Free University of Berlin
Department of Mathematics and Computer Science
Institute of Mathematics

THE FIXED PIVOT TECHNIQUE FOR POPULATION BALANCE EQUATIONS

A thesis presented for the
Bachelor of Science
in
Mathematics

Student *Max Rütten*
Matriculation number: 5172946

Submission Date 19.02.2025

Supervisor *Prof. Dr. Volker John,*
PD Dr. Alfonso Caiazzo

Freie Universität Berlin

Fachbereich Mathematik, Informatik und Physik

SELBSTSTÄNDIGKEITSERKLÄRUNG

Name: Rütten	(BITTE nur Block- oder Maschinenschrift verwenden.)
Vorname(n): Max	
Studiengang: B. Sc. Mathematik	
Matr. Nr.: 5172947	

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Bachelorarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht.

Datum: 19.02.2025 Unterschrift: _____

Contents

Chapter 1

Introduction

Many natural and industrial processes involve systems of particles that interact dynamically, changing in size or structure over time. Understanding and predicting how these particle distributions evolve is crucial in fields ranging from chemical engineering to environmental science.

To describe these complex interactions mathematically, we use population balance equations (PBEs). PBEs capture the time evolution of a particle population distributed across a continuous range of sizes. The processes of interest, breakage and aggregation, cause particles to either fragment into smaller pieces or combine to form larger clusters. These equations provide a framework for quantifying how the number and size of particles change under these competing mechanisms. While PBEs can be extended to include additional effects like particle growth, this work focuses specifically on breakage and aggregation.

The continuous form of a PBE is expressed as a integro-partial differential equation (IPDE), which represents the number density of particles as a function of time and particle size. However, the continuous nature of the equation makes it challenging to solve directly, especially for complex systems. To overcome this, numerical methods are employed to discretize the problem. An approach, based on the work of Kumar and Ramkrishna [KR96a], involves converting the continuous IPDE into a system of integro-ordinary differential equations (IODEs) by dividing the size domain into discrete intervals or cells.

The method we will use for this discretization is the fixed pivot method. Instead of treating the particle distribution as continuous, this method concentrates the mass of particles at specific representative points, known as pivots, within each interval. These pivots serve as anchor points for mass conservation and allow the equations to be solved with greater numerical stability. The concept of number density is crucial in this framework, providing a way to calculate the distribution of particles across the cells accurately.

This thesis presents a comprehensive exploration of the fixed pivot method for solving PBEs, focusing on its application to systems governed by breakage and aggregation. We begin by introducing the continuous IPDE form of the PBE and its discretization into a system of IODEs, as developed in prior research. We then detail the construction of the fixed pivot method and explain how it enables accurate simulations by concentrating the mass only at the pivots. Various factors that influence the evolution of the particle system are analyzed, including the aggregation

kernel, which describes the likelihood of particle collisions, the breakage frequency, which governs the rate at which particles fragment, and the nature of the breakage process, with a particular emphasis on binary breakage.

Following this theoretical foundation, the thesis delves into the numerical methods used to solve the resulting system of IODEs, highlighting techniques for ensuring stability and accuracy. We also outline the implementation process, discussing key challenges and the strategies employed to maintain mass conservation throughout the simulation. To validate the model, we present numerical studies that test both the conservation of mass and the logical physical behavior of the particle system. Finally, we address some known limitations of the fixed pivot method, such as its potential loss of accuracy under certain conditions, and briefly mention future directions for refining and improving the method.

Chapter 2

Notation

Symbol	Description
$v,\ v'$	volume of particles
v_i	grid points
x_i	center of grid intervals
x_1	lower bound pivot
x_M	upper bound pivot
t	time
T	end time
$n(v,t)$	number density of volume v at time t
$N(t)$	total number of particles at time t
$N_i(t)$	number of particles at time t of volume x_i
M	number of grid intervals
$Q(v,v')$	aggregation frequency of particles $v,\ v'$
$\Gamma_k = \Gamma(x_k)$	breakage frequency of x_k
$\gamma(v)$	number of daughter particles from breakage of mass v
h	grid step length
$a(v,x_i)$	portion of volume v distributed to x_i
$b(v,x_{i+1})$	portion of volume v distributed to x_{i+1}
η	combination of $a(v,x_i)$ and $b(v,x_{i+1})$
R_{Da}	death term due to aggregation
R_{Ba}	birth term due to aggregation
R_A	combined aggregation term
R_{Db}	death term due to breakage
R_{Bb}	birth term due to breakage
R_B	combined breakage term
$\delta_{k,i}$	Kronecker delta
$\beta(v,v')$	number of particles born in $[v, v+dv]$ due to v' breaking

Chapter 3

Continuous model for aggregation and breakage

In this chapter, we focus on the mathematical formulation of population balance equations and their role in describing particle systems. Specifically, we are interested in the evolution of the *number density* $n(v,t)$ over time t and particle volume v. The number density $n(v,t)$ represents the distribution of particles within a system, where $n(v,t)dv$ denotes the number of particles with volumes in the interval $[v, v+dv]$ at time t.

The formulation is based on the work of [KR96a], whose approach introduces a continuous integro-partial differential equation to describe the dynamics of particle systems. The PBE for simultaneous aggregation and breakage is given as

$$\begin{aligned}\frac{\partial n(v,t)}{\partial t} =& \frac{1}{2}\int_0^v n(v-v',t)n(v',t)Q(v-v',v')\,dv' \\ &- \int_0^\infty n(v,t)n(v',t)Q(v,v')\,dv' \\ &+ \int_v^\infty \beta(v,v')\Gamma(v')n(v',t)\,dv' \\ &- \Gamma(v)n(v,t).\end{aligned} \tag{3.1}$$

This equation models the time evolution of the number density $n(v,t)$ due to the aggregation and breakage processes across the continuous particle volume space over time in $(0,T]$, with final time T. In the following, we provide a brief introduction to the variables and functions that appear in this equation.

- $n(v,t)$: The number density function, representing the distribution of particles as a function of their volume v and time t.
- $Q(v,v')$: The aggregation kernel, which describes the rate at which two particles of volumes v and v' combine to form a larger particle. The kernel can depend on the physical properties of the system, such as the size of the particles and the interaction forces.
- $\beta(v,v')$: The daughter distribution function, which describes the fraction of particles of volume v formed as a result of the breakage of a volume parent

particle v'.

- $\Gamma(v)$: The breakage rate, which determines the frequency with which particles of volume v undergo breakage.

The terms in Equation (3.1) correspond to distinct physical processes:

1. The first term represents the birth of particles of volume v due to the aggregation of smaller particles.
2. The second term accounts for the death of particles of volume v as they aggregate with others to form larger particles.
3. The third term describes the birth of particles of volume v as a result of the breakage of larger particles.
4. The fourth term represents the death of particles of volume v as they undergo breakage.

Solving the IPDE numerically is challenging in part because of its integral terms that couple the number density across the particle volume space. To address this, we rely on the fixed pivot method, a numerical discretization technique that transforms the continuous equation into a system of integro-ordinary differential equations:

$$\frac{dN_i}{dt} = R_{Ba} - R_{Da} + R_{Bb} - R_{Db}, \tag{3.2}$$

where $R_{Ba},\ R_{Da},\ R_{Bb},\ R_{Db}$ represent the birth and death terms due to aggregation and breakage, respectively.
The next section develops the fixed pivot method in detail, explaining how the volume domain is discretized, how pivot points are chosen, and how birth/death terms are computed.

Chapter 4

Fixed pivot method

4.1 The fixed pivot method

4.1.1 Grid and pivot allocation

The fixed pivot method is a numerical method for discretizing the population balance equations. Instead of treating the particle distribution as continuous, this method concentrates the volumes of particles at specific representative points, known as pivots, within each interval. These pivots serve as anchor points for volume conservation and allow equations to be solved with greater numerical stability. The concept of number density is crucial in this framework, providing a way to calculate the distribution of particles across cells accurately.

Grids

To apply the fixed pivot method, we partition the continuous domain into a finite number of grid points, each representing the volume values of the particles. The grid serves as the basis for defining intervals within which we calculate the distribution of particles.

One of the simplest grids is the equidistant grid, where the grid points v_i are evenly spaced with a fixed step length h. The grid points are defined as:

$$v_i = ih.$$

Another type is the geometrical grid, where the grid points are spaced such that the distance between them increases (or decreases) exponentially according to the step ratio h. The grid points are defined as:

$$v_{i+1} = hv_i.$$

The geometrical grid is particularly useful for scenarios where particle size distributions follow an exponential pattern.

Pivots

Within the intervals defined by the grid points $[v_i, v_{i+1}]$, the pivots x_i are introduced to represent the midpoint of each interval. These pivots act as representative

volumes for their cells. The pivot x_i is defined as:

$$x_i = \frac{v_i + v_{i+1}}{2}.$$

For instance, if $v_1 = 1$ and $v_2 = 3$, then the pivot is calculated as:

$$x_1 = \frac{1+3}{2} = 2.$$

Volume allocation functions

To distribute a given volume v across adjacent pivots, we introduce the allocation functions $a(v, x_i)$ and $b(v, x_i)$, which proportionally distribute the volume v between the pivots x_i and x_{i+1}. These functions are defined as:

$$\begin{aligned} a(v, x_i) &= \frac{x_{i+1} - v}{x_{i+1} - x_i}, \quad \text{and} \quad a(v, x_i) = 0 \quad \text{for } v \notin [x_i; x_{i+1}], \\ b(v, x_{i+1}) &= \frac{v - x_i}{x_{i+1} - x_i}, \quad \text{and} \quad b(v, x_{i+1}) = 0 \quad \text{for } v \notin [x_i; x_{i+1}]. \end{aligned} \tag{4.1}$$

These fractions sum up to one, ensuring that the entire volume is allocated correctly:

$$a(v, x_i) + b(v, x_i) = \frac{x_{i+1} - v}{x_{i+1} - x_i} + \frac{v - x_i}{x_{i+1} - x_i} = 1.$$

For example, if $v = 3$, $x_1 = 0$, and $x_2 = 10$, the allocations are:

$$a(3, x_1) = \frac{10 - 3}{10 - 0} = \frac{7}{10}, \quad b(3, x_2) = \frac{3 - 0}{10 - 0} = \frac{3}{10}.$$

Dirac delta distribution

The fixed pivot method assumes that the volume of particles within each cell is concentrated at the pivot x_i. To mathematically model this concentration, we use the Dirac delta distribution, $\delta(v - x)$, which is defined as:

$$\int_{-\infty}^{\infty} \delta(v - x)\, dv = 1, \quad \text{and} \quad \delta(v - x) = 0 \quad \text{for } v \neq x.$$

Using the Dirac delta distribution, the number density $n(v, t)$ within a cell can be represented as a point located at the pivot x_i. This assumption aligns with the goal of the fixed pivot method of simplifying the continuous distribution into discrete representative points.

4.2 Discretization of the population balance equation

Having defined the grid and pivot points in the previous section, we now focus on transitioning from the continuous problem (3.1) to a discrete one. This involves discretizing the PBE to compute particle dynamics numerically.

The key quantity we are interested in is the number density, $n(v,t)$, which describes the concentration of particles around a given volume v at time t. By understanding $n(v,t)$, we can derive important properties of the system, such as the total number of particles, their total volume, and the distribution of particle volumes. These quantities are obtained using the moments of the number density.

Moments of the number density

The k-th moment of the number density is defined as:

$$M_k = \int_0^\infty v^k n(v,t)\, dv.$$

The zeroth moment, M_0, gives the total number of particles in the system. The first moment, M_1, represents the total volume of particles. Higher moments, such as the second moment, M_2, can describe other system properties, such as the distribution's variance.

Using the zeroth moment, we define $N_i(t)$ to be the number of particles at time t within the volume range $[v_i, v_{i+1}]$:

$$N_i(t) = \int_{v_i}^{v_{i+1}} n(v,t)\, dv. \tag{4.2}$$

Here, $N_i(t)$ quantifies the number of particles associated with the i-th cell in the grid.

Total number of particles

By summing over all grid intervals, we obtain the total number of particles in the system:

$$N(t) = \sum_{i=1}^{M} N_i(t),$$

where M is the total number of grid intervals. This provides a discrete measure of the system's size over time.

Instead of observing changes in the number density over the entire continuous domain, we now focus on the dynamics of particle numbers at discrete pivot points x_i within intervals $[v_i, v_{i+1}]$. This leads to the discrete form of the population balance equation:

$$\begin{aligned}
\frac{dN_i(t)}{dt} = \frac{1}{2}\int_{v_i}^{v_{i+1}} \int_0^v n(v-v',t)n(v',t)Q(v-v',v')\, dv'\, dv \\
- \int_{v_i}^{v_{i+1}} n(v,t) \int_0^\infty n(v',t)Q(v,v')\, dv'\, dv \\
+ \int_{v_i}^{v_{i+1}} \int_v^\infty \beta(v,v')\Gamma(v')n(v',t)\, dv'\, dv \\
- \int_{v_i}^{v_{i+1}} \Gamma(v)n(v,t)\, dv.
\end{aligned} \tag{4.3}$$

This equation accounts for particle interactions such as aggregation and breakage. In later sections, we will discuss each term in detail and derive discrete formulations suitable for computation.

Discrete number density

To compute the discrete form of the PBE, we need a discrete representation of the number density $n(v,t)$. Using the fixed pivot method, the number density is assumed to be concentrated at the pivot points x_k, resulting in the following discrete formulation:

$$n(v,t) = \sum_{k=1}^{M} N_k(t)\delta(v - x_k), \tag{4.4}$$

where $\delta(v - x_k)$ is the Dirac delta distribution. The Dirac delta ensures that the number density is only non-zero at the pivot points, reflecting the fixed pivot method's assumption of concentrated volume and ensuring that all particle properties are evaluated solely at the pivot points.

In this section, we introduced the moments of the number density and their role in characterizing the system. We defined $N_i(t)$, the number of particles in each interval, and derived the discrete form of the population balance equation. Furthermore, we introduced the discrete representation of $n(v,t)$ using the Dirac delta distribution.

In the following chapters, we will explore each term of the discrete PBE, adapt them for computational purposes, and ensure volume conservation.

Chapter 5

Aggregation

In the context of population balance equations, aggregation refers to the process by which two or more particles combine to form a larger particle. We need to observe two different results to understand the aggregation. First, the birth of a new bigger particle due to the combination of smaller particles. Secondly, the death of the smaller particles involved in the aggregation process. We will introduce a discrete formulation of both cases and combine them to acquire the final set of equations that model the pure aggregation.

5.1 Death term

Firstly we discuss the discretization of the loss term due to the aggregation of two particles. The second term in (4.3):

$$\int_{v_i}^{v_{i+1}} n(v,t)dv \int_0^{\infty} n(v',t)Q(v,v')dv'.$$

Substituting (4.4) into it yields:

$$\int_{v_i}^{v_i+1} \sum_{k=1}^{M} N_k(t)\delta(v-x_k)dv \int_0^{\infty} \sum_{k=1}^{M} N_k(t)\delta(v'-x_k)Q(v,v')dv'.$$

The inner integrand is only non-zero exactly at every cell center $v' = x_k$. For the outer integrand we even have a more strict situation. With $v = x_i$ being the only possibility for volume to be concentrated in the cell $[v_i, v_{i+1}]$ the only summand in the outer integrand that is non-zero is $N_i(t)$. With this, we can reduce the integrals to this discrete version of the death term due to breakage:

$$R_{Da} := N_i(t) \sum_{k=1}^{M} N_k(t)Q(x_i,x_k). \tag{5.1}$$

5.2 Birth term

Now we are interested in the birth term of aggregation. We discuss the birth of a particle of volume v from aggregation of two particles of volume v' and $v - v'$. The

first term in (4.3):

$$\frac{1}{2}\int_{v_i}^{v_{i+1}} dv \int_0^v n(v-v',t)n(v',t)Q(v-v',v')dv'.$$

Things to notice are the necessity for the $\frac{1}{2}$ to prevent double counting, since for every pair $(v', v-v')$ the for our purposes equal pair $(v-v', v')$ is also calculated. Also, in contrast to the loss term, the inner integral does not need to be evaluated up to ∞, since v' and $v-v'$ that aggregate to v can only lie between zero and v. The first step to discretize this term is to use our idea of splitting the volumes to the center points nearby introduced in (4.1). For this, we replace the first integral by two integrals. With this we define:

$$\begin{aligned} R_{Ba} := &\frac{1}{2}\int_{x_i}^{x_{i+1}} a(v,x_i)dv \int_0^v n(v-v',t)n(v',t)Q(v-v',v')dv' \\ &+ \frac{1}{2}\int_{x_{i-1}}^{x_i} b(v,x_i)dv \int_0^v n(v-v',t)n(v',t)Q(v-v',v')dv'. \end{aligned}$$

Now we introduce a new function $\eta_i(j,k)$ with the following properties:

$$\eta_i(j,k) = \begin{cases} a(x_j+x_k,x_i), & v \in [x_i, x_{i+1}], \\ b(x_j+x_k,x_i), & v \in [x_{i-1}, x_i], \\ 0, & \text{else.} \end{cases} \tag{5.2}$$

With this I can substitute the $a(v,x_i)$ and $b(v,x_i)$ with $\eta_i(j,k)$ and again with substitution of (4.4) we obtain:

$$\begin{aligned} R_{Ba} = &\frac{1}{2}\int_{x_i}^{x_{i+1}} \eta_i(j,k)dv \int_0^v Q(v-v',v') \sum_{k=1}^{M} N_k(t)\delta((v-v')-x_k) \sum_{j=1}^{M} N_j(t)\delta(v'-x_j)dv' \\ &+ \frac{1}{2}\int_{x_{i-1}}^{x_i} \eta_i(j,k)dv \int_0^v Q(v-v',v') \sum_{k=1}^{M} N_k(t)\delta((v-v')-x_k) \sum_{j=1}^{M} N_j(t)\delta(v'-x_j)dv'. \end{aligned}$$

Adding the outer integrals and combining the sums in the inner integral yields:

$$R_{Ba} = \frac{1}{2}\int_{x_{i-1}}^{x_{i+1}} \eta_i(j,k)dv \int_0^v Q(v-v',v') \sum_{k,j\in[1,M]} N_k(t)N_j(t)\delta((v-v')-x_k)\delta(v'-x_j)dv'.$$

To simplify the expression for R_{Ba}, we begin by considering the specific properties of our volumes of interest. The outer integral considers values v in the interval $[x_{i-1}, x_{i+1}]$ and calculates the expected contribution due to aggregation for each of these values. In this context, in the inner integrand, we are only interested in volumes $v' = x_j$ for some index j, and simultaneously $(v-v') = x_k$ for a corresponding index k. Every other combination yields 0 in the inner integrand. Additionally, we can eliminate the outer integral by only taking into account the pairs of interest that match the interval around our pivot. We do this under the condition that the sum of these volumes $x_k + x_j$ must be within the interval $[x_{i-1}, x_{i+1}]$. This condition ensures that we only consider combinations of volumes that contribute to the volume

v withing its range of influence. Consequently, we can express the R_{Ba} without the need of integration as:

$$R_{Ba} = \frac{1}{2} \sum_{k,j \in [1,M]} \eta_i(j,k) Q(x_k, x_j) N_k(t) N_j(t), \; x_{i-1} \leq (x_k + x_j) \leq x_{i+1}.$$

This equation captures the rate of birth of new particles due to aggregation by summing over all feasible pairs of indices k and j, weighted by the aggregation kernel $Q(x_k, x_j)$, their number distributions $N_k(t)$ and $N_j(t)$ and their contribution to the pivot elements calculated by $\eta_i(j,k)$. The factor $\frac{1}{2}$ still accounts for the symmetric nature of v' and $v - v'$.
However, we can eliminate the factor $\frac{1}{2}$ by limiting the indices of our sum to $k \geq j$, thus eliminating the symmetric cases. To address the cases where $k = j$, which would still be double counted, we introduce the term $1 - \frac{1}{2}\delta_{k,j}$ into the sum. Here, $\delta_{k,j}$ is the Kronecker delta, which equals 1 when $k = j$ and is 0 otherwise. This ensures that the cases which we are double counting are contributing half. Thus the final expression for our R_{Ba} becomes:

$$R_{Ba} = \sum_{k,j \in [1,M]}^{j \geq k} \left(1 - \frac{1}{2}\delta_{k,i}\right) \eta_i(j,k) Q(x_k, x_j) N_k(t) N_j(t), \; x_{i-1} \leq (x_k + x_j) \leq x_{i+1}. \tag{5.3}$$

5.3 Pure aggregation

We achieve the term for the rate of pure aggregation by combining (5.1) and (5.3):

$$\begin{aligned} R_A &:= R_{Ba} - R_{Da} \\ &= \sum_{\substack{k,j \in [1,M] \\ x_{i-1} \leq (x_k + x_j) \leq x_{i+1}}}^{j \geq k} \left(1 - \frac{1}{2}\delta_{k,i}\right) \eta_i(j,k) Q(x_k, x_j) N_k(t) N_j(t) \\ &\quad - N_i(t) \sum_{k=1}^{M} N_k(t) Q(x_i, x_k). \end{aligned} \tag{5.4}$$

5.4 Aggregation kernel

The aggregation kernel, denoted as $Q(v - v', v')$, describes the rate at which a particle of volume v forms from the aggregation of two smaller particles, one with volume $v - v'$ and the other with volume v'. Similarly to breakage, aggregation plays a crucial role in particle systems and directly influences the evolution of the particle size distribution. The aggregation kernel captures how likely two particles of different sizes are to combine and form a new particle, thus controlling the frequency and outcome of aggregation events.
In general, kernel functions such as $Q(v - v', v')$ are mathematical representations that describe interaction rates in a variety of processes, such as particle aggregation.

These kernels can take on various forms, depending on the physical assumptions of the system, with each form corresponding to different types of interaction mechanism between particles. Depending on the assumed system behavior, we can model $Q(v - v', v')$ using different forms of kernel functions. In the following, we explore three common choices for the aggregation kernel and discuss their physical implications.
Firstly, the constant kernel:

$$Q(v - v', v') = c,$$

for some constant $c \in \mathbb{R}$. This kernel assumes that the rate of aggregation is independent of particle size, which means that all particles, regardless of their volume, are equally likely to aggregate. Physically, this implies that the probability of aggregation depends solely on the concentration of particles, with no preference for the size of the interacting particles.
Secondly, the sum kernel:

$$Q(v - v', v') = (v - v') + v' = v.$$

This kernel assumes that the aggregation rate depends linearly on the size of the particles involved. Larger particles are more likely to aggregate compared to smaller particles, as the interaction rate is proportional to the combined volume of the two particles.
Lastly, the multiplicative kernel:

$$Q(v - v', v') = (v - v')v'.$$

This kernel models the situation where the aggregation rate depends on the product of the volumes of the two particles. This implies that larger particles are exponentially more likely to aggregate, leading to rapid aggregation of the largest particles in the system.

Chapter 6

Breakage

Breakage is a fundamental phenomenon encountered in a wide range of scientific and engineering fields. It describes the process by which a larger particle splits into smaller fragments due to various external forces such as mechanical stress, chemical reactions, or thermal effects. To study the breakage in our model, we need to observe again two different terms. The death term represents the rate at which particles of a given size are lost from the system due to breakage. Larger particles break into smaller ones, and as a result, the number of particles in the original size class decreases over time. This is the "death" of particles in a specific size class. For the birth term we are interested in the rate that new particles enter our system. When a particle breaks, it produces smaller fragments that fall into different size classes. This process is captured by the birth term in the population balance equation, which accounts for the rate at which new particles are created in specific size classes as a result of breakage. As with the aggregation, the interaction between these birth and death terms defines the evolution of the number density in the system.

6.1 Death term

We again start with the discretization of the death term. The death term accounts for the rate at which particles of size v are lost as they break into smaller fragments, depending on their frequency they may break and depending on how many particles are able to break. In a discrete interval between two points in the grid v_i and v_{i+1} the rate of death is given by the fourth term in in (4.3):

$$\int_{v_i}^{v_{i+1}} \Gamma(v) n(v,t) dv.$$

Here $\Gamma(v)$ is the frequency a particle of volume v breaks. Choices for $\Gamma(v)$ will be discussed later, especially for binary breakage. To acquire our equations we again substitute (4.4) into the integral to obtain:

$$\int_{v_i}^{v_{i+1}} \Gamma(v) \sum_{k=1}^{M} N_k(t) \delta(v - x_k) dv.$$

Thanks to the Dirac delta distribution, the only value of k where the summand is non-zero is $k = i$. And again with the Dirac delta distribution, the only v in

the given interval where the integrals does not vanish is at our pivot x_i. And this is indeed the only valid pivot because it is the only pivot in the interval $[v_i, v_{i+1}]$ Therefore the expression simplifies to:

$$\begin{aligned} R_{Db} &:= \int_{v_i}^{v_{i+1}} \Gamma(v) \sum_{k=1}^{M} N_k(t)\delta(v - x_k)dv \\ &= \int_{v_i}^{v_{i+1}} \Gamma(v) N_i(t)\delta(v - x_i)dv \\ &= \Gamma(x_i)N_i(t). \end{aligned} \tag{6.1}$$

And thus we arrived at a equation that only depends on our breakage frequency at our pivot and the number of particles at our pivot. Note that the death term does not depend on how many new particles are born due to the death. Unfortunately, our final equation for the birth term will not be as simple, due to the allocation of the new born particles in our system.

6.2 Birth term

To study the birth term, we need a way to account for the fact that a particle of volume v' can split into an arbitrary amount of daughter particles each with a volume smaller than v'. To conserve the volume in our system we also need to make sure that the volumes of the daughter particles sum up to the volume v'. The rate of birth in a discrete interval between two grid points v_i and v_{i+1} is given by the third term in (4.3):

$$\int_{v_i}^{v_{i+1}} \int_{v}^{\infty} \beta(v, v')\Gamma(v')n(v', t)dv'dv.$$

Here, $\beta(v, v')dv$ describes the number of daughter particles in the range from v to $v + dv$ that are born due to breakage of a particle of volume v'. The outer integral ranges from v_i to v_{i+1}, because we are interested in the rate that new particles are born in this interval. The inner integral ranges from v to ∞, because for each interval $[v, v + dv]$ only particles of volume v' with $v' \geq v$ could break into volumes $[v, v + dv]$. Because we only want to the study the values at our pivots, we need again to distribute any new particles partial to the nearest pivots. Therefore, we split the outer integral in two parts, one for the distribution of volumes between x_i and x_{i+1} and one for the distribution of volumes between x_{i-1} and x_i. We here make use of our functions $a(v, x_i)$ for the first case and $b(v, x_i)$ for the second one. We define the new expression as:

$$\begin{aligned} R_{Bb} := &\int_{x_i}^{x_{i+1}} a(v, x_i) \int_{v}^{\infty} \beta(v, v')\Gamma(v')n(v', t)dv'dv \\ &+ \int_{x_{i-1}}^{x_i} b(v, x_i) \int_{v}^{\infty} \beta(v, v')\Gamma(v')n(v', t)dv'dv. \end{aligned}$$

As always, to reduce the equation to just take values at our pivot points, we substitute (4.4):

$$R_{Bb} = \int_{x_i}^{x_{i+1}} a(v, x_i) \int_v^\infty \beta(v, v')\Gamma(v') \sum_{k=1}^{M} N_k(t)\delta(v - x_k)dv'dv$$
$$+ \int_{x_{i-1}}^{x_i} b(v, x_i) \int_v^\infty \beta(v, v')\Gamma(v') \sum_{k=1}^{M} N_k(t)\delta(v - x_k)dv'dv.$$

Our strategy here involves the following steps. Firstly, we pull the $\beta(v, v')$ and the $\Gamma(v')$ into the sum. Then we notice that the sum and therefore the inner integrand vanishes for each $v' \neq x_k$ for any k. Also, since we are only taking v' from $[x_{i-1}, \infty)$, it is sufficient to only consider $k \geq i$, because for $k < i$ we always have $v' > x_k$. With that the inner integral sign disappears and we factor everything that does not depend on v out of the outer integral. With these steps we derive our final equation for the birth term due to breakage:

$$\begin{aligned}
R_{Bb} &= \int_{x_i}^{x_{i+1}} a(v, x_i) \int_v^\infty \beta(v, v')\Gamma(v') \sum_{k=1}^{M} N_k(t)\delta(v - x_k)dv'dv \\
&+ \int_{x_{i-1}}^{x_i} b(v, x_i) \int_v^\infty \beta(v, v')\Gamma(v') \sum_{k=1}^{M} N_k(t)\delta(v - x_k)dv'dv \\
&= \int_{x_i}^{x_{i+1}} a(v, x_i) \int_v^\infty \sum_{k=1}^{M} \beta(v, v')\Gamma(v')N_k(t)\delta(v - x_k)dv'dv \\
&+ \int_{x_{i-1}}^{x_i} b(v, x_i) \int_v^\infty \sum_{k=1}^{M} \beta(v, v')\Gamma(v')N_k(t)\delta(v - x_k)dv'dv \\
&= \int_{x_i}^{x_{i+1}} a(v, x_i) \sum_{k\geq i}^{M} \beta(v, x_k)\Gamma(x_k)N_k(t)dv \\
&+ \int_{x_{i-1}}^{x_i} b(v, x_i) \sum_{k\geq i}^{M} \beta(v, x_k)\Gamma(x_k)N_k(t)dv \\
&= \sum_{k\geq i}^{M} \Gamma(x_k)N_k(t) \int_{x_i}^{x_{i+1}} a(v, x_i)\beta(v, x_k)dv \\
&+ \sum_{k\geq i}^{M} \Gamma(x_k)N_k(t) \int_{x_{i-1}}^{x_i} b(v, x_i)\beta(v, x_k)dv.
\end{aligned}$$

To combine the two sums we introduce the term $n_{i,k}$, which is defined as following:

$$\begin{aligned} n_{i,k} &= \int_{x_i}^{x_{i+1}} a(v, x_i)\beta(v, x_k)dv \\ &+ \int_{x_{i-1}}^{x_i} b(v, x_i)\beta(v, x_k)dv \\ &= \int_{x_i}^{x_{i+1}} \frac{x_{i+1} - v}{x_{i+1} - x_i}\beta(v, x_k)dv \\ &+ \int_{x_{i-1}}^{x_i} \frac{v - x_{i-1}}{x_i - x_{i-1}}\beta(v, x_k)dv. \end{aligned} \tag{6.2}$$

We can interpret $n_{i,k}$ as the quantification of how much of the broken volume from larger particles x_k ends up in the population of smaller particles x_i. For example, if we have $n_{i,k} = \frac{1}{2}$ for some i and k, that means that half of the volume from the breakage of x_k would end up as volume at x_i. Unlike to the aggregation term, it is not helpful to use the introduced $\eta_i(j, k)$ term, because the integrals would need to be handled independently anyways. With this simplification we get:

$$R_{Bb} = \sum_{k=i}^{M} n_{i,k}\Gamma(x_k)N_k(t). \tag{6.3}$$

Again, unlike all the other terms we discussed until now, this final expression did not remove all the integral signs. Therefore, it is in general required to solve them with an appropriate numerical method such as the trapezoidal rule.

6.3 Pure breakage

We achieve the term for the rate of pure breakage by combining (6.1) and (6.3):

$$\begin{aligned} R_B &= R_{Bb} - R_{Db} \\ &= \sum_{k=i}^{M} n_{i,k}\Gamma(x_k)N_k(t) \\ &- \Gamma(x_i)N_i(t). \end{aligned} \tag{6.4}$$

6.4 Breakage frequency

The breakage frequency, denoted as $\Gamma(v')$, describes how often a particle of volume v' undergoes breakage. To accurately mode a population balance equation, it is important to make an appropriate choice for $\Gamma(v')$. Depending on the assumptions of our system, different choices seem logical. We will briefly cover some choices and compare them in our numerical studies. The first option would be a constant breakage frequency, i.e., for some constant $c \in \mathbb{R}$:

$$\Gamma(v') = c.$$

This represents a breakage frequency that does not depend on the volume of the given particle. That is, every particle independently of their volume has the same

probability of breaking into new smaller particles.
Another option to consider is a linear dependently breakage frequency, i.e.:

$$\Gamma(v') = v'.$$

Here the frequency depends linearly on the volume of the given particle, meaning that a larger particles tends to break more often than a smaller one.
And finally, we consider a quadratic breakage frequency, i.e.:

$$\Gamma(v') = (v')^2.$$

With this choice, the breakage frequency depends even stronger on the volume of the given particle. Meaning that larger particles are even more likely to break than smaller ones.

6.5 Binary breakage

The function $\beta(v, v')$ represents the breakage distribution and describes how a parent particle v' breaks into smaller daughter particles. Specifically, $\beta(v, v')dv$ gives the number of daughter particles with volumes in the range $[v, v + dv]$ produced when a larger particle of volume v' undergoes the process of breaking. The choices of this function yields many important physical properties. For our purposes, it gives the number of daughter particles formed from each breakage and ensures the conservation of volume. We will from now on only consider binary breakage events. That means that any breakage of a parent particle can only result in the birth of exactly two new smaller daughter particles. Firstly, we discuss the condition that $\beta(v, v')$ must satisfy to model binary breakage. We need to ensure that whenever a particle of volume v' undergoes breakage, exactly two new particles are formed, or in our discretization the volume of two particles are distributed between 0 and v'. For this we introduce a normalization condition:

$$\int_0^{v'} \beta(v, v')dv = 2.$$

The integral captures the total number of daughter particles generated from the breakage process by adding all possible volumes of daughter particles v. So if we were interested in the breakage into three particles for example, the integral would need to be equal to three, The next condition will ensure that we conserve the volume of the parent particle v'. For binary breakage where v' breaks into v_1 and v_2 we can state this with a simple equation like:

$$v' = v_1 + v_2.$$

But we can also generalize this for all kinds of breakage processes. For this, we want that the sum over all daughter particle volumes is equal to the volume of the parent. To represent the volume of the daughter particles of volume v, weighted by our distribution function $\beta(v, v')$, we use the expression $v\beta(v, v')$. Summing this expression over all possible daughter particles yields a volume conservation condition:

$$\int_0^{v'} v\beta(v, v')dv = v'.$$

To further refine our model of binary breakage, we now introduce the concept of uniform binary breakage. This assumption implies that when a parent particle breaks into two daughter particles, the probability of the size of the daughter particles is uniformly distributed over the possible volume range. In other words, each potential daughter particle size is equally likely to be formed, provided that the sum of their volumes equals that of the parent particle. To mathematically express this uniformity, we impose the condition that the breakage distribution function $\beta(v, v')$ is constant with respect to the volume of the daughter particles v. This leads to the conclusion that the breakage distribution is independent of the size of the daughter particles, meaning that the distribution is spread uniformly across all possible breakage sizes.
This uniform behavior can be formulated as:

$$\beta(v, v') = \frac{2}{v'}.$$

To verify that this choice for $\beta(v, v')$ satisfies the two conditions given earlier, we substitute it into the normalization and volume conservation condition:

$$\int_0^{v'} \frac{2}{v'}\, dv = \frac{2}{v'} \cdot v' = 2,$$

$$\int_0^{v'} v \cdot \frac{2}{v'}\, dv = \frac{2}{v'} \int_0^{v'} v\, dv = \frac{2}{v'} \cdot \frac{(v')^2}{2} = v'.$$

Thus, the form $\beta(v, v') = \frac{2}{v'}$ is the correct choice for modeling uniform binary breakage. It not only satisfies the conditions of particle number and volume conservation, but also ensures that the breakage process is uniformly distributed over all possible daughter particle sizes.
With this choice for $\beta(v, v')$, we can solve $n_{i,k}$ analytically. Recalling the discrete expression of $n_{i,k}$:

$$n_{i,k} = \int_{x_i}^{x_{i+1}} a(v, x_i)\beta(v, x_k)dv + \int_{x_{i-1}}^{x_i} b(v, x_i)\beta(v, x_k)dv.$$

Substituting $\beta(v, x_k) = \frac{2}{x_k}$ and the known forms of $a(v, x_i)$ and $b(v, x_i)$:

$$\begin{aligned} n_{i,k} &= \frac{2}{x_k}\left(\int_{x_i}^{x_{i+1}} a(v, x_i)dv + \int_{x_{i-1}}^{x_i} b(v, x_i)dv\right) \\ &= \frac{2}{x_k}\left(\int_{x_i}^{x_{i+1}} \frac{x_{i+1} - v}{x_{i+1} - x_i}dv + \int_{x_{i-1}}^{x_i} \frac{v - x_i}{x_i - x_{i-1}}dv\right). \end{aligned}$$

Now we solve these integrals. Starting with the first:

$$\begin{aligned} \int_{x_i}^{x_{i+1}} \frac{x_{i+1} - v}{x_{i+1} - x_i}dv &= \frac{1}{x_{i+1} - x_i}\left(\frac{x_{i+1}^2}{2} - x_{i+1}x_i + \frac{x_i^2}{2}\right) \\ &= \frac{(x_{i+1} - x_i)^2}{2(x_{i+1} - x_i)} \\ &= \frac{x_{i+1} - x_i}{2}. \end{aligned}$$

Next, for the second integral:

$$\begin{aligned}\int_{x_{i-1}}^{x_i} \frac{v - x_{i-1}}{x_i - x_{i-1}} dv &= \frac{1}{x_i - x_{i-1}} \left(\frac{x_i^2}{2} - x_{i-1}x_i + \frac{x_{i-1}^2}{2} \right) \\ &= \frac{(x_i - x_{i-1})^2}{2(x_i - x_{i-1})} \\ &= \frac{x_i - x_{i-1}}{2}.\end{aligned}$$

Together, we have:

$$\begin{aligned}n_{i,k} &= \frac{2}{x_k}(\frac{x_{i+1} - x_i}{2} + \frac{x_i - x_{i-1}}{2}) \\ &= \frac{2}{x_k} \frac{x_{i+1} - x_{i-1}}{2} \\ &= \frac{x_{i+1} - x_{i-1}}{x_k}.\end{aligned}$$

Thus, we have derived the analytical solution for $n_{i,k}$, bypassing the need for numerical methods such as the trapezoidal rule.
The analytical solution for $n_{i,k}$ is quite intuitive when we think about it in the context of uniform binary breakage.
Since we are assuming uniform binary breakage, the distribution of daughter particles is uniform across all possible breakage sizes, meaning that the size of the intervals where the smaller particles fall becomes a key factor. The numerator, $x_{i+1} - x_{i-1}$, represents the width of the interval that surrounds the particle sizes around x_i, essentially capturing how much of the volume from a broken parent particle is contributed to the population within that range.
At the same time, the denominator, x_k, represents the size of the parent particle that undergoes breakage. Since the breakage is uniform, the ratio between the size of the interval of interest and the size of the broken particle gives us a straightforward quantification of how much of the broken volume is likely to end up in the interval around x_i. Thus, this expression captures the essence of binary breakage: the size of the interval and the size of the broken particle are the primary factors that determine how the volume is redistributed among the smaller particles.

Chapter 7

Numerical methods

In this work, two key numerical methods were employed to solve the population balance equation, specifically to compute the breakage term and solve the resulting system of IODEs. For the breakage term, which involves calculating integrals over the particle volume distribution, the trapezoidal rule was used. The trapezoidal rule is a simple and widely used method for numerical integration, where the area under a curve is approximated by dividing the curve into trapezoids rather than rectangles. This method provides a good balance between simplicity and accuracy, making it well suited for handling the integral expressions in the breakage term, particularly when using a discrete-volume grid as in the fixed pivot method.
The formula for a single interval $[a, b]$ is given by:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\left(f(a)+f(b)\right).$$

For n equally spaced points $x_0, x_1, \ldots, x_n$ with $h = \frac{b-a}{n}$, the extended form of the trapezoidal rule is:

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left(f(x_0)+2f(x_1)+2f(x_2)+\cdots+2f(x_{n-1})+f(x_n)\right).$$

To solve the system of IODEs arising from the population balance equation, we first approximate integrals using the trapezoidal rule before applying a Runge-Kutta method for time integration.
A Butcher tableau is a common way to represent the coefficients of a Runge-Kutta method, summarizing how each intermediate stage is computed and combined to advance the solution of ordinary differential equations. The general form of the Butcher tableau is given below.

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \\ & \hat{b}_1 & \hat{b}_2 & \cdots & \hat{b}_s \end{array}$$

In this tableau the coefficients a_{ij} determine how previous intermediate steps influence each stage, while c_i represents the time at which each stage is evaluated. The weights b_i and $\hat{b_i}$ produce the fifth- and fourth-order solutions, respectively, which are used for adaptive step size control.

The intermediate steps k_i are computed using:

$$k_i = f\left(t + c_i h, y + h\sum_{j=1}^{i-1} a_{ij}k_j\right),$$

where h is the time step, t is the current time, y is the current value of the solution, and f is the derivative function of the IODE.

Finally, the solutions are calculated by:

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i,$$

and

$$\hat{y}_{n+1} = y_n + h\sum_{i=1}^{s} \hat{b}_i k_i.$$

This general tableau form can be used to describe any Runge-Kutta method, including higher-order or adaptive schemes, such as the RK45 method from the SciPy library used in this work.

RK45 (Dormand-Prince method) is an adaptive step size method, which means that it adjusts the time step dynamically based on the local error estimate at each step, allowing for efficient and accurate integration over time. The method combines fourth-order and fifth-order Runge-Kutta methods, using the difference between the fourth- and fifth-order results to estimate the local error and adapt the step size dynamically. The underlying scheme can be represented by its Butcher tableau.

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

In the Butcher tableau, the first set of coefficients represents the fourth-order method, while the second set corresponds to the fifth-order method.
Together, these methods provide a comprehensive numerical framework: The trapezoidal rule ensures accurate computation of the integrals in the breakage term, while RK45 effectively handles the temporal evolution of the particle size distribution through the system of IODEs.

Chapter 8

Implementation

This chapter focuses on the implementation and testing of the methods discussed in the previous sections. The goal is to provide a detailed account of how the fixed pivot technique was applied to solve population balance equations and to document the iterative process of development.
The chapter is structured as follows. We begin with the implementation of pure aggregation, followed by pure breakage, and we finally address the simultaneous occurrence of breakage and aggregation.
It is important to note that my initial implementations were not always immediately correct. Throughout this chapter, we describe the steps taken during implementation, the challenges encountered, and the corrections applied to resolve the errors.
For all tests conducted with a correctly functioning code, we provide precise details regarding the grid, initial conditions, and kernels used. In cases where the code was still under development and contained errors, these details are not always explicitly stated, as they were not the primary focus during those stages of testing.

8.1 Pure aggregation

We will start with the implementation of the pure aggregation process. That means we want to solve this system of ODEs:

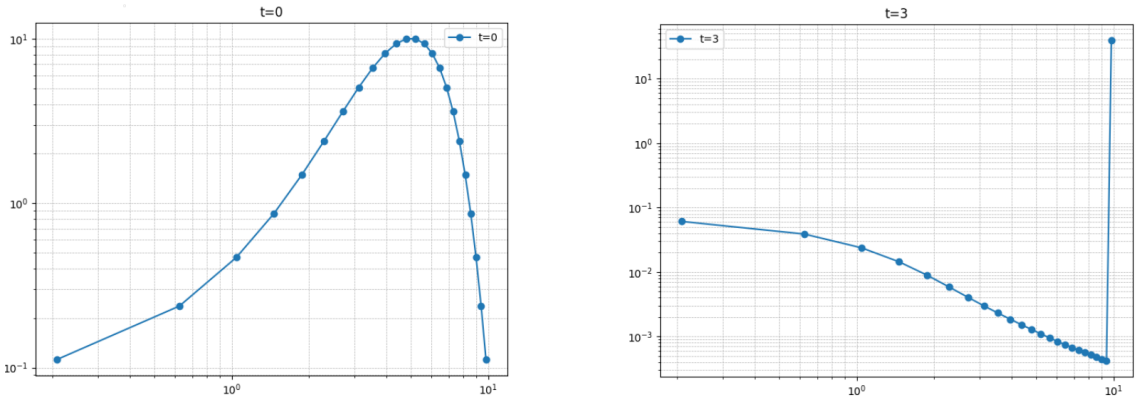
$$\begin{aligned}\frac{dN_i(t)}{dt} = &\sum_{\substack{k,j\in[1,M]\\ x_{i-1}\leq(x_k+x_j)\leq x_{i+1}}}^{j\geq k}\left(1-\frac{1}{2}\delta_{k,i}\right)\eta Q(x_k,x_j)N_k(t)N_j(t)\\ &- N_i(t)\sum_{k=1}^{M}N_k(t)Q(x_i,x_k).\end{aligned} \tag{8.1}$$

As mentioned in the previous chapter, this will be done with the RK45 method of the SciPy library, in particular with the solve_ivp() function from that library. This function requires a function of the right-hand side as an input. This can be done quite easily for the most part. The only interesting part of the right-hand side is the restriction of the first sum and the calculation of $\eta_i(j,k)$ given by (5.2). We can use the value of $\eta_i(j,k)$ to handle the interval restriction. Whenever we look at a pair of pivots that does not match the restriction we set $\eta_i(j,k)=0$. The next thing

to consider are the boundary pivots, that is, x_1 and x_M. The fraction of volume that is allocated to a pivot x_i due to the aggregation of x_j and x_k is described by $\eta_i(j,k)$. The boundary at $i = 1$ can be problematic for the $x_{i-1} \leq v \leq x_i$ part of $\eta_i(j,k)$ since x_0 is not defined. But we can just set this $\eta_i(j,k) = 0$ because the smallest aggregation possible in our system is $x_1 + x_1$, which cannot lie in the interval of interest. Now we consider the other boundary pivot x_M. For this pivot, we cannot calculate $\eta_M(j,k)$ with $x_j + x_k \geq x_M$, since x_{M+1} is not defined. Here we started by setting $\eta_M(j,k) = 1$, which in hindsight was the wrong choice. Now, to verify that the code is working correctly, we checked the total volume in the system before and after the calculations using the first moment of the number density. We discretize it using the discrete version of the number density:

$$\begin{aligned} M_1 &= \int_0^\infty vn(v,t)dv \\ &= \int_0^\infty v \sum_{k=1}^{M} N_k(t)\delta(v - x_k)dv \\ &= \sum_{k=1}^{M} x_k N_k(t). \end{aligned}$$

Testing my code with different types of grid and different aggregation kernels $Q(x_k, x_j)$ always resulted in a significant loss in volume. To understand what went wrong, we did a test on a geometric grid, with a multiplicative aggregation kernel and a normal distributed initial condition. Figure 8.1 shows the initial state of my system on the left and the solution on the right. The x-axis is the volume at the pivots, and the y-axis is $N_i(t)$.



(a) Initial distribution

(b) End distribution

Figure 8.1: Pure aggregation with product kernel

The first thing we noticed is the pivot x_M which is an extreme outlier compared to all other pivots. But this makes a lot of sense with my current code, considering that every aggregation process such that $x_k + x_j \geq x_M$ contributes only to x_M. And, especially, the more particles are assigned to x_M, the more likely it is that the particles from x_M aggregate with other particles to further increase the number of particles in x_M. That means that some kind of exponential tail is to be expected.

To fix this problem, we have reorganized the volume allocation to x_M. My solution is quite simple. If we have $x_k + x_j \geq x_M$, we still want this new volume assigned to the boundary x_M. But not with $\eta_M(j,k) = 1$. Instead we want to assign the ratio of the new particle volume and x_M, i.e:

$$\eta_M(j,k) = \frac{x_j + x_k}{x_M},\ x_j + x_k \geq x_M.$$

With these changes the volume is conserved for all my simulations with different kinds of grids, step sizes, aggregation kernel, and initial conditions.
The next step of verification is to observe whether the behavior of the system mimics the behavior found in nature. We would assume that since no breakage events can occur in this simulation, the number of particles in x_M would only increase over time. We would also assume that the number of particles between x_1 and x_{M-1} would decrease, since larger particles are more likely to aggregate and therefore vanish from the system. However, we can test this behavior a little more precisely. We changed the initial condition to be the sum of two normal distributions that are centered around different pivots. They are centered around μ and λ. We call the pivots, the distributions are centered around x_μ and x_λ, respectively, and we have $x_\mu < x_\lambda$. With that, the most particles are in the regions around x_μ and x_λ, and therefore these are the particles that are the most likely to aggregate with each other. Hence, at some point in the simulation, it can be expected that a lot of particles are assigned at x_i with $|x_i - (x_\alpha + x_\beta)|$ being minimal with $\alpha \in \{\mu, \lambda\}$ and $\beta \in \{\mu, \lambda\}$ where the cases where α and β are distinct are identical. I will use the notation $x_i = x_{\mu+\lambda}$ for this pivot. To test this, I saved after each time step the three pivots with the most particles assigned to them and plotted these results in Figure 8.2. The x-axis is for the time steps, and the y-axis is the i values of the pivots. These pivots shown in the plot will from now on be named maximal pivots without differentiating whether they have the most, second most, or third most particles assigned to them.

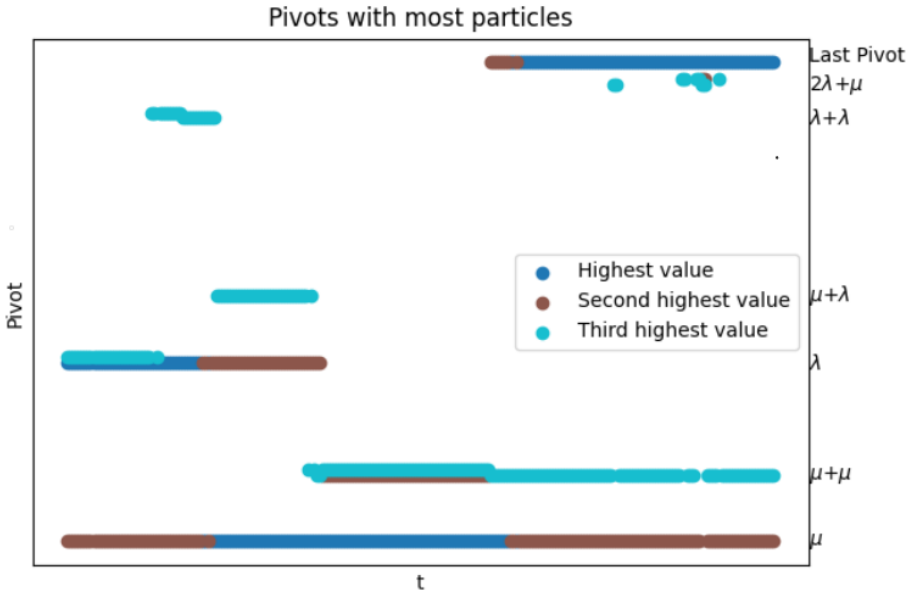


Figure 8.2: Evolution of the maximal pivots over time

The interpretation of this plot matches our expectation. At the start of the simulation the pivots with the most particles are x_μ, x_λ, and a pivot near x_λ. The first new maximal pivot during the run-time is $x_{\lambda+\lambda}$. This makes sense since x_λ and x_μ have the most particles assigned to them, but $x_\mu < x_\lambda$ and therefore the particles in x_λ are more likely to aggregate with particles of the same volume than the particles in x_μ. The next maximal pivot is the expected $x_{\mu+\lambda}$. Followed by $x_{\mu+\mu}$, which follows the same logic as $x_{\lambda+\lambda}$. The maximal pivot $x_{2\lambda+\mu}$ can also be explained in the same way. We can also see the behavior that after some time, many particles aggregate to x_M and then stay there. Another observation is that x_λ stops being a maximal pivot during the run-time while x_μ stays a maximal pivot throughout the whole simulation. This can again be explained by the fact that $x_\mu < x_\lambda$ and therefore the particles at x_λ are more likely to aggregate and disappear from the system.

Although the approach to handling $\eta_M(j,k)$ ensures volume conservation and demonstrates some logical behavior, I am not entirely satisfied with it. Specifically, the allocation of all newly formed particles exclusively to the pivot x_M whenever a particle aggregates with one of volume x_M is problematic. This behavior does not align with the principles of the fixed pivot method, which is based on partial volume allocation. The assignment of all aggregated particles to a single pivot oversimplifies the system.

To address this issue, I adopted a new approach: I designed the initial condition and the grid so that the particles do not reach the boundary x_M. This was achieved by modifying both the initial condition and the grid design. I increased the number of pivots to provide a finer resolution near the upper boundary and set the initial condition to zero for pivots close to x_M. Furthermore, I implemented a stop event in the solve_ivp() function, which stops the simulation as soon as particles with $v > x_M$ are formed.

In the following, I will conduct two tests to evaluate this new strategy, focusing on the logic and consistency of the resulting particle distributions.

Algorithm 1 System Aggregation Computation

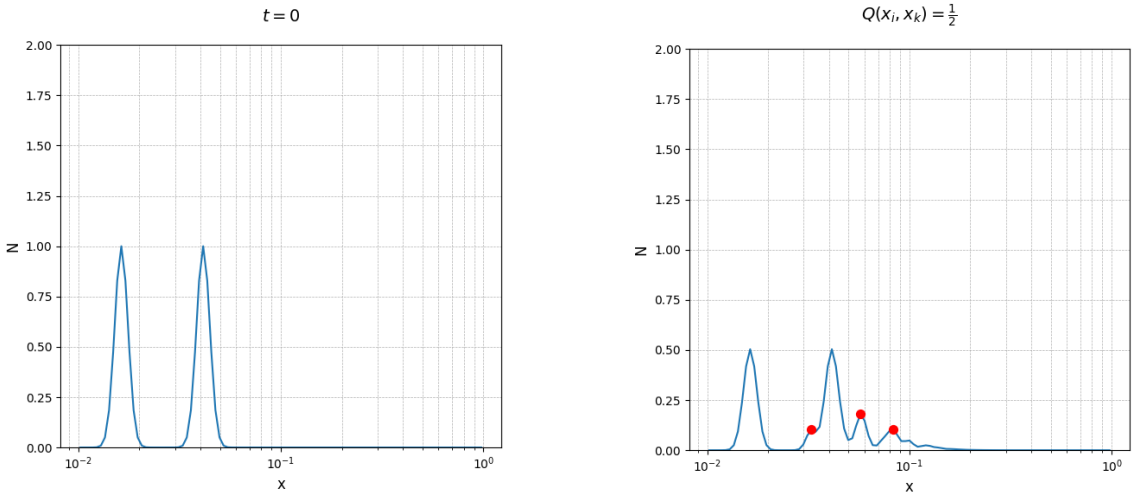
Input: t, N, x
Output: dN
$M \leftarrow$ length of N
$dN \leftarrow$ array of zeros with length M
for $i \leftarrow 0$ to $M-1$ **do**
    $agg\_loss \leftarrow 0$
    $agg\_birth \leftarrow 0$
    for $k \leftarrow 0$ to $M-1$ **do**
        for $j \leftarrow k$ to $M-1$ **do**
            $\eta \leftarrow 0$
            $\delta \leftarrow 1$ if $j = k$ else 0
            $v \leftarrow x[k] + x[j]$
            if $i \neq (M-1)$ **and** $x[i] \leq v \leq x[i+1]$ **then**
                $\eta \leftarrow (x[i+1] - v)/(x[i+1] - x[i])$
            else if $i = 0$ **then**
                $\eta \leftarrow 0$
            else if $i = (M-1)$ **and** $v > x[i]$ **then**
                $\eta \leftarrow 0$
            else if $x[i-1] \leq v \leq x[i]$ **then**
                $\eta \leftarrow (v - x[i-1])/(x[i] - x[i-1])$
            end if
            $agg\_birth \leftarrow agg\_birth + (1 - \frac{1}{2} \times \delta) \times \eta \times Q(x[k], x[j]) \times N[j] \times N[k]$
        end for
        if $k \neq (M-1)$ **then**
            $agg\_loss \leftarrow agg\_loss + Q(x[i], x[k]) \times N[k]$
        end if
    end for
    if $i \neq (M-1)$ **then**
        $dN[i] \leftarrow -N[i] \times agg\_loss$
    end if
    $dN[i] \leftarrow dN[i] + agg\_birth$
end for
return dN

For the first example, we perform a test where the initial condition is a sum of two normal distributions with low variance, each centered around different pivots. My expectation is that, similar to the previous test, during the simulation, a significant number of particles will aggregate at pivots corresponding to the three cases of sums of the pivots around which the normal distributions are centered.
In this test, the grid consists of 100 pivots on a geometric scale ranging from 0.01 to 1. The aggregation kernel $Q(x_i, x_k) = \frac{1}{2}$ is constant, simplifying the aggregation rate for all pairs of particles. The initial condition consists of two normal distributions

with $\mu = 1$ and $\sigma = 0.1$. The first distribution is centered around the 10th pivot, while the second is centered around the 30th pivot.
For all tests in this thesis, the maximum step size for the numerical solver is set to 0.001. However, for this particular test, I chose a small end time of $t = 0.2$, to focus on the behavior of the system shortly after aggregation began. This allows us to analyze how the system evolves in its early stages and to verify that particle aggregation behaves as expected.
Due to the aggregation process, the particles initially centered at the 10th and 30th pivots are expected to combine and form larger particles. Based on the aggregation behavior under the peaks of the normal distributions, the following volumes are significant.

- Aggregation of two particles at x_{10} produces a volume closest to x_{25},
- Aggregation of a particle at x_{10} with one at x_{30} produces a volume closest to x_{37},
- Aggregation of two particles at x_{30} produces a volume closest to x_{45}.

These pivots are colored red.



(a) Initial distribution (b) End distribution

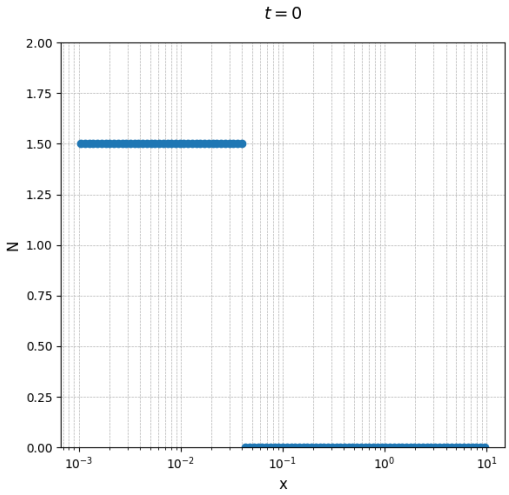
Figure 8.3: Pure aggregation with constant kernel

From the plots in Figure 8.3, we observe that, as expected in aggregation-dominated systems, the average particle volume increases over time, while the total number of particles in the system decreases. In addition, distinct peaks appear at the red-marked pivots. This indicates that a comparatively large number of particles have aggregated and accumulated at these specific pivots. These peaks confirm that particles tend to aggregate around the volumes corresponding to the sums of the initial peaks, as predicted.

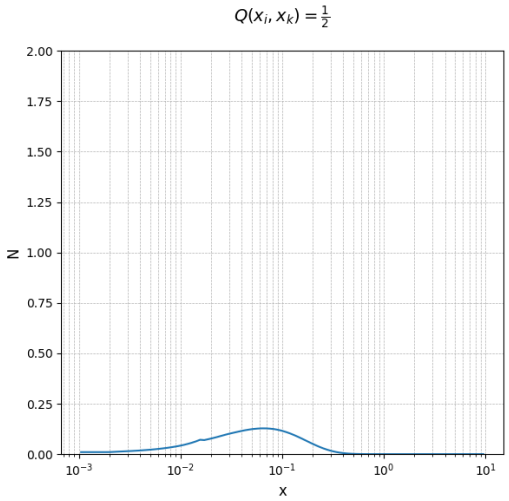
In the second test, the focus is to observe the influence of the aggregation kernel on the dynamics of the system. Similarly to the first test, the geometric grid

spans 0.01 to 1 with 100 pivots. However, in this case, the end time is extended to $t = 1$ to observe the longer-term behavior of the system.
The initial condition for this test is simple: the values of $N_i(0)$ for the first 40 pivots are set to 1.5, while all remaining pivots are initialized to 0. This ensures that the majority of particles start concentrated at smaller volumes, and we do not reach our right boundary.
To analyze the effect of the aggregation kernel, the test is performed twice: first, using a constant kernel $Q(x_i, x_k) = \frac{1}{2}$. Second, using a product kernel $Q(x_i, x_k) = 1000 x_i x_k$.

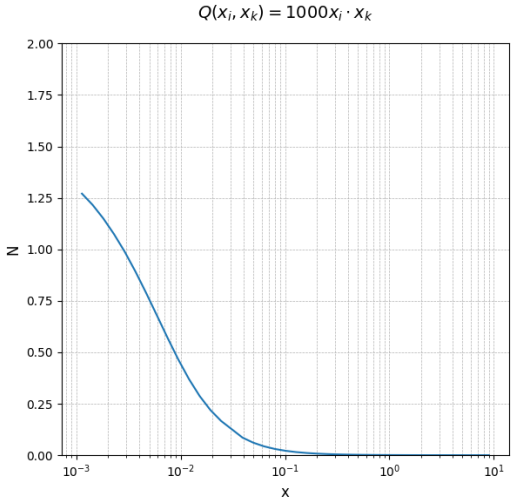
The expectation is that the behavior of the system will differ significantly between the two kernels. For the constant kernel, aggregation occurs uniformly across all pair of particles, regardless of their size. In contrast, the product-based kernel amplifies the aggregation rate for larger particles due to its dependence on the product of their volumes. As a result, it is anticipated that with the product kernel, more particles with smaller volumes will remain in the system compared to the constant kernel. The results can be seen in Figure 8.4.



(a) Initial distribution



(b) Final distribution with the constant kernel at $t = 1$



(c) Final distribution with the product kernel at $t = 1$

Figure 8.4: Pure aggregation with different kernels

For the product kernel, the aggregation rate increases with particle size, since larger particles have a higher probability of aggregating. This results in the following observations: the number of particles decreases from left to right in the size distribution. This is because larger particles aggregate more frequently, while smaller particles are less reactive and persist longer in the system.
Since the system allows only pure aggregation (no breakage), particles can only be removed by merging into larger ones. Consequently, the distribution contains still many particles of smaller size.

For the constant kernel, the aggregation probability is independent of particle size. Every particle pair has the same likelihood of merging, leading to a significantly different distribution pattern. Unlike the product kernel, there is no clear monotonic decrease in particle number toward larger sizes. Instead, the distribution appears more uniform.

8.2 Pure breakage

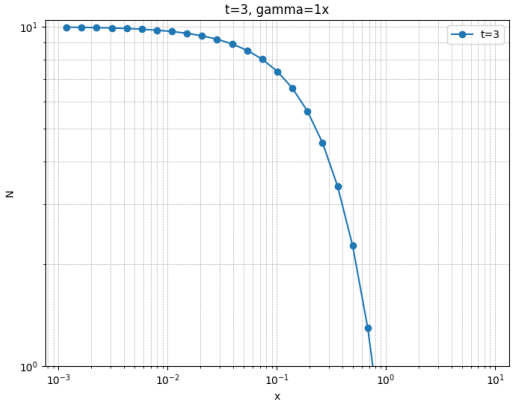
For the pure breakage process we want to solve this system of IODEs:

$$\frac{dN_i(t)}{dt} = \sum_{k=i}^{M} n_{i,k}\Gamma(x_k)N_k(t) - \Gamma(x_i)N_i(t).$$

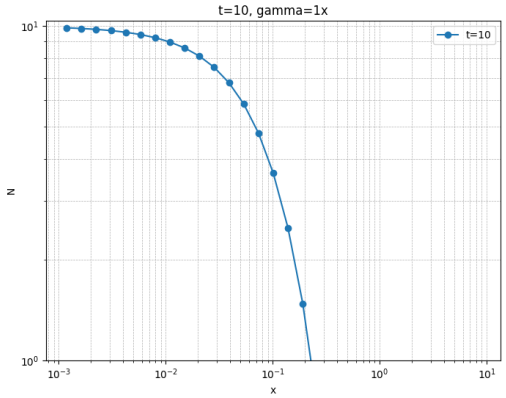
We started by firstly just implementing and testing the ODEs for the death term:

$$\frac{dN_i(t)}{dt} = -\Gamma(x_i)N_i(t).$$

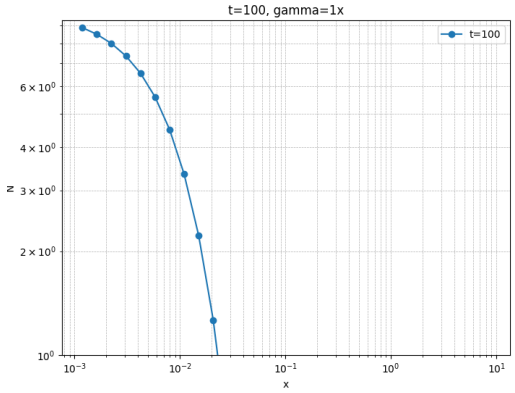
This implementation is very straightforward and does not involve any interesting choices for boundary pivots. Also, there is a really intuitive behavior when the breaking particles leave the system. We wanted to first test this behavior and then expand the ODEs with the birth term of breakage to get the system of IODEs for the pure breakage. Of course, without the birth term, no conservation of volume is to be expected.
The behavior we wanted to test is that with longer simulations more and more particles vanish from the system, especially the larger ones. For this we chose a linear dependent breakage frequency Γ, a uniform initial condition where each pivot starts with 10 particles of their given volume and plotted the results at $t = 3$, $t = 10$ and $t = 100$.



(a) Distribution at t=3



(b) Distribution at t=10



(c) Distribution at t=100

Figure 8.5: Pure breakage death term

We can see the expected behavior in Figure 8.5. The breakage frequency depends on the volume of the particles, and therefore the population in our system should decrease from right to left. The number of particles at our first pivots remain almost untouched, while over time the number of particles in the larger pivots start to vanish completely.

Now we expand the breakage process with the birth term. The only new factor introduced into the system is the $n_{i,k}$ term given in (6.2). Recall its definition:

$$n_{i,k} = \int_{x_i}^{x_{i+1}} \frac{x_{i+1} - v}{x_{i+1} - x_i} \beta(v, x_k) dv + \int_{x_{i-1}}^{x_i} \frac{v - x_{i-1}}{x_i - x_{i-1}} \beta(v, x_k) dv.$$

We started the implementation without the analytical solution and instead used the trapezoidal rule for general $\beta(v, x_k)$. Here are two interesting cases to consider. The first is $n_{1,k}$. This describes the contribution of the breakage of a particle at x_k to the first pivot x_1. Since in this case we are only interested in particles that break into the interval $[x_1, x_2]$, we can set the second integral to be equal to zero.
Similar for the other boundary pivot x_M. It is not possible for a particle of size x_M

or less to break into (x_M, ∞). Hence, we set the first integral from $n_{M,k}$ to be equal to zero. With these assumptions, the volume is conserved completely.

Algorithm 2 System Breakage Computation

Input: t, N, x
Output: dN
$M \leftarrow$ length of N
$dN \leftarrow$ array of zeros with length M
for $i \leftarrow 0$ to $M - 1$ **do**
    $break\_sum \leftarrow 0$
    for $k \leftarrow 0$ to $M - 1$ **do**
        if $k \geq i$ **then**
            $break\_sum \leftarrow break\_sum + n(i, k) \times \Gamma(k) \times N[k]$
        end if
    end for
    $dN[i] \leftarrow break\_sum - (\Gamma(i) \times N[i])$
end for
return dN

To further analyze the behavior of pure breakage, we examine the system under a unit initial condition with varying breakage frequencies. The grid consists of 1000 pivots, geometrically spaced between 1×10^{-4} and 10. The initial condition is uniform, with a value of 0.2 at each pivot and $T = 1$.

We evaluated four distinct breakage frequencies:

- $\Gamma(x_k) = 1$
- $\Gamma(x_k) = 2$
- $\Gamma(x_k) = x$
- $\Gamma(x_k) = 2x$

These tests serve two purposes: first, to compare the effects of constant versus linearly dependent breakage frequencies, and second, to investigate the impact of the additional factor of 2 in both cases.

For constant breakage frequencies ($\Gamma(x_k) = 1$ and $\Gamma(x_k) = 2$), we expect a uniform reduction in particle concentration at all pivots, with the higher frequency leading to faster depletion. In contrast, linear breakage frequencies ($\Gamma(x_k) = x$ and $\Gamma(x_k) = 2x$) introduce a size-dependent variation, where larger particles break more frequently, resulting in a redistribution of volume towards smaller pivots. The additional factor of two in $\Gamma(x_k)$ is anticipated to amplify these effects, accelerating both the depletion of larger particles and the accumulation at smaller pivots.

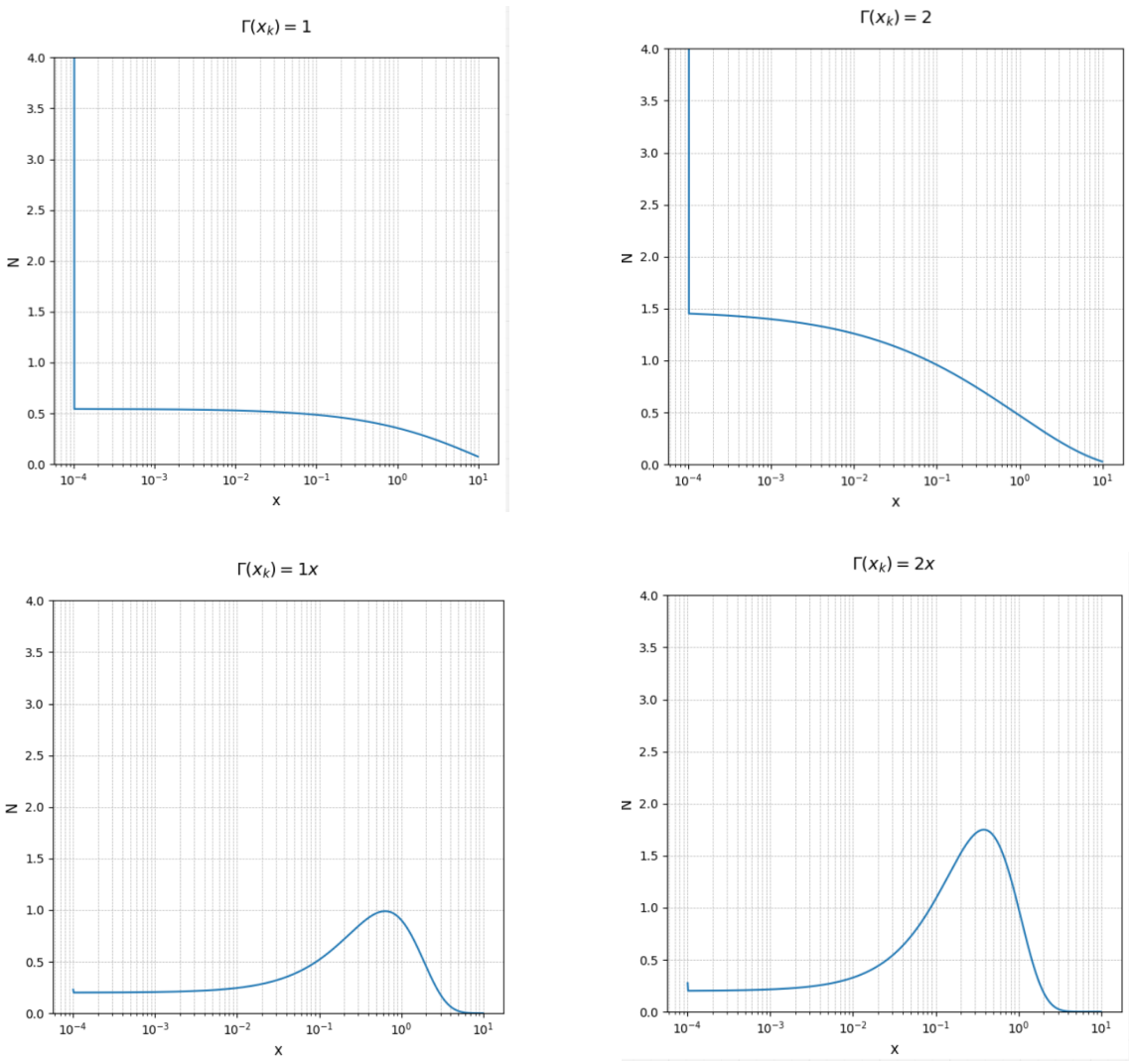


Figure 8.6: Binary breakage with uniform initial condition

For constant breakage frequencies $\Gamma(x_k) = 1$ and $\Gamma(x_k) = 2$, the breakage probability is independent of the particle size, which means that all particles break at the same rate, regardless of their size. This leads to the following observation seen in Figure 8.6:

Since breakage always results in smaller fragments, large particles can only lose volume, while small particles accumulate volume from breakage events.
A significant outlier is observed at the first pivot, where the particle concentration is much higher than in neighboring pivots. The first pivot has a value of about 30 and 100 respectively. This can be explained by three effects.

- All breakage processes have a probability of producing fragments that fall into the smallest pivot.
- The smaller the breaking particle, the higher the probability that at least one fragment lands in the smallest pivot.

- Particles in the smallest pivot cannot undergo further breakage, leading to accumulation over time.

Increasing the frequency of breakage from $\Gamma(x_k) = 1$ to $\Gamma(x_k) = 2$ strengthens these effects, accelerating the redistribution of particles and making accumulation at small sizes even more pronounced.

For linear breakage frequencies $\Gamma(x_k) = x$ and $\Gamma(x_k) = 2x$, the breakage probability increases with particle size. Unlike in the constant case, the resulting distribution does not monotonically increase toward smaller particles. Instead, we observe a peak at an intermediate particle size, leading to the following insights:

- Small particles have low breakage rates due to their small $\Gamma(x_k)$ values, resulting in fewer breakage events and lower accumulation in this region.
- Larger pivots represent wider size intervals, increasing the probability that breakage fragments land in them.

The first pivot is still a small outlier, but its relative prominence is greatly reduced. The total volume of the system is dominated by large particles, and the reduced breakage rate in small sizes prevents excessive accumulation at the smallest pivot.

The transition from constant to linear breakage frequencies results in a shift in the particle concentration from small to intermediate sizes. Furthermore, increasing the breakage factor from $\Gamma(x_k) = x$ to $\Gamma(x_k) = 2x$ amplifies all trends: In both the constant and linear cases, higher breakage rates accelerate the redistribution of particles and intensify the observed effects.

In the previous test, we observed that for linear breakage frequencies, the resulting particle distributions exhibited a shape similar to a normal distribution. Furthermore, we noted that for higher breakage frequencies, the peak of the distribution was both higher and shifted further to the left compared to lower breakage frequencies. In the following test, we investigate this behavior in more detail.

The computational grid consists of 1000 pivots, logarithmically spaced between 10^{-5} and 10. The simulation runs until a final time of $T = 1$. The initial condition is given by a normal distribution centered around the 500th pivot with parameters $\mu = 1$ and $\sigma = \frac{1}{2}$.

The breakage frequency is defined as a linear function $\Gamma(x_k) = a \cdot x_k$,

where we conduct the test for different values of a, specifically:

$$a = 1, \quad a = 5, \quad a = 10, \quad a = 100.$$

Since higher values of a lead to increased breakage rates, we anticipate a stronger shift of the distribution peak toward smaller particle sizes. Additionally, because of the geometric spacing of the grid, this shift is expected to result in a higher peak, as smaller pivots correspond to lower volume fractions.

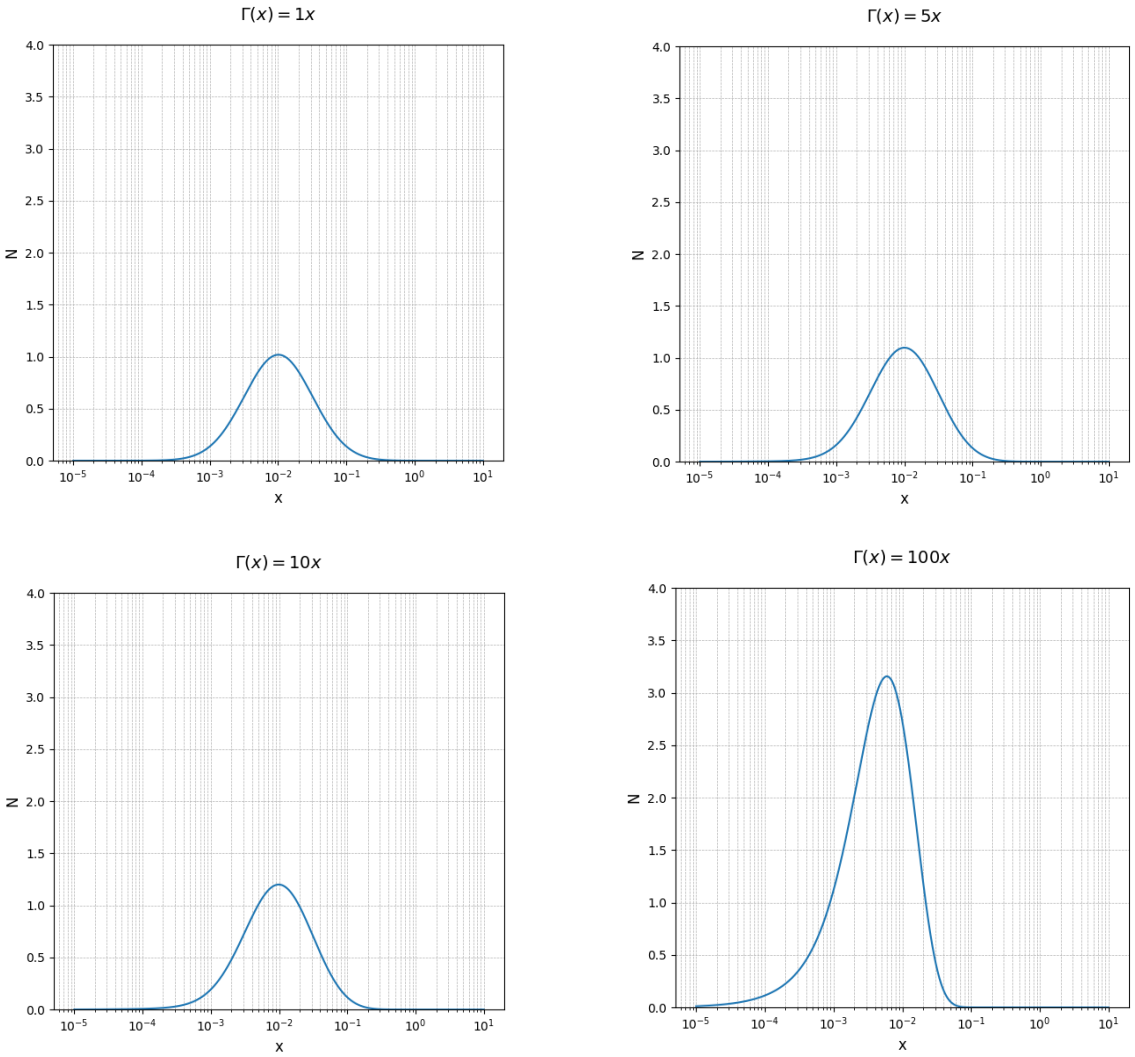


Figure 8.7: Binary breakage at $t = 1$ with normal distributed initial condition

We can see in Figure 8.7 that with different values for a the peak of the distributions indeed changes. Increasing the value of a increases the frequency of breakage. Therefore, with more particles breaking, the amount of smaller particles increases faster and the peak of the distribution shifts more to the left.
Not only does the position of the peak change, but also the number of particles at the peak. With more breakage processes happening and the volume being conserved, it only makes sense that a left shift also implies more particles at the peak, and a right shift implies less particles in comparison. This can be seen in Table 8.1.

a	max pivot	$N_{\max}$	interval
initial condition	500	1	[0.0100;0.1020]
1	500	1.0191	[0.0100;0.1020]
5	499	1.0979	[0.0099;0.0100]
10	498	1.9998	[0.0097;0.0099]
100	430	3.1569	[0.0059;0.0060]

Table 8.1: Peaks of distributions with different values of a

8.3 Simultaneous breakage and aggregation

In this section, we extend our previous analysis by considering the simultaneous occurrence of breakage and aggregation. Unlike the separate cases of pure aggregation or pure breakage, here both processes interact dynamically, influencing the evolution of the particle size distribution.

From an implementation standpoint, this extension was straightforward as it only required combining the already established aggregation and breakage codes. No additional modifications or special cases were necessary.

To examine the interaction between aggregation and breakage, we performed a couple of simulations with slightly different aggregation kernels and breakage frequencies. We chose

$$\Gamma(x_k) = ax_k,$$

where a is a scaling parameter that determines the rate at which larger particles break apart. This choice favors the breakage of larger particles.

For the aggregation kernel we chose:

$$Q(x_k, x_j) = \frac{1}{bx_k x_j},$$

where b controls the strength of aggregation. The larger b is, the weaker is the aggregation. This kernel favors smaller particles aggregating, as interactions between large particles become increasingly rare.

The grid consists of 50 pivots, geometrically spaced between 10^{-3} and 10. The initial condition is normal distributed around pivot 25 with $\mu = 1$ and $\sigma = 0.3$. The simulation ran until $T = 1$.

In Figure 8.8 one can see the initial condition and three different combinations of breakage frequencies and aggregation kernels.

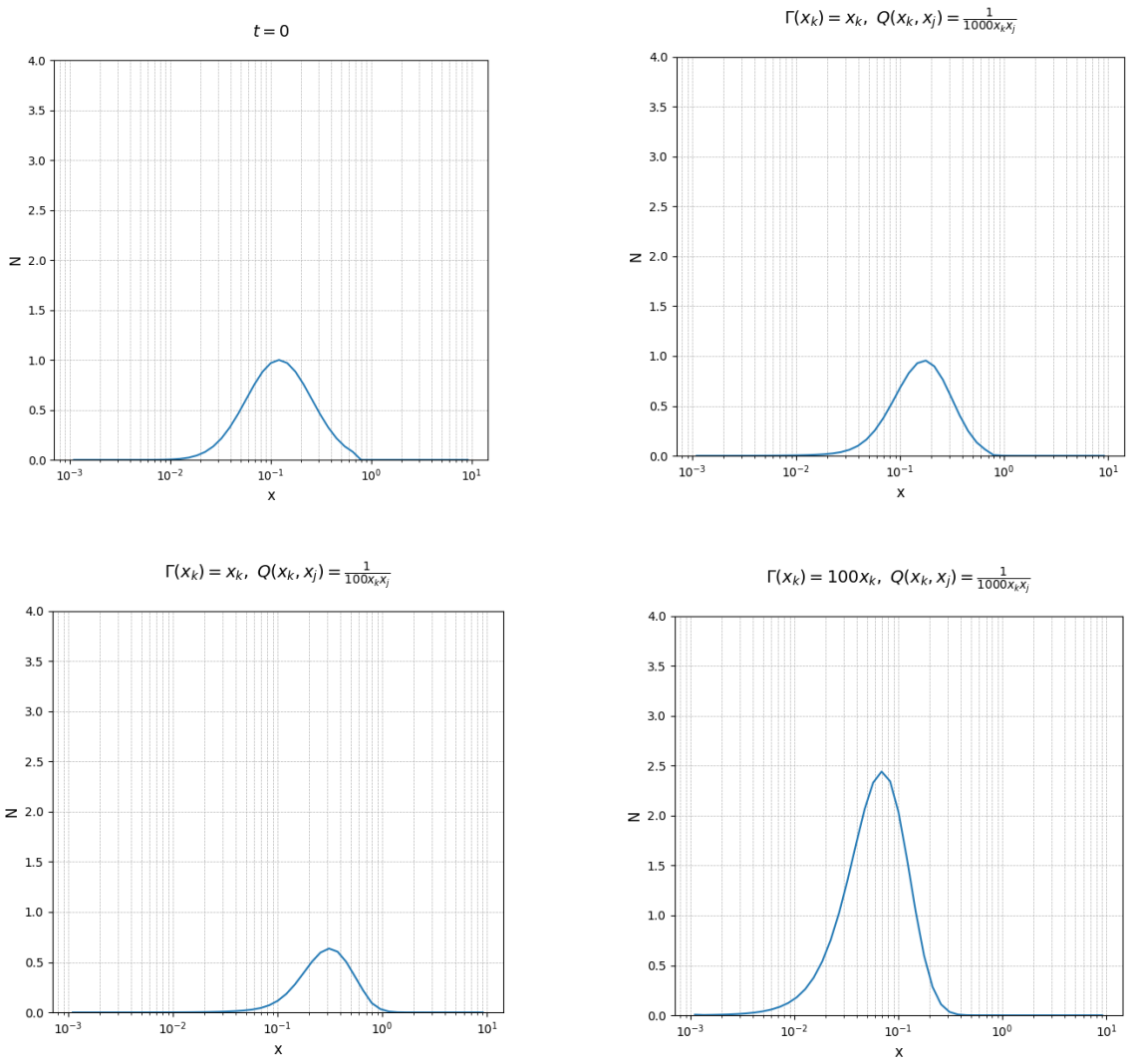


Figure 8.8: Simultaneous breakage and aggregation with normal distributed initial condition

To understand the differences of the final distributions, one can look at the results in Table 8.2. There, one can see, for each combination of a and b used, the respective peak, the average volume of particles, and the number of total particles left in the system.
When setting $a = 1$ and $b = 1000$, the final distribution remains close to the initial condition, with only a minor shift of the peak from pivot 25 to pivot 27. This slight movement to the right indicates that while breakage and aggregation are both active, they nearly counterbalance each other, leading to only a small change in the overall particle distribution. The number of particles remains relatively stable, with only a slight decrease due to aggregation outpacing breakage a little bit.

Increasing the aggregation strength by reducing b to 100 leads to a more pronounced shift in the distribution peak, which now moves further to the right, settling around the pivot 30. This is expected as a stronger aggregation kernel causes smaller particles to combine more rapidly, leading to a net increase in the number of larger particles. Consequently, the average particle size increases, and the peak broadens

slightly, reflecting a redistribution of mass towards larger sizes.

In contrast, when the breakage rate is increased by setting $a = 100$ while keeping $b = 1000$, the peak shifts in the opposite direction, moving from pivot 25 to pivot 22. With larger particles now breaking apart much more frequently, the overall particle population shifts towards smaller sizes, increasing the total number of particles in the system. Due to the conservation of volume, this increase in particle count must correspond to a decrease in the average particle size. The higher breakage rate prevents the particles from growing too large, ensuring that the mass remains more concentrated in the smaller size range.
Overall, these behaviors match what we previously saw in the simulation of pure breakage and pure aggregation.

a	b	max pivot	average volume	total particles
initial condition	initial condition	25	0.15379	9.94
1	1000	27	0.188110	8.09674
1	100	30	0.32275	4.73661
100	1000	22	0.07124	21.457633

Table 8.2: Peaks and average volume of distributions with different kernels

To further analyze the behavior of simultaneous breakage and aggregation, we extend our study to an initial condition consisting of two distinct peaks rather than a single normal distribution. The goal is to observe whether these two peaks remain separate over time or merge into a single dominant peak.
For this simulation, we used the same geometric grid, but with 85 pivots instead of 50. This allows for a wider separation between the two initial peaks.
We use the same aggregation kernel and breakage frequency as before, with parameters $a = 10$ and $b = 1000$.
The initial condition is defined as the sum of two normal distributions, centered at pivots 30 and 50, each with $\sigma_1 = \sigma_2 = 0.2$ and with $\mu_1 = 4$ and $\mu_2 = 0.5$. These normal distributions contain roughly the same volume.
The simulation runs until a final time of $T = 1$, and the results can be seen in Figure 8.9

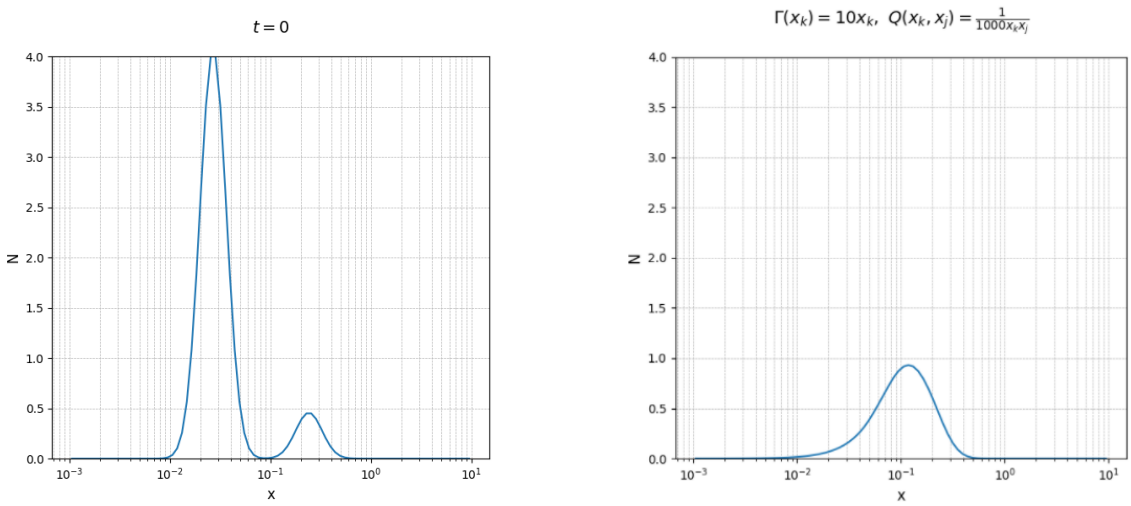


Figure 8.9: Simultaneous breakage and aggregation with the sum of two normal distributions as initial condition

The results indicate that the two initially separate peaks rapidly merge into a single peak. Shortly after the simulation begins, the distributions shift toward each other, forming a single dominant peak. From that point onward, the system behaves exactly as in the first test case, with aggregation and breakage shaping the final steady-state distribution in the same manner.
This can be explained by different factors:

- The aggregation kernel strongly favors the combination of small particles, which means that the mass is naturally redistributed across the grid.
- Additionally, breakage contributes to this merging effect by constantly redistributing the mass to smaller sizes. Since larger particles are more likely to break, they generate fragments that populate the size range between the two original peaks.
- Furthermore, due to uniform breakage, there is no bias for particles to break nearby peaks.

The simulations conducted in this thesis have demonstrated that the combined effects of breakage and aggregation behave according to physical intuition and theoretical expectations.
In every simulation, the volume was conserved almost exactly, ensuring the numerical correctness of the implemented methods and validates the chosen discretization and the numerical integration technique applied.
However, despite these successful validations, potential challenges and limitations remain. Certain special cases or extreme parameter choices may introduce numerical instabilities or unexpected behaviors that were not encountered in the tested scenarios.

Chapter 9

Outlook

The fixed pivot technique, as demonstrated in this thesis, provides an effective and volume-conserving method for solving population balance equations. It upholds physical principles, as evidenced by the logical and consistent behavior observed in various numerical scenarios discussed.

However, in [KR96b] it is highlighted that the technique can occasionally lead to over prediction of aggregation rates. This issue is particularly evident when using a coarse grid in which the number density between adjacent cells exhibits significant variations. In these cases, the analytical solution shows discrepancies that underline the limitations of the fixed pivot approach when dealing with steep gradients in number density.

One potential solution to mitigate this overprediction is to refine the computational grid, thus reducing the sharp contrasts in number density between cells. By employing a finer grid, the technique can better capture subtle variations and improve accuracy. This would of course increase the computational time.

Alternatively, adapting the fixed pivot approach to the moving pivot technique offers another promising avenue. In the moving pivot method, instead of having a fixed pivot at the center of each cell, each cell is assigned a pivot whose position can vary dynamically. This flexibility allows the pivot to adjust based on the local number density distribution: for regions with gradual changes in number density, the pivot remains near the cell center, whereas in regions with steep gradients, the pivot shifts towards the left edge of the cell. If the number density in such a steep-gradient cell transitions towards a more uniform distribution, due to processes like aggregation or breakage, the pivot will naturally migrate back towards the center.

By allowing the pivot position to reflect the variations in number density more accurately, the moving pivot technique holds the potential to enhance the representation physical behavior in population balance models.

Bibliography

[KR96a] Sanjeev Kumar and D. Ramkrishna. “On the solution population balance equations by discretization - 1. A fixed pivot technique”. In: *Chemical Engineering Science* 51 (1996).

[KR96b] Sanjeev Kumar and D. Ramkrishna. “On the solution population balance equations by discretization - 2. A moving pivot technique”. In: *Chemical Engineering Science* 51 (1996).

Chapter 10

Code

File 1: main.py

```
import numpy as np
import time
import matplotlib
from scipy.integrate import solve_ivp, trapezoid
from initialize import *
from plot import *

# Aggregation kernel
def Q(xk, xj):
    return xk * xj

# Left interval calculation of eta
def eta_left(x0, x1, v):
    return (v - x0)/(x1 - x0)

# Right interval calculation of eta
def eta_right(x0, x1, v):
    return (x1 - v)/(x1 - x0)

# Breakage frequency
def gamma(x):
    return 100*x

# Binary breakage
def beta(xk):
    return 2 / xk

# Precomputation of nik with trapezoid rule
def nik_precompute(x_values, M):
    n_ik_matrix = np.zeros((M, M))

    for i in range(M):
        for k in range(i, M):  # Only calculate when k >= i
            x0 = x_values[i - 1] if i > 0 else 0  # Handle boundary
                case for i=0
            x1 = x_values[i]
            x2 = x_values[i + 1] if i < M - 1 else x1  # Handle
               boundary case for i=M-1
            xk = x_values[k]
```

```
38
39              # Check for special cases and calculate n_ik
40              if i == k:
41                  v2 = np.linspace(x0, x1, 20)  # Values for second
                        integration
42                  integrand2 = (v2 - x0) / (x1 - x0) * beta(xk)
43                  n_ik_matrix[i, k] = trapezoid(integrand2, v2)
44              elif i == 1:
45                  v1 = np.linspace(x1, x2, 20)  # Values for first
                        integration
46                  integrand1 = (x2 - v1) / (x2 - x1) * beta(xk)
47                  n_ik_matrix[i, k] = trapezoid(integrand1, v1)
48              else:
49                  v1 = np.linspace(x1, x2, 20)  # Values for first
                        integration
50                  integrand1 = (x2 - v1) / (x2 - x1) * beta(xk)
51                  integral1 = trapezoid(integrand1, v1)
52
53                  v2 = np.linspace(x0, x1, 20)  # Values for second
                        integration
54                  integrand2 = (v2 - x0) / (x1 - x0) * beta(xk)
55                  integral2 = trapezoid(integrand2, v2)
56
57                  n_ik_matrix[i, k] = integral1 + integral2
58
59      return n_ik_matrix
60
61  # Precomputation of nik with analytical solution
62  def nik_precompute_ana(x_values, M):
63      n_ik_matrix = np.zeros((M, M))
64
65      for i in range(M):
66          for k in range(i, M):  # Only calculate when k >= i
67              x0 = x_values[i - 1] if i > 0 else 0  # Handle boundary
                    case for i=0
68              x1 = x_values[i]
69              x2 = x_values[i + 1] if i < M - 1 else x1  # Handle
                    boundary case for i=M-1
70              xk = x_values[k]
71
72              # Check for special cases and calculate n_ik
73              if i == k: #second integral
74                  n_ik_matrix[i, k] = (x1 - x0) / xk
75              elif i == 1:  #first integral
76                  n_ik_matrix[i, k] = (x2 - x0) / xk
77              else: #default case
78                  n_ik_matrix[i, k] = (x2 - x0) / xk
79
80      return n_ik_matrix
81
82  # Right side of pure aggregation
83  def system_agg(t, N, x):
84      M = len(N)
85      dN = np.zeros(M)
86
87      for i in range(M):
88          agg_loss = 0
```

```
        agg_birth = 0
        for k in range(M):
            for j in range(k, M):
                eta = 0
                delta = 1 if j == k else 0
                v = x[k] + x[j]

                if i != (M - 1) and x[i] <= v <= x[i + 1]:
                    eta = eta_right(x[i], x[i + 1], v)
                elif i == 0:
                    eta = 0
                elif i == M - 1 and v > x[i] :
                    eta = 0
                elif x[i - 1] <= v <= x[i]:
                    eta = eta_left(x[i - 1], x[i], v)
                Q_value = Q(x[k], x[j])
                agg_birth += (1 - 1/2 * delta) * eta * Q_value * N[
                    j] * N[k]

            if k != (M-1):
                Q_value = Q(x[i], x[k])
                agg_loss += Q_value * N[k]
        if i != (M-1):
            dN[i] = -N[i] * agg_loss
        dN[i] += agg_birth

    return dN

# Right side of pure breakage
def system_break(t, N, x, n_ik_matrix, gamma_values):
    M = len(N)
    dN = np.zeros(M)
    for i in range(M):
        break_sum = 0
        for k in range(M):
            if k >= i:
                break_sum += n_ik_matrix[i, k] * gamma_values[k] *
                    N[k]
        dN[i] +=  break_sum - (gamma_values[i] * N[i])
    return dN

# Right side of simul aggregation and breakage
def system(t, N, x, n_ik_matrix, gamma_values):
    M = len(N)
    dN = np.zeros(M)
    for i in range(M):
        break_sum = 0
        agg_loss = 0
        agg_birth = 0
        for k in range(M):
            if k >= i:
                break_sum += n_ik_matrix[i, k] * gamma_values[k] *
                    N[k]
            for j in range(k, M):
                eta = 0
                delta = 1 if j == k else 0
                v = x[k] + x[j]
```

```
                if i != (M - 1) and x[i] <= v <= x[i + 1]:
                    eta = eta_right(x[i], x[i + 1], v)
                elif i == 0:
                    eta = 0
                elif i == M - 1 and v > x[i] :
                    eta = 0
                elif x[i - 1] <= v <= x[i]:
                    eta = eta_left(x[i - 1], x[i], v)
                Q_value = Q(x[k], x[j])
                agg_birth += (1 - 1/2 * delta) * eta * Q_value * N[
                   j] * N[k]

            if k != (M-1):
                Q_value = Q(x[i], x[k])
                agg_loss += Q_value * N[k]
        if i != (M-1):
            dN[i] = -N[i] * agg_loss
        dN[i] += agg_birth + break_sum - (gamma_values[i] * N[i])

    return dN




if __name__ == '__main__':
# Initialize grid and pivots
    grid = create_grid_geo(0.00001,10,50)
    pivots = create_pivots(grid)
    M = len(pivots)
    init = create_initial_uniform(1,M)
    for i in range(25,M):
        init[i] = 0         # Set pivots near right boundary to 0
    t_start = 0
    t_end = 1

# Precompute nik matrix and gamma values
    n_ik_matrix = nik_precompute_ana(pivots, M)
    gamma_values = np.array([gamma(x) for x in pivots])


# Solve the ODE system
    start_time = time.time()


    # Event function to stop when last pivot accumalates volume
    def stop_on_last_pivot(t, N, x, n_ik_matrix, gamma_values):
        M = len(N)
        print(f"t={t:.4f}, N[M-1]={N[M - 1]:.4e}")
        tolerance = 1e-12
        return N[M-1] - tolerance

    # Set the event function to trigger when crossing zero
    stop_on_last_pivot.terminal = True  # Stop integration when the
        event is triggered
    stop_on_last_pivot.direction = 1  # Only trigger when crossing
       zero in the positive direction
```

```
197
198     # Solve the system
199     solution = solve_ivp(
200         system,
201         [t_start, t_end],
202         init,
203         max_step = 0.2,
204         method='RK45',
205         args=(pivots, n_ik_matrix, gamma_values),
206         events=stop_on_last_pivot
207     )
208
209 # Extract results
210     t_event = solution.t_events[0]  # Time at which the event
            occurred
211     end_time = time.time()
212     elapsed_time = end_time - start_time
213     print(f"Elapsed time: {elapsed_time:.4f} seconds")
214     N_solution = solution.y[:, -1]  # Values of the solution
215
216     # Find pivot with maximum number of particles
217     max = [0,0]
218     for i in range(len(N_solution)):
219         if N_solution[i] >= max[0]:
220             max[0] = N_solution[i]
221             max[1] = i
222     print(f"maximum of: {max[0]} at pivot: {max[1]}")
223
224     # Check volume conservation
225     mass_start = check_mass_conservation(init, pivots) # Volume of
            the init. cond.
226     mass_end = check_mass_conservation(N_solution, pivots)
227     print(f"Init mass: {mass_start}")
228     print(f"Final mass: {mass_end}")
229     print(f"Percent change in mass: {check_mass_change(mass_start,
            mass_end)}%")
230     print(f"stop time:  {t_event}")
231
232 # Plot results
233     fig, axs = plt.subplots(1, 3, figsize=(18, 6))  # 1 row, 3
            columns of subplots
234     plot(axs[0], pivots, init, False)  # First subplot
235     plot_log(axs[1], pivots, N_solution, False)  # Second subplot
236     plot(axs[2], pivots, N_solution, False)  # Third subplot
237     plt.tight_layout()
238     plt.show()
```

File 2: initialize.py

```
1 import numpy as np
2 from scipy.integrate import solve_ivp, trapezoid
3
4
5 # Pivots are the midpoints between consecutive grid points
6 def create_pivots(v):
```

```
    x = (v[:-1] + v[1:]) / 2
    return x

# Create geometric grid, given min/max point and number of points
def create_grid_geo(min, max, size):
    return np.logspace(np.log10(min), np.log10(max), size)

# Create geometric grid, given min point, number of points and
    stepsize
def create_grid_geo_h(min, h, size):
    grid = [min]
    for i in range(1, size):
        grid.append(grid[i - 1] * h)
    return np.array(grid)

# Create equisdistant grid
def create_grid_equi(min, max, size):
    return np.linspace(min, max, size)

# Creates uniform init. cond.
def create_initial_uniform(value, size):
    return np.zeros(size) + value

# Creates normal distribution
def create_initial_normal(mu, sigma, size):
    x = np.linspace(-3, 3, size)
    y = (1 / (sigma * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x - 0)
        / sigma) ** 2)
    y *= mu / y[size // 2]
    return y

# Creates normal distribution, centered around pos
def create_initial_normal_shifted(mu, sigma, size, pos):
    x = np.linspace(-3, 3, size)
    y = (1 / (sigma * np.sqrt(2 * np.pi))) * np.exp(-0.5 * (x /
       sigma) ** 2)
    y *= mu / y[size // 2]
    y_shifted = np.zeros(size)
    center_index = size // 2
    shift_amount = pos - center_index
    for i in range(size):
        shifted_index = i + shift_amount
        if 0 <= shifted_index < size:
            y_shifted[shifted_index] = y[i]
    return y_shifted

# Functions used to check volume conservation
def check_mass_conservation(N, x_values):
    total_mass = np.sum(N * x_values)
    return total_mass

def check_mass_conservation_int(N, pivots):
    return trapezoid(N * pivots, pivots)

def check_mass_change(v_1, v_2):
    if v_1 == 0:
        return None
```

```
    change = (v_2 - v_1) / v_1 * 100
    return change
```

File 3: plot.py

```
import matplotlib.pyplot as plt
import numpy as np

def plot(ax, x, y, show_line_only):
    """
    Plots a graph with a log-scaled x-axis and a linearly scaled y-axis

    Parameters:
        ax: The axis to plot on
        x: X-axis values
        y: Y-axis values
        show_line_only: If True, only the line is shown. If False, points are highlighted instead of a line
    """
    if show_line_only:
        ax.semilogx(x, y, '-', markersize=6)  # Line only
    else:
        ax.semilogx(x, y, 'o', markersize=6)  # Dots only

    ax.set_xlabel("x", fontsize=12)
    ax.set_ylabel("N", fontsize=12)
    ax.set_ylim(top=4)
    ax.set_ylim(bottom=0)
    ax.grid(which="both", linestyle="--", linewidth=0.5)

    ax.text(
        0.5, 1.05, r"$\Gamma(x_k)=100x_k,~Q(x_k,x_j)=x_kx_j$",
        transform=ax.transAxes,
        fontsize=14,
        ha="center",
        va="bottom"
    )

def plot_log(ax, x, y, show_line_only):
    """
    Plots a graph with log-log scaled axis

    Parameters:
        ax: The axis to plot on
        x: X-axis values
        y: Y-axis values
        show_line_only: If True, only the line is shown. If False, points are highlighted instead of a line
    """
    ax.text(
        0.5, 1.05, r"$\Gamma(x_k)=100x_k,~Q(x_k,x_j)=x_kx_j$",
        transform=ax.transAxes,
        fontsize=14,
        ha="center",
```

```
48          va="bottom"
49      )
50      if show_line_only:
51          ax.loglog(x, y, '-', markersize=6)
52      else:
53          ax.loglog(x, y, 'o', markersize=6)
54      ax.set_xlabel("x", fontsize=12)
55      ax.set_ylabel("N", fontsize=12)
56      ax.grid(which="both", linestyle="--", linewidth=0.5)
```