that within one such volume element the physical conditions are spatially uniform. Of course, this is but an approximation of the physical reality. Also, the representation of the heating surface of the reactor as a simple uniform capacity is an approximation. With the equations that describe the transfer of mass and energy

and with the mass and energy balances of each system, we can describe the dynamic and/or static behaviour of the process, provided that enough additional information is added.

□

The two examples illustrate how a physical topology of a process can be constructed with just two building blocks, systems and connections.

3.2 Domains

Connections can be either mass, heat or work connections and therefore a process model can consist of several domains. In this work a domain has the following definition:

DEFINITION 3.2.1 *A <typed> **domain** is an interconnected network of capacities (i.e. primitive systems), in which all connections are of the same type <type>.*

As a rule, single systems, i.e. systems without connections, do not belong to any specific domain. An exception to this rule is a system, which has a capacity to store certain species (see chapter 4). Such a system is a (single) mass domain.

We distinguish 4 types of domains, namely: mass, heat, work and energy domains. In energy domains all types of connections are considered. A mass domain can hold several species domains, depending on the presence of those species in the systems and connections of the mass domain.

Example 3.3: *Domains Within a Physical Topology*

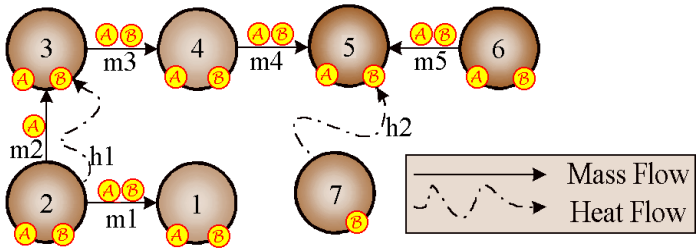

Figure 3.3: *Physical topology of a process consisting of 8 systems, 6 mass connections and 2 heat connections.*

Consider the physical topology shown in figure 3.3. This physical topology consists of 7 systems, 5 mass connections and 2 heat

connections. There are 2 species present: species A is present in systems 1, 2, 3, 4, 5 and 6 and flows through all mass connections. Species B is present in the systems 1, 2, 3, 4, 5, 6 and 7 and flows through the mass connections m1, m3, m4 and m5.

We can identify the following domains:

- *2 Mass domains: (1, 2, 3, 4, 5, 6, m1, m2, m3, m4, m5) and (7).*
- *2 Heat domains: (2, 3, h1) and (5, 7, h2).*
- *1 Energy domain: (1, 2, 3, 4, 5, 6, 7, m1, m2, m3, m4, m5, h1, h2).*
- *1 Species domain for species A: (1, 2, 3, 4, 5, 6, m1, m2, m3, m4, m5).*
- *3 Species domains for species B: (1, 2, m1), (3, 4, 5, 6, m3, m4, m5) and (7).*

□

3.3 Fundamental Time Scale Assumptions

Physical topologies are the abstract representation of the containment of the process in the physical sense. They visualise the principle dynamic contents of the process model and therefore the construction of a physical topology is the most fundamental part of the modelling process. Any changes in the physical topology will substantially affect the structure and contents of the final model.

The structuring of the process implements the first set of assumptions in the modelling process. The resulting decomposition is, in general, not unique. However, the resulting model depends strongly on the choice of the decomposition. As a rule: the finer the decomposition, the more complex will the resulting model be.

The decision of defining subsystems is largely based on the phase argument, where the phase boundary separates two systems. The second decision criterion utilises the relative size of the capacities and the third argument is based on the relative velocity with which subsystems exchange extensive quantities. Another argument is the location of activities such as reactions. The relative size of the model components, measured in ratios of transfer resistance and capacities, termed time constants, is referred to as "granularity". A large granularity describes a system cruder, which implies more simply, than a model with a finer granular structure. It seems apparent that one usually aims at a relative uniform granularity, as these systems are best balanced and thus

more conveniently solved by numerical methods. Models of different granularity help in analysing the behaviour of the process in different time scales. The finer the granularity, the more the dynamic details and thus the more of the faster effects are captured in the process description.

Since each model is constructed for a specific goal, a process model should reflect the physical reality as accurately *as needed*. The accuracy of a model intended for numerical simulation, for example, should (in most cases) be higher than the accuracy of a model intended for control design.

To illustrate the concepts mentioned above, consider the following example concerning a stirred tank reactor:

Example 3.4: ***Stirred Tank Reactor***

Figure 3.4a shows a stirred tank reactor, which consists of an inlet and outlet flow, a mixing element, a heating element and liquid contents. If the model of this tank is to be used for a rough approximation of the concentration of a specific component in the outlet flow or for the liquid-level control of the tank, a simple model suffices. The easiest way to model this tank is to view it as an ideally stirred tank reactor (ISTR) as shown in figure 3.4b. This implies that a number of assumptions have been made regarding the behaviour and properties of the tank. The most important assumption is the assumption that the contents of the tank is ideally mixed and hence displays uniform conditions over its volume. Another assumption can be that heat losses to the environment are negligible.

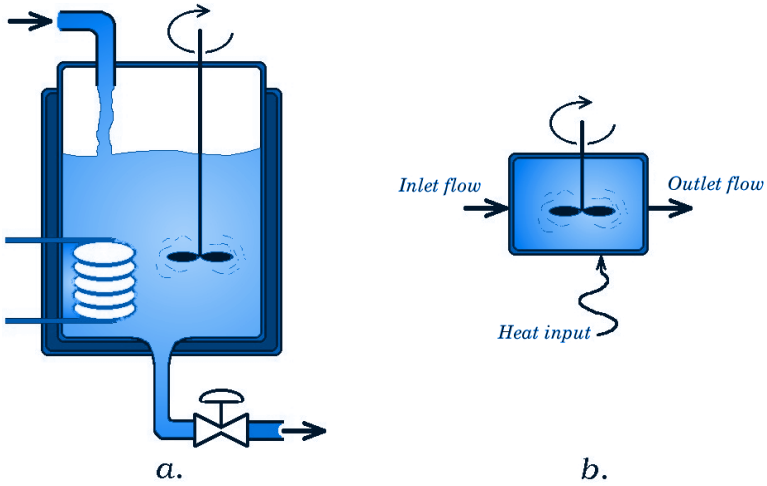

Figure 3.4: *a. Stirred tank reactor; b. Ideally Stirred Tank Reactor (ISTR).*

After making these and maybe some more assumptions, the modeller can write the total mass balance, component mass balances and the energy balance of the reactor. With these equations and some additional information (e.g. kinetics of reaction, physical properties of the contents, geometrical relations, state variable transformations, etc.) the modeller can describe the dynamic and/or static behaviour of the reactor.

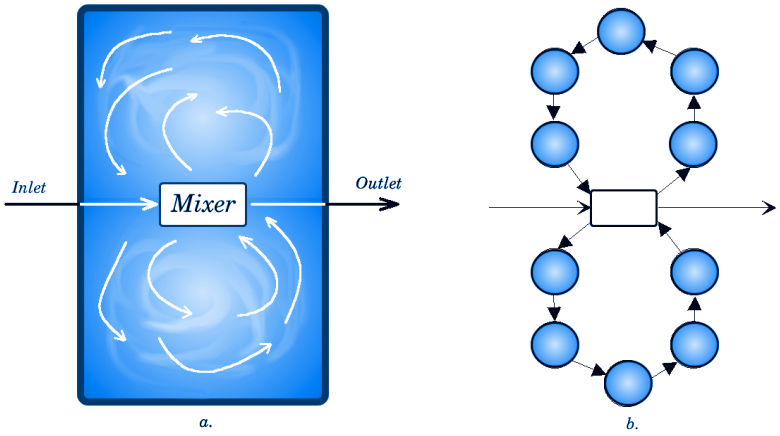

Figure 3.5: a. *Graphical representation of a possible mixing in a tank; b. Possible decomposition of this process into volume elements for the construction of a mathematical model.*

If the tank has to be described on a much smaller time-scale and/or the behaviour of the tank has to be described in more detail, then the ISTR model will not suffice. A more accurate description often asks for a more detailed model. In order to get a more detailed description the modeller could, for example, choose to try to describe the mixing process in the tank (see figure 3.5a). Figure 3.5b shows a possible division of the contents of the tank into smaller parts. In this drawing a circle represents a volume element which consists of a phase with uniform conditions. Each volume element can thus be viewed as an ISTR. The arrows represent the mass flows from one volume to another. In order to describe the behaviour of the whole tank, the balances of the fundamental extensive quantities (component mass, energy and work usually suffice) must be established for each volume element. The set of these equations, supplemented with information on the extensive quantity transfer between the volumes and other additional information, will constitute the

mathematical description of the dynamic and/or static behaviour of the reactor.

The model of the mixing process could, of course, be further extended to get a more accurate description[1]*. The number of volume elements could for example be increased or one could consider back mixing or "cross" mixing of the liquid between the various volume elements. The conduction of heat to each volume could also be modelled. One could model a heat flow from a heating element to each volume, or only to those volume elements which are presumed to be the nearest to the heating element, etc. As one may imagine there are many ways to describe the same process. Each way usually results in a unique mathematical representation of the behaviour of the process, depending on the designers view on and knowledge of the process, on the amount of detail he wishes to employ in the description of the process and, of course, on the application of the model.*

□

When employing the term time scale, we use it in the context of splitting the relative dynamics of a process or a signal (the result of a process) into three parts, namely a central interval where the process or signal shows a dynamic behaviour. This dynamic window is on one side guarded by the part of the process that is too slow to be considered in the dynamic description, thus is assumed constant. On the other side the dynamic window is attaching to the sub-processes that occur so fast that they are abstracted as event - they just occur in an instant. Any process we consider requires these assumptions and it is the choice of the dynamic window that determines largely the fidelity of the model in terms of imaging the process dynamics.

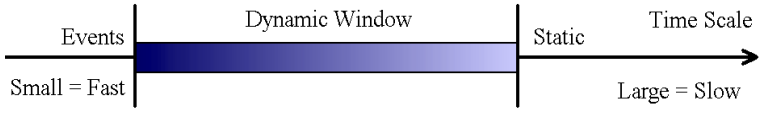

Figure 3.6: *The dynamic window in the time scale.*

One may argue that one should then simply make the dynamic window as large as probably possible to avoid any problems, which implies an increase

[1]In principle, if one increases the complexity of this description, one approaches the type of models that result from approximating distributed models using computational fluid dynamic packages.

in complexity. Philosophically all parts of the universe are coupled, but the ultimate model is not achievable. When modelling, a person must thus make choices and place focal points, both in space as well as in time. The purpose for which the model is being generated is thus always controlling the generation of the model (Aris 1978)(Apostel 1960) and the modeller, being the person establishing the model, is well advised to formulate the purpose for which the model is generated as explicit as possible.

A window in the time scales must thus be picked with the limits being zero and infinity. On the small time scale one will ultimately enter the zone where the granularity of matter and energy comes to bear, which limits the applicability of macroscopic system theory and at the large end, things get quite quickly infeasible as well, if one extends the scales by order of magnitudes. Whilst this may be discouraging, having to make a choice is usually not really imposing any serious constraints, at least not on the large scale. Modelling the movement of tectonic plates or the material exchange in rocks asks certainly for a different time scale than modelling an explosion, for example. There are cases, where one touches the limits of the lower scale, that is, when the particulate nature of matter becomes apparent. In most cases, however, a model is used for a range of applications that usually also define the relevant time-scale window.

The dynamics of the process is excited either by external effects, which in general are constraint to a particular time-scale window or by internal dynamics resulting from an initial imbalance or internal transposition of extensive quantity. Again, these dynamics are usually also constraint to a time-scale window. The maximum dynamic window is thus the extremes of the two kinds of windows, that is, the external dynamics and the internal dynamics. A "good" or "balanced" model is in balance with its own time scales and the time-scale within which its environment operates. In a *balanced model*, the scales are adjusted to match the dynamics of the individual parts of the process model. Balancing starts with analysing the dynamics of the excitations acting on the process and the decision on what aspects of the process dynamics are of relevance for the conceived application of the model. What has been defined as a system (capacity) before may be converted into a connection later and vice versa as part of the balancing process. This makes it difficult, not to say impossible, to define systems and connections totally hard. The situation clearly calls for a compromise, which, in turn, opens the door for suggesting alternative compromises. There is not a single correct choice and there is certainly room for arguments but also for confusion. Nevertheless a decision must be taken. Initially one is tempted to classify systems based on their unique property of representing capacitive behaviour of volumes, usu-

ally also implying mass. In a second step one may allow for an abstraction of volumes to surfaces, because in some applications it is convenient to also abstract the length scale describing accumulation to occur inside, so-to-speak, or on each side of the surface.

For the sake of modelling, some of the defined systems within a physical topology can be defined as "non-dynamical systems", because the dynamics of these systems is either outside the dynamic window, or is not of interest for the model (for example, the source and sink system in example 3.1). The systems can be divided into six "sub-classes":

- Lumped
- Distributed
- Source
- Sink
- Steady State
- Composite

In this work we primarily use the *lumped systems* to describe the dynamics of a process. *Distributed systems* are systems that do not display strictly uniform intensive properties (such as temperature, pressure, composition, density, etc.) within their boundaries. They are a common phenomenon within the modelling community, but, because of their infinite order, one often does not like to have them in the model. One way to remedy this is to let the distributed system undergo a model reduction, thereby mapping them from an infinite-order into a finite-order system. With this it is accepted that the higher frequency properties of the process are not appropriately reflected in the model. The results are models that apply to the intermediate time scale.

Source and *sink systems* represent resource systems, the dynamics of which are not of interest for the model. These systems are usually infinitely large and only the intensive properties are defined, since the dynamics (which involve the extensive quantities) of these systems are ignored. Another property of source and sink systems is that the values of the defined intensive properties can be altered infinitely fast.

When the dynamics of a system are very fast as compared to the dynamics of its surrounding systems, a pseudo steady-state assumption is made for this system. This means that the dynamics of a *steady-state system* are ignored (this will be further explained in section 6.4).

Composite systems are introduced to address groups of systems, such that large and complex processes can be handled easier. The next section discusses the hierarchical organisation of physical topologies.

3.4 Hierarchical Organisation

A processing plant has a physical structure and this structure is represented by the physical topology of the plants symbolic model. The physical topology is the basic frame of the model. It represents the modellers view of the plants physical structure: It describes the containment of and the interactions between the different internal structural elements. Once constructed it determines the model order and with it an important portion of the dynamics. The physical topology of a processing plant is the foundation for the process information, which is added in the next two stages of the modelling process.

The physical topology is constructed by defining the components of the processing plant in the form of simple thermodynamic systems, which will eventually describe the physical behavior of a spatial domain, occupied by the plant. At this first stage of the model definition process, though, these physical primitive systems can be seen as container-like items that have not yet been filled nor further specified.

In the previous examples, the number of primitive components (simple systems) of the physical topology was very small. These processes could easily be represented as a network of primitive systems and connections. For larger and/or more complex process models, this flat topology would obviously not be very suitable. The number of systems does not need to grow very large before it becomes rather difficult to maintain an overview over the defined model of the process. The design, documentation and manipulation of these models would become very hard. Therefore, an abstraction has to be introduced in the form of a hierarchical definition of the model. This additional structure is laid over the flat topology in such a way that it allows to address groups of components c.q. composite systems. The result is a tree of systems, with the modelled process as the root node.

There exist basically two techniques for developing a physical topology of a process, namely the top-down and the bottom-up approach.

A modeller, who uses the top-down approach, starts with a simple initial model, which is based on a crude design. This crude model is then refined by adding details in several stages. In each stage the resolution of the model is increased by refining an element.

In the bottom-up approach, combinations of the simple systems define the next level of systems in the hierarchy (namely *composite systems*) and so forth

until finally on the top level the global system is defined, which of course represents the model of the process (the model of the process is a thermodynamic universe). This grouping of individual components is arbitrary, but obviously follows a logical scheme associated with physical location and functionality as well as a degree of interaction between the components.

When using the bottom-up approach, a modeller focuses on the reuse of existing model parts that have been already designed for other models. Clearly, all parts of a model may not have been defined previously. Thus, both methods are often used in the model building process. A good example to illustrate the hierarchical decomposition of a process is a distillation column.

Example 3.5: ***Hierarchical Decomposition of a Distillation Process***

A distillation column is a multi-staged column where components are separated due to differences in their vapor pressure. A typical distillation column consists of a column with a central feed, a reboiler to supply the necessary energy, and a condenser for removing most of the energy at the top (see figure 3.7b). The overhead vapor stream is cooled and completely condensed, and then it flows into the reflux drum. The liquid from the reflux drum is partly pumped back into the column (to the top tray) and the rest is removed as the distillate product.

At the base of the distillation column, the liquid product stream is split into two streams. The bottom product is drawn from the column, whilst the rest is recycled into the heater for evaporation and is returned to the base of the column.

The distillation column itself is considered to be composed of a feed tray and two sections, which here consist of three trays each.

Figure 3.7 shows a possible way to decompose the distillation process. This will finally lead to a flat topology of the process (figure 3.7e), which consists only of thermodynamic simple systems (circles and squares) and connections (arrows). For the construction of the physical topology as represented in fig 3.7e, the following assumptions have been made:

- *Each tray is assumed to consist of two homogeneous phases (a liquid phase L and a vapor phase V) and a 'boundary phase' B, separating the two homogeneous phases.*
- *The condenser and the reboiler can both be decomposed into three lumped systems and a steady-state (boundary) system.*

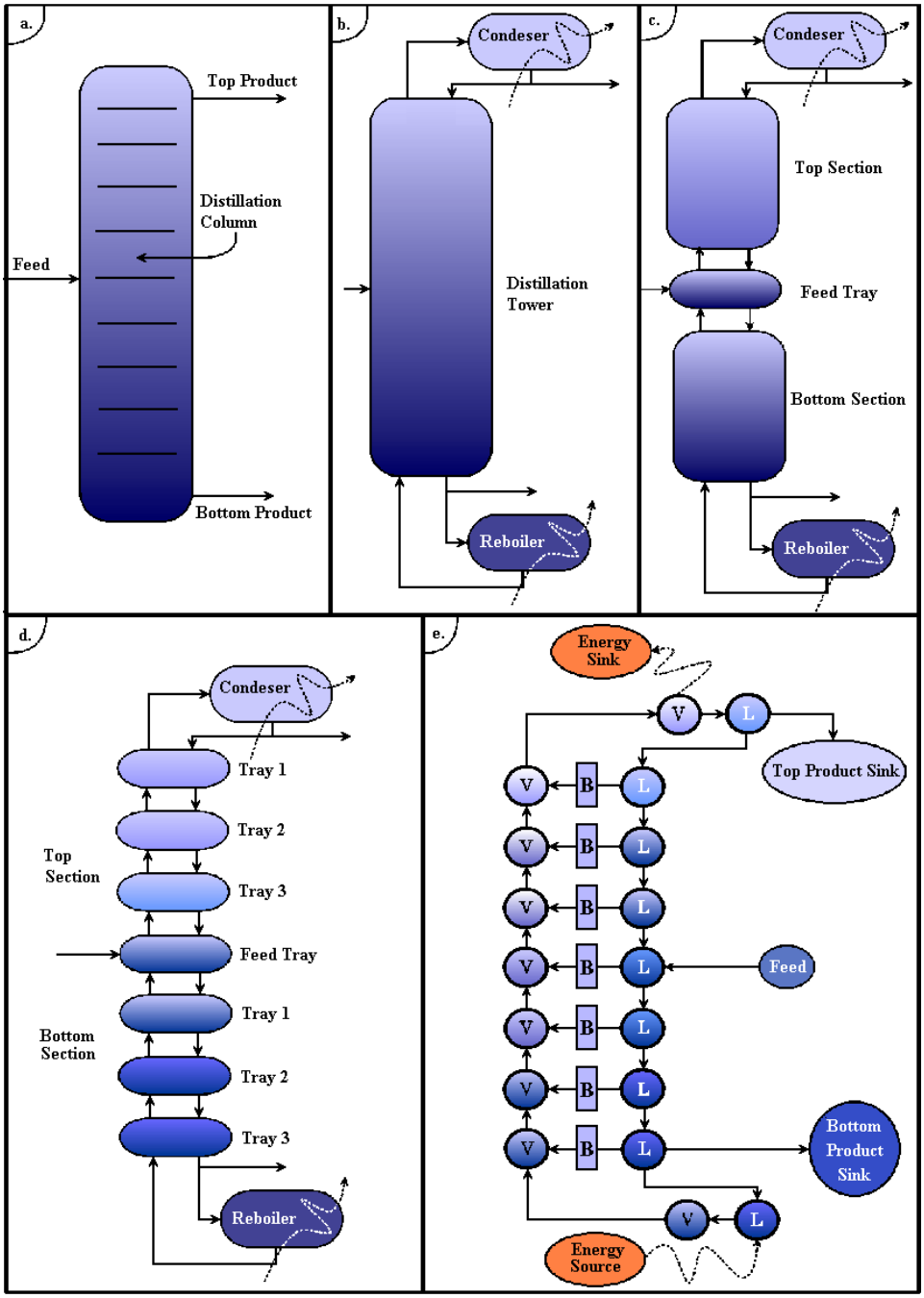

Figure 3.7: *Possible decomposition of a distillation process, in five steps.*

One lump as a heating/cooling medium and the other two as homogeneous process phases (liquid and vapor phase).

- *Heat losses from the column to the surroundings are assumed negligible.*
- *The momentum balances for each tray are neglected.*

The reference co-ordinate systems of the connections are chosen based on the normal flow direction of the streams. The mass streams connecting the liquid and the vapor phases on each stage are selected in such a way that the lighter components flows in the positive direction, and the heavier components in the negative direction.

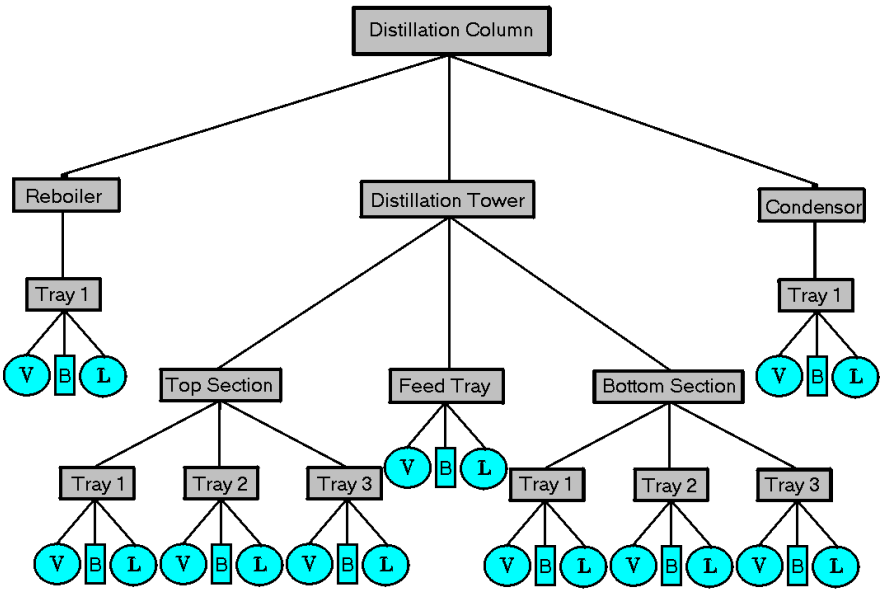

Figure 3.8: *Tree structure representation of a typical distillation column*

It appears natural to group the different systems and form a hierarchical tree for the representation of the column. Figure 3.8 shows such a hierarchical tree for the distillation process. For simplicity, the connections between the simple systems are left out in this figure, because only the tree representation of the physical topology shall be illustrated with this example. Also, the feed stream source and the product sinks, as well as the energy source and sink, are left out in this representation.

The tree is composed of five hierarchical levels. The root of the tree represents the highest hierarchical level, which of course is the

complete distillation column. The column is composed of the distillation tower, the reboiler and the condenser. These three components form the next hierarchical level. The distillation tower is again composed of three components, namely a feed tray, and a bottom and top section, which both consist of three trays. The two heat exchangers, the reboiler and the condenser, are considered to consist of one "tray". Each tray is assumed to be composed of two simple systems (a liquid phase and a vapor phase) and a steady-state system (representing the phase boundary). The leaves of the tree are simple systems, the other nodes are composite or composed systems.

In this analysis a top down point of view was taken. When constructing the tree from bottom up, one could start the description as follows: The liquid and vapor phase, which are both simple systems, are grouped to form a composite system, namely a tray. Trays in turn are grouped to form the bottom and top section of the distillation tower, etc.

Either way, the hierarchical structuring of a process will result in a tree of systems, the leaves of which are the thermodynamic simple systems which thus define the physical topology of the process.

□

Hierarchical grouping of the simple systems will always result in a hierarchical tree of systems with the modelled plant as the root node. The representation of the physical topology of a processing plant in the form of a tree is merely a matter of convenience. The handling of complex systems is supported by hierarchical arrangement of the model components. The hierarchical tree which is superimposed on the basic network, the physical topology, serves as a means to increase the level of abstraction. It increases the efficiency of handling the components and it facilitates the grasping of the essence of the whole process or parts of it (this will be illustrated in chapter 8). *The system, however, is completely defined on the lowest level, the leaves of the strictly hierarchical multi-way tree, which represent the simple thermodynamic systems.* The refinement "multi-way" relates to the fact that each subsystem may divide into an arbitrary number of subsystems. The tree is limited to a "strictly hierarchical" decomposition, because the conservation principles must hold for any part of the system. This means that none of the elements can be part of two subsystems at the same time; cross links between branches of the tree are thus prohibited. Why the employed tree structure has to be strictly hierarchical can be proved with the following simple proof:

Proof. *Proof for strictly hierarchical decomposition*
(Preisig, 1991c)

Let the overall system Σ, which represents the thermodynamic universe, be composed of the two subsystems Σ_1 and Σ_2. The system Σ_1 in turn consists of the subsystems Σ_{11}, Σ_{12} and Σ_{13}. System Σ_2 is composed of Σ_{13} and Σ_{21}. Figure 3.9 represents this tree structure. The leaves of the tree, i.e. the systems on the lowest level, which don't have any subsystems, represent the simple thermodynamic systems. The other nodes represent the composite systems. As illustrated, this tree is not strictly hierarchical because of the cross links between Σ_1 and Σ_2 to Σ_{13}.

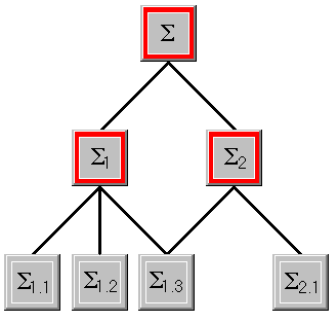

Figure 3.9: *A non-hierarchical tree.*

Denoting the mass balance of the overall system Σ as M_Σ, the following relation exists:

$$M_\Sigma = M_{\Sigma_1} + M_{\Sigma_2} \tag{3.1}$$

correspondingly, the mass balances for Σ_1 and Σ_2 are as follows:

$$M_{\Sigma_1} = M_{\Sigma_{11}} + M_{\Sigma_{12}} + M_{\Sigma_{13}} \tag{3.2}$$

$$M_{\Sigma_2} = M_{\Sigma_{13}} + M_{\Sigma_{21}} \tag{3.3}$$

substitution of the equations 3.2 and 3.3 in the overall mass balance equation 3.1 results in:

$$\begin{aligned} M_\Sigma &= (M_{\Sigma_{11}} + M_{\Sigma_{12}} + M_{\Sigma_{13}}) + (M_{\Sigma_{13}} + M_{\Sigma_{21}}) && (3.4)\\ &= M_{\Sigma_{11}} + M_{\Sigma_{12}} + 2M_{\Sigma_{13}} + M_{\Sigma_{21}} && (3.5) \end{aligned}$$

Thus, the subsystem Σ_{13} is counted twice, which contradicts the conservation principles. ■

Chapter 4

Chemical Distribution: The Species Topology

Processing plants comprise of various physical, chemical and/or biological processes in which their behaviour depends on the properties and the amounts of the involved mass components. Mass is a carrier of capacity and it forms the body or volume of the systems in the physical structure of the plant. Hence it is essential in the formulation of process models.

Knowledge about the involved species and reactions is often required for the behavioural description of a processing plant, because the behaviour of the plant is often species dependent. For example, the dynamics of reactions and the transfer of extensive quantities are characterised by kinetic laws and transfer laws, respectively. These laws, which are usually part of the fundamental models of processing plants, are defined as variables and constants, whose values (at least partially) depend on the physical properties of the involved species and the reactions between these species.

4.1 Chemical Distribution

The idea of the construction of the species topology is to facilitate the definition and identification of species and reactions occurring in various parts of the processing plant. The species topology defines which species are present in which part of the process and thus defines the basics of the chemistry/biology of a processing plant model. This enables the generation of the relevant balances of extensive quantities for the systems.

The species topology is superimposed on the physical topology. Its construction efficiently aids the model designer in the definition of the chemistry/biology of the processing plant because it generates knowledge about which species are present in every system in a physical topology and about

which reactions occur in every reactive system.

It should be noted that at this stage of the model definition process the quantities of the species in the systems are of no relevance. The association of a certain species with a certain system only implies that this system has a capacity for this particular species. If, for example, a system Σ is associated with only one species, S, then this implies that species S is the only species that can be present in system Σ. It does *not* imply that S has to be present in Σ. At certain moments or maybe all the time during an evolution/simulation of the process, species S may even be absent. At these particular moments the system has no mass, but of course still has a capacity to store the species S.

The existence of certain species and reactions in one part of the system does not imply that they are the same in all parts of the system. In principle, however, every chemical or biological species could be present in any part of the system, so one could decide not to construct the species topology and only generate a global set of species. This would mean that one defines only one set of species, which comprises all species that appear in the whole plant. Every primitive system of the physical topology would then consist of all these species. For each species in each system the species mass balances could then be generated, which would of course result in a flood of equations. In most cases, many of these equations would be obsolete, because species are usually constrained to a group of systems and therefore do not appear in every system. The species of the fluids in the two parts of a shell and tube heat exchanger, for example, are usually not the same.

To aid in the definition of the species topology, species and reaction databases are used and three concepts are introduced, "*permeability*", "*uni-directionality*" and "*reactions*", which support the idea of locality of chemical or biological species. These three concepts will facilitate the definition of the species topology and will be illustrated in the next sections.

The permeability and uni-directionality concepts are associated with connections and therefore with the common boundary element which the two connected systems have in common.

4.2 Permeability of Mass Connections

The permeability is defined as a property of a mass connection, which constrains the mass exchange between systems by making the species transfer selective. The mass connection is then defined as a semi-permeable thermodynamic wall. An example of a process where the permeability concept could come in handy is a simple diffusion process. The application of the selective

constraint on the transportation of species can also assist in the representation of separation processes, such as ultra-filtration or reverse osmosis.

Example 4.1: ***Diffusion of a Component from one Phase to Another***

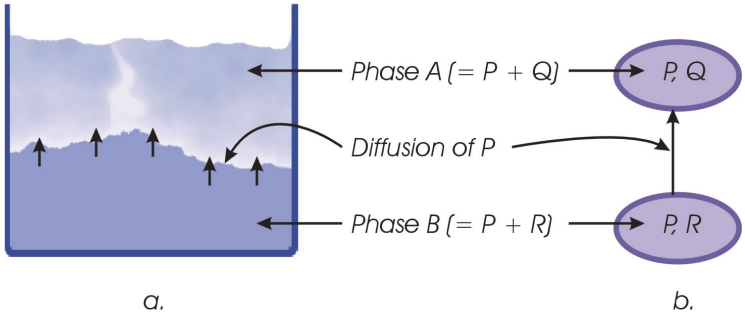

Figure 4.1: a. *A tank with two immiscible phases,* b. *Possible flat topology of diffusion process.*

Consider a tank which contains two immiscible phases A and B (figure 4.1a.) Phase A consists of species P and Q. Phase B consists of species P and R. The simplest way to describe the diffusion of species P from phase A to phase B (or the other way around) is shown in figure 4.1b, which represents a flat topology of the diffusion process. The two phases are each modelled as a simple lumped system and between the systems exists a mass connection which describes the transfer of all species between those systems. Remember that the direction of the connection is only a convention, defining a reference direction for the actual stream. The actual direction of the flow depends on the driving force, which in this case would be the difference in chemical potentials of P in each phase.

To prevent the species Q and R from being transferred through the connection, the permeability concept is introduced: The connection between the two systems is defined as a semi-permeable thermodynamic wall, which is only permeable for species P. This means that only the species P will be transferred through the connection and that, consequently, the transfer terms of the other species (Q and R) will not appear in the (component) mass balances of the systems because they are not transferred.

□

Note that in principle the permeability concept does not have to be limited to mass c.q. species. The concept can be readily extended to any extensive quantity. In practice, though, this is not necessary as one normally distinguishes between mass, heat, momentum and work transfer, which are all handled individually (Preisig *et al.* 1990), (Preisig *et al.* 1989).

4.3 Uni-directionality of Mass Connections

The concept of uni-directionality is also a property that is related with mass connections. The species topology, as it was described so far, allows for bi-directional flow in any mass connection. Since the flow is (normally) dependent on the state of the two connected systems, nothing can be said a priori about the direction of the actual flow. In fact, in many situations the flow would change in the course of the time interval over which the process is being observed. Species that are not inhibited by a defined permeability may thus appear on both sides of the connection, when computing the species topology. In many applications, though, one-directional flows are assumed and thus species propagate in a constraint flow pattern across the network. This desired behaviour calls for defining two classes of mass connections, namely *uni-directional* and *bi-directional*, where the latter should be the normal case as it leaves the system unconstraint. Inversion of the flow for a uni-directional flow would have to be handled as an exception as it contradicts the validity range of the model.

4.4 Reactions

The reaction concept more or less speaks for itself. A chemical or biological reaction is a mechanism that transforms a set of species, the reactants, into a set of different species, the products. At this stage of defining the species topology it is only necessary to know which species are transformed and which species are generated in every (reactive) thermodynamic system. The properties of reactions (kinetic laws, heat of reaction, etc.) and the conditions under which the reaction may take place are introduced at a later stage.

Chapter 5

Mathematical Description: The Equation Topology

Having completed the first two stages of the modelling process, it is already possible to generate the dynamic part of the process model, namely the (component) mass and energy balances for all the elementary systems, using the conservation principles (Westerweele 1999). The resulting (differential) equations consist of flow rates and productions rates, which are not further specified at this point. In order to fully describe the behaviour of the process, all the necessary remaining information (i.e. the mechanistic details) has to be added to the symbolic model of the process. So, in addition to the balance equations, other relationships (i.e. algebraic equations) are needed to express transport rates for mass, heat and momentum, reaction rates, thermodynamic equilibria, definition of intensive variables, and so on. The resulting set of differential and algebraic equations (DAEs) is called the *equation topology*.

From a certain point of view the modelling process can thus be regarded as a succession of equation-picking and equation-manipulation operations. The modeller has, virtually at least, a knowledge base containing parameterised equations that may be chosen at certain stages in the modelling process, appropriately actualised and included in the model. The knowledge base is, in most cases, simply the physical knowledge of the modeller, or might be a reflection of some of his beliefs about the behaviour of the physical process.

In this chapter a classification of the variables and equations that are occurring when modelling physical/chemical systems is made. Next, a concise, abstract canonical form is presented, which can represent the dynamics of any physical-chemical-biological process. After that, the algebraic equations, that are necessary to complete the mathematical model of the process, are classified. The proposed classifications make modelling into an almost trivial activity. A link to linear system and control theory is given by transforming the model

representation such that is resembles a linear state space description. At the end of the chapter substitutions and variable transformations are criticised and control is briefly discussed.

5.1 Variable Classification

The conservation of extensive quantities forms the foundation for describing the dynamic behaviour of systems. Therefore, the set of conserved quantities are called fundamental extensive quantities or the primary state. In section 5.3 it will be shown that the dynamic part (i.e. the differential equations) of physical-chemical-biological processes can be represented in a concise, abstract canonical form, which can be isolated from the static part (i.e. the algebraic equations). This canonical form, which is the smallest representation possible, incorporates very visibly the structure of the process model as it was defined by the person who modelled the process (Westerweele and Preisig 2000): The system decomposition (*physical topology*) and the species distribution (*species topology*) are very visible in the model definition. The transport and production rates always appear linearly in the balance equations, when presented in this form:

$$\dot{\underline{\mathbf{x}}} = \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}} \tag{5.1}$$

in which

$\underline{\mathbf{x}}$	::	Fundamental state vector
$\underline{\mathbf{z}}$	::	Flow of extensive quantities
$\underline{\mathbf{r}}$	::	Kinetics of extensive quantity conversion

The purpose of the following classification is to provide a more effective link between thermodynamics and system theory. The classification of variables that is presented here is, in the first place, based on the structural elements of the modelling approach (namely systems and connections). The subclassification has its origins in thermodynamics. The variables that appear in the mathematical model of a process are divided into three main groups, namely *system variables, connection variables* and *reaction variables.*

5.1.1 System Variables

System variables are variables that are defined within the boundary of a system (or sometimes a group of systems). They can be subdivided into two main groups:

- The basic type of variables are the *fundamental extensive variables* or *fundamental state variables* ($\underline{\mathbf{x}}$) representing the quantities for which

conservation laws are valid and for which we write balance equations (See section 5.2). In most chemical processes, these are component masses (total mass is the sum of the component masses), energy, and momentum.

- The second group of variables is called *application variables* or *secondary state variables* ($\underline{\mathbf{y}}$), which is a gathering of two categories, *intensive* and *geometrical variables*, as well as other variables not covered by the previous classification. The secondary state variables are quantities that are derived from the primary state ($\underline{\mathbf{x}}$). The secondary state must be the result of a mapping of the primary state ($\underline{\mathbf{y}}=\underline{\mathbf{f}}(\underline{\mathbf{x}})$). Thus in any model, this link *must* exist.

  Intensive variables are those that provide a local characterisation of the process. We distinguish among them *potentials* that are driving forces for flows such as temperature, pressure, chemical potential, and *physical properties* characterising the "quality" of the material such as density, or concentration.

  Geometrical variables are those that characterise the dimension configuration of the process, independent of the presence of the physical process. They may describe the extent of the process such as volume, area, etc. or give localised characteristics such as porosity, tortuosity, etc.

  The choice of the intensive and geometric variables that characterise the process is very much application dependent in the sense that it is determined by the final goal of the model. A common property of the intensive and geometric variables is that, in general, these are the variables for which direct information through measurements is available.

5.1.2 Connection Variables

Connection variables are associated with physical connections and can be divided into two groups:

- *Flow rates* of fundamental extensive quantities represent the rates by which the respective quantities are exchanged between two interconnected elementary systems. An important aspect that we retain for our analysis is that some transfers may induce flows of several fundamental extensive quantities. For example, a mass transfer will induce flows of component masses for each substance present in the transferred material, but also flows of momentum and energy. Similarly, a momentum transfer induces also an energy transfer in the form of mechanical work. In general, each physical connection has assigned a unique fundamental flow and, possibly, several (or no) induced flows.

We distinguish in our model between *fundamental or primary flow variables* ($\underline{\mathbf{z}}$) and *secondary flow variables*. The primary flow variables are the variables that appear directly in the balance equations. Secondary flow variables are related to the primary flow variables via an algebraic expression and do not appear in the balance equations (when, of course, no substitutions have been made). For example, the mass flow rate (primary flow variable) of a mass stream of incompressible fluid can be easily expressed as a function of the corresponding volumetric flow (secondary flow variable). The flow rates are usually defined as a function of the secondary state variables of the two interconnected systems ($\underline{\mathbf{z}}$=$\underline{\mathbf{h}}$($\underline{\mathbf{y}}_{or}$, $\underline{\mathbf{y}}_{tar}$)). The variables that represent flow rates will be marked by a hat "ˆ". For example, $\hat{m}$ will typically denote a mass flow.

- *Application variables* are a gathering of *physical properties, geometrical variables* and other variables that are related to the connection (i.e. the common boundary of the two interconnected systems).

5.1.3 Reaction Variables

The last group, the reaction variables, will also be split up into two subgroups:

- *Production rates* ($\underline{\mathbf{r}}$) of fundamental extensive quantities represent the rates by which the respective quantities are produced inside an elementary system. The typical example are the chemical reaction kinetic rates. These variables are associated with those phenomena in which transformations of fundamental extensive quantities occur inside an elementary system. Such is the case for chemical reactions in which component masses are being transformed into one another. As in the case of flow rates, we can speak of fundamental production rates and induced production rates. In the case of chemical reactions, it is common practice (see (Aris 1969)) to use the extent rate of a reaction to describe the evolution of the reaction. The consumption rates of reactants and the production rates of the reaction products are expressed in terms of the extent rate of the reaction and thus become induced production rates. The production rates are usually defined as a function the secondary state variables of a system ($\underline{\mathbf{r}}$=$\underline{\mathbf{g}}$($\underline{\mathbf{y}}$)). The variables that represent production rates will be marked by a tilde "˜".

- *Application variables* are a gathering of variables that are related to the reaction (for example, the pre-exponential kinetic constant or the activation energy of a reaction).

5.2 Fundamental State Variables and Equations

The behaviour of a system is characterised by the evolution of its state with time. The choice of the elementary system is driven by the choice of the time-scale in which the dynamics of the system is to be resolved. This choice is fundamental to the analysis.

In order to characterise the behaviour of a system, information is needed about the natural state of this system (at a given time) and about the change of this state with time. The natural state of a system can be described by the values of a set of *fundamental extensive* quantities, while the change of state is given by the balance equations of those fundamental variables. The fundamental extensive variables represent the "extent" of the system c.q. the quantities being conserved in the system. In other words: they represent quantities for which the conservation principle and consequently also the superposition principle applies. So, for these variables, the balance equations are valid. In most chemical processes, the fundamental variables are: component mass, total energy and momentum. But other extensive quantities (charge, for example) might be necessary sometimes and will obviously have to be present in a generic modelling tool.

The dynamic behaviour of a system can be modelled by applying the conservation principles to the fundamental extensive quantities of the system. The principle of conservation of any extensive quantity x of a system s states that what gets transferred into the system must either leave again or is transformed into another extensive quantity or must accumulate in the system (In other words: no extensive quantity is lost). Symbolically we can write a balance equation for the fundamental extensive variable x^f that characterises the elementary system s as:

$$\frac{d}{dt}x_s^f = \sum_c \alpha_c \hat{x}_c + \sum_p \tilde{x}_p \tag{5.2}$$

where the first sum is taken over all the physical connections c which contribute to the exchange of x_s^f and the second sum is taken over all production/consumption processes p. Further, $\hat{x}_c$ represents the corresponding flow rate of extensive quantity x through connection c and $\tilde{x}_p$ corresponds to the production rate (which is assumed to be of negative sign if it corresponds to a consumption process).

The direction of the flow through a connection is defined relative to the reference co-ordinate system, which, in essence, is introduced for the consistency of the models. This means that a flow is positive if it moves in the reference direction and negative when it moves in the opposite direction. Adding the predicate of direction to every flow allows for dynamic changes of the directionality without affecting the structure of the involved equations. Further,

the transfer is defined only once and is then incorporated in the two sets of balance equations describing the behaviour of the two connected systems. The coefficients $\alpha_c \in \{-1, 1\}$ indicate the conventional direction of the flow rate $\hat{x}_c$.

5.3 Balance Equations

This section focuses on the representation of the dynamic part of process models with the aim to generate a minimal representation, which, though, still shows clearly the structural elements for further use. This allows us to separate the consecutive operations, such as model reduction and implementing various assumptions, clearly from the development of the primary model. Another important application of identifying structures in the modelling process, is to use the extra available information to perform efficient model manipulations in order to achieve, for example, superior numerical performance. Ideas to use model structure for DAE index reduction, for example, can be found in (Marquardt 1995), (Moe 1995) and (Westerweele and Preisig 2000).

Whilst we consider this approach elementary, the suggested approach is clearly new. We present a new concise, abstract canonical form, which is able to represent a very wide range of dynamic equations of first-principle process models, including physical, chemical and biological processes (Westerweele 2001). We call it canonical because it is minimal whilst, at the same time, showing all the structural elements of the process models clearly: the physical topology and the species topology, both of which were defined in the preceding chapters.

5.3.1 Mass Balances

Mass is one of the prime extensive quantities when modelling physical-chemical processes. The generic balance equation (5.2) applies to any of the species present in the system, whereby the number of independent balances is equal to the number of species, that is, if no constraining assumptions are being made. The total mass balance is always the sum of the component balances, thus it is linearly dependent on the component masses.

In most applications it is convenient to use the component mass balances as the basic set of equations. The accumulation of the component masses in a system (Σ) is balanced by the transfer across the system boundary and the internal conversion through reactions:

$$\underline{\dot{\mathbf{n}}}_\Sigma = \sum_{\forall m} \alpha_m \underline{\hat{\mathbf{n}}}_m + \underline{\tilde{\mathbf{n}}}_\Sigma \tag{5.3}$$

where

$\underline{\mathbf{n}}_\Sigma$	::	Vector of species, in moles
$\underline{\hat{n}}_i$	::	Molar massflow of mass connection i
α_i	::	Unit direction of reference co-ordinate $\in \{-1, 1\}$
$\underline{\tilde{\mathbf{n}}}_\Sigma$	::	Production rate for all species

This equation represents the component mass balances of any dynamic system, which is defined in a model of a physical-chemical process. We are interested in finding a concise, abstract general form which represents the complete model of any physical-chemical process. In the following subsections the balance equations (5.3) will be rewritten with the aim of finding a general form for the representation of an interconnected network of systems.

Matrix Notation for Production Term

The production term $\tilde{\mathbf{n}}$, which appears in the component mass balance for a particular species A_j of an elementary system Σ, includes the information about the rate of consumption or production of this species. This production term is therefore defined as the sum of the rate expressions of each elementary reaction, which involves the species A_j. Kinetic rate equations or "reaction laws" are empirical relations, which describe the rate of conversion of one species (A_j) in an elementary reaction as a function of the concentration of all the species in the system. The empirical relations are very often power laws, with the exponents reflecting the number of molecules of that particular species involved in the reaction.

Chemical reactions are defined as stoichiometric equations that relate a number of reactant molecules with a number of product molecules:

$$|\nu_1|A_1 + |\nu_2|A_2 + ... \rightleftarrows ..... + |\nu_{n-1}|A_{n-1} + |\nu_n|A_n \tag{5.4}$$

With

ν_i	::	Stoichiometric coefficient (>0 for products, <0 for reactants)
A_i	::	ith species

The coefficients ν_i in this "equation" are called the *stoichiometric coefficients*. In order to use the reaction concept in mathematical modelling, the "reaction equation" is slightly generalised, such that the chemical reaction can be represented in a form that is almost like an equation. That is, for every reaction a *generalised stoichiometric equation* is defined:

$$\sum_{i=0}^{n} \nu_i A_i = 0 \tag{5.5}$$

or, in matrix notation:

$$\underline{\boldsymbol{\nu}}^T \underline{\mathbf{A}} = 0 \tag{5.6}$$

With,

$\underline{\boldsymbol{\nu}}$ :: Stoichiometric coefficient vector = $(\nu_1\ \nu_2\ .....\nu_n)^T$ for n species
$\underline{\mathbf{A}}$ :: Species vector, containing all 'reactive' species of the system under consideration = $(A_1\ A_2\ .....A_n)^T$ for n species

So, for a system involving multiple reactions, one can write the reactions in the following generalised fashion:

$$\underline{\underline{\mathbf{S}}}\,\underline{A} = \underline{\mathbf{0}} \tag{5.7}$$

With,

$\underline{\underline{\mathbf{S}}}$ :: Stoichiometric matrix = $(\underline{\boldsymbol{\nu}}_1\ \underline{\boldsymbol{\nu}}_2.....\underline{\boldsymbol{\nu}}_m)^T$ for m reactions. (Dimension: m x n)

Example 5.1: ***Stoichiometry***

Given a reaction system:

$$A + B \rightleftarrows C \tag{5.8}$$
$$4B + 2D \rightleftarrows 2E + F \tag{5.9}$$
$$C \rightleftarrows 4D \tag{5.10}$$
$$C + 2E \rightleftarrows A + 2D + G \tag{5.11}$$

the stoichiometric matrix is:

$$\underline{A}^T = \begin{pmatrix} A & B & C & D & E & F & G \end{pmatrix} \tag{5.12}$$

$$\underline{\underline{\mathbf{S}}} = \begin{pmatrix} -1 & -1 & 1 & & & & \\ & -4 & & -2 & 2 & 1 & \\ & & -1 & 4 & & & \\ 1 & & -1 & 2 & -2 & & 1 \end{pmatrix} \tag{5.13}$$

□

For a species A_j participating in a reaction r the following relation always holds:

$$r_{r,A_j} = \overrightarrow{r_{r,A_j}} - \overleftarrow{r_{r,A_j}} \tag{5.14}$$

With

r_{r,A_j} :: Rate of formation of species A_j by reaction r
$\overrightarrow{r_{r,A_j}}$ :: Rate of production (or consumption) of species A_j by forward reaction of r
$\overleftarrow{r_{r,A_j}}$ :: Rate of consumption (or production) of species A_j by backward reaction of r

This means that the rate, with which the species A_j is produced (or consumed), is defined as the rate of the forward minus the rate of the backward reaction.

Obviously, the rate of production and consumption of the products and reactants are related through the stoichiometric coefficients. For each reaction, a unique quantity ξ_r can be defined that is a normalised rate of reaction and which can be shown to be the time derivative of the extent of reaction:

$$\xi_r = \frac{r_{r,A_j}}{\nu_{r,A_j}} \tag{5.15}$$

In homogeneous reaction systems, the production rate for a species is the cumulative production rate for this species over all reactions. If the dimensions of the reaction rates of a system are defined in mass per time unit then the reaction term $\underline{\tilde{\mathbf{n}}}$, which occurs in the component mass balances of a system may be written as follows:

$$\underline{\tilde{\mathbf{n}}} = \underline{\underline{\mathbf{S}}}^T \, \underline{\boldsymbol{\xi}} \tag{5.16}$$

With

$\underline{\xi}$ :: Normalised reaction rates vector $= (\xi_1 \; \xi_2 \; .....\xi_m)^T$ for m reactions

General Notation for Accumulation and Transfer Terms

The mass connections describe the exchange of mass across the boundary seperating two adjacent systems. The direction of the flow through a connection is defined relative to the reference co-ordinate system, which, in essence, is introduced for the consistency of the models. This means that a flow is positive if it moves in the reference direction and negative when it moves in the opposite direction. Adding the predicate of direction to every flow allows for dynamic changes of the directionality without affecting the structure of the involved equations. Further, the transfer is defined only once and is then incorporated in the two sets of balance equations describing the behaviour of the two connected systems.

We will now consider an elementary system in which no reactions occur and which is part of a mass domain (i.e. a set of mass-interconnected systems) that holds k species. The system Σ has m mass connections. The component

mass balances of this system read:

$$\underline{{}^{*}\dot{\mathbf{n}}}_{\Sigma} = \sum_{\forall m} \alpha_m \underline{{}^{*}\hat{\mathbf{n}}}_m \tag{5.17}$$

in which $\underline{{}^{*}\mathbf{n}}_{\Sigma}$ is the vector of molar masses of system Σ for all k species present in the mass domain and $\underline{{}^{*}\hat{\mathbf{n}}}_m$ is the vector of molar mass flows (for all k species) of mass connection m.. This is not the minimal representation, for this would only include the component masses of the species that are present in the system. Also, not all the species that are actually present in the system have to flow through the mass connections. Some connections may be unidirectional or have a specified permeabililty, whereby the permeability of a mass connection is a property which constrains the mass exchange of certain species between the connected systems (Preisig 1994*b*). All this calls for the introduction of a selection matrix $\underline{\underline{\mathbf{P}}}$, which we define as follows:

$$\underline{\underline{\mathbf{P}}} = \underline{\boldsymbol{\gamma}} \odot \underline{\underline{\mathbf{I}}}_k \tag{5.18}$$

where

$\underline{\boldsymbol{\gamma}}$	::	Logical vector selecting species from a species set, dimension k
$\odot$	::	Logical operator: eliminates ith row if γ_i is zero
$\underline{\underline{\mathbf{I}}}_k$	::	Indentity matrix, dimension $k \times k$
$\underline{\underline{\mathbf{P}}}$	::	Selection matrix

The resulting matrix $\underline{\underline{\mathbf{P}}}$ is derived from the identity matrix $\underline{\underline{\mathbf{I}}}_k$ by eliminating the i^{th} row for each γ_i being zero. The matrix thus has dimension $l \times k$ where $l \leq k$ and l is the number of ones in $\underline{\boldsymbol{\gamma}}$. Multiplying a vector that represents all species of the mass domain (in a specific system or mass connection) with the selection matrix $\underline{\underline{\mathbf{P}}}$ (of this specific system or mass connection) results in a vector of present species (in this specific system or mass connections):

$$\underline{\mathbf{n}} = \underline{\underline{\mathbf{P}}}\, \underline{{}^{*}\mathbf{n}} \tag{5.19}$$

in which

$\underline{\mathbf{n}}$	::	Vector of molar present masses

Premultiplied with its transposed, the selection matrix yields a vector that is the same as the vector with all species, but with zeros substituted for species not being present. With $\underline{{}^{*}\mathbf{n}}$ being a vector in which some elements are undefined, because the respective species is not present, the operation of above equation substitutes for the undefined element a zero. The product $\underline{\underline{\mathbf{P}}}^T \underline{\underline{\mathbf{P}}}\, \underline{{}^{*}\mathbf{n}}$ has thus the same information contents as $\underline{{}^{*}\mathbf{n}}$. Thus we write $\underline{{}^{*}\mathbf{n}} \simeq \underline{\underline{\mathbf{P}}}^T \underline{\underline{\mathbf{P}}}\, \underline{{}^{*}\mathbf{n}}$.

Using these definitions, the component mass balances read:

$$\underline{{}^{*}\dot{\mathbf{n}}}_{\Sigma} = \sum_{\forall m} \alpha_m \underline{\underline{\mathbf{P}}}_m^T \underline{\underline{\mathbf{P}}}_m \underline{{}^{*}\hat{\mathbf{n}}}_m = \sum_{\forall m} \alpha_m \underline{\underline{\mathbf{P}}}_m^T \underline{\hat{\mathbf{n}}}_m \tag{5.20}$$

If we finally introduce the systems selection matrix $\underline{\underline{\mathbf{P}}}_{\Sigma}$, the reduced representation becomes:

$$\underline{\dot{\mathbf{n}}}_{\Sigma} = \underline{\underline{\mathbf{P}}}_{\Sigma}{}^{*}\underline{\dot{\mathbf{n}}}_{\Sigma} = \sum_{\forall m} \alpha_m \underline{\underline{\mathbf{P}}}_{\Sigma}\, \underline{\underline{\mathbf{P}}}_{m}^{T}\, \underline{\hat{\mathbf{n}}}_{m} \tag{5.21}$$

It should be noted that:

$$\underline{\underline{\mathbf{P}}}^{T}\, \underline{\underline{\mathbf{P}}} = \mathrm{diag}(\underline{\gamma}) \tag{5.22}$$

Example 5.2: ***Selection Matrix Properties***

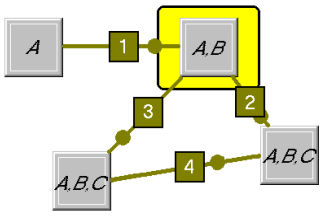

Figure 5.1: *A massdomain with four systems and three species.*

Consisder a small mass domain consisting of four systems, four mass connections and three species. The highlighted system (see fig. 5.1) only holds two of the three species present in the mass domain. For this system the following relations hold:

$$\underline{\gamma} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \quad => \quad \underline{\underline{\mathbf{P}}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{5.23}$$

$$\begin{bmatrix} n_A \\ n_B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} n_A \\ n_B \\ undefined \end{bmatrix} \tag{5.24}$$

$$\begin{bmatrix} n_A \\ n_B \\ 0 \end{bmatrix} \cong \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} n_A \\ n_B \\ undefined \end{bmatrix} \tag{5.25}$$

□

Representation for Interconnected Network of Systems

We shall now consider an interconnected network of systems in a mass domain consisting of n systems, m connections, k species but no reactions. The mass balance equations of a system Σ of this mass domain is now running over all connections, also including connections that are not attached to this system Σ. This is readily achieved by modifying the definition of α to include "no flow", thus for a connection j that is not connected to the system Σ: $\alpha_{\Sigma,j} = 0$:

$$\underline{\dot{\mathbf{n}}}_{\Sigma} = \sum_{\forall m} \alpha_{\Sigma,m} \underline{\underline{\mathbf{P}}}_{\Sigma} \underline{\underline{\mathbf{P}}}_{m}^{T} \underline{\hat{\mathbf{n}}}_{m} \tag{5.26}$$

Where

$\alpha_{\Sigma,i}$ :: Redefined unit direction of reference co-ordinate $\in \{-1, 0, 1\}$

This means that for all n systems the terms $\underline{\underline{\mathbf{P}}}_{m}^{T} \underline{\hat{\mathbf{n}}}_{m}$ in the summation term will be exactly the same (because $\underline{{}^{*}\hat{\mathbf{n}}}_{m} \simeq \underline{\underline{\mathbf{P}}}_{m}^{T} \underline{\hat{\mathbf{n}}}_{m}$). A concise abstract form, which represents the component mass balances of all n systems of the mass domain, can now be formulated:

$$\underline{\dot{\mathbf{n}}} = \underline{\underline{\Gamma}}_{\Sigma} \underline{\underline{\mathbf{A}}}_{k} \underline{\underline{\Gamma}}_{m}^{T} \underline{\hat{\mathbf{n}}} \tag{5.27}$$

or, equivalently:

$$\underline{\dot{\mathbf{n}}} = \underline{\underline{\Gamma}}_{\Sigma} \underline{\underline{\mathbf{A}}}_{k} \underline{\underline{\Gamma}}_{m}^{T} \underline{\underline{\Gamma}}_{m} \underline{{}^{*}\hat{\mathbf{n}}} \tag{5.28}$$

For this representation the following definitions are made:

$$\underline{\mathbf{n}} = [\underline{\mathbf{n}}_1^T \ \underline{\mathbf{n}}_2^T \ ....\underline{\mathbf{n}}_n^T]^T \quad \text{and} \quad \underline{\hat{\mathbf{n}}} = [\underline{\hat{\mathbf{n}}}_1^T \ \underline{\hat{\mathbf{n}}}_2^T \ ....\underline{\hat{\mathbf{n}}}_m^T]^T \tag{5.29}$$

$$\underline{\underline{\Gamma}} = \text{blockdiag}(\underline{\underline{\mathbf{P}}}_{i}) \tag{5.30}$$

$$\underline{\underline{\mathbf{A}}}_{k} = \underline{\underline{\mathbf{A}}} \otimes \underline{\underline{\mathbf{I}}}_{k} \tag{5.31}$$

where

$\underline{\underline{\mathbf{A}}}$ :: Interconnection matrix $= [\underline{\boldsymbol{\alpha}}_{\Sigma 1} \underline{\boldsymbol{\alpha}}_{\Sigma 2} ..... \underline{\boldsymbol{\alpha}}_{\Sigma n}]^T$, dimension $n \times m$

$\underline{\boldsymbol{\alpha}}_{\Sigma i}$ :: Vector of unit direction of reference co-ordinates of system Σ, with $\alpha_{i,j} \in \{-1, 0, 1\}$

$\otimes$ :: Kronecker tensor product, which multiplies each element of the matrix $\underline{\underline{\mathbf{A}}}$ with the matrix $\underline{\underline{\mathbf{I}}}_{k}$ (dimension $\underline{\underline{\mathbf{A}}}_{k} = (nk) \times (mk)$)

If (some of) the systems of the mass domain incorporate reactions then the equation 5.27 simply becomes:

$$\underline{\dot{\mathbf{n}}} = \underline{\underline{\Gamma}}_{\Sigma} \underline{\underline{\mathbf{A}}}_{k} \underline{\underline{\Gamma}}_{m}^{T} \underline{\hat{\mathbf{n}}} + \underline{\underline{\Gamma}}_{\Sigma} \underline{\underline{\mathbf{S}}}_{\Sigma}^{T} \underline{\boldsymbol{\xi}} \tag{5.32}$$

in which

$$\underline{\underline{\mathbf{S}}}_{\Sigma}^{T} = \text{blockdiag}(\underline{\underline{\mathbf{S}}}_{i}^{T}) \qquad \text{with } i \in \{1, 2, ..., n\} \tag{5.33}$$

and $\dim(\underline{\underline{\mathbf{S}}}_{i}^{T})$ = (number of species in mass domain)×(number of reactions in system i)

5.3.2 Energy Balances

Energy is the second most important extensive quantity in chemical process modelling. The energy balance has quite an involved history as it has been the central object of thermodynamics. Many books have been written on the subject of terminology of the thermodynamics and thermostatics and a lot has been talked about the shortcomings of the theories itself. Grijseels gives an outline of the theories that have been developed over the last century and gives a clear view on what the problems are (Grijseels 1999). Such a discussion goes beyond the scope of this project and we will simply accept the basic assumptions that form the foundation of the thermodynamic theory.

In the most generic form, total energy is balanced. Total energy is the sum of internal, kinetic and potential energy, all of which are associated with mass. The communication with other systems, though, is not limited to these three forms of energy, but also heat and work affect the energy content of a system.

Particularly heat is an interesting form of energy. Whilst today heat conduction is known to be based on kinetics - energy transfer on the molecular level - early in the last century, heat was thought to be a massless material that could be moved from one body to another. It was named caloric.

Work is an accumulation of various other energy related effects, such as shaft (mechanical) work, volume work, electrical work and others.

A total energy balance for an arbitrary system, assuming reversibility of all processes, may be written in the from

$$\dot{E} = \sum_{\forall m} \alpha_m \hat{E}_m + \sum_{\forall h} \alpha_h \hat{q}_h + \sum_{\forall l} \alpha_l \hat{w}_l \tag{5.34}$$

with:

E	::	Total energy := $U + K + P$
U	::	Internal energy
K	::	Kinetic energy
P	::	Potential energy
$\hat{E}$	::	Energy stream
$\hat{q}$	::	Heat stream
$\hat{w}$	::	Work stream

The total energy balance, as represented by equation 5.34, is almost never used in this form. Usually a modified version of this basic energy balance is employed for modelling a process component. This modified version is derived from the basic energy balance through some simplifications and assumptions. Caution should be taken, though, when one uses derived energy models, because they are often incorrect or used incorrectly. One could easily introduce faults when one is further simplifying a derived model, because of lack of knowledge about previous assumptions and derivation steps. Knowledge about the common assumptions and the derivation steps is thus essential for the correct use of the different simplified energy models.

In the following, we will derive the frequently used enthalpy balance and internal energy balance from 5.34. Firstly, the attention focuses on the work terms, in particular the volumetric work. Each mass stream is associated with a volumetric work term because mass is of finite volume. Input streams of mass add volumetric injection work and mass output stream are associated with volumetric ejection work. Further, the volume of the system itself may change with time. First split the sum of work terms into mass flow related work, system volume work and other work terms:

$$\sum_{\forall l} \alpha_l \hat{w}_l := \sum_{\forall m} \alpha_m \hat{w}_m^V + \alpha_s \hat{w}_s^V + \sum_{\forall k} \alpha_k \hat{w}_k \tag{5.35}$$

The injection and ejection work terms are given by:

$$\hat{w}_m^V := p_m \hat{V}_m \tag{5.36}$$

and the system's volume work by:

$$\alpha_s \hat{w}_s^V = -p \frac{dV}{dt} \tag{5.37}$$

The volume work terms associated with the injection and ejection of mass across the system boundary is moved from the work flows to the mass flows and therefore the index set of the work streams changes, indicated by the new index k:

$$\dot{E} = \sum_{\forall m} \alpha_m (\hat{E}_m + \hat{w}_m^V) + \sum_{\forall h} \alpha_h \hat{q}_h + \sum_{\forall k} \alpha_k \hat{w}_k - p \frac{dV}{dt} \tag{5.38}$$

Since the pair of internal energy and volume work shows up in every term associated with mass, it was defined as a new quantity, called enthalpy:

$$H := U + pV \tag{5.39}$$

$$\begin{aligned} \frac{d(U+K+P)}{dt} &= \sum_{\forall m} \alpha_m (\hat{H}_m + \hat{K}_m + \hat{P}_m) + \sum_{\forall h} \alpha_h \hat{q}_h \\ &\quad + \sum_{\forall k} \alpha_k \hat{w}_k - p \frac{dV}{dt} \end{aligned} \tag{5.40}$$

Up to this point, no assumptions have been made and equation 5.38 still contains the same information as the original total energy balance (5.34). The next step involves some assumptions, which are:

$$\frac{dK}{dt} = 0 \quad \text{[system does not move]} \tag{5.41}$$

$$\frac{dP}{dt} = 0 \quad \text{[system does not move]} \tag{5.42}$$

$$\sum_{\forall m} \alpha_m \hat{K}_m = 0 \quad \text{[no net change of kinetic energy]} \tag{5.43}$$

$$\sum_{\forall m} \alpha_m \hat{P}_m = 0 \quad \text{[no net change of potential energy]} \tag{5.44}$$

which says that the system is assumed not to move during the process and that the effects of kinetic and potential energy streams are compensating each other or, at least, small compared to other energy effects in the system.

Applying these assumptions results in the now reduced form:

$$\frac{dU}{dt} = \sum_{\forall m} \alpha_m \hat{H}_m + \sum_{\forall h} \alpha_h \hat{q}_h + \sum_{\forall k} \alpha_k \hat{w}_k - p\frac{dV}{dt} \tag{5.45}$$

and, making use of the fact that $\frac{dpV}{dt} = p\frac{dV}{dt} + V\frac{dp}{dt}$, we get the enthalpy balance:

$$\frac{dH}{dt} = \sum_{\forall m} \alpha_m \hat{H}_m + \sum_{\forall h} \alpha_h \hat{q}_h + \sum_{\forall k} \alpha_k \hat{w}_k + V\frac{dp}{dt} \tag{5.46}$$

The enthalpy balance over all systems, written in matrix notation, becomes:

$$\underline{\dot{\mathbf{H}}} = \underline{\underline{\mathbf{A}}}_m \underline{\hat{\mathbf{H}}}_m + \underline{\underline{\mathbf{A}}}_q \underline{\hat{\mathbf{q}}} + \underline{\underline{\mathbf{A}}}_k \underline{\hat{\mathbf{w}}}_k + \underline{\underline{\mathbf{V}}}\,\underline{\dot{\mathbf{p}}} \tag{5.47}$$

With

$\underline{\underline{\mathbf{A}}}_i$ :: Interconnection matrices
$\underline{\underline{\mathbf{V}}}$ :: $diag(\underline{V})$

It should be noted here that the terms $\underline{\hat{\mathbf{H}}}_m$ are enthalpy flows that are induced by corresponding mass flows. So, a mass flow always induces a flow of energy.

Assuming that the pressure in each system does not change (a very common assumption), eliminates the $\underline{\underline{\mathbf{V}}}\,\underline{\dot{\mathbf{p}}}$ term from the enthalpy balance and leaves only flow terms. If this assumption is not made the balance equation can be rearranged to produce:

$$\underline{\dot{\mathbf{H}}} - \underline{\underline{\mathbf{V}}}\,\underline{\dot{\mathbf{p}}} = \underline{\underline{\mathbf{A}}}_m \underline{\hat{\mathbf{H}}}_m + \underline{\underline{\mathbf{A}}}_q \underline{\hat{\mathbf{q}}} + \underline{\underline{\mathbf{A}}}_k \underline{\hat{\mathbf{w}}}_k \tag{5.48}$$

In chapter 6 we will learn that in order to get an index 1 DAE, there *has to* be an algebraic equation that directly or indirectly (for example via the temperature) relates the enthalpy (H) of a system to the pressure (p) in that system. Only in that case the number of equations and "unknown" variables will be the same. This means that the pressure in a system will always be calculated via pseudo steady-state relations and pressure changes in a system shall therefore always be "fast".

The energy balance is (almost) always used to obtain information about the temperature in the system and it is very often seen that this balance is further modified (by doing further transformations and substituting the component mass balances into the energy balance) such that eventually a "temperature balance" is obtained. I will not perform this, often cumbersome derivation here and state that it is often not necessary to do so. The same results can be obtained when the transformation is not made.

5.3.3 Conclusions

In this section we have derived that the variables that are appearing in the balance equations (i.e. transport and production variables) always appear linearly for physical and chemical systems. The nonlinearities of a process will therefore always emerge in the algebraic relations of the model.

The balance equations can always be abstracted with the following simple form:

$$\frac{d}{dt}\underline{\mathbf{x}} = \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}} \tag{5.49}$$

In which,

$\underline{\underline{\mathbf{A}}}$	::	Interconnection matrix
$\underline{\underline{\mathbf{B}}}$	::	Stoichiometric matrix,
$\underline{\mathbf{x}}$	::	Primary state vector
$\underline{\mathbf{z}}$	::	Transport rate vector
$\underline{\mathbf{r}}$	::	Reaction rate vector

The matrices $\underline{\underline{\mathbf{A}}}$ and $\underline{\underline{\mathbf{B}}}$ are completely defined by the model designers definition of the physical and species topology of the process under investigation. Therefore these matrices can automatically be constructed by a computer program. The only things a model designer has to do to complete the model are:

- Provide a link between the transport and reaction rate vectors and the primary state vector. Each element in the transport and reaction rate vectors has to be (directly or indirectly) linked to the primary state vector. This "linking" is done with one or more algebraic equations. If certain elements of the rate vectors are not defined in the algebraic

equations, the mathematical system will have too many unknowns and can consequently not be solved. Exceptions to this rule will be handled in chapter 6.

- Give a mapping which maps the primary state of each system in a secondary state. This mapping is necessary because usually transport and reaction rates are defined as functions of secondary state variables (a heat flow can, for example, be expressed as a function of temperature difference).

5.4 Algebraic Equations

In addition to the balance equations, we need other relationships to express thermodynamic equilibria, reaction rates, transport rates for heat, mass, momentum, and so on. Such additional relationships are needed to complete the mathematical modelling of the process. A model designer should be allowed to choose a particular relationship from a set of alternatives and to connect the selected relationship to a balance equation or to another defined relationship. As with the variables, we divide the algebraic equations into three main classes, namely *system equations, connection equations* and *reaction equations.*

5.4.1 System Equations

For each system that is defined within the physical topology of a process, a mapping is needed which maps the primary state variables ($\underline{\mathbf{x}}$) into a set of secondary state variables ($\underline{\mathbf{y}} = \underline{\mathbf{f}}(\underline{\mathbf{x}})$). The primary states of a system are fundamental quantities for describing the behaviour of the system. The fundamental state is defined intrinsically through the fundamental behaviour equations. The application of fundamental equations of component mass and energy balances intrinsically defines component mass and energy as the fundamental state variables. Alternative state variables are required for the determination of the transfer rate of extensive quantities and their production/consumption rate.

The equations that define secondary state variables do not have to be written in explicit form, but it has to be possible to solve the equations (either algebraically or numerically) such that the primary state can be mapped into the secondary state (this is further explained in chapter 10). This means that each defined equation *has to* define a new variable. Equations that link previously defined variables together are not allowed, since the number of equation would then exceed the number of variables and the set of equations of this system would thus be overdetermined. Consider the following example:

Example 5.3: *Redundant Equations*

It is often seen that people insist on using so-called "normalising" equations, such as the sum of the fractions equals one, to complete their model definition. Such an equation is actually redundant when you think about it, because the definition of fraction intrinsically implies this:

$$\underline{\mathbf{x}} := \frac{\underline{\mathbf{n}}}{\underline{\mathbf{e}}^T \underline{\mathbf{n}}} \tag{5.50}$$

With

$\underline{\mathbf{x}}$ :: *Molar fractions vector*
$\underline{\mathbf{n}}$ :: *Molar mass vector*
$\underline{\mathbf{e}}$ :: *unity vector* $= [1\ 1\ 1\ 1.....]^T$

Premultiplying this definition with the transposed unity vector $\underline{\mathbf{e}}^T$ *gives:*

$$\underline{\mathbf{e}}^T \underline{\mathbf{x}} := \frac{\underline{\mathbf{e}}^T \underline{\mathbf{n}}}{\underline{\mathbf{e}}^T \underline{\mathbf{n}}} = 1 \tag{5.51}$$

Equations 5.50 and 5.51 are linerarly dependent. So, adding the equation $\underline{\mathbf{e}}^T \underline{\mathbf{x}} = 1$ *to the model does not add any new information and could actually make computations more difficult. With our modelling method, an equations like this is not allowed, since it does not define a new variable.*

□

Due to the nature of the different secondary variables, the system equations are subdivided in some subclasses:

State variable transformations.

State variable transformations are relationships that provide links between the internal state of a system and various state variables appearing in kinetic laws, transfer laws, and other definitions such as physical and geometrical properties. For example, the concentration $\underline{\mathbf{c}}$ of the components in a system can be characterised by their molar masses $\underline{\mathbf{n}}$ divided by the volume of the system V:

$$\underline{\mathbf{c}} := \frac{\underline{\mathbf{n}}}{V} \tag{5.52}$$

It should be noted here that, although these equations are called state variable transformations it is not implied that these equations actually

have to be used for substitutions and transformations. It is very often seen that modellers insist on doing substitutions, hereby transforming the fundamental balance equations. These often cumbersome transformations are usually not necessary for solving the problem under investigation and can usually be omitted.

Physical property relations.

Processing systems may consist of a variety of materials in the form of pure materials, mixtures, dispersions or any other combination of materials in any state of aggregation. The modelling of these systems potentially requires knowledge about the physical properties of all involved materials. Examples of physical properties are: viscosity, thermal conductivity, diffusivity, partial molar enthalpy and density.
While they are usually thought of as constants, they may change with changing conditions in the system. They can also be a function of geometrical properties and/or indirectly a function of other physical properties. The thermal diffusivity, for example, can be defined as a quotient of the thermal conductivity divided by the density times the heat capacity.

Geometrical property relations.

The body volume and the boundary area (surface) of a system are two geometrical properties of a system. These properties can be characterised by alternative sets of geometrical properties. For example, the body volume and boundary surface of a cylindrical system can be characterised by the radius and the length of the cylinder.
Because the system can change its shape as a function of the changing state, the geometrical properties of a system are a function of the systems state, and with it a function of the systems physical properties. For example, increasing the temperature of a system at a constant pressure, can expand the volume and boundary of a system. This can be considered as a result of changing the density (a physical property) of the contents of the system. The volume and mass of a system are always linked through the density.

Equations of state

Equations of state are equations that express algebraic relations between the application variables that characterise an individual system and that are supposed to hold at each moment during the evolution of the process. The term "equations of state" is borrowed from the thermodynamics as it is used, for example, for the well-known relation between the pressure p, the molar volume V_n and the temperature T of an ideal gas

$$pV_n = RT \tag{5.53}$$

called the *equation of state of the ideal gas*. Notice, however, that we use this term here in a broader sense.

Besides the enumerated types of equations there might be other relations that need to be considered during the modelling process.

5.4.2 Connection Equations

The flow rates, which emerge in the balance equations of a system, represent the transfer of extensive quantities to and from adjacent systems. These flow rates can be specified or linked to transfer laws, which are usually empirical or semi-empirical relationships. These relationships are usually functions of the states, and the physical and geometrical properties of the two connected systems. For example, the rate of conductive heat transfer Q through a surface A_t between two objects with different temperatures can be given by:

$$Q := UA_t(T_1 - T_2) \tag{5.54}$$

This relationship depends on the temperatures T_1 and T_2 of the two objects respectively. Temperature is of course a (secondary) state variable. The rate of heat transfer also depends on the overall heat transfer coefficient U, which is a physical property of the common boundary segment between the two systems, and on the total area of heat transfer A_t, which is a geometrical property.

A transfer law thus describes the transfer of an extensive quantity between two adjacent systems. The transfer rate usually depends on the state of the two connected systems and the properties of the boundary in between.

In some cases it's difficult to find a reliable equation, which describes the flow through a connection. If one has no equation for the flow one may have to find one or one has to make certain time scale assumptions. In the latter case, the flow rate is not specified, but constraints are imposed on the state variables of the connected system (such that the number of equations in the mathematical model equals the total number of variables). To make the states satisfy the constraints the flows are forced to satisfy certain values. So, in these cases, the flow rates are not defined in the algebraic equations, but are present in the differential equations and this will lead to so-called "high-index" models The causes and solutions of high-index process models will be discussed in chapter 6.

For each defined connection a model designer thus has a choice: either specify the dynamics or impose constraints on differential variables of the interconnected systems.

5.4.3 Reaction Equations

Depending on the time scale of interest, we can divide reactions into three groups (Westerweele and Preisig 2000):

- Very slow reactions (slow in the measure of the considered range of time scales). These reactions do not appreciatly occur and may be simply ignored.
- Reactions that occur in the time-scale of interest. For these reactions kinetic rate laws can be used.
- Very fast reactions (relative to the considered time scale), for which is assumed that the equilibrium is reached instantaneously.

As the non-reactive parts do not further contribute to the discussion, they are left out in the sequel. For the "normal" reactions the reaction rates of the reactions in the relevant times scale must be defined by kinetic rate equations. The production terms are linked to kinetic laws, which are empirical equations. They are usually written as a function of a set of intensive quantities, such as concentrations, temperature and pressure. For example, the reaction rate r of a first-order reaction taking place in a lump is given by:

$$r := Vk_0 e^{-E/RT} c_A \tag{5.55}$$

where

r	::	Reaction rate of a first-order reaction
V	::	Volume of the system
k_0	::	Pre-exponential kinetic constant
E	::	Activation energy for the reaction
R	::	Ideal gas constant
T	::	Temperature of the reacting system
c_A	::	Concentration of component A

Temperature and concentration(s) of the reactive component(s) are state variables of the reactive system. Reaction constants and their associated parameter such as activation energy and pre-exponential factors are physical properties. In some cases, also geometrical properties of the system are part of the definition of the kinetic law, such as the porosity or other surface characterising quantities.

The fast (equilibrium) reaction rates are not defined, because the equilibrium reactions are considered to have very fast dynamics relative to the time scale of the process[1]. For these reactions only the reaction outcome has to be

[1] If one simply does not know the rate expression, the assumption of reaching the equilibrium quickly may also be done. This assumption may or may not be valid given the dynamic time window.

given in the form of an equilibrium relation, which should hold at every time instant. This is usually a nonlinear, algebraic relation that relates the masses of the involving species to each other (e.g.: for a reaction $A \rightleftarrows B + C$ one may write: $K = \frac{c_A}{c_B c_C}$). Consequently, unlike the situation where no equilibrium reactions occur, the initial values of the masses of the involved species cannot be arbitrarily chosen because the quantities of some species in the system are now directly related to the quantities of some other species in the system. This results in some differential equations of the system being directly related to each other. Also, the production terms of the equilibrium reactions occur only in the component mass balances and cannot be determined directly from system equations. As with the undefined connections, the result is a high index model. The problems and solutions regarding high index models will be handled in chapter 6.

For each defined reaction a modeller must either specify the kinetics or give an equilibrium relation.

5.4.4 Summary

A mathematical model of a process can be abstracted with the following fom:

$$\begin{array}{ll} \text{Balance equations:} & \frac{d}{dt}\underline{\mathbf{x}} = \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}} \\ \text{Variable declarations:} & \underline{\mathbf{y}}_\Sigma = \underline{\mathbf{f}}(\underline{\mathbf{x}}_\Sigma) \\ \text{Transfer laws:} & \underline{\mathbf{z}}_c = \underline{\mathbf{g}}_1(\underline{\mathbf{y}}_{or}, \underline{\mathbf{y}}_{tar}) \text{ or } \underline{\mathbf{0}} = g_2(\underline{\mathbf{y}}_{or}, \underline{\mathbf{y}}_{tar}) \\ \text{Production laws:} & \mathbf{r}_{i,\Sigma} = \underline{\mathbf{h}}_1(\underline{\mathbf{y}}_\Sigma) \text{ or } \mathbf{0} = \underline{\mathbf{h}}_2(\underline{\mathbf{y}}_\Sigma) \end{array} \tag{5.56}$$

In this (simplified!) form, the balance equations run over all the fundamental extensive quantities ($\underline{\mathbf{x}}$) of all defined systems, over all the defined connections ($\underline{\mathbf{z}}$) and over all the defined reactions ($\underline{\mathbf{r}}$) of the physical topology. The matrices $\underline{\underline{\mathbf{A}}}$ and $\underline{\underline{\mathbf{B}}}$ are completely defined by the model designers definition of the physical and species topology of the process under investigation.

For each system in the physical topology a mapping must exist, which maps the primary state (i.e. the fundamental state $\underline{\mathbf{x}}_\Sigma$) into a secondary state ($\underline{\mathbf{y}}_\Sigma$). For each connection either the transfer law must be given, which defines the flow ($\underline{\mathbf{z}}_c$) through the connection as a function of secondary state variables of the two interconnected systems, or a constraint must be given. For each reaction either the production law must be given, which defines the reaction rate ($\mathbf{r}_{i,\Sigma}$) of the reaction as a function of secondary state variables of the system the reaction takes place in, or a constraint must be given. When a constraint is given for a connection or a reaction, two or more fundamental state variables are linked together, either directly or indirectly. This means that not all state

variables are independent and that some kind of index reduction method shall have to be applied. A detailed discussion on the problems and their solutions, when introducing assumptions, is given in chapter 6.

Each of the groups of equations in the model form has its own distinct functionality within the model as each encapsulates a specific kind of knowledge. From a methodological point of view the classification of the model equations has the advantage of separating the sources of modelling knowledge and implicitly the sources of uncertainty in the model.

5.5 Linear and Linearised Models

Most models of processes that are of interest contain nonlinear algebraic equations (when modelling with our method the differential equations will never be nonlinear). A large part of the literature on models, however, is devoted to linear systems, mainly because there is no general mathematical theory for the analytic solution of nonlinear equations. A reason for studying linear models is the availability of analysis tools that enable strong results on system and control theoretic properties such as stability, controllability, observability, optimality, robustness, etc.. Furthermore, first order approximations are in many cases sufficient to characterise the local behaviour of the nonlinear model. This means that often analysis based on linearisations reveals properties of the model locally (Sontag 1990)(Kailath 1980). The application of linear models is, however, restricted since desirable and expected behaviour of the model can only be guaranteed for operating conditions that are close to the point of linearisation.

5.5.1 Linearisation

Linearisation is the process by which we approximate nonlinear equations with linear ones. It is widely used in the study of process dynamics and design of control systems for the following reasons (Stephanopoulos 1984):

- Closed-form, analytic solutions can be obtained for linear systems. Thus a complete and general picture of a process's behaviour can be obtained, independently of the particular values of the parameters and input variables. This is not possible for nonlinear systems, and computer simulation only provides the behaviour of the system at specified values of inputs and parameters.

- Most of the significant developments toward the design of effective control systems have been limited to linear processes.

The operation "linearisation" is straightforward: a non-linear function is approximated (locally) by a linear function. Linearisation is typically done by using a Taylor expansion of a function and neglecting the higher-order terms. The (first order) Taylor expansion series of a scalar function around a point x_0 looks like this:

$$f_{lin}(x) \approx f(x_0) + \left(\frac{df(x)}{dx}\right)_{x_0} (x - x_0) = a + bx \tag{5.57}$$

Vector functions can be linearised in a similar way, yielding:

$$\underline{\mathbf{f}}_{lin}(\underline{\mathbf{x}}) \approx \underline{\mathbf{f}}(\underline{\mathbf{x}}_0) + \underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}_0)\,(\underline{\mathbf{x}} - \underline{\mathbf{x}}_0) \tag{5.58}$$

where $\underline{\underline{\mathbf{J}}}$ is the so-called Jacobian of $\underline{\mathbf{f}}$. The Jacobian is a matrix with the same number of rows as $\underline{\mathbf{f}}$ and with the same number columns as there are elements in $\underline{\mathbf{x}}$. Each element (i,j) is calculated as:

$$\mathbf{J}_{i,j} = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} \tag{5.59}$$

i.e. the i'th function differentiated with respect to the j'th variable. To evaluate the Jacobian in a given point, $\underline{\mathbf{x_0}}$, one simple inserts the values of $\underline{\mathbf{x}}$, in the analytical Jacobian. For example, if

$$\underline{\mathbf{f}}(\underline{\mathbf{x}}) = \begin{pmatrix} f(x_1, x_2) \\ f(x_2) \end{pmatrix} = \begin{pmatrix} x_1 x_2 + 2 \\ \sqrt{x_2} \end{pmatrix} \tag{5.60}$$

then the Jacobian is

$$\mathbf{J}_{1,1} = \frac{\partial f_1}{\partial x_1} = x_2, \quad \mathbf{J}_{1,2} = \frac{\partial f_1}{\partial x_2} = x_1 \tag{5.61}$$

$$\mathbf{J}_{2,1} = \frac{\partial f_2}{\partial x_1} = 0, \quad \mathbf{J}_{2,2} = \frac{\partial f_2}{\partial x_2} = \frac{1}{2\sqrt{x_2}} \tag{5.62}$$

So a linear approximation to $\underline{\mathbf{f}}(\underline{\mathbf{x}})$ in the point $\underline{\mathbf{x}}_0 = \begin{pmatrix} x_{1,0} \\ x_{2,0} \end{pmatrix}$is

$$\begin{aligned} \underline{\mathbf{f}}_{lin}(\underline{\mathbf{x}}) &\approx \underline{\mathbf{f}}(\underline{\mathbf{x}}_0) + \underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}_0)\,(\underline{\mathbf{x}} - \underline{\mathbf{x}}_0) \\ &= \begin{pmatrix} x_{1,0}x_{2,0} + 2 \\ \sqrt{x_{2,0}} \end{pmatrix} + \begin{pmatrix} x_{2,0} & x_{1,0} \\ 0 & \frac{1}{2\sqrt{x_{2,0}}} \end{pmatrix} \begin{pmatrix} x_1 - x_{1,0} \\ x_2 - x_{2,0} \end{pmatrix} \\ &= \begin{pmatrix} x_{2,0}x_1 + x_{1,0}x_2 - x_{1,0}x_{2,0} + 2 \\ \frac{1}{2}\sqrt{x_{2,0}} + \frac{1}{2\sqrt{x_{2,0}}} x_2 \end{pmatrix} \end{aligned} \tag{5.63}$$

Linearisation of vector functions is a very common technique and is used in many engineering activities.

5.5.2 State Space Description for Linear Models

A state space representation casts the model of the process into two equations, the state equation which describes the dynamic behaviour of the system in terms of the internal states and the measurement equation which describes the static combination of the internal states and the inputs to measured quantities. For linear, time-constant processes the equations take the following form:

$$\begin{array}{rl} \text{the state equation:} & \underline{\dot{\mathbf{x}}} = \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{x}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{u}} \\ \text{the measurement equation:} & \underline{\mathbf{y}} = \underline{\underline{\mathbf{C}}}\,\underline{\mathbf{x}} + \underline{\underline{\mathbf{D}}}\,\underline{\mathbf{u}} \end{array} \tag{5.64}$$

With,

$\underline{\underline{\mathbf{A}}}$	::	System matrix	$[n \times n]$
$\underline{\underline{\mathbf{B}}}$	::	Input or control matrix	$[n \times m]$
$\underline{\underline{\mathbf{C}}}$	::	Output or measurement matrix	$[p \times n]$
$\underline{\underline{\mathbf{D}}}$	::	Direct coupling matrix	$[p \times m]$
$\underline{\mathbf{x}}$	::	State variable vector	$[n]$
$\underline{\mathbf{u}}$	::	Input vector	$[m]$
$\underline{\mathbf{y}}$	::	Output vector	$[p]$
n	::	Order of the system	

It should be noted that the here defined variables are not the same as the ones defined in the previous paragraphs.

State space representations are not unique. Any representation can be chosen dependent on the application. The proof is straightforward:

Proof. Proof for non-uniqueness of state space representations.

Define a new state $\underline{\tilde{\mathbf{x}}} = \underline{\underline{\mathbf{T}}}\,\underline{\mathbf{x}}$ where $\underline{\underline{\mathbf{T}}}$ is non-singular. Substituting this new state into the previous representation yields

$$\underline{\underline{\mathbf{T}}}\,\underline{\dot{\mathbf{x}}} = \underline{\underline{\mathbf{T}}}\,\underline{\underline{\mathbf{A}}}\,\underline{\mathbf{x}} + \underline{\underline{\mathbf{T}}}\,\underline{\underline{\mathbf{B}}}\,\underline{\mathbf{u}} \tag{5.65}$$

$$\underline{\widetilde{\dot{\mathbf{x}}}} = \underline{\underline{\mathbf{T}}}\,\underline{\underline{\mathbf{A}}}\,\underline{\underline{\mathbf{T}}}^{-1}\,\underline{\tilde{\mathbf{x}}} + \underline{\underline{\mathbf{T}}}\,\underline{\underline{\mathbf{B}}}\,\underline{\mathbf{u}} \tag{5.66}$$

$$= \underline{\underline{\tilde{\mathbf{A}}}}\,\underline{\tilde{\mathbf{x}}} + \underline{\underline{\tilde{\mathbf{B}}}}\,\underline{\mathbf{u}} \tag{5.67}$$

$$\underline{\mathbf{y}} = \underline{\underline{\mathbf{T}}}^{-1}\,\underline{\underline{\mathbf{C}}}\,\underline{\mathbf{x}} + \underline{\underline{\mathbf{D}}}\,\underline{\mathbf{u}} \tag{5.68}$$

$$\underline{\mathbf{y}} = \underline{\underline{\tilde{\mathbf{C}}}}\,\underline{\tilde{\mathbf{x}}} + \underline{\underline{\mathbf{D}}}\,\underline{\mathbf{u}} \tag{5.69}$$

which has the same input-output behaviour than the original representation. ■

An important reason to use state space representations for models is the fact that a lot of standard methods for analysing such models have been developed.

This section builds a bridge between the widely used state space description for linear models and linear models as derived with our modelling methodology. There is some overlap in used notations in both descriptions, but this should not give rise to any problems. Whenever a symbol relates to a state space description, it will carry the superscript '$*$' from here on.

In the previous paragraphs we learned that a model designer only has to supply 3 kinds of algebraic relations, namely the system equations, which provide a mapping for the primary states to the secondary states of the systems, connection equations, which define the flow rates and reaction equations, which define the reaction rates. Constraints will not be considered in this paragraph.

As derived in section 5.3, the balance equations can be written as:

$$\underline{\dot{\mathbf{x}}} = \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}} \tag{5.70}$$

The definitions of the flow ($\underline{\mathbf{z}}$) and reaction rates ($\underline{\mathbf{r}}$) are usually given as explicit relations. The flows are either manipulatable inputs ($\underline{\mathbf{u}}$) to the process or a function of the state of the connected systems:

$$\underline{\mathbf{z}} = \underline{\underline{\mathbf{C}}}_1\,\underline{\mathbf{y}} + \underline{\underline{\mathbf{D}}}\,\underline{\mathbf{u}} \tag{5.71}$$

$$\underline{\mathbf{r}} = \underline{\underline{\mathbf{C}}}_2\,\underline{\mathbf{y}} \tag{5.72}$$

The mapping from the primary state ($\underline{\mathbf{x}}$) to the secondary state ($\underline{\mathbf{y}}$) in the linear case has the following form:

$$\underline{\underline{\mathbf{E}}}\,\underline{\mathbf{y}} = \underline{\underline{\mathbf{F}}}\,\underline{\mathbf{x}} \tag{5.73}$$

and since each equation has to define a new variable, the matrix $\underline{\underline{\mathbf{E}}}$ is square and non-singular (i.e. invertible).

The link to the state space notation is now easily made:

$$\underline{\mathbf{y}} = \underline{\underline{\mathbf{E}}}^{-1}\,\underline{\underline{\mathbf{F}}}\,\underline{\mathbf{x}} \tag{5.74}$$

$$\underline{\mathbf{z}} = \underline{\underline{\mathbf{C}}}_1\,\underline{\underline{\mathbf{E}}}^{-1}\,\underline{\underline{\mathbf{F}}}\,\underline{\mathbf{x}} + \underline{\underline{\mathbf{D}}}\,\underline{\mathbf{u}} \tag{5.75}$$

$$\underline{\mathbf{r}} = \underline{\underline{\mathbf{C}}}_2\,\underline{\underline{\mathbf{E}}}^{-1}\,\underline{\underline{\mathbf{F}}}\,\underline{\mathbf{x}} \tag{5.76}$$

$$\underline{\dot{\mathbf{x}}} = \left(\underline{\underline{\mathbf{A}}}\,\underline{\underline{\mathbf{C}}}_1 + \underline{\underline{\mathbf{B}}}\,\underline{\underline{\mathbf{C}}}_2\right)\underline{\underline{\mathbf{E}}}^{-1}\,\underline{\underline{\mathbf{F}}}\,\underline{\mathbf{x}} + \underline{\underline{\mathbf{A}}}\,\underline{\underline{\mathbf{D}}}\,\underline{\mathbf{u}} \tag{5.77}$$

$$= \underline{\underline{\mathbf{A}}}^*\,\underline{\mathbf{x}} + \underline{\underline{\mathbf{B}}}^*\,\underline{\mathbf{u}} \tag{5.78}$$

$$\underline{\mathbf{y}}^* = \underline{\underline{\Omega}}\,\underline{\underline{\mathbf{E}}}^{-1}\,\underline{\underline{\mathbf{F}}}\,\underline{\mathbf{x}} \tag{5.79}$$

$$= \underline{\underline{\mathbf{C}}}^*\,\underline{\mathbf{x}} \tag{5.80}$$

Since $\underline{\mathbf{y}}^*$ represents the output vector, the matrix $\underline{\underline{\Omega}}$ is a selection matrix selecting the desired output from the set of available state variables.

The above derivation presumes that the flows are directly written as a function of the secondary state variables and that all defined secondary state variables are needed for the definition of the flow and reaction rates. This is, in general, not the case. Often, secondary flow variables ($\underline{\mathbf{z}}_2$), which do not appear directly in the balance equations, are defined and are linked to the primary flow variables via an algebraic expression. Also, not all defined secondary state variables are needed for the calculation of the flow and reaction rates. Taking an inverse of a large matrix costs a lot of computational effort. So, in order to reduce the computational effort, the secondary state vector of each system is divided into three subvectors:

$$\underline{\mathbf{y}} = \begin{bmatrix} \underline{\mathbf{y}}_1 \\ \underline{\mathbf{y}}_2 \\ \underline{\mathbf{y}}_3 \end{bmatrix} \tag{5.81}$$

in which $\underline{\mathbf{y}}_2$ is a vector of secondary states that are actually needed to calculate flow and reaction rates. $\underline{\mathbf{y}}_1$ are secondary states that are only needed for the calculation of $\underline{\mathbf{y}}_2$. $\underline{\mathbf{y}}_3$ are states that are not needed for the calculation. They only provide output information and do not have to be calculated during a computation run, but can be calculated afterwards. The mapping of the secondary state now takes the form:

$$\underline{\underline{\mathbf{E}}}_{11}\,\underline{\mathbf{y}}_1 = \underline{\underline{\mathbf{F}}}_1\,\underline{\mathbf{x}} \tag{5.82}$$

$$\underline{\underline{\mathbf{E}}}_{22}\,\underline{\mathbf{y}}_2 = \underline{\underline{\mathbf{E}}}_{\mathbf{21}}\,\underline{\mathbf{y}}_{\mathbf{1}} + \underline{\underline{\mathbf{F}}}_2\,\underline{\mathbf{x}} \tag{5.83}$$

$$\underline{\underline{\mathbf{E}}}_{33}\,\underline{\mathbf{y}}_3 = \underline{\underline{\mathbf{E}}}_{\mathbf{31}}\,\underline{\mathbf{y}}_{\mathbf{1}} + \underline{\underline{\mathbf{E}}}_{\mathbf{32}}\,\underline{\mathbf{y}}_{\mathbf{2}} + \underline{\underline{\mathbf{F}}}_3\,\underline{\mathbf{x}} \tag{5.84}$$

and the flow and reaction rate are defined as:

$$\underline{\mathbf{z}} = \underline{\underline{\mathbf{C}}}_1\,\underline{\mathbf{y}}_2 + \underline{\underline{\mathbf{Z}}}\,\underline{\mathbf{z}}_2 + \underline{\underline{\mathbf{D}}}_1\,\underline{\mathbf{u}} \tag{5.85}$$

$$\underline{\mathbf{z}}_2 = \underline{\underline{\mathbf{C}}}_2\,\underline{\mathbf{y}}_2 + \underline{\underline{\mathbf{D}}}_2\,\underline{\mathbf{u}} \tag{5.86}$$

$$\underline{\mathbf{r}} = \underline{\underline{\mathbf{C}}}_3\,\underline{\mathbf{y}}_2 \tag{5.87}$$

For each system the vector $\underline{\mathbf{y}}_2$ can be written explicitly:

$$\underline{\mathbf{y}}_1 = \underline{\underline{\mathbf{E}}}_{\mathbf{11}}^{-\mathbf{1}}\,\underline{\underline{\mathbf{F}}}_1\,\underline{\mathbf{x}} \tag{5.88}$$

$$\underline{\mathbf{y}}_2 = \underline{\underline{\mathbf{E}}}_{\mathbf{22}}^{-\mathbf{1}}\left(\underline{\underline{\mathbf{E}}}_{\mathbf{21}}\,\underline{\underline{\mathbf{E}}}_{\mathbf{11}}^{-\mathbf{1}}\,\underline{\underline{\mathbf{F}}}_{\mathbf{1}} + \underline{\underline{\mathbf{F}}}_2\right)\underline{\mathbf{x}} \tag{5.89}$$

If the $\underline{\mathbf{y}}_2$ vectors of each system are stacked together we get the $\underline{\mathbf{y}}_2$ of the

complete process. The following state space representation results:

$$\begin{aligned}\dot{\underline{\mathbf{x}}} &= \left(\underline{\underline{\mathbf{A}}}\,\underline{\underline{\mathbf{C}}}_{\mathbf{1}} + \underline{\underline{\mathbf{A}}}\,\underline{\underline{\mathbf{Z}}}\,\underline{\underline{\mathbf{C}}}_{\mathbf{2}} + \underline{\underline{\mathbf{B}}}\,\underline{\underline{\mathbf{C}}}_{3}\right)\underline{\underline{\mathbf{E}}}_{\mathbf{22}}^{-\mathbf{1}}\left(\underline{\underline{\mathbf{E}}}_{\mathbf{21}}\,\underline{\underline{\mathbf{E}}}_{\mathbf{11}}^{-\mathbf{1}}\,\underline{\underline{\mathbf{F}}}_{\mathbf{1}} + \underline{\underline{\mathbf{F}}}_{2}\right)\underline{\mathbf{x}} \\ &\quad + \underline{\underline{\mathbf{A}}}\left(\underline{\underline{\mathbf{D}}}_{1} + \underline{\underline{\mathbf{Z}}}\,\underline{\underline{\mathbf{D}}}_{2}\right)\underline{\mathbf{u}} \qquad (5.90)\end{aligned}$$

$$\underline{\mathbf{y}}^{*} = \underline{\underline{\Omega}}\,\underline{\mathbf{y}} = \underline{\underline{\Omega}}\begin{bmatrix}\underline{\mathbf{y}}_1\\ \underline{\mathbf{y}}_2\\ \underline{\mathbf{y}}_3\end{bmatrix} \qquad (5.91)$$

This may look difficult and cumbersome, but since these are all straightforward linear operations, they can easily be automated by a computer program and the only thing a model designer has to do is to provide the necessary equations.

Example 5.4: ***State Space Model of a Simple Tank***

Consider a tank which has an inflow $\hat{\mathbf{m}}_1$ *and an outflow* $\hat{\mathbf{m}}_2$*. The height in the tank has to be controlled with the inlet flow and the volumetric flow rate of the outlet flow is linear with the liquid heigth in the tank.*

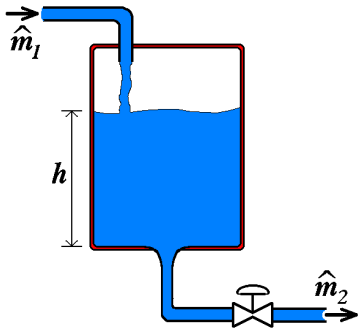

Figure 5.2: *Tank with inlet and outlet*

If ideal mixing and constant density are assumed, a simple model can easily be derived:

Balance Equation:

$$\dot{m} = \hat{m}_1 - \hat{m}_2 \quad or \quad \dot{m} = \begin{bmatrix} 1 & -1 \end{bmatrix}\begin{bmatrix}\hat{m}_1\\ \hat{m}_2\end{bmatrix} \qquad (5.92)$$

Flow Rate Equations:

$$\hat{m}_1 = \rho\hat{V}_1 \qquad (5.93)$$

$$\hat{m}_2 = \rho\hat{V}_2 \qquad (5.94)$$

$$\hat{V}_2 = \alpha h \qquad (5.95)$$

System Equations:

$$m = \rho V \tag{5.96}$$
$$h = \frac{V}{A} \tag{5.97}$$

From this we can derive the above mentioned vectors and matrices:

$$\underline{\mathbf{x}} = m \quad \underline{\mathbf{y}}_1 = V \quad \underline{\mathbf{y}}_2 = h \quad \underline{\mathbf{z}} = \begin{bmatrix} \hat{m}_1 \\ \hat{m}_2 \end{bmatrix}$$
$$\underline{\mathbf{z}}_2 = \hat{\mathbf{V}}_\mathbf{2} \quad \underline{\mathbf{u}} = \hat{V}_1 \quad \underline{\underline{\mathbf{C}}}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \underline{\underline{\mathbf{C}}}_2 = \alpha$$
$$\underline{\underline{\mathbf{E}}}_{11} = \rho \quad \underline{\underline{\mathbf{E}}}_{22} = 1 \quad \underline{\underline{\mathbf{E}}}_{21} = \frac{1}{A} \quad \underline{\underline{\mathbf{F}}}_1 = 1$$
$$\underline{\underline{\mathbf{F}}}_2 = 0 \quad \underline{\underline{\mathbf{Z}}} = \begin{bmatrix} 0 \\ \rho \end{bmatrix} \quad \underline{\underline{\mathbf{D}}}_1 = \begin{bmatrix} \rho \\ 0 \end{bmatrix} \quad \underline{\underline{\mathbf{D}}}_2 = 0$$

Which results in the state space matrices:

$$\underline{\underline{\mathbf{A}}}^* = -\frac{\alpha}{A} \quad \underline{\underline{\mathbf{B}}}^* = \rho \quad \underline{\underline{\mathbf{C}}}^* = \frac{1}{\rho A}$$

□

5.5.3 State Space Description for Linearised Models

When a state space description is desired for a linearised model, a small problem arises. As derived in section 5.5.1, a linearised equation is of the form:

$$f_{lin}(x) = a + bx \tag{5.98}$$

and since such an equation contains a constant (a) it cannot be directly used for the transformations described in the previous paragraph. One way to overcome this problem is to introduce the concept of the *deviation variable*. Deviation variables represent the deviation of the variables around a steady state working point. So if a process is at its steady state, the deviation variables will be zero. The deviation variable x' of a variable x is defined as:

$$x' = x - x_s \tag{5.99}$$

The algebraic relations of a process model can be written as:

$$\underline{\mathbf{0}} = \underline{\mathbf{f}}(\underline{\mathbf{x}}) \tag{5.100}$$

where $\underline{\mathbf{x}}$ here represents all the variables that are present in the algebraic relations. A linearisation around the steady state values $\underline{\mathbf{x}}_s$ would result in:

$$\underline{\mathbf{0}} = \underline{\mathbf{f}}(\underline{\mathbf{x}}_s) + \underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}_s)\,(\underline{\mathbf{x}} - \underline{\mathbf{x}}_s) \tag{5.101}$$

If we now substract the steady state conditions $\underline{\mathbf{0}} = \underline{\mathbf{f}}(\underline{\mathbf{x}}_s)$ from this relation and introduce the deviation variables $\underline{\mathbf{x}}' = \underline{\mathbf{x}} - \underline{\mathbf{x}}_s$ we get:

$$\underline{\mathbf{0}} = \underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}_s)\underline{\mathbf{x}}' \tag{5.102}$$

When we also substract the steady state conditions from the balance equations we ge the following result:

$$\begin{aligned}
\dot{\underline{\mathbf{x}}}' &= \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}}' + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}}' && (5.103)\\
\underline{\mathbf{z}}' &= \underline{\underline{\mathbf{C}}}_1\,\underline{\mathbf{y}}' + \underline{\underline{\mathbf{D}}}\,\underline{\mathbf{u}}' && (5.104)\\
\underline{\mathbf{r}}' &= \underline{\underline{\mathbf{C}}}_2\,\underline{\mathbf{y}}' && (5.105)\\
\underline{\underline{\mathbf{E}}}\,\underline{\mathbf{y}}' &= \underline{\underline{\mathbf{F}}}\,\underline{\mathbf{x}}' && (5.106)
\end{aligned}$$

Example 5.5: ***Linearised State Space Model of a Simple Tank***

Consider the previous example. Now the volumetric flow of the outlet is defined as:

$$\hat{V}_2 = \alpha\sqrt{h} \tag{5.107}$$

The linearised model then simply becomes:

$$\dot{m}' = \hat{m}_1' - \hat{m}_2' \tag{5.108}$$

$$\begin{aligned}
\hat{m}_1' &= \rho\hat{V}_1' && (5.109)\\
\hat{m}_2' &= \rho\hat{V}_2' && (5.110)\\
\hat{V}_2' &= \frac{\alpha}{2\sqrt{h_s}}h' && (5.111)
\end{aligned}$$

$$\begin{aligned}
m' &= \rho V' && (5.112)\\
h' &= \frac{V'}{A} && (5.113)
\end{aligned}$$

The state space representation of this model can be generated using the approach mentioned in the previous paragraph.

□

If one chooses not to linearise around a steady state working point but around any working point of the process, the solutions becomes slightly different. Linearising the model of the process around a working point results in the following representation:

$$\underline{\dot{\mathbf{x}}} = \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}} \tag{5.114}$$

$$\underline{\mathbf{z}} = \underline{\underline{\mathbf{C}}}_1\,\underline{\mathbf{y}} + \underline{\underline{\mathbf{D}}}\,\underline{\mathbf{u}} + \underline{\underline{\mathbf{K}}}_1 \tag{5.115}$$

$$\underline{\mathbf{r}} = \underline{\underline{\mathbf{C}}}_2\,\underline{\mathbf{y}} + \underline{\underline{\mathbf{K}}}_2 \tag{5.116}$$

$$\underline{\underline{\mathbf{E}}}\,\underline{\mathbf{y}} = \underline{\underline{\mathbf{F}}}\,\underline{\mathbf{x}} + \underline{\underline{\mathbf{K}}}_3 \tag{5.117}$$

Where $\underline{\underline{\mathbf{K}}}_1$, $\underline{\underline{\mathbf{K}}}_2$ and $\underline{\underline{\mathbf{K}}}_3$ are constant matrices that appear due to the linearisation of the algebraic equations. The differential equations are not extended with a constant matrix because these equations are already linear. To get the linear state space representation we get:

$$\underline{\mathbf{y}} = \underline{\underline{\mathbf{E}}}^{-1}\left(\underline{\underline{\mathbf{F}}}\,\underline{\mathbf{x}} + \underline{\underline{\mathbf{K}}}_3\right) \tag{5.118}$$

$$\begin{aligned}\underline{\dot{\mathbf{x}}} &= \left(\underline{\underline{\mathbf{A}}}\,\underline{\underline{\mathbf{C}}}_1 + \underline{\underline{\mathbf{B}}}\,\underline{\underline{\mathbf{C}}}_2\right)\underline{\underline{\mathbf{E}}}^{-1}\,\underline{\underline{\mathbf{F}}}\,\underline{\mathbf{x}} + \underline{\underline{\mathbf{A}}}\,\underline{\underline{\mathbf{D}}}\,\underline{\mathbf{u}} \\ &\quad + \left(\underline{\underline{\mathbf{A}}}\,\underline{\underline{\mathbf{C}}}_1 + \underline{\underline{\mathbf{B}}}\,\underline{\underline{\mathbf{C}}}_2\right)\underline{\underline{\mathbf{E}}}^{-1}\underline{\underline{\mathbf{K}}}_3 + \underline{\underline{\mathbf{A}}}\,\underline{\underline{\mathbf{K}}}_1 + \underline{\underline{\mathbf{B}}}\,\underline{\underline{\mathbf{K}}}_2 \qquad (5.119)\\ &= \underline{\underline{\mathbf{A}}}^*\,\underline{\mathbf{x}} + \underline{\underline{\mathbf{B}}}^*\,\underline{\mathbf{u}} + \underline{\underline{\mathbf{K}}} \qquad (5.120)\end{aligned}$$

If we now consider the matrix $\underline{\underline{\mathbf{K}}}$ to be an "input" to the system, we get the following result:

$$\underline{\dot{\mathbf{x}}} = \underline{\underline{\mathbf{A}}}^*\,\underline{\mathbf{x}} + \underline{\underline{\tilde{\mathbf{B}}}}^*\,\underline{\tilde{\mathbf{u}}} \tag{5.121}$$

5.6 Substitution, yes or no?

It is often seen that model designers insist on eliminating the extensive variables from the model equations. The main reason that is brought up for this preference to write a model that does not involve the extensive variables is that often only the evolution of the application variables is of interest. Also, the transfer laws and kinetic laws are usually given in terms of intensive state variables. Therefore most model designers think they *must* transform the accumulation terms and perform a so-called state variable transformation. Most textbooks which cover modelling also perform these transformations, often even without mentioning why. But are these, often cumbersome, state variable transformations necessary to solve the considered problems?

In most cases, the transformations are *not* necessary. There are several reasons to consider the differential algebraic equations (DAEs) (5.56) directly,

rather than to try to rewrite them as a set of ordinary differential equations (ODEs) (Brenan *et al.* 1996): First, when modelling physical processes, the model takes the form of a DAE, depicting a collection of relationships between variables of interest and some of their derivatives. These relationships may be generated by a modelling program (such as the Modeller). In that case, or in the case of highly nonlinear models, it may be time consuming or even impossible to obtain an explicit model. Computational causality is not a physical phenomenon, so it is rather inconvenient if a model designer has to determine the (numerically) correct causality of the equations (Cellier and Elmqvist 1993)(Thevenon and Flaus 2000). Also, reformulation of the model equations tends to reduce the expressiveness (Fábián 1999). Furthermore, if the original DAE can be solved directly it becomes easier to interface modelling software directly with design software. Finally, reformulation slows down the development of complex process models, since it must be repeated each time the model is altered, and therefore it is easier to solve the DAE directly.

These advantages enable researchers to focus their attention on the physical problem of interest. There are also numerical reasons for considering DAEs (Brenan *et al.* 1996). The change to explicit form, even if possible, can destroy sparsity and prevent the exploitation of system structure.

Small advantages of transforming the model to ODE form can be that for (very) small systems an analytical solution is available and that sometimes less information of physical properties is needed when substitutions are being made (sometimes, some of the parameters can be removed from the system equations when substitutions are made). Another advantage could be that, by doing substitutions, some primary state variables are removed from the model description which could make the code faster, because less variables have to be solved. In general, though, these advantages do not outweigh the disadvantages.

If one does want to perform substitutions, I recommend that these are done at the very end of the model development and not, as is generally seen, as soon as possible. Postponing the substitutions as long as possible gives a much better insight in the model structure during model development.

5.7 Control

Control is added in the last step of our modelling methodology and can be seen "seperately" from the previous steps, since for most models that are meant for control, first the model without the controllers is constructed. The control part is usually added in a later step and can be superimposed on the model without affecting the previous modelling steps. Controllers process measurement and setpoint information. The first taken from the process state,

the latter being an input to the process and determining its desired behaviour. Control generates (information) signals as output, which in general affect the flow of extensive quantities inside the process.

Adding a controller to a process has only one goal, namely to modify the dynamic behaviour to the process to be controlled, with the objective of imposing a desired behaviour. The controller may be introduced for different reasons. The controller may force the process to follow a trajectory (the servo or steering problem) or the controller may serve the purpose of compensating for undesired effects of the environment has on the process (disturbance rejection).

Whatever the control objectives are, there is always a need to monitor the performance of the process that needs to be controlled. This is done by measuring the values of certain process variables (such as concentrations, fluid height, temperatures, flow rates, pressures, etc.). The measurements are processed by the controller, which then steers some input variables in order to control the process. Usually a process has a number of available input variables which can be adjusted freely. Which ones are selected to use as input (i.e. manipulatable) variables is a crucial question, as the choice will affect the quality of the control actions we take (Stephanopoulos 1984).

Clearly, there is a flow of information to and from the controller and therefore two new elements need to be introduced, namely the *information system* (i.e. the controller itself) and the *information connection* (through which the information "flows": The target object of the information connection reads variables of the origin object).

The input (i.e. the measurement) to the information system can, in principle, be any time dependent variable of the process. Consequently, the information connection that provides the input to the information system can be connected to any system (either lumped, distributed, source, sink, steady state or composite), any connection or any other information system (cascade control, providing "new" setpoints) of the physical topology of the process. Most of the time, however, the measurements are closely related to the primary states of a process and the information will come from lumped systems.

The controller receives the measurement information and decides what action should be taken. The variables that can be manipulated by the controller are usually some connection characteristics, such as the position of a valve. The control valve is the most frequently encountered final control element, but not the only one. Other typical final control elements for chemical processes are: relay switches (providing on-off control), variable-speed pumps and variable-speed compressors (Stephanopoulos 1984).

From a physical point of view, the variables that can be manipulated will

always be connection variables and never system or reaction variables, since a controller cannot change the state of a system directly. It can only manipulate the flow of extensive quantities, which in turn affect the state of the two interconnected systems. Sometimes, however, either for convenience or as a simplification, a controller is made to directly affect some secondary state variables of a source or sink system. A controller that directly influences the temperature of a heat stream, for example, is physically impossible but often seen in modelling (especially if the temperature of the stream can be adjusted relatively quickly compared to the dynamics of the controlled system). The output of an information system can therefore be connected to any connection, any source or sink system or any other information system.

The possible information flows are summerised in figure 5.3.

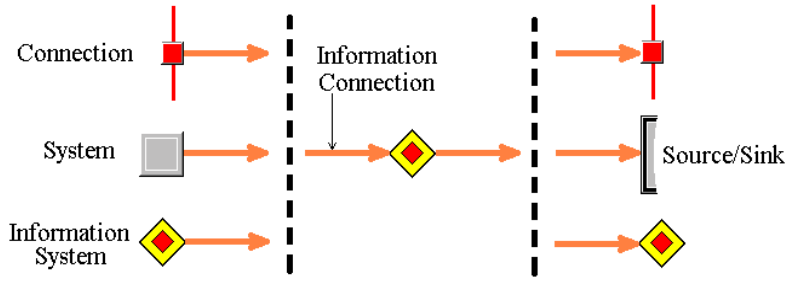

Figure 5.3: *Possible information flows to and from information systems.*

Controllers are dynamic elements. The controller equations can consist of algebraic, differential and integral equations and since the controller normally has a state, the order of the overall model is increased by adding control elements to the physical topology of the process.

Further discussion of control, which obviously has a lot more to it than the few things that are mentioned in this brief section, is not considered to be relevant for pursuing our goal of formulating a systematic modelling methodology. It is left to the reader to find out more about the numerous subjects on control, such as feedback control, feedforward control, inferential control, PID, LQR, LQG, , H_2, H_∞, state feedback, cascade control, etc.. All these types of controllers can be implemented in an information system, as long as they are realisable and presented in a DAE form.

Chapter 6

Assumptions: Problems and Solutions

Constraints and assumptions describe all kinds of algebraic relationships between process quantities which have to hold at any time. A volume constraint, for example, restricts the volume of a simple system or the sum of volumes of a set of systems. Assumptions may result in models which are cumbersome to solve. Potential problems can very often be avoided at an early stage of the model development by keeping clear of certain assumptions or by directly dealing with the cause of the potential problems.

This chapter addresses the structure and representation of modelling assumptions. One can distinguish between several types of assumptions: Structural assumptions (i.e. the construction of the physical topology), order of magnitude assumptions (very small versus very large) and assumptions on relative time scale (very slow versus very fast). These assumptions are usually introduced with a goal, namely to simplify the description of the behaviour by neglecting what is considered insignificant in the view of the application one has in mind for the model. Whilst indeed such assumptions do simplify the description, they are also the source of numerous problems, such as index problems, which make the solving of the equations very difficult.

In this chapter the assumptions that can lead to computational problems are classified. Next, each class is analysed on how the assumptions affect the solvability of the model equations. The proposed solutions of these problems correspond to the Simple and Full Index Reduction Algorithm as presented in (Moe 1995). Furthermore, the impact of making steady-state assumptions and the effects of introducing events in the model description will be discussed.

6.1 High-index Models

Dynamic process models, as derived with our modelling methodology, consist of differential and algebraic equations (DAEs). Unfortunately, most engineers have little knowledge of the theory of DAEs, since most of the calculations that have to be performed during education are steady-state simulations. If dynamics are considered, a mathematical description of the (usually very simple) model is derived in the form of ordinary differential equations (ODEs).

One of the major advantages of writing a model in DAE form as opposed to ODE form, is that a modeller does not have to perform a set of often cumbersome mathematical manipulations, such as substitution and symbolic differentiation (Cellier 1991)(Brenan *et al.* 1996)(Bujakiewics 1994). A good book describing the basic mathematical theory of DAEs and presenting and analysing numerical methods is (Brenan *et al.* 1996). Our work is concerned with a specific type of DAEs, so the needed theory is limited.

According to (Moe 1995), dynamic process models can be divided into *low index models* (index 0 and 1) and *high index models* (index two and higher and some index one models). The *differential index* or *index* is a measure of the problems related to initialisation and integration of dynamic process models. The problems related to solve a dynamic process model increase with increasing index. (Brenan *et al.* 1996) give the following general definition for the index of nonlinear DAEs:

DEFINITION 6.1.1 *The minimum number of times that all or part of* $F(t, y, \dot{y}) = 0$ *must be differentiated with respect to* t *in order to determine* $\dot{y}$ *as a continuous function of* y, t *(i.e.:* $\dot{y} = G(t, y)$*), is the* index *of the DAE*

They stress that it is not recommended to perform this series of differentiations as a general solution procedure for DAEs. Rather the number of such differentiation steps that would be required in theory turns out to be an important quantity in understanding the behaviour of numerical methods.

According to the definition an ODE (either explicit or implicit) has index zero. DAEs with index zero and one are generally much simpler to understand (and much simpler to solve) than DAEs with index two or higher.

With our modelling method we strive to produce semi-explicit index one models, since these can be easily used for simulation by any DAE-solver. Higher index models must either be simulated directly by a special integrator which tackles high index DAEs, or be transformed to semi-explicit index one and integrated. Two simple tests that guarantee structurally semi-explicit index one models are (Moe 1995):

- All *algebraic* variables must be present in the algebraic equations.

- It must be possible to assign each algebraic equation to an *algebraic* variable. Assignment of all equations and variables must be possible.

If one or several algebraic variables are absent from the algebraic equations, then the model index is two or higher.

Example 6.1: ***Index Test***

Consider the following simple linear models A and B:

$$\begin{array}{ll} \textbf{A} & \textbf{B} \\ \dot{x}_1 = y_1 - y_2 & \dot{x}_1 = y_1 - y_2 \\ \dot{x}_2 = y_2 & \dot{x}_2 = y_2 \\ y_1 = 10 & y_1 = 10 \\ y_2 = 5x_1 - x_2 & 0 \ = 5x_1 - x_2 \end{array} \tag{6.1}$$

Both models look very similar. They both have differential variables x_1 and x_2 and algebraic variables y_1 and y_2. Model A has index one, because all the algebraic variables are present in the algebraic relations and an algebraic equation can be assigned to each of the algebraic variables.

The algebraic variable y_2 of model B only appears in the differential equations and is not present in the algebraic equations of this model. Hence, the index of this model exceeds unity. In contrast to model A, the initial values of the differential variables of model B may not be chosen independently and the model, in this form, cannot be integrated by most available integrators. Before the model can be integrated, the index has to be reduced either by the solver or the modeller himself [1]*. Methods for reducing the index of a model are discussed in section 6.3.*

□

6.2 Assumptions Leading to High-index Models

Many high-index problems are caused by a model purpose that is not carefully considered, because the modeller does not want to include some of the rapid dynamics in the model, by assumptions that may not be essential or because the modeller wants to include certain variables in the model (Moe 1995). Our

[1] There are some numerical methods available that can solve certain types of higher index models directly (Petzold and P.Lötstedt 1986)(Brenan *et al.* 1996), but these are far from straightforward.

modelling method forces a model designer to be more aware of the assumptions he makes. Therefore, potential high index model formulations can be detected and/or avoided at an early stage of the model development. Knowing the causes of high index formulations helps the model designer in carefully considering the modelling purpose and the assumptions he wants to make.

Assumptions that impose direct or indirect constraints on the differential variables lead to high index models. But a modeller cannot simply impose some constraints on the differential variables. A constraint is always imposed by some "driving force" (i.e. a flow or reaction, since these are the only "forces" that appear in the differential equations), which forces the differential variables to adhere to the constraint. This means that *instead of* giving a description for the rate, a flow or reaction remains "unmodelled" and a (direct or indirect) constraint on the differential variables is given.

There can be several reasons why a modeller wants to introduce assumptions:

- Only slow dynamics of the process are of interest. In this case the rapid dynamics can be neglected.
- Difficulties in finding reliable rate equations may force a modeller to make quasi steady-state assumptions.
- In order to perform model reduction, simplifying assumptions may be introduced.

In section 3.3 the term time scale was introduced. Modelling systems in a range of time scales, the capacity terms are chosen accordingly but also the transport and the production terms. For parts being outside of the time scale in which the dynamics are being modelled, a pseudo steady-state assumption is made. For example, (very) fast reactions - fast in the measure of the considered range of time scale - are assumed to reach the equilibrium (for all practical purposes) instantaneously, and very slow ones do not appreciatly occur and may be simply ignored. For the effects of small and large capacities, the singular perturbation theory is applicable (see section 6.4).

6.3 Index Reduction Algorithms

In section 5.3 it was derived that the differential variables (i.e. the primary state variables) are manipulated by the flows and reactions (see, for example, equation 5.49). In low index models these flows and reactions are either specified or expressed by rate equations. In high index models, constraints are

imposed on the primary state variables, either directly or indirectly. To make these variables satisfy the constraints and thus avoid inconsistent simulation results, the *unmodelled flows* and *unmodelled reactions* are forced to satisfy certain conditions.

From the point of view of the numerical solution, it is desirable for the DAE to have an index which is as small as possible (Brenan *et al.* 1996). Numerous methods have been proposed to achieve a reduction of the index by differentiating the equations that cause constraints on the primary state variables, e.g. (Gear and Petzold 1984), (Gear 1988), (Gear 1990), (Bachmann and Pallaske 1990), (Bachmann and Pallaske 1989), (Fábián and Rooda 2001), (Chung and Westerberg 1990), (Mattsson and Söderlind 1993), (Pantelides 1998). Differentiation is, however, not always necessary (Westerweele and Preisig 2000)(Moe 1995). Sometimes it is possible to prevent the high index from occurring by applying algebraic approaches that utilise modelling insight and respective algebraic procedures.

6.3.1 Simple Index Reduction Algorithm

The easiest way to reduce the index of a dynamic process model is to apply the Simple Index Reduction Method (Moe 1995). For convenience, the flows and reactions that appear in the balance equations are split up in a "normal" (subscript n) and an "unmodelled" (subscript u) part:

$$\underline{\dot{\mathbf{x}}} = \underline{\underline{\mathbf{A}}}_n \, \underline{\mathbf{z}}_n + \underline{\underline{\mathbf{B}}}_n \, \underline{\mathbf{r}}_n + \underline{\underline{\mathbf{A}}}_u \, \underline{\mathbf{z}}_u + \underline{\underline{\mathbf{B}}}_u \, \underline{\mathbf{r}}_u \tag{6.2}$$

This form is used to reduce the index of a model when certain assumption are being made. There are numerous situations in which this form is applicable. In general, these situations arise from mixing some "steady-state thinking" into a dynamical context (Weiss 2000). It is the case of the so-called quasi steady-state assumptions. For example, the assumption that a chemical reaction within a chemical process is constantly at equilibrium (i.e. an algebraic relation between concentrations is assumed to hold at all time). This is a quasi-equilibrium, since addition of reactants will move the equilibrium point of the reaction according to Le Chateliers principle, towards consumption of the added reactants. Examples of this type of assumptions can be found in (Moe *et al.* 1995), (Marquardt 1995) and (Pantelides 1998).

High index process models arise when unmodelled flows and/or reactions are used for describing the process. If these flows and reactions are of no interest, then the index problem can be resolved by eliminating the undefined terms ($\underline{\mathbf{z}}_u$ and $\underline{\mathbf{r}}_u$) by forming linear combinations of the balance equations. This can be achieved by multiplying equation 6.2 with a matrix $\underline{\underline{\Omega}}$, of which the rows

constitute a basis for the null-space of $\begin{pmatrix} \underline{\underline{\mathbf{A}}}_u & \underline{\underline{\mathbf{B}}}_u \end{pmatrix}$, i.e. $\underline{\underline{\Omega}}\begin{pmatrix} \underline{\underline{\mathbf{A}}}_u & \underline{\underline{\mathbf{B}}}_u \end{pmatrix} = \underline{\underline{\mathbf{0}}}$. This results in:

$$\begin{aligned} \underline{\underline{\Omega}}\,\dot{\underline{\mathbf{x}}} &= \underline{\underline{\Omega}}\,\underline{\underline{\mathbf{A}}}_n\,\underline{\mathbf{z}}_n + \underline{\underline{\Omega}}\,\underline{\underline{\mathbf{B}}}_n\,\underline{\mathbf{r}}_n + \underline{\underline{\Omega}}\begin{pmatrix} \underline{\underline{\mathbf{A}}}_u & \underline{\underline{\mathbf{B}}}_u \end{pmatrix}\begin{pmatrix} \underline{\mathbf{z}}_u \\ \underline{\mathbf{r}}_u \end{pmatrix} \\ &= \underline{\underline{\Omega}}\,\underline{\underline{\mathbf{A}}}_n\,\underline{\mathbf{z}}_n + \underline{\underline{\Omega}}\,\underline{\underline{\mathbf{B}}}_n\,\underline{\mathbf{r}}_n \end{aligned} \tag{6.3}$$

If we now define new variables:

$$\underline{\mathbf{x}}^* = \underline{\underline{\Omega}}\,\underline{\mathbf{x}} \tag{6.4}$$

we get:

$$\dot{\underline{\mathbf{x}}}^* = \underline{\underline{\Omega}}\,\underline{\underline{\mathbf{A}}}_n\,\underline{\mathbf{z}}_n + \underline{\underline{\Omega}}\,\underline{\underline{\mathbf{B}}}_n\,\underline{\mathbf{r}}_n \tag{6.5}$$

By multiplying the balance equation by $\underline{\underline{\Omega}}$ and introducing an algebraic relation for each unmodelled flow and reaction, we have reduced the order of the original system and eliminated the undefined transfer and production terms. So, for a model that has n balance equations (i.e. the order of $\underline{\mathbf{x}}$ is n) and in which k unmodelled flows and reactions are defined, the condensed representation of the model comprises $(n-k)$ differential equations and k algebraic equations, which together describe the dynamics of the model. (Note that for an unmodelled flow,which transports i species, i algebraic relations have to be defined). This Simple Index Reduction Method eliminates all the unmodelled terms from the model via simple algebraic manipulations. If the values of the removed variables are of interest, either post processing of the simulation results is required, or an algorithm, which does not eliminate these variables, must be applied (Moe 1995).

The result of the Simple Index Reduction Method is a semi-explicit DAE of index one and can be solved with standard algorithms. Chapter 10 will elaborate on this.

Example 6.2: ***CSTR With Equilibrium Reactions*** *(Westerweele and Preisig 2000)*

Consider a CSTR with 6 components A, B, C, D, E and F. The reactor has a constant volumetric in- and outflow $\widehat{V}$ and there are three reactions taking place in this reactor. Two of these reactions (the first and the third) are considered to equilibrium reactions:

$$\begin{array}{ll} A \rightleftarrows B + D & K_1 = \frac{c_B c_D}{c_A} \\ B \rightleftarrows C + D & \widetilde{\xi}_2 = (k_{2a} c_B - k_{2b} c_C c_D) V \\ D + E \rightleftarrows F & K_3 = c_D c_E \end{array} \tag{6.6}$$

Normally one would expect that the dimension of the state space of the system equals the number of components which are associated

with the system (when energy of the system is not modelled!) and that the model of the system would include 6 differential equations (one for each component). But since two of the occurring reactions are considered to be equilibrium reactions, two of those differential equations are replaced by algebraic relationships and the actual dimension of the state space of the system is reduced to four.

Unreduced balance equations:

$$\underline{\dot{\mathbf{n}}} = \underline{\underline{\Gamma}}_{\Sigma}\underline{\underline{\mathbf{A}}}_{k}\underline{\underline{\Gamma}}_{m}^{T}\underline{\hat{\mathbf{n}}} + \underline{\underline{\Gamma}}_{\Sigma}\underline{\underline{\mathbf{S}}}_{eq}\,\underline{\boldsymbol{\xi}}_{eq} + \underline{\underline{\Gamma}}_{\Sigma}\underline{\underline{\mathbf{S}}}_{r}\,\underline{\boldsymbol{\xi}}_{r} \tag{6.7}$$

in which:

$$\underline{\underline{\mathbf{S}}}_{eq}^{T} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 1 & -1 \\ 0 & -1 \\ 0 & 1 \end{bmatrix};\quad \underline{\underline{\mathbf{S}}}_{r}^{T} = \begin{bmatrix} 0 \\ -1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix};\quad \widetilde{\xi}_{r} = \widetilde{\xi}_{2};\quad \underline{\underline{\Gamma}}_{\Sigma} = \underline{\underline{\Gamma}}_{m} = \underline{\underline{\mathbf{I}}};$$

$$\underline{\underline{\mathbf{A}}} = \begin{bmatrix} \underline{\underline{\mathbf{I}}} & -\underline{\underline{\mathbf{I}}} \end{bmatrix};\quad \widehat{\underline{\underline{\mathbf{n}}}}^{T} = \begin{bmatrix} \widehat{\underline{\mathbf{n}}}_1 & \widehat{\underline{\mathbf{n}}}_2 \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{c}}_1\widehat{V} & \underline{\mathbf{c}}\widehat{V} \end{bmatrix}$$

Finding the left null-space of $\underline{\underline{\Gamma}}_{\Sigma}\underline{\underline{\mathbf{S}}}_{eq}$ *results in:*

$$\underline{\underline{\Omega}} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{6.8}$$

The index reduced model of the system can be written as follows:

$$\begin{array}{rcc} \underline{\underline{\Omega}}\,\underline{\dot{\mathbf{n}}} & = & \underline{\underline{\Omega}}\underline{\underline{\Gamma}}_{\Sigma}\underline{\underline{\mathbf{A}}}_{k}\underline{\underline{\Gamma}}_{m}^{T}\underline{\hat{\mathbf{n}}} + \underline{\underline{\Omega}}\underline{\underline{\Gamma}}_{\Sigma}\underline{\underline{\mathbf{S}}}_{r}^{T}\widetilde{\underline{\boldsymbol{\xi}}}_{r} \\ K_1 & = & \frac{c_B c_D}{c_A} \\ K_3 & = & c_D c_E \\ \underline{\mathbf{c}} & = & \frac{\underline{\mathbf{n}}}{V}; \end{array} \tag{6.9}$$

This model can be easily solved with a DAE solver when the vector $\underline{\hat{\mathbf{n}}}$ *and the constants* K_1, k_{2a}, k_{2b}, K_3 *and* V *are appropriately defined and appropriate initial conditions are given (see section 10.3 for more details on solving of a simulation problem).*

□

6.3.2 Full Index Reduction Algorithm

Unfortunately, it is not always possible to use the elegant Simple Index Reduction Method whenever an unmodelled flow or reaction occurs. This seems to be especially the case for "partially unmodelled flows". In this case the primary flow variable is defined, but a secondary flow variable is introduced, which remains unmodelled. For example, the molar mass flow $\underline{\widehat{\mathbf{n}}}$ through a connection can be defined as the concentration $\underline{\mathbf{c}}$ times the volumetric flow rate $\widehat{V}$:

$$\underline{\widehat{\mathbf{n}}} = \underline{\mathbf{c}}\widehat{V} \tag{6.10}$$

Notice that the molar mass flow $\underline{\widehat{\mathbf{n}}}$ is a vector, while the (unmodelled) volumetric flow rate $\widehat{V}$ is a scalar. Since a scalar is unmodelled, only one algebraic constraint may be given. In many cases this will be something like constant volume or constant pressure.

It is not possible to remove the unmodelled variables from the differential equations by algebraic manipulations, so a different, more general index reduction algorithm shall have to be applied in such cases.

The Full Index Reduction algorithm (Moe 1995) removes differential equations from the model definition and replaces them with new algebraic equations until the number of differential equations and dynamic degrees of freedom are the same. The algorithm does not remove variables nor does it introduce new variables.

The constraint equations always contain variables that are directly or indirectly linked to the primary state variables. All equations (i.e. the definitions $\underline{\mathbf{y}} = \underline{\mathbf{f}}(\underline{\mathbf{x}})$) that link these variables to the primary state variables $\underline{\mathbf{x}}$ are collected. Next, these equations, which usually are a small subset of all system equation, are rewritten as $\underline{\mathbf{0}} = \underline{\mathbf{f}}^*(\underline{\mathbf{y}}^*)$ and differentiated with respect to time:

$$\frac{\partial \underline{\mathbf{f}}^*(\underline{\mathbf{x}}^*, \underline{\mathbf{y}}^*)}{\partial t} = \frac{\partial \underline{\mathbf{f}}^*(\underline{\mathbf{y}}^*)}{\partial \underline{\mathbf{y}}^*}\frac{\partial \underline{\mathbf{y}}^*}{\partial t} + \frac{\partial \underline{\mathbf{f}}^*(\underline{\mathbf{x}}^*, \underline{\mathbf{y}}^*)}{\partial \underline{\mathbf{x}}^*}\frac{\partial \underline{\mathbf{x}}^*}{\partial t} = \underline{\underline{\mathbf{F}}}_y \dot{\underline{\mathbf{y}}}^* + \underline{\underline{\mathbf{F}}}_x \dot{\underline{\mathbf{x}}}^* = \underline{\mathbf{0}} \tag{6.11}$$

in which the matrices $\underline{\underline{\mathbf{F}}}_x$ and $\underline{\underline{\mathbf{F}}}_y$ are Jacobians that can be a function of primary and/or secondary state variables and $\underline{\mathbf{x}}^*$ and $\underline{\mathbf{y}}^*$ are subsets of all primary and all secondary state variables respectively. The left null-space $\underline{\underline{\Omega}}_y$ of $\underline{\underline{\mathbf{F}}}_y$ is calculated and multiplied with 6.11:

$$\underline{\underline{\Omega}}_y \underline{\underline{\mathbf{F}}}_y \dot{\underline{\mathbf{y}}}^* + \underline{\underline{\Omega}}_y \underline{\underline{\mathbf{F}}}_x \dot{\underline{\mathbf{x}}}^* = \underline{\underline{\Omega}}_y \underline{\underline{\mathbf{F}}}_x \dot{\underline{\mathbf{x}}}^* = \underline{\mathbf{0}} \tag{6.12}$$

(In the case $\underline{\mathbf{y}}^* = 0 : \underline{\underline{\Omega}}_y = \underline{\underline{\mathbf{I}}}$).

Since $\underline{\mathbf{x}}^*$ is a subset of $\underline{\mathbf{x}}$ we can write:

$$\underline{\mathbf{x}}^* = \underline{\underline{\mathbf{P}}}\,\underline{\mathbf{x}} \tag{6.13}$$

in which $\underline{\underline{\mathbf{P}}}$ is a selection matrix, selecting the elements of $\underline{\mathbf{x}}$ which are present in the vector $\underline{\mathbf{x}}^*$.

The new algebraic equations are obtained by substituting the original balance equations in 6.12 using 6.13:

$$\underline{\underline{\Omega}}_y \underline{\underline{\mathbf{F}}}_x \underline{\underline{\mathbf{P}}} \left(\underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}} \right) = \underline{\mathbf{0}} \tag{6.14}$$

The number of rows of $\underline{\underline{\Omega}}_y$ indicate how many constraints were defined and how many differential equations have to be removed from the original balance equations. The differential equations cannot be removed at random, but must be removed without loss of information and without introducing additional degrees of freedom to the model. This is done by satisfying the following conditions:

- For each row in the matrix $\underline{\underline{\Omega}}_y \underline{\underline{\mathbf{F}}}_x$ (which corresponds to a single constraint) one primary state has to be removed by removing the differential equation it appears in. This can be done by multiplying the original balance equations with a selection matrix.
- For each row, only a state variable that is multiplied by a non-zero element (in the multiplication $\underline{\underline{\Omega}}_y \underline{\underline{\mathbf{F}}}_x \dot{\underline{\mathbf{x}}}^*$ for the specific row) may be removed.
- No state variable that already has been removed can be removed again.

As a result, the following 'new' differential and new algebraic equations are obtained:

$$\underline{\underline{\mathbf{P}}}_x \dot{\underline{\mathbf{x}}} = \underline{\underline{\mathbf{P}}}_x \left(\underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}} \right) \tag{6.15}$$

$$\underline{\mathbf{0}} = \underline{\underline{\Omega}}_y \underline{\underline{\mathbf{F}}}_x \underline{\underline{\mathbf{P}}} \left(\underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}} \right) \tag{6.16}$$

in which $\underline{\underline{\mathbf{P}}}_x$ is a selection matrix which defines which state variables are removed from the original balance equations. These 'new' equations togheter with the original algebraic equations and constraints form an index-1 DAE. The equations 6.16 are used to compute the unmodelled quantities indirectly. If you want the unmodelled quantities to be solved directly (i.e. explicitly), some substitutions and transformations have to be performed.

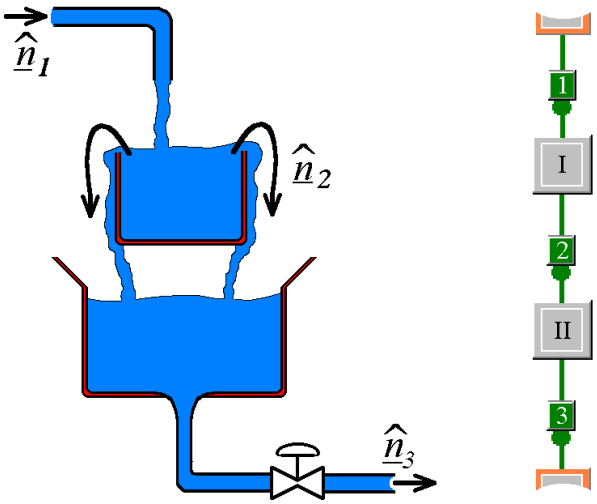

Figure 6.1: *Overflowing tank and its physical topology.*

Example 6.3: ***Overflowing Tank***

A process consists a mixing tank that flows over into a second tank (see figure 6.1). The liquid that is fed to the tank system contains two components (A and B). Suppose that it is difficult to find a reliable relation that describes the flow $\underline{\hat{n}}_2$ out of the overflow tank. Therefore the assumption is made that the liquid level in the tank is constant (this means that the liquid volume dynamics are assumed to be very rapid compared to the composition dynamics). The assumption of constant volume is, of course, physically incorrect (Moe 1995). A physically correct description of the overflow would have included a relation between the liquid level and the amount of liquid which is flowing out of the tank. Though practically, the variation in the volume is hardly ever relevant in these circumstances and the resulting low index model will be stiff.

For simplicity the density is considered to be constant and in order to make things visible, sometimes the matrix equations are written out as scalar equations (the conversion to the standard matrix notation is, of course, easily made).

Balance equations:

$$\begin{aligned}
\dot{n}_{A,I} &= \hat{n}_{A,1} - \hat{n}_{A,2} && (6.17)\\
\dot{n}_{B,I} &= \hat{n}_{B,1} - \hat{n}_{B,2} && (6.18)\\
\dot{n}_{A,II} &= \hat{n}_{A,2} - \hat{n}_{A,3} && (6.19)\\
\dot{n}_{B,II} &= \hat{n}_{B,2} - \hat{n}_{B,3} && (6.20)
\end{aligned}$$

System equations (same for both systems):

$$\begin{aligned}
\underline{\mathbf{c}} &= \frac{\underline{\mathbf{n}}}{V} && (6.21)\\
m &= \underline{\mathbf{e}}^T \underline{\underline{\mathbf{M}}}\, \underline{\mathbf{n}} = M_A n_A + M_B n_B && (6.22)\\
V &= \frac{m}{\rho} && (6.23)
\end{aligned}$$

Connection equations:

$$\begin{aligned}
\hat{n}_{A,1} &= c_{A0}\hat{V}_1 \quad \hat{n}_{B,1} = c_{B0}\hat{V}_1 && (6.24)\\
\hat{n}_{A,2} &= c_{A,I}\hat{V}_2 \quad \hat{n}_{B,2} = c_{B,I}\hat{V}_2 \quad \hat{V}_2 =? \Rightarrow V_I = V_{I,\max} && (6.25)\\
\hat{n}_{A,3} &= c_{A,II}\hat{V}_3 \quad \hat{n}_{B,3} = c_{B,II}\hat{V}_3 \quad \hat{V}_3 = f(n_{A,II}, n_{B,II}) && (6.26)
\end{aligned}$$

Where

$n_{i,j}$	::	*molar mass of component i in system j*
$\hat{n}_{i,k}$	::	*molar mass flow of component i through connection k*
m_i	::	*total mass of system i*
M_i	::	*molar mass of component i*
V_i	::	*Volume of system i*
$c_{i,j}$	::	*molar concentration of component i in system j*
$\hat{V}_k$	::	*Volumetric flow rate of connection k*
ρ	::	*density*

The constraint equation ($V_I = V_{I,\max}$) constrains the volume and makes the volume V_I of system I no longer state (nor time) dependent. The volume is related to the mass of the system (equation 6.23) and the mass is related to the component masses (equation 6.22). Differentiating the relevant equations results in:

$$\begin{aligned}
\underline{\underline{\mathbf{F}}}_y \dot{\underline{\mathbf{y}}}^* + \underline{\underline{\mathbf{F}}}_x \dot{\underline{\mathbf{x}}}^* &= \underline{\mathbf{0}}\\
\begin{pmatrix} 1 \\ -1 \end{pmatrix} (m_I) + \begin{pmatrix} 0 & 0 \\ M_A & M_B \end{pmatrix} \begin{pmatrix} n_{A,I} \\ n_{B,I} \end{pmatrix} &= \underline{\mathbf{0}} \qquad (6.27)
\end{aligned}$$

$$\Rightarrow \underline{\underline{\Omega}}_y = \begin{pmatrix} 1 & 1 \end{pmatrix} \Rightarrow \underline{\underline{\Omega}}_y \underline{\underline{\mathbf{F}}}_x = \begin{pmatrix} M_A & M_B \end{pmatrix} \tag{6.28}$$

which results in the following additional algebraic equation:

$$\begin{aligned} 0 &= \begin{pmatrix} M_A & M_B \end{pmatrix} \begin{pmatrix} \dot{n}_{A,I} \\ \dot{n}_{B,I} \end{pmatrix} && (6.29) \\ &= M_A \hat{n}_{A,1} + M_B \hat{n}_{B,1} - M_A \hat{n}_{A,2} - M_B \hat{n}_{B,2} && (6.30) \end{aligned}$$

Removing one of the states of the original equations (either $n_{A,I}$ or $n_{B,I}$) and replacing its differential equation with the new algebraic relation results in an index-1 DAE, which can be easily solved by a DAE solver.

Quite obviously, when equation 6.30 is worked out, it will follow that, in this simple case (constant density), the volumetric outflow $\hat{V}_2$ will be the same as the volumetric inflow $\hat{V}_1$:

$$\begin{aligned} 0 &= (M_A c_{A0} + M_B c_{B0})\,\hat{V}_1 - (M_A c_{A,I} - M_B c_{B,I})\,\hat{V}_2 \\ &= (M_A n_{A0} + M_B n_{B0})\,\frac{\hat{V}_1}{V_0} - (M_A n_{A,I} - M_B n_{B,I})\,\frac{\hat{V}_2}{V_I} \\ &= \frac{m_0}{V_0}\hat{V}_1 - \frac{m_I}{V_I}\hat{V}_2 = \rho\hat{V}_1 - \rho\hat{V}_2 \\ \Leftrightarrow\ & \hat{V}_2 = \hat{V}_1 \end{aligned}$$

Note: *Very often, similar examples are used in textbooks without anybody noticing that there is (or was) an index problem related to the example. This has the following reasons: The total mass balance and all but one component mass balances (mostly the solvent is left out) are written down and, without knowing it, no link is being made between the component masses and the total mass:*

$$\begin{aligned} \dot{m}_I &= \rho\hat{V}_1 - \rho\hat{V}_2 \\ \dot{n}_{A,I} &= \hat{n}_{A,1} - \hat{n}_{A,2} \\ V &= \frac{m}{\rho} \\ \underline{\mathbf{c}} &= \frac{\mathbf{n}}{V} \end{aligned}$$

Making the assumption of constant density and constant volume results in constant total mass, which easily gives:

$$0 = \rho\hat{V}_1 - \rho\hat{V}_2 \Rightarrow \hat{V}_2 = \hat{V}_1$$

When a modeller is not aware of the fact that the component masses were not related to the total mass in this case, he could get into trouble when things get a bit more complicated (for example when the density is a function of the mole fractions).

□

6.4 Steady-State Assumptions

The need for (pseudo) steady-state assumptions may arise for different reasons. Often the purpose is to either simplify the dynamic behaviour of the system by idealising components or it is to compensate for information that is not readily available or difficult to model for other reasons. The idealisation of components is context-wise related to the subject of singular perturbation. Since the model is build for a specific purpose, one may choose to ignore the dynamic effects of certain parts of the system and assume that they are either extremely slow thus can be approximated as reservoir providing an infinite-size source of matter, energy, etc., or one may make assumptions of instantaneous dynamics for small-capacity parts.

If one critically analyses models, as they are published in the literature, one finds that such assumptions are omnipresent. In fact they are intrinsically in the basics as one mostly builds models on the assumption of reversibility, which does imply that one deals with small capacity elements that move along trajectories on the equilibrium surface.

One often finds problems where two systems of largely different nature are coupled. For example a "large" system that is connected to a "small" system. Often the dynamic effects of the small system may be ignored, but very often too, the small system makes all the difference. Models in which a phase transition takes place are of this nature in that the phase boundary separating the two phases is very important when describing the flow of mass from one phase to another. This concept may also be applied to time scales, such as fast and slow systems. For these systems the singular perturbation theory is applicable. For fast systems a pseudo steady-state assumption is made:

$$\varepsilon\underline{\dot{\mathbf{x}}} = \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}} \tag{6.31}$$

which yields

$$\underline{\mathbf{0}} = \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}} \tag{6.32}$$

if $\varepsilon := 0$. The capacity effects of the fast systems are omitted and this reduces the state of the process because the dynamic equations of this parts are set

to zero. The state of a fast capacity becomes a function of the state of its environment, which mathematically reflects into the elimination of the state. So, for these fast systems the accumulation terms vanish and, consequently, the primary state variables are "undefined" for these systems. The balance equations of the fast systems are still valid and are used to calculate "undefined" variables either directly or indirectly. Quite obviously, an unmodelled flow can be calculated directly via a steady-state assumption.

In general, a number of steady-state assumptions that remove n states result in the following set of algebraic equations (only for the fast capacities):

$$\underline{\mathbf{0}} = \underline{\underline{\mathbf{A}}}\,\underline{\mathbf{z}} + \underline{\underline{\mathbf{B}}}\,\underline{\mathbf{r}} \tag{6.33}$$

$$\underline{\mathbf{y}} = \underline{\mathbf{f}}(\underline{\mathbf{x}}) \tag{6.34}$$

$$\underline{\mathbf{z}} = \underline{\mathbf{g}}(\underline{\mathbf{y}}, \underline{\mathbf{y}}_e) \tag{6.35}$$

$$\underline{\mathbf{r}} = \underline{\mathbf{h}}(\underline{\mathbf{y}}) \tag{6.36}$$

from which n variables (either flow rates ($\underline{\mathbf{z}}$), reaction rates ($\underline{\mathbf{r}}$) and/or states ($\underline{\mathbf{x}}$ or $\underline{\mathbf{y}}$)) can be calculated (less variables can be calculated when index reduction algortihms affect the steady-state balance equations (see example 6.6)). In this set the $\underline{\mathbf{y}}_e$ represent the (secondary) states of the environment systems (i.e. systems that are connected to the steady-state systems) and are the "driving forces" of the algebraic equations. A bipartite graph analysis (Duff and Reid 1986) could provide the means for finding the different sets of variables that can be computed from this set of algebraic equations. Normally, though, each steady-state assumption can be handled locally and it is clear for which variables the equations are solved.

Example 6.4: ***Calculation of Unmodelled Flow***

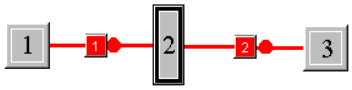

Figure 6.2: *Physical topology of simple process*

Three systems are interconnected via heat connections. The enthalpy balance of the middle system (2) of figure 6.2 is:

$$\dot{H} = \hat{q}_1 - \hat{q}_2 \tag{6.37}$$

If the second flow ($\hat{q}_2$) remains unmodelled, then a steady-state assumption (on system 2: $\dot{H} := 0$) can be made to calculate this

flow via the balance equation:

$$\hat{q}_2 = \hat{q}_1 \tag{6.38}$$

If there would have been no inlet flow to system 2 and a steady-state assumption is made, this would, of course, result in no flow through connection 2 (i.e. $\hat{q}_2 = 0$).

□

As mentioned, the steady-state assumption essentially makes the state of a system a function of the state of its environment. The state of the environment can thus be used to "redefine" the (secondary) state (i.e. the intensive quantities) of the system.

Example 6.5: ***Calculation of State of a Steady-State System***

Consider figure 6.2 once again. The heat flows through the connections 1 and 2 are now defined as functions of the temperatures of the interconnected systems:

$$\hat{q}_1 = U_1 A_1 (T_2 - T_1) \tag{6.39}$$
$$\hat{q}_2 = U_2 A_2 (T_3 - T_2) \tag{6.40}$$

If a steady-state assumption is made for system 2, the primary state of the system is eliminated and, consequently, also the secondary state, because this results from a mapping from the primary state. The temperature T_2 of system 2 is now defined as a function of the states of the environment (T_1 and T_3) and can be calculated from the balance equation of this system:

$$0 = \hat{q}_1 - \hat{q}_2 \tag{6.41}$$
$$\Leftrightarrow T_2 = \frac{U_1 A_1 T_1 + U_2 A_2 T_3}{U_1 A_1 + U_2 A_2} \tag{6.42}$$

***Note**: When one or both flows remain unmodelled, the balance equation of system 2 is effectively removed from the model by applying the Simple Index Reduction Method. The introduced constraint (e.g.: $T_2 = T_1$ and/or $T_2 = T_3$) equations are then used to calculate T_2.*

□

The two examples showed that a steady-state assumption can be used to either calculate an unmodelled flow (or reaction) directly or calculate the (secondary) state of the system, that is redefined by its environment. Both possibilities do not lead to index problems, since the states are removed directly form the model description. These two possibilities can, however, not be mixed, because one would run into mathematical problems. If, for example, the inlet flow ($\hat{q}_1$) is defined as in equation 6.39 and we wish to calculate $\hat{q}_2$ via a steady state assumption, there would be a problem because there is no information on the temperature T_2. When the steady-state assumption calculates the state of a system, then at least one of the connections must, of course, reference this state. Otherwise the resulting set of equations will be inconsistent and it will not be possible to calculate the state.

Conceptional difficulties arise when a steady-state assumption reduces a system that is connected with more than two connections that remain unmodelled (and for which no constraint equations are given). In these cases, one of the flows is calculated via the steady-state assumption and the other flows are defined as a function of the calculated flow. So, in these cases, some flows can be a function of another flow and not, which is the general case, of the state variables of the connected systems.

Mass may be exchanged between two phases, each of which is in a different state of aggregation. Mass crossing the boundary thus changes state of aggregation. Defining the phase boundary as a thermodynamic wall localises the mechanics associated with the phase change into the boundary. The boundary is assumed to be a surface and has thus no volume. Consequently, no fundamental extensive quantity is accumulated in the boundary itself: the flow of extensive quantity entering the surface from one side must balance the flow leaving the surface on the other side. Clearly, the two systems cannot be simply connected with a connection, since connections are idealised transport systems in which no reactions occur and a phase change *is* a reaction. Therefore, phase boundaries are approximated with steady-state systems, in which the two phases coexist and where the phase change takes place.

This phase boundary concept can, of course, be further expanded to phase boundaries where no state of aggregation change or reaction takes place. In these cases, the transfer of a component from one phase to another can be viewed as a "pseudo reaction". At the phase boundary, the transferred component acts as two "different" components - one at each phase - that interact with each other.

Example 6.6: ***Phase Equilibrium***

Consider a process that consists of two phases (phase I and Phase II). In each phase component A is dissolved and the transfer from

the bulk phase to the phase boundary is fast. Also, the transfer from one phase to another is considered to be fast. When the phase boundary is represented by a steady-state system a physical topology similar to the one shown in figure 6.2 will result. The component mass balances for component A can be easily derived from this topology (keeping in mind that, at the phase boundary, this component will act as two "different" components):

$$\dot{n}_{A(I),1} = -\hat{n}_{A,1} \tag{6.43}$$
$$\dot{n}_{A(I),2} = \hat{n}_{A,1} - \xi \tag{6.44}$$
$$\dot{n}_{A(II),2} = -\hat{n}_{A,2} + \xi \tag{6.45}$$
$$\dot{n}_{A(II),3} = \hat{n}_{A,2} \tag{6.46}$$

where,

$n_{A(i),j}$	::	*Molar mass of component A in phase i at system j*
$\hat{n}_{A,k}$	::	*Molar mass flow of A through connection k*
ξ	::	*Conversion rate of A from phase I to phase II*

The following assumptions are made:

- *flows are fast. Constraint equations:*

$$c_{A(I),1} = c_{A(I),2} \tag{6.47}$$
$$c_{A(II),3} = c_{A(II),2} \tag{6.48}$$

  with

  $c_{A(i),j}$ :: *Molar concentration of component A in phase i at system j*

- *"reaction" is fast:*

$$c_{A(I),2} = k c_{A(II),2} \tag{6.49}$$

- *steady-state assumption for system 2:*

$$\dot{n}_{A(I),2} = 0 \tag{6.50}$$
$$\dot{n}_{A(II),2} = 0 \tag{6.51}$$

Applying the Simple Index Reduction Mechanism results in:

$$\dot{n}_{A(I),1} + \dot{n}_{A(II),3} = 0 \tag{6.52}$$
$$c_{A(I),1} = c_{A(I),2} \tag{6.53}$$
$$c_{A(II),3} = c_{A(II),2} \tag{6.54}$$
$$c_{A(I),2} = k c_{A(II),2} \tag{6.55}$$

Note 1: *Since the steady-state balance equations are effectively removed by the application of the Simple Index Reduction Method, these equations are no longer required to compute unknown quantities. So, when k unmodelled quantities, that appear (once or twice) in l steady-state balance equations, are removed by applying an index reduction method, then l-k (or zero if $k > l$) "unknown" quantities can be computed with the remaining equations. The unknown quantities of the steady-state system are now effectively calculated via the imposed constraints.*

Note 2: *In this case, the same results would have been obtained when the systems 1 and 3 would have been connected via a single connection (with system 2 not being present). If a fast flow assumption would be made (e.g. $c_{A,1} = kc_{A,3}$), mathematically the same model would result. The former case is, however, more general and can be easily extended to multiple connections.*

□

6.5 Events and Discontinuities

In this work, the dynamic behaviour of a process is abstracted to the point at which it can be represented by the smooth, continuous change of a series of state variables. This allows such a process to be represented mathematically as a set of DAEs. The majority of processes, however, cannot be considered entirely continuous, but also experience significant *discrete changes* or *discontinuities* superimposed on their predominantly continuous behaviour (Barton 1992). Such discrete changes, often referred to as *events*, can either be planned changes, called *time events* (for example, planned operational changes, such as start-up and shut-down, feed stock and/or product changes, process maintenance, switching a valve) or non-planned changes, called *state events* (overflow of a tank, (dis)appearance of a phase, inversion of a flow).

The exact time of occurrence of time events is either known a priori or it can be computed from the occurrence of a previous event. The time of occurrence of state events is not known in advance because it depends on the system fulfilling certain *state conditions*.

In addition to the two types of events we can classify events in an other way, namely in events that do not change the (dynamic) model structure and events that change the (dynamic) model structure. With events that do not change the model structure, the balance equations are not altered, but some algebraic equations are changed. This usually results in a non-smooth continuation of the primary state variables.

Example 6.7: ***Tank with Fast and Equilibrium Reaction***

Consider a tank with an inlet (consisting of two components A and B) and an outlet. The contents of the tank consist of four components (A, B, C and D) among which two fast reactions occur, namely:

$$A + B \rightarrow C \tag{6.56}$$

$$A \rightleftarrows D \tag{6.57}$$

The first reaction (6.56) is a very fast, one way reaction. This means that there is no measurable inverse reaction and one of the two components A or B will always be completely consumed (i.e.: $n_A = 0$ or $n_B = 0$). The other reaction (6.57) is an equilibrium reaction, which means that there will always be a certain ratio between components A and D ($c_A = k * c_D$). Because both reactions are fast in the context of the viewed time scale, the reaction terms that appear in the balance equations can be removed be using the Simple Index Reduction Algorithm.

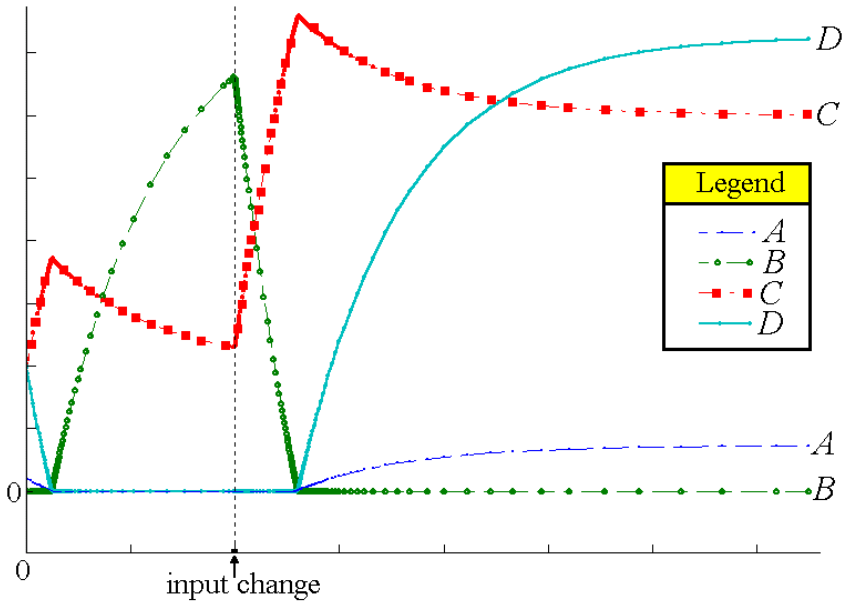

Figure 6.3: *Simulation of a tank in which a fast and an equilibrium reaction take place.*

Figure 6.3 shows the result of a simulation of the tank. In this figure we can see three events taking place. At the start of the

simulation there is an excess of A in the tank, which means that there will be no B, since all B that is available directly reacts with A to form C. The inlet flow contains more B than A and this results in diminishing quantities of A (and, consequently, D) in the tank until the amount of A (and D) in the tank reaches zero. At this point we encounter the first event, for now there will be an excess of B available in the tank and A (and D) will be non-existent.

At a certain point in time the input is changed such that the inlet flow will contain more A than B. A second event is the result. The amount of B will decrease until is reaches zero. This will initiate the third event, such that A (and D) will be present again.

Obviously, the first and last event were state events, while the second event was a time event.

□

With events that change the (dynamic) structure of the model, not only algebraic equations are altered, but also (parts of) the dynamic balance equations, because the events trigger certain index problems. This means that the number of differential variables will increase or decrease after the event has occurred. Examples of these events are a tank that suddenly overflows (when the overflow is modelled with a constraint (constant volume of the contents), the number of differential variables will have to decrease, due to the introduced index problem), a suddenly starting (or stopping) equilibrium reaction or a system that runs empty. This type of events can cause jumps in the primary state variables, although it usually does not.

Example 6.8: ***Tank of Small Diameter***

This example shows what kind of problems may arise when a system is discretised into a set of interconnected subsystems.The physical topology of a tank with a small diameter is considered to consist of four interconnected systems, each with a maximum volume of $V_{\max}$*. The volumetric outlet flow (*$\hat{V}_5$*) is a function of the height (c.q. the pressure) of the liquid in the tank. The height is a "composite" quantity because it is derived from the sum of volumes of the four systems.*

If the tank is almost full (i.e. $V_{II} = V_{III} = V_{IV} = V_{\max}$*), the flow rate through connection 2 can be calculated by making the "assumption":* $V_{II} = V_{\max}$*, and subsequently applying the Full Index Reduction Method. Similarly the flows of connections 3 and 4 can be calculated. (The result will, of course, be that all these flows are equal to the outlet flow if constant density is assumed).*

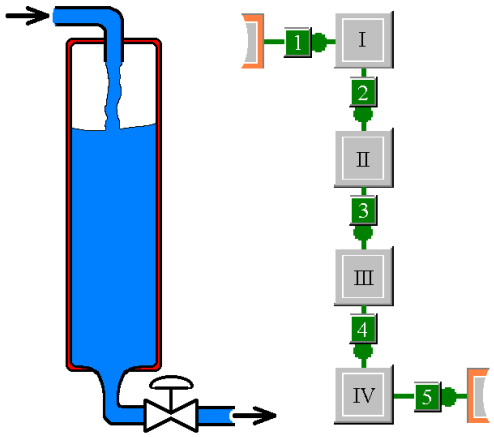

Figure 6.4: *A tank with small diameter and a possible physical topology.*

A problem arises when system I becomes completely empty. The assumption that $V_{II} = V_{\max}$ *is no longer valid, because* $V_{II} < V_{\max}$ *and* $V_I = 0$. *The liquid no longer accumulates in system I and the states of system I become obsolete. The flow through connection 2 will now be calculated by making a steady-state assumption over system I. (The result will be that the flow through connection 2 equals the flow through connection 1). This event will remove the states of system one and also the Full Index Reduction on system II will have to be "undone", because connection 2 no longer constrains this system. The mathematical description for connection 2 will be as follows:*

$$\underline{\hat{\mathbf{n}}}_2 : \begin{cases} V_I > 0 : & V_{II} = V_{\max} \\ & \textit{(calcutate } \underline{\hat{\mathbf{n}}}_2 \textit{ via Full Index Reduction)} \\ V_I = 0 : & \underline{\dot{\mathbf{n}}}_I = \underline{\mathbf{0}} \\ & \textit{(calculate } \underline{\hat{\mathbf{n}}}_2 \textit{ via Steady-State Assumption)} \end{cases}$$

Clearly, the dynamic structure of the model changes when the event occurs. The handling of events that change the dynamic structure of the model by (numeric) solvers is far from trivial and goes beyond the scope of this thesis.

□

Whatever type of event, there will always be some kind of discontinuity or "non-smoothness" in the DAE. To solve DAEs with discontinuities successfully, the time integration has to be stopped at the discontinuity and restarted

with new consistent initial conditions (Majer and Gilles 1995). These consistent initial conditions can usually be quite easily found from the conditions of the process just before the event occurred. In many cases (especially when the dynamic model structure is not altered) the "new" initial conditions are actually the same as the conditions just before the event and it will often not be necessary to stop and restart the time integration.

6.6 Variables and Algebraic Equations for Composite Systems

Sometimes it is convienient, or even necessary, to introduce variables and algebraic equations for composite systems. In the last example of the previous section a "composite" quantity was, for example, introduced to describe the height of the liquid in the tank. In this case this height was used to calculate the magnitude of the outlet flow.

Composite variables are defined via algebraic relations as a function of the variables of the subsystems. In the case of extensive quantities, the composite variable will, obviously, be the sum of these quantities of the subsystems, since the physical topology is defined as a strictly hierarchical multiway tree.

When the composite variables are just defined and used, no mathematical problems will arise, since the new variables are defined using already defined variables. When, however, a composite variable is constraint (for example: the assumption that the volume of a composite system is constant), mathematical problems could arise. Two cases can be distinguished:

- *All used variables of the subsystems are directly or indirectly related to their primary state variables and the composite variable holds a constraint.* In this case an index problem arises, which will have to be dealt with using either the Simple or the Full Index Reduction Method. As with other high-index cases, there has to be an unmodelled flow (or reaction) which makes the application of the index reduction method possible.

- *The composite variable is used to calculate a variable of one of the subsystems.* When the composite variable is constraint and all but one of the used variables is properly defined in the subsystems, then the composite equation can be used to calculate the remaining variable. Consider, for example, a tank in which a liquid and a gas phase are present and which has an inlet and an outlet flow. The volume V of the tank is fixed. The volume of the liquid V_L can be calculated from the liquid mass. The remainder part of the tank that is not occupied with liquid is the volume that is occupied by the gas phase ($V_G = V - V_L$). In these special

cases, a variable of a subsystem is calculated using state information of other subsystems. This solution will not result in an index problem, but the computational causality (this subject is discussed in more detail in section 10.3) of the involved variables and equations should be carefully considered in these cases.

Note: Strictly speaking we are not talking about a simulation problem in these cases, since the calculated variable (V_G in our example) of the subsystem is not directly related to the primary state variables (the mass of the gas phase). The definition $m_G = \rho_G V_G$ could, in this case be used to "identify" the density ρ_G (parameter identification).

6.7 Summary and Conclusions

This chapter showed that assumptions that "simplify" the process description very often lead to computational difficulties. Luckily, since the assumptions are usually introduced locally, the imposed problems stay local and can also be solved locally. The following list summarises the assumptions a model designer can make during the development of a process model:

- **Unmodelled flow** ⇒ High index process model.

  Solutions:

  - Introduce a constraint that directly or indirectly influences primary state variables and apply the Simple Index Reduction Method.

    Result: The unmodelled flow is removed from the model description by making linear combinations of the balance equations. The index and dimension of the model will decrease.

  - Introduce a constraint equation and apply the Full Index Reduction Method.

    Result: The index and dimension of the model have decreased because some states have been removed and the flow is calculated from "new" equations, which are generated from combining some balance equations with differentiated algebraic equations.

  - Apply a steady-state assumption to one of the connected systems.

    Result: The dimension of the model is reduced because the states of the steady state system have been removed and the flow is calculated via the balance equations of the steady state system.

- **Unmodelled reaction** ⇒ High index process model.

  Solutions:

  – Introduce a constraint that directly or indirectly influences primary state variables and apply the Simple Index Reduction Method.

  Result: The unmodelled reaction is removed from the model description by making linear combinations of the balance equations. The index and dimension of the model will decrease.

  – Introduce a constraint equation and apply the Full Index Reduction Method.

  Result: The index and dimension of the model have decreased because some states have been removed and the reaction is calculated from "new" equations, which are generated from combining some balance equations with differentiated algebraic equations.

  (-) Apply a steady-state assumption to the system in which the reaction is taking place.

  Result: The dimension of the model is reduced because the states of the steady state system have been removed and the reaction is calculated via the balance equations of the steady state system.

  Note: This is in general not possible, because usually the number of removed states is larger than the number of unmodelled reactions that have to be calculated. The assumption must therefore be partially used to calculate some other unmodelled quantities and this is most of the time not trivial.

- **Steady-state assumption** ⇒ Reduces the dimension of the model.

  A number of unmodelled quantities (either flows, reactions or secondary state variables) equal to the number of removed states have to be calculated from the steady-state balance equations.

A major advantage of this classification of assumptions is that it is very easy to do the book keeping. An automatic documentation of all assumptions that are made during the model development is the result. All assumptions become very easily retractable and changeable.

In all cases, the introduction of a "simplification" leads to a reduction of the dimension of the original model and therefore to a model reduction. Many simplifications are solved by an index reduction which leads to a model reduction. Index reduction and model reduction are, however, not the same thing. The purpose of an index reduction is to transform the model such that a model results that can be solved by problem solving software. The goal of

model reduction is to simplify the model description such that a (much) faster and simpler code is formed that still represents the actual process behaviour with acceptable accuracy. So, although index reduction is a model reduction method, model reduction is certainly more than just index reduction. This work was not concerned with model reduction, but with the construction of consistent dynamic process models.

Part II

Implementation

Chapter 7

Introduction to Implementation

Solving process engineering without the help of computer-based tools is for almost any problem an unthinkable proposition. Process simulation, process design, controller design, controller testing, data acquisition and model identification, parameter fitting, valve and pump selection, column sizing are just a few examples taken from a very large catalogue of chemical plant related operations that are almost exclusively done with computer-based tools. Considering the fact that multiple solutions to individual problems are available from different sources such as software houses or supplier companies, the catalogue of tools is rather large. A process engineer, who is more and more involved in the integrated design of processing plants, is thus faced with a multitude of different software tools, which he uses more or less frequently depending on his allocation. Since each of these packages has been developed in isolation, each has a different interface and applies to different problems though the tools may be used to solve problems associated with the same plant.

Often the software is very complex and needs a specialist to define and run problems. Data transfer between tools is not standardised and requires special arrangements or add-on's. All features which do not enhance team work and productivity.

The key to improving efficiency is an increased use of common process information and effictive exchange of data, models and information between the applications.

During my research period I have developed a computer tool, called the Modeller, based on the ideas and concepts which are described in the previous chapters. The Modeller is a computer-aided modelling tool to interactively define and modify process models. It aims to effectively assist in the develop-

ment of these process models and facilitate hierarchical modelling of process plants through a user friendly interface.

The Modeller constructs the process models from primitive building blocks, being simple thermodynamic systems and connections, as discussed in section 3.1. It does not, in distinction to existing flow sheeting packages, build on unit models. The Modeller generates symbolic models in the form of differential-algebraic equations consisting of component mass and energy balances, augmented with transfer laws, physical and geometrical property relations and kinetic laws. In this part the implementation of the modelling method in the computer-aided modelling tool the Modeller is discussed.

This part starts with describing the construction and manipulation of physical topologies (chapter 8) for PCB processes with the aid of the Modeller. A special graphical user interface has been developed (and implemented) to handle physical topologies of arbitrary complexity. With the modelling tool, the physical topology of a process can be built using two main operations, namely refining an existing system (the top-down approach) or grouping systems (the bottom-up approach).

The next chapter discusses how one can construct and manipulate the species topology of a physical topology, using the Modeller. The definition of the species topology is initialised by assigning sets of species and reactions to some elementary systems. To aid in this definition, species and reaction databases are used. The user may also specify the permeability (i.e. selective transfer of species) of individual mass transfer connections. The distribution of the species over all systems is automated and uses the facts that assigned species can propagate through permeable mass connections and species may generate "new" species (via reactions), which in turn may propagate and initiate further reactions.

Finally, the implementation and handling of variables and equations, which constitute the generated models, is discussed. With the information on the physical and species topology Modeller can automatically generate balances of fundamental extensive quantities (component mass, energy and momentum) of every elementary system. In order to fully describe the behaviour of the process, the flow rates and production rates involved in the balance equations need to be specified. So, in addition to the balance equations other relationships are needed to express thermodynamic equilibria, reaction rates, transport rates for heat, mass, momentum, and so on. A model designer can select a particular relationship from a set of alternatives and connect the selected relationship to a balance equation or to another defined relationship. In this way, each term in a balance equation can be expanded by defining it as a function of some quantities, which in turn may be expanded again and again. The resulting set of equations (the output of the modelling tool) forms

a mathematical representation for the behaviour of the process in a specific form, which could serve as an input for problem solving tools.

7.1 Implementation Details

This part, Implementation, mainly describes how the implementation works and not how it was implemented. Although the actual implementation required a lot of work and was far from trivial, I do not consider the implementation details to be an important contribution to this thesis. Elaborating on these details would certainly not aid in the overall understanding of the modelling methodology and its implementation for most of the readers. I do wish to mention something on the actual implementation and especially on some of the low-level modules I have written, since these form the basis for the entire program.

The software tool, the Modeller, was implemented using the BlackBox Component Builder 1.4 (http://www.oberon.ch). The BlackBox Component Builder is an integrated development environment optimised for component-based software development. It consists of development tools, a library of reusable components, a framework that simplifies the development of robust custom components and applications, and a run-time environment for components.

In BlackBox, the development of applications and their components is done in Component Pascal. This language is a descendant of Pascal, Modula-2, and Oberon (Pfister and Szyperski 1994). It provides modern features such as objects, full type safety, components (in the form of modules), dynamic linking of components, and garbage collection. The *entire* BlackBox Component Builder is written in Component Pascal: all library components, all development tools including the Component Pascal compiler, and even the low-level run-time system with its garbage collector. In spite of its power, Component Pascal is a small language that is easy to learn and easy to teach.

When building or manipulating process models, the construction and manipulation of an earlier step has great influence on the construction and composition of the later steps. Manipulating later steps will have no influence on the composition of earlier steps and will therefore not alter the entered information of these steps. In implementation terms this means that an earlier step should not know about later steps. So, a tool which constructs only a physical topology should have no knowledge of species, variables or equations. A tool which implements the construction of a species topology must have knowledge about the physical topology but not of variables and equations.

When the different steps are implemented in different components, actions taken on one specific step can be clearly separated from actions taken on another step. A clear separation of the different modelling steps can make operations such as error detection and implementation of new procedures in specific steps a lot less time consuming.

The program, the Modeller, is build from several modules, which all handle certain aspects of the implementation The main modules of the program are very applicably called CamPhysTops, CamSpecTops and CamEqTops. On top of these modules other modules are implemented, which handle the graphical user interface (GUI), graphical representation, man-machine interaction, etc.

To form a firm basis on which the entire program could be build, a few general purpose, low-level modules were written. The two most important of these general purpose modules were named CamTools and CamLists.

The module CamTools forms the foundation for undoable and redoable operations. This module uses the implementation of the original BlackBox undo/redo mechanism in such a way that more than one operation can be undone or redone within one call. Appendix A gives the interface definition of the module CamTools and explains how the undo/redo mechanism works.

The module CamLists defines the data types List and ListReader, which can be used to handle all kinds of operations that involve lists. Any storable object can be added to a list. All manipulations to a list (adding and removing objects) which alter the contents of the list are implemented as operations. This means that all manipulation actions are undoable (and redoable).

Module CamPhysTops defines all the basic tools and operations that are needed to construct physical topologies. The module defines the types System and Connection, which are the two primitive building blocks for the construction of physical topologies. Type PhysTop represents physical topologies, consisting of systems and connections. It provides all the operations and procedures needed to construct physical topologies.

A physical topology can be extended by laying, for example, a species topology or a variable topology on top of this physical topology. Also, different graphical representations of the same physical topology could be implemented. Type Extension implements the procedure HandleModification which handles all the modifications made to the physical topology. An extension of the physical topology may implement this procedure to react to changes made in the physical topology. For example, a graphical representation (or a species topology) has to update its data when a system is added or removed. An extension can associate an attribute with every system and every connection of the physical topology. In this way, information of an extension that belongs to a specific system (or connection) is carried by this system (or connection).

An attribute carries specific information of a specific system or connection. This information should, of course, be related to the extension to which the attribute belongs. A graphical attribute (related to a graphical extension) of a system, for example, could contain information on the location of this system on the screen. Every system and every connection can only contain one attribute of a specific extension. This means that every extension can define two attributes: one for systems and one for connections.

The graphical user-interface of the Modeller only contains two main elements. These are the model windows in which the physical topologies of the different models can be constructed and a properties toolbox, which displays the properties and information of the selected object(s) or of the entire model (when no selection is made). The properties toolbox is fully context dependent, which means that if the selection is changed, the properties toolbox will immediately addapt and show the properties of the new selection. In this way it is always clear on which object(s) certain actions are performed.

Chapter 8

Construction of the Physical Topology

The construction and manipulation of the physical topology in an easy and fast manner, without compromising on the consistency of the process models, is one of the primary objectives of the computer-aided modelling tool Modeller. The tool should therefore allow for any structural change in any order. In this section the basic operations one can perform on a physical topology are discussed.

8.1 Handling Complexity

Elementary systems and connections are the primitive building blocks for constructing a mathematical model for a process. A (correctly defined) connection is always defined between two elementary systems. Thus, a network consisting of elementary systems and connections has a flat structure from a topological point of view. To aid in the handling of large and complex processes, the physical topology is organised in a strictly hierarchical multi-way tree. This means that the systems can be hierarchically grouped in composite systems, such that groups of components can be addressed. So, an additional tree structure is introduced, which is laid over the "flat" physical topology.

The visualisation of larger processes can still get rather complicated. The overall model of a large process could easily outgrow the screen and one could easily lose grasp of the whole process, especially if one would only show the flat topology. This implies that we have to find a representation of the tree that only shows one specific part of the tree with detail. The rest of the model should be shown with very little detail. We should keep in mind, though,

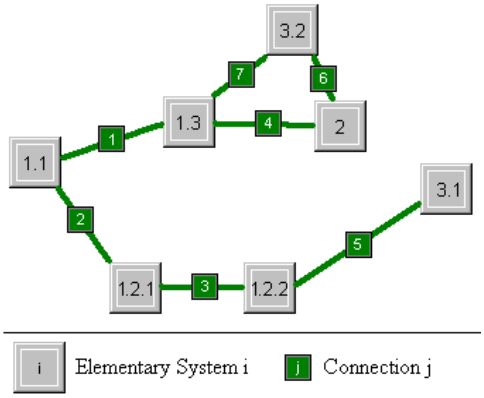

Figure 8.1: *Example of a 'flat' physical topology.*

that we always want to show the complete thermodynamic universe, i.e. the complete process.

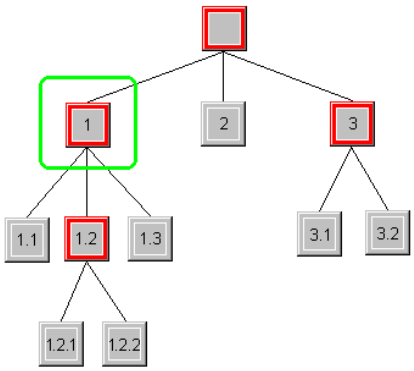

Figure 8.2: *Tree structure representation of a process, without showing the connections*

A generic approach to this problem is to associate the view with the nodes of the tree. In this approach we limit the graphical display of a process to two successive hierarchical layers. One (composite or primitive) system is chosen as the 'displayed' system, here called the focus system (Westerweele 1999). This splits the overall model into two parts; the focus system and its environment. The focus system shows its subsystems (if it has any) in a frame. The environment systems of the focus system are displayed outside this frame. The following example illustrates what this means in terms of graphical representation:

Example 8.1: ***Visualisation of a Process***

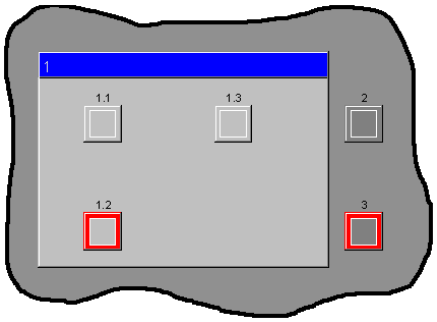

Figure 8.3: *Graphical representation of the tree with system 1 as the focus system. The connections between the systems are not shown.*

> *Consider a process that can be hierarchically decomposed in a tree of systems as represented in figure 8.2. The complete process, which is modelled as a thermodynamic universe, is represented by the root node. The root node is composed of three subsystems, '1', '2' and '3'. These three systems together also represent the complete process. So, if we chose system '1' as the focus system, we can graphically represent the whole thermodynamic universe c.q. the complete process by showing only the three subsystems. Figure 8.3 shows that graphical representation. The systems '1.1', '1.2' and '1.3' build up the focus system '1' and are therefore located inside the window frame which constitutes the focus system. In order to represent the complete process, the systems '2' and '3' have to be shown as environment systems of system '1'. These systems are drawn outside the focus system window, because they are not part of the focus system. In this way the whole process is graphically represented by showing one system (the focus system) in detail, and the rest of the process (the environment systems of the focus system) with little detail.*

□

The example shows that in this hierarchical representation of systems, information about the structure of the lower levels is hidden.

8.2 Unique System Identifiers

It is possible to give each system (either composite or primitive) a name, such that a model designer can keep an overview. To help the model designer

with the handling of complex processes, a unique label c.q. identifier for each system is introduced. An example of these unique identifiers is already shown in the figures 8.2 and 8.3. The definition of a hierarchical tree of identifiers facilitates a notation for the representation of an arbitrary hierarchical system. The described tree structure does not impose any limitations on the depth of the tree nor on the number of branches from each node. Every node is uniquely labelled by an identifier.

Each identifier consists of a sequence of numbers, which are the branch numbers that must be chosen to arrive in the node being identified, starting at the root node of the tree. So, a node identifier uniquely describes the location of the node in the tree.

In the Modeller, the root node of a process is always indicated with '0' or 'Top Level', because this node represents the thermodynamic universe. The first subsystem of the root is always identified by '1', the second by '2', etc. The first system of the third subsystem of the root will be labelled '3.1'. The first subsystem of the latter will have '3.1.1' as identifier, etc., etc.

The node identifiers are thus algebraically constructed by concatenating the consecutively chosen branch numbers. The 'deeper down' a node is located in the tree, the 'larger' its identifier will be (or the more numbers its identifier will comprise). Whenever the (tree structure of the) physical topology of a process is altered, the Modeller will update all identifiers of the systems that were influenced. In this way the identifiers of the systems (nodes) won't loose their meaning and the systems identifier still describes the location of the system in the tree.

8.3 Basic Tree Operations

A physical topology is organised as a strictly hierarchical tree. This tree can be built using two main operations, namely by either refining an existing system (the top-down approach) or by grouping systems (the bottom-up approach). The operations that are used for the refining and grouping, and some other operations that are necessary for the manipulation of the physical topology will be discussed with the aid of simple examples. The examples will illustrate the influence of the operations on the tree structure and on the graphical representation of the tree.

- **Adding an Elementary System.**

  When constructing a physical topology for a process, one of the most important operations is of course the adding of new systems. A new elementary system is always inserted as an elementary system with a

user defined name. The new system will be inserted as a subsystem of the focus system. As a rule a new system is always added to the right of the existing subsystems (if any) of the focus system. If the focus system does not have any subsystems yet, it is still a simple or primitive system. The insertion of a new system will turn the focus system into a composite system with one subsystem.

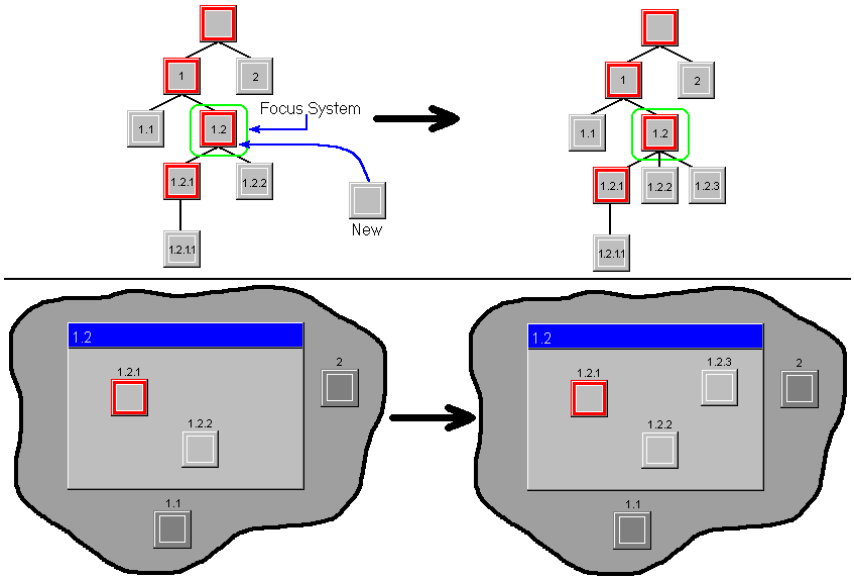

Figure 8.4: *Adding a new system*

- **Selecting Systems**

  For a number of operations, such as deleting, moving, grouping, dragging and copying, it is necessary to have the facility of selecting systems. When grouping or copying, only subsystems of the focus system (i.e. internal systems) can be selected for reasons of keeping the (tree) representation consistent.

- **Removing Selected Systems**

  Any system, composite or elementary, can be removed from a tree. After the selected systems have been removed, the system identifiers will be automatically updated. Care should of course be taken when one removes one (or more) composite systems, because all the subsystems of that composite system will also be removed and the information of this part of the tree will be lost.

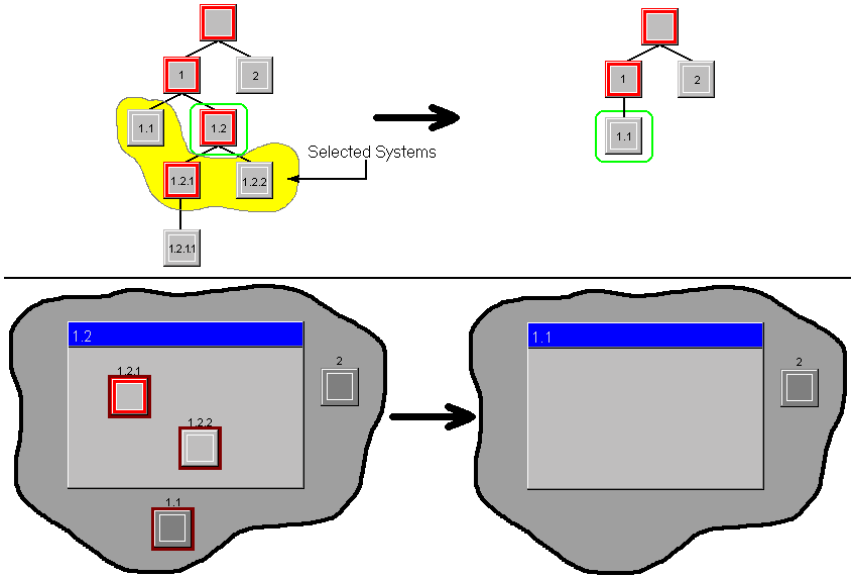

Figure 8.5: *Removing systems.*

- **Grouping Selected Systems**

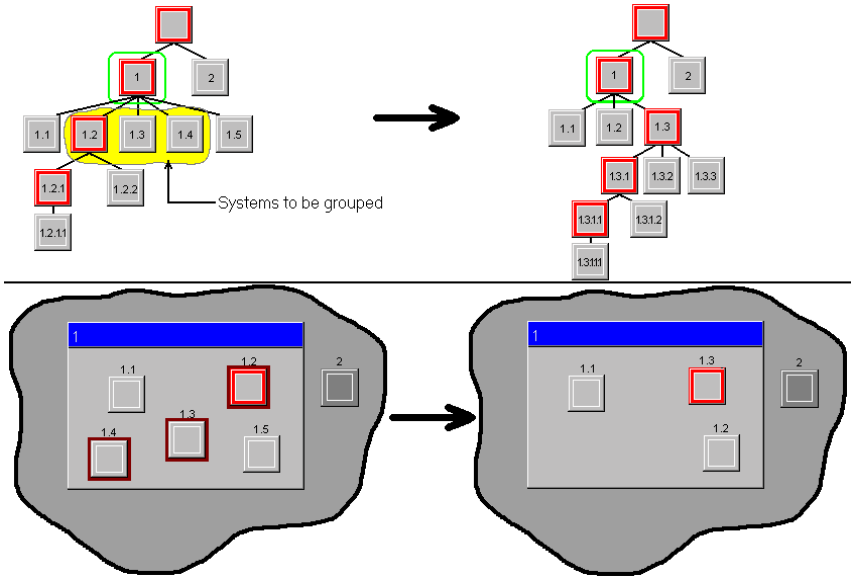

Figure 8.6: *Grouping systems*

As the number of subsystems in a system increases, the management

of the subsystems is likely to become easier by grouping a set of those subsystems and introducing an intermediate system to represent them. This operation is called grouping. The grouping operation can be defined by a series of previously discussed operations. First we select the subsystems of the focus system that we want to group. Then the selected systems are removed and a new system is inserted to the right of the remaining subsystems (if any). The removed systems are finally reinserted, but now with the new system as their parent system. Grouping systems does not alter the (flat) physical topology of the process. It is merely a matter of convenience and management of the hierarchy.

- **Degrouping a System**

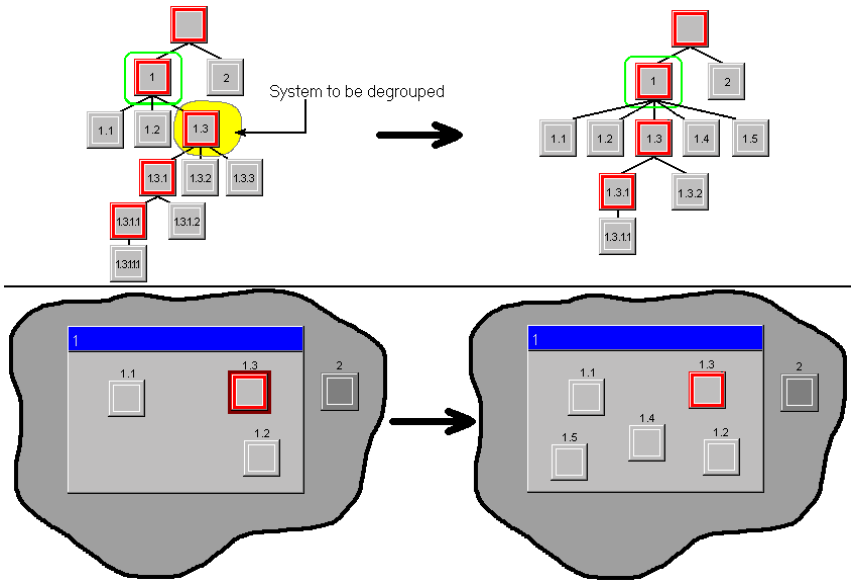

Figure 8.7: *Degrouping a system.*

The degrouping operation is the inverse of the grouping operation. In this case an intermediate system, i.e. the system that is going to be degrouped, is removed after the subsystems of that intermediate system have become subsystems of the intermediate systems parent. Elementary systems can, of course, not be degrouped, because they do not have any subsystems.

- **Copying or Saving Selected Systems**

The selected systems can be copied. This operation makes an exact copy of the selected elementary and/or composite systems, and stores

them in a buffer. For reasons of compatibility and consistency, only a set of systems that are subsystems of the focus system can be copied (or selected for copying). The saving of the selected systems is basically the same operation as the copying, with the difference that the systems are now stored on a disk, so that they can be used any time for the construction of any physical topology.

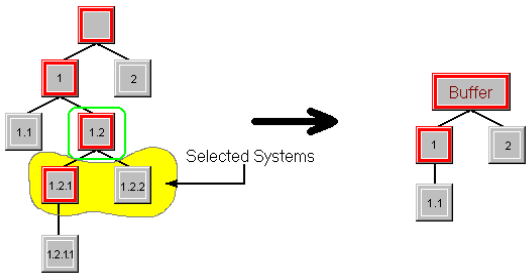

Figure 8.8: *Copying/exporting systems*

- **Pasting Copied or Imported Systems**

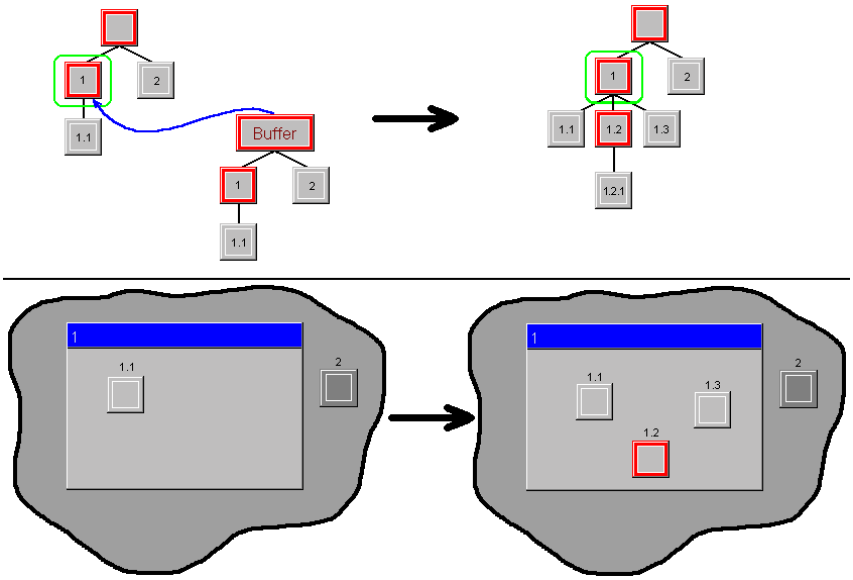

Figure 8.9: *Pasting/Importing Systems*

The pasting of copied or (from disk) imported systems is an operation that is very similar to the insertion of a new system. The paste operation

adds the systems which are stored in the buffer or on disk as subsystems of the focus system. Again the rule applies that "new" systems are always added to the right of the existing subsystems.

- **Moving a System**

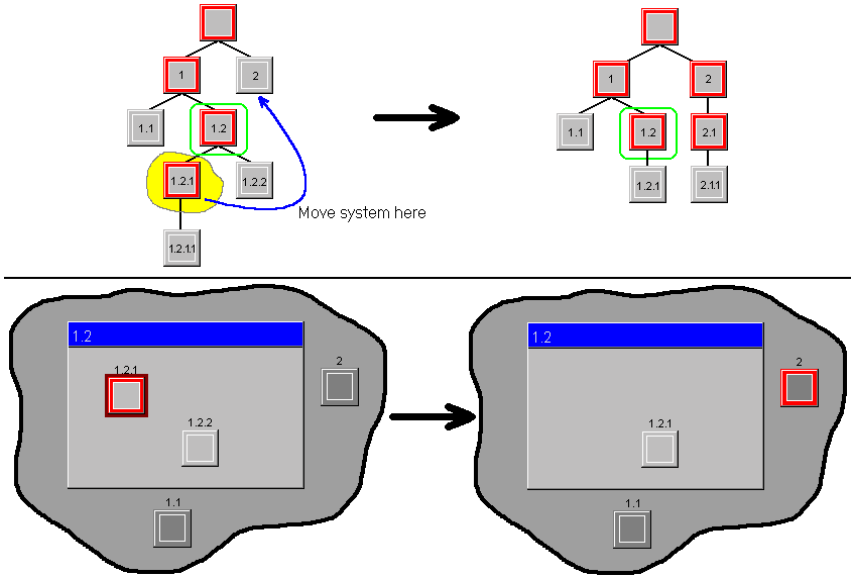

Figure 8.10: *Moving a system within the tree structure.*

Sometimes it is necessary to move a system within the tree structure. The new parent of the system can be any system of the tree that is not a subsystem of the system that is moved.

After each operation the identifiers of all the systems that were influenced by the operation, are updated.

8.4 Representation of Connections

The graphical representation we employ to represent a hierarchical tree does not show every part of the model in detail. Most parts of the model are shown with very little detail, hiding the structure of their lower levels. Consequently, not every connection will be visible in this representation. Only connections which have an origin and/or target that is a subsystem of the focus system, will be displayed. Connections between two environment systems are not shown for reasons of surveyability. With this approach, we can distinguish between two "classes" of graphical connections, namely internal and external connections.

An internal connection is a connection between two systems that are both subsystems of the focus system. An external connection has one system as origin or target which is not a subsystem of the focus system.

Connections are defined between two elementary thermodynamic systems, but in our representation it is not always possible to show both of these systems at the same time. Therefore, it is necessary to introduce "(graphical) connections" between composite systems. What this means will become clear in the following sections.

8.4.1 Making a New Connection

A correctly defined connection is always defined between two elementary systems. If a model designer wishes to generate a new connection, he must first define the type of connection he wishes to generate. In other words, the modeller must define what kind of extensive quantity will be transferred through the new connection. For now the choices are limited to: a mass, heat or work connection (and information connections for the 'transfer' of information). Then he must select a system that will serve as the origin of the connection and, subsequently, he must select a target system.

For ease of establishing a connection, the modelling tool automatically invokes a zoom-in operation upon the selection of a composite system as either origin or target system. This means that the selected composite system will be set as focus system and the model designer can select a subsystem of this system. This is done because a connection always has to be defined between elementary systems.

8.4.2 Graphical Representation of Connections

The graphical representation of connections is best explained with the aid of a simple example.

Example 8.2: ***Graphical Representation of Connections***

> *Consider a process for which the tree structure representation is defined as in figure 8.2. Let the following connections be given between the elementary systems (see figure 8.11a; see figure 8.1 for a flat topology of this example process):*

Connection number	*Origin System*	*Target System*
1	*1.1*	*1.3*
2	*1.1*	*1.2.1*
3	*1.2.1*	*1.2.2*
4	*2*	*1.3*
5	*1.2.2*	*3.1*
6	*3.2*	*2*
7	*1.3*	*3.2*

If we take system 1 as the focus system these connections will be graphically represented as shown in figure 8.11b.

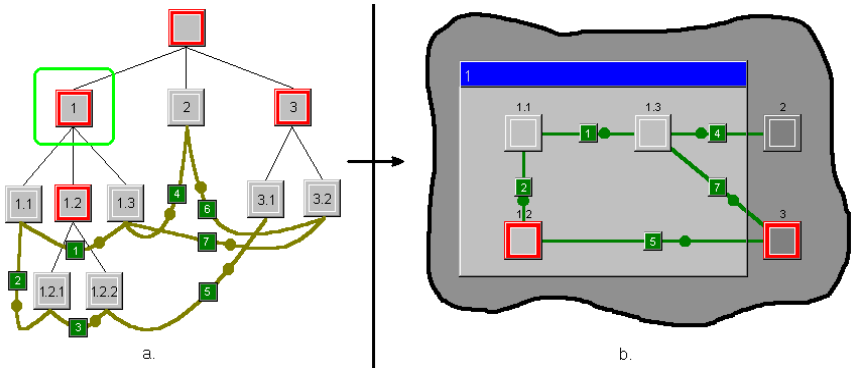

Figure 8.11: a. *Tree structure with connections between elementary systems.* b. *Graphical representation with system 1 as the focus system.*

The following remarks can be made concerning these connections and their representation if system 1 is the focus system:

- *Connections '1' and '2' are both internal connections of system 1 (and are both external connections of system 1.1). Connection '1' is shown as a connection between two elementary systems. For connection '2' this is not possible, because the target of the connection (the elementary system 1.2.1) is not displayed on the screen. The connection is therefore displayed as a connection between the elementary system 1.1 and composite system 1.2, for system 1.2.1 is a subsystem of system 1.2.*
- *Connection '3' is an internal connection of the system 1.2 and is not displayed on the screen because both the origin and the target of this connection are subsystems of system 1.2. If one wants to show this connection on the screen, one must zoom*

in on system 1.2 (and set this system as the focus system). So, only internal connections of the focus system are shown on the screen. Internal connections of any other system are always hidden.

- *Connections '4', '5' and '7' are external connections of the focus system 1, because these connections cross the boundary of system 1.*
- *Connection '6' is not shown in this representation, for this connection is established between environment systems of the focus system 1.*

Figure 8.12 shows the graphical representation of the process with the tree root as the focus system.

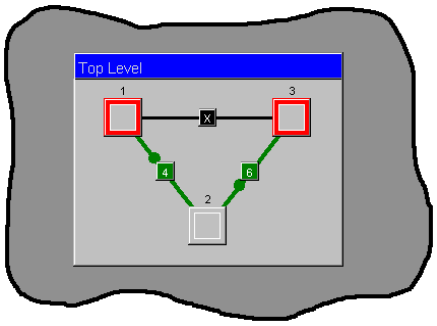

Figure 8.12: *Graphical representation with the tree root as the focus system.*

Remarks:

- *As you can see, a problem arises here, for connection '5' and '7' are both defined between the (composite) systems 1 and 3. In the graphical representation it is displayed as only one connection 'X', while it in fact is a "composite" or "multiple" connection, consisting of the connections '5' and '7'. In this way an arbitrary number of connections between two (elementary or composite) systems can be defined and represented. If every single connection would have to be displayed in the graphical representation, the screen could become very crowded and one would lose a certain amount of surveyability. Imagine, for example, the display of ten connections between a pair of systems.*

- *Information about the connections contained in a "composite" connection can be easily accessed with the modelling tool.*
- *"Single" connections show the user-defined directionality of the connection graphically and composite connections do not (and can not). The directionality of the connections of a composite connection can be easily accessed with the modelling tool. Remember, however, that the origin and target of a connection only define a reference co-ordinate system for the actual flow (of extensive quantities). The actual direction of the flow depends on a difference in potential between the systems.*
- *All defined systems are subsystems of the tree root and consequently there are no external systems and no external connections.*

□

8.5 Consequences of Manipulations on the Physical Topology

Manipulation of the physical topology can give rise to two kind of problems with the already defined connections. The first arises when one wishes to refine an elementary system which is connected to other systems. The second problem occurs when one deletes, copies or saves a (composite or elementary) system which has one or more external connections. Both of these actions will cause the involved connection to be, at least temporary, not properly defined or "undefined". The two problems give rise to two different kinds of "undefined" connections, namely "*loose connections*" and "*open connections*". The following will discuss how these two types are handled in the modelling tool.

8.5.1 Loose Connections

Loose connections show up when one wants to refine an elementary system that has connections. This means that one zooms in on an elementary system, which is already connected to other systems, in order to add subsystems to this system and thus change it into a composite system. As soon as one or more subsystems are inserted to an elementary system with connections, all these connections will loose their meaning/relevance because they are now no longer defined between two elementary systems.

A solution to this problem is to introduce (temporary) loose connections. A *loose connection* is defined as a connection of which one or two ends are

connected to a composite system. A loose connection thus has no physical meaning, it is just a temporary state of a connection to help the model designer in fast and easy modelling. The "loose ends" of a loose connection have to be reconnected to a subsystem of the system the connection was originally connected to, in order to form a properly defined connection again.

Example 8.3: ***Loose Connections***

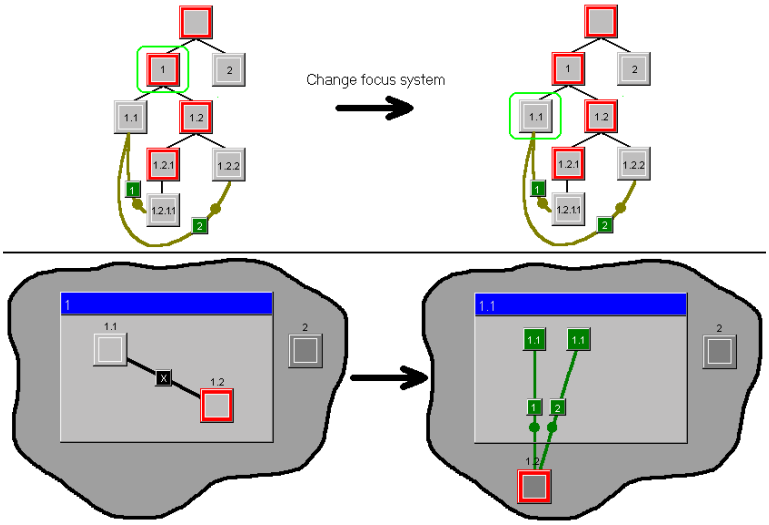

Figure 8.13: *Appearance of loose connections, when zooming in on a connected elementary system.*

> *Consider a system (system 1 in figure 8.13) which is connected to two other systems (systems 1.2.1.1 and 1.2.2). If we now zoom in on system 1.1 (as shown in figure 8.13), the connections that are connected to the system 1.1.are shown as loose connections, but they are, in fact, still correctly defined connections, because system 1.1 is still an elementary system. If we would add a new system to system 1.1, the connections would loose their meaning, because system 1.1 would then turn into a composite system and connections must always be defined between two elementary systems.*
>
> *The "1.1" in the temporary endpoints of the loose connections denote that the loose connections* have to *be reconnected to a subsystem of system 1.1. The reconnection of the loose connections (if necessary) can be done at any point during the model definition.*

□

8.5.2 Open Connections

An *open connection* is defined as a connection that is only connected to one system. The other system of the connection (either the target or the origin) is not yet defined. Connections that are not connected to any system are no connections and therefor do not exist. So, if the result of a manipulation on the physical topology is an open connection with no connected systems, then this connection is deleted.

Just like a loose connection, an open connection has no physical meaning and is just a temporary state to help the model designer. Open connections can appear by one of the following actions:

- Deleting a system that is connected to systems that are not deleted.
- Copying or exporting a set of systems that are connected to systems that are not being copied or exported. The copied systems will contain connections that are open because these connections are only connected at one side.
- Deliberately making a new connection that is only connected to one system (either the origin or the target of the connection). This can be of help, for example, in the construction of not finished, but reusable model parts, such as heat exchangers. In the overall model, a shell and tube heat exchanger must be connected to a source and sink for both the contents of the shell and tube. The heat exchanger as a stand alone model is not (yet) connected. If a model designer imports the model of the heat exchanger into another model, he will not forget to properly connect the heat exchanger to the existing model, because the open connections of the heat exchanger have to be reconnected.

Open connections may be reconnected at any time to any elementary system of the physical topology. They can, of course, also be deleted.

A special feature of Modeller that only applies to open connections is that they can be hidden in order to increase the surveyability when the model designer is not focussing on the open connections.

Example 8.4: ***Open Connections***

Figure 8.14 shows that by deleting system 1.1 three open connections will form. Standard, only those open connections are shown for which the connected endpoint is visible on the screen. In this example this means that only connections 2 and 3 are visible and connection 1 is hidden, because system 1.1.1.1 is not visible on the screen. It is possible, though, to show all the open connections on the screen.

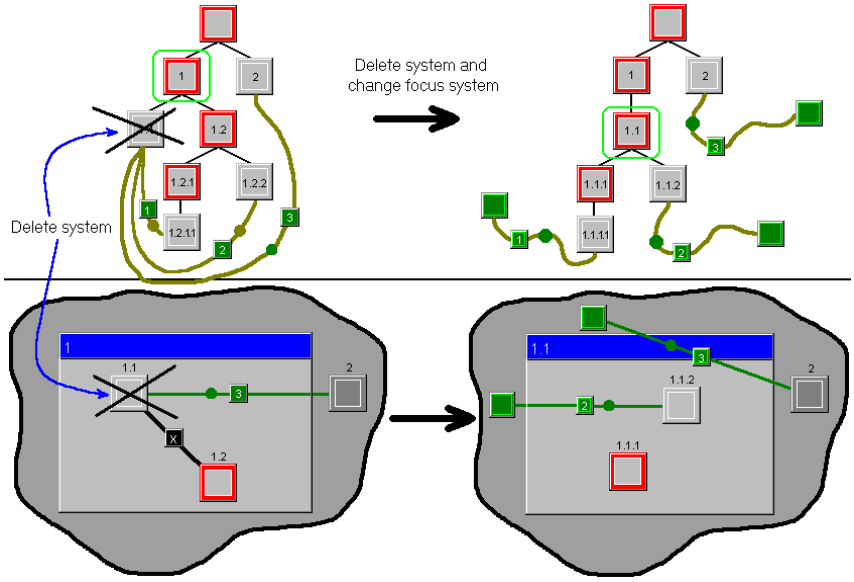

Figure 8.14: *Appearence of open connections, when deleting a connected elementary system.*

In contrast to loose connections, the temporary endpoints of the open connections do not contain a number. This means that an open connection can be reconnected to any elementary system of the physical topology.

□

8.5.3 Possible States of Connections

As mentioned, connections can have different (temporary) states to help the model designer with the construction of process models. The final model can, of course, not contain any "undefined" connections, so these have to be dealt with before the model designer generates the end result. Figure 8.15

summarises the possible temporary states of connections, during the model construction phase:

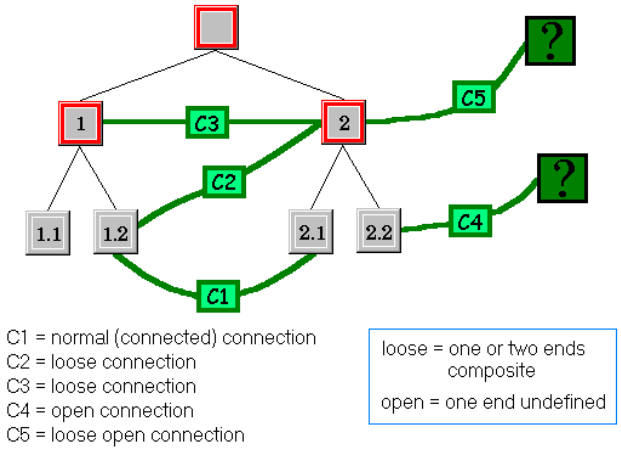

Figure 8.15: *Possible states of connections.*

Remarks:

- The loose end of connection C2 (i.e. system 2) can only be reconnected to (elementary) subsystems of this composite end (i.e. system 2.1 or 2.2). The same accounts for the loose ends of connections C3 and C5
- The open ends of connections C4 and C5 can be reconnected to any elementary system (except the (elementary) system it is already connected to (system 2.2 in case of reconnecting connection C4)).

8.6 Repetitive Structures

An additional advantage of the temporary states of the connections is that these can be used for the definition of so-called repetitive structures. These repetitive structures can be used to approximate the behaviour of distributed systems or to build large, interconnected physical topologies in a very quick manner. The implementation of repetitive structures in Modeller is limited (for now) to one-dimensional repetitions. Expansion to two and three-dimensional repetitions should, in principle, be trivial.

The following example illustrates the implementation of repetitive structures. In principle, the repeated structure can be a tree of any size and depth with any kind of interconnection, but to keep things simple we use a 'flat topology' as repetitive structure.

Example 8.5: ***Repetitive Structures***

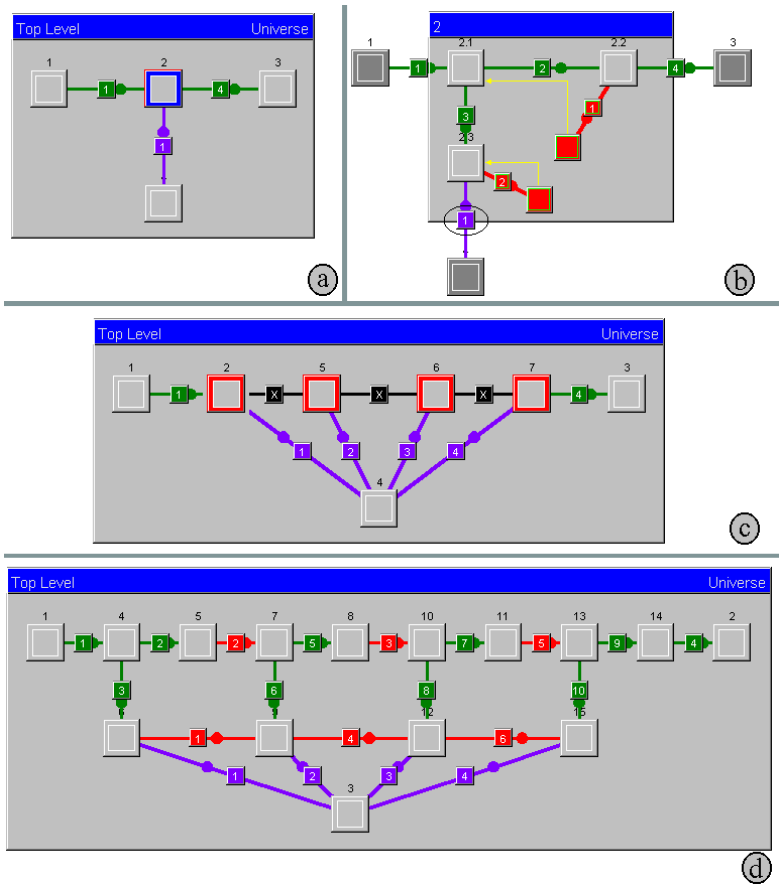

Figure 8.16: *Repeating a repetitive structure. a-b) repetitive structure. c-d) repeated repetitive stucture.*

Consider the physical topology shown in figures 8.16a and 8.16b. The physical topology consists of 6 systems (1, 2.1, 2.2, 2.3, 3 and 4), 4 mass connections, 2 heat connections and 1 work connection. The system 2 is the one that is to be 'repeated' (three times, in this example). In this example, the heat connections are used for the interconnection of the repeated structures. The 'repetitive system' of heat connection 1 is system 2.1, which means that the connection will be connected to the system 2.1 of the next copy when the structure is repeated. The repetitive system of heat connection 2 is system 2.3. The (encircled) work connection is set to be 'connected to all repeated structures'. Mass connections 2 and 3 are

internal to the structure and will be copied and be internal to each copy of the repeated structure. Mass connection 1 has its target in the repeated structure and will only be connected to the first of the repeated structures. Mass connection 2 has its origin in the structure that is to be repeated and will only be connected to the last copy, when the structure is repeated.

Figure 8.16c shows the result when system 2 is repeated three times. Figure 8.16d shows the result of the repetition when the composite systems (2, 5, 6 and 7) shown in figure 8.16c are degrouped.

□

The repetive structure building function can seriously speed up the construction of larger models, because, when the repetitive stucture is properly initialised, the resulting structure will also be properly initialised.

Chapter 9

Construction of the Species Topology

The definition of the species topology of a process is initialised by assigning sets of species (and/or reactions) to some systems. Species, as well as reactions should be selected from corresponding databases. So, before the species topology can be defined, a species and a compatible reactions database must be defined. Such a database contains a list of species and a list of possible reactions between those species. A species and reactions database should, of course, be editable by the user in order to satisfy the specific needs of the user.

In the following, the complete process the model designer wishes to develop is referred to as the plant.

9.1 Defining Plant Species and Reactions

Processing plants involve different types of processes, such as mineral, petrochemical, pharmaceutical, polymeric, or biological with various species and reactions. The definition of the species in various parts of a processing plant requires data about these species. This data can be found in species and reactions databases. In general, a processing plant does not involve all the species and reactions that are listed in a database. Therefore, a model designer has to select a subset from this list, which comprises all the species and reactions that appear in the (model of the) plant and which thus serves as the "database" for that particular plant. This plant database is then a collection of all species and reactions that may exist in the plant. The selected subset of species will be called the *plant species*. The selected reactions will be called the *plant reactions*. These subsets have to be formed to improve the surveyability and to prevent unnecessary computations involving species and reactions that are not present in the plant.

For the construction of the species topology, the species name, its formula or more specifically its index in a database is sufficient, but the database can and must, of course, contain other information, such as specific physical properties. Similarly for reactions, only information about which reactants and which products are involved in a reaction, is sufficient for the construction of the species topology. However, stoichiometric coefficients and other reaction data are most likely to be connected with the database.

With Modeller, species and reactions can be selected from one or more user specified databases to form the plant species and reaction sets. To aid a model designer in defining a plant species and reaction set for his process, the following actions can be performed:

- **Adding species** to the plant species set. The user specified species will be copied from the database and added to the plant species set.
- **Adding reactions** to the plant reaction set. Whenever a new reaction is added to the plant reactions set, the plant species set will automatically be extended with all involved species(i.e. the reactants and products of the reaction),which were not yet listed in this set.
- **Removing reactions** from the plant reaction set. The user specified reactions will be removed from the plant reaction set and will consequently also be removed from all the systems in which they were injected.
- **Removing species** form the plant species set. Whenever a species is removed from the plant species set, all reactions of the plant reactions set, which contain this species (either as a reactant or a product), will be removed from the plant reactions. The species and reactions that were removed from the plant sets will, of course, also be removed from all the systems they were used in.

9.2 Injecting Species and Reactions

The definition of the species topology is initialised by "injecting" species and/or reactions into elementary systems. This means that the model designer has to assign a set of species and/or reactions to some elementary systems. These species can only be selected form the defined plant species and reactions, since these sets represent all species and reactions that may exist within the plant.

After the assignment of the *injected species* and *injected reactions* to a specific system, the modelling tool will (re)calculate (parts of) the species distribution. This means that the species will propagate into other systems

through mass connections. Within the systems, the species may undergo reactions and generate "new" species, which in turn may propagate further and initiate further reactions. This eventually results in a specific species distribution over the elementary systems, which is referred to as the *species topology* of the processing plant. The distribution process will be clarified in section 9.4.

The foregoing made clear that the species that are actually associated with a system can have either one of the following sources:

- **Initialised species**: injected to a system by a model designer
- **Transferred species**: arrived through one of the mass connections
- **Product species**: result of a reaction.

The reactions that are associated with a system can only be the result of a manual injection by a model designer.

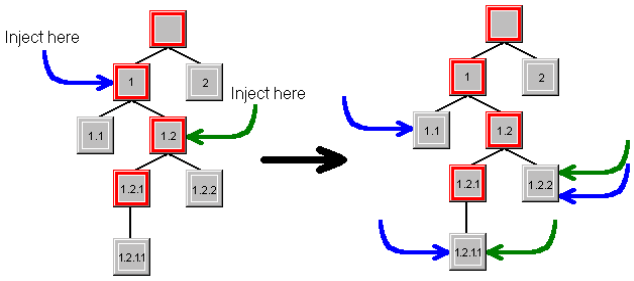

Figure 9.1: *Injecting of species and/or reactions at composite systems.*

Species and reactions may also be injected at composite systems. The injected species and/or reactions will then actually be injected in all the elementary systems that are subsystems of the composite system (see figure 9.1). The same accounts for the removal of injected species or reactions: If a species or reaction is to be removed from a composite system, then this species or reaction will actually be removed from the injected species or injected reactions sets (only if the specific species or reaction is listed, of course) of all the elementary systems that are subsystems of the composite system (see figure 9.2).

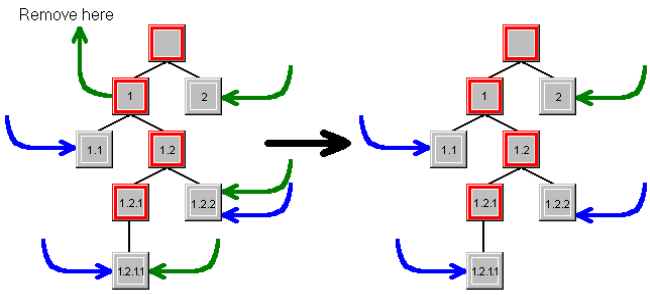

Figure 9.2: *Removing of injected species and/or reactions from a composite system.*

The injection of a reaction into an elementary system does not automatically imply that this reaction can "happen" in this system and thus that the products of this reaction can be formed. If not all reactants of a reaction are available in a system, then this reaction *cannot* take place in this system. In such a case, the system will have an injected reaction but this reaction will not be active (it is disactivated). So, the reaction will not take place in the system in this case. When the species distribution is changed and the reaction can take place again, it will automatically be "re-activated".

It should be noted that the presence of an "activated" reaction in a system does not imply that this reaction *has* to happen in this system. It implies that this reaction *may* happen in this system.

9.3 Permeability and Uni-Directionality of Mass Connections

In the physical topology of a plant, mass connections establish the communication paths between pairs of systems, without specifying which species may or may not be transferred. As mentioned, permeability and uni-directionality are introduced as properties of mass connections. They constrain the mass exchange between systems by making the species transfer respectively selective or uni-directional.

The permeability of a mass connection can be defined in two complementary forms:

- **Permeable Species Set:** This set includes all species that may pass through the mass connection. All other species of the plant species set can not be transferred via this mass connection.

- **Non-Permeable Species Set:** This set is formed by selecting all species that are not be transferable through the mass connection.

Both complementary approaches are enabled with the modelling tool since not having one or the other approach may force a model designer to (repetitively) define large sets of species.

Often, most connections in a model of a process represent convective flows, which by definition communicate all species. Therefore, the default permeability of a mass connection is defined by an empty non-permeable species set and thus as permeable for all species.

Which species are actually transferred through a connection depends on the species sets of the two connected systems. In reality, it also depends on the actual transfer direction, which depends on the states of the two connected systems. Since the latter information is not available during the construction of the physical and species topology, mass connections are assumed to transfer potentially in both directions (by default). Species may therefore exist on both sides of a connections, that is, if a species appears on one side of a connection and it is not inhibited by a defined permeability, it may also appear on the other side. This policy is very suitable for safety and hazard studies, as the resulting species topology is a maximum solution, showing where the different species *may* occur (and showing which reactions are potentially possible) in different parts of the process' model. This approach, though, will generate more equations than necessary for the description of common processes. That is why the concept of *uni-directional* mass transfer is introduced. This can reduce the number of equations, because species are only allowed to propagate in one direction when the species distribution is (re)calculated and this can reduce the number of species in an involved system. The resulting species topology, based on this policy, could reflect more closely the actual situation in the process.

9.4 Propagation of Species

Whenever an operation is executed which modifies the current species distribution, a mechanism has to be activated which updates the species distribution over all elementary systems of the affected mass domains. To facilitate in the definition of such a mechanism, every elementary system contains:

- a list of injected species (here called: *Injected-Species*).
- a list of 'actual' species (*Species-List*).
- a list of injected reactions (*Injected-Reactions*).

- a list of 'actual'or 'active' reactions (*Reaction-List*).

The update mechanism employed in the **Modeller** makes use of the following algorithm:

```
PROCEDURE UpdateSpeciesTopology (Mass-Domain)

FOR ALL elementary systems OF Mass-Domain DO
    CLEAR Species-List
    CLEAR Reaction-List
END

FOR ALL elementary systems OF Mass-Domain DO
    FOR ALL injected-species OF system DO
        IF injected-species IS NOT IN Species-List OF system THEN
            PropagateSpecies (injected-species, system)
        END
    END
END

END UpdateSpeciesTopology;
```

This procedure first clears all the information about the current species distribution of the affected mass domain (*Mass-Domain*), by removing all species and reactions from the actual species and reaction lists of all involved elementary systems. Then the procedure injects all the species which are listed in the lists of injected species of the primitive systems. Remember that all injected species of a specific system were assigned manually by a model designer and stored in the injected species list of the corresponding system.

The injection of the species is done by making use of the procedure PropagateSpecies. This procedure takes care of the propagation of species and the initiation of set reactions. The procedure has a specific system and a specific species as argument. The algorithm for this procedure can be described as follows:

```
PROCEDURE PropagateSpecies (Species, System)

ADD Species TO Species-List OF System

FOR ALL injected-reactions OF System DO
    IF reaction NOT IN Reaction-List OF System THEN
        IF ReactionIsPossible(reaction, System) THEN
            ADD reaction TO Reaction-List OF System
            FOR ALL products OF reaction NOT YET IN Species-List OF System DO
                PropagateSpecies(not-listed-product, System)
            END
        END
    END
END

FOR ALL connected-mass-connection OF System DO
    IF SpeciesMayPassThrough(connection, Species, System) THEN
        IF Species NOT YET IN Species-List OF other-connected-system THEN
            PropagateSpecies(Species, other-connected-system)
        END
    END
END

END PropagateSpecies
```

The PropagateSpecies procedure is a more complicated procedure then the UpdateSpeciesTopology procedure. This is partly due to the fact that the procedure is a recursive procedure (the procedure calls itself). The procedure has the argument "(*Species*, *System*)". This means that this procedure uses the species and system given in the argument for some specific operations, which have to be performed with this species and system.

The procedure starts with adding the species "*Species*" to the species list (*Species-List*) of the elementary system "*System*". After that, the procedure checks for every injected reaction of the system "*System*" wether it is already listed in the reaction list (*Reaction-List*) of the system. Whenever an injected reaction (*reaction)* is not yet listed in *Reaction-List*, the procedure ReactionIsPossible checks if the system *System* contains all the reactants of that not yet activated reaction. If all reactants of the reaction are available in the system, then this means that this reaction may happen in the system and the reaction will be "re-activated" by adding it to the *Reaction-List* of the system. As the reaction is possible now, it may produce species that are not yet listed in the *Species-List* of the system. Any species that is not yet listed in the list, will be added to the system *System* by calling the procedure

PROPAGATESPECIES. This means that the procedure will call itself, but with different arguments. The "first" PROPAGATESPECIES will continue after the "second" has finished. Within the "second", of course, a "third", "fourth", etc. can be called.

When all injected reactions of the system "*System*" have been checked, the procedure continues with checking all mass connections of the system. The procedure SPECIESMAYPASSTHROUGH examines wether the species *Species* may pass through the current mass connection. The species can be prohibited to pass through the connection in two ways: it can be inhibited by a defined permeability or the connection in uni-directional and the flow of species is in the other direction. If the mass connection permits the species *Species* to pass through to the system that is connected to the system *System* via this connection, the procedure will check if this species is already listed in the species list of the other system of the connection. When this is not the case, PROPAGATESPECIES will be invoked with the species and this other system as argument. In this way the species "*Species*" will propagate to all the connected systems and will be properly distributed.

The following example will illustrate the working of the update procedure.

Example 9.1: ***Automatic Species Distribution***

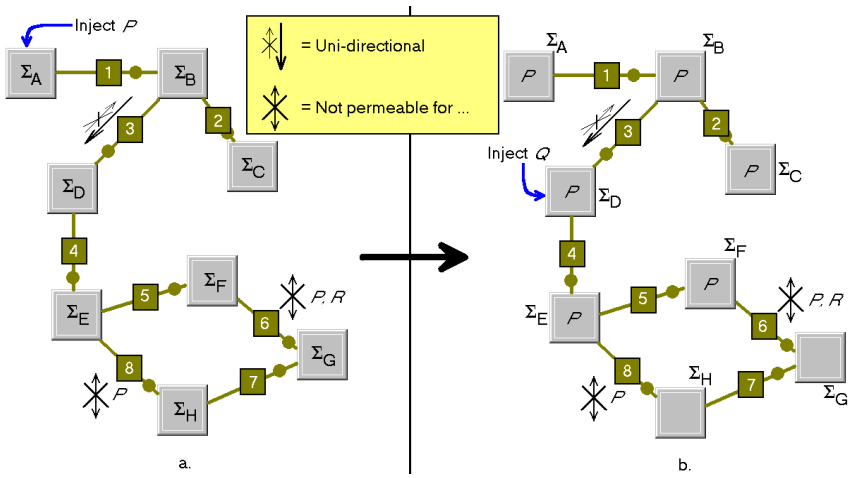

Figure 9.3: *a). Flat topology of the process, b). Distribution of species P from its injection point to other connected systems.*

Figure 9.3a shows the flat topology of a process. The plant species set of this process model consists of the species P, Q and R, and the list of possible reactions contains only the reaction: $P + Q \rightarrow R$. *Between the systems* (Σ_A *through* Σ_H) *some mass connections*

('1' through '8') are defined, some of which are uni-directional or impermeable for some species: Connection '3' is uni-directional, connection '6' is impermeable for species P and R, and connection '8' for species P.

The species P is injected in system A. When the species topology is updated, the update procedure executes the following major steps:

1. *All species and reaction lists of all elementary systems are cleared.*
2. *System Σ_A is the only system of the mass domain with an injected species, namely the species P. This species is added to the species list of system Σ_A by calling the procedure* PropagateSpecies*(P, Σ_A).*
3. *There are no injected reactions defined yet, so all operations involving reactions are skipped.*
4. *The procedure checks all mass connections of Σ_A, which are permeable for species P. Connection '1' is permeable for P and has system Σ_B as other connected system. System Σ_B does not yet have species P in its species list, so* PropagateSpecies*(P, Σ_B) will be invoked.*
5. *The procedure* PropagateSpecies*(P, Σ_B) checks all mass connections of Σ_B. The first one is connection '1', which has Σ_A as other connected system. Σ_A already has species P in its species list, so nothing happens.*
6. *The next connection of system Σ_B is checked. Connection '2' has system Σ_C as other connected system, so* PropagateSpecies*(P, Σ_C) is called.*
7. *This procedure will again check all the connections of the specified system, here Σ_C. Σ_C only contains one connection. This connection is connected to the system Σ_B, which already has the species P listed in its species list. This means that the procedure* PropagateSpecies*(P, Σ_C) ends and the procedure* PropagateSpecies*(P, Σ_B) will continue.*
8. *The procedure continues by checking the next connection, connection '3' and thus by invoking* PropagateSpecies*(P, Σ_D).*

Etc. The update mechanism will continue in this way, until the complete species topology is updated. The resulting species distribution is represented in figure 9.3b.

Then species Q is injected in system Σ_D. If the update mechanism is activated again, then the updating procedure starts again with system Σ_A, were it finds the injected species P. This means that the species P will distribute itself again as described above. After the distribution of species P, the UpdateSpeciesTopology *procedure will check for other injected species in system Σ_A and check for injected species in other systems. When the procedure gets to system Σ_D it will find species Q as injected species. This means that the procedure* PropagateSpecies*(Q, Σ_D) will be invoked, and that the species Q will distribute itself in the same way as species P. The resulting species topology is represented in figure 9.4a.*

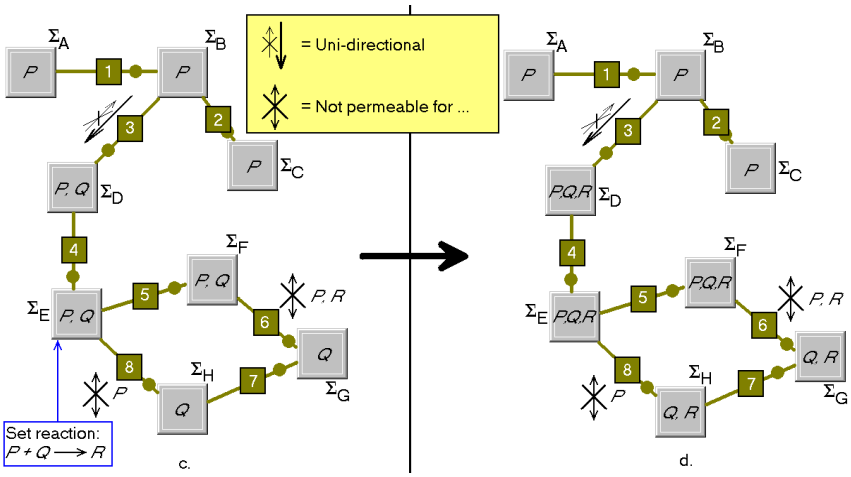

Figure 9.4: a. *Distribution of species P and Q, Distribution of species P, Q and R.*

As one can see in this representation, there are three systems, Σ_D, Σ_E and Σ_F, which contain both species P and Q. In those systems the reaction $P + Q \rightarrow R$ could be set active. If the reaction is injected in system Σ_E, then the update procedure will be invoked. The procedure starts with clearing all the species and reaction lists. Then the procedure continues with the distribution of species P from system Σ_A. When the update mechanism gets to system Σ_E, it will check if the reaction of this system is possible. The reaction is not possible yet, because only species P is listed in the species list of Σ_E at this time, which means that not yet all reactants are available. As a consequence the reaction will remain disactivated (i.e. not added to the reaction list) and the propagation of P continues.

After P is distributed, Q will be distributed again from system Σ_D. When the update mechanism gets to system Σ_E again, it will check if the reaction is possible. Both reactants are available in the system, which means that the reaction will be activated (i.e. added to the reaction list) and that the procedure PROPAGATESPECIES*(R, Σ_E) will be invoked. Consequently the species R will be distributed before the distribution of species Q continues. (The resulting species topology is displayed in figure 9.4b).*

When the update mechanism gets to system Σ_E again, it skips the reaction, because it is already activated. In this way no unnecessary computations are made. When one would remove the injected species P from system Σ_A or remove one of the connections '1', '3' or '4', the species P would be removed from the species list of system Σ_E. This would mean that the reaction $P + Q \rightarrow R$ cannot take place anymore in system Σ_E because not all the reactants are available. As a consequence the reaction would be disactivated and species R would not be generated and thus not be distributed. This means that the species R would vanish from the species topology. Re-introducing the species P to the system Σ_E would reactivate the reaction, and species R would again be generated and distributed.

□

9.5 Consequences of Manipulations

The application of some modification to the physical structure of a model, such as deletion of a system or the removal of a (mass) connection, will invalidate (parts of) the constructed species topology. The species distributions of all affected mass domains and the mass domains themselves are therefore reconstructed by the modelling tool after any significant change of the physical topology.

Also, modifications to the species topology itself will influence the species distribution. For example, the removal of an injected species, the "removing" of a reaction or the changing of the permeability of a connection will have an effect on (parts of) the species topology. Every action that affects the species distribution of a particular mass domain will invoke an update mechanism which will recalculate the species distribution of that domain.

Chapter 10

Construction of the Equation Topology

The equation topology forms a very important part of the modelling process, for with the information of this topology the complete model of the process is generated. The objective of the equation topology is the generation of a mathematically consistent representation of the process under the view of the model designer (who mainly judges the relative dynamics of the various parts, thus fixes intrinsically the dynamic window to which the model applies). In order to efficiently produce dynamic process models, the Modeller must, of course, appropriately deal with variables and equations.

This chapter deals with the implementation of the variables and equations that arise in the modelling of physical, chemical and/or biological processes when the structured modelling methodology, which was covered in part I of this thesis, is used. The chapter will not cover all implementation details of the equation topology. Only the most relevant topics (with respect to the modelling methodolgy) will be dealt with.

10.1 Balance Equations

In section 5.3 it was shown that the balance equations can be automatically generated when the first two steps of the modelling approach, namely the construction of the physical and species topology, have been completed. The matrix notation of the balance equations makes use of the so-called stream or interconnection matrices and stoichiometric matrices. Since we have defined three basic connections to describe the flow of the extensive quantities, three stream matrices can be derived from the physical topology: A mass stream matrix, a heat stream matrix and a work stream matrix. Only the extensive quantities of lumped systems (distributed systems are not considered) are bal-

anced, since for source, sink and steady state systems they are not defined. For steady state systems the balance equations are also required, but the accumulation terms are set to zero (see section 6.4), so only for lumped and steady state systems the stream matrices are needed. For the ease of representation and, consequently, the manipulations, all (elementary) systems are therefore sorted (i.e. numbered) in a list, in which first all lumped systems are listed. Consecutively the steady state, source and sink systems are listed. The list of all elementary systems of the process represents the complete process. The definition of composite sytems serves the ease of organising the process model and therefore composite systems are not listed.

Next, all correctly defined mass, heat and work connections are listed successively, excluding the open and loose connections as they are considered as non-existent. Each connection contains a "direction" vector, which denotes to which two systems the connection is connected. With this ordered information, the stream matrices ($\underline{\underline{\mathbf{A}}}$) of each type of connection can now easily be computed. For each type of connection the following function is called:

```
FUNCTION [A] = StreamMatrix (numLump, numStSt, connections)

numCon = LENGTH(connections)
IF numCon > 0 THEN
    A = EmptySparseMatrix(numLump + numStSt, numCon)
    FOR i = 1 TO numCon DO
        FOR j = 1 TO 2 DO      (* 1 = origin, 2 = target *)
            IF connections(i, j) <= numLump + numStSt THEN
                A(connections(i, j), i) = (-1)^j
            END
        END
    END
ELSE
    A = 0
END

END StreamMatrix
```

The function StreamMatrix has the number of lumped systems (*numLump*), the number of steady state systems (*numStSt*) and the connections themselves (*connections*) as an input. The output of the function consists of the stream matrix ($\underline{\underline{\mathbf{A}}}$). The term *connections* is defined as a (*number of connections*$\times$ 2) matrix. The first column of this matrix respresents the (numbers of the) origin systems of each connection. The second column refers to all the target systems.

Example 10.1: ***Construction of the Stream Matrix***

Figure 10.1 shows the flat topology of a process in which three lumps (systems 1, 2 and 3), two steady state systems (4, 5), a source (6) and a sink (7) are defined. The systems are already sorted in the above-described manner.

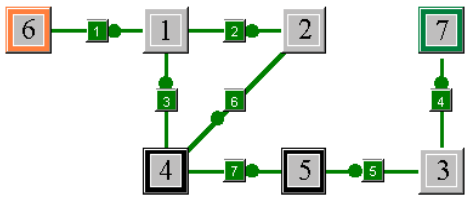

Figure 10.1: *Flat topology of a process with lumps (1,2,3), steady state systems (4,5), a source (6) and a sink (7).*

It can be easily derived from this figure that the matrix connections will be defined as follows:

$$connections = \begin{pmatrix} 6 & 1 \\ 1 & 2 \\ 4 & 1 \\ 3 & 7 \\ 3 & 5 \\ 2 & 4 \\ 4 & 5 \end{pmatrix} \tag{10.1}$$

and that numLump = 3, numStSt = 2 and numCon = 7.

The StreamMatirx *procedure starts by allocating a sparse empty matrix of dimension (5×7). Next it runs through all the connections: The number of the origin of the first connection (i = 1) is 6 (connections(1,1) = 6) and this is larger than numLump +numStSt, so nothing happens. The target (j = 2) of this connection has number 1 (connections(1,2) = 1), which is smaller than numLump + numStSt. Therefore, the entry of the stream matrix of the lumped systems at (1,1) will be adapted:* $\underline{\underline{\mathbf{A}}}(1,1) = (-1)^2 = 1$. *For connection 3 (i = 3) the origin has number 4 (connections(3,1) = 4), which is smaller than numLump + numStSt, so* $\underline{\underline{\mathbf{A}}}(4,3) = (-1)^1 = -1$.

Following this procedure for all the connections will result in the

following two stream matrix:

$$\underline{\underline{\mathbf{A}}} = \begin{pmatrix} 1 & -1 & 1 & . & . & . & . \\ . & 1 & . & . & . & -1 & . \\ . & . & . & -1 & -1 & . & . \\ . & . & -1 & . & . & 1 & -1 \\ . & . & . & . & 1 & . & 1 \end{pmatrix} \tag{10.2}$$

□

Section 5.3.1 showed that the mass stream matrix is not used directly in the component mass balances, but that some additional operations have to be performed, since the component mass flows consist, in general, of a vector of components. First, the stream matrix is Kronecker multiplied with a unity matrix, which has the same dimension as the number of species that are present within the considered mass domain. The resulting matrix is pre and post multiplied with the sytem and connection selection matrix respectively. These operations, as well as the formation of the stoichiometric matrix for all systems, are straightforward and need no further explanation. In Chapter 11 examples illustrate all details of the derivations.

10.2 Equation Classification

The algebraic equations, that are needed to complete the model definition, where devided into three main groups in section 5.4. Modeller contains a database in which a set of possible algebraic equations is listed. In this database, which can, of course, be further expanded by a model designer, the main groups are further subdivided. Carefull consideration should be given, though, were new algebraic equations are placed. An algebraic equation can be placed in one of the following groups (the groups that actually contain the equations are bold faced):

* **System equations**

- Connection equations

  – Mass connections

    - Bi-directional
      * Unmodelled ⇒ **Assumptions**
      * **Rate**
    - Uni-directional
      * Unmodelled ⇒ **Assumptions**

        * **Rate**
    - Heat connections
        * Unmodelled ⇒ **Assumptions**
        * **Rate**
    - Work connections
        * Unmodelled ⇒ **Assumptions**
        * **Rate**

- Reaction equations
    * Unmodelled ⇒ **Assumptions**
    * **Rate**

* **Species equations**

* **Control equations**

Two "new" groups were added to the main groups, namely species and control equations. These new groups are there for convenience only, because normally they can be placed within the other groups.

The species equations are equations that represent physical properties of species (or groups of species). Most of the time, these equations would be assigned to the group of systems equations, but because these equations often hold independent of the system and can be easily coupled to a species database, a separate group is defined.

Control action usually affects some connection characteristics (such as the position of a valve), but sometimes, for convenience or as a simplification, a controller directly affects the secondary state variables of a source or sink system (for example, when a controller directly controls the temperature of a stream, which is, of course, physically impossible, but often seen in modelling). Also, the control equations are usually very different from the other algebraic equations, since they can also hold differential and integral terms. For these reasons the control systems and connections where defined in section 5.7 and their equations are placed in a separate group.

When a model designer is constructing a process model, this equation classification comes in handy, because for each modelling object only a limited range of equations can be choosen and this reduces the chances of making silly mistakes. When, for example, a mass connection is defined as being bi-directional, the model designer *must* choose if he wants to define the flow rate through the connection or if he wishes to leave this flow unmodelled. Either

way, he is directed to a set of possibilities: a set of possible rate equations or a set of possible contraints. This way the modeller becomes more aware of what assumptions he makes. Also, the assumptions are automatically documented and very easily traced and thus very easily changed.

10.3 Computational Order

In the fourth and fifth steps of our modelling approach the algebraic equations are added to the model definition. For each modelling object (i.e. system, connection or reaction) these algebraic equations can, in principle, be choosen randomly from the database. In doing so,the problem arises that not every numerical equation solver will be able to solve the equations since the equations are not in the so-called correct computational order and are not always in (correct) explicit form. There are solvers, such as DAE-1 solvers, that can handle implicit algebraic equations, but when the equations are simplified by performing preliminary symbolic manipulations, a much more efficient computational code can be obtained. A very important step to achieve an efficient computational code for DAEs, is to solve the equations for as many algebraic variables as possible, so that it is not necessary to introduce these variables as unknowns in the numerical DAE-solver, since they can be calculated at any call of the residual routine from the information available (Carpanzano 1996). Consider the simple pair of equations:

$$x_2 - 2x_1 = 4 \tag{10.3}$$

$$x_1 - 7 = 0 \tag{10.4}$$

In order to solve these equations directly, they must be rearranged into the form:

$$x_1 = 7 \tag{10.5}$$

$$x_2 = 2x_1 + 4 \tag{10.6}$$

The implicit equation 10.4 cannot readily be solved for x_1 by a numerical program, whilst the explicit form, namely 10.5, is easily solved for x_1 and only requires the evaluation of the right-hand-side expression. Equation 10.3 is rearranged to give 10.6 for x_2, so that when x_1 is known, x_2 can be calculated.

The rearranged form of the set of equations can be solved *directly* because it has the correct *computational causality*. This computational causility is, quite obviously, not a physical phenomenon, but a numerical artifact (Elmqvist and Otter 1993)(Thevenon and Flaus 2000)(Cellier and Elmqvist 1993). Take, for example, the ideal gas law:

$$pV = nRT \tag{10.7}$$

This is a static relation, which holds for any ideal gas. This equation does not describe a cause-and-effect relation. The law is completely impartial with respect to the question wether at constant temperature and constant molar mass a rise in pressure causes the volume of the gas to decrease or wether a decrease in volume causes the pressure to rise. For a solving program, however, it *does* matter wether the volume or the pressure is calculated from this equation.

It is rather inconvienient that a model designer must determine the correct computational causality of all the algebraic equations that belong to each modelling object, given a particular use of the model (simulation, design, etc.). It would be much easier if the equations could just be described in terms of their physical relevance and that a computer program automatically determines the desired causality of each equation and solves each of the equations for the desired variable by means of symbolic manipulation.

There exist computer programs that allow for model descriptions that are not (yet) in the correct causal form, because they are built on so-called non-causal languages. Within the last two decades a variety of these object-oriented, non-causal languages for physical systems modelling have been develloped. Examples of these languages and/or their implementations in computer programs are ASCEND (Piela and Westerberg 1991), Dymola (Elmqvist 1978), Omola (Mattsson and Andersson 1992), Modelica (Mod 2000), gProms (Barton 1992) (Barton and Pantelides 1993), YAHMST (Thevenon and Flaus 2000), ABACUSS (Freehery and Barton 1996), χ (Fábián 1999), SimGen (Dormoy and Furic 2000). All these programs manipulate and reorder the entered equations in such a way that a solution can be found for a specific problem.

The entered equations still have to adhere to some conditions:

- **For any set to be solvable, there must be exactly as many unknowns as equations**
- **It must be possible to rearrange the equations such that the system of equations can be solved for all unknowns.**

The first condition, called the Regularity Assumption in (Weiss 2000), is obviously a necessary condition. It can be checked immediately and all numerical DAE solvers take this preliminary check.

In order to solve a set of equations efficiently, the equations must be rearranged in Block Lower Triangular (BLT) form with minimal blocks, that can be solved in a nearly explicit forward sequence (Abbott 1996), (Barton 1995), (Barton 2000), (Elmqvist and Cellier 1995), (Egelhoff *et al.* 1998), (Cellier and Elmqvist 1993), (Thevenon and Flaus 2000), (Dormoy and Furic 2000).

Example 10.2: ***Block Lower Triangular Form of Algebraic Equations***

Consider the following simple example, consisting of four nonlinear equations and four unknown variables:

$$\begin{array}{ll} & \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{array}{l} f_1(x_1, x_3, x_4) \\ f_2(x_1, x_2) \\ f_3(x_4) \\ f_4(x_2, x_4) \end{array} & \underline{\underline{\mathbf{M}}} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \end{array} \tag{10.8}$$

The matrix $\underline{\underline{\mathbf{M}}}$ *represents the* incidence matrix *and is a compact representation of the structure of the set of equations. This matrix signals wether a variable occurs in an equation, or not. By permuting variables and equations, this set of equations can be brought to BLT-form:*

$$\begin{array}{ll} & \begin{array}{cccc} x_4 & x_2 & x_1 & x_3 \end{array} \\ \begin{array}{l} f_3(x_4) \\ f_4(x_2, x_4) \\ f_2(x_1, x_2) \\ f_1(x_1, x_3, x_4) \end{array} & \underline{\underline{\mathbf{M}}}' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \tag{10.9}$$

This process is called the partitioning *of the set of equations. The strictly lower triangular form of the permuted incidence matrix characterises the fact that the nonlinear equations can be solved one at a time in a given sequence. First,* x_4 *can be calculated from function* f_3 *and then, succesively,* x_2 *form* f_4*,* x_1 *from* f_2 *and* x_3 *form* f_1*. In the case where an equation or a block of equations cannot be solved symbolically for the required variables, either because it is not possible or it is too cumbersome, a numerical root solver may be employed for solving this task.*

□

There exist efficient algorithms to transform to block lower triangular form, e.g. (Tarjan 1972) and (Duff and Reid 1986). Many references state that it is, in general, not possible to transform the incidence matrix $\underline{\underline{\mathbf{M}}}$ to a strictly lower triangular form, but that there are most likely to be square blocks of dimension > 1 on the diagonal of the incidence matrix. These blocks correspond to equations that have to be solved simultaneously.

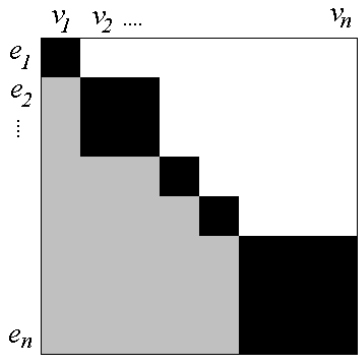

Figure 10.2: *BLT form*

In figure 10.2 (taken from (Thevenon and Flaus 2000)) the incidence matrix of a set of equations, which are transformed to BLT form, is shown. White areas indicate that the variables do not appear in the corresponding equation, grey areas that they may or may not appear, and black areas represent the variables still unknown in the block and which can be computed from the corresponding equations. So, a block of this matrix indicates which set of variables can be computed if the previous ones are known.

An observation I made during my research was that blocks are not very likely to occur when no constraining assumptions have been made (i.e. when all connections and reactions are defined via rate expressions) during the model devellopment. The reason for this becomes apparent when viewing the information flow as it is shown in figure10.3:

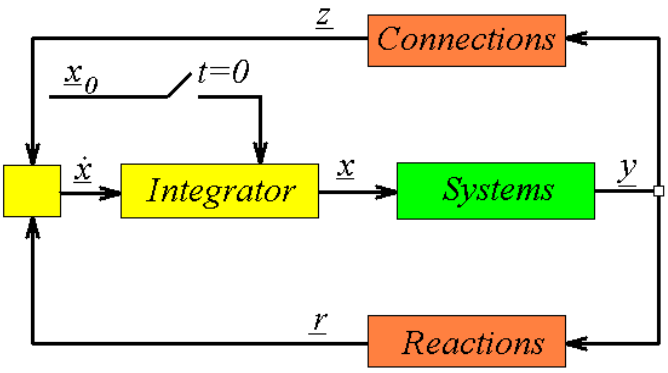

Figure 10.3: *Information flow for a simulation.*

When a DAE-1 model is formulated and proper initial conditions have been defined, then the information flow of a simulation can be depicted as in figure

10.3. Starting from the initial conditions $\underline{\mathbf{x}}_0$, the secondary state variables $\underline{\mathbf{y}}$ of all the systems can be calculated. Subsequently, the flow rates $\underline{\mathbf{z}}$ of all the defined connections and the reaction rates $\underline{\mathbf{r}}$ of all the defined reactions can be calculated. These rates are the inputs of the balance equations, so now the integrator can compute values for the primary state variables $\underline{\mathbf{x}}$ on the next time step. With these variables, the secondary state $\underline{\mathbf{y}}$ can be calculated again and the loop continues until the defined end time is reached.

For now, we are only concerned with the algebraic equations (the right hand side of figure 10.3), which means that we can consider the primary state variables $\underline{\mathbf{x}}$ of each system as known. Systems are only interacting with each other through connections and therefore the calculation of the secondary variables of each system can be done completely independent of other systems. The system equations map the primary state into a secondary state for each individual elementary system and each new equation has to define a new (secondary) variable. In some cases two or more equations may introduce two or more new variables, such that these equations have to be solved simultaneously in order to get a value for the variables. This does, however, not occur very frequently. As a matter of fact, I only know of one case in which this occurs, namely the introduction of temperature (I will elaborate on this a bit further down). A conclusion that can be drawn from this, is that system equations can, in many cases, be permuted in a strictly lower triangular form. For connection and reaction equations a similar conclusion can be drawn. For these equations the secondary variables of the systems ($\underline{\mathbf{y}}$) can be considered as known (see figure 10.3).

Blocks with a dimension > 1 on the diagonal of the incidence matrix are, in general, a result of making assumptions.

Example 10.3: ***Tank with Equilibrium Reaction***

Consider a simple isotherm batch reactor in which a reaction $A \rightleftarrows B$ takes place. The component mass balances are:

$$\dot{n}_A = -\tilde{\xi} \tag{10.10}$$

$$\dot{n}_B = \tilde{\xi} \tag{10.11}$$

We assume constant volume V (note: this is not *a constraining assumption, since the volume is, in this case, not related to the primary state variables) and define the concentration:*

$$c_A = \frac{n_A}{V} \tag{10.12}$$

$$c_B = \frac{n_B}{V} \tag{10.13}$$

If the reaction rate $\tilde{\xi}$ is defined as a function of the concentrations, there will be no problem in solving this model. When, however, a steady-state assumption is made for the reaction[1] *(e.g. $c_A = kc_B$), the index increases and, consequently, an index reduction has to be performed:*

$$\dot{n}^* = \dot{n}_A + \dot{n}_B = 0 \tag{10.14}$$

The algebraic relations and their incidence matrix for the reduced problem are:

$$\begin{array}{l} n^* = n_A + n_B \\ c_A = \frac{n_A}{V} \\ c_B = \frac{n_B}{V} \\ c_A = kc_B \end{array} \qquad \underline{\underline{\mathbf{M}}} = \begin{array}{c} \begin{array}{cccc} n_A & n_B & c_A & c_B \end{array} \\ \left[\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{array}\right] \end{array} \tag{10.15}$$

n^* has become the new primary state variable, which can be computed from the integration and is thus available for the computation of the algebraic part of the model. The incidence matrix of the algebraic part cannot be permuted into a strictly lower triagonal form. The minimal block size in the BLT form is four.

The only approach that leads to a strictly lower triangular form is to perform symbolic substitutions, such that all the unknown variables can be written as a function of n^*:

$$\begin{array}{l} n_A = \frac{kn^*}{k+1} \\ n_B = \frac{n^*}{k+1} \\ c_A = \frac{kn^*}{V(k+1)} \\ c_B = \frac{n^*}{V(k+1)} \end{array} \tag{10.16}$$

This solution is, however, only practical for small and/or linear systems of equations. The complexity of symbolic solutions grows rapidly with the number of equations (and consequently also variables). In the case of nonlinear relations, particularly when the popular polynomial models are involved, multiple solution branches (only one of which is correct) are obtained, which makes the task for the numerical solver near impossible.

Another method that can be applied is called tearing *and is a simple technique, introduced by (Kron 1962), to reduce large systems*

[1] This is, of course, a rather silly assumption to make in this small model, since this eliminates the dynamics from the model. The example is only used for demonstrating the effects assumptions have on the algebraic equations.

of algebraic equations to smaller systems of equations. Tearing means breaking algebraic loops in the dependency structure of equations and variables. A subset of the variables is chosen ase tearing variables *and a subset of the equations is chosen as* residual equations. *The choices are made in such a way that the remainder of the variables can be calculated in sequence utilising the remainder equations, under the assumptions that the tearing variables are known (Elmqvist and Cellier 1995).*

Another possibility is to use a DAE-1 solver to solve the imposed problem. In that case we do not introduce the new state variables n^*, but let the solver deal with the algebraic constraint:

$$\begin{array}{l} \dot{n}_A + \dot{n}_B = 0 \\ 0 = kc_B - c_A \end{array} \quad \Leftrightarrow \quad \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{n}_A \\ \dot{n}_B \end{pmatrix} = \begin{pmatrix} 0 \\ kc_B - c_A \end{pmatrix} \tag{10.17}$$

Each time step values for n_A and n_B will result, which means that the system equations do not have to be transformed and the secondary state variables (c_A and c_B in this case) can be easily calculated. The algebraic constraint just "replaces" one of the differential equations. The Modelica *uses this solution whenever the Simple Index Reduction method is applied. In many cases it may be possible that a "faster" code can be generated, but it was not an objective of this work to build the most efficient code possible. The main goal is to be able to generate consistent process models, which can (optionally) be postprocessed to obtain computational more efficient code.*

□

There is one case in which the formation of blocks on the diagonal of the incidence matrix seems unavoidable (even when no constraining assumptions have been made), namely the introduction of temperature. The problems that arise are due to the link between the energy balance and the component mass balances (see equation 10.19). To keep things simple we will now consider the enthalpy balance for one system with constant pressure and negligible mixing enthalpies:

$$\dot{\mathbf{H}} = \underline{\underline{\mathbf{A}}}_m \hat{\underline{\mathbf{H}}}_m + \underline{\underline{\mathbf{A}}}_q \hat{\underline{\mathbf{q}}} + \underline{\underline{\mathbf{A}}}_k \hat{\underline{\mathbf{w}}}_k \tag{10.18}$$

$$\mathbf{H} = \underline{\mathbf{h}}^T \underline{\mathbf{n}} \tag{10.19}$$

$$\underline{\mathbf{h}} = \underline{\mathbf{h}}^0 + \underline{\mathbf{c}}_p (T - T_{ref}) \tag{10.20}$$

where

$\underline{\mathbf{h}}$	::	Vector of partial molar enthalpies
$\underline{\mathbf{h}}^0$	::	Vector of partial molar enthalpies at reference temperature
$\underline{\mathbf{c}}_p$	::	Vector of heat capacities
T_{ref}	::	Reference temperature

From equations 10.19 and 10.20 the variables $\underline{\mathbf{h}}$ and T need to be solved (all other variables are assumed to be known at this point).

$$\underline{\underline{\mathbf{M}}}_1 = \begin{matrix} \\ 1 \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{matrix} h_1 & h_2 & h_3 & T \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{matrix} \qquad \underline{\underline{\mathbf{M}}}_2 = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} \underline{\mathbf{h}} & T \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \end{matrix}$$

The incidence matrix $\underline{\underline{\mathbf{M}}}_1$ shows the occurrence of $\underline{\mathbf{h}}$ and T in the equations 10.19 and 10.20 when written out for three components. Clearly this block cannot be decomposed into smaller blocks, so the equations will either have to be solved simultaneously or have to be transformed by doing substitutions and rearrangements.

Usually this problem is solved by substituting equation 10.20 in 10.19 (and thereby removing all partial molar enthalpies $\underline{\mathbf{h}}$) and then solving the resulting equation for the temperature T. This solution can, however, get very cumbersome. Especially when the heat capacities are also given as a function of the temperature this can become far from trivial.

Incidence matrix $\underline{\underline{\mathbf{M}}}_2$ shows the occurrence of the variables $\underline{\mathbf{h}}$ and T in the vector case. Although this incidence matrix is lower triangular, it does not mean that the problem can be solved directly, because $\underline{\mathbf{h}}$ is a vector and can, of course, not be solved by the scalar equation 10.19. Similarly, the scalar T can not be solved be the vector equation 10.20. A way to work around this problem is to use the capabilities of the DAE-solver and solve equations 10.18 and 10.19 numerically for the enthalpy H and the temperature T respectively:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{\mathbf{H}} \\ \dot{\mathbf{T}} \end{pmatrix} = \begin{pmatrix} \underline{\underline{\mathbf{A}}}_m \underline{\hat{\mathbf{H}}}_m + \underline{\underline{\mathbf{A}}}_q \underline{\hat{\mathbf{q}}} + \underline{\underline{\mathbf{A}}}_k \underline{\hat{\mathbf{w}}}_k \\ \underline{\mathbf{h}}^T \underline{\mathbf{n}} - \mathbf{H} \end{pmatrix} \tag{10.21}$$

Subsequently, the remaining enthalpies $\underline{\mathbf{h}}$ can easily be computed explicitly from 10.20. An obvious advantage of this method is that no transformations or substitutions have to be made.

The "problems" that arise by making assumptions locally, can be dealt with locally. This is different from what is being done by current modelling languages (such as Modelica, ABBACUS II, ASCEND): they all gather all information from a modelling-session, then throw it all on one big stack and then the symbolic analysis is done.

10.4 Output of Modeller

After completion of all the modelling steps, the Modeller is ready to generate consistent model equations. Since the output consists of mathematical equations, an interface to any problem solving package could, in principle, be written. Due to time limitations, though, I have limited the interfaced solving packages to one. I have choosen for a link to Matlab, since Matlab is a widely used matrix oriented mathematical tool, in which a lot of the initial calculations (such as null space and stream matrix calculations) can easily be performed. The generated output (which, at the moment, is oriented on simulation) consists of 2 files: A documentation file and a code file, which can be directly plugged into the problem solving package. An example of these two files is listed in Appendix B.

10.4.1 Documentation File

The generated documentation file contains information on all assumptions made by the model designer. It starts by listing the basic time-scale assumptions the model designer has made by giving the built tree structure and all the (correctly) defined connections between the elementary systems of the physical topology. Open and/or loose connections are not listed in the documentation and are also not used in the code file, since these connections are not properly defined. Next, all species and reactions that are used in the model (which per definition are subsets of the defined species and reactions) are listed and all used equations and the variables used in these equations are given. The number of different equations used in a model is very often not a function of the complexity (i.e. the number of systems and connections) of the model, but tends to stay rather low, indifferent of the size of the model.

After this general information on the model structure, more detailed information is printed for each system, reaction, connection and species, such that a person, who is interested in the model, can read which assumptions have been made on the lowest level.

10.4.2 Code File

The generated code file is coded such that it can be directly plugged into the problem solving package the model designer wants to use. For now, it is only possible to generate a Matlab m-file, for use with the Matlab software. The code file is written as a 'self-executing'-file, which means that only the name of the code file (standard saved as "modcode.m") has to be entered in the Matlab command window and the defined simulation problem will be solved. The results of the simulation can be viewed by a specially designed output-viewer, which is convieniently structured and can plot all the calculated data.

The generated m-file is subdivided into a number of subfunctions and is organised as follows:

- *Initialisation information.* In this part of the output all stream matrices, null spaces and other initialisation information parts are calculated and all global parameters are initialised.
- *Call solver and build output.* The solver is called with the initialisation information. After solving the defined constants of all modelling objects as well as other information (such as names or present species) is mapped to the output variables. During solving the dynamic variables are mapped to the output variable via the function *OutputBuild*.
- *Main code.* This part is called at least every time step during calculation and is subdivided as follows:
  - For each system the secondary state variables are calculated form the primary state variables.
  - For each connection all connection variables are calculated from all previously calculated variables.
  - For each reaction all reaction variables are calculated from all previously calculated variables.
  - All differential equations are set up for computing the next time step via the DAE solver. Also all equations (such as constraint equations or steady state balance equations) from which variables have to be calculated numerically via the DAE solver, are set up.
- *Additional code.* In this last part functions, such as a stream matrix calculation function, or a selection matrix construction code, are listed that are called on by the previous code.

The generated output file can be executed by simple entering the name of the file in the Matlab command window.

Part III

Examples and Conclusions

Chapter 11

Examples

In this chapter the modelling method is illustrated with some examples. These examples try to show that modelling does not have to be a difficult and time consuming task when adhering to the five step modelling method. Using a computer program, which builds on a structured modelling approach, such as the Modeller, can seriously speed up the construction of dynamic physical-chemical-biological process models.

The examples are structured as follows:

1. **Problem Description**. Process models are always established for a certain purpose, which the person, who establishes the model, has in mind as he works on it. Thus, for each example, a short motivation and objective of the model is given.

2. **Step 1: Physical Topology.** For each example a simple model structure is mapped out. It is one of the infinite many possible, but also one of the most obvious choices for the posed problem.

3. **Step 2: Species Topology**. The definition of the species topology requires the definition of the species sets, the reactions and the permeabilities and directionalities of all mass connections. With the species topology being defined, the component mass balances are established.

4. **Step 3: Balance Equations**. Although the balance equations are generated automatically with the information of the first two steps, it is, in some cases, interesting to have a closer look at them.

5. **Step 4a: Connection and Reaction Equations**. The transfer laws and kinetic laws that express the flow and production rates are chosen. In the case of unmodelled flows or reactions the appropriate constraining assumptions are defined.

6. **Step 4b: System Equations and Control**. For each system within the physical topology the secondary state variables, which are used to describe flows and reactions, are linked to the primary state variables (component mass and energy) through a set of definitions.

7. **Discussion**. The outcome of the modelling session is discussed and analysed with respect to the given objective of the model. Also, if necessary, the dynamic behaviour of the information processing units, such as controllers (Step 5) is added and discussed

When constructing process models these steps do not necessarily have to be done is this exact sequence. Sometimes it can even be handy to (partially) interchange some of the steps in the sequence. Especially steps 4a and 4b can easily be interchanged. It should be noted, though, that, in general, changes in earlier steps will influence the later steps and that changing later steps will, in general, have no effect on the definition of earlier steps.

11.1 Tank With Level Measurement

11.1.1 Problem Description

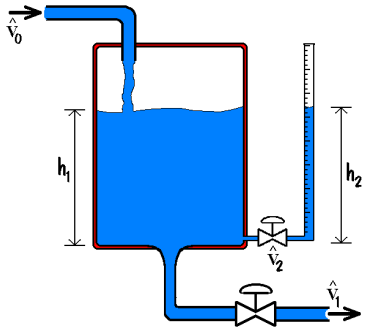

Figure 11.1: *Tank with level glass*

(This example is taken from (Preisig 1996)). It is often desired to control, or at least measure the liquid level in a continuously stirred tank reactor. For this purpose one could attach a vertical, see-trough tube to the tank which is connected to a terminal at the bottom of the tank. Often, the connecting pipe is of small diameter or has a safety valve built in. It is of interest to explore what the level measurement shows as the tank is operating in a dynamic mode, as the level may change relatively rapidly in the tank due to fluid that is fed or removed from the tank.

11.1.2 Step 1: Physical Topology

The physical topology is easily constructed when the contents of both the tank and the level glass are considered to be ideally mixed. A source and sink system are introduced, since connections can only exist between two systems (this prevents connections from "dangling").

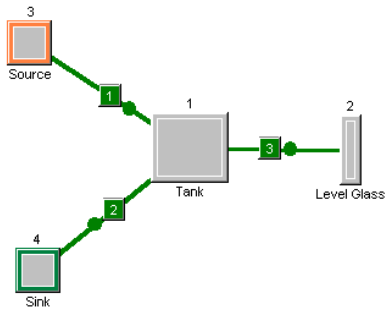

Figure 11.2: *Physical topology for the levelglass problem.*

11.1.3 Step 2: Species Topology

Since we are only interested in the level measurement, we require only total mass or one component, which is present in all systems and all connections of the physical topology.

11.1.4 Step 3: Balance Equations

The energy balance is not relevant for this problem. Thus, only the mass balances of the tank and the level glass are required:

$$\begin{pmatrix} \dot{n}_1 \\ \dot{n}_2 \end{pmatrix} = \begin{pmatrix} 1 & -1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{n}_1 \\ \hat{n}_2 \\ \hat{n}_3 \end{pmatrix} \tag{11.1}$$

Note: throughout the entire thesis, we have always used the molar mass balances. The step to mass (in kg) is easily made by multiplying the molar mass n $[mol]$ of a component with its specific molar mass M $[kg/mol]$.

11.1.5 Step 4a: Connection and Reaction Equations

We choose to express the mass flow rates in terms of volumetric flow rates. For an arbitrary mass stream $\hat{n}$, the mass flow rate is expressed in terms of the volumetric flow and the density:

$$\hat{n}_{mi} = \rho_{mi}\hat{V}_{mi} \tag{11.2}$$

where ρ_{mi} is the density (in mol/m^3) of the mass stream i, and $\hat{V}_{mi}$ is the respective volumetric flow rate (in m^3).

Assuming that the mass stream flows between the elementary systems 1 and 2, the density of the mass stream will be the density of the substance in

the system from which the stream originates:

$$\rho_{m3} = \begin{cases} \rho_1 & \text{if } \hat{V}_3 > 0 \\ \rho_2 & \text{if } \hat{V}_3 < 0 \end{cases} \tag{11.3}$$

Using the "sign"-function, this can also be written as

$$\rho_{m3} = \frac{1}{2}\left(\left(1 + sign\,\hat{V}_3\right)\rho_1 + \left(1 - sign\,\hat{V}_3\right)\rho_2\right) \tag{11.4}$$

For $x \neq 0$ this sign-function can be interpreted as

$$sign\,(x) = \frac{x}{\sqrt{x^2}} \tag{11.5}$$

In this case, however, we shall assume that the density of the liquid is constant in all parts of the process. That is, for each density we have:

$$\rho_1 = \rho_2 = \rho \tag{11.6}$$

and consequently

$$\hat{n}_{mi} = \rho \hat{V}_{mi} \tag{11.7}$$

The volumetric flow rate from the tube to the level glass is driven by the difference in static pressures p at the bottom of the two vessels. The following relation is a good candidate for expressing this rate:

$$\hat{V}_3 = c\sqrt{\frac{|p_1 - p_2|}{\rho}}sign(p_1 - p_2) \tag{11.8}$$

where c represents the valve constant including the resistance effects of the pipe.

11.1.6 Step 4b: System Equations

It this step we need to define all the needed secondary state variables. The definitions account for both systems ($\Sigma = 1, 2$) in this case.

As we are interested in a model that accounts for the levels, it is useful to use the expression:

$$p_\Sigma = \rho g h_\Sigma \tag{11.9}$$

which links the hydrostatic pressure p_Σ (introduced in equation 11.8 describing the flow between the two systems) as a function of the liquid height h_Σ of the system Σ. g is the gravity constant. In the case of cylindrical tanks we get:

$$h_\Sigma = \frac{V_\Sigma}{A_\Sigma} \tag{11.10}$$

in which A_Σ is the cross-section area of the tank and the dynamically changing volume V_Σ is defined as

$$V_\Sigma = \frac{n_\Sigma}{\rho} \tag{11.11}$$

All secondary state variables (p_Σ, h_Σ and V_Σ) are now linked to the primary state variable (n_Σ).

11.1.7 Discussion

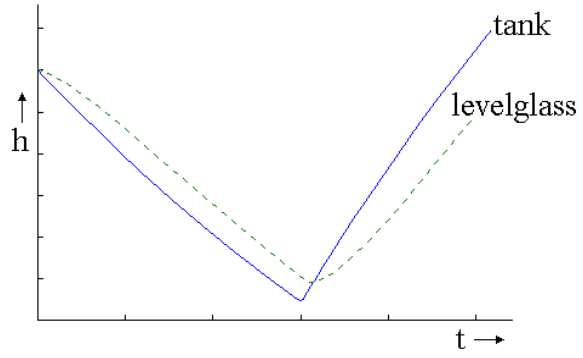

Figure 11.3: *Height of liquid in the tank and in the levelglass as a function of time*

Figure 11.3 shows a possible response of the liquid height in the tank and level glass if the inflow is stopped and at a certain time started again. As one can see, the level glass lags behind the "real" height of liquid in the tank, which can be ascribed to the small diameter of the interconnecting pipe (reflected in the value of the valve constant c). The phase lag of the level glass can be reduced by increasing the value of the valve constant c (for example, by opening the valve and thereby creating a larger diameter for the interconnection), such that the flow through the interconnecting pipe increases.

A controller that controls the liquid height in the tank via the inflow with measurements of the height in the level glass will, of course, be more difficult to implement and tune than a controller that gets its input directly from the tank itself. The measurements that are needed for control should always be taken as close as possible to the system(s) that actually needs to be controlled.

11.2 Extraction Process

11.2.1 Problem Description

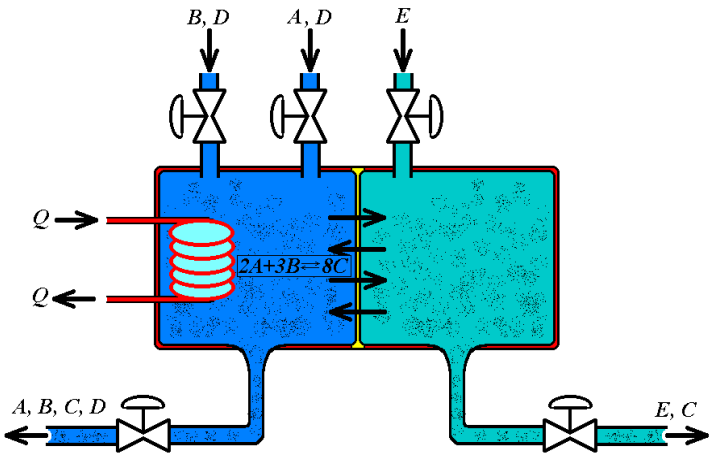

Figure 11.4: *Co-current extraction process.*

The process consists of two tanks (a reaction and a product tank) that are separated via a semi-permeable membrane. The reaction tank is fed with two feed streams, one containing the reactant A dissolved in inert solvent D, the other feeds reactant B dissolved in D. In the tank, a reaction that forms the wanted product C takes place: $2A + 3B \rightleftarrows 8C$. The reactions is exothermic and the generated heat is removed by a cooling device, which is filled with a coolant Q. The membrane separating the two tanks is only permeable for the product C. The product C enters the product tank, which is fed with solvent E, via diffusion through the membrane. The diffusion rate is driven by the difference in chemical potential (approximated with concentration in this case) of the two phases. Both tanks have an outlet flow.

A model of this process could give better insight in how the concentration of the product C in the solvent E can be increased.

11.2.2 Step 1: Physical Topology

A simple model structure is mapped out in figure 11.5. Clearly, there are two mass domains in this physical topology. One is the reactor/extractor and a second is the cooling medium. The heat connection between the cooler and the reaction phase connects the two domains to form an interconnected network. If this connection would not be there, we would have two sperate models that do not influence each other.

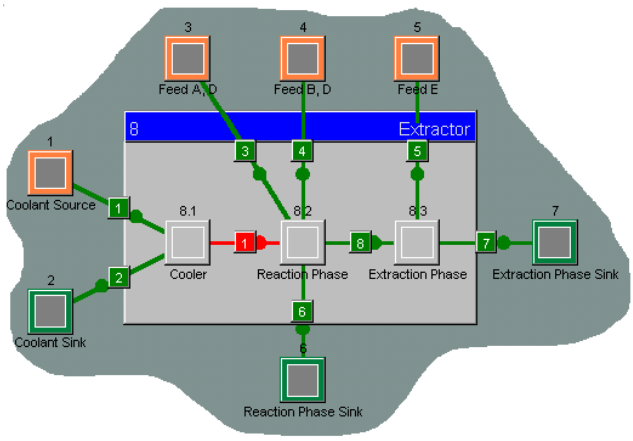

Figure 11.5: *Physical topology of the extracting reactor.*

11.2.3 Step 2: Species Topology

The definition of the species topology requires the definition of the species set, the reactions in all systems and the permeabilities and directionality for all mass streams. Often one defines the complement of the permeabilities because the default is a mass transfer of all species, thus with a permeability that includes all species. In this case, all mass streams are uni-directional and are transferring all species, with exception of the mass transfer between the two phases. This interface is assumed bi-directional and permeable only for species C.

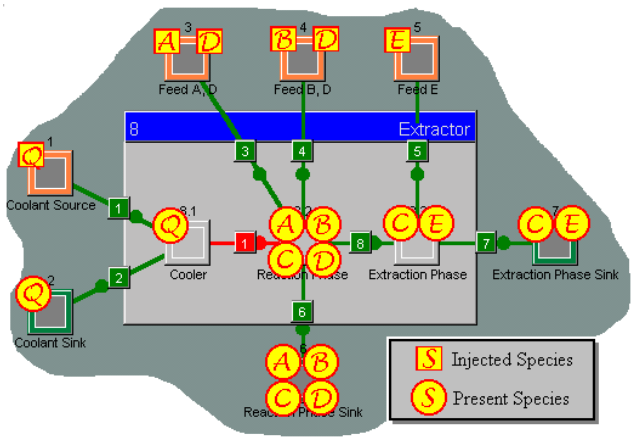

Figure 11.6: *Species topology of extraction process.*

The plant species set $\mathcal{S}$ and reaction set $\mathcal{R}$ are defined as follows:

$$\begin{aligned} \mathcal{S} &= \{A, B, C, D, E, Q\} \\ \mathcal{R} &= \{2A + 3B \rightleftarrows 8C\} \end{aligned}$$

In this case, the species topology is initiated by defining which species are present in the source systems, defining where the reaction takes place and by defining the directionalities and permeabilities of the mass connections. The resulting species topology is visualised in figure 11.6.

11.2.4 Step 3: Balance Equations

Both mass and energy balances are required for this problem. The component mass balances of the systems 8.1, 8.2 and 8.3 can be abstracted by:

$$\underline{\dot{\mathbf{n}}} = \underline{\underline{\Gamma}}_{\Sigma} \underline{\underline{\mathbf{A}}}_{k} \underline{\underline{\Gamma}}_{m}^{T} \underline{\hat{\mathbf{n}}} + \underline{\underline{\Gamma}}_{\Sigma} \underline{\underline{\mathbf{S}}}_{\Sigma}^{T} \underline{\boldsymbol{\xi}} \tag{11.12}$$

where the different matrices are easily calculated from the physical topology and species topology. If we assume constant pressure or negligible pressure differences, the enthalpy balance reads:

$$\underline{\dot{\mathbf{H}}} = \underline{\underline{\mathbf{A}}}_{m} \underline{\hat{\mathbf{H}}}_{m} + \underline{\underline{\mathbf{A}}}_{q} \underline{\hat{\mathbf{q}}} \tag{11.13}$$

The interconnection matrices $\underline{\underline{\mathbf{A}}}_{m}$ and $\underline{\underline{\mathbf{A}}}_{q}$ are defined as:

$$\underline{\underline{\mathbf{A}}}_{m} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 \end{bmatrix} \tag{11.14}$$

$$\underline{\underline{\mathbf{A}}}_{q} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \tag{11.15}$$

The structure of the matrices $\underline{\underline{\Gamma}}_{\Sigma}$, $\underline{\underline{\Gamma}}_{m}$ and $\underline{\underline{\mathbf{S}}}_{\Sigma}$ is given in figure 11.7 and shows very nicely which species are present in each system, connection or reaction.

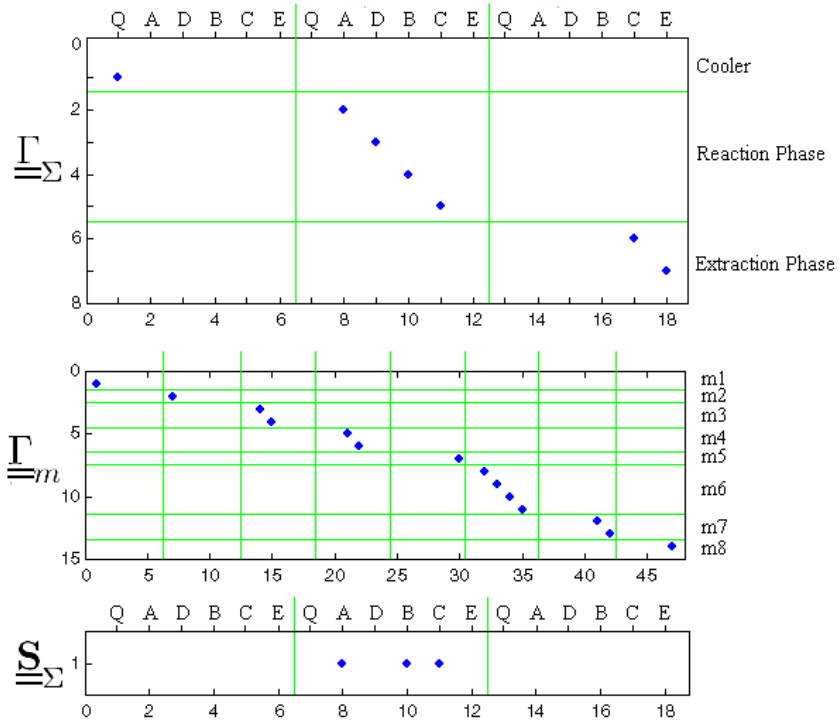

Figure 11.7: *Structure of the matrices* $\underline{\underline{\Gamma}}_{\Sigma}$, $\underline{\underline{\Gamma}}_{m}$ *and* $\underline{\underline{\mathbf{S}}}_{\Sigma}$.

11.2.5 Step 4a: Connection and Reaction Equations

For all (unidirectional) mass flows except mass flow m8 the following relations are given:

$$\underline{\hat{\mathbf{n}}}_{mi} = \underline{\mathbf{c}}_{or}\hat{V}_{mi} \tag{11.16}$$

$$\hat{H}_{mi} = \underline{\mathbf{h}}_{or}^{T}\underline{\hat{\mathbf{n}}}_{mi} \tag{11.17}$$

in which

$\underline{\mathbf{c}}_{or}$ :: Concentration in the origin system of mass connection i

$\underline{\mathbf{h}}_{or}$ :: Vector of molar enthalpies in the origin system

Mass connection 8 is bi-directional, so the involved equations are a bit more difficult to read:

$$\underline{\hat{\mathbf{n}}}_{mi} = k(\underline{\mathbf{c}}_{or} - \underline{\mathbf{c}}_{tar}) \tag{11.18}$$

$$\hat{H}_{mi} = \frac{1}{2}((\underline{\underline{\mathbf{I}}} + diag\,(sign(\underline{\hat{\mathbf{n}}})))h_{or} + (\underline{\underline{\mathbf{I}}} - diag\,(sign(\underline{\hat{\mathbf{n}}})))h_{tar})^{T}\underline{\hat{\mathbf{n}}} \tag{11.19}$$

where

$\underline{\underline{\mathbf{I}}}$	::	Unity matrix
$diag(\underline{\mathbf{x}})$	::	Operator which makes a diagonal matrix with vector $\underline{\mathbf{x}}$ on its diagonal
$\underline{\mathbf{c}}_{tar}$		Concentration in the target system of mass connection i
$\underline{\mathbf{h}}_{tar}$		Vector of molar enthalpies in the target system

The defined semi-permeability of the connection (i.e. the fact that not all species defined in the origin and target of the connection flow through the connection) is dealt with by the computer implementation.

The heat flux through the heat connection is simply modelled by:

$$q = UA(T_{or} - T_{or}) \tag{11.20}$$

The reaction in system 8.2 is modelled as a kinetic rate expression:

$$r = k_f c_A c_B - k_b c_C \tag{11.21}$$

11.2.6 Step 4b: System Equations

For each system the secondary state variables c, T and h need to be defined. In the source and sink systems they can be given as input parameters, but in the dynamic systems they need to be defined as a function of the primary state variables ($\underline{\mathbf{n}}$ and H):

$$\underline{\mathbf{c}} = \frac{\underline{\mathbf{n}}}{V} \tag{11.22}$$

$$\underline{\mathbf{h}} = \underline{\mathbf{h}}^0 + \underline{\mathbf{c}}_p(T - T_{ref}) \tag{11.23}$$

$$H = \underline{\mathbf{n}}^T \underline{\mathbf{h}} = \underline{\mathbf{n}}^T(\underline{\mathbf{h}}^0 + \underline{\mathbf{c}}_p(T - T_{ref})) \tag{11.24}$$

The temperature T can be calculated from equation 11.24.

11.2.7 Discussion

A possible response of the concentrations in the reaction and extraction phase to a step change in the inlet concentrations is plotted in figure 11.8[1]. Figure 11.9 shows the response of the temperatures in the cooler and the reaction and extraction phase. It is assumed that there are no heat losses to the environment and the reaction and extraction phase are isolated from each other, such that the extraction phase only heats up by diffusion of product C through the membrane.

[1]The values of the parameters are purely academic and choosen such that the dynamics of the involved processes (i.e. reaction and diffusion) are both visible.

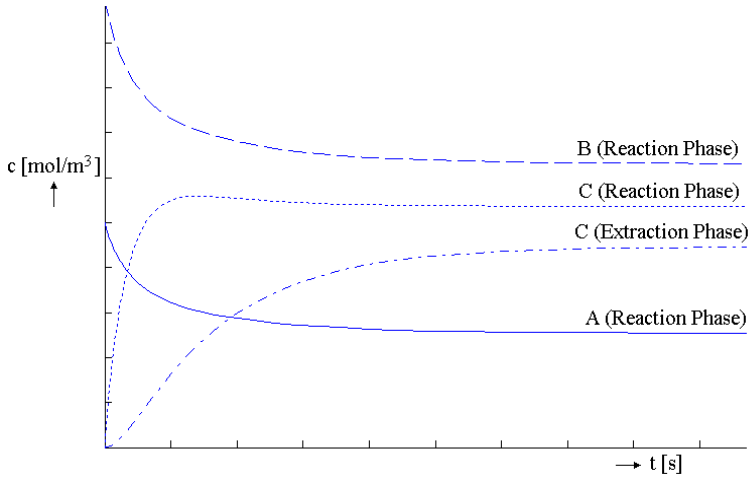

Figure 11.8: *Concentration of reactants and products in the reaction and extraction phase.*

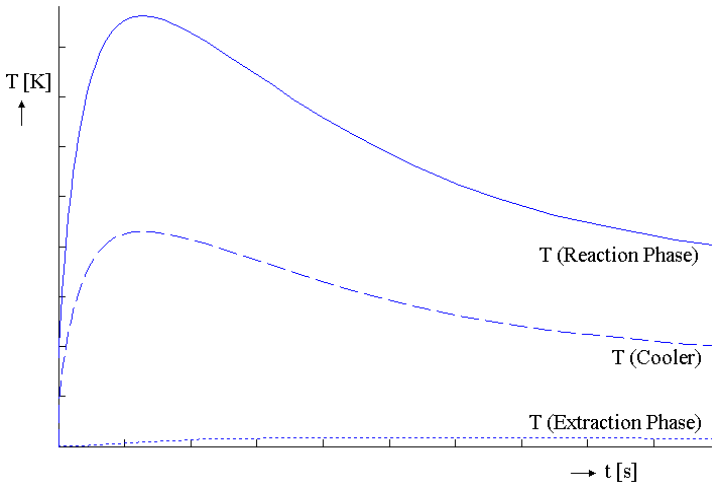

Figure 11.9: *Response of the temperatures off all lumps.*

This model was build within an hour (with the most time spent on finding 'nice' parameters) and yet shows clearly all assumptions that have been made (i.e. time scale assumptions (physical topology), species distribution (species topology), reaction kinetics, heat flows, etc.). An example of the documentation file and code file that can be generated for this model by the Modeller is given in Appendix B. The code file is chosen to give all defined parameters and variables as output. A small Matlab program was written to very easily

access and plot the output information.

The assumptions made for this model can be very easily changed. For example, a very fast (equilibrium) reaction, very fast diffusion (unmodelled flow rate) or very fast heat conduction could be considered. Also, the number of lumps could be changed (for example, by applying the repetitive structure functionality) in order to more accurately describe the mixing processes taking place in both phases. No matter what changes are made, it will be clear (also to persons who did not model this model themselves) which assumptions have been made.

A similar model structure could have been used for a process consisting of one tank with immiscable phases. If the reaction is considered to take place at the phase boundary, things become a little bit different. An example of such a process is given in the next paragraph.

11.3 Dynamic Flash

11.3.1 Problem Description

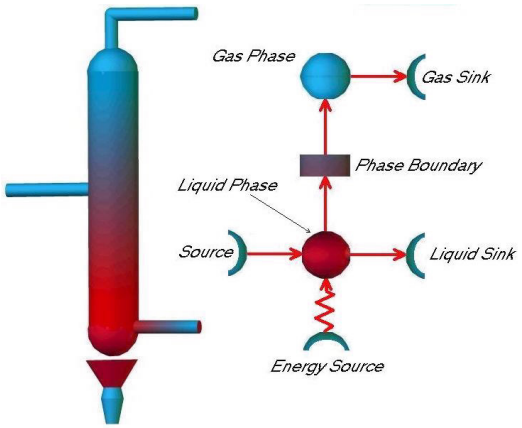

Figure 11.10: *Flash unit*

In this example, a two-phase flash unit is considered. A liquid feed stream enters the flash unit. The liquid is heated and is assumed to be in thermodynamic equilibrium with its gas phase. The goal of the flash unit is to (at least partially) separate the components that entered the unit. The dynamic modelling of processes involving phase equilibrium has had considerable attention over the years and a lot of approaches to solving the problem have been proposed. Most proposals are (mostly not-trivial) work arounds in order to get to a solvable model, because no systematic modelling methodology is

around and people are not familiar with steady-state assumptions, fast flows and equilibrium reactions.

When our structural modelling method is adhered, the modelling becomes easy and, additionally, it becomes very clear (also to people that did not develop the model themselves) where and what assumptions have been made in order to get to the end result.

11.3.2 Step 1: Physical Topology

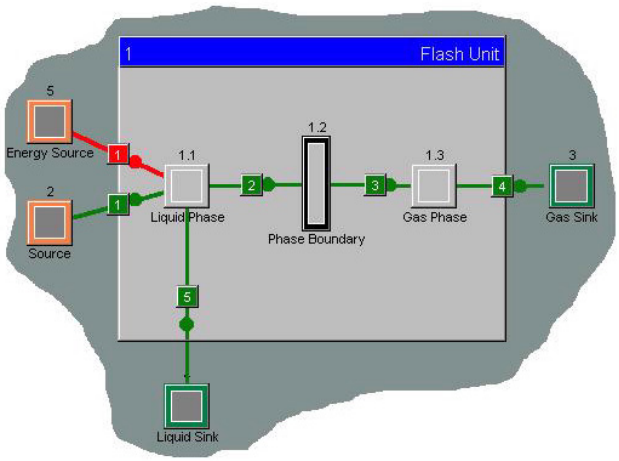

Figure 11.11: *Physical topology of the flash unit*

The flash unit is divided into two phases (a liquid and a gas phase) that are separated from each other via a phase boundary. At the phase boundary the two phases co-exist and this is also the place where the equilibrium reactions (liquid $\rightleftarrows$ gas) take place. The liquid phase is heated via a heat transfer from an energy source.

11.3.3 Step 2: Species Topology

In this example we will look at only 2 components that need to be separated. With our method a phase transition is a reaction and the liquid and gas components are different species (because these species have (very) different properties). The plant species and reaction sets therefore become:

$$\begin{aligned} \mathcal{S} &= \{A, a, B, b\} \\ \mathcal{R} &= \{A \rightleftarrows a, B \rightleftarrows b\} \end{aligned}$$

Figure 11.12 shows the species topology, which is laid on top of the physical

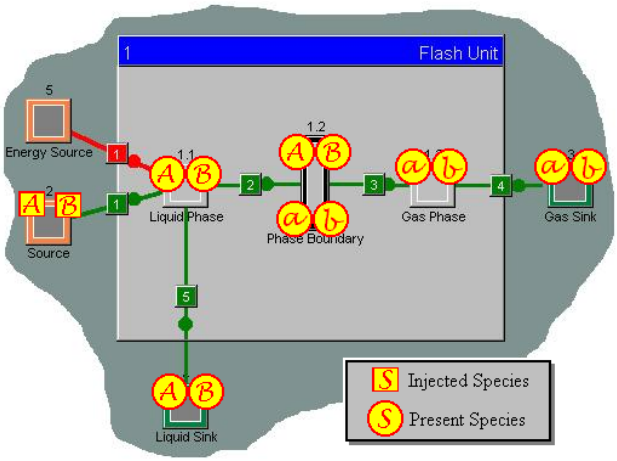

Figure 11.12: *Species topology for the flash problem.*

topology. The connection from the liquid phase to the boundary is only permeable for the liquid components (A and B) the connection from the boundary to the gas phase only for the gaseous components (a and b). The diffusion of components to and from the phase boundary are considered to very fast. The mass connections 2 and 3 are therefore fast (or unmodelled) connections.

The two reactions are taking place at the phase boundary, at which the liquid and gas phase co-exist.

11.3.4 Step 3: Balance Equations

Since there are unmodelled connections and reactions defined, there is an index problem that needs to be resolved. The rates of the unmodelled objects are of no interest and therefore the index problem can be resolved by applying the Simple Index Reduction method. In the following I will completely write out the component mass balance equations in order to show the effect of the index reduction and the steady-state assumption. The computer program **Modelica** uses a matrix notation to resolve the problems. The following balance equations can be substracted from the information of the physical and species topology:

$$\dot{n}_{A,1} = \hat{n}_{A,m1} - \hat{n}_{A,m2} - \hat{n}_{A,m5} \tag{11.25}$$

$$\dot{n}_{B,1} = \hat{n}_{B,m1} - \hat{n}_{B,m2} - \hat{n}_{B,m5} \tag{11.26}$$

$$\dot{n}_{A,2} = \hat{n}_{A,m2} - r_A \tag{11.27}$$

$$\dot{n}_{B,2} = \hat{n}_{B,m2} - r_B \tag{11.28}$$

$$\dot{n}_{a,2} = -\hat{n}_{a,m3} + r_A \tag{11.29}$$

$$\dot{n}_{b,2} = -\hat{n}_{b,m3} + r_B \tag{11.30}$$

$$\dot{n}_{a,3} = \hat{n}_{a,m3} - \hat{n}_{a,m4} \tag{11.31}$$

$$\dot{n}_{b,3} = \hat{n}_{b,m3} - \hat{n}_{b,m4} \tag{11.32}$$

Applying the Simple Index Reduction Algorithm to remove the unmodelled quantities $\hat{n}_{A,m2}$, $\hat{n}_{B,m2}$, $\hat{n}_{a,m3}$, $\hat{n}_{b,m3}$, r_A and r_B gives:

$$\dot{n}_{A,1} + \dot{n}_{A,2} + \dot{n}_{a,2} + \dot{n}_{a,3} = \hat{n}_{A,m1} - \hat{n}_{a,m4} - \hat{n}_{A,m5} \tag{11.33}$$

$$\dot{n}_{B,1} + \dot{n}_{B,2} + \dot{n}_{b,2} + \dot{n}_{b,3} = \hat{n}_{B,m1} - \hat{n}_{b,m4} - \hat{n}_{B,m5} \tag{11.34}$$

and with the steady-state assumption of the phase boundary this becomes:

$$\dot{n}_{A,1} + \dot{n}_{a,3} = \hat{n}_{A,m1} - \hat{n}_{a,m4} - \hat{n}_{A,m5} \tag{11.35}$$

$$\dot{n}_{B,1} + \dot{n}_{b,3} = \hat{n}_{B,m1} - \hat{n}_{b,m4} - \hat{n}_{B,m5} \tag{11.36}$$

In the case we would not have made the equilibrium reaction assumption, the balances obviously would have become:

$$\dot{n}_{A,1} = \hat{n}_{A,m1} - \hat{n}_{A,m5} - r_A \tag{11.37}$$

$$\dot{n}_{B,1} = \hat{n}_{B,m1} - \hat{n}_{B,m5} - r_B \tag{11.38}$$

$$\dot{n}_{a,3} = -\hat{n}_{a,m4} + r_A \tag{11.39}$$

$$\dot{n}_{b,3} = -\hat{n}_{b,m4} + r_B \tag{11.40}$$

In any case, these equations, together with the equilibrium relations for the unmodelled objects, form an index-one DAE, which can be solved by standard integrators.

If the heat transfer is also considered to be fast (to and from the phase boundary), the enthalpy balance results in:

$$\dot{H}_1 + \dot{H}_3 = \hat{H}_{m1} - \hat{H}_{m4} - \hat{H}_{m5} + q_1 \tag{11.41}$$

in which the time derivative of the pressure is considered to be negligible.

11.3.5 Step 4a: Connection and Reaction Equations

The mass flows through the mass connections m1, m4 and m5 are considered to be uni-directional and are given by:

$$\underline{\hat{\mathbf{n}}}_{mi} = \underline{\mathbf{c}}_{or}\hat{V}_{mi} \tag{11.42}$$

For connection m1 the volumetric is given. For m5 it is modelled as a function of the height of fluid of the liquid phase (i.e. the origin system of the connection):

$$\hat{V}_{m5} = \alpha h_{or} \tag{11.43}$$

The volumetric flow rate of connection m4 is modelled as a function of the difference in pressure between the gas phase and the gas sink:

$$\hat{V}_{m4} = \beta(p_{or} - p_{tar}) \tag{11.44}$$

The flows to and from the phase boundary is considered to be 'fast' and therefore the mass flows of the involved connections remain unmodelled. The equilibrium relations of the semi-permeable mass connections m2 and m3 are given respectively by:

$$\underline{\mathbf{x}}_{or} = \underline{\mathbf{x}}_{tar} \tag{11.45}$$

and

$$\underline{\mathbf{p}}_{i,or} = \underline{\mathbf{p}}_{i,tar} \tag{11.46}$$

where $\underline{\mathbf{x}}_{or}$ and $\underline{\mathbf{x}}_{tar}$ represent the fractions of liquid phase components and $\underline{\mathbf{p}}_{i,or}$ and $\underline{\mathbf{p}}_{i,tar}$ represent the partial pressures of the gas phase components.

The equilibrium 'reactions' between the gas phase and liquid phase are modelled with Raoults law:

$$p_a = x_A p_a^{sat} \tag{11.47}$$

and

$$p_b = x_B p_b^{sat} \tag{11.48}$$

in which p_i^{sat} is the saturation pressure of the pure gas component i.

The temperature dependence of the saturation pressure can be modelled by the Antoine relation:

$$p_i^{sat} = \exp(A_i - \frac{B_i}{C_i + T_{sys}}) \tag{11.49}$$

where A_i, B_i and C_i are the Antoine constants of component i and T_{sys} is the temperature in the system..

The equations of the unmodelled objects clearly link the secondary state variables (and therefore, indirectly, also the primary state variables) of the involved systems to each other. This is necessary to make sure that the index problem, introduced by these unmodelled objects, can be properly dealt with, such that an index-1 process model results.

11.3.6 Step 4b: System Equations

The connections connected to the liquid phase system dictate that the secondary variables $\underline{\mathbf{x}}$, $\underline{\mathbf{c}}$, h and T should be defined. If the (molar) density of the liquid phase ρ_l and the cross section area A are assumed to be constant, the following definitions could result (presented in block-lower-triangular form):

$$n_t = \underline{\mathbf{e}}^T \underline{\mathbf{n}} \quad (= \sum_{\forall i} n_i) \tag{11.50}$$

$$V_l = \frac{n_t}{\rho_l} \tag{11.51}$$

$$\underline{\mathbf{c}} = \frac{\underline{\mathbf{n}}}{V_l} \tag{11.52}$$

$$\underline{\mathbf{x}} = \frac{\underline{\mathbf{n}}}{n_t} \tag{11.53}$$

$$h = \frac{V_l}{A} \tag{11.54}$$

$$H = \underline{\mathbf{n}}^T \underline{\mathbf{h}} = \underline{\mathbf{n}}^T (\underline{\mathbf{h}}^0 + \underline{\mathbf{c}}_p (T - T_{ref})) \tag{11.55}$$

For the gas phase system the variables $\underline{\mathbf{c}}$, $\underline{\mathbf{p}}_i$, p and T need to be defined. The total volume V_t of the flash unit is constant and therefore the volume of the gas phase is considered to be

$$V_g = V_t - V_l \tag{11.56}$$

The partial pressures of the gas components are calculated via the ideal gas law:

$$\underline{\mathbf{p}}_i = \frac{\underline{\mathbf{n}} RT}{V_g} \tag{11.57}$$

$$p = \underline{\mathbf{e}}^T \underline{\mathbf{p}}_i \tag{11.58}$$

$$\underline{\mathbf{c}} = \frac{\underline{\mathbf{n}}}{V_g} \tag{11.59}$$

$$H = \underline{\mathbf{n}}^T \underline{\mathbf{h}} = \underline{\mathbf{n}}^T (\underline{\mathbf{h}}^0 + \underline{\mathbf{c}}_p (T - T_{ref})) \tag{11.60}$$

11.3.7 Discussion

The phase equilibrium reactions are considered to be very fast, which means that the liquid phase and gas phase are always in equilibrium. A slight change in the input concentrations and/or flow rate will therefore directly affect the output concentrations and/or flow rates of the components in both phases. If the value of β in equation 11.44 is very large, this means that the pressure of the gas phase above the liquid phase will constantly have almost the same value (high gain control). A fixed value of this pressure means that the composition of the gas phase and, consequently, also of the liquid phase is fixed. A change in input conditions will therefore only reflect in the volumes of the two phases (see figure 11.13 a and c).

If the value of β is low, a change in the input concentrations will almost not affect the liquid levels of the phases, but the composition of both phases will change (see figure 11.13 b and d).

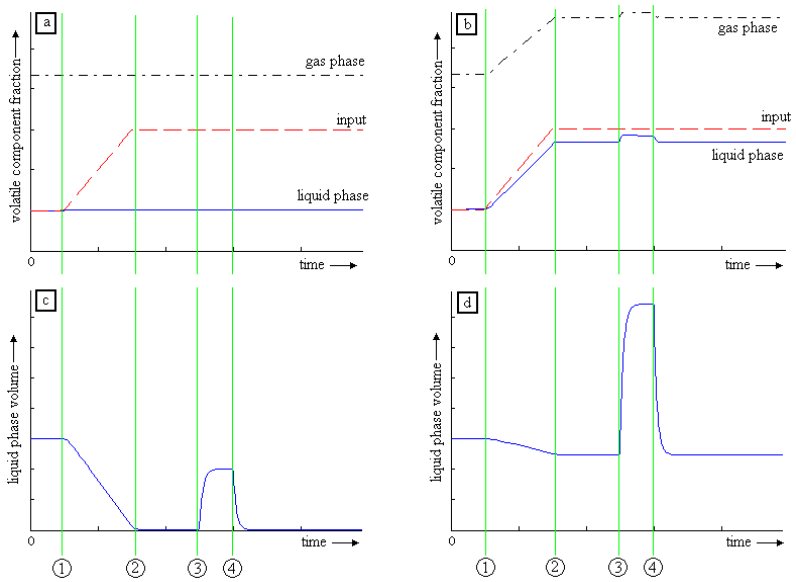

Figure 11.13: *Simulation of flash example with high (a and c) and low (b and d) value for β.*

Figure 11.13 shows a simulation run for both a large and a small value for β. The input conditions for both simulations are as follows:

At first the liquid input flow rate and input concentrations are such that the pressure of the vapor is exactly equal to the environment pressure. There will be no gas flow at this point. At point (1), the fraction of the more volatile component A is increased linearly, such that the saturation pressure above

the liquid will increase. A gas flow and/or decrease in liquid volume will be the result. At point (2), the increase is stopped. At point (3), the volumetric flow rate of the input is doubled And at point (4), the volumetric flow rate is changed back again. To keep the discussion simple, thermal effects where not taken into consideration.

11.4 Distillation Column

11.4.1 Problem Description

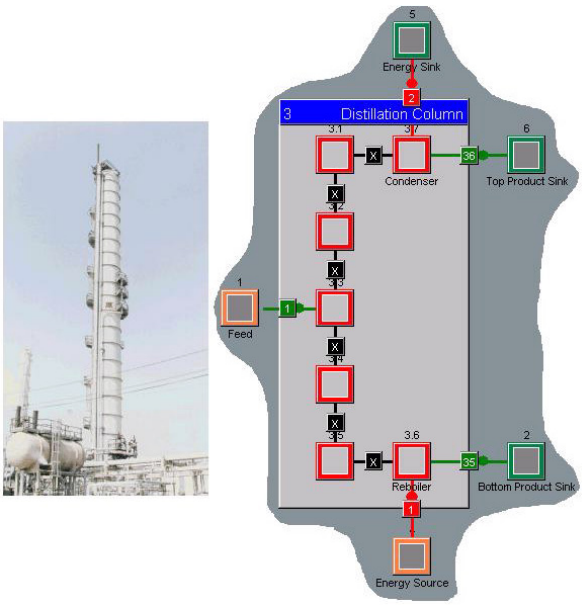

Figure 11.14: *Distillation tower and a suggested (simple) physical topology.*

A description of a distillation column has already been given in example 3.4. In this example, though, we will consider a slightly different physical topology for the trays. In order not to make things too complex we consider the column to consist of five trays, a reboiler and a condenser (as shown in figure 11.14).

11.4.2 Step 1: Physical Topology

The left picture in figure 11.15 schematically shows a small part of the distillation column (namely two trays). Each tray consists of two phases, a liquid phase and a vapor phase, which are assumed to be ideally mixed. The vapor phase is considered to be in (or at least very close to) equilibrium with the

liquid phase it is just above. The vapor phase comes in contact with the liquid phase just above it and condenses over there. So, these two phases are considered to have (almost) the same composition. If not all vapor is considered to condense in the tray above, a gas flow could be considered from the vapor phase to the vapor phase in the next tray. A connection should then be defined between these two phases, the flow through which could be driven by a pressure difference. The liquid on a tray can flow to a lower tray if the tray is flooded (which it usually is).

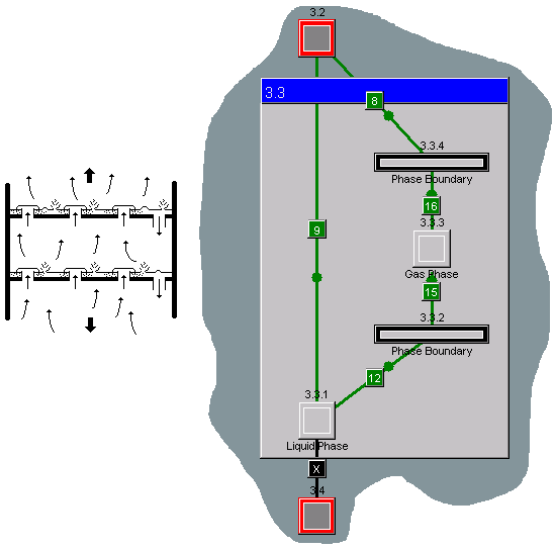

Figure 11.15: *Physical representation of a tray and a physical topology of it.*

A physical topology of a tray of the distillation column is presented in the right picture of figure 11.15. Using a tray as a repetitive cell (explained in section 8.6), a distillation column of any number of trays can be easily build. After generating the repeated structure the parts that should be slightly different, such as the feed tray, the reboiler and the condenser, can be easily modified.

11.4.3 Step 2: Species Topology

The species distribution in each tray is similar to that of the previous example (the dynamic flash). The liquid phases hold the liquid components (A and B), the gas phases hold the vapor components (a and b) and the phase boundaries hold all components as well as all phase reactions.

11.4.4 Step 3: Balance Equations

The component mass balances of all the systems in the physical topology can be abstracted by

$$\underline{\underline{\Omega}}_1 \underline{\dot{\mathbf{n}}} = \underline{\underline{\Omega}}_1 \underline{\underline{\Gamma}}_\Sigma \underline{\underline{\mathbf{A}}}_k \underline{\underline{\Gamma}}_m^T \underline{\hat{\mathbf{n}}} + \underline{\underline{\Omega}}_1 \underline{\underline{\Gamma}}_\Sigma \underline{\underline{\mathbf{S}}}_\Sigma^T \underline{\boldsymbol{\xi}} \tag{11.61}$$

or by

$$\underline{\underline{\Omega}}_2 \underline{\dot{\mathbf{n}}} = \underline{\underline{\Omega}}_2 \underline{\underline{\Gamma}}_\Sigma \underline{\underline{\mathbf{A}}}_k \underline{\underline{\Gamma}}_m^T \underline{\hat{\mathbf{n}}} \tag{11.62}$$

if all reactions are considered to be equilibrium reactions. The matrices $\underline{\underline{\Gamma}}_\Sigma$, $\underline{\underline{\mathbf{A}}}_k$, $\underline{\underline{\Gamma}}_m^T$ and $\underline{\underline{\mathbf{S}}}_\Sigma^T$ are calculated from the physical topology and species topology. The matrix $\underline{\underline{\Omega}}_1$ or $\underline{\underline{\Omega}}_2$ is a non-square matrix, which is computed from the unmodelled objects as explained in paragraph 6.3.1.

If we assume negligible pressure differences, the enthalpy balances read:

$$\underline{\underline{\Omega}}_3 \underline{\dot{\mathbf{H}}} = \underline{\underline{\Omega}}_3 \underline{\underline{\mathbf{A}}}_m \underline{\hat{\mathbf{H}}}_m + \underline{\underline{\Omega}}_3 \underline{\underline{\mathbf{A}}}_q \underline{\hat{\mathbf{q}}} \tag{11.63}$$

11.4.5 Step 4a: Connection and Reaction Equations

The connection and reaction equations are, in principle, very similar to the ones defined in the previous example, with the exception of the reaction taking place in the phase boundary above the vapor phase (system 3.3.4). Since the vapor phase is considered to have the same composition as the liquid phase just above it, the following relation can be defined:

$$\underline{\mathbf{x}}_{liq} = \underline{\mathbf{x}}_{vap} \tag{11.64}$$

in which, $\underline{\mathbf{x}}_{liq}$ is the vector of fractions of the liquid components in the liquid phase and $\underline{\mathbf{x}}_{vap}$ is the vector of fractions of the gas components in the vapor phase. If a reaction rate is modelled, the following relation could be considered:

$$r = k(\underline{\mathbf{x}}_{liq} - \underline{\mathbf{x}}_{vap}) \tag{11.65}$$

where a large k would give the same results as defining an unmodelled reaction and lower values for k would introduce some more 'dynamics' between the compositions of the two phases.

11.4.6 Step 4b: System Equations

As with the connection and reaction equations, the system equations are similar to the ones defined in the flash unit example. The only major difference between the two examples is the number of systems and connections that are defined.

11.4.7 Discussion

A distillation column may be viewed as a set of integrated, mostly cascaded, flash tanks. The integration, however, gives rise to a complex and non-intuitive behaviour, and it is difficult to understand the distillation column based on the knowledge about the behaviour of the individual pieces (the flash tanks) (Skogestad 1997).

The purpose of this example was to show how a larger scale model can be easily set up using the modelling methodology and its implementation, presented in this thesis. This example shows that with growing complexity of models, the modelling itself does not and should not become exponentially more time consuming and/or difficult.

A thorough discussion of a distillation column would be very lengthy (many textbooks have been written on this topic). For more information about distillation columns I, therefore, refer to the abundance of material written on this subject (e.g. (Rademaker 1975), (Buckley 1985), (Shinskey 1984)).

Chapter 12

Conclusions and Future Work

In this chapter the main conclusions are drawn with respect to the developed modelling methodology and its implementation and suggestions are given for further research on both modelling and implementation.

12.1 Conclusions

12.1.1 Systematic Modelling Methodology

In this work a systematic modelling methodology for the construction of dynamic process models was developed and implemented in a computer-aided modelling tool. This tool, called "the Modeller", was developed to help model designers with a systematic construction of physical-chemical-biological process models. It supports hierarchical model development based on the fundamental principles through an interactive, graphical user interface.

The development of the process models is done in five steps. In the first step, the user must establish a physical structure of the process. This structure represents the users view on the physical containment of the process. This means that the process is decomposed into simple thermodynamic systems and physical connections that represent the interactions between these systems. The simple thermodynamic systems, which represent lumped capacities able to store extensive quantities, and the connections, which represent paths for the transfer of these quantities across the common boundaries of the systems, form the physical topology of the process. To aid in the handling of large and complex processes, the physical topology is organised in a strictly hierarchical multi-way tree. A special user interface was implemented in the modelling tool to handle this tree structure and its graphical representation on the screen.

In the second stage a species topology is superimposed on the physical topology, describing the distribution of chemical and/or biological species among the defined simple systems. The species topology uses the concept

of permeability, which enables the definition of selective transfer of species, and the concept of reaction, which is used to define species reactions within a system.

With the information of the first two steps, the balance equations of all the relevant extensive quantities can be automatically generated in the third step. The rigorous implementation of the rule that connections may only be defined between two simple thermodynamic systems, combined with the introduction of a reference co-ordinate system for each connection, results in the generation of symbolic balance equations which are consistent within the context of the user-defined structure of the model.

The fourth step consists of the definition of the mechanistic details, which means that the transfer laws of each transfer are defined and that additional relationships, such as kinetics, physical property relations and geometric relations are introduced in the model. Also assumptions (such as steady-state assumptions and order of magnitude assumptions) that constrain the original model can be introduced at this stage. The index-problems that can result from introducing constraints, can be easily resolved by applying either the Simple or Full Index Reduction Method to the affected local domain.

In the fifth step information processing units, such as controllers, can be added to the model definition.

After completion of these steps, the Modeller is ready to generate consistent model equations. The resulting dynamic model will always be an index 1 DAE, which can either be used for direct solving of specific problems, or be further modified by applying mathematical manipulations, such as linearisation or model reduction.

The software tool is designed to effectively assist a model designer with the construction of consistent process models and to (very) significantly reduce the needed time effort. With the modelling tool, the process models can be built using two main operations, namely refining existing systems (the *top-down* approach) or grouping systems (the *bottom-up* approach). These two operations can be used promiscuously and all performed actions can be undone (multiple undo/redo mechanism). The software allows to store, retrieve, import or export models (or parts of models) at any stage of the model definition. This allows for a safe mechanism of model reuse.

Reflecting back on the main questions that where posed in chapter 1 of this thesis, the following could be said:

1. The methodolgy that was outlined in this thesis, as well as the many examples that were worked out, show that there is certainly much more structure in the process of modelling than is generally thought. By adhering to the five step modelling methodology, modelling automatically

becomes more of a science than an art. It makes modelling straightforward and transparent.

2. Using this structured methodology will speed up the modelling process significantly. A proof for this can be found in the fact that the examples in chapter 11 were setup and worked out in an amazingly short period of time. The software recently found an industrial application in which it could be compared in terms of efficiency and reliability with the traditional modelling approach of a highly educated chemical engineer. Whilst the first models were established over a period of a PhD, with the software it was a matter of weeks to establish a whole set of new models. Here it should be mentioned that the main effort was not on finding and modifying model structures and writing input files to solvers, but the main effort was spent on finding appropriate physical property data and corresponding descriptions. The software Modeller thus increased efficiency at least one order of magnitude.

3. Models are not only increasingly substituting for experimental work, but are basic ingredients to all aspects of process design, control and operations. Thus there is a great need for process models of different accuracy and with different focus on details. Quick and correct modelling of processes is thus obviously of central interest.

   The Modeller project had the objective to implement a formal procedure for the construction, maintenance and modification of process models. The theory is based on basic physical principles, thus is in its core white-box modelling, but enables the use of black-box components for the description of transfer, properties and kinetics. Thus overall it implements a grey-box modelling approach. Theory was developed and later implemented that allows for the introduction of order-of-magnitude assumptions such as fast flows, reactions, small capacities, phase equilibria, constant properties and constant volumes, to mention the main ones. The resulting mathematical problems are resolved internally. In some cases these assumptions have no effect on the equation structure, whilst in other cases index problems are generated. The theoretical development focused on resolving these problems for each type of assumption. The software generates always index 1 DAEs, thus structurally solvable equations systems

   Although the computer can make the modelling task a lot easier for model designers, the main order-of-magnitude assumptions and model details will always depend on the application for which the model is to be used, the amount of accuracy that has to be employed and the view and knowledge the model designer has on the process. This information

cannot be captured in any computer program.

12.1.2 Minimal Representation of Process Models

A new concise, abstract canonical form, which is able to represent a very wide range of dynamic equations of first-principle process models, was presented. It is called canonical, because it is minimal whilst, at the same time, showing all the structural elements of the process models clearly. The definition of two basic structural elements, namely the physical topology and the species topology, was shown to be sufficiently rich to yield this minimal representation of mass and energy transfer networks. The resulting algebraic structures, mostly sparse matrices and vectors, are simple to generate and easy to interpret. Moreover, the canonical form allows us to formalise important model reduction methods, which implement common order of magnitude assumptions, such as extremely fast reactions and unmodelled streams. The achieved formalisation is the basis of our model-generating software tool, the Modeller.

Another advantage of the concise canonical form was that a bridge to (linear) state space models could be easily established, such that all the analysis tools, that have been developed for linear models over the years, could be used on models that are developed (and linearised) with our modelling methodology.

A disadvantage of many modelling languages (such as Modelica) is that a lot of redundant equations (for example: "concentration in system equals concentration in mass connection") are being generated, because so-called "Ports" or "Connectors" are being used for the transfer of information from one modelling object to another. Some of the languages actually remove those redundant equations again during the compilation of the model. But since these equations are, in my opinion, obsolete, they should not have been introduced in the first place. Modeller only lists those algebraic equations that were actually defined by the model designer.

Another disadvantage that is often seen in other model generating products, is that these do not give direct access to the code that is generated. The code generator and the solver are usually directly linked. The code that is generated, after compilation of the model, is directly passed to the integrated solver and is hidden from the end user. It is therefor not clear what kind of actions have been performed on the model and the model designer does not know how the index reductions, substitutions and other (automated) mathematical transformations have affected the model. The Modeller is a "stand-alone" model generator that is not connected to any specific problem solver. It generates an output file, containing the mathematical model, which can be easily modified by the end user. Another advantage of this open structure and the structured output of the Modeller is that the user can build his

own "translator", which can translate the output file, such that it can serve as an input to his own solver programs.

12.1.3 Automatic Documentation of Assumptions

Process models are always established for a certain purpose, which the person, who establishes the model, has in mind as he works on it. For every natural system, natural in distinction to mathematical, one has a choice from a large variety of possible models. Which one is chosen depends largely on the purpose and the experience of the model designer. The model is thereby of secondary importance. The designer has an ultimate objective in mind for which the model is only one component: a necessary means to achieve the objective. It is thus not too surprising that a model designer does not necessarily document all the decisions taken during the modelling process. Consequently, the open literature is rich on models that are badly documented or not documented at all.

The main objective of this work was to introduce a step-by-step approach to modelling. An important advantage of the developed modelling methodology is that all introduced assumptions are automatically "highlighted". Whenever an assumption is made, it is very clear where the assumption is made and what the impact will be on the model. This is very important, because if the application of a model shows that the model was not good enough, then it is the assumptions that must be examined first for their viability. Thus models are developed in well-defined steps and all elements, including the assumptions are introduced in explicit form. This makes assumptions very easily retractable and changeable.

12.1.4 Local Handling of Assumptions

All assumptions that are made during the modelling process always affect only a local domain (namely the domain for which the assumption is made). For example, a fast flow assumption only affects the connection for which the assumption is made and the two interconnected systems. The problems that are a result of constraining assumptions are always handled locally, which means that there is no exponential growth of complexity when the model increases in complexity.

Most assumptions can be handled via matrix operations, leaving the "original model" intact. This is, again, a good way of keeping track of all the assumptions that have been made and a good way to see what the effects of the assumptions are on the original model.

12.1.5 Education

The modelling of chemical-physical-biological processes is one of the most important tasks of a process engineer, for these models are used on a large scale for all kinds of engineering activities, such as process control, optimisation, simulation, process design and fundamental research. The construction of these models is, in general, seen as a difficult and very time consuming task and is preferably handed over to "modelling experts". Likewise, most of the undergraduate chemical engineering curricula are model-based. However, the lack of formalisation and systematisation associated with model development leads most students and engineers to view modelling as an art, not as a science. In many cases, the emphasis of courses, which cover model development, shifts to mathematical techniques needed for the solving of differential and algebraic equations for specific cases. The fundamental concepts needed to develop the models is often pushed to the background. Also, in existing chemical engineering courses the emphasis is mostly on steady-state modelling and dynamic modelling is regarded as a difficult special case. Steady-state models are, however, derived from dynamic models, which essentially makes a steady-state model a special case of a dynamic model.

In this thesis a systematic approach to model development was presented. Also a number of difficulties (and their solutions) model designers can run into during model development where classified and some, often unnecessary but frequently seen, mathematical manipulations, such as symbolic substitutions, were criticised. By developing a course in which the concepts, presented in this thesis, are applied, students and engineers could learn that formulating (and solving) dynamic process models does not have to be difficult. Also, the emphasis of such a course could be laid on fundamental principles, such that students may develop a better understanding of the basic principles and the impact of imposed assumptions. Unfortunately, due to unforeseen circumstances, we were not able to setup such a course (yet).

The Modeller provides the tools for developing relatively complex models in a quick and easy manner. Because the balance equations are generated automatically and the algebraic equations of each modelling object can be picked from an appropriate list of alternatives, the tedious, time consuming and error-prone tasks can be avoided. From this point of view, the Modeller would be a valuable aid in the teaching of chemical process modelling, especially for those students, who are less experienced with computer programming.

12.2 Suggestions for Further Research

Several issues remain to be resolved. For both the modelling and the implementation part I propose some directions that could be followed for further

research or development.

12.2.1 Modelling

From a theoretical viewpoint, this work could be extended in several directions, some of which are:

- The modelling methodology is currently limited to handling of lumped systems. It should be extended to also cover distributed systems or at least have the possibility to mimic the behaviour of distributed systems (for example with a large number of lumped systems) in an easy and structural way. Since a distributed system is a function of position, a coordination system is used to represent this type of systems. Hence, the main problem in extending to distributed systems is in connecting a lumped system with a distributed system or a distributed system with another distributed system. Another problem that needs to be resolved is the fact that the balance equations of distributed systems almost always balance intensive (c.q. secondary state) variables[1]. In our modelling methodology for lumped systems the differentiated variables are always extensive (c.q. primary state) variables. A clear and consistent link between these two cases should be formulated.

  Note: The distributed parameter description of a system is very often formed by first looking at an infinitely small part of the considered system. This small part can be described as a lumped system with in and out flows. Subsequently, the description is expanded by integrating over the entire volume of the system, resulting in the distributed parameter description. Distributed systems can be solved by numerical solvers by dividing them into a large number of lumps (so-called "mesh cells") and then solving the obtained large system of equations[2]. Clearly, this makes the step of first making a distributed parameter description redundant. The same result could have been obtained by directly subdividing the original system into small lumps. Doing the latter could mean that the results are less accurate, more equations are needed and the no mathematical relations of a distributed profile (such as $(T(x), T(r), c(x)$, etc.) can be easily formed (which is, for that matter, only possible for very simple distributed systems). But it would give the additional advantages

[1] One can also write the extensive quantities in distributed form, but people have gotten used to see the formulation in intensive quantities.

[2] This three-point finite difference approximation is not the only method available for solving distributed systems. Also other approximations, that cannot straightforwardly be interpreted as a series of lumped systems, are available. These other methods would result in more complex structures.

that instationary situations could be handled easily, the initial conditions do not necessarily have to be smooth, no mathematical description of the geometry is needed, no state variable transformations need to be performed and the extensive quantities remain their physical meaning and do not have to be removed from the mathematical description.

- This work was limited to mass and energy. The discussed framework is, though, of a more general nature, such that the now existing modelling methodology and, consequently, also the computer program can readily be extended to accommodate any other fundamental extensive quantity, such as momentum and electrical charge.
- Incorporation of more basic thermodynamic concepts, in particular in the area of state variable transformation
- A modelling course for students and chemical engineers should be developed in which all the aspects of dynamic modelling are covered and which educates people to quickly construct consistent process models for all kinds of applications.

12.2.2 Implementation

The current version of the program is a third version after the first version of T.Y. Lee (Lee 1990) and the second of A.Z. Mehrabani (Mehrabani Zeinabad 1995). It is a complete redesign and rewrite of the earlier versions with the objective to open it up for further development. This is considerably eased by the facilities offered by the Component Pascal language, as well as the current software architecture of the program, which is strongly modularised and component-oriented. The first two projects were primarily focussed on an implementation of the physical and species topology, from which (scalar) balance equations could be generated. The theoretical background of later steps (i.e. the algebraic equations and assumptions) as well as a formalisation of all steps of the modelling methodology was developed during this research project.

There are three main directions in which the research and implementation of Modeller should continue: interfacing with other software, improving the user interface and new capabilities.

Concerning the first of these directions, a few of the immediate objectives are:

- Interfacing to major flow-sheeting, general purpose simulators and algebraic manipulators. Currently only an output file is generated that is suited for running simulations in Matlab. Interfaces to other problem solving software should be written.

- Integration of sophisticated knowledge-based systems for the selection of the applicable physical laws. These systems should provide assistance in selecting an appropriate mechanistic detail from a set of alternatives. A user should be able to choose a proper transfer law, kinetic law or any other relationship from a wide range of alternatives.

- Integration of species and (compatible) reactions databases, which contain information about physical properties and other specific quantities of species and reactions. The species and reactions databases are now completely user-defined lists. This makes it possible to generate species and reactions which may not exist (which may, of course, be a good thing for educational purposes). A better species editor and a compatible consistent reactions editor have to be written.

The following ideas would contribute to the improvement of the user interface of the program:

- Construct a better variable and equation editor. The variables used by the equation editor and consequently by the modeller can be defined by the user. Such variables have no dimensions at this stage, so a lot of "nonsense" equations could be generated. The variable editor and the equation editor have to be improved in the future. Great effort should be put in the construction of an equation editor which enforces consistency of equations.

The last of the three directions addresses some ideas for new capabilities, which should be introduced to the program:

- Handling of distributed systems. Currently the Modeller is limited to lumped systems. The extensions for handling both lumped and distributed systems should be considered. These extensions can, of course, only be implemented when the appropriate underlaying theory has been sufficiently developed.

- Multi-dimensional repetitive process structures. When a model designer builds a model which consist of many identical parts, he should be able to build one basic construction block (consisting of systems and connections), which he then could multiply in more than one direction in an easy and structurally consistent way.

- Implementation of the Full Index Reduction Method. Due to time limitations it was, unfortunately, not possible to implement the needed symbolic manipulations (such as symbolic differentiation and null-space calculation) for the application of the Full Index Reduction Method.

- Implementation of model reduction, which fits the model into a user-defined time-scale window.

Bibliography

Abbott, Kirk A. (1996). Very Large Scale Modeling. PhD thesis. Carnegie-Mellon University. Pittsburgh, Pennsylvania.

Apostel, L. (1960). Towards the formal study of models in the non-formal sciences from the concept and the role of the model in mathematics an dnatural and social sciences. D. Reidel Publishing Company.

Aris, R. (1978). Mathematical modeling techniques. Vol. 24 of *Research notes in mathematics*. Pitman.

Aris, Rutherford (1969). *Elementary Chemical Reactor Analysis*. Prentice Hall. Englewood Cliffs, N.J.

Asbjørnsen, O.A., B. Meyssami and C. Sørlie (1989). Structuring the knowledge for process modeling from first principles. IAKE'89.

Bachmann, R., L. Brüll T. Mrziglod and U. Pallaske (1989). A contribution to the numerical treatment of differential algebraic equations arising in chemical engineering. *Dechema-Monographs* **116**, 343–349.

Bachmann, R., L. Brüll T. Mrziglod and U. Pallaske (1990). On methods for reducing the index of differential algebraic equations. *Computers and Chemical Engineering* **14**(11), 1271–1273.

Barton, Paul I. (1992). The Modelling and Simulation of Combined Discrete/Continuous Processes. PhD thesis. Imperial College of Science, Technology and Medicine. London.

Barton, Paul I. (1995). Structural analysis of systems of equations. Department of Chemical Engineering, MIT, Cambridge. http://yoric.mit.edu/abacuss2/doc.html.

Barton, Paul I. (2000). The equation oriented strategy for process flowsheeting. Department of Chemical Engineering, MIT, Cambridge. http://yoric.mit.edu/abacuss2/doc.html.

Barton, P.I. and C.C. Pantelides (1993). gproms - a combined discrete/continuous modelling environment for chemical processing systems. *Simulation Series* **25**, 25–34. http://www.psenterprise.com.

Bieszczad, Jerry (2000). A Framework for the Language and Logic of Computer-Aided Phenomena-Based Process Modeling. PhD thesis. Massachusetts Institute of Technology. Massachusetts, USA.

Blachman, Nancy and Michael Mossinghoff (1998). *Maple V, Quick Reference*. Brooks/Cole. http://www.maplesoft.com.

Brenan, K.E., S.L. Campbell and L.R. Petzold (1996). *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM. Philadelphia.

Buckley, P.S., Luyben W.L. Shunta F.S. (1985). *Design of distillation column control systems*. Instrument Society of Amerca. Research Triangle Park, USA.

Bujakiewics, Pawel (1994). Maximum weighted matching for high index differential algebraic equations. PhD thesis. Technical University Delft. Delft, The Netherlands.

Carpanzano, Emanuele (1996). A graph theoretic approach to the tearing problem. Internal Report 1996#158. Department of Electronics and Computer Science, Politecnico di Milano. Italy.

Cellier, F.E. and H. Elmqvist (1993). Automated formula manipulation supperts object-oriented continuous-system modeling. *IEEE Control System Magazine* **13**(2), 28–38.

Cellier, François E. (1991). *Continuous System Modeling*. Springer Verlag. New York.

Chung, Y. and A.W. Westerberg (1990). A proposed numerical algorithm for solving nonlinear index problems. *Ind. Eng. Chem. Res.* **29**, 1234–1239.

Denn, Morton M. (1986). *Process Modeling*. Longman Scientifical & Technical.

Dormoy, Jean-Luc and S. Furic (2000). A transformational approach to code generation for numerical simulation: the simgen system. Submitted to ASE 2000.

Duff, I.S., A.M. Erismann and J.K. Reid (1986). Direct methods for sparse matrices. Oxford Science Publications.

Egelhoff, C.J., D.M. Blackketter and J.P. Glumbik (1998). Innovative equation management techniques for impementation in kinematics and dynamics courses. In: *Frontiers in Education Conference*. Tempe, Arizona, USA.

Elmqvist, H. and S.E. Mattsson (1997). Modelica - the next generation modeling language: An international design effort. In: *1st World Congress on System Simulation*. Singapore.

Elmqvist, H., M. Otter and F.E. Cellier (1995). Inline integration: A new mixed symbolic/numeric approach for solving differential– algebraic equation systems. In: *ESM'95, SCS European Simulation MultiConference*. Prague, Czech Republic.

Elmqvist, H.and F.E. Cellier and M. Otter (1993). Object-oriented modeling of hybrid systems. European Simulation Symposium. Delft, The Netherlands. pp. xxxi – xli.

Elmqvist, Hilding (1978). A Structured Model Language for Large Continuous Systems. PhD thesis. Lund Institute of Technology. Lund, Sweden.

Evans, L.B. (1990). Process modelling: What lies ahead. *Chemical Engineering Progress* pp. 42–44.

Fábián, G (1999). A Language and Simulator for Hybrid Systems. PhD thesis. Eindhoven University of Technology. Eindhoven, The Netherlands.

Fábián, G, D.A. van Beek and J.E. Rooda (2001). Index reduction and discontinuity handling using substitute equations. *Mathematical and Computer Modelling of Dynamical Systems* **7**(2), 173–187.

Freehery, W. and P.I. Barton (1996). A differentiation-based approach to dynamic simulation and optimization with high-index differential-algebraic equations. In: *SIAM Computational Differentiation* (G. Corliss M. Berz, C. Bischof and A. Griewank, Eds.). pp. 239–253. http://yoric.mit.edu/abacuss2/abacuss2.html.

Gear, C.W. (1988). Differential-algebraic equation index transformations. *SIAM J. Sci. Stat. Comput.* **9**(7), 39–47.

Gear, C.W. (1990). Differential algebraic equations, indices and integral algebraic equations. *SIAM J. Numer. Anal.* **27**(6), 1527–1534.

Gear, C.W. and L.R. Petzold (1984). Ode methods for the solution of differential/algebraic systems. *SIAM J. Numer. Anal.* **21**(4), 716–728.

Grijseels, Steven (1999). On the subject of single phase simple systems: A thermodynamic modelling concept. Internal Report NR-2159. Technische Universiteit Eindhoven. The Netherlands.

Hangos, K.M. and I.T. Cameron (2001). *Formal Representation of Assumptions in Process Modelling*. in print.

Himmelblau, D.M. and K.B. Bischoff (1968). *Process Analysis and Simulation. Deterministic Systems*. John Wiley & Sons.

Hindmarsh, A.C. (1983). Odepack, a systematized collection of ode solvers. In: *Scientific Computing* (R.S. Stepelman, Ed.). North-Holland, Amsterdam. pp. 55–64.

Jensen, Anne Krogh (1998). Generation of Problem Specific Simulation Models with an Integrated Computer Aided System. PhD thesis. Technical University of Denmark, Department of Chemical Engineering. Kastrup, Denmark.

Kailath, Thomas (1980). *Linear Systems*. Information and System Sciences. Prentice-Hall, Inc.. Englewood Cliffs.

Kron, G. (1962). Diakoptics - the piecewise solution of large-scale systems. MacDonald & Co., London, UK.

Lee, Tae Yeoung (1990). The development of an object-oriented environment for the modeling of physical, chemical and biological systems. PhD thesis. Texas A&M University. USA.

Lohmann, Bernd (1998). Anzätze zur Unterstützung des Arbeitsablaufes bei der rechnerbasierten Modellierung verfahrenstechnisher Prozesse. PhD thesis. RWTH Aachen, Germany.

MacKenzie, S. A., P. J. Gawthrop R. W. Jones and J. W. Ponton (1991). Systematic modelling of chemical processes. *IFAC symposium on Advanced Control of Chemical Processes, ADCHEM'91.*

Majer, C., W. Marquardt and E.D. Gilles (1995). Reinitialization of daes after discontinuities. *Computers and Chemical Engineering* **19**, S507–S512.

Marquardt, W. (1991). Dynamic process simulation - recent progress and future challenges. CPC IV.

Marquardt, W. (1992). An object-oriented representation of structured process models. *European Symposium on Computer Aided Process Engineering-1* pp. S329–S336.

Marquardt, W. (1994*a*). Towards a process modelling methodology. Technical report. Aachen University of Technology. Aachen, Germany.

Marquardt, W. (1995). Numerical methods for the simulation of differential-algebraic process models. In: *Methods of model based control* (R. Berber, Ed.). Vol. 293 of *NATO-ASI Ser. E, Applied Sciences*. pp. 42–79. Kluwer Academic Publ. Dordrecht, The Netherlands.

Marquardt, Wolfgang (1994*b*). An advanced framework for computer-aided process modeling. In: *ASPENWORLD'94*. Vol. 1 of *T5*. Boston MA, USA.

MATLAB user's guide (1992). MATH WORKS Inc. http://www.mathworks.com.

Mattsson, S.E. and G. Söderlind (1993). Index reduction in differential-algebraic equations using dummy derivatives. *SIAM J. Sci. Stat. Comput.* **14**(3), 677–692.

Mattsson, S.E. and M. Andersson (1992). Omola - an object oriented modelling language. In: *Recent Advances in Computer-Aided Control Systems Engineering* (M. Jamshidi and C.J. Herget, Eds.). Vol. Studies in Automation and Control. Amsterdam: Elsevier Science Publishers. pp. 291–310.

Mehrabani Zeinabad, Arjomand (1995). Computer aided modelling of physical-chemical-biological systems. PhD thesis. University of New South Wales. Sydney, Australia.

Mod (2000). *Modelica^{TM} - A Unified Object-Oriented Language for Physical Systems Modeling - Tutorial*. http://www.modelica.org.

Moe, Håvard I., S. Hauan, K.M. Lien and T. Hertzberg (1995). Dynamic model of a system with phase- and reaction equilibrium. *Comput.Chem.Eng.* **19**, S513–S518.

Moe, Håvard I. (1995). Dynamic process simulation: studies on modeling and index reduction. PhD thesis. Norwegian University of Science and Technology. Trondheim, Norway.

Ogunnaike, Babatunde A., W. Harmon Ray (1994). *Process Dynamics, Modeling, and Control*. Oxford University Press.

Pantalides, C.C. (1998). Speedup - recent advances in process simulation. *Comput. Chem. Eng.* **12**(7), 745–755. www.aspentech.com.

Pantelides, C.C. (1998). Dynamic behaviour of process systems. Technical report. Centre for Process Systems Engineering, Imperial College. London, UK.

Petzold, L.R. (1983). A description of dassl: A differential/algebraic system solver. In: *Scientific Computing* (R.S. Stepleman et al, Ed.). North-Holland, Amsterdam. pp. 65–68.

Petzold, L.R. and P.Lötstedt (1986). Numerical solution of nonlinear differential equations with algebraic constraints ii: Practical implications. *SIAM J. Sce. Statist. Comput.* **7**, 720–733.

Pfister, C. and C. Szyperski (1994). *Framework Tutorial and Reference*. 12 ed.. Oberon microsystems, Inc. Solothurnerstr. 45 CH-2043 Basel, Switzerland. Introduction to Oberon/F (object-oriented programming language).

Piela, P. (1989). ASCEND - An Object Oriented Environment for the Development of Quantitative Models. PhD thesis. Carnegie Mellon University. Pittsburg, USA. http://www-2.cs.cmu.edu/ ascend/.

Piela, P., R.McKelvey and A. Westerberg (1993). An introduction to the ascend modeling system: Its language and interactive environment. *Manag. Inf. Syst.* **3**, 91–121. http://www-2.cs.cmu.edu/ ascend/.

Piela, P.C., T.G. Epperly K.M. Westerberg and A.W. Westerberg (1991). An object-oriented computer environment for modeling and analysis: The modeling language. *Computers and Chemical Engineering* **15**, 53–72.

Preisig, H.A. (1994*a*). Components of a computer-based process engineering environment. Technical report. University of New South Wales. Kensington, USA, NSW 2033.

Preisig, H.A., Ed.) (1991*a*). *On Computer-Aided Modelling For Design*. AIChe Annual Meeting. Los Angeles, USA. 138e.

Preisig, H.A., Ed.) (1991*b*). *View On The Architecture Of A Computer-Aided Process Engineering Environment*. AIChE Annual Meeting. Los Angeles, USA. 137b.

Preisig, H.A., Guo, D.-Z. and Mehrabani, A.Z., Eds.) (1991). *Computer-Aided Modelling: A New High-Level Interface to Process Engineering Software*. CHEMECA-91. New Castle, Australia. 954-960.

Preisig, H.A., Lee, T.Y. and Little, F., Eds.) (1990). *A Prototype Computer-Aided Modelling Tool for Life-Support System Models*. 20th Intersociety Conference on Environmental Systems. SAE Technical Paper Series No 901269. Williamsburgh, USA. 10 pages.

Preisig, H.A., Lee, T.Y., Little, F. and Wright, B., Eds.) (1989). *On The Representation of Life-Support System Models*. 19th Intersociety Conference on Environmental Systems. SAE Technical Paper Series No 891479. San Diego, California, USA. 13 pages.

Preisig, Heinz A. (1991*c*). On computer-aided modelling for design. In: *Annual Meeting AIChE*. Los Angeles, USA.

Preisig, Heinz A. (1994*b*). Computer aided modeling: species topology. In: *Preprints IFAC Symp. ADCHEM'94*. Kyoto, Japan. pp. 143–148.

Preisig, Heinz A. (1996). *Dynamic Systems and Control Technology*. Eindhoven University of Technology. 5600 MB Eindhoven, The Netherlands. college material.

Rademaker, O.J., Rijnsdorp J.E. Maarleveld A. (1975). *Dynamics and control of continuous distillation columns*. Elevier. Amsterdam, The Netherlands.

Sargent, R.W.H. (1983). Advances in modelling and analysis of chemical process systems. *Computers Chem. Engng.* **7(4)**, 219–237.

Sargent, R.W.H. (1989). Process design - whats next?. In: *Proceedings of the Third International Conference on Foundations of the Computer-Aided Proces Design*. Elsevier. pp. 528–553.

Shinskey, F.G. (1984). *Distillation Control*. McGraw-Hill. New York, USA.

Skogestad, Sigurd (1997). Dynamics and control of distillation columns; a tutorial introduction. *Trans. IChemE*.

Sontag, Eduardo D. (1990). *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Vol. 6 of *Texts in Applied Mathematics*. Springer-Verlag. New York.

Stephanopoulos, George (1984). *Chemical Process Control. An Introduction to Theory and Practice*. Prentice-Hall, Inc.. Englewood Cliffs, New Jersey 07632.

Tarjan, R.E. (1972). Depth-first search and lnear graph algorithms. *SIAM Journal of Computing* **1**(2), 146–160.

Telnes, K. (1992). Computer Aided Modeling of Dynamic Processes Based on Elementary Physics. PhD thesis. The Norwegian Institute of Technology. Trondheim.

Thevenon, Luc and Jean-Marie Flaus (2000). Modular representation of complex hybrid systems: Application to the simulation of batch processes. *Accepted for publication in the Journal of Simulation Practice.*

Tisza, L. (1961). The thermodynamics of phase equilibrium. *Annals of Physics* **13**(1), 92.

Vázquez-Román, R., J.M.P. King and R. Banares-Alcántara (1996). KBMoSS: A process engineering modelling support system. *Computers Chem. Engng.* **20**, S309–S314.

Weiss, M., H.A. Preisig (2000). Structural analysis in the dynamical modelling of chemical engineering systems. *Mathematical and Computer Modelling of Dynamical Systems* **6(4)**, 325–264.

Westerweele, M.R., H.A. Preisig (2001). Minimal representation of first principle process models. ESCAPE. Elseview Science Ltd.. Scanticon Comwell Kolding. pp. S47–S52.

Westerweele, M.R., H.A. Preisig M. Weiss (1999). Concept and design of Modeller, a computer-aided modelling tool. *Computers and Chemical Engineering* pp. S751–S754.

Westerweele, M.R., M. Akhssay and H.A. Preisig (2000). Modelling of systems with equilibrium reactions. IMACS Symposium on Mathematical Modelling. Vienna University of Technology, Austria. pp. 697–700.

Wolfram, Stephen (1991). *Mathematica, a system for doing mathematics by computer.* Addison-Wesley Publishing Co.. Redwood City, California. http://www.wolfram.com.

Zeigler, Bernard P. (1984). *Multifaceted Modeling and Discrete Event Simulation.* Academic Press. London, UK.

Appendix A

Module CamTools

DEFINITION CamTools;

IMPORT Stores, Dialog;

CONST

clean = 0;

invisible = 1;

notUndoalble = 2;

VAR

stack: RECORD

undoList, **redoList**: Dialog.List;

maxShow: INTEGER

END;

PROCEDURE **BeginModification** (type: INTEGER; st: Stores.Store);

PROCEDURE **BeginScript** (st: Stores.Store; IN opName: Stores.OpName; VAR script: Stores.Operation);

PROCEDURE **ClearUndoRedoListsOf** (st: Stores.Store);

PROCEDURE **Do** (st: Stores.Store; IN opName: Stores.OpName; op: Stores.Operation);

PROCEDURE **EndModification** (type: INTEGER; st: Stores.Store);

PROCEDURE **EndScript** (st: Stores.Store; script: Stores.Operation);

PROCEDURE **GetUndoRedoListsOf** (st: Stores.Store);

PROCEDURE **MaxShowNotifier** (op, from, to: INTEGER);

PROCEDURE **Redo**;

PROCEDURE **RedoGuard** (VAR par: Dialog.Par);

PROCEDURE **RedoListNotifier** (op, from, to: INTEGER);

PROCEDURE **Undo**;

PROCEDURE **UndoGuard** (VAR par: Dialog.Par);

PROCEDURE **UndoListNotifier** (op, from, to: INTEGER);

```
END CamTools.
```

The module CamTools forms the foundation for undoable and redoable operations. This module uses the implementation of the original BlackBox undo/redo mechanism in such a way that more than one operation can be undone or redone within one call. The next few alineas contain some excerpts from the BlackBox documentation and explain how the undo/redo mechanism works.

Operations are managed per document. Every document contains two operation stacks; one is the undo stack, the other is the redo stack. Executing an operation for the first time pushes the object on the undo stack and clears the redo stack. When the user performs an undo, the operation on top of the undo stack is undone, removed from the undo stack, and pushed onto the redo stack. When the user performs a redo, the operation on top of the redo stack is redone, removed from the redo stack, and pushed onto the undo stack. A document's undo and redo stacks are cleared when the document is saved (check point).

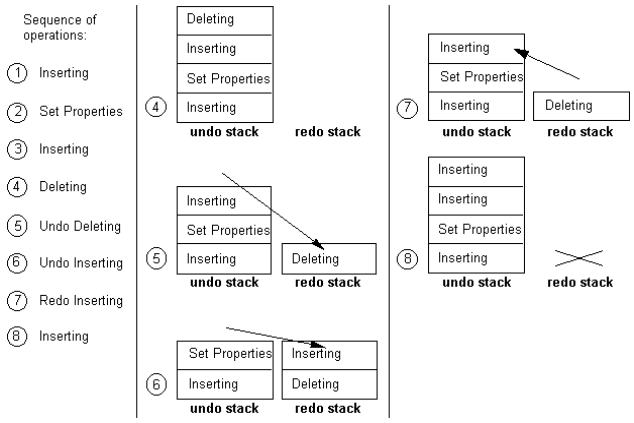

Figure A.1: *Sequence of operations and resulting modifications of undo and redo stacks.*

Some operations, such as a moving drag & drop (in contrast to the normal copying drag & drop), modify two documents simultaneously. In these rare cases, two operations are created: one for the source document (e.g., a delete operation) and one for the destination document (e.g., an insert operation).

In Figure A.1, a sequence of operations is shown, from 1) to 8). For the last five situations, the resulting undo and redo stacks are shown. For example, after operation 5), the undo stack contains the operations Inserting, Set Properties, and Inserting (from top to bottom of stack), while the redo stack contains Deleting.

The undo/redo mechanism is only concerned with the persistent state of a document. This is the state which can be saved in a file. Modifications of temporary state, such as a view's scroll position, are not recorded as operations, and thus cannot be undone.

The BlackBox Component Framework is unique in that its undo/redo mechanism is component-oriented: it allows to compose undoable operations into compound operations, which are undoable as a whole. Without this capability, nested operations would be recorded as a flat sequence of atomic operations.

Consider what this would mean for the end user. It would mean that the user could execute a menu command, which causes the execution of a hierarchy of operations. So far so good. But when the user wanted to undo this command, he or she would have to execute Edit->Undo individually for every single operation of the command, instead of only once.

For this reason, BlackBox provides support for arbitrarily nested operations: the module CamTools exports the procedures BeginScript and EndScript. In this context, "script" means a sequence or hierarchy of atomic operations which is undoable as a whole.

Abstract operations are very light-weight. They only provide one single parameterless procedure, called Do. This procedure must be implemented in a reversible way, so that if it is executed an even number of times, it has no effect. If it is executed an odd number of times, it has the same effect as when it is executed once.

A simple and correct way to implement this Do procedure of an operation is as follows: Within the procedure first save the old state, then set the new state and finally set the undo state. The following example clarifies the working of the Do procedure.

Example 1.1: *The Undo/Redo Mechanism*

> *Consider an object (**obj**) which has one parameter (**obj.x**) and an operation (**op**) which has a parameter (**op.x**) and a reference to the object (**op.obj**). The procedure Do of the operation **op** whould then look as follows:*

```
PROCEDURE (op: Operation) Do;
VAR x (*temporary state *)
BEGIN
    (* Save old state *)
    x := op.obj.x;
    (*Set new state *)
    op.obj.x := op.x
    (*Set undo state *)
    op.x := x
END Do;
```

*Say the parameter (**x**) of the object (**obj**) is set to **obj.x = old_state** and we want to change the parameter to **new_state**. First a new operation must be defined and the operations object must be set to the object which we want to change (**op.obj := obj**). Next the desired new state of the object is passed to the new operation (**op.x := new_state**). Finally the operations procedure Do is called. The following happens:*

1. *The old state is saved in a temporary parameter:* ***x := old_state***
2. *The new state is set:* ***op.obj.x := new_state***
3. *The undo state is set:* ***op.x := old_state***

*So, after the operation is performed, the operation (**op**) contains the object (**obj**) which now has its parameter set to the new state and the parameter of the operation (**op.x**) is set to the old state. When the operation is undone the Do procedure is called again:*

1. *The old state is saved in a temporary parameter:* ***x := new_state***
2. *The new state is set:* ***op.obj.x := old_state***
3. *The undo state is set:* ***op.x := new_state***

The operation is now executed two times and as you can see this has the same effect as not executing the operation at all. The procedure is thus implemented in a reversible way, so that if it is executed an even number of times, it has no effect. If it is executed an odd number of times, it has the same effect as when it is executed once.

□

Appendix B

Example of Modeller Output Files

This appendix lists the documentation file and code file of the extractor example (discussed in paragraph 11.2) as they are generated by the current implementation of the Modeller.

The documentation file documents all the assumptions made by the model designer and lists all the defined equations and variables. The code file is a Matlab m-file, which can be executed by typing the name of the file in the Matlab command window.

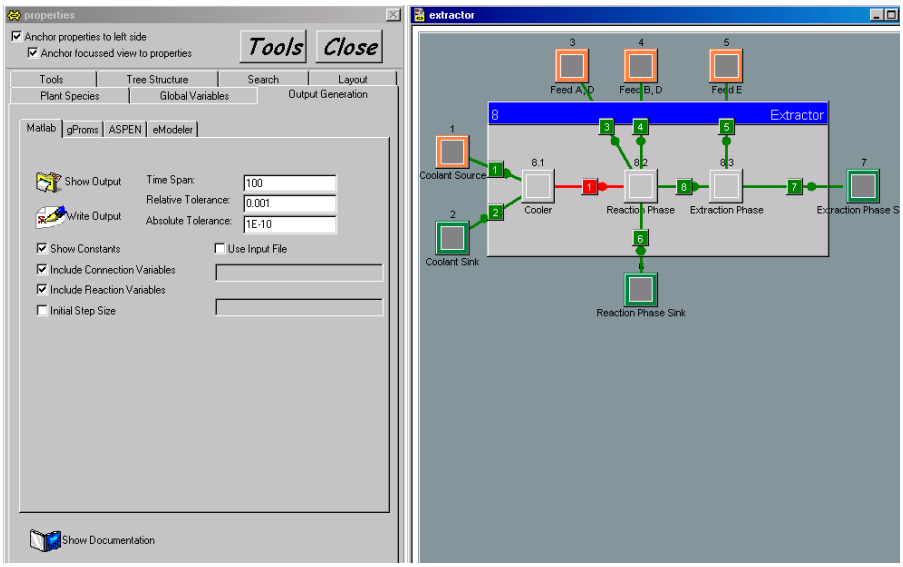

Figure B.1: *Screenshot of output generation tab.*

MODELLER DOCUMENTATION FILE:

MODEL NAME: Extraction Process

HIERARCHICAL STRUCTURE:

(Top Level) Extraction Process
- **-(1) Coolant Source**
- **-(2) Coolant Sink**
- **-(3) Feed A, D**
- **-(4) Feed B, D**
- **-(5) Feed E**
- **-(6) Reaction Phase Sink**
- **-(7) Extraction Phase Sink**
- -(8) Extractor
  - **-(8.1) Cooler**
  - **-(8.2) Reaction Phase**
  - **-(8.3) Extraction Phase**

CONNECTIONS:

INDEX	TYPE	SOURCE --||-- TARGET
m01	Mass	**Coolant Source** (1) --||-- **Cooler** (8.1)
m02	Mass	**Cooler** (8.1) --||-- **Coolant Sink** (2)
m03	Mass	**Feed A, D** (3) --||-- **Reaction Phase** (8.2)
m04	Mass	**Feed B, D** (4) --||-- **Reaction Phase** (8.2)
m05	Mass	**Feed E** (5) --||-- **Extraction Phase** (8.3)
m06	Mass	**Reaction Phase** (8.2) --||-- **Reaction Phase Sink** (6)
m07	Mass	**Extraction Phase** (8.3) --||-- **Extraction Phase Sink** (7)
m08	Mass	**Reaction Phase** (8.2) --||-- **Extraction Phase** (8.3)
h01	Heat	**Cooler** (8.1) --||-- **Reaction Phase** (8.2)

PLANT SPECIES:

1) Species Q - Q
2) Species A - A
3) Species D - D
4) Species B - B
5) Species C - C
6) Species E - E

OCCURING REACTIONS:

1) 2A + 3B ==> 8C

MAPPING OF SYSTEMS:

1) (8.1) Cooler
2) (8.2) Reaction Phase
3) (8.3) Extraction Phase
4) (1) Coolant Source
5) (2) Coolant Sink
6) (3) Feed A, D
7) (4) Feed B, D
8) (5) Feed E
9) (6) Reaction Phase Sink
10) (7) Extraction Phase Sink

USED EQUATIONS:

1) hs = spec.h0 + spec.cp*(T - glob.Tref)
2) c = n/V
3) T = (H - n'*spec.h0)/(n'*spec.cp) + glob.Tref
4) r = kf*sys.c(1)^2*sys.c(2)^3 - kb*sys.c(3)^8
5) nhat = or.c*Vhat
6) Hhat = or.hs'*nhat
7) Hhat = ((diag(1+sign(nhat))*or.hs + diag(1-sign(nhat))*tar.hs)/2)'*nhat
8) nhat = k*(or.c - tar.c)
9) q = U*A*(or.T - tar.T)

USED VARIABLES:

1) **hs** - Specific Enthalpy
2) **h0** - Specific Ref Enthalpy
3) **cp** - Specific Heat
4) T - Temperature
5) Tref - Reference Temperature
6) **c** - Concentration
7) **n** - Molar Mass
8) V - Volume
9) H - Enthalpy
10) r - Reaction Rate
11) kb - Constant
12) kf - Constant
13) **nhat** - Molar Mass Flow
14) Vhat - Volumetric Flow
15) Hhat - Enthalpy Flow
16) k - Constant
17) q - Heat Flow
18) U - Overall Heat Transfer Coef
19) A - Area

Global Variables:
Tref = 0.0

DETAILS:

System 1 (= 8.1 Cooler)
Type: Lump
Species In System: {1}
Initial Conditions:
n = [0.1]
H = HFromT(n, spec.h0, spec.cp, 300, glob.Tref)

Constants:
V = 0.1

Equations:
c = n/V
T = (H - n'*spec.h0)/(n'*spec.cp) + glob.Tref
hs = spec.h0 + spec.cp*(T - glob.Tref)

System 2 (= 8.2 Reaction Phase)
Type: Lump
Species In System: {2, 3, 4, 5}
Initial Conditions:
n = [0.5; 20.0; 1.0; 0.0]
H = HFromT(n, spec.h0, spec.cp, 300, glob.Tref)

Constants:
V = 1.0

Equations:

c = n/V
T = (H - n'*spec.h0)/(n'*spec.cp) + glob.Tref
hs = spec.h0 + spec.cp*(T - glob.Tref)

Reactions In System:
1) **2A + 3B ==> 8C**
r = kf*sys.c(1)^2*sys.c(2)^3 - kb*sys.c(3)^8
(1) = A
(2) = B
(3) = C
kb = 0.08
kf = 0.11

System 3 (= 8.3 Extraction Phase)
Type: Lump
Species In System: {5, 6}
Initial Conditions:
n = [0.0; 21.0]
H = HFromT(n, spec.h0, spec.cp, 300, glob.Tref)

Constants:
V = 1.0

Equations:
c = n/V
T = (H - n'*spec.h0)/(n'*spec.cp) + glob.Tref
hs = spec.h0 + spec.cp*(T - glob.Tref)

System 4 (= 1 Coolant Source)
Type: Source
Species In System: {1}
Constants:
c = [1.0]
T = 300.0

Equations:
hs = spec.h0 + spec.cp*(T - glob.Tref)

System 5 (= 2 Coolant Sink)
Type: Sink
Species In System: {1}

System 6 (= 3 Feed A, D)
Type: Source
Species In System: {2, 3}
Constants:
c = [0.5; 10.0]
T = 300.0

Equations:
hs = spec.h0 + spec.cp*(T - glob.Tref)

System 7 (= 4 Feed B, D)
Type: Source
Species In System: {3, 4}
Constants:
c = [10.0; 1.0]
T = 300.0

Equations:
hs = spec.h0 + spec.cp*(T - glob.Tref)

System 8 (= 5 Feed E)
Type: Source
Species In System: {6}
Constants:
c = [21.0]
T = 300.0

Equations:
hs = spec.h0 + spec.cp*(T - glob.Tref)

System 9 (= 6 Reaction Phase Sink)
Type: Sink
Species In System: {2, 3, 4, 5}

System 10 (= 7 Extraction Phase Sink)
Type: Sink
Species In System: {5, 6}

Mass Connection m01
Origin: System 4 (= 1 Coolant Source)
Target: System 1 (= 8.1 Cooler)
Species Through Connection: {1}
Connection is Uni-directional
Vhat = 1.0

Equations:
nhat = or.c*Vhat
Hhat = or.hs'*nhat

Mass Connection m02
Origin: System 1 (= 8.1 Cooler)
Target: System 5 (= 2 Coolant Sink)
Species Through Connection: {1}
Connection is Uni-directional
Vhat = 1.0

Equations:
nhat = or.c*Vhat
Hhat = or.hs'*nhat

Mass Connection m03
Origin: System 6 (= 3 Feed A, D)
Target: System 2 (= 8.2 Reaction Phase)
Species Through Connection: {2, 3}
Connection is Uni-directional
Vhat = 0.01

Equations:
nhat = or.c*Vhat
Hhat = or.hs'*nhat

Mass Connection m04
Origin: System 7 (= 4 Feed B, D)
Target: System 2 (= 8.2 Reaction Phase)
Species Through Connection: {3, 4}
Connection is Uni-directional
Vhat = 0.01

Equations:
nhat = or.c*Vhat
Hhat = or.hs'*nhat

Mass Connection m05
Origin: System 8 (= 5 Feed E)
Target: System 3 (= 8.3 Extraction Phase)
Species Through Connection: {6}
Connection is Uni-directional
Vhat = 0.01

Equations:
nhat = or.c*Vhat
Hhat = or.hs'*nhat

Mass Connection m06
Origin: System 2 (= 8.2 Reaction Phase)
Target: System 9 (= 6 Reaction Phase Sink)
Species Through Connection: {2, 3, 4, 5}
Connection is Uni-directional
Vhat = 0.01

Equations:
nhat = or.c*Vhat
Hhat = or.hs'*nhat

Mass Connection m07
Origin: System 3 (= 8.3 Extraction Phase)
Target: System 10 (= 7 Extraction Phase Sink)
Species Through Connection: {5, 6}
Connection is Uni-directional
Vhat = 0.01

Equations:
nhat = or.c*Vhat
Hhat = or.hs*nhat

Mass Connection m08
Origin: System 2 (= 8.2 Reaction Phase)
Target: System 3 (= 8.3 Extraction Phase)
Species Through Connection: {5}
Connection is Bi-directional
k = 0.05

Equations:
nhat = k*(or.c - tar.c)
Hhat = ((diag(1+sign(nhat))*or.hs + diag(1-sign(nhat))*tar.hs)/2)*nhat

Heat Connection h01
Origin: System 1 (= 8.1 Cooler)
Target: System 2 (= 8.2 Reaction Phase)
U = 1.0E+3
A = 1.0

Equations:
q = U*A*(or.T - tar.T)

SPECIES DATA:

1) Species Q - Q
cp = 1.0E+3
h0 = 5.0E+5

2) Species A - A
cp = 1.0E+3
h0 = 5.0E+5

3) Species D - D
cp = 1.0E+3
h0 = 5.0E+5

4) Species B - B
cp = 800.0
h0 = 6.0E+5

5) Species C - C
h0 = -5.0E+6
cp = 1200.0

6) Species E - E
cp = 1.0E+3
h0 = 5.0E+5

```
% MODELLER Output File.

function mod_out = modcode

global WTBAR mod_out mod_int par
mod_out = [];
mod_int = [];
par = [];
massCons(1,:) = [4, 1];     % m01
massCons(2,:) = [1, 5];     % m02
massCons(3,:) = [6, 2];     % m03
massCons(4,:) = [7, 2];     % m04
massCons(5,:) = [8, 3];     % m05
massCons(6,:) = [2, 9];     % m06
massCons(7,:) = [3, 10];    % m07
massCons(8,:) = [2, 3];     % m08
 par.m08_PermOr = SelectionMatrix(4, [4]);
 par.m08_PermTar = SelectionMatrix(2, [1]);
heatCons(1,:) = [1, 2];     % h01

num = 3;      % Number of Lumps + Steady-State Systems

A_mass = StreamMatrix(num, massCons);
A_heat = StreamMatrix(num, heatCons);

numSpec = 6;      % Number of Species
I = speye(numSpec);
Ak = kron(A_mass, I);

GammaS = [];      % Initialise System Selection Matrix GammaS
GammaS = blkdiag(GammaS, SelectionMatrix(numSpec, [1]));     % System  1
GammaS = blkdiag(GammaS, SelectionMatrix(numSpec, [2, 3, 4, 5]));     % System  2
GammaS = blkdiag(GammaS, SelectionMatrix(numSpec, [5, 6]));     % System  3

S = [];      % Initialise Stoichiometric Matrix S
S = blkdiag(S, sparse(0, 1*numSpec));
S = blkdiag(S, sparse([0 -2 0 -3 8 0]));   % System  2
S = blkdiag(S, sparse(0, 1*numSpec));

GammaM = [];      % Initialise Mass Connection Selection Matrix GammaM
GammaM = blkdiag(GammaM, SelectionMatrix(numSpec, [1]));     % m01
GammaM = blkdiag(GammaM, SelectionMatrix(numSpec, [1]));     % m02
GammaM = blkdiag(GammaM, SelectionMatrix(numSpec, [2, 3]));     % m03
GammaM = blkdiag(GammaM, SelectionMatrix(numSpec, [3, 4]));     % m04
GammaM = blkdiag(GammaM, SelectionMatrix(numSpec, [6]));     % m05
GammaM = blkdiag(GammaM, SelectionMatrix(numSpec, [2, 3, 4, 5]));     % m06
GammaM = blkdiag(GammaM, SelectionMatrix(numSpec, [5, 6]));     % m07
GammaM = blkdiag(GammaM, SelectionMatrix(numSpec, [5]));     % m08

A_n = GammaS*Ak*GammaM';
par.A = A_n;
B_n = GammaS*S';
par.B = B_n;
par.Am = A_mass;
par.Aq = A_heat;

% Write Global Variables
par.Tref = 0.0;

% Initial Conditions:
xx(1:1, 1) = [0.1];      %  n_sys01
xx(2:5, 1) = [0.5;20.0;1.0;0.0];     %  n_sys02
xx(6:7, 1) = [0.0;21.0];     %  n_sys03
xx(8, 1) = HFromT(xx(1:1, 1), [5.0E+5], [1.0E+3], 300, par.Tref);      % H_sys01
xx(9, 1) = HFromT(xx(2:5, 1), [5.0E+5; 5.0E+5; 6.0E+5; -5.0E+6], [1.0E+3; 1.0E+3; 800.0; 1200.0], 300, par.Tref);
% H_sys02
xx(10, 1) = HFromT(xx(6:7, 1), [-5.0E+6; 5.0E+5], [1200.0; 1.0E+3], 300, par.Tref);      % H_sys03

par.tspan = 100;


% Call Solver
WTBAR = waitbar(0, 'Solving DAEs ......');
options = odeset('OutputFcn', @OutputBuild, 'MStateDependence', 'none', ...
 'AbsTol', 1E-10, 'RelTol', 0.001);

[t1, x1] = ode15s(@f, [0; par.tspan], xx, options);
close(WTBAR);

%  Build Extra Output Information
species = {'Q'; 'A'; 'D'; 'B'; 'C'; 'E'};
len_t = ones(length(mod_out.t),1);
mod_out.sys01.name = 'Cooler';
mod_out.sys01.id = '8.1';
mod_out.sys01.species = species([1]);
mod_out.sys01.V = len_t*0.1';
mod_out.sys02.name = 'Reaction Phase';
mod_out.sys02.id = '8.2';
mod_out.sys02.species = species([2, 3, 4, 5]);
mod_out.sys02.V = len_t*1.0';
mod_out.sys02.reaction1.formula = '2A + 3B ==> 8C';
mod_int.sys02.reaction1.kb = len_t*0.08';
mod_int.sys02.reaction1.kf = len_t*0.11';
mod_out.sys03.name = 'Extraction Phase';
mod_out.sys03.id = '8.3';
mod_out.sys03.species = species([5, 6]);
mod_out.sys03.V = len_t*1.0';
mod_out.sys04.name = 'Coolant Source';
mod_out.sys04.id = '1';
mod_out.sys04.species = species([1]);
mod_out.sys04.c = len_t*[1.0]';
mod_out.sys04.T = len_t*300.0';
mod_out.sys06.name = 'Feed A, D';
mod_out.sys06.id = '3';
mod_out.sys06.species = species([2, 3]);
mod_out.sys06.c = len_t*[0.5; 10.0]';
mod_out.sys06.T = len_t*300.0';
mod_out.sys07.name = 'Feed B, D';
mod_out.sys07.id = '4';
mod_out.sys07.species = species([3, 4]);
mod_out.sys07.c = len_t*[10.0; 1.0]';
mod_out.sys07.T = len_t*300.0';
mod_out.sys08.name = 'Feed E';
mod_out.sys08.id = '5';
mod_out.sys08.species = species([6]);
mod_out.sys08.c = len_t*[21.0]';
mod_out.sys08.T = len_t*300.0';
mod_out.con_m01.species = species([1]);
mod_out.con_m01.c = len_t*[1.0]';
mod_out.con_m01.Vhat = len_t*1.0';
mod_out.con_m02.species = species([1]);
mod_out.con_m02.Vhat = len_t*1.0';
mod_out.con_m03.species = species([2, 3]);
mod_out.con_m03.c = len_t*[0.5; 10.0]';
```

```
mod_out.con_m03.Vhat = len_t*0.01';
mod_out.con_m04.species = species([3, 4]);
mod_out.con_m04.c = len_t*[10.0; 1.0]';
mod_out.con_m04.Vhat = len_t*0.01';
mod_out.con_m05.species = species([6]);
mod_out.con_m05.c = len_t*[21.0]';
mod_out.con_m05.Vhat = len_t*0.01';
mod_out.con_m06.species = species([2, 3, 4, 5]);
mod_out.con_m06.Vhat = len_t*0.01';
mod_out.con_m07.species = species([5, 6]);
mod_out.con_m07.Vhat = len_t*0.01';
mod_out.con_m08.species = species([5]);
mod_out.con_m08.k = len_t*0.05';
mod_out.con_h01.U = len_t*1.0E+3';
mod_out.con_h01.A = len_t*1.0';

% ---------------------------------------------------------------------
function status = OutputBuild(t, xx, flag);
%  Build Output Information

global mod_out mod_int
status = 0;
if nargin < 3 | isempty(flag)
i = length(mod_out.t) + 1;

mod_out.t(i, 1) = t;
mod_out.sys01.n(i,:) = xx(1:1, 1);
mod_out.sys01.H(i,:) = xx(8, 1);
mod_out.sys01.c(i,:) = mod_int.c_sys01;
mod_out.sys01.T(i,:) = mod_int.T_sys01;
mod_out.sys01.hs(i,:) = mod_int.hs_sys01;
mod_out.sys02.n(i,:) = xx(2:5, 1);
mod_out.sys02.H(i,:) = xx(9, 1);
mod_out.sys02.c(i,:) = mod_int.c_sys02;
mod_out.sys02.T(i,:) = mod_int.T_sys02;
mod_out.sys02.hs(i,:) = mod_int.hs_sys02;
mod_out.sys02.reaction1.r(i,:) = mod_int.r(1);
mod_out.sys03.n(i,:) = xx(6:7, 1);
mod_out.sys03.H(i,:) = xx(10, 1);
mod_out.sys03.c(i,:) = mod_int.c_sys03;
mod_out.sys03.T(i,:) = mod_int.T_sys03;
mod_out.sys03.hs(i,:) = mod_int.hs_sys03;
mod_out.sys04.hs(i,:) = mod_int.hs_sys04;
mod_out.sys06.hs(i,:) = mod_int.hs_sys06;
mod_out.sys07.hs(i,:) = mod_int.hs_sys07;
mod_out.sys08.hs(i,:) = mod_int.hs_sys08;
mod_out.con_m01.nhat(i,:) = mod_int.nhat(1:1, 1)';
mod_out.con_m01.Hhat(i,:) = mod_int.Hhat(1);
mod_out.con_m02.nhat(i,:) = mod_int.nhat(2:2, 1)';
mod_out.con_m02.Hhat(i,:) = mod_int.Hhat(2);
mod_out.con_m03.nhat(i,:) = mod_int.nhat(3:4, 1)';
mod_out.con_m03.Hhat(i,:) = mod_int.Hhat(3);
mod_out.con_m04.nhat(i,:) = mod_int.nhat(5:6, 1)';
mod_out.con_m04.Hhat(i,:) = mod_int.Hhat(4);
mod_out.con_m05.nhat(i,:) = mod_int.nhat(7:7, 1)';
mod_out.con_m05.Hhat(i,:) = mod_int.Hhat(5);
mod_out.con_m06.nhat(i,:) = mod_int.nhat(8:11, 1)';
mod_out.con_m06.Hhat(i,:) = mod_int.Hhat(6);
mod_out.con_m07.nhat(i,:) = mod_int.nhat(12:13, 1)';
mod_out.con_m07.Hhat(i,:) = mod_int.Hhat(7);
mod_out.con_m08.nhat(i,:) = mod_int.nhat(14:14, 1)';
mod_out.con_m08.Hhat(i,:) = mod_int.Hhat(8);
mod_out.con_h01.q(i,:) = mod_int.q(1);
else
switch(flag)
case 'init'
mod_out.t = [];
OutputBuild(t(1), xx, '');
otherwise
end
end

% ---------------------------------------------------------------------

function dxdt = f(t, xx)
global WTBAR mod_int par;
waitbar(t/par.tspan, WTBAR);
% Initialise Rate Variables
mod_int.nhat = zeros(14, 1);
mod_int.Hhat = zeros(8, 1);
mod_int.q = zeros(1, 1);
mod_int.r = zeros(1, 1);

% System Equations:
% System  1   ( = 8.1  Cooler)
mod_int.c_sys01 = xx(1:1, 1)/0.1;
mod_int.T_sys01 = (xx(8, 1) - xx(1:1, 1)'*[5.0E+5])/(xx(1:1, 1)'*[1.0E+3]) + par.Tref;
mod_int.hs_sys01 = [5.0E+5] + [1.0E+3]*(mod_int.T_sys01 - par.Tref);

% System  2   ( = 8.2  Reaction Phase)
mod_int.c_sys02 = xx(2:5, 1)/1.0;
mod_int.T_sys02 = (xx(9, 1) - xx(2:5, 1)'*[5.0E+5; 5.0E+5; 6.0E+5; -5.0E+6])/(xx(2:5, 1)'*[1.0E+3; 1.0E+3; 800.0;
1200.0]) + par.Tref;
mod_int.hs_sys02 = [5.0E+5; 5.0E+5; 6.0E+5; -5.0E+6] + [1.0E+3; 1.0E+3; 800.0; 1200.0]*(mod_int.T_sys02 -
par.Tref);

% System  3   ( = 8.3  Extraction Phase)
mod_int.c_sys03 = xx(6:7, 1)/1.0;
mod_int.T_sys03 = (xx(10, 1) - xx(6:7, 1)'*[-5.0E+6; 5.0E+5])/(xx(6:7, 1)'*[1200.0; 1.0E+3]) + par.Tref;
mod_int.hs_sys03 = [-5.0E+6; 5.0E+5] + [1200.0; 1.0E+3]*(mod_int.T_sys03 - par.Tref);

% System  4   ( = 1  Coolant Source)
mod_int.hs_sys04 = [5.0E+5] + [1.0E+3]*(300.0 - par.Tref);

% System  6   ( = 3  Feed A, D)
mod_int.hs_sys06 = [5.0E+5; 5.0E+5] + [1.0E+3; 1.0E+3]*(300.0 - par.Tref);

% System  7   ( = 4  Feed B, D)
mod_int.hs_sys07 = [5.0E+5; 6.0E+5] + [1.0E+3; 800.0]*(300.0 - par.Tref);

% System  8   ( = 5  Feed E)
mod_int.hs_sys08 = [5.0E+5] + [1.0E+3]*(300.0 - par.Tref);


% Connection Equations:
% Connection m01

mod_int.nhat(1:1, 1) = [1.0]*1.0;
mod_int.Hhat(1, 1) = mod_int.hs_sys04'*mod_int.nhat(1:1, 1);

% Connection m02

mod_int.nhat(2:2, 1) = mod_int.c_sys01*1.0;
mod_int.Hhat(2, 1) = mod_int.hs_sys01'*mod_int.nhat(2:2, 1);
```

```
% Connection m03

mod_int.nhat(3:4, 1) = [0.5; 10.0]*0.01;
mod_int.Hhat(3, 1) = mod_int.hs_sys06'*mod_int.nhat(3:4, 1);

% Connection m04

mod_int.nhat(5:6, 1) = [10.0; 1.0]*0.01;
mod_int.Hhat(4, 1) = mod_int.hs_sys07'*mod_int.nhat(5:6, 1);

% Connection m05

mod_int.nhat(7:7, 1) = [21.0]*0.01;
mod_int.Hhat(5, 1) = mod_int.hs_sys08'*mod_int.nhat(7:7, 1);

% Connection m06

mod_int.nhat(8:11, 1) = mod_int.c_sys02*0.01;
mod_int.Hhat(6, 1) = mod_int.hs_sys02'*mod_int.nhat(8:11, 1);

% Connection m07

mod_int.nhat(12:13, 1) = mod_int.c_sys03*0.01;
mod_int.Hhat(7, 1) = mod_int.hs_sys03'*mod_int.nhat(12:13, 1);

% Connection m08

mod_int.nhat(14:14, 1) = 0.05*(par.m08_PermOr*mod_int.c_sys02 - par.m08_PermTar*mod_int.c_sys03);
mod_int.Hhat(8, 1) = ((diag(1+sign(mod_int.nhat(14:14, 1)))*par.m08_PermOr*mod_int.hs_sys02 + diag(1-
sign(mod_int.nhat(14:14, 1)))*par.m08_PermTar*mod_int.hs_sys03)/2)'*mod_int.nhat(14:14, 1);

% Connection h01

mod_int.q(1, 1) = 1.0E+3*1.0*(mod_int.T_sys01 - mod_int.T_sys02);


% Reaction Equations
% Reaction 1 2A + 3B ==> 8C   (System  2   ( = 8.2  Reaction Phase))
mod_int.r(1) = 0.11*mod_int.c_sys02(1)^2*mod_int.c_sys02(3)^3 - 0.08*mod_int.c_sys02(4)^8;


   % Dynamic Balance Equations
   dxdt(1:7) = par.A*mod_int.nhat + par.B*mod_int.r;
   dxdt(8:10) = par.Am*mod_int.Hhat + par.Aq*mod_int.q;
dxdt = dxdt';


% --------------------------------------------------------------------
function M = mass(t)
global par
M = par.MassMatrix;

% --------------------------------------------------------------------
function A = StreamMatrix(num, con)
% Builds the stream matrix for lumped systems and steady state systems.
% The systems have to be ordered. First lumps, then steady state, source
% and sink systems successively.
% num is the number of lumped systems plus the number of steady state systems.
% con is a (numCon x 2) matrix. First column are the origin systems of
% all the defined connections. Second column refers to the target systems.
```

```
if size(con,1) > 0
   A = sparse(num, size(con,1));
   for i=1:size(con,1)
      for j=1:2           % 1 = origin, 2 = target
         if con(i, j) <= num
            A(con(i,j), i) = (-1)^j;
         end
      end
   end
else
   A = 0;
end

% --------------------------------------------------------------------
%  SelectionMatrix
%           P = SelectionMatrix(numSpec, specs) Returns the 'Selection Matrix' P of the
%           vector specs (dimension M). P is a (sparse) identity matrix in which each
%           row i, for which i is not an element of specs, is eliminated.
%           So, if specs contains M species, then the dimension op P will be M-by-numSpec.

function P = SelectionMatrix(numSpec, specs)

if size(specs, 1)==1 & length(numSpec)==1,
   P = sparse(size(specs,2), numSpec);
   specs = sort(specs); % The species must be sorted from low to high
   for i=1:size(specs,2),
     P(i,specs(1,i)) = 1;
   end
elseif isempty(specs);
    P = sparse(0, numSpec);
else
  error('Input must be a row vector');
end

% --------------------------------------------------------------------
function H = HFromT(n, h0, cp, T, Tref);

H = n'*(h0 + cp*(T - Tref));
```

Samenvatting

Het belangrijkste doel van dit onderzoek was het ontwikkelen van een systematische modelleermethode voor het ontwerpen van op fysisch inzicht gebaseerde, dynamische modellen van macroscopische fysisch, chemisch en/of biologische processen. Het modelleren van dergelijke processen is een van de belangrijkste taken van een process engineer. Deze modellen worden namelijk op grote schaal gebruikt voor allerlei ingenieurs activiteiten, zoals procesregeling, optimalisatie, simulatie, procesontwerp en fundamenteel onderzoek. Het opstellen van deze modellen wordt, over het algemeen, als een moeilijke en zeer tijdrovende zaak gezien en wordt het liefst aan "experts" overgelaten. Dit hoeft niet zo te zijn als een duidelijke, stapsgewijze methode wordt aangehouden.

De modelleermethode is gebaseerd op het hiërarchisch opdelen van een proces in thermodynamische systemen en bestaat uit vijf stappen: **(i)** Deel het proces op in kleinere delen, die elk afzonderlijk goed (wiskundig) te beschrijven zijn. In deze stap wordt de zogeheten *Physical Topology* geconstrueerd, gebruik makend van slechts twee bouwstenen, namelijk *Systemen*, die capaciteiten voorstellen die de fundamenteel extensieve grootheden (zoals componenten massa, energie en impuls) kunnen opslaan, en *Connecties*, die de overdracht van de fundamenteel extensieve grootheden tussen deze systemen beschrijven. **(ii)** Construeer de *Species Topology*. De Species Topology beschrijft de distributie van chemische en biologische componenten, alsook hun transformaties in andere componenten in de verschillende delen van het proces in de physical topology. **(iii)** Schrijf voor elke relevante extensieve grootheid (zoals bijvoorbeeld componenten massa of energie) van elk systeem de corresponderende balansvergelijkingen. **(iv)** Voeg definities en toestandsvergelijkingen aan de modelvergelijkingen toe en kies de overdrachts- en reactievergelijkingen die het transport en de reacties beschrijven. Definieer ook eventuele beperkende aannames, welke eventueel tot hoge-index DAE's (Differential Algebraic Equations) kunnen leiden. De ontstane wiskundige problemen kunnen altijd door model reductie direct worden opgelost. **(v)** Voeg het dynamisch gedrag van informatie verwerkende units, zoals regelaars, toe.

Deze vijf stappen hoeven niet per sé in deze exacte volgorde te worden

afgewerkt - althans niet voor het complete model. Het wordt aan de model ontwerper overgelaten wanneer details aan specifieke delen van het proces worden toegevoegd.

Met behulp van de vijf stappen kan gemakkelijk een consistent wiskundig model van het onderzochte proces worden gegenereerd. Dit model kan direct gebruikt worden voor bijvoorbeeld simulaties of dynamisch ontwerp of het model kan eerst worden vereenvoudigd door bijvoorbeeld linearizatie of model reductie toe te passen.

Een ander belangrijk doel van het onderzoek is het implementeren van deze modelleermethode in een nieuw computerprogramma, "The Modeller". Dit programma is ontworpen met het doel modelontwerpers te assisteren bij het maken van consistente procesmodellen en de tijdsduur die hiervoor nodig is (ruim) te verkorten. Met dit programma kunnen op twee basis manieren proces modellen worden opgebouwd, namelijk door reeds gedefinieerde systemen te verfijnen (de *top-down* methode) of door systemen te groeperen (de *bottom-up* methode). De verschillende methoden kunnen door elkaar gebruikt worden en alle acties die worden uitgevoerd kunnen ongedaan gemaakt worden (multiple undo/redo mechanisme). Modellen (of delen daarvan) kunnen op elk moment van de model definitie worden opgeslagen, geladen, geïmporteerd of geëxporteerd. Op deze manier is er een veilig mechanisme voor hergebruik van modellen.

Het computerprogramma is geïmplementeerd met BlackBox Component Builder 1.4 (Component Pascal). Dit is een met Java te vergelijken component georiënteerde programmeertaal.

Curriculum Vitae

11 April 1973	Born in Oostburg.
1985 - 1991	Secondary School, Scholengemeenschap 't Zwin, Oostburg.
1991 - 1997	Master student at the Eindhoven University of Technology, Department of Chemical Technology, System and Control Group. Main Subject: Modelling.
1997 - 2002	PhD research project, as doctoral student (AiO-4) with the "Systems and Control Group" of the department of Chemical Technology, University of Technology Eindhoven.
2002 - 2003	3 month research project for FORD Automotive, constructing a dynamic simulation model of a catalytic converter for diesel engines, Aachen, Germany.
Hobbies	Collecting and playing musical instruments, board games, squash, pool billiard, computer programming, reading, spending time with wife and kids.

Index