Design of a General Architecture for the Integration of Process Engineering Simulation and Computational Fluid Dynamics

by

Fabrizio Bezzo

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

December 2001

Centre for Process Systems Engineering and
Department of Chemical Engineering
Imperial College of Science, Technology and Medicine
London SW7 2BY

Abstract

Computational fluid dynamics (CFD) and process simulation are important tools for the design and optimisation of chemical processes. The two technologies are largely complementary, each being able to capture and analyse some of the critical process characteristics. Their combined application can, therefore, lead to significant industrial benefits. This is especially true for systems, such as chemical reactors, in which steady performance, dynamics and control strategy depend on mixing and fluid flow behaviour.

A new approach is presented for the integration of the capabilities of CFD technology and process simulation via a general architecture and an interface that allows the automatic exchange of critical variables between two independent packages working together and the simultaneous solution of the modelling equations. The proposed architecture generates a network of process simulation zones (coarse grid) overlapping a fine CFD grid in order to take into account the hydrodynamics of the system. The CFD model is continuously updated from the parameters and variables calculated by the simulation model in the corresponding portion of the CFD domain. The approach permits a local description of a system according to criteria of homogeneity and uniformity of selected properties within each zone.

First, a general way is demonstrated of describing and generating an interface between a CFD grid and a simulation zone model when the mapping is established a-priori, and achieving a simultaneous solution. Second, a method is demonstrated of automatically generating a suitable zone model, according to a variety of criteria aimed at identifying "well-mixed" zones.

Finally, special attention is given to the search for and implementation of general methods to boost calculation speed and to diminish the computational burden, which could otherwise heavily restrict the use of the suggested integration technique.

The feasibility of the new approach is demonstrated by integrating a widely used CFD package (Fluent by Fluent Inc.) within a general-purpose process simulator (gPROMS by Process Systems Enterprise Ltd.). Examples for dynamic process applications illustrate the benefits and the unique achievements of the suggested method, which represents the first practical attempt to improve process design and simulation by integrating process modelling and CFD tools in a generic way.

Acknowledgements

I would like to express my gratitude to my supervisor, Professor Sandro Macchietto, who made this project possible and helped me with his guidance and support throughout this work.

I am very grateful to Professor Costas Pantelides for his invaluable and essential advice and help in many critical areas of the thesis.

I would like to acknowledge Dr. A. Kakhu who taught me how to use gPROMS and, in particular, the gPROMS Foreign Objects.

I am grateful to all companies who gave a big contribution to this work: Fluent Inc. and in particular Dr. Marshall, Dr. B. Hutchings and Dr. J. Schuetze for their technical help and generosity; PSE and in particular Dr. Z. Urban for sharing with me his outstanding experience in modelling and simulation; SGI Inc. and in particular Dr. L. Zullo and R. LaRoche for introducing me to CFD simulation and giving me so many fundamental suggestions; Bayer and in particular Dr. D. Leineweber for his critical discussions which led to a better understanding of the issues considered in chapter 5. The financial support from the Centre for Process Systems Engineering is gratefully acknowledged.

Finally, I would like to remember the late Lakis Liberis: his humour and his technical advice helped me a lot during many critical periods of this research.

There are a lot of others that I wish to acknowledge: the staff in the centre, the staff in the department, all my office mates. I would like to thank Neil who carefully and critically read this thesis. I would like to thank my friends here and in Italy who made this experience, in the happy moments as well as in the difficult ones, simply extraordinary. Antonietta, Dylan, Katina, Laura, Leo, Luca, Michele, Omar, Ping, Rita, Sara, Teresa, Ugo, Viviana, thank you very much!

Finally, I would like to say "grazie" to my parents and my sister Elena: thanks for having been close to me throughout these years.

Contents

List of Figures

List of Tables

Nomenclature

Here only the list of symbols concerning **zones** *and* **zone modelling** *will be considered. These symbols appear throughout this thesis and their knowledge is important for the general understanding of the entire work. Other symbols concerning more specific issues are explained when introduced in the chapters.*

$A_{cc'}$	area of the surface between neighbouring cells c and c'
c	CFD cell
$\mathcal{C}$	set of CFD cells (grid)
F	mass flowrate between neighbouring zones
$F_{zz'}$	mass flowrate between neighbouring zones z and z'
f	mass flowrate between CFD cells
$f_{cc'}$	mass flowrate between CFD cells c and c'
$\mathbf{F}(c, c')$	flowfield: set of all mass flowrates f
nz	number of zones
np	number of properties required to set up zone network
$\mathcal{N}_c$	set of zones neighbouring cell c
$\mathcal{N}_z$	set of zones neighbouring zone z
$\mathbf{P}$	set of properties required to set up zone network
$Z(c)$	zone to which cell c belong
$\mathcal{Z}$	set of zones
$\mathcal{Z}_i$	set of internal zones
$\mathcal{Z}_e$	set of environment zones
$\Delta\mathbf{P}$	tolerances for properties $\mathbf{P}$ in the zoning algorithm
$\phi_{cc'}$	flux ($f_{cc'}/A_{cc'}$) between cells c and c'
Θ	automatic zoning algorithm

Chapter 1

Process Modelling and CFD: a Review

1.1 Introduction

Computational Fluid Dynamics (CFD) and process simulation are important tools for the design and optimisation of chemical processes. CFD is a particularly powerful tool for the study of fluid dynamics and mixing processes within individual items of process equipment. The aim is often to avoid expensive experimentation and to use the information and insight gained in order to obtain a better design of the unit. The ability of CFD to handle complex equipment geometry is especially important in this context. Nonetheless, despite many recent improvements, CFD's ability to describe the physics involved (other than the flow behaviour) in several application areas and/or to solve the underlying numerical problems is still limited. Such areas include, among others, the important classes of complex reactive systems and multiphase processes with multicomponent phase equilibria. Moreover, CFD is currently primarily used for steady state simulations. Performing realistic dynamic simulations is often problematic due to excessive computational times and difficulties with the description of some of the discontinuities that typically occur in dynamic processes.

Process simulation is also well-established in the process industry, being used to study individual unit operations as well as multiple interconnected units or even entire plants. The latest generation of process simulation tools is extremely flexible, being able to represent a very wide range of multicomponent, multiphase and reactive systems. It can also deal with both steady-state and dynamic models subject to discontinuities. However, most of the models used by process simulation tools either ignore all

spatial variations of properties within each unit operation (invoking the "well-mixed" assumption) or are limited to simple idealised geometries (e.g. cylinders with 1- or 2-dimensional property variations). Moreover, even in models that represent mass and heat transfer phenomena to a high degree of detail, the treatment of fluid mechanics is usually quite rudimentary. This results in a poor ability to predict the quality of the products and to design optimal control strategies when hydrodynamic effects are important. It is very difficult to address detailed equipment design, scale-up and complex operations. Hence, a significant amount of pilot plant work is required, with the associate expenditure of time and money.

In view of the above, a combined approach is strongly advocated and there are clear benefits to be gained from a closer integration of the two technologies. Mixing is a general problem concerning most chemical processes. In some important cases (such as crystallisation or polymerisation), the combined understanding of fluid flow behaviour and the other physical and chemical phenomena is essential to predict, optimise and control the entire process. The design of chemical processes could receive great benefits from a more general approach capable of improving the description of complex dynamics, and the analysis and optimisation of equipment and process behaviour. The main objectives of this thesis are to investigate these general aspects of process design and to develop a new methodology and approach to solve some of the main problems currently impeding a combined utilization of CFD and process simulation.

In this chapter, after introducing the general problem of multiscale modelling, a short review of the existing literature on the subject is considered. First an overview and critical analysis of the main techniques in process simulation is presented. The overview will focus on the aspects related to the objectives of this thesis, i.e. general process simulation capabilities and deficiencies with respect to CFD. The research at Imperial College and, in particular, the SpeedUp and gPROMS projects will be considered as the main references in the field.

Next, the technology and use of CFD will be briefly surveyed. Again the intention is not to give a comprehensive survey on CFD, but only to examine characteristics and drawbacks related to CFD in the process industry. The main differences in the numerics adopted in the two technologies will be outlined pointing out their advantages and drawbacks.

Finally, we will focus our attention on the aspects more closely related to our present investigation, i.e. the use of CFD to integrate the description of process phenomena

and to improve traditional methodologies in process simulation.

1.2 Multiscale Modelling

The use of CFD and process simulation is best understood within the wider context of the multiscale nature of phenomena and operations in process engineering (Villermaux, 1995). In general, the scale of these phenomena range from the nanoscale of molecules, atoms and sub-atomic particles (involving distances of $\mathcal{O}(10^{-10})$ m and times under $\mathcal{O}(10^{-12})$ s) to the megascale of global supply chain (with distances of $\mathcal{O}(10^{7})$ m and times of $\mathcal{O}(10^{8})$ s). Intermediate scales include the microscale of particles, eddies and bubbles, the mesoscale of process equipments and the macroscale of process plants. The widely different scales of both space and time involved in process engineering pose great challenges to the attempts to encode our knowledge in terms of mathematical models.

Integrating CFD and process simulation is part of this effort to combine different scales of modelling. CFD is concerned with the description of fluid flow behaviour at a microscale, while process modelling tools are used at the meso and macroscale of process equipments and plants (overall conversions, production rates, etc.).

The traditional way of addressing multiscale complexity can, perhaps, best be described by *scale decoupling*. This simply focuses scientific and engineering endeavour on each individual scale with the aim of building the best possible understanding and description of the phenomena taking place at that scale. For example, such descriptions may take the following forms according the specific scale:

- **nanoscale:** models of molecular and intra-molecular motion
- **microscale:** mixing and fluid flow
- **mesoscale:** detailed modelling of unit operations
- **macroscale:** multipurpose plant design and scheduling
- **megascale:** dynamics of supply chains

Of course, the different scales are not independent of each other. After all, our mesoscale models of dynamic unit operation models invariably require some description of the behaviour of both the materials involved and fluid flow, which are precisely the objects of study at nanoscale and microscale respectively. The traditional approach to dealing with such interactions has largely been based on *scale aggregation*, leading to simplified descriptions of behaviour at each scale in terms of quantities that are directly relevant

to higher scales. Nonetheless, the separation between scales is kept. Process simulation is mainly used to address problems at a meso-macro scale, while CFD is a tool to describe microscale phenomena. The scope and usage of process simulation and CFD as well as the attempts to improve their capabilities in dealing with a wider range of phenomena will be described in the next sections. The problem of multiscale modelling will be resumed in chapter 2 and studied in depth from a different perspective.

1.3 Process Simulation Overview

The primary purpose of modelling tools is to facilitate the construction of mathematical models of the process under consideration. In a broader sense it could be stated that any software capable of handling, at least partially, the above goal is a process modelling tool. Nonetheless, especially during the last two decades the concepts *process simulation* and *process modelling tools* have acquired a more specific meaning. We adopt the definition given by Pantelides and Britt (1995) in their review of multipurpose process modelling environments:

> A formal definition for a process modelling tool is rather difficult to produce, and, ..., one based merely on strict *functional* capabilities would probably be too wide to be useful. Instead, we have to consider the *extent* to which different software tools support the process modelling activity. This includes the ease of use of the tools and the complexity of process that they can handle with reasonable effort, rather than merely what functions they might, in principle, be capable of. We will therefore restrict our attention to software that support the *high-level declarative* definition of *mathematical* models of *complex* processes, as well as the construction of models of novel unit operations from first principles.

The modelling tools may be classified into two categories (e.g. Westerberg *et al.*, 1979) according to the *modular* and *equation-oriented* approaches.

Modular simulators are usually structured through a process flowsheet. Modules describing individual unit operations are linked together by means of connections representing the required flow of information. In the modular approach each model will need to solve the equations of the corresponding mathematical model. An important feature of modular processing tools is the need to organise and order the computations carried out by the individual models: this often involves the analysis of the flowsheet to

identify partitions that may be solved independently or sets of streams which have to be torn to remove cyclic dependencies (Westerberg *et al.*, 1979). Perhaps the first sequential modular simulator (Biegler *et al.*, 1997) was the Flexible Flowsheet developed in 1958 at M.W. Kellogg Corp. to deal with the iterative procedure for determining steady-state overall heat and material balances for a process. The development of more advanced methods for the decomposition and solution of modular flowsheet has led to the definition of more standardised and efficient simulation tools developed by a few specialised companies. The user will have to select the models of interest (contained in a library), provide the model parameters and connect them according to the process specifications. In general, the user will have very little access to the models which cannot be adapted to the particular case. Thus, although the modular approach is a powerful and accessible tool to solve many standard engineering problems, it lacks the flexibility to support the implementation and solution of more complex models (Marquardt, 1996).

In an equation-oriented package one fundamental difference is that the unit operation modules do not need to solve their own equations. Equations are passed to a higher-level executive which is responsible for assembling them into a suitable form and eventually passes all equations to one or more numerical solvers (Braunschweig *et al.*, 2000). Equation-oriented simulators were conceived as multi-purpose process modelling environments: their architecture is such as to support the construction and maintenance of models irrespective of the application for which the latter are used (Pantelides and Britt, 1995). Achievements of equation oriented simulation in the description of industrial processes and procedures are impressive, indeed so much so that they are currently employed for almost all aspects of plant design and operations (Foss *et al.*, 1998).

The need to go beyond the solution of steady-state problems was the main stimulus for the development of dynamic simulation packages able to deal with control system design, hazard analysis and operability studies. A review of the early work in this area is given by Shacham *et al.* (1982). The SpeedUp project was started at Imperial College to tackle these issues (Sargent and Westerberg, 1964, Perkins and Sargent, 1982) in order to provide for an integrated approach to the optimum design of complex processes, and thus eliminate much of the need for intervention by the designer in the intermediate stages. Pantelides (1988) further developed SpeedUp into a consistent language for defining steady-state, optimization and dynamic flowsheeting problems: a high-level declarative language is used for describing mathematical models in terms

of sets of variables and the ordinary differential and algebraic equations (DAEs) that relate them. Models of more complex unit operations may be formed from instances of the basic models, while a model of the entire plant may be formed by combining instances of both models and unit operations into a flowsheet.

Following the classification suggested by Marquardt (1996), the SpeedUp and g-PROMS modelling approaches may be described as *general modelling languages*: they are designed to support hierarchical decomposition of complex models in order to facilitate reuse and modification of existing models by means of object-oriented programming. Other well known general modelling languages are ASCEND (Piela *et al.*, 1991), which was one of the first modelling environments to use object-oriented concepts for the organisation of modelling data and the recently developed ABACUSS II and DAEPACK packages (Feehery *et al.*, 1997, Tolsma and Barton, 2000).

Another important category is represented by the *process modelling languages* (Marquardt, 1996): although very similar to general modelling languages, these languages are designed to match the specific issues of a particular application domain in the language definition (i.e. elements tailored to chemical engineering applications are included in the language definition). Typical examples are MODEL.LA (Stephanopoulos *et al.*, 1990) and VEDA (Marquardt, 1992, Bogusch and Marquardt, 1995).

The first process modelling packages were mainly concerned with the representation of continuous processes. However, physical systems present many intrinsic discontinuities: the opening and closing of a safety valve or the bursting of a safety disk are common phenomena which present non-trivial issues in modelling. The modelling of operating procedures in a process plant is another area where discrete actions are imposed on a system. Joglegar and Reklaitis (1984) described one of the first specialized packages to model batch operations. Barton and Pantelides (1994) dealt with the issue through the design of a new system capable of addressing the simulation of processes presenting combined discrete and continuous characteristics, leading to the development of the gPROMS package.

Further advances came from the recognition that many processes are intrinsically distributed, i.e. the conditions within them vary with respect to both time and space (Pantelides and Barton, 1993). This led to the creation of a generation of simulators able to solve mixed systems of partial and ordinary differential and algebraic equations (PDAEs) and integral PDAEs (IPDAEs) (Oh and Pantelides, 1996): gPROMS supports the direct modelling of distributed parameter systems by allowing the process variables and equations to be distributed over one or more distribution domains. However, the modelling capabilities are limited to regular geometries (e.g. Figure 1.1), although they

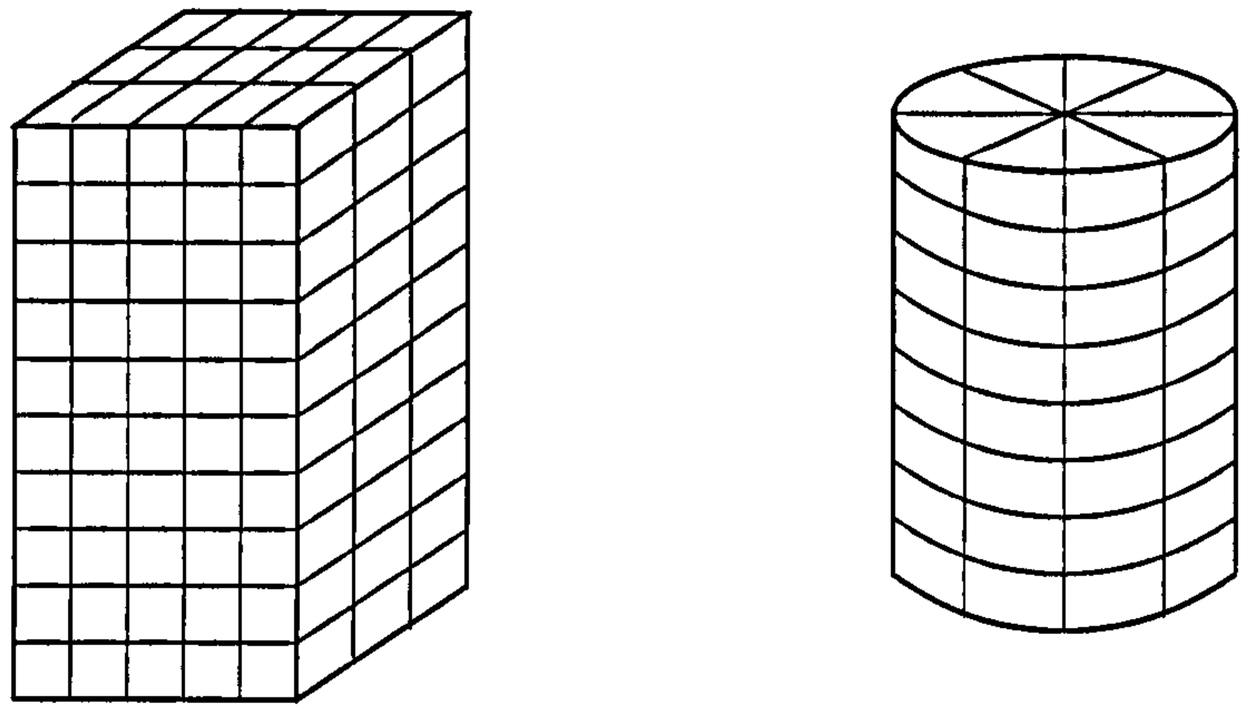


Figure 1.1: Distributed domains in process simulation.

can be applied to an arbitrary number of dimensions.

The great diversity of model-based applications make it very unlikely that a single process modelling environment will have the resources and the flexibility to cope with all potential opportunities (Pantelides and Britt, 1995). An open architecture is an important first step to address this problem: the process modelling environment would act as *model server* responding to requests issued by one or more applications concerning process models defined within the environment. Many collaborative projects (e.g. the CAPE-OPEN project (Braunschweig *et al.*, 2000)) have started to improve the standardisation between simulation tools to allow complex process modelling tasks and model based applications to be performed via the collaborative use of a wide variety of software components. The ABACUSS II and DAEPACK packages (Feehery *et al.*, 1997, Tolsma and Barton, 2000) demonstrate great flexibility in incorporating third party models and numerical algorithms as well as being embedded in other applications. Kakhu *et al.* (1998) illustrated the open software architecture in gPROMS: this important feature will be further commented on in the next chapter.

Another important aspect within process simulation tools is their capability to perform process optimisation. Several general-purpose process simulation tools provide facilities for steady-state optimisation. However, many typical optimisation problems in the process industry require the optimisation of process dynamics. The incorporation of robust optimisation technology within general modelling languages facilitates the use of more complex process models and ensures consistency between related applications such as dynamic simulation and dynamic optimisation (Pantelides and Britt, 1995). An important contribution in this direction was given by Smith and Pantelides (1999) who presented an algorithm for the solution of nonconvex mixed integer nonlinear programming (MINLP) problems and incorporated it within a multi-purpose process modelling

environment (gPROMS).

As a conclusion of this review, we can summarise that state-of-the-art equation-oriented process modelling tools are capable of handling:

- complex DAEs systems (i.e. steady-state and dynamic problems)
- PDAEs (and IPDAEs) within regular (rectangular or polar) domains
- discrete operations
- optimisation

Unfortunately, some important restrictions still limit the usage of process simulation when hydrodynamics is important. One restriction is that the distributed domains that can be described by current technology must be expressed as the cartesian product of one or more line segments. A second issue concerns the reliability and efficiency of the numerical solvers: the 3-dimensional models of fluid flow cannot easily be solved using the generic numerical codes typically implemented in process modelling tools (Neumann, 2001). The above limitations are already addressed quite well by a complementary type of technology: Computational Fluid Dynamics (CFD). In the rest of this thesis, we will deal with process simulation tools of the *general modelling language* type.

1.4 Computational Fluid Dynamics Overview

Computational Fluid Dynamics is the field concerning the representation and numerical solution of equations describing fluid dynamics. The field has become so important that it now occupies the attention of about one third of all researchers in fluid mechanics (Ferziger and Peric, 1999). A CFD software is essentially a computer program for modelling fluid flow by solving the Navier-Stokes equations.

The roots of CFD can be traced back to the 1920s, but it was only the advent of computers which allowed CFD to become a design tool (Badcock *et al.*, 2000). About twenty years ago Chapman (1979), while overviewing the state-of-art of CFD in aerodynamics, foresaw the possibility and the potential exploitation of such a technique as an alternative to experimental facilities as soon as computing power would allow the implementation of suitable models and numerics. In fact, over the past two decades CFD has undergone enormous progress, and is now capable of meeting many of these earlier hopes.

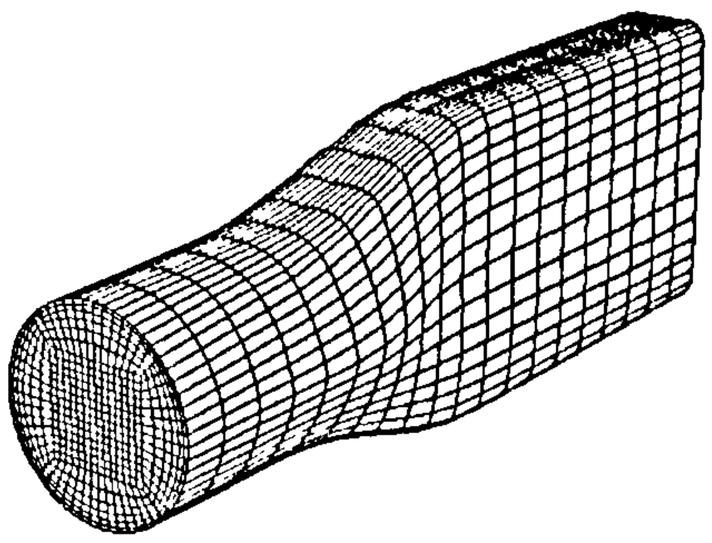


Figure 1.2: Structured grid.

1.4.1 Grid Generation

The key area of development has been in the capability of handling complex geometry domains. The essential element is the construction of a discrete grid on which to represent the field equations in finite form and to implement the associated boundary conditions. Grid generation strategies can be classified as (Soni, 2000):

1. *Cartesian grids.* A network of uniform spaced grid lines is placed in a rectangular box application. Boundary conditions are established by cutting interior geometrical entities within grid lines. Notwithstanding the simple discretisation approach and the promise of highly automated grid generation, many numerical issues remain and significant development is needed (Soni, 2000).

2. *Structured grids* (Figure 1.2). The structured grid is represented by a network of curvilinear coordinate lines such that a one to one mapping can be established between physical and computational space. The grid points conform to the solid surfaces/boundaries and hence provide the most accurate way to specify boundary conditions. For complicated configurations, a block-structured approach is considered: the domain is divided into subregions which are individually meshed and then patched together at common interfaces.

3. *Unstructured grids* (Figure 1.3). Cells are attached to each other face-to-face in an arbitrary topology. The grid information is represented by a set of coordinates (nodes) and the connectivity between nodes. The unstructured grids offer great geometric flexibility, a high degree of local resolution control and a high potential for automation.

4. *Hybrid grids* (Figure 1.4). These grids allow polygonal cells with different number of sides. In general, structured grids are generated near the solid components, while unstructured grids are used to mesh the remaining region.

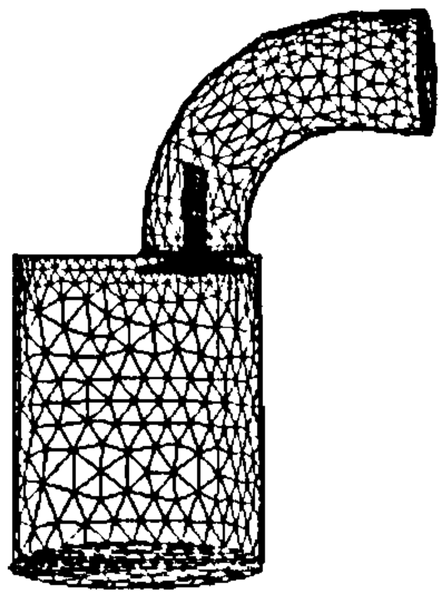


Figure 1.3: Unstructured grid.

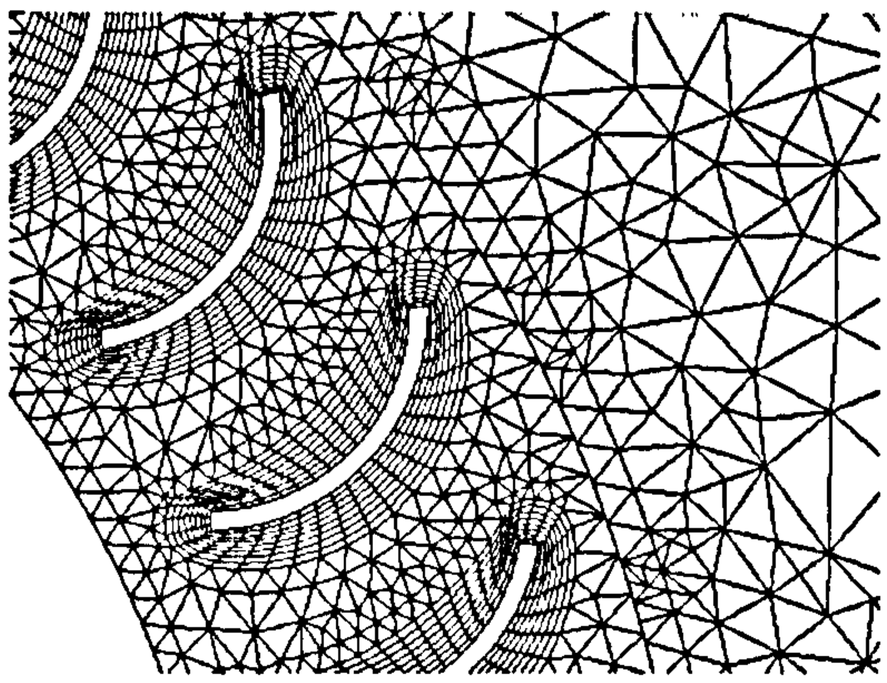


Figure 1.4: Hybrid grid.

5. *Gridless methods.* A cloud of points is placed in the fields and the numerical solution of the equations is obtained without requiring any type of explicit connectivity links between points. The method is still under development.

In the past (1980s) the meshing of a grid could take up to 6 months work. It is quite natural that most software development effort was aimed at reducing this time. The recognition that the finite volume method (§ 1.5) could be applied to unstructured and hybrid grids was one of the breakthroughs in CFD development (Gosman, 1998). In fact, although some of the most efficient numerical procedures can be implemented only on structured grids (Badcock *et al.*, 2000), unstructured grid generation may be highly automated thanks to the possibility of eliminating some of the rigid requirements needed by structured meshing (Gosman, 1998). Nowadays, the meshing of even complex geometries may take one or two days if unstructured grids are adopted (still 2-3 weeks in case of a structured multiblock grid) (Soni, 2000).

Furthermore, the need to handle moving boundaries (to model, e.g., mixing vessels or pumps and compressors) has pushed the research effort towards the development of techniques capable of considering motion within the domain. Several approaches such as:

- *moving meshes*: involving distortion of the computational mesh;
- *sliding meshes*: portions of the mesh are allowed to slide relative to each other at a common interface;
- *multiple rotational frames*: local rotational mesh is simulated rather than directly invoked by means of transformations of CFD calculations at the interface;

allow the treatment of many situations where motion simulation is needed (Gosman, 1998). Another recent advancement in meshing techniques is represented by the use of *adaptive grids.* It is in general difficult to set up an adequate grid at the first attempt and often the mesh has be to modified to enable the solution procedure to satisfy the solution requirements. A great deal of effort has been invested in developing error measures as a criterion for *adapting* the mesh after a computed solution. Algorithms have been developed to allow an automatic refinement of grids (a review is given in Sheppard (1988)) and most commercial CFD packages are capable of refining the mesh after a first solution is found. More recently, self-adaptive methods have been introduced (e.g. Xu *et al.*, 1998, Pain *et al.*, 2001) to dynamically adapt a grid to the transient behaviour of fluid flow.

1.4.2 CFD applications

The improvements in meshing capabilities, the use of parallel computing and the improvement in the physical modelling of turbulent flows, have deeply changed the use of CFD in industry. Computational fluid dynamics is no longer an exclusive tool of aerospace and automotive engineers. Applications are common in chemical and food industry, environmental science, biochemistry, etc.

The main use of CFD is still the simulation of fluid flow to improve the *design of process equipment.* Typical uses are the design of aircrafts, automobiles and turbines. In the chemical industry the first applications regarded the design of cyclons, driers and mixing devices. The use of CFD to understand the mixing effectiveness in stirred tank reactors (e.g. Jongen, 2000, Revstedt *et al.*, 2000) and other mixers is a very common practice (Chemical Engineering Online, 2001). Gordon and Richardson (1997) reviewed some of the recent CFD applications in the food industry: clean-room design, static

mixers, pipe flow design are some examples. Another recent application of CFD is the study of environmental flows as described, for instance, in the works of Gosman (1999a) and Kim and Boysan (1999): the study illustrated how flows within buildings play an important role in determining the environment within them and sometimes the integrity of the structure itself.

An important area of CFD use regards the modelling of engine related flows and *combustion reactors* in general. Very tailored numerics (Eaton *et al.*, 1999) are adopted to solve these special problems and, although some simplifications are required, CFD is already a common tool in engine analysis and design (Gosman, 1999b). Several studies (e.g. Shah and Fox, 1999, Kolhapure and Fox, 1999) have been carried out in the attempt to incorporate complex kinetics within CFD simulations, but a general and computationally efficient method seems unlikely to be discovered soon.

CFD use also includes the simulation of *non-Newtonian fluids.* On the one side specialised codes have being developed to describe specific processes such as extrusion and transport of melted polymers. On the other side the treatment of non-Newtonian and viscous fluids has been implemented in the general CFD packages to allow a better representation of typical CFD analyses where fluid rheology is important. For instance, CFD simulations are applied to meet the needs of bioprocesses in pharmaceutical industries (e.g. Unger *et al.*, 2000) and the food industry (e.g. Abdul Ghani *et al.*, 1999).

An important area of CFD exploitation in the process industry concerns the simulation of *multiphase flows.* Better understanding of gas-solid and gas-liquid-solid flows in chemical reactors is one of the major issues in chemical industry (Sundaresan, 2000): applications include critical processes such as fluidised beds, bubble columns, production of solid suspensions, etc. The most practical way to simulate hydrodynamics is through continuum models that treat the coexisting phases as impenetrating continua. This is a convenient approach in the sense that several analogies can be found between different systems (Krishna *et al.*, 1998). Other more complex models have been developed (Li *et al.*, 1999, Pain *et al.*, 2001) to address some of the specific issues which affect the simulation of multiphase flows. Nonetheless, many issues are yet to be solved such as the treatment of coalescence and break-up of bubbles or the handling of gas-liquid-solid systems.

Recently, CFD techniques have been applied to complex flow problems where other phenomena have been incorporated. In particular, there are some first attempts aimed at building a link between CFD and precipitation. Wei and Garside (1997), Al-Rashed and Jones (1999), and Baldyga and Orciuch (2001) include a precipitation model writ-

ten in terms of the moments of the crystal size distribution within a CFD model. Although the incorporation of a full population balance was not feasible, these papers demonstrate the possibility to model complex phenomena within CFD, at least for steady-state (Wei and Garside, 1997, Baldyga and Orciuch, 2001) or very short dynamic simulations (Al-Rashed and Jones, 1999).

CFD packages are very specialised and efficient numerical solvers for a special set of equations. However, they do not have the flexibility to include complex user-defined set of equations. Hamill and Baché (1998) point out that *complex systems of stiff differential equations cannot be handled* by CFD codes. For instance, in the case of combustion models the set of reactions is simplified and averaged to be described by CFD simulators (e.g., Gosman, 1999b, Eaton and *et al.*, 1999, Wild and Boysen, 1995). The *high difficulty in solving highly non-linear rate terms* with differing time scales and in dealing with the interaction of turbulence mixing leads to solution approaches capable of solving only specific sets of equations within a well-defined time scale (Eaton *et al.*, 1999). One additional limitation is pointed out by Birtigh *et al.* (2000): most CFD packages are developed to solve one special problem but their use cannot be extended further, because they *lack the flexibility to interface with libraries and models* already available in the process industry.

1.5 An Overview of Numerics

A rather simplistic but effective distinction between process modelling tools and CFD packages is that the first are designed to deal with very general models (the objective is to produce solutions for any set of user-defined equations), whereas the latter are highly tailored to obtain the best solution to a well-defined but restricted set of equations.

Thus, it is clear that the modelling approaches used in process modelling and CFD are fundamentally different. General modelling languages (§ 1.3) are not restricted to specific categories of modelling problems and applications. The set of modelling equations are defined by the user (although model libraries may be available) according to the problem specific requirements. As a result, the type of equations which need solving cannot be classified into a single class: equations may be algebraic and differential, elliptic, parabolic or hyperbolic, and so on.

On the other hand, CFD models always consider the representation of fluid flow. The model core is constituted of the set of continuity and momentum equations: by setting the fluid flow characteristics (e.g. incompressible or compressible, laminar or

turbulent, etc.) the user triggers the most suitable model defined within the package. Although additional phenomena (energy balance, multiphase and reactive flows) have been incorporated within codes, that did not change the general approach: the number of available models is *limited* and set within the package[1].

The above modelling distinctions have generated a different approach to the solution algorithms which present fundamental differences in the numerical methods.

1.5.1 Process Simulation Solution Methods

Process simulation deals with the solution of mixed systems of integral, partial differential, and algebraic equations (IPDAEs). The last generation modelling tools demonstrate a good capability of solving a wide range of those systems. The systems of IPDAEs are usually solved using the *method of lines* (MOL) family of numerical methods (Schiesser, 1991). The spatial domain is discretised leading to a system of time-dependent ordinary differential equations and algebraic equations (DAEs). Equation-oriented modelling tools solve the set of DAEs by *direct integration*: equations are solved simultaneously by the same numerical algorithm, usually based on backwards differentiation formulae (BDF) plus (modified) Newton-Raphson methods for the solution of non-linear equations, with special treatment of initialisation and discontinuity (a review is given in Pantelides and Barton, 1993). Structural decomposition techniques are adopted to take advantage of the highly sparse and irregular matrix structure. These methods are capable of solving highly non-linear stiff equations and thus are one of the most general choices to solve any system of equations. Only the spatial domain is discretised and, thus, the integration time step for dynamic systems may be adapted depending on stability and accuracy criteria.

However, notwithstanding much progress in modelling tools for the description of general PDAE systems, the mathematical infrastructure is still in its infancy. A proper general treatment of initial and boundary conditions is still under investigation and, accordingly, no completely automatic handling of complex geometry seems possible (Neumann, 2001). Furthermore, the general numerical approach in process simulation requires direct matrix calculations associated with a very high memory consumption, especially due to large number of equations arising in complex distributed systems (Neumann, 2001).

---

[1]Many commercial CFD packages allow the setting of user-defined properties (e.g. kinetics) by means of external procedures (e.g. the *User Defined Subroutines* in the Fluent package). However, that does not change the software structure since the number and type of properties and parameters which may be set are hard-coded in the CFD package and the user has to provide the numerical methods to solve the set equations implemented within the external procedures.

1.5.2 CFD Solution Methods

The most common types of equations arising in fluid dynamics are conservation equations for mass, composition, momentum and energy. They constitute a specialised system of partial differential equations. CFD has been developed to solve those types of PDEs in a complex geometry. CFD packages commonly adopt the discretisation of both temporal and spatial domains and the time step Δt is user-specified and constant leading to a set of purely nonlinear algebraic equations (Neumann, 2001). The CFD approach leads to a lack of flexibility and robustness when dynamic simulations are required: in general, CFD tools are used to model steady-state phenomena. On the other hand, discretisation of both domains allows the flexible adaptation of the step size in the spatial domain. The most common discretisation approaches are the Method of Finite Elements (FE) and the Method of Finite Volumes (FV). More details about these and other methods may be found, e.g., in the books of Fletcher (1991). Here only the basic definition of the FV method (used in most commercial packages) will be given.

The FV method uses the integral form of balance equations as its starting point. The solution domain is subdivided into a finite number of contiguous control volumes (*cells*) and the balance equations are applied to each control volume. At the centroid of each control volume lies a computational node at which the variable values are calculated. Interpolation is used to express variable values at the control volume surface in terms of the centre values. Surface and volume integrals can be approximated using suitable quadrature formulae. As a result, one obtains an algebraic equation for each control volume.

Both the FE method and the FV method are suitable for irregular computational domains and both methods can also be applied with generalised coordinates (Fletcher, 1991). The finite volume method has the additional advantage of discretising directly the conservation form of the governing equations. This implies that the discretised equations preserve the conservation laws, allowing for the discretisation error.

The discretisation process introduces an error that can be reduced, in principle, by refining the grid until the discrete equations are faithful representations of the governing equations. In general, it is desirable to arrange the mesh so as to produce fine and regular cells in the regions of major interests and close to boundaries (Fletcher, 1991).

Most CFD codes exploit the special structure of the arising equations, which are mainly the Navier-Stokes equations plus energy balance. In several cases special approaches have been defined to solve set of equations describing a specific type of flow.

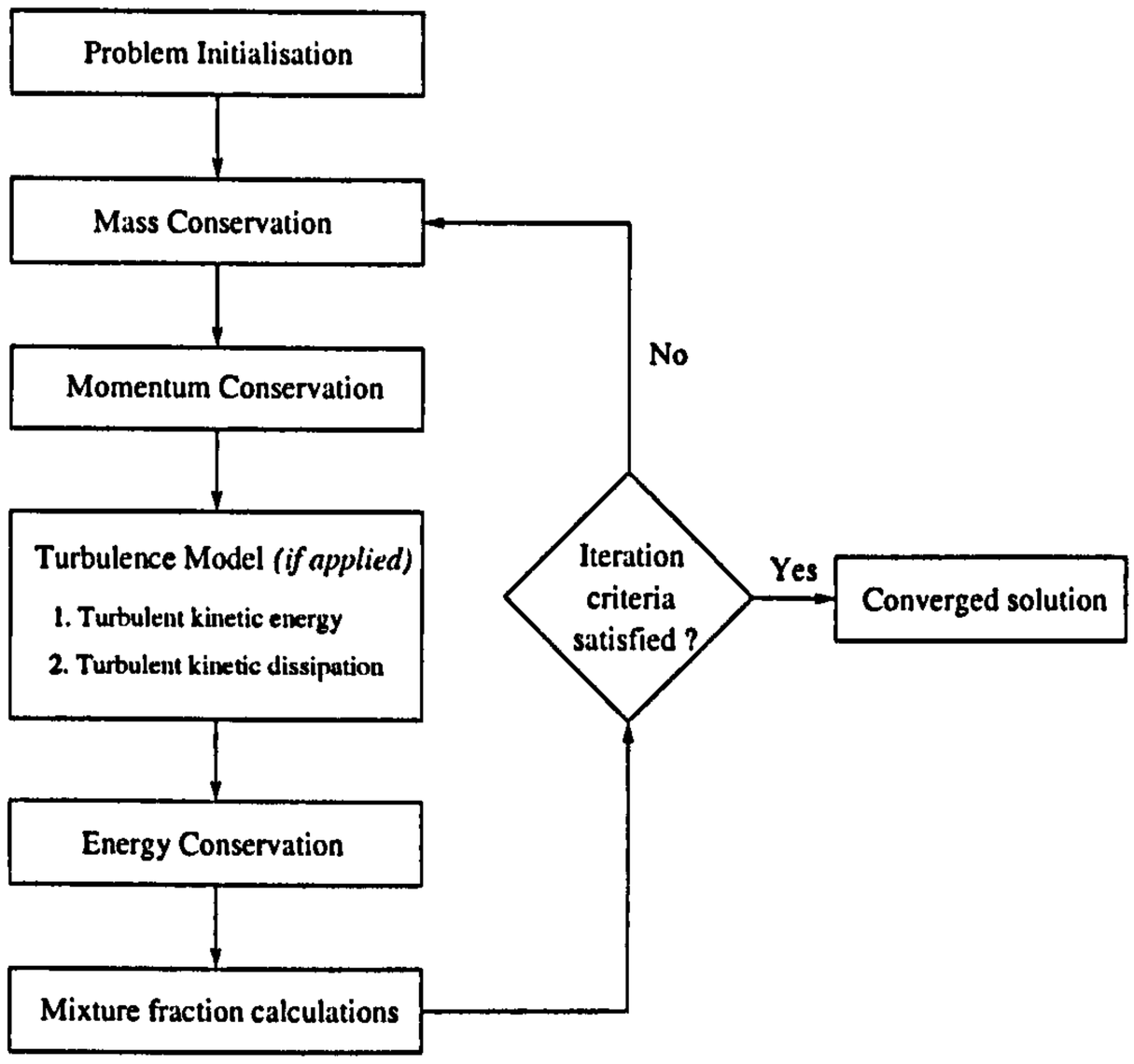


Figure 1.5: Model solution sequence for a reactive process.

CFD codes rely on iterative solution strategies to obtain an approximate solution to the set of discretised equations. This requires much less effort than a direct (exact) solution, but attention has to be paid to solution accuracy and convergence at each iterative stage. Most codes use a method called the *Semi-Implicit Method for Pressure Linked Equations* (SIMPLE) developed by Caretto *et al.* (1972) to solve the Navier-Stokes equations.

The use of highly tailored solution methods make CFD codes very impervious to the introduction of new sets of equations. For instance, Eaton *et al.* (1999) point out that if complicated reacting systems need describing, then a set of submodels and associated solvers has to be coupled to the main CFD methods. The reason for this division is that each of these specialised sets of equations requires a different numerical solution approach. Most solution strategies (Eaton *et al.*, 1999) adopt a scheme as shown in Figure 1.5 to arrive at a converged solution. As a result, the solution procedure becomes slower as well as more unstable. Large and stiff reactive systems cannot be handled and alternative models such as the use of a probability density function (PDF) are necessary (e.g. Baldyga and Orciuch, 2001).

1.6 Combined Strategies

It has to be recognised that any aggregation operation involves an inherent approximation. As our demands for model accuracy at a given scale (e.g. the mesoscale of

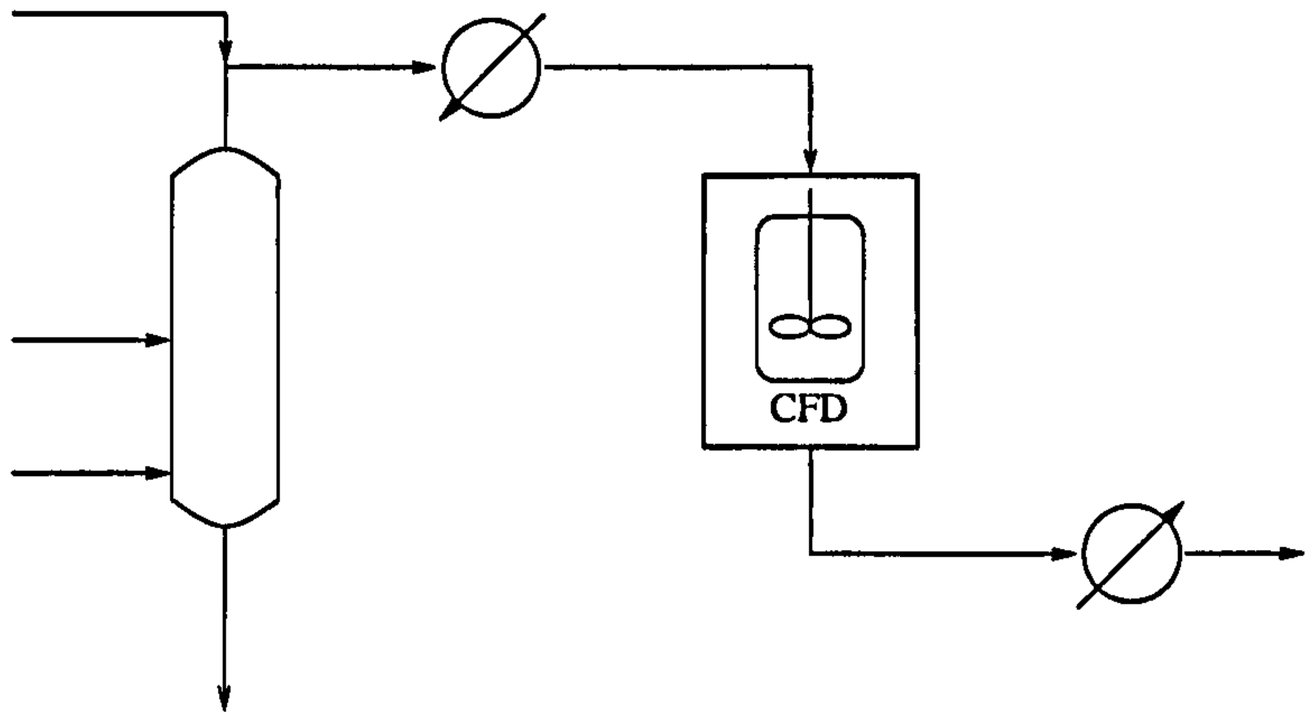


Figure 1.6: CFD models as unit operations in a process simulation flowsheet.

process equipment) become more stringent while the predictive accuracy of models at lower scales improves, there often comes a point at which the loss of useful information involved in the aggregation of lower-level behaviour becomes unacceptable. In recent years some work has tried to improve the modelling capabilities of CFD and process simulation by incorporating additional information and models.

A first approach is the incorporation of CFD models as unit operations within a process simulation flowsheet. Some process simulation (of the modular type) and CFD companies have moved together for the definition of more comprehensive design and simulation tools (e.g. Shanley, 2000, FluentNews, 2000) allowing unit operation coupling. We define this integration strategy as *spatial partitioning*. The coupling involves the exchange of input/output stream information only and the use of the process simulation physical property libraries to improve the fluid description in the CFD simulation. According to this approach, process simulation and CFD unit operation models are used separately and the only connection is obtained by common inlets/outlets (illustrated in Figure 1.6).

A very different approach is to split the constituent equations of a single process system into two submodels. The first is solved within a simulation package, the second in a CFD package. We define this approach as *model partitioning*. More details about spatial and model partitioning are given in chapter 2.

Here we will consider some works which considered the issue of incorporating hydrodynamics within process models in some critical fields of the process industry. Some of these works adopt a clear model partitioning approach; others mainly rely on some kind of model discretisation in order to describe the fluid flow behaviour.

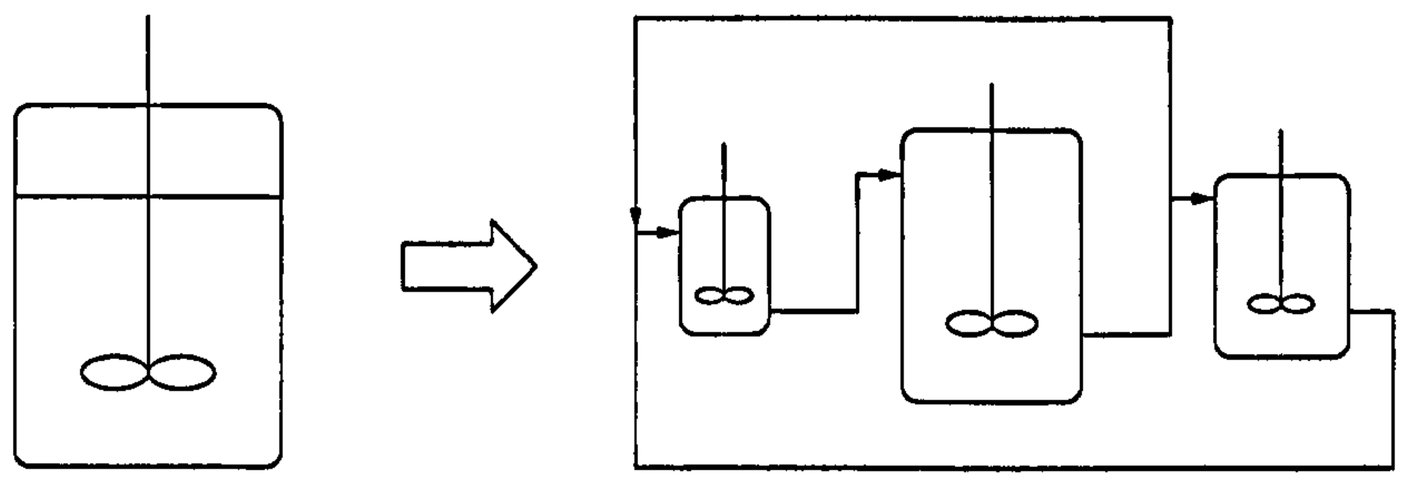


Figure 1.7: Modelling polymerisation using a three-compartment model after CFD simulation.

Polymerisation

Vivaldo-Lima *et al.* (1998) simulated a suspension polymerisation in a tank reactor by means of a compartment mixing model, i.e. a model within which a single unit operation is described by means of two or more models representing different mixing regimes. In this section the terms *compartment*, *region*, *zone* will be used with an equivalent meaning, with the choice based on the terminology used in the original papers. According to this approach, CFD simulation is used to obtain the distribution of energy of dissipation in the vessel and, accordingly, the types of mixing regimes in it. This data is used to estimate the size and connectivity of each compartment and the value of the energy of dissipation in it (Figure 1.7). Maggioris *et al.* (1998, 2000) adopted a similar approach to take into account large spatial variations in kinetic energy in order to predict the evolution of droplet sizes in a suspension polymerisation system. CFD simulations at different agitation rates and viscosities estimate the volume of different regions and the exchange flowrate between regions. A two-region model was adopted. In these works CFD is used to detect and separate regions presenting substantial differences in terms of flow characteristics. The concept of compartments is adopted to improve the details and the structure of a process simulation model where polymerisation reactions are included. Detailed population balances are used to describe the polymer droplet sizes in each region.

Fluidised Beds and Bubble Columns

Bauer and Eigenberger (1999, 2001) developed a zone model of a gas-liquid bubble column using CFD information. The model is based upon the interplay of a simplified reactor zone model and the model of the hydrodynamics. Information about the liquid flow pattern is provided to the zone model by the hydrodynamic model, while the

hydrodynamic model obtains its required information about local mean bubble sizes and the local mass flux between gas and liquid from the zone model. No population balance is solved to compute bubble sizes; a simpler statistical approach is pursued. It is demonstrated that an iterative approach where hydrodynamics are updated yields results which are substantially different from a simulation within which hydrodynamics are assumed to be "frozen" at the initial value (Bauer and Eigenberger, 2001). Shantanu *et al.* (2000) used CFD to define a macroscopic compartment model to describe the flow pattern of a gas-solids riser. The kinetic model and the transport equations are implemented in the compartment model. Each compartment is a model either of a gas-phase or solid-phase mixing cell. Shimizu *et al.* (2000) presented a study where bubble break-up and coalescence are taken into account to evaluate gas hold-ups and gas-liquid mass transfer rates. A compartment model is applied to describe the bubble movements. Nonetheless, in this work CFD is not adopted to improve hydrodynamics modelling and the definition of the compartment model.

Crystallisation

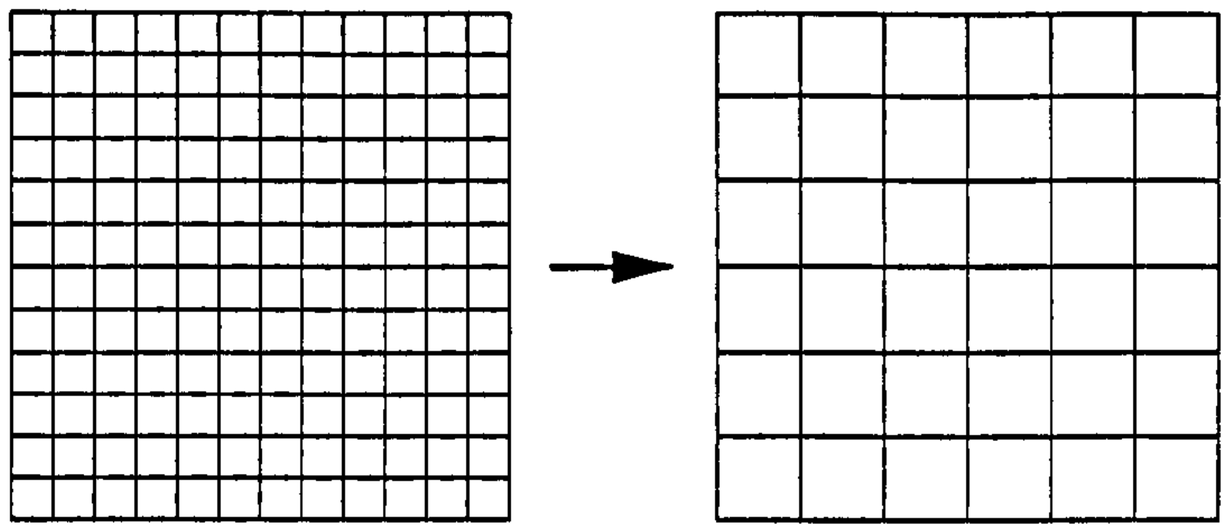
Mixing and shear stress affect both the growth and dissolution of crystals (some recent works dealing with this issue are those by Torbacke and Rasmuson, 2001, Ilievski *et al.*, 2001, Sherwood and Ristic, 2001). Kramer *et al.* (1999) and Bermingham *et al.* (2000) used hydrodynamic information to obtain a subdivision of a crystalliser main body into multiple compartments where phase equilibria, crystal population balance and energy balance are considered. Urban and Liberis (1999) developed a very complex model where CFD is used to subdivide the equipment in a network of uniformly mixed regions, within which a model with a full population balance is implemented. In this approach, results from process simulation are used to update the hydrodynamics, while the CFD solution returns the turbulent dissipation energy in each zone. The procedure is repeated until convergence is obtained.

Stirred Tank Reactors

Stirred tank reactors represent a fundamental piece of equipment in the process industry. Notwithstanding this, scale-up and modelling issues are still far from being solved. Baldyga *et al.* (1992, 1997) demonstrated that the way in which reagents are mixed may have a large influence on the product distribution of chemical reactions: they considered the effect of mixing at different scales by developing a model which takes into account micro- and meso-mixing through two parameters, the values of which depend upon the energy dissipation rate. Samant and Ng (1999) formulated a procedure for

the development of liquid-phase agitated reactors to scale up reactors from laboratory to production scale. The reaction performance in various operating regimes is estimated by macroscopic correlations (e.g. use of Damköhler numbers) and experimental data. Mann *et al.* (1982, 1984, 1987) adopted a different approach to the mixing issue. Instead of using general correlation and parameters, they divided the domain into a network of connected well-mixed zones. The network definition is based on experimental data on the hydrodynamics of stirred tank. Although micro-mixing phenomena cannot be depicted through this method, the network of zones approach provides a unique space-time description of mixing behaviour. Mann and El-Hamouz (1995) adopted the network of zones model to simulate a triplet of consecutive/competitive reactions in a batch stirred reactor. They demonstrated that imperfect mixing is responsible for great variations in the reactants/products distribution at three equipment scales.

The same approach was adopted by Vlaev *et al.* (1995, 2000) to model a stirred tank bioreactor within which Thylosin production based on the cultivation of *Streptomyces fradiae* is simulated through a network of zones. A similar method is also utilised by Vrabel *et al.* (2000) to develop a flow model based on the general knowledge of both non-aerated and aerated stirred vessels. A number of compartments (55 and 70) are used to model different impeller and reactor configurations and scales. Local gas hold-up distribution is obtained and related to the rheology of the system. Nagy *et al.* (1995) simulated a glutamic acid fermentation in a stirred tank by coupling a mixing model with fermentation kinetics. The mixing model of the reactor equipped with three turbines consists of three regions corresponding to the stirrer sections. The model takes into account the joint liquid-gas flow and the gas flow. The flow separation is made possible by dividing each region into an ideally mixed compartment and two cascades of tanks-in-series: joint flow and gas flow follow different patterns within this structure. Fermentation kinetics and mass transfer are included in the mixing model. The fermentation of glutamic acid is then simulated to show pH fluctuations at different control and scale conditions. Brucato *et al.* (1999) followed a different approach: they used two CFD simulations at different grid definitions to simulate two parallel reactions in a batch reactor. Two CFD models are used: a fine grid model simulates steady-state hydrodynamics; a coarse grid model, within which fluid flow behaviour is imported from the first model, is utilised to describe mixing and reaction dynamics in the reactor (Figure 1.8).



Steady-state grid Dynamic grid

Figure 1.8: Modelling reactions using two CFD models with different grid definition (Brucato *et al.*, 1999).

Other Applications

Two important papers using an interesting approach to CFD - process simulation integration can be found in a Mitsubishi Chemical project (Urban *et al.*, 1997), and in a paper by Marias (2000). In the first case a catalytic multitube-shell reactor was simulated by using a CFD package (CFX) and a process simulator (gPROMS). The CFD package predicts the coolant temperature within the shell, while a detailed simulation model is used to model the complex reaction scheme on the tube side. The coolant heat transfer coefficient and temperature profiles computed by the CFD package are used by the simulation model as boundary conditions. Wall temperatures on the tube section are returned to the CFD code. Marias adopted the same method to simulate a rotary kiln incinerator. A CFD code is used to model turbulent combustion and radiation in the gas phase within the combustion chamber. Process simulation is used to run a simplified model for the pyrolysis and burning of the waste bed. The coupling is performed through heat and mass transfers at the gas/solid interface.

Although some other interesting studies may be found in several fields[2] , the papers described above represent the panorama of this kind of research within process industry and demonstrate the effort which has been devoted in recent years to improve the simulation of processes where both hydrodynamics and other complex phenomena are critical for more accurate modelling. Table 1.1 summarises some of the basic references addressing the issues discussed in this chapter.

[2]We would like to mention the original application of Bush *et al.* (1998) who developed a hybrid model to study the spatial distribution of vapor uptake within the nasal cavities of rats: values for the gas mass transfer coefficients and gas flows in nasal compartments are determined by CFD simulations and then used as input to a physiologically based pharmacokinetic (PBPK) simulation of toxicant transport through tissue stacks.

Multiscale modelling	Villermaux (1995) *ref.* [154]
Process simulation overview	Pantelides and Britt (1995) *ref.* [109] Marquardt (1996) *ref.* [94] Biegler *et al.* (1997) *ref.* [17]
CFD overview	Gosman (1998) *ref.* [52] Birtigh *et al.* (2000) *ref.* [19]
Grid generation	Soni (2000) *ref.* [137] Pain *et al.* (2001) *ref.* [103]
Process simulation numerics	Pantelides and Barton (1993) *ref.* [108] Neumann (2001) *ref.* [98]
CFD numerics	Fletcher (1991) *ref.* [37] & [38]
Applications of a combined strategy	Shanley (2000) *ref.* [129] Maggioris *et al.* (1998) *ref.* [84] Bauer and Eigenberger (1999) *ref.* [14] Bermingham *et al.* (2000) *ref.* [16] Urban and Liberis (1999) *ref.* [152]

Table 1.1: Main references

1.7 Conclusions and Objectives

The previous sections showed that there are a number of existing problems in the area of CFD and process simulation that remain to be addressed, in particular:

- the strong limitations within process simulation in describing mixing and hydrodynamics:
  - ▷ modelling of 3D flows in complex geometries are difficult/impossible to describe
  - ▷ general PDAE systems may be solved in special cases only
  - ▷ direct matrix calculations require very high computational times in complex distributed systems;
- the limited flexibility of CFD packages in terms of modelling phenomena other than fluid flow behaviour:
  - ▷ highly tailored numerics restrict the implementation of generic sets of equations
  - ▷ poor stability and accuracy affect the treatment of general dynamic models
  - ▷ external libraries and models cannot be interfaced;

- the existence of many processes of industrial importance which need a better and simultaneous understanding of both hydrodynamics and other physical and chemical phenomena to properly address critical issues such as

  ▷ definition of a control strategy

  ▷ scale-up and scale-down

  ▷ equipment and process design.

Several examples have been presented demonstrating the effort and interest of the scientific and industrial community to obtain better tools to tackle those issues. The deficiencies within process simulation and CFD tools individually indicate that a combined approach is strongly needed. Many complex systems cannot be described without some kind of interaction between CFD and process simulation, in particular:

- complex reactive systems in a stirred reactor;

- polymerisation models;

- heterogeneous systems such as:

  ▷ crystallisers

  ▷ liquid-solid reactors

  ▷ fluidised beds

  ▷ bubble columns;

- bioreactors;

- multitube reactors;

This thesis continues the approach to integration which has appeared in several recent papers and explores a more general solution to some of the issues, in particular delivering:

a. the design of a general open architecture where both CFD and process simulation are introduced;

b. the definition of a general procedure to handle several categories of processes where CFD and process simulation can be used simultaneously;

c. the design of a general procedure for mapping process simulation models and CFD over highly discretised domains;

d. an approach for the solution of some robustness and efficiency issues, since the coupling of CFD and process simulation may result in convergence difficulties and a high computational effort.

In the next chapter these issues will be introduced through the more general topic of *multiscale* modelling. It will be explained how the objectives of this thesis represent a central issue in the effort to define models including the description of several phenomena at different scales. The class of problems our approach aims to address will also be specified. A first example will illustrate the general design concepts and the benefits from the proposed architecture.

Chapter 3 is concerned with the definition of a general architecture to obtain the integration between a CFD and a process simulation model. The design issues will be discussed as well as the required assumptions.

Chapter 4 will deal with some important numerical aspects which arise when incompressible fluids are considered. A solution will be proposed.

Chapter 5 will take into account the problem of setting up a network of *zones* to obtain a closer coupling between CFD and process simulation.

Chapter 6 will consider procedures for automatically applying the zoning methods described in the previous chapter. Some applications illustrate the approach.

Chapter 7 is concerned with efficiency of the numerical calculations. The definition of local models and the numerical methods which are needed to estimate their parameters will be considered. The performance of the local models in some specific application examples is illustrated and discussed.

Chapter 8 will consider a comprehensive example where the suggested architecture and all solution techniques are implemented and demonstrated. The unique benefits will be illustrated.

Finally, conclusions and future areas to be exploited will be considered.

Chapter 2

Multiscale Process Modelling

In the previous chapter some of the issues related to process modelling and an overview of the state-of-the-art CFD and process modelling tools have been discussed. In this chapter the problem of the integration of CFD and process simulation within the more general class of multiscale modelling[1] is treated in more detail.

2.1 Multiscale Modelling

In the previous chapter (§ 1.2) the problem of multiscale modelling within the process industry was introduced and the need to consider phenomena occurring at different scales was pointed out. As was noted, the traditional approach in dealing with such interactions has largely been based on *scale aggregation.* For example,

- the nanoscale's detailed descriptions of the behaviour of matter are aggregated into equations of state and relatively simple kinetic laws so that they can be used in higher-level models;

- the complexities of fluid flow at the microscale are aggregated into a well-mixed region (or network of well-mixed regions) or residence time distribution approximations used for modelling process equipment in higher scale models;

- the details of dynamic behaviour of batch processing equipment studied at the mesoscale are replaced by the simple concept of a task with a finite duration and fixed demands on resources, of the type that can be used for plant scheduling;

- the large networks of interacting resources and tasks used for modelling multipurpose plants at the macroscale are replaced by a few simple linear constraints

[1] I am deeply indebted to Prof. C.C. Pantelides for the suggestions and material regarding multiscale modelling.

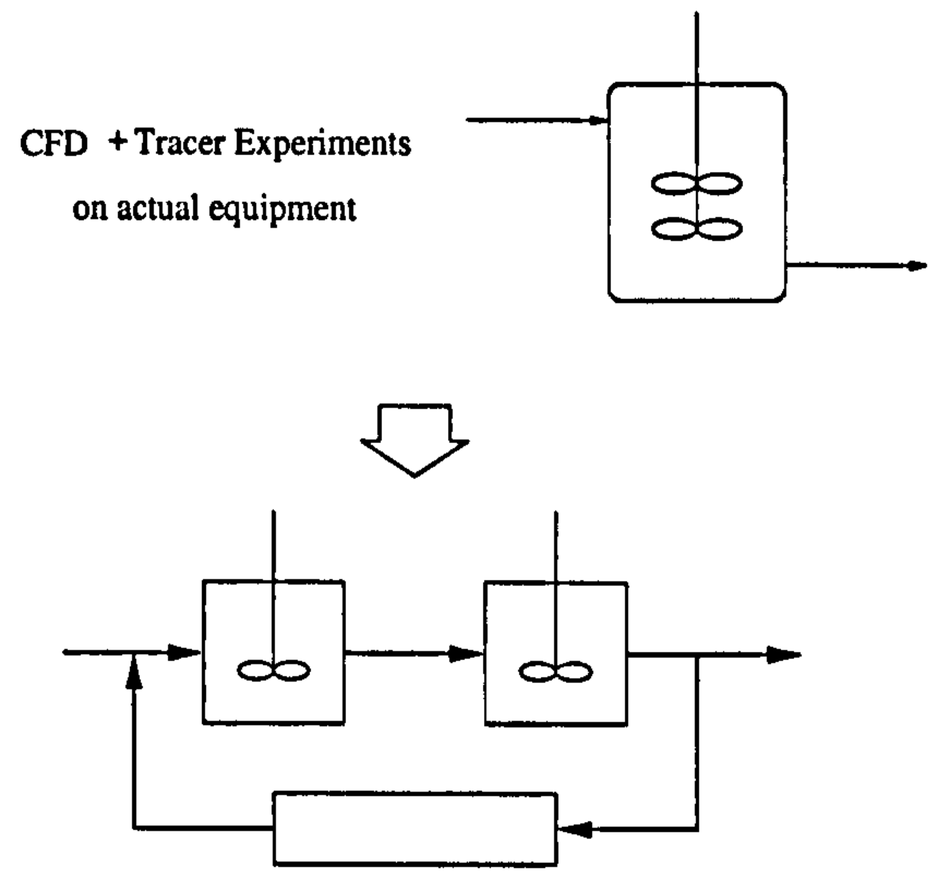


Figure 2.1: *Scale Aggregation.*

describing overall production capacity for the purposes of modelling supply chains involving interacting manufacturing and distribution operations.

The scale aggregation approach has proven very successful in handling the inherent complexity of process engineering, representing a pragmatic trade-off between the predictive accuracy of a model and its complexity at both the conceptual and the computational levels. Figure 2.1 illustrates the scale aggregation in the design of a model for a stirred tank reactor. Experiments and/or CFD simulation provide information regarding the fluid flow behaviour in the tank. Such information is used to model the reactor by a network of some well-known (and simpler) reactor patterns (e.g. well-mixed and plug-flow). Some of the works considered in § 1.6 may be included in this category. For instance, the zone approach developed by Mann *et al.* (1982, 1984, 1987) may be regarded as a state-of-the-art scale aggregation approach to incorporate hydrodynamics within a process simulation model. Experiments and CFD simulations are the tools used a-priori to set up a highly structured network which attempts to reproduce the complexity of fluid flow behaviour in a stirred tank.

However, in many cases (§ 1.6) it has become important to consider a deeper link between scales in representing industrial processes: hence, a demand for *scale integration.*

2.1.1 Scale Integration

Scale integration involves the use of description of phenomena at different scales within the same model. At present, the most common ways of achieving this are the so-called serial and parallel integration strategies (Maroudas, 2000).

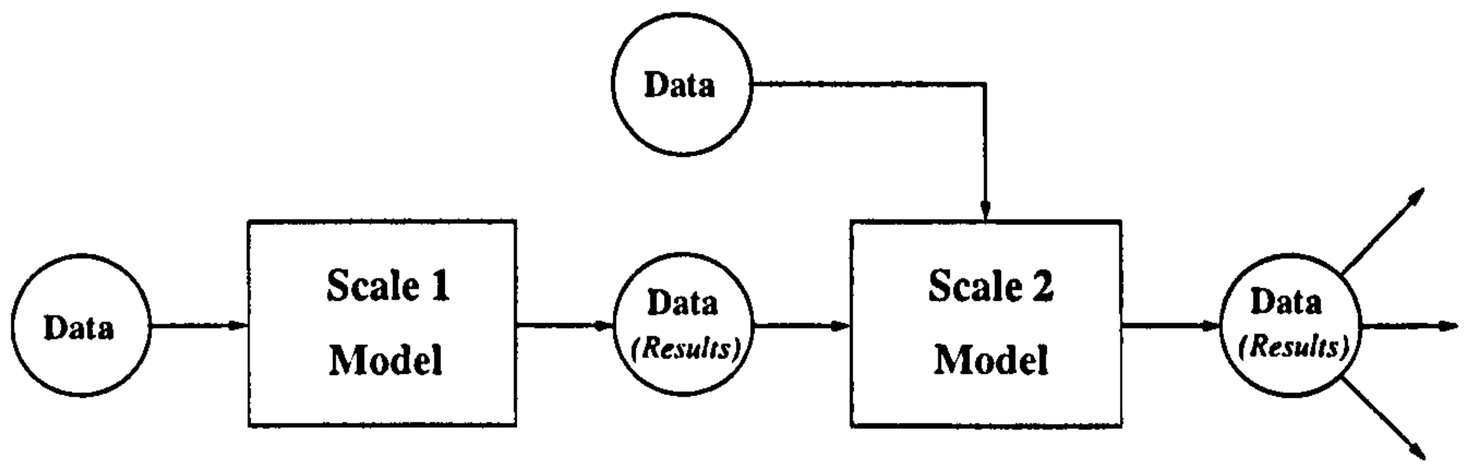


Figure 2.2: *Serial Integration.*

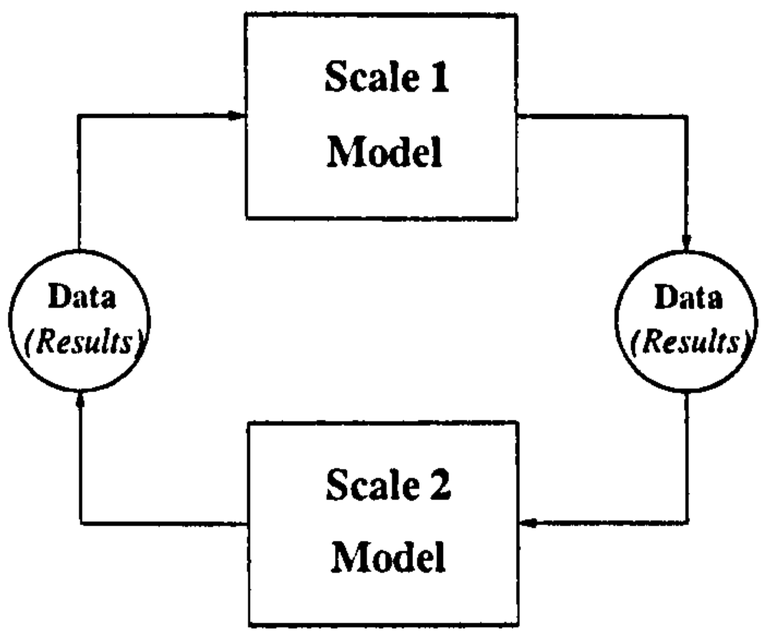


Figure 2.3: *Parallel Integration.*

A *serial integration* strategy is one in which the finer scale model is simply used to generate some of the parameters of data required by the higher scale one (Figure 2.2). This is done once a-priori and in a *sequential* manner (no iteration). This is not very different from the scale decoupling approach except that the aggregate description (e.g. the equation of state) is more formally derived from the lower-scale (rather than, for instance, being determined empirically by fitting of experimental data). The works by Vivaldo-Lima *et al.* (1998) and Maggioris *et al.* (1998, 2000) in polymerisation and by Kramer *et al.* (1999) and Bermingham *et al.* (2000) in crystallisation (see § 1.6) belong to this category. As in the zone approach by Mann *et al.* (1982, 1984, 1987), a CFD simulation is used to set a network which approximates the hydrodynamics in the equipment; however other critical parameters (e.g. value of the energy of dissipation) are estimated and included in the process simulation model.

A more advanced approach is achieved when a *parallel integration* strategy is adopted. Parallel integration involves the *simultaneous* use of descriptions at different scales applied to the same computational domain. The results of one description form inputs to the other and vice-versa (Figure 2.3). Thus, an iteration between the two models is normally required to achieve a consistent overall description. In § 1.6 there are a few examples where the combination of CFD with process modelling tools is

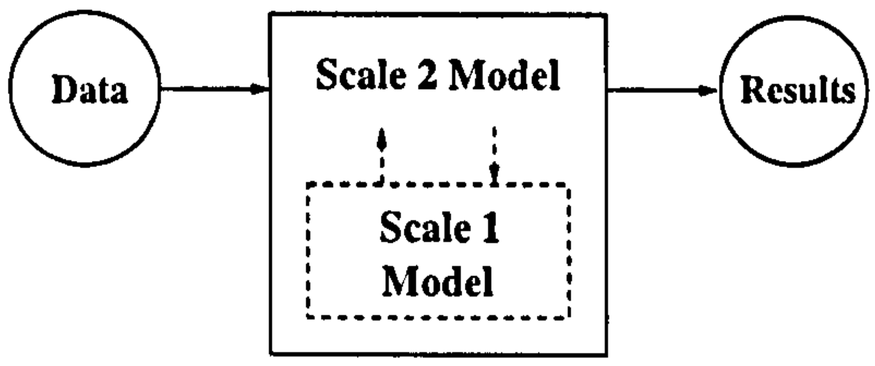


Figure 2.4: *Hierarchical Integration.*

obtained by means of a parallel strategy. Bauer and Eigenberger (1999, 2001) for bubble columns, Urban and Liberis (1999) in the case of crystallisation and Marias (2000) to simulate a rotary kiln incinerator adopted a procedure where the CFD model is periodically updated thanks to information derived from the process simulation model and vice-versa. In addition to the serial and parallel strategies mentioned above, we could also envisage a third *hierarchical integration* strategy (Pantelides, 2000) in which the finer-scale model is formally embedded within the higher-scale model to represent a set of relations among macroscopic quantities occurring in the latter (Figure 2.4). Nonetheless, we prefer including hierarchical integration within the larger class of parallel integration. Although more details will be added in chapter 7, the main distinguishing feature of hierarchical integration is the approach to the problem of coordinating the two parallel models (one the main issues addressed by this thesis). In general, parallel integration considers two models requiring specific numerics to be solved. Integration of process simulation and CFD calls for the coordination of two different solvers in order to achieve a converged comprehensive solution. In particular, it needs addressing issues of:

- initialisation
- sequence of calculations
- methods to achieve convergence.

From now on the term *hierarchical integration* will be used only to stress the fact that *parallel integration* is achieved in the manner described above. It does not imply an alternative and independent approach.

Finally, we mention the possibility of a *simultaneous strategy* in which the higher-scale model is formed utilising directly the finer scale descriptions. For example, advances in numerical methods for dynamic simulation of increased computing power now routinely allow us to build dynamic (macroscale) models of individual equipment items

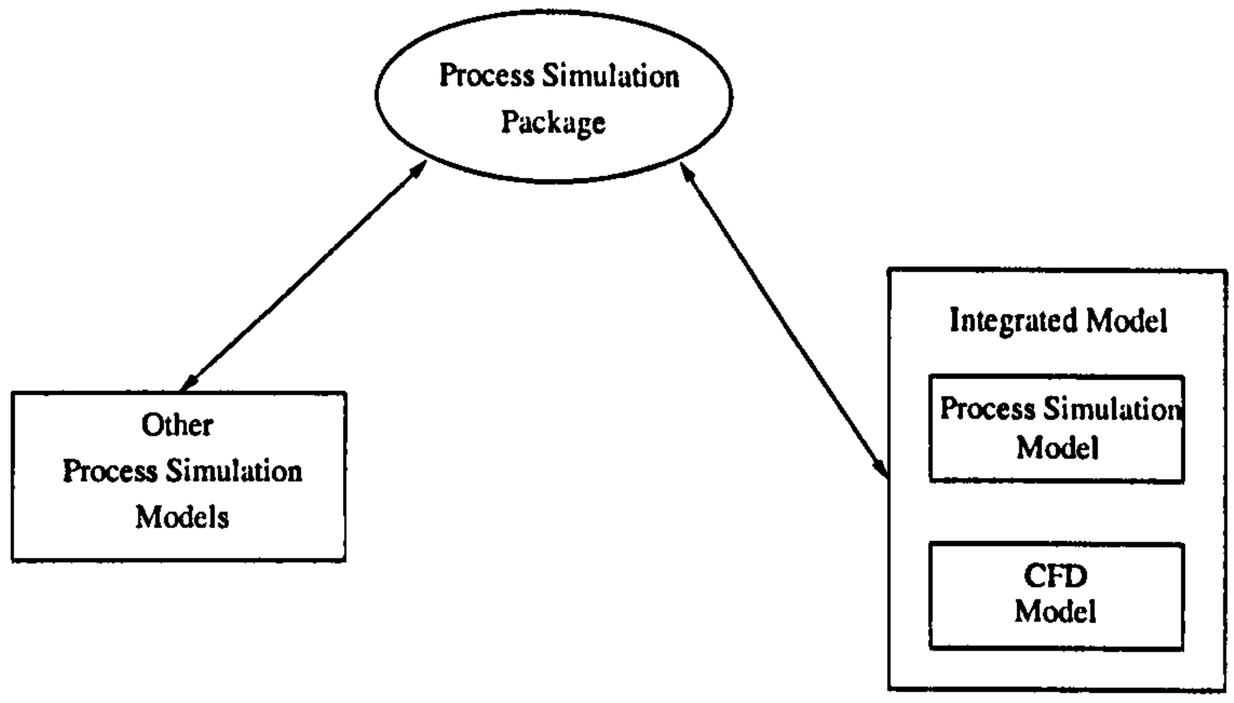


Figure 2.5: Control of the simulation is held by process simulation package.

without the need for any simplification at this stage. Some initial techniques to deal with CFD and process simulation according to a simultaneous strategy approach are discussed by Neumann (2001). However, this is at present computationally prohibitive.

2.2 Integration of CFD and Process Simulation

In this thesis we follow the hierarchical/parallel approach to the integration of CFD and process simulation, based on some premises which are explained in the following subsections.

2.2.1 Hierarchy in the Integrated Model

The overall software architecture is designed such that the control of the simulation is held by the process simulation package (Figure 2.5). If we refer to Figure 2.4, the *Scale 2 Model* is represented by the process simulation package, while the *Scale 1 Model* is a CFD package. A justification for such a design is that, in addition to the region that is common to both the process simulation model and the CFD model, the former model will often also include other entities (e.g., other unit operations) that are completely irrelevant to the latter; thus, the process simulation tool has a wider view of the process being modelled, and it alone has responsibility for the interactions among these regions. According to the hierarchical approach, the CFD calculation may be viewed as providing certain well-defined calculation services to the process simulation model. The CFD tool becomes a provider of fluid dynamical services in much the same way as a thermo-physical property package interfaced to the process simulation tool provides thermodynamic services to it.

As mentioned, the process simulation tool used for this work is the commercial

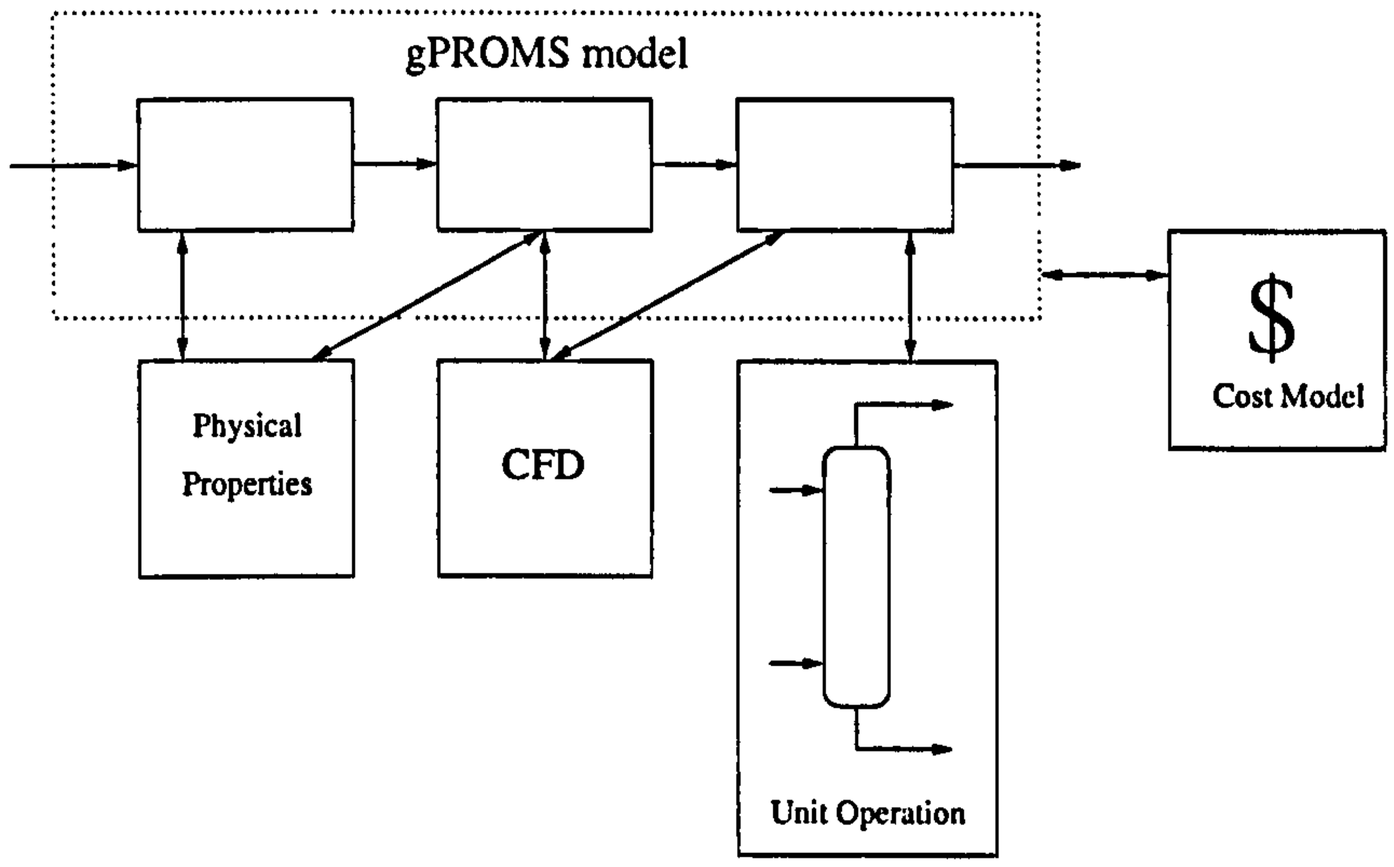


Figure 2.6: The gPROMS Foreign Object Interface.

package gPROMS. In the context of this work, gPROMS is an interesting choice because it itself possesses a substantial capability for modelling systems where properties vary with spatial position as well as time. Its functions already overlap to a certain extent with that of CFD packages. However, gPROMS is currently limited to simple, regular geometries (see chapter 1); therefore, there are still benefits to be gained by coupling it to a CFD package.

From a practical point of view, a major advantage of using gPROMS is its open architecture that allows:

- external software to be incorporated within gPROMS;

- gPROMS itself to be incorporated within other software.

Of particular interest to us is the gPROMS' Foreign Object Interface (gPROMS Advanced User's Guide, 1999). A "foreign" object (Kakhu *et al.*, 1998) is simply an external piece of software that provides certain computational services to native gPROMS models; examples of such foreign objects (Figure 2.6) include a physical property package computing thermodynamic and transport properties, a spreadsheet carrying out costing calculations, or a software module written in a conventional programming language (e.g. FORTRAN) to simulate an individual unit operation (e.g. a distillation column).

The Foreign Object Interface is a general protocol that gPROMS employs for all its communication with foreign objects of whatever type. The actual communication between gPROMS and the external software is implemented using "middleware" based on CORBA (Common Object Request Broker Architecture) which allows the two items

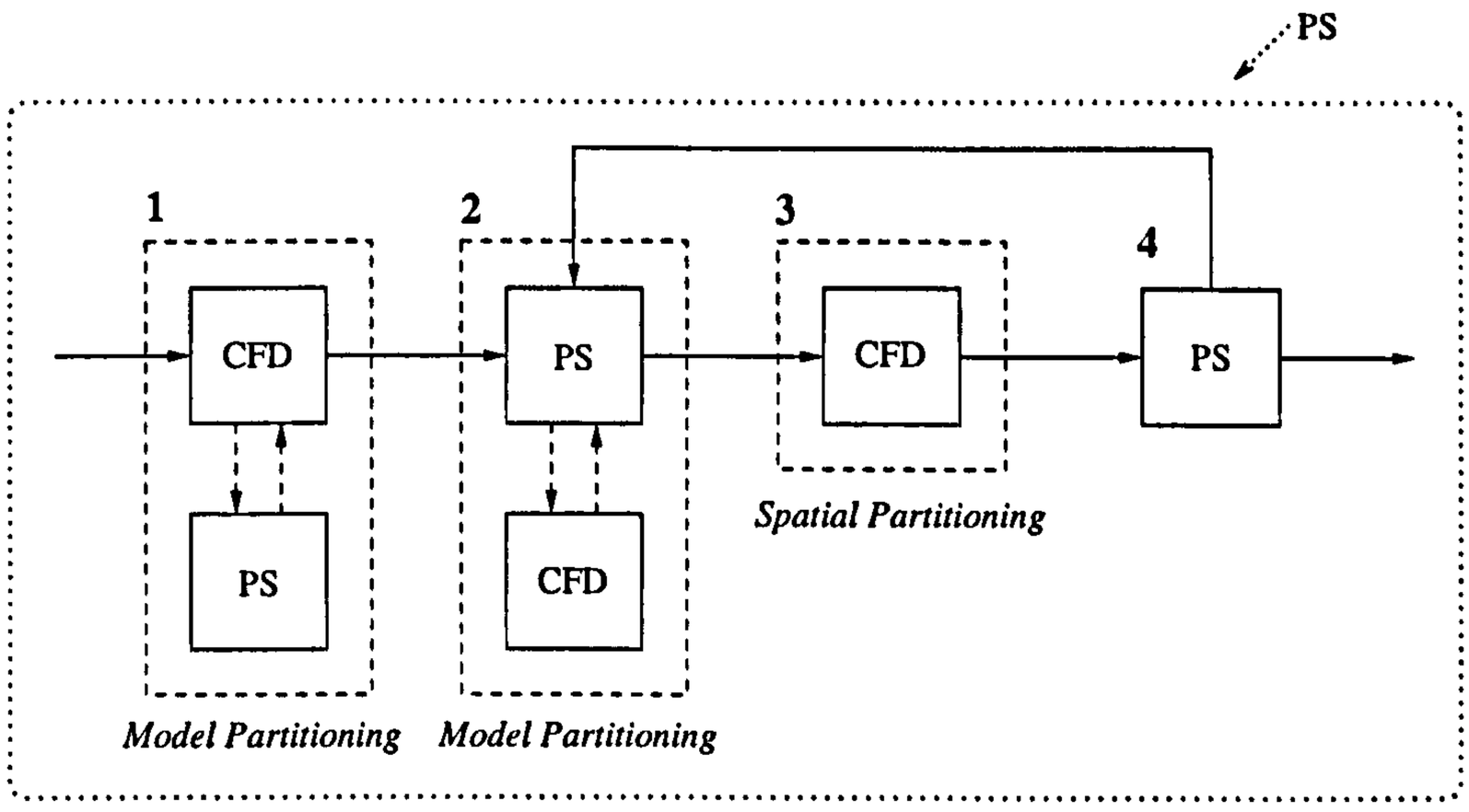


Figure 2.7: A CFD - process simulation model, where both *spatial* and *model* partitioning definitions are applied (PS stands for Process Simulation).

of software to communicate in a manner which is completely transparent to the user even if they are executing on quite different computers linked by a network. This is especially important for the purposes of the interface considered in this thesis as CFD software may run on specialised hardware. The possibility of incorporating a variety of "objects" in other traditional simulation software has recently been enhanced by the development and definition of open standards (e.g. CAPE OPEN).

Most numerical algorithms for dynamic and steady-state simulation make use of the values of the partial derivatives of the model equations with respect to the variables occurring in them. The complexity of CFD calculations is such that it is not possible to supply the exact values of the derivatives of the outputs which need computing. Nonetheless, the gPROMS's interface is capable of numerically generating approximations of the partial derivatives of the functions (Kakhu *et al.*, 1998), thus avoiding the implementation of special procedures to estimate the required derivatives.

2.2.2 Common Spatial Domain

Let us consider Figure 2.7. A unit operation network is represented. The network is described by a process simulation model. One unit operation (No. 4) is simulated via process simulation. The remaining models are described through the concepts of *spatial* and *model* partitioning as defined in § 1.6. Unit 3 represents a typical case of spatial partitioning: only input/output stream information is exchanged between a CFD model and other unit operations by means of the process simulation network model. Unit 1 and 2 are described according to a model partitioning approach. Similarly to Unit 3, Unit

1 stream information is exchanged between a CFD model and other unit operations; however, in this case a process simulation model is used to describe some of the process phenomena (e.g. reactions or physical properties) within the unit operation model. Unit 2 is inversely modelled: the model equations are split into two submodels, but it is the process simulation model handling the stream information from/to other unit operations.

This thesis will focus on the model partitioning issue as illustrated by Unit 2 in Figure 2.7. In fact, even spatial partitioning may be interpreted as a model partitioning case within which the process simulation model is "empty" and just provides the links to other units (or portions of equipment) in the process network. We also assume that the process simulator may be used to model portions of equipment or processes that do not involve CFD modelling. On the contrary, we will not consider the case illustrated by Unit 1, i.e. when process simulation is incorporated within a CFD model.

According to the model partitioning assumption, both the CFD package and the process simulation package model the *same spatial region* (e.g. the interior of a process equipment unit) using different approximations: the CFD model will use a fine-scale grid to obtain a local solution for the velocity field; the process simulation package will consider physical phenomena at a more macroscopic level.

Later in this chapter an example will be given where the model of a reactor is partitioned between a homogeneous process simulation model and a grid-discretised CFD model. The following chapters of the thesis will be dedicated to a more sophisticated approach within which model partitioning is obtained through a discretisation of both process simulation and CFD domains.

2.2.3 *Weakly-coupled* Models

Model partitioning is a decoupling procedure whereby it is assumed that phenomena at different scales may be solved by independent models exchanging critical parameters. Given a general model

$$F(x_1, x_2, \ldots, x_{n'}, x_{n'+1}, \ldots, x_{n''}, x_{n''+1}, \ldots, x_n) = 0 \tag{2.1}$$

representing a set of phenomena, it is assumed that the model may be split into two separate models (CFD and process simulation):

$$\begin{cases} F_{CFD}\,(x_1, x_2, \ldots, x_{n'}, x_{n'+1}, \ldots, x_{n''}) = 0 \\ F_{PS}\,(x_{n'+1}, \ldots, x_{n''}, x_{n''+1}, \ldots, x_n) = 0 \end{cases} \tag{2.2}$$

having in common a (small) subset of variables $x_{n'+1}, \ldots, x_{n''}$. The two models are solved separately, each using the most suitable numerical solution method. The general solution is obtained updating the two models by means of common variables $x_{n'+1}, \ldots, x_{n''}$ (parallel integration). The flux of information and procedures to obtain a common solution are managed by the process simulation package which adopts its own numerical methods (Newton-Raphson and quasi-Newton methods) to solve the overall problem.

However, there are cases where the interdependence among different phenomena is such that no sensible decoupling is feasible. Many polymerisation reactors cannot be modelled without a simultaneous description of fluid flow behaviour and kinetics. The system rheology presents complex computational issues (Keunings, 2000) and even a clear definition of the different scales within a process is not viable (e.g. the wide distribution of molecule sizes do not allow a separation among the scale of kinetics, micro- and macro-flow).

Evaluating the dependence among different phenomena is not always easy. We will distinguish two limit classes of problems depending on the degree of interdependence between hydrodynamics and other physical and chemical phenomena. On the one side we consider *strongly-coupled* systems, i.e. systems within which a separation of fluid dynamics equations from other equations is very difficult or impossible. On the other side, we define *weakly-coupled* systems, i.e. systems demonstrating a clear separation between different scales and phenomena. We point out that such categorisation is a clear simplification since systems do not necessarily belong to either one class or the other, but present a wide range of degrees of interdependency going from one side of the spectrum to the other. In general, this is still an open issue and in the future it will be desirable to develop practical procedures to identify strongly-coupled and weakly-coupled systems.

As an example, let us consider a stirred reactor and suppose a reaction with the following characteristics is occurring in the tank:

a. batch process

b. very dilute solution

c. initial uniform composition throughout the domain

d. negligible heat of reaction

e. no interaction between hydrodynamics and reaction rate

In this process, the fluid flow behaviour does not affect the yield of the process, and composition and reaction rate do not alter the hydrodynamics. Process modelling and CFD tools may be used independently of each other without incurring any simplification or mistake.

Now let us remove one of the above conditions and assume, e.g., that one of the reactants is locally injected into the tank, i.e. the initial composition is not uniform. It is clear that in this case yield and composition distribution depend on mixing. The use of CFD may be employed to obtain hydrodynamics data so as to describe mixing in the tank. Unless more complicated phenomena such as micromixing need describing, this a case of a weakly-coupled system: CFD results may be obtained independently and then incorporated in a process simulation model.

If the "dilute solution" hypothesis is also removed and it is assumed that composition affects, for instance, the fluid viscosity, then the system becomes more complicated since the hydrodynamics needs updating too, depending on physical properties which are correlated to composition and, accordingly, to the reaction model. This process is certainly more strongly coupled than the previous one. However, if the *phenomena described by the CFD model are much faster* than those which would be described by the process simulation model, then we may still proceed to a decoupled solution. In fact, we may check experimentally whether the time it takes for the velocity field to settle down following any external disturbance (e.g. in mixing intensity) is much shorter than the characteristic time constants associated with the mass and energy dynamics. The main consequence of this approach is that CFD calculations may be performed as (quasi-) steady-state and updated whenever required by the process simulation software (and vice-versa). This is a very important assumption which will be maintained throughout this work.

Finally, let us suppose that there also exists a correlation between reaction and hydrodynamics. It is certainly difficult to assert that the system is weakly-coupled. Nonetheless, in some instances that may well be regarded as an acceptable approximation. Several bioprocesses, industrial crystallisation, some types of suspension polymerisation demonstrate characteristics which may justify the independent handling of hydrodynamics equations and "other" modelling equations:

- great difference in time scale of the different phenomena (e.g crystal growth and establishment of the fluid velocity field);

- simple relations between different phenomena (e.g. use of parameters to describe nucleation and breakage after shear stress).

Other systems cannot be treated in this way. For instance, for many polymerisation processes the time scales of polymer formation/growth and hydrodynamics are similar. Furthermore, fluid flow behaviour cannot be accurately described without including a molecular representation of the system. Rheology, reaction rate and fluid dynamics cannot be handled separately. Even if a fully dynamic integrated model between CFD software and a process simulation tool were possible, it would be extremely hard (if not impossible) to decide which equations should be included in the CFD model and which in the process simulation one. The system is certainly strongly-coupled. The only viable approach to solve such a strongly-coupled model may well require a simultaneous solution of all equations, whose exploitation is not the objective of this thesis. Our aim is to deliver a methodology capable of dealing with weakly-coupled systems which, thus, represent the target of this research work.

2.3 A First Approach

In the following section a novel framework is presented to achieve a hierarchical integration between CFD and process simulation. The interfacing architecture proposed, and the example it will be applied to, will let us introduce some of the general ideas supporting the design and implementation of our approach.

The problem is tackled by decoupling an overall simulation model into two categories of submodels (Figure 2.5):

1. regions and/or equipment which are totally described in the simulation package

2. regions and/or equipment which are modelled through a hierarchical integration of a CFD model and a process model

From now on we will refer to the integrated models as *internal models*. The process simulation only models and all the information (properties as well as flowrates) exchanged by the internal model with the outside will be named *environment*. In this way we establish a very important separation in terms of design and implementation between the integration procedure related to the internal models and the rest according to the definition of spatial partitioning.

In the first category there will be a process simulation model connecting these submodels, including operational procedures, control systems, etc.

For each equipment in the second category, the CFD package is "wrapped" within an *object* interface that includes a number of "methods" (i.e. externally accessible procedures). Each method returns the value of a certain quantity required by the process

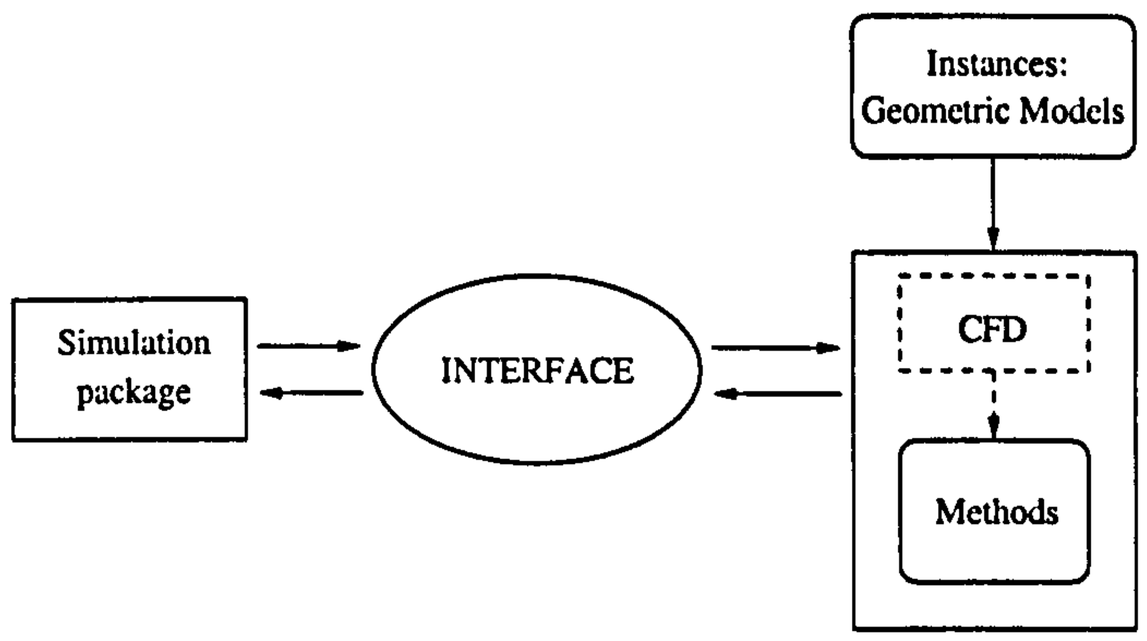


Figure 2.8: Interface flowsheet.

simulation model - its "output" - (e.g. a surface-averaged heat transfer coefficient) for given values of its "inputs". The latter typically comprises fluid properties (e.g. density and viscosity) and other important parameters (e.g. impeller speed). For the purpose of this work, a special implementation of this interface was developed.

The above protocol (Figure 2.8) essentially defines the behaviour of a general class of computational fluid dynamic foreign objects, all implemented by the same CFD package (in this work we used Fluent 4.5 by Fluent Inc., but the approach has been demonstrated with other packages). Specific instances of this class may correspond to individual items of equipment or spatial regions, each modelled separately. Each instance is characterised by its own geometry and also by the way this is represented within the CFD package (e.g. the discretisation grid). The definition of an instance in terms of this information must be declared before the combined simulation is initiated. State-of-the-art CFD packages, such as Fluent, provide pre-processors that allow the user to define the geometry and to derive an appropriate discretisation grid using a graphical user interface. Albeit not strictly necessary, one or more preliminary CFD calculations may also be carried out as part of this preparatory phase. These usually provide the user with some insight as to the behaviour of the CFD model; they also generate solution points that may be stored to be used later for providing good initial guesses to the CFD calculations that take place during the process simulation.

During a simulation, gPROMS issues calls to the various methods of the Foreign Object passing to them the current values of their inputs and requesting the values of their outputs. In the simplest implementation, each such call triggers a CFD calculation which is initialised using previously generated result points. When the CFD calculation converges, the quantity required by the method is computed by post-processing the results to obtain method results (Figure 2.8). Control is then returned to gPROMS.

Before applying the integrated model an initialisation procedure is required. In the

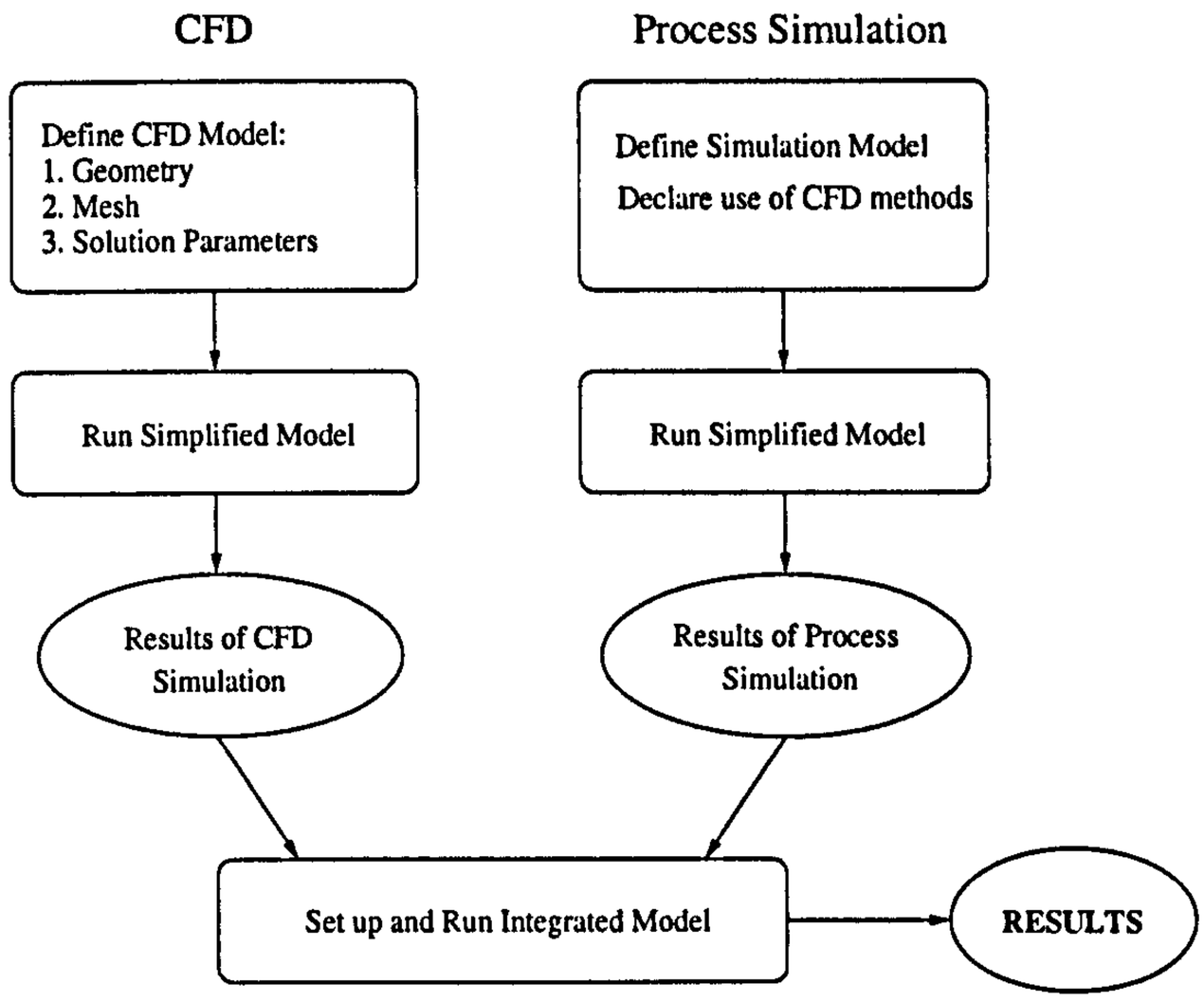


Figure 2.9: Initialisation procedure and solution.

initialisation procedure (Figure 2.9):

1a. the simulation model is set up and run using assumed hydrodynamic properties;

1b. the CFD model is set up and run using assumed physical property assumptions;

2. results from the simplified models are used to set up the integrated model (initial and boundary conditions) and to verify the adopted model partition;

3. the integrated model is run. Results from a previous CFD simulation may be stored in a database (e.g. in a file) and re-used at each iteration to save computation time during the simulation, via a hot start.

CFD simulations are run as *steady-state* cases: in fact, computational burden and some numerical issues do not presently allow the integration of dynamic process simulation with dynamic hydrodynamics.

2.4 The Example Process

The above approach to modelling is illustrated with reference to a semi-batch reactor (shown schematically in Figure 2.10) which forms part of an existing pilot plant at Imperial College, London. The tank is fitted with a double impeller centred shaft and a jacket that can be used for either cooling or heating. Temperature control in the tank is achieved by controlling the temperature and flowrate of the jacket water

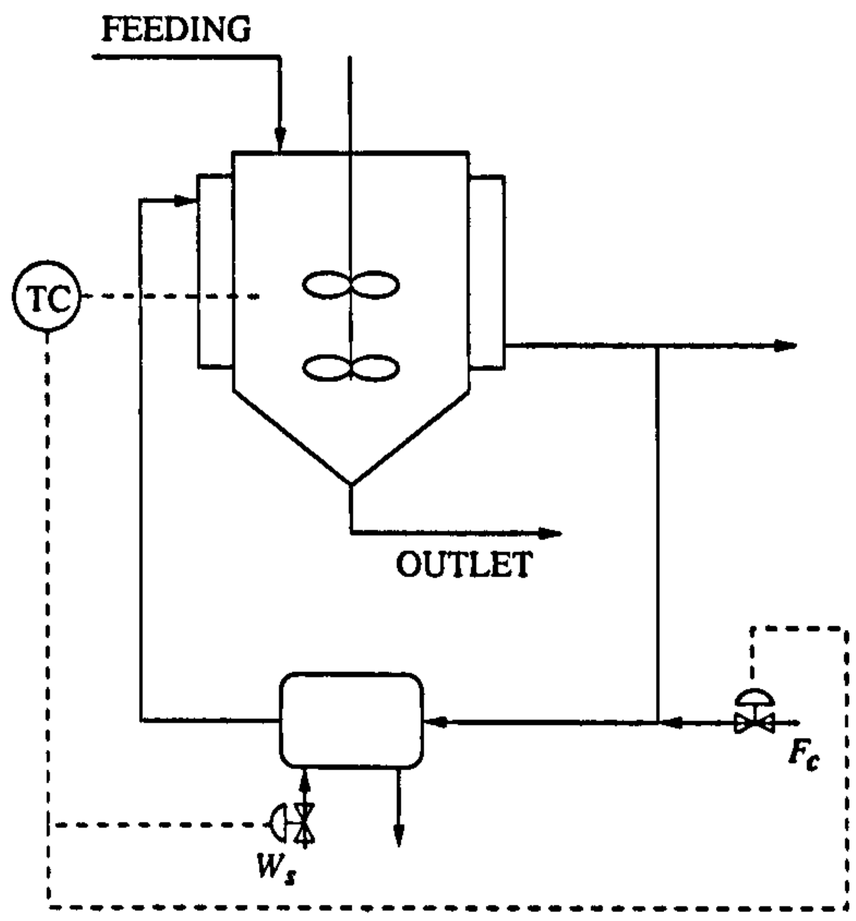


Figure 2.10: Semi-batch reactor.

inlet. The water inlet temperature is controlled by means of a heat exchanger on the water-recycling loop. The manipulated variables are cooling water flowrate F_c and vapour stream W_s in the heat exchanger. The reactor is also equipped with a sparging system able to inject steam directly inside the tank for rapid heating. More detailed descriptions of this unit and its ancillary equipment and operation have been presented by Liu (1995) and Liu and Macchietto (1995). For the purposes of this example, the reactor is always operated in batch mode.

As exemplified in Figure 2.11, the model is first decoupled into equipment and procedures which are completely described in process simulation models only and the models where an integrated use of both types of software is required. From this first division we obtain:

Process Simulation Models (Environment):

- Jacket and cooling/heating system model
- Control system
- Operation procedures

Integrated Models (Internal Models):

- Tank model

In the internal model, both gPROMS and the CFD code are used to model the stirred tank, with the following division of responsibilities (model partitioning):

gPROMS:

- composition balance (dynamic)
- energy balance (dynamic)
- physical property estimation

Fluent:

- momentum balance
- process-side heat transfer coefficient estimation

Here, the gPROMS tank model is based on that proposed by Macquart-Moulin (1998). It assumes (for the time being) uniform concentration and temperature in the tank. Two models of different complexity are used for the jacket: one (*Uniform_Jacket*) assumes a uniform water temperature profile (well-mixed jacket), while the second (*Distributed_Jacket*) accounts for axial (vertical) variations of temperature in the jacket. The heat transfer coefficient on the jacket side is taken to be a constant 1500 W/m^2K.

A 3D model of the tank was developed in Fluent. The model geometry and CFD grid were defined by means of a pre-processor package (MixSim by Fluent, Inc.) that allows an easy meshing of the domain. Two different versions of this model are used, employing single and double mixing impeller centred shafts respectively (each containing over 30,000 computational cells). In both cases, the process-side heat transfer coefficient is computed by post-processing the fluid field results of the CFD computation in a *User Defined Subroutine*[2] making use of correlations presented by Launder and Spalding (1974). Local values of heat transfer coefficient h_i are calculated from the local values of kinetic energy calculated by Fluent for each grid cell and then averaged over the reactor wall heat exchange surface A (assumed constant) to obtain the global heat transfer coefficient h according to the equation:

$$h = \frac{\sum_{i=1}^{no\,wallcells} h_i A_i}{A}, \tag{2.3}$$

where A_i is the surface area available for heat transfer for each wall cell i.

An important parameter in this CFD model is the impeller speed that ultimately determines the intensity of heat transfer between the process fluid and the jacket. Although this parameter is not used directly by the gPROMS-side submodel, it is

[2]Procedure available in Fluent to postprocess CFD results and implement user calculations. Similar post-processing facilities are available in most commercial codes.

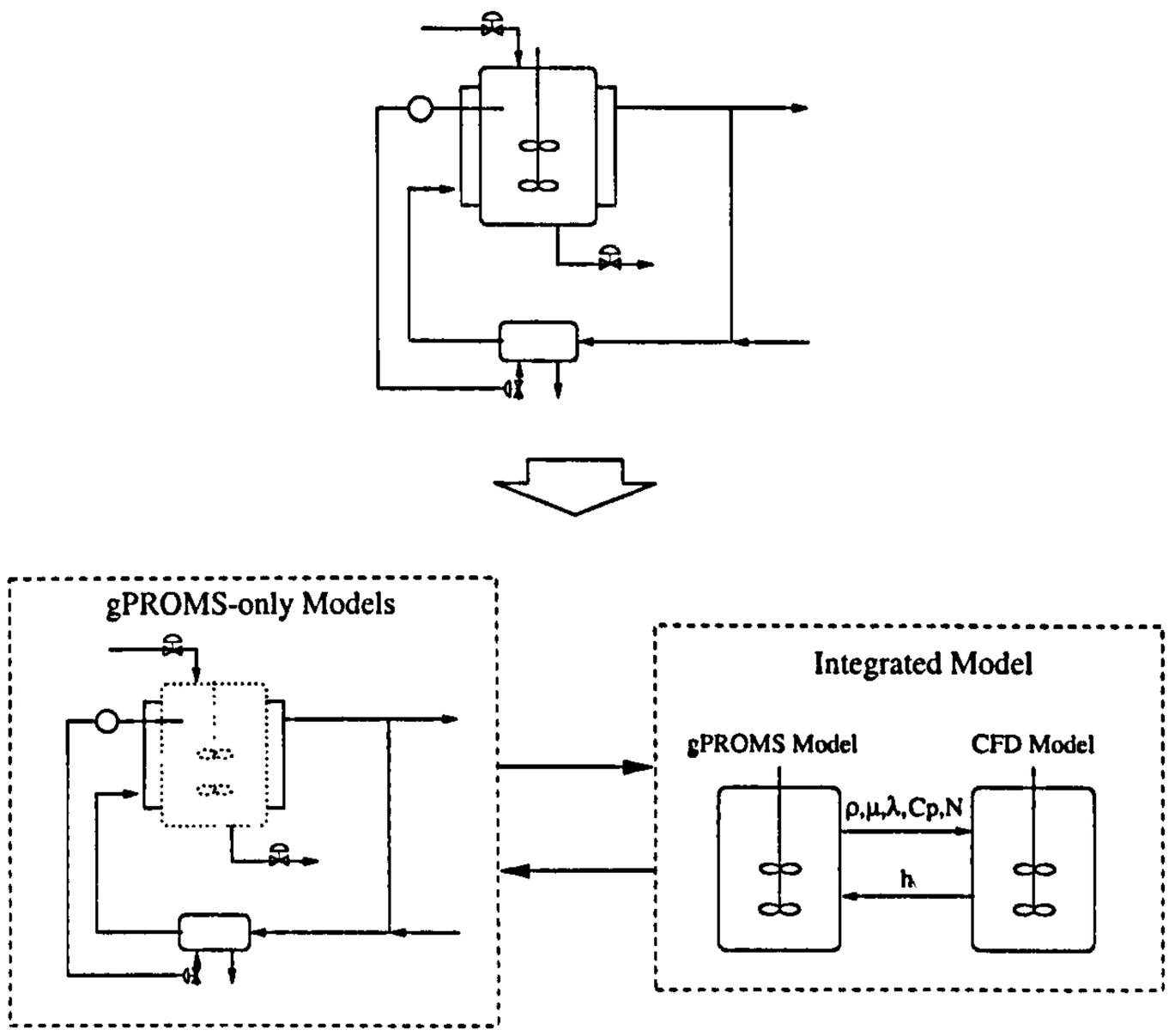


Figure 2.11: CFD-Process Simulation Coupling in the semi-batch reactor.

introduced into it so that it can be manipulated as a design or control variable in the overall simulation. The process simulation package merely passes its value (N), together with physical properties (viscosity μ, density ρ, heat capacity C_p and conductivity λ), to the CFD-side model where it is actually used (Figure 2.11). As stated in the previous sections, gPROMS is used as the master package and handles the overall simulation by issuing calls to the CFD software when heat transfer coefficient updates are needed.

The above combined model has been used for the dynamic simulation of two different systems.

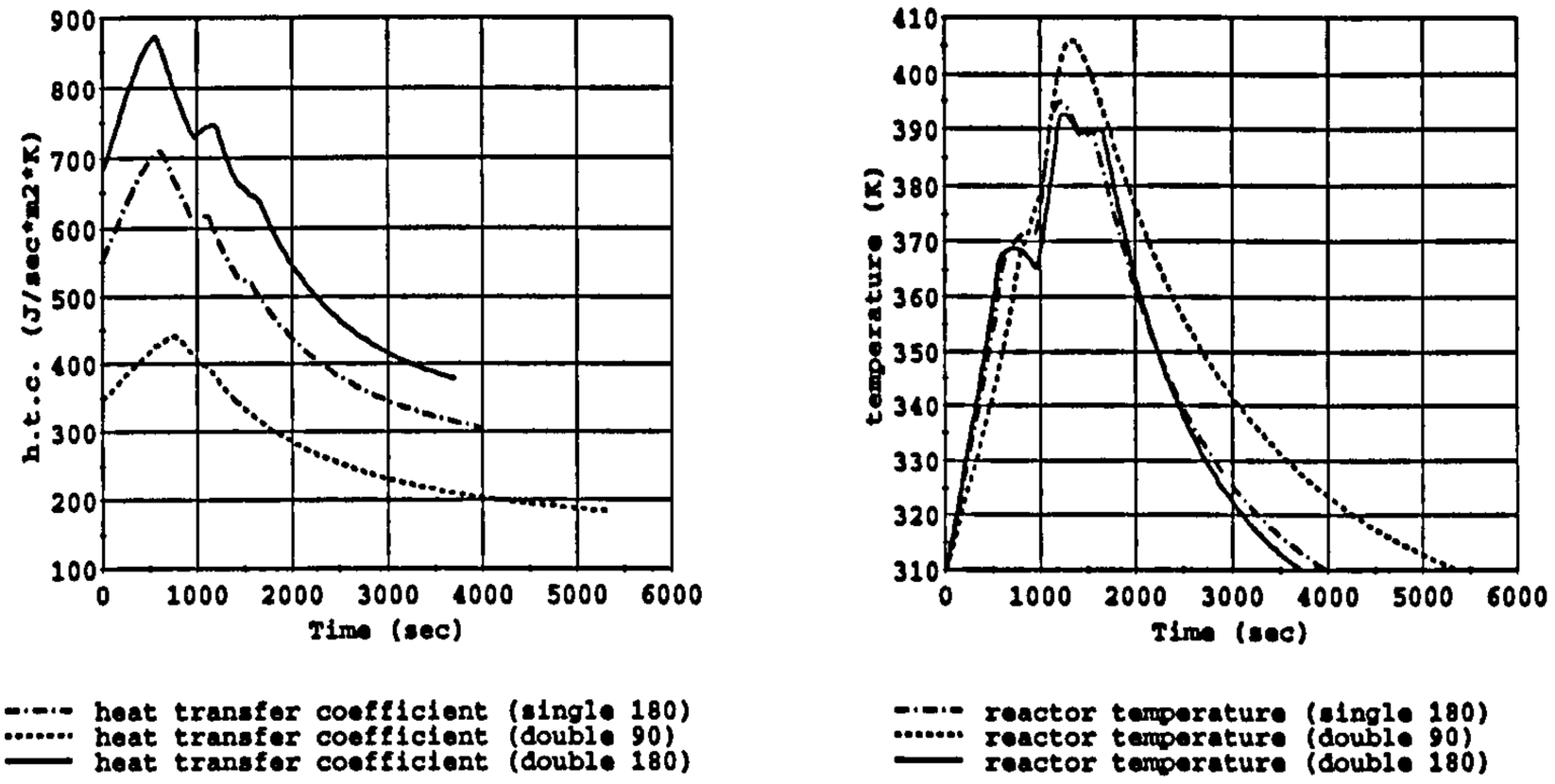
2.4.1 Example 1: Simulation of an Esterification Reaction

This example considers the exothermic ($\Delta H = -60$ kJ/mole) esterification of methanol using acetic anhydride (Macquart-Moulin, 1998):

$$\underset{\text{Methanol}}{CH_3OH} + \underset{\text{Acetic Anhydride}}{(CH_3CO)_2O} \rightarrow \underset{\text{Methyl Acetate}}{CH_3COOCH_3} + \underset{\text{Acetic Acid}}{CH_3COOH}$$

50% of a given initial molar charge in the tank is constituted by the two reactants in equal proportions. The remaining 50% is made up by the products (in equal proportions) used as solvent.

The simulation studies for this example implement the following simple control



(a) (b)

Figure 2.12: Effects of impeller geometry and speed agitation: process-side heat transfer coefficient (a) and reactor temperature (b).

strategy:

1. Start with an initial reactor temperature $T = 310$ K.
2. Heat until the temperature reaches 365 K.
3. Maintain the temperature constant at this level until the methanol molar fraction drops below 0.15.
4. Heat until the temperature reaches 390 K.
5. Maintain the temperature constant at this level until the methanol molar fraction drops below 0.05.
6. Cool the reactor contents down to 310 K.

Temperature control in the reactor is achieved by manipulating the vapour stream W_s (Figure 2.10) using a simple proportional controller. The water flowrate F_c is kept constant.

Simulations have been carried out using the double impeller centred shaft at two different velocities (RPM= 180 and RPM= 90) and also with the single impeller centred shaft (RPM= 180). In all cases, the impeller speed has been kept constant throughout the simulation. Model *Uniform_Jacket* is chosen.

Some of the results obtained are illustrated in Figure 2.12. Figure 2.12a illustrates the sensitivity of the heat transfer coefficient to fluid flow behaviour and process con-

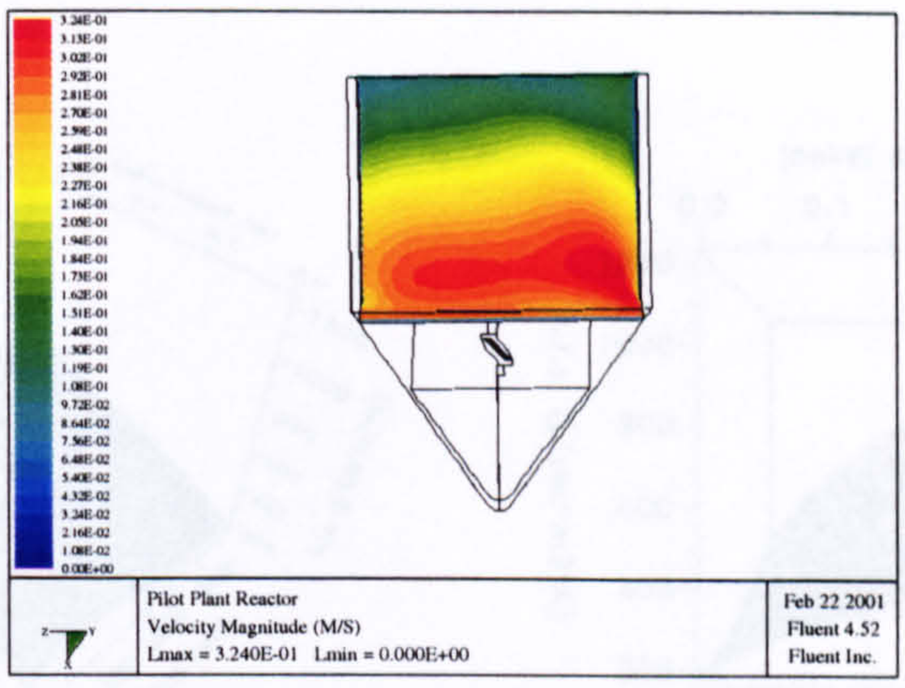


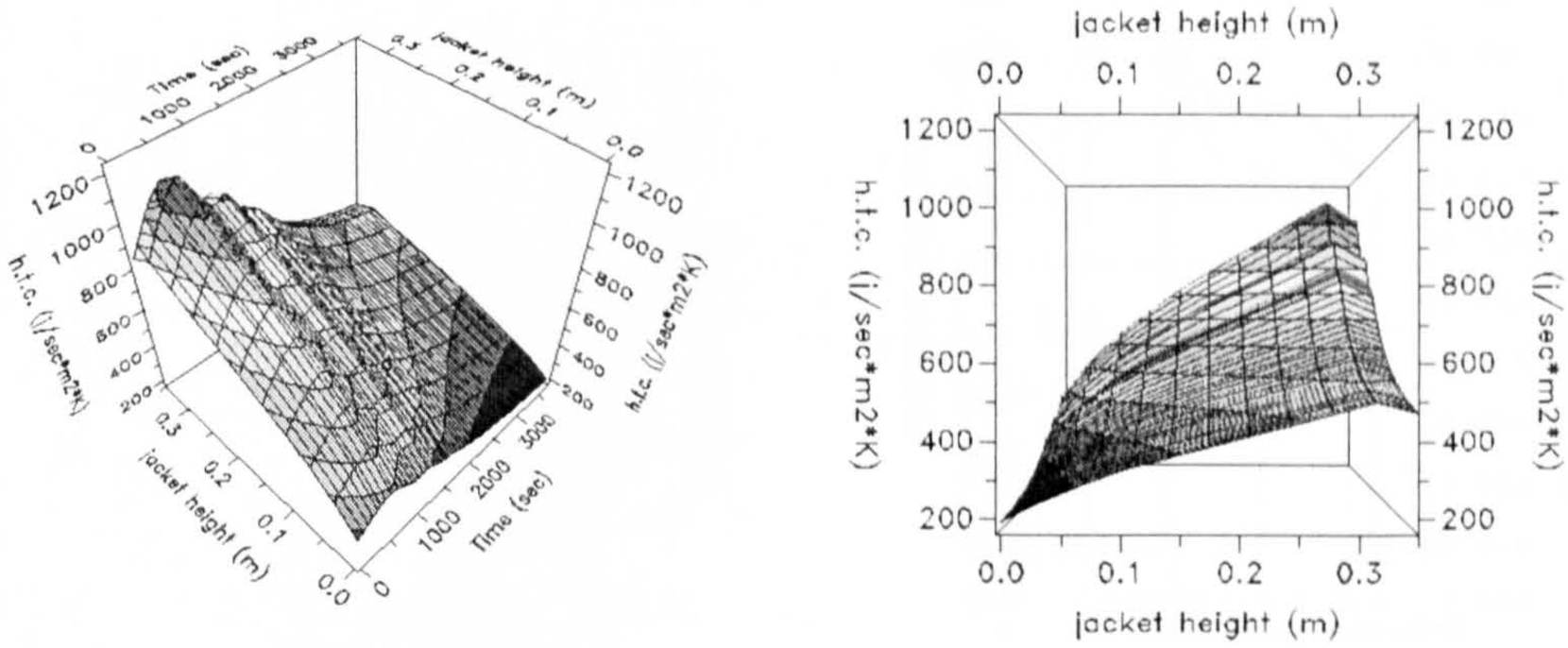
Figure 2.13: Velocity magnitude profile along the tank wall.

ditions (temperature and physical properties) changing during the reaction. As can be seen from Figure 2.12b, the impeller speed can significantly affect the process dynamics and the effectiveness of control strategy. In particular, a low impeller speed (RPM= 90) impairs the capability of the controller to regulate the process behaviour as the heat transfer coefficient is not sufficiently high for ensuring good temperature control from the jacket. There is a significant overshoot in the temperature of about 15 K. However, at higher RPM, the system is able to follow set points quite closely.

The effects of impeller type can also be seen from these results. As expected, a double impeller leads to higher process-side heat transfer coefficients than a single impeller operated at the same speed. However, the effect on the temperature profile is less pronounced, due to the interactions with the control system.

It is also interesting to consider the effect of agitation on batch times. A double impeller operated at the lower speed of RPM= 90 leads to each batch requiring about 40% more time than when it is operated at RPM = 180. Even using a single impeller at the same speed makes the process about 8% slower. The differences between different impeller designs and speeds are even more significant if one considers only the "dead times" involved in the operating procedure described above. These are the times necessary to heat the reactor up from 310 K to 365 K and to cool it down from 390 K to 310 K (steps 2 and 6), during which little reaction takes place. Compared to the best case of a double impeller at RPM= 180, the use of a double impeller at RPM= 90 increases these dead times by about 65%, while the use of a single impeller at RPM= 180 increases them by about 15%. Overall, agitation and impeller design have a major influence on the controllability and productivity of the process.

Let us now test the validity of our hypotheses, in particular the uniform jacket assumption. Let us consider the velocity distribution in the reactor. Figure 2.13 shows

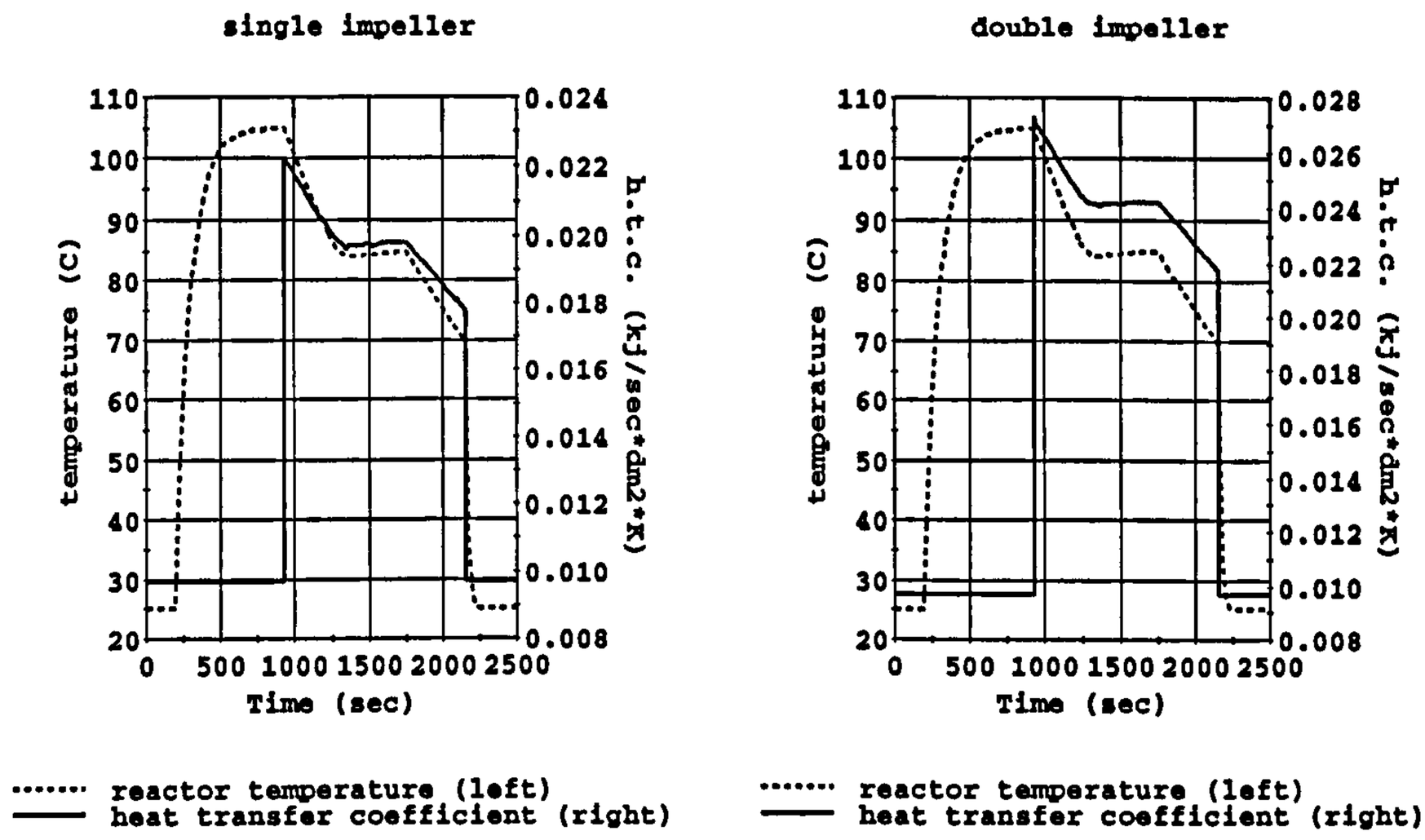


(a) (b)

Figure 2.14: Variation of process-side heat transfer coefficient with time and vertical position. The value 0 on the *jacket height* axis corresponds to the top of the tank. Case (b) emphasises spatial variations.

a typical 2D profile of the magnitude of the velocity vector within the tank (at the wall; 1/4 circumference between two baffles is represented). It can be seen that this varies appreciably vertically along the wall separating the tank from the surrounding jacket. Also, the profile is not uniform around the tank circumference due to the effects of the baffles. This has a corresponding significant effect on the process-side heat transfer coefficient.

In order to investigate the influence that this variation may have on the process performance, we divided the reactor heat exchange surface into 10 adjacent horizontal zones and modified the post-processing of the CFD results to compute a separate (local) heat transfer coefficient within each zone. The gPROMS model then takes account of the variation in both heat transfer coefficient and jacket fluid temperature along the jacket (model *Distributed_Jacket*). The results of the *Distributed_Jacket* model for the double impeller case were found to be very similar in terms of time and temperature changes to those of the *Uniform_Jacket* model. However, it is interesting to observe the variation of the heat transfer coefficient behaviour shown in Figure 2.14. In particular, it appears that throughout the batch significant heat exchange occurs only in a limited part of the heat exchange surface (the bottom). It may well be possible to modify the impeller design position to improve uniformity in heat exchange.



(a) (b)

Figure 2.15: Reactor temperature and process-side heat transfer coefficient for single and double impeller.

2.4.2 Example 2: Simulation of Starch Gelatinisation and Degradation

Our second example utilises the same tank reactor as in Example 1. It uses a more sophisticated control scheme based on the mentioned work by Liu and Macchietto (1995) who proposed a nonlinear model-based control method for the temperature control of the batch reactor. The solution (dilution water and starch slurry) in the tank is heated through direct steam injection to a temperature of 105° C where starch gelatinisation occurs. After cooling down to 85° C, the injection of an appropriate enzyme triggers a degradation reaction that is slightly exothermic. The mixture is then cooled down to 75° C and discharged. Beyond the initial sparge heating phase, temperature is regulated by means of jacket control.

In this case, we employ the *Uniform_Jacket* model. As in the previous example, the heat transfer coefficient on the reactor side is computed by the CFD model. Since specific data for the water/starch mixture are not available and considering the difficulty of handling non-Newtonian and very viscous liquids, water-like properties have been assumed for the process liquid. This is a heavy simplification (see, e.g. chapter 8) and we cannot expect the simulation results to be representative of the actual case. Nonetheless, this example demonstrates the possibility of applying the proposed approach to integrate a CFD mixing calculation within a more complicated process

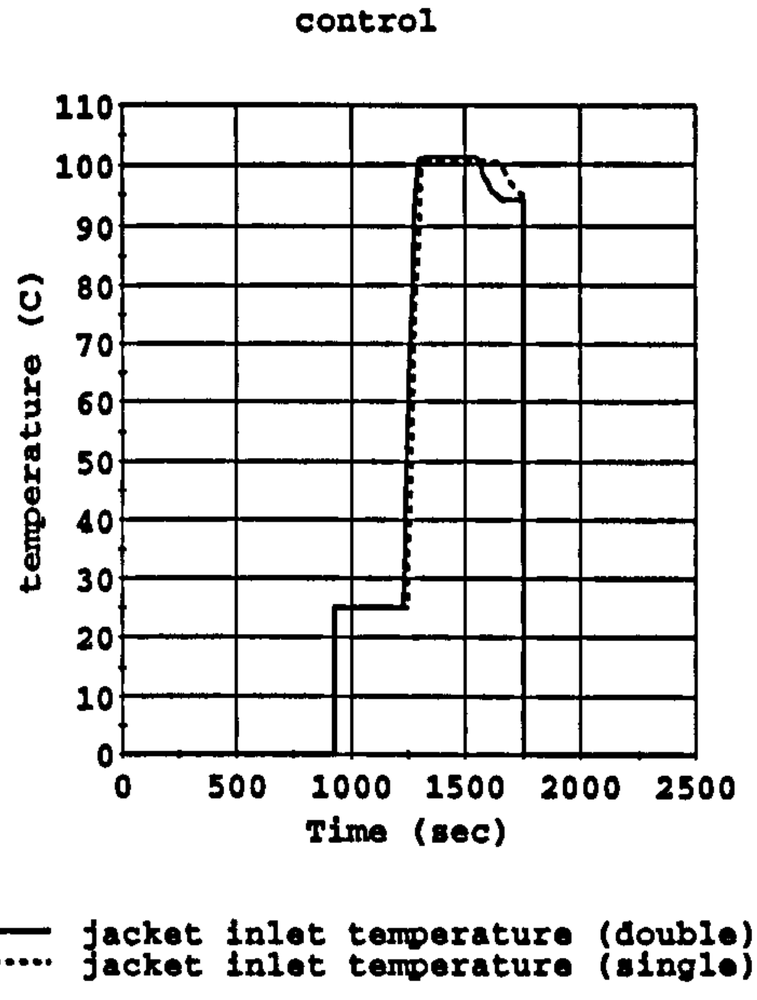


Figure 2.16: The effect on the jacket inlet temperature.

simulation model handling (Liu, 1995):

- an advanced control system model;
- complex operating procedures;
- several process simulation models describing the ancillary equipment (tanks, heat exchangers, pumps, valves, etc.).

Simulations were carried out for both impeller geometries.

A typical design/operation question here is whether a single or double impeller should be used. The former is less expensive but has lower heat exchange performance, as noted in the previous example. A second relevant question is whether the cooling equipment and advanced system control can adequately cope or the more expensive double impeller should be used. The results in Figure 2.15a-b show the effectiveness of the model based controller in compensating for the rather large variations (up to 25%) in heat transfer coefficient when using the two impeller types. Figure 2.16 shows the variation of the temperature at the jacket inlet as a result of control action in the two cases. In conclusion, there is little justification for using the more expensive double impeller as it does not seem to improve the heat transfer appreciably. This example demonstrates the applicability of our technique to a complex dynamic model involving an advanced control strategy.

2.5 Towards a Discrete Model

The previous examples demonstrate some of the potential capabilities and use of a model where CFD and process simulation are integrated to obtain a detailed interaction between fluid mechanics, heat transfer, reaction and control strategy. The CFD package is handled as a hydrodynamic property provider by the process simulation package which controls the overall simulation according to the stated principles of parallel integration. A clear division of responsibilities is achieved by partitioning the overall model between the two simulation tools while a strong link between the different scale models is maintained by exchanging critical parameters. Nonetheless, it is quite clear that the suggested integration still presents some drawbacks. The main ones are:

- the reactor mixing calculations accounting for spatial variation are placed entirely on the CFD model, while the process simulation model assumes that the entire region of interest is well-mixed;

- the CFD is only used to compute parameters which are affected by the fluid flow behaviour; no information concerning the fluid flow itself is passed to the higher-scale model.

The computation described of a local heat transfer coefficient taking into account local variations of velocity represents a first attempt to use CFD to obtain data which would allow specifications at lower-scale level. In the next chapters a more general approach is introduced. The objective will be to define a more complete design capable of exploiting CFD capabilities in order to obtain a more detailed representation of process equipment. In this chapter, however, we introduced some first design concepts which will be maintained throughout this work:

- Parallel integration is the approach followed to combine CFD packages and process modelling tools.

- The process simulation software is used as top level program to handle the flux of information between the two packages and overall convergence.

- There is an important distinction between integrated models and the rest of the model. From now on, we will define the integrated model as *internal model*, while what is outside it will be called *environment*.

2.6 Key Results

The main achievements illustrated in this chapter are:

- The incorporation of CFD and process simulation integration within the more general class of *multiscale modelling.*

- The definition of an approach to achieve *parallel* integration between CFD packages and process modelling tools. The control of the simulation is held by the process simulation package, while the CFD tool becomes a provider of fluid dynamical services. Both the CFD and process simulation tools model the same spatial region using different approximations.

- The definition of a class of practical problems which the suggested integration approach addresses. The fundamental characteristic of these problems is that it is possible to decouple the model equations (*weakly-coupled* systems), i.e. phenomena at different scales are solved by independent models exchanging critical parameters. One of the basic hypotheses underlying this approach is that the phenomena described by the CFD model are assumed to be much faster than those defined in the process simulation model. As a result, CFD calculations are performed as steady-state cases.

- The design of a *case study* to illustrate feasibility, flexibility and benefits of the designed architecture. A clear division of modelling responsabilities is achieved within the CFD and process simulation tools while a simultaneous simulation is carried out by exchanging critical parameters.

Chapter 3

The Zone Interface Design

In the previous chapter the concepts of multiscale modelling and parallel integration were discussed. The applicability of the approach was demonstrated by means of an example.

In this chapter those ideas will be further developed to achieve a tighter and more formal integration at a local level for the purpose of establishing a relation between a process simulation *grid* and the CFD mesh.

The ideas described in this chapter are intended to form the basis for a library of models, each aimed at a specific type of processing equipment (e.g. reactors, crystallisers etc.) of general geometry and under general conditions of mixing. Each member in this family will:

- operate with a fixed process simulation input file;
- have responsibility for soliciting the necessary information from the user and making it available to gPROMS via a foreign object.

That will allow a user-friendly definition and coupling of process simulation - CFD models as well as the possibility of easily changing the model equations and parameters.

3.1 The Definition of *Zones*

The scope of this new method is the design of a general interface capable of exchanging information between a process simulation model and a CFD software taking into account local mixing and fluid dynamic conditions. We will use a subdivision of the domain in the process simulation model to describe the fluid flow behaviour, according to the idea developed in previous work (e.g Vivaldo-Lima *et al.*, 1998, Urban and Liberis, 1999, Mann and Knysh, 1984). The domain subdivision produces a network of *zones*

representing physical regions in the process equipment. Each single zone is considered *well-mixed* and homogeneous. The criteria used to define such zones may be based on the equipment geometry or on the distribution of one or more fluid properties.

We thus consider systems which can be modelled as networks of well-mixed zones. These include:

- homogeneous or pseudo-homogeneous reactors/bioreactors
- mixers
- crystallisers.

The transient behaviour of each perfectly-mixed zone is modelled in gPROMS, and so is the behaviour of the entire network.

The function of the CFD package is to compute the total mass flowrates between the zones, the pressure in each zone and properties based on fluid flow behaviour. This is achieved by solving:

- the total mass balance
- the momentum balance for the velocity and pressure for given fluid physical properties.

This computation is carried out at steady-state, thereby assuming that the fluid dynamics operate on a much faster time-scale than the rest of the phenomena of interest.

3.2 The Simulation Model Domain

For each internal model (see chapter 2), we consider a physical system occupying a given spatial domain and surrounded by an environment with which it exchanges mass and energy (see Figure 3.1). The model of the physical system is partitioned into process simulation and CFD submodels.

The spatial domain of interest is divided into a number of *internal zones.* An example of such a division for a 2-dimensional domain is shown in Figure 3.2. In this case, we have 6 internal zones, labelled with capital Roman letters $A,..,F$. The process simulation submodel for each internal zone is assumed to be well-mixed and homogeneous.

The environment interacting with the system of interest is divided into a number of *environment zones.* An example of such a division is shown in Figure 3.3. In this case, we have two environment zones, labelled with lower case Roman letters a and b. Unlike

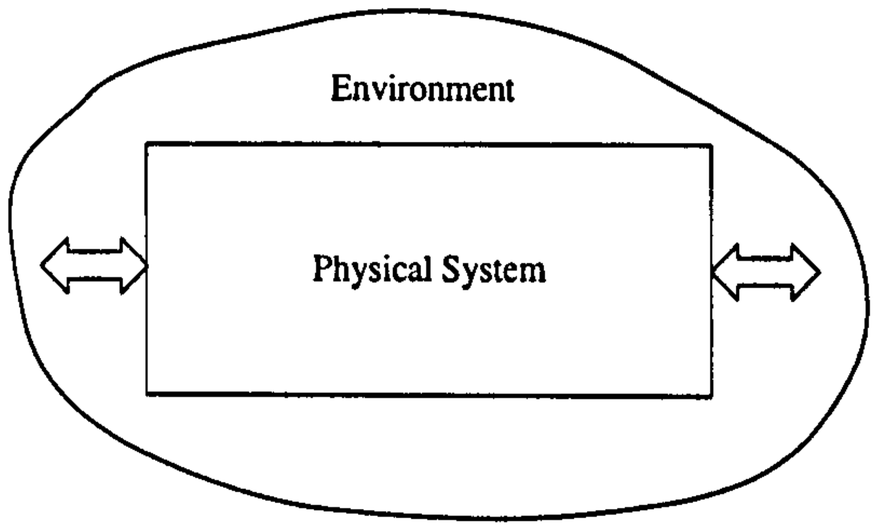


Figure 3.1: Physical System and its Environment.

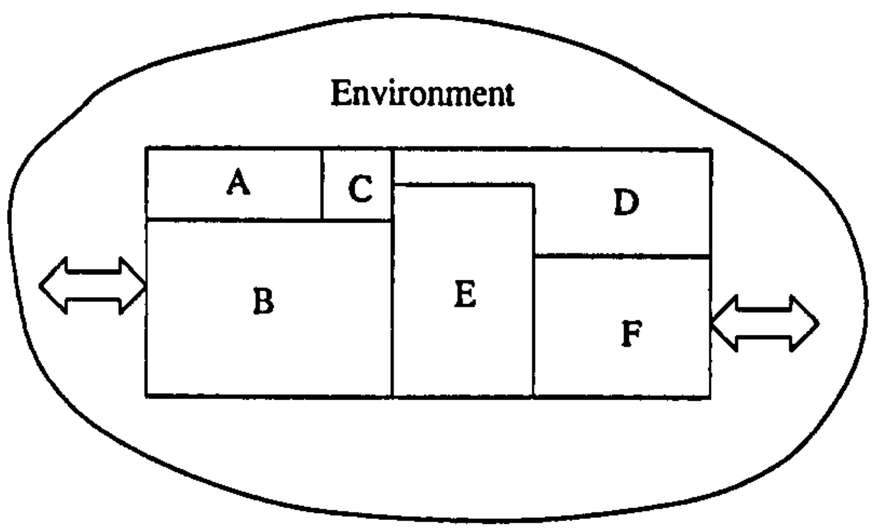


Figure 3.2: Division of domain into Internal Zones.

internal zones an environment zone is not necessarily a well-mixed and homogeneous region. It indicates a gate by which information is passed to the internal model. The scope of environment zones is to link the internal model to the environment.

An internal zone may exchange material and/or energy with one or more

a) internal zones;

b) environment zones.

An environment zone may exchange material and/or energy with one or more internal zones and contains the problem boundary conditions. Internal zones may interact with neighbouring internal zones since they are physically in contact. Also, they may communicate to environment zones since internal zones may be physically connected to the

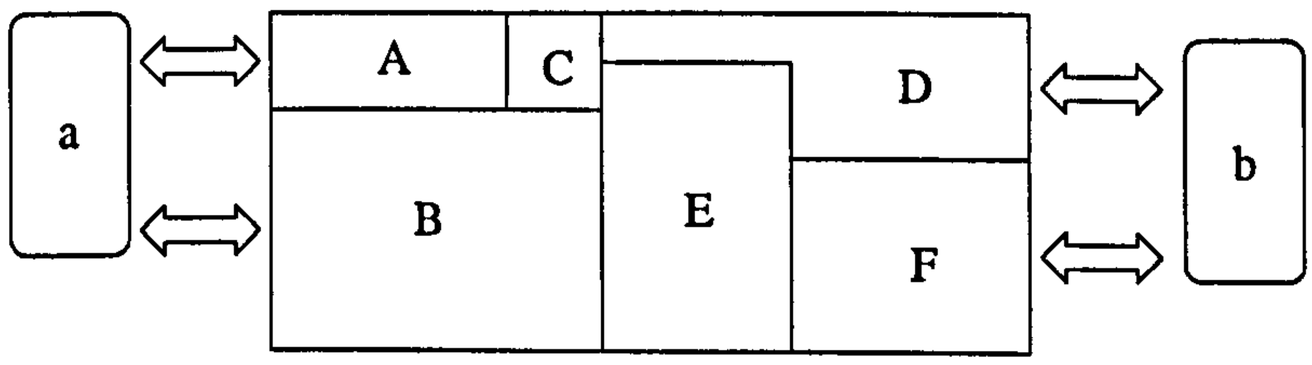


Figure 3.3: Division of environment into Environment Zones.

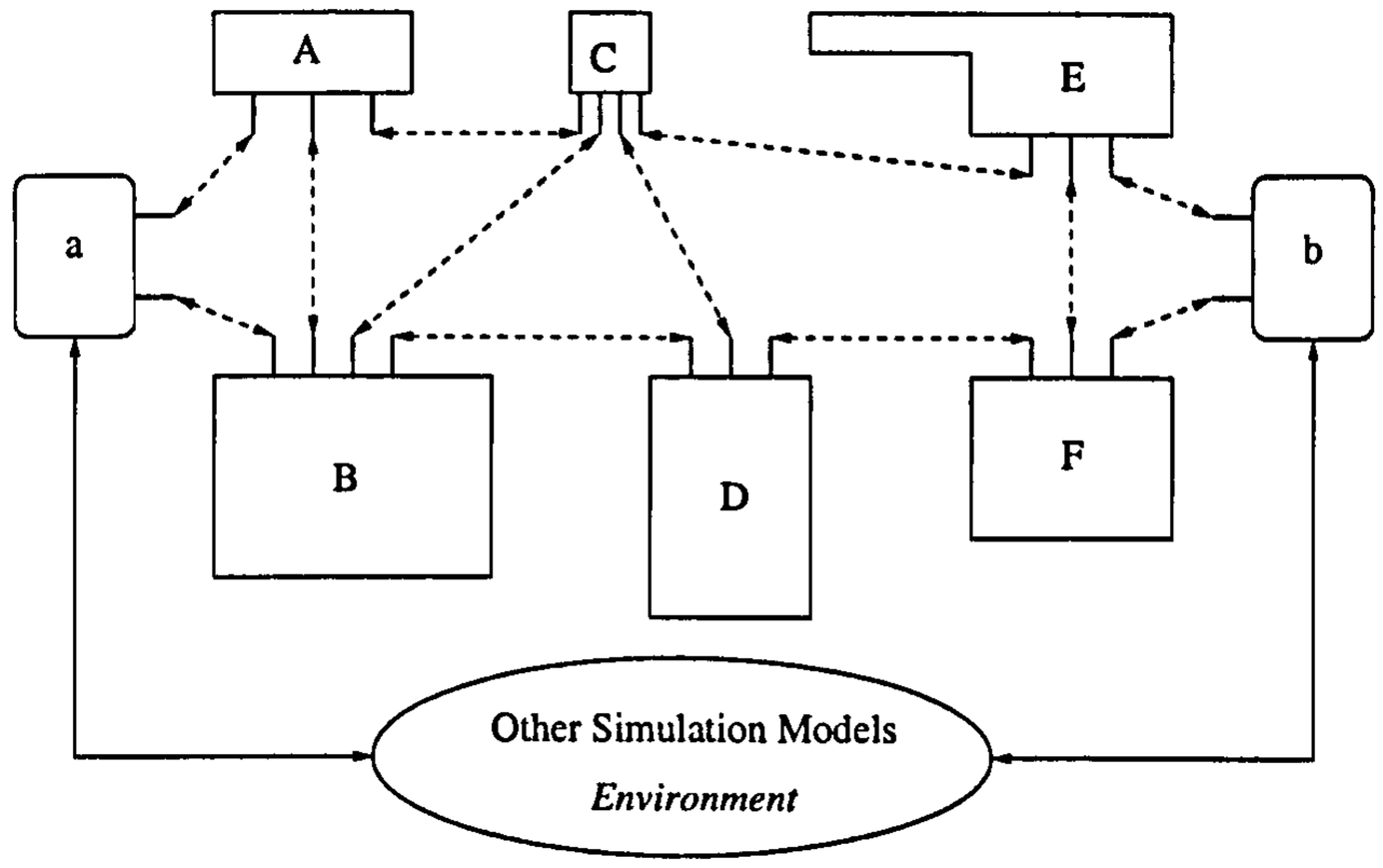


Figure 3.4: Zone Network.

environment. However, environment zones are gates: they do not need to communicate with each other. Nonetheless, they are modelled to allow an interaction to other process simulation models, i.e. to the environment as defined in the previous chapter.

Each such interaction takes place via a different *interface* that connects a *port* of one zone with the port of another zone. In general, the exchange of material and/or energy across each interface is assumed to be bi-directional. Each zone (internal or environment) has *at least* one port. However, the number of ports may differ from one zone to another in the same domain.

The internal and environment zones and the interfaces connecting them form a *zone network*. An example of such a network is shown in Figure 3.4 where the domain has been disassembled into its internal zones. The ports on each zone are now shown explicitly, and the interfaces appear as bi-directional arrows connecting two ports in different zones. Both directions of each interface are characterised by the following information:

a) a mass flowrate;

b) a vector of intensive properties of the material, such as:

- composition
- temperature
- solid fraction
- particle size distribution
- molecular weight distribution;

c) one or more flux quantities (e.g. energy, electric charge) that do *not* depend on the mass flux (i.e. they are not proportional to the mass flowrate).

According to the above definitions, interactions among zones are handled by interfaces which exchange fluxes and intensive properties. Intensive properties are conveyed by means of the mass fluxes, i.e. the flux F_P through an interface of intensive property P is obtained as the product $F \cdot P$ between the property and the mass flowrate F. Other flux quantities (e.g. the heat flux through a conductive wall) are expressed by ad hoc fluxes.

3.3 The CFD Computation Domain

The CFD solution domain is subdivided into a finite number of small control volumes (*cells*) collectively referred to as mesh or grid. From the CFD point of view, an internal zone is a collection of neighbouring cells, while a zone interface is a boundary between adjacent zones or a zone and the environment (e.g. an external equipment boundary or a flow inlet/outlet). The zonal surfaces will be defined along the common cell surface boundaries of the bordering cells between adjacent regions.

The mass flowrate in each direction of each interface is to be determined by the CFD computation. On the other hand, intensive properties of the material and flux quantities not depending on the mass flux are known only by the process simulation package and, therefore, have to be determined by it. The CFD computation requires fluid property information such as:

- the fluid viscosity;
- the fluid density or the parameters characterising the relation between density and pressure (i.e. the equation of state);

over the domain of interest. The elements of the fluid property information vector may vary depending on the type of CFD computation required. For instance, in an incompressible liquid phase computation, the vector may include the density of the liquid; on the other hand, for a compressible (gas-phase) computation, the vector may include the compressibility factor for the fluid. However, in almost all cases, these quantities will depend on intensive properties (e.g. composition and temperature) that are not known to the CFD package. Consequently, they have to be determined by gPROMS and supplied to the CFD package.

In addition to the information mentioned above, the CFD computation may require additional information such as:

- boundary conditions, e.g.
  - the inlet mass or volumetric flowrate, or the velocity;
  - upstream or downstream pressures;
- operating parameters, e.g.
  - the agitation speed of mixing devices;
- numerical solution parameters, e.g.
  - the convergence tolerance of the CFD computation.

Often, these parameters will be of no interest to the process simulation model and can be hidden from it. This is the case, for example, when the values of the parameters are to be kept constant throughout a given computation. In other cases, however, the values of these parameters may change during the computation. For example, this may happen:

- during an interactive simulation in which the user performs certain manual manipulations;
- if automatic control action is applied to these parameters to achieve a certain control objective;
- during an optimisation of process performance which relies on manipulation of these parameters.

In all such cases and in order to ensure that the CFD computation remains a well-defined mathematical function as far as gPROMS is concerned, these parameters must be made known to gPROMS even if it does not actually know their physical interpretation. We will refer to these quantities as `CFDParameter`.

3.4 Zone Modelling in Process Simulation

This section is dedicated to the design of internal and environment zone models. This will be obtained by using the gPROMS language to describe the characteristics that each zone has to fulfill to demonstrate the required features within a zone network model.

The following minimal requirements are imposed on the *internal zone model* in gPROMS:

Given:

- all information (mass flowrate and fluid properties) characterising the inlet direction of all ports;

- the mass flowrates characterising the outlet direction of all ports;

Determine:

- information characterising the outlet direction of all ports;

- the fluid property information in the zone.

The *environment zone model* in gPROMS must determine:

- all information characterising the outlet direction of all ports;

- the fluid property information.

To implement the above ideas in gPROMS, we have to define several entities in order to build a general structure within which to implement the specific models.

The gPROMS language allows the definition of `STREAM` variables (gPROMS Introductory User's Guide, 1997). Streams are simply subsets of variables in a model. They are used to interface different submodels within a higher-level model. We define an `Interface` stream type that comprises all information listed at the end of section 3.3, but not the mass flowrate.

Two separate models, namely `InternalZone` and `EnvironmentZone` are introduced. These two models represent any internal and environment zone in the domain. They are modules whose structures are independent of the number, size and location of zones. This general architecture is obtained by defining a number of parameters and variables which do not depend on the process which is being modelled (Figure 3.5):

1. The number of ports in each of the above models is described by a parameter `NoPort`. This parameter establishes the number of neighbouring internal and environment zones.

2. The inlet and outlet direction of each port are modelled as separate `STREAM`s in the models, called `Inlet` and `Outlet` respectively.
   `STREAM Inlet` is declared as `ARRAY(NoPort) OF Interface`. `STREAM Outlet` is declared as a simple `Interface`. According to the definition of streams in the gPROMS language, `Inlet` and `Outlet` are vehicles conveying information which

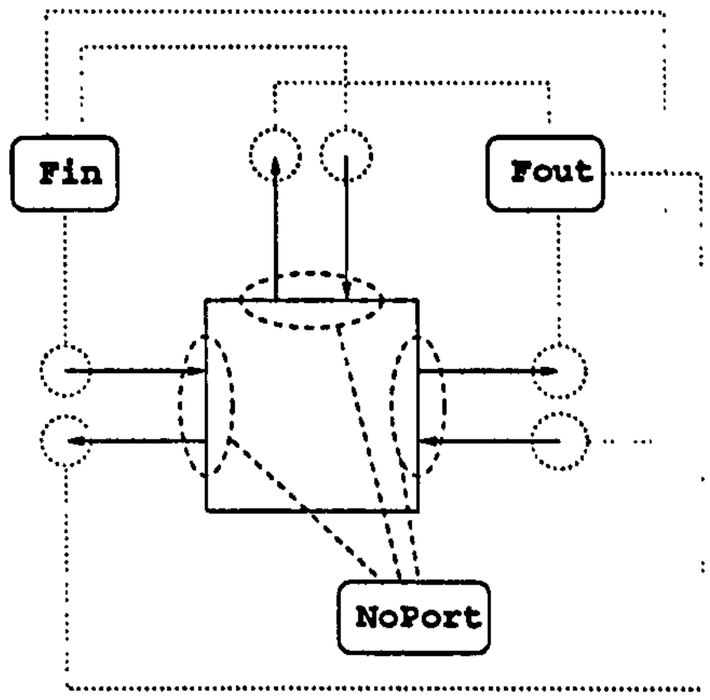


Figure 3.5: Zone parameters

has to be exchanged among zones. `STREAM Outlet` is not declared as an array because all outlet streams contain the same intensive properties since the zone is assumed to be perfectly-mixed.

3. Each model contains variables `Fin` and `Fout`, declared as arrays of length `NoPort` of mass flowrates, and representing the mass flowrates in the inlet and outlet ports respectively.

4. Each model contains variable `P`, declared as a pressure, and representing the pressure in the zone.

The two zone models must be designed to handle and communicate information regarding the process fluid. Thus, they contain:

1. A parameter `NoFluidProperty` representing the number of properties that characterise the fluid for the purposes of the CFD computation.

2. A vector of variables of length `NoFluidProperty` called `FluidProperty` that is used to hold the fluid property information. For instance, fluid viscosity (or parameters for non-Newtonian laws) and density are contained in the `FluidProperty` vector.

3. A set of equations that defines the values of the variables in vector `FluidProperty`, usually as functions of the values of the intensive properties within the zone.

In addition to the structural and fluid property information, model `InternalZone` must consider a real parameter `V` corresponding to the volume of the zone. The volume is set as a parameter since *we assume it to be constant.* This main assumption is a direct

consequence of the fixed volume domain in the CFD simulation. Although there exist CFD codes capable of dealing with variable volumes (e.g. Tome *et al.*, 1999), they are very tailored to restricted uses. For this reason, we decided to stick to available capabilities of commercial process simulation codes and implement the interface upon the constant volume hypothesis.

Typically, the zone models will also contain:

- other variables that describe the phenomena taking place within the system;
- equations that characterise the transient behaviour, fulfilling the requirements set at the beginning of this section.

In addition to containing the structural and fluid property information, `MODEL EnvironmentZone` must fulfil the requirements needed to characterise the outlet direction of all ports.

Tables 3.1 and 3.2 contain the outlines of the gPROMS implementation of the generic models `InternalZone` and `EnvironmentZone`. The gPROMS language is used (symbols # and { } designate comments).

Table 3.1: The `InternalZone` model.

```
MODEL InternalZone

PARAMETER
 # Numbers of Ports and Fluid Properties
 NoPort AS INTEGER
 NoFluidProperty AS INTEGER

 # Volume of zone
 V AS REAL

 # Other parameters
 .  .  .

VARIABLE
 # Mass flowrates
 Fin, Fout AS ARRAY(NoPort) OF MassFlowrate

 # Pressure
 P AS Pressure

 # Fluid property
 FluidProperty AS ARRAY(NoFluidProperty) OF NoType

 # Intensive property variables and/or fluxes in inlet and
 # outlet streams
 .  .  .
```

```
 # Other variables
 .  .  .

STREAM
 Inlet :  { ...  intensive properties and/or fluxes}
                      AS ARRAY(NoPort) OF Interface
 Outlet :  { ...  intensive properties and/or fluxes} AS Interface

EQUATION
 # Definition of fluid properties
 FluidProperty(1) = .....  ;
 .  .  .
 FluidProperty(NoFluidProperty) = .....  ;

 # Other equations
 .  .  .

END # model InternalZone
```

Table 3.2: The **EnvironmentZone** model.

```
MODEL EnvironmentZone

PARAMETER
 # Numbers of Ports and Fluid Properties
 NoPort AS INTEGER
 NoFluidProperty AS INTEGER

 # Other parameters
 .  .  .

VARIABLE
 # Mass flowrates
 Fin, Fout AS ARRAY(NoPort) OF MassFlowrate

 # Pressure
 P AS Pressure

 # Fluid property
 FluidProperty AS ARRAY(NoFluidProperty) OF NoType

 # Intensive property variables and/or fluxes in inlet and
 # outlet streams
 .  .  .

 # Other variables
 .  .  .

STREAM
 Inlet :  { ...  intensive properties and/or fluxes}
                      AS ARRAY(NoPort) OF Interface
 Outlet :  { ...  intensive properties and/or fluxes} AS Interface
```

```
EQUATION
 # Definition of fluid properties
 FluidProperty(1) = .....  ;
 .  .  .
 FluidProperty(NoFluidProperty) = .....  ;

 # Other equations
 .  .  .

END # model EnvironmentZone
```

Note that these templates show the minimal requirements for the models. They will need to be customised to deal with specific applications (e.g. reactors, crystallisers etc.). An example will be considered at the end of this chapter.

3.5 Zone Network Modelling

3.5.1 Structural Issues

The properties of the zone models make it possible to define a completely generic model of a *zone network* in gPROMS which is *independent* of the specific process equations contained in each zone model. This section and sections 3.5.2 and 3.5.3 will describe the characteristics of the zone network model as designed in the gPROMS language.

We define a model `ZoneNetwork` comprising:

1. a number `NoIZone` of instances of `InternalZone`;

2. a number `NoEZone` of instances of `EnvironmentZone`.

The previous parameters represent the number of internal and environment zones in the network. They do not change and are fixed at the beginning of the simulation. The `ZoneNetwork` model also comprises a number of parameters to specify interfaces and connections among zones:

1. a number `NoIIInterface` of interfaces, each connecting a port of an internal zone with a port of a different internal zone;

2. a number `NoIEInterface` of interfaces, each connecting a port of an internal zone with a port of an environment zone.

As well as `NoIZone` and `NoEZone`, parameters `NoIIInterface` and `NoIEInterface` are set at the beginning of the simulation. We define the two zones at either side of each interface as *left* and *right* zones, respectively. In the case of an interface between two

internal zones, the characterisations *left* and *right* have no geometrical connotation whatsoever and they just signify the two ends of an interface and distinguish them from each other. For an interface between an environment zone and an internal zone, the characterisations *left* and *right* denote the internal zone and the environment zone respectively.

For each interface between internal zones we define:

- its `IILeftZone`, an index in the range [1,...,`NoIZone`] denoting the internal zone at one end of the interface;
- its `IIRightZone`, an index in the range [1,...,`NoIZone`] denoting the internal zone at the other end of the interface;
- its `IILeftPort`, an index in the range [1,...,`IZone(IILeftZone).NoPort`] denoting the port of internal zone `IILeftZone` that is connected to the interface;
- its `IIRightPort`, an index in the range [1,...,`IZone(IIRightZone).NoPort`] denoting the port of internal zone `IIRightZone` that is connected to the interface.

For each interface between an environment and an internal zone (or environment interface) we set:

- its `IELeftZone`, an index in the range [1,...,`NoEZone`] denoting the internal zone at one end of the interface;
- its `IERightZone`, an index in the range [1,...,`NoIZone`] denoting the environment zone at the other end of the interface;
- its `IELeftPort`, an index in the range [1,...,`IZone(IELeftZone).NoPort`] denoting the port of internal zone `IELeftZone` that is connected to the interface;
- its `IERightPort`, an index in the range [1,...,`EZone(IERightZone).NoPort`] denoting the port of environment zone `IERightZone` that is connected to the interface.

Figure 3.6 illustrates the above nomenclature. In the Figure there are two internal interfaces (`NoIIInterface=2`), 1 and 2 respectively, and two environment interfaces (`NoIEInterface=2`), *I* and *II* respectively.

Let us suppose that internal zone *I1* is a left zone and internal zone *I2* is a right zone with respect to interface 1, while internal zone *I2* is a left zone and internal zone *I3* is a right zone with respect to interface 2. According to the convention between internal and environment zones, we obtain that internal zones *I1* and *I3* are left zones

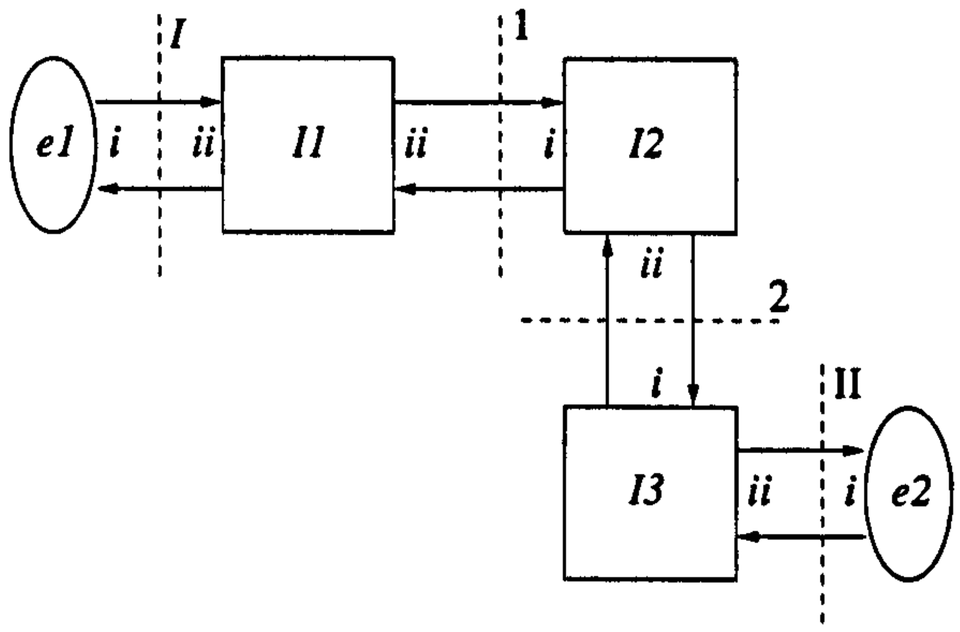


Figure 3.6: A network example.

and environment zones *e1* and *e2* are right zones with respect to interfaces *I* and *II*.

Thus, if we consider all the interfaces, we obtain:

- parameter `IILeftZone(1)=1`, i.e. the left zone with respect to the first internal interface is the first internal zone *I1*;
- parameter `IIRightZone(1)=2`, i.e. the right zone with respect to the first internal interface is the second internal zone *I2*;
- parameter `IILeftZone(2)=2`, i.e. the left zone with respect to the second internal interface is the second internal zone *I2*;
- parameter `IIRightZone(2)=3`, i.e. the right zone with respect to the second internal interface is the third internal zone *I3*;
- parameter `IELeftZone(1)=1`, i.e. the left zone with respect to the first environment interface is the first internal zone *I1*;
- parameter `IERightZone(1)=1`, i.e. the right zone with respect to the first environment interface is the first environment zone *e1*;
- parameter `IELeftZone(2)=3`, i.e. the left zone with respect to the second environment interface is the third internal zone *I3*;
- parameter `IERightZone(2)=2`, i.e. the right zone with respect to the second environment interface is the second environment zone *e2*.

Ports in each zone are numbered as $i, ii, \ldots$ to the maximum value `NoPort` defined within each zone model. With regard to the ports, the parameters setting the port number of the left and right zones facing each interface assume the following values:

- parameter `IILeftPort(1)=2`, i.e. the first internal interface is faced by the second port of its left zone;
- parameter `IIRightPort(1)=1`, i.e. the first internal interface is faced by the first port of its right zone;
- parameter `IILeftPort(2)=2`, i.e. the second internal interface is faced by the second port of its left zone;
- parameter `IIRightPort(2)=1`, i.e. the second internal interface is faced by the first port of its right zone;
- parameter `IELeftPort(1)=1`, i.e. the first environment interface is faced by the first port of its left zone;
- parameter `IERightPort(1)=1`, i.e. the first environment interface is faced by the first port of its right zone;
- parameter `IELeftPort(2)=2`, i.e. the second environment interface is faced by the second port of its left zone;
- parameter `IERightPort(2)=1`, i.e. the second environment interface is faced by the first port of its right zone.

The `ZoneNetwork` model includes a parameter `NoFluidProperty` which has the same meaning as the corresponding parameters in the individual zone models. It is assumed that all zones in a particular problem will be characterised by the same set (and, therefore, number and order sequence) of fluid property parameters. Consequently, we rely on standard parameter propagation mechanisms in gPROMS to fix the values of `NoFluidProperty` in the individual zones simply by setting it in the `ZoneNetwork` model.

`ZoneNetwork` includes a parameter `NoCFDParameter` which is used to hold the length of the `CFDparameter` vector and a parameter `NoCFDMethods` which is used to set the length of the `CFDMethods` (both `CFDparameter` and `CFDMethods` will be discussed in § 3.5.2). The `ZoneNetwork` model includes a parameter (`ZoneNet`) identifying the foreign object `gCFD` which is responsible to set all the parameters values according to the specific zone network: the object provides methods that return a variety of constant quantities, including:

- the values of the parameters defining the structure of the zone network;
- the number of ports in each zone;

- the volume of each internal zone;

- the length of the fluid property vector;

- the length of the CFD parameter vector.

Thus, those values are not directly set within the gPROMS model, but are retrieved from the external object according to a procedure that will be discussed in chapter 5.

Table 3.3 contains a set of gPROMS declarations implementing the ideas described above.

Table 3.3: Parameters in the **ZoneNetwork** model.

```
MODEL ZoneNetwork

PARAMETER
 ZoneNet AS FOREIGN_OBJECT "gCFD"

 # Other Foreign Objects
 .  .  .

 # Numbers of Internal and Environment Zones
 NoIZone AS INTEGER
 NoEZone AS INTEGER

 # Internal-Internal Interface
 NoIIInterface AS INTEGER
 IILeftZone AS ARRAY(NoIIInterface) OF INTEGER
 IIRightZone AS ARRAY(NoIIInterface) OF INTEGER
 IILeftPort AS ARRAY(NoIIInterface) OF INTEGER
 IIRightPort AS ARRAY(NoIIInterface) OF INTEGER

 # Internal-Environment Interfaces
 NoIEInterface AS INTEGER
 IELeftZone AS ARRAY(NoIEInterface) OF INTEGER
 IERightZone AS ARRAY(NoIEInterface) OF INTEGER
 IELeftPort AS ARRAY(NoIEInterface) OF INTEGER
 IERightPort AS ARRAY(NoIEInterface) OF INTEGER

 NoFluidProperty AS INTEGER
 NoCFDParameter AS INTEGER
 NoCFDMethods AS INTEGER

 # Dummy parameters
 IVol AS ARRAY(NoIZone) OF REAL
 NIPort AS ARRAY(NoIZone) OF INTEGER
 NEPort AS ARRAY(NoEZone) OF INTEGER

UNIT
 # Internal and Environment Zones
 IZone AS ARRAY (NoIZone) OF InternalZone
 EZone AS ARRAY (NoEZone) OF EnvironmentZone
```

```
SET
 NoIZone := ZoneNet.NumberOfInternalZones() ;
 NoEZone := ZoneNet.NumberOfEnvironmentZones() ;

 NoIIInterface := ZoneNet.NumberOfInternalInternalInterfaces() ;
 IILeftZone := ZoneNet.InternalInternalLeftZones() ;
 IIRightZone := ZoneNet.InternalInternalRightZones() ;
 IILeftPort := ZoneNet.InternalInternalLeftPorts() ;
 IIRightPort := ZoneNet.InternalInternalRightPorts() ;

NoIEInterface := ZoneNet.NumberOfInternalEnvironmentInterfaces () ;
IELeftZone := ZoneNet.InternalEnvironmentLeftZones() ;
IERightZone := ZoneNet.InternalEnvironmentRightZones() ;
IELeftPort := ZoneNet.InternalEnvironmentLeftPorts() ;
IERightPort := ZoneNet.InternalEnvironmentRightPorts() ;

IVol := ZoneNet.InternalZoneVolume() ;
NIPort := ZoneNet.InternalZoneNumberOfPorts() ;
NEPort := ZoneNet.EnvironmentZoneNumberOfPorts() ;

FOR i := 1 TO NoIZone DO
 IZone(i).V := IVol(i) ;
 IZone(i).NoPort := NIPort(i) ;
END

FOR i := 1 TO NoIZone DO
 EZone(i).NoPort := NEPort(i) ;
END

NoFluidProperty := ZoneNet.NumberOfFluidProperties() ;
NoCFDParameter := ZoneNet.NumberOfCFDParameters() ;
NoCFDMethods := ZoneNet.NumberOfCFDMethods() ;
```

3.5.2 Computational Aspects

The interaction between gPROMS and the main CFD computation is performed within the `ZoneNetwork` model via some methods provided by the `ZoneNet` foreign object. These methods:

1. Take as inputs:
   - the fluid property information in each internal and environment zone;
   - a vector of parameters controlling the CFD computation.
2. Compute as outputs:
   - the mass flowrate in each of the two directions of each interface (method `MassFlowrate`);

- the pressure in each zone (method `ZonePressure`);
- other CFD-estimated properties (e.g. viscosity in a non-Newtonian fluid) which may be required by the internal and environment zones. The method name may be chosen from a library containing available computations (e.g. `NonNewtonViscosity`, `DissipationEnergy`, etc.); we will refer to them by the general name `CFDProperties`.

In view of the functional description of method `MassFlowrates` provided, the `ZoneNetwork` model needs to perform a number of operations:

1. Copy the fluid property information from the individual zones into a global vector of fluid property information, `GlobalFluidProperty`; the dimension of the latter is (`NoIZone` + `NoEZone`) × `NoFluidProperty`.

2. Invoke the methods `ZonePressure`, `MassFlowratesLeftToRight`, `MassFlowratesRightToLeft` and `CFDProperties`:

   (a) passing to each one of them the vectors `GlobalFluidProperty` and `CFDParameter` as inputs;

   (b) obtaining the following information as their outputs:

   - `ZonePressure`: a vector of pressure variables `P(.)` of dimension (`NoIZone` + `NoEZone`);
   - `MassFlowratesLeftToRight`: a vector of mass flowrate variables `F(.,1)` of dimension (`NoIIInterface`+`NoIEInterface`);
   - `MassFlowratesRightToLeft`: a vector of mass flowrate variables `F(.,2)` of dimension (`NoIIInterface`+`NoIEInterface`);
   - `CFDProperties`: a vector of zone property variables `CFDMethods(i,.)` of length (`NoIZone` + `NoEZone`).

3. Equate the pressures `P` to the appropriate pressures in all internal and environment zones in the network.

4. Equate the mass flowrates `F` to the appropriate flowrates in the left and right ports of each interface.

5. Equate each zone property `CFDMethods(i)` (`i=1,...,NoCFDMethods`) to the appropriate methods in all internal and environment zones in the network.

The `ZoneNetwork` also needs to establish the connections between the `Inlet` and `Outlet` ports of the various zones as determined by the interfaces in the network, thereby

ensuring the equality of the intensive and other properties appearing in these ports. ZoneNetwork model is shown in Table 3.4.

Table 3.4: Variables in the **ZoneNetwork** model.

```
MODEL ZoneNetwork

PARAMETER  .....

UNIT  .....

VARIABLE
 GlobalFluidProperty AS ARRAY(NoIZone+NoEZone, NoFluidProperty)
                          OF NoType
 CFDParameter AS ARRAY(NoCFDParameter) OF NoType

 P AS ARRAY(NoIZone+NoEZone) OF Pressure
 F AS ARRAY(NoIIInterface+NoIEInterface, 2) OF MassFlowrate
 CFDMethods AS ARRAY(NoCFDMethods,NoIZone+NoEZone) OF NoType

SET  .....

EQUATION
 # Copy fluid property information from individual zones into global
 # vector
 FOR i := 1 TO NoIZone DO
  GlobalFluidProperty(i, ) = IZone(i).FluidProperty ;
 END
 FOR i := 1 TO NoEZone DO
  GlobalFluidProperty(i+NoIZone, ) = EZone(i).FluidProperty ;
 END

 CFDparameters = ZoneNet.Parameters() ;

 # Invoke CFD computation
 P = ZoneNet.ZonePressure(GLobalFluidProperty,CFDParameter) ;
 F(,1) = ZoneNet.MassFlowratesLeftToRight
                 (GLobalFluidProperty,CFDParameter) ;
 F(,2) = ZoneNet.MassFlowratesRightToLeft
                 (GLobalFluidProperty,CFDParameter) ;

 FOR i := 1 TO NoCFDMethods DO
  CFDMethods(i)=ZoneNet.CFDProperties(GLobalFluidProperty,CFDParameter) ;
 END

 # Internal-Internal Interfaces
 FOR i := 1 TO NoIIInterface DO
  # Assign computed mass flowrates to individual interfaces
  F(i,1) = IZone(IILeftZone(i)).Fout(IILeftPort(i))
           = IZone(IIRightZone(i)).Fin(IIRightPort(i)) ;
  F(i,2) = IZone(IILeftZone(i)).Fin(IILeftPort(i))
           = IZone(IIRightZone(i)).Fout(IIRightPort(i)) ;

  # Establish interface connections for intensive properties
  IZone(IILeftZone(i)).Outlet =
```

```
            IZone(IIRightZone(i)).Inlet(IIRightPort(i)) ;
 IZone(IIRightZone(i)).Outlet =
            IZone(IILeftZone(i)).Inlet(IILeftPort(i)) ;
END

# Internal-Environment Interfaces
FOR i := 1 TO NoIEInterface DO
 # Assign computed mass flowrates to individual interfaces
 F(i+NoIIInterface,1)
            = IZone(IELeftZone(i)).Fout(IELeftPort(i))
            = EZone(IERightZone(i)).Fin(IERightPort(i)) ;
 F(i+NoIIInterface,2)
            = IZone(IELeftZone(i)).Fin(IELeftPort(i))
            = EZone(IERightZone(i)).Fout(IERightPort(i)) ;

 # Establish interface connections for intensive properties
 IZone(IELeftZone(i)).Outlet =
            EZone(IERightZone(i)).Inlet(IERightPort(i)) ;
 EZone(IERightZone(i)).Outlet =
            IZone(IELeftZone(i)).Inlet(IELeftPort(i)) ;
END

# Zone pressures
FOR i := 1 TO NoIZone DO
 P(i) = IZone(i).P ;
END
FOR i := 1 TO NoEZone DO
 P(i+NoIZone) = EZone(i).P ;
END

 # Zone properties
 FOR i := 1 TO NoCFDMethods DO
  FOR j := 1 TO NoIZone DO
   CFDProperty(i,j) = IZone(j).Property(i) ;
  END
  FOR j := 1 TO NoEZone DO
   CFDProperty(i,j+NoIZone) = EZone(j).Property(i) ;
  END
 END

END # model ZoneNetwork
```

We also assume that each environment interface may accept *only* one non-null flowrate (either inlet or outlet). Although the interface is designed so as to allow two mass flowrates (one inlet *and* one outlet), every interface with the external environment will always set one mass flowrate equal to 0. As a result, every internal zone will have as many neighbouring environment zones as the number of inlets/outlets to/from the zone.

3.5.3 Additional Information for Process Simulation

The `ZoneNetwork` model presented above obtains most of the required information through direct access to methods provided by a foreign object of class `gCFD`. The foreign object is a gPROMS parameter and as such a "value" must be set. This issue will discussed in chapter 5.

Other additional details are required in order to have a well-defined mathematical problem that can be solved using simulation. As a minimum, this includes:

a) the vector of CFD parameters, `CFDParameters`;

b) information sufficient to allow the computation of the intensive properties in the outlet direction of all environment zone ports and the fluid properties in all environment zones;

c) information sufficient to determine the initial state in each internal zone.

The information under points *(a)* and *(b)* can be specified:

- as a set of constant values (e.g. assigned in a gPROMS `PROCESS` section);
- as a set of time-varying quantities, each with a given time variation (e.g. assigned in a gPROMS `PROCESS` section and/or modified in a gPROMS `SCHEDULE` either explicitly or via a foreign process);
- in terms of a set of equations relating them to other process quantities (e.g. control laws specified as equations in the `PROCESS` section or via stream connectivity with other model instances in a higher-level model).

The `CFDParameters` vector is a useful vehicle to pass arguments for the CFD computations. For instance, it may be used to set and change hydrodynamics parameters (e.g. the rotation speed of an impeller) along the simulation. Furthermore, it may be used to set numerical parameters such as convergence criteria in the CFD package or the way CFD computations are issued (see chapter 7).

3.6 The Work Process in a Zone Network Model

In chapter 2 (§ 2.3) the general ideas concerning the design and work process of the interfacing architecture were described. Here those concepts will be re-stated in view of the new zone design. A typical work process will involve the following steps:

Step 1: *Geometry definition*

- Using a CFD pre-processor, define the geometry of the equipment of interest.

Step 2: *Process simulation grid definition (Zoning)*

- Specify any additional information required by the CFD computation (e.g. boundary conditions, fluid properties).
- Perform a preliminary CFD calculation.
- From the computed flow field, determine an appropriate set of regions.
- Define the regions using an appropriate tool and export the information on the grid in an appropriate format (e.g. in a ASCII file).

Step 3: *Problem specification*

- Using the domain-specific tool, read in the grid information exported at the previous step.
- Provide any additional information necessary to complete the specification of the problem. These will typically be used to form a foreign object supplying services to the gPROMS simulation.

Step 4: *Run simulation*

- From within the domain-specific tool, start the simulation. This creates and exposes one or more foreign object and/or output channel interfaces. It then starts a gPROMS run with the pre-existing gPROMS input file.
- Interact with the simulation via the domain-specific tool.
- Terminate the simulation.

3.7 An Example: Reactors

The interface described in this chapter will now be applied to the general class of *isothermal reactors*. This will show how the model can be set up through a network of well-mixed zones. In particular, the flexibility and practicability of the suggested structure will be demonstrated. It will be shown how a zone network model may be implemented by defining the internal and environment zone models, while the zone network is independent of those specific models. The simulation side reactor model

Symbol	Description
c	Number of components
F^{in}, F^{out}	Inlet and outlet mass flowrates, respectively
M_i	Mass holdup of component i
M_T	Total mass holdup
ni, no	Number of inlets and outlets, respectively
nr	Number of reactions
P	Pressure
R_r	Reaction expression r
T	Temperature
V	Zone volume
v	Specific volume
w_i	Molecular weight of component i
X_i^{in}, X_i	Inlet and reactor mass fraction of component i, respectively
nu_{ir}	Stoichiometric coefficients for component i and reaction r

Table 3.5: List of symbols for isothermal reactor model.

(list of symbols in Table 3.5) for each well mixed zone is constituted of the following system of equations:

$$\frac{dM_i}{dt} = \sum_{j=1}^{ni} F_j^{in} X_{i,j}^{in} - \left(\sum_{j=1}^{no} F_j^{out} \right) X_i + V w_i \sum_{i=1}^{nr} \nu_{ir} R_r(X,T) \qquad (3.1)$$

$$i = 1, 2, \ldots, c-1$$

$$M_T = \sum_{i=1}^{c} M_i \qquad (3.2)$$

$$X_i M_T = M_i \qquad i = 1, 2, \ldots, c \qquad (3.3)$$

$$V = M_T v(X, T, P) \qquad (3.4)$$

Assuming that the inlet streams are completely specified (i.e. F_j^{in}, $X_{i,j}^{in}$ are known function of time) as well as the outlet mass flowrates F_j^{out}, the above is a system of $(2c+1)$ DAEs in the $(2c+1)$ unknowns M_i, X_i $(i = 1, \ldots, c)$, M_T (pressure P is obtained from CFD calculations).

We will now demonstrate the way in which the structure discussed in the previous sections is implemented in this specific case.

3.7.1 `InternalZone` Design

The above equations have to be included in the `InternalZone` model structure. A general class of foreign objects was designed and implemented to facilitate the user's modelling of general-purpose reactors. The object is a modified version of the MultiFlash interface. MultiFlash is a standard package providing thermodynamics properties, which may be linked to gPROMS through a standard procedure (gPROMS Advances User's Guide, 1999). We used the existing implementation to design an object capable of including reactions in a gPROMS model according to a user-friendly approach. This object represent a new and general approach to include kinetics in a gPROMS model[1]. The idea is to allow the description of a set a reactions taking place in a *material* whose thermodynamics properties are defined by MultiFlash.

For non-equilibrium reactions, it is assumed that each reaction may be described by the Arrhenius formula:

$$R_r = A_r \exp\left(-\frac{E_r}{RT}\right) \prod C_i^{\pi_i}, \tag{3.5}$$

where A_r and E_r are the Arrhenius factor and activation energy, respectively. Eqn. (3.5) is expressed via the concentration vector **C**, but alternative solutions (mass fraction, partial pressure) are available.

In the case of equilibrium reactions, expression (3.5) is replaced by an equilibrium relation of the form:

$$-RT \log K_r = \sum \nu_{ir} G_i^\circ \equiv \Delta G_r^\circ \tag{3.6}$$

where K_r is the equilibrium constant and ΔG_r° is the standard Gibbs energy change of reaction.

The scope of the the object (`ReactingMaterial`) is to return physical properties (from MultiFlash) and the constants and variables required to describe the set of reactions. It will have to return:

- the number of chemical species (method `NoComp`);

- the number of reactions taking place (method `NReact`);

---

[1] I would like to acknowledge Dr. Anthony Kakhu for his help in the object implementation within the MultiFlash interface

- the stoichiometry of the reactions occurring in the reactor: the stoichiometry is returned by means of a matrix containing the stoichiometric coefficients for each species and reaction in the system (ν_{ir} in eq. (3.1)). The method is called `StoichMatrix`;

- the vector of reaction rates R_r at specified temperature T, pressure P and composition (method `ReactionRates`);

- the vector equilibrium constant K_r at a specified temperature (method `EquilibriumConstants`);

- the vector of molecular weights (method `MolecularWeight`).

Internally, the methods will need a set of information provided by the user:

1. the name of the Multiflash file describing the material that takes part in these reactions;

2. the number of reactions in the system;

3. the type of each reaction:
   - equilibrium
   - irreversible
   - reversible;

4. the way composition is declared (units of measurement);

5. the phase (liquid, vapour, both) in which reactions occur;

6. the stoichiometry of the reactions, i.e. for each species and for each reaction:
   - name of the species involved
   - stoichiometric coefficients;

7. kinetic/equilibrium data, i.e. in case of equilibrium reaction:
   - standard Gibbs energy change of reaction ΔG°_r;

   otherwise:
   - order coefficients π_i (forward and backward if the reaction is reversible)
   - Arrhenius factor A_r (forward and backward if the reaction is reversible)
   - activation energy E_r (forward and backward if the reaction is reversible).

Table 3.6 contains an example of a `ReactingMaterial` declaration command file for a system where only one first order reaction $A \rightarrow B$ occurs.

Table 3.6: Command file for the `ReactingMaterial` object.

```
MultiFlash_Commandfile_Name
REACTION_NUMBER 1
 DATA_REACTION 1
 TYPE Irreversible
 UNITS Concentration
 PHASE Liquid
 STOICHIOMETRY
  A 1
  B -1
 END-STOICHIOMETRY
 ORDER
  A 0
  B 1
 END-ORDER
 ARRHENIUS_FACTOR 5e11
 ACTIVATION_ENERGY 6e4
END-REACTION
```

The `Internal_Zone` model described in § 3.4 can be easily used to include a reactor model within which `ReactingMaterial` methods are also implemented. The modified `InternalZone` model is illustrated in Table 3.7.

Table 3.7: `InternalZone` model: case of a isothermal reactor.

```
MODEL InternalZone

PARAMETER
 # Parameters common to all models
 NoPort AS INTEGER
 NoFluidProperty AS INTEGER

 # Volume of zones
 V AS REAL

 # Other Parameters
 Material AS FOREIGN_OBJECT "ReactingMaterial"
 NoComp AS INTEGER
 T AS REAL

 # kinetic parameters
 Nreact AS INTEGER
 StoichMatrix AS ARRAY(Nreact,NoComp) OF REAL

VARIABLE
 # Mass flowrates
```

```
Fin, Fout AS ARRAY(NoPort) OF MassRate

# Fluid properties
Fluidproperty AS ARRAY(NoFluidProperty) OF NoType

# Pressure
P AS Pressure

# Other variables
 # Molar holdup
 M AS ARRAY(NoComp) OF Moles
 # Total molar holdup
 MT AS Moles
 # Molecular weight
 MW AS MolarWeight
 # Molecular weight in the inlet streams
 MWIn AS ARRAY(NoPort) OF MolarWeight
 # Molar fraction
 X AS ARRAY(Nocomp) OF Fraction
 # Molar fraction in the inlet streams
 Xin AS ARRAY(NoPort,Nocomp) OF Fraction

STREAM
 Inlet:  Xin, MwIn AS ARRAY(NoPort) OF Interface
 Outlet:  X, Mw AS Interface


SET
 Nreact := Material.NumberOfReactions ;
 StoichMatrix := Material.StoichiometricCoefficients ;
 NoComp := Material.NumberOfComponents ;

EQUATION
 # Definition of fluid properties
 # density
 FluidProperty(1) = Material.VapourDensity(T,P,x) ;
 # viscosity  FluidProperty(2) = Material.VapourViscosity(T,P,x) ;

 # Component material balances:  eqn.  (3.1)
 FOR i := 1 TO NoComp-1 DO
  $M(i)$ = SIGMA(Fin*XIn(,i)/(MwIn*1e-3))
           - SIGMA(Fout/(Mw*1e-3))*X(i)
           + V * SIGMA(StoichMatrix(,i)*
           Material.ReactionRates(T,P,x)) ;
 END

 # Total molar holdup:  eqn.  (3.2)
 MT = SIGMA(M) ;

 # Mole fractions in reactor:  eqn.  (3.3)
 MT * X = M ;

 # Mean molecular weight
 MW = SIGMA(X * Material.MolecularWeight()) ;

END # model InternalZone
```

ReactingMaterial is set as a parameter. As said, fluid properties (viscosity and density) are estimated by means of the same object which is linked to the thermodynamics library of MultiFlash. In the EQUATION section we can see that there is only one set of differential equations: they calculate the molar holdup M in the volume.

3.7.2 EnvironmentZone Design

Table 3.8 contains the EnvironmentZone model. No significant differences are introduced in the model, apart from the use of the object ReactingMaterial to set the number of species and estimate the viscosity and density values.

Table 3.8: EnvironmentZone model: case of a isothermal reactor.

```
MODEL EnvironmentZone

PARAMETER
 # Number of Ports and Fluid Properties
 NoPort AS INTEGER
 NoFluidProperty AS INTEGER

 # Other parameters
 Material AS FOREIGN_OBJECT "ReactingMaterial"
 NoComp AS INTEGER
 T AS REAL

VARIABLE
 # Mass flowrates
 Fin, Fout AS ARRAY(NoPort) OF MassRate

 # Fluid properties
 Fluidproperty AS ARRAY(NoFluidProperty) OF NoType

 # Pressure
 P AS Pressure

 # Other variables
 X AS ARRAY(NoComp) OF Fraction
 Xin AS ARRAY(NoPort,NoComp) OF Fraction
 MW AS MolarWeight
 MWIn AS ARRAY(NoPort) OF MolarWeight

STREAM
 Inlet:  XIn,MwIn AS ARRAY(NoPort) OF Interface
 Outlet:  X,Mw AS Interface

SET
 NoComp := Material.NumberOfComponents ;
```

```
EQUATION
 # Definition of fluid properties
 #density
 FluidProperty(1) = Material.LiquidDensity (T,P,x) ;
 #viscosity
 FluidProperty(2) = Material.LiquidViscosity(T,P,X) ;

END # model EnvironmentZone
```

The `Network` model does not change. Its general design does not require any modification. Needed parameters and variables are retrieved from the CFD computations and a database containing information concerning the zone network (see chapter 5).

3.8 Key Results

The main achievements illustrated in this chapter are:

- The definition of a *general interface* capable of exchanging information between a gPROMS model and a CFD model through a very compact and efficient structure. Specific types of processing equipment (e.g. reactors, crystallisers) are modelled by means of a structure operating with fixed process simulation files capable of handling hydrodynamic information obtained by a CFD package.

- The definition of a *language* allowing an easy to use representation of a simulation model mapping an equipment domain by means of a *network of zones.* As shown in the isothermal reactor example, a model can be defined by means of *internal* and *environment* zone models. The entire domain is mapped by means of a zone network model which is *independent* of the specific model for each zone. Thus, simulation models can be easily changed without modifying the overall network model. The internal zone design allows an easy definition of the simulation model for each well-mixed zone.

- The design of a zone representation *mapping* the fine mesh used in the CFD model. A correspondence is achieved between the fine CFD grid and the network of zones: each internal zone represents a set of CFD cells and each cell is part of a zone.

- The design of a general method for treating *reactive systems* within process simulation models. A user-friendly procedure is designed to incorporate reactive systems within a process simulation model.

Chapter 4

The Constant Volume Assumption

The interface proposed in the previous chapter defines a general architecture for integrating a process modelling tool and a CFD package. In this chapter our attention will be focused on an important numerical and physical issue underlying one of the fundamental assumptions of our design, i.e. the fact that zone volumes are assumed to be constant throughout the simulation.

4.1 Zone Model Assumptions

The coupling of a process simulation zone network model with a CFD simulation model is based on the following central assumptions:

- the fluid dynamics act on a much faster scale than the rest of the physical phenomena of interest;
- the flow field is updated only from time to time by performing a new steady-state CFD calculation;
- the total mass flowrates between the simulation zones are calculated by the CFD package and are held constant between two CFD calculations;
- the zone volumes are provided by the CFD package and are held fixed throughout the whole dynamic process simulation [1] ;

[1] If a batch process is being simulated, this assumption involves constant mass in the system and, therefore, constant average density in the domain.

- the fluid densities within the zones are calculated by the process simulation package and may vary between two CFD calculations.

One contradiction that may result from the above statements regards the fixed volume assumption and the simultaneous possibility for the fluid densities to vary. It may appear that either the volumes or the mass holdups are not conserved.

Let V be the volume of a given internal zone and ρ the current density within that zone, $M_T = \rho V$ the corresponding total mass holdup in the zone, F_i^{in} and F_i^{out} its inlet and outlet mass flowrates and ρ_i^{in} the inlet densities. Then the steady-state (remember that CFD calculations are performed in a steady-state simulation) mass balance for the internal zone is

$$\frac{dM_T}{dt} = \sum_{i=1}^{ni} F_i^{in} - \sum_{i=1}^{no} F_i^{out} = 0, \tag{4.1}$$

where ni and no are the number of inlets and outlets, respectively. This equation is satisfied for the mass flowrates calculated by each steady-state CFD simulation, based on the current densities provided by gPROMS.

The fixed volume assumption for an internal zone can be written as:

$$\frac{dV}{dt} = \sum_{i=1}^{ni} \frac{F_i^{in}}{\rho_i^{in}} - \frac{1}{\rho} \sum_{i=1}^{no} F_i^{out} + \Phi^V = 0, \tag{4.2}$$

where we have included the volume source Φ^V in order to describe volume changes due to chemical reactions, mixing or changes of temperature and pressure. For the densities used in, and the mass flowrates computed by, the CFD simulation, the magnitude of the volume source is determined by eqn. (4.2). However, if the mass flowrates F_i^{in} and F_i^{out} are now "frozen" during the dynamic gPROMS simulation of the zone network, then eqn. (4.2) is violated whenever the densities ρ_i^{in}, ρ and the volume source Φ^V are allowed to change dynamically. While eqn. (4.1) ensures that the total mass holdup M_T in the internal zone remains constant, it is clear that the zone volume V *must* change for non-constant density ρ according to

$$V = \frac{M_T}{\rho}$$

for the mass balance to be fulfilled. In other words, there may be an inconsistency between the results of the CFD calculations fulfilling eqn. (4.1) and the gPROMS calculations which will satisfy the conditions expressed by eqn. (4.2).

However, we have to remember that:

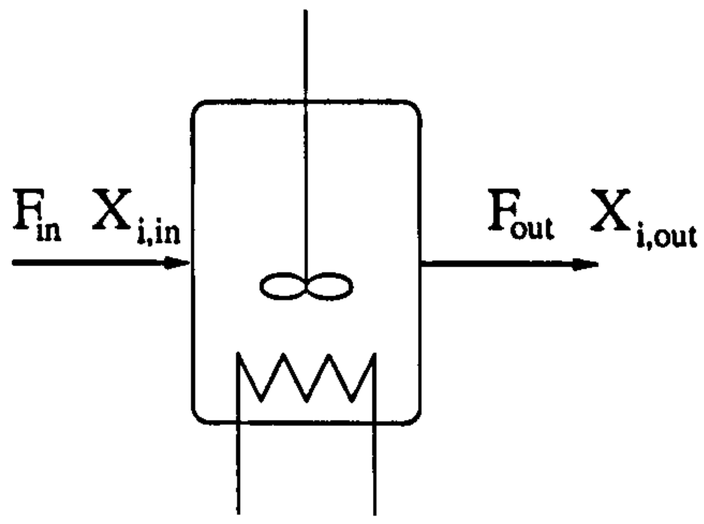


Figure 4.1: Incompressible well-mixed reactor.

- A *zone* corresponds to a fixed part of space under consideration. Consequently, it has a fixed volume by definition.

- From the practical point of view, a zone corresponds to a fixed subset of the control volumes in the CFD mesh. Again, this directly implies that its volume is constant (we assume to use standard fixed mesh).

- Unlike mass or energy, volume is not a conserved quantity. There is no independent way of determining Φ^V. In fact, eqn. (4.2) *is* the only way of determining Φ^V, e.g. for a fixed volume zone, we have $dV/dt = 0$ and therefore:

$$\Phi^V = -\sum_{i=1}^{no} \frac{F_i^{in}}{\rho_i^{in}} + \frac{1}{\rho} \sum_{i=1}^{no} F_i^{out}.$$

Thus, the main responsibility for modelling a system so that mass is conserved is borne by the gPROMS models of the individual zones and the overall zone network. The issue will be examined by considering the general class of reactor models. First, incompressible fluids will be discussed; then compressible fluids will be taken into account.

4.2 Incompressible Well-Mixed Reactor

Let us consider an incompressible well-mixed reactor as illustrated in Figure 4.1. The model may represent any internal zone in the integrated model of a reactor containing an incompressible fluid. We assume that there are one inlet F^{in} and one output F^{out} (this simplification will not affect the generality of our conclusions; differences and specifications regarding the multi-inlet/ outlet case will be discussed in some footnotes). In order to ensure that mass balance is always fulfilled, all components are considered. The system of equations representing the reactor is as follows:

- mass balance:

$$\frac{dM_i}{dt} = F^{in}X_i^{in} - F^{out}X_i + Vw_i\sum_{i=1}^{nr}\nu_{ir}R_r(X,T) \qquad i = 1,2,\ldots,c \tag{4.3}$$

- energy balance (Q is a known heat source):

$$\frac{dU}{dt} = F^{in}h^{in} - F^{out}h + Q \tag{4.4}$$

- total mass in the volume:

$$M_T = \sum_{i=1}^{c} M_i \tag{4.5}$$

- mass fractions:

$$X_iM_T = M_i \qquad i = 1,2,\ldots,c \tag{4.6}$$

- internal energy:

$$U = M_t u(X,T) \tag{4.7}$$

- volume constraint:

$$V = M_T v(X,T) \tag{4.8}$$

- enthalpy computation:

$$h = h(X,T) \tag{4.9}$$

where w_i is the molecular weight of component i and u, v are the specific internal energy and volume respectively.

This comprises $(2c+5)$ DAEs in the $(2c+4)$ unknowns M_i, X_i, M_T, U, T, h. Note that we are not counting as variables either V (which is a constant) or the mass flowrates F_j^{in}, F_j^{out} as these are specified externally (from CFD calculations). The number of equations in the above system exceeds the number of variables by one. Therefore, one equation must be either redundant or inconsistent. Clearly, inconsistency is undesirable as it would mean that the system has no solutions. We can make the system consistent by ignoring one of the flowrate values returned by the CFD code for each zone, instead

treating it as an unknown to be worked out from the model equations. The other available solution would be to drop one of eqns. (4.3) without adding any new variable as in model in § 3.7. This solution, however, is not advisable since corrections after $\Phi^V \neq 0$ in eqn. (4.2) would affect the overall mass balance. On the contrary, *by choosing a mass flowrate as a new unknown variable, we ensure that mass balance is always fulfilled.* Of course, this does not answer the question whether the solution is *physically* consistent. This issue will be later discussed in § 4.6.

Flowrate F^{out} is chosen as the new unknown. Assuming that the inlet stream is completely specified (i.e. F^{in}, X^{in}, h^{in} are known functions of time), the above is a system[2] of $(2c+5)$ DAEs in the $(2c+5)$ unknowns M_i, X_i $(i = 1, \ldots, c)$, U, M_T, T, F^{out}, h. However, it is quite easy to observe that the new set of DAEs present some typical complications of DAE systems. In fact, if we specify the initial conditions $M_i(0), i = 1, \ldots, c$ and $U(0)$, we can calculate $M_T(0)$ from eqn. (4.5), $X(0)$ from eqns. (4.6), T from eqn. (4.7), but at this point we realise that both side of eqn. (4.8) are already known, i.e. the system is of index greater than 1 (Pantelides, 1988). During the last twenty years several works have appeared to improve our understanding of DAE systems (e.g., Gear and Petzold, 1984, Pantelides, 1988, Unger *et al.*, 1995). The above reactor model may be reduced to an equivalent Index-1 set of equations.

4.2.1 Reactor Model Index Reduction

The DAE system (4.3)–(4.9) is of index 2 or higher. In order to determine the set of consistent initial conditions, we will apply the algorithm suggested by Pantelides (1988) for the structural analysis of DAE systems. The underlying idea of this algorithm is to determine a subset of k equations, which upon differentiation produce fewer than k new variables. This so called *minimal structural singular subset* is differentiated and the algorithm is performed again, until no further minimal structurally singular subset is identified. Note that "new variable" is meant in the context of a consistent initialisation problem, where $\mathbf{u}(0)$ and $\dot{\mathbf{u}}(0)$ are considered to be distinct variables.

---

[2] If there are several in/out flowrates (*ni* inlets and *no* outlets respectively), the first two sets of equations (4.3) and (4.4) become:

$$\frac{dM_i}{dt} = \sum_{j=1}^{ni} F_j^{in} X_{ij}^{in} - \left(\sum_{j=1}^{no} F_j^{out}\right) X_i + V w_i \sum_{i=1}^{NR} \nu_{ir} R_r(X,T) \qquad i = 1, \ldots, c \qquad (4.3')$$

$$\frac{dU}{dt} = \sum_{j=1}^{ni} F_j^{in} h_j^{in} - \left(\sum_{j=1}^{no} F_j^{out}\right) h_i + Q, \qquad (4.4')$$

but the number of unknowns (and equations) is obviously the same, with F^{out} replaced by one (and only one) F_j^{out}. The other outlets are considered to be specified.

The algorithm examines the structure of the system by means of a so called *incidence matrix.* Let us consider the reactor model. The incidence matrix ($2c+5$ equations for $3c+6$ unknowns) for the system is:

	$\dot{M}_i$	$\dot{U}$	M_i	U	X_i	M_T	T	F^{out}	h
Eqns. (4.3)	×				×		×	×	
Eqn. (4.4)		×						×	×
Eqn. (4.5)			×			×			
Eqns. (4.6)			×		×	×			
Eqn. (4.7)				×	×	×	×		
Eqn. (4.8)					×	×	×		
Eqn. (4.9)					×		×		×

If we differentiate the $c+3$ equation subset constituted of eqns. (4.5), (4.6), (4.7) and (4.8), it is easy to observe that only $c+2$ new variables ($\dot{X}_i$, $\dot{M}_T$ and $\dot{T}$) will be produced. After differentiation, the incidence matrix shows that no more singular subsets of equations exist. Concluding, the algorithm has produced $c+3$ new equations by differentiation. This leaves $3c+8$ equations to determine $4c+8$ unknowns at the initialisation step and, thus, there are c (and not $c+1$) dynamic degrees of freedom, according to (Neumann, 2001):

Definition 1 *Variables* $\mathbf{u}$ *or their time derivatives* $\dot{\mathbf{u}}$ *which can be assigned arbitrary initial conditions and still allow consistent initialisation are called dynamic degrees of freedom.*

The number of differentiations that each equation undergoes may be used to estimate the differentiation index. This is based on the rule that the index should equal the maximum number of times any equation was differentiated if the derivative of every variable appears in the final system of equations produced by the algorithm. If the derivatives of some variables do not appear in the final system (F^{out} in the present case), the index should be one greater than the maximum number of differentiations the algorithm has performed. Thus, the reactor model DAE system is at index 2.

To reduce the index, differentiate eqn. (4.8) with respect to time:

$$0 = \dot{M}_T v(X,T) + M_T \left(\sum_{i=1}^{c} \frac{\partial v}{\partial X_i} \frac{dX_i}{dt} + \frac{\partial v}{\partial T} \frac{dT}{dt} \right) \tag{4.10}$$

First consider $\frac{dX_i}{dt}$. From eqn.(4.6), we have:

$$X_i = \frac{M_i}{M_T} \quad \Rightarrow \quad \frac{dX_i}{dt} = \frac{\dot{M}_i}{M_T} - \frac{\dot{M}_i}{{M_T}^2}\dot{M}_T = \frac{\dot{M}_i}{M_T} - \frac{X_i}{M_T}\dot{M}_T. \tag{4.11}$$

Inverting eqn. (4.11) in eqn. (4.10), we obtain:

$$0 = \dot{M}_T\left(v - \sum_{i=1}^{c} X_i \frac{\partial v}{\partial X_i}\right) + \sum_{i=1}^{c} \frac{\partial v}{\partial X_i}\frac{dM_i}{dt} + M_T \frac{\partial v}{\partial T}\frac{dT}{dt}. \tag{4.12}$$

An expression for dT/dt can be obtained from eqn. (4.7) by differentiating with respect to time:

$$\frac{dU}{dt} = \dot{M}_T\left(u - \sum_{i=1}^{c} X_i \frac{\partial u}{\partial X_i}\right) + \sum_{i=1}^{c} \frac{\partial u}{\partial X_i}\frac{dM_i}{dt} + M_T \frac{\partial u}{\partial T}\frac{dT}{dt}. \tag{4.13}$$

We could use eqn. (4.13) to obtain dT/dt in terms of the other quantities appearing in it; then substitute this expression in eqn. (4.12), replace dM_i/dt by eqns. (4.3) and (4.4), so finally obtaining and algebraic expression for F^{out}. However, this algebraic manipulation is unnecessary: all we need to do is add the two equations

$$0 = \left(\sum_{i=1}^{c} \frac{dM_i}{dt}\right)\left(v - \sum_{i=1}^{c} X_i \frac{\partial v}{\partial X_i}\right) + \sum_{i=1}^{c} \frac{\partial v}{\partial X_i}\frac{dM_i}{dt} + M_T \frac{\partial v}{\partial T} D_T \tag{4.12'}$$

and

$$\frac{dU}{dt} = \left(\sum_{i=1}^{c} \frac{dM_i}{dt}\right)\left(u - \sum_{i=1}^{c} X_i \frac{\partial u}{\partial X_i}\right) + \sum_{i=1}^{c} \frac{\partial u}{\partial X_i}\frac{dM_i}{dt} + M_T \frac{\partial u}{\partial T} D_T \tag{4.13'}$$

to our model eqns. (4.3)–(4.9), treating D_T as a new variable (i.e. *not* using the relation $D_T \equiv dT/dt$).

Of course, now we have added two new equations but only one new variable, so the system is over-specified. We need to drop one of the original equations (4.3)–(4.9). The original index reduction method (e.g. by Gear and Petzold, 1984, Gear, 1990) would drop the algebraic constraint (4.8). On the other hand, the more recent method of Mattsson and Söderlind (1993) (based on Bachmann *et al.*, 1990) would drop one of the differential equations (4.3). Mathematically, the two are completely equivalent. However, numerically it is better to drop one of the equations (4.3) so as to ensure that no numerical "drift" of eqn. (4.8) takes place. Note that it is unimportant which one of eqns. (4.3) we drop: the implicit DAE integrator we use treats all variables (differential and algebraic) in a similar manner during the corrector iterations and all of them can

be subjected to bounds. Hence we can always ensure that $M_i \geq 0$, $\forall i$.

Overall, after the changes the **zone model** becomes a square, Index-1 DAE system. The system has $2c+6$ equations, i.e. (4.3) ($i = 1, \ldots, c-1$ only), (4.4)–(4.9), (4.12′) and (4.13′), in the variables: M_i, X_i ($i = 1, \ldots, c$), U, M_T, T, F^{out}, h, D_T. The model is mathematically equivalent to model (4.3)–(4.9): it conserves mass of all species for any temporal variation of the input variables $F^{in}(t)$, $X^{in}(t)$, $h^{in}(t)$, $Q(t)$.

Incompressible fluids create numerical issues within CFD solvers, too. Most CFD codes use either a method called *Semi-Implicit Method for Pressure-Linked Equations* (SIMPLE) developed by Caretto *et al.* (1972) or a slight modification of the same method called SIMPLEC (van Doormal and Raithby, 1984) to solve the Navier-Stokes equations. Neumann (2001) demonstrated that the underlying reason for such algorithms is the need to solve an index-3 problem inherent to systems where the density is independent of the pressure. Since the CFD codes do not perform a proper index reduction of the system of partial differential equations under consideration, an iterative procedure, which is in fact very similar to the proper index reduction, has been established in the past without a sound understanding of the mathematical reason for doing so.

In this section we have obtained a derivation for the general case of an incompressible non-ideal, non-isothermal system. Now we will describe some special cases, which may be found in the process industry. It will be shown how in such cases the model can be easily represented without mathematical complications.

4.2.2 I. Ideal Mixing with Negligible Temperature Effect on Density

In this case,

$$v(X,T) \approx v(X) = \sum_{i=1}^{c} X_i v_i^0, \tag{4.14}$$

where v_i^0 is the specific volume of pure component i.
Therefore $\frac{\partial v}{\partial X_i} = v_i^0$ and $\frac{\partial v}{\partial T} = 0$. Substituting these and eqn. (4.14) in eqn. (4.12) we obtain:

$$0 = \sum_{i=1}^{c} v_i^0 \frac{dM_i}{dt} \,. \tag{4.15}$$

Combining eqn. (4.15) with (4.3), we obtain:

$$0 = \sum_{i=1}^{c} v_i^0 F^{in} X_i^{in} - \sum_{i=1}^{c} v_i^0 F^{out} X_i + V \sum_{i=1}^{c} x_i v_i^0 \sum_{r=1}^{nr} \nu_{ir} R_r$$

which leads to the common volumetric flow relation:

$$F^{out} v = F^{in} v^{in} + V \sum_{r=1}^{nr} R_r \sum_{i=1}^{c} w_i v_i^0 \nu_{ir} \tag{4.16}$$

where $\sum_{i=1}^{c} w_i v_i^0 \nu_{ir}$ is the volumetric change per mol of reaction r.

4.2.3 II. Ideal Mixing with Non-Zero Coefficient of Liquid Expansion

Now

$$v(X,T) = \sum_{i=1}^{c} X_i \left(v_i^0 + \alpha_i T\right) \tag{4.17}$$

where α_i is the volumetric expansion for pure component i.

From eqn. (4.17) we derive

$$\begin{cases} \dfrac{\partial v}{\partial X_i} &= v_i^0 + \alpha_i T \\ \dfrac{\partial v}{\partial T} &= \sum_{i=1}^{c} X_i \alpha_i \end{cases}$$

Substituting in eqn. (4.12) we obtain

$$0 = \sum_{i=1}^{c} \left(v_i^0 + \alpha_i T\right) \frac{dM_i}{dt} + M_T \left(\sum_{i=1}^{c} X_i \alpha_i\right) \frac{dT}{dt} \tag{4.18}$$

where we can get $\dfrac{dT}{dt}$ from eqn. (4.13). For an ideal mixture, $\sum_{i=1}^{c} X_i u_i^0(T) = u$, therefore eqn. (4.13) becomes:

$$\frac{dT}{dt} = \frac{1}{M_T c_v} \left(\frac{dU}{dt} - \sum_{i=1}^{c} u_i^0 \frac{dM_i}{dt}\right) \tag{4.19}$$

where c_v is the specific heat capacity at constant volume.

Combining eqns. (4.18) and (4.19), we obtain the constraint:

$$0 = \sum_{i=1}^{c} \left(v_i^0 + \alpha_i T\right) \frac{dM_i}{dt} + \frac{1}{c_v} \left(\sum_{i=1}^{c} X_i \alpha_i\right) \left(\frac{dU}{dt} - \sum_{i=1}^{c} u_i^0 \frac{dM_i}{dt}\right) \tag{4.20}$$

which can be used to determine F^{out} by substituting in eqns. (4.3) and (4.4).

4.2.4 Constant Density

There are several reacting systems (e.g. bioreactors, highly diluted systems with negligible temperature effect on density) which can be modelled assuming a uniform and constant density throughout the process. In such cases, the volume issue does not appear at all and eqns. (4.1) and (4.2) may be condensed into the expression:

$$\frac{dM_T}{dt} = \rho \frac{dV}{dt} = \sum_{i=1}^{ni} F_i^{in} - \sum_{i=1}^{no} F_i^{out} = 0, \tag{4.21}$$

The total mass in the volume (in each zone volume) is constant along the simulation and may be set as a constant parameter. Eqn. (4.8) becomes unnecessary and the system (4.3)–(4.7), (4.9) is the new reactor model in the unknowns M_i, X_i, U, T and F^{out}. It is easy to demonstrate that the incidence matrix does not include any minimal structural singular subset and no reduction index procedure is needed.

The model may be solved also by dropping the unknown F^{out} and one of the eqns. (4.3), because volume and total mass do not change and the mass balance will be always fulfilled.

4.3 Compressible Well-Mixed Reactor

Let us now consider a compressible well-mixed reactor (geometry is the same as in Fig. 4.1. The model is expressed by the following system of equations:

- mass balance:

$$\frac{dM_i}{dt} = F^{in} X_i^{in} - F^{out} X_i + V w_i \sum_{i=1}^{nr} \nu_{ir} R_r(X, T, P) \qquad i = 1, 2, \ldots, c \tag{4.22}$$

- energy balance:

$$\frac{dU}{dt} = F^{in} h^{in} - F^{out} h_i + Q \tag{4.23}$$

- total mass in the volume:

$$M_T = \sum_{i=1}^{c} M_i \tag{4.24}$$

- mass fractions:

$$X_i M_T = M_i \qquad i = 1, 2, \ldots, c \tag{4.25}$$

- internal energy:

$$U = M_t u(X, T, P) \tag{4.26}$$

- volume constraint:

$$V = M_T v(X, T, P) \tag{4.27}$$

- enthalpy computation:

$$h = h(X, T, P) \tag{4.28}$$

Apparently, this comprises $(2c + 5)$ DAEs in the $(2c + 5)$ unknowns M_i, X_i, M_T, U, T, h and P. However, P is not an unknown because its value is established from CFD calculations and thus, we have to drop one variable, i.e. one of the flowrate values F^{out} returned by the CFD code. From the gPROMS point of view, the pressure equation may be regarded as a complex function of composition, temperature and pressure itself (in the viscosity law)

$$P = P[X, T, \mu(X, T, P)]. \tag{4.29}$$

Let us consider the incidence matrix ($2c + 5$ equations for $3c + 6$ unknowns) of the system (4.22)–(4.29):

	$\dot{M}_i$	$\dot{U}$	M_i	U	X_i	M_T	T	F^{out}	h	P
Eqns. (4.22)	×				×		×	×		×
Eqn. (4.23)		×						×	×	
Eqn. (4.24)			×			×				
Eqns. (4.25)			×		×	×				
Eqn. (4.26)				×	×	×	×			×
Eqn. (4.27)					×	×	×			×
Eqn. (4.28)					×		×		×	×
Eqn. (4.29)					×		×			×

In this case no minimal structural singular subset can be identified. The DAE system is Index-1 and can be consistently initialised.

Incompressible fluid models may be solved also by excluding the pressure computation from CFD calculation and incorporate its calculation in the gPROMS model. According to this approach, all flowrates are estimated by the CFD code and the reactor model becomes the set of eqns. (4.22) – (4.28) in the unknowns M_i, X_i, U, M_T, T, P and h. If the hydrodynamics are not critical in determining the pressure in the equipment (e.g. the gas flow velocity approaches or exceeds the speed of sound), the suggested system will characterise the process well since mass balance is fulfilled and mass flowrates are all computed by CFD.

4.4 The CFD/Process Simulation Model

4.4.1 Cycles in a Network

In the previous sections we have demonstrated that, in general, modelling of incompressible well-mixed reactors give rise to some numerical issues which may be overcome by

1. adding one unknown to the model (mass flowrate F^{out});

2. reducing the DAE index.

As a result, for a *single zone* each internal zone model z should contain one unknown flowrate[3] F_z^{out}, i.e. one mass flowrate which is *not* computed by the CFD package. In other words we need to drop one equation

$$F_z^{out} = F_i(CFD) \qquad \forall z \in \mathcal{Z}_i$$

where $\mathcal{Z}_i$ is the set of internal zones.

In a *network of zones* as defined in chapter 3 any outlet from an internal zone is also an inlet to a neighbouring internal or environment zone. Thus, each unknown outlet flowrate and (unknown) inlet flowrate are linked by the set of equations:

$$F_z^{out} = F_{z'}^{in} \qquad \forall z \in \mathcal{Z}_i, \quad z' \in \mathcal{Z}_i \cup \mathcal{Z}_e \tag{4.30}$$

where $\mathcal{Z}_e$ is the set of environment zones.

Eqns. 4.30 may seem to overspecify the system, but that is not true: in fact, according

[3] Notation F_z^{out} is used to name the *only unknown* outlet with respect to zone z, although it may not be the only outlet in the zone.

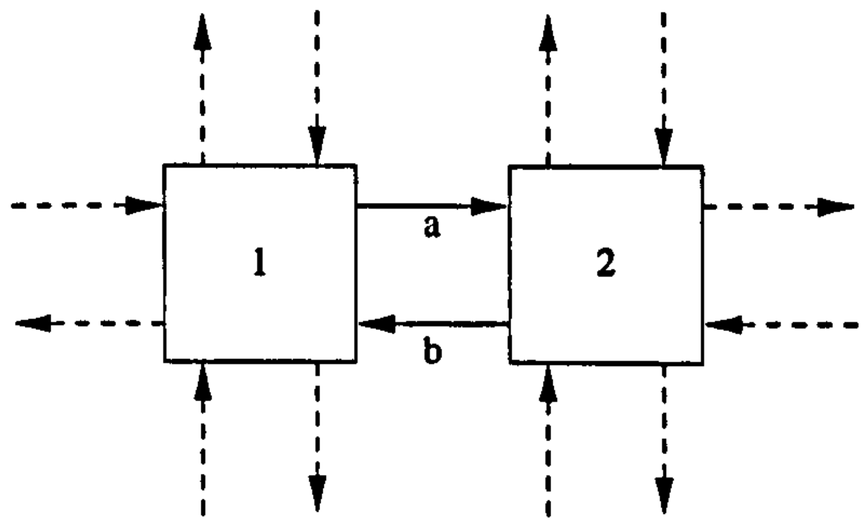


Figure 4.2: A simple example illustrating why *cycles* produce singular systems: flowrates a and b are both unknowns. The system is indeterminate.

to the general structure defined in chapter 3, flowrates are defined for each *interface* and not for each zone. As a result, if an outlet flowrate F_z^{out} is dropped for zone z, then $F_{z'}^{in}$ is also dropped for neighbouring zone z'. Thus, eqns. (4.30) are numerically correct. However, they may still lead to some physical issues in the system boundary conditions. Let us consider an environment interface between an internal zone z and an environment zone z'. Let us assume that the environment zone model represents an inlet flowrate towards the internal zone. By definition, the interface also takes into account an outlet flowrate, which should be set equal to 0. Nonetheless, that outlet flowrate may be chosen as the unknown flowrate for zone z and be given a non-zero value affecting the problem boundary conditions. To address this problem, we define a subset $\mathcal{Z}_e^* \subset \mathcal{Z}_e$ of environment zones where the outlet mass flowrate is *not* equal to zero. Eqns. (4.30) are reformulated as:

$$F_z^{out} = F_{z'}^{in} \qquad \forall z \in \mathcal{Z}_i, \quad \forall z' \in \mathcal{Z}_i \cup \mathcal{Z}_e^* \tag{4.30'}$$

The so-defined network model may still be badly posed. For instance, let us consider two neighbouring internal zones (Figure 4.2) belonging to a network of zones (connections to other zones are represented by dashed lines). Now let us suppose that mass flowrate a is chosen as unknown F_1^{out} and mass flowrate b is unknown F_2^{out}. Eqns. (4.30') state that

$$\begin{cases} F_1^{out} = F_2^{in} \\ F_2^{out} = F_1^{in} \end{cases}$$

Remembering also the network structure (see chapter 3), that gives:

$$\begin{cases} X_{1,i} = X_{2,i}^{in} \\ X_{2,i} = X_{1,i}^{in} \end{cases}$$

and substituting in eqns. (4.3′) for zones 1 and 2, we obtain the system:

$$\begin{cases} F_2^{out} X_{2,i} - F_1^{out} X_{1,i} = Q_1 \\ F_1^{out} X_{1,i} - F_2^{out} X_{2,i} = Q_2 \end{cases}$$

which is clearly singular (Q_1 and Q_2 represent all other terms of eqns. (4.3′); it can be demonstrated that $Q_1 = Q_2$). This is a well known problem which was studied in the analysis of network flow diagrams (e.g. Sargent and Westerberg (1964)) to avoid singularities in the solution approach.

Some definitions (Carré, 1979, West, 1996) about graphs and cycles in a graph may help explaining the problem:

Definition 2 *A* graph $\mathcal{G} = (\mathcal{X}, \mathcal{A})$ *consists of*

(i) a finite set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, *whose elements are called* nodes *(or* vertices*),*

(ii) a subset $\mathcal{A}$ *of the Cartesian product* $\mathcal{X} \times \mathcal{X}$, *the elements of which are called* arcs.

If $(x_i, x_j) \in \mathcal{A}$, *where* x_i, x_j *are nodes, then* x_i *and* x_j *are* adjacent.

Definition 3 *A* directed graph *or* digraph $\mathcal{G}$ *consists of a node set* $\mathcal{X}$ *and an arc set* $\mathcal{A}$, *where each arc is an* ordered pair *of nodes.*

Definition 4 *A* path *is a finite sequence of arcs of the form*

$$\mu = (x_{i_0}, x_{i_1}), (x_{i_1}, x_{i_2}), \ldots, (x_{i_{r-1}}, x_{i_r}),$$

i.e. a finite sequence of arcs in which the terminal node of each arc coincides with the initial node of the following arc [4].
A path whose endpoints *are distinct is said to be* open*; whereas a path whose endpoints coincide is called a* closed path *or* cycle.

Definition 5 *A* tree *is an acyclic graph* $\mathcal{G} = (\mathcal{X}, \mathcal{A})$ *in which one node* x_r *has no predecessors and every other node has exactly one predecessor.*

[4] *For some authors this is the definition of* walk, *while a* path *is a walk without any repetition of nodes.*

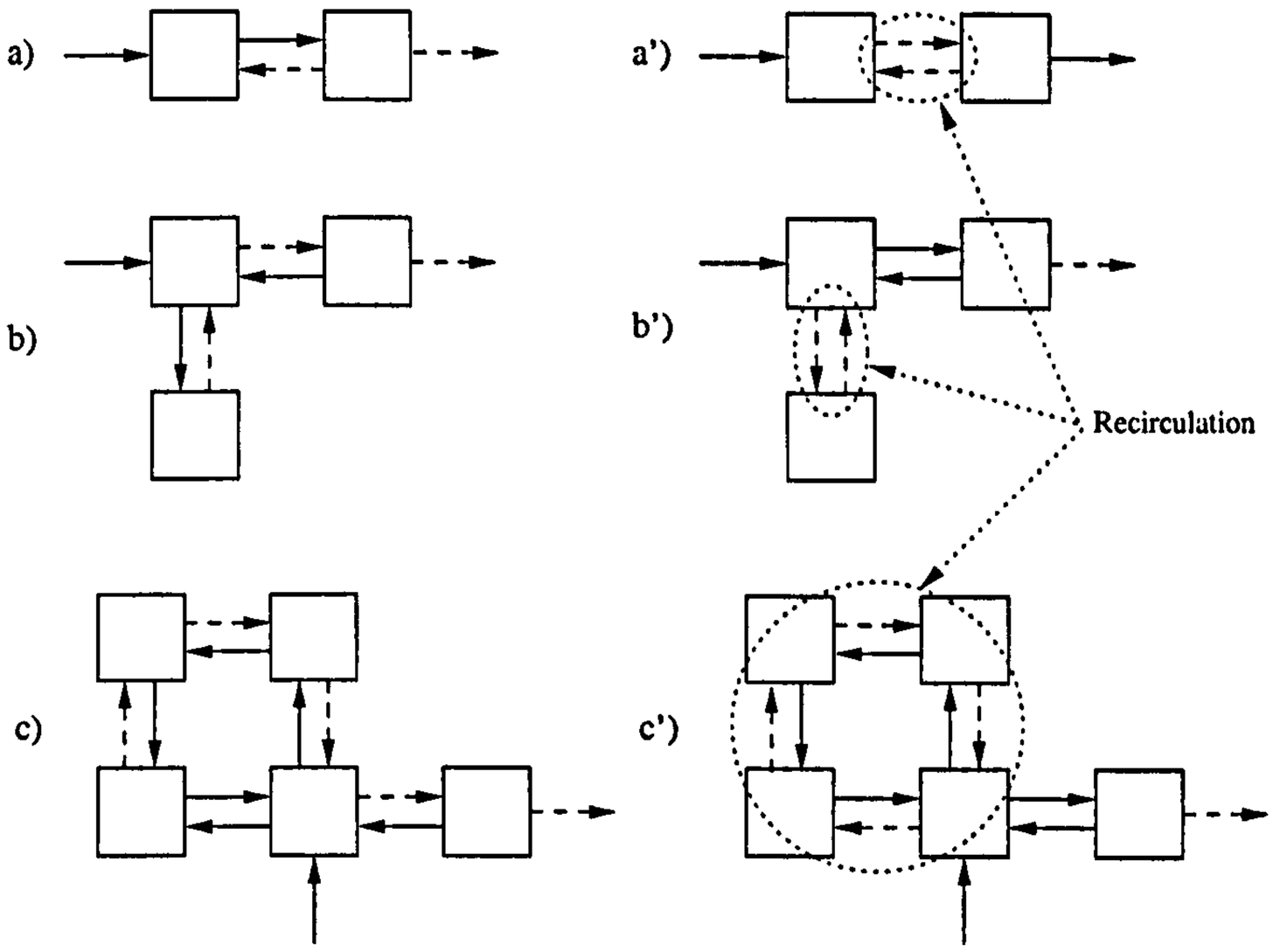


Figure 4.3: The left side (a,b,c) shows correct choices for unknown flowrates F^{out} (dashed arrows). On the right side (a',b',c') the same zone network present a feasible choice of unknown F_{out} leading to singular systems.

A zone network is a digraph $\mathcal{G}_N$ whose arcs are the flowrates between zones. The network whose nodes are the zones and whose arcs are the unknown flowrates F_z^{out} is a digraph $\mathcal{G}_F$ which is derived from the zone network ($\mathcal{G}_F \subseteq \mathcal{G}_N$).

According to well known graph theory (Carrè, 1979, West, 1996), it is easy to demonstrate that if there exists a cycle $\mathcal{S} \subseteq \mathcal{G}_F$, then the zone network model becomes singular.

Concluding, the issue is to identify which of the CFD flowrates in the zone network should be used as additional degrees of freedom and thus, to establish an acyclic digraph $\mathcal{G}_\mathcal{F}$ avoiding singularities in the solution matrix (Figure 4.3). First we will consider continuous processes, i.e. equipments presenting at least one outlet and one inlet. These are *open networks* because there are mass flowrates which are directed towards the environment.

Then batch processes will be taken into account. Batch processes have no environment zones representing inlets or outlets. They are *closed networks* and any digraph $\mathcal{G}_F$ contains at least one cycle.

4.4.2 An Algorithm for Open Networks

In an open network it is always possible to define $\mathcal{G}_F$ in such a way that no cycles are included. Let us consider the network of zones $\mathcal{G}_N$ and choose an internal zone z_i. Let

us define $\mathcal{N}_{z_e}$ as the set of internal zones which are neighbour to a generic environment zone z_e.

Since in $\mathcal{G}_N$ all zones have at least one neighbouring zone, it is certainly possible to build a path $\mathcal{P} \subseteq \mathcal{G}_F$ going from z_i to internal zone $z_j \in \mathcal{N}_{z_e} \mid z_e \in \mathcal{Z}_e^*$ which does not contain any cycle (*tree*). Now let us consider zones $z \in \{\mathcal{G}_N \setminus \mathcal{P}\}$. This set of zones is again an open network because at least one of them is neighbour to at least a zone $z' \in \mathcal{P}$ (otherwise $\mathcal{G}_N$ would not be a connected set), i.e. there is at least one mass flowrate F_z^{out} exiting the set of zones $\{\mathcal{G}_N \setminus \mathcal{P}\}$. The procedure can be iterated till all zones are included in a path. We define $\mathcal{G}_F \equiv \bigcup \mathcal{P}$. We point out that no cycle is obtained by the union of the paths $\mathcal{P}$ since they are built so as to allow only one-directional links between them, i.e. if $\mathcal{P}'$ and $\mathcal{P}''$ are neighbours, then there may exist a mass flowrate F^{out} from $\mathcal{P}'$ to $\mathcal{P}''$ (or from $\mathcal{P}''$ to $\mathcal{P}'$), but not vice-versa.

After demonstraing the existence of at least one acyclic $\mathcal{G}_F$, a procedure is needed capable of detecting the unknown flowrates F^{out} such as to result in an acyclic $\mathcal{G}_F$. We define an algorithm such that:

Given:

a) a set of internal zones $\mathcal{Z}_i$;

b) a set of external zones $\mathcal{Z}_e^*$;

c) a flow field $F_z(z, z')$, $\forall z \in \mathcal{Z}_i$, $\forall z' \in \mathcal{N}_z \subset (\mathcal{Z}_i \cup \mathcal{Z}_e^*)$, where $\mathcal{N}_z$ is the set of zones which are neighbour to z;

Derive the location of unknown output flowrates $F_z^*(z, z')$, $\forall z \in \mathcal{Z}_i$

The algorithm is structured as follows:

ALGORITHM FLOWRATE ($\mathcal{Z}_i$, $\mathcal{Z}_e^*$, F_z)

1. Initialization.
   a. $\forall z \in \mathcal{Z}_i$, $G(z) = 0$
   b. $\forall z \in \mathcal{Z}_e^*$, $G(z) = 1$
   c. $\forall z \in \mathcal{Z}_i, \forall z' \in \mathcal{N}_z$, $F_z^*(z, z') = 0$
2. Start setting a tree of unknown flowrates.
   a. $\forall z \in \mathcal{Z}_i$, $G^*(z) = 0$
   b. Select[5] $z \in \mathcal{Z}_i \mid G(z) = 0$

[5] After selecting an element e from a set $\mathcal{E}$, it results that $\mathcal{E} = \mathcal{E} \setminus \{e\}$

c. Select $z' \in \mathcal{N}_z$

d. $\mathcal{L} = \{(z, z')\}$

e. $F_z^*(z, z') = 1$

3. WHILE $\mathcal{L} \neq \{\}$ DO

a. Select (z, z') from $\mathcal{L}$

b. $G^*(z) = 1$

c. IF $G^*(z') = 0$ THEN

I. $G(z) = 1$

II. IF $G(z') = 0$ THEN

i. Select $z'' \in \mathcal{N}_{z'}$

ii. $\mathcal{L} = \{(z, z')\} \cup \mathcal{L}$

iii. $\mathcal{L} = \{(z', z'')\} \cup \mathcal{L}$

iv. $F_z^*(z', z'') = 1$

v. $G(z') = 1$

END IF

ELSE

I. $F_z^*(z, z') = 0$

II. $G(z) = 0$

III. IF $\mathcal{N}_{z'} \neq \{\}$ THEN

$G^*(z) = 0$

END IF

END IF

END DO

4. Go to 2.

Figure 4.4 illustrates the algorithm by means of a simple example considering the following steps (note that only environment Zones *ii* and *iii* belong to $\mathcal{Z}_e^*$):

a) initially, internal zones in $\mathcal{G}_N$ are numbered (1 – 6);

b) Zone 1 is selected and an initial sequence of zones (1-4-6) and flowrates ($\overrightarrow{14}$-$\overrightarrow{46}$-$\overrightarrow{65}$) is obtained;

c) Zone 5 is added to the sequence; first-choice flowrate $\overrightarrow{51}$ is discharged since it leads to already listed Zone 1;

d) second outlet flowrate $\overrightarrow{56}$ is considered, but again it must be discharged since it leads to already listed Zone 6; since no other outlets are available, Zone 5 is discharged from the list as well as the outlet $\overrightarrow{65}$;

e) outlet $\overrightarrow{6iii}$ from Zone 6 is selected: it is an external flowrate leading to environment zone *iii* (set $\mathcal{Z}_e^*$); a suitable path $\mathcal{P}_1$ is found (red flowrates and zones);

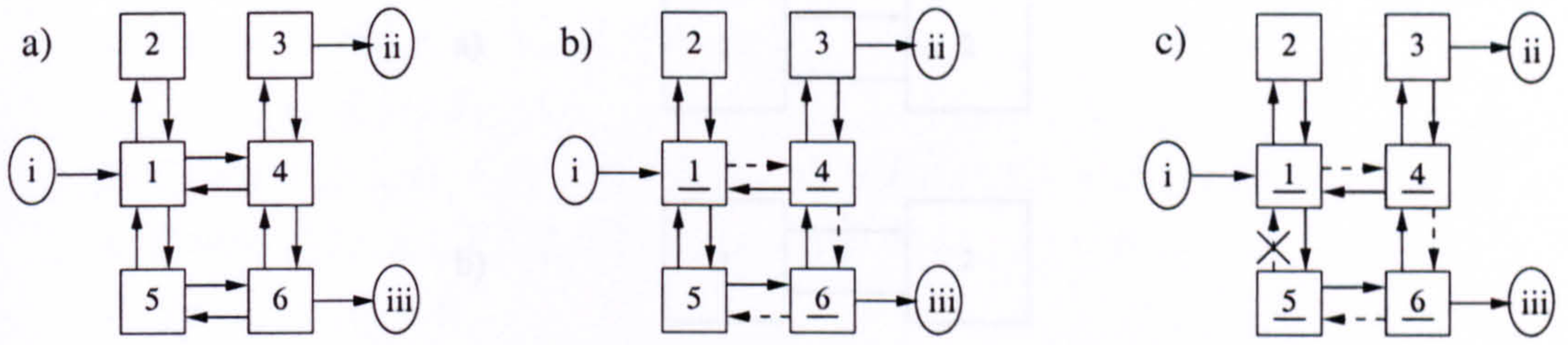


Figure 4.4: *Flowrate Algorithm*: one example.

f) Zone 2 is considered: the only available outlet $\overrightarrow{21}$ points to Zone 1 which already belongs to $\mathcal{P}_1$; Zone 2 and outlet $\overrightarrow{21}$ constitutes the second path $\mathcal{P}_2$;

g) Zone 3 is selected; first-choice outlet $\overrightarrow{3ii}$ leads to environment Zone ii (set $\mathcal{Z}_e^*$): path $\mathcal{P}_3$ is obtained;

h) eventually, Zone 5 is selected; being the last one, any of its outlets would lead either to an internal zone belonging to an existing path or to a outlet-type environment zone; in the specific case, outlet $\overrightarrow{51}$ is chosen and path $\mathcal{P}_4$ is obtained. And thus, $\mathcal{G}_F \equiv \{\mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3 \cup \mathcal{P}_4\}$.

4.4.3 Closed Networks

Closed networks cannot be reduced to a set of acyclic paths. As it is, algorithm FLOWRATE cannot be applied because no environment zone will be found.

Nonetheless, a solution can be found if we introduce a new flowrate by splitting one of the flowrates in the network. For instance, let us consider Figure 4.5a representing the simplest case of closed network $\mathcal{G}_N$. If we apply the solution method developed for incompressible fluids, both flowrates should be assumed to be unknown and the

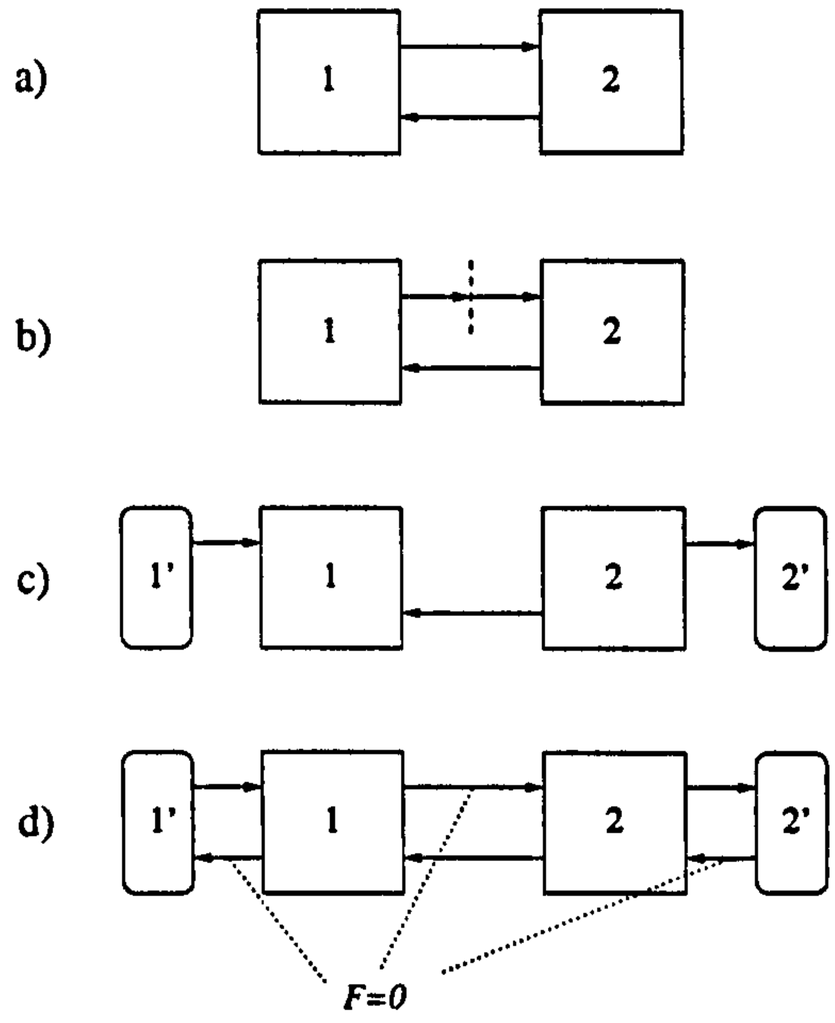


Figure 4.5: A solution for batch systems

DAE system is singular. Let us split flowrate $\overrightarrow{12}$ (4.5b) into two flowrates and assume that these newly created flowrates are connected to two "virtual" environment zones $1'$ and $2'$ (4.5c). Concluding, flowrate $\overrightarrow{12}$ is removed and transformed into two flowrates $\overrightarrow{1'2}$ and $\overrightarrow{22'}$. The CFD-computed value of $\overrightarrow{12}$ is given to both $\overrightarrow{1'2}$ and $\overrightarrow{22'}$. Since any interface is defined as bi-directional (§ 3.2), flowrates $\overrightarrow{12}$, $\overrightarrow{11'}$ and $\overrightarrow{2'2}$ are added to the network. Their flowrate values are set equal to zero (4.5d). The initial closed network $\mathcal{G}_N$ is turned into a open network $\mathcal{G}'_N$ which can be handled by algorithm FLOWRATE.

The procedure can be applied to any closed network. For a generic closed network, it can be summarised as follows:

1. select two neighbouring zones z_i and z_j;

2. define two environment zones $z_{i'}$ and $z_{j'}$;

3. remove flowrate $\overrightarrow{ij}$ by splitting it into inlet $\overrightarrow{j'j}$ and outlet $\overrightarrow{ii'}$;

4. assign flowrate $\overrightarrow{ij}$ value to flowrates $\overrightarrow{j'j}$ and $\overrightarrow{ii'}$;

5. define new zero-flowrates $\overrightarrow{ij}$, $\overrightarrow{jj'}$ and $\overrightarrow{i'i}$.

The newly defined network $\mathcal{G}'_N$ is an open network to which algorithm FLOWRATE may be applied. The procedure is implemented in the following algorithm:

ALGORITHM OPEN_CLOSE ($\mathcal{Z}_i$, $\mathcal{Z}_e$, F_z)

1. IF $\mathcal{Z}_e = \{\}$ THEN

a. Select $z \in \mathcal{Z}_i$, $z' \in \mathcal{N}_z$
b. Define $z_e, z'_e \mid z_e \in \mathcal{N}_z, z'_e \in \mathcal{N}_{z'}$;
c. $\mathcal{Z}_e = \{z_e, z'_e\} \cup \mathcal{Z}_e$
d. Define $F_z(z_e, z), F_z(z', z'_e) \mid F_z(z_e, z) = F_z(z', z'_e) = F_z(z, z')$
e. Define $F_z(z, z_e), F_z(z'_e, z') \mid F_z(z, z_e) = F_z(z'_e, z') = 0$
f. Set $F_z(z', z) = 0$
g. Call FLOWRATE ($\mathcal{Z}_i$, $\mathcal{Z}_e$, F_z)

2. ELSE Call FLOWRATE ($\mathcal{Z}_i$, $\mathcal{Z}_e$, F_z)

3. Stop.

Algorithm OPEN_CLOSE checks if $\mathcal{G}_N$ is open or closed; if $\mathcal{G}_N$ is a closed network, then it is turned into an open network $\mathcal{G}'_N$. Eventually, algorithm FLOWRATE is applied. Therefore, any kind of network can be handled by the suggested zone modelling architecture.

4.5 The Network Interface

The network interface as described in § 3.5.1 and § 3.5.2 is not sufficient to deal with an `InternalZone` model for incompressible fluids as discussed in this chapter. In fact, not all mass flowrates `F` should be assigned a value from the foreign object `ZoneNet` because one mass flowrate for each internal zone one is treated as an unknown to be computed by gPROMS.

The problem is overcome by adding the additional parameters

- `NoXIIInterface`

- `NoXIEInterface`

to the `ZoneNetwork` model.

Parameter `NoXIIInterface` represents the number of internal interfaces which contain an unknown flowrate. Parameter `NoXIEInterface` is added only for clarity reasons. It is not strictly necessary because the number of unknown flowrates is equal to the number of internal zones and, accordingly, the number of environment interfaces to which the unknown flowrate is referred to is given by `NoIzone` minus `NoXIIInterface`[6].

Furthermore we order the zone network in such a way that that unknown flowrate is always an outlet `Fout` with respect to a *right* zone according to the definition given

[6] Note that there *cannot* be two unknown flowrates related to the same interface because that would constitute a cycle.

in § 3.5.1. The setting of parameter `NoXIIInterface` as well as all the other network parameters is retrieved through foreign object `gCFD`.

Table 4.1 outlines the changes made to the `ZoneNetwork` model previously described in Tables 3.3 and 3.4.

Table 4.1: New `ZoneNetwork` model.

```
MODEL ZoneNetwork

PARAMETER
 ZoneNet AS FOREIGN_OBJECT "gCFD"

 # Numbers of Internal and Environment Zones
 ...

 # Internal-Internal Interface
 NoXIIInterface AS INTEGER
 ...

 # Internal-Environment Interfaces
 NoXIEInterface AS INTEGER
 ...

 # Other Parameters
 ...

UNIT
 ...

VARIABLE
 ...

SET
 NoXIIInterface := ZoneNet.NumberOfXInternalInterfaces()
 NoXIEInterface := NoIZone-NoXIIInterface
 ...

EQUATION
 ...
 F(,1) = ZoneNet.MassFlowratesLeftToRight
                    (GLobalFluidProperty,CFDParameter) ;
 F(,2) = ZoneNet.MassFlowratesRightToLeft
                    (GLobalFluidProperty,CFDParameter) ;

# Internal-Internal Interfaces
FOR i := 1 TO NoIIInterface-NoXIIInterface DO
 # Assign computed mass flowrates to individual interfaces
 F(i,1) = IZone(IILeftZone(i)).Fout(IILeftPort(i))
                  = IZone(IIRightZone(i)).Fin(IIRightPort(i)) ;
 F(i,2) = IZone(IILeftZone(i)).Fin(IILeftPort(i))
                  = IZone(IIRightZone(i)).Fout(IIRightPort(i)) ;
END

FOR i := NoIIInterface-NoXIIInterface TO NoIIInterface DO
```

```
 # Assign computed mass flowrates to individual interfaces
 F(i,1) = IZone(IILeftZone(i)).Fout(IILeftPort(i))
                = IZone(IIRightZone(i)).Fin(IIRightPort(i)) ;
 IZone(IILeftZone(i)).Fin(IILeftPort(i))
                = IZone(IIRightZone(i)).Fout(IIRightPort(i)) ;
END

# Internal-Environment Interfaces
FOR i := 1 TO NoIEInterface-NoXIEInterface DO
 # Assign computed mass flowrates to individual interfaces
 F(i+NoIIInterface,1)
                = IZone(IELeftZone(i)).Fout(IELeftPort(i))
                = EZone(IERightZone(i)).Fin(IERightPort(i)) ;
 F(i+NoIIInterface,2)
                = IZone(IELeftZone(i)).Fin(IELeftPort(i))
                = EZone(IERightZone(i)).Fout(IERightPort(i)) ;
END

FOR i := NoIEInterface-NoXIEInterface TO NoIEInterface DO
 # Assign computed mass flowrates to individual interfaces
 F(i+NoIIInterface,1) = IZone(IELeftZone(i)).Fout(IELeftPort(i))
                = EZone(IERightZone(i)).Fin(IERightPort(i));
 IZone(IELeftZone(i)).Fin(IELeftPort(i))
                = EZone(IERightZone(i)).Fout(IERightPort(i)) ;
END

 ...

END # model ZoneNetwork
```

4.6 The Volume Issue: a Physical Point of View

The previous sections explored how the "volume issue" may be overcome from a structural and numerical point of view by introducing additional variables in the simulation. The only "consistency" requirement for the overall solution we obtain is that all mass is accounted for. This depends on writing correctly the simulation zone and zone network models. The previous sections provide a solution to that. As seen from the simulation side, what the CFD package does is to externally specify a piecewise constant variation of the inter-zone mass flowrate over time. The overall solution obtained by gPROMS will *always* be consistent irrespective of how these flowrates have been calculated, how accurate they are, or the number/duration of the time pieces (i.e. the frequency of updating the flowrates). Even if we replace the CFD computation of the flowrates by a random number generator that gPROMS calls at a certain (also random) time points, the results of the gPROMS simulation will still conserve mass at all times. There-

fore, the question is: *although numerically consistent, are CFD solutions physically consistent with the gPROMS model?*

Steady-state CFD solution values satisfy eqn. (4.1). According to the zone model for incompressible fluids, not all flowrates computed by the CFD package are passed to the gPROMS zone network model, since one F^{out} stream is an unknown variable for each zone. Nonetheless, it is obvious that only the F^{out} set consistent to the CFD calculations is the one satisfying eqn. (4.1).

Let us consider the simulation model for a very simple case, i.e. a non reactive system consisting of an ideal liquid-phase mixture. Let us write the general correlation for a zone j:

$$v_j = \sum_{i=1}^{c} X_{ij} v_i^0, \tag{4.31}$$

where v_i^0 is the specific volume of pure component i.
If we take eqn. (4.31) and multiply by M_T, we obtain:

$$M_{T,j} v_j = \sum_{i=1}^{c} X_{ij} M_{T,j} v_i^0 \tag{4.32}$$

From eqn. (4.8), the left hand side is V_j. Using eqn. (4.6) to simplify the right hand side, we obtain:

$$V_j = \sum_{i=1}^{c} M_{ij} v_i^0. \tag{4.33}$$

Now differentiating this with respect to time, and since V_j is constant, we get :

$$0 = \sum_{i=1}^{c} v_i^0 \frac{dM_{ij}}{dt}, \tag{4.34}$$

which, using eqn. (4.3′), leads to:

$$\sum_{i=1}^{c} \sum_{k} F_{jk}^{in} X_{ijk}^{in} v_i^0 = \sum_{i=1}^{c} X_{ij} v_i^0 \sum_{k} F_{jk}^{out} \tag{4.35}$$

Now, according to eqn. (4.31), the expression $\sum_{i=1}^{c} X_{ik} v_i^0$ is simply the specific volume of the contents of zone k, i.e. v_k. Similarly, $\sum_{i=1}^{c} X_{ij} v_i^0$ is just v_j. Therefore, the above

can be written as:

$$\sum_k F_{jk}^{in} v_k = v_j \sum_k F_{jk}^{out}. \tag{4.36}$$

If we compare eqn. (4.36) to eqn. (4.1)

$$\sum_k F_{jk}^{in} = \sum_k F_{jk}^{out}, \tag{4.1}$$

we realise that the two expressions are compatible only if:

$$\sum_k F_{jk}^{in} (v_k - v_j) = 0, \tag{4.37}$$

which, in dynamic conditions, may *not* be true (e.g. because of composition changes or thermal effects).

Whenever dynamic flow conditions affecting local density apply, eqn. (4.1) is not valid. Nonetheless, this is not surprising: one fundamental assumption in our integration approach is that at any time t the hydrodynamics may be modelled as steady-state. This is true when the process is actually steady-state, and it is a good approximation when the process is dynamic but phenomena described by the process simulation tool are governed by a larger time scale.

The fact that computational reasons impede a continuous CFD update of the fluid flow pattern also has to be considered. In the solution approach suggested in this chapter, hydrodynamic changes are borne by the set of unknown flowrates in the network. However, the specified flowrates are the result of a previous steady-state simulation and, as such, they may be physically incorrect. Nonetheless, it should be pointed out that, in many practical cases, the difference between our solution and a "correct" approach is so small that the reliability of the simulation is still intact from an engineering point of view. Furthermore, some of the techniques which will be described in chapter 7 aim at tackling this issue by detecting when a CFD update is required.

Therefore, the zone network approach defined for the description of incompressible fluids is certainly "correct" for steady state processes. In general, it is an approximation of a physically consistent solution when a dynamic process is being modelled. However, this a sensible approximation whenever the time scale of the hydrodynamics is much faster than the time scale of the phenomena described in the process simulation zone model.

4.7 Non-Constant Volume: Some Ideas

We have demonstrated that closed networks (batch processes) may be treated as open ones and, as a result, we are always able to obtain a solution. However, is this solution meaningful? The solution is certainly realistic when the total volume (i.e. average density in the domain) remains approximately constant throughout the process. On the contrary, if volume is not constant, the solution method will obtain a solution by varying the total mass in the domain. Nonetheless, many important processes present significant changes in the total volume. If these changes are small a potential solution may be found within present capabilities of commercial CFD codes.

For each zone volume, we introduce an additional algebraic state variable $V_j = M_{T,j} v_j$ and we consider all flowrates specified by the CFD simulation. The new zone network model is integrated between t_i and t_{i+1}, assuming that the next CFD calculation is required at time t_{i+1}. From the resulting zone volumes $V_j\left(t_{i+1}^-\right)$, the new total CFD volume:

$$V^{\mathrm{CFD}}\left(t_{i+1}\right) = \sum_{j=1}^{nz} V_j\left(t_{i+1}^-\right) \tag{4.38}$$

is calculated. Of course $V^{\mathrm{CFD}}\left(t_{i+1}\right)$ will in general be different from the initial total CFD volume $V^{\mathrm{CFD}}(0)$.

Based on the assumption that the CFD zone volume fractions

$$\theta_j = \frac{V_j^{\mathrm{CFD}}}{V^{\mathrm{CFD}}}$$

are constant, the new CFD volumes $V_j^{\mathrm{CFD}}\left(t_{i+1}\right)$ are given by

$$V_j^{\mathrm{CFD}}\left(t_{i+1}\right) = \theta_j V^{\mathrm{CFD}}\left(t_{i+1}\right). \tag{4.39}$$

For each internal zone j, we now define the volume difference

$$\Delta V_j\left(t_{j+1}\right) = V_j\left(t_{i+1}^-\right) - V_j^{\mathrm{CFD}}\left(t_{i+1}\right). \tag{4.40}$$

A positive $\Delta V_j\left(t_{j+1}\right)$ means that the zone j has grown at expense of other internal zones which have negative volume differences. We can re-initialize the system, e.g., by "pooling" all positive volume differences from the zones which have grown and the using this pool to fill up the zones which have shrunk. Clearly, this will change the

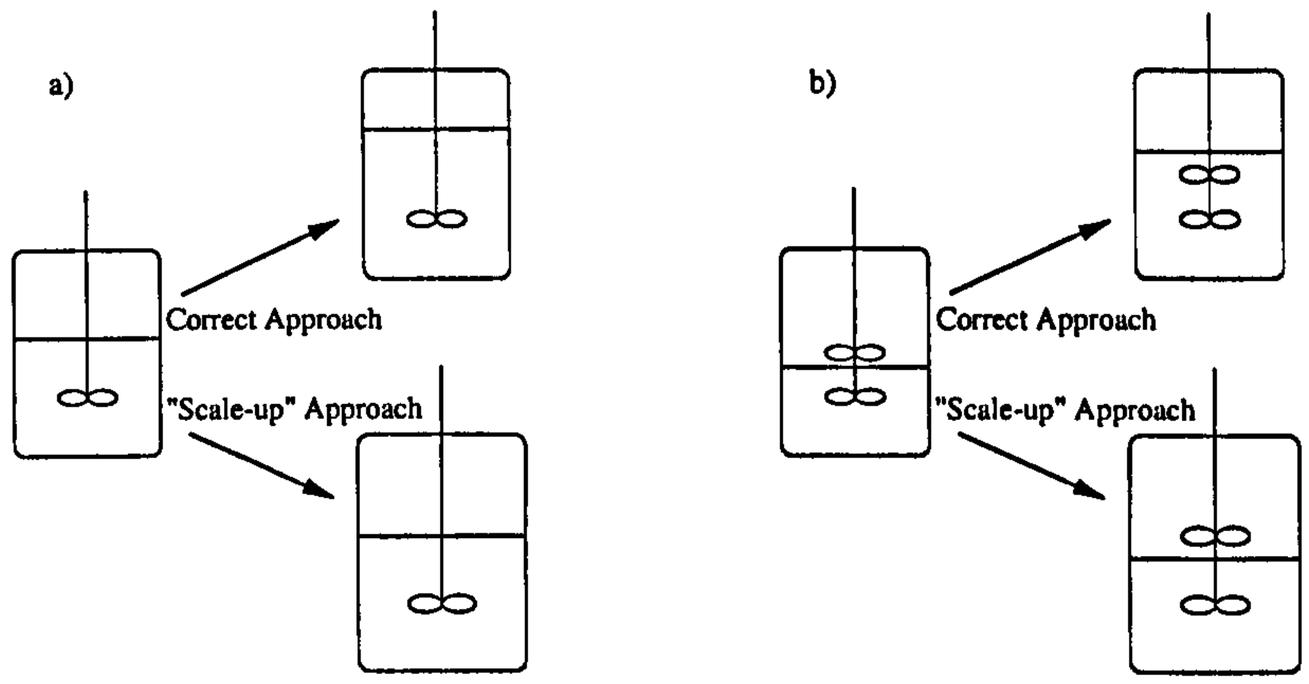


Figure 4.6: *Scale-up/down* method.

densities of the zones which receive material from the pool, i.e.,

$$v_j\left(t_{i+1}^{+}\right) \neq v_j\left(t_{i+1}^{-}\right)$$

whenever $\Delta V_j\left(t_{j+1}\right) < 0$.

After scaling [7] all CFD volume cells—and hence the complete system—by the factor

$$\gamma\left(t_{i+1}\right) = \frac{V^{\mathrm{CFD}}\left(t_{i+1}\right)}{V^{\mathrm{CFD}}(0)}, \tag{4.41}$$

the next CFD calculation is performed based on the new set of specific volumes $v_j\left(t_{i+1}^{+}\right)$. This provides the new mass flowrates $F_{jk,in}\left(t_{i+1}^{+}\right)$ and $F_{jk,out}\left(t_{i+1}^{+}\right)$ to be used during the next gPROMS simulation.

It is still an open question for which kind of practical processes this solution approximation may be a reasonable. The method may apply to batch systems where the average density varies and we need to conserve the total mass in the process. The main problem stays in the fact that such a method scales up/down the whole geometry. This is acceptable if the volume changes are small and fluid dynamics do not vary (Fig. 4.6a), but cannot be adopted when volume changes are great and/or fluid dynamics mutate (Fig. 4.6b). Thus, the main assumption is that volume changes do *not* affect fluid dynamics. Nonetheless, this method is able to take into account volume variation into batch processes without affecting the mass balance[8].

However, it is clear that no proper solution may be obtained until a suitable CFD representation is found for variable volume processes by means of dynamic gridding or

[7] This is possible in many CFD packages, such as Fluent or CFX.

[8] The *close network* method described in § 4.4.1 will be able to return some numerical results but variations in the total volume will be compensate by modifying the total mass in the system.

other techniques.

4.8 Key Results

The main achievements illustrated in this chapter are:

- The analysis of *incompressible fluid modelling* according to the general structure developed in chapter 3. In particular, it was demonstrated that
  - ▷ the zone network model as described in chapter 3 will not meet the overall mass balance, when incompressible fluids are considered;
  - ▷ incompressible fluid models produce DAE systems of index 2 or higher, unless suitably written;
  - ▷ a reduction procedure to a well posed Index 1 model was presented for a general class of non-ideal non-isothermal reactors.
- The development of a procedure to treat incompressible fluids within the zone model architecture. This is achieved by ignoring one of the flowrate values returned by the CFD code for each zone, instead treating it as an unknown variable to be computed from the process simulation model equations. The design of the zone network model was modified to consider incompressible fluids. It was shown that this approach leads to a correct numerical solution or a good approximation when the process is
  - ▷ at steady-state;
  - ▷ dynamic, but the process simulation tool describes phenomena governed by a larger time scale than the hydrodynamics.

  The approach is not correct when the hydrodynamics and the process simulation phenomena exhibit similar dynamics.
- The definition of *open* and *closed* networks and the design of a practical procedure to generate a suitable zone network model in the case of incompressible fluids. The procedure can be applied to closed network only if there are no significant changes in the total volume.

Chapter 5

Zoning Methods

This chapter is dedicated to the implementation of the methods designed to define the zone network according to the interface described in chapters 3 and 4. First, we will discuss how zones may be easily set up manually and how information is treated by the interface. Second, we will consider an approach to *automatically* achieve an effective zoning which is appropriate for the simulation of fluid flow behaviour.

5.1 Zones from CFD grids

A zone is a collection of neighbouring cells. Furthermore, we impose that a zone is a set of connected cells: if the zone contains more than one cell, each cell must have at least one face which is common to a different cell belonging to the same zone. Although this assumption is unnecessary from a numerical point of view since the interface as defined in chapter 3 can also handle disconnected zones, a zone should represent a homogeneous, well-mixed region and, as such, it must be a connected set of cells.

The zone volume V is the sum of the volumes V_c of the cells constituting the zone. Two neighbouring zones (named **A** and **B**) hold a common interface which has the area A, defined as the sum of the cell faces $A_{cc'}$ which are shared by all cells c belonging to zone **A** and c' to zone **B**. Information regarding cell volumes and areas can be easily retrieved by CFD grid files and, to the best of our knowledge, all commercial CFD packages make those data available to the user.

The mass flux through the interface is split into two mass flowrates, one for each direction, to represent the fact that mass exchange is in general bi-directional. Figure 5.1 illustrates the nomenclature. Two mass flowrates are associated to the interface between two neighbouring zones; the mass flowrate from **A** to **B** is F^+ and the mass flowrate from **B** to **A** is F^-. They are computed by aggregating the mass flowrates of

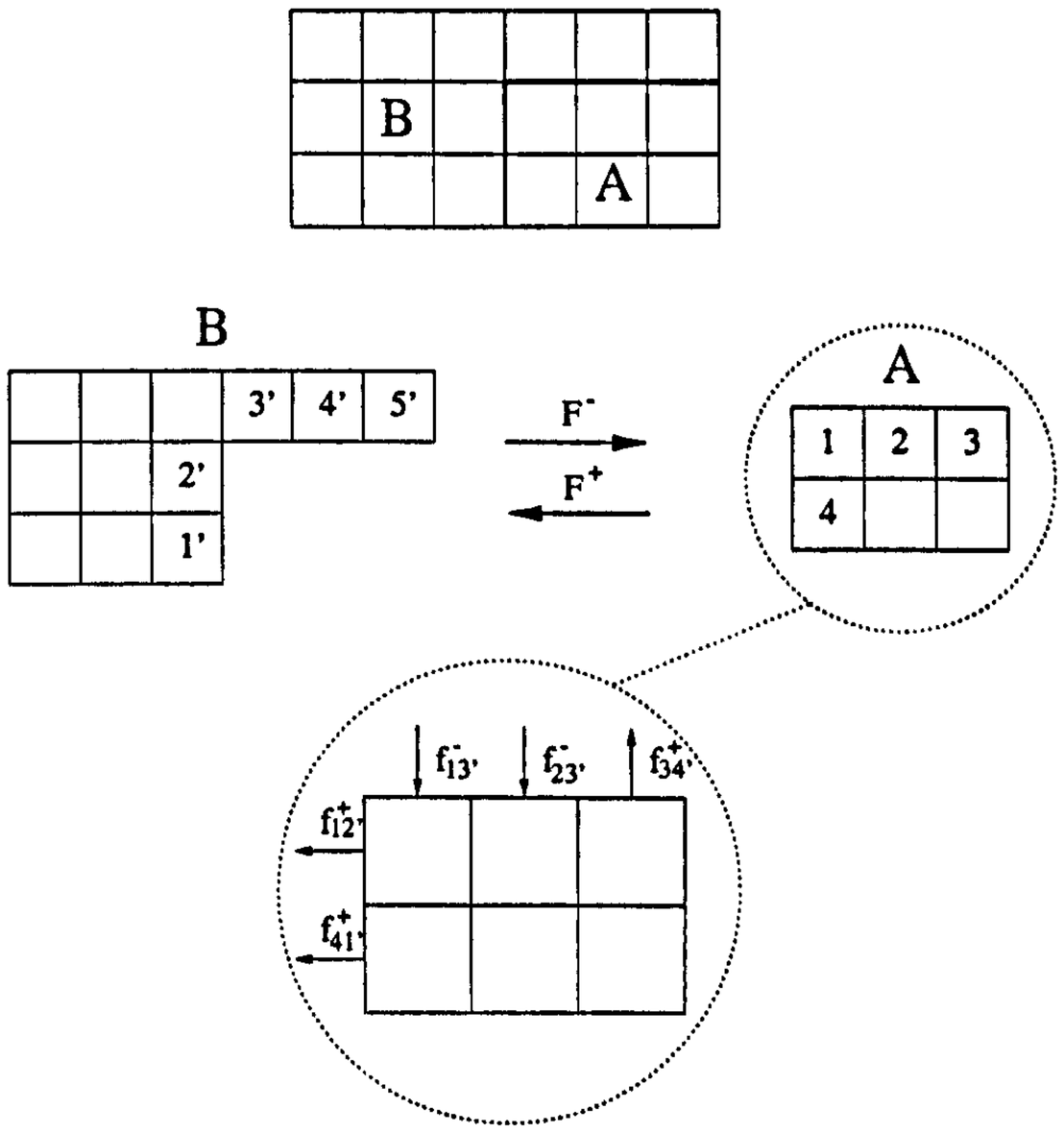


Figure 5.1: Flowrates between zones and cells.

every single cell face[1] which belongs to the interface between zone **A** and **B**. We call $f^+_{cc'}$, the mass flowrates *from* the generic cell c in zone **A** *to* cell $c' \in \mathcal{N}_c$ (where $\mathcal{N}_c$ is the set containing all the cells neighbouring cell c) in zone **B** (i.e. zone **A** outlets) and, vice-versa, $f^-_{cc'}$ the mass flowrates *to* cell c of **A** *from* cell c' of **B** (i.e. zone **A** inlets). The two mass flowrates F^+ and F^- between the two zones are then defined as: $F^+ = \sum f^+_{cc'}$ and $F^+ = \sum f^-_{cc'}$. Note that for each cell face the flow is mono-directional, i.e. there is *only* one flowrate, either $f^+_{cc'}$ or $f^-_{cc'}$. Some CFD packages make all $f_{cc'}$ directly available to the user. Otherwise, the velocity field resulting from a CFD computation may be used to calculate the mass flowrates through every cell face according to:

$$\begin{aligned} f^+_{cc'} &= \rho_c v_{cc'}/A_{cc'} && \text{if } v_{cc'} > 0 \\ f^-_{cc'} &= \rho_{c'}(-v_{cc'})/A_{cc'} && \text{if } v_{cc'} < 0, \end{aligned}$$

where ρ_c is density within cell c and $v_{cc'}$ is the velocity component normal to surface $A_{cc'}$ (positive if flows goes from c to c'). The computation of $v_{cc'}$ may be rather complex since the velocity vector $\mathbf{v}_c$ is computed at the cell centre. Interpolation techniques are required to compute velocity on the cell faces, based on centre values in neighbouring

[1] For a single cell there may be more than one face on the interface.

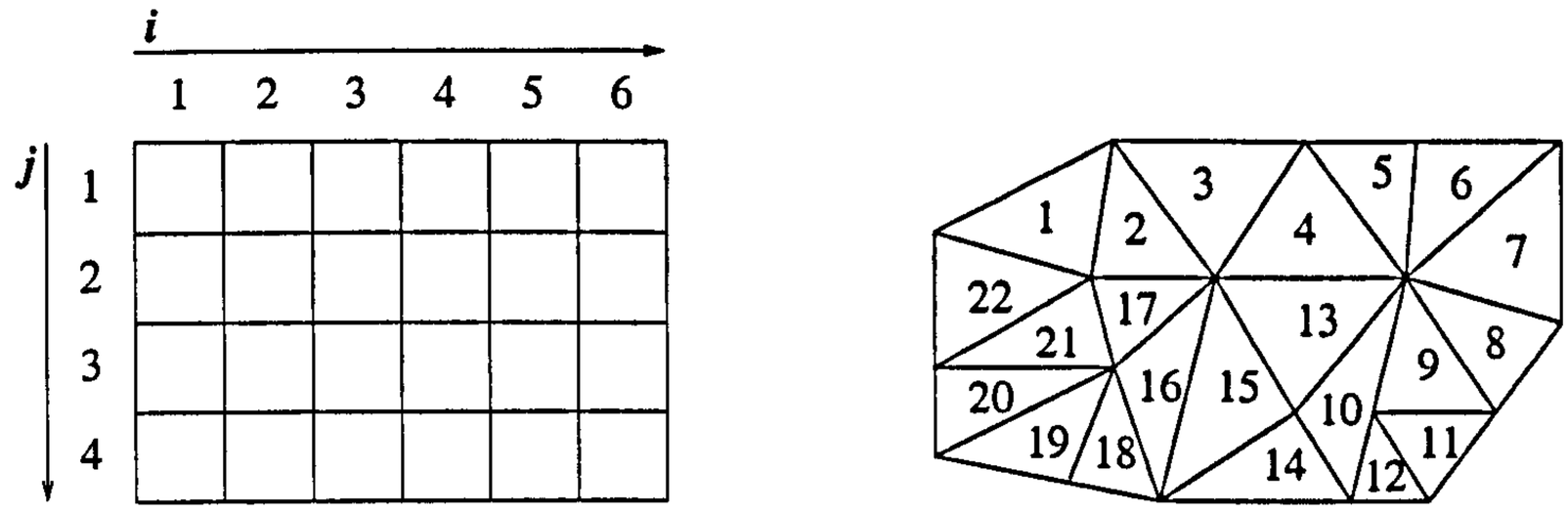


Figure 5.2: Numbering in structured and unstructured grids.

cells. Several techniques may be adopted. The simplest method is the linear formula:

$$v_{cc'} = \frac{v_c \Delta x_c + v_{c'} \Delta x_{c'}}{\Delta x_c + \Delta x_{c'}},$$

where $v_c, v_{c'}$ are the components of vectors $\mathbf{v}_c$ and $\mathbf{v}_{c'}$ normal to face $A_{cc'}$ and $\Delta x_c, \Delta x_{c'}$ are the distances from the centres of neighbouring cells c and c' to face $A_{cc'}$. Components v_c and $v_{c'}$ are computed considering the grid configuration.

Information regarding external inlets/outlets and heat fluxes through conducting walls are stored as boundary conditions in the computational cells of the grid. Most CFD packages offer the possibility to easily spot these data and make them available to the user[2]. Thus CFD data regarding environment zones are in general easy to obtain.

5.1.1 Structured and Unstructured Grids

The procedures to define zones in a CFD mesh were applied to the CFD simulator Fluent 4.5. Fluent 4.5 is a structured grid solver. Structured grids present a simple topology: cells can be easily identified by means of i, j, k indices according to each main axis (cartesian or cylindrical). With unstructured grids this would be more difficult since cells are progressively numbered without any relation to the coordinate axes. Figure 5.2 illustrates the difference by means of a 2D example. Two indices identify any cell in the structured grid (left) and we know that cell (i, j) follows cells $(i-1, j)$ and $(i, j-1)$ along axes i and j, respectively. Unstructured grids (right) do not work in that way and the relation between cells i and $i+1$ cannot be retrieved by just observing geometry and coordinate axes.

On the other hand, as far as flux calculations and averaging processes are concerned, there is no difference in processing results within a structured solver or an unstructured

[2]Besides computational cells, most CFD packages define other types of cells to store special boundary conditions such as: inlets, outlets, symmetric regions, cyclic regions, walls, conducting walls, etc.

one. Thus, all definitions and concepts in § 5.1 are independent of the type of CFD solver.

A structured solver was chosen here because of this topological simplicity. Definition of zones and their spatial relation is easier within a structured grid. Nonetheless, we point out that the generality of our results and the zoning algorithms which are proposed in this chapter is not affected by the type of meshing in the CFD code.

5.2 Manual Zone Declaration and Implementation

The definition of zones within a CFD domain is a very important issue. Some CFD packages contain some intermediate "units" (e.g. *blocks*, *patches*) which are similar to the concept of zones. These units are defined through a pre-processor as building blocks during the definition of the geometry and prior to the meshing procedure, and are then recognised by the main software. However, many details concerning the zone definition depend on physical phenomena which can be observed and analysed only after a CFD simulation. These aspects cannot be estimated from the geometry of the equipment or from grid considerations only. Furthermore, the designed interface should be independent of the specific CFD package which is adopted to perform the flow calculations. Thereby, the object `gCFD` (§ 3.5.1) will refer not to the CFD code itself but to a *topology file* containing all the necessary information. First of all, we specify:

1. the CFD package, i.e. the name of specific software which is adopted to perform CFD calculations;

2. the name of the CFD file containing the grid and other required inputs for the problem under examination[3].

After zones have been set, it is assumed that internal zones may be constituted of elementary units or *blocks*. Each block typically represents a very simple geometry (parallelepiped or a list of cells) which is used to set up zones. Besides these elementary blocks, a special block *domain*, representing the entire computational grid, is also defined. Zones are then built up by means of simple set operations involving blocks. For instance, in Figure 5.3 Zones 1, 2 and 3 are defined by means of Blocks *i*, *ii* and *iii* and *domain* as follows:

- Zone 1 = *i*

[3] In the case of the Fluent package this name indicates both the grid and the necessary information to start a CFD simulation (`.CAS` file) and the results of a previous simulation (`.DAT` file).

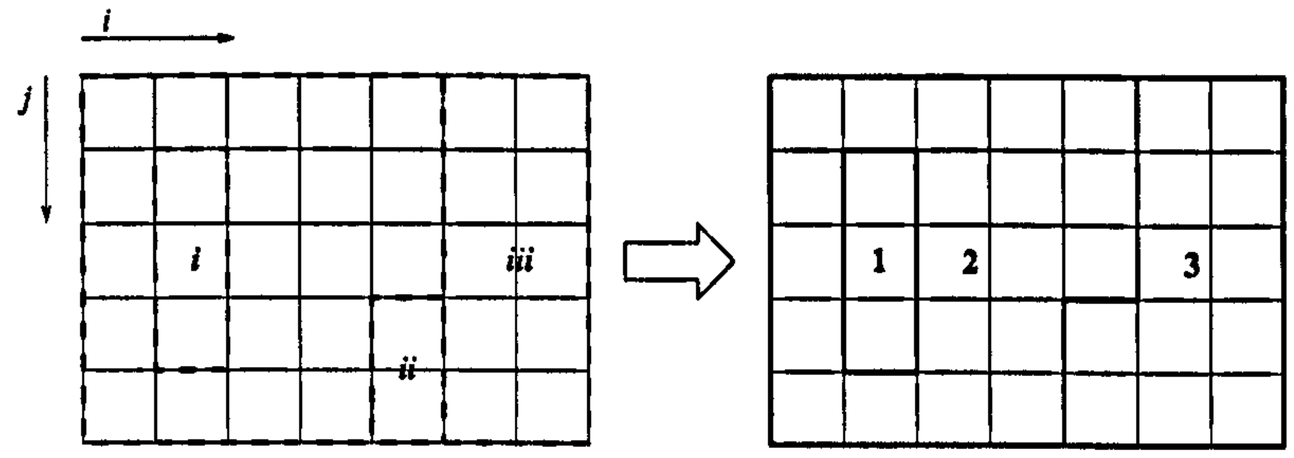


Figure 5.3: *Zones* from *blocks*.

- Zone 2 = $domain - (i + ii + iii)$

- Zone 3 = $ii + iii$

Enviroment zones are defined directly since they are not constituted of computational cells, but represent gates to the external environment.

Finally, connections among zones are required to establish the network. The command file is structured so as to contain those information according to the following syntax:

1. keyword `CODE` followed by package name;

2. keyword `PROBLEM` followed by CFD file name;

3. keyword `BLOCK_DECLARATION` followed by block declarations[4]:

   - name of block;
   - keyword `TYPE` followed by either `CELLBLOCK` (a parallelepiped of cells) or `CELLLIST` (a list of cells);
   - list of cells in a block; within structured grids `CELLBLOCK`s are defined by cell intervals along each main axis (keywords `ICELL`, `JCELL` and `KCELL`)[5];

4. keyword `END_BLOCK`;

5. keyword `ZONE_DECLARATION`. For each internal zone we declare:

   - keyword `INTERNAL_ZONE` followed by the zone name;
   - zone composition in terms of blocks;

   For each environment we declare:

---

[4] Block *domain* need not declaring.

[5] For instance, `ICELL 5:7` means: cells at I coordinates from 6 to 7.

- ENVIRONMENT_ZONE followed by the zone name;

6. keyword END_ZONE;

7. keyword INTERFACE_DECLARATION. For each keyword INTERFACE is followed by the names of the two neighbouring zones.

The command file for the example illustrated in Figure 5.3 is shown in Table 5.1 (two environment zones are also incorporated).

Table 5.1: Command file to set up a zone network.

```
CODE Fluent4
PROBLEM CFD_filename

BLOCK_DECLARATION
 block1
  TYPE CELLBLOCK  ICELL 2:2  JCELL 2:4
 block2
  TYPE CELLBLOCK  ICELL 5:5  JCELL 4:5
 block3
  TYPE CELLBLOCK  ICELL 6:7  JCELL 1:5
END_BLOCK

ZONE_DECLARATION
 INTERNAL_ZONE Izone1
  +block1
 INTERNAL_ZONE Izone2
  +domain -block1 -block2 -block3
 INTERNAL_ZONE Izone3
  +block2 +block3
 ENVIRONMENT_ZONE Ezone1
 ENVIRONMENT_ZONE Ezone2
END_ZONE

INTERFACE_DECLARATION
 INTERFACE Izone1:Izone2
 INTERFACE Izone2:Izone3
 INTERFACE Izone2:Ezone1
 INTERFACE Izone3:Ezone2
END_INTERFACE
```

The method allows an easy and flexible definition of zones in the CFD domain. The user needs to identify a number of blocks within the mesh, which are declared by means of the cells forming each simple unit. Zones are then easily declared by means of simple block operations and by defining the connectivity among zones. Furthermore, information in the command file are sufficient to retrieve all the parameters (zones, interfaces, ports) which are required by the gPROMS model.

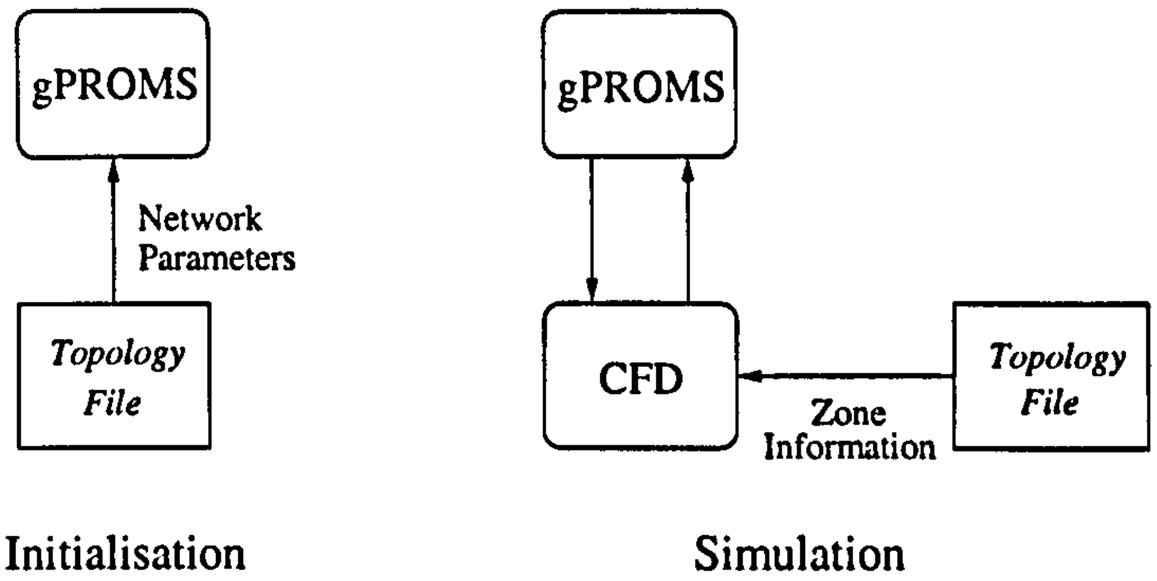


Figure 5.4: Information flux between gPROMS and CFD.

Figure 5.4 briefly illustrates the flux of information in the integrated model. During this initialisation phase, gPROMS reads the zone network definition from the topology file. CFD is not involved. During a simulation, variables and parameters are exchanged between gPROMS and CFD. The topology file is used at this time by the CFD code to obtain information regarding topological relations between cells and zones.

The procedure is general and independent of the type of package. Zones can be defined after geometry is built up and after analysing the results of a preliminary CFD simulation. This procedure permits the easy definition of a zone network in a manual way. However, the criteria used to establish a suitable network of zones entirely depend upon a user's experience. The next sections will discuss the design of a more powerful tool, which enables the automatic generation of a network of zones according to a number of homogeneity and "well-mixedness" criteria.

5.3 Automatic Zoning

The objective here is to design a procedure which is capable of automatically defining a suitable network of zones. According to the definition given in § 3.1, a zone is a region within the process equipment which is assumed to be homogeneous and well-mixed. The criteria which will be discussed in this chapter result in a number of approaches to mark out regions fitting the above definition for internal zones. We define a *zoning method* as an algorithm such that:

Given:

a) a set of cells $\mathcal{C}$;

b) a flow field $\mathbf{F}(c, c')$, $\forall c \in \mathcal{C}$, $\forall c' \in \mathcal{N}_c$, where $\mathcal{N}_c$ is the set of cells which are neighbour to c;

c) a set of np property values $\mathbf{P}(c) = [P_1(c), \ldots, P_{np}(c)]$, $\forall c \in \mathcal{C}$.

Derive:

a) number of zones nz

b) cell-zone allocation $Z(c)$, $\forall c \in \mathcal{C}$

For example, the CFD computational grid is the set of cells $\mathcal{C}$, the mass flowrates between any pair of cells c, c' are represented by the flow field $\mathbf{F}(c, c')$, while the domain distribution of properties such as viscosity, energy of dissipation, etc. is contained in the set $\mathbf{P}(c)$. A zoning algorithm, therefore, is a function Θ such as that:

$$\{nz, Z(\cdot)\} = \Theta\left(\mathcal{C}, \mathbf{F}, \mathbf{P}\right). \tag{5.1}$$

The following sections will detail some suitable functions Θ.

5.3.1 First Method: *Geometric*

The first method (Θ_{geom}) simply cuts the domain into a number of regular parallelepiped-shape zones. Underlying this approach, there is the assumption that a suitable network may be established just by dividing the CFD grid into a number of *small enough* zones. No method is defined to assess the quality of the resulting network. The user's knowledge of the process and the available computational power are the only parameters to determine the zone size.

The number of zones is set by the user by defining the number of intervals desired in each main direction[6]. The algorithm is easily structured as follows:

ALGORITHM GEOMETRIC ($\mathcal{C}$)
Given

i. the number of zones in each direction $\mathbf{n} = [n_x, n_y, n_z]$

ii. the number of cells in each direction $\mathbf{N} = [N_x, N_y, N_z]$;

1. Initialisation.

   a. $\forall c \in \mathcal{C}$, $Z(c) = 0$

   b. $nz = 0$

   c. number of cells per zone in direction i: $nc_i = N_i/n_i$

[6]The method works for structured grids where the cells are orientated and numbered along cartesian or cylindrical axes. In an unstructured grid the algorithm, as it is, is not valid. Nonetheless, it may be substituted by an equivalent algorithm based on the total number of zones or on the size of each zone.

2. Start a new zone:

   a. select the lowest index[7] cell c such that $Z(c) = 0$; if none exists, exit

   b. $nz = nz + 1$

   c. $C_i^* = C_i(c) + nc_i$; if $C_i^* > N_i$, $C_i^* = N_i$;

   d. $I^* = C_x^* + N_x[(C_y^* - 1) + N_y(N_z^* - 1)]$;

3. FOR each cell $c' \in \mathcal{C} \mid Z(c') = 0, I(c') \leq I^*$ DO

   $Z(c') = nz$

   END FOR

4. Go to 2.

This algorithm is a special case of the general algorithm Θ since only the set of cells $\mathcal{C}$ is required to establish a set of zones. Property distribution $\mathbf{P}$ and flowfield $\mathbf{F}$ are not arguments of Θ_{geom}.

As an example, in a structured grid constituted of $50 \times 10 \times 50$ cells, if the user specifies $[10, 2, 10]$ intervals, the algorithm will create a network of 200 cubic zones containing $5 \times 5 \times 5$ cells each. If the number of intervals along a main axis is not a divisor of the number of cells in that direction, then the algorithm will create some bigger zones to satisfy the required number of intervals. For example if we consider the same $50 \times 10 \times 50$ grid and specify $[10, 3, 10]$ intervals, we will obtain two hundred $5 \times 3 \times 5$ zones and one hundred $5 \times 4 \times 5$ zones. The approach differs from the manual definitions in § 5.2, because zones are automatically set up and, accordingly, there is no need to a-priori define *blocks* and zone connectivity.

5.3.2 Second Method: *Delta*

The second method (Θ_Δ) defines the zones as the regions within which the value of one or more properties $\mathbf{P}$ is uniform with respect to a given tolerance vector $\Delta\mathbf{P}$. In other words, a zone is considered to be well-mixed if it is homogeneous within a predefined tolerance with respect to a set of properties.

Cells are considered to belong to the same zone z as long it is true that

$$\mathbf{P}(c) - \mathbf{P}(c') \leq \Delta\mathbf{P} \quad \forall c, c' \in z \ .$$

[7] The cell index $I(c)$ is defined as $I(c) = C_x(c) + N_x[(C_y(c) - 1) + N_y(C_z(c) - 1)]$, where $C_x(c)$, $C_y(c)$ and $C_z(c)$ are the cell coordinates according to axes x, y and z.
The maximum cell coordinates in a zone are named C_x^*, C_y^* and C_z^*, respectively. The corresponding cell index is called I^*.

The number and the type of properties is chosen by the user from a library of available properties implemented in the interface (for example, energy of dissipation, velocity magnitude, effective viscosity, etc.). The algorithm consists of two main steps:

1. select a first cell c;

2. aggregate neighbouring cells till the vector of maximum values $\mathbf{P}^{\mathbf{max}}$ and the vector of minimum values $\mathbf{P}^{\mathbf{min}}$ for properties $\mathbf{P}$ in all cells is such that $\mathbf{P}^{\mathbf{max}} - \mathbf{P}^{\mathbf{min}} \leq \Delta\mathbf{P}$.

The algorithm formal structure is as follows:

ALGORITHM DELTA $(\mathcal{C}, \mathbf{F}, \mathbf{P})$
Given a vector of tolerances $\Delta\mathbf{P}$:

1. Initialisation.
   a. $\forall c \in \mathcal{C},\ Z(c) = 0$
   b. $nz = 0$
2. Start a new zone
   a. Select a cell c such that $Z(c) = 0$; if none exists, exit
   b. $nz = nz + 1$
   c. $Z(c) = nz$
   d. $\mathcal{L} = \{c\}$
   e. $\mathbf{P}^{\mathbf{min}} = \mathbf{P}(c)$; $\mathbf{P}^{\mathbf{max}} = \mathbf{P}(c)$
3. WHILE $\mathcal{L} \neq \{\}$ DO
   a. Select cell c from list $\mathcal{L}$
   b. FOR each cell $c' \in \mathcal{N}_c \mid Z(c') = 0$ DO
      I. Calculate $\mathbf{P}' = min\left(\mathbf{P}^{\mathbf{min}}, \mathbf{P}(c')\right)$
      II. Calculate $\mathbf{P}'' = max\left(\mathbf{P}^{\mathbf{max}}, \mathbf{P}(c')\right)$
      III. IF $\mathbf{P}'' - \mathbf{P}' \leq \Delta\mathbf{P}$ THEN
         i. $Z(c') = nz$
         ii. $\mathbf{P}^{\mathbf{min}} = \mathbf{P}'$; $\mathbf{P}^{\mathbf{max}} = \mathbf{P}''$
         iii. $\mathcal{L} = \mathcal{L} \cup \{c'\}$
      END IF
   END FOR
   END WHILE
4. go to 2.

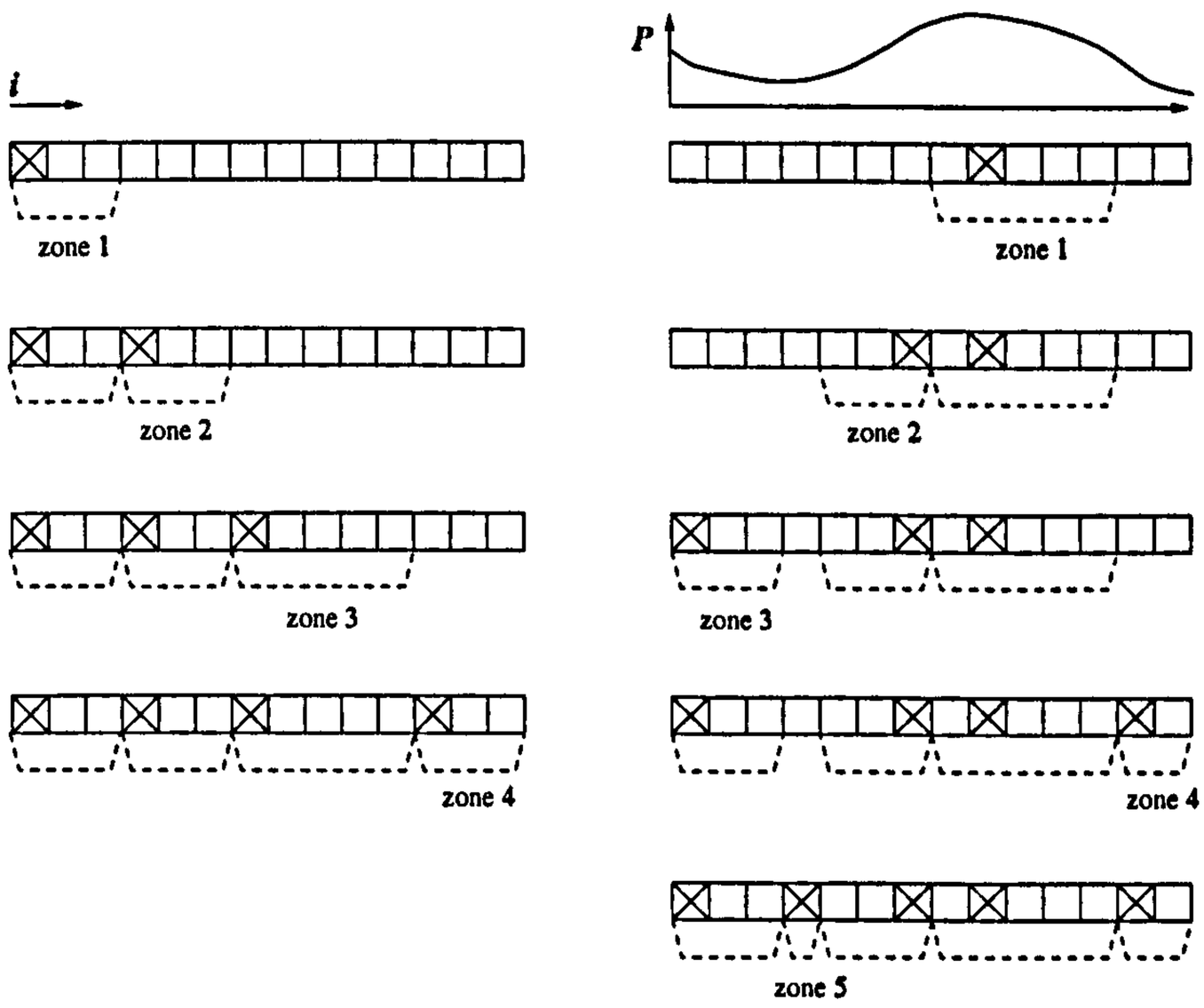


Figure 5.5: Both the number and the shape of the zones depend on the first cell to which the zoning algorithm is applied (crossed cells).

The first step (2a) sets the first cell of a new zone. Of course, the resulting network of zones depends on which cell is selected as first cell. The simplest option (*minimum-index* strategy) to perform this step is to choose the cell (such as $Z(c) = 0$) having the minor index $I(c)$ (§ 5.3.1) as the initial cell. The left side of Figure 5.5 illustrates this approach for a 1D grid. Another suitable choice is to initiate the zoning algorithm from the cell storing the maximum (or minimum) value of

$$P_{tot} = \frac{P_1}{P_{1,av}} + \ldots + \frac{P_{np}}{P_{np,av}} \quad \text{where} \quad P_{i,av} = \sum_{j=1}^{nc} P_i(c_j)$$

in the domain (*maximum-property* or *minimum-property* strategy). The right side of Figure 5.5 shows the second approach with the maximum property method. Our implementation of algorithm Θ_Δ adopts the minimum-index strategy. In a structured grid, the procedure ensures that a new zone will have at least one cell which is neighbouring to previously set zones.

The second main step defines the order which is adopted to select suitable neighbouring cells. In our implementation, it was decided to select the cell c whose property values are the most distant from the bounds $\mathbf{P^{max}}$ and $\mathbf{P^{min}}$. Given an initial set $\mathcal{L}$ of cells which are neighbours to one or more cells $c \mid Z(c) \neq 0$ and present values $\mathbf{P^{max}}$ and $\mathbf{P^{min}}$:

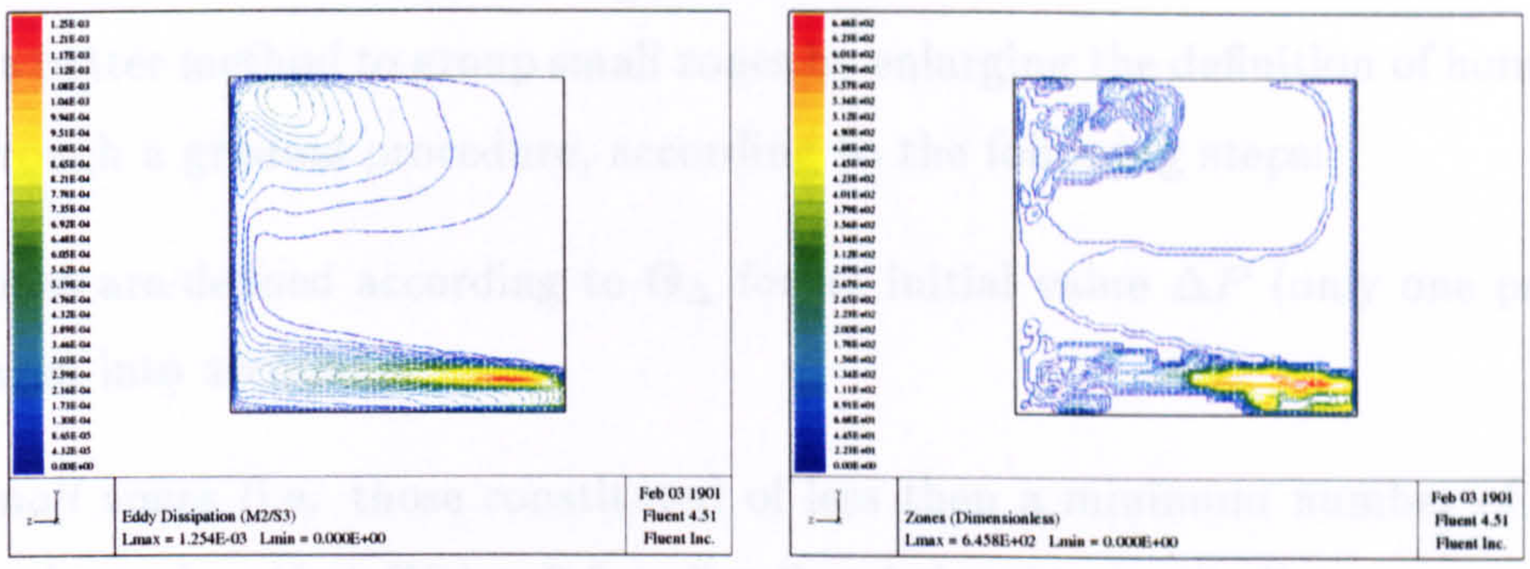


Figure 5.6: Dissipation field as a result of a CFD simulation. Two section of a vessel are represented.

1. compute the distance from the bounds according to:

$$D\left(c, \mathbf{P^{min}}, \mathbf{P^{max}}\right) = \min_i \min \left[\frac{\left(P_i(c) - P_i^{min}\right), \left(P_i^{max} - P_i(c)\right)}{\frac{1}{2}\left(P_i^{max} - P_i^{min} + \varepsilon\right)} \right], \qquad (5.2)$$

$$\forall c \in \mathcal{L}$$

2. Select $c : \max_{c \in \mathcal{L}} D\left(c, \mathbf{P^{min}}, \mathbf{P^{max}}\right)$

Parameter ε in eqn. (5.2) is a small value to avoid division by 0. The suggested method builds up a zone by searching new suitable cells according to a criterion of minimum change in the vector of property $\mathbf{P}$.

The capability of the algorithm to define zones whose shape and dimension is similar to the property distribution field in the CFD domain is illustrated in Figure 5.6. The picture represents a section of a vessel (more details in chapter 6): on the left is the contour plot (lines of constant magnitude) of the eddy dissipation rate distribution field (SI units of m^2/s^3); on the right the contour plot displays the zones computed by Θ_Δ ($\Delta P = 0.0005$ m^2/s^3, $nz = 120$). It is clear that the algorithm is able to reproduce very accurately the distribution pattern of the property of interest.

5.3.3 Third Method: *Grad_delta*

The third zoning algorithm ($\Theta_{grad\Delta}$) is a variation of the method of § 5.3.2 in order to obtain a small number of zones when a high discretisation is required. In fact, several simulations have demonstrated that small values of the property interval ΔP may produce a very large number of zones (too large for the available computational capabilities). Nonetheless, in some cases a high resolution may be needed to locally describe some of the phenomena occurring in the process.

A solution is obtained by the algorithm that we will describe next. The algorithm defines a better method to group small zones by enlarging the definition of homogeneous zone through a gradual procedure, according to the following steps:

1. zones are defined according to Θ_Δ for an initial value ΔP (only one property is taken into account);
2. *small* zones (i.e. those constituted of less then a minimum number of cells) are discharged so that $Z(c) = 0$ for all cells c belonging to small zones;
3. a larger ΔP value is set and the algorithm Θ_Δ is run again (only the cells $c \mid Z(c) = 0$ are considered).
4. the algorithm is concluded when either an acceptable number of zones or a pre-set upper bound for ΔP is reached.

The algorithm is the following:

ALGORITHM G_DELTA ($\mathcal{C}, \mathbf{F}, P$)
Given:

i. maximum and minimum value for the tolerance ΔP, ΔP_{max} and ΔP_{min}

ii. maximum number of zones nz_{max}

iii. minimum number of cells per zone nc_{min}

iv. weight $\alpha > 0$:

1. Initialization.
   a. $\forall c \in \mathcal{C}$, $Z(c) = 0$
   b. $nz = 0$
   c. $\Delta P = \Delta P_{min}$
2. Perform DELTA ($\mathcal{C}$, $\mathbf{F}$, P)
3. $n(Z(c)) = 0$
4. IF ($nz > nz_{max}$ AND $\Delta P \leq \Delta P_{max}$) THEN
   a. $nz = 0$
   b. FOR each cell c DO
      I. $n[Z(c)] = n[Z(c)] + 1$
      II. IF $n[Z(c)] = nc_{min}$ THEN
          $nz = nz + 1$
      END IF
   END FOR

c. FOR each cell $c \mid n[Z(c)] < nc_{min}$ DO

$$Z(c) = 0$$

END FOR

d. $\Delta P = \Delta P + \alpha \Delta P$;

e. Go to 2.

ELSE Stop.

The problem of the method is that it uses a less strict definition of homogeneity for the regions where the property gradient is sharper.

5.3.4 Fourth Method: *Strong*

The fourth algorithm (Θ_S) defines some correlation to link the concepts of mixing and zone connectivity. Zones interact through mass flowrates from one zone to another. The number, the dimension and the shape of the zones affect the distribution of mass fluxes and of those properties associated with them. The CFD finite volumes or finite elements solution is already a very fine network setting mass fluxes among the cells. In general, although each cell is theoretically connected to each neighbour, some main stream lines are established according to the principal mass fluxes in the domain. Our approach will rely on the idea that it is possible to identify *weak* connections in the network based on the main directions of mass fluxes through a single cell rather than just topological (cell proximity) or field distribution arguments. The weak connections are neglected and a simplified network is left (Figure 5.7). The approach aims at identifying sets of cells representing highly mixed portions of the equipment. These zones may detect segregated regions (i.e. regions demonstrating recirculation, but little mass transfer out of themselves) or compartments demonstrating recirculation or backward mixing in a otherwise clearly directed flow. The latter case may be exemplified by a tube reactor. If the flow regime is a perfect plug-flow, no mixing occurs and algorithm Θ_S identifies as many zones as there are cells. On the contrary, if there is radial mixing and backward flow, then algorithm Θ_S aggregates cells belonging to recirculation regions, although there may not be dead regions in the tube. This method is capable of detecting segregation in the fluid flow behaviour. Although it is not guaranteed that segregated regions are homogeneous and well-mixed, it is rather obvious that homogeneous and well-mixed zones should not contain segregated regions. Relying upon this necessary condition, we adopted this criterion to identify zones within a fluid domain.

The identification of weak connections is achieved by means of the algorithm described below:

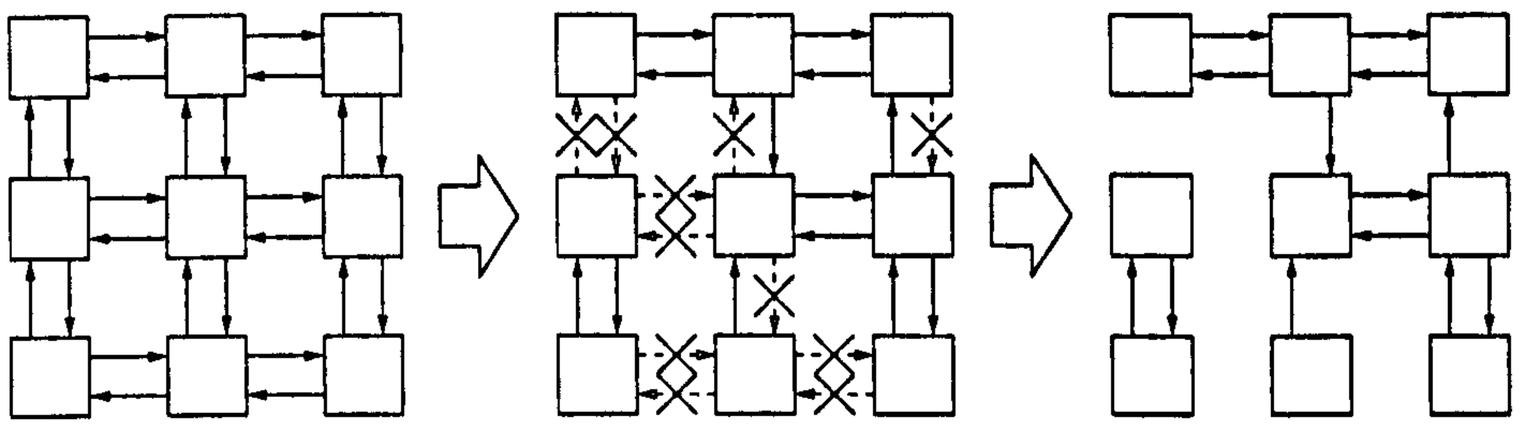


Figure 5.7: Definition of a new network when *weak* connections are deleted.

ALGORITHM SIMPLIFY ($\mathcal{C}$, $\mathbf{F}$)
Given weight $w \in [0, 1]$:

1. Initialisation.

$$\mathcal{L}_S = \{\}$$

2. FOR each cell $c \in \mathcal{C}$ DO

   a. Calculate $\phi_{max,c} = \max(\phi_{cc'}) = \max\left(\frac{f_{cc'}}{A_{cc'}}\right)$

   b. FOR all fluxes *from* cell c *to* cell c' $\phi^+_{cc'} \mid \phi^+_{cc'} \geq w(\phi_{max,c})$ DO

$$\mathcal{L}_S = \mathcal{L}_S \cup \{(c, c')\}$$

   END FOR

3. Stop.

Figure 5.7 illustrates the procedure:

a. all mass fluxes are considered;

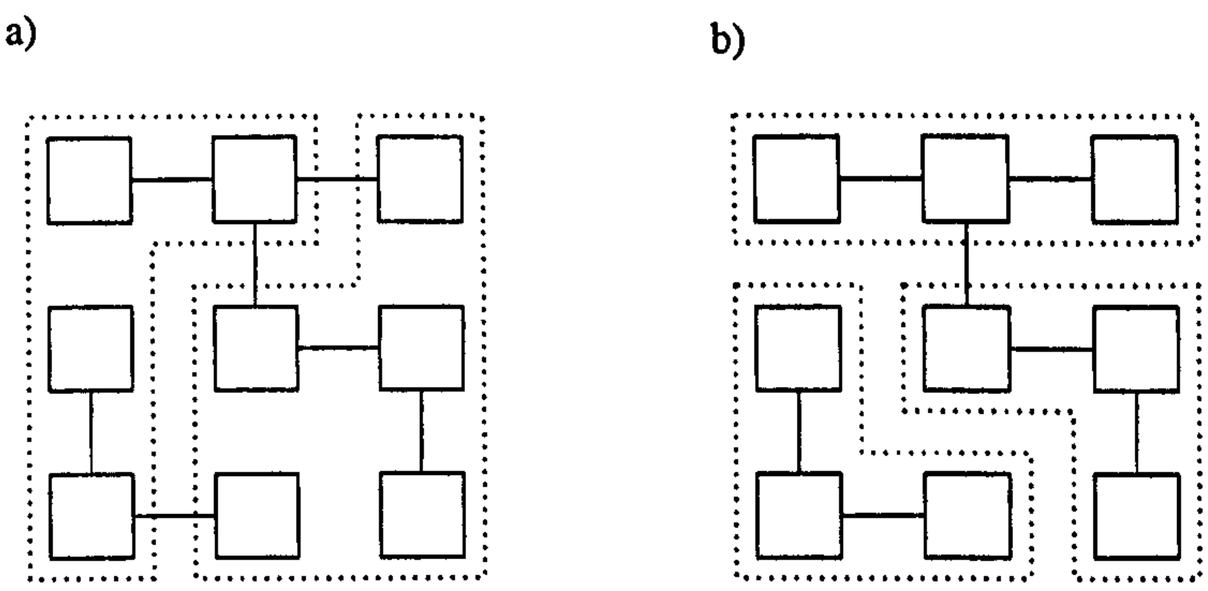
b. for each cell any outlet flux which is *negligible* with respect to the maximum inlet/outlet cell flux is neglected;

c. major fluxes are kept as ordered couples (c, c') in the list $\mathcal{L}_S$.

Note that the weight w is the parameter defining whether a flux should be neglected or not. If $w = 0$, then no flux is discharged; if $w = 1$ only one flux (either inlet or outlet) for each cell is kept.

After the cancellation of *weak*[8] connections, the cell network may become split into two or more sub-networks (Figure 5.7c). Accordingly, the first rule for the definition of zones is as follows: *a zone is constituted of cells belonging to the same* sub-network, *after the* weak *connections are neglected* (Figure 5.8). The network which is obtained

[8] Since it may be misleading, we want to make clear that there is no relation between the term *weak* connection and the rigorous definition of *strongly connected* that will introduced later in this section.



NO YES

Figure 5.8: Definition of a new network of zones: the zoning is acceptable only within connected areas.

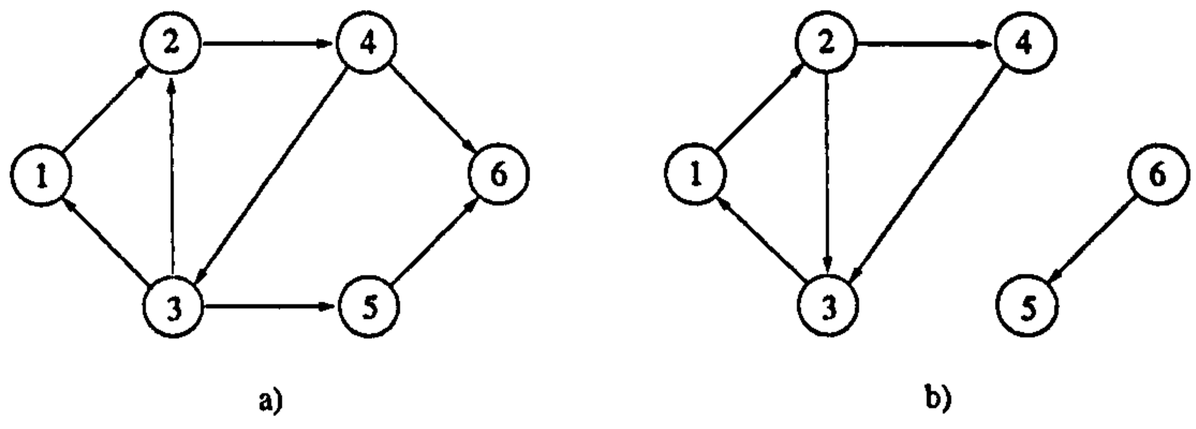


Figure 5.9: The graph shown in a) is connected, while the graph shown in b) is disconnected. The latter graph has two components.

after the cancellation of *weak* connection is a *digraph* (§ 4.4.1). The CFD grid is a network of cells which can be represented by means of a digraph $\mathcal{G}_N$. Algorithm SIMPLIFY simplifies the structure of the CFD network producing a reduced graph $\mathcal{G}_S$ which maintains only the arcs responsible for the main stream lines in the fluid flow.

Now, we will introduce some definitions about graphs.

Definition 6 *We say that two nodes x_i and x_j are* connected *if the graph contains at least one path from node x_i to node x_j. A graph is* connected *if every pair of its nodes is connected (Figure 5.9a); otherwise the graph is* disconnected *(Figure 5.9b). We refer to the maximal connected subgraphs of a disconnected network as its* components.

Definition 7 *A digraph G is* strongly connected *or* strong *if for every ordered pair $x_i, x_j \in X$ there is a x_i, x_j path in G.*

Definition 8 *The* strong components *of a digraph G are the maximum strongly connected subgraph of G (Fig. 5.10). A digraph is strongly connected if and only if it has only one strong component.*

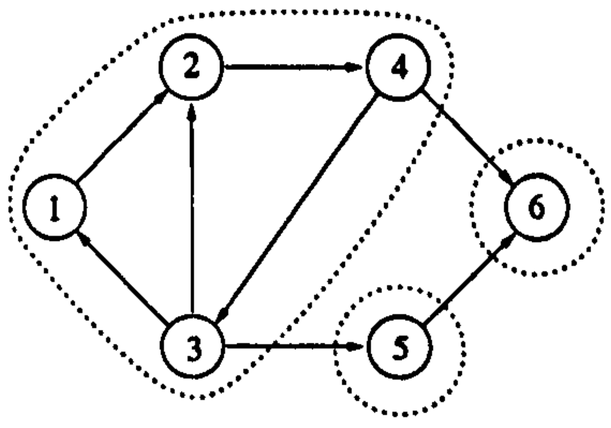


Figure 5.10: The strong components of this graph are the node sets $\{1,2,3,4\}$, $\{5\}$ and $\{6\}$.

Strong components can be identified within graph $\mathcal{G}_S$. There are efficient methods to obtain all the strong components in a graph (Sargent and Westerberg, 1964, Tarjan, 1972). We used the approach developed by Tarjan (1972) implemented as TARJAN ($\mathcal{C}$, $\mathcal{L}_S$) to detect highly mixed regions in the simplified graph $\mathcal{G}_S$.

Several simulations demonstrated that the application of the Tarjan algorithm as stated is of no practical use for our purposes, since in most cases (even for very small values of weight w) the result is a very limited number of large zones and a too great number of strong components constituted of very few cells (often only one cell). The suggested Θ_S algorithm overcomes this problems by using an aggregation procedure which is analogous to the one of the $\Theta_{grad\Delta}$ algorithm, as follows:

1. an initial weight w is set and algorithms SIMPLE and TARJAN are run;
2. *small* zones (i.e. constituted of less then a minimum number of cells) are eliminated ($Z(c) = 0$ for all cells c belonging to small zones);
3. a smaller w value is set and the algorithms SIMPLE and TARJAN are run again (only the cells $c \mid Z(c) = 0$ are considered).
4. the algorithm is concluded when either an acceptable number of zones or a pre-set lower bound for w is reached.

The formal structure of algorithm Θ_S is the following:

ALGORITHM STRONG ($\mathcal{C}$, $\mathbf{F}$)
Given

i. minimum and maximum value for weight w, w_{min} and w_{max}, respectively

ii. maximum number of zones nz_{max}

iii. minimum number of cells per zone nc_{min}

iv. coefficient $\alpha > 0$

1. Initialisation.

a. $\forall c \in C,\ Z(c) = 0$

b. $nz = 0$

c. $w = w_{max}$

2. Perform SIMPLE (C, F)

3. Perform TARJAN (C, $\mathcal{L}_S$)

4. $n[Z(c)] = 0$

5. IF ($nz > nz_{max}$ AND $w \geq w_{min}$) THEN

a. $nz = 0$

b. FOR each cell c DO

I. $n[Z(c)] = n[Z(c)] + 1$

II. IF $n[Z(c)] = nc_{min}$ THEN

$nz = nz + 1$

END IF

END FOR

c. FOR each cell $c \mid n[Z(c)] < nc_{min}$ DO

$Z(c) = 0$

END FOR

d. $w = w - \alpha w$;

e. Go to 2.

ELSE Stop.

The zoning is achieved without defining any homogeneity criterion. The algorithm does not require any property distribution **P** knowledge since the zone network is set up by considering mass fluxes in the equipment only.

5.3.5 Fifth Method: *Hybrid*

The last algorithm (Θ_{hyb}) is a combination of the Θ_Δ and Θ_S algorithms. This method is based on the idea that that a good procedure should demonstrate an ability to detect homogeneous regions as well as segregated ones. The procedure (Figure 5.11) goes through the following steps:

1. algorithm Θ_S is run ($w_{max} = w_{min}$);

2. a network of *subsets* is established: the network is made of strongly connected sets containing more than a number nc_S cells plus a set constituted of all remaining cells;

3. within each network subset algorithm Θ_Δ (or $\Theta_{grad\Delta}$) is performed to detect homogeneous zones.

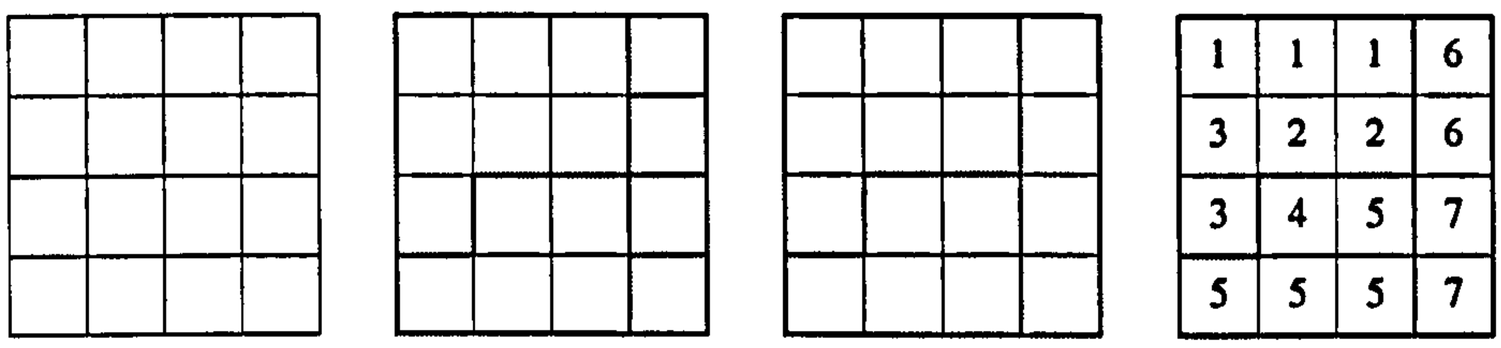


Figure 5.11: *Hybrid* method: the cell network (a) is first divided into strongly connected components according to algorithm *strong* (c), then the strong components containing less then $ncs_{,min}$ are grouped into one network subset leading to a network of three subsets (c); finally, zones (identified by a number) are defined within each subset by applying a suitable algorithm.

The formal structure of the algorithm (Θ_Δ is adopted) is as follows:

ALGORITHM HYBRID ($\mathcal{C}$, $\mathbf{F}$, $\mathbf{P}$)
Given

i. weight w (i.e. $w_{max} = w_{min} = w$)

ii. tolerance $\Delta\mathbf{P}$

iii. minimum number of cells within a strongly connected component nc_S:

1. Initialisation.

   a. $\forall c \in \mathcal{C}$, $Z(c) = 0$

   b. $nz = 0$

2. Perform SIMPLE ($\mathcal{C}$, $\mathbf{F}$)

3. Perform TARJAN ($\mathcal{C}$, $\mathcal{L}_S$)

4. $\forall c \in \mathcal{C}$, $Z'(c) = Z(c)$

5. $\forall c \in \mathcal{C}$, $Z(c) = 0$, $S(c) = 0$

6. $nz = 0$

7. FOR each cell $c \in \mathcal{C} \mid Z(c) = 0, S(c) = 0$ DO

   a. $\mathcal{C}' = \{c\}$

   b. $n = 1$

   c. $S(c) = 1$

   d. FOR every cell $c' \in \mathcal{C} \mid Z'(c') = Z'(c)$ DO

      I. $\mathcal{C}' = \mathcal{C}' \cup \{c'\}$

      II. $n = n + 1$

      III. $S(c') = 1$

   END FOR

   e. IF $n \geq nc_S$ THEN

      Perform DELTA ($\mathcal{C}'$, $\mathbf{F}$, $\mathbf{P}$)

END IF

END FOR

8. Perform DELTA ($\mathcal{C}$, $\mathbf{F}$, $\mathbf{P}$)

9. Stop.

The implemented algorithm aims at

- identifying regions showing little interaction with each other (e.g. dead zones) by analysing strong components in the domain;

- detecting homogeneous zones within each of above segregated regions by observing the distribution of a set of properties.

5.4 New Zone Declaration and Implementation

In § 5.2 we discussed the way zoning information is manually defined for the gPROMS-CFD interface. If zones are automatically set up, the *topology file* has to be slightly modified.

We assume that it is still possible to set a number of internal subsets whose boundaries cannot be shifted or modified. The zoning algorithm will be applied within each of the user defined subsets. The reason for this is that it may be convenient to divide the CFD domain into a number of compartments which are a-priori identified upon the following criteria:

- geometry of the equipment: pipes, tanks, impeller regions may need specific investigation independently of the criteria adopted to set the zones;

- user specific requirements: experimental data may suggest regions of critical importance.

Thus, the block and zone declaration[9] of the command file is not modified from what described in § 5.2. No information regarding environment zones and interfaces are required[10].

The command file will now contain a section for automatic zoning (to be applied to each subset) as follows:

---

[9]Terminology is the same as in § 5.2. However, the zone declaration defines cell subsets, which the zoning algorithm is applied to, and *not* zones.

[10]Nonetheless, the implementation is such that these instruction may be either dropped altogether or kept as in § 5.2. In the latter case, they will be simply neglected.

1. name of the zoning method;

2. zoning parameters (e.g. tolerances, etc.) required by the method.

The syntax is fairly simple:

1. keyword `ZONE_ALGORITHM_TYPE` followed by the name of the algorithm;

2. keyword `ALGORITHM_DECLARATION` followed by a list of keywords identifying the parameters which are specific to the declared algorithm followed by the value of the parameter.

3. keyword `END_ALGORITHM`

For instance, the file to set up algorithm $\Theta_{grad\Delta}$ is illustrated in Table 5.2. The command file includes all the necessary information to set up a network of zones through an automatic algorithm.

Table 5.2: Command file to set up zone network.

```
PROBLEM CFD_filename

BLOCK_DECLARATION
 block1
  TYPE CELLBLOCK  ICELL 2:2  JCELL 2:4
 block2
  TYPE CELLBLOCK  ICELL 5:5  JCELL 4:5
 block3
  TYPE CELLBLOCK  ICELL 6:7  JCELL 1:5
END_BLOCK

ZONE_DECLARATION
 INTERNAL_ZONE Izone1
  +block1
 INTERNAL_ZONE Izone2
  +domain -block1 -block2 -block3
 INTERNAL_ZONE Izone3
  +block2 +block3
 #The environment zone declaration is unnecessary
 ENVIRONMENT_ZONE Ezone1
 ENVIRONMENT_ZONE Ezone2
END_ZONE

ZONE_ALGORITHM_TYPE Grad_Delta

ALGORITHM_DECLARATION
Max_Property_Delta Value1
Min_Property_Delta Value2
Max_ZoneNumber Value3
Min_CellNumber Value4
END_ALGORITHM
```

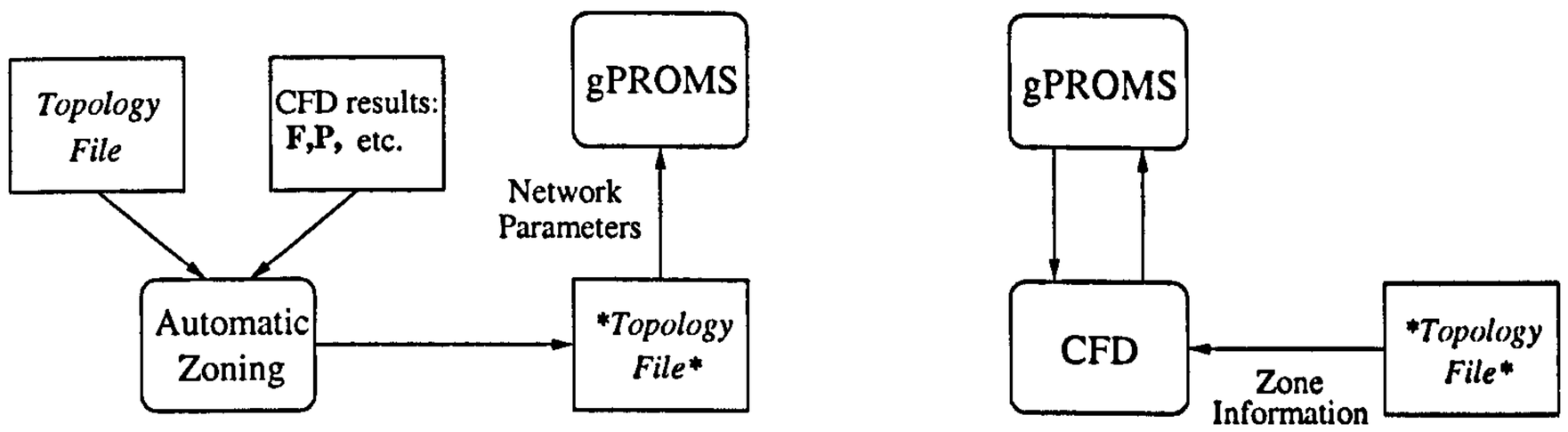


Figure 5.12: Information flux between gPROMS and CFD.

At the beginning of a simulation gPROMS will look for the network parameters in the topology file. The topology file as in § 5.2 is sufficient to set all network parameters and to establish a (CFD cell)-(gPROMS zone) correspondence. That is incorrect if zones are established through an automatic mechanism. Figure 5.12 illustrates the information flowsheet during automatic zone generation. During the initialisation phase, prior CFD results and geometry information are required by the automatic zoning methods to set up the zone network. A new topology file (**topology file** in Figure 5.12) is created and used by gPROMS in the same way as the topology file in § 5.2 (see Figure 5.4). The new command file contains all the information concerning the network. Data for flow to and from the environment zones are also retrieved from the CFD package. This is based on the capability of the CFD packages to recognise inlets, outlets, conducting walls, etc. and to make those information available to the user (§ 5.1). Table 5.3 contains an example of topology file as created after the automatic zoning algorithm is performed.

Table 5.3: Topology file as generated by object `gCFD`.

```
PROBLEM CFD_file_name
PREPROCESS
ZONE_DECLARATION
Number_of_Internal_Zones
I1
I2
...
number_of_Environment_Zones
E1
E2
...
END_ZONE
INTERFACE_DECLARATION
number_of_Internal_Internal_Interfaces
I1:I2
```

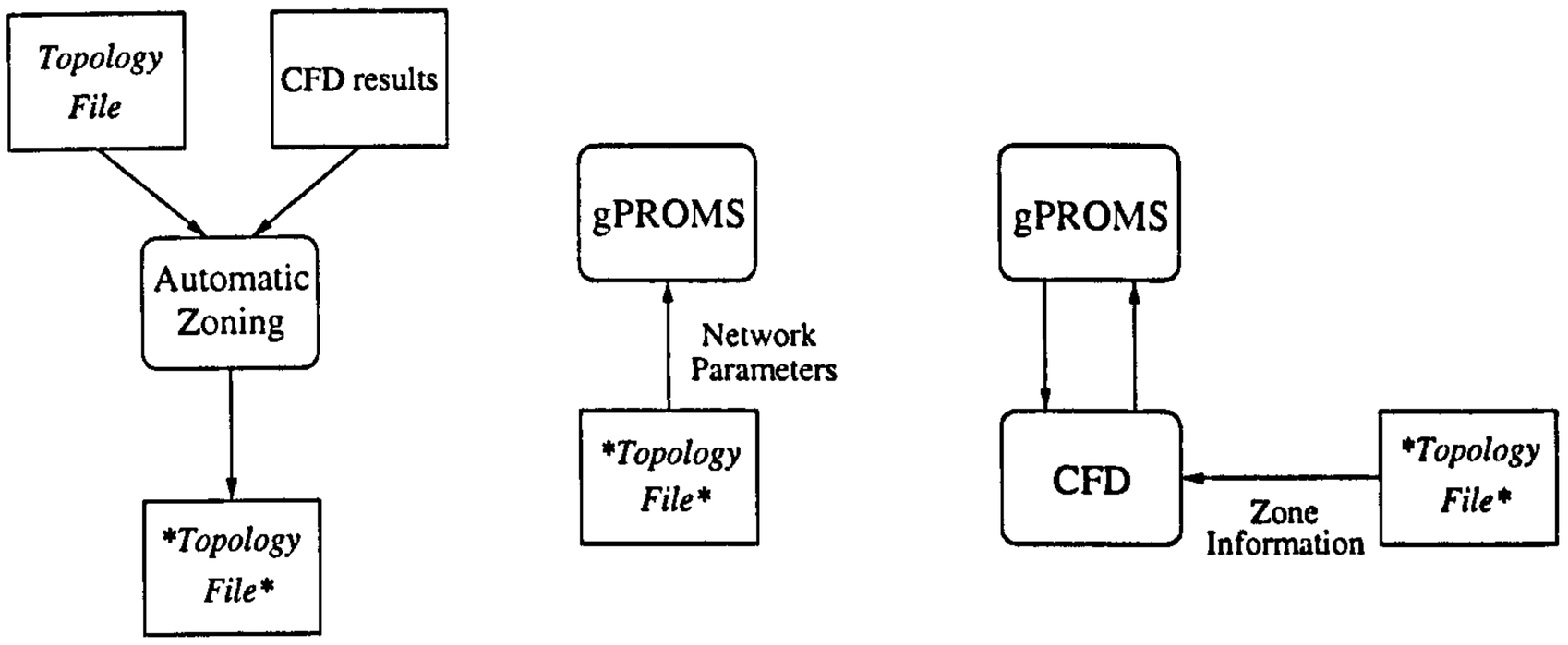


Figure 5.13: Information flux between gPROMS and CFD.

```
...
number_of_Internal_Environment_Interfaces
I1:E1
...
END_INTERFACE
```

Keyword PREPROCESS tells the interface that the automatic zoning algorithm was executed and that the file which is being read is not the original topology file, but a file automatically generated with a slightly different (more efficient) structure.

A procedure was implemented in order to run the zoning algorithm independently of the integrated simulation as a pre-processing step. The procedure uses the topology file in Table 5.2 to create the new topology file of Table 5.3. In this way it is possible to check the zone network before running a simulation. Figure 5.13 illustrates the simulation steps if zoning algorithms are run independently:

auto-zoning: zones are set up thanks to information and parameters contained in a topology file defined by the user; a new topology file containing the zone network parameters is obtained;

initialisation: at the beginning of the simulation gPROMS retrieves network parameters from the new command file; the CFD package is not involved at this stage;

simulation: variables and parameters are exchanged between gPROMS and the CFD package; network data are retrieved by the CFD package from the new command file.

5.5 Assessing Zoning Quality

Let us consider the general form (5.1) of the zoning algorithm. Zones are set up depending on flow field **F** and the distribution of a set of properties **P**. Property **P** distributions are used to detect well-mixed regions in the domain. The choice of set **P** should reflect the type of properties which are critical to the process. In general, setting zones according to properties which are little related to the phenomena occurring in the equipment should be avoided. For instance, if a crystallisation process is being simulated, well-mixing or velocity distribution are certainly less important than phenomena affecting the nucleation, growth and breakage of crystals such as shear stress or energy of dissipation (on the contrary in a reacting system, mixing may be the fundamental issue).

In this section an a-priori analysis will be defined to assess the quality of the zoning network. Although the approach does not offer any information about the discretisation effectiveness in describing the process phenomena which are to be evaluated, the proposed analysis is able to detect and quantify the effect on a simulation result of averaging CFD properties. In other words, a tool is given *to quantify* the error which is borne from the necessity of using zone averaged values for CFD properties and, accordingly, *to suggest* the need for a network redefinition.

Ideally, there should a correlation linking at least one variable in the simulation model to one or more properties $\mathbf{P}^*$ among the set **P**. In such a case, the subset $\mathbf{P}^*$ is not only used to set up the zone network, but it represents a set of variables which are computed by the CFD model and are required in the simulation zone model. Each zone lumped simulation model can be expressed as:

$$L(\mathbf{X}, \mathbf{P}^*) = 0 \quad \Rightarrow \quad \mathbf{X} = \mathbf{X}(\mathbf{P}^*) \ ,$$

where **X** is the set of variables used in the simulation model for the same domain as the CFD model.

Let us consider the case when $\mathbf{X} = X$, i.e. a single variable in the simulation model is a function of the property $\mathbf{P}^*$ and $\mathbf{P}^* \equiv \mathbf{P}$ and let us compare expressions:

$$X = \bar{X} = \frac{1}{V_z} \int_z X(\mathbf{P}) dV \tag{5.3}$$

vs.

$$X(\bar{\mathbf{P}}) \quad \text{where} \quad \bar{\mathbf{P}} = \frac{1}{V_z}\int_z \mathbf{P}\,dV, \tag{5.4}$$

Eqn. (5.3) represents the rigorous way to compute an average X over the volume V_z of zone z. Eqn. (5.4) is the method based on the zoning approach: properties $\mathbf{P}$ are averaged over the zone volume and the averaged values are then passed to the process simulation model. If X is a linear function of $\mathbf{P}$, then eqns. (5.3) and (5.4) are equivalent and, therefore, the size of a zone does not affect the results of the X computation within that zone. The case is different if $X = X(\mathbf{P})$ is a non-linear function. Eqn. (5.3) may be written as:

$$\begin{aligned}\bar{X} = \frac{1}{V_z}\int_z \Bigg[& X(\bar{\mathbf{P}}) + \sum_i \left.\frac{dX}{dP_i}\right|_{\bar{\mathbf{P}}} (P_i - \bar{P}_i) + \\ & \frac{1}{2}\sum_i\sum_j \left.\frac{d^2X}{dP_i dP_j}\right|_{\bar{\mathbf{P}}} (P_i - \bar{P}_i)\,(P_j - \bar{P}_j) + \ldots \Bigg] dV,\end{aligned} \tag{5.5}$$

Considering that

$$\int_z (\mathbf{P} - \bar{\mathbf{P}})\,dV = 0,$$

we can write:

$$\bar{X} \approx X(\bar{\mathbf{P}}) + \frac{1}{2}\frac{1}{V_z}\sum_i\sum_j \left.\frac{d^2X}{dP_i dP_j}\right|_{\bar{\mathbf{P}}} \int_z (P_i - \bar{P}_i)\,(P_j - \bar{P}_j)\,dV \tag{5.6}$$

If we use the definition of covariance matrix for set $\mathbf{P}$ in zone z

$$\mathrm{cov}_{ij}(\mathbf{P}) = \frac{1}{V_z}\int_z (P_i - \bar{P}_i)\,(P_j - \bar{P}_j)\,dV,$$

eqn. (5.6) can formally be written as:

$$\bar{X} - X(\bar{\mathbf{P}}) \approx \frac{1}{2} X\left(\left.\frac{d}{d\mathbf{P}}\right|_{\bar{\mathbf{P}}}\right) \cdot \mathbf{cov}_z \cdot \left(\left.\frac{d}{d\mathbf{P}}\right|_{\bar{\mathbf{P}}}\right)^T. \tag{5.7}$$

When set $\mathbf{P}$ is constituted of only one property P, eqn. (5.7) becomes:

$$\bar{X} - X(\bar{P}) \approx \frac{1}{2}\left.\frac{d^2X}{dP^2}\right|_{\bar{P}} \sigma_z^2, \tag{5.7'}$$

where σ_z^2 is the standard deviation for property P in zone z.

Eqns. (5.7) and (5.7′) may be used to define a validity criterion for each zone z. Given

an acceptable tolerance ΔX, a set of cells is a valid zone if

$$\Delta X \geq \frac{1}{2} X \left(\frac{d}{d\mathbf{P}}\bigg|_{\bar{\mathbf{P}}} \right) \cdot \mathbf{cov}_z \cdot \left(\frac{d}{d\mathbf{P}}\bigg|_{\bar{\mathbf{P}}} \right)^T \tag{5.8}$$

within that zone. If only one property is used to define zone, we obtain:

$$\sigma_z^2 \leq \frac{2\Delta X}{\left(\frac{d^2 X}{dP^2} \right)_{\bar{P}}} \tag{5.8'}$$

For instance, let us consider the relation between the effective viscosity η and the mass transfer coefficient $k_L a$ for a non-Newtonian fluid in an aerated stirred reactor (more details in chapter 8):

$$k_L a = \alpha \eta^{-2/3} \tag{5.9}$$

where α is a constant. The second derivative is represented by the expression:

$$\frac{d^2 k_L a}{d\eta^2} = \frac{10}{9} \alpha \eta^{-8/3} \, . \tag{5.10}$$

Thus, eqn. 5.8′ can be written as:

$$\sigma_z^2 \leq \frac{2\Delta k_L a}{\left(\frac{d^2 k_L a}{d\eta^2} \right)_{\bar{\eta}}} \tag{5.11}$$

Thus, if a tolerance $\Delta k_L a$ is defined for the mass transfer coefficient, then it is easy to verify whether each zone is acceptable or not. As an example, let us consider a 1D case such that spatial distribution of effective viscosity along spatial coordinate l is such that $\eta = l + \eta_0$, where η_0 is the viscosity at $l = 0$. In this case a plot representing variable $k_L a$ in the spatial domain may be easily drawn in the form $k_L a = \eta^{2/3}$ (assuming $\alpha = 1$) as in Figure 5.14 (where $\eta_0 = 0.5$). Let us suppose that three equal zones ($\Delta l = 2$) are defined. The effective viscosity standard deviation is the same in each zone and given by

$$\sigma^2 = \frac{1}{\Delta l} \int_0^2 (l - \bar{\eta}_1 - \eta_0)^2 \, dl = \frac{1}{3}$$

where $\eta_1 = 1.5$ is the average effective viscosity in Zone 1. Considering that:

$$\frac{d^2 k_L a}{d\eta^2}\bigg|_{\bar{\eta}} = \frac{10}{9} \bar{\eta}^{-8/3} \, ,$$

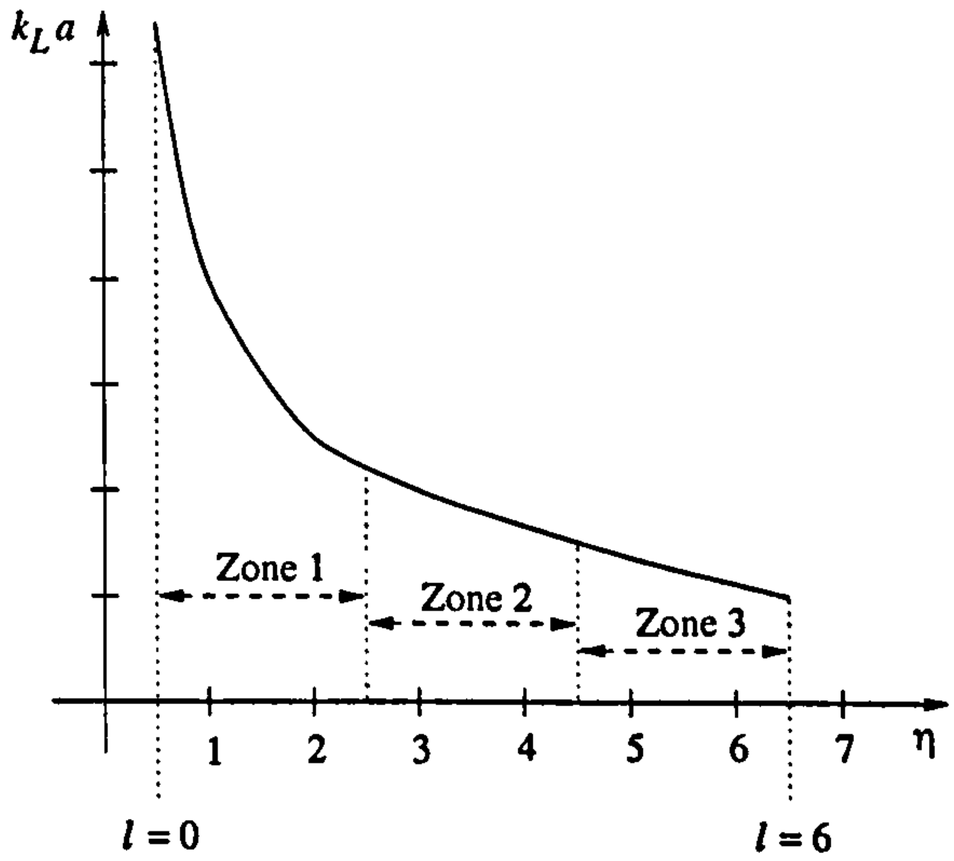


Figure 5.14: Example.

eqn. (5.8′) becomes:

$$\frac{1}{3} \leq \frac{2\Delta k_L a}{0.377}, \quad \frac{1}{3} \leq \frac{2\Delta k_L a}{0.039}, \quad \frac{1}{3} \leq \frac{2\Delta k_L a}{0.012}$$

for Zones 1, 2 and 3, respectively. Given, e.g., a tolerance $\Delta k_L a = 1 \times 10^{-2}$, then Zones 2 and 3 satisfy the condition, but Zone 1 fails the test. In that region, the high non-linearity of the function requires a higher discretisation.

The second derivative of X with respect to $\mathbf{P}$ may not be easily estimated. Relations (5.8) and (5.8′) are replaced by

$$\Delta X \geq \frac{1}{2} \boldsymbol{\xi} \cdot \mathbf{cov}_z \cdot \boldsymbol{\xi}^T \tag{5.12}$$

and

$$\sigma_z^2 \leq \frac{2\Delta X}{\xi^2}, \tag{5.12′}$$

where the product $\xi_i \xi_j$ is an upper bound to $\left.\frac{d^2 X}{dP_i dP_j}\right|_{\bar{\mathbf{P}}}$.

Relations (5.8)-(5.8′) and (5.12)-(5.12′) represent a *validity criterion* for a zone z, which should be satisfied throughout the simulation. If several variables $\mathbf{X}$ are affected by properties $\mathbf{P}$, relation (5.8) becomes:

$$\begin{aligned} \Delta X_1 &\geq \tfrac{1}{2} \boldsymbol{\xi}_1 \cdot \mathbf{cov}_{z,1} \cdot \boldsymbol{\xi}_1^T \\ &\vdots \\ \Delta X_n &\geq \tfrac{1}{2} \boldsymbol{\xi}_n \cdot \mathbf{cov}_{z,n} \cdot \boldsymbol{\xi}_n^T \end{aligned}$$

This criterion represents a first step towards a better understanding of a zone network model. However, we have to point out that relation $X = X(\mathbf{P})$ cannot always be found. In many examples the set of properties P can be related only to the CFD model. It may happen that either there is no way to attribute some properties to a single zone (e.g. zoning algorithm Θ_S) or there is no available relation to link properties to a variable X. For instance, the energy of dissipation ϵ affects the mixing effectiveness within a certain vessel and, as such, is often used to detect and separate different mixing regimes (e.g. Baldyga *et al.*, 1997, Kresta, 1998, Ng and Yianneskis, 2000). Nonetheless, there appears to be no useful correlation to link ϵ to any variable in a mixing model. In such a case the validity criterion may still be used, but should be simplified as in

$$\sigma_z^2 \leq \delta_P, \tag{5.12''}$$

where δ_P is a suitable tolerance. Unfortunately, the choice of the tolerance δ_P is unrelated to the simulation model and only experience can advise about the most proper value.

5.6 Key Results

The main achievements illustrated in this chapter are:

- The definition of manual and automatic procedures to set and exchange zone network information between the CFD and process simulation models. A *topology file* is created to
  - ▷ initialise the process simulation model by passing the required network information;
  - ▷ define the mapping between the process simulation and CFD models during the integrated simulation.
- The design of a series of methods capable of identifying well-mixed and homogeneous zones according to a number of criteria. The criteria are based on:
  - ▷ a set of properties representative of the fluid flow behaviour;
  - ▷ segregation and recirculation according to the well-known definition of *strong components* within a graph.

The *zoning algorithms* based on these criteria enable the automatic definition of a network of zones and the setting of all the parameters required for the integrated simulation.

- The definition of a first a method to assess the quality of zoning in a network of zones: the proposed analysis is able to detect and quantify the effect on a simulation result of averaging CFD properties.

Chapter 6

Zoning Application and Results

This chapter will discuss the practical application of the procedures designed to establish a network of zones. Results of several simulations will be compared and analysed to evaluate the performance of the suggested zoning algorithms.

6.1 Zoning Application

The algorithms described in the previous chapter allow the definition of a network of zones. Here we will suggest a procedure to establish which methods should be adopted and how they should be used. In fact, there are still some "degrees of freedom" which need considering:

- how to select properties required by the zoning methods?

- how to originate CFD fields to apply algorithms to?

The zoning algorithms provide a powerful tools to automatically establish a network of zones. However, the way of using those tools is an engineering problem depending on process and modelling knowledge.

6.1.1 Manual Adjustment

Let us suppose that there is no clear knowledge of the interactions between physical and chemical phenomena and hydrodynamics. That might be due to the great complexity of the process (i.e. there are relations, but they are not identifiable) or to the ignorance concerning the actual existence of such phenomena. In such cases it may be convenient to test a very general algorithm to understand whether and in what respect fluid flow behaviour affects simulation results. The zoning algorithm should be fast to implement and easy to manipulate. Algorithm Θ_{geom} may be a suitable choice: the zone network

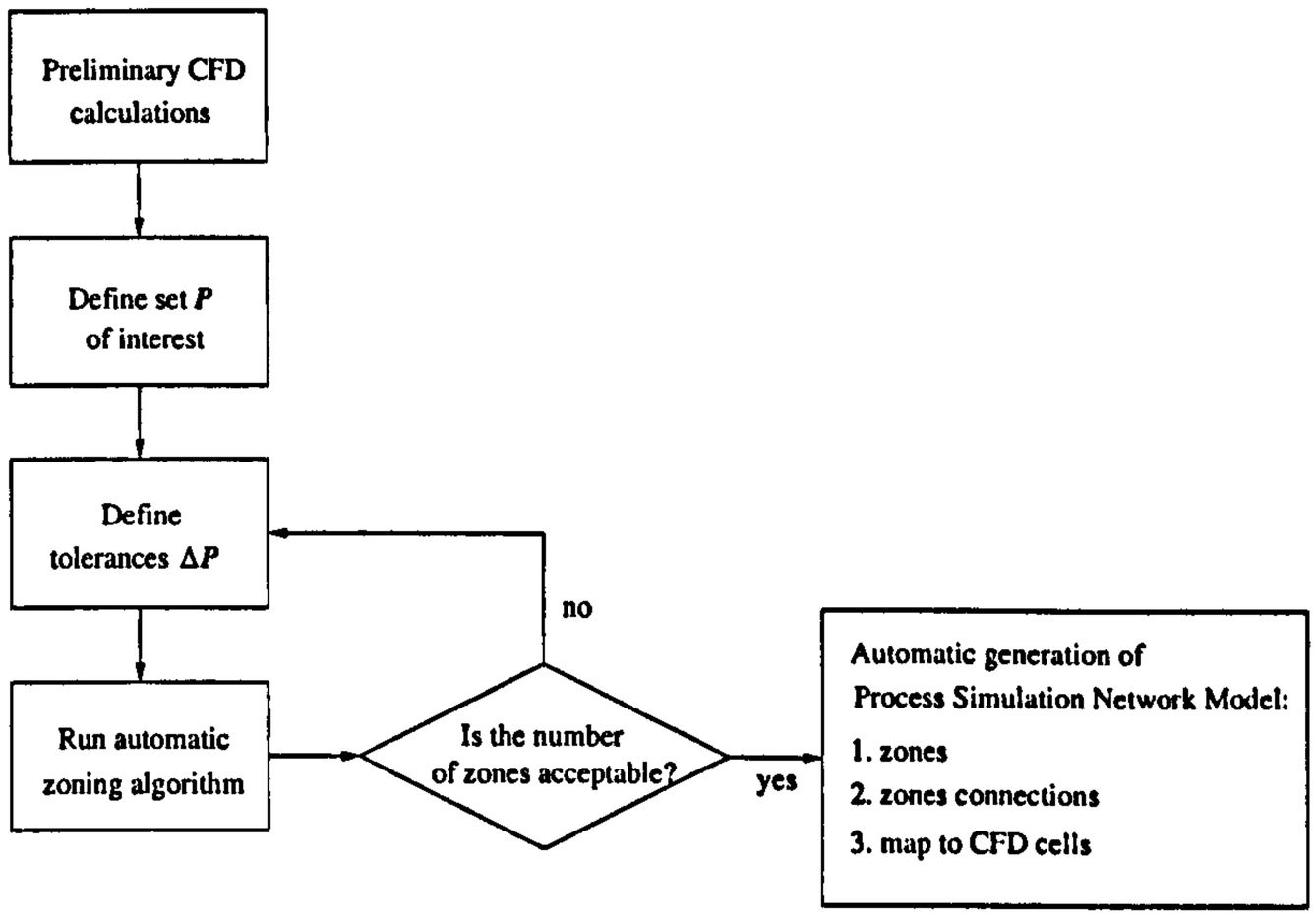


Figure 6.1: Suggested procedure improves traditional modelling and scale-up.

is easily set up and no information is required since the cell size is the only required input. The number and shape of zones may be improved by running a few simulations considering different network of zones and comparing results.

6.1.2 Automatic Zoning Based on Critical Property Distributions

The process may heavily depend on the distribution of a set of properties which are easy to identify and which may be computed by CFD. In such a case, the condition of homogeneity with respect to one or more properties is the principle upon which the zoning procedure relies. In such a case, a preliminary (simplified) CFD simulation generates a field of properties approximating the distribution of those properties in the actual process. If the process is dynamic, then it may be necessary to consider results from CFD calculations at different time steps and use the property distributions at different times as different sets of properties.

Any of the algorithms Θ_{Δ}, $\Theta_{grad\Delta}$ or Θ_{H} fits the requirements. Algorithms Θ_{Delta} and $\Theta_{grad\Delta}$ strictly stick to the distribution of a set of properties **P**; algorithm Θ_{H} is built to simultaneously consider the effects of recirculation. Tolerance Δ**P** is chosen to obtain a suitable number of zones and should be adjusted till a computationally acceptable number of zones is obtained.

Figure 6.1 illustrates the zoning workflow. The main drawback of this procedure is that the zoning is very tailored to map the properties of interest.

6.1.3 Mixing Pattern Identification

In most practical cases within the process industry, CFD is used to identify mixing patterns within equipment. Laboratory and pilot plant experiments supported by CFD simulations are traditionally used to retrieve data to set up unit operation models which are capable of taking into account some of the hydrodynamics and the other fluid flow phenomena (Figure 6.2). Here CFD mixing simulations are used to obtain data to start a zoning algorithm capable to set up a complex model which drastically improve the modelling capabilities of process simulation tools. The zone network is then the framework upon which an integrated model (chapter 2) is established and run. The scope of the procedure illustrated here is to capture and describe those mixing phenomena. The basic idea is to simulate a tracing experiment to understand the mixing pattern and, then, use the data thus generated to set up a network of zones. The following steps illustrates the method:

a. run a dynamic CFD mixing simulation between two species based on fluid flow properties similar to the ones in the actual process;

b. take the concentration (or mass fraction) of one of the species at several time steps and build up vector **P** by considering those concentration distributions; additional properties may be added to **P** in case they also affect the process (e.g. the energy of dissipation in the simulation of a crystallisation process);

c. set tolerance $\Delta\mathbf{P}$;

d. run a suitable zoning algorithm (i.e. Θ_{Δ}, $\Theta_{grad\Delta}$ or Θ_{H});

e. tune $\Delta\mathbf{P}$ till the number of zones is acceptable.

The procedure adopts a CFD mixing simulation to set the zone network. The method is rather complex and time consuming, but it allows the definition of zones which are actually representative of regions of different mixing regimes. In a continuous process, the simulation may be carried out by

- computing the flow field with only one species considered;
- setting the inlet composition with only the second species injected.

In a batch process the experiment is carried out by

- computing the flow field with only one species considered;
- injecting an amount of the second species somewhere in the vessel at a given time.

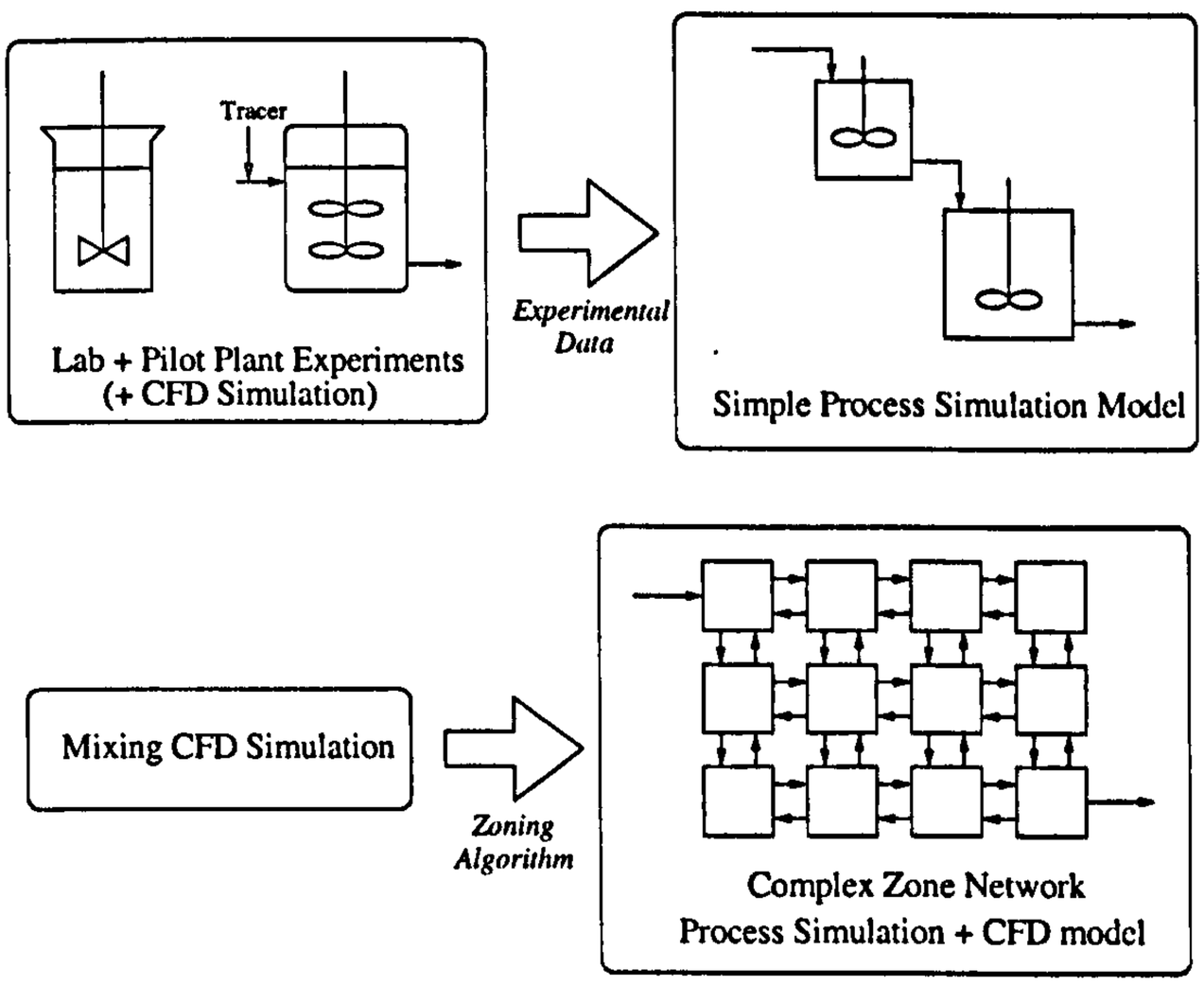


Figure 6.2: Suggested procedure improves traditional modelling and scale-up.

In both cases the experiment is concluded as soon as a homogeneous composition has been reached.

6.2 Mixing Test Example

A series of mixing tests were designed for a small application example to demonstrate how the zoning algorithms work and the effectiveness of their approach. Mixing simulations are performed in the simple geometry illustrated in Figure 6.3 corresponding to a $10 \times 2 \times 10$ m tank with a 1×1 m inlet and outlet (highlighted in blue and red, respectively, in the figure). The inlet and outlet are offset with respect to the x-coordinate, thus creating a 3D flow pattern. The example reproduces and improves a first test that was carried out during a collaborative project with Bayer. Such simple box structure shows rather interesting hydrodynamics exhibiting recirculation and segregation phenomena. The geometry is meshed by means of a structured grid containing 25000 (no. $50 \times 10 \times 50$) computational cells. Because of the regular geometry, cells are all of the same shape and size.

The simulation experiment is carried out through four main steps.

Step 1. *Preliminary CFD simulation*

A steady-state CFD simulation is performed to solve the Navier-Stokes equations and simulate hydrodynamic behaviour in the volume. Parameters for the simulation are

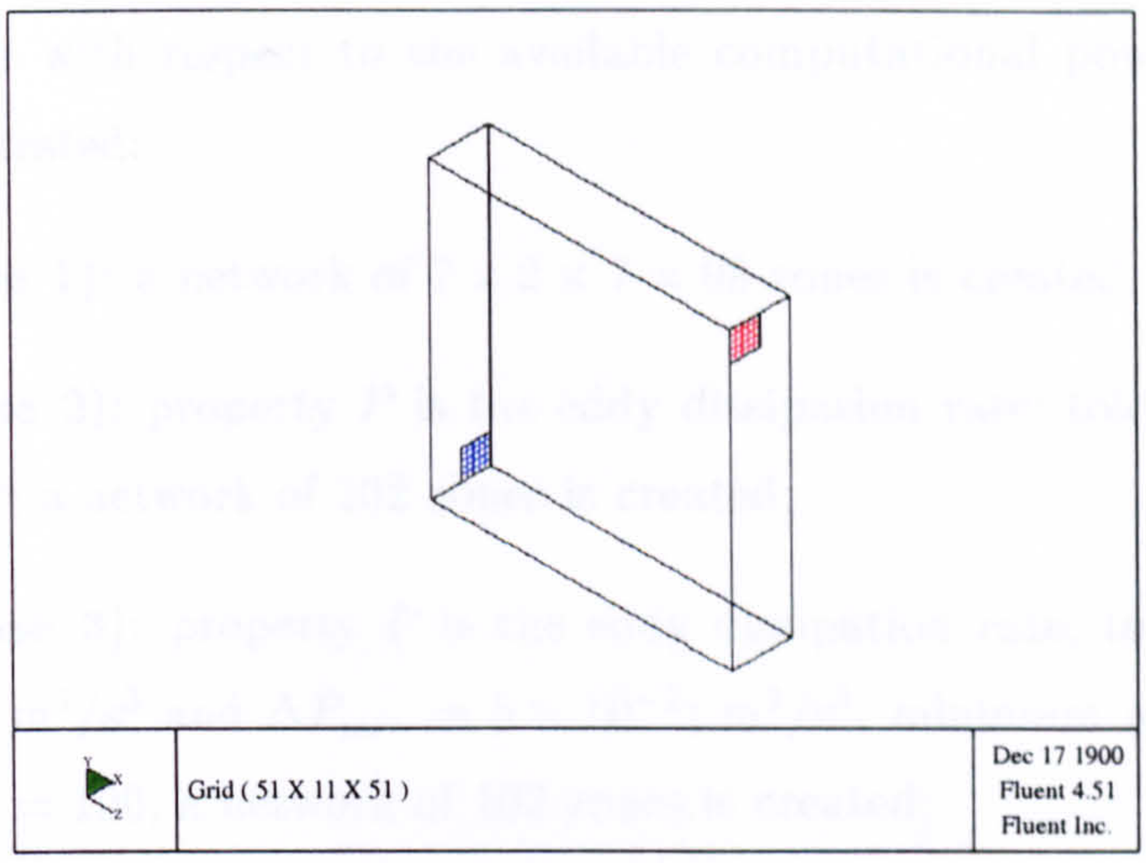


Figure 6.3: Geometry. Inlet and outlet are spotlighted in blue and red respectively.

the following:

- water-like fluid physical properties (density $\rho = 1000$ kg/m^3, viscosity $\mu = 1 \times 10^{-3}$ kg/m-s);
- one chemical species (species 1) in the domain;
- inflow velocity at the inlet: 0.5 m/s (normal to the inlet surface and corresponding to a mass flowrate equal to 500 kg/s).

This preliminary simulation produces the fluid velocity field and estimates the eddy dissipation energy distribution (see Figure 5.6), which will be used in this test to establish a network of zones for a simulation case.

Step 2. *CFD Mixing Pattern Identification*
A dynamic mixing simulation is then performed by injecting a second species in the tank. This is done by switching the inlet composition from $x_1 = 1$, $x_2 = 0$ to $x_1 = 0$, $x_2 = 1$ where x_1, x_2 are mass fractions of species 1 and 2 respectively. Species 2 is treated as a tracer and its physical properties are the same as the first species. Thus, the velocity field does not vary during the mixing and results of the first CFD calculations are still valid. Composition results at different time steps are saved along the simulation.

Step 3. *Zoning*
The previous simulations are used to define the zones according to the methods described in chapter 5. All the algorithms are tested with parameters tuned in order to obtain a similar number of zones (about 100) so that results of a CFD-simulation model

may be compared. Algorithm Θ_S (§ 5.3.4) is not used since the resulting number of zones is excessive with respect to the available computational power. The following algorithms were tested:

- Θ_{geom} **(case 1)**: a network of $7 \times 2 \times 7 = 98$ zones is created.

- Θ_{Delta} **(case 2)**: property P is the eddy dissipation rate; tolerance $\Delta P = 5.2 \times 10^{-4}$ m^2/s^3; a network of 102 zones is created;

- $\Theta_{grad\Delta}$ **(case 3)**: property P is the eddy dissipation rate; tolerances $\Delta P_{max} = 1.9 \times 10^{-4}$ m^2/s^3 and $\Delta P_{min} = 5 \times 10^{-2}$; m^2/s^3; minimum number of cells per zone $nc_{min} = 100$; a network of 102 zones is created;

- Θ_H **(case 4)**: property P is the eddy dissipation rate; tolerances $\Delta P_{max} = 2.8 \times 10^{-4}$ m^2/s^3 and $\Delta P_{min} = 5 \times 10^{-2}$ m^2/s^3; minimum number of cells per zone $nc_{min} = 100$; mininum number of cells within a strong component $nc_S = 600$; a network of 105 zones is created;

- Θ_Δ **(case 5)**: set of properties $\mathbf{P}$ is constituted by the mass fraction distribution of species 2 at time steps $t = 100$ s, $t = 200$ s, $t = 400$ s and $t = 600$ s; tolerances $\Delta\mathbf{P} = [0.30, 0.30, 0.15, 0.075]$[1] ; a network of 103 zones is created.

Step 4. *Integrated Mixing Simulation*

An integrated CFD-gPROMS simulation is performed for the tracing experiment described in step 2 using the networks of zones established at step 3 to reproduce mixing in the equipment. The process simulation model describes each zone as a perfectly mixed vessel whose composition only depends on the zone inlet composition. Initial conditions are set by imposing composition $x_1 = 1$, $x_2 = 0$ for all internal zones and $x_1 = 0$, $x_2 = 1$ in the inlet environment zones. Each zone model is expressed by the following equations:

- mass balance:

$$\frac{dM_i}{dt} = \sum_{j=1}^{ni} F_j^{in} X_{ij}^{in} - \sum_{j=1}^{no} F_j^{out} X_i \qquad i = 1, 2 \tag{6.1}$$

[1] Different tolerance values are taken into account since composition becomes more uniform during the blending and smaller values are required to properly detect the distribution of the species. The very large tolerance values for time steps $t = 100$ s and $t = 200$ s depend on the fact that the mixing process initially exhibits sharp gradients which would otherwise produce an excessive number of zones.

- total mass in the volume:

$$M_T = M_1 + M_2 \tag{6.2}$$

- mass fractions:

$$X_i M_T = M_i \qquad i = 1, 2 \tag{6.3}$$

Since density is constant eqn. (4.8) relating volume to total holdup is unnecessary as explained in § 4.2.4.
Composition results of integrated simulations are eventually passed back to CFD computational cells in order to obtain CFD plots of the gPROMS-CFD computations.

6.2.1 Assessment Criteria

The results from the CFD-gPROMS simulation are compared to those from the rigorous CFD mixing simulation. The following criteria are used in the comparison (x is the mass fraction of the injected species, nc is the number of computational cells in the CFD grid):

- Average mass fraction $\bar{x} = \dfrac{\sum_{i=1}^{nc} x_i}{nc}$
- Standard deviation $\sigma = \left(\dfrac{1}{nc} \sum_{i=1}^{nc} (x_i - \bar{x})^2 \right)^{0.5}$
- Third moment $\mu = \dfrac{1}{nc} \sum_{i=1}^{nc} (x_i - \bar{x})^3$
- Error $\varepsilon = \left(\dfrac{1}{nc} \sum_{i=1}^{nc} (x_{i,z} - x_{i,cfd})^2 \right)^{0.5}$,

  where the subscripts z and cfd denotes the gPROMS-CFD and the CFD simulations, respectively
- Relative error $\varepsilon_r = \varepsilon / \bar{x}_{cfd}$

Note that criteria are not normalised with respect to the cell volumes because in this specific case cells have all the same size (so, e.g., $\bar{x} = \sum_{i=1}^{nc} x_i V_i / V = \sum_{i=1}^{nc} x_i / nc$).

The average composition $\bar{x}$ is a good indicator to detect whether the mixing models estimate a different residence time. In fact, the average composition at a certain time

estimates the amount of tracer which is still within the tank. Results are also compared to the case of perfect mixing in the tank, expressed by the differential equation:

$$M_T \frac{dx}{dt} = F^{in} x^{in} - F^{out} x, \tag{6.4}$$

whose solution (note that $F \equiv F^{in} = F^{out}$ at steady-state) is:

$$x(t) = x^{in} \left(1 - e^{-\frac{F}{M_T} t}\right), \tag{6.5}$$

i.e. an exponential behaviour towards $x(t) = x^{in}$.

The second assessment criterion is the mass fraction standard deviation σ. Standard deviation is a helpful index to understand the species segregation in the vessel. Although a single parameter is certainly not sufficient to outline the species distribution in the vessel, it is nonetheless a useful tool to compare different simulations and detect how far they are from a perfect mixing case ($\sigma = 0$).

The third criterion is the third moment μ. It evaluates the symmetry of the species distribution. In other words, moment μ points out if the system looks like a Gaussian distribution (i.e. distribution of values $x < \bar{x}$ reflects distribution of values $x > \bar{x}$) or, on the contrary, most computational cells present mass fraction values $x < \bar{x}$ or vice-versa.

The fourth and fifth criteria estimate the error with respect to the rigorous CFD simulation. The error ε is an absolute measure of the average difference in a cell between CFD and CFD-gPROMS simulations. Relative error ε_r normalises ε with respect to average mass fraction $\bar{x}$.

Simulations are also compared through plots of mass fraction at various times in different sections of the vessel. As it usually happens in CFD simulations, they probably offer the best aggregate understanding of the model results and the most significant tool to compare different simulations at a glance.

The previous criteria evaluate the results of the simulations by considering the entire domain at once. Other assessment criteria are defined for a *local* evaluation of the results. The following criteria are defined:

- the number of cells nc_z within zone z.

- the rigorous average mass fraction $\bar{x}_{CFD,z}$ computed over CFD cells corresponding to simulation zone z ($\bar{x}_{CFD,z}$ is compared to the average mass fraction $\bar{x}_z$ within zone z).

	CFD	Case 1	Case 2	Case 3	Case 4	Case 5	Perf. Mix.
time $t = 100$							
$\bar{x}$	0.2213	0.2163	0.2355	0.2283	0.2187	0.2192	0.2212
σ	0.2514	0.2322	0.1390	0.1621	0.1684	0.2203	0
μ ($\times 10^2$)	1.93	1.33	0.72	0.95	0.86	1.92	0
ε	0	0.0328	0.0376	0.0337	0.0332	0.0235	–
ε_r(%)	0	14.82	16.99	15.23	15.00	10.60	–
time $t = 200$							
$\bar{x}$	0.3622	0.3580	0.3758	0.3650	0.3588	0.3745	0.3935
σ	0.2044	0.2014	0.0870	0.1238	0.1157	0.1868	0
μ ($\times 10^2$)	0.61	0.47	0.12	0.20	0.16	0.60	0
ε	0	0.0298	0.0324	0.0306	0.0311	0.0224	–
ε_r(%)	0	8.23	8.94	8.44	8.60	5.98	–
time $t = 300$							
$\bar{x}$	0.4890	0.4800	0.4997	0.4932	0.4838	0.4971	0.5276
σ	0.1406	0.1385	0.0621	0.0939	0.0874	0.1509	0
μ ($\times 10^2$)	0.48	0.41	0.09	0.17	0.14	0.49	0
ε	0	0.0192	0.0235	0.0216	0.0201	0.0165	–
ε_r(%)	0	4.00	4.70	4.38	4.15	3.40	–

Table 6.1: Simulation results at times $t = 100$ s, $t = 200$ s and $t = 300$ s.

- the standard deviation σ_z within zone z according to equation:

$$\sigma_z = \left(\frac{1}{nc_z} \sum_{i=1}^{nc_z} (x_{iz} - \bar{x_z})^2 \right)^{0.5} . \tag{6.6}$$

  where x_{iz} is species 2 mass fraction computed by the CFD simulation at cell i in zone z. If zone z is a perfectly mixed region, then $\sigma_z = 0$.

- the difference Δ_z between the maximum and the minimum mass fraction value computed by the CFD simulation within zone z.

6.2.2 Results

Simulation results are compared at times $t = 100$ s, $t = 200$ s, $t = 300$ s. For longer simulation times, differences close up since $x \rightarrow 1$ throughout the domain.

Table 6.1 contains the calculated test assessment criteria values for all the simulations. The average $\bar{x}$ always presents similar values, although Case 2 gives a sensibly higher value reflecting an over-estimation of the residence time. This is confirmed by the values of the standard deviation and the third moment: low values in Case 2 express a higher grade of mixing than in the CFD simulation. The CFD case suggests that blending occurs quite slowly. The fact that the perfect mixing case based on eqn. (6.5) seems to present an initial behaviour which is more similar to the CFD case should

not change our analysis. As plots will clarify, this is due to the fact that poor mixing immediately channels the tracer towards the outlet. Thus, at the beginning of the simulation, the amount of tracer within the tank is similar to a perfect mixing example, which also assumes (although for the opposite reason) that tracer is detectable in the outlet composition from the very beginning. Nonetheless, while simulation proceeds, the perfect mixing example starts diverging from the CFD case, while all zone network simulations more correctly approximate the rigorous average composition.

The best accuracy comes from case 5, i.e. by a simulation using the concentration distribution at several time steps to establish the zone network. The zone model appears to reasonably predict the concentration distribution and evolution along the mixing process. Among the other cases, only Case 1 seems to properly describe the segregation phenomena and it is the only example (apart from case 5) presenting a level of flow symmetry (see μ) which is closer to that of the CFD simulation. Cases 2÷4 produce simulation results showing high differences in the distribution shape, which appears to be much more regular than computed from the CFD simulation. As mentioned, simulation results become more precise as time goes on and mixing increases (see the relative error ε_r).

Figures 6.4 and 6.5 give a visual comparison of the 6 (CFD plus the 5 cases) simulations at time $t = 200$ s. Figure 6.4 depicts a section of the domain which intersects the inlet (x-coordinate = 9), while the section in Figure 6.5 intersects the outlet (x-coordinate = 2). These plots allow a good evaluation of the zoning capabilities of the algorithms proposed and confirm the information derived by numerical results in Table 6.1. Cases 2÷4 clearly result in too high a degree of mixing. Although they are capable of reproducing the general shape of the distribution (especially in the inlet area), the resulting mixing simulation is definitely incorrect (notwithstanding some improvement after using algorithms $\Theta_{grad\Delta}$ and Θ_H). On the other hand, it appears that a simple and generic algorithm like Θ_{geom} may achieve a better simulation of the phenomena in this case. It is evident that the choice of unsuitable properties may lead to zone networks which do not properly describe the phenomena of interests. The eddy dissipation rate (used in cases 2÷4), although often used to determine different mixing regimes in a vessel, may lead to a very imprecise (although qualitatively correct) solution of mixing phenomena. On the contrary, case 5 confirms that a good choice of the property set **P** will lead to very good results in representing the phenomena of interest.

The zone model obtained from mixing pattern data (case 5) is also used to observe the evolution of the concentration distribution for a longer period. Figures 6.6 ÷ 6.9

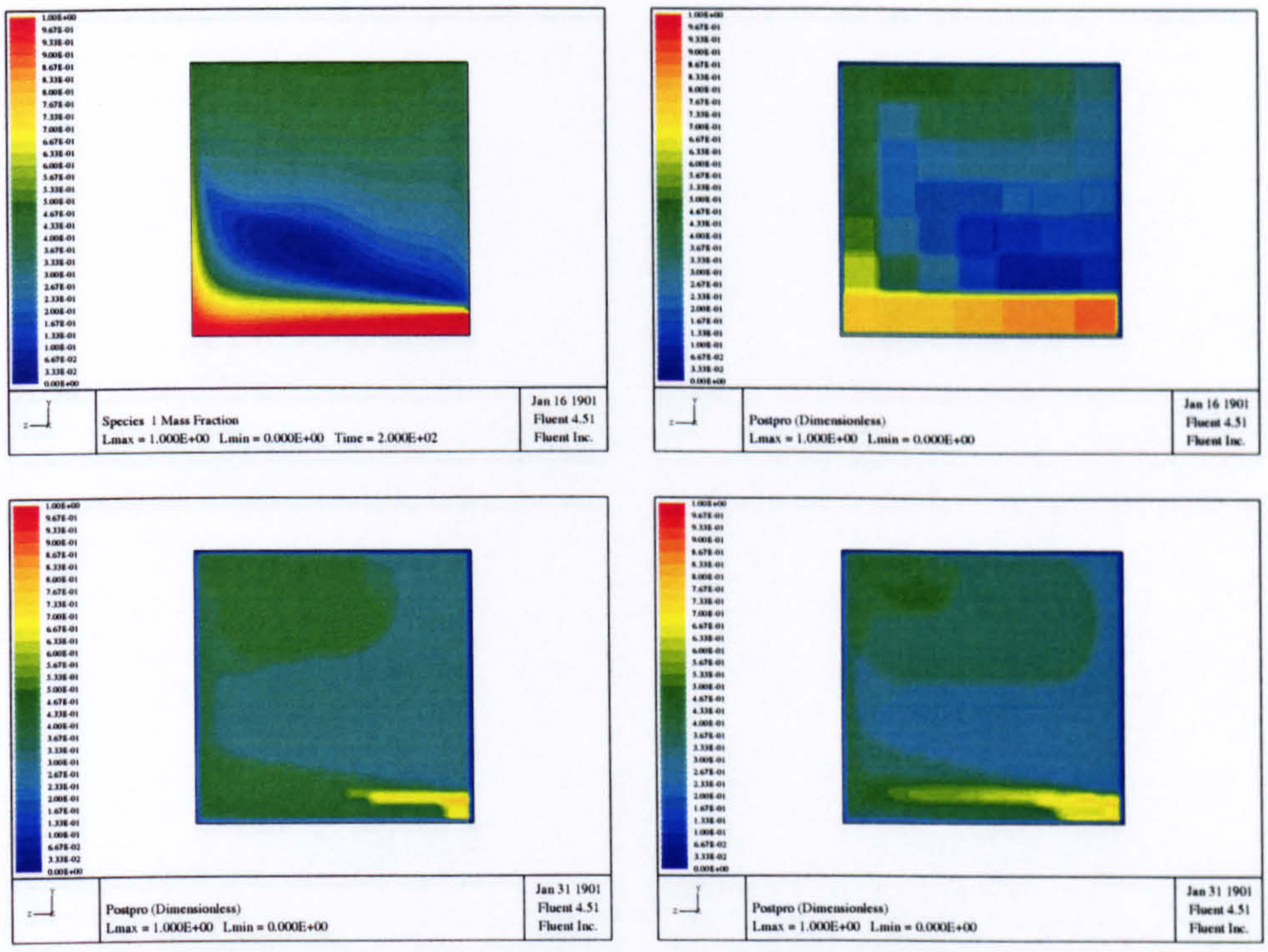


Figure 6.4: Time $t = 200$ s. Slice close to the inlet. At the top left CFD simulation; then from left to right: case 1, case 2 and case 3; case 4 and case 5.

compare the CFD simulation with the zone model at times $t = 100$ s, $t = 200$ s, $t = 300$ s and $t = 600$ s. Each figure compares the rigorous CFD simulation to the integrated model at two different x-coordinates. Table 6.2 compares the same simulations using the criteria defined in § 6.2.1 at times $t = 100$ s, $t = 200$ s, $t = 300$ s and $t = 600$ s.

Both plots and criteria demonstrate a good agreement between the CFD simulation and the results based on the zone model. The error is greater at time step $t = 100$ s, but that is rather expected because:

- at the beginning, the mixing pattern presents sharp gradient and a very localised composition distribution: the gPROMS-CFD model contains some large zones which do not properly handle such a situation;

- within vector $\Delta \mathbf{x}$, tolerances at $t = 100$ s and $t = 200$ s are larger and, accordingly,

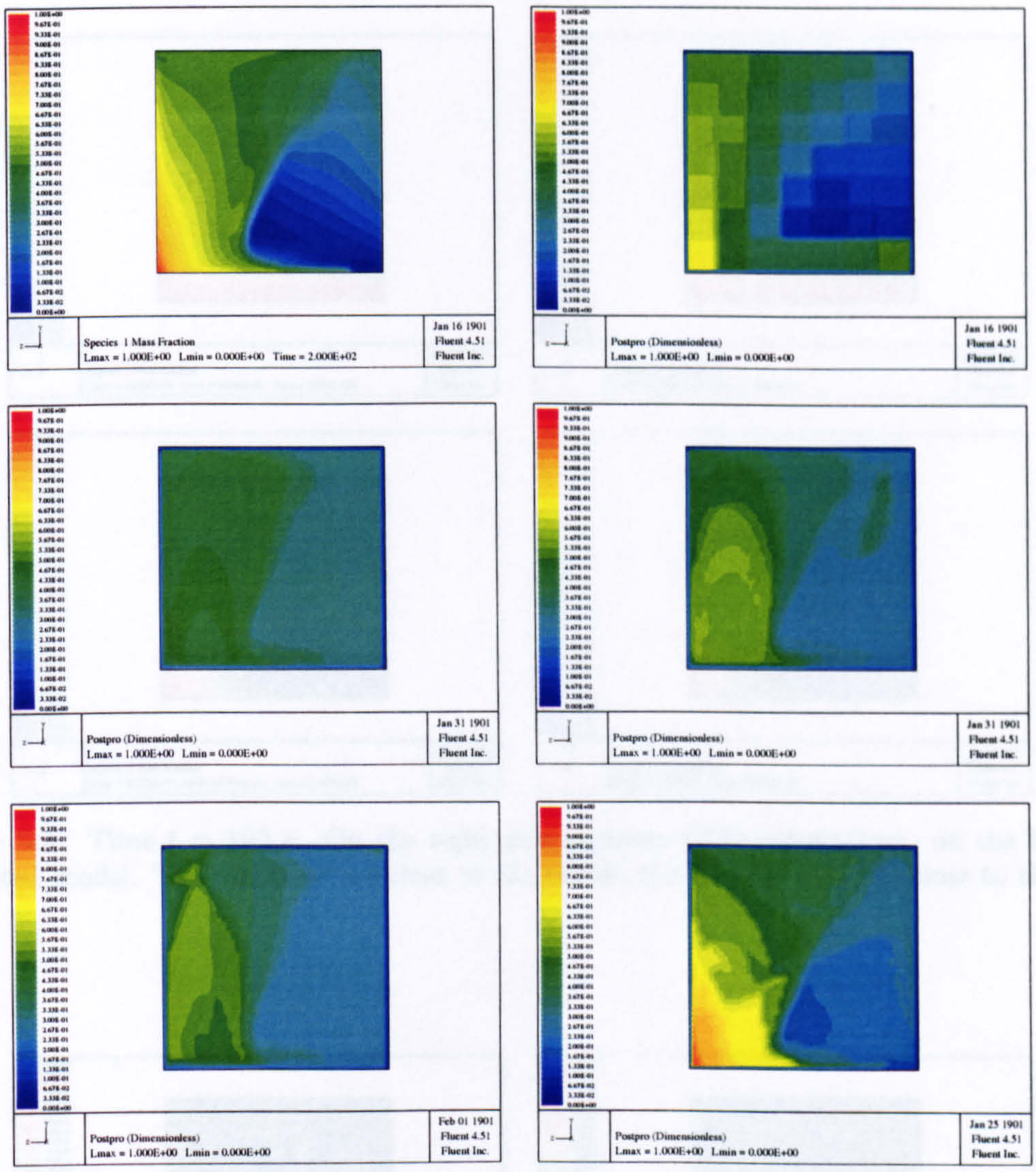


Figure 6.5: Time $t = 200$ s. Slice close to the outlet. At the top left CFD simulation; then from left to right: case 1, case 2 and case 3; case 4 and case 5.

	CFD	$t = 100$ sec
$\bar{x}$	0.2213	0.2192
σ	0.2514	0.2203
μ $(\times 10^2)$	1.93	1.92
ε	0	0.0235
$\varepsilon_r(\%)$	0	10.6

	CFD	$t = 200$ sec
$\bar{x}$	0.3622	0.3745
σ	0.2044	0.1868
μ $(\times 10^2)$	0.61	0.60
ε	0	0.0224
$\varepsilon_r(\%)$	0	5.98

	CFD	$t = 300$ sec
$\bar{x}$	0.4890	0.4971
σ	0.1406	0.1509
μ $(\times 10^3)$	4.80	4.94
ε	0	0.0165
$\varepsilon_r(\%)$	0	3.4

	CFD	$t = 600$ sec
$\bar{x}$	0.7366	0.7386
σ	0.078	0.079
μ $(\times 10^3)$	0.69	0.66
ε	0	0.0111
$\varepsilon_r(\%)$	0	1.5

Table 6.2: Simulation results at different time steps.

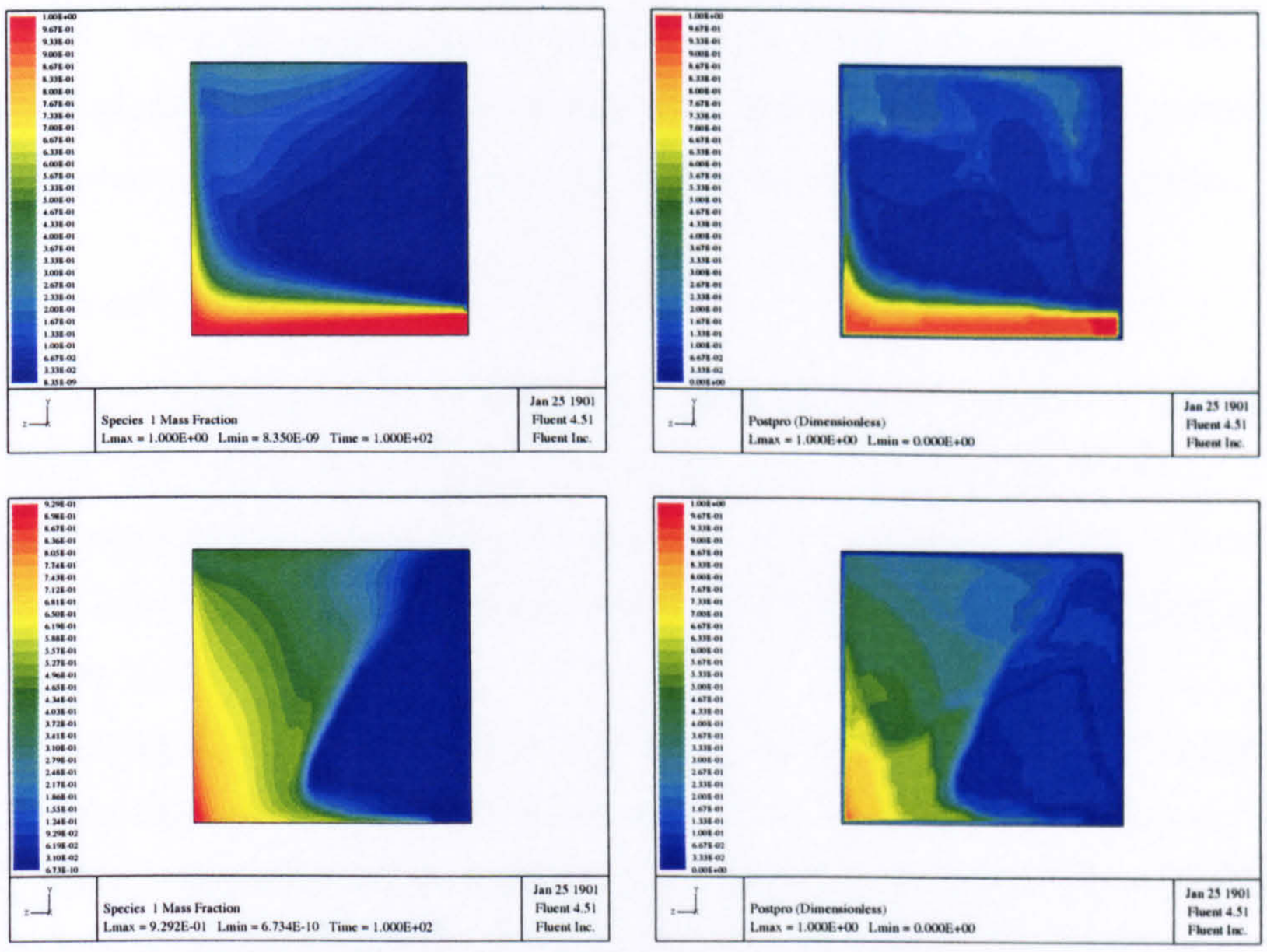


Figure 6.6: Time $t = 100$ s. On the right side rigorous CFD calculations; on the left side integrated model. The top slices are close to the outlet; the bottom ones are close to the inlet.

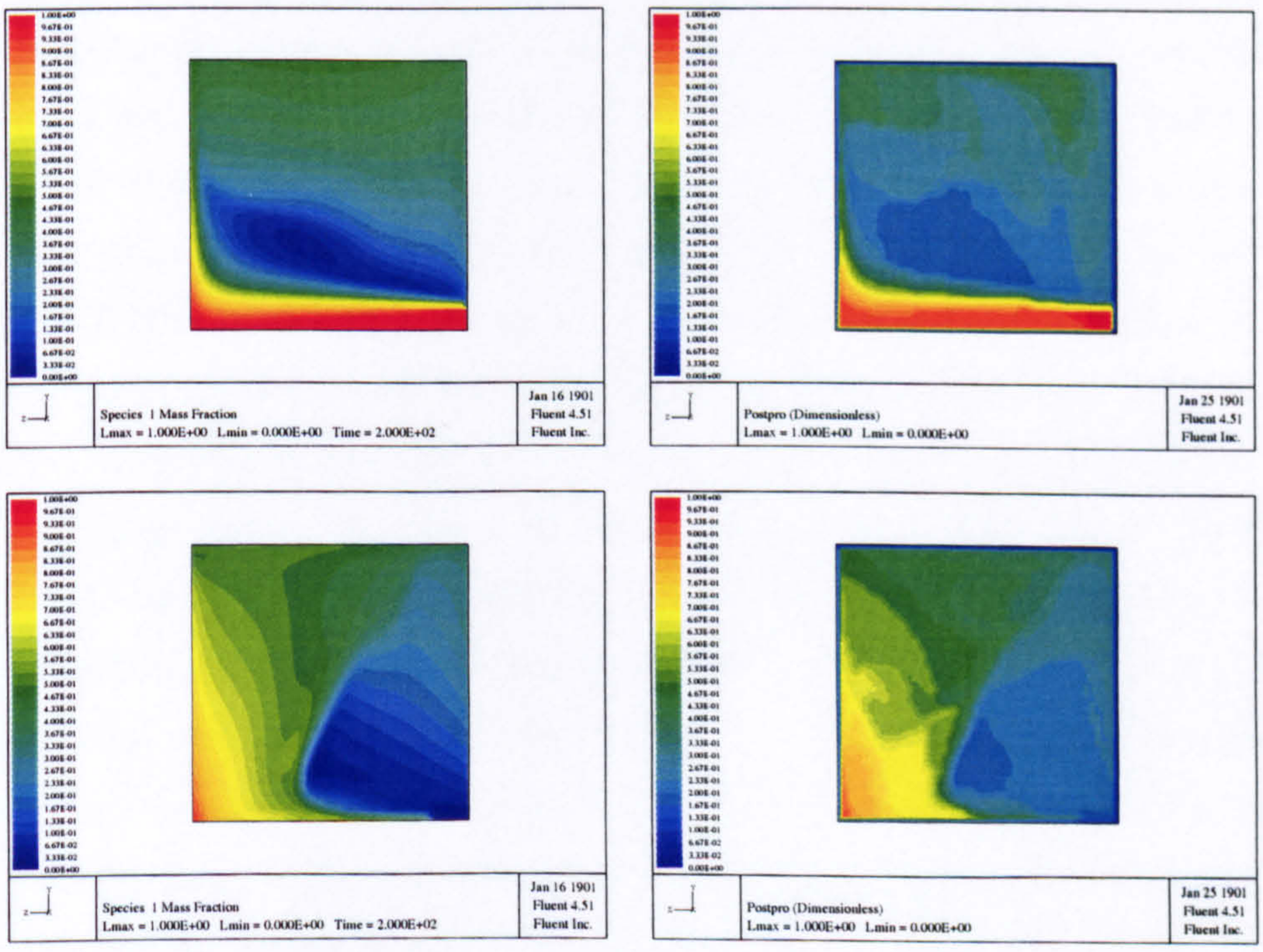


Figure 6.7: Time $t = 200$ s. On the right side rigorous CFD calculations; on the left side integrated model. The top slices are close to the outlet; the bottom ones are close to the inlet.

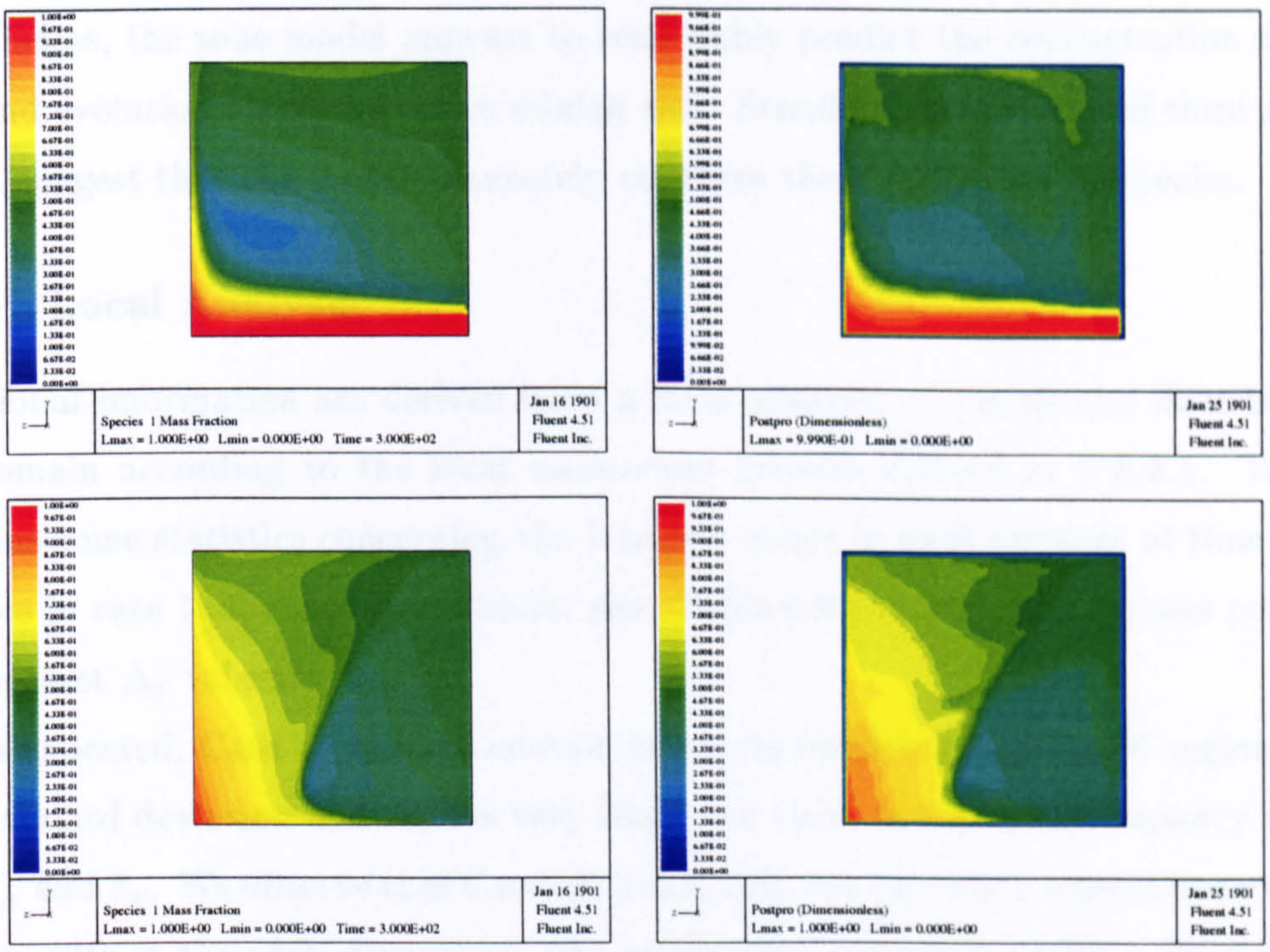


Figure 6.8: Time $t = 300$ s. On the right side rigorous CFD calculations; on the left side integrated model. The top slices are close to the outlet; the bottom ones are close to the inlet.

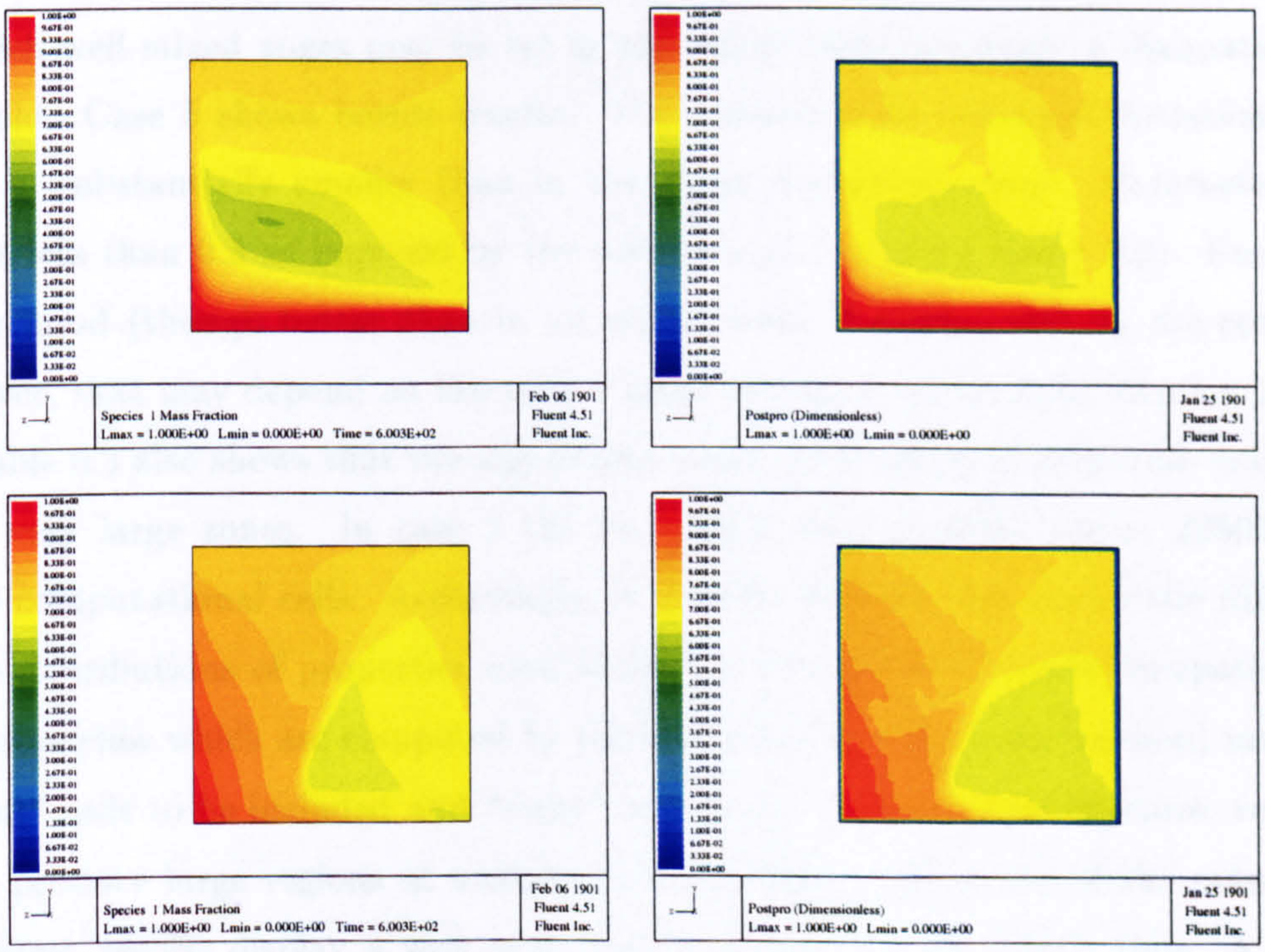


Figure 6.9: Time $t = 600$ s. On the right side rigorous CFD calculations; on the left side integrated model. The top slices are close to the outlet; the bottom ones are close to the inlet.

the capability of capturing species distribution is diminished.

Nonetheless, the zone model appears to reasonably predict the concentration distribution and evolution along the entire mixing test. Standard deviation and third moment values suggest that the model accurately captures the distribution of species.

6.2.3 Local Analysis

Additional information are derived from a local analysis of the species distribution in the domain according to the local assessment criteria defined in § 6.2.1. Table 6.3 contains some statistics concerning the 5 largest zones in each network at time $t = 200$ s. Since in case 1 all zones have similar size, Table 6.3 considers the 5 zones presenting the greatest Δ_z values.

As expected, Case 1 does not contain zones representing well-mixed regions. Both the standard deviation and Δ_z are very large and there is a great discrepancy between $\bar{x}_{CFD,z}$ and $\bar{x}_z$. We observe that Cases 2, 3 and 4 do not represent a great improvement: homogeneity in the eddy dissipation rate values does not imply uniform concentration. The σ_z values indicate a wide distribution within the region. That is confirmed by Δ_z values: we can observe that a single zone contains cells that have very different composition values. Similar considerations may be drawn by observing the $\bar{x}_{CFD,z}$ and $\bar{x}_z$ values in a zone. It is obvious that, at least in this instance, the principle according to which well-mixed zones may be set as regions of uniform energy of dissipation does not hold. Case 5 shows better results. The concentration standard deviation within zones is substantially smaller than in the other simulation cases. Parameter Δ_z is always less than 0.3 as imposed by the tolerance at $t = 200$ s (see § 6.2). Results are not as good (though better than in all other cases) if $\bar{x}_{CFD,z}$ and $\bar{x}_z$ are compared. However, that may depend on the rather large tolerance set for time step $t = 200$ s.

Table 6.3 also shows that the algorithms based on property distribution may create some very large zones. In case 2 the two larger zones contain about 22500 out of 25000 computational cells. Accordingly, it is quite obvious that even little differences in the distributions of properties used to set the zone network (eddy dissipation rate) and properties which are computed by the integrated model (mass fraction) may cause "wrong" cells to be included and "right" ones to be excluded. Furthermore, very large zones produce large regions at uniform concentration: that is one of the reasons why simulation results display a very well-developed blending after little time. A method to reduce the size of zones without amplifying their number will be suggested in § 6.5.

Case 1	nc_z	$\bar{x}_{CFD,i}$	$\bar{x}_i$	σ_z	Δ_z
zone 1.1	384	0.1994	0.5238	0.2716	0.9893
zone 1.2	336	0.6828	0.8952	0.3962	0.9996
zone 1.3	336	0.2560	0.4212	0.2395	0.9136
zone 1.4	336	0.2877	0.3790	0.2178	0.8473
zone 1.5	336	0.3349	0.3702	0.1983	0.7966
Case 2	nc_z	$\bar{x}_{CFD,i}$	$\bar{x}_i$	σ_z	Δ_z
zone 2.1	12237	0.2302	0.2998	0.1361	0.9936
zone 2.2	10327	0.4681	0.4320	0.1762	0.9960
zone 2.3	1302	0.5466	0.4840	0.1284	0.6828
zone 2.4	269	0.6086	0.4699	0.0999	0.3551
zone 2.5	154	0.6837	0.7927	0.1641	0.6711
Case 3	nc_z	$\bar{x}_{CFD,i}$	$\bar{x}_i$	σ_z	Δ_z
zone 3.1	8916	0.3730	0.2701	0.1369	0.941
zone 3.2	8472	0.1846	0.2534	0.1180	0.2740
zone 3.3	3474	0.5068	0.4467	0.1757	0.8584
zone 3.4	1274	0.5828	0.4620	0.1455	0.6832
zone 3.5	586	0.5560	0.5013	0.1211	0.6449
Case 4	nc_z	$\bar{x}_{CFD,i}$	$\bar{x}_i$	σ_z	Δ_z
zone 4.1	7193	0.2709	0.2500	0.1481	0.8971
zone 4.2	6108	0.2278	0.3248	0.1261	0.7349
zone 4.3	5305	0.4524	0.4058	0.1831	0.9079
zone 4.4	2258	0.5969	0.5155	0.1852	0.8148
zone 4.5	677	0.5556	0.4146	0.1306	0.6117
Case 5	nc_z	$\bar{x}_{CFD,i}$	$\bar{x}_i$	σ_z	Δ_z
zone 5.1	3760	0.1336	0.2083	0.0492	0.2992
zone 5.2	3443	0.4967	0.4358	0.0300	0.2006
zone 5.3	2770	0.3080	0.2511	0.0679	0.2997
zone 5.4	1662	0.4838	0.4684	0.0384	0.1805
zone 5.5	1642	0.4967	0.5453	0.0565	0.2570

Table 6.3: Simulation results at time $t = 200$ s.

6.3 A batch example

In this section we will investigate the case of a batch system, the double impeller stirred tank reactor described in chapter 2. The CFD grid is constituted by 27000 computational cells representing a quarter of a reactor. It is assumed that geometrical symmetry ensures periodical flow behaviour. Cyclic boundary conditions are assigned in order to diminish the computational burden by simulating only a sector of the whole geometry. Special *cyclic* cells are set to identify opposite planes across which the flows in the computational model are identical. Figure 6.10 illustrates an application of cyclic boundary conditions in a 2D example. The periodical swirling flow may be computed through a grid meshing a quarter of the geometry. The flow entering the computational model through one cyclic plane is identical to the flow exiting the domain

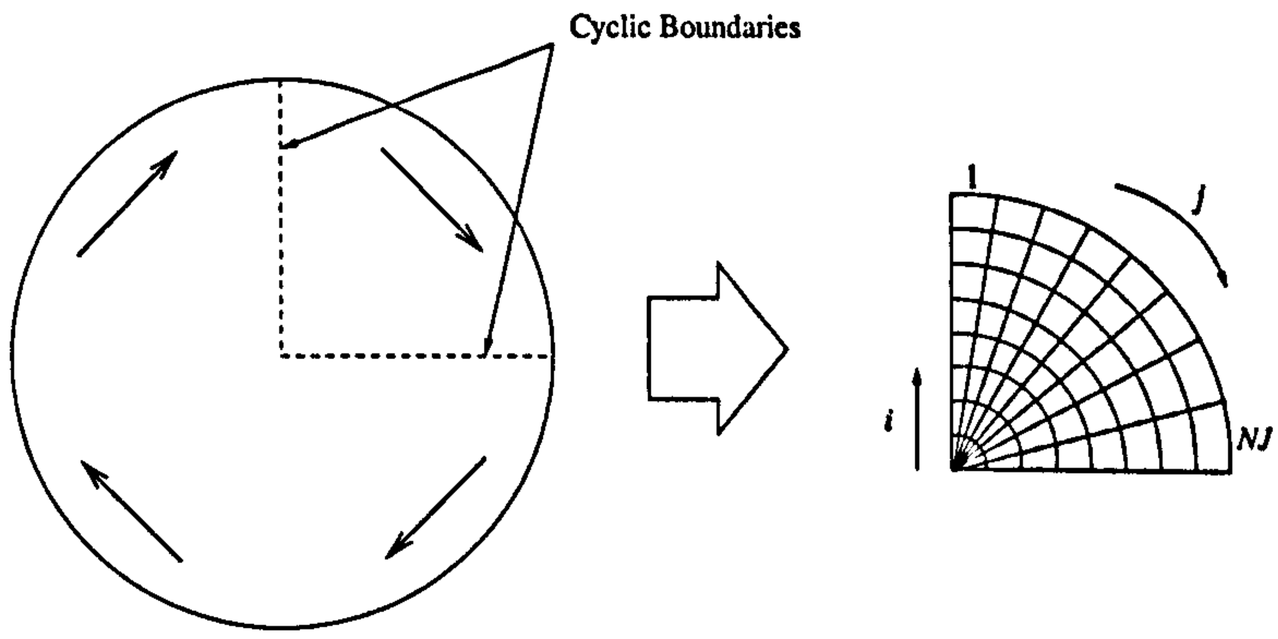


Figure 6.10: Use of cyclic boundaries.

through the opposite cyclic plane. Cyclic planes are always used in pairs as illustrated in this example. The CFD package treats the flow at a cyclic boundary as though the opposing cyclic plane is a direct neighbour of the live cells at the first cyclic boundary. Thus, when calculating the flow through the cyclic boundary adjacent to a live cell, the CFD package invokes the flow conditions at the live cell adjacent to the opposite cyclic plane.

Special procedures are implemented in the zoning algorithms to recognise cyclic boundary conditions. With reference to Figure 6.10, if cells $j = 1$ and $j = NJ$ belong to the same zone, then mass flowrates at the cyclic boundaries are not considered (i.e. they are internal flowrates); otherwise an interface is established between the two zones, and flowrates are computed as between neighbouring zones.

Zones were established in the batch reactor by means of a tracing experiment. However, the nature of results of tracing experiments depends on the location of the injection points. Two experiments were taken into account. As illustrated in Figure 6.11, first (experiment 1) the tracer is injected close to the double impeller in a highly turbulent region of the tank; then (experiment 2) it is injected in the bottom part of the tank underneath the impeller, i.e. in a poorly mixed region of the equipment. Impeller rotation speed is set at 300 rpm in both cases. The CFD mixing simulations demonstrate that in the first case the tracer is immediately dissolved into the bulk and a uniform concentration is obtained in a very short time throughout the domain. The second case exhibits a rather different behaviour. The tracer is trapped into the bottom region and blending is much slower. As a result, composition in the tank is always very uniform, except for the region under the impeller, which takes a longer time to reach the same concentration as the rest of the tank.

The network of zones is obtained using algorithm Θ_Δ. In experiment 1, the set of properties $\mathbf{P}$ is constituted of the mass fraction of species 2 at time steps $t = 2$ s

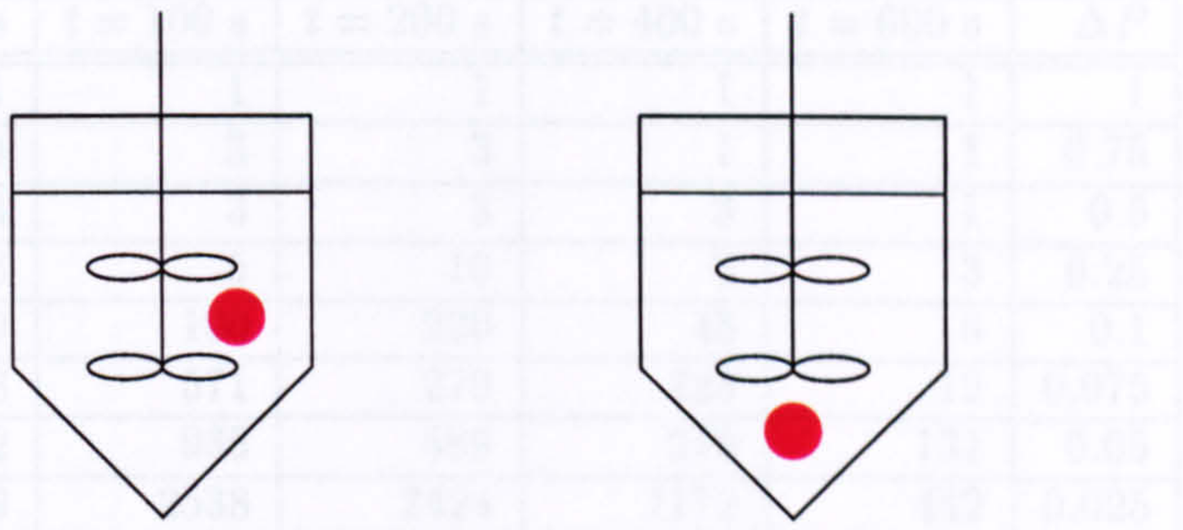


Figure 6.11: Injection points.

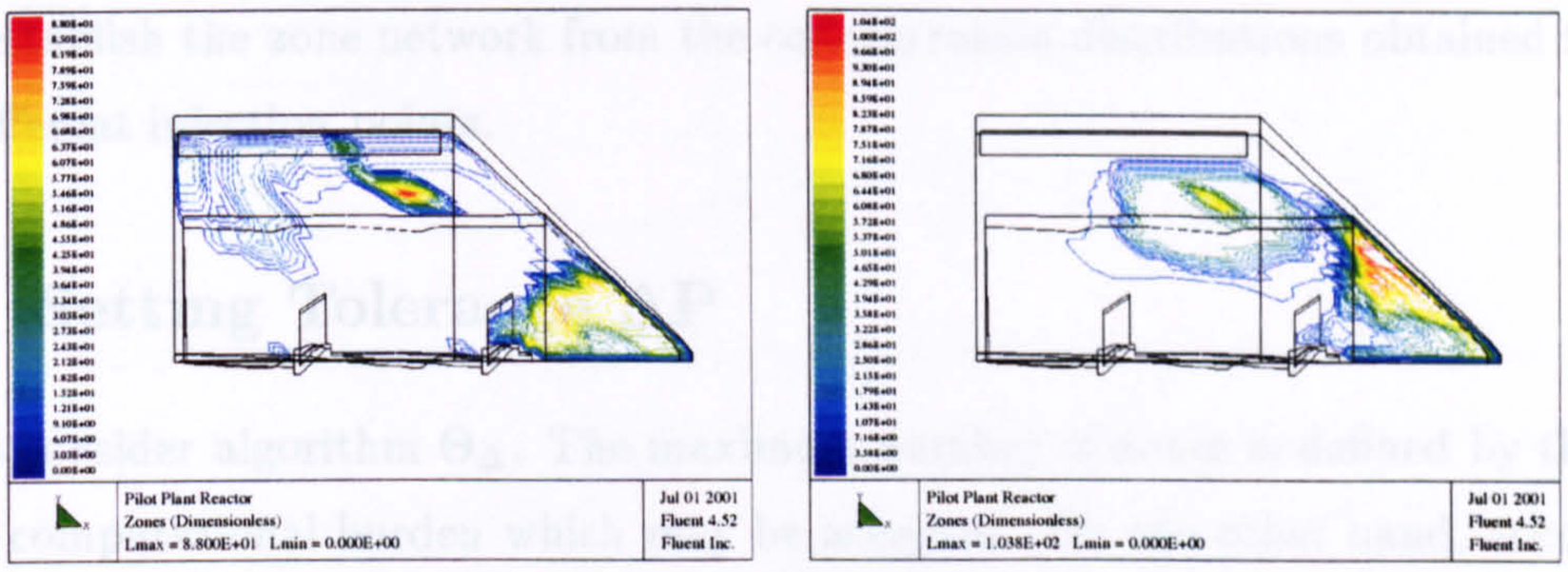


Figure 6.12: Zones obtained from mixing experiments at two different injection points.

and $t = 4$ s. The tolerance is set to $\Delta\mathbf{x} = [0.002, 0.00035]$. In experiment 2, the set of properties $\mathbf{P}$ is constituted of the mass fraction of species 2 at time steps $t = 3$ s and $t = 6$ s. The tolerance is set to $\Delta\mathbf{x} = [0.01, 0.007]$. The tolerance is higher in the latter case, because the more difficult mixing process produces higher gradients under the impeller.

Figure 6.12 shows the two zone networks corresponding to experiments 1 and 2 (100 and 104 zones were obtained, respectively). As predictable, the network resulting from experiment 1 better describes mixing phenomena in the central and top parts of the reactor, while zones from experiment 2 outline the mixing pattern in the bottom of the tank. Nonetheless, the general shape is rather similar and we can see that both networks detect what seem to be the critical regions in the reactor, i.e. the region underneath the impeller and a recirculation area situated in the central part of the reactor close to the walls. Although not identical, networks which are produced after a sufficient time of mixing may correctly detect the basic mixing behaviour, even if the tracer has been injected at different locations. Accordingly, we can state that in this case the injection zone is not critical to determine the zone network. An alternative route would be to carry out more than one tracing experiment (for example, experiments 1 and 2) and

$t = 50$ s	$t = 100$ s	$t = 200$ s	$t = 400$ s	$t = 600$ s	ΔP
1	1	1	1	1	1
4	3	3	1	1	0.75
4	3	3	3	1	0.5
6	5	10	3	3	0.25
169	190	220	45	6	0.1
398	571	279	228	12	0.075
872	935	589	279	131	0.05
2279	2538	2424	1172	442	0.025

Table 6.4: Number of zones with respect to ΔP.

then establish the zone network from the concentration distributions obtained from all the different injection points.

6.4 Setting Tolerance ΔP

Let us consider algorithm Θ_Δ. The maximum number of zones is defined by the maximum computational burden which may be accepted. On the other hand, accuracy in the simulation results depends on the tolerance ΔP. The smaller it is, the better the zone network is capable of capturing the distribution of properties **P**. The problem can be formulated as follows: *given* a maximum number of zones nz_{max}, *compute* the minimum tolerance $\Delta\mathbf{P}'$ producing a number of zones nz such that $nz \leq nz_{max}$.

If there is only one property, $\mathbf{P} = P$, tolerance $\Delta\mathbf{P}' = \Delta P'$ can be estimated by observing the general behaviour of the function $nz = \Theta(\Delta P)$, which, of course, may assume only integer values. Table 6.4 demonstrates how the number of zones vary with respect to ΔP for the mixing tracing experiment in the box tube illustrated in Figure 6.3. ΔP is expressed in terms of the mass fraction of species 2. The mass fraction distribution is considered at time steps $t = 50$ s, $t = 100$ s, $t = 200$ s, $t = 400$ s and $t = 600$ s. The number of zones appears to be approximately in inverse proportion to the value of ΔP. However, a closer analysis reveals that the behaviour of the function $nz = \Theta(\Delta P)$ is not monotonic. Locally, the relation is more complex and presents fluctuations which are not easily predictable. Table 6.5 contains some data which shows this irregular behaviour for $0.075 \leq \Delta P \leq 0.05$ at $t = 600$ s. Such behaviour is explained by the fact that the ΔP value affects the shape of the zones and the path followed by the algorithm to build the zones.

The choice of an appropriate $\Delta\mathbf{P}'$ is even more complex when several properties are taken into account. For instance, let us consider one property P_1. Let us set ΔP_1 such

ΔP	N $(t = 500$ s)
0.075	12
0.072	13
0.070	103
0.068	89
0.066	33
0.064	20
0.062	115
0.060	170
0.058	100
0.056	166
0.054	122
0.052	164
0.050	131

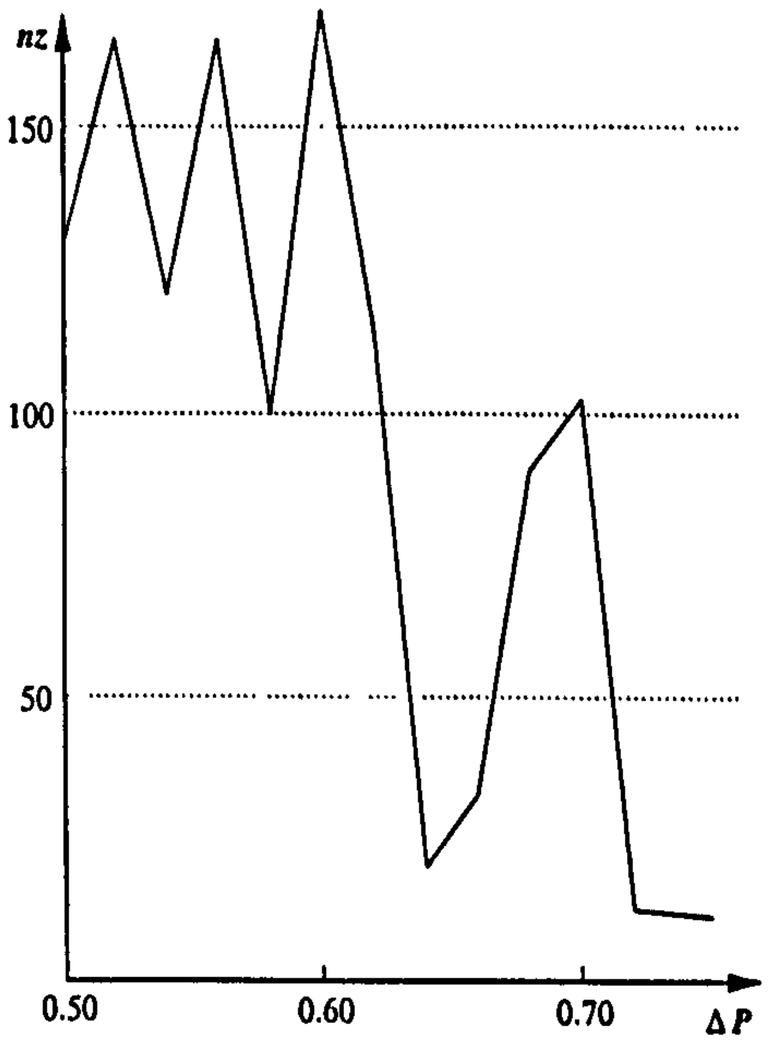


Table 6.5: Number of zones with respect to ΔP in simulation at $t = 600$ s.

that

$$nz_1 = \Theta_\Delta(\Delta P_1) > 1.$$

Then let us take a second property P_2 and a value of ΔP_2 such that

$$nz_2 = \Theta_\Delta(\Delta P_2) = 1,$$

i.e. let us suppose that $P_2(c) - P_2(c') < \Delta P_2, \forall c, c' \in \mathcal{C}$.
Finally, let us consider $\mathbf{P} = [P_1, P_2]$, $\Delta\mathbf{P} = [\Delta P_1, \Delta P_2]$ and compute

$$nz_{12} = \Theta_\Delta(\Delta\mathbf{P}).$$

In general, we will obtain

$$nz_{12} \neq nz_1.$$

Several tests on algorithm Θ_Δ have demonstrated that adding one property to the $\mathbf{P}$ array will always affect the result of the algorithm independently of tolerance $\Delta\mathbf{P}$. In fact, the search path in Θ_Δ is affected by all components of array $\mathbf{P}$ (see § 5.3.2).

Case 2	Case 3	Case 4	Case 5
43	56	66	46
42%	58%	65%	44%

Table 6.6: Number of single-cell zones.

6.5 A Practical Method to Diminish the Number of Zones

It was noted that the methods (with the exception of Θ_{geom}) to define a zone network will create few very large zones and a great number of tiny zones. This is due to the fact that

a. there may exist localised regions within which properties **P** exhibit sharp gradients (and, on the other hand, there may be large regions with a rather uniform property distribution);

b. tiny groups of cells among larger zones may be left out from adjacent zones so as to remain independent.

Table 6.6 contains the number of single-cell zones in cases $2 \div 5$ of § 6.2.2 and their percentage with respect to the total number of zones. The number of single-cell zones is about 50 % of the total number of zones in the model. These tiny zones highly increase the complexity of the solution without giving much in terms of information. One reasonable approach is to define a *minimum* number of cells into which the domain may be divided. In other words, we set a discretisation bound defining a minimum size for a zone.

An algorithm was developed to reduce the number of zones by aggregating the single cell or small zones, which is executed after the original zoning algorithm has been performed. We define:

- the set of zones $\mathcal{Z}$
- the set $\mathcal{N}_z$ of zones which are neighbour to zone z, $\forall z \in \mathcal{Z}$
- the minimum number of cells per zone n_{min}

The procedure goes through steps:

a. consider zones one by one;

b. if a zone z contains $n_z < n_{min}$ cells, then consider neighbouring zones and select the smallest zone $z' \in \mathcal{Z}$;

c. merge zones z and z';

d. update set $\mathcal{Z}$.

The algorithm is structured as follows:

ALGORITHM AGGREGATE
Given minimum number of cells per zone n_{min}:

1. Initialization.
   a. $\forall z \in \mathcal{Z}$ compute number of zone cells n_z
   b. $\mathcal{Z}_i = \{c \mid Zc = z\}$
   c. $yes = 1$
2. WHILE $yes = 1$ DO
   a. $yes = 0$
   b. FOR each $z \in \mathcal{Z}$ DO
      I. IF $n_z < n_{min}$ THEN
         i. Select $z' \in \mathcal{N}_z \mid n_{z'} = \min(n_j \mid j \in \mathcal{N}_z)$
         ii. $n_z = n_z + n_{z'}$
         iii. $\mathcal{Z} = \mathcal{Z} \setminus \{z'\}$
         iv. $nz = nz - 1$
      END IF
      II. IF $n_z < n_{min}$ THEN
         $yes = 1$
      END IF
   END FOR
3. Stop.

The algorithm was tested for cases 2 and 5 of § 6.2.2 first aggregating any single cell zone (**case 2′** and **case 5′**) and then any zone of less than 5 cells (**case 2″** and **case 5″**). Results are in Table 6.7. It may be observed that the aggregation of small zones does not significantly affect the simulation results. On the other side, the smaller number of zones sensibly diminishes the simulation computational time, which in the mixing test example exclusively depends on gPROMS computations.

6.6 Key results

The main achievements illustrated in this chapter are:

- The definition of some practical procedures to apply the zoning methods defined in chapter 5. The procedures suggest which zoning method should be used and how it should be applied. They are based on:

	CFD	Case 2	Case 2′	Case 2″	Case 5	Case 5′	Case 5″
N	25000	102	67	45	103	66	46
μ	0.3622	0.3758	0.3757	0.3687	0.3745	0.3748	0.3757
σ	0.2044	0.0870	0.0868	0.0820	0.1868	0.1858	0.1801
μ_3 ($\times 10^3$)	6.10	1.17	1.13	1.02	9.68	9.51	8.62
ε	0	0.0324	0.0323	0.335	0.0224	0.0224	0.0228
ε_r (%)	0	8.94	8.59	9.09	5.98	5.98	6.07
CPU time (s)	–	1575	805	403	1533	780	382

Table 6.7: Comparison between simulations where zones are the results of algorithm Θ_Δ (case 2 and 5 of § 6.2.2) and simulations where subsequent aggregation of small zones is applied.

- ▷ manual adjustment when there is no clear knowledge of the interactions between physical and chemical phenomena;
- ▷ distribution of critical properties when the process depends on the distribution of an identifiable set of properties;
- ▷ identification of a mixing pattern when mixing is the critical hydrodynamic phenomenum in the process.

- The design and analysis of two *mixing tests* to compare different zoning applications. These tests demonstrate the importance of properties upon which the zone network is built up. An incorrect selection of properties may produce a network of zones unable to describe the critical phenomena occurring in the process, but a proper selection, based on prior tracing tests, results in a good simulation of the process phenomena.

- The definition of a method to decrease the number of zones and speed up calculations. The automatic zoning algorithms may produce a large number of very small zones: a procedure aggregating the smallest zones speeds up calculations by a factor of 2-4 without affecting the solution accuracy.

Chapter 7

Efficiency and Robustness

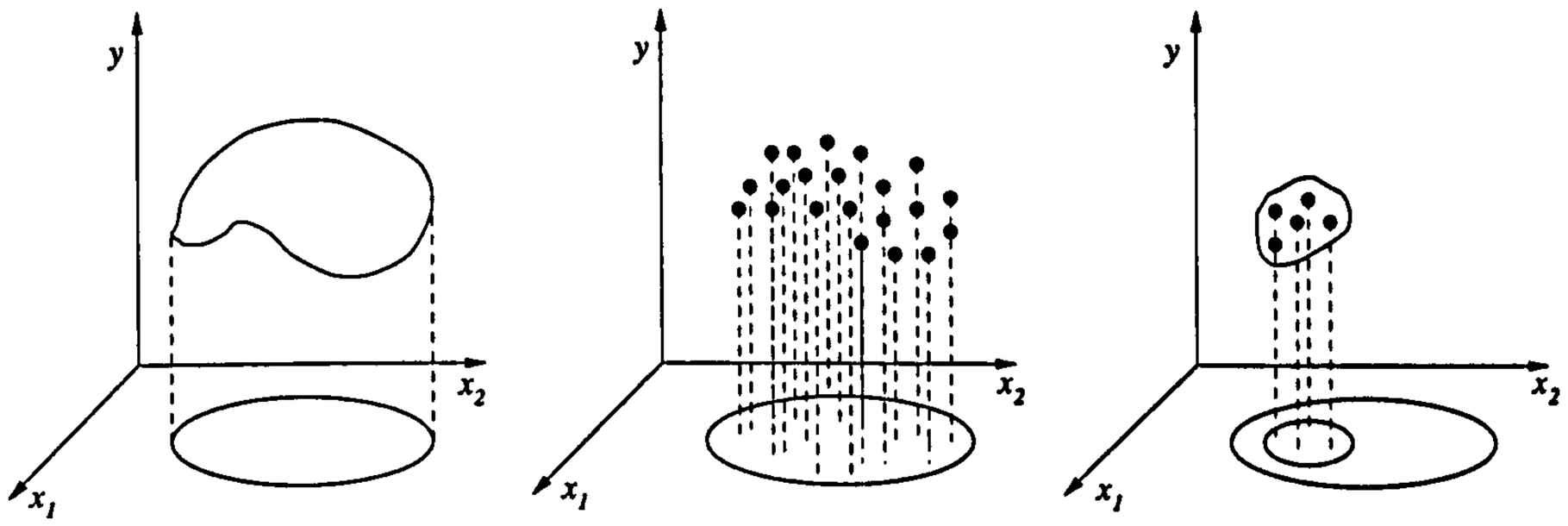
In this chapter we will focus our attention on two computational issues affecting the suggested integration approach: robustness and efficiency of calculations. In particular, efficiency is one of the main issues limiting the combined use of CFD and process simulation due to the computational burden of this type of approach. CFD calculations are time expensive and the need to continuously update hydrodynamics parameters with data deriving from process simulation calculations could make an integrated approach impractical.

As mentioned in chapter 2, a first practical method to speed up CFD calculations is to always consider a *hot start*: CFD solution data files are always saved so that, when updates are needed, calculations start from a set of results "close" to the ones which have to be computed. However, this does not decrease the number of CFD calls. A more efficient method is pursued in this chapter by adopting alternative and simplified fluid dynamics models instead of CFD calculations. These models are local approximations (*local models*) of the more rigorous and complex models embedded in the CFD packages and are aimed to minimise the need for full CFD calculations. These local models are "adjusted" to match a small number of rigorous CFD solutions.

7.1 Local Models: Introduction

CFD calculations are a very heavy computational burden, which may become unbearable in a dynamic simulation demanding continual updating. Some tests demonstrated that even simple models (e.g. example in chapter 2) may take days before a simulation is completed. The use of simple local models to substitute most, if not all rigorous calculations appears to be an appealing alternative technique to tackle the problem.

CFD calculations can be viewed as a very complex function whereby a vector of



Rigorous function
$\mathbf{Y} = F(\mathbf{x})$

Database

Local model
$\mathbf{y} = f(\mathbf{x})$

Figure 7.1: From rigorous models to local models.

outputs are calculated from a vector of inputs provided by the process simulation model. Let us define such a function as

$$\mathbf{Y} = \mathbf{F}(\mathbf{x}), \tag{7.1}$$

where $\mathbf{Y}$ and $\mathbf{x}$ are vectors of outputs and inputs, respectively. Let us suppose there is a database $\mathcal{D}$ storing a subset $\mathcal{Y}$ of outputs $\{\mathbf{Y}_1, \mathbf{Y}_1, \ldots, \mathbf{Y}_m\}$ and a subset $\mathcal{X}$ of inputs $\{\mathbf{x}_1, \mathbf{x}_1, \ldots, \mathbf{x}_m\}$ such that $\mathbf{Y}_1 = \mathbf{F}(\mathbf{x}_1)$, $\mathbf{Y}_2 = \mathbf{F}(\mathbf{x}_2), \ldots,$ $\mathbf{Y}_m = \mathbf{F}(\mathbf{x}_m)$.
We define a *local model* a function

$$\mathbf{y} = \mathbf{f}(\alpha, \mathbf{x}), \tag{7.2}$$

where

$$\alpha = \alpha\,(\mathbf{Y}_i \in \mathcal{Y}, \mathbf{x}_i \in \mathcal{X}) \tag{7.3}$$

are adjustable model parameters estimated from a set of rigorous inputs and outputs. The vector of outputs $\mathbf{y}$ has the same cardinality as vector $\mathbf{Y}$ in (7.1). Ideally, the local model (7.2) is able to approximate the behaviour of model (7.1) in a suitable ball of $\mathbf{x}$ and return an output $\mathbf{y} \approx \mathbf{Y}$.

Local models rely on the hypothesis that the results of a generic function $\mathbf{F}$ can be estimated by a different and simpler function $\mathbf{f}$ by means of adjustable parameters α able to adapt function $\mathbf{f}$ to the rigorous function $\mathbf{F}$. Such parameters must be estimated from rigorous calculations of function $\mathbf{F}$ and "tuned" to allow the local model (7.2) to represent the behaviour of the rigorous function (7.1). The rigorous model is usually

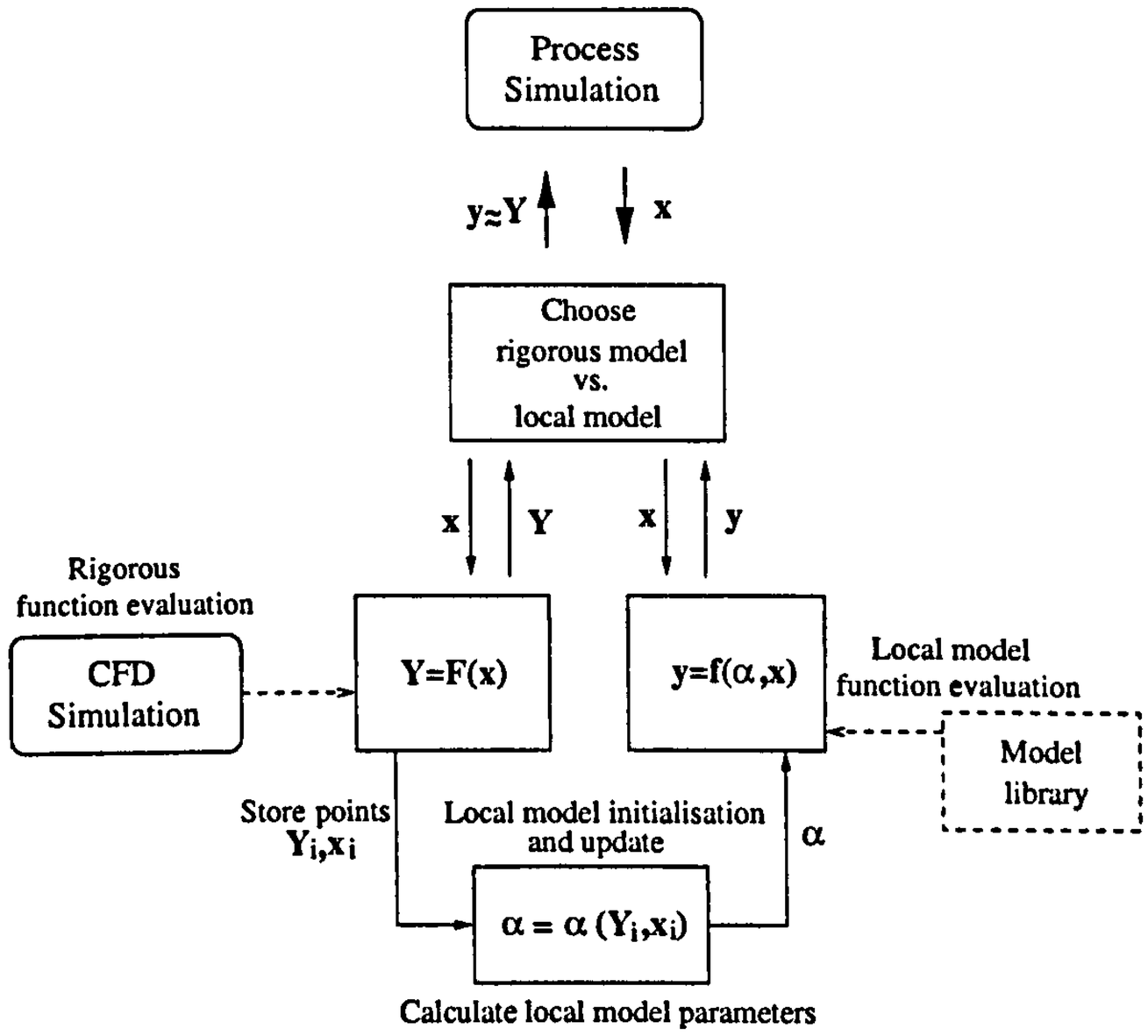


Figure 7.2: Information flow.

a very complex function depending on several variables. Database $\mathcal{D}$ containing input and output pairs is a discrete representation of the function map. For the purpose of fitting parameters $\boldsymbol{\alpha}$, the local models may use a subset of the discrete representation to approximate a portion of the function map (Figure 7.1).

The minimum number of points ($\mathbf{x}$, $\mathbf{Y}$ pairs) which are needed to estimate the parameters in a local model depends on the local model itself, i.e. the number of parameters in the model. To estimate n parameters, at least n points are needed for an interpolation algorithm, although estimation techniques such as the Least Square (LS) method allow a greater number of data points to be used.

Figure 7.2 summarises the flow of information required and the fundamental issues of the method which will be discussed in this chapter:

- ▷ definition of local models (chosen from a *library*);
- ▷ initialisation of local models;
- ▷ local model updating:
  - storage of rigorous data
  - selection of data points for parameter estimation
  - parameter estimation;

▷ choosing a model: rigorous vs. local to ensure

- accuracy
- continuity.

Local models which are presented are first developed for a single zone model. Their application to a network of zones is then discussed in § 7.6.

7.1.1 Literature Review

The benefits of using local models instead of a more expensive rigorous model have been widely investigated to simplify thermodynamics calculations within process simulation. In fact, it was noted that the calculation of physical properties of pure components and mixtures treated in a plant used to contribute a major share of the total cost of the computer time.

Leesley and Heyen (1977) developed a first interpolation scheme to avoid complex algorithms to estimate the equilibrium constant K_i for component i in vapour-liquid equilibrium calculations. Instead of using rigorous calculations based on fugacity and activity coefficients calculations, a simplified model (based on some physical assumption) was suggested:

$$\log K_i = A_{1,i} \log P^s(T) + A_{2,i} - \log P \tag{7.4}$$

where P^s is the known vapour pressure of a reference component and $A_{1,i}$ and $A_{2,i}$ are the parameters to be estimated. Parameters in the local models are estimated from two sets of values (K_i, P, T) from a data bank. In fact, the aim of this work was not to predict K_i values, but to interpolate them. Barrett and Walsh (1979) improved the method flexibility by developing an algorithm capable of setting up model of enthalpies and fugacities in a dynamic manner. The algorithm is capable of

- assessing the correct local model to use at a particular point in the simulation;
- maintaining the accuracy of such models with respect to the rigorous thermodynamics calculation for a specified level of accuracy.

A database of local models describing different simulation conditions was created and each model was characterised by an error so as to define its domain of acceptable accuracy.

Chimowitz *et al.* (1983) and Macchietto *et al.* (1986) approached the problem of process design and simulation by proposing simple models locally approximating rigor-

ous and expensive TP property models. Parameters were re-calculated using recursive parameter updates (similarly to the method developed by Åström *et al.*, 1977 for self-tuning regulators). A modified least-squares procedure was adopted so that a single evaluation of a rigorous thermodynamic property permits a simultaneous evaluation of all the parameters in the local model.

Hillestad *et al.* (1989) focused their research on detecting adequate procedures to decide when to update the local model parameters. They suggested a general method to estimate the model error and to update parameters without causing the typical blow-up problem in the local approximations of the recursive numerical methods (their work was based upon a general procedure developed by Saelid *et al.*, 1985).

Ledent and Heyen (1994) adopted the approach in Macchietto *et al.* (1986), introducing a method to simplify local models by identifying unnecessary terms (and parameters). Støren and Hertzberg (1997) demonstrated the usefulness of local thermodynamics models to accelerate dynamic optimisation calculations. Special attention was given to avoid discontinuities in the local model solutions which may affect the integrator stability.

All the previous works adopt *physical local models* (see § 7.3), i.e. models estimating property values through a simplified representation of the physical phenomena affecting those properties. We investigated the possibility of developing *general local models* which are independent from the calculations involved. Apart from well-known linear estimation methods, there are several examples where non-linear models have been used to estimate parameters. We have taken into account only some of the possible techniques.

Iiguni *et al.* (1997) presented a non-linear adaptive estimation method where local approximation was achieved by means of a special kind of radial basis function, which is a well-known tool for non-linear interpolation in a multidimensional space. Radial basis functions theory has then been mostly developed by Powell (see review in Powell, 1992). There are several types of radial basis functions: a very important case is multiquadric interpolation, which was introduced by Hardy (1971) to represent topography from scattered data.

Another important class of non-linear estimation functions is the Shepard's interpolation (Shepard, 1968). As presented by the author, it does not have very good fitting properties, even if it demonstrates a high flexibility in treating interpolation for multivariable functions. However, Gordon and Wixom (1978), Franke and Nielson (1980) and Renka (1988) suggested some modifications making the method more efficient and

accurate. Some interesting applications of this method concern interpolation in CFD calculation to transfer solution data from one computational grid to another (Shen *et al.*, 1993) and the representation of potential energy surfaces in dynamic calculations for electronic structures by using a limited number of accurate data points (Nguyen *et al.*, 1995).

Other estimation methods such as Kalman filters or wavelet functions and other approximation methods (DeVore, 1998) have been considered during the thesis work, but then abandoned since they did not demonstrate additional properties with respect of radial basis functions or Shepard interpolation, at least for our application.

7.2 Parameter Estimation

The estimation of parameters for local models is the most difficult and delicate task of defining models capable of delivering a good approximation of rigorous functions. The rigorous thermodynamic models considered in the works mentioned in § 7.1.1 are still relatively simple if compared to CFD calculations and, furthermore, analytic derivatives with respect to inputs can be easily calculated. This allows the definition of robust procedures to evaluate local model errors and update parameters.

CFD calculations are extremely complex and time expensive. Derivatives for the property models cannot be obtained and, thus, an analysis to estimate and predict errors in the local model estimation is certainly more difficult.

The next subsections briefly consider some of the techniques to estimate the local model parameters.

7.2.1 Standard Least Square

The most important estimation technique is the classical least squares (LS) method. The basic linear least squares problem can be stated as follows: *given a real* $m \times n$ *matrix* $\mathbf{A}$ *of rank* $k \leq min(m,n)$ *and given a real* m*-vector* $\mathbf{b}$*, find a real* n*-vector* $\boldsymbol{\alpha}_0$ *minimizing the Eucledian length of* $\mathbf{A}\boldsymbol{\alpha} - \mathbf{b}$.

Solving a problem using the least square method requires the application of several techniques in order to produce a reliable and effective solution algorithm. It is necessary to store all rigorous data points $\mathbf{Y}_i$, $\mathbf{x}_i$. A minimum number of n linearly independent points is needed to initialise the method. For a more detailed analysis of the problem we refer to Lawson and Hanson (1995).

A detailed description of our implementation of the method and the techniques to treat nearly-singular matrices is presented in appendix A.

7.2.2 Recursive Least Squares

Recursive LS methods (RLS) (Soroush, 1998) are advocated when large data banks should be avoided. Only the latest set (usually one) of rigorous points is sufficient to update all parameters and no data bank is required. However, in our case the main problem of the recursive approach is that to define the variance-covariance matrix in a robust manner needs an initialisation step for which disturbances are introduced (Soroush, 1998). CFD calculations would make this initialisation phase too computationally expensive.

RLS will not be considered for parameter estimation in the local models presented in this work. More details about the method are available in the book of Bierman (1972).

7.2.3 Linear Interpolation

The LS algorithm may be used to solve interpolation problems. The problem $\mathbf{A}\boldsymbol{\alpha} = \mathbf{b}$ has always a unique solution, if $\mathbf{A}$ is a square $m \times m$ matrix of rank m. Although the standard LS method can be used to solve the problem, well-known basic Gauss elimination is preferred to solve the system by means of a **LU**-factorisation of matrix $\mathbf{A}$.

A procedure has also been implemented in order to avoid that set of suitable inputs $\mathcal{X}$ may contain couples of quasi linear dependent inputs, i.e., after defining a tolerance ε, the following relation has to be satisfied for each $\mathbf{x}_i, \mathbf{x} \in \mathcal{X}$

$$\|\mathbf{x}_i - a\mathbf{x}_i\| > \varepsilon, \forall a \in \mathbb{R}. \tag{7.5}$$

If matrix $\mathbf{A}$ is singular, then a different set of inputs $\mathbf{x}$ must be chosen from $\mathcal{X}$, if available, or new points $\mathbf{Y}_i$, $\mathbf{x}_i$ must be calculated.

Both LS and linear interpolation methods require an *initialisation* step to store the minimum number of points required for the estimation of the local model parameters. The initialisation step occurs at the beginning of the simulation or at any other time there are no rigorous CFD points available. The noise which typically affects CFD calculations may cause severe problems in the local model estimation capabilities as well as in the possibility to obtain a model solution. The issue will be discussed in § 7.5.2.

7.3 Physical Local Models

Physical models are not generic correlations. They represent the same phenomena as the rigorous models do, but in a simplified manner. The physical meaning of the terms in the model expression determines the predictive capacity of the model itself. As such, physical models must be used for well defined conditions and are not in general suitable for different situations.

Furthermore, the local model function should be explicit in order to assure speed of calculation. It is also very important that the parameters to be fitted to the rigorous model are linear in the expression, in order to ensure a fast, robust and consistent evaluation of the parameters (Perregaard, 1992). Two specific examples are described and used for some of the simulations presented in this thesis: a model for estimating the heat transfer coefficient and one for estimating the effective viscosity in a non-Newtonian fluid.

7.3.1 Example I: Heat Transfer Coefficient

The use of several physical models is proposed here to estimate the heat transfer coefficient h in the reactor model described in chapter 2. Index i will be used to refer to local property values, i.e. the value of the properties computed at the centre of a grid cell i by CFD calculations. Otherwise, properties are computed within a single zone.

The standard correlations to calculate the heat transfer coefficient in a stirred reactor (on the reactor side) have the form (e.g., Fletcher (1987)):

$$Nu = C(Re)^{\alpha}(Pr)^{\beta} \tag{7.6}$$

where:

- C is a coefficient dependent on the geometry of impeller and tank;
- Nu is the Nusselt number hD/λ with h, D and λ the heat transfer coefficient, tank diameter and fluid conductivity, respectively;
- Re is the Reynolds number $(dN^2\rho)/\mu$ with d, N, ρ and μ the impeller diameter, impeller rotation speed, fluid density and fluid viscosity, respectively;
- Pr is the Prandtl number $(C_p\mu)/\lambda$ with C_p the fluid heat capacity.

The formula can be linearised such that (Model HTC:Exp):

$$\log(Nu) = \gamma + \alpha\log(Re) + \beta\log(Pr) \tag{7.7}$$

where $\gamma = \log C$.

The model proved to return quite satisfactory results if parameters are estimated by means of the LS technique. A large number of points are needed to obtain a reasonable estimation of the parameters (see § 7.8.1). On the other hand, if linear interpolation is adopted to estimate the model parameters α, β and γ, the results are often rather poor. It also demonstrates a different sensitivity to the physical properties that are used to calculate the heat transfer coefficient with respect to the rigorous model. In particular the model is much more sensitive to density changes than the rigorous model.

An alternative local model having a closer correspondence to the rigorous one was developed by simplifying the rigorous model actually used to locally calculate h. The heat transfer model used for rigorous CFD calculations (Launder and Spalding, 1974, Fluent 4.4 User's Guide Volume, 1997) is expressed by the following correlation (valid for cells i on an exchange surface):

$$h_i = \frac{\rho_i C_{p_i} C_\mu^{0.25} k_i^{0.5}}{T_i^*} \tag{7.8}$$

where:

- C_μ is an empirical constant related to viscosity;
- k_i is the turbulent kinetic energy in cell i per mass unit;
- $T_i^* = Pr_t \left[\frac{1}{k}\log(Ey_i^*) + f(Pr_i)\right]$

  with

  $$f(Pr_i) = \frac{\pi/4}{\sin(\pi/4)} \left(\frac{A}{\kappa}\right)^{0.5} \left(\frac{Pr_i}{Pr_t} - 1\right) \left(\frac{Pr_t}{Pr_i}\right)^{0.25};$$

  $$y_i^* = \frac{\rho C_\mu^{0.25} k_i^{0.5} y_i}{\mu};$$

  A, κ, E = constant;

  Pr_t = Prandtl turbulent number (supposed to be constant);

  $$Pr_i = \frac{C_{p_i}\mu_i}{\lambda_i};$$

  y_i = distance from centre of cell i to the wall.

After some substitutions and considering the relation between kinetic energy per mass

unit and velocity (i.e. $k_i^{0.5} = v_i$), the following expression can be obtained:

$$h_i = \frac{A_1 \rho v_i}{A_2 + A_3 \left[\frac{1}{\kappa} \log \left(\frac{\rho v_i y_i}{\mu_i} \right) + f(Pr_i) \right]} \tag{7.9}$$

where

- $A_1 = C_\mu^{0.25}$
- $A_2 = \log(E) Pr_t / k$
- $A_3 = Pr_t \log(E A_1)$

are constants.

After multiplying both members of eqn. (7.9) by y_i/λ_i, we can write:

$$Nu_i = \frac{A_1 Re_i Pr_i}{A_2 + A_3 [\frac{1}{\kappa} \log Re_i + f(Pr_i)]} \tag{7.10}$$

where the Reynolds number Re_i and Pr_i are differently defined from eqn. (7.6). This equation can be linearised as:

$$\frac{1}{Nu_i} = \alpha \frac{1}{Re_i Pr_i} + \beta \frac{\frac{1}{\kappa} \log Re_i + f(Pr_i)}{Re_i Pr_i}. \tag{7.11}$$

Eqn. (7.11) is still a rigorous model for h, where

$$v_i = f(\mu_i, \rho_i, C_{p_i}, \lambda_i, N, tank\ geometry).$$

Now, we assume that non-dimensional numbers as defined in eqn. (7.6) can substitute the local non-dimensional numbers contained in eqn. (7.11), so that (Model HTC:TwoP):

$$\frac{1}{Nu} = \alpha \frac{1}{RePr} + \beta \frac{\frac{1}{\kappa} \log Re + f(Pr)}{RePr}. \tag{7.12}$$

This model is linear in the two parameters α and β, and relies on the actual physical meaning of non-dimensional numbers in expression (7.6). Several tests have demonstrated that the model has the tendency to slightly overestimate the first term $1/(RePr)$. Thus, a simpler local model (Model HTC:OneP):

$$\frac{1}{Nu} = \alpha \frac{\frac{1}{\kappa} \log Re + f(Pr)}{RePr} \tag{7.13}$$

has also been considered and tested. Since model (7.13) contains only one parameter,

it may be hazardous to use only one measured point to estimate the parameter. For that reason only the LS method was used to estimate parameter α.

7.3.2 Example II: Non-Newtonian Viscosity

This physical model will be applied to the example treated in chapter 8. The model describes a batch bioprocess involving a non-Newtonian fluid whose viscosity is computed by CFD calculations. For Newtonian fluids the shear stress is proportional to the strain rate:

$$\tau = \mu \dot{S} \tag{7.14}$$

where

$$\dot{S} = \frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} , \tag{7.15}$$

with x_i, x_j the orthogonal axis coordinates.

For non-Newtonian fluids, the viscosity μ becomes a function of $\dot{S}$, and is described by variable η (effective viscosity):

$$\tau = \left[\eta(\dot{S})\right] \dot{S}. \tag{7.16}$$

The power-law model assumes that non-Newtonian flow may be modelled according to the relation:

$$\tau = k\dot{S}^n = \left(k\dot{S}^{n-1}\right) \dot{S} \tag{7.17}$$

or, equivalently, to

$$\eta = k\dot{S}^{n-1}. \tag{7.18}$$

where n and k are parameters which depend on fluid characteristics (composition and temperature). Since local values of $\dot{S}$ are not easy to calculate, the value of the effective viscosity η is estimated by means of general correlations. In the case of stirred tank reactors, eqn. (7.18) is often expressed through the correlation (Metzner and Otto, 1957):

$$\eta = k(aN)^{n-1} \tag{7.19}$$

Property	Model name	y	x	α
Heat transfer coefficient	HTC:Exp	$log(Nu)$	d, N, ρ, μ, Cp	α, β, γ
Heat transfer coefficient	HTC:TwoP	$1/Nu$	d, N, ρ, μ, Cp	α, β
Heat transfer coefficient	HTC:OneP	$1/Nu$	d, N, ρ, μ, Cp	α
Viscosity	Visco1	$log(\eta/k)$	n	α
Viscosity	Visco2	$log(\eta/k)$	n	α, β

Table 7.1: Performance of different local models

where N is the impeller rotation speed and a is a geometry constant.

Similarly to correlation (7.19) two local models are suggested to estimate viscosity η. The first model linearises eqn. (7.19) to produce model Visco1:

$$\log \frac{\eta}{k} = \alpha(n-1) \tag{7.20}$$

The second model (Visco2) contains an additional parameter:

$$\log \frac{\eta}{k} = \alpha(n-1) + \beta \tag{7.21}$$

Parameter β is added to give more flexibility to the estimation capabilities of the local model. The performance of these local models in a process simulation-CFD zone model will be commented on in § 7.8.2.

7.3.3 Conclusions on Physical Local Models

Table 7.1 recapitulates the models described in § 7.3.1 and 7.3.2 outlining outputs **y**, inputs **x** and parameters $\boldsymbol{\alpha}$. The physical models described in this chapter have been considered because these properties were used in this work: they are an example demonstrating how physical models may be built up. There is a long list of other properties which may be similarly described by means of physical local models. These should demonstrate the following characteristics:

- dependence on fluid flow behaviour;
- existence of a rigorous model (i.e. they *can* be computed by means of CFD calculations);
- existence of or possibility of deriving a simplified estimation model.

Mass and heat transfer coefficients, diffusion coefficients, dissipation energy, etc. belong to this category. A library of useful physical local models could be created and introduced in the CFD - process simulation interface to improve calculation efficiency and estimation capabilities of local models.

7.4 General Local Models

General local models are output estimation methods having only mathematical but no physical correspondence to rigorous models. The big advantage is that they can be implemented independently of the underlying rigorous models which they have to represent. The disadvantage is that general models may not be as precise as good physical models.

7.4.1 Linear Models

Linear models are the simplest type of general models. The local model (Model Linear) is represented by the equation:

$$\mathbf{y} = \alpha_0 + \alpha_1 x_1 + \ldots + \alpha_n x_n \tag{7.22}$$

where $x_1 \ldots x_n$ are the inputs used by the rigorous model. The $n+1$ set of parameters α_i need at least $n+1$ points to be estimated. For instance, the heat transfer coefficient may be estimated by the linear model:

$$h(\mu, \rho, \lambda, C_p, N) \approx h = \alpha_0 + \alpha_1 \mu + \alpha_2 \rho + \alpha_3 \lambda + \alpha_4 C_p + \alpha_5 N. \tag{7.23}$$

Eqn. (7.22) is not the only possible form for a general linear model. Input and output values from previous rigorous calculations may be included, too. However, this latter approach is not considered in this work.

7.4.2 Non-Linear Models: Radial Basis Functions

Non-linear models are in principle more flexible and precise than linear models. Two non-linear implementations have been tested. The first class is represented by *radial basis functions*. Given a set of points $\{\mathbf{x}_i : i = 1, \ldots, m\}$ in $\mathbb{R}^n$ producing a set of outputs $\{\mathbf{y}_i : i = 1, \ldots, m\}$ in $\mathbb{R}^s$, a radial basis function has the form:

$$\mathbf{f}(\mathbf{x}) = \sum_{i=1}^{m} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|), \quad \mathbf{x} \in \mathbb{R}^n \tag{7.24}$$

where ϕ is a fixed function from $\mathbb{R}^+$ to $\mathbb{R}$. The choice of ϕ presents a wide range of possibilities (i.e., Powell, 1992). We only tested the use of the most common class of radial basis functions, i.e., the multiquadric interpolant (Hardy, 1971). This class is

determined by defining ϕ as

$$\phi = \sqrt{\sum_{j=1}^{n}(x_j - x_{j,i}) + R^2}. \tag{7.25}$$

Thus, eqn. (7.24) becomes (Model Radial):

$$\mathbf{f}(x_1, \ldots, x_n) = \sum_{i=1}^{m} \lambda_i \sqrt{\sum_{j=1}^{n}(x_j - x_{j,i}) + R^2} \tag{7.26}$$

where R is a user defined parameter.

When interpolation is used, coefficients λ_i are computed by solving the symmetric system of linear equations of order m generated by interpolation conditions, i.e.,

$$\mathbf{f}(x_{1,i}, \ldots, x_{n,i}) = \mathbf{y}_i, \quad \text{for } i = 1, \ldots, m.$$

where $\mathbf{y}_i$ are the rigorous outputs corresponding to inputs $\mathbf{x}_i$. This linear system can be written in the matrix form $\mathbf{Az} = \mathbf{B}$, where

$$\begin{aligned}
\mathbf{z} &= [\lambda_{i1}, \ldots, \lambda_{im}] \qquad i = 1, \ldots, s \\
\mathbf{B} &= [y_{i1}, \ldots, y_{im}] \qquad i = 1, \ldots, s \\
\mathbf{A} &= [a_{ij}] = \left[\left(\textstyle\sum_{j=1}^{n}(x_j - x_{j,i}) + R^2\right)^{0.5}\right].
\end{aligned}$$

This approach is a very powerful non-linear interpolation method, where all m points are used for interpolation. Nonetheless, the method did not prove to be suitable for this specific problem. Non-linearity makes the local model rather unstable when extrapolation is needed, as is often the case during CFD calculations. The main limitations, however, concern the very structure of the model. Inputs are treated uniformly: there are no parameters weighting them one by one. When the model is used to produce topographic maps, the two inputs (coordinates on the plan) are qualitatively the same and the number of inputs is small (two). This does not hold for CFD calculation where there may be many inputs which may have little or no relationship with each other. Furthermore, the choice of parameter R is rather ambiguous. There are suggestions (Carlson and Foley, 1992) for setting R when only two inputs are involved, but no method is available for more than two inputs. Different values for R have been tested but without significant improvement in the model, which remains very unstable and unreliable (see § 7.8.1).

7.4.3 Non-Linear Models: Shepard's Interpolation

Shepard's interpolation is one of the most interesting approaches for non-linear interpolation (and approximation, as well).

Let $\mathbf{p}$ and $\mathbf{q}$ denote generic element in $\mathbb{R}^n$. Next, a generic function ϕ on $\mathbb{R}^n \times \mathbb{R}^n$ is defined, subject to the condition

$$\phi(\mathbf{p}, \mathbf{q}) = 0 \quad \text{if and only if } \mathbf{p} = \mathbf{q}$$

Typical examples are $\|\mathbf{p} - \mathbf{q}\|$ or $\|\mathbf{p} - \mathbf{q}\|^2$. Then we set up (in analogy with classical Lagrange interpolation formulas (Kinkaid and Cheney, 1996) the following functions:

$$u_i(\mathbf{x}) = \prod_{\substack{j=1 \\ j \neq i}}^{m} \frac{\phi(\mathbf{x}, \mathbf{x}_i)}{\phi(\mathbf{x}_i, \mathbf{x}_j)} \tag{7.27}$$

These functions have the cardinal property

$$u_i(\mathbf{x}_j) = \delta_{ij}.$$

Thus, an interpolant at the given points is provided by the function:

$$\mathbf{f}(\mathbf{x}) = \sum_{i=1}^{m} \mathbf{y}_i u_i(\mathbf{x}) \tag{7.28}$$

where $\mathbf{y}_i$ are outputs from rigorous function $\mathbf{F}(\mathbf{x}_i)$. Although, as defined, Shepard's method does not have very good fitting properties, a modified Shepard's method was suggested by Franke and Nielson (1980) as a more efficient and stable interpolation procedure. The modified expression has the form (general expression for Model Shepard):

$$\mathbf{f}(\mathbf{x}) = \frac{\sum_{j=1}^{m} W_j(\mathbf{x}) \mathbf{Q}_j(\mathbf{x})}{\sum_{i=1}^{m} W_i(\mathbf{x})} \tag{7.29}$$

where, as in the standard method, $\mathbf{Q}_j$ (called *nodal* functions) are functions satisfying $\mathbf{Q}_j(\mathbf{x}_j) = \mathbf{y}_j$. The definition of $\mathbf{Q}_j$ is a very delicate matter, which may affect the reliability of the local model. Literature (e.g, Renka, 1988, Carlson and Foley, 1992, Iiguni *et al.*, 1997) reports several cases where bivariate quadratic functions or Taylor series expansions are used. Taylor series expansion cannot be used in our cases, since the complexity of the model does not allow such reduction for CFD equations. Bivariate quadratic functions have been chosen, even if:

a. bivariate quadratic functions have been tested only for 2D and few 3D examples;

b. bivariate functions can get extremely computationally demanding when many variables are involved.

Bivariate quadratic function $\mathbf{Q}_j$ is defined by

$$\mathbf{Q}_j(\mathbf{x}) = \mathbf{Q}_j(x_1, \ldots, x_n) = \sum_{i=1}^{n} \alpha_{ij}(x_i - x_{ij})^2 + \sum_{l \neq k} \beta_{(lk)j}(x_l - x_{lj})(x_k - x_{kj}) + \sum_{i=1}^{n} \gamma_{ij}(x_i - x_{ij}) + y_j \tag{7.30}$$

Coefficients α, β and γ are determined by minimising:

$$\sum_{\substack{i=1 \\ i \neq j}}^{m} \omega_i(\mathbf{x}_j)[\mathbf{Q}_j(\mathbf{x}_i) - \mathbf{y}_i]^2 \tag{7.31}$$

for

$$\omega_i(\mathbf{x}) = \left[\frac{(R_q - d_i)_+}{R_q d_i}\right]^2$$

$$(R_q - d_i)_+ = \begin{cases} R_q - d_i & \text{if } d_i < R_q \\ 0 & \text{if } d_i \geq R_q \end{cases}$$

where R_q is a radius of influence about point $\mathbf{x}_i$. Computational effort is easily predictable: for instance, if $n = 5$ (e.g. heat transfer coefficient case), the number of parameters is $20m \times s$. As a consequence, for each iteration, $m \times s$ LS problems must be solved and at least 20 points from a rigorous model must be stored before the local model may be applied. It is important to notice that only points whose radii of influence include $\mathbf{x}_j$ have non-zero contributions to the LS fit. Thus, $\mathbf{Q}_j$ is locally defined. Diameter d_i denotes the Eucledian distance between $\mathbf{x}$ and $\mathbf{x}_i$.

The relative weights W_j in eqn. (7.29) are defined by the functions

$$W_j(\mathbf{x}) = \left[\frac{(R_w - d_j)_+}{R_w d_j}\right]^2 \tag{7.32}$$

where R_w is a radius of influence about point $\mathbf{x}_j$ and d_j is the Eucledian distance between $\mathbf{x}$ and $\mathbf{x}_j$. The data $\mathbf{x}_j$ only influences interpolated values at points within radius R_w.

Some comments must be added about the two radii of influence R_q and R_w. Since R_q varies with i, then all nodes must be tested as possible candidates for inclusion in the fit. That results in high problem complexity. So it was preferred to fix R_q for each function Q_j. The value of a radius may be set at a fixed uniform value (Franke and Nielson, 1980) or we can choose R_w and R_q just large enough to include m_w and m_q points for fixed values of m_w and m_q (Renka, 1988). In our case, the number m of points is relatively small, so it appeared reasonable to include most points. The radius is defined according to the following procedure:

a. Among the m points $\mathbf{x}_i$ find $D_{max} = \max_{\mathbf{x}_i} (\|\mathbf{x} = \mathbf{x}_i\|)$;

b. $R_q = D_{max}/2$.

Since R_q denotes the radius of influence of the data points on the nodal functions, while R_w denotes the radius of influence of nodal functions on the interpolating function, it is recommended that $R_w \leq R_q$. We set $R_q = \sqrt{2}R_w$ (Franke and Nielson, 1980).

Initially, Shepard's interpolation has been tested in the case when $\mathbf{Q}_j$ are bivariate quadratic functions. The ability to fit CFD rigorous models appears to be very good, but calculations are excessively expensive.

Thus, simpler expressions were derived from the bivariate quadratic functions. Best results were given by the form:

$$\begin{aligned} \mathbf{Q}_j(\mathbf{x}) = \quad & \mathbf{Q}_j(x_1, \ldots, x_n) = \sum_{i=1}^{n} \alpha_{ij}(x_i - x_{ij})^2 + \\ & \sum_{i=1}^{n} \gamma_{ij}(x_i - x_{ij}) + \mathbf{y}_j, \end{aligned} \tag{7.33}$$

which was implemented and tested.

7.5 Robustness

Robustness is a critical issue in running a simulation of this type. It must be ensured that data coming from CFD calculations as well as from local model estimation do not destabilise the numerical methods of the process simulation package. This work mainly addresses the issues concerned with dynamic simulations. Although many results are general and may be applied to steady-state cases, the search for specific procedures to ensure a robust solution of steady-state models has not been considered. The next sections develop an approach to increase robustness by dealing with the following aspects:

- initialisation and updating of local models;
- filtering of CFD results (which may be affected by noise);
- control of accuracy in results of local models.

7.5.1 Flowrate Reconciliation

CFD packages typically adopt convergence criteria which allow a looser convergence than process simulation packages like gPROMS. In particular, the CFD mass balance usually presents a small discrepancy which may affect the more precise process simulation calculations. That is due to the need to allow a looser tolerance on the overall mass balance either because it is necessary to reduce the computational time or, more often, because tighter criteria would not obtain a converged solution. In the case of incompressible fluids the problem is solved by computing one flowrate per zone in the process simulation model (see chapter 4); however, for compressible fluids (or in the case of incompressible fluids at constant density) it is essential that the set of CFD computed mass fluxes always satisfy the mass balance in each zone. A reconciliation procedure was designed and implemented to satisfy this basic condition.

If we consider the set $F_{kk'}$ of flows from zone k to zone k' (the set of internal zone is $\mathcal{Z}_i$, while the set of environment zones is $\mathcal{Z}_e$), the mass balance can be stated as:

$$\sum_{\forall k' \neq k} F_{kk'} = \sum_{\forall k' \neq k} F_{k'k} \quad \forall k \in \mathcal{Z}_i \tag{7.34}$$

The flowrates from the CFD solution $\hat{F}_{kk'}$ satisfy eqn. (7.34) according to the tolerance set in the CFD package, but that may not be sufficient if a tighter process simulation tolerance is defined. Thus, *given* a set of flowrates $\hat{F}_{kk'}$ as computed from CFD calculations, the objective of this approach is to *derive* a set of flowrates $F_{kk'}$ satisfying eqn. (7.34). In a more formal way, we want to determine $F_{kk'}$ such that:

$$\sum_{\forall k' \neq k} F_{kk'} - F_{k'k} = 0 \qquad \forall k \in \mathcal{Z}_i \tag{7.35}$$

$$\min_{F_{kk'}} \frac{1}{2} \sum_{k,k'} \left(F_{kk'} - \hat{F}_{kk'} \right)^2 \quad \forall k, k' \in \mathcal{Z}_i, \mathcal{Z}_e \tag{7.36}$$

In order to find the optimal solution, the simplest way is to form the Lagrangian:

$$\mathcal{L}(F_{kk'}, \lambda_k) = \frac{1}{2} \sum_{k,k'} \left(F_{kk'} - \hat{F}_{kk'} \right)^2 + \sum_{k \in \mathcal{Z}_i} \lambda_k \sum_{\forall k' \neq k} (F_{kk'} - F_{k'k})$$

and then consider the system:

$$\frac{\partial \mathcal{L}}{\partial F_{kk'}} = 0 = \begin{cases} F_{kk'} - \hat{F}_{kk'} + \lambda_k - \lambda_{k'} & k \in \mathcal{Z}_i, \quad k' \in \mathcal{Z}_i \\ F_{kk'} - \hat{F}_{kk'} + \lambda_k & k \in \mathcal{Z}_i, \quad k' \in \mathcal{Z}_e \\ F_{kk'} - \hat{F}_{kk'} & k \in \mathcal{Z}_e, \quad k' \in \mathcal{Z}_i \end{cases} \tag{7.37}$$

The reconciled flowrates are given by solving the eqn. (7.35) and (7.37) for the unknown $F_{kk'}$ and λ_k.

The *reconciliation* is applied only to flowrates $F_{kk'}$ for which $\hat{F}_{kk'} > 0$. Otherwise, the procedure may produce meaningless negative flowrates[1].

7.5.2 Filtering of CFD results

As noted, CFD typically allows looser convergence criteria in the simulation calculations. This does not only affect mass balance requirements, but may cause some fluctuations in the computed results. This results in a certain level of noise which may affect the process simulation numerical methods, because of the contradictory outputs which its predictor-corrector methods receive. A noisy calculation of CFD results may well affect the gPROMS solution and make the simulation crash. The problem is critical during the *initialisation* phase (see § 7.2.3) when only CFD rigorous calculations are required. The noise in CFD calculations may confound the response of different, but close sets of inputs $\mathbf{x}$. Different outputs $\mathbf{Y}$ become indistinguishable from each other without a clear correspondence to their inputs $\mathbf{x}$. The estimated local model parameters produce unreliable models so that, technically, no initialisation is obtainable.

The solution is achieved by means of a very simple filter introduced to minimise the effect of the noise of CFD calculations. Given a vector of inputs $\mathbf{x}$, CFD rigorous calculations are carried out only if

$$\left| \frac{x_i^* - x_i}{x_i^*} \right| > \delta \quad i = 1, \ldots, n \tag{7.38}$$

where δ is a preset tolerance and vector $\mathbf{x}^*$ is the vector of inputs used to compute the last rigorous set of outputs $\mathbf{Y}^*$. Otherwise, $\mathbf{Y} = \mathbf{Y}^*$. The filter substitutes the noisy function $\mathbf{Y} = \mathbf{F}(\mathbf{x})$ with a piece-wise function. Values of tolerance δ of $\mathcal{O}(10^{-5})$ proved to impede simulation crashes.

---

[1] There might still be positive $\hat{F}_{kk'} \approx 0$ which, after the reconciliation, could turn into negative flowrates. In order to avoid complex constrained optimisation algorithms, in the practical implementation we defined a tolerance ϵ such that if $\hat{F}_{kk'} < \epsilon$, $\hat{F}_{kk'}$ is assumed to be equal to 0. Several simulations have demonstrated that very small corrections are required to satisfy the mass balances in each zone and that tolerance ϵ can be chosen very small without resulting in negative fluxes.

7.5.3 Inputs Check

The necessary condition for the existence of a local model is the availability of a number of distinct points from rigorous calculations, sufficient to estimate the required parameters. Nonetheless, existence does not imply suitability: the outputs $\mathbf{y}$ from the local model will be a good approximation of rigorous CFD outputs $\mathbf{Y}$ only if the set of inputs $\mathcal{X}$ used to estimate local model parameters $\boldsymbol{\alpha}$ is representative of the region to which $\mathbf{x} \mid \mathbf{y} = \mathbf{f}(\boldsymbol{\alpha}, \mathbf{x})$ belongs.

It has been noted that a stored vector $\mathbf{x}_j \in \mathcal{D}$ may be used to estimate parameters of a local model only if it belongs to a ball of $\mathbf{x}$ of a defined radius ρ, i.e.

$$\mathbf{x}_j \in \mathcal{X} \Rightarrow \left[\sum_{i=1}^{n}\left(\frac{w_i(x_{ij}-x_i)}{x_{ij}}\right)^2\right]^{0.5} < \rho \tag{7.39}$$

Vector $\mathbf{w} \mid w_i \geq 1$ is used to take into account the fact that the rigorous model $\mathbf{F}$ may present a different sensitivity to different variables. The more sensitive to x_i the CFD calculations, the greater w_i should be.

It is possible to make the control more flexible by adding a procedure able to modify the ball radius during the simulation. At the beginning of the simulation, radius ρ is set to ρ_{in}. If there are sufficient points $\mathbf{x}$ to estimate the local model parameters, but condition (7.39) is not satisfied, then both CFD and local model calculations are carried out. If $\mathbf{Y} \approx \mathbf{y}$ within a tolerance ξ, then ρ is enlarged. Formally, the algorithm is as follows:

ALGORITHM ENLARGE
Given the maximum radius ρ_{max}, a suitable tolerance ξ, a database $\mathcal{D}$, the number $\hat{n}$ of points required to estimate parameters in the local model:

1. Initialisation
   a. $\rho = \rho_{in}$
   b. $m^* = 0$
   c. $m = 0$
   d. $\mathcal{X} = \mathcal{X}_{max} = \{\}$
2. IF at least $\hat{n}$ distinct points $\mathbf{x}_j \in \mathcal{D}$, THEN
   a. Consider points $\mathbf{x}_j \in \mathcal{D}$ one by one and compute distance

$$d_j = \left[\sum_{i=1}^{n}\left(\frac{w_{i,max}(x_{ij}-x_i)}{x_{ij}}\right)^2\right]^{0.5} \tag{7.40}$$

   IF $d_j < \rho_{max}$ THEN

I. $\mathcal{X}_{max} = \mathcal{X}_{max} \cup \{\mathbf{x}_j\}$
II. $m^* = m^* + 1$
III. change indexes in $\mathcal{X}_{max}$ such that $d_l < d_{l+1}$

b. IF $m^* \geq \hat{n}$ points in $\mathcal{X}_{max}$, THEN
I. Consider points $\mathbf{x}_j \in \mathcal{X}_{max}$: IF $d_j < \rho$, THEN
i. $\mathcal{X} = \mathcal{X} \cup \{\mathbf{x}_j\}$
ii. $m = m + 1$
II. IF $m \geq \hat{n}$ THEN estimate parameters for local models and compute $\mathbf{y}$
ELSE
i. Call CFD model and compute $\mathbf{Y}$
ii. Consider the first $\hat{n}$ points in $\mathcal{X}_{max}$
iii. Estimate parameters for local models and compute $\mathbf{y}$
iv. IF

$$\left[\sum_{i=1}^{p}\left(\frac{Y_i - y_i}{Y_i}\right)^2\right]^{0.5} < \xi, \tag{7.41}$$

THEN $\rho = d_{\hat{n}}$
ELSE call CFD model and compute $\mathbf{Y}$

3. ELSE call CFD calculations and compute $\mathbf{Y}$

The approach produced good results, even if enlarging the ball radius might cause the system robustness to deteriorate.

7.5.4 Control of Estimated Outputs

It has been observed that it is necessary to check whether the estimated outputs $\mathbf{y}$ are close to the ones used for the estimation or not. Outputs from local models are accepted if they are within an interval defined by the set $\mathcal{Y}$ whose inputs $\mathcal{X}$ were used to calculate the model parameters. The control procedure is defined by the following algorithm:

ALGORITHM CHECK
Given $\bar{\mathbf{Y}}$ (as in ALGORITHM SCREEN_OUTPUT) and set $\mathcal{Y}$

1. Calculate standard deviation:

$$\sigma_i = \left[\frac{1}{m}\sum_{j=1}^{m}\left(Y_{ij} - \bar{Y}_i\right)^2\right]^{0.5} \tag{7.42}$$

2. IF estimated output $\bar{Y}_i - 3\sigma_i \leq y_i \leq \bar{Y}_i + 3\sigma_i$ THEN accept it
ELSE call rigorous CFD calculations.

When using the LS estimation method, further safeguards are introduced in the estimation procedure by controlling the value of the residual norm defined by eqn. (A.17) as explained in appendix A.

If the procedure allows an enlargement of tolerance for set $\mathcal{X}$ as in § 7.5.3, then whenever the control for the estimated output fails the tolerance is reset to its initial value, i.e. $\rho = \rho_{in}$.

7.6 Application of Local Models with Multiple Zones

A multiple zone model presents some specific issues in the application of local models. In the previous sections, we assumed that each local model was defined for a single zone (which may even represent the entire domain). When a set of zones is taken into account, different approaches may be examined to tackle the problem. For instance:

- should the local model be a function accepting all the inputs (for each of the zones) and returning all the required outputs at once? Or should the local model be divided into a set of local models corresponding to the number of zones?

- should the types of estimated properties affect the way local models are used? i.e. should the type of local model depend on the network properties?

Some general approaches are next described to define a procedure to handle multiple zones models.

7.6.1 Network Dependent Outputs

Some outputs cannot be associated with a specific zone, but are just a *property of the network*. Properties representing fluxes among zones (e.g. mass or heat fluxes) belong to this category. It is impossible to relate a subset of them to any specific zone. In this case, given an output $\mathbf{Y}$ dependent on np variables and a network of nz zones, the local model $\mathbf{f}$ is described by the function:

$$\mathbf{f}: \quad \mathbb{R}^n \longmapsto \mathbb{R}^p \quad \text{where} \quad \begin{cases} n = np \times nz = \text{number of inputs} \\ p = \text{number of outputs} \end{cases}$$

A great number of data has to be collected before an estimation scheme may be applied and the computational burden may become very expensive. In some important cases, a solution may be found by means of some simplifying assumptions. For instance, mass flowrates do not usually demonstrate a high dependence on physical properties: variations are slow and small. In this case, a first alternative (Method 1) is to abandon local models and to use a very simple procedure such as:

1. Get inputs $\mathbf{x}$ and compute CFD rigorous outputs $\mathbf{Y}$;

2. Store outputs $\mathbf{Y}$ as $\mathbf{Y}^*$ and corresponding inputs $\mathbf{x}$ as $\mathbf{x}^*$;

3. Get new inputs $\mathbf{x}$. Given a suitable tolerance δ, if $\frac{|x_i - x_i^*|}{x_i^*} < \delta$, then return $\mathbf{Y}^*$, else call CFD rigorous calculations.

This procedure may be pushed even further to allow greater change in inputs $\mathbf{x}$ if we observe that outputs $\mathbf{Y}$ demonstrate little dependence on those inputs. In such a case, method 1 is modified into Method 2:

1. get inputs $\mathbf{x}$ and compute CFD rigorous outputs $\mathbf{Y}$;

2. store outputs $\mathbf{Y}$ as $\mathbf{Y}^*$ and corresponding inputs $\mathbf{x}$ as $\mathbf{x}^*$;

3. get new inputs $\mathbf{x}$. Given an initial tolerance $\delta = \delta_{in}$, if $\frac{|x_i - x_i^*|}{x^*} < \delta$, then return $\mathbf{Y}$. Else:

   a. call CFD rigorous calculations to compute new outputs $\mathbf{Y}$;

   b. given a suitable tolerance δ_y, if $\frac{|Y_i - Y_i^*|}{Y_i^*} < \delta_y$, then increase tolerance δ.

Another solution is represented by an approach which is a trade-off between efficiency needs and input-output dependency. According to this method, the number of available inputs determines the local model structure. Let us assume the general non-linear model:

$$\mathbf{y} = \alpha + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \gamma_i x_i^2 + \sum_{i \neq j} \eta_{ij} x_i x_j. \tag{7.43}$$

All outputs $\mathbf{Y}$ from rigorous calculations and the associated inputs $\mathbf{x}$ are stored in database $\mathcal{D}$. The algorithm (Method 4) works as follows:

- if a number $m < n + 1$ of data is available, the model is used as $\mathbf{y} = \alpha$;

- if $n < m < 2n+1$ data are available, we can use the linear model (i.e., parameters α are included);

- if $2n < m < 2n + \binom{m}{n}$ data are available, we can adopt the quadratic form, but without considering the crossing terms (i.e. up to parameters γ only);

- if $m \geq 2n + \binom{m}{n}$ data are available, then the complete form is sustainable.

7.6.2 Network Independent Outputs

A different approach is taken into account when a second category of outputs is considered, i.e. outputs which can be defined as *network-independent* (or *quasi*-network-independent). This category includes those outputs totally or strongly related to the physical properties of one single zone. Even if not theoretically correct, many intensive properties (energy of dissipation, shear stress, effective viscosity, etc.) fall into this class. In this case the interpolation function $\mathbf{f}$ becomes a set of nz functions:

$$\begin{aligned} \mathbf{f}_1 &: \mathbb{R}^{np} \longmapsto \mathbb{R}^{p_1} \\ &\vdots \\ \mathbf{f}_{nz} &: \mathbb{R}^{np} \longmapsto \mathbb{R}^{p_{nz}} \end{aligned}$$

where p_i is the number of outputs for each zone i. In this case (Method 5) the number of data needed to start a local model estimation is substantially reduced.

The user is given the possibility to choose the most suitable interpolation scheme. The array `CFDparameter` (see chapter 3) is used for this purpose. The parameter value is:

- 0 if we require only one CFD computation. In this case it is assumed that fluid flow behaviour does not (significantly) change along the simulation. This approach is called Method 0.
- $1 \div 5$ if Methods $1 \div 5$ are adopted.

These parameters are set for each category of output obtained by the CFD solution. Different outputs may require different interpolation methods (e.g. in a liquid phase reactor, we may decide to use Method 0 for zone pressures, Method 1 for mass flowrates and Method 5 for the energy of dissipation in each zone).

7.7 The Interface Design

It is now possible to outline the final design of the interface and all the procedures and methods for integrating CFD and process simulation. The overall integration can be split into two main processes: the *declarative* and *computational* processes.

The declarative process has been described through chapters $3 \div 6$ and consists of the definition of the process simulation model (which may be steady-state or dynamic) and its relation to the CFD model (which is steady-state). This is achieved through:

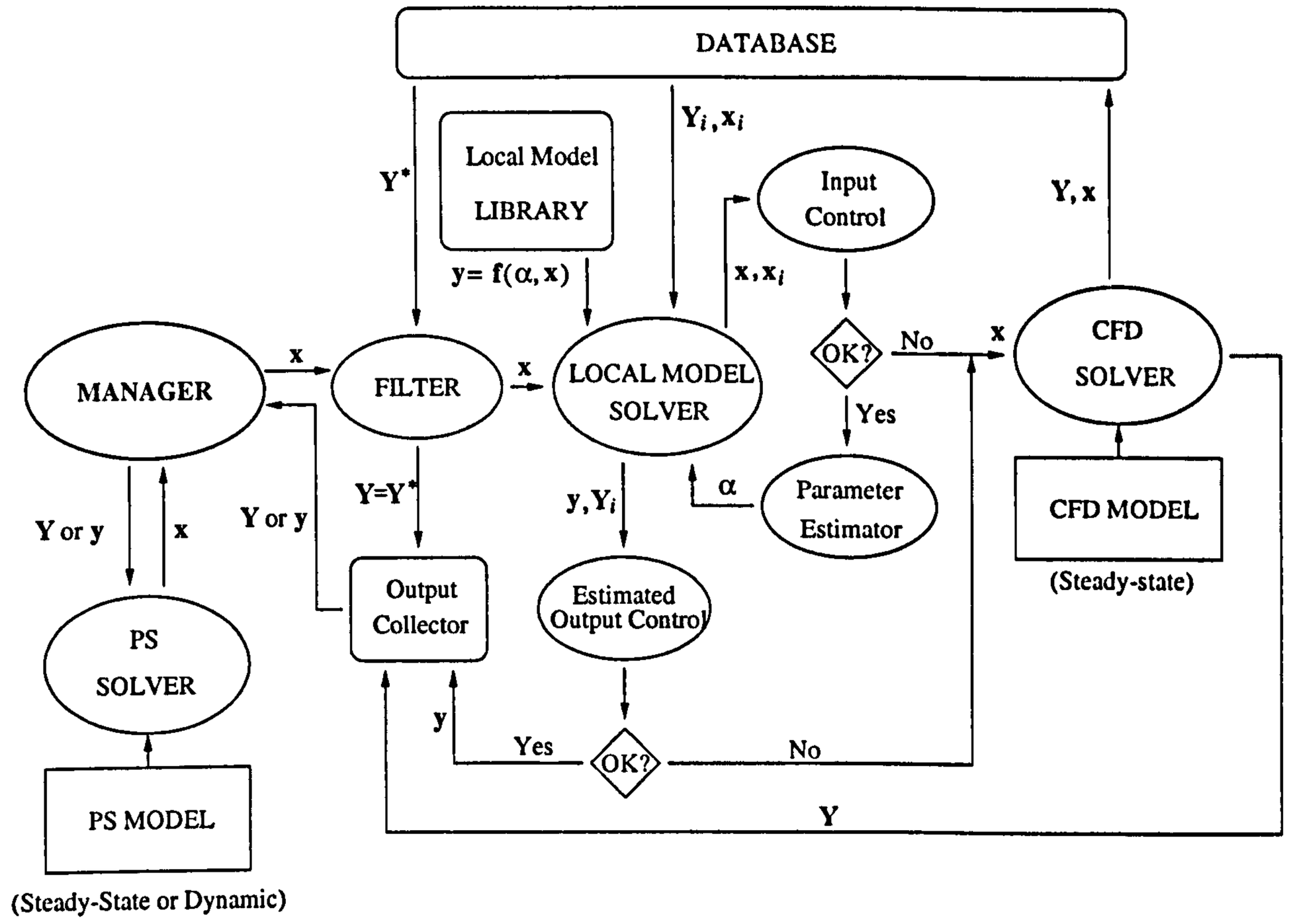


Figure 7.3: Data flux in the interface (PS stands for process simulation).

a. the design of a model, i.e. defining zones and their connection (chapter 3). In particular the design of:

   - the internal zone model
   - the environment zone model
   - the zone network model;

b. the definition of methods capable of establishing a network of zones (chapter 5);

c. the implementation of functions to compute the variables and parameters required by the local models.

The computational process handles the flux of information between the two packages as outlined in Figure 7.2. As observed in chapter 2, the process simulation software handles the flux of information between the process simulation and CFD solvers. The procedures to ensure convergence and speed to the integrated solution process are included in this context. Figure 7.3 refines the ideas illustrated in Figure 7.2 and describes the computational process by incorporating the functions and algorithms defined throughout this chapter:

a. the outputs/inputs flux is handled by the simulation MANAGER which starts the interfacing procedure (these procedures may be interpreted as internal functions of the program manager);

b. the FILTER checks inputs $\mathbf{x}$ and determines whether to return previous outputs $\mathbf{Y}^*$ (if available and suitable) or to issue an output calculation request;

c. inputs $\mathbf{x}$ accepted by the filter are handled by the LOCAL MODEL SOLVER, which

   i. considers a local model $\mathbf{y} = \mathbf{f}(\boldsymbol{\alpha}, \mathbf{x})$ from the local model LIBRARY

   ii. verifies if in the database there exists a suitable subset $\mathcal{X}$ of inputs $\mathbf{x}_i$ which may be used for estimating the local model parameters

   iii. if a suitable subset $\mathcal{X}$ is found, then computes local model outputs $\mathbf{y}$ after estimating parameters $\boldsymbol{\alpha}$;
   otherwise, forwards inputs $\mathbf{x}$ to the CFD SOLVER;

   outputs $\mathbf{y}$ are compared to subset $\mathcal{Y}$ of previous rigorous outputs; if estimation $\mathbf{y}$ is rejected, then inputs $\mathbf{x}$ are sent to the CFD SOLVER;

d. outputs $\mathbf{Y}$, $\mathbf{Y} = \mathbf{Y}^*$ or $\mathbf{y}$ are collected and sent back to the MANAGER and then to the process simulation PS SOLVER.

The design of the interface in terms of:

- flux of information management
- filtering criteria
- efficiency procedures (local model management)
- robustness methods

is independent of the model structure. For instance, the examples in chapters 2 and 8 are handled in exactly the same way, although the former utilises a single zone and the latter adopts a network with multiple zones.

7.8 Local Model Performance Analysis

Several tests have been carried out to assess and demonstrate the effectiveness of local models in accelerating calculation speed. The first test utilises the batch reactor described in chapter 2. It was chosen because of its simplicity (a single zone) and the possibility of running several tests in a reasonable time. A second test uses a multiple zone network to check the effect of local models in speeding up calculations.

7.8.1 Test No.1

Local models are compared against each other and to a base case for which a simulation is carried out utilising just rigorous CFD calculations (local models are not used). The base case only utilises the filtering procedure described in § 7.5.2. We compared models according to criteria of

1. convergence (C), i.e. whether a solution is obtained or not;

2. accuracy (A); three levels of accuracy are considered: *good* (G) if simulated heat transfer coefficient (which is the variable computed using CFD calculations) does not differ more than 1% from the base case; *medium* (M) if it does not differ more than 5% from the base case; *bad* B if it differs more than 5% from the base case;

3. CPU reduction; two parameters are defined. The parameter η (acceleration factor) is defined as the value of the ratio

$$\eta = \frac{\text{CPU}_B \text{ Time}}{\text{CPU}_i \text{ Time}}$$

between case 1 (the base case) and case i.

The parameter $gp\%$ is defined as the ratio[2]

$$gp\% = \frac{gPROMS \text{ CPU Time}}{\text{total CPU Time}} ;$$

4. Number of rigorous CFD function evaluations (N_{cfd}).

Table 7.2 illustrates the results in terms of computational speed for the simulation using the model of chapter 2 (§ 2.4.1). The output is represented by the heat transfer coefficient h, while inputs (process simulation outputs) are fluid viscosity and density. The case numbers in the Table correspond to the following implementations:

Case 1: Base case (rigorous CFD).

Case 2: Linear model of § 7.4.1 (Model Linear). Model parameters estimated via interpolation (§ 7.2.3). Radius ρ (eqn. (7.39)) is kept constant and equal to 5×10^{-2}.

Case 3: Linear model of § 7.4.1 (Model Linear). Model parameters estimated via interpolation. Radius ρ is kept constant and equal to 1×10^{-2}.

[2]The gPROMS CPU time also includes the calculations performed to estimate local model parameters and to solve local models.

	Base	*General Local Models*					*Physical Local Models*			
CASE	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
C	YES	YES	YES	YES	NO	YES	YES	YES	YES	YES
A	-	B	M	G	/	G	M	G	G	G
η	1	6.6	1.4	12.9	/	8.3	6.7	37.6	51.2	16.8
$gp\%$	0.03	2.31	1.08	1.72	/	2.84	0.44	2.35	2.60	0.80
N_{cfd}	1664	390	1248	178	/	203	226	46	35	97

Table 7.2: Test 1a. Performance of different local models

Case 4: Linear model of § 7.4.1 (Model Linear). Model parameters estimated via LS method (§ 7.2.1). Radius ρ is kept constant and equal to 5×10^{-2}.

Case 5: Radial Basis Function of § 7.4.2 (Model Radial). Model parameters estimated via interpolation. Radius ρ is kept constant and equal to 5×10^{-2}.

Case 6: Shepard's interpolation of § 7.4.3 (Model Shepard). Nodal functions Q_j are defined as in eqn. (7.33). Radius ρ is kept constant and equal to 5×10^{-2}.

Case 7: Physical local model of § 7.3.1: exponential law (Model HTC:Exp) defined by equation (7.7). Model parameters estimated via LS method. Radius ρ is kept constant and equal to 5×10^{-2}.

Case 8: Physical local model: two parameter model (Model HTC:TwoP), described by eqn. (7.12). Model parameters estimated via LS method. Radius ρ may be changed (§ 7.5.3); radius initial value is equal to 5×10^{-2}.

Case 9: Physical local model: one parameter model (Model HTC:OneP), described by eqn. (7.13). Model parameters estimated via LS method. Radius ρ may be changed (§ 7.5.3); radius initial value is equal to 5×10^{-2}.

Case 10: Physical local model: one parameter model (Model HTC:OneP), described by eqn. (7.13). Model parameters estimated via LS method. Radius ρ is kept constant and equal to 5×10^{-2}.

Test No. 1a

Since the simulation of the base case requires very expensive calculations the operating policy of § 2.4.1 was simplified. The batch is considered completed when molar fraction of reactants is less than 0.1. CFD function **Y** requires two inputs (viscosity μ and density ρ) to deliver the heat transfer coefficient value.

Table 7.2 outlines the performances of different local models. The simulation is successfully carried out in all cases but Case 5: the adopted radial basis function proved too unstable and the simulation crashed. The estimation of the heat transfer coefficient is very unsatisfactory in Case 2: simple interpolation techniques for estimating the model parameters are quite unreliable. Accuracy improves if a very small tolerance radius ρ is chosen (Case 3). However, the better performances are given by general and physical local models using LS estimation techniques. Case 7 also achieves a rather poor *medium* accuracy: as noted in § 7.3.1, the model presents a different sensitivity to inputs $\mathbf{x}$ with respect to the rigorous CFD model.

Results in Table 7.2 demonstrate the great effectiveness of physical models in this simulation. Even with constant radius ρ (Case 10), the local model CFD:OneP produces an acceleration in calculation time of almost 17 times the base case. When the tolerance is automatically adjusted, results become even more impressive (Case 9): the simulation becomes over 50 times faster while retaining good accuracy. Another significant result is the good performance of simple linear models with the least square method (Case 4). Shepard's interpolation (Case 6) is rather effective and accurate and, being a non-linear interpolation technique, it may allow more flexibility and accuracy than linear models if these prove to be inadequate.

In all these simulations, the gPROMS CPU time represents a small percentage (maximum 2.84%) of the overall computational time. CFD calculations are the principal burden and it is clear that the acceleration factor η reflects the number of rigorous CFD evaluations (see N_{cfd}) in the simulation. These are drastically reduced.

The excellent general performance of local models in this example is a consequence of the lack of drastic changes in hydrodynamics and of the behaviour of the most important input in this example, i.e., viscosity (Figure 7.4), which in the last part of the simulation assumes values that are already stored in the database.

Test No. 1b

The better performing models (Cases 4 and 9) were also tested in a more demanding simulation where the impeller speed N is assumed to vary with temperature T (K) according to:

$$N = \begin{cases} 90 & \text{if } T \leq 320 \\ 90 + 0.75(T - 320) & \text{if } T > 320 \end{cases}$$

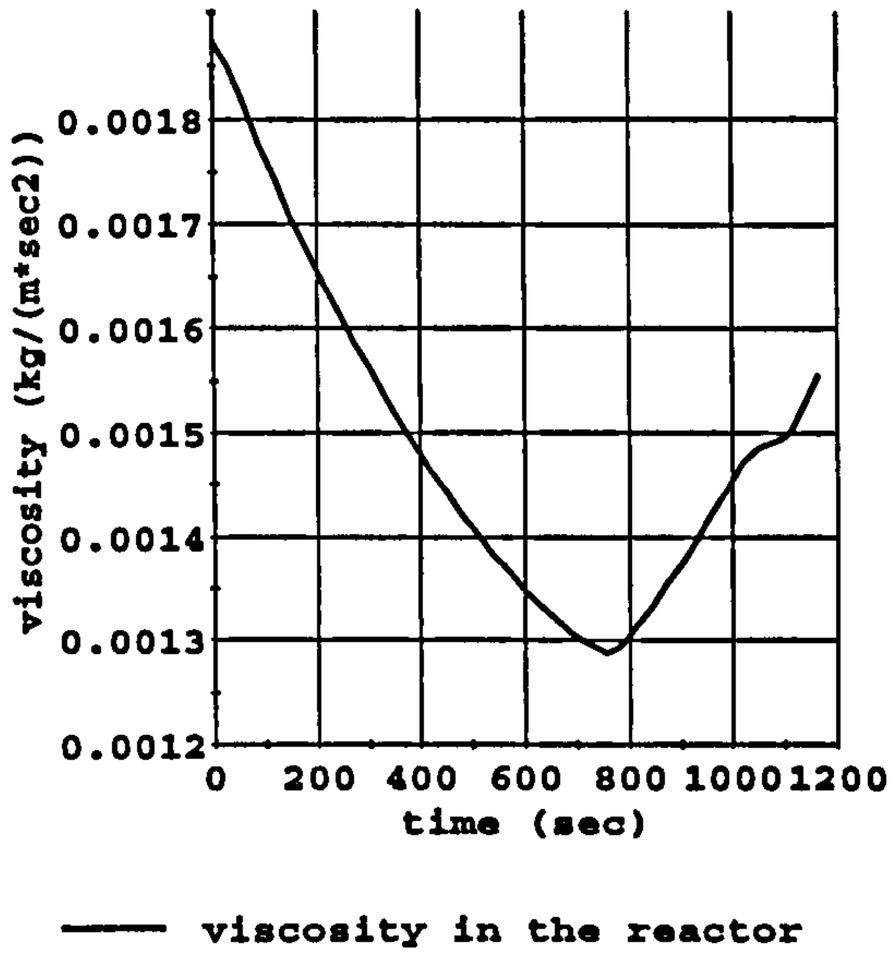


Figure 7.4: Fluid viscosity in the reactor during simulation.

This control law on rotation speed has no physical meaning, but it provides a good test both of the effectiveness and robustness of using local models since the impeller speed drastically affects the fluid flow behaviour. Also, it introduces a discontinuous function in one of the input variables. In this example, rotation speed is added to the other two inputs, viscosity and density.

Table 7.3 summarises the results. The use of local models decreases computational time by a factor of 3-4, i.e. not as much as before. Accuracy and convergence are achieved: great changes in hydrodynamics seem not to affect the predictive ability of the local models. Besides, the local model procedure is capable of handling discontinuity in the input variables. When local models are used, a limited number of updates using the rigorous CFD model are required, but each of these updates requires long computations since hydrodynamic conditions may be heavily changed. Furthermore, in this example the rotation speed (Figure 7.5) never assumes the same value twice (apart from the initial constant period) so that local models continuously need updating by means of rigorous CFD calculations.

The ability of the local models to decrease calculation times is strongly affected by the type of simulations we are considering. When updates in hydrodynamics are complex and computationally demanding, the effect of local models is diminished since fewer but much more expensive CFD calls are required. The burden may be decreased when the simulation presents a periodic behaviour or, at least, some repetition in the inputs involved in the CFD calculations. In these cases the interface is able to

CASE	1	4	9
C	YES	YES	YES
A	–	G	G
η	1	3.4	4.0
$gp\%$	0.03	0.60	0.26
N_{cfd}	1950	158	104

Table 7.3: Test 1b. Performance of different local models (case numbers and symbols are the same as in Table 7.2).

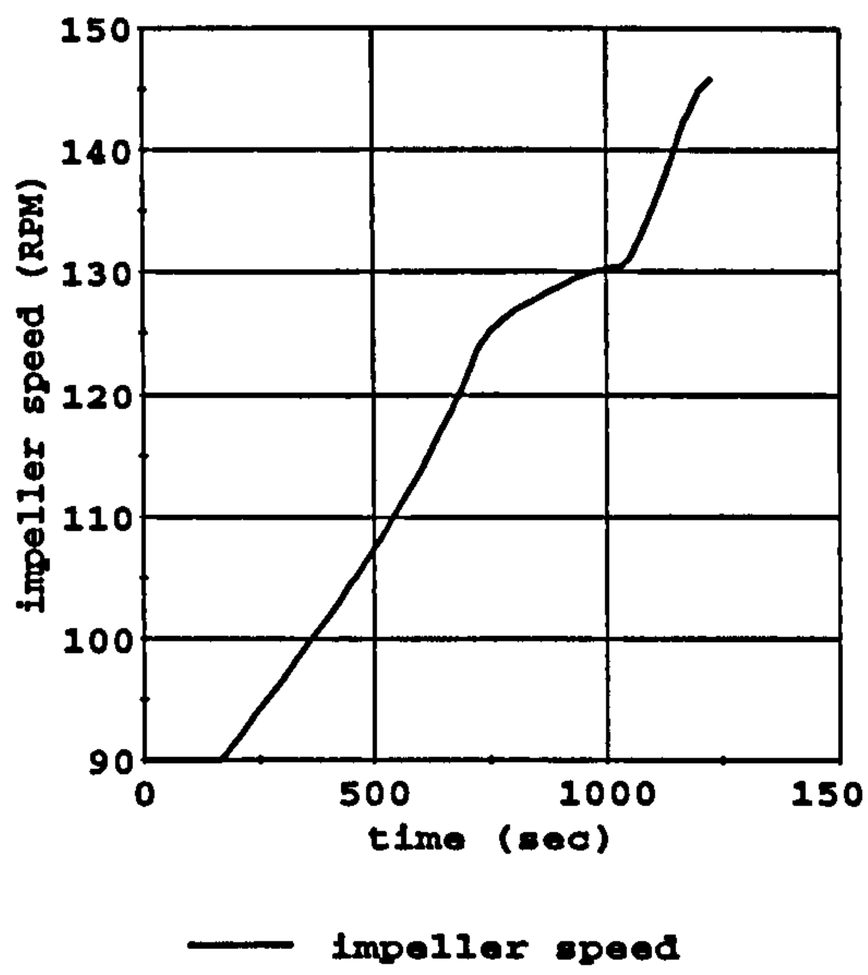


Figure 7.5: Impeller speed during simulation.

recognize situations which have been already handled and, therefore, avoid expensive CFD updates.

7.8.2 Test No.2

The second test involves a more complicated zone network model. The process, which is described in chapter 8, is a stirred tank bioreactor for the production of polymer xanthan gum. A population balance model is implemented to represent the microbial growth in the reactor. The broth rheology is modelled via a non-Newtonian correlation.

Here we want to demonstrate the effect of local models when dealing with a gPROMS-CFD zone model representing a reactor with a non-Newtonian fluid. Furthermore, we want to test the performance for a system showing a highly complicated process simulation model, too. CFD function Y requires two inputs (parameters k and n of eqn. (7.18)) to deliver the following outputs:

- mass flowrates F_{ij} and F_{ji} between each couple of neighbouring zones i and j
- pressure P_i for each zone i
- effective viscosity η_i for each zone i

The value of pressure in the tank is almost constant throughout the simulation. Furthermore the variable pressure is not used for any calculation in the gPROMS model. For that reason values P_i are computed at the beginning of the simulation and maintained constant for the rest of the simulation (Method 0 of § 7.6).

Mass flowrates are treated as piece-wise functions according to Method 1 of § 7.6. Although in the specific case it is not true that the velocity field shows little variation during the process, nonetheless such variations are not steep or sudden. If tolerance δ is chosen reasonably small, changes in mass flowrates are smooth and physically acceptable.

Local models are exploited in the computation of the effective viscosity. It was assumed that effective viscosity η_i is a *local* property, i.e. it is estimated from data for each single zone and not the entire network. In fact, effective viscosity depends on the value of viscosity parameters and shear stress in the zone and not on the connections among zones. Four local models are tested:

1. linear model $\eta_i = \alpha_i n + \beta_i k + \gamma_i$
2. quadratic model $\eta_i = \alpha_i + \beta_1 n + \beta_2 k + \gamma_i n^2 + \gamma_i k^2 + 2\nu_i nk$ as in § 7.6
3. physical model $\log \frac{\eta_i}{k} = \alpha_i (n-1)$ (eqn. (7.20))
4. physical model $\log \frac{\eta_i}{k} = \alpha_i (n-1) + \beta_i$ (eqn. (7.21))

The model effectiveness could not be compared to a base case with only rigorous CFD calculations as such calculation was not possible because of the excessive computational effort required (hence, the accuracy test cannot be performed). From this perspective, the use of local models is already successful, marking the line between a feasible simulation and unsustainable computational time. Table 7.4 shows the results of several simulations with a number of zones $nz = 20$. In this table the accuracy line testifies whether the results are smooth (G) or there are discontinuities or fluctuations which cannot be justified (B). The acceleration factor η cannot be computed either, so the total CPU time (CPU_{tot}) is reported as an additional result metric. In a zone model, the gPROMS computational time is not negligible and, thus, the number of CFD calculations is not sufficient to identify the most efficient simulation run.

	gen. loc. models		*phys. loc. models*	
CASE	**1**	**2**	**3**	**4**
C	YES	YES	YES	YES
A	G	B	G	G
$gp\%$	21.3	19.5	22.3	25.0
N_{cfd}	115	125	110	104
CPU_{tot} (s)	40000	49000	35000	32000

Table 7.4: Test2. Performance of different local models in a zone model.

N° of zones	5	10	15	20	25
gPROMS CPU time	500	2100	3700	8200	15300
CFD CPU time	23700	24000	24500	25000	26000
Total CPU time	24200	26100	28200	33200	41300
N° of gPROMS eqns.	800	2100	5700	10000	17600

Table 7.5: CPU time dependence of number of zones.

Once again the physical models are the best performing local models (Table 7.4). Accuracy is poor only in case 2. The general nonlinear model shows a rather unstable behaviour causing small discontinuities and slightly oscillating results. The gPROMS calculations account for about 20-25 % of the total CPU time. The solution of the process simulation model is more demanding and the large number of CFD outputs ask for more expensive estimation of local model parameters and solution of the problem.

Table 7.4 was derived from a simulation using a 20 zone model. The data in Table 7.5 relate the computational time to the number of zones in the model. We can observe that the CFD computational time does not vary a lot by increasing the number of zones. That happens because the number of necessary updates (due to changing physical properties) is little affected by the number of zones. On the contrary, the gPROMS CPU time drastically depends on the number of zones as that determines the size of the modelling equation system. For $nz \geq 20$ the gPROMS CPU time contributes substantially to the total computational time.

7.9 Key Results

The main achievements illustrated in this chapter are:

- The definition and development of *local model* procedures to decrease computational time in an integrated simulation. Local models are simple functions

approximating the rigorous CFD calculations. Their accuracy depends on

- ▷ the types of models which may be *physical* (simplified representations of a specific physical phenomena) or *general* (independent of the phenomena which need approximating);
- ▷ the model parameters which have to be estimated and continuously updated from previous rigorous calculations.

- The definition of procedures to ensure robustness in calculations in order to obtain a converged solution and a reliable estimation from local models. In particular it was ensured that:
  - ▷ CFD inputs are filtered to avoid noisy outputs;
  - ▷ local model parameters are estimated only if there is a sufficient number of stored rigorous calculations representative of the region being considered. Otherwise, rigorous CFD calculations are carried out;
  - ▷ outputs estimated by a local model are compared to rigorous CFD calculations representative of the same region in order to check their accuracy. If accuracy criteria are nor satisfied, rigorous CFD calculations are carried out.

- The demonstration via several tests of the effectiveness and robustness of local models in speeding-up calculations. In general, local models allow a significant acceleration in the process simulation - CFD calculations. In some cases, their use marks the line between a feasible simulation and an unsustainable computational time.

Chapter 8

A Bioreactor Simulation Example

In this chapter we will consider the application of the interface and the related procedures described in the previous chapters to a realistically complex example demonstrating the main benefits and the type of information which may be obtained by using our integration design. The example is of a bioreactor model presenting a complex rheology affecting mass transfer and the overall performance (yield, conversion, batch time, etc.).

8.1 Introduction

Bioreactors represent a wide and strategic class of processes within the process industry. Furthermore, they are a typical category for which the effect of the interactions between hydrodynamics and other phenomena is highly important on the overall performance. One of the critical parameters in an aerated bioreactor is the mass transfer coefficient between the gas (air or oxygen) and the liquid (the biological broth) phases, which has an impact on both the yield and growth of microorganisms. A review of some of the most common general correlations may be found in the books of Atkinson (1991) or in the work concerning non-Newtonian fluids by Badino Jr. *et al.* (2001). Although more correlations may be obtained to model specific systems, there is no effective approach to generalise these expressions so that uncertainty and inaccuracy remain when scale-up and scale-down are required. For instance, Enfors *et al.* (2001) report the complex response to mixing and mass transfer in large scale bioreactors: the use of general correlations cannot predict the process behaviour due to the heterogeneity of the system. Nienow (2000) reviews the critical aspects concerning mixing in the manufacture of biological products showing how tank and impeller design as well as other fluid dynamic parameters have an impact on biological performance. Hewitt *et*

al. (2000) point out that hydrodynamics may be a critical issue even at pilot plant scale because of shear stress effects. The effect of shear stress on cells is documented in several works (e.g. Papoutsakis, 1991) showing the potential damage to cells (in particular animal cells) in bioreactors due to agitation and/or aeration. An additional issue derives from the fact that many fermentation fluids exhibit very complex rheological properties (Reuss *et al.*, 1982) which heavily affect the mass transfer coefficient as demonstrated, for instance, in the work by Li *et al.* (1995) .

The lack of suitable physical models and the general complexity of the process make simulation a difficult task. It seems to us that most models and parameters available in the literature are tailored to describe specific situations: they can reproduce a well-defined process, but cannot be used to predict the process behaviour at a different scale or regime. Our method suggests a more advanced approach to the problem in order to take care of physical phenomena such as mixing and shear-stress effect which, in general, cannot be otherwise represented. Although several simplifying assumptions will still have to be accepted, the model represents a first step forward into a new modelling approach and a clear demonstrations of the capabilities of the designed integration.

8.2 The Process

The process which was chosen to test the CFD-gPROMS integration procedure is the biological production of xanthan gum. Xanthan gum is commercially the most important microbial polysaccharide (Serrano-Carreón *et al.*, 1998). The polymer is produced by cultivation of the bacteria *Xanthomonas campestris.* The success of xanthan gum is due to its very special rheological properties. Xanthan is a water-soluble polysaccharide which has found applications in a variety of industries as a viscosifying, texturising and suspending agent. The remarkable stability of the compound over a wide range of salt concentration, temperature and pH values makes it also suitable for enhanced oil recovery by polymer flooding (Peters *et al.*, 1989). For this application, high demand for low-cost xanthan is anticipated and, therefore, interest in optimising the production conditions is growing. The worldwide market is valued at between 600 and 800 million dollars per year, and growing at an annual rate of 6-7 % (Cacik *et al.*, 2001).

Batch operation is the most common mode for this process at production scale. The fermentation time may vary from 60 up to 120 hours depending on the fermentator types and process strategies. Xanthan production has been performed using a narrow temperature interval from 25 to 34 °C (Garcìa-Ochoa *et al.*, 1998). The initial medium is a water solution containing the carbon source (glucose or sucrose) and other

ingredients (mainly nitrogen). The inocolum with the microorganisms is next injected into the solution. Air (or pure oxygen) is sparged in the vessel. Usually the pII is set to an initial value of 7.0, but whether or not the pH is controlled during the process is case-dependent (Garcìa-Ochoa *et al.*, 1995). The economics of xanthan production are very dependent on the final gum concentration (Zhao *et al.*, 1991), and this, in turn, is a function of process policies (e.g. batch or fed-batch operations) together with good mixing. The limiting step is the ability to maintain the highly viscous broth in a well-mixed state: variables such as dissolved oxygen and pII are critical and bad mixing leads to poor control and, therefore, lower yields and/or lower xanthan quality (Serrano-Carreòn *et al.*, 1998). Unfortunately, xanthan gum fermentation is one of the most complex fermentation processes in terms of rheological property variations and associated mixing problems. Changes in viscosity during culture exceed four orders of magnitude (Serrano-Carreòn *et al.*, 1998). At the beginning of the process, the broth is practically water, but, as xanthan concentration increases, it becomes a very viscous pseudoplastic fluid exhibiting yield stress.

8.3 The Model

The process model describes:

- the kinetics
- the broth rheology and mass transfer

during the xanthan gum production.

8.3.1 The Kinetic Model

The approach to model microbial systems may be classified according to the number of components used in the cellular representation: cellular representations which are multicomponent are called *structured* and single component representation are designated as *unstructured* (Bailey and Ollis, 1986). Unstructured models are the simplest and most commonly adopted approach for modelling microbial systems. This type of model describes the microorganisms as an abstract "biomass". Most models developed for describing xanthan production are unstructured (Garcìa-Ochoa *et al.*, 1995, Liakopoulou-Kyriakides *et al.*, 1997, Serrano-Carreòn, 1998). We will adopt the more detailed metabolic kinetic model developed by Garcìa-Ochoa *et al.* (1998): although the description of microbial growth is carried out by means of an unstructured model, the model is metabolically structured because it takes into account the carbon source

metabolism into the cell, according to a set of reactions as a simplified scheme of the intracellular metabolism.

The kinetic model is represented by the following set of differential equations (variable and parameter descriptions are given in Table 8.1):

- The microbial growth rate:

$$r_X = \frac{dC_X}{dt} = k_X \cdot C_X \left(\frac{C_{X_0}}{Y_{XN}} + C_{N_0} \right) \left(1 - \frac{C_X}{C_{X_0} + Y_{XN} \cdot C_{N_0}} \right) \tag{8.1}$$

- The substrate (carbon source) consumption:

$$\begin{aligned} r_S = \frac{dC_S}{dt} = \quad & \alpha_1 \left[-\frac{\alpha_2}{\alpha_3} \cdot \frac{dC_P}{dt} - \alpha_4 \cdot K \cdot C_{O_2} \cdot C_X \cdot \right. \\ & \left. \left(1 - \alpha_5 \cdot \frac{1 + \alpha_6 \cdot k' \cdot C_{O_2}}{\alpha_5 + \alpha_6 \cdot Y_{ATP,P}} \right) \right] - \frac{1}{Y_{XO_2}} \cdot \frac{dC_X}{dt} \end{aligned} \tag{8.2}$$

- The xanthan gum production:

$$r_P = \frac{dC_P}{dt} = \alpha_3 \cdot K \cdot C_{O_2} \cdot C_X \cdot \frac{1 + \alpha_6 \cdot k' \cdot C_{O_2}}{\alpha_5 + \alpha_6 \cdot Y_{ATP,P}} \tag{8.3}$$

- The oxygen balance:

$$\begin{aligned} r_{O_2} = \frac{dC_{O_2}}{dt} = \quad & -\frac{\alpha_7}{\alpha_3} \cdot \frac{dC_P}{dt} - \alpha_8 \cdot K \cdot C_{O_2} \cdot C_X + \\ & k_L a \left(C^*_{O_2} - C_{O_2} \right) - \frac{1}{Y_{OX}} \cdot \frac{dC_X}{dt} \end{aligned} \tag{8.4}$$

The biomass is described as a function of nitrogen source initial concentration C_{N_0}. Oxygen concentration is not considered in the biomass kinetics: Garcìa-Ochoa *et al.* (1995) show that *Xanthomonas campestris* is not very sensitive to the oxygen mass transfer and can keep on growing even in anaerobic conditions (although not for very long). However, if there is not enough dissolved oxygen, xanthan production does not occur.

The other equations are derived from modelling the xanthan production from glucose according to the cell metabolism (ATP cycle according to the work by Pons *et al.*, 1989). Some correlations may be found to express some of the kinetic parameters as a function of temperature (Garcìa-Ochoa *et al.*, 1998). However, in our model the system is considered perfectly isothermal (28 °C). The assumption is perfectly reasonable because of the very small heat generation in the process. Additionally, pH too is supposed to be uniform and constantly equal to 7.

Symbol	Code	Description	Value
C_1	P	Geometric parameter	2×10^{-3}
C_2	P	Initial oxygen mass transfer coefficient (s^{-1})	20
C_X	V	Biomass concentration ($g \cdot l^{-1}$)	0.053 (I)
C_S	V	Substrate concentration ($g \cdot l^{-1}$)	37 (I)
C_P	V	Xanthan concentration ($g \cdot l^{-1}$)	0 (I)
C_{O_2}	V	Dissolved oxygen concentration ($g \cdot \cdot l^{-1}$)	2.5×10^{-4} (I)
C_{X_0}	V	Initial biomass concentration ($g \cdot l^{-1}$)	0.053
C_{X_0}	V	Initial nitrogen concentration ($g \cdot l^{-1}$)	0.25
$C_{O_2^*}$	P	Equilibrium dissolved oxygen concentration ($g \cdot l^{-1}$)	2.5×10^{-4}
k	V	Index in the power law model ($Pa \cdot s^n$)	
K	P	Kinetic constant ($g^{-1}h^{-1}$)	79.9
k'	P	Model constant (mol^{-1})	5.77×10^4
k_X	P	Specific biomass growth rate ($g^{-1}h^{-1}$)	0.535
k_La	V	Oxygen mass transfer coefficient (s^{-1})	
n	V	Index in the power law model	
N	V	Impeller agitation speed (s^{-1})	
V_s	V	Superficial air velocity ($m \cdot s^{-1}$)	1.1×10^{-3}
w	P	Weight	5
$Y_{ATP.P}$	P	Macroscopic yield of ATP into xanthan gum	34
Y_{XN}	P	Macroscopic yield of nitrogen into biomass	6.073
Y_{XO_2}	P	Macroscopic yield of oxygen into biomass	239.03
Y_{XS}	P	Macroscopic yield of substrate into biomass	6.073
α_1	P	Stoichiometric coefficient	180
α_2	P	Stoichiometric coefficient	5.94
α_3	P	Stoichiometric coefficient	923.2
α_4	P	Stoichiometric coefficient	1/12
α_5	P	Stoichiometric coefficient	3.58
α_6	P	Stoichiometric coefficient	4
α_7	P	Stoichiometric coefficient	0.3
α_8	P	Stoichiometric coefficient	0.5
η	V	Effective viscosity ($Pa \cdot s$)	

Table 8.1: Parameters (P) and variables (V) used in the xanthan gum fermentation model (Garcìa-Ochoa *et al.*, 1998). Values are given for parameters, assigned variables, initial variables (indicated by I).

8.3.2 The Mass Transfer Model

The mass transfer coefficient k_La depends on

a. effective viscosity of the broth η

b. mechanical power input (which is function of the impeller agitation speed N)

c. superficial gas (air) velocity V_s

Several correlations have been suggested to estimate the value of the mass transfer coefficient (see, e.g., Garcìa-Ochoa, 2000 and Garcìa-Ochoa and Gòmez, 1998). The following correlation is used for this work:

$$k_La = C_1 \cdot V_s^{2/3} \cdot N^2 \cdot \eta^{-2/3} \,. \tag{8.5}$$

Correlation (8.5) is a function of macroscopic parameters in the reactor. The version used of the CFD simulator Fluent is not capable of dealing with multiphase non-Newtonian fluids. The available choice stands between multiphase fluid modelling and non-Newtonian fluid modelling. As it will be later clarified, the critical issue in modelling this process is the complex rheological behaviour of the broth. Hence, we preferred a higher precision in the description of the broth rheology rather than the multiphase model. Thus, we assume that the superficial air velocity V_s is constant and uniform throughout the domain and equal to 1.1×10^{-3} ms^{-1} (Garcìa-Ochoa and Gòmez, 1998).

The second important parameter is the mechanical power input (related in expression (8.5) to the square power of the impeller speed N). The use of the impeller speed N is a rather approximate measurement of the power transferred to the fluid. For instance, the distribution of the energy of dissipation in the vessel would deliver a more detailed and more precise information about energy in the fluid. Furthermore, this information is usually available from any CFD simulator. Unfortunately, a correlation adopting the energy of dissipation or other hydrodynamic variables in the calculations of the mass transfer coefficient is not available. It was decided to keep the macroscopic value N even for our zone based modelling approach.

The third parameter is the effective viscosity η. Thomson and Ollis (1980) describe the evolution of broth rheology by means of a power law relation of the type of eqn. (7.18):

$$\eta = k\dot{S}^{n-1}. \tag{7.18}$$

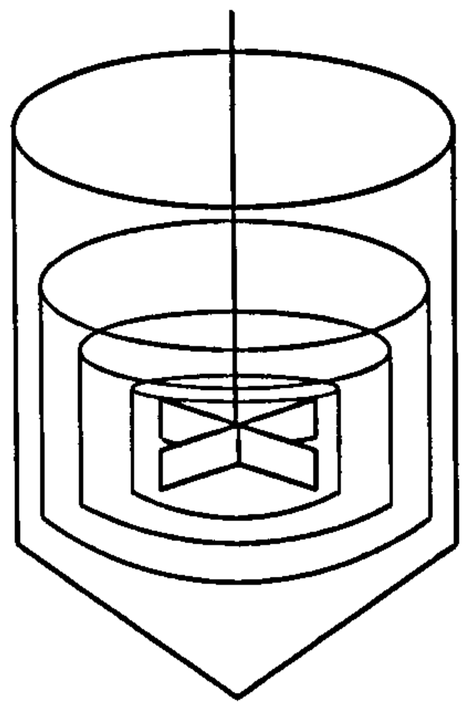


Figure 8.1: Representation of the *concentric boxes* model (Serrano-Correòn *et al.*, 1998).

Values of parameters k and n are estimated from experimental data: they appear to be very sensitive to the concentration of polymer in the solution. Serrano-Carreòn *et al.* (1998) developed a more complicated model suggesting the existence of an apparent yield stress τ_0 in the xanthan gum solution at polymer concentration higher than 10 g/l (Bingham type behaviour). Apparent yield stress increases with the xanthan concentration. Their model assumes a "concentric boxes" behaviour in the reactor. As illustrated in Figure 8.1, the well-mixed volume or active volume is considered as a region delimited by the apparent yield stress for a given xanthan concentration. The model suggested by Serrano-Carreòn *et al.* takes into account a radius r which determines the volume of the active reactor and which is a function of the effective viscosity. The moving fluid is described by a power-law correlation. Garcìa-Ochoa and Gòmez (1998) suggest the use of the Casson model (k_c is a model constant):

$$\tau^{1/2} = \tau_0^{1/2} + k_c \dot{S}^{1/2} \; . \tag{8.6}$$

This is then simplified by neglecting the apparent yield stress τ_0 which is assumed to be negligible at standard operating condition. Cacik *et al.* (2001) again adopt correlation (7.18) to describe rheology in the tank and propose the following correlations to estimate indices k and n in the power-law equation:

$$\begin{aligned} \log k &= 0.07327 + 1.49319 \log C_P + 0.2763 (\log C_P)^2 \\ n &= 0.12865 + 0.4675 \exp(-0.5002 C_P) + 0.4156 \exp(-7.79 C_P) \end{aligned} \tag{8.7}$$

We followed this last approach to model the biological broth. In fact, the use of a power-law model to describe rheology in the system seems to be widely accepted in the literature. The use of different correlations presents several complications mainly

because of convergence issues: some attempts with a Bingham-fluid model revealed severe problems in the Fluent solver capability of obtaining a converged solution.

Eqn. (8.5) is then used adopting a *local* value of effective viscosity as computed by CFD calculations. In their work, Li *et al.* (1995) observe that at the beginning of the process, the mass transfer coefficient remains constant or even increases, notwithstanding an increase in the system viscosity. This phenomenum, which seems due to initial interactions between the broth (cells plus xanthan gum) solution and the air, lasts till the xanthan concentration is less than 1 g/l. This effect was incorporated in eqn. (8.5) derived by data in the paper by Li *et al.* (1995) to give:

$$k_L a = C_2 \cdot \frac{1}{1+\exp[-w(1-C_P)]} + C_1 \cdot V_s^{2/3} \cdot N^2 \cdot \eta^{-2/3} \frac{1}{1+\exp[w(1-C_P)]} \cdot \quad (8.8)$$

8.4 The Integrated Model

The CFD-process simulation integration approach is used to simulate the xanthan gum production in a batch stirred reactor. According to the design developed in chapter 3 the overall model is split into a CFD model based on a fine grid and a process simulation model represented by a network of zones. The system is batched and, accordingly, there are no environment zones.

8.4.1 The Process Simulation Model

All zones are *internal* and each zone model is constituted by the following set of equations:

- Composition balances:

$$V\frac{dC_i}{dt} = \frac{1}{\rho}\sum_{j=1}^{ni} F_j^{in} C_{ij}^{in} - \left(\sum_{j=1}^{no} F_j^{out}\right) C_i + V r_i \quad (8.9)$$

where $C_i = C_X, C_S, C_P, C_{O_2}$.

- Mass transfer equation:

$$k_L a = C_2 \cdot \frac{1}{1+\exp[-w(1-C_P)]} + C_1 \cdot V_s^{2/3} \cdot N^2 \cdot \eta^{-2/3} \frac{1}{1+\exp[w(1-C_P)]} , \quad (8.8)$$

- Physical property equations:

$$\begin{aligned} \log k &= 0.07327 + 1.49319 \log C_P + 0.2763 (\log C_P)^2 \\ n &= 0.12865 + 0.4675 \exp(-0.5002 C_P) + 0.4156 \exp(-7.79 C_P) \end{aligned} \quad (8.7)$$

Density ρ is assumed to be constant ($\rho = 1000$ g/l) and uniform throughout the domain (Garcìa-Ochoa and Gòmez, 1998). The inputs **x** which are passed to the CFD model (variable `FluidProperty` in model `InternalZone`) are the parameters n and k described by eqns. (8.7). The Fluent package is not capable of accepting distributed values (i.e different for each cell) of n and k. Thus, the values of this properties are volume averaged along the domain:

$$n = \frac{\sum_{i=1}^{nz} n_i V_i}{\sum_{i=1}^{nz} V_i} \quad \text{and} \quad k = \frac{\sum_{i=1}^{nz} k_i V_i}{\sum_{i=1}^{nz} V_i}$$

The assumption does not affect the overall results since the mixing time (process lasts 2-4 days) is capable of coping with local variations of composition, even in a system demonstrating such a complicated rheology. Local variations in the effective viscosity depend on the agitation within the vessel, i.e. on the shear stress distribution. The great difference in terms of time scales between biological reactions and fluid flow phenomena also justifies the model partition and the decoupling of process equations (which are, thus, treated as *weakly coupled*).

8.4.2 The CFD model

The reactor is the double impeller stirred tank described in chapter 2. The steady-state CFD model contains about 34000 computational cells in a structured grid. The non-Newtonian power-law model expressed by eqn. (7.18) is used to describe the broth rheology. The model delivers the mass flowrates between zones and the effective viscosity as outputs **Y**.

8.4.3 Zone Topology

Zones are defined according to effective viscosity distribution. In fact, effective viscosity is by far the most important property, since mixing is less critical because of the long residence time. The system evolves during the process and viscosity values change a lot as xanthan concentration increases. Figure 8.2 describes the system evolution from a situation at low xanthan concentration (1 g/l) to an advanced processing stage at high polymer concentration (10 g/l). It is interesting to observe how the velocity field changes. When viscosity is relatively small the fluid moves throughout the whole domain, although velocity magnitude is greater where viscosity is lower. As xanthan concentration increases the system starts behaving according to the "concentric box" model proposed by Serrano-Carreòn *et al.* (1998): there is mass transportation only in the region close to the impeller; elsewhere the fluid is almost still. An analysis of

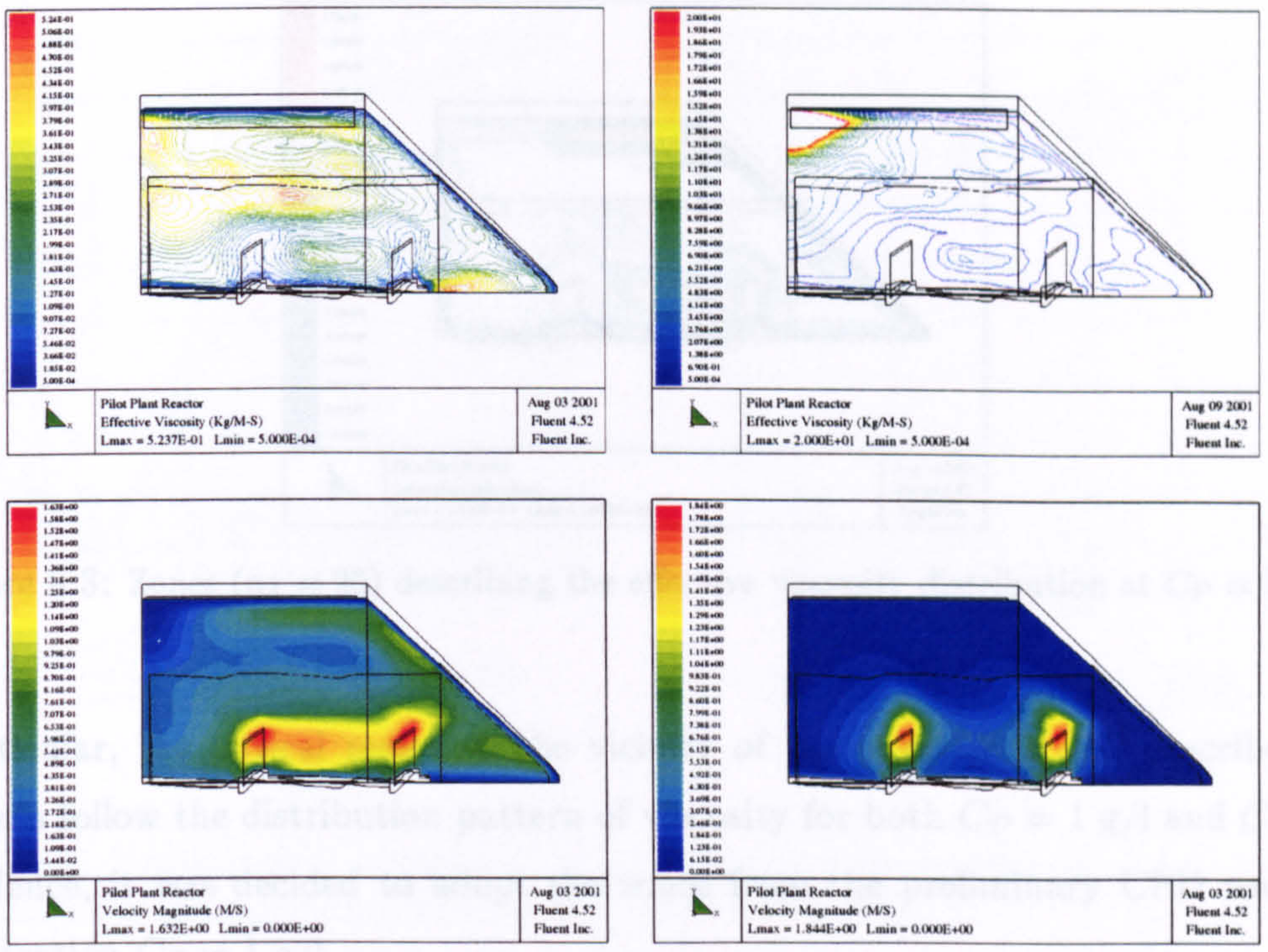


Figure 8.2: Viscosity (top) and velocity (bottom) distribution at a xanthan concentration equal to 1 g/l (left) and 10 g/l (right).

effective viscosity confirms the velocity results, since effective viscosity is low in the impeller region and much higher near the tank walls.

The great change in physical properties may pose some problems in the definition of the zone network, since zones should be able to describe regions with different viscosities over the entire process. However, Figure 8.2 shows that the shape of the viscosity distribution does not change greatly. Algorithm Θ_Δ (§ 5.3.2) is used to automatically set up a network of zones. Two CFD simulations at xanthan gum concentration $C_P = 1$ g/l ($\Rightarrow \quad k = 1.07, \; n = 0.4123$) and $C_P = 10$ g/l ($\Rightarrow \quad k = 24.34, \; n = 0.1670$) are run. Each of the resulting flow fields is used to initialise the algorithm Θ_Δ and to set up a network of zones based on the effective viscosity distribution corresponding to that specific concentration. The tolerance $\Delta\eta$ defines the number of zones at a given concentration. Some tests demonstrated that the two flow fields produce very similar zone networks as long as the number of zones is the same. Therefore, the network of zones used for the simulation was not derived from a combination of two or more flow fields at different xanthan gum concentrations, but was set up using a single flow field. Figure 8.3 illustrates the zone shapes ($nz = 20$) in the same tank section as displayed in Figure 8.2, which were obtained for the case $C_P = 1$ g/l. We can verify that the zone network is capable of reproducing the general behaviour of viscosity distribution.

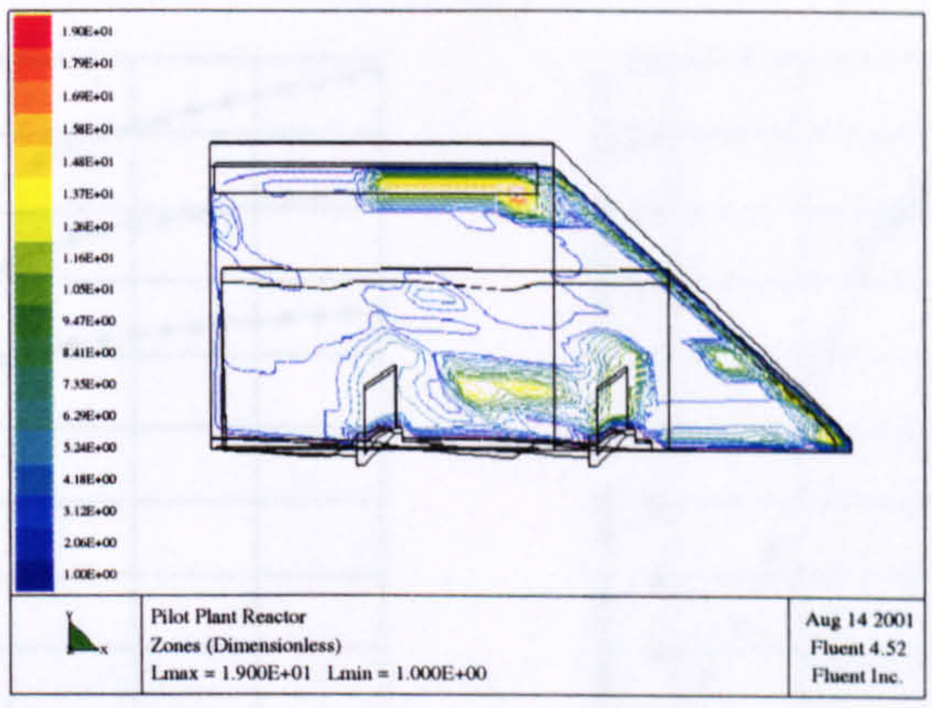


Figure 8.3: Zones ($nz = 20$) describing the effective viscosity distribution at $C_P = 1$ g/l.

In particular, the critical region in the vicinity of the impeller is well described and the zones follow the distribution pattern of viscosity for both $C_P = 1$ g/l and $C_P = 10$ g/l. Hence, it was decided to adopt the zones from the preliminary CFD results at concentration $C_P = 1$ g/l.

8.5 The Simulation

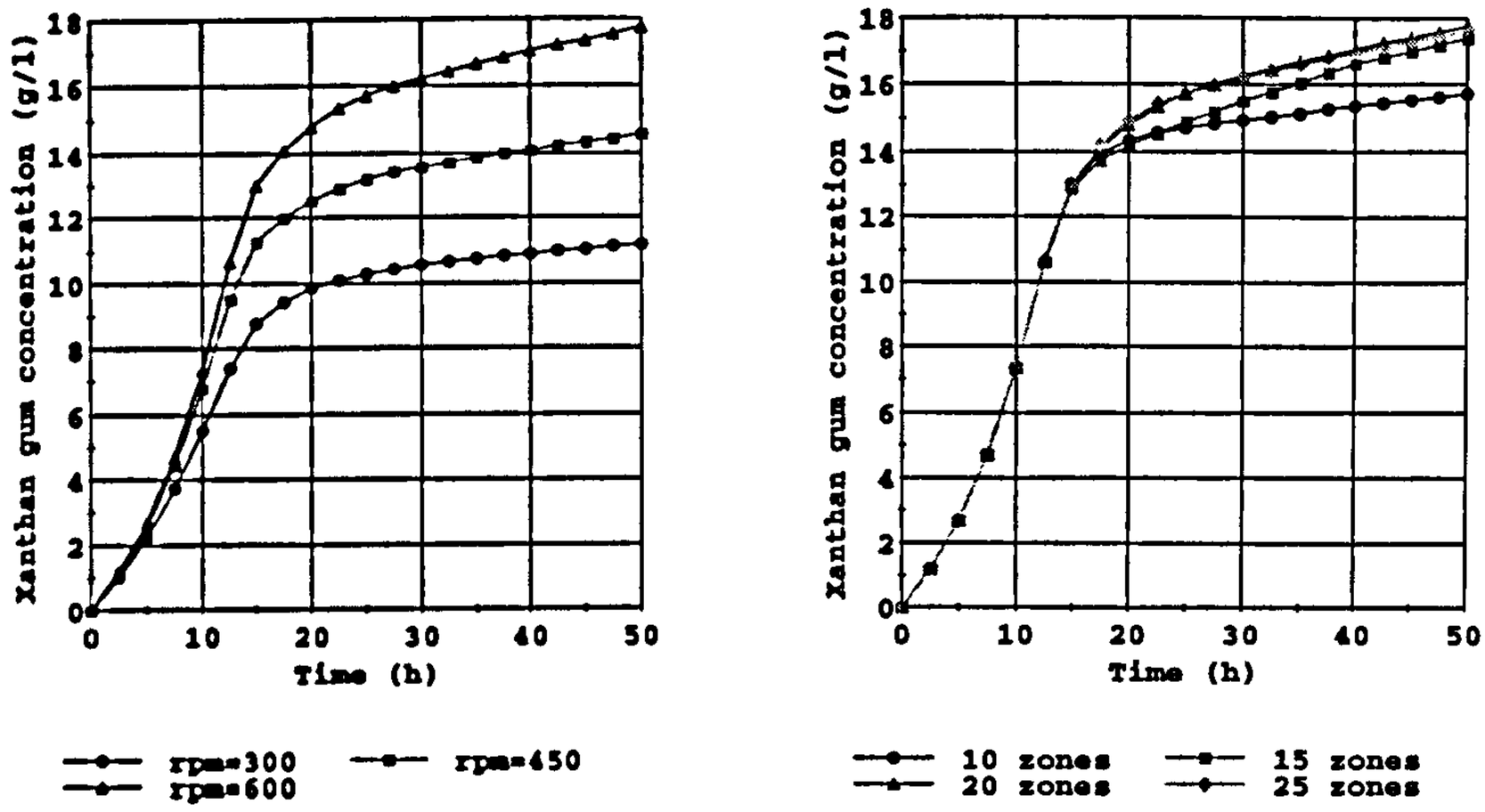
The process simulation model takes care of kinetics and mass transfer equations. It computes the values of parameters n and k in the power-law eqn. (7.18) and passes them the CFD model via the linking procedures. The CFD model simulates the flow field corresponding to the values n, k received by gPROMS and returns mass flowrates F^{in}, F^{out} between zones and effective viscosity η for each zone.

Local models are implemented to accelerate computations. As demonstrated in § 7.8.2, local model (7.21)

$$\log \frac{\eta}{k} = \alpha(n-1) + \beta \tag{7.21}$$

is used to approximate the effective viscosity η with the best results in terms of efficiency and accuracy. Mass flowrates are treated as piece-wise functions according to Method 1 of § 7.6. The process is batch and no operating procedures are implemented. The simulation stops after 50 hours.

Three operating conditions are considered representing different impeller rotation speeds (RPM $= 300, 450, 600$). Figure 8.4(a) shows the effect of mixing on xanthan gum production. As expected, the greater the rotation speed of the impeller the higher the polymer yield, since mass transfer is both directly and indirectly (via viscosity)

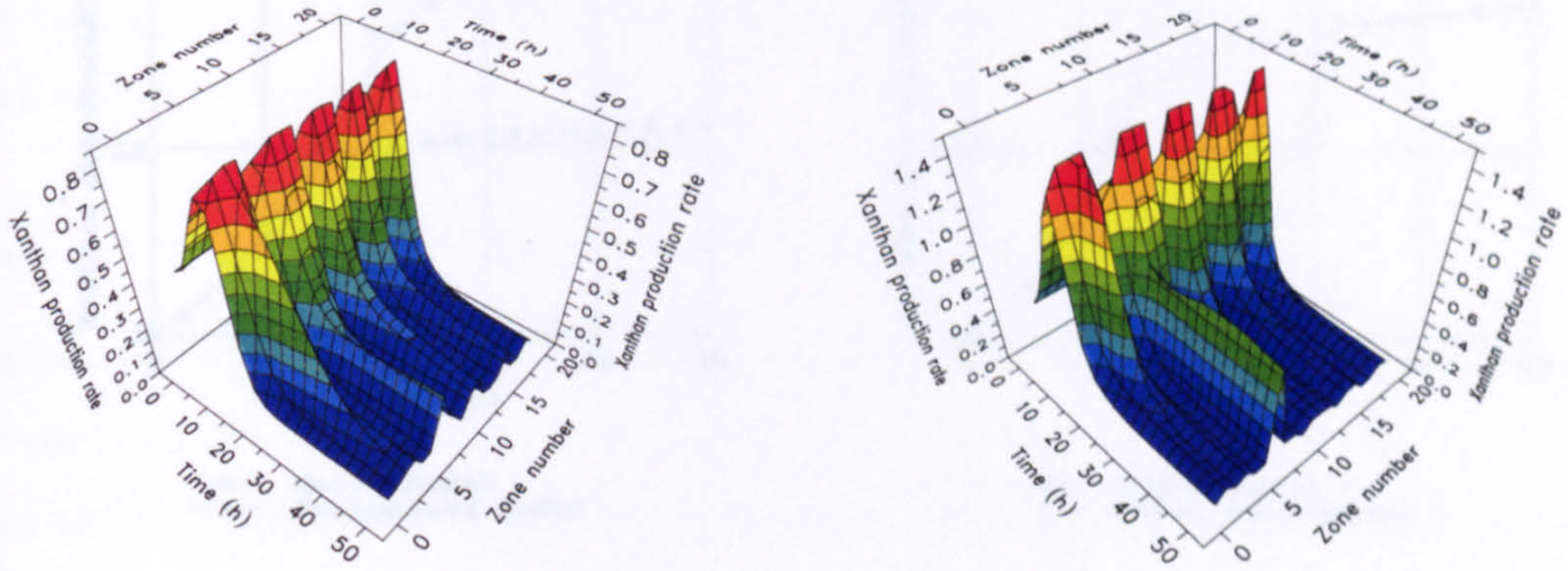


(a) (b)

Figure 8.4: Xanthan gum production. In (a) comparison among different rotation rates. In (b) comparison among different number of zones at RPM= 600.

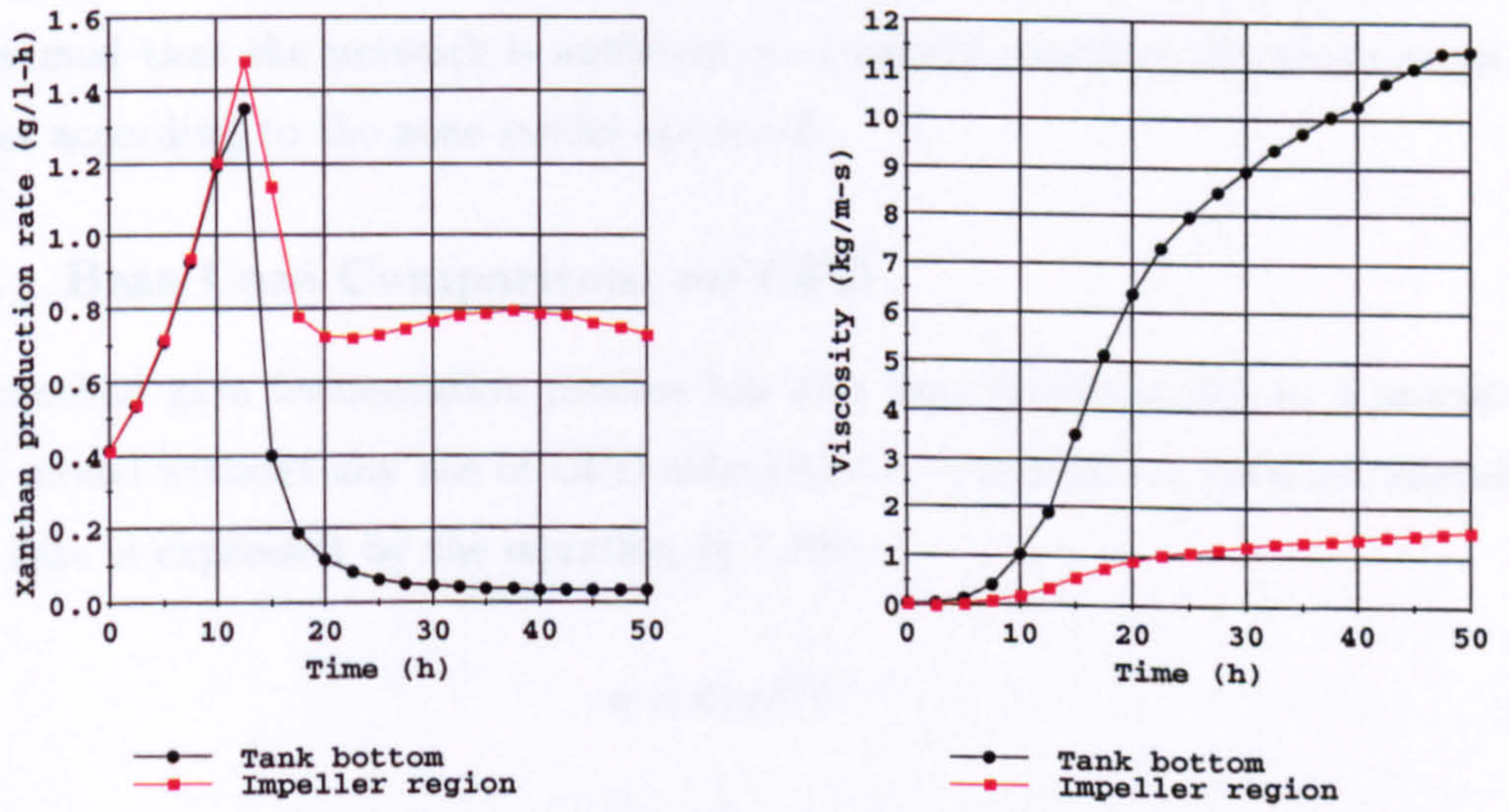
affected by the agitation in the vessel. The effect of viscosity on xanthan gum production is illustrated in Figure 8.5 which shows the xanthan gum production rate at two different agitation rates (RPM= 300 and RPM= 600). The *Zone number* axis takes into account the local variations in different zones. Production rates present significant local variations depending on the local viscosity. At the beginning of the process most zones demonstrate a great increase of the rate due to the growing biomass (although some peripheral zones soon stop being productive). After about 10 h the increasing viscosity makes the rate dramatically drop throughout the domain except in those zones representing the region surrounding the impeller. That is even more evident at high rotation speed (Figure 8.5b compared to 8.5a). Figure 8.6 shows the difference in the effective viscosity and, as a result, in the xanthan gum production rate between a zone in the impeller region and one at the bottom of the tank (RPM= 600). We can notice that after an initial surge, followed by a sudden drop (at time $t = 10$ h), the production rate (as well as the viscosity) remains fairly constant in the impeller region. On the contrary, in the tank bottom the rate steadily decreases to nearly zero production.

Figure 8.4b illustrates the relation between number of zones and results (at RPM= 600). By increasing the number of zones we are able to refine the resolution of regions at different mixing regimes and, accordingly, improve the simulation. Too few zones (e.g. 10 zones) may not capture the property distribution and may merge together regions demonstrating a different behaviour with respect to the phenomena of interest.



(a) (b)

Figure 8.5: Xanthan gum production rate in the vessel at RPM= 300 (a) and RPM= 600 (b).



(a) (b)

Figure 8.6: Xanthan gum production rate (a) and effective viscosity (b) in two different regions of the tank at RPM= 600.

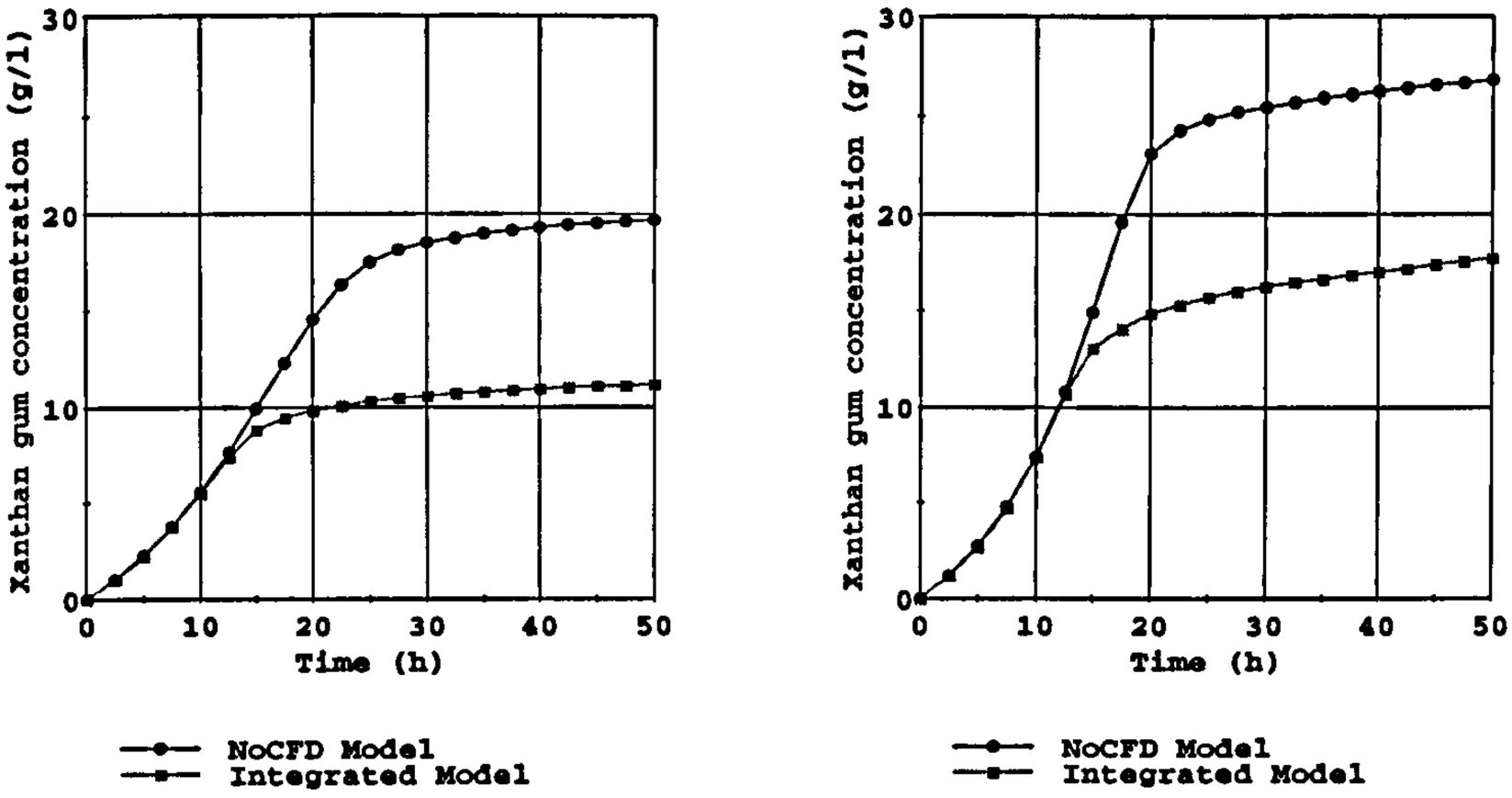


Figure 8.7: Xanthan gum production simulation. Comparison between a model where viscosity is estimated by means of eqn. (7.19) (no CFD) and a 20 zone gPROMS-CFD model (Integrated). RPM= 300 (a) and RPM= 600 (b).

In this case, we may observe that a limited number of zones may underestimate the xanthan gum production. By increasing the number of zones, the model gradually approximates a limiting solution so that simulations at $nz = 20$ and $nz = 25$ are practically coincident. This is also a way to assess the "right" number of zones nz: if increasing nz does not cause significant changes in the simulation results, then it may be assumed that the network is sufficient to properly describe the phenomena in the process according to the zone model approach.

8.5.1 Base Case Comparison: no CFD

The xanthan gum fermentation process has also been implemented as a process simulation model without any use of CFD calculations. The relation between viscosity and shear rate is expressed by the equation (§ 7.19):

$$\eta = k(aN)^{n-1} \ . \tag{7.19}$$

Other equations (kinetics and mass transfer) are the same as in the integrated model. Figure 8.7 compares the simulated xanthan gum concentration according to this model with the concentration resulting from a 20 zone gPROMS-CFD model. Results demonstrate a clear difference in the two models in predicting the effective viscosity in the stirred tank (and hence conversion). At the beginning of the process, the two models

produce very similar results, but, as soon as viscosity effects become essential in predicting the products of the fermentation, results start diverging. In the CFD-gPROMS model, some regions present a very high viscosity which does not allow any mass transfer between the gas phase and the cells. Only regions close to the impeller remain productive; the remaining zones of the reactor soon become a very viscous and improductive broth with little reaction.

8.5.2 A Simulation with Variable Agitation

Previous simulations have demonstrated that high rotation speed improves xanthan gum yield in the batch reactor. However, it is not possible to use very high rotation speed at the beginning of the fermentation since that affects xanthan gum production because of cell damage (Garcìa-Ochoa *et al.*, 2000). As soon as viscosity increases, the cells of *Xanthomonas campestris* become rather insensitive to shear: that is probably due to a viscous layer of broth which is gradually formed around the cells protecting them (Peters *et al.*, 1989). Usually initial fermentation conditions utilise an impeller rotation speed of about 200-300 rpm, which is then gradually increased to 700-1200 rpm (Garcìa-Ochoa *et al.*, 1998, Serrano-Carreòn *et al.*, 1998). This process procedure was not taken into account in the previous simulations. In view of the above, simulations at RPM= 450 and RPM= 600 represent unrealistic situations since very high initial agitation speed may cause cell damage.

A model has been implemented considering a variable rotation speed for the impeller, as a function of average effective viscosity in the tank $\bar{\eta}$ calculated as a volume average over all zones:

$$\bar{\eta} = \frac{\sum_{i=1}^{nz} \eta_i V_i}{\sum_{i=1}^{nz} V_i} \tag{8.10}$$

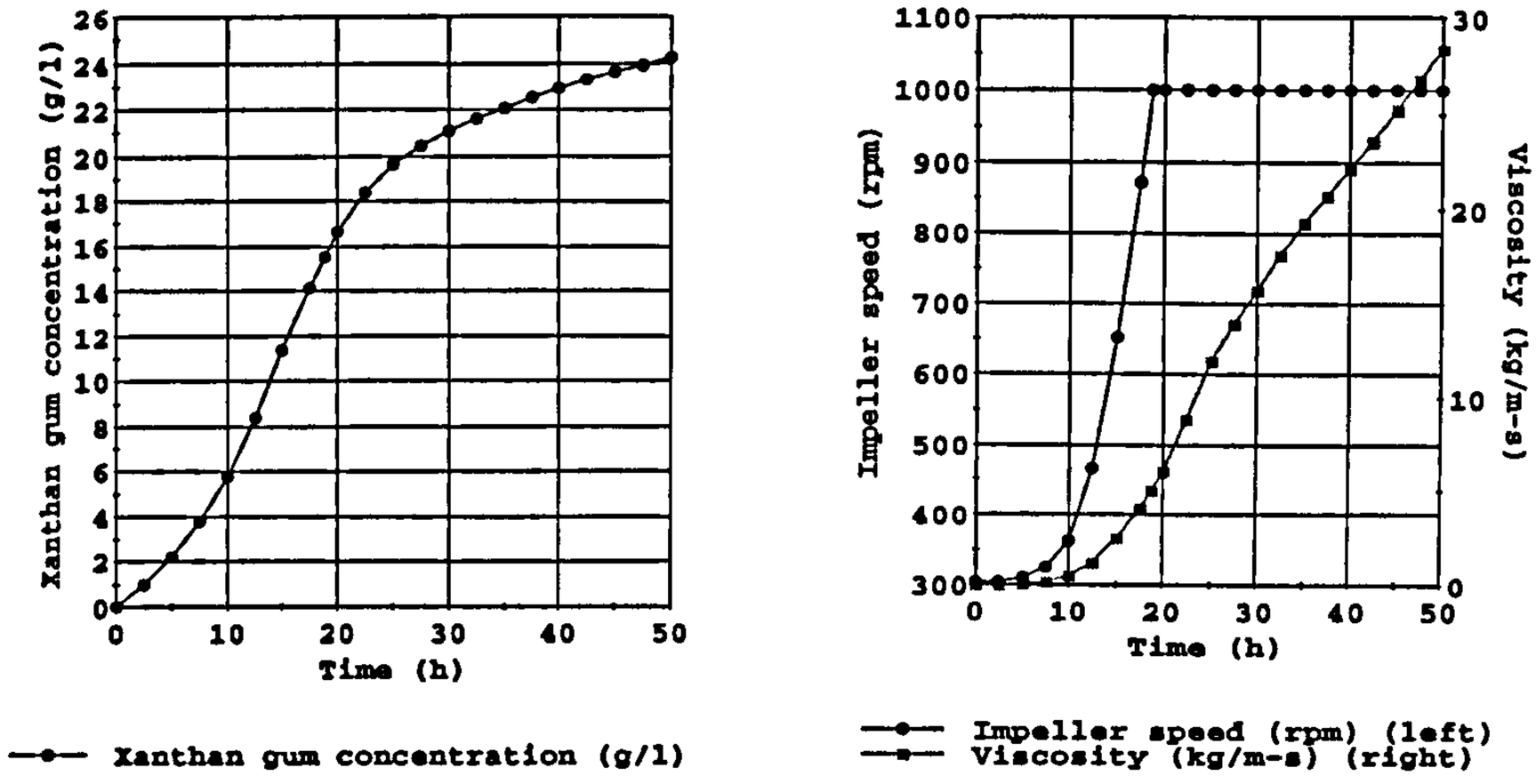
according to the correlation:

$$N = \begin{cases} N_{min} + \dfrac{(N_{max} - N_{min})}{\bar{\eta}_{max}} \cdot \bar{\eta} & \text{if } \bar{\eta} < \bar{\eta}_{max} \\ N_{max} & \text{if } \bar{\eta} \geq \bar{\eta}_{max} \end{cases} \tag{8.11}$$

where $N_{min} = 300$ rpm, $N_{max} = 1000$ rpm and $\bar{\eta}_{max} = 5$ Pa $\cdot$ s.

Local model (7.21) has to be modified to take into account the variable speed of agitation. That is done by considering the correlation:

$$\eta = k(aN)^{n-1} \tag{7.19}$$



(a) (b)

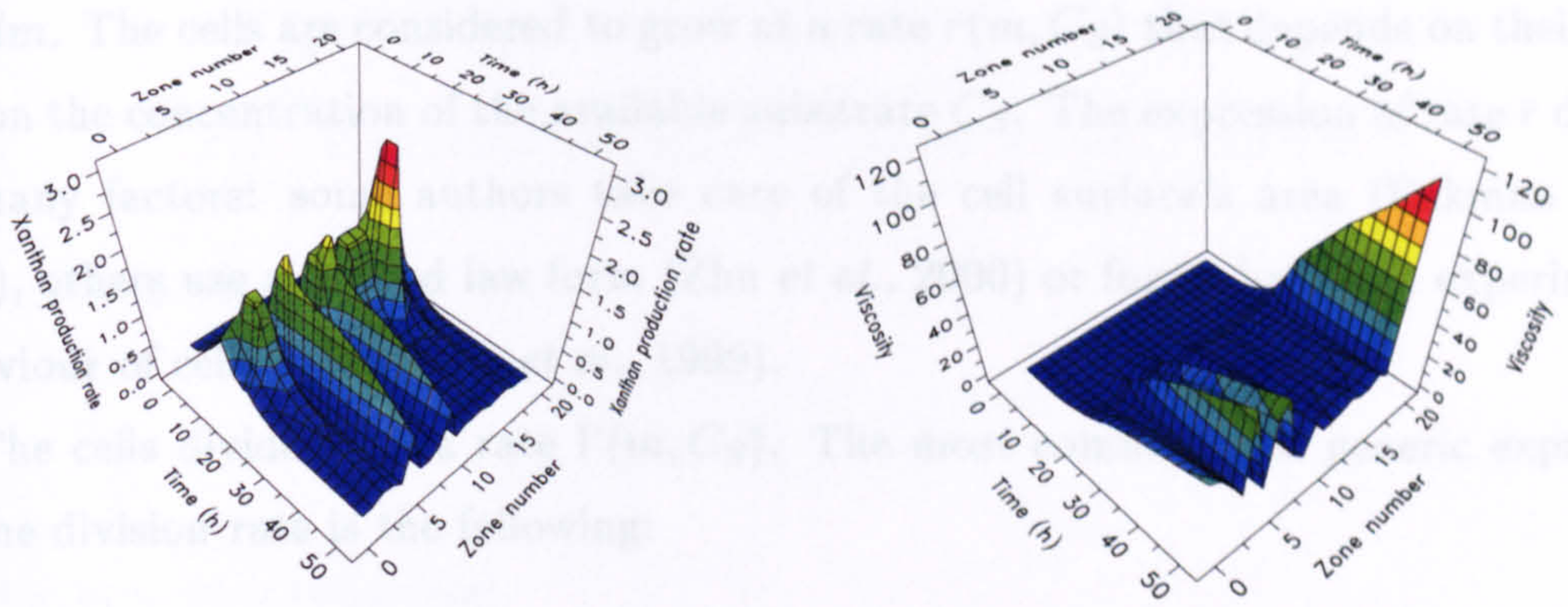
Figure 8.8: Xanthan gum production at variable agitation: simulation results of xanthan gum concentration (a); agitation speed and viscosity (b).

which is transformed into the linear model:

$$\log \frac{\eta}{k} = \alpha(n-1) + \beta(n-1)\log(N) \ . \tag{8.12}$$

Eqn. (8.12) proved good estimation capabilities in approximating effective viscosity at variable agitation. Variations in the rotation speed are communicated to the CFD model by means of the `CFDParameter` vector (see chapter 3).

Figures 8.8 and 8.9 illustrate some of the simulation results. In Figure 8.8a we can see that the xanthan gum concentration keeps growing until the end of the fermentation thanks to the increasing agitation. In Figure 8.8b the relation between viscosity and agitation speed is illustrated: maximum speed is reached after about 20 h and maintained till the end of the simulation. The effect of high final agitation is shown well by Figure 8.9. Several zones maintain a high productivity until the end of the simulation, although the agitation is not sufficient to maintain a reasonable production rate in the whole domain. Some zones presenting an initial intense xanthan production suddenly fall into ineffectiveness because of mutating fluid dynamic conditions. The behaviour of the effective viscosity within the tank (Figure 8.9b) demonstrates the fact that some regions within the reactors move towards complete immobility, while others maintain a reasonably low viscosity.



(a) (b)

Figure 8.9: Xanthan gum production at variable agitation: xanthan production rate (a) and viscosity (b) throughout zone domain.

8.6 A Population Balance Model

In this section we want to demonstrate the high flexibility of the suggested integration design. A population balance model is implemented within the internal zone model without changing anything in the network ($nz = 20$) and CFD models. The use of population balance models is used in other important fields in the process industry. For instance, crystallisation processes may be properly modelled only by means of a crystal population balance model. The size of crystals depends on fluid dynamic properties such as shear stress and the energy of dissipation. Therefore, the approach can certainly be extended to crystallisation modelling.

All models so far considered the bacteria population as a uniform mass called biomass treated as a chemical species produced according to certain kinetics. An alternative approach is to use a structured population balance model: cellular populations are treated as distinct individuals which can be differentiated from each other. In recent years some papers have demonstrated the viability of such an approach and its usefulness in handling complicated bio-systems: e.g., Zhu *et al.* (2000) and Godin *et al.*(1999) have used cell population balance models to deal with biological systems exhibiting oscillating or periodic behaviour. All these models are based on some works which appeared in the 1960s and early 1970s and in particular on the fundamental paper by Eakman *et al.* (1966) defining the general equations for the description of a population of cells in a mixed vessel. We will consider the formulation of those

equations recently given by Mantzaris *et al.* (1999) .

Let $W(m,t)$ be the number of cells which at time t have a mass between m and $m+dm$. The cells are considered to grow at a rate $r(m,C_S)$ that depends on their mass and on the concentration of the available substrate C_S. The expression of rate r depend on many factors: some authors take care of the cell surface's area (Eakman *et al.*, 1966), others use a Monod law form (Zhu *et al.*, 2000) or forms based on experimental behaviour of cells (Mantzaris *et al.*, 1999).

The cells divide with a rate $\Gamma(m,C_S)$. The most common and generic expression for the division rate is the following:

$$\Gamma(m,S) = \frac{f(m)}{1-\int_0^m f(m')dm'} r(m,C_S) \ , \tag{8.13}$$

where $f(m)$ is the division probability density function which is assumed to depend only on the cell mass and to be defined as a left-hand side truncated Gaussian distribution with set mean μ_f and standard deviation σ_f:

$$f(m) = \frac{1}{\sqrt{2\pi\sigma_f^2}} \cdot \exp\left[-\frac{(m-\mu_f)^2}{2\sigma_f^2}\right] \ . \tag{8.14}$$

During binary cell division, assuming no loss of cell mass during the division process, the mass of the parent cell must be divided between the two daughter cells. The partition of the mother cells into two daughter cells is described by the partitioning function $P(m,m',C_S)$, which expresses the probability that a mother cell of mass m' will give birth to a daughter cell of mass m. As is common in the literature, we assume the partitioning function to be independent of substrate concentration. As in the paper by Mantzaris *et al.* (1999), the function is taken to be a symmetrical beta distribution defined by the following equation:

$$P(m,m') = \frac{1}{B(q,q)}\frac{1}{m'}\left(\frac{m}{m'}\right)^{q-1}\left(1-\frac{m}{m'}\right)^{q-1} \tag{8.15}$$

where q is a preset parameter. A symmetrical beta distribution is defined by the expression:

$$B(q,q) = \int_0^1 u^{q-1}(1-u)^{q-1}du \ . \tag{8.16}$$

Finally, it is assumed that the effect of cell death is negligible. If the system is batch, then the cell population dynamics can be modelled by a set of two integro-differential

equations. The first is the cell population balance:

$$\frac{\partial W(m,t)}{\partial t} + \frac{\partial}{\partial m}[r(m,C_S)W(m,t)] = 2\int_m^\infty \Gamma(m',C_S)P(m,m')W(m',t)dm' - \Gamma(m,C_S)W(m,t) \tag{8.17}$$

subject to the initial condition:

$$W(m,0) = W_0(m) \tag{8.18}$$

and the boundary condition:

$$W(0,t) = 0\ . \tag{8.19}$$

The second is the mass balance on the substrate:

$$\frac{dC_S}{dt} = \frac{1}{Y}\int_0^\infty r(m,C_S)W(m,t)dm \tag{8.20}$$

subject to the initial condition

$$S(0) = S_0\ . \tag{8.21}$$

The physical meaning of eqn. (8.19) is that there exist no cells of zero mass at any time. Several suggestions exist for the initial distribution $W_0(m)$ (e.g. Subramanian and Ramkrishna, 1971): in our case, the initial condition is taken to be a left-hand side truncated Gaussian with the mean of μ_{in} and the standard deviation of σ_{in}.
Eqn. (8.17) consists of four terms:

1. the accumulation term: $\frac{\partial W(m,t)}{\partial t}$
2. the growth term: $\frac{\partial}{\partial m}[r(m,C_S)W(m,t)]$
3. the division term: $\Gamma(m,C_S)W(m,t)$
4. the birth term: $2\int_m^\infty \Gamma(m',C_S)P(m,m')W(m',t)dm'$

The division term describes the loss of cells of mass m due to their division into two daughter cells of smaller masses. The birth term represents the production of cells of mass m through the division of mother cells of greater masses.

The gPROMS package is capable of solving several types of partial integro-differential equations thanks to its methods for dealing with distributed systems. However, eqn.

(8.17) cannot be reduced to any of these types. The reason for that is that gPROMS can handle only integral functions of type (e.g. Symeonidis, 1997):

$$f(v) = \int_{v}^{v_{max}} g(u)du \tag{8.22}$$

which may be replaced by a function $\psi(v,t)$ defined via the differential equation:

$$\frac{\partial \psi}{\partial v} = g(v) \quad \forall v \in [v, v_{max}) \tag{8.23}$$

and the boundary condition:

$$\psi(v_{max}, t) = 0 \ . \tag{8.24}$$

Eqn. (8.17) shows a different structure because of the partitioning function $P(m,m')$, which is function of both m and m'. The problem may be solved by not using a distributed form for the population balance model, but replacing eqns. (8.17) and (8.20) with discretised equations:

$$\frac{dW(m_i,t)}{dt} = \begin{cases} 0 \quad \text{if } i = 1 \\ \dfrac{r(m_i,C_S)W(m_i,t) - r(m_{i-1},C_S)W(m_{i-1},t)}{\Delta m} + \\ 2\sum_{j=i}^{nm} \Gamma(m_j,C_S)P(m_i,m_j)W(m_j,t)\Delta m - \\ \Gamma(m_i,C_S)W(m_i,t) \quad \text{if } i = 2,n \end{cases} \tag{8.25}$$

and

$$\frac{dC_S}{dt} = \frac{1}{Y}\sum_{1}^{nm} r(m_i,C_S)W(m_i,t)\Delta m \tag{8.26}$$

where nm is the number of intervals used to discretise cell mass.

The model above was implemented within each internal zone. In order to implement the cell population balance model several parameters and the initial distribution $W_0(m)$ are required. Unfortunately, these are not available for this process. For this simulation and the purpose of demonstrating the flexibility of our design, we adopted parameters from a different system. The only paper containing parameters and initial conditions for a process describing a real system is presented in the work by Godin *et al.* (1999) for a self-cycling fermentation of *Acinetobacter calcoaceticus.* Since gPROMS

μ_f	σ_f	μ_{in}	σ_{in}	Y	T_d (h)
0.575	0.125	0.2875	0.0675	0.5	5

Table 8.2: Values of parameters used in the cell population balance model (Mantzaris *et al.*, 1999).

was not able to cope with the different scales of modelling equations (from $\mathcal{O}(10^{-13})$ to $\mathcal{O}(10^{25})$) suggested in the paper, the solution was found by adopting the normalised parameters suggested in the paper by Mantzaris *et al.* (1999). Thanks to those parameters and initial conditions modelling equations describe variables in a range of orders of magnitude between $\mathcal{O}(10^{-4})$ and $\mathcal{O}(10^{5})$.

Results cannot be compared to those illustrated in the other sections of this chapter since the population balance model is based on data derived from a different system. However, we observe that parameters and initial conditions are chosen in order to return realistic values for averaged concentration over mass and number of cells. The cell growth rate was also taken from Mantzaris *et al.* (1999) (no dependency on substrate C_S):

$$r(m_i) = \frac{\log 2}{T_d} m_i \tag{8.27}$$

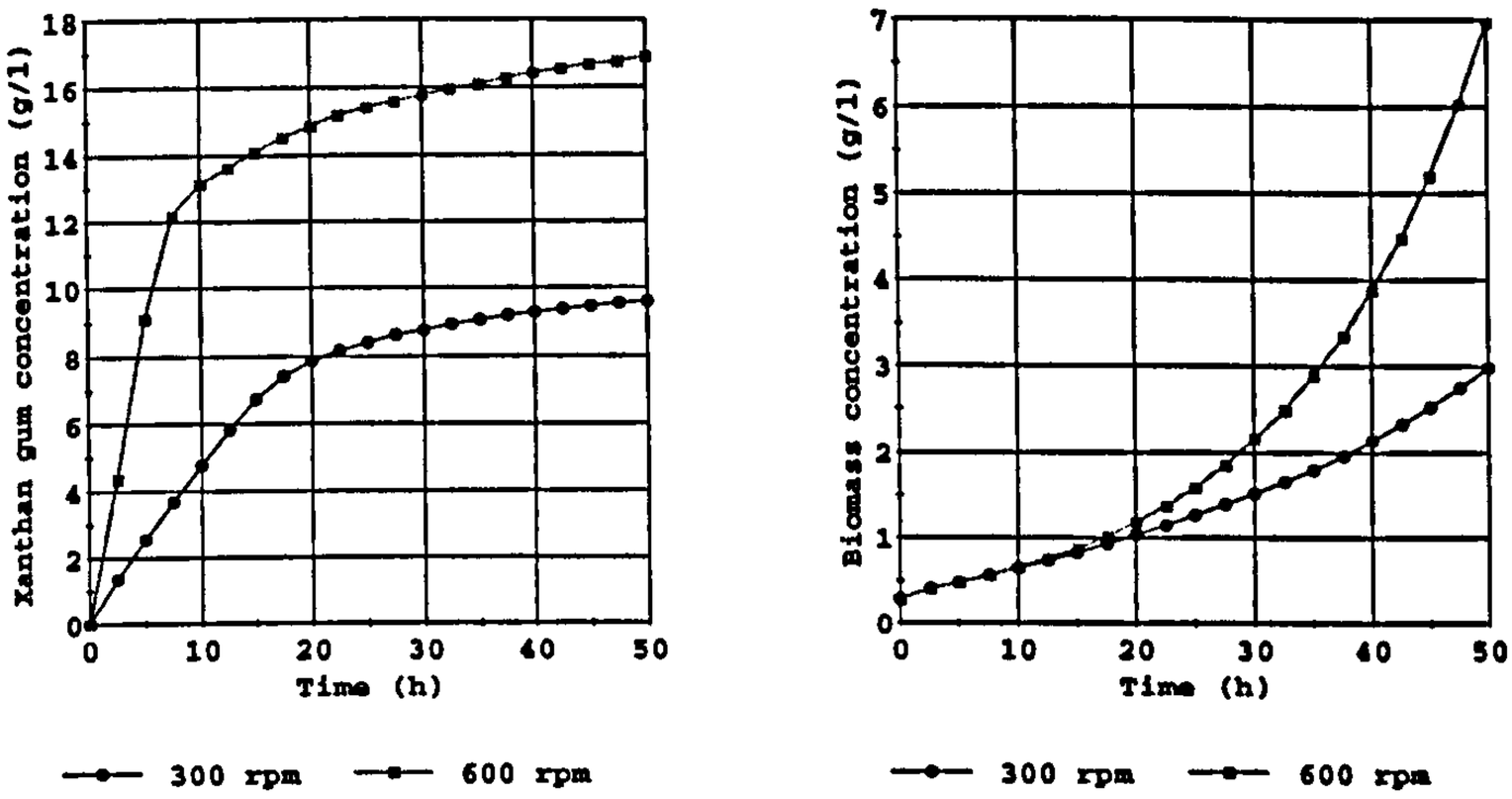
where T_d is the average doubling time of the population. Eqn. (8.27) has been modified in order to consider an oxygen influence on the cell growth. This is not true for a xanthan gum process, but, since it is a phenomenum common to most aerobic processes, it has been decided to observe the effect of mass transfer on a cell population balance. Eqn. (8.27) was therefore turned into:

$$r(m_i) = \frac{1}{2} \cdot \frac{\log 2}{T_d} m_i \left(1 + \frac{\eta_{in}}{\eta}\right) \tag{8.28}$$

Values of all parameters used in the model are contained in Table 8.2. The rest of the model is identical to the xanthan gum model previously described. Biomass concentration is calculated from the first moment of the cell mass distribution:

$$C_X = \sum_{1}^{nm} W(m_i, t) m_i \Delta m \ . \tag{8.29}$$

Two simulations have been carried out at different impeller rotation speeds, RPM= 300 and RPM= 600 respectively. Figure 8.10 shows the results concerning biomass and



(a) (b)

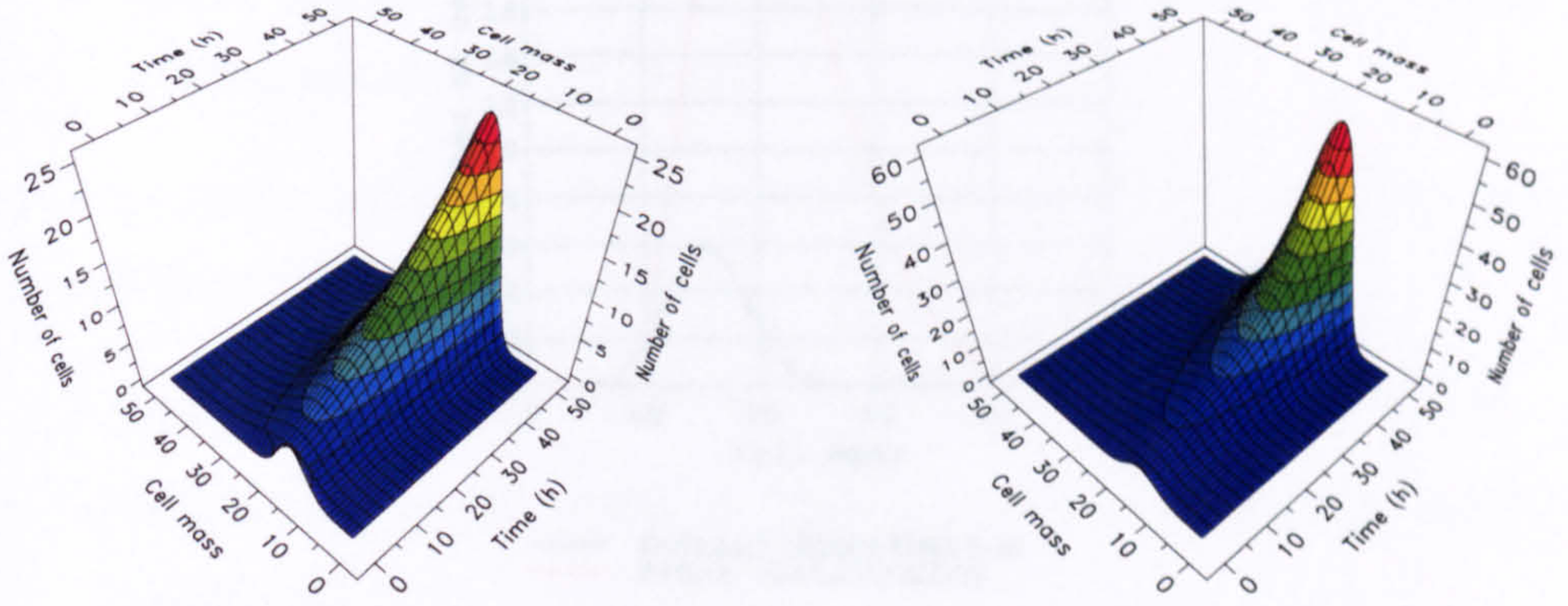
Figure 8.10: Comparison between xanthan gum (a) and biomass (b) concentration at RPM= 300 and RPM= 600. A cell population balance model is used in each zone.

xanthan gum concentration during the process according to the population balance model. Biomass concentration computed from eqn. (8.29) demonstrates the effect of mass transfer variation on the cell growth.

The results displayed in Figure 8.11 show the effect of oxygen mass transfer on the growth and distribution of cells. The greater impeller speed increases the growth rate and, thus, the biomass in the reactor. Figure 8.12 shows the plot in Figure 8.11a from different angles so as to illustrate the evolution of cell mass during the process. Given an initial distribution of cells, the cells go through growth and division phenomena to reach the final distribution (clearly illustrated in Figure 8.13). The number of cells initially moves towards a wider distribution: the peak of the distribution diminishes (b), while cells start growing. After a critical average mass is reached the cell division becomes important in representing the cell distribution in the tank. Thanks to division, cells multiply and their number greatly increases. The cell mass average as well as the number of "big" cells slightly increase throughout the process. The shape of the distribution curve depends on the division rate $\Gamma(m, C_S)$ and on the division probability function $p(m, m')$.

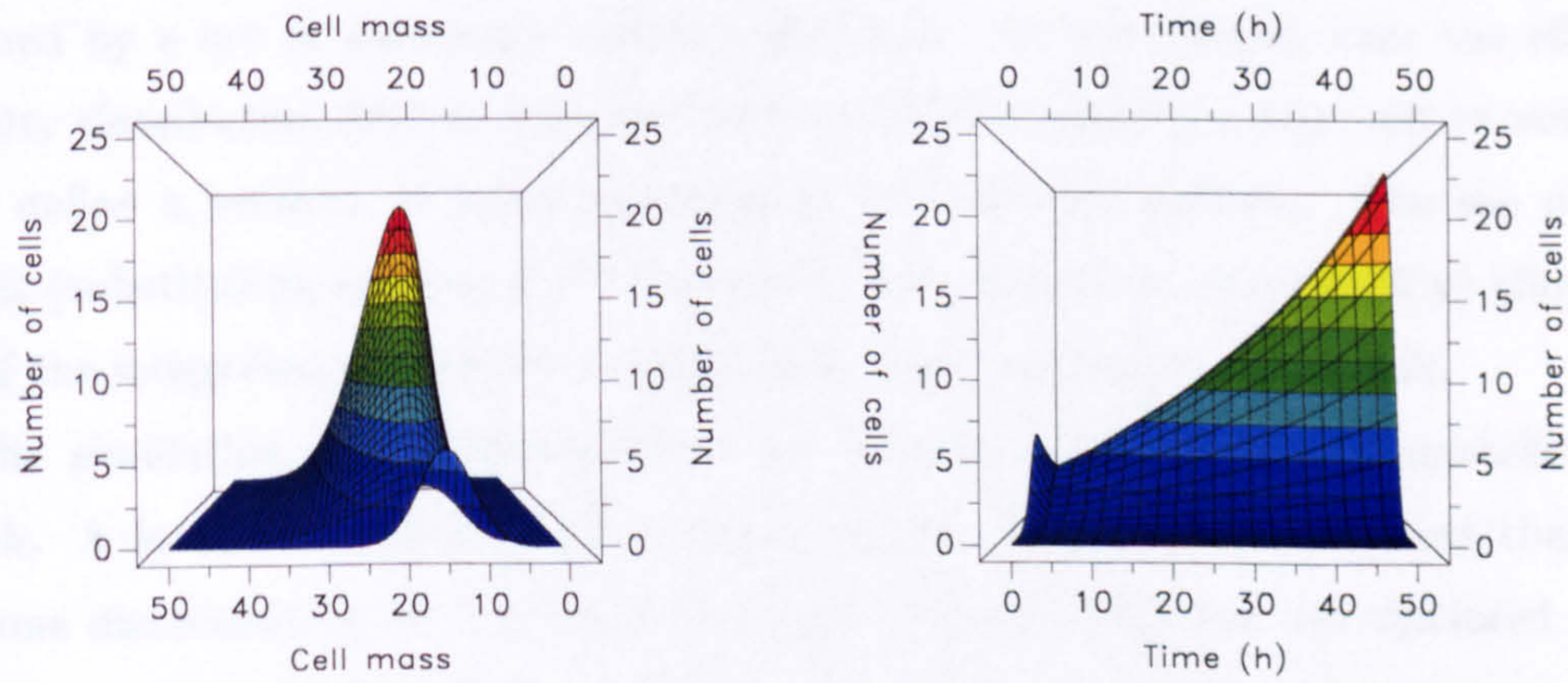
8.7 Conclusions and Key Results

This chapter has demonstrated the interface capability of dealing with complex models demanding robust process simulation methods for a numerical solution of the model



(a) (b)

Figure 8.11: Distribution of cell mass at RPM= 300 (a) and RPM= 600 (b).



(a) (b)

Figure 8.12: Evolution of mass of cells during the process.

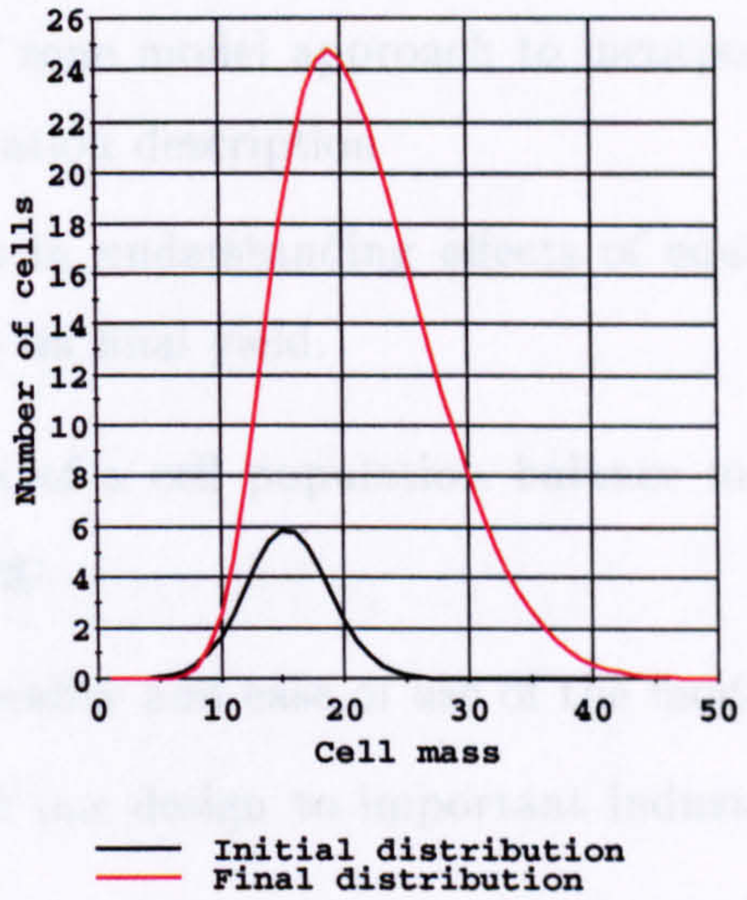


Figure 8.13: Initial and final cell mass distribution.

equations. The interface design allows a clear partition of

a. process modelling equations

b. CFD equations

by means of the *internal* and *environment* zone models, and, of course, the CFD model. The correspondence between the two models is achieved via the network model exchanging the variables $\mathbf{x}$ and $\mathbf{Y}$ and defining the zone model topology. This can be obtained by a set of *automatic zoning* algorithms. In this specific case the effective viscosity distribution derived from preliminary CFD calculations was used to automatically define a network of zones representing the viscosity pattern. The use of local models (substituting rigorous CFD outputs $\mathbf{Y}$ with approximate outputs $\mathbf{y}$) allows the run of the integrated simulation in reasonable time (see results in § 7.8.2).

The simulation results demonstrate the benefits of an integrated modelling approach. A local description of the critical process phenomena is obtained thanks to the zone discretisation of the process domain. New possibilities are disclosed in the area of process control: the monitoring of hydrodynamic variables such as shear stress, energy of dissipation may be used to improve the yield and quality of the products. For instance, if there existed an expression relating the shear stress (or the energy of dissipation) to the cell damage and the mass transfer coefficient, then a control strategy could be implemented to optimise the agitation during the xanthan gum production.

We can summarise the main achievements illustrated in this chapter as:

- The implementation of a bioprocess model demonstrating:
  - ▷ applicability of zone model approach to incorporate hydrodynamics within a process simulation description
  - ▷ unique benefits in understanding effects of equipment characteristics (e.g. rotation speed) on final yield.
- The implementation of a cell population balance model within the *same* zone model demonstrating:
  - ▷ flexibility, generality and ease of use of the modelling architecture
  - ▷ extendibility of our design to important industrial processes such as crystallisation.

Chapter 9

Final Comments and Future Research

This thesis has considered the design and implementation of a general interface to integrate computational fluid dynamic simulators and process modelling tools. A general architecture and methods for exchanging critical variables between the two packages to obtain a converged solution of a combined model has been demonstrated. Furthermore, a procedure has been defined for easily defining and setting up process simulation models capable of mapping the CFD models by identifying homogeneous, well-mixed *zones*. In this chapter a summary of the modelling, design and simulation, constituting the contribution of the thesis, is presented. This is followed by a discussion of the main achievements and some suggestions for future research.

9.1 Summary

Computational fluid dynamics and process simulation are well established tools for the design and optimisation of chemical processes. State-of-the-art process modelling tools are capable of handling complex DAEs systems, PDAEs within rectangular domains and optimisation. Unfortunately, they are not able to deal with complex domains and to produce satisfactory results when complex equations concerning fluid flow models need solving. On the other side, CFD tools are based on very tailored numerics specifically designed to deal with momentum and mass balance equations in a complex geometry. However, they cannot handle systems of stiff differential equations (e.g. complex kinetics) and lack the flexibility to interface with other libraries and models.

This is best understood within the wider context of the multiscale modelling. The widely different scales of both space and time involved in process engineering pose

serious problems to the definition and solution of suitable mathematical models. Process simulation and CFD are techniques which have been designed to tackle problems at different scales. Nonetheless, in many practical cases it has become necessary to use an approach combining the two technologies in order to best describe processes where mixing and fluid flow behaviour interact with physical and chemical phenomena. Chapters 2 and 3 have defined a general method for scale integration between CFD and process simulation. *Parallel* integration has been achieved by introducing a *master* package capable of handling information between the two packages. In this thesis the master program is the process simulation package itself thanks to the open architecture of gPROMS and its flexibility in interfacing to external software.

The main contribution for the design of the interface is based upon the idea of describing the fluid flow behaviour through a subdivision of the domain in the process simulation model. The domain subdivision produces a network of *zones* representing physical regions in the process equipment. A single zone is considered well-mixed and homogeneous. The transient behaviour of each perfectly-mixed zone is modelled in the process simulation package, and so is the behaviour of the entire network. The main function of the CFD package is to compute the total mass flowrates between the zones and properties depending on hydrodynamics within each zone. The interaction between the integrated model and the external environment (represented by other models, mass and energy input/outputs) is handled by defining a number of *environment* zones representing some kind of connection gates.

Chapter 4 has been dedicated to the solution of a specific problem arising from our definition of zone model, i.e. the apparent incompatibility between constant volume zones and variable density in the fluid domain. The issue has been solved by analysing the structure of the set of DAEs describing a reactor and the differentiation index of the system, and by changing the degree of freedom of the system. In particular, incompressible fluids at varying density may still be handled by dropping one of the flowrates computed by the CFD model for each process simulation zone. An algorithm has been developed for automatically identifying those flowrates and avoid potential singularities in the model solution.

Chapters 5 and 6 have dealt with the practical issue of setting criteria to define the zones. The problem has been first considered in terms of the interface design. General procedures have been identified to pass zoning parameters between the CFD and process simulation models. Then, a number of methods have been defined for an automatic generation of zones based on criteria of homogeneity and connectivity among cells in the CFD grid. The effectiveness of those methods and the implemented zoning

algorithms has been verified through a number of mixing tests. These experiments have been incorporated within a more general view for designing integrated models and setting up a network of zones: a number of procedures have been identified to obtain the best network of zones depending on the specific problem.

The issues of efficiency and robustness of the integration between CFD and process simulation have been considered in chapter 7. CFD calculations require a very heavy computational burden, which may become unbearable in a dynamic simulation demanding continual updating. The use of alternative simpler models (*local models*) to substitute rigorous CFD calculations appears to be a good way to overcome the problem. Physical local models, i.e. models approximating CFD calculation by means of correlations describing the same phenomena in a simplified manner, and general local models, i.e. models which may be applied independently of nature of the problem, have been presented and compared to each other. A number of criteria to ensure robustness of calculations and reliability of results has been described and implemented.

Finally, a dynamic model of a typical batch reactor has been presented to demonstrate the main advantages of the approach and the benefits obtained. A batch bioprocess for the production of the biopolymer xanthan gum has been simulated. The complex rheology is captured by means of a non-Newtonian CFD model, and mass transfer and kinetics through a process simulation model. Zones were automatically identified in order to capture the important effect of shear stress on viscosity and, accordingly, on mass transfer and conversion. A cell population balance was also introduced to demonstrate the capability of our integrated design of dealing with complicated process simulation models and obtaining a converged solution, and the ease of substituting simulation models of various complexity within the same framework.

9.2 Achievements

This thesis represents the first practical attempt to improve process design and simulation by integrating process modelling and CFD tools in a generic way. The following conclusions can be drawn:

- A general interface has been designed to handle a parallel integration of process simulation and computational fluid dynamics (*model partitioning*). In particular:
  - ▷ variables and parameters are exchanged between the process simulation and CFD packages in a automatic way

▷ process simulation and CFD models are continuously updated to obtain results representing the dynamics of the system under examination

▷ a model independent interface design has been defined for interfacing CFD and process simulation models

▷ process simulation model design defines a network of zones (coarse grid) overlapping the fine CFD grid in order establish a *local* correspondence between the two models.

- A theoretical analysis has been carried out to deal with systems of variable density. A solution based on the reduction of the differentiation index of the set of DAEs has been proposed. An algorithm has been designed for the automatic solution of the problem.

- A procedure and language have been designed to easily define zones in a model. In particular:

  ▷ information regarding zone topology and connections are exchanged in a easy and general form (CFD package independent)

  ▷ zones may be set up automatically according to criteria ensuring homogeneity and well-mixing

  ▷ zone network parameters are passed to the integrating interface without user intervention

  ▷ methods based upon mixing tests and knowledge of critical process properties have been designed to set up the most convenient network of zones.

- *Local model* procedures to decrease computational time in an integrated simulation have been defined and implemented. Procedures to ensure robustness in calculations have also been described and implemented.

- Actual feasibility of the approach and resulting benefits have been demonstrated via realistically complex dynamic examples in the bioengineering area.

9.3 Future Research

It is the fate of most research projects to open more new problems than solutions. Several achievements have been obtained in this work, but there are many areas which need further investigation.

9.3.1 Modelling Issues

In chapter 2 it was observed that the suggested integration procedure can be used to model *weakly-coupled* systems, but not the *strongly-coupled* ones. It is necessary to understand more about these two classes of problems. No general procedure was defined to identify whether a process is described by a set of equations which are weakly coupled or not. Engineering common sense, analysis and process knowledge (e.g. the time scale of the process phenomena) are powerful tools, but a more structured and scientific categorisation is required.

On the other side, the handling of *strongly-coupled* systems is a task which cannot be avoided. In some cases, solutions may be found through the incorporation of modelling equations and robust solvers within CFD packages. This may be considered as a dual approach to the one proposed in this thesis. Process modelling tools may be used to solve a specific set of equations within CFD models. A second and more complicated approach consists of a full integration of different scales within a single model and a single solver, according to the definition of *simultaneous* integration given in chapter 2. The cited work of Neumann (2001) is a first step in this direction.

However, even within the weakly-coupled system class considered here there are important modelling issues which have not been explored yet. The most important ones concern:

- the modelling of multiphase processes (e.g. fluidised beds, vapour-liquid systems)
- the use of dynamic re-zoning (which may be required in processes going through a variety of hydrodynamic states)
- the handling of transport by diffusion[1] in a zone network model
- greater flexibility and automation in handling initial and boundary conditions in the zone network interface

9.3.2 Numerical Issues

The use of physical local models has been demonstrated to be very useful in accelerating calculations without affecting convergence towards the solution. However, this was done only for a couple of the most important process variables (heat transfer coefficient,

[1] Zones exchange fluxes through common interfaces. These fluxes are always defined by means of a convective property (e.g. mass flowrate). Presently, diffusion may be taken into account either by setting a fixed diffusion term or by introducing some transfer coefficient to exchange properties between zones.

effective viscosity). A library of physical local models should be created to exploit their accuracy in as many cases as possible. Furthermore, the efficiency and robustness procedure needs more investigation to improve the general response of local models. That may be done by:

a. introducing some kind of adaptivity to the rigorous model (e.g. possibility to automatically change local models during simulation);

b. using statistical methods to improve the predictive capabilities of models

The use of local models or general interpolating procedures may become essential for optimisation or sensitivity analysis problems. In such cases, the use of rigorous calculations may be computationally prohibitive and need replacing with database analysis and estimation techniques.

Further work should be invested to tackle the problems of data *aggregation* and *disaggregation* between process simulation and CFD. According to our approach a process simulation zone represents a set of CFD computational cells. The effect of alternative averaging techniques to aggregate CFD cell values into a process simulation zone and the search for effective methods to distribute zone data over a number of cells have not been considered in detail in this thesis.

Next, it is necessary to investigate the error propagation in the suggested approach and quantify the effect of:

- the model partitioning

- the use of steady-state CFD calculations

- the use of an approximate local model.

Finally, the use of a parallel solution should be considered to further speed up calculations. That may involve some re-thinking of the model structure and present procedures to handle the flux of information between the two packages.

9.3.3 Scale-up and Process Design

Further research in a variety of application areas is needed to investigate the benefits of the integration approach proposed for process scale-up and design. More case studies and verification of results against experimental work are required to establish the class of problems which may be addressed by the framework developed in this thesis. These experiments and simulations should give more details towards the definition of a design

where simulation and numerical techniques are used to optimise the process through a synergetic use of available technologies. Some application areas include:

- some types of polymerisation processes (e.g. suspension polymerisation)
- crystallisation processes
- bubble columns and fluidised beds
- novel control strategies based on fluid flow variables.

Appendix A

The Least Square Method

The linear least squares problem can be stated as follows: *given a real* $m \times n$ *matrix* $\mathbf{A}$ *of rank* $k \leq min(m, n)$ *and given a real* m*-vector* $\mathbf{b}$, *find a real* n*-vector* $\mathbf{x}_0$ *minimizing the eucledian length of* $\mathbf{Ax} - \mathbf{b}$.

Algorithms which have been implemented in this local model approach effect an orthogonal decomposition of matrix $\mathbf{A}$ in the form $\mathbf{HRK}^T$ (where $\mathbf{H}$ and $\mathbf{K}$ are orthogonal matrices) and then calculate a LS solution. An important property of orthogonal matrices is the preservation of eucledian norm after multiplication. Thus, for any m-vector $\mathbf{y}$ and any $m \times m$ orthogonal matrix $\mathbf{Q}$

$$\|\mathbf{Qy}\| = \|\mathbf{y}\|. \tag{A.1}$$

In LS, this means that

$$\|\mathbf{Q}(\mathbf{Ax} - \mathbf{b})\| = \|\mathbf{Ax} - \mathbf{b}\| \tag{A.2}$$

and minimizing $\|\mathbf{Ax} - \mathbf{B}\|$ is equivalent to the minimization of $\|\mathbf{Q}(\mathbf{Ax} - \mathbf{b})\|$.

Before starting to build a good algorithm to solve Problem LS four theorems will be considered (proof is given in the book by Kinkaid and Cheney, 1996).

Theorem 1 *Let* $\mathbf{A}$ *be an* $m \times n$ *matrix whose rank* k *satisfy* $k < n \leq m$. *There is an* $m \times m$ *orthogonal matrix* $\mathbf{Q}$ *and an* $n \times n$ *permutation matrix* $\mathbf{P}$ *such that*

$$\mathbf{QAP} = \begin{bmatrix} \mathbf{R}_{k\times k} & \mathbf{T}_{k\times(n-k)} \\ \mathbf{0}_{(m-k)\times k} & \mathbf{0}_{(m-k)\times(n-k)} \end{bmatrix} \tag{A.3}$$

where $\mathbf{R}$ *is a* $k \times k$ *upper triangular matrix of rank* k.

Theorem 2 Let $[\mathbf{R}\colon \mathbf{T}]$ *be a* $k \times n$ *matrix where* $\mathbf{R}$ *is of rank* k. *There is an* $n \times n$ *orthogonal matrix* $\mathbf{W}$ *such that*

$$[\mathbf{R}\colon \mathbf{T}]\mathbf{W} = [\hat{\mathbf{R}}_{k\times k}\colon \mathbf{0}_{k\times(n-k)}] \tag{A.4}$$

where $\hat{\mathbf{R}}$ *is a lower triangular matrix of rank* k.

Theorem 3 Let $\mathbf{A}$ *be an* $m \times n$ *matrix of rank* k.
Then there is an $m \times m$ *orthogonal matrix* $\mathbf{H}$ *and an* $n \times n$ *orthogonal matrix* $\mathbf{K}$ *such that*

$$\mathbf{H}^T\mathbf{A}\mathbf{K} = \mathbf{R},\ \mathbf{A} = \mathbf{H}\mathbf{R}\mathbf{K}^T \tag{A.5}$$

where

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{A.6}$$

Here $\mathbf{R}_{11}$ *is a* $k \times k$ *nonsingular triangular matrix.*

Theorem 4 Suppose that $\mathbf{A}$ *is an* $m \times n$ *matrix of rank* k. $\mathbf{H}$, $\mathbf{R}$ *and* $\mathbf{K}$ *are such as in Theorem 3. Define the vector*

$$\mathbf{H}^T\mathbf{b} = \mathbf{g} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \end{bmatrix} \begin{matrix} \}k \\ \}m-k \end{matrix}$$

and introduce the new variable

$$\mathbf{K}^T\mathbf{x} = \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \begin{matrix} \}k \\ \}n-k \end{matrix}$$

Define $\tilde{\mathbf{y}}_1$ *to be the unique solution of*

$$\mathbf{R}_{11}\mathbf{y}_1 = \mathbf{g}_1$$

Then:

(1) All solutions to the problem of minimizing $\|\mathbf{Ax} - \mathbf{b}\|$ *are of the form*

$$\hat{\mathbf{x}} = \mathbf{K} \begin{bmatrix} \tilde{\mathbf{y}}_1 \\ \mathbf{y}_2 \end{bmatrix} \tag{A.7}$$

where $\mathbf{y}_2$ *is arbitrary;*

(2) Any such $\hat{\mathbf{x}}$ *gives rise to the same residual vector* $\mathbf{r}$ *satisfying*

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}} = \mathbf{H} \begin{bmatrix} \mathbf{0} \\ \mathbf{g}_2 \end{bmatrix}; \tag{A.8}$$

(3) The norm of $\mathbf{r}$ *satisfies*

$$\|\mathbf{r}\| = \|\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}\| = \|\mathbf{g}_2\|; \tag{A.9}$$

(4) The unique solution of minimum length is

$$\tilde{\mathbf{x}} = \mathbf{K} \begin{bmatrix} \tilde{\mathbf{y}}_1 \\ \mathbf{0} \end{bmatrix}. \tag{A.10}$$

A special case of orthogonal decomposition is called *QR decomposition*, represented as follows:

$$\mathbf{A} = \mathbf{Q}^T\mathbf{R} = \mathbf{Q}^T\mathbf{R}\mathbf{I}_n \tag{A.11}$$

This decomposition is allowed when $m \times n$ matrix has $Rank = n$. Householder transformations (Householder, 1958 and Lawson and Hanson, 1995) allow to build a matrix $\mathbf{Q}$ as in (A.11). Householder transformations are not the only suitable method to solve LS problems. For instance Renka (1988) used Givens transformations (Lawson and Hanson, 1995) for the Shepard's method explained in §7.4.3. The Householder method was chosen for our implementation for its major robustness (Lawson and Hanson, 1995); accuracy is similar and often depends on details of the codes. The matrix $\mathbf{Q}$ is computed as product of Householder transformations

$$\mathbf{Q} = \mathbf{Q}_n \cdots \mathbf{Q}_1 \tag{A.12}$$

where each $\mathbf{Q}_i$ has the form

$$\mathbf{Q}_i = \mathbf{I}_m + b_i^{-1}\mathbf{u}^{(i)}\mathbf{u}^{(i)T} \tag{A.13}$$

where $\mathbf{u}^i$ is an m-vector satisfying

$$\|\mathbf{u}^i\| \neq \mathbf{0}\| \quad \text{and} \quad b_i = -\frac{\|\mathbf{u}^i\|^2}{2} . \tag{A.14}$$

In the algorithm that was implemented for this work, for a given vector $\mathbf{v}^i$ the vector $\mathbf{u}^i$ is determined so that the following relation is satisfied (where we set $\mathbf{v}^i \equiv \mathbf{v}$ and $\mathbf{u}^i \equiv \mathbf{u}$):

$$\mathbf{Q}_i\mathbf{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_{p-1} \\ s \\ v_{p+1} \\ \vdots \\ v_{l-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \equiv \mathbf{u} \tag{A.15}$$

where

$$s = -\sigma\left(v_p^2 + \sum_{i=l}^{m} v_i^2\right)^{1/2}$$

with

$$\sigma = \begin{cases} +1 \text{ if } v_p \geq 0 \\ -1 \text{ if } v_p < 0 \end{cases}$$

The effect of the matrix $\mathbf{Q}$ in transforming $\mathbf{v}$ to $\mathbf{u}$ can be described by means of three non-negative integer parameters p, l and m as follows:

1. If $p > 1$, components 1 through $p - 1$ are to be unchanged;

2. Component p is permitted to change. This is called the pivot element;

3. If $p < l-1$, components $p+1$ through $l-1$ are to be left unchanged;

4. If $l < m$, components l through m are to be zeroed.

It is possible to demonstrate (Lawson and Hanson, 1995) that the computational steps necessary to produce the $m \times m$ orthogonal matrix $\mathbf{Q}$ (satisfying conditions imposed by integer parameters p, l and m) are:

$$
\begin{aligned}
u_i &= 0 && i = 1, \ldots, l-1 \\
u_p &= v_p - s \\
u_i &= 0 && i = p+1, \ldots, l-1 \\
u_i &= v_i && i = l, \ldots, m \\
b &= s u_p \\
\mathbf{Q} &= \begin{cases} \mathbf{I}_m + b^{-1}\mathbf{u}\mathbf{u}^T & \text{if } b \neq 0 \\ \mathbf{I}_m & \text{if } b = 0 \end{cases}
\end{aligned}
$$

These transformations have been implemented in an algorithm called ORTHO $(m, n, \mathbf{A}, \mathbf{h})$. This algorithm builds matrix $\mathbf{Q}$ as in (A.12). Inputs are integers m, n and the $m \times n$ matrix $\mathbf{A}$. The output consists of the nonzero portion of the upper triangular matrix $\mathbf{R}$ stored in the upper triangular portion of the matrix $\mathbf{A}$, the scalars $u_i^{(i)}$ stored in the ith row h_i of the array named $\mathbf{h}$, and the remaining nonzero portions of the vectors $\mathbf{u}_i$ stored as columns in the subdiagonal part of the ith column of matrix $\mathbf{A}$. Index p, l m of expression (A.15) are set as $p = i$, $l = i+1$ and $m = m$, for $i = 1, \ldots, n$.

If, as it was supposed, Rank($\mathbf{A}$)= n, then the $n \times n$ matrix $\mathbf{R}_{11}$ (A.6) is non singular and the solution $\hat{\mathbf{x}}$ can be obtained (as stated in Theorem 3) by computing

$$\mathbf{g} = \mathbf{Q}\mathbf{b}$$

partioning g as

$$\mathbf{g} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \end{bmatrix} \begin{matrix} \}n \\ \}m-n \end{matrix}$$

and solving

$$\mathbf{R}_{11}\mathbf{x} = \mathbf{g}_1$$

for the solution vector $\hat{\mathbf{x}}$.

The outputs from algorithm ORTHO, m and n, and the array $\mathbf{b}$ (holding the m-vector $\mathbf{b}$ of LS problem) are utilised as inputs for the algorithm SOLVER $(m, n, \mathbf{A}, \mathbf{h}, \mathbf{b})$, which produces the n-vector $\hat{\mathbf{x}}$, solution of the LS problem, in the first n entries of array $\mathbf{b}$.

The norm ρ of the residual vector $\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}}$ can be computed as

$$\mathbf{r} = \mathbf{Q}_1 \cdots \mathbf{Q}_n \begin{bmatrix} \mathbf{0} \\ \mathbf{g}_2 \end{bmatrix} \tag{A.16}$$

and

$$\rho \equiv \|\mathbf{r}\| = \|\mathbf{g}_2\| \tag{A.17}$$

The residual norm ρ is used to judge the quality of the LS estimation. If the ρ value is too high the estimation is probably unreliable. Parameter estimation is accepted only if $\rho < 10^{-2}$. Otherwise, rigorous CFD calculations are called.

Algorithm SOLVER is based on the assumption that the matrix $\mathbf{A}$ is of rank n. There many cases, however, when this condition is not fulfilled. Nonetheless, it is possible to build an algorithm able to successfully treat this issue. If one matrix is so close to a rank deficient matrix that changes of the order of magnitude of the data uncertainty could convert the matrix to a deficient one, then the problem is ill-conditioned and one algorithm able to tackle the situation should be found. One technique to stabilise the problem is to replace $\mathbf{A}$ with a similar rank-deficient matrix $\tilde{\mathbf{A}}$ and solve the LS problem for $\tilde{\mathbf{A}}\mathbf{x} \cong \mathbf{b}$. The procedure is implemented through the following steps (see Theorems 1 and 2):

a)

$$\mathbf{QAP} = \mathbf{R} \equiv \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix}$$

with $\mathbf{R}_{11}$, $\mathbf{R}_{12}$ and $\mathbf{R}_{22}$, $k \times k$, $k \times (n-k)$ and $(m-k) \times (n-k)$ matrices, respectively.

b)

$$\mathbf{Qb} = \mathbf{c} \equiv \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} \begin{matrix} \}k \\ \}m-k \end{matrix}$$

c)

$$[\mathbf{R}_{11} : \mathbf{R}_{12}]\mathbf{K} = [\mathbf{W} : \mathbf{0}]$$

d)

$$\mathbf{W}\mathbf{y}_1 = \mathbf{c}_1$$

e)

$$\mathbf{y}_2 \text{ arbitrary}$$

As stated in Theorem 4, the minimal length solution of LS problem is given by $\mathbf{y}_2 = \mathbf{0}$ and, thus, the value zero is given to the $(n-k)$-vector $\mathbf{y}_2$.

f)

$$\mathbf{x} = \mathbf{P}\mathbf{K}\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \equiv \mathbf{P}\mathbf{K}\mathbf{y}$$

g)

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}\| = \|\mathbf{c}_2 - \mathbf{R}_{22}\mathbf{y}_2\| \; (= \|\mathbf{c}_2\| \text{ if } \mathbf{y}_2 = \mathbf{0})$$

If matrix $\mathbf{A}$ is singular, matrix $\mathbf{R}_{22}$ is $\mathbf{0}$. When $\mathbf{A}$ is ill conditioned, the objective is to determine k in order to have $\mathbf{R}_{11}$ well-conditioned and $\|\mathbf{R}_{22}\|$ small. As suggested in the book by Lawson and Hanson (1995), the rank k is determined as the largest index i such that $|r_{ii}| > \tau$, where τ is a non-negative absolute tolerance parameter. The parameter τ depend on errors on $\mathbf{A}$ and $\mathbf{b}$, on the computational algorithm and on the machine, which is utilized. Setting $\tau = 0.005$ appeared to give good results. Algorithms SOLVER and ORTHO are used to solve the $k \times k$ matrix $\mathbf{R}_{11}$.

References

[1] A.G. Abdul Ghani, M.M. Farid, X.D. Chen, and P. Richards. An investigation of deactivation bacteria in a canned liquid food during sterilization using computational fluid dynamics (CFD). *J. Food Engng.*, 42:207–214, 1999.

[2] M.H. Al-Rashed and A.G. Jones. CFD modelling of gas liquid reactive precipitation. *Chem. Engng. Sci.*, 54:4779–4784, 1999.

[3] K.J. Åström, U. Borisson, L. Ljung, and B. Wittenmark. Theory and applications of self tuning regulators. *Automatica*, 13:457–476, 1977.

[4] B. Atkinson and F. Mavituna. *Biochemical engineering and biotechnology handbook.* Macmillan Publishers, Houndmills, UK, 1991.

[5] R. Bachmann, L. Brüll, T. Mrzigold, and U. Pallaske. On methods for reducing the index of differential algebraic equations. *Comput. Chem. Engng.*, 14:1271–1273, 1990.

[6] K.J. Badcock, B.E. Richards, and M.A. Woodgate. Elements of computational fluid dynamics on block structured grids using implicit solvers. *Progr. Aero. Sci.*, 36:351–392, 2000.

[7] A.C. Badino Jr., M.C.R. Facciotti, and W. Schmidell. Volumetric oxygen transfer coefficients (k_La) in batch cultivations involving non-Newtonian broths. *Biochem. Engng. J.*, 8:111–119, 2001.

[8] J.E. Bailey and D.F. Ollis. *Biochemical engineering fundamentals.* McGraw-Hill Book Co., Singapore, 1986.

[9] J. Baldyga and J.R. Bourne. Interaction between mixing on various scales in a stirred tank reactor. *Chem. Engng. Sci.*, 47:1837–1848, 1992.

[10] J. Baldyga, J.R. Bourne, and S.J. Hearn. Interaction between chemical reactions and mixing on various scales. *Chem. Engng. Sci.*, 52(4):457–466, 1997.

[11] J. Baldyga and W. Orciuch. Barium sulphate precipitation in a pipe – an experimental study and CFD modelling. *Chem. Engng. Sci.*, 56:2435–2444, 2001.

[12] A. Barrett and J.J. Walsh. Improved chemical process simulation using local thermodynamics approximations. *Comput. Chem. Engng.*, 3:397–402, 1979.

[13] P.I. Barton and C.C.Pantelides. Modeling of combined discrete/continuous processes. *AIChE J.*, 40:966–979, 1994.

[14] M. Bauer and G. Eigenberger. A concept for multi-scale modeling of bubble columns and loop reactors. *Chem. Engng. Sci.*, 54:5109–5117, 1999.

[15] M. Bauer and G. Eigenberger. Multiscale modeling of hydrodynamics, mass transfer and reaction in bubble column reactors. *Chem. Engng. Sci.*, 56:1067–1074, 2001.

[16] S.K. Bermingham, A.M. Neumann, H.J.M. Kramer, P.J.T. Verheijen, G.M. van Rosmalen, and J. Grievink. A design procedure and predictive models for solution crystallisation processes. In *Proc. 5th Conf. Foundations of Computer-Aided Process Design*, volume 96 of *AIChE Symposium Series No. 323*, pages 250–264. M.F. Malone, J.A. Trainham and B. Carnaham (eds.), CACHE-AIChE Publications, 2000.

[17] L.T. Biegler, I.E. Grossmann, and A.W. Westerberg. *Systematic methods of chemical process design.* Prentice Hall PTR, Upper Saddle River, NJ, 1997.

[18] G.J. Bierman. *Factorization methods for discrete sequential estimation.* Academic Press, Inc., London, UK, 1972.

[19] A. Birtigh, G. Lauschke, W.F. Schierholz, D. Beck, C. Maul, N. Gilbert, H.G. Wagner, and C.Y. Werninger. CFD in chemical process engineering from an industrial perspective. *Chemie Ingenieur Technick*, 72:175–193, 2000.

[20] R. Bogusch and W. Marquardt. A formal representation of process model equations. *Comp. Chem. Engng.*, 19:S211–S216, 1995.

[21] B.L. Braunschweig, C.C. Pantelides, H.I. Britt, and S. Sama. Process modeling: the promise of open software architectures. *Chem. Engng. Progr.*, 96:65–76, 2000.

[22] A. Brucato, M. Ciofalo, F. Grisafi, and R. Tocco. On the Simulation of stirred tank reactors via computational fluid dynamics. *Chem. Engng. Sci.*, 3:397–402, 1999.

[23] M.L. Bush, C.B. Frederick, J.S. Kimbell, and J.S. Ultman. A CFD-PBPK hybrid model for simulating gas and vapor uptake in the rat nose. *Toxic. and Appl. Pharm.*, 150:133–145, 1998.

[24] F. Cacik and R.G. Dondo amd D. Marques. Optimal control of a batch bioreactor for the production of xanthan gum. *Comput. Chem. Engng.*, 25:409–418, 2001.

[25] L.S. Caretto, A.D. Gosman, S.V. Patankar, and D.B. Spalding. Two calculation procedures for steady, three-dimensional flows with recirculation. In *Proc. Third Int. Conf. Numer. Meth. Fluid Dyn.*, Paris, France, 1972.

[26] R.E. Carlson and T.A. Foley. The parameter R^2 in multiquadric interpolation. *Comp. Math. Applic.*, 55(2):291–300, 1991.

[27] B. Carré. *Graphs and networks.* Clarendon Press, Oxford, UK, 1979.

[28] D.R. Chapman. Computational aereodynamics development and outlook. *AIAA J.*, 17(12):1293–1313, 1979.

[29] Computer simulation saves $15,000 by solving difficult mixing problem, 2001. in www.chemicalonline.com.

[30] E.H. Chimowitz, T.F. Anderson, S. Macchietto, and L.F. Stutzman. Local models for representing phase equilibria in multicomponent nonideal vapor-liquid and liquid-liquid systems. 1. Thermodynamics approximation functions. *Ind. Eng. Chem. Process Des. Dev.*, 25:674–682, 1986.

[31] R.A. DeVore. Nonlinear approximation. *Acta Numerica*, pages 51–150, 1998.

[32] J.M. Eakman, A.G. Fredrickson, and H.M. Tsuchiya. Statistics and dynamics of microbial cell populations. *Chem. Engng. Progr.*, 62(69):37–49, 1966.

[33] A.M Eaton, L.D. Smoot, S.C. Hill, and C.N. Eatough. Components, formulations, solutions, evaluation, and application of comprehensive combustion models. *Prog. Energy Combust. Sci.*, 25:387–436, 1999.

[34] S.O. Enfors, M. Jahic, A. Rozkov, B. Xu, M. Hecker, B. Jürgen, T. Krüger, T. Schweder, G. Hamer, D.O'Beirne, N. Noisommit-Rizzi, M. Reuss, L. Boone, C. Hewitt, C. McFarlane, A. Nienow, T. Kovacs, C. Trägårdh, L. Fuchs, J. Revstedt, P.C. Friberg, B. Hjerteger, G. Blomsten, H. Skogman, S. Hjort, F. Hoeks, H.Y. Lin, P. Neubauer, R. van der Lans, K. Luyben, P. Vrabel, and Å. Manelius. Physiological responses to mixing in large scale bioreactors. *J. Biotech.*, 85:175–185, 2001.

[35] W.F. Feehery, J.E. Tolsma, and P.I. Barton. Efficient sensitivity analysis of large-scale differential algebraic equations. *Appl. Numer. Math.*, 25(1):41–54, 1997.

[36] J.H. Ferziger and M. Peric. *Computational methods for fluid dynamics.* Springer, Berlin, Germany, 1999.

[37] C.A.J. Fletcher. *Computational techniques for fluid dynamics. Volume 1. Fundamental and general techniques.* Springer-Verlag, Berlin Heidelberg, Germany, 1991.

[38] C.A.J. Fletcher. *Computational techniques for fluid dynamics. Volume 2. Specific techniques for different flow categories.* Springer-Verlag, Berlin Heidelberg, Germany, 1991.

[39] P. Fletcher. Heat transfer coefficient for stirred batch reactor design. *The Chemical Engineer*, pages 33–37, 1987.

[40] Fluent, Inc., Lebanon, NH, USA. *FLUENT 4.4 User's Guide*, 1997.

[41] B.A. Foss, B. Lohmann, and W.Marquardt. A field study of the industrial modeling process. *J. Proc. Contr.*, 8:325–338, 1998.

[42] R. Franke and G. Nielson. Smooth interpolation of large set of scattered data. *Int. J. Numer. Methods Engng.*, 15:181–200, 1980.

[43] F. García-Ochoa and E. Gómez. Mass transfer coefficient in stirred tank reactors for xanthan gum solutions. *Biochem. Engng. J.*, 1:1–10, 1998.

[44] F. García-Ochoa, E. Gomez-Castro, and V.E. Santos. Oxygen transfer and uptake rates during xanthan gum production. *Enzyme Microb. Tech.*, 27:680–690, 2000.

[45] F. García-Ochoa, V.E. Santos, and A. Alcón. Xanthan gum production: an unstructured kinetic model. *Enzyme Microb. Tech.*, 17:206–217, 1995.

[46] F. García-Ochoa, V.E. Santos, and A. Alcón. Metabolic structured model for xanthan production. *Enzyme Microb. Tech.*, 23:75–82, 1998.

[47] C.W. Gear. Differential algebraic equations, indeces and integral algebraic equations. *SIAM J. Numer. Anal.*, 27:1527–1534, 1990.

[48] C.W. Gear and L.R. Petzold. ODE methods for the solution of differential algebraic systems. *SIAM J. Numer. Anal.*, 21:716–728, 1984.

[49] F.B. Godin, D.G. Cooper, and A.D. Rey. Development and solution of a cell mass population balance model applied to the SCF process. *Chem. Engng. Sci.*, 54:565–578, 1999.

[50] S. Gordon and P. Richardson. The application of fluid dynamics in the food industry. *Trends Food Sci. Tech.*, 8:119–124, 1997.

[51] W.J. Gordon and J. Wixon. On Shepard's method of metric interpolation to bivariate and multivariate data. *Math. Comp.*, 32:253–264, 1978.

[52] A.D. Gosman. Developments in industrial computational fluid dynamics. *Chem. Engng. Res. Des.*, 76:153–161, 1998.

[53] A.D. Gosman. Developments in CFD for industrial and environmental applications in wind engineering. *J. Wind Engng. Ind. Aerodyn.*, 81:21–39, 1999.

[54] A.D. Gosman. State of the art of multi-dimensional modeling of engine reacting flows. *Oil & Gas Sci. Tech.*, 54:149–159, 1999.

[55] N. Hamill and G. Baché. CFD applications in the process industry. In *Computational Technologies for Fluid/Thermal/Strctural/Chemical Systems with Industrial Applications*, No. 377-2, pages 1–9. ASME, 1998.

[56] R.L. Hardy. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. Res.*, 76:1905–1915, 1971.

[57] C.J. Hewitt, G. Nebe-Von Caron, B. Axelsson, C.M. McFarlane, and A.W. Nienow. Studies related to the scale-up of high-cell-density E. coli fed-batch fermentations using multiparameter flow cytometry: effect of a changing microenvironment with respect to glucose and dissolved oxygen concentration. *Biotech. Bioengng.*, 70(4):381–390, 2000.

[58] M. Hillestad, C. Sørlie, T.F. Anderson, I. Olsen, and T. Hertzberg. On estimating the error of local thermodynamics models. A general approach. *Comput. Chem. Engng.*, 13:789–796, 1989.

[59] A.S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. ACM*, 5:339–342, 1958.

[60] Y. Iiguni, I. Kawamoto, and N. Adachi. A nonlinear adaptive estimation method based on local approximation. *IEEE Trans. Signal Proc.*, 45(7):1831–1841, 1997.

[61] D. Ilievski, M. Rudman, and G. Metcalfe. The separate roles of shear rate and mixing on gibbsite precipitation. *Chem. Engng. Sci.*, 56:2521–2530, 2001.

[62] Fluent Inc. Partnerships drive expanded product capabilities. *FluentNews*, 9(2):16–17, 2000/2001.

[63] R.B. Joglegar and G.V. Reklaitis. A simulator for batch and semi-continuous processes. *Comp. Chem. Engng.*, 8:315–327, 1984.

[64] T. Jongen. Characterization of batch mixers using numerical flow simulations. *AIChE J.*, 46(11):2140–2150, 2000.

[65] A.I. Kakhu, B.R. Keeping, Y. Lu, and C.C. Pantelides. An open software architecture for process modelling and model-based applications. In *Proc. 3rd Conf. Foundations of Computer-Aided Process Operations*, volume 94 of *AIChE Symposium Series No. 320*, pages 518–524. J.Pekny and G.Blau (eds.), Cache Publications, 1998.

[66] R. Keunings. A survey of computational rheology. In *XIIIth International Congress on Rheology*, 2000.

[67] S.E. Kim and F. Boysan. Application of CFD to environmental flows. *J. Wind. Engng. Ind. Aerodyn.*, 81:145–158, 1999.

[68] D. Kinkaid and W. Cheney. *Numerical analysis.* Brooks/Cole Publishing Company, Philadelphia, USA, 1996.

[69] N.H. Kolhapure and R.O. Fox. CFD analysis of micromixing effects on polymerization in tubular low-density polyethylene reactors. *Chem. Engng. Sci.*, 54:3233–3242, 1999.

[70] H.J.M. Kramer, S.K. Bermingham, and G.M. van Rosmalen. Design of industrial crystallisers for a given product quality. *J. Crystal Growth*, 199:729–737, 1999.

[71] S. Kresta. Turbulence in stirred tanks: anisotropic, approximate and applied. *Can. J. Chem. Engng.*, 76:563–576, 1998.

[72] R. Krishna, J.M. van Baten, and J. Ellenberger. Scale effects in fluidized multiphase reactors. *Powder Technology*, 100:137–146, 1998.

[73] B.E. Launder and D.B. Spalding. *Lectures in mathematical models of turbulence.* Accademic Press, Pacific Grove, CA, USA, 1974.

[74] C.L. Lawson and R.J. Hanson. *Solving least square problems.* SIAM, Philadelphia, USA, 1995.

[75] T. Ledent and G. Heyen. Dynamic approximation of thermodynamic properties by means of local models. *Comp. Chem. Engng.*, 18S:S87–S91, 1994.

[76] M.E. Leesley and G. Heyen. The dynamic approximation method of handling vapor-liquid equilibrium data in computer calculations for chemical processes. *Comp. Chem. Engng.*, 1:109–112, 1977.

[77] G.Q. Li, H.W. Qiu, Z.M. Zheng, Z.L. Cai, and S.Z. Yang. Effect of fluid rheological properties on mass transfer in a bioreactor. *J. Chem. Tech. Biotech.*, 62:385–391, 1995.

[78] Y. Li, J. Zhang, and L.S. Fan. Numerical simulation of gas-liquid-solid fluidization systems using a combined CFD-VOF-DPM method: bubble wake behaviour. *Chem. Engng. Sci.*, 54:5101–5107, 1999.

[79] M. Liakopoulou-Kyriakides, E.S. Tzanakakis, C. Kiparissidis, L.V. Ekateriniadou, and D.A. Kyriakidis. Kinetics of xanthan gum production from whey by constructed strains of *Xanthomonas campestris* in batch fermentations. *Chem. Engng. Tech..*, 20:354–360, 1997.

[80] Z.H. Liu. *An advanced process manufacturing system-design and application to a food processing pilot plant.* PhD thesis, University of London, UK, 1995.

[81] Z.H. Liu and S. Macchietto. Model based control of a multipurpose batch reactor. *Comp. Chem. Engng.*, 19S:S477–S488, 1995.

[82] S. Macchietto, E.H. Chimowitz, T.F. Anderson, and L.F. Stutzman. Local models for representing phase equilibria in multicomponent nonideal vapor-liquid and liquid-liquid systems. 3. Parameter estimation and update. *Ind. Eng. Chem. Process Des. Dev.*, 25:674–682, 1986.

[83] E. Macquart-Moulin. Batch reactor safety verification. Internal report (unpublished), Centre for Process Systems Engineering, University of London, UK, 1998.

[84] D. Maggioris, A. Goulas, A.H. Alexopoulos, E.G. Chatzi, and C. Kiparissides. Use of CFD in prediction of particle size distribution in suspension polymer reactors. *Comput. Chem. Engng.*, 22S:S315–S322, 1998.

[85] D. Maggioris, A. Goulas, A.H. Alexopoulos, E.G. Chatzi, and C. Kiparissides. Prediction of particle size distribution in suspension polymerization reactors: effect of turbulence nonhomogeneity. *Chem. Engng. Sci.*, 55:4611–4627, 2000.

[86] R. Mann and A.M. El-Hamouz. A Product distribution paradox on scaling up a stirred batch reactor. *AIChE J.*, 41(4):855–867, 1995.

[87] R. Mann and P. Knysh. Utility of a network of interconnected backmixed zones to represent mixing in a closed stirred vessel. In *Fluid Mixing II*, pages 127–145. I.Chem.E. Symp. Series 89, 1984.

[88] R. Mann, P. Knysh, E.A. Rasekoala, and M. Didari. Mixing in a closed stirred vessel: use of networks of zones to interpret mixing in a closed stirred vessel. In *Fluid Mixing III*, pages 49–60. I.Chem.E. Symp. Series 108, 1987.

[89] R. Mann and P. Mavros. Analysis of unsteady tracer dispersion and mixing in a stirred vessel using interconnected networks of ideal flow zones. In *Papers Presented at the 4th European Conference on Mixing, Noordwijkerhout, Netherlands, April 1982*, pages 35–47. BHRA Fluid Engineering, Cranfield, Bedford, UK, 1982.

[90] N.V. Mantzaris, J.J. Liou, P. Daoutidis, and F. Srienc. Numerical solution of a mass structured cell population balance model in an environment of changing substrate concentration. *J. Biotech.*, 71:157–174, 1999.

[91] F. Marias. Coupling of gPROMS and Fluent: application to a rotary kiln incinerator. Internal report (unpublished). Centre for Process Systems Engineering, University of London, UK, 2000.

[92] D. Maroudas. Multiscale modeling of hard materials: challenges and opportunities for chemical engineering. *AIChE J.*, 46(5):878–882, 2000.

[93] W Marquardt. An object-oriented representation of structured process models. *Comput. Chem. Engng.*, 16S:S329–S336, 1992.

[94] W Marquardt. Trends in computer-aided process modeling. *Comput. Chem. Engng.*, 20(6/7):591–609, 1996.

[95] S.E. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM J. Sci. Comput.*, 14(3):677–692, 1993.

[96] A.B. Metzner and R.E. Otto. Agitation of non-Newtonian fluids. *AIChE J.*, 1:3–10, 1957.

[97] E. Nagy, M. Neubeck, B. Mayr, and A. Moser. Simulation of the effect of mixing, scale-up and pH-value regulation during glutamic acid fermentation. *Bioproc. Engng.*, 12:231–238, 1995.

[98] J. Neumann. Mathematical modelling of distributed systems. MPhil to PhD transfer report. Centre for Process Systems Engineering, University of London, UK, 2001.

[99] K. Ng and M. Yianneskis. Observations on the distribution of energy dissipation in stirred vessels. *Chem. Engng. Res.*, 78(A3):334–341, 2000.

[100] K.A. Nguyen, I. Rossi, and D.G. Truhlar. A dual-level Shepard interpolation method for generating potential energy surfaces for dynamic calculations. *J. Chem. Phys.*, 103(13):5522–5530, 1995.

[101] A.W. Nienow. Mixing: studies at the University of Birmingham on this traditional technology critical in the manufacture of new biological products. *Food Bio. Proc.*, 78-C:145–151, 2000.

[102] M. Oh and C.C. Pantelides. A modelling and simulation language for combined lumped and distributed parameter systems. *Comput. Chem. Engng.*, 20:611–633, 1996.

[103] C.C. Pain, S. Mansoorzadeh, and C.R.E. de Oliveira. A Study of bubbling and slugging fluidised beds using the two-fluid granular temperature model. *Int. J. Multiph. Flow*, 27:527–551, 2001.

[104] C.C. Pain, A.P. Umpleby, C.R.E. de Oliveira, and A.J.H. Goddard. Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations. *Comput. Meth. Appl. Mech. Engng.*, 190:3771–3796, 2001.

[105] C.C. Pantelides. SpeedUp: recent advances in process simulation. *Comput. Chem. Engng.*, 12(7):745–755, 1988.

[106] C.C. Pantelides. The consistent initialisation of differential-algebraic systems. *J.Sci.Stat.Comput.*, 9(2):213–231, 1988.

[107] C.C. Pantelides. Advances in process simulation: multiscale process modelling. In *Bayer Conference, Leverkusen, Germany*, 2000.

[108] C.C. Pantelides and P.I. Barton. Equation-oriented dynamic simulation: current status and future perspectives. *Comput. Chem. Engng.*, 17S:S263–S285, 1993.

[109] C.C. Pantelides and H.I. Britt. Multipurpose process modeling environments. In *Proc. of the Conf. on Foundations of Computer-Aided Process Design '94*, pages 128–141. L.T. Biegler and M.F. Doherty (eds.), CACHE Publications, Austin, TX, 1995.

[110] E.T. Papoutsakis. Fluid-mechanical damage of animal cells in bioreactors. *Trends Biotech.*, 9:427–437, 1991.

[111] J.D. Perkins and R.W.H. Sargent. SPEEDUP: a computer program for steady-state and dynamic simulation and design of chemical processes. In *Selected Topics on Computer-Aided Process Design and Analysis*, volume 78 of *AIChE Symposium Series No. 214*, pages 1–11. AIChE Publications, 1982.

[112] J. Perregaard. Model simplification and reduction for simulation and optimization of chemical processes. *Comput. Chem. Engng.*, 17:465–483, 1992.

[113] H.U. Peters, H. Herbst, P.G.M. Hesselink, H. Lünsdorf, A. Schumpe, and W.D. Deckwer. The influence of agitation rate on xanthan production by *Xanthomonas campestris*. *Biotech. Bioengng.*, 34:1393–1397, 1989.

[114] P.C. Piela, T.G. Epperly, K.M. Westerberg, and A.W. Westerberg. ASCEND: an object-oriented environment for modeling and analysis: the modeling language. *Comput. Chem. Engng.*, 15:53–72, 1991.

[115] A. Pons, C.G. Dussap, and J.B. Gros. Modelling *Xanthomonas campestris* batch fermentations in bubble columns. *Biotech. Bioengng.*, 33:394–405, 1989.

[116] M.J.D. Powell. The theory of radial basis function approximation in 1990. In *Advances in numerical analysis. Vol.III. Wavelets, subdivision algorithms, and radial basis functions. Edited by Will Light*, pages 105–210. Oxford University Press, Oxford, UK, 1992.

[117] Process Systems Enterprise Ltd., London,UK. *gPROMS Introductory User's Guide*, 1997.

[118] Process Systems Enterprise Ltd., London,UK. *gPROMS Advanced User's Guide*, 1999.

[119] R.J. Renka. Multivariate interpolation of large sets of scattered data. *ACM Trans. Mathem. Software*, 14(2):139–148, 1988.

[120] M. Reuss, D. Debus, and G. Zull. Rheological properties of fermentation fluids. *Chem. Eng.*, pages 233–236, 1982.

[121] J. Revstedt, L. Fuchs, T. Kovacs, and C. Trägårdh. Influence of impeller type on the flow structure in a stirred reactor. *AIChE J.*, 46(12):2373–2382, 2000.

[122] S. Saelid, O. Egeland, and B. Foss. A solution to the blow-up problem in adaptive controllers. *Model. Ident. Contr.*, 6(1):39–56, 1985.

[123] K.D. Samant and K.M. Ng. Development of liquid-phase agitated reactors: synthesis, simulation and scaleup. *AIChE J.*, 45(11):2371–2391, 1999.

[124] R.W.H. Sargent and A.W. Westerberg. "SPEED-UP" in chemical engineering design. *Trans. Instn. Chem. Engrs.*, 42:T190–T197, 1964.

[125] W.E. Schiesser. *The numerical methods of lines – Integration of partial differential equations.* Academic Press, San Diego, CA, USA, 1991.

[126] L. Serrano-Carreón, R.M. Corona, A. Sánchez, and E. Galindo. Prediction of xanthan fermentation development by a model linking kinetics, power drawn and mixing. *Proc. Biochem.*, 33(2):133–146, 1998.

[127] M. Shacham, S. Macchietto, L.F. Stutzman, and P. Babcock. Equation oriented approach to process flowsheeting. *Comp. Chem. Engng.*, 6(2):79–95, 1982.

[128] J.J. Shah and R.O. Fox. Computational fluid dynamics simulation of chemical reactors: application of in situ adaptive tabulation to methane thermochlorination chemistry. *Ind. Engng. Chem. Res.*, 38:4200–4212, 1999.

[129] A. Shanley. CFD for the real world. *Chem. Engng.*, 107(11):139–143, 2000.

[130] R. Shantanu, M.P. Dudukovic, and P.L. Mills. A two-phase compartments model for the selective oxidation of *n*-butane in a circulating fluidized bed reactor. *Catalysis Today*, 61:73–85, 2000.

[131] C.Y. Shen, H.L. Reed, and T.A. Foley. Shepard's interpolation for solution-adaptive methods. *J. Comput. Phys.*, 106:52–61, 1993.

[132] D. Shepard. A two dimensional interpolation function for irregularity spaced data. In *Proc. 23rd Nat. Conf. ACM*, pages 517–523, 1968.

[133] M.S. Sheppard. Approaches to the automatic generation and control of finite element meshes. *Appl. Mech. Rev.*, 41:169–184, 1988.

[134] J.N. Sherwood and R.I. Ristic. The influence of mechanical stress on the growth and dissolution of crystals. *Chem. Engng. Sci.*, 56:2267–2280, 2001.

[135] K. Shimizu, S. Takada, K. Minekawa, and Y. Kawase. Phenomenological model for bubble column reactors: prediction of gas hold-ups and volumetric mass transfer coefficients. *Chem. Engng. J.*, 78:21–28, 2000.

[136] E.M.B. Smith and C.C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Comp. Chem. Engng.*, pages 457–478, 1999.

[137] B.K. Soni. Grid generation: past, present and future. *Appl. Numer. Math.*, 32:361–369, 2000.

[138] M. Soroush. State and parameter estimations and their applications in process control. *Comput. Chem. Engng.*, 23:229–245, 1998.

[139] G. Stephanopoulos, G. Henning, and H. Leone. MODEL.LA. A modeling language for process engineering – I. The formal framework. *Comput. Chem. Engng.*, 14:813–846, 1990.

[140] S. Støren and T. Hertzberg. Local thermodynamics models used in sensitivity estimation of dynamic systems. *Comput. Chem. Engng.*, 21:S709–S714, 1997.

[141] G. Subramanian and D. Ramkrishna. On the solution of statistical models of cell populations. *Math. Biosci.*, 10:1–23, 1971.

[142] S. Sundaresan. Modeling the hydrodynamics of multiphase flow reactors: current status and challenges. *AIChE J.*, 46(6):1102–1105, 2000.

[143] I.D. Symeonidis. Dynamic modelling and simulation of crystallisation processes. Master's thesis, University of London, UK, 1997.

[144] R. Tarjan. Depth-first search and linear graphs algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[145] N. Thomson and D.F. Ollis. Extracellular microbial polysaccharides. II. Evolution of broth power-law parameters for xanthan and pullulan batch fermentation. *Biotech. Bioengng.*, 22:875–883, 1980.

[146] J.E. Tolsma and P.I. Barton. DAEPACK: an open modeling environment for legacy models. *Ind. Engng. Chem. Res.*, 39:1826–1839, 2000.

[147] M.F. Tomé, S. McKee, L. Barratt, D.A. Jarvis, and A.J. Patrick. An experimental and numerical investigation of container filling with viscous liquids. *Int. J. Num. Meth. Fluids*, 31:1333–1353, 1999.

[148] M. Torbacke and Å.C. Rasmuson. Influence of different scales of mixing in reaction crystallization. *Chem. Engng. Sci.*, 56:2459–2473, 2001.

[149] D.R. Unger, F.J. Muzzio, J.G. Aunins, and R. Singhvi. Computational and experimental investigation of flow and fluid mixing in the roller bottle bioreactor. *Biotech. Bioengng.*, 70(2):117–130, 2000.

[150] J. Unger, A. Kröner, and W.Marquardt. Structural analysis of differential algebraic equation systems - Theory and application. *Comp. Chem. Engng.*, 19(8):867–882, 1995.

[151] Z. Urban, T. Ishikawa, and Y. Natori. Modelling of a multitubular catalytic reactor using a CFX-gPROMS hybrid approach. In *DERC Mini-Symposium, Santa Clara, CA, USA*, 1997.

[152] Z. Urban and L. Liberis. Hybrid gPROMS-CFD modelling of an industrial scale crystalliser with rigorous crystal nucleation and growth kinetics and a full population balance. *Proc. Chemputers 1999 Conference, Dusseldorf, Germany*, 1999.

[153] J.P. van Doormal and G.D. Raithby. Enhancements of the SIMPLE method for predicting incompressible fluid flows. *Numer. Heat Transfer*, 7:147–163, 1984.

[154] J Villermaux. Future challenges in chemical engineering research. *Chem. Engng. Res. Des.*, 73:105–109, 1995.

[155] E. Vivaldo Lima, P.E. Wood, A.E. Hamielec, and A. Penlidis. Calculations of the particle size distribution in suspension polymerization using a compartment-mixing model. *Canadian J. of Chem. Engng.*, 76:495–505, 1998.

[156] D. Vlaev, R. Mann, V. Lossev, S.V. Vlaev, J. Zahradnik, and P. Seichter. Macromixing and *Streptomyces Fradiae*. Modelling oxygen and nutrient segregation in an industrial bioreactor. *Chem. Engng. & Res.*, 78-A3:354–362, 2000.

[157] S.D. Vlaev, R. Mann, and V. Lossev. An analysis of the effect of rheology on local gas hold-up: the case of Thylosin production. *Canadian J. of Chem. Engng.*, 76:495–505, 1995.

[158] P. Vrabel, R.G.J.M. van der Lans, K.C.A.M. Luyben, L. Boon, and A.W. Nienow. Mixing in large-scale vessels stirred with multiple radial or radial and axial up-pumping impellers: modelling and measurements. *Chem. Engng. Sci.*, 55:5881–5896, 2000.

[159] H. Wei and J. Garside. Application of CFD modelling to precipitation systems. *Chem. Engng. Res. Des.*, 75(A2):219–227, 1997.

[160] D.B. West. *Introduction to graph theory.* Prentice Hall, Upper Saddle River, NJ, USA, 1996.

[161] A.W. Westerberg, H.P. Hutchinson, R.L. Motard, and P. Winter. *Process Flow-sheeting.* Cambridge University Press, Cambridge, UK, 1979.

[162] P.N. Wild and H.F. Boysan. Modelling mass transfer, chemical reactions and combustion. In *Industrial Computational Fluid Dynamics: Lecture Series 1995-03.* von Karman Institute for Fluid Dynamics, Chaussée de Waterloo, Belgium, 1995.

[163] X. Xu, C.C. Pain, A.J.H. Goddard, and C.R.E. de Oliveira. An automatic adaptive meshing technique for Delaunay triangulations. *Comput. Meth. Appl. Mech. Engng.*, 161:297–303, 1998.

[164] X.M. Zhao, A.W. Nienow, S. Chatwin, C.A. Kent, and E. Galindo. Improving xanthan gum fermentation by changing agitators. In *Proc. 7th European Mixing Conf.*, pages 227–283, 1991.

[165] G.Y. Zhu, A. Zamamiri, M.A. Henson, and M.A. Hjortsø. Model predictive control of continuous yeast bioreactors using cell population balance models. *Chem. Engng. Sci.*, 55:6155–6167, 2000.