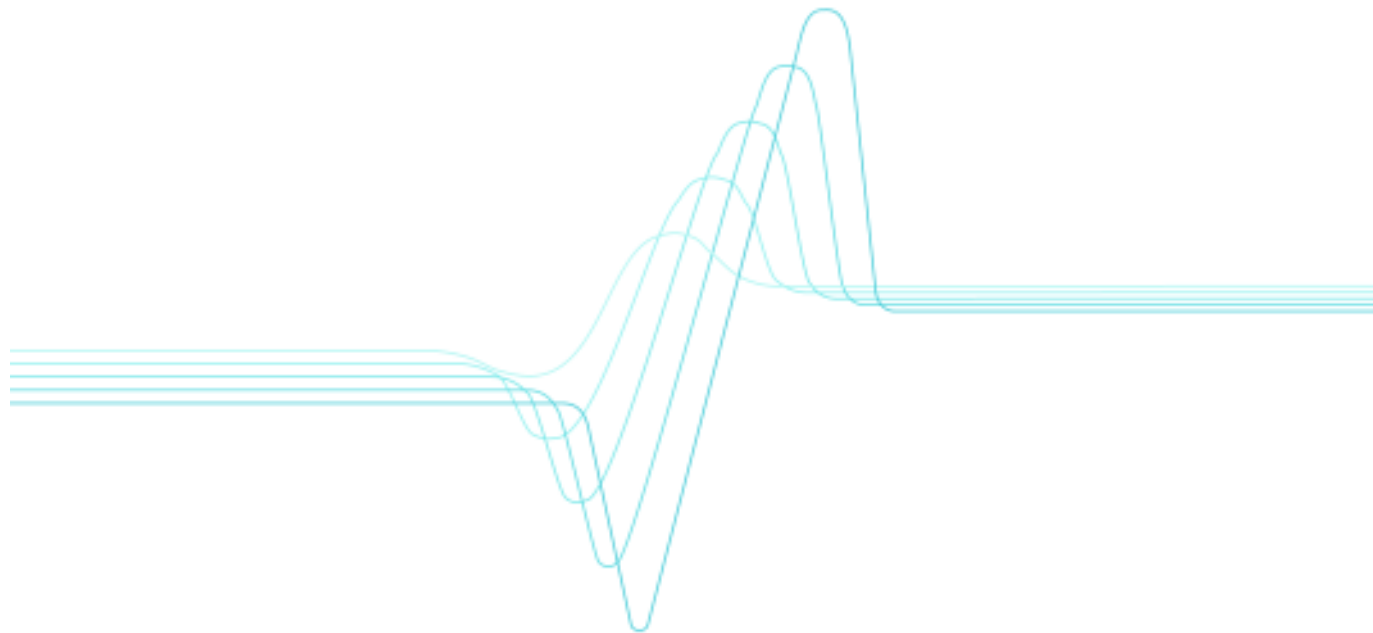Kaj Juslin

A Companion Model Approach to Modelling and Simulation of Industrial Processes

A Companion Model Approach to Modelling and Simulation of Industrial Processes

Kaj Juslin

VTT Industrial Systems

Abstract

Modelling and simulation represent tremendous possibilities if extensively taken up by engineers as a working method (IMTI 2003). However, if detailed writing of equations and programming is required in addition to inverse engineering efforts for finding the necessary design data, it will only remain attractive to mathematicians, physicists and computer programmers. When considering the use of modelling and simulation tools in an engineering design project, there is no time to write equations, to consult suppliers' experts, or to manually transfer data from one repository to another.

Available commercial modelling and simulation software for process and automation interoperability studies, the advances in process and automation design methodologies, evolving specification and communication standards, as well as applicable computer software architectures, have recently been dealt with in VTT Publications (Pasanen 2001; Karhela 2002). Internet based service repositories have developed rapidly, making it possible for equipment manufacturers to supply "extended products", including design data needed by engineers engaged in process and automation integration. A request has been recognized to make the modelling and simulation software suited for relevant structured design information as it is provided for in product and design databases. Initiatives involving automated model generation have been encouraged (IMTI 2002).

The companion model approach for specification and solution of process simulation models, as presented herein, is developed from the above premises. The focus is on how to tackle real world processes, which from modelling point of view are dynamic, very stiff, very nonlinear and only piecewise continuous, without extensive manual interventions of human experts. An additional challenge, to solve the arising equations fast and reliably, is dealt with, as well.

Preface

In the early seventies, my interest for the emerging science of computer simulation was initiated by Professor Lauri Aura, heading the Power Electronics Laboratory at Helsinki University of Technology. My favourite reading included Computer Analysis of Circuits (Comer 1971) and Numerical Initial Value Problems in Ordinary Differential Equations (Gear 1971). Not very surprisingly, my master's thesis dealt with the simulation of silicon controlled rectifier circuits (Juslin 1973). At VTT, I became engaged with model-based testing of control concepts and real control systems, first focusing on the dynamics of nuclear power plants (Juusela & Juslin 1976).

The basic ideas presented in this thesis were initially applied to speed up the simulation of the full-scope training simulator of the Loviisa nuclear power plant (Juslin 1983a). Later on, they were implemented into the APROS simulation software in VTT's research program on Numerical Simulation of Processes, 1986–1988. Application specific libraries were concluded for nuclear power plants, combustion power plants, and paper mills. Hundreds of users all around the world gave useful feedback. New requests were raised in discussions within the APROS team. The service-oriented architectures became forerunners in software implementations. The development of an appropriate architectural framework was called for and it became a fascinating hobby for me. I acknowledge the related financial support from Fortum Foundation and VTT making it possible to conclude the developments in this thesis.

I want to thank my supervisor Professor Heikki Koivo for his support and tutoring in writing the thesis. I am grateful to Associate Professor Niels Houbak from Technical University of Denmark and Professor (Emeritus) Raimo Ylinen from University of Oulu for providing expert criticism and valuable suggestions. I am very obliged to all my colleagues, nobody named or forgotten, for their inspiration and support. In particular, I want to express my cordial gratitude to my near people, for their understanding and patience, during all the times I have been deeply occupied by my computer at home, faithfully accompanied by my cat.

Helsinki, April 2005

Kaj Juslin

Contents

Appendices

A: Specification Data View

B: Solution Data View

Abbreviations

ACL   APROS Communication Library

APROS Advanced Process Simulator

ASCII   American Standard Code for Information Interchange

ASL   APROS Specification Language

CAD   Computer Aided Design

CAE   Computer Aided Engineering

CCGT   Combined Cycle Gas Turbine

CSSL   Continuous System Simulation Language

DAE   Differential Algebraic Equation

DCS   Distributed Control System

DLL   Dynamically Linked Libraries

EPRI   Electric Power Research Institute

ERP   Enterprise Resource Planning

IAEA   International Atomic Energy Agency

ICT   Information and Communication Technology

IMACS International Society for Mathematics and Computers in Simulation

IMTI   Integrated Manufacturing Technology Initiative

LTE	Linear Transition Equation
MES	Manufacturing Execution System
MTE	Mechanistic Transition Equation
OECD	Organisation for Economic Co-operation and Development
OLE	Object Linking and Embedding
OPC	OLE for Process Control
OWL	Web Ontology Language
PDM	Product Data Management
PLM	Product Lifecycle Management
RDF	Resource Description Framework
SOA	Service-Oriented Architectures
SVG	Scaleable Vector Graphics
VCM	Versatile Companion Model
VTT	Technical Research Centre of Finland
W3C	World Wide Web Consortium
VDI	Verein Deutscher Ingenieure
X3D	Extensible 3D Graphics
XML	Extensible Mark-Up Language

Glossary

Absolute Stable
An absolute stable numerical integration method does not impose any stability restrictions on the applied step-length (Lambert 1991).

Companion Model
A companion model is a dual representation of an equation that equivalently describes for instance a specific physical dependency. See Subsection 3.4.1 regarding companion models.

Full-Scope Replica Training Simulator
A training simulator that replicates the real control room and includes all the models needed to reproduce the functionality of the real plant that can be operated or observed from the control room.

Integration Error
An integration error occurs whence applying numerical integration on a digital computer. At large step-lengths it is dominated by truncation errors, whereas round-off errors dominate at small step-lengths.

Ill-Conditioned Problem
A problem is considered ill-conditioned if very small relative perturbations in the parameters make relatively large variations in the solution.

Iteration Error
An iteration error occurs when the numerical iteration is stopped before the converged solution is achieved.

Monotonic Function
A continuously increasing or decreasing function that can be solved by Newton's iteration from an arbitrary starting point is considered as monotonic. See Subsection 4.6.3.

Node Matrix Equation
A node matrix equation is set up to calculate the node variables in a physical network. See Subsection 4.6.1. If applied to an electrical network the node voltages are calculated based on relevant branch admittances and source currents.

Precisely Absolute Stable
A precisely absolute stable numerical integration method has a stability region comprising exactly the left half plane (Lambert 1991).

Round-Off Error
A round-off error arises in an algebraic operation when the accuracy of the result is deteriorated because of limited number of available digits for calculation and storage (Wilkinson 1994).

Sparse Matrix
A matrix including so many zero elements that it pays off to use sparse matrix technique when solving the related linear equation system is regarded as sparse. See Subsection 4.6.2.

Stiff System
A dynamic system involving so much differing time constants that implicit methods are preferred for the solution of the relevant differential equation system is considered as stiff. In a stiff system the ratio of the largest and smallest absolute value of its eigenvalues is large (Lambert 1991).

Strongly Absolute Stable
A strongly absolute stable numerical integration method has a stability region that includes the whole left plane and in addition extends to parts of the right plane (Lambert 1991).

Truncation Error
A truncation error arises when a series approximation is truncated leaving the remaining terms unconsidered. Higher order numerical integration methods typically cause smaller truncation errors.

1. Introduction

1.1 Digital Information Society Challenges

Traditional document management easily results in piles of papers. Documents are nowadays very easily produced, which, however, does not help up the situation. Finding the required data and updating the documentation has become the real problem. In practice, separate piles have been gathered for the actors involved in distinct plant life-cycle phases such as design, construction, operation and maintenance, or in the use and disposal of ready-made products. Severe gaps in the information flow have been identified. Now there is a growing demand for computerized information management that would cover the whole value chain including various stakeholders, such as: project consultants, raw material and equipment suppliers, product manufacturers, maintenance service providers, wholesalers, distributors and customers. Recycling leads to closed loops within the chains. High quality and security records require traceability of the raw materials as well as transportation and manufacturing sequences applied to the products. Such information needs to be available to all the stakeholders, especially to the customers.

The modern Information and Communication Technology (ICT) already enables life-cycle lasting information management of products or production processes. There are computerised tools and data repositories with regard to drawings, component data, functional descriptions, operational procedures, measurement recordings, maintenance procedures, commercial and other knowledge e.g. related to markets, product quality, legal and environmental issues. In automotive industry, the required formal specifications of interfaces between the different tools and actors have readily been developed, enabling efficient co-operation of subcontractors in supply chains, as well as easy application of various design and service software tools. Evolving standard interfaces make it possible to build up Enterprise Resource Planning (ERP) systems, Manufacturing Execution Systems (MES) and Distributed Control Systems (DCS) piece by piece and to find the best suiting service tools from a large variety of software vendors. In the process industry, the standardisation process at large has just begun.

Computerised simulation models constitute to the various kind of formal documentation available. Different models are needed for different purposes, such as 3-D analysis, process integration analysis, control system design evaluation, optimisation of operational procedures, operator support systems, operator training simulators, pro-active maintenance, production management, logistics, and control of business processes. A wide take-up and implementation of formal model interfaces and data specifications as well as development of standard simulation platforms, will enable re-use of once finalised and verified computerised models for different purposes, at different life-cycle stages and by different actors.

An important requirement has been that all necessary data for the simulation model specifications shall be possible to obtain from product databases and computerised design drawings. Therefore, it is utterly important to certify that product databases really include such information as needed. The process equipment manufacturers are obviously the best experts to provide the relevant information on their own products. The computerised process and automation design drawings shall depict the integration of the specified components and formally replicate in required detail the relevant process structure, the positioning of the components and their connections.

1.2 Knowledge Management and Process Models

A restructuring is going on in the organisation of business operations in process industry. Networking trends have led to outsourcing of less strategic operations, justified by the possibility to concentrate on key business sectors. This development is underpinning the manufacturing industry actors to take up the challenge in service sector, extending their responsibility to the whole life-cycle of their products, including the construction, commissioning, operation, and maintenance.

The critical mass of multidisciplinary process knowledge usually available at a production site has until now been based on highly skilled personnel with a long record of gained experience, sometimes already commencing from the construction of the plant. Ageing and increased mobility of employees, calls for new recruitments, enhanced training or other solutions. Networking and

outsourcing are topical issues. A trend towards highly specialised knowledge centres is foreseen. Remote support and service functions are established. Restructuring the business operations puts high demands on knowledge management. There are several open issues to solve, where modelling and simulation could play a significant role in making the tacit knowledge of experienced personnel more explicit:

- How to make the knowledge of design engineers easily available and understandable for operational and maintenance personnel?

- How to store and transfer the knowledge of experienced operators to novices?

- How to manage changes in industrial processes as well as in business processes?

There is no simple answer to these questions. Significant developments have been made with regard to Product Data Management (PDM) and Product Lifecycle Management (PLM) systems. Certainly, rigorous model specification repositories could complement the list of available support facilities. The successive take up of e-business has been one driving force to make services and required data literally ubiquitous according to the needs of global networked business. The e-business is a forerunner with regard to the digitalisation of other operations.

It has been stated that a picture could contain more information than a thousand of words, and that a computerised simulation model could contain more knowledge than a thousand of pictures. There would be several expectations from an intelligent model, if just available:

- The process designer would like to have a model to evaluate new features,

- The automation engineer would like to test new control functionality,

- The production engineer would like to optimise the use of the production units,

- The operator would like to check out operational procedures in advance, and

- The trouble-shooter would like to have a what-if tool for diagnostics.

Is it possible to develop, update and re-use models suitable for all these purposes? Could the requirements for such models be specified?

Relevant models of industrial processes need to be dynamic and have a large enough operational range to reliably replicate the real plant in all operational situations of interest. The acceptance criteria of full-scope training simulators for power plants require that an experienced operator shall not notice any such performance deviation that could violate the purpose of the training. It shall be possible to use the model not only for studies of normal operational procedures, but also for emergency situations.

However, the model shall not be too complicated. It shall be easy to assign parameters to the model. Relevant data possibly available from design databases shall be easy to import. The models shall be easy to combine. Integrated models of large industrial plants shall run in real-time, or even much faster for prediction and optimisation purposes. It shall be possible to attach such models to on-line process measurements, real control systems, optimising tools, user interfaces and documentation software. It shall be possible to model a plant correctly enough, even before any measurements are available from the plant. It shall be easy to fine-tune the model when measurements are available.

1.3 Disparate Tools and Requirements

There is a large variety of simulation tools available on the market. This implies that it shall be possible to choose the best suitable tool for a specific purpose. However, sometimes the effort in learning to use a new tool, just for one case, might be larger than the benefit. On the other hand, choosing the wrong tool has resulted in very discouraging experiences and reluctance for simulation in general. Choosing simulation software for dynamic process simulation purposes only relying on commercial brochures is not an easy task. It is not even easy to recognize what type of simulation model is needed for a specific purpose.

- *General Purpose Simulation Language* models have been very useful at university level for teaching purposes. The student can efficiently learn how to specify the equations for small-scale models. However, it is by far too laborious and error prone to specify equation by equation a complete production process.

- *Extended Steady-State Simulation* models have been used for tank level dynamic studies. In general, they do not fulfil the needs for dependable control system functionality testing at a sufficient large operational and dynamic range.

- *Transfer Function* models implemented with control system function blocks were previously used for control system design evaluation. However, they only supported studies of small transients close to a selected steady state. Shut down and start up studies were completely out of scope.

- *Accident Analysis Code* models have required highly skilled physicist for their input specification. For instance, the user has been fully responsible for the successful spatial discretisation of the process flow sheet to suitable control volumes. Analysis programs have usually not been optimized for real-time large-scale applications.

- *Full-Scope Training Simulator* models have not been flexible enough for such easy implementation of changes in the process scheme as needed in an interactive design and engineering.

- *Engineering Simulator Software* models should have the accuracy of accident analyser codes, the scalability of full-scope training simulators and the flexibility required for interfacing to semantic design databases.

An early requirement for the developments described in this thesis was to facilitate for efficient dynamic simulation of the entire power plant process in real-time in a full-scope training simulator. Later on evolving requirements focused on the capability needed for testing of the functionality of modern control systems. Human interface system response times of 250 ms are specified as a preference for control systems in nuclear power plants (NRC 1996). A

functioning control system test bench model shall calculate all necessary process measurements with the frequencies needed by the control system. Operation in real-time with a time-step of 50 ms or even shorter for distinct parts of the process model shall be possible. It is evident that appropriate software has to be particularly developed for such a purpose.

The software shall be equipped with model libraries of relevant process components. It shall comply with the requirement of easy and fast graphically supported model configuration. It shall have access to manufacturer specific parameter repositories required by the generic component models. As an upcoming requirement, it shall provide the possibility to automatically generate the model framework and parameters starting from CAD and PDM information available for the plant. This kind of integration of tools will be possible subject to an extensive take up of specification standards in process industry, as well as by simulation software suppliers. The adequate simulation engine needs to be suited for automated dynamic model specification using presently available design data-bases as well as future semantic knowledge repositories. Such a unified simulation engine has not been available on the market.

1.4 Software for Power Plant Training Simulators

The decisive requirements imposed from the early beginning on developments of the versatile companion model approach described in this thesis was that the simulation algorithms and platform shall be fast, scalable, and fail-safe enough especially for use in full-scope replica training simulators of nuclear and fossil-fuelled power plants. A review of software tools for development of real-time power plant simulators is thus of interest. In the following, only such suppliers are listed that recently have supplied modelling and simulation software for full-scope replica nuclear or fossil fuelled power plant simulators.

- Corys T.E.S.S. has developed *Alises*, the fully integrated modelling and simulation environment in cooperation with Tractebel Energy Engineering, Belgium. It is designed to produce high-fidelity dynamic simulation models with less effort, lower development times and lower costs. Unmatched flexibility in model building, combining different types of powerful solvers of differential equations governing the thermal

hydraulic and electrical processes are combined with sequential solvers of algebraic equations governing the behaviour of control logics and regulations. Corys has more than 20 years of experience in the field (Corys 2005).

- GSE Systems Inc. has developed the high fidelity simulation platform *SimSuite Power*. It includes all the tools necessary for effective and complete power plant modelling to enable the development of training simulators for both nuclear balance of plant and fossil power plants. The products are based on more that 30 years of accumulated knowledge in the field of power plant simulation, originating from Singer-Link, General Physics International, S3 Technologies, and EuroSim AB. GSE Systems Inc., together with its RNI Technologies and GSE Power Systems divisions, provide a wide range of solutions to the world's nuclear simulation market (GSE 2005).

- The SimSci-Esscor solution of Invensys Systems Inc. includes the *DYNSIM-Power* dynamic simulation system for power industries applications. It is based on 35 years of experience. An intuitive graphical interface enables to keep the simulator models current with the plant (Invensys 2005). Invensys and Hyperion Systems Engineering Ltd announced in January 2005 an exclusive operator training simulator partnership agreement (Hyperion 2005).

- L-3 Communications Mapps Inc. was set up in February 2005 including power plant simulation business segments from CAE Inc. With over three decades of experience in power plant simulation developments Mapps Inc. has pioneered many of the principal advances in simulator design and functionality introducing real-time graphical component-based simulation in 1991 and making power plant simulation available over the Internet in 2001. The rapid model development environment *ROSE* includes single phase and two-phase thermal-hydraulics solvers (Mapps 2005).

- *Microfusion Engineering Laboratories. Inc*. has developed since 1995 an advanced, first principles, precision simulation engine called *THINK*, Thermal Hydraulic Integrated NetworK. It solves the time and spatially

dependent equations for the conservation of mass, momentum and energy using an implicit integration method to allow for real or faster than real-time simulation. It decomposes mixture flow into the basic component liquid and vapour flow rates and allows noncondensibles and solids to be present with the stream (Microfusion 2005).

- nHance Technologies Inc. was formerly the Simulation Services unit of Framatome Technologies Inc. before Framatome ANP, Advanced Nuclear Power, divested the unit in March 2001. nHance Technologies Inc. has developed the *MMS-RTC*, Real-Time Capable Modular Modelling System, including nuclear power plant component libraries and graphical user interface. Framatome ANP Inc. is a wholly owned subsidiary of Framatome ANP SAS, France, created in anticipation of the joint venture of the nuclear businesses between Framatome SA and Siemens AG (nHance 2005). The original MMS, developed by EPRI, the Electrical Power Research Institute, Palo Alto, California, USA, was based on the ACSL simulation language (Mitchell 1991).

- High-fidelity training simulators from Rheinmetall Defence Electronics GmbH provide a full reproduction of the control room equipment. All functions and processes of the real plant can be simulated by extensive mathematical models computed in real-time whilst running on modern multiprocessor computers. Rheinmetall was taking control of STN ATLAS Elektronik GmbH's simulation technology units in August 2003. The accumulated record of deliveries includes pressurized and boiling water reactors as well as graphite moderated reactors (Rheinmetall 2005).

- *SimPort* of Science Applications International Corporation (SAIC) is an object oriented real-time simulation environment for nuclear and fossil power plants. This leading edge simulation technology incorporates RELAP5 R/T developed by Data Systems & Solutions together with Idaho National Engineering Laboratories (INEL 2005). Data Systems & Solutions is a Rolls-Royce and SAIC owned company (SAIC 2005).

- Tecnatom s.a. develops simulators including thermal hydraulic and neutronic codes with best estimate quality adapted to real-time.

20

Tecnatom uses *TRAC-RT* a Real-Time implementation of the Transient Reactor Analysis Code developed at Los Alamos National Laboratory (LANL 2005). The man-machine interface used is based on its own developments, as well (Tecnatom 2005).

- Thales Training and Simulation is part of the worldwide Thales Group. Thales uses its generic models to cover the fluid networks (steam, water, oil, gas, etc…), the instrumentation and control circuits, and the electrical networks. These generic models use CAD-based code generators. The associated *CMS*, Configuration Management System, is particularly suited to the production and maintenance of large amounts of application software. Thales has more than 25 years of experience in the design and manufacturing of power plant simulators originating from LMT and Thomson in France, Rediffusion and Link Miles in UK, Wormald Technology in Australia, and Burtec in USA (Thales 2005).

- Trax LLC has since 1987 provided both full-scope and partial scope operator training simulators for various fossil power plant designs and process industry worldwide based on the *ProTRAX* simulation system. The ProTRAX graphic-based model building system performs all required code manipulation to generate an executable FORTRAN program (Trax 2005).

- Technical Research Centre of Finland (VTT) has in co-operation with Fortum Nuclear Services Ltd developed *APROS*, the Advanced Process Simulator software. It enables real-time simulation of both homogeneous and non-equilibrium thermal-hydraulic flows. It is applied by VTT's customers both for detailed engineering analysis work and for construction of complete full-scope replica training simulators. It includes both dedicated (ACL) and standardised (OPC) real-time communication interfaces and it has successfully been used for pre-testing of real control system configurations (VTT 2005).

- The full-scope nuclear power plant simulator from Westinghouse provides a plant control room replica for the training of nuclear power plant operators. The simulator emulates plant-specific thermal-hydraulic, electrical, and instrumentation and control systems. The

thermal-hydraulic models of *SIMARC*, SIMulator Advanced Real-time Code, simulate the primary and secondary thermal-hydraulic behaviour of the plant's nuclear steam supply system. The models are based on design-grade nuclear plant analysis codes (Westinghouse 2005).

- Yokogawa simulators serve a diverse range of customers worldwide in the field of fossil fired, gas turbine, and co-generation power plants by its *TICSS*, TechComm Integrated Configuration and Simulation System. TechComm Simulation in Australia become a subsidiary of Yokogava in October 1998 (Yokogava 2005).

It shall be noted that the trade marks and business names encountered herein belong to their relevant owners. Many of the products have a long history of developments. Some of the real-time simulation software tools are clearly based on previously developed platforms or analysis codes. Sometimes the accumulation of experience can be traced from the curriculum of developers involved. How the codes have been enhanced for the real-time simulation purpose has usually not been presented in public. The software tools involved are usually supplied in connection to training simulator deliveries.

1.5 Purpose and Methodology

The purpose of this thesis is to introduce for the scientific community an algorithmic framework that enables fast running accurate simulation of large-scale heterogeneous industrial processes, and to present a novel data architecture for its implementation intended to provide for straightforward connection to evolving semantic design data-bases.

The algorithms presented herein have already twice been implemented in commercial software platforms. The functionality of the algorithmic approach has been conclusively demonstrated by an extensive amount of recorded experiences from implementations and applications as described in Chapter 2. The different appearances of the previously undisclosed *Versatile Companion Model* as linear graph, linear equation, linear dynamic equation and nonlinear dynamic equation are presented Chapter 3. The strongly related mathematical solution system architecture is outlined in Chapter 4. The fundamental

algorithmic principles have not previously been presented in public. Neither have the data architectures of the existing implementations.

The detailed specification of completely new layered data architecture, matching both the algorithmic and semantic requirements encountered has been concluded in two separate appendices focusing on the *Specification Data View* and the *Solution Data View*. The new data architecture has to be considered as a synthesis of previous implementation experiences. It is included to form a basis for future implementation work. Recommendations for efficient implementation are given both with regard to the real-time data organisation in computer memory and the accomplishment of the various algorithms.

1.6 Presentation of Algorithms and Data Structures

The presentation of algorithms and data structures in this publication shall give a general understanding for a reader, neither being a specialist in computer programming nor in software specification standards. Thus, no preference is given to any programming environment, modelling tool, or to any commercial database.

To achieve a very compact and illustrative appearance, the author has introduced structured tables for presentation of the algorithmic architectures as well as for the relevant data structures. When applicable, the variables and parameters appearing in the equations or tables are described in the concerned chapters.

Whilst entering into the different technical domains, the author doesn't intend to cover any complete set of physical mechanisms. The selection presented shall however demonstrate a sufficient range of possibilities to provide a concrete basis for further implementations.

1.7 Lumped-Parameter Dynamic Systems

Basic concepts regarding modelling and dynamic simulation of lumped-parameter continuous-time systems are introduced. We shall first consider linear

time-invariant first-order systems that can be modelled by a *state equation* written in a differential vector-matrix form

$$\underline{x}'(t) = A\underline{x}(t) + B\underline{u}(t)$$

together with an *output equation* written in a algebraic vector-matrix form

$$\underline{y}(t) = C\underline{x}(t).$$

All variables related to the *state vector* $\underline{x}$, the derivative of the state vector with respect to time $\underline{x}'$, the *input vector* to the system $\underline{u}$ as well as the *output vector* from the system $\underline{y}$ are considered at the same time instance t. All elements of the *state matrix A*, the *input matrix B* and the *output matrix C* are constant. A is an $n \times n$ matrix, B is an $n \times p$ and C is an $n \times q$ matrix, where n is the number of state variables, p the number of input variables and q the number of output variables. The traditional *system equations* comprise of the related state and output equations.

We shall now consider how to calculate the direct solution of the state variables at a new time instance when we know the initial state values at the time t. The state vector $\underline{x}_e$ and output vector $\underline{x}_e$ can be exactly solved at a new time instance e by the relevant difference equations in vector-matrix form

$$\underline{x}_e = \Psi(\Delta_e)\underline{x}_t + \Gamma(\Delta_e)\,\underline{z}_t$$

$$\underline{y}_e = C\underline{x}_e.$$

where Δ_e is the applied time-step, $\Psi(\Delta_e)$ the *state difference matrix, $\Gamma(\Delta_e)$* the *input difference matrix* and $\underline{z}_t$ the input vector at the old time instance t. The state difference matrix can be written as the exponent of matrix A for the applied time-step Δ_e

$$\Psi(\Delta_e) = exp\,[A(\Delta_e)].$$

When calculating the input difference matrix the exponent of A is integrated from $\boldsymbol{0}$ to Δ_e

$$\Gamma(\Delta_e) = \{ \int exp \, [A(\tau)] \ d\tau \ \} \ B.$$

The calculation of the state transfer matrix (Porter & Crossley 1972) for a time interval τ

$$exp[A(\tau)] = U\{exp \, [\, \Lambda(\tau) \,] \, \} \, U^{-1}$$

implies calculation of the eigenvalues λ_i, which are solved from determinant

$$|A-\lambda I| = 0,$$

the relevant eigenvectors $\underline{u}_i$ from the equations

$$A\underline{u}_i = \lambda_i \, \underline{u}_i$$

the modal matrix from the eigenvectors

$$U = [\underline{u}_1,...\underline{u}_n],$$

and, subject that the eigenvalues are distinct, the eigenvalue matrix

$$\Lambda = diag[\lambda_1,... \, \lambda_n],$$

and its exponent

$$exp \, [\, \Lambda(\tau) \,]= diag[exp(\lambda_i\tau), ... \, exp(\lambda_n\tau)].$$

The *difference equations* are feasible to use with relatively small linear time-invariant systems which can be studied with constant time-step whereas the state and input difference matrices only need to be calculated once. In our case, we need to solve large sparce nonlinear systems. In addition, we need to change the time-step lengths in order to manage with the discontinuities.

The original *state equations* are well suited for processes with an inherent structure that can be depicted by input/output block diagrams, like control systems, and where the derivatives of the state variables are easily attained in an explicit form. In Section 4.3 numerical integration methods are described that

can be used to solve state equations having the derivative of the state vector $\underline{x}'(t)$ readily available.

For integrated processes that inherently have a networked structure including network branches with transition flow variables like current v_{ij} and network nodes with local property variables like voltage u_i, there are many possibilities to write the relevant integrated equations. The first issue is to define intermediate nodes in long branches if necessary in order to reach necessary accuracy with lumped parameters or to avoid higher than first order equations describing the branches. Just writing the equations according to Kirchoff's law encounters for instance the following type of equations: The algebraic sum of branch currents to each node is equal to zero, and the algebraic sum of voltage drops around each loop is equal to zero.

Caution should be taken that each loop has an own branch not included in other loops; otherwise, the system will be over determined. Further, no loop can include only capacitors. For a resistive network the branch, currents and the node voltages can be solved from the arising matrix equation subject that the voltage of one reference node is known. It is an algebraic equation system. Each inductor or capacitor included in the network provides for one additional dynamic state variable, the change rate of current and the accumulated charge respectively. Now a differential algebraic equation system is obtained. It is pointed out that organising this equation system into state equation form is a large effort and it usually results in loss of the sparsity (Brenan et al. 1989).

Specific *loop matrix equations* have also been applied where the loop currents are the basic variables to be solved and the node voltages and currents are the secondary variables (Juslin 1973). The method had some benefit before the introduction of sparse matrix methods because the basic number of loops and accordingly the number of state variables is usually smaller than the number of nodes.

Applying the companion model approach and assigning the node voltage vector $\underline{u}$ as basic variables to be solved and the branch current vector $\underline{v}$ to secondary variables as set forth in Subsection 4.6.1 we get the *node matrix equation* of the form Au = b to first solve the node voltages whereupon the branch currents can be solved from each branch equation.

1.8 Unified Modelling and Simulation System

In this section basic features of the integrated solver and data architecture of a *Unified Modelling and Simulation System* are introduced. Distinctive features and improvements with regard to the foreseen functionality are concluded as follows:

- A *Modelling Engine* supports concurrent engineering in a virtual work group environment and connects to individual modelling interfaces, to main project repositories for relevant process and automation design information, to supplier repositories for process component data, and to joint integrated model specifications.

- The *Hierarchical Model Specification Formalism* introduced makes use of design requirements and relevant information regarding process dimensions, connections, physical mechanisms and empiric correlations.

- The formal specification facilitates automated generation of the required parameters of the related *Mechanistic Transition Equations, Linear Transition Equations,* and *Versatile Companion Models* in accordance with linearisation and discretisation methods employed.

- A set of *Elementary Physical Mechanisms* covering several different physical domains is made available for the detailed model specification. Each mechanism has a pre-defined set of specification attributes regarding both parameters and connections.

- Nonlinear correlations are pre-analysed and, if required, divided into distinct *Monotonic Regions* to allow for efficient and stable iterative correction of the nonlinearities.

- A methodology is developed to manage the *Crossing of Break Points* such as monotonic region borders or temporally induced discontinuities. The time-step is adjusted to hit first approaching break point. The numerical integration scheme is adapted to the situation.

- The hierarchical model specification is subsequently flattened down to a *Linear System Graph*. The resulting heterogeneous graph is reorganised into a hierarchical set of *Homogeneous Zones* applying tearing and combining procedures based on *Model Coefficient Analysis*. Most suited time-steps are applied to the numerical integration in relevant homogeneous zones.

- The structure of the matrix equations arising from each identified *Implicit Island* is pre-optimised (see Section 4.6) with regard to positive definiteness, diagonal dominance, sparsity and symmetry. Time consuming checks and pivoting procedures are accordingly not needed during the calculation.

- *Prediction of State Variables* enables calculation of nonlinear parameter estimates before entering into a new time-step. Excessive iterations are accordingly avoided.

- The *Vectorised Solver Architecture* ensures that state variables are assigned to positions according to their sequence of use. This enables pipelined processing and also high hit rate of cache memory.

- The completely *Table Driven Simulation Engine* ensures that neither compiling nor linking procedures are needed upon changes in model parameters or configuration. However, for embedded time-critical applications a further optimised *Run-Time Model* can be generated.

All these issues are dealt with in more or less detail in following chapters. Both top down and bottom up approaches are applied to enlighten the description of the architecture. The historical perspective provides for substantiation.

1.9 Contents of the Main Chapters

Guidance to the reader is found necessary. The study is concluded in following main chapters:

Introduction. The role of rigorous dynamic simulation models as knowledge repositories is emphasised. The requirements on modelling tools for dynamic simulation of full-scope industrial processes in real-time are dealt with. Relevant existing simulation suppliers and tools are listed. The limitations of available tools are discussed. Suggested enhancements are presented. Combination of diverse modelling paradigms, such as mechanistic and empiric modelling, has to be employed to obtain successful results.

Implementations and Applications. The first implementation aimed at improving the performance of the full-scope training simulator at the Loviisa nuclear power plant. The idea to develop an engineering simulator was emerging in mid eighties (Kurki & Jokela 1984) to aid the control system concept evaluation of conventional power plants. The recognised need to develop modular nuclear plant analyser software, as well, resulted in establishing the Advanced Process Simulator (APROS) software development project in the framework of a research program on Numerical Simulation of Processes at VTT (Mattila 1987). Subsequent initiatives to further develop the simulation platform and its libraries to support modelling of a larger variety of industrial processes are presented, as well.

Model Specification Architecture. Nodes and branches in a structured graph has been considered suitable for the specification of interconnections and dependencies of an industrial process plant. The alternative causality modes of the *Versatile Companion Model* are explained. Its graph and equivalent static, dynamic and nonlinear representations are described. Higher-level models are defined by combining the lover-level models. The whole plant is specified in a hierarchic manner, suitable for a *Vertical Interfaces* with CAD-type graphics tools and semantic design databases.

Solution System Architecture. The solver architecture is outlined. Its *Horizontal Interface* focuses on on-line communication with external models, other software tools, and automation systems. Numerical integration methods and error accumulation mechanisms are dealt with. Emphasis is paid to efficient solution of nonlinear systems and sparse matrix equations.

Guidelines for Software Implementation. The chapter deals with topics necessary to consider when implementing the architecture to source code, thus maintaining efficient use of computer cache memory, advanced fetching of data, as well as running parallel threads and processes. Of concern are the specification and computation data structures as well as the on-line connection support. Recommendations on items identified that require further elaboration are presented, both related to detailed scientific issues and to general standardisation efforts.

Summary and Discussion. The benefit of the unified architecture is dealt with. One of the goals, to allow also a non-specialist to specify models seems to be much closer. Tolerating rough spatial discretisation of the process, simplifies the use of available design data as input. The proposed automated flattening down of the hierarchical model specification, and the evaluation, tearing up and optimised recombination of the graph elements speeds up the calculation and provides for increased robustness. The functionality of basic algorithmic principles has been demonstrated by the implementations made. The new unified architecture is considered to have all possibilities to form a basis for open source modelling and simulation software platforms for future virtual factory initiatives.

The author has found it necessary to refer to 154 separate publications: Reports on previous or parallel original research made by others, experiences from present implementations of the companion model approach, information on application model developments, and results from relevant simulation studies. It shall be noted that the author has contributed to 49 of the application oriented publications, either as the main writer or in close cooperation with experts in the field.

The specification and solution architecture related chapters are complemented with two appendices focusing on the relevant computer memory data organisation:

Specification Data View. The data organisation and representation in the *Modelling Engine* is discussed. It needs to support a hierarchic architecture enabling the specification of relevant graph elements, thus supporting easy configuration of the on-line solution database structure.

Solution Data View. The emphasis is to construct a most compact and efficient run time *Simulation Engine*, easy to embed in control systems and training simulators. The data organisation needs to support vectorised calculation of the coefficient equations, as well as scalar processing of code arising from solution of sparse matrices.

2. Implementations and Applications

2.1 Fast Solution of Linear Equation Systems

An important driving force for the development of real-time simulation tools for large scale processes was the emerging trend in beginning of the eighties to require full-scope replica simulators, including a copy of the real control room, for the training of nuclear power plant operators. At that time, research efforts were for instance assigned to the development fast solution methods applicable to large sets of linear equations. Encouraging experiences of computer solution of large sparse positive definite systems were published (George & Liu 1981).

Influenced by these experiences the author developed the sparse matrix solver implementation described in Subsections 4.6.2. The construction of a sparse matrix into a linked list storage from relevant tabulated versatile companion model parameters as well as the employed pre-optimisation of the solution order, enabled a fast factorisation and solution without any need for pivoting.

An execution time comparison of linear equation system solvers was made by the author (Juslin 1983a), including direct and iterative solvers as well as sparse and full matrix solvers. The solution speed improvements achieved by the author and his colleagues were reported in the IMACS Transactions on Scientific Computation (Juslin et al. 1985). Further experiences on real-time solution of sparse matrices were published in the Research Notes 615 of VTT (Juslin & Silvennoinen 1986).

2.2 Initial Thermal Hydraulic Implementations

The solution of pipe network dynamics consumed a considerable amount of the available computation power in the early industrial plant analyser codes. The calculation of a short transient could take several days on a mainframe computer. The iterative thermal hydraulic solvers then available had developed from steady-state codes and were very time consuming.

In the late seventies the staggered grid method was given publicity (Patankar 1980). It was applied for spatial discretisation of the pipe networks to be analysed. The author found that this method as described in detail in Subsection 3.6.2 well resembled linear graphs for the pipe network description. The author had previously used traditional companion models for simulation of electrical networks. However, to allow for the required changes in causality, e.g. related to reversed flow during the simulation, the author needed to develop the specific versatile companion model as described in Subsection 3.4.2.

A comparison of solution methods for pipe network analysis was contributed (Juslin & Siikonen 1983). The full-scope replica training simulator for the Loviisa nuclear power plant in Finland was under construction and the available thermal hydraulic solvers needed improvements. The implementation of the companion model and the related sparse matrix solver was made by the author in cooperation with personnel from Nokia Electronics Oy, the simulator supplier. This was possibly the earliest large scale commercial implementation of the versatile companion model for solution of thermal hydraulic networks. For competitive reasons, the fundamentals of the implemented methods were, however, then not published.

Other simulator installations were made during the following years making use of this solver implementation such as the full-scope training simulator of the Pacs nuclear power plant in Hungary and the NORS research simulator at HAMMLAB, the OECD supported man machine research laboratory at Institutt for Energiteknikk (IFE) in Halden, Norway. The first solver implementation was initially installed on PDP-11/70 computers and later on transported to VAX computers of Digital Equipment Corporation (DEC).

2.3 Parallel Implementation Efforts

One of the earliest applications of traditional companion models to digital computer simulation of power electronic circuits was contributed by the author (Juslin 1983b). An application of companion models to the dynamic simulation of load flow in electrical power networks was completed by the author (Juslin 1984b). Here the traditional companion model and the sparse matrix solver were implemented applying complex number arithmetics. The author has with interest

noted recent application of traditional companion models in the field of power electronics (Solodovnik et al. 2003).

Experiences on applications for radiator network calculation were at an early stage discussed by the author with co-workers (Laitinen et al. 1986). A first use case on fast dynamic simulation of district heating networks was concluded (Juslin et al. 1987a).

2.4 Fast Material Properties Calculation

Real-time simulation of extensive thermal hydraulic networks called for the development of exceptionally fast calculation methods of material properties. Experiences on fast material property calculation of water and steam were reported in the Research Notes 807 of VTT (Lilja & Juslin 1987). The properties and relevant derivatives were tabulated using an unequal division storage scheme to save computer memory, which was very expensive at that time. More frequent tabulation was used in regions with larger changes in the property values to ensure the necessary correctness. The tabulation of pre-calculated derivatives enabled fast calculation of continuous derivatives in addition to the relevant property values between the tabulated points. Despite of the unequal division of tabulated values the correct indices for the calculation were found by indirect addressing. No search was needed. The complete range of the VDI steam tables was covered. The above developments also contributed to speed up the calculation of the Loviisa full-scope nuclear power plant training simulator. Material property functions are dealt with in Subsection 3.7.3.

The necessity to have access to material properties of multi-component mixtures was recognized. Experiences with iterative flash calculation methods were made in an EUREKA project called CHEDYN in co-operation with Belsim s.a. and University of Liege (Juslin et al. 1990, Juslin et al. 1991). This approach was functional but very slow as experienced from a multicolumn distillation plant application of dynamic simulation (Bärman et al. 1993). An approach to fast calculation of thermal-hydraulic properties by neural networks was reported (Lilja & Hämäläinen 1999). A software tool is currently under development at VTT, expected to automatically teach a neural network with data originating

from measurements or an iterative solver. This opens new horizons for rapid dynamic analysis of chemical process industry plants.

2.5 The APROS Platform Development

It is not only sufficient to have fast calculation. In addition, the modelling shall be efficient and not require programming skills. The author has been instrumental in the specification work and directing the development of the Advanced Process Simulator (APROS) software. The basic APROS software platform was developed within the framework of the VTT research programme on Numerical Simulation of Processes 1986–1988 (Mattila 1987) as a joint effort of VTT and Fortum. The applied solution of one-dimensional flow and heat transfer processes was reported (Hänninen 1988). The APROS software environment for process simulation and model development was documented (Silvennoinen et al. 1989). The multidimensional flow calculation developed was dealt with (Niemi et al. 1989) as well as efficiency of parallel and vector implementations (Niemi & Tommiska 1990). The application of the APROS Specification Language (ASL) for specification of industrial processes (Juslin 1990), and the use of GRINAP, the UNIX based Graphics Interface for APROS (Juslin et al. 1993) were reported. A Windows based Graphics Design interface GRADES developed by Process Vision Oy has been taken into use, as well.

The experienced transportability of the APROS simulation engine programmed in FORTRAN is remarkable. It has been running on VAX mainframe computers, CRAY supercomputers, ALLIANT mini-supercomputers, many UNIX workstations, and now on ordinary personal computers with Windows XP operating system. A full-scope nuclear power plant simulator including control system models is now run on a laptop computer in real-time. The hardware cost is not an issue for taking up simulation as a working method.

2.6 International Dissemination

The advent of the APROS software platform has been presented for an extensive international forum, with a special emphasis on its functionality for analysis purposes and its nuclear applications: IAEA Specialists' meeting on training

simulators for nuclear power plants (Juslin et al. 1987b), European Congress on Simulation (Juslin 1987), International Nuclear Simulation Symposium (Hänninen et al. 1987a), American Nuclear Society's winter meeting (Hänninen et al. 1987b), Nuclear Europe 1987:11/12 (Mattila & Winter 1987), Numerical Heat Transfer 12(1987)1 (Siikonen 1987), European Simulation Multi-conference (Silvennoinen et al. 1988), EPRI Conference on power plant simulation and modelling (Porkholm et al. 1988), IMACS World Congress on scientific computation (Juslin et al. 1988), ENS/ANS conference on thermal reactor safety (Tiihonen et al. 1988), Topical meeting on advances in nuclear engineering computation and advanced radiation shielding (Puska et al. 1989), Scandinavian Simulation Society annual conference (Puska & Juslin 1989), and Beijing international simulation conference (Juslin et al. 1989).

Performance improvements made and customer experiences in different application domains were published as well during the following years. However, the in-depth utilisation of the versatile companion model and related sparse matrix concept to thermal-hydraulic circuits was intentionally not brought to public. Nevertheless, some endeavours to build platforms for the same purpose could be noted.

2.7 Combustion Power Plant Applications

The first applications on simulation of conventional coal-fired power plants and their control systems making use of APROS software were reported (Hänninen et al. 1988; Juslin & Kurki 1989; Kurki et al. 1989). An engineering simulator for a peat-fired power plant was developed (Leppäkoski et al. 1990). A qualitative 3-D estimator of burning process was discussed (Juslin & Lilja 1991). A generic training simulator for a Combined Cycle Gas Turbine plant was presented (Ollikainen et al. 2002).

APROS Combustion is now a very mature product, presently in use by many power plant manufacturers and utility companies. The multifunctional features have also been exploited extensively by control system suppliers and added value developers. The use in process design evaluation, automation system optimisation, and development of training simulators for their customers have been described (Herzog et al. 1994; Douzdouzani et al. 2003). Modern control

solutions and relevant training simulators for gas & steam turbines, combined cycles, desalinations and thermal power plants have been developed (ABB 2005a).

2.8 Nuclear Power Plant Applications

The extensive use of dynamic simulation in nuclear power plant design was strongly emphasised (Laukia et al. 1990). Experiences on the development and first test applications of the plant analyser for the Loviisa NPP were given large publicity (Puska & Porkholm 1991; Kantee et al. 1991; Tiihonen et al. 1991; Porkholm et al. 1991). The one- and three-dimensional PWR nuclear core models were presented (Puska 1991). The Kola nuclear power plant analyzer design project was reported (Porkholm et al. 1994). Among other interesting transient studies, an application of APROS to analysis of a small break loss of coolant accident was published (Al-Falahi et al. 1995). A detailed report on nuclear core modelling in multifunctional simulators was published (Puska 1999). A 3-D core model for a BWR plant analyser was presented (Puska & Norrman 1999). The construction a full-scope replica nuclear power plant training-simulator situated in Chasma, Pakistan was reported (Shah 2002).

A full-scope model of the Forsmark 3 BWR nuclear power plant was developed (Karlsson et al. 2001). This HAMBO model is now in use by the OECD Halden Reactor Project for development of new control room concepts. The extent of the HAMBO simulator supplied with a three dimensional real-time nuclear reactor model is presented in Table 2.1.

Table 2.1. Extent of the HAMBO simulator.

Modelled Items	Number
Thermal hydraulic control volumes	2623
Heat structure elements	6058
Turbine sections, valves, pumps, and fans	1850
Electrical system bus bars	410
Control system signals	65170

Topical applications of *APROS Nuclear* focus on independent process design analysis as well as pre-testing of digital control system configurations. Both control system and process upgrades in existing plants and pre-analysis of new plants are considered.

2.9 Pulp and Paper Mill Applications

The development of unit operations models for pulp and paper production plants was proceeding systematically. The first relevant application was a recovery boiler plant model (Juslin & Tuuri 1992). Next application was a multi-effect black liquor evaporation plant model (Juslin & Niemenmaa 1994). A displacement pulping process model was introduced (Juslin & Pollari 1994). Experiences on detailed dynamic simulation of paper machines were published (Tuuri et al. 1995b). The modelling of a rotary limekiln was reported (Karhela et al. 1998b). Modelling and simulation of a bleach plant was concluded (Lappalainen et al. 1999; Tervola et al. 1999). Model based enhancements of the grade change at a board mill were discussed (Lappalainen et al. 2001). A dynamic model of a disk filter was given publicity (Savolainen et al. 2001). A model for oxygen delignification of craft pulp was presented (Pigg et al. 2002).

Further developments of pulp boiler models have been made in recent projects. Still, some unit operations of the caustisizing plant are in the queue for development into the APROS Paper unit operation library.

2.10 Control System Evaluation and Testing

Gained experience on exercises related to simulator based early verification of plant and automation concepts have been brought to public (Leppäkoski et al. 1991; Tuuri et al. 1995a; Leppäkoski 1995; Soutolahti et al. 1995; Bärman et al. 1995). The introduction of standard interfaces had a large impact on the developments. The first connection of APROS simulation engine to a real DCS system using OPC standards was reported (Karhela et al. 1999). Simulation aided process automation testing principles were dealt with (Rinta-Valkama et al. 2000). The integration of process and automation design was discussed (Paljakka et al. 2000). A performance evaluation of the implemented OPC-based

I/O of APROS was concluded (Peltoniemi et al. 2001). How a customised dynamic simulator optimally could support process and control engineering at mill site was reported (Tuuri et al. 2001). A self-access studying environment for control engineering education was presented (Lilja et al. 2003). A training simulator for turbine and unit controls has been taken into use (ABB 2005b).

The real take up of advance evaluation and testing of control systems is, however, subject to the availability of the required functionalities in the virtual control system installations. Virtual control system installations shall accept exactly the same configuration files as the real control system installation, they shall interface automatically with the simulation platform, accept start and stop commands, provide for change of time scale, and successfully read and write complete snap shots including state variables and historical data. Repeated scenarios initiated from a specific snap shot shall be identical and not e.g. depend on computer system load.

2.11 Internet and Component Based Applications

Applications of web browser and software component technologies to operator user interfaces in process simulation have been reported (Karhela et al. 1998a, Karhela et al. 1999). Specific component frameworks for modelling and simulation have been taken into use (Laakso et al. 1999). An Internet-based equipment data and model gallery has been developed, giving easy access to real component parameters when building up the process flow sheet (Karhela et al. 2001; Nappa et al. 2002). The Gallery query language specification has been published (Kondelin & Karhela 2003).

The development seems to go towards virtual working groups and web-based services of modelling and simulation centres, supporting stake holders engaged in research, design, maintenance, operation, trouble shooting, self-learning or training. International efforts directed to development of the semantic web are supposed to make world wide knowledge bases available in future, including design knowledge and applicable regulations.

2.12 The Software Development Process

Several full-scope training simulators representing the first generation of the versatile companion model implementations are still in daily use. APROS, representing the second generation of implementations, is now established as a commercial software applied to many industrial sectors. It is a valuable computer-aided engineering tool for several hundred active users, worldwide. It is multifunctional in the sense that it supports basic process analysis and development, automation system concept development, testing of real control system functionality, as well as the development and running of full-scope training simulators. It is suited for detailed analysis of selected separate parts of a process as well as for studies of large integrated processes. It can run slower than real-time for studies of fast transients, exactly in real-time in training sessions, or faster than real-time for predictive operational studies. A comprehensive assessment of the APROS simulation software development process has been made (Silvennoinen 1996), concentrating on research and development cooperation and technology transfer as strategic instruments. The basic software platform produced as an outcome of the initial investment in the three-year APROS development project, has been an enabling key element in numerous subsequent application directed research, development, and consultation initiatives, indeed providing for added value development, testing, verification and validation of the software platform and its solver services.

In the beginning, the APROS platform was considered as strategic engineering software, not to be made available at all for competitors. It was soon accepted that a large user group facilitates the required financial resources to maintain the high quality of the software and to support its transportation to new operation system versions and hardware platforms. A key issue has been that the application model repository files, as an outcome of detailed specification work, shall be easy to re-use when releasing new versions of the software. The model specifications can be considered as important carriers of formal design information over the full life-cycle of the target plants. An additional continuous challenge is to keep up with arising new requirements of the software as generated by the user group members. When systematically moving into new application areas, relevant knowledge needs to be gathered, by customers, suppliers and developers in co-operation. In such situations, basic research and development undertakings have arisen, which are very multidisciplinary and

well suited for a research organisation like VTT. New model libraries have then been developed, either for the sole use of the customer, or to be included in the maintenance program and commercially available for all users. Such developments as well as customers' own application developments have been reported in hundreds of international publications.

The new data architecture specification presented in this thesis is a synthesis of ideas arisen from the broad range of application experiences referred to above. The author has developed separate prototypes to initially test some features with regard to the new data architecture. These software prototypes have been made using Object Pascal source code in the Borland's Delphi programming environment (Warner & Goldsman 1996). The prototypes include a graphics interface program for model specifications, and the connection to a simulation engine program. In order to enable the programs to easily communicate with XML scripts (Goldfarb & Prescod 2000) the author made a dedicated XML parser implementation.

In accordance with these tests and all previous experience, the fast and reliable calculation of large integrated industrial processes turns out to require efficient real-time data-base organisation in addition to efficient mathematical algorithms including capable nonlinearity and discontinuity handling, robust discretisation with respect to space and time, vectorisable calculation of the matrix elements, rapid sparse matrix solvers, and fast and reliable material property look-up. The enabling key technology related to the Versatile *Companion Model* is described in the next chapter.

3. Model Specification Architecture

3.1 Layered Model Specification

The layered architecture proposed is supposed to enable efficient model specification and easily conducted dynamic simulation of large integrated industrial processes. Different user categories and tasks have been considered: Engineering users, designers of generic components, programmers of external functions, as well as maintainers of component galleries, model repositories and experiment recordings. Applicable scopes regarding the development of algorithms and software have also been touched upon. The different actors need access to different features. The user's and developer's views in the following subsections should be considered as requirements for the functionality.

3.1.1 Engineering User's View

The *Engineering User* has access to the *Modelling Interface,* the *Simulation Interface,* the *Component Galleries*, the *Model Repositories* and the *Experiment Recordings*. The engineering user builds up the *Graphical Flow Sheet* using the modelling interface. He places out node symbols and connects them with branch symbols. Each symbol has *Terminals* (connection points). A branch symbol has two terminals, one at each end. A node symbol has at least one terminal. The symbols are usually designed to graphically resemble real system *Components*. The node symbols are expandable to indicate the size of the real components and the branch symbols are stretchable to allow for making the connections. Looking at how the component models are constructed, we can distinguish between Assembled *Components* and *Generic Components*:

- **Assembled Component.** A flow sheet depicting an integrated sub-process specified by the engineering user as indicated above, can be assigned an own graphical symbol supplied with distinct published terminals. Model repositories containing previously assembled components provide for easy re-use. Modified components may be stored as new instances.

- **Generic Component.** Automated generation of the descriptive internal flow sheet is applied for commonly used process components. Such a generic component has an own graphical symbol, a specific parameter list and a related script for the generation. Access to supplier specific component parameters over the Internet or from local design databases is essential for effective use.

The flow sheet created by the generic component script can be accessed by the graphical modelling interface tool. It should, however, be noted that if the structure of the flow sheet or the lower level parameters are modified manually, then the generic component becomes an assembled component. The assembled component can not be re-generated based on the generic component parameters.

3.1.2 Generic Component Designer's View

The *Generic Component Designer* specifies the items of the required parameter list and the relevant script for automated generation of the internal flow sheet. He also designs a suitable symbol including required terminals. The flow sheet of a generic component can include simple components as well as other generic components. The building blocks available for the use of the generic component designer are as follows:

- **Simple Terminal.** A simple terminal can only connect to other terminals of a type resembling the local state variable in consideration, such as pressure, specific enthalpy, mass fraction, or voltage.

- **Compound Terminal.** A compound terminal is made up of a suitable set of simple terminals or other compound terminals. A compound terminal for connection of pipes in a steam network usually comprises the simple terminals for pressure and specific enthalpy. The generic component designer can specify suitable compound terminals.

- **Simple Branch.** A simple branch provides for flow or hold up of mass, energy, electric charge or any other single transition entity. All physical mechanisms are included in the branches. The simple branch is the graphics view of the companion model to be described later on in very

detail. A simple branch has two published simple terminals, one at each end.

- **Compound Branch.** A compound branch is a generic or assembled component that can include several simple branches or other compound branches in parallel. A compound branch may also include compound branches and nodes in series. At application level it is, however, shown as a single branch component. A compound branch has two published compound terminals, one at each end.

- **Simple Node.** A simple node represents a single property variable to be taken into account such as pressure, enthalpy, concentration, or voltage. A simple node does not itself include any physical mechanisms. It only connects simple branches. A simple node has at least one published simple terminal.

- **Compound Node.** A compound node is a generic or assembled component that represents a collection of local simple or compound nodes. A compound node can also include simple local branches depicting e.g. reactions between substances involved. A compound node has at least one published simple or compound terminal.

3.1.3 Simple Branch Configurator's View

The *Simple Branch* is the core component in the sense that it is the target for the specification of all real world local interactions that should be considered. The *Simple Branch Configurator* selects the *Separate Mechanisms* that are supposed to have impact on e.g. the flow or hold-up in a specific simple branch. He can also choose required *Material Properties* and *Customl Functions*.

- **Separate Mechanisms.** Application domain specific libraries of readily implemented algorithms for calculation of distinct elementary physical mechanisms need to be available. An interface is needed for externally programmed separate mechanisms.

- **Material Properties.** Lookup of basic material properties by using pre-calculated and tabulated parameter sets or by access to specific external material property calculation software.

- **Custom Functions.** Custom functions are typically input/output models, such as automation system function blocks. They can include internal state variables. The supplier of these external function blocks is responsible for their performance.

3.1.4 Repository Maintainer's View

The process component manufacturers and process integration designers can maintain themselves libraries of relevant component specifications and process design concepts. However, they can also outsource these tasks to a specific *Modelling and Simulation Service Centre.* The models can be operated over the Internet for experimentation or training purposes.

- **Component Galleries.** Process component manufacturers have the best knowledge to supply model parameter specifications of their components to be available for engineering users over Intranet, Extranet or Internet.

- **Model Repositories.** Integrated plant models can be specified by authorised engineering users into distinct model repositories with dedicated access rights for relevant development teams, project participants, or user groups.

- **Experiment Recordings.** Corresponding recordings of simulation experiments as well as measurements from real plants can be stored into experiment repositories, thus being available for model validation or process diagnosing purposes.

3.1.5 Software Developer's View

The *End User* has the possibility to develop own source code for new *Separate Mechanisms*, *Material Properties*, and *Custom Functions*. The *Dynamically Linked Library* (DLL) standard interface provides for easy implementation of such external functions without the need for compiling and linking of the modelling and simulation engines. The expert of physical mechanisms has access to the distinct specification layers of the versatile companion model. The expert of material properties can include customised material properties calculation functions or empiric correlations subject that they are made up of concatenating *Monotonic Regions*. The programmer of the external functions can decide if the functions need to be included in the iterative nonlinearity correction procedure. The external functions can be algebraic or dynamic, as well as linear or nonlinear.

The *Platform Developer* is responsible for the easy transporting of the modelling and simulation platforms and the relevant graphics software to new operation system and computer hardware environments, whence decided necessary. User's guides and installation instructions need continuous maintenance, as well. The quality assurance requires traceability of changes in the source code, and application of required verification and validation procedures. The accumulated amount of application developments made shall easily survive any updating of the platform. The platform developer coordinates the software development work.

The *Modelling Engine Developer* is responsible for the development of the hierarchical model specification formalism, the access to evolving semantic databases, the automated flattening down of the *Hierarchical Model Specifications* to a *Linear System Graph Specification* and the preparation of the data structure for the models to be run on one or several simulation engines in parallel. Concepts such as *Local Regions*, *Homogeneous Zones*, *Function Blocks*, and *Implicit Islands* are identified. They are basic concepts related to the real-time database organisation optimised for the use of the table driven solvers of the simulation engine.

- **Local Regions.** Suitable parts of a larger model graph are separated into local regions and if required run in parallel on multiple simulation

engines. Each simulation engine provides for solving the included homogeneous zones, in conformance with the arising requirements.

- **Homogeneous Zones.** Specific zones of the graph with very weak or very strong internal dependencies are separated to own zones and calculated in parallel with the remaining normal zone. Different time-steps can be applied to each homogeneous zone.

- **Function Blocks.** Function blocks have specific input, output and possible internal state variables. Explicit, semi-implicit and implicit function blocks are identified within each homogeneous zone. Typical explicit or semi-implicit function blocks are those that originate from sequential control system operations or from calculation of companion model parameters. External functions are attached as DLLs and also treated as function blocks.

- **Implicit Islands.** An Implicit island is a specific kind of function block. An implicit island is made up from implicitly connected companion models. The relevant matrix equation is solved by sparse matrix methods.

The *Simulation Engine Developer* is responsible for the efficient implementation of the relevant data structures and solvers. He also needs to consider the real-time communication issues and the interface to the simulation graphics. He is also responsible for generation of run-time models.

Separate developers can be assigned for the *Modelling Graphics,* the *Simulation Graphics*, the *Component Gallery*, the *Model Repositories* and the *Experiment Recordings*. There shall be provision for both browser based graphics clients and the use of independent proprietary graphical tools. The graphical interfaces shall be very thin and just include information needed for opened windows. For security reasons also decoding, encoding and authentication procedures shall be provided for.

The concepts needed to handle the nonlinear differential and algebraic equations of a supposedly *Piecewise Monotonic World* are described in detail in the following chapters. Starting from linear structured graphs made up from simple

nodes and branches, specifications are made layer by layer to complete with a hierarchical plant wide model description. The real world equations are formulated to comply with the notation of *Mechanistic Transition Equations*, they are linearised to *Linear Transition Equations* in order to enable implicit solution and finally temporally discretised to comply with the *Versatile Companion Models* related to the simple branches of the system graph.

3.2 Structured Graphs

In systems modelling, research has been focused on three graph-based paradigms: block diagrams, bond graphs, and linear graphs (Sinha et al. 2001). Control systems comprise control elements supplied with input and output signal connections and are intrinsically built up as block diagrams. Accordingly, it is straightforward to model them as well as other typical function blocks with block diagram graphs. Bond graphs require that the causality is specified at the time of model creation, whereas linear graphs easily can cope with causality changes such as the change of fluid flow directions. Another reason to choose linear graphs is that they reflect the process system topology directly unlike the bond graph or block diagram models. The relationship between physical systems and linear graphs was already recognized about 50 years ago (Trent 1955; Branin 1966). The author's choice for the developments was to apply linear graphs and block diagrams combined.

A *Graph* is made up *Nodes* (vertices) and *Branches* (edges). In a *Connected Graph* each branch is connected at both its ends to nodes and there is at least one path from a node to any other node in the graph. The most simple connected graph is accordingly made up of two nodes connected by a branch. A *Directed Graph*, is a connected graph where each branch has a specified direction of dependency. A *Linear Graph* is a directed graph where there is a linear equation for each branch expressing the relation between the involved graph variables.

A *Simple Branch* has only one *Transition Variable* for instance representing mass flow or current. A *Simple Branch* has a specified positive flow direction. The simple branch has one *Simple Branch Terminal* at each of its ends. A *Simple Node* has only one *Local Variable,* for instance representing pressure or voltage. A simple node has any number of *Simple Node Terminls*. A simple branch

terminal and a simple node terminal can only be connected if they represent the same *Domain,* which relates to the type of local variable in consideration. The terminals of a simple branch can be of different domain. The *Degree* of a simple node is the number of simple branches connected to that node. A *Simple Branch* can be depicted by relevant *Branch Elements* in parallel between the branch terminals representing the contribution of the distinct terms of the linear equation making up the transition variable.

Graphs are applicable at very detailed level for description of real world performance, but also at large. A *Grouped Graph* contains simple branches and nodes or other grouped graphs. A grouped graph can consist of separate connected graphs. For higher abstraction level use, the internals of a grouped graph are hidden and the grouped graph can be depicted by a suitable *Symbol*. The grouped graph can only be connected via published terminals. A grouped graph describes one or several unit operations or even a whole production unit. Grouping of graphs to form higher level graphs enables temporary hiding of too detailed information. It therefore essentially simplifies the use of the graphs. The *Compound Nodes* and *Compound Branches* as specified in Subsection 3.1.2 are grouped graphs.

All grouped graphs in this architecture are eventually structured out the needs of the process design of process components, unit operations, subprocesses or even a complete plant. Because they all have been built up by grouping of lower level graphs it is possible to flatten down the whole model to a consistent set of simple branches and nodes of a *System Graph* to form a basis for the mathematical solution of their integrated functioning.

The availability of this kind of layered graphical description is expected to lower the threshold to take up and use modelling and simulation for studies of dynamic performance of large integrated systems. Depending on the expectations of a specific simulation study there might be a need of different levels of detail and viewpoints of the models. Sometimes it is sufficient only to know the production time and capacity of a chemical plant. In some cases, however, even the detailed chemical kinetics in each reactor needs to be considered.

In the following, the study is concentrated to process industry applications. The level of detail of the models is restricted to make real-time studies of large

integrated processes possible and feasible. The lowest level of graphs or dependencies considered, hence relates to the dependencies set up by the first physical and chemical principles. These are complemented when necessary by material properties and component specific empiric correlations.

3.3 Mechanistic Transition Equation

The different actors and views as discussed above in Section 3.1, impose diverse requirements on the architecture of the modelling and simulation platform. The specification of the geometric structure of the plant and its components, the physical, chemical and empiric behaviours involved require all to be carefully considered when designing the model specification database, the real-time calculation database, and the mathematical solver.

The time constants encountered in an industrial process can differ with several orders of magnitude. Therefore, the scope of study needs to be considered carefully. In some context a specific phenomenon is so fast that it is convenient to simulate it as instantaneous algebraic equations, subject that the fast dynamics does not impact on the result. In some other context, a specific phenomenon is so slow that it can be considered as a constant boundary condition during the term of study. In any case, provision should be made to cope with *Stiff Systems*.

The interdependencies of some variables can be so strong that the only way to solve the integral problem is to first combine them for a joint solution, and thereafter calculate the details. Some interdependencies can be so loose, that they can be totally omitted. As we shall later on find out, the coefficients of the versatile companion model contain information for evaluation of such inter-dependencies. This information will, however, be destroyed upon formation of the matrix equation elements required for the integrated solution. Some interdependencies are almost linear, others are nonlinear but still monotonic, and some involve even discontinuous transitions between separate continuous regions. The modelling and simulation platform needs according to these prerequisites a solver prepared for all the situations arising when describing the real word.

Many of the effective phenomena are related to the well-known conservation equations for mass, energy and momentum in combination with empiric correlations for thermal dynamic, transportation and other properties. Sometimes important product quality properties depend on the complete processing history. Sometimes only vague or statistical relationships can be given. Provision shall be made for handling of relevant multi-paradigm or hybrid models.

A real process is usually by its construction divided into suitable control volumes, for each of which homogeneous distribution of relevant properties can be anticipated. If considered necessary for the studies, large tanks or long pipes can be divided into smaller volumes, by the user or automatically by the simulation software. In such a way the model is discretised with respect to space into control volumes.

Each component of a dynamic system model can be considered as representing a relationship between two *Local* property variables x_i and x_j and the relevant *Transition* flow variable v_{ij} (MacFarlane 1964). Accordingly, the dependency between the terminal pressures of a pipe section in consideration and the relevant transition flow could be given as an implicit function,

$$g_{ij}(x_i, x_j, v_{ij}, dv_{ij}/dt, \smallint v_{ij}dt, t) = 0.$$

This *Mechanistic Transition Equation (MTE)* depicts one or several separate physical mechanisms having impact on the transition flow v_{ij} between the local property variables x_i and x_j. The MTE is either *Linear* or *Nonlinear*. The MTE is either *Static* or *Dynamic*. The MTE can include terms dependent on time t, on acceleration dv_{ij}/dt and on accumulation $\smallint v_{ij}dt$ effects. The MTE is assumed to be *Piecewise Monotonic*. The MTE can as well include parameters that explicitly depend on other external variables.

If there is no accumulation of the transition variables in an infinitesimal terminal point we can assume that the sum of incoming and outgoing flows is zero. For all the flows v_{ij} connected to terminal j we can thus write $\Sigma_i\, v_{ij} = 0$. Note that there are usually only a few flows connected to a single terminal. All the equations $g_{ij} = 0$ and $\Sigma_i\, v_{ij} = 0$ required to describe the model of concern together form a system of differential and algebraic equations. Note as well that some of the equations can be nonlinear. If N_x is the number of dependent local

variables x_i and N_v the number of dependent transition variables v_{ij} then the number of variables to be solved is $N=N_x+N_v$. In addition, the equations can include external variables x_i or v_{ij} acting as boundary conditions to the system. We describe this system of N equations in N unknowns as $G^N = 0$, where $G^N : R^N \rightarrow R^N$. The assumption is that all such interdependencies of a system, at least for a very short time instance, can be considered as linear an depicted by a *System Graph* consisting of simple branches and nodes.

Several simulation and modelling paradigms and languages have been developed. They can be classified according to the following criteria (Fishwick 1998): graph-based versus language-based paradigms, procedural versus declarative models, multi-domain versus single-domain models, continuous versus discrete models, and functional versus object-oriented paradigms. Graphs have been used to represent interconnected systems in many modelling domains. We shall now focus on the linear graph concept extended to represent the versatile companion model and the relevant equivalent models.

3.4 Versatile Companion Model (VCM)

3.4.1 Origin of the Companion Model

The fundamentals of *Linear Circuit Theory* originate from Georg Simon Ohm's (1789–1853) law published 1827, Gustav Robert Kirchoff's (1824–1887) laws known from 1840s, the principle of superposition proclaimed 1853 by Herrman von Helmholtz (1821–1894), and James Clerk Maxwell's (1831–1897) ideas composed 1864 in his publication "A dynamic theory of electromagnetic field". From this starting point, any linear circuit can be solved: Given a specification of all sources and impedances in the circuit, a set of linear equations can be found and solved to yield any voltage and current in the circuit.

One of the most surprising concepts to arise from linear circuit theory is the *Equivalent Circuit*: no matter how complex the circuit, from the viewpoint of any pair of terminals, the circuit behaves as if it consisted only of a source and an impedance. Two equivalent circuit structures predominate: the voltage-source equivalent circuit and the current-source equivalent circuit. A review of the origin of the equivalent circuit concept has recently been made (Johnson 2003a

& 2003b). The concept of voltage-source equivalent was published by Hermann von Helmholtz in 1853. Thirty years later in 1883, Léon Charles Thévenin (1857–1926) published the same result apparently unaware of Helmholtz's work. The generality of the equivalent source network was not appreciated until forty-three years later. Then, in 1926, Edward Lawry Norton (1898–1983) wrote an internal Bell Laboratory technical report that described the usefulness in some applications of using the current-source form of the equivalent circuit. In that same year, Hans Ferdinand Mayer (1895–1980) published the same result and detailed it fully.

The current-source equivalent circuit was considered as a dual model or *Companion Model* to the voltage-source equivalent circuit. The current-source companion model can be attained from the voltage-source representation of a network branch according to Ohm's law, $u_{br} = z_{br}\,i_{br} + e_{br}$, by solving for the branch current i_{br} and replacing the branch voltage u_b by relevant node voltages $u_{br} = u_i - u_j$, the branch impedance z_{br} by the relevant branch admittance $y_{br} = z_{br}^{-1}$ and the voltage-source e_{br} by the relevant current-source $j_{br} = e_{br}\,z_{br}^{-1}$ to get $i_{br} = (u_i - u_j)\,y_{br} + j_{br}$. Whence the current-source companion models for all interconnected branches in a circuit under study are specified, and at least one branch is connected to an external reference node, all the internal voltages and the relevant branch currents can be solved from the arising linear equation system as is described in detail in Subsection 4.6.1. The introduction of the current-source companion model made performance calculations of linear electrical circuits very straightforward, in fact, already by using pencil and paper.

3.4.2 Versatile Companion Model Equation and Branch

The main emphasis in this section is to introduce a generalisation of the companion model concept for a broader use than just linear electrical circuits. The intention has been to formulate a powerful but yet simple representation allowing for easy specification and solution of large and complicated dynamic systems including components of various domains. An important requirement has been to also encompass such nonsymmetric and nonlinear dependencies that are typical for thermal dynamic and chemical processes. It is supposed that the dependency, at least for a short time instance, between two local variables and a

relevant transition variable can be described by the *Versatile Companion Model (VCM)* equation,

$$x_i g_f - x_j g_b + s_{ij} - v_{ij} = 0$$

The VCM equation describes the linear declarative dependency between the *Transition Variable* v_{ij} from node i to node j, the *Local Variables* x_i and x_j, the *Forward* and *Backward Coefficients* g_f and g_b, and the *Source Term* s_{ij}. The coefficients and the source term together form the three *VCM Parameters*. The related *VCM Branch* shown in Figure 3.1 is according to previous specifications a *Simple Branch* as it connects two local variables by a single transition variable. It has, however, an internal structure comprising at most three parallel *Branch Elements*: *Forward Element* $x_i g_f$, *Backward Element* $x_j g_b$, and *Source Element* s_{ij}. They are connected by joint terminals to the two adjacent nodes x_i and x_j. The VCM parameters are usually not constant. They typically change for each time-step or iteration.
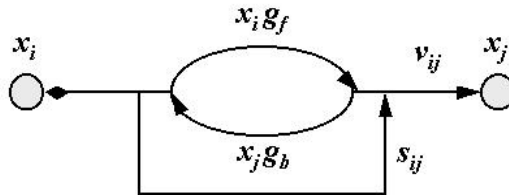
Figure 3.1. The versatile companion model branch.

3.4.3 VCM Provides Alternative Causality Modes

Structured graphs define easier the causality of real processes than modelling languages or equation based approaches (Paredis et al. 2001). In a conventional companion model the coefficients $g_f = g_b$ and the related branch is considered as *A-Causal*. VCM branches arising from linear electrical circuits with independent voltage sources are a-causal. Relevant heat diffusion branches are also a-causal. Additional causality modes are specified for the versatile companion model. If both coefficients g_f and g_b exist but $g_f \neq g_b$ then the branch is *Bicausal*.

Chemical reaction rates are typically bicausal. If only the other of the coefficients exists, the branch is *Unicausal*. Thermal convection equations are typically unicausal. Upon reversed convection flow a *Forward-Cause* branch changes to a *Backward-Cause* branch. If both coefficients are zero and only the source term is left, the branch causality is considered as *Forced*. A piston pump generates a forced flow of non-compressible media within a certain range of operation. If also the source term is zero then the branch *Disconnected*. If the coefficients in an a-causal branch are relatively very large then the branch and its adjacent nodes are combined to form a *Lumped* node. The novelty with the versatile companion model is that it has alternative causality modes and that causality changes during the simulation can be inherently considered.

3.4.4 VCM Connects to Suitable Variable Instances

An other unique feature of the versatile companion model is that it also has access to alternative instances of the local and transition variables. They inherently provide for on-line flexibility to choose from explicit, implicit and semi-implicit solution methods. The following instances of the *Local Variable* are available: $x_{i\hat{e}}$, x_{ie}, x_{ik}, x_{it}, x_{iu} and x_{iv}. The first letter of the sub-index stands for the node index. The following letter of the sub-index denotes the relevant local variable instance in the numerical solution scheme. It is marked by $\hat{e}$ for a new value to be predicted, by e to be implicitly calculated, by k for the old iteration step value in the correction of nonlinearities, by t for the last situation accepted as a correct solution, by u for the previous time-step value, and by v for the two time-steps old value.

The following instances of the *Transition Variables* are available: $v_{ij\hat{e}}$, v_{ije}, v_{ijk}, v_{ijt}, v_{iju} and v_{ijv}. The positive direction of a branch is shown by its sub-indexes. It starts from the instance of the node value as denoted by the first sub-index and stops at the instance of the node value as denoted by the second sub-index. The following letters refer to the branch instance in the numerical solution scheme, in the same way as for the node variables. A branch accumulates mass, energy, momentum, electrical charge, or quality properties.

3.4.5 VCM Supports Procedural and Declarative Modelling

It shall be noted that the node and branch structure of the versatile companion model supports both declarative and procedural modelling. Any local or transition variable can be implicitly or explicitly solved, or considered as a boundary condition. A pipe element in a flow network requires declarative specification to allow the flow to turn into any direction. The pipe junctions are thus only connection points not anticipating any flow direction. On the other hand, control system components involving dedicated inputs and outputs are intrinsically procedural. In a *Procedural Modelling* scheme explicit input variables x and output variables y are specified and the relevant equations or functions $y = f(x)$ are usually written in a certain sequence (Muetzelfield 2004). A *Declarative Modelling* scheme, however, promotes the use of implicit dependencies such as $f(x,y) = 0$ which allows for specification of the dependencies in any order.

3.4.6 VCM Suits Application to Multiple Domains

Typical local and transition variables applied in the formulation of the transition equations and relevant versatile companion models representing different physical *Domains* are described in Table 3.1.

Table 3.1. Domain specific local and transition variables.

Domains		Transition v_{ij}	Local x_i	Local x_j
1	*Electrical circuits*	electrical current	voltage	voltage
2	*Hydraulic circuits*	mass flow	pressure	pressure
3	*Convection*	energy flow	specific enthalpy	specific enthalpy
4	*Heat structures*	energy flow	temperature	temperature
5	*Fluid to wall*	energy flow	specific enthalpy	temperature
6	*Concentration*	substance flow	mass fraction	mass fraction
7	*Chemical reaction*	production rate	mass fraction	mass fraction
8	*Rotating mass*	shaft power	rotation speed	rotation speed
9	*Electrical power*	complex current	complex voltage	complex voltage
10	*Control systems*	not applicable	control signal	control signal

Electrical networks are listed as distinct domains for time domain studies of direct current circuits and for load flow calculations of alternating current power networks. In hydraulic circuit calculation separate domains are identified for mass flow, energy convection, heat diffusion in structures, and heat transfer from fluid to structures. The transportation of different substances as well as reactions between substances need also to be separately considered. The rotating masses convey power from the hydraulic domain to the electrical power domain. The control systems models have, however, been retained as function blocks to conform with the functionality of real control systems.

In following sections it is shown how equations arising from very different physical domains can be manipulated in such a way that they fit into the linear directed graph presentation of the versatile companion model. The usual steps to be taken are: Spatial discretisation, linearisation of nonlinear terms, temporal discretisation, and finally the algebraic manipulation needed to describe the transition variable as a linear function of the local variables. A graph made up from versatile companion models depicts a linear equation system that easily allows for an integrated solution. In the following section the *Linear Dynamic Equivalent* of the versatile companion model is described.

3.5 Linear Dynamic Equivalent of VCM

The linear dynamic equivalent model is specified to extend the versatile companion model concept with such static and dynamic equations that for instance describe linear electrical circuits including elements like linear inductors, capacitors, resistors and voltage sources. According the to accumulated experience from previously made implementations, a small basic set of only five differential and algebraic equations is not only sufficient for the description of above mentioned linear electrical circuit elements but also adequate to describe relevant instant dependencies in other domains such as thermal hydraulic circuits subject to application of suitable linearisation procedures as described in Section 3.6. We shall restrict the study to first order differential equations. It shall be noted that higher order differential equations easily can be replaced by a set of lower order equations if suitable new variables are specified.

3.5.1 Linear Transition Equations

The required basic set of relevant *Linear Transition Equations* (*LTE)* involving two local variables x_i and x_j connected by a transition variable v_{ij}, has been collected into Table 3.2. The coefficients included in the relevant algebraic and first order differential terms of the equations are called the *LTE Parameters*: R_f, R_b, L, C, E and I.

Table 3.2. Linear transition equations.

LTE	Linear transition equations connecting v_{ij}, x_i and x_j	
1	$R_f^{-1} x_i - R_b^{-1} x_j + I - v_{ij} = 0$	*Bicausal static dependency*
2	$x_i - x_j - R\,v_{ij} + E = 0$	*A-causal static dependency*
3	$x_i - x_j - C^{-1}\!\int v_{ij}\,dt + E = 0$	*A-causal dynamic dependencies*
4	$x_i - x_j - L\,dv_{ij}/dt + E = 0$	
5	$x_i - x_j - R\,v_{ij} - L\,dv_{ij}/dt + E = 0$	

Detailed example derivations of such LTE models for each physical domain of interest will follow in the later part of this chapter. In order to build the solution structure layer by layer, we shall, however, first consider how to get versatile companion models from this basic set of LTE models.

3.5.2 Versatile Companion Models from LTE Models

In a professional simulation tool for industrial processes used by design engineers, not mathematicians, the general approach is that the numerical integration method is automatically chosen as stable as possible. Linear multi-step methods, both explicit and implicit, are based on polynomial approximations in the time domain supposing that the differential equations are linear including coefficients that are constant with respect to time. The differential equations describing thermal-hydraulic networks are very nonlinear and stiff. According to the conclusions made in Section 4.3 the choice of one-step implicit methods is quite obvious. In addition, one-step methods enable easy adaptation of the time-step to cope with largely varying change-rate and spuriously arriving discontinuities. The coefficients of implicit integration

methods of an order not exceeding 2 can be chosen in such a way that the integration method becomes *Absolute Stable*, i.e. the stability region of the method is at least the whole left-half plane (Broenink 2004). In accordance with the detailed analysis in Section 4.3 three alternative temporal discretisation methods are applied: Backward Euler rule, Trapezoid rule, and Gear's Second Order rule. They are all absolute stable. Backward Euler rule is always used to start up the calculation after the crossing of monotonic region borders or when passing time dependent discontinuities. Upon time-step extensions following a transient or whilst time-step reductions are needed e.g. for finding discontinuity points, the Trapezoid rule is applied. Backward Euler and the Trapezoid rule are truly one-step methods, not requiring calculation of intermediate points, and they are according to author's experience very well suited the companion models. For situations when operating with constant time-steps after that large transients have been passed the Gear's second order method is introduced. It has not been needed in previous implementations. Its drawback is that it requires equidistant time-steps. It is included to show that the companion model approach also suits multi-step methods.

We shall now show how the linear transition equations easily can be discretised with respect to time just by substitution of their relevant *Derivative Terms* dv_{ije}/dt and *Accumulation Terms* $\int_e v_{ij}\, dt$ by applying each of the rules described in detail in Subsection 4.3.1:

- **Backward Euler rule.** The backward Euler rule is solved to give an estimate for the derivative term at time e as well as to estimate the accumulation term during the time-step Δ_e as follows:

$$dv_{ije}/dt \approx \Delta_e^{-1}(v_{ije} - v_{ijt}),\ \text{and}$$

$$\int_e v_{ij} dt \approx \Delta_e\, v_{ije}$$

- **Trapezoid rule.** If the temporal derivative of the transition variable at time t we is denoted v'_{ijt} we can write applying the Trapezoid rule:

$$dv_{ije}/dt \approx 2\,\Delta_e^{-1}(v_{ije} - v_{ijt}) - v'_{ijt},\ \text{and}$$

$$\int_e v_{ij} dt \approx \tfrac{1}{2}\,\Delta_e\,(v_{ije} + v_{ijt}).$$

- **Gear's second order rule.** Applying the Gear's second order rule described in Subsection 4.3.1 we need to have access to the transition variable v_{iju} and the accumulation variable $q_t = q_u + \int_t v_{ij} dt$:

$$dv_{ije}/dt \approx \Delta_e^{-1} \left({}^3\!/_2\, v_{ije} - 2v_{ijt} + {}^1\!/_2\, v_{iju} \right), \text{ and}$$

$$\int_e v_{ij} dt \approx {}^2\!/_3\, \Delta_e\, v_{ije} + {}^1\!/_3\, q_t - {}^1\!/_3\, q_u.$$

Table 3.3 shows the expressions for calculation of branch coefficients g_f and g_b and source values s_{ij} of the relevant VCM equation from the linear transition equation parameters (see Table 3.2) complemented with old state variables when applying the three different rules.

Table 3.3. Companion model parameters from linear transition equation parameters and old state values.

LTE	g_f	g_b	s_{ij}
			Backward Euler rule
1	R_{kf}^{-1}	R_{kb}^{-1}	I_k
2	R_k^{-1}		$g\,E_k$
3	$\Delta_e^{-1}\,C_k$		$g\,E_k$
4	$\Delta_e\,L_k^{-1}$		$g\,E_k + v_{ijt}$
5	$(R_k + \Delta_e^{-1}\,L_k)^{-1}$		$g\,(E_k + \Delta_e^{-1}L_k\,v_{ijt})$
			Trapezoid rule
1	R_{kf}^{-1}	R_{kb}^{-1}	I_k
2	R_k^{-1}		$g\,E_k$
3	$2\,\Delta_e^{-1}C_k$		$g\,E_k - v_{ijt}$
4	${}^1\!/_2\,\Delta_e\,L_k^{-1}$		$g\,E_k + v_{ijt} + {}^1\!/_2\,\Delta_e\,v'_{ijt}$
5	$(R_k + 2\,\Delta_e^{-1}\,L_k)^{-1}$		$g(E_k + 2\Delta_e^{-1}L_k\,v_{ijt} + L_k\,v'_{ijt})$
			Gear's 2nd order rule
1	R_{kf}^{-1}	R_{kb}^{-1}	I_k
2	R_k^{-1}		$g\,E_k$
3	${}^3\!/_2\,\Delta_e^{-1}\,C_k$		$g\,E_k - {}^1\!/_2\,\Delta_e^{-1}\,q_t + {}^1\!/_2\,\Delta_e^{-1}q_u$
4	${}^2\!/_3\,\Delta_e\,L_k^{-1}$		$g\,E_k + {}^4\!/_3\,v_{ijt} - {}^1\!/_3\,v_{iju}$
5	$(R_k + {}^3\!/_2\,\Delta_e^{-1}L_k)^{-1}$		$g\,(E_k - 2\Delta_e^{-1}L_k\,v_{ijt} + {}^1\!/_2\,\Delta_e^{-1}L_k\,v_{iju})$

In order to follow up the derivation of the expressions in Table 3.3 just consider for example the linear transition equation number 3 in Table 3.2 depicting accumulation in a branch,

$$x_i - x_j - C_k^{-1} \int v_{ij}\, dt + E_k = 0.$$

The coefficients originating from previous iteration step value are denoted with sub-index k. We apply for instance the Trapezoid rule and substitute the relevant integral expression $\int_e v_{ij}\, dt \approx {}^1\!/_2\, \Delta_e(\, v_{ije} + v_{ijt})$ in the linear transition equation and we get $x_{ie} - x_{je} - {}^1\!/_2\, C_k^{-1}\, \Delta_e\,(\, v_{ije} + v_{ijt}\,) \; + E_k = 0.$ If we solve this equation with respect to the estimated flow v_{ije} we can write,

$$v_{ije} = x_{ie} 2\Delta_e^{-1} C_k - x_{je} 2\Delta_e^{-1} C_k - v_{ijt} + 2\Delta_e^{-1} C_k E.$$

If we compare this equation with the VCM equation written $v_{ije} = x_{ie} g_f - x_{je} g_b + s_{ij}$ at the time instance e we will find that the coefficients $g = g_f = g_b = 2\, \Delta_e^{-1} C_k$ and the source term $s_{ij} = g\, E_k - v_{ijt}$ as set forth in Table 3.3.

When evaluating different temporal discretisation methods it shall be kept in mind that they need to be efficient enough to suit real time simulation of full scope industrial plants. The same goal shall also be considered when evaluating solution methods for nonlinear equation systems. In the following section the *Nonlinear Mechanistic Equivalent* of the versatile companion model is described.

3.6 Nonlinear Mechanistic Equivalent of VCM

A typical mechanistic branch is specified by a single equation, linear or nonlinear, algebraic or differential. It combines simple local and transition variables. The parameters of this equation are called mechanistic parameters. The nonlinear terms of the equation are linearised to enable implicit solution which in turn requires iterative nonlinearity error correction each time-step. The linearised equation is identified with a suitable LTE model. The relevant expressions for calculation of the parameters of the LTE model are presented. The described procedure is repeated to an extent required to demonstrate the

applicability to separate physical mechanisms, domain by domain in the following subsections, starting with the electrical circuits domain.

3.6.1 Electrical Circuits

In electric circuitry simulation, branch currents represent the transition variables and node voltages the local variables. The parameters R, R_f, R_b, L, C, I and E of Table 3.2 describe initially constant resistances, inductances, capacitances, source currents and voltages. In the occasion of nonlinearities these constants are substituted with relevant instantly linearised values R_k, R_{kf}, R_{kb}, L_k, C_k, I_k and E_k in a Newton type of iterative nonlinearity correction, as discussed in Subsection 4.6.3.

Semiconductors can appear highly nonlinear or even discontinuous in their behaviour, just depending on the time resolution in consideration (Juslin 1973). Semiconductor layers in some cases encompass a nonlinear capacitive behaviour $C(u)$ depending on the voltage over the capacitor $u = x_i - x_j$. Let us just consider the linearisation of this challenging nonlinearity in detail as an illustrative example of nonlinear electrical circuits.

- **Sample Nonlinear Capacitor Branch.** Consider a capacitor connected between two electrical network nodes. The node voltages are denoted as x_i and x_j and the relevant branch current v_{ij}. In accordance with the MTE notation $g_{ij}\,(\,x_i\,,\ x_j\,,\ dv_{ij}/dt,\ v_{ij}\,,\int v_{ij}\ dt,\ t\,) = 0$ we can write the nonlinear dependency between the node and branch variables at the time instance e as

$$x_{ie} - x_{je}\ - \int_{\Delta e} [C(x_i, x_j)]^{-1} v_{ij}\ dt + E_t = 0,$$

  where E_t is the initial voltage over the capacitor at time t and Δe the applied time-step. The nonlinear capacitance $C(x_i, x_j)$ is specified as a function of the related local variables.

- **Linearisation of the Transition Equation.** The principles of linearisation and stability of iterative correction of nonlinear dependencies are dealt with in detail in Subsection 4.6.3. The charge q_e

$= \int_{\Delta e} v_{ij} \, dt + q_t$ in the capacitor is linearly integrated over the time-step Δ_e. Let us for simplicity consider the voltage over the capacitor as $u = x_i - x_j$. Then the derivative of the charge with respect to voltage at a specific operation point u can be written $q'(u) = dq(u)/du$. Taylor's first order expansion of the charge with respect to the voltage gives $q_e \approx q_k + q'(u_k)(u_e - u_k)$ where u_e is the estimated new voltage difference and u_k the previous value in a Newton-Raphson's type of iterative correction of the nonlinearity errors. We substitute the integrated q_e with its expansion with respect to voltage and get $q_k + q'(u_k)(u_e - u_k) = \int_{\Delta e} v_{ij} \, dt + q_t$. If we denote $C_k = q'(x_{ik}, x_{jk})$ and $E_k = C_k^{-1}(q_t - q_k) + x_{ik} - x_{jk}$ we can write the linearised transition equation dependency as

$$x_{ie} - x_{je} - C_k^{-1} \int_{\Delta e} v_{ij} \, dt + E_k = 0$$

thus well suited for implicit integration from time instance t to $e = t + \Delta_e$. It is supposed that both the function $q(u)$ and its derivative $q'(u)$ are available. The above formulation complies with the linear transition equation number 3 in Table 3.2. We have above derived the expressions for the required linear parameters C_k and E_k.

- **Temporal Discretisation.** In Subsection 3.5.2 the application of numerical integration in order to cope with stiff systems and to manage discontinuities is dealt with in detail. By application of the backward Euler rule, the Trapezoid rule or the second order Gear's rule as presented in Table 3.3, the relevant versatile companion model parameters g_f, g_b and s_{ij} are obtained.

- **Integrated Solution.** The procedures for computing the required parameters a_{ij} and b_i of the relevant node matrix equation $A \, \underline{x}_e = \underline{b}$ to describe an interconnected network made up of all transition equations in concern are set forth in Subsection 4.6.1.

- **Computational Sequence.** The iterative nonlinearity error correction and the proceeding from time-step to time-step entails that some variables need to be computed more frequently than others. Specific calculated variables are stored to be reused in next iteration or time-step to considerably speed up the calculation. We can for instance look at the

role of the different variables, the initiation of a new time-step, the iteration procedure and the acceptance of the new time-step for this example branch. The connection to other branches is only considered in the implicit solution of the voltages. Whilst entering into a new time-step of the length Δ_e we calculate the explicit derivatives of the state variables at time t, and the predictions of the state variables at time e in order to initiate the reference values for iterative nonlinearity correction: $x_{ik} \leftarrow x_{i\hat{e}}$, $x_{jk} \leftarrow x_{j\hat{e}}$ and $v_{ijk} \leftarrow x_{j\hat{e}}$. At each iteration step there are several variables to be calculated in sequence:

1. Voltage u_k and linear transition equation parameters C_k and E_k,
2. Companion model parameters g_f, g_b and s_{ij},
3. Node matrix equation parameters a_{ij} and b_i,
4. Node voltages x_{ie} and x_{je} from $\mathbf{A} \underline{x}_e = \underline{b}$,
5. Secondary variables u_e and v_{ije} from node voltages,
6. Linearised charge q_e used in the calculation, and
7. Correct charge $q_{\underline{e}}$ from the nonlinear correlation $q_{\underline{e}} = q(u_e)$

- **Computation Control.** If the difference $q_{\underline{e}} - q_e$ is not accepted and the maximum step number is not yet exceeded the reference values are substituted with the calculated estimates $x_{ik} \leftarrow x_{ie}$, $x_{jk} \leftarrow x_{je}$ and $v_{ijk} \leftarrow v_{ije}$, and the iteration is continued. The iteration is re-started with an updated time-step length for the purpose of exact hitting a state dependent break-point or if the maximum number of iterations is exceeded. If the iteration is accepted then the relevant time dependent variables are updated $x_{it} \leftarrow x_{ie}$, $x_{jt} \leftarrow x_{je}$, $v_{iju} \leftarrow v_{ijt}$, $v_{ijt} \leftarrow v_{ije}$, $q_u \leftarrow q_t$, $q_t \leftarrow q_{\underline{e}}$ and $t_t \leftarrow t_t + \Delta_e$ and the simulation enters into next time-step.

Three example electrical circuit branches, each with a few mechanisms in series are shown in Figure 3.2. Relevant linear transition equations and the procedures to calculate the versatile companion model parameters are indicated as well. The backward Euler rule has been used for the temporal discretisation. One of the intentions with the figure is to give the reader a general view of the applied modelling principle. Similar linearisation studies as above could be focused on other nonlinear electrical circuit devises, such as resistors and inductors. The methodology is, as we shall find out, to a large extent domain independent. We

shall follow up the study in other technical domains that also involve nonlinear physical mechanisms.

$$x_{ie} - x_{je} - R_k v_{ije} + E_k = 0$$

$$g_f = R_k^{-1}, \quad s_{ij} = g_{ij} E_k$$

$$x_{ie} - x_{je} - C_k^{-1} \int v_{ij} \, dt + E_k = 0$$

$$g_f = C_k \Delta_e^{-1}, \quad s_{ij} = g_f E_k$$

$$x_i - x_j - R_k v_{ij} - L_k \, dv_{ij}/dt + E_k = 0$$

$$g_f = (L_k \Delta_e^{-1} + R_k)^{-1}, \quad s_{ij} = g_f (L_k \Delta^{-1}_e v_{ijt} + E_k)$$
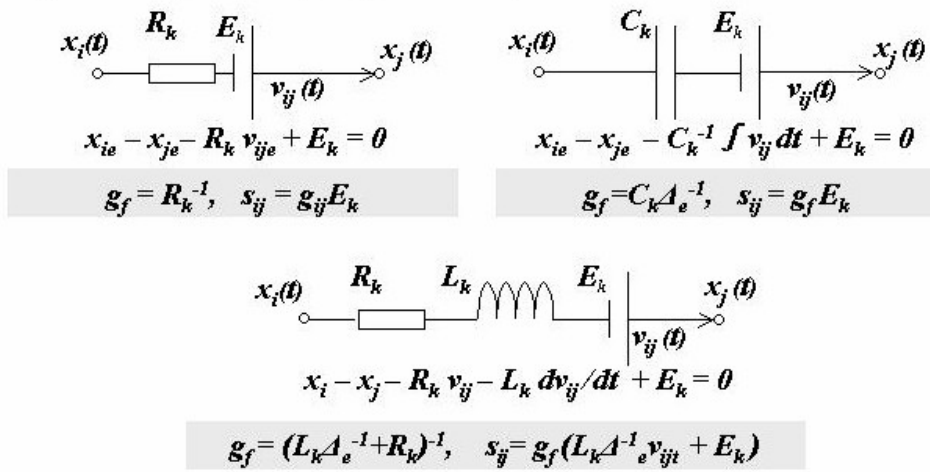
Figure 3.2. Versatile companion model branch parameters g_f, g_b and s_{ij} are presented for each example electrical circuit branch. The equivalent linear transition equations are specified by seriesconnected mechanistic elements such as resistors R_k, capacitors C_k, inductors L_k and voltage sources E_k.

Our example in the Electrical Circuit domain is easily modified to depict the nonlinear compressibility behaviour in the domain of Hydraulic Circuits. If we replace the relevant node voltages x_i and x_j with pressures p_i and p_j, the current v_{ij} with mass flow m_{ij} the charge q with the mass $V\rho(p)$ and the nonlinear capacitance $C(u)$ with $Vd\rho(p)/dp$, then the equation describes the Compression Head mechanism, where V denotes the size of the control volume, $\rho(p)$ the nonlinear density as a function of pressure and $d\rho(p)/dp$ the density derivative with respect to pressure at constant enthalpy conditions. The relevant mechanistic transition equation is presented at line 10 in Table 3.4a and the linear parameters at line 10 in Table 3.4b. As follows we shall consider a sample set of hydraulic mechanisms and their equivalent representation as LTE models.

3.6.2 Hydraulic Circuits

The interactions of the implicit mechanisms need to be taken into account to get a physically meaningful result at the end of each iteration at each time-step. The explicit coefficients need only to be calculated once at each time-step. The separate mechanisms include both variables that are not implicitly solved and in addition nonlinear relations to the state variables. We need to linearise the nonlinear relationships and choose from available explicit and semi-implicit values for calculation of the linear parameters. In Table 3.4a a typical set of separate hydraulic mechanisms has been listed. For convenience the pressure difference $p_i - p_j$ is denoted by Δp and the mass flow dM/dt by m.

Table 3.4a. Mechanistic transition equations (MTE) in the hydraulic domain.

MTE	Separate mechanisms		Mechanistic parameters	
1	*Injection flow*	$m = k_I \rho \omega$	k_I	*Volume injection / revolution*
2	*Laminar loss*	$\Delta p = k_L m$	k_L	*Laminar flow loss coefficient*
3	*Turbulent loss*	$\Delta p = k_T m^2/\rho$	k_T	*Turbulent flow loss coefficient*
4	*Additional loss*	$\Delta p = f_A(m,\rho)$	f_A	*Additional loss from pipe structures*
5	*Centrifugal head*	$\Delta p = k_C \rho_b \omega^2$	k_C	*Effective radius*
6	*Elevation head*	$\Delta p = g_o \rho_b z_b$	z_b ρ_b	*Elevation and mean density between branch connections*
7	*Connection head*	$\Delta p = g_o(\rho_{io} z_{io} - \rho_{jo} z_{jo})$	z_{io} ρ_{io}	*Height and mean density from top of control volume to outlet*
8	*Static flux head*	$\Delta p = k_S m^2 * ((\rho_i A_i^2)^{-1} - (\rho_j A_j^2)^{-1})$	k_S	*Static flux head coefficient*
9	*Momentum*	$\Delta p = k_D dm/dt$	k_D	*Dynamic coefficient L/A*
10	*Compression head*	$m = d\rho/dp V_c dp/dt$	V_c	*Control volume compressibility branch*
11	*Turbine loss*	$p_i^2 - p_j^2 = m^2 T_i k_E$	k_E	*Elliptic loss coefficient*

Some mechanisms are nonlinear with respect to the state variables. For instance, the forced injection mass flow achieved with a piston pump, depends strongly on the density of the media of the relevant branch. The density has not been chosen as a fully implicitly solved state variable. Accordingly, we have to choose from the old iteration step or the old time-step value. Using the last iteration step value in setting up the dynamic parameters as shown in Table 3.4b ensures that, upon convergence, we can not take more mass from the input side node than what there exists, which is the case if we apply the old time-step value.

Table 3.4b. LTE parameters for hydraulic systems.

Node variables x_i, x_j			p_i, p_j = pressures [Pa]		
Branch variable v_{ij}			m_{ij} = mass flow [kg/s]		
MTE	R_k	L_k	C_k	E_k	I_k
1	0	0	0	0	$k_I \rho_{ik} \omega_t$
2	k_L	0	0	0	0
3	$2k_T|m_k|/\rho_{ok}$	0	0	$-R\, m_k/2$	0
4	0	0	0	$f_c(m_t,\rho_t)$	0
5	0	0	0	$k_C \omega_t^2 \rho_{ok}$	0
6	0	0	0	$\rho_b g_o z_{ij}$	0
7	0	0	0	$\rho_{iz} g_o z_{io} - \rho_{jz} g_o z_{jo}$	0
8	$2k_S|m_k|((\rho_{ik} A_i^2)^{-1} - (\rho_{jk}A_j^2)^{-1})$	0	0	$-R\, m_k/2$	0
9	0	k_D	0	0	0
10	0	0	$V(d\rho/dp)_k$	$C_k^{-1}V(\rho_t - \rho_k) + p_k$	0
11	$2k_E|m_k|T_{it}k_E/(p_{ik}+p_{jk})$	0	0	$-R\, m_k/2$	0

Spatial discretisation in the hydraulic domain is exemplified by a long pipe that is structurally specified by flow area and length. The pipe is spatially discretised applying the *Staggered Grid* approach (Patankar 1980) into overlapping *Control Volumes* for calculation of mass *Accumulation* and *Momentum* as shown in Figure 3.3. The relevant physical mechanisms are allocated to simple branches in the associated *Hydraulic Graph*. The internal pressures *p* and mass flows *m* are the dependent variables. The external reference pressure is denoted as p_r. The required number of control volumes depends on the scope of study.
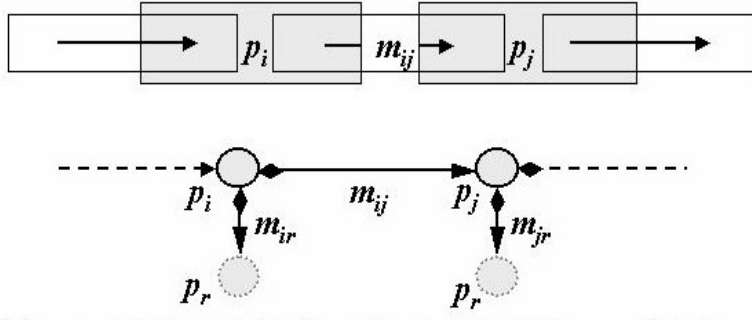
Figure 3.3. Staggered grid approach for one-dimensional spatial discretisation of compressible flow in a pipe. Overlapping control volumes for momentum and accumulation are modelled with relevant branches in the hydraulic graph.

3.6.3 Thermal Fluid Dynamics

Sample equations describing thermal fluid mechanisms arising from convection, accumulation, diffusion and radiation are shown in Table 3.5a.

Table 3.5a. Mechanisms in the thermal fluid domain.

MTE	Separate mechanisms		Mechanistic parameters	
1	*Energy source*	$w = P$	P	*Heating power*
2	*Convection*	$w = m_{ij} k_{ioh} h_i$	k_{ioh}	*Enthalpy outlet ratio($m_{ij}>0$)*
3	*Convection*	$w = m_{ji} k_{joh} h_j$	k_{joh}	*Enthalpy outlet ratio($m_{ij}<0$)*
4	*Accumulation*	$\Sigma w_i = \rho_i V_c \, dh_i/dt$	V_c	*Control volume*
5	*Diffusion*	$w = k_d (T_i - T_j)$	k_d	*Diffusion coefficient*
6	*Radiation*	$w = k_r (T_i^4 - T_j^4)$	k_r	*Radiation coefficient*

The convection depends on the connection point enthalpy and the outlet ratio. The ratio is zero if the flow is not outflow from the node. Therefore, two separate mechanisms are available enabling a possible change of flow direction. The relevant discretised and linearised equations to calculate the dynamic branch parameters are shown in Table 3.5b.

Table 3.5b. Linear parameters from thermal fluid equations.

Node variables x_i, x_j			h_i, h_j = specific enthalpies [kJ kg^{-1}]		
Branch variable v_{ij}			w_{ij} = energy flow [kJ/s]		
MTE	$1/R_f$	$1/R_b$	L	C	E	I
1	0	0	0	0	0	w_i
2	$k_{io}m_{ij}$	0	0	0	0	0
3	0	$k_{jo}m_{ji}$	0	0	0	0
4	0	0	0	$\rho_{ik} V_i$	0	0
5	0	0	0	0	0	$k_d (T_{ik}-T_{jk})$
6	0	0	0	0	0	$k_r (T_{ik}^4-T_{jk}^4)$

Spatial discretisation in the thermal domain is exemplified by an energy storage tank for water of two different temperatures. It is structurally specified by its dimensions including the elevation of its inlet and outlet pipe connections as shown in Figure 3.4. The branches for energy diffusion $w_{12} = k_{12}(T_1-T_2)$ and accumulation $w_r = d(\rho Vh)/dt$ are connected by the specific enthalpy nodes h_1 and h_2 as depicted in the *Thermal Graph*. The level of the surface between the two fluids changes in accordance with relevant hydraulic solution. The mass flows depicted are originating from the preceding solution of relevant hydraulic graph.
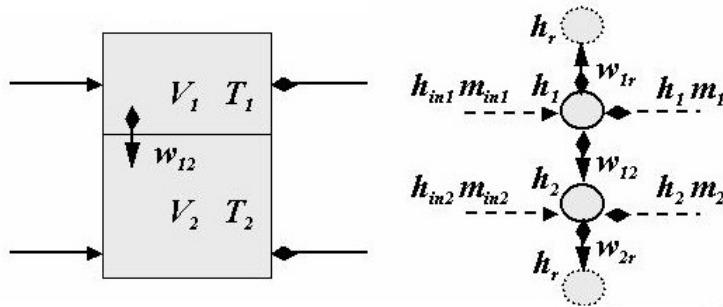
Figure 3.4. The mechanistic specification of thermal diffusion between two fluids of different density in a storage tank is depicted by the associated thermal graph.

3.6.4 Heat Structures

Heat structure mechanisms are shown in Table 3.6a. Conduction and accumulation mechanisms are considered for one-dimensional solid structures. Radiation is considered between adjacent solid surfaces. The conduction and radiation coefficients depend on the material properties in consideration as well as the physical shape (Hänninen 1988). Relevant equations to calculate the linear transition equation parameters are shown in Table 3.6b. The radiation is strongly nonlinear with temperature.

Table 3.6a. Mechanisms in the heat structure domain.

MTE	Separate mechanisms		Mechanistic parameters	
1	*Energy source*	$w = P_h$	P_h	*Heating power*
2	*Conduction*	$w = k_c(T_i - T_j)$	k_c	*Conduction coefficient*
3	*Accumulation*	$w = \rho_i V_i c_s dT_i/dt$	c_s	*Specific heat capacity*
4	*Radiation*	$w = k_r(T_i^4 - T_j^4)$	k_r	*Radiation coefficient*

Table 3.6b. Linear parameters for heat structures.

Node variables x_i, x_j		T_i, T_j = temperatures [^0K]	
Branch variable v_{ij}		w_{ij} = energy flow [kJ/s]	
MTE	R	C	I
1	*0*	*0*	P_h
2	$L/A \ k_c$	*0*	*0*
3	*0*	$\rho_{ik} V_i c_s$	*0*
4	*0*	*0*	$k_r(T_{ik}^4 - T_{jk}^4)$

3.6.5 Fluid to Wall Heat Transfer

Fluid to wall heat transfer mechanisms are shown in Table 3.7a. The heat conduction correlation depends significantly on the surface velocity of the fluid as well as on its surface density. These factors depend much on the detailed

structure of the surface of the heat exchanger elements. The radiation coefficient depends on the density and thickness of the fluid. Relevant equations to solve for the dynamic parameters are shown in Table 3.7b.

Table 3.7a. Mechanisms in the fluid to wall heat transfer domain.

MTE	Separate mechanisms		Mechanistic parameters	
1	*Conduction*	$w=k_T\ (T_F-T_W)$	k_T	*Conduction correlation*
2	*Radiation*	$w=k_r\ (T_F^4-F_W^4)$	k_R	*Radiation coefficient*

Table 3.7b. Linear parameters for fluid to wall heat transfer.

Node variable x_i		h_i = specific enthalpy [kJ kg^{-1}] of fluid	
Node variable x_j		T_j = temperature [^{0}K] of wall	
Branch variable v_{ij}		w_{ij} = energy flow [kJ/s]	
MTE	$1/R_f$	$1/R_b$	I
1	$k_t\, m_{ik}\, Cp_{ik}$	$k_t\, m_{ik}$	$k_t\, m_{ik}\ (T_{ik} - Cp_{ik}\, h_{ik})$
2	0	0	$k_r\ (T_{ik}^4 - T_{jk}^4)$
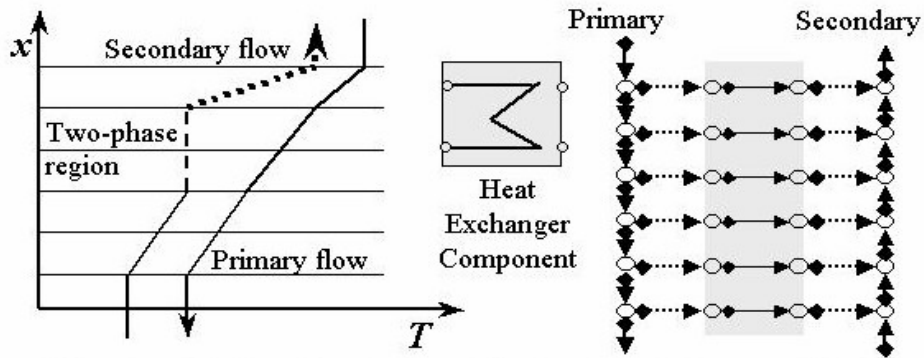
Figure 3.5. Spatial discretisation of a counter current heat exchanger to a thermal graph comprising primary and secondary side flow branches, fluid to wall heat transfer branches and heat transfer branches in the metal wall.

The spatial discretisation of combined convection and diffusion effects in a process component is exemplified by the *Counter Current Heat Exchanger* as shown in Figure 3.5. It is originally specified by relevant dimensions, nominal flows *m*, pressures *p* and enthalpy *h* values. Primary side is filled with pressurised water. Note that secondary water temperature is constant in the two-phase region. The *Thermal Graph* includes simple branches for heat convection (thick), heat transfer (dotted) and heat diffusion in the structures (thin). Each node shown in the graph is in fact a compound node which includes a hidden internal branch for the accumulation of energy.

3.6.6 Concentrations

The concentration of a substance in a control volume can be specified as volumetric concentration, molar fraction or mass fraction. In this context, we have chosen *mass fractions* making it easy to apply the conservation rules of mass. Concentration mechanisms are shown in Table 3.8a. The transportation of a substance depends on the outlet concentration, which can differ from the homogeneous concentration of the whole control volume, e.g. because of phase separation. Relevant equations to solve for the dynamic parameters are shown in Table 3.8b.

Table 3.8a. Mechanisms in the fluid concentration domain.

MTE	Separate mechanisms		Mechanistic parameters	
1	*Substance source*	m_{qi}	m_{qi}	*Injected substance*
2	*Transportation*	$m_{qi}=m_{ij}k_{ioc}c_{qi}$	k_{ioc}	*Concentration outlet ratio*
3	*Transportation*	$m_{qj}=m_{ji}k_{joc}c_{qj}$	k_{joc}	*Concentration outlet ratio*
4	*Concentration*	$\Sigma m_{qi}= \rho_i V_i \, dc_{qi}/dt$	ρ_i	*Density of control volume*
5	*Diffusion*	$m_{qi} = k_q (c_i-c_j)$	k_q	*Diffusion coefficient*

Table 3.8b. Linear parameters from fluid concentration equations.

Node variables x_i, x_j			c_{qi}, c_{qj} = mass fractions of substance q	
Branch variable v_{ij}			m_{ij} = mass flow of mixture [kg/s]	
MTE	$1/R_f$	$1/R_b$	C	I
1	0	0	0	m_{qi}
2	$k_{qio}m_{ij}$	0	0	0
3	0	$k_{qjo}m_{ji}$	0	0
4	0	0	$\rho_i V_i$	0
5	0	0	0	$k_q\,(c_{qik}-c_{qjk})$

3.6.7 Chemical Reactions

Dynamic chemical reaction mechanisms concerning substance **q** in reaction volume **i** are shown in table 3.9a. Very often, the chemical reaction control volume is limited to the volume of a specific phase. Slow diffusion within and especially between the separate phases is often the main limitation factor of chemical reaction speed. Relevant equations to solve for the dynamic parameters are shown in Table 3.9b. Note the reaction rate coefficients R_f and R_b usually are different in the forward and backward directions.

Table 3.9a. Mechanisms in the chemical reaction dynamics domain.

MTE	Rate mechanisms *[kg/s]* for substance **q**		Mechanistic parameters	
1	*Accumulation rate of* **q** *in volume* **i**	$m_{qi}= V_i\,\rho_i\,dc_q/dt$	V_i	*Reaction volume* **i**
			ρ_i	*Density in* **i**
			c_{qi}	*Massfraction of* **q** *in* **i**
2	*Consumption*	$m_{fqi}=-r_{qf\,i}(V,\rho,T,p,c_{1i}..c_{ni})$	r_{fqi}	*Rate by reaction* **f** *in* **i**
3	*Production*	$m_{bqi}=r_{qbi}(V,\rho,T,p,c_{1i}..c_{ni})$	r_{bqi}	*Rate by reaction* **b** *in* **i**
4	*Flow* $m_{ij}>0$	$m_{qi,qj} = m_{ij}c_{qi}$	m_{ij}	*Flow between* **i** *and* **j**
5	*Flow* $m_{ij}<0$	$m_{qj,qi} = m_{ji}c_{qj}$	c_{qj}	*Massfraction of* **q** *in* **j**

Table 3.9b. Linear parameters from chemical reaction equations.

Node variables x_i, x_j	c_p, c_q = mass fractions of substances p and q		
Branch variable v_{ij}	m_{pq} = mass transfer from substance p to q [kg/s]		
MTE	R_f	R_b	C	J
1	0	0	$\rho_i V_i$	0
2	dr_{qf} / dc_p	0	0	$- r_{qf}(V,\rho,T,p,c_{1i}..c_{ni})$
3	0	dr_{qb}/ dc_q	0	$r_{qb}(V,\rho,T,p,c_{1i}..c_{ni})$
4	0	0	0	$- m_{ij}c_{qi}$
5	0	0	0	$m_{ji}c_{qj}$

3.6.8 Rotating Masses

The dynamics of a process plant depends to a significant extent also on the mechanism of the rotating machines involved, such as pumps, turbines, motors, and generators. Table 3.10a describes some elementary mechanisms, such as electrical power source, rotation energy, shaft torsion, bearing friction and gas friction. The local variable is the rotation speed ω_i and the transition variable the shaft momentum M_{ij}. The relevant discretised equations for calculation of the dynamic parameters from the mechanisms are shown in Table 3.10b.

Table 3.10a. Mechanisms in the rotating mass domain.

MTE	Separate mechanisms		Mechanistic parameters	
1	*Power source*	$M_{ij} = P_e/\omega_i$	P_e	*Motor power*
2	*Rotation energy*	$\Sigma_j M_{ij}=k_J d\omega_i/dt$	k_J	*Inertia coefficient*
3	*Shaft torsion*	$\omega_i - \omega_j = k_T dM_{ij}/dt$	k_T	*Torsion coefficient*
4	*Bearing friction*	$M_{ij} = k_B \omega_i$	k_B	*Bearing friction coefficient*
5	*Gas friction*	$M_{ij}= k_G \omega_i^2$	k_G	*Gas friction coefficient*

Table 3.10b. Linear parameters from rotating mass equations.

Node variable x_i	ω_i = Rotation speed $[s^{-1}]$			
Branch variable v_{ij}	M_{ij} = Shaft momentum $[Ws]$			
MTE	**R**	**L**	**C**	**E**	**J**

MTE	**R**	**L**	**C**	**E**	**J**
1	*0*	*0*	*0*	*0*	P_e/ω
2	*0*	*0*	k_J	*0*	*0*
3	*0*	k_T	*0*	*0*	*0*
4	k_B	*0*	*0*	*0*	*0*
5	$2k_G|\omega_{i\,k}|$	*0*	*0*	$-k_G\,\omega_{i\,k}/2$	*0*

3.6.9 Power Distribution Networks

Electrical power distribution network mechanisms are shown in Table 3.11a. The local state variable is the complex main voltage of the node and the transition state variable is the complex line current. The phasor currents and voltages are depicted in an imaginary frame supposed to rotate with the alternating current frequency of the interconnected network island in consideration.

Table 3.11a. Mechanisms in the electrical power network domain.

MTE	Separate mechanisms		Mechanistic parameters	
1	*Synchronous generator*	$\underline{U}_i - \underline{U}_j = -\underline{E}_e + jX_d\underline{I}_{ij}$	$\underline{jX_d}$	*Transient reactance*
			$\underline{E}_e$	*Voltage behind X_d*
2	*Resistive load*	$\underline{U}_i - \underline{U}_j = R_{ij}\underline{I}_{ij}$	R_i	*Load resistance*
3	*Reactive load*	$\underline{U}_i - \underline{U}_j = jX_{ij}\underline{I}_{ij}$	R_i	*Load resistance*
4	*Voltage loss*	$\underline{U}_i - \underline{U}_j = \underline{Y}_{ij}^{-1}\underline{I}_{ij}$	$\underline{Y}_{ij}$	*Branch admittance*
5	*Squirrel cage motor*	$\underline{U}_i - \underline{U}_j = -\underline{E}_m + \underline{Z}_m\,\underline{I}_{ij}$	$\underline{E}_m$	*Reverse voltage*
			$\underline{Z}_m$	*Motor impedance*
6	*Transformer primary and secondary sides*	$\underline{U}_1 - \underline{U}_0 = \underline{Z}_{10}\,\underline{I}_{10} + \underline{Z}_{12}\,\underline{I}_{20}$ $\underline{U}_2 - \underline{U}_0 = \underline{Z}_{21}\,\underline{I}_{10} + \underline{Z}_{20}\,\underline{I}_{20}$	$\underline{Z}_{10}$	*Primary side impedance*
			$\underline{Z}_{20}$	*Secondary impedance*
			$\underline{Z}_{12}$	*Mutual impedance*
			$\underline{Z}_{21}$	*Mutual impedance*

If the fast transients of the distribution network itself are neglected, the dynamics of the load flow and voltages is determined by the mechanical time constants of the rotating machines and the relevant equipment for voltage and frequency control. The versatile companion model coefficients can be calculated as shown in Table 3.11b.

Table 3.11b. Companion model parameters for electrical power networks.

Node state variables $\underline{x_j}$, $\underline{x_j}$, $\underline{x_1}$, $\underline{x_2}$	$\underline{U_i}$ = Node complex voltage [V]
Branch state variables $\underline{y_{ij}}$, $\underline{y_{12}}$, $\underline{y_{10}}$, $\underline{y_{20}}$	$\underline{I_{ij}}$ = Branch complex current [A]
MTE $\quad\quad$ $\underline{g_{ij}}$, $\underline{g_{12}}$, $\underline{g_{10}}$, $\underline{g_{20}}$,	$\underline{S_{ij}}$
1 $\quad$ jX_{dt}^{-1}	$\underline{E_{ek}}\, jX_{dt}^{-1}$
2 $\quad$ R_{ij}^{-1}	0
3 $\quad$ jX_{ij}^{-1}	0
4 $\quad$ $\underline{Y_{ij}}$	0
5 $\quad$ $\underline{Z_{mt}}^{-1}$	$\underline{E_{mk}}\,\underline{Z_{mt}}^{-1}$
6 $\quad$ $\underline{g_{12}} = \underline{Z_{12}}\,(\underline{Z_{10}}\,\underline{Z_{20}} - \underline{Z}^2{}_{12})^{-1}$ $\quad$ $\underline{g_{10}} = \underline{Z_{20}}\,(\underline{Z_{10}}\,\underline{Z_{20}} - \underline{Z}^2{}_{12})^{-1} - \underline{g_{12}}$ $\quad$ $\underline{g_{20}} = \underline{Z_{10}}\,(\underline{Z_{10}}\,\underline{Z_{20}} - \underline{Z}^2{}_{12})^{-1} - \underline{g_{12}}$	0

It is interesting to note that the two transformer equations, including mutual impedance, result in three separate companion model branches. The same principle can be applied for friction between the phases in separated phase flow calculations.

3.7 Function Blocks for Procedural Modelling

Connected function blocks, such as used in control systems, are often presented by block diagrams. A simple transfer function block has one input, one output, and possible parameters and internal state variables. Compound transfer function blocks may have several inputs and outputs. The block diagram can include simple operations on the signals like addition and multiplication. The signals

may be either analog or binary. Binary events are introduced from operations like limit checks and maximum signal selection on analog signals.

3.7.1 Control System Operations

Main emphasis is to use the digital control system software just as it is in a simulator, making it possible to test the control system's real functionality with the model. In many cases, however, it is necessary to test the control system functionality even before any control system vendor has been chosen, whereas the simulation platform itself shall provide suitable control system modules for this purpose. In addition, many process plants still have hydraulic, pneumatic, and hardwired electrical semiconductor or even relay based control systems in use, which in any case need to be simulated. A sample set of analogue control system operations has been described in Table 3.12. The letter s denotes the Laplace operator.

Table 3.12. Functions in the control system domain.

Nr	Separate operations		Parameters	
1	*Set value*	$u_{out} = k_{set}$	k_{set}	*set value*
2	*Amplifier*	$u_{out} = k_P \, u_{in}$	k_P	*Amplification coefficient*
3	*Adder*	$u_{out} = u_{in1} + u_{in2}$		
4	*Multiplier*	$u_{out} = u_{in1} \, u_{in2}$		
5	*Derivator*	$u_{out} = sT_D \, u_{in}$	T_D	*Derivative time constant*
6	*Integrator*	$u_{out} = (sT_I)^{-1} \, u_{in}$	T_I	*Integration time constant*
7	*Measurement*	$u_{out} = k_s \, u_{in} + k_o$	k_s	*Scaling constant*
8	*Control output*	$u_{out} = k_s \, u_{in} + k_o$	k_o	*Offset constant*
9	*Filter*	$u_{out} = (1 + sT_F)^{-1} \, u_{in}$	T_F	*Filter time constant*

The analogue control system operations include scaling, limitations and amplifications and are as such not suitable for any conservation clauses. They

are with preference described as function blocks. This applies to binary control system operations, as well.

3.7.2 Delay Operator

Real processes include many instances of transportation time delay. A delay module can also be included in the control system functionality. Usually the delay time shall be considered as variable rather than constant. For instance, the delay in a pipe depends on the flow velocity. In fact, the direction of the flow could change, as well. Usually the pressures and relevant flows are calculated implicitly. The connection of properties, e.g. concentrations, to delay is by definition explicit. Instead of just connecting the node mass fractions x_i, x_j with one implicit concentration branch, they are connected by two separate branches to the explicit nodes x_{it}, x_{jt} of the delay operator. The delay is usually attained by a "circular" table in computer memory whereas the input and output address vectors are moving, depending on time-step Δt, volumetric flow rate $m_{ij}\rho_i^{-1}$, or conveyor speed $\omega_C r_C$. Interpolation is used for smooth connections to the tabulated values. The total delay depends on set delay time t_D, pipe volume V_{ij} or conveyor length L_{ij}. Table 3.13 shows the application to control signals, flow branches and mechanical conveyers.

Table 3.13. Delay operator mechanisms.

Domain	I/O variables	Accum.	Increment	Parameters
Control	*Signal u*	*Time t_D*	Δt	Δt =*Time-step*
Convection	Specific enthalpy h_i and mass flow m_{ij}	*Volume V_{ij}*	$\Delta V_{ij} =$ $m_{ij}\rho_i^{-1}\Delta t$	ρ_i =*Input density*
Concen-tration	Concentration c_{qi} and mass flow m_{ij}	*Volume V_{ij}*	$\Delta V_{ij} =$ $m_{ij}\rho_i^{-1}\Delta t$	ρ_i =*Input density*
Conveyor	Concentration c_{qi} and mass flow m_{ij}	*Length L_{ij}*	$\Delta L_{ij} =$ $\omega_C r_C \Delta t$	ω_C =*Motor speed* r_C =*Eff. radius*

3.7.3 Material Property Functions

Typical material properties needed in dynamic simulation are shown in Table 3.14. It is supposed that the thermal hydraulic equilibrium in a control volume is completely specified by the pressure, the mass fractions and the specific enthalpy of the mixture. The other properties of a homogeneously mixed control volume, such as gas, liquid and solid fractions, as well as phase specific mass fractions, enthalpies, densities, specific heat coefficients, and viscosities, can be looked up or calculated. In real-time simulation applications, non-iterative methods for fast look up of the material properties are preferred (Nelles 2001). First, they provide a repeatable value for each call with same input parameters as they are not contaminated with iteration noise, secondly, they are several orders of magnitude faster.

Table 3.14. Material properties at thermal dynamic equilibriumin control volume.

Node variables as input				
p	*Pressure at top of volume*			
h	*Specific enthalpy of mixture*			
c_q	*Substance mass fractions in mixture* $q=1..N_q$			
Material properties as output	Mixture	Gas	Liquid	Solid
Temperature	T			
Density dependency of pressure	$(d\rho/dp)_h$			
Density dependency of enthalpy	$(d\rho/dh)_p$			
Specific enthalpy		h_g	h_l	h_s
Phase mass fractions		α_g	α_l	α_s
Substance mass fractions $q=1..N_q$		$c_{q,g}$	$c_{q,l}$	$c_{q,s}$
Density	ρ	ρ_g	ρ_l	ρ_s
Speed of sound	u	u_g	u_l	u_s
Viscosity	η	η_g	η_l	η_s
Specific isobaric heat capacity	c_p	$c_{p,g}$	$c_{p,l}$	$c_{p,s}$

If the temperatures of the phases are equal to the temperature of a homogeneous mixture then a thermal dynamic equilibrium situation has been reached. It shall be noted that the time constants for diffusion between phases usually are very

nonsymmetric, that is, boiling is faster than condensation, and solidifying faster than liquefying.

From the viewpoint of practical design work, it is beneficial that the supporting software recognizes the separate isolated fluid departments of a process model, based on the connections in the flow sheet. Possible changes in the number of substances considered in a specific department then only need to be made to relevant instance of department specifications.

3.7.4 Separation and Mixing Operations

Separation in process industry is achieved applying suitable physical mechanisms. Phase separation is usually achieved easily because of different densities of the phases. If the density difference is small then centrifugal separators can be used. In distillation columns, the difference in concentration distribution of liquid and gas phases at the same temperature is used to separate different substances from a mixture.

Mechanical filters are frequently used to separate solid phase objects from liquid phase. Osmotic liquid filters can be used to separate dissolved large molecules from smaller. Drying by heating or vacuum is also one kind of separation. An interesting detail is that phase separation rate and mixing rate usually are very different. Dissolving a gas into liquid takes very long time if only the diffusion mechanism transition the liquid surface in the tank is used. It sometimes also takes very long time to reach thermal dynamic equilibrium. By efficient mixing, the effective surface area can be extended considerably and the process speeded up. On the other hand, separation of gas from liquid phase for instance by decreasing the pressure is very fast, boiling appears everywhere in the liquid. The same kind of speed difference applies for dissolving rate of solid phase into fluid compared to extraction rate from fluid.

In addition, the persistence of separation invoked by temperature and density differences within the same phase needs to be considered. In general, the diffusion is very slow if no convection, either natural or forced is present. In practise, we only have strong mixing conditions in such parts of the process that involve turbulent flow. Most pipes and tanks perform like a combination of

mixers, splitters and delay lines. Accordingly, the position of an input or output flow connection in a simple tank has much impact on the resulting performance.

The amount of different substances to be transported from a specific connection point outlet to a branch depends on the flow rate of mixture in the branch, the elevation of the connection point in the tank, the node densities and phase distribution at the connection point, characterised by a set of parameters as set forth in Table 3.15.

Table 3.15. Separation and mixing mechanisms regarding a homogeneous branch outlet from a phase separating tank.

Outlet area fractions	$a_{og} + a_{ol} + a_{os} = 1$
Outlet density	$\rho_o = a_{og}\rho_g + a_{ol}\rho_l + a_{os}\rho_s$
Outlet enthalpy	$h_o = (a_{og}h_g\rho_g + a_{ol}h_l\rho_l + a_{os}h_s\rho_s)/\rho_o$
Outlet mass fraction of substance q	$c_{oq} = (a_{og}c_{gq}\rho_g + a_{ol}c_{lq}\rho_l + a_{os}c_{sq}\rho_s)/\rho_o$
Enthalpy outlet ratio	$k_{oc} = h_o/h$
Concentration outlet ratio	$k_{oc} = c_o/c$
Branch density	$\rho_b = (\rho_{io} + \rho_{jo})/2$

If the outlet area fractions comply with the relevant volumetric phase fractions, then the outlet properties will have the properties of the homogeneous tank mixture in the outlet node. Likewise, the enthalpy and concentration outlet ratios will be equal to one. A good approximation for the branch density is the mean value of the densities of adjacent nodes in a smoothly changing situation. Long pipes connecting nodes with very different densities require the mass content to be integrated supposing that the volume flow in different parts of the pipe is equal. Just substituting branch density with outlet density may give an incorrect result and result in oscillations upon change of flow direction.

The concentration outlet ratios are used as splitting ratios between any substances, even originating from the same phase. Typical applications are centrifugal separators or filters. Sometimes it is preferable to use a source

branch, for either forced injection or removal of a certain substance. In a local compound node with no volume or splitting specified, there will be no hold-up and accordingly immediate ideal mixing of the input stream takes place. If there is a volume specified there will be hold-up and ideal mixing of the content, taking into account all input and output streams.

3.7.5 Mechanistic Parameters from Input Specifications

The coefficients and parameters needed by the mechanistic equations for on-line calculation are not very easily found by a occasional user. Instead, a set of preparatory equations needs to be specified in advance to allow the user to enter easily available data such as dimensions and nominal values of operation. These equations need to be calculated only upon change of input. Some examples on such parameters are given in Tables 3.16a and 3.16b.

Table 3.16a. Pre-calculated mechanistic parameters from input specifications.

Rotary pump			
Centrifugal head /Effective radius k_C	$k_C = \Delta p_S / (\rho_N \omega_N^2)$	Δp_S	*Shut-off head*
		ρ_N	*Nominal fluid density*
		ω_N	*Nominal speed*
Turbulent loss / Loss coefficient k_T	$k_T = \rho_N / (\Delta p_S - \Delta p_N)/m_N$	Δp_N	*Nominal head*
		m_N	*Nominal mass flow*
Compression in tank			
Compression head/ Control volume V_C	$V_C = \pi d_{vt}^2 z_{vt}/4$	d_{vt}	*Tank diameter*
		z_{vt}	*Tank height*
Pipe transition element			
Dynamic head / Head coefficient k_{LA}	$k_{LA} = 4L_{hp}/(\pi d_{hp}^2)$	d_{hp}	*Pipe inner diameter*
		L_{hp}	*Pipe length*
Turbulent loss / Loss coefficient k_T	$= \Delta p_N \rho_N / m_N^2$	m_N	*Nominal mass flow*
		Δp_N	*Nominal pressure loss*
		ρ_N	*Nominal fluid density*

Table 3.16b. Pre-calculated mechanistic parameters from input specifications.

Turbine			
Turbine loss / Elliptic loss coefficient k_E	$k_E=(p_{in}^2-p_{out}^2)/(m_N^2 T_{in})$	T_{in}	*Nom. inlet temperature*
		m_N	*Nominal mass flow*
		p_{in}	*Nom. inlet pressure*
		p_{out}	*Nom. outlet pressure*
Transformer			
Transformer model admittances	$\underline{Y}_{12}=\mu_0(\underline{Z}_o^2-\underline{Z}_0\underline{Z}_k)^{0.5}/(\underline{Z}_0\underline{Z}_k)$ $\underline{Y}_{10}=1/\underline{Z}_k - \underline{Y}_{12}$ $\underline{Y}_{20}=\mu_0^2/\underline{Z}_k - \underline{Y}_{12}$	$\underline{Z}_0$	*No-load impedance*
		$\underline{Z}_k$	*Short circuit impedance*
		μ_0	*No-load voltage ratio*

3.7.6 Secondary Variables from Solved States

Upon solution of the state variables, secondary variables of interest are calculated for the component branches. They can be calculated when the estimates of relevant state variables are known. If included in the iteration scheme, they need to be calculated upon each iteration step. Otherwise it is sufficient if they are calculated each time-step. Typical secondary variables are described in Table 3.17.

Table 3.17. Secondary variables from solved state values.

Rotary pump			
Shaft power	$P_{rp}=k_C\,\omega_k^2\,m_t$	k_C	*Effective radius*
		ω_t	*Speed*
		m_t	*Mass flow*
Separated phase tank			
Liquid level	$z_l=(1-\alpha_l)\,z_{vt}$	α_l	*Volumetric gas fraction*
		z_{vt}	*Vertical tank height*

3.8 Structured Components of Real World Processes

Grouped graphs are specified as to comply with structured components of real world processes. Higher level components are built up by underlying components in a hierarchical manner. Figure 3.6 illustrates how compound branches and nodes are made up from simple branches and nodes. It shows a continuously stirred tank reactor attached to two pipes. Each relevant *Compound Branch* includes simple branches of different domains in parallel: Hydraulic, thermal, and concentration branches. A compound branch can include other compound branches in series, as well. The *Compound Node* includes several simple nodes and simple local branches, describing local phenomena such as compressibility, heat capacity, reactions, or hold up of substances.
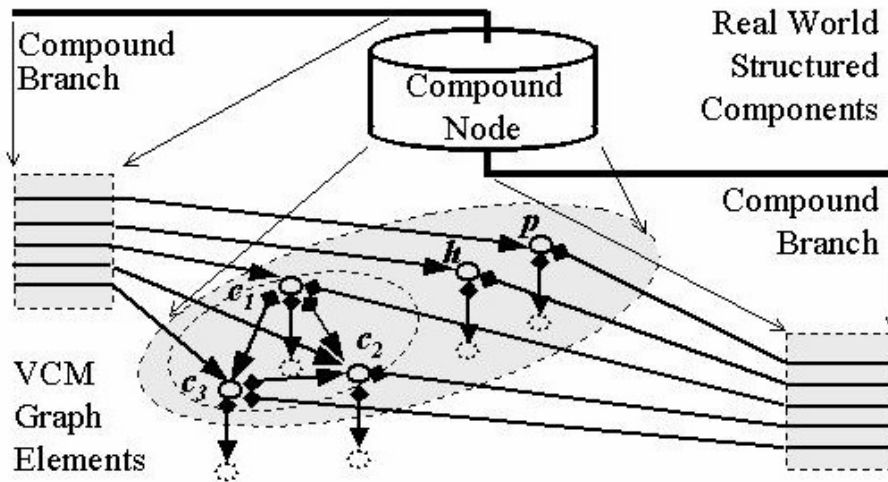
Figure 3.6. A compound node connects two compound branches. Each compound branch includes simple branches depicting flow of mass, energy and three separate substances. The simple nodes related to pressure p, specific enthalpy h, and the mass fractions c_1, c_2 and c_3 are also connected to vertical branches depicting accumulation of mass, energy and substances. In addition there are local branches describing chemical reactions between the substances.

A compound node can include other compound node types, as well. Typical combinations of substances can for instances be specified into pre-defined compound node types. A compound branch can also be made up of other compound branches. A district-heating pipe assembly has usually two spatially

discretised pipes in parallel, the outgoing water and the return water pipes with a common insulation structure.

A *Generic Component* can comprise branches and nodes as well as function blocks. The required input parameters of the underlying mechanistic equations are listed for the user to fill in. Provision is made to use a dedicated script to calculate the all required parameters from a smaller set of user given attribute values. For instance, a model for studies of the compressible flow in a long pipe has to be discretised spatially to a required number of limited control volumes depicting transition branches as well as local compressibility branches in accordance with the staggered grid approach. The local volume is typically made up from half of the volume of each adjacent pipe element. The input parameters include the dimensions of the pipe and the requested number of control volumes for spatial discretisation. The generation script specifies the underlying structure based on the given parameters. If the underlying structure of a generic component is changed manually, the component will evolve into a assembled component, a kind of prototype which is not possible to generate again by the original script. Pre-designed generic components are stored as types in generic component script repositories. Parameter lists of real components corresponding to pre-designed generic component types can be stored as instances in generic component parameter repositories.

Assembled Components are usually specified graphically by drawing the process diagram and filling in the query forms for each separately included component. Accordingly, by combination of components it is possible to build up models of more and more complicated unit operations. For example a heat exchanger usually includes component models for hydraulic flow, thermal flow, heat conduction in the pipe walls as well as for the heat transfer between pipe surface and fluid. The local volumes also include material properties look-up for boiling and condensing phenomena.

The internal structure of connected *Function Blocks* of a complicated controller is specified graphically to a structured component, and provided with its own symbol. Parameters of specific interest of the underlying components are collected to an assembled parameter list of the higher-level symbol. Accordingly, a assembled component can be obtained, by drawing its structure using specification graphics, by modifying a generic component, or by

combining selected parts of other assembled or generic components. Assembled components are stored in assembled component repositories. The structured component model is an important hierarchical building block for construction of large integrated process models. The structured component has two views, its *Flow Sheet* view reveals its internal structure for modifications, its *Symbol* view is practical to use when designing higher-level flow sheets.

3.9 Plant Model Repositories and Related Services

Complete plant model specifications need to be stored for the whole life-cycle of the plant. The model specification shall survive, all the time, even if the computer hardware and operating system is updated or the simulation server software is improved with new versions. Flow sheets of industrial processes, such as process and instrumentation diagrams, include much accumulated design knowledge expressed in a very compact way. Modern information technology, and the possibility to make the above flow sheets alive, enables suppliers or operators of industrial plants to establish efficient remote diagnostics and service centres. The measurements from real plants are easily available over Internet. Measurement recordings cam be compared to simulation results.

In connection to a remote simulation service centre, virtual expert teams can be formed if required. The trainees and teachers can form virtual simulator training classes. The remote experts and the trouble-shooters on the plant can form virtual support teams. It can be transparent for the users, on which computer and where the simulation server resides. It is important that the virtual team responsible for updating the models get access to all planned modifications and relevant work orders of the real plant. Of course, the added value is that the changes can be tested in advance on the simulator, before implementation at the plant. The operators can be trained on the new features in advance, as well.

3.10 Unified Specification Framework

In this chapter the author has described his novel developments of a *Unified Specification Framework* to link hierarchical process structure and physical mechanisms with relevant mathematical equations in a very intuitive and

transparent way, well suited for the engineering way of thinking. It supports a declarative approach to model specification instead of sequential programming procedures. A hierarchy of graphs is an attractive starting point for developing graphical user interfaces, both for specification of models and for monitoring of simulation results. It also presents means for the required information hiding necessary for the managing of large model entities and to protect possible proprietary data. The *Specification Data View* presented in Appendix A is very complementary to this chapter describing the data structures required for implementation. For the purpose of structured specification of the process plant model data, evolving semantic web technologies should be taken into account as indicated in Chapter 5. Let us, however, first have a look at the solution system architecture.

4. Solution System Architecture

The hierarchical specification architecture of a modelled heterogeneous process is flattened down to a *System Graph* of simple nodes and branches. Further, the system graph is partitioned into separate *Homogeneous Zones* based on *Strong, Normal* and *Weak* connections identified by comparing the versatile companion model parameter values. Strongly connected nodes are temporarily joined and weak branch connections are temporarily omitted for the solution in the in the normal zone. Suitable integration schemes and time-steps are applied for the separate homogeneous zones. This partitioning increases the sparsity, which considerably speeds up the implicit solvers, besides that, it prevents serious round-off errors. An added value is that the partitioning also provides for application of coarse grain parallelism, involving parallel threads or even distributed processing. The total system graph is then divided into *Local Regions* to be processed by separate simulation engines.

Solution with different time-steps for separate homogeneous zones has been applied e.g. for pressure and energy solution already in the early Loviisa training simulator. There is normally at least two orders of magnitude difference between the speed of pressure transients and heat transients. In the early implementation the time-steps used for the energy solution were only slightly longer than those for the pressure solution. Hence the calculated energy solution values could as such successfully be used as boundary conditions for the more frequently updated pressure solution without any attempts to extrapolate the boundary conditions. See Subsection 4.6.4 for state prediction issues. The concept of local regions was also implemented whence the computation of separate loosely connected parts of the real plant was manually allocated to separate computers.

4.1 Stiff and Steady Equations

Some dynamic equations of the process are sometimes so stiff or steady that they need special treatment because of restricted computer real value representation. Attempts to solve the various dependencies in one single set of equations easily

result in a matrix singularity, literally frozen variables or eliminated feedback. The problem has in previous implementations been overcome by manual partitioning of the system, applying scaling of the variables and different time-step lengths for related parts of the model. Very stiff dependencies have manually been replaced by algebraic dependencies that, however, introduces all the uncertainty problems related to solution of nonlinear algebraic equations whereas the history is eliminated. Unable to benefit from the history, we may occasionally end up in erroneous solutions. A serious study of the system would require the calculation of all its system eigenvalues, which would be very time consuming and only valid for small perturbations around a selected steady state of a nonlinear dynamic process. In the new implementation it is proposed to automatically take into consideration that the linear transition equation view of the versatile companion model gives access to the inductive or capacitive time constants of relevant branch. They are not system time constants but represent a good starting point for the proposed automated estimation of suitable initial time-steps for the separate homogeneous zones.

Other criteria may have impact on the time-step, such as the *Courant Number Restriction (*Courant et al. 1967) initially considered for pipe branches. The Courant number is specified as $N_C = u\,\Delta t\,/\,\Delta L$, where u is the flow velocity. Maintaining a suitable Courant number less than one requires that the distance travelled by advection during one time-step Δt is not larger than one spatial ΔL. In explicit systems maintaining the Courant number less than one is required for numerical stability, in implicit systems it is not required but recommended especially during transients to maintain the accuracy. Very large differences in the Courant number in adjacent branches indicates that the spatial discretisation is not very optimal. It is proposed to automatically normalise spatial discretisation based on initially given flow specifications at model construction time based on Courant number evaluation. At run-time it is proposed to automatically watch the Courant number and adjust the time-step accordingly.

4.2 Homogeneous Zones

All structures are flattened down to a linear *System Graph.* The versatile companion model parameters resulting from the temporal discretisation of the linear transition equations with chosen initial time-step length, provide for

further analysis. A *Normal Monitoring Time-step* is chosen just depending on the transients in consideration of the simulation study. A *Fast Time-step* is chosen at least one order of magnitude shorter and a *Slow Time-step* is chosen at least one order of magnitude longer. Interconnected branches fitting between the fast and slow monitoring time-steps are considered as belonging to a *Normal Homogeneous Zone*. Slower branches are assigned to the *Weak Zones* and faster branches to the *Strong Zones*. An interconnected part of a stronger zone is temporarily represented by a *Lumped Node* for calculation in an adjacent weaker zone. Results from weaker zones are used as *Interpolated Boundary Conditions* when calculating more frequently a stronger zone. The calculation sequence is discussed in detail in Section 4.7.

Consider an example model comprising a very short pipe connection between two nodes, which further are connected at each side by ordinary long pipes. An attempt to implicitly solve the pressures of both nodes may cause severe numerical problems. The nodes are replaced with a stand-in node for the solution in the normal zone. The relevant strong zone comprises accordingly the two eliminated nodes and the short connecting branch. First, the pressures and flows of the normal zone are solved. Thereupon, the flow in short branch is solved considering the pre-calculated flows of the long branches as source values. The pressure difference in the short branch can be accurately calculated, as well. Table 4.1 gives a typical overview of pipe network elements classified into different homogeneous zones.

Table 4.1. Classification of pipe network elements into different zones.

Weak	Normal	Strong
Closed valves	*Pumps*	*Short pipe joints*
Small leakages	*Long pipes*	*Tank internal flows*
Heat losses to environment	*Heating of pipe walls*	*Heat exchanger plates*

When simulating large thermal hydraulic processes in real-time, then the monitoring time-step of the normal region might be 500 ms, the monitoring time-step for the strong fast region might be one order of magnitude shorter, such as 50 ms, and the time-step for the weak slow region one order of magnitude longer, such as 5 s. Figure 4.1 indicates such updating frequencies for
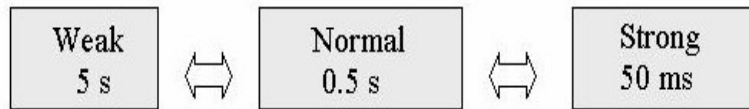
Figure 4.1. Weak, normal and strong zones of a system graph of an industrial process are calculated with different time-steps. The time-step bandwidths often originate from the requirements of an interconnected digital control system.

the said zones. If for instance the focus of the study is on some fast switching transient, then the normal region will be assigned a very short time-step used for monitoring the transient, and the time-steps for weak and strong zones are shifted accordingly.

4.3 Numerical Integration

When modelling an industrial process for the target purposes of optimizing the operational procedures, testing of new automation concepts, or training of experienced operators, it is crucial that the models perform correctly in all upcoming situations. For instance, opening and closing of a control-valve has a strong impact on associated time constants of the system. Accordingly, the resulting numerical problem can be stiff or not, just depending on the operational situation. Stiff problems are, however, very difficult for solvers not specifically designed for them. Several numerical integration methods have been developed especially for stiff applications.

As we shall later on find out, preference is made to use *Absolute Stable* methods that do not impose any stability restrictions on the step-length. The *Precisely Absolute Stable* methods have a stability region comprising exactly the left half plane. They are considered as the most secure methods for stiff systems (Lambert 1991). Their tendency to oscillate in some situations can easily be treated without degradation of the accuracy. Smoothing of such initial oscillations have been applied with good success (Lindberg 1971). Using the

notations introduced in Section 3.5 the calculated value v_t at time-step t can be corrected subject to constant time-step as follows:

$$v_t \; \leftarrow \; {}^1\!/_4 \, (\, v_e + 2v_t + v_u \,).$$

The *Strongly Absolute Stable* methods have a stability region that includes the whole left plane and in addition extends to parts of the right plane. Their solution can, however, be successively damped too much and produce misleading results in some rare cases when the system has eigenvalues with positive real parts.

Higher order methods seem advantageous because of their smaller truncation errors that enable relatively longer step-lengths. The higher order is achieved by increasing the number of *steps* considered in linear multistep methods, or increasing the number of *stages* within a Runge-Kutta step. There are however some drawbacks form choosing higher order methods than necessary:

- After each discontinuity a multistep integration method needs to be subsequently started up with a sequence of relevant lower step-number calculations, staring with a one-step method. Further, the sufficient number of old time-step values need to be recorded, which causes overhead, as well.

- A higher order Runge-Kutta method needs several intermediate stage calculations of the variables during the time-step, whereas the order is built up subsequently. Additional errors will be accumulated if a nonlinear system is supposed to be linear for all the intermediate calculation stages, which may degrade the benefit from the higher order.

- The equation systems arising from the nonlinear target processes in consideration, such as pipe networks, are usually intrinsically implicit and sparse. Symbolic manipulation of the mathematical expressions for formulation of explicit derivatives could be very difficult if not even impossible. Probably the required manipulation would result in loss of the sparsity (Brenan et al. 1989). Accordingly, the required derivatives in higher order methods need usually to be calculated numerically applying several steps or stages as indicated above.

- In real-time simulation of industrial processes the request for relatively short maximum time-steps originates from the communication frequencies of relevant digital control systems. Accordingly, we can not make use of the potential benefit of longer time-steps enabled by higher order integration even if the fast transients have passed.

- The very frequent changes of time-step length required to cope with both nonlinearities and approaching discontinuities makes truly one-step methods to look very advantageous. The nonlinearities appearing can be very severe: The ratio of the density of steam and water in two-phase flow conditions can be in the order of one to hundred thousand.

- The simulation engine shall calculate in real-time such a detailed simulation model of an integrated plant that possibly includes more than ten thousand state variables. The variables to be presented shall be updated at least each hundred millisecond to be available for the control room equipment in a nuclear power plant training simulator. Such requirements do not allow for more sophisticated mathematical methods than what necessarily is needed.

- The integration must be fail-safe in all situations. A full-scope training simulator needs to run reliably from day to day, from year to year. There is no possibility to manually change the integration method according to possible new demands arising for each transient under study. The eigenvalues can instantly change over a very large range e.g. as a result from a guillotine break of a large pipe.

To be completely sure that the numerical integration not is the root of instability, we require that the integration method used is Absolute Stable. Accordingly, we have chosen to initiate the integration after each discontinuity with a Strongly Absolute Stable method and thereupon continue with a Precisely Absolute Stable method. The following types of numerical integration methods have been studied from this point of view: The linear multistep methods, The Runge-Kutta methods and the predictor-corrector methods. The most suitable ones have been applied in Subsection 3.5.2.

4.3.1 Linear Multistep Methods

It has been shown (Dahlquist 1963) that an explicit Linear Multistep Method (LMM) cannot be absolute stable. The order of an absolute stable LMM cannot exceed two. The second-order absolute stable LMM with smallest truncation error is the one-step implicit Adams-Moulton method (Lambert 1991). This method is also known as the *Trapezoid Rule*. It is precisely absolute stable. Using the same notations as in Section 3.5 we can write

$$v_e = v_t + \tfrac{1}{2}\Delta_e(v'_e + v'_t).$$

The one-step Backward Differentiation Formulae (BDF) is a first order method having a larger truncation error than the Trapezoid rule. It is also known as Gear's first order method or the *Backward Euler Rule*. It is strongly absolute stable.

$$v_e = v_t + \Delta_e\, v'_e.$$

To exemplify a two-step Backward Differentiation Formulae, the *Second Order Gear's Rule* has been implemented. It is strongly absolutely stable, as well. For equidistant time-steps we can write

$$v_e = \tfrac{4}{3}\, v_t - \tfrac{1}{3}\, v_u + \tfrac{2}{3}\,\Delta_e\, v'_e.$$

It is possible to apply the second order Gear's rule for moderately non-equal time-steps, whereas the applicable coefficients have to be re-calculated each time the time-step length changes. The second order Gear's rule also needs to be initiated with a the one-step method after each discontinuity. For higher order Gear's methods the unstable region is gradually entering the left plane.

4.3.2 Runge-Kutta Methods

Comparing the truncation errors of implicit Runge-Kutta methods complying with the requirement of no intermediate calculation stages, the second order Lobatto IIIA method appears as most suitable. In fact, it is the same as the *Trapezoid Rule*. Another possible choice is the one-stage Radau IIA method,

which is also called implicit Euler's method or the *Backward Euler Rule*. Higher order Runge-Kutta methods will not be considered.

4.3.3 Predictor-Corrector Methods

Taylor's series expansion for state prediction is dealt with in Subsection 4.6.4. A typical predictor-corrector pair is formed by the explicit *two-step Adams Bashforth* method

$$v_e = v_t + \tfrac{1}{2}\Delta_e(3v'_t - v_u)$$

used as predictor and the implicit one-step Trapezoid rule used as corrector. It shall be noted that the two-step method needs to be started with the one-step Adams Bashforth method, also known as the *Forward Euler Rule*

$$v_e = v_t + \Delta_e v'_t.$$

The importance of the predictor becomes very evident when considering stiff nonlinear problems in which the usual fixed point iteration needs to be replaced with Newton iteration to ensure convergence for reasonable step-lengths. A good prediction can provide for a close-enough starting point for the Newton iteration to ensure fast convergence. The nonlinearity based errors in the coefficients of the equations as well as the numerical integration related errors are corrected simultaneously during each iteration made. The overall stability is governed by the implicit corrector method if the iteration is continued to convergence, that is until the iteration error is of the order of the round-off error. Combined with this kind of extrapolation the effective order of the Trapezoid rule can be raised to 4 (Lambert 1991).

4.3.4 The Theta Method

The *Theta Method*, depending on the value of its factor θ, combines the features of both Backward Euler for $\theta = 0$, Trapezoid rule for $\theta = \tfrac{1}{2}$, and Forward Euler for $\theta = 1$.

$$v_e = v_t + \Delta_e\,((1\text{-}\theta)v'_e + \theta\,v'_t).$$

It is accordingly Strongly Absolute Stable for $\theta < \frac{1}{2}$, Precise Absolute Stable for $\theta = \frac{1}{2}$, and less than full Absolute Stable otherwise. It has order one in general but order two only for $\theta = \frac{1}{2}$. An interesting feature of the Theta Method is that it yields the exact solution for a suitably chosen value of θ (Lambert 1991). Applying a technique called *Exponential Fitting* to a test function $v' = \lambda\,v$ the exact solution can be given by

$$\theta = -\,\Lambda^{-1} - e^{\Lambda}(1 - e^{\Lambda})^{-1},$$

where $\Lambda = \lambda\,\Delta_e$. It shall be noted that the Absolute Stability is preserved for all values of $\Lambda < 0$. In a more complicated physical system, λ stands for the real part of the eigenvalue related to relevant state variable v. Direct calculation of eigenvalues is only feasible for constant coefficient linear systems of restricted size. Sometimes it can, however, be beneficial to estimate the worst cases that are smallest and largest possible absolute values of the real part of the eigenvalue. The choice can also depend on the intention of the user, whether he wants to see the accurate representation of fast transients or if he only is interested in the slower dynamics. Sometimes, vibrations with the speed of voice of the media in consideration are of interest, sometimes only much slower transients fit into the scope of the study.

4.4 Initial and steady states

Consistency control of initial state variables need to be made before starting the simulation after a modification of model structure or upon passing a discontinuity. This is especially important for a *Differential-Algebraic Equation (DAE)* system $F(t,y,y') = 0$ that comprises both ordinary differential equations and algebraic equations. The *Index Number* of a DAE system indicates the minimum number of subsequent differentiations of relevant parts of the system needed to get an implicit *Ordinary Differential Equation (ODE)* system (Brenan et al. 1989).

If considered necessary, the system is initiated with relatively small time-steps applied in order to reach a *Dynamic Initial State*. For an *Index One* system one

Backward Euler step is sufficient to correct mismatches in initial values. Respectively in a *Index Two* system two Backward Euler steps are needed. The Linear Transition Equations as set forth in Subsection 3.5.1 are at most index two systems. Each Backward Euler step needs to be iterated to convergence if the system is nonlinear. The dynamic initial state is of basic interest in dynamic simulation because many real processes never reach a static steady state.

A *Static Steady State* is considered as equilibrium of a process resulting from a certain set of boundary conditions. An important issue to remember is, however, that in many occasions the resulting state depends on the operational path transition which the static steady state has been reached. In a static nonlinear equation system, there can in fact be a several more or less correct steady state solutions.

In order to stabilize the iterative solution and keep the states close to a given initial guess, artificial dynamics is sometimes applied in steady-state solvers. A well-structured dynamic solver has, on the other hand, the possibility to temporarily streamline all the time constants involved, to fasten up the reaching of a static steady state.

4.5 Positive Definite or Diagonally Dominant

It is advisable not to solve the whole model as a single large implicit island. Partitioning and grouping methods are applied to cope with numerical problems resulting from too big differences in adjacent branch values. Tearing up of loops helps to preserve the sparsity of implicit islands. Partitioning into separate homogeneous zones gives the possibility to use different time-step propagation for the separate parts of the model. All these precautions aim at getting well-behaving matrix equations to solve, with all the non-zero element values within the same order of magnitude, possessing diagonal elements strong enough to avoid the need for pivoting.

The resulting matrix should preferably be positive definite or diagonally strong enough to avoid numerical instability. Fortunately, there is a very simple rule that easily can be implemented. Each connected graph is analysed before the construction of relevant matrix to ensure that at least one branch is connected to

an external node. This is achieved in a pipe network with non-compressible fluid if at least one branch is connected to an external node. Also in an isolated pipe network the requirement is fulfilled if the compressibility of the fluid is taken into account by at least one compressibility branch connected to external zero pressure.

In a dynamic situations the connection between two nodes could be instantly zero, for instance if parallel inductive and capacitive branches compensate each other at a specific frequency. To overcome these kind of situations, the diagonal elements are checked during the factorisation procedure. If the diagonal element for some reason becomes zero for one iteration step, the relevant node variable is not solved but replaced with previous iteration step value. Zero divide does not occur and at next iteration step the diagonal element usually already keeps a non-zero value. A good hint is not to idealize the real world too much: There are usually resistances and energy losses in all branches, including electrical capacitors, inductors and voltage sources.

Table 4.2. Solving implicit islands.

Update explicit state dependent parameters
Make estimates for new time-step, node and branch variables
→ Calculate linear transition equation elements from nonlinear equations
Calculate companion model coefficients
Determine solution order if requested
Build sparse matrix
Factorise matrix
Solve node and branch variables
Look up material properties
Solve secondary variables
Iterate nonlinearity errors

4.6 Implicit Islands

Each terminal of a versatile companion model branch can be connected, either (fully) implicitly, semi-implicitly (last iteration value) or explicitly (last time-step value), to the adjacent nodes. An *Implicit Island* is formed by a set of implicitly interconnected nodes and branches. At the boundary of the island there are usually branches belonging to the island but connected semi-implicitly or explicitly to nodes of other islands. The sequence for solving implicit islands is shown in Table 4.2.

Same node can be included only in one island as an *Internal Node* but it can be included as an *External Node* in several islands. Similarly, a branch can be included only in one island as an *Internal Branch* but as an *External Branch,* it can be included in several islands. The smallest possible implicit island is made up of a single internal node connected by a single internal branch to an external node. Note again that at least one internal branch needs to be connected to an external node in each implicit island in order to have a reference for the solution.

4.6.1 Matrix Equations from VCM Parameters

The measured positive flow direction in a pipe can be specified as is convenient for the user. Internal branches connecting internal nodes may have any specified positive direction, however, an internal branch v_{ji} connecting an implicit island to an external node is temporarily during the course of solution directed outwards.

The general conservation law depicts that the sum of the inward v_{ji} and outward v_{ij} streams of a non-accumulating simple node i shall be zero as follows:

$$\Sigma_{j,in} \; v_{ji} \; - \Sigma_{j,out} \; v_{ij} \; = \; 0.$$

Repeating the relevant expressions for all internal nodes i of the implicit island and substituting the stream values v_{ij} with the relevant companion model terms $x_i g_f - x_j g_b + s_{ij}$ gives the necessary set of equations to specify a linear system for solving all its internal node values. The equations are organised into a linear *Node Matrix Equation*,

$$A \underline{x} = \underline{b},$$

where A is as called the *Node Matrix*, $\underline{b}$ the *Source Vector,* and $\underline{x}$ the *Solution Vector.*

The causality and the direction of the versatile companion model branch as well as the connection type of each of its terminals are considered. An external branch is always considered as forced. All other causalities considered are related to internal branches. The composition of the node matrix and the source vector from companion model parameters is described in detail in Table 4.3. The data architecture for storage of the branch parameters are set forth in Table A.10 in Appendix A. The data architecture for storage of the sparse matrix equation elements is described in Table B.6. in Appendix B.

Table 4.3. Matrix equation elements from VCM parameters.

Causality of relevant companion model branch	Branch directed from node i to j		Branch related increments to the node matrix equation $A\underline{x} = \underline{b}$ elements.					
	i	j	a_{ii}	a_{jj}	a_{ij}	a_{ji}	b_i	b_j
Forced branch	internal	internal	0	0	0	0	$-s_{ij}$	s_{ij}
$v_{ij} = s_{ij}$	internal	external	0	0	0	0	$-s_{ij}$	0
Unicausal branch	internal	internal	g_f	0	0	$-g_f$	0	0
$v_{ij} = g_f x_i$	internal	external	g_f	0	0	0	0	0
Unicausal branch	internal	internal	0	g_b	$-g_b$	0	0	0
$v_{ij} = -g_b x_j$	internal	external	0	0	0	0	$g_b x_j$	0
Bicausal branch	internal	internal	g_f	g_b	$-g_b$	$-g_f$	$-s_{ij}$	s_{ij}
$v_{ij} = g_f x_i - g_b x_j + s_{ij}$	internal	external	g_f	0	0	0	$g_b x_j - s_{ij}$	0
A-causal branch	internal	internal	g	g	$-g$	$-g$	$-s_{ij}$	s_{ij}
$v_{ij} = g(x_i - x_j) + s_{ij}$	internal	external	g	0	0	0	$g x_j - s_{ij}$	0

4.6.2 Sparse Matrix Operations

A matrix including so many zero elements that it pays off to use sparse matrix technique when solving the related linear equation system is regarded as sparse.

The construction and solution of small sized matrix equations can well be done using standard full matrix techniques, including factorisation and pivoting. However, for large matrices sparse matrix methods need to be employed. Pivoting should be avoided. It is important to remember that a system matrix formed as above including all possible connections implicitly is usually not very sparse. Grouping of the equations combined with tearing of looser dependencies should be used. Whence there anywhere is a need to iterate for correction of non-linearity, the errors from non-implicit connections will also be eliminated.

The other reason for sparse techniques, besides the speed, is the reduced need for memory to store the matrix during the calculation. A linked list scheme is flexible enough to allow for fill in during the factorisation. Some precautions need to be taken regarding the factorisation order. In the worst case, we may end up with filling the whole matrix during the factorisation.

4.6.2.1 Optimised Solution Order

Optimisation of the solution order is apparently needed to avoid unnecessary fill in of non-zero elements in the usually very sparse matrices resulting from the partitioning measures. The implicitly connected graph contains the information needed for the optimisation of the solution order of the matrix equation variables. A simple algorithm that keeps track on the number of branches connected to each node during the initial construction of the matrix is usually good enough. The algorithm can be described as a simulated elimination of nodes and adjacent branches from the graph, starting with the nodes having the smallest number of connections, and continuously keeping book on the remaining connections. If, however, such an elimination of a node removes a connection between two adjacent nodes, and they do not previously have a directly interconnecting branch, then a relevant fill in branch is created.

4.6.2.2 Matrix Factorisation

The bi-factorisation (Zollenkopf 1971) has been considered efficient for the very sparse network matrices originating from thermal hydraulic circuits and electrical networks. Usually only two or three pipes are connected to same pipe joint. An important benefit is that the scheme is easy to use for complex value matrices of power networks, as well. There is no need to calculate time

consuming square roots during the factorisation. In addition, the scheme can be applied for fully symmetric matrices requiring computation and storage positions just for half of the number of the off-diagonal elements. The bi-factorisation needs only to be done if there has been a change of the matrix element values or connections. The factorisation scheme used for the sparse matrices constructed is presented in Table 4.4a. The factorisation is performed within the data structures set forth in Table B.6. A notable benefit of the method is that the factorised matrix is re-using the storage positions of the original matrix. Possible new matrix elements required during the factorisation are easily added to the end of the linked list.

Table 4.4a. Sparse matrix factorisation.

Start with diagonal element $a(k,k)$ from top left	
→	Invert the diagonal element

	Check all column elements $a(i,k)$ down and all row elements $a(k,j)$ to the right from the diagonal element: If both elements exist then
	    if $a(i,j)$ exists then update $a(i,j) \leftarrow a(i,j) - a(k,j)*a(i,k)* a(k,k)$
	    otherwise create a new element $a(i,j) \leftarrow - a(k,j)*a(i,k)* a(k,k)$

	Check all column elements $a(i,k)$ down from the diagonal element: If $a(i,k)$ exists then update $a(i,k) \leftarrow - a(i,k)*a(k,k)$
	Check all row elements $a(k,j)$ to the right from the diagonal element: If $a(k,j)$ exists then update $a(k,j) \leftarrow - a(k,j)*a(k,k)$
	Repeat for all diagonal elements k

4.6.2.3 Matrix Solution

Upon the factorisation, the solution of the node variables is simply done by multiplication of the source vector with the left and right hand factor matrices. The source vector is successively developing to the solution vector, as shown in Table 4.4b. The new node variable values in $b(k)$ need to be copied to their positions in the state variable x_e repository as described in Table A.16 before the relevant branch variables v_{ij} are updated. The sparse matrix factorisation scheme is very fast for typical industrial network applications.

Table 4.4b. Solving node and branch variables.

Start with diagonal $a(k,k)$ elements from top left
→ Update source vector value for all existing column elements $a(i,k)$ below the diagonal $b(i) \leftarrow b(i) + a(i,k) *b(k)$
Update source vector value for the diagonal $b(k) \leftarrow a(k,k)*b(k)$
Repeat for all diagonal elements k
Start with diagonal $a(k,k)$ elements from bottom right
→ Update source vector value for all existing row elements $a(k,j)$ to the left from the diagonal $b(k) \leftarrow b(k) + a(k,j) *b(j)$ to obtain the solution
Copy the $b(k)$ values to their positions in the state value x repository
Repeat for all diagonal elements k
Solve the relevant branch variables related to the component $v_{ij} = g_f x_i - g_b x_j + s_{ij}$

Typical computation times per node are shown in Figure 4.2 as a function of total number of nodes for comparison of the efficiency with regard to other methods to solve a set of linear equations (Juslin 1983a).
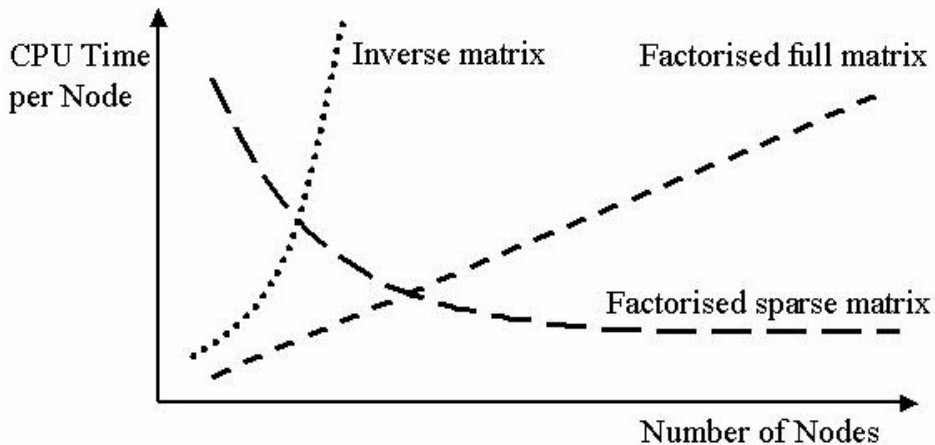

Figure 4.2. Computation times per node as a function of the number of nodes are compared when solving a linear equation system A$\underline{x}$=$\underline{b}$ by computing the inverse of the matrix, applying full matrix factorisation and sparse matrix factorisation.

The sparse matrix equation solver has too much overhead to be economical for small matrices. The sparse matrix approach becomes faster than full matrix equation solvers for dimensions larger than about ten. In a certain region of network dimensions the solution time for sparse matrices seems to be almost linear with respect to the dimension. For larger matrices the data organisation and the size of available fast cache memory has an important role. Finally, the solution time increases dramatically when semiconductor memory size is exceeded and swapping to disk memory locations is required. This limit is, however, reached much earlier when operating with full matrices.

4.6.3 Nonlinear Equations

The nonlinearity error control and compensation needs to be applied separately for each domain taking into account the type of nonlinearity and the operative range of the variables. For example, in a thermal hydraulic network, the accumulated control volume density resulting from branch flows is compared with the density calculated from the nonlinear material properties. In some cases, the compensation of the resulting error can be delayed to the initiation of the following time-step. However, if an implicit solution of strongly connected state variables such as pressures is requested, then an iterative correction method needs to be applied until the remaining errors are fully acceptable before proceeding to the next time-step.

4.6.3.1 Newton's Method

The quadratic convergence feature of the Newton's method makes it very attractive for fast calculating codes. However, there are strict requirements on the type of nonlinearity and on the starting point that needs to be "close enough" to the expected solution, otherwise the method can converge to a wrong solution, slow down to negligible corrections or diverge.

Taylor series are applied to the linearisation of the nonlinear terms in the equations of concern. Let us consider the nonlinear terms $f(x)$ or $f(v)$ and their derivatives $f_x(x)$ or $f_v(v)$ related respectively to a local variable x_i or a transition variable v_{ij} in some of the nonlinear *Mechanistic Transition Equations* included in the modelled system. Both state variables are denoted as x just for the

purpose of this linearisation study. We restrict the study to a given interval for the variable in consideration denoted as

$$\Omega = (x_a, x_b).$$

We can write the first order Taylor expansion of the nonlinear term $f(x_e)$ based on the state value x_k calculated at previous iteration step,

$$f(x_e) \approx f(x_k) + (x_e\text{-}x_k)\, f_x\,(x_k).$$

A set of *Linear Transition Equations* is formed when all nonlinear terms of the original equations are substituted with expansions as above. The linear transition equations are discretised with respect to time to form *Versatile Companion Models* according Subsection 3.5.2. The transition variables v_{ij} are eliminated from the equation system as set forth in Subsection 4.6.1 and we can write for the remaining local state variables

$$A(\underline{x}_k)\ \underline{x}_e\ =\ \underline{b}(\underline{x}_k).$$

When both the linear system matrix $A(\underline{x}_k)$ and the source vector $\underline{b}(\underline{x}_k)$ elements have been updated with previous iteration step $\underline{x}_k$ values we can solve the new state vector estimate $\underline{x}_e$. Thereupon we can solve all relevant transition variables v_{ije}. Note that in this formulation the derivative terms of the linear transition equations are not collected to a separate Jacobean matrix. They are conveniently included in the linear system matrix, still preserving its sparsity. The reason for keeping the different nonlinearities as well as their derivatives distinct and not lumped together, is to make it possible to consider their physical background and relevant validity range during the simulation. A simple nonlinear turbulent flow resistance mechanism is studied in Figure 4.3, where pressure drop over a control valve with mass flow m is $\varDelta p$. Correct choice of the primary variable, either m or $\varDelta p$, provides for global convergence of the Newton type correction of nonlinearity errors.
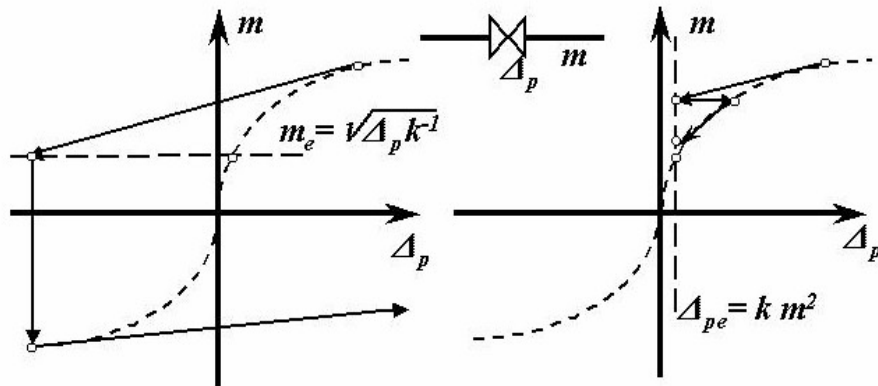
Figure 4.3. To the left the mass flow is iterated as a function of pressure and seems to diverge. To the right the pressure is iterated as a function of mass flow and is rapidly converging.

4.6.3.2 Monotonic Regions

The derivative $f_x(x_k)$ with respect to the state variable in consideration is either analytically solved from the function $f(x)$ or calculated numerically from the following central difference expression, where Δ_x denotes a small deviation applied to the state value.

$$f_x(x_k) \approx (f(x_k + \Delta_x) - f(x_k - \Delta_x))/(2\Delta_x)$$

The function $f(x)$ needs to fulfil a set of requirements. The standard assumptions (Kelley 2003) for convergence are as follows:

1. Equation $f(x) = 0$ has a solution $x = x^*$ within Ω

2. The derivative is non-singular $\| f_x(x)^{-1} \| \leq \alpha$ within Ω

3. The derivative is Lipschitz continuous within Ω

The Lipschitz continuity means that the difference $f_x(x_i)-f_x(x_j)$ is roughly proportional to the difference $x_i - x_j$ for any values of x_i and x_j within Ω (Gockenbach 2003). We can write the Lipschitz condition as follows,

$$\| f_x(x_i) - f_x(x_j)\| \leq L \| x_i - x_j \|$$

where L is the Lipschitz constant. The Newton's method is proved to converge quadratically (Sebah & Gourdon 2001) if also the following Kantorovitch conditions apply

4. $\| f_x(x)^{-1} f(x)\| \leq \beta$ within Ω,

5. $\| f_{xx}(x)\| \leq \gamma$ within Ω, and

6. $\alpha \beta \gamma < 1/2$ within Ω.

A function that confirms with the above conditions is characterized in this thesis as *Monotonic Function*. In practice it means that it is either strictly monotonically increasing or strictly monotonically decreasing but never constant within the interval of consideration. If the nonlinearities of $f(x)$ are more complicated than above then separate *Monotonic Regions* Ω_K each confirming with these requirements need to be identified and the crossing of the region borders to be considered as crossing of discontinuities. To enable spurious iteration beyond the borders of the each region, the regional functions are monotonically extended making use of the gradients at the border. If the solution converges outside of the said monotonic region, the iteration needs to be restarted taking into use the relevant adjacent monotonic region. This approach provides for global convergence.

In order to study monotonic regions of nonlinear dependencies, also dedicated steam table look-up functions (Wagner & Krause, 1998) were programmed in Object Pascal by the author. The concept of monotonic regions is illustrated in Figure 4.4 by two surfaces extracted from the steam tables. The left surface shows the correct water region provided with an extended monotonic surface replacing the two phase and steam regions. The right surface shows the correct steam and two-phase regions provided with an extended monotonic surface replacing the waterside. The intersection line between the two surfaces if placed

in the same coordinate system depicts the saturation line from waterside until the critical point and continues with the critical enthalpy line for higher pressures in the supercritical region.
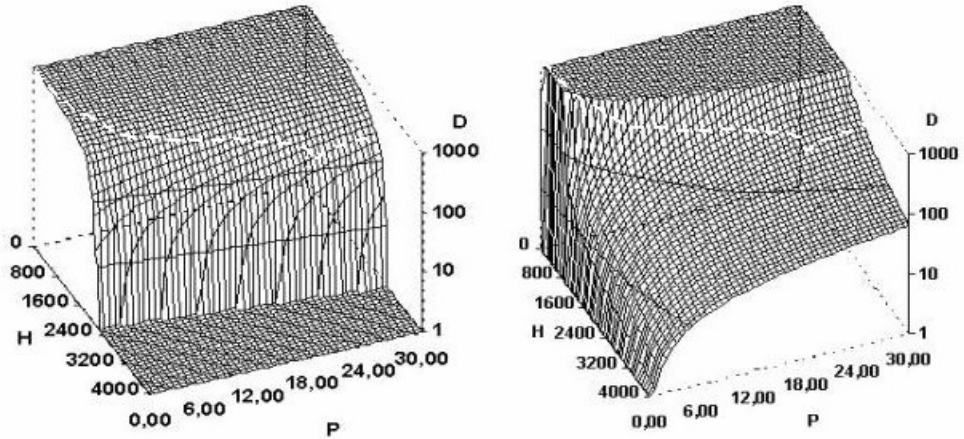

Figure 4.4. Logarithmic density surfaces for water and two-phase region as a function of pressure and specific enthalpy. The water region is monotonically extended as "overheated" and the two-phase region as "supercooled".

4.6.4 State Prediction

In order to provide a better starting point for the iterative refinement of the non-linearities encountered for the state variables than just the previous time-step values, a prediction procedure is applied. Taylor's expansion is used in an explicit manner with respect to time, whereas x_t denotes the known state value calculated at t time instance, x'_t its first derivative with respect to time, x''_t its second derivative, Δ_e the anticipated new time-step length and $x_{\hat{e}}$ the predicted new state value.

$$x_{\hat{e}} = x_t + \Delta_e\, x'_t + \Delta_e^{\,2} x''_t / 2! + O(\Delta_e^{\,3})$$

108

The second order approximation is normally used. Note that the error term $O(\Delta_e^3)$ vanishes if the behaviour of the state variable during the time interval can be exactly described by a second order polynomial as a function of time. The first order approximation is used when a time-step change is required e.g. to exactly catch a discontinuity and the zero order approximation is used immediately after the crossing of a discontinuity or when starting the up the simulation.

Backward numerical differentiation is used to approximate the required first order derivative x'_t with respect to time based on the previous time-step length Δ_u and the relevant value of the state variable x_u.

$$x'_t \approx (x_t - x_u) / \Delta_u$$

Backward numerical differentiation is also used to approximate the required second order derivative x''_t with respect to time, based on the old derivative value x'_u stored from previous time-step as well as the present value x'_t.

$$x''_t \approx (x'_t - x'_u) / \Delta_u$$

Having access to predicted state variables makes it possible to pre-calculate selected material properties and estimate the distances to approaching discontinuities, as well.

Prediction is also proposed to be used to the calculation of boundary conditions in the interconnect of separate homogeneous zones with different time-steps, as well as for connection of separate local regions. In the case of communication between separate asynchronously running simulation engines a consistent set of data for the publishing of a boundary condition comprises the state value itself, its first and second order derivatives and the time-stamp.

4.6.5 Time-Step Control

There are several intrinsic constraints on the time-steps used. User sometimes wants to define general minimum and maximum time-steps allowed, subject to the nature of the study. The time-step specified for communication outside the
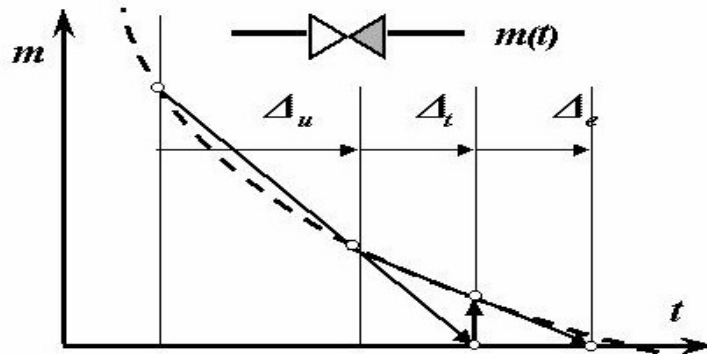
Figure 4.5. In order to avoid mass errors during the simulation of the operation of a check valve, a simulation step needs to be calculated exactly at the time for change of flow direction in the valve resulting in the closure of the valve.

model or recording purposes also needs to be considered. Too long time-steps can result in an unwanted increase of the number of iterations required for compensation of the nonlinearities. In addition, too short time-steps increases computation time and may lead to numerical problems. The simulation error control can automatically decrease or increase the time-step length within set applied limitations. Sometimes the request for shorter time-step comes from the discontinuity control.

Typical discontinuity events in process simulation are originating from change of flow direction in a check valve or the total filling up of a tank. Other discontinuities can arise from the operation of the simulated control system. It is very important to precisely identify the discontinuity points making it possible to perform a simulation calculation just at each break point. The prediction of the involved variables can be used for estimation of the crossing times in advance. If the estimate after calculation of the simulation step is found not to be close enough to the break point, the procedure is repeated.

Let us consider a linear extrapolation example with regard to pipe networks. The exact time for closure of a check valve is estimated by successive adaptation of the time-step as shown in Figure 4.5. The decreasing mass flow *m* as a function of time *t* transition the check valve is indicated by the dotted line.

110

4.7 Running the Simulation

Usually a division of the total model into three distinct temporal zones is sufficient. It is convenient if the time-steps of the different zones are multiples of each other's. Of course, initiation of large transients or crossing discontinuities require intermediate calculation steps, as well.

An interesting issue is how to combine the different temporal zones. Linear interpolation from the old instance values to estimated next time-step values of slower zones could provide boundary conditions for the faster zones that are more accurate than just constant values. However, for this approach to work properly and always proceed forward corresponding to real-time, the instances for reaching discontinuities need to be predicted very carefully. Table 4.5 shows the running sequence of a local simulation engine.

Table 4.5. Running a local simulation engine including homogeneous zones.

→ Request input variables from other zones, simulation engines or control systems
→ → Calculate sequence of function blocks and implicit islands
Iterate if required for correction of nonlinearities
Iterate if required for passing of discontinuity
Release output variables to other zones
Calculate each *temporal zone* at requested time instances

The different input-output modules within a temporal zone, such as external function blocks and separate implicit islands, need to be calculated in a sequence, starting from input values at the boundary of the temporal zone. If the sorting algorithm encounters an algebraic loop, iteration of the sequence is needed. In addition, nonlinear algebraic dependencies can require iteration, as well. In an environment with several simulation engines or connections to virtual control systems, a synchronization of the different communicating actors is needed, if the experiments are supposed to be repeatable and not depending of occasional load changes of different parts of the computer system.

4.8 Tracking and Prediction

Advanced operator support facilities can include a *Predictive Simulator* enabling the operator to try out what-if scenarios, before applying the actions in consideration to the real plant. The predictive simulator needs to be initiated with the exact situation at the real plant, including possible malfunctions. Whereas the predictive simulator has much more internal state variables than what is available from any measurements, it is convenient make use of a *Tracking Simulator* that only needs a few measurements as boundary variables to be able to run in parallel with the real process imitating the real process behaviour. A snapshot of all internal state variables of the tracking simulator is copied to the predictive simulator before running each scenario. The predictive simulator needs to run several hundred times faster than real-time to provide fast enough responses for the operator.

The tracking simulator needs to be identical with the predictive simulator, replicating both process and automation systems. Diagnostics tools can be attached to the tracking simulator to identify possible malfunctions of the process, and apply them on request to the model as well. Adaptive tools can be used to adjust the models according to foreseen slow changes in some distinct process component parameters. Table 4.6 shows the running sequence of a predictive simulator.

Table 4.6. Predictive what-if or optimising simulator.

Run tracking simulator continuously in parallel with real process
→ Copy snapshot of tracking simulator to predicting simulator on request
Make possible control operations to predictive simulator
Run estimate curves much faster than real-time
Test manually or apply automatic optimisation scheme for operational procedures

Optimisation tools can run the predictive simulator for automated evaluation of best operational strategy of the plant, and advise the plant operators accordingly. The optimal strategy can even be employed without any intervention of the operators in model-based control applications.

4.9 Error Sources

Validation of the performance of a simulation model is made by comparing simulated results with real measurements. Modern digital control systems provide facilities for easy recording of transients in a real plant. We need, however, to look critically at the measured values. The inaccuracy of the measurements, the time constants of the transducers and the sampling errors may cause surprises. Specific *Data Reconciliation* methods have been developed (Amand et al. 2000) to identify *Measurement Errors* in order to validate measurement data. When comparing a simulation experiment with recorded measurements from a real plant we need to take into account that the model might look better than it really is. If, for instance, the model includes a control system actively correcting the controlled variables, the remaining errors in the process models can influence on possibly not measured variables such as valve positions, and remain hidden. There are many sources for errors in the simulation model: The specification of the spatial discretisation, the physical mechanisms considered, the selection of relevant mathematical formulas, the numerical solution methods applied, and the implementation of the computational model.

The *Spatial Discretisation Errors* originating from the division of a process plant model into distinct *Control Volumes* can be much more prominent than the numerical integration errors. It seems unreasonable to discretise with respect to time with much less than one second as a time slot but at the same time apply a spatial discretisation granularity of tens of cubic metres or tonnes. Another severe misconception is to consider all tanks as ideal mixers. Rapid mixing is only achieved by the introduction of mixing energy. In most tanks and pipes the plug flow conditions are prevailing. Only fast turbulent flow in narrowly dimensioned pipes results in some mixing. The ratio of mixing and plug flow depends thus not only on the geometry but also on the flow velocity.

The *Mechanistic Presumption Errors* relate to a wrong view on which physical phenomena to consider with respect to the scope of study. For instance, should separate phase flow conditions, momentum flux related pressure drop, or density imposed separation be considered in the specific case?

The *Material Property Errors* are introduced from approximate material property functions. Relatively correct information is available for pure substances. To consider several substances in separated phase conditions, *Flash Calculations* need to be made for evaluation of the mass fraction distribution in each phase in each control volume. The applicable *Equations of State* can usually only be tuned for correct calculation close to one operational point.

The *Parameter Errors* are probably the most common errors. They usually arise from wrong understanding of the meaning or the unit of measure of model parameter attributes in concern or just from typing errors. The remedy is to have computerised access to semantic process design databases and component information galleries. Including a sensibleness check for the parameter values is helpful, as well.

The *Empiric Correlation Errors* relate to such dependencies that only can be correctly determined by measurement from the real target process or from pilot equipments. Typical empiric correlations are related to friction, turbulence, mixing, reaction rates and heat transfer.

The *Round-Off Error* depends on the number of available digits for calculation and storage as well as on the degree of ill-conditioning of the system. A problem is considered *Ill-Conditioned* if very small relative perturbations in the parameters make relatively large errors in the solution. The applied sequence of calculation of separate parts of an expression can also have significant impact on the resulting error (Wilkinson 1994). Round-off errors are usually not tracked in real-time simulation and are sometimes very unpredictable. One possibility for checking the impact is to recompile the complete model with a different floating point number representation and compare the results.

An *Index Reduction Error* can arise when reducing the index of a DAE system (Brenan et al. 1989), because the algebraic constraints, such as conservation of mass or energy, not are fulfilled by the resulting ODE. In the versatile companion model approach index reduction is fortunately not needed.

A *Local Truncation Error* originates from the truncation of the number of terms considered in a series approximation being the basis for the development of e.g. a numerical integration method. The Principal Local Truncation Error $\boldsymbol{e}_{plt}$ of any

numerical integration method can be estimated subject to constant time-step applying the Richardson's method $e_{plt} = (v_e - v_{\underline{e}})/(2^n - 1)$ where n is the order of the method (Lambert 1991). Applying the Trapezoid rule, the state variable calculated at each time-step is calculated from $v_e = v_t + \frac{1}{2}\Delta_e(v'_e + v'_t)$. The result can be compared with a supposedly less accurate value $v_{\underline{e}} = v_u + \Delta_e(v'_{\underline{e}} + v'_u)$ achieved by using the double time-step length for the calculation. It shall be noted that when this check is made the computation effort is doubled. Another possibility is to repeat the simulation study with a forced shorter time-step and compare the results.

If the system has nonlinear coefficients, iteration is required for their correction at the end of each time-step. For stability reasons, the iteration needs to be stopped in such a way that the *Nonlinearity Correction Error* remains at least one order of magnitude larger than underlying round-off or relevant other error. Nested iterations are also very time consuming. For this reason, nested iterations shall be avoided as far as possible. Concurrent iterations are hence preferable, but much more demanding from stability point of view. See Subsection 4.6.3.

Crossing a discontinuity appearing either in a variable itself or its derivative may cause a significant *Discontinuity Passing Error*. To minimize the error the numerical integration time-step needs to be modified to exactly hit the discontinuity point. A linear iteration method to estimate the discontinuity time instance has been successfully applied by the author (Juslin 1973). In order to reduce the number of the required iterations also polynomial approximation methods can be used (Elmegaard & Houbak 2001). See Subsection 4.6.4.

There is usually an optimum for the applicable time-step with regard to the total accumulated *Global Error*. For longer time-steps the truncation error increases and for shorter time-steps the round-off error becomes dominant. Applying even longer time-steps usually results in more iterations for correction of nonlinearities and may finally cause instability. Decreasing the time-step even more can totally stop the integration of slow variables and feedback loops can be eliminated causing instability. It is interesting to note that the optimal time-step for accuracy often also provides for the fastest calculation.

Implementation Errors arise from insufficient specification, documentation and bad project management resulting in misconceptions of and between computer software programmers, or just in unresolved simple printing errors.

4.10 Verification and Validation

Good software development practices do not ensure correct operation of simulation engine software. It could encounter half a million lines of code written by tens of programmers and scientists. An extensive verification and validation procedure needs to be employed before any release of a new software version. The procedure includes both separate tests and integral tests. The software shall allow for easy testing of each mechanism and correlation separately, for testing of assembled process component models, as well as for the testing of large integrated process models. The verification and validation work processes are specified as follows (Carson 2002):

- *Verification* occurs when the model developer exercises an apparently correct model for the specific purpose of finding and fixing modelling errors.

- *Validation* occurs when the model developer and people knowledgeable of the real system or new/modified design jointly work to review and evaluate how a model works.

There exist several internationally recognised test facilities, such as PACTEL at Lappeenranta University of Technology (Tuunanen et al. 1998). They are usually built for physical performance related research issues. Series of carefully recorded experiments have been made at these facilities. When a sufficient number of test facilities are modelled with modular simulation software and it calculates the performance of selected experiments correctly enough, then it is assumed that the modular software will calculate correctly also in other configurations. The procedure shall be repeated each time the source code is changed before the release of a new version (Ylijoki & Norrman 2004).

Edwards' pipe test (Edwards & O'Brien 1970) is chosen in this context to exemplify the use of separate effect tests. The purpose is to examine the sudden depressurisation of a horizontal pipe discharged from one end. It is initially

containing water at a pressure of 6.895 Mpa and a temperature of 242 °C. Its length is 4096 mm and diameter 73 mm. The discharge flow area is 15% smaller than the area of the actual pipe. Both measured values and simulated results are shown in Figure 4.6.
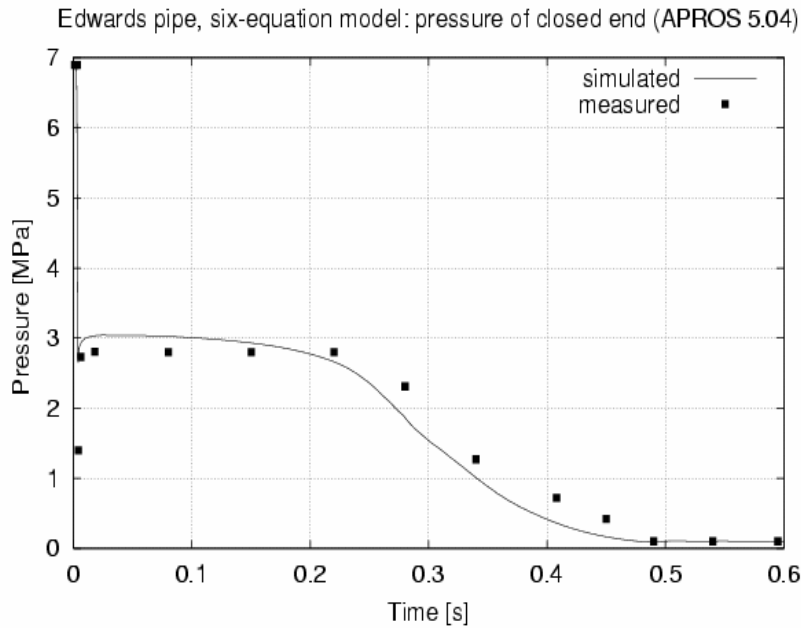

Edwards pipe, six-equation model: pressure of closed end (APROS 5.04)

Figure 4.6. Measured values and simulated results show the pressure at the closed end of the Ewards pipe during a discharge flow experiment.

It should be noted that it is not enough that the simulation code itself is verified and validated. The application models are presently manually specified by drawing process and automation diagrams and filling in required parameters of the components included. Manual specification errors are easily introduced. Accordingly, also the application models need to be validated to ensure correct results.

4.11 Use Case of an Integrated Model

The use of sufficiently verified and validated simulator code is successfully demonstrated by the construction and employment of an integrated model of the

Loviisa nuclear power plant. In 1995 Fortum Oy decided to commence a modernization and power uprating project in which major parts of the thermal hydraulic analyses of the revised Final Safety Analysis Report (FSAR) were calculated using the APROS simulation software. A completely new simulation model of Loviisa NPP was build to ensure that all the input data was correct and that the data sources were properly documented. The model included the whole primary circuit with safety systems, steam generators, steam lines and safety-critical automation systems. The model was extensively validated against measurement data from plant commissioning tests etc. About 30 different initiating events and scenarios were calculated and their sensitivity to various parameters investigated (Kantee et al. 1998).
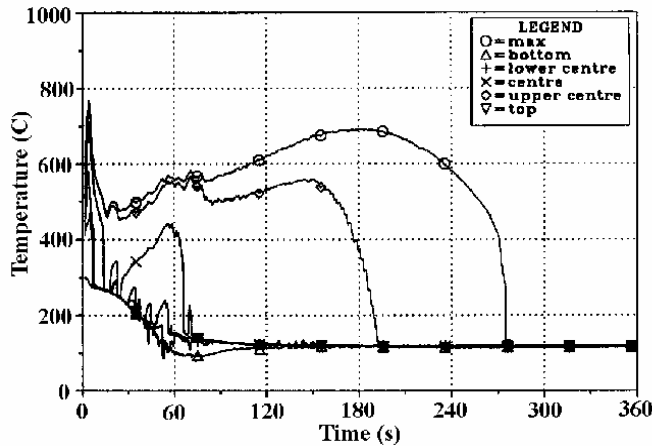

Figure 4.7. Hot rod cladding temperature during a simulated large break loss off cooling accident supposed to take place at full power production at a pressurised water moderated nuclear power plant in the beginning of fuelling cycle.

The analysis simulations using the uprated power level included large break and small break loss of coolant accidents, Anticipated Transients Without Scram (ATWS), primary to secondary leakages, several different pump trips, line breaks, blackouts and valve malfunctions etc. Figure 4.7 depicts the cladding temperature of the hottest fuel rod in the reactor core during a Large Break Loss of Coolant Accident (LBLOCA).

APROS proved to be such an excellent tool for safety analysis that Fortum Nuclear Services Ltd. currently is doing practically all the safety analyses with APROS. The earlier major tool RELAP5 code has a role in assessing the APROS analysis results. The power uprating project was successfully concluded in 1999. The Loviisa plant is now operating at uprated power level producing on average 50 MWe extra power per unit. Depending on the price of electricity this translates into additional revenue of 10–15 M€/year and unit (Sim-Serv 2005).

4.12 Solution Data View

The *Solution Data View* is presented in detail in Appendix B. It is very complementary to the solution framework presentation in this chapter. It manifests the author's proposal on the data structures for implementation. The data structure and the service-oriented operations between the data sets are concluded in Figure 4.7.
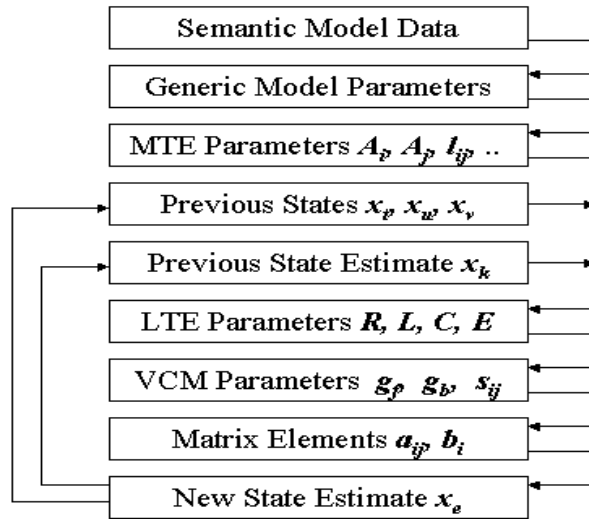

Figure 4.7. The layered data structure enables to easily distinguish distinct data sets and relevant operations between the data sets. The Mechanistic Transition Equation (MTE) parameters stand as a basis for further calculation of dynamic coefficients for Linear Transition Equations (LTE), Versatile Companion Models (VCM) and matrix equation elements.

In this chapter the author has described his original developments of a *Unified Solution Framework* making use of the specification framework presented in previous chapter. The author presents his approach to flatten down the whole model specification to a linear system graph and subsequent re-organisation of the graph to suite a hierarchical table driven solution scheme. In this context the author also publishes for the first time his earlier developments with regard to the implementation of sparse matrix solution of the equations arising from the versatile companion model.

The author has on purpose made use of implementation-neutral structured tables for the specification. A large variety of data organisation and software implementation technologies have been applied since the dawn of computerized dynamic simulation. Today, semantic data-bases and service-oriented software architectures are developing rapidly.

5. Guidelines for Software Implementation

5.1 Service-Oriented Architectures

Traditionally simulation software implementations included all functionalities in one integrated software program. This approach is still in use, and it is very efficient in embedded monolithic applications. The increased use of Dynamically Linked Libraries (DLL) in present implementations promotes easy customisation and replacement of even smaller separate parts of a specific program. The distribution of more comprehensive tasks to several separate programs has been made applying proprietary client/server architecture implementations, like in APROS. The unified solution framework is presented by the author presuming that the software is divided into suitable distributed large grain tasks in a Service-Oriented Architecture (SOA) as follows: The simulation engine, the modelling engine, the component gallery, the model repository, the experiment repository, the design graphics and the monitoring graphics. Standardised interfaces between the separate modules as well as to external software are requested. In addition, the author presumes that the fine grain tasks e.g. between the data structures inside the simulation engine are implemented using SOA principles, as well. Considerable benefits are expected considering the maintainability and re-use of the separate components. For instance, the user interface can be updated separately without the need to change any other modules. The end user may connect his corporate standard user interface to the system, as well, subject that it is SOA compatible.

Next generation of software application architectures must address the reality that design, production, maintenance as well as business processes already operate cross the application boundaries. Applications must be enabled to operate across the artificial boundaries of disparate applications that need to work together to support other applications. Furthermore, there is a huge amount of diverse application software already installed and in operation. A successful architecture will need to support both integration and extension of old applications to give companies the full potential of their current software investments. With all of these capabilities, the new architecture will initially be used to pull together diverse applications to create an effective combined application. Eventually, the next generation of software applications will also

embrace these architectural capabilities in the applications themselves. The operational processes in industry are increasingly supported by the underlying IT infrastructure and relevant applications. Two application paradigm shifts have arisen since the eighties: First the evolution from mainframe to client/server architectures and now the further development to Web-based application architectures. All leading vendors such as IBM, Microsoft, SAP, BEA, Sun, HP and Oracle believe it is important to quickly complete the transition from large, monolithic client/server architectures to versatile service-oriented architecture. Web services are going to be the critical ICT infrastructures for the next 20 years (Wiehler 2004).

5.2 Vertical and Horizontal Connections

Modern ICT already enables the establishment of virtual working groups integrating experts all over the world. In the same way, it shall be possible for the users of advanced simulation tools to ubiquitously operate the graphics interfaces for model development and monitoring of simulation results, without knowing exactly where in the network the component galleries, model repositories, modelling engines or the simulation engines actually reside. It shall be possible for several users to concurrently develop separate parts of a larger model. The system shall facilitate both for separate test of partial models and integrated tests of a large model. Model component galleries and repositories of integrated models shall be available for the developers, over either Intranet, Extranet or Internet as indicated in Figure 5.1.

Standards for *Vertical Specification* combined with standards for *Horizontal Communication* are enabling the establishment of *Virtual Engineering Service Centres*. The intention is that the graphics software shall be very thin and only contain the information seen on the active screens. It seems that an XML type of formality is persistent enough for long-term future re-use regarding the interactive vertical model specification from the graphics to the modelling engine and further to the simulation engine. The OPC standard, on the other hand, facilitates for easy configurable horizontal communication for presentation of simulation results on-line, for communication between simulation engines, or for connections to real or virtual control systems.
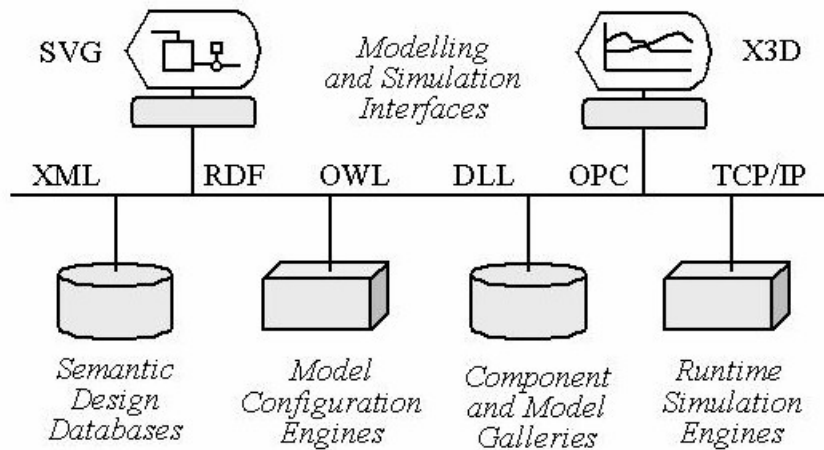
Figure 5.1. A distributed simulation environment providing for remote services.

5.3 Connecting to CAD Systems

An interesting question relates to whether it is possible to transport process specifications from intelligent PI diagrams or 3-D CAD systems to the modelling engine. A suitable set of suitable generic component models accepting the geometric information needs to be readily available. A small number of simple geometric elements like pipes, pumps and valves form the main part of the 3-D specification work. To build up different kinds of tanks, a set of cylindrical, spherical and conical building blocks, both horizontal and vertical, are needed. The important issue is that the intelligent CAD drawing knows which elements are connected, like two pipe ends, and not just happen to be situated close together. From the modelling engine point of view it is required that this kind of connection information can be used to e.g. assemble tank elements to a complete tank model, enabling inlets and outlets at required positions.

In the longer run, web-based graphics tools need to be considered for CAD like model specification and simulation control. The Scalable Vector Graphics (SVG) standard, issued for two-dimensional applications, has reached a large take up from web software developers (SVG 2003). The X3D extensible 3D

graphics proposal, still under development is expected to provide for standardised three-dimensional web applications in near future (X3D 2003).

5.4 Semantics and Ontology for Process Information

The specification data of a large simulation application is can persistently be stored in a single XML file containing only ASCII characters. For efficient elaboration of the data, it is suggested to be inserted into a core resident hierarchical database structure. The test implementation of the simulation engine includes a dedicated XML parser interfacing to this kind of database. In order to enlighten the issue a short XML specification script is presented in Figure 5.2, showing how to modify a single parameter value of a compound node specification in the model repository.

```
<DataRepository  Operation="Modify">
    <drName>My_Project</drName>
    <ComponentModel>
        <cmName>My_evaporation_plant</cmName>
        <CompoundNode>
            <cnName>Supply_tank_1</cnName>
            <ParameterRecord>
                <prName>Total_volume</prName>
                <prUnit>m3</prUnit>
                <prValue>6.7D01</prValue>
            </ParameterRecord>
        </CompoundNode>
    </ComponentModel>
</DataRepository>
```

Figure 5.2. XML script for updating the supply tank volume.

A data structure for different subject and domain specific objects with related attributes and functionality has bee laid out, a kind of ontology. Ontology defines the terms used to describe and represent an area of knowledge. Ontology

includes computer-usable definitions of basic concepts in the domain and the relationships among them. These definitions enable to encode knowledge within a domain and knowledge that spans domains and make that knowledge reusable. World Wide Web Consortium (W3C) has issued the Resource Description Framework (RDF) and the Web Ontology Language (OWL) recommendations. They are semantic web standards that provide a framework for asset management, enterprise integration and the sharing and reuse of data on the web (W3C 2004). The standard formats for data sharing span application, enterprise, and community boundaries – all of these different types of "user" can share the same information, even if they don't share the same software. Now the semantic web emerges as commercial-grade infrastructure for sharing data and is supposed to strongly compete with established knowledge and databases.

In order to facilitate easy interfacing of present or future engineering tools it is important that a comprehensive set of open source accessing software tools are made available.

Research and development efforts are presently focused on semantic process design repositories. The concern is that also the needs from the point of dynamic modelling and simulation are taken into account when implementing domain and component specific information. This is needed to enable modelling and simulation to become an integrated part in both the design and operation of industrial processes.

In the Scandinavian countries there are strong enterprises engaged in the design and use of industrial processes especially in the fields of energy, pulp and paper, and minerals. Suppliers of automation systems, consulting services and engineering software from Scandinavia operate worldwide, as well. Concerned Scandinavian actors should join and assign necessary efforts to the specification and standardisation efforts urgently needed in respective domain. A sincere take up of ICT technologies in the design and business processes might be the key to success.

5.5 Fast Running Code

5.5.1 Speeding-Up by Parallel Processing

Optimisation of calculation speed is an important issue in real-time simulation of extensive process models. Efficient computer database organisation was already in the 1970's an important issue for international research (Martin 1977). In order to speed up the calculation in the first full-scope training simulators, the complete real-time database needed to reside in the computer's semiconductor memory, which in the 1980's still was a very limited resource. The use of memory needed then to be utterly optimised. Fortunately, computer memory is nowadays very affordable and it can be used to an extent that is optimal to increase the speed. Fast serial connection of computers, enables efficient *Coarse Grain Parallelism* by simulation of suitable large subprocesses with only few weak interconnections on separate computers. Several processors with joint memory on the same computer provide for *Medium Grain Parallelism* processing independent subroutines or threads in parallel. *Fine Grain Parallelism* can be achieved by pipelined vector processing of do-loops empowered by parallel fetching of required data from memory in advance to on-chip registers, which in fact also speeds up scalar processing. To simplify the task of an optimising compiler, alternative processing routes is resulting from if-checks must be avoided as far as possible. Pre-fetching of all possibly needed data is not economical. As fetching times are long compared to processing frequency, data organisation in memory is still crucial. If concurrently needed data resides in adjacent memory locations, it is also possible to take advantage of cache-memory services for speeding up memory access. A relevant interesting issue is: Should the sparse matrices be stored in vectors or records (Houbak 1981). Considering the efficient computer memory management of adjacent variables, the record option is preferable for data accessed by code that is difficult to vectorise such as sparse matrix factorisation.

5.5.2 Vectorised Code Enhances Memory Management

To benefit from computers that support vectorisation, the variables are as far as possible allocated to arrays in a consequent order to allow for simple addressing. The do-loops shall not include any if-checks or other alternative paths. The

Pascal code shown in Figure 5.3 operates with only two vectors, vector **R** including all floating point values, and vector **p** including all required integer addresses to fetch distributed content of the **R** vector. The starting address for the consequent values in the value vector to be calculated is denoted by **r1** and the length of the vector to be updated by **len**. The parameters **p1, p2** and **p3** stand for the starting addresses in the pointer vector for access to the distributed parameter values.

```
for i:= 0 to len-1
    do R[i+r1]:= R[i+r1] + R[p[i+p1]]*R[p[i+p2]] + R[p [i+p3]];
```

Figure 5.3. Sample do-loop with typical indirect addressing.

In practice, if the do-loop length is known at compile time and it is pretty short, some optimising compilers just write the equations with pre-calculated addresses.

5.5.3 Pre-Calculated Addresses Enable Optimisation

The accelerated speed of modern processors has resulted in a situation whereas the present core memory look-up times could be compared with disk memory look-up times in the old times. For improved efficiency fast intermediate cache memories are used. Modern compilers are provided with parallel look-ahead optimising capability to arrange for search of required data from core memory in ahead. They are however unable to cope with code including frequent if-checks and accordingly many possible future paths. It is often also impossible for the compiler to know at compile time the lengths of possible vectors. Indirect addresses require multiple fetches at run-time. If an intelligent code generator is used that instead of performing the calculations writes the resulting code with direct indexing to a file, fast running code can be achieved, which is very easily optimised for modern processors. Figure 5.4 shows an example of such generated code. It can be used in a run-time simulation engine with a fixed model specification included. Whence computer memory even nowadays is

much more affordable than speed, this is an attractive alternative for use of models in embedded system, for control or prediction.

$$R[1037] := R[1037] + R[2238]*R[42164] + R[7544];$$
$$R[1038] := R[1038] + R[8349]*R[38145] + R[8867];$$
$$R[1039] := R[1039] + R[5795]*R[67696] + R[1578];$$

Figure 5.4. Accelerated code with pre-calculated addresses.

Gained experience regarding solution of sparse matrices with pre-calculated addresses indicates a reduction of execution time with 60% or more. The efficient use of cache memories play an important role in such tests. A basic requirement is that the whole program code as well as the data areas shall reside in semiconductor memory during the test. The pre-calculation of addresses results in increased program size, which needs to be considered. For reasonably small integrated networks, in the range of 100 nodes, the increase was about 200%. Memory size is presently, however, not limiting factor. Compilers for future processors with architectures increasingly supporting on-chip parallelism are supposed to efficiently optimise this kind of code.

5.5.4 Easily Transportable Software

In the previous presentation, no indications have been made regarding preferred choice of databases, compilers or operation systems. The graphics interface seems though for the moment best fitted for object-oriented implementation. The hierarchical structure of the specification data implies that it fits in an object-oriented repository as well as in a relational database. It could, however, be questioned if well structured data needs to be put into proprietary databases. The capacity and speed of modern mass storage devices allow for storage of huge amounts of data in easily accessible XML based text files. Reference is made to the world wide web.

The simulation engine requires a computer programming language that has access to efficient mathematical routine libraries, and is implemented with very carefully optimised compilers. For the simulation engine it is beneficial, if the compiler can produce optimised code for the specific hardware available from time to time. The instant simulation database needs to be in core and provide for an indexed access to all variables. High performance is achieved if mathematical algorithms, data organisation, software compiler, operation system, and computer hardware are interacting seamlessly. Easy transportability to new hardware and operation system platforms is a crucial requirement.

5.5.5 Programming Methodologies

In order to stimulate the planning of next implementation of the VCM approach a short review is made regarding different previously applied methods to carry out the required functionality of a simulation system for dynamic process models.

- *General Purpose Programming Languages,* either functional or object oriented, are still used for direct programming of simulation models. The programs usually comprise a main program and a set of subroutines. Intrinsic arithmetic subroutines that are supplied with the compiler are frequently used. Sometimes external mathematical libraries are called. The edited source code needs to be compiled and linked before executing the program. This applies for each change made in the model code.

- *Continuous System Simulation Languages (CSSL)* were introduced with specific notations for derivation and integration that are required for solving of dynamic model equations (Augustin et al. 1967). Typical CSSL based functional simulation language codes are first pre-compiled to some general purpose programming language before subsequent compiling and linking. This tends to make the model development process slow.

- *Artificial Intelligence Software* based simulation tools were the targets for research efforts in the 1980's. The Interval Simulation and Reasoning

(ISIR) software represents a hybrid-modelling environment for combined differential equation and discrete model based reasoning (Välisuo 1994). The model specification was written in a Prolog-like language.

- *Interpretation Based Simulation* software platforms that not require compilation of the model specification have been made available. The Direct Executing Simulation in Real-time (DESIRE) language code is interpreted during the execution (Korn & Korn 2003).

- *Object Oriented Simulation Language* developments were initiated in accordance with the general trends in software development, by the *Modelica* design group (Elmqvist et al. 1997). The development started from the robotics background. Presently component libraries are under development both for electrical and hydraulic networks.

- *Table Driven Simulation* platforms have been developed such as the Dynamic Network Analysis (DNA) simulator (Lorenzen 1995). It makes use of the Nordsiek formulation for temporal discretisation and the Newton method with a numerically calculated Jacobean for solution of the resulting nonlinear equation system.

According to the proposed architecture in this thesis, the application developer does not need to invoke compiling and linking. The modelling engine reads the process structure based specifications from a batch file repository or accepts detailed on-line modifications from a design graphics user interface. The modelling engine supplies the simulation engine with relevant input values and indexes to enable service-oriented solver functionality. An extensive set of elementary physical mechanisms is pre-programmed in the simulation engine. Only one instance of each type of equation is needed. During the simulation, the relevant transition equations and mathematical solvers are applied as services to the data structures. Thus, the process structure based application model specification does not need to include any equations.

6. Discussion and Conclusions

6.1 Items Identified for Further Development

A walk through from structure related process specification techniques to efficient numerical solution methods has been made. It is emphasized that several issues still need deepened consideration before commencing final implementation work. The next step would be to select a software platform and develop suitable prototypes for dedicated studies and further refinement. In the following, the author has selected some specific issues for discussion.

- **Knowledge base for design synthesis.** Whence elaborating the versatile companion model related framework for the streamlined specification and solution of process simulation models as presented in this publication, there have been numerous, sometimes conflicting, challenges to comply with. An implementation programmer getting familiar with the architecture requires openness and transparency, enabling him make optimal use of the resembling features of his software development environment. On the other hand, the engagement of an application user that only occasionally uses the tool requires that all features and options that not are necessary for his application work to be hidden. Automated model generation from semantic process design databases is even more challenging. Traditionally, a successful spatial discretisation of an industrial plant process, including specification of suitable control volumes, manual tearing up of very weak connections, and combination of strongly connected variables, has required very comprehensive practical experience, in addition to very dedicated physics, mathematics and even programming knowledge. The direct integration with design databases requires that implicit knowledge of the experienced application user needs to be made explicit and included in the knowledge base of the system. The unified system is designed to enable automatic support of relevant engineering rules at appropriate stages of input data preparation. However, the establishment and use of the required knowledge base is an important issue for further research and development.

- **Combination of homogeneous zones.** The information contained in the coefficients of the companion models are used for optimisation of the solution by automated tearing and combination before construction of the separated matrix equations to be solved. Further, the solution order in each arising sparse matrix equation is optimised in advance before construction of the matrix to avoid the need for fill in of new elements and pivoting during the factorisation. This is a very important advantage compared to traditional methods. A theory on how to efficiently combine partitioned zones running with independent time-step control is definitely an interesting issue for further research.

- **Concatenated monotonic regions.** The developed solver system facilitates for automated non-linearity error corrections. Whilst including physical mechanisms into a new versatile companion model branch type, the application programmer has access to a service for introduction of new monotonic regions related to relevant nonlinear parameters and correlations. It seems that a suitable methodology has been found that can handle the traditional ambiguity of nonlinearities. It works with such correlations that can be made up of monotonic regions. Can all nonlinear piecewise continuous dependencies be described with concatenated monotonic regions? Further research is needed.

- **Empiric Data Reconciliation.** Access to new methodologies and tools for automated reconciliation of measured data e.g., for specification of empiric correlations is strongly called for. Combination of data reconciliation and principal component analysis, two recognised statistical methods used for plant monitoring and fault detection, has been proposed for increased efficiency (Amand et al. 2000) in on-line applications. It seems that measured values would require additional properties attached, such as: unit, time, validity range, sensor accuracy, delay function and noise level. With access to such information, the quality of further calculated values could be followed up accordingly.

- **Mitigation of Round-Off Errors.** In the early days of the computers, scaling of variables was successfully applied to secure a reasonable accuracy of the calculated results. Now, even if computers operating with double precision floating-point arithmetic of 64-bit world length

have reached the desktop, the round-off errors are still out of on-line control. As a specific remedy, iterative refinement of ill-conditioned matrix equation solutions has been applied (Wilkinson 1994). A check of all the included equations might be needed. Sometimes the solution order has a significant influence on the accumulated error, however, an optimising compiler might change the indicated order.

- **Implementation Issues.** Requirements arising from the intentions to use virtual plant models as knowledge repositories during the whole process life cycle have been presented. Interconnections to CAD, CAE, PDM, PLM, ERP, MES and DCS systems have been considered as very essential for the take-up of simulation. Networking, specialisation and outsourcing is providing for new business opportunities. Trusted modelling and simulation support centres can offer to develop, store and update complete integrated virtual models of their clients' processes. Different stakeholders involved with design, construction, operation, trouble-shooting, training, maintenance, logistics or trading can make use of this kind of support. The unified system framework has been designed taking into account the evolving software technologies of web services and service-oriented architectures (SOA 2005).

The challenge has been to write a specification of a unified system at such a generic level regarding the algorithms and specifications, that neither a selection of operating system, computer architecture, nor programming language has been required to do in advance. There is no doubt, that an implementation in a specific environment initiates many new interesting questions to be solved.

6.2 Conclusions on the Functionality

The motivation for the theoretical developments completed by the author originates from a request to solve rapidly and correctly enough such extensive models that is needed for the dynamic simulation of complete industrial plants in sufficient detail and at least in real-time. The process models in focus include both thermal, hydraulic, chemical and nuclear phenomena. In addition, models for automation and electrical systems are considered. The necessity to integrate multidisciplinary models has influenced the development of the mathematical

solvers. The central contribution described in this thesis encompasses the versatile companion model with relevant equivalent representations and the closely coupled sparse node matrix equation solver. The approach has been implemented into several commercial software platforms long before the advent of object-oriented, component-based or service-oriented software technologies. Numerous successful applications in many different domains have been referred to in Chapter 2.

- **Versatile Companion Model.** The author has independently extended the electrical equivalent circuit principle introduced e.g. by Norton and Thevenin to accomplish the versatile companion model that is easily applicable for calculation of the dynamics of thermal hydraulic networks. The versatile companion model seamlessly supports the construction of relevant sparse node matrix equations. The versatile companion model is described in Section 3.4 from the structured graph and the static equation point of view, in Section 3.5 from the dynamic equation point of view, and in Section 3.6 from the nonlinear mechanistic equation point of view.

- **Sparse Node Matrix Equation Solver.** The author has independently applied the bi-factorisation method introduced by Zollenkopf making use of linked list matrix element storage architecture and sparse matrix techniques to develop a sparse node matrix equation Solver. Both symmetric and nonsymmetric node matrices as arising from thermal hydraulic circuits are supported. The sparse node matrix solver is described in Section 4.6 including the construction, factorisation, solution and the correction of nonlinearity errors.

- **Hard Coded First Implementation in Full-Scope Simulators.** The author developed the versatile companion model and relevant sparse node matrix equation solver with the primary aim to speed up the calculation of the extensive pipe network models of the full-scope training simulator of the Loviisa nuclear power plant. They were found to conform to the requirements on limited memory usage, computational efficiency, accuracy and robustness to be used in such a simulator (Juslin 1983a). The first applications were hard coded into existing software environments and replaced accordingly previous code. Any model developments in such an environment require sufficient understanding of the implementation to avoid

introduction of inconsistencies. Several full-scope training simulators based on this first implementation of the solution framework are still in daily use.

- **Table Driven Second Implementation in APROS.** Later on, the author was responsible for the concerted development of the APROS modelling and simulation platform (Juslin et al. 1988). APROS makes use of the second-generation implementation of the solution framework. The software developed is in worldwide use. It is supplied as binary code in accordance with valid license agreements. The configuration of the modelled networks can be made without any programming or compilation efforts by either using a dedicated graphics interface or a native specification script.

- **Verification and Validation.** The full-scope training simulator applications have all been validated against measurements from the target plants. The requirement is that an experienced operator not shall find any differences from the performance of the real plant that could impair the training purposes. The performance of each new version of the APROS code is extensively validated with a sequence of simulated experiments that are compared to measurement recordings from test equipments or to calculations made by other codes (Ylijoki & Norrman 2004).

- **Proven Modelling and Simulation Technology.** The large number successful applications of these implementations talk for themselves. Evidently the solution architecture works. A large number of publications, mostly focusing on applications, have been made during the years by the author, by his colleagues, and by the customers. However, the versatile companion model approach, the implementation of sparse node matrix solvers, as well as the implemented data architectures have not previously been published because of competitive reasons.

- **Unified Solution Platform for Semantic Models.** Based on all previous experiences, the author has now independently concluded the novel specification of the third generation implementation architecture as presented herein. The openness of the new architecture shall enable the user to graphically access even the lowest level of model specification, that is, the separate mechanisms of the versatile companion model branch. The goal has been to specify a unified data and solution framework that easily can make

use of such hierarchical model specifications that automatically can be extracted from future semantic plant models. However, the final service-oriented software implementation work is not within the scope of this thesis.

The correct performance of the already implemented simulation engines has obviously been confirmed. The experience gained has been gathered by the author and herewith concluded into a proposal on novel software architecture for future needs of ubiquitous computing. The automated simulation model construction principles resemble well the evolving semantic specification standards for process plants. The unified architecture is considered to have all possibilities to form a basis for an open source modelling and simulation software platform needed to accomplish future virtual factory initiatives.

References

ABB, 2005a. Modern Control Solutions, Training Simulators for Gas & Steam Turbines, Combined Cycles, Desalinations and Thermal Power Plants. ABB Utilities GmbH, Mannheim, Germany, site http://search.abb.com/library/ ABBLibrary.asp?DocumentID=9AKK100580A0458&LanguageCode=en& DocumentPartID=&Action=Launch visited March 30, 2005, 4 p.

ABB, 2005b. Modern Control Solutions, Training Simulator for Turbine and Unit Controls. ABB Utilities GmbH, Mannheim, Germany, site http://library.abb.com/GLOBAL/SCOT/scot221.nsf/VerityDisplay/C4D2290F41 2A53C6C1256E840027C9A4/$File/PT_ModConSol_Sim_TurbUnitC_E_Rev1. pdf visited March 30, 2005, 4 p.

Al-Falahi, S., Hänninen, M. and Porkholm, K. 1995. Analysis of a Small Break Loss-Of-Coolant Accident of Pressurized Water Reactor by APROS. Seventh International Topics Meeting on Nuclear Reactor Thermal-Hydraulics (NURETH7), Saratoga Springs, New York, USA.

Amand, T., Heyen, G. and Kalitventzeff, B. 2000. Plant monitoring and fault detection: Synergy between Data Reconciliation and Principal Component Analysis. LASSC, University de Liege, Belgium, http://www.ulg.ac.be/lassc/ bibli/AmandHeyen-2000.pdf, visited December 30, 2004, 6 p.

Augustin, D.C., Fineberg, M.S., Johnson, R.N. Sanson, F.J. and Strauss, J.C. 1967. The SCi Continuous System Simulation Language (CSSL). Simulation, 9 pp. 281–303.

Branin, F.H. 1966. The Algebraic-Topological Basis for Network Analogies and the Vector Calculus. Symposium on Generalized Networks, Brooklyn, New York, pp. 453–491.

Brenan, K.E., Campbell, S.L. and Petzold, L.R. 1989. Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. Elsevier Science Publishing Co., New York, 210 p.

Broenink, J.F. 2004. Numerical methods for complex ODE / DAE systems, University of Twente, EE Department, Enschede, Netherlands, http://www.ce.utwente.nl/dynsys/NumMethods.pdf, visited November 5, 2004, 21 p.

Bärman, R., Juslin, K. and Laakso, R. 1993. Application of an Object Oriented Approach to Mechanistic Modelling of a Distillation Plant. SIMS'93, 35th Annual Meeting of the Scandinavian Simulation Society. Kongsberg, Norway, Trondheim: SINTEF, 1993, 8 p.

Bärman, R., Juslin, K. and Laakso, R. 1995. An Advanced Dynamic Simulation Environment for Design and Analysis of Chemical Processes and Their Control Systems. Collection of Nordic Process Control Papers, Nordic Process Control Workshop VI, Åland College of Engineering, Mariehamn, Finland, Åbo Akademi, Process Control Laboratory, Turku, Finland, 1995, 7 p.

Carson, J.S. 2002. Model Verification and Validation. Proceedings of the 2002 Winter Simulation Conference, SCS, La Jolla, USA, pp. 54–58.

Comer, D.J. 1971. Computer Analysis of Circuits. International Textbook Company. College division, London, 351 p.

Corys 2005. Alises, the fully integrated modeling and simulation environment. CORYS T.E.S.S., Grenoble, France, site http://www.corys.com/technologie /alices.php visited March 19, 2005.

Courant, R., Friedrichs, K.O. and Lewy, H. 1967. On the Partial Difference Equations of Mathematical Physics. IBM Journal of Research and Development 11, pp. 215–234.

Dahlquist, G. 1963. A Special Stability Problem for Linear Multistep Methods. BIT, 3, pp. 27–43.

Douzdouzani, A., Paice, A., Gomez, M.M. and Periago, J.M.M. 2003. Long Distance Simulation for Dulces Nombres CCGT. Modern Power Systems, August 2003, pp. 45–48.

Edwards, A.R. and O'Brien, T.P. 1970. Studies of phenomena connected with the depressurization of water reactors. Journal of the British Nuclear Society, 9(1970), pp. 125–135.

Elmegaard, B. and Houbak, N. 2001. Handling Discontinuities in Dynamic Simulation of Energy Systems. Proceedings of ECOS 2001, July 4–6, 2001, Istanbul, Turkey, pp. 895–904.

Elmqvist, H., Broenink, J., Brück, D., Ernst, T., Fritzon, P., Jeandel, A., Juslin, K., Klose, M., Mattsson, S.E., Otter, M., Sahlin, P., Tummescheit, H. and Vanhgheluwe, H. 1997. Modelica – A Unified Object-Oriented Language for Physical Systems Modeling. http://test.modelica.org/documents/Modelica1.pdf [referenced 29.8.2004]. Modelica Association, 73 p.

Fishwick, P.A. 1998. A Taxonomy for Simulation Modelling Based on Programming Language Principles. IIE Transactions, Vol. 30, pp. 811–820.

Gear, C.W. 1971. Numerical Initial Value Problems in Ordinary Differential Equations. Prentice-Hall Inc. Englewood Cliffs, New Jersey, 250 p.

George, A. and Liu, J.W. 1981. Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall Series in Computational Mathematics, Englewood Cliffs, New Jersey, 324 p.

Gockenbach, M.S. 2003. Newton's Method. http://www.math.mtu.edu/~msgocken/ma5630spring2003/lectures/newton/newton.ps, visited June 19, 2004, 8 p.

Goldfarb, C.F. and Prescod, P. 2000. The XML Handbook. Prentice Hall PTR, Upper Saddle River, NJ, 1013 p. ISBN 0-13-014714-1.

GSE 2005. SimSuite Power the high fidelity simulation platform. GSE Systems, Inc., Columbia, Maryland, USA, site http://www.gses.com/powerproducts.html visited March 19, 2005.

Herzog, R., LeFebve, D., Olia, H., Ruchti, C., Svoboda, C. and Wilke, W.-S. 1994. Compact Simulators – An Efficient Way to Process Understanding. VGB Kraftwerkstechnik 74, Number 6, June 1994, 5 p.

Houbak, N. 1981. Use of Records in Sparse Matrix Programs. In: Sparse Matrices and Their Uses, Ed. Duff, I.S., Academic Press, London, pp. 343–348. ISBN 0-12-223280-1.

Hyperion 2005. A NEW Force for Operator Training Systems in the Process and Power Generation. Hyperion Systems Engineering Ltd, Nicosia, Cyprus, Industries, site http://www.hyperion.com.cy/EN/news/press_releases/_d_v678 visited March 20, 2005.

Hänninen, M., Puska, E.K. and Nyström, P. 1987a. Design and Analysis of Nuclear Processes with the APROS. Int. Nuclear Simulation Symposium, Schliersee, FRG, Berlin 1987, Control Data GmbH, pp. 188–204.

Hänninen, M., Juslin, K. and Porkholm, K. 1987b. A Flexible Simulation System for the Design and Analysis of Nuclear Processes. ANS, American Nuclear Society's 1987 Winter Meeting, Los Angeles.

Hänninen, M., Juslin, K., Porkholm, K. and Malkamäki, E. 1988. Simulation of a Conventional Power Plant with APROS. 5th Heat Transfer Summer School, Lappeenranta, August 1988.

Hänninen, M. 1988. A Solution of One-Dimensional Flow and Heat Transfer Processes with Difference Method. Licentiate Thesis, Lappeenranta University of Technology.

IMTI, 2002. Automated Generation of Models for Design and Manufacturing. September 30, 2002. White paper. Integrated Manufacturing Technology Initiative. IMTI Inc. Oak Ridge, TN 37831. http://www.imti21.org/msam/AutoModelGen.pdf.

IMTI, 2003. Modelling & Simulation for Affordable Manufacturing. Technology Road mapping Initiative. V3.2. January 18, 2003. Integrated Manufacturing Technology Initiative. IMTI Inc. Oak Ridge, TN 37831. http://www.imti21.org/msam/MSAM.pdf.

INEL 2005. RELAP5-3D, Idaho National Engineering Laboratories. Idaho Falls, Idaho, USA, site http://www.inel.gov/relap5/relap5-3.htm visited March 21, 2005.

Invensys 2005. Power Simulators. Invensys Systems Inc./Simsci-Esscor, Lake Forest, California, USA, site http://www.simsci-esscor.com/us/eng/simsciSimulators/power/default.htm visited March 20, 2005.

Johnson, D.H. 2003a. Scanning our Past, Origin of the Equivalent Circuit Concept: The Current-Source Equivalent. Proceedings of the IEEE, Vol. 91. No. 5, May 2003, pp. 817–821.

Johnson, D.H. 2003b. Scanning our Past, Origin of the Equivalent Circuit Concept: The Voltage-Source Equivalent. Proceedings of the IEEE, Vol. 91. No. 4, April 2003, pp. 636–640.

Juslin, K. 1973. Numerical Design of a Controllable and Fast Operating Direct Current Converter. Master's Thesis: Helsinki University of Technology, Power Electronics Laboratory. 64 p. + app. 9 p. (in Swedish).

Juslin, K. 1983a. Execution Time Comparison of Linear Equation Systems Solvers. Research Report, Technical Research Centre of Finland, Electrical Engineering Laboratory, Espoo, 8 p. (in Finnish).

Juslin, K. 1983b. Simulation of Power Electric Circuits. SIMS'83, Scandinavian Simulation Society, Annual Meeting, Odense, Denmark, 14 p.

Juslin, K. and Siikonen, T. 1983. Solution Methods for Pipe Network Analysis. IAEA/NPPCI Specialists' Meeting, Nuclear Power Plant Training Simulators, Espoo, 11 p.

Juslin, K. 1984a. Application of Simulation Techniques in On-Line Process Industry. INSKO publications 1978–84: Simulation in Energy Production and Process Industry, Porvoo, Finland, 14 p. (in Finnish).

Juslin, K. 1984b. Simulation of the Electrical Systems of a Power Plant – an Example of Modelling Principles. Joint Finnish and Soviet Union Symposium on Design Principles of Training Simulators for Power Plants, Helsinki, 16 p. (in Finnish).

Juslin, K., Silvennoinen, E. and Karppinen, J. 1985. Experiences on Real-Time Solution of Sparse Network Equations. IMACS Transactions on Scientific Computation 1985, Vol. 3, Modelling and Simulation in Engineering, North-Holland, Netherlands, 6 p.

Juslin, K. and Silvennoinen, E. 1986. Real-Time Solution Approach for Sparse Network Equations. Espoo, Finland, Technical Research Centre of Finland, Research Notes 615, 38 p. + app. 5 p. ISBN 951-38-2050-5.

Juslin, K., Ranne, A. and Silvennoinen, E. 1987a. Operation of District Heating Network Supported by a Dynamic Simulation Model. SIMS'87, 29th Annual Meeting, the Scandinavian Simulation Society, Sonderborg Technicum, Sonderborg, Denmark.

Juslin, K., Hänninen, M., Puska, E.K., Kurki, J. Nyström, P. and Porkholm, K. 1987b. Application of an Advanced Process Simulator. IAEA, Specialists' Meeting on Training Simulators for Nuclear Power Plants, Toronto, Canada.

Juslin, K. 1987. APROS Advanced Process Simulator for Efficient Analysis of Dynamical Systems and Processes. European Congress on Simulation, Prague, September 21–25, 1987.

Juslin, K., Silvennoinen, E., Kurki, J. and Porkholm, K. 1988. APROS: An Advanced Process Simulator for Computer Aided Design and Analysis. 12th IMACS World Congress on Scientific Computation, Paris, France.

Juslin, K., Mattila, L. and Kurki, J. 1989. Simulation of Nuclear and Conventional Power Plants with APROS. Beijing International Simulation Conference, Beijing, China, 5 p.

Juslin, K. and Kurki, J. 1989. The Use of APROS for Control System Design of a Fossil Fired Power Plant. EPRI Conference on Power Plant Controls and Automation, Miami, Florida, USA, 11 p.

Juslin, K., Lahdenperä, E. and Kaijaluoto, S. 1990. Dynamic Simulation of Chemical Processes Using the CHEDYN Package. Finnish Society for Automatic Control, 1990 Annual Meeting, 5 p. (in Finnish).

Juslin, K. 1990. Application of the Structured Query Language of APROS for Specification of Industrial Processes. The joint Finnish-Soviet Symposium on Information Technology and Economic Modelling, Helsinki, Finland, 13 p.

Juslin, K., Kaijaluoto, S., Kalitventzeff, B., Kilakos, A. and Lahdenperä, E. 1991. An Equation Oriented Software Package for Dynamic Simulation of Chemical Processes. COPE'91, Barcelona, Spain, 5 p.

Juslin, K. and Lilja, R. 1991. Qualitative 3-D estimator of burning process. 2nd Int. Symposium on Coal Combustion. Beijing, CN, 7–10 Oct. 1991. Tsinghua University Beijing. Beijing, 6 p.

Juslin, K. and Tuuri, S. 1992. Dynamic Simulation of a Recovery Boiler Using the APROS Simulation Program. TAPPI Proceedings, 1992 International Chemical Recovery Conference, Seattle WA, USA, pp. 293–303.

Juslin, K., Kondelin, K. and Suutari, J. 1993. Graphics Based Modelling and Simulation of a Combined Cycle Plant. ESS'93, European Simulation Symposium. Delft, the Netherlands, San Diego CA: The Society for Computer Simulation, 1993, 2 p.

Juslin, K. and Niemenmaa, A. 1994. Dynamical Simulation of a Multi-Effect Black Liquor Evaporation Plant. Proc. of SIMS'94, Annual Meeting of the Scandinavian Simulation Society. Stockholm, Sweden, Computer Solutions Europe, 1994, 4 p.

Juslin, K. and Pollari, H. 1994. Mass and Energy Dynamics of a Displacement Pulping Process. ESS'94, European Simulation Symposium, Istanbul, Turkey. San Diego CA: The Society for Computer Simulation, 1994, 5 p.

Juusela, A. and Juslin, K. 1976. A Nonlinear Simulation Model of a BWR Nuclear Power Plant. In: Technical Research Centre of Finland, Electrical Engineering Laboratory Report 20, 83 p.

Kantee, H., Kontio, H., Plit, H., Kallio, H., Savolainen, S., Norrman, S. and Virtanen, E. 1998. Application of Apros Simulation Software in Safety Analyses of Loviisa NPP Power Uprating Project. 6th International Conference on Nuclear Engineering, ICONE-6, May 10–15, 1998, San Diego, California, USA, The American Society of Mechanical Engineers, ASME, 1998, 12 p.

Kantee, H., Kyrki-Rajamäki, R., Miettinen, J., Vanttola, T., Komsi, M. and Tuomisto, H. 1991. Accident Analyses for the Loviisa VVER-440 Reactors. ANS. American Nuclear Society, International Topical Meeting, Safety of Thermal Reactors, Janzen Beach Red Lion, Oregon, USA, pp. 623–630.

Karhela, T., Kuikka, S. and Paljakka, M. 1998a. Application of Web Browser and Software Component Technologies in Operator User Interfaces in Process Simulation: A Case Study on Dynamic Simulation of Rotary Lime Kiln. Proceedings of the EUROSIM'98. Simulation Congress. Espoo, pp. 111–117.

Karhela, T., Lappalainen, J., Peltola, H. and Juslin, K. 1998b. Dynamic Simulation Model of Rotary Lime Kiln. 1998 TAPPI Proceedings. International Chemical Recovery Conference. Hyatt Westshore Tampa, Florida, pp. 1081–1093.

Karhela, T., Paljakka, M., Laakso, P., Mätäsniemi, T., Ylijoki, J. and Kurki, J. 1999. Connecting Dynamic Process Simulator with Distributed Control System Using OPC Standard 1999. TAPPI Proceedings. Process Control, Electrical, and Information Conference. Georgia World Congress Centre, Georgia, pp. 329–337.

Karhela, T., Kondelin, K., Mätäsniemi, T. and Paljakka, M. 2001. Knowledge and Model Gallery for Process Integration. Finnish Society of Automation Publications 24, Seminar Days 01, Helsinki Fair Centre, pp. 55–63. ISBN 952-5183-17-3. (in Finnish).

Karhela, T. 2002. A Software Architecture for Configuration and Usage of Process Simulation Models. Software Component Technology and XML-based Approach. Espoo 2002. Technical Research Centre of Finland, VTT Publications 479. 129 p. + app. 19 p. ISBN 951-38-6011-6; 951-38-6012-4. http://www.vtt.fi/inf/pdf/publications/2002/P479.pdf

Karlsson, T., Jokstad, H., Meyer, D.B., Nihlwing, C., Norrman, S., Puska E.K., Raussi, P. and Tiihonen, O. 2001. The HAMBO BWR Simulator of HAMMLAB, HWR-663, OECD Halden Reactor Project, Report Restricted to Halden Project Use Only, 51 p.

Kelley, S.T. 2003. Solving Nonlinear Equations with Newton's Method, http://www4.ncsu.edu:8030/eos/users/c/ctkelley/www/newton.PS.pdf, visited July 19, 2004, 92 p.

Kondelin, K. and Karhela, T. 2003. Gallery mark-up and query language specification. VTT Industrial Systems, Espoo. VTT Research Notes 2184. 111 p. ISBN 951-38-6117-1; 951-38-6118-X. http://www.vtt.fi/inf/pdf/tiedotteet/2003/T2184.pdf

Korn, G.A. and Korn, T.M. 2003. DESIRE/2000: Fast, Easy Interactive Modeling and Simulation of Dynamic Systems. Neural Networks, and Fuzzy Logic. http://members.aol.com/gatmkorn/ [referenced 29.8.2004].

Kurki, K. and Jokela, P. 1984. Preliminary Assessment on Use of Simulation in Power Plant Design. Report SYA-0195, Imatran Voima Oy, Helsinki, 20 p. + app. 79 p. (in Finnish).

Kurki, J., Kovanen, J., Malkamäki, E., Juslin, K., Hänninen, M. and Leino, J. 1989. An Engineering Simulator for Fossil-Fuelled Power Plant. 7th Power Plant Dynamics Control & Testing Symposium, Knoxville, Tennessee, USA, 9 p.

Laakso, P., Mätäsniemi, T., Karhela, T. and Paljakka, M. 1999. Component Frameworks for Modelling and Simulation. Scandinavian Simulation Society (SIMS) Meeting, Linköping, Sweden, October 18–20 1999, pp. 30–38.

Laitinen, A., Juslin, K. and Silvennoinen, E. 1986. A Simulation Program for Radiator Network Calculation. SIMS'86, the 28th Annual Meeting of the Scandinavian Simulation Society, Gothenburg, Sweden, 9 p.

Lambert, J.D. 1991. Numerical Methods for Ordinary Differential Systems: the Initial Value Problem. John Wiley & Sons Ltd., West Sussex, England, 293 p.

LANL 2005. Transient Reactor Analysis Code (TRAC). Los Alamos National Laboratory, Los Alamos, New Mexico, USA, site http://www.lanl.gov /source/orgs/d/d5/trac.shtml visited March 21, 2005.

Lappalainen, J., Tuuri, S., Karhela, T., Hankimäki, J., Tervola, P., Peltonen, S., Leinonen, T., Karppanen, E., Rinne, J. and Juslin, K. 1999. Direct Connection of Simulator and DCS Enhances Testing and Operator Training. Proceedings of TAPPI Engineering / Process & Product Quality Conference, Hilton Anaheim, Anaheim CA, TAPPI Press (1999), pp. 495–502.

Lappalainen, J., Myller, T., Vehviläinen, O., Tuuri, S. and Juslin, K. 2001. Enhancing grade changes with dynamic simulation. Proceedings of the TAPPI Engineering Conference 2001, San Antonio, Texas, 11 p.

Laukia, A., Komsi, M. and Lilja, M. 1990. Dynamic Simulation in Nuclear Power Plant Design. IAEA Specialists' meeting on Advanced Technologies for Cost Reduction of Water Cooled Reactor Plants, Vantaa, Finland, 5 p.

Leppäkoski, J., Kurki, J. and Juslin, K. 1990. An Engineering Simulator for Peat Fired Power Plant. Finnish Society for Automatic Control, 1990 Annual Meeting, 7 p. (in Finnish).

Leppäkoski, J., Kurki, J. and Juslin, K. 1991. Experience on Simulator Based State Controller Design for Power Plants. 13th IMACS World Congress on Computation and Applied Mathematics, Trinity College, Dublin, Ireland, 7 p.

Leppäkoski, J. 1995. Experience on Verification of Process Control Concept by Simulation. Automation 95, Finnish Society for Automatic Control, Helsinki, Finland, pp. 300–302. (in Finnish).

Lilja, R. and Juslin, K. 1987. Fast Calculation of Material Properties Applied to Water and Steam. Technical Research Centre of Finland, Research Notes 807, Espoo, Finland, 30 p. ISBN 951-38-3036-5.

Lilja, R., Ollikainen, T. and Laakso, P. 2003. Self-access studying environment for control engineering education. The 6th IFAC Symposium on Advances in Control Education (ACE 2003), 16–18 June 2003, Oulu, 5 p.

Lilja, R. and Hämäläinen, J. 1999. Modelling of thermodynamic properties of substances by neural networks. International Joint Conference on Neural Networks (IJCNN'99). Washington, DC, 10–16 July 1999. International Neural Network Society; Neural Networks Council of IEEE.

Lindberg, B. 1971. On Smoothing and Extrapolation for the Trapezoidal Rule. BIT, 11, pp. 29–52.

Lorenzen, B. 1995. Power Plant Simulation. Doctoral thesis, Laboratory for Energetics, Technical University of Denmark, Lyngby. ISBN 87-7475-165-4.

MacFarlane, A.G.J. 1964. Engineering Systems Analysis. Addison-Wesley Publishing Company, Reading, Massachusetts, 272 p.

Mapps 2005. ROSE the Rapid Model Development and Runtime Environment. L-3 Communications MAPPS Inc., Montreal, Canada, site http://www. mapps.l-3com.com/L3_MAPPS/Products_and_Services/Power_Systems_and_ Simulation /Technology/rose.shtml visited March 19, 2005.

Martin, J. 1977. Computer Database Organisation. Prentice Hall, Englewood Cliffs, NJ, 713 p. ISBN 0-13-165423-3.

Mattila, L. 1987. Numerical Simulation of Processes. VTT Research Program 1986–1988, 4th Heat Transfer Summer School, Lappeenranta, August 10–15, 1987, 9 p.+ app. 10 p.

Mattila, L. and Winter, M. 1987. Advanced Simulation Software. Nuclear Europe 1987:11/12, p. 53.

Mitchell, E. 1991. ACSL for Windows: Installation and How to Use. Mitchell & Gauthier Assoc. ISBN 0925649015.

Microfusion 2005. THINK: Thermal Hydraulic Integrated NetworK. Microfusion Engineering Laboratories, Inc., Norcross, Georgia, USA, site http://www.microfusionlab.com/ visited March 20, 2005.

Muetzelfield, R. 2004. Position paper on declarative modelling in ecological end environmental research. Office for Official Publications of the European Communities, Luxembourg, 88 p. ISBN 92-894-5212-9. http://www.decmod. org/documents/dmeer.pdf.

Nappa, M., Karhela, T. and Hurme, M. 2002. Internet-based Equipment and Model Gallery. Proceedings of the ESCAPE-12 conference, Hague, Netherlands.

Nelles, O. 2001. Nonlinear System Identification. Springer-Verlag Berlin Heidelberg, pp. 219–238. ISBN 3-540-67333369-5.

nHance 2005. MMS For Windows: The Modular Advantage. nHance Technologies Inc, Lynchburg, Virginia, USA, site http://www.nhancetech. com/Products.htm visited March 19, 2005.

Niemi, J., Juslin, K. and Hänninen, M. 1989. Multidimensional Flow Calculation in the APROS Process Simulator. International Symposium on Numerical Methods in Engineering, Lausanne.

Niemi, J. and Tommiska, J. 1990. Parallel and Vector Implementation of APROS Simulator Code. 1st Int. Conf. on Supercomputing in Nuclear Applications, SNA'90, Mito, Japan. Japan Atomic Energy Research Institute, pp. 341–345.

NRC, 1996. Human System Interfaces Design Review Guideline. Process and Guidelines, NUREG-0700, Rev. 1, Vol. 1, U.S. Nuclear Regulatory Commission (NRC), Office of Nuclear Regulation, Washington DC, USA.

Ollikainen, T., Heino, A. and Porkholm, K. 2002. Generic Training Simulator for a Combined Cycle Gas Turbine Power Plant. Proceedings of the 43rd Conference on Simulation and Modelling (SIMS 2002). September 26–27, Oulu, Finland, pp. 139–144.

Paljakka, M., Karhela, T., Laakso, P., Tuuri, S., Lappalainen, J. and Juslin, K. 2000. Simulation integrates process and automation development. Automation Technology Review 2000, pp. 52–58.

Paredis, C.J.J., Diaz-Calderon, A., Sinha, R. and Khosla, P.K. 2001. Composable Models for Simulation-Based Design. Engineering with Computers, June 2001, Vol. 17, No. 2, pp. 112–128.

Pasanen, A. 2001. Phenomenon-driven process design methodology. Computer implementation and test usage. Espoo, Technical Research Centre of Finland. VTT Publications 438. 140 p. + app. 26 p. ISBN 951-38-5854-5; 951-38-5855-3. http://www.vtt.fi/inf/pdf/publications/2001/P438.pdf

Patankar, S.V. 1980. Numerical Heat Transfer and Fluid Flow. Hemisphere Publishing Corporation. 197 p.

Peltoniemi, J., Karhela, T. and Paljakka, M. 2001. Performance Evaluation of OPC-based I/O of a Dynamic Process Simulator. Proceedings of the 2001 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), Orlando, Florida, pp. 231–236, SCS, ISBN 1-56555-240-7.

Pigg, J., Tuuri, S. and Juslin, K. 2002. Model for oxygen delignification of kraft pulp. SIMS 2002 Proceedings of the 43rd Conference on Simulation and Modelling. Oulu, FI, 26–27 Sept. 2002. Juuso, E. and Yliniemi, L. (Eds.). Finnish Society of Automation; University of Oulu, Oulu, pp. 166–171.

Porkholm, K. Hänninen, M. and Juslin K. 1988. Application of an Advanced Process Simulator to Power Plant Simulation. EPRI Conference on Power Plant Training Simulators and Modelling, Charlotte, USA, 10 p.

Porkholm, K., Hänninen, M. and Niemi, J. 1991. Simulation of the Feed-water Line Break of Loviisa Nuclear Power Plant. International Conference on Dynamics and Control in Nuclear Power Stations, British Nuclear Society, Westminster, London, United Kingdom, October 22–24, 1991. 7 p.

Porkholm, K., Hänninen, M., Shutov, V. and Ovtcharova, I. 1994. APROS-based Kola Nuclear Power Plant Analyzer. SCS Simulation Multiconference. April 11–13, 1994, Hyatt Regency Aventine, Lajolla, San Diego CA, USA, 6 p.

Porter, B. and Crossley, R. 1972. Modal Control, Theory and Applications. Taylor & Francis Ltd., London, United Kingdom, 230 p.

Puska, E.K. and Juslin, K. 1989. Solution of APROS Nuclear Reactor Kinetics. SIMS'89, Scandinavian Simulation Society Annual Meeting, Bergen, Norway, 12 p.

Puska, E.K., Hänninen, M., Miettinen, J., Virtanen, J., Kurki, J. and Linden, U. 1989. Simulation of Loviisa Power Plant Transients with APROS. Proc. of the Topical Meeting on Advances in Nuclear Engineering Computation and Radiation Shielding, Santa Fe, New Mexico, USA, ANS 1989, pp. 28:1–12.

Puska, E.K. and Porkholm, K. 1991. Plant Analyzer of Loviisa NPP. ANS/ENS International Topical Meeting on Advances in Mathematics, Computations, and Reactor Physics, Greentree Marriott, Pittsburgh, PA, USA, 12 p.

Puska, E.K. 1991. One- and Three-dimensional Nuclear Reactor Core Models for Plant Analyzer. ANS, American Nuclear Society, International Topical Meeting, Safety of Thermal Reactors, Janzen Beach Red Lion, Portland, Oregon, USA, 8 p.

Puska, E.K. and Norrman, S. 1999. APROS BWR plant analyser with 3-D core. CD-ROM Proceedings of the 7th International Conference on Nuclear Engineering, Tokyo, Japan, The Japan Society of Mechanical Engineers. Paper ICONE-7314, 10 p.

Puska, E.K. 1999. Nuclear reactor core modelling in multifunctional simulators. Espoo, Technical Research Centre of Finland, VTT Publications 376. 67 p. + app. 73 p. ISBN 951-38-5361-6; 951-38-5364-0. http://www.vtt.fi/inf/pdf/publications/1999/P376.pdf

Rheinmetall 2005. Power Plant Process Simulation. Rheinmetall Defence Electronics GmbH, Bremen, Deutchland, http://www.rheinmetall-detec. com/index.php?lang=3&fid=1712&action=pd, site visited March 20, 2005.

Rinta-Valkama, J., Välisuo, M., Karhela, T., Laakso, P. and Paljakka, M. 2000. Simulation Aided Process Automation Testing. Proceedings of IFAC's Conference on Computer Aided Control System Design (CACSD), University of Salford, UK, Elsevier Science, pp. 277–280. ISBN 0-08-043660-9.

SAIC 2005. SimPort Training Simulator. Science Applications International Corporation (SAIC), San Diego, California, US, site http://www.saic.com/ products/simulation/simport/ visited March 20, 2005.

Savolainen, J., Tuuri, S. and Juslin, K. 2001. Dynamic model of a disc filter. Proceedings of the SIMS 2001 Conference. Porsgrunn, NO, 8–9 Oct. 2001, pp. 397–406.

Sebah, P. and Gourdon, X. 2001. Newton's Method and High Order Iterations. http://numbers.computation.free.fr/Constants/Algorithms/newton.ps visited July 19, 2004. 10 p.

Shah., M. 2002. The First Full-Scope Training Simulator at Chashnupp-1. Nuclear Engineering International, February 2002, pp. 28–29.

Siikonen, T. 1987. Numerical Method for One-Dimensional Two-Phase Flow. Numerical Heat Transfer 12, 1, pp. 1–18.

Silvennoinen, E., Raiskinen, I. and Juslin, K. 1988. Introduction to APROS Simulation Environment. ESM'88, European Simulation Multi-conference, Nice, France.

Silvennoinen, E., Juslin, K., Hänninen, M., Tiihonen, O., Kurki, J. and Porkholm, K. 1989. The APROS Software for Process Simulation and Model Development. Technical Research Centre of Finland, Research Reports 618, 106 p. + app. 19 p. ISBN 951-38-3463-8.

Silvennoinen, E. 1996. Research and Development Cooperation and Technology Transfer as Strategic Instruments: A General Assessment and a Case Study of Simulation Software Development. Acta Polytechnica Scandinavica, Ma 77, Helsinki, Published by the Finnish Academy of Technology, 154 p. ISBN 951-666-471-7, ISSN 1238-9803.

Sim-Serv, 2005. Loviisa Nuclear Power Plant Uprating Project, Success Stories. Web site http://www.sim-serv.com/success_stories.php of Sim-Serv, World's First Virtual Centre in Simulation, was visited May 5, 2005, 4 p.

Sinha, R., Liang, V.C., Paredis, C.J.J. and Khosla, P.K. 2001. Modelling and Simulation Methods for Design of Engineering Systems. Journal of Computing and Information Science in Engineering, March 2001, Vol. 1, No. 1, pp. 84–91.

SOA 2005. Web Services and Service-Oriented Architectures, Barry & Associates Inc, South Burnsville, Minnesota, USA, site http://www.service-architecture.com/contact.html was visited April 20, 2005.

Solodovnik, E.V., Cokkines, G.J. and Melipoulos, A.P.S. 2003. Utilisation of Automatic Differentiation in VTB Simulation Environment. At web site: http://vtb.engr.sc.edu/pubs/reports/ASS2000c.doc [referenced 30.11.2003].

Soutolahti, H., Iso-Herttua, P., Kortesoja, P. and Juslin, K. 1995. Experience on Dynamic Model Based District Heating Network and Control System Design Evaluation. 5th International Symposium on Automation of District Heating Systems, Otaniemi, Espoo Finland, Nordic Energy Research Programme 1995, Helsinki University of Technology, Espoo, Finland, 6 p.

SVG 2003. Scalable Vector Graphics. The Word Wide Web Consortium. http://www.w3.org/Graphics/SVG/ [referenced 30.11.2003].

Tecnatom 2005. Total or partial scope training simulators, Tecnatom s.a., Madrid, Spain, site http://www.tecnatom.es/english/catalogo/ficha_producto_pag.php?idProducto=226 visited March 21, 2005.

Tervola, P., Lappalainen, J., Rinne, J., Leinonen, T., Peltonen, S., Karhela, T. and Juslin, K. 1999. Bleach Plant Training Simulator Featuring Enhanced Linkage between Simulator and DSC. Proceedings of TAPPI 1999 Pulping Conference, Renaissance Orlando Resort, Orlando FL, October 31 – November 3 1999, Vol. 3, 1031–1045.

Thales 2005. Power Plant Simulation at Thales Training and Simulation, TT&S. Cergy, France, site http://www.thalesgroup.com/all/pdf/PowerPlant.pdf visited March 19, 2005.

Tiihonen, O., Puska, E.K., Virtanen, J. and Kurki, J. 1988. Advanced Simulator for Reactor Safety Analysis. Int. ENS/ANS Conf. on Thermal Reactor Safety, NUCSAFE'88, Avignon, France, Societe Francaise d'Energie Nucleaire, Proc. Vol. 4, pp. 1482–1491.

Tiihonen, O., Porkholm, K., Hänninen, M. and Linden, U. 1991. Experiences of the Loviisa Plant Analyzer Based on the APROS Simulation Environment. IAEA Specialist Meeting on Training Simulators for Safe Operation in Nuclear Power Plants, Balatonfüred, Hungary, 9 p.

Trax 2005. Training Simulators, Trax LLC, Lynchburg, Virginia, the site http://www.traxcorp.com/simulators.html visited March 20, 2005.

Trent, H.M. 1955. Isomorphisms between Oriented Linear Graphs and Lumped Physical Systems. The Journal of the Acoustical Society of America, Vol. 27, pp. 500–527.

Tuunanen, J., Kouhia, J., Purhonen, H., Riikonen, V., Puustinen, M., Semken, R.S., Partanen, H., Saure, I. and Pylkkö, H. 1998. General description of the PACTEL test facility. Technical Research Centre of Finland, Espoo, VTT Research Notes 1929, 35 p. + app. 74 p. ISBN 951-38-5338-1; 951-38-5339-X. http://www.vtt.fi/inf/pdf/tiedotteet/1998/T1929.pdf

Tuuri, S., Juslin, K. and Aalto, H. 1995a. Early Verification of a New Power Plant and Automation Concept during the Pre-Design Project. Automation 95, Finnish Society for Automatic Control, Helsinki, 5 p. (in Finnish).

Tuuri, S., Juslin, K., Niemenmaa, A., Laukkanen, I. and Lappalainen, J. 1995b. Better Integral Control of Paper and Board Process using Dynamic Simulation. Automation 95, Finnish Society for Automatic Control, Annual Meeting, May 3–5, 1995, Helsinki, Finland, 5 p. (in Finnish).

Tuuri, S., Lappalainen, J. and Juslin, K. 2001. Customized dynamic simulator supports process & control engineering at mill state. Automation Technology Review 2001. VTT Automation. Espoo, pp. 42–47.

Wiehler, G. 2004. Mobility, Security and Web Services, Technologies and Service-Oriented Architectures for a new Era of IT Solutions. Publicis Corporate Publishing, Erlangen, Germany, 219 p.

VTT 2005. APROS – the Advanced Process Simulation Environment, Technical Research Centre of Finland, Espoo, Finland, site http://www.vtt.fi/tuo /63/APROS /index.htm visited March 22, 2005.

Välisuo, H. 1994. Model Based Reasoning and Control of Process Plants. Doctoral thesis, Helsinki University of Technology, Espoo, VTT Publications 178, 155 p. + app. 12 p. ISBN 951-38-4416-1.

W3C 2004. World Wide Web Consortium Issues RDF and OWL Recommendations. http://www.w3.org/2004/01/sws-pressrelease [referenced 1.8.2004].

Wagner, W. and Krause, A. 1998. Properties of water and steam. Springer-Verlag Berlin Heidelberg, 354 p. ISBN: 3-540-64339-7.

Warner, S.L. and Goldsman, P. 1996. Delphi 2 by Example, Published by the Que Corporation, Indianapolis. 479 p. ISBN 0-7897-0592-3.

Westinghouse 2005. Full-scope Simulators. Westinghouse Electric Company LLC, Madison, Pennsylvania, USA, site http://www.westinghouse-nuclear.com/C3b3f.asp visited March 20, 2005.

Wilkinson, J.H. 1994. Rounding Errors in Algebraic Processes. Dover Publications, New York, 161 p.

X3D 2003. Extensible 3D Graphics. The Web3D Consortium, Inc. http://www. web3d.org/x3d/index.html [referenced 1.8.2004].

Ylijoki, J. and Norrman, S. 2004. Validation of APROS Version 5.04, Research Report PRO5/7804/04, Espoo, Finland, 131 p.

Yokogava 2005. Power Plant Simulators. Yokogawa Electric Corporation, Tokyo, Japan, site http://www.yokogawa.com/pwr/products/simulators/pwr-prd-sim-001en.htm visited March 20, 2005.

Zollenkopf, K. 1971. Bi-Factorisation – Basic Computational Algorithm and Programming Techniques. Large Sparse Sets of Linear Equations. Academic Press 1971, pp. 75–96.

Appendix A: Specification Data View

General Real-Time Data Management Issues

Core resident data is grouped to state variables, parameters, constants, structural dependencies, temporary data, and interface specifications. There are different requirements imposed on the functionality and structure of the instant in-core database.

The on-line process specification benefits from an object-oriented structure. Is shall be possible to consider changes in the process structure even between the simulation time-step. There is no time for re-compilation and linking of the whole task. On the other hand, the mathematical solvers benefit from a flat structure with a continuous data area, which enables the application of long simple do-loops without internal if-structures. This is the code, which modern compilers and processors can optimize in a very efficient way, applying parallel fetching of required variables long in advance both at chip level and at external cache memory level.

Other requirements arise from the needs of training simulators to specificly store the state variables to disk memory as state snap-shots at certain intervals during the simulation to enable for backtracking and replay functionality. This data is copied in binary format. The full snap-shots need only to be made on request after changes in the model structure. It includes also the information on model parameters, constants and connections.

To facilitate for interactive model specification and simulation control, an ASCII based interface is needed for communication between the simulation engine, graphical user interfaces, and model repositories. The user interfaces shall provide for both model specification and presentation of simulation results. Specified parts of a model instance shall be possible to store and retrieve from such a repository. From the real-time requirements point of view, all variables are included in separate dynamic arrays, such as: Name list, double precision state variables, sparse matrix elements, delay variables, complex state variables, sparse matrix elements, and structured data types for configuration purposes.

Plant Specification Data

There are several application needs for design data repositories. A process component manufacturer wants to publish model data specifications of his products to be easily taken up by design engineers. The specifications can be stored and accessed locally on a file, or it can be published either in intranet, extranet or internet. Different subprocess suppliers in a delivery project can combine component models to comply with their parts of the deliveries. The control system supplier can link the different subprocess model specification data into an integrated model specification repository, and use it for evaluation the functionality of the integrated control concept, or even test the real digital control system configuration with the model. A design data repository can accordingly include a complete set of component specifications from a component manufacturer, such as a pump manufacturer. It can include all relevant subprocess design alternatives, including automation system configuration, from e.g. a boiler manufacturer. The attributes of the data repository are specified in Table A.1.

Table A.1. DataRepository attributes.

drName	Instance name	*drAuth*	User authorisation list entry
drUses	Used repositories list entry	*drHist*	Event history
drIdoc	Instance documentation	*drCM*	*Component Model* list entry

The repository instance name needs to be distinct from other repositories referred to. The linkage to external repositories reduces the need to copy their specifications as a whole or partly to the repository instance in question. It also enables distributed maintenance of the relevant specifications by the most knowledgeable parties, as well as it reduces the need to update many instances.

The repository instance documentation provides a mean for free format documentation end messaging, especially useful in a multi-user workgroup environment. In virtual environments the user authorisation becomes very important, as well as and automated event history on who has done what. The component list entry provides a mean to access the content of the repository.

Flow-Sheet Based Specification

Component Model

Interconnected component models represent the application modelling user's view of the process and its physical relationships. The specification of parameters and interconnects could be done directly into a data base, but it is preferable to make use of intelligent process and automation diagrams with explanatory component symbols and related parameter specification forms.

The component model data structure as specified in Table A.2 allows for many *Component Types*. It can comprise of only one single connection point, or a structure of function blocks and branch mechanisms with relevant connection points and joints. It could include process components, as well. This feature provides for a hierarchical specification of an integrated model. The component can be viewed either by its own graphical symbol or if relevant by its internal graphical diagram. The symbol can include several graphical elements, such as connection points, enunciators and monitors. The elements can also depict different structural parts of the real component, and provide easy access to specify related attributes. Each component type can be used as a template to develop new *Component Instances* having the same structure, but possibly different specification parameter values. If it has the same parameter values as well, except for the name, it is a *Component Clone*. An existing component type can be used as a prototype for developing new component types with modified structures. Upon specification of the type, it can be decided which parameter values are type specific and which instance specific.

The parent of a process component is either a higher-level process component or the design data repository whereto the component itself and all its descendants belongs. A component specification can be used by several experiments. The component specification applies for both generic and composed components.

The user link list includes a reference to all graphical windows where the component is concurrently displayed. Only one user at a time, the active user, is allowed to modify the component. The modifications are broadcasted to all concurrent users. This functionality forms the basis for possible interactive

operation of the members of a design team, each one connected to the Internet and residing virtually anywhere.

Table A.2. Component Model attributes.

Structure attributes			
cmType	Type	*cmSE*	*SymbolElement* list entry
cmFB	*FunctionBlock* list entry	*cmCM*	*ComponentModel* list entry
cmCN	*CompoundNode* list entry	*cmCB*	*CompoundBranch* list entry
cmMC	*ModelConnector* list entry	*cmTdoc*	Documentation of type
cmPrnt	Parent: Component or Repository	*cmSieb*	Sibling list link
Structure generation attributes			
cmPR	*ParameterRecord* list entry	*cmScri*	Script index
Component instance attributes			
cmName	Instance name	*cmIdoc*	Documentation of instance
cmCon	Confidentiality of instance	*cmUpd*	Instance updates' history
cmUse	User link list	*cmAct*	Active user
cmIrect	Symbol boundary rectangle	*cmIpos*	Symbol position in window

The implementation of the graphics interface implementation has to be extremely thin, only including information needed for instant graphical operations, such as mowing graphical primitives and junctions in the visible windows. This facilitates for concurrent browser based read and operate access to the same component from several locations in the Internet.

Symbol Element

The symbol element specification according to Table A.3 is the primary connection to the graphics view of the component as a composite symbol published in a graphical window. Each element in the symbol can have a specified interactive function. The graphics primitive's entry includes the specification of the shape of the symbol element as well as its relative position in the symbol.

Table A.3. SymbolElement attributes.

seName	Instance name	*seMC*	*ModelConnector* list index
seType	Type	*seWin*	Window index
seGP	*GraphicsPrimitives* list entry	*seIdoc*	Documentation of instance
sePrnt	Parent: Component	*seSieb*	Sibling entry

Model Connector

The model connector specification in Table A.4 is used for establishment of junctions between components on the flow sheet, as well as between structural objects of the component itself, such as function blocks, process nodes, process branches, or even to individual coefficients of the branch mechanisms. The same connector can be referred to from several elements, although it is owned by the lowest level structural object in consideration.

Table A.4. ModelConnector attributes.

mcNam	Instance name	*mcMatch*	Matching connector type
mcType	Type	*mcIdoc*	Documentation of instance
mcPrnt	Parent: *CM, PN, PB, FB, or BM*	*mcSieb*	Sibling entry
mcCmp	Component name	*mcElem*	Element name

Connector Junction

The connector junction as specified in Table A.5 is only involved in the logical attachment of connectors. When making connections, the compatibility of the connection points is to be checked.

Table A.5. ConnectorJunction attributes.

cjX1	Connector index	*cjX2*	Connector index
cjPrnt	Parent: Component	*cjSieb*	Sibling list link

Mechanistic Functionality Specification

Function Block

Function blocks provide for flow sheet type graphical specification of connection to imported code that can make use of the general services provided, e.g. with respect to updates of state variables upon accepted new time-step. The function block attributes are set forth in Table A.6.

Table A.6. FunctionBlock attributes.

fbType	Type	*fbSRi*	Input *StateRecord* list entry
fbPR	*ParameterRecord* list entry	*fbSRo*	Output *StateRecord* list entry
fbSRn	Native *StateRecord* list entry	*fbTdoc*	Type documentation
fbPrnt	Parent: Component	*fbSieb*	Sibling entry
fbName	Instance DLL name	*fbIdoc*	Instance documentation

Compound Branches and Nodes

The local volume of a tank can represent just a single node or a required set of local nodes and local branches, to include the mechanisms describing the tank in consideration. The compound node attributes are presented in Table A.7.

Table A.7. CompoundNode attributes.

cnType	Type	*cnMN*	*MechanisticNode* list entry
cnMB	*MechanisticBranch* list entry	*cnCN*	*CompoundNode* list entry
cnPR	*ParameterRecord* list entry	*cnMC*	*ModelConnector* list entry
cnPrnt	Parent: Component	*cnSieb*	Sibling entry
cnName	Instance name	*cnIdoc*	Instance documentation

The transition capacity of a pipe can represent one or several mechanistic branches. The compound branch attributes are presented in Table A.8.

Table A.8. CompoundBranch attributes.

cbType	Type	**cbMB**	**MechanisticBranch** list entry
cbMCfr	From **ModelConnector** index	**cbMCto**	To **ModelConnector** index
cbPR	**ParameterRecord** list entry	**cbCB**	**CompoundBranch** list entry
cbPrnt	Parent: Component	**cbSieb**	Sibling entry
cbName	Instance name	**cbIdoc**	Instance documentation

Mechanistic Branches and Nodes

Connected mechanistic branches and nodes form together the fundamental structure conveniently describing the functionality of typical process components. They are also called simple nodes and branches. The mechanistic node attributes are presented in Table A.9.

Table A.9. MechanisticNode attributes.

mnType	Type	**mnSR**	**StateRecord** index
mnPrnt	Parent: CM or CN	**mnSieb**	Sibling entry

The mechanistic node comprises one single state variable, such as voltage, pressure, specific enthalpy, or concentration. It is assumed that the node itself does not accumulate any current, mass, or energy. For such purposes, specific mechanistic local branches are available. The mechanistic branch attributes are presented in Table A.10.

Table A.10. MechanisticBranch attributes.

mbType	Type	**mbSRbr**	Branch **StateRecord** index
mbTfr	From node type	**mbSRfr**	From node **StateRecord** index
mbTto	To node type	**mbSRto**	To node **StateRecord** index
mbPrnt	Parent: CM, CN or CB	**mbSieb**	Sibling entry
mbBM	**BranchMechanism** list entry	**mbSij**	Source variable entry s_{ij}
mbGf	Forward coefficient entry g_f	**mbGb**	Backward coefficient entry g_b

Branch Mechanisms

The functionality of a specific mechanistic branch is completely defined by the actual selection of branch mechanisms operating on the branch in series, such as turbulent loss and dynamic head representing separate hydraulic mechanisms acting concurrently in a transition branch. Further, it is also possible to define the relevant numerical integration scheme for each branch. The branch mechanism attributes are presented in Table A.11.

Table A.11. BranchMechanism attributes.

bmType	Type	*bmTE*	*TransitionEquation* list entry
bmInt	Integration method chosen	*bmMC*	*ModelConnector* list entry
bmPrnt	Parent: MB	*bmSieb*	Sibling entry
bmRf	Forward coefficient value R_f	*bmRb*	Backward value R_b
bmC	Capacitive coefficient value C	*bmE*	Internal state value E
bmI	Source impact value I	*bmL*	Inductive coefficient value L

Transition Equation

Accordingly, new mechanistic branch types can be specified just by selecting relevant mechanisms. Each mechanism can include several equation elements, each provided with specific input parameters or coefficients. The transition equation attributes are presented in Table A.12.

Table A.12. TransitionEquation attributes.

teType	Equation type	*teTdoc*	Type documentation
tePR	*ParameterRecord* list entry	*teName*	Instance name
tePrnt	Parent: BM	*teSie*	Sibling entry

Constant Coefficients

Each variable that will be constant during the simulation can be calculated at initiation time by relevant equations. This provides for specification of easily available input parameters, such as nominal values for a component. The constant coefficient attributes are shown in Table A.13.

Table A.13. ConstantCoefficients attributes.

ccDyn	Equation Type	*ccTdoc*	Type documentation
ccPR	*ParameterRecord* list entry	*ccSRo*	Output **StateRecord** list index
ccPrnt	Parent: Any	*ccSie*	Sibling entry

Secondary Coefficients

Secondary coefficients can be calculated based on state variables, parameters, and constants, at the end of each iteration step or time-step. The relevant attributes are shown in Table A.14.

Table A.14. SecondaryCoefficients attributes.

scTyp	Equation type	*scSRi*	Input **StateRecord** list entry
scPR	*ParameterRecord* list entry	*scSRo*	Output **StateRecord** list index
scPrnt	Parent: Any	*scSieb*	Sibling link

All parameters requested by the arising element and coefficient equations can be made available for relevant symbol elements, as well as the initial values for relevant local and transition state variables.

Parameter Records

The parameter record specifies user input and output parameters' user supplied names, engineering units, instant values as well as given defaults, minimum and maximum values. The relevant attributes are shown in Table A.15. References to parameter records can be made from many other records.

Table A.15. ParameterRecord attributes.

prType	Type	*prUnit*	Engineering unit
prMax	Maximum value	*prMin*	Minimum value
prName	Instance name	*prValue*	Value
prPrnt	Parent: Any	*prSieb*	Sibling entry

State Records

The state record as specified in Table A.16 is used for all state variables. The different instances of the state variables are stored in sequence in arrays of relevant data type.

Table A.16. StateRecord attributes.

srType	Type	*srDom*	Domain name
srTt	Index to values: - at time t	*srTe*	- at new estimate $t+\Delta t_e,\ k+1$
srTk	- at previous iteration $t+\Delta t_e,\ k$	*srTu*	- at previous time-step $t-\Delta t_u$
srTv	- at preceding step $t-\Delta t_u-\Delta t_v$	*srName*	Instance name
srPrnt	Parent: MN, MB, or FB	*srSieb*	Sibling link

Appendix B: Solution Data View

General

The experiment related solution data is detached from the specification data, enabling easy extraction of a compact model entity with specified interface parameter list and control commands like start and stop. This structure enables two separate binary snap shots for the experiment, one including all experiment data and the other only comprising the dynamic variables of the experiment.

Experiments

Several experiments can be specified in the design data repository of a large process model. The content of a specific experiment is most efficiently specified graphically, selecting areas of the flow sheet, or including and excluding separately individual components. In a distributed design environment, model specifications from several modelling engines can be included in one experiment. The total model can on user's request be partitioned and distributed to several simulation engines. For each simulation engine a local experiment header is specified according to Table B.1.

Table B.1. ExperimentHeader attributes.

ehName	Instance name	*ehCL*	*ComponentLink* list entry
ehCque	Control queue	*ehJL*	*JunctionLink* list entry
ehTstep	Time-step	*ehRate*	Real-time ratio
ehTstrt	Start time	*ehTstop*	Stop time
ehTime	Time instant	*ehZone*	Zone list entry

Temporary link list worktables as specified in Table B.2 are needed to flatten down the process specification data structures for efficient computation, such as lists of included components, junctions, nodes and branches.

Table B.2. Sample local experiment link lists.

ComponentLink list			
clCM	*ComponentModel* index	*clSieb*	Sibling link
JunctionLink list			
jlCJ	*ConnectorJunction* index	*jlSieb*	Sibling link
NodeLink list			
nlMN	*MechanisticNode* index	*nlSieb*	Sibling link
BranchLink list			
blMB	*MechanisticBranch* index	*blSieb*	Sibling link

From all local experiment data, separate zones are identified and zone specific connection lists are generated. A zone can include several implicit equation groups to be solved concurrently, or even only explicit equations to be sorted with regard to inputs and outputs in a sequential manner. Island connection link lists are specified in Table B.3.

Table B.3. Island connections.

BranchConnection link list			
bcMB	*MechanisticBranch* index	*bcCfr*	Connection type from-node
bcCto	Connection type to-node	*bcSieb*	Sibling entry
NodeConnection link list			
ncMN	*MechanisticNode* index	*ncCon*	Number of implicit connections
ncOrd	Solution order within island	*ncSieb*	Sibling entry

Zone Handler

The temporal zone handler executes all equations of the zone and controls the zone specific time advancement. Each zone can have a different simulation time-step. The zone control is responsible for updating the state variables upon acceptance of a new time-step. The zone handler attributes are presented in Table B.4.

Table B.4. ZoneHandler attributes.

zhBC	*BranchConnections* list entry	*zhNC*	*NodeConnections* list entry
zhT	Zone accepted time instance	*zhK*	Estimate iteration number
zhEdt	New estimate time-step	*zhUdt*	Previous time-step
zhVdt	Preceding time-step	*zhCcon*	Input coefficients' entry
zhCblc	Function block computation entry	*zhCdyn*	Dynamic parameters' entry
zhCsta	Static parameters' entry	*zhSmx*	Sparse matrix storage entry
zhCout	Output coefficients' entry	*zhName*	Instant name
zhPar	Parent: Experiment	*zhSieb*	Sibling link

Different types of equations in the zone are then solved, providing for updated temporary parameters, variables and coefficients. The sorting of all equations of same type to be solved in a sequence, enables for pipelining and fine grain parallelisation. Several entries to the equation computation handler are included to enable the correct sequence of calculation. The record attributes for equation calculation in a specified zone are shown in Table B.5.

Table B.5. EquationComputation attibutes.

ecEqu	Equation type	*ecInr*	Number of instances
ecPnr	Number of parameters	*ecPix*	Parameter index list entry
ecPrnt	Parent: Island	*ecSieb*	Sibling entry

Sparse Matrix Storage

From the calculation point of view, it is not required to construct the matrices to be solved in advance. The content and structure is specified by the included set of forward and backward elements, as well as the source elements of the companion model branch k between the nodes i and j.

The matrix equation arising form the implicitly connected components can efficiently be stored applying a linked list scheme as set forth in Table B.6. Before construction of the sparse matrix from the companion model coefficients the degrees of the internal nodes are recorded. The factorisation order is chosen starting from the nodes with smallest number of connections, to get a reasonably small fill in of new elements. The factorisation order needs only to be re-optimised if there has been a major change in the connections.

Table B.6. SparseMatrix attributes.

smA	All element values $a(i,j)$, starting with diagonal $a(k,k)$, $k = 1..N$
smDwn	Link index to next element $a(i,k)$ *down* from diagonal in column k
smRow	*Row i* index of above element
smRgh	Link index to next element $a(k,j)$ *right* from diagonal in row k
smCol	*Column j* index of above element
smB	Source vector value $b(k)$
smX	Index to relevant state variable $x(k)$

A separate temporary storage is used for each sparse matrix. It is needed because the matrix elements are overwritten during factorisation, and the input vector is overwritten during solution phase. The separation, however, also enables the use of medium grain task assignment on parallel processors.

Discontinuity Handler

The separation of the calculation into zones, reduces the immediate impact of crossing of discontinues only to the zone itself. The data structure implies that the overlapping monotonic regions of a correlation are continuously calculated for relevant input, whereas the output to be used is chosen by the discontinuity handler. The discontinuity handler attributes are presented in Table B.7.

Table B.7. DiscontinuityHandlar attributes.

dhType	Discontinuous function type
dhName	Instance name
dhSta	Index to state variable for monotonic region
dhLow	Index to lower bound of monotonic region if relevant
dhUpp	Index to upper bound of monotonic region if relevant
dhPre	Link to next lower region if relevant
dhNext	Link to next upper region if relevant

Nonlinear Correlation Data

Nonlinear functions representing empiric correlations for material properties used in real-time dynamic simulation of integrated industrial processes need to be smooth and very rapidly calculated. Traditional methods involving slowly converging nested iterations and relevant discontinuously jumping iteration noise are not very suitable for dynamic simulation. Computer memory is much more affordable than processor speed. Accordingly, several types of tabulated approaches have shown to be feasible. The fastest and simplest method: linear interpolation is however suitable only for up to two-dimensional cases. The tabulation frequency can be larger in regions with major changes, which saves memory considerably. For a larger number of dimensions nonlinear description methods such as neural networks are used to reduce the required amount of stored data. On the other hand the calculation is slower, and the preparation of the data is much more complicated. Off-line preparation of the material property data is very important. It can include following phases:

1. Gathering of measurement data, or
2. Calculation of reasonable input values using iterative flash calculation
3. Elimination of corrupted data.
4. The identification of monotonic regions
5. Extension of the regions by the derivatives at the boundaries.
6. Teaching the neural network
7. Checking its accuracy.

Author(s)
    Juslin, Kaj

Title

A Companion Model Approach to Modelling and Simulation of Industrial Processes

Abstract

Modelling and simulation provides for huge possibilities if broadly taken up by engineers as a working method. However, when considering the launching of modelling and simulation tools in an engineering design project, they shall be easy to learn and use. Then, there is no time to write equations, to consult suppliers' experts, or to manually transfer data from one tool to another.

The answer seems to be in the integration of easy to use and dependable simulation software with engineering tools. Accordingly, the modelling and simulation software shall accept as input such structured design information on industrial unit processes and their connections, as provided for by e.g. CAD software and product databases.

The software technology, including required specification and communication standards, is already available. Internet based service repositories make it possible for equipment manufacturers to supply "extended products", including such design data as needed by engineers engaged in process and automation integration. There is a market niche evolving for simulation service centres, operating in co-operation with project consultants, equipment manufacturers, process integrators, automation designers, plant operating personnel, and maintenance centres.

The companion model approach for specification and solution of process simulation models, as presented herein, is developed from the above premises. The focus is on how to tackle real world processes, which from the modelling point of view are heterogeneous, dynamic, very stiff, very nonlinear and only piece vice continuous, without extensive manual interventions of human experts. An additional challenge, to solve the arising equations fast and reliable, is dealt with, as well.

Keywords

    industrial processes, process simulation, simulation models, simulation software, software implementation, systems architecture, model specification, structured graphs, companion models, sparse matrices

Activity unit

    VTT Industrial Systems, Tekniikantie 12, P.O.Box 1301, FI–02044 VTT, Finland

VTT PUBLICATIONS

554 Holttinen, Hannele. The impact of large scale wind power production on the Nordic electricity system. 2004. 82 p. + app. 111 p.

555 Rintala, Kai. The economic efficiency of accommodation service PFI projects. 2004. 286 p. + app. 193 p.

556 Kiiskinen, Laura-Leena. Characterization and heterologous production of a novel laccase from *Melanocarpus albomyces*. 2004. 94 p. + app. 42 p.

557 Mäki-Asiala, Pekka. Reuse of TTCN-3 Code. 2005. 112 p.

558 Ebersberger, Bernd & Lehtoranta, Olavi. Pattern of Innovative Activities among Finnish Firms. 2005. 197 p.

559 Kiihamäki, Jyrki. Fabrication of SOI micromechanical devices. 2005. 87 p. + app. 28 p.

560 Tuulari, Esa. Methods and technologies for experimenting with ubiquitous computing. 2005. 136 p. + app. 2 p.

561 Janne Merilinna. A Tool for Quality-Driven Architecture Model Transformation. 2005. 106 p. + app. 7 p.

562 Backman, Ulrika. Studies on nanoparticle synthesis via gas-to-particle conversion. 2005. 45 p. + app. 62 p.

563 Eerikäinen, Hannele. Preparation of nanoparticles consisting of methacrylic polymers and drugs by an aerosol flow reactor method. 2005. 112 p. + app. 55 p.

564 Wihersaari, Margareta. Aspects on bioenergy as a technical measure to reduce energy related greenhouse gas emissions. 2005. 93 p. + app. 71 p.

565 Kallio, Marke. The elastic and damping properties of magnetorheological elastomers. 2005. 146 p.

566 Kaasinen, Eija. User acceptance of mobile services – value, ease of use, trust and ease of adoption. 2005. 151 p. + app. 64 p.

567 FUSION Yearbook. Association Euratom-Tekes. Annual Report 2004. Ed. by Seppo Karttunen & Karin Rantamäki. 2005. 129 p. + app. 13 p.

568 Hanhijärvi, Antti, Ranta-Maunus, Alpo & Turk, Goran. Potential of strength grading of timber with combined measurement techniques. Report of the Combigrade-project – phase 1. 2005. 81 p. + app. 6 p.

569 Katina, Kati. Sourdough: a tool for the improved flavour, texture and shelf-life of wheat bread. 2005. 92 p. + app. 81 p.

570 Kataja, Kari J. Numerical modelling of near field optical data storage. 2005. 102 p. + app. 63 p.

571 Järnström, Helena. Muovimattopinnoitteisen lattiarakenteen VOC-emissiot sisäilmaongelmatapauksissa. 2005. 73 s. + liitt. 14 s. (RTE)

572 Kiviniemi, Arto. Requirements management interface to building product models. 2005. 328 p.

574 Juslin, Kaj. A Companion Model Approach to Modelling and Simulation of Industrial Processes. 2005. 155 p. + app. 15 p.