Efficient numerical treatment of aggregation integrals in multivariate population balance equations

Vom Promotionsausschuss der
Technischen Universität Hamburg
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation

von
Robin Christoph Ahrens

aus
Hamburg, Deutschland

2020

Gutachter:
Prof. Sabine Le Borne (TUHH)
Prof. Peter Benner (MPI Magdeburg)

Datum der mündlichen Prüfung:
8. Juni 2020

Acknowledgements

Foremost, I would like to express my gratitude towards my advisor Prof. Sabine Le Borne for her continuous support over the course of my studies and research. Her keen eye for possible extensions of existing methodology led my path over the last years and I want to thank her for several fruitful discussions. Her patience in correcting flaws and uncovered edge cases in my definitions and proofs has shown to be a skill of immeasurable value to me and hopefully will be to others as well.

Furthermore I want to thank the remainder of my thesis committee Prof. Peter Benner and Prof. Daniel Ruprecht for their invested time and for insightful discussions and questions during my defense.

This thesis would not be possible without the financial support of the DFG under SPP 1679 (Dynamische Simulation vernetzter Feststoffprozesse). I would like to thank them as well as my colleagues and collaborators in this project for providing me with a physical and worldly application of the abstract mathematics I encoutered. I especially have to thank Prof. Stefan Heinrich, Prof. Volker John and Prof. Kai Sundmacher and their respective graduate students.

Additionally I want to extend my gratitude towards every person in the institute of mathematics at TUHH, regardless of whether their field as adjacent to mine or not. Several important discussions were held in seminars, in offices or during lunchtime, of which only some pertained mathematics. I will remember these as fondly as I will my colleagues.

My personal acknowledgement goes in the largest part to my girlfriend who spend her christmas break to comfort me finishing this thesis. I could imagine a lot of things more exciting to spend these free days on then helping me to sit in front of my keyboard and staring at letters on paper. I am eternally grateful to her for everything she has done and will do in the years to come.

Secondary acknowledgement goes to SingING and the Theater AG. These supplied the necessary amounts of distractions and other things to focus on before starting the problem anew at the next day. Humming a melody or reciting a dialogue on your way home are healthier alternatives to trying to solve equations or making sense of barely visualised data in your head while trying to cross a street in the dark.

Finally I want to thank the reader for your interest in my research. I hope it helps you on your quest for knowledge or at least tells you what not to try.

Contents

List of Figures

List of Tables

Acronyms

CAT Cell average technique

FFT Fast Fourier transformation

FP Fixed pivot

IVP Initial value problem

MOM Method of moments

ODE Ordinary differential equation

QMOM Quadrature method of moments

PC Piecewise constant

PL Piecewise linear

PBE Population balance equation

SVD Singular value decomposition

TT Tensor-train

TT-RC Tensor-train renormalization-cross

Chapter 1

Introduction

In this chapter, a general overview of population balance equations (PBEs) is given. In section 1.1 we introduce the processes considered in the modeling of population balance equations and formulate the equations used to generate a mathematical model of the physical process. In section 1.2 we outline the novel contributions of this thesis and section 1.3 will summarize the structure of this thesis.

1.1 Population balance equations

Applications of population balance equations can be found in a wide variety of processes, where particles are dispersed and dissolved in an environmental phase. These include fields like astrophysics, meteorology, geology, biophysics, aeronautics, civil engineering and (most notably) chemistry and biochemistry. Population balance equations are used in these fields to study crystallization in disperse phases (such as heterogeneous solid-liquid or solid-gas phases), the separation of mixtures of two liquids or reactor equipment such as fluidized bed reactors, microbiological reactors and other processes.
In all of these processes the dispersed phase consists of a population of particles, where each particle can be characterized by internal properties like mass, surface area, chemical composition or electric charge. These properties associate every particle with a coordinate vector $\mathbf{v} = (v_1, v_2, \ldots, v_d) \in \mathbb{R}^d$, where d is the number of observed properties. This thesis will only cover additive properties that are well suited in the setting of aggregation.
The properties of particles change over time due to physical processes, like growth (see [51] and [24]), breakage (see [33] and [6]), nucleation (see [30]) and aggregation (see [30], [37] and [47]). Schematic illustrations of these four processes are shown in Figure 1.1.

Growth: Growth of particles happens when dissolved mass spreads over the surface of a particle that becomes larger in the process. This does not af-

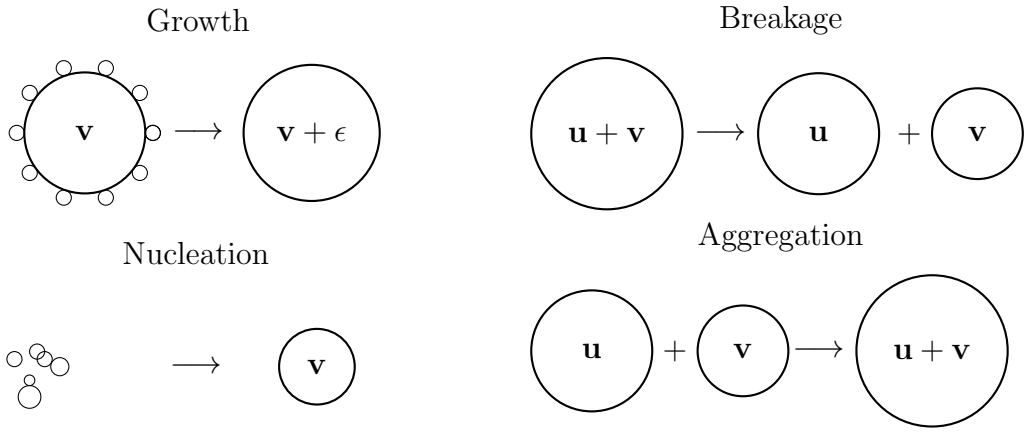


Figure 1.1: The four main processes in PBE: Growth, breakage, nucleation and aggregation.

fect the number of particles but increases the internal properties of the particle.

Breakage: In the breakage process large particles break into two or more fragments of smaller size. This increases the number of particles, while the sum of internal properties remains constant.

Nucleation: Nucleation is a process in which minuscule particles (most often dissolved in the environmental phase) precipitate and form new particles. This does increase the number of particles and gives an increase in the total sum of internal properties.

Aggregation: When two particles collide, they have a chance of sticking together and form stable bounds at the point of contact. This process is called aggregation (or coagulation) and is the opposite process of breakage. It will preserve the sum of internal properties but decreases the number of particles.

Due to these processes, the internal properties of involved particles change over time as does the overall number of particles, which complicates the naive approach of tracking all particles individually. The key idea of population balance equation is to consider a number density function $f(\mathbf{v}, t)$, which describes the amount of particles with internal properties $\mathbf{v}$ at a time t. A thorough introduction to population balance equations can be found in [62].
We assume that all particles are distributed homogeneously in space so we can neglect the location of a particle (usually referred to as external or spatial coordinate) and do not need to model the transport through the reactor. With

this assumption, the number density function is governed by the population balance equation, a partial integro-differential equation, given by

$$\begin{aligned}\frac{\partial f(\mathbf{v},t)}{\partial t} + \nabla_{\mathbf{v}} \cdot (G(\mathbf{v}) f(\mathbf{v},t)) &= Q(f) \\ &= Q_{\text{break}}(f) + Q_{\text{nuc}}(f) + Q_{\text{agg}}(f).\end{aligned} \tag{1.1}$$

The second term on the left hand side of (1.1) represents the growth of particles at the rate $G(\mathbf{v})$, while $Q(f)$ on the right hand size summarizes the appearing and vanishing particles due to breakage ($Q_{\text{break}}(f)$), nucleation ($Q_{\text{nuc}}(f)$) and aggregation ($Q_{\text{agg}}(f)$). This thesis focuses solely on the numerical treatment of the aggregation process, neglecting the influence of growth, breakage and nucleation completely. The dynamic evolution of $f(\mathbf{v},t)$ is then governed by an ordinary differential equation (ODE) of the form

$$\begin{aligned}\frac{\partial f(\mathbf{v},t)}{\partial t} &= Q_{\text{agg}}(f)(\mathbf{v},t) \\ &= Q^{\text{source}}(f)(\mathbf{v},t) - Q^{\text{sink}}(f)(\mathbf{v},t),\end{aligned} \tag{1.2}$$

where

$$Q^{\text{source}}(f)(\mathbf{v},t) = \frac{1}{2} \int_0^{v_1} \cdots \int_0^{v_d} \kappa(\mathbf{u}, \mathbf{v}-\mathbf{u}) f(\mathbf{u},t) f(\mathbf{v}-\mathbf{u},t)\, d\mathbf{u} \tag{1.3}$$

and

$$Q^{\text{sink}}(f)(\mathbf{v},t) = f(\mathbf{v},t) \int_0^{\infty} \cdots \int_0^{\infty} \kappa(\mathbf{u}, \mathbf{v}) f(\mathbf{u},t)\, d\mathbf{u} \tag{1.4}$$

denote the number density of particles with internal properties $\mathbf{v}$ appearing (source) and vanishing (sink) in the aggregation process. These two terms (1.3) and (1.4) will be referred to as source term and sink term throughout this thesis. Here, $\kappa(\mathbf{v},\mathbf{u})$ stands for the aggregation kernel function, describing the frequency at which particles with internal properties $\mathbf{v}$ and $\mathbf{u}$ are aggregating. We note that (1.2) is also referred to as the Smoluchowski coagulation equation [67]. With a given initial distribution $f(\mathbf{v},0) = f_0(\mathbf{v})$, (1.2) yields an initial value problem (IVP) that is to be solved.

In addition to the number density function $f(\mathbf{v},t)$ one is also interested in the so-called moments $\mu^{\mathbf{e}}(f)(t)$, that are defined by

$$\begin{aligned}\mu^{\mathbf{e}}(f)(t) &:= \int_0^{\infty} \cdots \int_0^{\infty} \prod_{m=1}^{d} v_m^{e_m} f(\mathbf{v},t)\, dv_m \cdots dv_1 \\ &=: \int_{\mathbf{0}}^{\infty} \mathbf{v}^{\mathbf{e}} f(\mathbf{v},t)\, d\mathbf{v}\end{aligned} \tag{1.5}$$

for a vector of exponents $\mathbf{e} := (e_1, \ldots, e_d) \in \mathbb{N}^d$. Most notable are the zero-th moment with $\mathbf{e} = (0, \ldots, 0)$ that denotes the total number of particles and the first m-th moments with $\mathbf{e} = (0, \ldots, 0, 1, 0, \ldots, 0)$ that denote the total amount of the m-th property that is present in all particles. If the m-th internal property is denoting the mass of a single particle, this moment denotes the total mass of all particles. These first moments will be constant over time during the process of aggregation because we are focusing on additive properties.
Definition (1.5) also shows the short-hand notation we will apply for multivariate integrals over the remainder of this thesis.

1.2 Contributions of this thesis

This thesis focuses on the efficient and accurate evaluation of the aggregation terms in population balance equations (1.3) and (1.4), the right-hand side of (1.2) in the multivariate setting. The source integral of newly formed particles (see (1.3)) is quadratic with respect to the number density $f(\mathbf{v}, t)$ and of convolution type, usually being the bottleneck in numerical approaches of this equation, as it has a complexity of $\mathcal{O}(N^2)$ (N denotes the degrees of freedom in a discretization, often exponential in the number of dimensions d) in a straight forward implementation (see [5]). This makes a high number of degrees of freedom infeasible, even in the univariate setting.

Efficient algorithms for the univariate case were developed in [45], [43] and [44] (based on earlier theoretical foundations from [27] and [26]) and we investigated their application to dynamic flowsheet simulation in [66]. Our main contribution lies in the adaptation of these algorithms to the multivariate setting and investigation of their potential. These algorithms reduce the complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log(N))$ for univariate problems where every particle is defined by one single property.
We furthermore investigate the influence of our chosen discretization on the conservation of moments. The source term (1.3) will require additional attention in the form of a projection. We present a framework that allows for the conservation of all moments $\mu^{\mathbf{e}}(f)(t)$ with $\|\mathbf{e}\|_\infty < 2$.

We also overcome the so-called *curse of dimensionality* that makes high dimensional problems infeasible to process on non-supercomputers, due to limitations in available storage space. The required space is exponential in d, making a traditional approach infeasible for high dimensional problems. We adapt an efficient storage technique, namely the tensor-train (TT) format, to make it applicable in the setting of PBEs and give new results to increase the efficiency of algorithms to make better use of this format in our setting.This reduces the storage complexity to be linear in d, which makes high-dimensional problems tractable.

All presented algorithms were implemented using the *C++* programming language and we show extensive numerical tests to show the accuracy, efficiency and advantages of the presented methods.

The efficient algorithms for the univariate case of one particle property allow us to tackle the closely related problem of kernel-estimation. It is an inverse problem that aims to find a kernel function $\kappa(u, v)$ for a given number density function $F(v, t)$ (denoted by a capital letter as it is given). We here show two new approaches to solve this ill-conditioned problem in the univariate setting for discrete-in-time measurements and show the promising results with numerical simulations. The programming language MATLAB is used to obtain these results.

This thesis is based upon and extends our previously published and submitted works [2], [1], [3], [4] and [66]. The aim of the papers and this thesis is to explore the applicability of the theoretical foundations of the univariate aggregation-problem to the multivariate setting and to solve the closely related problem of kernel-approximation.

1.3 Outline of contents

In chapter 2 we review existing methods to solve the aggregation problem in PBEs including Monte Carlo methods, finite volume and finite element methods. We go further in-depth for two popular algorithms, namely the Fixed pivot (FP) method and the Cell average technique (CAT) in the multivariate setting.

In chapter 3 we discuss the discretization of the property space and the pre-requisite for the application of our algorithms in the case of full tensor storage. We show the necessary steps towards efficient evaluation of source and sink terms and introduce the projection to accurately predict all moments $\mu^{\mathbf{e}}(f)(t)$ with $\|\mathbf{e}\|_\infty < 2$ during the simulation. We compare the resulting algorithm with the existing methods introduced earlier via different numerical experiments.

In chapter 4 we give a thorough introduction to the tensor-train (TT) format that we use to overcome the storage complexity of full tensor representation. We introduce the basic concept of efficient tensor storage and the representation used in the TT-format. We present the necessary arithmetic operations in the context of aggregation and develop an improvement of the truncation algorithm. We will again show numerical results of the resulting algorithms to show their effectiveness.

In chapter 5 we return to the univariate setting and present two approaches to solve the inverse problem of kernel estimation. This problem is concerned with finding an approximation of the aggregation kernel $\kappa(u, v)$ from a given number density function $F(v, t)$ at multiple discrete time points t_0 to t_m.

In chapter 6 we provide a summary of the algorithms we developed and numerical results we have shown in the preceding chapters. We give conclusions about their quality and use-cases. We finally give an outlook and mention further adaptations of the presented algorithms and data structures towards faster and more accurate calculations and towards a wider range of applications.

Chapter 2

Established numerical methods

In this chapter, we provide a short introduction to several already established numerical methods used to tackle the problem of aggregation and population balance equations in general. In section 2.1 we shortly summarize existing numerical methods. Section 2.2 and 2.3 give more detail about the fixed pivot method and the Cell average technique, two very popular methods, that are in essence very similar to our approach. We will later use these methods to compare our new approach with these methods.

2.1 Summary of established methods

Analytic solutions to population balance equations are only given in a few cases of aggregation kernel $\kappa(\mathbf{u}, \mathbf{v})$ and initial distribution $f_0(\mathbf{v})$. Some of these results can be found in [23] and [20] for the constant kernel ($\kappa_C(\mathbf{u}, \mathbf{v}) = \lambda$) and the sum-kernel ($\kappa_\Sigma(\mathbf{u}, \mathbf{v}) = \sum_{m=0}^{d} (u_m + v_m)$) respectively. They are obtained by means of Laplace transformation. We will use these as reference solutions and compare the results obtained by other numerical methods with them in sections 3.5 and 4.5.
Another approach (currently restricted to univariate distributions) based on homotopy perturbation and power-series was presented in [34] and is able to confirm the previously obtained results and give new results [63].

A popular method to solve univariate PBEs is the Method of moments (MOM) or Quadrature method of moments (QMOM) from [19] and [49]. These methods are able to track the moments $\mu^{\mathbf{e}}(f)(t)$ very accurately with small computational complexity. The resulting particle distribution however is not directly accessible. Further development towards reconstruction was made in [6]. An extension to bi- and multivariate distributions was mentioned in [71] but does not overcome this drawback.

Finite volume methods are most suitable for equations that experience both aggregation and breakage because the first moment is automatically preserved (see [21], [22]). These methods are not concerned with the number of particles

of a given size (i.e, the particle distribution $f(\mathbf{v}, t)$) but the total mass of particles of any size (i.e., one first moment). Finite Element methods are somewhat similar as they approximate the particle distribution $f(\mathbf{v}, t)$ in a linear space spanned by basis functions, whose weights are determined by forcing an integral residual to zero (more details in [48] and [52]). They are of particular interest when trying to find a steady state of a given system that includes breakage.

A probalistic approach to solve the PBE is given by Monte Carlo methods ([36], [46]). Instead of tracking the particle distribution $f(\mathbf{v}, t)$, we keep track of a random sampling of particles from this distribution. We simulate their aggregation via randomizing which particles may aggregate in events, while progressing in time. These results must be taken with some caution, as they are the result of randomness and always show different outcomes if repeated. One can reduce this influence by including more sampled particles which in turn increases the computational time, see [32].

2.2 Fixed Pivot method

This section is devoted to the fixed pivot method, which is often considered to be the easiest approach to discretize a PBE. It was introduced in [41] and generalized to multiple inner coordinates in [69]. We here present a version that uses tensor product discretization. The generalization for different discretizations is beyond the scope of this short introduction. The method is based on the choice of so-called *pivot points* $\mathbf{v_i} = (v_{i_1}, \ldots, v_{i_d})$, where particles are assumed to be concentrated. We define grid points $\mathbf{g_i} = (g_{i_1}, \ldots, g_{i_d})$ with $g_i \leq v_i \leq g_{i+1}$ and finally cells

$$C_{\mathbf{i}} = [g_{i_1}, g_{i_1+1}] \times \cdots \times [g_{i_d}, g_{i_d+1}].$$

An example grid for $d = 2$ is given in Figure 2.1.

Each pivot is associated with a macroscopic variable

$$N_{\mathbf{i}}(t) = \int_{C_{\mathbf{i}}} f(\mathbf{v}, t) d\mathbf{v}, \tag{2.1}$$

resulting in a particle distribution

$$\begin{aligned} N(\mathbf{v}, t) &= \sum_{i_1=0}^{n_1-1} \ldots \sum_{i_d=0}^{n_d-1} N_{\mathbf{i}} \cdot \delta(\mathbf{v} - \mathbf{v_i}) \\ &=: \sum_{\mathbf{i=0}}^{\mathbf{n-1}} N_{\mathbf{i}} \cdot \delta(\mathbf{v} - \mathbf{v_i}) \end{aligned} \tag{2.2}$$

with the Kronecker-delta $\delta(\mathbf{v})$. We will use this notation of a multivariate sum over the remainder of this thesis to reduce notational overhead.

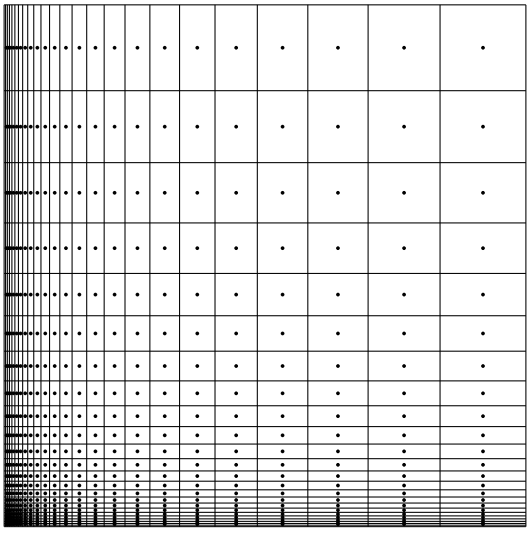


Figure 2.1: Example for a grid used in the fixed-pivot method for $d = 2$. This geometric grid is more accurate for smaller particles. Pivots $\mathbf{v_i}$ are shown with dots.

With this discretization in mind, the source term (1.3) can be expressed in terms of Kronecker-deltas as well and becomes

$$Q^{\text{source}}(\mathbf{v}, t) = \frac{1}{2} \sum_{\mathbf{i}=0}^{\mathbf{n-1}} \sum_{\mathbf{j}=0}^{\mathbf{n-1}} N_{\mathbf{i}}(t) N_{\mathbf{j}}(t) \kappa(\mathbf{v_i}, \mathbf{v_j}) \cdot \delta(\mathbf{v} - (\mathbf{v_i} + \mathbf{v_j})), \tag{2.3}$$

which is not always representable in the pivot expression (2.2) because the resulting aggregates $\mathbf{v_i} + \mathbf{v_j}$ are not positioned at any of the pivot points $\mathbf{v_k}$. The sink term (1.4) is described by

$$Q^{\text{sink}}(\mathbf{v}, t) = \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n-1}} N_{\mathbf{i}}(t) \delta(\mathbf{v} - \mathbf{v_i}) \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n-1}} \kappa(\mathbf{v_i}, \mathbf{v_j}) N_{\mathbf{j}}(t) \tag{2.4}$$

and is already in the pivot expression (2.2).
To approximate (2.3) in terms of (2.2), a moment preserving projection is chosen to redistribute aggregates to nearby pivots. The aggregate of particles of sizes $\mathbf{v_i}$ and $\mathbf{v_j}$ has properties given by $\widehat{\mathbf{v}}_{\mathbf{i,j}} := \mathbf{v_i} + \mathbf{v_j}$, which is not a pivot point for all combinations of $\mathbf{i}$ and $\mathbf{j}$. We find the multi index $\mathbf{k}$ that satisfies $\mathbf{v_k} \leq \widehat{\mathbf{v}}_{\mathbf{i,j}} < \mathbf{v_{k+1}}$ in an elementwise sense, i.e., the 2^d pivots surrounding the aggregate in the state space. The aggregate is then divided among these pivots to preserve any 2^d moments. The usual choice of the first 2^d moments $\mu^{(0,\ldots,0)}(f)(t)$ to $\mu^{(1,\ldots,1)}(f)(t)$ are conserved by assigning

$$e^{\mathbf{q}}(\mathbf{i}, \mathbf{j}) = \prod_{m=1}^{d} \left| \frac{(v_{i_m} + v_{j_m}) - v_{k_m+1-q_m}}{v_{k+1} - v_k} \right|$$

to the nearby pivot $\mathbf{v_{k+q}}$ for each given $\mathbf{q} \in \{0,1\}^d$, i.e., the factor is proportional to the distance to the pivot on the other side.
With these projections, (2.3) becomes

$$Q^{\text{source}}(\mathbf{v},t) = \frac{1}{2}\sum_{\mathbf{i=0}}^{\mathbf{n-1}}\sum_{\mathbf{j=0}}^{\mathbf{n-1}} N_{\mathbf{i}}(t)N_{\mathbf{j}}(t)\kappa(\mathbf{v_i},\mathbf{v_j}) \cdot \sum_{\mathbf{q=0}}^{\mathbf{1}} e^{\mathbf{q}}(\mathbf{i},\mathbf{j})\delta(\mathbf{v}-\mathbf{v_{k+q}}). \quad (2.5)$$

We can see one disadvantage of the fixed pivot method in (2.5) and (2.4) as there is a triple summation in the source term and a double summation in the sink term. This implies a computational complexity of $\mathcal{O}((n^d)^2 \cdot 2^d)$ for the source term and $\mathcal{O}((n^d)^2)$ for the sink term. This is quadratic in the number of pivot points that are itself exponential in the number of dimensions. This immediately shows a computational bottleneck in the calculations.
Many variations of this procedure have been proposed (for example [12], [42]) that use different meshes or moving pivots $\mathbf{v_i}$. They still share the downside of the high computational complexity but reduce the number of required pivots to reach a certain accuracy.

2.3 Cell Average Technique

This section is devoted to the cell average technique (CAT), which is closely related to the fixed pivot method, as it also works with a macroscopic representation (2.2). It was presented in [35], [39] and generalized to the multivariate case in [15], [38] and [65]. It differs from the FP method by making greater use of the introduced cells $C_{\mathbf{i}}$. We similarly to the FP method find

$$B_{\mathbf{k}}(t) = \frac{1}{2}\sum_{\mathbf{v_i}+\mathbf{v_j}\in C_{\mathbf{k}}} N_{\mathbf{i}}(t)N_{\mathbf{j}}(t)\kappa(\mathbf{v_i},\mathbf{v_j}) \quad (2.6)$$

as the total amount of particles appearing in a cell $C_{\mathbf{k}}$ and define

$$G_{\mathbf{k},m}(t) = \frac{1}{2}\sum_{\mathbf{v_i}+\mathbf{v_j}\in C_{\mathbf{k}}} N_{\mathbf{i}}(t)N_{\mathbf{j}}(t)\kappa(\mathbf{v_i},\mathbf{v_j}) \cdot (\mathbf{v}_{i_m}+\mathbf{v}_{j_m}) \quad (2.7)$$

as the total inflow of the m-th property into that cell. We then find the average

$$\overline{\mathbf{v}}_{\mathbf{k},m} := \frac{G_{\mathbf{k},m}(t)}{B_{\mathbf{k}}(t)} \quad (2.8)$$

of the m-th property of the newly formed particles in $C_{\mathbf{k}}$.
Instead of projecting each newly formed particle individually (as it is done in the FP scheme) we assume all particles in $C_{\mathbf{k}}$ have properties denoted by $\overline{\mathbf{v}}_{\mathbf{k}} \in \mathbb{R}^d$ and do the projection from there.
This procedure is considered more accurate than the fixed pivot technique (see again [35], [39]) in tracking the population density and higher order moments,

even though we loose consistency with some cross-moments, where $\sum_{m=1}^{d} e_d \geq 2$. This method shows a lower computational complexity of $\mathcal{O}(n^{2d} + (2n)^d)$ for the source term as the projection is not inside the inner loop anymore and this leads to overall shorter computational times. Several extensions have been proposed (e.g. [40]) but are again beyond the scope of this introduction.

Chapter 3

Discretization and full tensor storage

This chapter is devoted to introducing the discretization of the particle state space and discretizing the number density function $f(\mathbf{v}, t)$ and the kernel function $\kappa(\mathbf{v}, \mathbf{u})$ that we will use in this thesis. This will be done in section 3.1 where we also clarify our notation of tensors and tensor elements. In section 3.2 we present the core assumption of this work and show the possible simplifications of the equations from thereon. Section 3.3 combines the two previous sections towards an algorithm for efficient numerical evaluation source and sink terms. Section 3.4 introduces the reader to the Fourier transform, its properties and the efficient evaluation via fast Fourier transformation (FFT).
To focus on the mathematical evaluation we use dimensionless particle properties and a dimensionless time.

3.1 Discretization of the property space

To evaluate the right hand side of the population balance equation (1.2) numerically we discretize the state space. For a state space of dimension $d \in \mathbb{N}$, define the index set $\mathcal{M} := \{1, \dots, d\}$. For every index $m \in \mathcal{M}$, we choose a cutoff value $v_m^{\max} \in \mathbb{R}_+$ and assume

$$f(\mathbf{v}, t) = 0 \quad \text{if} \quad \exists m : v_m > v_m^{\max}. \tag{3.1}$$

In addition we choose $\mathbf{n}^{\mathcal{M}} = \big(n_1, \dots, n_d\big) \in \mathbb{N}^d$ and define index and multi-index sets

$$\begin{aligned} I_m &:= \{0, \dots, n_m - 1\}, & I'_m &:= \{0, \dots, n_m\}, \\ I_{\mathbf{n}^{\mathcal{M}}} &:= I_1 \times \ldots \times I_d, & I'_{\mathbf{n}^{\mathcal{M}}} &:= I'_1 \times \ldots \times I'_d. \end{aligned}$$

We divide each interval $[0, v_m^{\max}]$ into n_m uniform subintervals of length

$$h_m := \frac{v_m^{\max}}{n_m}, \quad m \in \mathcal{M},$$

and define sets of uniform gridpoints $\mathcal{G}_m := \{i_m \cdot h_m \mid i_m \in I'_m\}$ and the tensor grid $\mathcal{G}^{\mathcal{M}} := \bigtimes_{m=1}^{d} \mathcal{G}_m = \{\mathbf{g_i} \mid \mathbf{i} = (i_1, \ldots, i_d) \in I'_{\mathbf{n}^{\mathcal{M}}}\}$ with gridpoints

$$\mathbf{g_i} := (i_1 h_1, \ldots, i_d h_d)$$

in the multi-dimensional hyper-rectangle $[\mathbf{0}, \mathbf{v}^{\max}] := [0, v_1^{\max}] \times \cdots \times [0, v_d^{\max}]$, leading to

$$N^{\mathcal{M}} := \prod_{m=1}^{d} n_m \tag{3.2}$$

cells $\mathcal{C}_{\mathbf{i}}^{\mathcal{M}} := (i_1 h_1, (i_1+1)h_1) \times \cdots \times (i_d h_d, (i_d+1)h_d)$ for multi-indices $\mathbf{i} = (i_1, \ldots, i_d) \in I_{\mathbf{n}^{\mathcal{M}}}$, each one of volume

$$V^{\mathcal{M}} := \prod_{m=1}^{d} h_m. \tag{3.3}$$

One example of this grid with $d = 2$ and $\mathbf{n} = (8, 4)$ can be seen in Figure 3.1. We will call a grid $\mathcal{G}$ *symmetric*, if $n_1 = \cdots = n_d$ and $v_1^{\max} = \cdots = v_d^{\max}$ as all $\mathcal{G}_m$ are equal.
We discretize $f(\mathbf{v}, t)$ to be constant in every cell of $\mathcal{G}^{\mathcal{M}}$, and from now on assume that $f(\mathbf{v}, t)$ is already of this form and identify it with the corresponding tensor $f \in \mathbb{R}_{\geq 0}^{n_1 \times \ldots \times n_d}$, i. e.,

$$f(\mathbf{v}, t) = f(\tilde{\mathbf{v}}, t) =: f_{\mathbf{i}} \quad \text{if} \quad \mathbf{v}, \tilde{\mathbf{v}} \in \mathcal{C}_{\mathbf{i}}^{\mathcal{M}}.$$

We will use $\mathbb{R}^{N^{\mathcal{M}}}$ to refer to $\mathbb{R}^{n_1 \times \cdots \times n_d}$ for brevity and drop the index $\mathcal{M}$ everywhere if $\mathcal{M} = \{1, \ldots, d\}$.

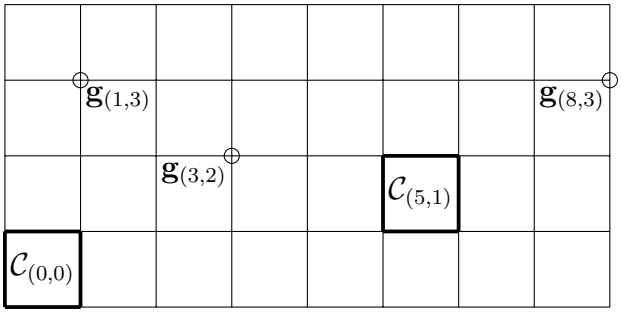


Figure 3.1: A uniform tensor grid $\mathcal{G}$ with $d = 2$, $\mathbf{n} = (8, 4)$ and $h_1 = h_2$.

3.2 Kernel separability

Throughout this thesis, we will heavily rely on the separability of the kernel function. A kernel function $\kappa(\mathbf{u}, \mathbf{v})$ will be called *separable with rank* k, if it can be written in the form

$$\kappa(\mathbf{u}, \mathbf{v}) = \sum_{\nu=1}^{k} \alpha^{\nu}(\mathbf{u}) \cdot \beta^{\nu}(\mathbf{v}) \tag{3.4}$$

with a set of k functions α^ν, $\beta^\nu : \mathbb{R}^N \mapsto \mathbb{R}$. This allows for a rewrite of (1.3) and (1.4) as follows. Using the definitions

$$\phi^\nu(\mathbf{v}, t) := f(\mathbf{v}, t) \cdot \alpha^\nu(\mathbf{v}) \tag{3.5}$$

and

$$\psi^\nu(\mathbf{v}, t) := f(\mathbf{v}, t) \cdot \beta^\nu(\mathbf{v}), \tag{3.6}$$

we have

$$Q^{\text{source}}(f)(\mathbf{v}, t) = \frac{1}{2} \sum_{\nu=1}^{k} \int_{\mathbf{0}}^{\mathbf{v}} \phi^\nu(\mathbf{u}, t) \psi^\nu(\mathbf{v} - \mathbf{u}, t) \, d\mathbf{u} \tag{3.7}$$

for the source term and

$$Q^{\text{sink}}(f)(\mathbf{v}, t) = \sum_{\nu=1}^{k} \psi^\nu(\mathbf{v}, t) \cdot \int_{[0,\infty]^d} \phi^\nu(\mathbf{u}, t) \, d\mathbf{u} \tag{3.8}$$

for the sink term, respectivly, and have both as the sum of k simpler integrals. We then define

$$Q_\nu^{\text{source}}(\mathbf{v}, t) := \frac{1}{2} \int_{\mathbf{0}}^{\mathbf{v}} \phi^\nu(\mathbf{u}, t) \psi^\nu(\mathbf{v} - \mathbf{u}, t) \, d\mathbf{u} \tag{3.9}$$

as a single term in (3.7). This integral is a pure convolution integral as opposed to being of convolution type, where the kernel function depends on both $\mathbf{u}$ and $\mathbf{v} - \mathbf{u}$. We similarly define

$$Q_\nu^{\text{sink}}(\mathbf{v}) := \psi^\nu(\mathbf{v}, t) \cdot \int_{[0,\infty)^d} \phi^\nu(\mathbf{u}, t) \, d\mathbf{u} \tag{3.10}$$

for the sink term (3.8). Here, the multivariate integral is independent of $\mathbf{v}$.
This assumption of separability is not always fulfilled by all kernels used in practical applications. But we can always find an approximate kernel $\widetilde{\kappa}(\mathbf{u}, \mathbf{v})$ that fulfills (3.4) by various means. These include (Chebyshev) interpolation, cross approximation, exponential sums or Taylor sums (see [7], [8]). Many kernels used in practice (and all used in this thesis) allow for a separable approximation with exponentially decaying error, i.e., the error reduces significantly with an increase in the rank k. This allows us to replace the exact kernel with a separable approximation without this dominating the resulting error.
A separable approximation can be computed from a discretized kernel as well. This is achieved by finding a low-rank approximation of a discrete matrix $K \in \mathbb{R}^{N \times N}$, most often done by computing the (truncated) singular value decomposition. This gives kernel factors in matrix form with $\alpha, \beta \in \mathbb{R}^{N \times k}$ to satisfy

$$K \approx \alpha \cdot \beta^T. \tag{3.11}$$

3.3 Evaluation of a piecewise constant source and sink term

We assume $f(\mathbf{v}, t)$ to be piecewise constant at all times and require the right-hand side of (1.2) to comply with this discretization. This section is devoted to the efficient evaluation of (3.9) and (3.10) and the necessary steps to uphold the piecewise constant structure by suitable projections if necessary.
We drop the time-variable t in the following sections to reduce notation. The differential equation (1.2) is autonomous i.e., does not explicitly depend on the current time.

3.3.1 Source term

The source term is given by

$$Q^{\text{source}}(\mathbf{v}) = \int_0^{v_1} \dots \int_0^{v_d} \kappa(\mathbf{u}, \mathbf{v}-\mathbf{u}) f(\mathbf{u}) f(\mathbf{v}-\mathbf{u}) \, d\mathbf{u}$$

as we recall (1.3). With the assumption of a separable kernel (see section 3.2) we defined in (3.9)

$$Q_\nu^{\text{source}}(\mathbf{v}) = \int_{\mathbf{0}}^{\mathbf{v}} \phi^\nu(\mathbf{u}) \psi^\nu(\mathbf{v}-\mathbf{u}) \, d\mathbf{u}$$

which is a multivariate convolution integral.
The piecewise constant discretization of $f(\mathbf{v})$ and $\kappa(\mathbf{u}, \mathbf{v})$ directly carries over to $\phi^\nu(\mathbf{u})$ and $\psi^\nu(\mathbf{v}-\mathbf{u})$. With this, we evaluate $Q_{\nu,\text{pl}}^{\text{source}}$ at grid point

$$Q_{\nu,\text{pl}}^{\text{source}}(\mathbf{g_{i+1}}) = V \cdot \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{i}} \phi_{\mathbf{k}}^\nu \cdot \psi_{\mathbf{i}-\mathbf{k}}^\nu \tag{3.12}$$

by substituting the integrals over a piecewise constant function by a summation over the cell values scaled by the volume V from (3.3) . The additional subscript pl stands for *piecewise linear* as the integral over a piecewise constant function is piecewise linear and continuous. The discrete values obtained in this convolution-sum denote the function values at gridpoints $\mathbf{g_{i+1}}$ with a shift $\mathbf{1} = (1, \dots, 1)$. This shift is necessary because $Q_{\nu,\text{pl}}^{\text{source}}(\mathbf{g_i}) = 0$ if any $i_m = 0$ since every particle has non-zero properties and no particles with a zero-property can be formed by aggregation.
The unshifted and unscaled variant of this convolution sum can be efficiently evaluated by the methods we will present in section 3.4, scaling and shifting is trivial afterwards.

Remark 1. *The convolution of two functions each with support in* $[\mathbf{0}, \mathbf{v}^{max}]$ *lies in* $[\mathbf{0}, 2 \cdot \mathbf{v}^{max}]$ *as the maximal properties add up. We have chosen values* v_m^{max} *for the underlying grid* $\mathcal{G}$ *and discard any larger particles. Since we neglect breakage, all particles will hence eventually leave our computational domain, i.e. we lose mass over time.*
The choice of v_m^{max} *should reflect this growth of particles in the considered time frame and be adapted if necessary.*
One could enlarge the property space either during a simulation or restart the numerical experiments with a different grid. This enlargement can either be done by coarsening the grid (and keeping N *constant) or appending additional cells (increasing* N *in the process).*
A resulting particle can only leave the considered domain if one of the two aggregating particles has a property above $\frac{v_m^{max}}{2}$*. If this amount is negligible the resulting error is also small.*

We then compute

$$Q_{\text{pl}}^{\text{source}}(\mathbf{g_{i+1}}) = \sum_{\nu=1}^{k} Q_{\nu,\text{pl}}^{\text{source}}(\mathbf{g_{i+1}}) \tag{3.13}$$

to obtain the complete (piecewise linear) source term at gridpoints $\mathbf{g_{i+1}}$. This result is not compatible with the piecewise constant approach that is used for f and a suitable conversion (or projection) is necessary. One desired property of this projection $\mathcal{P}$ is the conservation of some moments $\mu^{\mathbf{e}}$ of the piecewise linear source term, i.e. we require

$$\begin{aligned} Q_{\text{pc}}^{\text{source}} &:= \mathcal{P}(Q_{\text{pl}}^{\text{source}}) \text{ with} \\ \mu^{\mathbf{e}}(Q_{\text{pl}}^{\text{source}}) &= \mu^{\mathbf{e}}(Q_{\text{pc}}^{\text{source}}) \end{aligned}$$

for some $\mathbf{e} \in \mathbb{N}^d$, where the subspript pc stands for *piecewise constant*.
We will present a general framework for this projection in subsection 3.3.3. A piecewise constant source term, projected in this way, is the final result of this procedure.

3.3.2 Sink term

The sink term is given by

$$Q^{\text{sink}}(\mathbf{v}) = f(\mathbf{v}) \cdot \int_0^\infty \cdots \int_0^\infty \kappa(\mathbf{u}, \mathbf{v}) f(\mathbf{u}) \, d\mathbf{u}$$

as we recall (1.4). With the assumption of a seperable kernel (see section 3.2), we defined (3.10) as

$$Q_\nu^{\text{sink}}(\mathbf{v}) = \psi^\nu(\mathbf{v}) \cdot \int_{[0,\infty]^d} \phi^\nu(\mathbf{u}) \, d\mathbf{u}$$

which is a simple integration and a scalar multiple since ψ^ν is piecewise constant.
The piecewise constant discretization of $f(\mathbf{v})$ and $\kappa(\mathbf{u}, \mathbf{v})$ allows for a rewrite of this integral into sums. For $\mathbf{v} \in \mathcal{C}_\mathbf{i}$ we then write

$$\begin{aligned} Q_\nu^{\text{sink}}(\mathbf{v}) &= V \cdot \psi_\mathbf{i}^\nu \underbrace{\sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n}-\mathbf{1}} \phi_\mathbf{j}^\nu}_{:=S_\nu} \\ &= V \cdot \psi_\mathbf{i}^\nu \cdot S_\nu \end{aligned}$$

with $S_\nu \in \mathbb{R}$ for one summand of the sink term. This function is already piecewise constant and does not require an additional projection. We then find the total sink term as

$$Q^{\text{sink}}(\mathbf{v}) = \sum_{\nu=1}^{k} Q_\nu^{\text{sink}}(\mathbf{v}) \tag{3.14}$$

which finishes this part.

3.3.3 Moments of a piecewise linear discretization

The piecewise linear convolution result

$$Q_{\text{pl}}^{\text{source}}(\mathbf{v}) := \sum_{\nu=1}^{k} Q_{\nu,\text{pl}}^{\text{source}}(\mathbf{v}) \tag{3.15}$$

is not compatible with the piecewise constant discretization for $f(\mathbf{v})$ and a projection is necessary. Similar to the idea presented in the FP or CAT methods from chapter 2 we are interested in a projection $\mathcal{P}$ that preserves some moments $\mu^\mathbf{e}(Q_{\text{pl}}^{\text{source}})$ of the source term. We will introduce a framework that allows for a preservation of moments based on an expression with basis functions. The following theorem shows the simple projection to preserve all moments $\mu^\mathbf{e}$ with $\mathbf{e} \in \{0,1\}^d$.

Theorem 1. *Let $Q_{pl}(\mathbf{v})$ be a continuous and piecewise linear function with respect to a uniform grid $\mathcal{G}$ and let $q_\mathbf{i} := Q_{pl}(\mathbf{g}_\mathbf{i})$ denote the function values at the gridpoints with $q_\mathbf{i} = 0$ at the boundary points, i.e. if $\exists m \in \{1, \dots, d\}$ with $i_m = 0$ or $i_m = n_m$. Then $W_{pc}(\mathbf{v})$ (piecewise constant with respect to $\mathcal{G}$) with cell values*

$$w_\mathbf{i} = \frac{1}{2^d} \cdot \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{1}} q_{\mathbf{i}+\mathbf{k}}$$

conserves all moments

$$\mu^\mathbf{e}(W_{pc}) = \mu^\mathbf{e}(Q_{pl})$$

for $\mathbf{e} \in \{0,1\}^d$.

Proof. The idea of the proof is as follows: We represent $Q_{\mathrm{pl}}(\mathbf{v})$ with respect to the standard basis of the space of continuous, piecewise linear functions and then consider the projection of a single basis function to the space of piecewise constant functions. Since its support consists of 2^d cells, we will be able to preserve 2^d moments.
With the notation for the evaluation at grid points $\mathbf{g_i}$ we write

$$Q_{\mathrm{pl}}(\mathbf{v}) = \sum_{\mathbf{i=1}}^{\mathbf{n}} q_{\mathbf{i}} \cdot \prod_{m=1}^{d} \Lambda_{i_m}(v_m) \tag{3.16}$$

with one dimensional basis functions

$$\Lambda_{i_m}(v_m) := \begin{cases} v_m/h_m - i_m + 1 & \text{, if } (i_m - 1) \cdot h_m \leq v_m \leq i_m \cdot h_m, \\ -v_m/h_m + i_m + 1 & \text{, if } i_m \cdot h_m \leq v_m \leq (i_m + 1) \cdot h_m, \\ 0 & \text{, else} \end{cases} \tag{3.17}$$

with support $[(i_m-1)h_m, (i_m+1)h_m]$ to be seen in Figure 3.2. These continuous

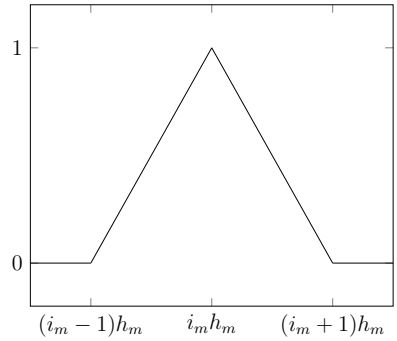


Figure 3.2: A piecewise linear basis function $\Lambda_{i_m}(\cdot)$ (3.17)

basis functions satisfy $\Lambda_{i_m}(i_m h_m) = 1$ and $\Lambda_{i_m}(i_m h_m \pm h_m) = 0$. As the

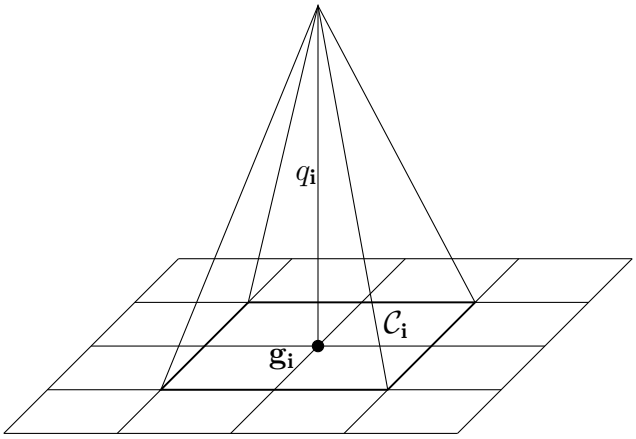


Figure 3.3: A multivariate piecewise linear function $T_{\mathbf{i}}^{\mathrm{pl}}(\cdot)$.

moments are linear in the argument, we concentrate on one summand $T_{\mathbf{i}}^{\mathrm{pl}}(\mathbf{v}) := q_{\mathbf{i}} \prod_{m=1}^{d} \Lambda_{i_m}(v_m)$ of (3.16) and distribute its contribution to the overall moments to the cells of its support. One function $T_{\mathbf{i}}^{\mathrm{pl}}(\mathbf{v})$ can be seen in Figure 3.3.
We calculate

$$\begin{aligned}\mu^{\mathbf{e}}(T_{\mathbf{i}}^{\mathrm{pl}}) &= \int_{\mathbf{0}}^{\infty} \mathbf{v}^{\mathbf{e}} \cdot T_{\mathbf{i}}^{\mathrm{pl}}(\mathbf{v}) \, d\mathbf{v} \\ &= q_{\mathbf{i}} \prod_{m=1}^{d} \int_{(i_m-1)h_m}^{(i_m+1)h_m} v_m^{e_m} \Lambda_{i_m}(v_m) \, dv_m = q_{\mathbf{i}} \prod_{m=1}^{d} \mathcal{I}_{m,e_m}^{\mathrm{pl},i_m}\end{aligned}$$

where we define

$$\mathcal{I}_{m,e_m}^{\mathrm{pl},i_m} := \int_{(i_m-1)h_m}^{(i_m+1)h_m} v_m^{e_m} \Lambda_{i_m}(v_m) \, dv_m = \begin{cases} h_m & \text{, if } e_m = 0 \\ h_m^2 i_m & \text{, if } e_m = 1 \end{cases} \tag{3.18}$$

to simplify the notation of the integral.
The vector of all moments $\mu^{\mathbf{e}}(T_{\mathbf{i}}^{\mathrm{pl}})$ with $\mathbf{e} \in \{0,1\}^d$ in lexicographical order can be obtained by the means of Kronecker products (here denoted by $\otimes$) via

$$\begin{bmatrix} \mathcal{I}_{1,0}^{\mathrm{pl},i_1} \\ \mathcal{I}_{1,1}^{\mathrm{pl},i_1} \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \mathcal{I}_{d,0}^{\mathrm{pl},i_d} \\ \mathcal{I}_{d,1}^{\mathrm{pl},i_d} \end{bmatrix} \cdot q_{\mathbf{i}}. \tag{3.19}$$

We will define

$$b_m := \begin{bmatrix} \mathcal{I}_{m,0}^{\mathrm{pl},i_m} \\ \mathcal{I}_{m,1}^{\mathrm{pl},i_m} \end{bmatrix} = \begin{bmatrix} h_m \\ h_m^2 i_m \end{bmatrix}$$

to refer a one of these Kronecker factors.
The support of $T_{\mathbf{i}}^{\mathrm{pl}}(\mathbf{v})$ is given by the 2^d cells $\mathcal{C}_{\mathbf{i}+\mathbf{k}}$ with $\mathbf{k} \in \{-1,0\}^d$ and we define a piecewise constant function $T_{\mathbf{i}}^{\mathrm{pc}}(\mathbf{v})$ with values $t_{\mathbf{i}+\mathbf{k}}$ at the cell $\mathcal{C}_{\mathbf{i}+\mathbf{k}}$. A possible function for $d = 2$ (with arbitrary values $t_{\mathbf{i}+\mathbf{k}}$) is shown in Figure 3.4.
The contribution of a value $t_{\mathbf{i}+\mathbf{k}}$ to the total moment is given by

$$\begin{aligned}\mu_{\mathcal{C}_{\mathbf{i}+\mathbf{k}}}^{\mathbf{e}}(t_{\mathbf{i}+\mathbf{k}}) &= \int_{\mathcal{C}_{\mathbf{i}+\mathbf{k}}} \mathbf{v}^{\mathbf{e}} t_{\mathbf{i}+\mathbf{k}} \, d\mathbf{v} \\ &= t_{\mathbf{i}+\mathbf{k}} \prod_{m=1}^{d} \int_{(i_m+k_m)h_m}^{(i_m+k_m+1)h_m} v_m^{e_m} \, dv_m = t_{\mathbf{i}+\mathbf{k}} \prod_{m=1}^{d} \mathcal{I}_{m,e_m}^{\mathrm{pc},i_m+k_m},\end{aligned}$$

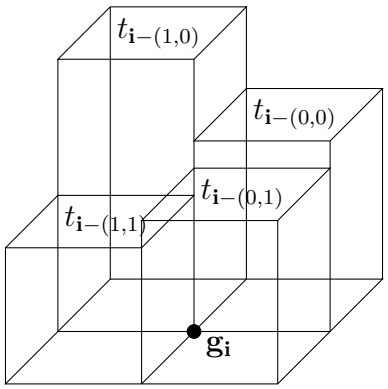


Figure 3.4: A multivariate piecewise constant function $T_{\mathbf{i}}^{\text{pc}}(\cdot)$ with values $t_{\mathbf{i}-\mathbf{k}}$.

where we computed

$$\mathcal{I}_{m,e_m}^{\text{pc},i_m} := \int_{(i_m)h_m}^{(i_m+1)h_m} v_m^{e_m}\, dv_m = \begin{cases} h_m & \text{, if } e_m = 0 \\ h_m^2 \cdot (i_m + 0.5) & \text{, if } e_m = 1 \end{cases} \tag{3.20}$$

to again simplify the notation of the integrals. We collect the assigned cell values $t_{\mathbf{i}+\mathbf{k}}$ in a vector

$$\mathbf{t} := \begin{bmatrix} t_{\mathbf{i}-[1,\ldots,1]} \\ t_{\mathbf{i}-[1,\ldots,0]} \\ \vdots \\ t_{\mathbf{i}-[0,\ldots,1]} \\ t_{\mathbf{i}-[0,\ldots,0]} \end{bmatrix} \in \mathbb{R}^{2^d}.$$

The moments of interest (in lexicographical order) of a function $T_{\mathbf{i}}^{\text{pc}}$ in all 2^d cells surrounding $\mathbf{g_i}$ can be expressed by a matrix-vector multiplication

$$G_1 \otimes \ldots \otimes G_d \cdot \mathbf{t} \tag{3.21}$$

with simple matrices

$$\begin{aligned} G_m &:= \begin{bmatrix} \mathcal{I}_{m,0}^{\text{pc},i_m-1} & \mathcal{I}_{m,0}^{\text{pc},i_m} \\ \mathcal{I}_{m,1}^{\text{pc},i_m-1} & \mathcal{I}_{m,1}^{\text{pc},i_m} \end{bmatrix} \\ &= \begin{bmatrix} h_m & h_m \\ h_m^2(i_m - 0.5) & h_m^2(i_m + 0.5) \end{bmatrix}. \end{aligned}$$

This gives a linear system defined by (3.21) and (3.19) as

$$G_1 \otimes \ldots \otimes G_d \cdot \mathbf{t} = b_1 \otimes \ldots \otimes b_m \cdot q_{\mathbf{i}} \tag{3.22}$$

implying a solution

$$\begin{aligned} \mathbf{t} &= v_1 \otimes \ldots \otimes v_d \cdot q_{\mathbf{i}} \\ &= \begin{bmatrix} t_{i_1-1} \\ t_{i_1} \end{bmatrix} \otimes \ldots \otimes \begin{bmatrix} t_{i_d-1} \\ t_{i_d} \end{bmatrix} q_{\mathbf{i}} \end{aligned}$$

with vectors

$$v_m := \begin{bmatrix} t_{i_m-1} \\ t_{i_m} \end{bmatrix} = G_m^{-1} \cdot b_m = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{3.23}$$

This gives

$$t_{\mathbf{i}+\mathbf{k}} = q_{\mathbf{i}} \cdot \prod_{m=1}^{d} t_{i_m+k_m} = \frac{q_{\mathbf{i}}}{2^d}$$

as value in every cell of the support of $T_{\mathbf{i}}^{\mathrm{pl}}$, i.e. the value at the grid point is equally distributed to the surrounding cells. This result is independent of the chosen grid point $\mathbf{g_i}$ and properties of the grid (h_m or V).
To know the value $w_{\mathbf{i}}$ of a piecewise constant function $W_{\mathrm{pc}}(\mathbf{v})$ in a cell $\mathcal{C}_{\mathbf{i}}$ we have to add up the contributions of the 2^d cornering grid-points $\mathbf{g_{i+k}}$ for $\mathbf{k} \in \{0,1\}^d$ around the cell which gives the final result of

$$w_{\mathbf{i}} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{1}} \frac{1}{2^d} q_{\mathbf{i}+\mathbf{k}}.$$

□

Remark 2. *This theorem is based on the projection of a single piecewise linear basis function onto its support of 2^d cells and offers 2^d degrees of freedom, allowing to preserve the same amount of moments here chosen to be all moments $\mu^{\mathbf{e}}$ with $\mathbf{e} \in \{0,1\}^d$. Other combinations of moments are possible and require the calculation of $\mathcal{I}_{m,e_m}^{pc,i_m}$ and $\mathcal{I}_{m,e_m}^{pl,i_m}$ for other values of e_m.*

Due to the restriction of the property space outlined in section 3.1 and the restriction of the convolution results pointed out in Remark 1 we generally do not fulfill the assumption of $Q^{pl}(\mathbf{g_i}) = 0$ for gridpoints on the upper boundary, i.e., a point $\mathbf{g_i}$ with any $i_m = n_m$. A function $T_{\mathbf{i}}^{pl}(\mathbf{v})$ on this boundary has support outside of the computational domain and so will the projected function $T_{\mathbf{i}}^{pc}(\mathbf{v})$. We chose to include these gridpoints in the projection step and discard any particles assigned beyond the boundary as neglecting their influence altogether will give a larger error in the conservation of moments.

3.4 Convolution via fast Fourier transformation

This section is devoted to the technique of (multivariate) fast Fourier transformation, its properties and application in this context.
Let $\widetilde{\phi}$ and $\widetilde{\psi}$ be tensors of size $\mathbb{R}^{2n_1 \times \cdots \times 2n_d}$ (we drop the superscript ν here)

where we used a process called *zero padding* to enlarge the tensors ϕ and ψ. Their full convolution result $\omega = \widetilde{\phi} * \widetilde{\psi}$ is given by entries

$$\omega_{\mathbf{j}} = \sum_{k_1=0}^{j_1} \cdots \sum_{k_d=0}^{j_d} \widetilde{\phi}_{\mathbf{k}} \widetilde{\psi}_{\mathbf{j}-\mathbf{k}} =: \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{j}} \widetilde{\phi}_{\mathbf{k}} \widetilde{\psi}_{\mathbf{j}-\mathbf{k}} \tag{3.24}$$

and is in $\mathbb{R}^{2n_1 \times \cdots \times 2n_d}$. We focus here on the unshifted and unscaled version of (3.12). This convolution can be calculated via Fourier transforms by using a multivariate formulation of the convolution theorem [53].

Remark 3. *By using a sequence of Fourier transforms we obtain a circular convolution, since the Fourier transform implicity performs a periodic continuation of the inputs with respect to all dimensions. The expansion of the inputs to* $\widetilde{\phi}, \widetilde{\psi} \in \mathbb{R}^{2n_1 \times \cdots \times 2n_d}$ *by adding zeros will lengthen the cycles of the periodic convolution to* $2\mathbf{n}$*, resulting in a linear convolution.*

This section will use $\mathrm{i} := \sqrt{-1}$ as the complex unit and not as an index for tensor entries. The symbols j and k will be used as indices.
The multivariate convolution theorem states that we can obtain $\widetilde{\omega} \in \mathbb{R}^{2n_1 \times \cdots \times 2n_d}$ by calculating

$$\widetilde{\omega} = \mathcal{F}^{-1}(\mathcal{F}(\widetilde{\phi}) \odot \mathcal{F}(\widetilde{\psi})). \tag{3.25}$$

In (3.25) we make use of the functions

$$\begin{aligned} \mathcal{F} \;:\; & \mathbb{R}^{2n_1 \times \cdots \times 2n_d} \to \mathbb{C}^{2n_1 \times \cdots \times 2n_d}, \quad \widetilde{\phi} \mapsto \mathcal{F}(\widetilde{\phi}) \qquad \text{with} \\ & (\mathcal{F}(\widetilde{\phi}))_{\mathbf{j}} := \sum_{\mathbf{s}=\mathbf{0}}^{\mathbf{2n-1}} \widetilde{\phi}_{\mathbf{s}} \cdot \prod_{m=1}^{d} e^{\mathrm{i}\pi s_m j_m / n_m} \end{aligned} \tag{3.26}$$

to denote the discrete Fourier transform, while

$$\begin{aligned} \mathcal{F}^{-1} \;:\; & \mathbb{C}^{2n_1 \times \cdots \times 2n_d} \to \mathbb{C}^{2n_1 \times \cdots \times 2n_d}, \quad \Omega \mapsto \mathcal{F}^{-1}(\Omega) \qquad \text{with} \\ & (\mathcal{F}^{-1}(\Omega))_{\mathbf{j}} := \frac{1}{2N} \sum_{\mathbf{s}=\mathbf{0}}^{\mathbf{2n-1}} \Omega_{\mathbf{s}} \cdot \prod_{m=1}^{d} e^{-\mathrm{i}\pi s_m j_m / n_m} \end{aligned} \tag{3.27}$$

denotes the inverse discrete Fourier transform. Finally, we use $\odot$ to denote the elementwise product of two tensors also known as *Hadamard product*.
We now focus on the efficient evaluation of (3.26) and derive the evaluation via a series of univariate discrete Fourier transforms. The evaluation of (3.27)

only differs in the signs in the exponent and the scaling by a constant factor. By taking a look at the definition of $\mathcal{F}$, we can rewrite it in the form of

$$\begin{aligned}\mathcal{F}(\widetilde{\phi})_j &= \sum_{\mathbf{s}=\mathbf{0}}^{2\mathbf{n}-1} \widetilde{\phi}_{\mathbf{s}} \cdot \prod_{m=1}^{d} e^{i\pi s_m j_m / n_m} \\ &= \sum_{s_d=0}^{2n_d-1} \cdots \left(\sum_{s_1=0}^{2n_1-1} \widetilde{\phi}_{\mathbf{s}} \cdot e^{\mathrm{i}\pi s_1 j_1 / n_1} \right) \cdots e^{\mathrm{i}\pi s_d j_d / n_d} \\ &= \left(\mathcal{F}^d \circ \mathcal{F}^{d-1} \circ \cdots \circ \mathcal{F}^1(\widetilde{\phi}) \right)_j, \end{aligned} \tag{3.28}$$

where we use

$$\begin{aligned}\mathcal{F}^m \quad : \quad & \mathbb{C}^{2n_1 \times \cdots \times 2n_d} \to \mathbb{C}^{2n_1 \times \cdots \times 2n_d}, \quad \widetilde{\phi} \mapsto \mathcal{F}^m(\widetilde{\phi}) \qquad \text{with} \\ & (\mathcal{F}^m(\widetilde{\phi}))_{\mathbf{j}} := \sum_{s=0}^{2n_m-1} \widetilde{\phi}_{j_1,\ldots,j_{m-1},s,j_{m+1},\ldots,j_d} \cdot e^{\mathrm{i}\pi s j_m / n_m} \end{aligned} \tag{3.29}$$

to denote the univariate discrete Fourier transformation of a d-dimensional tensor with respect to the m-th dimension. The compositon $\mathcal{F}^d \circ \cdots \circ \mathcal{F}^1(\widetilde{\phi})$ of these transformations with m from 1 to d will yield a complete discrete Fourier transformation.
Three properties of the discrete Fourier transform and complex arithmetic guarantee $\widetilde{\omega}$ to only have real entries. These are stated in the following lemma

Lemma 1. *a) The discrete Fourier transform $\Phi := \mathcal{F}(\widetilde{\phi}) \in \mathbb{C}^{2n_1 \times \ldots \times 2n_d}$ of $\widetilde{\phi} \in \mathbb{R}^{2n_1 \times \ldots \times 2n_d}$ shows complex-conjugate symmetry, i.e., $\overline{\Phi_{\mathbf{j}}} = \Phi_{\mathbf{2n-j}}$.*
b) The elementwise product of two tensors $\Omega = \Phi \odot \Psi \in \mathbb{C}^{2n_1 \times \ldots \times 2n_d}$ with complex-conjugate-symmetry again holds complex conjugate symmetry.
c) The inverse Fourier transform $\widetilde{\omega} = \mathcal{F}^{-1}(\Omega)$ of a complex-conjugate-symmetric tensor $\Omega \in \mathbb{C}^{2n_1 \times \ldots \times 2n_d}$ is real, i.e. $\widetilde{\omega} \in \mathbb{R}^{2n_1 \times \ldots \times 2n_d}$.

Proof. The propositions a) and c) are essentially equivalent. We only show a) as c) can be proven in the same manner. We observe the exponents in the evaluation of

$$\Phi_{\mathbf{2n-j}} = \sum_{\mathbf{s}=\mathbf{0}}^{2\mathbf{n}-1} \widetilde{\phi}_{\mathbf{s}} \prod_{m=1}^{d} e^{i\pi s_m (2n_m - j_m)/n_m}$$

and find

$$\begin{aligned}\Phi_{\mathbf{2n-j}} &= \sum_{\mathbf{s}=\mathbf{0}}^{2\mathbf{n}-1} \widetilde{\phi}_{\mathbf{s}} \prod_{m=1}^{d} e^{\mathrm{i}\pi s_m (2n_m - j_m)/n_m} \\ &= \sum_{\mathbf{s}=\mathbf{0}}^{2\mathbf{n}-1} \widetilde{\phi}_{\mathbf{s}} \prod_{m=1}^{d} e^{-\mathrm{i}\pi s_m j_m / n_m} e^{2\pi \mathrm{i} s_m} = \overline{\Phi_{\mathbf{j}}} \end{aligned}$$

by using the identity of $e^{2\pi \mathrm{i} s_m} = 1$.
To prove the proposition b), we simply show

$$\begin{aligned} \Omega_{2\mathbf{n}-\mathbf{j}} &= \Phi_{2\mathbf{n}-\mathbf{j}} \cdot \Psi_{2\mathbf{n}-\mathbf{j}} \\ &= \overline{\Phi_{\mathbf{j}}} \cdot \overline{\Psi_{\mathbf{j}}} = \overline{\Omega_{\mathbf{j}}} \end{aligned}$$

□

A one-dimensional discrete Fourier transform $\mathcal{F}^m$ can be calculated via the FFT algorithm (first introduced in [16]) in $\mathcal{O}(n_m \log(n_m))$ complexity, instead of the naive approach of $\mathcal{O}(n_m^2)$. This brings down the total complexity of the convolution from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log(N))$. The complex-conjugate symmetry allows us to reduce in the constant in the complexity without affecting the order of complexity.
After computing the full convolution result $\widetilde{\omega} \in \mathbb{R}^{2n_1 \times \cdots \times 2n_d}$ we restrict the result back to $\mathbb{R}^N$ to obtain the result ω of the convolution within the considered domain of $\mathcal{G}$.

3.4.1 Multi-dimensional Fourier transform of a tensor with zero-padding

We describe in this subsection how the special structure of the zero-padded input tensors $\widetilde{\phi}$ and $\widetilde{\psi}$ can be exploited to further decrease the complexity of the multivariate Fourier transform. It is based on the zero-padding that is necessary for a linear convolution result.
The d-dimensional discrete Fourier transform requires one-dimensional Fourier transforms along every dimension of the tensor (see (3.28)). We define intermediate results

$$\mathcal{F}^{1,m}(\widetilde{\phi}) := \mathcal{F}^m \circ \mathcal{F}^{m-1} \circ \ldots \circ \mathcal{F}^1(\widetilde{\phi}), \quad m \in \{1, \cdots, d\}$$

with $\mathcal{F}^{1,d}(\widetilde{\phi}) = \mathcal{F}(\widetilde{\phi}) = \Phi$ being a complete Fourier transform of $\widetilde{\phi}$.
The input tensor $\widetilde{\phi}$ consists mostly of zeros due to the padding in every internal coordinate, as at most n^d of the $(2n)^d$ entries holds a non-zero entry and their pattern can be exploited.
We recall the first one-dimensional Fourier transform $\mathcal{F}^1$ from (3.29) as a weighted summation with all indices fixed but the first. By the definition of $\widetilde{\phi}$, all of these summands are zero if $i_m > n_m$ holds for any m, rendering the calculation superfluous by the linearity of the Fourier transform because the result is the vector of only zeroes. By only calculating a Fourier transform for multi-indices $\mathbf{j}$ with $j_2 < n_2, \ldots, j_d < n_d$ we reduce the number of necessary calculations by a factor of $(1/2)^{d-1}$.
This idea can similary be adapted by observing the resulting zero-pattern in $\mathcal{F}^{1,m}(\widetilde{\phi})$. We have $\mathcal{F}^{1,m}(\widetilde{\phi})_{\mathbf{j}} = 0$, if $\exists k > m : j_k > n_k$. This directly implies a reduction of the required Fourier transforms by $(1/2)^{d-m}$. Only the last stage

of the transformation requires a consideration of the complete tensor. An illustration of the non-zero pattern of $\widetilde{\phi}$ and $\mathcal{F}^{1,m}(\widetilde{\phi})$ for $d = 3$ can be seen in Figure 3.5.

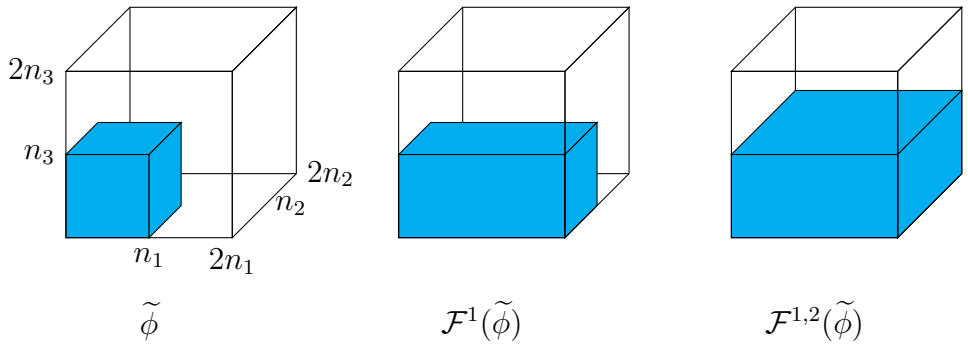


Figure 3.5: Illustration of the non-zero-pattern of $\widetilde{\phi}$ and the intermediate results of its Fourier transformation $\mathcal{F}(\widetilde{\phi})$.

The exploitation of the zero-pattern in this fashion reduces the complexity considerably. If we assume a symmetric grid as defined in section 3.1, i.e., all n_m to be equal ($n_1 = \cdots = n_d = n$), the total number of one-dimensional Fourier transformations amounts to

$$n^{d-1} + \ldots + 2^{m-1}n^{d-1} + \ldots + 2^{d-1}n^{d-1} = (2^d - 1)n^{d-1}$$

instead of $2^{d-1}n^{d-1}d$ without exploiting the zeros.

A similar idea can be followed in the calculation of the inverse Fourier transformation $\mathcal{F}^{-1}(\Omega)$. In this we have a tensor $\Omega \in \mathbb{C}^{2n_1 \times \ldots \times 2n_d}$ that is completely filled with non-zero entries as input and let $\mathcal{F}^{-1}(\Omega) =: \widetilde{\omega} \in \mathbb{R}^{2n_1 \times \ldots \times 2n_d}$ be the result of the inverse Fourier transform. We restricted the computational domain and so are only interested in $\omega \in \mathbb{R}^{n_1 \times \ldots \times n_d}$, a small part of $\widetilde{\omega}$ effectively assuming all other entries to be zero.
This assumption then gives the zero-pattern illustrated in Figure 3.5 for intermediates of the inverse Fourier transform. The non-shaded entries (we assume to be close to zero in abolute value) can be set to zero ealier as all influenced entries of $\widetilde{\omega}$ are non-shaded and will be set to zero as well. This allows us to skip the inverse FFT-calculations. This (similarly to the exploitation of non-zero entries in the forward transformation) reduces to the number of one-dimensional inverse Fourier transforms necessary to $(2^d - 1)n^{d-1}$ if we assume all n_m to be equal.

3.5 Numerical results

In this section, we will present numerical results of the methodology presented in this chapter and compare them with results obtained by the FP and CAT

Processor	Intel Xeon E5-2690 @ 2.60 GHz
RAM	16 GB
OS	Windows 10 Pro, 64 bit

Table 3.1: Specific hardware of the used computer.

methods (see sections 2.2 and 2.3). This section is divided into three separate numerical experiments, each highlighting an important property of the ansatz. These properties are the accuracy of a numerical density distribution $f(\mathbf{v}, t)$, the tracking of the moments $\mu^{\mathbf{e}}(f)(t)$ and the computational complexity.
The uniform grid $\mathcal{G}$ for a piecewise constant approximation (referred to as PC) will always be defined by $\mathbf{v}^{\max}$ and $\mathbf{n}_{\text{uniform}}$. Any simulation with the FP-scheme or the CAT will be done with respect to a geometric tensor grid. This will be given by the number of pivots $\mathbf{n}_{\text{geometric}}$ and a minimal and maximal grid line (g_0 and $g_n = v^{\max}$ respectively). The grid lines will be given by $g_i = \tau \cdot g_{i-1}$ with $\tau = \left(\frac{g_n}{g_0}\right)^{1/n}$ giving us n pivots with pivots $v_i = \frac{1}{2} \cdot (g_i - g_{i+1})$.
All simulations use the explicit Euler scheme to discretize the time derivative of the PBE with a constant time step denoted by Δt.
All simulations in this section are calculated on an office desktop computer with specifics given in Table 3.1.

3.5.1 Accuracy of a population density approximation

To assess the accuracy of our proposed discretization scheme we are going to compare a numerical result of an aggregation process with an analytical solution to (1.2). As we mentioned in section 2.1 a closed form solution can only be given for certain choices of initial distribution $f(\mathbf{v}, 0)$ and kernel $\kappa(\mathbf{u}, \mathbf{v})$. We here use the initial distribution

$$f(\mathbf{v}, 0) = N_0 \cdot \prod_{m=1}^{d} \frac{4v_m}{\xi_m^2} \cdot e^{\frac{-2v_m}{\xi_m}} \tag{3.30}$$

(with constants N_0 and ξ_m) and the multi-dimensional sum-kernel

$$\kappa_\Sigma(\mathbf{u}, \mathbf{v}) = \sum_{m=1}^{d} u_m + v_m. \tag{3.31}$$

The analytical solution was given in [20] as

$$\begin{aligned} f(\mathbf{v}, t) = N_0(1-\tau)e^{\frac{-v_\Sigma \tau}{N_0 \xi_\Sigma}} \cdot \left(\prod_{m=1}^{d} \frac{2}{\xi_m} e^{-\frac{2v_m}{\xi_m}} \right) \\ \sum_{k=0}^{\infty} \frac{1}{(k+1)!} \left(\frac{\tau v_\Sigma}{\xi_\Sigma} \right)^k \cdot \prod_{m=1}^{d} \frac{1}{(2k+1)!} \cdot \left(\frac{2v_m}{\xi_m} \right)^{2k+1} \end{aligned} \tag{3.32}$$

where we make use of the definitions

$$\xi_\Sigma := \sum_{m=1}^{d} \xi_m, \quad v_\Sigma := \sum_{m=1}^{d} v_m, \quad \tau := 1 - e^{-N_0 \xi_\Sigma t}.$$

We here simulate an asymmetric two dimensional problem with

$$N_0 = 1,\ \xi_1 = 0.1 \text{ and } \xi_2 = 0.15$$

and choose $\mathbf{v}^{\max}$ equal for a uniform and a geometrical grid while varying the degrees of freedom N.
To assess the error of a given discrete approximation $\widetilde{f}(\mathbf{v}, t)$ to $f(\mathbf{v}, t)$ we calculate the L_2 error

$$E(t) = \left(\int_0^{v_1^{\max}} \int_0^{v_2^{\max}} \left(\widetilde{f}(\mathbf{v}, t) - f(\mathbf{v}, t) \right)^2 \right) dv_2 dv_1)^{1/2}. \tag{3.33}$$

A macroscopic representation

$$N(\mathbf{v}, t) = \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n-1}} N_{\mathbf{i}} \cdot \delta(\mathbf{v} - \mathbf{v}_{\mathbf{i}})$$

used on a geometric grid (recall (2.2)) will be projected to a piecewise constant function by equally distributing $N_{\mathbf{i}}$ uniformly onto the associated gridcell. This is the simplest method to project the macroscopic representation $N_{\mathbf{i}}$ (formed by of (2.1)) to a piecewise constant representation both in terms of implementation and the conservation of the total number of particles.
We note that (3.33) differs from measures used in many other works (e.g. [12] compares macroscopic variables on the diagonal, [22] computes discrete L_1-errors, [40] sorts indices by macroscopic representations of the analytical solution) as we compare against the smooth analytical solution instead of discretizing it. This yields a possibly larger error. We approximate (3.33) by using 11×11 uniformly distributed points in every cell and the bivariate trapezoidal rule. We use 121 points regardless of the size of the cell to evaluate errors we assume to be smaller (due to a smaller cell) with increased accuracy.

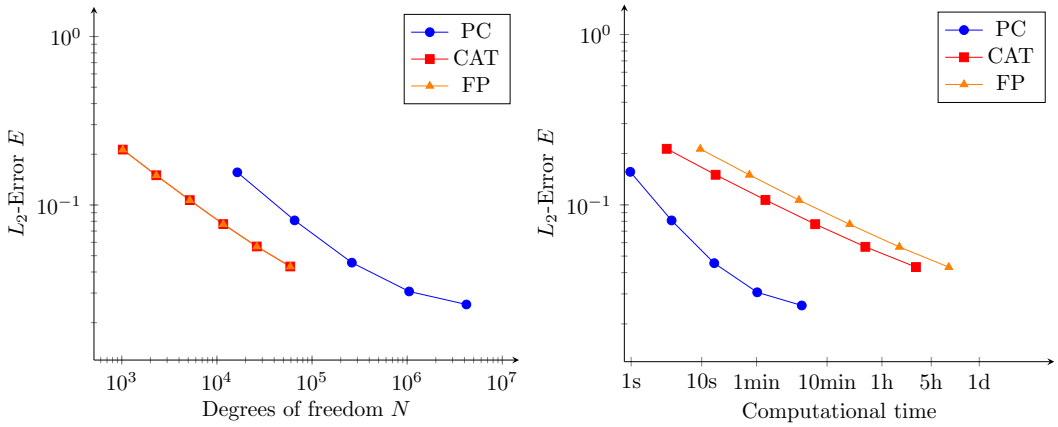
We are going to solve two problems over a different timespan T and choose a domain accordingly. The first problem will feature a short simulation with $T = 1$ and a small computational domain with $v_1^{\max} = v_2^{\max} = 1$ for both uniform and geometric grid. For a uniform grid we will vary a symmetric grid with

$$n_{\text{uniform}} \in \{128, 256, 512, 1024, 2048\},$$

the geometric grid will feature

$$n_{\text{geometric}} \in \{32, 48, 72, 108, 162, 243\}$$

with $g_0 = \frac{0.1 \cdot v^{\max}}{n_{\text{geometric}}}$ and $g_n = v^{\max}$. Both discretizations will feature $\Delta t = 0.1$.
In Figure 3.6a we show the error $E(1)$ (3.33) with respect to the degrees of freedom N. The geometric grid allows for a smaller error with a given N, as the important part of the domain (i.e. smaller particles) are represented with smaller cells to better capture the distribution. We see a near perfect overlap of the plots for the CAT and FP-approach. We furthermore see the error not decreasing beyond approximately $E \approx 3 \cdot 10^{-2}$ which we are able to attribute to the limited computational domain as particles grow outside our set boundaries.
Figure 3.6b shows the necessary computational time to calculate $\widetilde{f}(\mathbf{v}, 1)$ with the resulting error E when varying the degrees of freedom. Here we see a clear advantage of our proposed method, allowing a simulation to the best possible accuracy in about 4.5 minutes while the longest simulation using CAT took over 2.6 hours without reaching this accuracy.

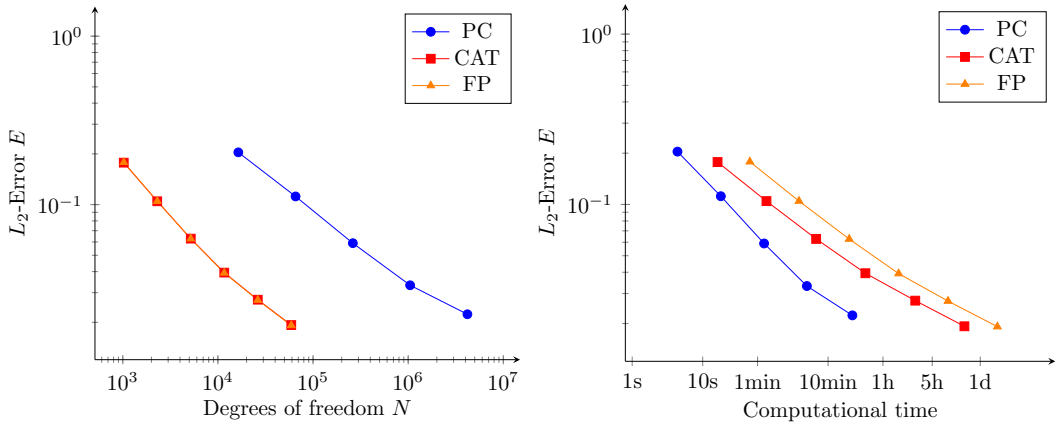


(a) (b)

Figure 3.6: L_2-error $E(1)$ (3.33) with respect to degrees of freedom N (left) and computational time (right) for a short simulation. The plots for CAT and FP-approach overlap on the left hand side.

The second problem will feature a longer time span of $T = 5$ and a larger computational domain with $v_1^{\max} = v_2^{\max} = 5$. We again choose a symmetric grid with $n_{\text{uniform}} \in \{128, 256, 512, 1024, 2048\}$ and $n_{\text{geometric}} \in \{32, 48, 72, 108, 162, 243\}$. We again use $g_0 = \frac{0.1 \cdot v^{\max}}{n_{\text{geometric}}}$ and time steps of $\Delta t = 0.1$.
We show the L_2 error $E(5)$ (3.33) with respect to the degrees of freedom N in Figure 3.7a and see the advantage of the geometric grid even more pronounced for a larger computational domain. The decrease in error when adding additional cells to a geometric grid is greater compared to the addition of more cells in a uniform grid.
Figure 3.7b compares the error with the computational time necessary to calculate $\widetilde{f}(\mathbf{v}, 5)$. We again see a benefit of using the uniform grid as the necessary

computational time for a given accuracy is lower compared to both techniques relying on a geometric grid (18 seconds compared to 80 and 240 seconds to reach $E \approx 10^{-1}$). This benefit is not as clear as for the short simulation as the plots are closer together.
We assume the benefit of the uniform grid to decrease as the computational domain grows larger (i.e. further increasing $\mathbf{v}^{\max}$) as the geometric grid is better suited to discretize wide spread distributions of very small and very large particles naturally occurring in longer simulations due to the absence of breakage or depletion. If the computational domain spans several orders of magnitude we expect an advantage of a geometric grid because the domain can be chosen very large while still using small cells for smaller particles.



(a) (b)

Figure 3.7: L_2-error E (3.33) with respect to degrees of freedom N (left) and computational time (right) for a long simulation $T = 5$. The plots for CAT and FP-approach overlap on the left hand side.

3.5.2 Tracking moments of a population balance equation

In section 1.1 we introduced the moments $\mu^{\mathbf{e}}(f)(t)$ of a distribution $f(\mathbf{v}, t)$ that are of interest in the numerical handling of PBEs. We recall them to be given by

$$\mu^{\mathbf{e}}(f)(t) := \int_{\mathbf{0}}^{\infty} \mathbf{v}^{\mathbf{e}} f(\mathbf{v}, t)\, d\mathbf{v}.$$

We can determine the temporal change in a moment by

$$\begin{aligned}\frac{d\mu^{\mathbf{e}}(f)(t)}{dt} &= \int_{\mathbf{0}}^{\infty} \mathbf{v}^{\mathbf{e}} \cdot \frac{df(\mathbf{v},t)}{dt}\, d\mathbf{v} \\ &= \int_{\mathbf{0}}^{\infty} \mathbf{v}^{\mathbf{e}} \cdot \left(Q^{\text{source}}(f) - Q^{\text{sink}}(f)\right)\, d\mathbf{v}.\end{aligned}$$

An expansion of the shorthand notation gives

$$\begin{aligned}\frac{d\mu^{\mathbf{e}}(f)(t)}{dt} = \frac{1}{2} &\int_{\mathbf{0}}^{\infty} \mathbf{v}^{\mathbf{e}} \cdot \int_{\mathbf{0}}^{\mathbf{v}} \kappa(\mathbf{u}, \mathbf{v}-\mathbf{u}) f(\mathbf{v}-\mathbf{u}) f(\mathbf{u})\, d\mathbf{u}\, d\mathbf{v} \\ &- \int_{\mathbf{0}}^{\infty} \mathbf{v}^{\mathbf{e}} \cdot f(\mathbf{v}) \int_{\mathbf{0}}^{\infty} \kappa(\mathbf{u}, \mathbf{v}) f(\mathbf{u})\, d\mathbf{u}\, d\mathbf{v}\end{aligned}$$

and in this subsection we will use a constant kernel

$$\kappa_C(\mathbf{u}, \mathbf{v}) = 1$$

to eliminate it from the equation. We have $f(\mathbf{v}-\mathbf{u}) = 0$ if any $v_m < u_m$ as there are no particles with a negative property. That allows us to extend the upper limit in the inner integral (with respect to $\mathbf{u}$) of the source term to infinity. The equation then reads

$$\begin{aligned}\frac{d\mu^{\mathbf{e}}(f)(t)}{dt} = \frac{1}{2} &\int_{\mathbf{0}}^{\infty} \mathbf{v}^{\mathbf{e}} \cdot \int_{\mathbf{0}}^{\infty} f(\mathbf{v}-\mathbf{u}) f(\mathbf{u})\, d\mathbf{u}\, d\mathbf{v} \\ &- \int_{\mathbf{0}}^{\infty} \mathbf{v}^{\mathbf{e}} \cdot f(\mathbf{v}) \int_{\mathbf{0}}^{\infty} f(\mathbf{u})\, d\mathbf{u}\, d\mathbf{v}\end{aligned}$$

which we further simplify by a substitution of $\mathbf{w} = \mathbf{v} - \mathbf{u}$ in the first line and a separation of integrals in the second. This gives

$$\begin{aligned}\frac{d\mu^{\mathbf{e}}(f)(t)}{dt} = \frac{1}{2} &\int_{\mathbf{0}}^{\infty} \int_{\mathbf{0}}^{\infty} (\mathbf{u}+\mathbf{w})^{\mathbf{e}} f(\mathbf{u}) f(\mathbf{w})\, d\mathbf{u}\, d\mathbf{w} \\ &- \int_{\mathbf{0}}^{\infty} \mathbf{v}^{\mathbf{e}} f(\mathbf{v})\, d\mathbf{v} \cdot \int_{\mathbf{0}}^{\infty} f(\mathbf{u})\, d\mathbf{u}\end{aligned}$$

where we expand the multivariate binomial. This gives

$$\frac{d\mu^{\mathbf{e}}(f)(t)}{dt} = \frac{1}{2}\int_{\mathbf{0}}^{\infty}\int_{\mathbf{0}}^{\infty}\sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{e}}\mathbf{u}^{\mathbf{k}}\cdot f(\mathbf{u})\cdot\mathbf{w}^{\mathbf{e}-\mathbf{k}}f(\mathbf{w})\cdot\prod_{m=1}^{d}\binom{e_m}{k_m}\,d\mathbf{u}\,d\mathbf{w}$$
$$-\int_{\mathbf{0}}^{\infty}\mathbf{v}^{\mathbf{e}}f(\mathbf{v})\,d\mathbf{v}\cdot\int_{\mathbf{0}}^{\infty}f(\mathbf{u})\,d\mathbf{u}.$$

We change the order of the summations and integrations in the first part and separate integrals with respect to $\mathbf{u}$ and $\mathbf{w}$. This reads

$$\frac{d\mu^{\mathbf{e}}(f)(t)}{dt} = \frac{1}{2}\sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{e}}\prod_{m=1}^{d}\binom{e_m}{k_m}\int_{\mathbf{0}}^{\infty}\mathbf{u}^{\mathbf{k}}\cdot f(\mathbf{u})\,d\mathbf{u}\cdot\int_{\mathbf{0}}^{\infty}\mathbf{w}^{\mathbf{e}-\mathbf{k}}f(\mathbf{w})\cdot\,d\mathbf{w}$$
$$-\int_{\mathbf{0}}^{\infty}\mathbf{v}^{\mathbf{e}}f(\mathbf{v})\,d\mathbf{v}\cdot\int_{\mathbf{0}}^{\infty}f(\mathbf{u})\,d\mathbf{u}.$$

We then find every integral to be of the form (1.5). We use this definition and find

$$\frac{d\mu^{\mathbf{e}}(f)(t)}{dt} = \frac{1}{2}\sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{e}}\mu^{\mathbf{k}}(f)(t)\cdot\mu^{\mathbf{e}-\mathbf{k}}(f)(t)\cdot\prod_{m=1}^{d}\binom{e_m}{k_m} - \mu^{\mathbf{e}}(f)(t)\cdot\mu^{\mathbf{0}}(f)(t). \tag{3.34}$$

This is an ODE for any moment based on moments of lower order. Most importantly, the change in any moment is independent of the particle distribution $f(\mathbf{v},t)$, whether it is discrete or continuous.
A comparison with the moments of an analytical solution (like (3.32)) will always contain (and most likely be dominated by) the discretization error. We here want to focus on the error made during the projection step presented in subsection 3.3.3 and observe the evolution of the moments of a discrete population density $f(\mathbf{v},t)$.

For this experiment, we use a discretized initial distribution of

$$f(\mathbf{v},0) = N_0\cdot\exp\left(\sum_{m=1}^{d}c_m\cdot(v_m-s_m)^2\right) \tag{3.35}$$

which is a multivariate Gaussian bell curve. We have chosen two dimensions ($d=2$) and use constants $c_1=c_2=-200$ and $s_1=s_2=0.1$. The scalar constant N_0 will be chosen to set $\mu^{\mathbf{0}}(f)(0)=0.1$ for each of the three discretization schemes individually.
We define the degree of aggregation

$$I_{\text{agg}}(t) = 1-\frac{\mu^{\mathbf{0}}(f)(t)}{\mu^{\mathbf{0}}(f)(0)} \tag{3.36}$$

as the fraction of particles that has already disappeared over time. We will run a simulation of the particle distribution for $T = 50$ seconds which results in $I_{\text{agg}}(50) \approx 0.7143$. A property space of $v_1^{\max} = v_2^{\max} = 10$ is sufficiently large for the resulting distribution to not have be significantly influencend by the bounded computational domain.
We choose $\mathbf{n}_{\text{uniform}} = (512, 512)$ and $\mathbf{n}_{\text{geometric}} = (51, 51)$ and time steps of $\Delta t = 0.1$ resulting in 500 time steps for these simulations. We chose a coarser geometric grid to make up for the longer computational time. The simulation with a uniform grid took around 18 minutes approximately the same time used by the CAT. The FP simulation took about 55 minutes.
We calculate all moments of the numerical solution over the course of the simulation for

$$\mathbf{e} \in \{0, \dots, 3\} \times \{0, \dots, 3\}$$

with a precision of 10^{-10} (to not let rounding error in the simulation dominate here) and denote them by $\widetilde{\mu}^{\mathbf{e}}_{\text{PC}}(f)(t)$, $\widetilde{\mu}^{\mathbf{e}}_{\text{CAT}}(f)(t)$ and $\widetilde{\mu}^{\mathbf{e}}_{\text{FP}}(f)(t)$ for piecewise constant, Cell Average and Fixed Pivot respectively. We define the relative error at each point in time

$$A^{\mathbf{e}}_X(t) := \frac{\widetilde{\mu}^{\mathbf{e}}_X(f)(t)}{\mu^{\mathbf{e}}(f)(t)} - 1 \tag{3.37}$$

for all three schemes, where we calculate $\mu^{\mathbf{e}}(f)(t)$ by solving the IVP given by (3.34) and $\mu^{\mathbf{e}}(f)(0) = \widetilde{\mu}^{\mathbf{e}}(f)(0)$ with the same constant time steps. This eliminates the influence of time-discretization from the consideration.
We show a selection of 4 of these 16 errors in Figure 3.8.

We see an interesting pattern in the plots in the top row. All three schemes mathematically guarantee conservation of the total numbers of particles and we see $A^{(0,0)}$ of less than 10^{-10}, the accuracy of our computations of $\widetilde{\mu}^{\mathbf{e}}(t)$. The accuracy of the PC-approach is even greater and we see the same for the FP and our PC approach for the first cross moment $\mu^{(1,1)}(t)$ in the top right. The CAT loses accuracy with this moment which can be seen here.
The plots in the bottom row ($A^{(2,2)}(t)$ on the left and $A^{(3,3)}(t)$ on the right) look similar in that the FP scheme shows the largest discrepancy from the beginning of the simulation. The CAT shows its superior accuracy for high order moments but throughout the simulation, the PC approach gets better and we observe an interesting self-correcting behavior not present in the other two schemes. We attribute the later superiority of our approach to the higher number of degrees of freedom and smaller cells used to discretize larger particles.

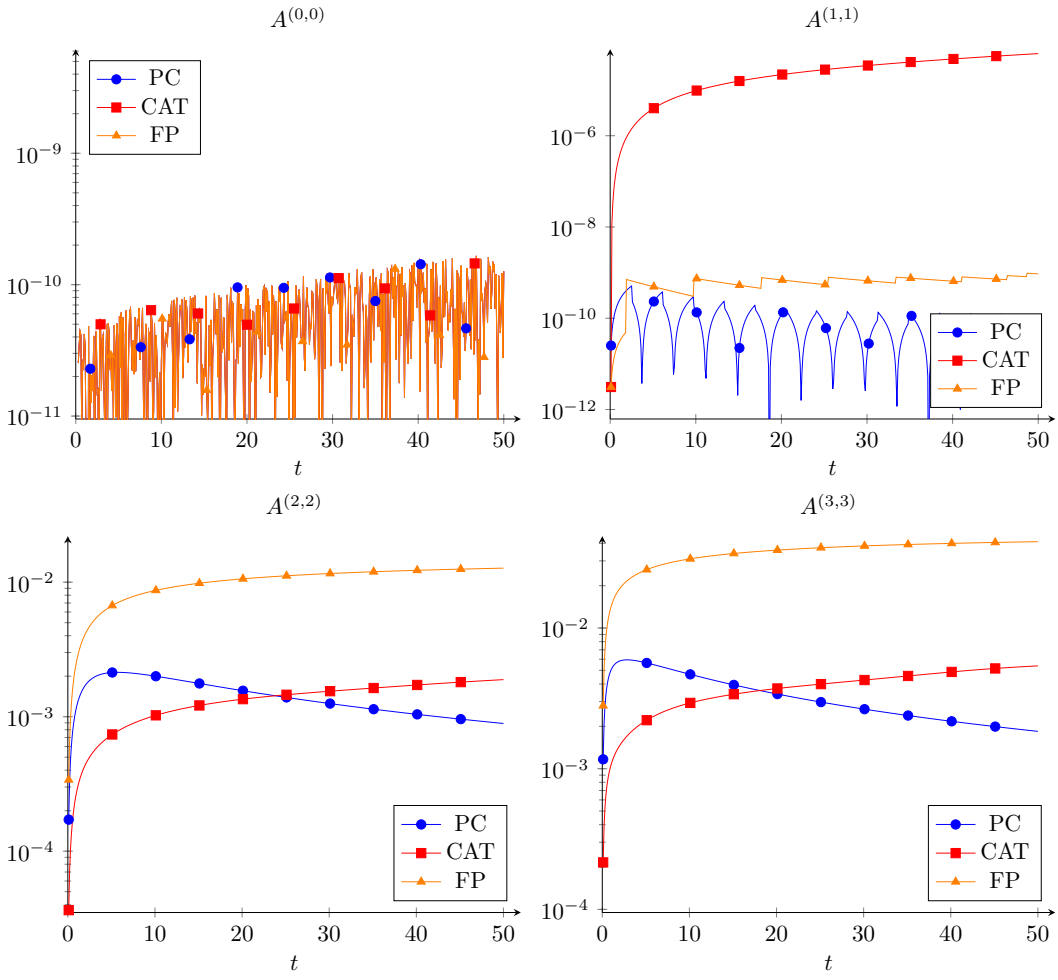


Figure 3.8: Relative error in the moments of a numerical simulation for a selection of moments, namely $A^{(0,0)}$ (top left), $A^{(1,1)}$ (top right), $A^{(2,2)}$ (bottom left), $A^{(3,3)}$ (bottom right).

3.5.3 Estimating the computational complexity

The third numerical test pertains to the complexity of the proposed algorithm and compares it with the complexity of the FP method and the CAT. We do this by simulating an aggregation process with $d = 2$ and $d = 3$ dimensions for varying degrees of freedom n_{uniform} and $n_{\text{geometric}}$. For this, we use the exponentially decreasing initial distribution

$$f(\mathbf{v}, 0) = \prod_{m=1}^{d} e^{-0.05 v_m}$$

and use the multivariate Brownian kernel

$$\kappa_B(\mathbf{u}, \mathbf{v}) = \sum_{m=1}^{d} \left(u_m^{1/3} + v_m^{1/3}\right) \cdot \left(u_m^{-1/3} + v_m^{-1/3}\right)$$

with a rank of $k = 2d + 1$.
We use a time step $\Delta t = 0.2$ and a final time of $T = 5$. This results in 25 time steps. For the simulation with $d = 2$, we use

$$\begin{aligned} n_{\text{uniform}} &\in \{32, 64, 128, 256, 512, 1024, 2048\} \\ &\text{and} \\ n_{\text{geometric}} &\in \{32, 39, 48, 59, 73, 90, 111, 137, 168, 207\}. \end{aligned}$$

We are not interested in the final density distribution $f(\mathbf{v}, 5)$ and only record the computational times for these simulations. We show these times for $d = 2$ in Figure 3.9.

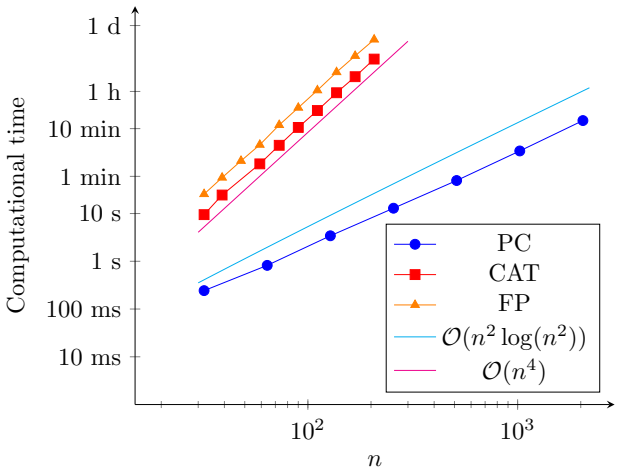


Figure 3.9: Computational time for different numerical schemes with respect to n for $d = 2$.

We see a clear advantage of our proposed method as it only takes approximately 15 minutes for a grid with $N = 2^{22} \approx 4.2 \cdot 10^6$ degrees of freedom. The

simulation with CAT takes just shy of 4 hours and 45 minutes for a grid with $N = 207^2 \approx 42 \cdot 10^3$ degrees of freedom. We also can confirm the theoretical estimates of complexity by including reference lines with the expected orders of complexity.
We run a similar simulation with three properties ($d = 3$) for

$$n_{\text{uniform}} \in \{32, 64, 128, 256\}$$
$$\text{and}$$
$$n_{\text{geometric}} \in \{32, 39\}$$

and show the computational time in Figure 3.10 and include reference lines for the expected orders of complexity.

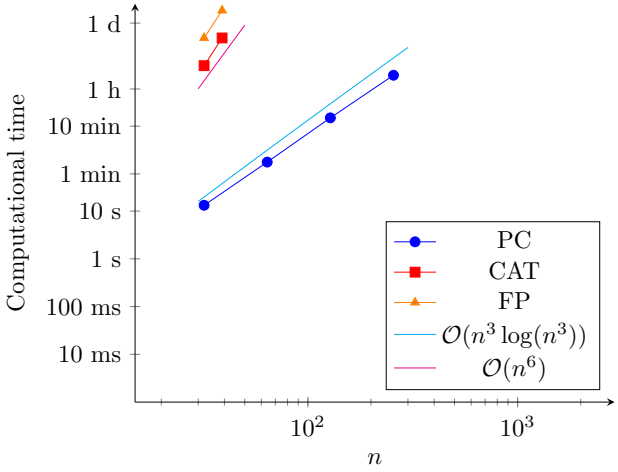


Figure 3.10: Computational time for different numerical schemes with respect to n for $d = 3$.

A simulation with only $n_{\text{geometric}} = 39$ took over 44 hours for these 25 time steps when using the FP-scheme and close to 12 hours with the CAT-scheme. Our proposed method took approximately 2 hours for a simulation with $n_{\text{uniform}} = 256$ confirming the superior complexity. We include reference lines with the expected complexity and see them match for the few presented points.
Further increasing the number of particle properties is not practically feasible on an office desktop computer as N grows exponentially. Storing these large high-dimensional tensors on a non-supercomputer is infeasible and a computation impossible even if more computational power is available by, for example, using parallel computing.

Chapter 4

Efficient storage and arithmetic in tensor-train format

In this chapter, we give a thorough introduction to efficient tensor storage. We present one particular format, namely the tensor-train (TT)-format (and the idea of its representation of tensors) in section 4.1. In section 4.2 we present arithmetic operations, that are used in this thesis.
Section 4.3 presents the truncation algorithm that allows to reduce the storage requirement of a tensor A and proposes an improvement over the literature. Section 4.4 explains the application of the format to the setting of PBEs and section 4.5 closes out with numerical results using a *C++*- implementation of this format. We here use the Eigen library ([25]) for underlying linear algebra routines.

4.1 Introduction

This section is dedicated to the introduction of the TT-format. This storage format for high-dimensional tensors was first introduced in [70] and reintroduced with mathematical rigor in [54]. The format was later found to be a special case of the *Hierarchichal tensor format*, which was first introduced in [29]. The TT-format does not rely on recursive definitions and tree structures and allows for a representation with less overhead and an easier understanding.

Given a tensor $A \in \mathbb{R}^{n_1 \times \dots \times n_d}$, its direct storage requires $N = \prod_{m=1}^{d} n_m$ real numbers. This is infeasible for a high number of dimensions (and $d = 3$ is sufficiently high in most cases) and a more efficient approach is needed to represent A.

Definition 4.1.1. Unfolding matrix
Given a tensor $A \in \mathbb{R}^N$ with entries $A_{\mathbf{i}}$, the m-th unfolding matrix

$$\hat{A}_{(1,m),(m+1,d)} \in \mathbb{R}^{(n_1 \cdots n_m) \times (n_{m+1} \cdots n_d)}$$

is given by

$$\hat{A}_{(1,m),(m+1,d)}(i_1, \ldots, i_m; i_{m+1}, \ldots, i_d) = A_{\mathbf{i}} \tag{4.1}$$

with so-called long indices $(i_1, \ldots, i_m)$ and $(i_{m+1}, \ldots, i_d)$.

An unfolding matrix allows us to represent a tensor as a matrix containing all data.

Example 4.1.1. *Given a tensor $A \in \mathbb{R}^{2\times3\times4\times5}$ with entries $A_{\mathbf{i}}$, the second unfolding matrix $\hat{A}_{(1,2),(3,4)} \in \mathbb{R}^{6\times20}$ is given by*

$$\hat{A}_{(1,2),(3,4)} = \begin{pmatrix} A_{0,0,0,0} & A_{0,0,0,1} & \cdots & A_{0,0,0,4} & A_{0,0,1,0} & \cdots & A_{0,0,3,4} \\ A_{0,1,0,0} & & & & & & A_{0,1,3,4} \\ A_{0,2,0,0} & & & & & & \\ A_{1,0,0,0} & & & \ddots & & & \vdots \\ A_{1,1,0,0} & & & & & & \\ A_{1,2,0,0} & A_{1,2,0,1} & & \cdots & & & A_{1,2,3,4} \end{pmatrix}.$$

The key to efficient tensor storage lies in a low-rank representation of these unfolding matrices, as it allows to reduce the required storage for a large matrix (or tensor).
Let $\hat{A}_{(1,m),(m+1,d)} \in \mathbb{R}^{n_1 \cdots n_m \times n_{m+1} \cdots n_d}$ be the m-th unfolding matrix of $A \in \mathbb{R}^N$. We denote its rank by r_m^A and find a representation as a product of two matrices with rank r_m^A. We obtain

$$\hat{A}_{(1,m),(m+1,d)} =: A_{\text{col}}^{(1,m)} \cdot A_{\text{row}}^{(m+1,d)}, \quad m = 1, \ldots, d-1 \tag{4.2}$$

with $\hat{A}_{\text{col}}^{(1,m)} \in \mathbb{R}^{n_1 \cdots n_m \times r_m^A}$ and $\hat{A}_{\text{row}}^{(m+1,d)} \in \mathbb{R}^{r_m^A \times n_{m+1} \cdots n_d}$. The indices row and col signal the orientation of the long indices. We define $r_0^A = r_d^A = 1$ for completeness.

We use these low-rank matrices to find the representation of a tensor as a tensor-train. This is given by d three-dimensional tensors $A^m \in \mathbb{R}^{r_{m-1}^A \times n_m \times r_m^A}$, the so-called *cores*. Each core A^m consists of n_m so-called *slices* $A_{i_m}^m \in \mathbb{R}^{r_{m-1}^A \times r_m^A}$. These cores define the entries $A_{\mathbf{i}}$ by calculating

$$A_{\mathbf{i}} = \prod_{m=1}^{d} A_{i_m}^m, \tag{4.3}$$

the matrix-matrix product of the i_m-th slices of A^m. The definitions of $r_0^A = r_d^A = 1$ guarantee this to be a scalar. An example of a tensor in TT-representation for $d = 3$ can be found in Figure 4.1. Next, we prove that a tensor with unfoldings $\hat{A}_{(1,m),(m+1,d)}$ of ranks r_m^A can be represented in the TT-format, using the following two definitions.

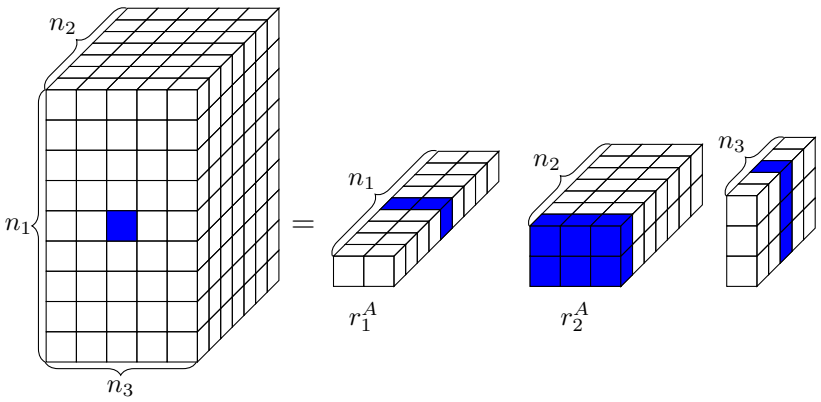


Figure 4.1: TT-representation for a three-dimensional tensor with $\mathbf{n} = (9, 7, 5)$ and ranks $r_1^A = 2$, $r_2^A = 3$. Marked entries are relevant to evaluate $\mathbf{i} = (4, 0, 2)$.

Definition 4.1.2. Oriented unfolding matrix of a core
Let $A^m \in \mathbb{R}^{r_{m-1}^A \times n_m \times r_m^A}$ be a core. Then

$$A_{col}^m := \begin{bmatrix} A_0^m \\ \vdots \\ A_{n_m-1}^m \end{bmatrix} \in \mathbb{R}^{r_{m-1}^A n_m \times r_m^A} \tag{4.4}$$

and

$$A_{row}^m := \begin{bmatrix} A_0^m & \dots & A_{n_m-1}^m \end{bmatrix} \in \mathbb{R}^{r_{m-1}^A \times n_m r_m^A} \tag{4.5}$$

are called the column- and row-wise unfoldings of the core A^m. They are two special unfolding matrices of the three-dimensional core A^m.

Definition 4.1.3. Concatenation of cores
Let $A^m \in \mathbb{R}^{r_{m-1} \times n_m \times r_m}$ and $A^{m+1} \in \mathbb{R}^{r_m \times n_{m+1} \times r_{m+1}}$ be adjacent cores of a TT-representation of A. The binary operation $\oplus$ (referred to as concatenation*) is defined by a blockwise Kronecker product given by*

$$A_{col}^m \oplus A_{col}^{m+1} := \begin{bmatrix} A_0^m \cdot A_0^{m+1} \\ A_0^m \cdot A_1^{m+1} \\ \vdots \\ A_{n_m-1}^m \cdot A_{n_{m+1}-2}^{m+1} \\ A_{n_m-1}^m \cdot A_{n_{m+1}-1}^{m+1} \end{bmatrix} \in \mathbb{R}^{r_{m-1}^A n_m n_{m+1} \times r_{m+1}^A} \tag{4.6}$$

$$A_{row}^m \oplus A_{row}^{m+1} := \begin{bmatrix} A_0^m \cdot A_0^{m+1} & A_0^m \cdot A_1^{m+1} & \dots & A_{n_m-1}^m \cdot A_{n_{m+1}-1}^{m+1} \end{bmatrix} \in \mathbb{R}^{r_{m-1}^A \times n_m n_{m+1} r_{m+1}^A} \tag{4.7}$$

With these defined one can extend the definition to redefine the usual matrix product via

$$A^m_{col} \oplus A^{m+1}_{row} := A^m_{col} \cdot A^{m+1}_{row} = \begin{bmatrix} A^m_0 \cdot A^{m+1}_0 & \dots & A^m_0 \cdot A^{m+1}_{n_{m+1}-1} \\ \vdots & & \vdots \\ A^m_{n_m-1} \cdot A^{m+1}_0 & \dots & A^m_{n_m-1} \cdot A^{m+1}_{n_{m+1}-1} \end{bmatrix} \in \mathbb{R}^{r^A_{m-1} n_m \times n_{m+1} r^A_{m+1}}. \tag{4.8}$$

We note that all three concatenations contain the same block-matrices that are arranged differently. They can easily be derived from one another by reshaping the result.
This definition can also be extended to sequences of concatenations of cores where longer products make up the resulting block matrix. The long indices are arranged in a lexicographically ascending order, column-wise and row-wise, respectively. The concatenation is associative as the underlying matrix products are associative and the arrangement of blocks is independent of the order of operation.

We now define a matrix M_m by repeated concatenation as

$$M_m := A^1_{\text{col}} \oplus \cdots \oplus A^m_{\text{col}} \oplus A^{m+1}_{\text{row}} \oplus \cdots \oplus A^d_{\text{row}} \in \mathbb{R}^{n_1 \cdots n_m \times n_{m+1} \cdots n_d}.$$

We find every entry $M_m(i_1, \dots, i_m; i_{m+1}, \dots, i_d)$ to be of the form

$$M_m(i_1, \dots, i_m; i_{m+1}, \dots, i_d) = \prod_{\ell=1}^{d} A^\ell_{i_\ell} = A_{\mathbf{i}}$$

and conclude that M_m equals $\hat{A}_{(1,m),(m+1,d)}$, the m-th unfolding matrix of A. Using the associative property of the concatenation we write

$$\begin{aligned} M_m &= (A^1_{\text{col}} \oplus \cdots \oplus A^m_{\text{col}}) \oplus (A^{m+1}_{\text{row}} \oplus \cdots \oplus A^d_{\text{row}}) \\ &= M^{(1,m)}_{\text{col}} \qquad\qquad \cdot M^{(m+1,d)}_{\text{row}}, \end{aligned}$$

where we find the low-rank factors $M^{(1,m)}_{\text{col}}$ and $M^{(m+1,d)}_{\text{row}}$ (with rank r^A_m) as candidates for $A^{(1,m)}_{\text{col}}$ and $A^{(m+1,d)}_{\text{row}}$ in (4.2).We remind the reader here of the use of the concatenation operator $\oplus$ as usual matrix multiplication in case of a column- and a row-unfolding. The index m can be chosen arbitrarily and shows the potential of the TT-format to represent the low-rank representations of all $\hat{A}_{(1,m),(m+1,d)}$ simultaneously.
We now assume a tensor $A \in \mathbb{R}^N$ that allows for low-rank representations as in (4.2) with known ranks r^A_m. We know cores $A^m \in \mathbb{R}^{r^A_{m-1} \times n_m \times r^A_m}$ to exist and show how to obtain these from the unfolding matrices $\hat{A}_{(1,m),(m+1,d)}$.
We know the first unfolding matrix $\hat{A}_{(1,1),(2,d)}$ can be written via core-concatenation as

$$\hat{A}_{(1,1),(2,d)} = (A^1_{\text{col}}) \oplus (A^2_{\text{row}} \oplus \dots \oplus A^d_{\text{row}}). \tag{4.9}$$

We know by (4.2) that $\hat{A}_{(1,1),(2,d)}$ has the low rank factors $A^{(1,1)}_{\text{col}}$ and $A^{(2,d)}_{\text{row}}$ and obtain the first core A^1_{col} as $A^{(1,1)}_{\text{col}}$. The right factor $A^{(2,d)}_{\text{row}}$ will be used next.
We turn our attention to the second unfolding matrix $\hat{A}_{(1,2),(3,d)}$ and write it as

$$\hat{A}_{(1,2),(3,d)} = A^{(1,2)}_{\text{col}} \cdot A^{(3,d)}_{\text{row}}$$

which we know can be written as a concatenation of cores in the form

$$\begin{aligned}\hat{A}_{(1,2),(3,d)} &= \left(A^1_{\text{col}} \oplus A^2_{\text{col}}\right) \oplus \left(A^3_{\text{row}} \oplus \cdots \oplus A^d_{\text{row}}\right) \\ &= A^1_{\text{col}} \oplus \left(A^2_{\text{col}} \oplus A^3_{\text{row}} \oplus \cdots \oplus A^d_{\text{row}}\right). \end{aligned} \tag{4.10}$$

The bracketed term in (4.10) is a concatenation of the cores A^2 to A^d as was $A^{(2,d)}_{\text{row}}$. We established all concatenation to have the same matrix-blocks, the difference in orientation of A^2 can be compensated by rearranging the entries of $A^{(2,d)}_{\text{row}}$. We denote this rearrangement as $\hat{A}_{(2,2),(3,d)} \in \mathbb{R}^{r^A_1 n_2 \times n_3 \cdots n_d}$. We replace $\oplus$ by a usual matrix product and see that A^2_{col} can be obtained as a low-rank factor of $\hat{A}_{(2,2),(3,d)}$. The existence of a low-rank decomposition of this matrix is guaranteed by the existence of the representation (4.2) of $\hat{A}_{(1,2),(3,d)}$. The right low-rank factor $A^{3,d}_{\text{row}}$ will be used in the next computation.
This process can be iterated to obtain an column-wise representation A^m_{col} of any core as left low-rank factor of $\hat{A}_{(m,m),(m+1,d)}$, a rearrangement of $A^{(m,d)}_{\text{row}}$, the right low-rank factor of the previous computation. The existence of these low-rank-factors with ranks r^A_m is guaranteed by the existence of low-rank factors $A^{(1,m)}_{\text{col}}$ and $A^{(m+1,d)}_{\text{row}}$ of the unfolding matrices $\hat{A}_{(1,m),(m+1,d)}$ assumned in (4.2).
A row-wise expression of the last core A^d_{row} can be found as the right low-rank factor of $\hat{A}_{(d-1,d-1),(d,d)} \in \mathbb{R}^{r^A_{d-2} n_{d-1} \times n_d}$ a rearragenment of $A^{(d-1,d)}_{\text{row}}$. We summarize this process in Algorithm 1.
The terms *left* and *right* are commonly used to refer to a direction, where *left* denotes cores with a lower dimensional index m and *right* to cores with a higher dimensional index.
For easier estimates of complexity we use $R_A \geq r^A_j \quad \forall j \in 1, \ldots, d$ and define

Algorithm 1 Exact TT-representation

Input: Full tensor $A \in \mathbb{R}^N$
Output: Cores A^1 to A^d and ranks r^A_1 to r^A_{d-1}
  for $m = 1$ to $d-1$ **do**
    Find unfolding matrix $\hat{A}_{(m,m),(m+1,d)}$ of $A^{(m,d)}_{\text{row}}$
    Obtain low-rank-representation of $\hat{A}_{(m,m),(m+1,d)} = A^m_{\text{col}} \cdot A^{(m+1,d)}_{\text{row}}$
    Set r^A_m as number of columns of A^m_{col}
    Save blocks of A^m_{col} as slices of A^m
  end for
  Save columns of A^d_{row} as slices of A^d

$$n_\Sigma := \sum_{j=1}^{d} n_j \tag{4.11}$$

as the total number of slices. With these, we can estimate storage requirements for the representation in the TT-format by $\mathcal{O}(R_A^2 n_\Sigma)$, which is significantly smaller than $N = \prod_{m=1}^{d} n_m$, as long as the ranks r_m^A are moderate. We drop the tensor-superscript for ranks r_m^A if there is only one tensor present.

4.2 Arithmetic operations

This section is devoted to presenting arithmetic operations, that are used in this thesis. Most of these algorithms were introduced in [54] or subsequent work and are repeated here for completeness. We will also derive estimates for the norm of the resulting cores. This will help us with the truncation procedure presented in section 4.3.

Definition 4.2.1. Orthogonal cores
Let $A^m \in \mathbb{R}^{r_{m-1}\times n_m \times r_m}$ be a core of a tensor train with slices $A_i^m \in \mathbb{R}^{r_{m-1}\times r_m}$. The core A^m is called column-orthogonal if $A_{col}^m \in \mathbb{R}^{r_{m-1}n_m \times r_m}$ satisfies

$$\left(A_{col}^m\right)^T A_{col}^m = \sum_{i=0}^{n_m-1} \left(A_i^m\right)^T A_i^m = I \in \mathbb{R}^{r_m \times r_m}.$$

We similarly call a core A^m row-orthogonal, if $A_{row}^m \in \mathbb{R}^{r_{m-1}\times n_m r_m}$ satisfies

$$A_{row}^m \left(A_{row}^m\right)^T = I \in \mathbb{R}^{r_{m-1}\times r_{m-1}}.$$

We will show in section 4.2.2 that any tensor $A \in \mathbb{R}^N$ in the TT-format can be expressed using $d-1$ orthogonal cores and one non-orthogonal core that separates the row-orthogonal and column-orthogonal cores in a constructive manner. We will call a representation $A_{\mathbf{i}} = \prod_{i=1}^{d} A_{i_m}^i$ *column-orthogonal* or *row-orthogonal*, if A^1 through A^{d-1} are column-orthogonal or A^2 through A^d are row-orthogonal, respectively. Throughout this work, we will make use of this definition and keep track of how orthogonality is preserved in arithmetic operations.
The following lemma will be helpful to estimate the norm of the result of an arithmetic operation of two orthogonal cores.

Lemma 2. *Let $A^m \in \mathbb{R}^{r_{m-1}\times n_m \times r_m}$ be a core of A. Then there holds*

$$\|A_{col}^m\|_2^2 \leq \sum_{i=0}^{n_m-1} \|A_i^m\|_2^2 \quad \text{and} \quad \|A_{row}^m\|_2^2 \leq \sum_{i=0}^{n_m-1} \|A_i^m\|_2^2. \tag{4.12}$$

If A^m is column-orthogonal, then there holds

$$\sum_{i=0}^{n_m-1} \|A_i^m\|_2^2 \leq r_m. \tag{4.13}$$

If A^m is row-orthogonal we have $\sum_{i=0}^{n_m-1} \|A_i^m\|_2^2 \leq r_{m-1}$.

Proof. We prove (4.12) by using the triangle inequality and write

$$\begin{aligned}\|A_{\text{col}}^m\|_2^2 = \quad & \| \left(A_{\text{col}}^m\right)^T A_{\text{col}}^m\|_2 = \left\| \sum_{i=0}^{n_m-1} \left(A_{\text{col}}^m\right)^T A_i^m \right\|_2 \\ \leq & \sum_{i=0}^{n_m-1} \| \left(A_{\text{col}}^m\right)^T A_i^m\|_2 = \quad \sum_{i=0}^{n_m-1} \|A_i^m\|_2^2.\end{aligned}$$

Now let A^m be column-orthogonal. By using the Frobenius norm we can prove second inequality (4.13)

$$\sum_{i=0}^{n_m-1} \|A_i^m\|_2^2 \leq \sum_{i=0}^{n_m-1} \|A_i^m\|_F^2 = \|A_{\text{col}}^m\|_F^2 = r_m$$

because all r_m columns of A_{col}^m have unit length resulting in $\|A_{\text{col}}^m\|_F = \sqrt{r_m}$ for a column-orthogonal core A^m. The same argument can be used to show the result for a row-orthogonal core. □

This lemma allows two things. We can estimate the norm of a core based on the norm of the slices, which will be useful to analyze the results of arithmetic operations. And we can estimate the sum of norms of the slices of orthogonal cores without investigating these individually.
The following lemma allows us to carry this result over into the concatenation operation and connect multiple cores.

Lemma 3. *Let $A^m \in \mathbb{R}^{r_{m-1} \times n_m \times r_m}$ and $A^{m+1} \in \mathbb{R}^{r_m \times n_{m+1} \times r_{m+1}}$ be adjacent cores of A. Then*

$$\|A_{col}^m \oplus A_{col}^{m+1}\|_2 \leq \|A_{col}^m\|_2 \cdot \|A_{col}^{m+1}\|_2 \tag{4.14}$$

and

$$\|A_{row}^m \oplus A_{row}^{m+1}\|_2 \leq \|A_{row}^m\|_2 \cdot \|A_{row}^{m+1}\|_2 \tag{4.15}$$

as well as

$$\|A_{col}^m \oplus A_{row}^{m+1}\|_2 \leq \|A_{col}^m\|_2 \cdot \|A_{row}^{m+1}\|_2 \tag{4.16}$$

hold.

Proof. Again we will only show this for (4.14), as (4.15) follows by the same argument. We note that (4.16) holds by the sub-multiplicativity of the spectral norm. We have

$$\begin{aligned}\|A^m_{\text{col}} \oplus A^{m+1}_{\text{col}}\|_2 &= \left\| \sum_{j=0}^{n_{m+1}-1} \sum_{i=0}^{n_m-1} A_j^{m+1^T} A_i^{m^T} A_i^m A_j^{m+1} \right\|_2^{1/2} \\ &= \left\| \sum_{j=0}^{n_{m+1}-1} A_j^{m+1^T} \Big(\sum_{i=0}^{n_m-1} A_i^{m^T} A_i^m \Big) A_j^{m+1} \right\|_2^{1/2} \end{aligned} \tag{4.17}$$

and use

$$\left\| \sum_{i=0}^{n_m-1} A_j^{m^T} A_j^m \right\|_2 = \|A^m_{\text{col}}\|_2^2$$

to replace this inner sum with $\|A^m_{\text{col}}\|_2^2 \cdot I$ which will not decrease the norm in (4.17) because we replace all singular values with the maximal singular value of A^m_{col}. From there on, we write

$$\begin{aligned}\|A^m_{\text{col}} \oplus A^{m+1}_{\text{col}}\|_2 &\leq \|A^m_{\text{col}}\|_2 \cdot \left\| \sum_{j=0}^{n_{m+1}-1} A_j^{m+1^T} A_j^{m+1} \right\|_2^{1/2} \\ &= \|A^m_{\text{col}}\|_2 \cdot \|A^{m+1}_{\text{col}}\|_2,\end{aligned}$$

which finishes the proof. □

We especially note that the concatenation of two orthogonal cores is still orthogonal.

4.2.1 Approximation of full tensors

The existence of a TT-representation of a tensor $A \in \mathbb{R}^N$ relies on a low-rank representation of all $d-1$ (non-trivial) unfolding matrices with ranks $r_m \ll N$ to eliminate the exponential complexity.
We are interested in the TT-format as data storage for numerical approximations with an inherent discretization error that cannot be avoided. Therefore we accept to replace each unfolding matrix $\hat{A}_{(1,m),(m+1,d)}$ by an approximate matrix $\hat{B}_{(1,m),(m+1,d)}$ with smaller rank to replace the tensor A with an approximation B in a TT-representation with lower ranks. This procedure was first introduced in [54] and is repeated here for completeness.
We measure the difference by

$$\|A - B\|_F := \left(\sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n}} (A_{\mathbf{i}} - B_{\mathbf{i}})^2 \right)^{\frac{1}{2}}, \tag{4.18}$$

the Frobenius norm of the difference of A and B. This norm is invariant under the permutation of indices or different unfolding matrices of A. It is also not an induced matrix norm, which is preferable in this setting of data storage.
In a practical implementation our goal is to find a tensor B with minimal ranks r_m^B that fulfills $\|A - B\|_F < \epsilon \cdot \|A\|_F$ (with a prescribed accuracy $\epsilon > 0$) by replacing the exact low-rank representation of unfolding matrices with an approximation. It is well known that dropping the smallest singular values (and ignoring the associated singular vectors) will give an optimal approximation with respect to the Frobenius norm. We alter Algorithm 1 by replacing the exact low-rank-representation in every iteration by low-rank-factors obtained via a truncated singular value decomposition (SVD) with accuracy τ_m to be determined later. We will refer to the matrices obtained in this step by $\hat{A}_{(m,m),(m+1,d)} = U_m S_m V_m^T + E_m$ with $\|E_m\|_F \leq \tau_m$ and will use $A_{\text{col}}^{(m+1,d)} = S_m V_m^T$ for the next step (after a suitable rearrangement). The obtained core $B_{\text{col}}^m = U_m$ is column-orthogonal by construction. By this we obtain an approximation $B \in \mathbb{R}^N$ with cores B^1 to B^d.
In order to investigate the error $\|A - B\|_F$ made during this process we need to inspect the obtained cores and error-matrices more closely. We will write concatenations without any column/row-indicators as we associate the unfolding matrix resulting from the concatenation with the d-dimensional tensors they represent. We can simply reshape the obtained matrices into tensors in $\mathbb{R}^N$. This allows for the reduction of notation and straight forward arithmetic of unfolding matrices. We start with the tensor A and the SVD of the first unfolding with

$$\begin{aligned} A &= E_1 + U_1 S_1 V_1^T \\ &= E_1 + B^1 \oplus A^{2,d} \end{aligned}$$

We then rearrange $A^{2,d}$ and calculate the truncated SVD of that matrix, resulting in

$$\begin{aligned} A &= E_1 + B^1 \oplus (E_2 + U_2 S_2 V_2^T) \\ &\dots \\ &= E_1 + B^1 \oplus (E_2 + B^2 \oplus (\dots \oplus (E_{d-1} + B^{d-1} \oplus B^d)) \end{aligned} \tag{4.19}$$

when we iterate the steps. We reorder the expression (4.19) and see one term to be given by $B_1 \oplus \dots \oplus B^d$, which is associated with the tensor B. We subtract this and take the squared Frobenius norm on both sides of the equation. This gives

$$\|A - B\|_F^2 = \|E_1 + B^1 \oplus (E_2 + B^2 \oplus (\dots \oplus E_{d-1}))\|_F^2. \tag{4.20}$$

We have $E_1^T \cdot B^1 = 0$ because the columns of E_1 and B^1_{col} are orthogonal by construction. This allows a split of the squared Frobenius norm in (4.20) into two separate norms with equality. We then have

$$\begin{aligned}\|A - B\|_F^2 &= \|E_1\|_F^2 + \|B^1 \oplus (E_2 + B^2 \oplus (\ldots \oplus E_{d-1})\|_F^2 \\ &= \|E_1\|_F^2 + \|B^1 \oplus E_2\|_F^2 + \|B^1 \oplus B^2 \oplus (\ldots \oplus E_{d-1})\|_F^2 \\ &= [\|E_1\|_F^2 + \|E_2\|_F^2 + \|B^2 \oplus (\ldots \oplus E_{d-1})\|_F^2\end{aligned}$$

as we use the invariance of the Frobenius norm under orthogonal matrices to elimnate B^1. We then repeat the argument of orthogonality to split the Frobenius norm and eliminate the orthogonal cores B^m, as they are orthogonal and have no influence on the Frobenius norm. This finally gives

$$\|A - B\|_F^2 = \|E_1\|_F^2 + \|E_2\|_F^2 + \cdots + \|E_{d-1}\|_F^2.$$

We then use the assumption $\|E_m\|_F \leq \tau_m$ to write

$$\|A - B\|_F^2 \leq \sum_{m=1}^{d-1} \tau_m^2.$$

If we assume $\tau := \tau_1 = \cdots = \tau_{d-1}$ we find one limit τ for all SVDs with

$$\|A - B\|_F \leq \epsilon \|A\|_F \leq \sqrt{d-1} \cdot \tau$$

and the later implies

$$\tau \leq \frac{\epsilon \|A\|_F}{\sqrt{d-1}} \tag{4.21}$$

as upper limit $\|E_m\|_F \leq \tau$ for all $m \in \{1, \ldots, d-1\}$. We modify Algorithm 1 to reflect this change in obtaining the low-rank representations and summarize this in Algorithm 2.

This algorithm was first introduced in [55] and still requires the assembly and handling of the complete tensor, which is infeasible in many applications. An improvement, namely the Tensor-train renormalization-cross (TT-RC) algorithm, was devised in [64] where an initial TT-tensor (often chosen as trivial) is iteratively improved by replacing two adjacent cores in every iteration until convergence is reached. We use this algorithm in all numerical simulations to be shown in section 4.5.

4.2.2 Orthogonalization of cores

To obtain orthogonal representations of a tensor we need to iteratively orthogonalize core after core. This can be done by replacing a pair of adjacent cores (A^m and A^{m+1}) by two new cores (B^m and B^{m+1}) with one being orthogonal.

Algorithm 2 Approximate TT-representation B of a full tensor A with error prescription

Input: Full tensor $A \in \mathbb{R}^N$ accuracy $\epsilon \in \mathbb{R}_{\geq 0}$
Output: Cores B^1 to B^d and ranks r_1^B to r_{d-1}^B
  Define temporary tensor $B = A \in \mathbb{R}^N$
  Compute $\tau = \frac{\epsilon \|A\|_F}{\sqrt{d-1}}$
  for $m = 1$ to $d-1$ **do**
    Define size of multi-index $N_R = \prod_{k=m+1}^{d} n_k$
    Find unfolding matrix $\hat{B} \in \mathbb{R}^{r_{m-1}^B \cdot n_m \times N_R}$ of B
    Compute SVD of $\hat{B} = USV^T + E$ with $\|E\|_F \leq \tau$ and obtain rank r_m^B
    Save blocks of U as slices of B_{col}^m
    Define temporary tensor $B = SV^T \in \mathbb{R}^{r_m^B \times n_{m+1} \times \ldots \times n_d}$
  end for
  Save columns of B as B_{row}^d

This can be done by a QR-decomposition for example. Given two adjacent cores $A^m \in \mathbb{R}^{r_{m-1} \times n_m \times r_m}$ and $A^{m+1} \in \mathbb{R}^{r_m \times n_{m+1} \times r_{m+1}}$, we can express all contained information by $A_{\text{col}}^m \oplus A_{\text{row}}^{m+1} \in \mathbb{R}^{r_{m-1} n_m \times n_{m+1} r_{m+1}}$.
Given a QR-decomposition $A_{\text{col}}^m = QR$, we can write

$$A_{\text{col}}^m \oplus A_{\text{row}}^{m+1} = Q \oplus RA_{\text{row}}^{m+1} \tag{4.22}$$

to express the same information. This results in

$$B_i^m := A_i^m \cdot R^{-1} \quad \text{and} \quad B_i^{m+1} := R \cdot A_i^{m+1} \tag{4.23}$$

with a column-orthogonal core B^m. Doing this with every pair of cores of a tensor from left to right gives a column-orthogonal representation.
If we calculate $A_{\text{row}}^{m+1} = L \cdot Q$ with $QQ^T = I$, we can similarly replace A^m and A^{m+1} with

$$B_i^m = A_i^m \cdot L \quad \text{and} \quad B_i^{m+1} = L^{-1} \cdot A_i^{m+1} \tag{4.24}$$

to obtain a row-orthogonal representation if iterated from right to left. The complexity of a complete orthogonalization is given by $\mathcal{O}(R_A^3 n_\Sigma)$.
An orthogonal representation can also be achieved via other decompositions that use orthogonal factors like the SVD as mentioned in section 4.2.1. We again mention that every tensor in the TT-format can be represented by using $d-1$ orthogonal cores and one non-orthogonal core seperating the column-orthogonal cores from the row-orthogonal cores. The presented procedure is a constructive way to obtain the desired orthogonality.

4.2.3 Addition of tensors

The addition of two tensors $A, B \in \mathbb{R}^N$ in TT-format yields a tensor $C \in \mathbb{R}^N$ with elements

$$C_{\mathbf{i}} = A_{\mathbf{i}} + B_{\mathbf{i}} = \prod_{m=1}^{d} A_{i_m}^m + \prod_{m=1}^{d} B_{i_m}^m.$$

The tensor C allows a TT-format with block-diagonal slices

$$C_{i_m}^m = \begin{bmatrix} A_{i_m}^m & 0 \\ 0 & B_{i_m}^m \end{bmatrix}, \ m \in \{2, \dots, d-1\},$$

and

$$C_{i_1}^1 = \begin{bmatrix} A_{i_1}^1 & B_{i_1}^1 \end{bmatrix}, \qquad C_{i_d}^d = \begin{bmatrix} A_{i_d}^d \\ B_{i_d}^d \end{bmatrix}.$$

The ranks of C are $r_m^C = r_m^A + r_m^B$ for all $m \in \{1, \cdots, d-1\}$. This addition requires no arithmetic operations but $\mathcal{O}\left(R_C^2 n_\Sigma\right)$ assignments in a direct implementation. We use $R_C \geq r_m^C$ as upper bound for all ranks r_m^C of C and n_Σ as defined in (4.11).
If A and B are represented in a column-orthogonal TT-format, we can estimate the norm of the column representation of the cores of C by

$$\begin{aligned} \|C_{\text{col}}^1\|_2^2 &\leq \sum_{i=0}^{n_1-1} \| \begin{bmatrix} A_i^1 & B_i^1 \end{bmatrix} \|_2^2 \\ &= \sum_{i=0}^{n_1-1} \|(A_i^1)^T A_i^1 + (B_i^1)^T B_i^1\|_2 \\ &\leq \sum_{i=0}^{n_1-1} \|{A_i^1}^T A_i^1\|_2 + \sum_{i=0}^{n_1-1} \|{B_i^1}^T B_i^1\|_2 \leq r_1^A + r_1^B \end{aligned} \tag{4.25}$$

for C^1 (we used (4.12), the triangle inequality and (4.13) respectively) and

$$\|C_{\text{col}}^m\|_2^2 = \left\| \begin{bmatrix} \sum_{i=0}^{n_m-1} {A_i^m}^T A_i^m & 0 \\ 0 & \sum_{j=0}^{n-1} {B_i^m}^T B_i^m \end{bmatrix} \right\|_2 = \left\| \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \right\|_2 = 1$$

for every other column-orthogonal core ($m \in \{2, \dots, d-1\}$).
The cores A^d and B^d are not orthogonal and we cannot say anything about C^d. We will see in section 4.3.1 that an estimate is not required for the rightmost core.

4.2.4 Hadamard product of tensors

The elementwise (or Hadamard) product of two tensors $A, B \in \mathbb{R}^N$ in TT-format, denoted by $C = A \odot B \in \mathbb{R}^N$, exploits properties of the matrix Kronecker product, denoted by $\otimes$. For scalars, there holds

$$C_{\mathbf{i}} = A_{\mathbf{i}} \cdot B_{\mathbf{i}} = A_{\mathbf{i}} \otimes B_{\mathbf{i}}.$$

The mixed-product property of the Kronecker product states that

$$\left(\prod_{m=1}^{d} A_{i_m}^m\right) \otimes \left(\prod_{m=1}^{d} B_{i_m}^m\right) = \prod_{m=1}^{d} \left(A_{i_m}^m \otimes B_{i_m}^m\right)$$

which allows us to reorder matrix products and Kronecker products. We use this and the TT-format (4.3) for A and B and obtain

$$\begin{aligned} C_{\mathbf{i}} &= \overbrace{(A_{i_1}^1 \cdots A_{i_d}^d)}^{A_{\mathbf{i}}} \otimes \overbrace{(B_{i_1}^1 \cdots B_{i_d}^d)}^{B_{\mathbf{i}}} \\ &= (A_{i_1}^1 \otimes B_{i_1}^1) \cdots (A_{i_d}^d \otimes B_{i_d}^d) \\ &= \prod_{m=1}^{d} \underbrace{(A_{i_m}^m \otimes B_{i_m}^m)}_{=:C_{i_m}^m} = \prod_{m=1}^{d} C_{i_m}^m, \end{aligned}$$

i. e., we have C in TT-format with ranks $r_m^C = r_m^A \cdot r_m^B$ and slices $C_i^m = A_i^m \otimes B_i^m$. This procedure requires n_Σ (4.11) of these Kronecker products, each requiring at most $R_A^2 \cdot R_B^2$ multiplications and assignments, leading to a total complexity of $\mathcal{O}(R_C^2 n_\Sigma)$.
If A and B are given in a column-orthogonal representation, we estimate

$$\begin{aligned} \|C_{\mathrm{col}}^m\|_2^2 &\leq \sum_{i=0}^{n_m-1} \|A_i^m \otimes B_i^m\|_2^2 \\ &= \sum_{i=0}^{n_m-1} \|A_i^m\|_2^2 \cdot \|B_i^m\|_2^2 \leq \min(r_m^A, r_m^B) \end{aligned} \tag{4.26}$$

for every core C^m but $m = d$. The last estimation follows from using either $\|A_i^m\|_2^2 \leq 1$ or $\|B_i^m\|_2^2 \leq 1$ in each summand.

4.2.5 Convolution of tensors

The convolution of two tensors $\phi, \psi \in \mathbb{R}^N$ in TT-Format with cores ϕ^m and ψ^m is denoted by $\omega = \phi * \psi \in \mathbb{R}^{2n_1 \times \ldots \times 2n_d}$. With zero-padded tensors $\widetilde{\phi}, \widetilde{\psi} \in \mathbb{R}^{2n_1 \times \ldots \times 2n_d}$ in TT-format, ω is given as

$$\begin{aligned}
\omega_{\mathbf{i}} &= \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{i}} \widetilde{\phi}_{\mathbf{j}} \widetilde{\psi}_{\mathbf{i}-\mathbf{j}} \\
&= \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{i}} \prod_{m=1}^{d} \widetilde{\phi}^m_{j_m} \otimes \widetilde{\psi}^m_{i_m - j_m} \\
&= \prod_{m=1}^{d} \sum_{j_m=0}^{i_m} \widetilde{\phi}^m_{j_m} \otimes \widetilde{\psi}^m_{i_m - j_m}.
\end{aligned}$$

This shows ω allows for a representation in the TT-format with the i_m-th slice of ω^m given by

$$\omega^m_{i_m} := \sum_{j_m=0}^{i_m} \widetilde{\phi}^m_{j_m} \otimes \widetilde{\psi}^m_{i_m - j_m} =: \left(\widetilde{\phi}^m * \widetilde{\psi}^m \right)_{i_m}. \tag{4.27}$$

The last definition in (4.27) denotes the convolution of two cores $\widetilde{\phi}$ and $\widetilde{\psi}$.
The zero-padding can easily be achieved by appending n_m zero-matrices of appropriate sizes to the m-th core. This will guarantee a linear convolution result (see Remark 3).
The ranks of ω are given by $r^\omega_m = r^\phi_m \cdot r^\psi_m$. We can estimate $\|\omega^m_{\text{col}}\|_2^2$, if $\widetilde{\phi}$ and $\widetilde{\psi}$ are given in a column-orthogonal representation by finding

$$\begin{aligned}
\|\omega^m_{\text{col}}\|_2^2 &\leq \sum_{i=0}^{2n_m-1} \| \sum_{j=0}^{i} \widetilde{\phi}^m_j \otimes \widetilde{\psi}^m_{i-j} \|_2^2 \\
&\leq \sum_{i=0}^{2n_m-1} \sum_{j=0}^{i} \|\widetilde{\phi}^m_j\|_2^2 \cdot \|\widetilde{\psi}^m_{i-j}\|_2^2 \\
&\leq \sum_{i=0}^{n_m-1} \|\psi^m_i\|_2^2 \cdot \sum_{j=0}^{n_m-1} \|\phi^m_j\|_2^2 \\
&\leq r^\psi_m \cdot r^\phi_m.
\end{aligned} \tag{4.28}$$

The computation of the convolution is of complexity $\mathcal{O}(R_C^2 n_m^2)$ for the m-th core and is quadratic in the number of cells. We can decrease this complexity by using the technique of fast Fourier transform described in section 3.4.

Fourier transform in TT-format

We recall via (3.25) the necessary steps for a multivariate convolution via Fourier transform. The Hadamard product in TT-format is already defined in

section 4.2.4 so we turn our attention to the Fourier transform $\Phi = \mathcal{F}(\widetilde{\phi})$ of $\widetilde{\phi}$ given in the TT-format.
Writing (3.26) with $\widetilde{\phi}$ in TT-format, we get

$$\begin{aligned}\Phi_{\mathbf{j}} &= \sum_{\mathbf{s}=\mathbf{0}}^{\mathbf{2n}-\mathbf{1}} \widetilde{\phi}^1_{s_1} \cdots \widetilde{\phi}^d_{s_d} \cdot e^{\mathrm{i}\pi s_1 j_1/n_1} \cdots e^{\mathrm{i}\pi s_d j_d/n_d} \\ &= \sum_{\mathbf{s}=\mathbf{0}}^{\mathbf{2n}-\mathbf{1}} (\widetilde{\phi}^1_{s_1} e^{\mathrm{i}\pi s_1 j_1/n_1}) \cdots (\widetilde{\phi}^d_{s_d} e^{\mathrm{i}\pi s_d j_d/n_d}) \\ &= \prod_{m=1}^{d} \underbrace{\sum_{s=0}^{2n_m-1} \widetilde{\phi}^m_s e^{\mathrm{i}\pi s_m j_m/n_m}}_{:=\Phi^m_{jm}}\end{aligned}$$

which is an expression for Φ in the TT-format.
We can interpret this expression as a Fourier transform along every dimension and define the core-wise Fourier transform by

$$\begin{aligned}\mathcal{F}^m \quad &: \quad \mathbb{R}^{r_m \times 2n_m \times r_{m+1}} \to \mathbb{C}^{r_m \times 2n_m \times r_{m+1}}, \quad \widetilde{\phi}^m \mapsto \mathcal{F}^m(\widetilde{\phi}^m) \quad \text{with} \quad (4.29) \\ &(\mathcal{F}^m(\widetilde{\phi}^m))_j := \sum_{s=0}^{2n_m-1} \widetilde{\phi}^m_s \cdot e^{\mathrm{i}\pi s j_m/n_m}.\end{aligned}$$

We note, that $\mathcal{F}^m$ consists of a Fourier transformation along the second (rank independent) dimension of the core, which can easily be implemented by a series of $r_{m-1} \cdot r_m$ one-dimensional fast Fourier transformations of length $2n_m$, resulting in a complexity of $\mathcal{O}(R_\phi^2 n_m \log(n_m))$ for the m-th core. We also want to point out the similarity to $\mathcal{F}^m$ from (3.29).
From there, we can easily define a core $\omega^m \in \mathbb{R}^{r_m \times 2n_m \times r_{m+1}}$ of $\omega = \widetilde{\phi} * \widetilde{\psi}$ by

$$\widetilde{\omega}^m = (\mathcal{F}^m)^{-1} (\mathcal{F}^m(\widetilde{\phi}^m) \odot \mathcal{F}^m(\widetilde{\psi}^m)) \tag{4.30}$$

implying every core of ω can be computed independently of any other.
It also implies that a convolution with respect to only a subset of dimensions can be done by evaluating (4.30) with respect to that subset of cores, while calculating the Hadamard-product of the remaining cores.

4.2.6 Add a dimension to a tensor

It regularly happens in population balance equations that a kernel function $\kappa(\mathbf{u}, \mathbf{v})$ is independent of some internal variable v_m that cannot be omitted from the consideration of the population of particles. The approximation of this function in TT-format via the TT-RC exploits a temporary omission.
Given an index set $\mathcal{I} := \mathcal{M} \setminus \{m\} = \{1, \ldots, m-1, m+1, \ldots, d\}$ and a tensor $A \in \mathbb{R}^{N^{\mathcal{I}}}$ in TT-format with $d-1$ cores, we now identify an entry by the multi-index

$$\mathbf{i}^{\mathcal{I}} := \left(i_1, \ldots, i_{m-1}, i_{m+1}, \ldots, i_d\right).$$

Given the augmented index set $\mathcal{M} = \mathcal{I} \cup \{m\} = \{1, \dots, d\}$, our goal is to obtain a tensor $C \in \mathbb{R}^{N^{\mathcal{M}}}$ in TT-format that satisfies

$$C_{i_1,\dots,i_{m-1},s,i_{m+1},\dots,i_d} = A_{i_1,\dots,i_{m-1},i_{m+1},\dots,i_d} \quad \forall s \in \{1, \dots, n_m\}. \tag{4.31}$$

The following lemma gives an easy representation for a possible TT-structure of C that does not require any arithmetic operations.

Lemma 4. *Given the index set $\mathcal{I} = \{1, \dots, m-1, m+1, \dots, d\}$ and a tensor $A \in \mathbb{R}^{N^{\mathcal{I}}}$ in TT-format with ranks $r_0^A, \dots, r_{m-1}^A, r_{m+1}^A, \dots, r_d^A$, the tensor $C \in \mathbb{R}^{N^{\mathcal{M}}}$ with $\mathcal{M} = \{1, \dots, d\}$ and slices*

$$C_i^k = \begin{cases} A_i^k & , \text{ if } k \neq m, \\ I_{r_m^A, r_m^A} & , \text{ if } k = m \end{cases}$$

with $r_m^A := r_{m-1}^A$ and identity matrix $I_{r_m^A, r_m^A} \in \mathbb{R}^{r_m^A \times r_m^A}$ satisfies (4.31).

Proof. The proposition follows from

$$C_{\mathbf{i}^{\mathcal{M}}} = \prod_{k=1}^{m-1} A_{i_k}^k \cdot I_{r_m^A, r_m^A} \cdot \prod_{k=m+1}^{d} A_{i_k}^k = A_{\mathbf{i}^{\mathcal{I}}}.$$

□

We note that the inserted core is not column-orthogonal because the columns of C_{col}^m do not have unit-length. Orthogonality can be re-established by scaling the core with a factor of $\frac{1}{\sqrt{n_m}}$. We balance this by multiplying the last, not orthogonal core, C^d with $\sqrt{n_m}$.
If $m = d$ (i.e. the core is appended at the rightmost position), a single orthogonalization is necessary. We note that this step is very cheap, as $A_{\text{col}}^{d-1} \in \mathbb{R}^{r_{d-1} n_{d-1} \times 1}$ has only a single column.

4.2.7 Summation of tensor elements

The summation of tensor elements will be used for a contraction with respect to certain dimensions. Here we focus on the contraction with respect to a single dimension m. Given a tensor $A \in \mathbb{R}^N$ in TT-format, we again define the index set $\mathcal{I} := \{1, \dots, m-1, m+1, \dots, d\}$ and a tensor $S \in \mathbb{R}^{N^{\mathcal{I}}}$ by summation along the dimension specified by m,

$$S_{\mathbf{i}^{\mathcal{I}}} := \sum_{j=0}^{n_m - 1} A_{i_1 \dots, i_{m-1}, j, i_{m+1}, \dots, i_d}.$$

We get a tensor with one dimension less in a process similar to a row/column summation of a matrix. In TT-format we can write

$$S_{\mathbf{i}^{\mathcal{I}}} = \sum_{j=0}^{n^m - 1} A_{i_1}^1 \cdots A_j^m \cdots A_{i_d}^d = \Big(\prod_{q=1}^{m-1} A_{i_q}^q \Big) \cdot \Big(\sum_{j=0}^{n^m - 1} A_j^m \Big) \cdot \Big(\prod_{q=m+1}^{d} A_{i_q}^q \Big).$$

The middle summation extends over all slices of the m-th core and yields a single matrix which can be multiplied with each slice of the preceding (or succeeding) core to produce S in TT-format. This procedure is of complexity $\mathcal{O}(n_m R_S^2)$ for the inner summation of the slices of the m-th core and $\mathcal{O}(n_{m-1} R_S^3)$ or $\mathcal{O}(n_{m+1} R_S^3)$ for the matrix multiplication with the preceding or succeeding core. We note that the orthogonality of the neighboring core is generally not upheld but can be re-established by the procedure from subsection 4.2.2. This is cheap because R_A does not increase in this summation and might even decrease.
This procedure can be extended to multiple dimensions by summation of all slices of every core individually before multiplying the resulting matrices to a neighboring core. An application of this procedure to all dimensions yields a sequence of matrix-matrix multiplications.
The complete summation of all tensor entries can be achieved with an arithmetic complexity of $\mathcal{O}(R_S^2 n_\Sigma)$ for the inner summations of slices and $\mathcal{O}(R_S^2 d)$ for the final matrix-vector multiplications.

Frobenius norm of a tensor

The calculation of the Frobenius norm $\|A\|_F^2 = \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n-1}} A_{\mathbf{i}}^2$ of a tensor $A \in \mathbb{R}^N$ is closely related to the summation of elements and can naively be computed by a complete summation of a Hadamard product of A with itself. But this requires the storage of a large intermediate result and an increase of the complexity of the summation due to high ranks.
A more efficient approach relies on ${A_{\mathbf{i}}}^2 = A_{\mathbf{i}}^T \cdot A_{\mathbf{i}}$ via the transpose of a scalar as

$$\begin{aligned} \|A\|_F^2 &= \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n-1}} A_{\mathbf{i}}^T \cdot A_{\mathbf{i}} \\ &= \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{n-1}} {A_{i^d}^d}^T \cdots {A_{i^1}^1}^T \cdot F_0 \cdot A_{i^1}^1 \cdots A_{i_d}^d, \end{aligned} \tag{4.32}$$

where we set $F_0 := 1$. We sort the N summands of (4.32) and group them by $(i_2, \ldots, i_d)$ and factor $\prod_{m=d}^{2} {A_{i_m}^m}^T$ to the left and $\prod_{m=2}^{d} A_{i_m}^m$ to the right. This leaves $F_1 := \sum_{i_1=0}^{n_1-1} {A_{i_1}^1}^T F_0 A_{i_1}^1 \in \mathbb{R}^{r_1 \times r_1}$, that all N/n_1 summands have in common. We iterate this process of grouping the latter indices and define

$$F_m = \sum_{i_m=1}^{n_m-1} {A_{i_m}^m}^T F_{m-1} A_{i_m}^m \in \mathbb{R}^{r_m \times r_m}$$

recursively. We then have $\|A\|_F = \sqrt{F_d} \in \mathbb{R}^{1\times 1}$ to compute the Frobenius norm with a complexity of $\mathcal{O}(n_\Sigma R_A^3)$.

If A is given in a row-orthogonal representation we can skip the calculations of F_0 to F_{d-1} because they are identities by design. Only the calculation of F_d is necessary which is of complexity $\mathcal{O}(n_d r_{d-1}^A)$. The same argument can be applied to a column-orthogonal tensor to only calculate F_1.

4.2.8 Splitting of a tensor

It will be beneficial to define a split of a tensor train A into two separate TT-structures G and K of lower dimensions.
We choose a split index $\tilde{d} < d$ and define index sets $\mathcal{I} := \{1, \dots, \widetilde{d}\}$ and $\mathcal{H} := \{\widetilde{d}+1, \cdots, d\}$ of dimensions associated with the first and the second set, respectively.
We use $\hat{r} := r_{\tilde{d}}$ to refer to the rank of A where the split takes place. We define tensors $G \in \mathbb{R}^{N^{\mathcal{I}} \times \hat{r}}$ and $K \in \mathbb{R}^{\hat{r} \times N^{\mathcal{H}}}$ in TT-format using cores

$$G^m := A^m,\ m \in \mathcal{I}, \quad \text{and} \quad G^{\text{last}} \in \mathbb{R}^{\hat{r} \times \hat{r} \times 1}, \tag{4.33}$$

$$K^{\text{first}} \in \mathbb{R}^{1 \times \hat{r} \times \hat{r}}, \quad \text{and} \quad K^m := A^m,\ m \in \mathcal{H}, \tag{4.34}$$

respectively, where K^{first} and G^{last} are essentially identity matrices, i.e. defined by $K^{\text{first}}_{1,i,j} = G^{\text{last}}_{i,j,1} = \delta_{ij}$.

Lemma 5. *Let $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a tensor in TT-format, and let tensors G, K in TT-format be defined as in (4.33), (4.34) for a split index $\widetilde{d}$. Then there holds*

$$A_{\mathbf{i}} = \sum_{\gamma=1}^{\hat{r}} G_{\mathbf{i}^{\mathcal{I}},\gamma} \cdot K_{\gamma,\mathbf{i}^{\mathcal{H}}}. \tag{4.35}$$

Proof. Let $e_\gamma \in \mathbb{R}^{\hat{r}}$ denote the γ-th unit vector. Starting from the right hand side of (4.35), there holds

$$\sum_{\gamma=1}^{\hat{r}} G_{\mathbf{i}^{\mathcal{I}},\gamma} \cdot K_{\gamma,\mathbf{i}^{\mathcal{H}}} = \sum_{\gamma=1}^{\hat{r}} \Big(\Big(\prod_{m=1}^{\tilde{d}} A^m_{i_m} \Big) \cdot e_\gamma \Big) \Big(e_\gamma^T \cdot \Big(\prod_{m=\tilde{d}+1}^{d} A^m_{i_m} \Big) \Big).$$

This summation can be seen as a scalar product of two vectors

$$G_{\mathbf{i}^{\mathcal{I}}} := \prod_{m=1}^{\tilde{d}} A^m_{i_m} \in \mathbb{R}^{1 \times \hat{r}}, \quad K_{\mathbf{i}^{\mathcal{H}}} := \prod_{m=\tilde{d}+1}^{d} A^m_{i_m} \in \mathbb{R}^{\hat{r} \times 1},$$

leading to

$$\sum_{\gamma=1}^{\hat{r}} G_{\mathbf{i}^{\mathcal{I}},\gamma} \cdot K_{\gamma,\mathbf{i}^{\mathcal{H}}} = G_{\mathbf{i}^{\mathcal{I}}} K_{\mathbf{i}^{\mathcal{H}}} = \prod_{m=1}^{\tilde{d}} A^m_{i_m} \prod_{m=\tilde{d}+1}^{d} A^m_{i_m} = \prod_{m=1}^{d} A^m_{i_m} = A_{\mathbf{i}}$$

which finishes the proof. □

This allows the split of a tensor A in TT-format into two separate TT-tensors of smaller dimensions with no arithmetic operations.
If A is given in a column-orthogonal representation, then G and K directly inherit the orthogonality. The added core G^{last} is not relevant for the column-orthogonality of G and K^{first} is column-orthogonal by definition.

4.2.9 Permutation of dimensions

The permutation of dimensions of a tensor $A \in \mathbb{R}^N$, comparable to the transposition of a matrix, is easily done in a full tensor format but more complicated in TT-format and will be discussed next.
The permutation of dimensions in TT-format can be done by iteratively interchanging two adjacent indices (similar to bubblesort) using a modification of Algorithm 2 to compute a TT-approximation of a full tensor.
Here we consider the exchange of a single index pair (i_m, i_{m+1}) of the tensor A with cores A^m to find cores B^m of B such that $B_{\dots i_{m+1}, i_m, \dots} = A_{\dots i_m, i_{m+1}, \dots}$.
It turns out that we can set $B^k = A^k$ for all $k \notin \{m, m+1\}$ and only need to "work" to find B_m and B_{m+1} which is described next.
All information from A^m and A^{m+1} is contained in $C := A^m_{\text{col}} \oplus A^{m+1}_{\text{row}}$. The computation of the tensor element $A_{\dots i_m, i_{m+1}, \dots}$ requires the matrix-matrix product $A^m_{i_m} A^{m+1}_{i_{m+1}}$ which is found in the i_m'th block row and i_{m+1}'st block column of the matrix C. A tensor B with $B_{\dots i_{m+1}, i_m, \dots} = A_{\dots i_m, i_{m+1}, \dots}$ should therefore have cores B^m, B^{m+1} such that its slices satisfy $B^m_{i_{m+1}} B^{m+1}_{i_m} = A^m_{i_m} A^{m+1}_{i_{m+1}}$, which leads to the problem of finding slices B^m_j, B^{m+1}_j such that

$$B^m_{\text{col}} \oplus B^{m+1}_{\text{row}} \overset{!}{=} \underbrace{\begin{bmatrix} A^m_1 A^{m+1}_1 & \cdots & A^m_{n_m} A^{m+1}_1 \\ \vdots & \ddots & \vdots \\ A^m_1 A^{m+1}_{n_{m+1}} & \dots & A^m_{n_m} A^{m+1}_{n_{m+1}}. \end{bmatrix}}_{=:D}$$

The slices of B^m and B^{m+1} result from a low-rank representation of the matrix D which can be computed (in an expensive way) via a singular value decomposition. More efficient ways could be pursued based upon adaptive cross approximation or random sampling approaches.
We note that the rank r^B_m of B can change in this process and an error is made during this permutation. The same error bound τ_m from (4.21) can be used here as the argument is identical.

4.3 Truncation of ranks

We have seen in section 4.2 that the low computational complexity and storage requirements of a tensor $A \in \mathbb{R}^N$ in the TT-format rely on the inner ranks $R_A \geq r^A_m$ to be small. Similar to the initial projection from subsection 4.2.1 we can reduce the ranks by replacing cores by approximate factors with a lower

rank. The basic algorithm follows from an argument similar to the one used in subsection 4.2.2 and was first introduced in [54]. We replace neighboring cores A^m and A^{m+1} by new cores B^m and B^{m+1} with a possibly lower rank $r_m^B \leq r_m^A$.
Let A have cores A^1 to A^d. We compute the truncated singular value decomposition of $A^1_{\text{col}} = USV^T + E^1$ and set the new core $B^1_{\text{col}} = U$ and multiply $SV^T \in \mathbb{R}^{r_1^B \times r_1^A}$ to each slice of A^2 to eliminate r_1^A and replace it by a (possibly) smaller r_1^B. Doing this for $m = 1, \ldots, d-1$ gives B in a column-orthogonal representation (as the new cores B^m are column-orthogonal by construction) with lower ranks r_m^B and reduced storage complexity. We can similarly iterate this process from right to left by finding the row-wise expression for the core in V^T and multiplying US to the preceding core.
This rank-reduction comes with a loss of accuracy as the errors (represented by E^m for each core) accumulate in a passage from left to right. Doing this, we obtain $m-1$ error-tensors T^m of the form

$$T^m = B^1_{\text{col}} \oplus \ldots \oplus B^{m-1}_{\text{col}} \oplus E^m \oplus A^{m+1}_{\text{row}} \oplus \ldots A^d_{\text{row}} \text{ for } m \in \{1, \ldots d-1\}. \tag{4.36}$$

If we assume A to be row-orthogonal (i.e., A^2 to A^d are row-orthogonal) we can compute

$$\|A - B\|_F^2 = \|\sum_{m=1}^{d-1} T^m\|_F^2 = \sum_{m=1}^{d-1} \|T^m\|_F^2 \tag{4.37}$$

$$= \sum_{m=1}^{d-1} \|E^m\|_F^2 = \sum_{m=1}^{d-1} \tau_m^2 \tag{4.38}$$

for the total error with the same argument based on orthogonality (splitting the summation in (4.37)) and unitary invariance already used in section 4.2.1. The equality between (4.37) and (4.38) requires the cores A^2 to A^d to be row-orthogonal. With this we can prescribe the same accuracy $\tau \leq \frac{\epsilon}{\sqrt{d-1}}\|A\|_F$ as in (4.21) for the iterated decompositions. We summarize this in Algorithm 3. This procedure is based on the assumption, that A is represented row-orthogonal. The orthogonalization via the procedure presented in subsection 4.2.2 of $d-1$ cores is of complexity $\mathcal{O}(R_A^3 n_\Sigma)$ which can be very high if A is the result of an elementwise multiplication or convolution and R_A is large.
A different approach to compute a truncation was presented in [60], where completely new cores are formed by using the TT-RC algorithm together with random sampling or cross-approximation. We, however, propose a different truncation-strategy based on singular value decompositions.

4.3.1 Improved truncation by estimation of core norms

We improve the above mentioned algorithm in order to eliminate the necessity of a row-orthogonal representation of A. The algorithm is based on the

Algorithm 3 Truncation of TT-ranks with errorbound ϵ from left to right

Input: TT-representation of $A \in \mathbb{R}^N$ with row-orthogonal cores A^2 to A^d and error bound $\epsilon > 0$

Output: Cores B^1 to B^d with ranks $r_m^B \leq r_m^A$ with $\|A - B\|_F \leq \epsilon \|A\|_F$

Calculate $\tau = \frac{\epsilon}{\sqrt{d-1}} \cdot \|A\|_F$

for $m = 1$ to $d - 1$ **do**

Calculate truncated SVD of $A^m_{\text{col}} = USV^T + E^m$ with $\|E^m\|_F < \tau$

Set $B^m_{\text{col}} = U$

Set $A^{m+1}_{\text{row}} = SV^T \cdot A^{m+1}_{\text{row}}$

end for

Set $B^1 = A^1$

following theorem.

Theorem 2. *Let $A \in \mathbb{R}^N$ be a tensor with a TT-representation with cores A^m with $\|A^m_{col}\|_2 < \xi_m$ known for $m \in \{1, \ldots, d-1\}$. Then a sequence of truncated singular value decompositions from right to left with upper limits on the errors*

$$\|E_m\|_F \leq \frac{\epsilon}{\sqrt{d-1} \cdot \prod_{k=1}^{m} \xi_m} \cdot \|A\|_F$$

gives a tensor B with $\|A - B\|_F \leq \epsilon \|A\|_F$

Proof. The proof is very much based on the reasoning behind Algorithm 3. We again look at the overall error and find

$$\|A - B\|_F^2 = \| \sum_{m=2}^{d} T^m \|_F^2 \tag{4.39}$$

with error-tensors

$$T^m = A^1_{\text{col}} \oplus \ldots \oplus A^{m-1}_{\text{col}} \oplus E^m \oplus B^{m+1}_{\text{row}} \oplus \ldots \oplus B^d_{\text{row}},$$

where E^m is the truncation error made in the m-th singular value decomposition for $m \in \{2, \ldots, d\}$.
The errors are orthogonal by construction (all E_m are orthogonal to B^m_{row}) and allow for a separation of the summation in (4.39) into

$$\|A - B\|_F^2 = \sum_{m=2}^{d} \|T^m\|_F^2.$$

We then use the unitary invariance of the Frobenius norm and write

$$\begin{aligned} \|T^m\|_F &= \|A^1_{\text{col}} \oplus \ldots \oplus A^{m-1}_{\text{col}} \oplus E^m \oplus B^{m+1}_{\text{row}} \oplus \ldots B^d_{\text{row}}\|_F \\ &\leq \|A^1_{\text{col}} \oplus \ldots \oplus A^{m-1}_{\text{col}}\|_2 \cdot \|E^m\|_F \cdot \|B^{m+1}_{\text{row}} \oplus \ldots \oplus B^d_{\text{row}}\|_2 \end{aligned}$$

and see the error being scaled by two large matrices. The second line is an upper limit, as $A^1_{\text{col}} \oplus \ldots \oplus A^{m-1}_{\text{col}}$ is generally not orthogonal.
We use Lemma 3 to further separate the matrices on the left and right hand side of $\|E^m\|_F$. We then obtain

$$\|T^m\|_F \leq \left(\prod_{k=1}^{m} \xi_k \right) \cdot \|E^m\|_F \cdot 1.$$

Using this estimation of $\|T^m\|_F$ in (4.21) gives the desired upper bound for $\|E^m\|_F$. □

This theorem allows us to skip the costly computation of large orthogonal cores in exchange for tighter bounds and an earlier and more expensive calculation of $\|A\|_F$. We will use the estimates derived for addition, Hadamard-product and convolution for this procedure. These are based on extremal cases to attain the maximum norm of a core and will usually maximize only one singular value (instead of all being maximized as needed for equality in Lemma 3).
To attain (nearly) optimal ranks we will follow up with an additional truncation from left to right that uses the orthogonality of all cores to obtain a tensor C. To still guarantee $\|A - C\|_F \leq \epsilon \|A\|_F$ we allow a *half step* for each of the two truncation-sweeps, allowing only half the maximal error in each truncation. This gives the total error bound for the first truncation with

$$\tau^m \leq \frac{\epsilon \cdot \|A\|_F}{2 \cdot \sqrt{d-1} \cdot \prod_{k=1}^{m} \xi_k}.$$

4.4 Application to population balance equations

This section is devoted to the application of the storage technique presented in this chapter to the problem setting of PBEs and aggregation in particular. The TT-storage was already applied to PBEs in [14] where it was used in a black-box approach. To go into more detail here we use the discretization introduced in section 3.1 with a uniform tensor grid and piecewise constant functions. The simplifications of the integrals introduced in 3.3 are expanded upon as the efficient storage format allows for further improvements.

4.4.1 Kernel separability

In section 3.2 we introduced the requirement of kernel separability by assuming

$$\kappa(\mathbf{u}, \mathbf{v}) = \sum_{\nu=1}^{k} \alpha^\nu(\mathbf{u}) \cdot \beta^\nu(\mathbf{v})$$

which allowed for the subsequent split of the source and sink terms into (3.9) and (3.10).
We focus here on the discrete case

$$\kappa_{\mathbf{i},\mathbf{j}} = \sum_{\nu=1}^{k} \alpha_{\mathbf{i},\nu} \cdot \beta_{\mathbf{j},\nu}$$

with kernel-factors α and β. A TT-representation of $\kappa \in \mathbb{R}^{N\times N}$ already fulfills the assumption of separability. The kernel factors α, β can be obtained by the procedure introduced in subsection 4.2.8 by splitting after the d-th core. As these are not k independent tensors in $\mathbb{R}^N$ we use ν as sub-index (in opposition to the super-index used in (3.4) to denote k different functions).
The resulting tensors $\alpha \in \mathbb{R}^{N\times k}$ and $\beta \in \mathbb{R}^{k\times N}$ differ in their ordering of cores. The artificial core (associated with the variable ν) is the last core of α while it is the first core of β. We permute the dimension of β to also have ν as the last index and the associated core on the right. This permutation allows us to use arithmetic operations like addition and elementwise multiplication in a straight forward way.
We again note that this permutation is relatively expensive as it involves low-rank factorizations of multiple (large) matrices $D \in \mathbb{R}^{r^{\beta}_{m-1} n_{m+1} \times r^{\beta}_{m+1} n_m}$, but we only need to do this permutation once because we assume the kernel to be time-independent and allow a one-time high complexity for optimal ranks. The assumption of a good TT-approximation is stronger than just separability as we assume separability at every dimension of the tensor with $2d$ dimensions.

4.4.2 Evaluation of the source term

We begin this subsection by recalling the discrete source term given by

$$Q_{\mathrm{pl}}^{\mathrm{source}}(\mathbf{g_{i+1}}) = \frac{V}{2} \sum_{\nu=1}^{k} \sum_{\mathbf{j=0}}^{\mathbf{i}} \alpha_{\mathbf{k},\nu} f_{\mathbf{k}} \cdot \beta_{\mathbf{i-k},\nu} f_{\mathbf{i-k}}.$$

We define tensors $\phi, \psi \in \mathbb{R}^{N\times k}$ with entries

$$\phi_{\mathbf{i},\nu} = \alpha_{\mathbf{i},\nu} \cdot f_{\mathbf{i}} \text{ and } \psi_{\mathbf{i},\nu} = \beta_{\mathbf{i},\nu} \cdot f_{\mathbf{i}}$$

and (in order to calculate them via the procedure from section 4.2.4) we have to add a trivial core to f (see subsection 4.2.6) in order to have the same number of cores with the same number of slices in f, α and β.
We then write

$$Q_{\mathrm{pl}}^{\mathrm{source}}(\mathbf{g_{i+1}}) = \frac{V}{2} \sum_{\nu=1}^{k} \sum_{\mathbf{j=0}}^{\mathbf{i}} \phi_{\mathbf{k},\nu} \cdot \psi_{\mathbf{i-k},\nu}$$

and calculate the convolution of all summands at the same time. This is possible by the addition of the artificial core instead of copying the cores to separate tensors. We only calculate the convolution of these cores once in every time step.
We denote the resulting tensor by $Q \in \mathbb{R}^N$ with $Q_{\mathbf{i}} = Q_{\mathrm{pl}}^{\mathrm{source}}(\mathbf{g_i})$.

Truncation of convolution results

The tensor $Q \in \mathbb{R}^N$ typically has the largest internal rank R_Q of every intermediate result as the necessary calculations (Hadamard product and convolution) in TT-format will multiply the ranks every time and we obtain an upper bound $R_Q \leq R_\alpha R_\beta R_f^2$. This makes the truncation of a convolution result an important and costly procedure. The improvement introduced in section 4.3.1 allows to skip the orthogonalization of high-rank cores.
The calculation of $\|Q\|_F$ is still of high complexity and will often dominate the computational time. If we have a lower bound $M \leq \|Q\|_F$ we are able to truncate some ranks in a first pass before calculating the Frobenius norm.
A possible estimation for $\|Q\|_F$ in the univariate case of $d = 1$ is given in the following Lemma.

Lemma 6. *Let $\widetilde{\phi}, \widetilde{\psi} \in \mathbb{R}^{2n \times k}$ be zero-padded extensions of $\phi, \psi \in \mathbb{R}^{n \times k}$, the low-rank factors of a symmetric matrix based on the eigenvalue decomposition, made from column vectors $\phi^\nu, \psi^\nu \in \mathbb{R}^{2n}$ for $\nu = 1, \ldots, k$. We define the (unshifted) convolution result $\omega \in \mathbb{R}^{2n}$ as $\omega_i = \sum_{\nu=1}^{k} \omega_i^\nu$ with vector convolutions $\omega_i^\nu := \sum_{j=0}^{i} \widetilde{\phi}_j^\nu \cdot \widetilde{\psi}_{i-j}^\nu \in \mathbb{R}$ with $i \in \{0, \ldots, 2n-1\}$ and $\nu \in \{1, \ldots, k\}$. Then*

$$\|\omega\|_F^2 \geq \sum_{\nu=1}^{k} \|\phi^\nu\|_2^2 \cdot \|\psi^\nu\|_2^2 \tag{4.40}$$

holds.

Proof. We begin this proof by separating the summations into individual kernel factors ν. If $\|\omega^\nu\|_F^2 \geq \|\phi^\nu\|_2^2 \cdot \|\psi^\nu\|_2^2$ holds then (4.40) holds as well by the triangle inequality. As ν can be treated as a constant, we drop it to reduce notation.
We have the convolution result

$$\omega = \begin{bmatrix} \phi_0\psi_0 \\ \phi_0\psi_1 + \phi_1\psi_0 \\ \phi_0\psi_2 + \phi_1\psi_1 + \phi_0\psi_2 \\ \cdots \\ \phi_{n-1}\psi_{n-1} \end{bmatrix}$$

and express its squared norm by

$$\begin{aligned}\|\omega\|_F^2 = &\phi_0^2\psi_0^2 \\ &+ \phi_0^2\psi_1^2 + 2\phi_0\phi_1\psi_0\psi_1 + \phi_1^2\psi_0^2 \\ &+ \phi_0^2\psi_2^2 + 2\phi_0\phi_2\psi_0\psi_2 + 2\phi_0\phi_1\psi_1\psi_2 + 2\phi_0\phi_2\psi_0\psi_2 + \phi_1^2\psi_1^2 + \phi_2^2\psi_0^2 \\ &+ \cdots \\ &+ \phi_{n-1}^2\psi_{n-1}^2\end{aligned}$$

as we expand the squared summations.
Observing the right hand side of the inequality we similarly expand the product of (squared) norms into

$$\begin{aligned}\|\phi\|_2^2\|\psi\|_2^2 = &\phi_0^2\psi_0^2 \\ &+ \phi_0^2\psi_1^2 + \phi_1^2\psi_0^2 \\ &+ \cdots \\ &+ \phi_{n-1}^2\psi_{n-1}^2.\end{aligned}$$

Subtracting these expressions and dividing by 2 leaves only the mixed terms of the expansion and yields

$$\phi_0\phi_1\psi_0\psi_1 + \phi_0\phi_2\psi_0\psi_2 + \phi_0\phi_1\psi_1\psi_2 + \cdots \geq 0 \tag{4.41}$$

to be shown. We will prove this by rewriting the left hand side of (4.41) as bilinear form with a positive semi-definite matrix.
All terms $\phi_i\phi_k\psi_j\psi_l$ in (4.41) obey $i+l = k+j \Leftrightarrow k-i = l-j$ by the definition of ω as a convolution. We then can group them by gathering all terms with a constant $k-i$ together, this gives (4.41) as

$$\underbrace{\begin{matrix}\phi_0\phi_1\psi_0\psi_1+\phi_0\phi_1\psi_1\psi_2+\ldots \\ \phi_1\phi_2\psi_0\psi_1+\phi_1\phi_2\psi_1\psi_2+\ldots\end{matrix}}_{k-i=1} + \underbrace{\begin{matrix}\phi_0\phi_2\psi_0\psi_1+\phi_0\phi_2\psi_1\psi_2+\ldots \\ \phi_1\phi_3\psi_0\psi_1+\phi_1\phi_3\psi_1\psi_2+\ldots\end{matrix}}_{k-i=2} + \ldots \geq 0$$

The combinations with $i = k$ (and $j = l$) are not present as they are already subtracted. With vectors

$$v_\phi = \begin{bmatrix}\phi_0\phi_1 \\ \phi_1\phi_2 \\ \cdots \\ \phi_{n-2}\phi_{n-1} \\ \phi_0\phi_2 \\ \cdots \\ \phi_0\phi_{n-1}\end{bmatrix} \in \mathbb{R}^{\frac{n^2+n}{2}} \text{ and } v_\psi = \begin{bmatrix}\psi_0\psi_1 \\ \psi_1\psi_2 \\ \cdots \\ \psi_{n-2}\psi_{n-1} \\ \psi_0\psi_2 \\ \cdots \\ \psi_0\psi_{n-1}\end{bmatrix} \in \mathbb{R}^{\frac{n^2+n}{2}}$$

and a symmetric positive semi-definite block-diagonal matrix

$$A = \begin{bmatrix} J_n & & 0 \\ & \ddots & \\ 0 & & J_1 \end{bmatrix} \text{ with } J_k = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix} \in \mathbb{N}^{k \times k}.$$

we now rewrite this as a bilinear form

$$v_\phi^T A v_\psi \geq 0 \tag{4.42}$$

The vectors ϕ and ψ are given to be scaled eigenvectors which means they are equal up to a scalar factor λ, i.e. $\phi = \lambda\psi$, which in turn gives us $v_\phi = \lambda^2 v_\psi$. We then replace v_ϕ in (4.42) to have

$$v_\psi^T (\lambda^2 A) v_\psi \geq 0$$

with a positive semi-definite matrix $\lambda^2 A$. An expression of this form will always be non-negative which proves the assumption from (4.41). □

We construct α and β as (scaled) singular values of the symmetric kernel matrix κ. By elementwise multiplication with f, we obtain ϕ and ψ, the (scaled) singular vectors of $\kappa \odot (f \cdot f^T)$, a symmetric matrix. The singular vectors and eigenvectors of a symmetric matrix are equal up to a sign, which can be included in λ and will cancel out in λ^2. A simple observation allows us to carry this result over to the multivariate case.

Corollary 1. *Let $\widetilde{\phi}, \widetilde{\psi} \in \mathbb{R}^{2N \times k}$ be zero-padded low-rank factors of a symmetric tensor and define their (unshifted) partial convolution result as*

$$\omega_{\mathbf{i}}^\nu := \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{i}} \widetilde{\phi}_{\mathbf{j}}^\nu \cdot \widetilde{\psi}_{\mathbf{i}-\mathbf{j}}^\nu.$$

with $\widetilde{\phi}^\nu, \widetilde{\psi}^\nu \in \mathbb{R}^{2N}$ making up $\widetilde{\phi}$ and $\widetilde{\psi}$. Then

$$\|\omega\|_F^2 \geq \sum_{\nu=1}^{k} \|\phi^\nu\|_F^2 \cdot \|\psi^\nu\|_F^2$$

holds.

Proof. We note that every multivariate convolution can be expressed as a univariate convolution with an appropriate zero-padding in the vectorizations of ϕ and ψ. This padding has no influence on the Frobenius norm allowing us to invoke the results of Lemma 6. □

Remark 4. *This Lemma and corollary allow us to obtain a lower bound of the Frobenius norm of a full convolution result. The necessary restriction to the domain of $\mathcal{G}$ (see Remark 1) has to follow after a truncation. A restriction*

reduces the Frobenius norm of ω and can invalidate the result for some density distributions f (e.g. a density distribution dominated by large particles).

A similar result for additions and Hadamard-products can not be obtained. Two non-zero tensors can result in a zero-tensor in these operations (A and $-A$ for addition and particular zero-patterns for a Hadamard-product).

With a lower bound $M := \sqrt{\sum_{\nu=0}^{k} \|\phi^\nu\|_F^2 \|\psi^\nu\|_F^2} \leq \|Q\|_F$ and upper bounds $\xi^m > \|Q^m_{\text{col}}\|_2$ we are able to calculate a truncation without orthogonolizing or calculating a Frobenius norm. This allows us to delay the calculation of the cores of a convolution right before a truncation to reduce the memory requirements of intermediates (which are especially large in case of the convolution). Similar to the non-optimal truncation in section 4.3.1 we follow this first sweep with a second one with orthogonal cores (by design) and the correct Frobenius norm of the tensor, which is cheap to calculate now.
We summarize this in Algorithm 4. We note that we do not use the additional factor $\frac{1}{2}$ introduced in 4.3.1 for two half steps. The error introduced through the first truncations is guaranteed to be small due to over-estimation and we allow for the full truncation error for the second sweep of truncations as well. Algorithm 4 also handles the restriction to cores with n_m slices. This is done by deleting these slices after the first truncation and orthogonalization afterward.
This algorithm skips the computation of $\|Q\|_F$ but requires a truncation of cores with $2n_m$ slices. This trade-off is not always advantageous as the computational time is hard to quantify. Numerical tests and comparisons with the standard-approach will be shown in subsection 4.5.1.

Moment-preserving projection

We introduced the moment-preserving projection in subsection 3.3.3 and apply the results of Theorem 1 to the TT-format. The projection W with entries $w_{\mathbf{i}} = \prod_{m=1}^{d} W^m_{i_m}$ of a tensor Q with entries $q_{\mathbf{i}} = \prod_{m=1}^{d} Q^m_{i_m}$ can be written as

$$\begin{aligned} w_{\mathbf{i}} &= \frac{1}{2^d} \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{1}} q_{\mathbf{i}+\mathbf{k}} \\ &= \frac{1}{2^d} \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{1}} \prod_{m=1}^{d} Q^m_{i_m+k_m}. \end{aligned}$$

Algorithm 4 Convolution-truncation of tensors ϕ and ψ

Input: Column-orthogonal TT-representations of $\phi \in \mathbb{R}^{N \times k}$ and $\psi \in \mathbb{R}^{N \times k}$ and truncation accuracy $\epsilon > 0$

Output: Cores Q^1 to Q^d of a piecewise linear convolution result

Calculate core norm bounds $\xi_m = \sqrt{r_m^\phi \cdot r_m^\psi}$.

Calculate lower bound for Frobenius norm $M = \sqrt{\sum_{\nu=1}^{k} \|\phi^\nu\|_F \cdot \|\psi_\nu\|_F}$

// No convolution w.r.t. artificial dimension

Compute $\omega = \phi^{d+1} \otimes \psi^{d+1}$

Make ω row-orthogonal by extracting R_{Trunc}

Set $L_{\text{Restr}} = \sum_{\nu=1}^{k} \omega_\nu$.

for $m = d$ down to 1 **do**

  Calculate $\omega^m = \phi^m * \psi^m$ via FFT with $2n_m$ slices

  Weigh result $C = \omega^m_{\text{row}} \cdot R_{\text{Trunc}}$

  Calculate allowed truncation error $\tau^m = \frac{\epsilon M}{\sqrt{d-1}} \cdot \prod_{k=1}^{m} \xi_k^{-1}$

  Determine truncanted SVD with $C = USV^T + E$ with $\|E\|_F \leq \tau^m$

  Set $R_{\text{Trunc}} = US$

  Restrict core $Q^m_{\text{col}} = V^T \cdot L_{\text{Restr}}$ to n_m slices

  Make Q^m row-orthogonal by extracting new L_{Restr}

end for

Set $Q^1 = Q^1 \cdot R_{\text{Trunc}} \cdot L_{\text{Restr}}$

// Q is orthogonal and had a non-optimal truncation

Calculate allowed truncation error $\tau = \frac{\epsilon}{\sqrt{d-1}} \cdot \|Q\|_F$

Set $R_{\text{Trunc}} = 1$

for $m = 1$ to $d - 1$ **do**

  Weigh core $C = R_{\text{Trunc}} \cdot Q^m_{\text{row}}$

  Determine truncated SVD with $C = USV^T + E$ with $\|E\|_F \leq \tau$

  Set $Q^m_{\text{row}} = U$ and $R_{\text{Trunc}} = SV^T$

end for

Weigh $Q^d_{\text{row}} = R_{\text{Trunc}} \cdot Q^d$

We use the same reasoning for expansion and factoring from subsection 4.2.7 to write

$$w_{\mathbf{i}} = \frac{1}{2^d} \prod_{m=1}^{d} \left(\sum_{k_m=0}^{1} Q^m_{i_m+k_m} \right)$$
$$W^m_{i_m} = \frac{1}{2} \cdot \left(Q^m_{i_m} + Q^m_{i_m+1} \right)$$

implying an algorithm of averaging two adjacent slices to form the slice of a projected core. The complexity of this algorithm is $\mathcal{O}(n_\Sigma R_Q^2)$ and does not increase the ranks.

4.4.3 Evaluation of the sink term

We begin this section by recalling the discrete sink term given by

$$Q^{\text{sink}}(\mathbf{v})|_{\mathcal{C}_\mathbf{i}} = V \cdot \sum_{\nu=1}^{k} f_i \beta_{\mathbf{i},\nu} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n-1}} f_{\mathbf{j}} \alpha_{\mathbf{j},\nu}.$$

We again use the definitions of ϕ and ψ to write

$$\begin{aligned} Q^{\text{sink}}(\mathbf{v})|_{\mathcal{C}_\mathbf{i}} &= V \cdot \sum_{\nu=1}^{k} \psi_{\mathbf{i},\nu} \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n-1}} \phi_{\mathbf{j},\nu} \\ &= V \cdot \sum_{\nu=1}^{k} \psi_{\mathbf{i},\nu} \cdot S_\nu. \end{aligned}$$

The inner sum can be calculated by the procedure introduced in subsection 4.2.7 and (formally) gives a tensor $S \in \mathbb{R}^k$ in the TT-format with a single core with k slices.
To calculate the multiplication we formally need to re-add d cores to S. Their slices are given by $I_{1,1} = 1$ as $R_S = 1$. We then calculate

$$Q^{\text{sink}}_{\mathbf{i},\nu} = \psi_{\mathbf{i},\nu} \cdot S_{\mathbf{i},\nu}.$$

by scaling every slice of ψ with the corresponding scalar in S. The first d cores are scaled by one and need no computation at all. The formal extension of S is not necessary.
This gives the complete sink term (already piecewise constant) after a summation over the last core of Q^{sink}.

4.5 Numerical results

We again are going to carry out three numerical simulations to show the merits of this approach. In subsection 4.5.1 we will examine the proposed improvement of the truncation algorithm from section 4.3. We also test the truncation

of a convolution result as presented in subsection 4.4.2.
We also replicate the tests from sections 3.5.1 and 3.5.2 with more particle properties. We do not compare with other discretization schemes but will alter the internal accuracy ϵ or the time-discretization respectively.

4.5.1 Computational benefit of the truncation improvement

The first series of numerical tests studies the influence of the proposed improvement of the truncation algorithm. For this we use the high-rank Brownian kernel

$$\kappa_B(\mathbf{u}, \mathbf{v}) = \sum_{m=1}^{d} \left(u_m^{1/3} + v_m^{1/3}\right) \cdot \left(u_m^{-1/3} + v_m^{-1/3}\right)$$

with $R = 2d + 1$. We combine this with the exponential initial distribution

$$f(\mathbf{v}, 0) = \prod_{m=1}^{d} e^{-0.05 v_m}$$

to use the same setup for the tests as we did in subsection 3.5.3.
We simulated several choices for d, n and ϵ for 200 timesteps with $\Delta t = 10^{-2}$. We are not interested in the density distribution and only record the computational time spent in the truncations after the calculations of additions, Hadamard products and convolutions, respectively, each with an accuracy of 10^{-3} seconds. We also log the computational time spent in the convolution itself to serve as a benchmark.
We do this once with the standard truncation algorithm and once for the proposed improvement via core norms from subsection 4.3.1. We also log the computational time for Algorithm 4 (grouped by time spent in the convolution and time spent in the truncation) and include them in the comparison for the truncation of a convolution and the convolution itself.
The reported times always include the calculation (or estimation) of the Frobenius norm and a sweep from right to left (either QR-decompositions for orthogonalization or SVDs for truncation, see Algorithm 3 and section 4.3.1) and left to right (always SVDs) for truncation. We will show histograms of the recoreded computational times where the two (or three if Algorithm 4 is included) distributions will be shown in the same axis to allow for an easy comparison.
We run the simulations with 4 different symmetric grids with $v^{\max} = 1$. These are

- a small grid $\mathcal{G}_S$ with $n = 128$ and $d = 2$, we are using $\epsilon = 10^{-7}$
- a medium grid $\mathcal{G}_M$ with $n = 512$ and $d = 5$, we are using $\epsilon = 10^{-10}$

- a high-dimensional grid $\mathcal{G}_D$ with $n = 128$ and $d = 10$, we are using $\epsilon = 10^{-10}$
- a fine discretization $\mathcal{G}_F$ with $n = 4096$ and $d = 2$, we are using $\epsilon = 10^{-7}$.

Truncation of additions

The addition of two tensors in the TT-format was presented in subsection 4.2.3. The internal ranks are added up and most cores of the results are already orthogonal if the operands were orthogonal themselves. There are two additions in every time step, one in the addition of Q^{source} and Q^{sink} and one to calculate the actual Euler-step. We add the required computational times together and show the distribution of the recoreded 200 times in Figure 4.2.

We first mention the short times required for this truncation. The longest

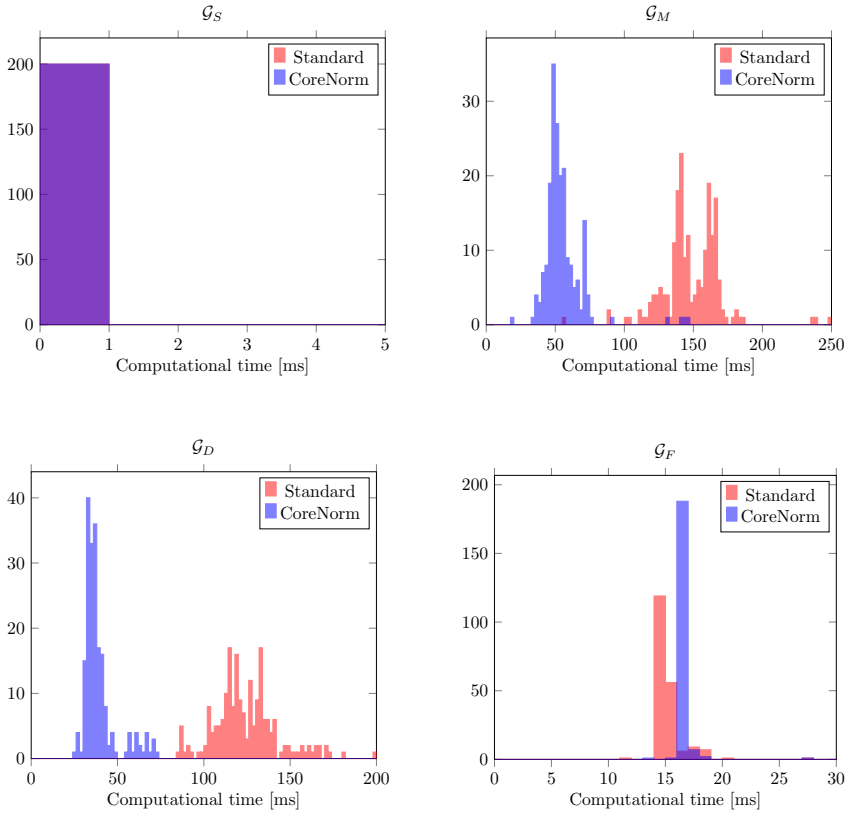


Figure 4.2: Histogram of the computational time for the truncation after additions on $\mathcal{G}_S$ (top left), $\mathcal{G}_M$(top right), $\mathcal{G}_D$ (bottom left) and $\mathcal{G}_F$ (bottom right) with the standard algorithm and the core norm truncation.

timing is just short of 250 milliseconds on $\mathcal{G}_M$. We see a slight improvement of the proposed approach in the case of $\mathcal{G}_M$ and $\mathcal{G}_D$ where the two distributions are clearly distinct and the proposed approach outperforms the standard

algorithm by a factor of at least 2. For a simulation with respect to $\mathcal{G}_F$ (on the bottom right) we see a slightly longer computational time for the proposed approach. The difference however is very small (the highest recoreded bins are 2 milliseconds apart with a time of about 15 ms) and seems insignificant. We were not able to record any compuational times for the small grid $\mathcal{G}_S$ as the computations were too fast to log any required time.

Truncation of Hadamard product

The Hadamard product was presented in section 4.2.4 and is used to calculate ϕ and ψ from (3.5) and (3.6). The inner ranks multiply and increase greatly making orthogonalization of large cores in the middle of ϕ and ψ computationally expensive. We add the required computational time for both Hadamard products and compute histograms based on these results. The resulting distributions can be seen in Figure 4.3.

The recoreded timings here suggest that the standard approach is better

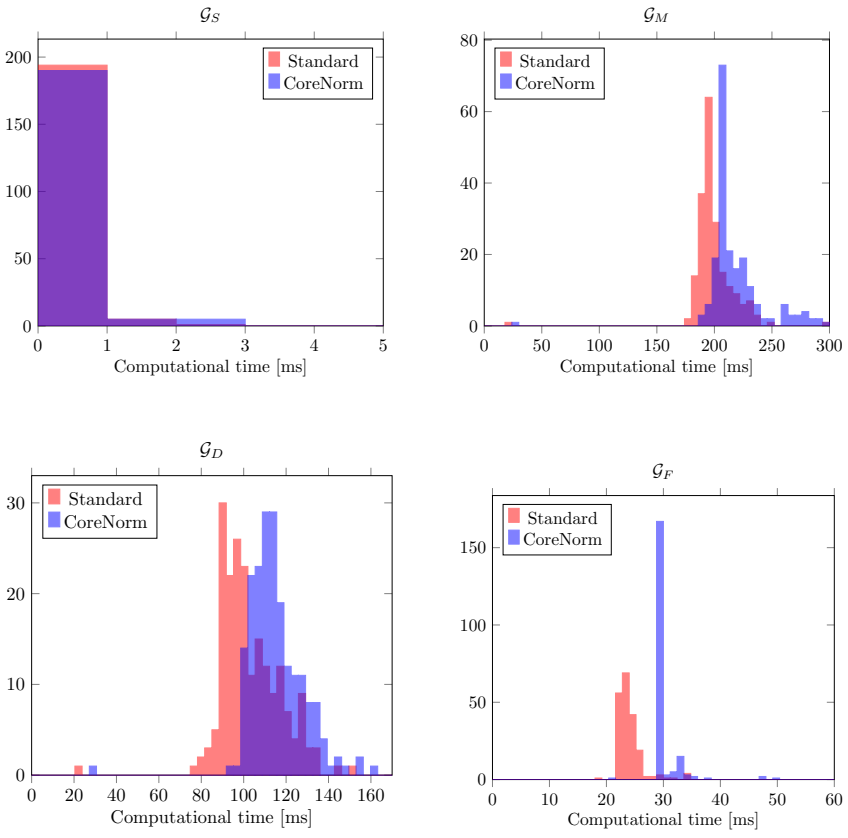


Figure 4.3: Histogram of the computational time for the truncation after Hadamard products on $\mathcal{G}_S$ (top left), $\mathcal{G}_M$(top right), $\mathcal{G}_D$ (bottom left) and $\mathcal{G}_F$ (bottom right) with the standard algorithm and the core norm truncation.

suited for the truncation of Hadamard products as we record shorter compu-

tational times in three of the four case studies (computations with respect to $\mathcal{G}_S$ are again too short to be recoreded properly). The difference is especially visible in the bottom right (grid $\mathcal{G}_F$) where the computational times of the newly proposed approach are very concentrated at below 30 millseconds. The standard approach shows significantly shorter computational times with the main cluster having a maximum at that value. The distirbutions for simulations with respect to $\mathcal{G}_D$ show more overlap as the distributions are wider spread with peaks between 80 and 120 milliseconds.

Truncation of convolution

The convolution and the estimation of the norms of the cores was presented in section 4.2.5. We also introduced the truncation of a convolution in section 4.4.2 based on the estimation of the Frobenius norm and include this method in the comparison. Each time step in an explicit Euler scheme requires a single convolution ($\widetilde{\omega}$, the convolution of $\widetilde{\phi}$ and $\widetilde{\psi}$) and this results usually has the highest rank R_ω of all intermediate results in each time step. This implies that this truncation requires more computational effort than the two previously shown. The reported computational times can be found in Figure 4.4.
We first notice that our assumption about this being the truncation with the highest effort is indeed true. A truncation of the convolution result requires significantly longer then a truncation after Hadamard products or addition. This is especially true for $\mathcal{G}_M$ and $\mathcal{G}_D$ where computational times is displayed in seconds and the longest computations taking over one minute for a single step.
We also see a significant improvement of the core-norm approach as the computational times (compared with the standard approach) are lower by a factor of 2 (seen in the bottom left) to 3 (seen in the top right and bottom right). The distributions show no overlap which again underlines the improvements of the new approach.
Algorithm 4 is not superior to the core norm approach as it takes longer than the core norm approach on three of the four grids (again excluding $\mathcal{G}_S$ here). It requires truncations of cores with $2n_m$ slices which is not compensated by the cheaper estimation of the Frobenius norm. On the high-dimensional grid $\mathcal{G}_D$ this approach even takes longer than the standard approach of truncation.

We show the computational times of the convolution itself (i.e., the Fourier transforms and the complex Hadamard product) in Figure 4.5 to give a benchmark to this computational time.
We see the computational times not differ too much between the three algorithms for truncation. This was to be expected as the convolutions are calculated the same across every approach. Larger differences (as seen in the bottom left) can be explained by different ranks after the truncation of the Hadamard product as errors are overestimated there.
When we compare the timings shown in Figure 4.5 to those in Figure 4.4 we

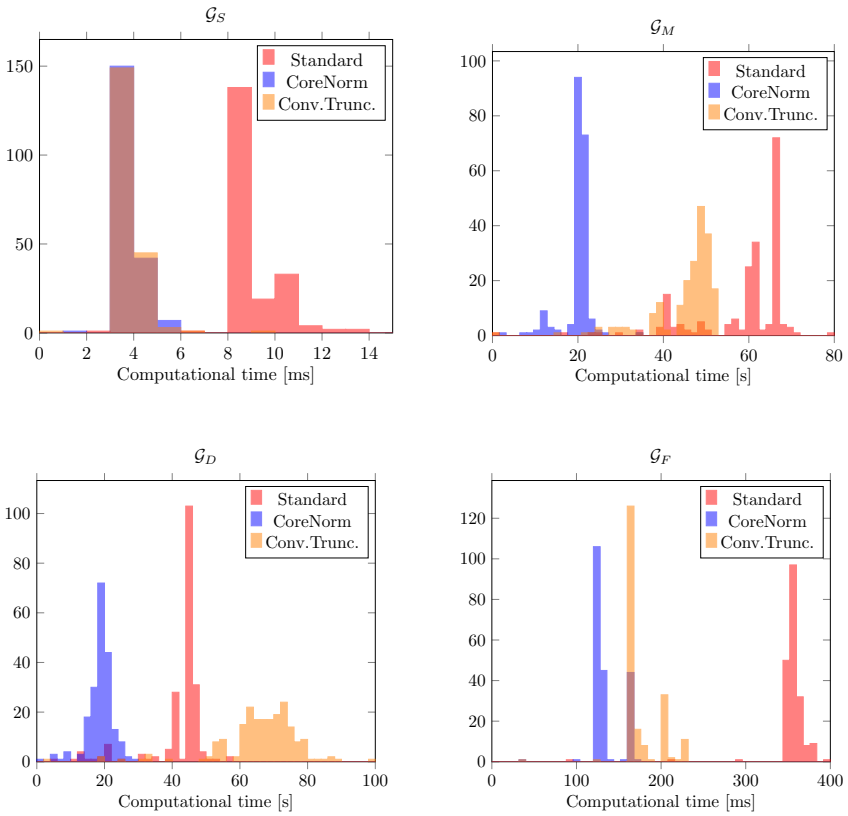


Figure 4.4: Histogram of the computational time for the truncation after a convolution on $\mathcal{G}_S$ (top left), $\mathcal{G}_M$(top right), $\mathcal{G}_D$ (bottom left) and $\mathcal{G}_F$ (bottom right) with the standard algorithm and the core norm truncation.

see that the time required for a truncation is of the same magnitude as the convolution beforehand. The truncation with the standard approach takes twice as long on both $\mathcal{G}_D$ and $\mathcal{G}_M$ compared to the convolution itself, which makes the gains due to core norms very significant to the overall computational time. We conclude that the core norm approach is very promising for the truncation of a convolution but is not beneficial for Hadamard products in the cases shown here. Further study into different grids and influence of ϵ can give further inside into this procedure.

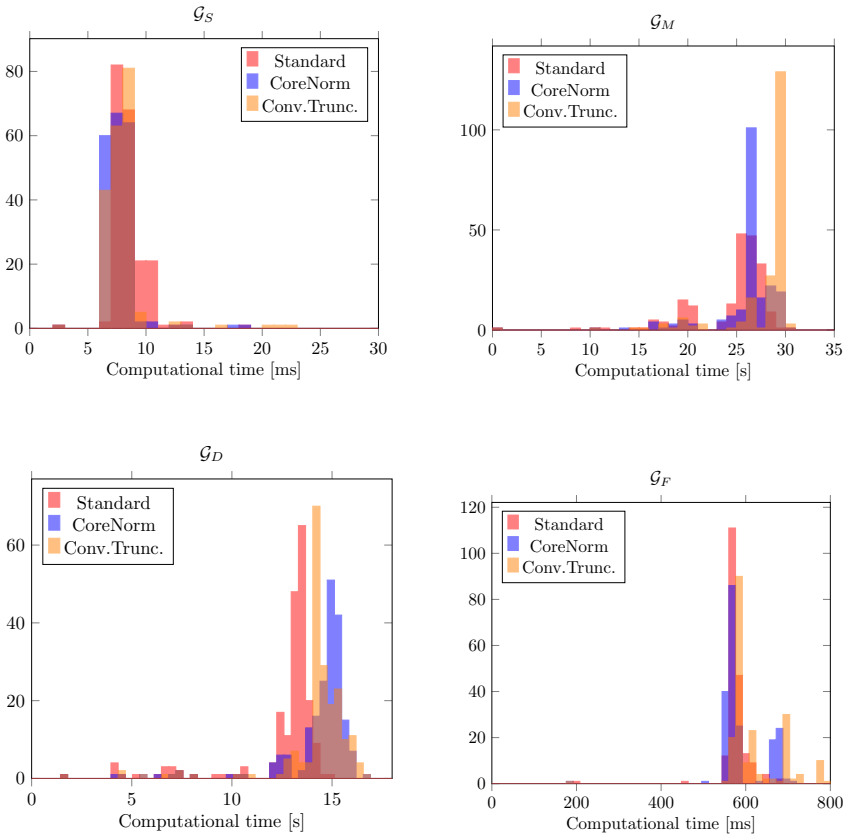


Figure 4.5: Computational time in ms/s for the convolutions on $\mathcal{G}_S$ (top left), $\mathcal{G}_M$(top right), $\mathcal{G}_D$ (bottom left) and $\mathcal{G}_F$(bottom right) with the truncation donw via the standard algorithm, the truncation based on core norms and the estimation of the Frobenius norm.

4.5.2 Accuracy of a population density approximation

In this subsection, we will repeat the numerical test from subsection 3.5.1 to assess the accuracy of a particle density distribution. We will use the same initial distribution (3.30)

$$f(\mathbf{v}, 0) = N_0 \cdot \prod_{m=1}^{d} \frac{4v_m}{\xi_m^2} \cdot e^{\frac{-2v_m}{\xi_m}}$$

and the same sum-kernel (3.31)

$$\kappa_\Sigma(\mathbf{u}, \mathbf{v}) = \sum_{m=1}^{d} u_i + v_i$$

in order to compare it to the analytic solution (3.32).
We use the same scalar constants but extend them to more dimensions via

$$N_0 = 1 \quad \text{and } \xi_m = \begin{cases} 0.1 & \text{, if } m \neq 2 \\ 0.15 & \text{, if } m = 2 \end{cases}$$

to capture a problem not symmetric in all properties.
We will use a symmetric grid with $n \in \{64, 128, 256, 512, 1024, 2048\}$ and $v^{\max} = 3$. We will vary the internal accuracy of a TT-tensor using $\epsilon \in \{10^{-2}, 10^{-5}, 10^{-7}\}$ to study its influence on the overall accuracy and the computational time. We use the standard algorithm for truncations of tensors as we study the influence of truncation on the accuracy and computational time. We will use a constant time step of $\Delta t = 0.1$ to simulate up to $T = 1$ in 10 steps to obtain a piecewise constant approximation $\widetilde{f}(\mathbf{v}, 1)$. We again define the L_2-error (3.33) and approximate the integral via quasi-Monte-Carlo integration (see [50]) as an iteration over all cells becomes infeasible for $d < 2$. The points are based on the Halton sequence ([17]) with $M = 1000dn$ points $\mathbf{u}_j \in [0, 3]^d$ and give the approximation

$$E \approx \left(\frac{1}{M} \cdot \sum_{j=1}^{M} (\widetilde{f}(\mathbf{u}_j, 1) - f(\mathbf{u}, 1)^2 \right)^{1/2} \tag{4.43}$$

to plot against the computational time.
We plot this for the two-dimensional problem in Figure 4.6 where we see an overall decline in the error with increasing the discretization n. The error is comparable in magnitude to the one reported in subsection 3.5.1 for a smaller computational domain. Most notably is the stagnation of improvement for $\epsilon = 10^{-2}$ when increasing n beyond 512. We explain this by the error propagation of internal truncations that starts to dominate the overall error as the plots for $\epsilon \in \{10^{-5}, 10^{-7}\}$ do not show this stagnation. We expect these plots to experience the same stagnation at some point with a finer discretization.

We also see a clear correlation between the inner accuracy ϵ and the computational time as a smaller ϵ leads to a higher rank $R_{\widetilde{f}}$ which in turn makes all computations more expensive. We still report a very short computational time with the longest simulation taking about 3.2 seconds. A similar simulation without efficient tensor storage took over 4 minutes (see subsection 3.5.1), a factor of approximately 82.
We see a similar behavior for $d = 3$ that we show in Figure 4.7. We overall report longer computations compared to $d = 2$ as was expected. We observe a similar stagnation for $\epsilon = 10^{-2}$ and see that the computational times for different values of ϵ are further apart, making the choice of a sufficient ϵ very important.

We show the results for $d = 4$ in Figure 4.8, where the stagnation for $\epsilon = 10^{-2}$ is obvious right from the start. There is no improvement of accuracy beyond $n = 128$ (the second point in the plot) here as the propagation of

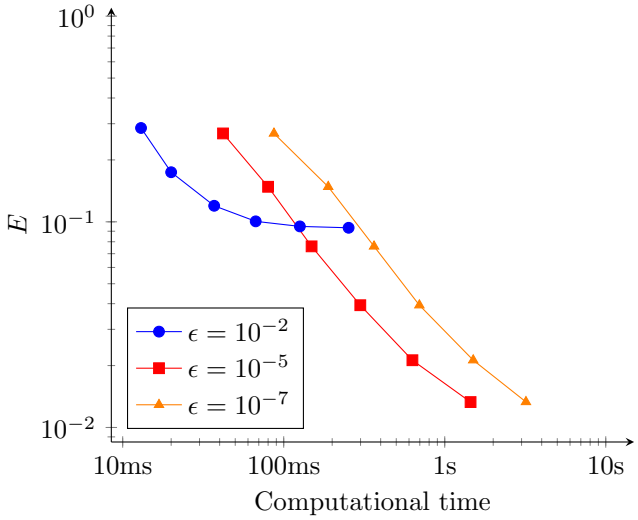


Figure 4.6: L_2-error E (4.43) with respect to computational time for $d = 2$ for $\epsilon \in \{10^{-2}, 10^{-5}, 10^{-7}\}$.

truncation errors dominates the error.
We note that these are the largest simulations of this thesis with up to $N = 2048^4 \approx 1.76 \cdot 10^{13}$ (over 17 trillion) cells. The straight forward storage of the arising tensors requires about 140 terabyte of memory but can be computed on off-the-shelf hardware in a reasonable time by using the TT-format.
We also see an inflection at $n = 256$ (the third point in the plots) where the plots deviate from an approximately straight line. A possible cause for this are hardware limitations as a considerable amount of time is spent on administrative tasks due to paging and page faults, see [68].
The simulation with $n = 2048$ and $\epsilon = 10^{-7}$ has not been possible (and is absent in Figure 4.8) due to limitations in memory. Smaller simulations for this accuracy feature $R_{\widetilde{f}} = 19$ at $t = 1$. This tensor would require modest 24 megabyte (if all $r^{\widetilde{f}}_m = R_{\widetilde{f}}$) but grows to a maximal $R_\omega \leq 784$ leading to a theoretical maximum of over 40 gigabyte of storage without any truncations.

4.5.3 Tracking moments of a population density distribution

This final numerical test is similar to the test from 3.5.2, where we compared the moments of a distribution from a numerical simulation to the analytic expression for those moments.
We again use the constant kernel $\kappa(\mathbf{u}, \mathbf{v}) = 1$ in order to use the ODE defined by (3.34). We use the multivariate Gaussian bell-curve defined by

$$f(\mathbf{v}, 0) = N_0 \cdot \exp\left(\sum_{m=1}^{d} c_m (v_m - s_m)^2\right) \tag{4.44}$$

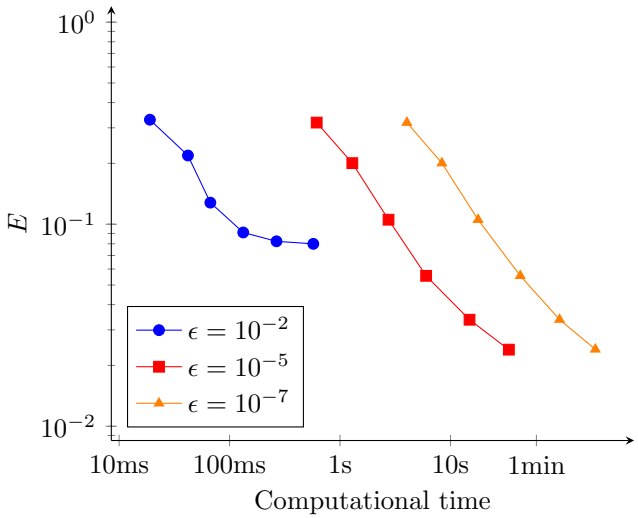


Figure 4.7: L_2-error E (4.43) with respect to computational time for $d = 3$ for $\epsilon \in \{10^{-2}, 10^{-5}, 10^{-7}\}$.

with $d = 3$. We again use $c_1 = c_2 = c_3 = 200$ and $s_1 = s_2 = s_3 = 0.1$. The scalar constant N_0 is used to set $\mu^{\mathbf{0}}(f)(0) = 0.1$. We will run simulations up to $T = 50$ which gives a degree of aggregation (3.36) of $I_{\text{agg}}(T) \approx 0.7143$. We choose $\mathbf{v}^{\max} = \big(10, 10, 10\big)$ as it gives a sufficiently large domain. We use $n_1 = n_2 = n_3 = 512$ cells in each direction, resulting in $N = 512^3 \approx 134 \cdot 10^6$ cells.
We will carry out two numerical simulations, one with $\Delta t = 0.1$ to give $\widetilde{\mu}^{\mathbf{e}}_C(t)$ and one with $\Delta t = 0.05$ to result in $\widetilde{\mu}^{\mathbf{e}}_F(t)$. The indices C and F stand for coarse and fine time steps, respectively. We use a high accuracy solution of (3.34) with variable stepsize and compare this with our results here. This shows the influence of Δt on the moments $\widetilde{\mu}^{\mathbf{e}}(t)$. We use the TT-format with a high accuracy of $\epsilon = 10^{-10}$ for this simulations.
We again compute the relative error

$$A^{\mathbf{e}}(t) := \frac{\widetilde{\mu}^{\mathbf{e}}(t)}{\mu^{\mathbf{e}}(t)} - 1 \tag{4.45}$$

for both fine and coarse time steps for all moments

$$\mathbf{e} \in \{0, \dots, 3\}^3.$$

We show a selection of these $A^{\mathbf{e}}(t)$ in Figure 4.9. We can see a clear benefit of smaller time steps as one expects. The decrease in $A^{\mathbf{e}}_F$ over $A^{\mathbf{e}}_C$ is very consistent. At the same time one can see the self-correcting behavior for $\mathbf{e} = (1, 1, 1)$, $\mathbf{e} = (2, 2, 2)$ and in part in $\mathbf{e} = (3, 3, 3)$, where the error decreases first but seems to rise after $t = 15$ for $\Delta t = 0.05$.
We observe a periodic behavior (especially prevalent) in $A^{(3,3,3)}$ that can be caused by the finite accuracy in the initial values. The moment $\mu^{(3,3,3)}(t)$ is a

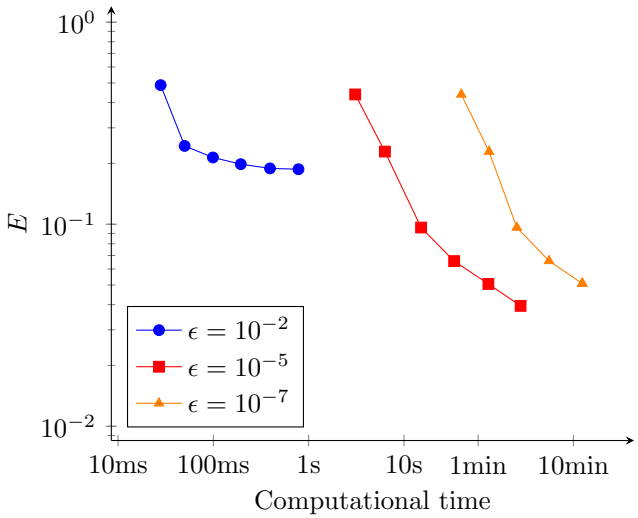


Figure 4.8: L_2-error E (4.43) with respect to computational time for $d = 4$ for $\epsilon \in \{10^{-2}, 10^{-5}, 10^{-7}\}$.

ninth-order moment with a close-to-zero initial value ($\mu^{(3,3,3)}(0) \approx 5.8 \cdot 10^{-10}$) and growth over several orders of magnitude to about $\mu^{(3,3,3)}(50) \approx 0.29$. We can not reliably study the relative error $A^{(3,3,3)}(t)$ here, as the absolute error in the initial value is experiencing the same level of growth.
The first moments $\mu^{\mathbf{e}}(f)$ for $\mathbf{e} \in \{(1,0,0),(0,1,0),(0,0,1)\}$ see no change in theory as the total property stays constant. In the left of Figure 4.10 we show $A^{(1,0,0)}(t)$ for the coarse and fine time-discretizations. We expect $A^{(1,0,0)}(t) = 0$ in theory as time steps do not have an influence on constant functions and we have a moment preserving projection. We see a non-zero error here as $A^{(1,0,0)}(t)$ grows from 10^{-9} to 10^{-6} over the time span shown here. We explain this error by repeated truncation of tensors. This truncation is not moment conserving and introduces a source of error here. This explains the larger error in $\mu_F^{(1,0,0)}(t)$ as smaller time steps require more calculations and more truncations. We again observe the periodic behavior in $\mu^{(1,0,0)}(t)$. We also show the relative error $A^{\mathbf{e}}(t)$ of another square-free moment $\mu^{(1,1,0)}(t)$ in the right of Figure 4.10, where we do not see a difference introduced by the time steps.

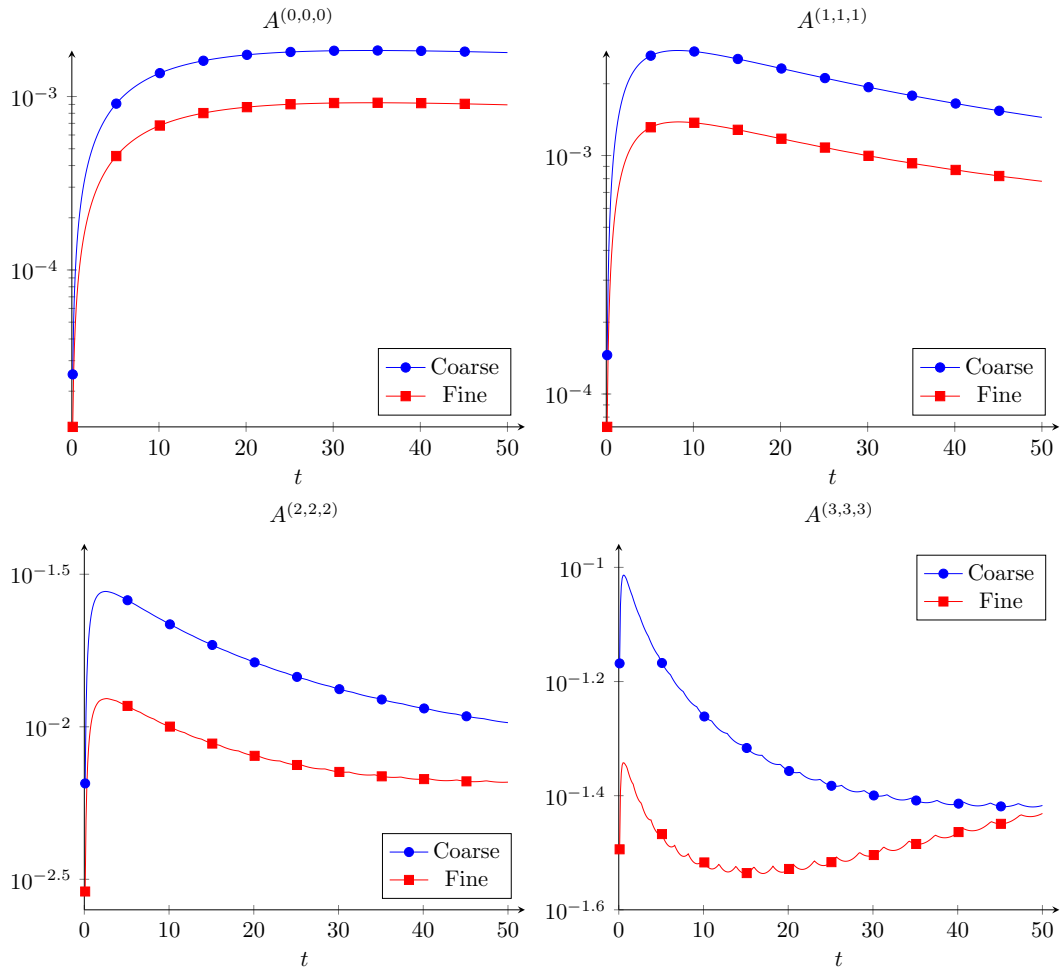


Figure 4.9: Relative error in the moments of a numerical simulation for a selection of moments, namely $A^{(0,0,0)}$ (top left), $A^{(1,1,1)}$ (top right), $A^{(2,2,2)}$ (bottom left), $A^{(3,3,3)}$ (bottom right).

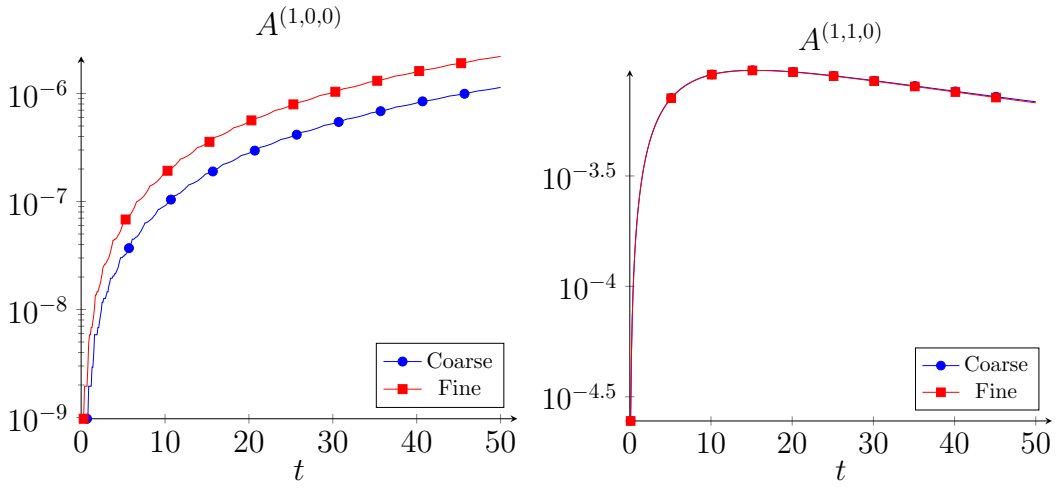


Figure 4.10: Relative error $A^{(1,0,0)}(t)$(left) and $A^{(1,1,0)}(t)$(right) for fine and coarse time discretizations. The plots for coarse and fine overlap on the right hand side.

Chapter 5

Estimation of aggregation kernels in univariate population balance equations

This chapter focusses on the problem of kernel estimation in the univariate setting. This is an inverse problem to reconstruct an approximate aggregation kernel $\kappa(u, v)$ from given particle distribution data $F(v, t)$ (using a capital F for given distributions). We here tackle the problem of data that is available at discrete time points t_0 to t_m.
We first introduce the problem structure and the already existing methods to tackle this problem in section 5.1. We will make note about their assumptions about the underlying data and resulting kernels.
We then re-introduce the discretization of the univariate property domain and make necessary assumptions about the kernel $\kappa(u, v)$ in Section 5.2 in order to estimate it. We will also establish the amount of data, that is available to us and its notation.
Section 5.3 defines the optimization procedure we use to reconstruct a kernel by establishing the target function and elaborate on the structure of the problem before showing numerical results in section 5.4.
All algorithms used in this chapter were implemented using MATLAB 2016b and subroutines provided in it.

5.1 Problem definition and existing methodology

The problem of kernel estimation that we are going to focus on in this chapter reads as follows:

Given a function $F(v,t) : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, find a symmetric kernel function $\kappa(u,v) : \mathbb{R}^2_{\geq 0} \to \mathbb{R}_{\geq 0}$ that fulfills

$$\frac{\partial F(v,t)}{\partial t} = \frac{1}{2} \int_0^v \kappa(u, v-u) F(u,t) F(v-u,t)\ du - F(v,t) \int_0^\infty \kappa(u,v) F(u,t)\ du. \tag{5.1}$$

This problem cannot be solved for every function $F(v,t)$ as a solution to (5.1) satisfies some necessary conditions like the monotone behavior of moments $\mu^e(f)(t)$ (derived straight from (1.5)). Existing methodology can be categorized into (at least) two separate approaches:

- Compute an approximation to the derivative on the left-hand side of (5.1) and compute the right-hand side of (5.1) using a reconstructed kernel where the reconstruction should minimize the error between left and right-hand side of (5.1) in some appropriate norm.
- Simulate the aggregation process with a reconstructed kernel (resulting in a particle distribution $f(v,t)$) where we try to minimize the error between $f(v,t)$ and $F(v,t)$ in an appropriate norm.

The first approach is pursued in [18] where Laurent polynomials are used as basis functions. The second approach is mentioned there but discarded for its high computational cost. A similar idea is used in [13]. A set of bilinear basis functions with compact support was used to set up and solve a system of linear equations at every time step. Both these approaches use a fixed set of symmetric basis functions $b_i(u,v) = b_i(v,u)$ and find weights α_i to approximate a kernel $\kappa(u,v) = \sum_i \alpha_i b(u,v)$ in order to approximate the time derivative. This approach is typically computationally less challenging but requires a high temporal resolution of the density distribution $F(v,t)$ to provide useful results.
The second approach is oftentimes used to fit a single parameter (linear factors in [9] and [61]) or a small number of parameters ([57] finds exponents of a rational function) when a certain form of the kernel is assumed. The cited works for this category also estimate constants regarding nucleation, breakage and growth at the same time, which we will not attempt. They also show their applicability with measured data from experiments instead of relying on simulations or analytical solutions. Due to the long time frame typically encountered in-between experimental measurements the estimation of a derivative is inaccurate and makes the first approach infeasible in these cases. The high computational cost associated with the simulation over the complete time frame is unavoidable and necessary here.
There exist some methods for parameter estimation in differential equations that are based on discontinuous data in the presence of measurement noise ([10], [58] or [59]) without a connection to population balances. However, the differential equations considered in these works have only a few degrees of

freedom and more measurements available making them non-applicable in this setting.

5.2 Discretization of the property space

This section introduces the suitable discretization of the one-dimensional domain of particle properties and further assumptions we are making about the kernel we want to estimate. We already established a discretization of the multivariate property space in 3.1 and will use the special case of $d = 1$. We will slightly modify the notation as some expression can simplified by choosing an alternative nomenclature. In order to make this chapter more independent of chapter 3 we will redefine a uniform grid here with a new modified nomenclature.
We choose a maximum particle property $v^{\max} \in \mathbb{R}_+$ and assume that no larger particle exists, i.e. $f(v,t) = 0$, if $v \geq v^{\max}$. For some $p \in \mathbb{N}$ we divide the interval $[0, v^{\max}]$ into $n := 2^p$ subintervals of equal length

$$h := \frac{v^{\max}}{n} \tag{5.2}$$

to obtain a grid we denote by $\mathcal{G}_p^{v^{\max}}$.
We restrict our analysis to functions $f(v,t)$ that are piecewise constant with respect to $\mathcal{G}_p^{v^{\max}}$ and assume this at every point in time. The mid-point of every cell $\mathcal{C}_i$ is denoted by v_i.
This allows us to represent a piecewise function as a nonnegative, time-dependent vector with entries

$$f(t) := \big(f_0(t), \ldots, f_{n-1}(t)\big) \in \mathbb{R}^n_{\geq 0}. \tag{5.3}$$

We also represent the measured data $F(v,t)$ with respect to the grid $\mathcal{G}_p^{v^{\max}}$ as a time-dependent vector

$$F(t_i) = \big(F_0(t_i), \quad \ldots, \quad F_{n-1}(t_i)\big) \in \mathbb{R}^n_{\geq 0}.$$

In view of this discretization, the goal is to reconstruct a kernel matrix $K \in \mathbb{R}^{n\times n}_{\geq 0}$ with values $K_{i,j} = \kappa(u,v)|_{\mathcal{C}_i \times \mathcal{C}_j}$ to represent a function $\kappa(\cdot,\cdot)$, piecewise constant with respect to $\mathcal{G}_p^{v^{\max}} \times \mathcal{G}_p^{v^{\max}}$.

Assumption 1. *To reduce the number of unknown coefficients in the kernel matrix $K \in \mathbb{R}^{n\times n}$, we assume it is symmetric and of rank $k(\ll n)$, i.e., it can be represented (or approximated) in the form*

$$K = U \cdot S \cdot U^T \tag{5.4}$$

with matrices $U \in \mathbb{R}^{n\times k}$ and $S \in \mathbb{R}^{k\times k}$. We require that $S = S^T$ to enforce the symmetry of K.

This assumption reduces the degrees of freedom in the kernel matrix K from $\frac{n(n+1)}{2}$ to $nk + \frac{k(k+1)}{2}$. It also allows us to use the algorithms introduced in section 3.3 to accelerate the calculations. For this we define $\alpha := US$ and $\beta := U$ to connect the notations of the kernel here and from section 3.2. We further define functions $Q^{\text{source}}(j, t; U, S)$ and $Q^{\text{sink}}(j, t; U, S)$ that compute the change of $f_j(t)$ with an estimated kernel given by U and S. The function $Q^{\text{source}}(j, t; U, S)$ includes the projection outlined in subsection 3.3.3. We will refer to the (thin/rectangular) matrix U as the *kernel basis*.
Our framework for the kernel estimation is related to [18] and [13] where coefficients for kernel functions within a linear space spanned by a number of given basis functions, e.g. Laurent polynomials, are to be found. In our framework, this corresponds to a given (fixed) matrix U with its entries given by the basis functions evaluated at the cell, i.e. $U_{i,j} = b_j(v)|_{\mathcal{C}_i}$. Here, we generalize this framework by including U in the optimization process.

5.3 Optimization problem

With these prerequisites, we define the following minimization problem

$$\begin{aligned} \underset{U,S}{\text{minimize}} \quad & E_2(U,S) := \left(h \sum_{i=0}^{m} \sum_{j=0}^{n-1} (F_j(t_i) - f_j(t_i))^2 \right)^{1/2} \\ \text{where} \quad & U \in \mathbb{R}^{n \times k},\ S = S^T \in \mathbb{R}^{k \times k}. \end{aligned} \tag{5.5}$$

Here, $F_j(t_i)$ is the given (measured) data and $f_j(t_i)$ are computed by numerical simulation of

$$\frac{df_j(t)}{dt} = Q^{\text{source}}(j, t; U, S) - Q^{\text{sink}}(j, t; U, S)$$

on the grid $\mathcal{G}_p^{v^{\max}}$. The factor h from (5.2) originates from the derivation of E_2 as integration over the functions represented by f and F to compare for different n.
A possible disadvantage of the above minimization problem results from the fact that absolute errors are considered. Cells with a small number of particles may have only a small influence on the kernel estimation since the error E_2 is dominated by index pairs (i, j) where $F_j(t_i)$ is large. To increase the sensitivity with respect to those cells with a small amount of particles, we define an error based on the χ^2-measure leading to

$$\begin{aligned} \underset{U,S}{\text{minimize}} \quad & E(U,S) := \left(h \sum_{i=0}^{m} \sum_{j=0}^{n-1} \frac{(F_j(t_i) - f_j(t_i))^2}{f_j(t_i) + \epsilon} \right)^{1/2} \\ \text{where} \quad & U \in \mathbb{R}^{n \times k},\ S = S^T \in \mathbb{R}^{k \times k}. \end{aligned} \tag{5.6}$$

Here, we weigh the difference of simulated and observed particles higher when the simulation indicates a small number of particles in a cell. Similar measures

to curve fitting are also used in machine learning [56]. We add $\epsilon = 10^{-7}$ to the denominator in (5.6) to ensure it is large enough to avoid numerical instabilities caused by the division. We still keep the constant factor h because it leads to better numerical results as it offsets the number of summands n.
We set $f_j(t_0) = F_j(t_0),\ j = 0, \ldots, n-1$, as the initial distribution which is always nonnegative. The constraint $f_j(t_i) \geq 0$ will be satisfied throughout the simulation when step sizes in the time discretization of the differential equation are chosen sufficiently small. The sink term $Q^{\text{sink}}(j, t; U, S)$, responsible for depletetion, will reduce $f_j(t)$ in each time step by a fraction of the current number of particles in that cell.
Since the objective is to determine a kernel that minimizes the error between measured and simulated density distribution, every evaluation of $E(U, S)$ requires a complete simulation (this methodology is of the second type) making it mandatory that efficient computational techniques are available. In fact, there are kn degrees of freedom in U and $\frac{k(k+1)}{2}$ degrees of freedom in S, leading to this number of evaluations of $E(U, S)$ for a single step in the evaluation process.
Most computational time during this optimization is spent in the evaluation of $Q^{\text{source}}(j, t; U, S)$ which only becomes feasible with Assumption 1 of a separable kernel.
We solve the minimization problem (5.6) using MATLAB's optimization routine *lsqnonlin* as well as *ode45* to solve the underlying differential equation. The routine *lsqnonlin* uses the Levenberg-Marquardt algorithm to search for local optima via finite differences.

If U has full rank k, its columns can be chosen orthonormal which turned out favorable in our numerical tests. We obtain orthonormal columns in U by replacing the current estimate U, S by Q, RSR^T where Q and R are the QR-factors of U.

Remark 5. *A kernel function $\kappa(u, v)$ is nonnegative, i.e. $\kappa(u, v) \geq 0$. This implies that the matrix K is elementwise nonnegative as well, which cannot be guaranteed without imposing complicated constraints on U and S. It is possible to restrict U and S to nonnegative matrices as well (which guarantees K to be non-negative). This, however, significantly reduces the search space for a fixed rank k and does not allow for orthonormal columns of U. Details about nonnegative matrix factorization (NNMF) are available in [11] and [31] but will not be used in this work.*

5.4 Numerical results

This section is devoted to numerical results using the proposed method to reconstruct a kernel from given (measured or simulated) data. In this work, to

be able to validate our results, we will reconstruct the following four different kernels from *measurements* $F(t_i)$ obtained through numerical simulation,

$$\text{Brownian}: \quad \kappa_B(u,v) = \left(u^{1/3} + v^{1/3}\right) \cdot \left(u^{-1/3} + v^{-1/3}\right) \tag{5.7}$$

$$\text{Shear}: \quad \kappa_S(u,v) = 2 \cdot \left(u^{1/3} + v^{1/3}\right)^{7/3} \tag{5.8}$$

$$\text{Sum}: \quad \kappa_\Sigma(u,v) = 5 \cdot (u+v) \tag{5.9}$$

$$\text{Peglow}: \quad \kappa_P(u,v) = 3 \cdot \left(u+v\right)^{0.7105} \cdot \left(uv\right)^{-0.062} \tag{5.10}$$

which are plotted in Figure 5.1. We note that the kernels κ_B and κ_Σ are separable, i.e., the kernel matrices can be represented in factored form (5.4) with $k = 3$ and $k = 2$, respectively. The discretized Brownian and sum kernel κ_B, κ_Σ can be written in the form

$$K_B: \quad U_{i,1} = \sqrt[3]{v_i},\ U_{i,2} = \sqrt{2},\ U_{i,3} = \ \sqrt[-3]{v_i}, \qquad S = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \tag{5.11}$$

$$K_\Sigma: \quad U_{i,1} = v_i,\ U_{i,2} = 5, \qquad S = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{5.12}$$

respectively, giving rise to initialize our optimizations with an antidiagonal matrix S. The other two kernels are not separable but can be approximated using a low rank k so that the resulting error in the simulation is dominated by discretization error, i.e., the number of cells n.

We use a bimodal initial distribution $f_{\text{init}}(v) = c \cdot \left(e^{-20v} + e^{-300\cdot(x-0.2)^2}\right)$ where the scaling coefficient c normalizes the function to

$$\mu^1(f)(t) = 10^{-2}.$$

We will use $v_{\max} = 1$ in all our numerical tests and hence leave out the superscript in $\mathcal{G}_p := \mathcal{G}_p^1$.
We obtain reference solutions with respect to a very fine grid $\mathcal{G}_{17}$ and take *measurements* for $m + 1 = 6$ equidistant time instances $t_i = i$ for $i \in \{0, \ldots, 5\}$. We obtain $\widetilde{F}(t_i) \in \mathbb{R}^{2^{17}}$ and consider it to be a distribution (perfectly) measured at time t_i. We coarsen it to the grid $\mathcal{G}_p$ by averaging over 2^{17-p} entries to obtain $F(t_i) \in \mathbb{R}^{2^p}$. This averaging process will preserve the total number of particles in the simulation. Throughout this section, we will use the following notation for discrete density distributions:

Grid	Reconstructed kernel	Exact kernel
$\mathcal{G}_p$	f	F
$\mathcal{G}_{17}$	$\widetilde{f}$	$\widetilde{F}$

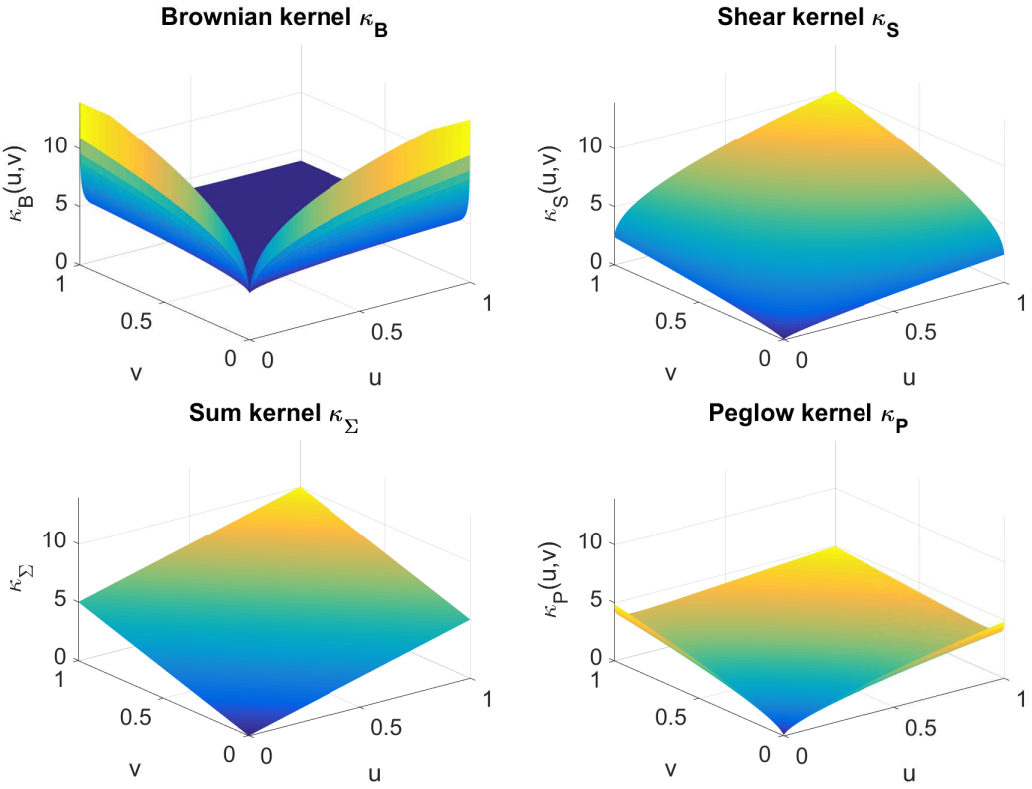


Figure 5.1: Four different aggregation kernels $\kappa(u, v)$.

The distributions $F(v, t_i)$ for κ_S and κ_P are shown in Figure 5.2. We see that the distributions have similar shapes and both have hardly any particles of mass greater than 0.5. The plots of $F \cdot F^T$ in the bottom row of Figure 5.2 show the amount of particles associated with an entry $K_{i,j}$ of the kernel matrix over all time points. We expect parts of the domain with many particles over the course of the simulation to have accurately estimated kernel values $K_{i,j}$.

We present numerical tests for two variants of the optimization problem, one with a fixed matrix factor U, i.e. optimization only with respect to S, and one with both U and S (5.4) included in the optimization.
To assess the quality of a reconstructed kernel for a practical application we use them for a simulation with a different distribution and compare the obtained results with the simulations with the correct kernel. The approximations are obtained by fitting data formed from a specific initial distribution f but should also hold a certain accuracy for other distributions to be of general use. For this additional step of validation we use the initial distribution $g(v, 0) = c \cdot ve^{-200(v-0.1)^2}$ (the factor c is again used for normalization of $g(v, 0)$ to $\mu^1(g)(t) = 10^{-2}$) and calculate $\widetilde{G}(v, 10)$ with the reference kernel and $\widetilde{g}(v, 10)$ with kernel factors U and S (approximated with respect to $\mathcal{G}_p$) with respect to the very fine grid $\mathcal{G}_{17}$. To obtain the estimated kernel on a finer grid

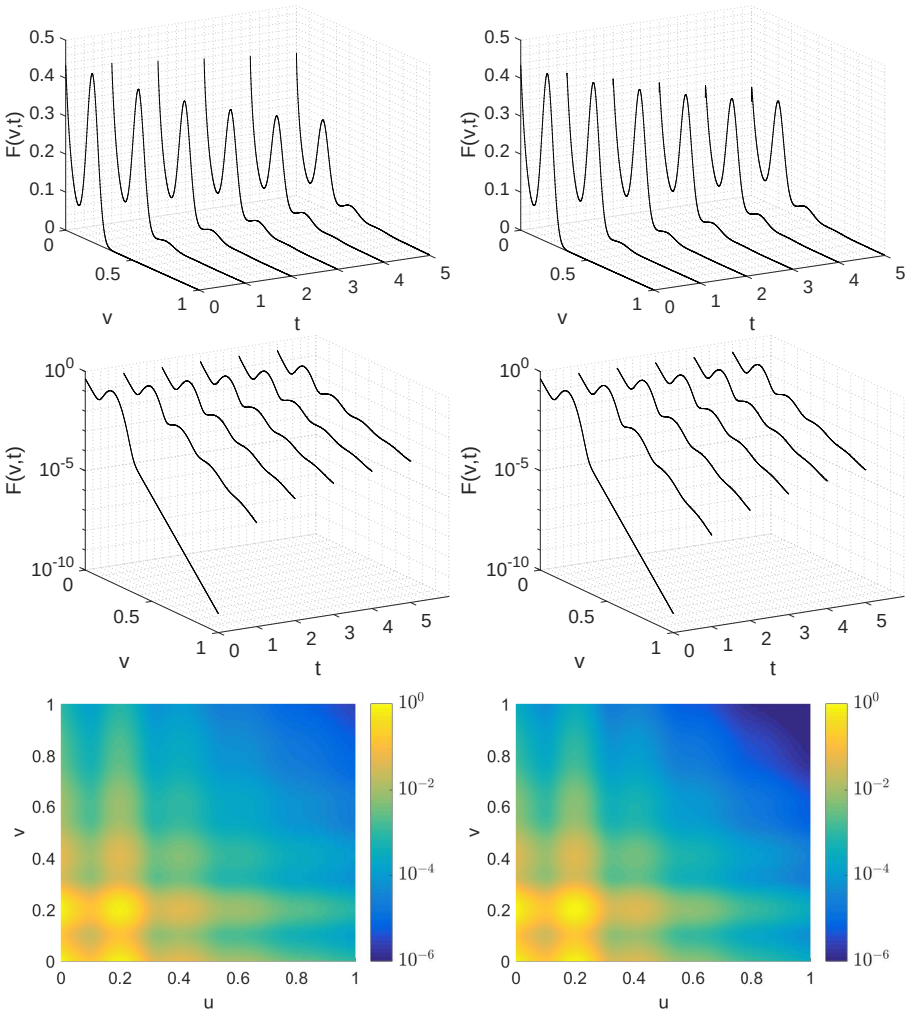


Figure 5.2: Evolution of the particle distribution $F(v,t)$ for time instances $t_i = 0,\ldots,5$, with κ_S (left) and κ_P (right) for linear (top) and logarithmic (middle) scaling of the y-axis. The bottom row shows $F \cdot F^T$ on logarithmic scale.

we simply assign the value of a coarse cell to all finer cells. We chose $g(v,0)$ in view of its maximum at $v = 0.1$ which is near the local minimum of $f(v,0)$.

5.4.1 Optimization including the kernel basis U

For a variable U we use the rank $k = 5$ and start the optimization process with

$$U_{j,i} = v_i^{1/j}, \quad S = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{5.13}$$
$$i = 1, \ldots, n \text{ and } j = 0, \ldots, n-1,$$

where v_i denotes the mid-point of the cell $\mathcal{C}_i$. We use this initial kernel because it is the pointwise evaluation of a smooth function and gives only moderate aggregation rates in the considered domain. High rates of aggregation will results in very small time steps in the solution of the differential equation to ensure the positivity of f. Our choice of S mimics the matrix present in the kernels κ_B (5.11) and κ_Σ (5.12). We also experience high sensitivity with respect to the initial conditions, as different initial conditions will lead to different local minima.
The kernel matrix that solves the optimization problem (5.6) is denoted by

$$K_{\text{est}} = U_{\text{est}} S_{\text{est}} U_{\text{est}}^T$$

for each of the four kernels. We plot the estimated kernels for $p = 10$ (results in $n = 1024$) together with the true kernels in the left of Figures 5.3 and 5.4. We also compute the pointwise relative errors

$$E_{\text{rel}}(v_i, v_j) = \frac{|K_{\text{est}}(v_i, v_j) - K_{\text{true}}(v_i, v_j)|}{K_{\text{true}}(v_i, v_j)}, \ 0 \leq i, j, \leq n-1 \tag{5.14}$$

and plot these on the right hand side of Figures 5.3 and 5.4.

For all those four kernels, we see a minimum of the relative error around $(0.2, 0.2)$ which we attribute to our choice of initial distribution with a peak at 0.2. We are not concerned about larger relative errors in the upper right triangle ($v_i + v_j > 1$) since the aggregation of two particles to one with mass greater than 1 and hence out of our computational domain should not occur, hence there is no (or hardly any) data to estimate the kernel in this region. The relative error of the shear kernel κ_S is small over the entire domain, a similar result is observed for the Brownian kernel κ_B. The sum and Peglow kernel approximation errors are smallest, where most data is available - this follows from comparison with the plots of $F \cdot F^T$ in Figure 5.2 (right).
We calculate $\widetilde{G}(v, 10)$ and $\widetilde{g}(v, 10)$ (always with respect to $\mathcal{G}_{17}$) with the reference and reconstructed kernel respectively and show them for $p = 10$ in Figure 5.5 on the interval $[0, 0.5]$ (the simulation was computed on $[0, 1]$). We see here

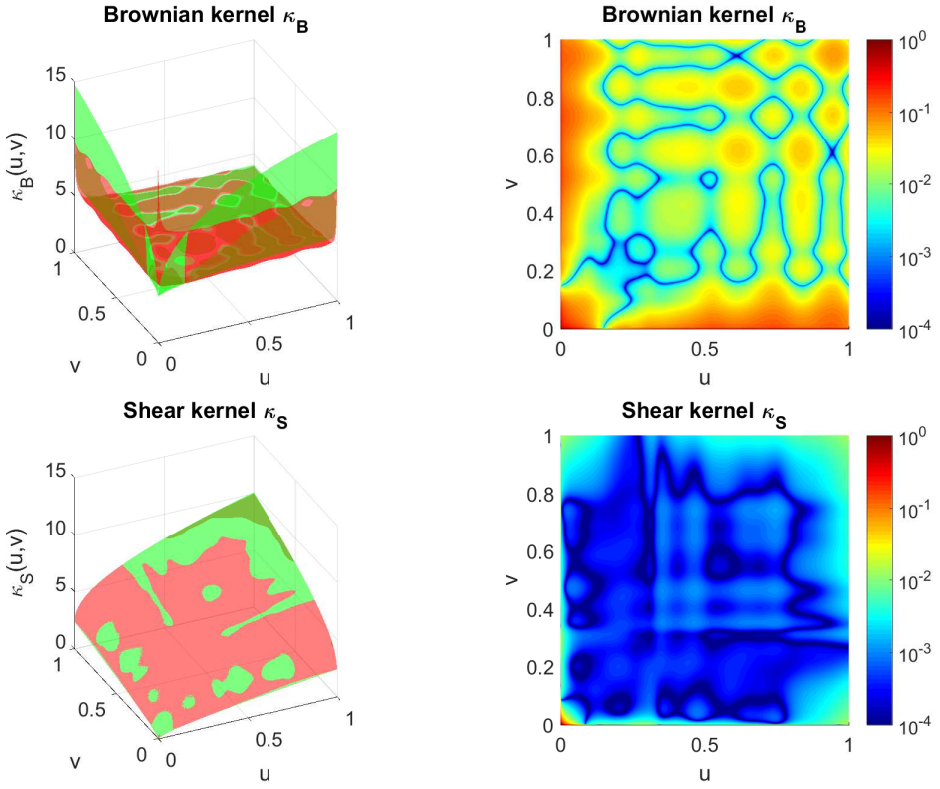


Figure 5.3: Left: Comparison between the true kernel (green) and the estimated kernel (red) with the kernel basis U included in the optimization. Right: (Logarithm of the) relative error (5.14) of the kernel. Top: Brownian kernel. Bottom: Shear kernel.

the larger error in the sum-kernel in the bottom left.

We also calculate the relative L_2 error

$$\text{err} := \left(\frac{\sum_{i=1}^{2^{17}} \left(\widetilde{G}(v_i, 10) - \widetilde{g}(v_i, 10) \right)^2}{\sum_{i=1}^{2^{17}} \widetilde{G}(v_i, 10)^2} \right)^{1/2} \tag{5.15}$$

between $\widetilde{G}(v, 10)$ and $\widetilde{g}(v, 10)$ for each kernel based on approximations for different grids $\mathcal{G}_5$ and $\mathcal{G}_{10}$ and present the results in Table 5.1. On the positive side, we have relative approximation accuracies of order $\mathcal{O}(10^{-2})$ on a relatively coarse grid with 2^5 pivots, i.e. a gridwidth of $h = 2^{-5} = 0.03125$. However, we observe hardly any improvement (and possibly even worse results) with respect to further refinement of the grid even though, as we will see in the following subsection, the framework would indeed allow for higher accuracies. These computations have the complexity of $\mathcal{O}(k^2 n^2 \log n)$ as we calculate $\mathcal{O}(kn)$ finite differences per optimization step, each of complexity $\mathcal{O}(kn \log n)$.

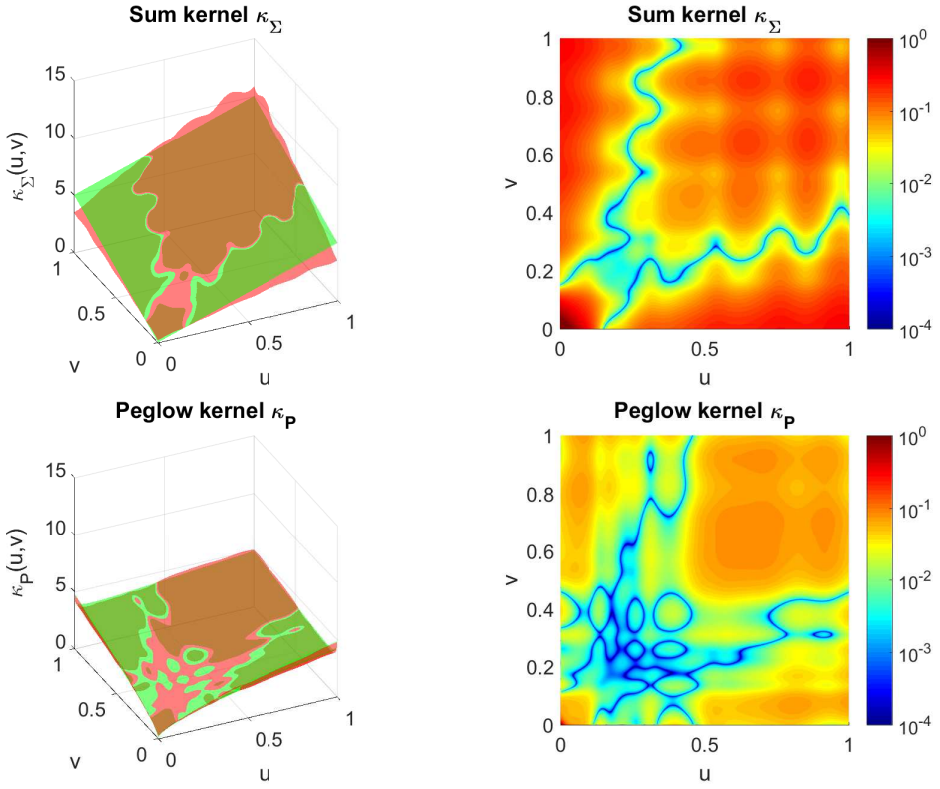


Figure 5.4: Left: Comparison between the true kernel (green) and the estimated kernel (red) with the kernel basis U included in the optimization. Right: (Logarithm of the) relative error (5.14) of the kernel. Top: Sum kernel. Bottom: Peglow kernel.

The number of optimization steps has little to no influence on the overall computational time. The optimization routine took about 90 seconds for $n = 128$ and about 50 minutes for $n = 1024$.

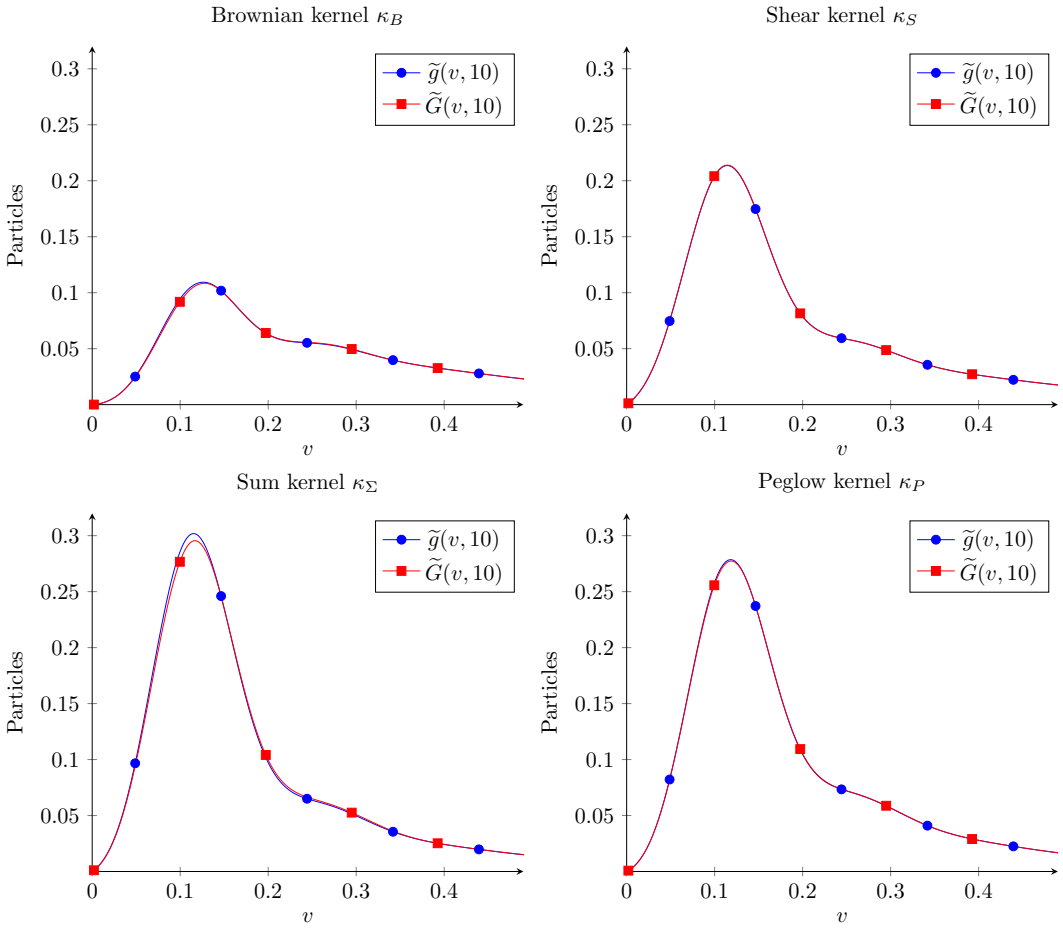


Figure 5.5: Particle distributions $\widetilde{g}(v, 10)$ (blue) and $\widetilde{G}(v, 10)$ (red) for kernels and their approximations with kernel basis U subject to optimization for $v \in [0, 0.5]$.

p \ kernel	κ_B	κ_S	κ_Σ	κ_P
5	$4.23 \cdot 10^{-2}$	$3.83 \cdot 10^{-2}$	$5.65 \cdot 10^{-2}$	$5.02 \cdot 10^{-2}$
6	$1.29 \cdot 10^{-2}$	$2.01 \cdot 10^{-2}$	$3.14 \cdot 10^{-2}$	$2.62 \cdot 10^{-2}$
7	$7.90 \cdot 10^{-3}$	$9.80 \cdot 10^{-3}$	$2.72 \cdot 10^{-2}$	$2.62 \cdot 10^{-3}$
8	$1.60 \cdot 10^{-3}$	$4.30 \cdot 10^{-3}$	$2.45 \cdot 10^{-2}$	$8.00 \cdot 10^{-3}$
9	$1.25 \cdot 10^{-2}$	$2.10 \cdot 10^{-3}$	$2.34 \cdot 10^{-2}$	$5.90 \cdot 10^{-3}$
10	$1.24 \cdot 10^{-2}$	$1.10 \cdot 10^{-3}$	$2.31 \cdot 10^{-2}$	$5.40 \cdot 10^{-3}$

Table 5.1: Relative L_2 error (5.15) between $\tilde{G}$ and $\tilde{g}$ with the kernel basis U included in the optimization for kernel approximations based on different grids $\mathcal{G}_p$.

5.4.2 Optimization with a fixed kernel basis U

We now fix the kernel basis U by choosing a set of basis functions and optimize only with respect to $S \in \mathbb{R}^{k\times k}$. We can allow for a larger rank k in this setting since the number of degrees of freedom is no longer linear in n but only quadratic in k, the maximum rank of the kernel matrix.
We choose $k = 7$ basis functions $b_j(v) = v^{j/3}$ with $j \in \{-3, \dots, 3\}$ and orthogonalize the resulting matrix via QR-decomposition. These basis functions are used because many theoretical kernels are based on length, surface and volume of interacting particles. The chosen exponents correspond to these properties and κ_B and κ_Σ are exactly representatable with the chosen basis. The initial kernel is given by the zero-matrix $S = 0$ as we experience better results with this configuration, as well in terms of resulting kernel as computational time.

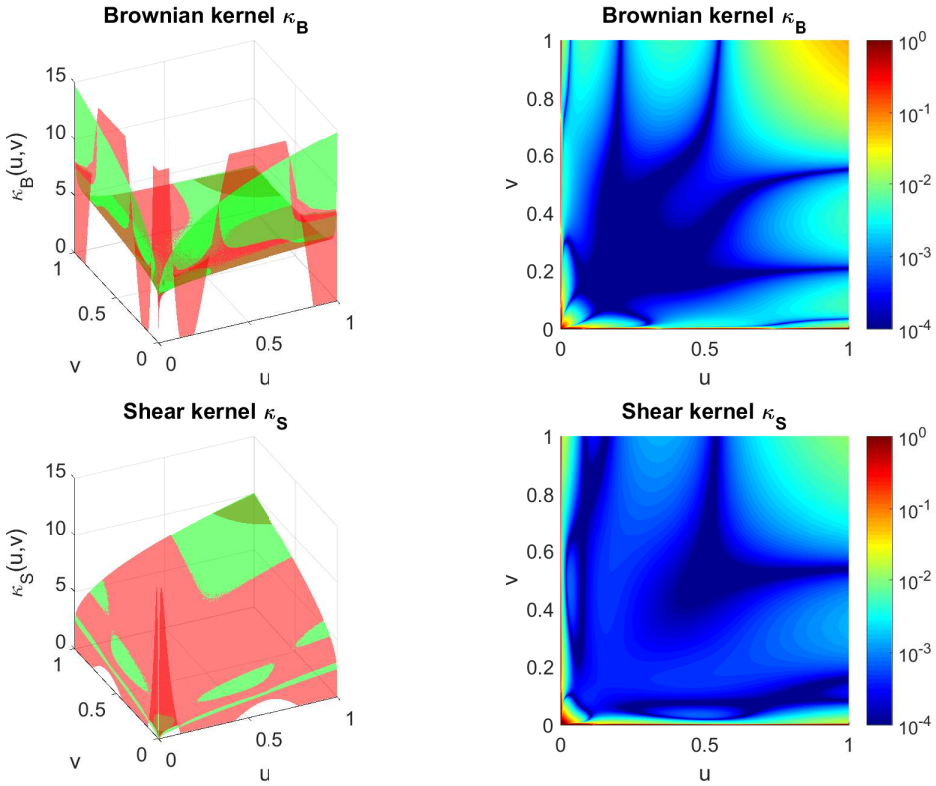


Figure 5.6: Left: Comparison between the true kernel (green) and the estimated kernel (red) for a fixed kernel basis U. Right: (Logarithm of the) relative error (5.14) of the kernel approximation. Top: Brownian kernel. Bottom: Shear kernel.

In Figures 5.6 and 5.7 we show the resulting kernel approximations based on $\mathcal{G}_{10}$ with the exact kernels (left) as well as the relative approximation errors (5.14) (right). We observe a clear improvement compared to the kernels estimated with variable U (shown in Figures 5.3 and 5.4) and offer two interpretations of this result:

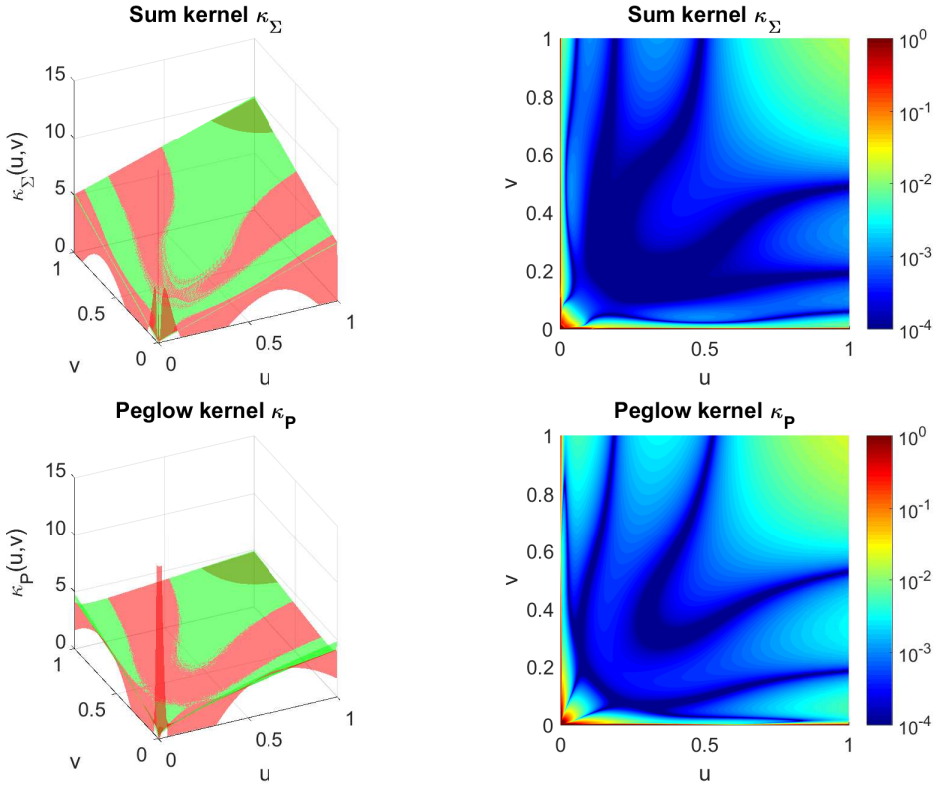


Figure 5.7: Left: Comparison between the true kernel (green) and the estimated kernel (red) for a fixed kernel basis U. Right: (Logarithm of the) relative error (5.14) of the kernel approximation. Top: Sum kernel. Bottom: Peglow kernel.

- The fixed basis U has been chosen to span a space that allows for accurate approximations of the exact kernels, hence the search space has been reduced significantly, allowing for an easier optimization
- Kernel estimation is an inverse problem that we propose to solve using a (non-convex) optimization. There could exist several local minima resulting in similar or identical sitributions f.

In the end, we are estimating a kernel that simulates results close to the measured results, not a kernel that is close to another kernel (in our experiments given, but in practice unknown).
Using these kernels together with the initial distribution $g(v,0) = c{\cdot}ve^{-200(v-0.1)^2}$, the resulting distributions $\widetilde{G}(v,10)$ and $\widetilde{g}(v,10)$ are shown in Figure 5.8 for $p = 10$ in the interval $v \in [0, 0.5]$.

The relative L_2 errors for approximations based on grids $\mathcal{G}_5$ to $\mathcal{G}_{10}$ are shown in Table 5.2.

Comparing these to the respective results for an included U in Table 5.1, we see less accurate kernel approximations on the coarser grids ($p = 5, 6, 7$) and more accurate results on a fine grid with $p = 10$. We estimate the order

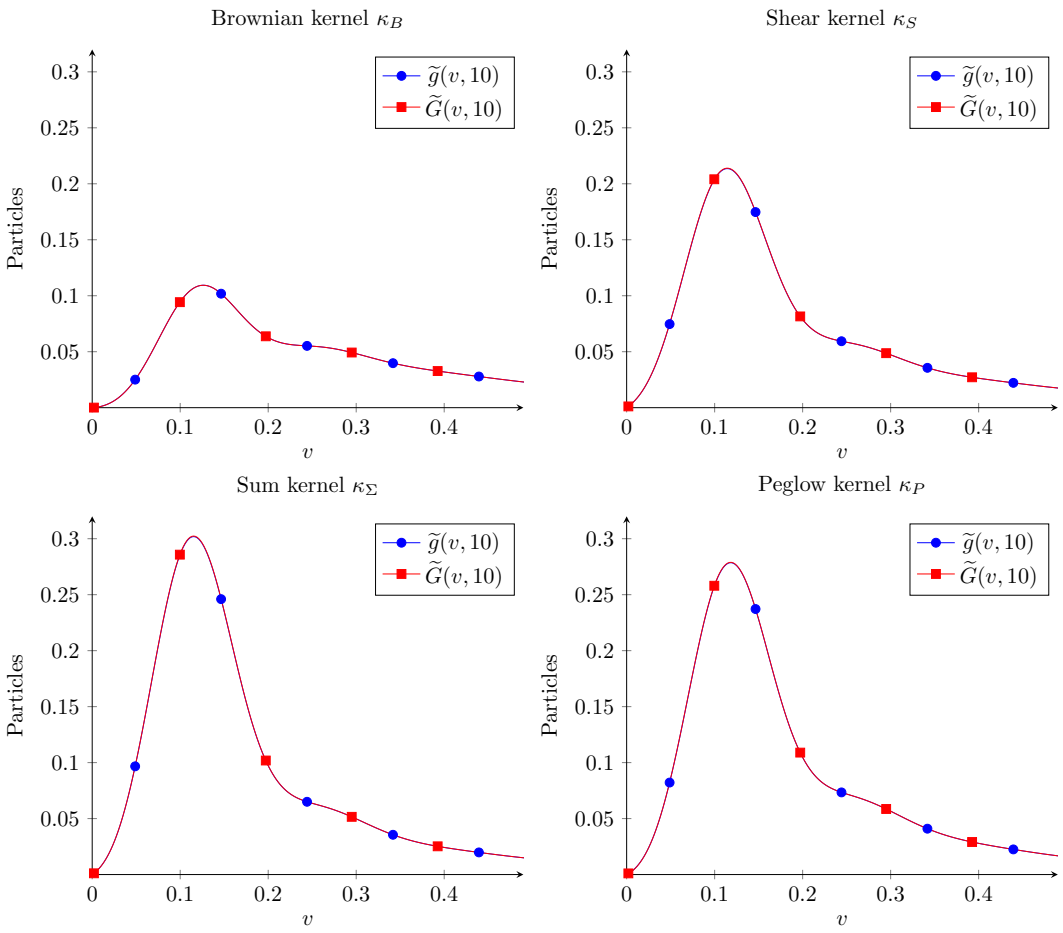


Figure 5.8: Particle distributions $\widetilde{g}(v, 10)$ (blue) and $\widetilde{G}(v, 10)$ (red) for kernels and their approximations with a fixed kernel basis U for $v \in [0, 0.5]$.

of convergence with $\mathcal{O}(2^{-2p})$, the error decreases very fast with an increase in the number of cells.
The computational complexity of this approach is estimated at $\mathcal{O}(k^3 n \log n)$ as there are $\mathcal{O}(k^2)$ computations of complexity $\mathcal{O}(kn \log n)$. This is supported by our numerical results as we take about 90 seconds for the most accurate simulation with $n = 1024$. Optimizations with a coarser grid ($p = 5$) takes only about 10 to 15 seconds. This setting sees a higher influence of the number of optimizations steps which is hard to estimate beforehand.

p \ kernel	κ_B	κ_S	κ_Σ	κ_P
5	$3.89 \cdot 10^{-1}$	$4.81 \cdot 10^{-1}$	$5.72 \cdot 10^{-1}$	$5.56 \cdot 10^{-1}$
6	$1.03 \cdot 10^{-1}$	$4.57 \cdot 10^{-1}$	$3.62 \cdot 10^{-1}$	$4.17 \cdot 10^{-1}$
7	$3.46 \cdot 10^{-2}$	$3.33 \cdot 10^{-2}$	$1.92 \cdot 10^{-2}$	$4.29 \cdot 10^{-2}$
8	$5.26 \cdot 10^{-3}$	$6.88 \cdot 10^{-3}$	$4.33 \cdot 10^{-3}$	$6.22 \cdot 10^{-2}$
9	$1.41 \cdot 10^{-3}$	$2.16 \cdot 10^{-3}$	$1.81 \cdot 10^{-3}$	$1.91 \cdot 10^{-2}$
10	$2.74 \cdot 10^{-4}$	$1.05 \cdot 10^{-3}$	$8.70 \cdot 10^{-4}$	$1.18 \cdot 10^{-3}$

Table 5.2: Relative L_2 error (5.15) between $\tilde{G}$ and $\tilde{g}$ for a fixed kernel basis U with approximations based on different grids $\mathcal{G}_p$.

Chapter 6

Conclusions and outlook

The subject of this work is efficient techniques to compute aggregation integrals that appear in multivariate population balances. The prerequisites for a fast and accurate evaluation of these aggregation integrals are a uniform tensor grid, the separation of kernel functions and the multivariate fast Fourier transformation. These are introduced in sections 3.1, 3.2 and 3.4 respectively. These prerequisites allow for a log-linear computational complexity (as opposed to quadratic) in the number of unknowns. We presented a simple framework for the conservation of moments in subsection 3.3.3 necessary to uphold physical invariants of the aggregation process.
All proposed algorithms and schemes were implemented using *C++* and we compared this implementation against the popular fixed pivot method and the cell-average technique in the setting of two or three internal properties with respecto to the accuracy of the particle distribution, the conservation of high-order moments and the computational complexity of the discretization schemes.
Our tests showed an advantage of our proposed method with respect to all tested characteristics as long as the computational domain is not too large compared to the smallest cell size, at which a geometrically scaled grid becomes preferable over a uniform grid. We also confirmed the theoretical estimates of log-linear complexity with numerical tests.

We explored a technique for efficient tensor storage in section 4.1 to lift the *curse of dimensionality*, the exponential complexity in both storage and computations with respect to the dimension of the underlying property space. This limited our straight forward approach from chapter 3 to a maximum of three dimensions. This storage format sacrifices the exactness of operations (lost in truncations) for a complexity that is linear in the number of involved particle properties which makes the computations feasible.
We examined the necessary arithmetic operations in section 4.2 and introduced improvements to the truncation operation in section 4.3. The truncation contributed a relevant portion to the overall computational time in our numerical

tests in subsection 4.5.1 and our proposed method decreased the required time for this operation. We additionally examined the influence of the inner accuracy of a tensor on the accuracy of a density distribution in high dimensions (up to 4 internal properties) and studied the influence of the moment preserving projection of high-order moments with three dimensions in subsections 4.5.2 and 4.5.3 respectively.

Our numerical tests show that we have "broken" the curse of dimensionality by using the TT-format. Simulations with several billion degrees of freedom can be run on an office desktop computer in a reasonable time. Simply storing these density distributions without efficient storage requires several terabyte of memory. Only with even more inner dimensions or finer discretizations (over 10 trillion cells) we reach the limit of off-the-shelf hardware and suggest using computational clusters or a supercomputer.

Based on the results of this work we mention some possible extensions and other problems that might be of interest for further research. Improvements for multivariate PBEs can be separated into three categories:

- Application of other phenomena to a uniform grid
- Improvements of the discretization
- Extension of the TT-format in a hierarchical fashion

The first point would allow a simulation of a process that includes additional particle phenomena like growth or nucleation. We are currently restricted to a process of pure aggregation. A derivation of the numerics of these processes based on our discretization allows for a simulation of more complex problems. The second point is based on our observation of the typical behavior of multivariate PBEs. The distributions have the tendency to concentrate among certain parts of the domain. A hierarchical grid with local refinements (as presented in [43] and [44] for the univariate case) can grant additional accuracy to important parts of the domain without sacrificing the log-linear complexity. An additional gain may be archived by using higher-order functions on single cells to better aproximate the smooth distributions in a typical PBE. This concept was introduced to univariate PBEs in [44] as well. The higher-order ansatz can reduce the number of cells without reducing the accuracy by introducing artificial coordinates (to store coefficients of higher-order functions) to a tensor. This additional dimensionality can make this approach infeasible for full tensor storage but can be beneficial when combined with the TT-storage. The third direction focuses on the TT-storage format itself. The global low-rank format relies on the existence of these low ranks. Due to successive pointwise multiplications and convolutions, this assumption might not be fulfilled as these operations increase the internal ranks of a TT-tensor. A hierarchical format (extending the $\mathcal{H}$-matrix format from [28] and not to be confused with the

hierarchical tensor format from [29]) based on local low-rank approximations on some parts of the domain and full tensor storage in other parts can reduce the maximum rank of a tensor. We expect this to come in exchange for a much more complicated tree-based structure and recursive algorithms. We assume this third point to be the most complex and cannot predict a possible outcome.

In chapter 5 we present the related problem of kernel estimation in the case of a one-dimensional property space. We present a novel framework that does not rely on a high temporal resolution or interpolation at intermediate time points. It allows for a highly accurate reconstruction of discrete aggregation kernels under the assumption that the kernel-matrix allows for a representation with a low-rank. This approach can either be based on theoretical knowledge of basis functions or can form these basis functions during the process. We tested these two approaches in section 5.4 to re-estimate known kernels from highly accurate measurements and verified the results by a simulation with a different density distribution.
There are some possible extensions to our process of kernel estimation presented in chapter 5. These can be separated into three categories:

- Usage of sampled particles
- Estimation of constants for other phenomena
- Application to multivariate problems

For our numerical tests in section 5.4 we used highly accurate distribution data without any noise. Sampled data (as we expect from an experimental measurement) will always have some noise present that needs to be addressed in future work.
Our optimization procedure was able to extract rates from an aggregation-only process without other phenomena present. Future projects may estimate breakage-kernels, nucleation-rates or growth-functions simultaneously using the theory introduced here. The inclusion of further phenomena can also make an accurate prediction harder as it complicates the structure.
We are currently restricted to the estimation of kernels in the setting of univariate PBEs. Introducing additional particle properties increases the complexity of the right-hand side of the ODE and the problem itself. This makes the estimation of aggregation kernels with more particle properties a challenging question.

Bibliography

[1] Robin Ahrens and Sabine Le Borne. “Tensor trains and moment conservation for multivariate aggregation in population balance modeling”. In: *Applied Numerical Mathematics* 153 (July 2020), pp. 473–491.

[2] Robin Ahrens and Sabine Le Borne. “FFT-based evaluation of multivariate aggregation integrals in population balance equations on uniform tensor grids”. In: *Journal of Computational and Applied Mathematics* 338 (Aug. 2018), pp. 280–297.

[3] Robin Ahrens and Sabine Le Borne. “Reconstruction of low rank aggregation kernels in univariate population balance equations”. In: *Advances in Computational Mathematics* (**submitted in November 2019**).

[4] Robin Ahrens et al. “Numerical Methods for Coupled Population Balance Systems Applied to the Dynamical Simulation of Crystallization Processes”. In: *Dynamic Flowsheet Simulation of Solids Processes*. Springer International Publishing, 2020, pp. 475–518.

[5] Felix Anker et al. “A comparative study of a direct discretization and an operator-splitting solver for population balance systems”. In: *Computers & Chemical Engineering* 75 (Apr. 2015), pp. 95–104.

[6] Menwer M. Attarakih, Christian Drumm, and Hans-Jörg Bart. “Solution of the population balance equation using the sectional quadrature method of moments (SQMOM)”. In: *Chemical Engineering Science* 64.4 (Feb. 2009), pp. 742–752.

[7] M. Bebendorf and S. Rjasanow. “Adaptive Low-Rank Approximation of Collocation Matrices”. In: *Computing* 70.1 (Feb. 2003), pp. 1–24.

[8] Steffen Börm and Lars Grasedyck. “Hybrid cross approximation of integral operators”. In: *Numerische Mathematik* 101.2 (June 2005), pp. 221–249.

[9] Allan S. Bramley, Michael J. Hounslow, and Rosemary L. Ryall. “Aggregation during Precipitation from Solution: A Method for Extracting Rates from Experimental Data”. In: *Journal of Colloid and Interface Science* 183.1 (Oct. 1996), pp. 155–165.

[10] Nicolas J-B. Brunel. “Parameter estimation of ODEs via nonparametric estimators”. In: *Electronic Journal of Statistics* 2.0 (2008), pp. 1242–1267.

[11] M. Catral et al. “On reduced rank nonnegative matrix factorization for symmetric nonnegative matrices”. In: *Linear Algebra and its Applications* 393 (Dec. 2004), pp. 107–126.

[12] Jayanta Chakraborty and Sanjeev Kumar. “A new framework for solution of multidimensional population balance equations”. In: *Chemical Engineering Science* 62.15 (Aug. 2007), pp. 4112–4125.

[13] Jayanta Chakraborty et al. “Inverse Problems in Population Balances. Determination of Aggregation Kernel by Weighted Residuals”. In: *Industrial & Engineering Chemistry Research* 54.42 (July 2015), pp. 10530–10538.

[14] Anwesha Chaudhury, Ivan Oseledets, and Rohit Ramachandran. “A computationally efficient technique for the solution of multi-dimensional PBMs of granulation via tensor decomposition”. In: *Computers & Chemical Engineering* 61 (2014), pp. 234–244.

[15] Anwesha Chaudhury et al. “An extended cell-average technique for a multi-dimensional population balance of granulation describing aggregation and breakage”. In: *Advanced Powder Technology* 24.6 (Nov. 2013), pp. 962–971.

[16] James W. Cooley and John W. Tukey. “An algorithm for the machine calculation of complex Fourier series”. In: *Mathematics of Computation* 19.90 (May 1965), pp. 297–297.

[17] Josef Dick, Frances Y. Kuo, and Ian H. Sloan. “High-dimensional integration: The quasi-Monte Carlo way”. In: *Acta Numerica* 22 (Apr. 2013), pp. 133–288.

[18] H. Eisenschmidt et al. “Estimation of aggregation kernels based on Laurent polynomial approximation”. In: *Computers & Chemical Engineering* 103 (Aug. 2017), pp. 210–217.

[19] Akinola Falola, Antonia Borissova, and Xue Zhong Wang. “Extended method of moment for general population balance models including size dependent growth rate, aggregation and breakage kernels”. In: *Computers & Chemical Engineering* 56 (Sept. 2013), pp. 1–11.

[20] J. M Fernández-Dıéaz and G. J Gómez-Garcıéa. “Exact solution of Smoluchowski’s continuous multi-component equation with an additive kernel”. In: *Europhysics Letters (EPL)* 78.5 (May 2007), p. 56002.

[21] Francis Filbet and Philippe Laurençot. “Numerical Simulation of the Smoluchowski Coagulation Equation”. In: *SIAM Journal on Scientific Computing* 25.6 (Jan. 2004), pp. 2004–2028.

[22] L. Forestier-Coste and S. Mancini. “A Finite Volume Preserving Scheme on Nonuniform Meshes and for Multidimensional Coalescence”. In: *SIAM Journal on Scientific Computing* 34.6 (Jan. 2012), B840–B860.

[23] F. Gelbard and Seinfeld J. “Coagluation and growth of a multicomponent aerosol”. In: *Journal of Colloid and Interface Science* 63(3) (1978), pp. 472–479.

[24] David M. Ginter and Sudarshan K. Loyalka. “Apparent size-dependent growth in aggregating crystallizers”. In: *Chemical Engineering Science* 51.14 (July 1996), pp. 3685–3695.

[25] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[26] W. Hackbusch. “Convolution of hp-functions on locally refined grids”. In: *IMA Journal of Numerical Analysis* 29.4 (Oct. 2008), pp. 960–985.

[27] W. Hackbusch. “Fast and exact projected convolution for non-equidistant grids”. In: *Computing* 80.2 (May 2007), pp. 137–168.

[28] Wolfgang Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*. Springer Berlin Heidelberg, 2015.

[29] Wolfgang Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer Berlin Heidelberg, 2012.

[30] M. J. Hounslow, R. L. Ryall, and V. R. Marshall. “A discretized population balance for nucleation, growth, and aggregation”. In: *AIChE Journal* 34.11 (Nov. 1988), pp. 1821–1832.

[31] Kejun Huang, Nicholas D. Sidiropoulos, and Ananthram Swami. “Non-negative matrix factorization revisited: uniqueness and algorithm for symmetric decomposition”. In: *IEEE Trans. Signal Process.* 62.1 (2014), pp. 211–224.

[32] Roberto Irizarry. “Fast Monte Carlo methodology for multivariate particulate systems—I: Point ensemble Monte Carlo”. In: *Chemical Engineering Science* 63.1 (Jan. 2008), pp. 95–110.

[33] Simon M. Iveson et al. “Nucleation, growth and breakage phenomena in agitated wet granulation processes: a review”. In: *Powder Technology* 117.1-2 (June 2001), pp. 3–39.

[34] Gurmeet Kaur et al. “Analytical Approach For Solving Population Balances: A Homotopy Perturbation Method”. In: *Journal of Physics A Mathematical and Theoretical* (June 2019).

[35] Margaritis Kostoglou. “Extended cell average technique for the solution of coagulation equation”. In: *Journal of Colloid and Interface Science* 306.1 (Feb. 2007), pp. 72–81.

[36] F. Einar Kruis, Arkadi Maisels, and Heinz Fissan. “Direct simulation Monte Carlo method for particle coagulation and aggregation”. In: *AIChE Journal* 46.9 (Sept. 2000), pp. 1735–1742.

[37] F. Einar Kruis et al. “A Simple Model for the Evolution of the Characteristics of Aggregate Particles Undergoing Coagulation and Sintering”. In: *Aerosol Science and Technology* 19.4 (Jan. 1993), pp. 514–526.

[38] J. Kumar et al. “The cell average technique for solving multi-dimensional aggregation population balance equations”. In: *Computers & Chemical Engineering* 32.8 (Aug. 2008), pp. 1810–1830.

[39] Jitendra Kumar et al. “An efficient numerical technique for solving population balance equation involving aggregation, breakage, growth and nucleation”. In: *Powder Technology* 182.1 (Feb. 2008), pp. 81–104.

[40] Rajesh Kumar, Jitendra Kumar, and Gerald Warnecke. “Numerical methods for solving two-dimensional aggregation population balance equations”. In: *Computers & Chemical Engineering* 35.6 (June 2011), pp. 999–1009.

[41] Sanjeev Kumar and D. Ramkrishna. “On the solution of population balance equations by discretization—I. A fixed pivot technique”. In: *Chemical Engineering Science* 51.8 (Apr. 1996), pp. 1311–1332.

[42] Sanjeev Kumar and D. Ramkrishna. “On the solution of population balance equations by discretization—II. A moving pivot technique”. In: *Chemical Engineering Science* 51.8 (Apr. 1996), pp. 1333–1342.

[43] Sabine Le Borne and Lusine Shahmuradyan. “Algorithms for the Haar Wavelet Based Fast Evaluation of Aggregation Integrals in Population Balance Equations”. In: *Appl. Numer. Math.* 108.C (Oct. 2016), pp. 1–20.

[44] Sabine Le Borne and Lusine Shahmuradyan. “Fast algorithms for hp-discretized univariate population balance aggregation integrals”. In: *Computers & Chemical Engineering* 97 (Feb. 2017), pp. 1–12.

[45] Sabine Le Borne, Lusine Shahmuradyan, and Kai Sundmacher. “Fast evaluation of univariate aggregation integrals on equidistant grids”. In: *Computers & Chemical Engineering* 74 (Mar. 2015), pp. 115–127.

[46] Yulan Lin, Kangtaek Lee, and Themis Matsoukas. “Solution of the population balance equation using constant-number Monte Carlo”. In: *Chemical Engineering Science* 57.12 (June 2002), pp. 2241–2252.

[47] Shouci Lu, Yuqing Ding, and Jinyong Guo. “Kinetics of fine particle aggregation in turbulence”. In: *Advances in Colloid and Interface Science* 78.3 (Nov. 1998), pp. 197–235.

[48] Alan Mahoney and Doraiswami Ramkrishna. “Efficient solution of population balance equations with discontinuities by finite elements”. In: *Chemical Engineering Science - CHEM ENG SCI* 57 (Apr. 2002), pp. 1107–1119.

[49] Daniele L. Marchisio and Rodney O. Fox. “Solution of population balance equations using the direct quadrature method of moments”. In: *Journal of Aerosol Science* 36.1 (Jan. 2005), pp. 43–73.

[50] William J. Morokoff and Russel E. Caflisch. “Quasi-Monte Carlo Integration”. In: *Journal of Computational Physics* 122.2 (Dec. 1995), pp. 218–230.

[51] J. Mydlarz and D. Briedis. “Growth rate dispersion vs size-dependent growth rate for MSMPR crystallizer data”. In: *Computers & Chemical Engineering* 16.9 (Sept. 1992), pp. 917–922.

[52] M. Nicmanis and M. J. Hounslow. “Finite-element methods for steady-state population balance equations”. In: *AIChE Journal* 44.10 (Oct. 1998), pp. 2258–2272.

[53] Henri J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms*. Springer Berlin Heidelberg, 1982.

[54] I. V. Oseledets. “Tensor-Train Decomposition”. In: *SIAM Journal on Scientific Computing* 33.5 (Jan. 2011), pp. 2295–2317.

[55] Ivan Oseledets and Eugene Tyrtyshnikov. “TT-cross approximation for multidimensional arrays”. In: *Linear Algebra and its Applications* 432.1 (Jan. 2010), pp. 70–88.

[56] Vidyullatha P and D. Rajeswara Rao. “Machine Learning Techniques on Multidimensional Curve Fitting Data Based on R- Square and Chi-Square Methods”. In: *International Journal of Electrical and Computer Engineering (IJECE)* 6.3 (June 2016), p. 974.

[57] Mirko Peglow et al. “A new technique to determine rate constants for growth and agglomeration with size- and time-dependent nuclei formation”. In: *Chemical Engineering Science* 61.1 (Jan. 2006), pp. 282–292.

[58] M. Peifer and J. Timmer. “Parameter estimation in ordinary differential equations for biochemical processes using the method of multiple shooting”. In: *IET Systems Biology* 1.2 (Mar. 2007), pp. 78–88.

[59] A.A. Poyton et al. “Parameter estimation in continuous-time dynamic models using principal differential analysis”. In: *Computers & Chemical Engineering* 30.4 (Feb. 2006), pp. 698–708.

[60] M. V. Rakhuba and I. V. Oseledets. “Fast Multidimensional Convolution in Low-Rank Tensor Formats via Cross Approximation”. In: *SIAM J. Sci. Comput.* 37.2 (Jan. 2015), A565–A582.

[61] Rohit Ramachandran and Paul I. Barton. “Effective parameter estimation within a multi-dimensional population balance model framework”. In: *Chemical Engineering Science* 65.16 (Aug. 2010), pp. 4884–4893.

[62] Doraiswami Ramkrishna. *Population balances: Theory and applications to particulate systems in engineering*. Elsevier, 2000.

[63] E Ruckenstein. “Growth kinetics and the size distributions of supported metal crystallites”. In: *Journal of Catalysis* 29.2 (May 1973), pp. 224–245.

[64] Dmitry Savostyanov and Ivan Oseledets. “Fast adaptive interpolation of multi-dimensional arrays in tensor train format”. In: *The 2011 International Workshop on Multidimensional (nD) Systems*. IEEE, Sept. 2011.

[65] Mehakpreet Singh et al. “Solution of bivariate aggregation population balance equation: a comparative study”. In: *Reaction Kinetics, Mechanisms and Catalysis* 123.2 (Jan. 2018), pp. 385–401.

[66] Vasyl Skorych et al. “Investigation of an FFT-based solver applied to dynamic flowsheet simulation of agglomeration processes”. In: *Advanced Powder Technology* 30.3 (Mar. 2019), pp. 555–564.

[67] Marian Smoluchowski. “Drei Vorträge über Diffusion, Brownsche Molekularbewegung und Koagulation von Kolloidteilchen”. ger. In: *Pisma Mariana Smoluchowskiego* 2.1 (1927), pp. 530–594.

[68] William Stallings. *Operating systems: internals and design principles*. Boston: Prentice Hall, 2012.

[69] Hugo M. Vale and Timothy F. McKenna. “Solution of the Population Balance Equation for Two-Component Aggregation by an Extended Fixed Pivot Technique”. In: *Industrial & Engineering Chemistry Research* 44.20 (Sept. 2005), pp. 7885–7891.

[70] Steven R. White. “Density matrix formulation for quantum renormalization groups”. In: *Phys. Rev. Lett.* 69 (19 Nov. 1992), pp. 2863–2866.

[71] Choongseok Yoon and Robert McGraw. “Representation of generally mixed multivariate aerosols by the quadrature method of moments: I. Statistical foundation”. In: *Journal of Aerosol Science* 35.5 (May 2004), pp. 561–576.