Modelling of gas-liquid precipitation systems

Stylianos Rigopoulos

University College London

October, 2002

Thesis submitted to the University of London for the degree of Doctor of Philosophy

ProQuest Number: 10011169

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

ProQuest 10011169

Published by ProQuest LLC(2016). Copyright of the Dissertation is held by the Author.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code.
Microform Edition © ProQuest LLC.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Acknowledgements

I wish to express my gratitude to my principal supervisor, Professor Alan Jones, for granting me the opportunity to carry out this exciting research, and for his support throughout the project. I also wish to thank my secondary supervisor, Dr. Panagiota Angeli for her support, and the Engineering and Physical Sciences Research Council (EPSRC) for sponsoring this work and for lending me for three months the Malvern Mastersizer S instrument, with which the experimental part of this work was carried out. Finally, I have benefited from the interaction with Dr. Giorgio Micale, Marcello Murru and Dr. Peter Cains.

The three years of effort that resulted in this work would not be spared without the support of my family, namely my mother, *Ερμιονη*, my father, *Βασιλειος* and my sister, *Μαρια*, who were always by my side, and of my wife, Hsing-Fen, whose love and caring provided me with inspiration and courage to go forward. The result is dedicated to them.

Abstract

The performance of gas-liquid precipitation in industrial reactors, such as bubble columns, is determined by the interplay between multiphase fluid dynamics, gas-liquid reaction engineering and crystallisation mechanisms such as nucleation, growth and agglomeration. The aim of this work is to develop efficient computational methods for dealing with these issues.

The coupling of turbulent mixing and chemical phenomena lies at the heart of multiphase reaction engineering, but direct CFD approaches are usually confronted with excessive computational demands. A hybrid approach is proposed here, where the quantification of mixing is accomplished through averaging the flow and concentration profiles resulting from a CFD flow field calculation and a computational ("virtual") tracer experiment. These results form the basis for a mapping of the CFD grid into a generalised compartmental model, where the chemistry calculations can be carried out efficiently. A phenomenological model of mass transfer and chemical reaction is employed to simulate the interfacial phenomena in gas-liquid reactors, but the main concept lends itself to a variety of chemical reaction engineering problems.

A first-order finite element scheme is proposed for solving the one-dimensional dynamic population balance equation with nucleation, growth, aggregation and breakage. The numerical scheme features improved stability by taking into account the convective nature of the growth term, and enhanced conservation of moments at non-uniform grids by identifying and mitigating the sources of conservation error in the evaluation of the aggregation source term. Exhaustive benchmarking is carried out against analytical solutions, and in all cases tested the method proves capable of producing accurate results at a reasonable CPU time.

Finally, the theoretical and computational methods developed in this work are applied to simulate the precipitation of $CaCO_3$ in a bubble column reactor. Dynamic measurements of the crystal size distribution (CSD) are carried out, and the evolution of the process is adequately reproduced by the model.

Contents

List of figures **6**

List of tables **10**

Notation **11**

Introduction **12**

1 Literature review **15**

1.1 Gas-liquid reaction engineering 15

1.1.1 Fundamentals of gas-liquid mass transfer with chemical reaction 15

1.1.1.1 Approximate models of absorption in turbulent dispersed flows 17

1.1.1.2 Effect of chemical reaction on mass transfer 19

1.1.2 Gas-liquid reactor models 21

1.2 CFD modelling of gas-liquid flow 22

1.2.1 Conventional models and Computational Fluid Dynamics . . 22

1.2.2 Overview of computational models 24

1.2.3 Fundamental models with interface tracking 25

1.2.4 Eulerian-Lagrangian model 26

1.2.5 Eulerian-Eulerian model 27

1.2.5.1 Interfacial forces 28

1.2.5.2 Bubble size distribution 31

1.2.5.3 Turbulence 32

1.3 The population balance 34

1.3.1 Origin and formulation of the PBE 34

1.3.2 Numerical solution of the PBE 36
1.4 Crystallisation and precipitation 39
1.4.1 Overview 39
1.4.2 Nucleation 41
1.4.3 Crystal growth 43
1.4.4 Agglomeration 44
1.4.4.1 Hydrodynamics 46
1.4.4.2 Interparticle forces 48
1.4.4.3 Solute concentration 48
1.4.4.4 Agglomeration kinetics 49
1.4.5 Mathematical modelling of crystallisation processes 53
1.4.5.1 MSMPR theory 54
1.4.5.2 Batch crystallisers 55
1.4.5.3 Inter-particle phenomena 57
1.4.5.4 Spatially distributed crystallisers 58

2 Hybrid CFD - reaction engineering approach 61
2.1 Introduction 61
2.1.1 Present challenges in multiphase reactor modelling 61
2.1.2 The need for hybrid approach 63
2.2 The generalised compartmental model 64
2.2.1 Basic concepts 64
2.2.2 Topology of the compartmental model 66
2.2.3 Constraints 69
2.2.4 Mass balances 69
2.2.5 Link with CFD 70
2.2.5.1 Overall tuning algorithm 70
2.2.5.2 Determination of network structure and flowrates . 71
2.2.5.3 The virtual tracer experiment 72
2.2.5.4 Extraction of mixing information from $k - \epsilon$ based CFD 74
2.2.5.5 Superposition of reaction engineering models 75
2.3 Application to bubble column reactors 76

2.3.1 Mixing in bubble column reactors 76
2.3.2 Model of interfacial mass transfer and chemical reaction . . . 77
2.3.2.1 Implementation of the penetration model 78
2.3.3 Interface equations 80
2.4 Computational implementation 81
2.4.1 Generalised compartmental model 81
2.4.2 Solution of interfacial PDEs 82
2.4.3 CFD modelling of vertical dispersed gas-liquid flows 83
2.4.3.1 Grid and time stepping 84
2.4.3.2 Boundary conditions 85
2.4.3.3 Additional phenomenological models 86
2.5 Case study: Parallel-consecutive reactions in a draft-tube bubble column 86
2.5.1 Validation of CFD model 87
2.5.2 Network structure and mean flowrates 89
2.5.3 Turbulent exchange 92
2.5.4 Superposition of chemical reactions 92

3 Finite element scheme for the time-dependent population balance 101
3.1 The need for a new approach 101
3.2 Formulation of the PBE 103
3.3 Preliminary considerations 107
3.3.1 Truncation of the domain 107
3.3.2 Grid generation 108
3.4 The main steps of the method 109
3.4.1 Approximation of the solution 111
3.4.2 Evaluation of integral terms 111
3.4.2.1 Aggregation source term 112
3.4.2.2 Aggregation sink term 115
3.4.2.3 Breakage source term 115
3.4.3 Formulation of weighted residuals 116
3.4.3.1 Growth term 117
3.4.3.2 Nucleation term 118

3.4.4 Temporal Integration 118
3.5 Results and discussion 119
3.5.1 Aggregation 119
3.5.1.1 Case I-1: Size-independent kernel 120
3.5.1.2 Case I-2: Sum (Golovin) kernel 120
3.5.1.3 Case I-3: Fluid shear and gravitational kernels . . . 121
3.5.2 Nucleation-growth 123
3.5.2.1 Case II-1: Size-independent linear growth and nucleation with flow 124
3.5.2.2 Case II-2: Size-dependent growth and nucleation with flow 124
3.5.2.3 Case II-3: Batch Size-independent nucleation-growth 125
3.5.3 Breakage 126
3.5.3.1 Case III-1: Uniform binary breakage 128
3.5.3.2 Case III-2: Parabolic binary breakage 129
3.5.3.3 Case III-3: Multiple breakage (fragmentation) . . . 129
3.5.4 Combined problems 130
3.5.4.1 Case IV-1: Steady-state nucleation - aggregation . . 131
3.5.4.2 Case IV-II: Batch aggregation-growth 132
3.5.5 Discussion 133

4 Gas-liquid precipitation of $CaCO_3$ in a bubble column reactor 136
4.1 Introduction 136
4.1.1 Industrial importance 136
4.1.2 Present challenges 137
4.1.3 Modelling of gas-liquid precipitation with the method of moments 138
4.1.4 Deficiencies of the moments solution 140
4.2 Experimental part 142
4.2.1 Experimental setup 142
4.2.1.1 Gas-liquid precipitation rig 142
4.2.1.2 Measurement of crystal size via laser light scattering 143

4.2.1.3 Examination of crystal morphology via Scanning Electron Microscopy 145
4.2.2 Results and discussion 146
4.2.2.1 Time course of precipitation 146
4.2.2.2 Product morphology 146
4.2.3 Interpretation of results 147
4.3 Process simulation 155
4.3.1 Kinetics of $CaCO_3$ precipitation 155
4.3.1.1 Reaction system 155
4.3.1.2 Supersaturation 157
4.3.1.3 Crystallisation kinetics 158
4.3.2 Modelling methodology 162
4.3.2.1 CFD modelling and virtual tracer experiment . . . 163
4.3.2.2 Superposition of precipitation 164
4.3.3 Simulation results 166

5 Conclusions 176
5.1 Hybrid modelling of multiphase reactors 176
5.2 Solution of the population balance with finite elements 177
5.3 Gas-liquid precipitation of $CaCO_3$ 178
5.4 Suggestions for future work 179

A Solution of the interface PDEs 181

B Source files and user subroutines for the CFX 4.3 code 183
B.1 CFX source file 183
B.2 Fortran user subroutines 188

C Fortran 90 source code 200
C.1 PBE 200
C.2 Network 227
C.3 Interface 246
C.4 Other programs 255

List of Figures

1.1 Values of the drag coefficient computed by various correlations. . . . 30

1.2 Evolution of the population density. 35

2.1 Mapping of a CFD grid into a compartmental network. 65

2.2 The building block of the multi-phase generalised compartmental model (two-phase). 68

2.3 Penetration theory: physical interpretation and implementation in the model 79

2.4 Notation used for the derivation of the discretised equations at the interface. 83

2.5 Schematic of a draft-tube bubble column. 87

2.6 Gas hold-up in the riser and downcomer, comparison of CFD model with the experiments of Wachi et al. (1991). 88

2.7 CFD results (left to right): Volume fraction, mean velocity profile (m/sec) and dimensionless scalar concentration (resulting from the virtual tracer experiment), axisymmetric simulation, column axis on the right. 90

2.8 Radial gas holdup profiles in the column at four axial locations. . . . 91

2.9 Compartmental network with two vertical layers of riser cells, to emulate the feeding policy of a bubble column 93

2.10 Concentration profiles in the riser, network of fig. 2.9. 94

2.11 Concentration profiles in the downcomer, network of fig. 2.9. 94

2.12 The network of 2.9 is improved to account for lateral turbulent exchange in the riser. 95

2.13 Concentration profiles in the riser, network of fig. 2.12. 96

2.14 Concentration profiles in the downcomer, network of fig. 2.12. 96
2.15 Equilibrium concentrations at different values of pH for the system CO_2 - HCO_3^- - $CO_3^=$ 97
2.16 A sample of simulated concentration profiles in the interface 98
2.17 Evolution of pH. 99
2.18 Evolution of chemical species' concentrations. 99
2.19 Physical and chemical absorption. 100

3.1 The main steps of the method. 110
3.2 Geometrical representation of the conservation error arising in non-uniform grids. 113
3.3 The aggregation source term is of higher order than the approximated population density (which is linear). To obtain exact quadrature, a number of points must be interpolated within each interval. 114
3.4 Growth makes the population density move towards larger sizes only - hence the need for upwind discretisation. 118
3.5 Aggregation, constant kernel, comparison of simulation with the analytical solution of Scott (1968) 121
3.6 Aggregation, sum kernel, comparison of simulation with the analytical solution of Scott (1968) 122
3.7 Aggregation, constant, sum, fluid shear and gravitational kernels, simulation results. 123
3.8 Steady-state size-independent nucleation-growth, comparison with the analytical solution of Randolph and Larson (1988). 125
3.9 Steady-state size-dependent nucleation-growth, comparison with the analytical solution of Randolph and Larson (1988). 126
3.10 Batch size-independent nucleation-growth, comparison with the analytical solution of Hounslow et al. (1988) 127
3.11 Batch uniform binary breakage, comparison of simulation and analytical solution of Ziff and McGrady (1985). 128
3.12 Batch parabolic breakage for h=0,1,2 and comparison with analytical solution (Ziff and McGrady, 1985) for h=2. 130

3.13 Batch ternary fragmentation, comparison of simulation and analytical solution of Ziff (1991). 131
3.14 Steady-state aggregation (constant kernel) and nucleation, comparison with semi-analytical solution of Hostomsky (1987). 133
3.15 Batch aggregation (constant kernel) and growth (size-dependent), comparison of simulation and analytical solution of Ramabhadran et al., (1976) 135
3.16 Evolution of first moment (mass) in the aggregation-growth problem - comparison of simulation and analytical solution of Ramabhadran et al. (1976). 135

4.1 Reactors for gas-liquid precipitation - (a) bubbling stirred tank, (b) flat interface stirred tank, (c) draft tube bubble column, (d) spray column (reprinted from Jones, 2002) 139
4.2 Stage-wise model for a draft-tube bubble-column reactor (Rigopoulos and Jones, 2001) 140
4.3 Mean size and pH predicted with the method of moments (Rigopoulos and Jones, 2001) 141
4.4 Experimental bubble column rig (21lt) 143
4.5 Evolution of mean size during a typical experiment 147
4.6 Evolution of CSD during a typical experiment 148
4.7 Evolution of pH and Ca^{++} during a typical experiment 149
4.8 Large-scale view of the precipitate, shortly after the induction time . 149
4.9 A closer view of the single product crystals (a) 150
4.10 A closer view of the single product crystals (b) 150
4.11 Close view of an agglomerate formed at an early stage of the process) 151
4.12 View of the precipitate at a later stage of the process (a) 151
4.13 View of the precipitate at a later stage of the process (b) 152
4.14 Close view of an agglomerate, at a later stage stage of the process . 152
4.15 An agglomerate during a very late stage of the process. 153
4.16 Left: shaded contours of volume fraction, right: passive scalar concentration (axisymmetric simulation, column axis on the right). . . . 165

4.17 Experimental and simulated evolution of pH and [Ca^{++}]: [$Ca(OH)_2$] = 3 mol/m^3, 0.00001 : 0.0001 m^3/s CO_2:N_2 168
4.18 Experimental and simulated evolution of pH and [Ca^{++}]b)[$Ca(OH)_2$] = 4 mol/m^3, 0.00001 : 0.0001 m^3/s CO_2:N_2 168
4.19 Experimental and simulated evolution of pH and [Ca^{++}]: c)[$Ca(OH)_2$] = 4 mol/m^3, 0.00001 : 0.00005 m^3/s CO_2:N_2 169
4.20 Simulation of the evolution of supersaturation in a typical experiment. 169
4.21 Experimental and simulated CSD, logarithmic plot. [$Ca(OH)_2$] = 3 mol/m^3, 0.00001 : 0.0001 m^3/s CO_2:N_2. 170
4.22 Experimental and simulated CSD, cumulative undersize plot. [$Ca(OH)_2$] = 3 mol/m^3, 0.00001 : 0.0001 m^3/s CO_2:N_2. 171
4.23 Experimental and simulated CSD, logarithmic plot. [$Ca(OH)_2$] = 4 mol/m^3, 0.00001 : 0.0001 m^3/s CO_2:N_2 172
4.24 Experimental and simulated CSD, cumulative undersize plot. [$Ca(OH)_2$] = 4 mol/m^3, 0.00001 : 0.0001 m^3/s CO_2:N_2 173
4.25 Experimental and simulated CSD, logarithmic plot. [$Ca(OH)_2$] = 4 mol/m^3, 0.00001 : 0.00005 m^3/s CO_2:N_2. 174
4.26 Experimental and simulated CSD, cumulative undersize plot. [$Ca(OH)_2$] = 4 mol/m^3, 0.00001 : 0.00005 m^3/s CO_2:N_2. 175

List of Tables

1.1 Some correlations for calculation of the drag coefficient. 30

2.1 Parameter values and their sources. 97

3.1 Results for the aggregation cases. 124

3.2 Nucleation-growth cases, parameters and results. 127

3.3 Breakage cases, parameters and results. 132

3.4 Combined cases parameters and results. 134

4.1 Parameter values used in the simulations. 162

Notation

Index notation is used throughout. Repeated indices imply summation while single indices denote iteration, i.e.,

$$\begin{aligned}\frac{\partial y_i}{\partial x_i} &= \frac{\partial y_1}{\partial x_1} + \frac{\partial y_2}{\partial x_2} + \ldots + \frac{\partial y_n}{\partial x_n} \\ \frac{\partial}{\partial x_i} &= \left[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \ldots, \frac{\partial}{\partial x_n}\right]\end{aligned}$$

Although effort was taken to ensure a consistent notation, due to the fact that different chapters deal with different problems, occasionally the same symbol may be used to denote different variables in different chapters (e.g. index i denotes chemical species in chapters 2, 4 but element boundaries in chapter 3). The notation is consistent within each chapter, however, and symbols are explained when first encountered.

The substantial derivative is defined as:

$$\frac{D}{Dt} = u_i \frac{\partial}{\partial x_i} + \frac{\partial}{\partial t}$$

Some of the acronyms frequently used are:

CFD Computational Fluid Dynamics

CSD Crystal Size Distribution

PBE Population Balance Equation

Introduction

> "If it is not permitted to us to penetrate to a complete knowledge concerning the motion of fluids, it is not to mechanics, or to the insufficiency of the known principles of motion, that we must attribute the cause. It is analysis itself which abandons us here, since all the theory of the motion of fluids has just been reduced to the solution of analytical formulae."
>
> Leonhard Euler, 1755

During the last decades, engineering and process design have been revolutionized by the introduction of computational methods. Computational fluid dynamics (CFD) have emerged as a powerful, fully predictive tool for the design of processes and equipment where fluid mechanics play a key role. The application of CFD to chemical reaction engineering has lagged, however, and is confronted with unique challenges. The coupling of CFD with chemical phenomena gives rise to issues not usually encountered in purely hydrodynamic applications. These issues are both physical and computational, and are unlikely to be resolved in the near future. Chemical phenomena occur at scales much smaller than those characteristic of hydrodynamic phenomena, and the computational resources required to resolve these scales are unlikely to be available soon, unless unexpected breakthroughs in computer hardware are due.

This leads to the inevitable conclusion that hybrid methods are required, if we are to proceed from the realm of empiricism and make use of the currently available computational resources, without pursuing unfeasible targets. To develop them we should relax our requirements for fundamental models and settle for approaches that, though fully predictive, are partly based on phenomenological concepts. Actually, an element of phenomenology is always inevitable; even purely hydrodynamic problems require turbulence modelling. The first point of focus of this work is therefore the

proposal of a hybrid methodology for chemical reactor modelling, that makes use of both CFD and reaction engineering concepts.

Statistical concepts are inherent in a variety of reaction engineering problems: crystals, bubbles, droplets, polymers and cells are some examples of entities encountered in multiphase reactors that cannot be adequately described by a single value but require a distribution. The most rigorous method to account for such processes is the population balance, an integro-differential equation that describes the evolution of a particle size distribution. This equation can only be solved numerically, and methods often encounter problems regarding stability and conservation of the moments. The second point of focus is the development of a finite element scheme that makes certain advances towards resolving these problems.

Precipitation is a crystallisation process driven by supersaturation induced by chemical reactions - instead of cooling, evaporation of solvent etc. Gas-liquid precipitation is therefore the formation of a solid crystalline product via chemical reactions involving a gas and a liquid phase. The analysis of this process is a truly interdisciplinary problem, as it must account for a unique combination of physical and chemical phenomena: multiphase fluid dynamics, that determine several key reaction engineering variables such as the gas hold-up, chemical reactions may occur at both the gas-liquid interface and bulk, and various particulate mechanisms such as nucleation, growth and agglomeration play a role in determining the crystal size distribution (CSD) of the product. Therefore it encompasses all the fundamental problems which the aforementioned computational methods aim to address, and provides an excellent case study of applying them. The precipitation of $CaCO_3$ via the reaction of gaseous CO_2 and dissolved $Ca(OH)_2$ is considered here.

This thesis is structured as follows. Chapter 1 reviews the state of the art in four distinct fields that are relevant to the material of this work:

Gas-liquid reaction engineering: The analysis of mass transfer and chemical reaction, and the development of multiphase reactor models.

Multiphase fluid dynamics: The contribution of CFD to the analysis of multiphase flow; current status and problems.

Population balance: Emphasis will be on numerical solution techniques and the

problems that they face.

Crystallisation: Mechanistic models describing the crystallisation mechanisms (nucleation, growth and agglomeration), that must be integrated in the population balance.

In chapter 2, the hybrid methodology for multiphase reactor modelling is presented. A phenomenological model of mass transfer and chemical reaction for use with bubble column reactors is also employed, though the main concepts are applicable to a variety of reaction engineering problems. A $k-\varepsilon$ - based code is employed for the CFD simulations, but the method is not restricted to it and will benefit from any future developments in turbulence modelling. The application of the method is demonstrated with a case study (CO_2 absorption in a bubble column).

The finite element method for solution of the population balance equation is developed in chapter 3. The accuracy of a numerical method can be tested by comparing against analytical solutions that have been found for special cases. Such benchmarking is carried out against solutions for size-dependent growth and nucleation, size-dependent aggregation with a variety of kernels, binary and multiple breakage with uniform and parabolic daughter particle distributions and combinations of the above, respectively.

Finally, these computational methods are applied to the gas-liquid precipitation problem (chapter 4). Experiments on $CaCO_3$ precipitation via reaction of CO_2 and $Ca(OH)_2$ in a $21lt$ bubble column are conducted and simulated with the aid of the model. Concluding remarks follow (chapter 5), while the computational implementation of the methods in this work is presented in the appendices in the form of Fortran 90 programs.

Chapter 1

Literature review

1.1 Gas-liquid reaction engineering

1.1.1 Fundamentals of gas-liquid mass transfer with chemical reaction

The interaction of mass transfer with chemical reactions at an interface lies at the heart of multiphase reaction engineering. The following discussion will outline the fundamental equations and the levels of approximation on which the current models are based. The fundamental mass transport equation for a chemical component can be written as [1]:

$$\frac{\partial C_i}{\partial t} + \frac{\partial}{\partial j}(u_j C_i) = \frac{\partial J_{ij}}{\partial j} + \sum_k r_k(C_i) \tag{1.1}$$

where J_{ij} is the diffusion of species i in the direction j, usually described by Fick's law. For a rigorous derivation of the transport equations see e.g. Landau and Lifshitz (1959) or Bird, Stewart and Lightfoot (1960). When eq. 1.1 is applied to the gas-liquid interface, the hydrodynamic conditions are usually not known, so it must be coupled with the momentum (Navier-Stokes) equations:

$$\frac{\partial \rho u_j}{\partial t} + \frac{\partial}{\partial j}(\rho u_j \cdot u_l) = \frac{\partial \tau_{jl}}{\partial j} - \frac{\partial p}{\partial j} + F_b \tag{1.2}$$

where τ_{jl} is the stress tensor, while F_b stands for body forces such as gravity, if present. The microscopic scale of interfacial phenomena means that a discretisation

[1] In this section index i stands for chemical species, j or l for spatial dimension (x, y, z) and k for chemical reaction.

in the sub-micron range is required to resolve them. Moreover, the location and shape of the interface is usually not known and must be determined by the solution. Therefore, few attempts have been made to resolve mass transfer along with the interfacial velocity field.

The early work of Lochiel and Calderbank (1964) considered mass transfer from bubbles as diffusion with a moving boundary and derived solutions for it using the approximations of boundary layer theory. Such models cannot be applied in many real situations, because they oversimplify both the velocity field and interface shape. Recent advances in CFD and interface tracking techniques have enabled the numerical simulation of free surface flow, and recently Shyy, Udaykumar, Rao and Smith (1997) have provided a review of numerical methods for simulation of interfaces. Liao and McLaughlin (2000) have solved the problem of mass transfer from bubbles numerically. Although such approaches are valuable for providing insight into the interfacial phenomena, they have no practical use for large scale reactor simulation, because that would imply resolving the whole domain with a microscopic grid. For the time being they are restricted to the simulation of single bubbles or small groups.

In order to formulate a numerically tractable interfacial mass transfer model with potential for coupling with complex reaction systems, simplification of the hydrodynamics is necessary. The most idealised hydrodynamic conditions emerge when the interface is stagnant, i.e. $u_j = 0$. In that case eq. 1.1 is reduced to the following:

$$\frac{\partial C_i}{\partial t} = \frac{\partial}{\partial j} J_{ij} + \sum_k r_j(C_i) \tag{1.3}$$

If the interface is flat and very long in both lateral dimensions, the above equation would be one-dimensional in terms of space. By applying Fick's law for the diffusive flux, we arrive at the following:

$$\frac{\partial C_i}{\partial t} = D\frac{\partial^2 C_i}{\partial x^2} + \sum_k r_k(C_i) \tag{1.4}$$

This is a set of two-dimensional parabolic PDEs. Danckwerts (1970) identified this model with the case of absorption into a semi-inflnite quiescent liquid and described experiments to which it can be directly applied. In multiphase reaction engineering, however, interest lies mainly in dispersed turbulent flows, which offer

rapid mass transfer rates. Since resolution of interfacial hydrodynamics in such flows is very difficult, the proposed models attempt to draw analogies with idealised cases where hydrodynamics play no role (i.e. $u_j = 0$) such as in eq. 1.4. It follows that they are to a large degree phenomenological; yet they are often used due to their tractability.

1.1.1.1 Approximate models of absorption in turbulent dispersed flows

The film model is based on the concept that the gas must be dissolved in the surface by molecular conduction before it can be transferred to the interior by convection (Danckwerts, 1970). Interphase mass transfer is assumed to takes place within a thin layer next to the surface under steady state conditions. Eq. 1.4 is then reduced to the following system of ODEs:

$$D\frac{d^2C_i}{dx^2} + \sum_k r_k(C_i) = 0 \tag{1.5}$$

If mass transfer resistance is in both gas and liquid phases, a two-film model is needed; however, this study is concerned with liquid-phase reactions where gas phase resistance is usually negligible (Deckwer, 1992). In the following, we shall refer to the liquid side of the interface. Initial conditions for the volatile components are estimated by an equilibrium relationship, such as Henry's law, while at the end of the film the profiles must approach the bulk concentrations. The only parameter, the film thickness, must be estimated experimentally. The concept of the film model is not very realistic, as it implies an unnatural discontinuity of concentration near the surface; moreover, its solution for the case of physical absorption, assuming that the bulk is in steady state operation, shows the liquid side mass transfer coefficient k_l to be proportional to the diffusion coefficient D_i, while experimental evidence rather suggests proportionality to $\sqrt{D}$ (Astarita, 1967). Nevertheless, it has been used in a number of studies due to its computational advantage of involving ordinary differential equations rather than partial, as do the other models.

A more realistic approach was proposed by Higbie (1935) and is known as the penetration model. Its physical reasoning is that elements or 'parcels' of liquid are assumed to be brought from the bulk to the surface by the flow pattern, and to remain there for a small amount of time, during which they behave like rigid

bodies. The contact time is so small that the depth of gas penetration into the liquid element is negligible compared to the depth of the element itself. This way, it draws an analogy between gas absorption into agitated liquids and absorption into a quiescent semi-infinite liquid, and eq. 1.4 can be applied directly.

One initial and two boundary conditions are required. The initial conditions express that in the beginning of the contact time the liquid element has the bulk concentrations everywhere. On the gas side of the interface, Dirichlet boundary conditions are imposed for the volatile component, specifying its concentration by an equilibrium condition such as Henry's law, while zero-flux conditions can be applied to non-volatile species. The assumption of small penetration depth and the approximation of the liquid element as a semi-infinite quiescent liquid are implemented by imposing the bulk side boundary conditions at an infinite distance from the interface:

$$t > 0,\ x \rightarrow \infty \ \Rightarrow \ C_i = C_{i,bulk} \tag{1.6}$$

The penetration model accounts for an important feature of the absorption mechanism: instead of being stationary, the interfacial layers are being constantly replenished. This replenishment can be either due to turbulence, as in the case of an agitated tank, or due to laminar flow over a packing or around a bubble, in which case the fluid is mixed after having crossed the bubble or solid object. The penetration model replaces the parameter of the film thickness with the contact time. In many cases it can only be empirically determined, but sometimes, such as in the case of rising bubbles, it can be estimated on the basis of physical arguments. On the other hand, the fact that partial differential equations are involved means that it is much more difficult to obtain solutions. For physical absorption (without reactions) the following analytical solution has been derived (Danckwerts, 1970):

$$C_i = (C_i^* - C_{i,bulk}) \cdot erf\left(\frac{x}{2\sqrt{D\tau}}\right) + C_{i,bulk} \tag{1.7}$$

Toor and Marchello (1958) proposed the following modification to the penetration model: the interface is assumed to have a fixed depth, such that the concentration gradients are zero when approaching the bulk. This has the same effect as the original Higbie model on the following condition:

$$\tau \leq \frac{\delta^2}{D} \tag{1.8}$$

This implies that it is more appropriate for short contact times. In effect it imposes the bulk-side boundary conditions at a finite distance δ instead of infinity, which is a major advantage for obtaining numerical solutions.

A shortcoming of the penetration model is that it assumes all elements to remain on the surface for exactly the same amount of time. To account for this, the Danckwerts model of surface renewal (Danckwerts, 1970) introduces the concept of age distribution: the contact time is not the same for all elements but follows a distribution instead. Eq. 1.4 still applies, but the absorption rate is calculated by integration over all element ages. Assuming that the probability of an element to be renewed is independent of its age, the average absorption rate would be:

$$J = s \int_0^\infty J e^{-s\theta} d\theta \tag{1.9}$$

In essence, though, the surface renewal theory replaces the parameter of the contact time with that of renewal rate s. More complicated models include the eddy diffusivity theory, due to King (1966), which treats the problem of absorption in turbulent flows by splitting the diffusion coefficient into molecular and eddy diffusivity. This, however, increases the number of parameters to two. Another two-parameter approach is the surface-rejuvenation model. Two-parameter models have been seldom applied in numerical studies, because their higher degree of mechanistic insight is countered by the increased uncertainty in the parameter estimation.

1.1.1.2 Effect of chemical reaction on mass transfer

Solution of eqs. (1.4) or (1.5) becomes very complicated in the presence of chemical reactions (chemisorption), as reaction kinetics are highly non-linear. A comprehensive survey of gas-liquid reactions may be found in Doraiswamy and Sharma (1983). In some cases it is possible to decouple these phenomena by comparing their relative rates through the Hatta number, defined as:

$$Ha = \frac{\text{rate of reaction}}{\text{rate of absorption}} \tag{1.10}$$

The exact expression of the Hatta number depends upon the particular reaction system. According to this comparison, the following cases or absorption regimes can be identified:

- If the chemical reactions are much slower than mass transfer (kinetic control) then the situation can be considered as physical absorption, with chemical reactions occurring subsequently in the bulk. In this case, analytical solutions for mass transfer in both the film and penetration models can be derived.

- When the reactions are instantaneous (diffusion control) the reactants are assumed to diffuse up to a reaction plane situated at some distance inside the interface, where reactions happen instantaneously. This is another simplified case where mass transfer can be decoupled from chemical reactions and analytical solution is possible.

- If the reactions are fast but not instantaneous, they will occur simultaneously with mass transfer along the interface. This is clearly the most complex case, as reaction kinetics must be included in the mass transfer equations. Astarita (1967) and Danckwerts (1970) have obtained analytical solutions for some simple reactions (e.g. zero order, first order irreversible), but in the general case of a complex arbitrary reaction system numerical treatment is necessary. Versteeg, Kuipers, Van Beckum and Van Swaaij presented such a treatment for the single reversible (1989a) and the parallel reversible (1989b) reaction problems. Their solutions was based on both the film and penetration models, and a spatial transformation was introduced to impose the infinity boundary condition of the later. A similar treatment by Glasscock and Rochelle (1989) included the eddy diffusivity model. Van Swaaij and Versteeg (1992) presented a review of the topic and emphasised the necessity for numerical approach.

Another dimensionless modulus used to describe the interaction of absorption with chemical reactions is the enhancement factor:

$$E_f = \frac{\text{rate of chemisorption}}{\text{rate of physical absorption}} \tag{1.11}$$

In the fast reaction regime chemisorption is faster than physical absorption, because the consumption of the volatile component as it is being absorbed increases the driving force for mass transfer; absorption is thus "enhanced". Analytical expressions for the enhancement factor cannot be derived but for the simplest cases, where chemisorption can be resolved.

The rigorous analysis of gas-liquid phenomena presented in this chapter refers to the microscopic interfacial region. We shall now see how these models can be implemented in the modelling of industrial multiphase reactors.

1.1.2 Gas-liquid reactor models

The flow patterns that arise in industrial-scale gas-liquid reactors are so complex that detailed modelling of interfacial phenomena is not possible. Due to this complexity design methods remained for a long time experimental. The simplest way to include mass transfer in a reactor model is in the form of overall coefficients:

$$J = k_l a(C^* - C) \tag{1.12}$$

Experimental approaches for determining overall mass transfer coefficients have been reviewed by, among others, Shah, Kelbar and Godbole (1982) and Deckwer (1992). Despite dimensionless analysis, empirical correlations for the mass transfer coefficient are rooted to the equipment used for extracting them and their reliability during scale-up is limited. Moreover, this approach can only be applied to steady state problems or simple reaction systems where reaction enhancement can be evaluated analytically. The complexity increases dramatically when mass transfer is accompanied by a fast chemical reaction. In this case mass transfer and chemical reactions are closely coupled, and the reaction enhancement effect can only be studied in detail by application of one of the gas-liquid absorption theories described above.

With respect to the hydrodynamic aspects of the problem, empirical stagewise or axial dispersion models have been frequently used. Deckwer, Hallensleben and Popovic (1980) employed the cell model with backflow to simulate physical absorption of CO_2 in a bubble column. Wachi and Morikawa (1986) applied the same model to the chlorination of ethylene, a problem of absorption with instantaneous chemical reaction. In the later, an analytical solution of the penetration model for the specific reaction system was employed to estimate the reaction enhancement, but the mass transfer coefficient was given by an empirical correlation. Schluter, Steiff and Weinspach (1992) adopted a similar approach for modelling mass transfer and reaction with both the cell model with backflow and the axial dispersion.

Wachi and Jones (1991) and Hostomský and Jones (1995) employed the film and penetration model respectively to model mass transfer, reaction and precipitation in a stirred tank reactor with a flat interface. A similar implementation of the penetration model was recently proposed to evaluate reaction enhancement in stirred vessels by Van Elk, Borman, Kuipers and Versteeg (2000), and the same authors made a stability and bifurcation analysis of the behaviour of gas-liquid reactors based on this approach (1999).

CFD models of gas-liquid reactors have appeared during the last decade, but are usually restricted to pure hydrodynamic simulation. Joshi (2001) has recently reviewed the current state of the art in the CFD modelling of bubble columns. Severe computational obstacles are met, however, when the coupling of gas-liquid mass transfer and hydrodynamics is considered. Torvik and Svendsen (1990) developed a steady-state two-dimensional CFD model of a gas-liquid reactor, while Fleischer, Becker and Eigenberger (1995, 1996) proposed an one-dimensional dynamic model for a thin and tall bubble column. In these studies, however, interfacial mass transfer and chemical reaction were decoupled. Recently Zahradnik et al. (2001) have applied the "network of zones" approach to gas-liquid reactor modelling. This approach, originating from the analysis of mixing in stirred tanks (Mann and Hackett, 1988), offers a computationally attractive alternative to the more expensive CFD models.

1.2 CFD modelling of gas-liquid flow

1.2.1 Conventional models and Computational Fluid Dynamics

Initially, hydrodynamic studies of bubble columns focused on experimental measurements and simple analytical models. The main design parameters were identified to be the gas superficial velocity, size, height-to-diameter ratio and the type and operation of sparger. These parameters were cast into dimensionless terms and correlated experimentally to the quantities that determine the performance of the reactor, such as slip velocity, liquid circulation rate, phase distribution and interfacial area. Such correlations along with analytical models were reviewed by Shah et al. (1981), Deckwer (1992), Kastanek et al. (1993) and Joshi, Ranade et al. (1990). Due to the complexity of the phenomena underlying multi-phase flow however, experimental

correlations lack wide predictive potential and remain tied to the equipment used for their determination.

One of the first theoretical methods for the analysis of multi-phase flow was the drift flux model, proposed by Wallis (1969). Several studies during the 80's such as Koide et al. (1984), Jones (1985) and Chisti, Halard and Moo-Young (1987) attempted to predict the liquid circulation rate based on energy balances, while turbulence and other features that result in energy losses were implemented in the form of empirical coefficients. These models contained simple algebraic equations and were useful for interpreting experimental data, though quantitative agreement was difficult to achieve.

The developments in digital computers and numerical methods during the 70's made possible the numerical solution of the fundamental differential equations of fluid dynamics in single-phase flows, yet this approach could not be easily extended to multi-phase systems. In contrary to single-phase systems, a gap in our fundamental knowledge of the physics underlying multi-phase flow, especially of interfacial phenomena, meant that it would remain an active area of research. Ishii (1975) made one of the first attempts to establish the appropriate form of the fundamental equations governing multi-phase flows, and discussed various kinds of statistical averaging (time, space, ensemble) through which they can be derived. Stewart and Wendroff (1984) presented a review of the theoretical advances to-date, and concluded that fundamental knowledge was still lacking. Insofar as the computational aspects are concerned Spalding (1980) proposed the Inter-Phase Slip Algorithm (IPSA), an efficient way for numerical solution of the Eulerian-Eulerian formulation of the multi-phase flow equations, that became the industry standard. In the following years, some multi-phase flow problems were approached with computational methods.

It was mostly during the 90's, however, that computational modelling of bubble columns in particular was investigated. Schwarz and Turner (1988) and Svendsen (1992) carried out steady-state calculations from the Eulerian-Eulerian viewpoint. Sokolichin and Eigenberger (1994 I,II) introduced dynamic modelling and compared it to steady state; subsequent researchers, such as Boisson and Malin (1996), Sanyal et al. (1999) also explored the transient approach. Delnoij et al. (1997) discussed the application of the Volume of Fluid (VOF), a method for interface tracking, to

bubble rise studies. A recent review of the state of the art was presented by Jakobsen et al. (1997). Most recently, Pfleger, Gomes et al. (1999) studied the applicability of turbulence models to gas-liquid flow, while Thakre and Joshi (1999) discussed the drag force formulation.

1.2.2 Overview of computational models

In contrary to single-phase fluid flow, where it is generally accepted that the Navier-Stokes equations constitute an accurate model of the phenomena involved (including turbulence, in terms of instant velocity), the physics underlying multi-phase flows are not yet completely understood, despite considerable experimental and theoretical effort. A major difficulty lies in the fact that the structure of multi-phase flows can take completely different forms. In the case of vertical gas-liquid flow, for instance, the basic regimes have been identified as bubbly flow, slug flow, churn-turbulent flow, annular flow and mist flow, the main parameter that determines the regime being the gas superficial velocity (or the slip velocity, in case the liquid too is moving). In general it is not possible to fully predict the flow structure from a fundamental model; instead, the flow regime to be simulated must be born in mind in order to formulate an appropriate multi-phase flow model. Some models are very regime-specific, based on the particularities of each structure, while others are more general; however, a-priori knowledge of the regime seems indispensable at present.

Multi-phase models for gas-liquid flows can be generally classified into three types:

1. Phenomenological models that comprise algebraic equations based on simplifying assumptions and mechanistic insight.

2. Semi-fundamental models, derived from the Navier-Stokes through statistical averaging procedures that, however, give rise to terms whose closure must be carried out empirically.

3. Fundamental models, based on the Navier-Stokes equations, that require tracking of all interfaces.

Models of the first type were the only option at an age when differential models were too expensive to compute; yet they can sometimes be combined with a CFD

code or assist in estimating parameters. They will not be dealt with in this discussion. The third class of models is applicable in separated flows, or for modelling microscopic phenomena in dispersed flows; it will be briefly discussed in the next section. At present, large-scale CFD modelling of dispersed flows can only be carried out with models of the second type.

1.2.3 Fundamental models with interface tracking

When a single set of Navier-Stokes equations is used to describe a multiphase flow field, the variables exhibit a discontinuous change at the interfaces, that must in general be implemented with appropriate jump conditions. The main complexity in this kind of analysis arises from the fact that the location of the discontinuities is not known *a priori,* and a method for interface tracking (i.e. simulating the detailed shape of the interface) is required. Such methods have been reviewed by Shyy, Udaykumar, Rao and Smith (1997). Whatever the method, however, an essential prerequisite for this level of approximation is that the grid be fine enough to resolve the structure of the interface. This limits the applicability of this class of models to the following situations:

- Separated flows, e.g. annular flows, channel flows etc.
- Microscopic studies of dispersed flows, where single or few bubbles are being simulated.

In the case of multiphase reactors the model can only be used for microscopic simulation of bubbles or droplets, involving extremely fine grids. In the latter case, numerical diffusion may spoil the shape of the interface. This problem may be faced with techniques such as the one proposed by Hirt and Nichols (1981), which locates the interface and then moves fluid to either side of it. The incorporation of surface tension is a problem, as the exact orientation of the interface is not known with precision. Delnoij, Kuipers and van Swaaij (1997) applied this method (commonly referred to as Volume of Fluid) to simulate the formation and trajectory of a single bubble through a quiescent liquid. The reported computation time is rather high, and certainly the method is unsuitable for modelling and scaling up a bubble column as it operates at the microscopic scale.

1.2.4 Eulerian-Lagrangian model

In this model the liquid phase is treated as a continuum and the bubbles as discrete particles, in a way similar to the tracking of solid particles in combustion models. The continuous phase is viewed from an Eulerian perspective, i.e. with respect to a fixed coordinate system, and the complete Navier-Stokes equations are applied using a variable density, averaged on a volume fraction basis. For the particulate (bubble) phase a Lagrangian perspective is adopted, solving a momentum equation for each individual bubble. The momentum equation for the bubbles is written, using the substantial derivative, as:

$$\frac{D\rho_g u_{g,j}}{Dt} = \sum F \tag{1.13}$$

This is the simplest form of Newton's law, and $\sum F$is the sum of forces acting on each bubble. These may include: gravity force, drag force, pressure force, virtual mass, Basset history term and transversal lift (Jakobsen et al., 1997).

With this type of model, not only phase distribution but also individual bubble trajectories can be calculated. Furthermore, a distribution of bubble sizes can easily be introduced, since each bubble is accommodated by a separate equation, and even bubble-bubble interactions can be incorporated. Thus, this type of model is the most detailed among multi-fluid models for simulating bubbly flows. However, it has an apparent disadvantage: when a large number of bubbles is to be simulated computational time can be excessive, as a separate transport equation has to be solved for each one of them. Thus, it has been used mainly for fundamental studies of hydrodynamic phenomena, or to assess the validity of closure laws (such as equations for drag coefficients) to be used in an Eulerian-Eulerian two fluid model (Delnoij et al., 1997). Several studies have employed the Eulerian-Lagrangian model to simulate bubble columns, notably Lapin and Lubbert (1994) and Sokolichin et al. (1997). The correct form of the fluid-particle and fluid-fluid forces is still disputed, though. It has also been suggested (Delnoij et al., 1997) that this type of model, combined with a volume tracking model, could be used to simulate the churn flow in the heterogeneous bubble regime.

1.2.5 Eulerian-Eulerian model

In this model, the two phases are treated as interpenetrating continua. The interpretation of this concept is that the values of the variables at each point are what we would measure on average if we were to carry out a large number of experiments with the same conditions. Consequently, a complete set of Navier-Stokes equations is written for each phase. The transport equation of a scalar φ takes the form:

$$\frac{\partial(u_j \rho \epsilon \varphi)}{\partial t} + \frac{\partial}{\partial x_j} \cdot (\epsilon \rho u_j \varphi) = \frac{\partial}{\partial j}(\epsilon J_{\varphi,i}) + \rho \cdot S_\phi + K \cdot \Delta\varphi + \dot{m} \cdot \varphi \tag{1.14}$$

where j denotes the spatial dimensions. This equation is derived via ensemble averaging procedures; one may consult Ishii (1975), Stewart and Wendroff (1981), Drew (1983) and Drew and Passman (1998) for details of the derivation. The first four terms represent accumulation, convection, diffusion and production or consumption (the source term - S_ϕ), and are identical to their single-phase counterparts apart for being multiplied with the local volumetric fraction (ϵ) of the phase in question. The last two terms in the right-hand side of the equation represent interfacial transport (usually proportional to the difference in property φ between the two phases, hence generally depicted as $k\Delta\varphi$) and transport due to thermodynamic change of phase. It is important to distinguish between the two: in a gas-liquid flow, for instance, the former could be used to describe gas-liquid mass transfer due to absorption, while the latter would denote condensation or evaporation.

The two aforementioned terms establish the link between the two sets of equations describing each phase. The model introduces, though, new parameters in order to describe interfacial transfer. Interfacial momentum transfer is always present in the form of drag force, while mass transfer does not always take place. For every property that is transferred through the interface a closure law has to be included in the model. The determination of these closure laws is, as it will be shown, one of the most important issues when applying Eulerian-Eulerian models. Another issue that arises in this approach is the size of the control volume compared to the bubbles. If the bubbles are sufficiently smaller than the control volume, then the concept of interpenetrating continua and of each control volume containing a certain fraction of each phase is valid; if coalescence phenomena are significant, however, the larger bubbles formed may well grow bigger than the control volume. This issue also plays

a role in the parametrisation of the stress tensor (Jakobsen et al., 1997).

The Eulerian-Eulerian type of model is less computationally intensive than the Eulerian-Lagrangian, and therefore the only one that can resolve the large-scale flow structures in an industrial bubble columns (Delnoij et al., 1997).

1.2.5.1 Interfacial forces

The main shortcoming of the Eulerian-Eulerian two-fluid model is the lack of universally accepted forms of the closure laws, i.e. expressions for the interfacial forces. Many features of the flow field depend on them, in particular the gas phase distribution. Considerable debate exists over which forces should be taken into account, and how they should be modelled. At present, the model has to rely on experimental evidence.

Drag force. The drag force is the dominant interfacial force appearing in the momentum equations. For this reason, most simulation studies in the literature neglect the other forces (virtual mass, lift etc.). In modelling it, the first issue to be resolved is whether bubble-bubble interactions should be taken into account. These interactions are not only through collisions but also due to interference of their flow fields. If the gas fraction is low, as is the case in the current study, then these interactions can be considered negligible (Jakobsen et al., 1997) and the equation for the drag force on isolated bubbles may be used. For a single bubble this equation takes the form:

$$F^*_{D,j} = C_D \frac{\pi}{4} d^2 \frac{1}{2} \rho_L \left(u_{G,j} - u_{L,j}\right) \left|u_{G,j} - u_{L,j}\right| \tag{1.15}$$

Thus, the drag force per unit volume of the bubble swarm will be given by:

$$F_{D,j} = C_D \frac{3}{4d} \varepsilon_G \rho_L \left(u_{G,j} - u_{L,j}\right) \left|u_{G,j} - u_{L,j}\right| \tag{1.16}$$

The main problem in this approach is the estimation of the drag coefficient C_D. Evidence suggests that it is a complex function of the Reynolds (N_{Re}), Eötvos (Eo) and Morton (M) numbers (Jakobsen et al., 1997). Furthermore, it is likely to be affected by the presence of other bubbles (Thakre and Joshi, 1999). Only in the case of creeping flow, when just the form drag is accountable, an analytical solution is possible, resulting in the well-known Stokes law. For other cases, a variety of

correlations for calculating C_D have been proposed, and some of them are presented on table 1.

An alternative approach is to assume that the bubbles have reached terminal velocity. In this case, the force balance requires that vertical forces (drag, weight and buoyancy) should sum up to zero.

$$F_{D,j} = W - B \Rightarrow C_D \frac{\pi}{4} d^2 \frac{1}{2} \rho_L \left(u_{G,j} - u_{L,j}\right) \left|u_{G,j} - u_{L,j}\right| = V \left(\rho_G - \rho_L\right) g \quad (1.17)$$

To allow for the fact that the slip velocity is bound to have a range rather than a unique value, the following correction can be applied, provided that an estimate of the average slip velocity is known:

$$F_{D,j} = \varepsilon_G \left(\rho_{G,j} - \rho_{LG,j}\right) g \frac{u_{G,j} - u_{L,j}}{\overline{u_{G,j} - u_{L,j}}} \quad (1.18)$$

This approach has been adopted by Schwarz and Turner (1988), Joshi et al. (1998) and Sokolichin and Eigenberger (1994, 1999). Using a common slip velocity of 0.2 m/s, $\rho_L = 1000$ kg/l and $\rho_L >> \rho_G$ the equation reduces to a constant:

$$F_{D,j} = 4.9 \cdot 10^4 \varepsilon_G \left(u_{G,j} - u_{L,j}\right) \quad (1.19)$$

This gives a constant terminal velocity for all bubbles, which compares quite well with experimental data. Yet this approach is unrealistic, as the slip velocity is not constant and depends on the particular gas-liquid system (Thakre and Joshi, 1999). In some cases, though, it has been reported to give better solution than the more complex methods (Jakobsen et al., 1997).

Another notable contribution to the estimation of drag force is the drift flux model of Wallis (1969). This can be used to yield the terminal rise velocity and subsequently the drag force, making use of experimental measurements of the gas void fraction. Thakre and Joshi (1999) presented such a methodology, where the drift flux model was expanded to account for the radial profile of liquid velocity.

Virtual or Added Mass force. When a bubble is accelerated in the liquid, the surrounding liquid is accelerated as well. This results in an additional force, proportional to the relative accelerations of the phases:

$$F_{a,j} = -\varepsilon_g C_a \rho_l \frac{D}{Dt} \left(u_{g,j} - u_{l,j}\right) \quad (1.20)$$

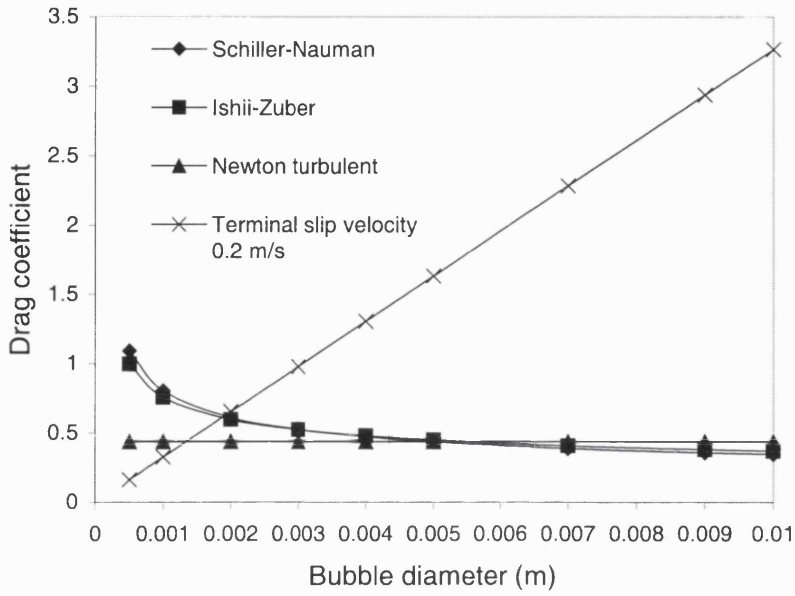

Figure 1.1: Values of the drag coefficient computed by various correlations.

C_D	Range	Named after:
$24/N_{Re}$	$0 < N_{Re} < 0.2$	Stokes
$24/N_{Re}(1 + 0.15N_{Re}^{0.687})$	$0 < N_{Re} < 500 - 1000$	Schiller and Nauman
$24/N_{Re}(1 + 0.1N_{Re}^{0.75})$	>>	Ishii and Zuber
$24/N_{Re} + 5.48N_{Re}^{0.573}) + 0.36$	>>	Ihme et al.
0.44	$500 - 1000 < N_{Re} < 1 - 2 \cdot 10^5$	Newton
$2/3Eo^{1/2}$	>>, distorted bubbles	Ishii and Zuber
$\frac{4}{3}\frac{gd}{u_T^2}\frac{\Delta\rho}{\rho}, u_T^2 = \frac{\mu}{\rho d}M^{-0.149}(J - 0.857)$	>>, ellipsoidal bubbles	Grace
8/3	spherical cap bubbles	Grace
$0.622/((1/Eo) + 0.235))$	500¡ $N_{Re} < 5000, ellips.$	Johansen and Boysan

Table 1.1: Some correlations for calculation of the drag coefficient.

The coefficient C_a has to be determined experimentally. According to Jakobsen et al. (1997), a value of 0.5 can be theoretically justified for rigid particles, but it is likely to decrease with increasing gas fractions. Sokolichin and Eigenberger (1994) report that 0.25 is a good value for bubbles while 0.5 is more suitable for solid particles.

Lift or Magnus force. This rotational force is exerted on rigid particles moving in a nonuniform flow field. It is given as the curl of the continuous phase velocity:

$$F_{l,j} = \varepsilon_g C_l \left(u_{g,j} - u_{l,j}\right) \times \left(\frac{\partial}{\partial x_j} \times u_{l,j}\right) \tag{1.21}$$

According to Svendsen (1992), this could account for the effect of bubbles accumulating in the centre in uniformly aerated bubble columns. The role of this force, though, was questioned by Sokolichin and Eigenberger (1994).

Other forces. Another force, not usually mentioned in most literature simulation studies, is the wall lubrication force. It exists in a region less than 5 bubble diameters from the wall, meaning that it can be seen only in sufficient fine grids. Considering, though, the fact that the control volume has to be substantially larger than the bubbles for the Eulerian-Eulerian model to be reliable, this force will probably be negligible. The Saffman force is another lateral force; Jakobsen et al. (1997) claimed that it is included in the formula for the Magnus force. Thakre and Joshi (1999) also followed this concept.

1.2.5.2 Bubble size distribution

Bubble size distribution has been studied by, among others, Kocamustafaogullari and Ishii (1983, 1995) who introduced the concept of "interfacial area concentration", and Bauer and Eigenberger (1999) who used a modified approach of "bubble number density". The later consists of a number balance for the total number of bubbles, accounting for its alteration by coalescence or redispersion phenomena. Once the number of bubbles has been determined, a mean diameter is applied to calculate the interfacial area. For a detailed estimation of the surface area, though, the most complete approach would be the population balance, which will be considered in detail later.

A population balance model needs to be supplied with functions describing bub-

ble coalescence and break-up kinetics, as well as with the initial distribution of the bubbles that have just emerged from the sparger. Though semi-empirical models of the former have appeared (e.g. Prince and Blanch, 1990; Luo and Svendsen, 1996) they are by no means definitive. Some researchers such as Tan and Harris (1986) and Geary and Rice (1991) have attempted to predict the bubble size as a function of the sparger geometry, but they consider only single holes and quiescent liquids. In a multi-hole sparger, the turbulent conditions arising from the bubble swarm would not allow for a single bubble size. In very high gas flowrates (heterogeneous flow) the bubble size may be dominated by inter-particle effects, but in homogeneous flow the initial distribution that arises from the sparger may prevail. Given the considerable uncertainty in our knowledge of both kinetics and initial conditions, the extra effort required for the solution of a population balance (which introduces a new independent variable associated with the bubble size, thus increasing the dimensionality of the problem) is probably not justified at present.

1.2.5.3 Turbulence

Turbulence is one of the major unresolved issues in fluid mechanics, and the state of the art in multi-phase flows is even more vague than in single-phase. Although it is generally accepted that the Navier-Stokes equations give an accurate account of fluid flow, even when turbulence is present, the small scale and high frequency of turbulent eddies mean that a numerical solution aiming to predict them would require extremely small spacial and temporal steps, which are out of reach of the current computational resources; moreover, turbulent flows are very sensitive to boundary conditions (a characteristic of chaotic phenomena), which can be neither determined with accuracy nor controlled. Therefore phenomenological models of turbulence have been developed, which attempt to predict probabilistic aspects of the flow. The following classes of models can be identified:

Direct Numerical Simulation (DNS): The direct solution of the Navier-Stokes equations, at very small scales and high frequencies, in order to resolve all turbulence scales. At present limited to low Reynolds numbers and simplified geometries, and unlikely to be of practical use in the near future.

Large Eddy Simulation (LES): A spatial filtering technique that resolves the larger eddies but uses models for the effect of the smaller (subgrid) scale turbulence.

Time-averaging techniques: Time-averaging results in equations that describe the mean properties of the flow, which is usually all that the engineer needs, but gives rise to a closure problem, as further terms are generated (products of fluctuations). Several closure approaches have been proposed, the most common ones being those based on the eddy-viscosity concept. The $k - \varepsilon$ model (Launder and Spalding, 1972) is currently the standard in industrial CFD - it employs two further differential equations to determine the effect of the unclosed terms on the mean properties. Other options are second-moment closures (Reynolds stress models) which employ equations for the unclosed terms themselves.

Probability density methods (PDF) : These methods attempt to determine the entire distribution of the turbulent properties, usually at single points in space and time. They are especially advantageous for reactive flows.

These approaches have been applied mainly to single-phase flows. According to Jakobsen et al. (1997), large eddy simulation should lend itself better to extrapolation into multi-phase flows, as its concepts are closer to the mechanisms prevailing in multi-phase flow. Such an attempt has not been done yet, because it is too expensive to compute. Therefore, most studies on gas-liquid flows either assume laminar flow, or use one of the single-phase turbulence models for the continuous (liquid) phase, usually the k-ε, either intact or with slight modifications. Sokolichin and Eigenberger (1994) defended the laminar approach, on the basis that the k-ε model cannot conceptually be extended to gas-liquid flow, and that it would dampen the transient character of the flow structure. In their results, the laminar simulation was shown to predict some vortices that were missed from a standard k-ε implementation. Pfleger, Gomes et al. (1999) conducted a numerical/experimental study focusing on this point: they concluded that, though the k-ε does dampen the chaotic behaviour exhibited by the laminar model, converging to a periodic oscillation instead, it represented better the experimental data on radial velocity. Two ways in

which the k-ε model can be modified to account for the particularities of gas-liquid flow are to include turbulence caused by bubble wakes (bubble-induced turbulence) and turbulent dispersion of the gas phase.

In general, large bubbles are expected to enhance turbulence, while small bubbles are thought to suppress it (Boisson and Malin, 1996). Bubble-induced turbulence was modelled by Sato (1981). It introduces a new term to the effective viscosity of the liquid phase, in addition to the contributions of the laminar and eddy viscosity terms:

$$\mu_{l,eff} = \mu_{l,lam} + \mu_{l,t} + \mu_{l,B} \tag{1.22}$$

Though it is generally accepted that bubble-induced turbulence plays a certain role, this approach to modelling it is still debatable. Pfleger, Gomes et al. (1999) did not include it in their simulations; Boisson and Malin (1996) and Svendsen et al. (1992) did.

Turbulent dispersion of the gas phase can have an effect on the radial velocity and fraction profiles. It is usually modelled via a transport equation for the gas fraction, including diffusion through eddy diffusivity. Again, this approach is by no means established. Pfleger, Gome et al. (1999) tested it in conjunction with the k-ε versus a plain k-ε model and found the compliance with experimental data to deteriorate.

1.3 The population balance

1.3.1 Origin and formulation of the PBE

During the last century, the population balance equation (PBE) has been established as a rigorous way of describing a variety of particulate systems such as cell populations, droplet and bubble swarms, granular solids, crystals, polymers, coagulating aerosols and even astrophysical systems. In all of these situations, the aim is to predict the evolution of the distribution of one or more properties that characterise the individual particulate entities, and the dynamic PBE is in essence a number balance equation describing this evolution (fig. 1.2).

The PBE originates in the work of Smoluchowski (1917), whose discrete equation for pure aggregation stemmed out of his study of Brownian motion. The equation

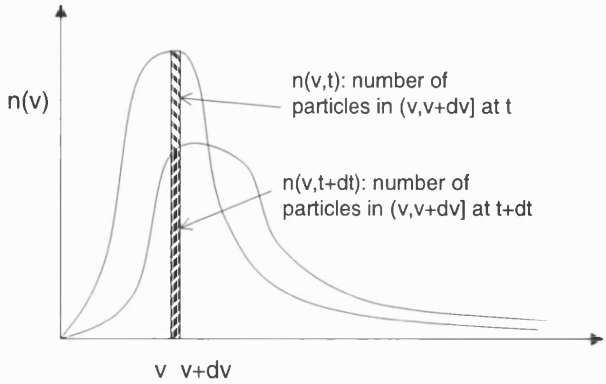

Figure 1.2: Evolution of the population density.

was later cast in a continuous form and extended to include the process of breakage. Thus, in the 1960's Hulburt and Katz (1964) and Randolph and Larson (1988) formulated the PBE for nucleation and growth phenomena. Subsequent milestones such as Drake's comprehensive survey (1972), the analysis of Ramkrishna and Borwanker (1973) and Ramkrishna's review (1985) did much to rationalise the PBE and establish it as a mainstream method for the analysis of particulate systems. In its most general form, the continuous PBE is a dynamic transport equation that describes the temporal evolution of population density as a result of four particulate mechanisms: nucleation, growth, aggregation and breakage, as well as transport due to the flow field:

$$\frac{\partial n(x_i,t)}{\partial t} = -\frac{\partial(u_i \cdot n(x_i,t))}{\partial x_i} + D\frac{\partial^2 n(x_i,t)}{\partial x_i^2} + \\ + B_{nuc} + B_{agg} - D_{agg} + B_{br} - D_{br} \quad (1.23)$$

Written thus, the PBE refers to a state space comprised of time, physical space, and an additional number of dimensions that represent the characteristic properties of the particles whose distribution we wish to calculate, often referred to as internal coordinates[2]. The gradient term in eq. 1.23, apart from input/output due to convective flow (differentiation in physical space), represents growth (differentiation in the particle size domain) or any other mechanism that operates on the population density in a convective way. as well as input/output due to convective flow. Similarly the diffusion term may represent both diffusion in physical space and ef-

[2]Both spatial and internal coordinates in eq. 1.23 are denoted by x_i.

fective diffusion of the population density, though the latter has usually no physical interpretation.

Unfortunately, the PBE yields analytical solutions for only a few special cases. Numerical solution of the PBE remains a considerable challenge, however, largely due to the integral form of the aggregation and breakage terms, and most approaches seem to have been tailored to specific applications (Rawlings et al., 1993). The aim of this work is to develop a numerical scheme that is capable of dealing effectively with an arbitrary particulate process, rather than being derived to match the needs of individual problems, while maintaining simplicity and speed. Particular emphasis will be put on stability and proper conservation of the moments.

1.3.2 Numerical solution of the PBE

Solutions to the discrete Smoluchowski equation are out of the scope of this paper; those interested may consult Drake's review (1972). We will only mention that its exact solution can be found by direct numerical simulation of every individual particle - a computationally expensive approach, feasible only for relatively small populations. Monte Carlo simulations are another class of methods that will not be considered here; they are generally more efficient for multivariate problems, where solutions to the continuous version become too involved (Ramkrishna, 1985).

Analytical solutions of the continuous population balance have been derived only for a few idealised cases (see e.g. Scott, 1968, Drake, 1972, Ramabhadran et al., 1976, or Hostomsky, 1986); they are very useful, though, for benchmarking numerical solutions. One of the oldest and most widespread methods of solving it is the method of moments (Hulburt and Katz, 1964; Randolph and Larson, 1988), which is based on the transformation:

$$L^k(f(n,v)) = \int_0^\infty v^k f(n(v),v)dv \tag{1.24}$$

By applying this transformation, it is sought to generate terms of the form:

$$m_{<k>} = \int_0^\infty v^k n(v)dv \tag{1.25}$$

which is identified as the k-th moment of the distribution. The most important moments are the zeroth, which expresses the total number of particles, and the first,

which is proportional to the total mass of the system (when particle volume is used as the independent variable). Thus the PBE is transformed from a partial integro-differential equation in terms of population density into a differential equation in terms of the k-th moment. Although this method does provide the exact solution for the moments, it does not retrieve the entire distribution. In some special cases the distribution may be retrieved from the moments if its form is known *ad hoc*, but generally this process is unstable (Randolph and Larson, 1988). Moreover, not all formulations of the PBE can be reduced to moment equations, since some of the terms resulting from the transformation may not exhibit the form of eq. (1.24) - in fact, this is usually the case with formulations that include agglomeration and breakage. Thus the use of moments is limited to special cases, usually involving nucleation and growth only.

- Methods that retrieve the distribution by approximating the solution with a series of trial functions, whose coefficients are to be determined so that their sum will satisfy the PBE. When these functions are global, the scheme is a variation of the method of weighted residuals; in essence, the method of moments also belongs to this class (Ramkrishna, 1985). When the functions are local we have a variation of the finite element method.

- Methods that discretise the spectrum of the independent variable into a number of intervals and subsequently use the mean-value theorem to transform the continuous PBE into a series of equations in terms of either number of particles – termed the M-I approach - or average population density in each class – the M-II approach. These methods are often referred to as "Discretised Population Balances" or DPBs.

Weighted residual methods with global functions were among the first to be tried, and one may consult Ramkrishna (1985) for a review of early attempts. It was soon realised, however, that global functions cannot capture the features of an arbitrary-shaped distribution, especially if it exhibits sharp changes and discontinuities. If a-priori knowledge of the shape of the resulting distribution is available, the trial functions can be tailored to accommodate it; in that case, the method converges and may even be computationally attractive. If generality and flexibility are

required, however, the trial functions must clearly be local. Finite element methods approximate the solution with piecewise low order polynomials that are only locally non-zero, and are thus flexible and capable of capturing highly irregular solutions. The first notable attempt to solve the PBE with finite elements was carried out by Gelbard and Seinfeld (1978), who considered both orthogonal collocation and spline collocation for the solution of a dynamic PBE with nucleation, growth and aggregation.

Meanwhile, an alternative approach emerged: the discretisation of the domain of the independent variable. As pointed out by Gelbard and Seinfeld (1978) this approach, if straightforward, leads to severe errors unless a uniform grid is employed - unrealistic for anything but systems exhibiting a very small size range. For processes such as crystallisation or coagulation of aerosols, where large particles can be created very soon via aggregation, a non-uniform, preferably exponential grid must be employed. Bleck (1970) pointed out that the computational effort can be severely reduced if the grid assumes a geometric progression of (2^j) - this way, the number of combinations of particles that must be considered to evaluate the aggregation terms is substantially reduced.

The next challenge for advocates of DPBs was to find a way of conserving the moments in such a grid. Batterham et al. (1981) modified this approach so that it conserves exactly the first moment with respect to volume (i.e. the total mass) by dividing particles that were created at non-nodal points into fragments whose volume would lie at the nearby nodal points. This artificial way of conserving mass, however, introduced errors in all the other moments, as well as in the predicted population density. The most established method in this series, proposed by Hounslow et al. (1988), is capable of conserving exactly both total mass and number of particles, as well as accommodating simultaneous growth and aggregation. All these methods are, however, constrained with respect to the choice of grid, as they are based on exploiting the properties of the geometric progression. Recent works by Litster et al. (1995) and Kumar and Ramkrishna (1996a,b,c) have succeeded in extending the concept of DPBs to finer grids; the latter also offers the advantage of conserving any two chosen moments. Also notable are the "method of classes" due to Marchal et al. (1988), which treats aggregation as a chemical reaction, and the work of Hill

and Ng (1995) on breakage processes, respectively.

After Gelbard and Seinfeld's pioneering work (1978), finite element and other function approximation methods seemed to attract fewer attention than DPBs; even the above authors made their own contribution to the DPBs (Gelbard et al., 1980). The main reasons for this trend are assumed to be the excessive, for that time, computational demands raised by these methods as well as the complexity of their implementation, in particular when applied to aggregation and breakage problems. Steemson and White (1988) used spline collocation to solve steady state problems of nucleation, growth and size dispersion; Eyre et al. (1988) also employed spline collocation, complemented with an adaptive grid. The recent advances in computational power, however, have brought finite element methods back to the forefront. Nicmanis and Hounslow (1998) solved the steady-state PBE with nucleation, growth, aggregation and breakage using collocation and Galerkin finite elements with cubic Lagrangian trial functions. The approach of Liu and Cameron (2001) to the dynamic PBE is also a function approximation method that uses wavelets to represent the solution, and focuses on accurate prediction of discontinuities. Recent reviews have been contributed by Rawlings et al. (1993), Costoglou and Karabelas (1994), Vanni (2000) and Lee (2001).

1.4 Crystallisation and precipitation

1.4.1 Overview

Crystallisation is one of the oldest, and yet most widely employed, unit operations. It is employed for the production of a large variety of crystalline products such as inorganic salts, chemicals and pharmaceuticals, as well as for separation and purification purposes. Comprehensive reviews of research in this field have been contributed by Nyvlt, Söhnel, Matuchova and Broul (1985), Randolph and Larson (1988), Söhnel and Garside(1992), Rawlings, Miller and Witkowski (1993), Mullin (2001), Mersmann et al. (2001), and Jones (2002).

According to Mullin (2001), the following crystallisation techniques can be distinguished:

- Solution crystallisation, the formation of a crystalline phase in a supersatu-

rated solution.

- Precipitation, fast crystallisation often driven by chemical reactions.
- Melt crystallisation, the separation of a multicomponent liquid mixture by cooling close to freezing point.
- Recrystallisation, the dissolution of an already crystallised material into fresh solvent and subsequent crystallisation by cooling, to remove soluble impurities.
- Sublimation, the production of a crystalline phase by direct condensation of a supersaturated vapour.

The driving force of crystallisation is supersaturation, which determines both its thermodynamics and kinetics. A solution can exist in one of three states: stable (no supersaturation) where crystallisation is impossible, metastable (low supersaturation) where crystallisation is unlikely to occur spontaneously but can easily be induced by a seed, and labile, where spontaneous crystallisation is probable but not inevitable (Mullin, 2001). Supersaturation also affects the rates at which crystallisation mechanisms proceed. A number of ways to express supersaturation have been proposed, such as the following: (Nyvlt et al., 1985)

$$\Delta C = C - C^* \tag{1.26}$$

$$S = \frac{C}{C^*} \tag{1.27}$$

$$\sigma = \frac{\Delta C}{C^*} = S - 1 \tag{1.28}$$

Here C^* is the saturation concentration. A solution can become supersaturated in several ways, such as cooling to reduce the solubility, solvent evaporation to increase solute concentration, and addition of a co-solvent (salting out). Of particular interest in this work is reaction crystallisation, or precipitation, where supersaturation is generated by a chemical reaction between two soluble reactants that produces a sparingly soluble material. Precipitation is distinguished from other crystallisation techniques, in that its analysis requires a combination of crystallisation and reaction engineering principles, and a fair amount of literature is dedicated to it (e.g. David and Klein, in Mersmann, 2001; Söhnel and Garside, 1992). Precipitation is charac-

terised by high degree of supersaturation and fast kinetics that usually lead to the production of smaller crystals.

Apart from the usual unit operation objectives such as yield and efficiency, the analysis of crystallisation processes is also concerned with the prediction of the physical properties of the crystal product, such as its morphology and CSD. These properties are determined by the relative strength and interaction of several fundamental crystallisation mechanisms, such as nucleation, crystal growth and agglomeration. The ability to predict and control the physical properties of the product is a major advantage to the designer, since it grants the potential for product engineering. Thus the effort to establish kinetic models of crystallisation mechanisms has been at the core of crystallisation research since its very onset, yet definitive expressions still eludes the efforts of workers in the field. What follows is a brief review of the theories available to date, with emphasis on those that enable the formulation of kinetic models.

1.4.2 Nucleation

Though from a thermodynamic point of view the crystalline state is stable in a supersaturated solution, it is the nucleation kinetics that determine whether and at what rate crystals will be formed. Nucleation is a step of key importance to the outcome of the crystallisation process, as it can have a profound effect on both crystal morphology and CSD. However, it is also one of the least understood due to the difficulties encountered in isolating it experimentally from other phenomena, as well as the variety of fundamentally different mechanisms that are collectively called "nucleation" (Rawlings et al., 1993).

It is usually classified into: (Mullin, 2001)

- Primary: Formation of crystal nuclei directly from the dissolved species, driven by supersaturation. Further divided in:
  - Homogeneous: Spontaneous formation of nuclei.
  - Heterogeneous: Induced by foreign particles.
- Secondary: Generation of nuclei from existing crystals, e.g. by attrition.

Of these only homogeneous nucleation has been approached by way of rigorous physical and mathematical reasoning, in the form of the so-called classical nucleation theory. This are based on thermodynamic concepts, such as the free energy required for nuclei formation. The free energy is maximum for a certain size of nuclei, called critical size, and subsequently decreases with size. The nucleation rate is estimated from the rate at which arrangements of molecules ("embryos") exceed the critical size, and is expressed by an Arrhenius-type equation. Very high supersaturation is required to overcome the energy maximum and cause spontaneous nucleation. For this reason, homogeneous nucleation is prevalent only in precipitation, where very high supersaturation levels are common.

The rate laws resulting from classical nucleation theory have seldom been applied for scale-up of industrial processes because, apart from the assumptions inherent in them, homogeneous nucleation is not a common phenomenon in industrial processes. Carefully conducted experiments in very clean equipment are required to ensure that nucleation occurs solely via the homogeneous mechanism. On the other hand the energy barrier to be overcome is much lower in heterogeneous and secondary nucleation, and actual crystallisers contain lots of crystals and impurities. For this reason, they will not be discussed here; for a review of them one may consult, among others, Nielsen (1964) or Mullin (2001).

Foreign surfaces and particles act as a catalyst in the nucleation process. The supersaturation levels required for heterogeneous nucleation are much lower than those for the homogeneous mechanism, but still too high for it to dominate in processes other than precipitation. Furthermore, primary nucleation is very difficult to predict and scale-up, because the very high levels of supersaturation it requires, when generated, cause a burst of crystals that is very difficult to control (Rawlings et al., 1993). Since heterogeneous nucleation is not easy to formalise, empirical power-law functions of supersaturation are usually employed to express actual nucleation rates:

$$B_0 = k_n S^n \tag{1.29}$$

In most industrial crystallisation processes apart from precipitation, secondary nucleation is the prevailing source of nuclei (Rawlings et al., 1993). In contrast with primary nucleation, secondary nucleation is much easier to control. Comprehensive reviews of the underlying mechanisms have been contributed by Botsaris (1968),

Garside and Davey (1980), Larson (1984) and Mersmann, Sangl, Kind and Pohlish (1988). Secondary nucleation is also modelled with empirical power law rules, that associate its rate with the two main driving forces, supersaturation and the total crystal mass:

$$B_0 = k_n S^{n1} M_c^{n2} \tag{1.30}$$

Nucleation is affected by several environmental factors and hydrodynamic factors. Agitation generally enhances nucleation, and may cause solutions were otherwise stable to nucleate. However, very intense agitation has been found to reduce nucleation in some cases, possibly by disrupting the molecular clusters (Mullin, 2001). As a result, scale-up of nucleation rates is mostly empirical, based on experiments conducted in conditions as close as possible to the actual ones.

1.4.3 Crystal growth

Crystal growth is a broad field on its own, and comprehensive surveys may be found in Garside (1984) and Mullin (2001). From a kinetic point of view, crystal growth may be viewed as a two-step process: diffusion of molecules or ions to the crystal surface, and incorporation into the lattice, though the reality is likely to be more complicated (Mullin, 2001). Most of the chemical engineering literature treats the diffusion step with an empirical mass-transfer coefficient, while alternative approaches include treating the diffusion through a stagnant film, or applying some form of boundary layer theory (Rawlings et al., 1993).

The incorporation of new molecules into the lattice has been phenomenologically described by several theories, most of which can be found in Mullin (2001). Early theories were based on thermodynamic arguments, but predicted lower growth rates than those experimentally observed. The BCF theory due to Burton, Carberra and Frank (1951) overcame this problem by assuming that growth occurs at a screw dislocation. Birth and spread models (O'Hara and Reid, 1973; van der Eerden, Bennema and Cherepanova, 1978) postulates that growth develops from surface nucleation that can occur at edges, corners or on faces of a crystal.

For engineering purposes, however, the semiempirical power law has become the standard representation of the growth rate (Rawlings et al., 1993), with an

Arrhenius-type expression to denote temperature dependence:

$$G_l = k_g e^{\frac{-E_g}{RT}} S^g \tag{1.31}$$

where G_l is growth with respect to a linear dimension (such as diameter, or radius) and S is a measure of supersaturation. The fact that size does not enter the expression is commonly referred to as McCabe's ΔL law; this is not always the case, however. Deviations from this are interpreted either as size-dependent growth or growth rate dispersion. Size dependent growth may be described by empirical functions such the following (Randolph and Larson, 1988):

$$G = G_0(1 + \gamma l)^\beta \tag{1.32}$$

Growth rate dispersion is interpreted as fluctuations in growth rate due to reasons such as variations in crystal structure, hydrodynamic environment, local supersaturation etc. (Randolph and White, 1977) and is usually modelled with the aid of a diffusion-like term.

1.4.4 Agglomeration

The importance of agglomeration in crystallisation processes is becoming more and more evident, a fact reflected in the recent literature. Agglomeration can have a profound effect in the CSD, and may determine the physical properties of the end product, as well as its potential for separation; depending on the application, either its promotion or prevention may be desirable. Its mathematical analysis is faced with severe hindrances, however.

Terms such as "aggregation", "flocculation", "coagulation" and "agglomeration" have been applied interchangeably to describe the tendency of particles to cluster together (Mullin, 2001), a phenomenon that appears in many different fields such as the study of aerosols and colloidal systems. In crystallisation, however, the formation of such clusters involves two clearly distinct phenomena: the clustering of crystals due to collisions and interparticle attractive forces, and the cementation of the cluster into a single particle by way of solid bridges, due to crystal growth in its interior. Therefore a clear, formal distinction between the physical states of the clusters during the various stages of the process is required. The most clear formalism is

probably due to ASTM (1985), also used by Hartel et al. (1986), and it will be adopted in this study. According to it, three kinds of clusters must be distingushed:

- Flocculates: groups of particles held together by interparticle forces.
- Aggregates: Tighter-bound clusters, existing during the transition from flocculates to agglomerates.
- Agglomerates: Fully cemented clusters, unified by crystalline material grown in the interior.

Some crystal groups resulting from malgrowth phenomena (e.g. twins or dendrites) may have the same appearence as agglomerates under microscopic examination, and are called primary agglomerates (Mersmann, 2001). This is a fundamentally different phenomenon, however, and in this section we shall refer only to agglomeration that results from inter-particle clustering.

The first stage of agglomeration, i.e. the formation of flocculates due to collisions and interparticle attraction, is akin to similar phenomena occurring in colloids and aerosols. Thus the starting point for the formulation of a mathematical description of agglomeration are the theories developed in those fields, most importantly the pioneering work of Smoluchowski (1917; see Chandrasekhar, 1945, for an account of Smoluchowski's work in English). The second step, however, i.e. the transformation into an agglomerate due to inter-cluster growth, is unique to crystallisation processes and has a dominating influence. As a result, theories that are directly transposed from aerosol collection to crystallisation often fail to describe crystal agglomeration. On the other hand, the formulation of models that properly account for both steps faces severe technical obstacles due to the different nature of the phenomena involved in each stage, as well as the lack of appropriate experimental data. To gain insight into the different stages of an agglomeration process through experiments, let alone isolate them, is a formidable task.

Many factors influence an agglomeration process (see Mersmann, 2001, for a detailed list), but the most important ones can be grouped under the following:

- Hydrodynamics, which determine the motion of particles in a suspension, and therefore the rate and efficiency of collisions.

- Interparticle forces, which govern the initial flocculation stage.
- Solute concentration (supersaturation), which induce growth of inter-cluster solid material, thus transforming the aggregate into an agglomerate.

A brief review of the knowledge available so far in each one of these fields follows.

1.4.4.1 Hydrodynamics

Any clustering process is second order, induced by collisions of particles:

$$-\frac{dN}{dt} = \beta N^2 \tag{1.33}$$

Thus the first thing to consider is the frequency of collisions, which must be derived from hydrodynamic arguments. The first theoretical study of aggregation problems is probably due to Smoluchowski (1916). He considered the problem of perikinetic aggregatetion, i.e. the clustering of particles due to thermal (Brownian) motion, and approached it as a diffusion process. Assuming ideal spheres, he arrived at the discrete population balance equation:

$$\frac{dn(v)}{dt} = \frac{1}{2}\sum_{v'=0}^{v'=v} k_a(v', v-v')n(v')n(v-v')dv' - \\ - n(v)\sum_{v'=0}^{\infty} k_a(v', v)n(v')dv' \tag{1.34}$$

The kernel he derived for perikinetic aggregation (see Chandrasekhar, 1945, or Drake, 1972, for the derivation) is:

$$k_a(v', v-v') = \frac{2kT}{3\mu}\left(\frac{(r(v')+r(v-v'))^2}{r(v')r(v-v')}\right) \tag{1.35}$$

By using the approximation:

$$\frac{(r(v')+r(v-v'))^2}{r(v')r(v-v')} \approx 4$$

Smoluchowski was able to reduce eq. 1.35 to the size-independent kernel, and obtain an analytical solution for the infinite set of eqs. 1.34. This approximation is conceptually valid for aggregation of a system initially monodisperse, for a short amount of time. However, the size-independent has been used extensively in the crystallisation literature, due to its simplicity and the mutual-cancellation of other effects.

Perikinetic motion is prevalent only for very small particles ($< 1\mu m$). When collisions are induced by hydrodynamic forces, aggregation is termed orthokinetic. Smoluchowski (1917) was also the first to derive a hydrodynamic kernel for orthokinetic aggregation, considering the case of shear-induced collisions:

$$k_a(v', v - v') = \frac{4}{3}\dot{\gamma}\,(v'^{1/3} + (v - v')^{1/3})^3 \tag{1.36}$$

Orthokinetic is likely to be the dominant aggregation mechanism in crystallisation and precipitation processes. Two mechanistic kernels have been proposed to describe orthokinetic aggregation in a turbulent flow field. The first one (diffusion mechanism) assumes that the aggregation rate is determined by number of encounters of particles entrained by turbulent eddies (Drake, 1972):

$$k_a(v', v - v') = 0.31\left(\frac{\epsilon}{\nu}\right)^{1/2}(v'^{1/3} + (v - v')^{1/3})^3 \tag{1.37}$$

The diffusion mechanism is dominant for fine particles ($< 10\mu m$). For larger particles the iniertial mechanism, has been proposed, according to which collision occur due to particles of different sizes moving at different velocities (Drake, 1972):

$$k_a(v', v - v') = 0.2\frac{\rho\epsilon^{3/4}}{\rho_0\nu^{5/4}}[v'^{1/3} + (v - v')^{1/3}]^2|v'^{1/3} + (v - v')^{1/3}|E_i \tag{1.38}$$

where ρ is the density of the particles, ρ_0 the density of the gas and E_i an efficiency factor. These kernels are, however, more applicable to the coagulation of aerosols. Aggregation can also be induced during gravitational settling, as particles of different sizes move at different velocities; the size dependency is similar to eq. 1.38. Several empirical kernels have also been proposed, such as the Thompson kernel; they may be found in Drake (1972) or Beckman and Farmer (1984).

The influence of hydrodynamics is likely to extend to other aspects of the agglomeration process than the collision frequency, however. It has been observed that agglomerates do not form during the induction time when crystals are very small (Mersmann, 2001), and do not exceed a maximum size. The former effect is accounted by most size-dependent kernels, but the latter seems to point to a disruption mechanism of the large aggregates, which is most likely hydrodynamic. Larger aggregates are subjected to higher stresses and are less likely to sustain them. So far only two hydrodynamic models of disruption have appeared, due to David et al.

(1991) and Mumtaz et al. (1997). The latter assumes a linear shear field, while the former employs turbulent quantities, but both rely on simplifying assumptions, as the real phenomenon is bound to be very complex.

1.4.4.2 Interparticle forces

The starting point of an agglomeration process is the formation of a flocculate, i.e. the clustering of two crystals during a collision due to interparticle forces. These arise from various interactions between both quantum-scale and macroscopic particles, but the latter are likely to dominate. One of the most established theories of interparticle forces in colloidal systems is the DLVO theory (Derjaguin and Landau, 1941; Verwey and Overbeek, 1948). The DLVO theory concentrates on two kinds of forces that dominate: Van der Waals and electrostatic. Several non-DLVO forces may also be accountable, such as solvation, structural and hydration forces (Mersmann, 2001).

Interparticle force theories are usually applied to study the aggregation of colloidal systems (e.g. Melis, Verduyn, Storti, Morbidelli and Baldyga, 1999), but have seldom been incorporated into crystal agglomeration rate models. The reason is that their role is confined to the initial stage of agglomeration, i.e. the formation of flocculates. The solid bridges that develop shortly afterwards give rise to lattice forces, which are much stronger, and it is probably those that dominate the overall agglomeration process (Bramley, Hounslow, Newman, Paterson and Pogessi, 1997).

1.4.4.3 Solute concentration

Completion of the second step in the agglomeration process, that is the cementation of the aggregate, requires crystal growth in certain spots in the interior of the cluster, called solid bridges or cementation sites. Growth requires the presence of supersaturation, and may be controlled by diffusion or surface integration. The experiments of Bramley et al. (1997) on precipitation of calcium oxalate showed that the agglomeration rate is better correlated with the limiting ion concentration, rather than the supersaturation. This implies that diffusion into the cementation cite is the more likely mechanism, at least for similar systems. Mumtaz et al. (1997) proposed a 2-D diffusion-reaction model of crystal growth in the cementation cite.

Collier et al. (1999) and Hounslow et al. (2001) have presented extensive experimental data on calcite and calcium oxalate that show aggregation and growth rates to be roughly proportional, all other things being equal. This supports the view that supersaturation must be reflected in the formulation of the aggregation kernel.

1.4.4.4 Agglomeration kinetics

Research in aggregation processes dates from Smoluchowski's pioneering work (1917), and has produced valuable insights in fields as diverse as aerosols, colloids etc. The application of this knowledge in the agglomeration of crystals, however, is still at a primitive stage. Among the first studies were those of Hartel et al. (1988) and Hartel and Randolph (1988) who studied the agglomeration of calcium oxalate, a system of medical importance (formation of urinary stones). They employed an MSMPR crystalliser and a Couette-Taylor reactor in series, in an attempt to isolate the agglomeration from the nucleation and growth phenomena.

As mentioned before, agglomeration of crystals is a two-step process. The first step involves the collision of particles (crystals or aggregates) that results in the formation of a flocculate. This process is akin to the coagulation (collection) of aerosol and colloidal systems, and theories originating in these fields can be inferred. This step is essentially a second-order process:

$$-\frac{dN}{dt} = \beta N^2 \tag{1.39}$$

where N is the total number of particles. The continuous population balance describing an aggregation process is:

$$\frac{dn(v)}{dt} = \frac{1}{2}\int_{v'=0}^{v'=v} k_a(v', v-v')n(v')n(v-v')dv' - \\ - n(v)\int_{v'=0}^{v'\to\infty} k_a(v', v)n(v')dv' \tag{1.40}$$

The simplest way to express the kernel, β, is as a size-dependent function that expresses the frequency of collisions, derived hydrodynamically or empirically:

$$\beta_a = k_a\beta(v, v') \tag{1.41}$$

Apart from frequency, the formation of the initial clusters (flocculates) also depends on the efficiency of collisions. Efficiency is generally defined as the ratio of the

number of collisions that lead to the formation of a flocculate to the total number of collisions, and depends on interparticle forces and hydrodynamic factors. Thus the kernel is often described as:

$$\beta_a = k_a b_{eff} \beta(v, v') \tag{1.42}$$

The second step is the growth of crystalline material between the clusters at spots called cementing sites. This step differentiates crystal agglomeration from other clustering processes, and few attempts at its mechanistic description have appeared so far. Most notable are the contributions of David et al. (1991) and Mumtaz et al. (1997).

David et al.(1991) associated this step with a "sticking probability" of the crystals (as opposed to the "encounter probability" term used by them to describe the first step), and proposed that it is related to the time required to build a solid bridge. Assuming growth of a cylindrical bridge between two spherical crystals, they arrived at an expression that shows the agglomeration kernel to be inversely proportional to the growth rate, as well as introducing a further size dependency:

$$t_{bridge} = \frac{d_s}{f_s G} \tag{1.43}$$

where d_s is the diameter of the smaller particle and f_s is the shape factor. They also assumed that the growth of a bridge may be interrupted if the two participating particles are dissociated by the turbulence, and attempted an estimation of the dissociation probability on the basis of turbulence parameters. The latter indicated that the time scale during which turbulent dissociation may occur is much smaller than the time scale required to build a bridge, and the authors concluded that agglomeration requires rapid growth of the bridge.

The model of David et al. (1991) does not consider the growth of the bridge in detail. Mumtaz et al. (1997) modelled the deposition of ions into a cementation site by means of a 2-D diffusion-reaction model. The solid bridge was approximated by two cones, building up from either sides of the site and meeting at a singular point. The interruption of the cementation process was related to the tensile stress applied on the bridge due to a linear shear field for a certain collision orientation, and the yield stress of the crystalline material.

The cementation of an aggregate is a complex phenomenon, however, and both approaches rely on simplifying assumptions. Though a couple of spheres may be a relevant model when considering the aggregation of two individual crystals, observed agglomerates usually comprise many individual crystals, which means that most aggregation events occur between crystals and aggregates or between aggregates themselves. An aggregate can be envisaged as a network of particles, linked with bridges (cementation cites) that were formed at different times. Each cementation site has therefore different strength and probability of rupturing, depending on its age of formation. Though the tensile forces applied on the bridge linking an isolated couple of spheres may be prone to the description of Mumtaz et al. (1997), the stresses to which the interior bridges are being subjected are probably very different. In a large aggregate the stress distribution on its surface is bound to be uneven, and this would subject the interior bridges to high shear forces; it may indeed be the cause that prevents the formation of very large agglomerates. The approach of David et al. (1991) also assumes that the aggregate does not change in volume until the solid bridge is completed. Furthermore, the hydrodynamic stresses to which is an aggregate is subjected during is life time are likely to be more complex than the linear shear field of Mumtaz et al., or the turbulent dissociation principle of David et al..

In the works of Mumtaz et al. (1997) and of Hounslow et al. (2001), the concept of aggregation efficency is extended to include the cementation step. In that context an effective collision is one that survives all obstacles until the formation of a completed agglomerate. Based on an exhaustive set of experimental data on stirred tanks, encompassing two systems (calcium carbonate and calcium oxalate) and a variety of different conditions, Hounslow et al.(2001) proposed a correlation for the overall efficiency of the agglomeration process. His correlation shows a linear dependency on growth and a strong hydrodynamic effect. Particle size does not enter directly, but only the mean size in the whole vessel has an influence. As a result, the predicted CSDs are similar in shape to those arising from the size independent model.

Inherent in these approaches (that employ the concept of efficiency for the whole process) is the notion that agglomeration is a single event that either succeeds or not

in creating a stable, cemented agglomerate. This notion, imported from the aerosol literature, is however inconsistent with one particular aspect of the agglomeration process. A cluster resulting from an initially succesful collision may exist, for a considerable amount of time, in a state where its bridges are partly cemented to a different extent each. During that time it may participate in collisions, and thus affect the population balance by adding up to the population density at its size class, even though the bridge initially responsible for it may eventually rupture and split it into fragments - different from the initial ones. To take this into account one needs to consider agglomeration as a the overall result of two separate events: aggregation and disruption. Such a process would be described by the following population balance equation:

$$\begin{aligned}\frac{dn(v)}{dt} = \frac{1}{2}\int_{v'=0}^{v'=v} k_a(v', v-v')n(v')n(v-v')dv' - \\ - n(v)\int_{v'=0}^{v'\to\infty} k_a(v', v)n(v')dv' \\ + \int_{v}^{\infty} \rho(v, v')S(v')n(v', t)dv' - S(v)n(v)\end{aligned} \tag{1.44}$$

This approach introduces two further issues: determining the disruption kernel and the size distribution of the fragments resulting from a disruption event. Hartel and Randolph (1988), who adopted this approach, assumed that disruption is size-dependent and that each event results in the generation of two equal-sized particles. There seems to be no theoretical justification for the latter.

The aggregation-disruption approach was also explored by Wachi and Jones (1991; 1995). They assumed that disruption is carried out in an attritionary manner, i.e. every breakage event involves the removal of a single crystal. To attain this, they introduced the crystal number S (i.e. the number of crystals in an aggregate) in the population balance:

$$\begin{aligned}\frac{dn(v,S)}{dt} = \frac{1}{2}\int_{S=1}^{\infty}\int_{v'=0}^{v'=v} k_a n(v', S)n(v-v', S)\, dS\, dv' - \\ - n(v)\int_{S=1}^{\infty}\int_{v'=0}^{\infty} k_a n(v', S)\, dS\, dv' \\ + \int_{S=1}^{\infty} \rho(v, v', S)S(v', S)n(v', S)dS' - S(v)n(v)\end{aligned} \tag{1.45}$$

The numerical analysis of this dual-coordinate population balance was studied by Wojcic and Jones (1998). Another advantage of this approach is that it allows for more accurate calculation of secondary particle characteristics (e.g. specific interfacial area), as the number of crystals within an aggregate is also being predicted. Zauner and Jones (2001) returned to the original aggregation-disruption population balance (eq. 1.44, but employed a parabolic daughter particle distribution (Hill and Ng, 1995; see also chap.3), also on the grounds of the attritionary mechanism.

However, there are several conceptual as well as practical problems involved in the aggregation-disruption approach. An aggregate contains bridges (cementation cites) of various ages and therefore strengths. It may be that the age of each bridge determines the probability of the aggregate being ruptured at that point, regardless of whether it links the aggregate to a single particle or not. For such an approach to be realised, tracking of individual particles in a Lagrangian manner would be necessary. The Wachi and Jones approach yields the number of individual crystals in an aggregate, but not their age since being entrapped in it. Moreover, it would be exceedingly difficult, if not impossible, to measure rates for aggregation and disruption in isolation, and hence obtain an estimate for the constants in eq. 1.44. Breakage rates obtained by subjecting pre-prepared agglomerates to stirring, such as those reported by Wojcic and Jones (1997), refer to the attrition of fully cemented agglomerates, rather than the weak aggregates that govern the crucial intermediate stage in the agglomeration process.

1.4.5 Mathematical modelling of crystallisation processes

The need to predict and control the physical properties of the product in industrial crystallisers, in particular the CSD, and to associate it with the design and operation parameters, meant that a rigorous mathematical framework for the description of crystallisation processes was required. Early researchers in the field, most notably Randolph (1964) and Randolph and Larson (1988), identified the population balance framework as ideally suited to this purpose.

As mentioned in section 1.3, the population balance is an integro-partial differential equation that describes the evolution of population density with respect to one or more independent variables that characterise a particulate system. In the

case of crystal populations we are mainly interested in size distribution, hence the independent variable must be a measure of crystal size, the most obvious choices being a linear dimension (e.g. diameter, radius) or volume (mass). The use of a linear dimension implies that particle shape must also be incorporated in the growth kinetics and crystal mass balances, thus making the equations system-dependent. Even if a shape factor can be constructed by knowledge of the crystal morphology, malgrowth and agglomeration can generate particles of varying shapes. Nevertheless, linear dimensions (commonly referred to as crystal size) have been frequently used in the crystallisation literature because the growth term can be considerably simplified in the case of size-independent growth - crystals of various sizes grow at the same rate, a phenomenon known as McCabe's δL law. When agglomeration and breakage are present, however, the use of particle volume is advantageous, since it is being conserved during these processes.

In section 1.3 it was stated that in the general case, the PBE cannot be solved analytically. To simplify the problem and obtain a tractable model, the following assumptions are possible:

- Steady state operation
- Size-independent growth
- No inter-particle phenomena (agglomeration, breakage)
- Spatial uniformity of concentrations and CSD
- Particles follow the streamlines of the flow

Several levels of approximation arise by adopting certain of these assumptions. Some of them result in analytical solutions that facilitate the analysis of experimental results but have limited applicability; others can be solved via complicated and time-consuming numerical methods; and others are simply intractable given the present computational resources. These levels will be outlined in the following discussion.

1.4.5.1 MSMPR theory

The first attempts at modelling crystallisation processes by means of population balance focused on the simplest and most common type crystalliser, the mixed-

suspension mixed-product removal (MSMPR). The MSMPR is the equivalent of a CSTR reactor, in that both the concentrations of chemical species and the product CSD are assumed to be spatially uniform. In its simplest form, MSMPR theory makes use of all of the assumptions stated above, resulting in a population balance of the form:

$$G_l \frac{dn(L)}{dL} + \frac{n(L)}{\tau} = 0 \tag{1.46}$$

In eq. 1.46 it is also assumed that the input to the reactor is clear liquor, i.e. free of patricles. Nucleation in this case may enter as an initial condition, the population of the "embryo" crystals:

$$L = L_0 \Rightarrow n(L) = n_{L_0} \tag{1.47}$$

The analytical solution of eq. 1.46 is the well-known MSMPR equation (Randolph and Larson, 1988):

$$n = n_{L_0} e^{-\frac{L}{G_l \tau}} \tag{1.48}$$

Thus the product of an MSMPR crystalliser exhibits an exponential CSD. Due to its simplicity and the availability of analytical solution, the MSMPR setup is often employed in experimental studies, in particular for the extraction of crystallisation kinetics. More complex crystallisation systems may exhibit size-dependent growth and inter-particle phenomena, in which case analytical solution is usually not possible. The absence of the time coordinate, however, still makes analysis of results and kinetics extraction much easier. Collier and Hounslow (1997) have proposed a method for extracting growth and agglomeration kinetics from MSMPR experiments; Hostomsky (1987) has derived a semi-analytical (based on Taylor expansion) solution for nucleation and size-independent agglomeration in an MSMPR.

1.4.5.2 Batch crystallisers

In the case of batch crystallisers time enters as an independent coordinate, and solution of a partial differential equation is necessary. Another complication is the fact that the solute species' concentrations, and hence supersaturation, are not constant either. At this point it must be recalled that crystallisation kinetics are functions of supersaturation, and vary with time too. This implies that the population balance must be coupled with the species' mass balances. The resulting equations in terms

of particle size are:

$$\frac{\partial n(L,t)}{\partial t} = -\frac{\partial(G_l(C_i)n(L,t))}{\partial L} \tag{1.49}$$

$$\frac{dC_i}{dt} = \sum_j r_j C_i \tag{1.50}$$

where r_j denotes consumption due to either chemical reactions (in precipitation) or crystal growth. Eq. 1.49 must be complemented by two initial conditions for the population density and one for each species' mass balance:

$$t = 0 \Rightarrow n(L,t) = n_{t_0} \tag{1.51}$$

$$L = L_0 \Rightarrow n(L,t) = n_{L_0} \tag{1.52}$$

$$t = 0 \Rightarrow C_i = C_{i,t_0} \tag{1.53}$$

Note that the terms r_j may refer to both chemical reactions (in precipitation) and consumption due to crystal mass deposition. Though the two-dimensional hyperbolic equation eq. 1.49 can be solved by the method of characteristics, the species' mass balances may involve non-linear reaction rates and in general analytical solution is not possible.

The next option is the method of moments (Hulburt and Katz, 1964). The k-th moment of the population density with respect to particle size is defined as:

$$m_k = \int_0^\infty L^k n(L,t)dL \tag{1.54}$$

The zeroth, second and third moments with respect to size have a physical meaning: they are proportional to the total number, surface area and volume (mass) of crystals, respectively. The moment transformation, already defined by eq. 1.24, is written below in terms of particle size:

$$L^k(f(n(L,t),L)) = \int_0^\infty L^k f(n(L,t),L,t)dL \tag{1.55}$$

By applying this transformation to the population balance, it is hoped that every term in the equation can be arranged into a function of the moments of the population density. Unfortunately, this is only possible when no inter-particle phenomena are present and the growth function assumes specific forms. In the case

of size-independent growth, the following infinite set of ODEs results:

$$\frac{dm_k}{dt} = kGm_{k-1} + L_0 \cdot B_0 \tag{1.56}$$

To obtain a closed set of equations the first four moments are required, (starting from the zeroth) as the third moment denotes the production of crystal mass, thus establishing the coupling with the species' mass balance. Eqs. 1.56, together with eqs. 1.52, thus form a closed set of ODEs that can be numerically integrated to yield the temporal evolution of the moments in a batch crystalliser. The mean crystal size can then be extracted from the moments, according to the equation (Randolph and Larson, 1988):

$$\overline{L}_{k_1,k_2} = (\frac{m_{k_1}}{m_{k_2}})^{\frac{1}{j-k}} \tag{1.57}$$

This general expression includes all possible definitions of mean size, such as the volume-average and area-average (Sauter-mean). In conclusion, it must be pointed out the method of moments cannot yield the CSD, but only averaged properties of it. Numerical methods are required to retrieve the full distribution.

1.4.5.3 Inter-particle phenomena

The description of inter-particle particulate phenomena, such as agglomeration and breakage, through a size-based population balance is very complicated. Hounslow, Ryall and Marshall(1988) presented a size-based population balance:

$$\frac{\partial n(L,t)}{\partial t} = -\frac{\partial(G_l(C_i)n(L,t))}{\partial L} + \\ + \frac{L^2}{2}\int_0^L \frac{\beta[(L^3-\lambda^3)^{\frac{1}{3}},\lambda]n[(L^3-\lambda^3)^{\frac{1}{3}},t]n(\lambda,t)}{(L^3-\lambda^3)^{\frac{2}{3}}}\,d\lambda - n(L,t)\int_0^\infty \beta(L,\lambda)n(\lambda,t)\,d\lambda \tag{1.58}$$

The above authors have also presented a numerical method for its solution. The method of moments is not applicable, as not all of the terms arising from the transformation can be arranged into moments.

Inter-particle phenomena conserve volume, hence the volume-based expressions for agglomeration shown in section 1.4.4.1 are much simpler. Analytical solutions for these models are more common in the aerosol literature (e.g. Scott, 1968; Ramabhadran et al., 1976; Drake, 1972) and are not very useful in crystallisation, because they usually assume that agglomeration is the only mechanism and require

an exponential initial distribution. An exception is the semi-analytical solution of Hostomsky (1987) for nucleation and size-indpendent agglomeration, which was developed to describe an MSMPR crystalliation experiment. In chapter 3, a numerical method for solving volume-based population balance will be developed.

1.4.5.4 Spatially distributed crystallisers

Having considered time-dependency, and interparticle phenomena, the last assumption to drop is that of spatial uniformity. This is one of the most recent and least developed fields in crystallisation research. The main ways in which spatially distributed properties may affect the outcome of crystallisation are:

- Size classification. This implies that particles do not follow the streamlines of the flow, and as a result the CSD is not the same at all points.
- Non-uniform supersaturation field. This may be due to vartiations in concentration or temperature. As a result crystallisation kinetics, which are functions of supersaturation, exhibit spatial variations that have a profound effect on CSD.
- Hydrodynamic effects in interparticle phenomena. Agglomeration, which usually occurs via the orthokinetic mechanism, and breakage are heavily dependent on local hydrodynamics. A non-uniform flow field can severely affect the course of these processes.

Size classification has been considered by early researchers (e.g. Randolph and Larson, 1988), but only in an empirical manner. The most common example is classified product removal, i.e. the removal of particles above or below a certain size. This may either arise naturally or be enforced (e.g. by the installation of a screening device in the outflow), and can be easily taken into account by introducing a classification factor in the model, which must be determined experimentally. Variations of CSD within the reactor have seldom been considered, however, and particles are usually assumed to follow the streamlines of the flow.

Dropping the last assumption would imply considering the particles as a separate phase, and coupling the population balance with the Navier-Stokes equations. If

size classification is also to be taken into account rigorously, a separate momentum balance must be solved for each particle size, resulting in a virtually intractable computational problem. In systems such as precipitation where the particles are very small ($< 10\mu m$), though, the interaction between particles and flow can be neglected. Since in this work we are dealing with such a system, no further consideration of this issue will be attempted.

Spatial variations in supersaturation can be very important in precipitation. In case of very fast reactions, "hot spots" of high supersaturation values may be generated around points of reactant input. Growth and nucleation kinetics are very sensitive to supersaturation, with high values of it usually favouring nucleation. This results in more crystals being produced, whose growth causes a rapid depletion of supersaturation, culminating in a CSD shifted towards the lower end of the size spectrum.

Pohorecki and Baldyga (1983,88) were among the first to proposed models for taking into account the spatial distribution of concentration. Tavare (1986) presented a comprehensive review of mixing models with relevance to precipitation. Generally, such models fall in either of two categories:

- Integration of population balance with the Navier-Stokes equations.
- Solution of the population balance within a compartmental model, custom-derived for a specific equipment, based on hydrodynamic considerations and possibly CFD.

Models of the first kind are usually based on the moments transformation, in order to minimise the number of extra differential equations that need to be solved alongside the Navier-Stokes equations; such are the models of Wei and Garside (1997) and van Leeuwen, Bruinsma and Van Rosmalen (1996). Recently Paschedag, Tornatora, Ausner and Signorino (2002) et al. have coupled the CFD with a discretised population balance comprising 40 classes. In these works the coupling between CFD and population balance is usually one-way, i.e. the Navier-Stokes equations are solved for each new time step, and the resulting concentrations are employed for propagating the PBE. Excessive computation time is the main problem faced by this approach.

Models of the second kind sometimes employ CFD for calculating average hydrodynamic properties of compartments (Zauner and Jones, 2000). The compartmental model is custom-derived for a specific kind of equipment, usually a stirred tank. In this work, this approach will benefit from a generalised compartmental model, that can be tuned to an arbitrary geometry.

Chapter 2

Hybrid CFD - reaction engineering approach

2.1 Introduction

2.1.1 Present challenges in multiphase reactor modelling

The performance of gas-liquid reactors relies on the combined outcome of multiphase fluid dynamics, interfacial mass transfer and chemical reactions. The detailed, simultaneous analysis of the above phenomena is out of reach given the currently available computing power, partly because they take place at different scales. Furthermore most of these phenomena are non-linear, and empirical models cannot be extrapolated too far from the set of design parameters and operating conditions under which they were extracted. As a result, multiphase reaction engineering is still confronted with major challenges from a modelling point of view.

The quantitative description of mixing is a major issue of reaction engineering. Early research attempted to quantify mixing without knowledge of the flow field, and a major step in this direction was taken when Danckwerts (1953,58) introduced the concept of Residence Time Distribution (RTD). The RTD arising from tracer experiments can be used to identify situations that lie between the two extremes of plug flow and complete mixing, and a variety of compartmental models, such as the tank-in-series and the cell model with backflow (or its continuous equivalent, the axial dispersion model) have been proposed to accommodate them (Mecklenburg

and Hartland, 1975; Nauman and Buffham, 1983). Also notable is the "network-of-zones" approach of Mann and Hackett (1988), a description of the mixing hydrodynamics that offers a computationally inexpensive alternative to CFD.

More fundamental information about mixing can be obtained through the Navier-Stokes equations which, though, require an excessive amount of computation to resolve all turbulence scales by direct numerical simulation (DNS). Shortcuts with variable levels of detail have been developed, ranging from the commonly used in engineering turbulent viscosity- based approaches such as the $k - \epsilon$ (Launder and Spalding, 1972) to the much more accurate Large Eddy Simulation (LES, see e.g. Pope, 2000), which, though sparingly used today, may become the industry standard in the future. The coupling of fluid dynamics with complex and often non-linear chemical reaction systems, however, results in a steep increase of computational time, that often prohibits the use if this approach for reactor scale-up purposes.

In the last decades, the reaction engineering community has started to explore the prospect of employing CFD in the modelling of multiphase reactors. Torvik and Svendsen (1990) made one of the first attempts to model mixing and reaction in a bubble column with CFD, based on the $k-\epsilon$ model, while mass transfer and chemical reaction were still dealt with by assuming constant reaction enhancement. Ranade (1995), Delnoij, Kuipers and van Swaaij (1997), Van den Akker (1997), Kuipers and van Swaaij (1997), and Dudukovic, Larachi and Mills (1999) among others, have reviewed the current state of the art regarding the application of CFD to chemical reaction engineering. The reviews of Jakobsen, Sannaes, Grevskott and Svendsen (1997) and Joshi (2001) focus on CFD modelling of gas-liquid flows, with relevance to bubble column reactors.

The direct application of CFD to chemical processes faces several obstacles, however. Even in single phase reactors, chemical reactions are described by highly non-linear terms that often cause numerical instabilities. More complex phenomena, such as interfacial mass transfer with chemical reaction, are encountered in multiphase or catalytic reactors, while in cases such as precipitation and biochemical engineering additional equations, such as the population balance, must be solved alongside the Navier-Stokes equations. The computational resources required are often prohibitive, while the complexity of the problems that arise from the coupling

of the fluid dynamics with the chemical phenomena means that the systems must be treated case-by-case.

2.1.2 The need for hybrid approach

Recently, "hybrid" approaches have emerged as an alternative. In those CFD is employed only for hydrodynamic simulation, while the chemical phenomena are resolved in a custom-built compartmental model. Although this decoupling cannot be applied to cases where the coupling between hydrodynamics and chemistry is very strong, such as in combustion, many chemical reactors with a liquid bulk phase fall into this category. Bauer and Eigenberger (1999) used a "zone model" to study a bubble column reactor; Bezzo, Macchietto and Pantelides (2000) developed an interface of communication between the gPROMS modelling software and a commercial CFD code; Zauner and Jones (2000) used a segregated feed model in conjunction with CFD to study precipitation in a stirred tank. Though these approaches represent essentially a compromise, their lifetime is likely to be quite long, as the full CFD solution would require a radical breakthrough in computer technology.

The approach proposed here belongs to this class. In its context, the detailed hydrodynamic information on meso- and macro- mixing that CFD provides is mapped into a more tractable compartmental model, suitable for efficient calculation of chemical reactions and other nonlinear phenomena (e.g. precipitation), while still incorporating the essential features of the geometry and operating conditions of the reactor. The main concept of the approach, the mapping of the CFD grid into the generalised compartmental model, is introduced in the next section. This is followed by a procedure for linking it to $k-\epsilon$ - based CFD, and an application to bubble column reactors by superimposing a phenomenological model of mass transfer with chemical reaction. The basic methodology is not limited to these, however, and may be used in conjunction with more advanced turbulence models, or other phenomenological reaction engineering models.

2.2 The generalised compartmental model

2.2.1 Basic concepts

An outline of the approach can be provided as follows: first, the CFD simulations are carried out on a grid that incorporates the detailed geometry of the reactor. Subsequently, a compartmental model is derived via mapping the CFD grid into a coarser network of compartments. The compartmental model, being much more computationally tractable than the CFD, is finally used for the reaction engineering calculations, by superposing on it the chemical reactions, population balance and any other complex models that may be desired. The implicit assumption is that the coupling between the hydrodynamics and the chemical reactions is one-way, i.e. the reactions have no significant feedback on the hydrodynamics. In gas-liquid reactors this is usually the case, as long as the reactions are not highly exothermal and the reaction product does not affect the mixture rheology (e.g. polymerisation, dense solid phase precipitation).

The mapping concept is best illustrated through the example of fig. 2.1, where five subdomains of the CFD grid are mapped into a compartmental model that retains only the fluxes between the original subdomains. The compartmentalisation must be based on physical reasoning, and will be discussed in more detail later; at the moment it will be pointed out that in fig. 2.1 compartment 5, entirely enclosed within 4, may be devised to contain a vortex that results in dead space. Another point that will become clear later is that the dimensionality of the CFD grid is irrelevant, i.e. a 3-D grid may also be mapped in the same way.

It is important to stress the difference in the physical interpretation of a compartment between the RTD models and the present approach. Whereas in RTD models a compartment arises from the fitting of an RTD curve but does not necessarily correspond to a physical part of the reactor, in the present approach the compartments are generated from the CFD grid, and are thus tightly bound to the geometry of the particular piece of equipment.

Conventional compartmental models rely on RTD experiments for the determination of their parameters. Rather than that, in this approach the parameters of the compartmental model are determined from CFD. Therefore the approach is fully

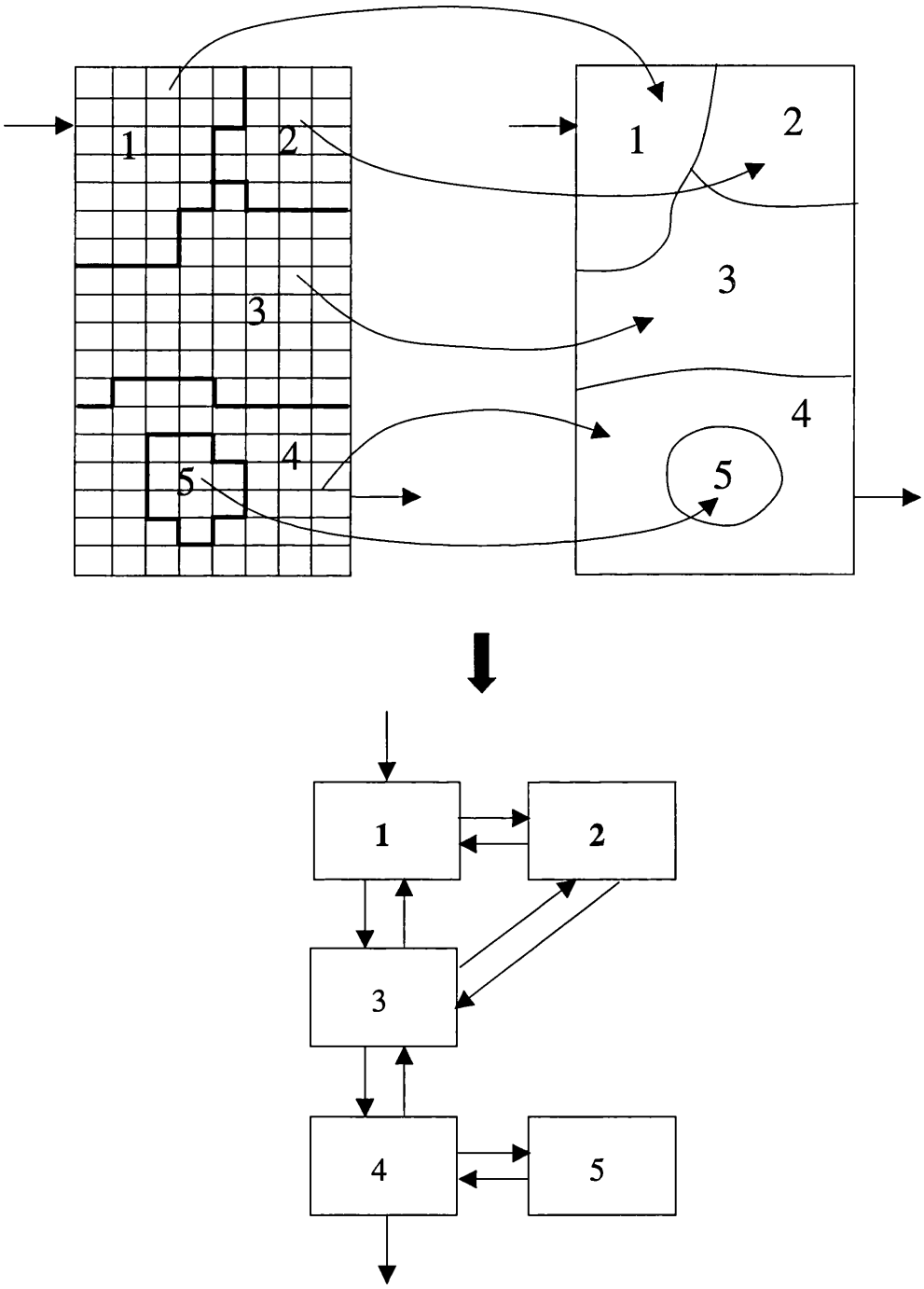

Figure 2.1: Mapping of a CFD grid into a compartmental network.

predictive and does not require experimental input at any of its stages. Experimental validation is required only with respect to the CFD model, and this is an issue that will not concern us here. The methodology that will be proposed for connecting the compartmental model to the CFD does not restrict the approach to a particular kind of CFD model; therefore, if more accurate models become available in the future, they may be used instead to obtain more accurate results.

The link between the CFD and the compartmental model is provided in the form of the virtual tracer experiment. This is a simulation of the dispersion of a passive tracer carried out with both the CFD and the compartmental model, in a manner that indicates to which extent the latter captures the mixing effects of the actual reactor, as simulated by the former. Once the compartmental model is able to carry out the virtual tracer experiment and yield approximately the same results as the CFD, the designer can be assured that it describes adequately the mixing behaviour of the reactor and proceed to introduce into it the chemical reactions and any other complex phenomena whose direct coupling with CFD would be impracticable.

To automate the mapping process we introduce the generalised compartmental model, a network of fully-mixed compartments with many degrees of freedom, that may be reduced into a specific compartmental model when its parameters are specified (via the CFD). The parameters are the number of compartments, the way they are connected and the values of the flowrates. The exact shape of the compartments and their boundaries is of no essence, since they are spatially uniform; instead, all that matters is the way they are connected.

2.2.2 Topology of the compartmental model

Let K be the number of compartments in the network. Each compartment will be identified by the compartment index, k:

$$k \in \{1, 2, .., K\} \tag{2.1}$$

Each compartment is allowed to have boundaries with a maximum of L neighbours, the indices of which are established with the mapping:

$$I : (k, l) \rightarrow I(k, l) \tag{2.2}$$

With each compartment k we associate a maximum of L connectivity flowrates, denoting flow towards its $l-th$ neighbour. Consequently, a pair of opposite direction flowrates is associated with each boundary. This two-way connectivity provides a means of introducing local turbulent exchange.

The inverse map of $I(k,l)$ yields the indices of the L (at most) compartments that have an outflow towards k:

$$I' : (k,l) \rightarrow I'(k,l) \tag{2.3}$$

The one-dimensional array of compartment indices, together with the maps $I(k,l)$ and $I'(k,l)$ establish the topology of the network, i.e. the way the cells are connected. Only $I(k,l)$ is required to specify the network; $I'(k,l)$ may be easily determined by the program.

The dimensionality of the CFD grid space is not retained in the compartmental model, which contains information only about how the compartments are linked. For example, it makes no difference whether the neighbours of a compartment are in the x, y or z direction, since all of them would be mapped in the same way. An important consequence of this is that CFD grids of any dimensionality may be mapped. In the example of fig. 2.1 the mapping of a 2-D grid was illustrated, but a 3-D one would be mapped in exactly the same way. The one-dimensional compartment index k is arbitratry, while all topological information is contained in $I(k,l)$ and $I'(k,l)$.

To describe multi-phase reactors, separate compartments are employed for each phase. In this work the structure of the network will be the same for all phases, to allow the implementation of mass exchange between the phases via the penetration model. Thus the building block of a two-phase compartmental network comprises the two compartments of each phase, in contact with up to L other compartments (fig. 2.2). The following notation is introduced:

$\mathbf{C}_{ikp}$ Concentrations of chemical species i, spatially averaged over compartment k at phase p.

$\mathbf{V}_{kp}$ Volume of compartment k at phase p.

$\mathbf{Q}^{in}_{kp}$ Inflow into the system (i.e. the whole reactor) at compartment k, at phase p.

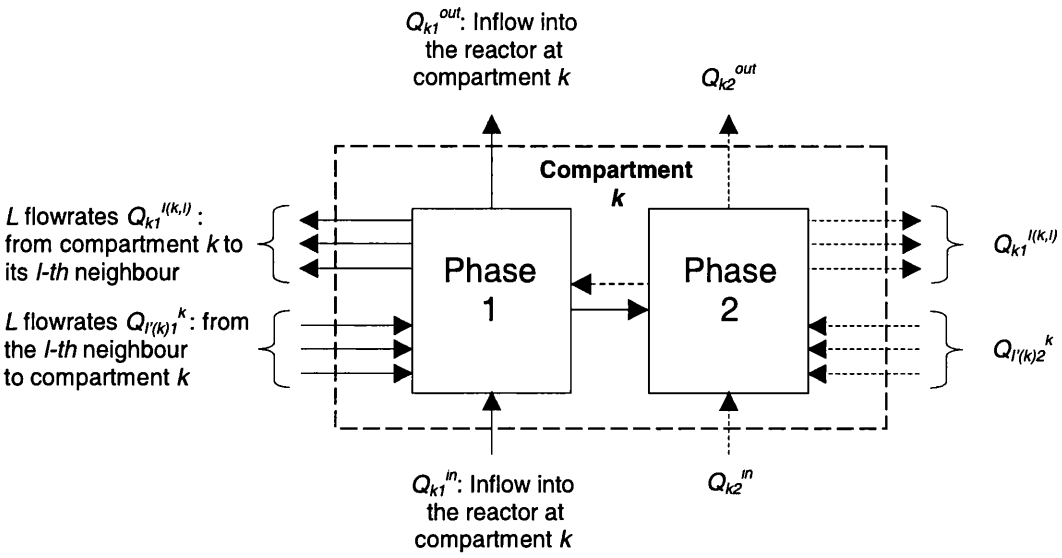

Figure 2.2: The building block of the multi-phase generalised compartmental model (two-phase).

$\mathbf{Q}_{kp}^{out}$ Outflow from the system at compartment k, at phase p.

$\mathbf{Q}_{kp}^{I(k,l)}$ Flow from compartment k, at phase p, towards its l-th neighbour, the compartment $I(k,l)$.

$\mathbf{Q}_{I'(k,l),p}^{k}$ Flow into compartment k, at phase p, from its l-th neighbour, the compartment $I'(k,l)$.

The total number of parameters thus are:

- Number of cells (1)
- Volumes of cells (K)
- Connectivity flowrates, maximum [KxL]

The total number of parameters that may be arbitrarily specified (i.e. degrees of freedom) is however constrained by the mass balances, as it will be shown in the next section.

2.2.3 Constraints

All of the parameters mentioned above may not be arbitrary specified. To begin with, the number of cells is inferred from the specification of the cell volumes. Moreover, the cell volumes in a phase must sum up to the total volume of the system in that phase:

$$V_p = \sum_{k=1}^{K} V_{kp} \tag{2.4}$$

For every cell in a phase the mass balance must be satisfied, resulting in K further constraints:

$$\sum_{l=1}^{L} Q_{kp}^{I(k,l)} + Q_{kp}^{out} = \sum_{l=1}^{L} Q_{I(k,l)p}^{k} + Q_{kp}^{in} \tag{2.5}$$

2.2.4 Mass balances

A compartment k has flowrates to and from L neighbouring cells, and there may also be input or output from the whole system at a location in the vicinity of k (fig. 2.2). The dynamic mass balance for the concentration of a chemical species i can be written as:

$$\frac{dC_{ikp}(t)}{dt} = \frac{1}{V_{kp}} \cdot \left[-Q_{kp}^{out} \cdot C_{ikp}(t) - \sum_{l=1}^{L} Q_{kp}^{I(k,l)} \cdot C_{ikp}(t) + \right.$$
$$\left. + Q_{kp}^{in} \cdot C_{ikp}^{in}(t) + \sum_{l=1}^{L} Q_{I'(k,l)p}^{k} \cdot C_{iI'(k,l)p}(t) \right] + f(C_i(t)) \tag{2.6}$$

were f is the reaction term (j is the reaction index):

$$f(C_i(t)) = \sum_{j=1}^{J} r_j(C_{ikp}(t))$$

This balance may be written more concisely as:

$$\frac{dC_{ikp}(t)}{dt} = W_{out} \cdot C_{ikp}(t) + W_{in} + f\left(C_i\right) \tag{2.7}$$

where:

$$W_{out} = -\frac{1}{V_{klp}} \cdot \left(\sum_{l=1}^{L} Q_{kp}^{I(k,l)} + Q_{kp}^{out} \right)$$

$$W_{in} = +\frac{1}{V_{kp}} \cdot \sum_{l=1}^{L} Q_{I(k,l)p}^{k} \cdot C_{iI(k,l)p}(t) + \frac{1}{V_{klp}} \cdot C_{ikp}^{in}(t) \cdot Q_{kp}^{in}$$

Note that, in the case of the gas phase, the reaction term may include the consumption due to chemical absorption. As this is a system of ordinary differential equations, initial conditions must be specified for its solution:

$$t = 0 \rightarrow C_{ikp} = C^0_{ikp} \tag{2.8}$$

2.2.5 Link with CFD

We shall now describe how the generalised compartmental model gets linked to the geometry and operating conditions of a specific reactor via a CFD simulation. As it has been mentioned, the topology of the compartmental model is specified through a mapping of the CFD grid cells onto the compartments. Apart from the topology, we need to determine the flowrates between the compartments.

In this work we shall distinguish between mixing due to the mean (time-averaged) flow field, and turbulent local mixing i.e. any localised mixing effects, caused by turbulence at scales higher than the Kolmogorov. To describe the former we shall use the mean flow information that is yielded by a CFD code supplied with a turbulence model based on time averaging, such as the $k - \epsilon$.

To account for the latter we shall employ the virtual tracer experiment, i.e. compute the dispersion of a passive scalar using the above CFD code. The flow and scalar concentration fields that result from CFD will subsequently be used to derive a compartmental model, on which the chemical reactions can be superimposed provided that they have no significant feedback on the hydrodynamics.

2.2.5.1 Overall tuning algorithm

The overall process of constructing and fine-tuning the generalised compartmental network can be summarised by the following algorithm:

Step 1 Carry out the flow field computation and virtual tracer experiment with CFD.

Step 2 Determine the network structure and estimate connectivity flowrates based on mean flow.

Step 3 Introduce two-way flowrates to account for local turbulent exchange.

Step 4 Evaluate the convergence between the compartmental model and the CFD tracer experiment. If poor, return to step 3. If not possible at all, return to step 2 and modify the structure.

Step 5 Having established a network that adequately describes the dispersion, superimpose the reaction engineering models to account for complex reaction systems, gas-liquid mass transfer, micromixing etc.

2.2.5.2 Determination of network structure and flowrates

At this point, the mapping of the CFD grid into the compartmental model can be made on the basis of a qualitative inspection of the flow and concentration fields yielded by CFD, based on the designer's intuition. Future work could consider automating the process through a quantitative algorithm. A few heuristic rules such as the following could be used to guide the decision:

- The flow should not change direction along a boundary.
- Streamlines should cross similar distances within a certain compartment.
- Regions of uniform concentration may be accomodated by a single compartment.

The implication of the first rule is that the all main flow features giving rise to macromixing, such as circulation loops, must be accounted for by the initial structure (apart from small vortices, which may be fully contained within a single compartment). The second rule suggests that all fluid elements entering a compartment will have similar residence times at that compartment. The third rule suggests that turbulent exchange does not have to be accounted for by the inter-compartmental flowrates alone, but may be inherent in the network itself.

The flowrate at any compartmental boundary can be calculated from the CFD results by summing up the flows through the CFD cell faces that comprise that boundary.From the equation of continuity, the total flow along any finite surface (such as a compartment) is:

$$J_j = \oint \rho u_j dA_j \tag{2.9}$$

Thus the flowrates between compartments can be calculated from the CFD results by summing up the flows through the CFD cell faces that comprise that boundary. Such information is readily available in most CFD codes, as the term $\rho u_j dA_j$ must be calculated to form the convective part of the coefficient matrix. The network thus generated features one-way connectivity; the two-way is reserved for the introduction of turbulent mixing. This will be carried out by adjusting the two-way flowrates until the virtual tracer experiment yields the same results with both the CFD and compartmental model. In the present work, this adjustment was carried out by trial and error; future work may consider more systematic methods of attaining it.

2.2.5.3 The virtual tracer experiment

As we saw in the preceding paragraph, the main flow and circulation patterns in the reactor are accounted for in the compartmental model by averaging the mean flow information yielded by the CFD, to yield the connectivity flowrates in the former. The network thus generated, however, would account only for the mixing due to the large-scale features (such as the circulation between riser and downcomer in a loop reactor), while leaving out the local mixing effects. A tubular reactor, for instance, would be reduced to a tank-in-series rather than an axial dispersion equivalent model. To incorporate local mixing into the compartmental model, we shall employ the virtual tracer experiment.

The main idea behind the concept of the virtual tracer experiment is that both the CFD and compartmental model must be able to simulate the dispersion of a passive scalar (tracer) and yield similar results. This will signify that the compartmental model captures the turbulent mixing behaviour of the actual reactor (up to the level of accuracy exhibited by the CFD model) and may be safely employed for the reaction engineering analysis. Several modifications of the two-way flowrates between compartments may be carried out so that the compartmental model converges with the CFD with respect to the simulation of the virtual tracer experiment; if that is not at all possible, it signifies a fundamental inadequacy of the compartmental structure, and that the network should be constructed anew.

The virtual tracer experiment must be designed in such a way that it is indicative

of the turbulent mixing during the actual reactor operation. In particular, the tracer must be introduced in a way that emulates the feeding policy of the reactor. In the case of a continuous flow reactor, where the reactant is fed through an inlet, this can be straightforwardly achieved via a stepwise introduction of the tracer at the inlet point. For a semi-batch bubble column, however, this goal is not so straightforward to attain, as the reactant is fed along the bulk at a variable rate, depending on the local presence of bubbles. The best way to emulate this feeding policy seems to be the stepwise introduction of the tracer into every cell/compartment of the CFD/compartmental model, as a step change at a value proportional to the local gas-liquid interfacial area. This can be computed on the basis of the local volume fraction and mean bubble diameter (d_b):

$$\dot{S} \sim \frac{\varepsilon_g \cdot V_{cell}}{(\pi d_b^3/6) \cdot V_{cell}} \cdot \pi d_b^2 \frac{6\varepsilon_g}{d_b} \tag{2.10}$$

An advantage of this approach is that the structure of the compartmental model will be maintained as simple as possible, by accounting only for the most important mixing events that determine reactor performance during actual operation. To illustrate this point, consider the following example. If a tracer is introduced at a single point in a bubble column, steep gradients are likely to be generated around that point. In order to capture this effect with the compartmental model, several small compartments will be required to accommodate that region. These gradients will not arise during the actual operation, however, because the introduction of the reactant is not that localised; in fact, it takes place throughout the whole of the riser section, resulting in a much more smooth concentration distribution. By introducing the tracer in a way that emulates the actual feeding policy, we ensure that the compartmentalisation process will result in the smallest number of compartments necessary to describe adequately the mixing effects that are of importance during actual reactor operation, and nothing more than that.

In the experimental RTD approach a passive tracer is introduced in an impulse, step or periodic manner, and the RTD is extracted from concentration measurements at the output. The shape of the RTD curve is indicative of a certain behaviour (e.g. plug flow, dead space), and a compartmental model is subsequently devised to emulate this behaviour. It is important to note, however, that there is no one-to-one correspondence between the detailed reactor design and RTD; in principle,

an infinity of reactors could yield the same RTD. The virtual tracer experiment, on the other hand, yields the detailed spatial distribution of the tracer. Since the CFD contains the detailed geometry and operating conditions of the reactor, it follows that the result of the virtual tracer experiment and hence of the compartmental model derived on its basis uniquely corresponds to the specific reactor under consideration.

Finally let us compare the present approach to the earlier hybrid models. These relied on predictions of turbulent quantities via CFD, such as turbulent kinetic energy and dissipation rate, which were used to estimate the flowrates between compartments through phenomenological equations. As Montante, Lee, Brucato and Yianneskis (2001) have shown, these quantities are severely underestimated by $k-\varepsilon$ - based simulations. On the other hand, this methodology is based on mean flow information and the virtual tracer experiment, and thus avoids the shortcomings of the $k-\varepsilon$ CFD simulations.

2.2.5.4 Extraction of mixing information from $k-\epsilon$ based CFD

In principle, the results from any CFD simulation may be averaged to parametrise the generalised compartmental model. Multiphase flow models were reviewed in chapter 1.2.2. Among the main kinds of models that can be employed (volume tracking, Eulerian-Lagrangian, Eulerian-Eulerian) it is was concluded that the Eulerian-Eulerian approach is best suited for resolving the main flow characteristics in large scale equipment within a reasonable computational time. With respect to turbulence modelling, for engineering - scale gas - liquid flow computations so far only the $k-\epsilon$ model has been applied and tested, though it is unlikely to be the most suitable (Jakobsen et al., 1997). For this reason, it has been employed for the case study presented here to illustrate the approach. The fact that the methodology is not linked to a specific CFD model, however means that it will benefit from any future developments in the field.

The $k-\epsilon$ model (Launder and Spalding, 1972) belongs to a family of models that are derived through Reynolds averaging, a temporal filter that decomposes the hydrodynamic variables into a mean and a fluctuation:

$$\phi = \overline{\phi} + \phi' \tag{2.11}$$

Reynolds averaging generates several terms of the kind $\overline{\phi_1'\phi_2'}$, i.e. mean values of products of fluctuations of different variables, resulting in a closure problem. The approach of the $k-\varepsilon$ model is to model the effect of these terms on the mean flow field by two extra transport equations, one for the turbulent kinetic energy (k) and one for the energy dissipation rate (ε). These equations and their derivation may be found in Launder and Spalding (1972). The mean flow field that results encompasses the gross features of the flow such as recirculation zones, large-scale vortices giving rise to dead space, bypassing routes etc, that give rise to macromixing. Further, localised mixing effects are revealed when one computes the field of a scalar quantity.

In order to establish a correspondence between the hydrodynamic information yielded by the $k-\epsilon$ and the mixing concepts used in reaction engineering, we shall distinguish between mixing due to the mean (time-averaged) flow field, and local turbulent exchange i.e. any localised mixing effects, lateral or axial, caused by turbulence at scales higher than the Kolmogorov. The former is fully described by the mean flow field, while the latter can be revealed through CFD by the computational equivalent of a tracer experiment. No mention is made to micromixing, which is not at all captured by the $k-\epsilon$ model. Its incorporation would require further phenomenological models, such as those of Baldyga, Bourne and Hearn (1997).

2.2.5.5 Superposition of reaction engineering models

Once a compartmental network that properly accounts for mixing has been established, various mechanistic reaction engineering models can be superimposed on it. Although the present work focuses on bubble column reactors, the main concept lends itself to other potential reaction engineering applications, such as:

- Single-phase reactors with complex nonlinear reaction systems, whose direct coupling with CFD would pose serious computational problems.
- Stirred tank reactors with fast reactions, where micromixing models can be superimposed in the compartments located at the inlets.
- Precipitation, where additional population balance equations describing crystal growth, agglomeration or breakage must be solved.

- Multiphase non-catalytic reactors (e.g. bubble columns), where interfacial mass transfer and chemical reaction models can be accommodated within the compartments. These models must be resolved at the interfacial scale, and therefore cannot be directly incorporated in a CFD code.
- Catalytic and biochemical reactors, where adsorption models or cell population balances may be needed.

Some of the reaction engineering models required for the above applications require extra hydrodynamic information: e.g. a gas-liquid mass transfer model requires knowledge of gas hold-up, or a particle agglomeration model (in precipitation) may be a function of the energy dissipation rate. This extra information may be obtained from the CFD simulations in the same way that the mixing information was extracted, i.e. by averaging the distributed information yielded by CFD within the compartments.

The present work focuses on bubble column reactors, and a way of introducing interfacial mass transfer and chemical reactions in the network according to the penetration theory will be demonstrated. This way, the absorption enhancement due to a fast, but not instantaneous, chemical reaction system can be properly taken into account. It must be noted, however, that the main concept lends itself to other potential reaction engineering applications, such as micromixing in stirred tanks and precipitation.

2.3 Application to bubble column reactors

2.3.1 Mixing in bubble column reactors

The traditional chemical reaction engineering approach to mixing is to envisage it as the sum of two processes: macromixing, i.e. the mixing of segregated fluid elements due to the gross features of the flow field, and micromixing, i.e. molecular-level transfer of material between the elements, thus bringing the reactants into contact (Danckwerts, 1958). An intermediate mixing scale (mesomixing) has also been suggested (Baldyga, Bourne and Hearn, 1997) to account for the mixing of the inlet plume with its surroundings. This approach has been developed mainly

for stirred tanks, where incomplete mixing can create very steep gradients in the case of fast reactions. This is not the case with gas-liquid reactors, however, for the following reasons:

- A very fast reaction would be exhausted at the interface.
- In case of a reaction that proceeds in the bulk the gradients are not as steep as in stirred tanks, due to the fact that the feed is not localised around an inlet point but distributed along the whole riser section.

The main causes of mixing in bubble columns are liquid circulation due to forced or natural convection, and lateral turbulent exchange of regions with high gas holdup (and hence reactant feed) with regions with low gas holdup. Thus no phenomenological model of micromixing is required; instead, a model of interfacial mass transfer and chemical reaction must be employed to link the compartments of the liquid phase with those of the gas phase.

2.3.2 Model of interfacial mass transfer and chemical reaction

Apart from the coupling of bulk-phase chemical reactions with hydrodynamics, the simulation of multiphase reactors requires the solution of additional models that account for interphase transport. In the case of gas-liquid reactors, conventional ways of accounting for mass transfer were reviewed in section 1.1.2. It was concluded that research so far has concentrated on problems where chemical reactions and mass transfer in the interface can be decoupled or simplified. This is possible in the following situations:

- Reactions too slow or instantaneous, that do not occur in the same time frame as mass transfer.
- Steady-state problems, where reaction enhancement is constant.
- Simple reaction systems, for which analytical solutions to the diffusion-reaction equation can be found.

Unless the time scales of a gas-liquid reaction and mass transfer differ by orders of magnitude (instantaneous or slow reaction regime), however, these two phenomena

are intricately coupled. The consumption of absorbent at the interface increases the driving force for mass transfer, a phenomenon known as reaction enhancement, and the amount of absorption can only be resolved by detailed solution of the diffusion - reaction equations that result from a phenomenological model such as the film, penetration and surface renewal. These models were reviewed in sec. 1.1.1.1.

2.3.2.1 Implementation of the penetration model

Among the most established absorption models, the penetration theory will be employed here because it is conceptually suited to the mechanism of bulk-interface interaction in dispersed multiphase flows: a bubble moving with a slip velocity relative to the bulk is in contact with a surrounding element of bulk fluid for a short amount of time, and subsequently overcomes it. The only parameter, the contact time, can be theoretically estimated from the bubble mean diameter and terminal velocity:

$$\tau = \frac{d_b}{u_b} \tag{2.12}$$

These concepts are implemented here according to the following scheme: the liquid element surrounding the bubble and the bulk are considered as two separate dynamic reactors that operate independent of each other and interact at discrete time intervals. In the beginning of the contact time, the interface is being detached from the bulk. Subsequently, the species mass balances equations are solved in each interfacial and bulk reactor. When overcome by the bubble, the liquid element returns to the bulk and gets mixed with it. Not all the liquid elements within a cell will contact the gas simultaneously, but since the contact time is very small compared to the process time the situation is equivalent to having a very large interfacial element emerging from the bulk, returning to be mixed and then re-emerging in the surface.

The concept of the penetration theory (Higbie, 1935) has previously been employed for modelling flat interface (Hostomsky and Jones, 1995) and stirred tank gas-liquid reactors (Van Elk, Borman, Kuipers and Versteeg, 2000). It offers a major advantage from a computational point of view: the bulk and the interfacial phenomena are decoupled, and each one can solved independent of each other, interacting via the mixing process. The new concentrations that arise from the mixing

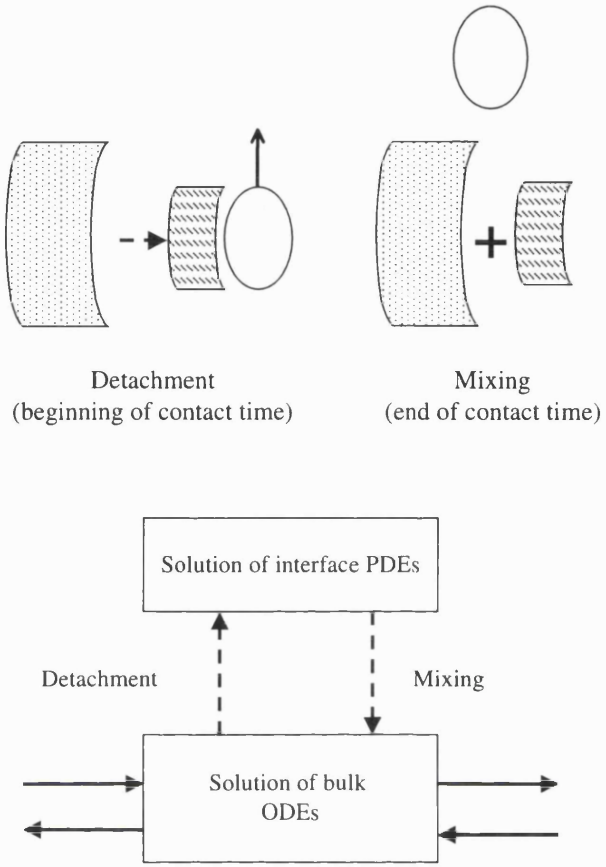

Figure 2.3: Penetration theory: physical interpretation and implementation in the model

are the initial conditions for the next contact time integration.

The main assumptions inherent in this implementation are:

- Isothermal operation. The inclusion of an energy balance is straightforward, provided that the temperature change is not strong to to the point of inducing a fundamental change in the hydrodynamics of the system.
- Dilute suspension, if the reaction product is a solid.
- Homogeneous bubbly flow, i.e. no bubble coalescence. This assumption allows the use of a mean bubble diameter and terminal velocity for the estimation of the contact time in the penetration model.

2.3.3 Interface equations

While the equations describing the bulk are being integrated along a contact time, mass transfer and chemical reactions take place at the interfacial elements, each one isolated from the others and the bulk network according to the penetration model concept. The latter assumes unsteady-state diffusion without convection on one spatial dimension (note that the notation of the compartmental model, such as indices referring to compartments and phases, does not apply to this section - the interfacial element is treated in isolation):

$$\frac{\partial C_i(x,t)}{\partial t} = D\frac{\partial^2 C_i(x,t)}{\partial x^2} + f_i(C_i(x,t)) \tag{2.13}$$

where $f_i(C_i(x,t))$ stands for the non-linear reaction terms in the balance of component i:

$$f_i(C_i(x,t)) = \sum_{j=1}^{J} r_j(C_i(x,t))$$

At the gas side of the interface ($x = 0$) the concentration of the absorbed component can be explicitly specified by an equilibrium relation such as Henry's law, while zero flux is the boundary condition for the non-volatile components:

Volatile species:

$$x = 0, t > 0 \Rightarrow C_i(x,t) = \frac{yP}{H} \tag{2.14}$$

Non-volatile species:

$$x = 0, t > 0 \Rightarrow \frac{dC_i(x,t)}{dx} = 0 \tag{2.15}$$

To impose boundary conditions at the bulk side we adopt the concept of Toor and Marchello (1958), according to which they can be specified at a fixed distance ($x = \delta$) from the interface, in contrast to the cumbersome ($x \rightarrow \infty$) proposed by the original Higbie model. This is valid on the condition that the concentration gradients are zero approaching the bulk at the given distance, certified by the following criterion:

$$\tau \leq \frac{\delta^2}{D} \tag{2.16}$$

For 1-10 mm bubbles with a terminal slip velocity of 20-30 cm/s this condition is easily satisfied with an interfacial length of 10-100μm. The boundary conditions

at the bulk side are the bulk concentrations at the beginning of the contact time, and therefore constitute the initial conditions as well:

$$x = \delta, t > 0 \Rightarrow C_i = C_{i,bulk} \tag{2.17}$$

$$t = 0, x > 0 \Rightarrow C_i = C_{i,bulk} \tag{2.18}$$

In the end of a contact time, the interfacial elements and the bulk cells are getting mixed. The concentrations that result from the mixing calculation constitute the initial conditions for integration of the bulk in the compartmental model, as well as the interface over the next contact time. The total amount of each component present in the interface is calculated from the concentration distribution at the end of the contact time as:

$$[C_i]_{tot}^{int} = \int_0^{\delta} C_i(x, \tau) dx \tag{2.19}$$

The initial conditions for the next contact time integration are then calculated as:

$$C_{iklp}^0 = \frac{[C_i]_{int}^{tot} + C_{iklp} \cdot V_{klp}}{V_{kl}^{int} + V_{klp}} \tag{2.20}$$

2.4 Computational implementation

2.4.1 Generalised compartmental model

In our implenentation, the set of ODEs (eqs. 2.7) that describe the bulk of each compartment are integrated with the LSODE set of subroutines, employing an Adams method. LSODE is available in the public domain via *www.netlib.org*, and is discussed by Hindmarsh (1983).

Successful tuning of the generalised compartmental model to the CFD requires, as it will be shown in the next chapter, consideration of many different structures for successsive refinement. Hence the success and usability of the method relies on an efficient computational implementation that facilitates the generation of structures as much as possible. In our code, all network characteristics are defined through the input files, in a comprehensive way that allows quick and easy modification. Each structure is checked for satisfaction of the mass balance (eq. 2.5) before proceeding.

The code has been programmed in FORTRAN 90, which allows elementary object oriented programming through the use of derived data types (structures). A

structure is constructed to describe the network compartments, while all properties associated with them (volumes, flowrates, concentrations etc.) are defined as elements of the structure, corresponding to the matrix notation employed here. Only a few seconds are required for simulating tracer experiments, as these involve just convection of material without reaction terms. Once the network has been tuned to the CFD results, the reactions can be superimposed. The runs with reaction are still very fast; about 30 compartments were employed for the case study presented here compared with thousands of CFD cells.

2.4.2 Solution of interfacial PDEs

Since the interfacial elements are isolated during the contact time, their equations describing them are not coupled, and therefore there is no need for an iterative solution. Each element can be treated individually, by integrating the system of two-dimensional non-linear parabolic PDEs that describe the interfacial phenomena. The discretised equations are here generated from integral forms of the differential equations, in a manner similar to the finite volume method method (Patankar, 1982). Functions of the nodal values may be used to approximate the variation of each term over the element, but eventually the solution constitutes of the nodal values only. The derivation of the discretised equation begins by integrating eq. (2.13) over an one-dimensional control volume:

$$\int_{x}^{x+dx} \frac{\partial C_i(x,t)}{\partial t} dx = \int_{x}^{x+dx} D \frac{\partial^2 C_i(x,t)}{\partial x^2} dx + \int_{x}^{x+dx} f(C_i(x,t)) dx \tag{2.21}$$

Diffusion terms are approximated with central differences. The grid and notation employed for the spatial and temporal discretisation is shown in fig. 2.4. Regarding the time integration, the following generalised implicit formula will be employed here:

$$\int_{t-1}^{t} C_{ip} dt = (aC_{ipt} + (1-a)C_{ipt-1})\,\delta t \tag{2.22}$$

This reduces to the fully implicit scheme for $\alpha = 1$, the explicit scheme for $\alpha =$

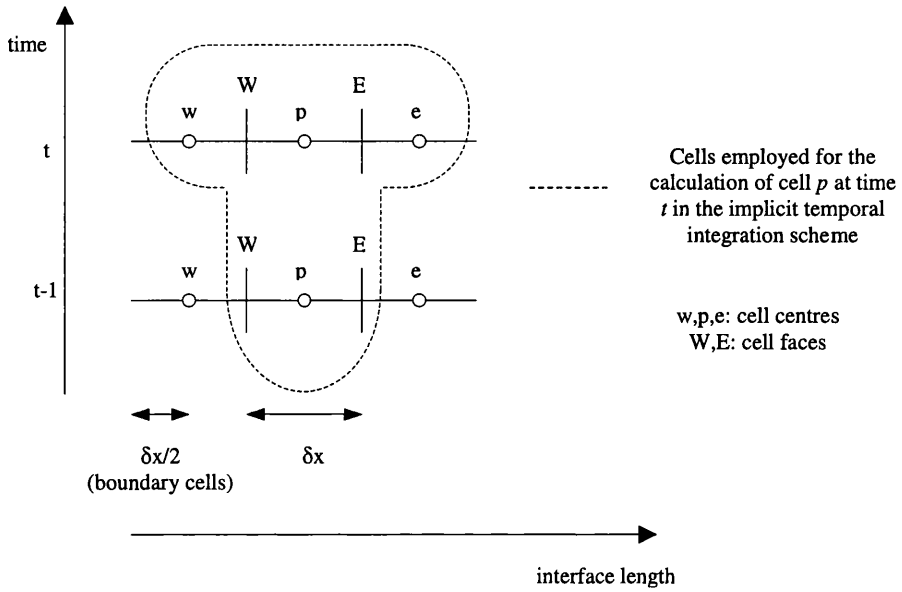

Figure 2.4: Notation used for the derivation of the discretised equations at the interface.

0 and the Crank-Nicholson for $\alpha = 0.5$. After some manipulation we obtain:

$$C_{i,p,t}\delta x - C_{i,p,t-1}\delta x =$$

$$D\frac{C_{i,e,t}}{\delta x}\delta t - \frac{2D}{\delta x}\delta t\left(aC_{i,p,t} + (1-a)C_{i,p,t-1}\right) + D\frac{C_{i,w,t}}{\delta x}\delta t + \int_{t-1}^{t}(f_{i,p}\delta x)dt \quad (2.23)$$

To proceed, the non-linear term f must be linearised. Newton's method is employed, with numerical calculation of the partial derivatives by perturbating f around C_i at the previous time point:

$$\left.\frac{\partial f_{i,p}}{\partial C_i}\right|_{t-1} = \left.\frac{f(C_i) - f(C_i + \delta C_i)}{\delta C_i}\right|_{t-1} \quad (2.24)$$

See Appendix A for more details on the linearisation scheme and the calculation of the coefficients.

2.4.3 CFD modelling of vertical dispersed gas-liquid flows

The CFD model was implemented using the CFX code (version 4.3, AEA Technology, Harwell, UK), which solves the multiphase Eulerian-Eulerian model with the IPSA (Inter-Phase Slip Algorithm - Spalding, 1980). Since the Eulerian-Eulerian model will be adopted in this study, it will now be considered in more detail. This

section will discuss several currently unresolved or debatable issues associated with its application, with relevance to the objectives of the present study.

2.4.3.1 Grid and time stepping

Several computational studies have compared transient with steady-state and two-dimensional with three-dimensional simulations of bubble columns (e.g. Sokolichin and Eigenberger, 1994; Jakobsen et al., 1997). It was found that only three-dimensional transient simulations are able of revealing the chaotic nature of such flows. In this study, though, CFD simulation is employed as a tool for estimating essentially average values of hydrodynamic quantities, and forms part of a tool intended for design and scale-up of equipment within a reasonable amount of computational time. Therefore the approach adopted here is a two-dimensional dynamic model that can be resolved in an ordinary workstation within a couple of days. The steady-state approach was also considered, but was found to be unstable and exhibit severe problems with mass conservation, at least in our solver (CFX 4.3). This is to be expected since several of the phenomena involved are highly dynamic, in particular the introduction of gas that causes an expansion of the column during the first seconds, and the accumulation of bubbles in the downcomer. A simulation of about 10 real-time seconds was found to be adequate for the main flow features to stabilise. The time stepping policy was to initiate with very small time steps (0.001 - 0.005 sec), and gradually switch to 0.01 sec once the main features of the flow had been established. Convergence was evaluated on the basis of the residuals, and especially of the continuity equation residual, as compared to the total mass in the reactor. It was ensured that the residual per time step would be less than 0.01% of the total mass.

A bubble column is a cylindrical axisymmetric geometry but it is well known that the motion of rising bubbles is swirling, and this can only be captured with a 3-D model (Sokolichin and Eigenberger, 1995). As in this study CFD is used to predict averaged quantities of gas hold-up and liquid flow, phase distribution along the cross-section is of little interest and 2-D modelling is considered to be sufficient. This also rules out the need for including non-drag forces that give rise to the swirling motion. Regarding the grid, Eulerian-Eulerian gas-liquid flow simulations have an inherent

limitation with respect to grid refinement: if the cells are similar to or smaller than the bubbles, the model loses its physical basis (Jakobsen et al., 1997; Sokolichin and Eigenberger, 1994). Therefore grids with with cells having dimensions in the order of 1 - 0.5 cm were investigated. Further refinement did not have significant effect, while the physical validity of the model would be challenged. Thus a grid of 0.5 cm cells was considered sufficiently fine, given that some flow details such as swirling motion would be lost anyway due to the two-dimensional reduction.

2.4.3.2 Boundary conditions

Standard no-slip conditions are imposed on the column walls and draft tube. The introduction and removal of the gas phase is a problem, however. An exact specification of gas inlet velocity would imply that its flow is fully developed at the sparger holes, which is clearly unrealistic. Moreover, the sparger holes are extremely small (0.5 - 1mm) and the grid is not fine enough to resolve them. Therefore, it is considered preferable to insert the gas as a source in the continuity equation at the cells next to the boundary. No momentum source was applied, as jetting is unlikely to occur at low superficial velocities and homogeneous flow.

To remove the gas, CFX 4.3 suggests the use of a "degassing" boundary condition, which means to model the free surface as a solid wall and remove via a sink any gas that reaches that point with an upward axial velocity. We consider this approach unrealistic, however, because it is an artificial way to model the free surface and, most important, because it does not reflect the initial expansion of the water column. As gas is being introduced, water level raises, and this can cause problems in the continuity equation if the free surface is modelled as a wall. This may be mitigated by introducing some dispersed gas as an initial condition, but that would require an a priori knowledge of the gas holdup. The approach adopted in this study is to leave some space filled with gas above the water column in the initial conditions for the volume fraction, and let the free surface expand as the gas is being introduced. Of course, the exact shape of the free surface cannot be precisely simulated with a two-fluid model, but it is of no relevance here. The boundary condition is imposed on the gas end of the column, and sets the pressure there equal to atmospheric. Thus the inserted gas is left to exit naturally, by increasing the pressure above the

column. The validity of the approach can be easily checked by a mass balance between the inlet gas flowrate and the amount that exits from the pressure boundary - in the simulations shown here these were approximately the same.

2.4.3.3 Additional phenomenological models

Another important issue is the turbulence modelling: the applicability of the $k-\epsilon$ model to multi-phase flows has been the subject of several studies (Schwarz and Turner, 1998; Pfleger et al., 1999; Sokolichin and Eigenberger, 1999). They seem to conclude that a combination of a $k-\epsilon$ model to describe the liquid phase with a laminar gas phase yields results closer to experimental evidence than other tested combinations (laminar-laminar, $k-\epsilon$ - $k-\epsilon$.). Following this, a standard $k-\epsilon$ model is applied here to the liquid phase only. It must be noted that turbulence modelling in multi-phase models is far from definitive, as the $k-\epsilon$ model was initially developed for single-phase flows. For this reason, many authors supplement it with additional models for the turbulent dispersion of the gas phase, which however introduce more unknown parameters. In this work, however, the simplest approach was chosen.

Two a priori assumptions must be made to use the Eulerian-Eulerian multiphase model. Regarding the bubble size, only a mean value is being used (in accordance to the reaction engineering model), and it can be estimated empirically. The second assumption concerns the drag force, which is responsible for the coupling between the equations either phase. The drag force for a bubble swarm is given by eq. 1.16. Correlations for determining the drag force coefficient were reviewed in chapter 1. Among them, the Schwarz and Turner approach (eq. 1.19, used with comparative success in several studies of vertical bubbly flows such as Sokolichin and Eigenberger (1994), is adopted for the purposes of this study.

2.5 Case study: Parallel-consecutive reactions in a draft-tube bubble column

In the example that follows, the usage of this methodology will be demonstrated via a gas-liquid reaction system (absorption of CO_2 into alkali solution) taking place in a draft tube bubble column. It must be stressed, however, that the particular

CFD implementation described below does not constitute a part of the methodology; in principle any CFD model can be used in conjunction with the generalised compartmental model.

2.5.1 Validation of CFD model

Before integrating with the reaction engineering model, the hydrodynamic predictions of the CFD simulation will be compared with the experimental data of Wachi, Jones and Elson (1991). In that work gas hold-up in both riser and downcomer sections and mean liquid circulation were measured in an internal draft tube bubble column (fig. 2.5), using the piezometric and hot pulse injection methods respectively. The column was 2.5 m high, 0.22 m inside diameter and 85 l in capacity, and gas was injected via a perforated plate in the bottom of the inner annulus (riser). Different liquids, gas flowrates and internal draft tubes were tested in that work, but the data presented here for comparison are for the air-water system and a 0.12 m diameter draft tube. Fig. 2.6 shows a comparison between experimental and simulated gas hold-up at five different flowrates.

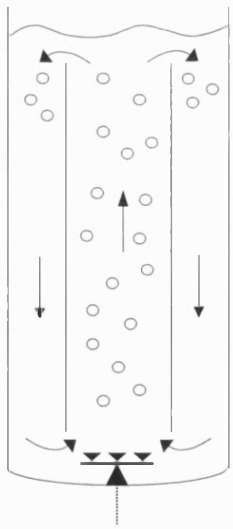

Figure 2.5: Schematic of a draft-tube bubble column.

The shortcomings of the Eulerian-Eulerian approach with respect to predicting flow details and radial phase distribution) have been pointed out by many authors, such as Jakobsen (2000). In this framework, however, CFD is employed for calculating averaged properties, and in this context the model seems to be adequate. At

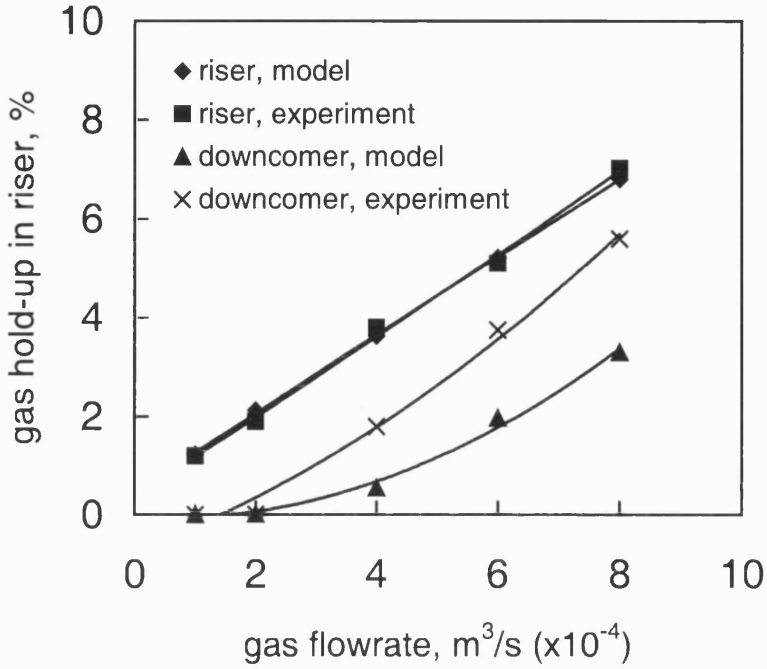

Figure 2.6: Gas hold-up in the riser and downcomer, comparison of CFD model with the experiments of Wachi et al. (1991).

the riser, agreement between CFD and experiment is excellent and may be considered quantitative at the low gas flowrates tested, which explore the whole range of homogeneous bubbly flow. A slight deviation at the highest flowrates may indicate the beginning of the transition to the heterogeneous regime. Gas entrapment in the downcomer is, however, underpredicted, a fact possibly related to the turbulence modelling which, for multiphase flow, is still at a primitive stage. The model would be more accurate if applied to external loop reactors, where gas is totally separated at the top. In this case, however, holdup in the riser is more important, as the entrapped bubbles in the downcomer are soon depleted from the reactant component and play no important role in the reaction engineering.

The compartmentalisation will be carried out for the steady state. For this reason, the concentration difference between two points in the column (one in the riser and one in the downcomer) was monitored until a pseudo-steady state was reached. The virtual tracer experiment was then carried out in a manner resembling

the feeding policy of bubble columns, eq. 2.10.

2.5.2 Network structure and mean flowrates

Fig. 2.7 shows the the flow field, tracer concentration and volume fraction profiles. Note that the column head is occupied by gas, as it can be seen in the volume fraction contour plot, and therefore the tracer concentration in that region is of no importance. Fig. 2.8 shows radial concentration profiles in the riser at different axial points. By inspecting these results, we draw the following conclusions regarding the structure of the compartmental model:

- The liquid phase flow field is dominated by a circulation loop, characteristic of airlift reactors, due to the differential gas holdup between the riser and downcomer. This loop is the main macromixing mechanism and must be accounted for by the structure of the network.
- The downcomer resembles a plug flow, with no observable gradient in the lateral direction. Therefore there is no need for lateral compartments in that region.
- The volume fraction plot (fig. 2.8) shows the gas to be concentrated in the middle of the inner annulus and reduce towards the wall. Consequently the reactant is fed mostly throughout the central region of the riser, and disperses towards the walls mainly through turbulent mixing. As a result, a lateral concentration gradient is generated in the riser.

The change in the direction of the flow, from upward in the riser to downward in the downcomer, suggests the use of two vertical layers of cells for the riser and downcomer, respectively. The fact that the gas holdup is concentrated in the middle of the inner annulus suggests that the riser be further splitted into 2 vertical layers, one for the gas-rich and one for the gas-poor region. Since the reactant is introduced to the bulk from wherever gas is present, this way we emulate the feeding policy of the reactor. Based on the above, a network comprising 10 horizontal and 3 vertical layers of cells is constructed. The cells are initially linked with flowrates reflecting only the convective mean flow, averaged along the cell boundaries. This network is depicted in fig. 2.9.

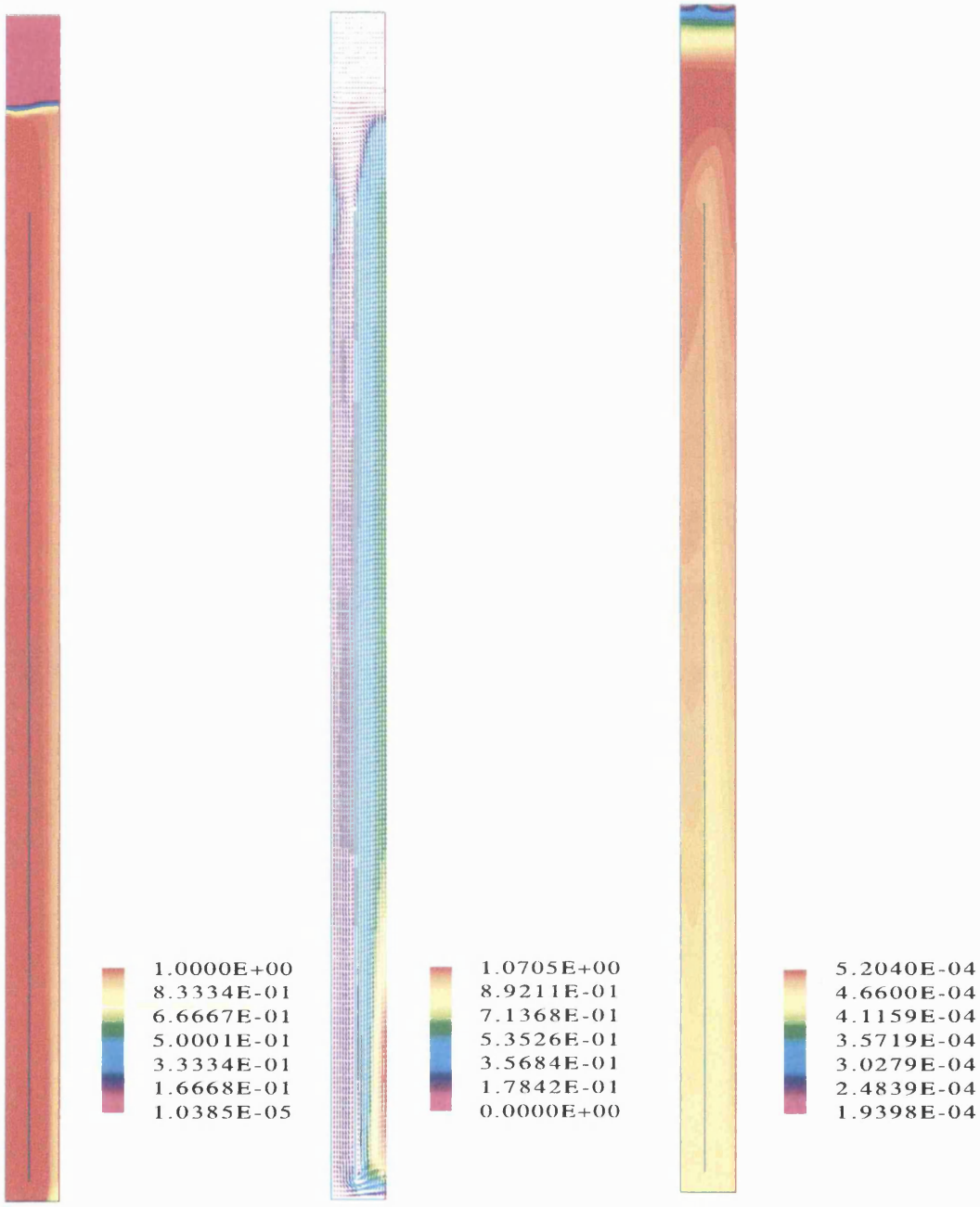

Figure 2.7: CFD results (left to right): Volume fraction, mean velocity profile (m/sec) and dimensionless scalar concentration (resulting from the virtual tracer experiment), axisymmetric simulation, column axis on the right.

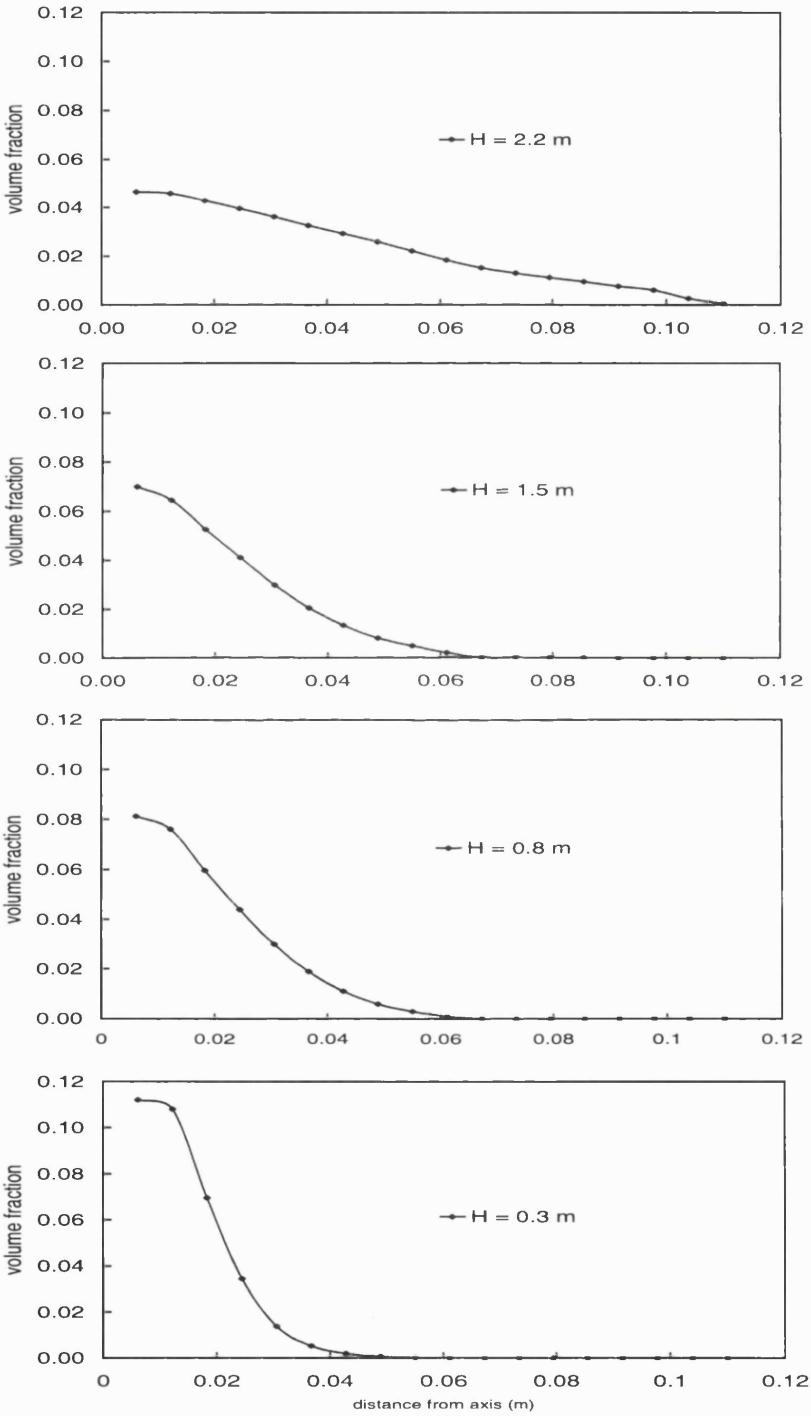

Figure 2.8: Radial gas holdup profiles in the column at four axial locations.

2.5.3 Turbulent exchange

Figs.2.10,2.11 show the axial concentration profiles in the two riser layers and in the downcomer. It is evident that this network yields a rather poor description of the riser concentrations, due to the lack of direct material exchange between the two vertical layers of riser compartments. The network is thus refined to the one shown in fig. 2.12, featuring lateral turbulent exchange. Agreement between the network and CFD is significantly improved, as shown by figs. 2.10,2.11. The discrepancy is now in the order of $1-4\%$, and is deemed acceptable for the purpose of this study. We are now ready to proceed with the superposition of the chemical phenomena.

2.5.4 Superposition of chemical reactions

Assume the following system of mass transfer accompanied by a system of parallel - consecutive reactions:

$$\begin{aligned} &CO_{2(aq)} + OH^- \rightleftharpoons HCO_3^- \quad [i] \\ &HCO_3^- + OH^- \rightleftharpoons CO_3^= \quad [ii] \end{aligned} \tag{2.25}$$

Kinetics and equilibrium parameters are taken from Astarita (1967) and are shown in table 2.1. Fig. 2.15 shows the molar fractions of $CO_{2(aq)}$, HCO_3^-, $CO_3^=$ at equilibrium conditions, obtained by for different values of pH in the range 7-13. At pH=12-13 the equilibrium is totally shifted towards $CO_3^=$; below pH=11, HCO_3^- exists in significant amounts, and around pH=10 its coexists with $CO_3^=$ at equal concentrations, while at pH=9 the molar fraction of $CO_3^=$ is less than 0.1. As pH approaches the neutral range, $CO_{2(aq)}$ dominates.

Parameter, units	Value	Source
Equilibrium constant of sub-reaction i, K_i, m^3 mol^{-1}	$6.1 \cdot 10^4$	Astarita, 1967
Equilibrium constant of sub-reaction ii, K_{ii}, m^3 mol^{-1}	5.88	>>
Kinetic constant of sub-reaction i, k_i, mol^{-1} m^3 s^{-1}	5.0	>>
Kinetic constant of sub-reaction ii, $k_i i$, mol^{-1} m^3 s^{-1}	$1 \cdot 10^4$	>>
Henry's constant, H, atm m^3 mol^{-1}	0.0346	>>

Table 2.1: Parameter values and their sources.

Fig. 2.16 shows a sample of the concentration profiles in the interface, as resulting by the application of the penetration model. Figs.2.17,2.18 show the time course of

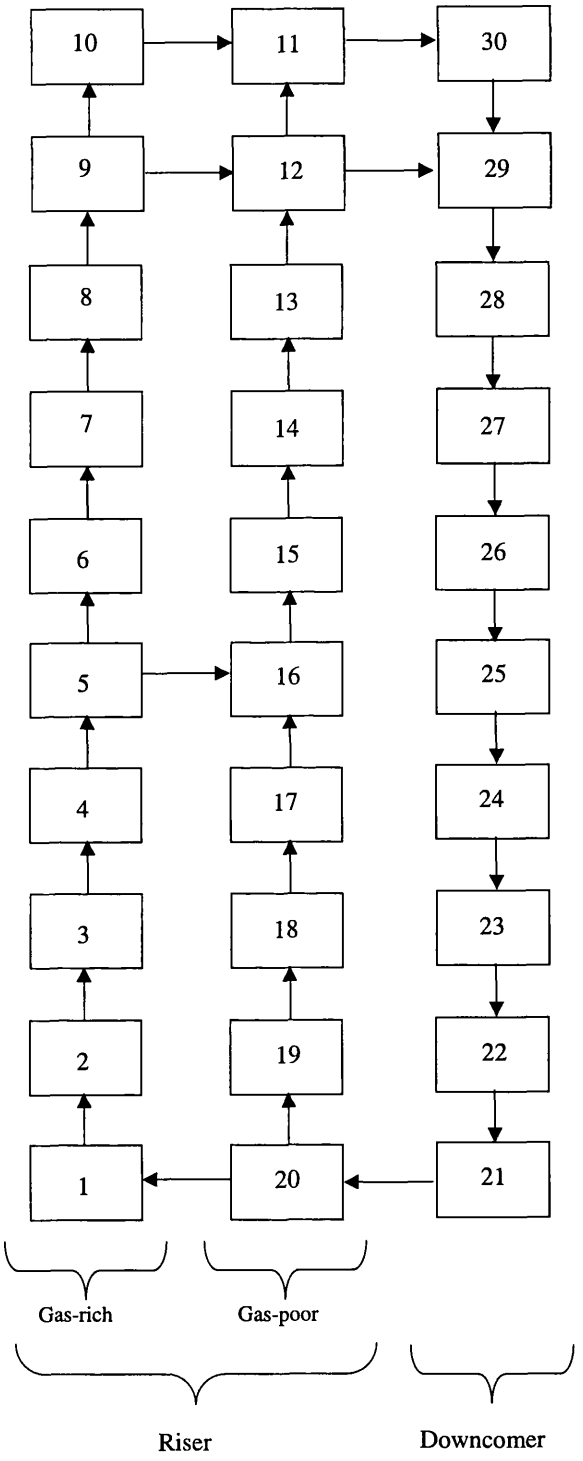

Figure 2.9: Compartmental network with two vertical layers of riser cells, to emulate the feeding policy of a bubble column

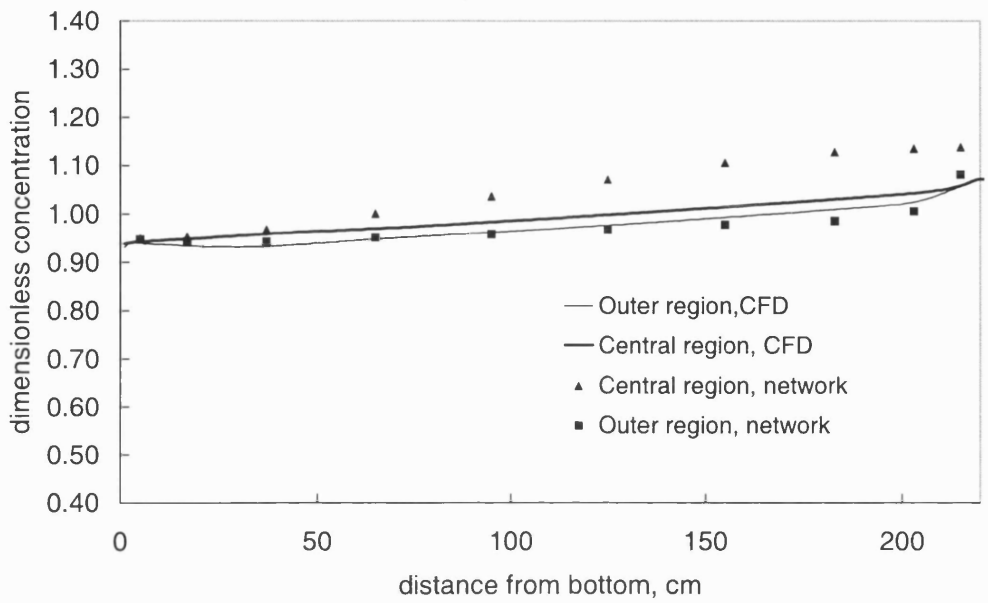

Figure 2.10: Concentration profiles in the riser, network of fig. 2.9.

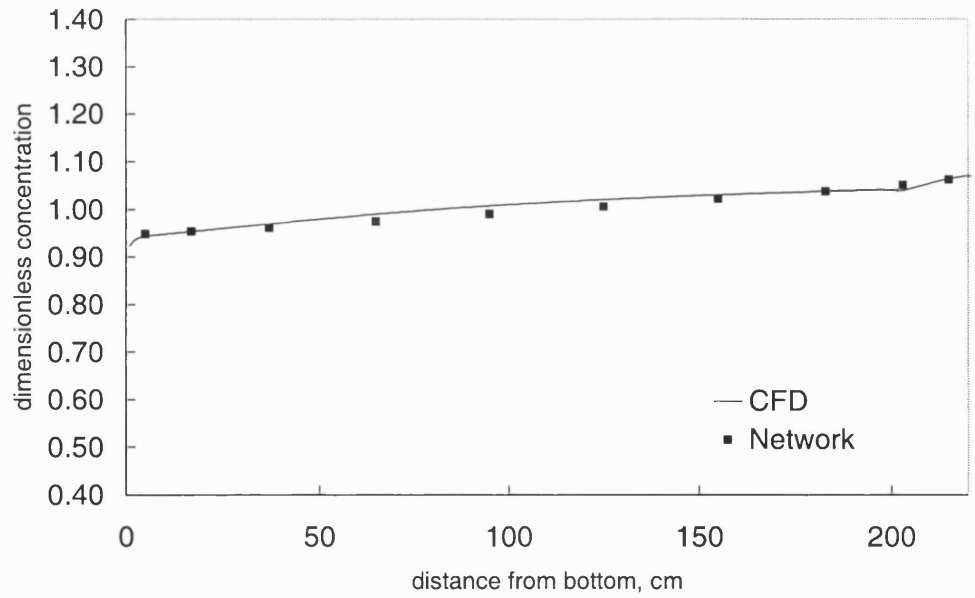

Figure 2.11: Concentration profiles in the downcomer, network of fig. 2.9.

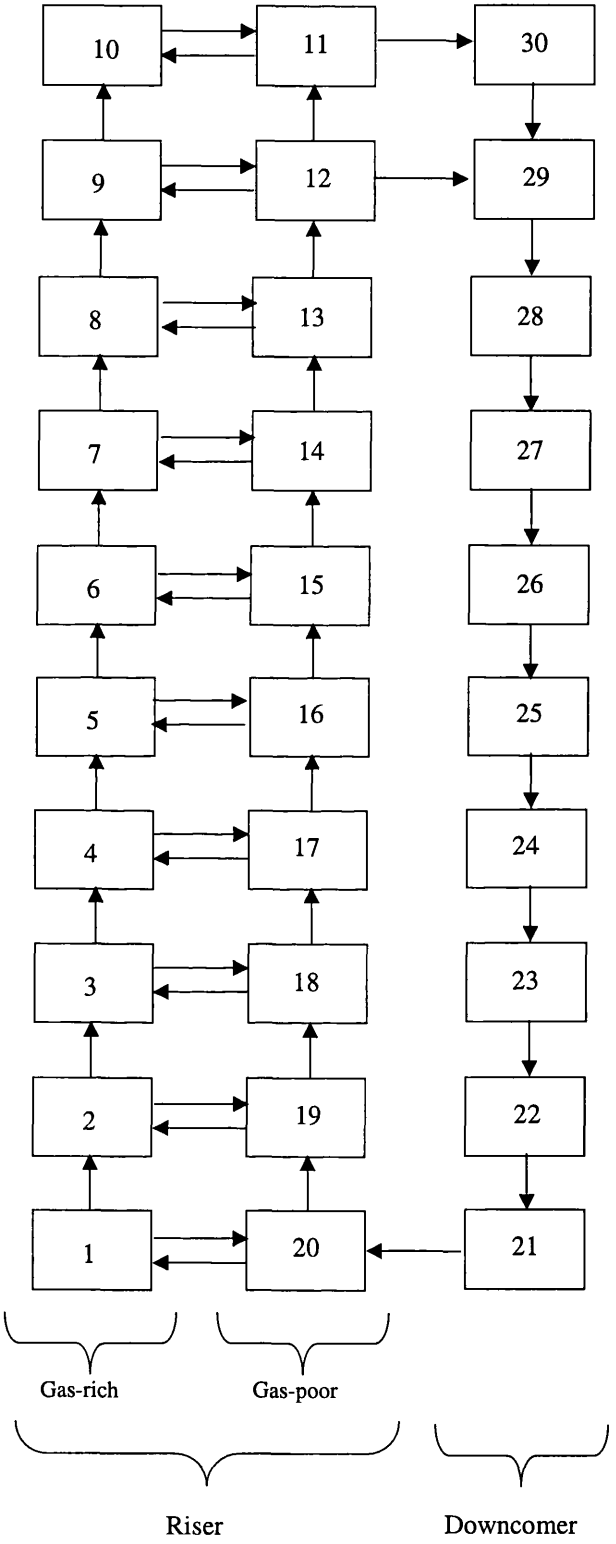

Figure 2.12: The network of 2.9 is improved to account for lateral turbulent exchange in the riser.

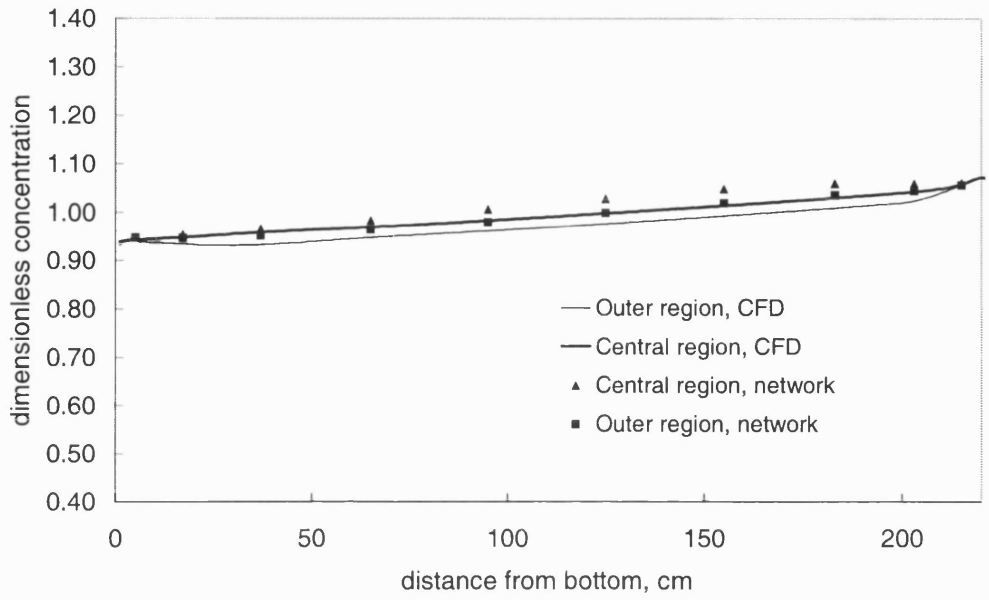

Figure 2.13: Concentration profiles in the riser, network of fig. 2.12.

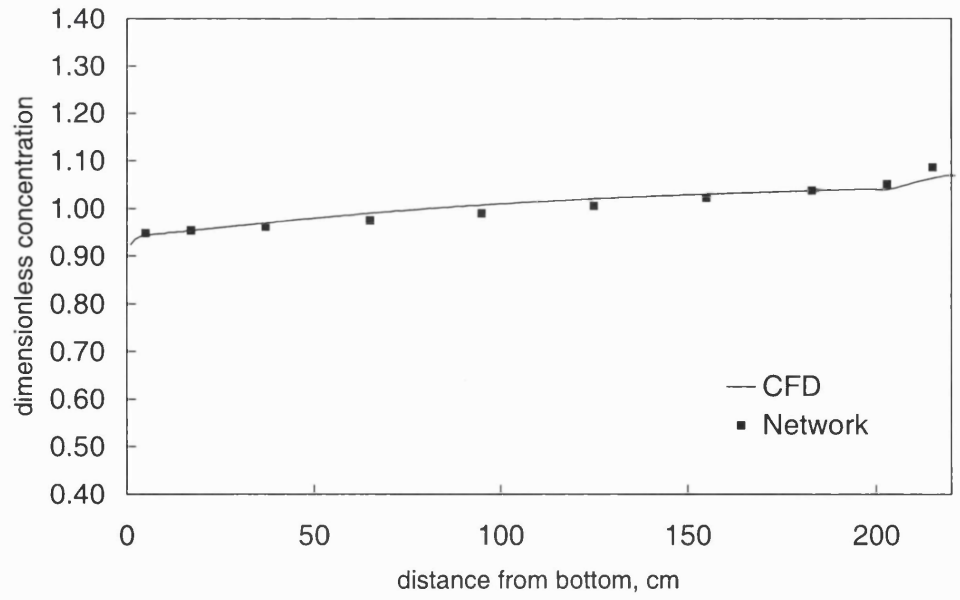

Figure 2.14: Concentration profiles in the downcomer, network of fig. 2.12.

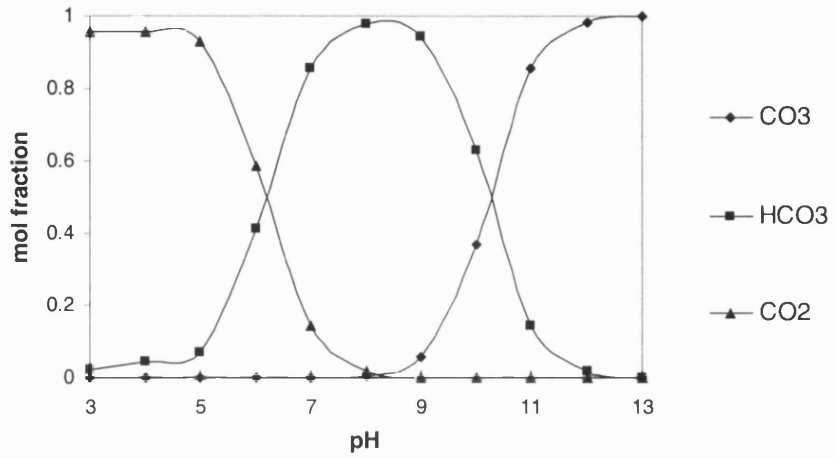

Figure 2.15: Equilibrium concentrations at different values of pH for the system CO_2 - HCO_3^- - $CO_3^=$

the concentration and pH profiles, averaged over the whole column. In the beginning of the process almost all of the absorbed gas is converted to $CO_3 =$. The increase in $CO_3^=$ concentration soon reaches a peak however, and is subsequently being reduced, while HCO_3^- starts to increase. This event is reflected in the pH curve by a change in slope, and can be explained by the shift of equilibrium of reaction (ii) in favour of HCO_3^- occurring around pH 11. At lower pH values the equilibrium of reaction (i) shifts towards CO_2, and another change in the pH slope is observed. After this point, chemical reactions become negligible and the effect is that of physical absorption. Later absorption stops and pH becomes constant, as there is no driving force due to the high bulk concentration of dissolved CO_2. The shape of the pH curve is in agreement with the experiments of Fleischer et al. (1996).

In fig. 2.19 we see a comparison of the total amount of CO_2 absorbed in the end of the chemical process to the amount that would have been absorbed during physical absorption. It is much higher in the chemical process, partly but not only due to the enhancement of mass transfer by the reaction at the interface. In the chemical process, the bulk conversion of CO_2 to $CO_3^=$ and HCO_3^- keeps the concentration of dissolved CO_2 in the bulk close to zero, thus maintaining the mass transfer driving force at high levels, until the time of the pH drop. On the other hand, the physical

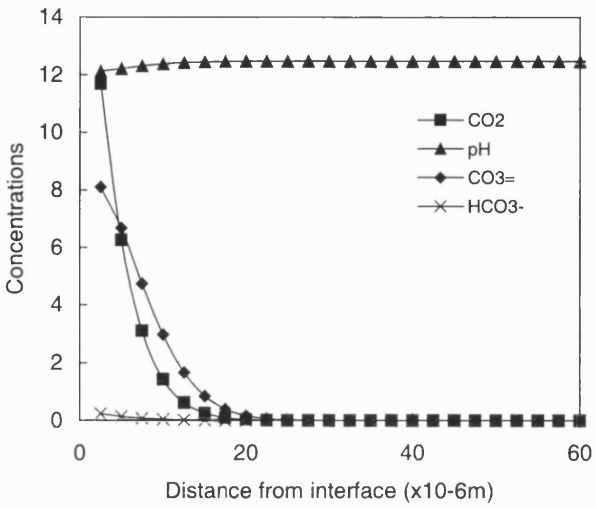

Figure 2.16: A sample of simulated concentration profiles in the interface

absorption process is sooner halted due to the accumulation of CO_2 in the bulk. This example illustrates the need for rigorous dynamic modelling in the analysis of batch absorption processes.

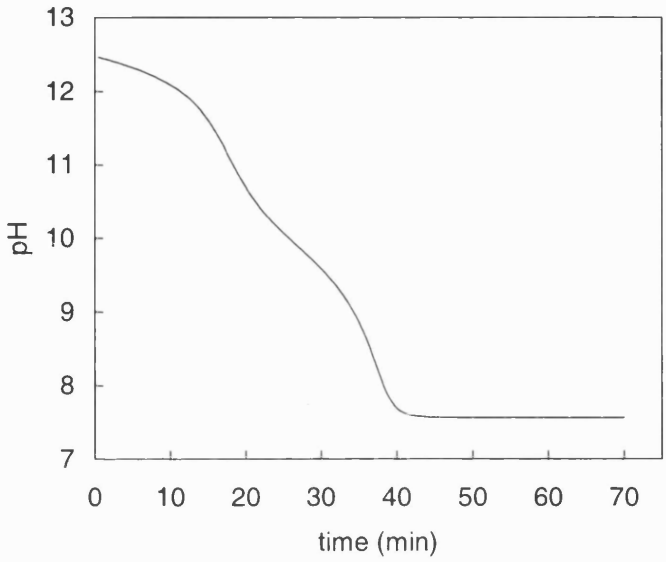

Figure 2.17: Evolution of pH.

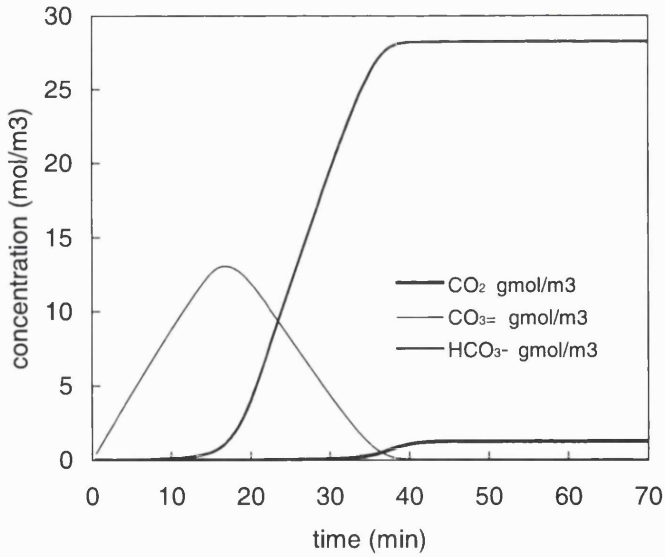

Figure 2.18: Evolution of chemical species' concentrations.

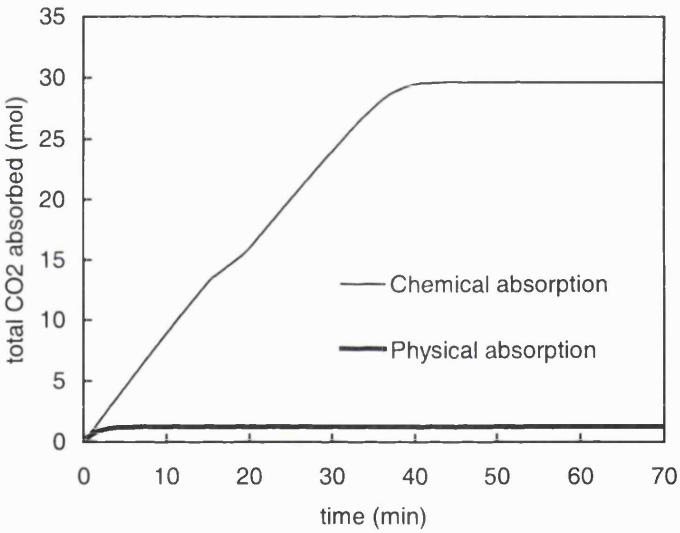

Figure 2.19: Physical and chemical absorption.

Chapter 3

Finite element scheme for the time-dependent population balance

3.1 The need for a new approach

The population balance equation (PBE) lies at the core of particulate process modelling, yet its numerical solution continues to pose severe challenges. In chapter 1 the existing numerical methods for solving the population balance were reviewed and classified into two categories: methods based on a function approximation of the solution (e.g. weighted residual, finite element) and methods that divide the domain of the independent variable into classes, commonly referred to as DPB (Discretised Population Balances). The main conclusions from the review can be summarised in the following:

Discretised population balances:

- are custom-derived methods tailored to solve individual problems efficiently but lack generality,
- are praised mainly for their ease of use and computational efficiency, and
- are engineered to predict some moments (usually one or two) exactly, but other moments may exhibit severe errors, as may do their reproduction of the distribution.

Function approximation methods:

- are a general and very flexible class of numerical methods and as such, when applied to the PBE, they can easily cope with any possible formulation, e.g. size-dependent growth, different agglomeration kernels etc.,
- retrieve the entire distribution with detail,
- are more difficult to implement and more computationally demanding, but
- their prediction of any moment is subject to errors.

In light of these conclusions, in this work we set out to derive a finite element scheme that will exhibit the following features:

Generality. The dynamic PBE with nucleation, growth, dispersion, aggregation and breakage will be considered, the only simplifications being spatial uniformity and one-dimensional domain of the particulate variable. The individual numerical needs of each one of these processes will be identified, and suggestions will be made for configuring the finite element scheme to optimally accommodate them.

Stability. To achieve this we shall employ linear trial functions, and collocate them in a way that reflects the convective nature of the growth term. This way, even problems with fronts such as the dynamic nucleation case can be dealt with accurately.

Conservation of moments. A good numerical method must be conservative, if inaccurate, even in a coarse grid. The schemes used in solid and fluid dynamics usually possess this property, but in the PBE conservation is not so straightforward due to presence of the integral terms. The possible sources of error will be identified, and ways of mitigating them will be proposed.

Ease of implementation. Owing to the use of linear trial functions, the resulting scheme is very simple and easy to incorporate in a computer code.

Balance of accuracy and speed. As it will be demonstrated by the test cases, very accurate results can be achieved within reasonable computational time,

even in desktop computers. The lower accuracy of the linear trial functions is compensated by the use of a finer grid.

3.2 Formulation of the PBE

In chapter 1, a general form of the population balance was written as:

$$\frac{\partial n(x_i,t)}{\partial t} = -\frac{\partial(u_i \cdot n(x_i,t))}{\partial x_i} + D\frac{\partial^2 n(x_i,t)}{\partial x_i^2} + \\ + B_{nuc} + B_{agg} - D_{agg} + B_{br} - D_{br} \quad (3.1)$$

The independent variables x_i in the PBE are of two types:

1. The dimensions of physical space, $\{x_i, i = 1,2,3\}$.
2. The properties of the population whose distribution we wish to calculate.

These two types are sometimes called external and internal coordinates (Randolph and Larson, 1988). The first issue to be discussed before proceeding with the formulation is the choice of internal variable. In crystallisation, it is a measure of particle size. Previous work (Gelbard and Seinfeld, 1978; Hounslow et al., 1988; Randolph and Larson, 1988;) has often favoured the use of particle diameter, due to the fact that it is the variable most commonly measured by particle sizing equipment. Moreover, certain particulate processes such as crystallisation often exhibit size independent growth, which facilitates the computation when a linear measure of size is used as independent variable. In this work, however, we have chosen to formulate the PBE in terms of particle volume in order to emphasise conservation through correct discretisation of aggregation and breakage terms. In these two processes volume is being conserved, and therefore it is the natural choice of an independent variable in the PBE. Furthermore, if particle diameter were chosen, the expression for the aggregation would include shape factors that are system-dependent. In what follows x_i will refer to the spatial coordinates ($i = 1,2,3$), while v to particle volume.

A spatially distributed PBE implies coupling with fluid dynamics. Such a coupling raises extreme computational demands and has only been accomplished through oversimplification of the PBE (see chapter 1). For this reason, the PBE will

be formulated for a spatially uniform system, by integrating eq. 1.23 over a uniform finite domain:

$$\frac{1}{V}\iiint_V \left[\frac{\partial n(v,t)}{\partial t} + \frac{\partial}{\partial x_i}[u_i \cdot n(x_i,t)] - D\frac{\partial^2}{\partial x_i^2}[n(x_i,t)] - \right.$$
$$\left. - B_{nuc} - B_{agg} + D_{agg} - B_{br} + D_{br}\right] dV = 0 \quad (3.2)$$

The velocity with respect to the internal dimension (volume) is in fact the growth rate. Thus the convection term can be split into differentiation along physical space and differentiation along the volume coordinate. The physical interpretation of the former is the change in the distribution due to inflow and outflow of particles with different distributions, while the latter denotes change due to growth.

$$\frac{\partial}{\partial x_i}[u_i \cdot n(v,t)] = \frac{\partial}{\partial x_i}[u_i \cdot n(v,t)] + \frac{\partial}{\partial v}[G(v,t) \cdot n(v,t)] \quad (3.3)$$

The integration over a spatially uniform domain reduces convection in physical space into an input - output term:

$$\frac{1}{V}\iiint_V \frac{\partial}{\partial x_i}[u_i \cdot n(v,t)]\, dV = \frac{n_{in}(v,t) - n(v,t)}{\tau} \quad (3.4)$$

where $n_{in}(v,t)$ is a known function denoting the input distribution.

Similarly, the diffusion term can be divided into diffusion in physical space and diffusion with respect to particle volume. The former can be neglected, since we are considering a spatially uniform domain. The latter is occasionally employed to account for growth rate dispersion, a phenomenon sometimes observed in crystallisation processes, where particles of the same size exhibit fluctuations in their growth rate. This phenomenon, occurring mainly in crystallisation, is attributed to the following reasons (Randolph and White, 1974): a) spatial effects (e.g. non-uniform supersaturation field, size classification due to flow field) and b) variations in particle structure. Diffusional transport implies the transport of a property due to the presence of gradients, but rather than this, it is clear that the fluctuations mentioned above occur due to another variable not being included in the model. Indeed, spatial effects can be taken into account by coupling PBE with fluid dynamics, while variations in particle structure can be described by extra internal coordinates. Diffusion will not be included in our formulation, but a note will be made on how it could be accounted for if required.

The remaining terms are not functions of the spatial coordinates, and therefore remain intact after the spatial integration. Nucleation is the appearence of particles at certain size (v_0), and is incorporated via a Dirac delta function. This means, however, that a sharp change in the profile will occur around the nuclei size, and the numerical method must be very stable to capture it.

$$B_{nuc} = B_0\delta(v - v_0) \tag{3.5}$$

Aggregation is described by the continuous Smoluchowski equation (see chapter 1). The birth term denotes generation of particles of size v due to aggregation of any combination of smaller particles:

$$B_{agg} = \frac{1}{2}\int_0^v \beta_a(v', v - v')n(v', t)n(v - v', t)dv'dv' \tag{3.6}$$

The meaning of the aggregation kernel has been explained in chapter 1. The death term represents the disappearence of particles of size v due to their aggregation with any other particle:

$$n(v, t)\int_0^\infty \beta_a(v, v')n(v', t)dv' \tag{3.7}$$

The breakage birth term denotes the rate at which particles of size v are being generated due to breakage of any particle $v' > v$. Thus integration from v to ∞ is required:

$$B_{br} = \int_v^\infty \beta_b N_p(v')b(v/v')n(v')dv' \tag{3.8}$$

The term in the integral has to account for three things:

1. The rate at which breakage events occur. This is accommodated by the breakage kernel: $\beta(v)$.

2. The number of daughter particles generated in each breakage event (degree of fragmentation), $N_p(v)$.

3. The size distribution of these particles, represented by the function $b(v/v')$

The number of daughter particles and their size distribution are accommodated by the functions k(v') and *b(v/ v')* respectively, and are subject to the following

constraints (Ramkrishna, 1985):

$$b(v/v') = 0 \; \forall \; v > v' \tag{3.9}$$

$$\int_0^{v'} b(v/v')dv = 1 \tag{3.10}$$

$$k(v') \int_0^{v'} vb(v/v')dv = 1 \tag{3.11}$$

The breakage death term is the rate at which particles of size v break into fragments of smaller size, and is represented simply by a polynomial term:

$$D_{br} = -\beta_b(v)n(v) \tag{3.12}$$

Thus we arrive at the form of the PBE we shall deal with:

$$\begin{aligned} \frac{\partial n(v,t)}{\partial t} = \frac{n_{in}(v,t) - n(v,t)}{\tau} - \frac{\partial}{\partial v}\left[G(v) \cdot n(v,t)\right] + B_0\delta(v - v_0) + \\ + \frac{1}{2}\int_0^v \beta_a(v - v', v')n(v - v', t)n(v', t)dv' - n(v,t)\int_0^\infty \beta_a(v, v')n(v', t)dv' + \\ + \int_v^\infty \beta_b N_p(v')b(v/v')n(v')dv' - \beta_b(v)n(v) \end{aligned} \tag{3.13}$$

It remains to define the boundary conditions. Eq. 3.13 is a first-order hyperbolic differential equation, and as such requires two initial conditions. The initial condition for the time domain is straightforward:

$$n(v, 0) = n_0(v) \tag{3.14}$$

With respect to the volume coordinate, the most physically realistic choice is to state that there are no particles of zero volume:

$$n(0, t) = 0 \tag{3.15}$$

The volume domain must then start from zero (rather than nuclei volume). Randolph and Larson (1988) proposed instead the condition $n(v_0, t) = B_0/G$ for nucleation - growth problems, which implies that the number of nuclei is determined by a balance between nucleation and growth. This condition is not valid for aggregation problems, because in that case nuclei may also be reduced via the aggregation mechanism, and moreover it implies that nuclei are of zero size, which is not physically realistic (Rojkowski, 1990).

3.3 Preliminary considerations

3.3.1 Truncation of the domain

Since the domain of the volume variable is infinite, to carry out computations it must be truncated to a finite value. This implies that the tail of the distribution is being cut off, and the $i-th$ moment is underpredicted by the amount:

$$\int_{v_{tr}}^{\infty} v^i n(v,t)dv \tag{3.16}$$

Clearly the underprediction is more significant the higher the order of the moment. Gelbard and Seinfeld (1978) provide analytical expressions of this error for cases that have an analytical solution, while Nicmanis and Hounslow (1998) provide a systematic procedure for selecting the upper limit in steady-state PBE, based on the second moment. To select the upper limit for a dynamic simulation, however, one needs insight into the effects of the truncation:

- In aggregation problems, particles forming aggregates bigger than the truncation point fall out of the computational domain, thus causing the moments to reduce. At the same time aggregation of particles from the domain with particles from the excluded tail is not being considered, which has a positive effect on the moments. The prevailing effect is a negative one, however, and in a pure aggregation problem the truncation error is easily spotted by a reduction in the first moment.

- In combined aggregation - breakage problems the effect of truncation is more difficult to estimate, as aggregates from the neglected tail can also break into particles within the computational domain.

- In growth problems the values of the population density that lie within the computational domain are not affected by the excluded tail. However, the whole body of distribution is moving towards bigger sizes, and once it reaches the truncation point the solution is clearly of no value.

Based on these remarks, we can deduce some heuristics for truncating the domain. For a dynamic simulation, the best practice is to use the moments to monitor

the solution and determine whether the truncation has any effect. Aggregation - breakage problems may be monitored as above on the basis of the first moment. In growth-dominated problems, the values of the population density that lie within the computational domain are not affected by the excluded tail. Since the whole body of distribution is moving towards bigger sizes, however, once it reaches the truncation point the solution is clearly of no value. Usually the size range to be covered can be estimated beforehand from the growth kinetics.

3.3.2 Grid generation

The choice of grid is very important, as it has profound effect on both accuracy and conservation of the scheme. It will be shown later that uniform meshes are more conservative; however, they cannot provide adequate accuracy, except for volume-independent growth problems (a rather unrealistic case) where the distribution moves with a constant speed. In aggregation and volume-dependent growth problems very big particles can be rapidly created, meaning that a long range of volume must be covered, while nucleation creates a large number of particles in the smaller size ranges, thus giving rise to steep gradients that require a locally refined grid. . Non-uniform meshes can suffer from errors in the conservation of moments, however, and it is the intention of this work to provide a more conservative algorithm. Most of the test cases to be presented here have been solved on an exponential grid generated with the following series:

$$v_i = v_{\min} + v_{\min} \cdot \frac{1-\alpha^{i-1}}{1-\alpha} \tag{3.17}$$

The volume v_{min}should correspond to the smallest particles present, i.e. the nuclei. By varying the coefficients v_0 an α, one can generate grids with an increasing degree of coarseness. A preliminary criterion to facilitate conservation in non-uniform meshes is:

$$v_i \leq 2v_{i-1} \tag{3.18}$$

This is because the biggest particle that can be formed by particles of sizes up to v_{i-1} is $2v_{i-1}$, hence if the next mesh point is bigger than $2v_{i-1}$ that the distribution will fall to zero at a place between the two points. To satisfy this with grids generated by eq. 3.17, the first two intervals are kept equal.

For best performance the grid must be tailored to the needs of the problem, while an adaptive grid is probably the only way to cope efficiently with different cases and long-time simulations. Remeshing algorithms, however, are out of the scope of this work. Previous researchers (Gelbard and Seinfeld, 1978; Hounslow et al., 1988) have often employed scaled variables to facilitate the prediction of population density at the lower end of the size spectrum. However, as Nicmanis and Hounslow (1998) have shown, this renders the correct prediction of large particles and of the moments more difficult, as errors in the scaled domain are amplified when transforming the solution back to the original coordinates. We subscribe to this view, and therefore the method presented here is developed over an unscaled domain.

3.4 The main steps of the method

The dynamic PBE is a hyperbolic integro-differential equation with respect to time and the particulate coordinate – in this case the particle volume. The finite element method is used for the discretisation of the particulate coordinate, thus transforming the PBE into a system of ordinary differential equations with respect to discretised values of the population density. Temporal integration is subsequently carried out to compute the values in the new time point. The approach can be summarised in the following steps:

Step 1 Approximation of the solution with trial functions.

Step 2 Evaluation of integral terms. The aggregation birth, death and breakage birth terms must be evaluated via numerical integration.

Step 3 Formulation of weighted residual equations. The trial functions are substituted into the rest of the terms, and any derivatives are analytically elaborated. The resulting function, called the *residual*, is multiplied by appropriate weighting functions, depending on which version of the finite element method is desired - common choices are collocation, Galerkin and subdomain. The residual is integrated over each element, analytically or numerically.

Step 4 Integration of the resulting ODEs along time, which can be carried out with any initial value ODE integration method.

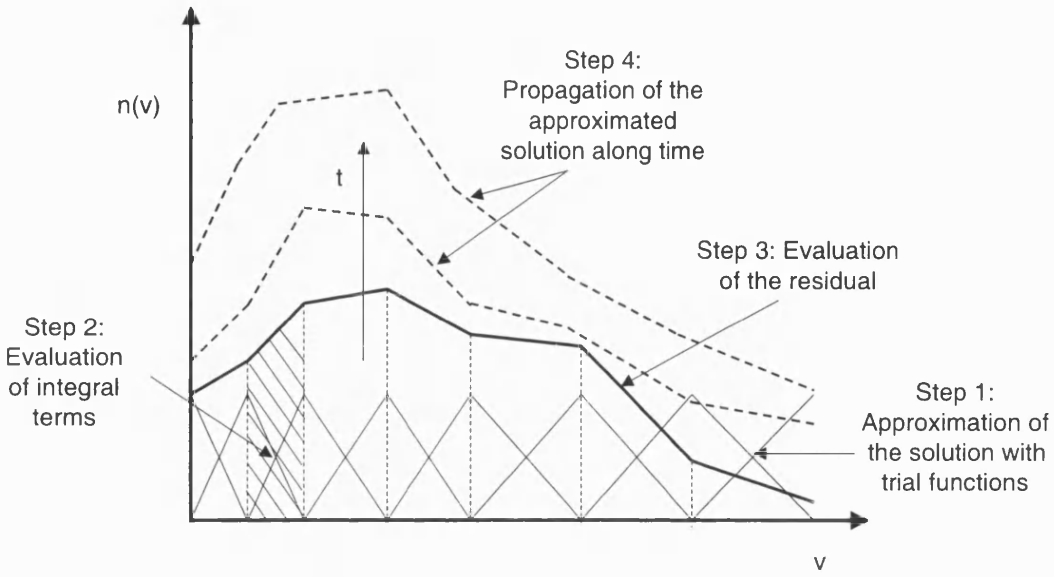

Figure 3.1: The main steps of the method.

It is useful to compare here this procedure with the implementation of the finite element method in more familiar applications, such as solid and fluid mechanics. The common practice in those cases is to derive a weak form of the original equation by using the divergence theorem, in order to lower the miminum required order of the approximating functions and to provide a means of introducing Neumann boundary conditions. This step is not required here, because the PBE is only first order with respect to both time and volume and the boundary conditions are all of Dirichlet type. Moreover, the discretised equations are often derived in terms of local variables defined within each element by a coordinate transformation. Although this practice does simplify the derivations, it cannot be applied to the PBE because of the aggregation source term which, as we shall see, involves values of the independent variable that are defined within different and often isolated elements. Finally, the application of the finite element method to an integro-differential equation requires an extra step, the evaluation of the integrals with the trial functions. It will be shown that this step is a potential source of errors in the conservation of moments if carried out in a straightforward manner.

The main steps of the method will now be elaborated.

3.4.1 Approximation of the solution

In finite element methods the solution is approximated with trial functions, which are low order Lagrangian or Hermitian polynomials that are non-zero only over a small part of the domain. These functions are written in such a way that their undetermined coefficients are the values of the solution at the nodal points. Previous work on the PBE employed mostly cubic splines (Gelbard and Seinfeld, 1978; Eyre and Wright, 1988; Steemson and White, 1988;) or high-order Lagrangian polynomials (Nicmanis and Hounslow, 1998). The scheme presented here, however, will be based on linear trial functions for two main reasons. Higher order trial functions require more evaluations and interpolations for the same number of nodal points, while first order functions can accommodate a much finer grid in the same amount of computational time - note that higher order does not necessarily imply higher accuracy, especially when the solution is irregular. Secondly, the use of linear functions enhances stability for growth problems. Furthermore, these functions simplify considerably the scheme and facilitate the analysis of conservation errors.

Two linear trial functions are required:

$$f_{1,2}(v) = \left(\frac{v - v_{i-1}}{v_i - v_{i-1}}, \frac{v_i - v}{v_i - v_{i-1}} \right) \tag{3.19}$$

The solution over an element is then approximated as:

$$n(v)_{[i-1,i]} = n_{i-1} \cdot \frac{v_i - v}{v_i - v_{i-1}} + n_i \cdot \frac{v - v_{i-1}}{v_i - v_{i-1}} \tag{3.20}$$

With linear trial functions each nodal point is also an element boundary. If the domain is divided into N elements, these correspond to $N + 1$ nodal points. In what follows, we shall use index i to denote a nodal point or an element bounded by $(i - 1, i]$ (point 0 corresponds to the lower boundary).

3.4.2 Evaluation of integral terms

The three integral terms of the PBE (aggregation birth, aggregation death, breakage birth) are the main potential sources of conervation error in the numerical solution. Note that conservation is not a synonym of accuracy, which can be improved mainly by refining the grid. The aim of a conservative numerical method is to conserve the moments of the distribution (the most important being the first, which denotes

conservation of mass) even in a coarse grid. To attain conservation, the sources generated by the discretisation must be balanced by the sinks over all elements. In the following, two potential sources of conservation error will be identified:

- Error due to grid non-uniformity.
- Error due to non-conservative quadrature.

These errors will be elaborated below, and ways of mitigating them will be proposed.

3.4.2.1 Aggregation source term

The aggregation source term is:

$$B_{agg}(v,t) = \frac{1}{2}\int_0^v \beta_a(v', v-v')n(v',t)n(v-v',t)dv' \tag{3.21}$$

The aggregation source term links each element with all the elements preceding it. To evaluate it at an element boundary v_m, the integral may be splitted into a sum of contributions from all preceding elements:

$$B_{agg}(v_m,t) = \sum_{i=1}^{m}\left[\frac{1}{2}\int_{v_{i-1}}^{v_i} \beta_a(v', v-v')n(v',t)n(v-v',t)dv'\right] \tag{3.22}$$

To calculate the contribution from each preceding element [i-1,i] we need the products of the nodal values at its boundaries, $[n_{i-1}, n_i]$ with the values at the complementary points, i.e. the points $[v_m - v_{i-1}, v_m - v_i]$. Suppose that these lie in a distant element k. Substituting the trial functions into eq. 3.22 we obtain:

$$B_{agg}(v_m,t) = \sum_{i=1}^{m}\left\{\frac{1}{2}\int_{v_{i-1}}^{v_i} \beta_a(v', v_m-v')\left[n(v_{i-1},t)\frac{v_i - v'}{v_i - v_{i-1}} + n(v_i,t)\frac{v' - v_{i-1}}{v_i - v_{i-1}}\right]\cdot \right.$$
$$\left. \cdot\left[n(v_{k-1},t)\frac{v_k - (v-v')}{v_k - v_{k-1}} + n(v_k,t)\frac{(v-v') - v_{k-1}}{v_k - v_{k-1}}\right]dv'\right\} \tag{3.23}$$

Further elaboration of eq. 3.23 is not possible, however, because it involves values of the independent variable that lie in two distinct, possibly isolated elements. This is the reason why the common practice of transformation into local element

coordinates cannot be applied to the PBE. Another important point is that in a non-uniform grid the complementary points are not, generally, element boundaries, and may not even lie within the same element. In that case, their nodal values must be interpolated from the boundaries of the elements where they belong. This is a potential source of conservation error, best understood via a geometric representation of the aggregation source term. Fig. 3.2 shows the distribution in the element i and the two neighbouring elements where the complementary points lie. By determining the nodal values by interpolation, the resulting integration corresponds to aggregation of the particles represented by the area of the following tetrahedra:

$$\Big[(v_i, 0), (v_{i-1}, 0), \big(v_{i-1}, n(v_{i-1})\big), \big(v(i), n(v_i)\big)\Big]$$
$$+$$
$$\Big[(v - v_{i-1}, 0), (v - v_i, 0), \big(v - v_i, n(v - v_i)\big), \big(v - v_{i-1}, n(v - v_{i-1})\big)\Big]$$

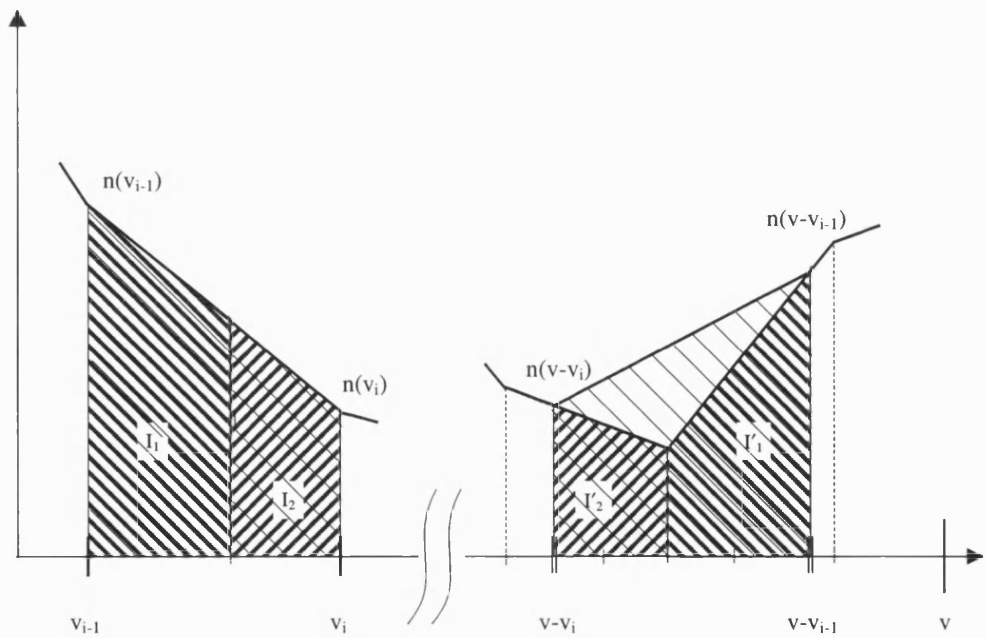

Figure 3.2: Geometrical representation of the conservation error arising in non-uniform grids.

In the case of fig. 3.2, however, the second tetrahedron includes a triangle of non-existing particles, which is the source of the conservation error. The correct aggregating particles are defined by the areas (I_1, I_1') and (I_2, I_2'). To obtain a conservative algorithm, the interval defined by the mirror points will be splitted

into a number of sub-intervals, separated by the inter-element boundaries that are located within that interval. If $K-1$ boundaries are contained, the integral over each element in eq. 3.23 is splitted into a sum of K contributions from the reseulting sub-intervals:

$$B_{agg}(v_m,t) = \frac{1}{2}\sum_{i=1}^{m}\sum_{k=1}^{K}\int_{v_{i,k-1}}^{v_{i,k}} \beta(v',v_m-v')n(v_m-v',t)n(v',t)dv' \tag{3.24}$$

The integrand of eq. 3.23 is non-linear, meaning that numerical integration must be applied. This can be another source of conservation error, since any errors in it will result in an imbalance between the aggregation birth and death terms. To eliminate this error, the integration must be exact. Since the inegrand comprises the product of the two trial functions, both first order polynomials, and the aggregation kernel, if the latter is linear exact integration can be accomplished by adopting the following rule:

$$\langle \text{order of quadrature} \rangle = \langle \text{order of kernel} \rangle + 2 \tag{3.25}$$

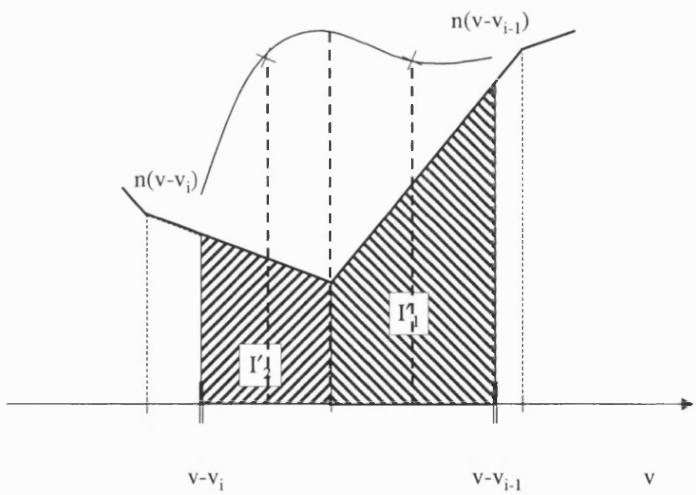

Figure 3.3: The aggregation source term is of higher order than the approximated population density (which is linear). To obtain exact quadrature, a number of points must be interpolated within each interval.

Fig. 3.3 shows the curve representing the aggregation source term, superimposed over the approximated solution at an element. Since the end-points of the elements are readily available, a Newton-Cotes closed formula is preferable. The second-order

formula, the familiar Simpson's rule, is also exact for third-order polynomials and needs only one interpolation. The use of linear trial functions proves advantageous here, as higher order functions would require more interpolations. In the case of the constant kernel, Simpson's rule yields:

$$B_{agg}(t) = \frac{1}{2}k_a \sum_{i=1}^{N}\sum_{k=1}^{K}(v_{i,k} - v_{i,k-1})\cdot$$
$$\cdot\left[\frac{1}{6}n(v_{i,k-1},t)n(v_{k-1},t) + \frac{4}{6}n(v_{i,k-1,1/2},t)n(v_{k-1,1/2},t) + \frac{1}{6}n(v_{i,k},t)n(v_k,t)\right] \tag{3.26}$$

Finally, a note will be made about non-linear kernels. In that case, exact integration can be attained via Gaussian quadrature, by using the kernel as weight function. Press et al. (1992) describe a procedure for computing the coefficients of Gaussian quadrature with arbitrary weight functions. This process is rather complex, however, and for practical purposes a Newton-Cotes formula is usually acceptable, as the test cases will demonstrate.

3.4.2.2 Aggregation sink term

The aggregation sink term is easier to treat, since its integrand involves just the product of the population density and the aggregation kernel:

$$D_{agg}(t) = n(v,t)\int_0^\infty \beta(v-v',v')n(v',t)dv' \tag{3.27}$$

As above, the evaluation should follow the rule:

$$\langle\text{order of quadrature}\rangle = \langle\text{order of kernel}\rangle + 1 \tag{3.28}$$

3.4.2.3 Breakage source term

The breakage source term can be written as:

$$\int_v^\infty \beta_b b'(v/v')n(v')dv' \tag{3.29}$$

where $b'(v/v')$ is a function that incorporates both degree of fragmentation and size distribution of fragments. To maintain conservation, integration should follow the same rules proposed above for aggregation: the quadrature should ideally be

exact for the combination of trial functions, breakage kernel and daughter particles distribution:

$$\langle \text{order of quadrature} \rangle = \langle \text{order of kernel} \rangle + \langle \text{order of distribution function} \rangle + 1 \tag{3.30}$$

3.4.3 Formulation of weighted residuals

So far, the integral terms of the PBE have been evaluated by way of numerical integration. The weighted residual expressions over each element are formulated by multiplying the PBE with an appropriate weight function and integrating:

$$\begin{aligned}
\int_{v_{i-1}}^{v_i} \left[w(v) \cdot \frac{\partial n(v,t)}{\partial t} \right] dv &= \int_{v_{i-1}}^{v_i} \left[w(v) \cdot \frac{n_{in}(v,t) - n(v,t)}{\tau} \right] dv - \\
- \int_{v_{i-1}}^{v_i} \left\{ w(v) \cdot \frac{\partial}{\partial v} \left[G(v) \cdot n(v,t) \right] \right\} dv &- \int_{v_{i-1}}^{v_i} \left\{ w(v) \cdot \left[B_0 \delta(v - v_0) \right] \right\} dv - \\
+ \int_{v_{i-1}}^{v_i} \left[w(v) \cdot B_{agg}(v,t) \right] dv &- \int_{v_{i-1}}^{v_i} \left[w(v) \cdot (D_{agg}(v,t)) \right] dv - \\
+ \int_{v_{i-1}}^{v_i} \left[w(v) \cdot B_{br}(v,t) \right] dv &- \int_{v_{i-1}}^{v_i} \left[w(v) \cdot (\beta_{br}(v) \cdot n(v,t)) \right] dv
\end{aligned} \tag{3.31}$$

Common choices of weight functions are:

$$\text{Collocation:} \quad w(v) = \delta(v - v_k) \quad \text{where k is a nodal point} \tag{3.32}$$

$$\text{Subdomain:} \quad w(v) = 1 \tag{3.33}$$

$$\text{Galerkin:} \quad w(v) c_i(v) = \frac{v - v_{i-1}}{v_i - v_{i-1}}, \frac{v_i - v}{v_i - v_{i-1}} \tag{3.34}$$

Collocation is an evaluation of the residual at the nodal points. The other schemes require integration of the residual over a whole interval, and are thus more expensive to compute. Moreover integral schemes cannot be combined with the algorithm proposed above for conservative evaluation of the aggregation source term because they require the integrand to be continuous. For these reasons, we choose to employ the collocation scheme.

3.4.3.1 Growth term

Substituting the trial functions in the growth term, we obtain:

$$\int_{v_{i-1}}^{v_i} \left\{ w(v) \cdot \frac{\partial}{\partial v} \left[G(v) \cdot n(v,t) \right] \right\} dv =$$

$$\int_{v_{i-1}}^{v_i} \left\{ \delta(v - v_k) \frac{\partial}{\partial v} \left[G(v) \cdot \left(n_{i-1}(t) \frac{v_i - v}{v_i - v_{i-1}} + n_i(t) \frac{v - v_{i-1}}{v_i - v_{i-1}} \right) \right] \right\} dv \quad (3.35)$$

In orthogonal collocation on finite elements (Finlayson, 1975), the trial functions are of higher than first order and the evaluation is performed on points interior to each element. As a result, information flows to the collocation point from both ends of the element. This strategy is not efficient for discretising the growth term, because growth is a convective mechanism. This implies that the population at a point is affected only by upstream points - particles may grow only from smaller to bigger sizes. The use of higher order schemes where information flows in both directions can result in instabilities and oscillations. Here we may draw an analogy with CFD, where upwind differencing of the convection term is employed when the flow is convection dominated. Since the PBE has to dampen the effect of growth, in order to attain an equivalent effect the following discretisation is suggested (fig. 3.4):

$$\int_{v_{i-1}}^{v_i} \left\{ \delta(v - v_k) \frac{\partial}{\partial v} \left[G(v) \cdot \left(n_{i-1}(t) \frac{v_i - v}{v_i - v_{i-1}} + n_i(t) \frac{v - v_{i-1}}{v_i - v_{i-1}} \right) \right] \right\} dv$$

$$= \frac{n_i(t)}{v_i - v_{i-1}} - \frac{n_{i-1}(t)}{v_i - v_{i-1}} + n_i(t) \cdot \frac{\partial}{\partial v} \left(G(v_i) \right) \quad (3.36)$$

An advantage of the finite element method is that it can cope with any size-dependent growth mechanism, given the volumetric growth function $G(v)$. Note that for the common case of linear size-independent growth, the volumetric growth is size-dependent:

$$G_l = \frac{dL}{dt} = \frac{\partial \left({}^{v}/_{k_v} \right)^{1/3}}{\partial v} \cdot \frac{dv}{dt} = \frac{1}{3k_v^{1/3}} v^{-2/3} G(v) \quad (3.37)$$

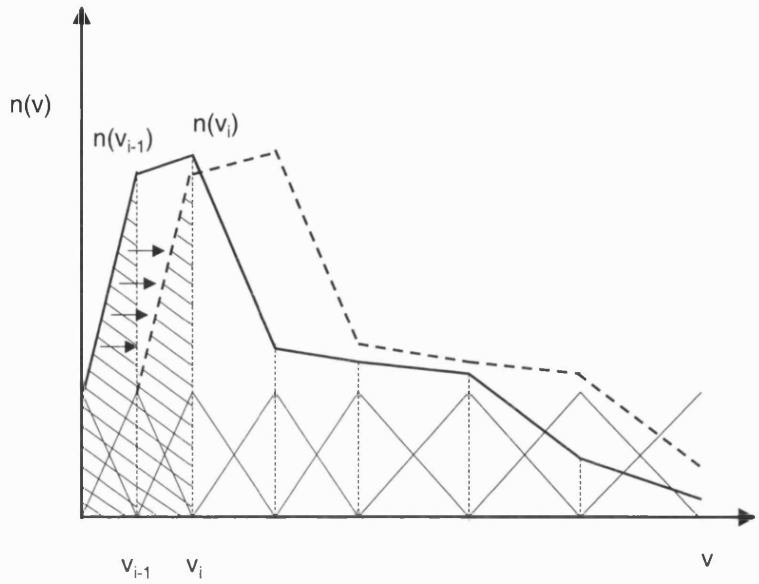

Figure 3.4: Growth makes the population density move towards larger sizes only - hence the need for upwind discretisation.

3.4.3.2 Nucleation term

Nucleation is implemented as a point source at the first nodal point. With finite elements, however, it is impossible to have true point sources, as trial functions are valid over a whole interval. Care must be taken, therefore, in order to make sure that the nucleation rate $B_0(t)$ reflects the number of particles produced in the first two intervals that are affected and has units of (time^{-1}volume^{-1}):

$$\int_{v_{i-1}}^{v_i} [\delta(v - v_0) \cdot B_0(t)]\, dv = \frac{B_0}{\frac{1}{2}(v_1 - v_0)} + \frac{B_0}{\frac{1}{2}(v_2 - v_1)} \tag{3.38}$$

3.4.4 Temporal Integration

Adopting the collocation scheme, and in the absence of diffusion, the steps described so far result in the transformation of the original integro-differential dynamic PBE into a set of ordinary differential equations of the form:

$$\begin{aligned} \frac{\partial n_i(t)}{\partial t} &= \frac{n_{in,i}(t) - n_i(t)}{\tau} - \frac{n_i(t)}{v_i - v_{i-1}} + \frac{n_{i-1}(t)}{v_i - v_{i-1}} - \\ -n_i(t) \cdot \frac{\partial}{\partial v}(G(v_i)) &+ \delta(v - v0) \cdot \left[\frac{B_0}{\frac{1}{2}(v_1 - v_0)} + \frac{B_0}{\frac{1}{2}(v_2 - v_1)} \right] + \\ &+ B_{agg}(v_i, t) - D_{agg}(v_i, t) + B_{br}(v_i, t) - \beta_{br}(v_i) \cdot n_i(t) \end{aligned} \tag{3.39}$$

Having performed the pointwise evaluations of the PBE terms at the nodal points, we have succeeded in transforming the original partial integro-differential equation into a set of ordinary differential equations, where the dependent variables are the nodal values. The integration of these ODEs along time is bound to introduce further errors in terms of both accuracy and conservation, that can only be mitigated by employing an accurate integration algorithm. The results presented here were obtained using the LSODE integration package (Hindmarsh, 1983), using the Adams class of methods with adaptive time-stepping. The source code of LSODE is available in public domain (e.g. via netlib.org). In most cases, the error in the first moment was maintained below 1%. The FORTRAN 90 programming language has been used for the computational implementation.

3.5 Results and discussion

The method will now be tested against analytical solutions of the PBE for simplified problems. The CPU times reported correspond to an average desktop PC (Pentium III 933 MHz).

3.5.1 Aggregation

The continuous time-dependent PBE for aggregation only is:

$$\begin{aligned}\frac{\partial n(v,t)}{\partial t} = \frac{1}{2}\int_0^v b_a(v-v',v')n(v-v',t)n(v',t)dv' - \\ -n(v,t)\int_0^\infty b_a(v,v')n(v',t)dv'\end{aligned} \tag{3.40}$$

Most analytical solutions have been derived for the exponential initial distribution:

$$n(v,0) = \frac{N_0}{v_0}\cdot e^{\left(-\frac{v}{v_0}\right)} \tag{3.41}$$

According to eq. 3.41, at $v = 0$ the distribution assumes a finite value ($n(0,0) = \frac{N_0}{v_0}$). This is incompatible with the initial condition used in our formulation ($n(0,0) = 0$), and is further unrealistic, as there are no particles of zero size. In the test cases, the initial distribution in the simulation starts from zero and assumes the values of the exponential distribution at the size of the nuclei. This results in a slight

disagreement with the analytical solution at sizes close to zero, more evident in the aggregation-growth case (IV-2).

The grid to be employed in the pure aggregation cases is non-uniform, following the exponential series of eq. 3.17. The parameters, summarised in table 3.1, were chosen so that aggregation may proceed to a large extent. This can be quantified by the index of aggregation, defined by Hounslow (1990) on the basis of the zeroth moment:

$$I_{agg}(t) = 1 - \frac{m_0(t)}{m_{0,i}} \tag{3.42}$$

3.5.1.1 Case I-1: Size-independent kernel

This kernel can be used as an approximation of Smoluchowski's equations for Brownian coagulation (perikinetic aggregation - see chapter 1). It is defined as:

$$\beta_a(v - v', v') = k_a \tag{3.43}$$

For the exponential initial distribution, Scott (1968) derived the following solution:

$$n(v) = k_a \frac{4N_0}{(T+2)^2} \cdot \exp\left(-\frac{v}{v_0} \cdot \frac{2}{T+2}\right) \tag{3.44}$$

where N_0 and v_0 are constants that denote the initial number of particles and average volume, respectively, and T is a dimensionless time defined as:

$$T = k_a N_0 t \tag{3.45}$$

A comparison of the numerical and analytical solutions for the population density, plotted on logarithmic axes in fig. 3.5, shows excellent agreement. The conservation of the first moment is also maintained at a good level, as shown in table 3.1, despite the use of a fairly coarse grid.

3.5.1.2 Case I-2: Sum (Golovin) kernel

This kernel is defined by:

$$\beta_a(v - v', v') = k_a(v + v') \tag{3.46}$$

A solution for this kernel, again with the exponential initial distribution (eq. 3.41), has also been provided by Scott:

$$n(v) = \frac{N_0(1-T)}{vT^{1/2}} \cdot \exp(-(1+T)v') \cdot I_1(2v'T^{1/2}) \tag{3.47}$$

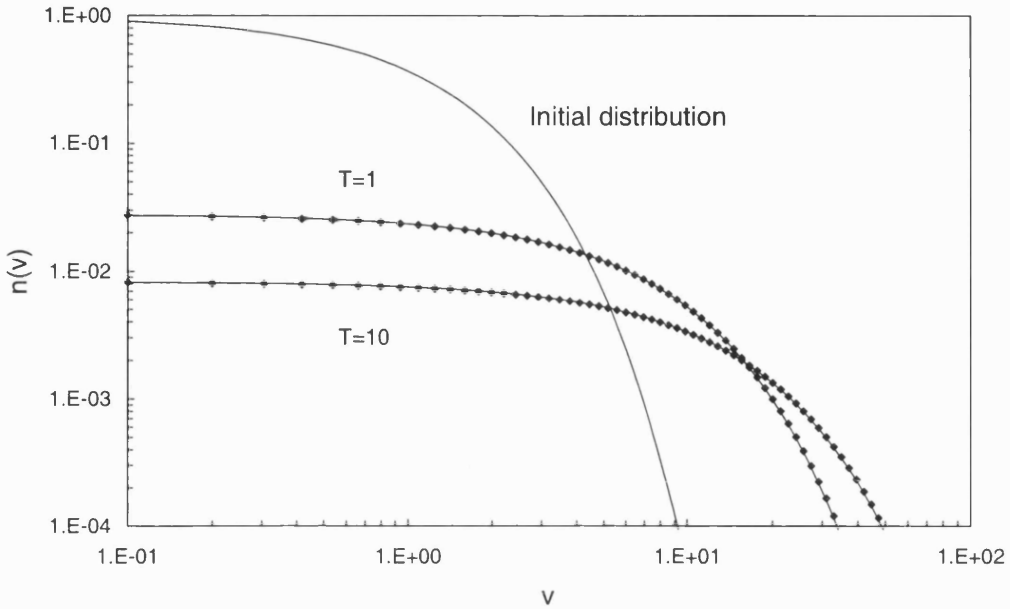

Figure 3.5: Aggregation, constant kernel, comparison of simulation with the analytical solution of Scott (1968)

where $I_1$is the modified Bessel function of the first kind of order one. The dimensionless time here is defined as:

$$T = b_a N_0 v_0 t \tag{3.48}$$

Results for the population density are plotted in fig. 3.6, while the moments and aggregation index are shown in table 3.1. As with the case of constant kernel, aggreement is very good.

3.5.1.3 Case I-3: Fluid shear and gravitational kernels

These kernels, describing orthokinetic aggregation and gravitational settling respectively, were discussed in chapter 1.

The shear kernel is defined as:

$$\beta_a(v - v', v') = k_a(v^{1/3} + v'^{1/3})^3 \tag{3.49}$$

As we saw in chapter 1, this kernel corresponds to Smoluchowski's theory of orthokinetic aggregation. Another kernel derived with physical arguments is the

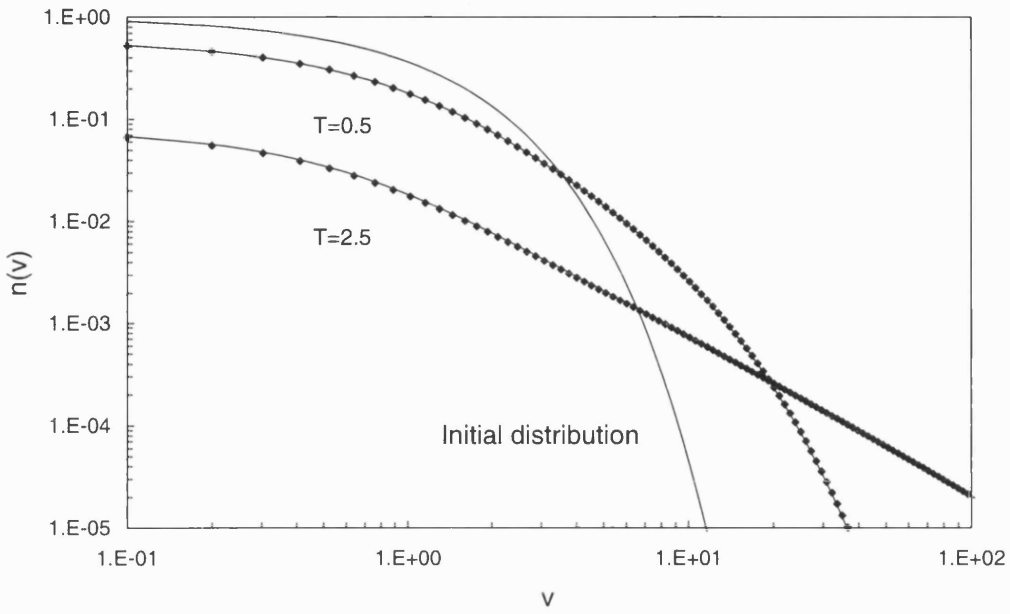

Figure 3.6: Aggregation, sum kernel, comparison of simulation with the analytical solution of Scott (1968)

gravitational coalescence (Drake, 1972):

$$\beta_a(v - v', v') = k_a(v^{1/3} + v'^{1/3})^2 \cdot \left| v^{2/3} + v'^{2/3} \right| \tag{3.50}$$

Although there are no analytical solutions for these kernels that we are aware of, they will be considered here due to their physical significance. Furthermore, it is interesting to see how the method behaves in the presence of a non-linear kernel. The solution may still be partly evaluated on the basis of the conservation of the first moment, which must remain constant in any aggregation process.

As discussed earlier, the aggregation source term with a non-linear kernel requires a custom-derived Gaussian quadrature to be integrated exactly. Before taking the effort to derive such a rule, however, it is worth to try first the Newton-Cotes quadrature used in the linear kernels. Table 3.1 shows the conservation of the first moment for both of these kernels, after employing Simpson's rule for evaluating the aggregation source term. Evidently the error is negligible, and this implementation should be sufficient for most cases. The steep increase in CPU time is due to the need for computing fractional exponents. In fig. 3.7, the distributions resulting from these kernels are compared with those of the sum and constant kernels. Aggregation

seems to be more intense with the Smoluchowski and gravitational kernels, a fact that can be deduced by the form of these kernels.

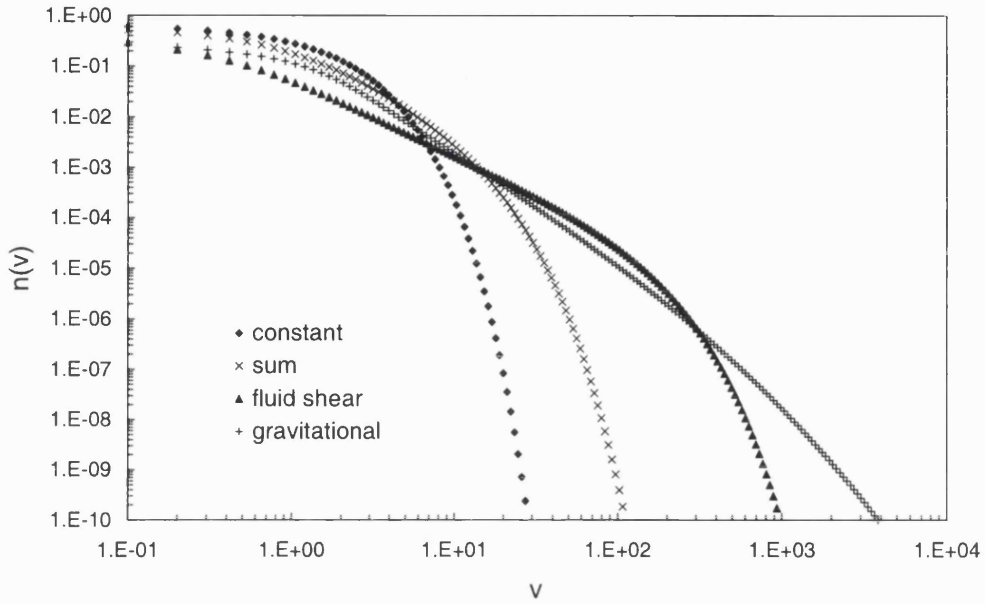

Figure 3.7: Aggregation, constant, sum, fluid shear and gravitational kernels, simulation results.

3.5.2 Nucleation-growth

Nucleation-growth problems have been studied extensively by, among others, Randolph and Larson (1988). The dynamic PBE for nucleation-growth with flow is:

$$\frac{\partial n(v,t)}{\partial t} = \frac{n_{in}(v,t) - n(v,t)}{\tau} - \frac{\partial}{\partial v}\left(G(v) \cdot n(v,t)\right) + B\delta(v - v_0) \tag{3.51}$$

As this is a hyperbolic equation, the main challenge for a numerical method is to capture the steep moving fronts and shocks that may emerge. Note that the analytical solutions for nucleation-growth have been derived by incorporating nucleation as a boundary condition ($n(v_0, t) = B_0/G$), while in this method nucleation enters as a point source. Parameters of the simulations are shown in table 3.2.

Case	I-1		I-2		I-3	
Kernel	k_a	k_a	k_a(v+v')	k_a(v+v')	shear (eq.23)	gravity
k_a	1	1	1	1	1	1
N_0	1	1	1	1	1	1
v_0	1	1	1	1	1	1
t (sec)	10	20	0.5	2	0.5	0.5
T	10	20	0.5	2	-	-
Grid type	exp.	exp.	exp.	exp.	exp.	exp.
α (grid exponent)	1.05	1.05	1.04	1.04	1.04	1.04
No. of elements	100	100	150	150	200	200
Size range	0.1-100	0.1-100	0.1-100	0.1-100	0.1-100	0.1-100
I_{agg}, %	82	90	39	86	77	63
Error in M_1, %	0.12	0.51	0.39	1.021	0.87	0.91
CPU (sec)	0.5	1	0.8	1.5	2	20

Table 3.1: Results for the aggregation cases.

3.5.2.1 Case II-1: Size-independent linear growth and nucleation with flow

Most analytical solutions refer to size-independent growth, for which case eq. 3.51 is simplified to:

$$\frac{\partial n(v,t)}{\partial t} = \frac{n_{in}(v,t) - n(v,t)}{\tau} - G\frac{\partial n(v,t)}{\partial v} + B\delta(v - v_0) \tag{3.52}$$

Note that in reality, growth can be independent only with respect to a linear size variable, and the case of volume-independent growth is considered here only for validation purposes. At steady-state, the solution to eq. 3.52 is the well-known MSMPR equation (eq. 1.48), written here in terms of volume:

$$n(v,\infty) = \frac{B}{G} \cdot e^{-v/G\tau} \tag{3.53}$$

Fig. 3.8 shows the time-dependent numerical solution and the steady-state analytical solution. After several residence times, the two curves merge. Comparison is very good, apart from the first two points, whose deviation has been explained above. Table 3.2 shows the numerical parameters employed in this case.

3.5.2.2 Case II-2: Size-dependent growth and nucleation with flow

A commonly used size-dependent growth function is (Randolph and Larson, 1988):

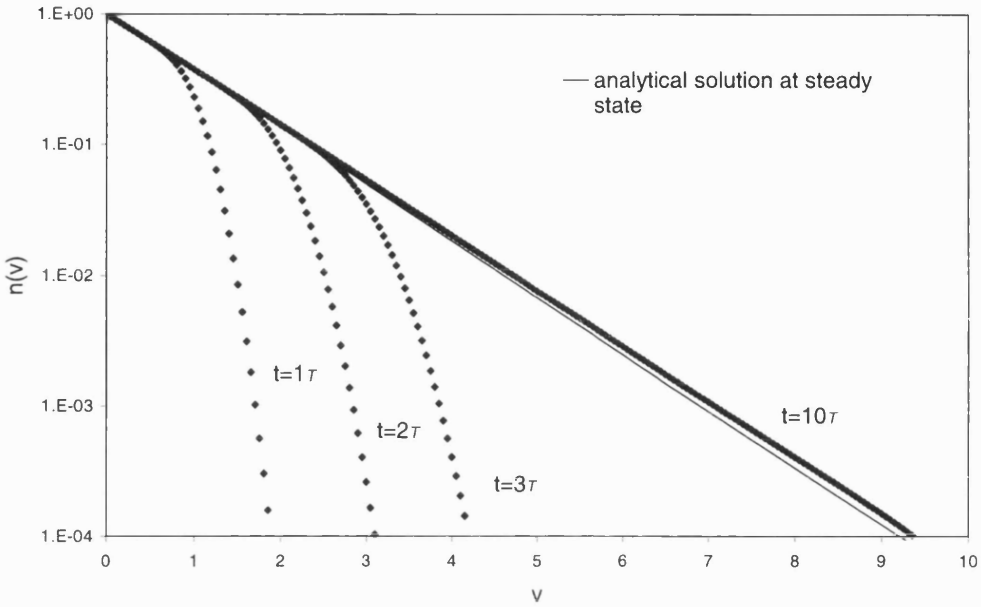

Figure 3.8: Steady-state size-independent nucleation-growth, comparison with the analytical solution of Randolph and Larson (1988).

$$G = G_0(1 + \gamma v)^{\beta} \tag{3.54}$$

This function involves the growth rate of nuclei, G_0, and two empirical parameters. The above authors also provide a steady-state solution for this class of functions:

$$n(v, \infty) = K \cdot \frac{B}{G} \cdot (1 + \gamma v)^{-\beta} \exp\left[\frac{-(1 + \gamma v)^{-\beta}}{G_0 \tau \gamma (1 - b)}\right] \tag{3.55}$$

where

$$K = \exp\left[\frac{1}{G_0 \tau \gamma (1 - b)}\right] \tag{3.56}$$

Fig. 3.9 shows the numerical solution at increasing values of the residence time. After 10 residence times, the simulation approximates the analytical solution at steady state.

3.5.2.3 Case II-3: Batch Size-independent nucleation-growth

The analytical time-dependent solution to eq. 3.51 for size-independent growth, no flow and no particles present at t=0, has been derived by Hounslow et al. (1988):

$$n(v, t) = \frac{B}{G} \cdot H(t - \frac{v}{G}) \tag{3.57}$$

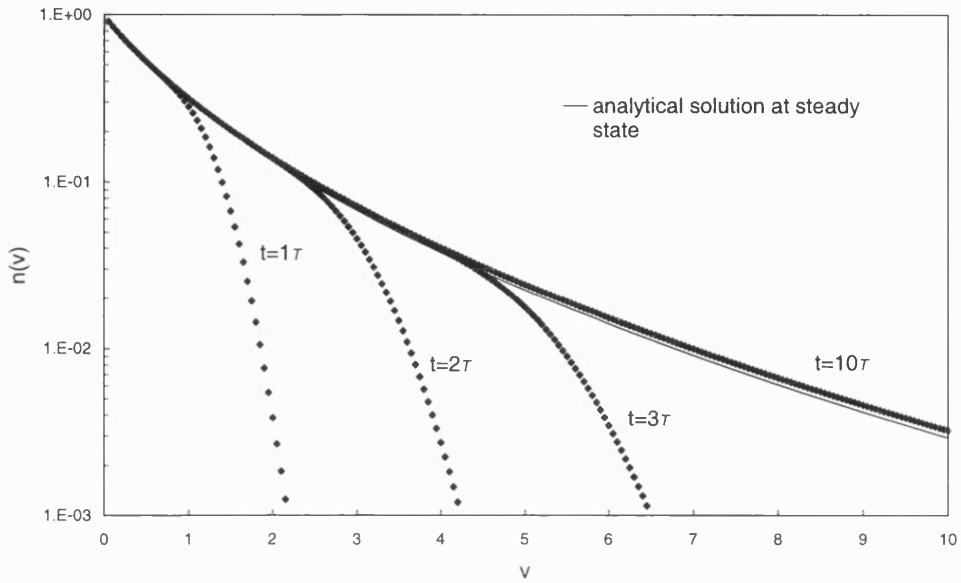

Figure 3.9: Steady-state size-dependent nucleation-growth, comparison with the analytical solution of Randolph and Larson (1988).

where $H()$ is the step function. This is a very ill-conditioned problem, as the solution is simply a discontinuity that moves with time. Such a discontinuity is very difficult to capture, and may become a source of numerical instability. The method of Hounslow et al. (1988), being of zero order, proved stable enough, but its prediction of the front was smeared by numerical dispersion. This effect can only be mitigated by the use of a finer grid, but in that method the grid is constrained to follow a geometrical progression. The finite element method, on the other hand, may employ a uniform grid, which is the most suitable for this case.

Fig. 3.10 compares numerical and analytical results. The stability of the method is owning to the first-order trial functions and the upwind discretisation of growth. Dispersion around the discontinuity is inevitable, since this is a first order method, but it can be controlled by refining the grid. A smaller time step was also chosen to mitigate the dispersion, but CPU time is still in the order of seconds.

3.5.3 Breakage

The equation for breakage without flow is:

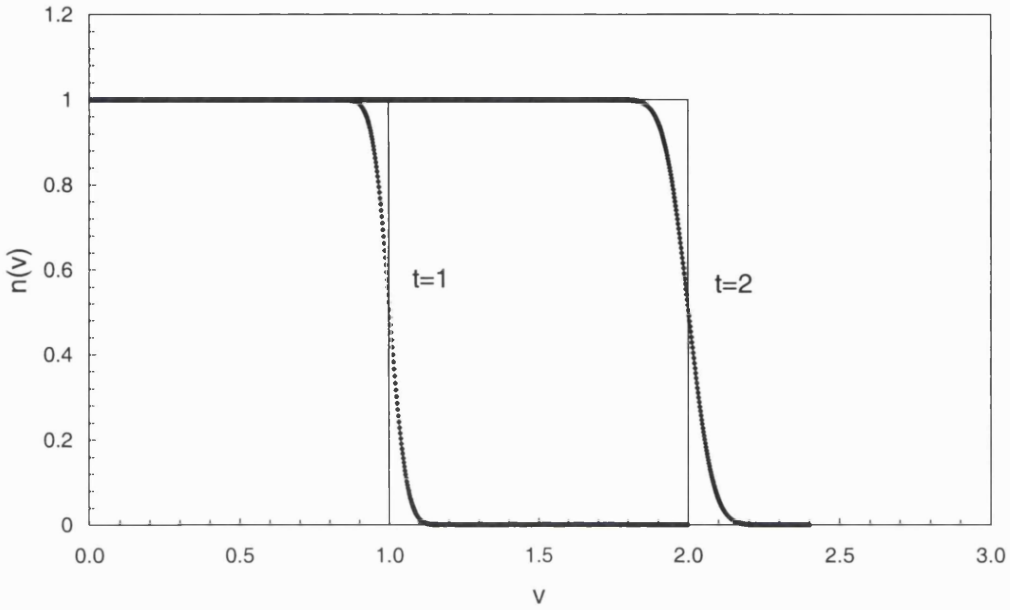

Figure 3.10: Batch size-independent nucleation-growth, comparison with the analytical solution of Hounslow et al. (1988)

$$\frac{\partial n(v,t)}{\partial t} = \int_{v}^{\infty} \beta_b(v')b(v/v)n(v',t)dv' - \beta_b(v)n(v,t) \tag{3.58}$$

The functions $N_0(v)$ and $b(v/v')$ denoting the degree of fragmentation and size distribution of daughter particles are often merged in to a single function, $b'(v/v')$. The possibilities for this function are numerous, but in this work we shall consider three representative cases: uniform binary breakage, parabolic binary breakage and multiple fragmentation. Parameters are shown in table 3.3.

Case	I-1		I-2	
Growth kernel	G_0	$G_0(1+\gamma v)^{\beta}$	G_0	G_0
B_0	1	1	1	1
G_0	1	1	1	1
γ	-	1	-	-
β	-	1	-	-
t (τ)	10	10	1	2
Grid type	uniform	uniform	uniform	uniform
Grid increment	0.5	0.5	0.002	0.002
No. of elements	200	200	1200	1200
Size range	0.1-10	0.1-10	0-002-2.4	0.002-2.4
CPU (sec)	1	1.5	5	8

Table 3.2: Nucleation-growth cases, parameters and results.

3.5.3.1 Case III-1: Uniform binary breakage

Uniform breakage implies that fragments of every size have the same probability to form. This situation may arise e.g. in polymer chains which, being one-dimensional, are as likely to break at any point. If the breakage is also binary, it can be represented by the following overall function:

$$b'(v/v') = \frac{2}{v'} \tag{3.59}$$

It can be easily confirmed that eq. 3.59 satisfies the constraints given by eqs. 3.9 - 3.11. Ziff and McGrady (1985) derived an analytical time-dependent solution for this case, assuming a first-order size-dependent kernel ($\beta(v) = k_b \cdot v$:

$$n(v,t) = \exp(-tv^2 - v) \cdot [1 + 2t(1+v)] \tag{3.60}$$

Fig. 3.5.3.1 shows the initial distribution and the distributions resulting from breakage at different times. A comparison between the simulation and the analytical solution shows that the approximation is very good.

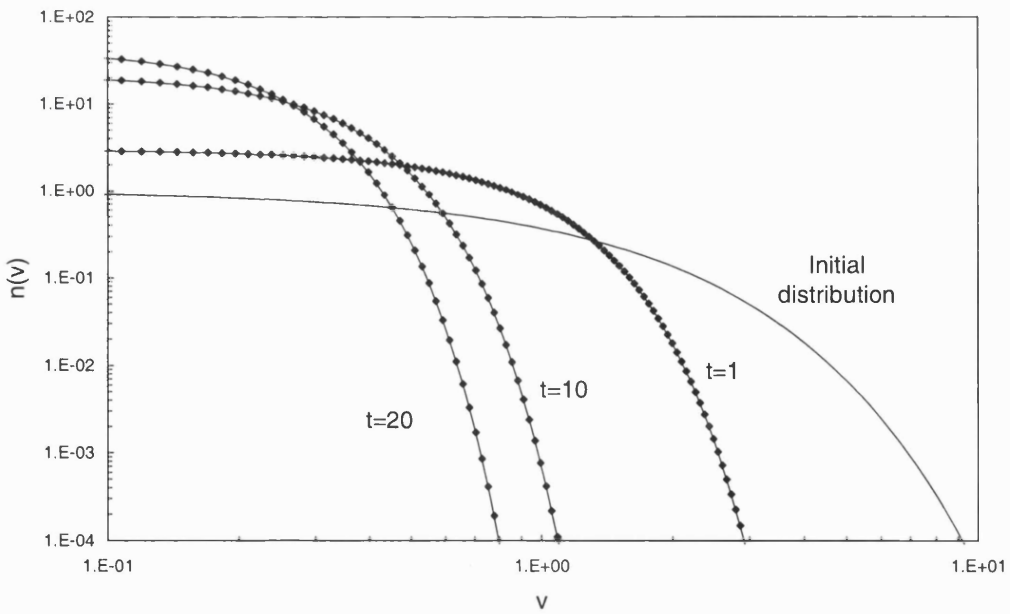

Figure 3.11: Batch uniform binary breakage, comparison of simulation and analytical solution of Ziff and McGrady (1985).

3.5.3.2 Case III-2: Parabolic binary breakage

Rigid particles are more likely to break into unequal pieces. This can be accounted for via the use of a parabolic "U-shape" function to describe the daughter paricle size distribution. An overall function describing parabolic binary breakage has been suggested by Hill and Ng (1995):

$$b(v,v') = \frac{24\left(1-\frac{hv'}{2}\right)\left(v^2 - vv' + \frac{v'^2}{4}\right)}{v'^3} + h \tag{3.61}$$

For $0 < h < 2/v$, this function represents a concave parabolic profile with decreasing steepness. At $h = 2/v$ the function is equal to the uniform binary breakage, while for $2/v < h < 3/v$ the function represents a convex parabola (an unlikely case). Outside this bounds the function assumes unrealistic negative values.

To the authors' knowledge there is no general analytical solution except for h=2 in which case the simulation approximates very well the solution for uniform binary breakage (eq. 3.59) ; moreover, the first moment is properly conserved in all three cases. As expected, parabolic breakage favours smaller and bigger particle sizes in the expense of the intermediate ones, resulting in a slightly bimodal distribution (fig. 3.5.3.2).

3.5.3.3 Case III-3: Multiple breakage (fragmentation)

Ziff (1991) derived analytical solutions for a class of fragmentation problems where the number of fragments and their size distribution are described by:

$$b'(v/v') = \frac{a\gamma}{v'}\left(\frac{v}{v'}\right)^{\gamma-2} + \frac{(1-a)\delta}{v'}\left(\frac{v}{v'}\right)^{\delta-2} \tag{3.62}$$

where α,γ,δ are parameters. The average number of fragments produced can be determined by the following:

$$\overline{N}(v') = \int_0^{v'} b(v/v')dv \tag{3.63}$$

Ziff derived complete time-dependent solutions for a subset of this class, defined by:

$$b'(v/v') = \frac{\delta\lambda}{\delta-\lambda}\left(\frac{v}{v'}^{\lambda-2} - \frac{v}{v'}^{\delta-2}\right) \tag{3.64}$$

$$\beta_b(v) = k_b \cdot v^{\lambda} \tag{3.65}$$

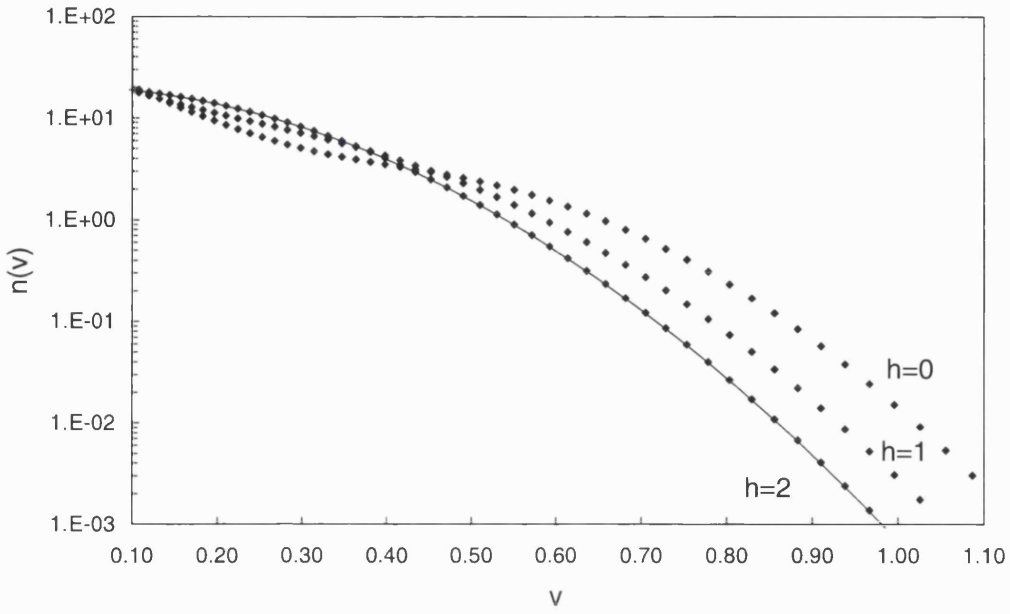

Figure 3.12: Batch parabolic breakage for h=0,1,2 and comparison with analytical solution (Ziff and McGrady, 1985) for h=2.

For a monodisperse initial condition, his solution is:

$$n(v,t) = I^{-1}\exp(-tI^{\lambda})\delta(v-I) + \lambda\delta t v^{\delta-2}\int_{v}^{I} v'^{\lambda-\delta-1}\exp(-tv'^{\lambda})dv' \tag{3.66}$$

$$n(v,0) = I^{-1}\delta(v-I) \tag{3.67}$$

Fig. 3.5.3.3 shows the numerical and analytical solution the case of ternary fragmentation ($\lambda = 2, \delta = 3$). A uniform grid was employed for this simulation. To implement the monodisperse initial condition, an initial distribution resembling a spike had to be employed, resulting in a rather ill-conditioned problem; nevertheless solution was easily obtained.

3.5.4 Combined problems

Combined problems (involving combinations of the mechanisms) are the most difficult to tackle, and very few analytical solutions have been found to compare with. At least two such cases are known to the authors.

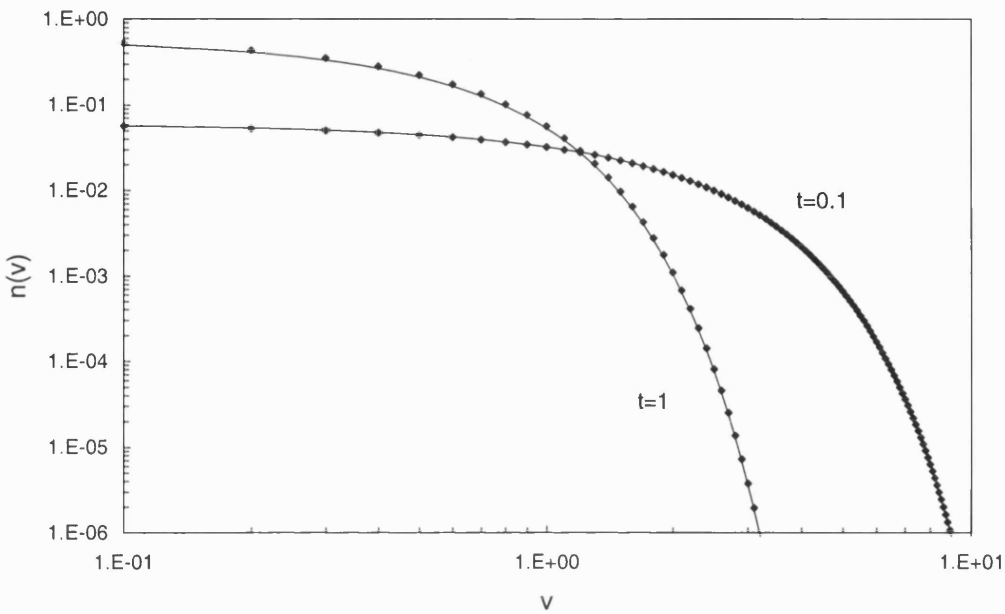

Figure 3.13: Batch ternary fragmentation, comparison of simulation and analytical solution of Ziff (1991).

3.5.4.1 Case IV-1: Steady-state nucleation - aggregation

Hostomsky (1987) derived a semi-analytical solution for steady-state nucleation and size-independent aggregation with flow and no particles in the feed. Under these conditions, the PBE may be written as:

$$\frac{1}{2}b_a \int_0^v n(v-v')n(v')dv' - n(v)b_a \int_0^\infty n(v')dv' + B_0\delta(v-v_0) - \frac{n(v)}{\tau} = 0 \qquad (3.68)$$

The solution is:

$$n(v) = \left[\frac{(1+2\alpha)v_0}{4\pi}\right]^{1/2} \cdot \frac{[\alpha/(\alpha+1/2)]^{v/v_0}}{b_a\tau v^{3/2}} \qquad (3.69)$$

where

$$\alpha = b_a B_0 \tau$$

Hostomsky's solution is semi-analytical because its derivation involves a Taylor series expansion, and therefore subject to error. It holds for $v > 10v_0$ and is more accurate for higher volumes. Hostomsky provided also an equation for the zeroth moment:

$$M_0 = \frac{(2b_a B_0 \tau^2 + 1)^{1/2} - 1}{b_a \tau} \qquad (3.70)$$

Case	III-1		
Breakage Kernel	$k_b v$	$k_b v$	$k_b v$
Daughter particle distribution	2/v	2/v	2/v
k_b	1	1	1
N_0	1	1	1
v_0	1	1	1
t	1	10	20
Grid type	exp.	exp.	exp.
Grid exponent	1.02	1.02	1.02
No. of elements	200	200	200
Size range	0.01-25	0.01-25	0.01-25
Error in M_1, %	0.41	0.53	0.64
CPU (sec)	14	20	23

Case	III-2			III-3	
Breakage Kernel	k_b	$k_b v$	$k_b\ \nu^\lambda$	$k_b\ \nu^\lambda$	$k_b\ \nu^\lambda$
Daughter particle distribution	parabolic	parabolic	parabolic	multiple	multiple
k_b	1	1	1	1	1
N_0	1	1	1	1	1
v_0	1	1	1	1	1
t	10	10	10	0.1	1
Grid type	exp.	exp.	exp.	uniform	uniform
Grid exponent	1.02	1.02	1.02	0.1	0.1
No. of elements	200	200	200	200	200
Size range	0.01-10	0.01-10	0.01-10	0.01-10	0.01-10
Error in M_1, %	0.54	0.53	0.48	0.11	0.68
CPU (sec)	20	20	20	10	14

Table 3.3: Breakage cases, parameters and results.

Fig. 3.14 compares this solution with the numerical simulation. A discrepancy in the first points is due to the semi-analytical solution, which does is inaccurate at that range. At $v > 10v_0$ agreement is quite good, and the zeroth moment is also well predicted (table 3.4).

3.5.4.2 Case IV-II: Batch aggregation-growth

This case was studied by Ramabhadran et al. (1976). For a size-independent aggregation kernel, a first-order linear size-dependent growth term ($G = k_g v$) and the exponential initial distribution they derived the following analytical solution:

$$n(v,t) = \frac{N_0}{v_0}(1-T^2)\exp\left(-\frac{2\Lambda T}{1-T}\right)\exp\left[-\frac{v}{v_0}(1-T)\exp\left(-\frac{2\Lambda T}{1-T}\right)\right] \quad (3.71)$$

where:

$$\Lambda = \frac{\sigma_1}{\beta_0 N_0}$$

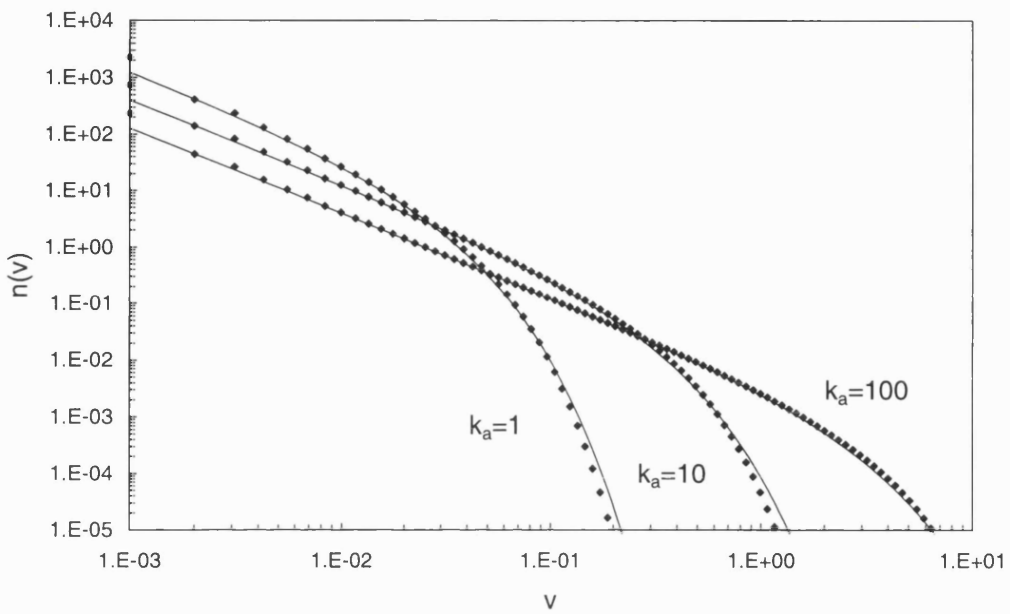

Figure 3.14: Steady-state aggregation (constant kernel) and nucleation, comparison with semi-analytical solution of Hostomsky (1987).

$$T = 1 - \frac{2}{2 + \tau/\Lambda}$$

$$\tau = k_g t$$

Fig. 3.15 shows the analytical and numerical results. A discrepancy in the beginning is, as explained above, due to the deviation of the initial condition used in the simulation from eq. 3.17.

An interesting feature in this case is the speed with which the distribution moves towards the larger sizes, due to a combination of very rapid growth and aggregation. Fig. 3.16 shows the mass distribution moving rapidly towards bigger sizes. A larger grid is required in this case, in order to cover sufficiently the whole range over which the distribution is moving.

3.5.5 Discussion

Our test cases indicated that the CPU time demanded from an average desktop PC for nucleation-growth problems is at about the same order of magnitude as the real time being simulated, while for problems involving aggregation and/or breakage it increases roughly by an order of magnitude due to the integral terms (e.g. a few

Case	IV-1			IV-2	
Agg. kernel	k_a	k_a	k_a	k_a	k_a
Growth kernel	G_0v	G_0v	1	1	100
B_0	-	-	10	10	10
k_a	1	1	1	1	1
N_0	1	1	-	-	-
v_0	1	1	-	-	-
G_0	0.1	0.1	-	-	-
β	1	1	-	-	-
t (sec)	3	10	-	-	-
τ	-	-	1	1	1
Grid type	exp.	exp.	exp.	exp.	exp.
Grid exponent	1.05	1.05	1.08	1.08	1.08
No. of elements	120	120	100	100	100
Size range	0.005-300	0.005-300	0.001-25	0.001-25	0.001-25
Error in M_0, %	-	-	0.021	0.023	0.029
CPU (sec)	5	7	2	2	2

Table 3.4: Combined cases parameters and results.

minutes of CPU are required for a few seconds of real time). Direct comparison with CPU times reported by other authors is not straightforward because they refer to different computers, but an evaluation can be attempted based on the mechanics of each method. The use of linear elements, in conjunction with our strategy for evaluating the integral terms, makes the number of computations and interpolations substantially fewer than those required for higher-order elements (as shown, for most aggregation kernels just one interpolation is enough). Thus the method is expected to be much faster than higher-order finite element collocation methods, such as cubic spline or orthogonal collocation, while integral formulations such as Galerkin are even more involved. Higher order methods may be more accurate, but the CPU time saved due to the first-order elements can be spent on refining the grid, a practice that is guaranteed to improve accuracy. DPBs, especially those based on a geometric series (such as the Hounslow method) should be even faster, because their way of evaluating the integral terms requires fewer computations. However, the important conclusion is that the CPU demands raised by this new method are affordable even with desktop PCs, while a workstation would be fast enough for any real case.

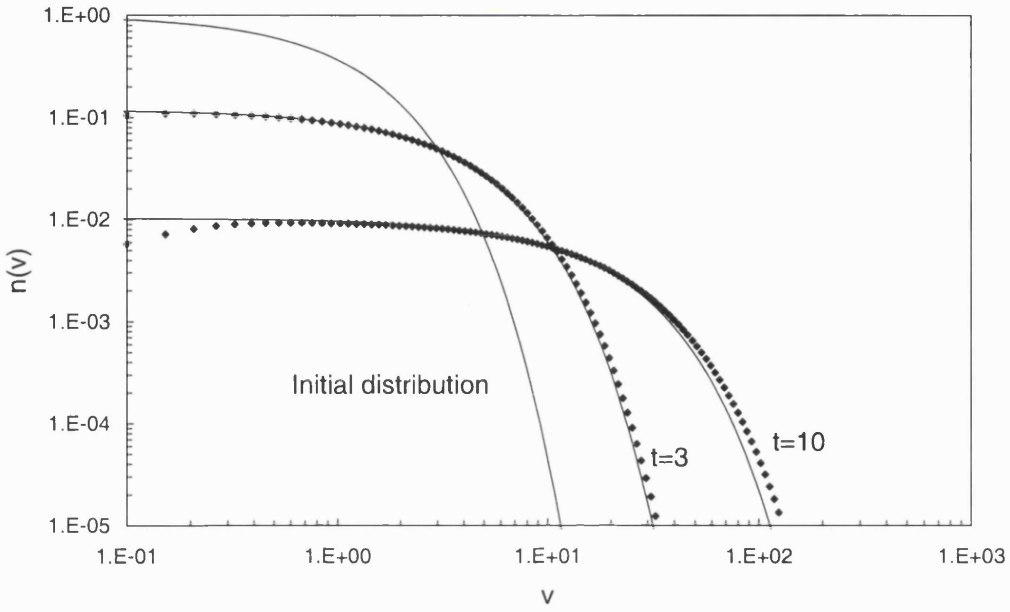

Figure 3.15: Batch aggregation (constant kernel) and growth (size-dependent), comparison of simulation and analytical solution of Ramabhadran et al., (1976)

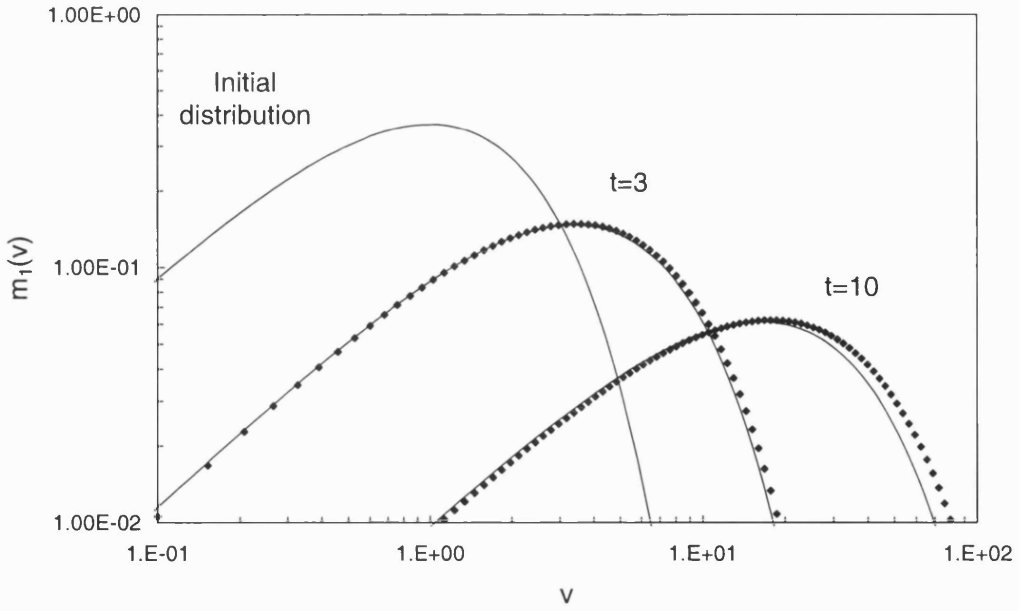

Figure 3.16: Evolution of first moment (mass) in the aggregation-growth problem - comparison of simulation and analytical solution of Ramabhadran et al. (1976).

Chapter 4

Gas-liquid precipitation of $CaCO_3$ in a bubble column reactor

4.1 Introduction

In this chapter, the theoretical and computational methods that were developed so far will be applied to the analysis of gas-liquid precipitation. Being a complex process whose outcome is determined by the interplay of fluid dynamics, chemical reactions and particulate processes, gas-liquid precipitation is a case study that fully demonstrates the need for a hybrid modelling approach. Moreover, accurate population balance modelling is essential in order to predict the crystal size distribution (CSD) of the precipitate, thus granting potential for product engineering.

4.1.1 Industrial importance

Gas-liquid precipitation is the formation of a solid crystalline product via chemical reactions involving a gas and a liquid phase. A variety of products, mostly inorganic crystals and speciality chemicals, are produced in this way. Wachi and Jones (1992) listed several such processes:

- Ammonium phosphate ($(NH_4)_2HPO_4$) , from gaseous NH_3 and phosphoric acid (H_3PO_4)

- Ammonium sulphate ($(NH_4)_2SO_4$), from $NH_{3(g)}$ and sulphuric acid (H_2SO_4)
- Barium carbonate ($BaCO_3$), by carbonation of either barium hydroxide ($Ba(OH)_2$ or barium sulphate ($BaSO_4$)
- Calcium carbonate, by carbonation of lime ($Ca(OH)_2$) solution
- Calcium fluoride (CaF_2) by calcium nitrate ($CaNO_3$) and hydrogen fluoride (HF)
- Gypsum ($CaSO_4$), from sulphur oxide (SO_2) and lime ($Ca(OH)_2$)
- Geothite (FeOOH) by air oxidation of alkaline suspension of ferrous hydroxide ($Fe(OH)_2$
- Potassium chloride (KCl), by ammoniation of sea bitterns containing chlorides of potassium, sodium and calcium
- Sodium bicarbonate ($NaHCO_3$) by carbonation of salt and ammonia
- Strontium carbonate ($SrCO_3$) by carbonation of strontium hydroxyde ($Sr(OH)_2$)
- Terepthalic acid ($COOH(C_6H_4)COOH$) by air oxidation of p-xylene in acetic acid with cobalt and manganese salts of metal bromide at 200 °C and 400 psi. The precipitated crystals are used for the manufacture of Dacron$^T M$ and Terylene$^T M$.

Despite its industrial importance (Jones, 2002), gas-liquid precipitation has only recently become the objective of rigorous study. A comprehensive review of the state of the art has been provided by Wachi and Jones (1995). The objective of the analysis and mathematical modelling is to provide means of controlling the precipitate properties, such as the crystal size distribution (CSD), by adjusting the process design and operation parameters.

4.1.2 Present challenges

Gas-liquid reactive precipitation was first studied in laboratory-scale reactors (fig. 4.1, (b) with a flat gas-liquid interface. Experimental and modelling studies on such equipment have been conducted by Yagi and co-workers (1986), Wachi and Jones

(1991) and Hostomsky and Jones (1995). Other laboratory-scale devices that have been employed, mainly for investigating the kinetics of gas-liquid precipitation, are an agitated tank with the gas being dispersed in the form of bubbles (Yagi, Iwazawa, Sonobe, Matsubara and Hikita, 1984; Yagi, 1990; Kotaki and Tsuge, 1990) and the Couette-Taylor reactor (Jung, Kang, Kim and Choi, 2000). The latter was employed for studying of morphology of precipitated $CaCO_3$ by careful control of solution composition and shear.

Industrial reactors for gas-liquid precipitation are shown in fig. 4.1 (a,c,d). A proper analysis of such equipment requires consideration of hydrodynamics, which determine several key reaction engineering parameters such as the mass transfer rate. Furthermore, prediction of the CSD can be accomplished by solution of the population balance equation. The coupling of the above requires excessive computational resources, and it has been realised (Wachi and Jones, 1995) that a stagewise model should be employed rather than a differential one.

The complexity is further increased by the fact that such processes are usually operated in batch or semi-batch mode, which implies that dynamic modelling is required. As the concentrations of the reacting species vary with time, supersaturation levels and crystallisation rates also vary. Consequently the population balance is strongly coupled with the reacting species' mass balances, thus ruling out analytical solutions even in the simplest cases (e.g. nucleation-growth, no agglomeration).

4.1.3 Modelling of gas-liquid precipitation with the method of moments

Earlier work (Rigopoulos and Jones, 2001) employed a stage-wise dynamic model for the simulation of a tall bubble column carrying out $CaCO_3$. Gas-liquid mass transfer and chemical reaction was accounted for via an implementation of the penetration theory similar to the one in this work. In fact, that model can be viewed as a specific case of the generalised compartmental model presented in chapter 2. Fig. 4.2 shows the representation of the bubble column in that work.

The population balance was formulated with crystal diameter as the independent coordinate (eq. 1.49), and coupled with the mass balances of the chemical species within each fully-mixed compartment. Mean crystal size was predicted with the

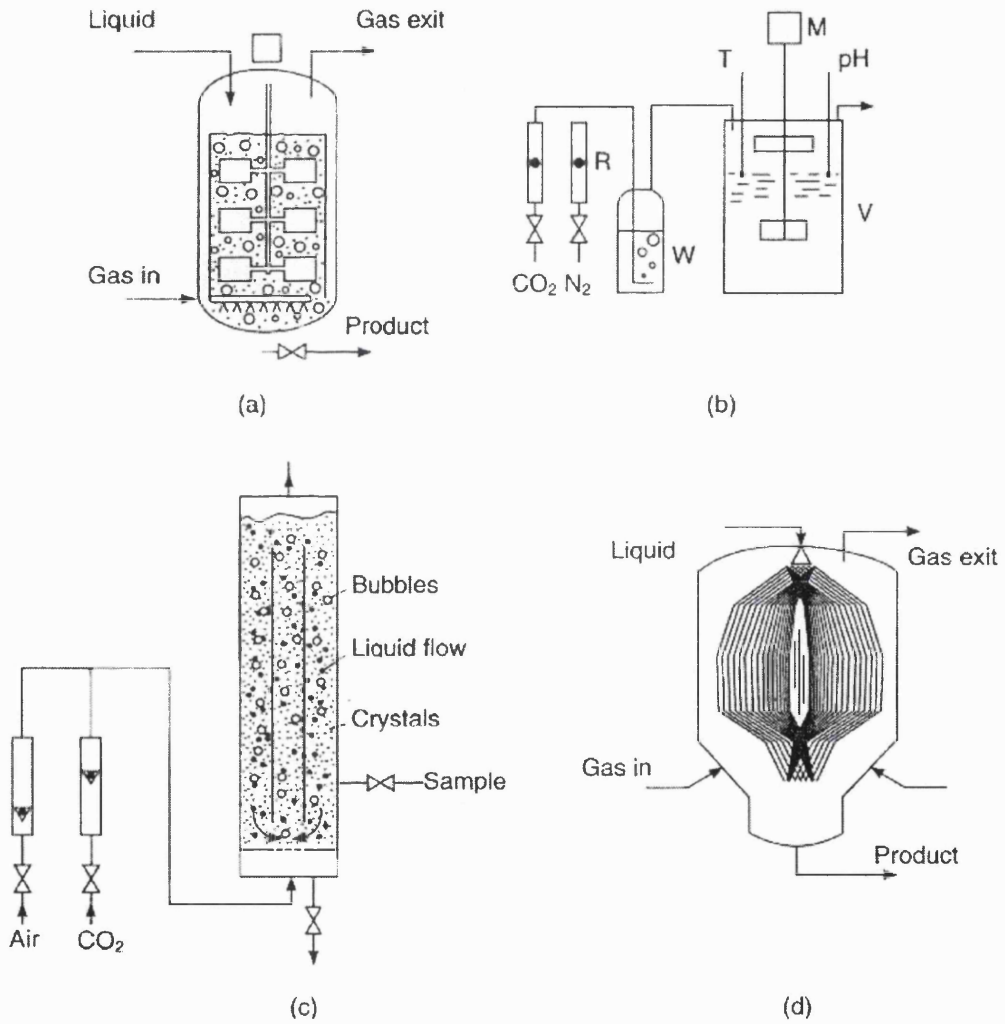

Figure 4.1: Reactors for gas-liquid precipitation - (a) bubbling stirred tank, (b) flat interface stirred tank, (c) draft tube bubble column, (d) spray column (reprinted from Jones, 2002)

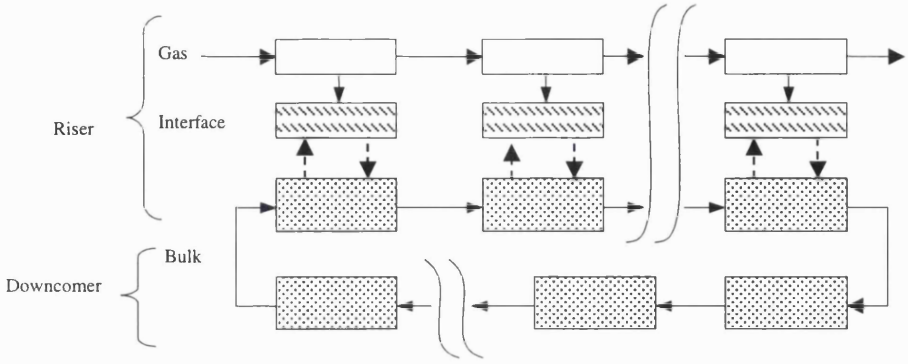

Figure 4.2: Stage-wise model for a draft-tube bubble-column reactor (Rigopoulos and Jones, 2001)

method of moments. This method, discussed in sec. 1.4.5.2, leads to a closed set of equations when nucleation and growth only are considered; it is not suitable for incorporating inter-particle events such as agglomeration and breakage.

The final model consisted of the following set of ODEs, describing the evolution of the moments and chemical species' concentrations within each compartment:

$$\frac{\partial m_j(t)}{\partial t} = \frac{m_{j,l-1} - m_{j,l}(t)}{\tau} + B(C_i(t))L_0^j + G(C_i(t)) \cdot j m_{j-1}(t) \tag{4.1}$$

$$\frac{dC_{i,l}(t)}{dt} = \frac{C_{i,l-1} - C_{i,l}(t)}{\tau} + \sum_{k=1}^{K} r_k(C_{1,l}(t), \ldots C_{n,l}(t)) \tag{4.2}$$

The zeroth, first, second and third moments were resolved. These quantities, as explained in sec. 1.4.5.2, denote the total number, length, surface area and volume of the crystals and comprise the minimum set of moments that leads to a closed system.

4.1.4 Deficiencies of the moments solution

Comparison with experimental results from an earlier work by Jones, Wachi and Delannoy (1992) is shown in fig. 4.3. The main trends were adequately predicted, but several questions such as the rapid increase in mean size after the pH drop, possibly attributed to agglomeration, were left unanswered. This was due to the fact that both the experimental results and model predictions reported only mean values of the CSD.

Agglomeration seems to be pronounced during gas-liquid precipitation, as the

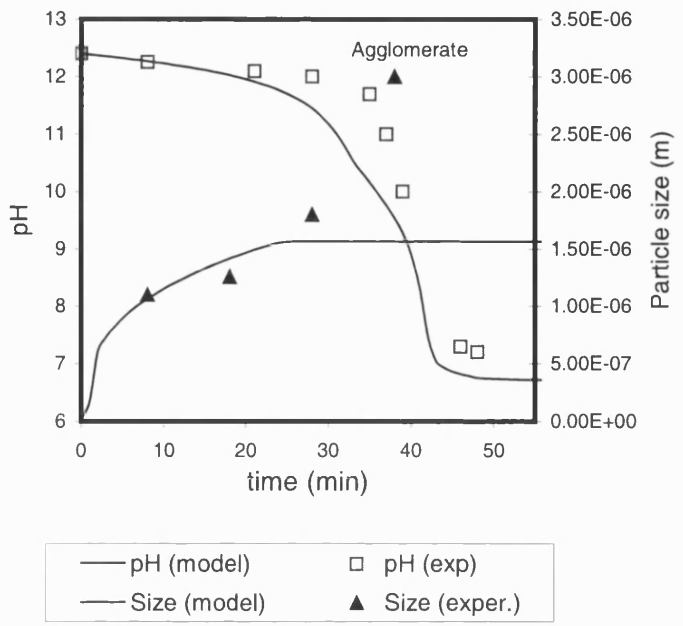

Figure 4.3: Mean size and pH predicted with the method of moments (Rigopoulos and Jones, 2001)

experiments of Kotaki et al. (1990) have shown. It has a profound effect in the CSD, and may determine the physical properties of the end product as well as its potential for separation; depending on the application, either its promotion or prevention may be desirable. Its mathematical analysis is complex, however: it requires detailed solution of the population balance, and moreover no definitive kinetic model is available.

In order to answer these questions new experiments of $CaCO_3$ precipitation are reported here, during which the entire CSD is retrieved via laser light scattering. In addition, the conservative finite element discretisation method of the dynamic volume-based population balance that was developed in chapter 3 will be employed to simulate the evolution of the CSD. Thus deeper insight can be drawn into the gas-liquid precipitation process and the underlying mechanisms; furthermore this process, encompassing a variety of phenomena and characteristic scales, provides an ideal vehicle for testing the theoretical methods developed in the preceding chapters.

4.2 Experimental part

The objective of this section is to describe the equipment and analytical methods employed in the experimental work, and to draw several clues regarding the precipitation mechanism from a preliminary inspection of the experimental results. These clues will prove to be of vital importance for configuring the kinetic model of $CaCO_3$ precipitation. It must be stressed that the present work does not constitute an exhaustive study of $CaCO_3$ precipitation kinetics. Such a study would require isolation of individual mechanisms, such as growth or agglomeration, which is not possible in a bubble column, an equipment whose simplicity of design gives rise to extraordinary complexity in the physical and chemical phenomena. A large body of literature has been devoted to that problem, employing special equipment such as the Couette-Taylor (Jung, Kang, Kim and Choi, 2000) and the Poiseuille flow crystalliser (Mumtaz and Hounslow, 2000). Rather than this, our purpose is to explore whether the complex hydrodynamic, reaction engineering and crystallisation phenomena taking place in an industrial type of reactor, such as the bubble column, can be investigated with the aid of a mathematical model. Precipitation kinetics tend to be system-dependent and unpredictable, and virtually no systematic study has been done on bubble columns; literature values will be used wherever possible, and a semi-empirical procedure for adjusting the remaining parameters will be outlined.

4.2.1 Experimental setup

4.2.1.1 Gas-liquid precipitation rig

The experimental work was conducted using a bubble column reactor, equipped with an internal concentric draft tube to enhance circulation and mixing. The outer cylinder was 0.77 m high, with internal diameter 0.215 m; the corresponding dimensions of the draft tube were 0.45 m and 0.135 m. Gas (a mixture of CO_2 and N_2) was injected through a sparger ring situated at the bottom of the inner annulus. In each experiment, the column was filled with approximately 21 lt of $Ca(OH)_2$ solution (Analar grade $Ca(OH)_2$ - deionised water). Before commencing the experiment, pure N_2 gas was being injected for some time to enhance the dissolution of the $Ca(OH)_2$ by intense circulation and mixing.

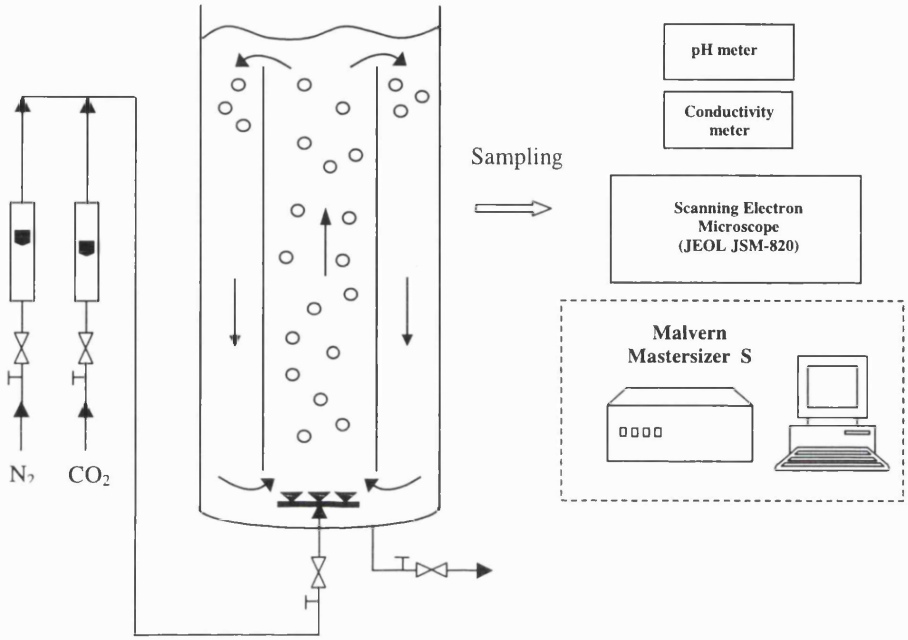

Figure 4.4: Experimental bubble column rig (21lt)

Since the aim was to observe the dynamic behaviour of the process, during the course of each experiment regular samples were taken and analysed immediately for pH, concentration of Ca^{++} and crystal size distribution. The CSD analysis was carried out with a laser scattering instrument (Malvern Mastersizer S), equipped with lenses valid for measuring particles in the range 0.05 - 900 m. The measurements were essentially off-line, but since the $CaCO_3$ suspension could be examined in the Mastersizer without any need for further processing and each measurement took only a few seconds to complete, no significant changes in the CSD could occur during the sampling procedure. To gain more insight into the crystal structure and morphology, selected samples were dried and further analysed with Scanning Electron Microscopy (JEOL JSM-820). These analytical methods will be further discussed below.

4.2.1.2 Measurement of crystal size via laser light scattering

The determination of particle size by laser scattering is based on the interaction between an incident light source and the solid particles. The amounts of reflection, refraction, and absorption of the incident wave depend on the refractive index of the particle, its geometry, and the wavelength of the incident light (Rawlings et al., 1993). By the use of Fourier and Reverse Fourier optics, this scattering is

imaged to an array of detectors at the focal plane of the optics. There is a direct relationship between the distribution of the scattered light energy on these detectors and the particle size distribution which gives rise to it. This method has become the preferred standard in many industries for characterization and quality control. The applicable range according to ISO13320 is $0.13000\mu m$ (Malvern Instruments, 2001). Instruments consist of:

- A laser as a source of coherent intense light of fixed wavelength. He-Ne gas lasers (=0.63m) are the most common as they offer the best stability (especially with respect to temperature) and better signal to noise than the higher wavelength laser diodes. It is to be expected when smaller laser diodes can reach 600nm and below and become more reliable that these will begin to replace the bulkier gas lasers.

- A suitable detector. Usually this is a slice of photosensitive silicon with a number of discrete detectors. It can be shown that there is an optimum number of detectors (16-32) increased numbers do not mean increased resolution. For the photon correlation spectroscopy technique (PCS) used in the range 1nm 1m approximately, the intensity of light scattered is so low that a photomultiplier tube, together with a signal correlator is needed to make sense of the information.

- Some means of passing the sample through the laser beam. In practice it is possible to measure aerosol sprays directly by spraying them through the beam. This makes a traditionally difficult measurement extremely simple. A dry powder can be blown sprayed into the environment. Particles in suspension can be measured by recirculating the sample in front of the laser beam.

Older instruments and some existing instruments rely only on the Fraunhofer approximation which assumes that the particle is much larger than the wavelength of light employed (ISO13320 defines this as being greater than 40λ i.e. 25m when a 625nm He-Ne laser is used), that all sizes of particle scatter with equal efficiencies and that the particle is opaque and transmits no light. These assumptions are never correct for many materials and for small material they can give rise to errors

approaching 30% especially when the relative refractive index of the material and medium is close to unity. When the particle size approaches the wavelength of light the scattering becomes a complex function with maxima and minima present. The latest instruments, such as the Mastersizer S used in this work, use the full Mie theory which completely solves the equations for interaction of light with matter. This allows completely accurate results over a large size range ($0.02 - 2000\mu m$ typically - $0.05 - 900\mu m$ in the case of the Mastersizer S). The Mie theory assumes the volume of the particle as opposed to Fraunhofer which is a projected area prediction. It is valid for all particulate materials in all transparent media liquids, gases or even transparent solids (Malvern Instruments, 2001).

4.2.1.3 Examination of crystal morphology via Scanning Electron Microscopy

Electron microscopy is a rather straightforward technique to determine the size and shape of microscopic particles. It can also reveal information on their composition and internal structure, e.g. by detecting the characteristic x-rays that are produced by the interaction of the electrons with matter, or by analysing how the electrons are diffracted. Three types of electron microscope are commonly encountered: transmission electron microscopy (TEM), scanning electron microscopy (SEM) and scanning transmission electron microscopy (STEM) which combines the two previous modes of operation.

In this work, SEM technology was employed for exploring crystal morphology. SEM is carried out by scanning the surface via a narrow electron beam and detecting the yield of either secondary or backscattered electrons as a function of the position of the primary beam, Contrast is caused by the orientation: parts of the surface facing the detector appear brighter than parts of the surface with their surface normal point away from the detector. The secondary electrons have mostly low energies ($5-50eV$) and originate from the surface region of the sample. Backscattered electrons come from deeper and carry information on the composition of the sample, because heavy elements are more efficient scatterers and appear brighter in the image. Dedicated SEM instruments have resolutions of about 5 nm. The main difference between SEM and TEM is that SEM sees contrast due to the topology and composition

of a surface, whereas the electron beam in TEM projects all information on the mass it encounters in a two-dimensional image, which, however, is of sub-nanometer resolution (Niemantsverdriet, 1993).

4.2.2 Results and discussion

4.2.2.1 Time course of precipitation

Typical plots of the temporal evolution of pH, [Ca^{++}], mean crystal size and CSD are shown in Figs. 4.5, 4.7, 4.6. The CSD is plotted as:

$$\begin{array}{lcl} \text{abscissa} & : & \dfrac{\text{volume \% in interval}}{\text{interval length (diameter unit)}} \quad \left[\dfrac{1}{\mu m}\right] \\ \text{ordinate} & : & \text{particle diameter} \quad [\mu m] \end{array}$$

The main features may be summarised as:

- Most of the precipitation occurs during a period of high pH (11-12), with [Ca^{++}] being consumed at a slightly decaying rate. This period is followed by a rapid fall in the pH, while [Ca^{++}] has reached a miminum. Subsequently, [Ca^{++}] increases again, indicating re-dissolution of some of the crystal mass.
- After an induction period, the mean crystal size increases at a slow rate. As soon as the pH drops a rapid increase occurs, eventually stabilising around $5 - 6\mu m$.
- The first CSD measurement (immediately after the induction period) is dominated by a single bell-shaped peak around $1\mu m$. The peak becomes gradually sharper, with a tail on its right. At t=240 sec a new peak appears at the size range $0.1\mu m - 1\mu m$. At the end of the precipitation, signified by the pH drop, the bimodal distribution remains.

4.2.2.2 Product morphology

Fig. 4.8 shows a large-scale view of the precipitate at an early stage of the process (soon after the end of the induction time), evidently comprising mostly single crystals. A closer view (figs. 4.9, 4.10) reveals that rombohedric calcites form the majority of the product crystals. Occasional agglomerates are also present, and in

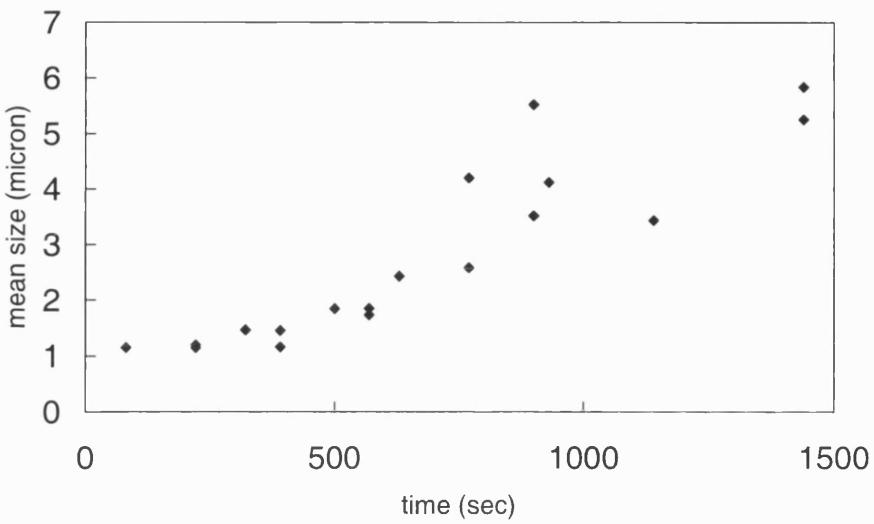

Figure 4.5: Evolution of mean size during a typical experiment

fig. 4.11 we get a close view at one of them. It is interesting that agglomerates appear before the pH drop, indicating that high pH does not completely prevent agglomeration, even if it has an inhibitory effect. The initial agglomerates have sharp, clearly defined edges, and the single crystals that comprise them can be clearly identified.

At a later stage, after the pH drop, the product is dominated by agglomerates (figs. 4.12, 4.13), confirming the increase in mean size. Meanwhile the edges of the agglomerates become smoother and smoother (fig. 4.14), until we reach a stage where only agglomerates exist, exhibiting an almost amorphous shape (fig. 4.15). At the same time the concentration of the precipitate is evidently lower.

4.2.3 Interpretation of results

The equilibrium of the CO_2 - OH^- system must be born in mind when attempting to interpret these results. In chapter 2, we plotted (fig. 2.15) the equilibrium concentrations of $[CO_3^=]$, $[HCO_3^-]$ and $[CO_{2,aq}]$, the prevailing species at the alkaline pH region, calculated with the data of Astarita (1967). $CO_3^=$, which contributes to the precipitation process, is present at a significant proportion only at pH > 10. At intermediate pH, HCO_3^- prevails; later, only $[CO_{2,aq}]$ is present.

The increase in mean size after the pH drop is in accordance with previous

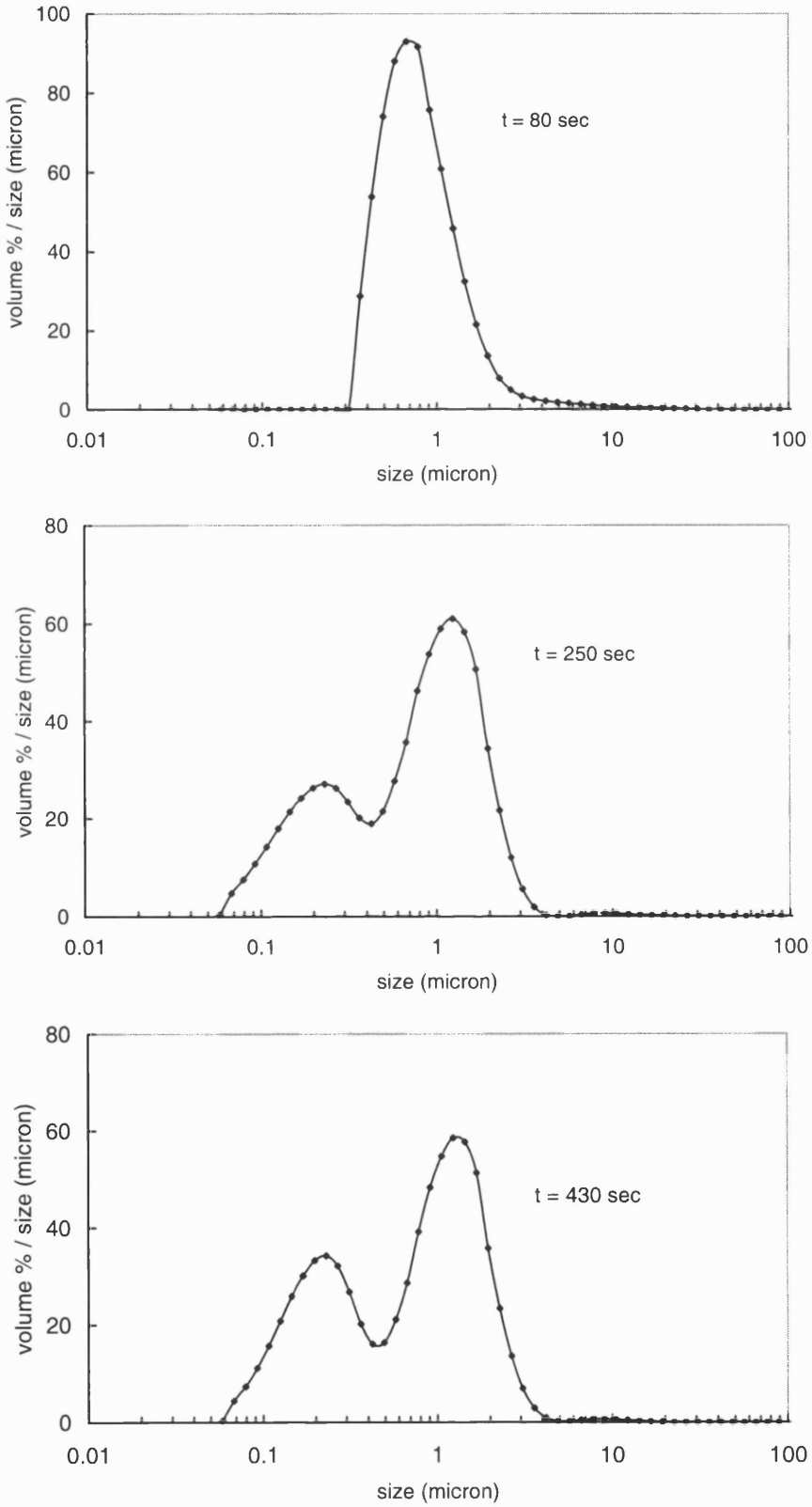

Figure 4.6: Evolution of CSD during a typical experiment

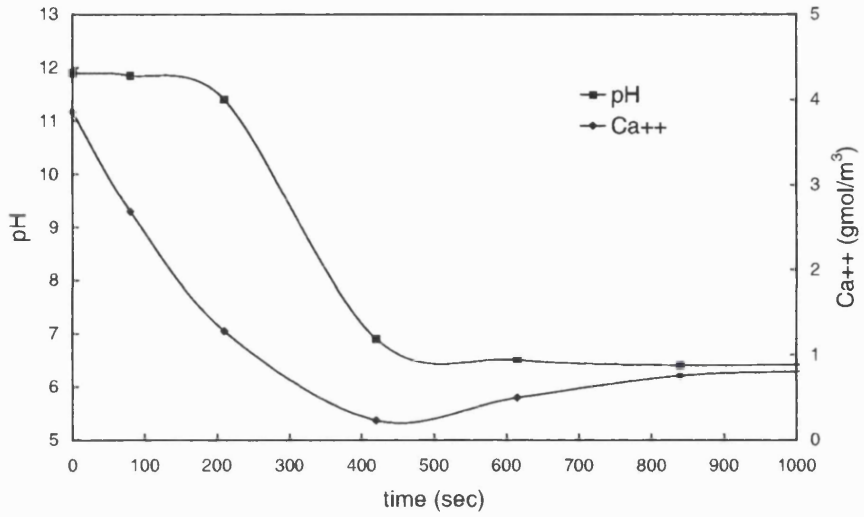

Figure 4.7: Evolution of pH and Ca^{++} during a typical experiment

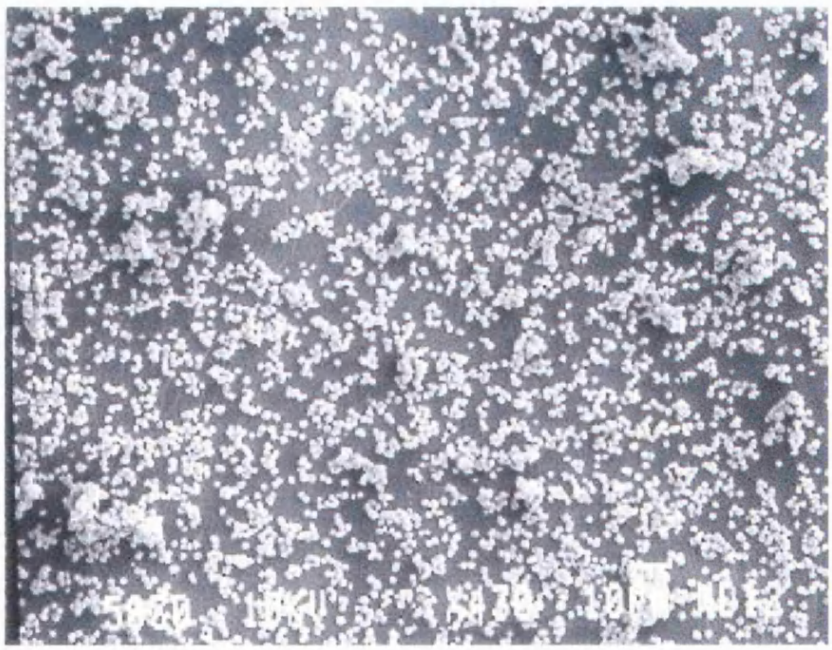

Figure 4.8: Large-scale view of the precipitate, shortly after the induction time

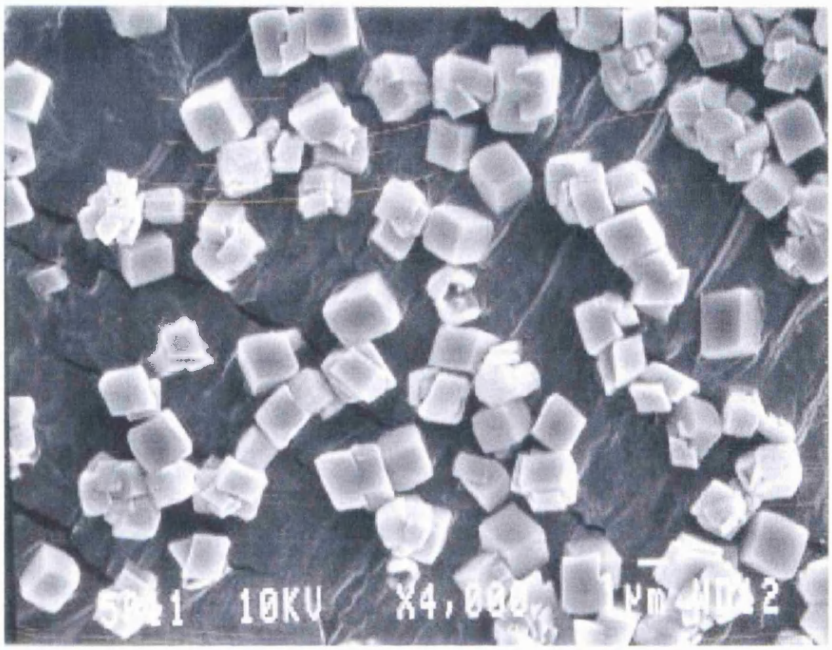

Figure 4.9: A closer view of the single product crystals (a)

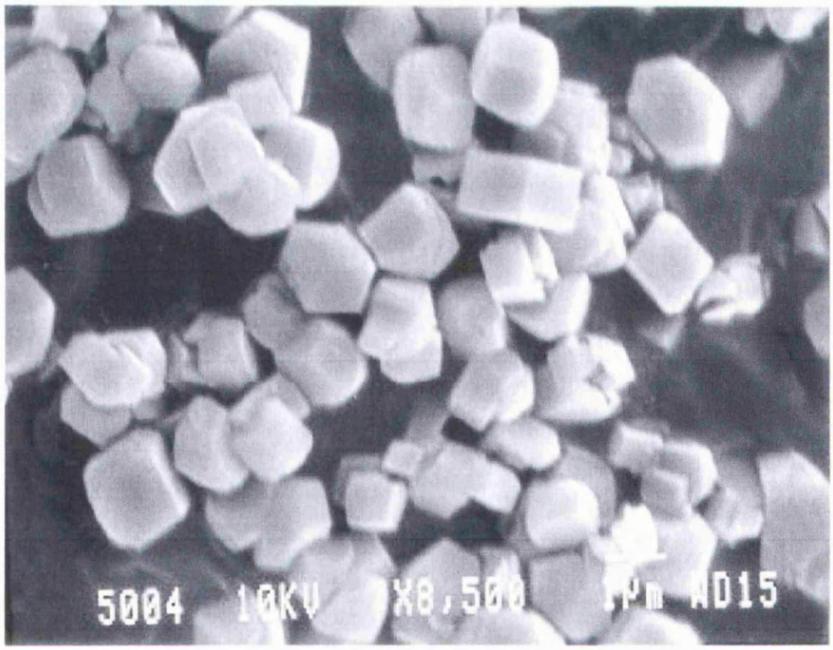

Figure 4.10: A closer view of the single product crystals (b)

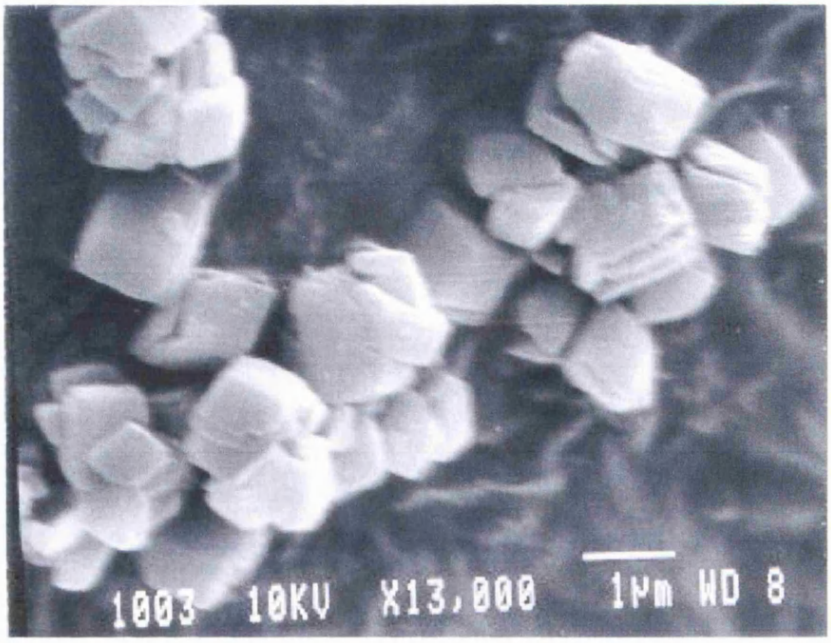

Figure 4.11: Close view of an agglomerate formed at an early stage of the process)

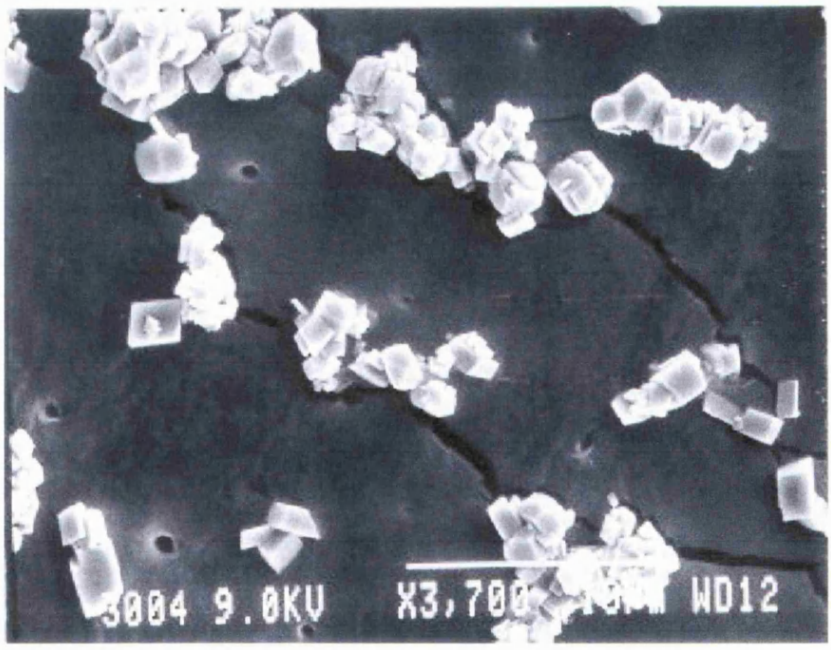

Figure 4.12: View of the precipitate at a later stage of the process (a)

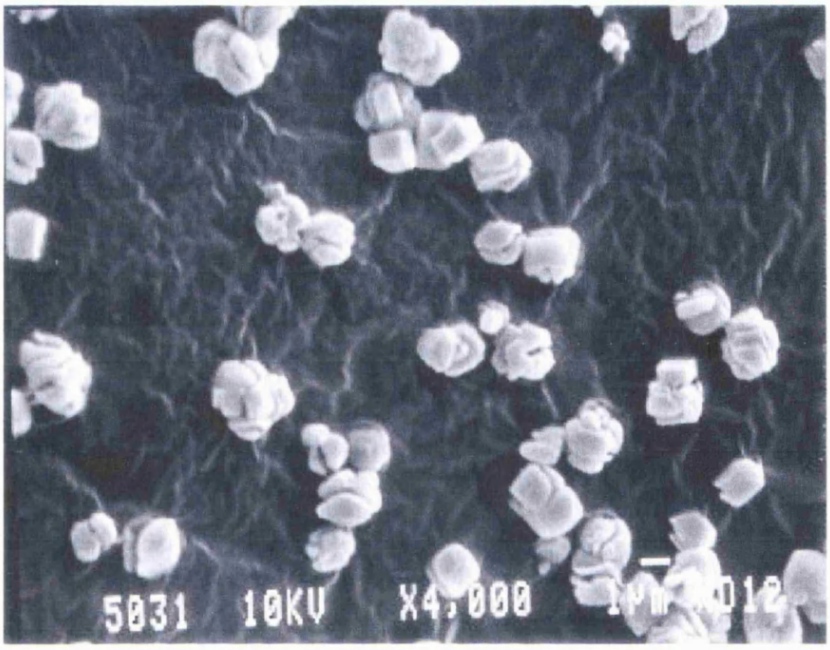

Figure 4.13: View of the precipitate at a later stage of the process (b)

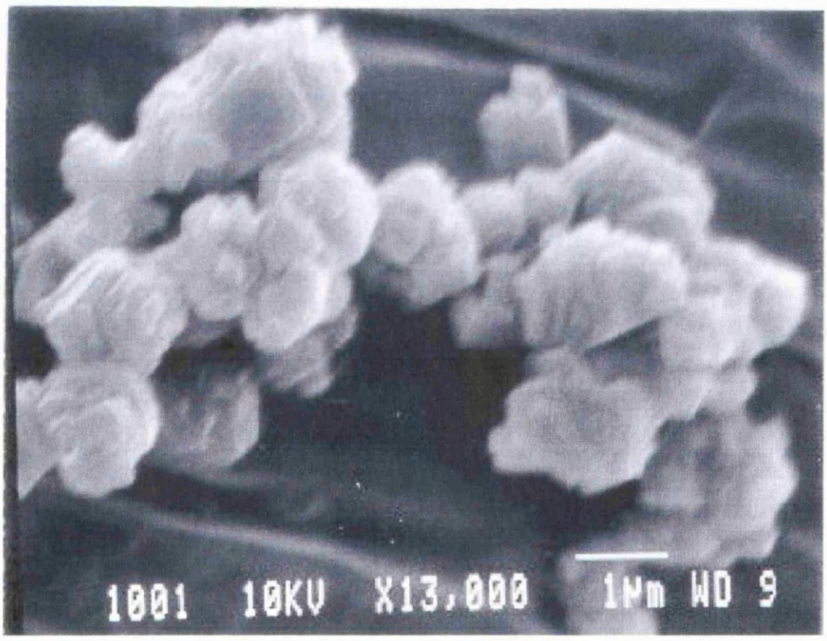

Figure 4.14: Close view of an agglomerate, at a later stage stage of the process

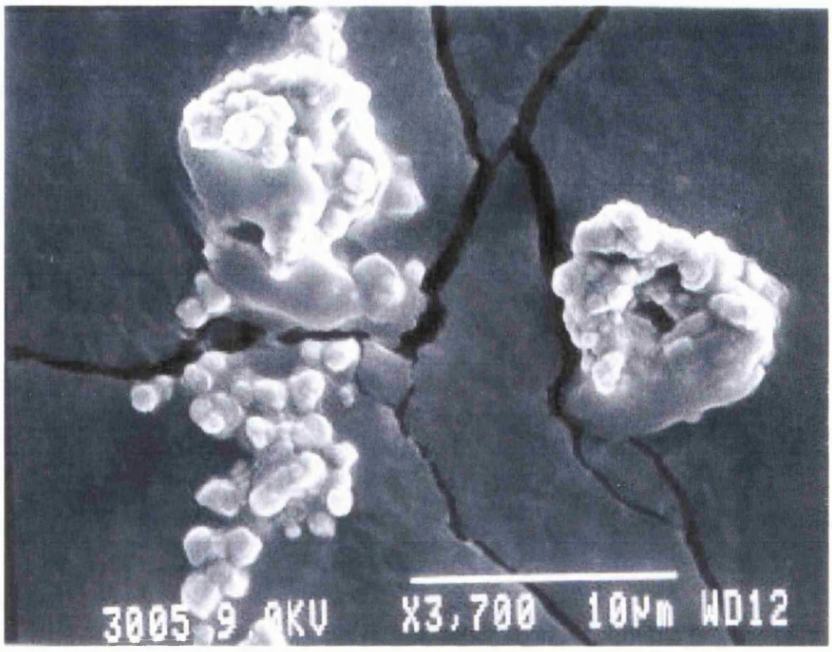

Figure 4.15: An agglomerate during a very late stage of the process.

experiments (Jones, Wachi and Delannoy, 1992; Wachi and Jones, 1995) and was previously explained in the following way: During the high pH stage particles experience slow growth, while agglomeration is being inhibited by the electrical charge on the particles' surface. Once the pH has fallen, the mean size increases rapidly due to agglomeration (Wachi and Jones, 1991). This interpretation, however, leaves an important question unanswered: how does agglomeration occur after the depletion of supersaturation, which is necessary to create solid bridges and unify the loose aggregates. Furthermore, the experiments of Kotaki and Tsuge (1990) on the same system clearly evidenced agglomeration at high pH; our own experiments also detected agglomerates during early stages (fig. 4.11).

The evolution of crystal shape provides vital clues for the understanding of the process. The loss of the sharp edges in the agglomerates, coinciding with the reappearence of free Ca^{++} after the pH drop suggests that the crystals are redissolving. Edges exhibit more rapid dissolution, thus leading to the almost amorphous material of fig. 4.15. The redissolution can be explained by the shift of the equilibrium from $CO_3^{=}$ towards HCO_3^{-}. At high pH, when $CO_3^{=}$ is favoured, a balance is maintained between the migration of ions from the crystal lattice to the surrounding solution

and deposition of ions in the lattice. After the pH drop, however, the $CO_3^{=}$ that migrates is converted to $HCO3^-$ and, as a result, the balance is disrupted. The overall dissolution process can be described by the following reaction:

$$CaCO_3 + H_2O \leftrightarrow Ca^{++} + HCO_3^- + OH^-$$

The dissolution of a crystal is proportional to its surface area. Smaller crystals with a higher surface area to volume ratio are going to disappear faster, leaving only the larger agglomerates in the system. This explains the fact that agglomerates dominate the precipitate after pH drop, despite the fact that supersaturation has been depleted. The increase in the mean crystal size after the pH drop is therefore only an apparent growth.

Bimodality in the CSD is usually explained by agglomeration or breakage. Though agglomerates have indeed been detected in the precipitate through SEM, the fact that the second peak appears at the small size range indicates that agglomeration is not the cause. Disruption of the weak aggregates is another candidate, but it must be ruled out by the fact that the new peak appears at sizes smaller than the size of the single crystals that comprise the agglomerates ($\approx 1\mu m$).

The crystals comprising the new peak must, therefore, be produced by a new wave of nucleation-growth. This could be the effect of secondary nucleation, which is proportional to the existing crystal mass (Botsaris, 1976; Garside and Davey, 1980). Size-independent growth imposes a third-order polynomial increase in crystal mass with respect to time. It is reasonable, then, that at some point a sharp increase in crystal mass will rapidly generate a large amount of secondary nuclei, thus inducing a new wave of crystal growth.

To confirm this speculation, an experiment was conducted during which the introduction of CO_2 was halted just before the new peak was expected to appear. N_2 continued to be introduced, maintaining the total gas flowrate constant, to ensure that the turbulent conditions would remain unchanged. After 30 min the distribution remained unchanged (i.e. single peak), indicating that disruption or breakage alone of the existing crystals does not account for the new peak. As soon as CO_2 was introduced again the new peak appeared, thus proving that it consists of freshly grown particles and does not occur without the presence of supersaturation.

4.3 Process simulation

4.3.1 Kinetics of $CaCO_3$ precipitation

4.3.1.1 Reaction system

The reaction of CO_2 into solutions of alkalis has been studied extensively, as it is both a commonly encountered process (gas cleaning, carbonate salts production, geophysical precipitation) as well as a classical example of the interaction between mass transfer and chemical reactions. For that reason it was used in chapter 2 to demonstrate the methodology for gas-liquid reactor modelling. Its kinetics will be examined in more detail in this chapter, in order to form a detailed kinetic model of $CaCO_3$ precipitation. Extensive reviews of this reaction system have been contributed by Danckwerts and Sharma (1966), Astarita (1967), Danckwerts (1970) and Juvecar and Sharma (1972).

The first step to consider in a CO_2 absorption system is the gas-liquid equilibrium, which determines the boundary condition for the volatile species:

$$CO_{2(g)} \rightarrow CO_{2(aq)} \qquad [i]$$

It is now generally accepted that the main resistance to mass transfer lies in the liquid phase (Juvekar and Sharma, 1972). Thus, step 1 can be approached by an equilibrium equation such as Henry's law:

$$Y_{CO2} \cdot P = \Re \cdot [CO_{2(aq)}] \tag{4.3}$$

In a system including several ionic species, however, it is advisable to take into account the influence of ionic strength (Danckwerts, 1970).

$$log\frac{H}{H_w} = -\sum_i I_i h_i \tag{4.4}$$

$$h_i = h_+ + h_- + h_g \tag{4.5}$$

The values for the contributions of cations (h_+), anions (h_-), gas (h_g) and compounds (h_i) are taken from Pohorecki and Moniuk (1988).

The kinetics of CO_2 absorption into alkali solutions are determined by the conversion of $CO_{2(aq)}$ into HCO_3^-, a reaction that proceeds at a fast, but finite rate.

This is followed by an instantaneous ionic reaction:

$$CO_{2(aq)} + OH^- \rightleftharpoons HCO_3^- \qquad [ii]$$

$$HCO_3^- + OH^- \rightleftharpoons CO_3^= \qquad [iii]$$

The precipitation of $CaCO_3$ may be written concisely as:

$$\mathrm{Ca}^{++} + CO_3^= \rightleftharpoons CaCO_{3(s)} \qquad [iv]$$

The differential equations describing the evolution of concentrations can be written as:

$$\frac{d[CO_{2(aq)}]}{dt} = r_1 - r_2 \tag{4.6}$$

$$\frac{d[OH^-]}{dt} = r_2 - r_3 \tag{4.7}$$

$$\frac{d[HCO_3^-]}{dt} = r_2 - r_4 \tag{4.8}$$

$$\frac{d[Ca^+ +]}{dt} = r_4 \tag{4.9}$$

The rate r_1 is the rate of mass transfer. It can be calculated from the penetration theory according to the methodology presented in Chapter 2: after the end of each contact time, the new concentration profiles in the interface are integrated along the penetration depth.

r_2, r_3 are the kinetic rates of steps [ii] and [iii] respectively, given by the following equations:

$$r_2 = k_2[CO_{2(aq)}][OH^-] - \frac{k_2}{K_2}[HCO_3^-] \tag{4.10}$$

$$r_3 = k_3[HCO_3^-][OH^-] - \frac{k_3}{K_3}[CO_3^=] \tag{4.11}$$

The values of the kinetic and equilibrium constants are due to Astarita (1967), are shown in table 2.1. The kinetic constant of the instantaneous reaction (iii) is approximated by a very high value. The ionic equilibrium of the CO_2 - OH^- system may involve several ionic species, depending on the pH. In this work we are interested in the alkaline range, where $CO_3^=$, which contributes to the precipitation process, is present at a significant proportion. Fig. 2.15 in chapter 2 shows the molar fractions of $CO_{2(aq)}$, HCO_3^-, $CO_3^=$ at equilibrium conditions, as a function of pH, obtained by for different values of pH in the range 7-13.

Expressing the precipitation step by means of the chemical equation (iv) obscures the underlying crystallisation mechanisms that determine its rate. The rate of $CaCO_{3(s)}$ production is essentially that of volumetric crystal growth, but this is size-dependent even when the linear growth is size-independent (McCabe's δL law). To obtain the rate of change for the whole crystal mass, we need to integrate the volumetric growth function over the whole range of individual crystal volumes:

$$r_4 = \frac{d[CaCO_{3(s)}]}{dt} = \int_{v_0}^{v_\infty} G(v)n(v)\,dv \cdot \frac{\rho_{CaCO_3}}{MW_{CaCO_3}} \tag{4.12}$$

Note how the CSD enters this expression, in the form of the population density $n(v)$. Thus the rate of crystal mass production is inherently coupled with the population balance, and to estimate it we need a complete kinetic model of precipitation that accounts for all crystallisation mechanisms, namely nucleation, crystal growth and interparticle phenomena such as agglomeration and breakage. In this work, the population balance will be discretised with the aid of the conservative finite element scheme developed in chapter 3. That method transforms the integro-partial differential PBE into a series of ordinary differential equations, in terms of the nodal values of population densities (eq. 3.39). Thus the final system of mass and population balance ODEs to be solved takes the form:

$$\frac{dn(v_i,t)}{dt} = f_1\left(n(v_1), n(v_2)...n(v_n), C_1, C_2, ...C_n\right) \tag{4.13}$$

$$\frac{dC_i(t)}{dt} = f_2\left(n(v_1), n(v_2)...n(v_n), C_1, C_2, ...C_n\right) \tag{4.14}$$

The next task is to select the kinetic laws to describe the rates of the crystallisation mechanisms.

4.3.1.2 Supersaturation

As we saw in chapter 1, most of the crystallisation mechanisms are heavily dependent on superaturation. This is particularly true of precipitation, where the high degree of supersaturation gives rise to rapid nucleation and growth rates. The role of supersaturation dependency is pronounced in batch experiments, where the species' concentrations vary with time. Consider a general ionic reaction of the type:

$$\alpha A^{\beta+} + \beta B^{\alpha-} \rightleftharpoons A_\alpha B_{\beta(s)}$$

Among the various measures of supersaturation reviewed in chap. 1, the most appropriate one for ionic precipitation processes is the relative supersaturation, λ_s:

$$\lambda_s = \frac{C_A^\alpha C_B^\beta}{K_{sp}} \tag{4.15}$$

where K_{sp} is the solubility product:

$$K_{sp} = [A^{\beta+}]^\alpha \, [B^{\alpha-}]^\beta \tag{4.16}$$

It is generally agreed that the driving force of supersaturation is best related to $\sqrt[\alpha+\beta]{\lambda_s} - 1$ (Bhatia, 1989). Furthermore, in an ionic solution, it is preferable to express supersaturation in terms of activities (Chakraborty and Bhatia, 1997; Collier and Hounslow, 1999). Thus the measure of supersaturation to be used in this work is:

$$\sqrt{\lambda_s} - 1 = \sqrt{\frac{\alpha_{Ca^{++}} \alpha_{CO_3^{=}}}{K_{sp}}} - 1 \tag{4.17}$$

Activities are calculated according to the Debye-Huckel equation, with a modification proposed by Davies (1962):

$$\alpha_i = \gamma_i \cdot C_i \tag{4.18}$$

$$log(\gamma_i) = -A_{DH} \, z_i^2 \, (\frac{\sqrt{I}}{1+\sqrt{I}} - 0.3I) \tag{4.19}$$

where I is the ionic strength, given in terms of the concentration and charge on each ionic species:

$$I = \frac{1}{2} \sum_i C_i z_i^2 \tag{4.20}$$

4.3.1.3 Crystallisation kinetics

The so-called inverse problem in crystallisation, i.e. the determination of crystallisation kinetics from CSD data, is very difficult to solve due to the fact that several mechanisms (nucleation, primary and secondary, growth, agglomeration, breakage) may be simultaneously active, and all of them are non-linear functions of supersaturation. The situation becomes more difficult in a batch experiment, where different mechanisms may prevail during different time periods. Most methods for kinetics extraction are based on steady-state experiments and study of a single mechanism in isolation.

Growth can be isolated by means of carefully conducted experiments, e.g. by using seeds and maintaining supersaturation within the metastable zone to suppress nucleation. Crystal growth of calcite has been extensively studied, and relative consensus seems to exist. The growth of an ionic crystal, such as $CaCO_3$, may be limited by either diffusion of ions to the crystal surface, or their subsequent reaction and integration into the lattice. If the precipitated crystals are small ($< 20\mu m$), sparingly soluble and in a well mixed suspension, the diffusion is bound to be very fast and the surface integration is usually the rate limiting step (Collier and Hounslow, 1999). Several researchers (Hostomsky and Jones, 1991; Collier and Hounslow, 1999; Tai and Hsu, 2001) have confirmed that in precipitation of calcite surface integration is indeed the rate limiting step. The following expression is usually employed to describe growth rate kinetics:

$$G = k_g(\lambda_s - 1)^2 \tag{4.21}$$

The experiments of the above authors seem to agree on the order of magnitude and, to a certain degree, on the values of the parameter k_g. The value used in this work is that of Collier and Hounslow (1999), shown in table 4.1.

Nucleation, on the other hand, is much more difficult to study due to the variety of mechanisms (homogeneous, heterogeneous, secondary) that may be responsible for it. A review of these mechanisms has been presented in chapter 1. Its kinetics are usually studied by examining the number of crystals produced during the induction time. The more rigorous classical theories of homogeneous nucleation are seldom applicable to industrial equipment, and empirical correlations are usually preferred. Primary nucleation depends mainly on supersaturation, and a power law of the following form is commonly encountered:

$$B_0 = k_{n1}(\lambda_s - 1)^{k_{n2}} \tag{4.22}$$

Secondary nucleation is induced by the existing crystals (Botsaris, 1976; Garside and Davey, 1980), and power laws describing it must involve the crystal mass as well:

$$B_0 = k_{ns}(\lambda_s - 1)^{p_{ns1}} M_c^{p_{ns2}} \tag{4.23}$$

Although several studies on calcium carbonate have appeared (Pakter, 1968; Swinney, Stevens and Peters, 1982; Koutsoukos and Kontoyannis, 1983), their results

cannot be extrapolated to equipment of different type and scale, where nucleation may occur via a different mechanism. Moreover, relatively few data have been reported on gas-liquid systems. Most important is the fact that during a batch process, different nucleation mechanisms may prevail during different periods. In our case, the experiments have indicated that in the beginning high supersaturation levels induce primary nucleation, but later, secondary nucleation gives rise to a new wave of crystal growth. The two mechanisms are fundamentally different, and may operate independent of each other. To account for these facts, our overall nucleation model consists of the sum of the two models, primary and secondary:

$$B_0 = k_{n1}(\lambda_s - 1)^{k_{n2}} + k_{n3}(\lambda_s - 1)^{k_{n4}} M_c^{k_{n5}} \tag{4.24}$$

The determination of agglomeration kinetics is confronted with the problem of distinguishing it from growth. Usually apparent growth is reported, i.e. the combined effect of growth and agglomeration. Bramley et al. (1996) showed that, in the case of size independent growth, agglomeration can be decoupled from growth, but their method applies to MSMPR experiments.

In chapter 1 agglomeration models were reviewed. It was concluded that no mechanistic model has so far succeeded in accounting for both aspects of the agglomeration process, i.e. the formation of flocculates due to collisions and interparticle attraction and the subsequent growth of crystalline material between the clusters at spots called cementing sites. The main obstacles were indentified to be the following:

- Particles of various states (flocculates, aggregates of various ages, firm agglomerates) are likely to be present in a crystal agglomerating system. CSD measurements, however, give no information on these states.

- The interparticle phenomena that modify the CSD are also various. Unsuccesful collisions, disruption of aggregates and breakage of firm agglomerates are all possible, depending on the conditions, and it is very difficult to distingush between these events.

- Current mechanistic models tend to consider an aggregate as a two-particle system, bridged at a cementing site; in reality, however, it may be a multiparticle cluster with bridges of various ages and strengths.

- The size-independent kernel seems to be in a good agreement with most agglomeration experiments in stirred tanks, and is used extensively. This agreement, however, is likely to be due to mutual cancellation of competing mechanisms, and we do not know whether it would be maintained in an arbitrary environment.

As a result, agglomeration kinetics are heavily system-dependent and difficult to scale-up, let alone apply to different kinds of equipment. The dependency of the agglomeration kernel on hydrodynamics has been investigated mainly in stirred tanks (Hounslow et al., 2000), whose flow field is very different from that of bubble columns. For this reason, no explicit dependency on hydrodynamics will be employed here; instead, the agglomeration constant will be adjusted to fit experiments with different gas flowrates.

Among the most common theoretical agglomeration kernels, we found the fluid shear kernel (orthokinetic aggregation) to provide the best fit of our experiments. This is in good aggreement with the theory, but in contrast with most of the experimental data on stirred tanks, which showed the size-independent kernel to be more suitable (Hostomský and Jones, 1991; Hounslow et al., 2000). It must be born in mind, however, that when agglomeration is the only inter-particle event in the population balance, the results refer to the net effect of an agglomeration-disruption equilibrium. Stirred tanks exhibit higher local shear forces, and therefore disruption of the weak, not fully cemented aggregates (especially the bigger ones) is likely to occur more often, yielding an apparent size-independent agglomeration. On the other hand, the more gentle agitation in the bubble column may be the reason why the effect of the size-dependent kernel is brought out.

MSMPR experiments (Hounslow et al., 2000) have shown agglomeration to be roughly proportional to growth. The underlying cause of both phenomena is supersaturation, and during a batch experiment supersaturation is varying. It seems appropriate, then, to correlate agglomeration to supersaturation directly. Given the second order dependency of growth on supersaturation, we may infer that a similar dependency to the agglomeration model. Our formulation thus is:

$$\beta_a(v', v - v') = k_a(\lambda_s - 1)^2[v'^{\frac{1}{3}} + (v - v')^{\frac{1}{3}}]^3 \tag{4.25}$$

Apart from the growth kinetics, the remaining parameters must be adjusted to fit the experiments. This is especially difficult in a batch experiment, where different mechanisms may prevail at different time periods, and each mechanism is a non-linear function of supersaturation. In our study the determination of the parameters was guided by: a) the mean size of the crystals immediately after the induction period, which is determined by the balance between primary nucleation and growth, b) the evolution of of Ca^{++} concentration, and c) the appearance of the second peak at the low size range, which indicates the effect of secondary nucleation.

Parameter, units	Value
Solubility product, K_{sp}, mol^2 m^{-6}	0.0047
Growth const., k_g, m/s	$6.2 \cdot 10^{-11}$
k_{n1} (eq. 4.24), s^{-1}	$7.5 \cdot 10^6$
k_{n2} (eq. 4.24)	6
k_{n3} (eq. 4.24), kg^{-1}	$1.6 \cdot 10^{14}$
k_{n4} (eq. 4.24)	0.1
k_{n5} (eq. 4.24)	1
Agglomeration constant, k_a, m^3/s, at $0.6 \cdot 10^{-4}$ m^3/s gas flowrate	1.2
Agglomeration constant, k_a, m^3/s, at $1.1 \cdot 10^{-4}$ m^3/s gas flowrate	1.8

Table 4.1: Parameter values used in the simulations.

4.3.2 Modelling methodology

Modelling precipitation in a gas-liquid reactor requires sufficient knowledge of its hydrodynamics. A direct coupling of chemical reactions and the population balance with a CFD code would raise extreme computational demands, however; so far, it has been attempted only with the method of moments (e.g. Wei and Garside, 1997). Moreover, distributed information about the fluid dynamic variables is excessive for a reaction engineering analysis of the process. Usually, knowledge of several key parameters such as gas hold-up and liquid circulation averaged over regions or in the whole reactor, is sufficient.

In this work, the problem will be approached with the aid of theoretical methods developed in the preceding chapters. In chapter 2, we presented a methodology for integrating hydrodynamics with complex chemical phenomena that cannot be easily

integrated with CFD. Precipitation obviously fits this description, as the population balance equation that describes it requires, after discretisation, a considerable number of differential equations to be solved alongside the hydrodynamics. Thus the advantages of the methodology are well demonstrated by this problem. Furthermore, in chapter 3 we developed a numerical method for solving the population balance equation, with particular relevance to problems including agglomeration and/or breakage. This method will be employed here for the solution of the population balance in $CaCO_3$ precipitation.

Wachi and Jones (1991a) and Hostomsky and Jones (1994), who studied precipitation in a laboratory reactor with a flat gas-liquid interface, accounted in their models for precipitation occurring in the interface as well as in the bulk. However, the laboratory reactor used in their experiments was poorly mixed (1.3 lt, 100-400 rpm), resulting in very low values of the gas-liquid mass transfer coefficient, and subsequently very high values of the contact time (2-16 sec). Under these conditions precipitation in the interface is accountable, but in a bubble column, as well as in most industrial equipment, the contact time is lower by several orders of magnitude (0.02 sec for 5 mm bubbles moving at a slip velocity of 0.25 m/sec). Therefore the effective interfacial volume is rather negligible compared to the bulk, and it seems reasonable to neglect any particulate phenomena at the interface, thus achieving a valuable reduction in CPU time.

4.3.2.1 CFD modelling and virtual tracer experiment

In gas-liquid reactors the interaction between the flow field and the chemical reactions is usually one-way, i.e. the gas-liquid flow establishes the above key parameters that determine the rate of reactions, but the reactions have no significant feedback effect on the hydrodynamics. In the case of $CaCO_3$ precipitation this is a reasonable assumption, provided that the suspension is dilute and the reactant gas is mixed with an inert component, so that the bubble size does not change significantly due to reactant depletion (as e.g. in the study of Fleicher et al., 1995). In our experiments the solids content was less than 1% and the CO_2 / N_2 ratio in the input gas ranged from 0.01 to 0.2, thus ensuring that the above assumption is valid.

The draft-tube bubble column used in the experiments resembles that of chapter

2, and its simulation with CFX was carried out in a similar manner. The virtual tracer experiment was also conducted along the lines of section 2.2.5.3, i.e. the passive scalar was introduced proportional to the local volume fraction (eq. 2.10). The results are shown in fig. 4.16.

The tracer concentration contours show a more or less uniform concentration throughout the column (note that the upper part of the column is occupied by gas, as the volume fraction contour shows, and the tracer concentration in that region is meaningless). This must be attributed to the short height of the column: due to intense circulation around the draft tube, the reactor functions as a CSTR. Minor concentration gradients in the top cannot be expected to play a major role in the outcome. This was confirmed by the experiments, where sampling in the riser and downcomer did not show any concentration difference beyond the margins of experimental error. Furthermore, the uncertainty in precipitation kinetics means that any fine detail in the hydrodynamic part of the model would be wasted. For this reason a single compartment will be employed to represent the column bulk.

4.3.2.2 Superposition of precipitation

In the preceding section it was decided that nucleation, both primary and secondary, size-independent growth and size-dependent agglomeration must be included in the model. The corresponding population balance equation, written in terms of particle volume as independent coordinate (in accordance with the method of chapter 3) is:

$$\begin{aligned}\frac{\partial n(v,t)}{\partial t} &= \frac{n_{in}(v,t)-n(v,t)}{\tau} - \frac{\partial}{\partial v}\left[G(v)\cdot n(v,t)\right] + B_0\delta(v-v_0)+ \\ &+\frac{1}{2}\int_0^v \beta_a(v-v',v')n(v-v',t)n(v',t)dv' - n(v,t)\int_0^\infty \beta_a(v,v')n(v',t)dv'\end{aligned} \tag{4.26}$$

This equation refers to a finite, fully-mixed domain, and can therefore be applied to the compartments of the generalised compartmental model. By applying the numerical method of chapter 3, it can be transformed into a series of ordinary differential equations (eqs. 3.39). Thus the final system of ODEs to be integrated has the form:

$$\frac{dC_i^B}{dt} = \sum_{k=1}^{K} r_k(C_i^B, n_j) \tag{4.27}$$

$$\frac{dn_j}{dt} = f_j(C_i^B, n_j) \tag{4.28}$$

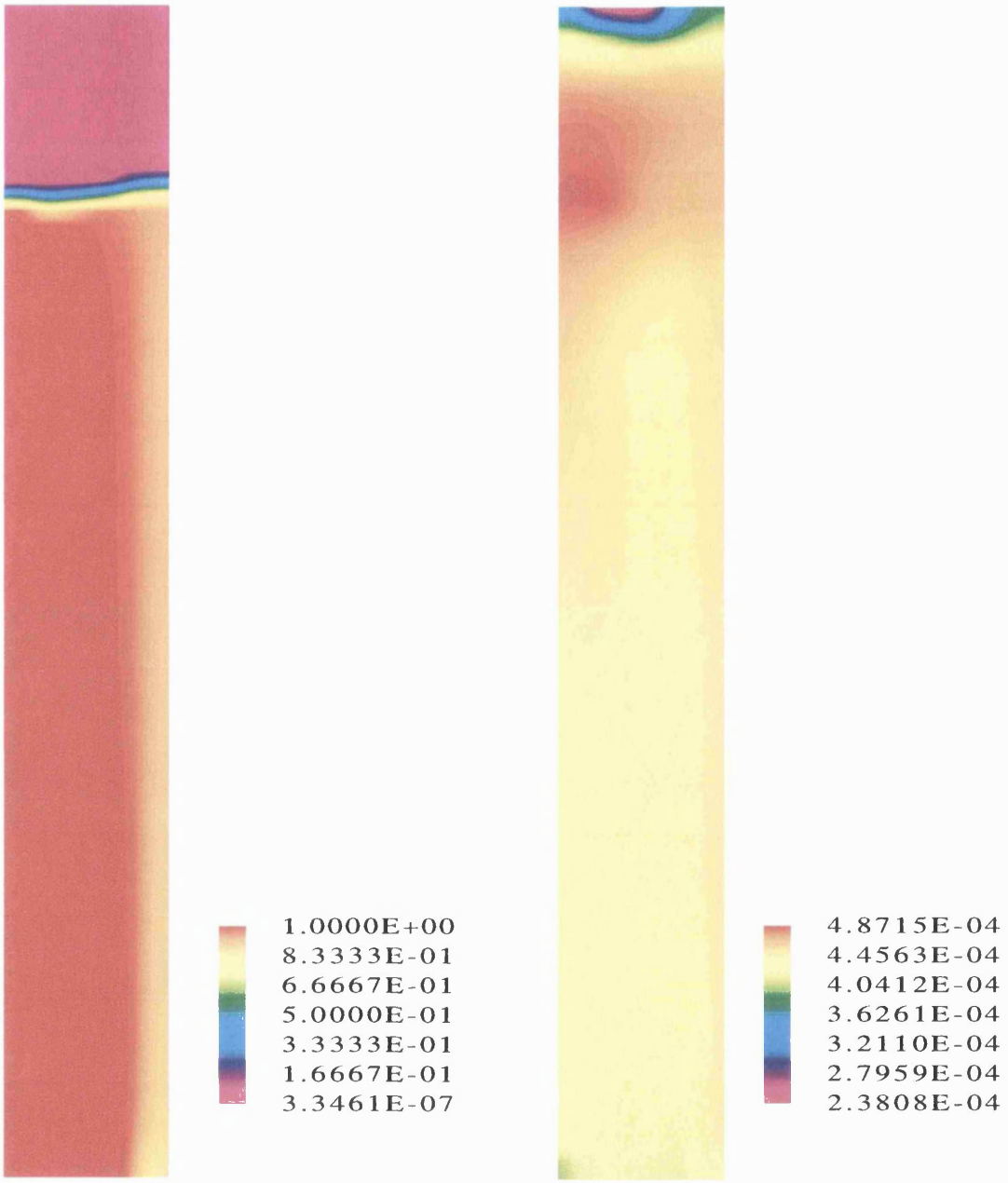

Figure 4.16: Left: shaded contours of volume fraction, right: passive scalar concentration (axisymmetric simulation, column axis on the right).

Here $f_j(C_i^B, n_j)$ stands for the right-hand side of the ODEs into which the original population balance is transformed via the finite element discretisation. Initial conditions are calculated from the mixing of bulk and interface at the end of the previous contact time. Note that the mass balances and the ODEs resulting from the population balance are coupled via the source terms. The coupling results because:

- precipitation kinetics are strongly dependent on supersaturation.
- The rate of consumption of the precipitating species is the rate of volumetric crystal growth, and must be calculated from the population balance.

4.3.3 Simulation results

Fig. 4.19 shows a comparison of the simulated and experimental results for the evolution of pH and calcium during the precipitation experiments. Overall, the time course of the process is well simulated. The measured pH curve seems to stabilize at a higher pH (7-8) than the predicted one, which falls down to pH 5-6. This is probably due to the kinetic model of the CO_2 - OH^- system, which included only those subreactions that are important in the high pH regime. At near-neutral pH, other ionic species apart from the ones considered ($CO_3^=, HCO_3^-$) may be present; in addition, the pH meter was calibrated for the alkaline region. Any further discrepancies may be attributed to either the kinetics or the gas hold-up predictions.

The Ca^{++} plot is indicative of the precipitation phenomena. During the induction time, i.e. the time during which the nuclei are formed and grow to observable size, consumption of substrate is minimal and Ca^{++} is constant. Subsequently, a rapid drop in Ca^{++} concentration indicates high growth rate. This is due to very high supersaturation levels, generated by the accumulation of $CO_3^=$ during the induction period. Rapid growth implies rapid consumption, and supersaturation soon falls to low levels, resulting in lower growth rate. This rate is almost constant till the pH drop, indicating a balance between consumption and generation of supersaturation. The redissolution of the crystals, resulting in the reappearance of free Ca^{++}, is not included in the model. The above phenomena are reflected in the supersaturation plot, shown in fig. 4.20: rapid increase during the induction time is followed by a rapid drop. Then supersaturation is constant, until dropping to zero

due to depletion of Ca^{++}.

Figures 4.21, 4.22, 4.23, 4.24, 4.25, 4.26 show comparisons of the experimental CSD measurements and the simulation results. In most cases, the main trends are reproduced, though agreement is not quantitative. The first CSD measurement usually exhibits the highest deviation from the modelling results, partly due to experimental error. This measurement was usually taken immediately after the induction time, when the solution was still too dilute, and the obscuration of the Mastersizer S was below the limit required for a reliable measurement; large foreign particles would easily shift the CSD towards larger sizes. This has happened in fig. 4.23, where the first measurement showed a much higher proportion of big particles than subsequent ones, which is clearly unrealistic. The cumulative undersize distributions (figs. 4.22, 4.24, 4.26) reveal that the deviation occurs mainly at the larger sizes, sometimes greater even than $20\mu m$, but such crystals were seldom observed by the SEM, which supports the possibility of the CSD being biased by foreign particles. The intermediate and final measurements, though, are in much better agreement with the predictions. Most important, the combination of secondary nucleation and agglomeration succeeds in reproducing the bimodality in the CSD.

The lack of quantitative agreement can be attributed to the uncertainty in the precipitation kinetics, since the reaction engineering model was validated by the concentration predictions. This was to be expected, as the precipitation mechanisms have not been adequately studied in the gas-liquid environment. The modelling of agglomeration seems to pose the most difficult obstacles, due to the lack of a definitive mechanistic model of the cementation step. This problem is especially pronounced in this study, due to two reasons: a) in a batch process the agglomeration kinetics are time-varying, depending on supersaturation; and b) the hydrodynamic environment that determines the aggregation-disruption equilibrium (or aggregation efficiency) in gas-liquid flows is fundamentally different from that in stirred tanks. The agglomeration model used in this study takes supersaturation into account, and thus achieves reasonable prediction of the extent and time scale of the phenomena during the semi-batch process. The sustaining or disruption of the weak aggregates is not mechanistically incorporated, however, and this may be the reason why the exact shape of the CSD is not accurately reproduced.

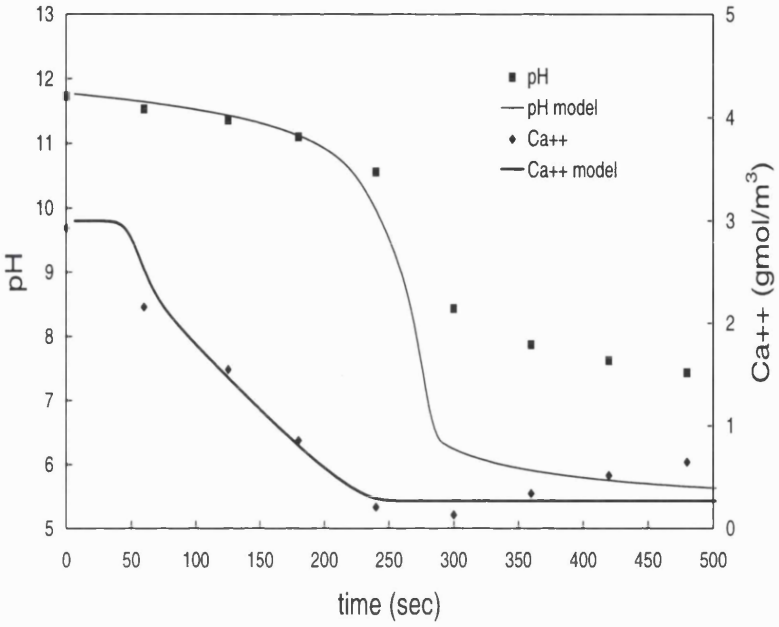

Figure 4.17: Experimental and simulated evolution of pH and [Ca^{++}]: [$Ca(OH)_2$] = 3 mol/m^3, 0.00001 : 0.0001 m^3/s CO_2:N_2

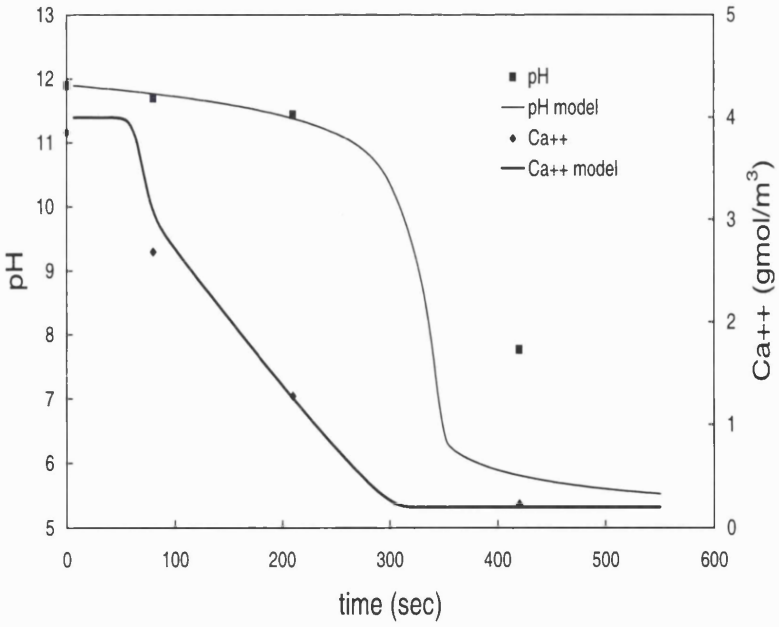

Figure 4.18: Experimental and simulated evolution of pH and [Ca^{++}]b)[$Ca(OH)_2$] = 4 mol/m^3, 0.00001 : 0.0001 m^3/s CO_2:N_2

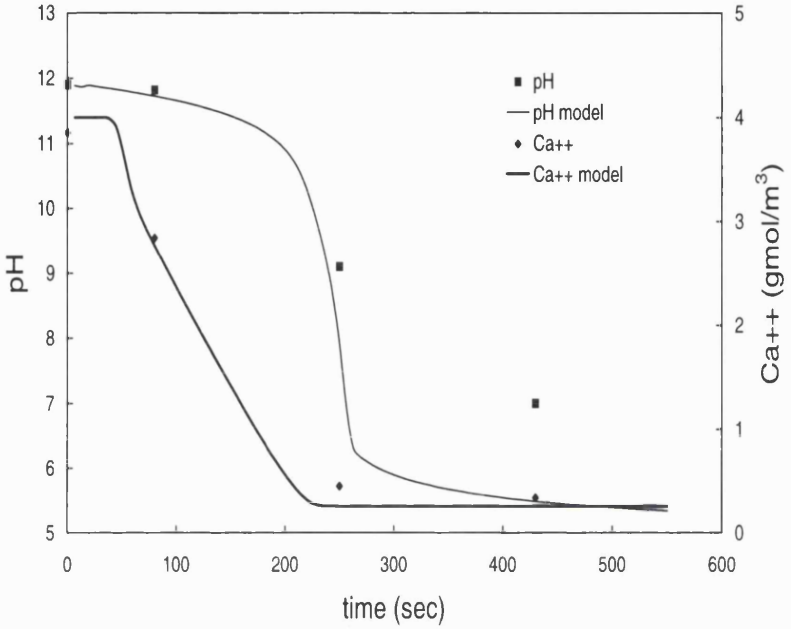

Figure 4.19: Experimental and simulated evolution of pH and [Ca^{++}]: c)[$Ca(OH)_2$] = 4 mol/m^3, 0.00001 : 0.00005 m^3/s CO_2:N_2

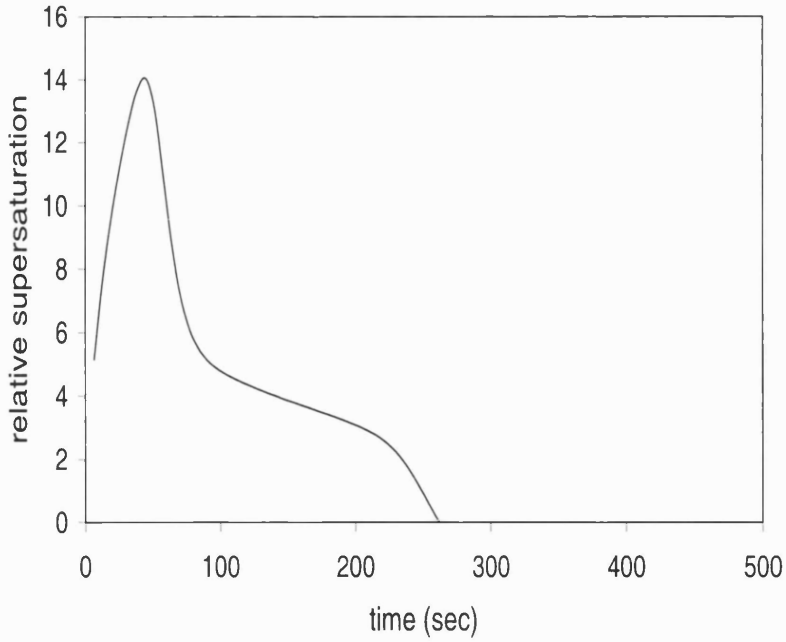

Figure 4.20: Simulation of the evolution of supersaturation in a typical experiment.

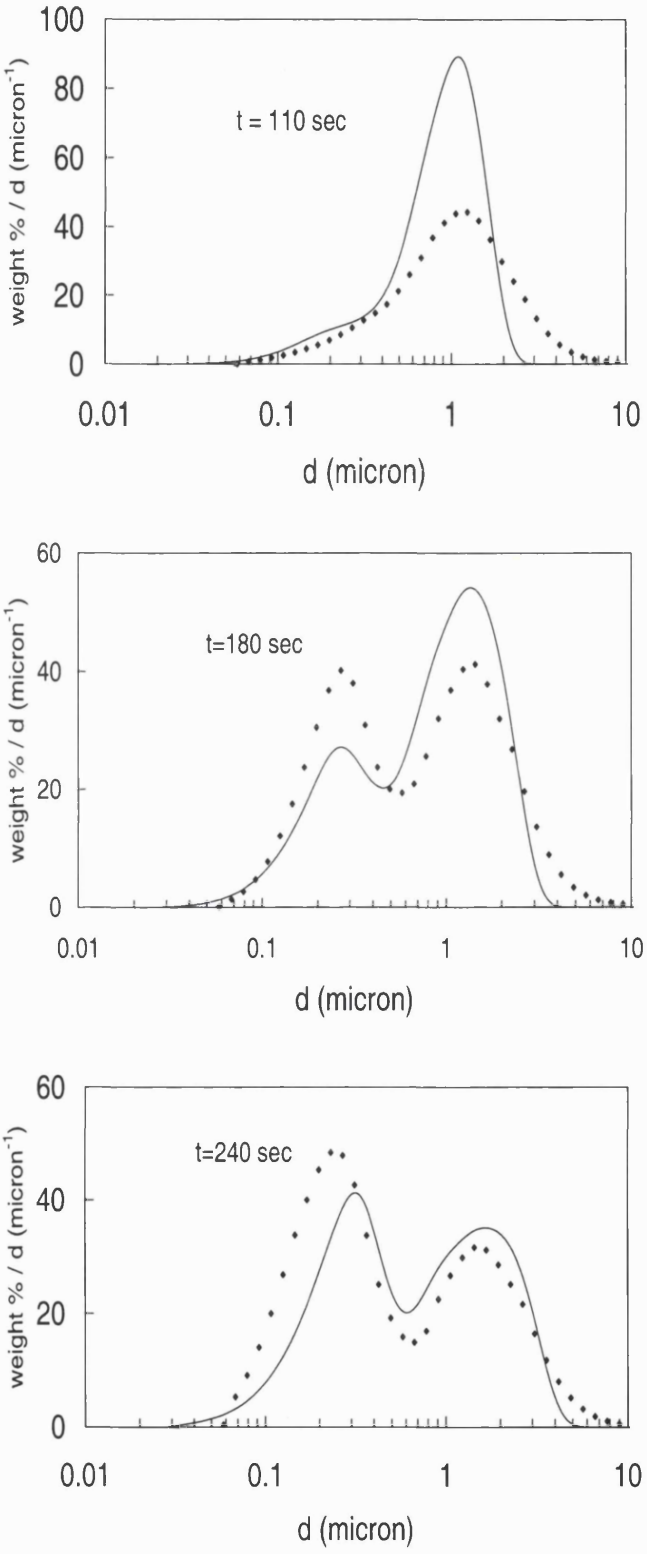

Figure 4.21: Experimental and simulated CSD, logarithmic plot. $[Ca(OH)_2] = 3$ mol/m^3, $0.00001 : 0.0001\, m^3/s$ CO_2:N_2.

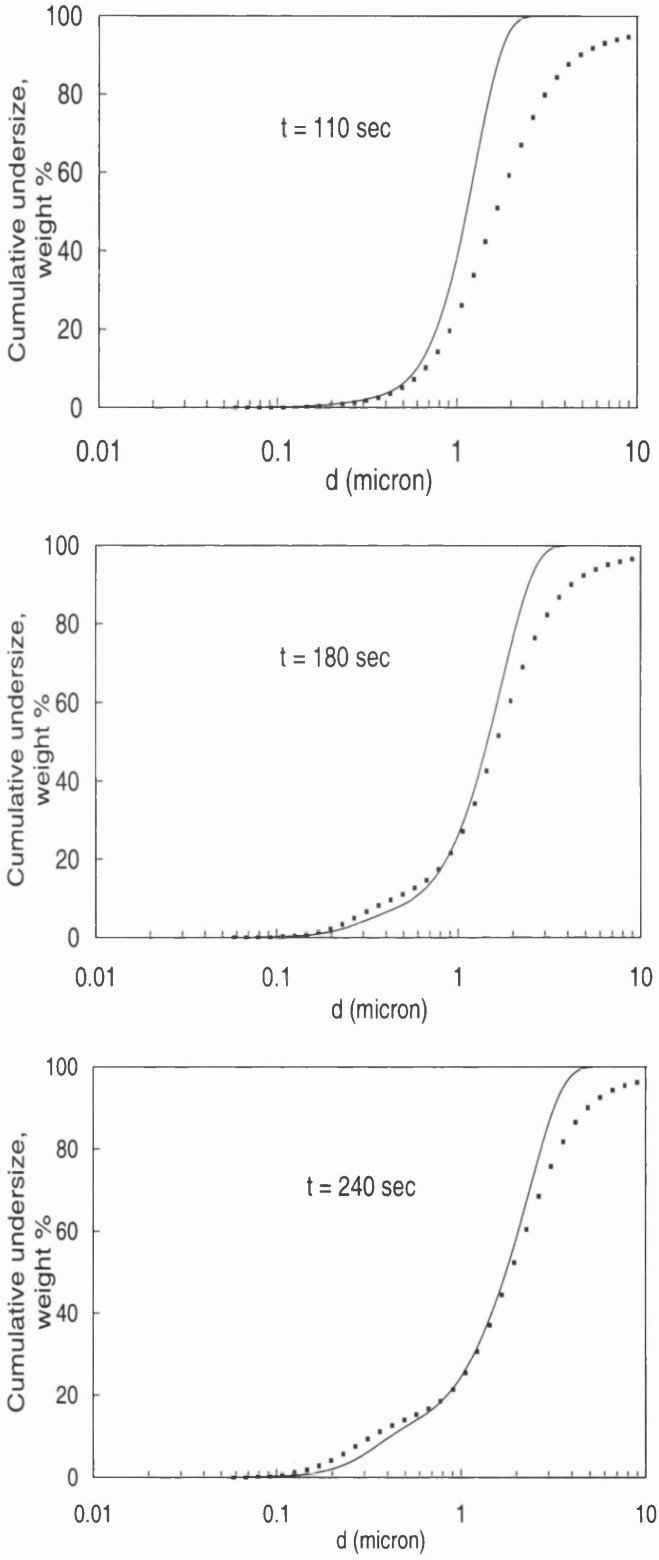

Figure 4.22: Experimental and simulated CSD, cumulative undersize plot. $[Ca(OH)_2] = 3\ mol/m^3$, $0.00001 : 0.0001\, m^3/s$ CO_2:N_2.

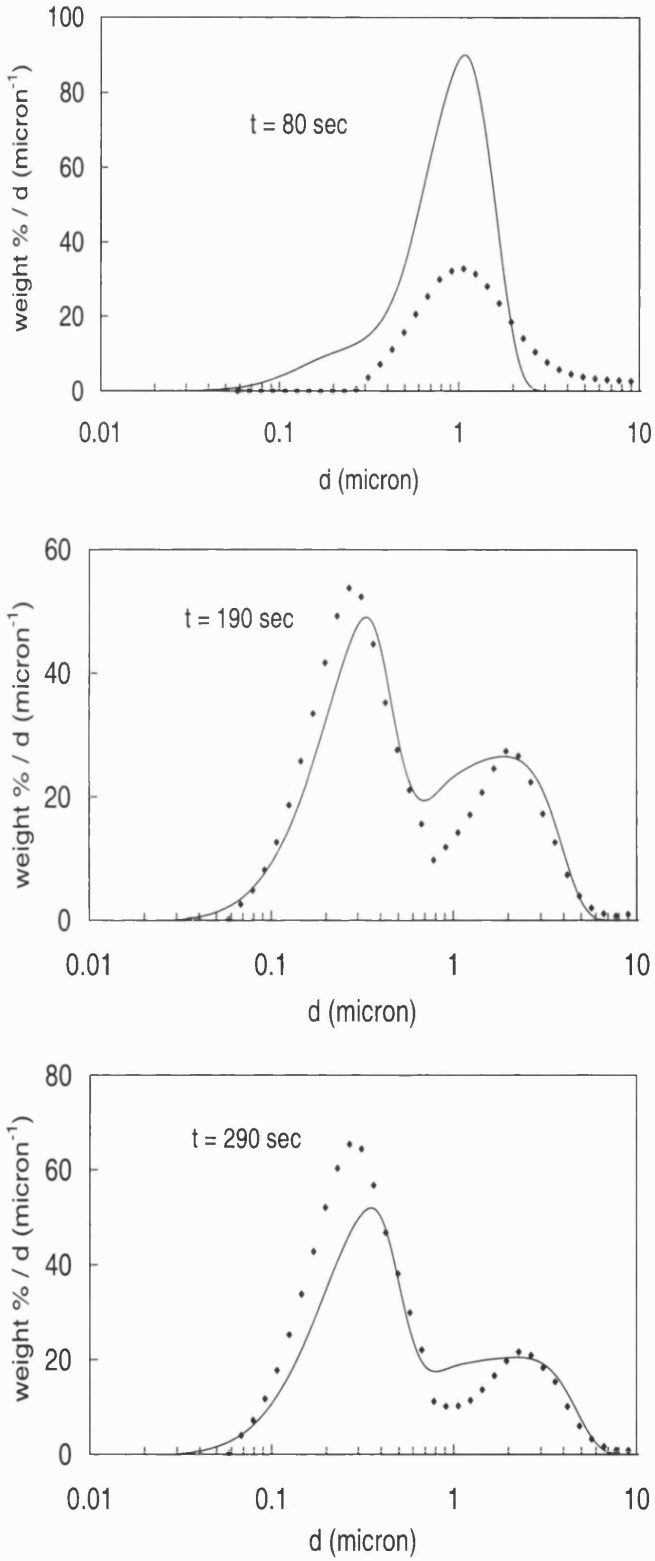

Figure 4.23: Experimental and simulated CSD, logarithmic plot. $[Ca(OH)_2] = 4$ mol/m^3, $0.00001 : 0.0001\, m^3/s$ CO_2:N_2

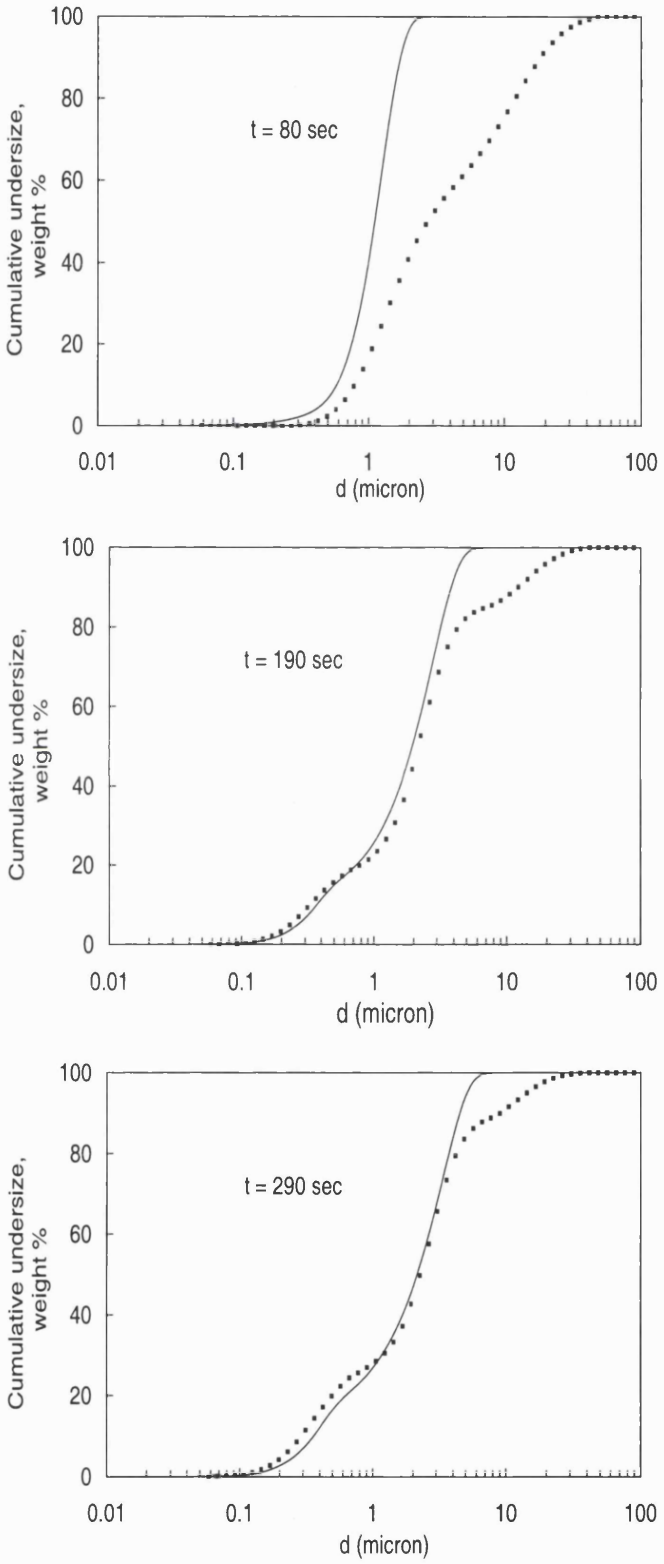

Figure 4.24: Experimental and simulated CSD, cumulative undersize plot. $[Ca(OH)_2] = 4\ mol/m^3$, $0.00001 : 0.0001\, m^3/s$ CO_2:N_2

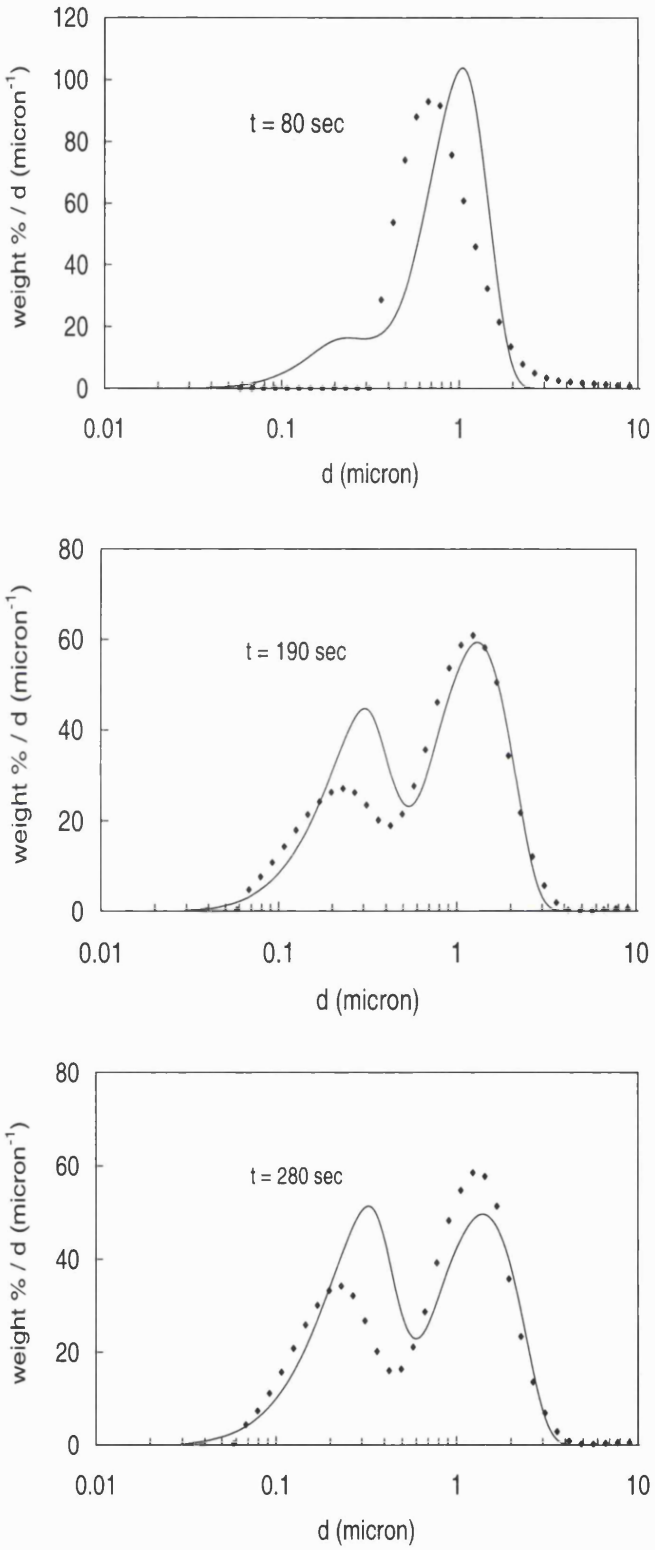

Figure 4.25: Experimental and simulated CSD, logarithmic plot. $[Ca(OH)_2] = 4$ mol/m^3, $0.00001 : 0.00005\, m^3/s$ CO_2:N_2.

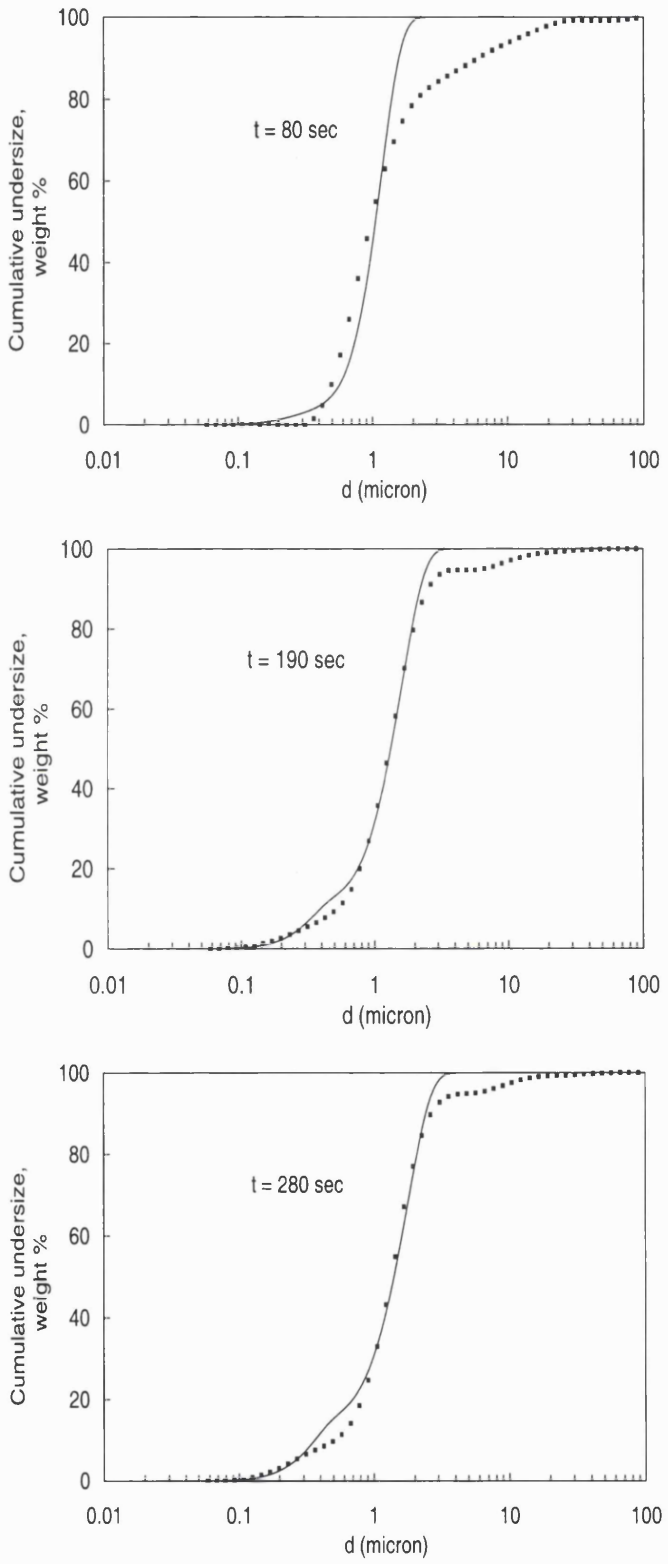

Figure 4.26: Experimental and simulated CSD, cumulative undersize plot. $[Ca(OH)_2] = 4\ mol/m^3$, $0.00001 : 0.00005\, m^3/s$ CO_2:N_2.

Chapter 5

Conclusions

This work has addressed fundamental issues in two fields of chemical engineering: the merging of fluid dynamics and chemical phenomena in multiphase reactors, and the population balance modelling of particulate processes. With respect to the first, a hybrid methodology that integrates CFD and reaction engineering principles was proposed; with respect to the second a numerical method based on finite elements was developed, that resolves several issues regarding conservation and stability.

The advances made in these fields were subsequently applied to a problem whose analysis is a case study of both: gas-liquid precipitation, i.e. the formation of a crystalline solid product via the reaction of a gas and a liquid. The conjunction of CFD, reaction engineering and population balance analysis has yielded a fully predictive and yet tractable model, that could be used for process design and scale-up purposes if supplied with accurate data on precipitation kinetics. Moreover, the detailed predictions of the evolution of crystal size distribution and concentrations, have yielded insight into the nature of the underlying phenomena. The main findings in each one of these fields will now be summarised.

5.1 Hybrid modelling of multiphase reactors

The main concept of this methodology is that mixing, both due to mean flow and turbulent exchange at the meso- and macro- scale, can be accounted for via averaging the flow and concentration fields resulting from a CFD flow simulation and a computational tracer experiment, and using these results to parametrise a generalised

compartmental model. The approach introduces no new phenomenological concepts, and its accuracy depends on the accuracy of the CFD (whose main phenomenological aspects are the turbulence and multiphase flow models) and the fine-tuning between the CFD and compartmental model tracer simulations. The former will benefit from any future advances; the later depends on the time and effort the process designer can afford to invest. Application to bubble column reactors was accomplished here by superimposing a phenomenological model of interfacial mass transfer and chemical reaction on the generalised compartmental model. The main concept, however, is more broad and may be applied to several other fields such as stirred tank reactors (in conjunction with phenomenological micromixing models), biochemical and precipitation reactors (in conjunction with population balance models).

5.2 Solution of the population balance with finite elements

In this work, a finite element scheme was proposed for solving dynamic population balance problems. The PBE and boundary conditions were formulated to accommodate an arbitrary problem involving any combination of the four main particulate mechanisms: nucleation, growth, aggregation and breakage. The numerical scheme consists of collocation on linear elements with an "upwind" propagation of first-order derivatives (i.e. growth) that result in enhanced stability. Integral terms are evaluated with simple Newton-Cotes quadrature, and a three-point formula (Simpson's rule) requiring just one interpolation was found to be sufficient for most kernels. Three sources of error in the conservation of moments were identified: quadrature error, effect of non-uniform grid, and error due to temporal integration, respectively. A rule for determining the order of quadrature was proposed to eliminate the first source of error, while an accurate way of evaluating the aggregation source term, based on a geometric interpretation of it, was developed to mitigate the second. Although the error cannot be completely eliminated, the results demonstrated that if the above rules are followed, it will be negligible.

The dynamic method was capable of reaching steady-state solutions with ease and speed. Nicmanis and Hounslow (1998), who studied the steady-state PBE

with finite elements, reported numerical problems such as matrix ill-conditioning in growth problems which could only be resolved by resorting to Galerkin methods, as well as difficulties in inverting the dense matrices resulting from the discretisation. On the other hand the dynamic method, guided by the solution at the previous time step, proved very easy to converge even with collocation and explicit integration, in which case no matrix inversion is required. This is a commonly encountered situation in numerical analysis: a steady-state problem is easier to solve by considering the dynamic case.

When comparing any finite element scheme with DPBs, it must be born in mind that the mechanisms of aggregation and breakage are essentially discrete, and therefore DPBs are a more natural choice for them. Moreover, being zero order, DPBs can accommodate more easily certain situations that are ill-conditioned for higher order methods (e.g. monodisperse initial conditions). The value of the finite element method is better realised when considering combined problems that involve both discrete and continuous mechanisms, in which case the generality of the method allows it to tackle a variety of different problems without being tailored to the needs of individual ones. The scheme proposed here features this generality while retaining some of the advantages of DPBs (stability and ability to deal with ill-conditioned problems) due to its low-order.

5.3 Gas-liquid precipitation of $CaCO_3$

In this work population balance, reaction engineering and hydrodynamic principles were integrated into a modelling approach for the gas-liquid precipitation process. By accounting for all of the above, the model aims to provide tools for analysis and scale-up of industrial type equipment used for this class of processes. The approach was subsequently used to simulate the precipitation of $CaCO_3$ by CO_2 absorption into lime, and experiments on a laboratory-scale rig were carried out for comparison. The main findings can be summarised in the following:

- The increase in mean size after the pH drop is due to the disappearence of the smaller crystals by dissolution.
- Agglomeration starts to take place at relatively high pH and proceeds to a large

extent, possibly because the aggregates are less frequently disrupted than in stirred tanks. For the same reason, the shear (orthokinetic) kernel succeeded in reproducing the net result of the aggregation-disruption processes better than the size-independent.

- Bimodality in the CSD was shown, by means of both experiment and model, to be the result of a new wave of nucleation-growth induced by the existing crystals (secondary nucleation).

The conjunction of the penetration theory and the CFD predictions of the gas hold-up seem to yield adequate description of the reactor performance, as the prediction of the concentrations' evolution shows. Therefore the the discrepancies in the CSD predictions can be attributed mainly to the uncertainty in precipitation kinetics. Systematic studies of the kinetics, as well as improved mechanistic models of agglomeration (especially in relation to hydrodynamics) are required to obtain quantitative agreement, but the main trends in the CSD can be explained and reproduced with the aid of the model.

5.4 Suggestions for future work

The work undertaken here could be followed up in several directions. To begin with, the hybrid CFD-reaction engineering methodology presented in chapter 2 is general and its main concept concept lends itself to other potential reaction engineering applications, such as:

- Single-phase reactors with complex nonlinear reaction systems, whose direct coupling with CFD would pose serious computational problems.
- Stirred tank reactors with fast reactions, where micromixing models can be included in the compartments around the inlets.
- Catalytic and biochemical reactors, where adsorption models or cell population balances may be employed.

In our case studies, the configuration of the generalised compartmental model on the basis of the CFD results was carried out through an iterative procedure led

by heuristics. Future work could explore the automation of the procedure, which would contribute further time-saving to the design stage.

The finite-element numerical method for solution of the time-dependent population balance is also applicable to a variety of physical and chemical systems, apart from crystallisation and precipitation. This is owed to the volume-based formulation, which excludes system-specific elements such as shape factors from the model, and to its stability and conservation properties that enable it to tackle both nucleation-growth and agglomeration-breakage problems. Some other systems that can be described by population balance equations are mentioned are: polymers, aerosols, emulsions, bubble swarms and cell populations. Future work could explore the applicability of the method to other systems.

Finally, our case study of $CaCO_3$ precipitation has shown that the conjunction of fluid dynamics, reaction engineering and crystallisation models within the framework of a hybrid approach is feasible and can lead to fully-predictive scale-up tools. Such tools must be supplied with reliable constitutive equations and kinetic data, however, and such data are currently lacking, regarding crystallisation mechanisms and agglomeration in particular. Future work in this field could focus on the development of reliable kinetic models, both mechanistic and empirical, for the crystallisation mechanisms so that fully predictive quantitative scale-up can be made possible.

Appendix A

Solution of the interface PDEs

A straightforward point-iterative linearisation is bound to encounter stability problems due to the non-linear reaction terms, while full computation of the Jacobian would require considerable resources. This complexity will be avoided by the use of an iterative scheme where the equations for each chemical component and moment are being solved sequentially, while on the same time utilising gradient information. The source term is linearised as:

$$\begin{aligned}&\int_{t-1}^{t} (f_{i,p}\delta x)dt = \\ &= \int_{t-1}^{t} \left[f_{i,p}|_{t-1} + \sum_{j\neq i} \frac{\partial f_{i,p}}{\partial C_j}\bigg|_{t-1} (C_{j,t} - C_{j,t-1}) + \frac{\partial f_{i,p}}{\partial C_i}\bigg|_{t-1} (C_{i,t} - C_{i,t-1}) \right] dt\delta x\end{aligned} \tag{A.1}$$

An advantage of this scheme is that the term:

$$\frac{\partial f_{i,p}}{\partial C_{i,p}}\bigg|_{t-1} \int_{t-1}^{t} C_{i,p,t}\delta x dt$$

The linearised form is:

$$\begin{aligned}C_{i,p,t}\delta x - C_{i,p,t-1}\delta x = D\tfrac{\delta t}{\delta x}C_{i,e,t} - \tfrac{2D\delta t a}{\delta x}C_{i,p,t} - \tfrac{2D\delta t(1-a)}{\delta x}C_{i,p,t-1} + \\ +D\tfrac{\delta t}{\delta x}C_{i,w,t} + f_{i,p}|_{t-1}\,\delta x\delta t + \sum_{j\neq i} \frac{\partial f_{i,p}}{\partial C_{j,p}}\bigg|_{t-1} (C_{j,p,t} - C_{j,p,t-1})\delta x\delta t + \\ + \frac{\partial f_{i,p}}{\partial C_{i,p}}\bigg|_{t-1} \int_{t-1}^{t} C_{i,p,t}\delta x dt - \frac{\partial f_{i,p}}{\partial C_{i,p}}\bigg|_{t-1} C_{i,p,t-1}\delta x\delta t\end{aligned} \tag{A.2}$$

At this point all terms in the discretised equation can be assembled to form three coefficients (α_w, α_p, α_e) to the variables at the three neighbouring points ($C_{j,p,t}$ $C_{j,e,t}, C_{j,w,t}$) at the new time point, and a constant term (b). Thus a tridiagonal matrix is formed, whose solution yields the values of the variable at the new time point:

$$\begin{bmatrix} a_{1p} & a_{1e} & & & \\ . & . & . & & \\ & a_{kw} & a_{kp} & a_{ke} & \\ & & . & . & . \\ & & & a_{nw} & a_{np} \end{bmatrix} \begin{bmatrix} C_{i1} \\ . \\ C_{ik} \\ . \\ C_{in} \end{bmatrix} = \begin{bmatrix} b_1 \\ . \\ b_k \\ . \\ b_n \end{bmatrix} \tag{A.3}$$

The coefficients are yielded by the expressions:

$$\alpha_p = \delta x + \frac{2D\delta t a}{\delta x} - \left.\frac{\partial f_{i,p}}{\partial C_{i,p}}\right|_{t-1} a\delta x\delta t \tag{A.4}$$

$$\alpha_w = -D\frac{\delta t}{\delta x} \tag{A.5}$$

$$a_e = -D\frac{\delta t}{\delta x} \tag{A.6}$$

$$\begin{aligned} b = &+C_{i,p,t-1}\delta x - \tfrac{2D\delta t(1-a)}{\delta x}C_{i,p,t-1} + f_{i,p}|_{t-1}\,\delta x\delta t + \\ &+\sum_{j\neq i} \left.\tfrac{\partial f_i}{\partial C_{j,p}}\right|_{t-1} (C_{j,p,t} - C_{j,p,t-1})\delta x\delta t - a\left.\tfrac{\partial f_{i,p}}{\partial C_{i,p}}\right|_{t-1} C_{i,p,t-1}\delta x\delta t \end{aligned} \tag{A.7}$$

Appendix B

Source files and user subroutines for the CFX 4.3 code

B.1 CFX source file

The following is a sample CFX source file used for modelling a bubble column. For information on the format of CFX source files, one should consult the CFX manual (AEA Technology, 1999).

```
/*************************************************************/
/* BUBBLE COLUMN                                             */
/* BY STELIOS RIGOPOULOS                                     */
/* 24/06/2002                                                */
/*************************************************************/

/* General settings ******************************************/

>>CFXF3D
 >>SET LIMITS
    TOTAL REAL WORK SPACE 50000000
    TOTAL INTEGER WORK SPACE 10000000
  >>OPTIONS
    TWO DIMENSIONS
    RECTANGULAR GRID
    CYLINDRICAL COORDINATES
    AXIS INCLUDED
    AXISYMMETRIC MODIFICATION
    TURBULENT FLOW
    BUOYANT FLOW
    NUMBER OF PHASES 2
    TRANSIENT FLOW
    USER SCALAR EQUATIONS 1
  >>USER FORTRAN
    USRTRN
```

```
    USRINT
    USRSRC
  >>PHASE NAMES
    PHASE1 'WATER'
    PHASE2 'GAS'

/* Define the grid ******************************************/

>>MODEL TOPOLOGY

  #CALC
  Bottom_I       = 5;
  Main_I         = 195;
  Top_I          = 40;
  Riser_J        = 10;
  Dcomer_J       = 8;
  GasInlet_I     = 1;
  GasInlet1_J    = 1;
  GasInlet2_J    = 4;

  RiserTop_I     = Bottom_I+Main_I;
  Total_I        = Bottom_I+Main_I+Top_I;
  Total_J        = Riser_J+Dcomer_J;
  Total_K        = 1;

  BEFORE_DT_I = Bottom_I-1;

  AFTER_DT_J = Riser_J+1;

  BEFORE_DT_J = Riser_J-1;

  AFTER_DT_I = RiserTop_I+1;

  GAS_BOUNDARY = 6;
  AFTER_GAS_BOUNDARY = 6+1;

  #ENDCALC

  #ENDCALC

  >>CREATE BLOCK
    BLOCK NAME 'BUBBLE_COLUMN'
    BLOCK DIMENSIONS #Total_I #Total_J #Total_K
  >>CREATE PATCH
    PATCH NAME 'TOP'
    PATCH TYPE 'PRESSURE BOUNDARY'
    BLOCK NAME 'BUBBLE_COLUMN'
    HIGH I
  >>CREATE PATCH
    PATCH NAME 'DRAFT-TUBE'
    PATCH TYPE 'THIN SURFACE'
    BLOCK NAME 'BUBBLE_COLUMN'
    PATCH LOCATION #Bottom_I #RiserTop_I #Riser_J #Riser_J 1 1
    HIGH J
  >>CREATE PATCH
    PATCH NAME 'CENTRE-LINE'
    PATCH TYPE 'SYMMETRY PLANE'
    BLOCK NAME 'BUBBLE_COLUMN'
    LOW J
  >>CREATE PATCH
    PATCH NAME 'SPARGER'
    PATCH TYPE 'USER3D'
    BLOCK NAME 'BUBBLE_COLUMN'
    PATCH LOCATION #GasInlet_I #GasInlet_I #GasInlet1_J #GasInlet2_J 1 1
 >>CREATE PATCH
    PATCH NAME  'DOWNCOMER'
    PATCH TYPE  'USER3D'
    BLOCK NAME  'BUBBLE_COLUMN'
```

```
    PATCH LOCATION  1 #RiserTop_I #Riser_J #Total_J 1 1
  >>CREATE PATCH
    PATCH NAME  'RISER'
    PATCH TYPE  'USER3D'
    BLOCK NAME  'BUBBLE_COLUMN'
    PATCH LOCATION 1 #RiserTop_I 1 #Riser_J 1 1
  >>CREATE PATCH
    PATCH NAME  'TOTAL_COLUMN'
    PATCH TYPE  'USER3D'
    BLOCK NAME  'BUBBLE_COLUMN'
    PATCH LOCATION 1 #Total_I 1 #Total_J 1 1

/* Model data and properties ************************************/

>>MODEL DATA
  >>SET INITIAL GUESS
    >>INPUT FROM FILE
      READ DUMP FILE
  >>TITLE
    PROBLEM TITLE 'DRAFT-TUBE BUBBLE COLUMN'
  >>AMBIENT VARIABLES
    PHASE NAME 'WATER'
    VOLUME FRACTION 0.0
  >>AMBIENT VARIABLES
    PHASE NAME 'GAS'
    VOLUME FRACTION 1.0
    END
/*   >>DIFFERENCING SCHEME
    VOLUME FRACTION 'UPWIND' */
  >>PHYSICAL PROPERTIES
    #CALC
    DENL = 1000.0;
    VISL = 1.0E-3;
    DENG = 1.2;
    VISG = 1.8E-5;
    #ENDCALC

   >>BUOYANCY PARAMETERS
     GRAVITY VECTOR -9.81 0.0 0.0
     BUOYANCY REFERENCE DENSITY #DENL
   >>FLUID PARAMETERS
     PHASE NAME 'WATER'
     DENSITY #DENL
     VISCOSITY #VISL
   >>FLUID PARAMETERS
     PHASE NAME 'GAS'
     DENSITY #DENG
     VISCOSITY #VISG

/****************************************************************/

   >>TRANSIENT PARAMETERS
     >>FIXED TIME STEPPING
       TIME STEPS 500*0.01

/****************************************************************/

   >>SCALAR PARAMETERS
     >>DIFFUSIVITIES
/*       PHASE NAME 'WATER'*/
       USER SCALAR1 1.0E-4

/****************************************************************/

/* Turbulence and multiphase model ******************************/

   >>TURBULENCE PARAMETERS
      >>TURBULENCE MODEL
```

```
        PHASE NAME 'WATER'
        TURBULENCE MODEL 'K-EPSILON'
        /*PARTICLE INDUCED TURBULENCE 'SATO'*/
      >>TURBULENCE MODEL
        PHASE NAME 'GAS'
        TURBULENCE MODEL 'LAMINAR'
        /*PARTICLE INDUCED TURBULENCE 'SATO'*/

      >>TURBULENT PRANDTL NUMBER
        PHASE NAME 'WATER'
        USER SCALAR1 1.0

    >>MULTIPHASE PARAMETERS
      >>PHASE DESCRIPTION
        PHASE NAME 'WATER'
        LIQUID
        CONTINUOUS
      >>PHASE DESCRIPTION
        PHASE NAME 'GAS'
        GAS
        DISPERSE
        MEAN DIAMETER 5.0E-3
        MINIMUM VOLUME FRACTION 1.E-8

      >>MULTIPHASE MODELS
        >>MOMENTUM
          INTER PHASE TRANSFER
        /*>>TURBULENCE
         HOMOGENEOUS*/

      >>INTER PHASE TRANSFER MODELS
        >>MOMENTUM
          FIRST PHASE NAME 'WATER'
          SECOND PHASE NAME 'GAS'
        >>PARTICLE DRAG MODEL
          FLOW REGIME 'NEWTON'
          NEWTON COEFFICIENT 0.66
        >>NON DRAG FORCES
          TURBULENT DISPERSION FORCE

/*********************************************************/

/* Input of gas ****************************************/

     >>SOURCES
        #CALC
        FMASS = 0.0002;
        #ENDCALC
        PATCH NAME 'SPARGER'
        PHASE NAME 'GAS'
        TOTAL FOR PHASE
        PRESSURE #FMASS 0.0
        VOLUME FRACTION #FMASS 0.0

/*********************************************************/

/*********************************************************/


/* Solver options **************************************/

    >>SOLVER DATA
      >>PROGRAM CONTROL
        #CALC
        IMON = 40;
        JMON = 5;
        KMON = 1;
        #ENDCALC
```

```
        OUTPUT MONITOR BLOCK 'BUBBLE_COLUMN'
        OUTPUT MONITOR POINT #IMON #JMON #KMON
        MINIMUM NUMBER OF ITERATIONS 3
        MAXIMUM NUMBER OF ITERATIONS 40
        MASS SOURCE TOLERANCE 1.0E-7

   /*   >>FALSE TIMESTEPS
        U VELOCITY 0.01
        V VELOCITY 0.01 */
      >>UNDER RELAXATION FACTORS
        ALL EQUATIONS 0.1

/*********************************************************/


/* Actual geometry and dimensions *************************/

      >>CREATE GRID

        #CALC

        Bottom_length = 0.05;
        Main_length = 1.95;
        Top_length = 0.4;
        Riser_width = 0.06;
        Dcomer_width = 0.05;

        Idx = (Bottom_length+Main_length+Top_length)/Total_I;
        Jdx = (Riser_width+Dcomer_width)/Total_J;
        Kdx = 2*3.14159;

        #ENDCALC

      >>SIMPLE GRID
        BLOCK NAME 'BUBBLE_COLUMN'
        DX #Total_I*#Idx
        DY #Total_J*#Jdx
        DZ #Total_K*#Kdx

/*********************************************************/


/* Boundary conditions ************************************/

>>MODEL BOUNDARY CONDITIONS
  >>PRESSURE BOUNDARIES
    PATCH NAME 'TOP'
    PRESSURE 0.0

/*********************************************************/


/* Output files *******************************************/

>>OUTPUT OPTIONS
  >>PRINT OPTIONS
  >>WHAT
   ALL VARIABLES
  >>ANIMATION DATA
    FILE NAME 'ANIM'
    Z 3.14
    TIME INTERVAL 0.01
    PHASE NAME 'WATER'
    USER SCALAR1
/*********************************************************/

>>STOP
```

B.2 Fortran user subroutines

CFX allows the use of Fortran user subroutines to customize certain elements of the solution. The following ones were used here:

USRTRN: Prints user-specified information.

USRINT: Specifies initial conditions.

USRSRC: Used to to introduce the passive scalar for the virtual tracer experiment.

```
      SUBROUTINE USRTRN(U,V,W,P,VFRAC,DEN,VIS,TE,ED,RS,T,H,RF,SCAL,
     +                  XP,YP,ZP,VOL,AREA,VPOR,ARPOR,WFACT,CONV,IPT,
     +                  IBLK,IPVERT,IPNODN,IPFACN,IPNODF,IPNODB,IPFACB,
     +                  WORK,IWORK,CWORK)
C
C*********************************************************************
C
C   USER SUBROUTINE TO ALLOW USERS TO MODIFY OR MONITOR THE SOLUTION AT
C   THE END OF EACH TIME STEP
C   THIS SUBROUTINE IS CALLED BEFORE THE START OF THE RUN AS WELL AS AT
C   THE END OF EACH TIME STEP
C
C   >>> IMPORTANT                                                  <<<
C   >>>                                                            <<<
C   >>> USERS MAY ONLY ADD OR ALTER PARTS OF THE SUBROUTINE WITHIN  <<<
C   >>> THE DESIGNATED USER AREAS                                  <<<
C
C*********************************************************************
C
C   THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINES
C      CUSR  TRNMOD
C
C*********************************************************************
C   CREATED
C      27/04/90  ADB
C   MODIFIED
C      05/08/91  IRH  NEW STRUCTURE
C      01/10/91  DSC  REDUCE COMMENT LINE GOING OVER COLUMN 72.
C      29/11/91  PHA  UPDATE CALLED BY COMMENT, ADD RF ARGUMENT,
C                     CHANGE LAST DIMENSION OF RS TO 6 AND IVERS TO 2
C      05/06/92  PHA  ADD PRECISION FLAG AND CHANGE IVERS TO 3
C      03/07/92  DSC  CORRECT COMMON MLTGRD.
C      23/11/93  CSH  EXPLICITLY DIMENSION IPVERT ETC.
C      03/02/94  PHA  CHANGE FLOW3D TO CFDS-FLOW3D
C      22/08/94  NSW  MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA
C      19/12/94  NSW  CHANGE FOR CFX-F3D
C      02/07/97  NSW  UPDATE FOR CFX-4
C      02/07/99  NSW  INCLUDE NEW EXAMPLE FOR CALCULATING FLUX OF A
C                     SCALAR AT A PRESSURE BOUNDARY
C
C*********************************************************************
C
C   SUBROUTINE ARGUMENTS
C
C     U      - U COMPONENT OF VELOCITY
C     V      - V COMPONENT OF VELOCITY
C     W      - W COMPONENT OF VELOCITY
C     P      - PRESSURE
C     VFRAC  - VOLUME FRACTION
```

```
C     DEN    - DENSITY OF FLUID
C     VIS    - VISCOSITY OF FLUID
C     TE     - TURBULENT KINETIC ENERGY
C     ED     - EPSILON
C     RS     - REYNOLD STRESSES
C     T      - TEMPERATURE
C     H      - ENTHALPY
C     RF     - REYNOLD FLUXES
C     SCAL   - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)
C     XP     - X COORDINATES OF CELL CENTRES
C     YP     - Y COORDINATES OF CELL CENTRES
C     ZP     - Z COORDINATES OF CELL CENTRES
C     VOL    - VOLUME OF CELLS
C     AREA   - AREA OF CELLS
C     VPOR   - POROUS VOLUME
C     ARPOR  - POROUS AREA
C     WFACT  - WEIGHT FACTORS
C     CONV   - CONVECTION COEFFICIENTS
C
C     IPT    - 1D POINTER ARRAY
C     IBLK   - BLOCK SIZE INFORMATION
C     IPVERT - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES
C     IPNODN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C     IPFACN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C     IPNODF - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C     IPNODB - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C     IPFACB - POINTER FROM BOUNDARY CENTERS TO BOUNDARY FACESS
C
C     WORK   - REAL WORKSPACE ARRAY
C     IWORK  - INTEGER WORKSPACE ARRAY
C     CWORK  - CHARACTER WORKSPACE ARRAY
C
C   SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C   BE SET  BY THE USER IN THIS ROUTINE.
C
C   NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-4 USING THE
C   ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4
C   USER MANUAL.
C
C*********************************************************************
C
C
      LOGICAL LDEN,LVIS,LTURB,LTEMP,LBUOY,LSCAL,LCOMP
     +       ,LRECT,LCYN,LAXIS,LPOROS,LTRANS
C
      CHARACTER*(*) CWORK
C
C+++++++++++++++++ USER AREA 1 ++++++++++++++++++++++++++++++++++++++++
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
C      CHARACTER*20 FNAME
C
C+++++++++++++++++ END OF USER AREA 1 +++++++++++++++++++++++++++++++++
C
      COMMON
     + /ALL/    NBLOCK,NCELL,NBDRY,NNODE,NFACE,NVERT,NDIM
     + /ALLWRK/ NRWS,NIWS,NCWS,IWRFRE,IWIFRE,IWCFRE
     + /ADDIMS/ NPHASE,NSCAL,NVAR,NPROP
     +         ,NDVAR,NDPROP,NDXNN,NDGEOM,NDCOEF,NILIST,NRLIST,NTOPOL
     + /CHKUSR/ IVERS,IUCALL,IUSED
     + /CONC/   NCONC
     + /DEVICE/ NREAD,NWRITE,NRDISK,NWDISK
     + /IDUM/   ILEN,JLEN
     + /LOGIC/  LDEN,LVIS,LTURB,LTEMP,LBUOY,LSCAL,LCOMP
     +         ,LRECT,LCYN,LAXIS,LPOROS,LTRANS
     + /MLTGRD/ MLEVEL,NLEVEL,ILEVEL
     + /SGLDBL/ IFLGPR,ICHKPR
     + /SPARM/  SMALL,SORMAX,NITER,INDPRI,MAXIT,NODREF,NODMON
     + /TIMUSR/ DTUSR
```

```
     + /TRANSI/ NSTEP,KSTEP,MF,INCORE
     + /TRANSR/ TIME,DT,DTINVF,TPARM
C
C+++++++++++++++++ USER AREA 2 ++++++++++++++++++++++++++++++++++++++++++++
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C     THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE
C     NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C+++++++++++++++++ END OF USER AREA 2 ++++++++++++++++++++++++++++++++++++
C
      DIMENSION
     + U(NNODE,NPHASE),V(NNODE,NPHASE),W(NNODE,NPHASE),P(NNODE,NPHASE)
     +,VFRAC(NNODE,NPHASE),DEN(NNODE,NPHASE),VIS(NNODE,NPHASE)
     +,TE(NNODE,NPHASE),ED(NNODE,NPHASE),RS(NNODE,NPHASE,6)
     +,T(NNODE,NPHASE),H(NNODE,NPHASE),RF(NNODE,NPHASE,4)
     +,SCAL(NNODE,NPHASE,NSCAL)
      DIMENSION
     + XP(NNODE),YP(NNODE),ZP(NNODE)
     +,VOL(NCELL),AREA(NFACE,3),VPOR(NCELL),ARPOR(NFACE,3)
     +,WFACT(NFACE),CONV(NFACE,NPHASE)
     +,IPT(*),IBLK(5,NBLOCK)
     +,IPVERT(NCELL,8),IPNODN(NCELL,6),IPFACN(NCELL,6),IPNODF(NFACE,4)
     +,IPNODB(NBDRY,4),IPFACB(NBDRY)
     +,IWORK(*),WORK(*),CWORK(*)
      DIMENSION SGNWL(6)
C
C+++++++++++++++++ USER AREA 3 ++++++++++++++++++++++++++++++++++++++++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
C
C
C+++++++++++++++++ END OF USER AREA 3 ++++++++++++++++++++++++++++++++++++
C
      DATA SGNWL / 1.0, 1.0, 1.0, -1.0, -1.0, -1.0 /
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K)=IPT((K-1)*ILEN*JLEN+(J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
C
      IVERS=3
      ICHKPR = 1
C
C+++++++++++++++++ USER AREA 4 ++++++++++++++++++++++++++++++++++++++++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUSED=1
C
       IUSED=1
C
C+++++++++++++++++ END OF USER AREA 4 ++++++++++++++++++++++++++++++++++++
C
       IF (IUSED.EQ.0) RETURN
C
C---- FRONTEND CHECKING OF USER ROUTINE
       IF (IUCALL.EQ.0) RETURN
C
C+++++++++++++++++ USER AREA 5 ++++++++++++++++++++++++++++++++++++++++++++
C
C     END OPERATION
C

C ---------------------------------------------------------------------

C           CALCULATIONS IN THE BEGINNING

C ---------------------------------------------------------------------
```

```
      IF(KSTEP.EQ.1) THEN

      CALL IPREC('BUBBLE_COLUMN','BLOCK','CENTRES',IPT,ILEN,JLEN,KLEN,
     +           CWORK,IWORK)
      FULLCEL = 0
      DO 101 I=1,ILEN
      INODE = IP(I,4,1)
      IF (VFRAC(INODE,2).LT.0.5) THEN
        FULLCEL = FULLCEL + 1
      ENDIF
  101 CONTINUE
      NODE2=IP(2,1,1)
      NODE3=IP(3,1,1)
      LIQLEVELI = FULLCEL * (XP(NODE3)-XP(NODE2))
      WRITE(81,*) LIQLEVELI

      ENDIF

C ------------------------------------------------------------------

C           CALCULATIONS IN THE END

C ------------------------------------------------------------------


      IF(KSTEP.EQ.NSTEP) THEN

C*****************************************************************
C
C-----CALCULATE GAS HOLDUP AND LIQUID VELOCITY IN DOWNCOMER
C
      CALL IPALL('DOWNCOMER','*','PATCH','CENTRES',IPT,NPT,
     +           CWORK,IWORK)
C
      SUMVOL = 0.0
      SUMVG  = 0.0
      SUMLIQVELD = 0.0
      DO 201 I=1,NPT
        INODE=IPT(I)
        SUMVOL = SUMVOL+ VOL(INODE)
        SUMVG  = SUMVG + VFRAC(INODE,2)*VOL(INODE)
        SUMLIQVELD = SUMLIQVELD + U(INODE,1)
  201 CONTINUE
      RGDOWN = SUMVG/SUMVOL*100.0
      AVELIQVELD = SUMLIQVELD/NPT
C
C*****************************************************************
C
C-----CALCULATE GAS HOLDUP AND AVERAGE VELOCITIES IN RISER
C
      CALL IPALL('RISER','*','PATCH','CENTRES',IPT,NPT,
     +           CWORK,IWORK)
C
      SUMVOL = 0.0
      SUMVG  = 0.0
      SUMGASVEL = 0.0
      SUMLIQUIDVEL = 0.0
      DO 301 I=1,NPT
        INODE=IPT(I)
        SUMVOL = SUMVOL+VOL(INODE)
        SUMVG  = SUMVG +VFRAC(INODE,2)*VOL(INODE)
        SUMGASVEL = SUMGASVEL + U(INODE,2)
        SUMLIQUIDVEL = SUMLIQUIDVEL + U(INODE,1)
  301 CONTINUE

      RGRISE = SUMVG/SUMVOL*100.0
      AVEGASVEL = SUMGASVEL/NPT
```

```
      AVELIQUIDVEL = SUMLIQUIDVEL/NPT
      SLIPVEL = AVEGASVEL - AVELIQUIDVEL
C
C*****************************************************************
C
C*****************************************************************
C
C-----CALCULATE GAS HOLDUP IN COLUMN
C
      CALL IPALL('TOTAL_COLUMN','*','PATCH','CENTRES',IPT,NPT,
     +           CWORK,IWORK)
C
      SUMVOL = 0.0
      SUMVG  = 0.0
      SUMGASVEL = 0.0
      SUMLIQUIDVEL = 0.0
      DO 401 I=1,NPT
        INODE=IPT(I)
        SUMVOL = SUMVOL+VOL(INODE)
        SUMVG  = SUMVG +VFRAC(INODE,2)*VOL(INODE)
  401 CONTINUE

      RGTOTAL = SUMVG/SUMVOL*100.0
C
C*****************************************************************
C
C*****************************************************************
C
C-----CALCULATE AVERAGE TURBULENT ENERGY DISSIPATION
C
      CALL IPALL('TOTAL_COLUMN','*','PATCH','CENTRES',IPT,NPT,
     +           CWORK,IWORK)
C
      SUMED = 0.0
      DO 501 I=1,NPT
        INODE=IPT(I)
        SUMED = SUMED+ED(INODE,1)
  501 CONTINUE

      AVEED = SUMED/NPT
C



C*****************************************************************
C
C
C*****************************************************************
C
C-----CALCULATE POSITION OF FREE SURFACE
C
      CALL IPREC('BUBBLE_COLUMN','BLOCK','CENTRES',IPT,ILEN,JLEN,KLEN,
     +           CWORK,IWORK)
      FULLCEL = 0
      DO 700 I=1,ILEN
      INODE = IP(I,4,1)
      IF (VFRAC(INODE,2).LT.0.5) THEN
        FULLCEL = FULLCEL + 1
      ENDIF
  700 CONTINUE
      NODE1=IP(1,1,1)
      NODE2=IP(2,1,1)
      RLIQLEVEL = FULLCEL * (XP(NODE2)-XP(NODE1))

      DO 730 I=1,ILEN
        DO 720 J=1,JLEN
          DO 710 K=1,KLEN
            INODE = IP(I,J,K)
```

```
            TOTALLIQ = TOTALLIQ + VOL(INODE)*VFRAC(INODE,1)
            IF (VFRAC(INODE,2).LT.0.5) THEN
              REALLIQ = REALLIQ + VOL(INODE)*VFRAC(INODE,1)
            ENDIF
  710     CONTINUE
  720   CONTINUE
  730 CONTINUE


      WRITE(NWRITE,*) P(IP(1,10,1),1)
C
C*****************************************************************
C
C-----PREPARE OUTPUT FILE
C
      OPEN (81, FILE='../RESULTS')

      WRITE(81,*) ' ******* GAS HOLDUP *******'
      WRITE(81,*) ' '
      WRITE(81,*) ' IN DOWNCOMER (%) = ', RGDOWN
      WRITE(81,*) ' IN RISER     (%) = ', RGRISE
      WRITE(81,*) ' '
      WRITE(81,*) ' TOTAL        (%) = ', RGTOTAL
      WRITE(81,*) ' '
      WRITE(81,*) ' TURBULENT ENERGY DISSIPATION = ', AVEED
      WRITE(81,*) ' '
      WRITE(81,*) ' ******* AVERAGE VELOCITIES *******'
      WRITE(81,*) ' '
      WRITE(81,*) ' GAS PHASE VELOCITY ', AVEGASVEL
      WRITE(81,*) ' '
      WRITE(81,*) ' AVERAGE LIQUID VELOCITY IN RISER ', AVELIQUIDVEL
      WRITE(81,*) ' '
      WRITE(81,*) ' AVERAGE LIQUID VELOCITY IN DOWNCOMER ', AVELIQVELD
      WRITE(81,*) ' '
      WRITE(81,*) ' SLIP VELOCITY ', SLIPVEL
      WRITE(81,*) ' '
      WRITE(81,*) ' LIQUID LEVEL ', RLIQLEVEL
      WRITE(81,*) ' '
      WRITE(81,*) ' TOTAL LIQUID ', TOTALLIQ
      WRITE(81,*) ' '
      WRITE(81,*) ' LIQUID UNDER FREE SURFACE ', REALLIQ

C
      ENDIF
C+++++++++++++++++ END OF USER AREA 5 ++++++++++++++++++++++++++++++++++
C
      RETURN
      END
C
C
C
C-----------------------------------------------------------------------
C
C
      SUBROUTINE USRINT(U,V,W,P,VFRAC,DEN,VIS,TE,ED,RS,T,H,RF,SCAL
     +                  ,CONV,XC,YC,ZC,XP,YP,ZP
     +                  ,VOL,AREA,VPOR,ARPOR,WFACT,DISWAL,IPT
     +                  ,IBLK,IPVERT,IPNODN,IPFACN,IPNODF,IPNODB,IPFACB
     +                  ,WORK,IWORK,CWORK)
C
C*********************************************************************
C
C   UTILITY SUBROUTINE FOR USER-SUPPLIED INITIAL FIELD.
C
C*********************************************************************
C
C   THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINE
C      CUSR  INIT
```

```
C
C*********************************************************************
C   CREATED
C      13/06/90 ADB
C   MODIFIED
C      07/08/91 IRH   NEW STRUCTURE
C      10/09/91 IRH   CORRECTION TO IUSED
C      26/09/91 IRH   ALTER ARGUMENT LIST
C      01/10/91 DSC   REDUCE COMMENT LINE GOING OVER COLUMN 72.
C      03/10/91 IRH   CORRECT COMMENTS
C      28/01/92 PHA   UPDATE CALLED BY COMMENT, ADD RF ARGUMENT,
C                     CHANGE LAST DIMENSION OF RS TO 6 AND IVERS TO 2
C      03/06/92 PHA   ADD PRECISION FLAG AND CHANGE IVERS TO 3
C      08/02/93 NSW   REMOVE REDUNDANT COMMENTS
C      23/11/93 CSH   EXPLICITLY DIMENSION IPVERT ETC.
C      03/02/94 PHA   CHANGE FLOW3D TO CFDS-FLOW3D, REMOVE COMMA
C                     FROM BEGINNING OF DIMENSION STATEMENT
C      03/03/94 FHW   CORRECTION OF SPELLING MISTAKE
C      09/08/94 NSW   CORRECT SPELLING
C                     MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA
C      19/12/94 NSW   CHANGE FOR CFX-F3D
C      30/01/95 NSW   INCLUDE NEW EXAMPLE
C
C*********************************************************************
C
C   SUBROUTINE ARGUMENTS
C
C     U      - U COMPONENT OF VELOCITY
C     V      - V COMPONENT OF VELOCITY
C     W      - W COMPONENT OF VELOCITY
C     P      - PRESSURE
C     VFRAC  - VOLUME FRACTION
C     DEN    - DENSITY OF FLUID
C     VIS    - VISCOSITY OF FLUID
C     TE     - TURBULENT KINETIC ENERGY
C     ED     - EPSILON
C     RS     - REYNOLD STRESSES
C     T      - TEMPERATURE
C     H      - ENTHALPY
C     RF     - REYNOLD FLUXES
C     SCAL   - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)
C     CONV   - CONVECTION COEFFICIENTS
C     XC     - X COORDINATES OF CELL CORNERS
C     YC     - Y COORDINATES OF CELL CORNERS
C     ZC     - Z COORDINATES OF CELL CORNERS
C     XP     - X COORDINATES OF CELL CENTRES
C     YP     - Y COORDINATES OF CELL CENTRES
C     ZP     - Z COORDINATES OF CELL CENTRES
C     VOL    - VOLUME OF CELLS
C     AREA   - AREA OF CELLS
C     VPOR   - POROUS VOLUME
C     ARPOR  - POROUS AREA
C     WFACT  - WEIGHT FACTORS
C     DISWAL - DISTANCE OF CELL CENTRE FROM WALL
C
C     IPT    - 1D POINTER ARRAY
C     IBLK   - BLOCK SIZE INFORMATION
C     IPVERT - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES
C     IPNODN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C     IPFACN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C     IPNODF - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C     IPNODB - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C     IPFACB - POINTER FROM BOUNDARY CENTERS TO BOUNDARY FACES
C
C     WORK   - REAL WORKSPACE ARRAY
C     IWORK  - INTEGER WORKSPACE ARRAY
C     CWORK  - CHARACTER WORKSPACE ARRAY
C
```

```
C   SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C   BE SET  BY THE USER IN THIS ROUTINE.
C
C   LOGICAL VARIABLE LRDISK IN COMMON BLOCK IOLOGC INDICATES WHETHER
C   THE RUN IS A RESTART AND CAN BE USED SO THAT INITIAL INFORMATION
C   IS ONLY SET WHEN STARTING A RUN FROM SCRATCH.
C
C   NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-F3D USING THE
C   ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4
C   USER MANUAL.
C
C*********************************************************************
      LOGICAL LDEN,LVIS,LTURB,LTEMP,LBUOY,LSCAL,LCOMP
     +       ,LRECT,LCYN,LAXIS,LPOROS,LTRANS
      LOGICAL LRDISK,LWDISK
C
      CHARACTER*(*) CWORK
C
C+++++++++++++++++ USER AREA 1 ++++++++++++++++++++++++++++++++++++++++
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
C
C+++++++++++++++++ END OF USER AREA 1 ++++++++++++++++++++++++++++++++
C
      COMMON
     + /ALL/    NBLOCK,NCELL,NBDRY,NNODE,NFACE,NVERT,NDIM
     + /ALLWRK/ NRWS,NIWS,NCWS,IWRFRE,IWIFRE,IWCFRE
     + /ADDIMS/ NPHASE,NSCAL,NVAR,NPROP
     +         ,NDVAR,NDPROP,NDXNN,NDGEOM,NDCOEF,NILIST,NRLIST,NTOPOL
     + /CHKUSR/ IVERS,IUCALL,IUSED
     + /DEVICE/ NREAD,NWRITE,NRDISK,NWDISK
     + /IDUM/   ILEN,JLEN
     + /IOLOGC/ LRDISK,LWDISK
     + /LOGIC/  LDEN,LVIS,LTURB,LTEMP,LBUOY,LSCAL,LCOMP
     +         ,LRECT,LCYN,LAXIS,LPOROS,LTRANS
     + /MLTGRD/ MLEVEL,NLEVEL,ILEVEL
     + /SGLDBL/ IFLGPR,ICHKPR
     + /TRANSI/ NSTEP,KSTEP,MF,INCORE
     + /TRANSR/ TIME,DT,DTINVF,TPARM
C
C+++++++++++++++++ USER AREA 2 ++++++++++++++++++++++++++++++++++++++++
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C     THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE
C     NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C+++++++++++++++++ END OF USER AREA 2 ++++++++++++++++++++++++++++++++
C
      DIMENSION
     + U(NNODE,NPHASE),V(NNODE,NPHASE),W(NNODE,NPHASE)
     +,P(NNODE,NPHASE),VFRAC(NNODE,NPHASE)
     +,TE(NNODE,NPHASE),ED(NNODE,NPHASE),RS(NNODE,NPHASE,6)
     +,T(NNODE,NPHASE),H(NNODE,NPHASE),RF(NNODE,NPHASE,4)
     +,SCAL(NNODE,NPHASE,NSCAL)
     +,DEN(NNODE,NPHASE),VIS(NNODE,NPHASE),CONV(NFACE,NPHASE)
      DIMENSION
     + XC(NVERT),YC(NVERT),ZC(NVERT),XP(NNODE),YP(NNODE),ZP(NNODE)
     +,VOL(NCELL),AREA(NFACE,3),VPOR(NCELL),ARPOR(NFACE,3)
     +,WFACT(NFACE),DISWAL(NCELL)
      DIMENSION
     + IPT(*),IBLK(5,NBLOCK)
     +,IPVERT(NCELL,8),IPNODN(NCELL,6),IPFACN(NCELL,6),IPNODF(NFACE,4)
     +,IPNODB(NBDRY,4),IPFACB(NBDRY)
      DIMENSION
     + IWORK(NIWS),WORK(NRWS),CWORK(NCWS)
C
C+++++++++++++++ USER AREA 3 +++++++++++++++++++++++++++++++++++++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
```

```
C
C+++++++++++++++++ END OF USER AREA 3 +++++++++++++++++++++++++++++++++++
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K)=IPT((K-1)*ILEN*JLEN+(J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
C
      IVERS=3
      ICHKPR = 1
C
C+++++++++++++++++ USER AREA 4 ++++++++++++++++++++++++++++++++++++++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUSED=1
C
       IUSED=1
C
C+++++++++++++++++ END OF USER AREA 4 +++++++++++++++++++++++++++++++++++
C
       IF (IUSED.EQ.0) RETURN
C
C---- FRONTEND CHECKING OF USER ROUTINE
      IF (IUCALL.EQ.0) RETURN
C
C+++++++++++++++++ USER AREA 5 ++++++++++++++++++++++++++++++++++++++++++

      FULL=0.999999
      EMPTY=1 - FULL

C+++++++++++++++++ END OF USER AREA 5 +++++++++++++++++++++++++++++++++++
C
      RETURN
      END


      RETURN
      END
      SUBROUTINE USRSRC(IEQN,ICALL,CNAME,CALIAS,AM,SP,SU,CONV
     +                  ,U,V,W,P,VFRAC,DEN,VIS,TE,ED,RS,T,H,RF,SCAL
     +                  ,XP,YP,ZP,VOL,AREA,VPOR,ARPOR,WFACT,IPT
     +                  ,IBLK,IPVERT,IPNODN,IPFACN,IPNODF,IPNODB,IPFACB
     +                  ,WORK,IWORK,CWORK)
C
C***********************************************************************
C
C   UTILITY SUBROUTINE FOR USER-SUPPLIED SOURCES
C
C   >>> IMPORTANT                                                   <<<
C   >>>                                                             <<<
C   >>> USERS MAY ONLY ADD OR ALTER PARTS OF THE SUBROUTINE WITHIN  <<<
C   >>> THE DESIGNATED USER AREAS                                   <<<
C
C***********************************************************************
C
C   THIS SUBROUTINE IS CALLED BY THE FOLLOWING SUBROUTINES
C      CUSR  SCDF  SCDS  SCED  SCENRG  SCHF  SCMOM  SCPCE  SCSCAL
C      SCTE  SCVF
C
C***********************************************************************
C   CREATED
C      08/03/90  ADB
C   MODIFIED
C      04/03/91  ADB  ALTERED ARGUMENT LIST.
C      28/08/91  IRH  NEW STRUCTURE
C      28/09/91  IRH  CHANGE EXAMPLE + ADD COMMON BLOCKS
C      10/02/92  PHA  UPDATE CALLED BY COMMENT, ADD RF ARGUMENT,
C                     CHANGE LAST DIMENSION OF RS TO 6 AND IVERS TO 2
C      03/06/92  PHA  ADD PRECISION FLAG AND CHANGE IVERS TO 3
C      23/11/93  CSH  EXPLICITLY DIMENSION IPVERT ETC.
```

```
C      07/12/93  NSW  INCLUDE CONV IN ARGUMENT LIST AND CHANGE IVERS
C                     TO 4
C      03/02/94  PHA  CHANGE FLOW3D TO CFDS-FLOW3D
C      03/03/94  FHW  CORRECTION OF SPELLING MISTAKE
C      08/03/94  NSW  CORRECT SPELLING
C      09/08/94  NSW  CORRECT SPELLING.
C                     MOVE 'IF(IUSED.EQ.0) RETURN' OUT OF USER AREA.
C                     INCLUDE COMMENT ON MASS SOURCES.
C      19/12/94  NSW  CHANGE FOR CFX-F3D
C      02/07/97  NSW  UPDATE FOR CFX-4
C
C***********************************************************************
C
C   SUBROUTINE ARGUMENTS
C
C     IEQN   - EQUATION NUMBER
C     ICALL  - SUBROUTINE CALL
C     CNAME  - EQUATION NAME
C     CALIAS - ALIAS OF EQUATION NAME
C     AM     - OFF DIAGONAL MATRIX COEFFICIENTS
C     SU     - SU IN LINEARISATION OF SOURCE TERM
C     SP     - SP IN LINEARISATION OF SOURCE TERM
C     CONV   - CONVECTION COEFFICIENTS
C     U      - U COMPONENT OF VELOCITY
C     V      - V COMPONENT OF VELOCITY
C     W      - W COMPONENT OF VELOCITY
C     P      - PRESSURE
C     VFRAC  - VOLUME FRACTION
C     DEN    - DENSITY OF FLUID
C     VIS    - VISCOSITY OF FLUID
C     TE     - TURBULENT KINETIC ENERGY
C     ED     - EPSILON
C     RS     - REYNOLD STRESSES
C     T      - TEMPERATURE
C     H      - ENTHALPY
C     RF     - REYNOLD FLUXES
C     SCAL   - SCALARS (THE FIRST 'NCONC' OF THESE ARE MASS FRACTIONS)
C     XP     - X COORDINATES OF CELL CENTRES
C     YP     - Y COORDINATES OF CELL CENTRES
C     ZP     - Z COORDINATES OF CELL CENTRES
C     VOL    - VOLUME OF CELLS
C     AREA   - AREA OF CELLS
C     VPOR   - POROUS VOLUME
C     ARPOR  - POROUS AREA
C     WFACT  - WEIGHT FACTORS
C
C     IPT    - 1D POINTER ARRAY
C     IBLK   - BLOCK SIZE INFORMATION
C     IPVERT - POINTER FROM CELL CENTERS TO 8 NEIGHBOURING VERTICES
C     IPNODN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING CELLS
C     IPFACN - POINTER FROM CELL CENTERS TO 6 NEIGHBOURING FACES
C     IPNODF - POINTER FROM CELL FACES TO 2 NEIGHBOURING CELL CENTERS
C     IPNODB - POINTER FROM BOUNDARY CENTERS TO CELL CENTERS
C     IPFACB - POINTER FROM BOUNDARY CENTERS TO BOUNDARY FACESS
C
C     WORK   - REAL WORKSPACE ARRAY
C     IWORK  - INTEGER WORKSPACE ARRAY
C     CWORK  - CHARACTER WORKSPACE ARRAY
C
C   SUBROUTINE ARGUMENTS PRECEDED WITH A '*' ARE ARGUMENTS THAT MUST
C   BE SET  BY THE USER IN THIS ROUTINE.
C
C   NOTE THAT WHEN USING MASS SOURCES, THE FLOWS THROUGH MASS FLOW
C   BOUNDARIES ARE UNCHANGED. THE USER SHOULD THEREFORE INCLUDE AT
C   LEAST ONE PRESSURE BOUNDARY FOR SUCH A CALCULATION.
C
C   NOTE THAT OTHER DATA MAY BE OBTAINED FROM CFX-4 USING THE
C   ROUTINE GETADD, FOR FURTHER DETAILS SEE THE VERSION 4
```

```
C   USER MANUAL.
C
C*********************************************************************
C
      LOGICAL LDEN,LVIS,LTURB,LTEMP,LBUOY,LSCAL,LCOMP
     +       ,LRECT,LCYN,LAXIS,LPOROS,LTRANS
C
      CHARACTER*(*) CWORK
      CHARACTER     CNAME*6, CALIAS*24
C
C+++++++++++++++++ USER AREA 1 ++++++++++++++++++++++++++++++++++++++++
C---- AREA FOR USERS EXPLICITLY DECLARED VARIABLES
C
C+++++++++++++++++ END OF USER AREA 1 ++++++++++++++++++++++++++++++++
C
      COMMON
     + /ALL/    NBLOCK,NCELL,NBDRY,NNODE,NFACE,NVERT,NDIM
     + /ALLWRK/ NRWS,NIWS,NCWS,IWRFRE,IWIFRE,IWCFRE
     + /ADDIMS/ NPHASE,NSCAL,NVAR,NPROP
     +         ,NDVAR,NDPROP,NDXNN,NDGEOM,NDCOEF,NILIST,NRLIST,NTOPOL
     + /CHKUSR/ IVERS,IUCALL,IUSED
     + /DEVICE/ NREAD,NWRITE,NRDISK,NWDISK
     + /IDUM/   ILEN,JLEN
     + /LOGIC/  LDEN,LVIS,LTURB,LTEMP,LBUOY,LSCAL,LCOMP
     +         ,LRECT,LCYN,LAXIS,LPOROS,LTRANS
     + /MLTGRD/ MLEVEL,NLEVEL,ILEVEL
     + /SGLDBL/ IFLGPR,ICHKPR
     + /SPARM/  SMALL,SORMAX,NITER,INDPRI,MAXIT,NODREF,NODMON
     + /TRANSI/ NSTEP,KSTEP,MF,INCORE
     + /TRANSR/ TIME,DT,DTINVF,TPARM
C
C+++++++++++++++++ USER AREA 2 ++++++++++++++++++++++++++++++++++++++++
C---- AREA FOR USERS TO DECLARE THEIR OWN COMMON BLOCKS
C     THESE SHOULD START WITH THE CHARACTERS 'UC' TO ENSURE
C     NO CONFLICT WITH NON-USER COMMON BLOCKS
C
C+++++++++++++++++ END OF USER AREA 2 ++++++++++++++++++++++++++++++++
C
      DIMENSION AM(NCELL,6,NPHASE),SP(NCELL,NPHASE),SU(NCELL,NPHASE)
     +,CONV(NFACE,NPHASE)
C
      DIMENSION
     + U(NNODE,NPHASE),V(NNODE,NPHASE),W(NNODE,NPHASE),P(NNODE,NPHASE)
     +,VFRAC(NNODE,NPHASE),DEN(NNODE,NPHASE),VIS(NNODE,NPHASE)
     +,TE(NNODE,NPHASE),ED(NNODE,NPHASE),RS(NNODE,NPHASE,6)
     +,T(NNODE,NPHASE),H(NNODE,NPHASE),RF(NNODE,NPHASE,4)
     +,SCAL(NNODE,NPHASE,NSCAL)
C
      DIMENSION
     + XP(NNODE),YP(NNODE),ZP(NNODE)
     +,VOL(NCELL),AREA(NFACE,3),VPOR(NCELL),ARPOR(NFACE,3)
     +,WFACT(NFACE)
     +,IPT(*),IBLK(5,NBLOCK)
     +,IPVERT(NCELL,8),IPNODN(NCELL,6),IPFACN(NCELL,6),IPNODF(NFACE,4)
     +,IPNODB(NBDRY,4),IPFACB(NBDRY)
     +,IWORK(*),WORK(*),CWORK(*)
C
C+++++++++++++++++ USER AREA 3 ++++++++++++++++++++++++++++++++++++++++
C---- AREA FOR USERS TO DIMENSION THEIR ARRAYS
C
C---- AREA FOR USERS TO DEFINE DATA STATEMENTS
C
C+++++++++++++++++ END OF USER AREA 3 ++++++++++++++++++++++++++++++++
C
C---- STATEMENT FUNCTION FOR ADDRESSING
      IP(I,J,K)=IPT((K-1)*ILEN*JLEN+(J-1)*ILEN+I)
C
C----VERSION NUMBER OF USER ROUTINE AND PRECISION FLAG
```

```
C
      IVERS=4
      ICHKPR = 1
C
C+++++++++++++++++ USER AREA 4 ++++++++++++++++++++++++++++++++++++++++++
C---- TO USE THIS USER ROUTINE FIRST SET IUSED=1
C
       IUSED=1
C
C+++++++++++++++++ END OF USER AREA 4 ++++++++++++++++++++++++++++++++++
C
       IF (IUSED.EQ.0) RETURN
C
C---- FRONTEND CHECKING OF USER ROUTINE
       IF (IUCALL.EQ.0) RETURN
C
C---- ADD TO SOURCE TERMS
      IF (ICALL.EQ.1) THEN
C

      ENDIF

C+++++++++++++++++ END OF USER AREA 5 ++++++++++++++++++++++++++++++++++
      ENDIF
C
C---- OVERWRITE SOURCE TERMS
      IF (ICALL.EQ.2) THEN
C
C+++++++++++++++++ USER AREA 6 ++++++++++++++++++++++++++++++++++++++++++
C
C
      ENDIF
      RETURN
      END
```

Appendix C

Fortran 90 source code

Several Fortran programs have been developed to implement the computational methods presented in this work. The programs are divided into three groups:

PBE: A set of subroutines for numerical solution of the population balance equation according to the finite element scheme presented in chap. 3.

Network: The computational implementation of the generalised compartmental model (chap. 2).

Interface: A PDE solver for the equations describing the interface (chap. 2).

Apart from the above, several auxiliary sets of subroutines are used to define problem-specific models:

Reaction: Defines the reaction kinetics.

Utilities: Provides functions for variables such as supersaturation, ionic strength etc.

Bulk: Provides a link with the ODE solver, in our case the LSODE (Hindmarsh, 1983).

Main: Initiates the computation.

C.1 PBE

```
!*****************************************************************************************
!
!                          FILE PBE_module.f90
!
! Calculates the nucleation, growth, aggregation and breakage source terms
! for temporal integration of the population densities at the nodal points of the grid
! discretised with the first-order conservative finite element collocation method
!
! This file contains:
! - Module PBE_module: all general declarations
! - Subroutine PBE_source: source terms due to finite element discretisation
! - Subroutine mesh: generates mesh and performs all preliminary operations
! - Subroutine agg_kernel_lib: library of aggregation kernels
! - Subroutine br_kernel_lib: library of breakage kernels
```

```
! - Subroutine PBE_initdata read values of particulate parameters
! - Subroutine PBE_initarrays: read values of particulate parameters and initialise distributions
!
! By Stelios Rigopoulos
! 11/01/2002
! 18/01/2002: first version of modular program
! 19/04/2002 new version
!
!*****************************************************************************************



!*****************************************************************************************

module PBE_module

!*****************************************************************************************
!
! Declaration of common variables related to grid and kernels
!
! by Stelios Rigopoulos
! 17/12/2001
! 19/04/2002 new version
!
!*****************************************************************************************

implicit none

real(8) :: agg_kernel, br_kernel_const, br_kernel_exp, br_dist_parabol
real(8) :: br_dist_gamma, br_dist_delta, br_dist_alpha
real(8) :: Nuc_ss_exp,G_ss_exp,Nuc_ss_coeff,G_ss_coeff
real(8) :: inc,v_nuc,v_0,threshold
real(8) :: N0,v0
real(8) :: G_coeff1,G_coeff2,G_coeff3,nuc,aggl_kernel_const

integer :: agg_kernel_type, breakage_kernel_type, breakage_dist_type, growth_kernel_type
integer :: speedupPBE,meshflag,initflag,inputflag
integer :: m

real(8), allocatable, dimension(:)       :: v,dv,v_exp13,v_exp23
integer, allocatable, dimension(:)       :: vhalf
integer, allocatable, dimension(:,:,:)   :: comp
real(8), allocatable, dimension(:,:,:,:) :: node,nodec
real(8), allocatable, dimension(:,:,:)   :: dvi,dvk
real(8), allocatable, dimension(:,:,:,:) :: agg_kernel_birth
real(8), allocatable, dimension(:,:,:)   :: agg_kernel_death
real(8), allocatable, dimension(:,:,:)   :: br_kernel_birth

!-----------------------------------------------------------------------------------------

end module PBE_module

!*****************************************************************************************



!*****************************************************************************************

subroutine PBESource(index,ni,Agg_birth,Agg_death,B_Birth,B_Death,Growth,Nuc_Birth)

!*****************************************************************************************
!
! Calculates the source term for temporal integration of the population balance
! Implementation for collocation on finite elements with linear base functions
! Data,kernels etc to be supplied by module "kinetics"
! Discrete pointwise evaluation of aggregation birth term
!
! This implementation is for:
```

```
! Any volume-dependent growth function if provided
! Any up to third order linear aggregation kernel if provided
! Any up to fifth order linear breakage kernel
! Higher or nonlinear order kernels: may be subject to error
!
! How kernels are set:
! - agg_kernel_order: order of the aggregation kernel, if linear can handle up to 3rd
! - br_kernel_order: total order of the breakage birth term = kernel + distribution + 1
!
! * Aggregation kernels
!   - Volume-independent kernel: set agg_kernel_type to 1
!   - Sum kernel: set agg_kernel_type to 2
!   - Product kernel: set agg_kernel_type to 3
!   - Smoluchowski kernel: set agg_kernel_type to 4
!   - Gravitational kernel: set agg_kernel_type to 5
!
! * Growth kernels
!   - Volume-independent kernel: set growth_kernel_type to 1
!   - Volume-deoendent due to G(v)=av^k: set growth_kernel_type to 2
!   - Volume-dependent due to Randolph and Larson function: set growth_kernel_type to 3
!
! * Breakage kernels
!   - Power law kernel, uniform binary breakage: set breakage_kernel_type to 1
!   - Power law kernel, parabolic binary breakage: set breakage_kernel_type to 2
!   - Power law kernel, multiple breakage according to simplified Ziff function: 3
!   - Power law kernel, multiple breakage according to generalised Ziff function: 4
!   - Exponential kernel: set breakage_kernel_type to 5
!
!                              * * * * * * * * *
!
! By Stelios Rigopoulos
! Development history:
! 30/10/2001: initial version
! 16/11/2001: improved conservation algorithm
! 02/12/2001: added breakage
! 03/12/2001: added product kernel
! 05/12/2001: new structure
! 11/12/2001: corrections
! 18/01/2002: new version for modular program
!
! Main variables
!
! m                    number of points
! index                point to be collocated
! ni                   population density
! ni_in                inlet population density
! v                    independent variable - volume
! dv                   lengths of intervals
!
!*****************************************************************************************

use PBE_module
!use reaction_module !for n

implicit none

integer, intent(in)                      :: index
real(8), intent(out)                     :: Agg_Birth,Agg_Death,B_birth,B_Death,Growth,Nuc_Birth

real(8), dimension(0:m), intent(in)      :: ni

!real(8), dimension(n), intent(in)        :: C

integer                                  :: i,k
integer                                  :: si
real(8)                                  :: term1,G_term1,G_term2
real(8)                                  :: n1,n3,nk1,nk3,v2,n2,vk2,nk2,dv13
!real(8)                                  :: G_coeff1,G_coeff2,G_coeff3,nuc
```

```
integer tflag
common tflag

!-----------------------------------------------------------------------------------------

!Growth term

!Dependency on environmental variables

!Call Growth_coeff(C)

!-----------------------------------------

if (growth_kernel_type==1) then

  !Volume-independent growth
  G_term1 = G_coeff1
  G_term2 = 0.

else if (growth_kernel_type==2) then

  !Volume-dependent growth, simple power law
  G_term1 = G_coeff1*(v(index)**G_coeff2)
  G_term2 = G_coeff1*G_coeff2*(v(index)**(G_coeff2-1.))

else if (growth_kernel_type==3) then

  !Volume-dependent growth, Randolph and Larson function
  G_term1 = G_coeff1*((1 + G_coeff2*v(index))**G_coeff3)
  G_term2 = G_coeff3*G_coeff1*G_coeff2*((1.+G_coeff2*v(index))**(G_coeff3-1.))

end if

Growth = - G_term1*ni(index)/dv(index) - G_term2*ni(index)

if (index>1) then

  Growth = Growth + G_term1*ni(index-1)/dv(index)       !dv(index-1)

end if

!-----------------------------------------------------------------------------------------

!Nucleation term

!Dependency on environmental variables

!Call Nucleation_rate(C)

!-----------------------------------------

if (index==1) then

  Nuc_Birth = Nuc*(2./(dv(1)+dv(2)))

else

  Nuc_Birth = 0.

end if

!-----------------------------------------------------------------------------------------

!Aggregation term

!Dependency on environmental variables
```

```
!Call Aggregation_rate(C,agg_kernel)

!----------------------------------------

!Aggregation death term

Agg_Death = 0.

if ((aggl_kernel_const>0.0).and.(tflag==1)) then

  do i=1,m

    if (agg_kernel_type==1) then

      Agg_Death = Agg_Death + dv(i) * 5.D-1 * ( ni(i-1) + ni(i))

    else

      dv13 = v(i)-v(i-1)
      v2 = 0.5*(v(i-1)+v(i))
      n2 = ni(i-1)*v(i)/dv13 - ni(i-1)*v2/dv13  +ni(i)*v2/dv13 - ni(i)*v(i-1)/dv13


      Agg_Death = Agg_Death + dV(i) * ( (1./6.)*ni(i-1)*agg_kernel_death(index,i,1) &
                & + (4./6.)*n2*agg_kernel_death(index,i,2) + &
                & (1./6.)*ni(i)*agg_kernel_death(index,i,3) )

    end if

  end do

  Agg_Death = - Agg_Death*agg_kernel*ni(index)

end if

!----------------------------------------

!Aggregation birth term

Agg_Birth = 0.

if ((index>0).and.(aggl_kernel_const>0.0).and.(tflag==1)) then

  !Integrate over intervals

  do i=1,vhalf(index)

    if (i==vhalf(index)) then

      if ( abs( v(index)*0.5-v(vhalf(index)-1) )<threshold ) then

        !no need to integrate element vhalf because v(index)/2 falls on a nodal point
        exit

      else

        si = 1

        n1 = ni(i-1)
        n3 = ni(i-1)*v(i)/dv(i) - ni(i-1)*node(index,i,si,3)/dv(i) + &
           & ni(i)*node(index,i,si,3)/dv(i) - ni(i)*v(i-1)/dv(i)
        nk3 = ni(i-1)*v(i)/dv(i) - ni(i-1)*nodec(index,i,si,3)/dv(i) + &
            & ni(i)*nodec(index,i,si,3)/dv(i) - ni(i)*v(i-1)/dv(i)
        nk1 = n3

        n2 = n1*node(index,i,si,3)/dvi(index,i,si) - n1*node(index,i,si,2)/dvi(index,i,si) + &
           & n3*node(index,i,si,2)/dvi(index,i,si) - n3*node(index,i,si,1)/dvi(index,i,si)
        nk2 = nk1*nodec(index,i,si,3)/dvk(index,i,si) - nk1*nodec(index,i,si,2)/dvk(index,i,si) + &
```

```
        & nk3*nodec(index,i,si,2)/dvk(index,i,si) - nk3*nodec(index,i,si,1)/dvk(index,i,si)

      Agg_birth = Agg_birth + dvi(index,i,si)* &
      & ( (1./6.)*n1*nk3*agg_kernel_birth(index,i,si,1) + &
        & (4./6.)*n2*nk2*agg_kernel_birth(index,i,si,2) + &
        & (1./6.)*n3*nk1*agg_kernel_birth(index,i,si,3) )

    end if

  else

    !Integrate over sub-intervals

    si = 0

    do k=comp(index,i,2),comp(index,i,1)

      !Define the sub-interval limits

      si = si + 1

      !Case 1:if sub-interval is a whole interval

      if ((comp(index,i,2)<k).and.(k<comp(index,i,1))) then

        nk1 = ni(k-1)
        nk3 = ni(k)
        n3 = ni(i-1)*v(i)/dv(i) - ni(i-1)*node(index,i,si,3)/dv(i) + &
           & ni(i)*node(index,i,si,3)/dv(i) - ni(i)*v(i-1)/dv(i)
        n1 = ni(i-1)*v(i)/dv(i) - ni(i-1)*node(index,i,si,1)/dv(i) + &
           & ni(i)*node(index,i,si,1)/dv(i) - ni(i)*v(i-1)/dv(i)

      !Case 2: if both points lie in the same interval

      else if  (comp(index,i,1)==comp(index,i,2)) then

        n1 = ni(i-1)
        n3 = ni(i)
        nk1 = ni(k-1)*v(k)/dv(k) - ni(k-1)*nodec(index,i,si,1)/dv(k) + &
            & ni(k)*nodec(index,i,si,1)/dv(k) - ni(k)*v(k-1)/dv(k)
        nk3 = ni(k-1)*v(k)/dv(k) - ni(k-1)*nodec(index,i,si,3)/dv(k) + &
            & ni(k)*nodec(index,i,si,3)/dv(k) - ni(k)*v(k-1)/dv(k)

      !Case 3: if sub-interval is the upper part of the primary interval

      else if (k==comp(index,i,2)) then

        n1 = ni(i-1)*v(i)/dv(i) - ni(i-1)*node(index,i,si,1)/dv(i) + &
           & ni(i)*node(index,i,si,1)/dv(i) - ni(i)*v(i-1)/dv(i)
        n3 = ni(i)
        nk1 = ni(k-1)*v(k)/dv(k) - ni(k-1)*nodec(index,i,si,1)/dv(k) + &
            & ni(k)*nodec(index,i,si,1)/dv(k) - ni(k)*v(k-1)/dv(k)
        nk3 = ni(k)

      !Case 4: if sub-interval is the lower part of the primary interval

      else if (k==comp(index,i,1)) then

        n1 = ni(i-1)
        n3 = ni(i-1)*v(i)/dv(i) - ni(i-1)*node(index,i,si,3)/dv(i) + &
           & ni(i)*node(index,i,si,3)/dv(i) - ni(i)*v(i-1)/dv(i)
        nk1 = ni(k-1)
        nk3 = ni(k-1)*v(k)/dv(k) - ni(k-1)*nodec(index,i,si,3)/dv(k) + &
            & ni(k)*nodec(index,i,si,3)/dv(k) - ni(k)*v(k-1)/dv(k)

      else

        write(*,*) 'severe exception, quitting'
```

```
          write(41,*) 'exception occurred at the sub-interval integration'
          stop

        end if

        n2 = n1*node(index,i,si,3)/dvi(index,i,si) - n1*node(index,i,si,2)/dvi(index,i,si) + &
           & n3*node(index,i,si,2)/dvi(index,i,si) - n3*node(index,i,si,1)/dvi(index,i,si)
        nk2 = nk1*nodec(index,i,si,3)/dvk(index,i,si) - nk1*nodec(index,i,si,2)/dvk(index,i,si) + &
            & nk3*nodec(index,i,si,2)/dvk(index,i,si) - nk3*nodec(index,i,si,1)/dvk(index,i,si)

        Agg_birth = Agg_birth + dvi(index,i,si)* &
        & ( (1./6.)*n1*nk3*agg_kernel_birth(index,i,si,1) + &
          & (4./6.)*n2*nk2*agg_kernel_birth(index,i,si,2) + &
          & (1./6.)*n3*nk1*agg_kernel_birth(index,i,si,3) )

      end do

    end if

  end do

  !Total birth term

  Agg_Birth = Agg_Birth*agg_kernel

end if

!-----------------------------------------------------------------------------------------------

!Breakage

!Dependency on environmental variables

!Call Breakage_rate(C,agg_kernel)

!-----------------------------------------

!Breakage birth term

B_birth = 0.

if ((abs(br_kernel_const)>0.).and.(index<m).and.(tflag==1)) then

  do i=(index+1),m

    dv13 = v(i)-v(i-1)
    v2 = 0.5*(v(i-1)+v(i))
    n2 = ni(i-1)*v(i)/dv13 - ni(i-1)*v2/dv13  +ni(i)*v2/dv13 - ni(i)*v(i-1)/dv13

    B_birth = B_birth + dv(i) * ( (1./6.)*ni(i-1)*br_kernel_birth(index,i,1) + &
    & (4./6.)*n2*br_kernel_birth(index,i,2) + &
    & (1./6.)*ni(i)*br_kernel_birth(index,i,3) )

  end do

end if

B_Birth = B_Birth*br_kernel_const

!-----------------------------------------

!Breakage Death term

B_Death = 0.

if ((abs(br_kernel_const)>0.).and.(tflag==1)) then

  if (breakage_kernel_type/=5) then
```

```
    B_Death =  - br_kernel_const*ni(index)*((v(index))**(br_kernel_exp))

  else if (breakage_kernel_type==5) then

    B_Death =  - ni(index)*br_kernel_const*exp(br_kernel_exp*v(index))

  end if

end if

!-----------------------------------------------------------------------------------------

!For scaled variables
!Agg_Birth = Agg_Birth * 5.23E-25
!Agg_Death = Agg_Death * 5.23E-25
!Growth = Growth * ((5.23E-25)**(-1./3.))

!-----------------------------------------------------------------------------------------

end subroutine PBESource

!*****************************************************************************************



!*****************************************************************************************

subroutine mesh()

!*****************************************************************************************
!
! Subroutine mesh
! Generates/reads the mesh
! Also determines complimentaries
!
! By Stelios Rigopoulos
! 31/10/2001 - initial version
! 17/11/2001 - added location of complimentaries
!
! m                     number of points
! ni                    population density
! v                     independent variable - volume
! dv                    lengths of intervals
! v_nuc                 size of nuclei - to be supplied by module kinetics
! comp(index,i,1)       element with point which, when added to v(i-1), will form v(index)
! comp(index,i,2)       element with point which, when added to v(i), will form v(index)
! inc                   factor by which the exponential grid is generated
!
!*****************************************************************************************

use PBE_module

implicit none

integer                                 :: i,index,k,k1,k2,l,si
real(8)                                 :: v_comp,temp,v_half
integer                                 :: maxdim

!-----------------------------------------------------------------------------------------

write(*,*) 'Creating mesh...'

allocate (v(0:m))
allocate (dv(m))

allocate (vhalf(m))
allocate (comp(m,m,2))
```

```
!-----------------------------------------------------------------------------------------

!Create mesh

if (meshflag==1) then

!Option 1: Exponential mesh

  v(0) = v_0
  v(1) = v_nuc !/10000.
  v(2) = v(1)+v_nuc !D-1 !2. !v_nuc

  do i=3,m

    v(i) = v_nuc + v_nuc*(1-inc**real(i-1))/(1-inc)

  end do

else if (meshflag==2) then

!Option 2: Uniform mesh

  v(0) = v_0
  v(1) = v_nuc

  do i=2,m

    v(i) = v(i-1) + inc

  end do

else if (meshflag==3) then

!Option 3: read from disk

  do i=0,m

    Read(63,*) v(i)

  end do

end if

!Element length

do i=1,m

  dv(i) = v(i)-v(i-1)

end do

!-----------------------------------------------------------------------------------------

!The following operations need to be carried out only when aggregation is present

if (abs(aggl_kernel_const)>threshold) then

!-----------------------------------------------------------------------------------------

!Initialise necessary arrays

write(*,*) 'Calculating sub-intervals...'

maxdim = 2

allocate (node(m,m,maxdim,3))
allocate (nodec(m,m,maxdim,3))
```

```
allocate (dvk(m,m,maxdim))
allocate (dvi(m,m,maxdim))

allocate(agg_kernel_birth(m,m,maxdim,3))
allocate(agg_kernel_death(m,m,3))

do index = 1,m

  do i=1,m

    comp(index,i,1) = 0.
    comp(index,i,2) = 0.

    do si=1,maxdim

      node(index,i,si,1)  = 0.
      nodec(index,i,si,1) = 0.
      node(index,i,si,2)  = 0.
      nodec(index,i,si,2) = 0.
      node(index,i,si,3)  = 0.
      nodec(index,i,si,3) = 0.

      dvi(index,i,si) = 0.
      dvk(index,i,si) = 0.

      agg_kernel_birth(index,i,si,1) = 0.
      agg_kernel_birth(index,i,si,2) = 0.
      agg_kernel_birth(index,i,si,3) = 0.

    end do

  agg_kernel_death(index,i,1) = 0.
  agg_kernel_death(index,i,2) = 0.
  agg_kernel_death(index,i,3) = 0.

  end do

end do

!-----------------------------------------------------------------------------------------------

!Locate the complementary points

do index=1,m

  !First locate the half volumes

  v_half = 5.D-1*v(index)

  k = 1

  do

    if (abs(v_half-v(k))<threshold) then

      !write(*,*) 'equality',index
      vhalf(index) = k + 1
      exit

    else if (v_half<v(k)) then

      vhalf(index) = k
      exit

    else

      k = k+1
```

```
    end if

  end do

  do i=1,index

    !Determine complimentaries

    k = 1

    v_comp = v(index)-v(i-1)

    do

      if (abs(v_comp-v(k))<threshold) then

        comp(index,i,1) = k
        exit

      else if (v_comp<v(k)) then

        comp(index,i,1) = k
        exit

      else

        k = k+1

      end if

    end do

    k = 1
    v_comp = v(index)-v(i)

    do

      if (abs(v_comp-v(k))<threshold) then

        comp(index,i,2) = k + 1
        exit

      else if (v_comp<v(k)) then

        comp(index,i,2) = k
        exit

      else

        k = k+1

      end if

    end do

  end do

end do

!-----------------------------------------------------------------------------------------------

!Integrate over intervals

do index=1,m

  do i=1,vhalf(index)
```

```
if (i==vhalf(index)) then

  if ( abs( v(index)*0.5-v(vhalf(index)-1) )<threshold ) then

    !no need to integrate element vhalf because v(index)/2 falls on a nodal point
    exit

  else

    si=1

    node(index,i,si,1)= v(i-1)
    node(index,i,si,3)=v(index)*0.5
    nodec(index,i,si,3) = v(index)*0.5 + ( v(index)*0.5-v(i-1) )
    nodec(index,i,si,1) = node(index,i,si,3)

    dvk(index,i,si) = nodec(index,i,si,3) - nodec(index,i,si,1)
    dvi(index,i,si) = node(index,i,si,3) - node(index,i,si,1)

    nodec(index,i,si,2) = 0.5*(nodec(index,i,si,1)+nodec(index,i,si,3))
    node(index,i,si,2) = 0.5*(node(index,i,si,1)+node(index,i,si,3))

  end if

else

  !Integrate over sub-intervals

  si = 0

  do k=comp(index,i,2),comp(index,i,1)

    !Define the sub-interval limits

    si = si + 1

    !Case 1:if sub-interval is a whole interval

    if ((comp(index,i,2)<k).and.(k<comp(index,i,1))) then

      nodec(index,i,si,1)=v(k-1)
      nodec(index,i,si,3)=v(k)
      node(index,i,si,1) = v(i) - ( nodec(index,i,si,3) - (v(index)-v(i)) )
      node(index,i,si,3) = v(i) - ( nodec(index,i,si,1) - (v(index)-v(i)) )

    !Case 2: if both points lie in the same interval

    else if  (comp(index,i,1)==comp(index,i,2)) then

      node(index,i,si,1)= v(i-1)
      node(index,i,si,3)= v(i)
      nodec(index,i,si,1) = v(index)-v(i)
      nodec(index,i,si,3) = v(index) - v(i-1)

    !Case 3: if sub-interval is the upper part of the primary interval

    else if (k==comp(index,i,2)) then

      node(index,i,si,1)= v(i) - (v(k) - (v(index)-v(i)))
      node(index,i,si,3)= v(i)
      nodec(index,i,si,1) = v(index)-v(i)
      nodec(index,i,si,3) = v(k)

    !Case 4: if sub-interval is the lower part of the primary interval

    else if (k==comp(index,i,1)) then

      node(index,i,si,1)= v(i-1)
```

```
          node(index,i,si,3)= v(i-1) + ((v(index)-v(i-1))-v(k-1))
          nodec(index,i,si,1) = v(k-1)
          nodec(index,i,si,3) = v(index) - v(i-1)

        else

          write(*,*) 'severe exception, quitting'
          write(41,*) 'exception occurred at the calculation of sub-interval nodes'
          stop

        end if

        dvk(index,i,si) = nodec(index,i,si,3) - nodec(index,i,si,1)
        dvi(index,i,si) = node(index,i,si,3) - node(index,i,si,1)

        nodec(index,i,si,2) = 0.5*(nodec(index,i,si,1)+nodec(index,i,si,3))
        node(index,i,si,2) = 0.5*(node(index,i,si,1)+node(index,i,si,3))

      end do

    end if

  end do

end do

!-----------------------------------------------------------------------------------------------

!Calculate kernels

!Value of kernel for birth term

write(*,*) 'computing aggregation kernels...'

do index=1,m

  do i=1,vhalf(index)

    if (i==vhalf(index)) then

      if ( abs( v(index)*0.5-v(vhalf(index)-1) )<threshold ) then

        !no need to integrate element vhalf because v(index)/2 falls on a nodal point
        exit

      else

        si=1

        call agg_kernel_lib(index,i,si,1)

      end if

    else

    si = 0

    do k = comp(index,i,2),comp(index,i,1)

      si = si + 1

      call agg_kernel_lib(index,i,si,1)

    end do

    end if

  end do
```

```
end do
!Value of kernel for death term
do index=1,m
  do i=1,m
    si=0
    call agg_kernel_lib(index,i,si,0)
  end do
end do
!Optional - min aggregation size
do index=1,m
  if (v(index)<((1./6.)*3.14*(1.5D-6)**3.)) then
!    call agg_kernel_min(index,1,maxdim,0)
  end if
end do
do index=1,m
  do i=1,m
    if ((v(i)+v(index))<((1./6.)*3.14*(1.5D-6)**3.)) then
!      call agg_kernel_min(index,0,maxdim,i)
    end if
  end do
end do
!-----------------------------------------------------------------------------------------------
write(*,*) 'preliminary aggregation operations complete'
end if
!-----------------------------------------------------------------------------------------------
!The following operations need to be carried out only when breakage is present
allocate(br_kernel_birth(m,m,3))
do index = 1,m
  do i=1,m
    br_kernel_birth(index,i,1) = 0.
    br_kernel_birth(index,i,2) = 0.
    br_kernel_birth(index,i,3) = 0.
  end do
end do
if (abs(br_kernel_const)>threshold) then
```

```
  do index=1,m-1

    do i=(index+1),m

      call br_kernel_lib(index,i)

    end do

  end do

  write(*,*) 'preliminary breakage operations complete'

end if

!-----------------------------------------------------------------------------------------

end subroutine mesh

!*****************************************************************************************



!*****************************************************************************************

Subroutine agg_kernel_lib(index,i,si,flag)

!*****************************************************************************************
!
! Library of aggregation kernels
!
! By Stelios Rigopoulos
! 18/04/2001 - initial version
!
!*****************************************************************************************

use PBE_module

implicit none

integer, intent(in) :: i,index,si,flag

integer             :: k

real(8) :: term,eff,Expfac,expexp
real(8) :: lamda,gamma

!-----------------------------------------------------------------------------------------

Expfac = 5.D10
Expexp = 0.65

lamda = 3.D5
gamma = 1.4

if (flag==1) then

  !Value of kernel for birth term

  if (agg_kernel_type==1) then

    !Volume-independent kernel

    agg_kernel_birth(index,i,si,1) = 1.
    agg_kernel_birth(index,i,si,2) = 1.
    agg_kernel_birth(index,i,si,3) = 1.

  else if (agg_kernel_type==2) then
```

```
  !Sum kernel

  agg_kernel_birth(index,i,si,1) = node(index,i,si,1) + nodec(index,i,si,3)
  agg_kernel_birth(index,i,si,2) = node(index,i,si,2) + nodec(index,i,si,2)
  agg_kernel_birth(index,i,si,3) = node(index,i,si,3) + nodec(index,i,si,1)

else if (agg_kernel_type==3) then

  !Product kernel

  agg_kernel_birth(index,i,si,1) = node(index,i,si,1) * nodec(index,i,si,3)
  agg_kernel_birth(index,i,si,2) = node(index,i,si,2) * nodec(index,i,si,2)
  agg_kernel_birth(index,i,si,3) = node(index,i,si,3) * nodec(index,i,si,1)

else if (agg_kernel_type==4) then

  !Fluid shear kernel

  term = ( node(index,i,si,1)**(1./3.)+nodec(index,i,si,3)**(1./3.) )
  agg_kernel_birth(index,i,si,1) = term*term*term

  term = ( node(index,i,si,2)**(1./3.)+nodec(index,i,si,2)**(1./3.) )
  agg_kernel_birth(index,i,si,2) = term*term*term

  term = ( node(index,i,si,3)**(1./3.)+nodec(index,i,si,1)**(1./3.) )
  agg_kernel_birth(index,i,si,3) = term*term*term

else if (agg_kernel_type==5) then

  !Differential force kernel

  agg_kernel_birth(index,i,si,1) = &
  & ( ((node(index,i,si,1)**(1./3.)+nodec(index,i,si,3)**(1./3.))**2.) * &
  & abs (((node(index,i,si,1)**(2./3.)-nodec(index,i,si,3)**(2./3.)))    ) )

  agg_kernel_birth(index,i,si,2) = &
  & ( ((node(index,i,si,2)**(1./3.)+nodec(index,i,si,2)**(1./3.))**2.) * &
  & abs (((node(index,i,si,2)**(2./3.)-nodec(index,i,si,2)**(2./3.)))    ) )

  agg_kernel_birth(index,i,si,3) = &
  & ( ((node(index,i,si,3)**(1./3.)+nodec(index,i,si,1)**(1./3.))**2.) * &
  & abs (((node(index,i,si,3)**(2./3.)-nodec(index,i,si,1)**(2./3.)))    ) )

else if (agg_kernel_type==6) then

  !Thompson kernel

  agg_kernel_birth(index,i,si,1) = &
  & ((nodec(index,i,si,3)-node(index,i,si,1))**2.)/(nodec(index,i,si,3)+node(index,i,si,1))

  agg_kernel_birth(index,i,si,2) = &
  & ((nodec(index,i,si,2)-node(index,i,si,2))**2.)/(nodec(index,i,si,2)+node(index,i,si,2))

  agg_kernel_birth(index,i,si,3) = &
  & ((nodec(index,i,si,1)-node(index,i,si,3))**2.)/(nodec(index,i,si,1)+node(index,i,si,3))

else if (agg_kernel_type==7) then

  !Fluid shear kernel, volume-dependent efficiency

  term = ( node(index,i,si,1)**(1./3.)+nodec(index,i,si,3)**(1./3.) )
  agg_kernel_birth(index,i,si,1) = term*term*term/(node(index,i,si,1)+nodec(index,i,si,3))

  term = ( node(index,i,si,2)**(1./3.)+nodec(index,i,si,2)**(1./3.) )
  agg_kernel_birth(index,i,si,2) = term*term*term/(node(index,i,si,2)+nodec(index,i,si,2))

  term = ( node(index,i,si,3)**(1./3.)+nodec(index,i,si,1)**(1./3.) )
```

```
    agg_kernel_birth(index,i,si,3) = term*term*term/(node(index,i,si,3)+nodec(index,i,si,1))

  else if (agg_kernel_type==8) then

    !Constant kernel, volume-dependent efficiency

    agg_kernel_birth(index,i,si,1) = 1./v(index)

    agg_kernel_birth(index,i,si,2) = 1./v(index)

    agg_kernel_birth(index,i,si,3) = 1./v(index)

  else if (agg_kernel_type==9) then

    !Fluid shear kernel, volume-dependent efficiency function

    term = ( node(index,i,si,1)**(1./3.)+nodec(index,i,si,3)**(1./3.) )
!   eff = 1. - ( (node(index,i,si,1)+nodec(index,i,si,3))/Vcrit**Expcrit )
!   eff = (1./v(index))**1.
    eff = exp(-Expfac*v(index)**expexp)
    if (eff<0.) then
      eff = 0.
    end if
    agg_kernel_birth(index,i,si,1) = term*term*term*eff

    term = ( node(index,i,si,2)**(1./3.)+nodec(index,i,si,2)**(1./3.) )
!   eff = 1. - ( (node(index,i,si,2)+nodec(index,i,si,2))/Vcrit**Expcrit )
!   eff = (1./v(index))**1.
    eff = exp(-Expfac*v(index)**expexp)
    if (eff<0.) then
      eff = 0.
    end if
    agg_kernel_birth(index,i,si,2) = term*term*term*eff

    term = ( node(index,i,si,3)**(1./3.)+nodec(index,i,si,1)**(1./3.) )
!   eff = 1. - ( (node(index,i,si,3)+nodec(index,i,si,1))/Vcrit**Expcrit )
!   eff = (1./v(index))**1.
    eff = exp(-Expfac*v(index)**expexp)
    if (eff<0.) then
      eff = 0.
    end if
    agg_kernel_birth(index,i,si,3) = term*term*term*eff

  else if (agg_kernel_type==10) then

    !Empirical kernel

    term = ( node(index,i,si,1)**(1./3.)*nodec(index,i,si,3)**(1./3.) )
    eff =  exp(-lamda*node(index,i,si,1)**(1./3.)) * 1./( (node(index,i,si,1)+&
    &nodec(index,i,si,3))**gamma )
    if (eff<0.) then
!     eff = 0.
    end if
    agg_kernel_birth(index,i,si,1) = term*eff


    term = ( node(index,i,si,2)**(1./3.)*nodec(index,i,si,2)**(1./3.) )
    eff = exp(-lamda*node(index,i,si,2)**(1./3.)) * 1./( (node(index,i,si,2)+&
    &nodec(index,i,si,2))**gamma )
    if (eff<0.) then
!     eff = 0.
    end if
    agg_kernel_birth(index,i,si,2) = term*eff

    term = ( node(index,i,si,3)**(1./3.)*nodec(index,i,si,1)**(1./3.) )
    eff = exp(-lamda*node(index,i,si,3)**(1./3.)) * 1./( (node(index,i,si,3)+&
    &nodec(index,i,si,1))**gamma )
    if (eff<0.) then
```

```
!     eff = 0.
    end if
    agg_kernel_birth(index,i,si,3) = term*eff

  else

    write(*,*) 'invalid data, fatal'
    stop

  end if

  !-----------------------------------------------------

else if (flag==0) then

  !Value of kernel for death term

  if (agg_kernel_type==1) then

    !Constant kernel

    agg_kernel_death(index,i,1) = 1.
    agg_kernel_death(index,i,2) = 1.
    agg_kernel_death(index,i,3) = 1.

  else if (agg_kernel_type==2) then

    !Sum kernel

    agg_kernel_death(index,i,1) = v(i-1)+v(index)
    agg_kernel_death(index,i,2) = 0.5*(v(i-1)+v(i)) + v(index)
    agg_kernel_death(index,i,3) = v(i)+v(index)

  else if (agg_kernel_type==3) then

    !Product kernel

    agg_kernel_death(index,i,1) = v(i-1)*v(index)
    agg_kernel_death(index,i,2) = 0.5*(v(i-1)+v(i)) * v(index)
    agg_kernel_death(index,i,3) = v(i)*v(index)

  else if (agg_kernel_type==4) then

    !Fluid shear kernel

    term = v(i-1)**(1./3.)+v(index)**(1./3.)
    agg_kernel_death(index,i,1) = term*term*term

    term = (0.5*(v(i-1)+v(i)))**(1./3.)+v(index)**(1./3.)
    agg_kernel_death(index,i,2) = term*term*term

    term = v(i)**(1./3.)+v(index)**(1./3.)
    agg_kernel_death(index,i,3) = term*term*term

  else if (agg_kernel_type==5) then

    !Differential force kernel

    agg_kernel_death(index,i,1) = &
    & ( (v(i-1)**(1./3.)+v(index)**(1./3.))**2.) * &
    & abs ( (v(i-1)**(2./3.)-v(index)**(2./3.)) )

    agg_kernel_death(index,i,2) = &
    & ( ((0.5*(v(i-1)+v(i)))**(1./3.)+v(index)**(1./3.))**2.) * &
    & abs (((0.5*(v(i-1)+v(i)))**(2./3.)-v(index)**(2./3.)))

    agg_kernel_death(index,i,3) = &
    & ( (v(i)**(1./3.)+v(index)**(1./3.))**2.) * &
```

```
    & abs ((v(i)**(2./3.)-v(index)**(2./3.)))

  else if (agg_kernel_type==6) then

    !Thompson kernel

    agg_kernel_death(index,i,1) = &
    & ((v(i-1)-v(index))**2.)/(v(i-1)+v(index))

    agg_kernel_death(index,i,2) = &
    & ((0.5*(v(i-1)+v(i))-v(index))**2.)/(0.5*(v(i-1)+v(i))+v(index))

    agg_kernel_death(index,i,3) = &
    & ((v(i)-v(index))**2.)/(v(i)+v(index))

  else if (agg_kernel_type==7) then

    !Fluid shear kernel, volume-dependent efficiency

    term = v(i-1)**(1./3.)+v(index)**(1./3.)
    agg_kernel_death(index,i,1) = term*term*term/(v(index)+v(i-1))

    term = (0.5*(v(i-1)+v(i)))**(1./3.)+v(index)**(1./3.)
    agg_kernel_death(index,i,2) = term*term*term/(v(index)+(0.5*(v(i-1)+v(i))))

    term = v(i)**(1./3.)+v(index)**(1./3.)
    agg_kernel_death(index,i,3) = term*term*term/(v(index)+v(i))

  else if (agg_kernel_type==8) then

    !Constant kernel, volume-dependent efficiency

    agg_kernel_death(index,i,1) = 1./(v(index)+v(i-1))

    agg_kernel_death(index,i,2) = 1./(v(index)+(0.5*(v(i-1)+v(i))))

    agg_kernel_death(index,i,3) = 1./(v(index)+v(i))

  else if (agg_kernel_type==9) then

    !Fluid shear kernel, volume-dependent efficiency function

    term = v(i-1)**(1./3.)+v(index)**(1./3.)
    !eff = 1. - ( (v(i-1)+v(index))/Vcrit**Expcrit )
    !eff = (1./(v(i-1)+v(index)))**1.
    eff = exp(-Expfac*(v(i-1)+v(index))**expexp)
    if (eff<0.) then
      eff = 0.
    end if
    agg_kernel_death(index,i,1) = term*term*term*eff

    term = (0.5*(v(i-1)+v(i)))**(1./3.)+v(index)**(1./3.)
    !eff = 1. - ( (0.5*(v(i-1)+v(i))+v(index))/Vcrit**Expcrit )
    !eff = (1./(0.5*(v(i-1)+v(i))+v(index)))**1.
    eff = exp(-Expfac*(0.5*(v(i-1)+v(i))+v(index))**expexp)
    if (eff<0.) then
      eff = 0.
    end if
    agg_kernel_death(index,i,2) = term*term*term*eff

    term = v(i)**(1./3.)+v(index)**(1./3.)
    !eff = 1. - ( (v(i)+v(index))/Vcrit**Expcrit )
    !eff = (1./(v(i)+v(index)))**1.
    eff = exp(-Expfac*(v(i)+v(index))**expexp)
    if (eff<0.) then
      eff = 0.
    end if
    agg_kernel_death(index,i,3) = term*term*term*eff
```

```
  else if (agg_kernel_type==10) then

    !Empirical kernel

    term = ( v(i-1)**(1./3.)*v(index)**(1./3.) )
    eff =  exp(-lamda*v(i-1)**(1./3.)) * 1./( (v(i-1)+v(index))**gamma )
    if (eff<0.) then
!     eff = 0.
    end if
    agg_kernel_death(index,i,1) = term*eff

    term = ( (0.5*(v(i-1)+v(i)))**(1./3.)*v(index)**(1./3.) )
    eff =   exp(-lamda*(0.5*(v(i-1)+v(i)))**(1./3.)) *
    &1./( ((0.5*(v(i-1)+v(i)))+v(index))**gamma )
    if (eff<0.) then
!     eff = 0.
    end if
    agg_kernel_death(index,i,2) = term*eff


    term = ( v(i)**(1./3.)*v(index)**(1./3.) )
    eff =  exp(-lamda*v(i)**(1./3.)) * 1./( (v(i)+v(index))**gamma )
    if (eff<0.) then
!     eff = 0.
    end if
    agg_kernel_death(index,i,3) = term*eff

  else

  write(*,*) 'invalid data, fatal'
  stop

  end if

end if

end subroutine agg_kernel_lib

!*****************************************************************************************



!*****************************************************************************************

Subroutine agg_kernel_max(index,flag,maxdim,i1)

!*****************************************************************************************
!
! Upper limit of aggregation kernel
!
! By Stelios Rigopoulos
! 28/05/2001 - initial version
!
!*****************************************************************************************

use PBE_module

implicit none

integer, intent(in)          :: index,flag,maxdim

integer, intent(in)          :: i1

integer                      :: i,si

!-----------------------------------------------------------------------------------------
```

```
if (flag==1) then

  do i=1,m

    do si=1,maxdim

      agg_kernel_birth(index,i,si,1) = 0.
      agg_kernel_birth(index,i,si,2) = 0.
      agg_kernel_birth(index,i,si,3) = 0.

    end do

  end do

else

  agg_kernel_death(index,i1,1) = 0.
  agg_kernel_death(index,i1,2) = 0.
  agg_kernel_death(index,i1,3) = 0.

end if

end subroutine agg_kernel_max

!*****************************************************************************************



!*****************************************************************************************

Subroutine agg_kernel_min(index,flag,maxdim,i1)

!*****************************************************************************************
!
! Lower limit of aggregation kernel
!
! By Stelios Rigopoulos
! 20/08/2001 - initial version
!
!*****************************************************************************************

use PBE_module

implicit none

integer, intent(in)           :: index,flag,maxdim

integer, intent(in)           :: i1

integer                       :: i,si

!-----------------------------------------------------------------------------------------

if (flag==1) then

  do i=1,m

    do si=1,maxdim

      agg_kernel_birth(index,i,si,1) = 0.
      agg_kernel_birth(index,i,si,2) = 0.
      agg_kernel_birth(index,i,si,3) = 0.

    end do

  end do

else
```

```
  agg_kernel_death(index,i1,1) = 0.
  agg_kernel_death(index,i1,2) = 0.
  agg_kernel_death(index,i1,3) = 0.

end if

end subroutine agg_kernel_min

!*****************************************************************************************



!*****************************************************************************************

Subroutine br_kernel_lib(index,i)

!*****************************************************************************************
!
! Library of breakage kernels
!
! By Stelios Rigopoulos
! 18/04/2001 - initial version
!
!*****************************************************************************************

use PBE_module

implicit none

integer, intent(in) :: i,index

real(8) :: term1,v2

!-----------------------------------------------------------------------------------------

v2 =  0.5*(v(i-1)+v(i))

if (breakage_dist_type==1) then

  !Binary breakage, uniform fragmentation

  br_kernel_birth(index,i,1) = 2.*(1./v(i-1))
  br_kernel_birth(index,i,2) = 2.*(1./v2)
  br_kernel_birth(index,i,3) = 2.*(1./v(i))

else if (breakage_dist_type==2) then

  !Binary breakage, parabolic distribution optimised for attrition

  br_kernel_birth(index,i,1) = &
  & ( 24 * ( v(index)**2. - v(index)*v(i-1) + (v(i-1)**2.)/4. ) / (v(i-1)**3.) )

  br_kernel_birth(index,i,2) = &
  & ( 24 * ( v(index)**2. - v(index)*v2 + (v2**2.)/4. ) / (v2**3.) )

  br_kernel_birth(index,i,3) = &
  & ( 24 * ( v(index)**2. - v(index)*v(i) + (v(i)**2.)/4. ) / (v(i)**3.) )

else if (breakage_dist_type==3) then

  !multiple breakage according to simplified Ziff function

  term1=br_dist_delta*br_kernel_exp/(br_dist_delta-br_kernel_exp)

  br_kernel_birth(index,i,1) = &
  & ( (1/v(i-1))*(term1) * ( (v(index)/v(i-1))**(br_kernel_exp-2.)-(v(index)/v(i-1))&
  & **(br_dist_delta-2.) ) )
```

```
  br_kernel_birth(index,i,2) = &
  &  ( 1/v2*(term1) * ( (v(index)/v2)**(br_kernel_exp-2.)-(v(index)/v2) &
  & **(br_dist_delta-2.) ) )

  br_kernel_birth(index,i,3) = &
  &  ( 1/v(i)*(term1) * ( (v(index)/v(i))**(br_kernel_exp-2.)-(v(index)/v(i))&
  & **(br_dist_delta-2.) ) )

else if (breakage_dist_type==4) then

  !multiple breakage according to generalised Ziff function

  br_kernel_birth(index,i,1) = &
  & ( (br_dist_alpha*br_dist_gamma/v(i-1))*(v(index)/v(i-1))**(br_dist_gamma-2.) &
  & + ((1-br_dist_alpha)*br_dist_delta/v(i-1))*(v(index)/v(i-1))**(br_dist_delta-2.) )

  br_kernel_birth(index,i,2) = &
  & ( (br_dist_alpha*br_dist_gamma/v2)*(v(index)/v2)**(br_dist_gamma-2.) + &
  & ((1-br_dist_alpha)*br_dist_delta/v2)*(v(index)/v2)**(br_dist_delta-2.) )

  br_kernel_birth(index,i,3) = &
  & ( (br_dist_alpha*br_dist_gamma/v(i))*(v(index)/v(i))**(br_dist_gamma-2.) &
  & + ((1-br_dist_alpha)*br_dist_delta/v(i))*(v(index)/v(i))**(br_dist_delta-2.) )

else if (breakage_dist_type==5) then

  !Binary breakage, generalised parabolic distribution

  br_kernel_birth(index,i,1) = &
  & ( 24*(1.-br_dist_parabol/2.) * ( v(index)**2.-v(index)*v(i-1)+(v(i-1)**2.)/4. ) &
  & / (v(i-1)**3.) + br_dist_parabol/v(i-1) )

  br_kernel_birth(index,i,2) = &
  & ( 24*(1.-br_dist_parabol/2.) * ( v(index)**2.-v(index)*v2+(v2**2.)/4. ) &
  & / (v2**3.) + br_dist_parabol/v2 )

  br_kernel_birth(index,i,3) =  &
  & ( 24*(1.-br_dist_parabol/2.) * ( v(index)**2.-v(index)*v(i)+(v(i)**2.)/4. ) &
  & / (v(i)**3.) + br_dist_parabol/v(i) )

else if (breakage_dist_type==6) then

!Erosion

  if ((i==index-1).or.(i==1)) then

  br_kernel_birth(index,i,1) = 1.
  br_kernel_birth(index,i,2) = 1.
  br_kernel_birth(index,i,3) = 1.

  else

  br_kernel_birth(index,i,1) = 0.
  br_kernel_birth(index,i,2) = 0.
  br_kernel_birth(index,i,3) = 0.

  end if

else

  write(*,*) 'invalid data, fatal'
  stop

end if

if (breakage_kernel_type==1) then
```

```
  !Power law kernel

  br_kernel_birth(index,i,1) = br_kernel_birth(index,i,1) * (v(i-1)**(br_kernel_exp))
  br_kernel_birth(index,i,2) = br_kernel_birth(index,i,2) * (v2**(br_kernel_exp))
  br_kernel_birth(index,i,3) = br_kernel_birth(index,i,3) * (v(i)**(br_kernel_exp))

else if (breakage_kernel_type==2) then

  !Exponential kernel

  br_kernel_birth(index,i,1) = br_kernel_birth(index,i,1) * exp(br_kernel_exp*v(i-1))
  br_kernel_birth(index,i,2) = br_kernel_birth(index,i,2) * exp(br_kernel_exp*v2)
  br_kernel_birth(index,i,3) = br_kernel_birth(index,i,3) * exp(br_kernel_exp*v(i))

else

  write(*,*) 'invalid data, fatal'
  stop

end if

end subroutine br_kernel_lib

!*****************************************************************************************



!*****************************************************************************************

subroutine PBE_initdata()

!*****************************************************************************************
!
! Initialises kinetic data
! By Stelios Rigopoulos
! 09/02/2002
!
!*****************************************************************************************

use PBE_module

implicit none

integer i,k,l

!-----------------------------------------------------------------------------------------

!Read all kinetic etc. data

do i=1,8

  read(31,*)

end do

read(31,5002) Agg_kernel_type
read(31,5001) aggl_kernel_const
read(31,5002) breakage_kernel_type
read(31,5002) breakage_dist_type
read(31,5001) br_kernel_const
read(31,5001) br_kernel_exp
read(31,5001) br_dist_parabol
read(31,5001) br_dist_gamma
read(31,5001) br_dist_delta
read(31,5001) br_dist_alpha
read(31,5002) growth_kernel_type
!read(31,5001) G_coeff1
!read(31,5001) G_coeff2
```

```
!read(31,5001) G_coeff3
!read(31,5001) Nuc

do i=1,3

  read(31,*)

end do

read(31,5001) Nuc_ss_coeff
read(31,5001) Nuc_ss_exp
read(31,5001) G_ss_coeff
read(31,5001) G_ss_exp

do i=1,3

  read(31,*)

end do

read(31,5002) meshflag
read(31,5002) m
read(31,5001) inc
read(31,5001) v_nuc
read(31,5001) v_0

do i=1,3

  read(31,*)

end do

read(31,5001) threshold
!read(31,5002) speedupPBE

!-----------------------------------------------------------------------------------------

5001 format(29X,F20.10)
5002 format(29X,I10)
5003 format(I6,F20.10)

end subroutine PBE_initdata

!*****************************************************************************************



!*****************************************************************************************

subroutine PBE_initarrays(ni,nin)

!*****************************************************************************************
!
! Initialises arrays
! By Stelios Rigopoulos
! 09/02/2002
!
!*****************************************************************************************

use PBE_module

implicit none

real(8), dimension(0:m), intent(out)      :: ni,nin

integer i,k,l

!-----------------------------------------------------------------------------------------
```

```
!Initialise population density arrays

do i=1,m

  ni(i) = 0.0
  nin(i) = 0.0

end do

ni(0) = 0.0
nin(0) = 0.0

!-----------------------------------------------------------------------------------------------

!Initial distribution

do i=1,6

  read(34,*)

end do

read(34,5002) initflag

ni(0) = 0. !Initial condition
ni(m) = 0.

if (initflag==0) then

  !Zero initial distribution

  do i=1,m-1

      ni(i) = 0. !Underflow check

  end do

else if (initflag==1) then

  !exponential initial distribution

  do i=1,5

    read(34,*)

  end do

  read(34,5001) N0
  read(34,5001) V0

  do i=1,m-1

    if ((i>1).and.(ni(i-1)<threshold)) then

      ni(i) = 0. !Underflow check

    else

      ni(i) =  N0/v0 * exp(-(v(i))/v0)

    end if

  end do

else if (initflag==2) then

  !Read from disk
```

```
  do i=1,12

    read(34,*)

  end do

  read (34,5002) k

  do i=1,k

    read(34,5003) l,ni(l)

  end do

end if

!-----------------------------------------------------------------------------------------------

!Input distribution

do i=1,6

  read(35,*)

end do

read(35,5002) inputflag

nin(0) = 0. !Initial condition
nin(m) = 0.

if (inputflag==0) then

  !Zero initial distribution

  do i=1,m-1

      nin(i) = 0. !Underflow check

  end do

else if (inputflag==1) then

  !exponential initial distribution

  do i=1,5

    read(35,*)

  end do

  read(35,5001) N0
  read(35,5001) V0

  do i=1,m-1

    if ((i>1).and.(nin(i-1)<threshold)) then

      nin(i) = 0. !Underflow check

    else

      nin(i) =  N0/v0 * exp(-(v(i))/v0)

    end if

  end do
```

```
else if (inputflag==2) then

  !Read from disk

  do i=1,12

    read(35,*)

  end do

  read (35,5002) k

  do i=1,k

    read(35,5003) l,nin(l)

  end do

end if

!-----------------------------------------------------------------------------------------

!Write initial and input distribution

do i=1,m

  write (111,*) v(i),ni(i)

end do

do i=1,m

  write (113,*) v(i),nin(i)

end do

!-----------------------------------------------------------------------------------------

5001 format(29X,F20.10)
5002 format(29X,I10)
5003 format(I6,F20.10)

end subroutine PBE_initarrays

!*****************************************************************************************
```

C.2 Network

```
!*****************************************************************************************
!
!                          FILE Network_module.f90
!
! By Stelios Rigopoulos
! Contains:
! - module network_module
! - subroutine network_initdata
! - subroutine gas_initarrays
! - subroutine network
!
! 05/02/2002: initial version
! 18/02/2002: new version - all in one subroutine
! 24/02/2002: first complete version
! 26/02/2002: corrected for output in gas flow
! 21/06/2002: new version for reaction network
```

```
!
! ATTENTION
!
! The following must be manually inserted:
! Dimensions in classes (network module)
! Dimension in Cin in functions F and bulk (reaction module)
!
!*****************************************************************************************



!*****************************************************************************************

module network_module

!*****************************************************************************************
!
! By Stelios Rigopoulos
! 18/02/2002: initial version
!
!*****************************************************************************************

implicit none

!-----------------------------------------------------------------------------------------

real(8) :: total_time
real(8) :: Pressure
real(8) :: Temperature
real(8) :: R_constant
real(8) :: d_bubble
real(8) :: contact_time
real(8) :: interface_depth
real(8) :: relerr_conc,relerr_pdensity
integer :: bulk_lin_itmax
integer :: interval_writeup

end module network_module

!*****************************************************************************************



!*****************************************************************************************

subroutine network_initdata

!*****************************************************************************************
!
! By Stelios Rigopoulos
! 18/02/2002: initial version
!
!*****************************************************************************************

use network_module

implicit none

integer k

!-----------------------------------------------------------------------------------------

do k=1,6

  read(81,*)

end do
```

```
read(81,1003) total_time
read(81,1003) Pressure
read(81,1003) Temperature
read(81,1003) R_constant
read(81,1003) d_bubble
read(81,1003) contact_time
read(81,1003) interface_depth
read(81,1003) relerr_conc
read(81,1003) relerr_pdensity
read(81,1001) bulk_lin_itmax
read(81,1001) interval_writeup
read(81,*)

1003 format(29X,F20.10)
1001 format(29X, I10)

end subroutine network_initdata

!*****************************************************************************************


!*****************************************************************************************

subroutine gas_initarrays(y_mol,n)

!*****************************************************************************************
!
! Initialises arrays
! By Stelios Rigopoulos
! 09/02/2002
!
!*****************************************************************************************

implicit none

integer, intent(in) :: n

real(8), dimension(n,2), intent(inout) :: y_mol

integer i

!-----------------------------------------------------------------------------------------

!Initial concentrations

rewind(83)

do i=1,8

  read(83,*)

end do

do i=1,n

  read(83,5001) y_mol(i,1),y_mol(i,2)

end do

5001 format(29X,F10.8,13X,F5.3)

end subroutine gas_initarrays

!*****************************************************************************************


!*****************************************************************************************
```

```
subroutine network()

!*****************************************************************************************
!
! By Stelios Rigopoulos
! 05/02/2002: initial version
! 18/02/2002: new version - all in one subroutine
!
! Generation of the reaction engineering network
! Controls the program flow
! Temporal loop according to the penetration theory
!
!*****************************************************************************************

use PBE_module
use reaction_module
use network_module

implicit none

!-----------------------------------------------------------------------------------------

!General declarations

!integer, allocatable, dimension(:,:,:) :: comp
!real(8), allocatable, dimension(:)     :: v,dv
!integer, allocatable, dimension(:)     :: vhalf

real(8), allocatable, dimension(:)     :: pdensity_in
real(8), allocatable, dimension(:)     :: concentration_in
real(8), allocatable, dimension(:)     :: mol
!real(8), allocatable, dimension(:)     :: Dif

real(8) :: total_M0,total_M1
real(8) :: totalmol

integer :: read_flow_flag

integer :: cells_X
integer :: cells_Y

integer :: i,j,k,l,k2
!integer :: m,n
integer :: index_ct,N_of_ct,bulk_lin_iter
integer :: iter_coup,maxiter_coup
integer :: converge
integer :: count

real(8) :: time
real(8) :: integration_time

real(8) :: weight_in,weight_out
real(8) :: dM0
real(8) :: vold
real(8) :: ni_temp,C_temp,temp,temp_2

real(8) :: vol_growthrate

real(8) :: tot_moles,tot_conc,tot_volume

!-----------------------------------------

!Specification of the generic class that describes the building block
!of the reaction engineering network

!Subclasses
```

```
type volume_class

  real(8) total
  real(8) bulk
  real(8) int
  real(8) gas

end type volume_class

type left_class

  integer X
  integer Y
  real flow
  real flow_2

end type left_class

type right_class

  integer X
  integer Y
  real flow
  real flow_2

end type right_class

type up_class

  integer X
  integer Y
  real flow
  real flow_2

end type up_class

type down_class

  integer X
  integer Y
  real flow
  real flow_2

end type down_class

!Main class

type cell_class

  !Location of cell
  integer :: X
  integer :: Y

  !Directions of mass exchange
  type (left_class)  :: left
  type (right_class) :: right
  type (up_class)    :: up
  type (down_class)  :: down

  !Input and cell concentrations in both phases
  real(8), dimension(6)     :: Cin,C
  real(8), dimension(6)     :: C_ave,C_temp
  real(8), dimension(6)     :: Cin_2,C_2

  !Input and cell population densities
  real(8), dimension(0:100)   :: ni,nin
  real(8), dimension(0:100)   :: ni_ave,ni_temp
```

```
  !Gas molar fractions
  real(8), dimension(6,2) :: y_mol

  !Amount of absorption of each gaseous component
  real(8),dimension(6) :: absorption

  !Total moles in the interface
  real(8), dimension (6) :: mol_interface

  !Moments
  real(8), dimension(0:1) :: Moment_bulk

  !Volume, total and of each phase
  type (volume_class) :: volume

  !Flow in and out of the system
  real(8) :: flow_in
  real(8) :: flow_out
  real(8) :: flow_2_in
  real(8) :: flow_2_out

  !Interfacial area
  real(8) :: A_interface

  !Mean diameter of particles
  real(8) :: Dmean

  !Temporary storage of population at nuclei size
  real(8) :: ni1_prev

end type cell_Class


!Define the compartments as an array of objects of the cell_class

type(cell_class), allocatable, dimension(:,:)   :: cell

!-----------------------------------------------------------------------------------------------

!                                        Main program

!-----------------------------------------------------------------------------------------------

!                                       Initialisation

m=size(cell(1,1)%ni)-1
n=size(cell(1,1)%C)

!Allocate global arrays

!allocate (v(0:m))
!allocate (dv(m))

!allocate (vhalf(m))
!allocate (comp(m,m,2))

allocate(pdensity_in(0:m))
allocate(concentration_in(n))

allocate(mol(n))

call file_open(1)

!Initialise population balance parameters

call PBE_initdata()

!Create mesh - uniform for all cells
```

```
!call mesh()

!Read environmental etc. parameters

call network_initdata()

!Initiate parameters related to interface

call interface_initdata()

!Initialise reaction engineering parameters

call reaction_initdata()

!Initialise bulk integration

call integrator_initdata()

!-----------------------------------------------------------------------------------------

!Create network

!Read network parameters

read(81,1001) cells_X
read(81,1001) cells_Y

allocate(cell(cells_Y,cells_X))

!Generate topology

do i=1,cells_Y !rows iteration

  do j=1,cells_X !columns iteration

    !Location of cell

    cell(i,j)%X = j
    cell(i,j)%Y = i

    !Location and connectivity of left cell

    if (j==1) then

      cell(i,j)%left%X  = 0
      cell(i,j)%left%Y  = 0

    else

      cell(i,j)%left%X  = j-1
      cell(i,j)%left%Y  = i

    end if

    !Location and connectivity of right cell

    if (j==cells_X) then

      cell(i,j)%right%X  = 0
      cell(i,j)%right%Y  = 0

    else

      cell(i,j)%right%X = j+1
      cell(i,j)%right%Y = i

    end if
```

```
    !Location and connectivity of up cell

    if (i==1) then

      cell(i,j)%up%X    = 0
      cell(i,j)%up%Y    = 0

    else

      cell(i,j)%up%X    = j
      cell(i,j)%up%Y    = i-1

    end if

    !Location and connectivity of down cell

    if (i==cells_Y) then

      cell(i,j)%down%X  = 0
      cell(i,j)%down%Y  = 0

    else

      cell(i,j)%down%X  = j
      cell(i,j)%down%Y  = i+1

    end if

  end do

end do

1003 format(29X,F20.10)
1001 format(29X, I10)

!-----------------------------------------------------------------------------------------------

!Read/generate flows

read(81,*)
read(81,1113) read_flow_flag

if (read_flow_flag==1) then

  do i=1,8

    read(81,*)

  end do

  i=0
  j=0

  do k=1,(cells_Y*cells_X)

    read(81,1110,advance = 'no') i,j

    read(81,1111) &
                & cell(i,j)%left%flow, cell(i,j)%right%flow, &
                & cell(i,j)%up%flow, cell(i,j)%down%flow, &
                & cell(i,j)%flow_in, cell(i,j)%flow_out, &
                & cell(i,j)%left%flow_2, cell(i,j)%right%flow_2, &
                & cell(i,j)%up%flow_2, cell(i,j)%down%flow_2, &
                & cell(i,j)%flow_2_in, cell(i,j)%flow_2_out, &
                & cell(i,j)%volume%bulk, cell(i,j)%volume%gas

  end do
```

```
end if

1113 format(29X,I10)
1110 format (1X,I4,I4,1X)
1111 format (F10.7,F10.7,F10.7,F10.7,F10.7,F10.7,1X,F7.4,F7.4,F7.4,F7.4,F7.4,F7.4,1X, E10.3,F8.4)

!Mass balance check

do i=1,cells_Y

  do j=1,cells_X

    temp = - cell(i,j)%left%flow - cell(i,j)%right%flow &
         & - cell(i,j)%up%flow - cell(i,j)%down%flow &
         & - cell(i,j)%flow_out

    temp_2 = - cell(i,j)%left%flow_2 - cell(i,j)%right%flow_2 &
         & - cell(i,j)%up%flow_2 - cell(i,j)%down%flow_2 &
         & - cell(i,j)%flow_2_out

    temp = temp + cell(i,j)%flow_in
    temp_2 = temp_2 + cell(i,j)%flow_2_in

    if ((cell(i,j)%left%Y>0).and.(cell(i,j)%left%X>0)) then

      temp = temp + cell(cell(i,j)%left%Y,cell(i,j)%left%X)%right%flow
      temp_2 = temp_2 + cell(cell(i,j)%left%Y,cell(i,j)%left%X)%right%flow_2

    end if

    if ((cell(i,j)%right%Y>0).and.(cell(i,j)%right%X>0)) then

      temp = temp + cell(cell(i,j)%right%Y,cell(i,j)%right%X)%left%flow
      temp_2 = temp_2 + cell(cell(i,j)%right%Y,cell(i,j)%right%X)%left%flow_2

    end if

    if ((cell(i,j)%up%Y>0).and.(cell(i,j)%up%X>0)) then

      temp = temp + cell(cell(i,j)%up%Y,cell(i,j)%up%X)%down%flow
      temp_2 = temp_2 + cell(cell(i,j)%up%Y,cell(i,j)%up%X)%down%flow_2

    end if

    if ((cell(i,j)%down%Y>0).and.(cell(i,j)%down%X>0)) then

      temp = temp + cell(cell(i,j)%down%Y,cell(i,j)%down%X)%up%flow
      temp_2 = temp_2 + cell(cell(i,j)%down%Y,cell(i,j)%down%X)%up%flow_2

    end if

    if (abs(temp)>1.D-8) then

      write(*,*) 'incorrect flows in the bulk phase, mass balance not satisfied'
      write(*,*) 'cell: ',i,j,' residual: ',abs(temp)
      write(*,*) 'press any key to continue'
      pause

    end if

    if (abs(temp_2)>1.D-8) then

      write(*,*) 'incorrect flows in the dispersed phase, mass balance not satisfied'
      write(*,*) 'cell: ',i,j,' residual: ',abs(temp)
      write(*,*) 'press any key to continue'
      pause
```

```
    end if

  end do

end do

!-----------------------------------------------------------------------------------------------

!Initialise population densities, concentrations and other variables

do i=1,cells_Y

  do j=1,cells_X

    call reaction_initarrays(cell(i,j)%C,cell(i,j)%Cin)
    call gas_initarrays(cell(i,j)%y_mol,n)

    do k=1,n

      cell(i,j)%C_2(k) = cell(i,j)%y_mol(k,1)*Pressure/(R_constant*Temperature)
      cell(i,j)%Cin_2(k) = cell(i,j)%C_2(k)

    end do

    do k=0,m

      cell(i,j)%ni(k)  = 0.
      cell(i,j)%nin(k) = 0.
      cell(i,j)%ni_temp(k) = 0.
      cell(i,j)%ni_ave(k) = cell(i,j)%ni(k)

    end do

    do k=1,n

      cell(i,j)%c_temp(k) = 0.
      cell(i,j)%c_ave(k) = cell(i,j)%c(k)

    end do

    do k=1,n

      cell(i,j)%mol_interface(k) = 0.

    end do

    do k=1,n

      cell(i,j)%absorption(k) = 0.

    end do

    do k=0,1

      cell(i,j)%Moment_bulk = 0.

    end do

    cell(i,j)%ni1_prev = 0.

    cell(i,j)%Dmean = 0.

    cell(i,j)%A_interface = 6.* cell(i,j)%volume%gas/d_bubble

    cell(i,j)%volume%int = cell(i,j)%A_interface * interface_depth

    !Correct volume of bulk by subtracting the interface
    cell(i,j)%volume%total = cell(i,j)%volume%bulk + cell(i,j)%volume%gas
```

```
    cell(i,j)%volume%bulk = cell(i,j)%volume%bulk - cell(i,j)%volume%int

  end do

end do

vol_growthrate = 0.

!Impulse

!cell(2,1)%C(1) = 1.

!-----------------------------------------------------------------------------------------------

!                                        Time loop

N_of_ct = int(total_time/contact_time)
time = 0.
count = 0

!Main loop (over contact times)

!Initial write-up ----------
write(*,*) 'Beginning of calculation'
write(*,*) ' '
!---------------------------

do index_ct=1,N_of_ct

  !Time step write-up --------
  write(*,*) ' '
  write(*,*) 'time is ', time
  write(*,*) ' '
  !---------------------------

!-----------------------------------------------------------------------------------------------

  !Step feed

!  do i=1,cells_Y

!    cell(i,1)%C(1) = cell(i,1)%C(1) + 0.01*contact_time

!  end do

  cell(1,1)%C(1) = cell(1,1)%C(1) + 0.01*contact_time * 1.2
  cell(2,1)%C(1) = cell(2,1)%C(1) + 0.01*contact_time* 1.2
  cell(3,1)%C(1) = cell(3,1)%C(1) + 0.01*contact_time* 1.1
  cell(4,1)%C(1) = cell(4,1)%C(1) + 0.01*contact_time* 1.
  cell(5,1)%C(1) = cell(5,1)%C(1) + 0.01*contact_time* 1.
  cell(6,1)%C(1) = cell(6,1)%C(1) + 0.01*contact_time* 1.
  cell(7,1)%C(1) = cell(7,1)%C(1) + 0.01*contact_time* 0.9
  cell(8,1)%C(1) = cell(8,1)%C(1) + 0.01*contact_time* 0.9
  cell(9,1)%C(1) = cell(9,1)%C(1) + 0.01*contact_time* 0.8
  cell(10,1)%C(1) = cell(10,1)%C(1) + 0.01*contact_time* 0.7
  cell(10,2)%C(1) = cell(10,2)%C(1) + 0.01*contact_time* 0.2

!-----------------------------------------------------------------------------------------------

  !Step 1: Detachment - calculation of interface

  !Interface write-up --------
  write(*,*) 'Calculating interfaces...'
  !---------------------------

  do i=1,cells_Y

    do j=1,cells_X
```

```
      if (cell(i,j)%volume%int > 0.) then

        Call Interface_element(cell(i,j)%C,cell(i,j)%A_interface,&
                              & cell(i,j)%y_mol,cell(i,j)%mol_interface)

      end if

    end do

  end do

  !Interface write-up --------
  write(*,*) 'Interfaces calculated'
  write(*,*) ' '
  !---------------------------

!-----------------------------------------------------------------------------------------------

  !Step 2: Calculation of bulk by iteration over cells

  !Bulk write-up -------------
  write(*,*) 'Calculating bulk...'
  !---------------------------

  bulk_lin_iter = 0

  do !Linearisation

    !Temporary storage of bulk concentrations and population densities
    !to use until convergence of C_ave is achieved

    do i=1,cells_Y

      do j=1,cells_X

        do l=0,m

          cell(i,j)%ni_temp(l) = cell(i,j)%ni(l)

        end do

        do k=1,n

          cell(i,j)%C_temp(k) = cell(i,j)%C(k)

        end do

      end do

    end do

    temp = 0.

    do i=1,cells_Y

      do j=1,cells_X

        !Calculate inputs and outputs for cell (X,Y)

        !-----------------------------------

        weight_in = (1./(cell(i,j)%volume%bulk))


        !Input population density -----------
        !Includes the flows from the respective compartments
```

```
do k=0,m

  pdensity_in(k) = 0.

  !Contribution from left

  if ((cell(i,j)%left%Y>0).and.(cell(i,j)%left%X>0)) then

    pdensity_in(k) = pdensity_in(k) + cell(cell(i,j)%left%Y,cell(i,j)%left%X)%ni_ave(k) * &
                                    & cell(cell(i,j)%left%Y,cell(i,j)%left%X)%right%flow

  end if

  !Contribution from right

  if ((cell(i,j)%right%Y>0).and.(cell(i,j)%right%X>0)) then

    pdensity_in(k) = pdensity_in(k) + cell(cell(i,j)%right%Y,cell(i,j)%right%X)%ni_ave(k) * &
                                    & cell(cell(i,j)%right%Y,cell(i,j)%right%X)%left%flow

  end if

  !Contribution from up

  if ((cell(i,j)%up%Y>0).and.(cell(i,j)%up%X>0)) then

    pdensity_in(k) = pdensity_in(k) + cell(cell(i,j)%up%Y,cell(i,j)%up%X)%ni_ave(k) * &
                                    & cell(cell(i,j)%up%Y,cell(i,j)%up%X)%down%flow

  end if

  !Contribution from down

  if ((cell(i,j)%down%Y>0).and.(cell(i,j)%down%X>0)) then

    pdensity_in(k) = pdensity_in(k) + cell(cell(i,j)%down%Y,cell(i,j)%down%X)%ni_ave(k) * &
                                    & cell(cell(i,j)%down%Y,cell(i,j)%down%X)%up%flow

  end if

  !Contribution from input to the system

  pdensity_in(k) = pdensity_in(k) + cell(i,j)%nin(k) * cell(i,j)%flow_in

end do

!Input concentration -------------

do k=1,n

  concentration_in(k) = 0.

  !Contribution from left

  if ((cell(i,j)%left%Y>0).and.(cell(i,j)%left%X>0)) then

    concentration_in(k) = concentration_in(k) + cell(cell(i,j)&
    &%left%Y,cell(i,j)%left%X)%C_ave(k) * &
     & cell(cell(i,j)%left%Y,cell(i,j)%left%X)%right%flow

  end if

  !Contribution from right

  if ((cell(i,j)%right%Y>0).and.(cell(i,j)%right%X>0)) then

    concentration_in(k) = concentration_in(k) + cell(cell(i,j)&
    &%right%Y,cell(i,j)%right%X)%C_ave(k) * &
```

```
            & cell(cell(i,j)%right%Y,cell(i,j)%right%X)%left%flow

        end if

        !Contribution from up

        if ((cell(i,j)%up%Y>0).and.(cell(i,j)%up%X>0)) then

          concentration_in(k) = concentration_in(k) + cell(cell(i,j)&
          &%up%Y,cell(i,j)%up%X)%C_ave(k) * &
              & cell(cell(i,j)%up%Y,cell(i,j)%up%X)%down%flow

        end if

        !Contribution from down

        if ((cell(i,j)%down%Y>0).and.(cell(i,j)%down%X>0)) then

          concentration_in(k) = concentration_in(k) + cell(cell(i,j)&
          &%down%Y,cell(i,j)%down%X)%C_ave(k) * &
              & cell(cell(i,j)%down%Y,cell(i,j)%down%X)%up%flow

        end if

        !Contribution from input to the system

        concentration_in(k) = concentration_in(k) + cell(i,j)%Cin(k) * cell(i,j)%flow_in

      end do

      !---------------------------------

      !The minus of output is in subroutine RHSide

      weight_out = (1./(cell(i,j)%volume%bulk))*&
                 & ( cell(i,j)%left%flow  + &
                 &   cell(i,j)%right%flow + &
                 &   cell(i,j)%up%flow    + &
                 &   cell(i,j)%down%flow  + &
                 &   cell(i,j)%flow_out )

      !---------------------------------

      !Calculate new bulk concentrations

      integration_time = contact_time
      iter_coup = 0

      Call Bulk(cell(i,j)%ni_temp,pdensity_in,cell(i,j)%C_temp,concentration_in,&
      & weight_in,weight_out,integration_time,&
      & cell(i,j)%moment_bulk,vol_growthrate)

      if (iter_coup>maxiter_coup) then

        maxiter_coup = iter_coup

      end if

      !---------------------------------

    end do

  end do

  !Update the average concentrations and densities and check for convergence

  converge = 1
```

```
do i=1,cells_Y

  do j=1,cells_X

    do k=1,n

      temp = cell(i,j)%C_ave(k)
      cell(i,j)%C_ave(k) = 5.D-1 * (cell(i,j)%C(k)+cell(i,j)%C_temp(k))

      if ((cell(i,j)%C_ave(k)>0.).and. &
      & ((abs((temp-cell(i,j)%C_ave(k))/cell(i,j)%C_ave(k))>relerr_conc))) then

        converge = 0

      end if

    end do

    do k=1,m

      temp = cell(i,j)%ni_ave(k)
      cell(i,j)%ni_ave(k) = 5.D-1 * (cell(i,j)%ni(k)+cell(i,j)%ni_temp(k))

      if ((cell(i,j)%ni_ave(k)>0.).and. &
      & ((abs((temp-cell(i,j)%ni_ave(k))/cell(i,j)%ni_ave(k))>relerr_pdensity))) then

        converge = 0

      end if

    end do

  end do

end do

!If linearisation did not converge, continue the linearisation loop

if  (converge == 0) then

  bulk_lin_iter = bulk_lin_iter + 1

  if (bulk_lin_iter>bulk_lin_itmax) then

    write(*,*) 'no convergence of network linearisation after iterations ', bulk_lin_iter
    stop

  else

     go to 500

  end if

end if

!If converged, accept new values and exit loop

do i=1,cells_Y

  do j=1,cells_X

    do k=1,n

      cell(i,j)%C(k) = cell(i,j)%C_temp(k)
      cell(i,j)%C_ave(k) = cell(i,j)%C_temp(k)

    end do
```

```
        do k=1,m

          cell(i,j)%ni(k) = cell(i,j)%ni_temp(k)
          cell(i,j)%ni_ave(k) = cell(i,j)%ni_temp(k)

        end do

      end do

    end do

    temp = 0.
    exit !from linearisation loop

    500 continue !the linearisation - if not converged

  end do

  !Bulk write-up -------------
  write(*,*) 'Bulk calculated'
  write(*,*) 'Iterations needed for cells coupling: ', bulk_lin_iter
  write(*,*) 'Iterations (max) needed for substrate-PSD coupling: ', maxiter_coup
  write(*,*) ' '
  !---------------------------

  !End of bulk calculation

!-----------------------------------------------------------------------------------------------

  !Step 3: Mixing of interface and bulk

  do i=1,cells_Y

    do j=1,cells_X

      if (cell(i,j)%volume%int > 0.) then

        do k=1,n

          cell(i,j)%C(k) = ( cell(i,j)%mol_interface(k) + cell(i,j)%C(k)*cell(i,j)%volume%bulk ) &
                         & / (cell(i,j)%volume%bulk + cell(i,j)%volume%int)

        end do

      end if

    end do

  end do

  do i=1,cells_Y

    do j=1,cells_X

      total_M0 = total_M0 + cell(i,j)%moment_bulk(0)
      total_M1 = total_M1 + cell(i,j)%moment_bulk(1)*cell(i,j)%volume%bulk

    end do

  end do

  !Results write-up ----------
  write(*,*) 'Total particles: ', total_M0
  write(*,*) 'Total volume of particulate: ', total_M1
  !write(*,*) 'Mean concentrations: ', (Mean_C(k),k=1,n)
  write(*,*) ' '
  !---------------------------
```

```
  !Update flows to assist convergence of next bulk linearisation

  do i=1,cells_Y

    do j=1,cells_X

      do k=1,n

        cell(i,j)%C_ave(k) = cell(i,j)%C(k)

      end do

      do k=1,m

        cell(i,j)%ni_ave(k) = cell(i,j)%ni(k)

      end do

    end do

  end do

!-----------------------------------------------------------------------------------------------

  !Step 4: Gas

  do i=1,cells_Y

    do j=1,cells_X

      if (cell(i,j)%volume%gas > 0.) then

        !Molar mass balance

        totalmol = 0.

        do k2=1,n

          mol(k2) = cell(i,j)%C_2(k2)*cell(i,j)%volume%gas

          !Flow in from neighboring cells

          if ((cell(i,j)%left%Y>0).and.(cell(i,j)%left%X>0)) then

            mol(k2) = mol(k2) + cell(cell(i,j)%left%Y,cell(i,j)%left%X)%C_2(k2) * &
                            & cell(cell(i,j)%left%Y,cell(i,j)%left%X)%right%flow_2 * contact_time

          end if

          if ((cell(i,j)%right%Y>0).and.(cell(i,j)%right%X>0)) then

            mol(k2) = mol(k2) + cell(cell(i,j)%right%Y,cell(i,j)%right%X)%C_2(k2) * &
                            & cell(cell(i,j)%right%Y,cell(i,j)%right%X)%left%flow_2 * contact_time

          end if

          if ((cell(i,j)%down%Y>0).and.(cell(i,j)%down%X>0)) then

            mol(k2) = mol(k2) + cell(cell(i,j)%down%Y,cell(i,j)%down%X)%C_2(k2) * &
                            & cell(cell(i,j)%down%Y,cell(i,j)%down%X)%up%flow_2 * contact_time

          end if

          if ((cell(i,j)%up%Y>0).and.(cell(i,j)%up%X>0)) then

            mol(k2) = mol(k2) + cell(cell(i,j)%up%Y,cell(i,j)%up%X)%C_2(k2) * &
                            & cell(cell(i,j)%up%Y,cell(i,j)%up%X)%down%flow_2 * contact_time
```

```
          end if

          !Flow into the system

          mol(k2) = mol(k2) + cell(i,j)%Cin_2(k2) * cell(i,j)%flow_2_in * contact_time

          !Absorption

          mol(k2) = mol(k2) - cell(i,j)%mol_interface(k2)

          !Flow out of the system

          mol(k2) = mol(k2) - cell(i,j)%C_2(k2) * cell(i,j)%flow_2_out * contact_time

          !Flow out to neighbouring cells

          mol(k2) = mol(k2) - cell(i,j)%C_2(k2) * &
                  & (cell(i,j)%left%flow_2 + cell(i,j)%right%flow_2 + &
                  &  cell(i,j)%up%flow_2 + cell(i,j)%down%flow_2 ) &
                  & * contact_time

          if (mol(k2)<0.) then

            mol(k2) = 0.

          end if

          totalmol = totalmol + mol(k2)

        end do

        !Calculate new volume

        vold = cell(i,j)%volume%gas
        cell(i,j)%volume%gas = totalmol*R_constant*temperature/pressure

        !Calculate new interfacial area

        cell(i,j)%A_interface = 6.* cell(i,j)%volume%gas/d_bubble

        !Calculate new Concentration

        do k2=1,n

          cell(i,j)%C_2(k2) = mol(k2)/cell(i,j)%volume%gas

        end do

        !Gas flow correction

        cell(i,j)%left%flow_2 = cell(i,j)%left%flow_2 * cell(i,j)%volume%gas / vold
        cell(i,j)%right%flow_2 = cell(i,j)%right%flow_2 * cell(i,j)%volume%gas / vold
        cell(i,j)%up%flow_2 = cell(i,j)%up%flow_2 * cell(i,j)%volume%gas / vold
        cell(i,j)%down%flow_2 = cell(i,j)%down%flow_2 * cell(i,j)%volume%gas / vold
        cell(i,j)%flow_2_out = cell(i,j)%flow_2_out * cell(i,j)%volume%gas / vold

      end if

    end do

  end do

!-----------------------------------------------------------------------------------------------

  !Final write-up ----------
  write(*,*) 'Total particles: ', total_M0
  write(*,*) 'Total volume of particulate: ', total_M1
  !write(*,*) 'Mean concentrations: ', (Mean_C(k),k=1,n)
```

```
  write(*,*) ' '

  time = index_ct*contact_time

!  write(130,1103) (v(k),k=1,m)

  count = count+1
  if (count>=interval_writeup) then

    count=0

    write(310,*)
    write(310,1100) time

    tot_volume = 0.
    tot_moles = 0.

    do i=1,cells_Y

      do j=1,cells_X

        tot_volume = tot_volume + cell(i,j)%volume%bulk
        tot_moles = tot_moles + cell(i,j)%C(1)*cell(i,j)%volume%bulk

      end do

    end do

    tot_conc = tot_moles/tot_volume

    do i=cells_Y,1,-1

      write(310,1101) ((cell(i,j)%C(k)/tot_conc,k=1,n),j=1,cells_X)

    end do

      write(311,1101) time, (cell(2,i)%C(1),i=1,cells_X)
!     write(311,1101) time, (cell(i+1,1)%C(1)-cell(i,1)%C(1),i=1,cells_Y-1)

    !do i=1,cells_Y

    !  do j=1,cells_X

    !   write(130,1105) i,j
    !    write(130,1103) (cell(i,j)%ni(k),k=1,m)

    !  end do

    !end do

  end if

  1100 format(F10.5)
  1101 format(<n*cells_X>(F15.9,3X))
  1103 format(<m*cells_X>(E15.7,3X))
  1104 format(<cells_Y>(F10.4,E15.7,3X))
  1105 format(I5,I5)

  !---------------------------

!-----------------------------------------------------------------------------------------------

end do

end subroutine network

!*********************************************************************************************
```

C.3 Interface

```
!*****************************************************************************************
!
!                        FILE Interface_module.f90
!
! Solution of interfacial phenomena
! This file contains:
! - Module PBE_module: all general declarations
! - Subroutine Interface_intdata: initialise the interface
! - Subroutine Interface_element: solution of interfacial phenomena
! - Subroutine Derive: calculate partial derivatives by perturbation
! - Subroutine Integrate_int: used to calculate total mol and moments in the interface
! - Subroutine Thomas: solution of tridiagonal systems using the Thomas algorithmse distributions
!
! By Stelios Rigopoulos
! 18/02/2002: initial version
! 12/04/2002: without zeroth moment implemented in bubble column
!
!*****************************************************************************************



!*****************************************************************************************

module interface_module

implicit none

real(8) :: contact_time,interface_depth, k_l,gas_holdup
real(8) :: Pressure,Temperature,Henry
real(8) :: alpha,pdeerr
integer :: Nx,Nt_int,Itmax_int,interval_int

!-----------------------------------------------------------------------------------------

end module interface_module

!*****************************************************************************************



!*****************************************************************************************

subroutine interface_initdata(Dif)

!*****************************************************************************************
!
! Reads integration-related data
! By Stelios Rigopoulos
! 18/02/2002
!
!*****************************************************************************************

use reaction_module !for n
use interface_module

implicit none

real(8), dimension(n), intent(inout)   :: Dif

integer i

!-----------------------------------------------------------------------------------------

do i=1,6
```

```
  read(71,*)

end do

read(71,5001) gas_holdup
read(71,5001) k_l
read(71,5001) interface_depth
read(71,5001) Pressure
read(71,5001) Temperature
read(71,5001) Henry
read(71,5001) alpha
read(71,5001) pdeerr
read(71,5002) Nx
read(71,5002) Nt_int
read(71,5002) Itmax_int
read(71,5002) interval_int

do i=1,3

  read(71,*)

end do

do i=1,n

  read(71,5001) Dif(i)

end do

5001 format(29X,F20.10)
5002 format(29X,I10)

end subroutine interface_initdata

!*****************************************************************************************



!*****************************************************************************************

subroutine Interface_element(Cinitial,interface_area,Difspecies,y_gas,&
                            & mol_interface)

!*****************************************************************************************
!
! Calculates the interfacial element according to the penetration model
! By Stelios Rigopoulos
! 18/02/2002
!
!*****************************************************************************************

use PBE_module
use interface_module
use reaction_module
use integrator_module

implicit none

real(8), intent(in)                 :: interface_area      !Interfacial area

real(8), dimension(n,2), intent(in) :: y_gas               !Molar fraction of gas species
real(8), dimension(n), intent(in)   :: Cinitial            !Initial concentrations
real(8), dimension(n), intent(in)   :: Difspecies          !Species' diffusion coefficients

real(8), dimension(n), intent(out)  :: mol_interface       !Total mol and moments in the end

real(8), dimension(n)               :: C0                  !Initial concentration and moments vector
real(8), dimension(n)               :: Dif                 !Diffusivity vector
```

```
real(8), dimension(Nx,n)          :: C                  !Concentrations and moments
                                                        !matrix (distributed on x and t)
real(8), dimension(Nx,n)          :: Cprev              !Cocentrations at previous time step
real(8), dimension(Nx,n)          :: Ctemp              !Concentrations at previous iteration
                                                        !(for linearisation)
real(8), dimension(Nt_int)        :: SSu                !Supersaturation

real(8)                           :: Area               !Integral of concentrations over contact time

real(8), dimension(Nx,Nx)         :: Coeff              !Coefficients matrix
real(8), dimension(Nx)            :: Constant           !Constants array
real(8), dimension(Nx)            :: Conc               !Array returned by Thomas algorithm

real(8), dimension(Nx,n,n)        :: pd                 !Partial derivative of equation m at point
                                                        ! Nx with respect to component m
real(8), dimension(n)             :: deriv              !Subroutine, calculates the partial
                                                        ! derivatives of the source terms
real(8)                           :: pder,source

real(8), dimension(n)             :: A_int               !Interphase concentration of volatile component

integer                           :: Converge           !Variable used in convergence check
integer                           :: iter               !Iteration index
integer                           :: count

integer                           :: i,j,k,t,x          !i:reaction index, j:species index

real(8)                           :: dx,dt              !length of spatial and temporal steps
real(8)                           :: pH                 !Function, calculates pH
real(8)                           :: vol_growthrate

integer                           :: jcrash

!-----------------------------------------------------------------------------------------

!Not used
vol_growthrate = 0.

!Preliminary calculations

!Calculation of interfacial equilibrium concentrations for volatile components

call Henry_const(Cinitial,Henry)

!write(*,*) Henry

do i=1,n

  if (y_gas(i,2)>0.) then

    A_int(i) = y_gas(i,1)*Pressure/Henry

  end if

end do

!Create the initial concentration/moment vector

do i=1,n

  C0(i) = Cinitial(i)
  Dif(i) = Difspecies(i)

end do

!Initial Conditions
```

```
do j=1,n

  do x=1,Nx

    C(x,j) = C0(j)

  end do

end do

count = 0
!Length of steps
dt=contact_time/Nt_int
dx = interface_depth/Nx

!-----------------------------------------------------------------------------------------------

!Time iteration

do t=1,Nt_int

  !Store the concentrations of the previous time step (for implicit scheme)

  do j=1,n

    do x=1,Nx

      Cprev(x,j) = C(x,j)

    end do

  end do

  !-------------------------------------------------------

  !Calculate all source terms and partial derivatives

  do i=1,n    !Loop over equations

    do x=1,Nx     !Loop over space points

      Call derive(Cprev(x,1:n),i,deriv,n) !Partial derivatives of equation i at point x

      do j=1,n

        pd(x,i,j) = deriv(j) !Partial derivative of equation i at point x with respect to species j

      end do

    end do

  end do

  !Linearisation iteration -------------------------------

  iter=0

  do       !To be repeated until linearisation has converged

    !Keep current values to evaluate convergence

    do j=1,n

      do x=1,Nx

        Ctemp(x,j) = C(x,j)
```

```
    end do
  end do
  !Generation of coefficients matrix -------------------
  do i=1,n   !Loop over equations
    do x=1,Nx
      pder = 0.0
      !Calculate source terms and partial derivatives at spatial point
      do j=1,n      !Loop over species
        pder = pder + ( pd(x,i,j)*(C(x,j)-Cprev(x,j)) )
      end do
      pder = pder - pd(x,i,i)*(C(x,i)-Cprev(x,i))
      !Coefficients ----------------------
      !Middle coefficient
      Coeff(x,x)   = dx + 2.0*alpha*Dif(i)*dt/dx - alpha*dx*dt*pd(x,i,i)
      !Left coefficient (not for first point)
      if (x>1) then
        Coeff(x,x-1) = -Dif(i)*dt/dx
      end if
      !Right coefficient (not for last point)
      if (x<Nx) then
        Coeff(x,x+1) = -Dif(i)*dt/dx
      end if
      !Constant term
      !source(Cprev(x,1:j),i)
      Call ReactionSource(i,Cprev(x,1:n),source)
      Constant(x)  = Cprev(x,i)*dx &
                   & - 2.0*Dif(i)*(1-alpha)*Cprev(x,i)*dt/dx &
                   & + source*dx*dt &
                   & + pder*dx*dt &
                   & - alpha*dx*dt*Cprev(x,i)*pd(x,i,i)
    end do    !End of space loop
    !Boundary conditions -----------------
    !1: Interface
    if (y_gas(i,2)>0.) then
      !Volatile component - Dirichlet condition
      Coeff(1,1) = Coeff(1,1) + alpha*Dif(i)*dt/dx
      Constant(1) = Constant(1) - Dif(i)*(1-alpha)*Cprev(1,i)*dt/dx + 2.0*A_int(i)*Dif(i)*dt/dx
```

```
    else

      !Non-volatile components - Newmann condition

      Coeff(1,1) = Coeff(1,1) - alpha*Dif(i)*dt/dx
      Constant(1) = Constant(1) + Dif(i)*(1-alpha)*Cprev(1,i)*dt/dx

    end if

    !2: Bulk - Dirichlet condition for all

    Coeff(Nx,Nx) = Coeff(Nx,Nx) + alpha*Dif(i)*dt/dx
    Constant(Nx) = Constant(Nx) - Dif(i)*(1-alpha)*Cprev(Nx,i)*dt/dx + 2.0*C0(i)*Dif(i)*dt/dx

    !-----------------------------------------------

    !Solve the algebraic equations

    Call Thomas(Conc,Coeff,Constant,Nx)

    do x=1,Nx

      C(x,i) = Conc(x)

    end do

  end do !End of equations loop

  !-------------------------------------------------------------

  ! Check for convergence

  Converge=1

  do j=1,n

    do x=1,Nx

      if ((abs(Ctemp(x,j))<1.D-50).or.(abs((Ctemp(x,j)-C(x,j))/Ctemp(x,j))<pdeerr)) then

        converge = converge * 1

      else

        converge = converge * 0
        jcrash = j

      end if

    end do

  end do

  if (Converge==1) then

    exit

  else if ((Converge==0).and.(iter>Itmax_int)) then

    write(*,*) 'Outer iteration in the interface did not converge after steps ',Iter
    write(*,*) 'crashed on component', jcrash
    stop

  else

    iter = iter + 1
```

```
    end if

  end do    !End of linearisation iteration --------------------

end do    !End of time integration

!-----------------------------------------------------------------------------------------------

  !Interface write-up

rewind(115)
write(115,*) t*dt

do i=1,Nx

  write(115,1198) real(i)*dx, C(i,1),C(i,2),C(i,3),C(i,4),C(i,5)

end do

!Calculate total mol in the interfacial element

do j=1,n

  !Area under curve C(x)-x
  Call Integrate_int(Area,C(1:Nx,j),dx,Nx)

  mol_interface(j) = Area*Interface_Area

  if ((mol_interface(j)<0.).and.(abs(mol_interface(j))>1.D-50)) then

    write(*,*) 'negative flux, stop - component ', j, ' flux', mol_interface(j)
    write(200,*) 'negative flux, stop - component ', j, ' flux', mol_interface(j)
!    stop

  end if

end do

1198 format (F10.8,3X,<n>(E15.9,3X))

end subroutine Interface_element

!*********************************************************************************************



!*********************************************************************************************

subroutine derive(C,index,deriv,n)

!*********************************************************************************************

!Calculates the partial derivatives of the non-linear terms
!Numerical
!22/01/2001

!*********************************************************************************************

implicit none

integer, intent(in)                :: n

real(8), dimension(n), intent(in)  :: C
integer, intent(in)                :: index
real(8), dimension(n), intent(out) :: deriv

real(8), dimension(n)              :: dC
real(8), dimension(n)              :: CNew
```

```
integer                          :: i,j
real(8)                          :: source
real(8)                          :: r,rnew
real(8)                          :: vol_growthrate !CAREFUL

!-----------------------------------------------------------------------------------------

!Not used
vol_growthrate = 0.

!Perturbation vector

do i=1,n

  dC(i) = 1.0D-8

end do

!Calculate source term

Call ReactionSource(index,C,source)
r = source

!Partial derivatives

do i=1,n

  CNew(i) = C(i)

end do

do i=1,n

  CNew(i) = CNew(i) + dC(i) !dC(index)
  Call ReactionSource(index,Cnew,source)
  rnew = source

  deriv(i) = (1/dC(i))*(rnew-r)
  CNew(i) = C(i)

end do

end subroutine derive

!*****************************************************************************************



!*****************************************************************************************

subroutine Integrate_int(Area,Conc,dx,Ne)

!*****************************************************************************************
!
! Integrates the concentrations along the interface
!
! By Stelios Rigopoulos
!
!*****************************************************************************************

implicit none
integer, intent(in)                :: Ne
real(8), intent(in)                :: dx
real(8), intent(out)               :: Area
real(8), dimension(Ne), intent(in) :: Conc
integer                            :: i
```

```
!-----------------------------------------------------------------------------------------

Area = 0.0

do i=1,Ne

  Area = Area+Conc(i)*dx

end do

end subroutine Integrate_int

!*****************************************************************************************



!*****************************************************************************************

subroutine Thomas(X,Coeff,constant,Ne)

!*****************************************************************************************
!
! Solution of tridiagonal matrices using the Thomas algorithm
!
! By Stelios Rigopoulos
!
!*****************************************************************************************

implicit none

integer, intent(in)                    :: Ne

real(8), dimension(Ne), intent(in)     :: constant
real(8), dimension(Ne,Ne), intent(in)  :: Coeff
real(8), dimension(Ne), intent(out)    :: X

integer                                :: i
real(8), dimension(Ne)                 :: P,Q

!-----------------------------------------------------------------------------------------

!Forward substitution
P(1) = Coeff(1,2)/Coeff(1,1)
Q(1) = constant(1)/Coeff(1,1)

do i=2,(Ne-1)

  P(i) = Coeff(i,i+1)/(Coeff(i,i)-Coeff(i,i-1)*P(i-1))
  Q(i) = (constant(i)-Coeff(i,i-1)*Q(i-1))/(Coeff(i,i)-Coeff(i,i-1)*P(i-1))

end do

P(Ne) = 0
Q(Ne) = (constant(Ne)-Coeff(Ne,Ne-1)*Q(Ne-1))/(Coeff(Ne,Ne)-Coeff(Ne,Ne-1)*P(Ne-1))

!Backward substitution
X(Ne) = Q(Ne)

do i=(Ne-1),1,-1

  X(i) = Q(i)-P(i)*X(i+1)

end do

end subroutine Thomas

!*****************************************************************************************
```

C.4 Other programs

```
!*****************************************************************************************
!
!                          FILE Bulk_module.f90
!
! Integration of the differential equations describing the bulk of a fully mixed cell with
! possible input and output, where chemical reactions and particulate processes take place
! This file contains:
! - Module integrator_module: general declarations
! - Subroutine integrator_initdata: initialise values
! - Subroutine RHSide: calculates the right hand side of the ODEs to be integrated
! - Subroutine Bulk: Performs the temporal integration
! - Subroutine Moments: Calculates the moments of the distribution
!
! By Stelios Rigopoulos
! 11/01/2002
! 18/01/2002: first version of modular program
! 03/02/2002: coupling of PSD and species' concentrations
! 31/03/2002: modification of vol_growthrate for scaled variables
!
!*****************************************************************************************


!*****************************************************************************************

module integrator_module

implicit none

integer :: Nt,tratio
integer :: flowflag, initflag, inputflag
integer :: interval
real(8) :: tau,tres
integer :: Itmax_coupling,coupling
real(8) :: err_coupling
real(8) :: relerr_n,abserr_n,relerr_C,abserr_C,machineprec
integer :: N_of_Int,maxno_of_steps
integer :: method

real(8) :: Total_Par

!-----------------------------------------------------------------------------------------

end module integrator_module

!*****************************************************************************************


!*****************************************************************************************

subroutine integrator_initdata

!*****************************************************************************************
!
! Reads integration-related data
! By Stelios Rigopoulos
! 14/02/2002
!
!*****************************************************************************************

use integrator_module

implicit none
```

```
integer i

do i=1,6

  read(61,*)

end do

read(61,5001) tau
read(61,5001) tres
read(61,5002) flowflag
read(61,5001) relerr_n
read(61,5001) abserr_n
read(61,5001) relerr_C
read(61,5001) abserr_C
read(61,5002) maxno_of_steps
read(61,5001) machineprec
read(61,5002) method

5001 format(29X,F20.10)
5002 format(29X,I10)

end subroutine integrator_initdata

!*****************************************************************************************



!*****************************************************************************************



!*****************************************************************************************

Subroutine F (neqn,t,y,yprime)

!*****************************************************************************************
!
! Calculates the right hand side of the ODEs to be integrated
!
! By Stelios Rigopoulos
! 14/01/2002
!
!*****************************************************************************************

use integrator_module
use PBE_module
use reaction_module

implicit none

integer, intent(in)                        :: neqn
real(8), intent(in)                        :: t

real(8), dimension(neqn), intent(in)       :: y
real(8), dimension(neqn), intent(out)      :: yprime

real(8)                                    :: reaction_source

integer i,index

real(8) :: weight_in,weight_out

real(8), dimension(6)                      :: Cin

common weight_in,weight_out,Cin

!-----------------------------------------------------------------------------------------
```

```
!Write current time

!write (*,*) t

!if the variable is a species' concentration

  do index=1,neqn

    Call ReactionSource(index,y,reaction_source)

    yprime(index)   = + reaction_source + weight_in*Cin(index) - weight_out*y(index)

  end do

end subroutine F

!*****************************************************************************************



!*****************************************************************************************

Subroutine Bulk(ni,nin,C,C_in,w_in,w_out,integration_time,&
               &moment,vol_growthrate)

!*****************************************************************************************
!
! Directs the integration of ODEs
! for the Shampine and Gordon ODE/DE/STEP integrator
!
! By Stelios Rigopoulos
! 28/01/2002
! 03/02/2002: Coupling algorithm
!
!*****************************************************************************************

use PBE_module
use integrator_module
use reaction_module

implicit none

real(8), dimension(0:m), intent(inout)   :: ni
real(8), dimension(0:m), intent(in)      :: nin

real(8), dimension(n), intent(inout)     :: C
real(8), dimension(n), intent(in)        :: C_in

real(8), intent(in)                      :: integration_time

real(8), intent(in)                      :: w_in,w_out

real(8), intent(inout)                   :: vol_growthrate

real(8), dimension(0:1), intent(inout)   :: moment

real(8), dimension(0:m)                  :: y

real(8), dimension(n)                    :: C_temp,C_prev,C_ave
real(8), dimension(m)                    :: ni_prev

real(8)                                  :: dt
real(8)                                  :: Vmean,Dmean,Moment1prev
integer                                  :: t,i,j, index

real(8)                                  :: count
```

```
real(8)                                 :: pH, agg_degree

real(8)                                 :: tin,tout

real(8)                                 :: weight_in,weight_out
real(8), dimension(6)                   :: Cin

real(8)                                 :: rtol,atol
integer                                 :: neq,itol,itask,istate,iopt,lrw,liw,mf

!++++++++++ For Shampine and Gordon ODE +++++++++++++

external f
integer                                 :: iflag
real(8), dimension (100+21*(n))         :: work
integer, dimension(5)                   :: iwork

common weight_in,weight_out,Cin

!-----------------------------------------------------------------------------------------

do i=1,n

  Cin(i) = C_in(i)

end do

weight_in = w_in
weight_out = w_out

!-----------------------------------------------------------------------------------------

!Integration of concentrations

tin = 0.
tout = integration_time

iflag = 1

  do

    call ode(f,n,C,tin,tout,relerr_C,abserr_C,iflag,work,iwork)

    if (iflag==6) then

      write(*,*) 'warning - an error with the temporal integration &
      &of the population density has occurred, time is ',tin,' iflag is', iflag
      write(200,*) 'warning - an error with the temporal integration of&
      & the concentrations has occurred, time is ',tin,' iflag is', iflag
      stop

    else if (iflag==2) then

      exit

    else

      write(*,*) 'continuing integration, flag is ',iflag, 't is ', tin

    end if

 end do

!-----------------------------------------------------------------------------------------

1190 format ('Number: ',E12.6,'   Volume: ',E12.6,'   Mean size: ', E12.6)
1195 format (F10.4, 3X, E12.6, 3X, E12.6, 3X, E12.6, 3X, E12.6, 3X, E12.6, 3X)
1198 format (F10.4, 3X, <n>E20.6)
```

```
1031 format (F6.2,A15,F9.2,A19)

end subroutine Bulk

!*****************************************************************************************



!*****************************************************************************************

Subroutine moments(ni,moment,agg_degree)

use PBE_module
use integrator_module
use reaction_module

implicit none

real(8), dimension(0:m), intent(in)    :: ni

real(8), dimension(0:1), intent(inout) :: moment
real(8), intent(inout)                 :: agg_degree

integer                                :: i
real(8)                                :: vm,nim

!-----------------------------------------------------------------------------------------

Moment(0) = 0.0
Moment(1) = 0.0

do i=1,m

  !Zeroth moment is calculated by trapezoid rule
  Moment(0) = Moment(0) + 5.D-1*(ni(i-1)+ni(i))*dv(i)

  !Interpolate mid-point
  Vm = (v(i-1)+v(i))*5.D-1
  Nim = ni(i-1)*v(i)/dv(i) - ni(i-1)*Vm/dv(i)  +ni(i)*Vm/dv(i) - ni(i)*v(i-1)/dv(i)

  !First and second moments are calculated by Simpson's rule
  Moment(1) = Moment(1) + dv(i)*((1./6.)*ni(i-1)*v(i-1) + (4./6.)*Nim*Vm + (1./6.)*ni(i)*v(i))

  Agg_degree = (Total_Par - Moment(0))/Total_Par

end do

end subroutine moments

!*****************************************************************************************



!*****************************************************************************************
!
!                        FILE utilities.f90
!
! General-purpose subroutines
! This file contains:
! - Subroutine Growth_coeff: calculates growth coefficients as functions of supersaturation
! - Subroutine Nucleation_coeff: calculates nucleation coefficients as functions of supersaturation
! - Subroutine SS: calculates supersaturation
! - Function pH: calculates pH
!
! By Stelios Rigopoulos
! 11/01/2002
! 18/01/2002: first version of modular program
```

```
!
!*****************************************************************************************


!*****************************************************************************************

subroutine Growth_coeff(C)

!Calculates growth coefficients
!This implementation is for:
!- Volume-dependent growth due to size-independent linear growth
!- Power law function
!Set Growth_kernel_type to 1 in PBE_source

use PBE_module
use reaction_module

implicit none

real(8), dimension(n), intent(in)        :: C

!real(8), intent(out) :: G_coeff1,G_Coeff2,G_coeff3

real(8)              :: SS

!-----------------------------------------------------------------------------------------

Call SuperSaturation(C,SS)

if (abs(SS-0.)>threshold) then

  !Growth rate constant, as function of supersaturation
  G_coeff1 = 3.*(kv**(1./3.)) * G_ss_coeff * (SS**G_ss_exp)

else

  G_coeff1 = 0.

end if

!Volume exponent, due to size-independent linear growth
G_coeff2 = 2./3.

G_coeff3 = 0.
!g_coeff1 = 1.
end subroutine Growth_coeff

!*****************************************************************************************


!*****************************************************************************************


!*****************************************************************************************

subroutine Nucleation_rate_add(C,mass)

!Calculates nucleation rate
!This implementation is for:
!- Power law function

use PBE_module
use reaction_module

implicit none
```

```
real(8), dimension(n), intent(in)        :: C
!real(8), intent(out) :: Nuc
real(8), intent(in) :: mass
real(8)              :: SS

!-----------------------------------------------------------------------------------------

Call SuperSaturation(C,SS)

if (abs(SS-0.)>threshold) then

!  Nuc  = 5.D6*(SS**6.) +1.D13*(SS**0.1)*mass**1.   !0.66
!  Nuc  = 1.D6*(SS**4.) +4.D12*(SS**0.1)*mass**1.

!  Nuc  = 1.D6*(SS**4.) + 5.D12*(SS**0.1)*mass**1.
!  Nuc  = 5.D7*(SS**4.) + 6.D11*(SS**0.1)*mass**1.
!  Nuc  = 1.D7*(SS**4.) + 1.D14*(SS**0.1)*mass**1.
!  Nuc  = 1.D8*(SS**4.) + 1.D15*(SS**0.1)*mass**1.
!  Nuc  = 2.D7*(SS**4.) + 1.D15*(SS**0.1)*mass**1.

!For shear kernel - last simulation
!  Nuc  = 1.D6*(SS**4.) + 1.D14*(SS**0.1)*mass**1.

!  Nuc  = 1.D6*(SS**6.) + 4.D13*(SS**0.1)*mass**1.



  Nuc  = 7.D5*(SS**6.) + 1.6D14*(SS**0.1)*mass**1.

else

  Nuc = 0.

end if

end subroutine Nucleation_rate_add

!*****************************************************************************************


!*****************************************************************************************

Subroutine SuperSaturation(C,SS)

!Calculates supersaturation

use reaction_module

implicit none

real(8), dimension(n), intent(in)        :: C
real(8), intent(out)              :: SS

real(8) iota, alpha3, alpha5, C3_cor, C5_cor

!-----------------------------------------------------------------------------------------

if ((C(3)<=0.0).or.(C(5)<=0.0)) then

  SS = 0.0

else


!Ionic strength
```

```
Iota = 0.5*( C(5)/1000*2**2. + C(3)/1000*2**2. + C(4)/1000 + C(2)/1000 )

alpha3 = exp (  -0.5085 * 2.**2. * ( (sqrt(iota)/(1+sqrt(iota)))-0.3*iota)  )
C3_cor = alpha3 * C(3)


alpha5 = exp (  -0.5085 * 2.**2. * ( (sqrt(iota)/(1+sqrt(iota)))-0.3*iota)  )
C5_cor = alpha5 * C(5)


!SS = sqrt(C(3)*C(5))-sqrt(Ksp)
SS = sqrt(C3_cor*C5_cor/Ksp)-1.

end if

if (SS<0.0) then

  SS = 0.0

end if


end subroutine SuperSaturation

!*****************************************************************************************



!*****************************************************************************************


real(8) function pH(BB)

!Calculates pH

implicit none
real(8)  :: BB

!-----------------------------------------------------------------------------------------

if (BB>0.0) then

  pH = 14+log10(BB*0.001)

else

  pH = 0.0

end if

end function

!*****************************************************************************************



!*****************************************************************************************

subroutine Aggregation_rate(C,agg)

!Calculates aggregation rate
!This implementation is for:
!- Dirac function of pH dependence

use PBE_module
use reaction_module
```

```
implicit none

real(8), dimension(n), intent(in)        :: C
real(8), intent(out) :: agg
real(8)              :: SS

!-----------------------------------------------------------------------------------------

Call SuperSaturation(C,SS)

if (abs(SS-0.)>threshold) then

  agg = aggl_kernel_const * SS**2.

else

  agg = 0.

end if

end subroutine Aggregation_rate

!*****************************************************************************************



!*****************************************************************************************

subroutine Breakage_rate(C,bre)

!Calculates breakage rate
!This implementation is for:
!- Dirac function of pH dependence

use PBE_module
use reaction_module

implicit none

real(8), dimension(n), intent(in)        :: C
real(8), intent(out) :: bre

!-----------------------------------------------------------------------------------------

write(*,*)
bre = br_kernel_const

!if (C(2)<0.001) then

!  bre = br_const

!else

!  bre = 0.

!end if

end subroutine Breakage_rate

!*****************************************************************************************



!*****************************************************************************************

subroutine Henry_const(C,Henry)

!Calculates Henry's constant
```

```
use reaction_module

implicit none

real(8), dimension(n), intent(in)        :: C
real(8), intent(out)        :: Henry

real(8) :: Hw,h_CaOH2,h_CaHCO32,h_CaCO3

!-----------------------------------------------------------------------------------------

Hw = 10**(9.1229-0.059004*293+0.000078857*293**2)

!CaOH2
h_CaOH2  = (0.000053 + 0.000066 - 0.000015) * (0.5*(C(2)**2.+C(2)))

!CaHCO32
h_CaHCO32 = (0.000053 + 0.000021 - 0.000015) * (0.5*(C(3)**2.+C(3)))

!CaCO3
h_CaCO3  = (0.000053 + 0.000021 - 0.000015) * (0.5*(C(4)**2.+C(4)**2.))

Henry = 10.**(-(h_CaOH2+h_CaHCO32+h_CaCO3))*Hw
Henry = 1./Henry*100.
end subroutine Henry_const

!*****************************************************************************************



!*****************************************************************************************

Subroutine file_open(file_flag)

!Opens and closes the necessary files

implicit none

integer, intent(in) :: file_flag

!-----------------------------------------------------------------------------------------

if (file_flag==1) then

  open(31,file='input/inputPBE.in')
  open(34,file='input/initdis.in')
  open(35,file='input/inputdis.in')
  open(41,file='input/inputreng.in')
  open(43,file='input/initconc.in')
  open(44,file='input/inputconc.in')
  open(61,file='input/integration.in')
  open(71,file='input/interface.in')

  open(101,file='output/moments.out')
  open(103,file='output/species.out')
  open(105,file='output/PSD.out')
  open(106,file='output/PSDlast.out')
  open(111,file='output/initdis.out')
  open(113,file='output/inputdis.out')
  open(115,file='output/interface.out')
  open(119,file='output/nucleation.out')

  open(200,file='output/error.log')

else

  close(31)
```

```
  close(34)
  close(35)
  close(41)
  close(43)
  close(44)
  close(61)
  close(71)
  close(101)
  close(103)
  close(105)
  close(111)
  close(113)
  close(115)
  close(200)

end if

end subroutine file_open

!*********************************************************************************************

!*********************************************************************************************
!
!                          FILE reaction_module.f90
!
! Chemical reaction engineering subroutines
! This file contains:
! - Module reaction_module: all general declarations
! - Subroutine reaction_source: source terms in species' balances due to reactions
! - Subroutine reaction_initdata: read values of reaction parameters
! - Subroutine reaction_initarrays: initialise concentrations
!
! By Stelios Rigopoulos
! 11/01/2002
! 18/01/2002: first version of modular program
!
!*********************************************************************************************



!*********************************************************************************************

module reaction_module

implicit none

integer :: n

real(8) k1,k2,Keq1,Keq2,rho,MW,Ksp,kv

!*********************************************************************************************

end module reaction_module

!*********************************************************************************************



!*********************************************************************************************

Subroutine ReactionSource(index,C,reaction_source)

!*********************************************************************************************
!
! Reaction kinetics
!
! This implementation is for:
! CaCO3 precipitation
```

```
!
! By Stelios Rigopoulos (14/01/2002)
!
!*****************************************************************************************

use reaction_module
use PBE_module !for v_nuc

implicit none

integer,intent(in)                :: index

real(8), dimension(n), intent(in) :: C

!real(8), intent(in)               :: vol_growthrate

real(8), intent(out)              :: reaction_source

real(8)                           :: Growth
real(8)                           :: r1,r2,r3

real(8) :: temp

!-----------------------------------------------------------------------------------------

!Define kinetics

!Call Nucleation_rate(C)

r1 = k1*C(1)*C(2) - (k1/Keq1)*C(4)
r2 = k2*C(4)*C(2) - (k2/Keq2)*C(3)
!r3 = rho*(1/MW)*vol_growthrate !+ rho*(1/MW)*v_nuc*Nuc

!-----------------------------------------------------------------------------------------

!Define non-linear source terms

if (index==1) then

  reaction_source = -r1

else if (index==2) then

  reaction_source = -r1 -r2

else if (index==3) then

  reaction_source = r2 !-r3

else if (index==4) then

  reaction_source = r1 -r2

else if (index==5) then

  reaction_source = 0. !-r3

end if

temp = 0.0

end subroutine ReactionSource

!*****************************************************************************************


!*****************************************************************************************
```

```
subroutine reaction_initdata

!*****************************************************************************************
!
! Reads kinetic data
! By Stelios Rigopoulos
! 09/02/2002
!
!*****************************************************************************************

use reaction_module

implicit none

integer i

!-----------------------------------------------------------------------------------------

do i=1,9

  read(41,*)

end do

read(41,5002) n

!Read kinetic data from file

read(41,5001) k1
read(41,5001) k2
read(41,5001) Keq1
read(41,5001) Keq2
read(41,5001) rho
read(41,5001) MW
read(41,5001) Ksp
read(41,5001) Kv

!-----------------------------------------------------------------------------------------

5001 format(29X,F20.10)
5002 format(29X,I10)

end subroutine reaction_initdata

!*****************************************************************************************



!*****************************************************************************************

subroutine reaction_initarrays(C,Cin)

!*****************************************************************************************
!
! Initialises arrays
! By Stelios Rigopoulos
! 09/02/2002
!
!*****************************************************************************************

use reaction_module

implicit none

real(8), dimension(n), intent(inout) :: C,Cin

integer i
```

```
!-----------------------------------------------------------------------------------------

do i=1,n

  Cin(i) = 0.
  C(i)= 0.

end do

!Initial distribution

rewind(43)

do i=1,6

  read(43,*)

end do

do i=1,n

  read(43,5001) C(i)

end do

!-----------------------------------------------------------------------------------------

!Input distribution

rewind(44)

do i=1,6

  read(44,*)

end do

do i=1,n

  read(44,5001) Cin(i)

end do

!-----------------------------------------------------------------------------------------

5001 format(29X,F20.10)
5002 format(29X,I10)

end subroutine reaction_initarrays

!*****************************************************************************************

program main

call network()

end program main
```

Bibliography

1. Astarita, G. (1967). *Mass transfer with chemical reaction*, Elsevier Publishing.

2. Baldyga, J., Bourne, J.R., & Hearn, S.J. (1997). Interaction between chemical reactions and mixing on various scales, *Chemical Engineering Science*, 52, 457-466.

3. Batterham, R.J., Hall, J.S., and Barton, G. (1981). Pelletizing kinetics and simulation of full-scale balling circuits, *Proc. 3rd Int. Symp. on Agglomeration*, Nurnberg, A136.

4. Bauer M. and Eigenberger G. (1999). A concept for multi-scale modelling of bubble columns and loop reactors, *Chemical Engineering Science*, Vol. 54, 5109-5117.

5. Beckman, J.R., and Farmer, R.W. (1984). Bimodal CSD barite due to agglomeration in an MSMPR crystalliser, *AIChE symposium series*, vol.80, no.240, 85-93.

6. Bezzo, F., Macchietto, S., & Pantelides, C. C. (2000). A general framework for the integration of computational fluid dynamics and process simulation, *Computers and Chemical Engineering*, 24, 653-658.

7. Bird, R.B., Stewart, W.E, and Lightfoot, E.N. (1960). *Transport phenomena*, John Wiley and Sons.

8. Bleck, R. (1970). A fast, approximate method for integrating the stochastic coalescence equation, *J. of Geophysical Res.*, vol.75 no.27, 5165-5171.

9. Boisson, N. and Malin, M.R. (1996). Numerical prediction of two-phase flow in bubble columns, *International Journal for Numerical Methods in Fluids*, Vol. 23, 1289-1310.

10. Botsaris, G.D. (1976). Secondary nucleation: a review. *6th Symposium of Industrial Crystallisation*, J.W. Mullin (ed.), Plenum Press, New York, 3-22.

11. Bramley, A.S., Hounslow, M.J. and Ryall, R.L. (1996). Aggregation during precipitation from solution. A method for extracting rates from experimental data. *Journal of colloid and interface science*, 183 (1), 155-165.

12. Bramley, A.S., Hounslow, M.J., Newman, R., Paterson, W.R., and Pogessi, C. (1997). The role of solution composition on aggregation during precipitation, *TransIChemE*, vol. 75, part A, 119-124.

13. Calderbank, P.H., and Lochiel, A.D., (1964) Mass transfer coefficients, velocities and shapes of carbon dioxide bubbles in free rise through distilled water, *Chemical Engineering Science*, vol. 19, 485-503.

14. Carey, G. F. and Finlayson, B. A. (1975). Orthogonal collocation on finite elements, *Chemical Engineering Science*, vol. 30, no. 5-6, 587-596.

15. *CFX v. 4.3 electronic manual* (1999). AEA Technology, Harwell, UK

16. Chakraborty, D., and Bhatia, S.K. (1996). Formation and aggregation of polymorphs in continuous precipitation. 2. Kinetics of $CaCO_3$ precipitation, *Industrial Engineering and Chemistry Research*, 35, 1995-2006.

17. Chandrasekhar, S. (1945). Stochastic, statistical, and hydromagnetic problems in physics and astronomy, in *Selected papers of S. Chandrasekhar*, Volume 3, The University of Chicago press.

18. Chisti, M.Y., Halard, B., and Moo-Young, M. (1988). Liquid circulation in airlift reactors, *Chemical Engineering Science*, Vol. 43 No. 3, 451-457.

19. Clift, R., Grace , J.R., and Weber, M.E. (1978). *Bubbles, drops, and particles*, Academic Press.

20. Cockx A., Do-Quang, Z., Line A., and Roustan, M. (1999). Use of computational fluid dynamics for simulating hydrodynamics and mass transfer in industrial ozonation towers, *Chemical Engineering Science*, Vol. 54, 5085-5090.

21. Collier, A.P., and Hounslow, M.J. (1999). Growth and aggregation rates for calcite and calcium oxalate monohydrate, *AIChE J.*, vol.45, no.11, 2298-2305.

22. Danckwerts, P.V. (1953), Continuous systems: Distribution of residence times, *Chemical Engineering Science* 2, 1-13.

23. Danckwerts, P.V. (1958). The effect of incomplmete mixing on homogeneous reactions, *Chemical Engineering Science* 8, 93-100.

24. Danckwerts, P.V. and Sharma, M.M. (1966). The absorption of carbon dioxide into solutions of alkalis and amines, *The Chemical Engineer (Trans IchemE)*, CE244.

25. Dankwerts (1970) *Gas - liquid reactions*, McGraw-Hill.

26. David, R., Marchal, P., Klein, J.-P., and Villermauxx, J. (1991). Crystallization and precipitation engineering III. A discrete formulation of the agglomeration rate of crystals in a crystallization process, *Chemical Engineering Science*, vol. 46, 205-213.

27. Davies, C.W. (1962). *Ion Association*, Butterworths, London.

28. De Angelis, V., and Banerjee, S., (1988). Numerical investigation of mass transfer phenomena at a gas-liquid interface, *Third International Conference on Multiphase Flow (ICMF'98).*

29. Deckwer, W.D. (1992). *Bubble column reactors*, John Wiley and Sons.

30. Deckwer, W.D., Hallensleben, J., & Popovic, M. (1980). Exclusion of gas sparger influence on mass transfer in bubble columns, *Canadian Journal of Chemical Engineering*, 58, 190-197.

31. Delnoij E., Kuipers J.A.M. and van Swaaij W.P.M. (1997). Computational fluid dynamics applied to gas-liquid contactors, *Chemical Engineering Science*, Vol. 52, 3623-3638.

32. Derjaguin, B.V., and Landau, L. (1941). Theory of the stability of strongly charged lyophobic sols and of the adhesion of strongly charged particles in solutions of electrolytes, *Acta Phtsicochim. URSS*, 14:633-662.

33. Doraiswamy, L.K., and Sharma, M.M. (1984). *Heterogeneous reactions: Analysis, examples, and reactor design*, John Wiley and sons.

34. Drake, R.L. (1972). A general mathematical survey of the coagulation equation, *in Topics in Current Aerosol Research*, vol. 3, pt. 2, Hidy, G.M., and Brock, J.R. (eds), Pergamon Press, NY, 202-end.

35. Drew, D.A., (1983), Mathematical modelling of two-phase flow, *Ann. Rev. Fluid Mech.*, vol.15, 261-291.

36. Drew, D.A., and Passman, S.L. (1998). *Theory of multicomponent fluids*, Springer.

37. Dudukovic, M.P., Larachi, F., & Mills, P.L. (1999). Multiphase reactors - revisited, *Chemical Engineering Science*, 54, 1975-1995.

38. Eyre, d., Wright, C.J., and Reuter, G. (1988). Spline-Collocation with adaptive mesh grading for solving the stochastic collection equation, *Journal of Computational Physics*, 78, p. 288-304.

39. Fleischer C., Becker S. and Eigenberger G. (1995). Transient hydrodynamics, mass transfer, and reaction in bubble columns: CO_2 absorption into NaOH solutions', *TransIChemE*, vol.73 part A, 649-653.

40. Fleischer C., Becker S. and Eigenberger G. (1996). Detailed modelling of the chemisorption of CO_2 into NaOH in a bubble column, *Chemical Engineering Science*, Vol. 51, No 10, 1715-1724.

41. Fletcher, C.A.J. (1990). *Computational techniques for fluid dynamics*, Springer-Verlag.

42. Garside, J., and Davey, R.J. (1980). Secondary contact nucleation: kinetics, growth and scale-up, *Chem, Eng. Commun*, vol.4, 393-424.

43. Geary, N.W., and Rice, R.G. (1991). Bubble size prediction for rigid and flexible spargers, *AIChE Journal*, vol. 37, no. 2, 161-168.

44. Gelbard, F., and Seinfeld, J.H. (1978). Numerical solution of the dynamic equation for particulate systems, *Journal of computational physics*, vol. 28, 357-375.

45. Gelbard, F., Tambour, Y., and Seinfeld, John H. (1980). Sectional Representations for simulating aerosol dynamics, *Journal of Colloid and Interface Science*, vol. 76, no. 2.

46. Glasscock, D.A. and Rochelle, G.T. (1989). Numerical simulation of theories for gas absorption with chemical reaction, *AIChE Journal*, Vol. 35 No. 8, 1271-1281.

47. Hartel, R.W., Gottung, B.E., Randolph, A.D., and Drach, G.W. (1986). Mechanisms and kinetic modelling of calcium oxalate crystal aggregation in a urine-like liquor, Part I: Mechanisms. *AIChE Journal*, vol. 32, no. 7, 1176-1185.

48. Hartel, R.W., and Randolph, A.D. (1986). Mechanisms and kinetic modelling of calcium oxalate crystal aggregation in a urinelike liquor, Part II: Kinetic modelling. *AIChE Journal*, vol. 32, no. 7, 1186-1195.

49. Higbie, R. (1935). The rate of absorption of a pure gas into a still liquid during a short time of exposure, *Trans. of AIChE*, 31,365-389.

50. Hill, Priscilla J. and Ng, Ka M. (1995). New discretization procedure for the breakage equation, *AIChE Journal*, vol. 41 no. 5, p. 1204-1216.

51. Hindmarsh, A.C. (1983). Odepack, a systematized collection of ode solvers, *Scientific Computing*, R. S. Stepleman et al. (eds.), North-Holland, Amsterdam, pp. 55-64.

52. Hirt, C.W. and Nichols, B.D. (1981). A computational method for free surface hydrodynamics, *ASME Transactions*, Vol. 103, 136-141.

53. Hostomský, J. (1987). Particle size distribution of agglomerated crystal product from a continuous crystallizer, *Collection Czechoslovak Chem. Commun.*, vol. 52, 1186-1197.

54. Hostomsky, J. and Jones A.G., (1995). A penetration model of the gas-liquid reactive precipitation of calcium carbonate crystals, *Trans. IchemE*, 241-245.

55. Hounslow, M.J. (1990). A discretized population balance for continuous systems at steady state, *AIChE Journal*, vol. 36, no. 1, 106-116.

56. Hounslow, M.J., Ryall, R.L., and Marshall, V.R. (1988). A discretized population balance for nucleation, growth, and aggregation, AIChE J., vol. 34, no. 11, 1821-1832.

57. Hounslow, M. J.,Mumtaz, H. S., Collier, A. P., Barrick, J. P. and Bramley, A. S. (2000). A micro-mechanical model for the rate of aggregation during precipitation from solution, *Chem. Eng. Sci.*, vol.56, 2543-2552.

58. House, W. (1980). Kinetics of crystallisation of calcite from calcium bicarbonate solutions, *J. Chem. Soc. Faraday Trans.*, vol.77, 341-359.

59. Hulburt, H.M. and Katz, S. (1964), Some problems in particle technology: a statistical mechanical formulation, *Chemical Engineering Science*, vol. 19, 572-594.

60. Ishii, M. and Zuber N. (1979). Drag Coefficient and relative velocity in bubbly, droplet or particulate flows, *AIChE Jouirnal*, Vol. 25, No. 5, 843-855.

61. Ishii, M. (1975). *Thermo-fluid dynamic theory of multi-phase flow*, Eyrolles.

62. Jakobsen H.A., Sannaes B.H., Grevskott S. and Svndsen H.F. (1997). Modelling of vertical bubble-driven flows, *Industrial and Engineering Chemistry Research*, Vol. 36, 4052-4074.

63. Jakobsen, H.A. (2000). Phase distribution phenomena in two-phase bubble column reactors, *Chemical Engineering Science*, vol. 56, 1049-1056.

64. Jones, A.G., (1985). Liquid circulation in a draft-tube bubble column, *Chemical Engineering Science*, Volume 40, Issue 3, 1985, Pages 449-462

65. Jones, A.G. (2001). *Crystallization process systems*, Butterworth-Heinemann.

66. Jones, A.G., Hostomsky, J., and Wachi, S. (1996). Modelling and analysis of particle formation during agglomerative crystal precipitation processes, *Chemical Engineering Communications*, vol. 146, 105-130.

67. Jones, A.G., Wachi, S. and Delannoy, C.C. (1992). Precipitation of calcium carbonate in a fluidised bed reactor, *Proceedings of the Seventh Engineering Foundation Conference on Fluidization*, 407-414.

68. Joshi,J.B., Ranade, V.V., Gharat, S.D., and Lele, S.S. (1990). Sparged loop reactors, *Canadian Journal of chemical engineering*, vol.68, 705-741.

69. Joshi, J.B. (2001). Computational flow modelling and design of bubble column reactors, *Chemical Engineering Science* 56, 5893-5933.

70. Juvekar, V.A. and Sharma, M.M. (1973). Absorption of CO_2 in a suspension of lime, *Chemical Engineering Science*, vol. 28, 825-837.

71. Kastanek, J., Zahradnik, J., Kratochvil, J., Cermak, J. (1993). *Chemical reactors for gas-liquid systems*, Ellis Horwood Series in Chemical Engineering.

72. King, C.J. (1966). Turbulent liquid phase mass transfer at a free gas-liquid interface, *Industrial and Engineering Chemistry Fundamentals*, 59(1), 1.

73. Kocamustafaogullari, G., and Ishii, M. (1995). Foundation of the interfacial area transport equation and its closure relations, *International Journal of Heat and Mass Transfer*, vol. 38, No.3, 481-493.

74. Koide, K., Iwamoto, S., Takasaka, Y., Matsuura, S., Takahashi, E., Kimura, M., and Kubota, H. (1984). Liquid circulation, gas holdup and pressure drop in bubble column with draught tube, *Journal of chemical Engineering of Japan*, vol 17, no.6, 611-618.

75. Kostoglou, M., and Karabelas, A.J. (1994) Evaluation of aero order methods for simulating particle coagulation, *Journal of Colloid and Interface Science*, vol. 163, p. 420-431.

76. Kotaki, Y., and Tsuge, H. (1990). Reactive crystallization of calcium carbonate by gas-liquid and liquid-liquid reactions, *Canadian Journal of Chemical Engineering*, 68, 435-442.

77. Kuipers, J.A.M., and van Swaaij, W.P.M. (1997). Application of computational fluid dynamics to chemical reaction engineering, *Reviews in Chemical Engineering*, 13, 1-118.

78. Kumar, S., and Ramkrishna, D. (1996a). On the solution of population balance equations by discretization - I. A fixed pivot technique, *Chemical Engineering Science*, vol. 51, no.8, 1311-1332.

79. Kumar, S., and Ramkrishna, D. (1996b). On the solution of population balance equations by discretization - II. A moving pivot technique, *Chemical Engineering Science*, vol. 51, no. 8, 1333-1342.

80. Kumar, S., and Ramkrishna, D. (1996c). On the solution of population balance equations by discretization - III. Nucleation, grpwth and aggregation of particles, *Chemical Engineering Science*, vol. 52, no.24, 4659-4679.

81. Lappin, A., and Lübbert, A. (1994). Numerical simulation of the dynamics of two-phase gas liquid flows in bubble columns, *Chemical Engineering Science*, Volume 49, Issue 21, 1994, 3661-3674.

82. Landau, L.D., and Lifshitz, E.M. (1959). *Fluid Mechanics*, Butterworth-Heinemann.

83. Larson, M.A. (1984). Advances in the characterisation of crystal nucleation, *AIChE symposium series*, vol.80, no.240, 39-44.

84. Lee, M.H. (2001). A survey of numerical solutions to the coagulation equation", *Journal of Physics A: Mathematical and General*, 34, 10219-10241.

85. Liao, Y., and McLaughlin, J.B., (2000). Dissolution of a freely rising bubble in aqueous surfactant solution, *Chemical Engineering Science*, vol. 55, 5831-5850.

86. Litster, J., Smit, D.J., and Hounslow, M.J. (1995). Adjustable discretised population balance for growth and aggregation, *AIChE Journal*, vol.41, p. 591-603.

87. Liu, Y. amd Cameron, I.T., (2001). A new wavelet-based method for the solution of the population equation, *Chemical Engineering Science*, vol. 56, 5283-5294.

88. Lochiel A.C., and Calderbank, P.H. (1964). Mass transfer in the continuous phase around axisymmetric bodies of revolution, *Chemical Engineering Science*, vol. 19, 471-484.

89. Luo, H. and Svendsen, H.F. (1996). Turbulent model for drop and bubble breakup in turbulent dispersions, *AIChE Journal*, vol. 42, No. 5, 1225-1233.

90. Mann, R., and Hackett, L.A. (1988). Fundamentals of gas-liquid mixing in a stirred vessel: An analysis using networks of backmixed zones, *Proceedings of the 6th European conference on mixing*, BHRA, Pavia, 321.

91. Marchal, P., David, R., Klein, J.P., and Villermaux, J. (1988). Crystallization and precipitation engineering - I. An efficient method for solving population balance in crystallization with agglomeration, *Chemical Engineering Science*, vol.43, no.1, p. 59-67.

92. Mecklenburgh, J.C, & Hartland, S. (1975). *Theory of backmixing*, Wiley-Interscience, New York.

93. Melis, S., Verduyn, M., Storti, G., Morbidelli, M., and Baldygan J. (1999). Effect of fluid motion on the aggregation of small particles subject to interaction forces, *AIChE Journal*, vol.45, no.7, 1383-1393.

94. Mersmann, A. (2001). *Crystallization technology handbook*, Marcel Dekker.

95. Mersmann, A., Sangl, R., Kind, M., And Pohlisch, J. (1988). Attrition and secondary nucleation in crystallisers, Chemical engineering and technology, 11, 80-88.

96. Montante, G.,Lee, K. C., Brucato, A., and Yianneskis, M. (2001). Experiments and predictions of the transition of the flow pattern with impeller clearance in stirred tanks, *Computers and Chemical Engineering*, 25, 729-735.

97. Mullin, J.W. (2001). *Crystallisation*, Butterworth-Heinemann.

98. Mumtaz, H.S., Hounslow, M.J., Seaton, N.A., and Paterson, W.R. (1997). Orthokinetic aggregation during precipitation: A computational model for calcium oxalate monohydrate, *Trans IChemE*, vol.75, 151-159.

99. Mumtaz, H. S. and M. J. Hounslow, M.J. (2000). Aggregation during precipitation from solution: an experimental investigation using Poiseuille flow, *Chemical Engineering Science*, Vol. 55, no.23, p. 5671-5681.

100. Nauman, E.B. and Buffham, B.A. (1983). *Mixing in continuous flow systems*, John Wiley and Sons.

101. Nicmanis, M., and Hounslow, M.J. (1998). Finite-element methods for steady-state population balance equations, *AIChE Journal*, vol. 44, no. 10, 2258-2271.

102. Niemansverdriet, J.W. (1993). *Spectroscopy in catalysis.*

103. Nývlt, J., Söhnel, Ot., Matuchová, M., and Broul, M. (1985). *The kinetics of industrial crystallization*, Elsevier.

104. Pakter, A. (1968). The precipitation of sparingly solublel alkaline-earth metal and lad salts: Nucleation and growth order during the induction period. *Inorg. Phys. Theor.*, 859,862.

105. Paschedag, A, Tornatora, M., Ausner, I., and Signorino, M. (2002). Modelling of agglomeration for $BaSO_4$ precipitation, *Proceedings of the 15th international symposium on industrial crystallisation*, Sorrento, Italy.

106. Patankar, S.V. (1980). *Numerical heat transfer and fluid flow*, Hemisphere Publishing.

107. Pfleger, D., Gomes, S., Gilbert, N., and Wagner, H.G., (1999). Hydrodynamic simulations of laboratory scale bubble columns : fundamental studies of the Eulerian-Eulerian modelling approach, *Chemical Engineering Science*, Vol.54, 5091-5099.

108. Pohorecki, R. and Moniuk, W. (1988). Kinetics of reaction between carbon dioxide and hydroxyl ions in aqueous electrolyte solutions, *Chemical Engineering Science*, 43, 1677-1684.

109. Pohorecki, R., and Baldyga, J. (1983). The use of a new model of micromixing for determination of crystal size in precipitation, *Chemical Engineering Science*, Volume 38, Issue 1, 1983, Pages 79-83.

110. Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. (1992). *Numerical Recipes in Fortran 77: The art of Scientific Computing*, Second edition, Cambridge University Press.

111. Ramabhadran, T.E., Peterson, T.W., and Seinfeld, J.H. (1976). Dynamics of aerosol coagulation and condensation, *AIChE Journal*, vol. 22, no. 5, 840-851.

112. Ramkrishna, D. (1985). The status of population balances, *Reviews in Chemical Engineering*, vol.3 no.1, 49-95.

113. Ramkrishna, D., and Borwanker, J.D. (1973). A puristic analysis of population balance - I, *Chemical Engineering Science*, 28, 1423-1435.

114. Ranade, V.V. (1995). Computational fluid dynamics for reactor engineering, *Reviews in Chemical Engineering*, Vol.11, no.3, 229-289.

115. Randolph, A.D., and Larson, M.A. (1988). *Theory of particulate processes*, Second edition, Academic Press.

116. Randolph, A.D., and White, E.T. (1977). Modelling size dispersion in the prediction of crystal size distribution, *Chemical Engineering Science*, vol.32, no.9, p. 1067-1076.

117. Rawlings, J.B., Miller, S.M., and Witkowski, W.R. (1993). Model identification and control of solution crystallization processes: a review, *Industrial and Engineering Chemistry Research*, vol. 32, 1275-1296.

118. Rigopoulos, S., and Jones, A.G. (2001). Dynamic modelling of a bubble column for particle formation via a gas-liquid reaction, *Chemical Engineering Science*, 56, 21-22, 6177-6183.

119. Rigopoulos, S., and Jones, A.G., in press. A finite element scheme for solution of the dynamic population balance, *AIChE Journal.*

120. Rojkowski, Z. (1990). Initial condition for population balance in an MSMPR crystallizer, *AIChE Journal*, 36, no.4, 630-632.

121. Romanainen, J.J. (1997). Numerical approach to modelling of dynamic bubble columns, *Chemical Engineering and Processing*, Vol. 36, 1-15.

122. Sanyal J., Vasquez S., Shantanu Roy and Dudukovic M.P. (1999). Numerical simulation of gas-liquid fluid dynamics in cylindrical bubble column reactors, *Chemical Engineering Science*, Vol. 54, 5109-5117.

123. Sato Y., Sadatomi, S., and Sekoguchi, K. (1981). Momenum and heat transfer in two-phase flow – Part I, theory, *International journal of multiphase flow*, 7, 167-177.

124. Schluter, S., Steiff, A., and Weinspach, P.M. (1992). Modelling and simulation of bubble column reactors, *Chemical Engineering and Processing*, Vol. 31, 97-117.

125. Schwarz, M.P. and Turner, W.J. (1988). Applicability of the standard k-ε turbulence model to gas-stirred baths, *Applied mathematical modelling*, Vol. 12, 273-279.

126. Scott, W.T. (1968). *Analytical studies of cloud droplet coalescence*, Journal of Atmospheric Sciences, 54-65.

127. Shah, Y.T., Kelkar, B.G. and Godbole, S.P. (1982). Design parameters estimations for bubble column reactors, *AIChE Journal*, Vol. 28 No.3, 353-378.

128. Shyy, W., Udaykumar, H.S., Rao, M.M., and Smith, R.W. (1997). *Computational fluid dynamics with moving boundaries*, Hemisphere publishing.

129. Smoluchowski, M.V. (1917). Mathematical theory of the kinetics of coagulation of colloidal systems, *Z.Phys.Chem.*, 92, 129-168.

130. Söhnel, O., and Garside, J. (1992). *Precipitation: basic principles and industrial applications*, Butterworth-Heinemann.

131. Sokolichin, A. and Eigenberger, G. (1994). Gas-liquid flow in bubble columns and loop reactors: Part I. Detailed modelling and numerical simulation, *Chemical Engineering Science*, Vol. 49, 5735-5746.

132. Sokolichin, A. and Eigenberger, G. (1999). Applicability of the standard k - ε turbulence model to the dynamic simulation of bubble columns: Part I. Detailed numerical simulations, *Chemical Engineering Science*, Volume 54, Issues 13-14, July 1999, Pages 2273-2284.

133. Spalding, D.B. (1980). Numerical computation of multi-phase fluid flow and heat transfer, in *Recent advances in numerical methods in fluids*, vol. 1, Pineridge Press, Swansea, 139-167

134. Steemson, M.L., and White, E.T. (1988). Numerical modelling of steady state continuous crystallization processes using piecewise cubic spline functions, *Computers and Chemical Engineering*, vol. 12 no.1, 81-89.

135. Stewart H.B. and Wendroff B. (1984). Two-phase flow: Models and methods, *Journal of Computational Physics*, Vol. 56, 363-409.

136. Swinney, L.D., Stevens, J.D. and Peters, R.W. (1982). Calcium carbonate crystallization kinetics, *Industrial and Engineering Chemistry Fundamentals.*, 21, 31-36.

137. Tai, C.Y. and Chen, P.-C. (1995). Nucleation, agglomeration and crystal morphology of calcium carbonate, *AIChE J.*, vol.41 no.1, 68-77.

138. Tan, R.B.H., and Harris, I. (1986). A model for non-spherical bubble growth at a single orifice, *Chemical Engineering Science*, Vol. 41, No. 12, 3175-3182.

139. Tavare, N.S. (1986). Mixing in continuous crystallisers, *AIChE Journal*, vol. 32 no.5, 705-732.

140. Thakre S.S., and Joshi J.B. (1999). CFD simulation of bubble column reactors: importance of drag force formulation, *Chemical Engineering Science*, Vol. 54, 5055-5060.

141. Toor, H.L. and Marchello, J.M. (1958). Film-penetration model for mass transfer and heat transfer, *AIChE Journal*, Vol.4 , 97-101.

142. Torvik, R., and Svendsen, H.F. (1990). Modelling of slurry reactors. A fundamental approach, *Chemical Engineering Science*, vol. 45, 2325-2332.

143. Van den Akker, H.E.A. (1997). On the status and merits of computational fluid dynamics, *4th international conference on bioreactor and bioprocess fluid dynamics*, edited by W.A. Nienow, 407-431.

144. Van Elk, E.P., Borman, P.C., Kuipers, J.A.M., and Versteeg, G.F., (2000). Modelling of gas-liquid reactors – implementation of the penetration model in dynamic modelling of gas-liquid processes with the presence of a liquid bulk, *Chemical Engineering Journal*, 76, 223-237

145. Van Elk, E.P., Borman, P.C., Kuipers, J.A.M., and Versteeg, G.F. (1999) Modelling of gas-liquid reactors stability and dynamic behaviour of gas-liquid mass transfer accompanied by irreversible reaction, *Chemical Engineering Science*, 76, 223-237.

146. Van Leeuven, M.L.J., Bruinsma, O.S.L., and Van Rosmalen, G.M. (1996). Influence of mixing on the product quality in precipitation, *Chemical Engineering Science*, vol.51 no. 11, 2595-2600.

147. Van Swaaij, W.P.M, and Versteeg, G.F., (1992). Mass transfer accompanied with complex chemical reactions in gas-liquid systems: an overview, *Chemical Engineering Science*, vol.47, 3181-3195.

148. Vanni, M. (2000). Approximate population balance equations for aggregation-breakage processes, *Journal of Colloid and Interface Science*, vol. 221, p. 143-160.

149. Versteeg, G.F., Kuipers, F.P., Van Beckum, F.P.H. and Van Swaaij, W.P.M. (1989). Mass transfer with complex reversible chemical reactions – I. Single reversible chemical reaction, *Chemical Engineering Science*, vol.44, no.10, 2295-2310.

150. Verwey, E.J.W., and Overbeek, J.T.G. (1948). *Theory of stability of lyophobic colloids*, Elsevier, Amsterdam.

151. Versteeg, G.F., Kuipers, F.P., Van Beckum, F.P.H. and Van Swaaij, W.P.M. (1990). Mass transfer with complex reversible chemical reactions – II. Parallel reversible chemical reaction, *Chemical Engineering Science*, vol.45, no.1, 183-197.

152. Wachi, S. and Jones, A.G. (1991). Aspects of gas-liquid reaction systems with precipitate solid formation, *Reviews in Chemical Engineering*, Vol. 11 No. 1, 1-51.

153. Wachi, S. and Jones, A.G. (1991). Mass transfer with chemical reaction and precipitation, *Chemical Engineering Science*, Vol. 46, No. 4, 1027-1033

154. Wachi, S., and Morikawa, H., (1986) Chlorination of ethylene in a boling bubble column reactor, *Journal of Chemical Engineering of Japan*, vol.20 no.3, 238-245.

155. Wachi, S., Jones, A.G., and Elson, T.P. (1991). Flow dynamics in a draft-tube bubble column using various liquids, *Chemical Engineering Science*, Vol.46 No. 2, 657-663.

156. Wallis, G.B., (1969). *One dimensional two-phase flow*, McGraw-Hill, New York.

157. Wei, H., and Garside, J. (1997). Application of CFD modelling to precipitation systems, *Trans. IChemE*, 75, Part A, 219.

158. Yagi, H., Iwazawa, A., Sonobe, R., Matsubara, T., and Hikita, H. (1984). Crystallization of calcium carbonate accompanying chemical absorption, *Ind. Eng., Chem., Fundam.*, 23, 153-156.

159. Yagi, H. (1986). Kinetics of solid production accompanying gas-liquid reaction, *Proc. World Congress III Chem. Eng., Tokyo,* vol.4, 20-23.

160. Jung, W.M., Kang, S.H., Kim, W., Chang, K.C. (2000). Particle morphology of calcium carbonate precipitated by gas-liquid reaction in a Couette-Taylor reactor, *Chem. Eng. Sci.*, vol. 55, 733-747.

161. Zahradnk, J., Mann, R., Fialová, M., Vlaev, D., Vlaev, S. D., Lossev, V. and Seichter, P. (2001). A networks-of-zones analysis of mixing and mass transfer in three industrial bioreactors, *Chemical Engineering Science*, Vol. 56, Issue 2, January 2001, Pages 485-492.

162. Zauner, R., and Jones, A.G. (2000). Scale-up of continuous and batch precipitation processes, *Industrial Engineering and Chemistry Research*, vol.39 no.7, 2392-2403.

163. Ziff, Robert M. (1991). New solutions to the fragmentation equation, *J. of Physics A Math. Gen.*, vol. 24, p. 2821-2828.

164. Ziff, Robert M. and McGrady, E.D. (1985). The kinetics of cluster fragmentation and depolymerisation, *J. of Physics A Math. Gen.*, vol. 18, p. 3027-3037.