A Coupled Stochastic-Deterministic Method for the Numerical Solution of Population Balance Systems

Dissertation zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

Eingereicht von
Clemens Bartsch

Berlin, April 2018

1. Gutachter: Prof. Dr. Volker John (*Betreuer*)
Freie Universität Berlin und
Weierstraß-Institut, Berlin

2. Gutachter: Prof. Dr. Sashikumaar Ganesan
Indian Institute of Science, Bangalore

Tag der Disputation: 28.08.2018

Für meine Großmütter, Dagmar und Veronika,
und meine Großväter, Ignaz und Gottfried.

Danksagung

Bedanken möchte ich mich bei allen, die zum Gelingen dieser Arbeit beigetragen und dafür gesorgt haben, dass mich nicht unterwegs der Mut verließ. Das ist zunächst Prof. Dr. Volker John, der mich geduldig, genau und inspirierend angeleitet hat. Ich hätte mir keinen besseren Betreuer wünschen können. Weiter danke ich herzlich Dr. Robert I. A. Patterson, der sich insbesondere in meinem ersten Jahr am WIAS rührend um mich gekümmert hat, und der mich in viele Geheimnisse der stochastischen Simulation eingeführt hat.

Danken möchte ich den Kolleg*Innen in der Forschungsgruppe *Numerische Mathematik und Wissenschaftliches Rechnen* des WIAS. Zuerst Ulrich Wilbrandt und Naveed Ahmed, mit denen eng zusammenzuarbeiten ich das Glück und das Vergnügen hatte. Laura Blank, Najib Alia und Jeanne Pellerin, mit denen ich (nacheinander) ein Büro teilen durfte. Franco Dassi, der mein "Coffee Neighbour" war. Weiter Felix Anker, Alfonso Caiazzo, Jürgen Fuhrmann, Alexander Linke, Christian Merdon, Michiel Renger, Timo Streckenbach, Hang Si, Marita Thomas, Wolfgang Wagner, Viktoria Wiedmeyer und allen Kolleg*Innen am WIAS und anderswo, die auf die eine oder andere Weise eine Idee, eine Einsicht, oder ein aufmunterndes Wort beigesteuert haben. Ein besonderer Dank auch an Marion Lawrenz, die jeden Tag so vielen den Rücken frei hält, wovon auch ich profitieren durfte.

Ich danke meinen wunderbaren Freundinnen und Freunden, stellvertretend seien genannt Johannes, Anja, Markus und Ulvi, sowie meiner großen, liebevollen Familie, und meiner geliebten Freundin Young-eun. Ohne euren Rückhalt, euer Verständnis und eure Liebe gäbe es weder mich noch diese Arbeit. Ich bin euch unendlich dankbar.

Schließlich möchte ich mich bei Willie Watson bedanken. Sein Tenor hat mir über so manches Konzentrations-Tief und aus so manchem Motivationsloch geholfen, und ich verspreche ihm hiermit, nach erfolgreicher Verteidigung alle seine Alben zu kaufen.

Contents

1 Introduction

In this thesis, an algorithm for the numerical solution of population balance systems is developed and investigated. Population balance systems (PBS) are systems of partial (integro-)differential equations, which describe the development of a population of particles in a fluid environment. A system of this type is defined on a time interval (t_0, t_1) with variable t, a spatial domain $\Omega_{\boldsymbol{x}}$ with variable $\boldsymbol{x}$ and a particle type space $\Omega_{\mathbf{m}}$ with variable $\mathbf{m}$. It comprises a population balance equation (PBE)

$$\frac{\partial}{\partial t} f + \boldsymbol{u} \cdot \nabla_{\boldsymbol{x}} f + \nabla_{\mathbf{m}} f = \mathcal{B}(f) - \mathcal{D}(f),$$

which is the "heart" of the population balance system; the incompressible Navier–Stokes equations (momentum balance and mass balance)

$$\begin{aligned} \frac{\partial}{\partial t} \boldsymbol{u} - \nu \Delta \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} + \nabla p &= \mathbf{f}_{\boldsymbol{u}} \\ \nabla \cdot \boldsymbol{u} &= 0; \end{aligned}$$

and a number of convection-diffusion-reaction equations of the type

$$\frac{\partial}{\partial t} c - \varepsilon \Delta c + \boldsymbol{u} \cdot \nabla c + r(c) = \mathbf{f}_c.$$

Population balance systems can be used for the description of multiple physical phenomena, but all of them have in common the presence of a large number of small entities ("particles"), which are transported by a fluid flow, and interact with each other and with the surrounding fluid. Those "particles" can be particles in the physical sense: sediment particles in a submarine environment, pollutant particles in the atmosphere, or crystals in a crystallization device are conceivable. In more exotic applications of population balance *equations*, the particles are, e.g., individuals in a herd of animals or genomes in a gene pool. In this thesis, the model systems come from chemical engineering, in particular from crystallization processes in fluid environments. Especially, we will regard population balance systems which include particle coagulation. Therefore the right-hand side of the PBE will always contain a coagulation integral term. The PBE can thus be regarded as an extended version of the Smoluchowski coagulation equation of Smoluchowski (1916).

Since population balance systems consist of three types of equations, each of which makes very specific demands to the numerical methods used for their solution, they are a natural application area for coupled methods of various type. Coupled or "splitting" methods are those, in which different numerical schemes are applied for each of the equations in the system, then coupling

the partial solutions after application of the individual schemes. The coupled method which will be developed and applied in this thesis is new and special in that way, that it brings together two very different numerical approaches. While the Navier–Stokes equations and the convection-diffusion-reaction equations will be solved with advanced deterministic finite element methods from the area of computational fluid dynamics (CFD), the population balance equation will be reformulated and solved in a stochastic manner, making use of a highly developed stochastic particle simulation algorithm. Just as the particle population is embedded into the fluid flow, this stochastic approach to its computation will be embedded into a CFD simulation framework. The resulting coupled solver will prove a powerful computational tool for the numerical solution of PBS: efficient, robust and flexible to extend. In the following we will first gather several aspects which motivate the coupled approach, and then give the outline of this thesis.

1.1 Motivation

In scientific computing, coupled methods are often useful for the numerical solution of systems of equations, when the system comprises very different types of equations. In the PBS case, three types of equations occur. For the population balance equation itself different numerical approaches have been proposed, deterministic and stochastic ones. A standard reference for the numerics of PBE is Ramkrishna (2000). Among the existing approaches stochastic methods of kinetic Monte Carlo type stand out because of their ability to include microscale characteristics of the particles, while the convergence of the numerical solution to the solution of the macroscopic PBE in a suitable notion of convergence was proven for several cases of interest, notably the Smoluchowski coagulation equation. From a computational point of view, classical methods for PBEs suffer from a "curse of dimensionality". The spaces on which they are defined, combining spatial domain and type domain, can easily become four, five or six dimensional – the computing time of classical methods like finite difference schemes blows up in such settings. Stochastic particle methods do not suffer from this issue.

The PBS has a multiscale character, which is done justice by a coupled stochastic-deterministic approach. For the fluid and the transported quantities a macroscopic modeling approach via classical PDEs seems rather nearby, since to the human perception and to classical mechanics the fluid appears as a homogeneous bulk, whose motion and composition can be well characterized by means and averages. For the particles which are transported by the fluid, this approach seems less obvious. Nevertheless it is the classical modeling assumption of PBEs that the particle population is dense and fine enough to by characterized by an averaged, macroscopic density function alone. But even in the classical model the microscopic character of the particles enters, namely into the formulation of source and sink terms which describe interactions of the particles with each other and with their fluid surroundings. Stochastic simulation methods reflect this microscopic character way better. There, "representative",

"computational" or "notational" particles are present in model and simulation, and those can be used to model microscopic processes to any desired level of detail. Still, macroscopic quantities can be gained by averaging over the population of computational particles, allowing for results of the aforementioned type, where the solution of the stochastic particle simulation (SPS) can be shown to converge against a solution of the macroscopic PBE. In that sense, the strong appeal of stochastic methods is the efficient yet exact introduction of the microscale into the numerical solution of the PBS.

The other equations of the PBS are the incompressible Navier–Stokes equations (NSE) and a number of convection-diffusion-reaction equations (CDRE). The complications of their analysis and numerics have motivated the emergence of an extensive literature on the subject. A standard reference for the numerics of CDRE is Roos et al. (2008), for the numerics of the incompressible NSE John (2016) offers a good overview, at least for finite element approaches. Finite elements are the basic discretization method that is used for the solution of NSE and CDRE in this thesis, yet several aspects justify their categorization as "advanced" finite element methods. In particular, we make use of a domain decompositioning method for the solution of the Navier–Stokes equations, assess solvers for linear saddle point problems and identify a suitable solver for to be applied in the full 3d setting, and make use of a specialized algebraic flux correction scheme for the numerical stabilization of the CDRE. The assessment of the solvers for saddle point problems, and the identification of an appropriate solver for the considered NSE problem is an original contribution of this thesis. This shows that the urge to solve the coupled PBS can also stimulate numerical research connected to its component equations.

The particular stochastic particle simulation method which we use has been developed, refined and applied in a relatively recent series of papers by the research group of Prof. Markus Kraft at the University of Cambridge, see, e.g., Patterson et al. (2011) for a representative of that series. Its particular appeal, beyond the mentioned general facts on SPS, is its computational efficiency and the robustness of its implementation.

Coupling this stochastic particle simulation method into a CFD solver framework is an undertaking which requires great attention to detail, although the splitting scheme itself is rather simple. Especially the spatial extension to 2d and 3d, which is the main contribution of this thesis, requires great care. Similar methods, coupling CFD and stochastic methods in two or three dimensions have mainly been proposed in the context of the Boltzmann equation, see, e.g., Wu and Lian (2003). For coagulating particles the work Liu and Chan (2017) comes to mind, where an aerosol in a wind channel is simulated, using a similar splitting scheme with different constituents.

The main result of this thesis, the instationary, fully coupled stochastic-deterministic algorithm in 3d, can be regarded as the last bridge stone between two lines of research which developed towards each other for some time now. From the view of the SPS, where in Patterson and Wagner (2012) spatial inhomogeneity and advective transport had come into play in 1d, the coupled algorithm appears as an extension to 2d and 3d, enriching the simulation by a thorough CFD flow computation. From the CFD point of view, in an approved

PBS framework (see Suciu (2013)), the most essential part has been exchanged, blazing a trail towards higher dimensional particles: the solution scheme for the central population balance equation. Since the author of this thesis received his mathematical training mostly in classical analysis and numerics of PDEs, he has the tendency to view things from the CFD perspective. This has influenced several decisions during the work at this thesis.

1.2 Outline

Let us give the outline of the thesis. The first three chapters are devoted to the introduction of the constituent equations of the population balance system, and to the particular methods which we will use for their numerical solution. In Chapter 2 the incompressible Navier–Stokes equations get introduced. We give their discretization in time with variants of the Crank–Nicolson method and in space with the finite element method. We then show how linear systems of equations of saddle point type emerge from this discretization, discuss several options for linear solvers for that type of system, and then show some results on computing time assessments which we contributed to. Those were published in Ahmed et al. (2018). In the final section of Chapter 2 we introduce a classical FEM domain decompositioning method that was used for the distributed memory parallelization of ParMooN, our CFD code. This parallelization was mainly done by Prof. Ganesan at IISc Bangalore, and extended to different classes of saddle point solvers by ourselves. Descriptions of the method, the software and some numerical results were published in Ganesan et al. (2016) and Wilbrandt et al. (2017).

The rather short Chapter 3 introduces scalar PDEs of convection-diffusion-reaction type, their discretization and variants of the finite element method suitable for their solution. Especially, we describe a scheme of algebraic flux correction type for the stabilization of the convective term in Section 3.2, and a possibility to deal with systems of reactively coupled CDRE in Section 3.3.

Chapter 4 is devoted to the introduction of stochastic particle methods for the solution of population balance equations, especially such which are suitable for the Smoluchowski coagulation equation. We concentrate on such methods which built on the Marcus–Lushnikov process, a very intuitive stochastic coagulation model. The chapter starts with a short literature overview and proceeds with a mathematical introduction of the Marcus–Lushnikov process. We go into great detail here, re-introducing textbook definitions and properties of Markov processes and some of their properties necessary for their simulation. The final Section 4.3 introduces the actual stochastic simulation algorithm we employ, mentioning and describing several variants and improvements.

After those introductory chapters, Chapter 5 has a central position and function. Here we introduce the population balance system in full, comment on its inherent coupling mechanisms, find a formulation that is suitable with the stochastic-deterministic method we have in mind, and finally give the coupling algorithm. Having formulated the central problem and method of the thesis in that way, we can proceed towards the main part, which comprises two modeling

and simulation projects of crystallization devices.

In Chapter 6 an axisymmetric 2d simulation of an experimental flow tube crystallizer for aspirin is performed with the stochastic-deterministic method. Experimental results on the original device were published by a chemical engineering group of TU Graz in Eder et al. (2010). Those experimental results are reproduced with the new coupled simulation method, using a one-dimensional particle model and an axisymmetric 2d spatial geometry.

Chapter 7 presents a full 3d simulation. There we model and simulate a crystallization experiment of potassium alum in a fluidized bed crystallizer that is operated by the process engineering group of Prof. Sundmacher at OVGU Magdeburg. Although the particle description is one-dimensional still, the spatial environment is simulated in 3d, and the flow field is slightly turbulent. This requires additional work on both the flow field simulation and the stochastic particle simulation, including a turbulent model and a particle wall reflection algorithm.

In Chapter 8 we first give a conclusion on what has been achieved in this work, and then an outlook on what lies ahead. There we list open problems, urging questions, and several ideas how to overcome those.

2 Numerical Methods for the Incompressible Navier–Stokes Equations

In this chapter we want to show how to gain numerical solutions of time-dependent incompressible flow problems in reasonable computing time. Incompressible flows are governed by the instationary, incompressible Navier–Stokes equations (NSE). In introductory textbooks to computational fluid dynamics, the instationary NSE are usually to be found in one of the last chapters (John (2016), Ferziger and Perić (2002), Sohr (2001), Temam (1977)), or not at all (Girault and Raviart (1986), Elman et al. (2005), Galdi (2011)). They contain several mathematical features, each of which introduces difficulties of its own, and it is sensible, not at last from an educational point of view, to introduce them one after the other. For that purpose, textbooks carefully lead the reader through the Stokes equations, the Oseen equations, and the steady-state Navier–Stokes equations. Each of these equations introduces new difficulties, and new basic concepts are to be understood. It is out of the scope of this thesis to give such a profound introduction to the subtleties of the Navier–Stokes equations. Nevertheless we want to introduce methods which are used in the following chapters. We also want to depict our own understanding of the different concepts necessary to compute a numerical approximation to a solution of the NSE.

Thus we pursue a pragmatic approach. Our goal is to perform direct numerical simulations of the NSE in the framework of the Galerkin finite element method. Several paths lead to that goal, and there is a particular path which we favor. We are going to follow this path and present the necessary details, point out other directions one could take, but do not follow them. In doing so, we hope to give the reader an understanding of how our simulation methods work and how they are motivated.

We start from the formulation of the full instationary NSE in Section 2.1, leaving out their derivation but commenting on the constituents of the equations and deriving a de-dimensionalized formulation. Section 2.2 deals with discretizations of the NSE in time and space, namely variants of a one-step theta-scheme and the finite element method. Then in Section 2.3 we dwell on fast solvers for the resulting linear systems and present some of our own results on that topic. Section 2.4 finally explains a domain decompositioning method we use for parallel computation and goes into detail on the parallelization of selected solvers for linear saddle point problems.

2.1 The instationary Navier–Stokes equations for incompressible flows

Let us start our investigation of the instationary Navier–Stokes equations for incompressible flows from their dimensionalized form as given in (John, 2016, p. 22). We will formulate them in 3d only, their 2d formulation is very similar. Let $\Omega \subseteq \mathbb{R}^3$ be a bounded Lipschitz domain and $T \in \mathbb{R}^+$. Then the equations look as follows:

$$\begin{aligned} \frac{\partial}{\partial t}\boldsymbol{u} - \nu\Delta\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u} + \nabla\frac{P}{\rho} &= \frac{\mathbf{f}}{\rho} \quad \text{in } (0,T)\times\Omega, \\ \nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } (0,T)\times\Omega. \end{aligned} \tag{2.1}$$

The first equation, the *momentum equation*, is derived from the principle of conservation of linear momentum. It is actually a vectorial equation, consisting of three coupled spatial components. The second equation, usually referred to as *continuity equation*, is derived from the principle of conservation of mass. We do not give the derivation from first principles of continuum mechanics here, but refer the reader to the first chapters of John (2016) or Ferziger and Perić (2002).

The target unknown function in (2.1) is the fluid velocity $\boldsymbol{u} : [0,T]\times\Omega \to \mathbb{R}^3$. The velocity is a vectorial quantity, consisting of three spatial components, $\boldsymbol{u} = (u_1, u_2, u_3)$. Let for now all of the velocity components have the (derived) SI unit $^{\mathrm{m}}/_{\mathrm{s}}$. The scalar function P is the pressure in Pascal (Pa). The pressure is often interpreted as a Lagrangian multiplier for the continuity equation, punishing violation of mass conservation in the variational formulation of the NSE, see the note Ozanski (2015) for a good explanation of that intuition at the example of the Stokes equations.

The coefficients appearing in the momentum equation are the kinematic viscosity ν $[^{\mathrm{m}^2}/_{\mathrm{s}}]$ and the fluid density ρ $[^{\mathrm{kg}}/_{\mathrm{m}^3}]$. Both are positive constants, a feature which reflects the incompressibility and homogeneity of the fluid. In a compressible setting a spatial dependency would be admitted to both. On the right-hand side, $\mathbf{f}$ is a volume force acting on the fluid within Ω. This might be gravitation, result from an electro-magnetic field, or could stem from another source which is included in the particular model.

We should note that the equations have to be closed with boundary conditions on $\partial\Omega$ and an initial condition $\boldsymbol{u}_0$. These must be compatible with each other in the sense of $\boldsymbol{u}|_{\partial\Omega} \to \boldsymbol{u}_0|_{\partial\Omega}$ for $t \to 0$, and the initial conditions must be divergence-free, $\nabla\cdot\boldsymbol{u}_0 = 0$, in some sense, see (John, 2016, p.25). To be precise, one requires $\boldsymbol{u}_0 \in H_{\mathrm{div}}(\Omega)$ ((John, 2016, p.334, Definition 7.6)), where

$$H_{\mathrm{div}}(\Omega) := \left\{\boldsymbol{v} \in L^2(\Omega) : \nabla\cdot\boldsymbol{v} = 0 \text{ in } \Omega \text{ and } \boldsymbol{v}\cdot\mathbf{n} = 0 \text{ on } \partial\Omega \text{ in trace sense}\right\}. \tag{2.2}$$

Since the choice of boundary conditions influences the formulation of a discrete-in-space equivalent to (2.1), we will postpone the matter to Section 2.2. Even there we will restrict ourselves to homogeneous Dirichlet ("essential") boundary conditions, for the sake of brevity. In applications we will use so-called natural boundary conditions, too, see (John, 2016, p.27, Remark 2.27).

We must further note that there are different possible formulations of the NSE in addition to (2.1). Especially for the viscous term $\nu\Delta\boldsymbol{u}$ and the convective term $(\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u}$ there exist several different re-arrangements. The form of the convective term given in (2.1) is known as its *convective* form. The other formulations are equivalent in the continuous setting, but might lead to non-equivalent spatial discretizations, depending on whether the continuity condition is fulfilled exactly or only in an approximate sense for the discrete functions.

It is convenient for the simulation of the NSE to introduce a de-dimensionalized formulation of the equations. Within this formulation the number of free coefficients is reduced to one, and with this *Reynolds number* one has a useful dimensionless quantity at hand, with which flows can be classified. Furthermore the process of de-dimensionalizing allows to bring the solution values into a computationally convenient regime of floating point numbers. Although not difficult, we want to make the point clear and therefore show the de-dimensionalization in detail.

To de-dimensionalize Equation (2.1) we have to choose a characteristic length $\tilde{L}$ and a characteristic velocity $\tilde{U}$ for the expected flow. A characteristic time $\tilde{T}$ then follows[1] by $\tilde{T} = \frac{\tilde{L}}{\tilde{U}}$. For special, well-understood problems as a flow through a straight tube, there are widely accepted standards how to choose the characteristic quantities of the problem. For other, less wide-spread examples, choosing characteristic quantities is a modeling decision. It is sometimes a good starting point to choose the orders of magnitude of the SI units used to describe the problem setup as characteristic values. To give an example: If the flow domain is described in millimeters, $\tilde{L} = 10^{-3}$ m is usually a good choice. This scaling is of no consequence for the analysis, but from a computational point of view one tries to get results scaled in the order of 1, in order to avoid rounding precision issues. With the characteristic quantities chosen, one defines the de-dimensionalized variables

$$\tilde{\boldsymbol{u}} = \frac{\boldsymbol{u}}{\tilde{U}}, \quad \tilde{\boldsymbol{x}} = \frac{\boldsymbol{x}}{\tilde{L}}, \quad \text{and } \tilde{t} = \frac{t}{\tilde{T}}.$$

De-dimensionalizing the Equations (2.1) is now simply a matter of coordinate transformation. One aims at reformulating (2.1) in terms of

$$\tilde{\boldsymbol{u}}(\tilde{t}, \tilde{\boldsymbol{x}}) := \frac{\boldsymbol{u}(t, \boldsymbol{x})}{\tilde{U}}.$$

We perform the transformation term by term and index the differential operators with the variable they act upon, to keep things clear. The transformations

[1] The characteristic time can be chosen independently, too, which yields a second dimensionless coefficient: The Strouhal number. Since this number is of little consequence for what follows, we omit its derivation here.

are

$$\begin{aligned}
\frac{\partial}{\partial t}\boldsymbol{u}(t,\boldsymbol{x}) = \frac{\partial}{\partial t}\left(\tilde{U}\tilde{\boldsymbol{u}}(\frac{t}{\tilde{T}},\frac{\boldsymbol{x}}{\tilde{L}})\right) &= \frac{\tilde{U}}{\tilde{T}}\frac{\partial}{\partial \tilde{t}}\tilde{\boldsymbol{u}}(\tilde{t},\tilde{\boldsymbol{x}}),\\
\nu\Delta\boldsymbol{u}(t,\boldsymbol{x}) = \nu\Delta_{\boldsymbol{x}}\tilde{U}\tilde{\boldsymbol{u}}(\frac{t}{\tilde{T}},\frac{\boldsymbol{x}}{\tilde{L}}) &= \nu\frac{\tilde{U}}{\tilde{L}^2}\Delta_{\tilde{\boldsymbol{x}}}\tilde{u}(\tilde{t},\tilde{\boldsymbol{x}}),\\
(\boldsymbol{u}(t,\boldsymbol{x})\cdot\nabla_{\boldsymbol{x}})\boldsymbol{u}(t,\boldsymbol{x}) &= \frac{\tilde{U}^2}{\tilde{L}}\left(\tilde{\boldsymbol{u}}(\tilde{t},\tilde{\boldsymbol{x}})\cdot\nabla_{\tilde{\boldsymbol{x}}}\right)\tilde{\boldsymbol{u}}(\tilde{t},\tilde{\boldsymbol{x}}),\\
\nabla_{\boldsymbol{x}}\frac{P(t,\boldsymbol{x})}{\rho} &= \frac{1}{\tilde{L}}\nabla_{\tilde{\boldsymbol{x}}}\frac{P(\tilde{t},\tilde{\boldsymbol{x}})}{\rho}.
\end{aligned} \tag{2.3}$$

As de-dimensionalized momentum equation we obtain

$$\begin{aligned}
&\frac{\tilde{U}}{\tilde{T}}\frac{\partial}{\partial\tilde{t}}\tilde{\boldsymbol{u}}(\tilde{t},\tilde{\boldsymbol{x}}) - \nu\frac{\tilde{U}}{\tilde{L}^2}\Delta_{\tilde{\boldsymbol{x}}}\tilde{u}(\tilde{t},\tilde{\boldsymbol{x}})\\
&\qquad + \frac{\tilde{U}^2}{\tilde{L}}\left(\tilde{\boldsymbol{u}}(\tilde{t},\tilde{\boldsymbol{x}})\cdot\nabla_{\tilde{\boldsymbol{x}}}\right)\tilde{\boldsymbol{u}}(\tilde{t},\tilde{\boldsymbol{x}}) + \frac{1}{\tilde{L}}\nabla_{\tilde{\boldsymbol{x}}}\frac{P(\tilde{t},\tilde{\boldsymbol{x}})}{\rho} = \tilde{\mathbf{f}}(\tilde{t},\tilde{\boldsymbol{x}}).
\end{aligned} \tag{2.4}$$

The continuity equation is transformed as

$$0 = \nabla_{\boldsymbol{x}}\cdot\boldsymbol{u}(t,\boldsymbol{x}) = \frac{1}{\tilde{L}}\nabla_{\tilde{\boldsymbol{x}}}\cdot\tilde{\boldsymbol{u}}(\tilde{t},\tilde{\boldsymbol{x}}).$$

To get rid of the coefficient in front of the non-linear convective term, we multiply the momentum equation (2.4) with $\tilde{L}/\tilde{U}^2$:

$$\begin{aligned}
&\frac{\partial}{\partial\tilde{t}}\tilde{\boldsymbol{u}}(\tilde{t},\tilde{\boldsymbol{x}}) - \frac{\nu}{\tilde{L}\tilde{U}}\Delta_{\tilde{\boldsymbol{x}}}\tilde{u}(\tilde{t},\tilde{\boldsymbol{x}})\\
&\qquad + \left(\tilde{\boldsymbol{u}}(\tilde{t},\tilde{\boldsymbol{x}})\cdot\nabla_{\tilde{\boldsymbol{x}}}\right)\tilde{\boldsymbol{u}}(\tilde{t},\tilde{\boldsymbol{x}}) + \frac{1}{\tilde{U}^2}\nabla_{\tilde{\boldsymbol{x}}}\frac{P(\tilde{t},\tilde{\boldsymbol{x}})}{\rho} = \frac{\tilde{L}}{\tilde{U}^2}\tilde{\mathbf{f}}(\tilde{t},\tilde{\boldsymbol{x}}).
\end{aligned} \tag{2.5}$$

In (2.5), due to the dependent choice of the characteristic time $\tilde{T}$, also the time derivative ended up with 1 as coefficient.

The inverse of the coefficient $\nu/\tilde{L}\tilde{U}$ of the viscous term is known as *Reynolds number*,

$$\mathrm{Re} := \frac{\tilde{L}\tilde{U}}{\nu}. \tag{2.6}$$

The Reynolds number is an important characteristic number of a flow, it allows for its classification. While a low Reynolds number indicates a slow, viscous flow, a high Reynolds number of several thousand (or higher) is an indicator of turbulent flows.

Let us now skip the tilde superscripts in the dimensionless notation, and re-declare ν to be the dimensionless viscosity,

$$\nu := \frac{1}{\mathrm{Re}}. \tag{2.7}$$

If we also re-define the right-hand side to be $\mathbf{f}$, and introduce the dimensionless pressure $p := \frac{P}{\rho \tilde{U}^2}$, the full NSE in their de-dimensionalized form read

$$\begin{aligned} \frac{\partial}{\partial t}\boldsymbol{u} - \nu\Delta\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p &= \boldsymbol{f} \quad \text{in } (0,T)\times\Omega, \\ \nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } (0,T)\times\Omega. \end{aligned} \tag{2.8}$$

These equations are the basic model of incompressible fluid dynamics. It is worth contemplating for a moment, why they are called *full* Navier–Stokes equations. First of all, equations (2.8) offer a *full* description of the flow of an incompressible fluid, details up to the smallest micro-scale can be resolved. On the other hand, the term *full* Navier–Stokes equations is used, because they are considered "complete" if compared with their simplified counterparts. There are several options of reducing the complexity of the equations. Among those are:

- Removing the time dependency and the time derivative $\frac{\partial}{\partial t}\boldsymbol{u}$. This results in the *stationary* Navier–Stokes equations, which do not carry time dependency. The simplification is only reasonable if a stationary behavior of the flow is to be expected due to a low velocity or high viscosity, i.e., for low Reynolds numbers, and, obviously, for time-independent data.

- Starting from the stationary Navier–Stokes equations, removing the nonlinear convective term $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$. This results in the well-understood, linear Stokes equations, which model a flow driven by viscous forces only. Such flows are also called *creeping flows*.

- Starting from the stationary Navier–Stokes equations, replace the nonlinear convective term by a linearized version $(\mathbf{w}\cdot\nabla)\boldsymbol{u}$, with a known *wind* $\mathbf{w}$. These *Oseen equations* have no physical meaning, but appear as auxiliary problems in several standard methods for solving the full Navier–Stokes equations.

- Utilizing a turbulence model. Turbulence models such as the Variational Multiscale (VMS) Method, the Large Eddy Simulation (LES) method or the k-ε-method offer complexity reductions which are applicable for flows with high Reynolds numbers.

All of these are firmly established in fluid dynamics. In contrast to these approaches, Equations (2.8) form the *full* Navier–Stokes equations.

The full NSE have three inherent sources of difficulties, as is remarked in (John, 2016, p.23). These difficulties are:

- the coupling of velocity and pressure,

- the nonlinearity introduced with the convective term,

- the property of convection dominance, which gets more critical the higher the Reynolds number becomes.

While the third source of difficulty is mainly a numerical concern, the other two have also fundamentally driven (and overshadowed) the analysis of the NSE, starting in the 1970s with the works Babuška (1970/1971) and Brezzi (1974). All three of these difficulties still govern the numerical simulation of the NSE today, and we will refer back to them occasionally.

2.2 Discretizing the Navier–Stokes equations

Analytical solutions to the Navier–Stokes equations are rare and only known for very special cases. Applying numerical schemes to the NSE and resorting to computational power is so common that the term "solving" the NSE is widely used in the sense of finding a numerical approximation to a solution. Since computers "understand" numerical problems only in the language of linear systems of equations, the continuous equations have to be broken down to such. For the full time-dependent NSE this is a process of multiple stages. They have to be discretized in time, a way to deal with their non-linearity has to be found, and finally a discretization in space is necessary. For this process many variants exist, which have different scopes, advantages, disadvantages and interconnections. It is out of the scope of this thesis to give an overview, let alone discuss several of them in detail. We will instead only explain those discretization strategies that we have sufficient experience with. These are the ones used in the later chapters of this thesis.

The decision on a spatial discretization method gives direction to the entire numerical scheme and is thus the most fundamental one. Here we opt for the finite element method (FEM), which is most successful in structural mechanics, but also widely acknowledged in computational fluid dynamics. The most classical textbook on the application of FEM to the Navier–Stokes equations is Girault and Raviart (1986), though restricted to the stationary case. A very recent monograph which we make heavy use of is John (2016), another recent introductory work is Layton (2008). For an introductory work on FEM, more focused on the basics of the method than on its applications in CFD see Braess (1997). The main alternatives to the FEM, the finite differences and finite volumes method are presented from an engineer's point of view in Ferziger and Perić (2002).

Among the advantages of the FEM one has to highlight that its numerical analysis is highly developed. Also it permits the use of unstructured spatial meshes and thus allows for computations on complicated geometries. It blends perfectly with state-of-the-art meshing programs. One disadvantage, as pointed out in (Ferziger and Perić, 2002, p.37), which but only appears with unstructured meshes, is that the resulting matrices do not have as compact a band sparsity structure as gained with finite volume or finite difference schemes on structured meshes. This is less desirable for solvers. We will raise the issue of fast solvers for the resulting equation systems in Section 2.3. Another drawback of the FEM is that it does not, in general, maintain conservation laws fulfilled by the continuous solutions. It is a valuable rule of thumb that numerical schemes should reflect properties of the continuous equations. For the Navier–Stokes

equations, which derive from two conservation laws, this requirement holds especially true, and some effort had to be put into FEM historically to make it suitable for the NSE.

Concerning the relation between temporal and spatial discretization, we apply what is known as horizontal method of lines, i.e., we apply the temporal discretization first and derive the spatial discretization second. From the many methods of temporal discretization available for stiff ordinary differential equations we will only present the second order convergent, implicit Crank–Nicolson method. This method is widely used, as it is A-stable and easy to implement. It also gives rise to an implicit-explicit (IMEX) method termed "stabilized linearly extrapolated Crank–Nicolson" (CNLE(stab), Ingram (2013)), which we want to present towards the end of the chapter. Applying CNLE(stab) reduces the computational work connected with the third step towards linear systems, the linearization of the convective term. If not applying CNLE(stab), it is necessary to linearize the convective term. The most widely used methods here are Picard and Newton iteration.

All in all we are concerned with what is known as "direct numerical simulation". In comparison to other approaches, no further modeling assumptions are made to compute a discrete solution of (2.7).

2.2.1 Temporal discretization

We start with a description of the temporal discretizations that we will use later. To get an intuition about the origin of the full scheme presented below, let us sketch how a semi-discretization in time is achieved. Therfore we bundle all spatial derivatives in an (unspecified) operator M. From the point of view of time, the momentum equation of (2.8) presents itself as a vectorial first order ordinary differential equation:

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \mathrm{M}(\boldsymbol{u}, p).$$

The most straightforward way of discretizing an ordinary differential equation of first order is to exchange the derivative by the forward difference and choose a one-step theta-scheme for the right-hand side. Choose thus a (for now constant) time step length $\Delta t \in \mathbb{R}^+$, let $n \in \mathbb{N}$, and $\theta \in [0, 1]$. Then, with known old solution $\boldsymbol{u}_k$, the new solution $\boldsymbol{u}_{k+1}$ is given implicitly (or explicitely, if $\theta = 0$) by:

$$\frac{\boldsymbol{u}_{k+1} - \boldsymbol{u}_k}{\Delta t} = (1-\theta)\,\mathrm{M}(\boldsymbol{u}_k, p_k) + \theta\,\mathrm{M}(\boldsymbol{u}_{k+1}, p_{k+1}). \tag{2.9}$$

The most prominent values for θ are 0 (explicit Euler scheme), 1 (implicit Euler scheme), and 0.5, which corresponds to the popular Crank–Nicolson time stepping scheme.

In the common Crank–Nicolson semi-discretization of (2.8),

$$\begin{aligned} \frac{\boldsymbol{u}_{k+1}-\boldsymbol{u}_k}{\Delta t} &= \frac{1}{2}\left(\nu\Delta\boldsymbol{u}_k - (\boldsymbol{u}_k\cdot\nabla)\boldsymbol{u}_k + f_k\right) \\ &\quad + \frac{1}{2}\left(\nu\Delta\boldsymbol{u}_{k+1} - (\boldsymbol{u}_{k+1}\cdot\nabla)\boldsymbol{u}_{k+1} - \nabla p_{k+1} + f_{k+1}\right), \\ \nabla\cdot\boldsymbol{u}_{k+1} &= 0, \end{aligned} \tag{2.10}$$

one notices a discrepancy. Though the spatial part of the equation depends on the pressure, besides the velocity, only the pressure of the current time step, p_{k+1}, appears in (2.11). The pressure of the previous time step p_k is missing from the formulation.[2]

The formulation (2.10) actually arises by applying the "recipe" (2.9) only with respect to the velocity $\boldsymbol{u}$ and then supplementing the pressure p_{k+1} as Lagrangian multiplier for the continuity equation. As is noted in Rang (2008), this strategy is due to the lack of an initial pressure p_0, which would close the scheme. The disadvantage of (2.10) is that the pressure p_{k+1} is now actually an approximation to $p(t_k + \frac{1}{2}\Delta t)$, i.e., to p at the wrong time. In Rang (2008) a *pressure corrected Crank–Nicolson scheme* is investigated, in which p_0 is included, which does but not grant any numerical advantages. Therefore we might as well stick with (2.10).

Multiplying (2.10) with Δt and reordering, we get at each time step $k+1$ a quasi-stationary Navier–Stokes problem:

$$\begin{aligned} \boldsymbol{u}_{k+1} - \frac{\Delta t}{2}\nu\Delta\boldsymbol{u}_{k+1} + \frac{\Delta t}{2}(\boldsymbol{u}_{k+1}\cdot\nabla)\boldsymbol{u}_{k+1} + \Delta t\nabla p_{k+1} & \\ = \boldsymbol{u}_k + \frac{\Delta t}{2}\left(\nu\Delta\boldsymbol{u}_k - (\boldsymbol{u}_k\cdot\nabla)\boldsymbol{u}_k + f_k + f_{k+1}\right) & \\ \Delta t\nabla\cdot\boldsymbol{u}_{k+1} = 0. & \end{aligned} \tag{2.11}$$

The continuity equation has been multiplied with Δt, too, which makes the computations easier, as we will see in Section 2.3. The numerical analysis also benefits from this transformation.

Since it is closely connected to the time discretization, we consider this the right place to present solution methods for the nonlinearity of (2.11). The most common methods are the Picard and Newton iteration, see John (2006) for a performance comparison of both.

For the Picard iteration, at time step $k+1$, the convective term in (2.11) is approximated as

$$(\boldsymbol{u}_{k+1}\cdot\nabla)\boldsymbol{u}_{k+1} \approx (\boldsymbol{u}_k\cdot\nabla)\boldsymbol{u}_{k+1}.$$

The solution $\boldsymbol{u}_{k+1}^0$ of the resulting Oseen equations is then put as "wind" into the convective term again, giving rise to an iterative solution procedure, where at step $n+1$ the actual convective term is replaced as

$$(\boldsymbol{u}_{k+1}^{n+1}\cdot\nabla)\boldsymbol{u}_{k+1}^{n+1} \approx (\boldsymbol{u}_k^n\cdot\nabla)\boldsymbol{u}_{k+1}^{n+1}.$$

[2] As we remember from Section 2.1, the coupling of velocity and pressure was the first of the stated inherent difficulties of the NSE.

This iteration is conducted until a sufficiently small residual is reached.

The Newton iteration proceeds in the same spirit, yet there the convective term at step $n+1$ is approximated as

$$(\boldsymbol{u}_{k+1}^{n+1} \cdot \nabla)\boldsymbol{u}_{k+1}^{n+1} \approx -(\boldsymbol{u}_{k+1}^{n} \cdot \nabla)\boldsymbol{u}_{k+1}^{n} + (\boldsymbol{u}_{k+1}^{n+1} \cdot \nabla)\boldsymbol{u}_{k+1}^{n} + (\boldsymbol{u}_{k+1}^{n} \cdot \nabla)\boldsymbol{u}_{k+1}^{n+1}.$$

In John (2006) it is shown that for a 3d reference problem, the Picard iteration is more efficient in terms of computational time than the Newton method, if combined with a multigrid preconditioned iterative solver (see Section 2.3 for a description of this kind of solver). A different picture is given by (John, 2016, p.371, Example 7.57), where the Newton iteration combined with an iterative solver is clearly faster than the Picard iteration. In general, the Picard iteration has a larger convergence radius and is thus less dependent on the choice of the initial iterate (John, 2016, p.319, Example 6.47).

Picard and Newton iteration can be applied to steady-state problems in the same spirit, but there one has to come up with an a priori initial guess. For time-dependent problems there exist very appealing approaches to combine time stepping and linearization of the convective term into one. These approaches are known as *IMEX* (implicit-explicit) *schemes*. We have particularly good experience with the CNLE(stab) approach from Ingram (2013) (*stable linearly extrapolated Crank–Nicolson scheme*), as we applied it for flow computations published in Wiedmeyer et al. (2017).

The basic idea of IMEX schemes is to bypass the nonlinear iteration in each time step by inserting a linear extrapolation from the former time steps as wind into both convective terms. In the CNLE(stab) of Ingram (2013) the convective term $(\boldsymbol{u}_{k+1} \cdot \nabla)\boldsymbol{u}_{k+1}$ on the left-hand side of Equation 2.11 is replaced by

$$((2\boldsymbol{u}_k - \boldsymbol{u}_{k-1}) \cdot \nabla)\boldsymbol{u}_{k+1}$$

and the convective term $(\boldsymbol{u}_k \cdot \nabla)\boldsymbol{u}_k$ on the right-hand side by

$$((2\boldsymbol{u}_{k-1} - \boldsymbol{u}_{k-2}) \cdot \nabla)\boldsymbol{u}_k.$$

In this way, one obtains a linear scheme whose stability has been proven without a limitation on the time step size in Ingram (2013). Also its implementation into an existing finite element code is rather easy. As a last remark, note that in order to perform an iteration of the CNLE(stab) scheme, two former velocity solutions must be known. At the first step, where only one former solution is known (i.e., the initial condition $\boldsymbol{u}_0$) this is not the case, and therefore $\boldsymbol{u}_1$ is best gained using one of the two classical schemes.

2.2.2 Spatial discretization with the finite element method

The horizontal method of lines proceeds with defining a spatial discretization of the time-discrete problem. We aim at the Galerkin finite element method, and will therefore have to introduce function spaces (discrete and continuous) and a weak formulation of the time discretized NSE.

Preliminarily, let us restate (2.11) in a more convenient fashion. Let $\tau := {}^{\Delta t}/_{2}$ and subsume the right-hand side of the momentum equation at time step k as

$\boldsymbol{b}_k$. We denote the wind in the convective term as $\boldsymbol{w}_k$. Note that $\boldsymbol{b}_k$ and $\boldsymbol{w}_k$ take different forms depending on whether we use the Crank–Nicolson plus Picard scheme or the CNLE(stab). The linearized problem[3] at time step k can then be stated as

$$\begin{aligned} \boldsymbol{u}_k - \tau\nu\Delta\boldsymbol{u}_k + \tau(\boldsymbol{w}_k \cdot \nabla)\boldsymbol{u}_k + 2\tau\nabla p_k &= \boldsymbol{b}_k, \\ 2\tau\nabla \cdot \boldsymbol{u}_k &= 0. \end{aligned} \tag{2.12}$$

The Galerkin finite element method introduces discrete function spaces, in which a discretized version of (2.12) can be stated.

We have reached a point in our discussion, where we cannot ignore boundary and initial conditions any longer. It is convenient for the presentation (and for the analysis, but not so much for the computation) to allow only homogeneous Dirichlet boundary conditions. For all $k \in \mathbb{N} \cup \{0\}$ we impose upon the velocity the condition

$$\boldsymbol{u}_k|_{\partial\Omega} \equiv \boldsymbol{0}.$$

Dirichlet conditions such as this are also referred to as "essential" boundary conditions in PDE literature, since they must be included into the choice of solution spaces and cannot simply be absorbed in the weak formulation of the equations.

For the time-discretized version of the instationary NSE, the continuous function spaces

$$V := H_0^1(\Omega) = \left\{ v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega \right\}$$

for each component of the velocity and

$$Q := L_0^2(\Omega) = \left\{ q \in L^2(\Omega) : \int_\Omega q(x)\mathrm{d}x = 0 \right\}$$

for the pressure offer an appropriate setting (John, 2016, pp.45-46). Both spaces are Hilbert spaces, in the following we will only use the scalar product of L_0^2, and will denote it $(\cdot,\cdot)$. The dual pairing of V' and V is denoted with angular brackets, $< \cdot, \cdot >$. The vectorial versions V^3 and $(V^3)'$ are written as $\mathbf{V}$ and $\mathbf{V}'$, for their dual pairing we use the same angular brackets. Furthermore, the right-hand side $\mathbf{f}$ must have been chosen in such a way that $\boldsymbol{b}_k$ can be interpreted as a continuous functional on $\mathbf{V}$, i.e., $\boldsymbol{b}_k \in \mathbf{V}'$.

The *weak* or *variational* formulation of equations (2.12) reads as follows.

Problem 2.2.1. ***Time-discretized and linearized Navier–Stokes equations.*** *Given $\boldsymbol{w}_k \in \boldsymbol{V}$ with $\nabla \cdot \boldsymbol{w}_k = 0$ and $\boldsymbol{b}_k \in \boldsymbol{V}'$, find $(\boldsymbol{u}_k, p_k) \in (\boldsymbol{V}, Q)$, which solve the equation system*

$$\begin{aligned} (\boldsymbol{u}_k, \boldsymbol{v}) + \tau\nu(\nabla\boldsymbol{u}_k, \nabla\boldsymbol{v}) + \tau((\boldsymbol{w}_k \cdot \nabla)\boldsymbol{u}_k, \boldsymbol{v}) - 2\tau(\nabla \cdot \boldsymbol{v}, p_k) &=< \boldsymbol{b}_k, \boldsymbol{v} > \\ -2\tau(\nabla \cdot \boldsymbol{u}_k, q) &= 0 \end{aligned}$$

for all test functions $(\boldsymbol{v}, q) \in \boldsymbol{V} \times Q$.

[3] In the case Crank–Nicolson plus Picard, a linear problem of that kind has to be solved several times per time step. It would be closer to the reality to speak about the linearized *problems* there.

The common procedure of finite element methods is to formulate Problem 2.2.1 in discrete function spaces instead of the continuous ones, which then leads to an algebraic system of linear equations, as we demonstrate in Section 2.3. The distinct feature of the conforming Galerkin method, which we show here, is that all test and ansatz functions belong to the same discrete space, which itself is a subspace of the continuous space.

Let thus $V_h \subseteq V$ and $Q_h \subseteq Q$ be finite-dimensional subspaces of the continuous spaces. Those conformity conditions ensure that the formulation of Problem 2.2.1 remains valid even if one replaces the continuous by the discrete ansatz and test function spaces.

To introduce a notion of finite elements, we slightly reformulate a thorough definition which we find useful, given by (Braess, 1997, p.72).

Definition 2.2.2. *A finite element is a triple* (T, Π, Σ) *with the following properties:*

(i) T *is a closed polyhedron in* $\mathbb{R}^d$.

(ii) Π *is a* s*-dimensional subspace of* $C(T)$ *with* $s \in \mathbb{N}$*, where* $C(T)$ *is the space of continuous functions from* T *to* $\mathbb{R}$*. Functions forming a basis of* $C(T)$ *are called shape functions.*

(iii) Σ *is a set of* s *linearly independent functionals on* Π*. From the linear independence of* Σ *follows that each* $p \in \Pi$ *is uniquely determined by the* s *values* $(\sigma(p) : \sigma \in \Sigma)$.

In this definition, s is the number of degrees of freedom. We will come back to that term in Section 2.2.3.

The crucial idea of the finite element method is to build up the spaces in the discrete formulation of a PDE from finite elements as given in the above Definition 2.2.2. To that aim, one demands a decomposition or *mesh* of the domain Ω into polyhedra T. The mesh $\mathscr{T}$ must fulfill

$$\Omega = \bigcup_{T \in \mathscr{T}} T$$

and the polyhedra T only intersect on sets of measure zero (vertices, edges, faces). A finite element space is gained by declaring a finite element on each $T \in \mathscr{T}$ and "patching" the elementwise function spaces Π_T together to form a space of real-valued functions on the whole domain Ω. This "patching" usually proceeds with the aim of obtaining functions of a certain global regularity, e.g., $C(\Omega)$ or $C^1(\Omega)$.

There are many concepts to obtain finite element spaces with that strategy, depending on the choices of $\mathscr{T}$, Π, and Σ, and we will not dwell upon the matter further at this point. Let us just note that the most commonly used finite element spaces restrict themselves to using just one class of geometric shapes in $\mathscr{T}$ (e.g. triangles in 2d or hexahedra in 3d) and sets Π, and Σ whose elements only differ by affine transformation. Such finite element spaces are called *affine families* (Braess, 1997, pp.71-72). Affine families allow for a very

far developed analysis of approximation and convergence properties, the basis of which is the *Bramble–Hilbert lemma* (Braess, 1997, p.76). An affine family possesses a *reference element* $(T_{\text{ref}}, \Pi_{\text{ref}}, \Sigma_{\text{ref}})$ - the elements of the family are gained from the reference element via affine transformation.

A special trait of the Galerkin FEM for the Navier–Stokes equations is that, due to the coupled nature of the equations, *two* finite element spaces must be chosen for their spatial discretization. One is for the components of the velocity and one is for the pressure function. It has been found in the 1970s (Babuška (1970/1971), Brezzi (1974)) that these must be chosen in accordance to each other. The spaces must be connected by the discrete inf-sup condition (John, 2016, p.55):

Definition 2.2.3. ***The discrete inf-sup condition for conforming finite element spaces.*** *Let $\boldsymbol{V}_h \subseteq \boldsymbol{V}$ and $Q_h \subseteq Q$ be the conforming finite element spaces used for the discretization of velocity and pressure, respectively. Further denote by $b : \boldsymbol{V}_h \times Q_h \to \mathbb{R}, \quad b(\boldsymbol{v}_h, q_h) = -(\nabla \cdot \boldsymbol{v}_h, q_h)$ the weak form of the divergence operator. Then the pair $(\boldsymbol{V}_h, Q_h)$ is said to fulfill the* ***discrete inf-sup condition****, if there exists $\beta > 0$ such that*

$$\inf_{q_h \in Q_h \setminus \{0\}} \sup_{\boldsymbol{v}_h \in \boldsymbol{V}_h \setminus \{\boldsymbol{0}\}} \frac{b(\boldsymbol{v}_h, q_h)}{\|\boldsymbol{v}_h\| \|q_h\|} \geq \beta. \tag{2.13}$$

This condition is a property of the divergence operator, controlled by its domain. In a non-conforming setting one has to replace the definition of b by a discrete sum over the elements of $\mathscr{T}$. For applications it is important, that the parameter β is independent of the grid size, since $\frac{1}{\beta}$ enters the finite element error estimation, and a behavior $\beta_h \overset{h \to 0}{\longrightarrow} 0$ for the grid-size parameter h slows down or rules out the convergence of the method (John, 2016, p.60).

The condition (2.13) can be regarded as a generalization of a coercitivity condition for a bilinear operator, which is defined on the Cartesian product of two different spaces – here $\mathbf{V}_h$ and Q_h. It is a crucial ingredient of the well-posedness of the discrete Problem 2.2.1 with $\mathbf{V}$ and Q replaced by their finite element space approximations $\mathbf{V}_h$ and Q_h.

Lots of effort has been put into showing inf-sup stability of different pairs of finite elements which are used for the NSE, see (John, 2016, pp.73). How extremely important this theoretical work is for practical computations is illustrated in (Braess, 1997, pp.147) at the instance of the historically favored $Q_1 - P_0$ element pair. This pair violates the discrete inf-sup condition. In practice, its instable behavior had been observed frequently, but only the finding of the discrete inf-sup condition provided an explanation of these instabilities. Furthermore, in (Elman et al., 2005, p.285) it is stated how important stable approximations are for the convergence of iterative solvers. See also Section 2.3.2 on the matter of fast solvers.

2.2.3 Remarks on finite element terminology

One has to admit that often in the literature some terms are used which have lost their original sharpness. The most prominent ones are, in our observation,

element and *degree of freedom*. The term *element* is precisely given in Definition 2.2.2. To be strict, even that definition does not yet cover all entities called finite elements, as for example geometrical shapes with curved boundaries are not contained. It is common to use the term *element* as well for parts of the entities defined in Definition 2.2.2, as for those assembled from them. It is used in the literature for

- just the geometrical entity T,
- the reference element of an affine family,
- the entire finite element space,

and, in the case of the Navier–Stokes equations with their tightly coupled velocity and pressure discretization, for

- a pair of finite element spaces used for the discretization of the Navier–Stokes equation.

Even fuzzier is the term *degree of freedom* (d.o.f.). In its original meaning, it denotes the parameters left in a certain physical or mathematical model to fit it to an actual case of application. This basically transfers to a finite element approximation. Given a domain $\Omega \subset \mathbb{R}^d$ with mesh $\mathscr{T}$ and a N-dimensional finite element space V_h with basis $(\varphi_{h,k})_{k=1,..,N}$, each finite element function $v_h \in V_h$ is determined uniquely by its coefficients $(\alpha_k)_{k=1,..,N}$. Each such coefficient, before determined, is a degree of freedom. One could put it like this: The possibility to make a choice of one coefficient of a finite element function is one degree of freedom. Usually there is a canonical way of choosing a basis for V_h, which is extending the basis functions of the single elements to the whole space. It is common to refer to each of the basis functions as a degree of freedom. If the basis functions can be identified with points in Ω or the reference element, it is common to call this point a degree of freedom, too. This occurs when finite elements whose interpolation conditions Σ consist of point evaluations of functions or their derivatives are used to build up V_h. Finally, if the degrees of freedom are numbered and a matrix is put up as described in Section 2.3.1, the indices of that matrix' rows and columns are referred to as degrees of freedom, too. We dwell so deeply on these matters, because the terms will accompany us throughout this thesis and will appear with all meanings explained here, usually without explicitely indicating with which exactly.

2.3 Linear saddle point problems and solvers

In this section we will show how a finite element approximation to Problem 2.2.1 leads to a system of linear algebraic equations, and how to solve such a system efficiently. We will explain the characteristics and difficulties of saddle point problems and present some of our own results on a comparison of linear solvers applied to them.

2.3.1 Deriving a discrete saddle point problem

As in the preceding section, let V_h and Q_h be finite element spaces for the velocity components and the pressure. Let the spaces fulfill the discrete inf-sup condition (2.13). Both spaces come with a set of natural basis vectors, depending on the particular choice of finite elements. We write

$$V_h = \bigoplus_{i=1,\dots,N} \varphi_i \qquad Q_h = \bigoplus_{i=1,\dots,M} \psi_i. \tag{2.14}$$

Here $N, M \in \mathbb{N}$ are the vector space dimensions of V_h and Q_h. A possible discrete velocity solution function at time step k, $\boldsymbol{u}_{k,h} \in \mathbf{V}_h$, has a unique representation as

$$\boldsymbol{u}_{k,h} = \left(\sum_{i=1}^{N} u_i^{(1)} \varphi_i, \sum_{i=1}^{N} u_i^{(2)} \varphi_i, \sum_{i=1}^{N} u_i^{(3)} \varphi_i \right),$$

and we collect the coefficients as

$$\bar{\boldsymbol{u}} := \left(\underline{u}^{(1)}, \underline{u}^{(2)}, \underline{u}^{(3)} \right) \in \mathbb{R}^{3N}.$$

The indices k and h are deliberately left out in the above definition, as we will not need them from here.

In the same spirit, the right-hand side $\boldsymbol{b}_{k,h}$ of the discrete momentum equation is uniquely determined by a vector of real coefficients,

$$\bar{\boldsymbol{b}} = \left(\underline{b}^{(1)}, \underline{b}^{(2)}, \underline{b}^{(3)} \right) \in \mathbb{R}^{3N}.$$

For the pressure p we get the representation

$$p_{k,h} = \sum_{i=1}^{M} p_i \psi_i, \tag{2.15}$$

with the coefficients vector

$$\bar{p} = (p_i)_{i=1,\dots,M} \in \mathbb{R}^M. \tag{2.16}$$

Thus, a solution $(\boldsymbol{u}, p)$ of the fully discretized Problem 2.2.1 is uniquely determined by the $3N + M$ unknown real coefficients $(\bar{\boldsymbol{u}}, \bar{p})$.

In order to gain the same number of linear equations from Problem 2.2.1, one withdraws to a finite set of test functions. As the momentum equation is linear in the test function $\boldsymbol{v}$ and the continuity equation linear in the test function q, any pair $(\boldsymbol{u}, p)$ of ansatz functions that solves momentum and continuity equation for all *basis* functions of $\mathbf{V}_h$ and Q_h will solve the equations for *any* pair of functions from $(\mathbf{V}_h \times Q_h)$. As there are $3N$ basis functions of $\mathbf{V}_h$ and M basis functions of Q_h, this yields the required number of linear equations.

We will now illustrate how the matrix that represents the linear system is derived. We will have to keep in mind that the actual discrete space we are

dealing with for the coupled problem is the Cartesian product $V_h \times V_h \times V_h \times Q_h$. Let us therefore fix some notation first.

The building blocks of the discrete weak formulation are the bases of V_h and Q_h, as in (2.14). With $\mathbf{V}_h$ let us denote the Cartesian product $V_h \times V_h \times V_h$. A basis of this space is formed by elements of the structure

$$\varphi_i^{(1)} = \begin{pmatrix} \varphi_i \\ 0 \\ 0 \end{pmatrix}, \ \varphi_i^{(2)} = \begin{pmatrix} 0 \\ \varphi_i \\ 0 \end{pmatrix}, \text{ and } \varphi_i^{(3)} = \begin{pmatrix} 0 \\ 0 \\ \varphi_i \end{pmatrix},$$

Equipped with this notation, a basis of $\mathbf{V}_h \times Q_h$ is formed by

$$\left\{ \begin{pmatrix} \varphi_i^{(d)} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \psi_j \end{pmatrix} \right\} \quad d = 1, 2, 3, \ i = 1, ..., N, \ j = 1, ..., M. \tag{2.17}$$

The matrix $\mathscr{A}$ gained by inserting the elements of this basis into the weak formulation exhibits a 4-by-4 block structure. The system of equations will have a block structure like

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & B_1^T \\ A_{21} & A_{22} & A_{23} & B_2^T \\ A_{31} & A_{32} & A_{33} & B_3^T \\ B_1 & B_2 & B_3 & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ p \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ 0 \end{pmatrix}. \tag{2.18}$$

The A blocks stand for the coupling of velocity test- and ansatz functions, the B blocks for the coupling of pressure and velocity, the C block for the coupling of pressure and pressure[4] – in our case it is just 0. The transposed relation between the B-blocks is the discrete counterpart of the adjoint relation of $-\nabla\cdot$ (divergence) and ∇ operator, see e.g. (Braess, 1997, p.143).

Next, we will demonstrate for the convective term, how the entries of the matrices are derived. Testing the convective term in the momentum equation of Equation (2.2.1) with a basis test function of the type $\begin{pmatrix} \varphi_j^{(d)} \\ 0 \end{pmatrix}$, fixed d and j, gives the expression (we omit the factor $-\tau$)

$$\left((\boldsymbol{w} \cdot \nabla \boldsymbol{u}), \begin{pmatrix} \varphi_j^{(d)} \\ 0 \end{pmatrix} \right). \tag{2.19}$$

Writing it more detailed and developing the unknown $\boldsymbol{u}$ into the basis, we get from this

$$(2.19) = \left(\left(w_1 \frac{\partial}{\partial x_1} + w_2 \frac{\partial}{\partial x_2} + w_3 \frac{\partial}{\partial x_3} \right) \begin{pmatrix} \sum_{i=1}^N u_i^{(1)} \varphi_i \\ \sum_{i=1}^N u_i^{(2)} \varphi_i \\ \sum_{i=1}^N u_i^{(3)} \varphi_i \end{pmatrix}, \begin{pmatrix} \varphi_j^{(d)} \\ 0 \end{pmatrix} \right).$$

[4]The pressure-pressure coupling is not present for our chosen discretization, but is apparent for certain stabilizing discretization, as, e.g., PSPG.

Evaluating the scalar product yields that terms stemming from the d-th component are non-zero. Thus we can advance

$$(2.19) = \sum_{i=1}^{N} u_i^{(d)} \left(\left(\sum_{e=1}^{3} w_e \frac{\partial}{\partial x_e} \varphi_i \right), \varphi_j \right), \tag{2.20}$$

where the scalar product is from $L^2(\Omega)$:

$$(2.19) = \sum_{i=1}^{N} \left(\int_\Omega \sum_{e=1}^{3} \left(w_e \frac{\partial}{\partial x_e} \varphi_i \varphi_j \right) \mathrm{d}\mathbf{x} \right) u_i^{(d)}. \tag{2.21}$$

From this expression we can see that the convective term is responsible for filling the blocks A_{11}, A_{22} and A_{33} of $\mathscr{A}$ with non-symmetric (in i and j) integral expressions $\int_\Omega \sum_{e=1}^{3} (w_e \frac{\partial}{\partial x_e} \varphi_i \varphi_j)\ \mathrm{d}\mathbf{x}$. In assembling $\mathscr{A}$ efficiently, integral expressions of this type are evaluated using quadrature formulas elementwise. Since for most finite elements the basis functions are elementwise polynomial, that evaluation is exact.

Let us make some further remarks from a computational point of view. From that perspective, the Navier–Stokes equations contain "good" and "bad" terms. The "good" terms are those that are due to time discretization and the viscous term. Both contribute symmetrically to the diagonal A blocks, which is beneficial for many solvers. From the convective term comes a non-symmetric contribution, which is less favorable. The off-diagonal A blocks would only be filled if we had chosen a different form of the convective term, see, e.g., (John, 2016, pp.285). Finally, note that all blocks are only sparsely filled with entries. This is due to the local character of the integral expressions, stemming from the localization of FE basis functions. The sparsity of the system and its saddle point structure are two features which sparked a lot of research effort of effective solvers for such linear systems. In the following section, we take a closer look at a selection of such solvers.

2.3.2 Solvers for saddle point problems

In this section we want to present some strategies for solving linear systems of the saddle point type (2.18), and compare their efficiency in different areas of application. This chapter contains material which has been published in Ahmed et al. (2018).

We will start from the same system as we do in Ahmed et al. (2018). Let us recast System (2.18) as

$$\mathscr{A} := \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \bar{\boldsymbol{u}} \\ \bar{p} \end{pmatrix} = \begin{pmatrix} \bar{\boldsymbol{b}} \\ 0 \end{pmatrix}, \tag{2.22}$$

which we want to call the generic discrete saddle point problem. The zero block is due to using inf-sup stable finite element spaces, which means that no stabilizing pressure-pressure coupling has to be applied.

In (Elman et al., 2005, p.285) it is stated that the distinct feature of System (2.22) is its indefiniteness. This indefiniteness is *fundamental*, meaning

that it does neither vanish nor degrade when refining the mesh. As with many features of the NSE, this is already encountered when dealing with the Stokes equations. The Navier–Stokes equations bring as further difficulty the non-symmetry of the A block and therefore the entire matrix, which is caused by the convective term (see Section 2.3.1). These two features – indefiniteness and non-symmetry, disable the use of several standard solvers for linear systems. Furthermore, both Stokes and Navier–Stokes equation cause the lower right zero block when spatially discretized with inf-sup stable elements. This causes zero entries on the diagonal, meaning that further standard preconditioners, like the Jacobi or Gauss-Seidel iteration, cannot be applied. It is therefore necessary to look out for specialized solver alternatives. It is especially worthwhile to find fast specialized solvers, because in a CFD finite element simulation the solving of the linear system of equations is often, especially in 3d problems, the most time consuming part.

In general, solvers for linear systems of equations can be grouped into the two main classes of *direct* and *iterative* methods. Direct methods compute an LU factorization of $\mathscr{A}$, and thus transform the initial system to a system with a triangular matrix, which can be solved easily. Iterative methods, on the other hand, use an initial guess for a solution and define an iterative procedure, in which new solution approximations are computed. A broad class of iterative solvers are *Krylov subspace* methods. In a Krylov subspace method a series of stacked subspaces of the solution space is created iteratively, and the solution is searched within those subspaces.

A definite advantage of direct solvers is that the factorization of $\mathscr{A}$ can be stored and reused to solve the same system with different right-hand sides again and again. An advantage of iterative methods is that one can fix an accuracy up to which the system should be solved, e.g., in terms of the residual, and stop the procedure when that accuracy is reached. It is a characteristic feature of iterative solvers that they bring up sub-systems and subproblems, to whose solution one can again choose among direct and iterative solvers, and exploit either of the two stated advantages, as we shall see soon.

Roughly speaking, direct solvers perform best for small and medium sized problems. The larger the problem is, the more likely it is that an iterative solver will be the faster alternative.[5]

As for iterative solvers, in order to develop their full potential, it is necessary to provide the solver with additional information about the problem to be solved. This process is known as *preconditioning*, we will briefly explain the general idea. Let us write System (2.22) in shorthand as

$$\mathscr{A}\bar{\boldsymbol{x}} = \bar{\boldsymbol{b}}. \tag{2.23}$$

Then one way to "inform" the solver about the problem, consists in multiplying equation (2.23) from the left with a fitting square matrix $\mathscr{P}^{-1}$, thus letting the

[5] What "small" and "medium sized" mean here is totally problem or problem class dependent. It is just a reliable perception that at some point iterative solvers become more efficient, due to their better asymptotic scaling.

solver deal with

$$\mathscr{P}^{-1}\mathscr{A}\bar{\boldsymbol{x}} = \mathscr{P}^{-1}\bar{\boldsymbol{b}}$$

instead of the original problem. This approach is denoted by *left preconditioning*. In the other strategy, *right preconditioning*, one inserts $\mathscr{P}^{-1}\mathscr{P}$ between $\mathscr{A}$ and $\bar{\boldsymbol{x}}$, and then solves the two-stage problem

$$\begin{aligned}\mathscr{A}\mathscr{P}^{-1}\bar{\boldsymbol{y}} &= \bar{\boldsymbol{b}},\\ \mathscr{P}\bar{\boldsymbol{x}} &= \bar{\boldsymbol{y}}.\end{aligned}$$

If the *preconditioner* $\mathscr{P}$ is chosen appropriately, a clustering of the eigenvalues is obtained, which is favorable for Krylov subspace methods (Elman et al., 2005, p.177), leading to faster convergence.

In approximating $\mathscr{A}^{-1}$ the inverse of the preconditioner carries information about the system to the solver. A "perfectly informed" preconditioner would be the matrix $\mathscr{A}$ itself. As the calculation of $\mathscr{A}^{-1}$ is in general by far too costly, and would render the Krylov method wrapped around it utterly useless, $\mathscr{P}^{-1}$ must instead be constructed by cleverly exploiting the structure of the underlying problem. It has to fulfill two opposing demands: $\mathscr{P}$ should combine a good approximation to $\mathscr{A}$ with a feasible computational effort when applying its inverse.

To do both of these demands justice, it is necessary to develop preconditioners that are specifically tailored to a certain class of problems and are able to exploit their distinct properties. In the case of saddle point problems like (2.22), which stem from the Navier–Stokes equations, especially the natural block-wise composition of the system matrix and its origin from a finite element discretization can be made beneficial.

In our work Ahmed et al. (2018) we compared two such tailored preconditioners. Those are the Least Squares Commutator preconditioner (LSC) of Elman et al. (2007); Elman and Tuminaro (2009) and the coupled geometric multigrid preconditioner with specialized smoothers. Those preconditioners stand representative for two classes of NSE preconditioners, which one might call split and coupled methods. Split methods like the LSC exploit the saddle point structure of $\mathscr{A}$ by treating the block rows belonging to momentum and continuity equation separately. Coupled methods treat the entire matrix at once. In Ahmed et al. (2018) different variants and setups of both LSC and geometric multigrid were assessed in terms of computing time, applied to different versions of a CFD benchmark problem. Complementary, the wide-spread package UMFPACK was included in the assessment, as a representative of the direct solvers. Direct solvers are often used as "black box" solvers, therefore it was interesting to compare it with more specialized methods.

All solvers were applied to a well established benchmark problem known as *flow around cylinder* example, see Turek and Schäfer (1996), where it was introduced. In this example a stationary or weakly time-dependent flow of a viscous incompressible fluid through a rectangular channel, streaming around a cylindrical obstacle has to be computed. There is a 3d and a 2d version of

this example, where the 2d version features a plan view of the channel. See Figure 2.1 for pictures of our a priori meshes for the problem. Those should also convey an idea of the problem setup. For the time-dependent problem the solution possesses a prominent flow structure, known as Kármán vortex street, see Figure 2.2.

Both problems, 2d and 3d, possess a stationary and an instationary version that are accepted as benchmark problems. In our comparison we regarded all four combinations. In the instationary case we solved the full time-dependent Navier–Stokes equations (2.8), using basically the same spatial and temporal discretization techniques as described in Section 2.2. In the stationary case, their stationary analogon was solved.

In all problem settings, four different finite element discretizations with inf-sup stable finite element pairs were used. The idea was to try out one approximation with a continuous pressure approximation and one with a discontinuous first order pressure approximation on both a triangular (tetrahedral) and quadrilateral (hexahedral) grid. The velocity spaces were then chosen accordingly, such as to guarantee inf-sup stability. The problem sizes were varied by gradually refining the a priori grids uniformly. This allowed for a variation of the number of degrees of freedom and thus illustrated the asymptotic behavior of the different solving strategies and offered some insight on application areas.

Let us in the following describe the two preconditioning strategies, which were in the focus of the comparison in Ahmed et al. (2018). Both were used as preconditioners for the FGMRES method, which is a popular iterative method and belongs to the class of Krylov subspace methods. It was introduced in Saad (1993). FGMRES can be used for any type of matrix, regardless of its symmetry (other than, e.g., the MINRES method) or even its definiteness (as, e.g., the CG method), which makes it suitable for the indefinite, non-symmetric Problem (2.22). Its advantage over its predecessor GMRES is that it can be used with a different preconditioner in every iteration. This enables the use of iterative methods for subproblems of the preconditioner, from which both LSC and multigrid preconditioning benefit. The drawback of FGMRES is that it needs double the amount of memory, compared to GMRES.

After the description of the preconditioning strategies, we will give a discussion of some of the results of Ahmed et al. (2018).

Least-squares commutator preconditioner The LSC preconditioner is derived from the LU decomposition of the matrix $\mathscr{A}$ and the approximation of the *pressure Schur complement* by keeping a certain operator commutator error small. A complete and self-contained introduction can be found in the textbook Elman et al. (2005), here we give our own reformulation of that introduction.

Let us start with a formal block-wise Gaussian elimination of $\mathscr{A}$ from (2.22). This gives the LU decomposition

$$\mathscr{A} = \begin{pmatrix} I & 0 \\ BA^{-1} & I \end{pmatrix} \begin{pmatrix} A & B^T \\ 0 & -BA^{-1}B^T \end{pmatrix} = LU. \tag{2.24}$$

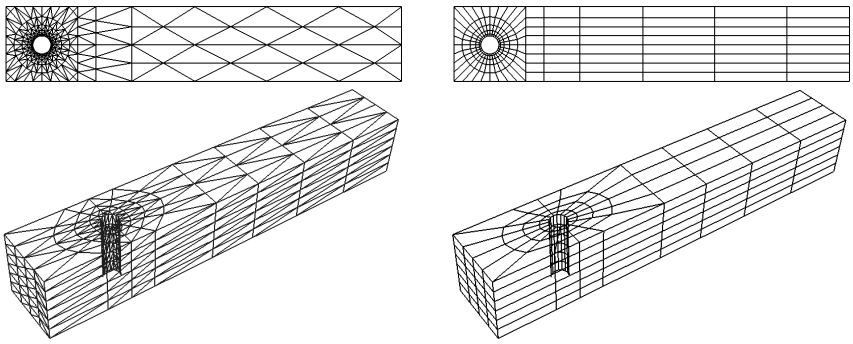

Figure 2.1: Initial grids of the flow around cylinder example. 2d triangles (upper left), 2d quadrilaterals (upper right), 3d tetrahedra (lower left), and 3d hexahedra (lower right).

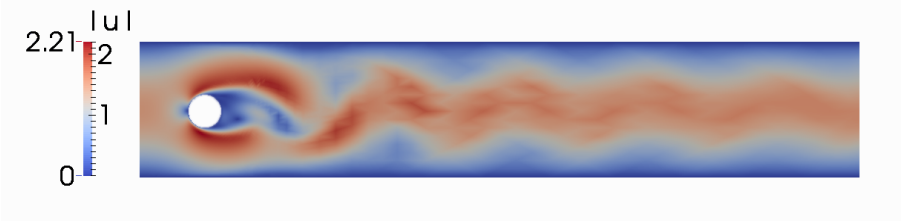

Figure 2.2: Flow around cylinder example, 2d instationary, solution snapshot. At intermediate Reynolds number a characteristic flow pattern, the Kármán vortex street, develops behind the cylindric obstacle.

The lower right matrix block is the *Schur complement* S of $\mathscr{A}$,

$$S := -BA^{-1}B^T.$$

Since from (2.24) it follows that $\mathscr{A}U^{-1} = L$, which has perfectly clustered eigenvalues, the upper triangular factor U is a good starting point for building preconditioners. Its drawback is the appearance of the Schur complement, which is not explicitly available and even if this would be the case, it is a dense matrix, since A^{-1} is not sparse in general. The difficulty to construct a better computable approximation to the Schur complement is addressed by the LSC preconditioner.

The basic idea of the LSC preconditioner is to look for a regular matrix $A_p \in \mathbb{R}^{m\times m}$ acting on (coefficients of) the pressure space that solves the equation

$$B^T A_p = AB^T \tag{2.25}$$

and thus gives, by transforming Equation (2.25) equivalently and multiplying with B from the left,

$$-BA^{-1}B^T = -BB^T A_p^{-1}. \tag{2.26}$$

The right-hand side of Equation (2.26) is a more convenient form of the Schur complement. For this form, applying U as a preconditioner requires approximating the action of $\left(-BB^T A_p^{-1}\right)^{-1}$, which is more easily done now since A_p is known and BB^T is positive definite and symmetric.

The remaining difficulty is that B^T is a full rank rectangular matrix and so Equation (2.25) is in general an overdetermined system and can only be solved in a minimizing sense

$$\min_{A_p} \left\| AB^T - B^T A_p \right\|, \tag{2.27}$$

with some matrix norm $\|\cdot\|$ that is to be defined.

One proceeds by recollecting the origin of the matrices in Equation (2.27) as discrete counterparts of the underlying continuous operators from the Navier–Stokes equations. In fact the matrix B^T stems from the finite element discretization of the gradient operator and the matrix A from a convection-diffusion operator acting on the velocity space,

$$-\nu\Delta + \boldsymbol{u}^m \cdot \nabla.$$

The unknown matrix A_p is further assumed to originate from the discretization of a hypothetical convection-diffusion operator acting on the pressure space. Problem (2.27) can then be interpreted as minimizing the discrete commutation error of velocity and pressure convection-diffusion operator with the gradient operator. To foster this interpretation, one has to account for the concrete choice of the finite element spaces and to introduce appropriate weights by multiplying with the inverses of the velocity and pressure mass matrices $Q \in \mathbb{R}^{n\times n}$ and $P \in \mathbb{R}^{m\times m}$.

One now replaces (2.27) by the minimizing problem

$$\min_{A_p} \left\| Q^{-1}AQ^{-1}B^T - Q^{-1}B^T P^{-1} A_p \right\|. \tag{2.28}$$

Observe that by multiplication from the left with $BA^{-1}Q$ and from the right with $A_p^{-1}P$ the right term inside the norm gives rise to a formula for the approximation of the Schur complement S:

$$S = -BA^{-1}B^T \approx -BQ^{-1}B^T A_p^{-1} P =: S_{\mathrm{LSC}}. \tag{2.29}$$

The last ingredient of the LSC is to specify the minimization problem (2.28) as minimizing columnwise in a Q-weighted vector norm

$$\|v\|_Q = \langle Qv, v\rangle^{\frac{1}{2}}.$$

This choice leads to the eponymous least squares problems

$$\min_{[a_p]_j} \left\| [Q^{-1}AQ^{-1}B^T]_j - Q^{-1}B^T P^{-1}[a_p]_j \right\|_Q, \quad j = 1, \ldots, m,$$

where the unknowns $[a_p]_j$ are the columns of A_p. The first order optimality conditions read

$$P^{-1}BQ^{-1}B^T P^{-1}\,[a_p]_j = \left[P^{-1}BQ^{-1}AQ^{-1}B^T\right]_j, \quad \forall j \in \{1, \ldots, m\}.$$

In this way, one obtains the representation

$$A_p = P\left(BQ^{-1}B^T\right)^{-1}\left(BQ^{-1}AQ^{-1}B^T\right). \tag{2.30}$$

The LSC Schur complement is finally obtained by replacing Q^{-1} with the inverse of the diagonal of Q, $D_Q^{-1} := (\mathrm{diag}(Q))^{-1}$ in (2.30) and inserting the arising formula into (2.29) :

$$S_{\mathrm{LSC}} = -\left(BD_Q^{-1}B^T\right)\left(BD_Q^{-1}AD_Q^{-1}B^T\right)^{-1}\left(BD_Q^{-1}B^T\right). \tag{2.31}$$

This expression approximates the lower right block in (2.24), i.e., the Schur complement S.

One step of application of the LSC is given in pseudocode in Algorithm 1. Note that in the application of the preconditioner, two pressure Poisson type problems have to be solved (Steps 2.1 and 2.3) by inverting the first and last term in parentheses in Equation (2.31). Additionally, a sub-system for the velocity has to be solved (Step 3.2), inverting the upper left matrix in (2.24). For these subproblems, one can exploit the aforementioned advantages of each solver class.

For the pressure subproblem we used a direct solver, which is especially useful in the time dependent case. The system is relatively small, compared to the entire system, and its entries do not depend on time. Therefore it is possible to re-use the once computed inverse of the system in all time steps, which gives a great efficiency advantage over iterative methods. For the solving of the velocity subsystem, the authors of Elman et al. (2005) distinguish between an "ideal" and an "iterative" version of the LSC preconditioner (and its predecessor, the PCD preconditioner). For the ideal version one uses a direct solver for the velocity subproblem, which gives fast convergence of the outer iteration, but is comparably slow, since the velocity subproblem is hardly smaller than the entire problem. The iterative version is way more efficient. Here, the velocity subproblem is solved with an iterative routine which is suitable for non-symmetric matrices. In (Elman et al., 2005, pp.359) the authors recommend to use a geometric or algebraic multigrid method for the velocity subproblem. With that or any other suitable iterative method, it is sufficient to solve the subproblem with relatively low accuracy only or perform just a fixed number of iterations and nevertheless experience fast convergence of the outer iteration. The Algorithm 1 must then receive an initial solution $(\boldsymbol{u}_0, p_0)$ as additional input.

Finally, note that the use of iterative methods for a subproblem, if not applied with a fixed number of iterations, necessitates an outer iterative method which allows for flexible preconditioning – as does the already mentioned Krylov subspace method FGMRES.

Geometric multigrid methods The other class of preconditioners which we assessed in Ahmed et al. (2018) were geometric multigrid methods. Originally developed as a solver framework, they showed to develop their full potential when used as preconditioners. Multigrid methods were extraordinarily popular during the 1990s and early 2000s, a standard monography is Hackbusch (2003). The methods show asymptotically optimal behavior and are well applicable to

Algorithm 1 Least Squares Commutator Preconditioner

Input: $\mathscr{A} = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}$, $\boldsymbol{b} = \begin{pmatrix} b_u \\ b_p \end{pmatrix}$, velocity mass matrix M_v

Output: Approximate solution $\boldsymbol{x}_{\mathrm{LSC}} = \begin{pmatrix} u \\ p \end{pmatrix}$ to $\mathscr{A}\boldsymbol{x} = \boldsymbol{b}$

Part 1 Set up Poisson matrix P

1.1 : $D_v^{-1} \leftarrow (\operatorname{diag} M_v)^{-1}$

1.2 : $P \leftarrow \left(BD_v^{-1}B^T\right)$

Part 2 Schur complement subsystem

2.1 : Solve $Px = b_p$ and update $b_p^* \leftarrow x$ ▷ First Poisson solve

2.2 : $b_p^* \leftarrow -BD_v^{-1}AD_v^{-1}B^Tb_p^*$

2.3 : Solve $Px = b_p^*$ and update $p \leftarrow x$ ▷ Second Poisson solve

Part 3 Velocity subsystem

3.1 : $b_{u,\mathrm{tmp}} \leftarrow b_u - B^Tp$

3.2 : Solve $Ax = b_{u,\mathrm{tmp}}$ and update $u \leftarrow x$ ▷ Velocity solve

return (u, p)

academic problems as the flow around cylinder example, where a hierarchy of spatial grids is easily obtained.

Given such a hierarchy of grids, the general idea of multigrid methods is to damp high frequency error contributions on fine grids and damp the low frequency error contributions on coarse grids, where they appear as high frequency contributions. This grid-wise damping is achieved by applying one or more steps of another iterative method, a process known as *smoothing*. The choice of that iterative method, the *smoother*, is key to the efficiency of the method for a certain problem or problem class. On the coarsest grid, where the problem is typically small, often a direct solver can be applied. Passing information between the grids is performed by grid transfer operators, e.g., L^2-projection. As we listed in (Ahmed et al., 2018, p.496), in order to define a certain geometric multigrid method, one has to specify the following constituents:

- the grid hierarchy,
- the grid transfer operators, i.e., restriction and prolongation,
- the grid cycle, i.e., the sequence in which the levels of the grid hierarchy are addressed,
- the smoother, i.e., an approximate solver on levels which are not the coarsest one,
- the solver on the coarsest grid.

In Ahmed et al. (2018), we made use of a grid hierarchy gained by a successive uniform refinement of the initial grid. As grid transfer operators we used L^2-projections that are described in Schieweck (2000). The employed grid cycle was the F-cycle, which is a hybrid between the standard V- (one recursive call) and W-cycle (two recursive calls). As smoothers we used several versions

of a block Gauss–Seidel method known as *Vanka* smoother, see Vanka (1986) for their original introduction. Smoothers of that class are, to our experience, the most efficient smoother option for coupled multigrid preconditioning of the Navier–Stokes equations.

Results of the assessment and discussion In this paragraph we want to briefly present and discuss the most important findings of our work Ahmed et al. (2018). There, the solving strategies presented above were applied to the flow around cylinder benchmark problem, both in 2d and 3d, stationary and instationary. The 3d instationary example needed a slight modification of the inflow condition in order to exhibit a truly instationary behavior. All discretizations were performed using the same or similar finite element techniques as described earlier in this chapter. A Picard iteration was used to resolve the non-linearity coming from the Crank–Nicolson time discretization, and it had to reduce the residual below a threshold of 10^{-8} in the Euclidean norm. We used four different finite elements both in 2d and 3d, those were P_2/P_1 and $P_2^{\text{bubble}}/P_1^{\text{disc}}$ on triangular/tetrahedral grids as well as Q_2/Q_1 and Q_2/P_1^{disc} on quadrilateral/hexahedral grids. In the stationary case, the most interesting question was, how the different solvers behave with respect to computing time when refining the grid and thus raise the number of degrees of freedom of the problem. In the time-dependent case we were more interested in the dependence of computing time on the time step length.

Proceeding as such, we could identify use cases for the different solvers, and give some advice on how to "fine tune" the solvers, regarding the many versions and parameters with which iterative solvers can be tweaked and tuned.

The direct solver UMFPACK was used as a black box solver and served as a reference point. The LSC preconditioner was implemented and tested in a standard version and in the version of Elman and Tuminaro (2009), the "boundary corrected LSC". We also compared the application of a direct (UMFPACK) and an iterative (BiCGstab with SSOR preconditioning) solver as sub-solver for the velocity subproblem. The multigrid preconditioner was tested in two versions, a standard approach and the "multiple discretization multilevel" approach of John et al. (2002). Both were used with three different Vanka smoothers: the *Cell Vanka smoother* for discontinuous-pressure type discretizations, the *Patch Vanka smoother* for continuous-pressure type discretizations and the *Nodal Vanka smoother* for both types.

The methods were implemented in the integrated research code ParMooN (Ganesan et al. (2016); Wilbrandt et al. (2017)) and executed sequentially on HP BL460c Gen9 workstations with 2x Xeon CPUs (2600 MHz clock rate).

Let us now describe our findings. When it comes to the stationary problem, we found that the FGMRES + LSC preconditioner is the best choice for continuous pressure approximations and the multigrid preconditioner with Cell Vanka smoother the fastest choice for discontinuous approximations. See Figure 2.3 for a representative illustration of the computing times with a discontinuous pressure finite element pair in 2d. With discontinuous pressure the Cell Vanka smoother can be used, and this strategy showed to be superior to both the

nodal and the patch Vanka. The LSC strategy was not competitive here, but for continuous pressure and small or medium sized problems it was the fastest option, only for the larger problems did the multigrid with Nodal Vanka take the lead. The version of the LSC with iterative solver for the velocity subproblem did not seem a good choice in the stationary case, as it was hard to make that solver converge, and if that succeeded, it was usually slower than the version with the direct solver. In the stationary case, the LSC showed the same asymptotic behavior as the UMFPACK direct solver. The UMFPACK solver was superior only for the smallest problems, but it offers a good starting point for checking implementations and debugging due to its robustness.

In the stationary 3d case the findings did very much conform with those in stationary 2d, with the additional observation that the UMFPACK solver could only handle the one or two smallest problems, and the largest problems (3 levels of uniform refinement, around 10^7 to $3 \cdot 10^7$ d.o.f.) could only be solved with the multigrid approaches.

The solvers showed a somewhat different behavior for the instationary problems. In 2d, especially the FGMRES + LSC(ite) strategy excelled, because contrary to all other candidates, the *total* computing time actually decreased, when choosing smaller time-steps. Though already among the fastest options for (the coarsest) time step 0.01, it was unbeatable for the smallest examined time step 0.0025, save the $P_2^{\text{bubble}}/P_1^{\text{disc}}$ element, were FGMRES + standard multigrid with Cell Vanka smoother was always superior. We attributed this behavior of the LSC(ite) to the better properties of the A-block of the matrix due to a smaller time-step, when the impact of the mass matrix is greatest. Obviously, the LSC(ite), especially the BiCGstab iteration used for the velocity sub-solve, profited most from this circumstance. On the Q_2/Q_1 Taylor–Hood discretization, the time advantage of LSC(ite) was the most remarkable. In 3d, the picture was essentially the same (see Figure 2.4). Except for the $P_2^{\text{bubble}}/P_1^{\text{disc}}$ element, were Cell Vanka multigrid performed best, the LSC(ite) strategy was superior to all others, showing a decrease in total computing time with smaller time-step. For the largest time-step, the inner iteration did not converge, there either standard multigrid or LSC(dir) were the fastest choice.

To conclude our findings in the stationary case:

- for small and medium sized problems with continuous pressure discretization, the FGMRES + LSC(dir) approach is a recommendable option,
- for problems with discontinuous pressure discretization and large problems in general, the standard multigrid strategy with Cell Vanka smoother (discontinuous p) or Nodal Vanka smoother (continuous p) should be pursued,
- the UMFPACK solver should best be used as a solver for sub-systems of the iterative methods only, and not as a stand-alone solver.

In the instationary case we can conclude:

- the FGMRES + LSC strategy with an iterative solver for the velocity subproblem is the solver of choice for both 2d and 3d. If the iteration of

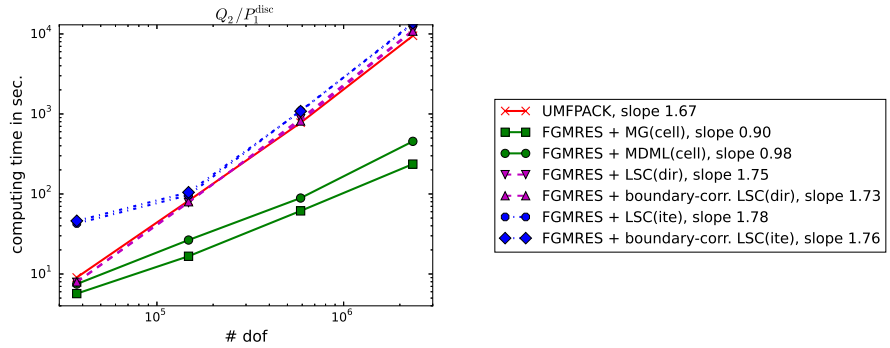

Figure 2.3: Steady-state flow around a cylinder in 2d: computing times on different grids and slope of best-fit line for Q_2/P_1^{disc} discretizations. Figure taken from (Ahmed et al., 2018, p.505).

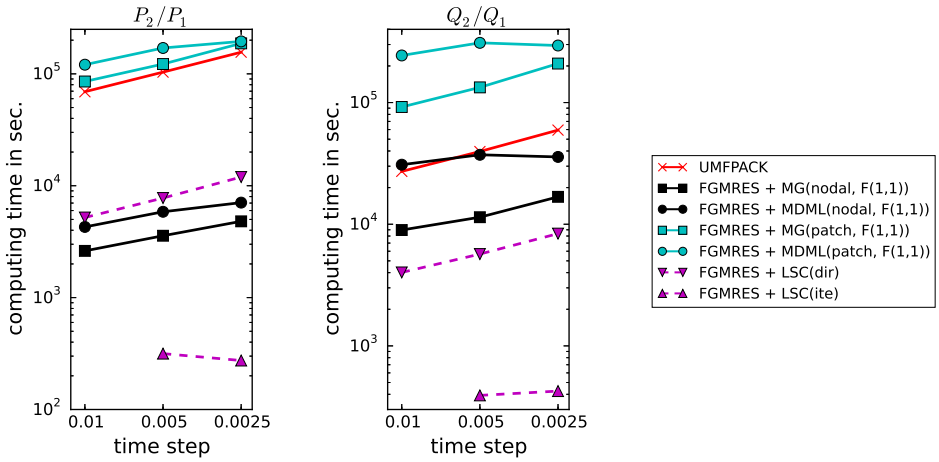

Figure 2.4: Instationary flow around cylinder in 3d: computing times on grid of first refinement level for different time step sizes. Only results for continuous-pressure discretizations are shown here. Figure taken from (Ahmed et al., 2018, p.510)

the velocity subproblem (BiCGstab) does not converge, it is often recommendable to choose a smaller time step rather than a different solver.

- For the (somewhat exotic) $P_2^{\text{bubble}}/P_1^{\text{disc}}$ element, the FGMRES + Cell Vanka multigrid strategy is a worthwhile, often the fastest, alternative.

With that we conclude our excursus to the performance of solvers for linear saddle point systems. Especially the findings on the instationary 3d problem will guide us later in this thesis. We included in Ahmed et al. (2018) a section containing some remarks on the parallelization and parallel performance of the solvers. In the following section, we want to describe in detail the parallelization of our own finite element code and the connected parallel solvers for linear systems.

2.4 A finite element domain decompositioning method

The solvers presented and assessed in Section 2.3 have in common that they were implemented and run in sequential execution, i.e., using one processor at a time. This is fair enough if one aims at assessing the efficiency of the algorithms themselves, but in order to catch up with the state of the art, it is necessary to regard parallelization of the linear solvers and the finite element method in general. In view of the capacities of super computers, their ever increasing number of processors, and the huge problem size which CFD examples reach easily, it is indispensable to use parallel code and parallelized linear solvers. For that purpose, we make use of the parallelized finite elements research code ParMooN. It is a subsequently parallelized version of the in-house finite elements package MooNMD (see John and Matthies (2004)), and it was presented in the articles Ganesan et al. (2016) and Wilbrandt et al. (2017). Most of the material of this section was published in Wilbrandt et al. (2017).

ParMooN supports a *single program, multiple data* (SPMD) approach on parallelism, making use of the Message Passing Interface (MPI) standard (see MPI-Forum (2015)). It relies on a decomposition of the domain, which is de-facto standard for parallelized finite element codes. Decomposing the computational domain and distributing it among the processes naturally leads to a parallelization of matrix-vector operations, if the matrix in question belongs to the finite element discretization of a partial differential equation. The localized character of the finite element method, reflected in the sparsity of the arising matrix, limits the computational overhead of communications.

In the following section we will present the domain decompositioning approach of ParMooN (Section 2.4.1), and how this can be turned to a parallelization of the finite element method (Sections 2.4.2 and 2.4.3). Additionally, we describe our parallelization of the LSC preconditioner in Section 2.4.4. This material was neither part of Wilbrandt et al. (2017) nor of Ahmed et al. (2018). Finally, we present some results on the parallel efficiency of ParMooN when applied to certain CFD problems in Section 2.4.5.

2.4.1 Decomposing the domain – own cells and halo cells

The first step of the parallelization is the cell-wise distribution of the computational domain among the participating processes. In MPI terminology, *process* denotes a stream of execution of a parallel program. All started processes execute the same code ("single program") and hold their own data ("multiple data"). The number of processes is in principle the user's choice. The processes are numbered with non-negative integers starting from 0, the process with number 0 is called the *root process* or just *root*, and it often gets assigned specific tasks.

ParMooN makes use of the METIS graph partitioning tool Karypis and Kumar (1995) for the domain decompositioning. At program start, all processes read in the same geometry and perform the same initial domain refinement steps. Upon reaching the first refinement level on which to perform computations, the root process calls the METIS library to compute a disjunct domain

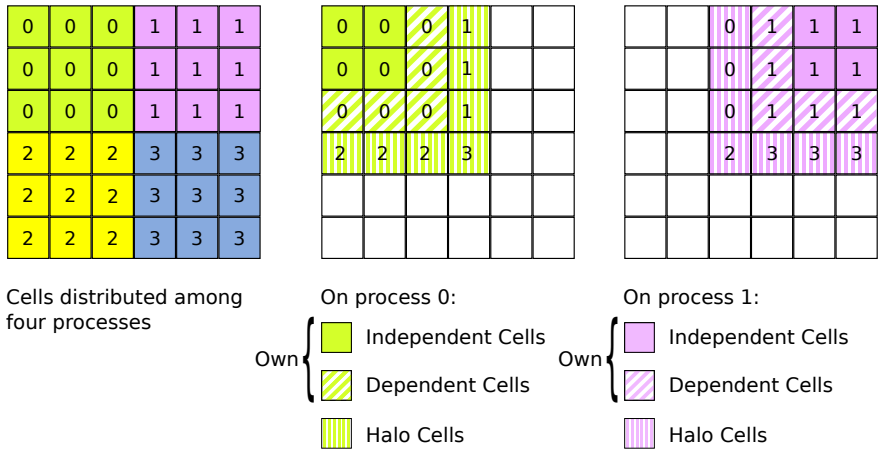

Figure 2.5: Different cell types due to domain decompositioning. The number inscribed into each cell identifies the process on which that cell is own cell. Figure taken from (Wilbrandt et al., 2017, p.79)

decomposition, i.e. to determine which process is going to be in charge of which mesh cells.

Next, root communicates the METIS output to the other processes. Each process is informed about which of the cells it will be responsible for. These cells are called *own* cells of a process. Each process then deletes a number of cells, maintaining only its own cells plus those cells which share a boundary face, edge or vertex with an own cell. In domain decompositioning methods these cells are commonly referred to as *halo* cells. A glance at Figure 2.5 might clarify that expression – the halo cells form a one-layer thick halo around the set of own cells.

The own cells are further divided into *dependent* and *independent* cells. The interface between halo cells and own cells is simply called the *interface*, and all own cells which share a piece of interface are called dependent, all other own cells are called independent cells.

The requirements on the domain decomposition are twofold: The computational load must be balanced (comparable number of cells on each process), and the needed amount of communication must be small (interface area as small as possible). Due to the deletion of cells, each process stores only a part of the entire problem (*multiple data*), but all process execute the same program code (*single program*). With its domain reduced to own cells plus halo, each process will set up a finite element space on that domain. Thus, one process can and will perform all further computations only on its known part of the domain. All tasks which are of global nature require communication between the processes. The organization of this communication is the subject of the following subsections.

2.4.2 Types of degrees of freedom

The domain decompositioning approach that is pursued in ParMooN naturally gives rise to a parallelization of those operations which are required when setting up and solving a finite element problem. A very important class among these are matrix-vector operations. During an introduction to the parallelization concept, we consider it helpful to keep matrix-vector multiplication, where the matrix stems directly from a finite element discretization, as an example in mind.

Firstly, the degrees of freedom (d.o.f.) on a certain process P need to be classified. Each d.o.f. will be assigned a class, depending on its localization in space and the classes of d.o.f. it couples with. Here, *coupling* of two d.o.f. means that their supports intersect on a set of non-zero measure, i.e., they belong to the same cell. This transfers directly into a property of the finite element matrix A: the coupling of d.o.f. i and j will lead to non-zero entries a_{ij} and a_{ji}. This connection is essential for the transfer of the domain decomposition into a parallelization of matrix-vector multiplication.

Next, we will list and shortly describe the d.o.f. classes in ParMooN. All d.o.f. that are localized in a cell known to P are called *known* d.o.f.. These are divided into *master* and *slave* d.o.f.. For a d.o.f. i, being a master d.o.f. on P means that P is responsible for the value of i – what that responsibility means exactly will be clarified below. Every d.o.f. in the entire problem is master on exactly one process. All known d.o.f. of P, which are not master on P are called *slave* d.o.f.. A d.o.f. can be slave on more than one process. The set of known d.o.f. on P is divided as

$$D^P_{known} = D^P_{master} \dot\cup D^P_{slave}.$$

The next level of classification, below the *master* and *slave* distribution, contains the classes of *independent*, *dependent*, *interface*, and *halo* d.o.f..

Independent d.o.f. All d.o.f. which lie in the own cells of P, but not in its dependent cells, are called *independent* d.o.f.. All P's independent d.o.f. are set as master d.o.f., they are not even known to any other process. They only couple to other master d.o.f. of P.

Dependent d.o.f. Those d.o.f. lying in P's dependent cells, but not in its halo cells, are called *dependent* d.o.f.. P is master of all its dependent d.o.f.. This denotation is motivated by the fact that dependent d.o.f. are in vicinity to the domain interface and therefore admit a certain dependency on other processes.

Interface d.o.f. All d.o.f. which lie on the intersection of dependent cells and halo cells, i.e., are located directly on the interface, are called *interface* d.o.f.. These d.o.f. are known to all adjacent processes as interface d.o.f., too. Only one of these processes will take master responsibility for a certain interface d.o.f.. On P, those interface d.o.f. which are master, are called *master interface* d.o.f., all others, for which neighboring processors take master responsibility, are called *slave interface* d.o.f. of P.

Halo d.o.f. All d.o.f. which lie in halo cells but not on the interface are called *halo* d.o.f.. Since all of them are dependent d.o.f. to neighboring processes, one of these will take master responsibility for them, on P all halo d.o.f. are slave d.o.f..

The above further classification of the d.o.f. is designed to yield a disjunct dissection of the classes master and slave d.o.f.:

$$D^P_{\text{master}} = D^P_{\text{independent}} \dot\cup D^P_{\text{dependent}} \dot\cup D^P_{\text{interface master}}$$
$$D^P_{\text{slave}} = D^P_{\text{interface slave}} \dot\cup D^P_{\text{halo}}.$$

Splitting halo d.o.f. and dependent d.o.f. It is convenient to refine the d.o.f. classification even one more step, in order to reduce the communication overhead of the program. The halo d.o.f. of P are further divided into *Halo(α)* d.o.f. and *Halo(β)* d.o.f.. Halo(α) d.o.f. are those, which couple with at least one (interface) master d.o.f. of P, while Halo(β) d.o.f. couple solely to other slave d.o.f. (interface d.o.f. and other halo d.o.f.).

A corresponding splitting of the dependent d.o.f. is performed. Those, which are connected to at least one (interface) slave d.o.f. are called *Dependent(α)* d.o.f., all those which are connected to master d.o.f. only (interface master, other dependent, independent) are called *Dependent(β)* d.o.f.. Note that all Dependent(β) d.o.f. of process P will be Halo(β) d.o.f. on all other processes where they are known. For Dependent(α) d.o.f. the matter is not as simple. Each of them is Halo(α) to at least one neighboring process, but can be Halo(β) to others. The relations are illustrated in Figure 2.6.

Note that this last level of classification described above does only make sense for problems containing only one finite element space. For a coupled problem like the Navier–Stokes equations it is not applicable, because the sets of interface masters and slaves will differ between both spaces, thus disabling a clear distinction between Halo/Dependent α and β d.o.f..

2.4.3 Operations, consistency and communication

Consistency levels There are basically two ways to store a global value distributedly in parallel computations. The first option is called *consistent* storage, the other is called *inconsistent* or *additive* storage. Consistent storage means that all processes which know a value also store it correctly, the value is the same over all processes which know it and the same as it would be in a sequential environment. We consider the term "inconsistent" storage rather deceptive, and will not use it. In our opinion, "additive" is much more to the point. In additive storage a global value is the sum of the values over all processes where it is known. In ParMooN, and in other parallel finite element codes which make use of a halo cell layer, mainly *consistent* storage and weakened concepts thereof play a role.

We will introduce now four stacked stages of consistency which can hold for finite element vectors in ParMooN, i.e., coefficient vector representations of functions from a finite element space. We call a finite element vector:

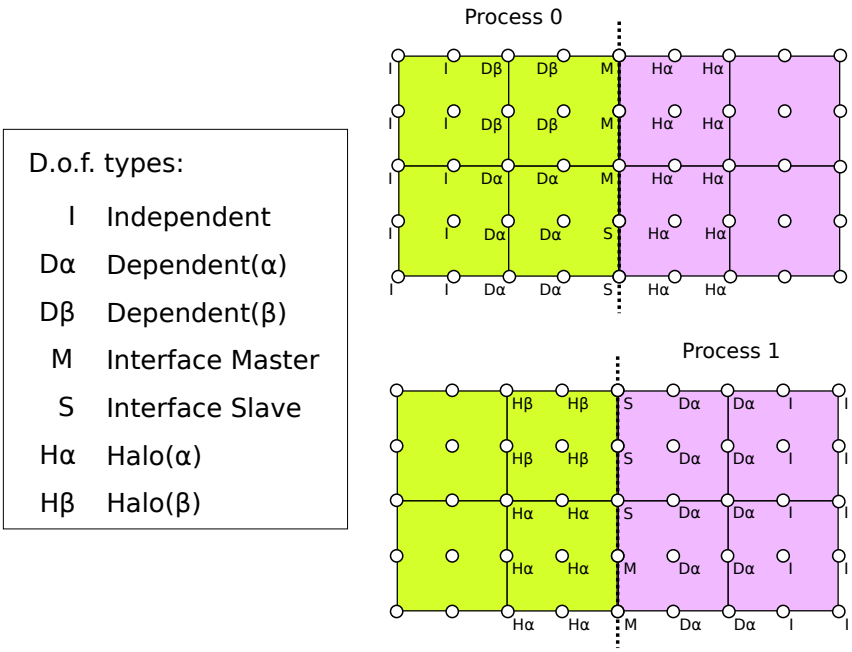

Figure 2.6: Types of degrees of freedom at the interface for Q_2 finite elements, from the view of process 0 and process 1. Figure taken from (Wilbrandt et al., 2017, p.80)

.

- *level-0-consistent*, if consistency holds only with regard to master d.o.f.. Each master d.o.f. on each process holds the same value as it would in a sequential computation. The values of slave d.o.f. are in an undefined storage state. Every parallel scalar-vector, vector-vector or matrix-vector operation must result in a finite element vector of at least this consistency level, otherwise the parallel implementation is faulty. During the implementation of such operations care must be taken of not losing level-0-consistency.

- *level-1-consistent*, if all master d.o.f. (level-0-consistency) and all slave interface d.o.f. are in consistent state. The values of all halo d.o.f. are in an undefined storage state.

- *level-2-consistent*, if consistency is established for all but the Halo(β) d.o.f. The values of Halo(β) d.o.f. are in an undefined storage state, while all other values are consistent.

- *level-3-consistent*, if all its d.o.f. are stored consistently. No d.o.f. values are in undefined storage state. This is the "actual" consistent storage state.

The main insight behind this classification is that in the presence of a halo cell layer, several algebraic operations have weaker consistency requirements of

their input data, than the full level-3-consistency. Restoring a certain state of consistency requires a certain amount of inter-process communication – the lower the required state of consistency, the lower the required amount of communication. Therefore, introducing the above categorization and updating always to the lowest possible consistency level will save some communication overhead and thus computing time.

Parallelizing algebraic operations Following the procedure of a finite element simulation, after the domain has been decomposed, each process assembles a finite element matrix on all its known cells. Maintaining the halo cell layers assures that all information to assemble the rows belonging to master d.o.f. is available on P. The complete finite element matrix will therefore be in a consistency state which one could term *row-wise level-1-consistency* – all rows belonging to master and slave interface d.o.f. are in consistent storage state.

Looking at matrix-vector multiplication of a row-wise level-1-consistent finite element matrix with a level-3-consistent finite element vector, the resulting finite element vector will be level-1-consistent.

Multiplication of a finite element vector with a consistent scalar will maintain the current consistency level, as will vector-vector addition. Scalar products require level-0-consistency of both vectors, where all slave d.o.f. will be skipped, and a globally additive reduce operation to get a consistent result.

Enforcing level-3-consistency of a finite element vector in ParMooN is always required when operations that require knowledge of the represented finite element function even on the halo cells is necessary. Such operations are for example matrix assembling with an input finite element function[6], grid transfer operations in multigrid, or gradient recovery. Level-3-consistency is also necessary for the input vector of a matrix-vector multiplication, if the matrix belongs to a coupled problem, see the remark in Section 2.4.2.

Enforcing certain consistency levels is a matter of communication. For each d.o.f. for which an update is necessary, the process where it is master on communicates its value to all processes where it is slave on, these simply reset its value to the received value. The required infrastructure is set up just once for a certain finite element space and can be reused whenever an update is necessary. In the next paragraph, we describe that communication infrastructure in some more detail.

Organizing communication When setting up the communication structure, for each non-independent master d.o.f. d, all those slave d.o.f. on other processes that are globally identical to d must be found. Certain master types match with certain slave types, forming three distinct pairs of master–slave relations. These relations are depicted in Table 2.1. To restore a certain consistency level, an update along the lines of one or more of these relations will be required.

[6] Think of the nonlinear term in the Navier–Stokes equations or the initial conditions in any time-dependent problem

Table 2.1: Master-slave relationship of d.o.f. types.

Relation (shorthand)	**Master type**	updates	**Slave type**
Interface (IMS)	Interface M.	$\longrightarrow$	Interface Sl.
Dependent(α)–Halo(α) (DHα)	Dependent(α)	$\longrightarrow$	Halo(α)
Dependent(β)–Halo(β) (DHβ)	Dependent(β)	$\longrightarrow$	Halo(β)

Note that it is not immediately possible to globally identify a d.o.f. in ParMooN, since each process creates its finite element space only across its own cells and numbers its d.o.f. locally, unaware of the other processes. To globally identify a d.o.f. in ParMooN we make use of a global *cell number* of a cell it is located in. Such a global cell number must be given to each cell before decomposing the domain, and is kept during the entire computation. When after decompositioning only uniform refinement steps are applied, a globally unique cell number can be given to children cells, too.[7] The consistent cell number and a likewise consistent cell-local d.o.f. number makes it possible to identify each d.o.f. globally.

Let us finish with some ParMooN-specific implementation details. The communication structure is separated into a data class "ParFEMapper", and a control class "ParFECommunicator". The process of setting up the ParFEMapper and ParFECommunicator requires some communication itself, this part is skipped here. We will just give an overview and description of those data fields of ParFEMapper which are relevant when updating the d.o.f. of a certain master-slave relation. These data fields are corresponding for all three relations (see Table 2.1), and we pick the interface (IMS) relation as an example.

For the IMS update, the ParFECommunicator wraps a call to the MPI function `MPI_Alltoallv`, whereby every process can send a different number of different values of the same type (`MPI_DOUBLE` in our case) to each other process. To control the `MPI_Alltoallv` call, the ParFEMapper stores the following data fields, where `mpi_size` is the total number of processes and `nInterfaceSlaves` is the number of interface slaves *local* to process P. The syntax of the listed data members is C-style, since ParMooN is a C++ code.

- `int* sendBufIMS`: The send buffer, filled with the values of all interface masters, each one possibly appearing more than once, which will then be sent to the other processes. Its total length equals the sum over all values of `sendCountsIMS`.

- `int* sendCountsIMS`: The send counts, an array of size `mpi_size`. Lists how many values P has to send to each other process.

- `int* sendDisplIMS`: The send displacement, array of size `mpi_size`. It lists, where in the array `sendBufIMS` the message for a certain process begins. For our purpose, we do neither work with overlap nor gaps, so

[7] A combination of parallelism and adaptive mesh refinement is not yet enabled in ParMooN.

`sendDisplIMS[i]` will simply hold the sum of `sendCountsIMS[0]` to `sendCountsIMS[i-1]`.

- `int* recvBufIMS`: The receive buffer, will be filled with sent values from the other processes in the communication routine. Its size equals `nInterfaceSlaves`.
- `int* recvCountsIMS`: The receive counts, of size `mpi_size`. It lists how many values are to be received from each process. The sum of its values will equal `nInterfaceSlaves`.
- `int* recvDisplIMS`: The receive displacement, analogous to send displacement. No gaps, no overlap.

Besides that data that is needed in the immediate control of `MPI_Alltoallv`, the ParFEMapper holds two arrays which allow to interpret the sent and received data, by mapping between send buffer or receive buffer and the local d.o.f.. These arrays are:

- `int* sentDofIMS`: Interpret `sentDofIMS[i]`$= d$ as: The i-th place in the send buffer `sendBufIMS` has to be filled with the value of local d.o.f. d.
- `int* rcvdDofIMS`: Interpret `rcvdDofIMS[i]`$= d$ as: The i-th value in the receive buffer `recvBufIMS` should update local d.o.f. d.

In the same manner the communication for DH(α) and DH(β) are organized. To set a certain level-0-consistent finite element vector into level-1-consistency, only a IMS update is required. For restoring level-2-consistency, an additional DH(α) update is necessary, and for level-3-consistency a DH(β) update on top.

2.4.4 Parallelization of the LSC preconditioner

Given the parallel data structure of ParMooN, the Least Squares Commutator preconditioner can be parallelized very efficiently. A parallelized version of the LSC algorithm known from Section 2.3.2 is given as Algorithm 2. There, at several places consistency updates of finite element vectors as described above are necessary, in order to maintain level-0-consistency after the matrix-vector multiplications.

The key issue in the parallelization is the computation of the pressure convection-diffusion matrix (Step 1.2). Remember that the matrix $B = (b_{ij})_{j\in\{1,\dots,3M\}}^{i\in\{1,\dots,N\}}$ is a finite element matrix, and therefore

$$b_{ij} \neq 0 \Leftrightarrow \text{d.o.f. } i \text{ and } j \text{ are in the same cell.}$$

Its (diagonally scaled) multiple with its own transposed from the right, $P = BD_v^{-1}B^T$, has a denser structure in general, since

$$\begin{aligned} p_{ij} \neq 0 &\Leftrightarrow \exists k \in \{1,\dots,3M\} \text{ s.t. } b_{ik} \neq 0 \text{ and } b_{kj} \neq 0 \\ &\Leftrightarrow \exists \text{ pressure d.o.f. } k \text{ which shares a cell with } i \text{ and a cell with } j. \end{aligned}$$

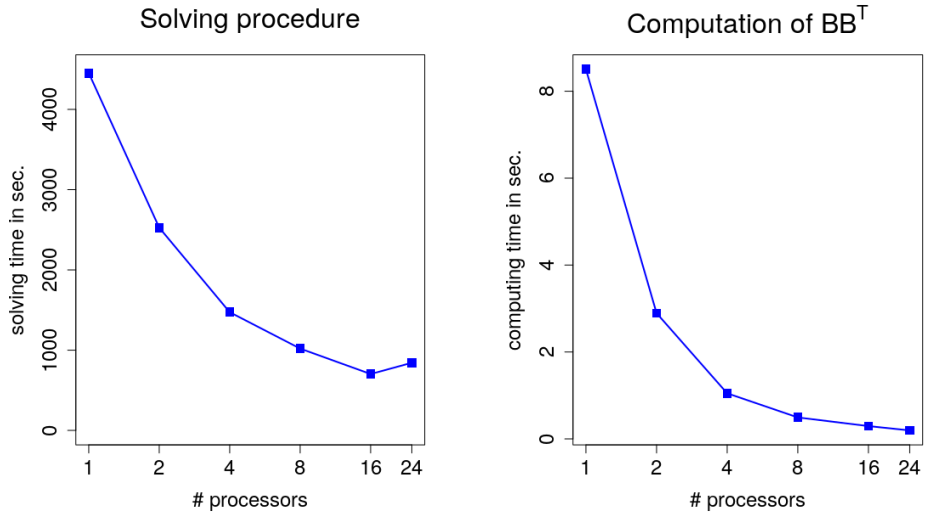

Figure 2.7: Total solving time (left) and time for the computation of $BD_v^{-1}B^T$ (right) depending on the number of processes for a parallelized version of the LSC preconditioner, applied to a 3d time-dependent Navier–Stokes example problem of approximately 125,000 d.o.f..

Therefore the matrix P does contain cell-wise next-to-nearest-neighbor interactions. This means especially that the parallelization strategy as described above, which was tailored to finite element matrices containing only nearest-neighbor interactions of finite element cells will not succeed for matrix-vector multiplications with the matrix P. Yet is it possible to compute an *additively* stored version of P. To show this quite plainly, it is for all i and j, disregarding the diagonal scaling,

$$p_{ij} = \sum_k b_{ik} b_{kj}. \tag{2.32}$$

The contributions $b_{ik}b_{kj}$ are correct on that process where k is master, due to the fact that B is a finite element matrix. If therefore each process adds only those contributions to its portion of P, where the "intermediary" d.o.f. is master, one finally finds the matrix P in additive storage. This matrix can then be handed over to the MUMPS solver, see Amestoy et al. (2001), an distributed memory direct solver, which supports additive storage of the system matrix.

As in the sequential case, for time-dependent problems one must compute the pressure Poisson matrix P only once, and can re-use its factorization for multiple applications of the LSC preconditioner.

The solver for the velocity subsystem must be parallelized, too. The ideal version of the preconditioner, using again the MUMPS solver for this system, showed to be useful for debugging purposes but not very efficient. For the iterative version, we achieved good results with a parallelized BiCGstab algorithm, preconditioned with a parallelized (block) SSOR sweep.

We include here some illustrative results of the performance of the parallelized version of the LSC preconditioner, see Figure 2.7. We chose a version

Algorithm 2 Parallelized Least Squares Commutator Preconditioner

Input: $\mathscr{A} = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}$, $\boldsymbol{b} = \begin{pmatrix} b_u \\ b_p \end{pmatrix}$ in consistency level 3, mass matrix M_v

Output: Approximate solution $\boldsymbol{x}_{\mathrm{LSC}} = \begin{pmatrix} u \\ p \end{pmatrix}$ to $\mathscr{A}\boldsymbol{x} = \boldsymbol{b}$

Part 1 Set up Poisson matrix P

1.1 : $D_v^{-1} \leftarrow (\operatorname{diag} M_v)^{-1}$

1.2 : $P \leftarrow \left(BD_v^{-1}B^T\right)$ ▷ Parallelized version.

Part 2 Schur complement subsystem

2.1 : Solve $Px = b_p$ and set $b_p^* \leftarrow x$ ▷ using MUMPS solver

2.2 : Update b_p^* to consistency level 3

2.3 : $b_u^* \leftarrow -D_v^{-1}B^T b_p^*$

2.4 : Update b_u^* to consistency level 2

2.5 : $b_u^* \leftarrow D_v^{-1} A b_u^*$

2.6 : Update b_u^* to consistency level 3

2.7 : $b_p^* \leftarrow B b_u^*$

2.8 : Solve $Px = b_p^*$ and set $p \leftarrow x$ ▷ using MUMPS solver

Part 3 Velocity subsystem

3.1 : Update p to consistency level 3

3.2 : $b_{u,\mathrm{tmp}} \leftarrow b_u - B^T p$

3.3 : Solve $Ax = b_{u,\mathrm{tmp}}$ and set $u \leftarrow x$ ▷ Parallel iterative solver

return (u, p)

of the time-dependent 3d Navier–Stokes problem of the batch crystallizer of Chapter 7, with approximately 125,000 degrees of freedom, and a time step length of 10^{-2} s. It is spatially discretized with 3d Taylor–Hood elements P_2/P_1, and the simulation was run for 10 s, starting from a zero initial solution and gradually increasing the inflow over the first 0.1 s. A Smagorinsky turbulence model with relatively large Smagorinsky coefficient of 10^{-2} is used. After 10 seconds of simulated time the inflow had reached the top of the crystallizer device, and we chose that time to end the illustrative assessment. We ran those computations with 1, 2, 4, 8, 16 and 24 processes, repeating each run five times, disregarding the fastest and slowest run and averaging over the remaining three runs. This averaged total solving time is plotted on the left side of Figure 2.7. Although initially a good parallel speedup can be achieved, the curve flattens out soon, with 24 processes the communication overhead consumes the speed-up and the computation is slower than with 16 processes. This behavior is to be expected for a relatively small problem like the studied one. Very interesting is the speedup achieved in an initial program part, the computation of the pressure Poisson matrix $BD_v^{-1}B^T$ Here, initially super-linear speed-up can be achieved, which is supposedly due to cache effects, Additionally, there is still an observable speed-up in this program part when comparing the run with 16 processes to that with 24 processes.

2.4.5 Parallel performance of ParMooN

The parallel scaling of certain other linear solvers implemented in ParMooN was examined and presented in Wilbrandt et al. (2017). Three different CFD benchmark problems of different type were solved, those were: a steady-state convection-diffusion problem, a time-dependent convection-diffusion problem and a steady-state incompressible Navier–Stokes system. In all cases, an FGMRES iteration with the "native" geometric multigrid method was compared to several parallel solvers from the PETSc library (Balay et al. (2017)) on a Q_2/Q_1 or Q_2/P_1^{disc} spatial discretization. Among the PETSc solvers were the FGMRES with SSOR or algebraic multigrid (BoomerAMG) preconditioning and the direct solver MUMPS.

The examinations were performed with from 2 to 24 processors. The problems had from 1 to 10 million d.o.f, and they had in common that the solvers showed good parallel scaling only up to 8 to 16 processors, as was to be expected the scaling was better for larger problems. While for the convection-diffusion problems the native multigrid solver was mostly outperformed by the SSOR preconditioned FGMRES of PETSc, for the stationary Navier–Stokes problem, the ParMooN FGMRES with multigrid preconditioning had a clear advantage over the external solver, here the PETSc implementation of FGMRES with LSC preconditioner. To give an idea of the type of results discussed in Wilbrandt et al. (2017), we include here the results for that stationary Navier–Stokes assessment in graphical form, see Figures 2.8 and 2.9. Both show solving time comparisons for the 3d stationary flow around cylinder example, which is described in Section 2.3.2. We used Q_2/P_1^{disc} elements, because the discontinuous pressure discretization enabled the use of the Cell Vanka smoother in the geometric multigrid preconditioner. Figure 2.8 shows the results of a solver time comparison for refinement level 2 (taking the hexahedral grid shown in Figure 2.1 as reference level 0), which yields a Navier–Stokes problem of around 0.9 million degrees of freedom. Figure 2.9 depicts the same for refinement level 3 (around 7 million d.o.f.), plus results on strong scaling for the ParMooN FGMRES with multigrid solver strategy. For this large problem relatively good scaling can be observed for up to 24 processors.

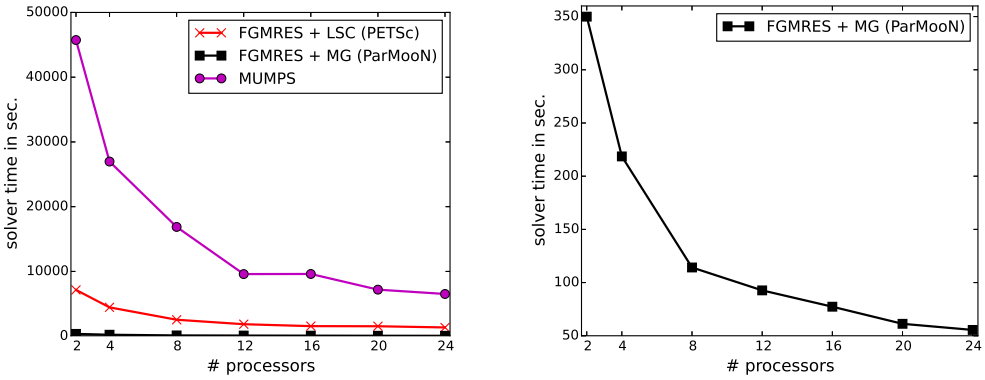

Figure 2.8: *Left:* Steady-state flow around cylinder example in 3d, solver times against number of processors with different parallelized linear solvers. Refinement level 2, around 0.9 million d.o.f.. *Right:* Closeup of solver time against number of processors for ParMooN FGMRES with geometric multigrid preconditioner. Figures taken from (Wilbrandt et al., 2017, p.87).

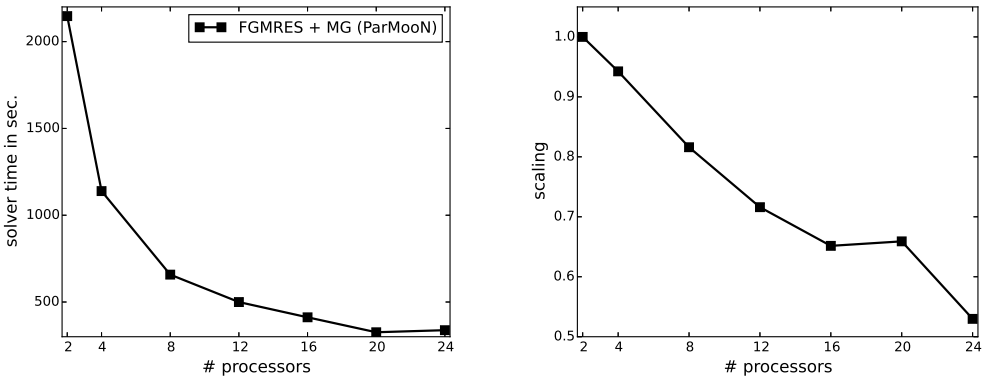

Figure 2.9: Steady-state flow around cylinder example in 3d, solver times (left) and scaling (right) on refinement level 3. The scaling is computed by $2 \cdot t_2/(p \cdot t_p)$, where p is the number of processors and t_p the corresponding time from the left picture. Figures taken from (Wilbrandt et al., 2017, p.87).

3 Numerical Methods for Convection-Diffusion-Reaction Equations

Time-dependent convection-diffusion-reaction equations (CDRE) are scalar second order partial differential of the generic type

$$\frac{\partial c}{\partial t} - \varepsilon \Delta c + \boldsymbol{u} \cdot \nabla c + r(c) = f \quad \text{in } (0, t_{\text{end}}) \times \Omega_{\boldsymbol{x}}. \tag{3.1}$$

As usual, an initial condition and boundary conditions have to be added in order to close the equation (3.1). Let in the following the domain $\Omega_{\boldsymbol{x}}$ be a sufficiently regular and connected subset of $\mathbb{R}^2$ or $\mathbb{R}^3$. The interval $(0, t_{\text{end}})$ is the temporal domain.

Usually, the unknown function $c : (0, t_{\text{end}}) \times \Omega_{\boldsymbol{x}} \to \mathbb{R}_0^+$ describes a macroscopic physical quantity in a fluid environment, e.g., a concentration of a chemical species, heat, or a moment of a particle size distribution. For the following explanations we stick to the notion of a chemical species concentration.

The spatial terms on the left-hand side, and the phenomena modeled by them, are eponymous for this type of PDE. Convective transport is introduced by the term $\boldsymbol{u} \cdot \nabla c$. There, $\boldsymbol{u}$ is a velocity field, the concentration c is transported inertialess by that field. The second order term $-\varepsilon \Delta c$ models diffusive transport. The diffusion parameter $\varepsilon \in \mathbb{R}^+$ is typically small compared to $\|\boldsymbol{u}\|$. If that is the case, $\frac{\varepsilon}{\|\boldsymbol{u}\|} << 1$, the CDRE is of *convection-dominated* type. The zeroth order term $r(c)$ models concentration gain and loss due to reaction, with the real-valued reaction function r. In case r is (affine) linear, so is the entire equation (3.1). In that case, we re-use the variable r in order to denote a reaction coefficient function, replacing the term $r(c)$ with some $r(t, \boldsymbol{x})c$. In case it is not, the non-linearity of the PDE is still restricted to the zeroth order term, and therefore not as essential as in the case of the Navier–Stokes equations, where the first order term is affected. Such a CDRE is also referred to as *quasilinear*.

The present chapter highlights a numerical methods which tackles a common problem of (systems of) convection-diffusion-reaction equations. A typical feature of convection-dominated CDRE is admitting solutions with sharp interior and boundary layers. Those layers are usually due to the dominant convective term or forcing boundary conditions, and only the presence of the second order term allows for smooth solutions at all. Examples in 1d can be used to illustrate the issue, and the issues that standard numerical methods face (Roos et al., 2008, Chapter 1). Essentially, on grids which are not fine enough to resolve the small spatial scales of the layers, standard methods either produce oscillatory solutions or solutions with a smeared layer. The construction and analysis of "stabilized" methods for CDRE, i.e., such methods, which

do neither produce oscillations nor smeared layers, have been an active field of research for several decades now. In the finite element community different "upwind stabilization" methods have been proposed, and from their ranks come the gold standard methods like the SUPG (Stream Upwind Petrov–Galerkin) method (Brooks and Hughes (1982)), see also the list in (Roos et al., 2008, p.84). These methods have in common that directed additional diffusion is introduced into the discretization. In applications, one is interested in physically consistent solutions, and at this point there is a problem with the classical methods. They tend to produce spurious over- and undershoots of the solution in the vicinity of steep interior or boundary layers. Yet in recent years a family of promising methods has emerged, they are called algebraic flux correction methods, and their main difference to upwind schemes is that they work on an algebraic level, i.e., after the PDE has been discretized *and* transferred to a linear system of equations. These methods are mathematically guaranteed to be free of over- and undershoots, and their practical usefulness in comparison to classical methods has been proven, i.a., in John and Schmeyer (2008).

One representative of algebraic flux correction methods, which is suitable for time-dependent CDRE, is the linear Crank–Nicolson FEM-FCT scheme of Kuzmin (2009). It will be used throughout this thesis in order to numerically stabilize CDRE, and shall be introduced in the following.

For that purpose, this chapter is organized as follows. In Section 3.1 the temporal (Crank–Nicolson) and spatial (FEM) discretization of Equation (3.1) is introduced. Section 3.2 introduces and explains the mentioned algebraic flux correction scheme. Section 3.3 finally offers a short digression to numerical schemes for the treatment of systems of CDRE that are coupled via the reactive term.

3.1 Finite element discretization

We will not enter the analysis of equations of type (3.1). Let us just briefly comment that, in order to gain analytical results, further regularity assumptions on the coefficient functions have to be made. Typical cases are constant coefficients, or smooth functions $\boldsymbol{u}$, r, and f (Roos et al., 2008, p.427), where $\boldsymbol{u}$ and r are bounded away from zero. In our purely numerical presentation, it is sufficient to require L^∞-boundedness of the coefficient functions in time and space. This is the setting of John and Schmeyer (2008), which is a survey article on finite element methods for convection–diffusion equations. Let thus in the following $\boldsymbol{u} \in L^\infty(0, t_{\text{end}}; W^{1,\infty}(\Omega_{\boldsymbol{x}}))^d$ be divergence-free with spatial dimension d, $r \in L^\infty(0, t_{\text{end}}; L^\infty(\Omega_{\boldsymbol{x}}))$, and $f \in L^2(0, t_{\text{end}}; L^2(\Omega_{\boldsymbol{x}}))$.

Our first discretization step is time discretization. Here and in the following we regard only the semi-implicit Crank–Nicolson scheme, as we did for the Navier–Stokes equations. Let $\Delta t \in \mathbb{R}^+$ be the constant time step size, and c^n denote the discrete solution at time step n. The Crank–Nicolson discretization

gives the following equation (a stationary CDRE) in each time step:

$$c^{n+1} + \frac{\Delta t}{2}\left(-\varepsilon\Delta c^{n+1} + \boldsymbol{u}^{n+1}\cdot\nabla c^{n+1} + r^{n+1}c^{n+1}\right) = c^n - \frac{\Delta t}{2}\left(-\varepsilon\Delta c^n + \boldsymbol{u}^n\cdot\nabla c^n + r^n c^n - \left(f^n + f^{n+1}\right)\right). \quad (3.2)$$

The spatial discretization is performed by application of the finite element method, as was described for the Navier–Stokes equations in Section 2.2. As usual, one has different options for the choice of a finite element space V_h. Assume such boundary conditions that the same space can be used as test- and ansatz space, e.g., homogeneous Neumann boundary conditions. In order to apply the algebraic flux correction scheme of Kuzmin (2009), the discrete space must be of first order (with the only notable exception of (Kuzmin (2008)), as is remarked in (Barrenechea et al., 2016, p.2428)). The finite element discretization is based on a variational formulation of Equation (3.2). A function $c_h^{n+1} \in V_h$ is called discrete solution to the CDRE in the $(n+1)$st time step if

$$\left(c_h^{n+1}, \varphi_h\right) + \frac{\Delta t}{2}\left(\left(\varepsilon\nabla c_h^{n+1}, \nabla\varphi_h\right) + \left(\boldsymbol{u}_h^{n+1}\cdot\nabla c_h^{n+1} + r_h^{n+1}c_h^{n+1}, \varphi_h\right)\right) = \left(c_h^n, \varphi_h\right) - \frac{\Delta t}{2}\left(\left(\varepsilon\nabla c_h^n, \nabla\varphi_h\right) + \left(\boldsymbol{u}_h^n\cdot\nabla c_h^n + r_h^n c_h^n - \left(f_h^n + f_h^{n+1}\right), \varphi_h\right)\right) \quad (3.3)$$

holds for all test functions $\varphi_h \in V_h$, and the boundary conditions are fulfilled, too. We skip those details here. Note that all coefficient functions must be projected to V_h in the discretization, this is denoted by the subscript h in the above equation.

Aforementioned standard stabilization methods proceed by modifying the Galerkin discretization, which is Equation (3.3). Algebraic flux corrections schemes, on the other hand, work one level higher: on the level of finite element matrices. This procedure is the subject of the following section.

3.2 The linear Crank–Nicolson FEM-FCT scheme

Converting the fully discretized convection-diffusion-reaction equation (3.3) to an algebraic linear system of equations is performed in the same manner as has been shown for the Navier–Stokes equations in Section 2.3.1. Picking a nodal basis $\Phi := (\varphi_i)_{i=1,\dots,N}$ of the finite element space V_h and reducing the equation (3.3) to the basis elements gives a linear system of equations for the coefficients of the solution function in the basis Φ. This system can be written in the following way:

$$\left(\mathbf{M} + \frac{\Delta t}{2}\mathbf{K}^{n+1}\right)\mathbf{c}^{n+1} = \left(\mathbf{M} - \frac{\Delta t}{2}\mathbf{K}^n\right)\mathbf{c}^n + \frac{\Delta t}{2}\left(\mathbf{f}^{n+1} + \mathbf{f}^n\right). \quad (3.4)$$

The unknown $\mathbf{c}^{n+1}$ is the solution coefficient vector, i.e., $\left(\mathbf{c}^{n+1}\cdot\Phi\right)$ will be the finite element solution. The other objects are the *mass matrix* $\mathbf{M}$, the *stiffness matrices* $\mathbf{K}^{n+1}$ and $\mathbf{K}^n$, and the right-hand side vectors $\mathbf{f}^{n+1}$ and $\mathbf{f}^n$,

where stiffness matrices and right-hand sides depend on the time step, as will be shown below.

The mass matrix stems from the spatial discretization of the time derivate, it is

$$\mathbf{M} = (m_{ij}) = \big((\varphi_j, \varphi_i) \big)_{i,j=1,\dots,N}.$$

The stiffness matrix assembles the terms from the spatial discretization of diffusion, convection, and reaction terms, i.e.,

$$\mathbf{K}^n = \big(\varepsilon(\nabla\varphi_j, \nabla\varphi_i) + (\boldsymbol{u}^n \cdot \varphi_j, \varphi_i) + (r^n\varphi_j, \varphi_i)\big)_{i,j=1,\dots,N}.$$

The interesting case is that of dominant convection. This motivated the symbol for the stiffness matrices: $\mathbf{K}$ for German "Konvektion".

The algebraic flux correction (AFC) scheme which is to be presented in this section was formally introduced in Kuzmin (2009), yet it builds on a more complex series of publications by the same author and affiliates. That work resulted in two families of AFC schemes for finite element discretizations: schemes of Total Variation Diminishing type (TVD), which are suitable for stationary and weakly time-dependent problems, and schemes of Flux Corrected Transport type (FCT), which are suitable for more strongly time-dependent problems that call for a (semi-)implicit time discretization. See the overview in (Kuzmin and Möller (2005)) for a comparison of both approaches, and the introduction to Barrenechea et al. (2015) for a short historical survey. Note that our survey of the method follows in parts that of (Suciu, 2013, pp.26).

The common idea of AFC schemes is to maintain a discrete maximum principle by adding "enough" artificial diffusion, and then strategically taking away a part of that artificial diffusion. This is performed by the application of flux limiters, which usually depend on an intermediate solution. In that sense, AFC schemes are predictor-corrector methods. The Crank–Nicolson FEM-FCT approach is semi-implicit as the time-stepping scheme itself, and it uses a forward Euler solution as a predictor step.

All methods of the FEM-FCT family start with the formulation of a non-oscillatory low order scheme, which initially replaces (3.4). The matrix

$$\mathbf{M}_L := \operatorname{diag}\big(m_i\big)_{i=1,\dots,N} \quad \text{with} \quad m_i := \sum_{j=1}^{n} (\varphi_j, \varphi_i)$$

is the lumped mass matrix, and

$$\mathbf{D}^n = \big(d_{ij}^n\big)_{i,j=1,\dots,N},$$

with

$$d_{ij}^n = \max\{-k_{ij}^n, 0, -k_{ji}^n\} \quad \text{and} \quad d_{ii}^n = -\sum_{j=1, j\neq i}^{N} d_{ij}^n$$

is the artificial diffusion matrix. The overly diffusive stiffness matrix is defined as

$$\mathbf{K}_L^n := \mathbf{K}^n + \mathbf{D}^n.$$

Then the low-order scheme, comprising the above low-order operators, is

$$\left(\mathbf{M}_L + \frac{\Delta t}{2}\mathbf{K}_L^{n+1}\right)\mathbf{c}^{n+1} = \left(\mathbf{M}_L - \frac{\Delta t}{2}\mathbf{K}_L^{n}\right)\mathbf{c}^{n} + \frac{\Delta t}{2}\left(\mathbf{f}^{n+1} + \mathbf{f}^{n}\right). \tag{3.5}$$

In the next step, one decomposes the difference of the residuals of the low-order scheme (3.5) and the Galerkin scheme (3.4). For a given approximate solution $\tilde{\mathbf{c}}^{n+1}$ these residuals are

$$\mathbf{r}_L = \left(\mathbf{M}_L + \frac{\Delta t}{2}\mathbf{K}_L^{n+1}\right)\tilde{\mathbf{c}}^{n+1} - \left(\mathbf{M}_L - \frac{\Delta t}{2}\mathbf{K}_L^{n}\right)\mathbf{c}^{n} - \frac{\Delta t}{2}\left(\mathbf{f}^{n+1} + \mathbf{f}^{n}\right)$$

for the low-order scheme, and

$$\mathbf{r}_G = \left(\mathbf{M} + \frac{\Delta t}{2}\mathbf{K}^{n+1}\right)\tilde{\mathbf{c}}^{n+1} - \left(\mathbf{M} - \frac{\Delta t}{2}\mathbf{K}^{n}\right)\mathbf{c}^{n} - \frac{\Delta t}{2}\left(\mathbf{f}^{n+1} + \mathbf{f}^{n}\right)$$

for the Galerkin scheme. Each component r_i of their difference

$$\mathbf{r} := \mathbf{r}_L - \mathbf{r}_G = (\mathbf{M}_L - \mathbf{M})\left((\tilde{\mathbf{c}}^{n+1} - \mathbf{c}^{n}\right) + \frac{\Delta t}{2}\left(\mathbf{D}^{n+1}\tilde{\mathbf{c}}^{n+1} + \mathbf{D}^{n}\mathbf{c}^{n}\right)$$

can be decomposed into so-called *raw antidiffusive fluxes* r_{ij}. For their definition, observe that

$$\begin{aligned} r_i = \sum_{j\neq i}\Bigg(&(m_{L,ij} - m_{ij})\left(\tilde{c}_j^{n+1} - c_j^{n}\right) + \frac{\Delta t}{2}\left(d_{ij}^{n+1}\tilde{c}_j^{n+1} + d_{ij}^{n}c_j^{n}\right)\Bigg) \\ &+ (m_{L,ii} - m_{ii})\left(\tilde{c}_i^{n+1} - c_i^{n}\right) + \frac{\Delta t}{2}\left(d_{ii}^{n+1}\tilde{c}^{n+1} + d_{ii}^{n}c^{n}\right) \end{aligned} \tag{3.6}$$

$$\begin{aligned} = \sum_{j\neq i}\Bigg(&(-m_{ij})\left(\tilde{c}_j^{n+1} - c_j^{n}\right) + \frac{\Delta t}{2}\left(d_{ij}^{n+1}\tilde{c}_j^{n+1} + d_{ij}^{n}c_j^{n}\right)\Bigg) \\ &+ \sum_{j\neq i}\left(m_{ij}\left(\tilde{c}_i^{n+1} - c_i^{n}\right) - \frac{\Delta t}{2}\left(d_{ij}^{n+1}\tilde{c}_i^{n+1} + d_{ij}^{n}c_i^{n}\right)\right) \end{aligned} \tag{3.7}$$

$$= \sum_{j\neq i}\left(\left(m_{ij} - \frac{\Delta t}{2}d_{ij}^{n+1}\right)\left(\tilde{c}_i^{n+1} - \tilde{c}_j^{n+1}\right) - \left(m_{ij} + \frac{\Delta t}{2}d_{ij}^{n}\right)\left(c_i^{n} - c_j^{n}\right)\right). \tag{3.8}$$

In step (3.7) the definitions of matrices $\mathbf{M}_L$ and $\mathbf{D}$ were used. Write the above sum as

$$r_i = \sum_{j=1, j\neq i}^{N} r_{ij}.$$

The raw antidiffusive fluxes r_{ij} are transformed into *limited* antidiffusive fluxes by multiplying each of them with a solution-dependent flux limiter $\alpha_{ij}(\tilde{\mathbf{c}}^{n+1}) \in [0,1]$ that will be defined later.

The general FEM-FCT flux corrected (or: high-order) scheme is the low-order scheme plus a correction term:

$$\begin{aligned} &\left(\mathbf{M}_L + \frac{\Delta t}{2}\mathbf{K}_L^{n+1}\right)\mathbf{c}^{n+1} \\ &\qquad = \left(\mathbf{M}_L - \frac{\Delta t}{2}\mathbf{K}_L^{n}\right)\mathbf{c}^{n} + \mathbf{r}^{*}\left(\mathbf{c}^{n+1}\right) + \frac{\Delta t}{2}\left(\mathbf{f}^{n+1} + \mathbf{f}^{n}\right). \end{aligned} \tag{3.9}$$

The corrected fluxes

$$r_i^* := \sum_{j \neq i} \alpha_{ij} \left(\mathbf{c}^{n+1}\right) r_{ij} \left(\mathbf{c}^{n+1}\right)$$

are gathered in the implicit flux correction vector $\mathbf{r}^* \left(\mathbf{c}^{n+1}\right)$.

Immediately one notices two things. The first observation is that scheme (3.9) is equal to scheme (3.5) (low-order) if $\alpha_{ij} \equiv 0$, and equal to scheme (3.4) (Galerkin) if $\alpha_{ij} \equiv 1$. Secondly, the flux correction term introduces a non-linearity to the equation. The further handling of that non-linearity distinguishes different algorithms of the FEM-FCT family. In the following we give a scheme which is an explicit-implicit formulation, requires just one solution of a linear system per time step, and is a good compromise between computational effort and accuracy.

The idea of the linear Crank–Nicolson FEM-FCT scheme is to compute the raw antidiffusive fluxes and the correction factors around an intermediate solution $c^{n+\frac{1}{2}}$, which is the explicit part of the low order scheme (3.5). Multiplying the general scheme (3.9) with $\mathbf{M}_L^{-1}$ gives

$$\begin{aligned} &\left(I + \frac{\Delta t}{2} \mathbf{M}_L^{-1} \mathbf{K}_L^{n+1}\right) \mathbf{c}^{n+1} \\ &\qquad = \mathbf{c}^n - \frac{\Delta t}{2} \mathbf{M}_L^{-1} \mathbf{K}_L^n \mathbf{c}^n + \mathbf{M}_L^{-1} \mathbf{r}^* \left(\mathbf{c}^{n+1}\right) + \frac{\Delta t}{2} \mathbf{M}_L^{-1} \left(\mathbf{f}^{n+1} + \mathbf{f}^n\right) \end{aligned} \tag{3.10}$$

Note that the explicit part $\mathbf{c}^{n+\frac{1}{2}} := \mathbf{c}^n - \frac{\Delta t}{2} \mathbf{M}_L^{-1} \mathbf{K}_L^n \mathbf{c}^n$ of that system corresponds to the solution of a forward Euler low-order scheme with a time step size of $\frac{\Delta t}{2}$. It can be used in order to gain a linearized form of the flux correction vector $\mathbf{r}^*$. For that purpose the approximation

$$\mathbf{c}^{n+\frac{1}{2}} \approx \frac{\mathbf{c}^{n+1} + \mathbf{c}^n}{2}$$

is made, reordering gives

$$\mathbf{c}^{n+1} \approx 2\mathbf{c}^{n+\frac{1}{2}} - \mathbf{c}^n. \tag{3.11}$$

This approximation is used for the computation of the raw fluxes,

$$\mathbf{r}^{n+\frac{1}{2}} := \mathbf{r}(2\mathbf{c}^{n+\frac{1}{2}} - \mathbf{c}^n).$$

The correction factors $\alpha_{ij}^{n+\frac{1}{2}}$ are computed for the explicit approximation (3.11). Inserting the resulting limited fluxes

$$r_i^{*,n+\frac{1}{2}} := \sum_{j \neq i} \alpha_{ij}^{n+\frac{1}{2}} r_{ij}^{n+\frac{1}{2}}$$

into (3.10) componentwise gives the final Crank–Nicolson FEM-FCT scheme.

For the computation of the limiting factors, usually Zalesak's flux limiter (Zalesak (1979)) is used, we do not repeat its formulation here.

The AFC scheme presented above was already implemented in the software package ParMooN, and the algorithm will be used for all convection-diffusion-reaction problems encountered in the numerical Chapters 6 and 7.

3.3 A note on systems of convection-diffusion-reaction equations

In the later course of this thesis, when population balance systems are introduced (Chapter 5) and solved numerically (Chapters 6 and 7), we will encounter only such convection-diffusion-reaction equations that are coupled to each other *indirectly* via a population balance equation. The reason is that the crystallizer systems we model contain just one dissolved species and a solute. Yet in chemical and biological applications one regularly encounters systems containing a number of different transported species, whose concentrations are *directly* coupled to each other by certain reaction terms. Therefore the incorporation of a *system* of reactively coupled CDRE seems a natural extension of the proposed methods for population balance systems.

A system of M reacting transported species $c_1, ..., c_M$ can be written in the generic form

$$\begin{aligned}
&\frac{\partial c_1}{\partial t} - \varepsilon_1 \Delta c_1 + \boldsymbol{u} \cdot \nabla c_1 + r_1(c_1, ..., c_M) = f_1 \\
&\frac{\partial c_2}{\partial t} - \varepsilon_2 \Delta c_2 + \boldsymbol{u} \cdot \nabla c_2 + r_2(c_1, ..., c_M) = f_2 \\
&\quad \vdots \\
&\frac{\partial c_M}{\partial t} - \varepsilon_M \Delta c_M + \boldsymbol{u} \cdot \nabla c_M + r_3(c_1, ..., c_M) = f_M,
\end{aligned}$$

or in vectorial form, with obvious notation, as

$$\frac{\partial \bar{c}}{\partial t} - \mathbf{E} \Delta \bar{c} + \mathbf{U} \nabla \bar{c} + R(\bar{c}) = F. \tag{3.12}$$

The vectorial function $R := (r_1, ..., r_M) : \mathbb{R}^M \to \mathbb{R}^M$ contains the (eventually nonlinear) reaction terms. Systems of the type (3.12) contain two further sources of numerical difficulty: nonlinearity and nonnegativity. Both issues are closely connected with each other, and closely connected to the time discretization. Therefore, complete numerical schemes for systems like (3.12) are usually fully integrated methods, solving all these difficulties at once.

A strategy that was pursued in Suciu (2013) was a Crank–Nicolson discretization in time, followed by a Picard-type iteration (usually only one step) in order to resolve the nonlinearity introduced by R. In addition, clipping of negative concentrations was performed in order to avoid spurious negative concentration. This is a common strategy (Formaggia and Scotti, 2011, p.1268) to avoid the problem of negative concentrations, which several time-stepping schemes (among them all one-step θ-schemes except for the implicit Euler, see Formaggia and Scotti (2011)) face.

An interesting alternative to such an ad-hoc approach are Patankar-type methods. Those were originally invented by Patankar (1980) and used in the context of heat transfer, and the main idea entered the collective memory as *Patankar trick*. Consider a system of ODEs with production (i.e., source) terms $P(\bar{c}) = (p_1, ..., p_M)$ and destruction (i.e., sink) terms $D(\bar{c}) = (d_1, ..., d_M)$

$$\dot{\bar{c}} = P(\bar{c}) - D(\bar{c})$$

and discretize it with the forward Euler method. Then for $i = 1, ..., M$ it is defined

$$c_i^{n+1} = c_i^n + \Delta t \left(p_i(\bar{c}^n) - d_i(\bar{c}^n) \right).$$

The "traditional" Patankar trick consists in scaling the troublesome destruction terms (those introduce non-positiveness) with the quotient of old and new solution:

$$c_i^{n+1} = c_i^n + \Delta t \left(p_i(\bar{c}^n) - d_i(\bar{c}^n) \frac{c_i^{n+1}}{c_i^n} \right).$$

This forward-Euler–Patankar scheme is unconditionally positive, i.e., preserves positivity of the numerical solution for any time step Δt. Based on it in a series of publications starting with Burchard et al. (2003), so-called *modified Patankar* schemes were developed. By scaling not only the destruction terms, but the production terms, too, schemes that are positivity preserving *and* mass conserving were developed, even higher-order Runge-Kutta schemes. So far, applications of modified Patankar schemes in the context of *partial* differential equations are few and relatively recent. We are aware of Meister and Ortleb (2014) (shallow water equations), Mabuza et al. (2014) (fluid structure interaction), and the short analysis in Ortleb and Hundsdorfer (2017). In Mabuza et al. (2014) they were used in combination with linear ALE-FCT schemes similar to the scheme presented in Section 3.2, and within a Crank–Nicolson discretization.

If the coupled algorithm of this thesis is ever applied to a population balance system comprising multiple chemical species that are coupled in their reaction term, we propose to follow the route taken in Mabuza et al. (2014) with the modified Patankar–Crank–Nicolson scheme.

4 Stochastic Particle Methods

In the mathematics of coagulating particles there are two fundamental approaches: A deterministic and a stochastic approach. Both approaches are built around the same set of ODEs, the *Smoluchowski coagulation equation*, which will be introduced first of all.

Let $\mathcal{M}$ be a one-dimensional particle state space. To fix ideas, let $m \in \mathcal{M}$ describe the mass of a particle in a continuous and unbounded manner, and therefore set $\mathcal{M} = (0, +\infty)$. Other common choices for the particle state space would be $\mathbb{Z}^+$ or $\{1, ..., N\}$. The time t should lie in $(0, +\infty)$. The *coagulation kernel* $K(m_1, m_2)$ gives the (time-independent) rate at which a *specific* pair of particles of masses m_1 and m_2 coagulates. The unknown function $f(t, m)$ gives the number of particles of mass m at time t in some predefined, homogeneous volume. Thus one can picture f as a spatially homogeneous mass-concentration spectrum evolving in time. Two-particle coalescence, i.e, the merging of two particles, forming a new, larger particle of their combined mass, is the only physical process present in the equation and is the only cause of changes in the spectrum f. With these preliminary considerations, the Smoluchowski coagulation equation reads

$$\frac{\partial}{\partial t} f(t,m) = \frac{1}{2} \int_{\substack{m_1, m_2 \in \mathcal{M} \\ m_1 + m_2 = m}} K(m_1, m_2) f(t, m_1) f(t, m_2) \, \mathrm{d}m_1 \mathrm{d}m_2 - \int_{\mathcal{M}} K(m_1, m) f(t, m_1) f(t, m) \, \mathrm{d}m_1 \quad \forall m \in \mathcal{M}. \quad (4.1)$$

This equation is a non-linear integro-differential equation, or from a different view, it is a system of such equations, since it contains one equation for each $m \in \mathcal{M}$. Depending on the choice of $\mathcal{M}$, it gives in total a finite, countably infinite or even uncountably infinite number of equations. The equation (4.1) has appeared under the names *stochastic coagulation equation*, *stochastic coalescence equation* or just *coagulation equation*. We will use the last option.

The coagulation equation is used to describe the time evolution of a population of particles of different masses, which interact with each other by coagulation. If the particles are physical particles like crystals, molecules or water droplets, the term "collision growth" is often used instead of "coagulation". We will use these terms interchangeably.

The equation (4.1) is central for both the deterministic and the stochastic approach to coagulation mathematics, but in different ways. Although both approaches describe first and foremost different *numerical* strategies to the solution of equation (4.1), there is more to it: they introduce a difference on the *modeling* level, which we want to describe briefly.

In the deterministic approach, the coagulation equation is derived by typical infinitesimal reasoning. The search for solutions, both analytical and numerical,

and their properties, proceeds with the instruments of the theory of differential equations. The stochastic approach, on the other hand, starts with the formulation of a stochastic process that describes state and change of state of a coagulating particle population. If the stochastic process is in accordance with the deterministic theory, one will find a link between the coagulation equation and the Kolmogorov forward equation of the process. In that sense, the coagulation equation arises naturally from the stochastic theory. Concluding, one could put it like this: While the deterministic approach *starts* from the coagulation equation, the stochastic approach *results* in it. And while the deterministic approach opens up different numerical strategies after the equation is formulated, the stochastic approach yields a numerical strategy *en passant*. This we will see in the remainder of this chapter.

Further thoughts on the subject, how stochastic processes lead to ODEs on the modeling level, and on what modeling assumptions the formulation of a stochastic process itself grounds, can be found in (van Kampen, 1981, ch. III.2).

In this work, we will focus on the stochastic approach, because we have in mind to use a stochastic simulation algorithm as part of our coupled method. The deterministic approach to mathematical coagulation and the resulting numerical methods will not be regarded beyond this introduction.

Let us elaborate on the history and interpretation of the coagulation equation (4.1), before we give the outline of the chapter. The original formulation appeared in Smoluchowski (1916), and it had been studied from the view of partial differential equations before the stochastic approach emerged in the 1960s. This might be a reason, why a central concern within the literature on the stochastic approach is to establish a connection between stochastic approach and coagulation equation. The original coagulation equation can be regarded as a reduced version of a population balance equation. This type of equations is surveyed in the textbook Ramkrishna (2000). With some knowledge of partial differential equations, one will have no difficulties to grasp a good "working interpretation" of Equation (4.1). The equation describes the time evolution of the space average of a fixed class of particles identified by their mass m. There are only two ways how the number of particles of mass m can change within an infinitesimal time $\mathrm{d}t$. First, by coagulation of two smaller particles whose masses add up to m, which raises the number of m-particles. Let m_1 and m_2 be such masses. As was mentioned before, $K(m_1, m_2)$ is the rate at which a *specific* pair of particles with these masses coagulates. The rate at which *any* pair of particles of masses (m_1, m_2) coagulates depends on the availability of such pairs, therefore on $f(t, m_1)$ and $f(t, m_2)$, in the form $K(m_1, m_2)f(t, m_1)f(t, m_2)$. For this gain of m-particles by coagulation, the first integral term (the "gain term") is responsible. The second significant event is a particle of mass m coagulating with any other particle and thus "disappearing" from the class of m-particles. The second integral term in (4.1) is responsible for that loss ("loss term"), the explanation is very similar to that of the gain term. The net change rate is then the difference of gain and loss term.

The coagulation kernel K is derived using a physical model of the coagulation process. The physical relevance of solutions of (4.1) is significantly influenced by the quality of this model. Therefore, kernel choice is a fundamental modeling

decision, which determines physical and analytical properties of the equation, and also implies numerical consequences.

The coagulation equation can be modified and expanded in order to depict a wider range of physical phenomena. This can be done by replacing one of the mathematical entities in the formulation of Equation (4.1). Doing so is necessary when one aims at a more sophisticated particle description and has to exchange the mass-only state space $\mathcal{M}$ for a higher dimensional space. It is also necessary when one incorporates a spatial dependency of f, turning it into a particle number concentration. Modifications of (4.1) can also be achieved by adding further terms on either side of the equality sign. Phenomena which make such a modification necessary include, e.g., particle growth, transport, breakage and particle insertion by nucleation or inflow.

Let us now give the structure of this chapter. In Section 4.1 we give a literature overview over stochastic particle methods. In Section 4.2 we present a specific branch of stochastic coagulation modeling, the Marcus–Lushnikov process. We include a short introduction to the theory of Markov jump processes. The stochastic simulation algorithm is rooted in the theory of Markov jump processes, and the Marcus–Lushnikov process in particular, which is why we decided to include Section 4.2, before turning to the more applied subjects. In the main section, 4.3, we describe the stochastic simulation algorithm, which will be part of our coupled algorithm, and the software *Brush*, in which it is implemented.

4.1 Literature review

In this section we want to give a short, and by no means complete, overview over the literature regarding stochastic approaches to the coagulation equation.

The most basic review article of the field is Aldous (1999). A more recent overview article on the usage of stochastic methods in coalescence theory is Berestycki (2009), but it leaves out the whole theory of Marcus–Lushnikov processes, which form the basis of our stochastic simulation approach. For our purposes, Aldous (1999) is the survey article of choice. Therein the author claims (although he calls it a "gross oversimplification") that in pre-2000 research, there have been two waves of theoretical interest in the coagulation equation. According to this perspective, the first wave occurred in the 1960s in the physical chemistry community and resulted in the deterministic theory. The stochastic theory is, according to the author, the fruit of the second wave of interest in the 1980s. There the leading researchers had a background in statistical physics. A survey of the first wave was given in Drake (1972). A more recent survey on the achievements of the deterministic theory can be found in Laurençot and Mischler (2004).

The most general survey paper of the "second wave" stochastic theory is Aldous (1999) itself, although it deals in large parts with constructions of stochastic processes given by the author in the 1990s. Therefore, besides summarizing the 1980s theory, it can be regarded as an initial paper of a *third* wave of interest, whose emergence it predicts, and which we want to name the

Markovian approach to coalescence. This wave is the latest one, it is mainly conveyed by applied probabilists, and in our opinion works like Norris (1999), Eibeck and Wagner (2001), Deaconu et al. (2002), Wagner and Eibeck (2003), Yaghouti et al. (2009), and Patterson (2013) belong to this third wave.

The works of this period are characterized by a great depth of mathematical theory, especially theory of Markov processes as laid out in Ethier and Kurtz (1986). Stochastic processes in general and Markov jump processes will be introduced in greater detail in Section 4.2.1, here we only give a short overview. A Markov process is a stochastic process $(X_t)_{t\in\mathbb{R}_0^+}$ that possesses the Markov property, i.e., for each finite set of times $\{t_0, ..., t_n\} \subseteq \mathbb{R}_0^+$ (in ascending order) holds

$$P(X_t = x | X_{t_0} = x_{t_0}, ..., X_{t_n} = x_{t_n}) = P(X_t = x | X_{t_n} = x_{t_n}).$$

This equation has a fundamental consequence. A Markov process is fully characterized by just two probability distributions – an *initial distribution* and a *transition probability distribution*. Although it is well known in all communities dealing with stochastic processes that stochastic processes become considerably more accessible when they possess the Markov property, it seems to us that the third wave of mathematical coagulation literature is especially aware of that fact and its theoretical consequences. This is why we called it *Markovian* approach to coalescence theory here.

Let us proceed with the literature overview. As was mentioned before, in the center of our attention is a stochastic process known as *Marcus–Lushnikov process*, because our numerical method as well as related analysis has its roots in the study of this process. Therefore we will concentrate on literature which gives some insight into the theory of Marcus–Lushnikov processes.

To the best of our knowledge, versions of the concept appeared at three different places independently between the late 1960s and early 1970s, which founded three different lines of literature which only converged in the third-wave literature. The first of these lines was initiated by Marcus (1968), who was the first to present the construction of the process. The process was then reformulated in Lushnikov (1978a) and Lushnikov (1978b). The authors of these articles are eponymous for the process. All three articles establish connections between the process and the coagulation equation.

The second place where the process appeared was within the work of Daniel T. Gillespie in Gillespie (1972) and Gillespie (1975). In Gillespie (1972) the process is presented, a physical heuristic for its derivation is given, and it is shown how the coagulation equations arise from the stochastic dynamics of the Marcus–Lushnikov process under additional, simplifying, assumptions. Gillespie (1975) shows a way to simulate the process numerically by a two-stage Monte Carlo method. It is an interesting and seemingly often overlooked fact that the theory developed in Gillespie (1972) and the algorithm presented in Gillespie (1975) for the coagulation of cloud droplets were transferred by the same author to systems of spatially homogeneous chemical reactions (Gillespie (1976, 1977)). The algorithm turned into a classic of computational chemistry, known as the Gillespie SSA, where SSA stands for *stochastic simulation algorithm*. Yet it is

seldom that one finds this connection noted in the literature, a notable exception being Wagner and Eibeck (2003). We find it necessary to point out this connection, because it establishes the usefulness of introductory material from computational chemistry for our purpose. In that sense, an excellent, practical introduction to the Gillespie SSA is Erban et al. (2007). A good reference for the basic connection of stochastic and deterministic models (for chemical reactions!) is Kurtz (1972).

The third place of independent emergence of the Marcus–Lushnikov process is Shah et al. (1977), and it focuses on developing a stochastic algorithm. This work is mainly known and cited in the engineering community specializing on population balance equations, e.g., in the standard reference Ramkrishna (2000), and we only mention it for the sake of completeness.

In order to prepare the understanding of third-wave articles, a detailed knowledge of Markov processes and their applications is required. A good introductory work explaining the usefulness of Markov processes in science is van Kampen (1981). Note also the short remarks on non-Markovian processes by the same author, van Kampen (1998), which is a quick read and underlines the importance of Markov processes in a dialectic way. A textbook on Markov processes with infinite-dimensional state spaces, which contains the powerful theory that is used, e.g., in Norris (1999) and Wagner and Eibeck (2003), is Ethier and Kurtz (1986). The existence proof in Patterson (2013) for a stochastic process including not only coagulation, but also particle birth, particle removal and, notably, convective particle transport is based on the theory of *piecewise deterministic Markov processes* as developed and presented in the textbook Davis (1993). This framework is very interesting, because it allows for seamless integration of an external transport mechanism into the stochastic formulation.

Apart from Gillespie (1975) and Shah et al. (1977), there are several more references which present the basic ideas of stochastic simulation algorithms. The most relevant publications, describing versions of the algorithm that we will use later, as it was used for the simulation of soot formation, are Patterson et al. (2011) and Patterson and Wagner (2012). A review article on previous stochastic simulation methods is Sabelfeld et al. (1996).

At this point we conclude the literature review and move on to the closer examination of the Marcus–Lushnikov process.

4.2 The Marcus–Lushnikov process

In this section we introduce the Marcus–Lushnikov process, which is a classical stochastic model for binary particle coagulation in a finite volume setting. It models coagulation as a continuous-time Markov process with finite state space. To fix ideas, consider a spatially homogeneous system of coalescing particles in some finite volume. Every particle is identified by its mass alone. The particle mass is given in terms of integer multiples of a unit mass m_0, i.e., an i-particle has mass $i \cdot m_0$. With this interpretation in mind, we will in the following speak about particles "of mass i", where "of mass $i \cdot m_0$" would be correct. Note that with this assumption the inner coordinate space is *discrete*

and *one-dimensional*. Since the setting requires a finite volume, the total mass M within the system is finite, $M < \infty$. Let M be an integer multiple of m_0, thus $M = N \cdot m_0$. Consequently the inner coordinate space is finite. The main idea of the Marcus–Lushnikov process is to represent the state of the system at time t as

$$X^{\mathrm{ML}}(t) = (n_1, ..., n_N).$$

There $n_i \in \{0, ..., N\}$ for all $i \in \{1, ..., N\}$. The interpretation is as follows. The value

$$\left(X^{\mathrm{ML}}(t)\right)_i = n$$

means that at time t there are n particles of mass i in the system – each component of $\left(X^{\mathrm{ML}}(t)\right)$ counts the number of particles of a certain mass in the system.

The conservation of mass within the particle system is expressed by the equation

$$\sum_{i=1}^{N} i \cdot n_i = N,$$

which must hold for all states $X^{\mathrm{ML}}(t)$.

Randomness enters the Marcus–Lushnikov process through the assumption of random *coagulation jumps*. A coagulation jump happens at a random time and changes the state of the system like

$$(n_1, ..., n_N) \longrightarrow (n_1, ..., n_i - 1, ..., n_j - 1, ..., n_{j+i} + 1, ..., n_N).$$

In this example, a particle of mass i and a particle of mass j merged into a particle of mass $i + j$.

Jumps of this type are supposed to happen at a stochastic rate

$$\chi_{\mathrm{coag}}(i, j) = K(i, j)\frac{n_i n_j}{N},$$

which is obviously dependent on the current state of the system. Behind this formula is the assumption that within an infinitesimal time, coagulation of two specific particles of masses i and j happens with probability

$$K(i, j) \ \ \mathrm{d}t,$$

where K is a coagulation kernel as known from the deterministic theory (Aldous, 1999, p.25), and $\mathrm{d}t$ an infinitesimal time span. This probability is then multiplied with the number of potential i- and j-partner particles, and scaled with $\frac{1}{N}$ for reasons explained later.

In the remainder of this section we proceed as follows. In Section 4.2.1 we introduce the notion and some important properties of Markov jump processes. An exact definition of the Marcus–Lushnikov process as a time-continuous, finite-state Markov process will be given in Section 4.2.2, its manifestations in the literature are discussed in Section 4.2.3. In Section 4.2.4 we show two canonic ways of establishing a connection between a stochastic process and a deterministic equation, and point out some classical results in that direction for the Marcus–Lushnikov process. In Section 4.2.5 we briefly raise some modeling strategies beyond the basic Marcus–Lushnikov process.

4.2.1 Markov jump processes

The Marcus–Lushnikov process belongs to a certain class of stochastic processes, which have a continuous time variable and map to a finite state space. The trajectories of these processes exhibit jumps, which explains the name *jump processes*. One observes the connection to the notion of "jump functions" in analysis. The theory of Markov jump processes, i.e., jump processes which additionally possess the Markov property, is very comprehensibly introduced in the textbook Norris (1998). In this subsection we want to provide the basic notions, so that we are able to give a precise definition of the Marcus–Lushnikov process in the next subsection. We base the following introduction on Norris (1998).

From now on assume that all appearing random variables are defined on the same probability space $(\Omega, \mathfrak{F}, P)$. Let us begin with the definition of a stochastic process.

Definition 4.2.1. *Let E be a countable set with power set $\mathfrak{E}$ and let $\mathfrak{T}$ be an index set. A **stochastic process** is a family of random variables*

$$(X_t)_{t\in\mathfrak{T}} : (\Omega, \mathfrak{F}, P) \longrightarrow (E, \mathfrak{E}).$$

*The codomain E which the random variables share is called the **state space** of the process. An element $i \in E$ is called a **state** of the process.*

For $\omega \in \Omega$ fixed, we call

$$X(t, \omega) := X_t(\omega) : \mathfrak{T} \longrightarrow E$$

*a **path** or **trajectory** of the process.*

Although this definition can be extended to non-countable state spaces with different sigma algebras, we settle for the countable case. The index set $\mathfrak{T}$ is usually interpreted as the time. For a stochastic process the information

$$X(t, \omega) = i$$

means that at time t the process finds itself in the state i. The variable ω from the sample space Ω, which is left unspecified, describes the underlying randomness. In practice two choices of time sets $\mathfrak{T}$ are of interest. These are given in the following definition.

Definition 4.2.2. *A stochastic process $(X_t)_{t\in\mathfrak{T}}$ is called*

(i) ***discrete-time**, if $\mathfrak{T} = \mathbb{N}_0$, and*

(ii) ***continuous-time**, if $\mathfrak{T} = \mathbb{R}_0^+$.*

For a discrete-time stochastic process we write $(X_n)_{n\in\mathbb{N}_0}$, and for a continuous-time stochastic process we write $(X_t)_{t\geq 0}$. These symbol conventions transfer accordingly to processes with a non-zero starting time $t_0 \neq 0$.

In order to allow for further analysis, stochastic processes need to be equipped with additional structure. The most common and fundamental class of processes are those that possess the Markov property. Loosely speaking, the

Markov property states that the past states of a process do not influence its future, provided its present state is known.

For a countable state space E, the definition and characterization of a Markov process makes use of two related classes of matrices, which we shall introduce in the next definition.

Definition 4.2.3. *A matrix $Q = (q_{ij})_{i,j\in E}$ with real valued entries is called a **Q matrix** if*

(i) $q_{ii} < 0 \quad \forall i \in E,$

(ii) $q_{ij} \geq 0 \quad \forall i, j \in E, i \neq j,$ *and*

(ii) $\sum_{j\in E} q_{ij} = 0 \quad \forall i \in E.$

Definition 4.2.4. *A matrix $P = (p_{ij})_{i,j\in E}$ with real valued entries is called a **stochastic matrix** if*

(i) $p_{ij} \in [0, 1] \quad \forall i, j \in E,$ *and*

(ii) $\sum_{j\in E} p_{ij} = 1 \quad \forall i \in E.$

That is to say, each row of P can be interpreted as a probability distribution on E.

Given an arbitrary Q-matrix Q, one can naturally define a stochastic matrix Π, which is called the *jump matrix* of Q.

Definition 4.2.5. *Let $Q = (q_{ij})_{i,j\in E}$ be a Q-matrix. Then one defines its **jump matrix** $\Pi = (\pi_{ij})_{i,j\in E}$ as*

$$\pi_{ij} = \begin{cases} \frac{q_{ij}}{-q_{ii}} & : i \neq j \text{ and } q_{ii} \neq 0 \\ 1 & : i = j \text{ and } q_{ii} = 0 \\ 0 & : \text{else.} \end{cases}$$

The jump matrix Π is indeed a stochastic matrix, as one verifies by elementary calculation.

We will now give the definition of a discrete-time Markov process, which will then be used in the next step to define continuous-time Markov processes as in (Norris, 1998, p.94).

Definition 4.2.6. *Let $(X_n)_{n\in\mathbb{N}_0}$ be a discrete-time stochastic process with a countable state space E and $P = (p_{ij})_{i,j\in E}$ a stochastic matrix. Let further λ be a probability distribution on E. Then $(X_n)_{n\in\mathbb{N}_0}$ is called a **discrete-time Markov process** with transition matrix P and initial distribution λ, if for all times $n \geq 0$ and states $i_0, ..., i_{n+1} \in E$ holds:*

(i) $\mathbb{P}(X_0 = i_0) = \lambda(i_0),$ *and*

(ii) $\mathbb{P}(X_{n+1} = i_{n+1} | X_0 = i_0, ..., X_n = i_n) = p_{i_n i_{n+1}}.$

Short hand for this definition is "$(X_n)_{n\in\mathbb{N}_0}$ is Markov(λ, P)". The definition comprises an interpretation of the stochastic matrix P. We stated above that each row of P forms a probability distribution on the state space E. If the process $(X_n)_{n\in\mathbb{N}_0}$ is in state $i \in E$ at some time n, then the row P_i gives the probability distribution of the *following* state, i.e, the state at time $n+1$. To put it even simpler: The matrix entry p_{ij} gives the probability that a process which is in state i at time n will be in state j at time $n+1$. This is why P is called the "transition" matrix of $(X_n)_{n\in\mathbb{N}_0}$. It gives the transition probabilities between possible states. Note that this formulation in terms of the elementary properties p_{ij} is only possible due to the countability of the state space E.

The Markov property can be extracted from this definition. It follows immediately from (ii) and can be stated as follows.

Definition 4.2.7. *A stochastic process $(X_n)_{n\in\mathbb{N}_0}$ with countable state space is said to possess the **Markov property**, if for any $n \in \mathbb{N}$, and any $i_0, ..., i_{n+1} \in E$ holds:*

$$\mathbb{P}(X_{n+1} = i_{n+1} | X_0 = i_0, ..., X_n = i_n) = \mathbb{P}(X_{n+1} = i_{n+1} | X_n = i_n).$$

This is the rigid formulation of the Markov property, which was explained before less formal: the future of the process does not depend on its past, provided its present is known. In Definition 4.2.7 the statement $X_{n+1} = i_{n+1}$ is the "future" of the process, its state at time $n+1$. Further, $X_0 = i_0, ..., X_n = i_n$ is its past, i.e., all the states it visited so far. Its present state is $X_n = i_n$. The above property states that the probability of a certain future state i_{n+1} is solely conditioned on the most recent state i_n, and independent of the former history of the process. The Markov property can be extended to processes with uncountable state spaces or continuous time parameters, and it could be used for the definition of Markov processes. This path is followed, e.g., in (Breiman, 1992, p.129,319).

But note the following. Since the setting we are interested in is that of a countable state space, every process that is regarded here is a *jump process*. That is a process, whose trajectories look like piecewise constant functions with isolated[1] jumps. Jump processes form a subclass of Markov processes. In the following, we want to present three possible definitions of Markov jump processes, all of which are equivalent and given in Norris (1998). These definitions reveal additional structure of the processes, and are therefore favored over a definition, which uses the Markov property alone. The three characterizations are called "jump chain - holding time characterization", "infinitesimal characterization" and "transition probability characterization". For the first one, we must introduce the concepts *jump time*, *holding time* and *jump chain* of a continuous-time jump process.

Definition 4.2.8. *Let $(X_t)_{t\geq 0}$ be a continuous-time jump process with countable state space E. Then the real valued random variables $(J_n)_{n\in\mathbb{N}_0}$ defined*

[1] Should the jumps not be isolated but possess accumulation points, that phenomenon is called explosion.

as

$$J_n = \begin{cases} 0 & : n = 0 \\ \inf\{t \geq J_n \,:\, X_t \neq X_{J_n}\} & : n > 0 \end{cases}$$

are the ***jump times*** *of* $(X_t)_{t\geq 0}$ *(with the convention* $\inf \emptyset = \infty$*).*
The ***holding times*** $(S_n)_{n\in\mathbb{N}}$ *are defined as*

$$S_n = \begin{cases} J_n - J_{n-1} & : J_n < \infty \\ \infty & : \textit{else.} \end{cases}$$

The ***jump chain*** $(Y_n)_{n\in\mathbb{N}_0}$ *of* $(X_t)_{t\geq 0}$ *is a discrete-time process on the same state space* E *defined by*

$$(Y_n)_{n\in\mathbb{N}_0} := (X_{J_n})_{n\in\mathbb{N}_0}\,.$$

Interpretation of these quantities is fairly easy. The jump times are just those times at which the process $(X_t)_{t\geq 0}$ "jumps", i.e., changes state, for the n-th time. They are random variables themselves. Holding times are complementary random variables, which measure the time of abidance of a process in its n-th state. The holding time S_n before the n-th jump can only be finite if an n-th jump does occur in finite time, this is the reason for the distinction of cases in the definition. Examining the jump chain of a process means looking at the process as if the jump times were purely ordinal, disregarding the time intervals and regarding only the order in which the jumps occur.

Finally one needs the notion of a right-continuous process.

Definition 4.2.9. *A continuous-time jump process* $(X_t)_{t\geq 0}$ *with countable state space* E *is called* ***right-continuous*** *if for each* $\omega \in \Omega$ *the trajectory*

$$X(t,\omega) := X_t(\omega) : \mathfrak{T} \longrightarrow E$$

is continuous from the right in any $t \in \mathfrak{T}$*.*

Now we can proceed to the three defining characterizations of continuous-time Markov processes with finite state space (Norris, 1998, p.94).

Definition 4.2.10. *Let* $(X_t)_{t\geq 0}$ *be a right-continuous process with finite state space* E*, and* $Q = \{q_{ij}\}_{i,j\in E}$ *a* Q*-matrix with jump matrix* Π*. Let further* $\lambda := X_0$ *be a probability distribution on* E*. Then* $(X_t)_{t\geq 0}$ *is called a* ***continuous-time Markov process*** *with initial distribution* λ *and generator matrix* Q*, if one of the following equivalent characterizations holds.*

(i) ***"Jump chain - holding time" characterization***
The jump chain $(Y_n)_{n\in\mathbb{N}_0}$ *of* $(X_t)_{t\geq 0}$ *is discrete-time Markov*(λ, Π) *and for each* $n \in \mathbb{N}$ *the holding times* $S_1, ..., S_n$ *are independent random variables with* $S_i \sim \mathrm{Exp}(-q_{Y_i Y_i})$*.*

(ii) ***Infinitesimal characterization***
For all $t, h \geq 0$*, conditional on* $X_t = i$*,* X_{t+h} *is independent of* $(X_s\colon s < t)$ *and, as* $h \downarrow 0$*,*

$$\sup_{t\geq 0} \mathbb{P}(X_{t+h} = j | X_t = i) = \delta_{ij} + q_{ij}h + o(h).$$

holds for all $j \in I$*.*

(iii) ***Transition probability characterization***
For all $n \in \mathbb{N}$, all times $0 \leq t_0 \leq t_1 \leq ... \leq t_{n+1}$, and all states $i_0, ..., i_{n+1}$ holds

$$\mathbb{P}(X_{t_{n+1}} = i_{n+1} | X_{t_0} = i_0, ..., X_{t_n} = i_n) = p_{i_n i_{n+1}}(t_{n+1} - t_n), \tag{4.2}$$

where $P(t) = \{p_{ij}(t)\}_{i,j \in E}$, $t \geq 0$ is the unique matrix solution of the "backward equation"

$$\frac{\partial}{\partial t} P(t) = QP(t), \quad P(0) = \mathbf{1} \tag{4.3}$$

and the "forward equation"

$$\frac{\partial}{\partial t} P(t) = P(t)Q, \quad P(0) = \mathbf{1}. \tag{4.4}$$

The symbol $\mathbf{1}$ denotes the corresponding unity matrix.

As short hand one writes "$(X_t)_{t \geq 0}$ is Markov(λ, Q)" for any process that can be characterized by one of the above conditions.

If E is countably infinite, (ii) cannot be used for the characterization of a Markov process, and the forward equation is not necessarily equivalent to the backward equation. Both properties hold only under the additional requirement of *minimality* of the process. This notion leads to the topic of explosions of jump processes, which we do not want to raise here.

The equivalence of the three characterizations is shown in (Norris, 1998, pp.94). That a Q-matrix and an initial distribution are indeed enough to construct a Markov process, and that such a process possesses the Markov property (even in a strong sense) is shown in the same place.

Each of the three characterizations has its own advantages. The jump chain - holding time characterization reduces the definition of a Markov jump process to the declaration of an initial distribution and a Q-matrix, which is very convenient when defining a stochastic process. It also explicitely decouples jump chain and waiting times, which opens up the common *Markov chain Monte Carlo* strategy for the computer simulation of a Markov jump process. The infinitesimal definition links the Markov property to the notion of differentiability. The transition probability characterization introduces a differential equation and Q as the "generator matrix", opening up for semigroup theory.

Equipped with a clear notion of a continuous-time Markov jump process, we can now proceed to introducing an example that is used in the stochastic theory of coagulation: the Marcus–Lushnikov process.

4.2.2 Definition of the Marcus–Lushnikov process

Although the Marcus–Lushnikov process (ML process) is a common notion in the literature on stochastic coagulation, there is no single generally used definition. In this subsection we give our own formal definition of the process. In the following Subsection 4.2.3 we want to discuss how this definition relates to those definitions given in the most prominent places in the literature. We

also discuss, what questions the literature dealt with and what results were achieved. In Subsection 4.2.4 we comment on one specific type of result, the connections between the ML process and the classical deterministic coagulation equation. Finally, we give a brief outlook on adaptations of the ML process and further stochastic models of coagulation in Subsection 4.2.5.

We start with the definition of the ML process as a Markov jump process. We use the characterizations of Definition 4.2.10. According to these characterizations, in order to define a specific Markov jump process, it is enough to define a state space, a Q-matrix and an initial distribution. Let us start with the sheer definition, and proceed to the interpretation afterwards.

Definition 4.2.11. *Let $N \in \mathbb{N}$. The state space of the N-particle Marcus–Lushnikov process is defined as*

$$E^{ML} := \left\{ \mathbf{n} = (n_1, ..., n_N) \in (\mathbb{N}_0)^N \mid \sum_{i=1}^{N} i \cdot n_i = N \right\}.$$

For $\mathbf{n}, \mathbf{m} \in E^{ML}$ we write $\mathbf{n} \xrightarrow{i,j} \mathbf{m}$, if there are $i, j \in \{1, ..., N\}$ with

$$\mathbf{m} = \mathbf{n} - \mathbf{e}_i - \mathbf{e}_j + \mathbf{e}_{i+j},$$

where $\mathbf{e}_i$, $\mathbf{e}_j$, $\mathbf{e}_{i+j}$ denote unit vectors in $(\mathbb{N}_0)^N$.
Let $K : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}_0^+$ be a symmetric coagulation kernel and $h \sim \frac{1}{N}$ a scaling parameter. Then the Q-matrix $Q^{ML} := \left\{ q_{\mathbf{nm}} \mid \mathbf{n}, \mathbf{m} \in E^{ML} \right\}$ of the N-particle Marcus–Lushnikov process is defined elementwise as

$$q_{\mathbf{nm}} = \begin{cases} hK(i,j)n_i n_j - \delta_{ij} hK(i,i) \left(\frac{n_i(n_i+1)}{2} \right) & : \mathbf{n} \xrightarrow{i,j} \mathbf{m} \\ -\sum_{\mathbf{k} \in E^{ML}, \mathbf{k} \neq \mathbf{n}} q_{\mathbf{nk}} & : \mathbf{n} = \mathbf{m} \\ 0 & : \textit{else.} \end{cases}$$

Let finally λ_{init} be a probability distribution on E^{ML}.
*The continuous-time stochastic process $(X^{ML})_{t \geq 0}$ that has state space E^{ML} and is Markov(λ_{init}, Q^{ML}) is called the N-particle **Marcus–Lushnikov process** with initial distribution λ_{init}.*
*If λ_{init} has the form $\lambda_{init}(N \cdot \mathbf{e}_1) = 1$, the process is called **monodisperse**, in any other case it is called **polydisperse**.*

The process is well-defined because E^{ML} is countable (even finite) and because Q^{ML} is a Q-matrix by construction.

Let us now come to an interpretation of the process as a stochastic model for particle coagulation. The monodisperse ML process describes the stochastic time evolution of a system of N uniform "unit particles" in a control volume, which are subject to coagulation, forming larger particle aggregates. We call such a system a particle *ensemble*. The ensemble can find itself in different states, i.e., all the states which have a representation in the state space E^{ML}. An element of E^{ML} is an N-tuple which encodes the information, how many particle aggregates of which "type" are present in the ensemble. The "type"

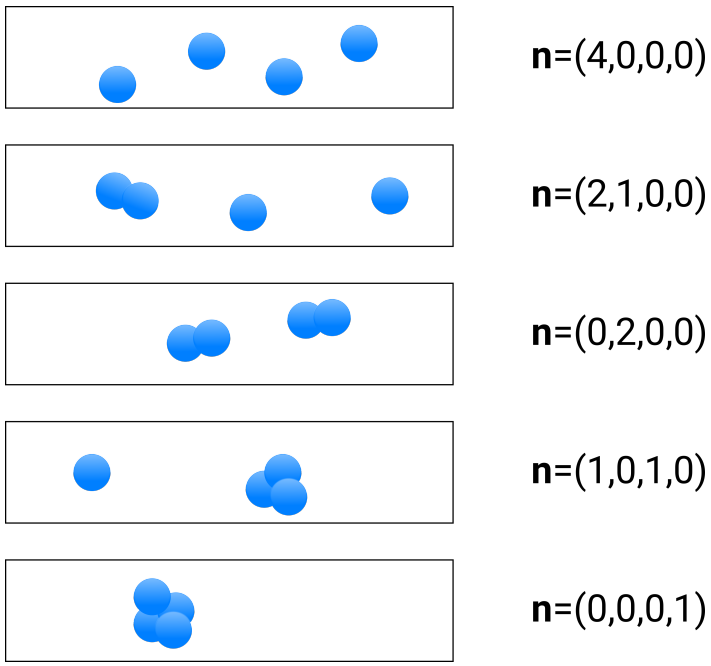

Figure 4.1: The state space E^{ML} of the 4-particle Marcus–Lushnikov process, all five possible states are illustrated and listed. Each blue circle represents a physical particle of unit mass, particles that overlap have coagulated and form a larger particle aggregate.

of the aggregate is simply the number of unit particles it consists of. This interpretation of the state space is illustrated in Figure 4.1. The number N is at the same time the number of unit particles in the ensemble and the type of the largest possible aggregate. The condition

$$\sum_{i=1}^{N} i \cdot n_i = N$$

on the elements of E^{ML} is mass conservation – the total mass in the system must be N times the mass of a unit particle at any time.

The ensemble changes state, i.e., jumps, at stochastic intervals. The jumps that are possible are those between states that are connected by the relation

$$\mathbf{n} \xrightarrow{i,j} \mathbf{m}, \tag{4.5}$$

which was given in the definition above. If two states $\mathbf{n}$ and $\mathbf{m}$ stand in that relation, it means that $\mathbf{m}$ may arise from $\mathbf{n}$ by a single two-particle coagulation, where one particle is of type i and the other is of type j. The stochastic rate at which such coagulations happen is given by $q_{\mathbf{nm}}$, the respective entry in the Q-matrix.[2] Let us have a closer look at the definition of $q_{\mathbf{nm}}$. The interesting

[2]This follows, e.g., from the infinitesimal characterization of a Markov jump process with finite state space.

case, $\mathbf{n}$ and $\mathbf{m}$ standing in Relation (4.5), means especially that $\mathbf{n} \neq \mathbf{m}$. If also $i \neq j$, the jump rate from state $\mathbf{n}$ to $\mathbf{m}$ is

$$q_{\mathbf{nm}} = hK(i,j)n_i n_j.$$

As is common in macro- and mesoscopic models of coagulation, the entire physics of the coagulation process itself is hidden in the kernel K. The ML process can be used as a stochastic framework for any reasonable coagulation kernel K, which is a big advantage. The value $K(i,j)$ gives the rate of a *specific* pair of one i-type and one j-type particle coagulating. But as the ML process does not distinguish between individual particles, this rate must be multiplied by the number of possible ij-pairings, i.e., by $n_i \cdot n_j$. The scaling parameter h is of order $\frac{1}{N}$. It keeps the total number of expected coagulations per infinitesimal time of order N, which is important if the particles in the ML process are regarded as *representatitives* of a certain number of physical particles.

For the diagonal case $i = j$, the rate expression with the Kronecker delta reduces to

$$q_{\mathbf{nm}} = hK(i,i)\frac{n_i(n_i-1)}{2}.$$

The factor $n_i(n_i-1)/2$ is just the number of possible pairings of particles of type i, the scaling factor h plays the same role as it did before.

The matrix entries $q_{\mathbf{nn}}$ give the rate of staying in state $\mathbf{n}$, compare the jump chain - holding time characterization of a Markov process. It is chosen as the additive inverse of the sum of the other entries in the same row, so as to guarantee the Q-matrix property of Q^{ML}.

The initial distribution λ_{init} gives the probabilities in which state the process starts at time 0. A monodisperse initial distribution means that the process starts from the state $N \cdot \mathbf{e}_1 = (N, 0, ..., 0)$ with probability 1. This is the state of N separate unit particles. Every initial distribution which allows for any other initial state with non-zero probability is called polydisperse, as was stated in the definition.

Let us close this subsection with an interesting point concerning the applicability of the Marcus–Lushnikov process. In its original form, as we presented it above, it does not contain any spatial information. Yet it assumes that the particles move in space, so that they can come near each other and coagulate. As those events are assumed to happen stochastically, a source of random particle movement must be present, and implicitly enter the model via the coagulation kernel K. In the case of particles embedded in a fluid environment such a source of randomness could be Brownian motion or turbulence, cf. (Marcus, 1968, p.133).

4.2.3 The process in the literature

In this subsection we want to point out some prominent places in the literature where the ML process emerged. For each of these, we will describe what version of the ML process is introduced, how it relates to our definition in Definition 4.2.11, and we will comment on the further results on the ML process gained in these original contributions.

The eponymous works Marcus (1968) and Lushnikov (1978a) introduce the state space that we denoted E^{ML}. The analysis of the process is done in terms of the probability of elementary events,

$$p_{\mathbf{n}_0\mathbf{n}}(t) := \mathbb{P}(X^{\mathrm{ML}}(t) = \mathbf{n} \mid X^{\mathrm{ML}}(0) = \mathbf{n}_0). \tag{4.6}$$

For the time evolution of these probabilities of elementary events an ad-hoc formulation of the *master equation* is set up. A master equation can be formulated for any time-continuous Markov process. It is a gain-loss equation for the probability of finding the process in a certain state $\mathbf{n}$ at time t, conditional on some initial state $\mathbf{n}_0$. The master equation set up in Marcus (1968); Lushnikov (1978a) turns out to be the forward equation in the transition probability characterization of Definition 4.2.10, with the matrix Q^{ML}. By showing this equivalence, we establish the connection between "our" process $(X^{\mathrm{ML}})_{t\geq 0}$ and the process introduced in the original works.

The master equation from the original works reads, in the notation of (4.6),

$$\frac{\mathrm{d}}{\mathrm{d}t}p_{\mathbf{n}_0\mathbf{n}}(t) = \sum_{\left\{\mathbf{m}\mid\exists i,j:\mathbf{m}\xrightarrow{i,j}\mathbf{n}\right\}} q_{\mathbf{mn}}p_{\mathbf{n}_0\mathbf{m}}(t) - \sum_{\left\{\mathbf{m}\mid\exists i,j:\mathbf{n}\xrightarrow{i,j}\mathbf{m}\right\}} q_{\mathbf{nm}}p_{\mathbf{n}_0\mathbf{n}}(t). \tag{4.7}$$

In this equation, the states $\mathbf{n}_0$, $\mathbf{n}$, and $\mathbf{m}$ are from E^{ML}, and the values $q_{\mathbf{mn}}$, $q_{\mathbf{nm}}$ are just off-diagonal entries of Q^{ML}. The equations state, loosely speaking, that the probability to find the process in state $\mathbf{n}$ at time t changes with a rate, which decomposes into gain- and loss terms. The first sum on the right-hand side contains the gain terms: the probabilities of the process at t being in some state $\mathbf{m}$, from which the state $\mathbf{n}$ can be reached by *one* coagulation event, multiplied with the rate at which such a coagulation event ought to happen ($q_{\mathbf{mn}}$). Conversely, the second sum assembles all loss terms, i.e., state probabilities and rates of leaving state $\mathbf{n}$ at time t.

Let us now show that (4.7) is a "diagonal-free" form of the forward equation of Definition 4.2.10. Firstly, note that the property (4.2) can be simplified to comprising just the two times $t_0 = 0$ and $t_1 = t$. It then reads

$$\mathbb{P}(X(t) = i_1 \mid X(0) = i_0) = p_{i_0 i_1}(t),$$

with $p_{i_0 i_1}(t)$ solving the component (i_0, i_1) of the matrix differential equation (4.4).

Choosing two arbitrary states $\mathbf{n}$ and $\mathbf{n}_0$ of the Marcus–Lushnikov process, the corresponding component equation of (4.4) is the evolution equation

$$\frac{\mathrm{d}}{\mathrm{d}t}p_{\mathbf{n_0 n}}(t) = \sum_{\mathbf{m}} p_{\mathbf{n_0 m}}(t)q_{\mathbf{mn}}. \tag{4.8}$$

Note here that the left-hand sides of (4.7) and (4.8) are equal. Both equations describe the time evolution of the probability of an elementary event, conditional on the initial state $X^{\mathrm{ML}}(0)$. The right-hand side of (4.8) can be

rearranged as follows:

$$\begin{aligned}\sum_{\mathbf{m}} p_{\mathbf{n_0 m}}(t) q_{\mathbf{mn}} &= \sum_{\mathbf{m}\neq\mathbf{n}} p_{\mathbf{n_0 m}}(t) q_{\mathbf{mn}} + p_{\mathbf{n_0 n}}(t) q_{\mathbf{nn}} \\ &= \sum_{\mathbf{m}\neq\mathbf{n}} p_{\mathbf{n_0 m}}(t) q_{\mathbf{mn}} - \sum_{\mathbf{m}\neq\mathbf{n}} p_{\mathbf{n_0 n}}(t) q_{\mathbf{nm}}.\end{aligned} \tag{4.9}$$

This holds, because Q^{ML} is a Q-matrix and thus has zero row sums. Combining (4.8) and (4.9), one can see that (4.8) is equivalent to (4.7), because from both sums in (4.9) all those terms where $\mathbf{m}$ and $\mathbf{n}$ do not stand in the coagulation relation drop out. With this, it was shown that the master equation of Marcus (1968); Lushnikov (1978a) defines exactly the process $(X^{\mathrm{ML}})_{t\geq 0}$, yet the definition in these original works is less formal. We comment briefly on the results of the papers. Proceeding from the elementary probability characterization of the process sketched above, Marcus (1968) formulates a rather unwieldy representation of $\mathbb{P}(X^{\mathrm{ML}}(t) = \mathbf{n} \mid X^{\mathrm{ML}}(0) = \mathbf{n}_0)$, which is based on the summation of all paths of $(X^{\mathrm{ML}})_{t\geq 0}$ which could possibly have led to state $\mathbf{n}$ from $\mathbf{n}_0$. In both works it is further shown that the time evolution of the expected values of the ML process are described by the classical coagulation equation, supposed the numbers of particles of different classes are uncorrelated in a certain sense. We will discuss that kind of result in Subsection 4.2.4.

Besides those two closely connected works, the same stochastic process was introduced in Gillespie (1972). There the process is examined componentwise, each particle size type n is regarded separately. Expressing it in our notation, the object of interest in Gillespie (1972) is $\mathbb{P}((X^{ML}(t))_n = m)$. This is the probability of the elementary event m of the n-component of the Marcus–Lushnikov process. Let us briefly give an outline of Gillespie (1972). For $\mathbb{P}((X^{ML}(t))_n = m)$, the author derives the master equation by basic combinatorial reasoning. This makes it necessary to find infinitesimal rates for possible coagulation jumps, those rates that appear in the off-diagonals of Q^{ML}. In his derivation, the author faces the problem that the components of the Marcus–Lushnikov process do depend on each other probabilistically, i.e., that the state of $(X^{ML}(t+\delta t))_n$ does not depend on the state of $(X^{ML}(t))_n$ alone. This is an interesting example for a more general result: isolated components of a Markov process are in general not Markov themselves, see (van Kampen, 1981, p.79). The author can thus only continue his chain of reasoning by assuming that the components are stochastically independent. This makes it possible to derive the componentwise master equations. These equations simplify considerably when ignoring diagonal coalescence, i.e., the coagulation of even-sized particles, and these simplified master equations then give rise to the classical coagulation equation. It appears as the evolution equation for the expected value.

The more recent contributions to the field, following what we called the Markovian approach in Section 4.1, like Guiaş (1997); Norris (1999); Eibeck and Wagner (2001); Deaconu et al. (2002), are mathematically very rigorous. In defining $(X^{\mathrm{ML}})_{t\geq 0}$ they employ the "jump time - holding time" characterization. Additionally, the authors choose a more general state space for the process. Instead of mapping to a subset of $\mathbb{Z}^N$, there $(X^{\mathrm{ML}})_{t\geq 0}$ maps into the

set $\mathfrak{M}^f$ of finite measures on $(0,\infty)$. Admissible as states of the ML process are then all those $\mu \in \mathfrak{M}^f$ which can be written as a sum of unit masses, see (Norris, 1999, p.95),

$$\mu = \sum_{i=1}^{p} \delta_{m_i}.$$

In this expression each summand can be interpreted as a particle of mass m_i. The process $(X^{\mathrm{ML}})_{t\geq 0}$ is then used as a tool to gain existence and uniqueness results for the coagulation equation, with different coagulation kernels K. The proofs make use of advanced stochastical and measure theoretical tools, and we refrain from restating any of these results here. Besides using the ML process as a tool for examining the coagulation equation, it appears as the central object of interest itself, as in Chapter 4 of Norris (1999), or the works Eibeck and Wagner (2001); Wagner and Eibeck (2003). In order to appreciate the results therein, which establish a connection between $(X^{\mathrm{ML}})_{t\geq 0}$ and the coagulation equation, we turn towards that type of results now.

4.2.4 Macroscopic equation, weak law of large numbers and deterministic limit

In this section we regard two types of results that establish a connection between the Marcus–Lushnikov process $(X^{\mathrm{ML}})_{t\geq 0}$ and the coagulation equation. This means that a stochastic process and a phenomenological, macroscopic differential equation get linked to each other. This cannot only be done for the ML process, but other stochastic processes allow for such a link to a deterministic equation, too. In the following paragraphs, we sketch two canonical way of establishing such a connection, and comment on some successful attempts of doing this for the ML process in the literature.

The idea of the first way is discussed in some generality in (van Kampen, 1981, pp.130). The basic idea is to show that the expected value of the process solves the macroscopic equation. Therefore, one has to derive such an equation from the forward- or master equation. We sketch the idea here.

Let an arbitrary Markov process $(Y_t)_{t\geq 0}$ with state space E describe some physical system in a stochastic manner. The process contains information on a mean value of the state of the system and information on the fluctuations around this mean.

If one starts with a deterministic description of the same system, in writing down a macroscopic equation, one expresses the hope that the stochastic fluctuations of the system around its mean are sufficiently small to describe the evolution of the system by an equation which contains only the mean value y. Information on the fluctuations around the mean is neglected in the macroscopic equation.

A link between both descriptions can by achieved by formulating the deterministic mean value y as the expected value of the process $(Y_t)_{t\geq 0}$:

$$y(t) := \mathbb{E}Y_t.$$

An evolution equation for the expected value y can be gained from the master equation of the process. If one can show that the master equation yields a

differential equation for the expected value of the process, and that differential equation is the same as the macroscopic equation, one has established the desired connection. The process introduces fluctuations into the description of a phenomenon, whose mean value is described by the macroscopic equation.

It is common knowledge in the physics community (see, e.g., Erban et al. (2007)) that a connection as sketched above can easily be derived from the master equation, if the model is linear, because the expectation is linear, too. But regarding the Marcus–Lushnikov process, not only does it have quadratic terms, which come from diagonal coagulation jumps of the form

$$(n_0, ..., n_N) \longrightarrow (n_0, ..., n_i - 2, ..., n_{2i} + 1, ..., n_N),$$

but to make things worse, all other jumps are essentially *bi*-linear. With these insights, it seems only natural that in Gillespie (1972), where a result of that type is derived, the author has to neglect diagonal coagulation (to drop the quadratic terms) and must assume stochastic independence (to get a grip on the bilinear terms). Only this enables to establish the classical connection between the ML process and the coagulation equation.

A closely related, yet more general way to establish a connection between process and macroscopic equation can be formulated in terms of a law of large numbers. The idea is made explicit in "Open Problem 3" of (Aldous, 1999, p.34). It is requested there to show for the Marcus–Lushnikov process a convergence in probability

$$X^{\mathrm{ML}}(t) \xrightarrow{P} f(t, \cdot) \quad \text{for } N \to \infty, \tag{4.10}$$

pointwise in t, where the limit $f(t, \cdot)$ is a deterministic function solving the coagulation equation. The process should belong to a general kernel K, subject only to rather weak boundedness conditions. The limiting procedure $N \longrightarrow \infty$ means letting the number of particles in a monodisperse initial distribution, i.e., the mass in the system, go to infinity. Note that at the same time the scaling parameter h goes to zero, keeping the product $h \cdot N$ constant, therefore keeping the total rate of coagulations events in $\mathscr{O}(N)$.

This limiting procedure is known as deriving the *hydrodynamic limit* of the microscopic model. Naming results of this type "weak law of large numbers" (WLLN) might be somewhat misleading, if one has the classical weak law of large numbers in mind. The term "large numbers" here does not refer to independent random variables with the same probability distribution, but to the large number of unit particles, and its relation to the classical WLLN is due to the type of convergence, i.e., convergence in probability. Another parallel is the identification of the limiting function as a function solving the coagulation equation - as was explained in the preceding paragraph, this is a property which one expects from the expected value. In classical WLLN results, the limit would be some common expected value.

A classical paper containing results in the WLLN direction for the ML process is Hendriks et al. (1985). A more recent line of development includes Jeon (1998); Norris (1999); Eibeck and Wagner (2001); Fournier and Giet (2004). An interesting contribution is Jacquot (2010), which establishes a hydrodynamic

limit for a related stochastic process, whose state space is able to keep track of the history of each single particle agglomeration.

Finally, we want to share some thoughts on the usefulness of results of the discussed type. First of all, the connection between process and coagulation equation is useful for the mutual verification of the models. Whether the process should justify the equation, or the equation should justify the process, depends on the viewpoint. Because the Smoluchowski coagulation equation is the historically earlier model, the convergence of $(X^{\mathrm{ML}})_{t\geq 0}$ to its solutions can be used to justify that the ML process is a valid coagulation model. On the other hand, since $(X^{\mathrm{ML}})_{t\geq 0}$ is a convincing microscopic model of coagulation, and richer in detail (fluctuations!) than the coagulation equation, its convergence behavior can be used to justify the macroscopic equation with equal rights.

Another use of those results are convergence proofs for computer algorithms based on the Marcus–Lushnikov process. They are of interest for showing that these algorithms yield a solution to the actual coagulation equation. Especially WLLN-type results can be used to show that by increasing the number N of computational particles, one has reason to expect a convergence towards the solution of the coagulation equation. Before we have a closer look on a class of simulation algorithms for the ML process (see Section 4.3), we want to give a short outlook beyond the subject of the current section.

4.2.5 Beyond the Marcus–Lushnikov process

It goes without saying that the Marcus–Lushnikov process is not the only stochastic model for coagulation. Yet the process is strikingly elegant, because it is at the same time very simple and very general. It is simple, because its construction blends very smoothly into the theory of time-continuous Markov jump processes. Also its state space is fairly simple, and the definitions of its jumps and jump rates are very comprehensible. One the other hand, it is rather general. This is mainly, because it is constructed in the same way for any kernel K and for any initial distribution λ_{init}. Another trait that contributes to its generality is its adaptability. In defining additional jumps, further phenomena as fragmentation (Gueron (1998); Guiaş (1997); Norris (1999)), insertion (Patterson and Wagner (2012)) and particle removal can be included. Note that particle insertion in general makes use of an *in*finite (yet countable) state space.

On top of that, the process can be extended to more general state spaces. Choosing a more general state space enables one to use more complex particle descriptions. When one extends it by more sophisticated constructions, phenomena like spatial inhomogeneity, diffusive and/or convective transport are within range. Of course, these modifications will make it hard to recognize the original ML process in it. A very interesting theoretical approach in that direction is the application of the theory of "piecewise-deterministic Markov processes" from Davis (1993) in Patterson (2013), which allows for a deterministic change of the process in between jumps, as do appear under the transporting effect of a laminar flow field.

One might ask, up to which point it is still just to call a stochastic model of coalescence a Marcus–Lushnikov process. To us it seems a good idea to keep

$(X^{\mathrm{ML}})_{t\geq 0}$ in mind as a "pivot", starting from which one can understand and classify other stochastic coagulation models.

A relatively recent overview article on these is Berestycki (2009), which discusses several stochastic models apart from $(X^{\mathrm{ML}})_{t\geq 0}$ with applications in population genetics. See also the very comprehensible constructions in (Aldous, 1999, Chps. 3,4) for special coagulation kernels, which give an idea, what possibilities there are in the stochastic theory of coagulation beyond the Marcus–Lushnikov process.

4.3 Stochastic simulation algorithms

In this section we want to present in detail the stochastic simulation algorithm (SSA), which we employ for the solution of population balance equations. The SSA in its many recent forms is the result of more than forty years of combined research effort. It has been improved and extended by different research groups on different occasions, and with different scopes of application in mind. It was developed further within different communities, and different names were coined for it, among them "Kinetic Monte Carlo", "Dynamic Monte Carlo", "Direct Simulation Monte Carlo", "Population Balance Monte Carlo" or "Gillespie SSA". The last of these names gives credit to Gillespie (1977), where the algorithm was introduced as a tool for the simulation of reactive chemical systems. This work is still exceedingly well-cited in the chemical community. Two years earlier, in Gillespie (1975), the same author presented a similar stochastic algorithm for the simulation of water droplet growth in a cloud. This simulation approach stands in direct relation to the Marcus–Lushnikov process, and it is the basis of the simulation algorithm we use. As was sketched above, the algorithm underwent several changes and improvements. By now it has a "layered" appearance, with different layers of improvement and enhancement shining through.

In order to keep the balance between comprehensibility and completeness of presentation, we will present each of the major improvements separately. We start with the basic, Direct Simulation Monte Carlo (DSMC), version in Subsection 4.3.1, setting it in relation to the Marcus–Lushnikov process and Markov jump processes in general. In Subsections 4.3.3 to 4.3.5 we present the major extensions and algorithmic improvements one after the other, basing each of them on the simple DSMC algorithm. By proceeding in that manner, we hope to keep the presentation well-ordered and more comprehensible, than by stating the complete SSA in its entirety at once.

4.3.1 The direct simulation Monte Carlo algorithm

The direct simulation Monte Carlo (DSMC) algorithm, as it was introduced in Gillespie (1975) and described, e.g., in Patterson and Wagner (2012), builds on a very fundamental insight on Markov jump processes. As can be seen from the "jump chain - holding time" - characterization in (4.2.10), in any Markov jump process the jump chain $(Y_n)_{n\in\mathbb{N}}$ and the holding times $(S_n)_{n\in\mathbb{N}}$ are independent

of each other. In other words: *what* jump will occur next and *when* it will occur are independent and can therefore be simulated separately.

The state space of the DSMC encompasses individual particles. The state of the simulated particle ensemble at some time t is represented as

$$X^{\mathrm{SSA}}(t) = \big(m_i(t)\big)_{i=1,\dots,N(t)}.$$

Here, contrary to our notation of the Marcus–Lushnikov process, $N(t)$ is not the number of "unit" particles, but the number of individual particles of any size that are present in the system at time t. Therefore $N(t)$ may vary in time, this does not run contrary to mass balance. The values m_i are the internal coordinates of the particles, for simplicity picture once more $m_i \in \mathbb{N}^+$ and m_i describing the mass of particle i.

Note for the sake of completeness that the state of the Marcus–Lushnikov process $X^{ML}(t)$ can be recovered from $X^{\mathrm{SSA}}(t)$ by counting the number of individual particles of the same masses. For some mass $m \in \mathbb{N}^+$ the relation is

$$\big(X^{ML}(t)\big)_m = \sum_{i=1}^{N(t)} \delta\Big(\big(X^{\mathrm{SSA}}(t)\big)_i, m\Big).$$

In that sense, the Marcus–Lushnikov process is a cumulated version of the SSA process. The individual-particle approach of the SSA has some advantages. Firstly, it allows for non-integer or multi-dimensional particle descriptions straightforwardly, by replacing the inner coordinate space of the m_i. Secondly, it allows for modern object-oriented implementation, as each individual particle can be realized as an instance of a general particle class.

The most basic DSMC algorithm starts from an initial state $X^{\mathrm{SSA}}(0)$ that was sampled according to some initial distribution λ_{init}, and then advances the state step by step, computing next jump time and next state alternately, until some end-time t_{end} is hit. The only possible jumps are coagulation jumps. In the individual-particle state space those jumps take the form

$$(m_I), (m_J) \longrightarrow (m_I + m_J), \tag{4.11}$$

and a renumbering of the particles will be necessary afterwards. Also, the jump reduces the current total number of particles $N(t)$ by 1. The coagulation of two distinct particles I and J takes place at rate

$$\lambda_{coag}(I, J) = hK(m_I, m_J),$$

with coagulation kernel K and scaling parameter h as in the definition of the ML process. Underlying here is the assumption that all possible coagulations are independent of each other. This implicit assumption was also present in the formulation of the Marcus–Lushnikov process. The common justification is that the overall number of particles is sufficiently large, so that interdependencies of coagulation events can be neglected.

The pairwise coagulation rates give the non-zero entries of the jump matrix Π^{SSA}, in that row which belongs to the current state of the system. Note that

the rows and columns of this jump matrix are indexed with *states* of X^{SSA} and *not* with individual particle numbers. Which of the possible jumps will happen next is discretely distributed according to the elementary probabilities

$$p_{IJ} := \mathbb{P}(I \text{ and } J \text{ coagulate}) = \frac{\lambda_{\mathrm{coag}}(I,J)}{\lambda_{\mathrm{coag}}}, \tag{4.12}$$

where λ_{coag} equals the sum of the individual jump rates:

$$\lambda_{\mathrm{coag}} := \frac{1}{2} \sum_{i=1}^{N(t)} \sum_{\substack{j=1 \\ j \neq i}}^{N(t)} \lambda_{\mathrm{coag}}(i,j).$$

The factor $\frac{1}{2}$ rules out double counting due to the symmetry of K. The holding time of the process is exponentially distributed with parameter λ_{coag}. Let $\tau_0 = 0$ and let τ_k be the time of the k-th jump, then the current holding time $\tau(t) := \tau_{k+1} - \tau_k$ is distributed as

$$\tau(t) \sim \mathrm{Exp}(\lambda_{\mathrm{coag}}). \tag{4.13}$$

After each step, the jump rates λ_{coag} and their sum have to be recalculated. With this, we have all ingredients assembled, and the basic version of the SSA is given in pseudo-code style as Algorithm 3. Most details of the algorithm will be examined in the following subsections. We will not go into detail on how random variables of a certain distribution get machine-generated, since implementations of such algorithms are available in wide-spread programming libraries like C++ boost, and based mainly on transformations of uniformly distributed random numbers. Algorithms of that kind are described, e.g., in Chapter 2 of Asmussen and Glynn (2007).

Based on Algorithm 3 we will now proceed to presenting those details and major enhancements that constitute our version of the SSA.

4.3.2 Majorant kernels and reduction of computational complexity

The only jump type that is present in the basic SSA, coagulation, is of quadratic (or rather: bilinear) nature. By that we mean that, just like in the deterministic coagulation equation, terms that are bilinear in the unknowns appear, because every coagulation event naturally involves two partners. This leads to big computational effort, especially in Step 5 of Algorithm 3, when programmed naively. The first algorithmic improvement which we want to present here was introduced in Eibeck and Wagner (2001). It transfers a common scheme of stochastic simulation, the acceptance-rejection method, to the SSA. Its key ingredient is a majorant kernel $\hat{K}$, whose summation is less expensive than that of K. As an illustrative example should serve the additive coagulation kernel

$$K(m_i, m_j) = m_i + m_j.$$

Step 5 in the SSA requires the computation and summation of all possible jump rates. This requires $\mathcal{O}(N(t)^2)$ operations (we skip the dependence on t in the

Algorithm 3 The basic version of the direct simulation Monte Carlo algorithm (DSMC or SSA).

Input: end time t_{end}, initial distribution λ_{init}
Step 0.1: Simulate initial state $X^{\text{SSA}}(0)$ according to λ_{init}, set $t := 0$.
Step 0.2: Compute and store all individual jump rates $\lambda_{\text{coag}}(i, j)$.
Step 1: Generate holding time τ according to (4.13).
if $t + \tau \geq t_{\text{end}}$ **or** $N(t) = 1$ **then**
terminate
else
Step 2: Generate pair (I, J) of coagulating particles according to (4.12).
Step 3a: Set $t := t + \tau$, update state $X^{\text{SSA}}(t)$ according to (4.11).
Step 3b: Do necessary renumbering of particles.
Step 4: Update those jump rates $\lambda_{\text{coag}}(i, j)$ that depend on former I or J.
Step 5: Compute sum λ_{coag} of jump rates.
goto Step 1.
end if

following occurrences of Landau notation). With an overall expected number of jumps in the order of N, this leads to a computational complexity of $\mathcal{O}(N^3)$, which is not satisfactory.

If we regard the Marcus–Lushnikov case $m \in \mathbb{N}^+$ and additionally assume $m \geq 2$, then a simple majorant kernel is

$$\tilde{K}(m_i, m_j) := m_i m_j.$$

Indeed $K(m_i, m_j) \leq \tilde{K}(m_i, m_j)$ holds for all $m_i, m_j \geq 2$. Connected to the majorant kernel are the majorant jump rates

$$\tilde{\lambda}_{\text{coag}}(I, J) := \frac{\tilde{K}(m_I, m_J)}{N}$$

with scaling parameter N. The majorant kernel $\tilde{K}$, and with it the majorant jump rates, sum in $\mathcal{O}(N)$, because

$$\tilde{\lambda}_{\text{coag}} := \frac{1}{2N} \sum_{i \neq j} m_i m_j = \frac{1}{N} \left(\sum_{i=1}^{N(t)} m_i \right)^2 - \frac{1}{N} \left(\sum_{i=1}^{N(t)} m_i^2 \right),$$

can be computed in $\mathcal{O}(N)$ operations. The main idea of the acceptance-rejection scheme is to determine jump times according to this majorizing rate, and then decide for each event whether it is a "real" jump or a numerical artifact, a "fictitious" jump. The first stage of this process is to decide, which pair of particles (I, J) is involved in the jump. This is sampled according to the majorizing elementary property

$$\tilde{p}_i := \frac{m_i}{\sum_{i=1}^{N(t)} m_i}$$

for the first coagulation partner I, and the same for the second coagulation partner J. Both steps can be performed in $\mathcal{O}(N)$.[3] The second stage is the decision, whether the current jump is real or fictitious. The (I, J)-jump is rejected as fictitious with probability

$$\mathbb{P}_{\text{fict}} = 1 - \frac{\lambda_{\text{coag}}(I, J)}{\tilde{\lambda}_{\text{coag}}(I, J)}. \tag{4.14}$$

The acceptance-rejection scheme changes the overall order of Algorithm 3, and it changes Steps 4 and 5. Instead of updating all involved jump rates, the computation of $\tilde{\lambda}_{\text{coag}}$ is the first task. Then, after a jump time has been determined and two coagulation partners were chosen, only for (I, J) must the real coagulation rate $\lambda_{\text{coag}}(I, J)$ be determined, in order to evaluate (4.14). If the jump is identified as fictitious, only the time t is updated, and the state of the system is left unchanged.

The extra cost for the procedure is connected to the number of fictitious jumps that are introduced. One has to make sure that for a given kernel K the majorant kernel $\tilde{K}$ is chosen such that the acceptance efficiency

$$\frac{\lambda_{\text{coag}}(I, J)}{\tilde{\lambda}_{\text{coag}}(I, J)}$$

is large enough. If it is too small, the additional work introduced with fictitious jumps exceeds the savings due to the majorant scheme, i.e., a locking phenomenon occurs.

A propose of a majorant for the Brownian coagulation kernel, with some asymptotic analysis of the acceptance efficiency and some numerical examples can be found in Goodson and Kraft (2002). A nice write-up for a more general kernel is presented in Patterson et al. (2011).

In the latter reference, another aspect for the reduction of the computational complexity, going back to Patterson (2007), is described. We will skip the details here, and only give a coarse overview. The amendment consists in introducing a binary tree data structure for the state of the ensemble. In that data structure each particle is represented by a leaf of a binary tree, and the non-leaf nodes contain summed up particle properties. That data structure can be initialized in $\mathcal{O}(\log N)$ operations, and allows for all further necessary operations to be performed in either $\mathcal{O}(1)$ or $\mathcal{O}(\log N)$.

With a total number of coagulations below $N(0)$ (remember that each coagulation event reduces the number of particles by 1), the total computational complexity is $\mathcal{O}(N \log N)$, if both algorithmic improvements are combined. This is a huge gain compared to the $\mathcal{O}(N^3)$ of the basic SSA.

4.3.3 The stochastic weighted algorithm

The basic DSMC experienced a major improvement in terms of variance reduction, when it was reformulated as the "Stochastic Weighted Algorithm" (SWA). The algorithm was initially introduced as "Mass Flow Algorithm" in Eibeck and

[3] Even if programmed naively.

Wagner (2001), the term SWA appears in Patterson et al. (2011), where also a very accessible formulation of the algorithm can be found. Another beneficial source on the subject is Kolodko and Sabelfeld (2003), where a class of weighted methods is introduced. A substantial literature overview on the subject can be found in Chapter 1 of Patterson and Kraft (2007). Two more recent proposals in the same direction are the "Weighted Flow Algorithm" of DeVille et al. (2011) and "Differentially-Weighted Monte Carlo" of Zhao et al. (2010).

Particle weighting is a trait that is implicitly present even in the basic version of the Marcus–Lushnikov process and the DSMC. There, the coagulation jump rate is weighted with a parameter h that scales with $\frac{1}{N}$. This scaling reflects the fact that even if more particles are incepted into the ensemble, its density is kept constant. This constant density shows itself in the property that the number of effective particle collisions is $\mathcal{O}(N)$. What is actually happening is that the volume of the system is raised, when more particles are added into the ensemble. This point is interesting. One has to bring to mind that the ensemble which the Marcus–Lushnikov process models and the DSMC simulates is a *representative* or *control* volume $V_{control}$ of a larger, well-mixed particle containing system like a cloud or a sooting flame. The number of particles present in the system is a computational choice, their density is enforced by the physics, and this relation is mediated by the scaling parameter h.

Going one step further, each particle in the ensemble can be understood as a purely *computational* particle, representing just as many physical particles as the relation of control volume and physical volume. Following this perception, one can ask why not each computational particle should represent a *different* number of physical particles.

This idea is made explicit in the SWA and related schemes. The SWA of Patterson et al. (2011) extends the algorithmic state space by tagging each computational particle with a real-valued weight $w_i \in [0, w_{max}]$, which can be interpreted as the number of physical particles per unit volume, which the computational particle represents. The state of the process at time t is then

$$X^{\mathrm{SWA}}(t) = (m_i, w_i)_{i=1,\dots,N(t)}.$$

The weighting changes the coagulation jumps, both the rate and the form of the update. Instead of removing a computational particle, a shifting of weight from the second to the first particle is performed, the second particle is left unchanged. The (I, J)-coagulation jump in weighted form is

$$\big(m_I, w_I\big), \big(m_J, w_J\big) \longrightarrow \big(m_I + m_J, \gamma(m_I, w_I, m_J, w_J)\big), \big(m_J, w_J\big).$$

A proposal for a symmetric weight shift function that ensures mass conservation is

$$\gamma(m_i, w_i, m_j, w_j) = \frac{w_i w_j}{w_i + w_j}.$$

See Patterson et al. (2011) for the proposal of a whole class of such shifting functions γ. The jump rate of an individual coagulation jump is

$$\lambda^{\mathrm{SWA}}_{\mathrm{coag}}(I, J) = hK(m_I, m_J)w_J,$$

where only the weight of the second particle enters. Besides the changing of the state space and the re-definition of the coagulation jump, the SWA proceeds just like the DSMC, and the techniques presented in the former (and following) subsections can be applied to it, with the necessary modifications.

An advantage of the SWA is the variance reduction (proven first in Eibeck and Wagner (2001)), which becomes especially apparent when advection is involved (see Subsection 4.3.5). The reason is that in the basic SSA large areas of the particle mass spectrum tend to "depopulate", which makes the algorithm prone to stochastic noise. In the SWA, this "depopulation" is prevented, since each coagulation event changes just one particle.

Another advantage is that the number of computational particles can be kept constant in the SWA. This is convenient from a computational viewpoint, because no additional measures to keep the number of computational particles in a convenient regime must be taken. On top of that, the data structure which administers the computational particles is easier to handle if the number of particles does not change, as, e.g., no renumbering is required.

4.3.4 Linear process deferment

Extending the basic SSA to particle systems which include more processes than just coagulation is, in principle, straightforward. In this subsection we want to present a numerical trick to deal in particular with particle surface reaction processes. Surface reaction processes are *linear* in that sense that the terms which describe them mathematically involve just one particle, usually to the first power.

In many model systems, linear processes happen a lot more frequently than bilinear processes. On the other hand, one occurrence of a linear process leads to only slight changes of the affected particle, when compared to coagulation. This observation sparked the idea of Patterson (2007), to "defer" specific linear processes. This strategy distinguishes the "Linear Process Deferment Algorithm" (LPDA).

In Patterson (2007) one finds an in-depth description in measure theoretic formulation, and a numerical comparison to a less involved operator-splitting approach, both methods applied to the simulation of soot formation in model flames. The measure-theoretic formulation of the DSMC dates back to Eibeck and Wagner (2001), and allows for better analysis of the convergence behavior towards a deterministic limes. We will not switch to that formulation here, but stick to our simpler notation.

Let an additional (linear) process R (for reaction) be present in the system. A single occurrence of a reaction jump changes a single particle I:

$$(m_I) \longrightarrow (R(m_I)).$$

The rate of this specific jump, $\lambda_{\text{reac}}(I)$, depends on just one particle. The sum of all reaction jump rates of all particles in the system is

$$\lambda_{\text{reac}} := \sum_{i=1}^{N(t)} \lambda_{\text{reac}}(i).$$

In a straightforward DSMC approach, the holding time of the new ensemble would be simulated as an exponentially distributed random variable with the summed parameter

$$\lambda := \lambda_{\text{reac}} + \lambda_{\text{coag}},$$

and the type of the next jump would be determined by sampling according to

$$\mathbb{P}(\text{reaction jump}) = \frac{\lambda_{\text{reac}}}{\lambda}, \text{ and } \mathbb{P}(\text{coagulation jump}) = \frac{\lambda_{\text{coag}}}{\lambda}.$$

In the LPDA, the linear processes (or rather: those that are chosen for deferment) are disregarded in the computation of the holding time, and made good for as soon as a particle is chosen for a non-deferred jump. In the implementation, all particles must hold a time tag of their last update. Whenever two particles (I, J) are chosen for a coagulation event, the linear processes which they might have experienced since their last update are performed. Updates of deferred events should also be performed just before output about the particle ensemble is written.

The reduction of computing time when applying this method is greatest when the frequency of linear processes dominates bi-linear ones. In Patterson (2007), three orders of magnitude could be gained in terms of computing time for certain simulations of sooting flames, without a severe loss in accuracy.

4.3.5 Spatial inhomogeneity and advective transport

The SSA and its variants have been extended recently to cover systems with spatial inhomogeneity, and especially advective transport by an external flow field. Examples are Zhao and Zheng (2013), as well as Lee et al. (2015) and Patterson and Wagner (2012). For our short presentation of the key points, we stick to the latter source, though the main idea is the same in all cases. It goes back to the Bird algorithm for the Boltzmann equation (Bird (1970)) and consists essentially in a splitting scheme, splitting the particle transport ("free streaming") and the particle interaction step. Consider a constant splitting time Δt_{split}. In the original Boltzmann application, the interaction step meant computing particle collisions, where particles exchanged momentum. In our Smoluchowski coagulation framework, the interaction step means computing particle coagulations, and, in extended versions, other particle processes.

Some fundamental alterations must be conducted in order to adapt the SSA to spatial inhomogeneity. The particle state space is complemented by a bounded spatial domain $\Omega \subseteq \mathbb{R}^d$, $d \in \{1, 2, 3\}$, which must then be discretized into a finite number of physically sensible and computationally feasible compartments K_j, $j = 1, ..., M$. In Patterson and Wagner (2012) the spatial domain is one-dimensional, and the compartments are just equally spaced sub-intervals.

Each computational particle gets tagged with a spatial coordinate $\boldsymbol{x}_i \in \Omega$. All particles whose spatial coordinate is located in the same cell K_j are regarded as one *ensemble*, and during the particle interaction step only particles within the same ensemble may interact. The state of the entire process at time t is

now

$$X^{\mathrm{SSA}}(t) = \bigcup_{j\in 1,\dots,M} (m_i^j, \boldsymbol{x}_i^j)_{i=1,\dots,N_j(t)},$$

where the coordinates of all particles in the same ensemble K_j were marked with superscript j.

An external flow field $\boldsymbol{u}$ is responsible for the transport of the particles. In the free streaming step, each computational particle is moved as if following the trajectories of the numerical velocity field $\boldsymbol{u}$ (up to first order)

$$\boldsymbol{x}_i^j \longrightarrow \boldsymbol{x}_i^j + \Delta t_{\mathrm{split}} \boldsymbol{u}(\boldsymbol{x}_i^j).$$

This transport step, inherited from the Bird algorithm, is performed for each particle, and whether the relocated particle does still belong to the same cell must be checked. If not, the particle must be relocated to the ensemble of the new cell.

The particle interaction step poses the difficulty that whenever two particles coagulate, the position of the new particle must be determined, without unwanted side effects. Two examples for such unwanted side-effects were noted in Patterson and Wagner (2012). If inserting the new particle, e.g., at the midpoint in between the two former positions, the particles amass more and more at the midpoint of the cell. Also, coagulation will "pull back" mass from the out-streaming periphery of a cell towards its in-streaming boundary, therefore numerically slowing down the particles. To avoid these side-effects, Patterson and Wagner (2012) proposed choosing a new particle position $\boldsymbol{y}$, of a particle which emerged from coagulation of the particles $\left(m_I, \boldsymbol{x}_I\right)$ and $\left(m_J, \boldsymbol{x}_J\right)$ stochastically, distributed according to the elementary probabilities

$$\mathbb{P}(\boldsymbol{y} = \boldsymbol{x}_I) = \frac{m_I}{m_I + m_J}, \text{ and } \mathbb{P}(\boldsymbol{y} = \boldsymbol{x}_J) = \frac{m_J}{m_I + m_J}.$$

This stochastic center-of-mass approach mitigates the two issues addressed above. The necessary changes when combining SWA and spatial inhomogeneity are addressed in the same work.

The SSA which we employ in our coupled method comprises all the extensions and algorithmic improvements which were described above. Note that combining all of them at once into one algorithm brings additional complexity, and adaptations are necessary. How should the stochastic weight of particle that moves into another cell be re-computed? How does one formulate the LPDA when fictitious coagulation jumps are present? We refrain from answering all of these questions here. The simulation framework *Brush*, developed at the Chemical Engineering Department of Cambridge University, contains a combination of all the algorithmic variants which we described above. Instead, we now move on to the coupling of that algorithm with advanced finite element methods for the external fluid quantities, which is the main undertaking of this thesis.

5 The Coupled Algorithm for Population Balance Systems

The aim of this chapter is to pinpoint and describe the type of population balance systems on which this work focuses, and outline the coupling algorithm which we use for their numerical solution. We intend this description to be so general that it can serve as a foundation to the simulation projects in the following chapters and can easily be transferred to similar problems.

The coupled model can be used to represent a system of species and particle populations that are transported in a flow and interact with each other. The interaction phenomena which are included in the basic model are particle growth by surface attachment (subsequently: *growth*) and particle collision growth (*coagulation*). Other phenomena that the model could be adapted to include dissolution and breakage of particles, particle nucleation and reaction of transported species with each other.

In the basic version of the model system five macroscopic quantities appear. These are the fluid velocity $\boldsymbol{u}$ and fluid pressure P, a species concentration c, the temperature T and a particle size distribution f. One must keep in mind that the species described by c will be that same species which the particles described by f consist of, but in dissolved, i.e., *fluid*, state.

The domain of the unknown functions consists of a spatial and a temporal part, and, in the case of f, of an inner coordinate space that describes the particles. We denote the spatial domain (*physical space*) by $\Omega_{\boldsymbol{x}} \subseteq \mathbb{R}^d$ with $d \in \{2, 3\}$. The time interval is $\Omega_t = (0, t_{\text{end}})$, with an arbitrary end time t_{end}. The spatial unit is m, the temporal unit s. The inner coordinate space (*property space*) is denoted by Ω_m. The type of property space is a defining feature of the particular model. Multi-dimensional property spaces are conceivable, discrete or continuous ones, the space can be bounded or unbounded. As before, we stick to a univariate, unbounded, continuous property space: $\Omega_m = (0, \infty)$ and interpret the inner coordinate as particle mass (in kg).

To fix ideas, let us comment on the units of the functions. Each component of the fluid velocity $\boldsymbol{u}$ is given in $^{\text{m}}/_{\text{s}}$, the pressure P in Pa. The species concentration is a molar concentration ($^{\text{mol}}/_{\text{m}^3}$). Temperature T is given in K. The function f is a number density function defined on the particle state space $\Omega_t \times \Omega_{\boldsymbol{x}} \times \Omega_m$. In our mass-based approach it would be more appropriate to call f the particle *mass* distribution, yet the notion particle *size* distribution (PSD) is more convenient. The unit of the PSD is $^{1}/_{\text{m}^3 \cdot \text{kg}}$.

The system consists of four partial differential equations[1] and one integro-differential equation. These are the Navier–Stokes equations for $\boldsymbol{u}$ and P,

[1] Counting the Navier–Stokes equations as two.

convection-diffusion-reaction equations for T and c, and a population balance equation for f.

The equations are instationary and the spatial domain is bounded. Therefore initial and boundary conditions will have to be supplied for each quantity. Boundary conditions depend heavily on the example, therefore we will not incorporate their formulation in the following presentation of the basic model, but postpone them to the concrete numerical examples.

In the basic model, the coupling of the equations occurs by the terms that are responsible for particle surface growth. Particle surface growth is strongly dependent on temperature and species concentration in the surrounding fluid. Therefore it links the quantities c, T, and f mutually to each other.

The model which we use is essentially that one of Hackbusch et al. (2012) and Suciu (2013). The main differences are that we choose particle mass instead of particle diameter as the internal coordinate and that we reformulate the growth term in the PBE in order to make it more inclined to the stochastic formulation. In the remainder of this chapter we will write down the macroscopic model equations and give an interpretation for each of them (Section 5.1), explain their inherent coupling (Section 5.2), and finally give our coupled algorithm in a pseudo-code like fashion (Section 5.3).

5.1 The constituent equations

In this subsection we present the constituent equations and point out some modeling concerns connected to them. For details on their numerical treatment, we refer to the respective chapters of this thesis.

5.1.1 Velocity field

In general, the velocity field $\boldsymbol{u}$ is determined by the full, instationary, incompressible Navier–Stokes equations. For certain examples, simplifications of the NSE can be chosen, for example a stationary version of the equations for laminar flows or the Stokes equations for creeping flows. Several academic examples, like plug flows or pipe flows, admit choosing an analytic expression for $\boldsymbol{u}$. The full incompressible Navier–Stokes equations and their numerics are treated in detail in Chapter 2 of this thesis. In their dimensioned form they read

$$\begin{aligned} \frac{\partial}{\partial t}\boldsymbol{u} - \nu\Delta\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u} + \nabla\frac{P}{\rho} &= \frac{\mathbf{f}}{\rho} \quad \text{in } (0,T]\times\Omega, \\ \nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } (0,T]\times\Omega. \end{aligned}$$

Besides the vectorial fluid velocity $\boldsymbol{u}$ the fluid pressure P appears as an unknown. The parameter ν $[\mathrm{m}^2/\mathrm{s}]$ is the kinematic viscosity of the fluid and ρ $[\mathrm{kg}/\mathrm{m}^3]$ its density. These parameters are assumed to be constant in space and time.

Note that none of the other unknown quantities, c, T, or f, appears anywhere in these equations. In our model the fluid velocity is connected to the other quantities only by a one-way coupling. The fluid flow field is responsible for the

transport of the other quantities, but it is not influenced by their state in any way. This is a modeling decision, and it is typically justified with the relatively small size of the transported particles and the low fluctuations of concentration and temperature. Excluding such backcoupling is also an economic decision. The direct numerical simulation of the NSE is computationally very expensive, and therefore it is occasionally very beneficial, if it does not have to be recomputed at each time step but can, e.g., be computed and stored in a pre-step.

Still, the fluid velocity plays a role beyond the pure transport of the other quantities. Namely, it is the driving force behind particle collision, and the modeling of the coagulation (collision growth) must be done accordingly. This means choosing a coagulation kernel that fits the physical properties of the flow.

5.1.2 Fluid temperature and species concentration

The basic model further contains two convection-diffusion equations. They describe quantities that are transported by the fluid flow with advection velocity $\boldsymbol{u}$, and are subject to diffusion. The first such quantity is the temperature T [K]. The energy balance reads

$$\frac{\partial}{\partial t}T - D_T \Delta T + \boldsymbol{u} \cdot \nabla T = g_T I_{\text{growth}}(c, T, f) \quad \text{in } (0, t_{\text{end}}) \times \Omega_{\boldsymbol{x}}. \tag{5.1}$$

The second quantity is a species concentration c [$^{\text{mol}}/_{\text{m}^3}$]. Let it for our purposes denote the *molar* concentration of the solute, although different concentration measures could be chosen. Its convection-diffusion equation is

$$\frac{\partial}{\partial t}c - D_c \Delta c + \boldsymbol{u} \cdot \nabla c = g_c I_{\text{growth}}(c, T, f) \quad \text{in } (0, t_{\text{end}}) \times \Omega_{\boldsymbol{x}}. \tag{5.2}$$

From a mathematical point of view, T and c play very similar roles in the model system, both of them act as transported species. Numerical methods for convection-diffusion equations as (5.1) and (5.2) are described in Chapter 3 The parameters D_T [$^{\text{m}^2}/_{\text{s}}$] and D_c [$^{\text{m}^2}/_{\text{s}}$] are constant diffusion parameters. The terms that contain the fluid velocity $\boldsymbol{u}$ model the advective transport. On the right-hand side, I_{growth} [$^{\text{kg}}/_{\text{m}^3 \cdot \text{s}}$] is a term that measures the intensity of the particle growth by attachment of dissolved material. It depends on all transported quantities, i.e., on c, T, and f, and on the spatial coordinate $\boldsymbol{x}$. The term manifests itself as an integral over the property space Ω_m, having the form

$$I_{\text{growth}}(c, T, f, t, \boldsymbol{x}) = \int_{\Omega_m} G(c, T, m) f(t, \boldsymbol{x}, m) \, \mathrm{d}m. \tag{5.3}$$

A particle growth model must be chosen for the growth rate G. Proper choices of such models depend on the example, see the chapters on numerical examples for some options.

The constant scaling parameters g_T [$^{\text{K} \cdot \text{m}^3}/_{\text{kg}}$] and g_c [$^{\text{mol}}/_{\text{kg}}$] scale the influence of the growth intensity on the respective quantity linearly. In those parts of $(0, t_{\text{end}}) \times \Omega_{\boldsymbol{x}}$ where particle growth appears, I_{growth} will be positive. In the remaining parts it is zero, thus $I_{\text{growth}} \geq 0$. Since particle growth by attachment

goes along with "consumption" of the dissolved material, we expect growth to lead to a sink in Equation (5.2), thus $g_c < 0$. At the same time energy will be released when material crystallizes, i.e., molecule bonds are formed, and thus energy released. This is why growth leads to a source term on the right-hand side of (5.1), $g_T > 0$. That is to say, the formation of crystals from a solute is usually exotherm (Mullin, 2001, p.62), but there are examples for endotherm crystallization processes, too, e.g., the crystallization of anhydrous sodium sulfate from an aqueous solution.

5.1.3 Particle number density function

The particle number density function, or particle size distribution, f [$1/\mathrm{m}^3\cdot\mathrm{kg}$] is subject to a population balance equation. Our stochastic approach to the solution of that type of integro-differential equations is described in detail in Chapter 4. In the univariate case which is studied here, the equation for f is

$$\frac{\partial}{\partial t} f + \boldsymbol{u} \cdot \nabla f + G(c,T)\frac{\partial}{\partial m} f = \mathcal{C}(f) \quad \text{in } (0, t_{\text{end}}) \times \Omega_{\boldsymbol{x}} \times \Omega_m. \tag{5.4}$$

This equation must hold on a higher-dimensional domain than the previous ones, because of the internal coordinate m [kg] for the particle description. The domain comprises temporal, physical, and property space; it is called the *particle state space*. Equation (5.4) is a classical formulation for a population balance, when the PSD is changed by transport along both the external and the internal coordinate. It appears, e.g., in (Ramkrishna, 2000, p.20, Eq. (2.7.9)) and is the basis of the model that is used in Hackbusch et al. (2012); Suciu (2013).

The second term on the left-hand side describes advective transport of the population by the velocity $\boldsymbol{u}$. The third term models transport along the internal coordinate. In our case it is a growth term which accounts for particle growth by attachment of dissolved material to crystals in the fluid. The growth rate G depends on concentration and temperature, and on the particle mass m. It is the same rate that appears in (5.3). The dependence on c and T is via a supersaturation model. The higher the supersaturation of the surrounding fluid, the greater one expects G to be. To model the dependence on m, it is sensible to choose a particle geometry description and let G depend on the surface area of a particle of mass m. In Chapter 6 we will choose the so-called Nyvilt model for the solubility (for dependence of G on c and T) and a simple spherical particle geometry (for dependence of G on m). In Chapter 7 a polynomial solubility model will be used, and the same simple geometry.

The term on the right-hand side is a sink-and-source term for particles of size m by further mechanisms. In our case, the only further mechanism is collision growth (coagulation). Coagulation is modeled as in the Smoluchowski coagulation equation: Particles of size m are created by collision of smaller particles at rate

$$\mathcal{C}_+(f,t,\boldsymbol{x},m) = \frac{1}{2}\int_{\Omega_m} K_{\text{coag}}(m-\mu,\mu) f(t,\boldsymbol{x},m-\mu) f(t,\boldsymbol{x},\mu)\ \mathrm{d}\mu.$$

At the same time, particles of size m collide with other particles, forming larger particle aggregates, and therefore disappear from the equation for size m at rate

$$\mathcal{C}_-(f,t,\boldsymbol{x},m) = \int_{\Omega_m} K_{\text{coag}}(m,\mu) f(t,\boldsymbol{x},m) f(t,\boldsymbol{x},\mu)\ \mathrm{d}\mu.$$

The function K_{coag} [m^3/s] is the *coagulation kernel*. It encodes the physics of the coagulation process and must be chosen according to the modeled system. The source expression $\mathcal{C}_+$ and the sink expression $\mathcal{C}_-$ form the net change of the particle population due to coagulation,

$$\mathcal{C} = \mathcal{C}_+ - \mathcal{C}_-.$$

Note that (5.4) with these coagulation terms on the right-hand side can also be regarded as a version of the classical Smoluchowski coagulation equation that is extended by a spatial coordinate $\boldsymbol{x}$, particle transport, and particle growth.

We intend to solve population balance equations with the stochastic particle method that is described in detail in Chapter 4. One finds that for this method the classical formulation (5.4) is not suitable. To be more precise, the growth term on the left-hand side does not fit into the framework and has to be reformulated. We perform this in the next paragraph.

Reformulation of the growth term Remember that for the solution of the PBE (5.4) we want to use the stochastic particle method that is presented in Patterson and Wagner (2012). It is based essentially on two sources: G.A. Bird's direct simulation Monte Carlo algorithm for the Boltzmann equation (Bird (1970)) and Gillespie's stochastic algorithm for the simulation of collision growth phenomena in clouds (Gillespie (1972, 1975)). While the Bird algorithm provides a way to deal with the advective transport part (by splitting), the Gillespie algorithm gives a tool to treat the coagulation part of the equation (by formulating a jump process). Particle growth, which is the third feature of the basic model PBE, is included in neither of these original sources, nor is it considered in Patterson and Wagner (2012). Yet in a previous publication, Patterson et al. (2011), particle surface reaction is included in model and simulation. The way it is done there is closely related to the publications Gillespie (1976, 1977), where a stochastic algorithm similar to that one introduced in Gillespie (1972) is applied to chemical reactions. To conclude this short digression into the literature: our aim is to simulate growth as a stochastic jump process, and therefore we have to find a different formulation of (5.4). The key is to understand particle growth as a particle surface *reaction* rather than as convective transport along the internal coordinate axis. Let us compare those two approaches and find the link between them.

Let for the moment the particle number density function $f : (0, t_{\text{end}}) \times \Omega_m \to \mathbb{R}_0^+$ only depend on time t and the internal (mass) coordinate m, i.e., assume spatial homogeneity. Further, let particle growth by attachment be the only reason for f to change, thus disregarding coagulation.

First, let us sketch the derivation of a population balance equation under the concept of "particle growth as transport along the inner coordinate axis".

This sketch follows the presentation in (Ramkrishna, 2000, pp.16). The main assumption is that for each $m \in \Omega_m$ there exists a (time-independent) growth rate $G(m)$ at which any particle of mass m grows. Fix an arbitrary interval $[a, b] \subseteq \Omega_m$ in the property space. The particle number balance in the interval $[a, b]$ is then given as

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_a^b f(t, m)\ \mathrm{d}m = G(a)f(t, a) - G(b)f(t, b).$$

On the right-hand side, the first term is the particle number flux into $[a, b]$ through a due to growth, and the second term the flux out of $[a, b]$ through b. If one assumes smoothness of f and G, both the time derivative and the right-hand side can be written under the integral,

$$\int_a^b \left(\frac{\partial}{\partial t} f(t, m) + \frac{\partial}{\partial m} \left(G(m) f(t, m) \right) \right)\ \mathrm{d}m = 0.$$

In a next step, the equation can be reduced to the integrand, because of the arbitrary choice of $[a, b]$ and the smoothness of G and f. This gives the final PBE (with the time derivative isolated on the left-hand side again),

$$\frac{\partial}{\partial t} f(t, m) = -\frac{\partial}{\partial m} G(m) f(t, m). \tag{5.5}$$

Assuming that $G(m) \equiv \textit{const.}$, and re-introducing the left-out features (advection, coagulation, dependence on c and T), one regains (5.4). The feature, which we want to emphasize, is that particle growth appears as a *transport* term in this form of the PBE.

In the second approach, which will allow for a more straightforward stochastic formulation, one starts with regarding particle surface growth as a chemical reaction. Each particle type $m \in \Omega_m$ acts as a chemical species and $f(t, m)$ can be interpreted as the number concentration of the species at time t. We stick to the assumption of spatial homogeneity here. One states that each growth reaction for a particle means a mass gain of $\mu > 0$. The growth reactions can be written as

$$m \xrightarrow{r_m} m + \mu,$$

where r_m is the rate at which this growth reaction takes place. A standard ODE description of this reacting system gives for every m

$$\frac{\partial}{\partial t} f(t, m) = r_{m-\mu} f(t, m - \mu) - r_m f(t, m). \tag{5.6}$$

The first term on the right-hand side stands for gain of m-type particles by growth of particles of size $m - \mu$, the second term for loss of m-type particles. Comparing (5.5) and (5.6) yields that in order to connect transport-based and reaction-based growth modeling,

$$-\frac{\partial}{\partial m} G(m) f(t, m) \approx r_{m-\mu} f(t, m - \mu) - r_m f(t, m) \tag{5.7}$$

must be fulfilled in some sense. Abbreviating $\varphi(t,m) := r_m f(t,m)$ for all $m \in \Omega_m$, we observe

$$\varphi(t,m-\mu) - \varphi(t,m) = -\mu \frac{\varphi(t,m-\mu)-\varphi(m)}{-\mu} = -\mu \left(\frac{\partial}{\partial m}\varphi(t,m) + \mathrm{o}(\mu) \right),$$

if φ is smooth. Thus, up to first order,

$$r_{m-\mu} f(t,m-\mu) - r_m f(t,m) = -\mu \frac{\partial}{\partial m} \left(r_m f(t,m) \right)$$

holds, and one can interpret the approximated equation (5.7) in this way.

This leads to the following conclusion. If one wants both approaches to model the same physical process from a macroscopic (first order) point of view, the identity

$$G(m) = \mu r_m \tag{5.8}$$

must hold.

The final formulation of the PBE Replacing the particle growth terms in (5.4) accordingly gives the following formulation of the basic model population balance equation:

$$\frac{\partial}{\partial t} f + \boldsymbol{u} \cdot \nabla f = \mathcal{C}(f) + \mathcal{G}(c,T,f) \quad \text{in } (0,t_{\text{end}}) \times \Omega_{\boldsymbol{x}} \times \Omega_m. \tag{5.9}$$

Here one fixes a mass growth increment μ [kg] and sets, using (5.7) and (5.8),

$$\mathcal{G}(c,T,f,m) = \frac{G(c,T,m-\mu)}{\mu} f(m-\mu) - \frac{G(c,T,m)}{\mu} f(m).$$

The form (5.9) of the PBE is better suited to be treated with the stochastic particle algorithm, as we will show next.

The Markov jump process formulation In order to solve the population balance equation (5.9) with the stochastic simulation algorithm, one must put it in the form of a Markov jump process. This formulation must comprise the same phenomena as the PBE does. In addition, the jump heights and jump rates must reflect the properties of the terms in the PBE. The equation is split into a transport part (advection, left-hand side) and a particle process part (growth and coagulation, right-hand side) (Patterson and Wagner, 2012, p.B292). The spatial domain $\Omega_{\boldsymbol{x}}$ is discretized into N compartments $K_j, j \in \{1,...,N\}$. Each compartment, or *cell*, holds a particle ensemble $\mathcal{E}_j$. The particles are allowed to interact with each other within their current ensemble. In that sense, coagulation and growth are de-localized[2] within the cells. Advection and particle interaction are simulated alternately. While advection is a deterministic step, governed by the fluid velocity $\boldsymbol{u}$, the particle processes are simulated with the SSA. For a more detailed description of the algorithm, see Chapter 4.

[2] In that sense that particles do not have to meet in the same point in space in order to coagulate. It is enough for them to be contained in a common cell.

Let us now formulate the stochastic jump process that takes place in each cell. Our presentation follows closely that one in Patterson et al. (2011). For the sake of simplicity, we give the jump process in terms of the "direct simulation algorithm" described in (Patterson and Wagner, 2012, Ch. 2.1). It has the advantage to spare some stochastic subtleties of the actually used stochastic weighted algorithm (Patterson and Wagner, 2012, Ch. 2.2).

Fix a spatial cell and the respective particle ensemble, $(K, \mathcal{E})$. Each particle e_i in $\mathcal{E}$ is represented by a spatial and an internal coordinate,

$$e_i = (\boldsymbol{x}_i, m_i),$$

with $\boldsymbol{x}_i \in K \subseteq \Omega_{\boldsymbol{x}}$ and $m_i \in \Omega_m$. The entire ensemble, consisting of $N_{\mathcal{E}}$ particles, is thus

$$\mathcal{E} = (e_1, ..., e_{N_{\mathcal{E}}}).$$

The state of the ensemble can change by particle growth jumps and by particle coagulation jumps. Starting at some time t, the system persists in state $\mathcal{E}(t)$ for an exponentially distributed waiting time τ,

$$P(\tau \geq s) = \exp(-\lambda(\mathcal{E})s).$$

The waiting time parameter $\lambda(\mathcal{E})$ is the sum of the individual rates of all jumps that are possible in $\mathcal{E}(t)$. Assembling these in a growth jump rate λ_{grow} and a coagulation jump rate λ_{coag} gives:

$$\lambda(\mathcal{E}) = \lambda_{\text{grow}}(\mathcal{E}) + \lambda_{\text{coag}}(\mathcal{E}).$$

Growth jump rate and coagulation jumps and their rates are treated separately in the following paragraphs.

Particle growth jumps A particle growth jump by a fixed growth height μ changes the state of a certain particle e_i in $\mathcal{E}$ according to

$$e_i = (\boldsymbol{x}_i, m_i) \longrightarrow (\boldsymbol{x}_i, m_i + \mu) =: \tilde{e}_i.$$

Growth jumps in $\mathcal{E}$ happen at the total rate

$$\lambda_{\text{grow}}(\mathcal{E}) = \sum_{i=1}^{N_{\mathcal{E}}} \frac{G(c, T, m_i)}{\mu}.$$

The particle e_i for which the next growth jump occurs is chosen uniformly with probability

$$\frac{G(c, T, m_i)}{\mu} \left(\lambda_{\text{grow}}(\mathcal{E})\right)^{-1}.$$

In this expression, c and T are assumed to be constant (in space) within the cell K.

Particle coagulation jumps Coagulation jumps affect two particles, denote them by e_i and e_j (with $i < j$). A coagulation jump has the form

$$e_i, e_j \longrightarrow (\xi(\boldsymbol{x}_i, \boldsymbol{x}_j), m_i + m_j) =: \tilde{e}_i$$

The particle e_j is removed from the ensemble. The placement of the new particle $\tilde{e}_i$ can be done in several ways. A simple, yet "dangerous" choice is to place the new particle halfway between the two coagulated ones:

$$\xi(\boldsymbol{x}_i, \boldsymbol{x}_j) := \frac{\boldsymbol{x}_i + \boldsymbol{x}_j}{2}.$$

This is dangerous as it can easily lead to numerical instabilities. A more stable approach is stochastic placement, with the center of mass as the expected value.

The total rate of coagulation jumps is the sum of all individual coagulation jump rates of particle pairs. It is calculated as

$$\lambda_{\text{coag}}(\mathcal{E}) = \frac{1}{2N_{\mathcal{E}}} \sum_{i,j=1}^{N_{\mathcal{E}}} K(m_i, m_j).$$

As in the particle growth case, the choice of two particles for a coagulation jump is made uniformly at random with the probabilities

$$P(e_i \text{ and } e_j \text{ chosen for coagulation}) = \frac{K(m_i, m_j)}{2N_{\mathcal{E}}}.$$

With these definitions of jumps and jump rates, the stochastics of the process are sufficiently defined. Evidently, correlations between particles are neglected.

Note that the stochastic reformulation can be interpreted as an extension of the Marcus–Lushnikov process (see Subsection 4.2). Compared to the Marcus–Lushnikov process, the state space is changed from number-based to particle-based and the model is extended by spatial coordinates, advective transport, and particle growth. In this sense, the jump process defined here is to the Marcus–Lushnikov process as the population balance equation (5.4) is to the Smoluchowski coagulation equation.

5.1.4 Overview of the basic model system

Let us now, in compact form, repeat the entire system of equations that form our basic model. The domain consists of the temporal domain $(0, t_{\text{end}})$, physical space $\Omega_{\boldsymbol{x}}$, and property space Ω_m. Together they form the *particle state space*. Fluid velocity $\boldsymbol{u}$, temperature T, molar concentration c, and particle number density f must fulfill the following set of equations.

Navier–Stokes equations:

$$\begin{aligned} \frac{\partial}{\partial t}\boldsymbol{u} - \nu\Delta\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u} + \nabla\frac{P}{\rho} &= \frac{\mathbf{f}}{\rho} \quad \text{in } (0, t_{\text{end}}) \times \Omega_{\boldsymbol{x}} \quad [\mathrm{m/s^2}] \\ \nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } [0, t_{\text{end}}] \times \Omega_{\boldsymbol{x}} \quad [\mathrm{1/s}] \end{aligned}$$

Convection-diffusion-reaction equations:

$$\frac{\partial}{\partial t}T - D_T\Delta T + \boldsymbol{u}\cdot\nabla T = g_T I_{\text{growth}}(c,T,f) \quad \text{in } (0,t_{\text{end}})\times\Omega_{\boldsymbol{x}} \quad [\text{K/s}]$$

$$\frac{\partial}{\partial t}c - D_c\Delta c + \boldsymbol{u}\cdot\nabla c = g_c I_{\text{growth}}(c,T,f) \quad \text{in } (0,t_{\text{end}})\times\Omega_{\boldsymbol{x}} \quad [\text{mol/s}]$$

Population balance equation:

$$\frac{\partial}{\partial t}f + \boldsymbol{u}\cdot\nabla f = \mathcal{C}(f) + \mathcal{G}(c,T,f) \quad \text{in } (0,t_{\text{end}})\times\Omega_{\boldsymbol{x}}\times\Omega_m \quad [1/\text{m}^3\cdot\text{kg}\cdot\text{s}].$$

The inherent coupling of these equations due to the particle growth term and the coupled algorithm is the subject of the following subsection.

5.2 The inherent coupling

In this subsection we want to derive a classification of the inherent coupling of the model system. We are interested in the question how the coupling of the equations transfers to a coupling of the two algorithms, the stochastic particle simulation and the deterministic flow simulation. Which program part is responsible for which part of the model system? Which communication is necessary between the two components of the simulation?

These questions are addressed in the following. We start from the mechanisms that are responsible for the coupling, then lead over to a taxonomy of the coupling phenomena and finally deduce a practical coupling strategy.

Two mechanisms are responsible for the coupling of the equations. Firstly, there is a one-way coupling of the fluid velocity $\boldsymbol{u}$ to the system of the other equations. This is due to the transport of the quantities c, T, and f with the fluid flow. The velocity, which is itself determined from the Navier–Stokes equations, appears as a coefficient in the other equations.

Secondly, the other unknown quantities are coupled to each other via particle surface growth. Concentration and temperature in the surrounding fluid affect the mass growth rate G of the crystals. Crystal growth leads to a local rise in fluid temperature and a local drop in concentration of dissolved species. A schematic overview of the coupling of the equations of the basic model system is diagrammed in Figure 5.1.

A relatively simple extension of the basic system consists in adding more dissolved species to the model and couple them to each other and to the temperature by including species reaction. Such a model has been examined in John and Roland (2010). Figure 5.2 displays the coupling scheme of this extension.

In order to understand the nature of the coupling, we want to state that it can be systemized on three different levels. These are the level of *functions*, the level of *equations*, and the level of *modeled phenomena*. For the Figures 5.1 and 5.2 we used the level of *functions*. This level can be easily linked to the level of *equations*, because there is a one-to-one correspondency between functions and equations. Each unknown function can be identified with one equation

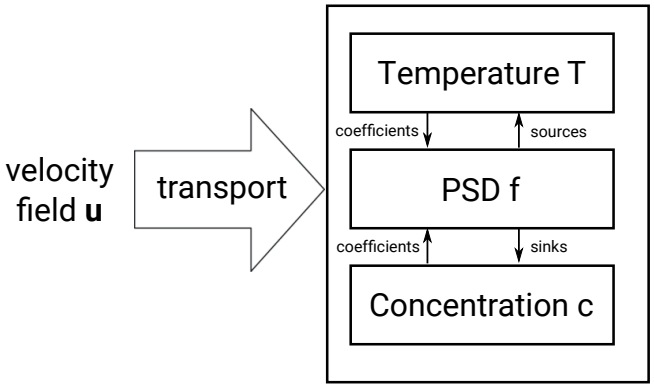

Figure 5.1: The coupling scheme of the basic model system. The fluid velocity field is one-way coupled to the species-particles system.

where it appears as the main unknown, e.g., $\boldsymbol{u}$ (and P) with the Navier–Stokes equations, c and T with their respective transport equation, and f with the PBE. In all other equations, the function will either appear as a coefficient or on the right-hand side. This observation leads to the level of *phenomena*. We found that considering coupling on this level is a suitable intellectual tool for setting up the splitting scheme.

Each physical phenomenon that can be modeled can be understood as depicted in Figure 5.3. A phenomenon is driven by coefficients. They can either be constant or depend on variable quantities within the system. Each phenomenon has then effects on the system. These can be of primary nature, i.e., directly change a variable quantity, or of secondary nature, i.e., change the coefficients of another phenomenon. In our model system, consider for example the phenomenon of particle growth. It is modeled to depend on concentration c, temperature T, and particle number density f. That means, c, T, and f enter that phenomenon as coefficients. Particle growth has a primary effect on all of these quantities, too, because by growth concentration is depleted, temperature raised, and the particle number density shifted towards bigger particles. Secondary effects of particle growth are visible in just one other phenomenon: Depending on the kernel, particle coagulation becomes more (or less) likely, the bigger the particles are.

Note that secondary effects are always mediated by a primary effect, and which phenomenon was responsible in the first place is of no concern to the secondary effect. In the governing equations, primary effects show themselves as right-hand sides, i.e., source and sink terms, while secondary effects manifest themselves as coefficients.

In our model system, two classes of quantities are distinguished. The first class, fluid quantities, comprises $\boldsymbol{u}$, c, and T. The second class, the particle quantities, consists only of the particle number density f.

With these notions at hand, we can state the following basic "formula" for our coupling approach:

Statement 5.2.1. *Three guidelines determine the coupling strategy on the phenomenon level.*

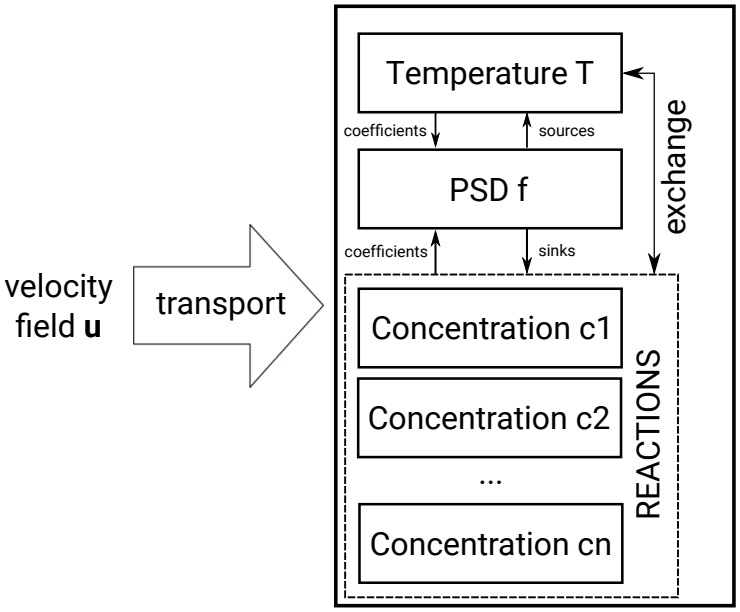

Figure 5.2: The coupling scheme of an extended model system. Species reaction leads to an internal coupling of the advected quantities.

(a) All phenomena that exhibit a primary effect on a particle quantity are subject to the stochastic simulation algorithm.

(b) All phenomena that exhibit primary effects solely on fluid quantities are subject to the flow simulation.

(c) "Communications" between the simulations are either in terms of right-hand sides (primary effects) or coefficients (secondary effects).

In our model system, each phenomenon with particle coefficients has also a primary effect on the particle quantity, and it will thus never be necessary to communicate coefficients from the stochastic simulation to the flow simulation. On the other hand, it is never necessary to communicate right-hand sides from the flow simulation to the stochastic simulation due to (a). That is, all phenomena, which exhibit a primary effect on a particle quantity are treated already *within* the stochastic simulation. Here is our approach on communication between the parts of the simulation in a nutshell:

Statement 5.2.2. *The stochastic particle simulation receives coefficients from the flow simulation, the flow simulation receives right-hand sides in return.*

A less "philosophical" approach on the coupling, more inclined towards computational mathematics by just stating the splitting scheme, is given in the next subsection.

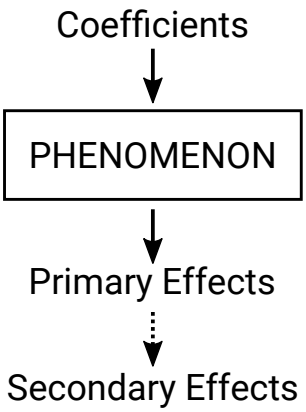

Figure 5.3: Each modeled physical phenomenon is determined by coefficients, and has primary and secondary effects.

5.3 The coupling algorithm

Before giving the algorithm, we have to agree on some more notation. For the computational domain we use the following symbols. The spatially discretized version of the particle state space is $\Omega^{\mathrm{k,\ part}} := (0, t_{\mathrm{end}}) \times \Omega_x^k \times \Omega_m$. The spatially discretized version of the space for the fluid quantities is $\Omega^{\mathrm{h,\ fluid}} := (0, t_{\mathrm{end}}) \times \Omega_x^h$. The subscripts k and h do not have any concrete interpretation yet. They do just indicate that the *spatial* component of the computational domains might be discretized differently for the particles and for the fluid. This necessitates some transfer of functions between Ω_x^k and Ω_x^h. Although the relation of these computational domains is unspecified at the moment, we will use the term "projection" for each operation that transfers functions between them, despite the precise mathematical meaning of that term.

The coupling algorithm, in pseudo code style, is given in Algorithm 4. The algorithm itself is a splitting scheme, and can be written down relatively simple. As so often, governing is in the details here. The constituents are the fluid solver, the stochastic particle simulation, and some means of projecting functions between these two entities. The latter can be pictured as a communication layer. In the pseudocode, "advance the SPS" means to run the stochastic simulation from its state $\mathcal{E}^k(t^{n-1}) = (\mathcal{E}_1, ..., \mathcal{E}_N)$ at time t^{n-1} to its state $\mathcal{E}^k(t^n)$ at time t^n. This means, perform as many transport splitting steps and all jumps of the particle ensembles that ought to occur in the time span $[t^{n-1}, t^n]$. In the formulation of Algorithm 4 we assumed a stationary velocity field $\boldsymbol{u}$. If $\boldsymbol{u}$ is time-dependent (as in the system of Chapter 7), one has to include its update into the time-loop. Equipped with this description of the model, the coupling philosophy and the pseudo-code version of the coupling algorithm, we can proceed towards practical applications of the resulting coupled method.

Algorithm 4 Coupling of computational fluid dynamic simulation (CFDS) and stochastic particle simulation (SPS)

Input: $\Omega^{\mathrm{k,\ part}}$ and $\Omega^{\mathrm{h,\ fluid}}$, initial and boundary data, Δt.

PRECOMPUTING:
Compute $\boldsymbol{u}$ by solving the Navier–Stokes equations in the CFDS.
Project $\boldsymbol{u}$ into Ω_x^k.
Initialize the SPS, c^0, and T^0 with initial data.

TIME LOOP:
$n := 0$
$t^0 := t_{\mathrm{start}} + \Delta t$
while $t^n < t_{\mathrm{end}}$ **do**
  $n := n + 1$.
  $t^n := t^{n-1} + \Delta t$
  PART A
    Project c^{n-1} and T^{n-1} to Ω_x^k
    Update the growth coefficients in the SPS with c^{n-1} and T^{n-1}
    Advance the SPS to time t^n
    Compute I^n_{growth} from the SPS ensembles.
  PART B
    Project I^n_{growth} to Ω_x^h.
    Update the source and sink terms in the CFDS with I^n_{growth}.
    Compute c^n and T^n with the CFDS.
end while

POST-PROCESSING
Perform post-processing on the gathered data

6 A 2d Axisymmetric Simulation of a Tubular Flow Crystallizer

This chapter is devoted to the modeling and simulation of an experimental flow crystallizer. We will use the coupling technique that was expounded in the previous chapter for the solution of this model system.

A flow crystallizer is a device for the production of crystalline substances. It consists mainly of a long, thin tube that is usually coiled up for practical reasons. A three-component dispersion is pumped into the tube at the one end, which will be called the inlet. The dispersion contains a solvent, a solute and seed crystals, which consist of the same material as the solute. The dispersion is warm and highly saturated. While the dispersion flows through the tube, mechanisms that result in a change of the crystal size distribution are excited. Such mechanisms include nucleation, surface attachment growth, particle collision growth, or particle breakage. The occurrence and intensity of these processes are controlled by several parameters. Among these are the fluid composition, the fluid velocity, the shape and makeup of the tube, and the surrounding temperature. At the tube outlet product crystals with properties different from the input crystals can be collected.

Flow crystallizers are a promising technology, e.g., in pharmaceutical production. They allow for very regular particle growth that can be accurately controlled. Regular shape and size are desirable properties of crystals that are used in medicants, as more accurate crystals mean better control over their medicinal effect. From a technical point of view flow crystallizers are interesting for two more reasons. Firstly, they can be operated continuously and do not, at least in theory, operate in intervals as the alternative batch crystallizers do. Secondly, knowledge transfer from the laboratory to the industry is relatively easy. Wherever fluids are involved, there are big differences between different scales, and so a typical development issue would be that a system behaves totally different at small scale (laboratory) and big scale (industry). This problem of scale-up does not apply to flow crystallizers: scaling up to the industrial measure does not mean bigger or longer tubes, it means *more* tubes, operated simultaneously (Eder et al., 2010, p.2247). Thus, the conditions in the individual tube stay the same in experiment and application. The problem of scale-up underlines the need for effective computer simulations in the field of industrial crystal growth, although for flow crystallizers it is admittedly less severe for the mentioned reason.

The main alternative to flow crystallizers are batch crystallizers. A batch crystallizer consists mainly of a large vessel in which a crystal suspension is stirred. Due to the shape of the vessel and the evolving flow field, crystals are held hovering in the center of the container and grow there. The most im-

portant growth mechanism in a batch crystallizer is *collision growth*, while in a flow crystallizer *surface growth* is largely responsible for the size gain of the crystals. One reason for this difference is the nature of the flow field. In a batch crystallizer a turbulent flow field develops, see the investigations in Chapter 7. The crystals follow the turbulent vortices, which results in a lot of effective particle collisions. Particles stick to each other and form larger aggregates. A flow crystallizer on the other hand is operated at laminar conditions, the flow field trajectories are aligned and particle collisions happen less frequently, surface growth takes the lead. This effect is desired, since it leads to relatively uniform particle growth, and it is boosted by cooling the flow crystallizer from the outside. As temperature drops, so does the solubility of the crystal material. Supersaturation of the surrounding solution increases, and the need to reduce supersaturation is the driving force behind surface attachment growth. One must confine the above statement to tube crystallizers operated at low velocity and smooth curvature. As soon as either velocity or curvature hit a critical number, a secondary flow structure (Dean vortices) begins to develop and particle coagulation becomes much more likely.

Collision growth can be an obstacle to the operation of a tube crystallizer, as can excessive primary nucleation (Eder et al., 2010, p.2249). Both can easily lead to blockage of the typically rather thin tube and thus cross the aim of a continuous operation of the crystallizer. A second issue is the goal to achieve a sharp size distribution of the product crystals. Collision growth runs contrary to this aim, because each collision event will result in a sudden jump of the size of the involved crystals, thus broadening the particle size distribution.

The flow crystallizer that is in the focus of this chapter was set up and operated by the group of Prof. Khinast at TU Graz. The crystalline model substance was acetylsalicylic acid (ASA), commonly known by its brand name aspirin. The solvent was pure ethanol (EtOH). Results were first reported in Eder et al. (2010). Modified setups were presented later in Eder et al. (2011) and Eder et al. (2012). The first of these works contains, alongside experimental data, a 1d ODE model of the experiment, and computational results that were gained with it. The work Besenhard et al. (2014) by the same group contains a model and simulation for the setup in Eder et al. (2012). Recently, the group applied the method to crystallization growth of the enzyme lysozyme (Neugebauer and Khinast (2015)).

The work (Eder et al. (2010)) is well cited, since the continuous operation of a flow crystallizer needs careful fine-tuning of parameters, in which the authors succeeded. The authors found four different parameter setups for which they could operate the crystallizer up to fifteen minutes without blockage, maintaining almost steady-state conditions at the outlet. In addition, ASA is not a commonplace engineering model substance, but an indispensable medicant, and therefore the experiments point in a distinct practical direction.

The simulations that will be presented in this chapter were performed in the spirit of a proof-of-concept example, proving the applicability of the stochastic-deterministic approach. The example is well feasible with the classical PBS and PBE methods, such as direct discretization : simulation (Suciu (2013)), method of moments (Marchisio et al. (2003)), or an operator splitting approach (Ahmed

et al. (2011)). Therefore, the example would be fit for a direct comparison of their effectivity and efficiency. This undertaking is not in the scope of this work and might be the subject of further studies. Instead, this chapter should prove the viability of our new coupled method by reproducing experimental results with reasonable computational effort, and enable us to identify further questions and directions of enhancement of the method.

The simulations are done in 2d, and we use a simple, one-dimensional particle model, by assuming spherical crystals. The reduction of the 3d geometry to a 2d computational grid has two steps. The first step is stretching out the coiled up tube, the second step is the assumption of axisymmetry. Thanks to these simplifications, we profit from a very easy geometry throughout this chapter.

The chapter is organized as follows. For a start, in Section 6.1 we state the mathematical model and connect it to the general model of Chapter 5.1. We also describe the experiment and insert a subsection on general modeling considerations. We postpone the detailed derivation of modeling parameters from the data reported in Eder et al. (2010) to an appendix (6.4) at the end of the chapter. In Section 6.2 we address computational issues, show computational results and comment on insights gained from the computations. In the final discussion in Chapter 6.3 we give an outlook on a 3d version of the algorithm.

6.1 Modeling a tube crystallizer

Let us present the mathematical model of the ASA tube crystallizer. We aim at a certain brevity of presentation, details on different aspects are given in Section 6.4. First of all the experiment to be modeled and simulated will be described. Then we list several general modeling considerations, which include considerations on the fluid density, the axisymmetric geometry, and the particle model. Note that in the following, we are going to use the terms *particles* and *crystals* interchangeably. The first term is closer to the mathematical model, the second term more related to the actual physical system. A nice glossary on particle terms can be found in (Randolph and Larson, 1988, p.17 f). Finally we go through all the equations, giving their parameters, boundary and initial conditions.

6.1.1 The experiment

In the experimental setup, as described in Eder et al. (2010), the crystallization takes place in a 15 m long polysiloxane tube that is coiled up in a box of dimensions 0.41 m $\times$ 0.24 m $\times$ 0.26 m. The inner diameter of the tube is 2 mm and the outer diameter is 4 mm, i.e., the tube has a wall thickness of 1 mm. The fluid that flows through the crystallizer is a mixture that is fed from two vessels. The first vessel contains a warm solution of ASA in ethanol, close to supersaturation. The second contains a well-mixed ASA seed crystal suspension, consisting of ethanol, dissolved ASA, and undissolved ASA in crystalline form. One peristaltic pump per vessel pumps the contents into a Y-fitting, from there the mixture flows into the tube. The temperature in the box, which contains the tube, is held at 24.3$\pm$1 °C. That is cool compared to the

temperature of the fluid at the inlet (see Table 6.1). The cooling results in a drop of supersaturation and thus in crystal surface growth. At inflow and outflow of the crystallizer a microscope and a specialized camera were installed, which allowed to gather data on the in- and outflowing crystals. With a connected computer an approximation of the crystal size distribution at in- and outflow could be determined from this data. In a supposedly strenuous process, the experimentators figured out four configurations to steadily operate the flow crystallizer, meeting several requirements at once. These requirements can be formulated as follows.

- Comparing inflowing and outflowing crystals, the peak of the crystal size distribution should move significantly towards larger particles, i.e., significant surface growth should take place within the device.
- The peak of the product crystal size distribution should be sharp, i.e., the particles at the outflow should be rather evenly sized.
- The tube should not be blocked due to excessive crystal growth.
- Finally, as much crystalline material as possible, given the other requirements, should be produced within a certain operating time.

The four parameter configurations, or "setups", which the experimentators identified, will accompany us throughout this chapter. It is our declared goal to computationally reproduce the experimental results for all four parameter sets. The target variables of the simulation are the mean $\bar{d}$ and standard deviation σ of the particle diameter at the outflow. We aim at reproducing them for all four setups. There are several constraints which we will also use for the validation and verification of our computational results. These are:

- mean and standard deviation of the crystal diameter at the inflow,
- mass flow rate of crystalline ASA at the outflow, and
- the conservation of the mass flux throughout the tube.

Those target quantities of the experiments are listed in Table 6.1.

6.1.2 General modeling considerations

There are some general restrictions guiding the entire modeling process. They are required to keep the model simple and the computing time feasible.

General assumptions on the particles The particles in the model represent physical entities, they represent ASA crystals of different sizes. Generally speaking, we employ what would be called a "quasihomogeneous" approach. This means that we regard the entire computational domain to be continuously filled with homogeneous matter. Its suspension character and its microscopic features are ignored at first. The particles then get re-introduced as zero-dimensional objects, which interact with the fluid only via their particle size distribution, i.e., a macroscopic observable.

Table 6.1: *Top*: Target data at the inlet. Crystal mass flow rate Q_{in}, crystal mass flux $\dot{m}_{\text{in}}$, particle number mean diameter $\bar{d}_{\text{in}}$, particle diameter standard deviation $\sigma(d)_{\text{in}}$, suspension temperature T_{in}. *Bottom*: Target data at the outlet. First number refers to the end time of the experiment (15, 11, 9, 9 minutes), number in brackets to an intermediate measurement time (9, 7, 6, 5 minutes).

Quantity	Q_{in}	$\dot{m}_{\text{in}}$	$\bar{d}_{\text{in}}$	$\sigma(d)_{\text{in}}$	T_{in}
Unit	$^{\text{m}^3}/_{\text{s}}$	$^{\text{kg}}/_{\text{s}}$	μm	μm	K
Setup 1	1.9e-7	0.0156	90	38	307.6
Setup 2	2.9e-7	0.0258	81	26	312.9
Setup 3	3.8e-7	0.0324	91	27	313.1
Setup 4	4.2e-7	0.0378	85	29	313.7

Quantity	$\dot{m}_{\text{out}}$	$\bar{d}_{\text{out}}$	$\sigma(d)_{\text{out}}$	T_{out}
Unit	$^{\text{kg}}/_{\text{s}}$	μm	μm	K
Setup 1	0.0888	243 (233)	65 (62)	297.5
Setup 2	0.1512	214 (215)	50 (43)	297.5
Setup 3	0.1932	192 (183)	49 (41)	297.5
Setup 4	0.195	166 (186)	44 (45)	297.5

This quasihomogeneous approach has consequences for the particle model: The particles have a position but no extension. They are described by the particle size distribution, which is a cumulated quantity. Finally, the particles are assumed to follow the streamlines of the macroscopic velocity field and do not backcouple on the velocity field. As for the internal coordinates, we make use of a one-dimensional model, particle mass m [kg] is the only inner coordinate.

Constant density In order to stay in the framework of incompressible fluids with non-varying density, the density ρ_{susp} of the fluid is kept constant. As the authors of Eder et al. (2010) do not provide a value for the density of the suspension, we deduce one from the information on the inflowing fluids. This process includes assumptions on the way ASA and EtOH mix. Especially, the assumption of constant density means that phase transition of ASA from dissolved to solid form, does not influence the density. Both phases are assumed to contribute to the overall density in the same way. The constant density of the suspension is

$$\rho_{\text{susp}} = 916.87 \ ^{\text{kg}}/_{\text{m}^3}.$$

For the derivation of this value from the experimental data, see Section 6.4.2.

Modeling domain Let us agree first that the computations will comprise only the 15 m long main portion of the tube, excluding the vessels, the Y-fitting and the mixing zone of the device. Also, the computational domain spans only

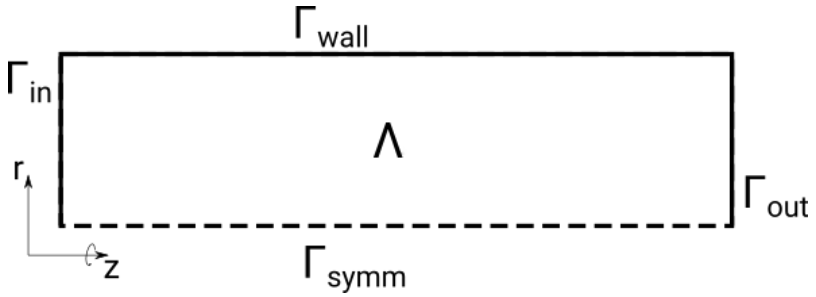

Figure 6.1: The modeling domain (symbolic), with denotations of the boundary pieces and the coordinates. The dashed line is the "spurious boundary", i.e., the symmetry axis.

the inner part of the tube, that part where the fluid flows. Although it might be of interest for the energy balance to simulate heat transport through the polysiloxane material, the tube wall is not part of the computational domain. This requires some assumptions on the boundary conditions for the energy balance equation, as we shall see in Section 6.1.6.

On top of that the geometry of the tube interior is simplified. The originally coiled up tube is straightened out, and an axisymmetric 2d approach is pursued. The meridian modeling domain is depicted in Figure 6.1. To fix notations: Λ is the 2d simulation domain, while Ω will denote the three-dimensional cylinder which is gained by rotation of Λ. The boundaries of Λ are in- and outflow boundary Γ_{in} and Γ_{out}. The wall boundary is Γ_{wall}, and Γ_{symm} is the spurious (i.e., non-physical) symmetry boundary in the center of the tube. The boundaries of Ω go unnamed, since we give all boundary conditions for the meridian domain only. As for the cylindrical variables, $z \in [0, 15]$ is the axial variable and $r \in [0, 0.001]$ is the radial variable, their values are given in m. Both directions are also depicted in Figure 6.1.

Some notation As it is our goal to simulate four different setups of the crystallizer, we use the superscript $[i]$ with $i \in \{1, ..., 4\}$ for the distinction of the data. Often, when some general statement should hold for all four setups, we skip the superscript.

From time to time it will be necessary to distinguish between quantities connected to different parts of the simulation domain or its surroundings. This we will denote with subscripts. It should be clear at all times, what these superscripts refer to. Let us just remark that subscripts *seed* and *sol* refer to the two inflow streams ("seed stream" - containing crystals, "solution stream" - ASA-EtOH solution without crystals), which then combine to the "feed stream", subscripted *feed*.

6.1.3 Velocity field

The velocity field is precomputed by solving the Navier–Stokes equations, which reduce to the Stokes equations in the tube setting. Backcoupling from the particles, concentration or temperature is not contained in the model. The

same approach was pursued in Hackbusch et al. (2012) and Anker et al. (2015), there it is justified by the small gradients of temperature and concentration and the general diluteness of the solution. These descriptions do not quite apply to the crystallizer regarded here, but to keep things simple, we stick to the approved principle throughout this chapter. A very pragmatic vindication of our no-backcoupling approach is that even the one-dimensional plug flow model presented in Eder et al. (2010) delivered acceptable results. We have thus reasons to expect the simulation not to fail due to the lack of backcoupling.

In Subsection 6.4.3 the Reynolds numbers and Dean numbers of the flows are calculated. All turn out to be small enough to expect laminar flows without secondary vortices (see Table 6.6 for the numbers).

A laminar flow in a tube or pipe develops a parabolic velocity profile. For this well-understood case, the analytic solution to the Navier–Stokes equations is given by the Law of Hagen–Poiseuille. In order to verify the correctness of our axisymmetric finite elements implementation, we decided to precalculate the solution numerically anyway, by solving the (axisymmetric) stationary Navier–Stokes equations (c.f. Equation (6.12)). The de-dimensionalized equations read in their strong form on Ω:

$$\begin{aligned} -\nu\Delta\boldsymbol{u} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \nabla p &= \mathbf{0} \\ -\nabla\cdot\boldsymbol{u} &= 0. \end{aligned}$$

The dimensionless viscosity ν is just $\frac{1}{Re}$, and the Reynolds number depends on the parameter set, see Table 6.6. The Navier–Stokes equations must be put into weak formulation and transformed to their axisymmetric formulation (Section 6.4.1). A solution $\boldsymbol{u} = (u_r, u_z)$ of the axisymmetric version of the Navier–Stokes equations should fulfill the following boundary conditions:

$$\begin{aligned} u_r = 0 \quad \text{and} \quad u_z = u_{\max}^{[i]}\left(0.001^2 - r^2\right) \quad &\text{on } \Gamma_{\text{in}} \\ \boldsymbol{u} = \mathbf{0} \quad &\text{on } \Gamma_{\text{wall}} \\ (\nu\nabla\boldsymbol{u} - p\mathbb{I})\mathbf{n} = \mathbf{0} \quad &\text{on } \Gamma_{\text{out}}. \end{aligned}$$

The axisymmetry boundary conditions on Γ_{symm} close the equation (see Equation (6.14)):

$$u_r = 0 \quad \text{and} \quad \frac{\partial u_z}{\partial r} = 0 \quad \text{on } \Gamma_{\text{symm}}.$$

The data $u_{\max}^{[i]}$ depends on the parameter setup. We use the values

$$\begin{aligned} u_{\max}^{[1]} &= 0.1209 \text{ m/s} \\ u_{\max}^{[2]} &= 0.1824 \text{ m/s} \\ u_{\max}^{[3]} &= 0.2419 \text{ m/s} \\ u_{\max}^{[4]} &= 0.2673 \text{ m/s}. \end{aligned}$$

They are chosen in such a way as to ensure the mass flow rates Q_{in} listed in Table 6.6. These maximum velocities are double the mean velocities, a well-known trait of parabolic flows.

The solution of the equation is only expected to transport the fully-developed profile at Γ_{in} through the whole tube. This means in particular that $u_r \equiv 0$ in Γ, which simplifies several of the following considerations.

6.1.4 Particle size distribution equation

The population balance equation to solve is the one that was derived in Section 5.1.3. We repeat it here, now taking into account the one-dimensional particle model, with inner coordinate m [kg]. The equation reads:

$$\frac{\partial}{\partial t} f + \boldsymbol{u} \cdot \nabla f = \mathcal{C}(f) + \mathcal{G}(c, T, f) \quad \text{in } (0, t_{\text{end}}) \times \Omega \times [0, \infty).$$

The sought function f [$^1/_{\text{m}^3\text{kg}}$] is the particle number density. The end time t_{end} depends on the experimental setup and will be given later.

The corresponding Markov jump process formulation is as described in Section 5.1.3. We give the boundary conditions in terms of the stochastic formulation, see Section 6.4.5 for details on the derivation and implementation.

At the inflow boundary Γ_{in} particles are inserted into the simulation domain, i.e., into the particle ensembles of those cells, which border the inception boundary. Particle inceptions are simulated as inception jumps, meaning that each ensemble $\mathcal{E}$ in contact to Γ_{in} gets equipped with an additional jump rate

$$\lambda_{\text{in}}(\mathcal{E})$$

and a corresponding jump, which adds a new particle to the ensemble. The inception jump rates are chosen such that in the very first layer of cells a certain ASA crystal mass concentration is achieved on expectation. The jump rates, superscripted with the respective parameter setup number, are

$$\begin{aligned}
\lambda_{\text{incept}}^{[1]} &= 395 \cdot 10^6 u_z \\
\lambda_{\text{incept}}^{[2]} &= 661 \cdot 10^6 u_z \\
\lambda_{\text{incept}}^{[3]} &= 486 \cdot 10^6 u_z \\
\lambda_{\text{incept}}^{[4]} &= 552 \cdot 10^6 u_z.
\end{aligned}$$

Their unit is $^{\#\text{particles}}/_{\text{m}^2\text{s}}$. Note that the velocity component u_z, which is orthogonal to the inception boundary, must be included in the formulation of the inception jump rate here. This is because the target particle concentration in a cell is proportional to the velocity in that cell. The differences of the inception jump rates are due to the different input particle size distributions, see Table 6.1. The position within a cell and the amount of ASA of which a newly incepted particle consists are determined stochastically. For details on this process see Section 6.4.5.

It is not necessary to formulate wall and symmetry axis boundary conditions at Γ_{wall} and Γ_{symm}, since $u_r = 0$ means that no wall or axis collisions happen.

At the outflow boundary particles are removed from the computation when the free-streaming step transports them to a point beyond Γ_{out}. This corresponds to standard outflow conditions, as we use them for the concentration and energy balance equations.

In the following two paragraphs we give the of coagulation rate and the surface growth rate.

Coagulation rate Recall the general form of the coagulation term

$$\mathcal{C}(f,t,\boldsymbol{x},m) = \frac{1}{2}\int_{\Omega_m} K(m-\mu,\mu)f(t,\boldsymbol{x},m-\mu)f(t,\boldsymbol{x},\mu)\ \mathrm{d}\mu \\ - \int_{\Omega_m} K(m,\mu)f(t,\boldsymbol{x},m)f(t,\boldsymbol{x},\mu)\ \mathrm{d}\mu.$$

There are plenty of options for defining the coagulation kernel K. Let us state two "traditional" kernels, which will be used in computations later. The first is the additive shear flow kernel, which is usually suitable for the laminar flow regime (Barthelmes et al. (2003)). In terms of particle mass, the coagulation kernel gives the likeliness that two 'near' particles of masses m_1 and m_2 coagulate in an infinitesimal time interval. It has the form

$$K(m_1,m_2) = \kappa^{\mathrm{add}}(m_1+m_2), \tag{6.1}$$

and the unit is m^3/s. The scaling parameter κ^{add} $[\mathrm{m}^3/\mathrm{s\cdot kg}]$ must be modeled or determined experimentally.

A second, simpler option is the constant coagulation kernel

$$K(m_1,m_2) = \kappa^{\mathrm{const}}, \tag{6.2}$$

where the scaling parameter is also to be specified.

Growth rate For the growth term we stick to the semi-empirical model used in Lindenberg et al. (2009) and Besenhard et al. (2014). It is leaned on the Arrhenius equation for the reaction speed constant of temperature dependent equations, but multiplied with a monomial supersaturation term. In Lindenberg et al. (2009) the model is used for a diameter-based 1d particle description. There the growth rate is:

$$G_d(c,T) = k_{G_1}\exp\left(-\frac{k_{G_2}}{RT}\right)(c_{\mathrm{sat}}(T)-c)^{k_{G_3}}.$$

The unit of G_d is m/s. The model is formulated in terms of absolute supersaturation. The parameters k_G were experimentally determined (Lindenberg et al. (2009)) to be

$$\begin{aligned} k_{G_1} &= 3.21\cdot 10^{-4}\ \mathrm{m}/\mathrm{s} \\ k_{G_2} &= 2.58\cdot 10^{-4}\ \mathrm{J}/\mathrm{mol} \\ k_{G_3} &= 1. \end{aligned}$$

For our mass-based approach we have to reformulate G_d to G_m $[\mathrm{kg}/\mathrm{s}]$, the mass growth speed. Because the simple model assumes spherical particles, there is the following dependency of particle mass on particle diameter:

$$m = \frac{\pi}{6}d^3\rho_{\mathrm{ASA}}.$$

Using the chain rule and $G_d = \frac{\mathrm{d}}{\mathrm{d}t}d$, one calculates

$$G_m(c,T,m) = \frac{\mathrm{d}}{\mathrm{d}t}m = \frac{\mathrm{d}}{\mathrm{d}d}m\frac{\mathrm{d}}{\mathrm{d}t}d = \frac{\pi}{2}d^2\rho_{\mathrm{ASA}}G_d(c,T).$$

Setting $A_O(m) = \pi d(m)^2$, the surface area of a spherical ASA particle of mass m, one obtains

$$G_m(c,T,m) = \frac{1}{2}A_O(m)\rho_{\mathrm{ASA}}G_d(c,T). \tag{6.3}$$

This particle mass growth rate depends linearly on the particle surface area. Another modeling decision concerns the supersaturation $c_{\mathrm{sat}}(T)$. We use here a fitted Nyvilt model. Depending on T $[K]$:

$$c_{\mathrm{sat}}(T) = 10^{27.769+\frac{-2500.906}{T}-8.323\log_{10}(T)}$$

as was suggested and given in Eder et al. (2010), see also Section 6.4.2.

6.1.5 Concentration balance equation

The concentration balance equation is a convection-diffusion equation. Its strong formulation in Cartesian coordinates is

$$\frac{\partial c}{\partial t} - D\Delta c + \boldsymbol{u}\cdot\nabla c = -\frac{1}{M_{\mathrm{ASA}}}I_{\mathrm{growth}}(c,T,f) \quad \text{on } \Omega\times(0,t_{\mathrm{end}}). \tag{6.4}$$

The unknown function c $[\mathrm{mol}/\mathrm{m}^3]$ describes the molar concentration of dissolved ASA. The diffusion coefficient D ought to be the diffusion coefficient of dissolved ASA in EtOH. As we are not aware of an exact value in the literature, we use the diffusion coefficient of another model substance, urea, in ethanol (Anker et al. (2015)). Since any numerical stabilization for convection-diffusion equations introduces spurious diffusion, the exact value of D is not as important as its order of magnitude. We set

$$D = 1.35\cdot 10^{-9} \quad [\mathrm{m}^2/\mathrm{s}].$$

The precomputed velocity is $\boldsymbol{u}$ $[\mathrm{m}/\mathrm{s}]$, and M_{ASA} $[\mathrm{kg}/\mathrm{mol}]$ is the molar mass of ASA as given in Table 6.2.

On the right-hand side, the term I_{growth} is the surface growth intensity of the ASA crystals. As was stated in Section 5.1.2, I_{growth} measures the occurrence of particle surface growth and has units $\mathrm{kg}/\mathrm{m}^3\mathrm{s}$. With the definition of I_{growth} given in (5.3), and the discussion of the growth term in the former section, the growth intensity term takes the form

$$I_{\mathrm{growth}}(c,T,f,t,\boldsymbol{x}) = \frac{\rho_{\mathrm{ASA}}}{2}\int_{[0,\infty)} A_O(m)k_{G_1}\exp\left(-\frac{k_{G_2}}{RT}\right)(c_{\mathrm{sat}}(T)-c)f(t,\boldsymbol{x},m)\,\mathrm{d}m. \tag{6.5}$$

The boundary conditions for the axisymmetric re-formulation of (6.5) are

$$\begin{cases} c = 1511.3 \text{ on } \Gamma_{\mathrm{in}} \\ \dfrac{\partial c}{\partial \mathbf{n}_\Gamma} = 0 \qquad \text{on } \Gamma_{\mathrm{wall}}\cup\Gamma_{\mathrm{out}}\cup\Gamma_{\mathrm{symm}}. \end{cases}$$

Table 6.2: Material constants of ASA and ethanol, used throughout this chapter.

Quantity	M_{ASA}	M_{EtOH}	ρ_{ASA}	ρ_{EtOH}
Unit	kg/mol	kg/mol	kg/m^3	kg/m^3
Value	0.18016	0.04607	1350	790

Interestingly enough, the boundary conditions at outflow, wall and symmetry boundary are all the same, although its interpretation differs. At Γ_{wall} it describes impermeability, at the outflow Γ_{out} it is a natural outflow condition, and at the symmetry axis Γ_{symm} it is the necessary symmetry condition that will be derived in Section 6.4.1. The origin of the Dirichlet value at the inflow Γ_{in} from the experimental data is discussed in Section 6.4.2.

Finally the equation is closed with an initial condition. We assume that there is no ASA present in the crystallizer before the experiment starts, thus one has

$$c(0,\cdot) = 0 \quad \text{on } \Omega.$$

6.1.6 Energy balance equation

The energy balance comes, as did the concentration balance, in shape of a convection-diffusion equation. In its 3d strong formulation it reads

$$\frac{\partial T}{\partial t} + \boldsymbol{u}\cdot\nabla T - \frac{\lambda_{\text{EtOH}}}{\rho_{\text{susp}} C_{\text{EtOH}}}\Delta T = \frac{\Delta h_{\text{cryst}}}{\rho_{\text{susp}} C_{\text{EtOH}}} I_{\text{growth}}(c,T,f) \text{ on } \Omega\times(0,t_{\text{end}}). \tag{6.6}$$

The sought quantity T is the temperature in Kelvin, $\boldsymbol{u}$ once more the precomputed velocity field. On the right-hand side appears again I_{growth}, the growth intensity as was discussed above. The constants which scale the influence of the source term and the relation of diffusive and advective transport are

$$\begin{aligned}
\lambda_{\text{EtOH}} &= 0.1676 && [\text{W}/\text{m}\cdot\text{K}] && \text{(thermal conductivity of ethanol)}\\
\rho_{\text{susp}} &= 916.87 && [\text{kg}/\text{m}^3] && \text{(assumed density of the suspension)}\\
C_{\text{EtOH}} &= 2441.3 && [\text{J}/\text{kg}\cdot\text{K}] && \text{(specific heat capacity of ethanol)}\\
\Delta h_{\text{cryst}} &= 1.6541\cdot 10^5 && [\text{J}/\text{kg}] && \text{(specific heat of crystallization).}
\end{aligned}$$

Note that where no other assumptions from the authors of the experimental paper were available (thermal conductivity and specific heat capacity of the suspension), we used the values of the solvent instead. The constant Δh_{cryst} is the specific heat of fusion of ASA. In Eder et al. (2010) we found the molar heat of fusion of ASA to be 29800 J/mol, which is assumed to be the same as the molar heat of crystallization. Dividing by M_{ASA} we get the specific heat of ASA crystallization as given above. It is the heat that will be released when one kilogram of dissolved ASA changes phase from dissolved to crystalline state. Both sides of Equation (6.6) have units T/s.

As have the preceding equations, (6.6) has to be put into its 2d axisymmetric formulation. We give the boundary conditions for that formulation. Those boundary conditions are

$$\begin{cases} T = T_{\text{feed}}^{[i]} & \text{on } \Gamma_{\text{in}} \\ \dfrac{\partial T}{\partial \mathbf{n}_\Gamma} = 0 & \text{on } \Gamma_{\text{out}} \cup \Gamma_{\text{symm}} \\ T = T_{\text{wall}}^{[i]} & \text{on } \Gamma_{\text{wall}}. \end{cases}$$

The Dirichlet value at the inflow, T_{feed}, depends on the parameter set i. The respective values come from measurements reported in Eder et al. (2010), and are given in Table 6.1. The Neumann conditions at outflow and symmetry axis are the same conditions that were used for the concentration balance equation. The wall boundary condition is special though, since here we prescribe the heat loss through the tube wall by imposing a temperature profile. This is a concession to excluding the tube wall from the modeling domain. For details see Section 6.4.4.

The equation is closed with an initial condition. We assume that the entire tube finds itself at ambient temperature before the hot suspension is pumped into it. Therefore

$$T(0, \cdot) = 297.5 \text{ K on } \Omega$$

is the initial condition.

6.2 Simulating the ASA tube crystallizer

In this chapter we want to put the modeling the was done so far to some numerical use. With the aim of reproducing the experimental results in all four setups, the general method and the specific model are brought together in a series of computer simulations. Firstly, in Section 6.2.1, we give the details on the computation. In Section 6.2.2 we present our results, which include a parameter study of the coagulation intensity parameter κ for the constant coagulation kernel, and the reproduction of the experimental data.

6.2.1 Details on the computation

For the numerical simulation of the example we used two in-house code bases, the finite element CFD package ParMooN (Ganesan et al. (2016); Wilbrandt et al. (2017)) and the stochastic particle simulation code Brush (Patterson et al. (2011)). A custom C++ interface layer between those two, which managed conversion and communication, was implemented. The 2d axisymmetric computational domain was discretized regularly into 5×150 rectangles, see Figure 6.2. This simple grid is used for all parts of the simulation, i.e., it is at the same time the finite element mesh for $\boldsymbol{u}$, c, and T, and the grid of ensemble cells for the SPS.

The Navier–Stokes equations were discretized with inf-sup stable Q_2/Q_1-elements, and for the convection-diffusion equations Q_1-elements were used.

This meant a problem size of 7528 d.o.f. for the Stokes equation and of 906 d.o.f for both convection-diffusion equations. Those systems are so small and so sparse that we could use a sequential direct solver (UMFPACK) for the arising linear systems. Some performance profiling proved that the time spent in those solvers was too small to justify further optimization effort here. In the SPS, computing time is dominantly determined by the number of computational particles per ensemble. We restricted the number of computational particles per cell to 128, which showed to be a good compromise of computing time and accuracy. We also fixed the maximal number concentration of ASA particles to $1.2 \cdot 10^{11}$ $1/\mathrm{m}^3$. This meant that each computational particle, upon insertion to the simulation domain, stood representative for 937.5 physical particles per cubic centimeter. These numbers could be kept constant in space in time, and could also be chosen the same for all four parameter setups.

Information transfer between SPS and CFD was done with L^2-projection operators. In fact, the Q_2-velocity was projected to a Q_0 function for both applications. The reason is that otherwise (projection to Q_1 for the CDREs, Q_0 for the SPS), an unwanted numeric effect takes place. Since c and T get transported with that velocity which is found at the respective quadrature points, they will be transported with a different velocity than f, which is transported only by the Q_0-velocity in the ensemble cell. It arises a lag between fluid quantities and particle transport, the conservation of mass flux in the tube will be lost.

The time step we chose was $\Delta t = 0.025$. With a coarser time step the CFL-like condition of the linear Crank–Nicolson FEM-FCT-scheme for the convection-diffusion equations would be hurt, and a finer time step did not produce significantly better results, as several prestudies showed. The same time step was used for CFD and SPS, i.e., there was exactly one transport step plus one process step per $\Delta t = 0.025$ in the SPS.

The simulations were performed on widely available computing workstations (HP BL460c Gen9 2xXeon, Fourteen-Core 2600MHz). We used eight cores per run, to make use of the inherent shared memory parallelism in the SPS. One simulation run took between 20 and 45 minutes, depending on the parameter setup and the choice of the coagulation parameter.

We ran simulations for all four parameter sets. In all cases, the simulated time was the reported operating time of the experimental tube crystallizer plus 100 seconds. Those additional 100 seconds are the "recording time", from which most of the results that are given in the next section were obtained.

Some more words are in order to collect further algorithmic or purely computational aspects which sped up computation considerably. For the SPS, we made use of the "linear process deferment algorithm" (LPDA) that was proposed in Patterson et al. (2006) and readily implemented in Brush (see Section 4.3.4). Instead of recomputing jump rates and updating particle properties after each occurrence of a "linear" process (particle inception and surface growth) the updates are deferred until the occurrence of a coagulation event and then made good for. Due to the relative rareness of coagulation events compared to growth events, this led to five to ten times faster computations in our setup.

An opportunity for speeding up the computations offered itself in caching

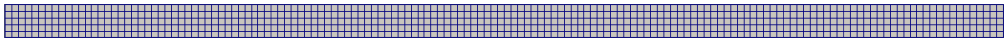

Figure 6.2: The computational mesh consists of 750 considerably stretched rectangular grid cells, aligned in five layers. The graphic above is scaled by a factor 500 in radial direction.

data that was necessary for transferring functions between the two simulation programs. We could re-use the geometric multigrid implementation of ParMooN, as long as the grids for SPS and CFD stood in a hierarchical relation to each other. In the current example, where the grids were identical, this was trivially the case.

6.2.2 Computational results

Simulations were conducted for all four parameter sets, with the goal to reproduce the average particle diameter $\bar{d}$ and its standard deviation σ as they were observed at the outlet of the experimental flow crystallizer (see Table 6.1). On top of that, the simulations had to fulfill several side conditions, so as to prove their physical plausibility. The coagulation kernel K and the coagulation intensity parameter κ had to be determined. In a first, unsuccessful attempt we tried the additive coagulation kernel (6.1). The simulations performed with this kernel led to an excessive coagulation of large particles, resulting in an almost unchanged median but unphysically large outliers of the particle size distribution, when increasing the value of κ. Another try was conducted using the Brownian coagulation kernel (see Chapter 7), but it yielded results very comparable to those gained with the constant coagulation kernel, for which we settled finally. With that kernel we were able to produce physically reasonable results. Values for κ were chosen between 10^{-13} and 10^{-11}, in order to determine that value, which allowed for the best fitting of the experimental $\bar{d}$ at the outlet. The outlet particle size distributions achieved with this parameter study are shown in Figure 6.3. These results were gained by recording the properties, especially the particle mass and diameter, of each particle that left the simulation domain at the outflow, over 10 seconds after the end time of the experiment. The properties of each measured stochastic particle were weighted with the product of its stochastic weight and the volume of the ensemble cell it belonged to last. Thus, the boxplots in Figure 6.3 are weighted boxplots of the "raw" output data of the computational particles. As the results are qualitatively similar for all four parameter sets, only the slowest and fastest setup are shown there.

It can be seen that increasing the coagulation parameter leads to a moderate increase of the median distribution, which is as expected. On the other hand, it also leads to an increased variance, as can be seen by the stretch of the boxes and whiskers. This effect was expected, too, since each collision growth event moves one particle far to the right of the median particle size. Collision growth is, due two its big jump height and relative sparseness, less uniform than surface growth, which happens often and results in small size increases only. One can also see that the number and the range of outliers increases, but not as severely

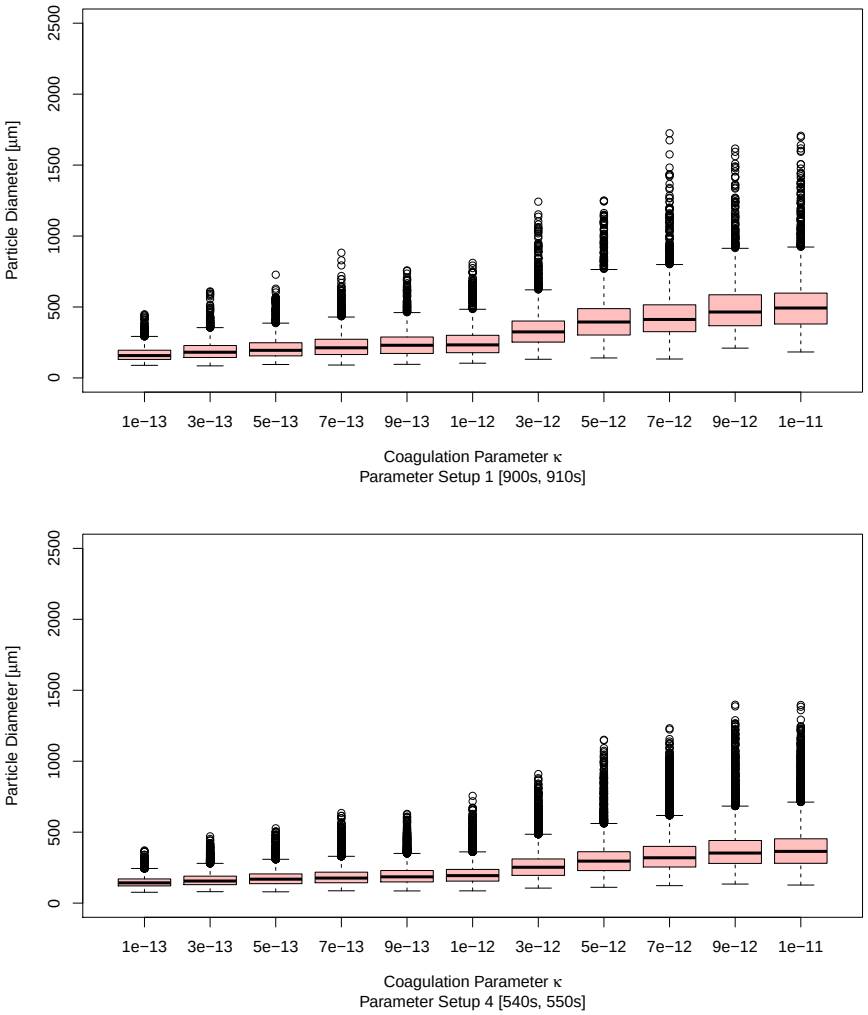

Figure 6.3: Boxplots of PSD for fastest and slowest parameter set, with different choices of the coagulation parameter κ in the constant coagulation kernel. Box whiskers stretch to 1.5 interquartile range of lower/upper quartile, X-axis scale is only ordinal.

as was the case for the additive kernel.

The dependence of the mean particle diameter on κ, disregarding standard deviation, is given in Figure 6.4 for all four parameter sets.

Comparing the results of the parameter studies to the experimental results, best-fitting values of κ could be determined, these range between $5 \cdot 10^{-13}$ and 10^{-12}. The simulation results proved rather sensitive to the parameter. For the two slow setups, Setups 1 and 2, $\kappa_1 = 9 \cdot 10^{-13}$ and $\kappa_2 = 10^{-12}$ proved to be the best choices. For the faster flowing setups, the best parameters were half an order of magnitude smaller, $\kappa_3 = 6 \cdot 10^{-13}$ and $\kappa_4 = 5 \cdot 10^{-13}$. The results (number mean diameter $\bar{d}$ and standard deviation σ) that were

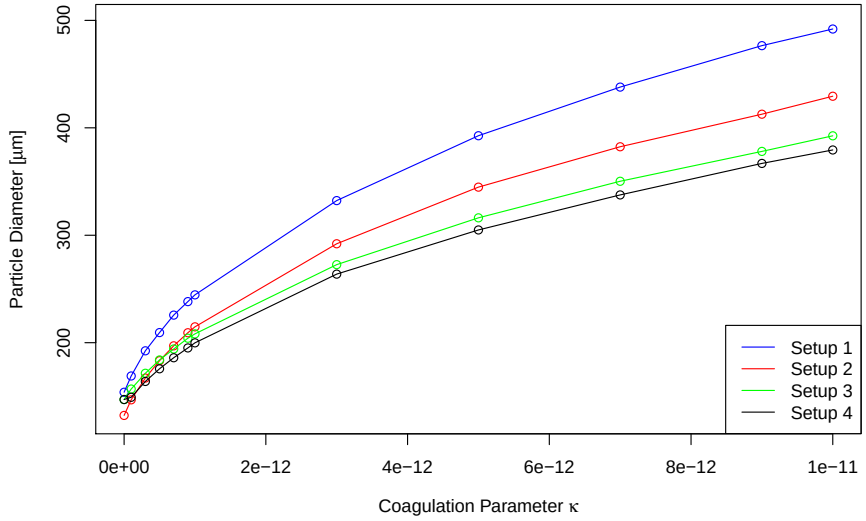

Figure 6.4: Mean particle diameter at the outflow, in dependence of κ, constant coagulation kernel. Data is averaged over 100 seconds past the end time of the respective experiment.

Table 6.3: Computational results for all four parameter sets at the outflow boundary, averaged over the given time interval. (Number) mean particle diameter $\bar{d}$, standard deviation σ, maximal observed particle diameter $d_{\max}$, crystalline ASA mass flux φ_m and mass flow rate $\dot{m}$, scaled so as to easily compare with the experimental data (Table 6.1).

Setup	Time [s]	$\bar{d}$ [μm]	σ [μm]	$d_{\max}$ [μm]	φ_m [kg/m^{2}s]	$\dot{m}$ [g/min]
1	[900,1000]	238	85	953	7.443	1.40
2	[660,760]	215	73	858	11.571	2.18
3	[540,640]	189	56	682	15.088	2.84
4	[540,640]	176	54	645	16.499	3.12

gained with these κ are given in Table 6.3, one should compare them to the experimental values in Table 6.1. One notices that, although the average could be reproduced sufficiently well, our computational results exhibit a somewhat too high standard deviation. The standard deviation of the computed particle size distribution is about 1.1 to 1.5 times as high as the standard deviation of the experimental data. The values were gained by averaging in time over all computational particles that left the computational domain at the outflow within the 100 s "recording time" described above. We also list the diameter of the largest observed particle. Those values are somewhat too large, and if they appeared in the actual experiment (tube diameter is 2000 μm), blockage would be very likely. Indeed, the authors of Eder et al. (2010) report the largest observed particles to be of diameter 500 μm, and observed no blockage. Finally,

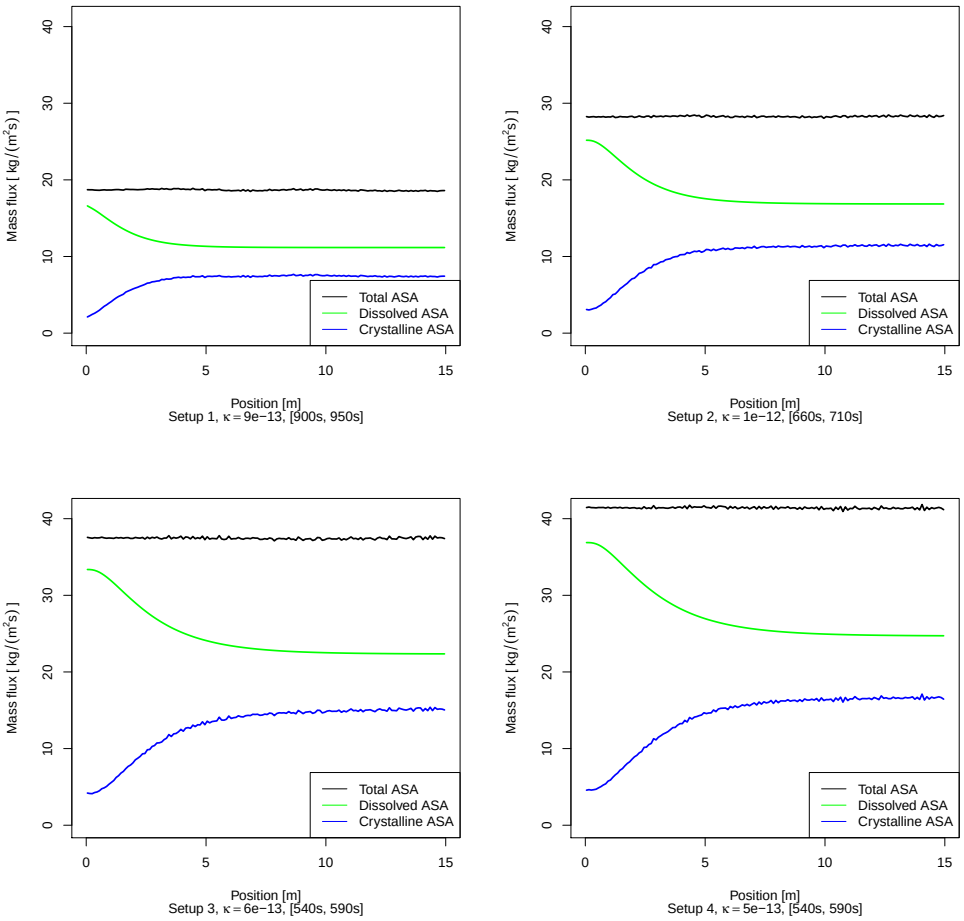

Figure 6.5: Time-averaged development of mass flux of crystalline and dissolved ASA along the tube, all four setups, best coagulation parameter. In all cases, stationary conditions were reached and the total ASA mass flux is constant (in space, on time average) along the tube.

the mass flux and mass flow rate of crystalline ASA (averaged over 100 s) is given in Table 6.3. These values are remarkably close to those reported by the experimentators, see Table 6.1. They serve as one of the side conditions mentioned in the introduction, underpinning the plausibility of the results.

Figures 6.5 and 6.6 show more results in the same spirit, supporting physical plausibility. In Figure 6.5 the ASA mass flux throughout the tube is shown, averaged over 50 s "recording time", when stationary conditions have already been reached for a while. One can see that the total ASA mass flux (crystalline plus dissolved ASA) is constant in z. This is as expected, since the velocity field is divergence-free and ASA mass must neither be lost nor gained. The green and blue lines show how dissolved ASA is used up by surface growth of the crystals. This process takes place, until supersaturation is zero and equilibrium of dissolved and crystalline ASA is reached. One can see that surface growth

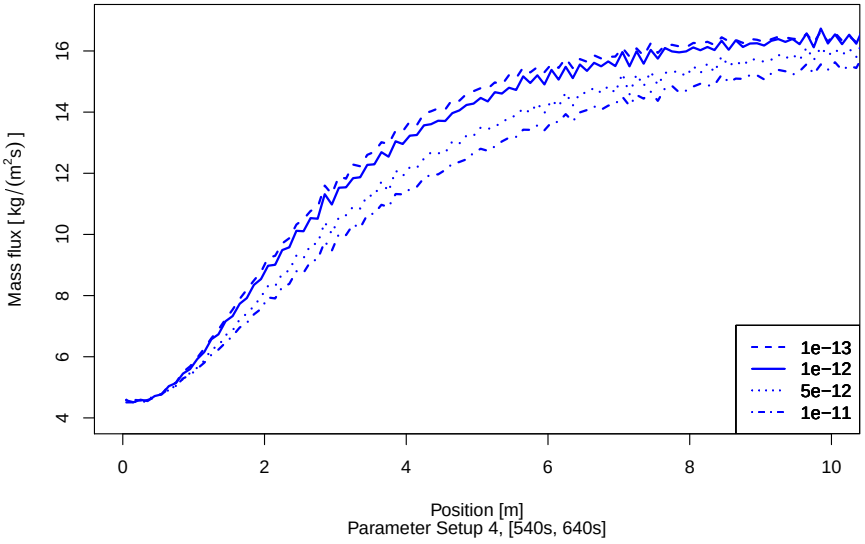

Figure 6.6: Time-averaged mass flux for four different coagulation parameters κ, in the first 10 m of the tube crystallizer after steady-state conditions have been reached.

is quickest in the slowest setup (1), where equilibrium is reached essentially within the first 3 meters of the tube. The faster the flow, the further in the tube equilibrium is reached. In the fastest setup (4), it is not reached before meter 8.

Figure 6.6 shows the first 10 meters of the tube in a close-up. It illustrates an expected, but interesting effect. There is a growth delay by coagulation. In the figure the mass flux for four different, well apart coagulation parameters κ is shown. One observes that the higher the coagulation parameter is, the shallower is the slope of the mass flux curve, i.e., the slower is the transition of ASA from dissolved to crystalline state by surface growth. The reason for this effect is the surface dependence of the growth intensity G_m (see Equation (6.3)). Collision growth maintains mass, but the total crystal surface area is reduced by each coagulation event. Therefore after coagulation less crystal surface is available for new material to attach, and surface growth is slowed down. This effect is clearly visible in Figure 6.6.

Finally, Figure 6.7 shows the net effect of the (simulated) operation of the flow crystallizer. For all four parameter setups, the initial (at $z = 0$) and final (at $z = 15$) distribution is given, in terms of probability. The data was gained the same way as described before, but applying the same procedure at both outlet and inlet. It can be seen that in all four cases the peak of the distribution moves towards the right, and additionally the initial sharpness is somewhat smeared. This effect is the stronger, the slower the flow is, i.e., the more time the particles have to form larger aggregates by coagulation. One can also observe that the histograms of the slowest setup are somewhat "jagged".

Table 6.4: Number mean particle diameter in μm and standard deviation (in brackets), for all four parameter setups and the particular optimal coagulation parameter. The given values refer to the first, second, third and fourth 25 s of the 100 s "recording" time interval at the end of each simulation.

Setup	First 25 s	Second 25 s	Third 25 s	Fourth 25 s
1	240.53 (85.92)	239.60 (85.24)	235.61 (83.51)	237.57 (87.15)
2	213.60 (73.11)	215.05 (73.94)	215.87 (72.90)	214.51 (73.62)
3	189.05 (56.46)	188.29 (56.81)	189.60 (56.55)	189.16 (56.26)
4	175.17 (53.06)	176.63 (53.43)	175.12 (53.22)	175.95 (54.47)

This is due to the lower number of in- and outflowing computational particles per second. The effect could be attenuated somewhat by collecting data for an even longer period of time.

Several further simulations and postprocessing steps were performed in order to foster the reliability of our results. All gave positive results. First we checked the stochastic stability of the final results of Table 6.3. This we did by sectioning the 100 s "recording time" at the end of each experiment into four 25 s intervals and comparing mean diameter and standard deviation of those particles leaving the tube within these intervals. The results are satisfactory conform, and they are shown in Table 6.4. The values are closer together for the faster flowing setups, which is a direct consequence of the higher number of computational particles leaving the domain. We then made sure that our approach of simulating a single trajectory of the stochastic process was sufficient, by running ten independent realizations of one example. We picked the fastest flowing setup again, with optimal coagulation parameter. Averaging the results of these runs (first 10 s of the recording time) gave a mean of 175 μm and, and a mean standard deviation of 53 μm. The fluctuations of the individual runs about those mean values can be quantified by their standard deviation: it is 0.62 for the mean values and 0.64 for the individual standard deviations. We consider both standard deviations sufficiently small to justify the one-trajectory approach in this case.

For the fastest flowing parameter setup we performed grid refinement tests. The grid was refined only in flow direction, thus reducing the stretch somewhat. The results for the best coagulation parameter showed a slight, but systematic dependence on the grid size. The finer the grid, the smaller was the number mean diameter, and the higher the standard deviation. Additionally, we performed the same grid refinement tests for a no-coagulation setup in order to exclude the SPS coagulation algorithm as a source for this dependency. The dependency was qualitatively and quantitatively the same for that setup, see Table 6.5 for the data.

All in all, with our new coupled method, we were able to reproduce the experimentally number mean diameter sufficiently well, with a very similar coagulation parameter for all four setups. The results are also physically plausible, except for a few outliers in size, which do but hardly influence the overall mean-

Table 6.5: Results of grid refinement tests. Parameter setup 4, two choices of coagulation parameter κ, five refinement levels. Values are particle number mean diameter and standard deviation in brackets, both in units μm. One observes a slight decline of the mean value and a slight increase of the standard deviation with refinement.

N cells	150×5	300×5	600×5	1200×5	1500×5
$\kappa = 5 \cdot 10^{-13}$	176 (54)	175 (54)	175 (54)	174 (57)	174 (58)
$\kappa = 0$	140 (31)	140 (30)	140 (31)	139 (33)	138 (34)

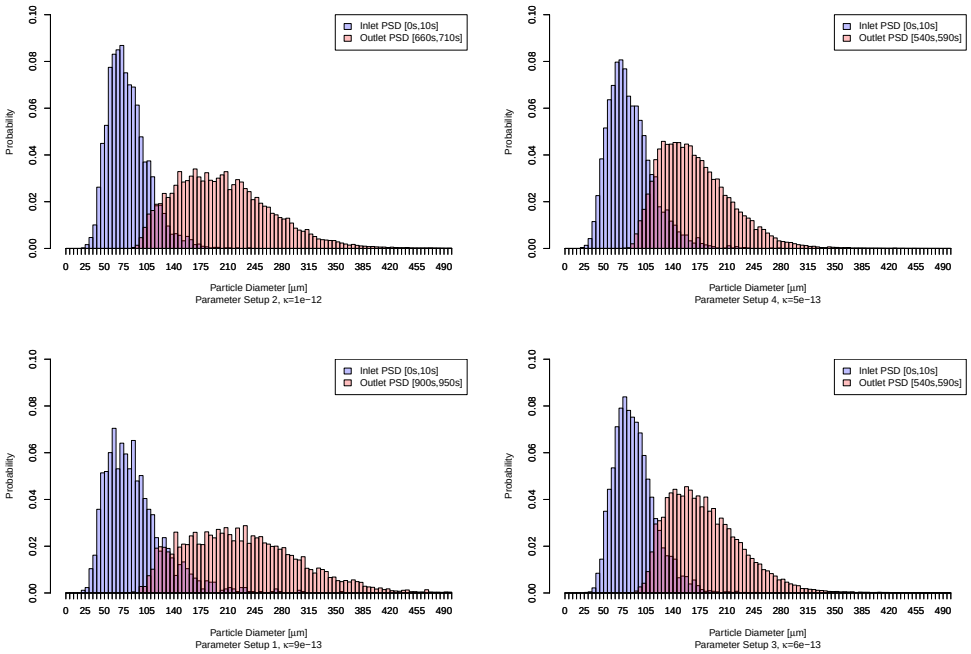

Figure 6.7: Simulation results for all four setups, best coagulation parameter κ. Particle size distribution at inlet and outlet is shown, measured over time interval [0 s, 10 s] (inlet) and the following intervals at the outlet: $[900, 950]$ (setup 1), $[660, 710]$ (setup 2), $[540, 590]$ (setup 3), $[540, 590]$ (setup 4).

ingfulness of the simulation results. The main problem is the overestimation of the standard deviation, the experimental PSD showed less variance. We can identify two sources of variance. One is particle coagulation, the other is the spread in residence time in the tube crystallizer that gets introduced by the discrete parabolic flow profile. The first source of variance could be reduced by re-modeling the proportion of surface growth and collision growth, favoring surface growth even more. Surface growth does not introduce as much variance as collision growth does. The second source of variance could be dealt with by either taking a step back towards a 1d model and transporting all particles with the same velocity, or by augmenting the 2d model. Experimental and

numerical results from Wiedmeyer et al. (2017) suggest that smaller particles travel slower through a tube crystallizer, since they follow the flow microstructure, thus spending more time in the crystallizer, having more time to grow. If this conjecture was applicable to the ASA crystallizer, too, then capturing the effect in the model could counterbalance the variance introduced with the different streaming layers, thus keeping the standard deviation even closer to the experimental data.

6.3 Outlook to 3d

In this section we want to come to a conclusion about the undertaking of this chapter and point out a further direction, focusing on the extension of the method to three spatial dimensions. We will comment on those difficulties that were overcome already and those that lie still ahead when extending the method to 3d in the subsequent chapter.

The newly developed coupling method was successfully applied to the axisymmetric 2d simulation of a flow crystallizer. Experimental results could be well reproduced for four different operating conditions of the crystallizer. The simulation results are physically plausible in all cases, and the computing time was within reasonable bounds. To conduct these simulations it was necessary to find a coupled formulation of the system in question, and to choose accurate and efficient numerical schemes for each subproblem. Transfer of information between the CFD simulation and the SPS had to be implemented and made efficient. The stochastic simulation had to be adapted to two spatial dimensions, since preceding simulations that included advection focused on the 1d case only. Together with the collected simulation experience we consider these achievements a good base for the extension of the method to a full 3d framework.

From the CFD perspective this almost certainly leads to larger problems, with more degrees of freedom and denser matrices, which means that the application of more sophisticated solvers, like those discussed in Section 2.3, will be necessary. Additionally, full 3d simulations are generally used in the context of non-stationary, but instationary laminar, or even turbulent flows. This increases the need for efficient, exact and robust discretizations and solvers for the CFD part of the simulation, both for the velocity field itself and for the transported quantities.

Adapting the SPS to be performed in a 3d flow domain, on the other hand, will require great attention to details, and these details are the main challenges to overcome. Several of the difficulties described below might have appeared in 2d already, but were not apparent in the ASA crystallizer example, because there all streamlines of the flow were aligned. One problem in the extension to 3d will be the choice and inclusion of a 3d geometry library that can be used to represent the spatial discretization of the computing domain for the SPS, managing the cells which hold the ensembles of computational particles. This library must fulfill several requirements, these are listed in Section 7.1.4. One of the requirements is an efficient search algorithm, which can be used to locate particles after the transport step. In case all particles move only within their

former cell or just to neighboring cells (as was the case for the axisymmetric 2d example), then locating all N computational particles after the transport step can be done in $\mathcal{O}(N)$ with a rather naive approach. But if particles overleap cells, a more refined technique, like directional search, will be required to keep the computational effort within bounds. An issue that is closely connected to the geometry are wall boundary conditions. We will have to formulate and implement a scheme which mimics particle-wall collisions. This also makes demands on the 3d geometry library.

All in all, a proof of concept of the new coupled stochastic-deterministic method for population balance systems has been achieved. Yet some work lies ahead before the method can be applied in the context of a full instationary 3d simulation – that is the subject of the subsequent, final chapter of this thesis.

6.4 Details on the modeling

In this section we provide details on the modeling of the axisymmetric 2d example. The order follows roughly that order in which the subjects appear in Section 6.1, where a briefer description of the model was given. Now we want to give background information on modeling decisions and features like the axisymmetric formulation (Subsection 6.4.1), fluid density and assumptions on the fluid composition (Subsection 6.4.2) and compute Dean number and Reynolds number of the flow (Subsection 6.4.3). We will further deduce the prescribed temperature field at the boundary (Subsection 6.4.4), and finally write some words about the implementation of axisymmetric particle inception in the SPS (Subsection 6.4.5).

Note that this section is not intended to be read at once, but the reader is expected to "jump" to any topic of interest from Section 6.1, which is the main chapter on the modeling.

6.4.1 The 2d axisymmetric setup

We aim at simulating the 3d flow crystallizer in two dimensions only. 2d simulations are in general computationally cheaper than 3d simulations, because they typically contain a smaller number of degrees of freedom, and especially because the arising matrices are sparser. Of course the problem must allow for such a reduction of the dimension. This is the case if, e.g., one dimension can be ruled out by the symmetry of the problem. That is exploited in the axisymmetric approach which we will pursue here.

Before settling on the axisymmetric approach and the modifications it makes necessary, let us comment on the main alternative which comes to mind. We would like to term it the "longitudinal section approach". Both approaches are depicted in Figure 6.8. Both have in common that the originally coiled up tube is straightened out. The longitudinal section approach proceeds by making a longitudinal "cut" through the straightened out tube. With this, one gains a rectangular domain. Each mesh cell of this domain is thought of as representing a prism, cuboid or triangular, depending on the 2d grid. This

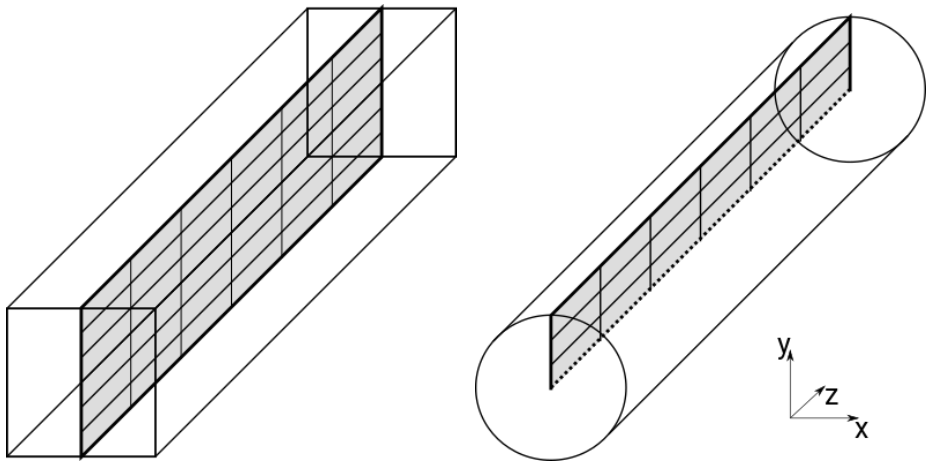

Figure 6.8: The longitudinal section approach (left) and the axisymmetric approach (right) in a schematic graphic. The grey surfaces represent the 2d computational domains. The longitudinal section approach is unsuitable for a circular tube cross section.

leads to the defining assumption that the quantities within the tube do not vary in the direction orthogonal to the cut plane.

There are several problems with the longitudinal section approach. First of all, the assumption of constant functions in x-direction is questionable. Secondly, the cross section of the represented tube is square or rectangular, unless each layer of cells gets "stretched" in the x-direction with an individual value, thus approximating a circular cross section. This approach leads straight into other difficulties. A third problem concerns boundary conditions. It is not possible to impose boundary conditions on the side walls of the domain, since these do not appear in the geometry at all. In addition, the boundary conditions on top and bottom wall are hard to model, because the proportion of surface area to volume will be different than in the real tube. Finally, the flow conditions in a tube with circular cross section cannot be accurately simulated with this approach. The slow-flowing layers near the wall will always be underrepresented, the influence of the fast-flowing layers in the center is overrated.[1] Summing up all these objections, it seems clear enough that the longitudinal section approach is a dead end here.

The axisymmetric method is a more sophisticated (and more realistic) approach to reduce the problem to two dimensions. It exploits the axisymmetry of the domain and postulates axisymmetry of the solution, i.e., one restricts the search for a solution to axisymmetric functions. The main advantage of the axisymmetric model is that it can represent a tube with circular cross section in two dimensions, maintaining the "correct" spatial relation of flow conditions. A major shortcoming of the modeling approach is that non-axisymmetric forces cannot be included.

[1] Especially this point troubled us in a first, unsuccessful, attempt to simulate the tube crystallizer with a longitudinal section approach.

The axisymmetric approach is frequently pursued in applied CFD literature, note, e.g., its application to a similar population balance system in Anker et al. (2015). Nevertheless, there is only a modest amount of theoretical works concerned with it. Ruas (2003) critically observes: "As far as axisymmetric two-dimensional problems are concerned, corresponding quadrilateral or triangular methods originally studied for plane flows have simply been applied all along." A notable exception in form of a textbook is Bernardi et al. (1999), and almost all theoretical results which we use or cite here can be found there. The textbook focuses on spectral methods, but the basics (and that is all we need here) of axisymmetric Stokes and Navier–Stokes equations are introduced in Chapters 1, 2, and 9. Furthermore, Ruas (2003) deals with numerical analysis of standard finite element methods for the axisymmetric Stokes equations. See also references therein for some further works in the same direction.

From an applied mathematician's point of view, it is a rather elementary task to gain an axisymmetric formulation of the Navier–Stokes equations, which can then be "fed" to a finite element code. We will show how it is done, along the lines of Ganesan and Tobiska (2008). The process can be summarized as follows: The standard weak formulation of the time-discretized NSE is transformed to cylindrical coordinates, then all terms that disappear due to symmetry assumptions are canceled and finally the azimuthal coordinate φ is integrated out, giving a factor of 2π on both sides of the equation, which can be divided away. Cylindrical coordinates are the most important ingredient, because they reflect the symmetry properties of the problem and allow for the final reduction to 2d.

From a theoretical point of view things get more involved, because the transformation leads to a setting of *weighted Sobolev spaces*. Weighted Sobolev spaces are a generalization of standard Sobolev spaces, for which most of the classical Navier–Stokes *and* finite element analysis has been performed. In our case they arise naturally: The transformation of integrals to cylindrical coordinates adds, as functional determinant, the factor r (the radial coordinate) to each and every integrand. This additional factor must be treated as part of the functional norm, i.e., r is the weight that constitutes the weighted Sobolev space. To grasp the general idea, let Λ be a domain in $\mathbb{R}^d$ and $\sigma : \Lambda \to \mathbb{R}^+$ a non-negative (positive a.e.) *weight* function. Then for $p \geq 1$ and any measurable real-valued function $u : \Lambda \to \mathbb{R}$ one declares the σ-weighted L^p-norm

$$\|u\|_{p;\sigma} := \left(\int_\Lambda |u|^p \sigma(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \right)^{\frac{1}{p}} .$$

The σ-weighted L^p-space $L^p(\Lambda, \sigma)$ is the space of all measurable functions for which the above integral exists,

$$L^p(\Lambda, \sigma) = \{u : \|u\|_{p;\sigma} < \infty\} .$$

Now the weighted Sobolev space $W^{k,p}(\Lambda, \sigma)$, $k \in \mathbb{N}_0$, consists of all those measurable functions, whose distributional derivatives up to order k exist and lie in $L^p(\Lambda, \sigma)$. In the following definition, α is a multiindex in the usual sense and notation. The space is defined as

$$W^{k,p}(\Lambda, \sigma) = \{u : \|D^\alpha u\|_{p;\sigma} < \infty \quad \forall \alpha \text{ with } |\alpha| \leq k\} .$$

That space is consequently equipped with the weighted Sobolev norm

$$\|u\|_{k,p;\sigma} = \left(\sum_{\alpha:\ |\alpha|\leq k} \|D^\alpha u\|^p \right)^{\frac{1}{p}}.$$

A semi-norm can be introduced by regarding only derivatives in the above formula, and in the case $p = 2$ a scalar product is introduced in the obvious fashion. We will not dwell on elementary properties of these spaces, all of them are Banach, and the $W^{k,2}(\Lambda, \sigma)$ are Hilbert, the interested reader is referred to Kufner (1980), which is an early standard reference for weighted Sobolev spaces.

One also learns from that work that weighted Sobolev spaces have been developed as a framework for partial differential equations with singularities[2] at the boundary $\partial\Lambda$. Therefore in Kufner (1980) the weight function comes always in terms of the distance of a point $\boldsymbol{x}$ to a certain "dangerous" portion of the domain boundary.

Note that in the definitions above one can also make use of the option to introduce different weights for each α-derivative, yet this is not necessary in our case. Anyway, the spaces we are interested in are only those with weights r or r^{-1} and $p = 2$. In cylindrical coordinates r gives the distance of a point from the symmetry axis Γ_{symm}. This symmetry axis is a part of the boundary, and in some sense it is that part of $\partial\Gamma$ where the problem has singularities. So to speak, although in our application the need for weighted Sobolev spaces arises from transformation, in the outcome the weight performs just as was motivated by Kufner (1980): it measures the distance from that boundary part which contains singularities.

In the remainder of this chapter we derive the 2d axisymmetric formulation of the Navier–Stokes equations, the boundary conditions at the symmetry axis, and cite theoretical results which ensure well-definedness. In a second paragraph we do the same for the convection-diffusion equations, and in a final paragraph describe necessary adaptations to the stochastic particle method, in order to use it in an axisymmetric version.

Axisymmetric 2d Navier–Stokes equations. As was stated below, we follow the idea of Ganesan and Tobiska (2008) to transfer the *weak* formulation of all equations to the new domain. This approach means essentially application of integral transformation, and it is therefore more accessible to a reader with background in mathematics, than the more physical approach in Bernardi et al. (1999), where the strong formulation is transformed. The gained variational formulations are the same.

We start from the de-dimensionalized, time-discretized formulation of the NSE, as given in Problem 2.2.1, but take a step back from there to the fully nonlinear problem (and skip the time step subscript k everywhere). Let us write the appearing bi- and trilinear forms in their full integral notation. We sum up

[2] Such singularities may have different sources: perturbed ellipticity, boundary data or details of the boundary geometry.

the three components of the momentum balance equations in order to gain one scalar equation. Recall that $\boldsymbol{b}$ stands for the entire right-hand side of the time-discretized problem. It contains terms that can be treated identically to those on the left-hand side when it comes to axisymmetry. Here is the formulation:

$$\begin{aligned}\int_\Omega \boldsymbol{u}\cdot\boldsymbol{v}\ \mathrm{d}\boldsymbol{x} - \tau\nu\int_\Omega \nabla\boldsymbol{u} : \nabla\boldsymbol{v}\ \mathrm{d}\boldsymbol{x} + \tau\int_\Omega (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\boldsymbol{v}\ \mathrm{d}\boldsymbol{x} \\ +2\tau\int_\Omega \nabla p\cdot\boldsymbol{v}\ \mathrm{d}\boldsymbol{x} = \int_\Omega \boldsymbol{b}\cdot v\ \mathrm{d}\boldsymbol{x}\end{aligned} \tag{6.7}$$

$$2\tau\int_\Omega (\nabla\cdot\boldsymbol{u})q\ \mathrm{d}\boldsymbol{x} = 0. \tag{6.8}$$

In the above formulation, $\boldsymbol{u}$ is the vector of Cartesian velocity components. Whenever we want to make that fact explicit, we write $\boldsymbol{u}_{\mathrm{cart}}$ for the vector of components and u_x, u_y, and u_z for the components. If $\mathbf{e}_x$, $\mathbf{e}_y$, and $\mathbf{e}_z$ denote the unit vectors of the Cartesian coordinate system, then

$$\boldsymbol{u}_{\mathrm{cart}} = u_x\mathbf{e}_x + u_y\mathbf{e}_y + u_z\mathbf{e}_z$$

holds, and accordingly for the Cartesian velocity test function $\boldsymbol{v}$.

Let $\boldsymbol{\psi} : [0,\infty)\times[0,2\pi)\times\mathbb{R}\to\mathbb{R}^3$ denote the standard cylindrical coordinate transformation, $\boldsymbol{\psi}(r,\varphi,z) = (r\cos\varphi, r\sin\varphi, z)$. A change of the reference system to cylindrical coordinates by $\boldsymbol{\psi}$ is what is called a *passive* transformation. The velocity vector field as a physical object stays untouched, but its components change along the change of the basis. One now seeks a cylindrical velocity

$$\boldsymbol{u}_{\mathrm{cyl}} = u_r\mathbf{e}_r + u_\varphi\mathbf{e}_\varphi + u_z\mathbf{e}_z,$$

which is physically equivalent to $\boldsymbol{u}_{\mathrm{cart}}$. The vectors $\mathbf{e}_r$, $\mathbf{e}_\varphi$, and $\mathbf{e}_z$ are the unit vectors of the cylindrical coordinate system, each of them tangential to the respective coordinate plane. In contrast to the Cartesian system, the vectors $\mathbf{e}_r$ and $\mathbf{e}_\varphi$ vary in space. The transformation from cylindrical to Cartesian unit vectors, in terms of (r,φ,z), is given as

$$\begin{pmatrix}\mathbf{e}_x\\ \mathbf{e}_y\\ \mathbf{e}_z\end{pmatrix} = \begin{pmatrix}\cos\varphi & \sin\varphi & 0\\ -\sin\varphi & \cos\varphi & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\mathbf{e}_r\\ \mathbf{e}_\varphi\\ \mathbf{e}_z\end{pmatrix}.$$

The transposed transformation (note that the matrix is orthogonal) applied to the Cartesian velocity vector components gives the cylindrical velocity vector components.[3] In terms of (r,φ,z), the transformation of the velocity components reads:

$$\begin{aligned}u_x\circ\boldsymbol{\psi} &= u_r\cos\varphi - u_\varphi\sin\varphi\\ u_y\circ\boldsymbol{\psi} &= u_r\sin\varphi - u_\varphi\cos\varphi\\ u_z\circ\boldsymbol{\psi} &= u_z.\end{aligned}$$

[3] This is done in the spirit of "(contravariant) physical components" as introduced, and distinguished from actual co- and contravariant components, in (Hung, 2002, p.6). Coordinate transformation is a complicated business.

The same relation is gained in Menküc and Wiechmann (2004), which is a useful source for anyone, who appreciates retracing transformations of any kind step by step.[4] When it comes to the pressure, it is a scalar quantity, and it is not affected by the transformation like the velocity. The relation between "old" and "new" pressure is the identity

$$p_{\text{cart}} \circ \boldsymbol{\psi} = p_{\text{cyl}}.$$

So far, the relations hold for any velocity field and pressure. Since we are only interested in axisymmetric solutions, we remove the dependency of any quantity on the angle φ. This means in particular that all partial derivatives in φ are zero:

$$\frac{\partial}{\partial\varphi}u_r = 0, \quad \frac{\partial}{\partial\varphi}u_\varphi = 0, \quad \frac{\partial}{\partial\varphi}u_z = 0, \text{ and } \frac{\partial}{\partial\varphi}p = 0. \tag{6.9}$$

Additionally, we adopt the assumption of zero tangential velocity from Ganesan and Tobiska (2008),

$$u_\varphi = 0, \tag{6.10}$$

which simplifies the upcoming calculations significantly. This assumption is motivated by the most basic example of an axisymmetric flow, the parabolic velocity solution in a pipe.

We end up with the reduced identities

$$u_x \circ \boldsymbol{\psi} = u_r \cos\varphi, \quad u_y \circ \boldsymbol{\psi} = u_r \sin\varphi, \quad u_z \circ \boldsymbol{\psi} = u_z, \quad p \circ \boldsymbol{\psi} = p. \tag{6.11}$$

These equations hold accordingly for the test functions $\boldsymbol{v}$ and q.

Let us now assume that Ω is an axisymmetric domain, and introduce $\Lambda \subset [0,\infty)\times\mathbb{R}$ with $\Lambda\times[0,2\pi) = \boldsymbol{\psi}^{-1}(\Omega)$. One can picture Ω as gained by rotation of Λ around the z-axis and call Λ the "meridian domain" (Bernardi et al., 1999, p.11). With the functional determinant $\|\det\boldsymbol{\psi}\| = r$ of the cylindrical coordinates, all integrals in (6.7) can be transformed successively. For the first integral, which comes from the time discretization, one has (see (Ganesan and Tobiska, 2008, p.124))

$$\begin{aligned}
\int_\Omega \boldsymbol{u}\cdot\boldsymbol{v}\ \mathrm{d}\boldsymbol{x} &= \int_\Omega u_x v_x + u_y v_y + u_z v_z\ \mathrm{d}\boldsymbol{x} \\
&= \int_{\Lambda\times[0,2\pi)} (u_r\cos\varphi v_r\cos\varphi + u_r\sin\varphi v_r\sin\varphi + u_z v_z)\, r\ \mathrm{d}r\mathrm{d}\varphi\mathrm{d}z \\
&= \int_{\Lambda\times[0,2\pi)} \left(u_r v_r\cos^2\varphi + u_r v_r\sin^2\varphi + u_z v_z\right) r\ \mathrm{d}r\mathrm{d}\varphi\mathrm{d}z \\
&= \int_{\Lambda\times[0,2\pi)} (u_r v_r + u_z v_z)\, r\ \mathrm{d}r\mathrm{d}\varphi\mathrm{d}z \\
&= 2\pi\int_\Lambda \boldsymbol{u}\cdot\boldsymbol{v}\ r\ \mathrm{d}r\mathrm{d}z.
\end{aligned}$$

[4] As does the author of these pages.

In the first step we used the transformation theorem and the identities (6.11) at once, and in the last step we made use of the assumption (6.10).

In the same way one proceeds for the other terms. They do but present the difficulty of comprising differential operators, and these must be transformed into the new coordinate system, too. This is achieved by longish calculations in the spirit of Menküc and Wiechmann (2004). Yet these calculations are simplified enormously by the symmetry assumptions. Let us only give the resulting transformed integrals of the remaining terms of the left-hand side of the momentum balance equation (constant prefactors were left out). They read

$$\begin{aligned}\int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} \;\; \mathrm{d}\boldsymbol{x} &= 2\pi \int_\Lambda \left(\nabla_{r,z} \boldsymbol{u} : \nabla_{r,z} \boldsymbol{v} + \frac{u_r v_r}{r^2} \right) \; r \; \mathrm{d}r\mathrm{d}z \\ \int_\Omega (\boldsymbol{u} \cdot \nabla) \boldsymbol{u} \cdot \boldsymbol{v} \;\; \mathrm{d}\boldsymbol{x} &= 2\pi \int_\Lambda ((\boldsymbol{u} \cdot \nabla_{r,z}) \boldsymbol{u}) \cdot \boldsymbol{v} \; r \; \mathrm{d}r\mathrm{d}z \\ \int_\Omega \nabla p \cdot \boldsymbol{v} \;\; \mathrm{d}\boldsymbol{x} &= 2\pi \int_\Lambda (\nabla_{r,z} p) \cdot \boldsymbol{v} \; r \; \mathrm{d}r\mathrm{d}z.\end{aligned}$$

It is somewhat astounding that the latter two of the above integrands differ from their 2d Cartesian equivalents only by the factor r. This is due to the axisymmetry of the problem, and not in general the case when transforming these integrals from Cartesian to cylindrical coordinates. Yet this circumstance is extremely helpful when adapting a standard (Cartesian) 2d finite element code to the axisymmetric formulation, because all one has to do is multiply the terms in these integrals with the coordinate r.[5]

The right-hand side of the momentum balance equation is skipped here, because no terms with new differential structure appear therein. In particular, the term $\int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \; \mathrm{d}\boldsymbol{x}$ can be treated just as the first integral, if the data $\boldsymbol{f}$ is assumed to be axisymmetric, too.

We finally transform the divergence integral of the continuity equation, where another additional term appears. This is

$$\int_\Omega (\nabla \cdot \boldsymbol{u}) q \;\; \mathrm{d}\boldsymbol{x} = 2\pi \int_\Lambda \left(\nabla_{r,z} \cdot \boldsymbol{u} + \frac{u_r}{r} \right) q \; r \; \mathrm{d}r\mathrm{d}z.$$

With all the integral terms transformed, one can easily write down the complete, time-discrete axisymmetric Navier–Stokes equations in their weak formulation (with $\tilde{\boldsymbol{b}}$ subsuming the right-hand side terms). This equivalent to (6.7) is

$$\begin{aligned}\int_\Lambda \Big[\boldsymbol{u} \cdot \boldsymbol{v} \;\; &- \tau\nu \left(\nabla_{r,z} \boldsymbol{u} : \nabla_{r,z} \boldsymbol{v} + \frac{u_r v_r}{r^2} \right) \\ &+ \tau((\boldsymbol{u} \cdot \nabla_{r,z}) \boldsymbol{u}) \cdot \boldsymbol{v} + 2\tau (\nabla_{r,z} p) \cdot \boldsymbol{v} \Big] \; r \; \mathrm{d}r\mathrm{d}z = \int_\Lambda \tilde{\boldsymbol{b}} \cdot v \; r \; \mathrm{d}r\mathrm{d}z \end{aligned} \tag{6.12}$$

$$2\tau \int_\Lambda \left(\nabla_{r,z} \cdot \boldsymbol{u} + \frac{u_r}{r} \right) q \; r \; \mathrm{d}r\mathrm{d}z = 0. \tag{6.13}$$

The question for well-definedness of that weak formulation brings us to the weighted Sobolev spaces which were mentioned earlier. It will become apparent

[5] And add the additional terms in viscous term and continuity equation, of course.

that radial and axial velocity component need different ansatz spaces. Introduce the spaces

$$H^1(\Lambda, r) := W^{1,2}(\Lambda, r) \qquad V^1(\Lambda, r) := H^1(\Lambda, r) \cap L^2(\Lambda, r^{-1}).$$

These can be used as velocity ansatz and test spaces. Indeed, $H^1(\Lambda, r)$ suffices for the axial component u_z, while the additional regularity of $V^1(\Lambda, r)$ is required for the radial component u_r, as will become clear shortly. The space $L^2(\Lambda, r)$ will suffice for the pressure. For the incorporation of boundary conditions in the sense of traces see (Bernardi et al., 1999, p.203). To show the well-definedness of the bilinear integrals is now an exercise in the application of Hölder's inequality. The well-definedness of the trilinear form makes use of the Sobolev-embedding of $H^1(\Omega)$ into $L^4(\Omega)$, which transfers to an embedding of $V^1(\Lambda, r)$ into $L^4(\Omega, r)$, see (Bernardi et al., 1999, p.204). Boundary conditions at the symmetry axis Γ_{symm} arise necessarily here. They are ((Ganesan and Tobiska, 2008, p.125))

$$u_r = 0 \text{ on } \Gamma_{\text{symm}} \quad \text{and} \quad \frac{\partial u_z}{\partial r} = 0 \text{ on } \Gamma_{\text{symm}}. \tag{6.14}$$

The Dirichlet condition for u_r is a consequence of the requirement $u_r \in V^1(\Lambda, r)$, which is easy to see if one assumes u_r to be continuous on Γ. In the non-continuous case it holds, too. This can be looked up in ((Bernardi et al., 1999, p.29)) and references therein.

The Neumann condition for u_z arises, as is common for conditions of this type, as a natural boundary condition when assuming regularity of the solution and recovering the strong formulation of the equation. When showing this, we restrict ourselves to the viscous term. Assume $\boldsymbol{u}$ to be a classical, smooth solution. We make use of the identity

$$\int_\Gamma (f \nabla \cdot \boldsymbol{G} + \boldsymbol{G} \cdot \nabla f) \ \mathrm{d}\boldsymbol{x} = \int_{\partial\Gamma} f(\boldsymbol{G} \cdot \mathbf{n}) dS, \tag{6.15}$$

for a scalar function f and a smooth vector field $\boldsymbol{G}$. The identity is a generalization of Green's formula and follows directly from the divergence theorem. Setting $\boldsymbol{G} = r\nabla_{r,z} u_z$ and $f = v_z$, we can proceed as follows. By definition one has:

$$\begin{aligned}\int_\Lambda \left(\nabla_{r,z}\boldsymbol{u} : \nabla_{r,z}\boldsymbol{v} + \frac{u_r v_r}{r^2}\right) \ r \ \mathrm{d}r\mathrm{d}z \\ = \int_\Lambda \left(\nabla_{r,z} u_r \cdot \nabla_{r,z} v_r + \nabla_{r,z} u_z \cdot \nabla_{r,z} v_z + \frac{u_r v_r}{r^2}\right) \ r \ \mathrm{d}r\mathrm{d}z.\end{aligned}$$

Regard only the second term of the right-hand side and apply (6.15):

$$\begin{aligned}\int_\Lambda (\nabla_{r,z} u_z \cdot \nabla_{r,z} v_z) \ r \ \mathrm{d}r\mathrm{d}z \\ = -\int_\Lambda \nabla_{r,z} \cdot (r\nabla_{r,z} u_z) v_z \ r \ \mathrm{d}r\mathrm{d}z + \int_{\partial\Lambda} v_z (r\nabla_{r,z} u_z \cdot \mathbf{n}) dS.\end{aligned}$$

The boundary term on the right-hand side contains a contribution from the symmetry axis:

$$\int_{\Gamma_{\text{symm}}} v_z(r\nabla_{r,z}u_z \cdot \mathbf{n})\ \mathrm{d}S = \int_{\Gamma_{\text{symm}}} v_z r\frac{\partial u_z}{\partial r}\ \mathrm{d}S.$$

As there is no term on the right-hand side of (6.12) to balance this contribution, one may conclude that $\partial u_z/\partial r = 0$ must hold in a variational sense, and therefore the boundary condition arises naturally.

Axisymmetric 2d convection-diffusion equations The transformation of the convection-diffusion equations to axisymmetry is fairly easy compared to the NSE, because c is just a scalar quantity. The transformation is given in (Anker et al., 2015, p.99), we restate the resulting equations here. A time-discretized model equation with test function χ and bundled right-hand side b in Cartesian coordinates and weak form is

$$\int_\Omega c\chi\ \mathrm{d}\boldsymbol{x} + \tau\varepsilon\int_\Omega \nabla c\cdot\nabla\chi\ \mathrm{d}\boldsymbol{x} + \tau\int_\Omega (\boldsymbol{u}\cdot\nabla c)\chi\ \mathrm{d}\boldsymbol{x} = \int_\Omega b\chi\ \mathrm{d}\boldsymbol{x}. \tag{6.16}$$

The time step subscript has been left out again. As before, we assume that the data is axisymmetric. This includes in particular the velocity field $\boldsymbol{u}$, which acts as a parameter here. Additionally, we only look for such solutions, which are axisymmetric themselves, i.e., fulfill

$$\frac{\partial c}{\partial \varphi} = 0.$$

As it was for the NSE, this assumption will substantially simplify the integrands. Transforming the integrals on the left-hand side one by one (ignoring prefactors) gives

$$\begin{aligned}
\int_\Omega c\chi\ \mathrm{d}\boldsymbol{x} &= 2\pi\int_\Lambda c\chi\ r\ \mathrm{d}r\mathrm{d}z\\
\int_\Omega \nabla c\cdot\nabla\chi\ \mathrm{d}\boldsymbol{x} &= 2\pi\int_\Lambda \nabla_{r,z}c\cdot\nabla_{r,z}\chi\ r\ \mathrm{d}r\mathrm{d}z\\
\int_\Omega (\boldsymbol{u}_{\text{cart}}\cdot\nabla c)\chi\ \mathrm{d}\boldsymbol{x} &= 2\pi\int_\Lambda (\boldsymbol{u}_{\text{cyl}}\cdot\nabla_{r,z}c)\chi\ r\ \mathrm{d}r\mathrm{d}z,
\end{aligned}$$

the right-hand side follows accordingly. Well-definedness of the above integrals is assured if one demands $c,\ \chi \in H^1(\Lambda, r)$ with the weighted Sobolev space as defined in the preceeding paragraph. A boundary condition for c at the symmetry axis arises naturally, the calculation is the same as was done for u_z above. The condition is

$$\frac{\partial c}{\partial z} = 0 \text{ on } \Gamma_{\text{symm}},$$

which should hold in the sense of a distributional derivative. This remark concludes our short discussion of axisymmetric convection–diffusion equations.

Axisymmetric stochastic particle method When it comes to the transformation of the population balance equation, we stick to a remark in (Anker et al., 2015, p.99). Due to the nature of the velocity field that we apply later, which has $u_\varphi = 0$ by assumption and even $u_r = 0$, the transport term does not change in the weak formulation beyond the multiplication with the functional determinant r. In a more general setting, the axisymmetric formulation of the PBE would require a more in-depth discussion, but here we restrict ourselves to discussing the necessary amendments of the stochastic particle method.

We made essentially two changes to the code. To scale the properties of any stochastic particle ensemble to the scale of its cell, the volume of the containing cell must be computed in an axisymmetric fashion. If K is a 2d computational cell which holds a particle ensemble, then K is actually the generating surface of a rotational body K_{rot}. So for the represented volume holds

$$V = 2\pi A(K)R,$$

where $A(K)$ is the surface area of K and R the distance (r-component) of its centroid.

A second necessary adaptation concerns the boundary conditions at the insertion boundary, which is discussed in more detail in Section 6.4.5.

A further issue is the delocalization of coagulation, which is a consequence of the spatial discretization approach of the SPS. Coagulation is discretized in such a way that it may only happen between computational particles which are located in the same ensemble, i.e., in the same cell. In the axisymmetric setup, where the computational cells represent rotational bodies, this means that the possible coagulation radiuses of the particles differ immensely. Particles in the outermost cells can coagulate with other particles that are in the same "flow layer", yet at the opposite wall of the tube, since both are represented by the same ensemble. In that sense, one might ask, whether the presence of particle collision growth does not counter-indicate an axisymmetric approach.

Finally let us make two remarks on issues which *should* be tackled, but were not, because our setup lacks the triggering features. First, if an independent diffusion of the particles was included in the method, e.g., in style of random diffusion jumps, it would be necessary to account for the deformation of space by the transformation. Secondly, there was no call for boundary conditions at the symmetry axis. Since in our case the radial velocity u_r disappears, particles move on trajectories that are strictly parallel to the axis. Therefore the symmetry boundary does not have any influence. If there would be a non-zero radial velocity component, boundary collisions could occur and one would have to introduce reflecting conditions of some kind. In a fully three-dimensional setting this problem will be faced in Chapter 7.

6.4.2 Solution density and inflow conditions for dissolved ASA

In modeling the system, especially when it comes to determining simulation input parameters from the parameters given in Eder et al. (2010), it is indispensable to make assumptions on the density ρ_{susp} (or just plain ρ in the following) of the suspension in the crystallizer. Physically, many processes that take place

in the tube will influence the density, and therefore ρ will vary in space and time. Yet from a computational point of view, a varying density is a big issue, because it enters the Navier–Stokes equation for the velocity as a parameter. If it varies, Navier–Stokes numerics gets more and more involved. Therefore it is useful to assume a *constant* density ρ in space and time throughout the simulation. Here is a glance at the consequences: We must assume the density to be unaffected by the local composition of the fluid, i.e., the variation in the proportion of ethanol, crystalline and dissolved ASA, unaffected by changes in the temperature field, and especially unaffected by the fluid pressure, which leads to the incompressibility assumption in the Navier–Stokes equations. All in all, we assume density variations to be small enough to be neglected in the regarded system.

Determining a reasonable spatial and temporal mean value of the suspension density, which can then be used as its constant value throughout the modeling and simulation process is the aim of this section. We will try do so in accordance to the experimental data and the assumptions (and measurements) which the authors of Eder et al. (2010) make, but we must note that not all of these can be fulfilled simultaneously.

To get the crystallizer started, the suspension that flows into the device comes from two different vessels. One contains a hot, only slightly undersaturated solution of ASA in ethanol. All quantities which describe the fluid in this vessel are subscripted with "sol" in the following, suggesting that the vessel contains a *sol*ution. A second subscript will denote the substance, which the quantity refers to. To give an example, $\beta_{\text{diss, sol}}$ $[\text{kg}/\text{m}^3]$ denotes the mass concentration of dissolved ("diss") ASA in the stream coming from the first vessel. Complementary, $\beta_{\text{EtOH, sol}}$ names the mass concentration of the solvent, ethanol, in the solution stream.

The second vessel contains the seed crystals, surrounded by a ASA-EtOH solution in equilibrium. We call the stream which comes from this vessel the *seed stream*. Quantities connected to the seed stream are subscripted "seed". Thus, $\beta_{\text{diss, seed}}$ is the mass concentration of dissolved ASA in the seed stream, $\beta_{\text{cryst, seed}}$ is its mass concentration of crystalline ASA, and $\beta_{\text{EtOH, seed}}$ is the mass concentration of ethanol in the seed stream. As described in Subsection 6.1.1, the two streams are combined into one in a Y-fitting. This combined stream then feeds the tube crystallizer, and therefore we name it the *feed stream* and all respective quantities get the subscript "feed". We will now show how we determined the density of the feed stream, and reason that its density is a good approximation to the constant overall density.

In the combined feed stream three species will be present. Therefore we can write

$$\rho_{\text{feed}} = \beta_{\text{EtOH, feed}} + \beta_{\text{diss, feed}} + \beta_{\text{cryst, feed}}.$$

While passing through the crystallizer ASA will go from dissolved into undissolved state by surface growth, and the overall temperature of the suspension will drop due to cooling. Since we assumed the density to be unaffected by both these circumstances, the density of ρ_{feed} will be preserved throughout the tube. Therefore ρ_{feed} must be modeled in such a way that it is representative

for the entire system.

As is the density of any homogeneous substance, ρ_{feed} is defined as

$$\rho_{\text{feed}} = \frac{m_{\text{feed}}}{V_{\text{feed}}}.$$

Since mass is conserved, we may write

$$\rho_{\text{feed}} = \frac{m_{\text{EtOH, feed}} + m_{\text{diss, feed}} + m_{\text{cryst, feed}}}{V_{\text{feed}}}.$$

One can see that the density of the suspension depends on how the volumes of the components combine to the volume of the suspension. Since we are not aware of any detailed model for our specific case, three sufficiently simple theoretical assumptions come to mind.

Assumption 1: Conservation of volume – ideal mixture. An ideal mixture is one, where the volume of the components is preserved. Here that would translate to

$$V_{\text{feed}} = V_{\text{EtOH, feed}} + V_{\text{diss, feed}} + V_{\text{cryst, feed}}$$

.

Assumption 2: Conservation of solvent volume – ideal solution. A second possibility is to assume that the volume of the solvent (EtOH) does not change in the process of adding ASA:

$$V_{\text{feed}} = V_{\text{EtOH, feed}}$$

This approach leads to a significant rise of the suspension density. According to (Randolph and Larson, 1988, p.82) the "ideal solution" assumption is fair enough for "suspension density less than 15 %". In our system, this amount is exceeded by far, plus it is a suspension we deal with, not a solution.

Assumption 3: Preservation of solvent density. This is not an assumption on the combined volume, but on the combined density itself. To consider it here is motivated by a remark in (Eder et al., 2010, p.2256), which reads "For the ASA-EtOH system, the [...] saturated solution density $\rho_{sol,sat}$ ([is] assumed to be equal to the density of EtOH) [...]". The authors state this when calculating the overall mass transfer coefficient. Although we do not deal with a solution but with a suspension, for this assumption we would take ethanol to be the solvent and ASA (dissolved *and* crystalline) to be the solute. Then the assumption would correspond to the choice:

$$\rho_{\text{feed}} = \rho_{\text{EtOH}}.$$

The authors of Eder et al. (2010) give two more hints on the overall density. The first is a measured density of the suspension at 40 °C, it is 898 kg/m^3 (Eder

et al., 2010, p.2253). The second is a conversion between molar concentration and mole fraction. There the authors state that in the *sol* stream the relation $c_{\text{diss, sol}} = 2.19 \cdot 10^3 \ {}^{\text{mol}}/_{\text{m}^3} \leftrightarrow \chi_{\text{diss, sol}} = 0.113 \ {}^{\text{mol}}/_{\text{mol}}$ holds for the amount of dissolved ASA. A simple, yet tedious calculation shows that this relation holds if and only if the "ideal solution" assumption holds.

Given the input data in Table 6.1, one can now exercise each of these assumptions for the *feed* stream. We will postpone these details. Let us for the moment conclude that

- for each of the three assumptions there is a hint in Eder et al. (2010) that would justify it
- because we postulated the density to be unaffected by ASA phase transition, we must choose exactly one of the assumptions (and cannot, e.g., choose different ones for dissolved and crystalline ASA)
- since Assumption 1 (ideal mixture) gives the best approximation to the value 898 ${}^{\text{kg}}/_{\text{m}^3}$ given by the authors, we stick to that assumption.

That is to say, with Assumption 1 we gain as total density:

$$\rho = 916.87 \ {}^{\text{kg}}/_{\text{m}^3}$$

and this value will be used furthermore.

Let us now go even further into detail and underpin the decision for Assumption 1 (ideal mixture) by giving the complete reasoning and calculation behind that value. At the same time, we will sort out the composition of the feed stream and thereby justify the inflow boundary values for crystalline and dissolved ASA which were used in Subsections 6.1.4 and 6.1.5.

All four different parameter sets from Eder et al. (2010) have the ratio of the volumetric flow rates from streams *seed* and *sol* in common. It is $1 : 3.5$ in each case. Also, the substance composition in the vessels is the same for each parameter set. Let us start with the composition of the *sol* stream. The authors state that the solution was gained by dissolving 0.05 kg of ASA per 0.1 kg of ethanol. Thus we know the mass fractions $w_{\text{diss, sol}} = \frac{1}{3}$ and $w_{\text{EtOH, sol}} = \frac{2}{3}$. Given the densities of ASA and EtOH (Table 6.2), one has $V_{\text{ASA}}(0.05 \text{ kg}) = 3.7 \cdot 10^{-5} \text{ m}^3$ and $V_{\text{EtOH}}(0.1 \text{ kg}) = 12.66 \cdot 10^{-5} \text{ m}^3$. Depending on which of the density/volume assumptions we follow, there will be different total densities in the outcome. The one which leads us closest to the given $\rho = 898 \ {}^{\text{kg}}/_{\text{m}^3}$ is the ideal mixture assumption. With that assumption both volumes simply add up:

$$\rho_{\text{sol}} = \frac{m_{\text{diss, sol}} + m_{\text{EtOH, sol}}}{V_{\text{sol}}} = \frac{0.05 + 0.1}{3.7 \cdot 10^{-5} + 12.66 \cdot 10^{-5}} \approx 916.87 \ {}^{\text{kg}}/_{\text{m}^3}. \tag{6.17}$$

Note that the "ideal solution assumption" overestimates the density severely (1185 ${}^{\text{kg}}/_{\text{m}^3}$), while the "preservation of solvent volume assumption" underestimates it (790 ${}^{\text{kg}}/_{\text{m}^3}$).

For the mass concentrations in the solution stream one obtains

$$\beta_{\text{diss, sol}} = w_{\text{diss, sol}}\rho_{\text{sol}} = 305.62 \text{ kg/m}^3$$
$$\beta_{\text{EtOH, sol}} = w_{\text{EtOH, sol}}\rho_{\text{sol}} = 611.25 \text{ kg/m}^3.$$

Let us now turn our attention to the *seed* stream. It consists of the same mass fractions of ASA and EtOH, as does the sol stream, with the difference that here a share of the ASA is in crystalline state. This, was our assumption, does not affect the suspension density. We can conclude that the density is the same as above: $\rho_{\text{seed}} = \rho_{\text{sol}}$. But we do still need the mass concentrations of dissolved and crystalline ASA in order to determine inlet boundary conditions for both. Recall that

$$\rho_{\text{seed}} = \beta_{\text{diss, seed}} + \beta_{\text{cryst, seed}} + \beta_{\text{EtOH, seed}}$$

holds. It is clear that $\beta_{\text{EtOH, seed}} = w_{\text{cryst, seed}}\rho_{\text{seed}} = 611.25 \text{ kg/m}^3$ and that $\beta_{\text{diss, seed}} + \beta_{\text{cryst, seed}} = w_{\text{ASA, seed}}\rho_{\text{seed}} = 305.62 \text{ kg/m}^3$. In order to determine which part of this total ASA mass concentration is in dissolved form, one has to know the solubility of ASA in EtOH at the given temperature $T_{\text{seed}} = 24.6$ °C (Eder et al., 2010, p.2249). There the authors suggest a Nyvilt-type model with fitted parameters. According to them, the temperature-dependent solubility of ASA in EtOH can be modeled as

$$\chi_{\text{ASA, sat}}(T) = 10^{27.769 + \frac{-2500.906}{T} - 8.323\log_{10}(T)},$$

with T in K. This relation is shown in Figure 6.9. Here the saturation concentration is given in mole fraction, i.e., in mol dissolved ASA per mol of EtOH *and* dissolved ASA. At the temperature T_{seed} this formula gives a mole fraction $\chi_{\text{ASA,sat}} \approx 0.0611$. Translating this into mass concentration of EtOH, crystalline and dissolved ASA in the seed stream will be achieved by the detour of mass fractions $w_{\cdot\text{ ,seed}}$. From the known $\chi_{\text{ASA, sat}}$ one computes (by knowing the molar masses of ASA and EtOH) the mass fraction of dissolved ASA in a saturated ASA-EtOH solution. It is

$$w_{ASA,sat} = \frac{\chi_{\text{ASA, sat}} M_{\text{ASA}}}{\chi_{\text{ASA, sat}} M_{\text{ASA}} + (1 - \chi_{\text{ASA, sat}}) M_{\text{EtOH}}} = 0.2029 \text{ kg/kg}.$$

In the seed stream the EtOH mass fraction is known, it is $w_{\text{EtOH, seed}} = \frac{2}{3}$. The leftover one third is distributed among dissolved and solid ASA. We then proceed with the following equation for the EtOH mass fraction in the entire seed stream:

$$w_{\text{EtOH, seed}} = (w_{\text{EtOH, seed}} + w_{\text{diss, seed}})\,(1 - w_{\text{ASA, sat}})\,.$$

The intuition behind this equation is that in equilibrium $1 - w_{\text{ASA, sat}}$ is the mass fraction of EtOH at the sum of dissolved ASA and EtOH. Isolating $w_{\text{diss, seed}}$ and inserting the known values, one gets

$$w_{\text{diss, seed}} = \frac{\frac{2}{3}}{1 - 0.2029} - \frac{2}{3} \approx 0.1697 \text{ kg/kg}$$

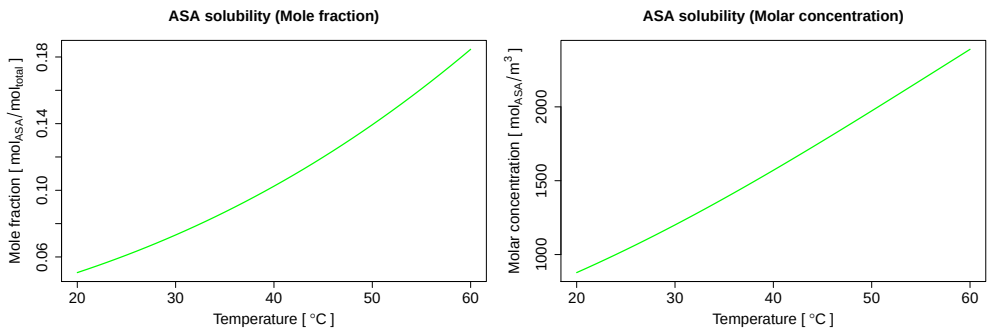

Figure 6.9: Solubility of ASA in EtOH (saturation concentration) according to a fitted Nyvilt model. Left: In terms of mole fraction. Right: In terms of molar concentration, under additional assumptions on density and solution behavior.

and by subtraction

$$w_{\text{cryst, seed}} = 1 - w_{\text{EtOH, seed}} - w_{\text{diss, seed}} \approx 0.1637.$$

With the mass fractions known and the constant density of 916.87 $^{\text{kg}}/_{\text{m}^3}$, we are now able to give the different mass concentrations in the seed stream. They are

$$\beta_{\text{diss, seed}} = 155.59\ ^{\text{kg}}/_{\text{m}^3} \quad \beta_{\text{cryst, seed}} = 150.1\ ^{\text{kg}}/_{\text{m}^3} \quad \beta_{\text{EtOH, seed}} = 611.2\ ^{\text{kg}}/_{\text{m}^3}.$$

The last step in calculating the composition of the *feed* stream is combining the *sol* and *seed* stream. We recall that for each parameter set, they combine in the volume ratio $1 : 3.5$ (*seed* : *sol*). If for their mixing behavior we employ the "ideal mixture" hypothesis again, we can write

$$\beta_{\text{EtOH, feed}} = \frac{1}{4.5}\beta_{\text{EtOH, seed}} + \frac{3.5}{4.5}\beta_{\text{EtOH, sol}} = 611.2\ ^{\text{kg}}/_{\text{m}^3}.$$

Accordingly, we get $\beta_{\text{diss, feed}} = 272.28\ ^{\text{kg}}/_{\text{m}^3}$ and $\beta_{\text{cryst, feed}} = 33.35\ ^{\text{kg}}/_{\text{m}^3}$. In terms of molar concentration, $c_{\text{diss, feed}} = 1511.3\ ^{\text{mol}}/_{\text{m}^3}$ is the dissolved ASA concentration at the inflow. These values form the basis for the inflow boundary values in Subsections 6.1.4 and 6.1.5, and for the mass flow computation in Subsection 6.4.4.

6.4.3 Reynolds number and Dean number of the flow

We will calculate the Reynolds and Dean number of the flow, to show that it stays well in the laminar regime and the development of secondary (Dean) vortices need not be expected.

Let us start with the Reynolds number. For a flow in a tube with spherical cross section the Reynolds number is typically calculated as

$$\text{Re} = \frac{d\rho\bar{u}}{\mu}.$$

Table 6.6: Key figures of the flow, for the four different parameter setups. Total mass flow was calculated based on volumetric flow rate data in Eder et al. (2010) and the density $\rho = 916.87$ $^{\mathrm{kg}}/_{\mathrm{m}^3}$ calculated in Section 6.4.2.

Quantity	Total mass flow	Mean velocity	Reynolds number	Dean number
Unit	$^{\mathrm{kg}}/_{\mathrm{s}}$	$^{\mathrm{m}}/_{\mathrm{s}}$	-	-
Setup 1	0.000174	0.0605	11.1	1.6
Setup 2	0.000263	0.0912	16.7	2.4
Setup 3	0.000348	0.1209	22.2	3.1
Setup 4	0.000385	0.1337	24.5	3.5

Here d is the tube diameter, $d = 0.002$ m, and ρ is the density of the medium. From the previous section we know that $\rho = \rho_{\mathrm{susp}} = 916.87$ $^{\mathrm{kg}}/_{\mathrm{m}^3}$. Further μ is the dynamic viscosity of the fluid. The authors of Eder et al. (2010) measured μ with a rheometer and determined it to be $\mu = 0.01$ $^{\mathrm{kg}}/_{\mathrm{ms}}$. Finally, $\bar{u}$ is the mean velocity, which was determined from the volumetric flow rates, which the authors reported for the four different setups, see Table 6.6 for the precise data. One can see that even for the fastest flowing setup the Reynolds number is only 24.5 and thus comfortably within the laminar regime.

For the calculation of the Dean number, which takes into account the curvature of the pipe, we use the following formula:

$$\mathrm{De} = \sqrt{\frac{r}{R}}\mathrm{Re}.$$

This is the recommended formula of (Berger et al., 1983, p.473), where several different possible definitions of the Dean number for laminar curved pipe flows are discussed. Here r is the inner tube radius, and R the radius of the coil. The authors of Eder et al. (2010) report that the crystallizer was wound up on a tube of diameter 0.1 m, therefore we set $R = 0.05$ m. The resulting Dean numbers for all setups are listed in Table 6.6. One observes that the ratio of inner radius to coil radius is so small that the Dean numbers stay in the order of 1. Comparing to the theoretical considerations in Dean (1928) and Hämmerlin (1958) this is beyond the critical threshold Dean number of 37, where secondary vortices start to develop.[6] Therefore we have strong hints that the flow in the experimental tube crystallizer can be well enough approximated by a classical Hagen–Poiseuille type flow in a straightened out tube.

6.4.4 Details on the temperature profile at the wall boundary

In the energy balance equation, we prescribe the temperature field at the wall boundary. This is done because the tubing material, whose thermal properties

[6] Note that the definitions of the critical number differ somewhat. In (Hämmerlin, 1958, p.223) it is given as $\frac{2\mathrm{Re}^2 d}{R} = 5740$. The threshold 37 given above is its translation to our definition of De, which follows Berger et al. (1983).

determine the heat loss through the pipe wall, is not part of the simulation domain. Instead we try to capture the influence of heat loss through the tube wall with the following modeling assumptions.

We start from Newton's law of cooling, applied to a two-dimensional cross-section of the pipe, which we let move at constant speed u_z through the pipe for the moment. The law reads

$$\frac{\mathrm{d}\tilde{Q}}{\mathrm{d}t} = hA(T(t) - T_{\mathrm{amb}}). \tag{6.18}$$

Here $\tilde{Q}$ [J/m] is the thermal energy per meter of the "slice", $T(t)$ [K] its temperature at time t (assumed to be constant over the cross section), and $T_{\mathrm{amb}} = 297.5$ K is the constant temperature of the surrounding air. The scaling constants are the (one-dimensional) surface of heat exchange $A = 2\pi r_{\mathrm{out}}$ [m] and the heat transfer coefficient h [W/m²·K]. This coefficient encodes thickness and insulation properties of the tubing:

$$h = \left(\frac{r_{\mathrm{out}}}{r_{\mathrm{in}} h_{\mathrm{in}}} + \frac{r_{\mathrm{out}} \ln\left(\frac{r_{\mathrm{out}}}{r_{\mathrm{in}}}\right)}{\lambda_{\mathrm{tube}}} + \frac{1}{h_{\mathrm{out}}} \right)^{-1}.$$

Terms like this appear frequently in the engineering literature (see, e.g., Harrison (2010)), the inserted values are taken from Eder et al. (2010) and listed in Table 6.7.

The left-hand side of (6.18) can be replaced by applying the identity

$$\frac{\mathrm{d}\tilde{Q}}{\mathrm{d}t} = \frac{\mathrm{d}T}{\mathrm{d}t}\tilde{C}, \tag{6.19}$$

where $\tilde{C}$ [J/K·m] is the heat capacity per meter suspension, and with $\tilde{m}_{\mathrm{susp}}$ [kg] being the *mass* per meter suspension:

$$\tilde{C} = \tilde{m}_{\mathrm{susp}} c_{p,\mathrm{susp}},$$

where $c_{p,\mathrm{susp}}$ [J/kg·K] is the specific heat capacity of the suspension.

Restating (6.18) by using (6.19) gives:

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{hA}{\tilde{m}_{\mathrm{susp}} c_{p,\mathrm{susp}}}(T(t) - T_{\mathrm{amb}}). \tag{6.20}$$

As the slice is assumed to move with constant velocity u_z (arising from the parameter setup!) through the tube, a change of variables to axial position z results in

$$\frac{\mathrm{d}T}{\mathrm{d}z} = \frac{hA}{\dot{m}_{\mathrm{susp}} c_{p,\mathrm{susp}}}(T(z) - T_{\mathrm{amb}}), \tag{6.21}$$

where we replaced $\tilde{m}_{\mathrm{susp}} u_z = \dot{m}_{\mathrm{susp}}$ [kg/s], the total mass flow rate. The product of mass flow rate and suspension specific heat capacity is computed as

$$\dot{m}_{\mathrm{susp}} c_{p,\mathrm{susp}} = \dot{m}_{\mathrm{ASA}} c_{p,\mathrm{ASA}} + \dot{m}_{\mathrm{EtOH}} c_{p,\mathrm{EtOH}},$$

Table 6.7: Outer pipe radius r_{out}, inner pipe radius r_{in}, inner heat transfer coefficient h_{in}, outer heat transfer coefficient, h_{out}, thermal conductivity of the tubing material λ_{tube}, ambient temperature T_{amb}, and specific heat capacity $c_{p,\cdot}$ of ASA and EtOH.

Quantity	r_{out}	r_{in}	h_{in}	h_{out}	λ_{tube}	T_{amb}	$c_{p,\text{ASA}}$	$c_{p,\text{EtOH}}$
Unit	mm	mm	$\text{W}/\text{m}^2\cdot\text{K}$	$\text{W}/\text{m}^2\cdot\text{K}$	$\text{W}/\text{m}\cdot\text{K}$	K	$\text{J}/\text{kg}\cdot\text{K}$	$\text{J}/\text{kg}\cdot\text{K}$
Value	2	1	306	70	0.3	297.5	1260	2400

Table 6.8: Initial temperature T_{feed} and mass flow rates $\dot{m}_{\cdot}$ of ASA and EtOH in the four different parameter setups.

Quantity	T_{feed}	$\dot{m}_{\text{ASA}}$	$\dot{m}_{\text{EtOH}}$
Unit	K	kg/s	kg/s
Setup 1	307.6	0.000058	0.000116
Setup 2	312.9	0.000088	0.000175
Setup 3	313.1	0.000116	0.000232
Setup 4	313.7	0.000128	0.000257

with partial mass flow rates for ASA and EtOH based on the computations of Section 6.4.2 (see Table 6.8).

Equation (6.21) is given as a one-dimensional heat balance equation in (Eder et al., 2010, p.2256), we wanted to sketch its derivation from Newton's law of cooling here. Now we can proceed to its elementary solution.

Under the assumption that all coefficients stay constant, (6.21) is a linear ordinary differential equation with constant coefficients. Given the obvious initial condition

$$T(0) = T_{\text{feed}},$$

which depends on the parameter setup (Table 6.8), one gets a unique analytic solution by the well-known variation of constant ansatz. That solution is

$$T(z) = T_{\text{amb}} + (T_{\text{feed}} - T_{\text{amb}}) \exp\left(-\frac{2\pi r_{\text{out}} h}{\dot{m}_{\text{susp}} c_{p,\text{susp}}}\right).$$

This exponentially decaying temperature profile was inserted as wall boundary condition into the mass balance equation of Subsection 6.1.6, in accordance to the one-dimensional model in Eder et al. (2010).

6.4.5 Particle inception at the inflow boundary

Three requirements must be fulfilled concerning the particle inception at the inflow boundary.

(1) The number mean and standard deviation of the diameter of the inserted particles must be as reported in Eder et al. (2010), see Table 6.1.

(2) The ASA crystal mass concentration must be $\beta_{\text{cryst, feed}} = 33.35\ {}^{\text{kg}}/_{\text{m}^3}$, as was determined in Subsection 6.4.2.

(3) As we assume a well-mixed dispersion at the inflow, the incepted particles must distribute evenly across the entire inflow boundary Γ_{in}, in an axisymmetric sense.

Let us deal with these requirements one after the other.

Inlet particle size distribution For each newly inserted particle its composition, i.e., its internal coordinate m must be determined. In the original source, mean and standard deviation of the diameter of incepted particles are given, and the authors supply histograms on the experimental inlet particle size distribution. These show distributions with one relatively sharp peak and a long tail to the right, and resemble the graph of the density functions of log-normal distribution. This, and the fact that log-normal distribution are relatively easy to calculate and compute, gives the direction: Fit a log-normal distribution to the given parameters and simulate the mass of any incepted particle as a realization of that distribution.

A small source of trouble is that our model takes mass as an inner coordinate, and our program takes the number of ASA molecules, while the original source lists mean and standard deviation of the particle *diameter*. But, by exploiting the relation of log-normal and exponential distribution, for a random variable $\mathcal{D}$ (the diameter of a ball shaped particle) and its transform $\mathcal{M} := \frac{\rho\pi}{6}D^3$ (its mass), the relation

$$\mathcal{D} \sim \text{Lognormal}\left(\mu, \sigma^2\right) \quad \Rightarrow \quad \mathcal{M} \sim \text{Lognormal}\left(3\mu + \log\left(\frac{\rho\pi}{6}\right), (3\sigma)^2\right)$$

holds. Similarly, but with a different pre-factor, $\mathcal{N}$, the number of ASA molecules, is related to $\mathcal{D}$. Fitting a log-normal random variable $\mathcal{D}$ to the data in Table 6.1 and transforming its parameters accordingly gives the parameters for the log-normal distribution $\mathcal{N}$ which are given in Table 6.9 for each experimental setup.

Inception jump rate Inception jumps differ from coagulation and growth jumps in so far as their rates depend on the velocity, that is to say on its axial component u_z. We assume that a homogeneous dispersion is fed into the crystallizer at Γ_{in}, which means that the concentration of crystalline ASA must be the same in all cells which border the inception boundary. The mass concentration of crystalline ASA relates to the inception rate λ_{incept} and the expected inception jump height $\mathbb{E}(\Delta m)$ [kg] as follows:

$$\beta_{\text{cryst, feed}} = \mathbb{E}(\Delta_{\text{mass}})\frac{\lambda_{\text{incept}}}{u_z}.$$

Let us call λ_{incept} the *surface* inception rate, since it describes the number of ASA crystals to be incepted per unit surface and time. Its unit is ${}^{1}/_{\text{m}^2\text{s}}$. Since β_{cryst} must be kept constant, the surface inception rate exhibits a linear

Table 6.9: Parameters of the log-normal probability distribution of the random variable $\mathcal{N}$ which determines the number of ASA molecules in a newly incepted crystal. For example $\mathcal{N}^{[1]} \sim \text{Lognormal}\left(\mu, \sigma^2\right)$ with $\mu = 34.83647$ and $\sigma^2 = 1.476438$. The column $E(\mathcal{N})$ holds the expected value of $\mathcal{N}$. Last column holds quotient of inception jump rate and axial velocity.

Quantity	$\mu_{\mathcal{N}}$	$\sigma^2_{\mathcal{N}}$	$E(\mathcal{N}) \cdot 10^{15}$	$\lambda_{\text{incept}}/u_z$
Unit	$[\cdot]$	$[\cdot]$	#molecules	$\#\text{jumps}/\text{cm}^3$
Setup 1	34.83647	1.476438	2.817664	39564
Setup 2	34.61936	0.8825735	1.685179	66152
Setup 3	34.98913	0.759342	2.293378	48609
Setup 4	34.7459	0.9909942	2.019033	55213

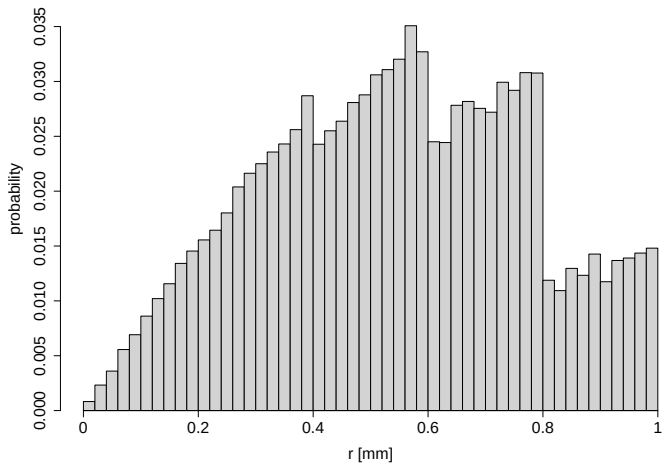

Figure 6.10: Radial component of the position of particle inceptions. Sampled over 100 seconds, Setup 4. The jumps in the plot are due to the different velocity components u_z in the five layers, which have an impact on the inception jump rate. Within each layer, the inception position probability grows linearly in r.

dependency on u_z, $\lambda_{\text{incept}} = \lambda_{\text{incept}}(u_z)$. It must be chosen in such a way that $\beta_{\text{cryst, feed}} = 33.35\ {}^{\text{kg}}/_{\text{m}^3}$ upon expectation, as was seen in Section 6.4.2. The expected mass gain $\mathbb{E}(\Delta_{\text{mass}})$ per mass gain can be computed from the expected value $\mathbb{E}(\mathcal{N})$ of numbers of ASA molecules (Table 6.9) per particle by multiplication with the molar mass M_{ASA}. Inserting these values into the above formula gives the constant values for ${}^{\lambda_{\text{incept}}}/_{u_z}$. Determining $\lambda_{\text{incept}}(\mathcal{E})$ for a certain particle ensemble $\mathcal{E}$ includes scaling with the measure of its share of the inception boundary, and multiplication with the velocity component $u_z(\mathcal{E})$. Axisymmetry must be regarded when determining the surface area of $\Gamma_{\text{in}}(\mathcal{E})$. The values of ${}^{\lambda_{\text{incept}}}/_{u_z}$, the basis of the computations, are given in Table 6.9.

Inlet particle position distribution In order to ensure a spatially homogeneous distribution of the particles across the cross section of the tube, it is necessary to take axisymmetry into account when determining the position of a newly incepted crystal. Assume a new crystal is to be incepted into ensemble $\mathcal{E}$ of cell C, whose share of the inception boundary is a straight line with starting point $A = (r_a, z_a)$ and end point $B = (r_b, z_b)$. Assume that $r_a \leq r_b$, i.e., A is closer or as close as B to the symmetry axis. A lengthy calculation, which is based on an application of the inverse sampling theorem, gives that drawing a uniform random number $u \sim \mathcal{U}([0,1])$ and setting the new position $P = (r_p, z_p)$:

$$\begin{aligned} r_p &= r_a + t(r_b - r_a) \\ z_p &= z_a + t(z_b - z_a), \end{aligned}$$

where

$$t = \frac{\sqrt{r_a^2 + u(r_b^2 - r_a^2)} - r_a}{r_b - r_a},$$

gives positions distributed in the sense of requirement (3). See Figure 6.10 for a visualization of the outcome.

7 A 3d Simulation Framework for a Fluidized Bed Crystallizer

In this chapter we want to present a simulation framework for a class of crystallizers making use of the coupled algorithm of Chapter 5. The main achievement compared to the simulations performed in the preceding chapter is the extension of the algorithm to an instationary, turbulent, full 3d setting.

Fluidized bed crystallizers ("Wirbelschichtkristaller") are crystallization devices
which are used in chemical engineering in order to grow crystal fractions into larger sizes. A photograph of a fluidized bed crystallizer is supplied as Figure 7.1. The central unit of the system is a bulgy vessel, here it has the form of an upside-down bottle of 50 cm height. The vessel is part of a closed circuit of a streaming suspension, which contains those chemical ingredients that are necessary to excite the desired crystal growth mechanisms. The flow enters at the bottom and exits through a filter at the top, entering a system of tubes and pumps, leading it back in at the bottom. The crystals themselves are held, by the filter and by choosing the inflow velocity low enough to avoid that particles reach the filter at all, inside the vessel, where they grow over time. In contrast to the tube crystallizer of the preceding chapter it is not surface attachment growth, but collision growth that is mainly responsible for the crystal size gain. Therefore, the resulting particles are aggregates, bringing along the distinct properties of crystal aggregates: greater surface, higher porosity, and in general less regular shapes than monocrystals gained by surface attachment growth have. A small amount of surface attachment growth cannot be avoided, since its main driving force, the supersaturation, is also a precondition for the formation of small solid bridges between primary particles, and thus a precondition of crystal agglomeration. A slight supersaturation is enforced by cooling, and temperature control is possible, since the crystallization vessel is surrounded by a cooling jacket and the pump and tube system are well isolated.

The main reason for particle collision growth inside a fluidized bed crystallizer is the non-laminar velocity field. It is responsible for particles following non-aligned trajectories, resulting in many collisions of particles of different sizes in various angles. A certain percentage of those collisions is "effective" in the sense that a solid bridge between two crystals gets formed: an aggregate is born. Compared to the flow tube crystallizer of the preceding chapter, both growth and aggregation rates are relatively low in the regarded fluidized bed crystallizer, which means that longer run times are needed in order to observe significant crystal growth – the residence time of the crystals in the device is longer.

The crystallization device that is in the center of attention of this chapter

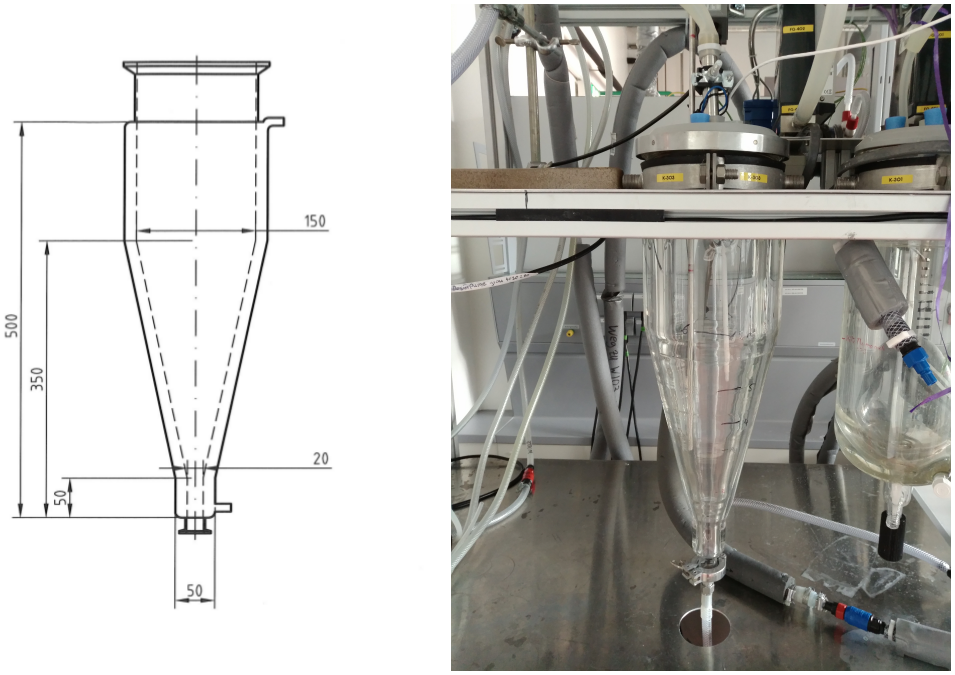

Figure 7.1: An experimental fluidized bed crystallizer, built up by the group "Lehrstuhl für Systemverfahrenstechnik" at OVGU Magdeburg. Engineering drawing and photograph are courtesy of Viktoria Wiedmeyer, 2018.

was built and operated by the group of Prof. Sundmacher at the university of Magdeburg, and is currently under experimental investigation, too. It is used for the crystallization of aluminum potassium sulfate dodecahydrate ($KAl(SO_4)_2 \cdot 12H_2O$), a substance also known by its trivial name *potash alum*. Potash alum has relatively narrow industrial use nowadays, e.g., it has medical use as an astringent in deodorants or alum blocks applied on small skin bleedings after shaving. Still it is of some interest as a model substance for crystallization, since it is inexpensive to purchase, innocuous, aggregates easily, and many of its material properties are well known. In addition it forms very shape regular octahedral crystals, which is convenient from a modeling point of view. Potash alum is soluble in water, and the solution kinetics are well known. Therefore the crystallizer is fed with a slightly oversaturated solution of potash alum in water, crystals are ideally only present in the main vessel, not in the exterior pump and tube system.

In experimental operation mode, the device works in cycles of 30 to 40 minutes, holding relatively small amounts of crystals. The total crystal mass held by the device during one experiment is of the order of several mg up to 3 g, corresponding to fractions of between 10^{-4} and $5 \cdot 10^{-2}$, of the total mass in the crystallizer. A pipette that can be positioned at variable heights in the vessel continuously pulls a small amount of suspension out of the system, towards a camera that takes pictures of the crystals (Figure 7.2). That way, after very specialized and refined image processing and data post-processing steps,

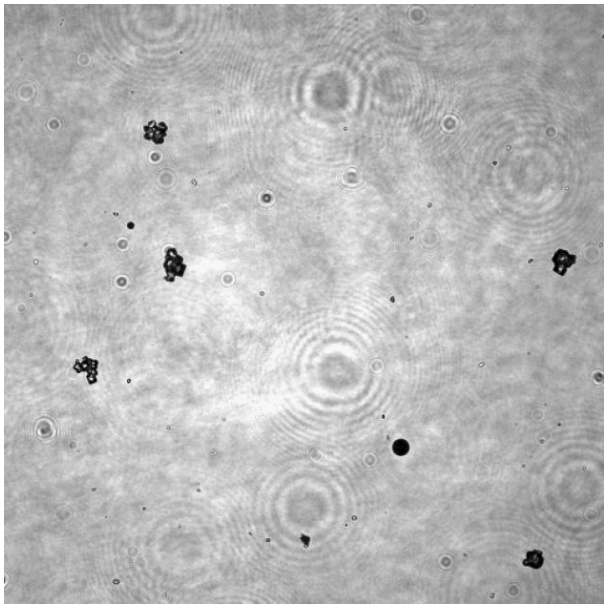

Figure 7.2: A snapshot of crystallizer suspension, containing aggregated potash alum crystals (black) and some air bubbles (round shaped black objects). Courtesy of Viktoria Wiedmeyer, 2017.

data on the development of the particle population in the crystallizer device is obtained.

The primary goal of the experimental crystallizer is to obtain particles grown to a very sharp and well defined size. A second goal is to observe and exploit a height separation of different crystal fractions, i.e., to "harvest" different, well-defined crystal fractions at different heights in the vessel.

For the simulation of the fluidized bed crystallizer we employed the coupled stochastic-deterministic approach that was developed and explained in the preceding chapters of this thesis. It is clear that a stationary, axisymmetric 2d simulation approach as was used in the case of the flow crystallizer in the preceding chapter will not lead to simulation success here. Although the geometry of the crystallizer vessel allows for an axisymmetric simulation in principle, the flow field within does not. This is, because the flow is expected to be instationary, if not even in the transition to a turbulent regime. For the simulation of turbulent flows, 2d simulations are not suitable, the step to full 3d must be undertaken. Computing a turbulent flow is a challenging task all by itself, we had to make use of specialized solvers (see Section 2.3) and of a turbulence model. To be precise, we used a Smagorinsky-type turbulence model with relatively few artificial viscosity, to get a grip on the mild turbulence of the flow. Despite the task of simulating the instationary flow in the vessel within reasonable computing time, the step to 3d brought further challenges. First, our stochastic particle simulation software Brush had to be equipped with a geometry layer that was able to represent a discretization of a three dimensional domain, and perform several specific tasks like point localization and compu-

tation of line-hull intersections. We made use of the mesh generation software TetGen (Si (2015)), which had to be slightly adapted for our purpose of mesh representation. At this place, we must thank Hang Si for his patient support on the matter.

As a second step the stochastic particle method, which had only been used in 1d and 2d settings with simple flows before, had to be equipped with boundary conditions in the particle transport step, in order to keep the particles within the simulation domain. Also, we had to refrain from the comfortable assumption that particles follow the macroscopic velocity field exactly. Keeping this assumption up would have not allowed for to capture the desired size separation effect in the model. Instead we took a first step towards a more realistic particle movement simulation by equipping the particles with a size dependent sedimentation model. This enabled us to observe a size separation effect in different layers of the simulated device. We are aware of the fact that this inclusion of particle sedimentation, and even the applied model, can only be a first step on the way to a detailed particle transport model. Particle drag and lift should be included, the results of particle inertia and friction with the surrounding, moving fluid molecules. A sedimentation model which is better suited for aggregates, taking into account their higher porosity and larger surface, should be used. Yet we think that the results that will be presented in the following capture the effect quite sufficiently, and we consider this a good motivation for working on the inclusion of more sophisticated transport models.

This chapter is organized as follows. First, in Section 7.1, we give the model of the crystallizer system in the required level of detail. Alongside each part of the model we comment on the numerical methods that were used for its solution. Section 7.2 is concerned with the presentation and discussion of numerical results, first of the flow field, then of the coupled simulation which includes the stochastic particle simulation. We end the chapter with a conclusion of the achieved in Section 7.3. An outlook on possible extensions and enhancements of the framework, and on further areas of application, is postponed to the final Chapter 8 of this thesis.

7.1 Modeling, physical and numerical details

In this section the mathematical model and all the necessary physical details for the crystallizer simulation framework will be supplied. Remember that a population balance system consists of the equations for fluid velocity and pressure $(\boldsymbol{u}, p)$ [m/s, Pa], temperature T [K] and concentration c [mol/m³]:

$$\begin{aligned}
\frac{\partial}{\partial t}\boldsymbol{u} - \frac{\mu}{\rho}\Delta\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u} + \nabla\frac{p}{\rho} &= \mathbf{g} && \text{in } (0,t_{\text{end}})\times\Omega_{\boldsymbol{x}} && [\text{m/s}^2]\\
\nabla\cdot\boldsymbol{u} &= 0 && \text{in } (0,t_{\text{end}})\times\Omega_{\boldsymbol{x}} && [1/\text{s}]\\
\frac{\partial}{\partial t}T - D_T\Delta T + \boldsymbol{u}\cdot\nabla T &= g_T I_{\text{growth}}(c,T,f) && \text{in } (0,t_{\text{end}})\times\Omega_{\boldsymbol{x}} && [\text{K/s}]\\
\frac{\partial}{\partial t}c - D_c\Delta c + \boldsymbol{u}\cdot\nabla c &= g_c I_{\text{growth}}(c,T,f) && \text{in } (0,t_{\text{end}})\times\Omega_{\boldsymbol{x}} && [\text{mol/s}]
\end{aligned} \tag{7.1}$$

as well as the population balance equation for the mass-based particle population density f [$^1/_{\mathrm{kg}\cdot\mathrm{m}^3\cdot\mathrm{s}}$] :

$$\frac{\partial}{\partial t} f + \boldsymbol{u} \cdot \nabla f = \mathcal{C}(f) + \mathcal{G}(c, T, f) \quad \text{in } (0, t_{\mathrm{end}}) \times \Omega_{\boldsymbol{x}} \times \Omega_m \quad [^1/_{\mathrm{m}^3\cdot\mathrm{kg}\cdot\mathrm{s}}].$$

All of these equations have to be equipped with suitable initial- and boundary conditions. The spatial domain $\Omega_{\boldsymbol{x}}$ is a sufficiently regular subset of $\mathbb{R}^3$ throughout this chapter, and the inner coordinate space Ω_{m} is one-dimensional and continuous, containing particle mass as the only inner coordinate. The end time t_{end} will be specified later. In order to present boundary conditions and numerical details properly in the following, we specify the spatial domain that is used for the simulations here. It comprises the inner part of the crystallization vessel shown in Figure 7.1, excluding the inflow and the outflow part where the filter is placed. In our approximation we also disregard any possible rounding at the inside, and instead assume the vessel to consist of a cylinder and a bottom-up truncated cone. The cylindrical inflow piece is not simulated for two reasons. Firstly to reduce the number of degrees of freedom in the computation, secondly in order to enforce convexity of the domain. This is necessary for the currently used particle location method, as will be explained later.

Formally, the computational domain is

$$\begin{aligned}\Omega_{\boldsymbol{x}} = &\left\{(x, y, z) \in \mathbb{R}^3 : \sqrt{x^2 + y^2} \leq 0.075, 0.3 \leq z \leq 0.45\right\} \\ &\cup \left\{(x, y, z) \in \mathbb{R}^3 : \sqrt{x^2 + y^2} \leq 0.01 + z \cdot \frac{0.065}{0.3}, 0 \leq z \leq 0.3\right\}.\end{aligned}$$

The measures in the engineering drawing in Figure 7.1 are given in mm, in the definition above we used SI units, thus m, instead. Pictures of discretized versions of the domain are supplied in Figure 7.3.

The physical constants and function expressions appearing in the equations above will be specified in the following, when examining the equations one by one. Starting with the Navier–Stokes equations, and continuing with temperature balance, mass balance and population balance equation, we will proceed to that task in the following subsections. We will often refer to certain "standard configuration" or "standard setup" of the simulations. That standard setup was chosen as a compromise between physical usefulness and computational feasibility and used for extensive numerical tests. The standard discretization of $\Omega_{\boldsymbol{x}}$ is given in Figure 7.4. It consists of 10752 tetrahedra, which are aligned in flow direction. More details on the standard setup are given in the following.

7.1.1 The velocity field

As it was before, the velocity field is itself not influenced in any way by the transported quantities, it is just one-way coupled into the system. But in contrast to the applications seen before, the velocity field is now truly instationary, even turbulent, and therefore the governing equations have to be solved and updated in each time step.

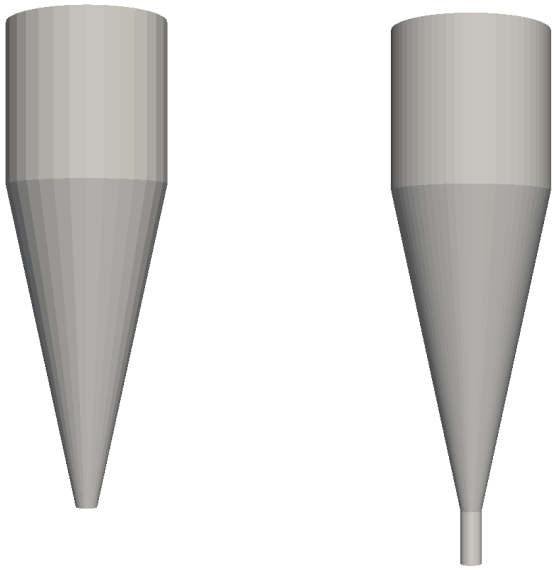

Figure 7.3: Two versions of the discretized computational geometry of the batch crystallizer. *Left*: A coarser version without inlet ($\Omega_{\boldsymbol{x}}$), this version was used as the standard setup of coupled simulations. *Right:* A finer version with inlet, it was used for parallel test computations of the velocity field.

In the Navier–Stokes equations three parameters appear. On the right-hand side, $\mathbf{g}$ is the downwards directed standard gravity, $\mathbf{g} = (0, 0, -9.80665)$ [m/s²]. The two remaining parameters in the Navier–Stokes equations, i.e., dynamic viscosity μ [kg/m·s] and fluid density ρ [kg/m³] are system-dependent. For the crystallizer they were determined to be

$$\mu = 0.0014 \text{ kg/m·s} \quad \text{and} \quad \rho = 1050 \text{ kg/m}^3$$

by the experimentalists. Both values are assumed to be constant throughout the course of the simulation, which is in our opinion well justified, since the temperature varies only about $\pm 1K$, and the crystal load is so small that its variation hardly influences the macroscopic density and/or viscosity.

The flow in the crystallizer is generated by a pump, its control variable is the mass flow per hour, $\dot{m}$ [kg/h]. That mass flow rate varies between 50 and 200 kg/h, depending on the experimental setup. In a typical, "medium" setup, with which we gathered a lot of computational experience, it is $\dot{m} = 93$ kg/h. Assuming a parabolic inflow profile at the bottom inlet, this translates into a characteristic average inflow velocity of

$$v \approx 0.08 \text{ m/s}.$$

Together with the choice of a characteristic length $L = 0.1$ m as a typical inner

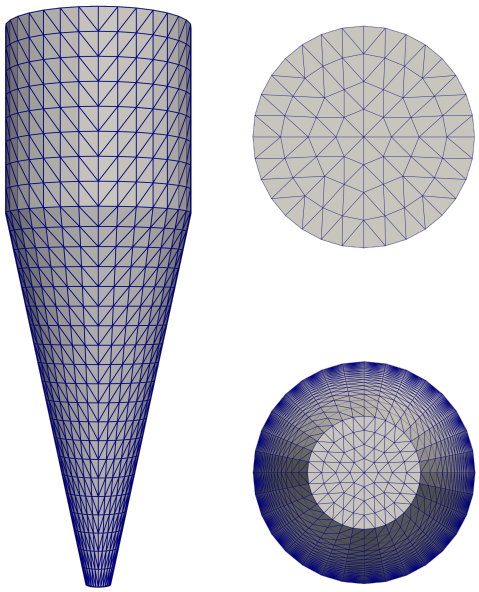

Figure 7.4: The discrete geometry of the "standard configuration". The inlet is cut off, the domain $\Omega_{\boldsymbol{x}}$ is regularly decomposed into 10752 tetrahedra of almost the same height. *Left:* Front view. *Right:* Top and bottom view.

diameter, one can compute a dimensionless Reynolds number

$$\mathrm{Re} = \frac{\rho \cdot v \cdot L}{\nu} = 6000.$$

This is well in the critical regime between laminar and turbulent behavior, and therefore a turbulent behavior may be expected with some right. Test runs of direct simulations of the Navier–Stokes equations fostered that hypothesis. Even on the finest test grid, the solver routines failed to converge at a relatively early point. Therefore we had to apply a turbulence model. We opted for a Large Eddy Simulation of Smagorinsky type. This relatively simple, but very popular and widely used turbulence model effectively introduces artificial, solution-dependent viscosity to the momentum balance equation, which then turns into

$$\frac{\partial}{\partial t}\boldsymbol{u} - \nabla \cdot (\nu + \nu_{\mathrm{Smago}} \|\nabla \boldsymbol{u}\|_F) \nabla \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} + \nabla \frac{p}{\rho} = \mathbf{g}.$$

The above is the *gradient form* of the strong formulation of the Smagorinsky momentum equation, which is its most accessible form. The dimensionless viscosity is $\nu := \frac{\mu}{\rho \cdot L_\infty \cdot v_\infty}$ with some length and velocity scale L_∞ and v_∞. Note that in the above equation also the viscous term is non-linear, which means that different solution spaces, a different time-discretized form and a different Picard iteration have to be used. As Smagorinsky-type models were already

implemented in ParMooN, we could make use of them here without digging too much into the details. An introduction to turbulence modeling, Large Eddy Simulations, and the Smagorinsky model is given in (John, 2016, Ch.8). The additional viscosity parameter ν_{Smago} is a product of two factors, the squared turbulent filter width δ^2 and a numerical Smagorinsky coefficient C_{Smago}. In our applications, δ was chosen to be $2h_k$, and the Smagorinsky coefficient was chosen in orders of magnitude between $1 \cdot 10^{-5}$ and $1 \cdot 10^{-3}$, depending on the configuration, i.e., fineness of the grid and inflow velocity. In our standard setup with $\dot{m} = 93$ kg/h, $C_{\text{Smago}} = 5 \cdot 10^{-4}$ appeared to be a good choice, which allowed to resolve the large vortices without adding overly much artificial viscosity.

Concerning solution methods, we could make good use of the knowledge layed out in Chapter 2. A Crank–Nicholson time discretization paired with a Picard iteration was used, for the standard configuration a relatively coarse, constant time step of $\Delta t = 0.05$ s showed to be suitable. As a solver for the linear problem in each step of the Picard iteration, a flexible GMRES (Saad (1993)) preconditioned with the standard Least Squares Commutator Preconditioner with iterative solver (BiCGstab with SSOR preconditioner) for the velocity subproblem was used. This combination of solvers had turned out to be a very recommendable choice for time-dependent saddle point problems in our assessment Ahmed et al. (2018). We can confirm that observation once more. For the standard setup, we restricted the number of Picard iterations to 5 and the number of FGMRES iterations per step to 10. The linear solver terminated after the initial residual was reduced by one order of magnitude (0.1), and no target residual was given. This is important for time-dependent problems, since with a fixed target residual it is possible that computations get stuck, and the discrete does not change anymore over time. Requiring residual reduction instead fixes that issue. The same is true for the inner iteration of the inexact LSC preconditioner, i.e., the BiCGstab iteration. This was required to reduce the (relatively large) initial residual by a factor 10^{-3}. The outermost iteration, the Picard iteration, was forced to be performed at least once per time step, and was set to reduce the summed residual of momentum and continuity equation below $5 \cdot 10^{-9}$. This was usually achieved within the first one or two iterations.

We must further note that we were able to produce plausible flow results only with a flow-aligned tetrahedral grid. That grid was gained in "Sandwich grid" manner, by carrying forward a 2d triangle grid of a circle to a structured 3d tetrahedral grid of a cylinder, and then stretching that grid to the desired shape of the crystallization vessel by applying a transformation. Unstructured tetrahedral grids introduced too much artificial deviation, which could not be made up with by further grid refinement due to limitations of the computing time, and on structured quad grids we were not able to find a converging solver. Therefore, the structured tetrahedral grid remained the grid of choice for the flow computations.

We assumed as initial condition a fluid at rest, i.e., $\boldsymbol{u} \equiv \mathbf{0}$ on $\Omega_{\boldsymbol{x}}$. The boundary conditions were a parabolic inflow at the bottom, and a parabolic outflow at the top, where the filter is located in the experiment. At the walls no-slip boundary conditions were used. Both inflow and outflow conditions were

adapted to the mass flow rate, such that the system stayed divergence-free. In the standard setup, this led to average inflow and outflow velocity of

$$v_{\text{avg,in}} \approx 0.0783 \text{ m/s}, \qquad v_{\text{avg,out}} \approx 0.0014 \text{ m/s}.$$

In order to avoid an impulsive start both these conditions were scaled by the time t over the first second of the simulation.

7.1.2 The temperature field

The temperature field in the crystallization vessel is subject to a convection-diffusion equation, with a source term due to particle growth on the right-hand side.

Let us first give the still unspecified quantities in the energy balance equation in System (7.1) and then proceed towards initial and boundary conditions and finally comment on the numerical procedures used for the solution of the equation. As it was in the 2d tube crystallizer example (compare Section 6.1.6), the diffusion coefficient D_T of the temperature equation is given as the quotient of thermal conductivity, suspension density and specific heat capacity of the suspension. In contrast to the ASA crystallizer model, we were provided with measurement data for the suspension, and did not have to refrain to parameters of the solvent only. The parameters are

$$\begin{aligned} \lambda_{\text{susp}} &= 0.6 \text{ W/m·K} && \text{(thermal conductivity of the suspension)} \\ \rho_{\text{susp}} &= 1050 \text{ kg/m}^3 && \text{(density of the suspension)} \\ C_{\text{susp}} &= 3841 \text{ J/kg·K} && \text{(specific heat capacity of the suspension).} \end{aligned}$$

And thus

$$D_T = \frac{\lambda_{\text{susp}}}{\rho_{\text{susp}} C_{\text{susp}}} \approx 1.5 \cdot 10^{-7} \text{ m}^2/\text{s}$$

is the diffusion coefficient. Note that, since numerical stabilization methods will add artificial diffusion anyway, it is the order of magnitude of the diffusion that is important in the computations, not so much the specific value.

On the right-hand side of the temperature equation stands a term that models temperature sources due to crystallization, i.e., surface attachment growth. It consists of the growth intensity $I_{\text{growth}}(c, T, f)$ and a positive scaling parameter

$$g_T = \frac{\Delta h_{cryst}}{\rho_{\text{susp}} C_{\text{susp}}}.$$

The crystallization enthalpy of the dodecahydrate is $\Delta h_{cryst} = 89.1$ kJ/kg, resulting in a scaling parameter $g_T = 0.0221$ K·m^3/kg. The growth intensity term is an integral over the property space Ω_m, we repeat here Equation (5.3):

$$I_{\text{growth}}(c, T, f, t, \boldsymbol{x}) = \int_{\Omega_m} G(c, T, m) f(t, \boldsymbol{x}, m) \, \mathrm{d}m. \tag{7.2}$$

The growth rate model G will be presented in the subsequent chapter on the particle size distribution.

Boundary conditions for the temperature equation were chosen rather simple for the standard setup. The experiments were either run at constant temperature or by cooling down slightly and linearly over the course of a whole experiment. The tube and pump system was well isolated, a cooling hull surrounding the main crystallization vessel was used to ensure a certain temperature profile. Since the temperature profile varied very slowly over time, we went for Dirichlet boundary conditions everywhere. In the standard setup the boundary value was a linear interpolation (in time) of the initial value $T_{start} = 288.95$ K (15.8°C) and the end temperature $T_{end} = 288.35$ K (15.2°C) at $t = 30$ min.

A question closely connected to the temperature is the solubility of the solvent in the solute, in our case potash alum dodecahydrate in water. We used a fitted, fourth-order solubility model communicated by the partner group at OVGU, and published in Temmel et al. (2016). It is formulated in terms of mass potash alum dodecahydrate per mass added water, i.e., a type of mass fraction. This decision makes perfect sense from the experimentators point of view, since it gives a recipe to produce a solution of the same concentration ("Dissolve 10 g of potash alum dodecahydrate in 1 kg of water."). At temperature $\tilde{T}$ [°C], the hydrate-per-added-water mass fraction in equilibrium is

$$w_{\mathrm{hyd},\mathrm{H_2O+}}(\tilde{T}) = a_1 + a_2\tilde{T} + a_3\tilde{T}^2 + a_4\tilde{T}^3 + a_5\tilde{T}^4 \quad \left[\frac{\text{kg hydrate}}{\text{kg added water}}\right], \quad (7.3)$$

with coefficients

$$\begin{aligned} a_1 &= 0.0506 & a_2 &= 0.0023 & a_3 &= 7.76 \cdot 10^{-5} \\ a_4 &= -2.43 \cdot 10^{-6} & a_5 &= 4.86 \cdot 10^{-8}. \end{aligned}$$

This solubility model is assured to be valid in a temperature range between 10 and 60 °C, and therefore valid for the relatively cool temperature at which the model crystallizer is operated (around 15 °C). The solubility model will enter the surface attachment growth model in Subsection 7.1.4 via the supersaturation, since, as has been stated before, the need to reduce supersaturation is the driving force behind surface attachment growth.

The temperature equation is discretized in time with the Crank–Nicholson approach, and in space using P_1 finite elements on the tetrahedral discretization. Additionally, the linear Crank–Nicholson FEM-FCT method (see Section 3.2) is used for algebraic stabilization. Since the resulting linear systems in every time step are relatively small in the standard setup a "black box" direct solver (UMFPACK) was used for their solution.

7.1.3 The concentration of dissolved potash alum

The equation for the concentration of dissolved aluminum potassium sulfate is the second convection-diffusion equation in System (7.1). It has units $^{\mathrm{mol}}/_{\mathrm{m}^3\cdot\mathrm{s}}$, where the amount of substance refers to the number of $KAl(SO_4)_2$ molecules. Potassium alum is a (dodeca)hydrate, i.e., its crystal structure incorporates 12 water molecules per unit cell. If it is dissolved, the crystal structure is

destroyed, and the formerly bonded water molecules merge into the solvent water. Therefore, the dissolved substance is an *an*hydrate and not a hydrate anymore, a difference that will be encountered again when the growth model is discussed.

In the concentration equation two constants appear: the diffusion coefficient D_c and the growth intensity scaling parameter g_c, which converts the intensity of particle attachment growth I_{growth} (see Equation (7.2)) into a loss of concentration of dissolved material. The diffusion coefficient of $KAl(SO_4)_2$ in water was reported by the group of Prof. Sundmacher to be

$$D_c = 5.4 \cdot 10^{-10} \text{ m}^2/\text{s}.$$

The growth scaling coefficient is just, see Equation (6.4),

$$g_c = -\frac{1}{M_{\text{anhydrate}}} \frac{M_{\text{anhydrate}}}{M_{\text{hydrate}}} = -\frac{1}{M_{\text{hydrate}}},$$

where the molar mass of the hydrate canceled out. The molar mass of the potash alum hydrate is

$$M_{\text{hydrate}} = 0.47438 \text{ kg/mol}. \tag{7.4}$$

In the concentration equation, the loss of free water molecules due to bonding in the hydrate crystal structure is disregarded. This assumption follows from the primary assumption of constant density.

In order to formulate the boundary conditions, let Γ_{in} denote the inlet (bottom) boundary of $\Omega_{\boldsymbol{x}}$, Γ_{out} the outlet (top) boundary and Γ_{wall} the entire wall boundary. The wall boundary is equipped with Neumann boundary conditions,

$$D_c \frac{\partial c}{\partial \mathbf{n}} = 0 \quad \text{on } \Gamma_{\text{wall}},$$

where $\mathbf{n}$ is the outward pointing normal vector. This condition implies that there is no change of concentration across the wall boundary. Together with the convective field being $\mathbf{0}$ along that boundary, this leads to the interpretation that no dissolved material passes the boundary. The same condition is posed on the outflow boundary Γ_{out}, but since there the convective field points in the direction of $\mathbf{n}$, unhindered convective transport out of the domain (in reality: through the filter) takes place.

For the inlet boundary condition we decided to implement a special, cyclic Dirichlet boundary condition. The fluidized bed batch crystallizer is operated as a closed system with regard to the dissolved species, which is a difference to the temperature. The 3d simulation comprises only the crystallizer vessel itself, the remainder of the device (tubes, filters, Y-fittings, pumps) is disregarded. Still these parts have to be accounted for in the concentration balance, since they are part of the fluid cycle. We pursue the following idea to model the impact of this peripheral devices on the mass balance.

Let Δt be the constant time step size and t_{cyc} the "cycle time", i.e., the average time it takes for the fluid to travel one round through the entire device.

Assume that the quotient $t_{\mathrm{cyc}}/\Delta t$ is a natural number n_{cyc} – it is the number of discrete time steps until the "fluid cycle" starts from the beginning. A list $\mathbf{c}_{\mathrm{in,cyc}} = \left(c_{\mathrm{in},0}, ..., c_{\mathrm{in},n_{\mathrm{cyc}}-1}\right)$ $[\mathrm{mol}/\mathrm{m}^3]$ of inlet concentrations is initialized, we call it the "inlet cycle". On startup, all values in that list are set to some constant initial concentration c_{start}. The values in the inlet cycle are chosen as inlet boundary values one after the other, starting from the beginning once the end is reached. This means that

$$c(t_{\mathrm{start}} + m\Delta t, \cdot) = c_{\mathrm{in},(m \bmod n_{\mathrm{cyc}})} \quad \text{on } \Gamma_{\mathrm{in}} \tag{7.5}$$

is the inlet boundary condition in the m-th discrete time step. The "consumption" of dissolved species due to attachment growth is taken into account by reducing a corresponding value in the inlet cycle at each time step. For that purpose, the amount of consumed species in some time step must be determined, and it must be converted to a reduction of the corresponding inlet value, which then receives an update in form of a subtraction.

Let $\mathbf{n}$ be the inward pointing normalized normal of Γ_{in}, $\boldsymbol{u}$ is the velocity field, which is parabolic in space and constant in time, with maximum value $u_{\max}$ at the inlet. For an arbitrary time step m the inlet concentration $c_{\mathrm{in},m}$ from the inlet cycle is "active". The total inflowing amount of substance during this time step is:

$$C_{\mathrm{in},m} = \Delta t \int_{\Gamma_{\mathrm{in}}} c_{\mathrm{in},m} \boldsymbol{u} \cdot \mathrm{d}\vec{\sigma} = \Delta t \; c_{\mathrm{in},m} \int_{\Gamma_{\mathrm{in}}} \boldsymbol{u} \cdot \mathbf{n} \; \mathrm{d}s \tag{7.6}$$

$$= \Delta t \; c_{\mathrm{in},m} \int_0^R \int_0^{2\pi} u_{\max} \left(1 - \left(\frac{r}{R}\right)^2\right) r \; \mathrm{d}\varphi \mathrm{d}r \tag{7.7}$$

$$= \Delta t \; c_{\mathrm{in},m} 2\pi u_{\max} \frac{R^2}{4}, \tag{7.8}$$

with the inlet radius $R = 0.01$ m.

If $m > n_{\mathrm{cyc}-1}$, i.e., at least the first cycle has been finished, then $c_{\mathrm{in},m}$ must be such that the amount of substance inflowing at time step m, $C_{\mathrm{in},m}$ must take into account the amount $C_{\mathrm{lost},m-n_{\mathrm{cyc}}}$, which was lost at time step $m - n_{\mathrm{cyc}}$. The simple equation reads

$$C_{\mathrm{in},m} = C_{\mathrm{in},m-n_{\mathrm{cyc}}} - C_{\mathrm{lost},m-n_{\mathrm{cyc}}}. \tag{7.9}$$

The loss term $C_{\mathrm{lost},m-n_{\mathrm{cyc}}}$ is just the spatial integral over the right-hand side of the concentration balance equation, multiplied with Δt. Combining equations (7.6) and (7.9) we get an equation for $\Delta c = c_{\mathrm{in},m-n_{\mathrm{cyc}}} - c_{\mathrm{in},m}$:

$$\Delta c = \frac{C_{\mathrm{lost},m-n_{\mathrm{cyc}}}}{2\pi u_{\max} \frac{R^2}{4}}.$$

By implementing such cyclic inlet boundary conditions as described above, a one- dimensional mass balance of the peripheral devices appears in the PBS. Note that this is based on a certain mixing assumption: in the periphery, the

fluid mixes so well that all the concentration sinks of one time step in the main vessel manifest themselves in the inflow of a *single* time step, t_{cyc} seconds later.

In the standard setup the initial condition $c_{\text{start}} = 207\ {}^{\text{mol}}/_{\text{m}^3}$ was chosen. This value corresponds to the saturation concentration at 17 °C. The cycle time was chosen as $t_{\text{cyc}} = 60$ s. The numerical methods for the concentration equation were those that were used for the temperature equation, see the preceding Subsection 7.1.2.

7.1.4 The population balance of potash alum dodecahydrate crystals

The numerical solution of the population balance equation for the potash alum dodecahydrate crystals falls in the responsibility of the stochastic particle simulation (SPS) program Brush. The particle model which we use is a most simple spherical model. In theory, the model is mass based, the inner coordinate $m \in \Omega_m$ describes particle mass in kg. In the program, the model is based on number of hydrate molecules per particle. The conversion between both worlds is done via Avogadro's constant N_A and the molar mass of potash alum dodecahydrate, see Equation (7.4). Despite the simple particle model several difficulties had to be overcome, especially due to the 3d geometry, and several modeling decisions had to be made. This subsection is split into four paragraphs, each of them dealing with one part of the model. Those paragraphs are *Convection and reflection*, *Sedimentation*, *Coagulation* and *Growth model*.

Convection and reflection In the SPS, particle transport and particle interaction are separated by a splitting scheme. For the 3d simulation we used a second order Strang splitting scheme that was already implemented in Brush and is documented in Celnik et al. (2007). The main assumption that particles follow the macroscopic velocity field inertialess[1] is kept up, but gets extended and in some sense broken by two more mechanisms: particle wall reflection and particle sedimentation (which is postponed to the next paragraph).

In order to represent a 3d geometry, a mesh representation backend had to be incorporated in Brush. After a failed attempt with the 3d Triangulations package of CGAL (Jamin et al. (2018)), which was admittedly not designed for the purpose of mesh representation, we decided to use a modified version of Tetgen (Si (2015)) as a mesh representation. The necessary tasks

- representation of a user defined tetrahedral mesh
- computation of cell volumes
- point location
- computation of hull intersection points

could be realized within that framework. Especially the first point was critical, since our flow computations required a grid aligned mesh (see Section 7.1.1),

[1] Except for the numerical inertia that gets introduced by the first order discretization of the velocity field.

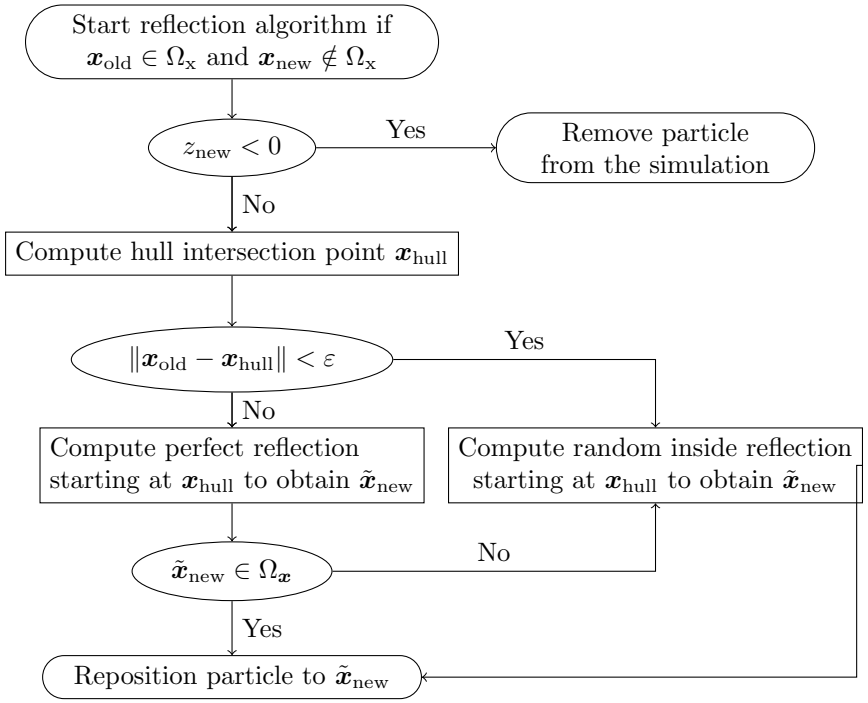

Figure 7.5: The particle reflection algorithm that was implemented in Brush. The former position of the particle is $\boldsymbol{x}_{\text{old}}$, the new (outside) position is $\boldsymbol{x}_{\text{new}}$, with z-component z_{new}. A small $\varepsilon > 0$ is given as wall vicinity tolerance.

and a certain relationship of the grids of CFD and SPS is a necessity for an efficient coupled algorithm (see Section 6.2.1).

Point location became necessary after each transport splitting step. After each such step, it must be determined in which cell a particle is contained now. For that purpose, Tetgen has a directed search algorithm implemented. For that, a convex domain is a premise. In our case that was achieved by cutting off the inflow, as is shown in Figure 7.4.

This point location is also able to detect if a particle has left the computational domain. If that case occurred, a decision had to be made, whether the particle left through the inflow boundary Γ_{in} or "left" through the solid or filter boundary $\Gamma_{\text{wall}} \cup \Gamma_{\text{out}}$. In the first case, the particle was measured and removed from the simulation. In the other case, a particle reflection was executed in order to reposition the particle inside the domain. The "ideal" particle reflection algorithm which we implemented is given in Figure 7.5.

Two different reflection mechanisms are incorporated in that algorithm: perfect reflection and random reflection, biased towards the inside of the domain. Both model elastic wall collisions, i.e., no kinetic energy is absorbed in the collision. Perfect reflection is performed when the start point $\boldsymbol{x}_{\text{old}}$ of the particle movement is well inside the domain (distance from the hull intersection point is greater than some ε). Then, the new position $\tilde{\boldsymbol{x}}_{\text{end}}$ of the particle is computed

in a two stage process. The reflection direction $\mathbf{e}$ is

$$\mathbf{e} = (\boldsymbol{x}_{\text{hull}} - \boldsymbol{x}_{\text{old}}) - 2\left((\boldsymbol{x}_{\text{hull}} - \boldsymbol{x}_{\text{old}}) \cdot \mathbf{n}\right)\mathbf{n},$$

where $\mathbf{n}$ is the outward pointing unit normal of that tetrahedra face where $\boldsymbol{x}_{\text{hull}}$ is located, and the norm of the reflection vector is set to be

$$\|\tilde{\boldsymbol{x}}_{\text{new}} - \boldsymbol{x}_{\text{hull}}\| = \|\boldsymbol{x}_{\text{new}} - \boldsymbol{x}_{\text{hull}}\|.$$

Random reflection is performed whenever either the starting point of a reflection is very close to the wall, or in case a double reflection would occur. Both these cases are very prone to robustness errors, therefore we decided to handle them in this safer manner. Random reflection is performed with an acceptance-rejection scheme. Directions $\mathbf{e}$ on the "inward" half sphere of radius $r := \|\boldsymbol{x}_{\text{new}} - \boldsymbol{x}_{\text{hull}}\|$ are created by drawing two random numbers $u_1, u_2 \sim \mathscr{U}[0,1]$ and setting

$$\mathbf{e} = r\left(\sqrt{1-z^2}\cos(\theta), \sqrt{1-z^2}\sin(\theta), z\right),$$

where

$$\theta = 2\pi u_1 \text{ and } z = -1 + 2u_2.$$

If the scalar product $\mathbf{e} \cdot \mathbf{n}$ is positive, the direction $\mathbf{e}$ is inverted, in order to point inwards. Further, a check is performed whether $\boldsymbol{x}_{\text{hull}} + \mathbf{e}$ is inside the domain. If so, that value is accepted and the particle moved there, otherwise, $\mathbf{e}$ is rejected and another random reflection direction is generated.

The random reflection algorithm is a mean of avoiding reflection in cases where serious robustness errors due to floating point arithmetic are to be expected. In fact there are more such cases, and several of them are just handled by throwing exceptions and relocating particles to cell centers in the current version of the program. In each computing run, the occurrence of such cases is logged, and it turned out that around 10^{-5} % of all occurring reflections fall into that category, therefore the issues did not get tracked down yet. Two sources of remaining robustness problems come to mind – the first is that the computational domain $\Omega_{\boldsymbol{x}}$, though convex in theory, is not necessarily convex in floating point representation. Therefore point location via directed search is not guaranteed to give correct results. The second point are rounding errors that occur when computing the hull intersection of lines that lie very close and/or almost parallel to the hull.

Sedimentation As was mentioned in the introduction a desired effect of the potash alum fluidized bed crystallizer is height separation of different crystal size fractions. To reflect such a behavior in the simulation, the model must be extended by some mechanism which exerts this feature. For that aim a simple sedimentation model for spherical particles was implemented in Brush. The model derivation and formulation can be found in (Berg, 1983, pp.58).

There, sedimentation of particles is modeled via a sedimentation rate, which results in a downward sinking velocity of particles and differs for different particle sizes. In order to apply the sedimentation rate that was proposed in Berg (1983), the following assumptions must be made:

- the particles are spherical
- the fluid velocity around the particles is so low that Stokes' law is a good enough approximation for the inverse particle mobility (frictional drag coefficient).

If those statements can be assumed at least approximately, the above model can be considered an approximation to start with. Its derivation starts from a formula for the sinking velocity $\boldsymbol{v}_s$ of a particle which is surrounded by moving fluid molecules:

$$\boldsymbol{v}_s = \frac{\mathbf{F}_{\text{down}}}{f}. \tag{7.10}$$

Here $\mathbf{F}_{\text{down}}$ [N] is the force acting downwards, and f [kg/s] is the frictional drag coefficient, which relates force and velocity. The frictional drag coefficient is related to the particle mobility μ of the Einstein–Smoluchowski relation by $\mu = \frac{1}{f}$. Specifying the forces acting downwards in (7.10), those are

$$\|\mathbf{F}_{\text{down}}\| = (m - V\rho)g.$$

In this equation, m is the particle mass and V the particle volume, $\rho = 1050\ \text{kg/m}^3$ is the density of the displaced fluid, the standard gravity $g = 9.81\ \text{m/s}^2$ is known. As can be seen, the force depends linearly on the mass difference of particle and displaced fluid. It remains to find a model for f, here is where Stokes' law for spherical particles in a viscous fluid at low velocity comes into play, and thus the two assumptions made above. According to that model (see (Berg, 1983, pp.58) for its detailed derivation) f can be approximated as

$$f = 3\pi\mu d,$$

with d being the diameter of the spherical particle, and the dynamic fluid viscosity μ, which is known to be $\mu = 0.0014\ \text{kg/m·s}$ for the model crystallizer.

Inserting everything into (7.10) and replacing $m = V\rho_{\text{cryst}}$ (where $\rho_{\text{cryst}} = 1760\ \text{kg/m}^3$) one finally obtains the following equation for the additional, downwards directed velocity component:

$$v_z = d^2 \cdot \frac{(\rho_{\text{cryst}} - \rho_{\text{fluid}})g}{18\mu}.$$

For the sake of rigorousness, here is the mass-based formulation:

$$v_z = \frac{\left(\frac{6}{\rho\pi}\right)^{\frac{2}{3}}(\rho_{\text{cryst}} - \rho_{\text{fluid}})g}{18\mu} m^{\frac{2}{3}}.$$

Numerical tests with this model showed that the sedimentation rate introduced that way was far too high (see Section 7.2). Especially large particles that entered the slow flow regimes at the sides of the inlet could not be moved upwards against the gravity anymore. Depending on the boundary conditions at the inlet (particle reflection or particle removal) this led to either excessive

coagulation in the areas close to the inflow or to a significant loss of particles through the inlet. In some test simulations up to 80 % of the total crystal mass was lost over time that way.

As an ad-hoc remedy, we introduced a sedimentation scaling parameter $\sigma \in [0, 1]$ with which the sedimentation effect could be down-scaled. With this, better results without or at least with less mass loss could be achieved. In the actual experiment the octahedral shape and aggregate nature of the potash alum crystals lead to a greater particle surface and porosity, which suggests that the actual sedimentation velocity is less than for spherical particles. Therefore, a different modeling approach could be fruitful here. The results achieved with the down-scaled spherical sedimentation model, which already show the desired height separation, can be taken as a motivation for incorporating a more refined sedimentation model.

Coagulation Collision growth, the dominant growth mechanism, is modeled with an integral term as known from the Smoluchowski coagulation equation. The Brownian coagulation kernel that will be used is temperature-dependent. That dependence on T is indicated in the following formulation of the coagulation integral:

$$\mathcal{C}(f, T, t, \boldsymbol{x}, m) = \frac{1}{2} \int_{\Omega_m} K_{\text{Brownian}}(T, m-\mu, \mu) f(t, \boldsymbol{x}, m-\mu) f(t, \boldsymbol{x}, \mu) \, \mathrm{d}\mu \\ - \int_{\Omega_m} K_{\text{Brownian}}(T, m, \mu) f(t, \boldsymbol{x}, m) f(t, \boldsymbol{x}, \mu) \, \mathrm{d}\mu.$$

The choice of the coagulation kernel is usually a delicate matter. In a first attempt of this framework we go for the Brownian kernel, since among the well-established literature kernels it favors coagulation of small with large particles over coagulation of equally sized particles and therefore guarantees relatively uniform growth. With a dimensionless coagulation scaling parameter κ, the mass-based kernel reads

$$K(T, m_1, m_2) = \kappa \frac{2 T k_B}{3\mu} \left(\frac{1}{d(m_1)} + \frac{1}{d(m_2)} \right) (d(m_1) + d(m_2)) \quad [\mathrm{m}^3/\mathrm{s}] \,.$$

In this equation $d(m)$ is the diameter of a spherical potash alum crystal of mass m, $d(m) = \sqrt[3]{6m/\rho_{\text{cryst}}\pi}$. Further, the Boltzmann constant k_B [J/K] appears, and μ denotes once more the dynamic viscosity of the surrounding fluid.

The particle interaction is subject to the stochastic particle simulation. Coagulation events take place as jumps in a Markov jump process, as is described in Chapter 4. In particular, a stochastic weighted algorithm as described in Subsection 4.3.3 is employed. The scaling parameter κ must, as usual, be fitted by means of numerical experiments.

Growth model For the implementation of a surface attachment growth model we could refer to very recent experimental results. In Temmel et al. (2016) an experimentally validated growth model is given. It is an Arrhenius-type equation, see also the corresponding paragraph in Subsection 6.1.4, where a

similar model was used for the growth of ASA crystals. The potash alum growth model is

$$G_d = \frac{\sqrt{2}}{\pi^{\frac{1}{3}}} k_{\mathrm{G}_1} \exp\left(-\frac{k_{\mathrm{G}_2}}{RT}\right) (S_{\mathrm{hyd,H_2O+}} - 1)^{k_{\mathrm{G}_3}} \quad [\mathrm{m/s}] \tag{7.11}$$

with fitted parameters

$$\begin{aligned} k_{\mathrm{G}_1} &= 5 \cdot 10^7 \ \mathrm{m/s}, \\ k_{\mathrm{G}_2} &= 75 \cdot 10^3 \ \mathrm{J/mol}, \\ k_{\mathrm{G}_3} &= 1.4. \end{aligned}$$

The prefactor $\sqrt{2}/\pi^{\frac{1}{3}}$ is due to conversion from an octahedral to a spherical particle model. The growth rate G_d is only computed for $S_{\mathrm{hyd,H_2O+}} > 1$, otherwise it is set to 0. The quantity $S_{\mathrm{hyd,H_2O+}}$ [kg/kg] is the relative supersaturation of the solution. In the original reference the mass of dissolved hydrate per added solvent (water) is used as a concentration measure. In order to use it in the simulations, conversions have to be made. Let $w_{\mathrm{hyd,H_2O+}}$ [kg/kg] be the current concentration in that measure and $w^{\mathrm{eq}}_{\mathrm{hyd,H_2O+}}(T)$ [kg/kg] the concentration in equilibrium, then the current (super)saturation is

$$S_{\mathrm{hyd,H_2O+}} = \frac{w_{\mathrm{hyd,H_2O+}}}{w^{\mathrm{eq}}_{\mathrm{hyd,H_2O+}}(T)}. \tag{7.12}$$

While the equilibrium concentration is given by the solubility curve (7.3) which originates from Temmel et al. (2016), too, the current concentration $w_{\mathrm{hyd,H_2O+}}$ must be gained by conversion from "our" molar concentration c, which is in units mol anhydrate per m^3 solution. First, display $w_{\mathrm{hyd,H_2O+}}$ as a function of the concentration measure $w_{\mathrm{anhyd,H_2O}}$ [kg/kg] (mass anhydrate per total mass water), which is closer to our viewpoint:

$$w_{\mathrm{hyd,H_2O+}} = \frac{w_{\mathrm{anhyd,H_2O}}}{w_{\mathrm{anhyd,hyd}} - w_{\mathrm{anhyd,H_2O}}}. \tag{7.13}$$

Here, $w_{\mathrm{anhyd,H_2O}} = M_{\mathrm{anhydrate}}/M_{\mathrm{hydrate}}$ is the constant mass fraction of anhydrate in the dodecahydrate crystal. Proceed with

$$w_{\mathrm{anhyd,H_2O}} = \frac{c\ M_{\mathrm{anhydrate}}}{\rho_{\mathrm{solution}}}. \tag{7.14}$$

Inserting (7.14) into (7.13) and that into (7.12) gives a growth rate term in dependence on T [K] and c [mol/m³], which is in accordance to our formulation of the model system. This growth rate can be inserted into the term given in (5.9), and into the growth intensity integrals on the right-hand sides of concentration and temperature equation.

Concerning numerics, the growth term can do without a scaling parameter. In the SPS, an LPDA approach (see Section 4.3.4) is used in order to reduce the computational effort, while maintaining the order of convergence. Still, in the regarded temperature regime, relative supersaturation and thus the growth rate are relatively low, collision growth dominates (both in terms of computational effort and in terms of effect on the system) in the simulations.

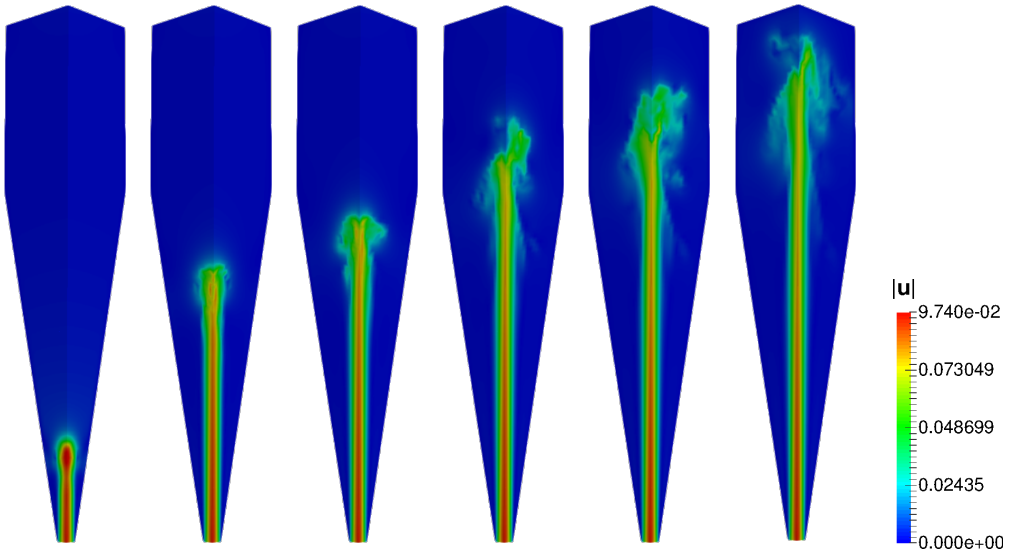

Figure 7.6: The development of the flow field in the startup phase, 56 $^{kg}/_{h}$ mass flow setup. Snapshots at times 1,5,7,10,12 and 15 s, unit is $^{m}/_{s}$.

7.2 Results of the 3d simulations

The 3d crystallizer computations were mainly performed for the "standard setup", which was set up to represent typical working conditions of the fluidized bed crystallizer reported by the group of Prof. Sundmacher. The simulated time was 1800 s (30 min), where in the first 30 s the flow field developed (see Figure 7.6). Particles were inserted between seconds 30 and 40. Though in the experiments particles were funneled into the device all at once, we simulated insertion uniformly in the entire domain and stretched over the interval $[30\text{ s}, 40\text{ s}]$, since the SPS profits from a relatively uniform spatial distribution of computational particles. Nevertheless, supporting simulations with insertion at just one place were performed, too, in order to visualize the spreading of particles in the device.

In the standard setup, 100 mg of crystalline material, divided equally between two particle size fractions were added. The first fraction comprised 50 mg particles of diameter 75 μm, either monodisperse, or following a log-normal distribution of 25 μm standard deviation. The second fraction contained particles of 125 μm diameter, again either monodisperse or log-normal with the same standard deviation. Although the theory is formulated in terms of particle mass, we chose particle sphere equivalent diameter in μm for the following presentation, since this quantity seems to be a more common particle description, making the results of this chapter better comparable to similar works.

The geometry was the standard discretization of Ω_x into 10752 tetrahedra, see Figure 7.4 and the description in Section 7.1. To each discretization cell an ensemble of maximum 256 computational particles was assigned. The conversion from computational to physical particles in the SPS is mediated by the number of physical particles which a full ensemble represents. In preliminary

Table 7.1: Details on the flow-only computations, 300 seconds simulated time. Time-step size Δt, Smagorinsky coefficient C_{Smago}, number of MPI processes used in the computation, and total computing time (wall time).

Setup	$\dot{m}$ (kg/h)	Δt (s)	C_{Smago}	# proc.	Comp. time (s)
Level 2	56	$5 \cdot 10^{-2}$	$1 \cdot 10^{-4}$	4	5050
	93	$5 \cdot 10^{-2}$	$1 \cdot 10^{-4}$	8	3600
Level 3	56	$5 \cdot 10^{-2}$	$1 \cdot 10^{-4}$	8	25,700
	93	$5 \cdot 10^{-2}$	$1 \cdot 10^{-4}$	8	36,920
Level 4	56	$2 \cdot 10^{-2}$	$1 \cdot 10^{-4}$	16	380,000
	93	$2 \cdot 10^{-2}$	$1 \cdot 10^{-4}$	16	411,230

simulation runs, $5.0 \cdot 10^8$ # physical particles/m^3 was determined to be a useful value for a full ensemble here. Since due to sedimentation physical particles amass at the bottom of the device, we found it necessary to multiply that value by 10 below 10 cm and by 100 below 5 cm in order to avoid permanent overfilling of ensembles close to the bottom. This setup led to a total number of roughly 150000 computational particles in the simulation domain after completion of the insertion at 40 s. That number is typically reduced by around 50% at the end of a computing run due to sedimentation loss through the inlet and overfilling/rescaling of ensembles.

Concerning the velocity field, the simulations in the standard setup were complemented by grid refinement studies. Two different mass flow setups were investigated both for the flow-only and the complete system. These were 56 kg/h, leading to a maximum inflow velocity of 0.094 m/s, and 93 kg/h, i.e., 0.161 m/s maximum inflow velocity. Both setups led to mildly turbulent flows. In the flow-only computations 300 s were simulated, on three different refinement levels of the same initial tetrahedra grid. The dimensions are:

- Level 2 (*standard setup*): 10,752 Tetrahedra, P_2/P_1 Galerkin discretization of the Navier–Stokes equations resulted in 51768 degrees of freedom.
- Level 3: 86,016 tetrahedra, 385,644 d.o.f.
- Level 4: 688,128 tetrahedra, 2,974,932 d.o.f.

For these runs we could make use of the distributed memory parallelism of ParMooN, and used the parallelized version of FGMRES and the LSC preconditioner that was described in Section 2.4.4. The number of processes used in the individual runs and the computing time for the 300 s numerical simulation can be found in Table 7.1. There also the time step and the Smagorinsky parameter is given. While the latter was chosen the same for all runs, the constant time step had to be reduced somewhat for the finest grid computations, in order to ensure convergence of the iterative solver.

The computational grids and snapshot results of the computed velocity solution are given in Figure 7.7 in terms of the norm of the discrete velocity solution.

All figures are vertical cuts through the three-dimensional computing domain. A distinct feature of all flow computations is the relatively stable primary flow area in the middle, stretching from inflow (bottom) to outflow (top). At the top, where the main flow hits the filter, vortices start to develop. These are much more pronounced in the setup with higher mass flow rate. While in the 56 $^{\mathrm{kg}}/_{\mathrm{h}}$ setup the vortices mostly varied in size over time, in the 93 $^{\mathrm{kg}}/_{\mathrm{h}}$ setup vortex shedding could be observed. Occasionally those vortices got strong enough to disturb the main flow in the center, which then rebuilt similar to the beginning. While the vortices could be resolved rather clearly on the Level 4 grid, the results with the coarsest grid are, as could be expected, rather smeared. Yet since the main features of the flow, i.e., central stream and vortices near the filter, could be preserved in the whole, this geometry was used for those runs which included particles and transport equations. This was mainly a concession to simulation time. Still, the results on the grid refinement are a clear statement for distributed-memory parallelization of the combined algorithm, since even for the Level-2 setup a notable gain in speed could be obtained with a parallelized solver for the velocity system.

Simulations for the entire population balance system were performed in sequential only. A typical run of 30 minutes simulated time took around 35 hours computing time on a single core, where the majority of the time (around 75 %) was spent in the flow computation.

The particle kinetics introduced in Section 7.1 left two dimensionless parameters unspecified: the sedimentation scaling parameter σ and the coagulation scaling parameter κ. While κ can be used in order to fit simulation results to experimental results (see, e.g., the 2d simulations in Chapter 6 for that procedure), σ has to be treated differently. That parameter was initially introduced, because the full sedimentation model of Section 7.1 resulted in a loss of large particles through the inflow that was so severe that the effects of coagulation and surface attachment growth were not at all visible in the development of the PSDs anymore. As soon as a particle grew above some threshold, it quickly left the simulation domain. For quantification of that effect we performed runs with a fixed, small coagulation parameter ($\kappa = 500$), and measured the development of the total mass in the system, in dependence on the sedimentation parameter. Surface attachment growth was disabled for these computations, since the mass transfer from dissolved to crystalline phase would have shadowed the mass loss through the inlet. The results of these no-growth runs are plotted in Figure 7.8. One sees clearly that even for a sedimentation parameter of $\sigma = 0.25$ the mass loss is considerable, around ten percent during the first twenty minutes. For $\sigma = 0.1$ almost all mass was kept in the system. Therefore all subsequent computations are performed with this sedimentation parameter. Even for this heavily reduced sedimentation a layering of particle size fractions in the crystallizer could be observed, as will be documented in the following.

With the sedimentation parameter fixed as such to $\sigma = 0.1$, runs with different coagulation parameters could be performed, comparing the development of the particle size distribution and/or its averaged quantities over the course of the simulation for different coagulation parameters. The main goal was to examine the dependence of the PSD on the height in the device, where the particle

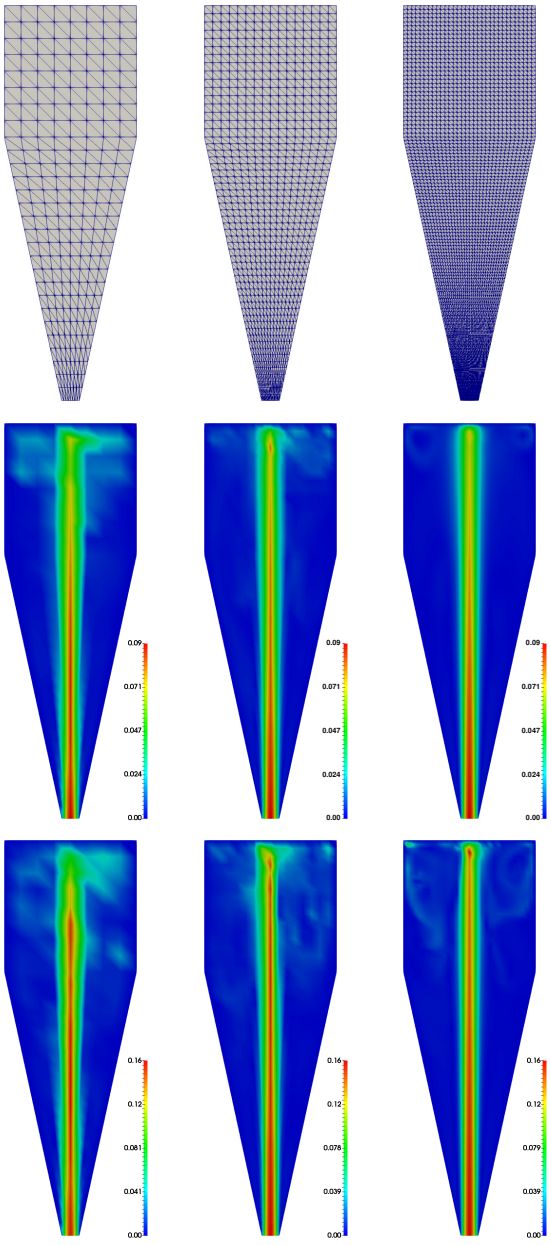

Figure 7.7: Flow simulation results ($\|\boldsymbol{u}\|$) for three different grid refinement levels, vertical cut plane. *Top row:* Computational grids, left to right: refinement level 2, level 3, and level 4. *Middle row:* Solution snapshots at 200 s on the respective grid, mass flow rate 56 $^{\mathrm{kg}}/_{\mathrm{h}}$. Units are $^{\mathrm{m}}/_{\mathrm{s}}$. *Bottom row:* Solution snapshots at 200 s on the respective grid, mass flow rate 93 $^{\mathrm{kg}}/_{\mathrm{h}}$.

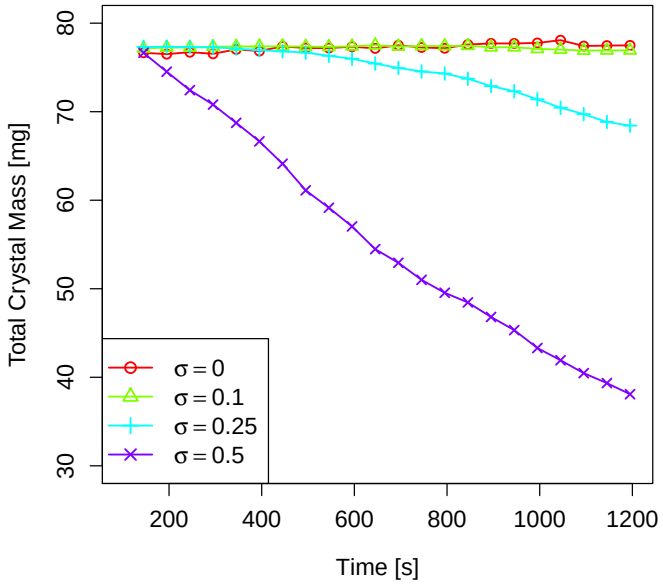

Figure 7.8: Total crystal mass in the system during coagulation-only simulations with different sedimentation parameters σ. Significant mass loss occurs for $\sigma = 0.5$. For $\sigma = 0.1$ mass loss is almost 0. Start mass was 77 mg, mass flow rate 56 $^{\mathrm{kg}}/_{\mathrm{h}}$, coagulation parameter $\kappa = 500$.

sampling was performed. Simulation data was gained by printing out snapshots of all computational particles present in the simulation domain every 10 seconds. This interval was large enough to ensure a complete exchange of particles in each ensemble, thus reducing stochastic dependence, and allowing for averaging over multiple of these snapshots without too much redundancy. This way, between 2 and 3 GB of data were collected per run. Additionally, all particles that left the device through the inflow were logged. Different post processing steps could be performed on those data.

Figures 7.9 and 7.10 show spatially summed, resp. averaged, quantities. In Figure 7.9, the development of the total particle mass is shown for both flow setups. A gain due to surface attachment growth is visible, as is mass loss through the inlet. The latter is the more pronounced the greater the coagulation parameter is, i.e., the larger particles exist. This is obvious, since larger particles sediment faster and therefore leave the simulation domain more often, taking with them a considerable part of the total mass. Note also that mass gain due to particle growth is higher in the slower flowing setup. From Figure 7.10, one learns that the difference in the development of the spatial average particle diameter is less distinct between the two flow setups. The particles in the slower setup grow only slightly larger than those in the faster setup upon average. As was expected, a higher coagulation parameter results in larger particles in total.

Figure 7.11 itemizes the development of the average particle diameter accord-

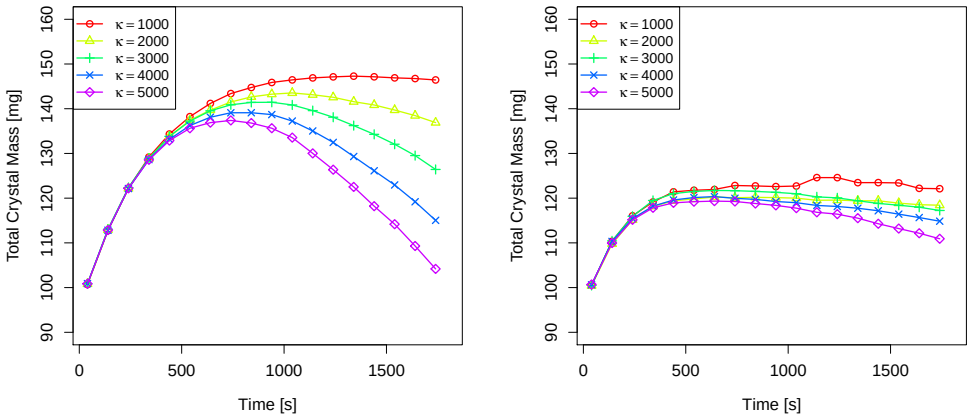

Figure 7.9: Development of the total crystal mass in the standard setup, depending on coagulation scaling parameter κ. Sedimentation parameter is fixed at $\sigma = 0.1$. Higher coagulation rates result in larger particles, which are more likely to slip out through the inflow due to sedimentation. *Left:* 56 $^{\mathrm{kg}}/_{\mathrm{h}}$ mass flow setup. *Right:* 93 $^{\mathrm{kg}}/_{\mathrm{h}}$ mass flow setup.

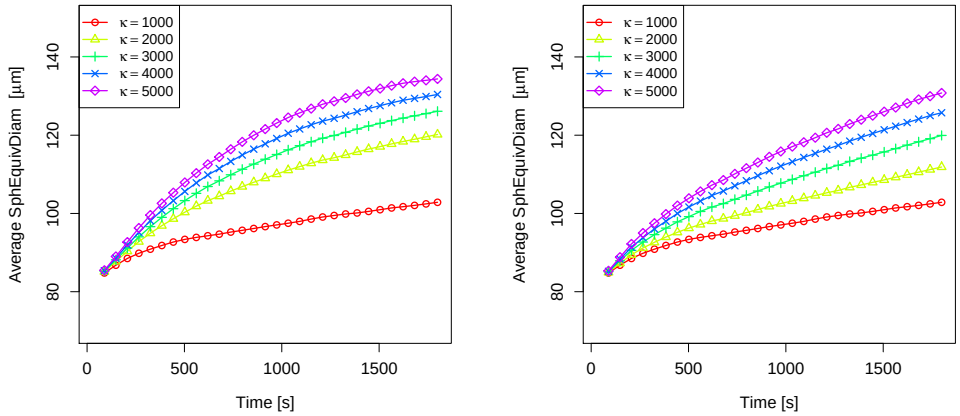

Figure 7.10: Temporal development of the spatially averaged sphere equivalent particle size diameter in the crystallizer device. *Left:* 56 $^{\mathrm{kg}}/_{\mathrm{h}}$ mass flow setup. *Right:* 93 $^{\mathrm{kg}}/_{\mathrm{h}}$ mass flow setup.

ing to the sample height (z-coordinate) in the crystallizer, for both setups. The data was collected by sampling over all computational particles which were located in a certain height interval $[z - 0.025, z + 0.025]$ of 5 cm width in a certain time interval. This procedure led to sample sizes of around 20,000 computational particles per time interval in the lower regions and of up to 100,000 computational particles per time interval in the higher regions of the device. This difference in sample size is the main reason for the different variances

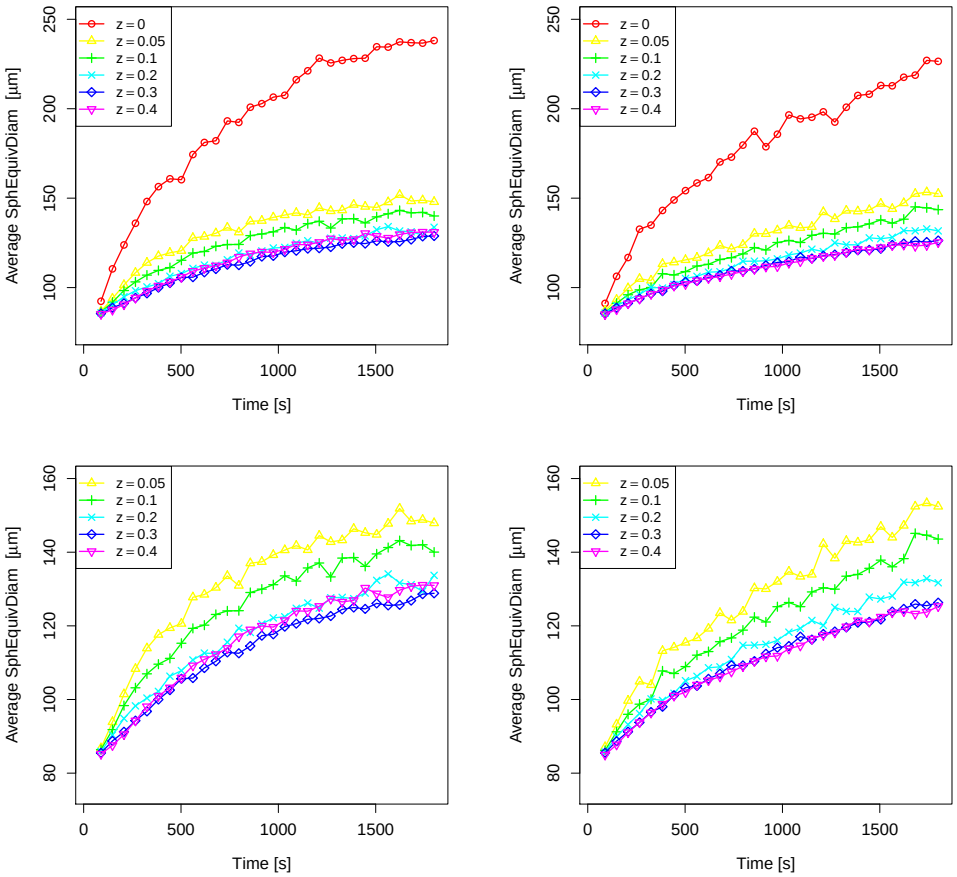

Figure 7.11: Temporal development of the spatially averaged sphere equivalent particle size diameter in different heights in the crystallizer device. Height z is given in m above the inlet. Bottom row figures show details of the top row figures (without the particles directly above the inlet). Coagulation parameter is $\kappa = 5000$ and sedimentation parameter is $\sigma = 0.1$. *Left:* 56 $^{\mathrm{kg}}/_{\mathrm{h}}$ mass flow setup. *Right:* 93 $^{\mathrm{kg}}/_{\mathrm{h}}$ mass flow setup.

visible in the plotted curves. The figures show that the difference in average particle size is the most distinct in the bottom of the crystallizer. While the curves for heights 0, 0.05, 0.1 and 0.2 move within clearly distinguishable regimes, there is hardly a difference between the curves for heights 0.3 and 0.4. In fact, it even seems, as if the particles at height 0.4 are slightly larger than those at 0.3, especially in the 56 $^{\mathrm{kg}}/_{\mathrm{h}}$ setup. A possible reasons for this behavior is as follows. Particles have a longer residence time in the vicinity of the filter, where they get reflected repeatedly and travel slowly to the sides of the device. This high residence time could lead to more coagulations taking place in those cells close to the top boundary, which is visible in averaging.

Figure 7.12 shows not just averaged quantities, but the development of the

entire particle size distribution for both setups at different heights in the form of histograms. Computational particles were organized in bins according to their diameter, the length of the bar for a certain particle class gives the probability of a random physical particle to fall into that class, according to the computed distribution. No temporal averaging has been performed here, the histograms represent particle populations at the given particular point in time. The sample heights were chosen such that, according to the observations of Figure 7.11, one could expect a perceptible shift in the PSD. This expectation was fulfilled, it is clearly visible from the figure that in vicinity of the inlet ($z = 0$) a relatively wide-spread population of large particle aggregates could be encountered, while 10 and 30 cm away from the inlet, smaller particles that had experienced less collision and surface growth events resided.

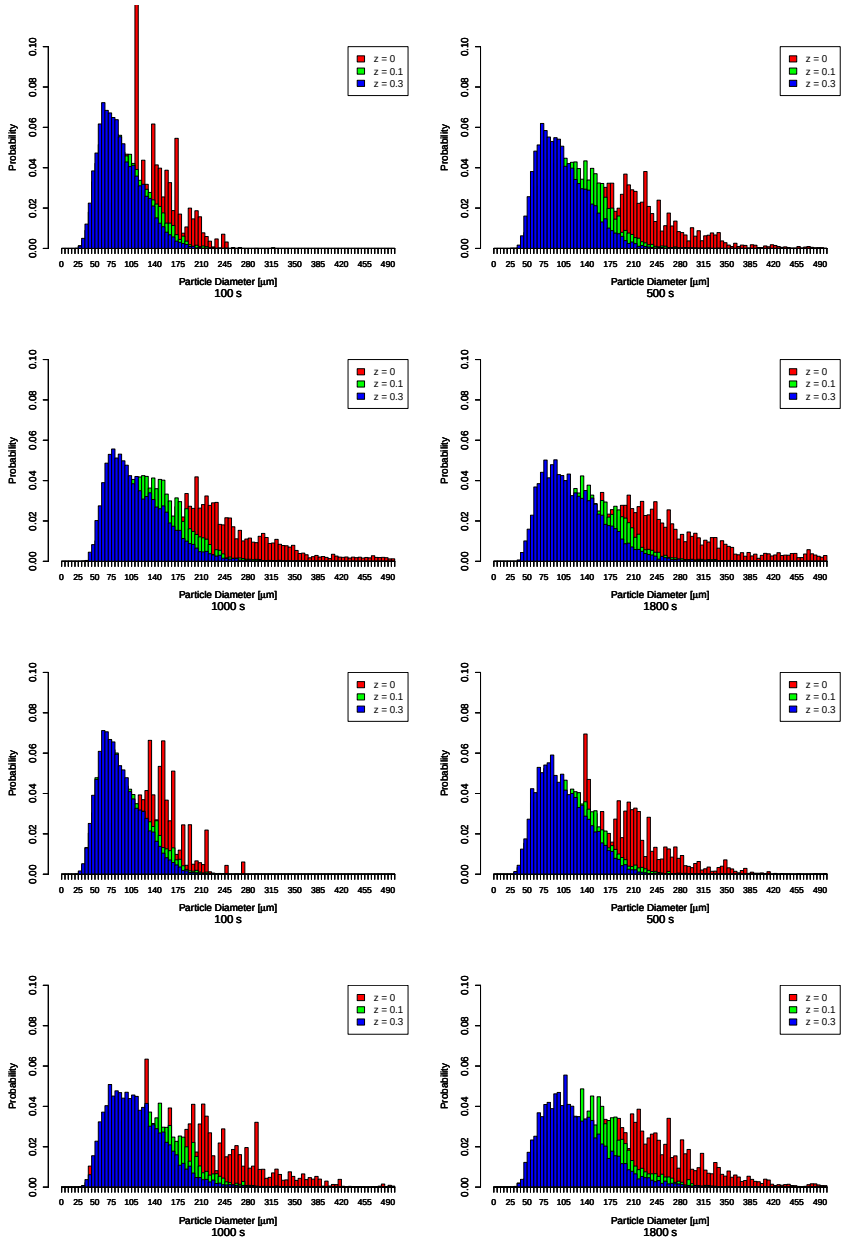

Figure 7.12: Snapshots of the PSD at different points in time and in different heights in the crystallizer. All figures were gained with coagulation parameter $\kappa = 5000$ and sedimentation parameter $\sigma = 0.1$. The initial distribution is the sum of two log-normal distributions. *First and second row:* 56 $^{\mathrm{kg}}/_{\mathrm{h}}$ mass flow setup. *Third and fourth row:* 93 $^{\mathrm{kg}}/_{\mathrm{h}}$ mass flow setup.

7.3 Summary of the findings

The model and simulation presented in this chapter have the characteristics of a simulation framework. Just as the modeled fluidized bed crystallizer itself is operated under different experimental conditions in order to find a configuration which gives the desired product particle distribution, the framework can be extended, fitted and applied to different application scenarios. The numerical results that were presented in this chapter were a demonstration that the coupling algorithm of stochastic particle simulation and deterministic flow solver that was successfully applied in a 2d setting before could be transferred to 3d, giving plausible results in manageable computing time, and opening up possibilities for further extensions which are out of scope for non-stochastic methods.

Therefore we consider it a success that on top of the combination of SPS and a turbulent flow computation, particle boundary reflection conditions and a particle sedimentation model could be built into the simulation framework. With those extensions we were able to carve out a dependence of the average particle size on the sample height in the crystallizer. This result is the most notable, and it is lined by the more basic result: A successful simulation of particle collision growth and surface attachment growth, based on experimental results, modeling decisions and empirical choices of numerical parameters, in full 3d. Further supporting results are the simulation of a turbulent flow in a non-standard geometry, using findings of Ahmed et al. (2018) on suitable solvers for time-dependent flow problems and a distributed-memory parallelization of the LSC preconditioner that was applied for these computations for the first time.

Already in the presented state, the simulation framework is suited to be used for different setups of the crystallization experiment. Examples of typical changes to the setup are the initial amount of particles and their distribution in physical and coordinate space, the inlet temperature and the cooling strategy at the wall, the total experimenting time, or the total mass flow. Concerning the mass flow, we found in numerical experiments that for a mass flow of 200 $\mathrm{kg/h}$ our standard grid is not suitable anymore, and no convergence of the solvers could be achieved. There, grid refinement is definitely necessary, possibly joined by a more sophisticated turbulence model.

Several changes and refinements of the model and the simulation itself come to mind, but since these are rather general in nature they will be the subject of the subsequent chapter, which contains a conclusion of the findings of this entire thesis, and an outlook on further enhancements and research directions.

8 Conclusion and Outlook

8.1 Conclusion

This thesis was concerned with the computer simulation of population balance systems. In particular, a new simulation method that coupled advanced deterministic finite element methods to a stochastic particle simulation of Kinetic Monte Carlo type was developed and applied. This process included a detailed formulation of the coupled system in question, which was suitable for treatment with the coupled method, and a presentation of the applied splitting scheme. In general, two types of coupling were present in the model: the first was a one-way coupling of the velocity field into the other equations, the second a coupling of the transported quantities (particle population balance, species concentration, system temperature) due to particle surface attachment growth. The second coupling type pointed to a distinct area of application, i.e., chemical engineering and in particular crystal growth in a moving fluid environment.

The coupled method was therefore applied to two model systems from chemical engineering. The first system was a tube crystallizer for aspirin. It was operated at relatively low fluid velocity, and therefore could be modeled with a simplifying 2d axisymmetric approach, crossing out several difficulties of the method which might have been encountered in a 3d setting. Simulations with which experimental data from Eder et al. (2010) could be reproduced efficiently and numerically robustly were performed, including particle collision growth with a simple coagulation kernel. Thus the coupled method could prove its practical applicability. In a second project, a fully instationary 3d simulation with a turbulent flow field was conducted. This necessitated several adjustments to the stochastic particle method, concerning the geometry representation and particle reflection boundary conditions. Additionally, a first step towards a more refined particle movement simulation was undertaken by including a simple particle sedimentation model. With those building blocks a fluidized bed crystallizer for the model substance aluminum potassium sulfate dodecahydrate (potash alum) could be simulated, including flow field, concentration of dissolved potash alum, system temperature, and population balance of the dodecahydrate crystals. Here no quantitative validation with experimental results was undertaken, yet some qualitative features, like crystal growth in general, and especially a particle size layering in the device could be reproduced.

Since for the simulation of those population balance systems basically three types of equations had to be solved with specialized methods, several results on these methods were gained “en passant”. For the Navier–Stokes equations, results of an assessment of fast linear solvers were presented, and the findings entered the 3d simulation of the fluidized bed crystallizer in form of the application of the Least Squares Commutator preconditioner for the time-dependent

flow computations. Further, a classical domain decompositioning method was presented in a new form, and used for the parallelization of the same preconditioner, finally applied for subsidiary flow computations for the potash alum crystallizer. Concerning convection-diffusion-reaction equations, for those a non-standard algebraic stabilization method was applied in 2d and 3d, hopefully underlining the usefulness and feasibility of the approach to a wider public. Finally, the stochastic particle simulation method was extended to axisymmetric 2d and to full 3d, where in its original framework it had only been applied to spatially homogeneous or 1d problems.

All in all, a new coupled stochastic-deterministic method for population balance systems in 2d and 3d has been proposed, implemented, and successfully tested numerically.

8.2 Outlook

This thesis is no exception to the general rule of thumb that answering one research question will give rise to ten others at least. That said, this last section can be regarded as a to-do list of tasks, which could be performed next, based on the achievements of this thesis.

The most urgent directions of method enhancement are in our opinion the extension to multidimensional particle spaces and the synchronization of the parallelization of the computer algorithms for stochastic and deterministic methods.

Let us first dwell on the latter point. Currently ParMooN, the CFD code that we used, profits from the implementation of a domain decompositioning method, with which the entire process from (uniform) grid refinement, discretization, matrix assembling, system solution and postprocessing can be performed by a number of CPUs at once. The stochastic particle solver Brush on the other hand relies on a shared-memory parallelization of the main loop. Currently, the interface layer between both programs does only operate in sequential runs, i.e., in coupled simulations only one processor can be used. Since in 3d the flow computations took the majority of the computing time, an implementation of an interface that allows for data transfer between distributed-memory CFD and shared-memory SPS would mean a severe improvement. In the long term, a distributed-memory parallelization of the SPS should be the goal, since this usually scales better to a large number of processors, and is a precondition for the application of the coupled method in HPC.

When it comes to higher dimensional particle models, the SPS is a very natural framework due to the presence of computational particles that mirror real-world particles. Computational particles can contain further information that comes with the greater number of internal coordinates way easier than non-stochastic models, which rely on averaged particle properties. Beyond particle mass, inner dimensions of interest could be a second or third parameter for the geometry (needle shaped, elliptical, cuboid particles,...), the number of particles contained in an aggregate, or further substances. The main challenge here is to define a sensible addition operation on the inner coordinate space, which represents coagulation, and likewise a coagulation kernel K, with a majorant

that allows to keep computational effort under control. In this thesis, only one-dimensional particle models were regarded, since our main goal was to extend the number of *spatial* coordinates in the system. An extension to more *property* coordinates should be the next step.

Further extensions come to mind. One option is to include further particle processes like nucleation, breakage, or dissolution into the PBE. All of these processes appeared in the PBE or SPS literature before, so adding them to model and simulation should be at relatively low cost. Another possibility, briefly touched earlier in this thesis, is the incorporation of fluid-fluid interactions (chemical reactions) into the model. From the CFD point of view this will require taking measurements that prevent reaction instabilities (see Section 3.3).

Regarding particle movement, the implementation of a different sedimentation model comes to mind, a model which is closer to the physical reality of crystal aggregates, and can do without a numerical scaling parameter. This issue suggests a further possibility for an extension of the method. A more sophisticated particle movement model should also include particle drag and lift forces. This is relatively straightforward in the stochastic framework, which allows to tag each particle with its current velocity and velocity gradient by including them into its state space, and then computing drag and lift velocity component on that basis – for a starting point we refer to the particle transport models in Berg (1983). One aspect of particle sedimentation could be built in at relatively low effort, namely a dependence of the sedimentation rate on the number of primary particles forming an aggregate, a number which can easily be tracked with the SPS. Aggregates have typically a greater surface than primary particles of the same mass, therefore one can expect those to sink slower than primary particles. Modeling and incorporating such a dependency could already give interesting results.

So far, no backcoupling of the particles to the fluid velocity, for example by letting the density vary, is included. Introducing any such approach will lead to abandoning the "quasi-homogeneous" assumption that the classical PBE model relies on, but it offers interesting opportunities for an extension of the method.

A question as old as coagulation modeling itself is that of the right choice of the coagulation kernel. The coagulation kernel is usually rooted in assumptions on particle transport. The constant kernel used in Chapter 6 and the Brownian kernel of Chapter 7 are probably not the ideal choices for the respective crystallizer setups. Instead, at least in the fluidized bed crystallizer setup, the transition regime kernel, which is suitable for the transition regime between laminar and turbulent flow, could be given a try, or even a discrete kernel gained from a kernel estimation procedure.

A final question has not been raised in this thesis, which focused on modeling and scientific computing, yet. From a mathematicians point of view, the time has come to underpin the computational results with some numerical analysis. This task will not only require breaking down the model system to a more handy form, but also choosing a suitable mathematical framework, which brings together analysis and stochastics theoretically, just as this thesis did in a practical way.

Bibliography

Naveed Ahmed, Gunar Matthies, and Lutz Tobiska. Finite element methods of an operator splitting applied to population balance equations. *J. Comput. Appl. Math.*, 236(6):1604–1621, 2011.

Naveed Ahmed, Clemens Bartsch, Volker John, and Ulrich Wilbrandt. An assessment of some solvers for saddle point problems emerging from the incompressible Navier—Stokes equations. *Comp. Meth. Appl. Mech. Eng.*, 331: 492–513, 2018.

David J. Aldous. Deterministic and stochastic models for coalescence (aggregation and coagulation): A review of the mean-field theory for probabilists. *Bernoulli*, 5(1):3–48, 1999.

Patrick R. Amestoy, Iain S. Duff, Jean-Yves L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.*, 23(1):15–41, 2001.

Felix Anker, Sashikumaar Ganesan, Volker John, and Ellen Schmeyer. A comparative study of a direct discretization and an operator-splitting solver for population balance systems. *Comput. Chem. Eng.*, 75:95–104, 2015.

Søren Asmussen and Peter W. Glynn. *Stochastic simulation: Algorithms and analysis.* New York, NY: Springer, 2007.

Ivo Babuška. Error-bounds for finite element method. *Numer. Math.*, 16:322–333, 1970/1971.

Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. `http://www.mcs.anl.gov/petsc`, 2017.

Gabriel R. Barrenechea, Volker John, and Petr Knobloch. Some analytical results for an algebraic flux correction scheme for a steady convection-diffusion equation in one dimension. *IMA J. Numer. Anal.*, 35(4):1729–1756, 2015.

Gabriel R. Barrenechea, Volker John, and Petr Knobloch. Analysis of algebraic flux correction schemes. *SIAM J. Numer. Anal.*, 54(4):2427–2451, 2016.

Georg Barthelmes, Sotiris. E. Pratsinis, and Hans Buggisch. Particle size distributions and viscosity of suspensions undergoing shear-induced coagulation and fragmentation. *Chem. Eng. Sci.*, 58(13):2893–2902, 2003.

Nathanael Berestycki. Recent progress in coalescent theory. *Ensaios Matematicos*, 6:1–193, 2009.

Howard C. Berg. *Random walks in biology.* Princeton University Press, Princeton, NJ, 1983.

S. A. Berger, L. Talbot, and L. S. Yao. Flow in curved pipes. *Annu. Rev. Fluid Mech.*, 15(1):461–512, 1983.

Christine Bernardi, Monique Dauge, and Yvon Maday. *Spectral methods for axisymmetric domains*, volume 3 of *Series in Applied Mathematics (Paris).* Gauthier-Villars, Éditions Scientifiques et Médicales Elsevier, Paris; North-Holland, Amsterdam, 1999.

Maximillian O. Besenhard, Roland Hohl, Aden Hodzic, Rafael J. P. Eder, and Johannes G. Khinast. Modeling a seeded continuous crystallizer for the production of active pharmaceutical ingredients. *Cryst. Res. Technol.*, 49(2-3): 92–108, 2014.

Graeme A. Bird. Direct simulation and the Boltzmann equation. *Phys. Fluids*, 13(11):2676, 1970.

Dietrich Braess. *Finite Elemente. Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie.* Berlin: Springer, 2nd, rev. edition, 1997.

Leo Breiman. *Probability.* Number 7 in Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

Franco Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers. *Rev. Française Automat. Informat. Recherche Opérationnelle Sér. Rouge*, 8(R-2):129–151, 1974.

Alexander N. Brooks and Thomas J. R. Hughes. Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations. *Comput. Methods Appl. Mech. Eng.*, 32:199–259, 1982.

Hans Burchard, Eric Deleersnijder, and Andreas Meister. A high-order conservative Patankar-type discretisation for stiff systems of production–destruction equations. *Appl. Numer. Math.*, 47(1):1–30, 2003.

Matthew Celnik, Robert I. A. Patterson, Markus Kraft, and Wolfgang Wagner. Coupling a stochastic soot population balance to gas-phase chemistry using operator splitting. *Combustion and Flame*, 41(2):158–176, 2007.

Mark H. A. Davis. *Markov Models and Optimization.* Number 49 in Monographs on Statistics and Applied Probability. Chapman & Hall, London & New York, 1st edition, 1993.

Madalina Deaconu, Nicolas Fournier, and Etienne Tanré. A pure jump Markov process associated with Smoluchowski's coagulation equation. *Ann. Probab.*, 30(4):1763–1796, 2002.

William B. Dean. Fluid motion in a curved channel. *Proc. R. Soc. Lond., Ser. A*, 121:402–420, 1928.

Robert E. L. DeVille, Nicole Riemer, and Matthew West. Weighted flow algorithms (WFA) for stochastic particle coagulation. *J. Comput. Phys.*, 230 (23):8427–8451, 2011.

R. L. Drake. A general mathematical survey of the coagulation equation. In G.M. Hidy and J. R. Brock, editors, *Topics in Current Aerosol Research (pt. 2)*, pages 201–376. Pergamon Press, New York, 1972.

Rafael J. P. Eder, Stefan Radl, Elisabeth Schmitt, Sabine Innerhofer, Markus Maier, Heidrun Gruber-Woelfler, and Johannes G. Khinast. Continuously seeded, continuously operated tubular crystallizer for the production of active pharmaceutical ingredients. *Crystal Growth & Design*, 10(5):2247–2257, 2010.

Rafael J. P. Eder, Elisabeth K. Schmitt, Julia Grill, Stefan Radl, Heidrun Gruber-Woelfler, and Johannes G. Khinast. Seed loading effects on the mean crystal size of acetylsalicylic acid in a continuous-flow crystallization device. *Cryst. Res. Technol.*, 46(3):227–237, 2011.

Rafael J. P. Eder, Simone Schrank, Maximilian O. Besenhard, Eva Roblegg, Heidrun Gruber-Woelfler, and Johannes G. Khinast. Continuous sonocrystallization of acetylsalicylic acid (ASA): Control of crystal size. *Crystal Growth & Design*, 12(10):4733–4738, 2012.

Andreas Eibeck and Wolfgang Wagner. Stochastic particle approximations for Smoluchoski's coagulation equation. *Ann. Appl. Probab*, 11(4):1137–1165, 2001.

Howard C. Elman and Ray S. Tuminaro. Boundary conditions in approximate commutator preconditioners for the Navier-Stokes equations. *Electron. Trans. Numer. Anal.*, 35:257–280, 2009.

Howard C. Elman, David J. Silvester, and Andrew J. Wathen. *Finite Elements and Fast Iterative Solvers. With Applications in Incompressible Fluid Dynamics*. Oxford University Press, New York, 2005.

Howard C. Elman, Victoria E. Howle, John Shadid, David Silvester, and Ray S. Tuminaro. Least squares preconditioners for stabilized discretizations of the Navier–Stokes equations. *SIAM J. Sci. Comput.*, 30(1):290–311, 2007.

Radek Erban, S. Jonathan Chapman, and Philip K. Maini. A practical guide to stochastic simulations of reaction-diffusion processes. arXiv preprint, 2007. URL `http://arxiv.org/abs/0704.1908`.

Stewart N. Ethier and Thomas G. Kurtz, editors. *Markov Processes*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, 1986.

Joel H. Ferziger and Milovan Perić. *Computational Methods for Fluid Dynamics.* Springer, Berlin, 3rd rev. edition, 2002.

Luca Formaggia and Anna Scotti. Positivity and conservation properties of some integration schemes for mass action kinetics. *SIAM J. Numer. Anal.*, 49(3):1267–1288, 2011.

Nicolas Fournier and Jean-Sébastien Giet. Convergence of the Marcus–Lushnikov process. *Methodol. Comput. Appl.*, 6(2):219–231, 2004.

Giovanni P. Galdi. *An Introduction to the Mathematical Theory of the Navier-Stokes Equations. Steady-state Problems.* Springer Monographs in Mathematics. Springer, New York, 2nd edition, 2011.

Sashikumaar Ganesan and Lutz Tobiska. An accurate finite element scheme with moving meshes for computing 3D-axisymmetric interface flows. *Internat. J. Numer. Methods Fluids*, 57(2):119–138, 2008.

Sashikumaar Ganesan, Volker John, Gunar Matthies, Raviteja Meesala, Shamim Abdus, and Ulrich Wilbrandt. An object oriented parallel finite element scheme for computations of pdes: Design and implementation. In *23rd IEEE International Conference on High Performance Computing Workshops, HiPC 2016 Workshops, Hyderabad, India, December 19-22, 2016*, pages 106–115, 2016.

Daniel T. Gillespie. The stochastic coalescence model for cloud droplet growth. *J. Atmospheric Sci.*, 29(8):1496–1510, 1972.

Daniel T. Gillespie. An exact method for numerically simulating the stochastic coalescence process in a cloud. *J. Atmospheric Sci.*, 32(10):1977–1989, 1975.

Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.*, 22(4):403–434, 1976.

Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

Vivette Girault and Pierre-Arnaud Raviart. *Finite Element Methods for Navier-Stokes Equations*, volume 5 of *Springer Series in Computational Mathematics.* Springer, Berlin, 1986.

Michael Goodson and Markus Kraft. An efficient stochastic algorithm for simulating nano-particle dynamics. *J. Comput. Phys.*, 183(1):210–232, 2002.

Shay Gueron. The steady-state distributions of coagulation-fragmentation processes. *J. Math. Biol.*, 37(1):1–27, 1998.

Flavius Guiaş. A Monte Carlo approach to the Smoluchowski equations. *Monte Carlo Methods and Applications*, 3(4), 1997.

Wolfgang Hackbusch. *Multi-Grid Methods and Applications*. Number 4 in Springer Series in Computational Mathematics. Springer, Berlin, 2nd edition, 2003.

Wolfgang Hackbusch, Volker John, Aram Khachatryan, and Carina Suciu. A numerical method for the simulation of an aggregation-driven population balance system. *nt. J. Numer. Methods Fluids*, 69(10):1646–1660, 2012.

Günther Hämmerlin. Die Stabilität der Strömung in einem gekrümmten Kanal. *Arch. Ration. Mech. Anal.*, 1:212–224, 1958.

P.S. Harrison. Pipelines. Thermopedia entry, Begell House Inc., 2010. URL `http://www.thermopedia.com/content/1031/`.

E. M. Hendriks, J. L. Spouge, M. Eibl, and M. Schreckenberg. Exact solutions for random coagulation processes. *Zeitschrift für Physik B Condensed Matter*, 58(3):219–227, 1985.

Ching-Mao Hung. Definition of contravariant velocity components. NASA Technical Report 20020079883, NASA Ames Research Center; Moffett Field, CA, 2002.

Ross Ingram. A new linearly extrapolated Crank–Nicolson time-stepping scheme for the Navier–Stokes equations. *Math. Comp.*, 82(284):1953–1973, 2013.

Stephanie Jacquot. A historical law of large numbers for the Marcus-Lushnikov process. *Electron. J. Probab.*, 15:605–635, 2010.

Clément Jamin, Sylvain Pion, and Monique Teillaud. 3D triangulations. In *CGAL User and Reference Manual, version 4.11.1*. CGAL Editorial Board, 2018.

Intae Jeon. Existence of gelling solutions for coagulation-fragmentation equations. *Commun. Math. Phys.*, 194(3):541–567, 1998.

Volker John. On the efficiency of linearization schemes and coupled multigrid methods in the simulation of a 3D flow around a cylinder. *Internat. J. Numer. Methods Fluids*, 50(7):845–862, 2006.

Volker John. *Finite Element Methods for Incompressible Flow Problems*, volume 526 of *Springer Series in Computational Mathematics*. Springer, Berlin, 2016.

Volker John and Gunar Matthies. MooNMD—a program package based on mapped finite element methods. *Comput. Vis. Sci.*, 6(2-3):163–169, 2004.

Volker John and Michael Roland. Simulations of 3D/4D precipitation processes in a turbulent flow field. In *Numerical mathematics and advanced applications 2009. Proceedings of ENUMATH 2009, the 8th European conference on numerical mathematics and advanced applications, Uppsala, Sweden, June 29–July 3, 2009.*, pages 479–487. Springer, Berlin, 2010.

Volker John and Ellen Schmeyer. Finite element methods for time-dependent convection-diffusion-reaction equations with small diffusion. *Comput. Methods Appl. Mech. Eng.*, 198(3-4):475–494, 2008.

Volker John, Petr Knobloch, Gunar Matthies, and Lutz Tobiska. Non-nested multi-level solvers for finite element discretisations of mixed problems. *Computing*, 68(4):313–341, 2002.

George Karypis and Vipin Kumar. Metis – unstructured graph partitioning and sparse matrix ordering system, version 2.0. Technical report, University of Minnesota, 1995.

Anastasia Kolodko and Karl Sabelfeld. Stochastic particle methods for Smoluchowski coagulation equation: Variance reduction and error estimations. *Monte Carlo Methods Appl.*, 9(4):315–339, 2003.

Alois Kufner. *Weighted Sobolev spaces*, volume 31 of *Teubner Texts in Mathematics*. Teubner, Leipzig, 1980.

Thomas G. Kurtz. The relationship between stochastic and deterministic models for chemical reactions. *J. Chem. Phys.*, 57(7):2976, 1972.

Dmitri Kuzmin. On the design of algebraic flux correction schemes for quadratic finite elements. *J. Comput. Appl. Math.*, 218(1):79–87, 2008.

Dmitri Kuzmin. Explicit and implicit FEM-FCT algorithms with flux linearization. *J. Comput. Phys.*, 228(7):2517–2534, 2009.

Dmitri Kuzmin and Matthias Möller. Algebraic flux correction. I: Scalar conservation laws. In *Flux-corrected transport. Principles, algorithms, and applications. Papers based on the workshop 'High-resolution schemes for convection-dominated flows: 30 years of FCT'. With foreword by Jay P. Boris.*, pages 155–206. Springer, Berlin, 2005.

Philippe Laurençot and Stéphane Mischler. On coalescence equations and related models. In Nicola Bellomo, Pierre Degond, Lorenzo Pareschi, and Giovanni Russo, editors, *Modeling and Computational Methods for Kinetic Equations*, pages 321–356. Birkhäuser, Boston, MA, 2004.

William J. Layton. *Introduction to the Numerical Analysis of Incompressible Viscous Flows*. Number 6 in Computational Science and Engineering Series. Society for Industrial and Applied Mathematics, Philadelphia, 2008.

Kok Foong Lee, Robert I. A. Patterson, Wolfgang Wagner, and Markus Kraft. Stochastic weighted particle methods for population balance equations with coagulation, fragmentation and spatial inhomogeneity. *J. Comput. Phys.*, 303:1–18, 2015.

Christian Lindenberg, Martin Krättli, Jeroen Cornel, Marco Mazzotti, and Jörg Brozio. Design and optimization of a combined cooling/antisolvent crystallization process. *Crystal Growth & Design*, 9(2):1124–1136, 2009.

Shuyuan Liu and Tat Leung Chan. A coupled CFD-Monte Carlo method for simulating complex aerosol dynamics in turbulent flows. *Aerosol Sci. Technol.*, 51(3):269–281, 2017.

Alexei A. Lushnikov. Certain new aspects of the coagulation theory. *Izv. Akad. Nauk SSSR Ser. Fiz. Atmosfer. i Okeana*, 14:738–743, 1978a.

Alexei A. Lushnikov. Coagulation in finite systems. *J. Colloid Interface Sci.*, 65(2):276–285, 1978b.

Sibusiso Mabuza, Dmitri Kuzmin, Sunčica Čanić, and Martina Bukač. A conservative, positivity preserving scheme for reactive solute transport problems in moving domains. *J. Comput. Phys.*, 276:563–595, 2014.

Daniele L. Marchisio, Jesse T. Pikturna, Rodney O. Fox, R. Dennis Vigil, and Antonello A. Barresi. Quadrature method of moments for population-balance equations. *AIChE Journal*, 49(5):1266–1276, 2003.

Allan H. Marcus. Stochastic coalescence. *Technometrics*, 10(1):133, 1968.

Andreas Meister and Sigrun Ortleb. On unconditionally positive implicit time integration for the DG scheme applied to shallow water flows. *Internat. J. Numer. Methods Fluids*, 76(2):69–94, 2014.

Benjamin Menküc and Ralf Wiechmann. Herleitung der Divergenz in Zylinderkoordinaten ausgehend von kartesischen Koordinaten, 2004. URL `http://www.menkuec.de/benjamin/files/-divergenz_herleitung_zylinder.pdf`.

MPI-Forum. MPI: A message-passing interface standard, version 3.1. Technical report, University of Tennessee, Knoxville, TN, 2015.

John W. Mullin. *Crystallization*. Butterworth-Heinemann, Oxford, 4th edition, 2001.

Peter Neugebauer and Johannes G. Khinast. Continuous crystallization of proteins in a tubular plug-flow crystallizer. *Crystal Growth & Design*, 15(3): 1089–1095, 2015.

James R. Norris. *Markov Chains*. Cambridge Series on Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, UK & New York, NY, 1st edition, 1998.

James R. Norris. Smoluchowski's coagulation equation: uniqueness, nonuniqueness and a hydrodynamic limit for the stochastic coalescent. *Ann. Appl. Probab*, 9(1):78–109, 1999.

Sigrun Ortleb and Willem Hundsdorfer. Patankar-type Runge–Kutta schemes for linear PDEs. *AIP Conference Proceedings*, 1863(1):320008, 2017.

Wojciech Ozanski. Lagrange multipliers and stationary Stokes equations, 2015. URL `http://www2.warwick.ac.uk/fac/sci/masdoc/people/studentpages/students2014/ozan ski/`.

Suhas V. Patankar. *Numerical Heat Transfer and Fluid Flow.* McGraw Hill, New York, NY, 1980.

Robert I. A. Patterson. *Numerical Modelling of Soot Formation.* PhD thesis, University of Cambridge, 2007.

Robert I. A. Patterson. Convergence of stochastic particle systems undergoing advection and coagulation. *Stoch. Anal. Appl.*, 31(5):800–829, 2013.

Robert I. A. Patterson and Markus Kraft. Weighted particle methods for the Smoluchowski coagulation equation. Technical Report 47, University of Cambridge, Department of Chemical Engineering, 2007.

Robert I. A. Patterson and Wolfgang Wagner. A stochastic weighted particle method for coagulation-advection problems. *SIAM J. Sci. Comp.*, 34:B290–B311, 2012.

Robert I. A. Patterson, Jasdeep Singh, Michael Balthasar, Markus Kraft, and James R. Norris. The linear process deferment algorithm: A new technique for solving population balance equations. *SIAM J. Sci. Comp.*, 28(1):303–320, 2006.

Robert I. A. Patterson, Wolfgang Wagner, and Markus Kraft. Stochastic weighted particle methods for population balance equations. *J. Comput. Phys.*, 230(19):7456–7472, 2011.

Doraiswami Ramkrishna. *Population Balances: Theory and Applications to Particulate Systems in Engineering.* Academic Press, San Diego, CA, 2000.

Alan D. Randolph and Maurice A. Larson. *Theory of Particulate Processes: Analysis and Techniques of Continuous Crystallization.* Academic Press, New York, NY, 1988.

Joachim Rang. Pressure corrected implicit θ-schemes for the incompressible Navier–Stokes equations. *Appl. Math. Comput.*, 201(1-2):747–761, 2008.

Hans-Görg Roos, Martin Stynes, and Lutz Tobiska. *Robust Numerical Methods for Singularly Perturbed Differential Equations. Convection-Diffusion-Reaction and Flow Problems.* Springer, 2nd edition, 2008.

Vitoriano Ruas. Mixed finite element methods with discontinuous pressures for the axisymmetric Stokes problem. *ZAMM - J. Appl. Math. Mech.* , 83(4): 249–264, 2003.

Youcef Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993.

Karl K. Sabelfeld, Sergey V. Rogasinsky, Anastasia A. Kolodko, and Alexander I. Levykin. Stochastic algorithms for solving Smolouchovsky coagulation equation and applications to aerosol growth simulation. *Monte Carlo Methods Appl.*, 2(1), 1996.

Friedhelm Schieweck. A general transfer operator for arbitrary finite element spaces. Preprint 00-25 172, Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, 2000.

B. H. Shah, Doraiswami Ramkrishna, and J. D. Borwanker. Simulation of particulate systems using the concept of the interval of quiescence. *AIChE Journal*, 23(6):897–904, 1977.

Hang Si. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Software*, 41(2):Art. 11, 36, 2015.

Marian von Smoluchowski. Drei Vorträge über Diffusion, Brownsche Bewegung und Koagulation von Kolloidteilchen. *Physikalische Zeitschrift*, 17:557–585, 1916.

Hermann Sohr. *The Navier-Stokes Equations*. Birkhäuser Verlag, Basel, 2001.

O. Carina Suciu. *Numerical Methods Based on Direct Discretizations for Uni- and Bi-Variate Population Balance Systems*. PhD thesis, Freie Universität Berlin, 2013.

Roger Temam. *Navier-Stokes Equations. Theory and Numerical Analysis*. North-Holland, Amsterdam, 1977.

Erik Temmel, Holger Eisenschmidt, Heike Lorenz, Kai Sundmacher, and Andreas Seidel-Morgenstern. A short-cut method for the quantification of crystallization kinetics. 1. Method development. *Crystal Growth & Design*, 16 (12):6743–6755, 2016.

Stefan Turek and Michael Schäfer. Benchmark computations of laminar flow around cylinder. In E. Hirschel, editor, *Flow Simulation with High-Performance Computers II*, volume 52, pages 547–566. Vieweg, 1996.

Nicolaas G. van Kampen. *Stochastic Processes in Physics and Chemistry*. Elsevier North-Holland, Amsterdam ; Boston, 1st ed edition, 1981.

Nicolaas G. van Kampen. Remarks on non-Markov processes. *Brazilian Journal of Physics*, 28(2), 1998.

Surya P. Vanka. Block-implicit multigrid solution of Navier–Stokes equations in primitive variables. *J. Comput. Phys.*, 65(1):138–158, 1986.

Wolfgang Wagner and Andreas Eibeck. Stochastic interacting particle systems and nonlinear kinetic equations. *Ann. Appl. Probab*, 13(3):845–889, 2003.

Viktoria Wiedmeyer, Felix Anker, Clemens Bartsch, Andreas Voigt, Volker John, and Kai Sundmacher. Continuous crystallization in a helically coiled flow tube: Analysis of flow field, residence time behavior, and crystal growth. *Ind. Eng. Chem. Res.*, 56(13):3699–3712, 2017.

Ulrich Wilbrandt, Clemens Bartsch, Naveed Ahmed, Najib Alia, Felix Anker, Laura Blank, Alfonso Caiazzo, Sashikumaar Ganesan, Swetlana Giere, Gunar Matthies, Raviteja Meesala, Abdus Shamim, Jagannath Venkatesan, and Volker John. ParMooN—A modernized program package based on mapped finite elements. *Computers & Mathematics with Applications*, 74(1):74–88, 2017.

J.-S. Wu and Y.-Y. Lian. Parallel three-dimensional direct simulation Monte Carlo method and its applications. *Comput. Fluids*, 32(8):1133–1160, 2003.

Mohammad R. Yaghouti, Fraydoun Rezakhanlou, and Alan Hammond. Coagulation, diffusion and the continuous Smoluchowski equation. *Stoch. Process. Appl.*, 119(9):3042–3080, 2009.

Steven T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *J. Comput. Phys.*, 31:335–362, 1979.

Haibo Zhao and Chuguang Zheng. A population balance-Monte Carlo method for particle coagulation in spatially inhomogeneous systems. *Comput. Fluids*, 71:196–207, 2013.

Haibo Zhao, F. Einar Kruis, and Chuguang Zheng. A differentially weighted Monte Carlo method for two-component coagulation. *J. Comput. Phys.*, 229 (19):6931–6945, 2010.

Zusammenfassung der Dissertation

In der vorliegenden Arbeit wird ein neuer Algorithmus zur numerischen Lösung von Populationsbilanzsystemen vorgeschlagen, detailliert beschrieben und in zwei Simulationsprojekten zur Anwendung gebracht. Die betrachteten Populationsbilanzsysteme stammen dabei aus dem Bereich der chemischen Verfahrenstechnik. Vor allem werden Kristallisationsprozesse in strömender Umgebung betrachtet. Die beschreibenden Populationsbilanzgleichungen sind Erweiterungen der klassischen Smoluchowski-Koagulationsgleichung, und erben von dieser die mit dem Koagulationsintegral verbundenen Schwierigkeiten, insbesondere in Hinblick auf höherdimensionale Partikelmodelle.

In dem neuen Algorithmus werden zwei sehr unterschiedliche Bereiche der numerischen Mathematik und des Wissenschaftlichen Rechnens zusammengebracht, nämlich eine stochastische Partikelsimulation, die auf einer Markovprozess Monte-Carlo Methode aufbaut, und (deterministische) Verfahren der numerischen Strömungsmechanik mit Finiten Elementen.

Stochastische Partikelsimulationen sind erprobte Methoden für die Lösung von Populationsbilanzgleichungen. Ihre Stärken sind, dass sie mikroskopische Informationen aus dem Modell aufnehmen können, gleichzeitig Konvergenz gegen Lösungen der makroskopischen Gleichung bieten, und des Weiteren numerisch effizient und robust sind. Die Einbettung einer solchen stochastischen Methode in eine deterministische Strömungssimulation eröffnet daher neue Möglichkeiten der Lösung gekoppelter Populationsbilanzsysteme, vor allem in Hinblick auf den mikroskopischen Charakter der Interaktion von Partikeln.

Das durch die Einbettung einer stochastischen Partikelsimulation in eine Finite Elemente Simulation gewonnene Simulationsverfahren wird zunächst auf ein Populationsbilanzsystem in einem, für die Herstellung von Aspirin verwendeten, achsensymmetrisch zweidimensional modellierten Rohrkristaller angewendet. Experimentelle Daten können in überschaubarer Rechenzeit reproduziert werden. Zudem wird das Verfahren auf drei Raumdimensionen erweitert und für die Simulation eines experimentellen Wirbelschichtkristallers benutzt. Dieses System ist voll instationär, die turbulente Strömung wird on-the-fly mitberechnet.

Die verwendeten numerischen Verfahren aus den Bereichen der Simulation der Navier–Stokes Gleichungen, der Konvektions-Diffusions-Reaktionsgleichungen und der stochastischen Partikelsimulation werden ausführlich eingeführt, motiviert und diskutiert. Ebenso werden die Kopplungsphänomene in den betrachteten Populationsbilanzsystemen und der Kopplungsalgorithmus selbst eingehend besprochen. Darüber hinaus werden eigene Ergebnisse aus dem Bereich der effizienten numerischen Simulation der Navier–Stokes Gleichungen präsentiert, namentlich ein Vergleich von schnellen Lösern für Sattelpunktprobleme und eine eigene Interpretation einer klassischen Gebietszerlegungs-Methode für die Parallelisierung der Finiten Elemente Methode.

Erklärung der Selbstständigkeit

Ich versichere hiermit, alle Hilfsmittel und Hilfen angegeben zu haben, und die Arbeit selbstständig und ausschließlich auf Grundlage der angegebenen Hilfsmittel und Hilfen angefertigt zu haben. Des Weiteren versichere ich, die Arbeit oder Teile der Arbeit nicht schon einmal in einem früheren Promotionsverfahren eingereicht zu haben.

Berlin, 7. April 2018

Clemens Bartsch