Efficient and Accurate Evaluation of Aggregation Integrals in Population Balance Equations

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

genehmigte Dissertation

von
Lusine Shahmuradyan

aus
Eriwan, Armenien

2017

Gutachter:	Prof. Dr. Sabine Le Borne Prof. Dr. Volker John
Tag der mündlichen Prüfung:	31.03.2017

Acknowledgments

First of all, I would like to express my deep gratitude to my supervisor Sabine Le Borne for her individual supervision and remarkable suggestions throughout my thesis. Without her support and encouragement this work would not have been possible.

I would like to thank the DFG SPP 1679 "Dynamische Simulation vernetzter Feststoffprozesse" for the financial support and for the opportunity to present my work in various workshops. I am also very grateful to all my colleagues who were part of this project. In particular I would like to thank Volker John, Kai Sundmacher, Stefan Heinrich, Viktoria Wiedmeyer, Felix Anker, Vasyl Skorych, and Maksym Dosta for our three-year long collaboration.

I owe special thanks to Bruno Rubino and the "MathMods" consortium who gave me an opportunity to do my master's degree in Germany, paving the way for my PhD journey.

I am also thankful to my friends around the world who made my days brighter and more memorable.

Moreover, I am very grateful to my parents and my brother for their unconditional love in all periods of my life.

Last but not least, I would like to thank my husband Jonas for enormous support that has been invaluable. Without his understanding and support this work could not be completed.

Contents

Nomenclature **vii**

1 Introduction **1**
1.1 Population balance equations 1
1.2 New results 4
1.3 Outline of contents 6

2 Existing numerical methods **9**
2.1 Summary of existing methods 9
2.2 Sectional method on equidistant grids 11
2.3 Fixed pivot method on geometric grids 11

3 Prerequisites for the fast evaluation of aggregation integrals **13**
3.1 Separable kernel approximation and initial density distribution . . . 13
3.2 Grids and spaces 16
3.2.1 Equidistant grids and piecewise constant functions 16
3.2.2 Nested grids refined toward the origin and Haar wavelets . . 17
3.2.3 Nested grids refined toward an arbitrary point and high order polynomial 25
3.3 Convolution by FFT 30

4 Evaluation of univariate aggregation integrals with piecewise constant functions **33**
4.1 Evaluation of univariate aggregation integrals on equidistant grids . 33
4.1.1 Discretization through piecewise constant approximation . . 34
4.1.2 Sectional method with FFT 39
4.1.3 Numerical results 41
4.2 Evaluation of univariate aggregation integrals on nested grids refined toward the origin 54
4.2.1 Source term 55
4.2.2 Sink term 61

4.2.3 Numerical results 63
4.2.4 Change of mass 71

5 Evaluation of univariate aggregation integrals with high order polynomials on nested grids refined toward an arbitrary point 75
5.1 Source term 75
5.2 Summarized algorithm to compute the source aggregation integral . 89
5.3 Projections and prolongations 89
5.4 Sink term 91
5.5 Numerical results 93

6 Evaluation of bivariate aggregation integrals with piecewise constant functions on uniform meshes 101
6.1 Introduction of the bivariate problem 101
6.2 Source term 104
6.3 Sink term 108
6.4 Numerical results 112

7 Conclusions and Outlook 117

Appendices 119

A Auxiliary ξ-coefficients 121

B Auxiliary γ-coefficients 122

Bibliography 125

Nomenclature

Latin Symbols

Symbol	Comment	Formula
a, b	Kernel factors	(3.6), (6.4)
$\mathcal{C}$	Two-dimensional mesh	(6.8)
d	Number of particle properties	
e	Change of mass	(4.19)
f	Particle density distribution	(1.2), (6.1)
$\mathcal{G}$	Grid	(3.16), (3.20), (3.37), (6.7)
h	Interval size on a uniform grid	(3.16), (3.18), (3.33)
L	Number of refinements on a nested grid	(3.19)
m	Mass of density function	(4.1)
M	Kernel separation rank	(3.6), (6.4)
$\mathcal{M}$	One-dimensional mesh	(3.38)
n	Number of intervals on a uniform grid	(3.16), (3.18), (3.33)
N	Number of intervals on a nested grid	(3.19)
$\mathcal{N}$	Number of particles	(2.2)
$\mathcal{O}$	Complexity	
p	Polynomial degree	(3.42)
Q_{sink}	Aggregation sink term	(1.3), (6.2)
Q_{source}	Aggregation source term	(1.4), (1.5), (1.6), (6.3)
s	Shift from previous level	(3.33)
$\tilde{s}$	Absolute shift	(3.34)
$\mathcal{S}$	Test space	(3.17), (3.21), (3.42) (6.10)
t	Time	(1.2), (6.1)
x, y	mass of particle properties	

Greek Symbols

κ	Aggregation kernel	(3.6), (6.4)
φ, ψ	Product of density function and kernel factor	(3.7), (3.8), (6.5), (6.6)
Φ, Ψ	Basis Functions	(3.22),(3.23), (3.43)

Subscripts

agg	Aggregation
break	Breakage
C	constant
col	column
B	Brownian
d	Diagonal
K	Kinetic
nuc	Nucleation
off-diag	Off-diagonal
pc	piecewise constant
pl	piecewise linear
r	reduced
S	Shear
G	Gravitational

Acronyms

ACA	Adaptive Cross Approximation
FFT	Fast Fourier Transformation
IFFT	Inverse fast Fourier Transformation
PBE	Population Balance Equation
PSD	Particle size distribution
QMOM	Quadrature method of moments

Chapter 1

Introduction

In this chapter a general overview of population balance equations (PBE) is provided. In section 1.1 we briefly introduce processes considered in the modeling of population balance equations and formulate the underlying equations. In section 1.2 we specify the key requirements and components needed for the new technique described in this thesis and point out the novel contributions of this work. Finally, in section 1.3, we summarize the structure of the thesis.

1.1 Population balance equations

Applications of population balance equations can be found in modeling a variety of production processes in chemistry and biotechnology, where particles are dispersed in an environmental phase. Examples are the crystallization and precipitation of pharmaceutical materials, the synthesis of polymers, the formulation of emulsions, the generation of nanoparticles by flame pyrolysis, the growth of living cell populations in fermentation processes, and the separation of fermentation broths by flocculation and sedimentation. In all these processes, the dispersed phase consists of a population of particles which can be characterized by property coordinates $x \in \mathbb{R}^d$, where d is the number of properties considered, e.g. the particle mass, the particle area, or the chemical composition, to mention only a few. The dynamics of the whole particle population is described through a population balance equations and is quantified by a number density function $f(x,t)$ which describes the property distribution of the particles at a given time t [67].
The properties of particles in the system change due to several physical and chemical processes. Although we focus in this thesis on the aggregation process, we will also briefly discuss some of the other processes, i.e. breakage [37], [61], [4], growth

[35], [59], [24], nucleation [64], [37], [36].

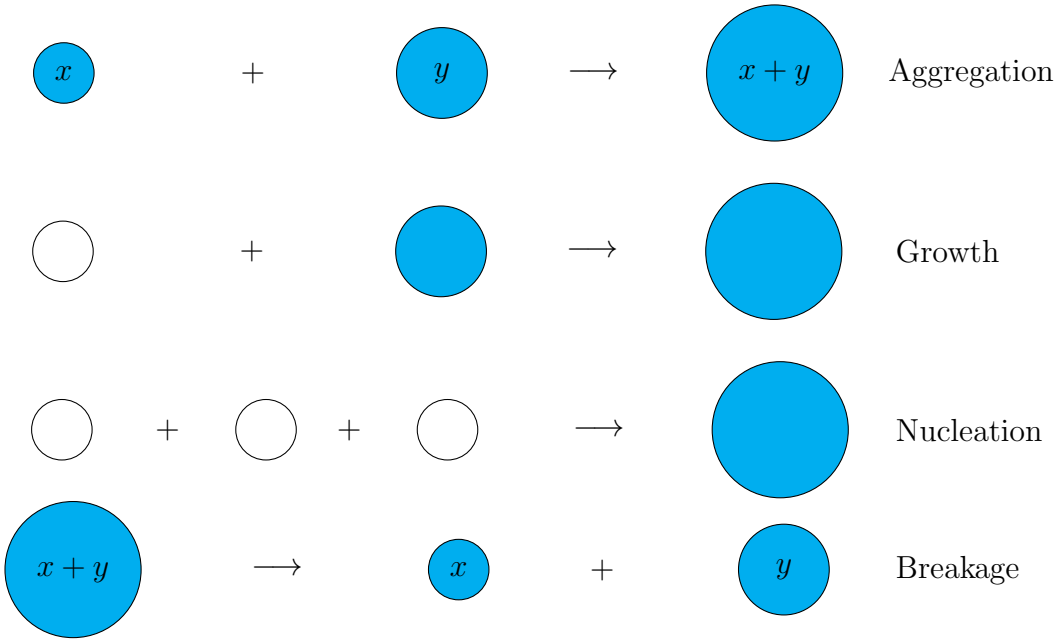


Figure 1.1: Processes in PBE: Aggregation, Growth, Nucleation, Breakage

Aggregation: Aggregation is a process in which two or more particles combine and form a larger particle (Figure 1.1, top). The surface of particles is covered by liquid binder and in the aggregation process the particles that are covered by a sufficiently thick layer of binder combine and form a larger particle. In the aggregation process the total mass of particles is preserved, while the number of particles is reduced.

Growth: Growth of particles happens when liquid binder spreads over the surface of particles (Figure 1.1, second). In this process the volume of particles increases while the number of particles remains constant.

Nucleation: Nucleation is a process in which vapor molecules condense and form new particles (Figure 1.1, third). In this process the number of particles changes, while the volume is not affected significantly.

Breakage: In the breakage process, large particles break into two or more fragments (Figure 1.1, bottom). It effects the number of particles but the total mass remains constant.

Nucleation, i.e. the birth of new particles in the continuous phase, is a local effect in the property space, while breakage and aggregation are long range effects, i.e. they describe the interaction of particles belonging to different parts of the property space.

A population balance equation describes the distribution change of particle properties caused by the above mentioned processes. Length or diameter are preferred properties for modeling the growth process. Since we focus on aggregation processes, we will choose as property the mass of particles. Assuming a constant density, particle mass can also be replaced with volume.

Due to growth, nucleation, aggregation, and breakage, the density function $f(x,t)$ varies dynamically with time. We assume that the particles are uniformly distributed in the space, such that the population density does not depend on the location of particles, usually referred to as an external coordinate. Neglecting these spatial variations, the dynamic evolution of the univariate density function is governed by a population balance equation which is a partial integro-differential equation

$$\frac{\partial f(x,t)}{\partial t} + \nabla_x \cdot (Gf(x,t)) = Q(f) = Q_{\text{nuc}}(f) + Q_{\text{break}}(f) + Q_{\text{agg}}(f), \qquad t \in [0,T]. \tag{1.1}$$

The second term of the left-hand side of (1.1) represents the growth of particles at the rate G. On the right-hand side, Q summarizes the birth and death terms of particles due to nucleation Q_{nuc}, breakage Q_{break} and aggregation Q_{agg}. Since we are interested in the aggregation process, we neglect terms modeling particle nucleation, growth and breakage. The dynamic evolution of f is then governed by a population balance equation of the form

$$\begin{aligned} \frac{\partial f(x,t)}{\partial t} &= Q_{\text{agg}}(f)(x,t) \\ &:= Q_{\text{source}}(f)(x,t) - Q_{\text{sink}}(f)(x,t), \qquad x \in (0,1],\ t \in [0,T], \end{aligned} \tag{1.2}$$

with aggregation source and sink terms

$$Q_{\text{source}}(f)(x,t) = \frac{1}{2} \int_0^x \kappa(x-y,y) f(x-y,t) f(y,t)\, dy, \tag{1.3}$$

$$Q_{\text{sink}}(f)(x,t) = f(x,t) \int_0^\infty \kappa(x,y) f(y,t)\, dy. \tag{1.4}$$

Here, $\kappa(x, y)$ stands for the kernel function describing the aggregation frequency in dependence on the properties of two aggregating particles, x and y. The source integral (1.3) describes the effect that new particles are generated by the combination of smaller ones, while the sink integral (1.4) quantifies the effect that particles are consumed by aggregation with others.

Note that we assume here that all particles have (non-dimensionalized) properties in the interval $(0, 1]$. Consequently, a particle of size x can only aggregate with particles up to the maximum size $(1 - x)$. The sink term then takes the following form

$$Q_{\text{sink}}(f)(x,t) = f(x,t) \int_0^{1-x} \kappa(x,y) f(y,t)\, dy, \tag{1.5}$$

providing mass conservation in $(0, 1]$. However, it restricts the validity of simulation results when larger particles are formed. An alternative form of the sink aggregation integral

$$Q_{\text{sink}}(f)(x,t) = f(x,t) \int_0^{1} \kappa(x,y) f(y,t)\, dy \tag{1.6}$$

with the upper limit 1 can be used. In this case while we compute the solution of (1.2) in (0,1], mass conservation holds in (0,2].

1.2 New results

This thesis focuses on the efficient and accurate evaluation of the aggregation integrals in population balance equations. The source aggregation integral (1.3) is quadratic with respect to f and is of convolution type, being usually numerically expensive to evaluate. In a previous study, the calculation of the aggregation integral took around 85% of the overall computational time of a population balance simulation [3].

It is desirable to develop a numerical technique which yields the density distribution, in particular the aggregation integral terms, at high precision and low computational costs. For this purpose, in a list of papers [28], [29], [30], [31], [32], [34], Hackbusch has introduced a new approach. In [28], an algorithm is developed where the density distribution is discretized through piecewise constant functions and the property coordinate x on equidistant grids or on nested grids refined toward the origin. Further treatment of the piecewise constant case with focus on

mass conservation is given in [29], [31]. The piecewise linear case is discussed in [32], and the general hp case for arbitrary polynomial order p on nested grids is developed in [30], [34]. This approach drastically reduces the evaluation cost of the most challenging integral term (1.3) in each time step to $\mathcal{O}((p+1)n \log n)$ if the interval of property coordinate x is divided into n subintervals.

This thesis is based upon and extends our previously published papers [48], [47], [49]. The aim of these papers and this thesis is to further explore the techniques whose theoretical foundation has been established by Hackbusch. In [49] numerical results illustrating algorithms for various kernel functions, an extension to the sink term and a comparison with the sectional method are provided for uniform grids. In [47], the evaluation of aggregation source and sink integrals is performed on nested grids refined toward the origin, and a comparison with the fixed pivot method is provided. In [48] numerical results are obtained through discretization with high order polynomials on nested grids refined toward an arbitrary point. The key components of the developed algorithms are a separable approximation of the aggregation kernel, a nested grid consisting of piecewise uniform portions, application of FFT to compute the aggregation (convolution) on such uniform portions and orthogonality of basis functions which in combination lead to efficient recursion formulas.

The novel contributions of this thesis consist of the development and numerical illustration of algorithms that permit the evaluation of the aggregation source and sink terms in almost optimal complexity. In particular, this thesis presents

- techniques for the approximation of different kernel functions taken from the literature. The need for a separable approximation does not appear to be a drawback of the proposed methods: Many of the kernel functions proposed in the literature are either separable themselves or have global or local exponentially convergent separable approximations. For a global approximation, we implement and compare two methods, using Chebyshev polynomials and the adaptive cross approximation (ACA) approach, methods that are new in terms of their application in this context. For kernel functions that are neither separable nor suited for a global separable approximation, we suggest blockwise separable approximations in a hierarchical fashion,

- the development of an efficient implementation of the (almost) optimal complexity evaluation technique for the source term of the aggregation integral whose theoretical foundation has been established by Hackbusch. We propose suitable data structures, expose connections between different parts of the algorithms that can be exploited for their efficient implementation and provide explicit algorithmic representations,

- the development of a novel efficient evaluation algorithm for the aggregation sink term on uniform and graded grids, exploiting properties of kernel separation and basis functions,
- numerical tests for the popular sectional approach on uniform grids and the fixed pivot technique on geometrically graded grids in comparison with the approach proposed in this paper,
- extensive numerical tests for different initial setups to illustrate the performance of the developed algorithms with respect to their accuracy and efficiency, leading to (heuristic) strategies for the choice of the discretization parameters and numerical illustrations of the constants involved in the complexity estimates,
- an extension of the theory established for univariate problems, i.e., mass is considered as the only particle property, to bivariate problems where a second property of particles is considered, providing also algorithmic representation and numerical tests.

1.3 Outline of contents

In chapter 2 we summarize existing numerical methods to solve population balance equations. Moreover, we discuss in more detail the sectional method on uniform grids and the fixed pivot method on geometric grids. These two methods are used for comparison with the new approach and algorithms introduced in this thesis.

In chapters 3, 4 and 5 we discuss one dimensional aggregation integrals. Chapter 3 introduces the key components, i.e. kernel separation, grids, orthogonal spaces and fast Fourier transformation (FFT), that are necessary for fast and efficient evaluation of aggregation integrals. In this chapter we also present the initial distribution and kernel functions taken from the literature that will be used for numerical tests.

In chapter 4 we discuss the evaluation of aggregation integrals using a piecewise constant approximations of the density function. In section 4.1 the method is applied on equidistant grids and in section 4.2 on nested grids refined toward the origin. The numerical results obtained in section 4.1 and 4.2 are compared with results obtained by sectional and fixed pivot methods, respectively. Moreover, in this chapter we discuss and compare different techniques for obtaining separable approximations for different kernel functions.

In chapter 5 the density distribution is approximated through high order polynomials and the grid is refined toward an arbitrary point, allowing a more accurate approximation. In this chapter we perform intensive numerical tests for different initial setups and establish criteria for the choice of the discretization parameters, i.e., polynomial degree and number of grid points.

In chapter 6 the approach proposed in section 4.1 is extended for treating bivariate aggregation integrals, significantly reducing the computational complexity. Here, we also summarize techniques existing in the literature for solving bivariate population balance equations.

All numerical tests in this thesis have been performed on a Lenovo X1 Thinkpad (Intel i5-5200U processor, 4 GB RAM, 2.2 GHz), and in all tests the time derivative in the population balance equation (1.1) is discretized through an explicit Euler scheme. All algorithms introduced in this thesis have been developed as stand-alone code in C++, and have been as well integrated in external software. Algorithms given in section 4.1 and 4.2 have been integrated into the MoonMD library [38] and algorithms in section 4.1 into Dyssol, a software developed in the framework of the DFG Schwerpunktprogramm "Dynamische Simulation vernetzter Feststoffprozesse – DynSim-FP" (SPP 1679).

The work in section 4.1 is published in Computers and Chemical Engineering [49] and the work in section 4.2 is published in Applied Numerical Mathematics [47]. A publication covering the chapter 5 has been submitted to Computers and Chemical Engineering [48]. This work has also been presented at different conferences through talks and posters: Among those are the Young Researchers Meeting, Plön, 2014; 10^{th} European Congress of Chemical Engineering (ECCE15), Nice, 2015; Summer School on Crystal Shape Engineering, Zurich 2015; several workshops within the Schwerpunktprogramm "Dynamische Simulation vernetzter Feststoffprozesse – DynSim-FP" (SPP 1679).

Chapter 2

Existing numerical methods

In this chapter we discuss different approaches for solving population balance equations introduced in the literature. In section 2.1 we summarize existing numerical methods. In sections 2.2 and 2.3 we discuss in more detail the sectional method on uniform grids and the fixed pivot method on geometric grids. Later, we compare the new approach for the fast and efficient evaluation of aggregation integrals presented in this thesis with those two techniques.

2.1 Summary of existing methods

Analytical solutions of PBEs with aggregation terms can be obtained only in a few cases, in particular for $\kappa(x, y) = \text{const}$. Laplace transformation is one method for obtaining the analytical solution. Thus, various numerical techniques have been developed to solve population balance equations. In most cases in addition to evaluating the particle property distribution, it is also required to preserve certain moments, i.e. the number of particles (zeroth moment) or mass (first moment). The k-th moment, of a density function $f(t, x)$ at time $t \in [0, T]$ is defined as

$$m_k(t) = \int_0^1 x^k f(t, x)\, dx. \tag{2.1}$$

One of the most popular numerical techniques is the sectional method that is based on fixed or moving pivotal points that approximate size distribution by finite numbers of size sections. The simplest approach to discretize the particle property distribution is using equidistant grids [71], [72], [39], [22], [69]. The disadvantage of equidistant grids is the necessity of taking a very large number

of intervals in order to obtain an accurate approximation, leading to very high computational cost. Toward the goal of (further) reducing the computational complexity, graded meshes have been proposed which are finer for small compared to large particle sizes. The motivation for using such a grid is given in situations with larger quantities of small particles compared to large ones, suggesting finer partitions for the small particles. The particular structure of the suggested nested grading results from the observation that the aggregation of particles of similar sizes leads to new particles of approximately twice the size. The geometric grid, defined through gridpoints $x_{i+1} = 2^{\rho} x_i$ with an initial gridpoint $x_0 > 0$ and a suitable $\rho \in (0, 1]$, is a popular example of a graded mesh.

Different approaches for sectional methods using geometric grids are proposed in [23], [36], [55]. Sectional methods are able to preserve two moments of the particle size distribution. In [45] a fixed pivot method has been proposed, where both the number of particles and mass are preserved and geometric grids are used. The method however over-predicts the number density for large particles. This drawback is overcome in the moving pivot method introduced in [46]. In [42] an improved sectional method, the cell average technique, has been established which computes the average size of the newborn particles in a cell before assigning it to the neighboring pivots and thereby the cell average technique achieves higher accuracy and better convergence. In these methods, if particles of pivot size aggregate to new particles of non-pivot size, they are distributed to the two neighboring pivots through weighting factors (see section 2.3), preserving two moments of the particle size distribution. However, the approach can be generalized to preserve more moments by distributing non-pivotal size particles to more than two pivots.

Another technique for solving population balance equations is the method of moments [5], [53], [56]. The standard method of moments is applicable to size-independent aggregation and breakage kernels only. The QMOM (quadrature method of moments) overcomes this problem [57], [27]. The advantage of QMOM to the sectional method is that it is able to track with very small error all the moments involved in the quadrature approximation. But the main disadvantage of using the QMOM is that the PSD itself is not accessible.

The finite volume method [7], [9], [13], [58], [74], [20] is suitable for solving aggregation-breakage equations when the objective is the prediction of the first moment, i.e. mass, because this approach preserves mass automatically. Further techniques for solving population balance equations are the finite elements method [54], [60], [68], and Monte Carlo simulation methods [40], [50], [51], [70].

2.2 Sectional method on equidistant grids

In this section we briefly summarize the macroscopic discrete formulation for equidistant grids, also known as the sectional method (see e.g. chapter 4.5 in [67]). This method may be considered as the simplest approach to discretize the population balance equation. The discretization steps are summarized as follows:

1. Define an equidistant grid $\mathcal{G} := (x_0, x_1, \ldots, x_n)$, $x_i = ih$ for $h = 1/n$, on the interval $[0, 1]$.
2. Define "macroscopic" variables

$$\mathcal{N}_i := \int_{x_{i-1}}^{x_i} f(x)dx, \tag{2.2}$$

   which are also referred to as "number concentrations" or "numbers of particles" associated with a characteristic size, e.g. x_i.
3. Discretize the kernel function, e.g., $\kappa_{i,j} := \kappa(x_i, x_j)$ for $i, j = 1, \ldots, n$.
4. Under the simplifying assumption (discretization) that the aggregation of particles of size x_i (counted in $\mathcal{N}_i$) with particles of size x_j (counted in $\mathcal{N}_j$) forms particles of size x_{i+j} (counted in $\mathcal{N}_{i+j}$), one obtains

$$Q_i = \frac{1}{2}\sum_{j=1}^{i-1} \kappa_{j,i-j}\mathcal{N}_j\mathcal{N}_{i-j} - \mathcal{N}_i \sum_{j=0}^{n-i} \kappa_{i,j}\mathcal{N}_j, \qquad i = 1, \ldots, n. \tag{2.3}$$

The first sum in (2.3) represents the source term of the aggregation (1.3) while the second sum represents the sink term (1.5). In this setup, no particles of size larger than x_n may be formed as explained in (1.5)

2.3 Fixed pivot method on geometric grids

In this section we briefly review the popular fixed pivot technique. The fixed pivot technique can be applied to general, unstructured grids. However, most often it is applied to geometric grids with vertices defined through $v_{i+1} = \tau v_i$ for given initial $v_0 = 0$, $v_1 > 0$, $\tau > 1$ (Figure 2.1). In the context of the fixed pivot method, such geometric grids are preferable over the locally refined nested grids (Figure 3.4) used for the methods developed in this thesis since the fixed pivot method is second order accurate on geometric grids and only first order accurate on locally refined grids [25, 43].

The grid vertices v_i divide the entire particle size range into n sections. The population of the particles in the i-th section $(v_i, v_{i+1}]$ is discretized by a macroscopic variable $\mathcal{N}_i$, representing particles of the pivot size $x_i = \frac{1}{2}(v_i + v_{i+1})$. If particles

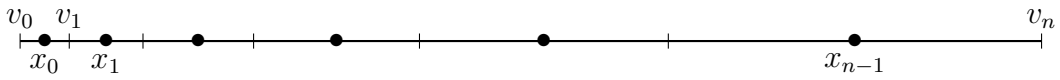


Figure 2.1: Geometric grid $v_0 = 0, v_1 = a, v_{i+1} = \tau v_i, i = 1, \ldots, n-1$.

of pivot size x_j and x_k aggregate to new particles of (non-pivot) size $x_j + x_k$, these new particles are distributed to the macroscopic variables $\mathcal{N}_i, \mathcal{N}_{i+1}$ of neighboring pivots sizes $x_i \leq x_j + x_k < x_{i+1}$ with weighting factors $\epsilon_i^{\pm}(\cdot)$ (2.5). This leads to the following fixed pivot method

$$\begin{aligned}\frac{dQ_i(t)}{dt} = &\sum_{x_i \leq x_j + x_k < x_{i+1}}^{j \geq k} (1 - 0.5 \cdot \delta_{j,k}) \cdot \epsilon_i^{+}(x_k + x_j)\kappa(x_k, x_j)\mathcal{N}_j\mathcal{N}_k \\ &+ \sum_{x_{i-1} \leq x_j + x_k < x_i}^{j \geq k} (1 - 0.5 \cdot \delta_{j,k}) \cdot \epsilon_i^{-}(x_k + x_j)\kappa(x_k, x_j)\mathcal{N}_j\mathcal{N}_k \qquad (2.4) \\ &- \mathcal{N}_i \sum_{j=1}^{n} \kappa(x_i, x_j)\mathcal{N}_j \qquad \text{for } \; i = 1, \ldots, n\end{aligned}$$

(with Kronecker $\delta_{j,k}$).

The distribution factors $\epsilon_i^{\pm}(\cdot)$ can be chosen such that any two moments of the density function are preserved. In the typical case of preserving the first two moments (number and mass), this requires the coefficients given in (2.5),

$$\epsilon_i^{\pm}(x) = \frac{x - x_{i\pm1}}{x_i - x_{i\pm1}}. \qquad (2.5)$$

While the fixed pivot method can be applied on arbitrary grids which can be chosen finer in regions of interest, its disadvantage is the quadratic complexity $\mathcal{O}(n^2)$ in the number of unknowns (pivots) n [45, 67]. Related methods such as the sectional method discussed in previous subsection, moving pivot technique [46] or the cell average technique [42] share this disadvantage of quadratic complexity.

Chapter 3

Prerequisites for the fast evaluation of aggregation integrals

In this chapter we introduce the key components required for the fast evaluation of aggregation integrals. In section 3.1 we discuss the separable approximation of aggregation kernels and provide insights on how to treat different types of kernels. We also introduce a list of kernels and density distributions that are used for the numerical results in the following chapters. In section 3.2 we discuss different types of grids that are suitable for our problem. In addition, we introduce function spaces and bases for these spaces. Finally, the section 3.3 reviews the fast Fourier transformation that can be used to compute convolution integrals in $\mathcal{O}(n \log n)$ complexity.

3.1 Separable kernel approximation and initial density distribution

The aggregation source and sink integrals (1.3), (1.5) include a kernel function $\kappa(x, y)$ which specifies the rate at which particles of mass x and y aggregate. In this thesis, to obtain numerical results, we will consider the following representative aggregation kernels taken from the literature [1, 11] and illustrated for $(x, y) \in (0, 1]^2$ in Figure 3.1:

$$\kappa_{\mathrm{C}}(x,y) := 1 \qquad \text{constant kernel} \tag{3.1}$$

$$\kappa_{\mathrm{B}}(x,y) := (x^{\frac{1}{3}} + y^{\frac{1}{3}}) \cdot (x^{-\frac{1}{3}} + y^{-\frac{1}{3}}) \qquad \text{Brownian motion (continuum)} \tag{3.2}$$

$$\kappa_{\mathrm{S}}(x,y) := (x^{\frac{1}{3}} + y^{\frac{1}{3}})^{\frac{7}{3}} \qquad \text{shear (non-linear velocity profile)} \tag{3.3}$$

$$\kappa_{\mathrm{G}}(x,y) := (x^{\frac{1}{3}} + y^{\frac{1}{3}})^2 \cdot |x^{\frac{2}{3}} - y^{\frac{2}{3}}| \qquad \text{inertia and gravitational settling} \tag{3.4}$$

$$\kappa_{\mathrm{K}}(x,y) := (x^{\frac{1}{3}} + y^{\frac{1}{3}})^2 \cdot (xy)^{\frac{1}{2}}(x+y)^{-\frac{3}{2}} \qquad \text{based on kinetic theory} \tag{3.5}$$

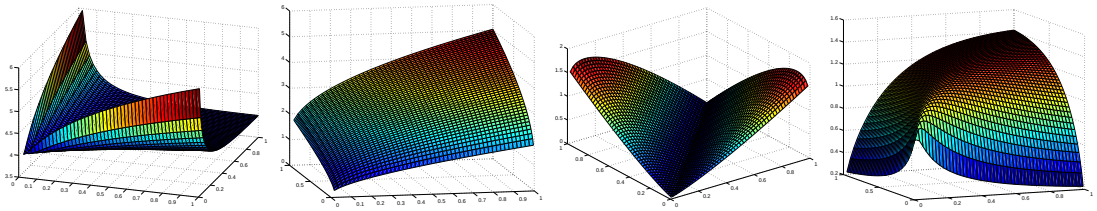


Figure 3.1: Aggregation kernels: Brownian κ_{B} (first), shear κ_{S} (second), gravitational κ_{G} (third), kinetic κ_{K} (fourth).

The efficient evaluation of the aggregation source and sink integrals will be based on a separable approximation of the kernel. A function $\kappa(x,y)$ is called separable with separation rank M if it can be expressed in the form

$$\kappa(x,y) = \sum_{\nu=1}^{M} a_\nu(x) b_\nu(y), \tag{3.6}$$

with some suitable functions $\alpha_\nu(x)$ and $\beta_\nu(x)$.

The aggregation source (1.3) and sink (1.5) terms can then be computed as sums of M terms of the form

$$Q_{\mathrm{source}}(x) = \tfrac{1}{2}\int_0^x a_\nu(x-y) f(x-y) f(y) b_\nu(y)\,dy = \tfrac{1}{2}\int_0^x \psi_\nu(x-y)\varphi_\nu(y)\,dy, \tag{3.7}$$

$$Q_{\mathrm{sink}}(x) = a_\nu(x) f(x) \int_0^{1-x} b_\nu(y) f(y)\,dy = \psi_\nu(x) \int_0^{1-x} \varphi_\nu(y)\,dy \tag{3.8}$$

for functions $\psi_\nu := a_\nu \cdot f,\ \varphi_\nu := b_\nu \cdot f$.

Among the five aggregation kernels presented above, only the constant kernel κ_{C} and Brownian motion kernels κ_{B} are separable. The Brownian kernel has separation rank 3 in view of

$$\kappa_{\mathrm{B}}(x,y) = 2 + x^{\frac{1}{3}} y^{-\frac{1}{3}} + x^{-\frac{1}{3}} y^{\frac{1}{3}} = \sum_{\nu=1}^{3} a_\nu(x) b_\nu(y), \tag{3.9}$$

with $a_1(x) = \sqrt{2}$, $a_2(x) = x^{\frac{1}{3}}$, $a_3(x) = x^{-\frac{1}{3}}$ and $b_1(y) = \sqrt{2}$, $b_2(y) = y^{-\frac{1}{3}}$, $b_3(y) = y^{\frac{1}{3}}$.
For the remaining three kernel functions, one may approximate $\kappa(x, y)$ either by a global separable rank-M approximation

$$\kappa(x, y) \approx \sum_{\nu=1}^{M} a_\nu(x) b_\nu(y), \tag{3.10}$$

or, if necessary, locally by

$$\kappa(x, y) \approx \sum_{\nu=1}^{M_{i,j}} a_\nu^{i,j}(x) b_\nu^{i,j}(y) \qquad \text{for } (x, y) \in I_i \times I_j \tag{3.11}$$

with subintervals $I_i, I_j \subset (0, 1]$. The global approach (3.10) will turn out to be suitable for the shear kernel κ_{S} as well as for the kinetic kernel κ_{K}. The gravitational kernel, however, requires locally different separable approximations (3.11).

We will provide details on the construction of separable approximations for the different kernels in subsection 4.1.3 on numerical results. In general, there exist several analytic approaches in the literature to derive such separable approximations, including (Chebyshev) interpolation, adaptive or hybrid cross approximation (ACA, HCA) algorithms, sinc approximation or approximation by exponential sums [33], [6], [26], [8].

Alternatively, a separable approximation could also be computed on the discrete side. This approach would require the computation of a low rank (global approach) or a blockwise low rank (local approach) approximation to the discrete kernel matrix $K = (\kappa_{i,j}) \in \mathbb{R}^{n \times n}$ where $\kappa_{i,j} := \kappa\left(\frac{i}{n}, \frac{j}{n}\right)$.

In numerical tests performed throughout the thesis we will use one of the following functions as the initial density distribution,

$$f_1(0, x) = e^{-1000(x-0.125)^2}, \tag{3.12}$$
$$f_2(0, x) = e^{-1100(x-0.1)^2} + e^{-1100(x-0.2)^2}, \tag{3.13}$$
$$f_3(0, x) = e^{-100(x-0.125)^2}, \tag{3.14}$$
$$f_4(0, x) = e^{-100(x-0.25)^2}. \tag{3.15}$$

This functions are plotted in Figure 3.2. Depending on numerical tests we will test these initial distributions for a different number of grid intervals and polynomial order.

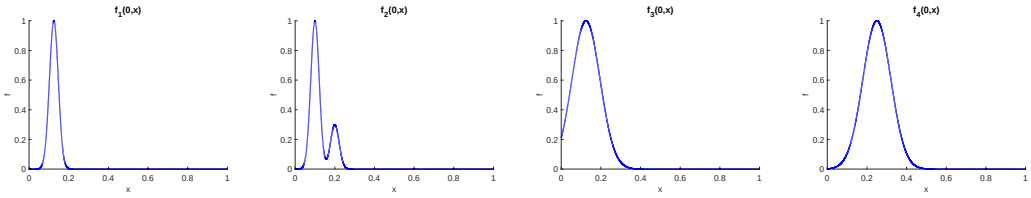


Figure 3.2: Left to right: Initial density distributions $f_1(0,x)$ (3.12), $f_2(0,x)$ (3.13), $f_3(0,x)$ (3.14), $f_4(0,x)$ (3.15).

3.2 Grids and spaces

In this section we discuss grids ($\mathcal{G}$) and spaces ($\mathcal{S}$) that are used to develop the algorithms for the efficient evaluation of aggregation integrals. In subsection 3.2.1 we will discuss the simplest case where equidistant grids and piecewise constant functions are used. In subsection 3.2.2 nested grids refined toward the origin are defined. Here the efficient evaluation of aggregation integrals requires the introduction of Haar wavelets that will be used as basis functions. Finally, in subsection 3.2.3 we extend the approach to grids refined toward an arbitrary point and approximate the density distribution f through polynomials of high order. For simplicity, in all discussed cases we approximate the factors $a_\nu(x)$ and $b_\nu(y)$ (3.6) of the separable kernel approximation through piecewise constant functions. The actual evaluation of aggregation integrals using the grids and spaces introduced in subsections 3.2.1, 3.2.2, 3.2.3 is given in sections 4.1, 4.2 and chapter 5, respectively.

3.2.1 Equidistant grids and piecewise constant functions

On the interval $[0,1]$ equidistant grids are defined as

$$\mathcal{G} := \{(x_0, x_1, \ldots, x_n) \mid x_i = ih, h = 1/n\}, \tag{3.16}$$

as shown in Figure 3.3.

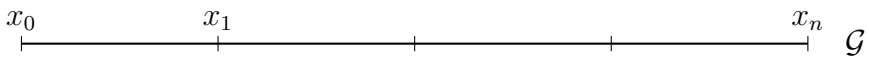


Figure 3.3: An equidistant grid $\mathcal{G}$ consisting of 5 nodes with $n = 4$ intervals and $h = 1/4$.

We define the function space

$$\mathcal{S} := \{f \mid f \text{ is piecewise constant with respect to the mesh } \mathcal{G}\}. \tag{3.17}$$

We assume that $f \in \mathcal{S}$. If this assumption is not satisfied, then f needs to be projected onto $\mathcal{S}$ using a (preferably mass-preserving) projection (see 2.1).

3.2.2 Nested grids refined toward the origin and Haar wavelets

A different type of graded grid, a so-called locally refined nested grid refined toward the origin, permits an evaluation algorithm of almost optimal complexity. Locally refined nested grids are similar to the geometrically graded grids (Figure 2.1) that we used in the fixed pivot method in the sense that intervals close to the origin are smaller than those further away. However, they differ since they are of a nested nature, i.e., the finer mesh results from the local refinement of (certain intervals of) a coarser one.

The fast algorithms for the evaluation of the aggregation integrals (1.3) and (1.5) will be based on a recursion through grid levels which requires the grid to have a structure of nested uniform grids, which will allow the application of FFT. An example for a locally refined nested grid is given in Figure 3.4, and the formal construction is as follows: The locally refined mesh is defined through $L+1$ levels

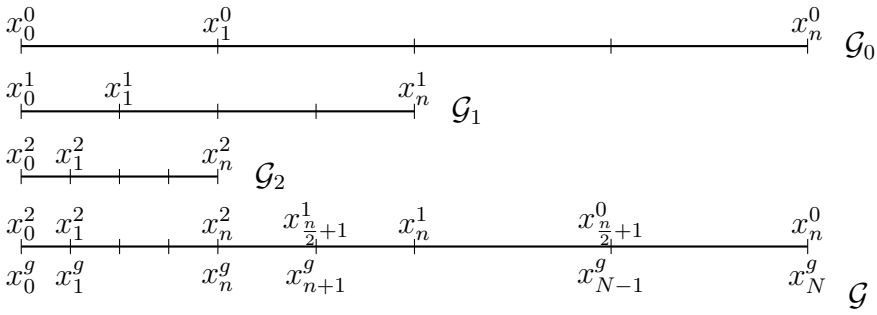


Figure 3.4: A coarse grid $\mathcal{G}_0$ consisting of $n = 4$ intervals, two refined grids $\mathcal{G}_1, \mathcal{G}_2$, (i.e., $L = 2$ refinements) and the resulting locally refined nested grid $\mathcal{G}$.

of equidistant grids, each consisting of n subintervals. We begin its construction with the coarsest grid $\mathcal{G}_0$ which consists of $n+1$ gridpoints $x_i^0 := ih,\ i = 0, \ldots, n$, resulting in n equidistant subintervals of equal length $h = 1/n$.

Assumption 1. *In all subsequent derivations and algorithms, we assume that n is a power of 2, i.e., $n = 2^\theta$ for some $\theta \in \mathbb{N}$.*

The grids $\mathcal{G}_\ell,\ \ell = 1, \ldots, L$, are obtained by dividing each of the first $\frac{n}{2}$ subintervals of the grid $\mathcal{G}_{\ell-1}$ into two equal-sized subintervals, leading to a grid on $[0, 2^{-\ell}]$

consisting of $n+1$ equidistant grid points x_i^ℓ and n intervals of size h_ℓ satisfying

$$h_\ell = 2^{-\ell}h = 2^{-\ell} \cdot \frac{1}{n} = 2^{-\ell-\theta}, \qquad x_i^\ell = ih_\ell \quad (\theta \in \mathbb{N}). \tag{3.18}$$

We now define the graded mesh $\mathcal{G}$ starting from the finest mesh $\mathcal{G} := \mathcal{G}_L$ and then repeatedly adding the last $\frac{n}{2}$ subintervals of the mesh on the next coarser level.

Figure 3.4 shows an example of an original grid $\mathcal{G}_0$ and $L = 2$ refined grids $\mathcal{G}_\ell,\ 1 \leq \ell \leq L$, each consisting of $n = 4$ subintervals as well as the resulting graded grid $\mathcal{G}$.

It is straightforward to see that the graded mesh consists of n intervals of length $2^{-L}h$ (finest level) and $\frac{n}{2}$ intervals of respective lengths $2^{-\ell}h$ for all other levels $0 \leq \ell < L$, leading to the total number of intervals

$$N = n + L \cdot \frac{n}{2} = \left(1 + \frac{L}{2}\right) n. \tag{3.19}$$

The graded mesh is characterized through the set of its $N+1$ gridpoints: Given the points of the (uniform) grids,

$$\mathcal{G}_\ell := \{x_i^\ell = i2^{-\ell} \cdot \frac{1}{n} \mid 0 \leq i \leq n\},$$

we have

$$\mathcal{G} \ := \ \bigcup_{\ell=0}^{L} \mathcal{G}_\ell \ = \ \mathcal{G}_L \cup \left\{ x_i^\ell \mid 0 \leq \ell < L,\ \frac{n}{2} + 1 \leq i \leq n \right\}. \tag{3.20}$$

As a next step, we define function spaces associated with the (uniform and graded) grids $\mathcal{G}_\ell$ and $\mathcal{G}$, following the notation introduced in [28]. The space we will use to approximate the density distribution function f and the aggregation integrals $Q_{\text{source}}(f)$ (1.3) and $Q_{\text{sink}}(f)$ (1.5) on the graded mesh $\mathcal{G}$ is simply

$$\mathcal{S} := \{f \mid f \text{ is piecewise constant with respect to the mesh } \mathcal{G}\}. \tag{3.21}$$

The dimension of $\mathcal{S}$ equals the number of intervals in $\mathcal{G}$, i.e., $\dim(\mathcal{S}) = N$ (3.19). In order to efficiently work with these functions, we will have to introduce a basis for $\mathcal{S}$, and to this end we define additional auxiliary function spaces as follows (some functions in these spaces are illustrated in Figure 3.5):

$\mathcal{V}_\ell(0,1]$: piecewise constant functions on an equidistant grid of size $h_\ell := 2^{-\ell}h$. (The underlying grid differs from the previously defined $\mathcal{G}_\ell$ since it covers the entire interval $(0,1]$, not only $(0, 2^{-\ell}]$).

$\mathcal{W}_\ell(0,1]$: the orthogonal complement of $\mathcal{V}_{\ell-1}(0,1]$ within $\mathcal{V}_\ell(0,1]$ (with respect to the L_2 inner product), i.e. $\mathcal{V}_\ell(0,1] = \mathcal{V}_{\ell-1}(0,1] \oplus \mathcal{W}_\ell(0,1]$.

$\mathcal{V}_\ell(0,2^{-\ell}]$: subspace of $\mathcal{V}_\ell(0,1]$, consisting of those functions in $\mathcal{V}_\ell(0,1]$ with support within $(0,2^{-\ell}]$.

$\mathcal{W}_\ell(0,2^{-\ell}]$: subspace of $\mathcal{W}_\ell(0,1]$, consisting of those functions in $\mathcal{W}_\ell(0,1]$ with support within $(0,2^{-\ell}]$.

Using these auxiliary spaces, we can write our previously defined function space $\mathcal{S}$ as

$$\begin{aligned}\mathcal{S} &= \mathcal{V}_0(0,1] \oplus \mathcal{W}_1(0,2^{-1}] \oplus \ldots \oplus \mathcal{W}_L(0,2^{-L}] \\ &= \mathcal{V}_0(0,1] + \mathcal{V}_1(0,2^{-1}] + \ldots + \mathcal{V}_L(0,2^{-L}] \\ &\subseteq \mathcal{V}_L(0,1] = \mathcal{V}_0(0,1] \oplus \mathcal{W}_1(0,1] \oplus \ldots \oplus \mathcal{W}_L(0,1].\end{aligned}$$

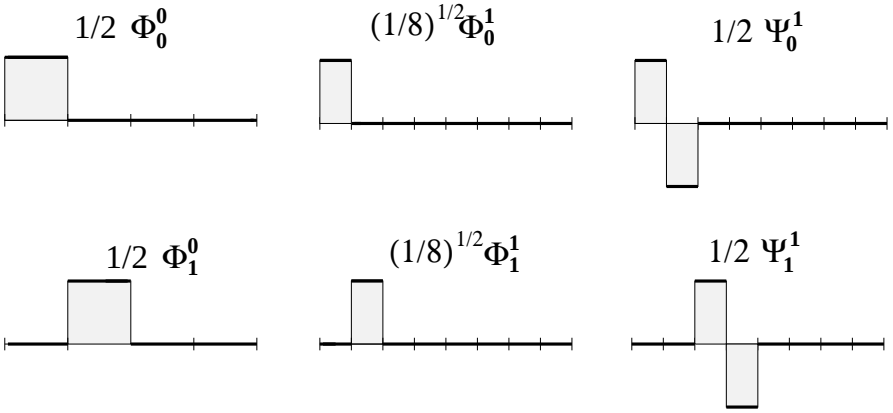


Figure 3.5: The first two (scaled) standard basis functions of $\mathcal{V}_0(0,1]$ (left), $\mathcal{V}_1(0,1]$ (middle) and $\mathcal{W}_1(0,1]$ (right) for the initial grid width $h = \frac{1}{4}$.

We will now provide bases for these subspaces. We first define the canonical basis functions (with support on a single interval) for the spaces $\mathcal{V}_\ell(0,1]$:

$$\Phi_j^\ell(x) := \begin{cases} \frac{1}{\sqrt{h_\ell}} & : \quad x \in (jh_\ell, (j+1)h_\ell]\,, \\ 0 & : \quad \text{else}, \end{cases} \qquad \text{for } j = 0, \ldots, 2^\ell n - 1. \tag{3.22}$$

The scaling has been chosen so that each basis is an orthonormal set. The basis functions Φ_j^ℓ, $j = 0, \ldots, n-1$, form a basis of $\mathcal{V}_\ell(0,2^{-\ell}]$. Some of these basis functions are illustrated in Figure 3.5 (left and middle, scaled by $\sqrt{h_\ell}$).

We next define basis functions for the orthogonal complement spaces $\mathcal{W}_\ell(0,1]$, which are also known as *Haar wavelets* and have support on two neighboring

intervals:

$$\Psi_j^\ell(x) = \begin{cases} +1/\sqrt{2h_\ell} & : \quad x \in (2jh_\ell, (2j+1)h_\ell], \\ -1/\sqrt{2h_\ell} & : \quad x \in ((2j+1)h_\ell, 2(j+1)h_\ell)] \\ 0 & : \quad \text{else}, \end{cases} \tag{3.23}$$

for $j = 0, \ldots, 2^{\ell-1}\, n - 1$. Two such basis functions (this time scaled by $\sqrt{2h_\ell}$) are illustrated in Figure 3.5 (right). The required orthogonality conditions are easily checked to be satisfied. Once more, a basis for $\mathcal{W}_\ell(0, 2^{-\ell}]$ is given by the first $\frac{n}{2}$ of these basis functions.

Given the above functions $\Phi_j^\ell(x)$ and $\Psi_j^\ell(x)$, there are (at least) the following two possibilities to form a basis for the space $\mathcal{S}$ of piecewise constant functions on the graded mesh $\mathcal{G}$:

1. Standard nodal basis, composed of (a subset of) canonical (scaled) basis functions $\Phi_j^\ell(x) \in \mathcal{V}_\ell(0, 1]$ on all levels of refinement:

$$f = \underbrace{\sqrt{h_L} \sum_{j=0}^{n-1} f_j^L \Phi_j^L}_{\substack{\text{piecewise constant on} \\ \text{finest grid } \mathcal{G}_L}} + \sum_{\ell=0}^{L-1} \underbrace{\sqrt{h_\ell} \sum_{j=\frac{n}{2}}^{n-1} f_j^\ell \Phi_j^\ell}_{\substack{\text{piecewise constant on} \\ \text{right half of grid } \mathcal{G}_\ell}} \tag{3.24}$$

   The scaling factors $(\sqrt{h_\ell}, \ell = 0, \ldots, L)$ ensure that the coefficients f_j^ℓ are the actual function values of f on the respective subintervals (see left of Figure 3.6).

2. Haar wavelet basis, composed of the standard nodal basis on the coarsest level, $\{\Phi_0^0(x), \ldots, \Phi_{n-1}^0(x)\}$, and complemented by Haar wavelet basis functions $\Psi_j^\ell(x)$ of the complement spaces $\mathcal{W}_\ell(0, 2^{-\ell}]$:

$$f = \underbrace{\sum_{j=0}^{n-1} F_j^0 \Phi_j^0}_{\substack{\text{piecewise constant on} \\ \text{coarsest grid } \mathcal{G}_0}} + \sum_{\ell=1}^{L} \underbrace{\sum_{j=0}^{n/2-1} F_j^\ell \Psi_j^\ell}_{\text{piecewise constant on grid } \mathcal{G}_\ell}. \tag{3.25}$$

The locations of supports for basis functions associated with coefficients f_j^ℓ in the nodal basis and coefficients F_j^ℓ in the Haar basis representation are illustrated in Figure 3.6.

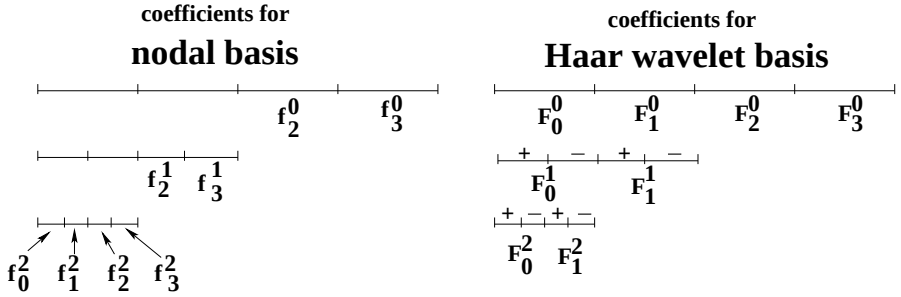


Figure 3.6: Locations of supports for basis functions associated with coefficients f_j^ℓ in the nodal basis (left) and coefficients F_j^ℓ in the Haar basis (right) representation.

We introduce the following two coefficient matrices to store these coefficients in the respective basis representations:

$$f_{\text{coeff}}^{\text{nodal}} = \begin{pmatrix} 0 & \dots & 0 & f_{n/2}^0 & \dots & f_{n-1}^0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & f_{n/2}^{L-1} & \dots & f_{n-1}^{L-1} \\ f_0^L & \dots & f_{n/2-1}^L & f_{n/2}^L & \dots & f_{n-1}^L \end{pmatrix} \in \mathbb{R}^{(L+1)\times n}, \tag{3.26}$$

$$F_{\text{coeff}}^{\text{Haar}} = \left(\begin{array}{ccc|ccc} F_0^0 & \dots & F_{n/2-1}^0 & F_{n/2}^0 & \dots & F_{n-1}^0 \\ \hline F_0^1 & \dots & F_{n/2-1}^1 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ F_0^L & \dots & F_{n/2-1}^L & 0 & \dots & 0 \end{array}\right) \tag{3.27}$$

$$=: \begin{pmatrix} * & * \\ F_{\text{fine}}^{\text{Haar}} & 0 \end{pmatrix} \in \mathbb{R}^{(L+1)\times n} \quad \text{with} \quad F_{\text{fine}}^{\text{Haar}} \in \mathbb{R}^{L\times\frac{n}{2}}. \tag{3.28}$$

We will later (in subsections 4.2.2 and 4.2.1) need transformations between these two basis representations. It is essential for our subsequent algorithm that the transformation of coefficients f_j^ℓ in the nodal basis (3.24) to the coefficients F_j^ℓ in the Haar basis (3.25), as well as the back transformation, is only of complexity $\mathcal{O}(N)$ (3.19), see e.g. [52].

Such a fast wavelet transformation is possible since the Haar basis is an orthonormal basis, and therefore the coefficients F_j^ℓ in (3.25) for the representation of a function f with respect to a Haar wavelet basis can be computed as

$$F_j^\ell = \int_0^1 f \cdot \Psi_j^\ell \, dx.$$

An implementation for the fast wavelet transformation that transforms the nodal basis coefficients $f_{\text{coeff}}^{\text{nodal}}$ (3.26) into the Haar basis coefficients $F_{\text{coeff}}^{\text{Haar}}$ (3.27) is provided in Algorithm 1.

Algorithm 1 $F = \text{nodal\_to\_Haar}(n, L, f)$ (preliminary version)
Nodal (3.24) to Haar (3.25) basis transformation using $f := f_{\text{coeff}}^{\text{nodal}}$ (3.26) and $F := F_{\text{coeff}}^{\text{Haar}}$ (3.27), n: number of intervals of the coarsest grid, L: number of grid refinements.

1: **for** $j = 0 : (n-1)$ **do**
2: $\quad r_j^L = f_j^L$;
3: **end for**
4: **for** $\ell = L : -1 : 1$ **do**
5: $\quad h_\ell = 2^{-\ell} \cdot \frac{1}{n}$; (grid width on level ℓ)
6: $\quad$ **for** $j = 0 : \left(\frac{n}{2} - 1\right)$ **do**
7: $\quad\quad F_j^\ell = h_L(r_{2j}^\ell - r_{2j+1}^\ell)\frac{1}{\sqrt{2h_\ell}}$;
8: $\quad\quad r_j^{\ell-1} = r_{2j}^\ell + r_{2j+1}^\ell$;
9: $\quad\quad r_{\frac{n}{2}+j}^{\ell-1} = 2^{L-\ell+1} \cdot f_{\frac{n}{2}+j}^{\ell-1}$;
10: $\quad$ **end for**
11: **end for**
12: **for** $j = 0 : \left(\frac{n}{2} - 1\right)$ **do**
13: $\quad F_j^0 = h_L \cdot r_j^0 \cdot \sqrt{n}$; $(\sqrt{n} = \frac{1}{\sqrt{h_0}})$
14: $\quad F_{\frac{n}{2}+j}^0 = h_L \cdot r_{\frac{n}{2}+j}^0 \cdot \sqrt{n}$;
15: **end for**
16: **return** (F);

In lines 8, 9, Algorithm 1 computes auxiliary values r_j^ℓ. These values will be required again in a subsequent Algorithm 8 in the scaled form

$$\tilde{r}_j^\ell := \frac{h_L}{\sqrt{h_\ell}} r_j^\ell. \tag{3.29}$$

Hence, we reformulate the (preliminary) Algorithm 1 into Algorithm 2 which returns these scaled values in a matrix

$$A := \left(\tilde{r}_j^\ell\right)_{\substack{0 \le \ell \le L \\ 0 \le j \le n-1}} \in \mathbb{R}^{(L+1)\times n} \tag{3.30}$$

in addition to the Haar basis coefficients $F = F_{\text{fine}}^{\text{Haar}}$ (3.28) for all but the coarsest level. In particular, the values F_j^0 are already included in the matrix A as follows from Lemma 1.

Algorithm 2 $(A, F) = \text{nodal\_to\_Haar}(n, L, f)$
Nodal (3.24) to Haar (3.25) basis transformation.

Input:	$n \in \mathbb{N}$;	(problem size)
	$L \in \mathbb{N}$;	(number of refinements)
	$f = f_{\text{coeff}}^{\text{nodal}} \in \mathbb{R}^{(L+1)\times n}$;	(nodal values (3.26));
Output:	$F := F_{\text{fine}}^{\text{Haar}} \in \mathbb{R}^{L\times \frac{n}{2}}$;	(Haar coefficients (3.28));
	$A \in \mathbb{R}^{(L+1)\times n}$;	(auxiliary matrix (3.30));

$h_L = 2^{-L} \cdot \frac{1}{n}$; (grid width on finest level L)
for $j = 0 : (n-1)$ **do**
  $\tilde{r}_j^L = \sqrt{h_L} f_j^L$;
end for
for $\ell = L : -1 : 1$ **do**
  for $j = 0 : \left(\frac{n}{2} - 1\right)$ **do**
    $F_j^\ell = (\tilde{r}_{2j}^\ell - \tilde{r}_{2j+1}^\ell)\frac{1}{\sqrt{2}}$;
    $\tilde{r}_j^{\ell-1} = (\tilde{r}_{2j}^\ell + \tilde{r}_{2j+1}^\ell)\frac{1}{\sqrt{2}}$;
    $\tilde{r}_{\frac{n}{2}+j}^{\ell-1} = \sqrt{h_L 2^{L-\ell+1}} f_{\frac{n}{2}+j}^{\ell-1}$;
  end for
end for
return (A, F);

Lemma 1. *Let $f_k \in \mathcal{S}$ with support $f_k \subset (0, 2^{-k}]$ for a graded mesh $\mathcal{G}$ with n intervals on the coarse (initial) mesh $\mathcal{G}_0$ and L levels of refinement. Then it holds that the Haar basis representation of f_k with respect to levels $k, k+1, \ldots, L$ (i.e., now taking the grid $\mathcal{G}_k$ as the coarsest grid),*

$$f_k = \underbrace{\sum_{j=0}^{n-1} F_j^k \Phi_j^k}_{\substack{\text{piecewise constant} \\ \text{on grid } \mathcal{G}_k}} + \sum_{\ell=k+1}^{L} \underbrace{\sum_{j=0}^{n/2-1} F_j^\ell \Psi_j^\ell}_{\text{piecewise constant on grid } \mathcal{G}_\ell}, \tag{3.31}$$

satisfies $F_j^k = \tilde{r}_j^k$, $j = 0, \ldots, n-1$, for the (scaled) coefficients of the (standard) basis functions Φ_j^k (3.22) on level k, where the coefficients $\tilde{r}_j^k$ are given in (3.29) with r_j^k computed in lines 8, 9 of Algorithm 1. In particular, for the coarsest level $k = 0$, this implies that the first row of the coefficient matrix F_{coeff}^{Haar} (3.27) is identical to the first row of the coefficient matrix A (3.30), i.e.,

$$(F_0^0, \ldots, F_{n-1}^0) = (\tilde{r}_0^0, \ldots, \tilde{r}_{n-1}^0). \tag{3.32}$$

Proof. We use induction (for $k = L, \ldots, 0$) to prove the representation (3.31) with coefficients $F_j^k = \tilde{r}_j^k$, $j = 0, \ldots, n-1$.
For $k = L$ it holds that f_L has support $f_L \subset (0, 2^{-L}]$, and its nodal (3.24) and Haar (3.25) basis representations with respect to level L (considering this now to be the only level) are

$$f_L = \sqrt{h_L} \sum_{j=0}^{n-1} f_j^L \Phi_j^L = \sum_{j=0}^{n-1} F_j^L \Phi_j^L,$$

i.e., it holds that $F_j^L = \sqrt{h_L} f_j^L \overset{\text{Alg. 1 line 2}}{=} \sqrt{h_L} r_j^L \overset{(3.29)}{=} \tilde{r}_j^L$.
We now assume that the proposition already holds for all f_j, $j = L, \ldots, k+1$, and show that it also holds for f_k.
Let now $f_k \in \mathcal{S}$ with support $f_k \subset (0, 2^{-k}]$. In order to represent f_k in a Haar basis with respect to levels $k, \ldots, L$ as in (3.31), we first note that $f_k|_{(2^{-(k+1)}, 2^{-k}]} \in V_k(0, 1]$, i.e., it is piecewise constant with respect to the coarsest considered level k on the right half of its support. Hence, there must hold $F_j^k = \sqrt{h_k} f_j^k$ for $j = \frac{n}{2}, \ldots, n-1$. This is realized in line 9 of Algorithm 1 along with (3.29):

$$\tilde{r}_j^k \overset{(3.29)}{=} \frac{h_L}{\sqrt{h_k}} r_j^k \overset{\text{Alg. 1 line 9}}{=} \frac{h_L}{\sqrt{h_k}} 2^{L-k} f_j^k \overset{(3.18)}{=} \sqrt{h_k} f_j^k = F_j^k, \qquad j = \frac{n}{2}, \ldots, n-1.$$

In order to show $F_j^k = \tilde{r}_j^k$ for $j = 0, \ldots, \frac{n}{2}-1$, we use the induction assumption: The Haar basis representation of f_k restricted to the left half of its support, $f_k|_{(0,2^{-(k+1)}]}$, with respect to levels $k+1, \ldots, L$ is given by

$$f_k|_{(0,2^{-(k+1)}]} = \sum_{j=0}^{n-1} \tilde{r}_j^{k+1} \Phi_j^{k+1} + \sum_{\ell=k+2}^{L} \sum_{j=0}^{n/2-1} F_j^\ell \Psi_j^\ell.$$

The only thing left to do is to rewrite the first sum with respect to a Haar basis on levels k and $k+1$, i.e, to determine coefficients F_j^k so that

$$\sum_{j=0}^{\frac{n}{2}-1} F_j^k \Phi_j^k + \sum_{j=0}^{\frac{n}{2}-1} F_j^{k+1} \Psi_j^{k+1} = \sum_{j=0}^{n-1} \tilde{r}_j^{k+1} \Phi_j^{k+1}.$$

Forming the inner product on both sides with Φ_j^k and exploiting orthogonality yields

$$\begin{aligned} F_j^k &= \left\langle \sum_{m=0}^{n-1} \tilde{r}_m^{k+1} \Phi_m^{k+1}, \Phi_j^k \right\rangle = \tilde{r}_{2j}^{k+1} \left\langle \Phi_{2j}^{k+1}, \Phi_j^k \right\rangle + \tilde{r}_{2j+1}^{k+1} \left\langle \Phi_{2j+1}^{k+1}, \Phi_j^k \right\rangle \\ &\overset{(3.22)}{=} (\tilde{r}_{2j}^{k+1} + \tilde{r}_{2j+1}^{k+1}) h_{k+1} \frac{1}{\sqrt{h_{k+1}}\sqrt{h_k}} \overset{(3.29)}{=} \frac{h_L}{\sqrt{h_{k+1}}} (r_{2j}^{k+1} + r_{2j+1}^{k+1}) \frac{\sqrt{h_{k+1}}}{\sqrt{h_k}} \\ &= \frac{h_L}{\sqrt{h_k}} (r_{2j}^{k+1} + r_{2j+1}^{k+1}) \overset{\text{Alg. 1 line 8}}{=} \frac{h_L}{\sqrt{h_k}} r_j^k = \tilde{r}_j^k. \end{aligned}$$

□

3.2.3 Nested grids refined toward an arbitrary point and high order polynomial

In this subsection we introduce nested grids that are refined toward an arbitrary point. An example for such a locally refined nested grid, here refined toward the coarse grid point $x_3^0 = \frac{3}{8}$ of $\mathcal{G}_0$, is given in Figure 3.7.

A nested grid in general is defined by the parameters L (number of refinements, leading to $L+1$ levels), n (number of equidistant intervals per level) and a set of shifts s_ℓ, $\ell = 1, \ldots, L$.
The grids $\mathcal{G}_\ell$, $\ell = 1, \ldots, L$, are obtained by dividing $\frac{n}{2}$ consecutive subintervals of the grid $\mathcal{G}_{\ell-1}$ each into two equal-sized subintervals. Hence, the (uniform) grids $\mathcal{G}_\ell$ consist of $n+1$ equidistant grid points x_i^ℓ and n intervals of size h_ℓ. The starting

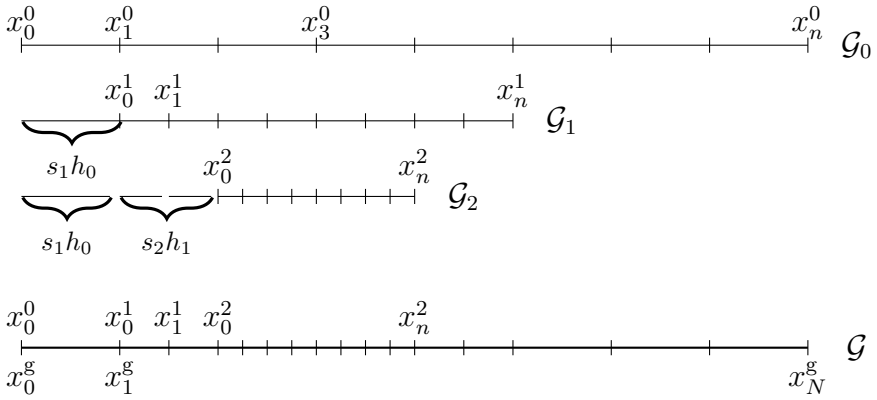


Figure 3.7: A coarse grid $\mathcal{G}_0$ consisting of $n = 8$ intervals, two refined grids $\mathcal{G}_1, \mathcal{G}_2$, (i.e., $L = 2$ refinements) and the resulting locally refined nested grid $\mathcal{G}$. The shifts of the starting points of refined grids with respect to the previous level are $s_1 = 1$ and $s_2 = 2$, resp.. The absolute shifts (3.34) are $\tilde{s}_1 = 2$ and $\tilde{s}_2 = 8$.

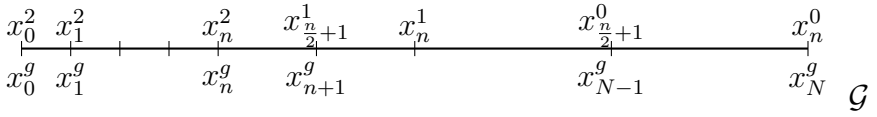


Figure 3.8: A nested grid refined toward $x_0^0 = 0$, originating from a coarse grid $\mathcal{G}_0$ of $n = 4$ intervals and $L = 2$ refinements of the two leftmost intervals on a given level.

point x_0^ℓ of the grid $\mathcal{G}_\ell$ is specified through a shift parameter s_ℓ, denoting the distance to the starting point $x_0^{\ell-1}$ of the next coarser grid $\mathcal{G}_{\ell-1}$ in terms of the number of intervals in $\mathcal{G}_{\ell-1}$, i.e., $x_0^\ell = x_0^{\ell-1} + s_\ell h_{\ell-1}$, $\ell = 1, \ldots, L$. We set $s_0 := 0$. Setting (the coarsest level interval width) $h := \frac{1}{n}$, we obtain

$$h_\ell = 2^{-\ell} h, \qquad x_i^\ell = \left(\sum_{k=1}^{\ell} s_k h_{k-1} \right) + i h_\ell, \qquad i = 0, \ldots, n \qquad \ell = 0, \ldots, L. \tag{3.33}$$

We define the absolute shifts $\tilde{s}_\ell$ to be the number of intervals (in terms of the interval size h_ℓ) by which the starting point x_0^ℓ of the grid $\mathcal{G}_\ell$ is shifted from the starting point $x_0^g = x_0^0$ of the nested grid $\mathcal{G}$,

$$\tilde{s}_0 = 0, \quad \tilde{s}_\ell = 2(\tilde{s}_{\ell-1} + s_\ell), \quad \ell = 1, \ldots L. \tag{3.34}$$

The nested grid $\mathcal{G}$ consists of n intervals of length $2^{-L}h$ (on the finest level) and $\frac{n}{2}$ intervals of respective lengths $h_\ell = 2^{-\ell}h$ for all other levels $0 \le \ell < L$, leading

to the total number of intervals given in (3.19).
This grid is characterized through the set of its $N+1$ grid points. Given the $(n+1)$ points (3.33) of the (uniform) grids

$$\mathcal{G}_\ell := \{x_i^\ell = \sum_{k=1}^{\ell} s_k h_{k-1} + i2^{-\ell}h \mid 0 \le i \le n\}, \tag{3.35}$$

we define

$$\begin{aligned} \mathcal{G} &:= \bigcup_{\ell=0}^{L} \mathcal{G}_\ell = \mathcal{G}_L \cup \left(\bigcup_{\ell=1}^{L} \mathcal{G}_{\ell-1} \backslash \mathcal{G}_\ell \right) && (3.36) \\ &= \mathcal{G}_L \cup \left(\bigcup_{\ell=1}^{L} \left\{ x_i^{\ell-1} \mid \ i \in \{0, \ldots, s_\ell - 1\} \cup \{s_\ell + \frac{n}{2} + 1, \ldots, n\} \right\} \right) && (3.37) \end{aligned}$$

The set $\mathcal{M}_\ell$ of n intervals associated with a grid $\mathcal{G}_\ell$ is defined as

$$\mathcal{M}_\ell \ := \ \left\{ I_i^\ell := \left(x_i^\ell, x_{i+1}^\ell\right] \mid 0 \le i \le n-1 \right\}, \tag{3.38}$$

and the intervals for the nested grid $\mathcal{G}$ (with intervals I_i^ℓ defined in (3.38)) are collected in

$$\mathcal{M}_\mathcal{G} := \mathcal{M}_L \cup \{I_i^{\ell-1} \mid \ i \in \mathcal{I}_\ell, \ 1 \le \ell \le L\} \tag{3.39}$$

for the index sets $\mathcal{I}_\ell$ of unrefined intervals on a given level ℓ defined by

$$\mathcal{I}_\ell := \{0, \ldots, s_\ell - 1\} \cup \{s_\ell + \frac{n}{2}, \ldots, n-1\}. \tag{3.40}$$

In addition, we define a set of intervals $\mathcal{M}_\ell^I$ in which the sizes of intervals coincide with those of the mesh $\mathcal{M}_\ell$, but they cover the entire domain $I := (0, 1]$,

$$\mathcal{M}_\ell^I := \{\hat{I}_i^\ell := (ih_\ell, (i+1)h_\ell] \mid 0 \le i \le 2^\ell n - 1\}. \tag{3.41}$$

To approximate the functions appearing in the aggregation integrals $Q_{\text{source}}(x)$ (1.3) and $Q_{\text{sink}}(x)$ (1.6) on the graded mesh $\mathcal{G}$, we will use the space of discontinuous piecewise polynomials,

$$\mathcal{S} = \mathcal{S}(\mathcal{M}_\mathcal{G}) := \{g \in L^\infty((0,1]) \ \mid \ g|_{I_i^\ell} \in \Pi_{p_i^\ell}, p_i^\ell \ge 1 \text{ for all } \ I_i^\ell \in \mathcal{M}_\mathcal{G}\} \tag{3.42}$$

where $\Pi_{p_i^\ell}$ denotes the space of polynomials of degree up to p_i^ℓ. The dimension of this space is given by

$$\dim(\mathcal{S}) = \sum_{I_i^\ell \in \mathcal{M}_\mathcal{G}} (p_i^\ell + 1)$$

and simplifies to $\dim(\mathcal{S}) = (p+1)N$ if a fixed polynomial degree p is used on all N intervals.
In (3.42) we chose polynomials of at least first order ($p \geq 1$), since in this case mass conservation is implied (see the proof in [29], Theorem 4.1). In the case of piecewise constant polynomials ($p = 0$), to yield mass conservation, additional modifications are required. This modifications are derived in [29]. This case is, however, not further pursued in this thesis.

In order to later define a (nested and orthonormal with respect to the L_2 inner product) basis for the introduced function space, we define shifted and scaled Legendre polynomials (see Figure 3.9 for an illustration). While other orthonormal sets of basis functions are also possible, it is shown in [31], subsection 8.1, that Legendre polynomials are the best choice. Given the Legendre polynomial $\mathcal{L}_\alpha$ ([30], Appendix A) of degree α, we define

$$\begin{aligned}
\Phi^0_{0,\alpha}(x) &:= \begin{cases} \sqrt{\frac{2}{h}}\mathcal{L}_\alpha(-1+\frac{2x}{h}) & : \ x \in (0,h), \\ 0 & : \ \text{else}, \end{cases} \\
\Phi^\ell_{0,\alpha}(x) &:= 2^{\ell/2}\Phi^0_{0,\alpha}(2^\ell x), \\
\Phi^\ell_{i,\alpha}(x) &:= \Phi^\ell_{0,\alpha}(x - ih_\ell) = 2^{\ell/2}\Phi^0_{0,\alpha}(2^\ell x - ih)
\end{aligned} \tag{3.43}$$

for polynomial degrees $\alpha \in \{0, \ldots, p\}$ with $p := \max\{p_i^\ell \mid I_i^\ell \in \mathcal{M}_\mathcal{G}\}$, levels $0 \leq \ell \leq L$ and shifts $0 \leq i \leq n-1$.

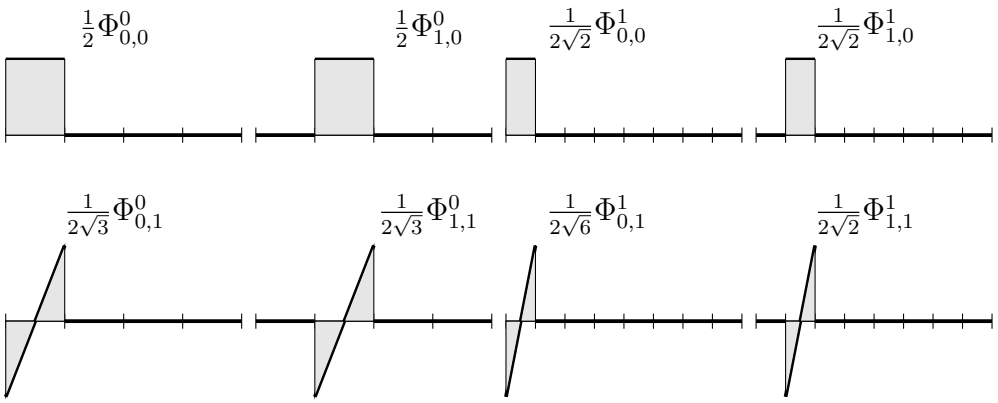


Figure 3.9: Scaled and shifted Legendre polynomials of degree $\alpha = 0$ (first row) and $\alpha = 1$ (second row); the first two per row on level $\ell = 0$, the second two on level $\ell = 1$ for an initial grid width $h_0 = \frac{1}{4}$.

Finally, we define orthonormal sets B and their spanned spaces $\mathcal{S}$ on all levels $0 \leq \ell \leq L$, both restricted to unrefined intervals appearing in the nested grid as

well as on the entire domain $I = (0, 1]$, as

$$\begin{aligned} B_\ell &= \{\Phi^\ell_{i,\alpha} \mid i \in \mathcal{I}_\ell,\ 0 \le \alpha \le\ p^\ell_i\}, & \mathcal{S}_\ell &:= \operatorname{span}(B_\ell), \\ B^I_\ell &= \{\Phi^\ell_{i,\alpha} \mid i \in \{0, \dots, 2^\ell n - 1\},\ 0 \le \alpha \le\ p^\ell_i\}, & \mathcal{S}^I_\ell &= \operatorname{span}(B^I_\ell). \end{aligned} \tag{3.44}$$

The spaces $\mathcal{S}^I_\ell$ are nested, i.e., $\mathcal{S}^I_{\ell-1} \subset \mathcal{S}^I_\ell$. The set $B_\mathcal{G} := \bigcup_{\ell=0}^{L} B_\ell$ forms an orthonormal basis of the space $\mathcal{S} = \bigoplus_{\ell=0}^{L} \mathcal{S}_\ell$ originally defined in (3.42). In the remainder of the thesis, we use a fixed polynomial degree p for all intervals, i.e., $p^\ell_i = p$ for all $I^\ell_i \in \mathcal{M}_G$. Given the basis $B_\mathcal{G}$, there exist coefficients $\varphi^\ell_{i,\alpha} \in \mathbb{R}$ such that a function $\varphi \in \mathcal{S}$ can be represented as

$$\varphi = \sum_{\ell=0}^{L} \underbrace{\sum_{i \in \mathcal{I}_\ell} \sum_{\alpha=0}^{p} \varphi^\ell_{i,\alpha} \Phi^\ell_{i,\alpha}}_{=:\varphi_\ell} = \sum_{\ell=0}^{L} \varphi_\ell, \qquad \varphi_\ell \in \mathcal{S}_\ell \subset \mathcal{S}^I_\ell. \tag{3.45}$$

Multiplying both sides of the equation (3.45) by $\Phi^m_{j,\beta} \in B_\mathcal{G}$, integrating over $\mathbb{R}$ and using the orthonormality of the basis functions

$$\int_{\mathbb{R}} \Phi^\ell_{i,\alpha}(x) \Phi^m_{j,\beta}(x)\ dx = \begin{cases} 1 & : \quad i = j,\ \ell = m,\ \alpha = \beta, \\ 0 & : \quad \text{else}, \end{cases} \tag{3.46}$$

yields the coefficients $\varphi^\ell_{i,\alpha}$ in (3.45), i.e,

$$\varphi^\ell_{i,\alpha} = \int_{I^\ell_i} \varphi(x)\ \Phi^\ell_{i,\alpha}(x)\ dx. \tag{3.47}$$

Since in (3.7) and (3.8) the factors $a_i(\cdot),\ b_i(\cdot)$ of the separable approximation of the kernel κ are approximated through piecewise constant functions and we approximate the density distribution f through a piecewise polynomial, the evaluation of the source integral is reduced to the convolution $\psi * \varphi$ of two functions $\psi, \varphi \in \mathcal{S}$ for a suitably chosen nested grid $\mathcal{G}$. Note that we approximate the factors $a_i(\cdot),\ b_i(\cdot)$ through piecewise constant functions for simplicity. While higher order approximation of these factors is also possible, in this case $\varphi,\ \psi \in \mathcal{S}$ would not necessarily hold, requiring a different treatment for spaces to which f and φ, ψ belong to.

3.3 Convolution by FFT

In this section, we discuss the fast Fourier transformation (FFT) on equidistant grids as an efficient means to evaluate the convolution sum

$$\omega_{i+1} := \sum_{j=0}^{i} \varphi_j \psi_{i-j}, \qquad i = 0, \ldots, n-1. \tag{3.48}$$

It is well-known that a sequence of three Fourier transformations can lead to the efficient computation of a convolution sum (see, e.g., (2.3.1.17) in [21] or [28]). The fast evaluation of (3.48) is based on the well known convolution theorem ([10]) stating that under suitable conditions the Fourier transform of a convolution is the pointwise product of Fourier transforms.
As a prerequisite in our particular setting, we need to expand the input vectors $(\varphi_0, \ldots, \varphi_{n-1})^T, (\psi_0, \ldots, \psi_{n-1})^T \in \mathbb{R}^n$ to

$$\begin{aligned} \bar{\varphi} &:= (\bar{\varphi}_0, \ldots, \bar{\varphi}_{2n-1}) := (\varphi_0, \ldots, \varphi_{n-1}, 0, \ldots, 0) \in \mathbb{R}^{2n}, \\ \bar{\psi} &:= (\bar{\psi}_0, \ldots, \bar{\psi}_{2n-1}) := (\psi_0, \ldots, \psi_{n-1}, 0, \ldots, 0) \in \mathbb{R}^{2n}. \end{aligned}$$

Then (3.48) can be equivalently written as

$$\omega_{\nu+1} := \sum_{\mu=0}^{2n-1} \varphi_\mu \psi_{\nu-\mu}, \qquad \nu = 0, \ldots, n-1. \tag{3.49}$$

Computing coefficients $\hat{\varphi}_\gamma$ and $\hat{\psi}_\lambda$, $\gamma, \lambda = 0, \ldots, 2n-1$ such that

$$\bar{\varphi}_\mu = \frac{1}{2n} \sum_{\gamma=0}^{2n-1} \hat{\varphi}_\gamma e^{i\gamma\mu\pi/n}, \quad \bar{\psi}_\mu = \frac{1}{2n} \sum_{\lambda=0}^{2n-1} \hat{\psi}_\lambda e^{i\lambda\mu\pi/n} \tag{3.50}$$

holds for all $\mu = 0, \ldots, 2n-1$ and inserting this representation into the convolution sum of the form (3.49) with extended values $\bar{\psi}$ and $\bar{\varphi}$, we obtain

$$\begin{aligned} \omega_{\nu+1} &= \sum_{\mu=0}^{2n-1} \left(\frac{1}{2n} \sum_{\gamma=0}^{2n-1} \hat{\varphi}_\gamma e^{i\gamma\mu\pi/n} \right) \left(\frac{1}{2n} \sum_{\lambda=0}^{2n-1} \hat{\psi}_\lambda e^{i\lambda(\nu-\mu)\pi/n} \right) \\ &= \frac{1}{2n} \sum_{\gamma=0}^{2n-1} \hat{\varphi}_\gamma \sum_{\lambda=0}^{2n-1} \hat{\psi}_\lambda e^{i\lambda\nu\pi/n} \underbrace{\left(\frac{1}{2n} \sum_{\mu=0}^{2n-1} e^{i\mu(\gamma-\lambda)\pi/n} \right)}_{= \frac{1}{2n} 2n \delta_{\gamma,\nu} \text{ (Kronecker delta)}} \\ &= \frac{1}{2n} \sum_{\gamma=0}^{2n-1} \underbrace{\hat{\varphi}_\gamma \hat{\psi}_\gamma}_{=:\hat{\omega}_{\gamma+1}} e^{i\gamma\nu\pi/n} = \frac{1}{2n} \sum_{\gamma=0}^{2n-1} \hat{\omega}_{\gamma+1} e^{i\gamma\nu\pi/n}. \end{aligned} \tag{3.51}$$

To summarize, the fast evaluation of (3.49) is based on the following three steps

1. Compute the coefficients $\hat{\varphi}_\gamma$ and $\hat{\psi}_\lambda$ such that (3.50) holds. It is well-known that these coefficients can be computed through fast Fourier transformations (FFTs) and are given by

$$\hat{\varphi} = \mathrm{FFT}(\bar{\varphi}), \quad \hat{\psi} = \mathrm{FFT}(\bar{\psi}) \quad \text{where } (\mathrm{FFT}(y))_\mu := \sum_{\gamma=0}^{2n-1} y_\gamma e^{-i\gamma\mu\pi/n}. \tag{3.52}$$

2. Compute $\hat{\omega}_{\gamma+1} = \hat{\varphi}_\gamma \cdot \hat{\psi}_\gamma$ for $0 \le \gamma \le 2n-1$.

3. Compute

$$\omega_{\nu+1} = \frac{1}{2n} \sum_{\gamma=0}^{2n-1} \hat{\omega}_{\gamma+1} e^{i\gamma\nu\pi/n} \qquad \text{for } 0 \le \nu \le 2n-1. \tag{3.53}$$

   We are only interested in the first n values $\omega_{\nu+1}$, $\nu = 0, \ldots, n-1$; however, we will use an inverse FFT (IFFT) which computes all $2n$ values

$$\omega = \mathrm{IFFT}(\hat{\omega}) \quad \text{where } (\mathrm{IFFT}(\hat{\omega}))_{\nu+1} := \frac{1}{2n} \sum_{\gamma=0}^{2n-1} \hat{\omega}_{\gamma+1} e^{i\gamma\nu\pi/n}. \tag{3.54}$$

Finally, the convolution by means of FFT is summarized through

$$\omega = \mathrm{IFFT}(\mathrm{FFT}(\bar{\varphi}) \odot \mathrm{FFT}(\bar{\psi})) \tag{3.55}$$

where $\odot$ denotes the componentwise multiplication of two vectors (in Step 2 above), FFT and IFFT are defined in (3.52) and (3.54), resp., and only the first n components of ω are required. These considerations result in Algorithm 3.

While Algorithm 3 is developed for equidistant grids, this can be adjusted to other grid types such as nested grids refined toward the origin and nested grids refined toward an arbitrary point. Here, the FFT can be applied due to the choice of the grid structure that consists of nested uniform portions. There also exist applications of FFT on non-uniform grids ([19], [65], [66]). In these approaches, however, smoothness of functions φ and ψ is required to reduce the approximation error of the convolution $\varphi * \psi$ and we do not guarantee this smoothness. In contrary, locally refined nested grids imply non-smoothness of these functions.
The computational cost of Algorithm 3 can be further reduced by a separable expansion of the kernel function, e.g., adaptive cross approximation (see subsection 4.1.3), with pivot points on the diagonal. Due to kernel symmetry with respect to x and y, this will lead to identical functions $\bar{\varphi}$ and $\bar{\psi}$ in lines 3 and 4 in Algorithm 3. As a result, we have to perform fast Fourier transformation only once. However,

Algorithm 3 $\omega = \text{FFT\_Convolution}(n, \varphi, \psi)$
(Convolution using fast Fourier transformations)

Input: Problem size $n \in \mathbb{N}$, vectors $\varphi, \psi \in \mathbb{R}^n$ for convolution;

Output: Convoluted sum $\omega \in \mathbb{R}^n$ $(\omega_{i+1} = \sum_{j=0}^{i} \varphi_j \psi_{i-j},\ i = 0, \ldots, n-1)$;

1: $\bar{\varphi} := (\varphi_0, \ldots, \varphi_{n-1}, 0, \ldots, 0)$; $(\bar{\varphi} \in \mathbb{R}^{2n})$
2: $\bar{\psi} := (\psi_0, \ldots, \psi_{n-1}, 0, \ldots, 0)$; $(\bar{\psi} \in \mathbb{R}^{2n})$
3: $\hat{\varphi} = \text{FFT}(\bar{\varphi})$; $(\hat{\varphi} \in \mathbb{R}^{2n}$, (3.52))
4: $\hat{\psi} = \text{FFT}(\bar{\psi})$; $(\hat{\varphi} \in \mathbb{R}^{2n}$, (3.52))
5: **for** $j = 0 : 2n - 1$ **do**
6: $\quad \hat{\omega}_{j+1} := \hat{\varphi}_j \cdot \hat{\psi}_j$; $(\hat{\omega} \in \mathbb{R}^{2n})$
7: **end for**
8: $\bar{\omega} = \text{IFFT}(\hat{\omega})$; $(\bar{\omega} \in \mathbb{R}^{2n}$, (3.54))
9: $\omega = \bar{\omega}(1 : n)$; $(\omega \in \mathbb{R}^n)$
10: **return** (ω);

this will also lead to a worse kernel approximation at points away from diagonal. This approach can be successfully applied for density distributions characterized by a single dominant peak, but yields worse results for density distributions with more than one peak.

Chapter 4

Evaluation of univariate aggregation integrals with piecewise constant functions

In this chapter we discuss the fast evaluation of aggregation source (1.3) and sink (1.5) integrals through an approximation of the density distribution with piecewise constant functions. In section 4.1 the evaluation is performed on uniform grids, while in section 4.2 on nested grids refined toward the origin.

4.1 Evaluation of univariate aggregation integrals on equidistant grids

Subsection 4.1.1 reviews the fast evaluation of the aggregation source integral through multiple use of fast Fourier transformations and presents it explicitly in algorithmic form. The definition of the grids and spaces used in this subsection are given in subsection 3.2.1. While originally developed for a discretization with piecewise constant functions, it is shown in subsection 4.1.2 how the same approach can be applied in the setting of sectional methods (section 2.2), leading to highly efficient algorithms. In section 4.1.3, we provide numerical results illustrating the performance of the proposed algorithms for different kernel functions and initial distributions.

4.1.1 Discretization through piecewise constant approximation

In this approach, both the (input) function f and the output $Q(f)$ are represented through vectors of length n and the major computational effort lies in the evaluation of a discrete convolution. An efficient approach for its evaluation through fast Fourier transformations was presented in subsection 3.3.

We require our discretizations to preserve mass. Recalling the definition (2.1), the mass, or the first moment ($k = 1$), of a density function $f(t, x)$ at time $t \in [0, T]$ is defined as

$$m(t) := m_1(t) = \int_0^1 x f(t, x)\ dx. \tag{4.1}$$

Since the total mass does not change when particles aggregate, the mass should remain constant for all $t \in [0, T]$. Hence, for discretizations $\widetilde{Q}_{\text{source}}(f) \approx Q_{\text{source}}(f)$ and $\widetilde{Q}_{\text{sink}}(f) \approx Q_{\text{sink}}(f)$ it must hold that

$$\int_0^1 x \frac{\partial f(x,t)}{\partial t}\ dx = \int_0^1 x \left(\widetilde{Q}_{\text{source}}(f)(x,t) - \widetilde{Q}_{\text{sink}}(f)(x,t)\right)\ dx = 0. \tag{4.2}$$

Remark 1. *Since mass conservation (4.2) is satisfied for the exact source and sink terms $\widetilde{Q}_{source}(f) = Q_{source}(f)$ and $\widetilde{Q}_{sink}(f) = Q_{sink}(f)$ for arbitrary kernel functions $\kappa(\cdot, \cdot)$ in (1.3) and (1.5), a (separable) approximation of $\kappa(\cdot, \cdot)$ by $\widetilde{\kappa}(\cdot, \cdot)$ as suggested in section 3.1 does not violate the mass conservation.*

The first discretization approach, which is described in more detail in [28], approximates f, $Q_{\text{source}}(f)$ and $Q_{\text{sink}}(f)$ through piecewise constant functions on a given equidistant grid as earlier introduced in subsection 3.2.1. The representing vectors contain the coefficients for their representations with respect to a given basis for the space of piecewise constant functions. Here, we will use the standard basis which consists of basis functions with support on exactly one subinterval.

The discretization process for the source integral $Q := Q_{\text{source}}$ (1.3) consists of the following steps (with details on these steps following subsequently):

1. Define the grid and space as explained in subsection 3.2.1.
2. Replace the kernel $\kappa(x, y)$ by a separable expression $\sum_{\nu=1}^{M} a_\nu(x) b_\nu(y)$, with piecewise constant $a_\nu(x)$ and $b_\nu(y)$ as in (3.7) leading to the sum of integrals

$$Q_{pl}^{\text{source}}(f)(x) = \frac{1}{2} \sum_{\nu=1}^{M} \int_0^x \varphi_\nu(y) \psi_\nu(x - y) dy \tag{4.3}$$

with (piecewise constant) functions $\varphi_\nu := b_\nu \cdot f$ and $\psi_\nu := a_\nu \cdot f$, since the density distribution $f \in \mathcal{S}$ as defined in (3.17), subsection 3.2.1. In view of Remark 1, mass is preserved in this step. ($Q_{pl}(f)$ is continuous and piecewise linear on $\mathcal{G}$ (3.16) since all integrands are piecewise constant; the subscript in Q_{pl} stands for "piecewise linear").

3. Evaluate $Q_{pl}(f)$ at the grid vertices x_{i+1}, $i \in \{0, \cdots n-1\}$ ($Q_{pl}(f)(x_1)$ is the value at the second grid point, it is already known that at the first grid point $Q_{pl}(f)(0) = 0$).

4. Project the piecewise linear function $Q_{pl}(f)$ onto $\mathcal{S}$ (3.17) to obtain the piecewise constant approximation $Q_{pc}(f) \in \mathcal{S}$ of the source integral $Q_{\text{source}}(f)$ (the subscript in Q_{pc} stands for "piecewise constant").

Step 1 is straightforward. The first part of Step 2, i.e., the separable approximation of the kernel function, has been the subject of section 3.1 and will be addressed in more detail in section 4.1.3 in the context of numerical results. If the kernel is approximated with factors $\tilde{a}_\nu$ and $\tilde{b}_\nu$, where $\tilde{a}_\nu, \tilde{b}_\nu \notin \mathcal{S}$, for a projection of kernel factors onto $\mathcal{S}$ (3.17), a simple one can be used such as

$$b_\nu(y) = \Pi_{\mathcal{S}} \tilde{b}_\nu(y) := \tilde{b}_\nu \left(\frac{x_{i+1} + x_i}{2} \right) \qquad \text{for } y \in (x_i, x_{i+1}). \tag{4.4}$$

The major computational effort of the above discretization process lies in Step 3 and will now be explained in more detail. We restrict our attention to a single integral in the sum of Step 2, and dropping the subscript ν, we focus on the evaluation of

$$Q_{pl}(f)(x_i) = \frac{1}{2} \int_0^{x_i} \varphi(y)\psi(x_i - y)dy$$

where both $\varphi(y)$ and $\psi(x_i - y)$, and therefore also their product, are piecewise constant on the equidistant grid $\mathcal{G}$ (3.16). Thus, these functions are uniquely determined through the values φ_j, ψ_j of $\varphi, \psi \in \mathcal{S}$ (3.17) on the intervals (x_j, x_{j+1}), $j = 0, \ldots, i$. Since each such interval has length $h := \frac{1}{n}$, the above integral can be evaluated as

$$Q_{pl}(f)(x_{i+1}) = \frac{h}{2} \sum_{j=0}^{i} \varphi_j \psi_{i-j} \tag{4.5}$$

for $i = 0, \ldots, n-1$. Consequently, the evaluation of this sum at the grid point x_{i+1} requires $i + 1$ multiplications and i additions, hence the direct evaluation for all grid points x_{i+1}, $i = 0, \ldots, n-1$ leads to a computational complexity $\mathcal{O}(n^2)$. In subsection 3.3 we reviewed an efficient way to evaluate these sums simultaneously

at all gridpoints x_{i+1}, $i \in \{0 \ldots, n-1\}$ in almost linear complexity $\mathcal{O}(n \log n)$ through multiple use of (inverse) fast Fourier transformations..

The projection (4.4) of Step 2 should not be used in Step 4 since it does not preserve mass, i.e., we wish to find a projection $Q_{pc}(f) := \Pi_{\mathcal{S}} Q_{pl}(f)$ so that it holds that

$$\int_0^1 x Q_{pl}(f)(x) dx = \int_0^1 x Q_{pc}(f)(x) dx.$$

For this we have to solve

$$\int_{x_i}^{x_{i+1}} x Q_{pl}(f)(x) dx = \int_{x_i}^{x_{i+1}} x Q_{pc}(f)(x) dx, \qquad i = 0, \ldots, n-1,$$

where Q_{pc} is constant on (x_i, x_{i+1}).

Since the integrand $xQ_{pl}(f)(x)$ is piecewise quadratic, such a mass-preserving projection is obtained using Simpson's rule,

$$\begin{aligned} Q_{pc}(f)(x) &:= \frac{(x_{i+\frac{1}{2}} + \frac{h}{6}) Q_{pl}(f)(x_{i+1}) + (x_{i+\frac{1}{2}} - \frac{h}{6}) Q_{pl}(f)(x_i)}{2 x_{i+\frac{1}{2}}} \\ &= \frac{(i + \frac{2}{3}) Q_{pl}(f)(x_{i+1}) + (i + \frac{1}{3}) Q_{pl}(f)(x_i)}{2i+1} \end{aligned} \tag{4.6}$$

for $x \in (x_i, x_{i+1})$ where $x_{i+\frac{1}{2}} := \frac{x_{i+1}+x_i}{2} = ih + \frac{h}{2}$ denotes the midpoint of the interval (x_i, x_{i+1}). In Algorithm 4, the projected function values are computed starting with the value on the first subinterval (x_0, x_1) up to (x_{n-1}, x_n).

Algorithm 4 $q = \text{project\_linear2constant}(n, \tilde{q})$;
(Mass-preserving projection (4.6) of a piecewise linear to a piecewise constant function.)

Input: $n \in \mathbb{N}$; (problem size)
$\tilde{q} = (\tilde{q}_0, \tilde{q}_1, \ldots, \tilde{q}_n) \in \mathbb{R}^{n+1}$; (function values at vertices $x_i = hi$ for $h = \frac{1}{n}$, $i = 0, \ldots, n$)
Output: $q = (q_0, \ldots, q_{n-1}) \in \mathbb{R}^n$; (piecewise const. values q_i on (x_i, x_{i+1}))

1: **for** $j = 0 : n-1$ **do**
2: $\quad q_j = \frac{1}{(2j+1)} \left((j + \frac{2}{3}) \cdot \tilde{q}_{j+1} + (j + \frac{1}{3}) \cdot \tilde{q}_j \right)$;
3: **end for**
4: **return** (q);

Algorithm 5 q = separableSourceAggregation(n, h, f, M, a, b);
(Evaluate the aggregation source integral (1.3) using piecewise constant approximations on an equidistant mesh.)

Input: $n \in \mathbb{N}$; (problem size, i.e., number of subintervals)
$h \in \mathbb{R}$ (grid size)
$f = (f_0, \ldots, f_{n-1}) \in \mathbb{R}^n$; (piecewise constant density function values)
$M \in \mathbb{N}$; (separation rank of kernel function, (3.10))
$a, b \in \mathbb{R}^{M \times n}$; (projection of kernel factors onto $\mathcal{S}$, (4.4))
Output: $q = (q_0, \ldots, q_{n-1}) \in \mathbb{R}^n$; (piecewise const. aggreg. $q_i \approx Q_{\text{source}}(f)|_{(x_i, x_{i+1})}$)

$\tilde{q} := (0, \ldots, 0) \in \mathbb{R}^n$;
for $\nu = 1 : M$ **do**
  for $j = 0 : n-1$ **do**
    $\psi_j = a_{\nu,j} \cdot f_j$;
    $\varphi_j = b_{\nu,j} \cdot f_j$;
  end for
  ω = FFT_Convolution(n, φ, ψ); ((4.5), see Algorithm 3 in subsection 3.3)
  $\tilde{q}$ += ω;
end for
$\tilde{q}$ *= $0.5 \cdot h$; (see (4.5))
q = project_linear2constant($n, (0, \tilde{q}_0, \ldots, \tilde{q}_{n-1})$); (Algorithm 4)
return (q);

Algorithm 5b q=directSourceAggregation(n, h, f, κ);
(Direct evaluation of the source aggregation term without use of FFT.)
Replace input M, a, b by the discrete kernel matrix $\kappa \in \mathbb{R}^{n \times n}$.

$\tilde{q} := (0, \ldots, 0) \in \mathbb{R}^n$;
for $i = 1 : n$ **do**
  for $j = 1 : i$ **do**
    $\tilde{q}_i$ += $\kappa_{i-j+1,j} \cdot f_{i-j+1} \cdot f_j$;
  end for
end for
$\tilde{q}$ *= $0.5 \cdot h$; (see (4.5))
q = project_linear2constant($n, (0, \tilde{q}_1, \ldots, \tilde{q}_n)$); (Algorithm 4)
return (q);

Algorithm 5 provides an implementation of these four steps to compute the aggregation source integral using piecewise constant approximations for the population density functions. Here, the grid size h is explicitly given as an input parameter in preparation for the general case, i.e., when we consider intervals $(a, b]$ different from $(0, 1]$ resulting in $h = \frac{b-a}{n} \neq \frac{1}{n}$. (This general case will be necessary in the subsequent Algorithm 7.) In the bottom part of Algorithm 5, following the efficient version using separable approximations and fast Fourier transformations, we also supply a direct implemention which requires two nested sums, yielding the undesirable complexity of $\mathcal{O}(n^2)$. In section 4.1.3, we will provide numerical tests comparing these two versions.

The discretization of the aggregation sink term (1.5) consists of the following steps:

1. Define the grid and space as explained in subsection 3.2.1.
2. Replace the kernel $\kappa(x, y)$ by the same separable expression as in (3.8) leading to

$$\begin{aligned} Q_{pl}^{\text{sink}}(f)(x) &= \sum_{\nu=1}^{M} \psi_\nu(x) \int_0^{1-x} \varphi_\nu(y) dy \\ &= \sum_{\nu=1}^{M} \psi_\nu(x) \bar{\varphi}_\nu(x) \qquad \text{with } \bar{\varphi}_\nu(x) := \int_0^{1-x} \varphi_\nu(y) dy \end{aligned}$$

   with piecewise constant $\psi_\nu := a_\nu f$ and $\varphi_\nu := b_\nu f$. The resulting $Q_{pl}(f)(x)$ is piecewise linear but in general discontinuous ($\psi_\nu(x), \varphi_\nu(y)$ are piecewise constant, $\bar{\varphi}_\nu(x)$ is continuous piecewise linear, hence $\psi_\nu(x)\bar{\varphi}_\nu(x)$ is piecewise linear, but in general discontinuous.)
3. Use the mass-preserving projection (4.6) to project (the continuous piecewise linear) $\bar{\varphi}_\nu(x)$ and hence $Q_{pl}(f)$ onto $\mathcal{S}$ to obtain the piecewise constant approximation $Q_{pc}(f) \in \mathcal{S}$ of $Q_{\text{sink}}(f)$.

The resulting algorithm is provided as Algorithm 6. It includes (at the bottom) an implementation of a direct evaluation of the sink term that does not exploit a separable expansion of the kernel function, leading to an $\mathcal{O}(n^2)$ complexity. In the direct evaluation, we assume that the kernel function κ is replaced by a piecewise constant approximation, i.e., $\kappa(x, y) = \kappa_{i,j} \in \mathbb{R}$ for all $(x, y) \in (x_{i-1}, x_i) \times (x_{j-1}, x_j)$. This leads to the piecewise linear but in general *discontinuous* sink term approximation $Q_{pl}(f)(x) = f(x) \int_0^{1-x} \kappa(x, y) f(y)\, dy$. In order to project it to a piecewise constant approximation $Q_{pc}(f)$, one needs to compute both left-

and right-hand limits $q^-(x_i) = \lim_{x \to x_i^-} Q_{pl}(f)(x)$ and $q^+(x_i) = \lim_{x \to x_i^+} Q_{pl}(f)(x)$ at the gridpoints x_i. In lines 5. and 7. of Algorithm 6b, we compute these limits using the notation $q_i^{\text{left}} = q^+(x_{i-1})$ (right-hand limit at left endpoint of the i'th interval (x_{i-1}, x_i)) and $q_i^{\text{right}} = q^-(x_i)$ (left-hand limit at right endpoint of the i'th interval (x_{i-1}, x_i)). In line 8., we then use a mass-preserving projection to project the function that is linear on $[x_{i-1}, x_i]$, given by its values at the endpoints q_i^{left} and q_i^{right}, to a constant.

4.1.2 Sectional method with FFT

In this section we will apply the approach based on kernel separation and FFT as in 4.1.1 to reduce the computational cost of the sectional approach introduced in the section 2.2.
In order to reduce the computational work necessary to evaluate (2.3) for all $i = 1, \ldots, n$, we pursue an approach analogous to the one described in the previous subsection 4.1.1, i.e., we replace the aggregation kernel $\kappa(x, y)$ by a separable approximation $\sum_{\nu=1}^{M} a_\nu(x) b_\nu(y)$ and define the discrete factors $\kappa_{i,j} := \sum_{\nu=1}^{M} a_\nu(x_i) b_\nu(y_j)$ to pave the way for fast Fourier transformation. Using the abbreviations $a_{\nu,i} := a_\nu(x_i)$, $b_{\nu,j} := b_\nu(x_j)$ and substituting the separable kernel approximation into (2.3) leads to

$$Q_i = \frac{1}{2} \sum_{\nu=1}^{M} \sum_{j=1}^{i-1} (a_{\nu,j} \mathcal{N}_j)(b_{\nu,i-j} \mathcal{N}_{i-j}) - \mathcal{N}_i \sum_{\nu=1}^{M} a_{\nu,i} \sum_{j=0}^{n-i} b_{\nu,j} \mathcal{N}_j, \qquad i = 1, \ldots, n. \quad (4.7)$$

If we focus on the first inner sum, drop the index ν and set $\varphi_j := a_j \mathcal{N}_j$, $\psi_{i-j} := b_{i-j} \mathcal{N}_{i-j}$, this sum reads as

$$\sum_{j=1}^{i} \varphi_j \psi_{i-j} \quad (4.8)$$

which we recognize to be equivalent (up to an index shift and scaling by $\frac{h}{2}$) to the convolution (4.5) obtained in the discretization with piecewise constant functions. Once again its efficient evaluation can be performed by the technique introduced in the section 3.3.

Due to its simplicity (and widespread use), we do not provide a separate algorithmic representation of this macroscopic discretization method.

We conclude this subsection with a remark on the similarities and differences of the two discretization methods:

Algorithm 6 q = separableSinkAggregation(n, h, f, M, a, b);
(Evaluate the aggregation sink integral (1.5) using piecewise constant approximation and separable kernel approximation on an equidistant mesh.)

Input:	$n \in \mathbb{N}$;	(problem size, i.e., number of subintervals)
	$h \in \mathbb{R}$	(grid size)
	$f = (f_0, \ldots, f_{n-1}) \in \mathbb{R}^n$;	(piecewise constant density function values)
	$M \in \mathbb{N}$;	(separation rank of kernel function, (3.10))
	$a, b \in \mathbb{R}^{M \times n}$;	(projection of kernel factors onto $\mathcal{S}$, (4.4))
Output:	$q = (q_0, \ldots, q_{n-1}) \in \mathbb{R}^n$;	(piecewise const. aggreg. $q_i \approx Q_{\text{sink}}(f)\|_{(x_i, x_{i+1})}$)

```
1: q := (0, ..., 0);
2: for ν = 1 : M do
3:    φ̄_n = 0;
4:    for j = 1 : n do
5:       φ̄_{n-j} = φ̄_{n-j+1} + h b_{ν,j} · f_j;        (φ̄_{n-j} = ∫_0^{1-x_{n-j}} φ_ν(y) dy)
6:    end for
7:    φ̄ := project_linear2constant(n, φ̄);
8:    for j = 0 : n − 1 do
9:       q_j += a_{ν,j} · f_j · φ̄_j;
10:   end for
11: end for
12: return (q);
```

Algorithm 6b q=directSinkAggregation(n, h, f, κ);
(Direct evaluation of the sink aggregation term without a separable kernel approximation.) Replace input M, a, b by the discrete kernel matrix $\kappa \in \mathbb{R}^{n \times n}$.

```
1: q̃ := (0, ..., 0);
2: q̃^right := (0, ..., 0);
3: for i = 1 : n do
4:    for j = 1 : (n − i) do
5:       q̃_i^right += κ_{i,j} · f_j;
6:    end for
7:    q̃_i^left = q̃_i^right + κ_{i,n-i+1} · f_{n-i+1};
8:    q_i = 1/(2i+1) · ( (i + 2/3) · q̃_i^right + (i + 1/3) · q̃_i^left );
9: end for
10: for i = 1 : n do
11:    q_i *= h;
12: end for
13: return (q);
```

Remark 2. *Using piecewise constant approximation (subsection 4.1.1), the particle density is given in the form of a vector* $(f_0, \ldots, f_{n-1})$ *where* f_i *denotes the (constant) density in the interval* (x_i, x_{i+1})*. In the convolution sum (4.5), one initially computes the particle densities* $\tilde{q}(x_i)$ *at the gridpoints* x_i *(which are subsequently used for a projection to a constant* $q(x_i)$ *on the interval* (x_i, x_{i+1})*). In order to contribute to* $\tilde{q}(x_i)$*, a particle in the interval* (x_j, x_{j+1}) *(accounted for through* f_j*) must aggregate with a particle in the interval* (x_{i-j}, x_{i-j+1}) *(accounted for through* f_{i-j}*) which justifies the subscripts in the sum (4.5).*

In the sectional method in section 2.2, the particle density is given in the form of a vector $(\mathcal{N}_1, \ldots, \mathcal{N}_n)$ *where* $\mathcal{N}_i$ *denotes the number of particles of size* i*. Now particles of size* j *(accounted for in* $\mathcal{N}_j$*) and size* $i - j$ *(accounted for in* $\mathcal{N}_{i-j}$*) aggregate to a particle of size* i *(accounted for in* $\mathcal{N}_i$*), leading to the subscripts in the sum (4.8).*

Thus, the different indices in the sums (4.5) and (4.8) result from the interpretation of density values f_i *as constants over an interval* (x_i, x_{i+1}) *versus point values* $\mathcal{N}_i$ *at a gridpoint* x_i*.*

4.1.3 Numerical results

In this subsection, we provide numerical results in order to illustrate the performance of the proposed algorithms. The section is divided into the following three parts, dealing with test problems of three different types:

- In the first part we provide numerical results for the Brownian kernel (3.2) function κ_B which is separable with separation rank 3 (see 3.9). We will provide timings to compare the evaluations of the aggregation source term using either the piecewise constant discretization of subsection 4.1.1 or the sectional method of subsection 4.1.2. Both methods can be implemented either directly (leading to double sums and complexity $\mathcal{O}(n^2)$) or with fast Fourier transformations, leading to complexity $\mathcal{O}(n \log n)$. We will also provide timings for the evaluation of the sink term and compare the solutions obtained with the two methods (discretization with piecewise constant functions versus sectional method).
- In the second part we provide numerical test results for the shear kernel function κ_S (3.3) which is not separable but well suited for a separable approximation since the approximation error decays exponentially with increasing rank. We will show how an increase in rank leads to an increase in computational time but also an increase in the accuracy of the computed solution. To obtain separable approximations of the kernel function we will

use and compare two approaches, through Chebyshev polynomials and using the adaptive cross approximation method (ACA). In this subsection, we will also comment on the kinetic kernel κ_K which displays a behaviour similar to the shear kernel with respect to its separable approximability.

- In the last part we provide numerical test results for the gravitational kernel function κ_G (3.4) which is neither separable nor suited for a global separable approximation. However, this kernel function is separable (with rank 4) on any (connected) subdomain of $(0,1] \times (0,1]$ that does not contain any point of the diagonal $\{(x,x) \mid x \in (0,1]\}$. This observation allows us to use the efficient implementations of the aggregation source and sink terms on hierarchically structured subdomains, leading to the complexity $\mathcal{O}(n(\log n)^2)$, which will be explained in more detail and illustrated with numerical results in this subsection.

Numerical results for the Brownian kernel

Here, we show results for the separable Brownian kernel (3.2) of separation rank 3 (see 3.9). In Table 4.1, we show the times (in seconds) consumed by the numerical computation of the source and sink integrals of the aggregation term. The comparison includes the discretization with piecewise constant functions (subsection 4.1.1) and the sectional method (subsection 4.1.2). In either case, the source term can be either computed directly, i.e., with the explicit evaluation of double sums, or with the separable approach and convolution through fast Fourier transformations. Both implementations are provided in Algorithm 5. We draw the following

Degrees of freedom n	128	256	512	1024	2048	4096
	Source term					
Sectional, no FFT	0.32	1.26	5.93	29.5	138	667
Piecew. const., no FFT	0.34	1.3	5.62	29	140	651
Sectional, FFT	2.78	5.8	14.1	27	59.3	125
Piecew. const., FFT	2.8	6	12.8	27.5	60	129
$2.7 \cdot 10^{-3} n \log_2 n$	2.76	5.53	12.4	27.6	60	132
	Sink term					
Sectional	0.29	0.52	0.98	1.9	3.7	7.4
Piecew. const.	0.27	0.48	0.94	1.87	3.6	7.5

Table 4.1: Times (in seconds) to compute the source and sink terms for 5000 time steps ($dt = 0.001, T = 0.1$).

conclusions from Table 4.1:

- The computational complexity using a discretization by piecewise constant functions is about the same as the complexity of the sectional method.
- The times to compute the source term without using FFT grow quadratically in the problem size.
- The times to compute the source term with FFT grow approximately as $2.7 \cdot 10^{-3} n \log_2 n$, i.e., almost linearly in the problem size.
- The times to compute the sink term, using the separable representation, grow linearly in the problem size and amount to less than 10% of the times for the source term (when using FFT), i.e., the computation of the source term dominates the overall computational time.

We next show the solutions that are computed for the initial distributions $f_1(0, x)$ and $f_4(0, x)$ (Figure 3.2). In particular, we show the distribution $f_1(t, x)$ after 4000 (left, top) and 15000 (right, top) timesteps and the distribution $f_4(t, x)$ after 500 (left, bottom) and 1000 (right, bottom) timesteps using the stepwidth $dt = 0.001$. We see that the initial peak develops into additional peaks which, roughly spoken, result from particles of the first peak aggregating to particles of twice the size, forming a second peak, and then particles of these two peaks aggregating to form a third peak, etc.

Figure 4.2 shows the evolution of the density distribution with initial distribution $f_1(0, x)$ over the time interval $t \in [0, 15]$, showing the decrease of the initial peak at $x = 0.125$ and the formation of additional peaks around $x = 0.25$ and later also around $x = 0.375$ and $x = 0.5$.

As a final test involving the Brownian kernel, we compare the results obtained from the two different discretization techniques, using piecewise constant functions or the sectional method. In Figure 4.3, we see that the computed results differ only slightly for a coarse discretization with $n = 40$ and approach each other with increasing n, here illustrated for $n = 100$. For the difference between these two methods, we also refer to the discussion in Remark 2.

Numerical results for the shear kernel

In this part, we show results for the shear kernel κ_S (3.3). In Figure 4.4 (left), we show the singular values of the discrete shear kernel matrix $K = (\kappa_S(x_i, y_j))_{i,j=0}^{n-1}$ for $n = 50$. The singular values of the shear kernel decay quickly and reach machine precision within the first 9 singular values so that this kernel will be suitable for a low-rank approximation.

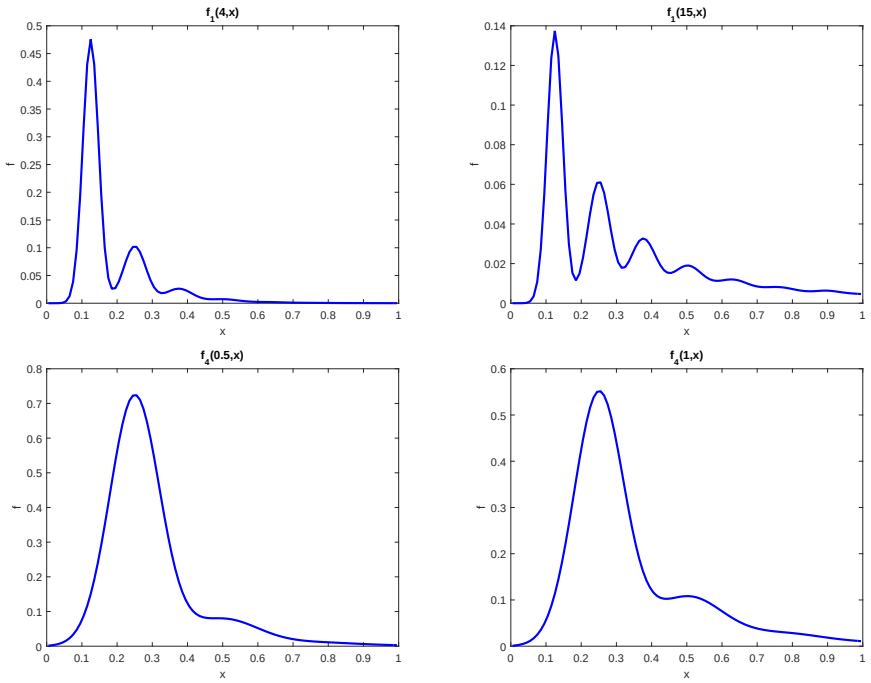


Figure 4.1: Top: Density distributions $f_1(4,x)$ (left), and $f_1(15,x)$ (right) after 4000 and 15000 time steps, respectively with $dt = 0.001$, n=100. Bottom: Density distributions $f_4(0.5,x)$ (left), and $f_4(1,x)$ (right) after 500 and 1000 time steps, respectively with $dt = 0.001$, n=100.

We will discuss two approaches to obtain a separable approximation of the shear kernel κ_{S}. For the first approach we use (Chebyshev) polynomials to obtain a separable (tensor product) approximation of the kernel κ_{S}. Using T_i and T_j to denote the i-th and j-th Chebyshev polynomials, resp. (or, in fact, any basis of the space of polynomials of degree $< M$), we determine coefficients $c_{i,j}$ to interpolate the kernel function κ_{S} at the Chebyshev nodes, i.e.

$$\begin{aligned}\kappa_S(x,y) &= \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} c_{i,j} T_i(x) T_j(y) \\ &= \begin{pmatrix} T_0 & \cdots & T_{M-1}\end{pmatrix}(x) \underbrace{\begin{pmatrix} c_{00} & \cdots & c_{0M} \\ \vdots & \ddots & \vdots \\ c_{M0} & \cdots & c_{MM}\end{pmatrix}}_{=:C} \begin{pmatrix} T_0 \\ \vdots \\ T_{M-1}\end{pmatrix}(y) \qquad (4.9)\end{aligned}$$

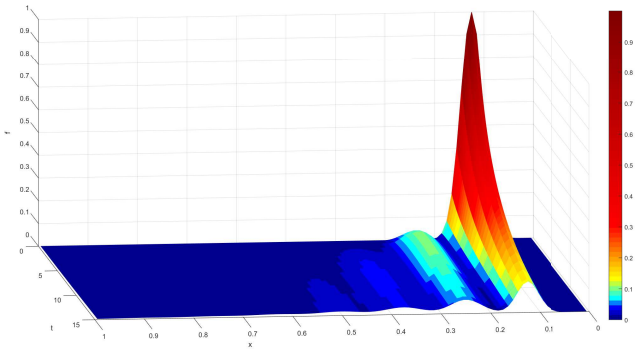


Figure 4.2: Evolution of the density distribution f_1 over time ($dt = 0.001$, 15000 time steps) for the initial distribution $f_1(0, x)$, n=100.

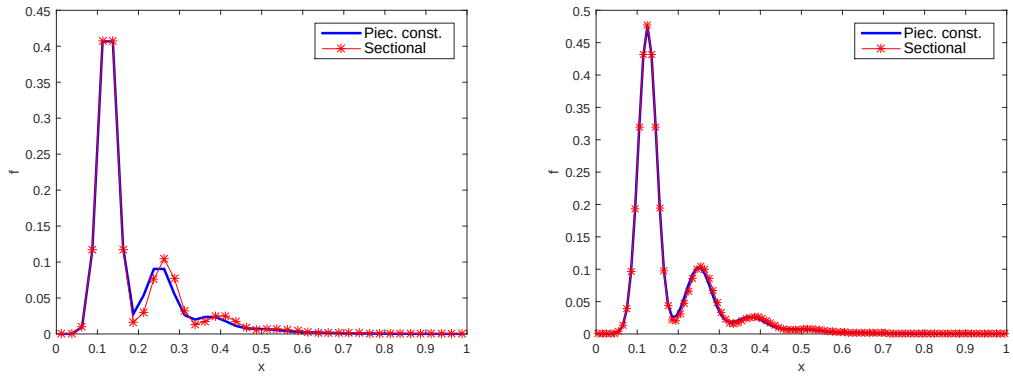


Figure 4.3: Density distributions for the piecewise constant and sectional methods after 4000 time steps, $dt = 0.001$ for the initial distribution $f_1(0, x)$, $n = 40$ (left) and $n = 100$ (right).

$$\forall\ (x, y) \in \left\{ \left(\frac{1}{2} + \frac{1}{2} \cos \frac{(2k-1)\pi}{2M}, \frac{1}{2} + \frac{1}{2} \cos \frac{(2\ell-1)\pi}{2M} \right) \mid 1 \leq k, \ell \leq M \right\}. \quad (4.10)$$

The coefficient matrix C is symmetric positive definite (since $\kappa_S(x, y) > 0$ and $\kappa_S(x, y) = \kappa_S(y, x)$), hence it is diagonalizable, i.e, there exist an orthogonal matrix Q and a diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ with $\lambda_i \geq \lambda_j > 0$ for $i < j$ such that $C = Q\Lambda Q^T$. This allows one to reduce the double sum (4.9) into a single sum in

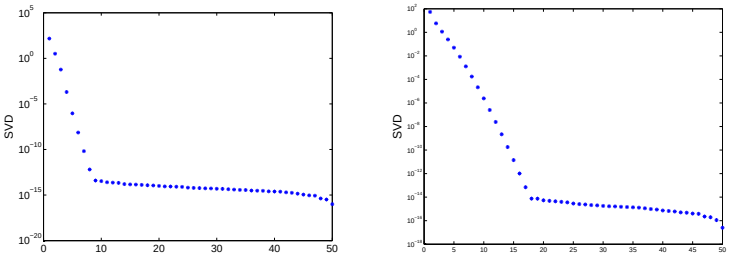


Figure 4.4: Singular values of discrete shear kernel matrix $K_S \in \mathbb{R}^{50\times 50}$ (left) and the discrete kinetic kernel matrix $K_K \in \mathbb{R}^{50\times 50}$ (right).

view of

$$\begin{aligned}\underbrace{\begin{pmatrix} T_0 & \cdots & T_{M-1}\end{pmatrix}}_{=:T}(x)Q^T\Lambda Q\begin{pmatrix} T_0 \\ \vdots \\ T_{M-1}\end{pmatrix}(y) &= (QT)^T(x)\Lambda\underbrace{(QT)(y)}_{=:\tilde{T}(y)} \\ &= \tilde{T}(x)^T\Lambda\tilde{T}(y) = \sum_{i=0}^{M-1}\lambda_i\tilde{T}_i(x)\tilde{T}_i(y)\end{aligned}$$

for polynomials $\tilde{T}_i$ of degree at most $M-1$. We can now approximate the kernel κ_S with increasing accuracy as we increase the rank M of the separable approximation. In Figure 4.5 (left), we show the solutions obtained for various ranks.
The second approach we discuss for obtaining a separable kernel approximation is the adaptive cross approximation method (ACA) [33], [6]. The separable cross approximation for a continuous, bivariate function $\kappa(x,y)$ is constructed recursively by setting

$$E_0(x,y) = \kappa(x,y),$$

and defining error functions

$$E_i(x,y) = E_{i-1}(x,y) - \frac{E_{i-1}(x_i,y)\,E_{i-1}(x,y_i)}{E_{i-1}(x_i,y_i)}$$

with interpolation points (x_i,y_i), $1 \le i \le M$ for a rank M-approximation. We choose the interpolation points (x_i,y_i) to be (scaled/shifted) Chebyshev nodes (4.10), meaning there are $\sqrt{M}$ nodes in each dimension. Then the separable cross approximation is defined as

$$\kappa^{(M)}(x,y) = \sum_{i=1}^{M}\frac{E_{i-1}(x_i,y)\,E_{i-1}(x,y_i)}{E_{i-1}(x_i,y_i)}.$$

The source aggregation integral can then be approximated as

$$
\begin{aligned}
& \frac{1}{2}\int_0^x \kappa^{(M)}(x-y,y) f(x-y) f(y)\ dy \\
= & \sum_{i=1}^{M} \frac{1}{2}\int_0^x \frac{E_{i-1}(x-y,y_i)E_{i-1}(x_i,y)}{E_{i-1}(x_i,y_i)} f(x-y) f(y)\ dy \\
= & \sum_{i=1}^{M} \frac{1}{2}\int_0^x \underbrace{\frac{E_{i-1}(x-y,y_i) f(x-y)}{E_{i-1}(x_i,y_i)}}_{:=\varphi(x-y)} \underbrace{E_{i-1}(x_i,y) f(y)}_{:=\psi(y)}\ dy \\
= & \sum_{i=1}^{M} \frac{1}{2}\int_0^x \varphi(x-y)\ \psi(y)\ dy.
\end{aligned}
$$

Figure 4.5 (right) shows the solutions obtained using adaptive cross approximation for rank = 1. For rank > 1, the computed results are improving in accuracy but are (almost) indistinguishable from the solution for the exact kernel in the resolution used in this figure, illustrating that highly accurate solutions are obtained for separable approximations of very low ranks. According to Figure 4.4 (left) we need a high separation rank to achieve an accurate approximation of the kernel. Despite this we reach highly accurate result for the source aggregation integral already for rank = 1. The reason is that rank = 1 computes the behavior of the shear kernel K_S near 0.25 (see figure 3.1, second), where the peak of the density distribution f_4 is placed.

An increase in rank leads to an increase in computational time, and Table 4.2 shows the times necessary to compute 2500 time steps for various choices of ranks. The computational time is the same for both discussed methods (Chebyshev and ACA). The approach of separable approximations and FFT with rank = 1 is superior (with respect to time) for the problem size $n \geq 512$, with rank = 2 for $n \geq 1024$ and finally with rank ≥ 4 for $n \geq 2048$. We recall that timings in Table 4.1 were obtained for a rank-3 representation of the Brownian kernel where the separable representation has also been exploited in the computation of the sink term. Here in Table 4.2, the timing includes the time to compute the sink term which has quadratic complexity if no separable approximation is available for its computation (see the bottom part of Algorithm 6).

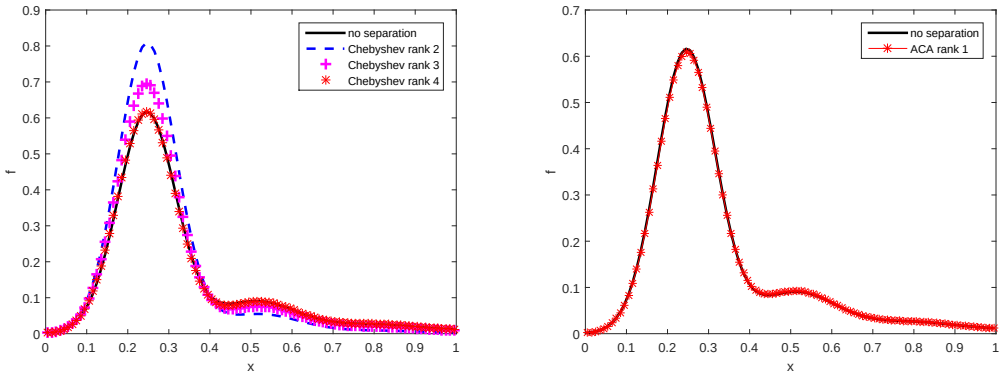


Figure 4.5: Density distribution after 1800 time steps ($dt = 0.001$) with the shear kernel, initial density distribution $f_4(0, x)$, n=100 using Chebyshev polynomials (left) and adaptive cross approximation (right).

Kinetic kernel

Here, we briefly discuss the kinetic kernel κ_k (3.5). A plot of the singular values of the discrete kinetic kernel function is given in Figure 4.4 (right). Similar to the shear kernel, the singular values decay exponentially, which implies that the kinetic kernel can also be successfully approximated through a separable approximation, similar to the shear kernel. This time, however, it takes rank 16 before machine precision is reached.

The kinetic kernel allows the following interesting observation: If the aggregation integral (1.3) is evaluated for a fixed x, the kernel function $\kappa(\cdot, \cdot)$ needs to be evaluated only for arguments with constant argument sum $(x-y)+y = x$. On the discrete side, this turns into summations (4.5) with the kernel κ now included in φ, ψ but again only being required for a constant argument sum $x+y = m = const.$ In the particular case of the kinetic kernel, this constant can be factored out,

$$\begin{aligned} \kappa_{\mathrm{K}}(x,y) &= m^{-\frac{3}{2}} \underbrace{(x^{\frac{1}{3}} + y^{\frac{1}{3}})^2 \cdot (xy)^{\frac{1}{2}}}_{=:\kappa_{\mathrm{pK}}(x,y)} \\ &=: m^{-\frac{3}{2}}\, \kappa_{\mathrm{pK}}(x,y) \quad \forall (x,y) \in \mathbb{R}^2 \text{ s.t. } x+y=m, \end{aligned}$$

which yields a separable "partial kinetic" kernel $\kappa_{\mathrm{pK}}(x,y) = x^{\frac{7}{6}}y^{\frac{1}{2}} + 2x^{\frac{5}{6}}y^{\frac{5}{6}} + x^{\frac{1}{2}}y^{\frac{7}{6}}$ of separation rank 3.

While this approach leads to an efficient computation (i.e., separation and FFT) of the source term of the aggregation, it cannot be applied to the computation of the sink term. Using a direct computation (as in the bottom of Algorithm 6) leads

Degrees of freedom	128	256	512	1024	2048	4096
	No separation					
No FFT	0.32	1.1	4.2	19.1	86.2	418
	Separation by ranks (Chebyshev and ACA)					
rank 1, FFT	0.66	1.27	2.6	5.4	11.7	25.3
rank 2, FFT	1.26	2.5	5.4	11.23	24	57
rank 4, FFT	2.3	5	12.5	22	47	102
rank 6, FFT	3.5	7.4	16.9	33	71.5	162

Table 4.2: Computational times (in seconds) for 2500 time steps (time stepping $dt = 0.001$) for the solution of (1.1) (including source and sink terms) for the shear kernel.

to quadratic complexity for the sink term, while using a (separable) approximation would now be different from the (exact) kernel used in the source term which would lead to a loss of mass conservation. Hence, this efficient approach for the source term does not lead to an overall efficient computation of both the source and sink aggregation terms unless a respectively elegant approach can be found for the sink term as well.

Numerical results for the gravitational kernel

In this part, we show numerical results for the gravitational kernel $\kappa_{\mathrm{G}}(x, y)$ (3.4). In Figure 4.6 (left), we show the singular values of the discrete gravitational kernel matrix $K_G = (\kappa_G(x_i, y_j))_{i,j=0}^{n-1}$ for $n = 50$.

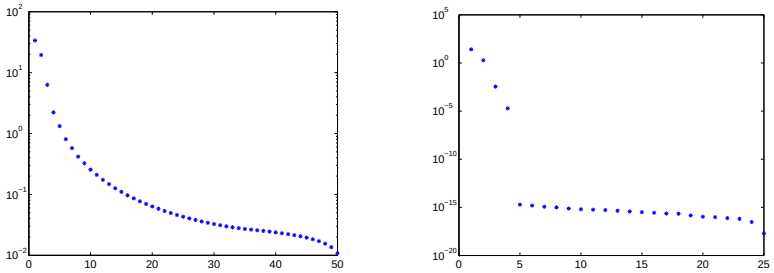


Figure 4.6: Singular values of the discrete gravitational kernel matrix $K_G \in \mathbb{R}^{50\times 50}$ (left) and its off-diagonal block $K_{\mathrm{G,off\text{-}diag}}$ (right).

Only the first few singular values decay exponentially so that a low-rank approximation is not promising, at least not when applied to the entire matrix. The situation changes if we consider the singular values of the off-diagonal block

$K_{\text{G,off-diag}} := (\kappa(x_i, y_j))_{\substack{0 \le i \le n/2-1 \\ n/2 \le j \le n-1}}$ of the discrete kernel Matrix K_G, displayed in Figure 4.6 (right), which corresponds to matrix entries for $(x_i, x_j) \in (0, \frac{1}{2}] \times (\frac{1}{2}, 1]$. Since $\kappa_{\text{G}}(x, y)$ can be written without using absolute values as

$$\kappa_{\text{G}}(x,y) = \begin{cases} (x^{\frac{1}{3}} + y^{\frac{1}{3}})^2 (x^{\frac{2}{3}} - y^{\frac{2}{3}}) & : x \ge y \\ (x^{\frac{1}{3}} + y^{\frac{1}{3}})^2 (-x^{\frac{2}{3}} + y^{\frac{2}{3}}) & : x < y \end{cases},$$

it follows that κ_G is of separation rank 4 when restricted to $x < y$ (or alternatively to $x \ge y$), explaining the singular values of the matrix block $K_{\text{G,off-diag}}$. The same applies to matrix blocks corresponding to entries (x_i, y_j) with $x_i > y_j$.

Hence, we propose to subdivide the domain $(0, 1] \times (0, 1]$ into the four subdomains $A,\ B,\ C,\ D$ as illustrated in Figure 4.7.

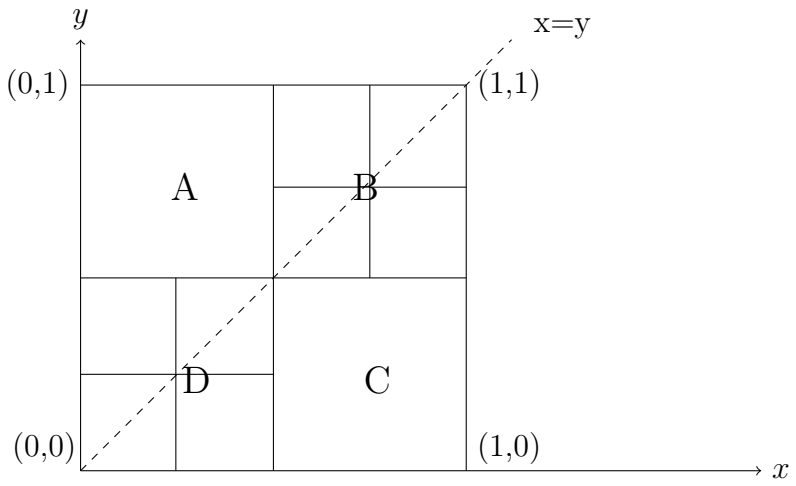


Figure 4.7: (Recursive) subdivision of the domain into four subdomains.

Restricted to the blocks A or C, the kernel function is separable with rank 4, hence we can use the efficient FFT approach for each of these blocks. In fact, since we compute the convolution of a function f with itself, the results for A and C will be identical and have to be computed only once (line 8. of Algorithm 7). Since the block B corresponds to particles of size bigger than the maximum size 1 ($x + y > 1$), the density distribution $f(x, y)$ vanishes on this block and no computation is necessary. For the remaining diagonal block D, we repeat this subdivision approach into four subblocks, now both subblocks on the diagonal are treated recursively (until we reach a problem size which is small enough for a direct computation of the aggregation term without FFT, line 25. of Algorithm 7) and one of the two offdiagonal blocks computed by the efficient separation and FFT approach (line 16. of Algorithm 7).

Algorithm 7 q = locallySeparableSourceAggregation(n, f, M, a, b, L, κ); (Evaluate the source integral (1.3) for piecewise constant approximations on an equidistant mesh, rank-M separable kernel representations in offdiagonal blocks.)

Input: $n \in \mathbb{N}$; (problem size, i.e., number of subintervals)
$f = (f_0, \ldots, f_{n-1}) \in \mathbb{R}^n$; (piecewise constant density function values)
$M \in \mathbb{N}$; (kernel separation rank in offdiagonal blocks)
$a, b \in \mathbb{R}^{M \times n}$; (projection of kernel factors onto $\mathcal{S}$, (4.4))
$\eta \in \mathbb{N}$; (number of subdivisions)
$\kappa \in \mathbb{R}^{(n2^{-\eta}) \times (n2^{-\eta}) \times (2^{\eta-1})}$; ($\kappa^{i \times j \times \ell}$ is the (i,j) component of the l-th diagonal block of the discrete kernel matrix)
Output: $q \in \mathbb{R}^n$; (piecewise const. aggreg. $q_i \approx Q_{\text{source}}(f)|_{(x_{i-1}, x_i)}$)

$q := (0, \ldots, 0)$;
$h := 1.0/n$;
$\tilde{f} := (f_0, \ldots, f_{n/2-1})$;
$\tilde{g} := (f_{n/2}, \ldots, f_{n-1})$;
$\tilde{a} := a[0 : \frac{n}{2} - 1, :]$; ($\tilde{a}$ is obtained for $x \leq y$);
$\tilde{b} := b[\frac{n}{2} : n - 1, :]$; ($\tilde{b}$ is obtained for $x \leq y$) ;
$\zeta := n/2$;
$(q_{n/2}, \ldots, q_{n-1})$ += 2 mod_separableSourceAggregation($n/2, h, \tilde{f}, \tilde{g}, M, \tilde{a}, \tilde{b}, \zeta$);
for $l = 1 : \eta - 1$ **do**
  $m := n2^{-l}$;
  for $i = 0 : 2^{l-1} - 1$ **do**
    $\tilde{a} := a[im : (i+1)m - 1, :]$; $\tilde{a}[\frac{m}{2} : m - 1, :] = 0$;
    $\tilde{b} := b[im : (i+1)m - 1, :]$; $\tilde{b}[0 : \frac{m}{2} - 1, :] = 0$;
    $\tilde{f} := (f_{im}, \ldots, f_{(i+1)m-1})$;
    $\zeta := 2im$;
    $(q_{2im}, \ldots, q_{2(i+1)m-1})$
      += 2 mod_separableSourceAggregation($m, h, \tilde{f}, \tilde{f}, M, \tilde{a}, \tilde{b}, \zeta$);
  end for
end for
$m = n2^{-\eta}$;
$\tilde{\kappa}$=zeros$(2m, 2m) \in \mathbb{R}^{2m \times 2m}$; (zero matrix)
for $i = 0 : 2^{\eta-1} - 1$ **do**
  $\tilde{f} := (f_{im}, \ldots, f_{(i+1)m-1}, 0, \ldots, 0)$;
  $\tilde{\kappa}(0 : m - 1, 0 : m - 1) := \kappa(:, :, i + 1)$;
  $\zeta = 2im$;
  $(q_{2im}, \ldots, q_{2(i+1)m-1})$ += mod_directSourceAggregation($2m, h, \tilde{f}, \tilde{\kappa}, \zeta$);
end for
return (q);

Here, we need to use slightly modified versions of the separable and direct evaluations of the source aggregation integral (Algorithm 5), denoted by

$$\textbf{mod\_separableSourceAggregation}(n, h, f, g, M, a, b, \zeta) \text{ and}$$
$$\textbf{mod\_directSourceAggregation}(n, h, f, \kappa, \zeta), \text{ resp.},$$

with the following three changes (which also require straightforward modifications in the Algorithms 4 and 3):

- In Algorithm 5, we had assumed a maximum particle size 1, and as a result, any convolution results for larger particles which had been computed when using the FFT-convolution had been discarded (see line 9 of Algorithm 3 where the convolution vector of length $2n$ is truncated back to length n.) In the evaluation for the offdiagonal subblocks of D (as well as any further offdiagonal blocks within the recursion) we need to retain these values (which amounts to no additional computational effort since these values had already been computed; they simply must not be discarded).

- There are now two density input vectors f, g instead of only f, and line 5 of Algorithm 5 needs to be replaced by $\varphi_j = b_{\nu,j} \cdot g_j$.

- The integer parameter ζ denotes an interval shift which is required in the mass-preserving projection of a piecewise linear to a piecewise constant function: In Algorithm 4, replace line 2 by

$$c_j = \frac{1}{(2(j+\zeta)+1)} \left((j+\zeta+\frac{2}{3}) \cdot q_{j+1} + (j+\zeta+\frac{1}{3}) \cdot q_j \right).$$

The resulting algorithm for computing the source term using a blockwise local separation of the kernel function is given in Algorithm 7. The application of this separation to the sink term for its efficient evaluation can be done with analogous modifications.

We estimate the complexity of this recursive approach as follows: Let $C(n)$ denote the cost to evaluate the aggregation term for the gravitational kernel for a problem of size n (in all four subdomains A, B, C, and D, even though this is only necessary on the recursive calls), and let k denote the constant in the complexity estimate $kn \log n$ to evaluate a low-rank kernel aggregation term using FFT for a problem of size n.

Assuming a problem size $n = 2^\theta$ and $1 \leq \eta \leq \theta - 1$ recursive subdivisions we obtain

$$
\begin{aligned}
C(n) &= \underbrace{\mathcal{O}\Big(k\frac{n}{2}\log\frac{n}{2}\Big)}_{\text{solve in subdomain } A(C)} + \underbrace{2C\left(\frac{n}{2}\right)}_{\text{recursive call in } B,D} \\
&= \mathcal{O}\left(k\frac{n}{2}\log\frac{n}{2} + k\frac{n}{2}\log\frac{n}{4} + k\frac{n}{2}\log\frac{n}{8} + ... + k\frac{n}{2}\log\frac{n}{2^{\eta-1}}\right) + 2^{\eta}\ C(\frac{n}{2^{\eta}}) \\
&= \mathcal{O}\left(k\frac{n}{2}\left(\log\frac{n^{\eta-1}}{2\cdot 4\cdot 8\cdot \ldots \cdot 2^{\eta-1}}\right)\right) + 2^{\eta}\ C(\frac{n}{2^{\eta}}) \\
&\leq \mathcal{O}\left(k\frac{n}{2}\left(\log\frac{n^{\theta-1}}{2\cdot 4\cdot 8\cdot \ldots \cdot 2^{\theta-1}}\right)\right) + \mathcal{O}(n) \\
&= \mathcal{O}\left(k\frac{n}{2}\left(\log n^{\theta}\right) + n\right) = \mathcal{O}\left(k\frac{n}{2}\theta\log n + n\right) = \mathcal{O}\left(k\frac{n}{2}(\log n)^2 + n\right).
\end{aligned}
$$

Hence, the problem can be solved with the complexity $\mathcal{O}(kn(\log n)^2)$.

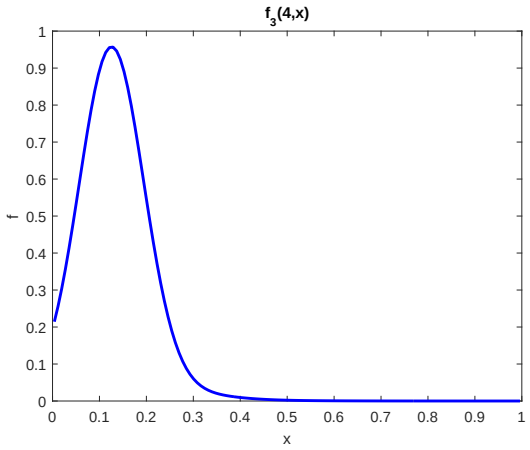


Figure 4.8: Density distribution $f_3(4, x)$ using initial density distribution $f_3(0, x)$ for $dt = 0.001$ after 4000 time steps and n=128.

Figure 4.8 illustrates numerical results for the gravitational kernel for the initial distribution $f_3(0, x)$. Here, there is almost no visible change between $f_3(0, x)$ and $f_3(4, x)$, since for the gravitational kernel, particles of equal size do not aggregate, and this initial density distribution has most particles of size $x \approx 0.125$.

In Table 4.3, we show the computational times (in seconds, only computing the source term) to compute 4000 time steps for the gravitational kernel. Similar to the timings in Table 4.1 for the Brownian kernel, the complexity is quadratic in

the problem size when the direct evaluation without kernel separation and FFT is used. Using the recursive approach with an efficient (FFT) evaluation of the source term in the offdiagonal blocks, we present results for various depths of recursion. The left column in Table 4.3 denotes the grid level on which the problem is solved directly, and we have marked those timings in bold that were the lowest ones obtained for a given problem size. The approach of using (offdiagonal) separable approximations and FFT reports faster results compared to the direct approach for tested problem of sizes starting with $n = 4096$. The results indicate that one may switch from the recursive to the direct solution around level $\eta = 3$ or 4, i.e. when a problem size of $n = \frac{4096}{2^3} = \frac{8192}{2^4} = 512$ is reached since the computational time is superior.

Degrees of freedom $n = 2^\theta$	1024	2048	4096	8192
grid level θ	10	11	12	13
	No separation			
No FFT	22	110	512	2462
grid level η for direct solver	off-diagonal rank-4			
2	39	121	426	1433
3	51	131	**325**	948
4	60	134	339	**814**
5	69	163	377	826
6	80	187	421	917

Table 4.3: Computational times (only source term computations) after 4000 time steps, $dt = 0.001$.

4.2 Evaluation of univariate aggregation integrals on nested grids refined toward the origin

Nested grids refined toward the origin, associated function spaces and bases for these spaces used in this section are defined and explained in subsection 3.2.2. In subsections 4.2.1 and 4.2.2, we discuss the efficient evaluation of the aggregation source (1.3) and sink (1.5) terms, respectively. In subsection 4.2.3 we provide numerical results, including a comparison with the fixed pivot method discussed in section 2.3. Finally, in the last subsection 4.2.4, we turn to the question of mass conservation. The fixed pivot method conserves mass (in addition to the number density) whereas the wavelet method as presented in this section has no

such guarantee. In our algorithms for the source and sink terms, we have used placeholders for projections onto the space $\mathcal{S}$ (3.21). While it is possible to derive (rather complicated) methods to conserve mass [29], in this section we simply illustrate the resulting change of mass when using straightforward projections.

4.2.1 Source term

In order to evaluate the aggregation source term (1.3) we assume that the (initial) density function f lies in the function space $\mathcal{S}$ (3.21), and that the kernel κ in (1.3), (1.5) allows for a separable approximation $\kappa(x,y) \approx \sum_{i=1}^{M} a_i(x)b_i(y)$ with functions $a_i(\cdot), b_i(\cdot) \in \mathcal{S},\ i = 1,\dots,M$. The aggregation source term (1.3) may then be computed as the sum of M terms of the form (see also (3.7))

$$Q(x) := \int_0^x \varphi(y)\psi(x-y)\,dy \tag{4.11}$$

for functions $\varphi := a_i \cdot f,\ \psi := b_i \cdot f \in \mathcal{S}$ (3.21). The major computational effort (yet of still almost linear complexity) lies in the evaluation of the function $Q(x)$ at all grid points $x_i^\ell \in \mathcal{G}$ (3.20) which is derived in this section. In a final step, we project the resulting function (a continuous, piecewise linear approximation of $Q(x)$) onto the approximation space $\mathcal{S}$ (3.21) of piecewise constant functions. The result is the (somewhat surprisingly compact) Algorithm 8.

Next, we will provide the detailed derivation of an algorithm for the efficient evaluation of integrals of the form (4.11) for functions $\varphi,\ \psi \in \mathcal{S}$ (3.21) at all grid points $x_i^\ell \in \mathcal{G}$ (3.20). For a given function $f \in \mathcal{S}$, the Haar basis representation (3.25) yields a decomposition

$$f = \sum_{\ell=0}^{L} f_\ell \qquad \text{with} \qquad f_0 \in \mathcal{V}_0(0,1],$$
$$f_\ell \in \mathcal{W}_\ell(0,2^{-\ell}] \subset \mathcal{W}_\ell(0,1],\ \ell = 1,\dots,L.$$

Let $\varphi, \psi \in \mathcal{S}$ be decomposed in this way, i.e.,

$$\varphi = \sum_{\ell=0}^{L} \varphi_\ell, \qquad \psi = \sum_{\ell=0}^{L} \psi_\ell. \tag{4.12}$$

We now restrict our attention to the evaluation of (4.11) for vertices $x_i^\ell \in (0.5,\,1]$, i.e., to vertices on the right half of the interval, all of which happen to be vertices

of the coarsest grid $\mathcal{G}_0$. All other vertices, i.e., vertices $x_i^\ell \in (0, 0.5]$, are also contained in one of the next finer grids $\mathcal{G}_\ell$, $\ell \geq 1$, and their treatment will follow later by recursion.

In view of the following two properties for functions flipped and shifted by x_i^0,

$$\begin{aligned} \psi_0 \in V_0(0,1] \quad &\Longrightarrow \quad \psi_0(x_i^0 - \cdot) \in V_0(0, x_i^0] \text{ for all } x_i^0 \in \mathcal{G}_0, \\ \psi_\ell \in W_\ell(0, 2^{-\ell}] \quad &\Longrightarrow \quad \psi_\ell(x_i^{\ell-1} - \cdot) \in W_\ell(0, x_i^{\ell-1}] \text{ for all } x_i^{\ell-1} \in \mathcal{G}_{\ell-1}, \end{aligned}$$

and in view of the orthogonality of the subspaces in

$$\mathcal{V}_0(0,1] \oplus \mathcal{W}_1(0,1] \oplus \ldots \oplus \mathcal{W}_L(0,1],$$

(note that here we need $\mathcal{W}_\ell(0,1]$ instead of $\mathcal{W}_\ell(0, 2^{-\ell}]$ since $\mathcal{W}_\ell(0, x_i^{\ell-1}]$ is not necessarily a subset of $\mathcal{W}_\ell(0, 2^{-\ell}]$, but it is certainly a subset of $\mathcal{W}_\ell(0,1]$, see also the example given in Figure 4.9), the convolution integral (4.11) for vertices $x_i^0 \in \mathcal{G}_0$ can be simplified into

$$\begin{aligned} q_i := \int_0^{x_i^0} \varphi(y)\psi(x_i^0 - y)\, dy \quad &\overset{(4.12)}{=} \quad \int_0^{x_i^0} \sum_{\ell=0}^{L} \varphi_\ell(y) \sum_{\nu=0}^{L} \psi_\nu(x_i^0 - y)\, dy \\ &= \quad \sum_{\ell=0}^{L} \int_0^{x_i^0} \varphi_\ell(y)\psi_\ell(x_i^0 - y)\, dy. \end{aligned} \tag{4.13}$$

This representation has a number of properties that allows its efficient computation.

- Since we assumed $x_i^0 \in (0.5, 1]$, we have to evaluate the convolution integrals in (4.13) only for the first two levels $\ell = 0, 1$ since the integrals on all other levels equal zero. An example given in Figure 4.9 illustrates the following explanation:
  - For $\ell \geq 2$, the supports of φ_ℓ, ψ_ℓ lie in $(0, 2^{-\ell}] \subseteq (0, 0.25]$.
  - For $\ell \geq 2$ and $x_i^0 \in (0.5, 1]$, the support of $\psi_\ell(x_i^0 - \cdot)$ therefore lies in $(x_i^0 - 0.25, x_i^0] \subseteq (0.25, 1]$.
  - Thus, φ_ℓ and $\psi_\ell(x_i - \cdot)$ have disjoint supports and their convolution must be zero.

  Hence, for vertices $x_i^0 \in \mathcal{G}_0$, (4.13) can be simplified further into

$$q_i \quad = \quad \underbrace{\int_0^{x_i^0} \varphi_0(y)\psi_0(x_i^0 - y)\, dy}_{Q_i^0 :=} + \underbrace{\int_0^{x_i^0} \varphi_1(y)\psi_1(x_i^0 - y)\, dy}_{Q_i^1 :=}. \tag{4.14}$$

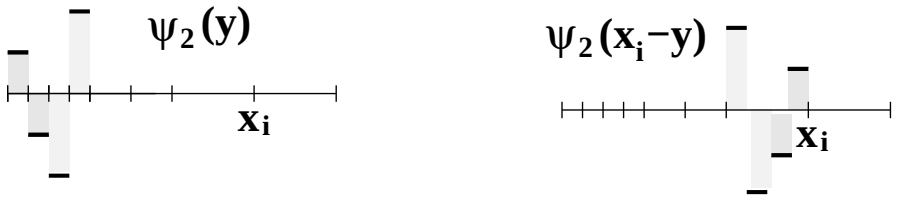


Figure 4.9: A function $\psi_2 \in \mathcal{W}_2(0, 0.25]$ (left) and the flipped and shifted $\psi_2(x_i - \cdot) \in \mathcal{W}_2(0, x_i]$ (right) for $x_i = 0.75 \in (0.5,\ 1] \cap \mathcal{G}_0$.

- The functions $\varphi_\ell(y), \psi_\ell(x_i^0 - y)$ $(y \in (0, x_i^0],\ \ell \in \{0, 1\})$, are each defined on an equidistant grid of width h_ℓ. The convolution of piecewise constant functions on equidistant grids, represented in a nodal basis, can be computed in almost linear complexity by exploiting the fast Fourier transformation (FFT) ([28, 49]). We will refer here to the algorithm FFT_convolution (see Algorithm 3). Hence, this approach can be applied to evaluate these two integrals if the functions are represented with respect to a nodal basis. On the coarsest level $\ell = 0$, the basis coefficients of the Haar basis correspond to basis coefficients of the nodal basis scaled by $\sqrt{h_0} = 1/\sqrt{n}$, and the FFT_convolution computes the first integral Q_i^0 in (4.14) for all $x_i^0 \in \mathcal{G}_0$ simultaneously, even though at this point we are only interested in vertices $x_i^0 \in \mathcal{G}_0 \cap (0.5,\ 1]$. If n is small enough where the direct evaluation without FFT is more efficient, we need to evaluate the integrals Q_i^0 only for those $(n/2)$ vertices $x_i^0 \in \mathcal{G}_0 \cap (0.5,\ 1]$. In Algorithm 8, the evaluation on level $\ell = 0$ is performed in line 12.

- The evaluation of Q_i^1 in (4.14) requires additional work since functions $\varphi_1(y)$, $\psi_1(x_i^0 - y)$ have been assumed to be represented with respect to a Haar basis in order to obtain the levelwise decomposition (4.12). Figure 4.10 (top left) illustrates a function $\psi_1(y) \in \mathcal{W}_1(0,\ 0.5]$. The coefficients for its representation in the Haar basis are $(F_0^1, \ldots, F_{\frac{n}{2}-1}^1)$ and thus, given the (scaling in the) definition of the Haar basis functions (3.23), $\psi_1(y)$ has the piecewise constant function values (i.e., nodal basis coefficients)

$$(f_0^\psi, f_1^\psi, \ldots, f_{n-2}^\psi, f_{n-1}^\psi) := \left(\frac{F_0^1}{\sqrt{2h_1}}, \frac{-F_0^1}{\sqrt{2h_1}}, \ldots, \frac{F_{\frac{n}{2}-1}^1}{\sqrt{2h_1}}, \frac{-F_{\frac{n}{2}-1}^1}{\sqrt{2h_1}} \right). \tag{4.15}$$

On the right in Figure 4.10, we illustrate the flipped and shifted functions $\psi_1(x_i^0 - y) \in \mathcal{V}_1(0, x_i^0]$ for some $x_i^0 \in (0.5,\ 1] \cap \mathcal{G}_0$. In particular, the support of the product function $\varphi_1(y)\psi_1(x_i^0 - y)$ lies within $[h_0,\ 0.5]$, and the integral Q_i^1 in (4.14) can be computed (using the index numbering illustrated in the

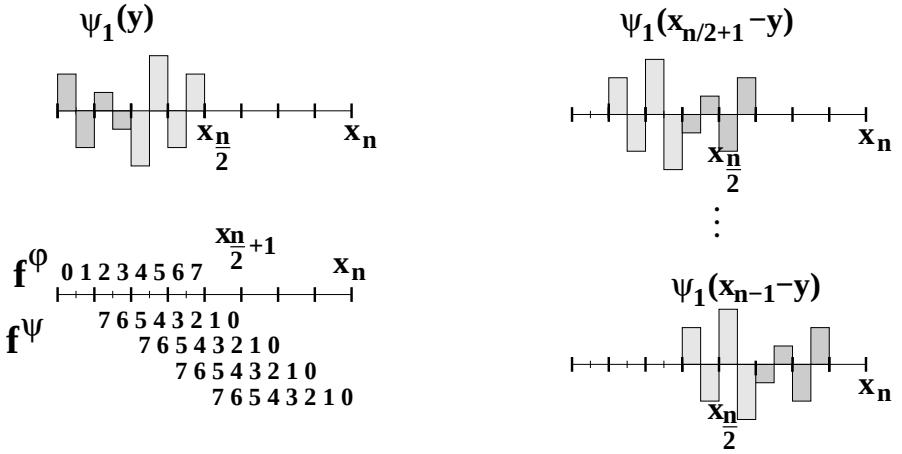


Figure 4.10: A function $\psi_1 \in \mathcal{W}_1(0, 0.5]$ (top left) and two flipped and shifted $\psi_1(x_i^0 - \cdot) \in \mathcal{V}_1(0, x_i^0]$ for different $x_i^0 \in (0.5,\, 1]$ (right, top and bottom). Bottom left: Illustration of index numbering in convolution (4.16).

bottom left of Figure 4.10) as

$$
\begin{aligned}
Q_i^1 &= \int_0^{x_i^0} \varphi_1(y)\psi_1(x_i^0 - y)\, dy \\
&= h_1 \sum_{j=1}^{n-i} \left(f^{\varphi}_{2(i-1)-(n-2j)} f^{\psi}_{n-2j+1} + f^{\varphi}_{2(i-1)-(n-2j)+1} f^{\psi}_{n-2j} \right) \\
&= h_1 \sum_{j=1}^{n-i} \left(f^{\varphi}_{2(i-1)-(n-2j)} (-f^{\psi}_{n-2j}) + (-f^{\varphi}_{2(i-1)-(n-2j)}) f^{\psi}_{n-2j} \right) \\
&= -2h_1 \sum_{j=1}^{n-i} f^{\varphi}_{2(i-1)-(n-2j)} f^{\psi}_{n-2j} \\
&\overset{(4.15)}{=} -2h_1 \sum_{j=1}^{n-i} \frac{F^{\varphi}_{i-1-(\frac{n}{2}-j)}}{\sqrt{2h_1}} \frac{F^{\psi}_{\frac{n}{2}-j}}{\sqrt{2h_1}} \\
&= -\sum_{j=1}^{n-i} F^{\varphi}_{i-1-(\frac{n}{2}-j)} F^{\psi}_{\frac{n}{2}-j}, \qquad i = \frac{n}{2}+1, \ldots, n.
\end{aligned}
$$

Defining G^{φ}, G^{ψ} to be F^{φ}, F^{ψ} in reversed order (line 16 in Algorithm 8),

i.e. $G_j^\varphi := F^\varphi_{\frac{n}{2}-1-j}$ and $G_j^\psi := F^\psi_{\frac{n}{2}-1-j}$ for $j = 0, \ldots, \frac{n}{2}-1$, this sum becomes

$$Q_i^1 \;=\; -\sum_{j=1}^{n-i} G^\varphi_{n-i-j} G^\psi_{j-1}, \quad i = \frac{n}{2}+1, \ldots, n.$$

Finally, setting $i = n-k+1$, we obtain

$$Q^1_{n-k+1} = -\sum_{j=1}^{k-1} G^\varphi_{k-1-j} G^\psi_{j-1}, \qquad k = 1, \ldots, \frac{n}{2}, \tag{4.16}$$

a convolution problem of size $\frac{n}{2}$ which is implemented in line 17 in Algorithm 8 and then, in line 19, added to the previously computed convolution on level $\ell = 0$.

This completes the evaluation of the convolution integral (4.11) for all $x_i^0 \in (0.5, 1]$, and it remains to evaluate (4.11) for $x_i^\ell \in (0, 0.5]$, $1 \le \ell \le L$. In order to apply the above argument recursively, i.e., to next evaluate (4.11) for all $x_i^1 \in \mathcal{G}_1 \cap (0.25,\, 0.5]$ which are the grid points on the right half of the level-1 grid $\mathcal{G}_1$, (of which there are again $n/2$) but only on the respective two coarsest levels, we need to make the following two preparations:

- Let φ_0, ψ_0 be the respective first terms of the decompositions of φ, ψ in (4.13), i.e., the functions in $\mathcal{V}_0(0,1]$ and hence the only functions in the decomposition (4.13) with possible support in (0.5, 1.0]. Since now $x_i^\ell \le 0.5$, it follows that both $\varphi_{0,\text{right}}$ and $\psi_{0,\text{right}}$, the restrictions of φ_0, ψ_0 to the right interval (0.5, 1.0], are irrelevant for the convolution $\int_0^{x_i^\ell} \varphi_0(y)\psi_0(x_i^\ell - y)\, dy$ (they are "outside the region of overlap") as is also illustrated by an example in Figure 4.11. Hence, in (4.13) we may replace φ_0, ψ_0 by two new functions φ_0^0, ψ_0^0 which are equal to φ_0, ψ_0 on (0, 0.5] but zero on (0.5, 1.0] without changing the convolution.

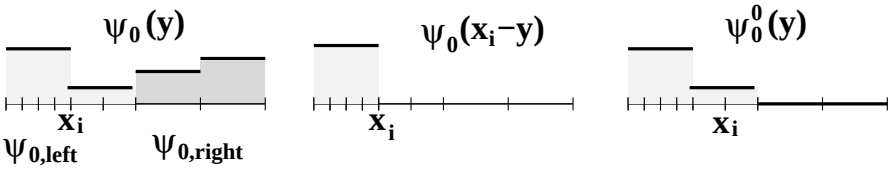


Figure 4.11: A function $\psi_0 \in \mathcal{V}_0(0,1]$ (left), the flipped and shifted $\psi_0(x_i - \cdot) \in \mathcal{V}_0(0, x_i]$ for $x_i = 0.25 \in (0,\, 0.5]$ (middle) and the function $\psi_0^0 \in \mathcal{V}_0(0,1]$ (right).

- Next, we note that the (new) partial sum $\varphi_0^0 + \varphi_1$, currently represented in a wavelet basis, can easily be transformed into a nodal basis representation

in $\mathcal{V}_1(0,\, 0.5]$. The same applies to the partial sum $\psi_0^0 + \psi_1$. Performing such basis transformations takes us to the situation that we need to compute a convolution of functions in wavelet basis representation with $\mathcal{G}_1$ instead of $\mathcal{G}_0$ as a coarsest grid. Hence we can proceed recursively, i.e., begin with the $n/2$ vertices in the right half of $\mathcal{G}_1$, etc.
This is where Lemma 1 becomes useful: The required nodal basis coefficients have already been computed as an auxiliary result in Algorithm 2 and are stored in the matrix A.

Once we reach the finest level L, we finish with a convolution involving all vertices on the (equidistant) grid $\mathcal{G}_L$ (line 22 of Algorithm 8). We have now computed the convolution integral (4.11) for all $x_i^g \in \mathcal{G}$. We complete the algorithm with a projection of the piecewise linear function given by these nodal values onto the space $\mathcal{S}$ of piecewise constant functions. The following remark discusses this projection.

Remark 3. *The described algorithm is implemented in Algorithm 8 and computes as an intermediate result a continuous, piecewise linear approximation to the aggregation source term, given by its values at the vertices of the graded mesh (whereas the exact aggregation source term is piecewise linear on an equidistant grid of the finest grid width h_L). The nodal values are stored in an $(L+1) \times n$ matrix q with $q_j^\ell = Q_{source}(f)(x_j^\ell)$ for $0 \le \ell \le L-1,\ n/2+1 \le j \le n$ (grid points on all but the finest level) and for $\ell = L, 1 \le j \le n$ (grid points on finest level) (4.11, 4.13).*

In a final step, this piecewise linear approximation needs to be projected onto the space $\mathcal{S}$ of piecewise constant functions on the graded mesh. Using $q_0^L := Q_{source}(f)(0) = 0$, we replace these nodal values by the projected piecewise constant function values on the interval $(x_{j-1}^\ell, x_j^\ell]$ to the left of the respective node x_j^ℓ. Line 23 of the Algorithm 8 provides a place holder for this final step. While there exist several straightforward approaches for such a projection, the question of mass conservation is quite involved and has been addressed in [29]. In our numerical results in section 5.5, we will use a mass-preserving projection from piecewise linear to piecewise constant functions. In subsection 4.2.4, we illustrate and comment on the change in mass which resulted previously through the piecewise constant approximation on a graded (instead of an equidistant finest level) grid.

Here, as in subsection 4.1.2 and 4.1.1, we use the algorithm *FFT_convolution* (n, φ, ψ) (Algorithm 3) of a fast convolution on an equidistant grid (used in lines 12, 17, 22 of Algorithm 8), that given two vectors $\varphi, \psi \in \mathbb{R}^n$, computes the convoluted sum $\omega \in \mathbb{R}^n$ with entries $\omega_i = \sum_{j=0}^{i} \varphi_i \psi_{i-j},\ i = 0, \ldots, n-1$ in $\mathcal{O}(n \log n)$ complexity. If the vectors $\varphi, \psi \in \mathbb{R}^n$ denote the (piecewise constant) function values of an equidistant grid of grid width $h = \frac{1}{n}$ with grid points $x_i = ih, i = 0, \ldots, n$, then it

holds that

$$\int_0^{x_i} \varphi(y)\psi(x_i - y)\, dy = h\omega_i, \quad i = 0, \ldots, n-1.$$

4.2.2 Sink term

In order to evaluate the aggregation sink term (1.5), we again assume that the (initial) density function f lies in the function space $\mathcal{S}$ (3.21), and that the kernel κ in (1.5) is approximated by a sum $\kappa(x,y) \approx \sum_{i=1}^{M} a_i(x)b_i(y)$ with functions $a_i(\cdot), b_i(\cdot) \in \mathcal{S},\ i = 1, \ldots, M$. The aggregation sink term (1.5) may then be computed as the sum of M terms of the form (see also (3.8))

$$Q(x) := \psi(x) \underbrace{\int_0^{1-x} \varphi(y)\, dy}_{=:\bar{\varphi}(x)}$$

for functions $\varphi := a_i \cdot f, \psi := b_i \cdot f \in \mathcal{S}$ which is the subject of this subsection. The first (and computationally most expensive) step is the evaluation of $\bar{\varphi}$ at all grid points $x_i^\ell \in \mathcal{G}$, defining a continuous, piecewise constant approximation to the function $\bar{\varphi}(x)$. It turns out that a representation of the density function φ with respect to a Haar wavelet basis once more simplifies the evaluation of $\bar{\varphi}(x)$ at the grid points as we will show next.

Let $\varphi \in \mathcal{S}$ be decomposed by level as in (4.12), i.e.,

$$\varphi \;=\; \sum_{\tilde{\ell}=0}^{L} \varphi_{\tilde{\ell}} \qquad \text{with } \varphi_0 \in \mathcal{V}_0(0,1],\ \varphi_{\tilde{\ell}} \in \mathcal{W}_{\tilde{\ell}}(0, 2^{-\tilde{\ell}}],\ \tilde{\ell} = 1, \ldots, L.$$

For $x_i^\ell \in \mathcal{G}$, it holds that

$$\bar{\varphi}(x_i^\ell) = \int_0^{1-x_i^\ell} \varphi(y)\, dy = \sum_{\tilde{\ell}=0}^{L} \int_0^{1-x_i^\ell} \varphi_{\tilde{\ell}}(y)\, dy = \int_0^{1-x_i^\ell} \varphi_0(y)\, dy \tag{4.17}$$

since $\int_0^{1-x_i^\ell} \varphi_{\tilde{\ell}}(y)\, dy = 0$ for all $1 \le \tilde{\ell} \le L$ (recall that $\varphi_{\tilde{\ell}}(y)|_{(\frac{1}{2},1]} = 0$ for $1 \le \tilde{\ell} \le L$). The integrand $\varphi_0(y)$ is piecewise constant on the coarsest grid $\mathcal{G}_0$, and according to the definition of hierarchical basis functions (3.23) and the representation of a function in the Haar wavelet basis (3.25), its function values are

$$\varphi_0(y) = \sqrt{n} \cdot F_j^0 \overset{(3.32)}{=} \sqrt{n} \cdot \tilde{r}_j^0 \quad \text{for } y \in (x_j^0, x_{j+1}^0),\ j = 0, \ldots, n-1.$$

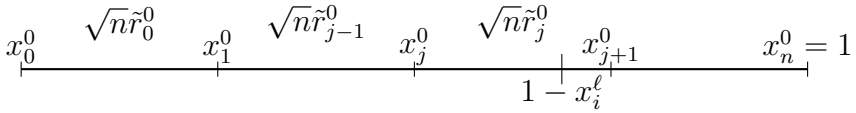
Algorithm 8 q= gradedMeshSourceAggregationRank1$(n,\ L,\ f,\ a,\ b)$;
(Evaluate the aggregation source integral using piecewise constant approximations on a graded mesh for separation rank 1, i.e. $\kappa(x,y) = a(x)b(y)$.)

Input: $n \in \mathbb{N}$; (number of intervals on coarse mesh)
$L \in \mathbb{N}$; (number of nested refinements of grid)
$f = (f_j^\ell)_{\substack{0\le\ell\le L\\ 0\le j<n}} \in \mathbb{R}^{(L+1)\times n}$; (nodal values of density function (3.26))
$\left(a_j^\ell\right)_{\substack{0\le\ell\le L\\ 0\le j<n}}, \left(b_j^\ell\right)_{\substack{0\le\ell\le L\\ 0\le j<n}} \in \mathbb{R}^{(L+1)\times n}$; (kernel factors, piecew. const. on $\mathcal{G}_\ell$)
Output: $q = (q_j^\ell)_{\substack{0\le\ell\le L\\ 1\le j\le n}} \in \mathbb{R}^{(L+1)\times n}$; (piecew. const. aggreg. $q_j^\ell \approx Q_{\text{source}}(f)|_{(x_{j-1}^\ell, x_j^\ell)}$)

for $j = 0 : (\frac{n}{2} - 1)$ **do**
  $\varphi_j^L = b_j^L \cdot f_j^L; \qquad \psi_j^L = a_j^L \cdot f_j^L;$
end for
for $\ell = 0 : L$ **do**
  for $j = \frac{n}{2} : (n-1)$ **do**
    $\varphi_j^\ell = b_j^\ell \cdot f_j^\ell; \qquad \psi_j^\ell = a_j^\ell \cdot f_j^\ell;$
  end for
end for
(A_φ, Φ) = nodal_to_Haar$(n,\ L,\ \varphi)$; (Algorithm 2)
(A_ψ, Ψ) = nodal_to_Haar$(n,\ L,\ \psi)$; (Algorithm 2)
for $\ell = 0 : (L-1)$ **do**
  $(a_1, \ldots, a_n)$= FFT_convolution$(n,\ A_\varphi(\ell,:),\ A_\psi(\ell,:))$;
  for $i = 1 : \frac{n}{2}$ **do**
    $q_{\frac{n}{2}+i}^\ell = a_{\frac{n}{2}+i}$;
  end for
  $Y := (\Phi_{\frac{n}{2}-1}^{\ell+1}, \ldots, \Phi_0^{\ell+1}); \qquad Z := (\Psi_{\frac{n}{2}-1}^{\ell+1}, \ldots, \Psi_0^{\ell+1});$
  $(c_1, \ldots, c_{\frac{n}{2}}) = -$ FFT_convolution$(\frac{n}{2},\ Y,\ Z)$;
  for $j = 1 : \left(\frac{n}{2} - 1\right)$ **do**
    $q_{\frac{n}{2}+j}^\ell$ += $c_{\frac{n}{2}-j}$;
  end for
end for
$q(L,:)$ = FFT_convolution$(n,\ A_\varphi(L,:),\ A_\psi(L,:))$;
q = project_linear2constant$(n, L, (0, q))$; (see Remark 3)
return (q);

In order to evaluate $\bar{\varphi}(x_i^\ell)$ for a grid point $x_i^\ell \in \mathcal{G}$, we first determine the index j of the coarse grid vertex x_j^0 such that

$$x_j^0 < 1 - x_i^\ell \leq x_{j+1}^0.$$



Since $x_i^\ell = ih_\ell = i2^{-\ell}h$ (3.18) and $h = \frac{1}{n} = 2^{-\theta}$, this inequality turns into

$$\begin{aligned} jh &< 1 - i2^{-\ell}h \leq (j+1)h \qquad | \cdot h^{-1} = n = 2^\theta \\ \iff \quad j &< 2^\theta - i2^{-\ell} \leq j+1 \end{aligned}$$

and hence

$$j = \lfloor 2^\theta - i2^{-\ell} \rfloor. \tag{4.18}$$

(Gauß bracket/floor function $\lfloor \cdot \rfloor$, i.e., $\lfloor r \rfloor$ is the largest integer smaller or equal to r.) Finally, we compute

$$\begin{aligned} \bar{\varphi}(x_i^\ell) &= \frac{1}{n} \cdot \sqrt{n} \sum_{k=0}^{j-1} \tilde{r}_k^0 + \sqrt{n}(1 - x_i^\ell - x_j^0)\tilde{r}_j^0 \\ &= A_{\Sigma,j-1} + \sqrt{n}(1 - x_i^\ell - x_j^0)\tilde{r}_j^0 \quad \text{with partial sums} \\ A_{\Sigma,j-1} &:= \frac{1}{\sqrt{n}} \sum_{k=0}^{j-1} \tilde{r}_k^0. \end{aligned}$$

The evaluation of $\bar{\varphi}(x_i^\ell)$ is implemented in Algorithm 9. The piecewise linear function defined through these vertex values is next projected onto the space $\mathcal{S}$ of piecewise constant functions and finally multiplied by $\psi(x) \in \mathcal{S}$ to obtain an approximation $\tilde{Q}(x_i^\ell) \in \mathcal{S}$ to the piecewise linear (in general discontinuous) $Q(x)$. These steps summarize the evaluation of the sink integral and are implemented in Algorithm 10.

4.2.3 Numerical results

In this subsection, we show numerical results for the methods proposed in the previous sections. We illustrate our methods through numerical tests for the following three kernels: constant kernel (3.1), Brownian kernel (3.2), nonlinear shear kernel (3.3).

Algorithm 9 $\bar{\varphi}$ = gradedMeshSinkAuxIntegral($n,\ L,\ f,\ b$);
(Evaluate $\bar{\varphi}_i = \int_0^{1-x_i} b(y)f(y)\,dy$, $i = 0, \ldots, N$, using piecewise constant approximation on a graded mesh)

Input: $n \in \mathbb{N}$; (intervals on coarse grid)
$L \in \mathbb{N}$; (number of nested grid refinements)
$f = (f_0, \ldots, f_{N-1}) \in \mathbb{R}^N$; (piecewise constant density function values)
$b = (\beta_0, \ldots, b_{N-1}) \in \mathbb{R}^N$; (projection of kernel factor onto $\mathcal{S}$)
Output: $\bar{\varphi} \in \mathbb{R}^{N+1}$; (auxiliary integral values (4.17))

for $j = 0 : (\frac{n}{2} - 1)$ **do**
  $\varphi_j^L = b_j^L \cdot f_j^L$;
end for
for $\ell = 0 : L$ **do**
  for $j = \frac{n}{2} : (n-1)$ **do**
    $\varphi_j^\ell = b_j^\ell \cdot f_j^\ell$;
  end for
end for
(A_φ, Φ)=nodal_to_Haar(n, L, φ);
$N = \left(1 + \frac{L}{2}\right) n$; (number of intervals on graded mesh)
$A_{\Sigma,0} = 0$;
$\bar{\varphi}_N = 0$;
for $k = 1 : n$ **do**
  $A_{\Sigma,k} = A_{\Sigma,k-1} + r_{k-1}^0 \cdot \frac{1}{\sqrt{n}}$;
end for
for $\ell = 0 : (L-1)$ **do**
  for $i = \frac{n}{2} : n-1$ **do**
    ind=floor$(n - i \cdot 2^{-\ell})$; (see (4.18))
    $\bar{\varphi}_{N - \frac{\ell \cdot n}{2} + i - n} = A_{\Sigma,\mathrm{ind}} + (1 - \frac{\mathrm{ind} + i \cdot 2^{-\ell}}{n}) \cdot r_{\mathrm{ind}}^0 \cdot \sqrt{n}$;
  end for
end for
for $i = 0 : (n-1)$ **do**
  ind=floor$(n - i \cdot 2^{-L})$ (see (4.18))
  $\bar{\varphi}_i = A_{\Sigma,\mathrm{ind}} + (1 - \frac{\mathrm{ind} + i \cdot 2^{-L}}{n}) \cdot r_{\mathrm{ind}}^0 \cdot \sqrt{n}$;
end for
return $(\bar{\varphi})$;

Algorithm 10 q = gradedMeshSinkAggregation($n,\ L,\ f,\ M,\ a,\ b$);
(Piecewise constant approximation of the aggregation sink integral (1.5))

Input: $n \in \mathbb{N}$; ($n = 2^\theta$ intervals on coarse grid)
$L \in \mathbb{N}$; (number of nested refinements of grid)
$f = (f_1, \ldots, f_N) \in \mathbb{R}^N$; (piecewise constant density function values)
$M \in \mathbb{N}$; (separation rank of kernel function)
$a, b \in \mathbb{R}^{M \times N}$; (projection of kernel factors onto $\mathcal{S}$)
Output: $q = (q_1, \ldots, q_N) \in \mathbb{R}^N$; (piecew. const. sink value $q_i \approx Q_{\text{sink}}(f)|_{(x_{i-1}, x_i)}$)

$N = \left(1 + \frac{L}{2}\right) n$; (number of intervals on graded mesh)
$q := (0, \ldots, 0)$; (initialize piecewise constant sink vector $q \in \mathbb{R}^N$)
for $\nu = 1 : M$ **do**
  $\bar{\varphi}$= gradedMeshSinkAuxIntegral($n,\ L,\ f,\ b(\nu, :)$);
  $\bar{\varphi}$ = project_linear2constant($n,\ L,\ \bar{\varphi}$); (see Remark 3)
  for $j = 1 : N$ **do**
    q_j += $a_j^\nu \cdot f_j \cdot \bar{\varphi}_j$;
  end for
end for
return (q);

The Brownian kernel has separation rank $M = 3$ since it can be expressed as shown in (3.9). Unlike the Brownian kernel, the shear kernel is not separable but can be expressed by a rank-M approximation where $M = 9$ leads to an approximation error of machine precision (10^{-15}). The approximation of the shear kernel with Chebyshev polynomials leads to highly accurate results for ranks $M \approx 5$ (see Figure 4.4). The approximation of the shear kernel using adaptive cross approximation reaches accurate results even for lower ranks $M \approx 2$, due to the same reason mentioned for uniform grids in subsection 4.1.3: The behavior of the shear kernel K_S is computed near to the peak of the density distribution.

The numerical results will be presented in four parts. The first three parts contain numerical results for the Brownian and constant kernels whereas in the last one we present results for the shear kernel.

In the first part, we show the advantage of the locally refined grids over uniform grids with the same number of grid points in terms of the accuracy of the computed density distribution. We next compare the wavelet method on locally refined nested grids to the fixed pivot method on a geometric grid, again for grids with the same number of grid points. It turns out that the computed solutions are comparable in accuracy while the method using wavelets on locally refined grids is

significantly faster. This has been expected in view of the almost linear complexity of the wavelet method compared to the quadratic complexity of the fixed pivot method.

In the second part we discuss timings for the wavelet method on nested grids with respect to the grid parameters n (number of intervals in initial subdivision) and L (number of recursive grid refinements), leading to a total number of $N = (1+L/2)n$ (3.19) intervals in the nested grid. We observe an initial increase in time when switching from a uniform to a nested grid (assuming the same number of total intervals) but only a very mild increase as we increase the number of levels L.

In the third part we provide numerical results for the shear kernel function κ_{S} for which we use a separable approximation. Here, we show that as the rank is increased in the separable approximation, so is the accuracy of the computed solution. We provide a comparison with timings for the fixed pivot method which does not require (and does not benefit from) a separable kernel approximation, i.e., it uses the exact kernel function. However, it requires quadratic work complexity which makes the separable approach on nested grids computationally faster even for higher separation ranks M.

Comparison of computations on uniform versus non-uniform grids

All results in this part of the subsection have been computed for the Brownian kernel (3.9) of separation rank 3. The first numerical test illustrates the advantage with respect to the accuracy of the computed solution of a locally refined grid over an equidistant grid with the same number of grid points. To this end, Figure 4.12 shows the density distributions $f_1(4, x)$ (left) and $f_1(11, x)$ (right) computed at times $t = 4$ and $t = 11$, resp., where the initial distribution $f_1(0, x)$ (3.12) has been chosen. We show results on a locally refined as well as a uniform grid, both with $N = 32$ grid points. As a reference solution, we compute a solution on a uniform grid with $N = 256$. The result on the locally refined grid is noticeably better than the solution on the uniform grid with the same number of grid points.

The second test compares the wavelet technique on a locally refined grid with the fixed pivot technique on a geometric grid. In Figure 4.13 (left) we show the computed density distributions $f_1(4, x)$ (3.12) ($dt = 0.1$) for both the fixed pivot technique and the wavelet technique where again $f_1(0, x)$ (3.12) is chosen as the initial density distribution. While the computed solutions are comparable (in fact, the L_2 norm of their difference is $1.9974e-05$), the computational times to obtain these results differ significantly. Timings for various problem sizes N are illustrated in Figure 4.13 (right), clearly showing the different computational complexities of these two approaches: $\mathcal{O}(N^2)$ for the fixed pivot method on a geometric grid and

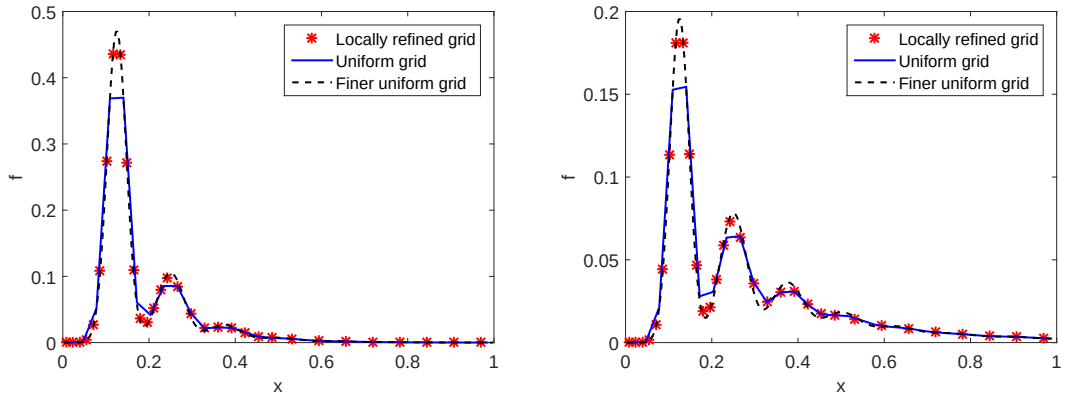


Figure 4.12: Density distributions $f_1(t,x)$ for initial density distribution $f_1(0,x)$ at $t=4$ (left) and $t=11$ (right) ($dt=0.1$) for a uniform grid ($N=32$), a locally refined grid ($L=2, n=16$, i.e., $N=32$) and a finer uniform grid ($N=256$) for the Brownian kernel.

$\mathcal{O}(N\log N)$ for the wavelet method on a locally refined nested grid. Additional timings for various settings of the nested grid are given in Table 4.4. These timings comprise the computation of 500 time steps ($dt=0.01$) of both the source and the sink integrals. Table 4.4 and Figure 4.13 (right) illustrate timings for constant as well as Brownian kernels. One can notice that the running time for the wavelet method on a locally refined grid for the constant kernel is around three times faster than the running time for the Brownian kernel. This is explained by the fact that the Brownian kernel is of separation rank three while the constant kernel is of rank one.

Degrees of freedom N	128	256	512	1024	2048
Nested grid settings $L=2,\quad n$	64	128	256	512	1024
Locally refined grid Brownian kernel (3.2)	0.76	1.4	2.7	5.78	11.8
Locally refined grid constant kernel (3.1)	0.38	0.61	1.1	2.07	4.2
Fixed Pivot (Brownian (3.2) or constant (3.1) kernels)	0.36	0.9	3.05	11.7	46.6

Table 4.4: Times (in seconds) to compute 500 time steps ($dt=0.01$) of the source and sink terms .

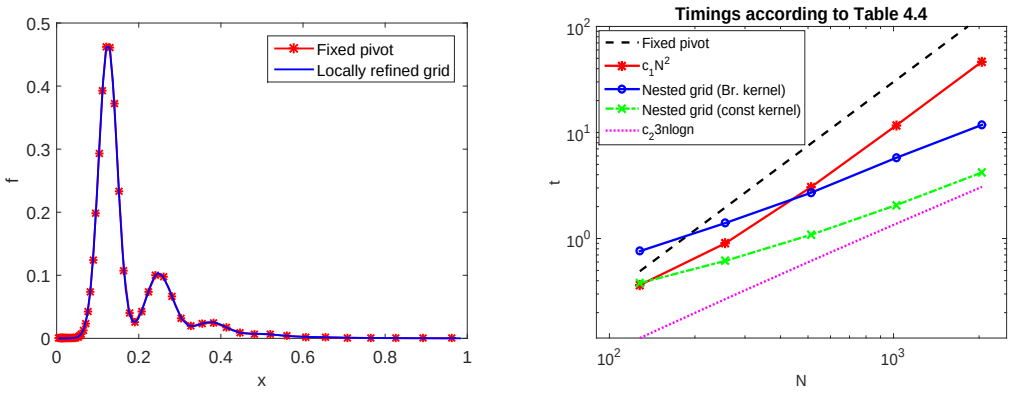


Figure 4.13: Left: Density distribution $f_1(4,x)$ for initial distribution $f_1(0,x)$ ($dt = 0.1$) using the fixed pivot method on a geometric grid ($N = 64$, $v_0 = 0$, $v_1 = h_L$, $v_N = 1$, $v_{i+1} = \tau v_i$, $i = 1,\dots,N-1$) and wavelets on a locally refined grid ($L = 2$, $n = 32$, i.e., $N = 64$).
Right: Computational times (seconds) for $f(5,x)$ for the fixed pivot and wavelet methods depending on the problem size N ($c_1 = 3e-5$, $c_2 = e-4$) for 500 time steps (dt=0.01).

Computational time with respect to grid parameters n, L

Here, numerical results are represented to illustrate the dependence of computational times on the parameters n, L in the construction of the nested grid. The timings in this subsection include only computations for the source term which are of complexity $\mathcal{O}((L+1)n\log n) = \mathcal{O}(N\log n)$ [28] and clearly dominate the cost of the sink term computations (which are of complexity $\mathcal{O}(N)$) .

In Figure 4.14 (left), we show the computational times for 500 time steps ($dt = 0.01$) for different combinations of levels L and number of finest level intervals n. In view of Assumption 1, i.e., $n = 2^\theta$ for some $p \in \mathbb{N}$, we choose $L \in \{0,2,6,14\}$ as the number of refinement levels so that the total number $N = (1+L/2)n$ (3.19) of intervals becomes a power of 2, i.e., $N = 2^\theta$, and timings for settings (n,L) leading to the same N can be directly compared. In Figure 4.14 (left), we also display curves for constant $N \in \{512, 1024, 2048, 4096\}$, showing the minor increase in cost when transitioning from a uniform to a nested grid.

On a uniform grid, the computational complexity is $\mathcal{O}(n\log n)$ whereas the complexity on a locally refined grid is $\mathcal{O}((L+1)n\log n)$. In Figure 4.14 (left), we have inserted respective lines for these complexity bounds for the uniform ($L = 0$) and

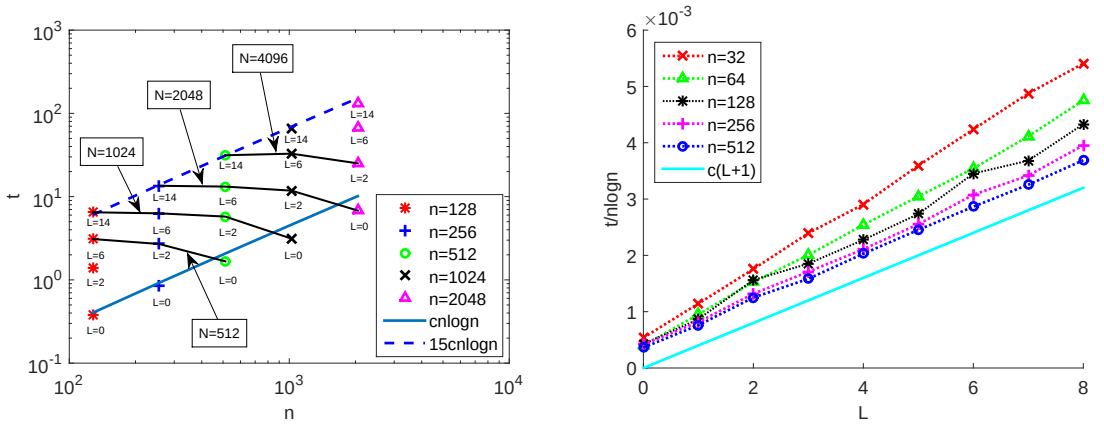


Figure 4.14: Computational times for various parameter choices (L, n) for the nested grid (source term, 500 time steps, $dt = 0.01$); Left: Illustration of $\mathcal{O}(n \log n)$ complexity for fixed L, $c = 4.5e\text{-}3$ in complexity graphs; Right: Illustration of $\mathcal{O}(L)$ complexity for fixed n, $c = 4e\text{-}4$ in complexity graph.

nested grids (for $L = 14$) which confirm these theoretical complexity orders.

The complexity estimate $\mathcal{O}((L+1)n \log n)$ for graded meshes even implies linear complexity in the problem size N ($\in \mathcal{O}((L+1)n \log n) = \mathcal{O}(N \log n)$) if n is fixed and the increase in N is obtained through an increase of the levels L of refinement. This is confirmed through the timings displayed in Figure 4.14 (right) for fixed $n \in \{32, 64, 128, 256, 512\}$ and increasing L, leading to the increase in N. Here, the slopes c_n of the lines $c_n L \approx \frac{t_n(L)}{n \log_2 n}$, where $t_n(L)$ denotes the measured timings for a problem in $N = (1 + \frac{L}{2})n$ unknowns, converge to a constant, i.e., $c_n \overset{n\to\infty}{\longrightarrow} c$. In fact, the slopes c_n decrease until n is large enough so that the FFT-based aggregation on uniform grids with n intervals reaches its asymptotic complexity $\mathcal{O}(n \log n)$.

Numerical tests for the shear kernel

In this part, we show numerical results for the shear kernel (3.3) which is not separable but allows for an exponentially convergent separable approximation with numerical separation rank 9. In particular, numerical tests suggest that less than ten singular values σ_k of the discrete kernel matrix $(\kappa(x_i, y_j))_{0 \le i,j \le N-1}$ are larger than typical machine precision ($\mathcal{O}(10^{-15})$) and these "numerically non-zero" singular values decay exponentially. In Figure 4.5 separable approximation of the shear kernel has been constructed through Chebyshev interpolation and adaptive cross approximation of the kernel function. Both methods are described in detail

in subsection 4.1.3.

The numerical results in this section are designed to show that the wavelet method on nested grids also works well for applications in which the kernel is not separable but needs to be replaced by a separable approximation. Here as well, we will illustrate results using Chebyshev polynomial and adaptive cross approximation.

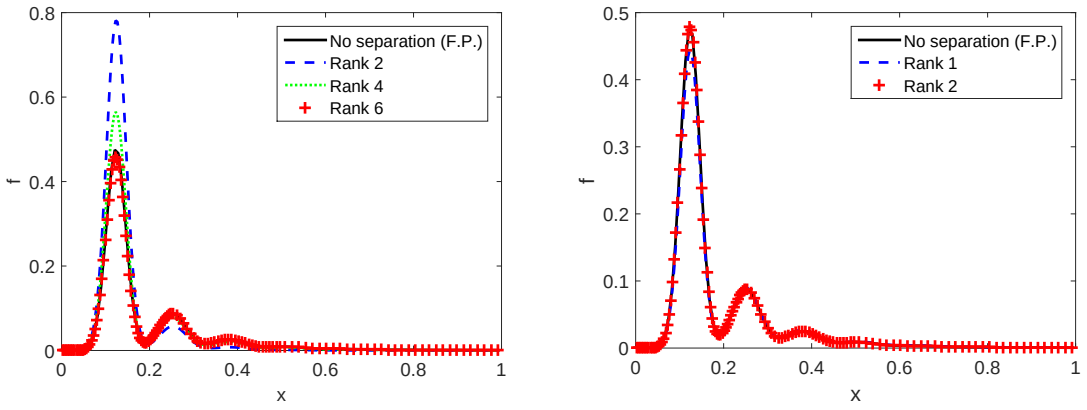


Figure 4.15: Left: Density distribution $f_1(15, x)$ with initial distribution $f_1(0, x)$ ($dt = 0.1$) for the fixed pivot method (geometric grid, $N = 128$) and for the wavelet method (locally refined grid, $L = 2$, $n = 64$) for different separation ranks using Chebyshev polynomials (left) and adaptive cross approximation (right)

Figure 4.15 (left) shows the computed density distributions $f_1(15, x)$ for the initial distribution $f_1(0, x)$ using different ranks in the separable approximation to the shear kernel using Chebyshev polynomials and Figure 4.15 (right) using adaptive cross approximation method. For Chebyshev polynomials, while the computed density distributions for ranks 2 and 4 still show visible differences in the results, this is no longer the case when using rank 6 (or higher). The adaptive cross approximation method leads to a very competitive approximation already for rank 2 solutions. Figure 4.16 illustrates the computational timings for 100 time steps various problem sizes N and separation ranks $M \in \{2, 4, 6, 8\}$ that are explicitly given in Table 4.5. It shows once more the clear advantage with respect to computational time of the wavelet method using separable kernel approximations compared to the fixed pivot method.

The timings in Table 4.5 confirm that for a fixed problem size N, the computational time increases linearly in the rank of the separable kernel approximation.

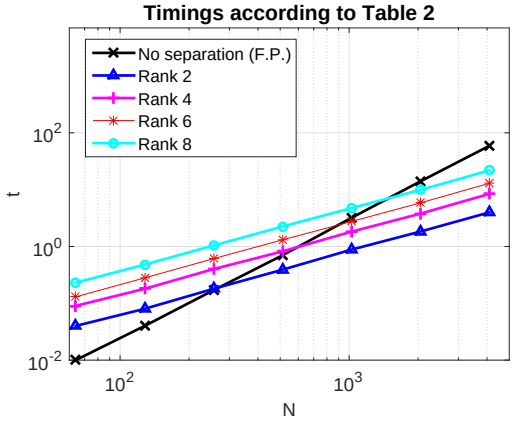


Figure 4.16: Computational time (in seconds) for sink and source terms (100 time steps, $dt = 0.1$) for the fixed pivot method (geometric grid) and the wavelet method (nested grid) using different separation ranks (Chebyshev and ACA).

4.2.4 Change of mass

In this subsection we will provide additional detail on the subject of mass preservation (recall (4.1)) which we already mentioned in Remark 3.

The discretization of the density function through piecewise constant functions leads to piecewise linear source (1.3) and sink (1.5) terms which need to be projected back to piecewise constant functions. The difficulty lies in the fact that for a locally refined nested grid, these functions are piecewise linear not on the nested grid but on an equidistant grid of the grid width h_L of the finest level of the nested grid. In Figure 4.17, we show a function that is piecewise linear on a fine grid with grid width $h_f = 1/8$. In the wavelet approach, this function is only evaluated at the grid points of the graded grid which becomes coarser towards the right part of the computational domain. In Figure 4.17, we show as an example a piecewise linear function on the coarser grid with grid width $h_c = 1/2$. Since these two piecewise linear functions have different mass, their mass-preserving projections to piecewise constant functions will yield different results. In this particular example of convex functions, the approximation on a coarser grid leads to an increase in mass. The approximation of a concave density distribution by a piecewise linear function on a given interval would lead to a loss of mass.

In [29], a rather involved method is developed that preserves mass for the source term while maintaining the almost linear complexity of its computation. A different approach is to consider both the source and sink terms simultaneously and enforce that mass is conserved. Here, we do not pursue either of these two ap-

degrees of freedom (N)		64	128	256	512	1024	2048	4096
n		32	64	128	256	512	1024	2048
wavelet method	rank 2	0.04	0.08	0.18	0.39	0.88	1.8	3.9
nested grid	rank 4	0.09	0.18	0.4	0.82	1.8	3.7	8.4
with L=2	rank 6	0.13	0.28	0.61	1.3	2.7	5.8	12.9
	rank 8	0.23	0.48	1.04	2.2	4.7	9.9	21.7
fixed pivot method		0.01	0.04	0.17	0.7	3.19	14	59

Table 4.5: Computational times (in seconds) for 100 time steps ($dt = 0.1$) for source and sink terms for the shear kernel for the wavelet method on locally refined grids for different separation ranks and for the fixed pivot method on a geometric grid.

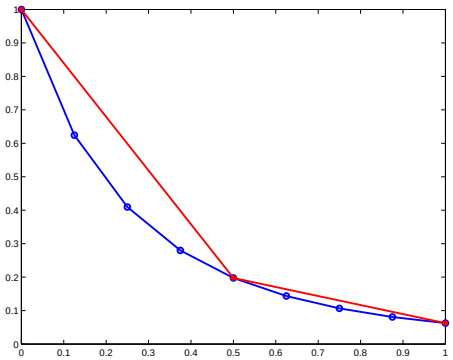


Figure 4.17: Piecewise linear functions on a coarse ($h_c = \frac{1}{2}$) and fine ($h_f = \frac{1}{8}$) grid.

proaches but use straightforward projections and monitor the resulting increase or loss of mass, depending on the (convexity or concavity of) the density function.

In Figure 4.18, we display the relative change of mass at time t (in percent) which is computed by

$$e(t) = \frac{m(t) - m(0)}{m(0)} \cdot 100 \ [\%]. \tag{4.19}$$

Given an initial density distribution $f(0,t)$, we first compute a mass preserving projection onto the space $\mathcal{S}$ of piecewise constant functions with function values C_i on $(x_i^g, x_{i+1}^g]$ satisfying

$$\int_{x_i^g}^{x_{i+1}^g} C_i x dx = \int_{x_i^g}^{x_{i+1}^g} x f(0,x) dx, \qquad i = 0, \ldots, N-1. \tag{4.20}$$

Figure 4.18 shows the change of mass for different nested grid configurations $(L = 2, n)$. As explained through Figure 4.17, the error in mass occurs through projections outside of the domain $(0, 2^{-L}]$ of the finest grid G_L, i.e., for grid sections corresponding to levels $\ell < L$ in which (piecewise linear) functions on a fine grid are approximated through functions that are piecewise linear on a coarser grid. Piecewise linear interpolation converges quadratically in the grid width, i.e.,

$$\|f - s\|_\infty \leq \frac{1}{8} h^2 \|f''\|_\infty$$

for any $f \in C^2[a, b]$ and its piecewise linear interpolant s on a uniform grid of width h, so we expect quadratic convergence which we indeed observe in Figure 4.18 (left): As we increase n by a factor of 2 (i.e., decrease h by a factor of $1/2$), the mass error decreases by a factor of $1/4$.

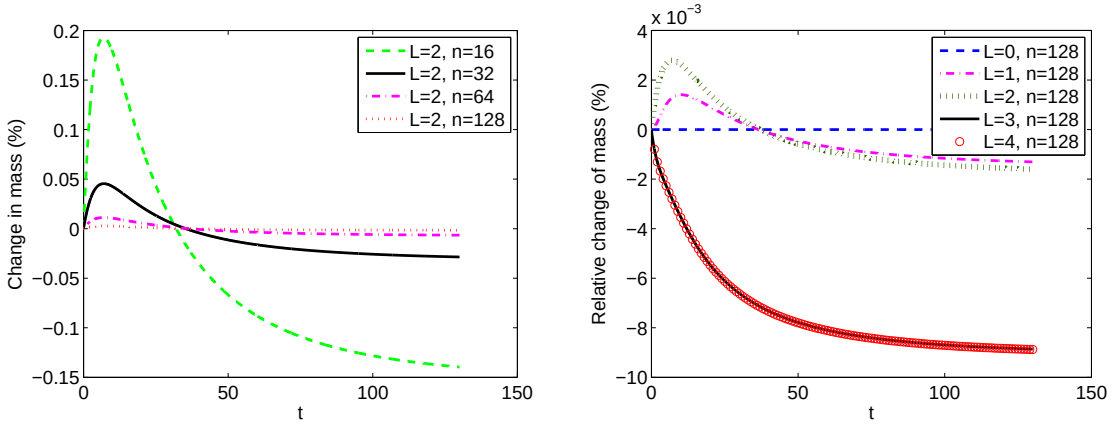


Figure 4.18: Change of mass over the time interval $t \in [0, 130]$ $(dt = 0.01)$ for the Brownian kernel using a mass preserving projection (4.20) of the initial distribution $f_1(0, x)$.

Figure 4.18 (right) shows the change of mass for fixed n and an increasing number of refinement levels L. On a uniform grid $(L = 0)$, the mass is preserved. As we add more levels, the mass error increases although we have additional grid points. The reason is once more that the exact convolution of a piecewise constant function is piecewise linear on the finest grid. For larger L, the size h_L of the finest grid decreases, which leads to an increasingly worse approximation on the coarse part of the nested grid. We attribute the fundamental change of the error curve when changing from $L = 2$ to $L = 3$ to the fact that the initial distribution $f(0, x) = e^{-1000(x-0.125)^2}$ accumulates a major portion of its mass on $(0, 0.25]$ which is included in the supports of the grids $\mathcal{G}_\ell$, $0 \leq \ell \leq 2$ and is hence covered by an

equidistant grid which is no longer the case for $\ell \geq 3$. However, a further increase of L does not yield any visible changes any more. This is once more due to the choice of initial density distribution which satisfies $|f(0,x)|_{\infty,(0,2^{-4}]} = f(0,2^{-4}) \leq 0.021$ as well as $|f'(0,x)|_{\infty,(0,2^{-4})} \leq f'(0,2^{-4}) \leq 2.52$, i.e., a finer resolution on $(0,2^{-L}]$ does not promise much difference in the computed results.

In both plots of Figures 4.18, the change in mass eventually tends to zero as t increases (i.e., $m'(t) \overset{t\to\infty}{\longrightarrow} 0$) which results from the fact that we consider an aggregation process with a maximum possible particle size: After a while, all particles have reached a size sufficiently large that they cannot aggregate any more.

Chapter 5

Evaluation of univariate aggregation integrals with high order polynomials on nested grids refined toward an arbitrary point

In this chapter we discuss the evaluation of aggregation integrals through approximating the density function by piecewise polynomials of order $p \geq 1$ on nested grids refined toward an arbitrary point. We discuss algorithms for the efficient evaluation of the source term (1.3) in sections 5.1, 5.2, 5.3 and sink term (1.6) in section 5.4. Section 5.5 provides numerical results, leading to (heuristic) criteria for the choice of polynomial degree and grid settings. The grids, basis functions and spaces used in this chapter are discussed in subsection 3.2.2.

5.1 Source term

We approximate the density distribution f through a piecewise polynomial and the factors $a_i(\cdot),\ b_i(\cdot)$ of the separable approximation of the kernel κ through piecewise constants functions so that the evaluation of the source integral is reduced to the convolution $\psi * \varphi$ of two functions $\psi, \varphi \in \mathcal{S}$ (3.42) for a nested grid $\mathcal{G}$ (3.37) refined toward an arbitrary point. Here as well, we chose a piecewise constant approximation of kernel factors $a_i(\cdot),\ b_i(\cdot)$ to ensure that functions φ, ψ belong to the same functions space $\mathcal{S}$, that f belongs to. Higher order approximation of kernel factors is also possible, but it would require a different treatment for spaces to which f and φ, ψ belong to.

Using the levelwise decomposition (3.45) and the symmetry of the convolution, a convolution $\psi * \varphi$ can be written as

$$\psi * \varphi \;=\; \sum_{\ell,\ell'=0}^{L} \psi_{\ell'} * \varphi_{\ell} \;=\; \sum_{0\le\ell'\le\ell\le L} \psi_{\ell'} * \varphi_{\ell} + \sum_{0\le\ell<\ell'\le L} \varphi_{\ell} * \psi_{\ell'}. \tag{5.1}$$

Hence, we only need to consider the convolution $\psi_{\ell'} * \varphi_{\ell}$ for $\ell' \le \ell$ since we obtain the second sum of convolutions by interchanging functions φ and ψ. (Since equality $\ell = \ell'$ does not occur in $\ell < \ell'$, we divide its contribution equally between the two sums, later in this section we provide further detail.)

Our goal is to compute the L_2-orthogonal projection $\omega := P(\varphi * \psi)$ onto the subspace $\mathcal{S} \subset L_2(\mathbb{R})$. Toward this goal, we compute L_2-orthogonal projections

$$\omega_{\ell''} = P_{\ell''}(\psi_{\ell'} * \varphi_{\ell}) \in \mathcal{S}_{\ell''}, \qquad \ell'' = 0, \dots, L, \tag{5.2}$$

where $\psi_{\ell'} \in \mathcal{S}_{\ell'}$ and $\varphi_{\ell} \in \mathcal{S}_{\ell}$.

Since the exact convolution $\psi_{\ell'} * \varphi_{\ell}$ typically lies in a much higher dimensional space than $\mathcal{S}_{\ell''}$, especially when $\ell'' \le \ell' \ll \ell$, its computation and subsequent projection are computationally inefficient. This will be avoided by projecting φ_{ℓ} to a coarser space before the convolution.

The computation of the projection $P_{\ell''}(\sum_{\ell'\le\ell} \psi_{\ell'} * \varphi_{\ell})$ (5.2) is split into the three parts

$$\overbrace{P_{\ell''}\Big(\sum_{\ell''\le\ell'\le\ell} \psi_{\ell'} * \varphi_{\ell}\Big)}^{=:\sum_1} + \overbrace{P_{\ell''}\Big(\sum_{\ell'<\ell''\le\ell} \psi_{\ell'} * \varphi_{\ell}\Big)}^{=:\sum_2} + \overbrace{P_{\ell''}\Big(\sum_{\ell'\le\ell<\ell''} \psi_{\ell'} * \varphi_{\ell}\Big)}^{=:\sum_3}. \tag{5.3}$$

In this chapter we consider nested grids refined toward an arbitrary point as illustrated in Figure 3.7. If the grid is refined toward the origin (Figure 3.8), only the first sum has to be considered in view of the following lemma.

Lemma 2. *If the grid is defined toward the origin, there holds $\sum_2 = \sum_3 = 0$ for the sums defined in (5.3).*

Proof. Assume that the grid is refined toward the origin, i.e., $s_{\ell} = 0\,\forall \ell \in \{0, \dots, L\}$ for the shifts (3.33) marking the start of the refined grids of level ℓ. A function $\varphi_{\ell''} \in \mathcal{S}_{\ell''}$ satisfies $\operatorname{supp}(\varphi_{\ell''}) \subset \left(\left(\frac{1}{2}\right)^{\ell''+1}, \left(\frac{1}{2}\right)^{\ell''}\right]$ for $\ell = 0, \dots, L-1$ and $\operatorname{supp}(\varphi_L) \subset \left(0, \left(\frac{1}{2}\right)^{L}\right]$ whereas the convolution of two functions $\varphi_{\ell} \in \mathcal{S}_{\ell}, \psi_{\ell'} \in \mathcal{S}_{\ell'}$ satisfies $\operatorname{supp}(\varphi_{\ell} * \psi_{\ell'}) \subset \left(\left(\frac{1}{2}\right)^{\ell+1} + \left(\frac{1}{2}\right)^{\ell'+1}, \; \left(\frac{1}{2}\right)^{\ell} + \left(\frac{1}{2}\right)^{\ell'}\right]$. Hence, $\ell' < \ell''$ implies

$\left(\frac{1}{2}\right)^{\ell''} \leq \left(\frac{1}{2}\right)^{\ell'+1} < \left(\frac{1}{2}\right)^{\ell+1} + \left(\frac{1}{2}\right)^{\ell'+1}$ and therefore $\mathrm{supp}(\varphi_{\ell''}) \cap \mathrm{supp}(\varphi_\ell * \psi_{\ell'}) = \emptyset$ which in turn leads to $\sum_2 = \sum_3 = 0$ if $\ell' < \ell''$. □

Before discussing the technical details of the computation of the sums $\sum_1$ $(\ell'' \leq \ell' \leq \ell)$, $\sum_2$ $(\ell' < \ell'' \leq \ell)$ and $\sum_3$ $(\ell' \leq \ell < \ell'')$, we provide two illustrative examples for linear $(p = 1)$ basis functions on a nested grid with $n = 4$ initial intervals, refined $L = 2$ times toward the interval midpoint 0.5, implying shifts $s_1 = s_2 = 1$. The plots in the middle of the top and bottom rows of Figure 5.1 show the exact convolution (piecewise cubic on $\mathcal{M}_1^I$) of the respective two piecewise linear functions shown in the left plots. The right plots show the projections of these exact convolutions to piecewise linear functions on grid levels $\ell'' = 0$ (black, dash-dot) $\ell'' = 1$ (red, dashed) and $\ell'' = 2$ (green, dash-dot).

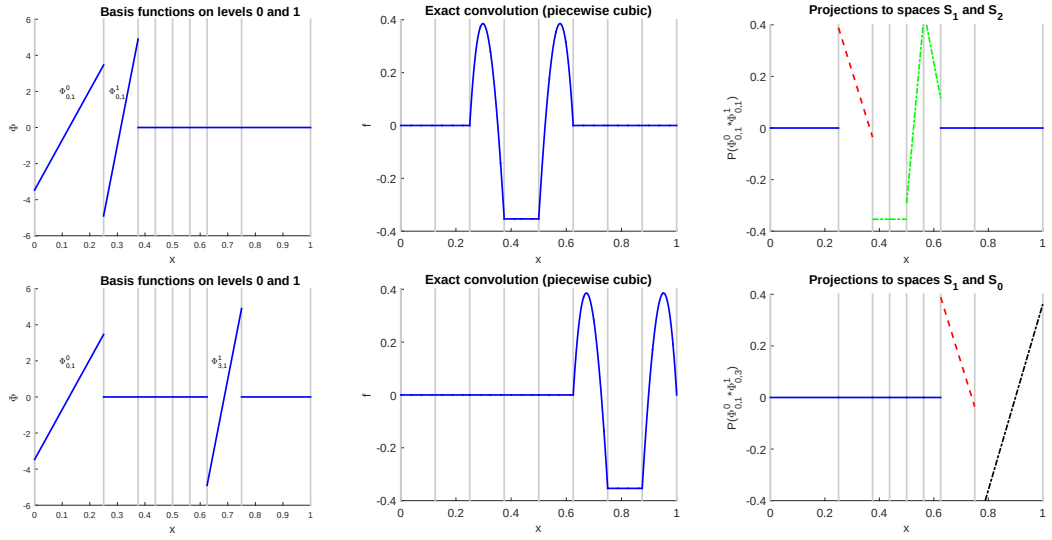


Figure 5.1: Nested grid with $n = 4, L = 2, s_1 = s_2 = 1$. Top: piecewise linear basis functions $\Phi_{0,1}^0$ and $\Phi_{0,1}^1$ (left), their piecewise cubic exact convolution (middle) and its projection onto the nested grid (right). Bottom: piecewise linear basis functions $\Phi_{0,1}^0$ and $\Phi_{3,1}^1$ (left), their piecewise cubic exact convolution (middle) and its projection onto the nested grid (right).

In this section, we reformulate algorithms developed in [30], [34] to prepare efficient evaluation of each of three sums discussed in (5.3). In order to reduce the computational complexity and apply FFT, all three sums are reformulated in terms of discrete convolutions involving vectors V and Γ in equations (5.7), (5.11) and (5.20), respectively, where V and Γ differ for the three cases.

The technical details of this and subsequent sections will lead to the following

algorithms:

- Algorithm 11 computes the coefficients of V and returns them as ψ (5.7), $\tilde{\psi}$ (5.11) or $\hat{\psi}$ (5.20), respectively.

- Algorithms 12 and 13 provide auxiliary coefficients for the computation of the entries of Γ.

- Algorithm 14 uses these subroutines and computes the desired sums $\sum_1, \sum_2, \sum_3$. It turns out that these vary only slightly from one another so that we can use the same algorithm after adjusting some of its input parameters (see Table 5.1).

- Algorithm 15 combines the computations of all three sums for both $\ell \leq \ell'$ and $\ell' \leq \ell$. The case $\ell = \ell'$ occurs in $\sum_1$ as well as in $\sum_3$. In (5.7) ($\sum_1$), the summation over ℓ occurs in Γ-coefficients (see 5.25) and the case $\ell = \ell'$ affects the computation of the convolution (5.7) through the coefficient c_2 (see Table 5.1 and further explanations in subsection 5.2). In (5.20) ($\sum_3$), the case $\ell = \ell'$ occurs in $\hat{\psi}$ (see (5.19) along with its explanation). In addition, Algorithm 15 performs projections (5.8) and prolongations (5.14) necessary for each case.

- Finally, Algorithm 16 shows the implementation of the computation of the sink term.

In the algorithms, expressions in the form "for $(\ell, i, k) \in \{0, \dots, L\} \times \{0, \dots, p\} \times \{0, \dots, n-1\}$ do" (e.g. line 2 of Algorithm 1) abbreviate nested for-loops and imply an order for the traversal through the indices starting from the first index of the given set and ending with its last index.

Computation of $\sum_1$ ($\ell'' \leq \ell' \leq \ell$)

In this part, we discuss the computation of the first sum $\sum_1$ in (5.3),

$$\omega_{\ell''} = P_{\ell''}(\psi_{\ell'} * \varphi_\ell) = \sum_{i \in \mathcal{I}_{\ell''}} \sum_{\alpha=0}^{p} \omega_{i,\alpha}^{\ell''} \Phi_{i,\alpha}^{\ell''} \in \mathcal{S}_{\ell''} \subset \mathcal{S}_{\ell''}^I, \qquad \text{for } \ell'' \leq \ell' \leq \ell.$$

Using the representations (3.45) with coefficients (3.47), we may write

$$
\begin{aligned}
\omega_{i,\alpha}^{\ell''} &= \int (\psi_{\ell'} * \varphi_\ell)(x)\ \Phi_{i,\alpha}^{\ell''}(x)\ dx \\
&= \sum_{\substack{j\in\mathcal{I}_{\ell'}\\ k\in\mathcal{I}_\ell}} \sum_{\beta,\chi=0}^{p} \psi_{j,\beta}^{\ell'}\varphi_{k,\chi}^{\ell} \underbrace{\int\int \Phi_{i,\alpha}^{\ell''}(x)\ \Phi_{j,\beta}^{\ell'}(y)\Phi_{k,\chi}^{\ell}(x-y)\ dxdy}_{=:\gamma_{(i,\alpha),(j,\beta),(k,\chi)}^{\ell'',\ell',\ell}} \\
&= \sum_{\substack{j\in\mathcal{I}_{\ell'}\\ k\in\mathcal{I}_\ell}} \sum_{\beta,\chi=0}^{p} \psi_{j,\beta}^{\ell'}\varphi_{k,\chi}^{\ell}\gamma_{(i,\alpha),(j,\beta),(k,\chi)}^{\ell'',\ell',\ell}, \qquad (5.4)
\end{aligned}
$$

i.e., for levels ℓ, ℓ', ℓ'' and integers i, j, k we define γ-coefficients

$$
\gamma_{(i,\alpha),(j,\beta),(k,\chi)}^{\ell'',\ell',\ell} := \int\int \Phi_{i,\alpha}^{\ell''}(x)\Phi_{j,\beta}^{\ell'}(y)\Phi_{k,\chi}^{\ell}(x-y)\ dxdy.
$$

For the case $\ell \geq \max(\ell', \ell'')$, the shift property (3.43) can be used to show that there holds ([34], Lemma C.2)

$$
\gamma_{(i,\alpha),(j,\beta),(k,\chi)}^{\ell'',\ell',\ell} = \gamma_{k-i2^{\ell-\ell''}+j2^{\ell-\ell'},(\alpha,\beta,\chi)}^{\ell'',\ell',\ell}
$$

for the simplified γ-coefficients

$$
\gamma_{\nu,(\alpha,\beta,\chi)}^{\ell'',\ell',\ell} := \int\int \Phi_{0,\alpha}^{\ell''}(x)\Phi_{0,\beta}^{\ell'}(y)\Phi_{\nu,\chi}^{\ell}(x-y)\ dxdy. \qquad (5.5)
$$

According to Lemma C.3 [34], we only need simplified coefficients for $\nu = 0$, $\ell'' = \ell' = \ell = 0$, i.e., $\gamma_{0,(\alpha,\beta,\chi)}^{0,0,0}$. Details on the computation of these simplified γ-coefficients are given in [34], explicit values for different index combinations in Appendix A.

For $\ell'' = \ell'$ and $i \in \mathcal{I}_{\ell'}$, equation (5.4) can be written as

$$
\begin{aligned}
\omega_{i,\alpha}^{\ell'} &= \sum_{\substack{j\in\mathcal{I}_{\ell'}\\ k\in\mathcal{I}_\ell}} \sum_{\beta,\chi=0}^{p} \psi_{j,\beta}^{\ell'}\varphi_{k,\chi}^{\ell}\gamma_{(i,\alpha),(j,\beta),(k,\chi)}^{\ell',\ell',\ell} \\
&= \sum_{j\in\mathcal{I}_{\ell'}} \sum_{\beta=0}^{p} \psi_{j,\beta}^{\ell'} \underbrace{\sum_{k\in\mathcal{I}_\ell} \sum_{\chi=0}^{p} \varphi_{k,\chi}^{\ell}\gamma_{k-(i-j)2^{\ell-\ell'},(\alpha,\beta,\chi)}^{\ell',\ell',\ell}}_{=:\Gamma_{i-j,(\alpha,\beta)}^{\ell',\ell}} \qquad (5.6) \\
&= \sum_{\beta=0}^{p} \sum_{j\in\mathcal{I}_{\ell'}} \psi_{j,\beta}^{\ell'}\Gamma_{i-j,(\alpha,\beta)}^{\ell',\ell}
\end{aligned}
$$

where we introduced Γ-coefficients

$$\Gamma^{\ell',\ell}_{i,(\alpha,\beta)} := \sum_{k\in\mathcal{I}_\ell}\sum_{\chi=0}^{p} \varphi^{\ell}_{k,\chi}\gamma^{\ell',\ell',\ell}_{k-i2^{\ell-\ell'},(\alpha,\beta,\chi)}, \qquad i\in\{0,\dots 2^{\ell'-\ell}n\}.$$

For $\ell' < \ell$, the functions in the convolution $\psi_{\ell'} * \varphi_\ell$ belong to levels with different interval sizes. Hence, following [34], Lemma 3.1., the Γ-coefficients are used to transport φ_ℓ to the coarser level ℓ'.

Introducing the vectors

$$\begin{aligned}\psi_{\ell',\beta} &:= (\psi^{\ell'}_{j,\beta})_{j\in\mathcal{I}_{\ell'}},\\ \Gamma_{\ell',\ell(\alpha,\beta)} &:= (\Gamma^{\ell',\ell}_{k,(\alpha,\beta)})_{k\in\mathcal{I}_\ell},\\ \omega_{\ell',\alpha} &:= (\omega^{\ell'}_{i,\alpha})_{i\in\mathcal{I}_{\ell'}},\end{aligned}$$

equation (5.6) turns into

$$\omega_{\ell',\alpha} = \sum_{\beta=0}^{p} \psi_{\ell',\beta} * \Gamma_{\ell',\ell,(\alpha,\beta)}, \qquad 0\le\ell'\le\ell, 0\le\alpha\le p. \tag{5.7}$$

Therefore, for $\ell'' = \ell'$, the sum $\omega_{\ell'',\alpha}$ in (5.7) is a convolution sum which by means of FFT can be computed in almost linear (instead of quadratic) complexity. Its implementation is given in Algorithm 14.

For computing $\omega_{\ell'',\alpha}$ on coarser levels $\ell'' = \ell'-1,\dots,0$, we use a (mass preserving) projection $\mathcal{R}_{\ell'-1} : \mathcal{S}^I_{\ell'} \to \mathcal{S}^I_{\ell'-1}$ that maps a function $\omega_{\ell'} \in \mathcal{S}^I_{\ell'}$ to a coarser space $\mathcal{S}^I_{\ell'-1} \subset \mathcal{S}^I_{\ell'}$. The respective coarsening operator $\mathcal{R}_{\ell-1}$ is derived in [34] and can be expressed through

$$\omega^{\ell-1}_{i,\alpha} = \sum_{\chi=0}^{\alpha} \xi_{\alpha,\chi}((-1)^{\chi+\alpha}\omega^{\ell}_{2i,\chi} + \omega^{\ell}_{2i+1,\chi}). \tag{5.8}$$

The computation of the coefficients $\xi_{\alpha,\chi}$ is detailed in [34], explicit values are given in Appendix B. Further details on (5.8) will follow in section 5.3, its implementation results lines 13-16 of Algorithm 15.

Computation of $\sum_2$ ($\ell' < \ell'' \le \ell$)

The second sum $\sum_2$ in (5.3) can be written as

$$\omega_{\ell''} = P_{\ell''}\left(\sum_{\ell'<\ell''\le\ell} \psi_{\ell'} * \varphi_\ell\right) = P_{\ell''}\left(\sum_{\ell'=0}^{\ell''-1} \psi_{\ell'} * \sum_{\ell=\ell''}^{L} \varphi_\ell\right).$$

Since $\ell'' > \ell'$, there holds $\mathcal{S}^I_{\ell'} \subset \mathcal{S}^I_{\ell''}$. We introduce a (trivial) prolongation $\mathcal{P}^{\ell''}_{\ell'} : \mathcal{S}^I_{\ell'} \to \mathcal{S}^I_{\ell''}$. Representing functions $\psi_{\ell'}, \ell' = 0, \ldots, \ell'' - 1$, with respect to the basis of the finer level ℓ'' allows us to easily add them. Denoting $\tilde{\psi}_{\ell''} := \sum_{\ell'=0}^{\ell''-1} \mathcal{P}^{\ell''}_{\ell'}(\psi_{\ell'})$, we obtain

$$P_{\ell''}\left(\sum_{\ell'=0}^{\ell''-1} \psi_{\ell'} * \sum_{\ell=\ell''}^{L} \varphi_\ell\right) = P_{\ell''}\left(\sum_{\ell'=0}^{\ell''-1} \mathcal{P}^{\ell''}_{\ell'}(\psi_{\ell'}) * \sum_{\ell=\ell''}^{L} \varphi_\ell\right) = P_{\ell''}\left(\tilde{\psi}_{\ell''} * \sum_{\ell=\ell''}^{L} \varphi_\ell\right) \tag{5.9}$$

for $\ell' < \ell'' \leq \ell$ (in fact, (5.9) holds for all $1 < \ell'' \leq L$).

Using $\tilde{\psi}^{\ell''}_{j,\beta}$, $j \in \{0, \ldots, 2^\ell n - 1\}$ to denote the coefficients of $\tilde{\psi}_{\ell''}$ with respect to the basis $B_{\ell''}$ (3.44), similar simplifications as in equation (5.6) lead to

$$\omega^{\ell''}_{i,\alpha} = \sum_{\beta=0}^{p} \sum_{j=0}^{i} \tilde{\psi}^{\ell''}_{j,\beta} \Gamma^{\ell'',\ell}_{i-j,(\alpha,\beta)}, \qquad i \in \mathcal{I}_{\ell''}, \tag{5.10}$$

and thus

$$\omega_{\ell'',\alpha} = \sum_{\beta=0}^{p} \tilde{\psi}_{\ell'',\beta} * \Gamma_{\ell'',\ell,(\alpha,\beta)}, \qquad 1 < \ell'' \leq L,\ 0 \leq \alpha \leq p, \tag{5.11}$$

for vectors $\omega_{\ell'',\alpha}$, $\tilde{\psi}_{\ell'',\beta}$ defined analogously to $\omega_{\ell',\alpha}$, $\psi_{\ell',\beta}$ in the computation of $\sum_1$. We first compute the coefficients of $\tilde{\psi}_{\ell''}$,

$$\tilde{\psi}_{\ell''} = \sum_{\ell'=0}^{\ell''-1} \mathcal{P}^{\ell''}_{\ell'}(\psi_{\ell'}) = \mathcal{P}^{\ell''}_0(\psi_0) + \mathcal{P}^{\ell''}_1(\psi_1) + \cdots + \mathcal{P}^{\ell''}_{\ell''-1}(\psi_{\ell''-1}). \tag{5.12}$$

The coefficients of each term $\mathcal{P}^{\ell''}_{\ell'}(\psi_{\ell'})$ can be computed recursively in view of

$$\mathcal{P}^{\ell''}_{\ell'}(\psi_{\ell'}) = \mathcal{P}_{\ell''}(\mathcal{P}_{\ell''-1}(\ldots(\mathcal{P}_{\ell'+1}(\psi_{\ell'})))),$$

for prolongations $\mathcal{P}_{\ell''+1} : \mathcal{S}^I_{\ell''} \to \mathcal{S}^I_{\ell''+1}$ between consecutive levels. Then, equation (5.12) can be reformulated as

$$\begin{aligned}
\tilde{\psi}_{\ell''} &= \mathcal{P}_{\ell''}(\psi_{\ell''-1}) + \mathcal{P}_{\ell''}(\mathcal{P}_{\ell''-1}(\psi_{\ell''-2})) + \cdots + \mathcal{P}_{\ell''}(\mathcal{P}_{\ell''-1}(\ldots(\mathcal{P}_1(\psi_0)))) \\
&= \mathcal{P}_{\ell''}(\psi_{\ell''-1} + \underbrace{(\mathcal{P}_{\ell''-1}(\psi_{\ell''-2} + \ldots + \mathcal{P}_2(\psi_1 + \mathcal{P}_1(\psi_0)))))}_{\tilde{\psi}_{\ell''-1}}) \\
&= \mathcal{P}_{\ell''}(\psi_{\ell''-1} + \tilde{\psi}_{\ell''-1}),
\end{aligned} \tag{5.13}$$

allowing us to obtain $\tilde{\psi}_{\ell''}$ in optimal complexity. The prolongation $\mathcal{P}_{\ell''}$ is defined in terms of the coefficients in the representation with respect to the bases $B_{\ell+1}$ and B_ℓ (3.44) [34],

$$\psi_{2j,\beta}^{\ell+1} = \sum_{\alpha=\beta}^{p} \psi_{j,\alpha}^{\ell} \xi_{\alpha,\beta} (-1)^{\alpha+\beta}, \qquad \psi_{2j+1,\beta}^{\ell+1} = \sum_{\alpha=\beta}^{p} \psi_{j,\alpha}^{\ell} \xi_{\alpha,\beta}. \tag{5.14}$$

The computation according to (5.13) leads to the following steps which are implemented in Algorithm 11.

1. We first assign to the projected coefficients $\tilde{\psi}_{\ell''}$ the values of not-projected coefficients $\tilde{\psi}_{\ell''} := \psi_{\ell''}$ for $\ell'' = 0, \ldots, L$ (lines 5-9, Alg. 11). Note that only the first n coefficients $\psi_{j,\beta}^{\ell''}$ from equation (5.10) are necessary for the computation of $\omega_{i,\alpha}^{\ell''}$ with $i \in \mathcal{I}_{\ell''}$.

2. We apply the recursion formula (5.14) to compute $\tilde{\psi}_{\ell''} := \psi_{\ell''} + \mathcal{P}_{\ell''}(\psi_{\ell''-1} + \tilde{\psi}_{\ell''-1})$ (lines 10-12, Alg. 11).

3. We subtract the corresponding level to obtain the final result $\tilde{\psi}_{\ell''} := \mathcal{P}_{\ell''}(\psi_{\ell''-1} + \tilde{\psi}_{\ell''-1})$ (lines 13-17, Alg. 11).

Computation of $\sum_3$ ($\ell' \leq \ell < \ell''$)

Here, we discuss the computation of the third sum $\sum_3$ in (5.3),

$$\omega_{\ell''} = P_{\ell''} \left(\sum_{\ell' \leq \ell < \ell''} \psi_{\ell'} * \varphi_\ell \right) = \sum_{\ell' \leq \ell < \ell''} P_{\ell''} \left(\psi_{\ell'} * \varphi_\ell \right).$$

Due to $\ell' \leq \ell < \ell''$, $\psi_{\ell'} * \varphi_\ell$ is a piecewise polynomial of at most degree $2p+1$ on $\mathcal{G}_{\ell''}$. We first compute $\sum_{\ell' \leq \ell < \ell''} \mathcal{P}_\ell^{\ell''}(\psi_{\ell'} * \varphi_\ell)$. Note that unlike the computation of $\sum_2$, here $\mathcal{P}_{\ell''}^{\ell''}$ maps polynomials of degree $2p+1$ of level ℓ' to polynomials of degree $2p+1$ of level ℓ''. The formula (5.8) in the computation of $\sum_1$ to compute the polynomial of order α on level $\ell - 1$ requires only the first α polynomials of the higher level ℓ. In the computation of $\sum_3$, however, in formula (5.21) we apply the prolongation formula (5.14) (see section 5.3, Algorithm 15, lines 25-32), and in order to compute the polynomial of order β from level $\ell + 1$, we require up to the highest existing polynomial order (which in this case is $2p+1$) from level ℓ. The L_2-orthogonal projection is obtained by simply omitting the coefficients $\omega_{i,\alpha}^{\ell}$ with $\alpha > p$. We separate the sum $\sum_{\ell' \leq \ell < \ell''} \mathcal{P}_\ell^{\ell''}(\psi_{\ell'} * \varphi_\ell)$ into two parts, one for the

Algorithm 11 $(\psi, \tilde{\psi}, \hat{\psi}) = \text{Coefficients}\ (n,\ L,\ p,\ m,\ \xi,\ f,\ a)$;

Input:
- $n \in \mathbb{N}$; (number of intervals on coarse mesh)
- $L \in \mathbb{N}$; (number of nested refinements of grid)
- $p \in \mathbb{N}$; (degree of polynomial)
- $\tilde{s} = (\tilde{s}_\ell)_{0\le\ell\le L} \in \mathbb{R}^{L+1}$; (absolute shift)
- $\xi = (\xi_{i,j})_{\substack{0\le i\le p\\ 0\le j\le p}} \in \mathbb{R}^{(p+1)\times(p+1)}$; (constant coefficients, see (5.14))
- $f = (f^\ell_{i,j})_{\substack{0\le i<p\\ 0\le\ell\le L\\ 0\le j<n}} \in \mathbb{R}^{(p+1)\times(L+1)\times n}$; (coefficients of density function)
- $a = (a^\ell_i)_{\substack{0\le\ell\le L\\ 0\le i<n}} \in \mathbb{R}^{(L+1)\times n}$; (kernel factor, piecew. const. on $\mathcal{G}_\ell$)

Output:
- $\psi = (\psi^\ell_{i,j})_{\substack{0\le i<p\\ 0\le\ell\le L\\ 0\le j<n}} \in \mathbb{R}^{(p+1)\times(L+1)\times n}$; (used in $\sum_1$)
- $\tilde{\psi} = (\tilde{\psi}^\ell_{i,j})_{\substack{0\le i<p\\ 0\le\ell\le L\\ 0\le j<n}} \in \mathbb{R}^{(p+1)\times(L+1)\times n}$; (used in $\sum_2$, see (5.13))
- $\hat{\psi} = (\hat{\psi}^\ell_{i,j})_{\substack{0\le i<p\\ 0\le\ell<L\\ 0\le j<n}} \in \mathbb{R}^{(p+1)\times L\times n}$; (used in $\sum_3$, see (5.18))

$\tilde{\psi} := (0, \dots, 0)$;
for $(\ell, i, k) \in \{0, \dots, L\} \times \{0, \dots, p\} \times \{0, \dots, n-1\}$ **do**
  $\psi^\ell_{k,i} = f^\ell_{k,i} \cdot a^\ell_k$;
end for
for $(\ell, i, k) \in \{0, \dots, L-1\} \times \{0, \dots, p\} \times \{0, \dots, n-1\}$ **do**
  if $k - \tilde{s}_\ell \ge 0$ & $k - \tilde{s}_\ell < n$ **then**
    $\tilde{\psi}^\ell_{k,i} += \psi^\ell_{k-\tilde{s}_\ell,i}$;
  end if
end for
for $(\ell, i, j, k) \in \{1, \dots, L\} \times \{0, \dots, p\} \times \{0, \dots, p\} \times \{0, \dots, \frac{n}{2} - 1\}$ **do**
  $\tilde{\psi}^\ell_{2k,i} += \tilde{\psi}^{\ell-1}_{k,j}\xi_{j,i}(-1)^{i+j}$; $\tilde{\psi}^\ell_{2k+1,i} += \tilde{\psi}^{\ell-1}_{k,j}\xi_{j,i}$;
end for
for $(\ell, i, k) \in \{0, \dots, L-1\} \times \{0, \dots, p\} \times \{0, \dots, n-1\}$ **do**
  if $k - \tilde{s}_\ell \ge 0$ & $k - \tilde{s}_\ell < n$ **then**
    $\tilde{\psi}^\ell_{k,i} -= \psi^\ell_{k-\tilde{s}_\ell,i}$;
  end if
end for
for $(\ell, i, k) \in \{0, \dots, L-1\} \times \{0, \dots, p\} \times \{0, \dots, n-1\}$ **do**
  if $k - \tilde{s}_\ell \ge 0$ & $k - \tilde{s}_\ell < n$ **then**
    $\hat{\psi}^\ell_{k,i} = \tilde{\psi}^\ell_{k,i} + 0.5\psi^\ell_{k-\tilde{s}_\ell,i}$;
  end if
end for
return $(\psi, \tilde{\psi}, \hat{\psi})$

level $\ell = \ell'' - 1$ and one for all other levels $\ell = \ell', \dots, \ell'' - 2$, leading to

$$\tilde{\omega}_{\ell''} := \sum_{\ell' \le \ell < \ell''} \mathcal{P}_\ell^{\ell''}(\psi_{\ell'} * \varphi_\ell) \tag{5.15}$$

$$= \sum_{\ell'=0}^{\ell''-1} \mathcal{P}_{\ell''-1}^{\ell''}(\psi_{\ell'} * \varphi_{\ell''-1}) + \sum_{\ell' \le \ell < \ell''-1} \mathcal{P}_\ell^{\ell''}(\psi_{\ell'} * \varphi_\ell). \tag{5.16}$$

Since $\ell'' > \ell'$, similar to the computation of $\sum_2$ we can represent functions $\psi_{\ell'}$ and $\psi_{\ell'} * \varphi_\ell$ with respect to the basis of the finer level $\ell'' - 1$,

$$\tilde{\omega}_{\ell''} := \mathcal{P}_{\ell''-1}^{\ell''}\Bigg(\underbrace{\sum_{\ell'=0}^{\ell''-1} \mathcal{P}_{\ell'}^{\ell''-1}(\psi_{\ell'})}_{\hat{\psi}_{\ell''-1}} * \varphi_{\ell''-1}\Bigg) + \mathcal{P}_{\ell''-1}^{\ell''}\left(\sum_{\ell' \le \ell < \ell''-1} \mathcal{P}_\ell^{\ell''-1}(\psi_{\ell'} * \varphi_\ell)\right).$$

Denoting $\hat{\psi}_{\ell''-1} := \sum_{\ell'=0}^{\ell''-1} \mathcal{P}_{\ell'}^{\ell''-1}(\psi_{\ell'})$, we obtain the recursion formula

$$\tilde{\omega}_{\ell''} = \mathcal{P}_{\ell''-1}^{\ell''}\big(\underbrace{\hat{\psi}_{\ell''-1} * \varphi_{\ell''-1}}_{\hat{\omega}_{\ell''}}\big) + \mathcal{P}_{\ell''-1}^{\ell''}(\tilde{\omega}_{\ell''-1}). \tag{5.17}$$

The coefficients $\hat{\psi}_{\ell''-1} = \sum_{\ell'=0}^{\ell''-1} \mathcal{P}_{\ell'}^{\ell''-1}(\psi_{\ell'})$ can be computed from the coefficients $\tilde{\psi}_{\ell''}$ used in $\sum_2$ and coefficients $\psi_{\ell''}$ used in $\sum_1$,

$$\hat{\psi}_{\ell''-1} = \sum_{\ell'=0}^{\ell''-1} \mathcal{P}_{\ell'}^{\ell''-1}(\psi_{\ell'}) = \underbrace{\sum_{\ell'=0}^{\ell''-2} \mathcal{P}_{\ell'}^{\ell''-1}(\psi_{\ell'})}_{\ell'=0,\dots,\ell''-2<\ell''-1=\ell} + \underbrace{\mathcal{P}_{\ell''-1}^{\ell''-1}\psi_{\ell''-1}}_{\ell'=\ell''-1=\ell} = \tilde{\psi}_{\ell''-1} + \psi_{\ell''-1}. \tag{5.18}$$

In (5.15), therefore also in (5.18), the first sum corresponds to the case $\ell = \ell'' - 1 > \ell'$ and the second sum to the case $\ell' = \ell'' - 1 = \ell$. Thus to divide the contribution of $\ell = \ell'$ we will use

$$\hat{\psi}_{\ell''-1} = \tilde{\psi}_{\ell''-1} + 0.5\psi_{\ell''-1}, \tag{5.19}$$

(see Algorithm 11, lines 18-22).

Writing the convolution in (5.17) as

$$\hat{\omega}_{\ell''} := \hat{\psi}_{\ell''-1} * \varphi_{\ell''-1} = \mathcal{P}_{\ell''-1}^{\ell''-1}(\hat{\psi}_{\ell''-1} * \varphi_{\ell''-1}),$$

we obtain similar to the computation of $\sum_1$ and $\sum_2$

$$\hat{\omega}_{i,\alpha}^{\ell''} = \sum_{\beta=0}^{p} \sum_{j=0}^{i} \hat{\psi}_{j,\beta}^{\ell''-1} \Gamma_{i-j,(\alpha,\beta)}^{\ell''-1,\ell''-1}, \qquad 0 \le \alpha \le 2p+1,$$

$$\hat{\omega}_{\ell'',\alpha} = \sum_{\beta=0}^{p} \hat{\psi}_{\ell''-1,\beta} * \Gamma_{\ell''-1,\ell''-1,(\alpha,\beta)}, \qquad 1 \le \ell'' \le L,\ 0 \le \alpha \le 2p+1. \tag{5.20}$$

The recursion formula (5.17) can be rewritten as

$$\tilde{\omega}_{\ell''} = \mathcal{P}_{\ell''}(\hat{\omega}_{\ell''-1}) + \mathcal{P}_{\ell''}(\tilde{\omega}_{\ell''-1}). \tag{5.21}$$

The prolongation (5.17) is implemented and explained further in section 5.3, Algorithm 15, lines 25-32.
Omitting the last $p+1$ coefficients of $\tilde{\omega}_{i,\alpha}^{\ell''}$ we obtain the final result

$$\omega_{i,\alpha}^{\ell''} = P_{\ell''}(\tilde{\omega}_{i,\alpha}^{\ell''}) \qquad 0 \le \alpha \le p. \tag{5.22}$$

Γ-coefficients

Here, we revisit the formulas given in [34], Lemma D.2, for the $\Gamma_{i,(\alpha,\beta)}^{\ell',\ell}$-coefficients. For $\ell' = \ell$, there holds

$$\Gamma_{i,(\alpha,\beta)}^{\ell,\ell} = \sum_{\chi=0}^{p} \left(\varphi_{i,\chi}^{\ell} + (-1)^{\alpha+\beta+\chi} \varphi_{i-1,\chi}^{\ell}\right) \gamma_{0,(\alpha,\beta,\chi)}^{\ell,\ell,\ell},\ 0 \le i \le n,\ 0 \le \alpha,\ 0 \le \beta \le p. \tag{5.23}$$

These coefficients are computed in Algorithm 12, and we refer to them as "diagonal" Γ-coefficients and denote them using a left subscript "d", i.e., ${}_{\mathrm{d}}\Gamma_{i,(\alpha,\beta)}^{\ell} := \Gamma_{i,(\alpha,\beta)}^{\ell,\ell}$.

For the remaining levels $\ell' = \ell - 1, \ldots, 0$, the following recursion formula is used

$$\begin{aligned} \Gamma_{i,(\alpha,\beta)}^{\ell',\ell} \;=\; & \sum_{p=0}^{\alpha} \sum_{q=0}^{\beta} \xi_{\alpha,p} \xi_{\beta,q} \Big((-1)^{\alpha+p} \Gamma_{2i-1,(p,q)}^{\ell'+1,\ell} + (1 + (-1)^{\alpha+\beta+p+q}) \Gamma_{2i,(p,q)}^{\ell'+1,\ell} \\ & + (-1)^{\alpha+\beta} \Gamma_{2i+1,(p,q)}^{\ell'+1,\ell} \Big), \qquad 0 \le i \le 2^{\ell'-\ell} n. \end{aligned} \tag{5.24}$$

In $\Gamma_{i,(\alpha,\beta)}^{\ell',\ell}$, the index ℓ denotes the contribution of φ_ℓ on the level ℓ'. For $\ell' \le \ell \le L$, the values $\Gamma_{i,(\alpha,\beta)}^{\ell',\ell}$ can be collected in

$$\Gamma_{i,(\alpha,\beta)}^{\ell'} := \sum_{\ell=\ell'}^{L} \Gamma_{i,(\alpha,\beta)}^{\ell',\ell}. \tag{5.25}$$

Algorithm 12 ${}_{\mathrm{d}}\Gamma =$ GammaDiagonal $(n,\ L,\ p,\ \gamma,\ f,\ b)$;

Input:	$n \in \mathbb{N}$;	(number of intervals on coarse mesh)
	$L \in \mathbb{N}$;	(number of nested refinements of grid)
	$p \in \mathbb{N}$;	(degree of polynomial)
	$\gamma \in \mathbb{R}^{(p+1)\times(p+1)\times(p+1)}$	(constant coefficients)
	$f = (f_{i,j}^{\ell})_{\substack{0\le i\le p \\ 0\le \ell\le L \\ 0\le j<n}} \in \mathbb{R}^{p\times(L+1)\times n}$;	(coefficients of density function)
	$b = (b_i^{\ell})_{\substack{0\le \ell\le L \\ 0\le i<n}} \in \mathbb{R}^{(L+1)\times n}$;	(kernel factor, piecew. const. on $\mathcal{G}_\ell$)
Output:	${}_{\mathrm{d}}\Gamma \in \mathbb{R}^{(p+1)\times(p+1)\times(L+1)\times(n+1)}$;	(see (5.23))

1: **for** $(\ell, i, k) \in \{0, \dots, L\} \times \{0, \dots, p\} \times \{0, \dots, n-1\}$ **do**
2: $\quad \varphi_{k,i}^{\ell} = f_{k,i}^{\ell} \cdot b_k^{\ell}$;
3: **end for**
4: ${}_{\mathrm{d}}\Gamma := (0, \dots, 0)$; $\qquad ({}_{\mathrm{d}}\Gamma \in \mathbb{R}^{(p+1)\times(p+1)\times(L+1)\times(n+1)};)$
5: **for** $(m, k, j, \ell) \in \{0, \dots, p\}^3 \times \{0, \dots, L\}$ **do**
6: $\quad {}_{\mathrm{d}}\Gamma_{0,(k,j)}^{\ell} += \varphi_{0,m}^{\ell}\gamma_{k,j,m}\sqrt{h_\ell}$; $\quad {}_{\mathrm{d}}\Gamma_{n,(k,j)}^{\ell} += (-1^{m+k+j})\varphi_{n-1,m}^{\ell}\gamma_{k,j,m}\sqrt{h_\ell}$;
7: $\quad$ **for** $i = 1 : n-1$ **do**
8: $\quad\quad {}_{\mathrm{d}}\Gamma_{i,(k,j)}^{\ell} += (\varphi_{i,m}^{\ell} + -1^{m+k+j}\varphi_{i-1,m}^{\ell})\gamma_{k,j,m}\sqrt{h_\ell}$;
9: $\quad$ **end for**
10: **end for**
11: **return** $({}_{\mathrm{d}}\Gamma)$;

In Algorithm 13, we compute these coefficients and refer to them as "reduced" coefficients since they are collected and defined only through one (instead of two) superscript level ℓ'. We use a left subscript "r" to denote ${}_{\mathrm{r}}\Gamma^{\ell'}_{i,(\alpha,\beta)} := \Gamma^{\ell'}_{i,(\alpha,\beta)} = \sum_{\ell=\ell'}^{L} \Gamma^{\ell',\ell}_{i,(\alpha,\beta)}$ (5.24).

Formula (5.24) along with (5.25) lead us to

$$\begin{aligned}\Gamma^{\ell'}_{i,(\alpha,\beta)} &= \sum_{p=0}^{\alpha}\sum_{q=0}^{\beta}\xi_{\alpha,p}\xi_{\beta,q}\Big((-1)^{\alpha+p}\Gamma^{\ell'+1}_{2i-1,(p,q)} + (1+(-1)^{\alpha+\beta+p+q})\Gamma^{\ell'+1}_{2i,(p,q)} \\ &\quad + (-1)^{\alpha+\beta}\Gamma^{\ell'+1}_{2i+1,(p,q)}\Big) := F_{i+s_{\ell'+1}}\left({}_{\mathrm{r}}\Gamma^{\ell+1}_{2i-1}, {}_{\mathrm{r}}\Gamma^{\ell+1}_{2i}, {}_{\mathrm{r}}\Gamma^{\ell+1}_{2i+1}\right), \qquad (5.26)\end{aligned}$$

where the F- coefficients are explained further in this subsection.
For fixed α, β, we omit the (α,β)-subscript in the Γ-coefficients for n intervals and L levels and define the matrix

$$\begin{pmatrix} {}_{\mathrm{r}}\Gamma^L_0 & \dots & {}_{\mathrm{r}}\Gamma^L_n \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ {}_{\mathrm{r}}\Gamma^0_0 & \dots & {}_{\mathrm{r}}\Gamma^0_n \end{pmatrix} = \begin{pmatrix} {}_{\mathrm{d}}\Gamma^L_0 & \dots & {}_{\mathrm{d}}\Gamma^L_n \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ {}_{\mathrm{d}}\Gamma^0_0 & \dots & {}_{\mathrm{d}}\Gamma^0_n \end{pmatrix} +$$

$$\begin{pmatrix} 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & F_{s_L}({}_{\mathrm{r}}\Gamma^L_0, {}_{\mathrm{r}}\Gamma^L_1) & F_{s_L+1}({}_{\mathrm{r}}\Gamma^L_1, {}_{\mathrm{r}}\Gamma^L_2, {}_{\mathrm{r}}\Gamma^L_3) & \dots & F_{s_L+\frac{n}{2}}({}_{\mathrm{r}}\Gamma^L_{n-1}, {}_{\mathrm{r}}\Gamma^L_n) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & F_{s_1}({}_{\mathrm{r}}\Gamma^1_0, {}_{\mathrm{r}}\Gamma^1_1) & F_{s_1+1}({}_{\mathrm{r}}\Gamma^1_1, {}_{\mathrm{r}}\Gamma^1_2, {}_{\mathrm{r}}\Gamma^1_3) & \dots & F_{s_1+\frac{n}{2}}({}_{\mathrm{r}}\Gamma^1_{n-1}, {}_{\mathrm{r}}\Gamma^1_n) & \dots & 0 \end{pmatrix}$$

with entries to be explained next. Since $\alpha, \beta \in \{0, \dots, p\}$, we have $(p+1)^2$ matrices of this form.

For the last (finest) level, the diagonal coefficients ${}_{\mathrm{d}}\Gamma$ coincide with the reduced coefficients ${}_{\mathrm{r}}\Gamma$ (as reflected through 0's in the first row of the last matrix and lines 3-5 in Algorithm 13). For the unrefined intervals $j = 0, \dots, s_\ell - 1$ and $j = s_\ell + \frac{n}{2} + 1, \dots, n$ (note that there are $(\dim(\mathcal{I}_\ell)+1)$ values), the diagonal coefficients ${}_{\mathrm{d}}\Gamma$ coincide with ${}_{\mathrm{r}}\Gamma$ as well (reflected through the remaining 0's in the last matrix and line 8, Alg. 13). For the remaining $\ell = 0, \dots, L-1$ rows and $j = s_\ell, \dots, s_\ell + \frac{n}{2}$ columns (refined intervals), the ${}_{\mathrm{r}}\Gamma$-coefficients are computed from the ${}_{\mathrm{d}}\Gamma$-coefficients of the given level ℓ (line 8 in Algorithm 13) and a function

$F_{i+s_{\ell+1}}({}_{\mathrm{r}}\Gamma^{\ell+1}_{2i-1}, {}_{\mathrm{r}}\Gamma^{\ell+1}_{2i}, {}_{\mathrm{r}}\Gamma^{\ell+1}_{2i+1})$, with $i = 0, \ldots, \frac{n}{2}$, taking as arguments three coefficients of the finer level $\ell+1$ from the formula (5.26) (lines 12-16 in Algorithm 13).

Algorithm 13 ${}_{\mathrm{r}}\Gamma =$ GammaReduced $(n,\ L,\ p,\ s,\ \xi,\ {}_{\mathrm{d}}\Gamma)$;

Input:	$n \in \mathbb{N}$;	(number of intervals on coarse mesh)
	$L \in \mathbb{N}$;	(number of nested refinements of grid)
	$p \in \mathbb{N}$;	(degree of polynomial)
	$s \in (s_\ell)_{0 \le \ell \le L} \in \mathbb{R}^{L+1}$;	(refinement shift)
	$\xi \in \mathbb{R}^{(p+1)\times(p+1)}$	(constant coefficients)
Output:	${}_{\mathrm{r}}\Gamma \in \mathbb{R}^{(p+1)\times(p+1)\times(L+1)\times(n+1)}$;	(see (5.24))

1: ${}_{\mathrm{d}}\Gamma =$ GammaDiagonal $(n,\ L,\ p,\ \gamma,\ f,\ b)$;
2: ${}_{\mathrm{r}}\Gamma := (0, \ldots, 0)$; $\quad ({}_{\mathrm{r}}\Gamma \in \mathbb{R}^{(p+1)\times(p+1)\times(L+1)\times(n+1)};)$
3: **for** $(i,j,k) \in \{0,\ldots,p\}^2 \times \{0,\ldots,n\}$ **do**
4: $\quad {}_{\mathrm{r}}\Gamma^L_{k,(i,j)} += {}_{\mathrm{d}}\Gamma^L_{k,(i,j)}$;
5: **end for**
6: **for** $(i,j,\ell) \in \{0,\ldots,p\}^2 \times \{L-1,\ldots,0\}$ **do**
7: $\quad$ **for** $k = 0 : n$ **do**
8: $\quad\quad {}_{\mathrm{r}}\Gamma^\ell_{k,(i,j)} = {}_{\mathrm{d}}\Gamma^\ell_{k,(i,j)}$;
9: $\quad$ **end for**
10: $\quad$ **for** $j_1 = 0 : j$ **do**
11: $\quad\quad$ **for** $i_1 = 0 : i$ **do**
12: $\quad\quad\quad {}_{\mathrm{r}}\Gamma^\ell_{s_{\ell+1},(i,j)} += \xi_{i,i_1}\xi_{j,j_1}((1+(-1)^{i+j+i_1+j_1})_{\mathrm{r}}\Gamma^{\ell+1}_{0,(i_1,j_1)} + (-1)^{j+j_1}{}_{\mathrm{r}}\Gamma^{\ell+1}_{1,(i_1,j_1)})$;
13:
14: $\quad\quad\quad {}_{\mathrm{r}}\Gamma^\ell_{s_{\ell+1}+\frac{n}{2},(i,j)} += \xi_{i,i_1}\xi_{j,j_1}((1+(-1)^{i+j+i_1+j_1})_{\mathrm{r}}\Gamma^{\ell+1}_{n-1,(i_1,j_1)} + (-1)^{i+i_1}{}_{\mathrm{r}}\Gamma^{\ell+1}_{n,(i_1,j_1)})$;
15: $\quad\quad\quad$ **for** $k = 1 : n/2 - 1$ **do**
16: $\quad\quad\quad\quad {}_{\mathrm{r}}\Gamma^\ell_{s_{\ell+1}+k,(i,j)} += \xi_{i,i_1}\xi_{j,j_1}((-1)^{i+i_1}{}_{\mathrm{r}}\Gamma^{\ell+1}_{2k-1,(i_1,j_1)}$
17: $\quad\quad\quad\quad\quad\quad +(1+(-1)^{i+j+i_1+j_1})_{\mathrm{r}}\Gamma^{\ell+1}_{2k,(i_1,j_1)} + (-1)^{j+j_1}{}_{\mathrm{r}}\Gamma^{\ell+1}_{2k+1,(i_1,j_1)})$;
18: $\quad\quad\quad$ **end for**
19: $\quad\quad$ **end for**
20: $\quad$ **end for**
21: **end for**
22: **return** $({}_{\mathrm{r}}\Gamma)$;

5.2 Summarized algorithm to compute the source aggregation integral

The convolution formulas (5.7), (5.11) and (5.20) from $\sum_1$, $\sum_2$ and $\sum_3$, respectively, vary only slightly from one another so that Algorithm 14 can accomplish these computations (using FFT, line 17) in all three cases by setting the parameters $x_1, x_2, c_1, c_2, c_3, v$ according to Table 5.1.
For the algorithm *FFT_convolution* of a fast convolution on an equidistant grid (used in line 17 Algorithm 14), we again refer to Algorith 3. In particular, given two vectors $\Gamma, V \in \mathbb{R}^n$, the algorithm

$$\omega = \text{FFT\_convolution}(n, \Gamma, V)$$

computes the convolution $\omega = \Gamma * V$, $\omega \in \mathbb{R}^n$, in $\mathcal{O}(n \log n)$ complexity.

Table 5.1: Input parameters for Algorithm 14 to compute the sums $\sum_1$, $\sum_2$ or $\sum_3$.

	$\sum_1$	$\sum_2$	$\sum_3$
$x_1=$	0	1	1
$x_2=$	p	p	$2p$
$c_1 =$	1	1	0
$c_2 =$	0.5	0	1
$c_3 =$	0	0	1
$v =$	ψ	$\tilde{\psi}$	$\hat{\psi}$

When computing the coefficients of $\sum_1$, we use ${}_{\mathrm{r}}\Gamma - 0.5{}_{\mathrm{d}}\Gamma$ to obtain symmetry for the case $\ell = \ell'$ that appears twice. In the computation on $\sum_2$, it always holds that $\ell' < \ell$ implying that we have only ${}_{\mathrm{r}}\Gamma$. And in the case $\sum_3$, we see in equation (5.20) that only ${}_{\mathrm{d}}\Gamma$ appears since the two levels appearing in Γ-coefficients are always equal.

5.3 Projections and prolongations

In this section, we introduce Algorithm 15 which performs the projection (5.8) for $\sum_1$ and the prolongation (5.21) for $\sum_2$. In addition, we combine the computation of all three cases to obtain the final result for the aggregation source integral. Since previous algorithms have been developed for $\ell' \leq \ell$, here we include also the case $\ell' \geq \ell$ and distinguish between these cases by using (left) subindices A and B. The additional subindices $1, 2$ and 3 correspond to the computation of $\sum_1$, $\sum_2$

Algorithm 14 $\omega = \text{Source}\ (n,\ L,\ p,\ v,\ {}_{\mathrm{d}}\Gamma,\ {}_{\mathrm{r}}\Gamma,\ x_1,\ x_2,\ c_1,\ c_2,\ c_3)$;

Input: $n \in \mathbb{N}$; (number of intervals on coarse mesh)
$L \in \mathbb{N}$; (number of nested refinements of grid)
$p \in \mathbb{N}$; (maximum degree of polynomial)

Output: $\omega = (\omega_{i,j}^{\ell})_{\substack{0\le i\le p\\ 0\le \ell\le L\\ 0\le j<n}} \in \mathbb{R}^{(p+1)\times(L+1)\times n}$; (coefficients of ω)

$\tilde{\psi} = \text{Coefficients}\ (n,\ L,\ p,\ m,\ \xi,\ f,\ a)$;
${}_{\mathrm{d}}\Gamma = \text{GammaDiagonal}\ (n,\ L,\ p,\ \gamma,\ f,\ b)$;
${}_{\mathrm{r}}\Gamma = \text{GammaReduced}\ (n,\ L,\ p,\ s,\ \xi,\ {}_{\mathrm{d}}\Gamma)$;
for $(\ell, i, j) \in \{x_1, \ldots, L\} \times \{0, \ldots, x_2\} \times \{0, \ldots, p\}$ **do**
  for $k = 0 : n-1$ **do**
    $a_k = v_{k,j}^{\ell - c_3}$;
  end for
  for $k = n : 2n-1$ **do**
    $a_k = 0$;
  end for
  for $k = 0 : n$ **do**
    $b_k = c_{1\mathrm{r}}\Gamma_{k,(i,j)}^{\ell} - c_{2\mathrm{d}}\Gamma_{k,(i,j)}^{\ell - c_3;}$
  end for
  for $k = n+1 : 2n-1$ **do**
    $b_k = 0$;
  end for
  $(r_0, \ldots, r_{2n-1}) = \text{FFT\_convolution}(2n, a, b)$;
  for $k = 0 : 2n-1$ **do**
    $\omega_{k,i}^{\ell} += r_k$;
  end for
end for
return (ω);

and $\sum_3$, respectively (lines 7-12). The lines 13-16 in Algorithm 15 implement the coarsening operator $\mathcal{R}_{\ell-1}$ according to the formula (5.8). The shift of a convolution s_ℓ^{c} (lines 14, 15) is defined as

$$s_\ell^{\mathrm{c}} = \tilde{s}_\ell - 2\tilde{s}_{\ell-1},$$

where $\tilde{s}_\ell$ is the absolute shift (3.34). s_ℓ^c is the shift between the starting points of the supports of functions defined on levels ℓ and $\ell-1$ for which a convolution is to be computed. The "if" condition on line 18 is included so that undefined values are not attempted to be accessed.

The coefficients $\hat{\omega}$ (5.20) are computed by calling Algorithm 14.

In lines 25-28 of Algorithm 15 we compute $\mathcal{P}_{\ell''}(\hat{\omega}_{\ell''-1})$ as given in equation (5.21), where we use prolongation $\mathcal{P}_{\ell''}$ according to (5.14). In lines 29-32, we apply the recursion (5.21) with $\mathcal{P}_{\ell''}(\tilde{\omega}_{\ell''-1})$. Finally, in lines 33-35, we set the values on refined intervals to zero, and in lines 36-38, we obtain the final result for the aggregation source integral.

5.4 Sink term

In this section, we discuss the evaluation of sink the aggregation integral (1.6) which, after a separable kernel expansion can be computed as the sum of M (kernel separation rank) terms of the form

$$Q_{\mathrm{sink}}(f)(x,t) = \psi(x) \int_0^1 \varphi(y)\, dy.$$

In previous sections we discussed the sink aggregation integral (1.5) securing mass conservation in (0,1]. Here, for simplicity, we considered the sink aggregation integral (1.6) computing the solution of the problem in (0,1] but with mass conservation holding in (0,2].

We compute the integral $\int_0^1 \varphi(y)dy$ using the representation (3.45),

$$\int_0^1 \varphi(y)dy = \sum_{\ell=0}^{L}\sum_{k=0}^{n} \varphi_{k,0}^{\ell} \underbrace{\int \Phi_{k,0}^{\ell}(x)dx}_{=\sqrt{h_\ell}} + \sum_{i=1}^{p}\sum_{\ell=0}^{L}\sum_{k=0}^{n} \varphi_{k,i}^{\ell} \underbrace{\int \Phi_{k,i}^{\ell}(x)dx}_{=0}. \tag{5.27}$$

Algorithm 15 ω = SourceProj $(p,\ n,\ L,\ \xi)$;

Input: $n \in \mathbb{N}$; (number of intervals on coarse mesh)
$L \in \mathbb{N}$; (number of nested refinements of grid)
$p \in \mathbb{N}$; (degree of polynomial)
$\xi \in \mathbb{R}^{(p+1)\times(p+1)}$; (constant coefficients)
Output: $\omega = (\omega_{i,j}^{\ell})_{\substack{0\le i\le p\\ 0\le \ell\le L\\ 0\le j<n}} \in \mathbb{R}^{(p+1)\times(L+1)\times n}$; (coefficients of ω)

$\tilde{\varphi}$ = Coefficients $(n,\ L\ ,p,\ m,\ \xi,\ f,\ a)$;
$\tilde{\psi}$ = Coefficients $(n,\ L,\ p,\ m,\ \xi,\ f,\ b)$;
${}_{\mathrm{d}}\Gamma_A$ = GammaDiagonal $(n,\ L,\ p,\ \gamma,\ f,\ b)$;
${}_{\mathrm{d}}\Gamma_B$ = GammaDiagonal $(n,\ L,\ p,\ \gamma,\ f,\ a)$;
${}_{\mathrm{r}}\Gamma_A$ = GammaReduced $(n,\ L,\ p,\ s,\ \xi,\ {}_{\mathrm{d}}\Gamma_A)$;
${}_{\mathrm{r}}\Gamma_B$ = GammaReduced $(n,\ L,\ p,\ s,\ \xi,\ {}_{\mathrm{d}}\Gamma_B)$;
${}_{A1}\omega$ = Source $(n,\ L,\ p,\ \tilde{\varphi},\ {}_{\mathrm{d}}\Gamma_A,\ {}_{\mathrm{r}}\Gamma_A,\ 0,\ p,\ n-1,\ 1,\ 0.5,\ 0)$;
${}_{A2}\omega$ = Source $(n,\ L,\ p,\ \tilde{\varphi},\ {}_{\mathrm{d}}\Gamma_A,\ {}_{\mathrm{r}}\Gamma_A,\ 1,\ p,\ 2n-1,\ 1,\ 0,\ 0)$;
${}_{A3}\omega$ = Source $(n,\ L,\ p,\ \tilde{\varphi},\ {}_{\mathrm{d}}\Gamma_A,\ {}_{\mathrm{r}}\Gamma_A,\ 1,\ 2p,\ 2n-1,\ 0,\ 1,\ 1)$;
${}_{B1}\omega$ = Source $(n,\ L,\ p,\ \tilde{\psi},\ {}_{\mathrm{d}}\Gamma_B,\ {}_{\mathrm{r}}\Gamma_B,\ 0,\ p,\ n-1,\ 1,\ 0.5,\ 0)$;
${}_{B2}\omega$ = Source $(n,\ L,\ p,\ \tilde{\psi},\ {}_{\mathrm{d}}\Gamma_B,\ {}_{\mathrm{r}}\Gamma_B,\ 1,\ p,\ 2n-1,\ 1,\ 0,\ 0)$;
${}_{B3}\omega$ = Source $(n,\ L,\ p\ \tilde{\psi},\ {}_{\mathrm{d}}\Gamma_B,\ {}_{\mathrm{r}}\Gamma_B,\ 1,\ 2p,\ 2n-1,\ 0,\ 1,\ 1)$;
for $(\ell,i,j,k) \in \{L\ldots,1\}\times\{0,\ldots,p\}\times\{0,\ldots,i\}\times\{0,\ldots,n-1\}$ **do**
  ${}_{\mathrm{A1}}\omega^{\ell-1}_{k+s^c_\ell,k} += \xi_{i,j}((-1)^{i+j}\,{}_{\mathrm{A1}}\omega^{\ell}_{2k,j} + {}_{\mathrm{A1}}\omega^{\ell}_{2k+1,j})$;
  ${}_{\mathrm{B1}}\omega^{\ell-1}_{k+s^c_\ell,k} += \xi_{i,j}((-1)^{i+j}\,{}_{\mathrm{B1}}\omega^{\ell}_{2k,j} + {}_{\mathrm{B1}}\omega^{\ell}_{2k+1,j})$;
end for
for $(\ell,i,k) \in \{0\ldots,L\}\times\{0,\ldots,p\}\times\{0,\ldots,n-1\}$ **do**
  if $k-\tilde{s}_\ell \ge 0$ & $k-\tilde{s}_\ell < 2n$ **then**
    ${}_1\omega^{\ell}_{k,i} += {}_{\mathrm{A1}}\omega^{\ell}_{k-\tilde{s}_\ell,i} + {}_{\mathrm{B1}}\omega^{\ell}_{k-\tilde{s}_\ell,i}$;
  end if
end for
for $(\ell,i,k) \in \{0\ldots,L\}\times\{0,\ldots,p\}\times\{0,\ldots,n-1\}$ **do**
  ${}_2\omega^{\ell}_{k,i} += {}_{\mathrm{A2}}\omega^{\ell}_{k,i} + {}_{\mathrm{B2}}\omega^{\ell}_{k,i}$;
end for
for $(\ell,i,j,k) \in \{1\ldots,L\}\times\{0,\ldots,2p+1\}\times\{i,\ldots,2p+1\}\times\{0,\ldots,\frac{n}{2}-1\}$ **do**
  ${}_3\omega^{\ell}_{2k,i} += \xi_{j,i}(-1)^{i+j}({}_{\mathrm{A3}}\omega^{\ell-1}_{k+s_\ell,j} + {}_{\mathrm{B3}}\omega^{\ell-1}_{k+s_\ell,j})$;
  ${}_3\omega^{\ell}_{2k+1,i} += \xi_{j,i}({}_{\mathrm{A3}}\omega^{\ell-1}_{k+s_\ell,j} + {}_{\mathrm{B3}}\omega^{\ell-1}_{k+s_\ell,j})$;
end for

29: **for** $(\ell, i, j, k) \in \{1 \ldots, L\} \times \{0, \ldots, 2p+1\} \times \{i, \ldots, 2p+1\} \times \{0, \ldots, \frac{n}{2} - 1\}$ **do**
30: $\quad {}_3\omega^{\ell}_{2k,i} += \xi_{j,i}(-1)^{i+j}({}_3\omega^{\ell-1}_{k+s_\ell,j})$;
31: $\quad {}_3\omega^{\ell}_{2k+1,i} += \xi_{j,i}({}_3\omega^{\ell-1}_{k+s_\ell,j})$;
32: **end for**
33: **for** $(\ell, i, k) \in \{1 \ldots, L\} \times \{0, \ldots, p\} \times \{0, \ldots, \frac{n}{2} - 1\}$ **do**
34: $\quad {}_1\omega^{\ell-1}_{k+s_\ell,i} = 0; \qquad {}_2\omega^{\ell-1}_{k+s_\ell,i} = 0; \qquad {}_3\omega^{\ell-1}_{k+s_\ell,i} = 0;$
35: **end for**
36: **for** $(\ell, i, k) \in \{0 \ldots, L\} \times \{0, \ldots, p\} \times \{0, \ldots, n-1\}$ **do**
37: $\quad \omega^{\ell}_{k,i} = 0.5({}_1\omega^{\ell}_{k,i} +_2 \omega^{\ell}_{k,i} +_3 \omega^{\ell}_{k,i})$;
38: **end for**
39: **return** (ω);

Remark 4. *In view of the orthonormality of the basis functions (3.46), the integral* $\int_0^1 \Phi^{\ell}_{k,i}(x)dx$ *simplifies to*

$$\int_0^1 \Phi^{\ell}_{k,i}(x)dx = \sqrt{h_\ell}\int_0^1 \frac{1}{\sqrt{h_\ell}}\Phi^{\ell}_{k,i}(x)dx = \sqrt{h_\ell}\int_0^1 \Phi^{\ell}_{k,0}(x)\Phi^{\ell}_{k,i}(x)dx$$
$$= \begin{cases} \sqrt{h_\ell} & : \ i = 0, \\ 0 & : \ i = 1 \ldots p. \end{cases}$$

Thus, the integral in (5.27) is computed through

$$\int_0^1 \varphi(y)dy = \sqrt{h_\ell}\sum_{\ell=0}^{L}\sum_{k=0}^{n}\varphi^{\ell}_{k,0} \tag{5.28}$$

which is implemented in Algorithm 16, lines 2-6. In lines 7-9, the resulting integral (5.28) is multiplied by the coefficients of ψ, leading to the final result of the aggregation sink integral $\psi(x)\int_0^1 \varphi(y)dy$.

5.5 Numerical results

In this section, we provide numerical results for the methods and algorithms proposed in the previous sections. The method can be applied for any polynomial

Algorithm 16 $z = \text{Sink}\ (n,\ L,\ p,\ \varphi,\ \psi)$;

Input:	$n \in \mathbb{N}$;	(number of intervals on coarse mesh)
	$L \in \mathbb{N}$;	(number of nested refinements of grid)
	$p \in \mathbb{N}$;	(degree of polynomial)
Output:	$z = (z_{i,j}^{\ell})_{\substack{0 \le i \le p \\ 0 \le \ell \le L \\ 0 \le j < n}} \in \mathbb{R}^{(p+1)\times(L+1)\times n}$;	(coefficients of z)

$c := 0$; ;
for $l = 0 : L$ **do**
  for $k = 0 : n-1$ **do**
    $c+ = \varphi_{k,0}^{\ell}\sqrt{h_\ell}$;
  end for
end for
for $(\ell, i, j) \in \{0, \ldots, L-1\} \times \{0, \ldots, p\} \times \{0, \ldots, n-1\}$ **do**
  $z_{k,i}^{\ell}+ = c\psi_{k,i}^{\ell}$;
end for
return (z);

order $p \geq 0$ while being mass conserving for all $p \geq 1$ (see proof in [29], Theorem 4.1).

As a kernel function, we use the Brownian kernel $\kappa_{\mathrm{B}}(x, y)$ (3.2). In our tests, we will use one of the initial density distributions $f_1(0, x)$ (3.12), $f_2(0, x)$ (3.13), $f_4(0, x)$ (3.15).

Figure 5.2 shows the distributions computed at time $t = 4$, $f_i(4, x)$ $(i \in \{1, 2, 4\})$, which have been computed on a uniform grid with $n = 2048$ intervals and polynomials of order $p = 5$ and as such serve as reference solutions for testing the accuracy of approximations to the density distribution computed with fewer degrees of freedom.

We will provide numerical tests to compare results with respect to the accuracy of the computed solutions and to the required computational time (in seconds). In the first part of this section, we will test the algorithms on uniform grids, varying the polynomial degree p and the grid width $\frac{1}{n}$, leading to $(p+1)n$ degrees of freedom. In the second part, we report on numerical tests performed on nested grids, illustrating the effect of adjusting the location of the grid refinement to the shape of the density distribution.

Uniform grids

In Figure 5.3, we show the difference between the reference solution computed

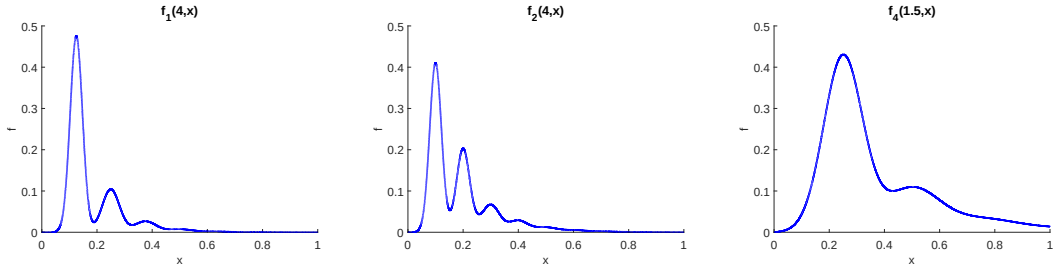


Figure 5.2: Computed distributions $f_1(4,x)$, $f_2(4,x)$, $f_4(1.5,x)$ for the initial distributions (3.12), (3.13) and (3.15) all using time stepwidth $dt = 0.01$, Brownian kernel, uniform grid for $p = 5$, $n = 2048$.

with $p = 5$, $n = 2048$ and the solution computed with $p = 3$, $n = 128$ for density distributions $f_1(4,x)$ and $f_2(4,x)$. The error is a piecewise polynomial function with $p = 5$ on 2048 intervals. Since the absolute value of the error is small on the right half $(0.5, 1]$ ($\leq 7.5 \cdot 10^{-6}$ for $f_1(4,x)$ and $\leq 1.25 \cdot 10^{-5}$ for $f_2(4,x)$), we show the error only on the left half $(0, 0.5]$. Larger errors appear only around the peaks of the computed density distributions $f_j(4,x)$, motivating local grid refinement as discussed in this chapter.

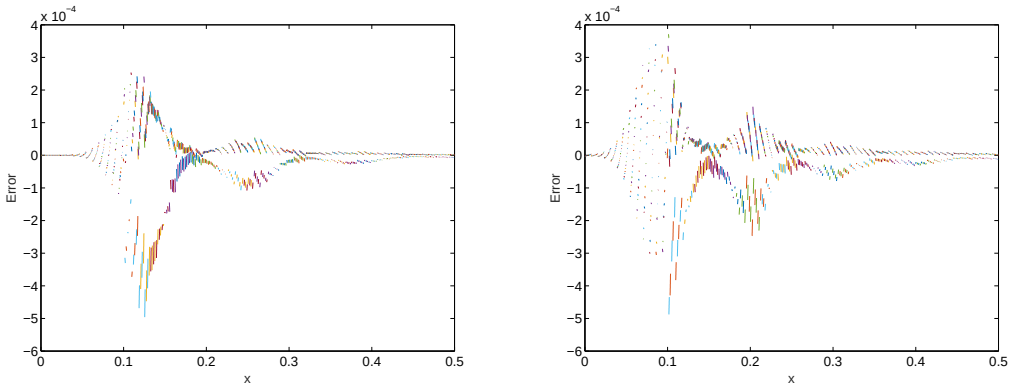


Figure 5.3: Error between the reference solution with $p = 5$, $n = 2048$ and approximations using $p = 3$, $n = 128$ for $f_1(4,x)$ (left) and $f_2(4,x)$ (right) (400 time steps, $dt = 0.01$, uniform grid, Brownian kernel).

In Figure 5.4, we show the L_2 errors between the reference and computed solutions with respect to the number of degrees of freedom, $(p+1)n$, (left plots) as well as with respect to the required computational time (in seconds, right plots), obtained for different combinations of n and p for the initial distributions $f_1(0,x)$ (3.12) and $f_2(0,x)$ (3.13) after 400 time steps with $dt = 0.01$. Each figure shows curves associated with a particular polynomial order p, and each curve has (up to) seven

markers corresponding to (from left to right) the number of uniform intervals $n = 16, 32, 64, 128, 256, 512, 1024$.

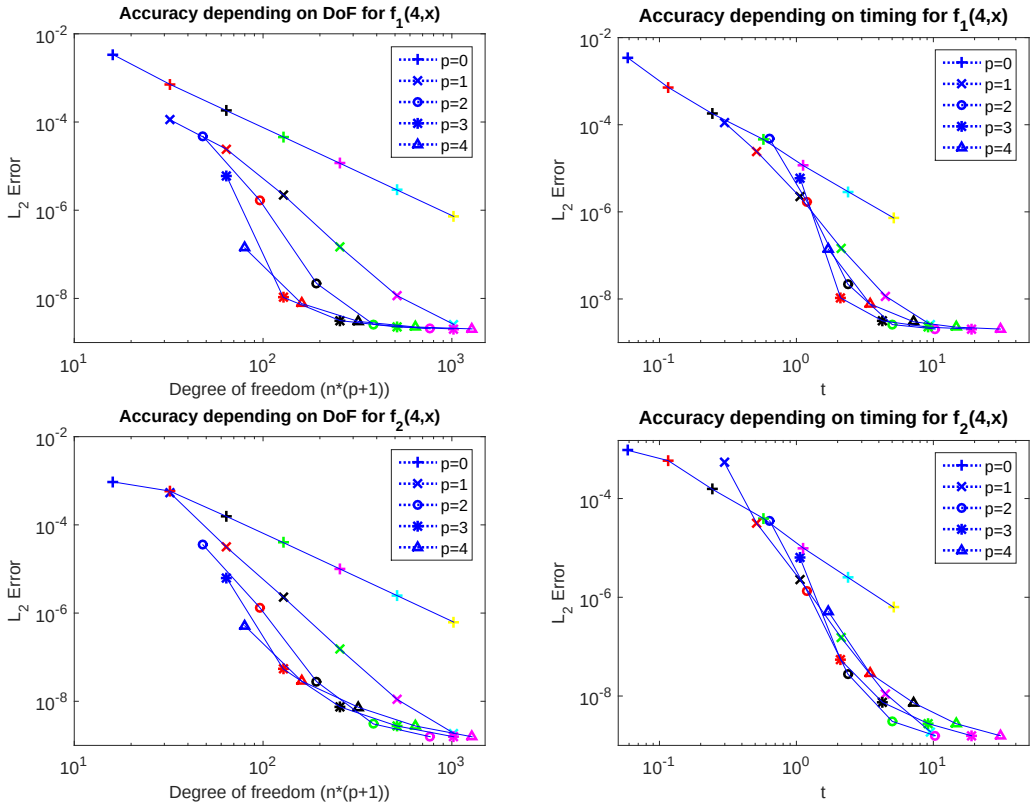


Figure 5.4: L_2 error depending on the number of DoFs (left) and on the computational time (in seconds, right) for the initial density distributions $f_1(0, x)$ (top) and $f_2(0, x)$ (bottom) after 400 time steps ($dt = 0.01$, uniform grid, Brownian kernel).

The plots show that higher order polynomials lead to more accurate solutions (left), however at the price of increased computational costs. In these two examples, a recommendation on the optimal choice of n and p appears to depend on the desired accuracy: less accurate solutions are computed fastest with low order polynomials while better accuracies are reached in shorter time using higher polynomial orders with a break-even accuracy around 10^{-6}.

The algorithms discussed in the previous sections are of (theoretical) complexity $\mathcal{O}((p+1)^2 N \log N) \approx \mathcal{O}((p+1)^2(L+1)n \log n)$ ([30]) which becomes $\mathcal{O}((p+1)^2 n \log n)$ on uniform grids. In Figure 5.5, we confirm these estimates through our numerical tests.

Nested grids

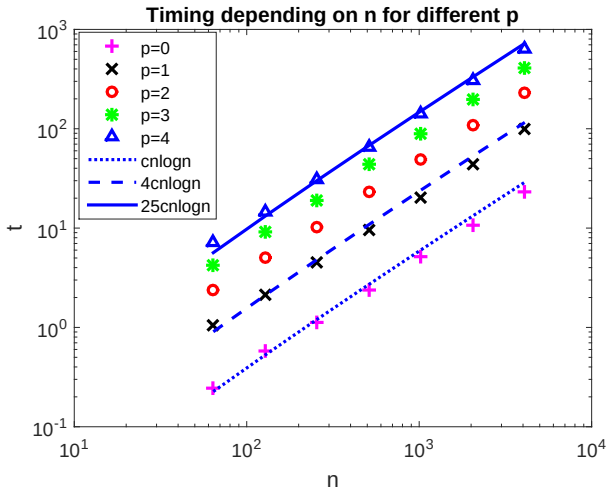


Figure 5.5: Computational time (in seconds) depending on n and p for 400 time steps ($dt = 0.01$, uniform grid, Brownian kernel) and reference complexity lines $c(p+1)^2 n \log n$ with $c = 8.5e-4$.

In Figure 5.6, we show numerical results for the initial distribution $f_4(0,x)$ (3.15) on two different nested grids with three levels of refinement (i.e, $L = 2$) and, for comparison, on a uniform grid ($L = 0$). The refinement occurs at different locations that are specified by the shift parameters $s_1 = s_2 = 0$ and $s_1 = 0, s_2 = \frac{n}{4}$, respectively. In the case $s_1 = s_2 = 0$, we refine first time in the interval $(0, \frac{1}{2}]$ and second time in the interval $(0, \frac{1}{4}]$ whereas $s_1 = 0, s_2 = \frac{n}{4}$ leads to the same refinement for the first level in the interval $(0, \frac{1}{2}]$ but to a different refinement for the second level in the interval $[\frac{1}{8}, \frac{3}{8}]$ around the peak of the initial distribution $f_4(0,x)$.

We perform tests both for piecewise linear ($p = 1$) and piecewise quadratic ($p = 2$) functions. The top left plot in Figure 5.6 shows the L_2 error of the computed approximation with respect to the number of degrees of freedom. For a fixed polynomial degree p, the results using the shift $s_1 = 0, s_2 = \frac{n}{4}$ are better compared to a uniform grid or, even worse, the "wrong" shift $s_1 = s_2 = 0$. Results obtained with polynomial degree $p = 2$ are all more accurate than those for polynomial degree $p = 1$, regardless of the nested refinement. In Figure 5.6 (top right), we show the computational time with respect to the number of degrees of freedom. As expected, computations on a uniform grid and on a nested grid refined toward the origin are faster than those on a nested grid with $s_1 = 0, s_2 = n/4$. In view of Lemma 2, a refinement toward 0 requires less time than a refinement using

$s_1 = 0, s_2 = \frac{n}{4}$ (or, in fact, any other $s_\ell > 0$) since only the first sum $\sum_1$ instead of all three sums $\sum_1, \sum_2, \sum_3$ in (5.3) have to be computed. Figure 5.6 (bottom) illustrates the L_2 error with respect to the computational time to compute $f_4(4, x)$. The results indicate that, at least for this chosen test case, uniform grids are very competitive since the local refinement introduces computational overhead which leads to computational times surpassing those for using uniform grids (leading to comparable accuracies).

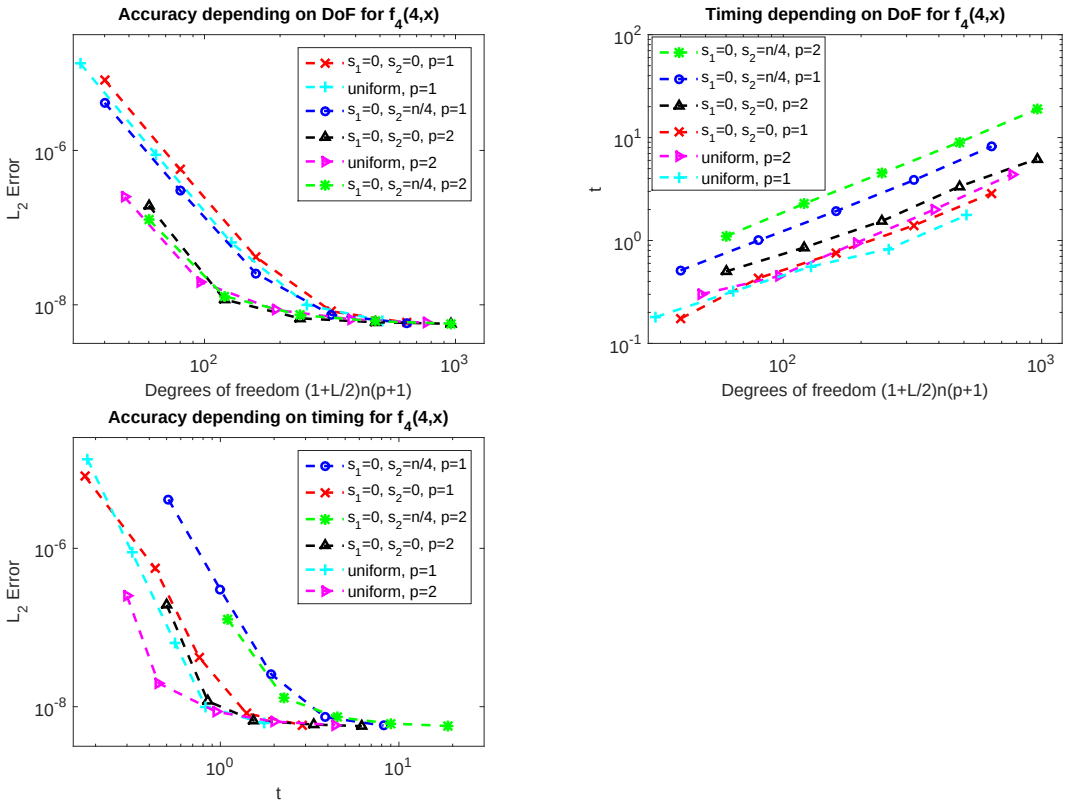


Figure 5.6: Numerical results on nested grids with three levels of refinement ($L = 2$), shifts $s_1 = s_2 = 0$ and $s_1 = 0, s_2 = n/4$ and on a uniform grid ($L = 0$), polynomial degree $p = 1$ and $p = 2$; all for $dt = 0.01$ and 150 time steps. Top left: L_2 error with respect to the number of degrees of freedom; Top right: Computational time with respect to the number of degrees of freedom; Bottom: L_2 error with respect to computational time.

In Figure 5.7, we illustrate the computational time for different combinations of L and n for fixed $p = 1$ for a grid that is refined toward $\frac{1}{2}$ using shifts $s_\ell = \frac{n}{4}$, confirming the linear complexity in the number of local grid refinements L.

The numerical tests in this section allow the following conclusions.

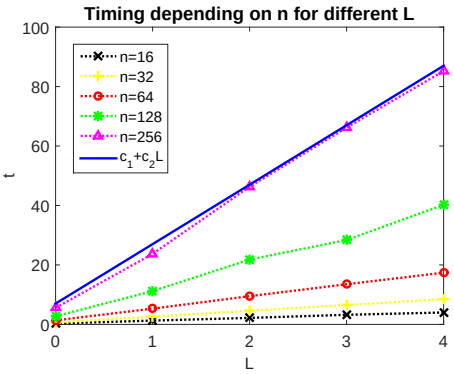


Figure 5.7: Computational time (seconds) for different combinations of L and n with shifts $s_\ell = \frac{n}{4}$ and fixed $p = 1$ for 400 time steps with $dt = 0.01$, in complexity graph $c_1 = 7$, $c_2 = 20$.

- Theoretical complexity estimates $\mathcal{O}((p+1)^2(L+1)n\log n)$ are confirmed through numerical tests (see Figure 5.5 for dependence on p, n, Figure 5.6 (top right) for nested refinement and Figure 5.7 for dependence on L).
- Numerical tests illustrate the L_2 error of computed approximations both in terms of the required storage (indirectly through the number of degrees of freedom) and the required computational time.
  - If the objective is to reduce the necessary number of degrees of freedom (while maintaining a certain accuracy), an increase of polynomial degree p is recommended rather than an increase in the number of intervals n (illustrated for uniform grids in Figure 5.4, left plots). Increasing the polynomial degree (on all intervals) also appears to be superior to a local grid refinement (see Figure 5.6, top left).
  - If the objective is to reduce the computational time (while maintaining a certain accuracy), the situation is different: Now a uniform grid appears to be preferable over a nested grid (see Figure 5.6, bottom), and an increase in polynomial degree beyond $p = 1$ does not show much benefit (if any) compared to $p = 1$ (on different uniform grids, leading to the same number of degrees of freedom, see Figure 5.4, right plots).

Chapter 6

Evaluation of bivariate aggregation integrals with piecewise constant functions on uniform meshes

6.1 Introduction of the bivariate problem

Different approaches and numerical results for bivariate population balance equations have for example been introduced in [2], [73], [41], [14], [15], [44], [12].
In [2], [73] and [41] the one-dimensional fixed pivot (see section 2.3) and the cell average techniques have been extended to solve two-dimensional population balance equations using a rectangular grid where a particle is assigned to four neighboring points. To solve multidimensional problems (considering d particle properties) with this method a particle is assigned to 2^d pivots. In [12] a new approach has been introduced for the fixed pivot scheme, where a particle needs to be represented only through $d + 1$ instead of 2^d pivot points, e.g., for bivariate problems triangular elements are chosen to fill in the two-dimensional space. The fixed pivot method has been applied to two different types of triangular grids and a better result has been reported for number density distribution using these triangular grids than the rectangular grids used earlier. Higher moments obtained with the fixed pivot method on rectangular and triangular grids have been compared in [44]. In the same paper the two types of triangular grids considered for the fixed pivot method have been applied also for the cell average technique. In [14] and [15] a new structured discretization, called X-discretization, of the two-dimensional space is proposed. Here, with the birth of larger particles, the computational domain is expanded, and with the death of smaller particles, it is contracted, reducing the

computational effort further.

In this chapter we discuss the discretization of the computational domain though an equidistant mesh. The direct implementation of this approach may lead to quadratic complexity in the number of unknowns. The complexity could amount to n^d for tensor grids where for discretization n degrees of freedom are used for any of the d properties, quickly leading to a computational bottleneck. Particularly, for bivariate problems the straight-forward evaluation of the integrals leads to $\mathcal{O}(n^4)$ complexity, which can be reduced to $\mathcal{O}(n^2 \log n^2) = \mathcal{O}(2n^2 \log n)$ applying fast Fourier transformation. Such a reduction of complexity for bivariate problems will be the subject of this chapter. There are various algorithms in the literature for reducing the complexity using multidimensional fast Fourier transform [17], [62], [16]. Here, we will discuss the most commonly used algorithm, known as the row-column algorithm [18], [63].

The partial integro-differential equation governing the density function f for bivariate problems is given as follows,

$$\frac{\partial f(x_1, x_2, t)}{\partial t} = Q_{\text{agg}}(f)(x_1, x_2, t) := Q_{\text{source}}(f)(x_1, x_2, t) - Q_{\text{sink}}(f)(x_1, x_2, t), \\ x_1, x_2 \in (0, 1],\ t \in [0, T], \quad (6.1)$$

with aggregation source and sink terms

$$Q_{\text{source}}(f)(x_1, x_2, t) = \\ \frac{1}{2} \int_0^{x_1} \int_0^{x_2} \kappa(x_1 - y_1, x_2 - y_2, y_1, y_2) f(x_1 - y_1, x_2 - y_2, t) f(y_1, y_2, t)\, dy_1\, dy_2, \quad (6.2)$$

$$Q_{\text{sink}}(f)(x_1, x_2, t) = \\ f(x_1, x_2, t) \int_0^{1-x_1} \int_0^{1.-x_2} \kappa(x_1, x_2, y_1, y_2) f(y_1, y_2, t)\, dy_1\, dy_2. \quad (6.3)$$

The source term (6.2) describes the birth of particles of property (x_1, x_2) from particles of property $(x_1 - y_1, x_2 - y_2)$ and (y_1, y_2). The sink term (6.3) corresponds to the death of particles (x_1, x_2) due to the aggregation with particles of property (y_1, y_2). Here we assume that particles have properties in the interval $(0, 1] \times (0, 1]$. A particle of property (x_1, x_2) can then aggregate only with particles of up to maximum property $(1 - x_1, 1 - x_2)$, leading to the integral upper limits in (6.3). The aggregation kernel $\kappa(x_1, x_2, y_1, y_2)$ describes the aggregation rate of particles with properties (x_1, x_2) and (y_1, y_2) to form particles of size $(x_1 + y_1, x_2 + y_2)$. Note that for this bivariate model both particle properties have to be "additive".

In the bivariate case as well, the efficient evaluation of aggregation integrals is based on a separable approximation of the kernel function and fast Fourier transformation. The separable kernel approximation of the bivariate aggregation kernel $\kappa(x_1, x_2, y_1, y_2)$ can be expressed in the form

$$\kappa(x_1, x_2, y_1, y_2) = \sum_{\nu=1}^{M} a_\nu(x_1, x_2) b_\nu(y_1, y_2). \tag{6.4}$$

There are various methods in the literature to derive separable approximations of the bivariate aggregation kernels, including Chebyshev interpolation and adaptive cross approximation. These methods have been described in subsection 4.1.3 for univariate kernels but can also be extended to derive separable approximations for the bivariate kernels [33]. After this approximation step, the aggregation source (6.2) and sink (6.3) integrals can be computed as sums of M terms of the form

$$Q_{\text{source}}(x_1, x_2) = \frac{1}{2} \int_0^{x_1} \int_0^{x_2} \psi_\nu(x_1 - y_1, x_2 - y_2) \varphi_\nu(y_1, y_2)\, dy_1\, dy_2 =$$

$$\frac{1}{2} \int_0^{x_1} \int_0^{x_2} a_\nu(x_1 - y_1, x_2 - y_2) f(x_1 - y_1, x_2 - y_2) b_\nu(y_1, y_2) f(y_1, y_2)\, dy_1\, dy_2, \tag{6.5}$$

$$Q_{\text{sink}}(x_1, x_2) = \psi_\nu(x_1, x_2) \int_0^{1-x_1} \int_0^{1-x_2} \varphi_\nu(y_1, y_2)\, dy_1\, dy_2 =$$

$$a_\nu(x_1, x_2) f(x_1, x_2) \int_0^{1-x_1} \int_0^{1-x_2} b_\nu(y_1, y_2) f(y_1, y_2)\, dy_1\, dy_2. \tag{6.6}$$

On the domain $(0, 1] \times (0, 1]$ (see figure 6.1) the grid $\mathcal{G}$ is characterized by $(n+1)^2$ points

$$\mathcal{G} := \{(x_i^1, x_j^2) \mid x_i^1 = ih, x_j^2 = jh, 0 \le i, j \le n\} \tag{6.7}$$

with $h = \frac{1}{n}$, where n is the number of intervals in each property dimension. The mesh $\mathcal{C}$ associated with this grid consists of n^2 cells and is defined as

$$\mathcal{C} := \bigcup_{i,j=0}^{n-1} \mathcal{C}_{i,j}, \tag{6.8}$$

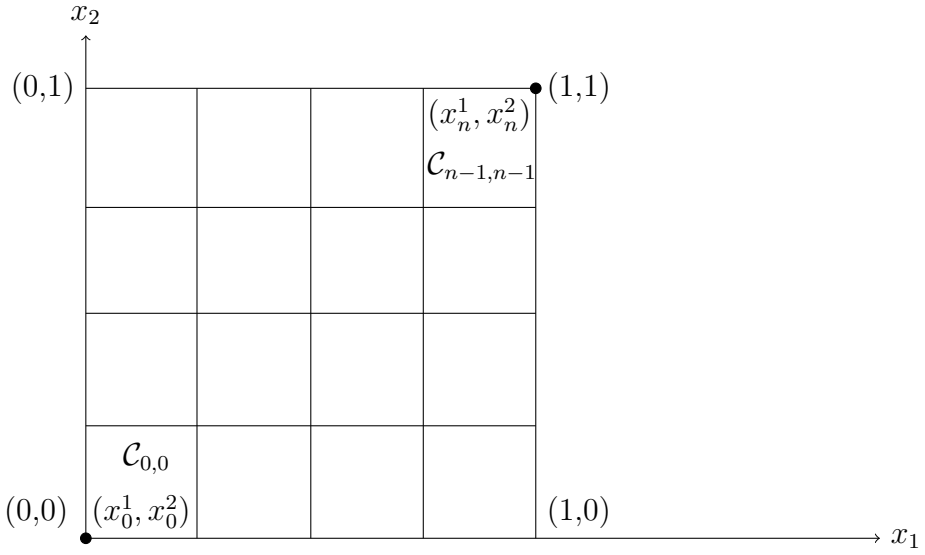


Figure 6.1: Uniform two-dimensional grid, $n = 4$.

where each cell is given by

$$\mathcal{C}_{i,j} := (ih, (i+1)h] \times (jh, (j+1)h] \qquad \text{with} \qquad i, j = 0 \ldots n-1. \tag{6.9}$$

We define the function space $\mathcal{S}$ as

$$\mathcal{S} := \{f \mid f \text{ is piecewise constant with respect to the mesh } \mathcal{C}\}. \tag{6.10}$$

The function values are assigned to the mid points $\left(x^1_{i+\frac{1}{2}}, x^2_{j+\frac{1}{2}}\right)$ of cells $\mathcal{C}_{i,j}$.

6.2 Source term

We drop the subscript ν in (6.5) and focus on the evaluation of

$$Q_{\text{source}}(x_1, x_2) = \frac{1}{2} \int_0^{x_1} \int_0^{x_2} \psi(x_1 - y_1, x_2 - y_2) \varphi(y_1, y_2)\, dy_1\, dy_2, \tag{6.11}$$

where the functions $\varphi, \psi \in \mathcal{S}$ (6.10) are uniquely determined through the constant values $\varphi_{k,\ell}, \psi_{k,\ell}$ on the cell $\mathcal{C}_{k,\ell},\ k = 0, \ldots i,\ \ell = 0, \ldots j$. The area of the cell $\mathcal{C}_{k,\ell}$ (6.9) is h^2, leading to the discrete formulation

$$\omega_{i+1,j+1} = \frac{h^2}{2} \sum_{k=0}^{i} \sum_{\ell=0}^{j} \varphi_{k,\ell} \psi_{i-k,j-\ell}, \qquad i, j = 0, \ldots, n-1. \tag{6.12}$$

In (6.12) $\omega_{i+1,j+1}$ is the value at the grid point (x^1_{i+1}, x^2_{j+1}). It is obvious that $\omega_{i,j} = 0$ if $i = 0$ or $j = 0$.
The efficient evaluation of (6.12) can be accomplished by a two-dimensional fast Fourier transformation. We introduce Algorithm 17 that receives as an input a matrix and outputs either its fast Fourier (FFT) or inverse fast Fourier (IFFT) transformation. The two-dimensional (inverse) fast Fourier transformation can be obtained from the one-dimensional Fourier transformation (section 3.3). For two-dimensional FFT we use the well-known row-column algorithm [18], [63]. In (6.13) we show how the two-dimensional FFTs can be expressed by means of one-dimensional FFT

$$\begin{aligned}(\mathrm{FFT2}(y))_{\mu,\nu} &:= \sum_{\alpha,\beta=0}^{2n-1} y_{\alpha,\beta} e^{-i\alpha\mu\pi/n} e^{-i\beta\nu\pi/n} = \sum_{\beta=0}^{2n-1} (\mathrm{FFT}_{\mathrm{row}}(y))_{\mu,\beta} e^{-i\beta\nu\pi/n} \\ &= (\mathrm{FFT}_{\mathrm{col}}(\mathrm{FFT}_{\mathrm{row}}(y)))_{\mu,\nu}, \end{aligned} \tag{6.13}$$

where the subscripts "row" and "col" denote row and column and $(\mathrm{FFT}_{\mathrm{row}}(y))_{\mu,\beta}$ and $(\mathrm{FFT}_{\mathrm{col}}(y))_{\mu,\nu}$ are defined as follows

$$(\mathrm{FFT}_{\mathrm{row}}(y))_{\mu,\beta} := \sum_{\alpha=0}^{2n-1} y_{\alpha,\beta} e^{-i\alpha\mu\pi/n}, \qquad (\mathrm{FFT}_{\mathrm{col}}(y))_{\mu,\nu} := \sum_{\beta=0}^{2n-1} y_{\mu,\beta} e^{-i\beta\nu\pi/n}.$$

As a prerequisite for the application of FFT similar to the one-dimensional case, we need to expand the given matrices $\psi, \varphi \in \mathbb{R}^{n\times n}$ to $\bar{\psi}, \bar{\varphi} \in \mathbb{R}^{2n\times 2n}$ by zero padding. Algorithm 17 computes the one-dimensional Fourier transformation of each column of the expanded matrix $\bar{\psi}$, then of each row of the result according to (6.13).

As the next step we develop Algorithm 18 that performs the evaluation of the sum (6.12) using the two-dimensional Fourier transformation (Algorithm 17).

First, we compute coefficients $\hat{\varphi}_{m,q}$ and $\hat{\psi}_{s,p}$, $m, q, s, p = 0, \ldots 2n-1$ such that

$$\bar{\varphi}_{k,\ell} = \frac{1}{4n^2} \sum_{m,q=0}^{2n-1} \hat{\varphi}_{m,q} e^{-imk\pi/n} e^{-iq\ell\pi/n} \tag{6.14}$$

$$\bar{\psi}_{k,\ell} = \frac{1}{4n^2} \sum_{s,p=0}^{2n-1} \hat{\psi}_{s,p} e^{-is(k-\nu)\pi/n} e^{-ip(\ell-\mu)\pi/n} \tag{6.15}$$

hold for all $\ell, k = 0, \ldots, 2n-1$. Substituting (6.14) and (6.15) in (6.12) yields the

Algorithm 17 $\hat{\psi} = $ (I)FFT2 (n, ψ);
(Two-dimensional fast Fourier transformation.)

Input: Intervals in each property dimension $n \in \mathbb{N}$, matrix $\psi \in \mathbb{R}^{n \times n}$ for convolution;
Output: Two-dimensional fast (inverse) Fourier transform $\hat{\psi} \in \mathbb{R}^{2n \times 2n}$ of matrix ψ;

1: **for** $(i, j) \in \{0, \ldots, n-1\} \times \{0, \ldots, n-1\}$ **do**
2: $\quad \bar{\psi}_{i,j} = \psi_{i,j}; \qquad \bar{\psi}_{n+i,n+j} = 0; \qquad \bar{\psi}_{i,n+j} = 0; \qquad \bar{\psi}_{n+i,j} = 0;$
3: **end for**
4: **for** $i = 0 : n-1$ **do**
5: $\quad \hat{\psi}_{i,:} :=$(I)FFT$(\bar{\psi}_{i,:})$;
6: **end for**
7: **for** $j = 0 : 2n-1$ **do**
8: $\quad \hat{\psi}_{:,i} :=$(I)FFT$(\hat{\psi}_{:,i})$;
9: **end for**
10: **return** $\hat{\psi}$;

following expression with extended matrices $\bar{\varphi}$ and $\bar{\psi}$. Then it holds that

$$
\begin{aligned}
\omega_{\nu+1,\mu+1} &= \sum_{k=0}^{2n-1} \sum_{\ell=0}^{2n-1} \bar{\varphi}_{k,\ell} \bar{\psi}_{\nu-k,\mu-\ell} \\
&= \sum_{k=0}^{2n-1} \sum_{\ell=0}^{2n-1} \left(\frac{1}{4n^2} \sum_{m,q=0}^{2n-1} \hat{\varphi}_{m,q} e^{-imk\pi/n} e^{-iq\ell\pi/n} \right) \left(\frac{1}{4n^2} \sum_{s,p=0}^{2n-1} \hat{\psi}_{s,p} e^{-is(k-\nu)\pi/n} e^{-ip(\ell-\mu)\pi/n} \right) \\
&= \frac{1}{4n^2} \sum_{m,q=0}^{2n-1} \hat{\varphi}_{m,q} \sum_{s,p=0}^{2n-1} \hat{\psi}_{s,p} e^{-is\nu\pi/n} e^{-ip\mu\pi/n} \left(\frac{1}{4n^2} \sum_{k,\ell=0}^{2n-1} \underbrace{e^{-i(m-s)k\pi/n} e^{-i(q-p)\ell\pi/n}}_{= \frac{1}{4n^2} 4n^2 \delta_{m,s} \delta_{q,p}} \right) \\
&= \frac{1}{4n^2} \sum_{m,q=0}^{2n-1} \underbrace{\hat{\varphi}_{m,q} \hat{\psi}_{m,q}}_{=: \hat{\omega}_{m+1,q+1}} e^{-im\nu\pi/n} e^{-iq\mu\pi/n} \\
&= \frac{1}{4n^2} \sum_{m,q=0}^{2n-1} \hat{\omega}_{m+1,q+1} e^{-im\nu\pi/n} e^{-iq\mu\pi/n}
\end{aligned}
\tag{6.16}
$$

with Kronecker delta $\delta_{m,s}\delta_{q,p} = \begin{cases} 1 & \text{if } m = s, q = p \\ 0 & \text{else.} \end{cases}$

In view of (6.16), the computation of the sum appearing in (6.12) can be summarized in three steps introduced in Algorithm 18 (see also three steps in section 3.3).

The computation of the source term (6.2) leads to a two-dimensional piecewise

Algorithm 18 ω = FFT_Convolution_2D (n, ψ, φ);
(Convolution using fast Fourier transformation.)

Input: Intervals in each property dimension $n \in \mathbb{N}$, matrix $\psi, \varphi \in \mathbb{R}^{n\times n}$ for convolution;

Output: Convoluted sum $\omega \in \mathbb{R}^{2n\times 2n}$ $(\omega_{i+1,j+1} = \sum_{k=0}^{i}\sum_{\ell=0}^{j} \varphi_{k,\ell}\psi_{i-k,j-\ell},\ \ i,j = 0,\dots,n-1)$;

$\hat{\psi}$=FFT2(n,ψ);
$\hat{\varphi}$=FFT2(n,φ);
for $i = 0 : 2n-1$ **do**
  for $j = 0 : 2n-1$ **do**
    $\hat{\omega}_{i+1,j+1} = \hat{\psi}_{i,j} \cdot \hat{\varphi}_{i,j}$
  end for
end for
$\bar{\omega}$=IFFT2$(n,\hat{\omega})$;
$\omega = \bar{\omega}(1:n, 1:n)$;
return ω;

bilinear function of the form $(a_1 + b_1x_1)(a_2 + b_2x_2)$, that has to be projected to the piecewise constant space $\mathcal{S}$ (6.10). We aim to find a mass preserving projection that maps the values of the function $\omega(x_1, x_2)$ at the grid points (x_i^1, x_j^2), (x_{i+1}^1, x_j^2), (x_i^1, x_{j+1}^2), (x_{i+1}^1, x_{j+1}^2) to a single value ω_c defined at the point $(x_{i+\frac{1}{2}}^1, x_{j+\frac{1}{2}}^1)$ (see Figure 6.3).
We denote the values of $\omega(x_1, x_2)$ at four grid points forming the cell $\mathcal{C}_{i,j}$ as follows

$$\begin{aligned}
\omega_{i,j} &:= \omega(x_i^1, x_j^2) = \omega(ih, jh),\\
\omega_{i+1,j} &:= \omega(x_{i+1}^1, x_j^2) = \omega((i+1)h, jh),\\
\omega_{i,j+1} &:= \omega(x_i^1, x_{j+1}^2) = \omega(ih, (j+1)h),\\
\omega_{i+1,j+1} &:= \omega(x_{i+1}^1, x_{j+1}^2) = \omega((i+1)h, (j+1)h).
\end{aligned} \tag{6.17}$$

The piecewise linear function that satisfies conditions (6.17) is uniquely defined as

$$\begin{aligned}
\omega(x_1, x_2) =& \frac{\omega_{i+1,j+1} + \omega_{i,j} - \omega_{i+1,j} - \omega_{i,j+1}}{h^2}(x_1 - ih)(x_2 - jh) +\\
&\frac{\omega_{i+1,j} - \omega_{i,j}}{h}(x_1 - ih) + \frac{\omega_{i,j+1} - \omega_{i,j}}{h}(x_2 - jh) + \omega_{i,j}.
\end{aligned} \tag{6.18}$$

Similar to the one-dimensional case, to preserve the total amount of x_1 particle property, e.g., total mass, the following equality has to hold

$$\int_{ih}^{(i+1)h}\int_{jh}^{(j+1)h} x_1\omega(x_1, x_2)\, dx_2\, dx_1 = \int_{ih}^{(i+1)h}\int_{jh}^{(j+1)h} x_1\omega_c\, dx_2\, dx_1. \tag{6.19}$$

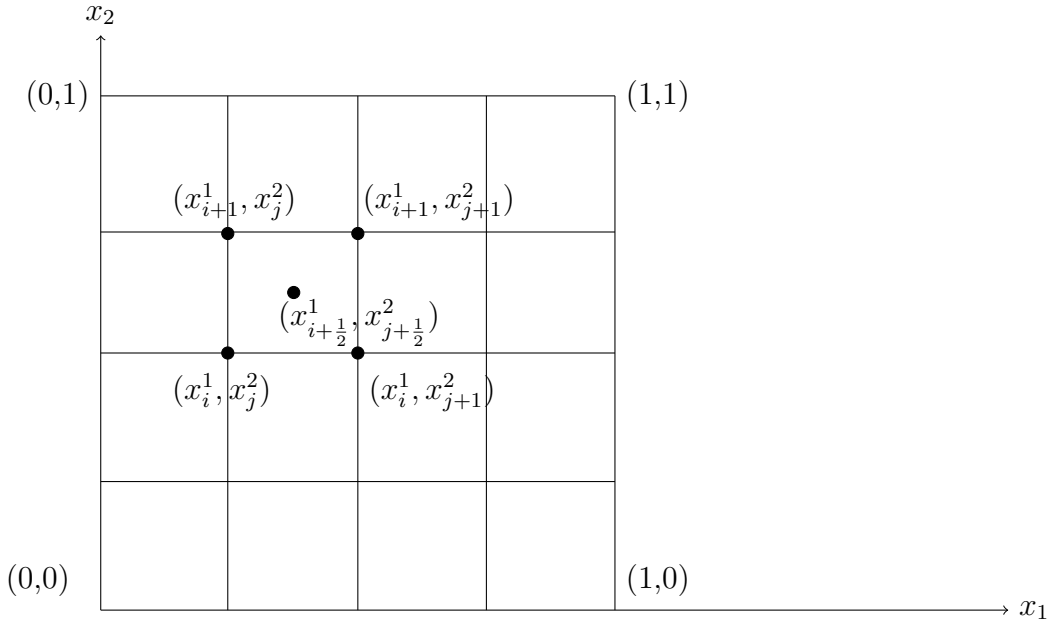


Figure 6.2: Uniform two-dimensional grid, $n = 4$.

A straight-forward integration in (6.19) with $\omega(x_1, x_2)$ from (6.18) leads to

$$\begin{aligned}\omega_c =& \frac{3i+2}{12i+6}(\omega_{i+1,j+1} + \omega_{i,j} - \omega_{i+1,j} - \omega_{i,j+1}) + \\ & \frac{3i+2}{6i+3}(\omega_{i+1,j} - \omega_{i,j}) + \frac{\omega_{i,j+1} - \omega_{i,j}}{2} + \omega_{i,j}.\end{aligned} \tag{6.20}$$

In Algorithm 19, line 7 yields the convoluted sum on the grid points. In line 9, we apply the boundary conditions, and in line 13 we insert the computed values. Finally, in line 18 we apply the mass conserving projection (6.20).

6.3 Sink term

In this section we discuss the efficient evaluation of the bivariate aggregation sink term (6.3) using a kernel separation, which leads to the form (6.6). The sink term can be rewritten as

$$Q_{\text{sink}}(x_1, x_2) = \psi(x_1, x_2) \int_0^{1-x_1} \int_0^{1-x_2} \varphi(y_1, y_2)\, dy_1\, dy_2 = \psi(x_1, x_2)\bar{\varphi}(x_1, x_2), \tag{6.21}$$

Algorithm 19 $q = \text{separableSourceAggregation\_2D}(n, f, a, b)$;
(Evaluate the aggregation source integral (6.2) using piecewise constant approximations on an equidistant mesh.)

Input: $n \in \mathbb{N}$; (intervals in each property dimension)
$f \in \mathbb{R}^{n \times n}$; (piecewise constant density function values)
$a, b \in \mathbb{R}^{n \times n}$; (kernel factors)
Output: $q \in \mathbb{R}^{n \times n}$; (piecewise const. aggreg. $q_{i,j} \approx Q_{\text{source}}(f)$ on cell $\mathcal{C}_{i,j}$)

1: **for** $i = 0 : n-1$ **do**
2: **for** $j = 0 : n-1$ **do**
3: $\psi_{i,j} = a_{i,j} \cdot f_{i,j}$;
4: $\varphi_{i,j} = b_{i,j} \cdot f_{i,j}$;
5: **end for**
6: **end for**
7: $\omega = \text{FFT\_Convolution\_2D}(n, \varphi, \psi)$;
8: **for** $i = 0 : n$ **do**
9: $\tilde{\omega}_{i,0} = 0; \qquad \tilde{\omega}_{0,i} = 0;$
10: **end for**
11: **for** $i = 0 : n$ **do**
12: **for** $j = 0 : n$ **do**
13: $\tilde{\omega}_{i+1,j+1} = \omega_{i,j}$;
14: **end for**
15: **end for**
16: **for** $i = 0 : n-1$ **do**
17: **for** $j = 0 : n-1$ **do**
18: $q_{i,j} = \frac{h^2}{2} \Big(\frac{3i+2}{12i+6}(\tilde{\omega}_{i+1,j+1} + \tilde{\omega}_{i,j} - \tilde{\omega}_{i+1,j} - \tilde{\omega}_{i,j+1}) + \frac{3i+2}{6i+3}(\tilde{\omega}_{i+1,j} - \tilde{\omega}_{i,j})$
19: $+ \frac{\tilde{\omega}_{i,j+1} - \tilde{\omega}_{i,j}}{2} + \tilde{\omega}_{i,j} \Big)$;
20: **end for**
21: **end for**

with

$$\bar{\varphi}(x_1, x_2) = \int_0^{1-x_1} \int_0^{1-x_2} \varphi(y_1, y_2)\, dy_1\, dy_2. \tag{6.22}$$

Lemma 3. *Denoting $\bar{\varphi}(x_i^1, x_j^2) = \bar{\varphi}_{i,j}$ and $\varphi(x_i^1, x_j^2) = \varphi_{i,j}$ in integral (6.22) it holds that*

$$\bar{\varphi}_{i,j} = \bar{\varphi}_{i+1,j} - \bar{\varphi}_{i+1,j+1} + \bar{\varphi}_{i,j+1} + h^2 \varphi_{n-1-i,n-1-j}. \tag{6.23}$$

Proof. We can rewrite terms in (6.23) as follows and successively combine them

$$\begin{aligned}
&\bar{\varphi}_{i+1,j} - \bar{\varphi}_{i+1,j+1} + h^2 \varphi_{n-1-i,n-1-j} + \bar{\varphi}_{i,j+1} \\
&= \int_0^{1-(x_i^1+h)} \int_0^{1-x_j^2} \varphi(y_1, y_2)\, dy_1\, dy_2 - \int_0^{1-(x_i^1+h)} \int_0^{1-(x_j^2+h)} \varphi(y_1, y_2)\, dy_1\, dy_2 \\
&+ \int_{1-(x_i^1+h)}^{1-x_i^1} \int_{1-(x_j^2+h)}^{1-x_j^2} \varphi(y_1, y_2)\, dy_1\, dy_2 \int_0^{1-x_i^1} \int_0^{1-(x_j^2+h)} \varphi(y_1, y_2)\, dy_1\, dy_2 \\
&= \int_0^{1-(x_i^1+h)} \int_{1-(x_j^2+h)}^{1-x_j^2} \varphi(y_1, y_2)\, dy_1\, dy_2 + \int_{1-(x_i^1+h)}^{1-x_i^1} \int_{1-(x_j^2+h)}^{1-x_j^2} \varphi(y_1, y_2)\, dy_1\, dy_2 \\
&+ \int_0^{1-x_i^1} \int_0^{1-(x_j^2+h)} \varphi(y_1, y_2)\, dy_1\, dy_2 = \int_0^{1-x_i^1} \int_{1-(x_j^2+h)}^{1-x_j^2} \varphi(y_1, y_2)\, dy_1\, dy_2 \\
&+ \int_0^{1-x_i^1} \int_0^{1-(x_j^2+h)} \varphi(y_1, y_2)\, dy_1\, dy_2 = \int_0^{1-x_i^1} \int_0^{1-x_j^2} \varphi(y_1, y_2)\, dy_1\, dy_2.
\end{aligned}$$

□

The computation of the sink aggregation integral is presented in Algorithms 20. The computation of the the piecewise linear function $\bar{\varphi}$ (6.22) using the formula (6.23) is shown in line 12 of the Algorithm 20.

We will demonstrate the formula (6.23) and the procedure presented in line 12 of the Algorithm 20 on the example of a single grid point. Computing the integral (6.22) at the grid point (x_3^1, x_2^2) is equivalent to the integration of the function φ

Algorithm 20 $q = \text{separableSinkAggregation\_2D}(n, f, a, b)$;
(Evaluate the aggregation sink integral (6.3) using piecewise constant approximations on an equidistant mesh.)

Input: $n \in \mathbb{N}$; (intervals in each property dimension)
$f \in \mathbb{R}^{n \times n}$; (piecewise constant density function values)
$a, b \in \mathbb{R}^{n \times n}$; (kernel factors)
Output: $q \in \mathbb{R}^{n \times n}$; (piecewise const. aggreg. $q_{i,j} \approx Q_{\text{sink}}(f)$ on cell $\mathcal{C}_{i,j}$)

1: **for** $i = 0 : n-1$ **do**
2: **for** $j = 0 : n-1$ **do**
3: $\psi_{i,j} = a_{i,j} \cdot f_{i,j}$;
4: $\varphi_{i,j} = b_{i,j} \cdot f_{i,j}$;
5: **end for**
6: **end for**
7: **for** $i = 0 : n$ **do**
8: $\bar{\varphi}_{i,n} = 0; \qquad \bar{\varphi}_{n,i} = 0;$
9: **end for**
10: **for** $i = n-1 : 0$ **do**
11: **for** $j = n-1 : 0$ **do**
12: $\bar{\varphi}_{i,j} = \bar{\varphi}_{i,j+1} + \bar{\varphi}_{i+1,j} - \bar{\varphi}_{i+1,j+1} + h^2 \varphi_{n-1-i,n-1-j}$;
13: **end for**
14: **end for**
15: **for** $i = 0 : n-1$ **do**
16: **for** $j = 0 : n-1$ **do**
17: $q_{i,j} = \psi_{i,j} \left(\frac{3i+2}{12i+6}(\bar{\varphi}_{i+1,j+1} + \bar{\varphi}_{i,j} - \bar{\varphi}_{i+1,j} - \bar{\varphi}_{i,j+1}) + \frac{3i+2}{6i+3}(\bar{\varphi}_{i+1,j} - \bar{\varphi}_{i,j}) \right.$
18: $\left. + \frac{\bar{\varphi}_{i,j+1} - \bar{\varphi}_{i,j}}{2} + \bar{\varphi}_{i,j} \right)$;
19: **end for**
20: **end for**

over the surface of the dashed cells $\mathcal{C}_{0,0}$ and $\mathcal{C}_{0,1}$. The first three terms on the right hand side of the equation on line 12 yield the integral over the surface of the cell $\mathcal{C}_{0,0}$ that has been computed in the previous step of the backward loop and is the value of the integral (6.22) at the grid point (x_3^1, x_3^2), since $\bar{\varphi}(x_2^4, x_3^4) = \bar{\varphi}(x_3^1, x_3^2)$=0. The last term of the equation on line 12 yields the surface of the cell $\mathcal{C}_{0,1}$, which is the product of the function value in the cell $\mathcal{C}_{0,1}$ and the factor h^2 appearing due to double integration.

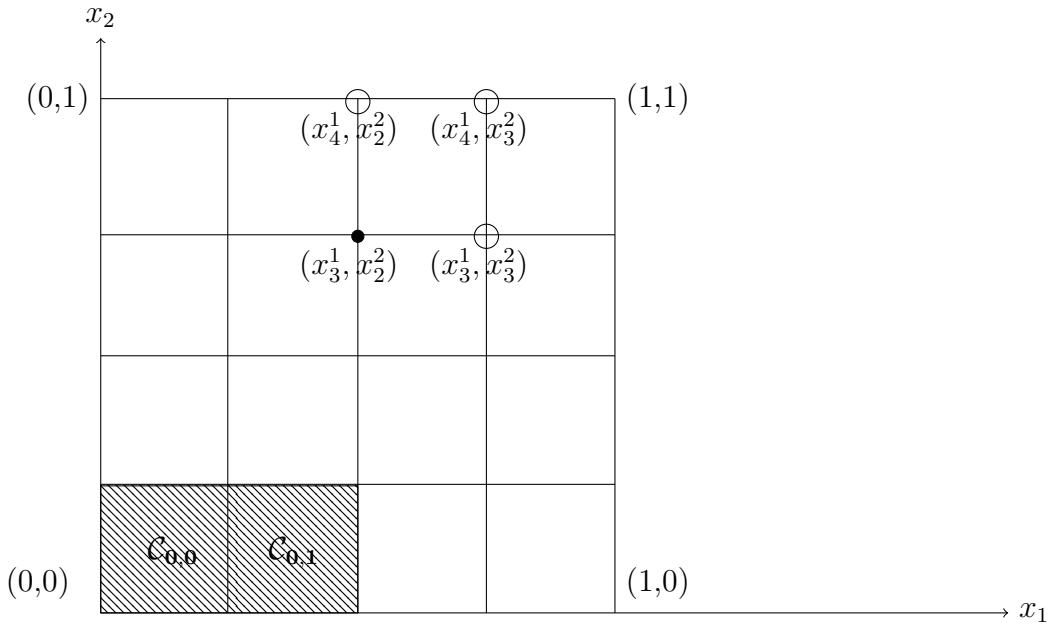


Figure 6.3: Uniform two-dimensional grid, $n = 4$.

Finally, in line 17 of Algorithm 20 we use the projection formula (6.20) and multiply it with the remaining part of the sink term $\psi(x_1, x_2)$ obtaining the final result.

6.4 Numerical results

In this section we show numerical results that were obtained using the algorithms presented in chapter 6. In our tests, we use one of the kernel functions (6.24), (6.25), (6.26).

$$\kappa_{\mathrm{C2}}(x_1, x_2, y_1, y_2) := 1 \qquad \text{constant kernel} \qquad (6.24)$$

$$\kappa_{\mathrm{B2}}(x_1, x_2, y_1, y_2) := (x_1^{\frac{1}{3}} + y_1^{\frac{1}{3}}) \cdot (x_1^{-\frac{1}{3}} + y_1^{-\frac{1}{3}}) \qquad \text{Brownian kernel} \qquad (6.25)$$

$$\kappa_{\Sigma 2}(x_1, x_2, y_1, y_2) := x_1 + x_2 + y_1 + y_2 \qquad \text{sum kernel} \qquad (6.26)$$

In (6.24) we assume that the kernel function is constant with respect to both properties, in (6.25) we assume that the aggregation kernel depends only on one of the particle properties, i.e., $\kappa_{\mathrm{B2}}(x_1, x_2, y_1, y_2) = \bar{\kappa}_{\mathrm{B2}}(x_1, y_1) \cdot \hat{\kappa}_{\mathrm{B2}}(x_2, y_2)$, with $\hat{\kappa}_{\mathrm{B2}}(x_2, y_2) = 1$, and finally in (6.26) the kernel function depends on both particle properties.
As an initial distribution we choose

$$f_5(0, x_1, x_2) := e^{-1000(x_1-0.125)^2} e^{-1000(x_2-0.125)^2}. \tag{6.27}$$

Figures 6.4 and 6.5 show the initial density distribution $f_5(0, x_1, x_2)$ (6.27) (top and bottom left) and the density distribution after 11000 time steps ($dt = 0.1$) for the sum kernel (6.26) (top and bottom right). While the figures on the top show property coordinates x_1 and x_2 only, the figures on the bottom are the three-dimensional equivalents, with density distribution f plotted against property coordinates x_1 and x_2. Figure 6.4 is plotted for $n = 32$ and Figure 6.5 for $n = 64$ grid points in each property dimension. We see that the approximated initial distribution and therefore also the computed result differs for $n = 32$ and $n = 64$, since $32^2 = 1024$ cells ($n = 32$) are not yet able to provide a good approximation.

In Figure 6.6 we show the density distribution after 5000 time steps (dt=0.1) for the Brownian kernel (6.25) and again for the initial distribution f_5 (6.27) (see Figure 6.5 top and bottom left) for $n = 64$ grid points in each dimension.

In all three Figures 6.4, 6.5 and 6.6 we notice that if we start with a distribution of particles with small mass in both dimensions, i.e., initial distributions placed on the left bottom corner of the domain (such as the initial distribution f_5 (6.27)), only a small part of the domain (mainly the part around the diagonal) is interesting for covering the evolution of the distribution. This result encourages the development of locally refined grids or refinement in hierarchical fashion around the diagonal which will lead to further reduction of computational time.

In table 6.1 we show the times consumed by the numerical computation of the source term for the constant kernel κ_{C2}. The comparison includes the source term computed directly with the explicit evaluation of quadruple sums, or with the separable approach and fast Fourier transformation. The table confirms the complexities $\mathcal{O}(2n^2 \log n)$ for the newly introduced approach and $\mathcal{O}(n^4)$ for the direct evaluation.

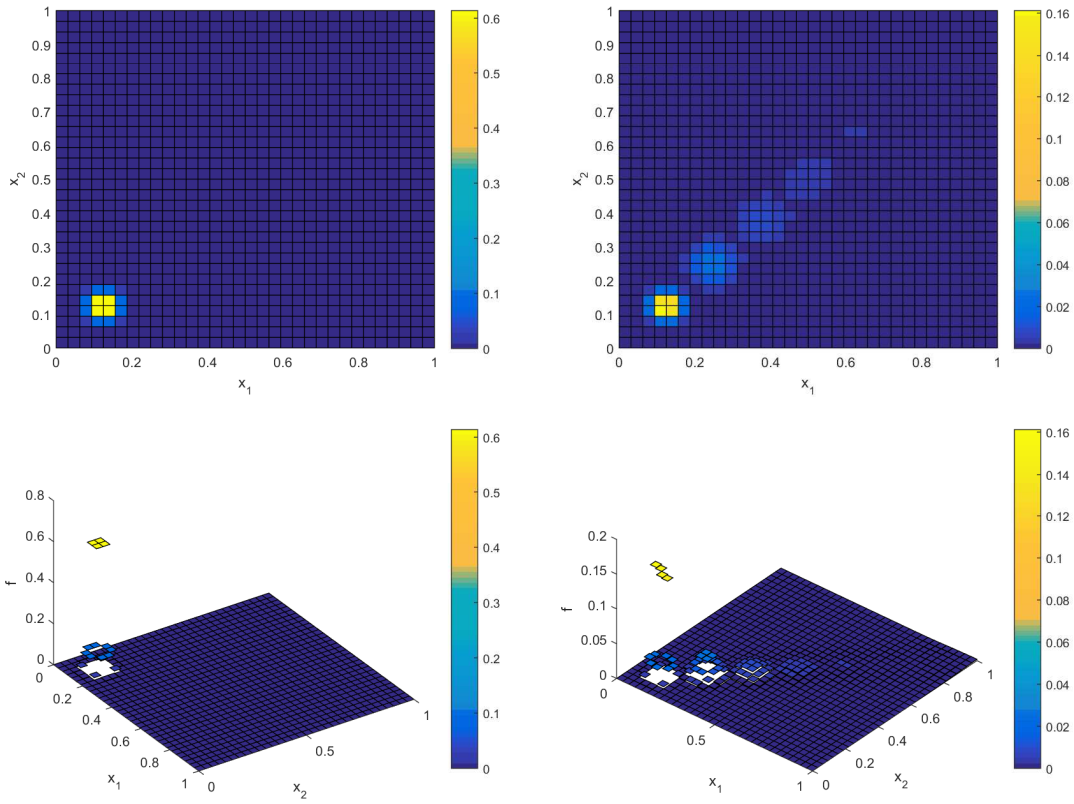


Figure 6.4: Density distribution after 11000 time steps ($dt = 0.1$) for initial distribution $f_5(0, x_1, x_2)$ (6.27), $\kappa_{\Sigma 2}$ (6.26) on a uniform grid, $n = 32$.

Intervals in each property dimension (n)	16	32	64	128	256
FFT	0.193	0.76	3.11	14	57
no FFT	0.16	2.5	38.1	676	7000+

Table 6.1: Times (in seconds) to compute the bivariate source terms for $f_5(0, x_1, x_2)$, $\kappa_{\mathrm{C}2}(x_1, x_2, y_1, y_2) = 1$, 100 time steps ($dt = 0.01, T = 1$).

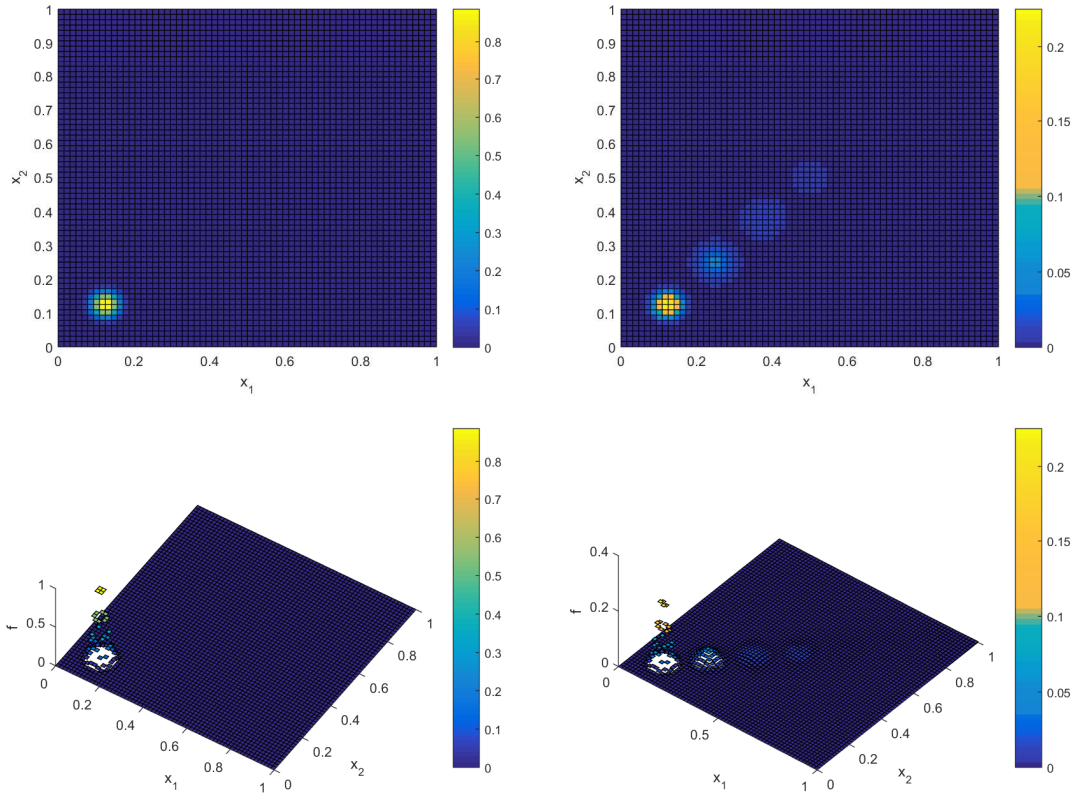


Figure 6.5: Density distribution after 11000 time steps ($dt = 0.1$) for initial distribution $f_5(0, x_1, x_2)$ (6.27), $\kappa_{\Sigma 2}$ (6.26) on a uniform grid, $n = 64$.

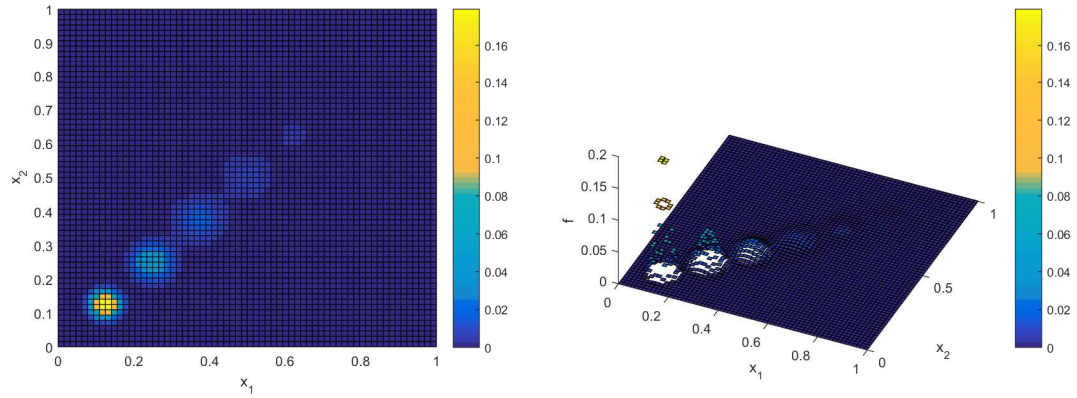


Figure 6.6: Density distribution after 5000 time steps ($dt = 0.1$) for initial distribution $f_5(0, x_1, x_2)$ (6.27), $\kappa_{\mathrm{B}2}$ (6.25) on a uniform grid, $n = 64$.

Chapter 7

Conclusions and Outlook

The subject of this work is the development and numerical illustration of efficient techniques to compute aggregation integrals that appear in population balance equations. The prerequisites for the fast and accurate evaluation of aggregation integrals are kernel separation and fast Fourier transformation that are introduced in sections 3.1 and 3.3. The evaluation of aggregation integrals is accomplished for four kind of problems

- univariate integrals approximated through piecewise constant functions on equidistant grids,
- univariate integrals approximated through piecewise constant functions on nested grids refined toward the origin,
- univariate integrals approximated through high order polynomial on nested grids refined toward an arbitrary point,
- bivariate integrals approximated through piecewise constant functions on equidistant tensor grids.

Suitable grids and spaces for the first three problems are described in section 3.2 and each of the problems is discussed in sections 4.1, 4.2 and chapter 5, respectively. The fourth, bivariate, problem is discussed in chapter 6.

In section 4.1, we have presented the basic algorithms for the discretization of the aggregation source and sink integrals using piecewise constant functions on equidistant grids, and the efficient evaluation of the resulting discrete equations through separable kernel approximations and the use of fast Fourier transformations. We have illustrated this approach for representative kernels from the literature. The Brownian kernel which is separable with rank 3, the gravitational kernel which re-

quires blockwise separable approximations in a hierarchical fashion, and the shear and kinetic kernels which can be successfully approximated through separable expansions using either Chebyshev polynomials or adaptive cross approximation. Our tests show that the adaptive cross approximation method is preferable over Chebyshev interpolation since it reaches highly accurate results already with rank-1 or rank-2 approximations (depending on the density distribution) because the behavior of the kernel is computed near the peak of the density distribution, while Chebyshev polynomials require a higher rank-5 approximation to reach acceptable accuracy. The algorithms are straightforward to implement, and the numerical tests confirm the superior complexity of the separable approach for all tested kernel functions: the almost linear $\mathcal{O}(n \log n)$ complexity of the FFT approach for the Brownian, shear and kinetic kernels, and $\mathcal{O}(n \log^2 n)$ for the gravitational kernel versus the quadratic complexity of the direct approach leads to a significant advantage for the FFT approach for problem sizes $n \geq 512$ or $n \geq 1024$ depending on the separation rank of the kernel.

In section 4.2 we illustrated a wavelet technique that allows to compute approximations to aggregation integrals on locally refined nested grids in complexity $\mathcal{O}(N \log N)$ where N denotes the discrete problem size. Numerical tests illustrate the superior performance compared to the well-known fixed pivot technique on a geometric grid which has computational complexity $\mathcal{O}(N^2)$. Besides kernel separation and fast Fourier transformation, another main ingredient of the wavelet method is the fast transformation between a nodal and a wavelet basis. Here, the particular nested grid can be seen as a drawback to the method: The fixed pivot technique is defined for general unstructured grids (we refer to [43] for a convergence analysis with respect to the underlying grid) whereas the wavelet method as described in this section is restricted to the nested grid which is recursively refined toward the origin.

The subject of chapter 5 is an efficient technique to compute aggregation integrals where the density function is approximated by a piecewise polynomial of order $p \geq 0$ on a nested (i.e., locally refined) grid refined toward an arbitrary point. Whereas a straightforward implementation would lead to complexity $\mathcal{O}((p+1)^2 N^2)$ where N denotes the number of intervals, the proposed algorithm leads to the reduced complexity $\mathcal{O}((p+1)^2 N \log N)$. Here, the additional key factors to yield this reduction are orthogonality of basis functions and efficient recursion formulas. While the observations from this chapter lead to heuristic recommendations which, in the simplest form, may be stated as "use piecewise linear polynomials on a uniform grid", they do depend on several input factors such as the choice of kernel function, the choice of initial distribution, the number of time steps that have been computed, and the choice of the grid, i.e., of the parameters n, p, L as well as the

shifts for local refinement. There may indeed exist scenarios where local refinement is the method of choice. However, these current results somewhat discourage the development of further generalizations such a locally varying polynomial degrees or moving grids that change as the density distribution changes over time.

Finally, in chapter 6 we present simple and efficient algorithms to compute bivariate aggregation source and sink integrals using piecewise constant functions on uniform meshes. The straight-forward evaluation of bivariate aggregation integrals may lead to $\mathcal{O}(n^4)$ complexity. Using a separable approximation of the kernel function and two-dimensional fast Fourier transform we reach $\mathcal{O}(2n^2 \log n)$ complexity. While the results in chapter 5 discourage the development of nested grids for univariate problems, the situation differs when we extend the algorithm to bivariate settings where not only mass but additional particle properties are considered. The numerical tests in this section highly encourage refinement of grids or blockwise decomposition in hierarchical fashion for bivariate problems.

Based on the results carried out in this work we mention the following problems that might be interesting for future development.

- We showed in chapters 4 and 5 that for the majority of test problems uniform grids with piecewise constant and piecewise linear functions can provide fairly good solutions and the further extension to locally refined grids and polynomial of order $p \geq 2$ is not encouraged, but the situation changes for bivariate problems. Based on the results in chapter 6, the development of nested grids as in section 4.2 or refinement of the grid in hierarchical fashion around the diagonal and possibly also approximation of the density function with high order polynomials appear to be interesting for two-dimensional problems.
- The computation of two-dimensional aggregation integrals is extremely expensive since the straight-forward evaluation may lead to $\mathcal{O}(n^{2d})$ complexity, with d number of particle properties. Hence, the most promising approach reached for multidimensional problems can be extended for evaluation of multidimensional aggregation integrals.

Appendix A

Auxiliary ξ-coefficients

Table A.1 gives the coefficients ξ^*, through which coefficients ξ used in section 5.1 can be computed.

$m \backslash n$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1	1	0	-1	0	2	0	-5	0	14	0	-42	0	132	0	-429	0
1		$\frac{1}{3}$	1	1	$-\frac{2}{3}$	-2	1	5	-2	-14	$\frac{14}{3}$	42	12	-132	33	429	$\frac{286}{3}$
2			$\frac{1}{5}$	1	2	1	-3	$-\frac{19}{5}$	6	12	-14	-38	36	123	-99	-407	286
3				$\frac{1}{7}$	1	3	4	-1	-10	-4	$\frac{176}{6}$	22	-67	-87	188	319	-550
4					$\frac{1}{9}$	1	4	$\frac{25}{3}$	6	-12	$-\frac{80}{3}$	13	85	$\frac{43}{9}$	-260	-113	$\frac{2398}{3}$
5						$\frac{1}{11}$	1	5	14	20	-1	-53	-50	113	229	-215	$-\frac{9206}{11}$
6							$\frac{1}{13}$	1	6	21	43	35	-58	-170	-7	509	406
7								$\frac{1}{15}$	1	7	$\frac{88}{3}$	77	$\frac{556}{5}$	$\frac{14}{3}$	-304	-397	$\frac{1498}{3}$
8									$\frac{1}{17}$	1	8	39	124	246	208	-321	-1090
9										$\frac{1}{19}$	1	9	50	186	461	657	46
10											$\frac{1}{21}$	1	10	$\frac{187}{3}$	265	781	1494
11												$\frac{1}{23}$	1	11	76	363	1234
12													$\frac{1}{25}$	1	12	91	482
13														$\frac{1}{27}$	1	13	$\frac{322}{3}$
14															$\frac{1}{29}$	1	14
15																$\frac{1}{31}$	1
16																	$\frac{1}{33}$

Table A.1: $\xi^*_{n,m}$-coefficients from which $\xi_{n,m}$ can be computed through $\xi_{n,m} = 2^{-n-\frac{1}{2}}\,\xi^*_{n,m}\sqrt{(2n+1)(2m+1)}$. Below the diagonal, ie. for $m > n$, equal to 0.

Appendix B

Auxiliary γ-coefficients

Here, we present explicit values of the γ-coefficients (5.5). As mentioned in section 5.1 we only need coefficients for $\nu = 0$, $\ell'' = \ell' = \ell = 0$. Since the computation of $\sum_3$ requires polynomial up to degree $2p + 1$ (see 5.20), to obtain results for $0 \leq p \leq 5$, we need to know $\gamma^{\ell'',\ell',\ell}_{\nu,(\alpha,\beta,\chi)}$ for $0 \leq \alpha \leq 11$, $0 \leq \beta, \chi \leq 5$. Table B.1 shows values for $\alpha = 0, 1$, Table B.2 for $\alpha = 2, 3, 4, 5$ and Table B.3 for $\alpha = 6, 7, 8, 9, 10, 11$, all three tables for $\beta, \chi = 1, \ldots, 5$.

$\beta\backslash\chi$	0	1	2	3	4	5
0	5.000000000E-1	-2.886751346E-1	0	0	0	0
1	-2.886751346E-1	0	1.290994449E-1	0	0	0
2	0	1.290994449E-1	0	-8.451542547E-2	0	0
3	0	0	-8.451542547E-2	0	6.299407883E-2	0
4	0	0	0	6.299407883E-2	0	-5.025189076E-2
5	0	0	0	0	-5.025189076E-2	0
$\alpha = 0$						

$\beta\backslash\chi$	0	1	2	3	4	5
0	2.886751346E-1	0	-1.290994449E-1	0	0	0
1	0	-3.464101615E-1	2.236067977E-1	3.779644730E-2	0	0
2	-1.290994449E-1	2.236067977E-1	8.247860988E-2	-1.463850109E-1	-1.844277784E-2	0
3	0	3.779644730E-2	-1.463850109E-1	-3.849001795E-2	1.091089451E-1	1.096586158E-2
4	0	0	-1.844277784E-2	1.091089451E-1	2.249416633E-2	-8.703882798E-2
5	0	0	0	1.096586158E-2	-8.703882798E-2	-1.480385306E-2
$\alpha = 1$						

Table B.1: Values of $\gamma^{0,0,0}_{0,(\alpha,\beta,\chi)}$ for $0 \leq \alpha \leq 1$.

$\beta \backslash \chi$	0	1	2	3	4	5
0	0	1.290994449E-1	0	-8.451542547E-2	0	0
1	1.290994449E-1	-2.236067977E-1	-8.247860988E-2	1.463850109E-1	1.844277784E-2	0
2	0	-8.247860988E-2	3.194382825E-1	-1.259881577E-1	-7.142857143E-2	-7.178841538E-3
3	-8.451542547E-2	1.463850109E-1	-1.259881577E-1	-1.490711985E-1	1.178093810E-1	4.247059929E-2
4	0	1.844277784E-2	-7.142857143E-2	1.178093810E-1	8.711953159E-2	-1.012534592E-1
5	0	0	-7.178841538E-3	4.247059929E-2	-1.012534592E-1	-5.733507635E-2
$\alpha = 2$						

$\beta \backslash \chi$	0	1	2	3	4	5
0	0	0	8.451542547E-2	0	-6.299407883E-2	0
1	0	3.779644730E-2	-1.463850109E-1	-3.849001795E-2	1.091089451E-1	1.096586158E-2
2	8.451542547E-2	-1.463850109E-1	1.259881577E-1	1.490711985E-1	-1.178093810E-1	-4.247059929E-2
3	0	-3.849001795E-2	1.490711985E-1	-2.725925593E-1	3.030303030E-2	8.504166129E-2
4	-6.299407883E-2	1.091089451E-1	-1.178093810E-1	3.030303030E-2	1.823744660E-1	-6.720751945E-2
5	0	1.096586158E-2	-4.247059929E-2	8.504166129E-2	-6.720751945E-2	-1.259881577E-1
$\alpha = 3$						

$\beta \backslash \chi$	0	1	2	3	4	5
0	0	0	0	6.299407883E-2	0	-5.025189076E-2
1	0	0	1.844277784E-2	-1.091089451E-1	-2.249416633E-2	8.703882798E-2
2	0	1.844277784E-2	-7.142857143E-2	1.178093810E-1	8.711953159E-2	-1.012534592E-1
3	6.299407883E-2	-1.091089451E-1	1.178093810E-1	-3.030303030E-2	-1.823744660E-1	6.720751945E-2
4	0	-2.249416633E-2	8.711953159E-2	-1.823744660E-1	2.007992008E-1	4.969967218E-2
5	-5.025189076E-2	8.703882798E-2	-1.012534592E-1	6.720751945E-2	4.969967218E-2	-1.794871795E-1
$\alpha = 4$						

$\beta \backslash \chi$	0	1	2	3	4	5
0	0	0	0	0	5.025189076E-2	0
1	0	0	0	1.096586158E-2	-8.703882798E-2	-1.480385306E-2
2	0	0	7.178841538E-3	-4.247059929E-2	1.012534592E-1	5.733507635E-2
3	0	1.096586158E-2	-4.247059929E-2	8.504166129E-2	-6.720751945E-2	-1.259881577E-1
4	5.025189076E-2	-8.703882798E-2	1.012534592E-1	-6.720751945E-2	-4.969967218E-2	1.794871795E-1
5	0	-1.480385306E-2	5.733507635E-2	-1.259881577E-1	1.794871795E-1	-1.150563653E-1
$\alpha = 5$						

Table B.2: Values of $\gamma^{0,0,0}_{0,(\alpha,\beta,\chi)}$ for $2 \leq \alpha \leq 5$.

$\beta \backslash \chi$	0	1	2	3	4	5
0	0	0	0	0	0	4.181210050E-2
1	0	0	0	0	7.278552446E-3	-7.242068244E-2
2	0	0	0	3.551569931E-3	-2.818971241E-2	8.726171907E-2
3	0	0	3.551569931E-3	-2.101137107E-2	6.035579363E-2	-7.374961314E-2
4	0	7.278552446E-3	-2.818971241E-2	6.035579363E-2	-7.564093585E-2	8.198451078E-3
5	4.181210050E-2	-7.242068244E-2	8.726171907E-2	-7.374961314E-2	8.198451078E-3	1.033265071E-1
$\alpha = 6$						

$\beta \backslash \chi$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	5.186152788E-3
2	0	0	0	0	2.018707235E-3	-2.008588338E-2
3	0	0	0	1.504655534E-3	-1.194283306E-2	4.442521691E-2
4	0	0	2.018707235E-3	-1.194283306E-2	3.664278773E-2	-6.516846365E-2
5	0	5.186152788E-3	-2.008588338E-2	4.442521691E-	2 -6.516846365E-2	4.683531500E-2
$\alpha = 7$						

$\beta \backslash \chi$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	1.257826807E-3
3	0	0	0	0	7.478892157E-4	-7.441403750E-3
4	0	0	0	7.478892157E-4	-5.936186627E-3	2.358624958E-2
5	0	0	1.257826807E-3	-7.441403750E-3	2.358624958E-2	-4.811435476E-2
$\alpha = 8$						

$\beta \backslash \chi$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	4.140507769E-4
4	0	0	0	0	3.302982550E-4	-3.286426189E-3
5	0	0	0	4.140507769E-4	-3.286426189E-3	1.349504317E-2
$\alpha = 9$						

$\beta \backslash \chi$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	1.645270785E-4
5	0	0	0	0	1.645270785E-4	-1.637023960E-3
$\alpha = 10$						

$\beta \backslash \chi$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	7.448707785E-5
$\alpha = 11$						

Table B.3: Values of $\gamma^{0,0,0}_{0,(\alpha,\beta,\chi)}$ for $6 \leq \alpha \leq 11$.

Bibliography

[1] D. J. Aldous. Deterministic and stochastic models for coalescence (aggregation and coagulation): A review of the mean-field theory for probabilists. *Bernoulli*, 5(1):3–48, 1999.

[2] A. Alexopoulos and C. Kiparissides. Solution of the bivariate dynamic population balance equation in batch particulate systems: Combined aggregation and breakage. *Chemical Engineering Science*, 62:5048–5053, 2007.

[3] F. Anker, S. Ganesan, V. John, and E. Schmeyer. A comparative study of a direct discretization and an operator-splitting solver for population balance systems. *Computers and Chemical Engineering*, 70:95–104, Apr. 2015.

[4] M. M. Attarakih, H.-J. Bart, and N. M. Faqir. Optimal moving and fixed grids for the solution of discretized population balances in batch and continuous systems: Droplet breakage. *Chemical Engineering Science*, 58:1251–1269, 2003.

[5] J. Barrett and J. Jheeta. Improving the accuracy of the moments method for solving the aerosol general dynamic equation. *Journal of Aerosol Science*, 27:31–39, 1996.

[6] M. Bebendorf and S. Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70:1–24, 2003.

[7] M. Bennett and S. Rohani. Solution of population balance equations with a new combined lax-wendro/crank-nicholson method. *Chemical Engineering Science*, 56:6623–6633, 2001.

[8] S. Börm and L. Grasedyck. Hybrid cross approximation of integral operators. *Numerische Mathematik*, 101:221–249, 2005.

[9] S. Bove, T. Solberg, and B. Hjertager. A novel algorithm for solving population balance equations: The parallel parent and daughter classes. *Chemical Engineering Science*, 60:1449–1464, 2005.

[10] R. N. Bracewell. *The Fourier transform and its applications*. McGraw-Hill Science/Engineering/Math, 3rd edition, 1999.

[11] A. Bramley, M. Hounslow, and R. Ryall. Aggregation during precipitation from solution: A method for extracting rates from experimental data. *Journal of Colloid and Interface Science*, 183:155–165, 1996.

[12] J. Chakraborty and S. Kumar. A new framework for solution of multidimensional population balance equations. *Chemical Engineering Science*, 62:4112–4125, 2007.

[13] S. Chang. The method of space-time conservation element and solution element - A new approach for solving the navier-stokes and euler equations. *Journal of Computational Physics*, 119:295–324, 1995.

[14] S. Chauhan, A. Chiney, and S. Kumar. On the solution of bivariate population balance equations for aggregation: X-discretization of space for expansion and concentration of computational domain. *Chemical Engineering Science*, 70:135–145, 2012.

[15] A. Chiney and S. Kumar. On the solution of bivariate population balance equations for aggregation: Pivotwise expansion of solution domain. *Chemical Engineering Science*, 76:12–25, 2012.

[16] E. Chu and A. George. *Inside the FFT black box: Serial and parallel fast Fourier transform algorithms*. CRC Press, 2000.

[17] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[18] P. Duhamel and M. Vetterli. Improved Fourier and Hartley transform algorithms: Application to cyclic convolution of real data. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 818–824, 1987.

[19] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM Journal on Scientific Computing*, 14:1368–1393, 1993.

[20] R. Filbert and P. Laurencot. Numerical simulation of the Smoluchowski coagulation equation. *SIAM Journal on Scientific Computing*, 25:2004–2028, 2004.

[21] R. Freund and R. Hoppe. *Stoer/Bulirsch: Numerische Mathematik 1*. Springer, 10th edition, 2007.

[22] F. Gelbard, Y. Tambour, and J. Seinfeld. Sectional representations for simulating aerosol dynamics. *Journal of Colloid and Interface Science*, 76:541–556, 1980.

[23] D. Gillette. A study of aging of lead aerosols. II. A numerical model simulating coagulation and sedimentation of a leaded aerosol in the presence of an unleaded background aerosol. *Atmospheric Environment*, 6:433–512, 1972.

[24] D. Ginter and S. K. Loyalka. Apparent size-dependent growth in aggregating crystallizers. *Chemical Engineering Science*, 51(14):3685—3695, 1992.

[25] A. K. Giri and E. Hausenbals. Convergence analysis of sectional methods for solving aggregation population balance equations: The fixed pivot technique. *Nonlinear Analysis: Real World Applications*, 14(6):2068–2090, 2013.

[26] L. Grasedyck. Adaptive recompression of $\mathcal{H}$-matrices. *Computing*, 74:205–223, 2005.

[27] S. Grosch, H. Briesen, W. Marquardt, and M. Wullkow. Generalization and numerical investigation of qmom. *A.I.Ch.E. Journal*, 53:207–227, 2007.

[28] W. Hackbusch. On the efficient evaluation of coalescence integrals in population balance models. *Computing*, 78:145–159, 2006.

[29] W. Hackbusch. Approximation of coalescence integrals in population balance models with local mass conservation. *Numerische Mathematik*, 106:627–657, 2007.

[30] W. Hackbusch. Convolution of hp-functions on locally refined grids - extended version. *Technical report 38/2007, Max-Planck-Institute for Mathematics in the Sciences*, June 2007.

[31] W. Hackbusch. Fast and exact projected convolution for non-equidistant grids. *Computing*, 80:137–168, 2007.

[32] W. Hackbusch. Fast and exact projected convolution of piecewise linear functions on non-equidistant grids. In P. M.H.Breitner, G.Denk, editor, *From Nano to Space. Applied Mathematics inspired*, pages 145–160. Springer, 2008.

[33] W. Hackbusch. *Hierarchical Matrices*. Springer, 2015.

[34] W. Hackbusch. Convolution of hp-functions on locally refined grids. *IMA Journal of Numerical Analysis*, 2009:960–985, 29.

[35] S. Heinrich, M. Peglow, M. Ihlow, M. Henneberg, and L. Mörl. Analysis of the start-up process in continuous fluidized bed spray granulation by population balance modelling. *Chemical Engineering Science*, 22:73–92, 1995.

[36] M. Hounslow, R. Ryall, and V. Marshall. A discretized population balance for nucleation, growth and aggregation. *AIChE Journal*, 38:1821–1832, 1988.

[37] S. Iveson, J. Litster, K. Hapgood, and B. Ennis. Nucleation, growth and breakage phenomena in agitated wet granulation processes: A review. *Powder Technology*, 117:3–39, 2001.

[38] V. John and G. Matthies. Moonmd - a program package based on mapped finite element methods. *Computing and Visualization in Science*, 6:163–170, 2004.

[39] Y. Kim and J. Seinfeld. Simulation of multicomponent aerosol dynamics. *Journal of Colloid and Interface Science*, 149:425–449, 1992.

[40] F. Kruis, A. Maisels, and H. Fissan. Direct simulation monte carlo method for particle coagulation and aggregation. *AIChE Journal*, 46:1735–1742, 2000.

[41] J. Kumar, M. Peglow, G. Warnecke, and S. Heinrich. The cell average technique for solving multi-dimensional aggregation population balance equations. *Computers and Chemical Engineering*, 32:181–1830, 2008.

[42] J. Kumar, M. Peglow, G. Warnecke, S. Heinrich, and L. Mörl. Improved accuracy and convergence of discretized population balance for aggregation: The cell average technique. *Chem. Eng. Sci.*, 61:3327–3342, 2006.

[43] J. Kumar and G. Warnecke. Convergence analysis of sectional methods for solving breakage population balance equations - I: The fixed pivot technique. *Numerische Mathematik*, 111:81–108, 2008.

[44] R. Kumar, J. Kumar, and G. Warnecke. Numerical methods for solving two-dimensional aggregation population balance equations. *Computers and Chemical Engineering*, 35:999–1009, 2011.

[45] S. Kumar and D. Ramkrishna. On the solution of population balance equations by discretization - I. A fixed pivot technique. *Chemical Engineering Science*, 51:1311–1332, 1996.

[46] S. Kumar and D. Ramkrishna. On the solution of population balance equations by discretization - II. A moving pivot technique. *Chemical Engineering Science*, 51:1333–1342, 1996.

[47] S. Le Borne and L. Shahmuradyan. Algorithms for the Haar wavelet based fast evaluation of aggregation integrals in population balance equations. *Applied Numerical Mathematics*, 108:1–20, 2016.

[48] S. Le Borne and L. Shahmuradyan. Fast algorithms for hp-discretized univariate population balance aggregation integrals. *Computers and Chemical Engineering*, 97:1–12, 2017.

[49] S. Le Borne, L. Shahmuradyan, and K. Sundmacher. Fast evaluation of univariate aggregation integrals on equidistant grids. *Computers and Chemical Engineering*, 74:115–127, 2015.

[50] K. Lee and T. Matsoukas. Simultaneous coagulation and breakage using constant-n monte carlo. *Powder Technology*, 110:82–89, 2000.

[51] Y. Lin, K. Lee, and T. Matsoukas. Solution of the population balance equation using constant-number monte carlo. *Chemical Engineering Science*, 57:2241–2252, 2002.

[52] A. K. Louis, P. Maaß, and A. Rider. *Wavelets*. Teubner, 1998.

[53] G. Madras and B. McCoy. Reversible crystal growth-dissolution and aggregationbreakage: Numerical and moment solutions for population balance equations. *Powder Technology*, 143-144:297–307, 2004.

[54] A. Mahoney and D. Ramkrishna. Efficient solution of population balance equations with discontinuities by finite elements. *Chemical Engineering Science*, 57:1107–1119, 2002.

[55] P. Marchal, J. K. R. David, and J. Villermaux. Crystallization and precipitation engineering - I. An effcient method for solving population balance in crystallization with agglomeration. *Chemical Engineering Science*, 43:59–67, 1988.

[56] D. Marchisio and R. Fox. Solution of population balance equations using the direct quadrature method of moments. *Journal of Aerosol Science*, 36:43–73, 2005.

[57] D. L. Marchisio, R. D. Vigil, and R. Fox. Quadrature method of moments for aggregation- breakage processes. *Journal of Colloid and Interface Science*, 258:322–334, 2003.

[58] S. Motz, A. Mitrovic, and E. D. Gilles. Comparison of numerical methods for the simulation of dispersed phase systems. *Chemical Engineering Science*, 57:4329–4344, 2002.

[59] J. Mydlarz and D. Briedis. Growth-rate dispersion vs size-dependent growthrate for msmpr crystallizer data. *Computers and Chemical Engineering*, 16(9):917–922, 1992.

[60] M. Nicmanis and M. Hounslow. A finite element analysis of the steady state population balance equation for particulate systems: Aggregation and growth. *Computers and Chemical Engineering*, 20:S261–S266, 1996.

[61] M. Nicmanis and M. J. Hounslow. Finite-element methods for steady-state population balance equations. *AIChE Journal*, 44(10):2258–2272, 1998.

[62] H. Nussbaumer. Digital filtering using polynomial transforms. *Electronics Letters*, 13(13):386–387, 1977.

[63] H. Nussbaumer. Fast Fourier transform and convolution algorithms. *Springer Series in Information Sciences. Springer-Verlag*, 1981.

[64] M. Peglow. Beitrag zur Modellbildung von eigenschaftsverteilten dispersen Systemen am Beispiel der Wirbelschicht-Sprühagglomeration. *PhD thesis, Otto-von-Guericke-University Magdeburg*, 2005.

[65] D. Potts. Schnelle Fourier-Transformationen für nichtäquidistante Daten und Anwendungen. *Habilitation thesis, Universität zu Lübeck*, 2003.

[66] D. Potts, G. Steidl, and A. Nieslony. Fast convolution with radial kernels at nonequispaced knots. *Numerische Mathematik*, 98:329–351, 2004.

[67] D. Ramkrishna. *Population balances - Theory and Applications to Particulate Systems in Engineering*. Academic Press, 2000.

[68] S. Rigopoulos and A. Jones. Finite-element scheme for solution of the dynamic population balance equation. *AIChE Journal*, 49:1127–1139, 2003.

[69] K. Sastry and P. Gaschignard. Discretization procedure for the coalescence equation of particulate processes. *Industrial and Engineering Chemistry Fundamentals*, 8:355–361, 1981.

[70] M. Smith and T. Matsoukas. Constant-number monte carlo simulation of population balance. *Chemical Engineering Science*, 53:1777–1786, 1998.

[71] A. Sutugin and N. Fuchs. Formation of condensation aerosols under repidly changing environmental conditions. Theory and method of calculation. *Journal of Aerosol Science*, 1:287–293, 1970.

[72] F. Tolfo. A simplified model of aerosol coagulation. *Journal of Aerosol Science*, 8:9–19, 1977.

[73] H. Vale and T. McKenna. Solution of the population balance equation for two-component aggregation by an extended fixed pivot technique. *Industrial & Engineering Chemistry Research*, 20:7885–7891, 2005.

[74] D. Verkoeijen, G. Pouw, G. Meesters, and B. Scarlett. Population balances for particulate processes - a volume approach. *Chemical Engineering Science*, 57:2287–2303, 2002.

Curriculum vitae

Personal information

First Name	Lusine
Last Name	Shahmuradyan
Citizenship	Armenian
Date of Birth	12.08.1989
Place of Birth, Country	Yerevan, Armenia

Education and work experience

09.1996 - 06.2006	"Quantum" college, Yerevan, Armenia
09.2006 - 07.2010	Bachelor of Radiophysics, Yerevan State University, Armenia
09.2010 - 07.2011	Attending Master Courses, Yerevan State University, Armenia Joint M. Sc degree Program in Industrial Mathematics
09.2011 - 02.2012	University of L'Aquila, Italy
03.2012 - 08.2012	University of Nice Sophia Antipolis, France
09.2012 - 09.2013	University of Hamburg, Germany
10.2013 - 09.2016	Research Assistant, Institute of Mathematics, Hamburg University of Technology, Germany