INAUGURAL-DISSERTATION

zur

Erlangung der Doktorwürde

der

Gesamtfakultät für Mathematik, Ingenieur- und Naturwissenschaften

der

Ruprecht-Karls-Universität Heidelberg

vorgelegt von

Karl Moritz Beutel, M. Sc.

aus Tübingen

Tag der mündlichen Prüfung: _____

Stochastic and deterministic methods for simulating the evolution of solid bodies in protoplanetary disks

Advisors:

Prof. Dr. Robert Strzodka, Institut für Technische Informatik
Prof. Dr. Cornelis P. Dullemond, Institut für Theoretische Astrophysik

Abstract

Planets emerge from the rotating disk of gas and dust surrounding young stars. Although the process of planet formation has been subject to theoretical and numerical studies for several decades, it has only very recently become possible to not only detect protoplanetary disks but also resolve their substructures. The annular accumulations of dust observed in many protoplanetary disks are suspected to provide favourable conditions for the formation of planetary cores through runaway growth.

To simulate the growth of rocky planets from agglomeration of dust grains, many orders of magnitude in particle number and mass have to be spanned, necessitating the use of statistical methods for abundant particles. However, the gravitational influence of the heaviest bodies must be accounted for with an N-body simulation in order to correctly describe features such as radial redistribution through gravitational scattering. A comprehensive simulation of planet formation thus needs to combine statistical and deterministic methods. Unlike grid-based statistical methods, representative particle methods allow for a natural combination with an N-body simulation, but they are often hampered by their high computational cost.

This work is geared towards simulation methods for planet formation processes as may occur in the dust rings observed in many protoplanetary disks. To this end, we develop an extension of the *Representative Particle Monte Carlo* method, overcoming some conceptual restrictions that heretofore impeded its use in simulating runaway growth processes. To address the problem of computational cost, we go on to devise a novel computational scheme for stochastic processes, herein referred to as the *bucketing scheme*, that enjoys linear scaling characteristics with regard to the number of representative particles, as opposed to the quadratic scaling characteristics of the traditional scheme. The bucketing scheme is built upon interval-valued calculations of the mutual interaction rates between representative particles, which are often non-trivial to implement. We therefore propose an 'interval-aware' programming paradigm to allow for a simpler implementation of interval-valued numerical routines. We test the simulation method, the computational scheme, and the programming paradigm by implementing a radially resolved statistical model of collisions and dynamical heating designed for studying runaway growth among planetesimals.

Our work lays the foundation for an efficient yet accurate hybrid simulation of protoplanetary growth processes with a combination of statistical and deterministic particle-based methods.

Kurzfassung

Planeten entstehen in den rotierenden Scheiben aus Gas und Staub, von welchen junge Sterne umgeben sind. Obgleich sich in den letzten Jahrzehnten viele Arbeiten durch theoretische Überlegungen und numerische Studien mit der Entstehung von Planeten befaßt haben, ist erst in jüngster Zeit neben dem Nachweis protoplanetarer Scheiben auch die Beobachtung ihrer inneren Struktur gelungen. Es wird vermutet, daß die Bedingungen in den ringförmigen Ansammlungen von Staub, die in vielen protoplanetaren Scheiben zu sehen sind, die Bildung von Planetenkernen durch rapide Wachstumsprozesse begünstigen.

Um in einer Computersimulation abbilden zu können, wie sich Gesteinsplaneten aus der Akkumulation von Staubteilchen bilden, müssen in bezug auf Teilchenzahl und -masse viele Größenordnungen überspannt werden. Für sehr große Teilchenzahlen ist das nur mit statistischen Methoden möglich. Gleichwohl führt der gravitative Einfluß größerer Objekte zu Erscheinungen wie der radialen Umverteilung, die nur mit einer N-Körper-Methode korrekt beschrieben werden können. Eine umfassende Simulation der Planetenentstehung muß also statistische und deterministische Methoden vereinen. Im Gegensatz zu gitterbasierten statistischen Methoden eignen sich Verfahren, die mit repräsentativen Teilchen arbeiten, auf natürliche Weise für die Verbindung mit einer N-Körper-Simulation; jedoch erfordern solche repräsentativen Methoden einen erheblichen Rechenaufwand, wodurch ihrer Nutzbarkeit Grenzen gesetzt sind.

In dieser Arbeit werden Methoden und Schemata entwickelt, um Prozesse der Planetenentstehung simulieren zu können, wie sie in den Staubringen protoplanetarer Scheiben vermutet werden. Zu diesem Zweck entwickeln wir zunächst eine Erweiterung der *Representative Particle Monte Carlo*-Methode, um diese für die Modellierung rapider Wachstumsprozesse anwendbar zu machen. Um den Kostenaufwand repräsentativer Teilchenmethoden zu verringern, konstruieren wir sodann ein neuartiges Rechenverfahren, das *bucketing scheme*, dessen rechnerischer Aufwand nicht wie beim herkömmlichen Verfahren in quadratischer, sondern nur noch in linearer Relation zur Zahl der repräsentativen Teilchen steht. Das *bucketing scheme* erfordert die Berechnung von Intervallbereichen der wechselseitigen Interaktionsraten zwischen repräsentativen Teilchen. Um die Anfertigung intervallwertiger Rechenvorschriften zu erleichtern, entwerfen wir ein Paradigma für intervallsensitives Programmieren (*interval-aware programming*). Wir erproben die Methode, das Rechenverfahren und das Paradigma in der Praxis, indem wir sie auf die Simulation eines radial auflösenden statistischen Interaktionsmodells anwenden, das für die Untersuchung rapider Wachstumsprozesse unter Planetesimalen entworfen wurde.

Auf Grundlage dieser Arbeit kann eine hybride Simulation zur Untersuchung protoplanetarer Wachstumsprozesse entwickelt werden, die statistische und deterministische Methoden kombiniert, ohne daß ihre rechnerische Effizienz die Genauigkeit ihrer Ergebnisse beeinträchtigt.

Acknowledgements

I would like to extend my gratitude to my advisors, Prof. Robert Strzodka and Prof. Cornelis P. Dullemond. Prof. Strzodka accompanied my dive into what had started as an astrophysical side project with utmost considerateness, giving me extraordinary freedom in every aspect of my work. His ideas and suggestions often led to substantial improvements, and his expertise proved invaluable especially in the arithmetic excursion in this work. Prof. Dullemond accompanied my efforts with infinite but bounded patience, and it is only thanks to his navigational aid that I have not yet drowned in the vast sea of astrophysical knowledge. With sanguine confidence he helped me through difficult times and guided me to the completion of this project. I also would like to thank Prof. Ralf Klessen and Prof. Guido Kanschat who have agreed to be part of my examination committee together with Prof. Strzodka and Prof. Dullemond.

I wish to thank all the people from the Application Specific Computing group. I am glad to have been a colleague of Christoph Klein, Andreas Wurz, and all the others who contributed to fostering an inviting and cheerful working environment. I want to express my appreciation to Kees Dullemond's Planet Formation group, especially Natalia Dzyurkevich, Peter Rodenkirch, Carolin Kimmig, Thomas Rometsch, and León-Alexander Hühn, for helpful discussions, sympathetic conversations, cake breaks, and for running the ever-instructive journal club. I thank Til Birnstiel, Sebastian Stammler, Joanna Drążkowska, and Tommy Chi Ho Lau (劉智昊) for patiently listening to my progress reports and for providing helpful comments in the ensuing discussions, and Andreas Schärtl and Oliver Keszöcze for kindly granting me access to their implementation of posits and valids.

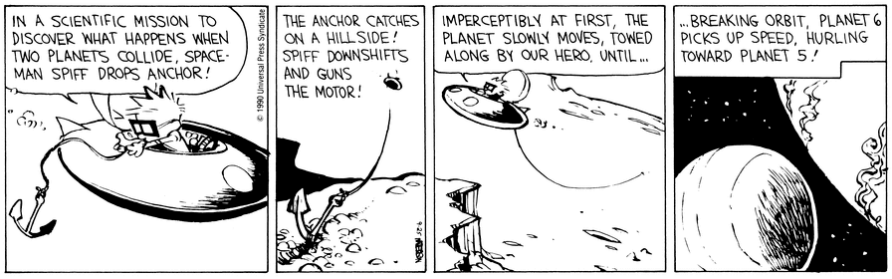
Charlotte Boys helped me cast complex ideas in simple, digestible examples. Stephan Eismann helped me structure my thoughts and the course of my work when I struggled to make progress on my own. Sophia Haude, Fabian Beutel, and Theodor Beutel kindly agreed to proofread parts of my manuscript. I greatly appreciate their efforts.

I am deeply grateful to my family and my friends. Without their unrelenting support, I would not have been able to stay the course and complete this work.

Moritz Beutel
Heidelberg
January 2024

Contents

Introduction 1



Bill Watterson, *Calvin and Hobbes*, September 25, 1990 (Andrews McMeel Syndication)

Planets are assumed to emerge from the gaseous protoplanetary accretion disk surrounding and feeding young stars. Although the core of this hypothesis is centuries old (Kant, 1755), a self-consistent model to explain the formation of planets has not yet been devised. However, great advances have been made in the last decades, owed to new experimental insights into the physical collision processes (e.g. Güttler et al., 2010; Blum, 2018; Wurm and Teiser, 2021), tangible progress in the theory of planet formation (reviewed by Drążkowska et al., 2022), as well as new observational capabilities that uncovered the abundance of dust substructures in protoplanetary disks (Andrews et al., 2018). Heavy planets embedded in protoplanetary disks are known to inhibit radial inward drift of dust and to cause adjacent annular dust accumulations, which are suspected to be sites of subsequent planet formation.

This work is concerned with methods and computational schemes for simulating the dynamics and interactions of solid bodies in protoplanetary disks. Specifically, we set out to develop a hybrid representative particle method that can be used to study the emergence of planetary cores from ensembles of planetesimals in planet-induced dust traps, focussing on the smooth transition from a representative many-particles regime to individual bodies and on an efficient computational scheme that combines rejection

sampling with interval arithmetic to obtain linear performance scaling characteristics with regard to the sampling resolution chosen.

1.1 Motivation

For a long time, little observational data had been available for comparison with model predictions of planet formation. Although the outcome of the process was of course known for our particular solar system, no direct records of previous states of this system were accessible (it was not until the last two decades that asteroids and comets would be probed to study the composition of planetesimals, see Watanabe et al. (2023)), and the existence of other planetary or even protoplanetary systems had only been conjectured. For lack of observational data, planet formation theory started with simple assumptions, such as a radial power-law profile of gas density and temperature (Weidenschilling, 1977b), and a strong coupling of dust to gas with a constant dust-to-gas ratio, typically assumed to be identical to the dust-to-gas ratio of $\sim 1:100$ in the interstellar medium (Bergin et al., 2013; Trapman et al., 2017). Weidenschilling (1977b) constrained the total gas and dust masses and the mass compositions with the material content of the Solar System. This 'minimum viable' model of the gas and dust distribution of the protoplanetary disk our planetary system emerged from is known as the *Minimum Mass Solar Nebula* (MMSN) model (see also Hayashi, 1981b).

The gas component of the protoplanetary disk, composed mainly of hydrogen and helium, is hard to observe directly, so observations of protoplanetary disks tend to be geared towards the dust component or rare tracer molecules in the gas disk. Even after the first detections of protoplanetary disks (O'dell et al., 1993) and the first confirmed exoplanet observations in the 1990s (Wolszczan and Frail, 1992), it would still take decades until the *Atacama Large Millimeter/submillimeter Array* (ALMA) was able to resolve ring-like substructures in the dust distribution of a protoplanetary disk (ALMA Partnership et al., 2015). The DSHARP survey (Andrews et al., 2018) subsequently proved the abundance of radial substructures such as arcs, gaps, rings, and spirals in protoplanetary disks. These features can be linked to the presence and formation of planets in the disk, though the specific relations are not unambiguously clear. A continuing process of planet formation seems to be a plausible explanation for their abundance (Dong et al., 2015; Zhang et al., 2015; Teague et al., 2018).

1.1.1 Growth barriers

Planet formation is assumed to be a complex growth process starting with the agglomeration of sub-micrometre dust grains (Safronov, 1972; Hayashi, 1981a), which grow until reaching approximately pebble size. Pebbles must then somehow form *planetesimals*, that is, bodies with multi-kilometre radii that are held together by self-gravitation rather than molecular forces. However, there are several barriers to continued growth by coagulation. High impact speeds make a sticking outcome less likely as the mass of target and projectile, and thus the kinetic impact energy, grows: this is referred to as the *bouncing barrier* (Zsom et al., 2010) and the *fragmentation barrier* (Güttler et al., 2010; Yamamoto et al., 2014; Blum, 2018; Wurm and Teiser, 2021), depending on the presumed outcome of high-velocity collisions. Additionally, in a gas disk with a homo-

geneous pressure gradient, particles drift inwards as a consequence of the aerodynamic drag force exerted by the gas which moves at sub-Keplerian speed, thus acting as 'headwind'. This drift is most efficient for pebble-sized objects. But if pebbles are efficiently removed, the growth process is depleted of material; in order to be sustained, further growth would need to occur on timescales shorter than the typical drift timescale. This is known as the *drift barrier* or *metre-size barrier* (Adachi et al., 1976; Weidenschilling, 1977b).

1.1.2 Formation of planetesimals and planetary embryos

Johansen et al. (2007) suggested that planetesimals might form by a gravitational collapse of pebble clumps incited by the streaming instability (Youdin and Goodman, 2005), a linear instability occurring in mixed laminar streams of gas and dust particles that develops unstable modes. This mode of planetesimal formation is expected to produce planetesimals of $\sim 100\,\mathrm{km}$ in radius, consistent with observations from Kuiper Belt objects (Simon et al., 2016; Li et al., 2019). The formation of planetesimals with the streaming instability and the conditions for its occurrence have been studied extensively by, for instance, Johansen et al. (2012); Carrera et al. (2015); Yang et al. (2017); Sekiya and Onishi (2018); Klahr and Schreiber (2020); Umurhan et al. (2020). Among other insights, it was demonstrated that a very high concentration of pebbles at the midplane was required to incite planetesimal formation. Thus, even though the streaming instability would sidestep the bouncing and fragmentation barriers that make continued accumulative growth of pebbles unviable, the radial drifting of planetesimals remains a problem.

One way to overcome the drift barrier would be a local reduction, or even reversal, of the radial pressure gradient, causing 'tailwind' and making particles drift outwards. Such a local *pressure bump* leads to accumulation of particles in a *dust trap* (e.g. Tanaka et al., 2002; Pinilla et al., 2012). And indeed, the results of the DSHARP survey have revealed an abundance of *rings*, that is, regions of high dust density, and *cavities*, or regions depleted of dust. The presence of these gap-like substructures has been linked to planets massive enough to carve a radial gap in the disk (Dullemond et al., 2018). Dust and gas are coupled aerodynamically, and a gap in the gas density distribution, tracked by a co-located gap in the dust distribution, would cause a local reversal of the radial gas pressure gradient, which then locally reverses the radial drift direction and thus accumulates pebble-sized material, allowing for subsequent growth of structures, most likely through gravitational collapse of pebble clumps (Stammler et al., 2019; Carrera et al., 2021).

For bodies that have reached planetesimal size, further growth by coagulation is feasible, and the fastest-growing bodies will experience runaway growth and oligarchic growth (Kokubo and Ida, 2000, 2002; Ormel et al., 2010) through the gravitational focussing effect (Safronov, 1972; Wetherill and Stewart, 1989; Kokubo and Ida, 1996), forming planetary embryos. Dust and pebbles may further contribute to efficient growth of planetesimals and planetary embryos by means of pebble accretion (Ormel and Klahr, 2010; Lambrechts and Johansen, 2012), that is, the aerodynamically assisted accretion of small bodies onto massive cores. Greatly oversimplifying, one could say that, for particles with a tight coupling to the gas such as pebbles, the damping exerted by the gas drag force can turn a close encounter between a pebble and the planetary embryo into a grav-

itational capture and, eventually, an accretion of the former; rather than being sharply deflected, the pebble will settle onto the embryo. This mechanism can increase the effective impact parameter and the collision cross-section far beyond the purely gravitational collision cross-section even with gravitational focussing being considered. Although not efficient at the presumed initial size of $\sim 100\,\mathrm{km}$ planetesimals (Liu et al., 2019), pebble accretion can significantly boost the growth rate of planetary embryos that have grown from mutual accretion of planetesimals.

1.1.3 Substructures in protoplanetary disks

A planet embedded in a protoplanetary disk will excite spiral waves (Goldreich and Tremaine, 1978, 1979), leading to gap opening through emergent shocks that transfer angular momentum to the gas disk (Rafikov, 2002). For planets exceeding the *thermal mass* $M_{\mathrm{th}} \equiv (h/r)^3 M_*$, where h is the disk scale height at the orbital radius r of the planet, the propagation of spiral waves already becomes non-linear on excitation, thus allowing the planet to open a gap at its own orbital location. Based on standard turbulent viscosity assumptions (Shakura and Sunyaev, 1973a), a simplified axisymmetric analytical model of the gap density structure has been developed by Kanagawa et al. (2015) and was subsequently refined by Kanagawa et al. (2017, 2020). With this analytical gap model, the structure of the gap can be related to properties of the disk and the gap-inducing planet. Because a hypothetical planet with suitable properties can be constructed for most gap observations, the assumption of a gap-inducing planet is the most flexible among the many proposed models of disk substructure formation. Such flexibility also renders the model prone to overuse; for example, a single planet may open multiple gaps, most of which are not co-located with the planet itself; from observing these gaps alone, one might wrongly infer multiple gap-carving planets.

Several other processes are known to contribute to the formation of substructures in the gas or dust profile, each with its own practical constraints. Beside influencing the disk structure, some of these processes may also directly contribute to the formation of planetesimals. One important hydrodynamical process is the streaming instability (Youdin and Goodman, 2005), already mentioned before as a possible means of converting pebble-mass particles to planetesimals through gravitational collapse, which has been suggested to also generate dust rings (Carrera et al., 2021). The gravitational backreaction from highly concentrated pebbles onto the gas may also disrupt the structure of the disk and induce a dust trap (Gonzalez et al., 2017). Gravitational instabilities, comprehensively discussed in the review of Kratter and Lodato (2016), can occur as the self-gravity of the gas begins to dominate the counteracting gas pressure. For perturbations induced by self-gravity to become unstable, the Toomre parameter $Q = c_{\mathrm{s}}\Omega_{\mathrm{K}}/(\pi G\Sigma)$, which relates the temperature (via sound speed c_{s}) to the gas surface density, needs to be $\lesssim 1$. These conditions may be plausibly met at large radii and early in the life of circumstellar disks, and the ensuing instability may emanate transient spirals capable of transporting angular momentum and trapping dust, thus accelerating planetesimal growth (Rice et al., 2004). There are several other hydrodynamical instabilities that can emerge when taking into account complex physical realities beyond a smooth 1-dimensional axisymmetric gas-and-dust model such as the MMSN model and which may contribute to the formation of the observed substructures. Two examples are the Rossby wave instabil-

ity (RWI; cf. Lovelace et al., 1999), which can form long-lived vortices, and the vertical shear instability (VSI; cf. Urpin and Brandenburg, 1998) that emerges from differential rotation along the vertical disk axis, which can produce an outcome similar to the Rossby wave instability. Additionally, the impact of magnetic fields in protoplanetary disks gives rise to magnetohydrodynamic processes which may also exhibit instabilities and facilitate the formation of structures. The list here is by no means complete; for a thorough review of protoplanetary disk processes that may lead to structure formation, see Bae et al. (2023).

Planets in protoplanetary disks are difficult to observe. Compared to the number of known dust substructures in protoplanetary disks, the number of embedded planets detected so far is small. A kinematical detection of an embedded planet in a protoplanetary disk was first claimed by Teague et al. (2018), who measured the rotation curve of the gas disk in HD 163296 by analysing interferometric observations of CO emission. They were able to reconstruct a gas pressure profile with such precision that other scenarios capable of generating gaps and rings could be ruled out, while a comparison with a model featuring two gap-carving planets showed excellent agreement. Keppler et al. (2018) discovered a planet in the PDS 70 system, and Haffert et al. (2019) found a second planet in the same system, both co-located with observable gaps in the disk structure. More recent work has documented the detection of AB Aurigae b and AS 209b (Currie et al., 2022; Bae et al., 2022). However, such observations are rare; in most cases, non-axisymmetric features observed in protoplanetary disks so far could not be unambiguously associated with a planet. Nevertheless, the scarcity of confirmed gap-carving planets is not contrary to the hypothesis that many of the observed substructures might emerge from planet-disk interaction. As observational sensitivity and coverage increases, the inferred constraints are expected to tighten, and more annular substructures may become eligible for attribution to certain causes even in cases where direct detection of planets remains improbable. Therefore, a better understanding of the dust and gas dynamics in the vicinity of planets is crucial to the interpretation of observational results.

1.2 Simulation techniques

Gravitational multi-body interactions involving heavy bodies are generally chaotic in nature, which necessitates a simulation with deterministic N-body calculations. But a gravitational N-body simulation costs n^2 operations per timestep for n mutually interacting bodies (barring approximations based on a multipole expansion which are not easily applicable to scenarios with a high turnover such as a revolving protoplanetary disk). Even if the algorithmic complexity of the simulation was lower, simulating all individual particles in such a ring would overburden every computer system existing to date. At the same time, the bulk of the information generated by such a simulation would concern the individual states and fates of relatively small and plentiful particles, which are of limited interest individually. Therefore, statistical methods are usually employed to simulate the dynamics of highly interactive systems with a large number of bodies. Typical examples are grid-based methods, as used in the dust model of Birnstiel et al. (2010), or stochastic representative particle methods, such as the Monte Carlo method used by Ormel et al. (2010).

1.2.1 Statistical simulation methods

The conceptual advantages and disadvantages of particle-centric ('Lagrangian') methods, which follow the state-space trajectories of individual particles, and grid-based ('Eulerian') methods, which simulate the distribution of particles, are well-understood. A grid-based code will necessarily introduce numerical diffusion, which negatively affects the precision of highly parameter-sensitive growth processes and renders grid-based methods inappropriate for the simulation of runaway growth processes (e.g. Stammler and Birnstiel, 2022). Also, a grid-based method will suffer from the so-called curse of dimensionality, which refers to the fact that adding a dimension to the space of properties and resolving it with $\mathcal{N}$ grid cells increases the computational effort by a factor of $\mathcal{N}^2$, thus imposing severe computational limitations on the possible consideration of additional parameters. Both deficiencies can be avoided by using a particle-based representative method instead.

Conversely, because an Eulerian code operates on the discretised particle number distribution, it handles cancellation effects in a quasi-steady-state process more gracefully than particle-based methods. For instance, colliding planetesimals may produce dust-mass fragments, but planetesimals may also accrete dust; under certain conditions, this cyclic process might enter a steady state where the net mass of dust barely changes even though planetesimals produce and accrete dust all the time. A grid-based method only operates on the sum of these effects, and because the net change in dust mass per grid cell is small, a large timestep may be used. Moreover, with implicit integration, stable results may be possible to obtain for even larger timesteps. The computational cost of both grid-based methods and particle-based methods with the traditional computational scheme has quadratic scaling characteristics (that is, the number of operations required per time increment scales quadratically with the number of grid cells or the number of representative particles, respectively); but a particle-based simulation must follow the evolution of its individual constituents, and therefore it cannot benefit from near-cancellation: a representative planetesimal in a steady-state system as sketched above will constantly experience cratering events and accretion of dust, and a particle-based simulation will be busy simulating them while a grid-based simulation only needs to operate on the net difference in dust and can thus simply leap ahead in time.

Particle-based simulations face the additional challenge that representative weights need to remain balanced to avoid undersampling, which impacts correctness, and oversampling, which is inefficient. Because representative particles may accrue representative weight during an interaction, a selection of sampling weights may become skewed even if it was balanced initially, which effectively reduces the resolution of the simulation. In the super-droplet method of Shima et al. (2009), this has proven challenging and had to be counteracted by choosing very large numbers of representative samples to ensure that enough resolution is retained by the end of the simulation. The Monte Carlo method of Ormel and Spaans (2008) relies on an explicit management of zoom factors, which entails continuously adding new representative particles and merging existing particles, to retain a balanced weight sampling when covering a high dynamic range of masses. The *Representative Particle Monte Carlo* (RPMC) method of Zsom and Dullemond (2008) is an alternative method designed for dust coagulation processes that has the conceptual advantage that it inherently maintains an equal-weight mass sampling

and thus avoids this complication; however, the method is built upon assumptions that make it inapplicable to runaway growth processes, where a single body may grow to a mass larger than the total mass fraction it was initially chosen to represent.

The growth of planetesimals has been subject to many numerical and analytical studies, and quasi-thermal models of planetesimal kinetics such as viscous stirring and dynamical friction have been developed, demonstrating that planetary embryos can emerge efficiently by means of runaway growth, and that runaway bodies will dynamically isolate in the oligarchic growth regime (e.g. Wetherill and Stewart, 1989; Ida, 1990; Ida and Makino, 1992a,b; Kokubo and Ida, 1996, 1998). In Ida and Makino (1993), an N-body code was used to study the interaction between a protoplanet and a population of planetesimals and found that the protoplanet decreases its own growth rate by gravitationally stirring the planetesimals, thereby reducing the effective collision cross-section because gravitational focussing is less effective at higher relative velocities. Their semianalytical formulae capture the statistics of the interaction between protoplanet and planetesimals, but they are of inherently local nature and thus not suitable for resolving a spatially inhomogeneous system. This deficiency was overcome by Ormel et al. (2010) who developed a geometrical interaction model that incorporates radial reach and locality: using a representative Monte Carlo method adapted from the dust coagulation method of Ormel et al. (2007), they reproduced the orbital separation characteristics of oligarchs previously described by Kokubo and Ida (1998) who had used an N-body code.

1.2.2 Hybrid statistical and deterministic methods

Statistical models have limited applicability with regard to individual bodies. Thus, as a planetary embryo emerges from an ensemble of planetesimals, it needs to be tracked as an individual object in order to correctly reproduce features such as the separation from the continuous mass distribution during runaway growth. Not all statistical methods are well-suited to the emergence of individual bodies. The representative particle method proposed by Ormel et al. (2007) naturally extends to individual bodies, whereas the RPMC method of Zsom and Dullemond (2008) breaks down as representative bodies begin to exceed the total mass fraction they represent.

Even if a statistical method extends to individual bodies, a statistical interaction model cannot do justice to the chaotic nature of interactions between individual bodies. Ida (1990) and Ida and Makino (1993) were able to describe the dynamics of a large ensemble of highly interactive particles with quasi-thermal semianalytical formulae; but the quasi-thermal picture cannot predict the occurrence or outcome of individual encounters between individual bodies, and it is therefore unclear how radial redistribution of particles, such as migration or gap opening, could be represented correctly by statistical means. Instead, it would be desirable to simulate the interactions of heavy individual bodies as N-body particles without foregoing the computational advantages of a statistical simulation. Such a hybrid method was hinted at by Ormel et al. (2010) but deferred to future work. Levison et al. (2012) developed *LIPAD*, a hybrid of an N-body code and a stochastic simulation. In this code, small bodies are represented as *tracers* whose mutual interaction is implemented with a representative Monte Carlo approach resembling the RPMC method of Zsom and Dullemond (2008), but which can additionally interact with heavy individual bodies through forces incorporated with N-body calculations.

The number of particles represented by a given tracer shrinks as the tracer's mass grows, and when the tracer ends up representing only itself, it is promoted to an embryo whose dynamics is determined strictly by N-body interactions. LIPAD was demonstrated to describe runaway growth correctly, though when applied to the product kernel test case of Wetherill (1990) it showed a slight inaccuracy which the authors attributed to a lack of resolution. LIPAD has been used successfully to study oligarchic growth processes (e.g. Voelkel et al., 2021a,b). Other authors have enriched N-body codes with statistical methods as well. More recently, Turrini et al. (2019) combined an N-body code with a radial binning method and a geometrical collision model to study the production of dust through collisions of gravitationally stirred planetesimals, and Bernabò et al. (2022) built upon this N-body code but replaced the binning approach with a particle-based statistical method. Aiming to study the formation and growth of planetesimals in a pressure bump, Lau et al. (2022) coupled the grid-based dust evolution code of Stammler and Birnstiel (2022), based on the dust model of Birnstiel et al. (2010), with an N-body code into a hybrid method that also includes gas drag, migration, and pebble accretion. Assuming that planetesimals emerge from dust through the streaming instability, they elegantly bridged the gap between the two simulation methods by directly promoting newly formed planetesimals to N-body objects.

1.3 Outline

To provide some physical context for the subsequent technical chapters, in Chapter 2 we start by introducing some basic physical processes at work in protoplanetary disks, culminating in a back-of-the-envelope model of runaway growth. We also discuss the different simulation methods usually employed for protoplanetary coagulation processes.

A simulation of solid body interactions that ranges from dust grains to planets is a numeric and computational challenge because it spans so many orders of magnitude in mass and at the same time needs to cover many degrees of freedom. Due to the sheer number of particles in an annular dust substructure of a protoplanetary disk, the method must be of statistical nature; at the same time, we want to allow the emergence of self-representing bodies which need to be treated individually, and which may have to be handled with N-body calculations.

The RPMC method as originally introduced in Zsom and Dullemond (2008) works well with dust distributions, but as a statistical method, it requires that all of the particles represented in the simulation have a high multiplicity, that is: for every representative particle j there must be $N_j \gg 1$ non-representative particles with similar properties. This renders the method unsuitable for runaway growth processes, where the heaviest particles can no longer be treated stochastically. The first concern of this work is to overcome this limitation. In Chapter 3, we formally define the RPMC method of Zsom and Dullemond (2008) and subsequently extend it such that even representative particles whose multiplicity approaches unity can be treated correctly. With this improvement, the RPMC method becomes suitable for simulating runaway growth. The correctness of the method is proven analytically and verified numerically for a simplified two-component scenario. The extended method is then tested extensively with the standard test kernels for the Smoluchowski equation, among them the product kernel

which exhibits runaway growth characteristics, and with a synthetic test case modelled after the gravitational focussing effect.

Our extensions to the RPMC method necessitate adding and removing representative particles during the simulation, which the original RPMC method had been designed to avoid. To accurately model the transition from representative particles with high multiplicity, or 'swarms', to individual bodies, swarms are split up into individual representative particles when their particle count drops below a certain threshold value N_{th}, and may thus significantly increase the number of representative particles required. This poses a challenge because, in the computational scheme traditionally employed for the RPMC method, n^2 interaction rates had to be computed and maintained for an ensemble of n representative particles. In Chapter 4, we thus devise a new computational scheme, dubbed the *bucketing scheme*, which enables us to conduct a representative particle simulation with substantially lower requirements in computation and storage. We develop cost models for the computational and memory requirements of both the traditional scheme and the bucketing scheme, and we verify them with extensive performance testing. With the bucketing scheme, we can easily afford the slightly higher computational requirements of the extended RPMC method of Chapter 3, and generally RPMC simulations with much higher resolution and higher-dimensional property spaces become viable. Moreover, we develop a location-aware extension to the bucketing scheme that enables us to efficiently apply the extended RPMC method to spatially resolved systems where interactions have limited reach, as in the oligarchic growth scenario. The efficiency of the scheme is tested with several standard tests and with a planetesimal growth model adapted from Ormel et al. (2010), also including a simple fragmentation model.

Although the bucketing scheme has very favourable performance characteristics compared to the traditional scheme, it is not a panacea. The bucketing scheme constructs and continually updates a grouping of representative particles in 'buckets' and keeps track of the mutual interaction rate bounds between all pairs of buckets, using rejection sampling to sieve the actual events from the surplus of events generated at bucket level. To this end, the simulation must be able to compute interaction rate bounds for pairs of buckets, which requires significant effort on the part of the implementor. To alleviate this effort, interval arithmetic has proven invaluable. Although software libraries implementing interval arithmetic are widely available, we found that implementing a given numerical routine for interval-valued arguments typically requires an approach very different from the straightforward computation for real-valued arguments, and the numerical routine would thus have to be implemented twice – for real-valued and for interval-valued quantities –, which is onerous and error-prone. In Chapter 5, we therefore propose a paradigm of *interval-aware programming*. Interval arithmetic can be viewed as a 'lifting' of real-valued arithmetic operations to interval-valued arguments; thus, by means of the composition theorem (cf. Sect. 5.1.5), a given arithmetic expression can already be interpreted generically, that is, the same expression can be applied either to real-valued or to interval-valued arguments. In this chapter, we carry this generic paradigm further and apply it to partial function definitions, or equivalently, runtime branches in code. We also demonstrate how the notion of interval-aware programming carries over into the integer domain, and how even iterator-based algorithms can be implemented in an interval-aware manner. The goal of this undertaking is to provide a set of tools and

semantic guidelines that, given a conventional implementation of a numerical routine, empower the programmer to construct an interval-aware equivalent of the routine, and then to compile it as either a real-valued or an interval-valued routine.

We aim to study the process of planetesimal growth in the dust trap of a gap-carving planet using the geometrical stirring and collision model developed by Ormel et al. (2010). The model describes dynamical heating and collision processes and is designed to study the dynamics of runaway growth. Although this model can handle interactions between individual bodies and representative particles, it still uses statistical methods to simulate interactions with and among individual bodies. As bodies become more massive, the outcome of a gravitational interaction becomes harder to predict statistically; in particular, we know of no statistical model for predicting the radial redistribution of bodies in a close encounter, and processes such as the planet-induced gap opening in a distribution of planetesimals cannot be simulated with this method. The extended RPMC method was designed to be compatible with continuous external operators (such as the aerodynamic drift induced by the protoplanetary gas disk) and deterministic N-body methods for self-representing particles. In Chapter 6, we elaborate on our adaptation of the geometrical model of Ormel et al. (2010) and its implementation with the extended RPMC method and the bucketing scheme. We then sketch a hybrid simulation method that complements the extended RPMC method with a continuous operator effectuating the aerodynamic drift by the protoplanetary gas disk and with a deterministic gravitational N-body code to simulate the parameter-space trajectories of self-representing particles, while maintaining the stochastic approach and the representative particle method for the smaller bodies.

Our contributions are then summarised in Chapter 7, and the thesis is concluded with a brief outlook towards possible future work.

1.4 Published works

Part of this work has been submitted for publication separately:

Chapter 3 has been published as Beutel and Dullemond (2023) under the title 'An improved Representative Particle Monte Carlo method for the simulation of particle growth'.

Chapter 4 will be published as Beutel et al. (in press) under the title 'Efficient simulation of stochastic interactions among representative Monte Carlo particles'.

An abridged version of Chapter 5 has been published in the conference proceedings for the Conference of Next Generation Arithmetic (CoNGA) 2023 under the title 'A Paradigm for Interval-Aware Programming' (Beutel and Strzodka, 2023).

The introductory text of Chapter 3, Sects. 3.1, 3.2, 3.3.2, and 3.4.1 as well the initial draft for Fig. 3.1 were largely contributed by Cornelis P. Dullemond. Everything else was written by me.

Basic physics of protoplanetary disks 2

Rigid Body (sings).

Gin a body meet a body
Flyin′ through the air,
Gin a body hit a body,
Will it fly? and where?
Ilka impact has its measure,
Ne′er a ane hae I,
Yet a′ the lads they measure me,
Or, at least, they try.

Gin a body meet a body
Altogether free,
How they travel afterwards
We do not always see.
Ilka problem has its method
By analytics high;
For me, I ken na ane o′ them,
But what the waur am I?

James Clerk Maxwell, *In Memory of Edward Wilson, Who Repented of what was in his Mind to Write after Section*

A star forms by gravitational collapse of a dense region in a molecular cloud. Angular momentum conservation forces the collapsing cloud into the shape of a rotating disk as it accretes onto the star. Most material is accreted quickly, leaving a residual disk of gas and dust which is typically of much lower total mass than the star itself. The gas disk is relatively long-lived, taking on the order of 10^7 years to vanish through slow accretion and evaporation. It is commonly assumed that planets are composed of the dust and gas components of this protoplanetary disk.

In this chapter, we introduce some basic physical processes at work in protoplanetary disks. It goes without saying that the field of planet formation is vast and cannot possibly be covered in its entirety. Moreover, the narrow yet heterogeneous audience of this work may comprise not only astrophysicists but also mathematicians and computer engineers. Therefore, here we aim to explain just the bare necessities of the kinetic processes in planet formation in order to provide some physical context for the simulation techniques discussed in subsequent chapters.

2.1 Orbital motion

An undisturbed body or a viscosity-free parcel of gas on a perfectly circular orbit at orbital distance r around a central star with mass M_* moves at the Keplerian angular velocity

$$\Omega_{\mathrm{K}}(r) = \sqrt{\frac{GM_*}{r^3}} \,, \tag{2.1}$$

with G the gravitational constant, or at the tangential velocity

$$v_{\mathrm{K}} = r\,\Omega_{\mathrm{K}} = \sqrt{\frac{GM_*}{r}} \,. \tag{2.2}$$

To derive this result, we start with Newton's equation of motion,

$$\mathbf{F} = m\ddot{\mathbf{r}} \,, \tag{2.3}$$

where $\mathbf{F}$ is the vector of external forces and where the vector $\mathbf{r}$ represents the position of the body in heliocentric coordinates, and transform it to a locally Cartesian frame which co-rotates around the third axis with constant angular velocity Ω,

$$\mathbf{r} = R_3(\Omega t)\,(\mathbf{a} + \mathbf{x}) \,. \tag{2.4}$$

Here, $R_3(\cdot)$ is a matrix representing an active rotation around the third axis, the position $\mathbf{a} \equiv \hat{\mathbf{e}}_1 a$ represents, without loss of generality, the center of the rotating frame at time $t = 0$, and $\mathbf{x}$ is the frame-relative Cartesian coordinate. We obtain the transformed equation of motion in the co-rotating frame,

$$\mathbf{F} + \mathbf{F}_{\mathrm{cor}} + \mathbf{F}_{\mathrm{ctf}} = m\ddot{\mathbf{x}} \,, \tag{2.5}$$

with the fictitious forces

$$\mathbf{F}_{\mathrm{cor}} = -2m\boldsymbol{\Omega} \times \dot{\mathbf{x}} \qquad \text{(Coriolis force)} \tag{2.6}$$

$$\mathbf{F}_{\mathrm{ctf}} = -m\boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times (\mathbf{a} + \mathbf{x})) \qquad \text{(centrifugal force)} \,, \tag{2.7}$$

where we defined $\boldsymbol{\Omega} \equiv \hat{\mathbf{e}}_3 \Omega$.

As we want to study the motion of the frame itself, we set $\mathbf{x} = 0$ and also $\dot{\mathbf{x}} = 0$, which implies $\mathbf{F}_{\mathrm{cor}} = 0$. Because $\boldsymbol{\Omega} \perp \mathbf{a}$, Graßmann's identity can then be employed to simplify the centrifugal force to

$$\mathbf{F}_{\mathrm{ctf}} = m\Omega^2 \mathbf{a} \,. \tag{2.8}$$

With the tangential velocity of the frame

$$\mathbf{v}_\varphi = \frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \boldsymbol{\Omega} \times \mathbf{r} \,, \tag{2.9}$$

whose magnitude is also $v_\varphi = \Omega r$ because $\boldsymbol{\Omega} \perp \mathbf{v}_\varphi$, we can express Eq. (2.8) as

$$\mathbf{F}_{\mathrm{ctf}} = m\frac{v_\varphi^2}{r}\hat{\mathbf{e}}_1 \,. \tag{2.10}$$

By imposing steady-state conditions on the equation of motion (Eq. (2.5)), $\ddot{\mathbf{x}} \stackrel{!}{=} 0$, we thus obtain the *force balance equation*,

$$\mathbf{F} + \mathbf{F}_{\mathrm{ctf}} = 0 \tag{2.11}$$

which implies that

$$\mathbf{F} \cdot \hat{\mathbf{e}}_1 + m\frac{v_\varphi^2}{r} = 0 \ . \tag{2.12}$$

If the only external force is the gravitational force exerted by the central object,

$$\mathbf{F} = \mathbf{F}_{\mathrm{grav}} = -\frac{GM_* m}{r^2}\hat{\mathbf{e}}_1 \ , \tag{2.13}$$

then the force balance equation is solved by $\Omega = \Omega_{\mathrm{K}}$, or equivalently $v_\varphi = v_{\mathrm{K}}$, as given in Eqs. (2.1–2.2).

2.1.1 General orbital motion

A circular orbit is a special case of an elliptical orbit with eccentricity $e = 0$. Elliptical, parabolic, and hyperbolic trajectories are the general solutions of the equations of motion (Eq. (2.3)) if the gravitational force of the central object is the only force considered, a situation usually referred to as the *two-body problem*.

Orbital motion can be parameterised by the *semimajor axis* a, the *eccentricity* e, the *inclination* i, the *argument of the periapsis* ω, the *longitude of the ascending node* Ω, and the *true anomaly* ν. For a given set of parameters, the position of an object in orbital motion is given in heliocentric Cartesian coordinates as

$$\mathbf{r} = R_3(\Omega)R_1(i)R_3(\omega) \cdot r \cdot \begin{pmatrix}\cos\nu & \sin\nu & 0\end{pmatrix}^{\mathsf{T}} \tag{2.14}$$

where $R_j(\varphi)$ denotes an active rotation matrix rotating by an angle φ about the j-th axis, and where the orbital distance r is given by

$$r = \frac{a\left(1-e^2\right)}{1+e\cos\nu} \ . \tag{2.15}$$

Without external forces, an orbit is defined by the time-invariant quantities a, e, i, ω, and Ω, with only the true anomaly ν changing with time.

As a measure of the location of the object on its orbit, the anomaly can be defined in different ways. The true anomaly ν has a simple geometrical meaning: it is the angle between the periapsis and the current position of the object as measured from the central object situated in the main focus of the orbital ellipse. By measuring the angle from the geometric center of the ellipse rather than the main focus, one obtains the *eccentric anomaly* ϵ which can be related to true anomaly in different ways, for instance

$$\cos\nu = \frac{\cos\epsilon - e}{1 - e\cos\epsilon} \ , \tag{2.16}$$

or alternatively

$$\tan\frac{\nu}{2} = \sqrt{\frac{1-e}{1-e}}\tan\frac{\epsilon}{2}\ . \tag{2.17}$$

Then there is the *mean anomaly* M, the angle of an object on a fictitious circular orbit with radius a. Mean anomaly is convenient because it has a linear relation to time:

$$M = \frac{2\pi}{T}(t - t_0) \tag{2.18}$$

where t is the current time, t_0 is some reference time, and

$$T = \frac{2\pi}{\Omega_\mathrm{K}} \tag{2.19}$$

is the *orbital period* of the body. Eccentric anomaly ϵ and mean anomaly M are related by *Kepler's equation*,

$$M = \epsilon - e\sin\epsilon\ , \tag{2.20}$$

which can be solved for ϵ either numerically or with a series expansion. With these relations, the true anomaly ν can be related to time t, and thus the position of the object $\mathbf{r}$ at a given time t can be predicted.

2.1.2 Many-body kinetics

Beside the gravity of the central star, bodies are subject to a number of additional forces that can not be neglected in a realistic model. First, all massive bodies exert a gravitational force. Therefore, in an ensemble of N massive bodies, body j feels the external gravitational force

$$\mathbf{F}_{\mathrm{grav},j} = \sum_{k=1,k\neq j}^{N} G\,m_j m_k \frac{\mathbf{r}_k - \mathbf{r}_j}{\|\mathbf{r}_k - \mathbf{r}_j\|^2}\ , \tag{2.21}$$

where m_k and $\mathbf{r}_k$ are the mass and the position of object k. Furthermore, in the presence of gas, particles feel the velocity-dependent gas drag force $\mathbf{F}_\mathrm{drag}$. The gas-relative velocity is typically non-zero even for bodies on circular planar orbits because a viscous gas disk will rotate at sub-Keplerian velocity, as will be discussed in Sect. 2.2.4. Additionally, turbulent fluctuations in the gas will cause corresponding fluctuations of the gas drag force.

For a highly coupled system in which these additional forces are significant, particle trajectories cannot be predicted analytically or semi-analytically, and hence the parameterisation of the undisturbed trajectory in Eq. (2.14) is not very useful. Instead, the global set of the coupled equations of motion (Eq. (2.3)) must be solved for the entire particle ensemble. This is referred to as *N-body simulation*.

The gravitational forces between massive bodies are straightforward to compute; but to determine the gas drag force $\mathbf{F}_\mathrm{drag}$, we first need to develop a model of the gas disk.

2.2 Gas disk

From the masses and compositions of the planets in the solar system, Weidenschilling (1977b) inferred a 'minimum viable' gas surface density profile that scales with $r^{-3/2}$, where r is the distance from the Sun. Based on these observations, the surface density (that is, the vertically integrated volume density) of the gas in the *Minimum Mass Solar Nebula* (MMSN) model is often modelled as a power-law (Hayashi, 1981b),

$$\Sigma_{\mathrm{MMSN}}(r) = 1.7 \times 10^3 \left(\frac{r}{\mathrm{AU}}\right)^{-3/2} \mathrm{g\,cm^{-2}} \,, \tag{2.22}$$

while the relation of dust to gas is assumed to match the $\sim 1:100$ dust-to-gas ratio of the interstellar medium. (It should be emphasised that neither constant dust-to-gas ratio nor the power-law exponent of $-\frac{3}{2}$, which Weidenschilling (1977b) found to be appropriate for the suspected planet-forming region of the solar system, nor generally the power-law nature of the surface density profile are solid assumptions; the gas profile cannot be observed directly, so we have to make do with plausible models.) More generally, the gas surface density may be written

$$\Sigma(r) = \Sigma_0 \left(\frac{r}{r_0}\right)^p \,, \tag{2.23}$$

where r_0 is some reference distance and Σ_0 is a normalisation constant chosen such that the surface density integrated from the inner edge r_{min} to the outer edge r_{max} matches the assumed total disk mass,

$$2\pi \int_{r_{\mathrm{min}}}^{r_{\mathrm{max}}} \mathrm{d}r\, r\, \Sigma(r) \stackrel{!}{=} \mathcal{M}_{\mathrm{disk}} \,. \tag{2.24}$$

2.2.1 Vertical structure

With the simplifying assumption that the gas is vertically isothermal, we can use the vertical hydrostatic balance equation to infer a vertical gas density of Gaussian structure (Armitage, 2017, §II.B.1),

$$\rho_{\mathrm{gas}}(r,z) = \rho_{\mathrm{gas},0}(r) \exp\left[-\frac{z^2}{2h^2(r)}\right] \,, \tag{2.25}$$

where ρ_{gas} denotes the volume density of the gas, z is the vertical elevation,

$$\rho_{\mathrm{gas},0}(r) = \frac{1}{\sqrt{2\pi}} \frac{\Sigma(r)}{H_p(r)} \tag{2.26}$$

is the mid-plane gas density, and

$$H_p(r) = \frac{c_{\mathrm{s}}(r)}{\Omega_{\mathrm{K}}(r)} \tag{2.27}$$

is the *pressure scale height*. In a vertically isothermal disk, the speed of sound

$$c_{\mathrm{s}}(r) = \frac{k_{\mathrm{B}} T(r)}{\mu\, m_{\mathrm{p}}} \tag{2.28}$$

varies only with the distance from the star r. k_{B} is the Boltzmann constant, m_{p} is the mass of the proton, and the gas is assumed to be a H_2–He mixture of interstellar composition with an average molecular weight of $\mu = 2.3$.

2.2.2 Turbulent viscosity and radial infall

A non-viscous fluid in circumstellar motion moves on stable orbits; but in the presence of viscosity, a steady-state inward motion emerges. As noted by Shakura and Sunyaev (1973b), the effects of turbulence can technically be considered an effective viscosity. Their formulation of turbulence-driven effective viscosity is usually dubbed 'α-viscosity' after the dimensionless parameter α. In this model, the kinematic viscosity ν is given by

$$\nu = \alpha c_\mathrm{s} H_p. \tag{2.29}$$

The radial velocity of gas under steady-state conditions then is

$$v_{r,\mathrm{gas}} = -\frac{3}{2}\frac{\nu}{r}\ , \tag{2.30}$$

(Armitage, 2017, §IV.A.5), indicating an inward motion of the gas.

2.2.3 Temperature

Irradiation by the central star and viscous heating through accretion are the most significant contributions to the temperature of the gas. Armitage (2017, §IIB,C) refers to disks in which one of these contributions dominates as *passive* and *active circumstellar disks*, respectively, arguing that both contributions may play a role in planet formation contexts depending on time and orbital distance. For a simple model of a limited radial segment of an accretion disk – for instance, a gap-carving planet together with its associated dust trap – which does not claim accuracy as a global model, it may be reasonable to assume a power-law relation between temperature T and orbital distance r, just as had been done for the surface density in Eq. (2.23):

$$T(r) = T_0 \left(\frac{r}{r_0}\right)^q\ . \tag{2.31}$$

If we additionally impose that accretion operate as a steady-state process, we can link surface density and temperature. The gas accretion rate $\dot{M}$ is given as

$$\dot{M} = -2\pi r \Sigma v_{r,\mathrm{gas}}\ . \tag{2.32}$$

Following the line of reasoning given in Dullemond (2013), we impose steady-state conditions, $\dot{M} \stackrel{!}{=} \mathrm{const}$ and insert Eqs. (2.23) and (2.31) to obtain the relation

$$p + q = -\frac{3}{2}\ . \tag{2.33}$$

2.2.4 Orbital velocity

In addition to the gravitational force, a gas parcel of mass m is subject to the pressure-gradient force $\mathbf{F}_\mathrm{p}$:

$$\frac{\mathbf{F}_\mathrm{p}}{m} = -\frac{1}{\rho_\mathrm{gas}}\nabla p\ , \tag{2.34}$$

where p denotes the pressure. As a consequence, the force balance plays out differently for a gas parcel than for a solid particle. By solving the force balance equation (Eq. (2.11)) for a gas parcel assuming $\mathbf{F} = \mathbf{F}_{\mathrm{grav}} + \mathbf{F}_{\mathrm{p}}$, we obtain the tangential gas velocity

$$\begin{aligned} v_{\varphi,\mathrm{gas}} &= v_{\mathrm{K}} \sqrt{1 + \frac{c_{\mathrm{s}}^2}{v_{\mathrm{K}}^2} \frac{\mathrm{d}\log p}{\mathrm{d}\log r}} \\ &\approx v_{\mathrm{K}} \left(1 + \frac{1}{2} \frac{c_{\mathrm{s}}^2}{v_{\mathrm{K}}^2} \frac{\mathrm{d}\log p}{\mathrm{d}\log r}\right) \end{aligned} \tag{2.35}$$

where pressure and density are related by the speed of sound c_{s},

$$p = \rho_{\mathrm{gas}} c_{\mathrm{s}}^2 \ , \tag{2.36}$$

and where it was assumed that $c_{\mathrm{s}}^2 / v_{\mathrm{K}}^2 \ll 1$. With the *nebula pressure parameter* η (Adachi et al., 1976; Inaba et al., 2001),

$$\eta \coloneqq \frac{v_{\mathrm{K}} - v_{\varphi,\mathrm{gas}}}{v_{\mathrm{K}}} \tag{2.37}$$

$$\approx -\frac{1}{2} \frac{c_{\mathrm{s}}^2}{v_{\mathrm{K}}^2} \frac{\mathrm{d}\log p}{\mathrm{d}\log r} \ , \tag{2.38}$$

we can conveniently express $v_{\varphi,\mathrm{gas}}$ as

$$v_{\varphi,\mathrm{gas}} \approx v_{\mathrm{K}} \left(1 - \eta\right) \ . \tag{2.39}$$

In a smooth gas disk, the pressure gradient $\mathrm{d}p/\mathrm{d}r$ is negative (a typical value might be $\mathrm{d}\log p/\mathrm{d}\log r = -2$), and hence η is positive. As is evident from Eq. (2.39), the gas then moves at sub-Keplerian velocity.

2.2.5 Planetary gaps

It can be seen in the results of the DSHARP survey (Andrews et al., 2018) that the dust profile of protoplanetary disks is rarely observed to obey a simple power-law relation; instead, substructures such as rings and gaps abound. If the observed cavities are linked to the presence of gap-carving planets (Dullemond et al., 2018), a power-law relation may still be reasonable as the envelope of the dust and gas profiles, and according models for planetary gaps 'carved' by embedded planets have been proposed by Kanagawa et al. (2015, 2017, 2020). Here we describe the simplified Gaussian model of Eriksson et al. (2021) based on Kanagawa et al. (2015), noting however that, according to Voelkel (2022), a gap model relying solely on Kanagawa's gap depth parameter K is oversimplified and inaccurate, and much better agreement with hydrodynamic simulations of gap opening could be demonstrated with the refined but more complicated model of (Kanagawa et al., 2017).

Given a gap-carving planet of mass M_{pl} at orbital distance a_{pl}, we first define the Kanagawa parameter K:

$$K = \left(\frac{M_{\mathrm{pl}}}{M_*}\right)^2 \left(\frac{H_p(a_{\mathrm{pl}})}{a_{\mathrm{pl}}}\right) \alpha^{-1} \ . \tag{2.40}$$

Using the depth of the gap at the orbital location of the planet given by

$$\frac{\Sigma_{\min}}{\Sigma(a_{\rm pl})} = \frac{1}{1+0.04K} \ , \tag{2.41}$$

we define the Gaussian gap profile as

$$G(r) = \frac{\Sigma_{\min}}{\Sigma(a_{\rm pl})} \exp\left[-\frac{(r-a_{\rm pl})^2}{2H_p^2}\right] \ . \tag{2.42}$$

The surface density is then modified to include the planetary gap at location $a_{\rm pl}$:

$$\Sigma(r) \rightarrow \Sigma(r) \cdot (1 - G(r))^{-1} \ . \tag{2.43}$$

2.2.6 Gas drag regimes

In the coordinates of a frame co-rotating with Kepler velocity $\Omega_{\rm K}$, the gas velocity $\mathbf{v}_{\rm gas}$ at $\mathbf{x} = 0$ is

$$\begin{aligned} v_{x,\rm gas} &= v_{r,\rm gas} = -\frac{3}{2}\frac{\nu}{r} \\ v_{y,\rm gas} &= v_{\varphi,\rm gas} - v_{\rm K} = -\eta v_{\rm K} \ . \end{aligned} \tag{2.44}$$

Gas acts upon solid particles by means of a drag force $\mathbf{F}_{\rm drag}$. The force exerted depends on the interaction regime, which in turn depends on particle and gas properties as well as on the relative gas velocity

$$\delta\mathbf{v} := \mathbf{v}_{\rm gas} - \mathbf{v} \ , \tag{2.45}$$

and acts opposite the direction of the particle motion relative to the gas,

$$\mathbf{F}_{\rm drag} = F_{\rm drag} \frac{\delta\mathbf{v}}{\|\delta\mathbf{v}\|} \ . \tag{2.46}$$

To describe the kinetics of a dust particle in non-eccentric planar orbital motion immersed in a gas disk, we therefore augment the external forces with the gas drag force, $\mathbf{F} = \mathbf{F}_{\rm grav} + \mathbf{F}_{\rm drag}$.

The drag timescale can be interpreted as a *stopping time*

$$t_{\rm s} = \frac{m\,\delta v}{F_{\rm drag}} \ , \tag{2.47}$$

where the magnitude of the drag force $F_{\rm drag}$ has a regime-dependent definition. Four different regimes are considered: the Epstein regime (Epstein, 1924), in which the mean free path of gas molecules is larger than the particle radius, and three different Stokes regimes depending on the Reynolds number. The following treatment is adapted from Weidenschilling (1977a) and Birnstiel et al. (2010).

The number density of gas molecules at the mid-plane is

$$n_{\rm gas} = \frac{N_{\rm gas}}{V} = \frac{p}{k_{\rm B}T} = \frac{\rho_{\rm gas}}{\mu m_{\rm p}} \tag{2.48}$$

where $N_{\rm gas}$ indicates the number of gas molecules around the mid-plane in some volume V, T is the gas temperature, μ is the average molecular weight of the gas, and $m_{\rm p}$ is the proton mass. Above relation follows from the ideal gas law $pV = Nk_{\rm B}T$ and from Eq. (2.28). The mean free path of a gas molecule can then be estimated as

$$\lambda_{\rm mfp} = (n_{\rm gas}\sigma_{\rm H_2})^{-1} \tag{2.49}$$

with the cross-section of H_2 molecules $\sigma_{\rm H_2} = 2 \times 10^{-15}{\rm cm}^2$.

The gas–dust interaction operates in the Epstein regime if $\lambda_{\rm mfp}/R \gtrsim \frac{4}{9}$, where R is the bulk radius of the dust particle. In this case, the drag force is given as

$$F_{\rm drag} = \frac{4}{3}\pi\rho_{\rm gas}R^2\bar{u}_{\rm gas}\delta v \tag{2.50}$$

with $\bar{u}_{\rm gas} = c_{\rm s}\sqrt{8/\pi}$ the mean thermal velocity of the gas molecules.[1]

Outside the Epstein regime, we need to consider the Reynolds number, a dimensionless quantity that amounts to the ratio of inertial and viscous forces:

$${\rm Re} = 2\frac{R\delta v}{\nu_{\rm mol}} \tag{2.51}$$

where the gas molecular viscosity is

$$\nu_{\rm mol} = \frac{1}{2}\bar{u}_{\rm gas}\lambda_{\rm mfp}\ . \tag{2.52}$$

The drag force can then be expressed as

$$F_{\rm drag} = \frac{\pi}{2}R^2\rho_{\rm gas}C_{\rm D}\delta v^2 \tag{2.53}$$

with a drag coefficient $C_{\rm D}$ depending on the Reynolds number:

$$\begin{aligned} C_{\rm D} &= 24{\rm Re}^{-1} && \text{for } {\rm Re} < 1\ , \\ C_{\rm D} &= 24{\rm Re}^{-0.6} && \text{for } 1 < {\rm Re} < 800\ , \\ C_{\rm D} &= 0.44 && \text{for } 800 < {\rm Re}\ . \end{aligned} \tag{2.54}$$

Equivalently, one can provide regime-dependent definitions of the stopping time $t_{\rm s}$, as is done, for instance, in Birnstiel et al. (2010), Eq. (10),[2] and then define the drag force through the general relation

$$\mathbf{F}_{\rm drag} = m\frac{\delta\mathbf{v}}{t_{\rm s}}\ . \tag{2.55}$$

[1] In Birnstiel et al. (2010) after Eq. (10), the mean thermal velocity is erroneously stated as $\bar{u} = c_{\rm s}\sqrt{\pi/8}$.

[2] The expression for the stopping time for $1 < {\rm Re} < 800$ (Stokes regime 2) in Eq. (10) of Birnstiel et al. (2010) contains an error: the denominator should read $9\nu_{\rm mol}^{0.6}\rho_{\rm g}u^{0.4}$ but reads $9\nu_{\rm mol}^{0.6}\rho_{\rm g}^{1.4}u^{0.4}$. The error can be traced back to Whipple (1973) and was replicated in Weidenschilling (1977a) and many subsequent publications.

The *Stokes number* St is a dimensionless number that characterises the interaction of gas and particles. It is defined as

$$\mathrm{St} = \frac{t_\mathrm{s}}{t_\mathrm{eddy}} \tag{2.56}$$

where t_eddy is the eddy turn-over time, which we take to be equal to the Kepler time $t_\mathrm{K} = \Omega_\mathrm{K}^{-1}$ following Birnstiel et al. (2010). The Stokes number therefore relates the timescale of orbital revolution to the stopping time of the particle. The smaller the value of St, the quicker a particle 'forgets' its specific orbital state as it is dragged along by the surrounding gas parcel which flows on a near-circular trajectory. On timescales greater than t_s, the orbital elements e and i thus have no secular significance as particle trajectories are damped towards circularity, with deviations dominated by turbulent stirring.

2.2.7 Damping and radial drift

Because gas orbits the star at sub-Keplerian speed, $v_{y,\mathrm{gas}} = -\eta v_\mathrm{K}$ (Eq. (2.44)), a body moving on a Keplerian orbit will be continuously exposed to a drag force, through which potential energy is subtracted from the body's orbit, forcing it to spiral inwards.

We follow Weidenschilling (1977a) in determining the steady-state drift velocity relative to a local coordinate system co-rotating at Kepler velocity. The co-rotating coordinate system introduced above is a Cartesian tangential frame; hence in a frame co-rotating with Kepler velocity at radial position a, a particle with frame-relative position $\mathbf{x} = \hat{e}_1(r-a)$ and velocity $\mathbf{v}$ will move at tangential velocity $v_\varphi = r\Omega_\mathrm{K}(a)$ in heliocentric coordinates. This velocity is notably different from the Kepler tangential velocity $r\Omega_\mathrm{K}(r)$ of a particle at orbital radius $r \neq a$. Thus, before imposing equilibrium conditions, we first transform to a sheared coordinate system

$$\begin{aligned}
&\mathbf{x} \to \mathbf{x} \\
&\mathbf{v} \to \mathbf{u} \coloneqq \mathbf{v} + \hat{e}_2\, r\left(\Omega_\mathrm{K}(a) - \Omega_\mathrm{K}(r)\right) \\
&\dot{\mathbf{v}} \to \dot{\mathbf{u}} = \dot{\mathbf{v}} + \hat{e}_2\, \dot{r}\left(\Omega_\mathrm{K}(a) + \frac{1}{2}\Omega_\mathrm{K}(r)\right)
\end{aligned} \tag{2.57}$$

such that, at any relative coordinate $\mathbf{x}$, a velocity $\mathbf{u} = 0$ corresponds to a velocity $v_\varphi = \Omega_\mathrm{K}(r)$:

$$\frac{\mathbf{F}_\mathrm{grav}}{m} + \frac{\mathbf{F}_\mathrm{drag}}{m} + \frac{\mathbf{F}_\mathrm{cor}}{m} + \frac{\mathbf{F}_\mathrm{ctf}}{m} = \dot{\mathbf{u}} - \hat{e}_2\, \dot{r}\left(\Omega_\mathrm{K}(a) + \frac{1}{2}\Omega_\mathrm{K}(r)\right) \ . \tag{2.58}$$

By evaluating at $\mathbf{x} = 0$, where $\dot{\mathbf{r}} = \mathbf{v} = \mathbf{u}$ and $\mathbf{F}_\mathrm{ctf} + \mathbf{F}_\mathrm{grav} = 0$, and imposing equilibrium conditions $\dot{\mathbf{u}} \overset{!}{=} 0$, we obtain

$$\frac{\mathbf{F}_\mathrm{drag}}{m} = -\frac{3}{2}\hat{e}_2 \dot{x}\Omega_\mathrm{K}(a) + 2\hat{e}_3 \times \mathbf{u} \ . \tag{2.59}$$

Using the Stokes number St (Eq. (2.56)), we can write (2.59) as a simple 2-dimensional system of equations,

$$\mathbf{u} = \mathbf{v}_\mathrm{gas} + \begin{pmatrix} 0 & 2\,\mathrm{St} \\ -\frac{1}{2}\,\mathrm{St} & 0 \end{pmatrix} \mathbf{u} \ . \tag{2.60}$$

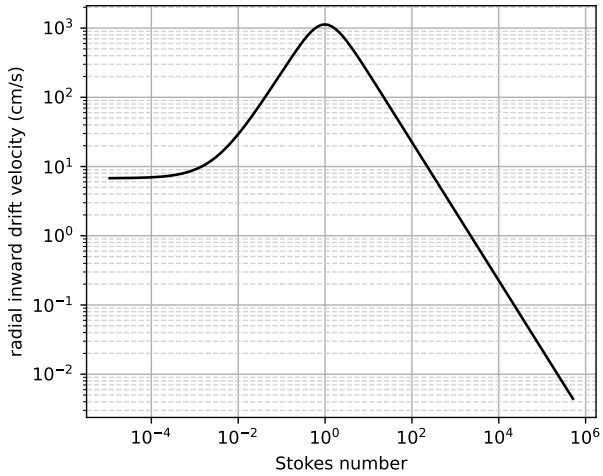


Figure 2.1: The radial equilibrium drift velocity of massive body on non-eccentric orbit as a function of Stokes number St as per Eq. (2.61).
The drift speed converges to 0 for $\mathrm{St} \to \infty$ and settles to the inwards gas drift speed $|v_{r,\mathrm{gas}}|$ for gas-bound particles with $\mathrm{St} \to 0$. At typical planet-forming radii of several AU under typical conditions in protoplanetary disks, the radial drift maximum for $\mathrm{St} = 1$ amounts to efficient removal of pebbles (that is, particles of sizes between few cm and about $1\,\mathrm{m}$). This is known as the *metre-size barrier* or the *drift barrier* to growth.

If the Stokes number is independent of the drift velocity, as is the case in the Epstein regime, this system of equations has the solution

$$\mathbf{u} = \frac{1}{1+\mathrm{St}^2} \begin{pmatrix} 1 & 2\,\mathrm{St} \\ -\frac{1}{2}\,\mathrm{St} & 1 \end{pmatrix} \mathbf{v}_{\mathrm{gas}} \ , \tag{2.61}$$

and thus

$$u_x = \frac{1}{1+\mathrm{St}^2} \left(v_{x,\mathrm{gas}} + 2\,\mathrm{St}\, v_{y,\mathrm{gas}} \right) \ . \tag{2.62}$$

The dependency of u_x on St is visualised in Fig. 2.1. As is evident from Eq. (2.62), particles are bound to the gas in the $\mathrm{St} \to 0$ limit, $u_x \to v_{x,\mathrm{gas}}$. By equating the two terms in Eq. (2.62), $v_{x,\mathrm{gas}} \stackrel{!}{=} 2\,\mathrm{St}\, v_{y,\mathrm{gas}}$, and by imposing a turbulent viscosity model as per Eq. (2.30), we find that the two terms are of equal magnitude for $\mathrm{St} \sim \alpha$, implying that aerodynamics can be neglected for dust particles with $\mathrm{St} \lesssim \alpha$, instead assuming that such particles simply follow the gas.

So far we have assumed that the unperturbed trajectory is non-eccentric, $e = 0$, and planar, $i = 0$. Determining the radial equilibrium drift for eccentric or inclined trajectories is more involved. For planetesimals – that is, for bodies with weak coupling to the gas – this has been investigated analytically by Adachi et al. (1976) and refined and verified numerically by Inaba et al. (2001), who give the following formulae for the

steady-state change rate of semimajor axis a, eccentricity e, and inclination i:

$$\frac{\tau}{a}\frac{\mathrm{d}a}{\mathrm{d}t} = -2\eta\sqrt{\left(\frac{2\,(2E+K)}{3\pi}e\right)^2 + \left(\frac{2}{\pi}\sin i\right)^2 + \eta^2}\,, \tag{2.63}$$

$$\frac{\tau}{e}\frac{\mathrm{d}e}{\mathrm{d}t} = -\sqrt{\left(\frac{2E}{\pi}e\right)^2 + \left(\frac{2}{\pi}\sin i\right)^2 + \left(\frac{3}{2}\eta\right)^2}\,, \tag{2.64}$$

$$\frac{\tau}{\sin i}\frac{\mathrm{d}\sin i}{\mathrm{d}t} = -\frac{1}{2}\sqrt{\left(\frac{2E}{\pi}e\right)^2 + \left(\frac{8}{3\pi}\sin i\right)^2 + \eta^2}\,, \tag{2.65}$$

where E and K, not to be confused with the Kanagawa parameter K defined in Eq. (2.40), denote the elliptic integrals $E \equiv E(3/4)$ and $K \equiv K(3/4)$, and where the characteristic time for the drag dissipation τ used in Inaba et al. (2001), Eqs. (24–26) is given by $\tau = (\mathrm{St}/\Omega_\mathrm{K})\,\eta$.

For non-eccentric planar trajectories, we find Eq. (2.63) to be approximately equivalent to the result obtained in Eq. (2.61). To verify this, we start from Eq. (2.61), approximate $\mathrm{St}/(1+\mathrm{St}^2) \approx \mathrm{St}^{-1}$ for $\mathrm{St} \gtrsim 1$ and neglect the $(1+\mathrm{St}^2)^{-1} v_{x,\mathrm{gas}}$ term, finding a result identical to the prescription of Eq. (2.63) with $e = 0$, $i = 0$ imposed.

When considering a group of similar particles, eccentricities and inclinations can be assumed to follow a Rayleigh-type distribution (Ida and Makino, 1992b). Inaba et al. (2001) state the following equations for the evolution of the dispersions $\langle e^2\rangle$ and $\langle \sin^2 i\rangle$:

$$\frac{\tau}{\langle e^2\rangle}\frac{\mathrm{d}\langle e^2\rangle}{\mathrm{d}t} = -2\sqrt{\frac{9}{4\pi}E^2\langle e^2\rangle + \frac{1}{\pi}\langle \sin^2 i\rangle + \frac{9}{4}\eta^2}\,, \tag{2.66}$$

$$\frac{\tau}{\langle \sin^2 i\rangle}\frac{\mathrm{d}\langle \sin^2 i\rangle}{\mathrm{d}t} = -\sqrt{\frac{1}{\pi}E^2\langle e^2\rangle + \frac{4}{\pi}\langle \sin^2 i\rangle + \eta^2}\,, \tag{2.67}$$

2.2.8 Turbulent stirring

So far we have only considered aerodynamic interactions between gas and dust. However, given that the gas surface density is initially much higher than the surface density of dust, one might expect that the gravitational dust–gas interaction is significant as well. The gravitational interaction between planets and the gas disk must be modelled with a multidimensional hydrodynamic simulation to correctly describe the emergence of non-axisymmetric features such as spiral waves excited by the planet; but for many purposes, the resultant gas density profile can be represented with a simplified 1-dimensional semi-analytical model as described in Sect. 2.2.5. The gravitational impact of a homogeneous gas disk is not very interesting; however, if the gas density is subject to turbulent fluctuations, as may be excited by the magnetorotational instability (MRI; cf. Balbus and Hawley, 1991; Laughlin et al., 2004), the density fluctuations will have a noticeable effect on massive bodies. Such fluctuations can be shown to boost the eccentricity of solid bodies (Ogihara et al., 2007; Ida et al., 2008),

$$e \propto \gamma\Sigma\, r^2\, (\Omega_\mathrm{K} t)^{1/2}\,, \tag{2.68}$$

where γ is a dimensionless number representing the strength of turbulent density fluctuations induced by the MRI.Consequentially, the rms eccentricity is stirred by turbulent

fluctuations,

$$\frac{\mathrm{d}\left\langle e^2\right\rangle}{\mathrm{d}t} \propto \gamma^2 \left(\Sigma\, r^2\right)^2 \Omega_\mathrm{K} \ . \tag{2.69}$$

The characteristic length scale of turbulent fluctuations is much greater than the size of any solid body embedded in the gas disk; therefore, the turbulent stirring of a body is independent of its mass and eccentricity. Turbulent stirring can thus significantly contribute to the dynamical heating of the system at all mass scales.

2.3 Surrogate models

Because of the sum in Eq. (2.21), an N-body simulation of an ensemble of N particles entails a computational complexity of order $\mathcal{O}(N^2)$. In presence of heavier bodies such as planets or planetary embryos whose influence by far dominates the gravitational dynamics, the gravitational impact of small particles becomes negligible beyond a certain distance, which would allow the construction of a scheme with complexity $\mathcal{O}(N \log N)$; but even so, N can be enormously large, for instance, $N \sim 10^{20}$, and thus even complexity $\mathcal{O}(N)$ would be unaffordable.

2.3.1 Continuum models

Physical systems comprising extremely large numbers of particles are often treated as continuous systems at sufficiently large scales. For the case of coagulation processes, a continuum view has first been proposed by Smoluchowski (1916), who introduced what is now known as the *Smoluchowski coagulation equation* for a continuous particle mass distribution function $f(m,t)$ which quantifies how the number of particles $f(m,t)\mathrm{d}m$ changes in an infinitesimal mass bin $m \in [m, m+\mathrm{d}m)$:

$$\begin{aligned}\frac{\partial}{\partial t} f(m,t) = &-f(m,t) \int_0^\infty \mathrm{d}m' \lambda(m,m') f(m',t) \\ &+ \frac{1}{2} \int_0^m \mathrm{d}m' \lambda(m', m-m') f(m',t) f(m-m',t) \ .\end{aligned} \tag{2.70}$$

$\lambda(m,m')$, also named the *coagulation kernel,* quantifies the coagulation rate of two bodies of masses m and m'. The meaning of the two terms can then be understood intuitively: for each infinitesimal mass bin $[m, m+\mathrm{d}m)$, the number of particles in the bin that grow to a mass $\tilde{m} \geqslant m+\mathrm{d}m$ must be subtracted from $f(m,t)\mathrm{d}m$, while the number of particles from other bins which grow to a mass $\tilde{m} \in [m, m+\mathrm{d}m)$ must be added to $f(m,t)\mathrm{d}m$.

A continuous model of a discrete system has to rely on certain assumptions in order to act as an adequate representation of the system. Usually, a hierarchy of scales is established in which the effects studied by means of continuous equations operate on scales strictly larger than the scale of quantisation where the discrete nature of the system becomes relevant. In the case of the Smoluchowski coagulation equation, one might first quantify the scale of the desired resolution in the mass distribution function $f(m,t)$ by a logarithmic mass difference $\Delta(\log m) \equiv \log q$ corresponding to a ratio of

masses $q > 1$, and then demand that, for a continuum approach to be justified, the number of particles in any given range $[m, q \cdot m)$ shall always be much greater than unity. If this requirement cannot be guaranteed, the continuum method may break down, as can be illustrated with a simple coagulation kernel for which Eq. (2.70) can be solved analytically. Several analytical solutions are discussed in Appendix 3.A; for simplicity let us consider the constant coagulation kernel

$$\lambda(m, m') = \Lambda \stackrel{!}{=} \text{const} \ , \tag{2.71}$$

for which an analytical solution is given by Eq. (3.99). Assuming $m/m^0 \gg 1$, $g \ll 1$, and $\Lambda t \gg 1$, we can approximate and simplify the solution,

$$m^2 f(m,t) \approx N^0 \left(\frac{2}{M\Lambda}\right)^2 \left(\frac{m}{t}\right)^2 \exp\left[-\frac{2}{m^0 M\Lambda}\frac{m}{t}\right] \ . \tag{2.72}$$

We find that the mass-weighted number distribution $m^2 f(m,t)$ is approximately self-similar, that is, the shape of $m^2 f(m,t)$ is near-invariant under a transformation of coordinates $m \to \alpha m$, $t \to \alpha t$, which implies that, for $t \to \infty$, the peak of the mass distribution will also grow to higher masses beyond all bounds. The self-similarity of the mass-weighted number distribution can also be observed in Figure 3.2. However, this cannot be correct for a coagulating system of discrete particles. Assuming the same constant coagulation kernel (Eq. (2.71)), the entire mass M will eventually converge into a single particle whose mass serves as a natural upper bound for the particle number distribution. The true convergence behaviour of such a discrete system of particles is reproduced in Fig. 3.3, ending up visibly different from the boundlessly self-similar analytic solution.

This specific problem may be avoided by representing the most massive bodies individually (cf. e.g. Weidenschilling, 1997). But there are other downsides to the continuum model approach. Because an analytical solution is not available in even vaguely realistic models, the Smoluchowski equation is usually solved numerically on a grid of non-infinitesimal mass bins $[m, m + \delta m)$, a technique used to model dust growth in protoplanetary disks at least since Weidenschilling (1980) (see also Wetherill and Stewart, 1989; Dullemond and Dominik, 2005; Birnstiel et al., 2010). However, grid-based methods are subject to the *curse of dimensionality*; adding a dimension to the property space and resolving it with $\mathcal{N}$ grid cells multiplies the original number of grid cells in the simulation by a factor $\mathcal{N}$; furthermore, because all grid cells can potentially interact, the number of operations is multiplied by a factor $\mathcal{N}^2$. To evade the curse of dimensionality, dynamic-average approximations have been used (e.g. Okuzumi et al., 2009). With such an approximation, only the per-bin averages of any additional properties are stored. This way, adding new property dimensions such as rms velocity or porous volume does not significantly add to the computational demands of the simulation; however, the number distribution is not resolved in the dimensions added to the parameter space, which may limit the usefulness of the method.

2.3.2 Representative models

The preceding section discussed the main drawbacks of the continuum model approach: the continuum approximation breaks down as particle numbers become small; and a

grid-based numerical method, as is usually employed to simulate the evolution of a continuous system, cannot resolve a high-dimensional property space without disproportionate computational effort. To avoid these problems, some authors have instead opted for a particle-based model (cf. e.g. Ormel et al., 2007; Zsom and Dullemond, 2008). The two types of models can be likened to the Eulerian and the Lagrangian picture of fluid dynamics. The Smoluchowski equation describes the flux of particles to and from a given infinitesimal bin of masses. Conversely, a particle-based model follows the mass trajectory of individual particles. Let the entirety of particles be indexed by numbers $1, \ldots, N$. Then, for a given particle j, the expected growth of mass is given by the equation

$$\frac{\mathrm{d}\langle m_j \rangle}{\mathrm{d}t} = \sum_{k=1}^{N} m_k \lambda(m_j, m_k) \ . \tag{2.73}$$

For a formal derivation, see Eqs. (3.113–3.117) in Appendix 3.B.

Often, a simulation cannot afford to keep track of every individual particle in the system. Instead, only a representative subset of n particles, which shall be indexed with $1, \ldots, n$, is followed. Representative approaches build upon the observation that, if there are many particles of a certain flavour (say, cm-sized pebbles), knowing all their individual properties (say, the precise masses of the individual pebbles) is not only computationally unviable but also not very interesting; instead, we would like to know their statistical properties (such as the average pebble mass, or the approximate mass distribution function sampled by the individual pebbles). A representative particle method will thus choose a representative subset of all particles in the system and then simulate only the interactions they partake in. The number distribution of all particles in the system is then extrapolated from the ensemble of representative particles. Without going into detail, here we only note that, for a system of n representative particles, Eq. (2.73) can be generalised to

$$\frac{\mathrm{d}\langle m_j \rangle}{\mathrm{d}t} = \sum_{k=1}^{n} \mathcal{K}_{jk}^{\mathrm{coll}} \ , \tag{2.74}$$

where $\mathcal{K}_{jk}^{\mathrm{coll}}$ denotes the mass change rate of representative particle j due to coagulation with particles represented by representative particle k.

Although the coagulation process may be treated as a continuous evolution of particle masses through solving Eq. (2.74) as a system of differential equations, growth by coagulation is in fact a discrete process composed of discrete coagulation events. Thus, if representative particle k represents a total number of N_k physical particles (that is, one representative particle and $(N_k - 1)$ non-representative particles), the mass change rate may be naturally decomposed as

$$\mathcal{K}_{jk}^{\mathrm{coll}} = (N_k - \delta_{jk})\, \lambda_{jk}^{\mathrm{coll}}\, (\Delta m)_{jk} \ . \tag{2.75}$$

Here, δ_{jk} is the Kronecker delta used to suppress self-coagulation of the representative particle in the case where $j = k$. $\lambda_{jk}^{\mathrm{coll}}$ is the particle collision rate for a pair of particles represented by particles j and k, respectively, and $(\Delta m)_{jk}$ is the expected change

of mass resulting from a coagulation of the given pair of particles. If all particles represented by representative particle k can be assumed to have the same mass m_k, then $(\Delta m)_{jk} = m_k$, but we note that Eqs. (2.74) and (2.75) are general enough to also allow for a more advanced concept of representation or for more nuanced collision outcomes such as bouncing or partial fragmentation.

In this view, it seems natural to simulate particle growth as a stochastic process. Instead of computing the evolution of the expected masses by solving a system of coupled differential equations continuously, we can thus employ a Monte Carlo method and simulate a possible realisation of the discrete growth process by sampling individual collision events. One example of a stochastic representative method will be elaborated in Sect. 3.3, where the Representative Particle Monte Carlo (RPMC) method originally devised by Zsom and Dullemond (2008) is defined formally. Treating coagulation as a stochastic process will result in higher noise compared to a continuous treatment, but it can be argued that a sampling model is more realistic. In a representative Monte Carlo simulation of a coagulation process, a possible trajectory through state space will be realised, whereas a continuous method must operate with expectation values and will therefore only yield expectation values for the individual particle masses, thus implying certain simplifying assumptions about the shape of the mass distribution such as unimodality. Conversely, a Monte Carlo method is agnostic of the distribution properties, and any distribution may emerge from it.

2.3.3 Dynamical heating

Previous works (e.g. Ida, 1990; Ida and Makino, 1993; Stewart and Ida, 2000) studied the nature of the emerging distributions of orbital eccentricity and inclination of planetesimals and proposed surrogate models that describe the evolution of the distribution average by a set of differential equations. First, the particle velocity $\mathbf{v}$ is decomposed as

$$\mathbf{v} = \mathbf{v}_\mathrm{K} + \Delta\mathbf{v} \tag{2.76}$$

where $v_\mathrm{K} = a\Omega_\mathrm{K}$ is the velocity of a body on a perfectly circular Kepler orbit with the same semimajor axis and $\Delta\mathbf{v}$ is the non-circular and non-planar deviation. This is sometimes referred to as the *epicycle approximation*. The magnitudes of the planar and vertical components are then related to the dispersions of eccentricity and inclination[3]:

$$\left\langle \Delta v_\mathrm{planar}^2 \right\rangle = v_\mathrm{K}^2 \left\langle e^2 \right\rangle \tag{2.77}$$

$$\left\langle \Delta v_\mathrm{vert}^2 \right\rangle = v_\mathrm{K}^2 \left\langle \sin^2 i \right\rangle \ . \tag{2.78}$$

For the mean square of the velocity deviation, this implies

$$\left\langle \Delta\mathbf{v}^2 \right\rangle = v_\mathrm{K}^2 \left(\left\langle e^2 \right\rangle + \left\langle \sin^2 i \right\rangle \right) \ . \tag{2.79}$$

We now can view the evolution of the rms eccentricity $\sqrt{\langle e^2 \rangle}$ and the rms inclination $\sqrt{\langle \sin^2 i \rangle}$ as a dynamical heating process. This idea can be traced back to Safronov (1969) and was elaborated in, for instance, Goldreich et al. (2004) and Ormel et al. (2010); a

[3] Most authors use i instead of $\sin i$, implicitly invoking the small-angle approximation.

detailed summary of the underlying reasoning is given in Ormel et al. (2010, §2.4 and Appendices B.2.1 and B.2.3).

Dynamical heating is usually modelled as two distinct processes. First, there is *viscous stirring*, which refers to the conversion between potential and kinetic (quasi-thermal) energy. Through a gravitational exchange between two circumstellar bodies, energy is extracted from or added to their Keplerian potentials, and added to or removed from their kinetic energy captured in $\langle \Delta \mathbf{v}^2 \rangle$, as in the gravitational slingshot manoeuvre of a space probe. Additionally, kinetic energy can be exchanged between the two interacting bodies, which is referred to as *dynamical friction*. As pointed out by Ormel et al. (2010, §2.4), '[the] distinction between the dynamical friction and viscous stirring interactions should not be interpreted as meaning that these belong to two distinct encounters. In contrast, a (single) encounter will both contribute to the friction as well as the stirring. We simply dissect collisionless encounters into a part that preserves the random energy (dynamical friction) and a part that does not (viscous stirring) (Ida, 1990)'. Ida and Makino (1993) and Goldreich et al. (2004) used this approach to study the dynamics of a two-component system, that is, an ensemble comprising two kinds of particles with masses m and $M \gg m$. Ormel et al. (2010) subsequently developed a generalisation of the method, applied it to an arbitrary number of particle species, and assembled a hybrid statistical model that can simulate collision and stirring processes in an arbitrary distribution of particle masses. The evolution of the mean squared values of eccentricity and inclination and of the average mass of a given representative particle with indices $j \in \{1, \ldots, n\}$ is then governed by a set of very general differential equations,

$$\frac{\mathrm{d}}{\mathrm{d}t} \left\langle e_j^2 \right\rangle = \sum_{k=1}^{n} \left(\mathcal{P}_{jk}^{\mathrm{vs}} + \mathcal{P}_{jk}^{\mathrm{df}} + \mathcal{P}_{jk}^{\mathrm{coll}} \right) \,, \tag{2.80}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \left\langle \sin^2 i_j \right\rangle = \sum_{k=1}^{n} \left(\mathcal{Q}_{jk}^{\mathrm{vs}} + \mathcal{Q}_{jk}^{\mathrm{df}} + \mathcal{Q}_{jk}^{\mathrm{coll}} \right) \,, \tag{2.81}$$

where $\mathcal{P}_{jk}^{\mathrm{vs}}$ and $\mathcal{Q}_{jk}^{\mathrm{vs}}$ denote the change rate of the squared rms eccentricity and squared rms inclination, respectively, of a representative particle j due to viscous stirring by all particles represented by representative particle k, and likewise, mutatis mutandis, for dynamical friction and for collisions.

The statistical evolution equations in Eqs. (2.80–2.81) allow for a variety of solution methods. They may be applied to a continuum model that keeps dynamic-average values of the rms eccentricities and rms inclinations for every mass bin; or they can be solved as differential equations alongside a Monte Carlo simulation of a mass coagulation process, as is done in Ormel et al. (2010). Furthermore, in a particle-based simulation, the equations can also be treated stochastically, and individual gravitational encounters can be simulated as a Monte Carlo process. Thus, in analogy with Eq. (2.75), the change rate contributions may be decomposed as

$$\mathcal{P}_{jk}^{\mathrm{int}} = (N_k - \delta_{jk})\, \lambda_{jk}^{\mathrm{int}} \left(\Delta e^2 \right)_{jk}^{\mathrm{int}} \,, \tag{2.82}$$

$$\mathcal{Q}_{jk}^{\mathrm{int}} = (N_k - \delta_{jk})\, \lambda_{jk}^{\mathrm{int}} \left(\Delta \sin^2 i \right)_{jk}^{\mathrm{int}} \,, \tag{2.83}$$

where the 'int' superscript stands for either 'vs', 'df', or 'coll' to indicate viscous stirring, dynamical friction, or collisional encounters, where $\lambda_{jk}^{\mathrm{int}}$ is the respective mutual particle

interaction rate for representative particle j and an interacting particle represented by representative particle k, and $(\Delta e^2)_{jk}^{\mathrm{int}}$ and $(\Delta \sin^2 i)_{jk}^{\mathrm{int}}$ are the average changes to the mean squared eccentricity and inclination of representative particle j inflicted by the interacting particle during an interaction.

2.4 Runaway growth

In Ormel et al. (2010), the hybrid statistical model was applied to study processes of *runaway growth* in which the collision rate is boosted by gravitational focussing, separating the heaviest bodies from the bulk of the particle mass distribution. By using the Monte Carlo method previously developed in Ormel and Spaans (2008), the majority of particles could be represented statistically, that is, by only tracking the rms values of an entire group of particles. Sufficiently massive bodies could be tracked as individual objects, which is crucial for the accurate modelling of the runaway growth process. The generalised model of Ormel et al. (2010), which we will adopt as a foundation of our work, will be introduced in detail in Sect. 6.1. In this section we will explain the principle of runaway growth by means of a simplified coagulation model.

We shall start by making the simplifying assumption that collisions between planetesimals always lead to 'hit-and-stick' coagulation. In principle, planetesimals can then grow without bounds through repeated collisional encounters. Once a body exceeds a characteristic mass, the interaction rate of the heaviest body is boosted by *gravitational focussing*, and runaway growth sets in. To understand this process conceptually, let us model accretional growth as a simple geometrical process of a homogeneous species of particles.

We assume that the relative motion, as defined in Eqs. (2.76–2.79), behaves quasi-thermally, which implies isotropy,

$$\left\langle e^2 \right\rangle = 2 \left\langle \sin^2 i \right\rangle \ . \tag{2.84}$$

If we regard particle motion as homogeneous and isotropic, we can express the geometric collision rate of a single particle as

$$\lambda = \rho_N \sigma v_{\mathrm{a}} \ , \tag{2.85}$$

where ρ_N is the number density of particles, and where σ denotes the geometric collision cross-section

$$\sigma = \pi R^2 \tag{2.86}$$

for spherical bodies with bulk radius R. v_{a} denotes the average approach velocity, that is, the average relative velocity at infinite distance, between any two particles, which can be related to the velocity deviation of particles, which in our isotropic model is treated as a random velocity,

$$v_{\mathrm{a}} \approx \sqrt{\left\langle \Delta \mathbf{v}^2 \right\rangle} \ . \tag{2.87}$$

The collision rate then begets a mass growth rate

$$\begin{aligned}\frac{\mathrm{d}m}{\mathrm{d}t} &= \lambda m \\ &= \rho_N m \sigma v_\mathrm{a} \,. \end{aligned} \tag{2.88}$$

Let us now assume that particles obey a vertical distribution of Gaussian shape:

$$\rho_N = \Sigma_N \frac{1}{\sqrt{2\pi}h} \exp\left[-\frac{z^2}{2h^2}\right] \tag{2.89}$$

with Σ_N the surface number density of particles, where the scale height h of the distribution is related to the rms inclination $\sqrt{\langle \sin^2 i \rangle}$,

$$h \approx r\sqrt{\langle \sin^2 i \rangle} \,. \tag{2.90}$$

We evaluate the number density ρ_N at midplane, $z = 0$, and then use the decomposition of the velocity deviation (Eq. (2.79)) and the isotropy relation (Eq. (2.84)) to find

$$\langle \sin^2 i \rangle = \frac{\langle \Delta \mathbf{v}^2 \rangle}{3 v_\mathrm{K}^2} \,, \tag{2.91}$$

which in turn can be used to eliminate v_a in the mass growth rate:

$$\frac{\mathrm{d}m}{\mathrm{d}t} \approx \Sigma_N m \sqrt{\frac{3}{2\pi}} \sigma \Omega_\mathrm{K} \tag{2.92}$$

where we used the relation $\Omega_\mathrm{K} = v_\mathrm{K}/r$ from Eq. (2.2).

If planetesimals grow by coagulation, their number, and hence their number density, will change over time. However, mass is conserved, and therefore the surface density $\Sigma = \Sigma_\mathrm{N} m$ remains invariant under coagulation. We can thus infer a scaling relation with mass m by inserting the geometric collision cross-section (Eq. (2.86)) and by substituting the bulk radius R using the spherical volume relation

$$m = \frac{4\pi}{3} \rho_\mathrm{s} R^3 \,, \tag{2.93}$$

where ρ_s is the solid density of the planetesimals, we find the scaling relation

$$\frac{\mathrm{d}m}{\mathrm{d}t} = \sqrt{\frac{3\pi}{2}} \Sigma \Omega_\mathrm{K} R^2 \tag{2.94}$$

$$\propto m^{2/3} \,, \tag{2.95}$$

and thus a growth timescale of

$$t_m \equiv m \left| \frac{\mathrm{d}m}{\mathrm{d}t} \right|^{-1} \propto m^{1/3} \,. \tag{2.96}$$

The growth timescale is found to increase with the particle mass m, which implies that growth, although continuing, slows down over time.

This changes, however, if we consider the effect of gravitational focussing (e.g. Armitage, 2017, §III.B.1). The mutual gravitational attraction increases the effective cross-section, enhancing the geometric cross-section by a factor of $(1 + \Theta)$,

$$\sigma \rightarrow \sigma\,(1 + \Theta) \ , \tag{2.97}$$

where $\Theta = v_{\rm esc}^2/v_{\rm a}^2$ is the Safronov number (e.g. Weidenschilling, 1989) with the *escape velocity*

$$v_{\rm esc} = \sqrt{\frac{2Gm}{R}} \tag{2.98}$$

and with G the gravitational constant. Due to momentum conservation, the approach velocity will decrease as bodies coagulate. Realistically, the rms velocities, related to the approach velocity as per Eq. (2.87), will grow again since mutual gravitational deflections lead to dynamical heating, but at the same time they will also be damped by gas drag. As modelling the interplay of these effects would require significantly more effort, we thus make the very simplistic approximation of keeping $v_{\rm a}$ invariant under growth. Then, $\Theta \propto v_{\rm esc} \propto m^{1/3}$, and the second term in Eq. (2.97) will eventually start to dominate, $\Theta \gtrsim 1$, as m grows. If we apply the $(1 + \Theta)$ factor to the mass growth rate (Eq. (2.94)) and the mass growth timescale (Eq. (2.96)), we find that, once the mass m has grown sufficiently large that $\Theta \gg 1$,

$$\frac{{\rm d}m}{{\rm d}t} \propto m^{4/3} \ , \qquad t_m \propto m^{-1/3} \ . \tag{2.99}$$

Once a critical mass has been reached and the Θ term begins to dominate, growth will therefore speed up.

In a heterogeneous distribution of particle masses, only the heaviest bodies will enter the accelerated growth regime, while the tail end of the distribution remains below the critical mass. The most massive bodies will 'run away' in terms of mass, eventually accumulating all dynamically accessible particles in their reach. This is referred to as *runaway growth*. In a spatially extended system, particles can accumulate only bodies within their gravitational reach – their 'feeding zone' –, which one might naïvely relate to the *Hill radius*

$$r_{\rm h} = r \left(\frac{m}{3M_*} \right)^{1/3} \ , \tag{2.100}$$

that approximately quantifies the distance from a body of mass m at orbital distance r from the central star at which the gravitational forces of the body and the central star are in balance. However, taking into account dynamical heating effects, Kokubo and Ida (1998) found that the typical orbital separation between concurrent runaway bodies ends up being much larger, $\sim 10 r_{\rm h}$, than the naïve estimate of $\sim r_{\rm h}$. Therefore, if spatial spread of particles is considered, the growth process may end with an ensemble of runaway bodies at spatial separation much greater than their mutual Hill radii.

The growing runaway body dynamically stirs up the smaller planetesimals with increasing efficiency as its mass increases, eventually dominating the stirring process. The ensuing growth of the rms velocities of the smaller planetesimals, which was neglected in above derivation, again damps the Θ term and decelerates growth. This regime is referred to as *oligarchic growth* (e.g. Chambers, 2006).

An improved Representative Particle Monte Carlo method for the simulation of particle growth 3

Reference: This chapter has been published as Beutel and Dullemond (2023).

A rocky planet is formed out of the agglomeration of around 10^{40} cosmic dust particles. As dust aggregates grow by coagulation, their number decreases. But until they have grown to hundreds of kilometres, their number still remains well above the number of particles a computer model can handle directly. The growth from micrometres to planetesimal-sized objects therefore has to be modelled using statistical methods, often using size distribution functions or Monte Carlo methods. However, when the particles reach planetary masses, they must be treated individually. This can be done by defining two classes of objects: a class of many small bodies or dust particles treated in a statistical way, and a class of individual bodies such as one or more planets. This introduces a separation between small and big objects, but it leaves open how to transition from small to big objects, and how to treat objects of intermediate sizes.

We aim to improve the *Representative Particle Monte Carlo* (RPMC) method, which is often used for the study of dust coagulation, to be able to smoothly transition from the many-particle limit into the single-particle limit. Our new version of the RPMC method allows for variable swarm masses, making it possible to refine the mass resolution where needed. It allows swarms to consist of few numbers of particles, and it includes a treatment of the transition from swarm to individual particles. The correctness of the method for a simplified two-component test case is validated with an analytical argument. The method is found to retain statistical balance and to accurately describe runaway growth, as is confirmed with the standard constant kernel, linear kernel, and product kernel tests as well as by comparison with a fiducial non-representative Monte Carlo simulation.

3.1 Introduction

How planets are formed is a difficult question to answer. We know that the process starts with micrometre-sized cosmic dust particles in a protoplanetary disk, which coagulate and form ever larger dust aggregates (Birnstiel et al., 2016). As these aggregates grow to a pebble size (of the order of millimetres to centimetres, henceforth conveniently called pebbles), they may, under certain conditions, form large overdensities in the disk that gravitationally collapse to form multi-kilometre sized planetesimals (Johansen et al., 2009; Wahlberg Jansson and Johansen, 2014). From there onwards, these planetesimals agglomerate gravitationally to form planetary embryos (Wetherill and Stewart, 1993; Kokubo and Ida, 1998), which, in turn, grow further via accretion of planetesimals and any remaining pebbles (Safronov, 1964; Ormel and Klahr, 2010; Lambrechts and Johansen, 2012; Bitsch et al., 2015).

One of the fundamental limiting factors of any numerical treatment of this problem is the vast number of particles involved. To form a single Earth-like planet, we need $\sim 10^{40}$ dust particles, $\sim 10^{30}$ pebbles, or $\sim 10^{10\cdots20}$ planetesimals (dependent on their mass). This would not necessarily be a problem if we could treat these as an equilibrium fluid, just in the way we treat a gas. Unfortunately, solid particles behave very differently than a gas, and thus require a more sophisticated treatment.

For small particles, the problem is often approached with a particle size distribution function $n(m)\mathrm{d}m$, giving the number of particles per unit volume for a particle mass interval $\mathrm{d}m$. This function is then sampled numerically on a grid in m, and the growth and fragmentation are then modelled using a version of the Smoluchowski coagulation equation (Smoluchowski, 1916). This method (which we call the continuum approach) has been used by numerous teams (e.g. Weidenschilling, 1980; Tanaka et al., 2005; Dullemond and Dominik, 2005) and is reasonably fast and efficient. Furthermore, it can be extended in space by adding the spatial dimensions to the coordinates (Birnstiel et al., 2010; Okuzumi et al., 2012; Drążkowska et al., 2019). It works very well for small particles because they are well coupled to the gas. These particles do not have independent orbital elements, but instead largely move along with the gas, except for a slow drift. However, with this method, it is hard to add independent particle properties. By the Eulerian nature of such a grid-based approach, all particles have the same properties for each location and mass. Approximate methods exist to overcome this limitation (Okuzumi et al., 2009; Stammler et al., 2017), but a full solution would require an extension of the dimensionality of the problem, creating a huge computational overhead. For instance, by adding only a porosity parameter given by the value p, the particle distribution function now becomes $n(m, p)\,\mathrm{d}m\,\mathrm{d}p$. This quickly pushes the problem beyond the limit of computational feasibility.

For large particles (planetesimals and upwards), the problem becomes even more severe, because now the particles are large enough to acquire their own independent orbital elements. A distribution function treatment as described above becomes unfeasible. Instead, the N-body method is the method of choice.

As a result of these (and other) complications, it is hard to develop a model of planet formation that starts with dust and ends with full-grown planets. The few complete dust-to-planets models available today involve a patchwork of different methods covering

different size regimes (e.g. Ormel et al., 2017; Schneider and Bitsch, 2021; Emsenhuber et al., 2021).

If we want to develop a single technique that can, at least in principle, handle large amounts of dust particles as well as individual planet-like bodies, and everything in between, the method must be based on a particle sampling approach. This could be a Monte Carlo method (stochastic), an N-body method (deterministic), or a blend of both. In this approach it is easy to add any number of particle properties to the problem, because we simply store these properties for each computational particle. Even if all particles are in the same location and have the same mass, we can then still handle a heterogeneous set of supplementary properties (porosity, charge, chemical composition, orbital elements, etc.), which is not possible for continuum methods.

Ormel et al. (2007) developed a Monte Carlo solver for dust coagulation in which particles can have not only a mass but also other properties such as porosity or ice content. This method stands at the basis of further works (e.g. Krijt et al., 2015) and overcomes the limitations of the continuum methods. While the method was developed for small particles, it has also been used for modelling runaway growth of planetesimal size bodies (Ormel et al., 2010).

An alternative Monte Carlo method was developed by Zsom and Dullemond (2008). They key difference with the method of Ormel et al. (2007) is that it is rigourously based on the concept of 'representative particles'. We therefore call this method the 'Representative Particle Monte Carlo' (RPMC) method and provide a brief review of the method in Section 3.2. This method has several advantages over other methods, one of them being its robustness and simplicity. But it has, so far, only been formulated in a somewhat restrictive sense: (1) All swarms must have the same mass, and (2) all swarms must contain a very large number of actual particles. As a result, this method has so far only been applied to problems of dust coagulation and fragmentation, where the number of actual particles is so large that condition (2) is always fulfilled. In this context the method has been successfully used for modelling dust coagulation, compactification, and fragmentation based on the full complexity of laboratory measurements (Zsom et al., 2010, 2011a). It has also been used for exploring the effect of external disturbances on the protoplanetary disk (Schneider and Bitsch, 2022), the effect of particle collisions during the gravitational collapse of a pebble cloud (Wahlberg Jansson et al., 2017), and for various other applications (e.g. Windmark et al., 2012; Drążkowska et al., 2014; Krijt and Ciesla, 2016; Krijt et al., 2016).

As the RPMC method stands, however, it cannot be used for problems where the growth proceeds to the point when swarms become individual particles (e.g. planets). In fact, the validity of the method, in the form presented by Zsom and Dullemond (2008), already formally breaks down well before that. And although the equal-mass swarms provide a certain robustness, and focus computational effort where the mass is, this condition can also severely limit the applications of the method. There are well-known particle growth problems where initially insignificant particles can grow to dominate the process, for example the planetesimal runaway growth regime (Greenberg et al., 1978) or the 'lucky particle' scenario (Windmark et al., 2012). To model these with a Monte Carlo approach either requires a very large number of representative particles, or the

clever splitting of swarms into smaller swarms to resolve particles of interest. This is only possible if the method allows swarms of different masses.

The goal of this chapter is to improve the RPMC method such that the drawbacks (1) and (2) are remedied. The method will then still have the advantages of robustness of the original RPMC method, but will gain in flexibility and applicability so that it can be used to model planet formation over a wide range of particle masses. In Chapter 4, we address another problem of the RPMC method: the n^2 scaling of all possible pair-wise interactions.

3.2 Conceptual summary of the original RPMC method

Let us consider the problem of coagulation or fragmentation of, say, $N = 10^{30}$ particles. Obviously we cannot model all of these particles individually. With the RPMC method we consider, of these 10^{30} particles, only n particles which together form a representative sample of the actual particles. The number n must be large enough that this sample provides a good enough representation of the true distribution. At the same time, it must be small enough to keep the computational cost manageable.

The sampling is mass-weighted. That means that a boulder with a mass of 1 000 kg will have a million times higher chance to be one of the representative particles than a pebble of 1 g. However, if there are a million times more pebbles than boulders (meaning that the total mass in pebbles equals the total mass in boulders), then the chance that representative particle i happens to be a pebble is the same as that it happens to be a boulder.

Now let us assume that representative particle i is a boulder of $m_\mathrm{B} = 1\,000\,\mathrm{kg}$. We let it undergo an event that causes it to fragment into a mass distribution $n_\mathrm{fr}(m)$. The total mass of the mass distribution is still 1 000 kg:

$$\int_0^{m_\mathrm{B}} m\, n_\mathrm{fr}(m)\, \mathrm{d}m \;=\; m_\mathrm{B}\ . \tag{3.1}$$

In the RPMC method, rather than splitting the numerical particle up into a large number of new numerical particles, we randomly choose a new mass for the representative particle using the probability distribution function

$$p(m)\, \mathrm{d}m = \frac{m\, n_\mathrm{fr}(m)}{m_\mathrm{B}}\, \mathrm{d}m\ . \tag{3.2}$$

In other words, we use mass-weighting to choose the new property of the representative particle, so the sampling remains mass-weighted. All the other fragments are 'forgotten'. If only a single such fragmentation event were to happen, such forgetfulness would be problematic, since a single randomly chosen particle does not represent an entire distribution of fragments. But since we are dealing with very large numbers of boulders, such fragmentation events will occur many times (each time for a different representative particle $j \neq i$). Statistically, the outcomes of many representative particles follow the fragment mass distribution $n_\mathrm{fr}(m)$.

The representative particles form a representation of the real (full) collection of particles at a given time. They contain all the information we have of the system at that

particular time. In order to reconstruct the full collection of particles from this limited information, the RPMC method follows the principle of Ockham's Razor: If we do not know more than the information we currently have of the representative particles, then the simplest assumption possible is that the other (unknown) particles are the same as the representative ones. Given that the representative particles follow a mass-weighted sampling, each of them represents a fraction $1/n$ of the total mass of solids. And so we divide the total mass up into n equal-mass parts, and assume each part to consist of particles with identical properties as their representative particle.

Although it might be possible to conceive different methods of extrapolating the full distribution of particles from the set of representative particles, our simple method has the advantage that it trivially guarantees the idempotency of the selection process. Regardless of the how representative particles are chosen from swarms (random pick, mass median, nearest to average, etc.), we would end up with an exactly equivalent set of representative particles after extrapolating a full particle ensemble and selecting a new set of representative particles.

We note that, so far, we have not used the concept of 'swarms'. The RPMC method does not require this concept. But to make it easier to talk about the method, and to compare it to the 'superparticles' often used in computational astrophysics (e.g. Johansen et al., 2007), it is convenient to define a swarm in the context of RPMC. The key lies in the mass-weighted sampling used by RPMC. If the total mass of solids is M, and we used n representative particles which are randomly sampling the solids in a mass-weighted fashion, then we can say that each representative particle represents a fraction $1/n$ of the total mass of solids. We then assign to each representative particle i a swarm of N_i particles with identical properties as the representative particle. The total mass of the swarm is $M_i = M/n$, so that $N_i = M_i/m_i$, where m_i is the mass of the individual representative particle. If the properties of the representative particle change, so will the properties of the other particles in the swarm. This also means that if the representative particle becomes more massive (i.e. m_i increases) due to, for instance, a collide-and-merge event with another particle, the other swarm particles also become more massive. As the total mass of the swarm stays constant, the number of particles in the swarm N_i will then drop. This is a purely statistical effect, not a true sudden disappearance of particles.

To better understand how the representative particles work, it is helpful to pretend that all particles are composites of the same elementary substance, and that a representative particle is identified with a 'tracer atom' of this substance. If tracer atoms were initially chosen at random from the entirety of atoms, then at any time the particle mass distribution of the system must be approximated by the mass distribution of representative particles, assuming that the number of particles is so much larger than the number of tracer atoms that the possibility of two tracer atoms ending up in the same particle can be neglected. In this picture, swarms just embody the simplest possible way of extrapolating an approximate global mass distribution from the mass distribution of representative particles.

We have not yet introduced any collisions between particles. The fragmentation event is, so far, a purely hypothetical spontaneous event affecting only the representative particle itself. The RPMC method can, however, easily handle collisions. Since each

swarm consists of truly many particles, when a representative particle of swarm j collides with a particle from swarm k, the probability that it hits the representative particle of swarm k is extremely low. Instead, it will most likely (with a probability close to 1) hit another particle of that swarm, leaving representative particle k alone. A collision nearly always affects both colliding particles. For instance, upon high-velocity impact, both particles can fragment. Or upon low-velocity impact, the particles can merge. But since with almost certainty the representative particle j will collide with a non-representative particle from swarm k, only representative particle j will fragment or merge, while representative particle k will be entirely unaffected as it is not involved in the collision. This means that only the properties of representative particle j, and by extrapolation those of swarm j, are changed, while swarm k stays unaffected. This asymmetric interaction between swarms is a key feature of the RPMC method. It means that on individual collision basis the results can be skewed, but statistically over many collisions the results are correct.

In Zsom and Dullemond (2008) the method is described in much more detail. It is explained how to randomly sample collision events, how to modify the representative particle involved, how to update the collision matrix, how to deal with pathological cases, etc. We discuss these issues in Section 3.4.3, where we define our improved version of the RPMC method.

An advantage of the RPMC method is that it naturally keeps the number of representative particles unchanged, even as particles merge or shatter. While the actual number of particles may decline (as particles coalesce) or increase (as particles shatter), the number of representative particles stays, by design, the same. This avoids the need of re-introducing computational particles as the number of particles declines (to keep the resolution optimal), or forcefully merging particles as the number of particles increases (to prevent runaway computational cost). Also, it naturally conserves mass. These properties make the RPMC method exceptionally stable and robust. But this robustness comes at a cost. First, the interactions between particles become asymmetric, which is not problematic but can be confusing. Furthermore, the method has the basic assumption of mass-weighted sampling, which automatically means that all swarms have the same total mass, namely M/n. And the method assumes that two representative particles have a negligible chance of colliding with each other, which requires that the swarm mass always is huge compared to the particle mass (many particles per swarm). Together, these disadvantages limit the applicability of the RPMC method. Addressing these shortcomings is the purpose of this chapter.

3.3 Mathematical formalisation of the RPMC method

3.3.1 Fundamentals

We consider an ensemble of N physical particles which interact through collisions. The number N will typically be very large (say, $N \sim 10^{30}$). Here and in the following, we refer to a 'physical particle' as any particle of the system we study, while only a (tiny) subset of these particles, the 'representative particles', are stored on the computer. Each physical particle k is associated with a particle mass m_k and possibly other properties. These particle properties, in particular the mass, may span many orders of magnitude.

We denote the full set of particle properties of particle k with the symbol $\mathbf{q}_k \in \mathbb{Q}$, where $\mathbb{Q}$ is the space of intrinsic properties (e.g. mass, charge, porosity) and statistical properties (e.g. velocity dispersion).

For every pair of particles j and k, we define the *raw collision rate* $\lambda(\mathbf{q}_j, \mathbf{q}_k)$ as the rate (in units of s^{-1}) at which both particles collide. It is necessarily commutative, $\lambda(\mathbf{q}_j, \mathbf{q}_k) = \lambda(\mathbf{q}_k, \mathbf{q}_j)$. If the particles are distributed homogeneously and isotropically in a volume V, the collision rate $\lambda(\mathbf{q}_j, \mathbf{q}_k)$ can be written as

$$\lambda(\mathbf{q}_j, \mathbf{q}_k) = V^{-1}\sigma(\mathbf{q}_j, \mathbf{q}_k)\,|\Delta\mathbf{v}(\mathbf{q}_j, \mathbf{q}_k)| \ , \tag{3.3}$$

where $\sigma(\mathbf{q}_j, \mathbf{q}_k)$ is the collision cross-section and $|\Delta\mathbf{v}(\mathbf{q}_j, \mathbf{q}_k)|$ the average relative velocity between the particles. Because particles cannot collide with themselves, we define a *true collision rate* λ_{jk} which explicitly suppresses self-collision:

$$\lambda_{jk} \coloneqq (1 - \delta_{jk})\lambda(\mathbf{q}_j, \mathbf{q}_k) \ . \tag{3.4}$$

The collision rate λ_j of particle j with any other particle in the ensemble is given by

$$\lambda_j \coloneqq \sum_{k=1}^{N} \lambda_{jk} \ . \tag{3.5}$$

The total rate of collisions is

$$\lambda \coloneqq \sum_{j=1}^{N} \sum_{k=j+1}^{N} \lambda_{jk} = \frac{1}{2} \sum_{j=1}^{N} \lambda_j \ . \tag{3.6}$$

Supposing, for the moment, that we had a small enough number of physical particles N that we can store them all on the computer (i.e. they would all be representative, though only of themselves), we would then proceed as follows to simulate the particle collisions. Following the method of Gillespie (1975), we choose a random time increment

$$\Delta t = -\lambda^{-1} \log(1 - \xi) \ , \tag{3.7}$$

where ξ is a uniform random number drawn from the interval $[0, 1)$. We then draw random indices j and k from the discrete distribution defined by the joint probability

$$P(j)P(k|j) = \frac{\lambda_{jk}}{2\lambda} \ , \tag{3.8}$$

where $P(j) = \frac{1}{2}\lambda_j/\lambda$ is the chance that particle j is involved in the collision event and $P(k|j) = \lambda_{jk}/\lambda_j$ is the chance that particle k is the collision partner of particle j given that particle j will suffer a collision. We then advance the system by the time increment Δt and let particles j and k collide. This collision will generally lead to a modification of the properties of the particles j and k involved. It can also lead to the merging of particles (in the case of coagulation) or to the creation of additional particles (by means of fragmentation). Therefore, the collision rate λ has to be recomputed after a collision. It also needs to be recomputed if the properties of the particles change by means of other processes.

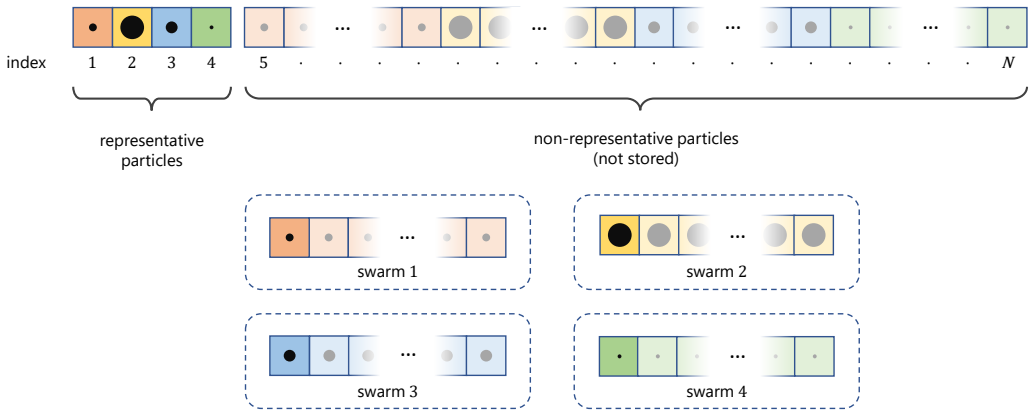


Figure 3.1: Indexing of the full ensemble of physical particles, with the representative particles ordered at the beginning of the sequence.
The entirety of particles in a given ensemble is indexed from 1 to N. The first n indices refer to RPs, and the remaining indices belong to non-RPs. Because N may be enormously large, tracking the entirety of particles is not feasible. Only the n RPs are actually modelled on the computer, while the properties of the $(N-n)$ non-RPs are estimated from the known properties of the n representative ones.

3.3.2 Representative particles

Simulating a full ensemble of $N \sim 10^{30}$ particles is computationally unfeasible. We therefore pick a representative subset of n *representative particles* with $n \ll N$ and simulate only them while still allowing them to interact with all the other physical particles. For convenience we abbreviate 'representative particle' as RP henceforth.

Also for notational convenience, we assume that all physical particles are indexed from 1 to N, but with the RPs arranged at the beginning of the ordering. In other words, the particles with indices 1 to n are the RPs, all others are not, as illustrated in Fig. 3.1.

Let us now define the *total collision rate of the RPs*, or more precisely: the total collision rate of all RPs (indices 1 to n) with all physical particles (indices 1 to N):

$$\bar{\lambda} := \sum_{j=1}^{n} \sum_{k=j+1}^{N} \lambda_{jk} \,. \tag{3.9}$$

We adapt Gillespie's method for the simulation of RPs by substituting $\bar{\lambda}$ for λ in Eq. (3.7). However, $\bar{\lambda}$ is not a computable quantity because the properties of the $(N-n)$ non-RPs are not known. To obtain a computable estimate of $\bar{\lambda}$, we first need to clarify how RPs actually represent other particles.

The basic assumption underlying the RPMC method is that a RP k can be regarded as a primus inter pares (first among equals), chosen from a larger *swarm* $\mathcal{S}_k$ of particles whose properties are similar to $\mathbf{q}_k$. Here, $\mathcal{S}_k$ is the index set of the N_k physical particles in the swarm k belonging to RP k. In other words: the n RPs divide the total ensemble of

N physical particles up into n swarms, each containing N_k physical particles, such that

$$N = \sum_{j=k}^{n} N_k \ . \tag{3.10}$$

Given that, for each swarm k, we only have information about the RP k, the simplest possible assumption about the properties of the other particles in that swarm is to say that they are not only similar in some unspecified way, but in fact identical:

$$\mathbf{q}_i \overset{!}{=} \mathbf{q}_k \quad \forall i \in \mathcal{S}_k \ . \tag{3.11}$$

What still needs to be defined is how large each swarm is. This is best defined through a conserved quantity. The only practical quantity that is strictly conserved, irrespective of the complexity of the model, is mass. And so we divide the total mass M up into swarms of masses M_k such that

$$M = \sum_{k} M_k \ . \tag{3.12}$$

In other words: each swarm k has a mass M_k, which is conserved throughout the simulation, unless a swarm is deliberately split or merged. The mass of a swarm and the number of particles in the swarm are directly related via

$$M_k = N_k m_k \ , \tag{3.13}$$

where m_k is the mass of each of the particles in the swarm k. In the original RPMC method of Zsom and Dullemond (2008) all swarms by design had the same mass, so that

$$M_k = \frac{M}{n} \ , \tag{3.14}$$

where M is the total mass in the system and n is the number of RPs, and thus also the number of swarms. We later show that this equal-mass partitioning of swarms is not necessary for the method to work.

3.3.3 Simulation of representative particles

To approximate the total collision rate of RPs in Eq. (3.9), we first split it in two terms,

$$\bar{\lambda} = \sum_{j=1}^{n} \left(\sum_{k=j+1}^{n} + \sum_{k=n+1}^{N} \right) \lambda_{jk} \ , \tag{3.15}$$

where the first term describes the collisions among RPs and the second term describes collisions between RPs and non-representative swarm particles.

By invoking the assumption that all particles in a swarm have identical properties, as stated by Eq. (3.11), we can replace the sum of RP–non-RP collision rates with the sum of RP–RP collision rates multiplied with the number of non-RP particles in the swarm,

$$\sum_{k=n+1}^{N} \lambda_{jk} = \sum_{i=1}^{n} (N_i - 1) \lambda(\mathbf{q}_j, \mathbf{q}_i) \qquad \text{for } j \in \{1, \ldots, n\} \ . \tag{3.16}$$

We note that the raw collision rate $\lambda(\mathbf{q}_j, \mathbf{q}_i)$ had to be used here instead of the true collision rate λ_{ji}. The reason is that, while a RP j cannot collide with itself, it may collide with another particle from its own swarm. In the true collision rate, λ_{jj} had been set to 0 to avoid self-collisions, so its use in Eq. (3.16) would unphysically avoid collisions between particles from the same swarm. Hence the use of the raw collision rate instead.

Rewriting the first term of Eq. (3.15) as

$$\sum_{j=1}^{n} \sum_{k=j+1}^{n} \lambda_{jk} = \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} \lambda_{jk} \ , \tag{3.17}$$

and using Eqs. (3.4, 3.16), we find

$$\bar{\lambda} = \sum_{j=1}^{n} \sum_{k=1}^{n} \left(N_k - \frac{1 + \delta_{jk}}{2} \right) \lambda(\mathbf{q}_j, \mathbf{q}_k) \ . \tag{3.18}$$

From this we can define the RP–swarm collision rate $\bar{\lambda}_{jk}$,

$$\bar{\lambda}_{jk} = \left(N_k - \frac{1 + \delta_{jk}}{2} \right) \lambda(\mathbf{q}_j, \mathbf{q}_k) \ , \tag{3.19}$$

and the cumulative collision rate of RPs $\bar{\lambda}_j$,

$$\bar{\lambda}_j = \sum_{k=1}^{n} \bar{\lambda}_{jk} \ . \tag{3.20}$$

Although λ_{jk} and $\lambda(\mathbf{q}_j, \mathbf{q}_k)$ are commutative, $\bar{\lambda}_{jk}$ is not.

The term $-(1 + \delta_{jk})/2$ in the above expressions accounts for the possibility that RP j collides with a particle from swarm k that is itself a RP. The probability that this happens is

$$\tilde{P}_{jk}^{\mathrm{rp}} = \frac{1 - \delta_{jk}}{N_k}. \tag{3.21}$$

For such (usually rare) events, we have to avoid double-counting, because this potential collision is also accounted for by RP k. So this collision has to be counted only half, meaning that we have to subtract $1/2$ from the number of particles N_k in the swarm k, as done in Eq. (3.18). If $j = k$, that is, if we consider collisions of RP j with members of its own swarm, then instead of subtracting just $1/2$ we have to subtract 1, because apart from the RP itself, there are only $(N_j - 1)$ particles in the swarm. Hence the term $-(1 + \delta_{jk})/2$ in Eq. (3.18).

3.3.4 The large-particle-number approximation of the original RPMC method

The original RPMC method now makes the additional assumption that the number of particles per swarm is much greater than unity,

$$N_k \gg 1 \qquad \forall k \in \{1, \ldots, n\} \ . \tag{3.22}$$

This allows neglecting the probability of RP–RP collisions ($\bar{P}_{jk}^{\mathrm{rp}} \approx 0$). With this assumption, we can define the simplified RP–swarm collision rate as

$$\tilde{\lambda}_{jk} = N_k \lambda(\mathbf{q}_j, \mathbf{q}_k) \approx \bar{\lambda}_{jk} \tag{3.23}$$

and the simplified cumulative collision rates as

$$\tilde{\lambda}_j = \sum_{k=1}^{n} \tilde{\lambda}_{jk} \ , \qquad \tilde{\lambda} = \sum_{j=1}^{n} \tilde{\lambda}_j \ . \tag{3.24}$$

Gillespie's method is then applied to simulate RP collisions by substituting λ with $\tilde{\lambda}$ in Eq. (3.7):

$$\Delta t = -\tilde{\lambda}^{-1} \log(1 - \xi) \ . \tag{3.25}$$

A RP suffering a collision is then chosen by drawing an index j from the discrete distribution with probability

$$\tilde{P}(j) = \frac{\tilde{\lambda}_j}{\tilde{\lambda}} \ . \tag{3.26}$$

Then a collision partner from swarm k is chosen, where the index k is drawn from the discrete distribution with the probability

$$\tilde{P}(k|j) = \frac{\tilde{\lambda}_{jk}}{\tilde{\lambda}_j} \ . \tag{3.27}$$

3.3.5 Stochastic representation of collisional outcomes

With this method we can randomly determine the time of the next collision event, the RP j suffering a collision, and the swarm k which represents the collision partner. The collision can have a variety of results: coagulation, fragmentation, pulverisation, cratering, etc. (Güttler et al., 2010; Blum, 2018; Wurm and Teiser, 2021), and it can yield a wide variety of particle distributions ranging from a single merged particle to an almost-continuous spectrum of fragment masses. This raises the question of how the resulting particle distribution can be incorporated into the simulation model.

Of course, in case of fragmentation or pulverisation, adding every resulting particle to the simulation is infeasible. Instead, we use the sampling nature of the Monte Carlo process to let the resulting particle distribution realise itself stochastically. Let

$$n^{\mathrm{coll}}(\mathbf{q}; \mathbf{q}_j, \mathbf{q}_k) =: n_{jk}^{\mathrm{coll}}(\mathbf{q}) \tag{3.28}$$

denote the number density distribution of the resulting fragments from a collision of RP j with a particle from swarm k. It is normalised as follows:

$$\int m(\mathbf{q})\, n_{jk}^{\mathrm{coll}}(\mathbf{q})\, \mathrm{d}\mathbf{q} = m(\mathbf{q}_j) + m(\mathbf{q}_k) \ , \tag{3.29}$$

where $m(\mathbf{q})$ is the mass of a particle with properties $\mathbf{q}$, and the integration is over all possible properties $\mathbf{q}$.

We now draw a single set of particle properties $\mathbf{q}^{\text{coll}}$ from the weighted distribution

$$P_{jk}(\mathbf{q}) = \frac{m(\mathbf{q}) n_{jk}^{\text{coll}}(\mathbf{q})}{m(\mathbf{q}_j) + m(\mathbf{q}_k)} \ . \tag{3.30}$$

In other words, a collision in our simulation always consumes the representative particle and replaces it with a new representative particle sampled from the particle mass distribution of the collision outcome,

$$\mathbf{q}_j \to \mathbf{q}^{\text{coll}} \ . \tag{3.31}$$

The masses of both swarms do not change,

$$M_j \to M_j \ , \quad M_k \to M_k \ . \tag{3.32}$$

For a single collision, the method always chooses a single outcoming particle, so that the number of RPs remains constant. If the collision leads to merging, then the resulting particle will have a mass $m(\mathbf{q}_j)+m(\mathbf{q}_k)$. If, instead, the collision leads to a distribution of fragments, then the resulting particle will be a single mass-weighted choice among those fragments. The fragment distribution will then emerge as large numbers of collisions lead to many samplings of similar fragment distributions.

As demonstrated in Zsom and Dullemond (2008), the RPMC method yields the expected result for the standard analytic coagulation tests, albeit with substantial noise owed to the Monte Carlo nature of the method. The method also has the desirable consequence that the number of RPs always remains constant, which allows for simpler and more efficient implementation of the method. Finally, the representativeness of the sampling remains optimal: every RP keeps representing the same total mass.

3.4 The improved RPMC method

As mentioned before, the RPMC method, as described in Sect. 3.3, has two major limitations: (1) all swarms have the same mass and (2) all swarms must contain a very large number of physical particles. Here we present an improved RPMC method that overcomes these limitations. We put these improvements to the test in later sections.

3.4.1 Unequal-mass swarms

The original RPMC method assumed that all swarms have equal mass $M_k = M/n$. The reason for this choice was that the representative particles were considered to be randomly chosen from the full set of particles, using mass as the weight factor. This fits well to the 'atomistic' picture mentioned above: we randomly pick n 'atoms', all having equal weight, and then consider, for each representative atom, the dust particle it is in to be the representative particle. This guarantees that the n representative particles together form an unbiased sampling of the full ensemble of particles.

However, as we saw in Sect. 3.3.2, the formalisation of the RPMC algorithm allows for unequal swarm masses $M_k \neq M/n$, as long as the sum of all these masses equals the

total mass M. The way this enters into the algorithm is through the modified number of particles per swarm

$$N_k = \frac{M_k}{m_k} \tag{3.33}$$

which appears in the collision rate calculation in Eq. (3.18). If we define the 'weight' of a RP as

$$W_k = \frac{M_k}{M} \tag{3.34}$$

then in the original RPMC method all RPs had equal weight (i.e. they were all equally 'important'). But in the improved RPMC method the RPs are now no longer necessarily of equal weight.

We can split any swarm up into a number of smaller swarms. For example, we might replace a RP k with weight W_k with two RPs l, m with weights $W_l + W_m = W_k$. Or to put it in terms of their swarms: we split the swarm of mass M_k up into two swarms of masses $M_l + M_m = M_k$. Because the particles in both swarms have identical mass, $m_l = m_m = m_k$, the total number of particles is not changed, $N_l + N_m = N_k$.

3.4.2 The limits to growth

A much more difficult problem is to enable the RPMC method to deal with swarms that contain only a few or even just a single particle.

The original RPMC method was constructed to simulate a growth process over many orders of magnitude with a constant number of RPs while keeping resolution optimal. The method crucially relies on the assumption stated in Eq. (3.22) that the number of particles in each swarm is much greater than unity, $N_k \gg 1 \, \forall k \in \{1, \ldots, n\}$. Equivalently, the particle mass is assumed to be much smaller than the swarm mass, $m_k \ll M_k \, \forall k \in \{1, \ldots, n\}$. This assumption allowed the use of the simplified collision rate $\tilde{\lambda}_{jk}$ defined in Eq. (3.23) instead of $\bar{\lambda}_{jk}$ in Eq. (3.18), and it allowed us to keep the number of RPs n constant because we could neglect RP–RP collisions.

When the particle mass m_k is no longer negligibly small compared to the swarm mass M_k, several problems appear with the original RPMC method. Firstly, the number of particles in a swarm is computed through $N_k = M_k/m_k$. Like all RP and swarm properties, N_k is an expectation value. Therefore, a fractional swarm particle count such as $N_k = 1.3$ is not per se problematic. But if we want to treat single-particle swarms as non-stochastic objects – for example, we might want to identify singular particles with N-body objects in a hybrid MC/N-body simulation – we must decide for an integer number of particles. However, by rounding N_k to an integer, we no longer conserve the total mass of the system. Secondly, by neglecting the possibility of RP–RP collisions, we were able to adopt the simplified RP–swarm collision rate of Eq. (3.23). $\tilde{\lambda}_{jk} = N_k \lambda(\mathbf{q}_j, \mathbf{q}_k)$. But for $N_k \sim 1$, this collision rate significantly overestimates the actual RP–swarm collision rate given in Eq. (3.19). In the extreme case of $N_k = 1$, we have

$$\tilde{\lambda}_{jk} = \lambda(\mathbf{q}_j, \mathbf{q}_k) \, ; \quad \bar{\lambda}_{jk} = \frac{1 - \delta_{jk}}{2} \lambda(\mathbf{q}_j, \mathbf{q}_k) \, . \tag{3.35}$$

(To solve this problem, one might be tempted to simply use the actual RP–swarm collision rate $\bar{\lambda}_{jk}$ instead of approximating it as $\tilde{\lambda}_{jk}$. However, it turns out that the approximation $\bar{\lambda}_{jk} \approx \tilde{\lambda}_{jk}$ must be made, otherwise the RPMC method is not statistically balanced, as demonstrated in Appendix 3.C.) Thirdly, a particle cannot grow in mass beyond the swarm mass: $m_k \leqslant M_k$, whereas in reality it might do so. The original RPMC method strictly keeps M_k constant in time, meaning that this creates an artificial upper bound to the mass a physical particle can acquire. When increasing the number of swarms n, we can decrease the number of particles N_k in swarms $k \in \{1, \dots, n\}$ to improve the resolution, but as a result, we also decrease the largest individual mass $m_k \leqslant M_k$ which the simulation can represent. Finally, if N_k is not large, the probability that RP j will collide with the RP of swarm $k \neq j$ is not negligibly small.

3.4.3 Extending the method

To overcome the problems outlined in the previous section, we propose some modifications to the RPMC method. As the basis of our extension, we first classify all swarms $k \in \{1, \dots, n\}$ in two categories: *many-particles swarms* with $N_k > N_{\text{th}}$, and *few-particles swarms* with $N_k \leqslant N_{\text{th}}$, where the simulation parameter N_{th} is the *particle number threshold*.

Every RP is associated with a swarm. When a RP j interacts with a particle from swarm k, we can use the above classification to define *interaction regimes*: the interaction operates in the *many-particles regime* if both swarms j, k are many-particles swarms, and in the *few-particles regime* if at least one swarm of j, k is a few-particles swarm.

We then use the following adaption of the effective RP–swarm collision rate:

$$\tilde{\lambda}_{jk} = N_{jk} \lambda(\mathbf{q}_j, \mathbf{q}_k) \tag{3.36}$$

where we define the *swarm multiplicity factor* N_{jk}:

$$N_{jk} := \begin{cases} N_k & \text{in the many-particles regime} \\ N_k - \frac{1+\delta_{jk}}{2} & \text{in the few-particles regime} \end{cases} \tag{3.37}$$

Interactions in the many-particles regime are handled with the conventional RPMC method: we assume that a RP j always interacts with a non-RP from swarm k. A collision then alters only the properties of RP j.

If the interaction operates in the few-particles regime, we make the simplest possible choice and impose that, in the swarm with fewer particles, all particles operate in 'lockstep', that is, they all do as the RP does, at the same time. Therefore, instead of RP–swarm interactions, we now consider swarm–swarm interactions. Imposing $N_j \leqslant N_k$ without loss of generality, the result of the collision is still determined following the procedure described in Sect. 3.3.5, $\mathbf{q}_j \to \mathbf{q}^{\text{coll}}$, but the swarm masses M_j and M_k now grow and shrink as mass is transferred to the swarm with fewer particles:

$$M_j \to M_j + N_j m_k \,, \qquad M_k \to M_k - N_j m_k \,. \tag{3.38}$$

By assumption of $N_j \leqslant N_k$, the swarm mass M_k cannot become negative.

Interactions in the few-particles regime will therefore upset the sampling balance: as swarm weights change, different RPs represent different fractions of the total mass. As was argued in Sect. 3.4.1, this does not impair the validity of the simulation, but it may lead to inefficient use of the available RPs. The particle number threshold N_{th} should therefore be chosen as small as possible so that the sampling is not unnecessarily disequilibrated. At the same time, N_{th} should be large enough such that, for $N_k \geqslant N_{\mathrm{th}}$, the simplified RP–swarm collision rate Eq. (3.23), $\tilde{\lambda}_{jk} = N_k \lambda(\mathbf{q}_j, \mathbf{q}_k)$, only negligibly overestimates the RP–swarm collision rate $\bar{\lambda}_{jk}$. In our tests we found satisfactory results with $N_{\mathrm{th}} \sim 10$.

This simple prescription allows unconstrained growth in the RPMC method while conserving mass and keeping the RP count unchanged. However, it has some implications that are detrimental to our goal of modelling runaway growth processes with high accuracy. First, unless N_j happens to be a power of 2, swarm self-interaction will still result in a fractional particle number. More gravely, the method usually does not resolve the runaway body individually. Let us consider a swarm j representing the heaviest bodies. By sweeping up smaller particles, the swarm has just become a few-particles swarm, $N_j = N_{\mathrm{th}}$. Let us assume $N_{\mathrm{th}} = 10$. The mass of RP j will grow further by coagulation of smaller bodies; but because of the mass transfer precept given in Eqs. (3.38), the swarm body count will stay the same, $N_j = 10$. In other words, we have 10 bodies simultaneously experiencing runaway growth. This is at odds with the mass separation characteristic of runaway growth, sometimes paraphrased as 'the winner takes it all'. Runaway growth happens when the accretion rate has a superlinear proportionality with mass. Then, the body which first enters the runaway growth regime has the largest as well as the fastest-growing accretion rate; it will separate from the particle mass distribution and 'starve out' its competitors. Even worse, if bodies in the runaway regime cannot accrete each other, and their swarms will therefore not self-interact, then the swarm particle count cannot drop below N_{th} at all, and runaway bodies will never be resolved individually.

We therefore relinquish a core property of the RPMC method: we allow the number of RPs to grow. We impose that a swarm j that reaches the particle number threshold, $N_j \searrow N_{\mathrm{th}}$, is split into N_{th} single-particle swarms. The RP of a single-particle swarm i with $N_i = 1$ represents only itself.

To accurately model a runaway process, the initial number of RPs n^0 and the particle number threshold N_{th} must then be chosen large enough that swarms whose RPs surpass the critical mass are already resolved individually:

$$N_{\mathrm{th}} n^0 > \frac{M}{m_{\mathrm{crit}}} \,. \tag{3.39}$$

There is still the question of how to handle fractional particle counts. A many-particles swarm k becomes a few-particles swarm as the swarm particle count falls to or below the particle number threshold, $N_k \leqslant N_{\mathrm{th}}$. This can happen in two different ways: (1) RP k may grow by accumulating mass through a collision in the many-particles regime where the swarm mass M_k is conserved, thereby decreasing the swarm particle count $N_k = M_k/m_k$; or (2) a few-particles swarm may subtract mass from swarm k through a few-particles interaction as per Eq. (3.38). In both cases, N_k may end up non-integral.

Some kind of compromise then has to be made when splitting up swarm k into individual RPs. Assuming non-integral N_k, some possibilities are: (i) slightly adjusting the mass m_k before the splitting such that N_k becomes an integral number; (ii) splitting up the swarm to $\lceil N_k \rceil$ particles, one of which is assigned a slightly lower mass $(N_k - \lfloor N_k \rfloor)m_k$. In our implementation we chose the latter option. We argue that, if N_{th} is chosen large enough, this adjustment is of negligible significance. Furthermore, the boosting technique described in the next section will actually make the swarm particle count N_k an integral number in most situations where a regime transition would occur.

Fragmentation can be another cause of fractional particle counts. Consider a few-particles swarm j, $N_j \leqslant N_{\mathrm{th}}$ whose representative particle collides with a particle from some other swarm k, resulting in a distribution of fragment masses. As per Sect. 3.3.5, a new mass m'_j is sampled from the number density distribution of the resulting fragments, while swarm j absorbs the mass of the impacting particles from swarm k as per Eq. (3.38). The new particle number can then become non-integral, for example if a cratering event leaves the new particle mass m'_j only slightly smaller than the combined mass, $m'_j = (1 - \epsilon)(m_j + m_k)$ with $\epsilon \ll 1$:

$$N'_j = \frac{M'_j}{m'_j} = \frac{N_j}{1 - \epsilon} \approx N_j(1 + \epsilon) \; . \tag{3.40}$$

This incongruity points at a more severe problem: statistical treatment is not appropriate for fragmentation of individual bodies. As an example, consider the collision of proto-Earth and a protoplanet called Theia, an event which is hypothesised to have formed the Moon by cratering. We assume that proto-Earth is represented by RP j with $N_j = 1$. With the collision method discussed above, the mass of proto-Earth and Theia would be combined into a new unified swarm, and the new mass of its representative body would be chosen from the distribution of the fragments as either cratered proto-Earth (most likely), the Moon (with a chance of $1 : 82$), or debris (unlikely). The collision would thus likely result in a swarm of 1.0123 Earth-mass planet(s) or, on rare occasions, 82 Moons. This is clearly meaningless and wrong.

To ameliorate this we suggest, for non-coagulating collisions involving self-representing RPs, to decree that every fragment whose mass exceeds a certain threshold m_{th} be represented individually, and therefore to add new RPs to the simulation as required. From the sub-m_{th} end of the fragment distribution one can then sample one or more statistically representative fragments as RPs. With this approach, the collision of proto-Earth and Theia would result in two self-representing RPs of Earth and Moon mass and one or more RPs representing a swarm of debris.

3.4.4 Boosting

Despite tracking only a small subset of the particles, the simulation still models every single interaction of the representative particles. For wide distributions of particle masses, this is a severe constraint on the efficiency of the simulation. For example, assume that a boulder with a mass of 1 000 kg accretes pebbles of mass 1 g. To grow to twice its initial mass, the boulder must accrete 10^6 pebbles, and each interaction is modelled as a separate RPMC event, which slows down the simulation tremendously. This

problem was noted by Zsom and Dullemond (2008, §2.3), who proposed to group accumulation events of small particles together, effectively decreasing the collision rate while proportionally increasing the gain of mass during an individual event. We now introduce a formalised version of this boosting procedure.

3.4.4.1 Grouping interactions

To overcome the inefficiency inherent in a method that tracks every individual collision event, we want to group together similar events. To this end, we introduce a *boost factor* β_{jk} for interactions between RP j and swarm k. If the collision can be assumed to lead to 'hit-and-stick' coagulation, the collision rate $\tilde{\lambda}_{jk}$ from Eq. (3.36) is substituted with the *boosted collision rate*

$$\tilde{\lambda}^{\mathrm{b}}_{jk} = N^{\mathrm{b}}_{jk} \lambda(\mathbf{q}_j, \mathbf{q}_k) \ , \tag{3.41}$$

where we defined the *boosted swarm multiplicity factor*

$$N^{\mathrm{b}}_{jk} \coloneqq \beta^{-1}_{jk} N_{jk} \ , \tag{3.42}$$

and the mass transferred to the RP during a collision then is

$$m_j \to m'_j = m_j + \beta_{jk} m_k \ . \tag{3.43}$$

For interactions in the few-particles regime, we again impose $N_j \leqslant N_k$ without loss of generality. During a boosted coagulation, mass is then transferred between swarms j and k as per

$$\begin{aligned} M_j \to M'_j &= M_j + \beta_{jk} N_j m_k \ , \\ M_k \to M'_k &= M_k - \beta_{jk} N_j m_k \ . \end{aligned} \tag{3.44}$$

It is clear that

$$\beta_{jk} \geqslant 1 \tag{3.45}$$

must hold, where $\beta_{jk} = 1$ neutralises the boosting effect. In the following we discuss further constraints for β_{jk}, and how a suitable boost factor can be determined while obeying these constraints.

3.4.4.2 Constraining the boost factor

Zsom and Dullemond (2008) propose to set a mass accumulation threshold $q_m \coloneqq \delta m / m$ and choose the boost factor such that each collision event grows the mass of the larger particle by at least q_m. For example, if we wanted to accept a relative mass gain of 5% during a single collision event, we would set $q_m = 0.05$ and choose an initial boost factor of

$$\beta^0_{jk} \coloneqq q_m \frac{m_j}{m_k} \ . \tag{3.46}$$

The boost factor β_{jk} is subject to additional constraints. If the collision operates in the many-particles regime, growth by coagulation will decrease the number of particles in swarm j as the mass of its RP grows,

$$N'_j = \frac{M'_j}{m'_j} = \frac{M_j}{m_j + \beta_{jk} m_k} \,. \tag{3.47}$$

To avoid an abrupt 'regime change', we would like to avoid boosting collisions such that the swarm particle count falls below the particle number threshold, and hence we require $N'_j \geqslant N_{\mathrm{th}}$, which implies the constraint

$$\beta_{jk} \leqslant \frac{m_j}{m_k} \left(\frac{N_j}{N_{\mathrm{th}}} - 1 \right) \,. \tag{3.48}$$

For the few-particles regime, we had imposed $N_j \leqslant N_k$ without loss of generality. This allowed us to pretend that all particles in a swarm j grow by accumulating one particle from swarm k each. When introducing a boost factor, we must ensure that the new swarm mass M'_k remains a non-negative value; hence it follows from Eq. (3.44) that the boost factor must satisfy

$$\beta_{jk} \leqslant \frac{N_k}{N_j} \,. \tag{3.49}$$

An interaction in the few-particles regime may change the mass of swarm k. If swarm k is a many-particles swarm, the loss of mass might push its swarm particle count $N'_k = N_k - \beta_{jk} N_j$ below N_{th}, which we again want to avoid:

$$\beta_{jk} \leqslant \frac{N_k - N_{\mathrm{th}}}{N_j} \,. \tag{3.50}$$

However, we want to allow for the depletion of a many-particles swarm, and thus apply this constraint only if the swarm would not be depleted because $\beta_{jk} < N_k / N_j$.

To summarise, closed-form expressions for the boost factor can be given as

$$\beta_{jk} = \max \left\{ 1, \min \left\{ \beta^0_{jk}, \frac{m_j}{m_k} \left(\frac{N_j}{N_{\mathrm{th}}} - 1 \right) \right\} \right\} \tag{3.51}$$

for interactions in the many-particles regime, and as

$$\beta_{jk} = \begin{cases} \max \left\{ 1, \min \left\{ \frac{N_k - N_{\mathrm{th}}}{N_j}, \beta^0_{jk} \right\} \right\} & \text{if } N_k > N_{\mathrm{th}} \text{ and } \beta^0_{jk} < \frac{N_k}{N_j} \\ \frac{N_k}{N_j} & \text{if } N_k > N_{\mathrm{th}} \text{ and } \beta^0_{jk} \geqslant \frac{N_k}{N_j} \\ 1 & \text{if } N_k \leqslant N_{\mathrm{th}} \end{cases} \tag{3.52}$$

for interactions in the few-particles regime.

3.4.4.3 Boosting and fragmentation

The boost factor effectively groups particles together before having them accreted. Boosting is therefore not directly applicable if collisions lead to other outcomes, such as pulverisation or cratering. Depending on the sophistication of the fragmentation model

used, a given collision may lead to either coagulation or fragmentation with a certain probability, or only a fraction of the combined mass is shattered to smaller pieces. For an overview of the findings of laboratory studies of dust coagulation and fragmentation, see Güttler et al. (2010); Blum (2018).

A simple way of keeping the benefits of boosting while allowing for fragmentation is to exploit the probabilistic nature of the simulation. When simulating a collision, the new mass of the RP is chosen randomly by sampling the fragment mass distribution as explained in Sect. 3.3.5. If a fraction $(1 - f_{\mathrm{frag}})$ of the total mass remains in one cohesive body, then the probability that this massive body is chosen as the RP is $(1 - f_{\mathrm{frag}})$. Equivalently, one could say that the collision rate $\tilde{\lambda}_{jk}$ is decomposed in two contributions for fragmentation and for coagulation,

$$\tilde{\lambda}_{jk} = f_{\mathrm{frag}}\tilde{\lambda}_{jk} + (1 - f_{\mathrm{frag}})\tilde{\lambda}_{jk} \ . \tag{3.53}$$

A boost factor is then applied only to the partial collision rate corresponding to coagulation:

$$\tilde{\lambda}^{\mathrm{b}}_{jk} = f_{\mathrm{frag}}\tilde{\lambda}_{jk} + \beta^{-1}_{jk}(1 - f_{\mathrm{frag}})\tilde{\lambda}_{jk} \ . \tag{3.54}$$

Boosting decreases the effective collision rate, hence the likelihood f_{frag} that the RP is sampled from the mass distribution of fragments must increase:

$$f_{\mathrm{frag}} \to \frac{f_{\mathrm{frag}}}{f_{\mathrm{frag}} + \beta^{-1}_{jk}(1 - f_{\mathrm{frag}})} =: f^{\mathrm{b}}_{\mathrm{frag}} \ . \tag{3.55}$$

Bukhari Syed et al. (2017) shows that the transition from bouncing to fragmentation is smooth: the mass of the largest remaining fragment

$$m'_{\mathrm{max}} = (1 - f_{\mathrm{frag}})(m_j + m_k) \ , \tag{3.56}$$

which is equal to the mass of the heavier particle in the case of bouncing, $m'_{\mathrm{max}} = \max\{m_j, m_k\}$, continuously decreases as the kinetic impact energy increases. If the swarm particle k is much lighter than RP j, $m_k \ll m_j$, a large number of cratering impacts are required to effectuate a significant change in the mass of the largest fragment. Similar to coagulation events, we can also group cratering events together by generalising the initial boost factor (Eq. (3.46)) and boosted mass transfer (Eq. (3.43)) as

$$\beta^0_{jk} := q_m \frac{m_j}{|\delta m|} \ , \tag{3.57}$$

$$m_j \to m'_j = m_j + \beta_{jk}\delta m \ , \tag{3.58}$$

where the transferred mass is given as

$$\delta m := (1 - f_{\mathrm{frag}})\,(m_j + m_k) - m_j \ . \tag{3.59}$$

3.5 Validating the statistical balance

The RPMC method is asymmetric at its core: when operating in the many-particles regime, it is assumed that a RP always interacts with a non-RP, and hence only the

properties of the RP are changed. This asymmetry seems natural in the atomistic picture illustrated in Sect. 3.2. However, in the RP–swarm picture, correctness is not obvious. Consider the grains-and-boulders example originally given in Zsom and Dullemond (2008, §3.1). We have an initial ensemble of particles with only two different masses: N_{B}^0 heavy bodies with mass m_{B}^0, henceforth called *boulders*, and N_{g}^0 light particles with mass $m_{\mathrm{g}} < m_{\mathrm{B}}^0$, from now on referred to as *grains*. We assume that mass is the only relevant particle property. We further assume that collisions are always of the hit-and-stick type, and that they happen only between boulders and grains. Because boulders do not collide with each other, their number cannot decrease. Likewise, because grains do not suffer mutual collisions, they cannot grow. Therefore, the number of boulders must remain invariant as they grow by accreting grains. In the RP–swarm picture, when a grain-mass RP hits and accretes a boulder, its entire swarm is converted to boulders, causing a sudden increase in the number of boulders. Conversely, when a boulder-mass RP j accretes a grain, its mass grows, $m_j \to m_j + m_{\mathrm{g}}$, but the mass of its swarm is not changed, $M_j \to M_j$. Therefore, the number of boulders in the swarm $N_j = M_j/m_j$ decreases, $M_j/m_j \to M_j/(m_j + m_{\mathrm{g}})$. This section will formally prove that, for the specific case of the grains-and-boulders example, the two effects indeed cancel statistically.

3.5.1 Grains and boulders

We first define the collision rate as

$$\lambda(m, m') = \begin{cases} \Lambda(m, m') & \min\{m, m'\} < m_{\mathrm{B}}^0 \text{ and} \\ & \max\{m, m'\} \geqslant m_{\mathrm{B}}^0 \\ 0 & \text{otherwise,} \end{cases} \tag{3.60}$$

where the mutual collision rate $\Lambda(m, m')$ is an arbitrary commutative function of the masses m, m'. Boulders can therefore collide with grains, but collisions among boulders and among grains are suppressed.

Let us now represent this system by a selection of $n = n_{\mathrm{g}} + n_{\mathrm{B}}$ RPs, n_{g} of which represent the grains, and n_{B} represent the boulders. For notational convenience, we group the RP indices in two index sets $\mathcal{I}_{\mathrm{g}}$ and $\mathcal{I}_{\mathrm{B}}$ at any given time t, where $\mathcal{I}_{\mathrm{g}}$ holds the indices of the grain-mass RPs, and the indices in $\mathcal{I}_{\mathrm{B}}$ refer to boulder-mass RPs:

$$\begin{aligned} \mathcal{I}_{\mathrm{g}} &:= \left\{ i \in \{1, \ldots, n\} : m_i < m_{\mathrm{B}}^0 \right\} , \\ \mathcal{I}_{\mathrm{B}} &:= \left\{ j \in \{1, \ldots, n\} : m_j \geqslant m_{\mathrm{B}}^0 \right\} . \end{aligned} \tag{3.61}$$

In the RPMC simulation, the total number of grains N_{g} and total number of boulders N_{B} are given by

$$N_{\mathrm{g}} = \sum_{i \in \mathcal{I}_{\mathrm{g}}} N_i , \quad N_{\mathrm{B}} = \sum_{j \in \mathcal{I}_{\mathrm{B}}} N_j . \tag{3.62}$$

With the mutual collision rate between a boulder j and a grain conveniently abbreviated as

$$\Lambda_j := \Lambda(m_{\mathrm{g}}, m_j) , \tag{3.63}$$

we find the RP–swarm collision rates of Eq. (3.23) to be

$$\begin{aligned}\tilde{\lambda}_{ij} &= N_j \Lambda_j =: \tilde{\lambda}_{\mathrm{g}j} \ , \\ \tilde{\lambda}_{ji} &= N_i \Lambda_j \end{aligned} \tag{3.64}$$

and the cumulative collision rates of Eq. (3.24) to be

$$\begin{aligned}\tilde{\lambda}_i &= \sum_{j' \in \mathcal{I}_\mathrm{B}} N_{j'} \Lambda_{j'} \equiv N_\mathrm{B} \langle \Lambda \rangle_\mathrm{B} =: \tilde{\lambda}_\mathrm{g} \ , \\ \tilde{\lambda}_j &= N_\mathrm{g} \Lambda_j =: \tilde{\lambda}_{\mathrm{B},j} \ , \end{aligned} \tag{3.65}$$

where $i \in \mathcal{I}_\mathrm{g}$ and $j \in \mathcal{I}_\mathrm{B}$, and where we defined the RP–swarm collision rate $\tilde{\lambda}_{\mathrm{g}j}$ for any grain-mass RP with boulder swarm j, the cumulative grain-mass RP collision rate $\tilde{\lambda}_\mathrm{g}$, and the cumulative RP collision rate $\tilde{\lambda}_{\mathrm{B},j}$ for boulder-mass RP j. The boulder average $\langle X \rangle_\mathrm{B}$ of a particle-specific quantity X_j was defined as

$$\langle X \rangle_\mathrm{B} \equiv \frac{1}{N_\mathrm{B}} \sum_{j \in \mathcal{I}_\mathrm{B}} N_j X_j \ . \tag{3.66}$$

A grain-mass RP can collide with a boulder, but it thereby turns into a boulder-mass RP which cannot collide with boulders anymore by choice of the collision rate function in Eq. (3.60). For a collision rate $\tilde{\lambda}_\mathrm{g}$, the probability that a given grain-mass RP has suffered a collision with a boulder during time Δt is given by the exponential distribution

$$P(\tilde{\lambda}_\mathrm{g} \Delta t) = 1 - e^{-\tilde{\lambda}_\mathrm{g} \Delta t} \ . \tag{3.67}$$

The expected number of grains at time $t + \Delta t$ is therefore

$$\begin{aligned}\mathrm{E}\big[N'_\mathrm{g}\big] &= \sum_{i \in \mathcal{I}_\mathrm{g}} e^{-\tilde{\lambda}_\mathrm{g} \Delta t} N_i \\ &= N_\mathrm{g} - \tilde{\lambda}_\mathrm{g} N_\mathrm{B} \Delta t + \mathcal{O}(\Delta t^2) \ . \end{aligned} \tag{3.68}$$

In the limit of $\Delta t \to 0$, this yields

$$\frac{\mathrm{d}N_\mathrm{g}}{\mathrm{d}t} = \lim_{\Delta t \to 0} \frac{\mathrm{E}\big[N'_\mathrm{g}\big] - N_\mathrm{g}}{\Delta t} = -\tilde{\lambda}_\mathrm{g} N_\mathrm{B} \ , \tag{3.69}$$

which is equivalent to Eq. (3.112) of the analytical model.

The RPMC method keeps swarm masses constant, $M'_j = M_j$. In the swarms that already had boulder mass at time t, the number of boulders must therefore decrease over a duration Δt:

$$\begin{aligned}\mathrm{E}\big[N'_{\mathrm{B}\to\mathrm{B}}\big] &= \sum_{j \in \mathcal{I}_\mathrm{B}} \mathrm{E}\left[\frac{M'_j}{m'_j}\right] \\ &= \sum_{j \in \mathcal{I}_\mathrm{B}} M_j \mathrm{E}\big[(m'_j)^{-1}\big] \ . \end{aligned} \tag{3.70}$$

Growth of boulders is a Poisson point process, so for a boulder-mass RP $j \in \mathcal{I}_\mathrm{B}$ with mass m_j at time t we have

$$\begin{aligned}\mathrm{E}\big[(m_j')^{-1}\big] &= \sum_{k=0}^{\infty} \frac{\mathrm{Pois}_k(\tilde{\lambda}_{\mathrm{B},j}\Delta t)}{m_j + k m_\mathrm{g}} \\ &= \frac{1}{m_j}\left(1 - \tilde{\lambda}_{\mathrm{B},j}\Delta t \frac{m_\mathrm{g}}{m_j + m_\mathrm{g}}\right) + \mathcal{O}(\Delta t^2)\ , \end{aligned} \tag{3.71}$$

and with $M_j = N_j m_j$, Eq. (3.70) evaluates to

$$\begin{aligned}\mathrm{E}\big[N'_{\mathrm{B}\to\mathrm{B}}\big] &= \sum_{j\in\mathcal{I}_\mathrm{B}} N_j \left(1 - \tilde{\lambda}_{\mathrm{B},j}\Delta t \frac{m_\mathrm{g}}{m_j + m_\mathrm{g}} + \mathcal{O}(\Delta t^2)\right) \\ &= N_\mathrm{B} - N_\mathrm{g} m_\mathrm{g} \Delta t \sum_{j\in\mathcal{I}_\mathrm{B}} \frac{N_j \Lambda_j}{m_j + m_\mathrm{g}} + \mathcal{O}(\Delta t^2)\ . \end{aligned} \tag{3.72}$$

At the same time, some of the grain-mass swarms become boulder-mass swarms after their RP hits a boulder and becomes part of it. The chance for a grain-mass RP $i \in \mathcal{I}_\mathrm{g}$ to collide with a boulder is again given by the exponential distribution given in Eq. (3.67),

$$P(i) \equiv P(\tilde{\lambda}_\mathrm{g}\Delta t) = \tilde{\lambda}_\mathrm{g}\Delta t + \mathcal{O}(\Delta t^2)\ . \tag{3.73}$$

Given that RP i undergoes a collision with a boulder, we now need to determine which boulder will be its collision partner. Because the collision rate is independent of the boulder mass, every boulder has the same chance of colliding with the given grain-mass RP i. The chance $P(j|i)$ that RP i collides with a boulder from swarm $j \in \mathcal{I}_\mathrm{B}$ is therefore given by the relative collision rate of a given grain with swarm j,

$$P(j|i) \equiv \frac{\tilde{\lambda}_{\mathrm{g}j}}{\tilde{\lambda}_\mathrm{g}} = \frac{N_j \Lambda_j}{N_\mathrm{B}\langle\Lambda\rangle_\mathrm{B}}\ . \tag{3.74}$$

If RP i collides with a boulder from swarm j, its entire swarm is converted to a swarm of boulders. The swarm mass remains the same, $M_i' = M_i$, while the grain RP sticks to the boulder and thereby grows to the mass $m_\mathrm{g} + m_j'$ by time $t + \Delta t$. The number of boulders in the new swarm will thus be

$$\begin{aligned}\mathrm{E}\big[N_i'\big] &= \mathrm{E}\left[\frac{M_i'}{m_\mathrm{g} + m_j'}\right] = M_i \mathrm{E}\big[(m_\mathrm{g} + m_j')^{-1}\big] \\ &= \frac{M_i}{m_\mathrm{g} + m_j} + \mathcal{O}(\Delta t)\ . \end{aligned} \tag{3.75}$$

The total number of boulders in the swarms which were composed of grains at time t but have been converted to boulders by time $t + \Delta t$ is therefore

$$\begin{aligned}\mathrm{E}\big[N'_{\mathrm{g}\to\mathrm{B}}\big] &= \sum_{i\in\mathcal{I}_\mathrm{g}} P(i) \sum_{j\in\mathcal{I}_\mathrm{B}} P(j|i)\mathrm{E}\big[N_i'\big] \\ &= \Delta t \sum_{i\in\mathcal{I}_\mathrm{g}} \sum_{j\in\mathcal{I}_\mathrm{B}} N_j \frac{M_i \Lambda_j}{m_j + m_\mathrm{g}} + \mathcal{O}(\Delta t^2)\ . \end{aligned} \tag{3.76}$$

Noting that

$$\sum_{i\in\mathcal{I}_\mathrm{g}} M_i = N_\mathrm{g} m_\mathrm{g} \ , \tag{3.77}$$

we find

$$\mathrm{E}\big[N'_{\mathrm{g}\to\mathrm{B}}\big] = N_\mathrm{g} m_\mathrm{g} \Delta t \sum_{j\in\mathcal{I}_\mathrm{B}} \frac{N_j \Lambda_j}{m_j + m_\mathrm{g}} + \mathcal{O}(\Delta t^2) \ . \tag{3.78}$$

We then obtain the expected total number of boulders at time $t + \Delta t$ by summing up the two contributions in Eqs. (3.72) and (3.78). The contributions cancel exactly to first order in Δt:

$$\begin{aligned} \mathrm{E}\big[N'_\mathrm{B}\big] &= \mathrm{E}\big[N'_{\mathrm{B}\to\mathrm{B}}\big] + \mathrm{E}\big[N'_{\mathrm{g}\to\mathrm{B}}\big] \\ &\overset{!}{=} N_\mathrm{B} + \mathcal{O}(\Delta t^2) \ . \end{aligned} \tag{3.79}$$

The RPMC method can thus be expected to keep the number of boulders N_B invariant,

$$\frac{\mathrm{d}N_\mathrm{B}}{\mathrm{d}t} = \lim_{\Delta t\to 0} \frac{\mathrm{E}[N'_\mathrm{B}] - N_\mathrm{B}}{\Delta t} \overset{!}{=} 0 \ . \tag{3.80}$$

This result demonstrates in the RP–swarm picture that the two paradoxical properties exactly cancel, and that the RPMC method is statistically correct. It is worth pointing out that not only the totality of $\mathrm{d}N_\mathrm{B}/\mathrm{d}t$ vanishes, but for every individual $j \in \mathcal{I}_\mathrm{B}$, the corresponding summands in Eqs. (3.72) + (3.78) cancel out. In other words, every swarm of boulders maintains its own statistical balance with the totality of grains.

3.5.2 Mutating swarm masses

As pointed out in Sect. 3.4.1, it is reasonable to have each RP i represent approximately the same relative mass $M_i \approx M/n$, where M is the total mass in the system. But our formal definition of the RPMC method does not impose any assumptions on the values of swarm masses M_i. This is crucial because the extension of the method presented in Sect. 3.4.3 transfers mass between swarms. Specifically, in an interaction that operates in the few-particles regime, a few-particles swarm – usually a single-particle swarm – may remove mass from a many-particles swarm. In the following we argue that this does not affect the statistical balance of the method.

It should first be noted that the statistical balance only concerns many-particles swarms, as only interactions between many-particles swarms operate in the asymmetric many-particles regime. To incorporate the effect of mass transfers, we admit that the mass M_i of any many-particles swarm i may be changed by an external source or sink modelled as

$$\frac{\mathrm{d}M_i}{\mathrm{d}t} = M_i f(m_i, t) \ , \tag{3.81}$$

where the continuous function $f(m, t)$ is the expected relative inflow/outflow rate of particles of mass m at time t. We justify this approach by arguing that, if k is a many-particles swarm and j is a single-particle swarm, we have $N_j = 1$ and $N_k > N_\mathrm{th}$, hence

$N_j \ll N_k$, and therefore the change $N_k \to N_k - 1$ is small enough to be modelled as a continuous process.

With the source or sink model of Eq. (3.81), swarm mass M_i' at time $t + \Delta t$ can be expanded as

$$\mathrm{E}\big[M_i'\big] = M_i + f(m_i, t)\Delta t \ . \tag{3.82}$$

This leads to a first-order deviation from N_g and N_B during time increment Δt,

$$\mathrm{E}\big[N_\mathrm{g}'\big] = N_\mathrm{g} + N_\mathrm{g}\left(f_\mathrm{g} - \tilde{\lambda}_\mathrm{g}\right)\Delta t + \mathcal{O}(\Delta t^2) \ , \tag{3.83}$$

$$\mathrm{E}\big[N_\mathrm{B}'\big] = N_\mathrm{B} + N_\mathrm{B}\left\langle f\right\rangle_\mathrm{B}\Delta t + \mathcal{O}(\Delta t^2) \ , \tag{3.84}$$

where the grain inflow rate f_g and the mean boulder inflow rate $\left\langle f\right\rangle_\mathrm{B}$ are defined as

$$f_\mathrm{g} \equiv f(m_\mathrm{g}, t) \ , \quad \left\langle f\right\rangle_\mathrm{B} \equiv \frac{1}{N_\mathrm{B}} \sum_{j \in \mathcal{I}_\mathrm{B}} N_j f(m_j, t) \ . \tag{3.85}$$

Any influence of inflow or outflow on the mass transfer terms in Eqs. (3.72) and (3.76) is of second order in Δt and hence does not upset the balance of Eq. (3.80). We also find

$$\frac{\mathrm{d}N_\mathrm{g}}{\mathrm{d}t} = N_\mathrm{g}\left(f_\mathrm{g} - \tilde{\lambda}_\mathrm{g}\right) \ , \tag{3.86}$$

$$\frac{\mathrm{d}N_\mathrm{B}}{\mathrm{d}t} = N_\mathrm{B}\left\langle f\right\rangle_\mathrm{B} \ , \tag{3.87}$$

which is equivalent to the result found by adding the source or sink term to the analytical model discussed in Appendix 3.B.

3.6 Numerical tests

To assess the correctness and applicability of the improved RPMC method, we run a series of numerical tests which can be grouped in three parts.

In Sect. 3.6.1, we first simulate two well-known coagulation kernels for which analytical solutions are available, the constant and the linear kernel, leading to self-similar mass distributions. Equivalent tests were already conducted in Zsom and Dullemond (2008); we additionally verify that unequal samplings of the mass leave the result unimpaired as claimed in Sect. 3.4.1.

Second, in Sect. 3.6.2 we study the transition to the few-particles regime and the simulation of runaway growth. A full-scale non-representative Monte Carlo simulation of the constant kernel is compared to a simulation with the extended RPMC method. We also study runaway growth by simulating the product kernel, a test kernel with an analytical solution available which develops a very sharp runaway characteristic. Third, we test a runaway kernel that mimics gravitational focussing, finding no effect of the simulation parameters on the evolution of the mass distribution as long as the resolution criterion in Eq. (3.39) is met.

Third, in Sect. 3.6.3 we adopt two variants of the grains-and-boulders example given by Zsom and Dullemond (2008) for which analytical predictions can be made. We study how well the RPMC method retains the statistical balance depending on the number of RPs used.

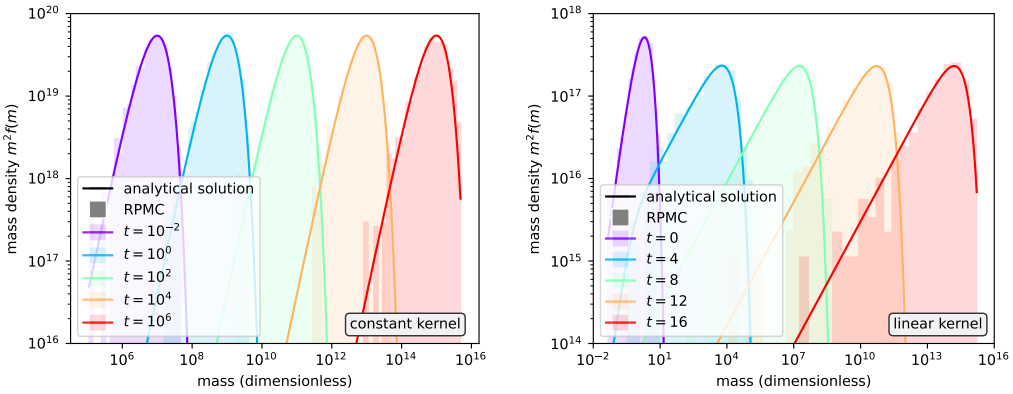


Figure 3.2: Analytical solutions and RPMC simulations for standard coagulation tests: (a) constant kernel $\lambda(m, m') = \Lambda = \text{const}$; (b) linear kernel $\lambda(m, m') = \Lambda_0 (m + m')$.
The RPMC simulation runs use $n = 1\,024$ RPs.

3.6.1 Self-similar solutions

The *constant kernel*

$$\lambda(m, m') = \Lambda = \text{const} \tag{3.88}$$

and the *linear kernel*

$$\lambda(m, m') = \Lambda_0 (m + m') \tag{3.89}$$

are well-known special cases of the Smoluchowski equation Smoluchowski (1916)

$$\begin{aligned} \frac{\partial}{\partial t} f(m,t) = &-f(m,t) \int_0^\infty \mathrm{d}m' \lambda(m,m') f(m',t) \\ &+ \frac{1}{2} \int_0^m \mathrm{d}m' \lambda(m', m-m') f(m',t) f(m-m',t) \,, \end{aligned} \tag{3.90}$$

an integrodifferential equation for the number density distribution $f(m,t)$ which describes coagulation processes in a continuous mass distribution. The first term represents the loss of particles at mass m due to coagulation with other particles, while the second term describes the gain of particles by coagulation of masses $m' + (m - m') = m$.

The results for a standard equal-mass sampling, $M_k = M/n \forall k$, are shown in Fig. 3.2, finding very good agreement with the analytical solutions, which are summarised in Appendix 3.A. The same tests were re-run with heterogeneous swarm mass samplings. For the constant kernel, swarm masses were drawn from the log-normal swarm number density distribution

$$f(M') \propto \exp\left[-\frac{(\log M' - \mu)^2}{2\sigma^2} \right] \,, \tag{3.91}$$

where we identified $\sigma \equiv \mu/20$ and chose μ such that the mean of the distribution is $\exp(\mu + \sigma^2/2) = M/n$. The samples M_k were normalised such that $\sum_k M_k = M =$

10^{20} held exactly. A log-space distribution was chosen because we wanted to assess the resiliency of the method not only to small variations in swarm mass; with the log-normal distribution given, swarm masses span several orders of magnitude. For the linear kernel test, we sampled the RP masses m_k from the initial particle number density distribution $f(m, 0)$ in Eq. (3.100) and then chose a constant swarm particle count $N_k \equiv \mathcal{N}\,\forall k$ such that $\sum_k m_k N_k = \mathcal{N}\sum_k m_k = M = 10^{18}$: instead of an equal-mass sampling, we chose an equal-count sampling of swarm masses. While this was done mainly for reasons of simplicity, the resulting swarm mass distribution also spans a few orders of magnitude. Both simulations yielded results not significantly different from those obtained with equal-mass samplings, demonstrating that the RPMC method works well despite considerable variations of swarm masses. The results are not shown separately because they are visually indiscernible from those in Fig. 3.2.

3.6.2 Regime transition and runaway growth

To quantify how well the regime transition works, we study three scenarios. First, we again run a simulation with the constant kernel in Eq. (3.88), but we let it run until all mass is concentrated in a single particle. We note that the analytical solution in Eq. (3.99) is not applicable in this case because it is a solution to the continuous Smoluchowski equation (Eq. (3.90)), which approximates discrete coagulation processes only for particle number densities much greater than unity, $f(m, t) \gg 1$. Instead, we run a fiducial simulation of $n = 10^8$ RPs with swarm particle counts $N_k = 1\forall k$ – that is, every RP only represents itself – and compare this to a regular RPMC simulation with $n = 1\,024$ RPs that transition into the few-particles regime as they grow. As can be seen in Fig. 3.3, the two simulation runs are in good agreement.

Next, we study the *product kernel*

$$\lambda(m, m') = \Lambda_0 m m' \,. \tag{3.92}$$

The coagulation process produces a runaway particle at dimensionless time $\eta = 1$. The analytical solution for the product kernel is discussed in Appendix 3.A.

In Fig. 3.4 we compare the particle number density of the analytical solution in Eq. (3.107) to an RPMC simulation with $n = 2\,048$ RPs, finding good agreement even in the runaway regime at $\eta > 1$. The figure can be directly compared to Fig. 5 in Ormel and Spaans (2008). Fig. 3.5, which is modelled after Fig. 2 in Wetherill (1990), shows the analytical prediction for the number of particles, the mass of small bodies, and the mass of the runaway particle. The runaway particle mass produced by the RPMC simulation is also shown and in good agreement with the prediction.

In our third test of runaway growth, we use a collision kernel inspired by gravitational focussing (cf. e.g. Armitage, 2017, §III.B.1): ,

$$\lambda(m, m') = \Lambda_0 \max\{m, m'\}^{2/3} \left[1 + \left(\frac{\max\{m, m'\}}{m_{\rm th}}\right)^{2/3}\right] \tag{3.93}$$

with some constant coefficient Λ_0 and a threshold mass $m_{\rm th}$. The collision rate therefore scales with $m^{2/3}$ for masses below $m_{\rm th}$ and with $m^{4/3}$ for masses above, which implies

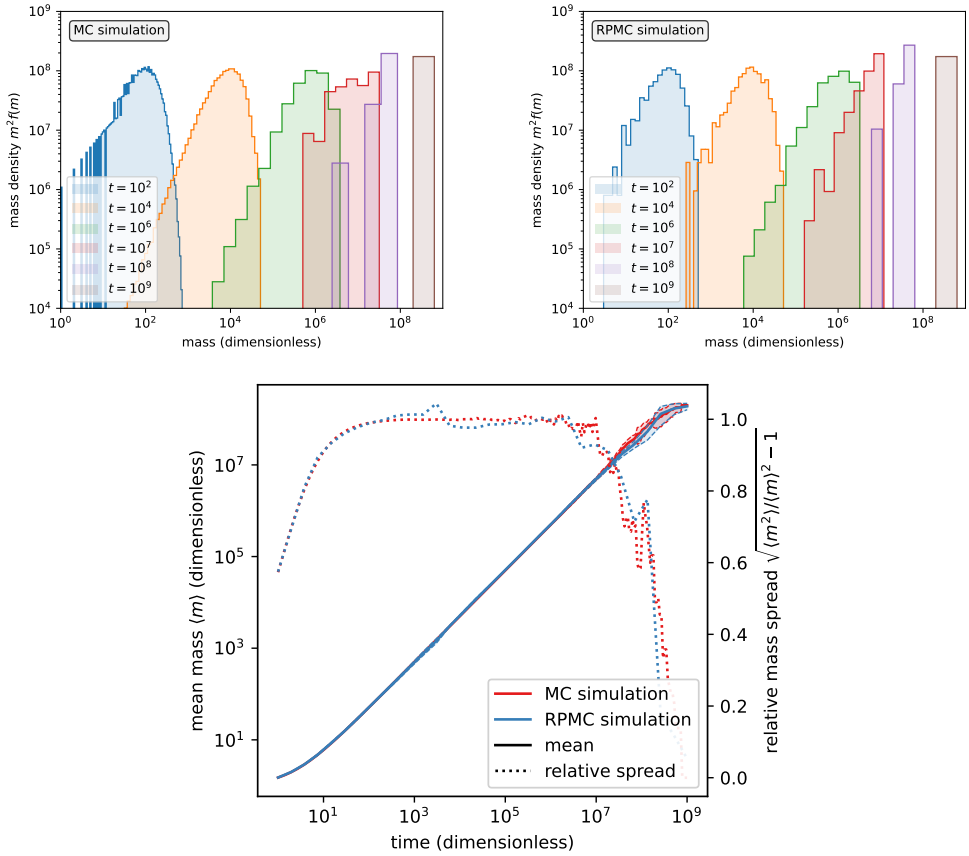


Figure 3.3: Transition into the few-particles regime for the constant kernel Eq. (3.88): (a) Fiducial non-representative Monte Carlo simulation with $n = 10^8$ particles; (b) RPMC simulation with $n = 1\,024$ RPs; (c) mean mass $\langle m \rangle$ and relative mass spread $\sqrt{\langle m^2 \rangle / \langle m \rangle^2 - 1}$ of both simulations in comparison.
The discontinuities in (a) at $t = 10^2$ are an aliasing effect caused by histogram binning. (a) and (b) show snapshots of a single simulation, while (c) shows results averaged over 10 runs, with statistical error bounds indicated the by dashed curves.

that particles above the threshold mass sweep up mass much more efficiently. As with any power-law collision rate, we expect the initial mass distribution to relax to a self-similar state while in the $\propto m^{2/3}$ regime. As the largest particles approach the threshold mass, we expect them to start 'running away', making the mass distribution bimodal. Finally, the most massive particle will end up accumulating all the remaining mass.

Results are given in Fig. 3.6, confirming our expectations: the initial distribution quickly becomes self-similar and stays that way until $t \sim 30$. After the higher-mass end of the distribution exceeds the threshold mass $m_{\text{th}} = 10^7$, a second peak starts to form, clearly visible at $t = 50$. At $t = 60$ the runaway particles are already depleting the tail end of the mass distribution, and almost all mass has concentrated in a single particle

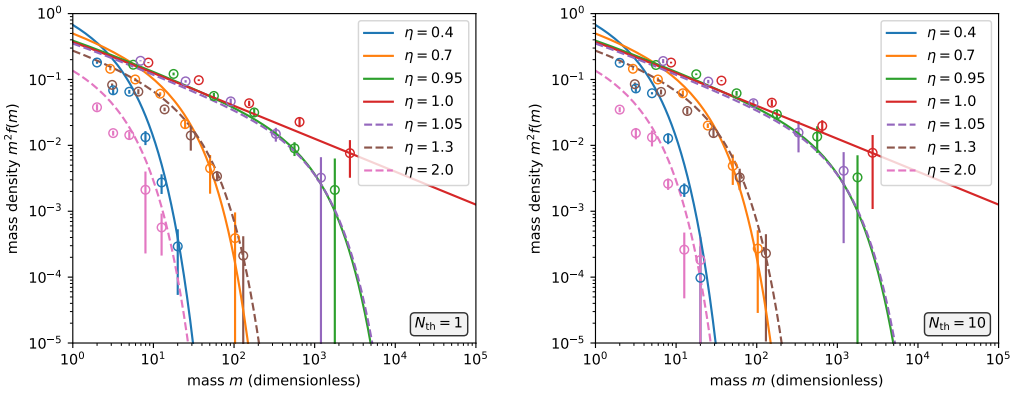


Figure 3.4: Particle mass density at different times for the product kernel coagulation test, compared to the analytical solution in Eq. (3.107) (solid curves).
The RPMC simulation uses $n = 2\,048$ particles and different values of the particle number threshold: (a) $N_{\text{th}} = 1$; (b) $N_{\text{th}} = 10$. The results shown were averaged over 5 runs, with a standard deviation indicated by the error bars.

by $t = 80$. We find no notable sensitivity to the number of RPs as long as the resolution criterion Eq. (3.39) is satisfied.

3.6.3 Grains and boulders

We simulate the grains-and-boulders example introduced and analysed in Sect. 3.5 to validate the results and to confirm that statistical balance is indeed retained. We first consider a mass-independent mutual collision rate,

$$\Lambda(m, m') = \Lambda_0 \tag{3.94}$$

with some constant Λ_0. An analytical prediction for the expected mean boulder mass $\langle m \rangle_{\text{B}}$ and its standard deviation $\sigma_{m_{\text{B}}}$ is developed in Appendix 3.B. It is shown for different initial conditions in Fig. 3.7. The model was then simulated with the RPMC method for the same set of initial conditions; results are given in Fig. 3.8. We observe very good agreement with the analytical prediction.

Boulders remain boulders after a hit-and-stick collision with a grain. The number of boulders therefore cannot change, $\mathrm{d}N_{\text{B}}/\mathrm{d}t = 0$, and thus

$$N_{\text{B}} = N_{\text{B}}^0 \,. \tag{3.95}$$

When simulating an ensemble of grains and boulders with the RPMC method, the number of boulders is

$$N_{\text{B}} = \sum_{\substack{j=1 \\ m_j \geqslant m_{\text{B}}^0}}^{n} \frac{M_j}{m_j} \,. \tag{3.96}$$

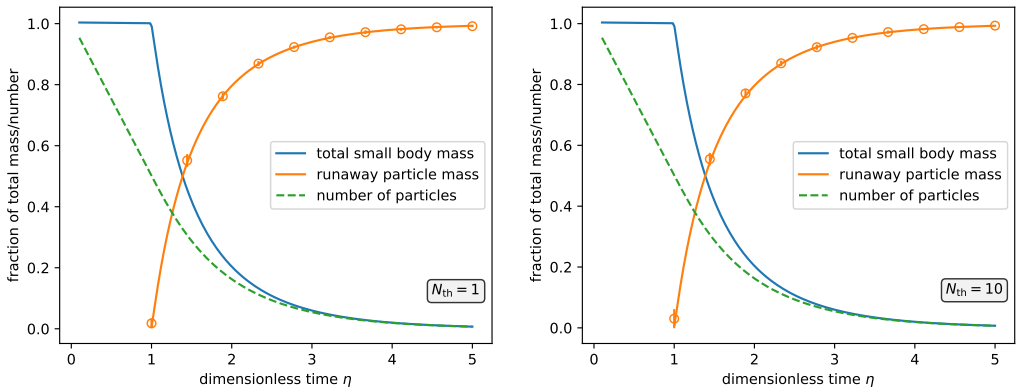


Figure 3.5: Fraction of particle mass in the runaway particle for the product kernel coagulation test, compared to the analytical solution in Eq. (3.107) (solid curves).
The RPMC simulation uses $n = 2\,048$ particles and different values of the particle number threshold: (a) $N_{\mathrm{th}} = 1$; (b) $N_{\mathrm{th}} = 10$. The results shown were averaged over 5 runs, with a standard deviation indicated by the error bars.

If the simulation operates in the many-particles regime, the swarm masses M_j do not change. Therefore, as boulder-mass RPs grow, the number of boulders per swarm $N_j = M_j/m_j$ decreases. At the same time, some grain-mass RPs collide with a boulder and therefore become boulder-mass RPs. Both effects contribute to the sum in Eq. (3.96), and as argued before we expect them to cancel exactly.

Fig. 3.9a visualises this statistical balance. As can be seen, the two contributions operate on slightly different timescales. It is more likely for a boulder to hit a grain than for a grain to hit a boulder; but when a boulder-mass RP accumulates a grain, it just grows a little and thus represents slightly fewer boulders than before, whereas a grain-mass RP that hits a boulder will instantly convert its entire swarm to boulders, causing a surge in total boulder count. We can observe the relative smoothness of the particle number decay as well as the more erratic particle number growth in Fig. 3.9a.

We naturally expect the resolution of the simulation to improve if we add more RPs. The influence of the number of RPs on the accuracy of the result is studied in Fig. 3.9b for different RP counts. The statistical precision of the final boulder mass is estimated as the standard deviation of the final masses in repeated runs of the simulation and shown as error bars around the mean value. We find it to be insensitive to the particle mass ratio $m_{\mathrm{B}}^0/m_{\mathrm{g}}$. However, the physical spread of the mass distribution, whose standard deviation is visualised as a filled area, does depend on the particle masses and the total masses as predicted by Eq. (3.124),

$$\frac{\sigma_{m_{\mathrm{B}}}^{\infty}}{m_{\mathrm{B}}^0} \equiv \lim_{t\to\infty} \frac{\sigma_{m_{\mathrm{B}}}}{m_{\mathrm{B}}^0} = \left(\frac{M_{\mathrm{B}}^0}{M_{\mathrm{g}}^0}\frac{m_{\mathrm{B}}^0}{m_{\mathrm{g}}}\right)^{-1/2} . \tag{3.97}$$

The larger the ratios $M_{\mathrm{B}}^0/M_{\mathrm{g}}^0$ and $m_{\mathrm{B}}^0/m_{\mathrm{g}}$, the smaller the physical spread $\sigma_{m_{\mathrm{B}}}$.

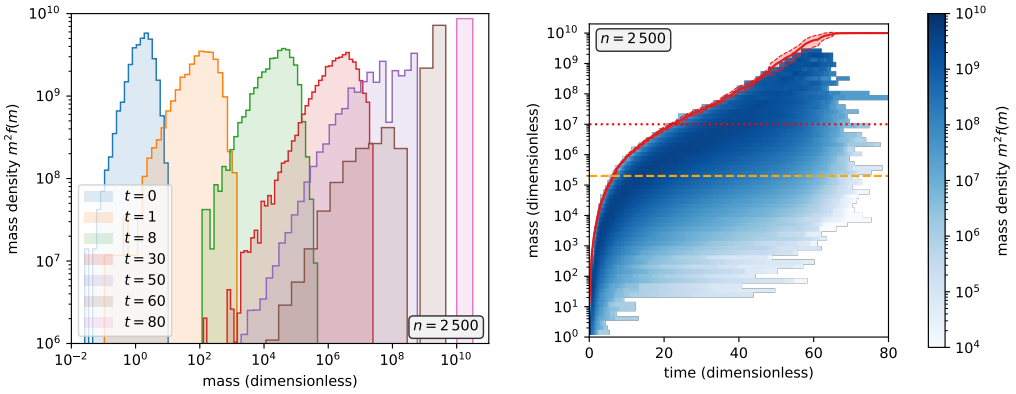


Figure 3.6: RPMC simulation with the runaway collision rate in Eq. (3.93), a dimensionless threshold mass of $m_{\mathrm{th}} = 10^7$ (indicated by dotted red horizontal line in (b)), a total mass of $M = 10^{10}$, a homogeneous swarm mass distribution, a particle number threshold of $N_{\mathrm{th}} = 20$, and $n = 2\,500$ RPs. Every individual swarm i has mass $M_i = M/n$; particles are resolved individually once they grow beyond the mass $M/(nN_{\mathrm{th}})$ (indicated by the dashed orange horizontal line in (b)). Each panel shows a time series of histograms. The mass-weighted particle number density is shown on the vertical axis of (a), whereas in (b) the mass-weighted histogram bin counts are colour-encoded on a logarithmic scale, and the mass of the runaway particle is shown separately (red curve). Histogram bin counts and mass of runaway particle are averaged over 10 runs, the error bounds of the runaway particle being indicated by red dashed curves.

Next, we consider a mutual collision rate linearly correlated to mass,

$$\Lambda(m, m') = \Lambda_0 \left(m + m'\right) \tag{3.98}$$

with some constant Λ_0. Fig. 3.10 compares the analytical prediction of the average boulder mass $\langle m \rangle_{\mathrm{B}}$ and its standard deviation $\sigma_{m_{\mathrm{B}}}$ derived in Appendix 3.B to an equivalent RPMC simulation, finding very good agreement.

3.7 Discussion

The proposed extensions to the RPMC method are strictly additive in the sense that they do not change how the method operates in the many-particles regime to which the original RPMC method was constrained. As such, the considerations of Zsom and Dullemond (2008, §3.2) with regard to the required number of representative particles continue to apply. In §3.3 of the same paper, two main limitations of the method are discussed. The first limitation – that swarms must consist of many particles – is overcome by our extension, but the other – that, as an explicit method, the RPMC method is ill-suited for simulating 'stiff' problems in which two adverse effects nearly balance each other – remains and is in fact aggravated by our extension of the method. When following the traditional implementation strategy, the variability of the number of repre-

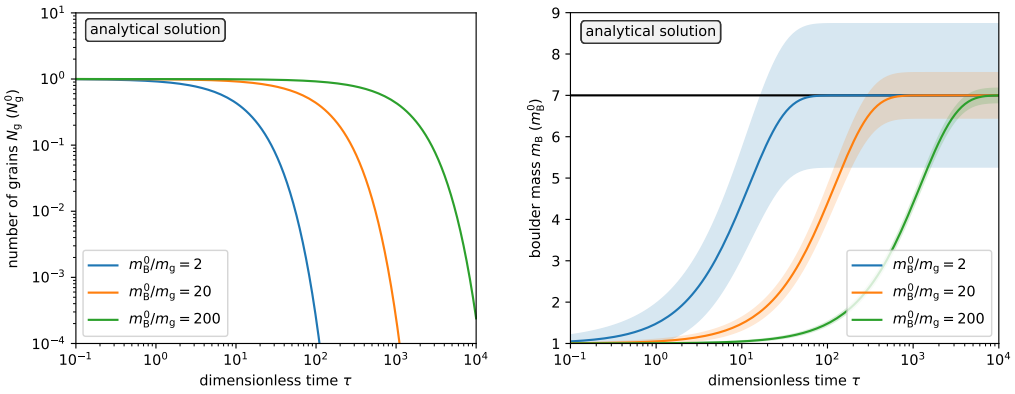


Figure 3.7: Analytical solution given in Eqs. (3.122, 3.118) to the grains-and-boulders model introduced in Sect. 3.5 for different boulder–grain particle mass ratios: (a) total number of grains N_g; (b) expected boulder mass $\langle m \rangle_B$. The total mass ratio of grains and boulders is $M_B^0/M_g^0 = 1/6$, so boulders are expected to grow to 7 times their initial mass, as indicated by the horizontal black line. The horizontal axes refer to dimensionless time $\tau = \Lambda N_g^0 t$. In (b), the filled area indicates the physical spread σ_{m_B} of the mass distribution given by Eq. (3.124).

sentative particles in the extended method adds substantially to the computational cost; however, in Chapter 4 we devise a different implementation strategy which is much less sensitive to the number of RPs.

3.7.1 Limitations

The original RPMC method was not well-suited to simulate systems in which two strong effects nearly balance each other, leading only to a small net change. As an example, Zsom and Dullemond (2008, §3.3) suggests a system in which coagulation and fragmentation processes lead to a 'semi-steady state' in which particles grow and fragment many times, and explains that, as an explicit method, the RPMC method must simulate the individual growth and fragmentation processes, rendering the problem 'stiff'. This limitation is not removed by our extension of the method, and can in fact be further aggravated by it: If a RP in such a system grows large enough that its swarm particle count falls below the particle number threshold N_{th}, it will be split up to individual RPs. If all of these RPs subsequently fragment to small particles which then recommence the growth cycle, the number of RPs will steadily grow while the particle number distribution remains stable. With an ever-increasing number of RPs, the simulation will quickly exceed its computational resources.

The root of this problem is an asymmetry in our extension of the method: we split up swarms as their particle mass grows larger, but we never merge multiple swarms into one. Unlike splitting, which can be done without consequences for the statistical outcome of the simulation, merging usually entails a loss of information because the particle properties $\mathbf{q}_j$, $\mathbf{q}_k$ of two swarms j, k to be merged will rarely be exactly identi-

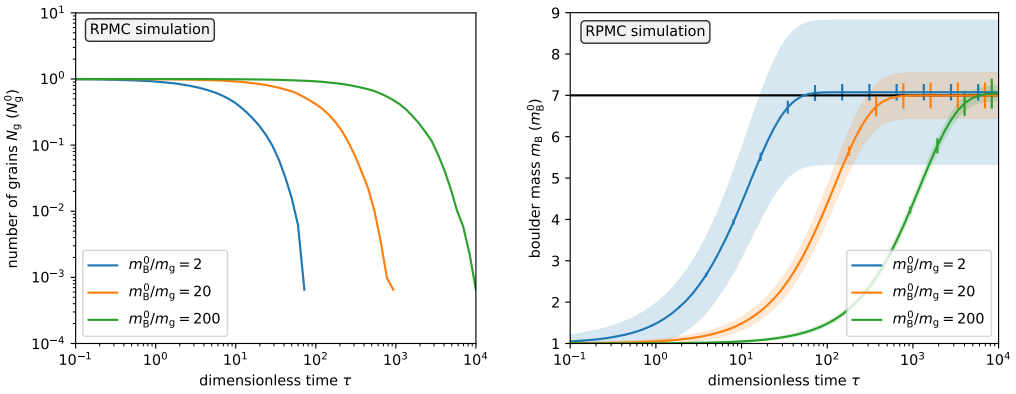


Figure 3.8: RPMC simulation of the grains-and-boulders model introduced in Sect. 3.5 for different boulder–grain particle mass ratios for comparison with Fig. 3.7: (a) total number of grains N_g; (b) average boulder mass $\langle m \rangle_\mathrm{B}$. The simulation uses $n = 1\,792$ RPs divided into $n_\mathrm{B} = 256$ boulder-mass RPs and $n_\mathrm{g} = 6n_\mathrm{B} = 1\,536$ grain-mass RPs with equal-weight swarms, $M_i = M_j \,\forall i, j \in \{1, \ldots, n\}$, and an initial number of $N_\mathrm{g}^0 = 10^8$ grains. The boulder masses in (b) are averaged over 10 consecutive runs; the filled area indicates the physical spread σ_{m_B} of the mass distribution, and the error bars indicate the standard deviation of the mean boulder mass over 10 runs. The horizontal black line indicates the expected final boulder mass $m_\mathrm{B}^\infty = 7m_\mathrm{B}^0$.

cal, and hence have to be averaged over in some way. A merging step was omitted for simplicity, and because it was deemed unnecessary for our main goal in extending the method, which is to simulate planetesimal growth to and beyond the runaway growth regime. However, there is nothing fundamentally precluding a merging step in the simulation, and such an addition would indeed be necessary before the method can be used to simulate semi-steady processes (which would however retain their stiffness even with merging). Merging was an integral part of the Monte Carlo method of Ormel and Spaans (2008); in this method, different 'species' (the semantic equivalent of swarms in the RPMC method) could be merged by averaging their properties. A concrete example of such a merging prescription is given in Krijt et al. (2015, §3.4). A similar approach could in principle be incorporated into the RPMC method.

3.7.2 Computational cost

It was mentioned that a growing number of RPs may significantly increase the computational cost of the simulation. In the traditional implementation described in Zsom and Dullemond (2008), n^2 RP–swarm interaction rates must be computed, stored, and updated for n RPs. Each time a swarm is split up into N_th individual RPs, this cost increases accordingly. Chapter 4 explores an alternative sampling method which avoids computing all n^2 RP–swarm interaction rates. Using a bucketing scheme, this alternative sampling method makes the computational cost less sensitive to the number of RPs

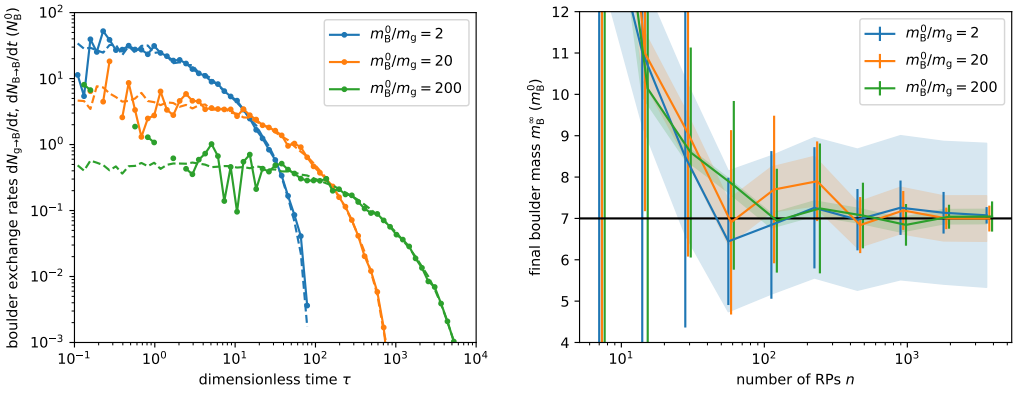


Figure 3.9: Further analysis of the RPMC simulation of the grains-and-boulders model: (a) contributions to $\mathrm{d}N_\mathrm{B}/\mathrm{d}t$: $\mathrm{d}N_{\mathrm{g}\to\mathrm{B}}/\mathrm{d}t$ (solid curve), $-\mathrm{d}N_{\mathrm{B}\to\mathrm{B}}/\mathrm{d}t$ (dashed curve); (b) statistics of final boulder mass m_B^∞ as a function of RP count n: standard deviation of the mean over 10 runs (error bars) and physical spread σ_{m_B} (filled area).
Simulation parameters as in Fig. 3.8. In (a), the full $n = 1\,792$ RPs were used; in (b), the number of RPs was varied. n was divided into n_B boulder-mass RPs and $n_\mathrm{g} = 6n_\mathrm{B}$ grain-mass RPs with equal-weight swarms, $M_i = M_j \forall i, j \in \{1, \ldots, n\}$. The horizontal black line indicates the expected final boulder mass $m_\mathrm{B}^\infty = 7m_\mathrm{B}^0$.

n and more dependent on the breadth of the occupied subspace of the parameter space $\mathbb{Q}$, and thereby makes the improved RPMC method computationally viable.

3.7.3 Boosting

In Sect. 3.4.4, a boosting scheme was proposed to ensure efficient simulation of growth processes where large bodies accrete large amounts of small bodies. Although an intuitive justification for the approach was given, it remained unclear how it impacted the statistical result of the simulation. This impact can be studied by means of the grains-and-boulders example:

In Sect. 3.6.3, we studied the influence of the boulder–grain mass ratio $m_\mathrm{B}^0/m_\mathrm{g}$ on the simulation outcome. We predicted (cf. Fig. 3.7) and confirmed numerically (cf. Fig. 3.8) a spread of the final boulder mass that was anticorrelated with the boulder–grain mass ratio $m_\mathrm{B}^0/m_\mathrm{g}$. Now, a boost factor $\beta_{jk} > 1$ would be equivalent to accreting more massive grains with a lower collision rate, and the effect of boosting is therefore quantitatively similar to the spread of the final boulder mass distribution in Eq. (3.97). Although the spread of the mass distribution increases as the boulder–grain mass ratio is increased, it was found that the fidelity of the RPMC method, quantified by the uncertainty of the expected final boulder mass, did not depend on the mass ratio (cf. Fig. 3.8b). Analogously, boosting seems justifiable provided that the spread of the mass distribution incurred does not exceed the desired precision of the result. This requirement may be significant when dealing with sharply peaked distributions as in the grains-and-

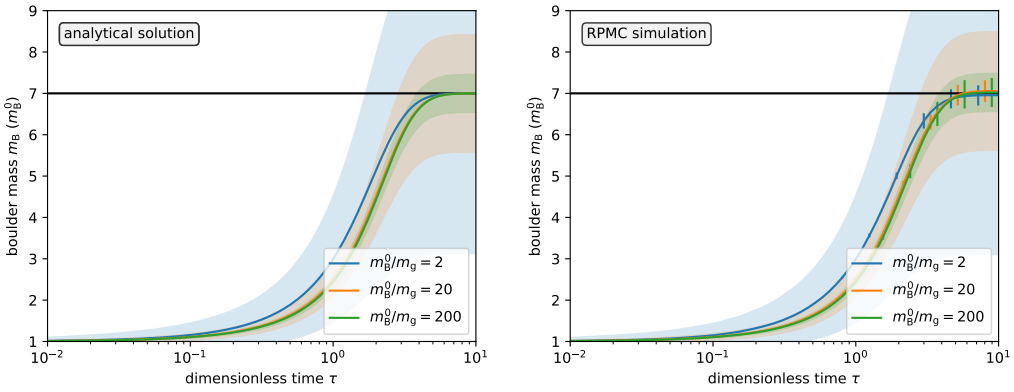


Figure 3.10: Average boulder mass $\langle m \rangle_B$ and the physical spread σ_{m_B} for a mass-dependent collision rate $\Lambda(m, m') = \Lambda_0 (m + m')$ for different boulder–grain particle mass ratios: (a) analytical model as given by Eqs. (3.126, 3.128); (b) RPMC simulation.
The simulation uses $n = 1\,792$ RPs divided into $n_B = 256$ boulder-mass RPs and $n_g = 6n_B = 1\,536$ grain-mass RPs with equal-weight swarms, $M_i = M_j \forall i, j \in \{1, \ldots, n\}$. The boulder masses in (b) are averaged over 10 consecutive runs; the filled area indicates the physical spread σ_{m_B} of the mass distribution, and the error bars indicate the standard deviation of the mean boulder mass over 10 runs. The horizontal black line indicates the expected final boulder mass $m_B^\infty = 7m_B^0$. The total mass ratio of grains and boulders is $M_B^0/M_g^0 = 1/6$, so boulders are expected to grow to 7 times their initial mass, as indicated by the horizontal black line. The horizontal axes refer to dimensionless time $\tau = \Lambda N_g^0 t$.

boulders example, but it becomes much less restricting for ensembles with a broad mass distribution such as the standard coagulation tests (cf. Fig. 3.2) which were conducted with a mass accumulation threshold of $q_m = 5\%$, leading to no significant deviation from analytical predictions.

3.7.4 Avoiding non-integral particle numbers

As mentioned in Sect. 3.4.3, non-integral swarm particle counts N_k pose a challenge when swarms become few-particles swarms and are thus split up to individual RPs. We note that the boost factor implicitly helps mitigating this problem. A many-particles swarm can become a few-particles swarm in two scenarios: (1) by accumulating mass through a collision in the many-particles regime, or (2) by losing mass through a collision in the few-particles regime. Eq. (3.48) constrains the boost factor such that, in scenario (1), the new swarm particle count does not fall below the particle number threshold, $N_j' \geqslant N_{\text{th}}$, thus obtaining a 'precision landing' of $N_j' = N_{\text{th}}$ if permitted by the other constraints. For the few-particles regime, by exhausting the constraint $\beta_{jk} \leqslant N_k/N_j$ given in Eq. (3.49) we can let swarm j deplete swarm k completely by choosing $\beta_{jk} = N_k/N_j$, again if permitted by the other constraints.

3.8 Summary and conclusions

We have presented a method for stochastic simulation of particle–particle interactions. The method builds upon the RPMC method introduced by Zsom and Dullemond (2008). The most severe limitation of the RPMC method was that, due to its asymmetric construction, the growth of a given representative particle was effectively constrained by the amount of mass it represented. Our extension of the method overcomes this growth barrier by allowing the coexistence of, and the transition between, representative particles and individual, self-representing particles. With our method, it is now possible to accurately model runaway growth processes while largely retaining the structural advantages of the RPMC method such as the stability of the sampling weights.

In Sect. 3.2 we gave a formal definition of the RPMC method, which we subsequently extended in Sect. 3.4.3. In Sect. 3.5 we proved for a generalised variant of the grains-and-boulders example given in Zsom and Dullemond (2008) that, despite the asymmetry in the modelling of interactions, the RPMC method is in fact statistically balanced and produces correct results. Finally, in Sect. 3.6 we conducted extensive numerical testing to validate that the extended RPMC method indeed produces plausible results for different cases of orderly growth and runaway growth.

In our extended version of the RPMC method, representative particles are added to the simulation as particles transition between the representative many-particles regime and the individual single-particle regime. We have thereby abandoned a key feature of the original RPMC method: a stable number of representative particles, which was useful because it allowed for predictable performance and simple implementation. In a traditional implementation of the RPMC method, adding new representative particles would introduce substantial computational cost. In Chapter 4 we present an implementation strategy that not only alleviates the cost of adding near-identical representative particles to the simulation but can also overcome the $\mathcal{O}(n^2)$ computational complexity and storage requirements of the traditional method.

With our extension, the RPMC method can be used to study growth processes over a vast dynamic range, extending into the oligarchic regime where individual bodies arise which cannot be treated stochastically. In this regard, it is similar to the Monte Carlo method of Ormel and Spaans (2008); in the few-particles regime, our method indeed faces similar challenges. A possible advantage of our method over other approaches may be that its main benefit, constant resolution through the intrinsic preservation of an equal-mass sampling, is retained. Interactions between many-particle swarms keep swarm masses unchanged; and swarms can lose but not gain mass through interactions with few-particles swarms, which implies that the resolution is never impaired.

With the proposed method, planetary growth processes can now be studied for the entire range of masses up to full-grown planets. Regimes of runaway growth or oligarchic growth, where individual particles accumulate large fractions of the total mass, can be faithfully represented. In addition, the transition to individual particles gives rise to the possibility of embedding a representative particle Monte Carlo simulation in another type of simulation which traces particles individually, for example a gravitational N-body simulation.

The original RPMC method has been used for coagulation problems involving more than just the mass as a particle property, for instance spatial resolution (Drążkowska et al., 2013, 2014) or additional intrinsic particle properties (e.g. Zsom et al., 2010; Krijt and Ciesla, 2016; Krijt et al., 2016; Windmark et al., 2012). The added capability, discussed in this chapter, of transiting to the few-particles regime allows to address a variety of problems involving the growth from pebbles, via planetesimals, to planets; runaway and oligarchic growth, and the subsequent N-body dynamics between the oligarchs, possibly driven by pebble accretion onto these oligarchs; or planetesimal and planet formation in dust rings while incorporating the N-body dynamics of these objects, with the newly formed planets starting to stir the pebbles and planetesimals in the ring.

3.9 Other contributors

The introductory text of Chapter 3, Sects. 3.1, 3.2, 3.3.2, and 3.4.1 as well the initial draft for Fig. 3.1 were largely contributed by Cornelis P. Dullemond.

Appendices

3.A Analytical solutions and equal-weight samplings for standard coagulation tests

For the constant kernel Eq. (3.88) and initial conditions $f(m,0) = N^0 \delta_{\mathrm{D}}(m - m^0)$, the Smoluchowski equation (3.90) has the analytical solution

$$\begin{aligned} f(m,t) &= N^0 g^2 (1-g)^{m/m^0 - 1} \ , \\ g &\equiv \left(1 + \frac{\tau}{2}\right)^{-1} \ , \\ \tau &\equiv M \Lambda t \end{aligned} \tag{3.99}$$

with the total dimensionless mass M (e.g. Ohtsuki et al., 1990).

Sampling an initial set of RPs with equal-weight swarms is easy because the definition of the number density distribution $f(m,0)$ implies that all RPs have the same mass initially. We simply select an ensemble of n RPs with homogeneous particle masses m^0 and swarm masses M/n.

For the linear kernel Eq. (3.89) with an initial number density distribution

$$\begin{aligned} f(m,0) &= N_0 \bar{m}^{-1} e^{-m/\bar{m}} \ , \\ N_0 &\equiv \frac{M}{\bar{m}} \ , \end{aligned} \tag{3.100}$$

and the initial average mass $\bar{m}$, the Smoluchowski equation has the analytical solution

$$\begin{aligned} f(m,t) &= N_0 m^{-1} \frac{g}{\sqrt{1-g}} e^{-m/\bar{m}(2-g)} I_1\left(2\frac{m}{\bar{m}}\sqrt{1-g}\right) \ , \\ g &\equiv e^{-\tau} \ , \\ \tau &\equiv M \Lambda_0 t \end{aligned} \tag{3.101}$$

with the total dimensionless mass M, where $I_1(\cdot)$ is the modified Bessel function of the first kind (e.g. Ohtsuki et al., 1990).

Drawing equal-weight samples for the non-trivial initial mass distribution of the linear kernel test is a bit more involved. Let us assume that some number density distribution $f(m)$ is to be sampled by n RPs of masses m and equal weight

$$M = mN(m) \stackrel{!}{=} \frac{M}{n} \ . \tag{3.102}$$

Because of the equal-weight sampling, masses of RPs must instead be drawn from the *mass-weighted density distribution*

$$f_{\mathrm{R}}(m) = M^{-1} m f(m) \ . \tag{3.103}$$

For the initial number density distribution of the linear kernel test (Eq. (3.100)), the mass-weighted density distribution is

$$f_{\mathrm{R}}(m,0) = \bar{m}^{-1} \frac{m}{\bar{m}} e^{-m/\bar{m}} \ . \tag{3.104}$$

We can then draw RP masses with inverse transform sampling by first computing the cumulative representative number density distribution

$$\begin{aligned} F_{\mathrm{R}}(m,0) &= \int_0^m \mathrm{d}m' f_{\mathrm{RP}}(m,0) \\ &= 1 - \left(\frac{m}{\bar{m}}\right) e^{-m/\bar{m}} \,, \end{aligned} \tag{3.105}$$

then inverting the relation $\xi = F_{\mathrm{R}}(m,0)$,

$$m = -\bar{m}\left[1 + W_{-1}\left(\frac{-1+\xi}{e}\right)\right] \,, \tag{3.106}$$

where $W_k(z)$ is the Lambert W function, and then drawing a uniform random number $\xi \in [0,1)$.

An analytical solution for the product kernel Eq. (3.92) was constructed by Trubnikov (1971) and made applicable for times $\eta > 1$ by Wetherill (1990). The particle number density is

$$\begin{aligned} f(k,\eta) &= \frac{(2k)^{k-1}}{k!k}\left(\frac{\eta}{2}\right)^{k-1} e^{-k\eta} \,, \\ k &\equiv \frac{m}{m^0} \,, \\ \eta &\equiv \Lambda_0 N^0 t \,, \end{aligned} \tag{3.107}$$

where m^0 is the initial particle mass, N^0 is the initial number of particles of mass m^0, $k \in \mathbb{N}$ is a dimensionless mass argument, and η is a dimensionless time coordinate.

As with the constant kernel test, all particles have the same mass initially, making the sampling of equal-weight swarms trivial.

Wetherill (1990) explains that, although the equation solved by Trubnikov (1971) becomes inconsistent for $\eta > 1$ when a runaway particle has separated from the continuous distribution, the solution in Eq. (3.107) still correctly describes the small body mass distribution, that is, the mass distribution of all particles except for the runaway particle. Using conservation of mass, we can infer the mass of the runaway particle by direct summation of Eq. (3.107).

3.B Analytical model for grains-and-boulders test

In this section we develop an analytical model for the grains-and-boulders example used in Sect. 3.5. We derive a system of differential equations for the remaining number of grains N_{g}, the average boulder mass $\langle m \rangle_{\mathrm{B}}$ and the variance of the boulder mass distribution $\sigma^2_{m_{\mathrm{B}}}$, and we give analytical expressions for the cases of constant and linear mutual collision rates, cf. Eqs. (3.94, 3.98).

For any given grain-mass RP, the grain–boulder collision rate is

$$\lambda_{\mathrm{g}} = \sum_{\substack{j=1 \\ m_j \geqslant m^0_{\mathrm{B}}}}^{N} \Lambda_j = N_{\mathrm{B}} \langle \Lambda \rangle_{\mathrm{B}} \,, \tag{3.108}$$

where we define the boulder average $\langle X \rangle_\mathrm{B}$ of a particle-specific quantity X_j as

$$\langle X \rangle_\mathrm{B} \equiv \frac{1}{N_\mathrm{B}} \sum_{\substack{j=1 \\ m_j \geqslant m_\mathrm{B}^0}}^{N} X_j \ , \tag{3.109}$$

(note that in Eq. (3.66) the boulder average was defined differently in the context of the RPMC method) and where the mutual collision rate of any grain with a boulder j was abbreviated as

$$\Lambda_j \coloneqq \Lambda(m_\mathrm{g}, m_j) \ , \tag{3.110}$$

The number of grains N_g decreases stochastically, so we can only reason about the expected value of N_g. At some time t, the expected value will change over a time increment Δt with the grain–boulder collision rate given in Eq. (3.108),

$$\mathrm{E}\big[N_\mathrm{g}'\big] = N_\mathrm{g} \left(1 - \lambda_\mathrm{g} \Delta t\right) \ , \tag{3.111}$$

where the time dependencies of quantities X are henceforth abbreviated as $X \equiv X(t)$, $X' \equiv X(t + \Delta t)$. We thus obtain the differential equation

$$\frac{\mathrm{d}N_\mathrm{g}}{\mathrm{d}t} = -N_\mathrm{B} N_\mathrm{g} \langle \Lambda \rangle_\mathrm{B} \ . \tag{3.112}$$

The growth of a given boulder mass $m_j \geqslant m_\mathrm{B}^0$ is described by a Poisson point process. The chance that boulder j accumulates k grains during a given time increment Δt,

$$m_j' = m_j + k m_\mathrm{g} \ , \tag{3.113}$$

is given by the Poisson distribution with the probability mass function $\mathrm{Pois}_k(\lambda_{\mathrm{B},j} \Delta t)$ defined as

$$\mathrm{Pois}_k(x) = \frac{x^k e^{-x}}{k!} \tag{3.114}$$

for which the expected value of k equals the variance,

$$\mathrm{E}[k] = \mathrm{Var}[k] = x \ . \tag{3.115}$$

For a time increment Δt, the expected average value of m_j' is thus

$$\mathrm{E}\big[\langle m' \rangle_\mathrm{B}\big] = \langle m \rangle_\mathrm{B} + m_\mathrm{g} \langle \Lambda \rangle_\mathrm{B} \, \Delta t \ . \tag{3.116}$$

By taking the difference quotients to the limit $\Delta t \to 0$, we find

$$\frac{\mathrm{d} \langle m \rangle_\mathrm{B}}{\mathrm{d}t} = m_\mathrm{g} N_\mathrm{g} \langle \Lambda \rangle_\mathrm{B} \ , \tag{3.117}$$

where we again note that the number of boulders cannot change, $N_\mathrm{B} = \mathrm{const}$. By relating Eqs. (3.112) and (3.117), the average boulder mass $\langle m \rangle_\mathrm{B}$ can thus be expressed in terms of the number of grains N_g as

$$\langle m \rangle_\mathrm{B} = m_\mathrm{B}^0 + m_\mathrm{g} \frac{N_\mathrm{g}^0 - N_\mathrm{g}}{N_\mathrm{B}} \ . \tag{3.118}$$

To find a differential equation for the variance of the boulder mass $\sigma^2_{m_\mathrm{B}}$ as a function of time, we evaluate the variance at times t and $t + \Delta t$,

$$\sigma^2_{m_\mathrm{B}} = \left\langle m^2 \right\rangle_\mathrm{B} - \langle m \rangle^2_\mathrm{B} \ , \tag{3.119}$$

$$\begin{aligned}\sigma^2_{m'_\mathrm{B}} &= \left\langle m'^2 \right\rangle_\mathrm{B} - \left\langle m' \right\rangle^2_\mathrm{B} \\ &= \frac{1}{N_\mathrm{B}} \sum_{\substack{j=1 \\ m'_j \geqslant m^0_\mathrm{B}}}^{N} \sum_{k=0}^{\infty} \mathrm{Pois}_k(N_\mathrm{g} \Lambda_j \Delta t) \left(m_j + k m_\mathrm{g}\right)^2 \\ &\quad - \left[\langle m \rangle_\mathrm{B} + m_\mathrm{g} N_\mathrm{g} \langle \Lambda \rangle_\mathrm{B} \, \Delta t\right]^2 \ .\end{aligned} \tag{3.120}$$

By subtracting Eq. (3.119) from Eq. (3.120) and taking the difference quotient for $\Delta t \to 0$, we obtain

$$\frac{\mathrm{d}\sigma^2_{m_\mathrm{B}}}{\mathrm{d}t} = \langle \Lambda \rangle_\mathrm{B} \, N_\mathrm{g} m_\mathrm{g}^2 + 2 m_\mathrm{g} \left[\langle m\Lambda \rangle_\mathrm{B} - \langle m \rangle_\mathrm{B} \langle \Lambda \rangle_\mathrm{B}\right] \ , \tag{3.121}$$

where the first term describes the shot noise originating from the discrete nature of the collision events, and the second term represents the covariance of boulder masses and collision rates.

For a constant mutual collision rate, $\Lambda(m, m') = \Lambda_0$, Eq. (3.112) is solved by exponential decay,

$$N_\mathrm{g} = N^0_\mathrm{g} e^{-\lambda_\mathrm{g} t} \ , \tag{3.122}$$

where N^0_g is the initial number of grains and the grain–boulder collision rate λ_g in Eq. (3.108) simplifies to

$$\lambda_\mathrm{g} = N_\mathrm{B} \Lambda_0 \ . \tag{3.123}$$

Because collision rates and boulder masses are uncorrelated, the variance is fully determined by shot noise, and we can directly integrate Eq. (3.121):

$$\sigma^2_{m_\mathrm{B}} = m^2_\mathrm{g} \frac{\langle m \rangle_\mathrm{B} - m^0_\mathrm{B}}{m_\mathrm{g}} \ , \tag{3.124}$$

where we note that the initial variance is zero, $(\sigma^0_{m_\mathrm{B}})^2 = 0$, because all boulders have the same mass initially. In a homogeneous Poisson point process, the variance of the number of collisions equals the expected number of collisions. We see that this also holds true for our non-homogeneous Poisson process if collision rates do not depend on particle masses.

Let us now study a linear mutual collision rate, $\Lambda(m, m') = \Lambda_0 \, (m + m')$ with some constant Λ_0. By evaluating Eq. (3.109), we find

$$\langle \Lambda \rangle_\mathrm{B} = \Lambda_0 \left(m_\mathrm{g} + \langle m \rangle_\mathrm{B}\right) \ . \tag{3.125}$$

Eq. (3.112) then has the solution

$$N_\mathrm{g} = N^0_\mathrm{g} \frac{b}{(b - a) \exp\left[b N^0_\mathrm{g} t\right] + a} \ , \tag{3.126}$$

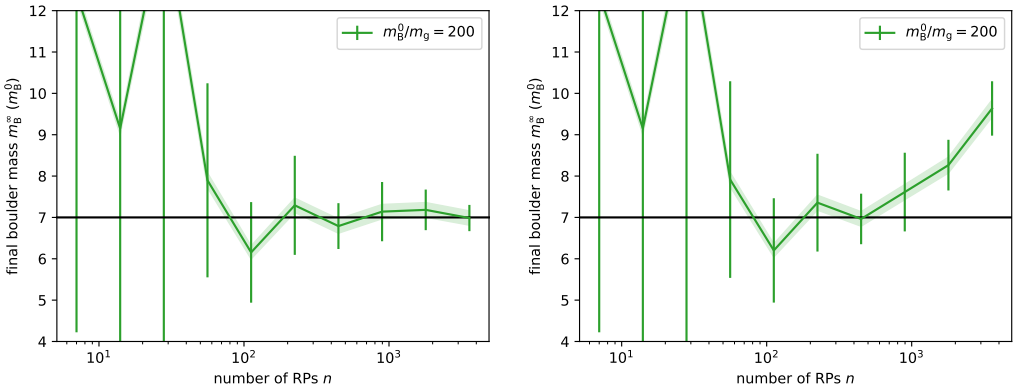


Figure 3.11: As in Fig. 3.9b, but with an initial number of only $N_g^0 = 8 \cdot 10^6$ grains: (a) using the approximate collision rates $\bar{\lambda}_{jk} \approx \tilde{\lambda}_{jk} = \lambda(\mathbf{q}_j, \mathbf{q}_k)$, and (b) using the true collision rates $\bar{\lambda}_{jk} = (1 - \delta_{jk})/2\lambda(\mathbf{q}_j, \mathbf{q}_k)$ instead. As can be seen, using the true collision rates in the many-particles regime would distort the statistical balance of the RPMC method.

where we defined the abbreviations

$$\begin{aligned} a &:= \Lambda_0 m_g \ , \\ b &:= a + \Lambda_0 \left(m_g + m_B^0\right) \frac{N_B}{N_g^0} \ . \end{aligned} \tag{3.127}$$

Using Eq. (3.121) and identifying the variance as per Eq. (3.119), we find the variance of the average boulder mass governed by the differential equation

$$\frac{\mathrm{d}\sigma_{m_B}^2}{\mathrm{d}t} = \langle \Lambda \rangle_B \, N_g m_g^2 + 2 m_g \Lambda_0 \sigma_{m_B}^2 \ , \tag{3.128}$$

where the covariance term describes the broadening of the variance due to the mass dependency of collision rates: smaller boulders will grow more slowly than average, whereas larger boulders will sweep up grains faster.

3.C RPMC approximation and statistical balance

In Item 2 of Sect. 3.4.2, one of the problems with extending the RPMC method to few-particles swarms was identified in the fact that the simplified RP–swarm collision rate of Eq. (3.23) overestimates the RP–swarm collision rate Eq. (3.19), which becomes significant for $N_k \sim 1$. An obvious way of attempting to mitigate this problem would be to use the RP–swarm collision rate $\bar{\lambda}_{jk}$ instead of its approximation $\tilde{\lambda}_{jk}$. But while this would lead to a statistically correct interaction rate, the resulting mass distribution would be skewed. In Fig. 3.11, it can be seen that the approximation $\bar{\lambda}_{jk} \approx \tilde{\lambda}_{jk}$ must be made for the RPMC method to be statistically balanced: without the approximation, the final boulder mass m_B^∞ does not converge to the expected value of $7m_B^0$ as the number of RPs n is increased.

The statistical imbalance can be understood by considering the different origins of the N_g and N_j factors in Eqs. (3.78) and (3.72): some originate from collision rates $\tilde{\lambda}_{\mathrm{B},j}$ and $\tilde{\lambda}_{\mathrm{g}j}$ and would hence be adjusted by summands of $-1/2$ and $-n_\mathrm{g}/2$, respectively, whereas others came from summation over all grain-mass and boulder-mass swarms and thus are not adjusted. The statistical balance equation (Eq. (3.80)) would then end up non-zero with a spurious dependence on the simulation parameters n_g and n_B^0.

3.D List of symbols

A cross-referenced overview of the most commonly used symbols in this chapter is given in Table 3.1.

Symbol	Description	Reference
β_{jk}	boost factor	Sect. 3.4.4
$\Lambda(\mathbf{q}, \mathbf{q}')$	raw grain–boulder collision rate	Sect. 3.5
Λ_j	mutual collision rate between boulder j and a grain	Eq. (3.63); Eq. (3.110)
$\lambda(\mathbf{q}, \mathbf{q}')$	raw collision rate	Sect. 3.3.1
$\lambda_j k$	true collision rate (with self-collision suppressed)	Eq. (3.4)
λ_j	cumulative collision rate of particle j	Eq. (3.5)
λ	total rate of collisions	Eq. (3.6)
$\bar{\lambda}_{jk}$	RP–swarm collision rate	Eq. (3.19)
$\bar{\lambda}_j$	cumulative collision rate of RP j	Eq. (3.20)
$\bar{\lambda}$	total rate of collisions of representative particles	Eq. (3.9)
$\tilde{\lambda}_{jk}$	simplified RP–swarm collision rate	Eq. (3.23)
$\tilde{\lambda}_j$	simplified cumulative collision rate of RP j	Eq. (3.24)
$\tilde{\lambda}$	simplified total rate of collisions of representative particles	Eq. (3.24)
$\tilde{\lambda}^\mathrm{b}_{jk}$	boosted collision rate	Eq. (3.41)
λ_g	grain–boulder collision rate	Eq. (3.108)
$\tilde{\lambda}_{\mathrm{g}j}$	RP–swarm collision rate for a grain-mass RP with boulder j	Eq. (3.64)
$\tilde{\lambda}_\mathrm{g}$	cumulative grain-mass RP collision rate	Eq. (3.65)
$\sigma^2_{m_\mathrm{B}}$	variance of boulder masses	Eq. (3.119)
$\mathcal{I}_\mathrm{g}$, $\mathcal{I}_\mathrm{B}$	index sets of grains and boulders	Eqs. (3.61)
M	total mass in the system	Sect. 3.3.1
M_k	total mass of particles in swarm k	Eq. (3.13)
m_g	grain mass	Sect. 3.5
m_B^0	initial boulder mass	Sect. 3.5
N	number of physical particles	Sect. 3.3.1
N_k	number of physical particles in swarm k	Sect. 3.3.1
N_{jk}	swarm multiplicity factor	Eq. (3.37)
N^b_{jk}	boosted swarm multiplicity factor	Eq. (3.42)
N_th	particle number threshold	Sect. 3.4.3
N'_g	number of grains after duration Δt	Eq. (3.68)
$N'_{\mathrm{B}\to\mathrm{B}}$	number of boulders remaining in boulder-mass swarms after duration Δt	Eq. (3.70)

$N'_{\mathrm{g}\to\mathrm{B}}$	number of boulders emerging from grain-mass swarms after duration Δt	Eq. (3.76)
N_{g}	number of grains	Eq. (3.62)
N_{g}^{0}	initial number of grains	Sect. 3.5
N_{B}	number of boulders	Eq. (3.62)
n	number of representative particles	Sect. 3.3.2
q_m	mass accumulation threshold	Sect. 3.4.4.2
$\mathbf{q}_k$	properties of particle k	Sect. 3.3.1
$\mathbf{q}^{\mathrm{coll}}$	properties of particle sampled from collision outcome	Sect. 3.3.5
W_k	sampling weight of RP k	Eq. (3.34)
$\langle\cdot\rangle_{\mathrm{B}}$	boulder average	Eq. (3.66); Eq. (3.109)

Table 3.1: List of commonly used symbols.

Efficient simulation of stochastic interactions among representative Monte Carlo particles 4

Reference: This chapter will be published as Beutel et al. (in press).

Interaction processes between discrete particles are often modelled with stochastic methods such as the *Representative Particle Monte Carlo* (RPMC) method which simulate mutual interactions (e.g. chemical reactions, collisions, gravitational stirring) only for a representative subset of n particles instead of all N particles in the system. However, in the traditionally employed computational scheme the memory requirements and the simulation runtime scale quadratically with the number of representative particles.

We want to develop a computational scheme that has significantly lower memory requirements and computational costs than the traditional scheme, so that highly resolved simulations with stochastic processes such as the RPMC method become feasible. In this chapter we propose the bucketing scheme, a hybrid sampling scheme that groups similar particles together and combines rejection sampling with a coarsened variant of the traditional discrete inverse transform sampling. For a ν-partite bucket grouping, the storage requirements scale with n and ν^2, and the computational cost per fixed time increment scales with $n \cdot \nu$, both thus being much less sensitive to the number of representative particles n. Extensive performance testing demonstrates the higher efficiency and the favourable scaling characteristics of the bucketing scheme compared to the traditional approach, while being statistically equivalent and not introducing any new requirements or approximations. With this improvement, the RPMC method can be efficiently applied not only with very high resolution but also in scenarios where the number of representative particles increases over time, and the simulation of high-frequency interactions (such as gravitational stirring) as a Monte Carlo process becomes viable.

4.1 Introduction

Studying the emergent dynamics of large discrete systems in numerical simulations is a common challenge in physics. The independent entities that constitute these systems (e.g. molecules in chemical reaction processes, as well as dust grains and pebbles in planetary formation processes) are often so numerous that a direct simulation is neither practically possible nor desirable. For example, an Earth-like planet might be formed by the coagulation of $\sim 10^{40}$ dust grains. The vast number of particles exceeds the capabilities of every conceivable computer; also, we are not interested in the individual fates of every single dust grain but rather in the evolution of their mass statistics.

A traditional method to model discrete systems is to view them as continuous systems whose statistical properties resemble those of the discrete system on certain scales. The continuous surrogate model might then be governed by a set of differential equations that could be solved analytically or numerically. For example, coagulation processes have traditionally been modelled by a coagulation equation. Given a continuous particle mass distribution function $f(m,t)$, the Smoluchowski coagulation equation (Smoluchowski, 1916) is an integrodifferential population balance equation that describes the change of the number of particles $f(m,t)\mathrm{d}m$ in an infinitesimal mass bin $m \in [m, m+\mathrm{d}m]$:

$$\begin{aligned}\frac{\partial}{\partial t} f(m,t) = &-f(m,t)\int_0^\infty \mathrm{d}m'\lambda(m,m')f(m',t) \\ &+\frac{1}{2}\int_0^m \mathrm{d}m'\lambda(m',m-m')f(m',t)f(m-m',t)\ ,\end{aligned} \tag{4.1}$$

where the first term accounts for the loss of particles that grow to larger masses through coagulation, and the second term comprises the particles newly formed by the coagulation of lighter particles. The Smoluchowski equation can be solved numerically with grid-based methods by representing a finite number of particle mass distribution samples with finite mass bins $[m, m+\delta m]$ (e.g. Weidenschilling, 1980; Tanaka et al., 2005; Dullemond and Dominik, 2005; Brauer et al., 2008). The resolution of the grid can be improved by reducing the width of mass bins δm, which increases the number of bins required. However, grid-based methods suffer from the so-called curse of dimensionality in multidimensional parameter spaces. In the example of protoplanetary growth, a model that associates particles only with a mass is a drastic simplification of the planet formation process; a more realistic treatment would have to consider the influence of additional properties such as particle *root mean square* (rms) velocity v or porous particle volume V. In the continuous approach, the particle mass distribution $f(m;t)$ would therefore become a multidimensional particle number distribution $f(m,v,V;t)$, and the multidimensional equivalent of the Smoluchowski equation (Eq. (4.1)) would have to integrate over all dimensions of the state space. To solve the equation numerically with a three-dimensional parameter space, the particle number distribution would then have to be discretised on a three-dimensional grid. For every grid cell, evaluating the right-hand side amounts to a three-dimensional summation; for a grid of size $\mathcal{N}_m \times \mathcal{N}_v \times \mathcal{N}_V$, this requires $(\mathcal{N}_m \times \mathcal{N}_v \times \mathcal{N}_V)^2$ arithmetic operations per timestep, where $\mathcal{N}_m$, $\mathcal{N}_v$, and $\mathcal{N}_V$ represent the number of grid cells in the dimensions of mass, rms velocity, and porosity. Such a simulation becomes extremely expensive if all dimensions are meant to be resolved. However, a lot of the work done is unnecessary: the state space often is not fully

occupied because its dimensions are not entirely uncorrelated; for instance, more massive bodies tend to be less porous due to gravitational compaction, and particles of similar mass tend to have similar rms velocities due to the quasi-thermal diffusive effects of viscous stirring and dynamical friction. The excessive cost of higher-dimensional grids was expounded in, for instance, Okuzumi et al. (2009), whose Eq. (3) gives an extended Smoluchowski equation comprising an additional dimension (porous particle volume V). The authors subsequently argued (in their Sect. 2.2) that adding a dimension with $\mathcal{N}$ bins would make the evaluation of the right-hand side of the Smoluchowski equation more expensive by a factor of $\mathcal{N}^2$, and therefore infeasible. Thus, instead of extending the dimensionality of their grid, they devised a dynamic-average approximation for the new parameter, tracking only the average porous volume $\bar{V}$ for every mass bin rather than resolving the porosity distribution.

Instead of simulating a discrete system as a whole or replacing it with continuous surrogate system, the evolution of the system can also be approximated by selecting a relatively small number of representative entities whose trajectories through state space are then followed, and by estimating the statistical properties of the entire system from the statistics of these representative entities. As an example, protoplanetary growth by coagulation has been simulated with a Monte Carlo method (Ormel and Spaans, 2008; Zsom and Dullemond, 2008; Ormel et al., 2010). The accuracy of the particle mass distribution inferred from a representative set of particles can then be improved by increasing the number of representative entities simulated. Compared to continuous models, representative sampling approaches have the advantage that they sample the state space sparsely. Because the state space value of every representative particle is physically realised, no computational effort is wasted on non-occupied parts of the state space. Also, a mass-weighted sampling automatically increases the sampling resolution in densely populated regions of the state space.

Although the sparsity of the state space sampling makes representative entity methods more viable for the simulation of higher-dimensional state spaces, this computational advantage is somewhat diminished by the quadratic number of possible interactions. For an ensemble of n representative entities, there are n^2 possible types of interactions that must be considered. A Monte Carlo method that simulates individual interactions must therefore know the mutual rates of interaction between all entities represented by any two representative entities in the system. In the computational scheme traditionally employed for such a simulation (Zsom and Dullemond, 2008, §2.1), all n^2 interaction rates between entities j and k are computed and stored. They must be updated as the properties of the entities change over the course of the simulation. For every entity that undergoes a change, $2n$ interaction rates have to be recomputed, making the representative method too demanding to model processes with both a high resolution and frequent interaction (Ormel et al., 2010, §2.6).

In Chapter 3, we amended the Representative Particle Monte Carlo (RPMC) method originally conceived by Zsom and Dullemond (2008). By its construction, the original RPMC method had the limitation that the number of particles N_j represented by each representative particle j needed to be much greater than unity, $N_j \gg 1$, which implied that the method could not be used to simulate runaway growth processes where individual particles accumulate the bulk of the available mass. To overcome this lim-

itation, in Chapter 3 we introduced a distinction between *many-particles swarms* and *few-particles swarms* (a swarm being the ensemble of physical particles represented by a given representative particle). The *particle regime threshold* N_{th}, which is a simulation parameter with a typical value of 'a few' ($N_{\mathrm{th}} = 1$ and $N_{\mathrm{th}} = 10$ were used in Chapter 3), separates many-particles swarms from few-particles swarms. The extended method then establishes different interaction regimes: the interaction between many-particles swarms is said to operate in the *many-particles regime* and proceeds exactly as in the original RPMC method, whereas interactions between a few-particles swarm and a many-particles swarm or between few-particles swarms, which are subsumed under the *few-particles regime*, follow a different procedure under which coagulation leads to mass transfer between swarms. This is essential for representative bodies which outgrow the total mass of their swarm.

To minimise the statistical error and allow individual growth, a many-particles swarm j is split up into N_{th} individual self-representing particles once its particle count N_j falls below the particle regime threshold N_{th}. New representative particles are then added to the simulation, and the representative particle of the swarm is thus replaced with N_j representative particles representing only themselves. As a consequence, a few-particles swarm k always has a particle count $N_k = 1$, although we note that this is not strictly required by the method, and the splitting of swarms may be waived if, other than for the runaway growth scenario, individual representation is not considered necessary.

Numerical methods similar to the representative Monte Carlo methods of Zsom and Dullemond (2008), Ormel and Spaans (2008), and to the extended method developed in Chapter 3 have emerged in other fields. In particular, the 'super-droplet' method of Shima et al. (2009) has gained traction in theoretical research on cloud and ice microphysics (e.g. Brdar and Seifert, 2018; Grabowski et al., 2019). The coalescence of super-droplets (cf. Shima et al., 2009, §4.1.4) is similar to the few-particles regime of the extended RPMC method but with particle counts N_k (ξ_k in the notation of Shima et al. (2009)) not necessarily equal to 1. The super-droplet method was found to generate correct results for several test cases by Unterstrasser et al. (2017).

As a consequence of the initial weight distribution chosen, Shima et al. (2009) had to use very large numbers ($n \sim 2^{17}..2^{21}$) of super-droplets in their simulations. As with the RPMC method, the cost of the traditional computational scheme scales quadratically with the number of super-droplets, which was found to be prohibitively expensive. Shima et al. (2009) therefore proposed a sub-sampling scheme where, in a given time interval, only $\sim n/2$ randomly chosen interaction pairs with an up-scaled interaction rate are considered rather than the full set of $\sim n^2$ interaction pairs. At the expense of additional statistical noise due to the sparse sampling of the matrix of possible interactions, the computational cost of this sub-sampling scheme therefore scales with n rather than n^2. The sub-sampling scheme, often referred to as 'linear sampling' by subsequent publications, was not employed in the study of Unterstrasser et al. (2017), however, it was used by Dziekan and Pawlowska (2017), who meticulously verified that it did not adversely affect their simulation outcomes.

In chapter, we propose a novel computational scheme that significantly reduces the cost for simulating representative entity methods without introducing a statistical coarsening or requiring approximations. Taking advantage of the fact that entities with similar

properties often have similar interaction rates, we group similar particles in 'buckets' and then compute and update bucket–bucket interaction rates, which are obtained as upper bounds for the enclosed entity–entity interaction rates by virtue of interval arithmetic. The interacting entities are then chosen with rejection sampling. The proposed scheme is an optimisation in the sense that it does not affect the statistical outcome of the simulation: no additional inaccuracies are introduced, and the stochastic process simulated is exactly equivalent to a simulation using the traditional computational scheme.

The computational scheme presented here is key to the improvement of the Representative Particle Monte Carlo (RPMC) method proposed in Chapter 3, which splits up swarms as their particle count falls below the particle number threshold N_{th}, and which thus dispensed with a core property of the original RPMC method, namely that the number of representative particles in the simulation shall remain constant. A growing number of representative particles is very costly with the traditional computational scheme where the computational effort scales quadratically with the number of representative particles. With the computational scheme presented in this chapter, the N_{th} representative particles emerging from a split swarm, whose statistical properties are initially identical and tend to remain similar, are not excessively more expensive to simulate than a single representative particle representing the N_{th} particles, making the performance impact of the many–few transition manageable.

In Sect. 4.2, we define a general stochastic process and discuss the traditional computational scheme for simulating it, and we establish a cost model for its storage and computation requirements. In Sect. 4.3, we introduce the bucketing scheme, a new sampling scheme for implementing a stochastic process. We prove that it is equivalent to the traditional sampling scheme, and we derive a detailed cost model demonstrating that the computational demands no longer scale with n^2. The efficient computation of interaction rates is discussed in Sect. 4.4. Interactions in a spatially resolved physical system are often modelled with a maximal radius of interaction; in Sect. 4.5 we introduce subbuckets to efficiently account for the fact that particles far apart cannot interact. Sect. 4.6 verifies our cost model by numerically studying the scalability of the scheme with a set of test problems, and then explores the conditions under which the different contributing terms of the cost model become dominant. Some practical limitations of the proposed scheme are discussed in Sect. 4.7.

4.2 Simulating a stochastic process

Consider an ensemble of n *entities* that interact through a discrete physical process (e.g. by colliding with each other). Each entity k is characterised by a vector of (possibly statistical) properties $\mathbf{q}_k \in \mathbb{Q}$, where $\mathbb{Q}$ is the space of properties. What constitutes an entity is purposefully left unspecified: it might be a physical body, a stochastic representative of a swarm of physical bodies, a surrogate object representing the consolidated influence of some physical effect, etc.

We model interactions as instantaneous events: a given interaction is assumed to occur at a precise time t and may instantaneously alter the properties of the entities involved. For example, if two colliding bodies j, k with masses m_j, m_k 'hit and stick',

then at some precise time t the mass of one body instantaneously changes to $m_j + m_k$, while the other body ceases to exist as a separate entity.

4.2.1 Simulating a Poisson point process

Let us assume that the *interaction rate* λ_{jk} of two entities j, k, in units of time^{-1}, is given by some function of the properties of entities j and k and the Kronecker delta δ_{jk},

$$\lambda_{jk} \coloneqq \lambda_{\mathrm{ent}}(\mathbf{q}_j, \mathbf{q}_k, \delta_{jk}) \ . \tag{4.2}$$

The third argument of this *entity interaction rate function* $\lambda_{\mathrm{ent}}(\mathbf{q}, \mathbf{q}', \delta)$ can be used to distinguish entity self-interaction from interaction of different entities. We emphasise that the interaction rate function has no explicit time dependency. Therefore, if for a given duration the properties $\mathbf{q}_j$, $\mathbf{q}_k$ of entities j and k do not change, the interaction of the entities j, k during that time interval is a homogeneous Poisson point process. We also note that $\lambda_{\mathrm{ent}}(\mathbf{q}, \mathbf{q}', \delta)$ need not be commutative in the first two arguments $\mathbf{q}$, $\mathbf{q}'$, or equivalently, that λ_{jk} need not be the same as λ_{kj}. For instance, interaction rates are asymmetric for the RPMC method (Zsom and Dullemond, 2008), which we refer to throughout this chapter.

The arrival time Δt of the next event in a Poisson point process with a given interaction rate λ follows the exponential distribution characterised by the probability distribution function

$$p(\Delta t) = \lambda \exp[-\lambda \Delta t] \ . \tag{4.3}$$

Using inverse transform sampling[1], we can determine a random arrival time by computing

$$\Delta t = -\lambda^{-1} \log(1 - \xi) \ , \tag{4.4}$$

where ξ is a uniform random number drawn from the interval $[0, 1)$.

4.2.2 Simulating a compound Poisson process

An ensemble of n entities comprises n^2 interaction processes, each of which can be considered a Poisson point process. Because an interaction of two entities j, k may change the properties $\mathbf{q}_j$ and $\mathbf{q}_k$, it may affect the interaction rates of the entities j and k with any other entity $i \in \{1, \ldots, n\}$. A Poisson point process is not homogeneous if interaction rates can suffer intermittent changes. It follows that the sampling method of Eq. (4.4) is applicable only if interactions are simulated in global order as a compound Poisson process, and that all affected interaction rates must be recomputed after an event has occurred.

A scheme for simulating interactions in an ensemble of n entities in global order has first been proposed by Gillespie (1975). We now give a brief summary of this scheme.

[1] An arbitrary normalised probability distribution $f(x)$ can be sampled by means of *inverse transform sampling* (e.g. Ross, 2014, §11.2.1): First, a random number ξ uniformly distributed in $[0, 1)$ is generated. Then, $\tilde{x} = F^{-1}(\xi)$ is evaluated, where $F(x)$ is the antiderivative of $f(x)$ and $F^{-1}(y)$ is its inverse. $\tilde{x}$ then is a random sample distributed according to the probability density function $f(x)$.

First, let λ_j be the *cumulative interaction rate* of entity j with any other entity in the ensemble,

$$\lambda_j := \sum_{k \in \mathcal{I}} \lambda_{jk} \ , \tag{4.5}$$

and let λ be the *global rate of interactions*,

$$\lambda := \sum_{j \in \mathcal{I}} \sum_{k \in \mathcal{I}} \lambda_{jk} = \sum_{j \in \mathcal{I}} \lambda_j \ , \tag{4.6}$$

where we abbreviate the set of all entity indices as

$$\mathcal{I} := \{1, \ldots, n\} \ . \tag{4.7}$$

The arrival time of the next interaction can be sampled as per Eq. (4.4). We then draw random indices j and k from the discrete distribution defined by the joint probability

$$P(j)P(k|j) = \frac{\lambda_{jk}}{\lambda} \ , \tag{4.8}$$

where

$$P(j) = \frac{\lambda_j}{\lambda} \tag{4.9}$$

is the chance that entity j is involved in the interaction event, and

$$P(k|j) = \frac{\lambda_{jk}}{\lambda_j} \tag{4.10}$$

is the chance that, given such an interaction, it is entity k which interacts with entity j. We then advance the system by the time increment Δt and carry out the interaction between entities j and k, allowing their properties to be altered. The interaction may also effectuate the creation of new entities (e.g. when colliding bodies fragment to smaller pieces) or the annihilation of entities (e.g. when colliding particles 'stick' and thus merge). Therefore, the global interaction rate λ has to be recomputed after an interaction.

4.2.3 Incremental updates

A routine that directly computes the global interaction rate λ for an ensemble of n entities entails n^2 evaluations of the entity interaction rate function $\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta)$, which may be prohibitively expensive for large ensembles.

An interaction of entity j with entity k may inflict changes on the two entities involved, wherefore the global interaction rate may change and must be recomputed. In Zsom and Dullemond (2008, §2.1), a way of alleviating the cost of recomputing λ was described. The authors observed that a change to the properties of entity j can only affect the interaction rates $\lambda_{ji}, \lambda_{ij} \forall i \in \{1, \ldots, n\}$. Therefore, one can store the entirety of pairwise interaction rates λ_{jk} in a two-dimensional array and the cumulative interaction rates λ_j, obtained as the column sum of the λ_{jk} array, in a one-dimensional array. Then, when the properties of entity j change during an interaction, only a row and a column in the λ_{jk} array have to be recomputed, and the λ_j array can be updated cumulatively. At the expense of a memory buffer holding $(n^2 + n)$ interaction rates, the interaction rates can thus be updated with only $(2n - 1)$ evaluations of the interaction rate function.

4.2.4 Discrete inverse transform sampling

Knowing the exact values for the global interaction rate λ, the cumulative interaction rates λ_j, and the individual interaction rates λ_{jk}, we can determine the time interval until the next event Δt by sampling the exponential distribution with Eq. (4.4). To find the indices j and k of the entities interacting, we can then employ *discrete inverse transform sampling*, a discrete variant of the inverse transform sampling scheme also described by Gillespie (1975): we first draw a uniformly distributed random number $\xi \in [0, 1)$, and then we compute the partial sum

$$\sum_{j'=1}^{j} P(j') = \frac{1}{\lambda} \sum_{j'=1}^{j} \lambda_{j'} \tag{4.11}$$

consecutively for indices $j = 1, \ldots, n$, stopping and picking the first index j for which

$$\sum_{j'=1}^{j} P(j') > \xi \,, \tag{4.12}$$

where $P(j)$ was defined in Eq. (4.9), and likewise for index k and the probability $P(k|j)$ defined in Eq. (4.10). With precomputed values for λ_j and λ_{jk} available, this operation is a simple cumulative summation of an array, taking n summation steps on average.

4.2.5 Example: Physical bodies

How the entity interaction rate function $\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta)$ should be defined depends on what is understood by an entity. As an example, we might identify entities with individual physical bodies that interact through collision. The *raw collision rate* $\lambda(\mathbf{q}, \mathbf{q}')$ is the rate (in units of time^{-1}) at which two bodies with properties $\mathbf{q}$, $\mathbf{q}'$ are expected to collide. It is necessarily commutative, $\lambda(\mathbf{q}, \mathbf{q}') = \lambda(\mathbf{q}', \mathbf{q})$. If the particles are distributed homogeneously and isotropically in a volume V, the stochastic collision rate $\lambda(\mathbf{q}, \mathbf{q}')$ can be written as

$$\lambda(\mathbf{q}', \mathbf{q}') = V^{-1} \sigma(\mathbf{q}, \mathbf{q}') \left| \Delta \mathbf{v}(\mathbf{q}, \mathbf{q}') \right| \,, \tag{4.13}$$

where $\sigma(\mathbf{q}, \mathbf{q}')$ is the collision cross-section and $|\Delta \mathbf{v}(\mathbf{q}, \mathbf{q}')|$ the average relative velocity between particles with properties $\mathbf{q}$, $\mathbf{q}'$. To account for the fact that a particle cannot collide with itself, and to avoid double counting of mutual interactions, the entity interaction rate function $\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta)$ would then be defined as

$$\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta) \coloneqq \frac{1}{2}(1 - \delta) \lambda(\mathbf{q}, \mathbf{q}') \,. \tag{4.14}$$

4.2.6 Example: Representative particles

Alternatively, an entity j might be identified with both a representative particle and an associated swarm of N_j particles, with the properties of representative particle and swarm both captured in $\mathbf{q}_j$. In that case, an entity can interact with itself (a representative body can collide with another particle from the swarm which its entity is associated with), and the *raw interaction rate* $\lambda(\mathbf{q}, \mathbf{q}')$ is not commutative: representative particle j

may be more likely to collide with a particle from swarm k than representative particle k is to collide with a particle from swarm j.

The extended RPMC method introduced in Chapter 3 distinguishes between many-particles swarms and few-particles swarms, where a swarm is a many-particles swarm if its number of particles exceeds the particle number threshold N_{th}. The *effective swarm particle count* is thus defined according to the interaction regime,

$$N^{\text{eff}}(\mathbf{q}, \mathbf{q}', \delta) = \begin{cases} N(\mathbf{q}') & \text{(many-particles regime)} \\ N(\mathbf{q}') - \frac{1+\delta}{2} & \text{(few-particles regime),} \end{cases} \tag{4.15}$$

where $N(\mathbf{q}_k) \equiv N_k$ is the number of swarm particles represented by entity k, and where an interaction between entities with properties $\mathbf{q}$ and $\mathbf{q}'$ is said to operate in the many-particles regime if both swarms are many-particles swarms, $N(\mathbf{q}) > N_{\text{th}}$ and $N(\mathbf{q}') > N_{\text{th}}$, and in the few-particles regime otherwise. For the RPMC method, the entity interaction rate function is then defined as

$$\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta) := N^{\text{eff}}\big(\mathbf{q}, \mathbf{q}', \delta\big)\, \lambda\big(\mathbf{q}, \mathbf{q}'\big) \ . \tag{4.16}$$

In the few-particles regime, double counting is compensated and self-interaction is suppressed for interactions between representative particles but not for interactions between a representative and a non-representative particle. In particular, because $N_{\text{th}} \geqslant 1$, Eq. (4.16) degenerates to Eq. (4.14) for $N(\mathbf{q}') = 1$.

4.2.7 Incorporating external effects

In many physical codes, multiple types of interactions have to be considered, and hence a given process of stochastic interactions needs to be interweaved with other processes of stochastic or deterministic nature such as direct N-body interaction or hydrodynamic processes. Entity properties may then undergo changes originating in physical processes which are modelled by other parts of the simulation and hence not represented as discrete events in the stochastic process at hand. Computing incremental updates after every stochastic event then no longer suffices as all interaction rates can be subject to external changes at all times. As an example, consider an ensemble of particles in quasi-kinetic motion which populate a volume V also occupied by a gas. The gas exerts a drag force on the particles, thus slowing them down and reducing the average relative velocity $|\Delta\mathbf{v}(\mathbf{q}, \mathbf{q}')|$, and thereby decreasing their mutual interaction rates over time as per Eq. (4.13).

As a simple means of incorporating external effects in a stochastic simulation, one could accumulate and coalesce external changes to entity properties and defer the recomputation of interaction rates until they exceed some absolute or relative change threshold. When the properties of an entity are changed by more than a given relative threshold, the interaction rates for the entity are updated.

4.2.8 Cost model

The memory requirements of this sampling scheme amount to approximately C_{mem} floating-point values:

$$C_{\text{mem}} = n^2 + (\alpha + 1) \cdot n + 1 \ , \tag{4.17}$$

where $\alpha = \dim \mathbb{Q}$ is the number of floating-point values needed to store the properties of an entity. For the sampling process, we need to store n^2 pair-wise interaction rates and $n + 1$ cumulative interaction rates.

To model the computational complexity, we first define some elementary computational costs:

- C_{op}, the cost of performing an elementary arithmetic operation (e.g. adding two numbers);
- C_{rand}, the cost of generating a random number distributed uniformly in $[0, 1)$;
- C_λ, the cost of evaluating the entity interaction rate function $\lambda_{\mathrm{ent}}(\mathbf{q}, \mathbf{q}', \delta)$;
- C_{action}, the cost of simulating an interaction between two given entities.

We are aware that, on a modern superscalar system, one cannot reason about the cost of elementary arithmetic operations without also considering how computations are executed and how memory is accessed. With some effort, the computational performance of a system with arithmetic vector units can far exceed the performance predicted by a simplistic linear cost model, and the cost of memory accesses may be anything from completely marginalised through caches, speculative execution, and other means of hiding latency to throttling the computational power through limited bandwidth or inefficient memory access patterns. Therefore, the cost units above are not meant to be identified with specific quantities (such as '10 CPU cycles'). Instead, our cost model aims to predict the scaling behaviour of the scheme. We use the cost units to identify which operations scale with n^2 or with n, from which we can infer which individual costs would be most worthwhile to reduce.

To initialise the simulation, all pair-wise interaction rates between entities must be computed and summed up column-wise. The cost of initialisation therefore is

$$C_{\mathrm{init}} = n^2 \left(C_\lambda + C_{\mathrm{op}}\right) + n\, C_{\mathrm{op}} \,. \tag{4.18}$$

To sample and simulate an interaction event, we need to draw three uniform random numbers, traverse up to n floating-point numbers two times during the sampling procedure described in Sect. 4.2.4, and then simulate the interaction itself. Assuming that, on average, half of the n interaction rates need to be traversed during sampling, this amounts to a per-event cost of

$$C_{\mathrm{event}} \approx n\, C_{\mathrm{op}} + 3\, C_{\mathrm{rand}} + C_{\mathrm{action}} \,. \tag{4.19}$$

When the properties of an entity have changed – that is, when the entity was subject to an interaction, or when the cumulative changes of external effects have exceeded a given threshold –, all interaction rates involving the entity need to be recomputed. Executed as an iterative update, this has a cost of

$$\begin{aligned} C_{\mathrm{update}} &= (2n - 1)\left(C_\lambda + C_{\mathrm{op}}\right) \\ &\approx 2n \left(C_\lambda + C_{\mathrm{op}}\right) \,. \end{aligned} \tag{4.20}$$

To estimate how the number of interactions scales with the number of entities n, we need to make further assumptions on the structure of the entity interaction rate function $\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta)$. Let us consider the case of representative entities in Eq. (4.16) and assume that the number of particles represented by an entity is large, $N_k \gg 1 \,\forall k$. Then, we approximate the entity interaction rate function as

$$\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta) \approx N(\mathbf{q}')\lambda(\mathbf{q}, \mathbf{q}') \; . \tag{4.21}$$

By definition, the global interaction rate (Eq. (4.6)) then is

$$\begin{aligned} \lambda &\approx \sum_{j,k \in \mathcal{I}}^{n} N_k \lambda(\mathbf{q}_j, \mathbf{q}_k) \\ &= n^2 \left\langle N_k \lambda(\mathbf{q}_j, \mathbf{q}_k) \right\rangle_{jk} \; , \end{aligned} \tag{4.22}$$

where $\langle x_{jk} \rangle_{jk}$ is the quantity x_{jk} averaged over all combinations of entities j, k. In this picture, a certain number of N physical particles is represented by n representative particles each associated with an entity. From $\sum_{k=1}^{n} N_k = N$ we infer

$$\langle N_k \rangle_k = \frac{N}{n} \; , \tag{4.23}$$

and because $\lambda(\mathbf{q}_j, \mathbf{q}_k)$ should not depend on N_j or N_k, we argue that the mean value $\langle N_k \lambda(\mathbf{q}_j, \mathbf{q}_k) \rangle_{jk}$ must be proportional to n^{-1}. Assuming an invariant number of physical particles N, we therefore obtain an approximate scaling behaviour of $\lambda \propto n$, or equivalently

$$\lambda \approx n\tilde{\lambda} \; . \tag{4.24}$$

with some average raw collision rate $\tilde{\lambda}$.

To consider the cost of changes imposed by external effects as sketched in Sect. 4.2.7, we assume that the average rate of updates per entity triggered by external effects is quantified by $\tilde{\lambda}_{\text{ext}}$, independent of the number of entities. The global average rate of external updates is therefore

$$\lambda_{\text{ext}} = n\tilde{\lambda}_{\text{ext}} \; . \tag{4.25}$$

The simulation cost rate – the cost of simulating an ensemble of n entities for a given time interval Δt divided by Δt – then is

$$\begin{aligned} \frac{C_{\text{sim}}(\Delta t)}{\Delta t} &= \lambda \left(C_{\text{event}} + C_{\text{update}} \right) + \lambda_{\text{ext}} \, C_{\text{update}} \\ &\approx n^2 \left[\tilde{\lambda} \left(3C_{\text{op}} + 2C_{\lambda} \right) + \tilde{\lambda}_{\text{ext}} \left(2C_{\text{op}} + 2C_{\lambda} \right) \right] \\ &\quad + n \, \tilde{\lambda} \left(3C_{\text{rand}} + C_{\text{action}} \right) \; . \end{aligned} \tag{4.26}$$

Despite being a vast improvement over direct summation, the incremental updating scheme is still an impediment to larger-scale simulations as its costs for a fixed time increment scale with n^2.

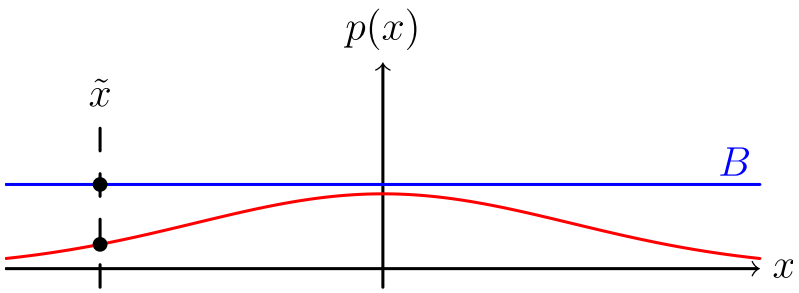


Figure 4.1: Schematic illustration of rejection sampling.
Given a continuous probability distribution function $p(x)$ (solid red line) defined on some domain $D \equiv [x^-, x^+]$, a single p-distributed sample $\tilde{x} \in D$ can be obtained as follows: First estimate an upper bound $B \geqslant p(x)\ \forall x \in D$ (solid blue line). Then sample a random number $\tilde{x}$ uniformly distributed in I (dashed vertical line). Evaluate $p(\tilde{x})$. Then accept the sample with a probability of $p(\tilde{x})/B$. Repeat until a sample is accepted.

4.3 An improved sampling scheme

In this part we devise a new scheme for computing interaction rates that avoids the n^2 proportionality of the costs for storage and computation entailed by the traditional sampling scheme. Using rejection sampling, a bucketing scheme, and interval arithmetic, we develop a generally applicable method for efficiently simulating large ensembles of representative particles.

In the traditional sampling scheme described in Sect. 4.2, the dominant cost is the computation and storage of the entity interaction rates λ_{jk}. Not only does this cost scale with n^2, but the interaction rate may also be expensive to compute by itself, for instance in a realistic physical collision model. We therefore strive to evaluate the interaction rate function as seldom as possible.

Exact values for λ, λ_j, and λ_{jk} are required to draw random samples with the discrete inverse transform sampling scheme described in Sect. 4.2.4. Because we would like to reduce the number of evaluations of the interaction rate function, we need to consider alternative sampling schemes.

4.3.1 Rejection sampling

The method of rejection sampling, which is illustrated in Fig. 4.1, can be used to generate event times for a non-homogeneous Poisson point process if an upper bound for the interaction rate λ is known, cf. Ross (2014, §11.5.1). Let us assume we can obtain an upper bound

$$\lambda^+ \geqslant \lambda \quad \forall \Delta t \leqslant T\ , \tag{4.27}$$

where t is the current time and T is the timescale on which external changes to the system need to be considered. We can then sample a potential event at time $t + \Delta t$ by sampling a time interval Δt from the exponential distribution given by Eq. (4.4) but

using λ^+ instead of λ. If $\Delta t > T$, no event occurs during the timescale T; we then update the system by applying the external operators, compute a new timescale T' from the external process and a new upper bound λ'^+, and update

$$\Delta t \leftarrow \frac{\lambda^+}{\lambda'^+}(\Delta t - T) \tag{4.28}$$
$$\lambda^+ \leftarrow \lambda'^+ \tag{4.29}$$
$$t \leftarrow t + T \tag{4.30}$$
$$T \leftarrow T' \,. \tag{4.31}$$

Once a small enough time interval has been sampled, $\Delta t \leqslant T$, we compute the exact interaction rate λ at time $t + \Delta t$ and choose to accept the sampled event time with a probability of

$$P_{\text{accept}} = \frac{\lambda}{\lambda^+} \,. \tag{4.32}$$

Regardless of acceptance, we update the times as

$$\begin{aligned} t &\leftarrow t + \Delta t \,, \\ T &\leftarrow T - \Delta t \,. \end{aligned} \tag{4.33}$$

If the event time was accepted, we simulate an interaction by determining the indices j and k of the interacting entities, and then repeat the process from the beginning.

This sampling scheme can also be applied to the compound Poisson process of n entities. To this end, we take λ to be the global interaction rate in Eq. (4.6). After an event has been accepted, the indices of the entities interacting are chosen with discrete inverse transform sampling of the joint probability given in Eq. (4.8). The probability that an interaction between entities j and k occurs at time $t + \Delta t$ is thus

$$P_{jk} \coloneqq P_{\text{accept}} P(j) P(k|j) = \frac{\lambda_{jk}}{\lambda^+} \,. \tag{4.34}$$

The proposed scheme still requires us to compute all interaction rates λ_{jk} and the global interaction rate λ at time $t + \Delta t$ in order to determine whether to accept the sampled event time Δt and to sample the indices j and k, and therefore retains the performance characteristics of the sampling scheme for a homogeneous Poisson process. But as we demonstrate in this chapter, the computational cost can be significantly reduced by flexible and more granular use of upper bounds.

4.3.2 Buckets

Rejection sampling is an efficient sampling scheme only if the upper bound λ^+ is sufficiently close to the true interaction rate λ: the more the upper bound overestimates the interaction rate, the more sampled events have to be rejected, nevertheless each incurring an evaluation of the entity interaction rate function $\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta)$.

In order to increase the efficiency of rejection sampling, we introduce the *bucketing scheme*, the concept of which is explained schematically in Fig. 4.2. The basic idea is

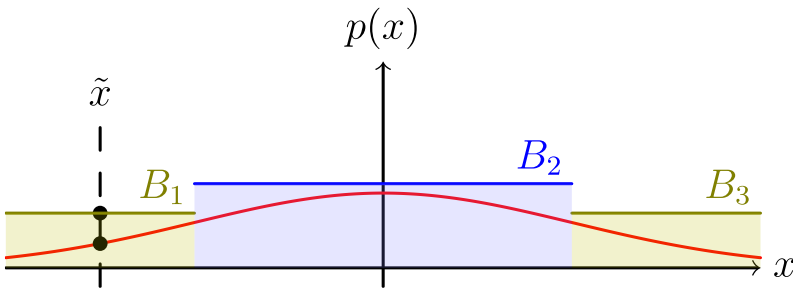


Figure 4.2: Schematic illustration of bucketing.
Rejection sampling as per Fig. 4.1 is inefficient if the upper bound B significantly overestimates the probability distribution function $p(x)$, necessitating a large number of evaluations of $p(x)$ per accepted sample. To increase efficiency, the domain D can be divided into disjoint subdomains $D_1, \ldots, D_\nu$ (shaded with different colours), henceforth referred to as 'buckets'. Then, separate upper bounds $B_1, \ldots, B_\nu$ (solid blue line) can be estimated for the individual buckets. To sample a value $\tilde{x}$ from $p(x)$, first choose a bucket $J \in \{1, \ldots, \nu\}$ with relative probability $|D_J|/|D|$ using discrete inverse transform sampling. Then, sample a value $\tilde{x}$ uniformly distributed in bucket D_J. Evaluate $p(\tilde{x})$ and accept the sample with a probability of $p(\tilde{x})/B_J$, and repeat the entire process until a sample has been accepted. If the per-bucket bounds B_J are lower than the global upper bound, fewer samples are rejected, and hence fewer evaluations of $p(x)$ are required.

to group similar entities in 'buckets', then to compute upper bounds of the interaction rates between buckets of entities. To group entities in buckets, we first need a *bucketing criterion*, that is, a function $B : \mathbb{Q} \to \mathbb{J}$ that maps a vector of entity properties $\mathbf{q}$ to an element $B(\mathbf{q})$ in some discrete ordered set $\mathbb{J}$. In simpler words, for any given entity j the function $B(\mathbf{q}_j)$ tells which bucket this entity is in. Each bucket is uniquely identified by an element $J \in \mathbb{J}$, henceforth referred to as the *label* of the bucket. A convenient choice for a label would be a d-dimensional vector of integers, $\mathbb{J} \equiv \mathbb{Z}^d$, which can be ordered lexicographically.

With B_j we denote the label of the bucket which entity j is currently in, which we initialise as

$$B_j \leftarrow B(\mathbf{q}_j) \ . \tag{4.35}$$

We refer to the set of entity indices associated with a bucket with label $J \in \mathbb{J}$ as

$$\mathcal{I}_J := \{j \in \mathcal{I} : B_j = J\} \ . \tag{4.36}$$

In other words, $\mathcal{I}_J$ is the index set of entities that are in bucket J. The number of entities in a bucket is denoted

$$n_J := \#\mathcal{I}_J \ , \tag{4.37}$$

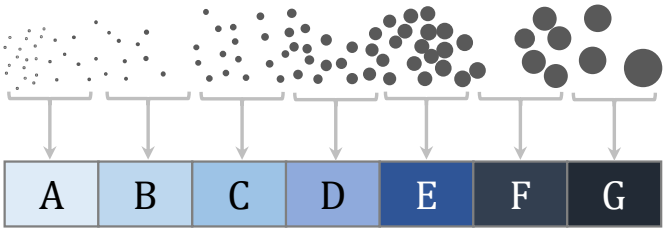


Figure 4.3: Schematic illustration of the mass bucketing criterion in Eq. 4.41. The orbs represent particles of different mass, visually represented as surface area, which are grouped into seven buckets.

consistent with $n = \#\mathcal{I}$, where $\#\mathcal{S}$ refers to the cardinality of some set $\mathcal{S}$. Buckets are necessarily disjoint,

$$\dot{\bigcup}_J \mathcal{I}_J = \mathcal{I} \,, \qquad \sum_J n_J = n \,, \tag{4.38}$$

meaning that each entity belongs to one, and only one, bucket. The set of occupied bucket labels, that is, the labels of the buckets which contain at least one entity, is

$$\mathcal{B} := \left\{ B_j : j \in \mathcal{I}, n_{B_j} > 0 \right\} \,, \tag{4.39}$$

the number of which is referred to as

$$\nu := \#\mathcal{B} \,. \tag{4.40}$$

Because the bucketing scheme samples the space of bucket labels $\mathbb{J}$ sparsely, the cost of storage and computational effort is a function of ν.

4.3.3 Example: Representative particles with mass

Pursuing the example given in Sect. 4.2.6 further, we identify entities with representative particles. We also assume that each representative particle j has a mass $m_j = m(\mathbf{q}_j)$, and that the collision rate $\lambda(\mathbf{q}_j, \mathbf{q}_k)$ is correlated with the masses m_j and m_k. The bucketing criterion is therefore chosen to be mass-dependent. A typical physical scenario studied with an RPMC coagulation simulation covers a dynamic range of several orders of magnitude in particle mass, hence a logarithmic dependency on mass is appropriate. A simple yet effective one-dimensional bucketing criterion thus is

$$B(\mathbf{q}) = \left\lfloor \theta_m \log_{10} \frac{m(\mathbf{q})}{m_0} \right\rfloor \tag{4.41}$$

for some reference mass m_0, where $m(\mathbf{q})$ is the mass of a representative particle with properties $\mathbf{q}$, and where the expression $\lfloor \cdot \rfloor$ denotes the floor function. The *bucket density* θ_m is a simulation parameter which specifies how many buckets per mass decade should be used, thus indirectly controlling the number of occupied buckets ν. The bucketing criterion in Eq. (4.41) is visualised schematically in Fig. 4.3.

As explained in Sect. 4.2.6, for the extended RPMC method, particle–swarm interaction rates differ between the different interaction regimes (the many-particles regime and the few-particles regime). Therefore, it is reasonable to also use the swarm multiplicity as part of the bucketing criterion. Another change brought about by the extended RPMC method is that swarm masses are allowed to vary. In the original definition of the method, the same fraction of the total mass had been assigned to each swarm. The interaction rate λ_{jk} has a strong dependency on the number of particles represented, as seen in the (near-)proportionality to $N(\mathbf{q}')$ in Eqs. (4.16), and therefore also depends on the swarm mass to which the number of particles relates as $N(\mathbf{q}) = M(\mathbf{q})/m(\mathbf{q})$, where $M(\mathbf{q})$ refers to the total mass represented by the swarm. For this reason, we also want to have the swarm mass be part of the bucketing criterion.

For the extended RPMC method we therefore propose the following three-dimensional bucketing criterion:

$$B(\mathbf{q}) = \left(C(\mathbf{q}), \left\lfloor \theta_M \log_{10} \frac{M(\mathbf{q})}{M_0} \right\rfloor, \left\lfloor \theta_m \log_{10} \frac{m(\mathbf{q})}{m_0} \right\rfloor \right) \tag{4.42}$$

with $M_0 = \mathcal{M}/n$ the average swarm mass, $\mathcal{M}$ the total mass of particles in the system, and the bucket density θ_M a simulation parameter specifying the number of buckets to use per decade of swarm masses. The classifier $C(\mathbf{q})$ is defined as

$$C(\mathbf{q}) = \begin{cases} 0\,; & N(\mathbf{q}) \leqslant N_{\text{th}} \text{ (few-particles swarm)} \\ 1\,; & N(\mathbf{q}) > N_{\text{th}} \text{ (many-particles swarm)} \end{cases} \tag{4.43}$$

with N_{th} the particle regime threshold.

The traditional RPMC method always employs an equal-mass sampling, $M_i = M_0$ $\forall i \in \mathcal{I}$, which stays unaltered over the course of the simulation, and it entirely operates in the many-particles regime, $N_i \gg 1\ \forall i \in \mathcal{I}$, and hence $N_i > N_{\text{th}}\ \forall i \in \mathcal{I}$. Therefore, the first and second components of Eq. (4.42) would always evaluate to 1 and $\lfloor \theta_M \rfloor$, rendering this bucketing criterion equivalent to Eq. (4.41) in this case.

4.3.4 Sampling an event

Let us assume that, given a bucket J and its constituent entities $j \in \mathcal{I}_J$, we can compute its *bucket properties*, which we denote as Q_J; and that, given two buckets J, K and their associated bucket properties Q_J, Q_K, we can compute an upper bound λ^+_{JK} for the interaction rates λ_{jk} between any entities j, k associated with these buckets,

$$\lambda^+_{JK} \geqslant \lambda_{jk} \quad \forall j \in \mathcal{I}_J, k \in \mathcal{I}_K\ , \tag{4.44}$$

or equivalently

$$\lambda^+_{JK} \geqslant \max_{j \in \mathcal{I}_J, k \in \mathcal{I}_K} \lambda_{jk}\ . \tag{4.45}$$

Bucket properties and the computation of bucket–bucket interaction rate bounds remain unspecified here but are discussed in detail in Sect. 4.4.

We then define the *upper bounds of the cumulative interaction rates* for all buckets J,

$$\lambda_J^+ = \sum_K n_K \lambda_{JK}^+ \ , \tag{4.46}$$

and the *upper bound of the global interaction rate*

$$\lambda^+ = \sum_J n_J \lambda_J^+ \ . \tag{4.47}$$

Inserting Eqs. (4.45) and (4.46) into Eq. (4.47) proves that λ^+ indeed constitutes an upper bound to the global rate of interactions λ (Eq. (4.6)):

$$\begin{aligned}
\lambda^+ &\geqslant \sum_{J\in\mathcal{B}} \sum_{K\in\mathcal{B}} n_J n_K \max_{j\in\mathcal{I}_J, k\in\mathcal{I}_K} \lambda_{jk} \\
&\geqslant \sum_{J\in\mathcal{B}} \sum_{K\in\mathcal{B}} \sum_{j\in\mathcal{I}_J} \sum_{k\in\mathcal{I}_K} \lambda_{jk} \\
&\overset{!}{=} \sum_{j\in\mathcal{I}} \sum_{k\in\mathcal{I}} \lambda_{jk} = \lambda \ .
\end{aligned} \tag{4.48}$$

Using the upper bounds λ^+ and λ_{JK}^+, we can now implement the rejection sampling scheme of Sect. 4.3.1 more efficiently by interchanging the acceptance decision and the selection of entity indices j, k:

1. First, we randomly choose bucket indices J, K with a probability

$$P(J)P(K|J) = \frac{n_J \lambda_J^+}{\lambda^+} \frac{n_K \lambda_{JK}^+}{\lambda_J^+} = \frac{n_J n_K \lambda_{JK}^+}{\lambda^+} \tag{4.49}$$

   using discrete inverse transform sampling as per Sect. 4.2.4.

2. We then randomly choose entity indices $j \in \mathcal{I}_J$, $k \in \mathcal{I}_K$ with the probability

$$P(j|J)P(k|K) = \frac{1}{n_J} \frac{1}{n_K} \tag{4.50}$$

   which is independent of j and k, amounting to uniform or unweighted sampling.

3. Then we compute the interaction rate λ_{jk} of the entities chosen. The interaction event is then accepted with a probability of

$$P_{\text{accept},jk} = \frac{\lambda_{jk}}{\lambda_{JK}^+} \ . \tag{4.51}$$

Combining all three probabilities, we obtain the selection probability

$$P(J)P(K|J) \cdot P(j|J)P(k|K) \cdot P_{\text{accept},jk} = \frac{\lambda_{jk}}{\lambda^+} \tag{4.52}$$

which we find identical to the combined sampling probability P_{jk} in Eq. (4.34), thereby proving that this selection process is equivalent.

4.3.5 The bucketing algorithm

We now describe how a general compound Poisson process as defined in Sect. 4.2.2 can be simulated with the bucketing scheme. The proposed algorithm consists of the following steps:

1. Initialise simulation.
   - 1.i. Distribute all entities j to buckets. To this end, for every entity j determine the entity bucket label $B_j \leftarrow B(\mathbf{q}_j)$. Assemble a list of occupied buckets and, for every occupied bucket, a list of the entities in it.
   - 1.ii. For all occupied buckets $J \in \mathcal{B}$, compute the bucket properties Q_J, which are a function of the properties $\mathbf{q}_j$ of the entities in the bucket, $j \in \mathcal{I}_J$.
   - 1.iii. For all pairs of buckets $J, K \in \mathcal{B}$, compute bucket–bucket interaction rate upper bounds λ^+_{JK}.
   - 1.iv. Compute the upper bounds for the cumulative bucket interaction rate bound λ^+_J for all occupied buckets $J \in \mathcal{B}$ by summation of $n_K \lambda^+_{JK}$, and compute the upper bound of the global interaction rate λ^+ by summation of all $n_J \lambda^+_J$.
2. Sample an event.
   - 2.i. Sample an event interarrival time Δt with Eq. (4.4) using λ^+ as the interaction rate.
   - 2.ii. If the time exceeds the timescale of external updates, $\Delta t > T$, apply the external updates, execute Step 3 for every entity whose properties underwent significant changes during the external update, then adjust Δt, t, T as per Eqs. (4.28–4.31), then start over with Step 2.i.
   - 2.iii. It can be assumed henceforth that $\Delta t \leqslant T$: the sampled tentative interaction event may occur within the external updating timescale. Thus, using discrete inverse transform sampling with the stored interaction rate bounds λ^+, λ^+_J, and λ^+_{JK}, randomly choose the buckets $J, K \in \mathcal{B}$ which contain the entities possibly determined to undergo an interaction event.
   - 2.iv. Using uniform sampling, randomly choose a pair of entities $j \in \mathcal{I}_J$, $k \in \mathcal{I}_K$.
   - 2.v. Compute the interaction rate λ_{jk}. Accept the event with probability $P_{\text{accept},jk}$ (Eq. (4.51)).
     Advance the system time by the event interarrival time, $t \leftarrow t + \Delta t$ (Eq. (4.33)).
   - 2.vi. If the event was accepted, simulate the interaction between entities j and k, thereby possibly altering the entity properties $\mathbf{q}_j \rightarrow \mathbf{q}'_j$, $\mathbf{q}_k \rightarrow \mathbf{q}'_k$, and then execute Step 3 for all entities whose properties have changed (none, j, k, or both j and k).
   - 2.vii. Continue with Step 2.i.
3. Subroutine: Update a given entity j.
   - 3.i. Determine the old and new bucket labels $J := B_j$, $J' := B'_j$. Move the entity to the new bucket if $J' \neq J$.

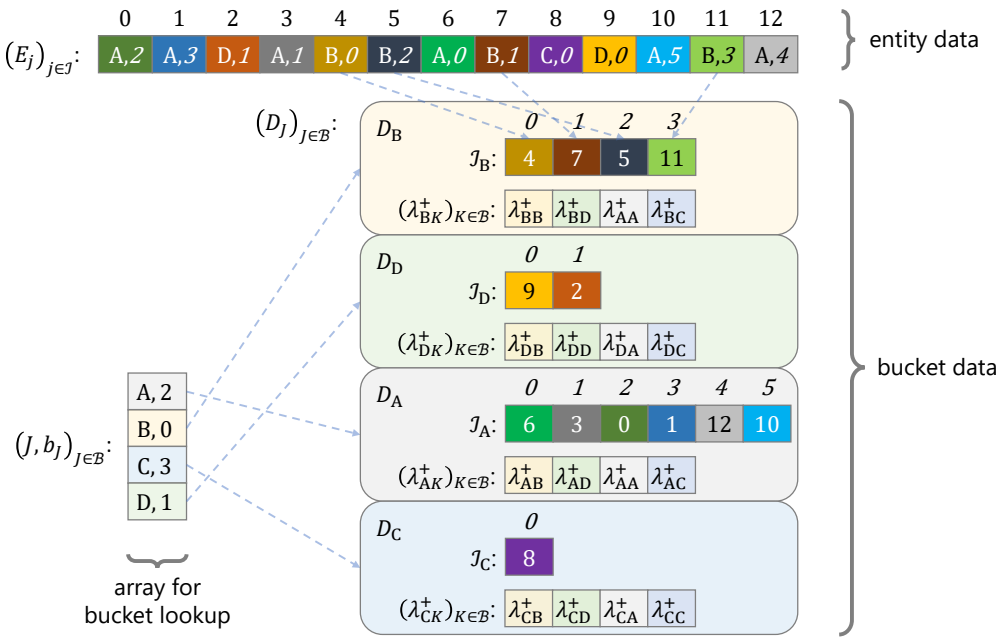


Figure 4.4: Example illustrating the data structures used to implement the bucketing scheme as described in Sect. 4.3.6.
This example comprises an ensemble of $n = 13$ entities partitioned in $\nu = 4$ buckets, with the bucket labels being designated as A, B, C, D, ordered lexicographically, and using 0-based numeric indices for all other indexing purposes. Entity indices are printed in normal type. The indices of the array entries at which the entity indices are stored in their associated buckets are printed in italic type. Element referral is indicated by dashed lines. Some elements of the D_J tuples were omitted in the visual representation.

3.ii. For bucket J', update (if necessary) or recompute (if due) the bucket properties $Q_{J'}$.
Sect. 4.4.4 precisely defines what 'updating' means and how we decide whether updating suffices or the bucket properties need to be recomputed.

3.iii. If the bucket properties were recomputed, or if the updating step caused an alteration of the bucket properties $Q_{J'}$ of the new bucket J', recompute the bucket–bucket interaction rate upper bounds $\lambda^+_{J'K}$ and $\lambda^+_{KJ'}$ for all occupied buckets $K \in \mathcal{B}$.

3.iv. For all occupied buckets $K \in \mathcal{B}$, update the cumulative interaction rate bounds λ^+_K incrementally to account for any changes to n_J, $n_{J'}$, λ^+_{KJ}, and $\lambda^+_{KJ'}$, and recompute λ^+ by summation of all $n_J \lambda^+_J$.

4.3.6 Data structures

The efficiency of the algorithm presented in Sect. 4.3.5 is bounded by the efficiency of its various mapping and enumeration steps. The data structures chosen to represent the simulation state are thus of critical importance.

In our reference implementation, we represent the simulation state with the following data structures:

- An array $(\mathbf{q}_j)_{j\in\mathcal{I}}$ of length n which holds the entity properties for all entities $j \in \mathcal{I}$.
- The upper bound for the total interaction rate λ^+.
- An array $(D_J)_{J\in\mathcal{B}}$ of length ν which for every occupied bucket $J \in \mathcal{B}$ stores a tuple

$$D_J := \left(J, Q_J, \mathcal{I}_J, (\lambda^+_{JK})_{K\in\mathcal{B}}, \lambda^+_J\right) \tag{4.53}$$

  holding the bucket label J, the bucket properties Q_J, an array of entities $\mathcal{I}_J$, an array of bucket–bucket interaction data upper bounds $(\lambda^+_{JK})_{K\in\mathcal{B}}$, and the cumulative bucket interaction rate upper bound λ^+_J.
  The order of the entity indices in the $\mathcal{I}_J$ arrays is arbitrary. The order of the $(D_J)_{J\in\mathcal{B}}$ array is arbitrary as well but must coincide with the order of the arrays of bucket–bucket interaction data upper bounds $(\lambda^+_{JK})_{K\in\mathcal{B}}$ for all $J \in \mathcal{B}$.
- An array $(J, b_J)_{J\in\mathcal{B}}$ of length ν which for every occupied bucket $J \in \mathcal{B}$ stores the index b_J of the corresponding element in the $(D_J)_{J\in\mathcal{B}}$ array.
  The array is ordered by J.
- An array $(E_j)_{j\in\mathcal{I}}$ of length n which for every entity $j \in \mathcal{I}$ stores a tuple

$$E_j := (B_j, o_j) \tag{4.54}$$

  holding the bucket labels B_j and the offsets o_j at which the entity indices are stored in the array held by D_{B_j}.

The relations between the arrays $(D_J)_{J\in\mathcal{B}}$, $(J, b_J)_{J\in\mathcal{B}}$, and $(E_j)_{j\in\mathcal{I}}$ are exemplified in Fig. 4.4. With these data structures, the bucketing algorithm can be implemented very efficiently, as we shall explore in the next section.

Although a particular choice of data structures is discussed here, the bucketing algorithm could of course be implemented differently. For example, instead of storing an array of entries $\mathcal{I}_J$ for every bucket $J \in \mathcal{B}$, all entries could be stored in a single contiguous array ordered by bucket index, thereby reducing the amount of memory reallocations required while increasing the cost of moving entities between buckets and the cost of adding or removing buckets.

4.3.7 Cost model

In the following we estimate the memory requirement and the computational cost of simulating a compound Poisson process with the bucketing scheme when implemented with the choice of data structures described in Sect. 4.3.6.

The bucket properties Q_J and the computation of upper bounds for bucket–bucket interaction rates is discussed in Sect. 4.4. For now, without specifying them any further, we only make two assumptions for the purpose of assessing the cost of the simulation. Firstly, given a bucket $J \in \mathcal{B}$ holding n_J entities, we assume that the bucket properties

Q_J can be computed by traversal of the entity properties $\mathbf{q}_j$ of all entities in the bucket $j \in \mathcal{I}_J$, and therefore with n_J computational steps. Secondly, given the bucket properties Q_J, Q_K for two buckets $J, K \in \mathcal{B}$, we assume that an upper bound for the bucket–bucket interaction rate λ^+_{JK} can be computed in constant time, that is, with computational effort independent of the number of entities held by buckets J and K.

In the bucketing scheme, the same n vectors of entity properties $\mathbf{q}_j$ as in the traditional sampling scheme need to be tracked, but in addition, only ν^2 upper bounds of bucket–bucket interaction rates as well as ν elements of per-bucket data such as bucket properties and cumulative interaction rate upper bounds must be stored, amounting to an approximate memory requirement of

$$C^{\mathrm{B}}_{\mathrm{mem}} = \nu^2 + (\alpha + 1) \cdot n + \beta \cdot \nu \tag{4.55}$$

floating-point values, where $\alpha = \dim \mathbb{Q}$ again is the number of floating-point values needed to represent the properties $\mathbf{q}_j$ of some entity j, and β represents the number of floating-point values required to store the properties Q_J of some bucket J.

In addition to the elementary computational costs defined in Sect. 4.2.8, we define the following additional costs specific to the bucketing scheme:

- C_Q, the cost of updating the set of bucket properties Q_J for a given bucket $J \in \mathcal{B}$ to account for an entity being added to or removed from the bucket;
- C_{λ^+}, the cost of computing an upper bound for the bucket–bucket interaction rate from two sets of bucket properties Q, Q'.

The proposed algorithm of the bucketing scheme employs dynamically sized data structures. The runtime cost and memory overhead of dynamic memory allocation is neglected in the following discussion.

To set up a simulation, we first traverse the list of entities $j \in \mathcal{I}$, evaluate the bucketing criterion $B(\mathbf{q})$ to determine the corresponding bucket label $B_j \leftarrow B(\mathbf{q}_j)$ for every entity, store the label in the E_j array, and store the list of all bucket labels in the sorted array of tuples $(J, b_J)_{J \in \mathcal{B}}$. This can be done with amortised $(n + \nu \log \nu)$ operations by accumulating and subsequently sorting occupied bucket labels. For every occupied bucket we allocate an entry in the D_J array which we refer to with the b_J index. We then iterate through the list of entities $j \in \mathcal{I}$ a second time, look up the bucket labels B_j in the E_j array, locate the bucket in the (J, b_J) array with binary search, and append the entity to the bucket's list of entity indices $(j)_{j \in \mathcal{I}_J}$, storing the entity index position in o_j, which entails an average number of $n \log \nu$ steps. With every bucket having available a list of the entities in it, we can now iterate through the array of buckets and compute and store the bucket properties Q_J for every bucket $J \in \mathcal{B}$ at a total cost of $\sum_J n_J \, C_Q = n \, C_Q$. Finally, we iterate over all pairs of buckets $J, K \in \mathcal{B}$ and compute and append the bucket–bucket interaction rate upper bounds λ^+_{JK} to the array $(\lambda^+_{JK})_{K \in \mathcal{B}}$ of bucket J, which has a cost of $\nu^2 \, C_{\lambda^+}$. With ν^2 arithmetic operations, we also compute the cumulative bucket interaction rate upper bounds λ^+_J and the total interaction rate upper bound λ^+. The total

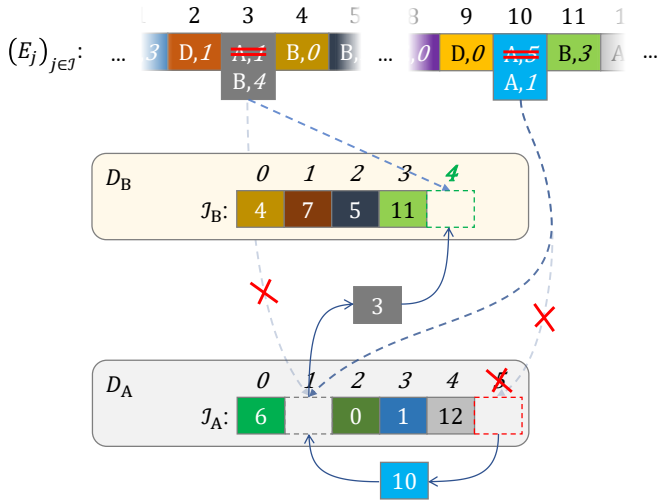


Figure 4.5: Example demonstrating how to move an entity to a different bucket in $\mathcal{O}(1)$ steps.
To move entity 4 from bucket A to bucket B, append an element to the list of entities in bucket B, and move the last entity in bucket A to the previous location of entity 4. Only two elements in the array $(E_j)_{j\in\mathcal{I}}$ have to be updated. Element movement is symbolised with solid lines, and element referral is indicated by dashed lines. Addition and removal of entries to and from arrays is indicated with boxes of green and red colour, respectively.

cost of initialising the simulation thus amounts to

$$\begin{aligned} C^{\mathrm{B}}_{\mathrm{init}} = &\left(n + \nu + \nu^2 + \nu\log\nu + n\log\nu\right) C_{\mathrm{op}} \\ &+ n\, C_Q \\ &+ \nu^2\, C_{\lambda^+} \,. \end{aligned} \tag{4.56}$$

Sampling an event candidate requires drawing one uniform random number for choosing an interarrival time, two random numbers and ν additions on average for determining a pair of buckets, and another two random numbers for choosing entity indices. An event candidate can be accepted or rejected, which is decided by computing the interaction rate between the entities sampled to determine the acceptance probability (Eq. (4.51)) and by choosing whether to accept by drawing another random number. Let us assume that the average probability of acceptance is $p \in (0, 1]$; then, for one event to be simulated, p^{-1} candidates have to be drawn on average, and the cost of sampling an event therefore is

$$\begin{aligned} C^{\mathrm{B}}_{\mathrm{event}} \approx\ &p^{-1}\nu\, C_{\mathrm{op}} \\ &+ p^{-1}\left(6\, C_{\mathrm{rand}} + C_\lambda\right) \\ &+ C_{\mathrm{action}} \,. \end{aligned} \tag{4.57}$$

When the properties $\mathbf{q}_j$ of an entity j are changed either by an interaction or by external events, the bucket properties Q_J of the corresponding bucket $J = B_j$, all bucket–

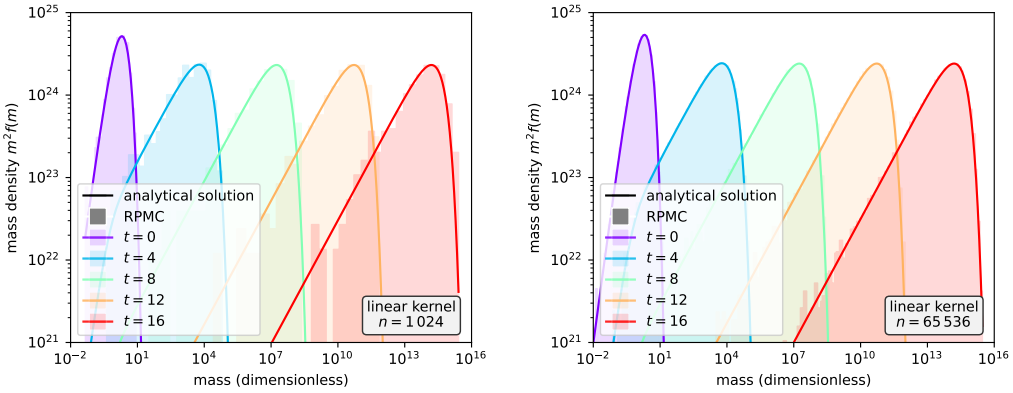


Figure 4.6: Analytical solutions and RPMC simulations for the coagulation test with the linear kernel (Eq. (4.63)) using the bucketing scheme. The RPMC simulation runs use $n = 1\,024$ and $n = 65\,536$ representative particles, respectively.

bucket interaction rate bounds involving bucket J, and all cumulative bucket interaction rate bounds need to be updated. Locating the bucket of entity j in the sorted array of tuples $(J, b_J)_{J \in \mathcal{B}}$ with binary search requires $\log \nu$ comparisons on average, and updating the upper bound for the interaction rate λ^+ requires recomputing and summing up an average number of ν bucket–bucket interaction rate bounds, amounting to an approximated cost of

$$\begin{aligned} C^{\mathrm{B}}_{\mathrm{update}} \approx\ & C_Q \\ & + (\nu + \log \nu)\, C_{\mathrm{op}} \\ & + \nu\, C_{\lambda^+} \ . \end{aligned} \tag{4.58}$$

Entities sometimes need to be moved to another bucket, and as a consequence, buckets sometimes need to be added and removed. With the data structures adopted for our reference implementation, either operation can be performed in amortised $\mathcal{O}(1)$ steps with a manoeuvre sketched in Fig. 4.5 which we refer to as 'castling' because of its vague resemblance of the eponymous chess move.

Following the reasoning in Sect. 4.2.8, we estimate the simulation cost rate as

$$\frac{C^{\mathrm{B}}_{\mathrm{sim}}(\Delta t)}{\Delta t} = \lambda \left(C^{\mathrm{B}}_{\mathrm{event}} + C^{\mathrm{B}}_{\mathrm{update}} \right) + \lambda_{\mathrm{ext}} C^{\mathrm{B}}_{\mathrm{update}} \tag{4.59}$$

$$\begin{aligned} & \approx \left(p^{-1} n\, \nu \right) \tilde{\lambda}\, C_{\mathrm{op}} \\ & + \left(p^{-1} n \right) \tilde{\lambda}\ (6\, C_{\mathrm{rand}} + C_\lambda) \\ & + (n\, \nu) \left[\tilde{\lambda} + \tilde{\lambda}_{\mathrm{ext}} \right] (C_{\lambda^+} + C_{\mathrm{op}}) \\ & + n \left[\tilde{\lambda}\, (C_Q + C_{\mathrm{action}}) + \tilde{\lambda}_{\mathrm{ext}} C_Q \right] , \end{aligned} \tag{4.60}$$

where a $(n \log \nu) \left[\tilde{\lambda} + \tilde{\lambda}_{\mathrm{ext}} \right] C_{\mathrm{op}}$ term has been neglected.

The cost estimate is more complex than the estimate given for the traditional scheme in Eq. (4.26), but it is evident that no term scales with n^2 or with $n \log n$. However, we observe that, because buckets are allocated sparsely, there cannot be more occupied buckets than entities,

$$n \overset{!}{\geqslant} \nu \,, \tag{4.61}$$

and thus $n\,\nu$ is an upper bound to ν^2: the cost rate of the bucketing scheme thus scales at least quadratically with the number of buckets, just as the cost rate of the traditional implementation scales quadratically with the number of entities.

The average probability of acceptance p quantifies the efficiency of the bucketing scheme. For an inefficient bucketing scheme, p is very small, and hence p^{-1} is large, allowing the first two terms in Eq. (4.60), which comprise the cost of rejection sampling, to dominate the simulation cost rate. Conversely, if the bucketing scheme is efficient, p, and thus p^{-1} as well, approaches unity, allowing the latter two terms in Eq. (4.60) to dominate. If we adjust the balance between rejection sampling and updating by refining or coarsening the bucketing scheme, or by making the bucketing scheme slightly hysteretic (cf. Sect. 4.4.5), then we can achieve better overall performance, as studied in Sect. 4.6.3.5.

4.4 Computing interaction rate bounds

In our description of the bucketing algorithm in Sect. 4.3.5, two aspects had remained unspecified: the nature and purpose of the bucket properties Q_J, and the computation of upper bounds for bucket–bucket interaction rates λ^+_{JK} from bucket properties Q_J, Q_K of two buckets $J, K \in \mathcal{B}$.

The definition of bucket–bucket interaction rate upper bounds given in Eq. (4.45) suggests that bounds could be computed as

$$\lambda^+_{JK} \leftarrow \max_{j \in \mathcal{I}_J, k \in \mathcal{I}_K} \lambda_{jk} \tag{4.62}$$

for two given buckets $J, K \in \mathcal{B}$. Naïvely, this would require the evaluation of the entity–entity interaction rate λjk for all pairs of entities $j \in \mathcal{I}_J$ and $k \in \mathcal{I}_K$, and therefore $n_J \cdot n_K$ evaluations of the entity interaction rate function $\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta)$. Because, for ν occupied buckets, a total number of ν^2 bucket–bucket interaction rate upper bounds need to be computed, and because $\sum_J n_J = n$, this amounts to n^2 evaluations of $\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta)$, which would be prohibitively expensive. But Eq. (4.62) can be evaluated much more efficiently, as we shall now demonstrate with an example.

4.4.1 Example: Linear kernel

The *linear kernel*

$$\lambda(m, m') = \lambda_0 \cdot \left(m + m'\right) \tag{4.63}$$

with some constant λ_0 leads to a special case of the Smoluchowski equation (Eq. (4.1)) for which the equation can be solved analytically (e.g. Ohtsuki et al., 1990), and which

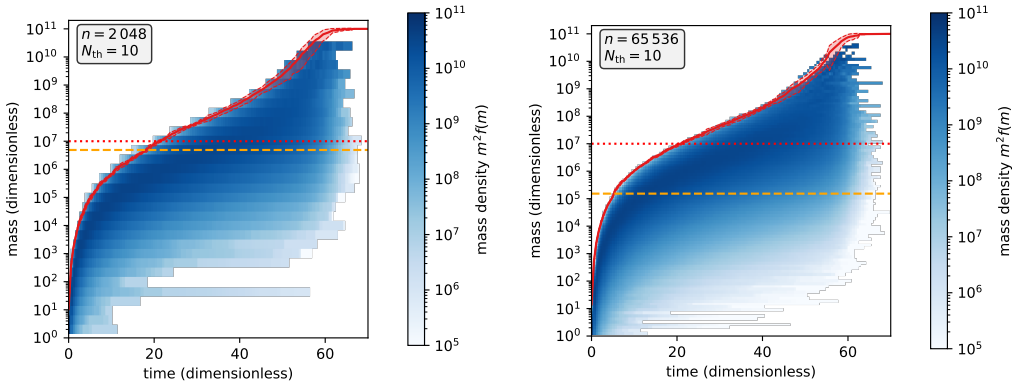


Figure 4.7: RPMC simulations for the coagulation test with the runaway kernel (Eq. (4.69)) using the bucketing scheme.
The RPMC simulation runs use $n = 2\,048$ and $n = 65\,536$ representative particles, respectively, and a particle regime threshold of $N_{\mathrm{th}} = 10$. The dimensionless threshold mass of $m_{\mathrm{th}} = 10^7$, marking the point where runaway growth ensues, is indicated by the dotted red horizontal lines, and the dashed orange lines indicates the representative particle mass $M/(nN_{\mathrm{th}})$ at which swarms are split up into individual particles. The plots show time series of histograms of the mass-weighted particle number density with bin counts colour-encoded on a logarithmic scale. The runaway particle is shown separately (red curve). Results have been averaged over 10 runs; the shaded surroundings of the red curve indicate the error bounds of the runaway particle mass.

is hence often used to verify numerical methods for the simulation of coagulation processes. In a representative particle simulation where all swarms are many-particle swarms, $N_k \gg 1\ \forall k \in \mathcal{I}$, the particle–swarm interaction rate λ_{jk} would then be

$$\lambda_{jk} = N_k \lambda_0 \cdot (m_j + m_k) \tag{4.64}$$

as per Eq. (4.16). Inserting this expression into Eq. (4.62), we find that a bucket–bucket interaction rate bound can be computed as a function of bucket-specific upper bounds,

$$\begin{aligned}
\lambda_{JK}^{+} &\leftarrow \max_{j\in\mathcal{I}_J, k\in\mathcal{I}_K} \lambda_0 \left(N_k m_j + M_k\right) \\
&= \lambda_0 \left[\max_{k\in\mathcal{I}_K} N_k \max_{j\in\mathcal{I}_J} m_j + \frac{M}{n}\right] \\
&= \lambda_0 \left[N_K^{+} m_J^{+} + \frac{M}{n}\right] ,
\end{aligned} \tag{4.65}$$

where we used the swarm mass $M_k = N_k m_k$, assumed an equal-mass sampling where every swarm holds the same fraction of the total mass $\mathcal{M}$,

$$M_k = \frac{\mathcal{M}}{n}\ \forall k \in \mathcal{I} , \tag{4.66}$$

and where the notation

$$x_K^- \coloneqq \min_{k \in \mathcal{I}_K} x_k \ , \quad x_K^+ \coloneqq \max_{k \in \mathcal{I}_K} x_k \tag{4.67}$$

refers to the bucket-specific lower and upper bounds x_K^-, x_K^+ of some property x_k for entities k in a bucket K. Computing a bucket-specific upper bound only requires traversal of all entities in the bucket, and hence n_J operations for a bucket J, or n operations for all buckets. In the case of the linear kernel, we could therefore define the bucket properties as a tuple

$$Q_J \coloneqq \left(m_J^+, N_J^+\right) \ , \tag{4.68}$$

which for any bucket J can be computed with n_J operations. Subsequently, given the bucket properties Q_J, Q_K of two buckets J and K, a bucket–bucket interaction rate upper bound λ_{JK}^+ can be computed directly as per Eq. (4.65), thereby meeting the complexity requirements stated in Sect. 4.3.7. We note that the upper bound given by Eq. (4.65) is *optimal*: it is an exact upper bound on the particle–swarm interaction rates λ_{jk} for $j \in \mathcal{I}_J$, $k \in \mathcal{I}_K$, or in other words: no better bound exists. This quality is difficult to retain except in the simplest of cases.

Results for an RPMC simulation of coagulation with the linear kernel using the bucketing scheme are shown in Fig. 4.6. The analytical solution is given as a reference.

4.4.2 Example: Runaway kernel

As another example, we consider the *runaway kernel*, a test kernel modelled after the gravitational focussing effect (cf. e.g. Armitage, 2017, §III.B.1) which was introduced in Sect. 3.6.2 for the purpose of studying the swarm regime transition:

$$\lambda(m, m') = \lambda_0 \max(m, m')^{2/3} \left[1 + \left(\frac{\max(m, m')}{m_{\mathrm{th}}}\right)^{2/3}\right] \ , \tag{4.69}$$

where λ_0 is some constant and m_{th} is a threshold mass.

As with the linear kernel, an upper bound of the bucket–bucket interaction rate can be given in terms of bucket-specific bounds,

$$\lambda_{JK}^+ \leftarrow \lambda_0 N_K^+ \left(m_{J,K}^+\right)^{2/3} \left[1 + \left(\frac{m_{J,K}^+}{m_{\mathrm{th}}}\right)^{2/3}\right] \ , \tag{4.70}$$

where we used the abbreviation $m_{J,K}^+ \equiv \max\{m_J^+, m_K^+\}$, the submultiplicativity of the maximum norm,

$$\max_i (a_i \cdot b_i) \leqslant \max_i a_i \cdot \max_i b_i \tag{4.71}$$

for sequences of positive semi-definite quantities $(a_i)_i$, $(b_i)_i$, and the monotonicity of exponentiation. Although this upper bound is no longer exact[2], it can be obtained directly

[2] As a counterexample, let us consider $M = 24$, $n = 3$, $m_1 = \frac{1}{8}$, $m_2 = 1$, $m_3 = 8$, and a two-bucket grouping $\mathcal{B} = \{\mathrm{A}, \mathrm{B}\}$ with $\mathcal{I}_\mathrm{A} = \{1, 3\}$ and $\mathcal{I}_\mathrm{B} = \{2\}$. Then, the upper bound of the mutual interaction

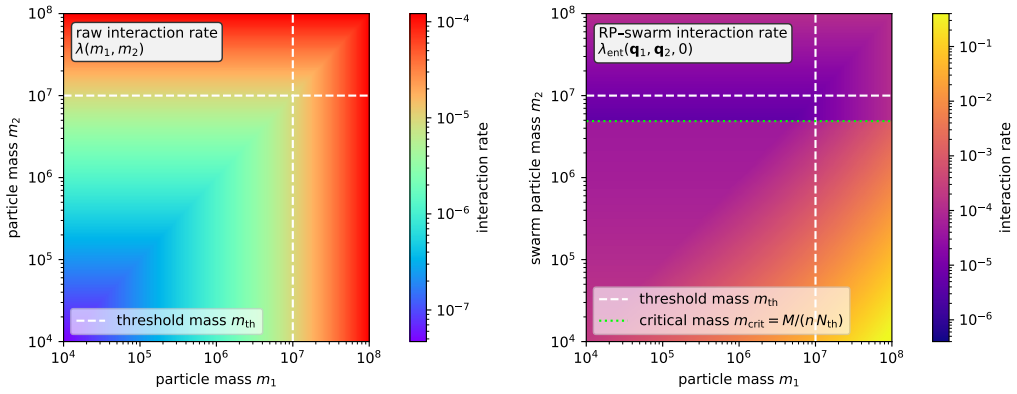


Figure 4.8: Raw interaction rate $\lambda(\mathbf{q}, \mathbf{q}')$ and entity interaction rate $\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta)$ for the runaway kernel (Eq. (4.69)).
Parameters are: $\lambda_0 = 10^{-10}$, threshold mass $m_{\text{th}} = 10^7$, total mass $\mathcal{M} = 10^{11}$, $n = 2\,048$ representative particles, particle regime threshold $N_{\text{th}} = 10$. Many-particles swarms ($N(\mathbf{q}) > N_{\text{th}}$) are assumed to have homogeneous swarm mass $M(\mathbf{q}) = \mathcal{M}/n$, whereas few-particles swarms ($N(\mathbf{q}) \leqslant N_{\text{th}}$) are assumed to have been split up to represent only themselves, $N(\mathbf{q}) = 1$, and thus have swarm mass $M(\mathbf{q}) = m(\mathbf{q})$; hence the discontinuity at m_{crit}.

from bucket properties

$$Q_J := \left(m_J^+, N_J^+\right) \ , \tag{4.72}$$

and thus also meets the complexity requirements stated in Sect. 4.3.7.

Results for an RPMC simulation of coagulation with the runaway kernel using the bucketing scheme are shown in Fig. 4.7. The runaway kernel is shown in Fig. 4.8 along with the emerging entity interaction rate $\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta)$. For comparison, two snapshots from the RPMC simulation are given in Fig. 4.9. In both figures, a discontinuity can be observed as swarms N_k are split up once their swarm particle count N_k no longer exceeds the particle regime threshold N_{th}. Self-representing particles, that is, representative particles k with a swarm particle count of $N_k = 1$, cannot interact with their own swarm, hence their self-interaction rate is $\lambda_{kk} = 0$, as seen in the lower left panel of Fig. 4.9. Indices are grouped by buckets, and the corresponding bucket–bucket interaction rate bounds λ_{JK}^+ are shown in the right panels of Fig. 4.9.

rate for buckets A, B obtained with Eq. (4.70) is $\lambda_{\text{AB}}^+ = 2\,176$, whereas the exact upper bound would be $\lambda_{\text{AB}}^+ = 160$.
We emphasise that, to demonstrate the inexactness of the bound in a simple example, we had to use a counterintuitive bucket grouping, with the heaviest and the lightest particle in the same bucket and the average-mass particle in another bucket. The excessiveness of bounds is attenuated by a bucketing scheme that is monotonic in every dimension of the bucket label.

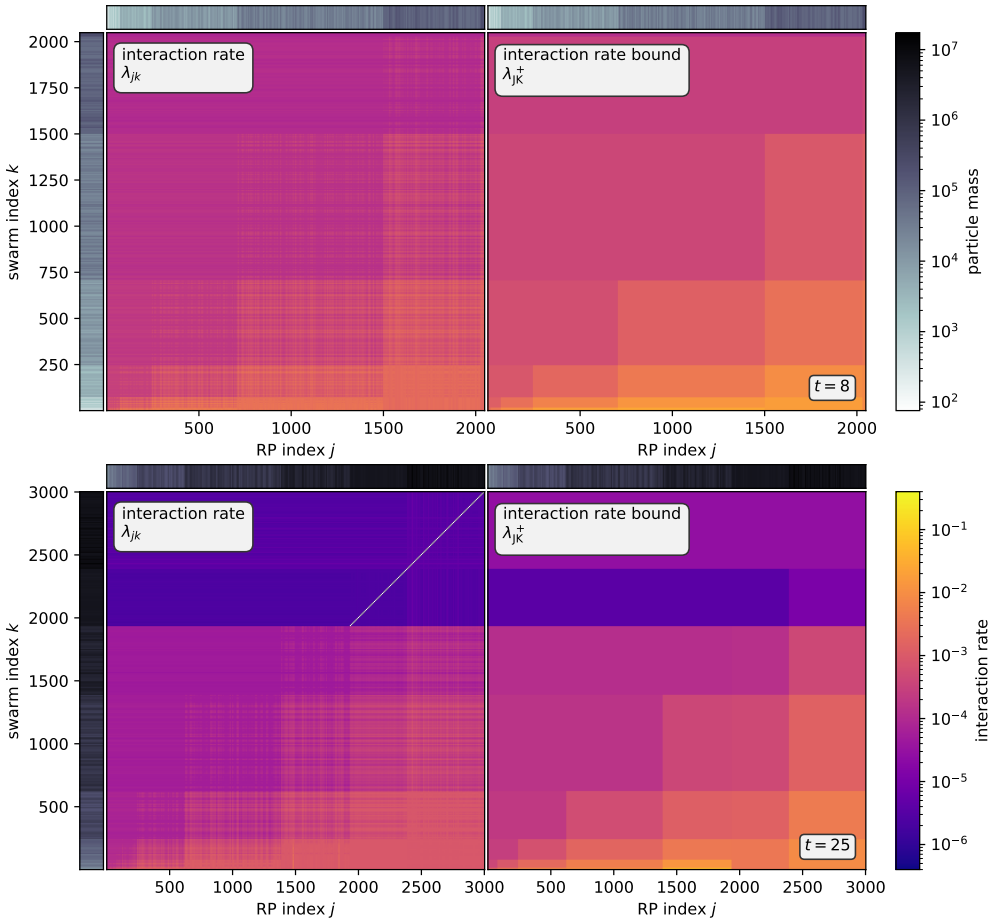


Figure 4.9: Snapshots of the entity interaction rates λ_{jk} and corresponding bucket–bucket interaction rate bounds λ^+_{JK} for the runaway kernel (Eq. (4.69)) in an RPMC simulation at different times $t = 8$, $t = 25$.
The particle masses of representative particles j and k are indicated at the sides. Parameters as in Fig. 4.8; bucketing as per Eq. (4.42) using $\theta_M = \theta_m = 2$.

4.4.3 General interaction rate bounds

We demonstrated how to efficiently compute upper bounds for the bucket–bucket interaction rates for the examples given in Sects. 4.4.1 and 4.4.2 by defining a set of bucket-specific upper bounds as bucket properties, inserting the respective interaction rate function into Eq. (4.62), and rewriting the upper-bound estimate as a function of the bucket properties. The generalisation of this approach leads to the concept of *interval arithmetic*, which is briefly introduced in Appendix 4.A along with interval notation such as $[x] \equiv [x^-, x^+]$ and the Fundamental Theorem of Interval Arithmetic. For a more thorough introduction to interval arithmetic, refer to Hickey et al. (2001); Moore et al. (2009); Alefeld and Herzberger (2012).

By virtue of the Fundamental Theorem of Interval Arithmetic, we can obtain an interval extension of the linear kernel (Eq. (4.63)),

$$\Lambda\big([m]\,,[m']\big) = \lambda_0 \cdot \big([m] + [m']\big) \ , \tag{4.73}$$

and likewise of the runaway kernel (Eq (4.69)),

$$\begin{aligned}\Lambda\big([m]\,,[m']\big) = \lambda_0 \cdot \big(\mathrm{Max}\big([m]\,,[m']\big)\,\big)^{2/3} \\ \times \left[1 + \left(\frac{\mathrm{Max}([m]\,,[m'])}{m_\mathrm{th}}\right)^{2/3}\right] \ ,\end{aligned} \tag{4.74}$$

where the exact interval extension of the max function is given by

$$\mathrm{Max}([x]\,,[y]) = \big[\max\big(x^-, y^-\big)\,, \max\big(x^+, y^+\big)\big] \ . \tag{4.75}$$

Likewise, for the RPMC method with any raw collision rate function $\lambda(\mathbf{q}, \mathbf{q}')$ with an interval extension $\Lambda([\mathbf{q}], [\mathbf{q}'])$, an interval extension of the entity interaction rate function $\lambda_\mathrm{ent}(\mathbf{q}, \mathbf{q}', \delta)$ in Eq. (4.16) can be obtained as

$$\Lambda_\mathrm{ent}\big([\mathbf{q}]\,,[\mathbf{q}']\,,\Delta\big) \coloneqq N^\mathrm{eff}\big([\mathbf{q}]\,,[\mathbf{q}']\,,\Delta\big)\ \Lambda\big([\mathbf{q}]\,,[\mathbf{q}']\big) \ , \tag{4.76}$$

where $[\mathbf{q}]$ denotes a vector of intervals, and where $\Delta \subseteq \{0, 1\}$, $\Delta \neq \varnothing$ is a set containing 1 if the interaction can be an entity self-interaction and 0 if the interaction can be an interaction between different entities (and may thus contain both 0 and 1). Equivalently, we might state

$$[\lambda_{JK}] = \Big[N^\mathrm{eff}_{JK}\Big]\, \Lambda([\mathbf{q}_J]\,,[\mathbf{q}_K]) \ , \tag{4.77}$$

where we abbreviate the effective swarm particle count and its interval extension as

$$N^\mathrm{eff}_{jk} \coloneqq N^\mathrm{eff}(\mathbf{q}_j, \mathbf{q}_k, \delta_{jk}) \ , \tag{4.78}$$

$$\Big[N^\mathrm{eff}_{JK}\Big] \coloneqq N^\mathrm{eff}([\mathbf{q}_J]\,,[\mathbf{q}_K]\,,\Delta_{JK}) \ , \tag{4.79}$$

and where the set-valued self-interaction indicator Δ_{JK}, the set extension of δ_{jk}, is given by

$$\Delta_{JK} \coloneqq \begin{cases} \{0\} & \text{if } J \neq K \\ \{1\} & \text{if } J = K \text{ and } n_J = 1 \\ \{0, 1\} & \text{if } J = K \text{ and } n_J > 1 \ , \end{cases} \tag{4.80}$$

in accordance with the definition of Δ given above: considering that we know only the buckets J, K which hold the interacting entities, we know that the interaction cannot be an entity self-interaction if the interacting entities are in different buckets, and that it is necessarily an entity self-interaction if both are in the same bucket and if that bucket only holds one entity. Otherwise, both self-interaction and non-self-interaction are possible.

In Sect. 3.4.4, the notion of *boosting* was introduced as a means of grouping together multiple similar events, and the *boosted interaction rate* was defined as

$$\lambda_{jk} \coloneqq N_{jk}^{\mathrm{b}} \lambda(\mathbf{q}_j, \mathbf{q}_k) \ , \tag{4.81}$$

where β_{jk}, indicating the number of events grouped together, is the *boost factor* for an interaction of representative particle j with swarm k, and where we introduced the *boosted swarm multiplicity factor* N_{jk}^{b} as

$$N_{jk}^{\mathrm{b}} \coloneqq \frac{N_{jk}^{\mathrm{eff}}}{\beta_{jk}} \overset{!}{=} N_{jk} \, V_{jk}^{\mathrm{eff}} \tag{4.82}$$

with the *boosted swarm particle count* $N_{jk} = N_k / \beta_{jk}$ and the *effective swarm particle count correction factor* $V_{jk}^{\mathrm{eff}} = N_{jk}^{\mathrm{eff}} / N_k$. A large value of β_{jk} counteracts a large number of swarm particles N_k, and hence an interval extension of λ_{jk} can be obtained using the algebraically favourable rendering of Eq. (4.82),

$$[\lambda_{JK}] \coloneqq [N_{JK}] \cdot \left[V_{JK}^{\mathrm{eff}} \right] \cdot \Lambda([\mathbf{q}_J], [\mathbf{q}_K]) \ . \tag{4.83}$$

Here, the interval extensions of $[N_{JK}]$ and $[V_{JK}^{\mathrm{eff}}]$ are obtained by composition as per the Fundamental Theorem of Interval Arithmetic, following the paradigm devised in subsequent Chapter 5 to obtain interval extensions of piecewise-defined functions such as $N^{\mathrm{eff}}(\mathbf{q}, \mathbf{q}', \delta)$.

We thus define the bucket properties Q_J for a bucket J as a vector of intervals of property bounds,

$$Q_J \coloneqq [\mathbf{q}_J] \ . \tag{4.84}$$

We note, however, that it may be useful, and allow for tighter results in the interval extension of the interaction rate function, if we augment Q_J with intervals of bounds of additional derived properties, as done in Eq. (4.68) for the linear kernel where we had defined the bucket properties Q_J to include bounds for the particle property m_j as well as the derived property $N_j = (M/n)/m_j$.

4.4.4 Updating bucket properties

We computed and stored an initial set of bucket property bounds $Q_J = [\mathbf{q}_J]$ for all occupied buckets $J \in \mathcal{B}$ as the Cartesian product of element-wise intervals

$$[\mathbf{q}_J] \equiv [q_{J,1}] \times \cdots \times [q_{J,d}] \ , \tag{4.85}$$

$$[q_{J,s}] \equiv [q_{J,s}^-, q_{J,s}^+] \leftarrow \left[\min_{j \in \mathcal{I}_J} q_{j,s}, \max_{j \in \mathcal{I}_J} q_{j,s} \right] \ , \tag{4.86}$$

where s denotes the component index of the d-dimensional property vector $\mathbf{q}$. We sometimes abuse notation by employing the quantity itself as a placeholder for its index or its component, for instance writing $q_{j,m}$ instead of $q_{j,1}$, and use the component and the quantity interchangeably, $m_j = m(\mathbf{q}_j) = q_{j,m}$. For example, if entity properties consisted of a mass m and a charge c, the property vector would be $\mathbf{q}_j = (m_j, c_j)$; index $s \equiv m$ would refer to the mass dimension and index $s \equiv c$ would refer to the charge dimension.

During simulation, an interaction between entities j and k usually changes some of their properties $\mathbf{q}_j$, $\mathbf{q}_k$. When the properties of an entity j change, the bucket property bounds $[\mathbf{q}_J]$, $[\mathbf{q}_{J'}]$ of the entity's old and new buckets J and J' (which may be identical) may also change. This is not always the case, though. As an example, consider a bucket with label A which contains three particles with masses $m_1 = 1\,\mathrm{g}$, $m_2 = 3\,\mathrm{g}$, and $m_3 = 7\,\mathrm{g}$. Presume that particle 2 suffers an interaction that grows its mass to $m_2' = 4\,\mathrm{g}$. Then the bucket property bounds for bucket A does not change; the minimum and maximum mass of any particle in the bucket are still 1 g and 7 g. In this case, the interaction rate bounds, which are a function of the bucket property bounds (Eq. (4.76) in the case of the RPMC method), do not change either and thus need not be recomputed in Step 3.iii, and no updates need to be performed in Step 3.iv of the bucketing algorithm.

Now let us instead assume that particle 3 undergoes a collision and grows to mass $m_3' = 12\,\mathrm{g}$. If the bucketing criterion $B(\mathbf{q})$ still places particle 3 in the same bucket A, its bucket property bounds has to expand given that the maximum particle mass in the bucket is now 12 g; subsequently, the bucket interaction rate bounds for bucket J need to be recomputed in Step 3.iii, and the cumulative bucket interaction rate bounds need to be updated in Step 3.iv. Expanding the bucket property bounds is an operation with constant complexity, entailing only a $\min$ and $\max$ function call per bucket property, and thus is of negligible cost compared to the subsequent recomputation of interaction rate bounds and updating of cumulative interaction rate bounds.

Let us consider a third example: particle 3 still grows to $m_3' = 12\,\mathrm{g}$ but the bucketing criterion $B(\mathbf{q})$ now places it in a different bucket with label B. Let the bucket mass bounds of bucket B be $[m_\mathrm{B}] = [11\,\mathrm{g}, 18\,\mathrm{g}]$. The bucket property bounds of bucket B thus do not change, and no interaction rate bounds need to be recomputed in Step 3.iii. However, the number of entities n_A and n_B in buckets A and B have changed, and therefore in Step 3.iv the cumulative bucket interaction rate bounds must be updated.

So far we only described the expansion of bucket property bounds. It is possible that the bounds $[\mathbf{q}_J]$ currently stored for a bucket J are wider than necessary for the particles remaining in the bucket. In the third example considered above, this is the case for bucket A after particle 3 has been relocated to bucket B. Excessive property bounds lead to excessive interaction rate bounds and thus to an unnecessarily high rejection rate in the sampling step. We therefore strive to keep bucket property bounds tight. But while we could always recompute the bounds of a bucket J to keep them as tight as possible, this would come at a cost of n_J operations because all entities in the bucket would need to be processed. We therefore defer this recomputation by counting the number of times an entity in bucket J has changed since the last recomputation of bounds. Once the number exceeds the number of entities in the bucket n_J times the number of occupied buckets ν, the bucket property bounds $[\mathbf{q}_J]$ are recomputed as per Eqs. (4.85–4.86).

This recomputation then occurs every $n_J \cdot \nu$ updates at most, and thus has constant amortised computational cost per update. We also recompute the bounds if the number of rejected events exceeds a certain threshold such as n and if particles have been added to or removed from the bucket since the last recomputation. This recomputation also has constant amortised computational cost per simulated event.

4.4.5 Widening

If a simulation has to process a large number of events which only cause very slight changes to entity properties, it may spend a disproportionate fraction of its time updating bucket–bucket interaction rate bounds. In such a situation, it may be beneficial to choose slightly wider bucket bounds, which we refer to as *widening*. To widen a bucket means that, whenever a bucket's property bounds are recomputed, its bounds are extended a little beyond the minimum and maximum property values of all entities in the bucket. So, instead of initialising the bucket property bounds with the minimum and maximum as per Eq. (4.86), they are widened as

$$[q_{J,s}] \leftarrow \left[w_s^-\left(\min_{j\in\mathcal{I}_J} q_{j,s}\right), w_s^+\left(\max_{j\in\mathcal{I}_J} q_{j,s}\right) \right] , \tag{4.87}$$

where s again denotes a component (e.g. mass, charge) of the vector of properties $\mathbf{q}$. How property s is to be widened can be specified by the user-supplied widening functions $w_s^-(\cdot)$ and $w_s^+(\cdot)$. The widening strategy should align with the bucketing criterion; for example, if a property such as mass is used for logarithmic bucketing as in Eqs. (4.41) and (4.42), it is best widened multiplicatively:

$$w_m^\pm(m') = m' \cdot 10^{\pm f/\theta_m} , \tag{4.88}$$

where $f \in [0, 1)$ is a simulation parameter indicating by which fraction of their width buckets should overlap. Conversely, a property used as a linear bucketing criterion would best be widened additively. For example, for a linear bucketing criterion such as

$$B(\mathbf{q}) = \left\lfloor \theta_x \frac{x(\mathbf{q}) - x_0}{\delta x} \right\rfloor , \tag{4.89}$$

where δx is a characteristic length and θ_x is the simulation parameter controlling the bucket width, a suitable widening function would be

$$w_x^\pm(x') = x' \pm f\,\delta x . \tag{4.90}$$

With widening, bucket property bounds exceed the strict bounds of the property values of their entities. This usually leads to overlapping buckets, and hence to a hysteresis effect in the bucket assignment of entities; with widening, an entity j may still reside in a bucket B_j different from the bucket it would be assigned to if the bucketing criterion $B(\mathbf{q}_j)$ was re-evaluated with its current properties $\mathbf{q}_j$. Therefore, the likelihood is increased that a particle can remain in the same bucket after its properties undergo minuscule changes; bucket property bounds need to be updated (that is, extended or recomputed) less often, and the subsequent recomputations of bucket–bucket interaction

rate bounds in Step 3.iii of the bucketing algorithm are avoided, all at the expense of a higher likelihood of rejection. The widening parameter f can thus be used to balance the sampling and the updating parts of the simulation. The impact of this parameter is studied experimentally in Sect. 4.6.3.5.

4.4.6 Efficiency and correctness

As observed in Sect. 4.4.1, it is usually not possible to estimate exact bounds if the interaction rate function is non-trivial. This does not constitute a problem – the bucketing scheme only requires an upper bound, not necessarily an exact one – but can negatively impact performance. By overestimating the bucket–bucket interaction rate, the acceptance probability in Eq. (4.51) decreases, and as a result we have to sample a larger number of potential events which are ultimately rejected. For optimal sampling performance, we thus desire a tight estimate of the bucket–bucket interaction rate upper bound. In a simulation which entails a large number of events bringing about only minute changes, some sampling performance may be traded for fewer updates, as proposed in Sect. 4.4.5, by widening the bucket property bounds by some fraction.

Regardless of whether a bound is exact or inexact, the fraction of rejected events is high if the entity–entity interaction rates λ_{jk} between entities in two given buckets J, K span many orders of magnitude. The computational cost of the simulation therefore depends on a suitable definition of the bucketing criterion $B(\mathbf{q})$, chosen such that the interaction rates λ_{jk} between entities in any two buckets are of similar magnitude, and on a carefully crafted interval extension of the interaction kernel which avoids excess in the computation of bounds.

The bucketing criterion indirectly controls the number of occupied buckets ν, and the simulation cost scales linearly with ν according to Eq. (4.60). The bucketing criterion thus must find a compromise between a small number of buckets and a high rate of acceptance. With an inefficient bucketing criterion, the bucketing scheme can still be prohibitively expensive compared to the traditional inverse transform sampling scheme. But even with a suitable bucketing criterion, the traditional scheme may outperform the bucketing scheme for small numbers of entities n.

We conclude this section by again emphasising that the correctness of the scheme is in no way affected by the choice of a bucketing criterion or by the chosen upper bound estimate for the interaction rate. The bucketing scheme is statistically equivalent to the traditional sampling scheme; it does not impose additional physical or statistical requirements and does not introduce new approximations. The results of a simulation conducted with the bucketing scheme thus statistically matches the results obtained with a traditional computational scheme; the simulation may only execute faster if an efficient bucketing criterion was chosen and if the upper bound estimates are not excessive. The correctness of the results is verified explicitly in Sect. 4.4.1, where the simulation result is compared to an analytical solution in Fig. 4.6. We also emphasise that the bucketing scheme had been used to run all simulations conducted in Chapter 3, of which many had been compared to analytical solutions as well.

4.5 Locality

In simple geometric interaction models such as Eq. (4.13), the particle distribution is assumed to be homogeneous and isotropic. Often, however, a physical system features spatial inhomogeneities. For example, in a protoplanetary disk, the gas and dust density distribution may exhibit radial features such as *rings*, an amassment of dust grains, and *cavities*, a radial zone with little or no material (e.g. Huang et al., 2018).

To resolve spatial features in a stochastic simulation, we first augment the vector of entity properties $\mathbf{q}_j$ with one or more components representing the location of the entity. In this section, we shall assume that the location of an entity j is represented by a one-dimensional quantity, noting that our approach can be generalised to higher dimensionality.

4.5.1 Local bucketing criteria

Typically, discrete interactions between entities are modelled as having a maximal spatial distance up to which interactions are possible. For example, two bodies j and k can collide only if their distance at their closest approach, represented by the *impact parameter* b, is not greater than their *interaction radius* R_{coll},

$$b \overset{!}{\leqslant} R_{\text{coll}} \ , \tag{4.91}$$

which for massless spherical bodies is the sum of their bulk radii R_j and R_k,

$$R_{\text{coll}} = R_j + R_k \ . \tag{4.92}$$

If the bodies are massive, collisions are possible at impact parameters greater than $R_j + R_k$ because the mutual gravitational attraction boosts the effective collision cross-section, an effect known as *gravitational focussing* (e.g. Armitage, 2017, §III.B.1):

$$R_{\text{coll}} = (R_j + R_k)\,(1 + \Theta) \ , \tag{4.93}$$

where $\Theta = 2G(m_j + m_k)/[(R_j + R_k)v_{\text{a}}^2]$ is the Safronov number (e.g. Weidenschilling, 1989) with G the gravitational constant and v_{a} the *approach velocity*, the relative velocity at infinite distance. Assuming both bodies have circular planar orbits around the same central object, a lower bound for the impact parameter can be obtained as the difference of their semimajor axes,

$$b \geqslant |a_j - a_k| \ . \tag{4.94}$$

If the orbit of a body is eccentric, its apsides (the points nearest to and farthest from the central object) have an orbital distance of $a\,(1 - e)$ and $a\,(1 + e)$, respectively, where a and e are the semimajor axis and the eccentricity of the orbit. Therefore, for two entities j, k with eccentric orbits, the interaction radius is enhanced further,

$$R_{\text{coll}} = (R_j + R_k)\,(1 + \Theta) + a_j e_j + a_k e_k \ . \tag{4.95}$$

It is clear that, if the lower bound for the impact parameter between two entities exceeds their maximal interaction distance, they cannot possibly interact. The raw interaction rate function $\lambda(\mathbf{q}_j, \mathbf{q}_k)$ must take this into account and evaluate to 0 if the entities

j, k are not in reach, implying that $\lambda_{jk} = 0$ in this case. But this does not hold true for the interval extension $\Lambda([\mathbf{q}_J], [\mathbf{q}_K])$ if a non-local bucketing criterion is used which places entities with similar intrinsic properties in the same bucket regardless of their orbital position. For example, with the mass bucketing criterion of Eq. (4.41), the interactions between buckets J and K contribute a term $n_J n_K \lambda^+_{JK}$ to the upper bound of the global interaction rate λ^+ (Eq. (4.47)), implying that every entity in bucket J could possibly interact with any entity in bucket K with a maximal interaction rate of λ^+_{JK}. In reality, the number of entity pairs from buckets J and K which could possibly interact may be much smaller than $n_J n_K$ simply because many entity pairs would be out of reach. A non-local bucketing criterion therefore results in an excessive rejection rate, as events for out-of-reach entity pairs (j, k) are generated and must be rejected individually by evaluating $\lambda(\mathbf{q}_j, \mathbf{q}_k)$.

An obvious remedy would be to use a *local* bucketing criterion instead. A local variant of the mass bucketing criterion given in Eq. (4.41) might additionally use the location $a_j \equiv a(\mathbf{q}_j)$ as a criterion for bucket labeling,

$$B(\mathbf{q}) = \left(\left\lfloor \theta_m \log_{10} \frac{m(\mathbf{q})}{m_0} \right\rfloor, \left\lfloor \frac{a(\mathbf{q}) - a_0}{\delta a} \right\rfloor \right) \ , \tag{4.96}$$

where a_0 is some reference location, and where δa is the spatial width of a bucket. If the interval extension $\Lambda([\mathbf{q}_J], [\mathbf{q}_K])$ of the raw interaction rate is crafted with sufficient care, it evaluates to 0 if the entities in the two buckets J and K are always out of reach.

Although this approach is valid, it is not very practical. First, it is not clear how δa should be chosen. Let the interaction radius of two entities j, k be given by a function

$$R_{jk} \equiv R(\mathbf{q}_j, \mathbf{q}_k) \ . \tag{4.97}$$

We could then set the smallest occurrent interaction radius as δa,

$$\delta a \sim \min_{j,k \in \mathcal{I}} R_{jk} \ . \tag{4.98}$$

In a representative particle method, we usually have to impose a lower limit $R_{jk} \geqslant R_{\min}$, where the simulation parameter $R_{\min}$ represents the smallest length to be resolved by the simulation, because of the finite number of representative entities. This gives rise to a good approximation for Eq. (4.98),

$$\delta a \sim R_{\min} \ . \tag{4.99}$$

We note that this choice of δa would grow the number of occupied buckets ν dramatically. Even worse: as the number of representative particles n is increased, we are able to afford a better resolution; but shrinking $R_{\min}$, and thus δa, then grows the number of occupied buckets ν even further. As we had found in Sect. 4.3.7, the cost of the simulation strongly depends on the total number of occupied buckets ν: the memory requirements (Eq. (4.55)) scale with ν^2, and the simulation cost rate (Eq. (4.60)) scales with $n \cdot \nu$ in the dominant terms. Even though the number of out-of-reach rejections can be greatly reduced with a local bucketing criterion, it is questionable whether this outweighs the cost of updating all the additional buckets.

4.5.2 Local sub-buckets

We would like to retain the benefit of a local bucketing criterion – elimination of most out-of-reach rejections – without having to pay for an exploding number of buckets. This desire gives rise to the notion of *spatial sub-bucketing*. The basic idea is that the entities in a given bucket are grouped in sub-buckets according to their location. Interaction rate bounds are computed for bucket–bucket pairings as before, but when computing an upper bound of the global interaction rate λ, the sub-buckets are used to estimate a maximal number of entity pairs $n^+_{JK} \leqslant n_J n_K$ from buckets J and K which could possibly interact, resulting in a better bound of the global interaction rate λ,

$$\lambda^+_{\text{loc}} = \sum_J \sum_K n^+_{JK} \lambda^+_{JK} \tag{4.100}$$

$$\leqslant \sum_J \sum_K n_J n_K \lambda^+_{JK} = \lambda^+ \ , \tag{4.101}$$

cf. Eqs. (4.46–4.47). Ideally, $n^+_{JK} \ll n_J n_K$, and thus $\lambda^+_{\text{loc}} \ll \lambda^+$.

Let an entity j be associated with a dimensionless *location* $l_j \equiv l(\mathbf{q}_j)$. Given a sub-bucket s of bucket J and a second bucket K, we need an efficient means of locating and enumerating the sub-buckets t in bucket K which are in reach of sub-bucket s. A sub-bucket t is considered 'in reach' if any of the entities it might hold could possibly interact with any entity possibly in s. For reasons that become apparent as this predicate is specified more precisely, the rounding behaviour of floating-point arithmetic makes it unsuitable for determining the range of reachable sub-buckets; we therefore opt to employ integer arithmetic instead. Sub-bucket widths therefore need to be *granular*, that is, multiples of some common length scale. This length scale is represented as unity in the dimensionless location l; on this dimensionless scale, the width of a sub-bucket is always a positive integer, a multiple of 1.

Location l can be defined according to the nature of the spatial coordinate. For example, the location of a body on some circumstellar orbit could be defined in terms of its semimajor axis a as

$$l(\mathbf{q}) := d_{\text{min}} \frac{a(\mathbf{q}) - a_0}{R_{\text{min}}} \ , \tag{4.102}$$

where the integer-valued simulation parameter $d_{\text{min}} > 0$ is a granularity measure, a_0 is some reference position, and R_{min} is the minimal resolved length scale. $l_j \equiv l(\mathbf{q}_j)$ then represents the location of entity j relative to the reference position a_0 in units of $(R_{\text{min}}/w_{\text{min}})$.

In accordance with the definition of location $l(\mathbf{q})$, the sub-bucketing scheme requires a function

$$d_{jk} \equiv d(\mathbf{q}_j, \mathbf{q}_k) \tag{4.103}$$

which quantifies the *interaction distance* between entities j and k, and its interval extension

$$[d_{JK}] = D([\mathbf{q}_J], [\mathbf{q}_K]) \ . \tag{4.104}$$

A dimensionless equivalent of the interaction radius introduced in Sect. 4.5.1, the interaction distance is defined as the greatest location distance at which two entities with entity properties $\mathbf{q}$ and $\mathbf{q}'$ could possibly interact,

$$\left|l(\mathbf{q}) - l(\mathbf{q}')\right| \overset{!}{\leqslant} d(\mathbf{q}, \mathbf{q}') \ . \tag{4.105}$$

For a location defined as a linear function of the semimajor axis as in Eq. (4.102), the interaction distance is given in terms of the interaction radius (Eq. (4.97)) as

$$d(\mathbf{q}, \mathbf{q}') \coloneqq d_{\min} \frac{R(\mathbf{q}, \mathbf{q}')}{R_{\min}} \ . \tag{4.106}$$

Other definitions of location and distance are possible as long as the domain of locations is a one-dimensional metric space with a Euclidean distance measure. For example, if interaction radii tend to scale with the position, $R_{jj} \propto a_j$, then a logarithmic definition would be more appropriate. Such a definition is elaborated in Appendix 4.B.

After entities have been grouped into buckets, the width of the sub-buckets in a bucket J is chosen as

$$w_J \coloneqq \max\left\{d_{\min}, \left\lceil d_{JJ}^{+} \right\rceil\right\} \ . \tag{4.107}$$

With the linear definition of location and distance in Eqs. (4.102) and (4.106), imposing $d_{\min}$ as a lower bound ensures that a sub-bucket cannot be narrower than the minimum resolved length scale. The bucket self-interaction distance is a reasonable compromise for bucket size because, in most physical models, interaction radii are of vaguely additive nature,

$$2R(\mathbf{q}, \mathbf{q}') \sim R(\mathbf{q}, \mathbf{q}) + R(\mathbf{q}', \mathbf{q}') \ . \tag{4.108}$$

Given an entity j in bucket J, the *sub-bucket index* $s_j \equiv s_J(l_j)$ of j is then defined as

$$s_J(l) \coloneqq \left\lfloor \frac{l}{w_J} + \frac{1}{2} \right\rfloor \ . \tag{4.109}$$

For every bucket, the simulation maintains a dynamic array of records indexed with s, where every record stores a dynamic array of entity indices in that sub-bucket. As before, entities can be moved between buckets and sub-buckets with constant complexity using the 'castling' manoeuvre (cf. Fig. 4.5).

4.5.3 Sub-bucket reach

By inverting Eq. (4.109), we find the range of entity locations associated with a given sub-bucket index s to be the interval-valued function

$$L_J(s) \coloneqq w_J \left([s, s+1) - \frac{1}{2} \right) \ . \tag{4.110}$$

A sub-bucket s from bucket J and a sub-bucket t from bucket K shall be called *in reach* of each other if an entity that would be grouped in sub-bucket s could possibly be within

the rounded interaction distance bound $\lceil d^+_{JK} \rceil$ of any entity that would belong in sub-bucket t, as visualised in Fig. 4.10. Given a sub-bucket s in a bucket J, we can thus combine Eqs. (4.109) and (4.110) to obtain the interval of sub-bucket indices $U_{K,J}(s)$ in bucket K which are in reach of s:

$$U_{K,J}(s) \coloneqq S_K\big(L_J(s) \pm \lceil d^+_{JK} \rceil\big) \ , \tag{4.111}$$

where the '$\pm$' is sloppy interval notation,

$$\big[a, b\big) \pm c = \big[a - |c|, b + |c|\big) \ , \tag{4.112}$$

and where $S_K([l])$ is the interval extension of $s_K(l)$,

$$S_K([l]) = \left\lfloor \frac{[l]}{w_K} + \frac{1}{2} \right\rfloor \ . \tag{4.113}$$

Using the identity

$$\big\lfloor [a, b) \big\rfloor = \big[\, \lfloor a \rfloor \,, \lceil b - 1 \rceil \,\big] \ , \tag{4.114}$$

we insert definitions into Eq. (4.111) to find

$$U_{K,J}(s) = \left[\left\lfloor \frac{w_J\,(2s-1) - \big(2\lceil d^+_{JK} \rceil - w_K\big)}{2w_K} \right\rfloor , \left\lceil \frac{w_J\,(2s+1) + \big(2\lceil d^+_{JK} \rceil - w_K\big)}{2w_K} \right\rceil \right] . \tag{4.115}$$

According to Eq. (4.107), w_J and w_K are integers, and therefore both interval bounds in Eq. (4.115) are fractions of integers. To understand why this is significant, let us pretend that the unrounded interaction distance bound d^+_{JK}, rather than $\lceil d^+_{JK} \rceil$, had been used to define the notion of being 'in reach' in Eq. (4.111) and the sub-bucket width in Eq. (4.107). Floating-point additions, multiplications, and divisions are subject to rounding error, which can lead to significant differences in $U_{K,J}(s)$ if the value of the fractions in Eq. (4.115) is close to an integer. We note that being 'in reach' is a commutative relation,

$$t \in U_{K,J}(s) \quad \Leftrightarrow \quad s \in U_{J,K}(t) \ . \tag{4.116}$$

However, due to different accumulation of rounding errors in the different algebraic computations of $U_{K,J}(s)$ and $U_{J,K}(t)$, a sub-bucket t from bucket K might be deemed in reach of a sub-bucket s from bucket J, while at the same time s would not be considered in reach of t. In the incremental updating approach explained in the subsequent Sect. 4.5.4, this would violate consistency. Conversely, if both numerators and denominators in Eq. (4.115) are assured to be integer, the calculation can be done with integer arithmetic, and no rounding error can occur. (For completeness let us add that, due to the rounding guarantees provided by most floating-point formats, the calculation can actually be done in floating-point arithmetic so long as numerator and denominator remain integer and lie within the contiguous range of integers that can be represented exactly by the floating-point format.)

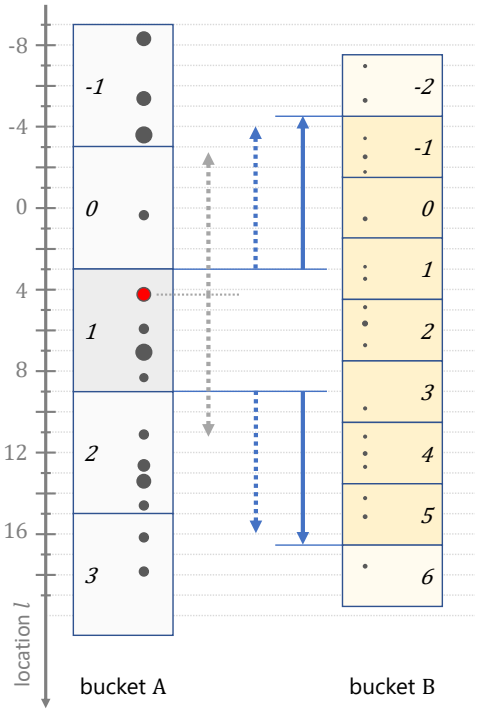


Figure 4.10: Sub-bucket grouping with interaction distance bounds.
The sub-buckets of two buckets A and B and the entities populating them are shown. Sub-bucket indices as per Eq. (4.109) are indicated in italic type. The arrows indicate: the range of possible A–B interactions, as per the bucket interaction distance bound d^+_{AB}, for the entity highlighted in red (gray, dotted); the range and the rounded range of possible interactions for sub-bucket 1 from bucket A (blue, dotted, and solid). In bucket B, the range of sub-buckets $U_{\mathrm{B,A}}(1)$ in reach of sub-bucket 1 from bucket A (Eq. (4.111)) are highlighted.

4.5.4 Bounding the number of possible interactions

Let now $n_{J,s}$ denote the number of entities in sub-bucket s of bucket J. To compute the upper bound of possibly interacting entity pairs n^+_{JK}, we have to iterate through all sub-buckets s in bucket J, obtain the sub-buckets $U_{K,J}(s)$ from bucket K which are in reach of s, and then accumulate the product $n_{J,s}n_{J,t}$ for all $t \in U_{K,J}(s)$:

$$\begin{aligned} n^+_{JK} &:= \sum_s \sum_{t \in U_{K,J}(s)} n_{J,s} n_{K,t} \\ &= \sum_s n_{J,s} n_{K,J,s} \ , \end{aligned} \tag{4.117}$$

where we abbreviate the number of entities in bucket K reachable from sub-bucket s of bucket J as

$$n_{K,J,s} := \sum_{t \in U_{K,J}(s)} n_{K,t} \ . \tag{4.118}$$

n^+_{JK} is commutative, $n^+_{JK} = n^+_{KJ}$, because the notion of being in reach is commutative as stated by Eq. (4.116). The interval bounds of the sequence of intervals $(U_{K,J}(s))_s$ are monotonic, which is to say: a sub-bucket $(s+1)$ does not reach further back than sub-bucket s, and sub-bucket s does not reach further forward than sub-bucket $(s+1)$. Therefore, the nested sum can be efficiently computed with a 'moving window' scheme. To this end, we iterate over sub-buckets s of bucket J. For every sub-bucket s, we compute the range of reachable sub-buckets $U_{K,J}(s)$ in bucket K. This range is compared to the range reachable by the previous sub-bucket $(s-1)$. We compute $n_{K,J,s}$ by taking the previous value $n_{K,J,(s-1)}$ and adding and subtracting sub-bucket entity counts $n_{J,t}$ for sub-buckets t as they fall in and out of reach. As we iterate through sub-buckets s, we cumulate the product $n_{J,s} n_{K,J,s}$. If the number of occupied sub-buckets in a bucket J are denoted ν_J, computing n^+_{JK} as per Eq. (4.117) requires only ν_J evaluations of the function $U_{K,J}(s)$ and two concomitant passes through the ν_K sub-buckets of bucket K.

When an entity j is updated in Step 3 of the bucketing algorithm, it may have to be moved from one bucket J to another bucket $J' \neq J$, or it may move from sub-bucket s to a different sub-bucket $s' \neq s$ of the same bucket $J' = J$. In either case, the cumulative values n^+_{JK} and $n^+_{J'K}$ must be updated, which can be done incrementally. Let $\delta n_{J,\tilde{s}}$ denote the number of entities to be added (or removed, if δ is negative) to (or from) some sub-bucket $\tilde{s}$ of some bucket J:

$$n_{J,\tilde{s}} \quad \to \quad {n_{J,\tilde{s}}}' \equiv n_{J,\tilde{s}} + \delta n_{J,\tilde{s}} \ . \tag{4.119}$$

To see how n^+_{JK} changes, we subtract the old value n^+_{JK} from the new value ${n^+_{JK}}'$:

$$\delta n^+_{JK} \equiv {n^+_{JK}}' - n^+_{JK} \ . \tag{4.120}$$

If $J = K$, the $\delta n_{J,\tilde{s}}$ may appear in both of the factors in Eq. (4.117). We therefore have to distinguish between $J = K$ and $J \neq K$. With some algebraic transformations we obtain

$$\delta n^+_{JK} = \begin{cases} \delta n_{J,\tilde{s}} \cdot n_{K,J,\tilde{s}} & \text{if } J \neq K \\ \delta n_{J,\tilde{s}} \cdot (\delta n_{J,\tilde{s}} + 2\, n_{K,J,\tilde{s}}) & \text{if } J = K \ , \end{cases} \tag{4.121}$$

where we used the number of reachable entities $n_{K,J,s}$ defined in Eq. (4.118). Just as the updated quantity n^+_{JK} itself, the incremental update δn^+_{JK} is commutative, $\delta n^+_{JK} = \delta n^+_{KJ}$. Computing δn^+_{JK} is less expensive than recomputing ${n^+_{JK}}'$, with the cost being near-proportional to the reach of sub-bucket $\tilde{s}$.

4.5.5 Sampling

If the global interaction rate bound λ^+ (Eq. (4.101)) is estimated as the sum of products $n_J n_K \lambda^+_{JK}$ for buckets J and K, every possible combination of entities $j \in \mathcal{I}_J$ and $k \in \mathcal{I}_K$

is considered with the same relative probability. The entity indices j and k can therefore be sampled independently, as done in Step 2.iv of the bucketing algorithm. However, if the sub-bucketing scheme is used, the local upper bound λ^+_{loc} (Eq. (4.100)) is computed instead as the sum of products $n^+_{JK}\lambda^+_{JK}$. We must therefore first sample the sub-bucket pairing s, t which contain the interacting entities j, k.

To sample s and t, we first draw a uniformly random integer $\zeta \in \mathbb{N}_0 \cap [0, n^+_{JK})$. We then use the 'moving window' scheme described in Sect. 4.5.4 to cumulate the sum over $n_{J,s}n_{K,J,s}$, stopping just before the cumulative value, which shall be denoted ψ, begins to exceed ζ. This determines the sub-bucket s of bucket J from which entity j is taken. Sub-bucket t of bucket K is determined by cumulating values $n_{K,t}$ for all sub-buckets in reach of sub-buckets s until the cumulative value exceeds $(\zeta - \psi)$. Once s and t have been determined, entity indices j and k are selected randomly with uniform probability from the arrays of entities held by the sub-buckets s and t.

4.6 Performance

The efficiency of the bucketing scheme has been studied formally in Sect. 4.3.7. In this section we empirically assess the computational efficiency of the scheme for two synthetic test cases as well as for a sophisticated physical model which comprises growth by coagulation, viscous stirring, and dynamical friction.

4.6.1 Linear kernel

In our first test, we simulate a coagulation process with the linear kernel (Eq. (4.63)) for different numbers of representative particles n using both the traditional inverse transform sampling scheme and the proposed bucketing scheme. Benchmark results for the linear kernel test are shown in Fig. 4.11, which shows the runtime for a single-threaded simulation of the linear kernel until $t = 16$; the specifications of the benchmark computer are given in Appendix 4.C. For the test we used simple mass bucketing as per Eq. (4.41) with two buckets per mass decade, $\theta_m = 2$. The bucketing is very efficient in this case; the average probability of event acceptance is measured to be $p \sim 0.18$, and the average number of active buckets is $\langle \nu \rangle \sim 18$, and is only weakly dependent on the value of n.

The gradual decreases in efficiency of the traditional scheme compared to the fitted n^2 curve can be correlated with increased memory latency as the size of the simulation data begins to exceed the on-chip caches. This is reasonable, as the computation of interaction rates for the linear kernel is arithmetically trivial (a single addition), which allows us to simplify the cost models of Sects. 4.2.8 and 4.3.7 using $C_\lambda \approx C_{\mathrm{op}}$. We thus expect the traditional scheme to run into a bandwidth limit. Conversely, the simulation data required for the bucketing scheme fits the per-socket inclusive L3 cache of our benchmark machine for all values of n examined. Results for the runs with $n = 1\,024$ and $n = 65\,536$ are reproduced in 4.6, showing increasingly accurate simulation of the desired outcome.

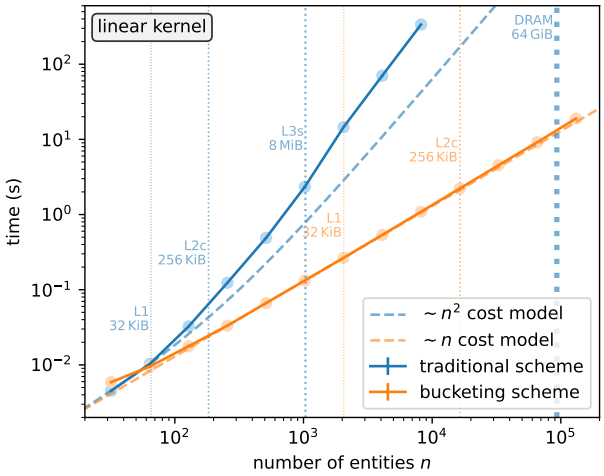


Figure 4.11: Performance analysis of the linear kernel test for the traditional scheme and the bucketing scheme.
The evolution of the linear kernel is simulated until $t = 16$. The dashed lines show fitted curves for simplified cost models linear and quadratic in n. The vertical dotted lines indicate the number of representative particles n for which the simulation data of either scheme exceeds the sizes of the L1 cache, the per-core L2 cache, the per-socket inclusive L3 cache, and the main memory of the benchmark machine.

4.6.2 Runaway kernel

Our second test investigates the runaway kernel (Eq. (4.69)). Unlike the linear kernel, which we studied in the many-particles regime because the analytical solution assumes continuous particle number densities, the runaway kernel is designed to test the runaway growth of an individual particle, and hence encompasses the transition from the many-particles regime to the few-particles regime. In the extended RPMC method, this transition occurs for a swarm k when the swarm particle count N_k falls below the particle regime threshold N_{th}. The swarm is then split into N_{th} individual representative particles each having a swarm particle count of 1, that is, representing only themselves, and the total number of representative particles n increases accordingly.

The runaway kernel test has scaling characteristics similar to the linear kernel test, as can be seen in Fig. 4.12, which shows the runtime for a single-threaded simulation of the runaway kernel until $t = 70$. Because the number of representative particles varies, runtimes are shown for the event-averaged number of representative particles $\langle n \rangle$, with the initial and maximal number of representative particles indicated separately. As for the linear kernel, super-quadratic inefficiencies of the traditional scheme can be correlated with the large working set size exceeding the on-chip caches; the interaction rate is almost as trivial to compute as for the linear kernel (assuming that the powers $(m_k)^{2/3}$, which are computationally expensive, have been precomputed and cached for each particle k), so we again expect to eventually be bandwidth-limited. The slightly super-linear runtime of the bucketing scheme for $\langle n \rangle \gtrsim 3 \cdot 10^5$ can be attributed to the event-averaged

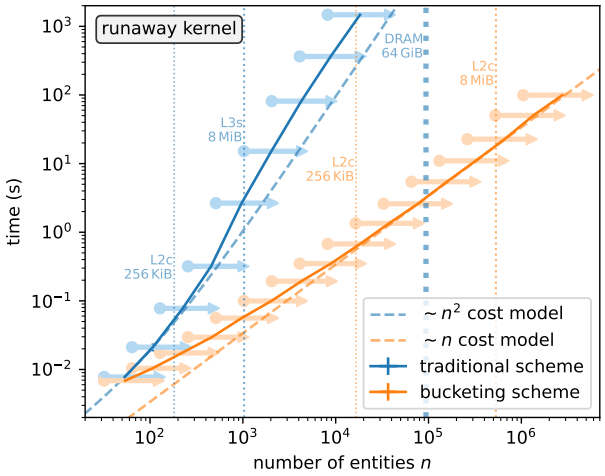


Figure 4.12: Performance analysis of the runaway kernel test for the traditional scheme and the bucketing scheme.
The evolution of the runaway kernel is simulated until $t = 70$. The dashed lines show fitted curves for simplified cost models linear and quadratic in n. The solid lines show the runtime for the event-averaged number of representative particles. For each tested configuration (solid horizontal lines), the initial number of representative particles (circle) and the maximum number of representative particles in the simulation (right arrow) are indicated. The vertical dotted lines indicate the number of representative particles n for which the simulation data of either scheme exceeds the sizes of the per-core L2 cache, the per-socket inclusive L3 cache, and the main memory of the benchmark machine.

number of occupied buckets $\langle \nu \rangle$ which grows slightly (from ~ 10 to ~ 20) as $\langle n \rangle$ increases. The number of occupied buckets increases because, with more representative particles available, more of the tail end of the distribution can be resolved, as is evident from Fig. 4.7.

4.6.3 Collision and velocity evolution

Although the bucketing scheme was demonstrated to perform substantially better than the traditional implementation for the linear and runaway kernel tests, we note that in these synthetic test cases, entities have only two properties: particle mass and swarm mass. As argued in Sect. 4.1, one of the main reasons to choose a Monte Carlo method over a grid-based method is that more properties can be added easily, whereas doing so in a grid-based method would increase the dimensionality of the grid, leading to soon-prohibitive costs in terms of both memory and computation resources. The computational scheme traditionally employed for the RPMC method is not sensitive to the dimensionality of the parameter space. This is not necessarily true for the bucketing scheme. If the particle–particle interaction rate is influenced by multiple properties – say, mass and rms velocity –, then the bucketing criterion should consider each of these

properties in separate dimensions, similar to what had been done in the refined three-dimensional bucketing criterion in Eq. (4.42). The bucketing scheme can be likened to a grid method in the subspace of the property space considered by the bucketing criterion; adding an additional property dimension to the bucketing criterion thus adds to the dimensionality of the grid. The curse of dimensionality is somewhat attenuated because the bucketing scheme uses a sparse grid of buckets, but even if the occurrence of different particle properties is perfectly correlated (e.g. a light particle always has a high rms velocity, and a heavy particle is always rather slow) the number of occupied buckets likely increases by some factor for every additional property considered, which can become significant if many additional properties are introduced. To assess the performance of the bucketing scheme with more than one particle property, we therefore turn to a more realistic example.

4.6.3.1 Ormel's model

Ormel et al. (2010) have presented a comprehensive stochastic model for the simulation of runaway growth of protoplanetary bodies. The model comprises collisional encounters (leading to coagulation or fragmentation), gravitational interactions (modelled as *viscous stirring* and *dynamical friction*, which are stochastic surrogate models for the exchange of kinetic energy between particles), and also gas drag forces and turbulent stirring. Collisional encounters were simulated with the Monte Carlo method of Ormel and Spaans (2008). Gravitational encounters, however, could not be simulated with a Monte Carlo method because of the excessive cost of computing interaction rate updates: '[...] it is too demanding to treat collisionless encounters also on an event-based approach. Due to their increased gravitational focussing [...] collisionless encounters are more frequent than collisional interactions by several orders of magnitude, especially when the system relaxes to a quasi-steady state in velocity space' (Ormel et al., 2010, §2.6).

To test the bucketing scheme in a more likely scenario, we use it to simulate the stochastic growth model introduced by Ormel et al. (2010), henceforth referred to as *Ormel's model*. However, unlike the original implementation, we model both collisional and gravitational encounters as events, which has become feasible thanks to the increased computational efficiency of the bucketing scheme. We simplify the model by neglecting the influence of the gas (effectively setting the gas density to $\rho_{\mathrm{gas}} = 0$). Without a gas, there is no turbulent stirring and no gas drag. Apart from this, our model is identical to the model presented in Ormel et al. (2010) save for minute technical details. An extension of Ormel's model implemented with the bucketing scheme is elaborated in Sect. 6.1; in the following we shall only briefly summarise the gist of the model before proposing a suitable bucketing criterion.

In our implementation of Ormel's model, an entity k is equipped with the following properties: *swarm mass* M_k, *particle mass* m_k, *orbital radius* r_k, *planar rms velocity* v_k, *vertical rms velocity* $v_{z,k}$, and *solid density* ρ_k. For a representative particle j and a swarm k, the interaction rates for collisions, dynamical friction events, and viscous stirring events are denoted $\lambda_{jk}^{\mathrm{col}}$, $\lambda_{jk}^{\mathrm{df}}$, and $\lambda_{jk}^{\mathrm{vs}}$. A swarm of particles k is confined to a radial annulus of $[r_k - h_{x,k}, r_k + h_{x,k}]$, where the scale length is given by $h_{x,k} = \max\{R_{\mathrm{min}}, v_k/\Omega(r_k)\}$ for many-particles swarms (that is, swarms k with $N_k > N_{\mathrm{th}}$) or by $h_{x,k} = v_k/\Omega(r_k)$ for few-particles swarms ($N_k \leqslant N_{\mathrm{th}}$). R_{min} is a simulation parame-

ter which bounds the radial resolution of the simulation, and $\Omega(r) = \sqrt{GM_*/r^3}$ is the Keplerian rotational frequency of a body on a circular orbit around a central object of mass M_* with G the gravitational constant. Because swarms are split up at the many-particles → few-particles transition, few-particles swarms contain only a single self-representing particle. Individual particles are not statistical representatives, and therefore imposing a minimal effective scale length is not necessary.

In Ormel's model, colliding particles can coagulate or fragment. In the very simple collision model parameterised by the *effective coefficient of restitution* ϵ, outcomes other than coagulation are possible only in the superescape regime, that is, if the approach velocity of the colliding particles exceeds their mutual escape velocity. It is assumed that, upon collision of particles j and k, a fraction f_{frag} of the combined mass $(m_j + m_k)$ ends up as fragments. For simplicity, a uniform fragment size of $R_{\mathrm{frag}} = 1\,\mathrm{cm}$ is assumed (slightly different from Ormel et al. (2010) who used $R_{\mathrm{frag}} = 1\,\mathrm{mm}$ and $R_{\mathrm{frag}} = 10\,\mathrm{cm}$). The mass fraction depends on the particle masses and the kinetic energy of the impact,

$$f_{\mathrm{frag}} = \frac{\epsilon E_{\mathrm{col}}}{\frac{1}{2}\,(m_j + m_k)\,v_{\mathrm{esc}}^2}\,, \tag{4.122}$$

where $v_{\mathrm{esc}} = \sqrt{4G(m_j + m_k)/(R_j + R_k)}$ is the mutual escape velocity, and where the kinetic energy of the impact is

$$E_{\mathrm{col}} = \frac{1}{2}\frac{m_j m_k}{m_j + m_k} v_{\mathrm{a}}^2 \tag{4.123}$$

with v_{a} the particle approach velocity. How exactly this fragmentation model can be incorporated into the extended RPMC method will be discussed in Sect. 4.6.3.1 of the subsequent chapter.

For the three types of interaction – collisions, viscous stirring, dynamical friction –, the *interaction radii* R_{jk}^{col}, R_{jk}^{df}, and R_{jk}^{vs} can be computed; two entities j, k can only interact if the particles they represent can get closer than the respective interaction radius. Interaction rate, interaction radius, and interaction outcome are also dependent on the *velocity regime*. Interactions of particles with low rms velocities operate in the *shear-dominated regime*, whereas a higher rms velocity places the interaction in the *dispersion-dominated regime*. A pair of particles with a stochastic relative velocity faster than their mutual escape velocity is in the *superescape regime* where viscous stirring and dynamical friction are irrelevant and where the effect of gravitational focussing is negligible.

Determining interaction rates or interaction radii for either type of interaction is complicated. Among other things, for a representative particle j and a swarm k the geometric overlap of the 'living zones' of the two swarms has to be computed, and the routines contain multiple `if–else` branches to distinguish between the different velocity regimes. In contrast to the synthetic linear and runaway kernel test cases of Sects. 4.6.1 and 4.6.2, the computation of interaction rates thus costs much more than a single arithmetic operation, $C_\lambda \gg C_{\mathrm{op}}$, which means that the simulation remains compute-bound for much larger working-set sizes than with the synthetic tests.

For the bucketing scheme, an interval extension of the interaction rate function is required. Due to the inherent complexity of the interaction rate function, constructing

a corresponding interval extension by hand would be very laborious and error-prone. Instead, we use the interval-aware programming paradigm set forth in Chapter 5 to implement the routines for computing interaction rates such that they can return either a real value for real-valued input arguments or an interval for interval-valued input arguments.

For all but the simplest operations, interval arithmetic is disproportionately expensive compared to real arithmetic. As an example, computing the sum of two intervals requires only two additions, but computing the product of two intervals requires at least four multiplications, three $\min$, and three $\max$ operations (cf. e.g. Hickey et al., 2001). As a result, the estimation of a bucket–bucket interaction rate upper bound is notably costlier than the computation of a particle–swarm interaction rate, $C_{\lambda^+} \gg C_\lambda$. As per the cost models in Eqs. (4.26) and (4.60), it is thus possible that the traditional scheme outperforms the bucketing scheme for relatively small numbers of representative particles, $n \sim \nu$.

To test the bucketing scheme with Ormel's model, we adopt the following test scenario, which is a gas-free version of the 1 AU test case in Ormel et al. (2010). We simulate a protoplanetary system with a central star of solar mass, $M_* = M_s$ with a narrow annulus at $a_0 = 1\,\mathrm{AU}$ of width $\Delta a = 0.04\,\mathrm{AU}$ that is uniformly populated by protoplanetary objects of initial mass $m_0 = 4.8 \cdot 10^{18}\,\mathrm{g}$ and solid density $\rho = 3\,\mathrm{g/cm^3}$. The mass surface density inside the ring is $\Sigma_0 = 16.7\,\mathrm{g/cm^2}$. The particles have an initial rms velocity of $v_0 = 4.7 \cdot 10^2\,\mathrm{cm/s}$. The gas density is assumed to be $\rho_{\mathrm{gas}} = 0$ for simplicity. To avoid undersampling, the radial resolution of the simulation is constrained by $R_{\min} = \Delta a/64$. The simulation is then run for a duration of $1.8 \cdot 10^5$ years of simulated time, modelling the interplay of particle coagulation and velocity evolution as a series of individual events. We test the model with two different values for the coefficient of restitution: $\epsilon = 0$, which implies that no fragmentation occurs and collisions always lead to coagulation; and $\epsilon = 0.01$, a value corresponding to a 'rubble pile' model of planetesimals (Ormel et al., 2010, §2.5.2).

The test particles are sampled with n randomly chosen representative particles, where n is varied during different simulation runs. Every representative particle k initially represents an equal-weight fraction of the total mass, $M_k = \mathcal{M}/n$, where the total mass is

$$\mathcal{M} = 2\pi r \Delta r \cdot \Sigma_0 \approx 0.16 M_\oplus \,. \tag{4.124}$$

In the bucketing criterion, all properties which quantitatively affect the interaction rate are considered separately:

$$\begin{aligned} B(\mathbf{q}) = \Bigg(& C(\mathbf{q}), \left\lfloor \theta_M \log_{10} \frac{M(\mathbf{q})}{M_0} \right\rfloor, \left\lfloor \theta_m \log_{10} \frac{m(\mathbf{q})}{m_0} \right\rfloor, \\ & \left\lfloor \theta_v \log_{10} \frac{v(\mathbf{q})}{v_0} \right\rfloor, \left\lfloor \theta_v \log_{10} \frac{v_z(\mathbf{q})}{v_0} \right\rfloor \Bigg) \,, \end{aligned} \tag{4.125}$$

where $C(\mathbf{q})$ is the swarm regime classifier defined in Eq. (4.43), particle mass, swarm mass, and planar and vertical rms velocities of a representative particle with properties $\mathbf{q}$ are given by $m(\mathbf{q})$, $M(\mathbf{q})$, $v(\mathbf{q})$, and $v_z(\mathbf{q})$, respectively, and where the simulation

parameters θ_m, θ_M, and θ_v control the number of buckets per decade of the respective quantity. Unless indicated otherwise, we adopt the values $\theta_m = \theta_M = 6, \theta_v = 5$. Ormel's model is employed to study runaway growth, which means that some representative particles transition into the few-particles regime where individual particles can absorb up the mass of other swarms. Following the discussion in Chapter 3, we adopt a particle regime threshold of $N_{\mathrm{th}} = 10$.

The described physical system is not homogeneous. From the range-limited stirring and coagulation processes, structures emerge on certain length scales, and therefore spatial resolution is required so these structures can be properly resolved. We therefore take advantage of the sub-bucketing extension of the bucketing scheme introduced in Sect. 4.5. In our simulations, we use a sub-bucket granularity of $d_{\mathrm{min}} = 8$. Because the interaction radii tend to scale with the orbital radius, we adopt the logarithmic definition of the interaction distance given in Appendix 4.B.

4.6.3.2 Boosting

As assessed by Ormel et al. (2010), the number of events that need to be simulated is vast because '[...] the MC code does not recognize that the collective effect of these encounters cancels out, but instead resolves the strongly fluctuating velocities of the bodies, which render the code very inefficient'. (Ormel et al., 2010, §2.6, cont'd) Although the MC simulation of the velocity evolution becomes feasible with the bucketing scheme, the general inefficiency of the model remains. However, the model can be made more efficient by applying the *boosting* technique, previously described for coagulation in Chapter 3, to stirring and friction events. With boosting, gravitational interactions which transport minuscule amounts of kinetic energy are grouped together, decreasing the rate while correspondingly increasing the kinetic energy transfer. Although this is an approximation, it does not impair the main benefit of simulating gravitational interactions with a Monte Carlo method, which is that sudden discrete changes of kinetic energy (a heavy particle 'kicking' a light one) can be modelled correctly. Unless noted otherwise, we apply the boosting technique to all our simulations of Ormel's model for collision events as well as for viscous stirring and dynamical friction, using a mass growth rate of 5% and an rms velocity change rate of 2%.

4.6.3.3 Results

A typical result obtained with the simplified Ormel's model showing the emergence of oligarchs is reproduced in Fig. 4.13 with three snapshots at $t = 1.3 \times 10^4$ yr, $t = 3.8 \times 10^4$ yr, and $t = 1.8 \times 10^5$ yr. Results for two runs with $\epsilon = 0$ (no fragmentation) and $\epsilon = 0.01$ (corresponding to a rubble pile model) are shown.

When comparing the results to Fig. 9 of Ormel et al. (2010), we note that the distribution structure of the colour-encoded dynamical temperature visibly differs; in particular, the correlation between particle size R and rms eccentricity seems absent in our result. This may be due to the negligence of the gas disk in our simplification of Ormel's model. A gas disk would orbit the central body at a lower orbital velocity and thus exert a gas drag force which effectively damps the rms eccentricities and rms inclinations (cf. Inaba et al., 2001) of massive bodies. The damping rate of a body is correlated with its

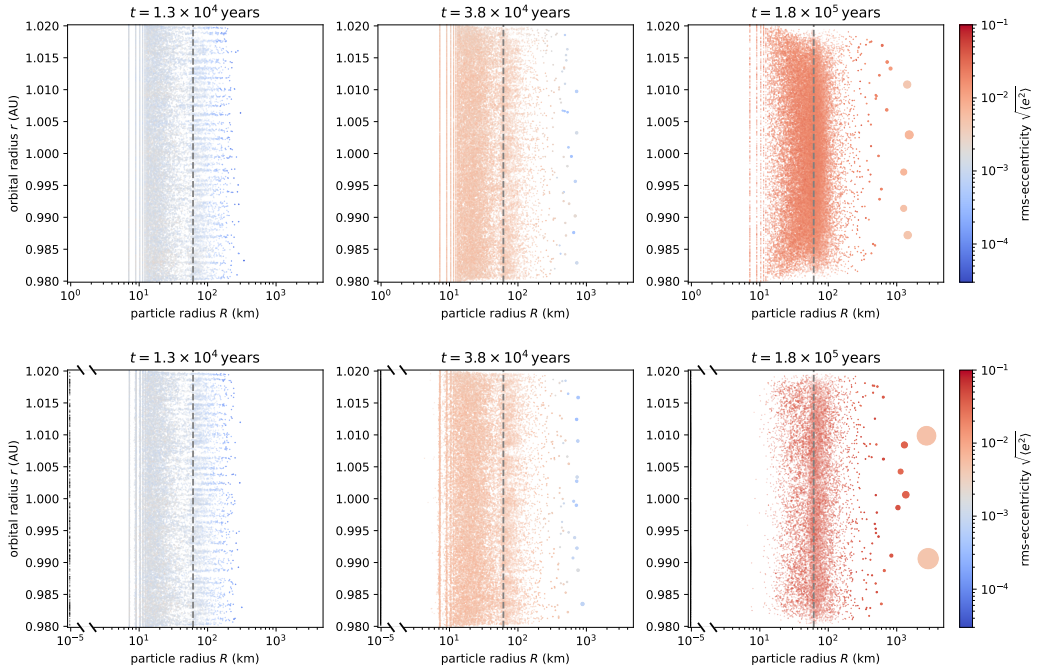


Figure 4.13: Results for the simplified Ormel's model starting with $n_0 = 32\,768$ representative particles.
Top row: $\epsilon = 0$ (no fragmentation); bottom row: $\epsilon = 0.01$ (rubble pile model). Every swarm is represented by a spherical shape with a colour indicating its planar rms eccentricity. The area covered by the shape represents the mass in the swarm. All swarms initially have the same mass, but individual particles can accrete mass held by other swarms and thereby grow beyond their initial swarm size. The vertical lines indicate the critical radius R_{crit} (Eq. (4.126)) which marks the approximate boundary between many-particles and few-particles swarms.

surface-to-volume ratio, and thus damping is more effective for smaller bodies, an effect that can be observed in Fig. 9 of Ormel et al. (2010) but not in our results. Another difference is the slight radial compaction visible in our results, which stems from the fact that, in our simulation, the boundaries in the radial dimension r are unconstrained, whereas Ormel et al. (2010) used periodic boundaries.

Including fragmentation has the obvious consequence that some mass is removed from the swarms of planetesimal-sized bodies. Fragments are assumed to be tightly coupled to the gas (the presence of which we conceptually assume even though we ignore its other effects here), and hence they neither self-accrete nor contribute to viscous stirring or dynamical friction. Instead, they are available as a mass reservoir for accreting planetesimals. This leads to a visible dissipation of masses in the lower end of the mass distribution. With fragmentation, the emerging oligarchs grow to slightly larger masses in the same amount of time, an observation consistent with Fig. 10 of Ormel et al. (2010) and in accordance with Wetherill and Stewart (1993), who found that including fragmen-

tation enables a self-regulation of the viscous stirring process, reducing the average rms velocities and thus increasing the effective collision cross-sections, which shortens the runaway growth timescale.

The dashed vertical lines in Fig. 4.13 indicate the spherical particle radius

$$R_{\mathrm{crit}} = \left(\frac{m_{\mathrm{crit}}}{\frac{4}{3}\pi\rho} \right)^{\frac{1}{3}} \tag{4.126}$$

corresponding to the critical mass m_{crit} defined as

$$m_{\mathrm{crit}} = \frac{\mathcal{M}}{n_0 N_{\mathrm{th}}} \tag{4.127}$$

(cf. Eq. (3.39)). This mass segregates many-particles and few-particles swarms under the simplifying assumption that there are n_0 swarms which all represent an equal fraction of the total mass, $M_k = \mathcal{M}/n_0 \, \forall k$. Representative particles with a mass greater than m_{crit} tend to be few-particles swarms, and because the RPMC method splits up swarms once their particle number falls below the particle number threshold N_{th}, such particles then are self-representing. The corresponding increase of resolution is visible in the vicinity of m_{crit}.

Although the geometric model employed by Ormel et al. (2010) for collisions, viscous stirring and dynamical friction works well as a statistical model for swarms comprising many particles, its assumptions break down for interactions between individual particles which have individual, not statistical, orbital properties. Specifically, features such as orbital resonances or radial redistribution cannot be reproduced with the geometric interaction model. Therefore, self-representing bodies exceeding a certain mass should be simulated with an N-body code. The inclusion of an N-body code exceeds the scope of this chapter, so we follow Ormel et al. (2010) in unconditionally applying the statistical interaction models for all swarms k regardless of their particle mass m_k or particle count N_k.

4.6.3.4 Performance

Although the runtime performance is no longer dominated by memory access times, the memory usage of the simulation poses a practical constraint for the possible scaling. In Fig. 4.14, the memory cost of simulating Ormel's model is explored for a variety of representative particle counts n and occupied bucket counts ν. It is evident that the traditional scheme soon exhausts the available memory when increasing resolution beyond $n = 10^3$ representative particles. In contrast, the memory cost of the bucketing scheme is much lower overall and scales with $n \cdot \nu^2$. To understand the kinks, we remember that buckets are allocated sparsely; because each entity is assigned to one and only one bucket, the number of entities n acts as an upper bound to the number of occupied buckets ν.

A performance analysis of Ormel's model comparing the traditional scheme and the bucketing scheme is reproduced in Fig. 4.15. The bucketing scheme again performs substantially better than the traditional scheme. Because of the relatively high bucket density chosen with $\theta_m = \theta_M = 6$, $\theta_v = 5$, the number of occupied buckets ν is large (saturating around $\nu \sim 350$), and thus the cost for bucket updating is relatively high even

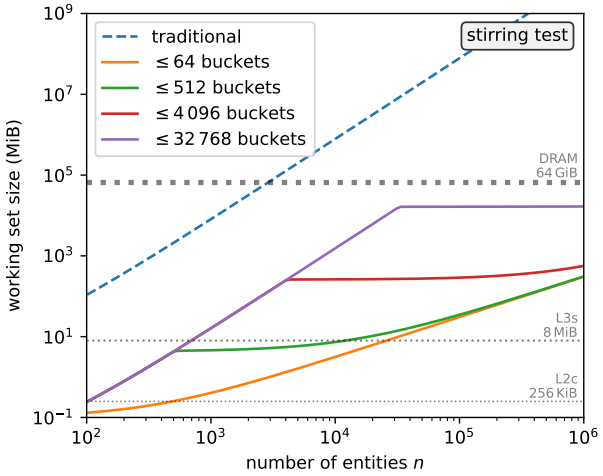


Figure 4.14: Memory cost analysis of simulating Ormel's model.
The dashed and solid lines show the memory required for simulating Ormel's model with the traditional implementation and the bucketing scheme, respectively. Some cache sizes and the main memory size of the benchmark machine are indicated with horizontal dotted lines.

for a small number of entities n. However, the computational cost grows only weakly as the number of entities n increases, even bettering the scaling behaviour predicted by the cost model of Sect. 4.3.7, as demonstrated by the ad hoc $\sim n^{2/3}$ cost model fit which seems to closely match the observed simulation performance.

Our analytical cost model does not explain the apparent sub-linear scaling of the bucketing scheme observed in Fig. 4.15. It is possible that the scaling may ultimately become linear in n, as predicted by the cost model, for even larger values of n which we did not test. We note, however, that we have not investigated the cost impact of the sub-bucketing extension of our scheme, which might cause the different scaling characteristics. An extended cost model would have to entail some analytical representation of the effects of constrained reachability, which we found difficult enough to eschew the effort.

The spatial resolution of the system is limited by the simulation parameter $R_{\min}$, which here was chosen as $R_{\min} = \Delta a/64$. Because we want to investigate the scaling behaviour of the bucketing scheme, this simulation parameter has remained invariant over all simulation parameter sets. Normally, this parameter would be chosen such as to avoid undersampling, which implies that it would be inversely proportional to the number of representative particles used, $R_{\min} \propto n^{-1}$. Choosing a larger $R_{\min}$ for smaller n would imply that swarms may be effectively spread out farther, allowing interactions between particles farther apart, and therefore allowing more interaction events and presumably slowing down the simulation. Conversely, choosing a smaller $R_{\min}$ for larger n, and thereby increasing the resolution of the simulation, would allow swarms to occupy narrower annuli, and thus decrease the number of interaction events because more particle pairings would be out of reach. Reducing $R_{\min}$ for larger n would thus presum-

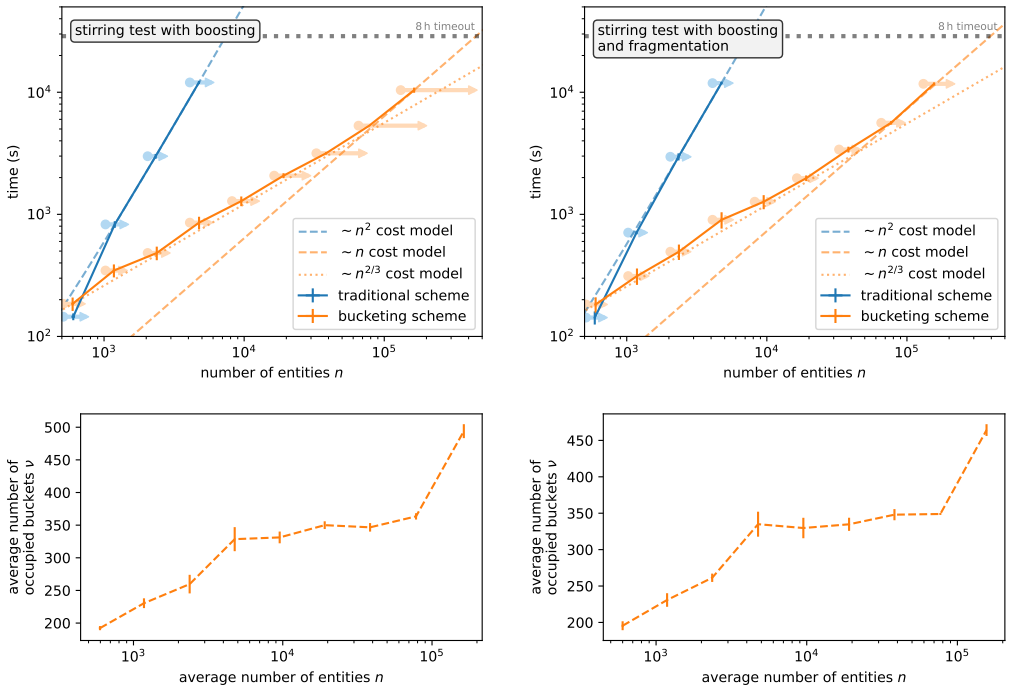


Figure 4.15: Performance analysis of Ormel's model for the traditional scheme and the bucketing scheme with boosted dynamical friction and viscous stirring.
Left column: $\epsilon = 0$ (no fragmentation); right column: $\epsilon = 0.01$ (rubble pile model). In the top panels, the dashed lines show fitted curves for simplified cost models linear and quadratic in n, and an additional ad hoc fit with a $n^{2/3}$ proportionality. The solid lines show the runtime for the event-averaged number of representative particles. For each tested configuration (solid horizontal lines), the initial number of representative particles (circle) and the maximum number of representative particles in the simulation (right arrow) are indicated. The bottom panels show the average number of occupied buckets for each of the simulation parameter sets.

ably speed up the simulation. We also note that, even without explicitly reducing $R_{\min}$, spatial resolution improves with increasing n because, as explained in Sect. 4.6.3.1, $R_{\min}$ is not imposed as a minimum effective scale length for particles which are represented individually; with more representative particles available to sample the system, the transition to individual particles occurs at lower masses, and therefore at an earlier time. This effect may also play into the sub-linear efficiency scaling observed.

The average number of occupied buckets shown in the bottom panels of Fig. 4.15 steadily increases at first, then seems to saturate for $n \gtrsim 4 \times 10^3$ (and begins to decrease slightly for larger n), indicating, though not conclusively proving, that $n \gtrsim 4 \times 10^3$ representative particles may be sufficient in covering the full dynamic range of the system, and, conversely, that the system may suffer from undersampling for $n \lesssim 4 \times 10^3$. After the average number of occupied buckets plateaus for $4 \times 10^3 \lesssim n \lesssim 10^5$, it again begins

to grow for $n \gtrsim 10^5$. This may indicate that the system is 'over-resolved': by increasing n_0, the critical mass m_{crit} (Eq. (4.127)) of regime transition is reduced, and swarms become few-particles swarms already at lower particle masses. By its design, the original RPMC method has a self-balancing property: all swarms retain the fraction of the total mass they were initialised with. Specifically, in an equal-weight sampling, every swarm k would be initialised with the same fraction $M_k = \mathcal{M}/n$ of the total mass $\mathcal{M}$, and therefore the ensemble of swarms would always remain an equal-weight sampling, which is beneficial because it maximises the sampling efficiency. In the extended RPMC method, the self-balancing property could be retained only for many-particles swarms. Therefore, by excessively increasing the resolution, and in turn by pushing the critical mass m_{crit} of regime transition far below the threshold mass for runaway growth, the statistical self-balancing property is gradually lost, leading to an increase in the average number of occupied buckets, which in turn slightly impairs the scaling characteristics.

As is evident from Fig. 4.15, the inclusion of a fragmentation model does not significantly affect the performance characteristics of the bucketing scheme. It was mentioned in Chapter 3 that the RPMC method, the same as any Lagrangian method, is 'ill-suited for simulating "stiff" problems in which two adverse effects nearly balance each other'. One could imagine a system with a steady-state mass distribution which nevertheless exhibits active growth and fragmentation; in such a system, a Lagrangian method has to follow the individual particles as they cycle through the growth and fragmentation process. This is, however, not the case in our fragmentation test: although mass is exchanged in both directions (planetesimals collide and form fragments, and fragments are accreted by planetesimals), the system does not have steady-state characteristics; instead, the system steadily evolves towards larger bodies (oligarchs) which eventually become dynamically isolated and thus are unlikely to fragment further.

Sect. 3.7.1 also elaborated that the 'stiffness' problem is aggravated by the proposed extension of the RPMC method because swarms that repeatedly follow the growth–fragmentation cycle may be repeatedly split up over the course of the simulation, leading to a steady growth in the number of representative particles. It was subsequently proposed that a merging step be incorporated to counteract the problem when simulating a system with steady-state characteristics. This advice remains appropriate with the bucketing scheme, although we note that the nature of the bucketing scheme helps reduce the cost induced by unnecessarily splitting up swarms since swarms with similar characteristics typically end up in the same bucket.

4.6.3.5 Impact of simulation parameters

The simulation parameters θ_m, θ_M, and θ_v, representing bucket densities in the respective dimensions of the space of bucket labels, need to be chosen large enough that the bucket–bucket interaction rate upper bounds are not excessive, which would lead to a very high probability of rejection and render the sampling process inefficient. At the same time, said parameters must be chosen as small as possible so as to minimise the total number of buckets occupied, and thereby minimise the cost of updating buckets and recomputing bucket–bucket interaction rates.

Fig. 4.16 demonstrates how these contradicting requirements lead to a practical trade-off. In the first panel, we see that, by increasing bucket densities, resulting in narrower

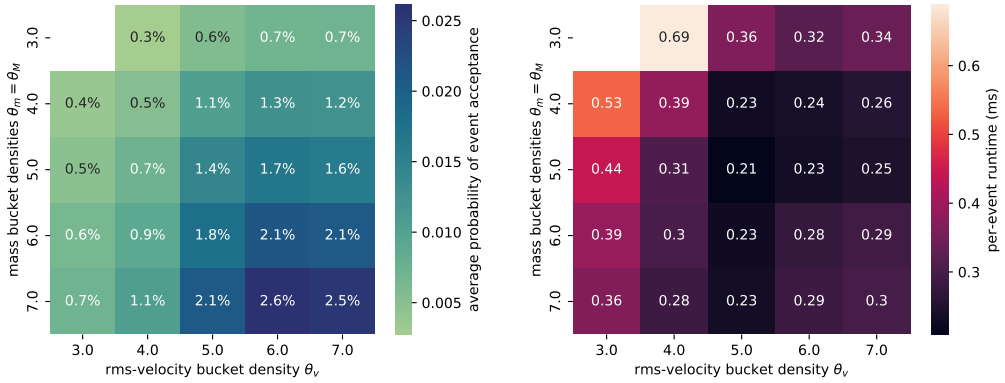


Figure 4.16: Parameter study of the bucket densities θ_v, θ_m, θ_M in Ormel's model.
The two panels show the average probability of event acceptance p and the per-event runtime cost, respectively, as functions of the rms velocity bucket density θ_v and the mass bucket densities $\theta_m = \theta_M$. The simulation uses the setup described in Sect. 4.6.3.1 with boosting, as in Fig. 4.15, for an initial number of $n_0 = 4\,096$ representative particles.

bucket parameter intervals, we can improve the quality of the bucket–bucket interaction rate bounds. With better bounds, the likelihood of rejection is lower, and the average probability of event acceptance p is higher. If fewer events need to be sampled only to be ultimately rejected, the simulation should run faster. In the second panel we can see, however, that the increased number of buckets which results from choosing a higher bucket density increases the cost of bucket updates, which at some point exceeds the gains from a higher acceptance probability. For the given set of parameters, a good choice for bucket density might thus be $\theta_v = \theta_m = \theta_M = 5$.

If similar events of low impact are aggregated, as done by the boosting technique mentioned in Sect. 4.6.3.2, the benefit of bucket widening is negligible. The simulation results in Figs. 4.15 and 4.16 have been run with boosting enabled. In these simulations, widening has proved unnecessary, and hence a widening fraction of $f = 0$ has been adopted. However, if the simulation is run without boosting, viscous stirring and dynamical friction events abound, rendering the simulation substantially slower. Widening can be useful in this situation, as demonstrated by Fig. 4.17. The bucketing scheme is very efficient in this case, yielding average acceptance probabilities of $p \sim 40\%$; however, because of the exorbitant number of events, a single simulation run takes between one and two hours to complete even though the width of the annulus has been reduced and the number of representative particles is relatively low. (A simulation employing the traditional scheme did not complete within the $8\,\mathrm{h}$ time limit of our benchmark machine.) Simulating viscous stirring and dynamical friction events as Monte Carlo events without boosting, albeit possible with the bucketing scheme, is thus not very practical.

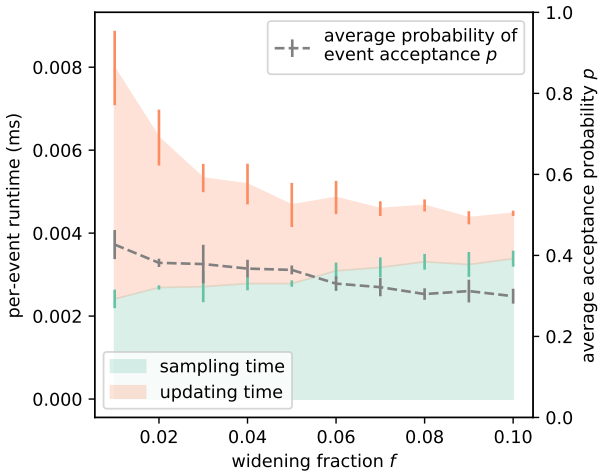


Figure 4.17: Parameter study of the widening fraction f in Ormel's model without boosting.
For this test, the width of the annulus was reduced to $\Delta a = 0.0025\,\mathrm{AU}$, fragmentation was suppressed ($\epsilon = 0$), and the system was sampled with an initial number of $n_0 = 128$ representative particles.

4.7 Discussion

For the synthetic test cases of Sects. 4.6.1 and 4.6.2, the proposed scheme shows a clear performance advantage over the traditional scheme. However, in the case of a more realistic test case simulating the interplay of multiple effects such as hit-and-stick coagulation and gravitational stirring, as studied with Ormel's model in Sect. 4.6.3, we note that the high dimensionality of the parameter space poses a challenge for the bucketing scheme. For competitive performance, we had to adapt the bucketing criterion of Eq. (4.125) which gave rise to a five-dimensional space of bucket labels and use relatively high bucket densities $\theta_m, \theta_M, \theta_v$, resulting in a relatively large number of ~ 450 occupied buckets on average. Even with a high-dimensional bucketing criterion and a high bucket density, the bucketing scheme has substantially outperformed the traditional scheme for larger entity counts n, showing superior and even sub-linear scaling characteristics for the range of parameters tested.

4.7.1 Limitations

Although we advertise the bucketing scheme as a general-purpose computational scheme for Monte Carlo simulations of compound Poisson processes, there are some practical limitations to its usefulness. The most obvious limitation is the conceptual complexity of the scheme, which makes it challenging to implement. We hope to alleviate this concern somewhat by subsequently publishing our implementation, written in C++20 and designed with reusability in mind, as an open-source software package. Nevertheless, a user of the package needs to understand how the scheme works in order to use it effectively.

Another requirement that may be challenging to satisfy is the need for an interaction rate function which can either operate on real values or on intervals. Although software packages for interval arithmetic are available (e.g. Brönnimann et al., 2006; Goualard, 2015) and the subsequent Chapter 5 sets forth a notion of interval-aware programming, devising an efficient interval extension of an arbitrary calculation is an unsolved problem and requires thorough mathematical study of the calculation and meticulous avoidance or containment of the so-called dependency problem that is inherent to interval arithmetic (e.g. Gustafson, 2017a, §16.2f).

4.7.2 Possible improvements

The lower memory requirements of the bucketing scheme allow to push the boundaries of our simulation much further than would be possible with the traditional scheme, as outlined in Fig. 4.14. In addition, the bucketing scheme outperforms the traditional scheme in every scenario tested, often scaling with n as opposed to the n^2 scaling of the traditional scheme. Nevertheless, our tests also expose some limitations of the bucketing scheme. An obvious limitation is the sequential nature of the algorithm. Also, as we saw in Sect. 4.6.3.5, selecting an efficient set of simulation parameters is not easy and may require a parameter study.

It had been argued in Sect. 4.3.7 that the cost rate of the bucketing scheme scales at least quadratically in the number of occupied buckets ν; therefore, if the bucket density is chosen such that ν is large, the cost of the simulation can be very high even for moderate entity counts n, and higher than the cost of the traditional scheme due to the greater cost and the excess of interval-valued calculations. This can be observed at the lower end of Fig. 4.15: in a run with an initial number of $n_0 = 512$ representative particles, the bucketing scheme even performs slightly worse than the traditional scheme. We note that this run is not meaningful physically because the system is severely undersampled with only $n_0 = 512$ representative particles; still, we would want to better the traditional scheme with a significant margin even for smaller entity counts n. We saw in Sect. 4.6.3.5 that the bucket density must be chosen as a trade-off between a lower number of occupied buckets ν and low rejection rates due to tighter estimates for the bucket–bucket interaction rate. This trade-off plays out differently for different entity counts n, and indeed the simulation with $n_0 = 512$ could have been sped up by selecting lower bucket densities. Our observations raise the question of whether and how the scheme could be improved towards better performance without having to conduct parameter studies and tuning of simulation parameters.

4.7.2.1 Parallelisation

Our reference implementation of both the traditional scheme and the bucketing scheme is a sequential code. It runs in a single thread and does not take advantage of modern multi-core processors or massively parallel architectures such as GPUs. Although the proposed design of the scheme is of inherently sequential nature due to the global ordering of events, the two main simulation steps – sampling an event and recomputing bucket–bucket interaction rate upper bounds – could be parallelised straightforwardly. In the vernacular of parallel computing, the updating step would be considered 'embarrassingly parallel'. In the sampling step, the overhead caused by rejection sampling could

be mitigated by sampling P events concurrently, where $P \in \mathbb{N}_+$ is the degree of parallelism desired, and discarding all but the first of the sampled events that were not rejected.

In parallel computers, synchronising state globally is a very expensive operation. When pushing the simulation to larger scales, the global ordering of the events will eventually become a bottleneck. A massively parallel computational scheme would presumably want to employ a 'divide and conquer' approach, assigning a fraction of the total particle ensemble to each worker. The forgiving nature of the bucketing scheme – its tolerance of minute changes of entity properties within the bucket property bounds – would prove beneficial in the inevitable merging of concurrently simulated events.

4.7.2.2 Adaptive bucketing

In all bucketing criteria proposed so far, the number of buckets per decade (or the linear equivalent) had to be chosen explicitly as a simulation parameter. Choosing appropriate simulation parameters requires some knowledge of the simulated model, and often some parameter study as well. But with a set of simulation parameters that assures reasonably efficient rejection sampling, the total number of buckets required can be too large, rendering bucket updates very costly.

But the number of buckets per decade need not be fixed. Different parameter values could be used over the course of the simulation, or even for different decades. In fact, the way the bucketing scheme is constructed, the bucketing criterion $B(\mathbf{q})$ can return an arbitrary result without impairing the correctness of the simulation; in particular it may return different results when called at different times. Specifically, $B(\mathbf{q})$ could also refer to some global state $\mathbf{g}$ (such as the current simulated time) or to simulation state $\mathbf{s}$ (such as the current number of occupied buckets) and should thus be written $B(\mathbf{q}, \mathbf{g}, \mathbf{s})$. Using this freedom, a bucketing criterion could incorporate a *feedback loop*. For example, for every bucket J currently occupied we could maintain a running average of the number of rejections required to sample an event for a representative particle in this bucket. If the required number of rejections, weighted by the relative interaction rate weight of the bucket

$$\frac{n_J \lambda_J^+}{\left\langle n_J \lambda_J^+ \right\rangle} = \nu \, n_J \frac{\lambda_J^+}{\lambda^+} \tag{4.128}$$

is low (a good reference value might be $2n_J$, the number of updates that would need to be computed had the traditional scheme been used inside the bucket instead of rejection sampling), the bucketing criterion may 'zoom out', locally coarsening the bucketing criterion and thereby allowing the bucket to be merged with adjacent buckets which also have been zoomed out of. Conversely, if the required number of rejections is very high in a zoomed-out bucket, then the bucketing criterion would 'zoom in', thus redistributing all entities in the bucket to finer-grained buckets. Simulation parameters such as θ_m, θ_M, θ_v would be retained, but they would only set the highest possible resolution of the bucketing criterion.

4.8 Summary and conclusions

We have presented a novel computational scheme for the simulation of a stochastic process. The scheme combines a variant of the traditional sampling scheme with rejection sampling by grouping entities in a sparse grid of buckets, applying the discrete inverse transform sampling scheme to the buckets, and then selecting entities by means of rejection sampling. The scheme is made generally applicable by employing interval arithmetic to compute interval-valued interaction rates. It was shown to outperform the traditional scheme in terms of computational complexity and memory requirement as well as in practical performance and scalability for a variety of numerical tests.

In Sect. 4.2, we recounted how a stochastic process can be simulated with the traditional scheme that can be traced back to Gillespie (1975). A cost model for memory and computational resources was derived in Sect. 4.2.8. The bucketing scheme was then introduced in Sect. 4.3. The simulation algorithm and the associated data structures were defined in Sects. 4.3.5 and 4.3.6, and a cost model for the bucketing scheme was derived in Sect. 4.3.7. Sect. 4.4 discussed the matter of computing interaction rate bounds with interval arithmetic, along with strategies to increase the efficiency of the updating step. In order to exploit the locally constrained nature of most physical interaction models, we then introduced the notion of sub-bucketing in Sect. 4.5. In Sect. 4.6, we test the bucketing scheme for two simple synthetic test cases and one more realistic elaborate model based on Ormel et al. (2010), finding superior performance and scaling characteristics with regard to the number of representative particles n in all cases. Some parameter studies regarding the simulation parameters were then conducted in Sect. 4.6.3.5.

In Chapter 3, we had extended the RPMC method to allow for smooth transition into the runaway growth regime where bodies need to be treated individually. The transition happens abruptly: a swarm of N_{th} particles is replaced with N_{th} individual self-representing particles. Mitigating the performance impact of adding new representative particles to a RPMC simulation, which the original RPMC method of Zsom and Dullemond (2008) had been designed to avoid, was the original goal of devising the bucketing scheme. With the proposed scheme, representative particles can now be added and removed at amortised constant computational cost. According to our cost model, the simulation cost rate scales with $n \cdot \nu$ with the number of buckets ν at most weakly dependent on n, which is substantially better than the traditional scheme for which the cost rate is proportional to n^2. The predicted scaling behaviour has been found to be accurate in simple homogeneous test cases. In our study of a complex test case with spatial resolution, we found the cost rate of the bucketing scheme to scale even sub-linearly with n.

Appendices

4.A Interval arithmetic

It is commonplace in mathematical notation to apply scalar functions to sets, as in

$$\sin \mathbb{R} = [-1, 1] \,. \tag{4.129}$$

Such notation makes implicit use of the intuitive *set extension*, which could be defined for some scalar function $f : \mathcal{D} \to \mathbb{R}$ and for some subdomain $\mathcal{S} \subseteq \mathcal{D}$ as

$$\begin{aligned} &F : \mathcal{P}(\mathcal{D}) \to \mathcal{P}(\mathbb{R}) \,, \\ &F(\mathcal{S}) \coloneqq \{f(x) \,|\, x \in \mathcal{S}\} \,. \end{aligned} \tag{4.130}$$

Although convenient in formal reasoning, computation with general sets is not feasible numerically because there is no obvious closed-form representation for arbitrary sets. However, we can construct a similar technique that is practically computable by constraining ourselves to *closed intervals*, that is, closed connected subsets of $\mathbb{R}$. Intervals can be represented numerically as an ordered tuple of two numbers a and b with $a \leqslant b$, which constitute the lower and upper bound of the interval. The corresponding closed interval is usually referred to as $[a, b]$.

Let us first introduce some notation. We denote the set of all intervals in some closed connected subset $\mathcal{S} \subseteq \bar{\mathbb{R}}$ of the affinely extended real numbers $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ as

$$[\mathcal{S}] \coloneqq \big\{[a, b] : a, b \in \mathcal{S}, a \leqslant b\big\} \,. \tag{4.131}$$

We shall henceforth denote interval-valued quantities as $[x]$, identifying the bounds of a closed interval $[x]$ as

$$[x] \equiv [x^-, x^+] \,, \tag{4.132}$$

and also use an analogous shorthand notation for vector-valued quantities $\mathbf{v} \in \mathbb{R}^u$,

$$[\mathbf{v}] \equiv \left[\mathbf{v}^-, \mathbf{v}^+\right] \,, \tag{4.133}$$

where an interval of vector-valued quantities $[\mathbf{v}^-, \mathbf{v}^+]$ is understood as the Cartesian product of element-wise intervals,

$$\left[\mathbf{v}^-, \mathbf{v}^+\right] \coloneqq \left[v_1^-, v_1^+\right] \times \cdots \times \left[v_u^-, v_u^+\right] \,, \tag{4.134}$$

and where the u-dimensional space of vectors of intervals is denoted as $[\bar{\mathbb{R}}]^u$.

For some real-valued function $f : \bar{\mathcal{S}} \to \bar{\mathbb{R}}$ defined on a subset $\mathcal{S} \subseteq \bar{\mathbb{R}}$, any function $F : [\mathcal{S}] \to [\bar{\mathbb{R}}]$ shall be called an *interval extension* of f if

$$f(x) \in F([x]) \;\forall x \in [x] \;\forall\, [x] \in [\mathcal{S}] \,, \tag{4.135}$$

that is, F is an interval extension of a function f if, for every interval $[x]$, the interval $F([x])$ contains the value $f(x)$ for every value x in the interval $[x]$.

Closed-form interval extensions can be defined for all elementary arithmetic operations. For example, for addition and negation:

$$[x] + [y] \coloneqq [x^- + y^-, x^+ + y^+] \,, \tag{4.136}$$

$$-[x] \coloneqq [-x^+, -x^-] \,. \tag{4.137}$$

For any monotonically increasing function $g : \mathcal{S} \to \bar{\mathbb{R}}$, an exact interval extension is given by

$$G([x]) = \left[g(x^-), g(x^+)\right] \,. \tag{4.138}$$

A theorem often referred to as the 'Fundamental Theorem of Interval Arithmetic' states that, given a closed-form real-valued arithmetic expression composed of real-valued arguments, an interval extension can be obtained by syntactically replacing every real-valued function and operator with its interval extension (e.g. Hickey et al., 2001). In practice, this means that interval extensions are *composable*: if some real-valued function f has an algebraic closed-form definition for whose building blocks (such as elementary arithmetic operations) interval extensions are known, then an interval extension of f can be constructed with a syntactic transformation.

4.B Logarithmic location and distance

Let an interaction radius function $R(\mathbf{q}, \mathbf{q}')$ be given as defined in Sect. 4.5. Further, let the position of entity j be given by $a_j \equiv a(\mathbf{q}_j)$. Let us also assume that interaction radii approximately scale with the position. As an example, for representative particles j, k in a protoplanetary disk where the rms velocities are low (the *Hill regime*), the interaction radius for collisions might be determined by the dimensionless *Hill radius* $r_\mathrm{h} = \sqrt{m/(3M_*)}$,

$$R_{\mathrm{coll},jk} = 2.5\, a_{jk} \max\left\{r_{\mathrm{h},j}, r_{\mathrm{h},k}\right\} \,, \tag{4.139}$$

where $a_{jk} = \sqrt{a_j a_k}$ is the geometric average of the semimajor axes of representative particles j and k. With a non-local bucketing scheme such as Eq. (4.125), two representative particles j, j' of equal mass $m_j = m_{j'}$ but located at different semimajor axes $a'_j > a_j$ end up in the same bucket J. For a lighter particle k, $m_k < m_j$, the interaction radii would therefore differ, $R_{\mathrm{coll},j'k} > R_{\mathrm{coll},jk}$. For a bucket K of lighter particles, $m_K^+ < m_J^+$, the bucket–bucket interaction radius bound $R^+_{\mathrm{coll},JK}$ would then be excessive for particle j, and may thus lead to tentative collision events between particles too far apart which will ultimately be rejected.

To avoid this, we define locations and interaction distances as scale-free quantities. First, we define the scale-free *interaction width*

$$\omega_{\mathrm{coll},jk} \coloneqq \frac{R_{\mathrm{coll},jk}}{a_{jk}} = \max\left\{r_{\mathrm{h},j}, r_{\mathrm{h},k}\right\} \,. \tag{4.140}$$

A scale-free definition of location can be given as

$$l(\mathbf{q}) \coloneqq \gamma \log \frac{a(\mathbf{q})}{a_0} \tag{4.141}$$

with a normalisation factor γ to be defined later.

To define the distance measure $d(\mathbf{q}, \mathbf{q}')$, we first recall that two particles with properties $\mathbf{q}$ and $\mathbf{q}'$ can interact only if their impact parameter, estimated as $b \geqslant |a - a'|$ as per Eq. (4.94), is not greater than their interaction radius,

$$|a - a'| \leqslant R(\mathbf{q}, \mathbf{q}') \ . \tag{4.142}$$

The *interaction range* of some particle with properties $\mathbf{q}$ thus is the interval of semimajor axes $[a']$ of particles with properties $\mathbf{q}'$ with which interaction may be possible.

Now let us assume that, rather than the interaction radius $R(\mathbf{q}, \mathbf{q}')$, the interaction width $\omega(\mathbf{q}, \mathbf{q}')$ is known, where

$$\omega(\mathbf{q}, \mathbf{q}') \stackrel{!}{=} \frac{R(\mathbf{q}, \mathbf{q}')}{\sqrt{a\, a'}} \ . \tag{4.143}$$

Inserting Eq. (4.143) into Eq. (4.142), we thus find the interaction range for some particle at semimajor axis a and interaction width ω to be defined implicitly by

$$|a - a'| \leqslant \sqrt{a\, a'}\omega \ . \tag{4.144}$$

Solving Eq. (4.144) for a' yields

$$a' \in a \left[1 + \frac{1}{2}\omega^2 \pm \omega\sqrt{\frac{\omega^2}{4} + 1}\right] \ , \tag{4.145}$$

where we again use sloppy interval notation, $[x \pm y] = [x - |y|, x + |y|]$.

The interaction distance $d(\mathbf{q}, \mathbf{q}')$ is defined by the Euclidean relation of Eq. (4.105). We insert Eq. (4.141) and take the values of a' farthest from a at which interaction could possibly occur as per Eq. (4.145), finding

$$\begin{aligned} d(\mathbf{q}, \mathbf{q}') &= \gamma \log \left|1 + \frac{1}{2}\omega^2 \pm \omega\sqrt{\frac{\omega^2}{4} + 1}\right| \\ &= \gamma \log \left(1 + \frac{1}{2}\omega^2 + \omega\sqrt{\frac{\omega^2}{4} + 1}\right) \ , \end{aligned} \tag{4.146}$$

where we abbreviated $\omega \equiv \omega(\mathbf{q}, \mathbf{q}')$. The normalisation factor γ is then defined by relating the *minimum interaction width*

$$\omega_{\min} \coloneqq \frac{R_{\min}}{a_0} \ , \tag{4.147}$$

which represents the smallest length resolved by the simulation at the reference radius, to an interaction distance of $d_{\min}$:

$$\gamma \equiv \frac{d_{\min}}{1 + \frac{1}{2}\omega_{\min}^2 + \omega_{\min}\sqrt{\frac{\omega_{\min}^2}{4} + 1}} \ . \tag{4.148}$$

4.C Benchmark computer specifications

The performance benchmarks shown in Figs. 4.11, 4.12 and 4.15 were run as exclusive single-threaded jobs on machines with one CPU of type Intel Xeon E3-1585 v5 (Skylake-H, 4 cores, 3.5–3.9 GHz) and with 64 GiB of random-access memory.

The parameter studies shown in Figs. 4.16 and 4.17 were run as concurrent single-threaded jobs on machines with one CPU of type AMD Epyc 7662 (Zen 2, 64 cores, 2–3.3 GHz) and with 256 GiB of random-access memory.

4.D List of symbols

A cross-referenced overview of the most commonly used symbols in this chapter is given in Table 4.1.

Symbol	Description	Reference
λ_{jk}	interaction rate between entities j and k	Eq. (4.2)
$\lambda_{\text{ent}}(\mathbf{q}, \mathbf{q}', \delta)$	entity interaction rate function	Eq. (4.2); Eq. (4.16) (RPMC method)
λ_j	cumulative interaction rate of entity j	Eq. (4.5)
λ	global rate of interactions	Eq. (4.6)
λ^+_{JK}	upper bound for interaction rate between buckets J and K	Eqs. (4.44–4.45)
λ^+_J	upper bound of cumulative interaction rate for bucket J	Eq. (4.46)
λ^+	upper bound of global cumulative interaction rate	Eq. (4.47)
λ^+_{loc}	locality-aware upper bound of global cumulative interaction rate	Eq. (4.100)
$\nu = \#\mathcal{B}$	number of occupied (non-empty) buckets	Eq. (4.40)
ν_J	number of sub-buckets in bucket J	Sect. 4.5.4
θ_x	bucket density with regard to entity property x	Eq. (4.35); Eqs. (4.41) and (4.42) (test kernels); Eq. (4.125) (Ormel's model)
B_j	label of bucket currently associated with entity j	Eq. (4.35)
$\mathcal{B}$	set of labels of occupied (non-empty) buckets	Eq. (4.39)
$B(\mathbf{q})$	bucketing criterion	Eq. (4.35); Eqs. (4.41) and (4.42) (test kernels); Eq. (4.125) (Ormel's model)
$C(\mathbf{q})$	regime classifier (RPMC method)	Eq. (4.43)
$d_{jk} = d(\mathbf{q}_j, \mathbf{q}_k)$	interaction distance between entities j and k	Eqs. (4.103), (4.146)
$\mathcal{I} = \{1, \ldots, n\}$	set of all entity indices	Eq. (4.7)

$\mathcal{I}_J$	set of entity indices in bucket J	Eq. (4.36)
$l_j = l(\mathbf{q}_j)$	location of entity j	Sect. 4.5.2; Eqs. (4.102), (4.141)
$\mathcal{M}$	total mass (RPMC method)	Sect. 4.4
M_k	total mass of swarm k (RPMC method)	Sect. 4.4
m_k	mass of representative particle k (RPMC method)	Sect. 4.4
n	number of entities	Sect. 4.2.2
$n_J = \#\mathcal{I}_J$	number of entities in bucket J	Eq. (4.37)
n^+_{JK}	maximal number of entity pairs in buckets J, K which could possibly interact	Eqs. (4.100), (4.117)
$N^{\text{eff}}(\mathbf{q}, \mathbf{q}', \delta)$	effective swarm particle count (RPMC method)	Eq. (4.15)
N_{th}	particle regime threshold	Sect. 4.1
N_k	number of particles in swarm k (RPMC method)	Sect. 4.4
$P_{\text{accept}} = \lambda/\lambda^+$	probability of acceptance of sampled event	Eq. (4.32)
$P_{jk} = \lambda_{jk}/\lambda^+$	probability of interaction between entities j and k occurring at time $t + \Delta t$	Eq. (4.34)
$P_{\text{accept},jk} = \lambda_{jk}/\lambda^+_{JK}$	probability of acceptance of event sampled for entities j, k from buckets J, K	Eq. (4.51)
$\mathbf{q}_j \in \mathbb{Q}$	vector of properties of entity j	Sect. 4.2
Q_J	vector of property bound intervals of bucket J	Sect. 4.3.4; Eq. (4.84)
$R_{jk} = R(\mathbf{q}_j, \mathbf{q}_k)$	interaction radius between entities j and k	Eq. (4.97)
R_{min}	smallest resolved length scale	Sect. 4.5
$s_j = s_J(l_j)$	sub-bucket index of entity j in bucket J	Eq. (4.109)
t	current time	Sect. 4.3.1
Δt	event interarrival time	Eq. (4.4)
T	timescale on which changes inflicted by external operators must be considered	Sect. 4.3.1
$w^{\pm}_x(x')$	widening functions	Sect. 4.4.5
w_J	sub-bucket width in bucket J	Eq. (4.107)
$[x]$, $[\mathbf{v}]$, $[\mathcal{S}]$	abbreviated interval notation	Appendix 4.A

Table 4.1: List of commonly used symbols.

A paradigm for interval-aware programming 5

References: An excerpt of this chapter has been published in Beutel and Strzodka (2023). This chapter adds the following contributions: the illustration in Fig. 5.1; the formal treatment of interval-aware branching in Sect. 5.3.3; the inference of constraints in Sect. 5.4.1; extension to discrete-valued intervals in Sect. 5.5 and Sects. 5.6.1–5.6.2.

Interval arithmetic is a well-known method for obtaining exact bounds on computational results even with inexact input data and numerical error introduced by finite-precision numerics. The *posit* format, which aims to surpass the precision efficiency of the conventional IEEE 754 floating-point format, is accompanied by *valids*, an adaption and generalisation of interval arithmetic. A calculation can be performed either with posits or with valids, yielding either an approximate result with high computational efficiency or rigorous lower and upper bounds on the result. However, Boolean relational predicates such as $a < b$ are ambiguous when applied to overlapping intervals, leading to logical inconsistency no matter how the ambiguity is resolved. A numerical routine which has data-dependent branches can thus return incorrect results when applied to intervals or valids.

In this chapter we propose to define relational predicates for interval types as set-valued predicates instead of Boolean predicates. The proposed relational predicates are logically consistent and have intuitive projections to the two-element Boolean algebra. Using these predicates, we can express a calculation with data-dependent branches such that it can operate either on numbers or on intervals, while easily constraining interval-valued comparands by the branch condition. With such *interval-aware* code we can obtain either an approximate result or good interval bounds. We have developed a C++ library which implements the proposed concepts for traditional interval arithmetic. Furthermore, we have adapted it to a posit and valid implementation, demonstrating the viability of the concept with both traditional and more recent interval formats.

5.1 Introduction

Numerical calculations often involve and result in quantities whose exact value is not known. This can have a variety of reasons: some numbers cannot be represented exactly by the finite-precision number formats used for a calculation; some numerical methods yield only approximate results; and sometimes the input values are not known exactly.

Uncertainties of input values are often propagated with semianalytical methods such as Gaussian error propagation. Conceptually, for every uncertain scalar value, its standard deviation is tracked along with the value, and rules for propagating the standard deviation through a mathematical calculation can be stated with methods of calculus (Ku, 1966). Using these rules, a numerical routine could be adapted such that it propagates Gaussian uncertainties along with the quantities it operates on, yielding a result along with an uncertainty as the result of the calculation. This transformation can even be mechanised by employing Automatic Differentiation (e.g. Christianson and Cox, 2006; Le, 2016). However, the simplicity of the method and its automatic applicability can be deceiving. For the method to be appropriate, certain mathematical requirements must be met concerning, for instance, the differentiability of the calculation and the statistical properties of input data, such as underlying distribution and correlation between quantities. Even when the error propagation method is applicable, a propagated uncertainty has only statistical meaning; even if exact lower and upper bounds of input quantities were known, only confidence limits but no precise bounds could be given for derived quantities.

The desire to propagate exact lower and upper bounds through a calculation led to the development of *interval arithmetic* (Moore, 1966; Moore et al., 2009; Alefeld and Herzberger, 2012). The underlying idea is that, for a routine which operates on real numbers and yields a real number as a result, an *interval extension* of the routine can be constructed which operates on lower and upper bounds of real numbers and yields a lower and upper bound of the result. For example, let us consider the real-valued function

$$f(a, b) := a + b\,. \tag{5.1}$$

Given two values $\tilde{a}$ and $\tilde{b}$ which are not known precisely but can be constrained with known lower and upper bounds, $\tilde{A}^- \leqslant \tilde{a} \leqslant \tilde{A}^+$ and $\tilde{B}^- \leqslant \tilde{b} \leqslant \tilde{B}^+$, we would like to infer lower and upper bounds on $f(\tilde{a}, \tilde{b})$. To this end, we construct an interval extension of f,

$$F(A, B) := A + B\,. \tag{5.2}$$

where $A \equiv [A^-, A^+]$, $B \equiv [B^-, B^+]$ denote real-valued intervals, and where interval-valued addition is defined as

$$A + B = [A^-, A^+] + [B^-, B^+] := [A^- + B^-, A^+ + B^+]\,. \tag{5.3}$$

The interval extension of f now gives a bounding interval for $f(\tilde{a}, \tilde{b})$:

$$f(\tilde{a}, \tilde{b}) \overset{!}{\in} F(\tilde{A}, \tilde{B}) \equiv F\big([\tilde{A}^-, \tilde{A}^+], [\tilde{B}^-, \tilde{B}^+]\big)\,. \tag{5.4}$$

5.1.1 Set extension

The interval extension is a derivative of the *set extension*, a commonplace notion in mathematical reasoning where sets are used as arguments of functions defined for set elements. For example, one might write

$$\sin[0,\pi) = [0,1] \tag{5.5}$$

as a shorthand for

$$\big\{\sin x : x \in [0,\pi)\big\} = [0,1]\,, \tag{5.6}$$

therein using the implicitly defined set extension of a given unary function $f : \mathbb{R} \to \mathbb{R}$:

$$f : \mathcal{P}(\mathbb{R}) \to \mathcal{P}(\mathbb{R})\,, \mathcal{S} \mapsto \big\{f(x) : x \in \mathcal{S}\big\}\,, \tag{5.7}$$

where $\mathcal{P}(\mathcal{S})$ is the powerset of some set $\mathcal{S}$. This definition of the set extension is intuitive and generalises to functions with higher-dimensional domains and codomains.

Functions such as $\sin : \mathbb{R} \to \mathbb{R}$ are *computable* in the sense that they can be approximated in floating-point arithmetic in a straightforward manner, with efficient implementations widely available. For a function defined in terms of arithmetic operations, a floating-point equivalent of the function can then often be constructed by individually mapping the constituting operations to their floating-point equivalents, rendering the function computable as well.

This does not apply to set-extended functions: a set may hold a finite, a countably infinite, or an uncountably infinite number of elements, and the set extension of a function may map a set of either cardinality to a set with a different cardinality. A general set has no obvious numerical representation.

5.1.2 Interval extension

A practical way of handling the set extension numerically is to represent sets by *enclosing intervals*. Let us denote the set of all intervals in some contiguous closed subset $\mathcal{S} \subseteq \bar{\mathbb{R}}$ of the affinely extended real numbers $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ as

$$[\mathcal{S}] \coloneqq \big\{[a,b] : a,b \in \mathcal{S}, a \leqslant b\big\}\,. \tag{5.8}$$

For a function $f : \mathcal{S} \to \bar{\mathbb{R}}$, an *interval extension* $F : [\mathcal{S}] \to [\bar{\mathbb{R}}]$ of the function f is then characterised by the property of inclusion,

$$\forall X \in [\mathcal{S}]\, \forall x \in X : f(x) \in F\big(X\big)\,. \tag{5.9}$$

Although similar definitions can be made for half-open or open intervals $(X^-, X^+]$, $[X^-, X^+)$, (X^-, X^+), here we will consider only closed intervals $X \equiv [X^-, X^+]$ to keep it simple. An interval extension F of a function f is called *precise* if, for every interval X in the domain of f, $F(X)$ is identical to the *minimal enclosing interval* of the image of f on X:

$$\forall X \in [\mathcal{S}] : F\big(X\big) \overset{!}{=} \mathcal{I}\big[f(X)\big]\,, \tag{5.10}$$

where the minimal enclosing interval of some non-empty closed set $\mathcal{S} \subseteq \bar{\mathbb{R}}$ is defined as

$$\mathcal{I}[\mathcal{S}] := [\inf \mathcal{S}, \sup \mathcal{S}] \ . \tag{5.11}$$

The definitions of an interval extension and of the precise interval extension trivially generalise to functions of higher arity.

Precise interval extensions can be given for all elementary arithmetic operators, are trivially obtained for all monotonic unary functions, and can be obtained with some effort for piecewise monotonic functions.

5.1.3 Posits

Posits, also known as *type III unums*, are an alternative number format proposed by Gustafson and Yonemoto (2017), among other things designed to be simpler than IEEE 754 floats both conceptually and in terms of the required implementation effort. One of the key differences in the initial proposal was that '[there] are no "NaN" (not-a-number) bit representations with posits; instead, the calculation is interrupted, and the interrupt handler can be set to report the error and its cause, or invoke a workaround and continue computing' (Gustafson and Yonemoto, 2017, §2.4), although more recent revisions of the posit specification have revised the stance that invalid calculations must cause an interrupt (Posit Working Group, 2022).

Posits can represent one special value which is not a real number. This value is usually designated as complex infinity, $\pm\infty$. The revised posit specification subsumes the special value under the umbrella term *not a real* (NaR) and states the following guiding principle for it: 'If an operation usually produces real-valued output, any NaR input produces NaR output', making NaR semantically equivalent to the 'quiet NaN' value in IEEE 754 floating-point arithmetic, that is, a special value which propagates the state of invalidity through calculations. However, for the purpose of comparisons the posit specification treats the special value as negative infinity, $-\infty$: according to Gustafson and Yonemoto (2017, §2.4), '[posits] share the same $a < b$ relation as signed integers', and because the special value is represented as the bit pattern with only the most significant bit set, it corresponds to the smallest representable integer in a two's complement representation, and is thus considered 'less than' any other representable value, a corollary explicitly confirmed by the posit specification (Posit Working Group, 2022).

5.1.4 Valids

In Gustafson and Yonemoto (2017), posits were introduced along with *valids*, defined as 'a pair of equal-size posits, each ending in a ubit', with the ubit indicating 'whether a real number is an exact float or lies in the open interval between adjacent floats'. Arguing that posits and IEEE 754 floats were designed to be 'cheap, fast and "good enough" for an established set of applications', valids were proposed as the rigorous counterpart of posits, allowing determination of exact lower and upper bounds of numerically computed quantities, as can be achieved for floating-point numbers with traditional interval arithmetic.

A *valid* usually represents an open, half-open, or closed interval of real numbers. Unlike posits, valids have not been defined exactly by Gustafson (2017b). However, a

formal definition of valids has been proposed by Schärtl (2021), who also developed an implementation of posits and valids—to our knowledge, the only existing and publicly accessible implementation of valids at this time—as part of the *aarith* library (Keszöcze et al., 2021). The definition of valids given by Schärtl (2021) adheres to the principle of having all possible bit patterns correspond to legitimate value representations which had been upheld in the definition of posits and was reinforced with regard to valids in Gustafson (2017b). With this definition, a valid can represent the following types of sets:

- the *empty set* $\varnothing$;
- the *full set* $\mathbb{R} \cup \{\mathrm{NaR}\}$;
- *regular valids*, that is, unit sets containing a single real value $\{a\}$, the proper intervals $[a, b]$, $(a, b]$, $[a, b)$, (a, b), and the improper intervals $(-\infty, b]$, $(-\infty, b)$, $[a, \infty)$, (a, ∞), and $(-\infty, \infty) = \mathbb{R}$, where $a, b \in \mathbb{R}$, $a < b$;
- *irregular valids*, the complements of regular valids, that is, the set difference $(\mathbb{R} \cup \{\mathrm{NaR}\}) \backslash V$ of the full set and a given regular valid $V \subsetneq \mathbb{R}$; among them a unit set $\{\mathrm{NaR}\}$ containing the NaR value.

Valids are conceptually similar to intervals in traditional interval arithmetic, but can be considered superior in several ways, in particular because half-open and open intervals can be represented, whereas traditional interval arithmetic is defined only for closed intervals.

5.1.5 Composition theorem

Given some interval extensions F, G of the functions $f : \bar{\mathbb{R}} \to \bar{\mathbb{R}}$ and $g : \mathcal{S} \to \bar{\mathbb{R}}$, we observe that the composition of the interval extensions $F \circ G$ is an interval extension of the composition $f \circ g$:

$$\forall X \in [\mathcal{S}]\, \forall x \in X : (f \circ g)(x) \in \big(F \circ G\big)\big(X\big)\ . \tag{5.12}$$

This theorem, which can be trivially extended to n-ary functions, has been referred to as the 'Fundamental Theorem of Interval Arithmetic' (Hickey et al., 2001; IEEE Computer Society, 2015). It is the underlying foundation of most practical applications of interval arithmetic: given some closed-form scalar arithmetic expression, an interval extension of the expression can be obtained with a syntactical transformation by substituting every scalar operation with a corresponding interval extension. The interval extension of an arithmetic expression obtained in this manner is called the *natural interval extension* of the expression.

As an example, an interval extension of the algebraic expression

$$x^2 - 2x \tag{5.13}$$

can be obtained with a syntactic replacement by replacing real-valued operation with its interval extension, yielding a composition of interval extensions,

$$X^2 - 2X\ , \tag{5.14}$$

where we defined some elementary interval operations as

$$[X^-, X^+]^2 := \begin{cases} \left[0, (\max\{-X^-, X^+\})^2\right] & \text{if } 0 \in X \\ \left[(X^-)^2, (X^+)^2\right] & \text{if } X^- > 0 \\ \left[(X^+)^2, (X^-)^2\right] & \text{if } X^+ < 0 \end{cases} \tag{5.15}$$

$$a[X^-, X^+] := \begin{cases} \left[aX^-, aX^+\right] & \text{if } a \geqslant 0 \\ \left[aX^+, aX^-\right] & \text{if } a < 0 \end{cases} \tag{5.16}$$

$$-[X^-, X^+] := [-X^+, -X^-] \tag{5.17}$$

$$X + Y := [X^- + Y^-, X^+ + Y^+]\,. \tag{5.18}$$

We only state the definitions for closed intervals for simplicity, but analogous definitions for half-open and open intervals are known.

The notion of mapping scalar arithmetic expressions to interval-valued expressions and taking advantage of their composability is then known as *interval arithmetic* (Moore, 1966; Moore et al., 2009; Alefeld and Herzberger, 2012). It has found a variety of uses, among them systematic containment of numerical rounding errors, propagation of uncertainties, constraint programming, and error analysis (e.g. Kearfott, 2000; Kearfott and Kreinovich, 2013). Efforts have been made towards efficient implementation with IEEE 754 floating-point numbers (Hickey et al., 2001) and efficient use of existing SIMD instructions for interval computations (Goualard, 2009). The IEEE Standard 1788 for Interval Arithmetic (IEEE Computer Society, 2015, 2018) formalises a set of concepts and operations common to different flavours of interval arithmetic and also specifies optional IEEE 754 compatibility. Software packages for interval arithmetic are available for various programming languages (e.g. Brönnimann et al., 2006; Goualard, 2015) which supply the interval extensions of elementary arithmetic operations as building blocks, often overloading arithmetic operators to allow for a natural syntactical representation which resembles the original real-valued calculation.

Despite the availability of a general mapping of scalar-valued expressions to interval-valued expressions, it would be unreasonable to expect that regular scalar code could automatically be extended to intervals. Even when an automatic transformation is possible, the results are often worse—that is, they yield intervals wider than necessary—compared to an interval-specific implementation, largely due to the *dependency problem*, discussed in Appendix 5.A.

5.1.6 Relational predicates

Another reason why scalar code may be unfit for intervals is the ambiguity of relational predicates. Even a simple routine, such as an implementation of the $\max$ function

$$\max(a, b) := \begin{cases} b & \text{if } a < b \\ a & \text{if } a \geqslant b\,, \end{cases} \tag{5.19}$$

may have data-dependent branches, usually conditioned on Boolean relational predicates such as $a < b$. For real numbers a and b, such a relational predicate is either *true* or *false*. However, it is not clear how relational predicates for intervals would be interpreted. Given two intervals $A := [A^-, A^+]$ and $B := [B^-, B^+]$, the predicate $A < B$

might reasonably be *true* or *false* if the intervals are disjoint; but the relational predicate becomes ambiguous for overlapping intervals. If $A^- < B^+$ but $A^+ > B^-$, then for some values $a \in A$ and $b \in B$ the predicate $a < b$ holds *true*, but $a' < b'$ is *false* for other values $a' \in A$ and $b' \in B$.

Most software packages which implement interval arithmetic define Boolean relational predicates for intervals, resolving this ambiguity by settling for a particular interpretation. The two obvious interpretations are often referred to as 'certainly' and 'possibly' (e.g. Sun Microsystems, Inc., 2001; Brönnimann et al., 2006; Goualard, 2015), where 'certainly' uses the definition

$$A \sim B :\Leftrightarrow \forall a \in A, b \in B : a \sim b \tag{5.20}$$

and 'possibly' is defined by

$$A \sim B :\Leftrightarrow \exists a \in A, b \in B : a \sim b\,, \tag{5.21}$$

with '$\sim$' representing a comparison: $=$, $\neq$, $<$, $\leqslant$, $>$, $\geqslant$.

However, opting for just one interpretation of the relational operators has several unfortunate consequences. Resembling the scalar relational operators syntactically, they are easy to mistake for equivalence relations or total orderings which are usually defined by relational operators $=$, $<$, or $\leqslant$, thus leading to undefined behaviour when used with common algorithms and data structures. Their inherent and unavoidable logical inconsistencies lead to brittle code, the semantics of which can be inadvertently altered by seemingly innocuous transformations. In fact, this problem is not specific to interval arithmetic. A similar problem appears already with the relational predicates for IEEE 754 floating-point types, which define a special value *not a number* (NaN) that has no ordering relation with real-valued or infinite floats.

In Sect. 5.2, we study the relational predicates commonly defined for floats, intervals, and type III unums (posits and valids), pointing out their shortcomings and exploring some of the consequences. In Sect. 5.3, we propose an alternative definition of set-valued relational predicates, which we then employ to write *interval-aware code*, that is, code which can operate either on scalars or on intervals, by taking advantage of the expressive freedom of the C++ programming language. In Sect. 5.4, we demonstrate how interval bounds can be tightened by inferring constraints from relational predicates, and how this technique can be used to obtain better bounds. In Sect. 5.5, these concepts are then adapted to intervals of discrete sets such as integers and random-access iterators. Finally, our reference implementation is briefly discussed in Sect. 5.6.

5.2 Relational predicates

The set of real numbers $\mathbb{R}$ is totally ordered by the $\leqslant$ relation. This does not necessarily hold true for sets of floating-point values, which are often used to approximately represent real numbers in numerical calculations. However, the fact that the ordering is only partial is often ignored by programmers, leading to surprising and misleading results when dealing with unordered values.

```
template <typename T>
T max(T a, T b) {
    if (a < b) return b;
    else return a;
}
```

Listing 5.1: Definition of a generic max function (Eq. (5.19)) in C++.

```
constexpr float empty = NAN;
template <std::floating_point T>
T max(T a, T b) {
    T x = empty;
    if (a < b) x = b;
    if (a >= b) x = a;
    return x;
}
```

Listing 5.2: NaN-aware definition of the max function in C++.

```
template <typename T>
T max2(T a, T b) {
    T x = empty;
    if (possibly(a < b)) assign_partial(x, b);
    if (possibly(a >= b)) assign_partial(x, a);
    return x;
}
```

Listing 5.3: Interval-aware definition of a generic max function in C++.

```
template <typename T>
T max3(T a, T b) {
    T x = empty;
    auto c = (a < b);
    if (possibly(c)) {
        auto bc = constrain(b, c);
        assign_partial(x, bc);
    }
    if (possibly(!c)) {
        auto ac = constrain(a, !c);
        assign_partial(x, ac);
    }
    return x;
}
```

Listing 5.4: Optimal interval-aware definition of a generic max function in C++.

5.2.1 IEEE 754 floating-point numbers

The most prominent example of a set of floating-point values not totally ordered by the $\leqslant$ relation is the IEEE 754 floating-point format (IEEE Computer Society, 2019). In addition to a finite subset of the real numbers, an IEEE 754 floating-point value, referred to as 'float' henceforth, can represent positive and negative infinity as well as a range of special values subsumed under the term 'not a number' (NaN). These values, which emerge for example as the result of undefined arithmetic operations such as $0 \div 0$, have no ordering relation with real-valued or infinite floats; they are unordered values in the partially ordered set of floats. NaN values tend to have 'contagious' semantics: according to the IEEE 754 standard, an arithmetic operation that involves at least one NaN value should produce a NaN as output value (IEEE Computer Society, 2019, §6.2.3). Although the philosophical and practical value of NaNs has been disputed—some authors have called it a 'logical error [to assign] a number to something that is, by definition, not a number' (Gustafson and Yonemoto, 2017)—, their contagious nature helps by propagating arithmetic undefinedness, clearly marking the result of a calculation invalid if it comprises an invalid operation or operates on invalid input, and thereby preventing oversight of mathematical or logical mistakes.

Unlike floats, integral and Boolean data types usually have no representation of an invalid value. Therefore, a NaN cannot be propagated to an integral or Boolean result. To avoid sweeping errors under the rug, the IEEE 754 standard specifies that attempting to convert a NaN value to an integer type without a NaN representation shall cause an invalid operation exception (IEEE Computer Society, 2019, §5.8). Likewise, the standard defines *unordered-signaling predicates* $=$, $>$, $\geqslant$, $<$, $\leqslant$, 'intended for use by programs *not* written to take into account the possibility of NaN operands' (IEEE Computer Society, 2019, §5.11), which cause an invalid operation exception if either operand is NaN.

Unfortunately, many implementations do not obey the IEEE 754 standard with regard to float-to-integer conversion and relational predicates. Most C and C++ compilers (e.g. GCC, Clang, Microsoft Visual C++) do not raise an invalid operation exception when converting a NaN value to an integer type, and the relational operators $=$, $>$, $\geqslant$, $<$, $\leqslant$ for floats are usually taken to be the *unordered-quiet predicates* which evaluate to *false* if either operand is NaN. With NaNs, the negations of unordered-quiet predicates $>$, $\geqslant$, $<$, $\leqslant$ are not equivalent to their inversion: for instance, the predicate $x > \mathrm{NaN}$ is *false*, and thus its logical negation $\neg(x > \mathrm{NaN})$ is *true*; but the inverse predicate $x \leqslant \mathrm{NaN}$ is also *false*, and therefore $\neg(x > y)$ and $x \leqslant y$ are not equivalent.

The consequences are most easily observed with the mundane library function `max`, of which a possible definition is given in Listing 5.1. This definition of the `max` function has two flaws: it is not strictly commutative, and it does not always propagate NaN values. Both flaws are readily demonstrated by evaluating `max(0.f, NAN)` and `max(NAN, 0.f)`, which yield `0.f` and `NaN`, respectively. The flaws can both be attributed to the non-equivalence of negated and inverted relational predicates: the `else` clause, which amounts to a logical negation of the `if` condition in Line 2, has the semantic meaning '$a \geqslant b$ or a, b unordered', which differs from the semantic meaning '$a \geqslant b$' that was intended by the programmer.

One possibility of rectifying these flaws would be to explicitly check for a partial ordering. We instead opt to express the function with a default value and *without using logical negations*, cf. Listing 5.2. Instead of expressing the piecewise function definition as the usual `if`–`else` chain, we use independent `if` statements with the understanding that *none of them might be executed*, in which case the default value `empty` is returned.

Expressing the relational conditions without logical negations resulted in the desired semantics in this case; but the actual rules for NaN-sensitive programming are necessarily more complicated. A NaN-aware program may employ logical negations but must place them carefully to ensure that a NaN argument ends up 'on the correct side' of a relational predicate. For example, let an interval $[a, b]$ be defined by the two numbers a and b with $a \leqslant b$. We could ensure that two floats `a`, `b` define a valid interval with the following runtime check:

```
if (!(a <= b)) throw std::invalid_argument("no interval");
```

This code appears to be unnecessarily convoluted, and an unsuspecting maintainer might be tempted to 'simplify' it:

```
if (a > b) throw std::invalid_argument("no interval");
```

But this apparent simplification changes the semantics of the code with regard to NaN arguments: the first version throws an exception if either `a` or `b` is NaN, whereas the second version does not.

Both examples highlight the inherent brittleness of NaN-sensitive code. Additionally, the fact that the unordered-quiet flavour of the equality predicate evaluates to *false* if either argument is NaN is often overlooked but has confusing implications of its own. A NaN value does not compare equal with itself, implying that negated self-comparison can evaluate to *true*:

```
if (a != a) throw std::invalid_argument("a is NaN");
```

Being non-reflexive, the equality predicate thus cannot be an equivalence relation, which undermines an elementary assumption commonly made for regular types. For general-purpose algorithms and data structures which rely on an equivalence relation (e.g. hash sets), or which require that an equivalence relation can be obtained from the relational predicate `<` using logical negation and the identity $a = b \Leftrightarrow \neg(a < b \vee a > b)$ (e.g. sort, binary search), the occurrence of NaNs thus silently causes undefined behaviour, which may become manifest in 'impossible' runtime actions or seemingly inexplicable corruption of data structures.

To formalise our observations: a set of relational predicates $=, \neq, <, \leqslant$ shall be called *logically consistent* if the following identities hold for all values a, b:

$$a = b \quad \Leftrightarrow \quad \neg\,(a \neq b)\ , \tag{5.22}$$

$$a < b \quad \Leftrightarrow \quad \neg\,(b \leqslant a)\ , \tag{5.23}$$

$$a = b \quad \Leftrightarrow \quad \neg\,(a < b \vee b < a)\ . \tag{5.24}$$

The unordered-quiet predicates for floats are not logically consistent, as neither of the three identities holds true in the presence of the NaN value. NaN-sensitive code thus must not rely on either of these identities.

5.2.2 Posits

Because the NaR value is treated as negative infinity in the context of relational comparisons, a conforming posit implementation will exhibit no surprising behaviour with relational comparisons: the relational predicates are logically consistent as per Eqs. (5.22–5.24), the equality predicate $=$ defines an equivalence relation, and the predicates $<$ and $\leqslant$ define a strict and a non-strict total order, respectively, on the set of all posit values. A `max` function implemented as in Listing 5.1 remains commutative in the face of NaRs; however, it does not promote invalidity: if only one argument is NaR, the other argument, which always compares greater than NaR, will be returned.

Diverging from the posit specification, some implementations of posits (e.g. Schärtl, 2021) do away with the ambivalent interpretation of the NaR value and consistently take it to be complex infinity, defining the ordering predicate $<$ as *false* if either operand is $\pm\infty$. With this interpretation, the set of relational predicates is not logically consistent, and the incongruence discussed for NaN floats in Sect. 5.2.1 will arise.

5.2.3 Valids and intervals

Lacking a formal definition, the exact semantics originally envisioned for valids, in particular the semantics of the relational predicates for valids, are not known. However, valids are modelled after *ubounds* (cf. Gustafson, 2017b, §3.6), an interval-like construct based on type I unums; and '[for] a ubound u to be less than a ubound v, the maximum of u must be less than the minimum of v' (Gustafson, 2017a, §8.1). The ubound interpretation of relational ordering thus corresponds to 'certainly' semantics of Eq. (5.20). In the implementation of Schärtl (2021), the same semantics had been chosen for valids, with the caveat that the NaR value is treated as complex infinity, and thus as unordered with regard to real numbers.

Gustafson (2017a) also discusses predicates for equality and inequality. Given two valids X, Y, an obvious candidate for an equality relation would be set equality, which is an equivalence relation. However, as noted in Gustafson (2017a, §8.2), a different notion of equality, there referred to as 'not nowhere equal', is often more useful. This notion matches the 'possibly' interpretation of Eq. (5.21). Unlike the set equality relation, it does not constitute an equivalence relation because it is not transitive. Likewise, one could also define an equality relation with 'certainly' semantics; in this interpretation, two sets would be considered 'certainly equal' only if either set is empty or if both sets are identical single-element sets, and the $=$ relation would not be an equivalence relation because it is not reflexive.

Whichever interpretation of the relational predicates is chosen, it can easily be verified that the set of interval predicates $=, \neq, <, \leqslant$ cannot be logically consistent. Checking the definitions of Eqs. (5.22–5.24) with the counterexample $a = b = [0, 1]$, we find all three identities invalid using either the 'certainly' or the 'possibly' semantics for relational predicates. The consequences are similar to those discussed for floats with NaN values in Sect. 5.2.1. To illustrate, let us consider how the generic implementation of the max function given in Listing 5.1 would behave if the relational predicates adhered to 'certainly' or 'possibly' semantics. Comparing with the set extension of the $\max$ func-

A	B	fiducial result $\mathrm{Max}(A,B)$	'possibly' semantics $\mathtt{max}(A,B)$	$\mathtt{max}(B,A)$	'certainly' semantics $\mathtt{max}(A,B)$	$\mathtt{max}(B,A)$
$[0,2]$	$[3,6]$	$[3,6]$	$[3,6]$	$[3,6]$	$[3,6]$	$[3,6]$
$[2,4]$	$[3,6]$	$[3,6]$	$[3,6]$	$[2,4]$	$[2,4]$	$[3,6]$
$[4,5]$	$[3,6]$	$[4,6]$	$[3,6]$	$[4,5]$	$[4,5]$	$[3,6]$

Table 5.1: Results of the generic max function in Listing 5.1 for different interval arguments assuming either 'possibly' or 'certainly' semantics for relational predicates, compared to fiducial results as per Eq. (5.25).
Wrong results are highlighted with red background, excessive results (i.e. resulting intervals which contain but also exceed the fiducial result) are highlighted with yellow background.

tion

$$\mathrm{Max}(X,Y) := \{\max(x,y) \,|\, x \in X, y \in Y\} \ , \tag{5.25}$$

here taken as the fiducial result, we see in Table 5.1 that, for both 'certainly' and 'possibly' semantics, the generic implementation of the max function is not correct, and not even commutative.

5.3 Set-valued logic

According to Gustafson (2017b), posits and valids are considered 'two modes of operation, selectable by the user'. Although the referenced work acknowledges that 'algorithms for valids are often quite different from the ones for floats, and vice versa', it implies that valids and posits should be exchangeable to some extent, for instance in the recommendation that 'valids are for where you need a provable bound on the answer. Or when you are still developing an algorithm and debugging its numerical behavior'. Traditional interval arithmetic has often been used in a similar manner, relying on the 'Fundamental Theorem of Interval Arithmetic' discussed in Sect. 5.1. Defining relational predicates seems to be worthwhile and even necessary to allow writing generic code which can work with either scalar values or with intervals or valids. This section explores how to define relational predicates in a way that retains logical consistency and avoids the pitfalls discussed in Sect. 5.2.

5.3.1 Set-extended relational predicates

Although Boolean relational predicates can be and have been defined for floats with NaNs, for intervals, and for valids, in the preceding sections we have seen that they are logically inconsistent, which makes them error-prone even in targeted use. Moreover, they lack properties commonly expected from relational predicates in generic code such as reflexivity and transitivity, and thus easily lead to unexpected or undefined behaviour.

The core of the problem is the definition of relational predicates as Boolean predicates. For real numbers, the Boolean relational predicates $=$, $\neq$, $<$, $\leqslant$ are intuitive, well-defined, and logically consistent; $=$ is an equivalence relation, $<$ defines a strict total

$A \wedge B$		B			
		None	*False*	*True*	*Both*
A	*None*	*None*	*None*	*None*	*None*
	False	*None*	*False*	*False*	*False*
	True	*None*	*False*	*True*	*Both*
	Both	*None*	*False*	*Both*	*Both*

$A \vee B$		B			
		None	*False*	*True*	*Both*
A	*None*	*None*	*None*	*None*	*None*
	False	*None*	*False*	*True*	*Both*
	True	*None*	*True*	*True*	*True*
	Both	*None*	*Both*	*True*	*Both*

A	$\neg A$
None	*None*
False	*True*
True	*False*
Both	*Both*

Table 5.2: Truth tables for conjunction, disjunction, and negation in the $\mathcal{P}(\mathbb{B})$ logic as per Eqs. (5.28–5.30).

order, and $\leqslant$ defines a non-strict total order. But let us consider how floats with NaNs would be represented mathematically. A float can be either some real number or 'not a number'; in mathematical terms, it is either a unit set $\{x\}$ containing a real number $x \in \mathbb{R}$, or the empty set. Likewise, valids can represent certain subsets of $\mathbb{R} \cup \{\mathrm{NaR}\}$, including the empty set. But if floats and intervals represent sets of real numbers, then relational predicates of floats or intervals should represent sets of Booleans. Given a Boolean predicate $\sim\colon \mathbb{R} \times \mathbb{R} \to \mathbb{B}$, where $\mathbb{B} = \{\mathit{false}, \mathit{true}\}$ denotes the two-element Boolean algebra, we can thus define a general set extension of the predicate as

$$A \sim B \coloneqq \{a \sim b \,|\, a \in A, b \in B\} \tag{5.26}$$

for any two subsets $A, B \subseteq \mathbb{R}$.

Expressing this in a procedural manner, to assemble the relational predicate $A \sim B$ we start with an empty set. Then, if any pair of elements $a \in A$, $b \in B$ exists for which the Boolean relational predicate $a \sim b$ does not hold true, we insert *false* into the set. Likewise, if any pair of elements $a \in A$, $b \in B$ exists for which the Boolean relational predicate $a \sim b$ holds true, we insert *true* into the set. We end up with one of the following sets: $\varnothing$, $\{\mathit{false}\}$, $\{\mathit{true}\}$, or $\{\mathit{false}, \mathit{true}\}$, in other words: with an element of the *powerset of the two-element Boolean algebra*, $\mathcal{P}(\mathbb{B})$, which itself is a 4-valued logic. We identify its elements with the following intuitive names:

$$\mathit{None} \equiv \varnothing\,,\ \mathit{False} \equiv \{\mathit{false}\}\,,\ \mathit{True} \equiv \{\mathit{true}\}\,,\ \mathit{Both} \equiv \{\mathit{false}, \mathit{true}\}\,. \tag{5.27}$$

A	possibly A	always A	contingent A	vacuous A
None	*false*	*true*	*false*	*true*
False	*false*	*false*	*false*	*false*
True	*true*	*true*	*false*	*false*
Both	*true*	*false*	*true*	*false*

Table 5.3: Truth tables for Boolean projections from $\mathcal{P}(\mathbb{B})$ as per Eqs. (5.31–5.34).

We can infer the truth tables for the $\mathcal{P}(\mathbb{B})$ logic, reproduced in Table 5.2, by defining their logical operators $\wedge$, $\vee$, $\neg$ as set extensions of the corresponding Boolean operators:

$$U \wedge V := \{u \wedge v \,|\, u \in U, v \in V\}, \tag{5.28}$$
$$U \vee V := \{u \vee v \,|\, u \in U, v \in V\}, \tag{5.29}$$
$$\neg U := \{\neg u \,|\, u \in U\} \tag{5.30}$$

for any two subsets of the Boolean powerset logic $U, V \in \mathcal{P}(\mathbb{B})$. When defined as a set extension as per Eq. (5.26), the relational predicates $=$, $\neq$, $<$, $\leqslant$ are logically consistent because the identities given in Eqs. (5.22–5.24) hold.

Programs usually employ relational predicates to make binary choices: to jump or not to jump, tertium non datur. Thus, if relational predicates are $\mathcal{P}(\mathbb{B})$-valued instead of $\mathbb{B}$-valued, a program must be able to interpret them as Boolean value somehow. We thus need to define suitable projections. Following up on the previous attempts to define relational predicates for intervals, we find the two intuitive projections possibly and always, defined as

$$\text{possibly}\, U := \mathit{true} \in U\,, \tag{5.31}$$
$$\text{always}\, U := \mathit{false} \notin U \tag{5.32}$$

for any set-valued element $U \in \mathcal{P}(\mathbb{B})$. We emphasise that, by virtue of its antisymmetric definition, the always projection yields vacuous truth for an empty set U. We go on to define two more useful projections:

$$\text{contingent}\, U := (U \equiv \mathit{Both})\,, \tag{5.33}$$
$$\text{vacuous}\, U := (U \equiv \varnothing)\,. \tag{5.34}$$

Truth tables for all four projections are shown in Table 5.3.

With the possibly and always projections, we can interpret a given relational predicate as having either 'possibly' or 'certainly' semantics. But unlike in previous attempts at defining relational predicates for intervals, the semantic interpretation is not attached to the predicate itself. A projection can be applied to a more complex logical expression as well, such as possibly $(x < -1 \vee x > 1)$ or always $(a \leqslant x < b)$, allowing for intuitive expressive freedom. Because the relational operators do not evaluate to Boolean values, they are not directly compatible with algorithms or data structures expecting Boolean predicates, thereby forcing the user to explicitly choose a particular projection.

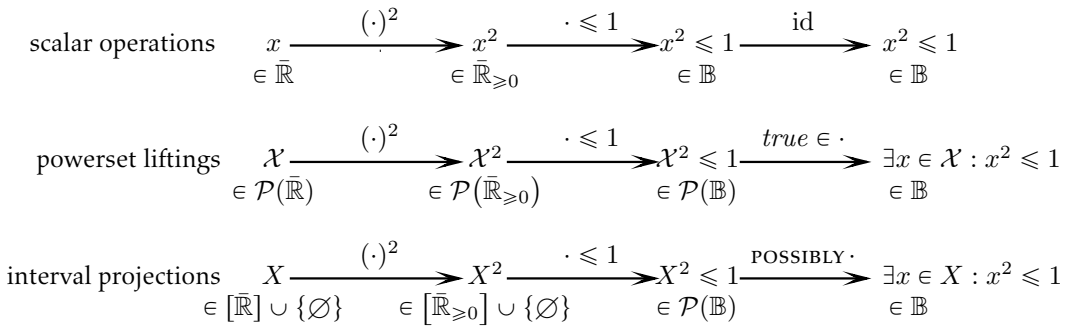


Figure 5.1: Diagram of algebraic relations between scalar arithmetic, arithmetic lifted to powersets, and interval arithmetic.
The set extension, cf. Eq. (5.7), is an intuitive lifting of arithmetic operations from sets to powersets. Because sets have no obvious numerical representation, we project sets $\mathcal{X}$ to enclosing intervals $\mathcal{I}[\mathcal{X}]$, which can be represented with a tuple of floating-point numbers. Instead of applying arithmetic operations to sets, we perform surrogate computations on their interval projections.

The relation of interval arithmetic and Boolean powerset logic is perhaps best understood by viewing interval arithmetic as a projection of set arithmetic. Functions can be trivially 'lifted' from elements to sets of elements, as was done for unary real-valued functions in Eq. (5.7). The 'lifting' of a function then maps sets from the powerset of the function's domain to the powerset of its codomain. Thus, a predicate $p : \bar{\mathbb{R}} \to \mathbb{B}$ would be 'lifted' to a set-valued predicate $P : \mathcal{P}(\bar{\mathbb{R}}) \to \mathcal{P}(\mathbb{B})$. One could say that we use interval arithmetic as a computable surrogate of set arithmetic, in which Boolean powerset logic naturally emerges. The algebraic relations are visualised with an example in Figure 5.1.

5.3.2 Writing NaN-aware code

We revisit the precondition-checking examples from Sect. 5.2.1. Given two scalar numbers a and b, we want to verify that they define a valid interval by asserting that $a \leqslant b$. If floats were considered sets containing either a single real number or nothing to indicate NaN ('not a number'), and if the relational operators for floats were defined according to Eq. (5.26) and returned $\mathcal{P}(\mathbb{B})$ values, we would write the check as

```
if (!possibly(a <= b)) {
    throw std::invalid_argument("no interval");
}
```

If either `a` or `b` was NaN, the predicate `a <= b` would evaluate to `None`, and `possibly(None)` is `false`; therefore, a NaN value will trigger the exception.

The check can be simplified to avoid the negation. However, the negation of a Boolean condition is not the same as a logical negation of the set-valued predicate. Thanks to the Boolean projections, the difference is evident: $\neg \text{POSSIBLY}\, U$ is clearly a different logical statement than $\text{POSSIBLY}\, \neg U$. We find that the following intuitive identities hold among

projections:

$$\neg\textsc{always}\,U \;\Leftrightarrow\; \textsc{possibly}\,\neg U \tag{5.35}$$
$$\neg\textsc{possibly}\,U \;\Leftrightarrow\; \textsc{always}\,\neg U \tag{5.36}$$
$$\textsc{contingent}\,U \;\Leftrightarrow\; \textsc{possibly}\,U \wedge \textsc{possibly}\,\neg U \tag{5.37}$$
$$\textsc{vacuous}\,U \;\Leftrightarrow\; \textsc{always}\,U \wedge \textsc{always}\,\neg U \tag{5.38}$$

Using Eq. (5.36), we can thus rewrite the check as

```
if (always(!(a <= b))) {
    throw std::invalid_argument("no interval");
}
```

and, taking advantage of logical consistency, we can get rid of the negation by inverting the relational operator with the identity Eq. (5.23):

```
if (always(a > b)) {
    throw std::invalid_argument("no interval");
}
```

which is as close as possible to the simplest but NaN-ignorant condition $a > b$ in Sect. 5.2.1.

5.3.3 Interval extension of piecewise-defined functions

With the interval-extended relational operators and the Boolean projections available, we can now specify how to obtain interval extensions for piecewise-defined functions.

Given a series of contiguous subsets $\mathcal{S}_1, \ldots, \mathcal{S}_n \subseteq \bar{\mathbb{R}}$, a series of functions $g_1 : \mathcal{S}_1 \to \bar{\mathbb{R}}, \ldots, g_n : \mathcal{S}_n \to \bar{\mathbb{R}}$ defined on these subsets, and a series of Boolean predicates $p_1, \ldots, p_n : \bar{\mathbb{R}} \to \mathbb{B}$, let a function $f : \bar{\mathbb{R}} \to \bar{\mathbb{R}}$ be defined by cases:

$$f(x) := \begin{cases} g_1(x), & \text{if } p_1(x) \\ \quad\vdots & \quad\vdots \\ g_n(x), & \text{if } p_n(x) \end{cases} \tag{5.39}$$

Now, let us assume that, for every $j \in \{1, \ldots, n\}$, $G_j : [\mathcal{S}_j] \to [\bar{\mathbb{R}}]$ is an interval extension of the function g_j, and $P_j : \mathcal{P}(\bar{\mathbb{R}}) \to \mathcal{P}(\mathbb{B})$ is the set extension of the Boolean predicate $p_j : \bar{\mathbb{R}} \to \mathbb{B}$. Then, the notation

$$F(X) := \begin{cases} G_1(X), & \text{if } P_1(X) \\ \quad\vdots & \quad\vdots \\ G_n(X), & \text{if } P_n(X) \end{cases} \tag{5.40}$$

is taken to refer to the minimal enclosing interval of the union of the intervals $G_j(X)$ for which the Boolean projection $\textsc{possibly}\,P_j(X)$ is *true*,

$$F(X) := \mathcal{I}\left[\bigcup_{j\in\mathcal{J}} G_j(X)\right] ,$$
$$\mathcal{J} := \left\{j \in \{1, \ldots, n\} : \textsc{possibly}\,P_j(X)\right\} . \tag{5.41}$$

In other words, the resulting interval is defined to enclose the intervals obtained by all branches $G_j(X)$ for which the predicate $p_j(x)$ might evaluate to *true* for any value $x \in X$. The function $F : [\bar{\mathbb{R}}] \to [\bar{\mathbb{R}}]$ therefore is an interval extension of the function $f : \bar{\mathbb{R}} \to \bar{\mathbb{R}}$. Again, the definition can be trivially generalised to functions and predicates of higher arity.

Using the basic notation introduced in Eq. (5.40), we can now obtain an interval extension of the max function by a syntactic transformation of Eq. (5.19),

$$\mathrm{Max}\left\{A, B\right\} := \begin{cases} B & \text{if } A < B \\ A & \text{if } A \geqslant B\,. \end{cases} \tag{5.42}$$

5.3.4 Writing interval-aware code

Taking advantage of the syntactic resemblence of a piecewise-defined function and its interval extension as defined in Sect. 5.3.3, we can express a routine like the max function such that it yields correct results for floats, posits, intervals, and valids (and also for floats with NaNs if the C++ language actually allowed redefining their built-in comparison operators).

In Listing 5.3 we give a new definition of a generic max function. The function signature is identical to the first definition in Listing 5.1, but the function was restructured in a fashion similar to the version in Listing 5.2: instead of `if`–`else` chains, `if` branches are expressed independently with the understanding that *none, one, or multiple branches might be executed*. Instead of directly returning a result, values are now accumulated in a local `result` variable, which is initialised with some generic empty value, `empty`, which equals NaN for floats, NaR for posits, an empty set for valids, or an uninitialised value for float-valued intervals. The branch condition $a < b$ is assumed to be set-valued, $(a < b) \in \mathcal{P}(\mathbb{B})$, and hence a Boolean projection has to be used to make a branching decision.

For multi-valued arguments `a` or `b` (i.e. intervals or valids), the two Boolean conditions `possibly(a < b)` and `possibly(a >= b)` are not necessarily exclusive: if both arguments overlap, both conditions evaluate to *true*, and both branches are executed. To avoid that an assignment from the previous branch is overwritten, conditional assignment must be performed with a function `assign_partial`. For floats or posits, `assign_partial(x, a)` asserts that `x` is still `empty` and then executes the assignment `x = a`. For intervals and valids, `assign_partial(x, a)` widens the set represented by `x` such that it encloses `a`.

Some results for the revised `max2` function of Listing 5.3 are shown in Table 5.4. The `max` routine is now commutative and returns correct results for all cases, although some results are too wide.

We emphasise that the C++ language does not allow to redefine operators such as `==` and `<` for built-in data types such as `float`. Therefore, when calling the revised `max2` function with arguments of type `float`, relational comparisons would in reality exhibit the traditional *unordered-quiet* behaviour. To actually obtain the results for NaN floats in Table 5.4, a user-defined data type would have to be created which wraps `float` but has set-valued relational operators.

domain type	A	B	fiducial result $\mathrm{Max}(A,B)$	experimental results `max2`(A,B)	`max2`(B,A)
floats	NaN	0	NaN	NaN	NaN
posits	NaR	0	0	0	0
posits	NaR	NaR	NaR	NaR	NaR
intervals	$[0,2]$	$[3,6]$	$[3,6]$	$[3,6]$	$[3,6]$
intervals	$[0,2]$	$[3,6]$	$[3,6]$	$[3,6]$	$[3,6]$
intervals	$[2,4]$	$[3,6]$	$[3,6]$	$[2,6]$	$[2,6]$
intervals	$[4,5]$	$[3,6]$	$[4,6]$	$[3,6]$	$[3,6]$
valids	NaR	0	0	0	0
valids	NaR	NaR	NaR	NaR	NaR
valids	$[2,4]$	$(3,\infty)$	$(3,\infty)$	$[2,\infty)$	$[2,\infty)$
valids	$[2,4]$	$[-\infty,6]$	$[2,6]$	$[-\infty,6]$	$[-\infty,6]$
valids	$[2,4]$	$3](6$	$[2,\infty)$	$[-\infty,\infty)$	$[-\infty,\infty)$

Table 5.4: Results of the revised interval-aware generic $\max$ function in Listing 5.3 for different interval arguments, compared to fiducial results as per Eq. (5.25).
Excessive results (that is, resulting intervals which contain but also exceed the fiducial result) are highlighted with yellow background. For floats and intervals, NaN is considered semantically equivalent to the empty set. For posits and valids, NaR is treated as negative infinity. Intervals cannot represent the empty set, but our interval datatype permits an uninitialised state. With '$3](6$' we denote the irregular valid $(\mathbb{R} \cup \{\mathrm{NaR}\})\backslash(3,6]$.

5.4 Constraints

To understand why some of the results returned by `max2` are wider than necessary, consider the example of $a = [2,4]$, $b = [3,6]$. Because the two intervals overlap, both relational predicates $a < b$ and $a \geqslant b$ evaluate to Both, which is then mapped to true by the POSSIBLY projection. The `assign_partial(x, b)` statement in the first branch finds `x` uninitialised and thus initialises it with $[3,6]$, the value of `b`. In the second branch, the statement `assign_partial(x, a)` widens the interval `x` such that it encloses all of `a`.

This behaviour is clearly suboptimal. Although not obvious in this case, it is also wrong. The underlying problem is more clearly illustrated by a different example in which the domains of the piecewise subfunctions actually differ. Let a function $s : \bar{\mathbb{R}} \to \bar{\mathbb{R}}$ be defined as

$$s(x) := \begin{cases} \sqrt{x} & \text{if } x \geqslant 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{5.43}$$

and implemented conventionally in Listing 5.5. We note that the first case $\sqrt{x}$ is defined only on a domain $\bar{\mathbb{R}}_{\geqslant 0}$. Therefore, if we attempt to construct an interval extension of s

```
float clampedSqrt(float x) {
    if (x >= 0) return sqrt(x);
    else return 0;
}
```

Listing 5.5: Definition of a clamped square root function in C++.

```
template <typename T>
T clampedSqrt2(T x) {
    T y = empty;
    if (possibly(x >= 0)) assign_partial(y, sqrt(x));
    if (possibly(x < 0)) assign_partial(y, 0);
    return y;
}
```

Listing 5.6: Interval-aware but ill-defined clamped square root function in C++.

as per Eq. (5.40),

$$S(X) := \begin{cases} \sqrt{X} & \text{if } X \geqslant 0 \\ 0 & \text{if } X < 0 \,, \end{cases} \tag{5.44}$$

and implement it as demonstrated in Listing 5.6, we end up with an undefined result for interval arguments reaching below 0,

$$S([-1,1]) = \mathcal{I}\Big[\sqrt{[-1,1]}, 0\Big] = \,? \tag{5.45}$$

because $\sqrt{\cdot}$ is not well-defined on codomain $\bar{\mathbb{R}}$ for negative arguments. However, as per its predicate, the first case of the piecewise-defined function should contribute only for arguments $\geqslant 0$. Practically spoken, when passing the interval $[-1,4]$ to the `clampedSqrt2` function in Listing 5.6, both branches are executed, and the `sqrt` function, which is not defined for negative arguments, is thus called with an interval argument $[-1,4]$. Thus, in defining the interval extension $F(X)$ for a piecewise-defined function $f(x)$, we need to somehow constrain the interval X for the individual cases according to the guarding predicate.

Because the first case in Eq. (5.43) is guarded by the condition $x \geqslant 0$, we know that the expression $\sqrt{x}$ will be evaluated only for non-negative arguments x. One can argue that, similarly, in Eq. (5.44) the interval X could be constrained to non-negative values before evaluating $\sqrt{X}$. The following interval extension would be perfectly well-defined:

$$S(X) := \begin{cases} \sqrt{X \cap \bar{\mathbb{R}}_{\geqslant 0}} & \text{if } X \geqslant 0 \\ 0 & \text{if } X < 0 \,, \end{cases} \tag{5.46}$$

How, and under which circumstances, can the necessary constraints of interval arguments be inferred from the guarding predicates?

```
template <typename T>
T clampedSqrt3(T x) {
    T y = empty;
    auto c = (x >= 0);
    if (possibly(c)) {
        auto xc = constrain(x, c);
        assign_partial(y, sqrt(xc));
    }
    if (possibly(!c)) {
        assign_partial(y, 0);
    }
    return y;
}
```

Listing 5.7: Corrected interval-aware clamped square root function in C++.

5.4.1 Inferring constraints

We first note that, for general predicates $P : [\bar{\mathbb{R}}] \to \mathcal{P}(\mathbb{B})$, no simple rule for automatically inferring constraints on the argument can be given. In order to 'read off' a constraint on an interval X from a relational expression, the expression would need to be recast as a logical combination of constraints of type '$X \sim A$', which may require inverting an arbitrary function. For example, the constraint $X^2 \leqslant 1$ would have to be transformed to the expression $X \geqslant -1 \wedge X \leqslant 1$. However, if we impose that constraints are to be stated in a normalised form composed of '$X \sim A$'-type relational expressions, which often is already the case in piecewise-defined functions, we can give a general rule for inferring constraints for intervals X from logical expressions. We emphasise that the requirement of linearity is imposed only for practical reasons; it could be overcome by an implementation capable of symbolically solving the predicate expression for the variable to be constrained.

In the following, we define the *constraint operator* $X|P$ for an interval X and a set-valued predicate expression P. First, the predicate expression P is evaluated; if it is *False*, the constraint operator has no effect,

$$X|P := X \quad \text{if } P = \mathit{False} \,. \tag{5.47}$$

If P evaluates to either *True* or *Both*, the constraint operator is defined as follows. For any relational expression which does not directly refer to the interval being constrained, the constraint operator has no effect:

$$X|(Y \sim Z) := X \,. \tag{5.48}$$

For elementary relational constraints of X, we have the following rules:

$$X|(X = A) := X \cap A \,, \tag{5.49}$$
$$X|(X \geqslant A) := \left[\max\{X^-, A^-\}, X^+\right] , \tag{5.50}$$
$$X|(X \leqslant A) := \left[X^-, \min\{X^+, A^+\}\right] \tag{5.51}$$

using the notation $X \equiv [X^-, X^+]$, $A \equiv [A^-, A^+]$ throughout. In order to infer optimal constraints from the relational operators $\neq$, $<$ and $>$, we would need to be able to

represent half-open and open intervals. But our system only handles closed intervals for reasons of simplicity, and so we have to leave relational expressions with $\neq$, $<$, and $>$ under-constrained:

$$X|(X \neq A) \coloneqq X \ , \tag{5.52}$$
$$X|(X > A) \coloneqq X|(X \geqslant A) \ , \tag{5.53}$$
$$X|(X < A) \coloneqq X|(X \leqslant A) \ . \tag{5.54}$$

We then define recurrence relations for predicate expressions composed from predicate subexpressions by logical operators $\wedge$ and $\vee$:

$$X|(P_1 \wedge P_2) \coloneqq \mathcal{I}\big[(X|P_1) \cap (X|P_2)\big] \ , \tag{5.55}$$
$$X|(P_1 \vee P_2) \coloneqq \mathcal{I}\big[(X|P_1) \cup (X|P_2)\big] \ , \tag{5.56}$$

pointing out that the constraint $X|(P_1 \wedge P_2)$ would not be well-defined if $P_1 \wedge P_2$ evaluated to *False* because then $X|P_1$ and $X|P_2$ would not overlap.

Any logical negation $\neg$ in a predicate expression shall be resolved by applying De Morgan's laws and by inverting relational operators:

$$\neg(P_1 \wedge P_2) \to (\neg P_1) \vee (\neg P_2) \ , \tag{5.57}$$
$$\neg(P_1 \vee P_2) \to (\neg P_1) \wedge (\neg P_2) \ , \tag{5.58}$$
$$\neg(X = Y) \to X \neq Y \ , \tag{5.59}$$
$$\neg(X \neq Y) \to X = Y \ , \tag{5.60}$$
$$\neg(X < Y) \to X \geqslant Y \ , \tag{5.61}$$
$$\neg(X \leqslant Y) \to X > Y \ , \tag{5.62}$$
$$\neg(X > Y) \to X \leqslant Y \ , \tag{5.63}$$
$$\neg(X \geqslant Y) \to X < Y \ . \tag{5.64}$$

Using the constraint operator defined by this set of rules, we can now refine the interval extension method for piecewise-defined functions previously given in Eq. (5.41):

$$F(X) \coloneqq \mathcal{I}\left[\bigcup_{j \in \mathcal{J}} G_j(X|P_j)\right] \ ,$$
$$\mathcal{J} \coloneqq \big\{j \in \{1, \ldots, n\} : \textsc{possibly}\, P_j(X)\big\} \ , \tag{5.65}$$

that is, when evaluating the interval-valued function G_j, we automatically constrain its argument X by its guarding predicate expression P_j.

With the refined definition of Eq. (5.65), the interval extensions of the $\max$ and s functions in Eqs. (5.42) and (5.44), which were obtained by a simple syntactic transformation of the scalar expressions in Eqs. (5.19) and (5.43), become well-defined and precise interval extensions.

The automatic inference of constraints is demonstrated in the final revisions of the `max` and `clampedSqrt` functions in Listings 5.4 and 5.7. The updated results for the `max3` function are presented in Table 5.5. In both code listings, the logical condition is stored in

domain type	A	B	fiducial result $\mathrm{Max}(A,B)$	experimental results `max3`(A,B)	`max3`(B,A)
intervals	$[2,4]$	$[3,6]$	$[3,6]$	$[3,6]$	$[3,6]$
intervals	$[4,5]$	$[3,6]$	$[4,6]$	$[4,6]$	$[4,6]$
valids	$[2,4]$	$(3,\infty)$	$(3,\infty)$	$(3,\infty)$	$(3,\infty)$
valids	$[2,4]$	$[-\infty,6]$	$[2,6]$	$[2,6]$	$[2,6]$
valids	$[2,4]$	$3](6$	$[2,\infty)$	$[2,\infty)$	$[2,\infty)$

Table 5.5: Results of the revised interval-aware generic $\max$ function in Listing 5.4 for different interval arguments, compared to fiducial results as per Eq. (5.25).
Results which are now optimally tight are highlighted with green background.

a variable `c`, from which the `constrain` function then infers which constraints the given interval can be subjected to. In `max3`, Line 6, the `constrain` function imposes the constraint $a < b$ on the variable b and returns its constrained value; for example, if a and b are closed intervals, $a \equiv [A^-, A^+]$, $b \equiv [B^-, B^+]$, then `constrain(b,c)` infers the constrained interval

$$\left[B_c^-, B_c^+\right] = \begin{cases} (A^-, B^+] & \text{if } A^- \geqslant B^- \\ [B^-, B^+] & \text{if } A^- < B^-, \end{cases} \tag{5.66}$$

in the first case omitting the part of the interval $[B^-, A^-]$ for which the condition can never be fulfilled. Because traditional interval arithmetic does not represent half-open intervals, the slightly excessive closed interval $[A^-, B^+]$ is produced instead of the correct interval $(A^-, B^+]$. With valids, the half-open interval can be represented correctly, and hence more accurate constraints are possible.

Thanks to inference of constraints, both `max3` and `clampedSqrt3` are well-defined interval extensions returning optimal results. In Listing 5.8, we also demonstrate how the technique applies to multiple branches, specifically for clamped linear interpolation,

$$y(x) = \begin{cases} y_1 & \text{if } x < x_1 \\ y_1 + \frac{x-x_1}{x_2-x_1}\,(y_2 - y_1) & \text{if } x_1 \leqslant x \leqslant x_2 \\ y_2 & \text{if } x > x_2\,, \end{cases} \tag{5.67}$$

also yielding optimal results. The inferred constraint ensures that the quotient $(x - x_1)(x_2 - x_1)$ in the $x_1 \leqslant x < x_2$ branch never exceeds $[0, 1]$.

5.5 Discrete-valued intervals

In Sect. 5.1, interval notation and the interval extension have been defined for affinely extended real numbers. But the notion of interval arithmetic can be easily extended to discrete ordered sets such as integers or to entities resembling integer differences, such as pointers, or random-access iterators in C++.

```
template <typename T, typename FloatT>
T interpolateLinear(T x,
        FloatT x1, FloatT x2, FloatT y1, FloatT y2) {
    T y = empty;
    auto below = (x < x1);
    if (possibly(below)) assign_partial(y, y1);
    auto above = (x > x2);
    if (possibly(above)) assign_partial(y, y2);
    auto c = !below & !above;
    if (possibly(c)) {
        auto xc = constrain(x, c);
        assign_partial(y,
            y1 + (xc - x1)/(x2 - x1)*(y2 - y1));
    }
    return y;
}
```

Listing 5.8: Interval-aware linear interpolation in C++.

The definitions of the set of intervals in Eq. (5.8), of the interval extension in Eqs. (5.9) and (5.10), and of the minimum enclosing interval in Eq. (5.11) can be naturally adapted to discrete domains and to operations defined thereon.

Like for real numbers, it is possible to construct precise interval extensions with constant computational complexity for elementary arithmetic operators and for monotonic unary functions, and the composition theorem in Eq. (5.12), which also applies to functions defined on discrete-valued domains, can be used to construct interval extensions, albeit not necessarily precise ones, with constant computational cost.

In a discrete ordered set, an element i can be uniquely associated with its predecessor and its successor, which we refer to as $\operatorname{pred} i$ and $\operatorname{succ} i$. For integers, predecessor and successor can simply be defined as

$$\operatorname{pred} i := i - 1 \,, \quad \operatorname{succ} i := i + 1 \,. \tag{5.68}$$

5.5.1 Precise discrete interval extensions

Because the values enclosed by an interval in a discrete domain can be enumerated, a precise interval extension of a discrete function can be obtained and even computed straightforwardly. Given a discrete function x_i for discrete arguments i and a discrete-valued interval $I \equiv [I^-, I^+]$ with finite width $|I| \equiv I^+ - I^- < \infty$, a computable precise interval extension of x_i can be obtained through enumeration:

$$X_I = \mathcal{I}\big[\{x_i : i \in I\}\big] \,. \tag{5.69}$$

We point out that the cost of computing the interval extension through enumeration is proportional to $1 + |I|$, which makes this approach viable only for relatively narrow intervals.

5.5.2 Constraining discrete-valued intervals

In Sect. 5.1.2 we opted to neglect the dedicated treatment of half-open and open intervals. As a consequence, when defining the constraint operator $X|P$ in Sect. 5.4.1 we had to leave intervals under-constrained for relational expressions with $\neq$, $<$, and $>$.

For discrete finite-valued domains, a distinction between closed, half-open, and open intervals is unnecessary because well-defined open or half-open intervals $[i, j)$, $(k, l]$, and (m, n) can be identified with closed intervals $[i, \operatorname{pred} j]$, $[\operatorname{succ} k, l]$, and $[\operatorname{succ} m, \operatorname{pred} n]$. Therefore, when applying relational constraints to discrete-valued intervals, we can now further refine the rules for the relational operators $\neq$, $>$, and $<$ given in Sect. 5.4.1, Eqs. (5.52–5.54):

$$I|\big(I \neq A\big) := \begin{cases} [\operatorname{succ} I^-, I^+] & \text{if } A^- = A^+ = I^- \\ [I^-, \operatorname{pred} I^+] & \text{if } A^- = A^+ = I^+ \\ I & \text{otherwise,} \end{cases} \tag{5.70}$$

$$I|\big(I > A\big) := \big[\max\{I^-, \operatorname{succ} A^-\}, I^+\big]\,, \tag{5.71}$$

$$I|\big(I < A\big) := \big[I^-, \min\{I^+, \operatorname{pred} A^+\}\big]\,. \tag{5.72}$$

5.5.3 Partitioned sequences

Discrete functions, represented here as sequences of values such as $(x_i)_{i=1}^n$, are often used for purposes of lookup. For example, let some continuous function $f(x)$ be sampled at n sampling points $x_1, \ldots, x_n$ and the sampling values be stored in a sequence $y_i := f(x_i)$. Then, to approximate the function $f(x)$ by a first-order interpolation scheme, the indices i, j of the sampling points x_i, x_j adjacent to the given value x are located, and an interpolated result is computed as a function of x_i, x_j, y_i, and y_j. The necessary lookup operations can be implemented with $\mathcal{O}(\log n)$ operations if the sampling points are ordered, $x_i < x_{i+1}$. We now define some operators useful for looking up element indices.

Let $(b_i)_{i=1}^n$, $b_i \in \mathbb{B}$ be a partitioned sequence of Boolean values, which is to say: the sequence is ordered such that the elements b_i which are *True* precede the elements which are *False*. Then, the *partition point index* operator is defined to evaluate to the index i of the first element b_i in the sequence which is *False*, or to $n+1$ if there is no such element:

$$\operatorname{part}\big[(b_i)_{i=1}^n\big] := \begin{cases} \min\limits_{i \in \{1,\ldots,n\}, \neg b_i} i & \text{if } \exists i \in \{1, \ldots, n\} : \neg b_i \\ n+1 & \text{otherwise}\,. \end{cases} \tag{5.73}$$

We note that, because $(b_i)_{i=1}^n$ is a partitioned sequence, the partition point index can be determined with binary search, and hence with $\mathcal{O}(\log n)$ operations.

For an ordered sequence $(x_i)_{i=1}^n$, $x_i \leqslant x_{i+1}$, we can define the *lower bound index* operator which evaluates to the index i of the first element x_i not smaller than the given comparand x. Together with the analogous *upper bound index* operator which identifies the index of the first element greater than x, it can be defined in terms of the partition

point index operator:

$$\text{lower}\big[(x_i)_{i=1}^n; x\big] := \text{part}\big[(x_i < x)_{i=1}^n\big] \,, \tag{5.74}$$
$$\text{upper}\big[(x_i)_{i=1}^n; x\big] := \text{part}\big[(x_i \leqslant x)_{i=1}^n\big] \,. \tag{5.75}$$

The precise interval extension of the partition point index operator, which will accept a sequence of Boolean powerset elements, $(K_i)_{i=1}^n$, $K \in \mathcal{P}(\mathbb{B})$, can be constructed as

$$\text{Part}\big[(K_i)_{i=1}^n\big] := \Big[\text{part}\big[(\textsc{always}\, K_i)_{i=1}^n\big], \text{part}\big[(\textsc{possibly}\, K_i)_{i=1}^n\big]\Big] \,. \tag{5.76}$$

Precise interval extensions of the lower bound and upper bound index operators can then be given by a natural syntactic transformation:

$$\text{Lower}\big[(x_i)_{i=1}^n; X\big] = \text{Part}\Big[\big(x_i < X\big)_{i=1}^n\Big] \,, \tag{5.77}$$
$$\text{Upper}\big[(x_i)_{i=1}^n; X\big] = \text{Part}\Big[\big(x_i \leqslant X\big)_{i=1}^n\Big] \tag{5.78}$$

for an interval argument $X \equiv [X^-, X^+]$.

5.5.4 Example: Nearest-neighbour interpolation

Let $\mathcal{S}$ denote a domain, and let a discrete sampling of some continuous function be given by an ordered sequence of sampling points $(x_i)_{i=1}^n$, $x_i \in \mathcal{S}$, $x_i < x_{i+1}$, and a corresponding sequence of sampling values $(y_i)_{i=1}^n$. A nearest-neighbour interpolation between $n \geqslant 2$ points of support can be defined in a directly computable form by means of the partition point index operator:

$$\begin{aligned}&\text{interp}^0\big[(x_i, y_i)_{i=1}^n; x\big] := y_k \\ &\text{with } k := \text{part}\Big[\big(\tfrac{1}{2}(x'_i + x'_{i+1}) < x\big)_{i=1}^{n-1}\Big] \,.\end{aligned} \tag{5.79}$$

wherein the expression

$$\frac{x_i + x_{i+1}}{2} \quad \text{for } i \in \{1, \ldots, n-1\} \tag{5.80}$$

denotes the midpoint between sampling points i and $i+1$. The precise interval extension of the operator $\text{interp}^0[(x_i, y_i)_{i=1}^n; x]$ is obtained by syntactically transforming scalar values x and k to intervals X and K,

$$\begin{aligned}&\text{Interp}^0\big[(x_i, y_i)_{i=1}^n; X\big] := y_K \\ &\text{with } K \equiv \text{Part}\Big[\big(\tfrac{1}{2}(x'_i + x'_{i+1}) < X\big)_{i=1}^{n-1}\Big] \,.\end{aligned} \tag{5.81}$$

5.5.5 Example: Linear interpolation

For $n \geqslant 2$, a linear interpolation of the samples can be constructed similarly,

$$\begin{aligned}&\text{interp}^1\big[(x_i, y_i)_{i=1}^n; x\big] := \begin{cases} y_1 & \text{if } k = 1 \\ \text{interp}^1_{k-1}\big[(x_i, y_i)_{i=1}^n; x\big] & \text{if } 1 < k \leqslant n \\ y_n & \text{if } k = n+1 \end{cases} \\ &\text{with } k := \text{lower}\big[(x_i)_{i=1}^n; x\big] \,,\end{aligned} \tag{5.82}$$

where for $1 \leqslant i < n$ the piecewise linear interpolation between sampling points i and $i+1$ is given by

$$\mathrm{interp}_i^1\big[(x_j, y_j)_{j=1}^n; x\big] \coloneqq y_i + \frac{x - x_i}{x_{i+1} - x_i}\,(y_{i+1} - y_i) \ . \tag{5.83}$$

Given the natural interval extension of Eq. (5.83),

$$\mathrm{Interp}_i^1\big[(x_j, y_j)_{j=1}^n; X\big] = y_i + \frac{X - x_i}{x_{i+1} - x_i}\,(y_{i+1} - y_i) \ , \tag{5.84}$$

an interval extension of the linear interpolation operator in Eq. (5.82) could be constructed with the usual syntactic transformations $x \to X$, $k \to K$ and with enumeration of $k \in K$ as per Eq. (5.69):

$$\begin{aligned}
&\mathrm{Interp}^1\big[(x_i, y_i)_{i=1}^n; X\big] = \begin{cases} y_1 & \text{if } K = 1 \\ \mathrm{Interp}_{K-1}^1\big[(x_i, y_i)_{i=1}^n; X\big] & \text{if } 1 < K \leqslant n \\ y_n & \text{if } K = n+1 \end{cases} \\
&\text{with } K \coloneqq \mathrm{Lower}\big[(x_i)_{i=1}^n; X\big] \ ,
\end{aligned} \tag{5.85}$$

However, we note that such an interval extension would be ill-defined. The piecewise linear interpolation between sampling points x_i and x_{i+1} is valid only for arguments x between the sampling points, $x_i \leqslant x \leqslant x_{i+1}$; therefore, if the interval X covers one or more points of support, $X^- < x_i < X^+$, it will exceed the domain of the interval-extended operator $\mathrm{Interp}_i^1[(x_j, y_j)_{j=1}^n; X]$. The enumeration rule of Eq. (5.69) is not sufficient here: when enumerating the indices i of the sampling points that enclose the interval X we must also constrain the interval such that $x_i \leqslant X \leqslant x_{i+1}$.

5.5.6 Partitioned function definitions

Let $\mathcal{S}$ be some domain and $\mathcal{T}$ some set, and let $(p_i)_{i=1}^n$ be a partitioned sequence of Boolean predicates $p_i : \mathcal{S} \to \mathbb{B}$, that is, a sequence ordered such that the sequence of Booleans $(p_i(x))_{i=1}^n$ is partitioned for every $x \in \mathcal{S}$. Also, let $(g_i)_{i=0}^n$ be a sequence of functions

$$\begin{aligned}
&g_0 : \mathcal{S}|(\neg P_1) \to \mathcal{T} \ , \\
&g_i : \mathcal{S}\big|\big(P_i \wedge \neg P_{i+1}\big) \to \mathcal{T} \quad \text{for } 1 \leqslant i < n \ , \\
&g_n : \mathcal{S}|P_n \to \mathcal{T}
\end{aligned} \tag{5.86}$$

where we naturally extend the constraint operator to sets,

$$\mathcal{S}|P \coloneqq \big\{x \in \mathcal{S} : P(x)\big\} \ . \tag{5.87}$$

A *partitioned function* is a function $f(x)$ defined as

$$f(x) \coloneqq \begin{cases} g_0(x) & \text{if } \neg P_1(x), k = 0 \\ g_k(x) & \text{if } P_k(x) \wedge \neg P_{k+1}(x) \\ & \quad \text{for } 1 \leqslant k < n \\ g_n(x) & \text{if } P_n(x), k = n \end{cases} \tag{5.88}$$

with the branch k determined in $\mathcal{O}(\log n)$ steps as

$$k \equiv \mathrm{part}\Big[\big(P_i(x)\big)_{i=1}^{n}\Big] - 1 \; . \tag{5.89}$$

An interval extension of the function $f(x)$ is then given by

$$F(X) := \begin{cases} G_0\big(X\big) & \text{if } \neg P_1\big(X\big), k = 0 \\ G_k\big(X\big) & \text{if } P_k\big(X\big) \wedge \neg P_{k+1}\big(X\big) \\ & \quad \text{for } 1 \leqslant k < n \\ G_n\big(X\big) & \text{if } P_n\big(X\big), k = n \end{cases} \tag{5.90}$$

with the contributing branches $k \in K$ determined by enumeration,

$$k \in K \equiv \mathrm{Part}\Big[\big(P_i(X)\big)_{i=1}^{n}\Big] - 1 \; . \tag{5.91}$$

For the special case of $p_i(x) := x_i < x$ with an ordered sequence $(x_i)_{i=1}^{n}$, the partitioned function $f(x)$ simplifies to

$$f(x) := \begin{cases} g_0(x) & \text{if } x \leqslant x_1, k = 0 \\ g_k(x) & \text{if } x_k < x \leqslant x_{k+1} \\ & \quad \text{for } 1 \leqslant k < n \\ g_n(x) & \text{if } x_n < x, k = n \end{cases} \tag{5.92}$$

with the branch k determined as

$$k \equiv \mathrm{lower}\big[(x_i)_{i=1}^{n}; x\big] \; . \tag{5.93}$$

and analogously for $p_i(x) := x_i \leqslant x$ and $\mathrm{upper}\big[(x_i)_{i=1}^{n}; x\big]$.

We can now define the linear interpolation operator as a partitioned function:

$$\mathrm{interp}^1\big[(x_i, y_i)_{i=1}^{n}; x\big] := \begin{cases} y_1 & \text{if } x \leqslant x_1, k = 0 \\ \mathrm{interp}_k^1\big[(x_i, y_i)_{i=1}^{n}; x\big] & \text{if } x_k < x \leqslant x_{k+1} \\ & \quad \text{for } 1 \leqslant k < n \\ y_n & \text{if } x_n < x, k = n \; , \end{cases}$$
$$\text{with } k := \mathrm{lower}\big[(x_i)_{i=1}^{n}; x\big] \; . \tag{5.94}$$

The natural interval extension of this rendering now leads to the correct constraints being inferred for the argument interval X of the piecewise linear interpolation function $\mathrm{interp}_k^1[(x_i, y_i)_{i=1}^{n}; x]$:

$$\mathrm{Interp}^1\big[(x_i, y_i)_{i=1}^{n}; X\big] := \begin{cases} y_1 & \text{if } X \leqslant x_1, k = 0 \\ \mathrm{Interp}_k^1\big[(x_i, y_i)_{i=1}^{n}; X\big] & \text{if } x_k < X \leqslant x_{k+1} \\ & \quad \text{for } 1 \leqslant k < n \\ y_n & \text{if } x_n < X, k = n \; , \end{cases}$$
$$\text{with } k \in K := \mathrm{Lower}\big[(x_i)_{i=1}^{n}; X\big] \; . \tag{5.95}$$

```
template <std::floating_point T>
T
interpolate_nearest_neighbour(
        ranges::random_access_range auto&& xs,
        ranges::random_access_range auto&& ys,
        T x) {
    dim n = ranges::ssize(xs);
    assert(n >= 2);
    assert(ranges::ssize(ys) == n);

    auto it = ranges::partition_point(
        index_range(n - 1),
        [&](index i) {
            auto xhalf = std::midpoint(
                xs[i], xs[i+1]);
            return xhalf < x;
        });
    auto i = *it;
    return at(ys, i);
}
// usage:
// auto xs = std::array{ 1., 2., 4., 8.};
// auto ys = std::array{ 1., 3., 9., -3. };
// double x = ...;
// auto y = interpolate_nearest_neighbour(xs, ys, x);
```

Listing 5.9: C++ implementation of Eq. (5.79) (nearest-neighbour interpolation) for scalar types.

5.6 Implementation

In Sect. 5.3.3, we defined a notation for piecewise definitions of interval-valued functions which is syntactically equivalent to the usual notation for piecewise-defined scalar-valued functions, thereby extending the previously introduced notion of a natural interval extension to piecewise function definitions. The notion of interval-valued piecewise function definitions was further refined in Sect. 5.4.1 with the introduction of the constraint operator. In Sect. 5.5, we studied the extension of interval arithmetic to discrete domains, and we introduced the notion of a partitioned function definition as a generalisation of piecewise function definition.

Transformation rules resembling this transformation of piecewise-defined functions can be given for software routines operating on scalar values in order to obtain routines which operate on intervals, similar to how arithmetic computations are adapted for other extensions of scalar arithmetic that have monadic structure, for instance, vectorisation (Karpiński and McDonald, 2017; Mabille and Corlay, 2022), automatic differentiation (Hogan, 2014), or automatic propagation of Gaussian uncertainties. With the generic programming facilities of expressive languages such as C++, it is even possible to write routines in such a way that they can be compiled either for scalar values, with little or no decrease in efficiency compared to a specialised scalar version, or for interval values.

```
template <floating_point_interval_arg T>
T
interpolate_nearest_neighbour(
        ranges::random_access_range auto&& xs,
        ranges::random_access_range auto&& ys,
        T x) {
    dim n = ranges::ssize(xs);
    assert(n >= 2);
    assert(ranges::ssize(ys) == n);

    auto [_, it] = intervals::partition_point(
        index_range(n - 1),
        [&](index i) {
            auto xhalf = std::midpoint(
                xs[i], xs[i+1]);
            return xhalf < x;
        });
    auto i = *it;
    return at(ys, i);
}
// usage:
// auto xs = std::array{ 1., 2., 4., 8.};
// auto ys = std::array{ 1., 3., 9., -3. };
// auto x = ...; // ‘double‘ or ‘interval<double>‘
// auto y = interpolate_nearest_neighbour(xs, ys, x);
```

Listing 5.10: C++ implementation of Eq. (5.79) (nearest-neighbour interpolation) as a generic function, and instantiation for scalar types and for interval types

We have developed a C++ library, aptly named *intervals* (Beutel, 2022), which implements traditional interval arithmetic along with $\mathcal{P}(\mathbb{B})$-valued relational predicates, Boolean projections, and automatic inference of constraints. We note that, unlike other packages such as GAOL (Goualard, 2015), our implementation does not yet have special precautions for floating-point rounding, such as ensuring that the lower bound is always rounded towards $-\infty$ and the upper bound is rounded towards $+\infty$ to make sure the true result is included in the result interval, and although it was crafted with efficiency in mind, no effort has been made to ensure that certain patterns of machine code such as vector instructions are generated.

The library defines a class template `interval<T>`, which represents a closed interval of values of a floating-point, integer, or random-access iterator type `T`, and a class template `set<B>` which represents an element of the powerset of the type `B`. Unary and binary arithmetic operations `+`, `-`, `*`, `/` are overloaded for `interval<T>` if `T` is an arithmetic type, and interval overloads are provided for standard mathematical functions such as `sin(x)`, `sqrt(x)`, or `log(x)`, and for some mathematical functions not present in the C++ Standard Library such as `square(x)` or `frac(x)`.

Relational operators `<`, `<=`, `>`, `>=`, `==`, `!=` are defined for arguments of either `interval<T>` and `set<bool>` type, all mapping to types convertible to `set<bool>`, and comparison opera-

tors `==`, `!=` are defined for all types `set<T>`. The operators for logical conjunction, disjunction and negation `&`, `|`, `!` – but not the short-circuiting variants `&&` and `||`[1] – are defined for `set<bool>`.

Additional function definitions such as `possibly`, `constrain`, `assign_partial` are provided to enable interval-aware programming as demonstrated in Listings 5.4, 5.7, and 5.8. When instantiated for non-interval types, these functions degenerate to trivial operations and introduce no runtime overhead, in particular `possibly(c)` becomes `c`, `constrain(x,c)` becomes `x`, and `assign_partial(y,x)` becomes `y = x`.

With the *intervals* library, functions can be implemented in an *interval-aware manner*, that is, with assignments and branches expressed such that an interval extension can automatically be generated by the compiler. For this to work, branch conditions must be expressed such that all interval-conditioned branches can be taken. This implies that we cannot use `else` clauses on `if` clauses with interval-dependent conditions, nor can we use interval-conditioned `goto`-like statements such as `return`, `break`, or `continue`.

5.6.1 Discrete interval extensions

A scalar implementation of the nearest-neighbour interpolation operator (Eq. (5.79)) is shown in Listing 5.9. A generic version which supports both scalar and interval types is shown in Listing 5.10; the only required change is replacing `std::ranges::partition_point` with the algorithm of the same name from the `intervals` package. The `std::ranges::partition_point` algorithm from the C++ Standard Library implements the partition point index operator (Eq. (5.73)); it accepts a range and a predicate the range is partitioned by and returns an iterator referencing the partition point. The 'intervals' package defines a `partition_point` algorithm with a similar interface which can operate on either Boolean or $\mathbb{K}_3^{\mathbb{S}}$ predicates and returns an iterator or an interval of iterators, respectively. With the `index_range` and `index_iterator` classes, index values can be used as iterators, which enables the use of index-based instead of value-based predicates in algorithms designed for iterable ranges. An index-based predicate is necessary here because in Line 15 we access two adjacent values to compute the midpoints between sampling points `xhalf`, cf. Eq. (5.80). In Line 18, dereferencing the index iterator `it` extracts its index, and in Line 19, the `i`-th element of range `ys` is returned. When calling `interpolate_nearest_neighbour` with an interval-valued argument `x`, `partition_point` returns an interval of iterators `it`, which upon dereferencing becomes an interval of indices `i`. Calling the `at` function with an index interval returns an interval of values using enumeration (Eq. (5.69)).

5.6.2 Partitioned functions and discrete-valued constraints

Our final example demonstrates how a partitioned function can be implemented generically in C++ using the example of the linear interpolation operator (Eq. (5.94)). In Listing 5.11, a scalar implementation of the linear interpolation operator is given, and evolved to a generic scalar/interval version in Listing 5.12.

The lower-bound and upper-bound index operators lower and upper were defined as special cases of the partition point index operator part. Likewise, the C++ Standard

[1] As per conventional advice, the short-circuiting operators `&&` and `||` should never be overloaded (e.g. Meyers, 1995, Item 7), which we nevertheless tried and quickly came to regret.

Library defines algorithms `std::ranges::lower_bound` and `std::ranges::upper_bound` as special cases of the `std::ranges::partition_point` algorithm. These algorithms accept an ordered range of elements and a comparand and return an iterator to the lower-bound or upper-bound element in the range. The 'intervals' package defines similar algorithms `lower_bound` and `upper_bound` which support both scalar-valued and interval-valued comparands and return an iterator or a range of iterators, respectively. In addition, they return a range of predicates which can be used to constrain a comparand interval.

To make piecewise linear interpolation interval-aware, as has been done in Listing 5.12, we use the `lower_bound` implementation from the *intervals* library.

In Listings 5.11 and 5.12, the lower-bound iterator of comparand `x` is obtained with the `lower_bound` algorithm in Line 11. When called with an interval argument `lower_bound` returns a set of predicates `preds` and an interval of lower-bound iterators `pos`. An interval of segment indices `i` can then be obtained by subtraction, as is done in Line 12. After the boundary cases have been dealt with in Lines 14–22, the piecewise linear interpolation is handled in Line 23ff. Given a discrete-valued interval `i`, the function `enumerate(i)` returns the range of values in the interval. In Line 25ff, the indices in the interval of indices `i` are then iterated over, and for each index the piecewise linear interpolation is computed, wherein the argument `x` is constrained by `preds[i]`, the partitioning predicate which corresponds to the range index `i`. When instantiated for normal floating-point types `T`, the code in 5.12 is equivalent to Listing 5.11 because the loop over j collapses to a single iteration with $j = i$.

We note that the refined inference of constraints on discrete-valued intervals in Eqs. (5.70–5.72) is crucial here. In Line 24, the interval of indices `i` is constrained by the expression `!below & !above`, which expands to `(i != 0) & (i != n)`. The `constrain` function will thus narrow `i` such that it contains neither the start index `i=0` nor the end index `i=n` of the range `xs`, which would result in out-of-bound range indexing in Lines 26–29.

5.6.3 Posits and valids

As a proof of concept, we ported the interval-aware infrastructure of the *intervals* library to posits and valids starting from the posit and valid implementation developed by Schärtl (2021) as part of the *aarith* library (Keszöcze et al., 2021), thereby demonstrating that *valid-aware programs* can be written using the same principles. The multitude of possible combinations of closed, half-open, open intervals and irregular valids renders constraint inference much more complicated than for the closed intervals of traditional interval arithmetics, but in reward the inferred constraints can be more precise than with traditional interval arithmetic.

5.7 Discussion

The transformation of general unconstrained Boolean control flow and assignment (including `else` clauses and `goto`-like statements) to interval-extendable control flow and conditional assignment could to some extent be automated. In fact, the proposed model of interval-extendable control flow bears many similarities to the stream-processing model used to program GPUs or SIMD registers of CPUs, and compilers such as CUDA

```
template <std::floating_point T>
T
interpolate_linear(
        ranges::random_access_range auto&& xs,
        ranges::random_access_range auto&& ys,
        T x) {
    dim n = ranges::ssize(xs);
    assert(n >= 2);
    assert(ranges::ssize(ys) == n);

    auto pos = ranges::lower_bound(xs, x);
    index i = pos - ranges::begin(xs);

    auto result = T{ };
    bool below = (i == 0);
    if (below) {
        result = ys[0];
    }
    bool above = (i == n);
    if (above) {
        result = ys[n-1];
    }
    if (!below && !above) {


        auto x0 = xs[i-1];
        auto x1 = xs[i];
        auto y0 = ys[i-1];
        auto y1 = ys[i];

        result =
            y0 + (x-x0)/(x1-x0)*(y1-y0);

    }
    return result;
}
// usage:
// auto xs = std::array{ 1., 2., 4., 8.};
// auto ys = std::array{ 1., 3., 9., -3. };
// double x = ...;
// auto y = interpolate_linear(xs, ys, x);
```

Listing 5.11: C++ implementation of Eq. (5.94) (linear interpolation) for scalar types

```
template <floating_point_interval_arg T>
T
interpolate_linear(
        ranges::random_access_range auto&& xs,
        ranges::random_access_range auto&& ys,
        T x) {
    dim n = ranges::ssize(xs);
    assert(n >= 2);
    assert(ranges::ssize(ys) == n);

    auto [preds, pos] = intervals::lower_bound(xs, x);
    auto i = pos - ranges::begin(xs);

    auto result = T{ };
    auto below = (i == 0);
    if (possibly(below)) {
        assign_partial(result, ys[0]);
    }
    auto above = (i == n);
    if (possibly(above)) {
        assign_partial(result, ys[n-1]);
    }
    if (auto c = !below & !above; possibly(c)) {
        auto ic = constrain(i, c);
        for (index j : enumerate(ic)) {
            auto x0 = xs[j-1];
            auto x1 = xs[j];
            auto y0 = ys[j-1];
            auto y1 = ys[j];
            auto xc = constrain(x, preds[j]);
            assign_partial(result,
                y0 + (xc-x0)/(x1-x0)*(y1-y0));
        }
    }
    return result;
}
// usage:
// auto xs = std::array{ 1., 2., 4., 8.};
// auto ys = std::array{ 1., 3., 9., -3. };
// auto x = ...; // ‘double‘ or ‘interval<double>‘
// auto y = interpolate_linear(xs, ys, x);
```

Listing 5.12: C++ implementation of Eq. (5.94) (linear interpolation) as a generic function, and instantiation for scalar and interval types

(NVIDIA, 2022) or ISPC (Pharr and Mark, 2012) are known to automatically transform branches to masked operations, which are akin to partial assignments in interval-extendable code. However, it is unclear to which extent the algebraic transformations often necessary in order to obtain sufficiently narrow result intervals (cf. Sect. 5.1.5) can be mechanised as well. Also, a programmer may wish to improve either the tightness or the computational efficiency of an interval extension by taking advantage of mathematical properties of the underlying expression such as monotonicity of which a compiler does not know. Therefore, instead of investing significant engineering effort into a domain-specific compiler which can automate only part of the transformation anyway, we preferred to provide a lightweight set of tools and rules for interval-aware programming.

Although our library has been written in C++, we emphasise that the proposed paradigm is not specific to this language. In fact, the handling of logical constraints and branch assignment with `constrain` and `assign_partial`, while consistent and straightforward, is still verbose in our implementation; a domain-specific language aware of set-valued logical predicates and relational constraints could provide more concise syntax and automatically infer and apply constraints, possibly even non-linear constraints, inside branches.

With the tools presented here, existing numerical routines may be adapted to support interval-valued arguments. Although our effort was prompted by the need to compute interaction rate bounds for the bucketing scheme introduced in Chapter 4, the interval-aware programming paradigm may be useful for other purposes. For instance, interval arithmetic is often used to bound the numerical error necessarily incurred from finite-precision arithmetic. With the proposed programming paradigm, existing code may be more easily adapted for support of rounding error analysis.

The practical benefit of error analysis with interval arithmetic is often limited by the expansive nature of interval–interval products, yielding resulting intervals which are correct but useless due to being too large. Often, the results are unnecessarily excessive because interval arithmetic cannot account for correlations that may exist between operands. The desire to track correlations between quantities has led to the development of refined arithmetic concepts such as affine arithmetic (de Figueiredo and Stolfi, 2004). The paradigm proposed in this chapter can in principle be applied to such higher-order refinements of interval arithmetic as well, although we note that arithmetic operations on intervals and inference of constraints will be much more complicated.

5.8 Conclusion

In this chapter we have discussed the shortcomings of traditional definitions of relational predicates for IEEE 754 floating-point types, intervals, and valids. Pursuing the goal of writing *interval-aware code*, that is, code that can work either with numbers or with intervals, we have proposed a set-valued definition of relational predicates as a better alternative, overcoming the logical inconsistencies inherent to Boolean relational predicates on intervals and allowing for intuitive expression of conditionals with intervals. We have demonstrated that branch conditions composed from relational predicates can be used to constrain interval arguments inside the branch, and that the inference of

constraints from a logical predicate can be automated. We have tested the viability of the proposed ideas by developing a library, *intervals*, which implements the proposed techniques for traditional interval arithmetic. Additionally, we have augmented an existing implementation of valids with the proposed set-valued relational predicates and worked out the corresponding constraint inference logic for valids, showing that the same principles can be applied to write *valid-aware code*, allowing for more accurate constraints since valids can also represent half-open or open intervals.

Appendices

5.A The dependency problem

The *dependency problem* (e.g. Gustafson, 2017a, §16.2f), is a fundamental limitation of interval arithmetic which is rooted in the fact that correlations or dependencies between quantities cannot be represented in their bounding intervals, resulting in unnecessarily loose bounds. The problem becomes apparent already in the simple example first given in Eqs. (5.13–5.14) of Sect. 5.1.5: as per the Fundamental Theorem of Interval Arithmetic (IEEE Computer Society, 2015; Hickey et al., 2001), an interval extension of the algebraic expression

$$x^2 - 2x \tag{5.96}$$

can be obtained as the composition of interval extensions of the constituting operations,

$$X^2 - 2X\,, \tag{5.97}$$

where $X \equiv [X^-, X^+]$ denotes an interval, and where we relied on the definitions of some elementary interval operations given in Eqs. (5.15–5.18). Inserting the interval $X = [0, 1]$ into Eq. (5.97), we obtain

$$[0, 1]^2 - 2[0, 1] = [0, 1] - [0, 2] = [-2, 1]\,. \tag{5.98}$$

However, if we first recast the expression in Eq. (5.96) as

$$(x - 1)^2 - 1\,, \tag{5.99}$$

then inserting $X = [0, 1]$ into its syntactic interval extension $(X - 1)^2 - 1$ yields the much narrower and, in fact, optimal interval bounds $[-1, 0]$. The two scalar expressions of Eqs. (5.96) and (5.99) may be equivalent, but their syntactic interval extensions are not.

5.B Code dependencies

To avoid redundancy, the `#include` statements and some other declarations needed to compile the code were omitted in our listings. Listing 5.13 shows the statements and declarations that need to be prepended to each of the listings in this chapter.

```cpp
// C++ Standard Library
#include <array>
#include <cmath>
#include <ranges>
#include <cassert>
#include <cstddef>
#include <concepts>
#include <algorithm>
namespace ranges = std::ranges;
using std::sqrt;
using index = std::ptrdiff_t;
using dim = std::ptrdiff_t;

// intervals
#include <intervals/set.hpp>
#include <intervals/interval.hpp>
#include <intervals/algorithm.hpp>
using namespace intervals;
```

Listing 5.13: Common prefix code for all C++ examples which use the *intervals* library.

Combining deterministic and stochastic methods to simulate dust dynamics in protoplanetary disks 6

Wenn man nur an sich denkt, kann man nicht glauben, daß man Irrtümer begeht, und kommt also nicht weiter. Darum muß man an jene denken, die nach einem weiterarbeiten. Nur so verhindert man, daß etwas fertig wird.

Bertolt Brecht, *Geschichten vom Herrn Keuner*

The preceding chapters have been very technical, straying far from the original goal of the thesis, which we shall now recall: to study the growth of planetesimals under influence of a nearby dust-trapping planet, and to investigate under which conditions planetesimals in a dust trap can grow to protoplanetary mass.

In our envisioned dust trap scenario, planetesimals initially abound, and a stochastic method such as the Representative Particle Monte Carlo (RPMC) method discussed in Chapter 3 is therefore well-suited for simulating their dynamics. However, to account for the physical realities of a protoplanetary disk, a simulation must cover more than just coagulation processes. Collisions may result not only in coagulation but also in cratering, pulverisation, or bouncing. Also, bodies interact not only through collisions but gravitationally as well. The effects of mutual gravitational deflection of abundant bodies can be modelled statistically, which blends well with the Monte Carlo method used to simulate collisions. However, a statistical treatment of gravitational deflections breaks down as growing planetesimals enter the runaway regime, accumulating all mass in their gravitational reach and becoming unique objects. Likewise, we know of no statistical model to describe how a dust-trapping planet affects the kinetics of planetesimals in its vicinity. For lack of a surrogate model, the dynamics emerging from the gravitational forces exerted by heavy individual bodies must be simulated with an N-body code.

As explained in Chapter 1, the significance of the dust trap scenario stems from the fact that, without the presence of a planet or some other irregularity in the radial gas

profile, bodies approaching centimetre to metre size would rapidly drift inwards, which would deplete the material available for further growth. This drift barrier renders the formation of planetary embryos through continuous growth improbable in protoplanetary disks with homogeneous density profiles. By locally reversing the pressure gradient of the gas disk, a dust-trapping planet effectively suspends the drift barrier, allowing bodies to accumulate in a ring around the local pressure maximum, and thereby establishing favourable conditions for continued growth. The dust–gas interaction is well-studied, and analytical or semi-analytical models of the damping of orbital parameters by the gas disk have been proposed (Adachi et al., 1976; Inaba et al., 2001), which can be implemented as continuous operators.

The RPMC method simulates a series of discrete events. At its core, the method needs to stochastically choose the interarrival time of the subsequent event based on the current state of the physical system. This approach is obviously inaccurate if external factors may have changed the system by the time the next event arrives. A rough framework for external operators in a simulation of discrete events has been sketched in Sect. 4.2.7.

This chapter is structured as follows. We start in Sect. 6.1 by introducing the statistical model of mutual collisional and collisionless encounters devised by Ormel et al. (2010), which has been referred to as *Ormel's model* in Chapter 4. Ormel's model also incorporates a simple model of fragmentation. Although the subject of fragmentation had been briefly touched in Sect. 3.4.4.3, its appropriate treatment requires more effort. Sect. 6.2 elaborates how fragmentation can be implemented effectively in the extended RPMC method. Then, in Sect. 6.3 we assemble a comprehensive model of gas-induced damping for bodies of a wide range of masses, from sub-μm-sized dust grains to planets. Aiming to form a coherent interaction model of the constituent parts of a protoplanetary disk, in Sect. 6.4 we briefly sketch the design of a hybrid simulation method that combines the statistical models of collisional and collisionless encounters with the gas-induced damping model and the direct (N-body) approach for simulating the kinetics of massive individual bodies. We conclude the chapter in Sect. 6.5 by discussing the remaining challenges in constructing a comprehensive simulation method.

6.1 Ormel's model

In Sects. 2.3.2–2.3.3 we sketched the foundations of a representative model for simulating particle growth processes during planet formation. In particular, we introduced a set of very general equations for the evolution of particle masses and rms velocities (Eqs. (2.74;2.80–2.81)). We explained how this model naturally enables a stochastic simulation of the interaction process, arguing that, even though a Monte Carlo method might appear to be an approximation of a continuous process, this is not the case as the interactions modelled are of discrete nature. A collision between representative particle j and a particle represented by representative particle k is not a gradual process: it either does or does not occur, and if it occurs, the squared rms eccentricity $\langle e_j^2 \rangle$ is changed by the full $(\Delta e^2)_{jk}^{\mathrm{coll}}$, and (assuming a hit-and-stick collision) the mass m_j grows by m_k. Likewise, in the dynamical heating model, a close gravitational encounter is a discrete and not a gradual process. This is significant because certain encounters, such as heavy

bodies viscously stirring small bodies, lead to a rapid and discontinuous change in particle properties (we may again consider the gravity assist manoeuvre undertaken by a space probe as an example), which would render a continuous system of equations 'stiff', and thus challenging to solve with a continuous method.

The decomposition of the contributions to the change rate terms into an interaction rate and a change (Eqs. (2.75;2.82–2.83)) therefore occurs naturally, and expressions for the two components can be derived by making certain geometrical assumptions and imposing certain symmetries. In this section, we adapt such a geometrical model of the collision and stirring process from Ormel et al. (2010), who used it to simulate runaway growth of protoplanetary bodies. *Ormel's model*, as it shall be called henceforth, comprises collisional encounters (leading to coagulation or fragmentation), gravitational interactions (modelled as viscous stirring and dynamical friction), gas drag forces, and turbulent stirring. In the following, we present a slight generalisation of Ormel's model and give expressions for interaction rates and change terms.

Ormel's model is a representative particle model: the trajectories of n representative particles through their state space, extrapolating the full particle distribution from the set of representative particles. Throughout this chapter, we use the notation and terminology introduced for the RPMC method in Chapter 3, though we note that Ormel's model is not specific to a particular method and can be simulated by other means (as was done by Ormel et al. (2010) who used a different Monte Carlo method combined with a quasi-continuous evolution of rms velocities).

In our adaptation of Ormel's model, a representative particle k has the following properties: *particle mass* m_k, *swarm mass* M_k and *swarm particle count* N_k, which are related through $M_k = N_k m_k$; *orbital radius* r_k; *planar rms velocity* v_k and *vertical rms velocity* $v_{z,k}$; and *solid density* ρ_k. A swarm of particles k is assumed to be located in a radial annulus of $[r_k - h_{x,k}, r_k + h_{x,k}]$. The *scale length* is defined to be

$$h_{x,k} = \begin{cases} \max\left\{R_{\min}, \dfrac{v_k}{\Omega_{\mathrm{K}}(r_k)}\right\} & \text{for many-particles swarms} \\ \dfrac{v_k}{\Omega_{\mathrm{K}}(r_k)} & \text{for few-particles swarms.} \end{cases} \tag{6.1}$$

The simulation parameter $R_{\min}$ bounds the radial resolution of the statistical simulation; but since a few-particles swarm contains only one individual self-representing particle, it is not treated statistically, and no minimal effective scale length needs to be imposed. Similarly to the scale length, the *scale height* of swarm k is defined as

$$h_{z,k} = \frac{v_{z,k}}{\Omega_{\mathrm{K}}(r_k)} \tag{6.2}$$

(but see Sect. 6.3.2 and Eq. (6.88) regarding the scale height for dust particles).

For a representative particle j and a swarm k, we refer to the interaction rates for collisions, dynamical friction events, and viscous stirring events as $\lambda_{jk}^{\mathrm{coll}}$, $\lambda_{jk}^{\mathrm{df}}$, and $\lambda_{jk}^{\mathrm{vs}}$. Interaction rates can be computed by imposing very simple geometric assumptions built upon *interaction radii* that are specific to the type of interaction considered. Interaction radii constrain the radial and vertical separations at which two given particles from swarms j and k can possibly interact. Let us assume that we know the exact positions of particles

j and k at a given time. Specifically, let the radial and vertical separations of the two particles be quantified by Δr and Δz. Then, the two particles can possibly interact if and only if $R_{r,jk}^{\mathrm{int,min}} \leqslant \Delta r \leqslant R_{r,jk}^{\mathrm{int}}$ and $R_{z,jk}^{\mathrm{int,min}} \leqslant \Delta z \leqslant R_{z,jk}^{\mathrm{int}}$.

6.1.1 A simple geometrical interaction model

With a representative method, we cannot assume that the exact positions of particles are known. A representative particle may have a known specific position at a given time, but it may represent an entire swarm of particles that have similar properties but are situated elsewhere. For example, a dust particle may represent a spatially homogeneous ring of dust particles of a given mass. Therefore, when computing interaction rates $\lambda_{jk}^{\mathrm{int}}$ between swarms, certain assumptions on the spatial distribution of the swarms must be made. Such assumptions are embodied in a *geometrical model*. A geometrical model can be used to decompose the interaction rate $\lambda_{jk}^{\mathrm{int}}$ into a product of a interaction timescale τ_{jk}^{int} and geometric factors accounting for the overlap of the swarm distributions. The geometric factors must take into account the minimal and maximal interaction radii $R_{r,jk}^{\mathrm{int,min}}$, $R_{r,jk}^{\mathrm{int}}$, and $R_{z,jk}^{\mathrm{int,min}}$, $R_{z,jk}^{\mathrm{int}}$. These geometric factors are referred to as *filling factors*.

In our framework, the interaction timescale τ_{jk}^{int} is not further specified. A good estimate can often be found using dimensional analysis or by comparison with an existing local interaction model, as is demonstrated with an example below. In a realistic model, we may additionally calibrate the interaction timescale with a fiducial N-body simulation, as was done indirectly in Ormel et al. (2010) by comparing with models from literature.

Regarding the geometrical interaction model which Ormel's model is built upon, we first point out that the derivation of the interaction rate as provided by Ormel et al. (2010) is somewhat incomplete. As indicated by their Fig. 22, the authors appear to have used a rigorous geometrical model to compute their interaction rates; however, this model is not formally specified. The semi-formal sketch of its derivation in Appendix C is misleading, as the interaction model visualised in their Fig. 22 cannot be derived from the strictly local interaction rate stated in their Eq. (C1). The shortcomings of the local interaction model that would emerge from this equation are briefly explored in Appendix 6.A. In this section, we formally construct a proper geometrical model of non-local interactions equivalent to the geometrical model visualised in Fig. 22 of Ormel et al. (2010).

A general expression for the *non-local interaction rate* between any two particles from swarms j and k can be given as

$$\lambda_{jk} = \int \mathrm{d}^3x\, n_j(\mathbf{x}) \int \mathrm{d}^3x'\, n_k(\mathbf{x}') K_{jk}(\mathbf{x}' - \mathbf{x}) \,, \tag{6.3}$$

where $K_{jk}(\mathbf{x}' - \mathbf{x})$, of unit (time^{-1}), is the interaction kernel for two particles from swarms j and k at some distance $\Delta\mathbf{x} := \mathbf{x}' - \mathbf{x}$. As a simple geometrical approximation, the number density distribution of a swarm i is modelled as

$$n_i(\mathbf{x}) \equiv N_j n_i^{(1)}(\mathbf{x}) \,, \tag{6.4}$$

$$n_i^{(1)}(\mathbf{x}) \equiv r_i^{-1} n_{r,i}^{(1)}(r) \times n_{\varphi,i}^{(1)}(\varphi) \times n_{z,i}^{(1)}(z) \,, \tag{6.5}$$

where the '(1)' superscript indicates a number density distribution for a single particle of the swarm, and where the radial, azimuthal, and vertical distributions are defined as

$$\begin{aligned} n^{(1)}_{r,i}(r) &= \begin{cases} (2h_{x,i})^{-1} & (r_i - h_{x,i}) \leqslant r \leqslant (r_i + h_{x,i}) \\ 0 & \text{otherwise,} \end{cases} \\ n^{(1)}_{\varphi,i}(\varphi) &= (2\pi)^{-1} \,, \\ n^{(1)}_{z,i}(z) &= \begin{cases} (2h_{z,i})^{-1} & -h_{z,i} \leqslant z \leqslant h_{z,i} \\ 0 & \text{otherwise,} \end{cases} \end{aligned} \tag{6.6}$$

in other words, the swarm is assumed to be distributed homogeneously in a toroidal ring of central radius r_i and with rectangular cross-section $2h_{x,i} \times 2h_{z,i}$. We assume that any azimuthal features that may emerge are smoothed out due to the gravitational coupling of the particles in the swarm, and can thus be neglected. We further assume that the interaction kernel can be decomposed into an interaction timescale τ_{jk} and independent radial and vertical components,

$$K_{jk}(\mathbf{x}' - \mathbf{x}) \equiv \frac{1}{8}(\tau_{jk})^{-1}\, \kappa_{r,jk}(r' - r)\, \kappa_{z,jk}(z' - z) \,, \tag{6.7}$$

the factor $\frac{1}{8}$ being motivated geometrically as will become apparent later on. We define the dimensionless interaction terms $\kappa_{r,jk}(\Delta r)$, $\kappa_{z,jk}(\Delta z)$ such that particles can interact only if their vertical and orbital distance lies within the minimal and maximal interaction radii denoted as $R^{\min}_{r,jk}$, $R_{r,jk}$ and $R^{\min}_{z,jk}$, $R_{z,jk}$:

$$\begin{aligned} \kappa_{r,jk}(\Delta r) &\equiv \begin{cases} \left(2\,(R_{r,jk} - R^{\min}_{r,jk})\right)^{-1} & R^{\min}_{r,jk} \leqslant |\Delta r| \leqslant R_{r,jk} \\ 0 & \text{otherwise} \end{cases} \\ \kappa_{z,jk}(\Delta z) &\equiv \begin{cases} \left(2\,(R_{z,jk} - R^{\min}_{z,jk})\right)^{-1} & R^{\min}_{z,jk} \leqslant |\Delta z| \leqslant R_{z,jk} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{6.8}$$

Inserting these definitions into the non-local interaction rate (Eq. (6.3)), we find that we can decompose it as a product of the dimensionless radial and vertical *filling factors* $\phi_{r,jk}$, $\phi_{z,jk}$,

$$\lambda_{jk} = \frac{1}{8}(\tau_{jk})^{-1}\, N_j N_k\, \phi_{r,jk}\, \phi_{z,jk} \,, \tag{6.9}$$

which we define as

$$\phi_{r,jk} \coloneqq \int_0^\infty \mathrm{d}r\, r\, r_j^{-1} n^{(1)}_{r,j}(r) \int_0^\infty \mathrm{d}r'\, r'\, r_k^{-1} n^{(1)}_{r,k}(r') \kappa_{r,jk}(r' - r) \,, \tag{6.10}$$

$$\phi_{z,jk} \coloneqq \int \mathrm{d}z\, n^{(1)}_{z,j}(z) \int \mathrm{d}z'\, n^{(1)}_{z,k}(z') \kappa_{z,jk}(z' - z) \,. \tag{6.11}$$

Let us first evaluate the vertical filling factor $\phi_{z,jk}$:

$$\begin{aligned} \phi_{z,jk} &= \int \mathrm{d}b\, \kappa_{z,jk}(b) \int \mathrm{d}z\, n^{(1)}_{z,j}(z)\, n^{(1)}_{z,k}(z + b) \\ &\equiv \int \mathrm{d}b\, \kappa_{z,jk}(b)\, \xi_{z,jk}(b) \\ &\equiv \Xi_{z,jk}(R_{z,jk}) - \Xi_{z,jk}\big(R^{\min}_{z,jk}\big) + \Xi_{z,jk}\big(-R^{\min}_{z,jk}\big) - \Xi_{z,jk}(-R_{z,jk}) \end{aligned} \tag{6.12}$$

where we define the vertical distance distribution $\xi_{z,jk}(b)$,

$$\xi_{z,jk}(b) := \int \mathrm{d}z\, n^{(1)}_{z,j}(z) n^{(1)}_{z,k}(z+b) \;, \tag{6.13}$$

and its antiderivative $\Xi_{z,jk}(b)$, the cumulative vertical distance distribution,

$$\Xi_{z,jk}(b) := \int_{-\infty}^{b} \mathrm{d}b' \xi_{z,jk}(b') \;. \tag{6.14}$$

The vertical distance distribution is antisymmetric under transposition, $\xi_{z,jk}(b) = \xi_{z,kj}(-b)$; the filling factor $\phi_{z,jk}$ is thus symmetric under transposition because $\kappa_{z,jk}(\Delta z)$ is an even function. For notational convenience we therefore impose without loss of generality that $h_{z,j} \leqslant h_{z,k}$. In order to evaluate the filling factor, we then integrate $\xi_{z,jk}(b)$ piece-wise:

$$\begin{aligned}
\xi_{z,jk}(b) &= \Big[(2h_{z,j})(2h_{z,k})\Big]^{-1} \max\Big\{0; \min\{h_{z,j}; h_{z,k}-b\} - \max\{-h_{z,j}; -h_{z,k}-b\}\Big\} \\
&= \Big[(2h_{z,j})(2h_{z,k})\Big]^{-1} \times \begin{cases} 0 & \text{if } b \leqslant b_1 \\ b-b_1 & \text{if } b_1 \leqslant b \leqslant b_2 \\ 2h_{z,j} & \text{if } b_2 \leqslant b \leqslant b_3 \\ b_4 - b & \text{if } b_3 \leqslant b \leqslant b_4 \\ 0 & \text{if } b_4 \leqslant b \;, \end{cases}
\end{aligned} \tag{6.15}$$

$$\Xi_{z,jk}(b) = \Big[(2h_{z,j})(2h_{z,k})\Big]^{-1} \times \begin{cases} 0 & \text{if } b \leqslant b_1 \\ \frac{1}{2}(b-b_1)^2 & \text{if } b_1 \leqslant b \leqslant b_2 \\ 2h_{z,j}\,(b-b_2+h_{z,j}) & \text{if } b_2 \leqslant b \leqslant b_3 \\ (2h_{z,j})(2h_{z,k}) - \frac{1}{2}(b_4-b)^2 & \text{if } b_3 \leqslant b \leqslant b_4 \\ (2h_{z,j})(2h_{z,k}) & \text{if } b_4 \leqslant b \;, \end{cases} \tag{6.16}$$

where we defined $b_1 < b_2 \leqslant b_3 < b_4$ as

$$\begin{aligned}
b_1 &:= -h_{z,j} - h_{z,k} \;, \\
b_2 &:= h_{z,j} - h_{z,k} \;, \\
b_3 &:= -h_{z,j} + h_{z,k} \;, \\
b_4 &:= h_{z,j} + h_{z,k} \;.
\end{aligned} \tag{6.17}$$

To better understand this result, let us first consider the special case $R^{\min}_{z,jk} = 0$, $R_{z,jk} \ll h_{z,j}, h_{z,k}$, where we can approximate

$$\begin{aligned}
\xi_{z,jk}(b) &\approx \Big[(2h_{z,j})(2h_{z,k})\Big]^{-1} \begin{cases} \min\{h_{z,j}; h_{z,k}\} - \max\{-h_{z,j}; -h_{z,k}\} & \text{if } |b| \leqslant R_{z,jk} \\ 0 & \text{otherwise} \end{cases} \\
&= \begin{cases} (2h_{z,k})^{-1} & \text{if } |b| \leqslant R_{z,jk} \\ 0 & \text{otherwise,} \end{cases}
\end{aligned} \tag{6.18}$$

$$\Xi_{z,jk}(b) \approx \begin{cases} 0 & \text{if } b < -R_{z,jk} \\ (2h_{z,k})^{-1}(R_{z,jk}+b) & \text{if } -R_{z,jk} \leqslant b \leqslant R_{z,jk} \\ (2h_{z,k})^{-1}(2R_{z,jk}) & \text{if } R_{z,jk} \leqslant b \;, \end{cases} \tag{6.19}$$

and thus obtain

$$\phi_{z,jk} \approx \frac{R_{z,jk}}{h_{z,k}} \,. \tag{6.20}$$

Contrariwise, in the special case $R^{\min}_{z,jk} = 0$, $R_{z,jk} \gg h_{z,j}, h_{z,k}$ we find

$$\begin{aligned}\phi_{z,jk} &= \Xi_{z,jk}(R_{z,jk}) - \Xi_{z,jk}\big(R^{\min}_{z,jk}\big) + \Xi_{z,jk}\big(-R^{\min}_{z,jk}\big) - \Xi_{z,jk}(-R_{z,jk}) \\ &= \Xi_{z,jk}(\infty) - \Xi_{z,jk}(-\infty) \\ &= 1 \,. \end{aligned} \tag{6.21}$$

Both cases are in agreement with the simplified expression

$$\phi_{z,jk} = \min\left\{1, \frac{R_{z,jk}}{h_{z,k}}\right\} \tag{6.22}$$

given in Ormel et al. (2010, Appendix C, Eq. (C6)).[1]

Evaluating the radial filling factor (Eq. (6.10)) is somewhat more involved. First, we argue that the annulus width of the swarm distribution can be assumed to be much smaller than the orbital radii of the particles, $h_{x,j} \ll r_j$, $h_{x,k} \ll r_k$. The support of the radial single-particle number density distribution $n^{(1)}_{r,j}(r)$ is therefore positive-valued, i.e. $r > 0 \,\forall r \in \operatorname{supp}\{n^{(1)}_{r,j}\}$. Therefore we may arbitrarily extend the lower integral limits:

$$\phi_{r,jk} \stackrel{!}{=} \int_{-\infty}^{\infty} \mathrm{d}r \, \frac{r}{r_j} n^{(1)}_{r,j}(r) \int_{-\infty}^{\infty} \mathrm{d}r' \, \frac{r'}{r_k} n^{(1)}_{r,k}(r') \kappa_{r,jk}(r' - r) \tag{6.23}$$

This will prove convenient when changing variables. Again invoking the narrow-annulus assumption, we also approximate $r/r_j \approx 1$, $r'/r_k \approx 1$:

$$\begin{aligned}\phi_{r,jk} &\approx \int \mathrm{d}r \, n^{(1)}_{r,j}(r) \int \mathrm{d}r' \, n^{(1)}_{r,k}(r') K_{r,jk}(r' - r) \\ &= \int \mathrm{d}b K_{r,jk}(b) \int \mathrm{d}r \, n^{(1)}_{r,j}(r) n^{(1)}_{r,k}(r + b) \\ &\equiv \int \mathrm{d}b K_{r,jk}(b) \xi_{r,jk}(b) \\ &\equiv \Xi_{r,jk}(R_{r,jk}) - \Xi_{r,jk}(R^{\min}_{r,jk}) + \Xi_{r,jk}(-R^{\min}_{r,jk}) - \Xi_{r,jk}(-R_{r,jk}) \end{aligned} \tag{6.24}$$

where the radial distance distribution $\xi_{r,jk}(b)$ and the cumulative radial distance distribution $\Xi_{r,jk}(b)$ were defined analogously to the vertical distance distributions. Using the same symmetry argument, we again impose without loss of generality that $h_{x,j} \leqslant h_{x,k}$. The piece-wise definitions of the radial distance distributions $\xi_{r,jk}(b)$ and $\Xi_{r,jk}(b)$ then read equivalent to the expressions derived for the vertical distance distributions (Eqs. (6.15) and (6.16)) but with $b_1 < b_2 \leqslant b_3 < b_4$ defined as

$$\begin{aligned} b_1 &\coloneqq r_k - r_j - h_{x,j} - h_{x,k} \,, \\ b_2 &\coloneqq r_k - r_j + h_{x,j} - h_{x,k} \,, \\ b_3 &\coloneqq r_k - r_j - h_{x,j} + h_{x,k} \,, \\ b_4 &\coloneqq r_k - r_j + h_{x,j} + h_{x,k} \,. \end{aligned} \tag{6.25}$$

[1] Note that said equation reads '$\phi_z = \min(h_{\mathrm{eff}}/R_z, 1)$' but should in fact read '$\phi_z = \min(R_z/h_{\mathrm{eff}}, 1)$', as is evident from the paragraph directly preceding it.

For comparison, we consider the special case $R^{\min}_{r,jk} = 0$, $R_{r,jk} \ll h_{x,j}, h_{x,k}$ and $r_j = r_k$, in which we obtain the approximate radial filling factor

$$\phi_{r,jk} \approx \frac{R_{r,jk}}{h_{x,k}} \ , \tag{6.26}$$

again in agreement with the result given in Ormel et al. (2010), Appendix C.

Although Eq. (6.22) gives a pragmatic approximation for the vertical filling factor $\phi_{z,jk}$, the radial filling factor $\phi_{z,jk}$ cannot be approximated in the same manner because we generally cannot assume that $r_j = r_k$. Therefore, when computing the interaction rate λ_{jk}, the radial filling factor must be evaluated using the full expression in Eq. (6.24) to correctly account for partial overlap of the volumes occupied by swarms j and k.

We now demonstrate how an estimate for the interaction timescale τ_{jk} can be given by comparing our geometrical model to the usual particle-in-a-box estimate of the local interaction rate for collisions. To this end, let us assume that the radial and vertical scale lengths of the two swarms j, k are identical, $h_{x,j} = h_{x,k} \equiv h_r$ and $h_{z,j} = h_{z_k} \equiv h_z$, and that the swarms have the same central orbital radius, $r_j = r_k \equiv r$. These assumptions imply that the two swarms occupy the same volume

$$V = (2\pi r)(2h_r)(2h_z) \ . \tag{6.27}$$

If we additionally impose that the minimal interaction radii are 0 and the maximal interaction radii are much smaller than the respective scale lengths, $R^{\min}_{r,jj} = 0$, $R_{r,jj} \ll h_{r,j}$, $R^{\min}_{z,jj} = 0$, and $R_{z,jj} \ll h_{z,j}$, we can approximate the filling factors according to Eqs. (6.20) and (6.26), obtaining

$$\lambda^{\rm coll}_{jk} = \frac{1}{8}(\tau_{jk})^{-1} N_j N_k \frac{R_{r,jk}}{h_r} \frac{R_{z,jk}}{h_z} \ . \tag{6.28}$$

This may now be compared with a particle-in-a-box estimate:

$$\lambda^{\rm coll,local}_{jk} = N_j N_k V^{-1} \sigma v_{\rm a} \ , \tag{6.29}$$

where $\sigma = \pi R_{r,jk} R_{z,jk}$ is the geometric interaction cross-section and $v_{\rm a}$ is the mutual approach velocity of the two particles. By imposing $\lambda^{\rm coll}_{jk} \sim \lambda^{\rm coll,local}_{jk}$ we thus find the interaction timescale to be

$$\tau^{\rm coll}_{jk} \sim \left[\frac{v_{\rm a}}{a}\right]^{-1} \ . \tag{6.30}$$

We point out that this simple estimate does not consider gravitational focussing. More realistic estimates are presented in the following section.

6.1.2 Nomenclature

In the following sections, we state the interaction radii and the interaction timescales for different types of interactions. We start by introducing a number of symbols, mostly adopting the nomenclature used in Ormel et al. (2010).

Each swarm j of particles is associated with a particle mass m_j; a mean orbital radius r_j; a planar rms velocity $v_j = r_j \langle e_j^2 \rangle^{1/2}$ and a vertical rms velocity $v_{z,j} = r_j \langle \sin^2 i_j \rangle^{1/2}$,

where $\langle e_j^2 \rangle^{1/2}$ and $\langle \sin^2 i_j \rangle^{1/2}$ are the rms eccentricities and rms inclinations, respectively; a particle bulk density ρ_j; and the number of particles in the swarm N_j. The *Hill radius* and the *Hill velocity* of a particle are defined to be

$$R_{\mathrm{h},j} \coloneqq r_j \left(\frac{m_j}{3M_*} \right)^{1/3} , \quad v_{\mathrm{h},j} \coloneqq \Omega_{\mathrm{K},j} R_{\mathrm{h},j} , \tag{6.31}$$

where M_* is the mass of the central object and $\Omega_{\mathrm{K},j} = (GM_*/r_j^3)^{1/2}$ is the average Keplerian angular velocity of particles in swarm j with the gravitational constant G. The average *solid radius* of assumedly spherical particles in swarm j is given as

$$R_j \coloneqq \left(\frac{4}{3} \pi \rho_j \right)^{-1/3} m_j^{1/3} . \tag{6.32}$$

When considering interactions between two particles from swarms j and k, we refer to the smaller and larger masses as m and M, respectively. We also define the velocities v_m, $v_{z,m}$ as the planar and vertical rms 'velocities of the lighter particle', though this applies only asymptotically in the technical definition we adopt:

$$v_m \coloneqq \mathrm{blend}(m_j, m_k; v_j, v_k) , \tag{6.33}$$

$$v_{z,m} \coloneqq \mathrm{blend}(m_j, m_k; v_{z,j}, v_{z,k}) , \tag{6.34}$$

which uses a form of linear interpolation,

$$\mathrm{blend}(a, b; x, y) \coloneqq [a + b]^{-1} (ax + by) . \tag{6.35}$$

This alternative definition has the advantage that it is commutative, yielding the same result under transposition of indices if $m_j = m_k$ but $v_j \neq v_k$ (and it is better suited for interval extension which is required for the bucketing scheme). Likewise, the mutual Hill radius R_{h} and mutual Hill velocity v_{h} are defined as Hill radius and velocity of the heavier particle.

In Ormel et al. (2010), the relative planar and vertical rms velocities are approximated as

$$w = \max \{v_j; v_k\} , \quad w_z = \max \{v_{z,j}; v_{z,k}\} . \tag{6.36}$$

In our generalised model, we also want to take particle drift velocities into account. In the presence of a gas disk, particles are subject to a systematic radial and azimuthal drift induced by the gas drag forces. The drift velocities of a particle depend on its Stokes number; for interactions between different particle species, a systematic difference between drift velocities must therefore also be taken into account. We therefore approximate the planar impact velocity as

$$w \coloneqq \sqrt{\max \{v_j^2; v_k^2\} + (v_{r,j} - v_{r,k})^2 + (v_{\varphi,j} - v_{\varphi,k})^2} . \tag{6.37}$$

An equation for the drift velocity $\mathbf{u}$ in a sheared rotating coordinate system was given in Eq. (2.60); we further discuss its general solution in Sect. 6.3.1. At the origin of the sheared rotating coordinate system, $\mathbf{x} = 0$, the radial and azimuthal drift velocities v_r,

v_φ relate to $\mathbf{u}$ as $v_r = u_x$ and $v_\varphi = v_\mathrm{K} + u_y$. The contribution of the settling speed to the vertical impact velocity w_z is neglected in this model.

The *escape velocity* for a particle in swarm j is

$$v_{\mathrm{esc},j}^2 := \frac{2Gm_j}{R_j} \,. \tag{6.38}$$

With the combined bulk radius of two particles

$$R_\mathrm{s} := R_j + R_k \,, \tag{6.39}$$

the mutual escape velocity of two particles is

$$v_\mathrm{esc}^2 := \frac{2G(m_j + m_k)}{R_\mathrm{s}} \tag{6.40}$$

and can be approximated as

$$v_\mathrm{esc}^2 \approx \max\left\{v_{\mathrm{esc},j}^2; v_{\mathrm{esc},k}^2\right\} \tag{6.41}$$

where we emphasise that the approximation is not only asymptotically correct but also exact for the important special case $m_j = m_k$, $R_j = R_k$.

We also define the abbreviations

$$h_{x,\max} := \max\left\{h_{x,j}; h_{x,k}\right\} \,, \quad h_{z,\max} := \max\left\{h_{z,j}; h_{z,k}\right\} \,. \tag{6.42}$$

6.1.3 Viscous stirring

In the case of viscous stirring, we distinguish between the *superescape regime* ($v_\mathrm{esc} \leqslant w$), the *dispersion-dominated (d.d.) regime* ($2.5\,v_\mathrm{h} \leqslant w \leqslant v_\mathrm{esc}$), and the *shear-dominated (s.d.) regime* ($w \leqslant 2.5\,v_\mathrm{h}$). No stirring takes place in the superescape regime. For the other regimes, the interaction radii are defined to be

$$R_{r,jk}^{\mathrm{vs,min}} = 0 \,, \tag{6.43}$$

$$R_{r,jk}^\mathrm{vs} = \begin{cases} R^{\mathrm{vs},0} & \text{in s.d. regime, or in d.d. regime for } R^{\mathrm{vs},0} < w/\Omega_\mathrm{K} \\ R^{\mathrm{vs,d}} & \text{in d.d. regime for } R^{\mathrm{vs},0} \geqslant w/\Omega_\mathrm{K} \,, \end{cases} \tag{6.44}$$

$$R_{z,jk}^{\mathrm{vs,min}} = 0 \,, \tag{6.45}$$

$$R_{z,jk} = R^{\mathrm{vs},0} \,, \tag{6.46}$$

where

$$R^{\mathrm{vs},0} := \begin{cases} R_\mathrm{s}\dfrac{v_\mathrm{esc}^2}{v_m w} & \text{in d.d. regime} \\ 2.5R_\mathrm{h} & \text{in s.d. regime,} \end{cases}$$

$$R^{\mathrm{vs,d}} := R_\mathrm{h}\sqrt{\frac{6v_\mathrm{h}}{v_m}} \,. \tag{6.47}$$

In to Ormel et al. (2010), Appendix B.2.3, the change in velocity due to viscous stirring is given as

$$\left(\Delta v^2\right)_{jk} := \left(\frac{m_j}{m_j + m_k}\right)^2 (\Delta v^{\mathrm{vs},0})^2 \tag{6.48}$$

where

$$\Delta v^{\mathrm{vs},0} := \begin{cases} v_m & \text{in d.d. regime} \\ 2.5 v_{\mathrm{h}} & \text{in s.d. regime.} \end{cases} \tag{6.49}$$

By comparison with Ormel et al. (2010) (B24)[2], we infer an interaction timescale of

$$\tau_{jk}^{\mathrm{vs}} = \left[\frac{2h_{z,\max}}{a}\Omega_{\mathrm{K}} \log\Lambda\right]^{-1} \tag{6.50}$$

with a 'Coulomb factor' $\Lambda = \exp(1) + h_{x,\max}/R_{r,jk}$. We also take the changes in rms eccentricities and rms inclinations to be

$$\begin{pmatrix} (\Delta e^2)_{jk}^{\mathrm{vs}} \\ (\Delta \sin^2 i)_{jk}^{\mathrm{vs}} \end{pmatrix} = \begin{pmatrix} f^{\mathrm{vs}}(\beta) \\ f_z^{\mathrm{vs}}(\beta) \end{pmatrix} \left(\frac{m_j}{m_j + m_k}\right)^2 (v^{\mathrm{vs}})^2 \,, \tag{6.51}$$

where $\beta = v_{z,j}/v_j$, and where the calibration factors $f^{\mathrm{vs}}(\beta)$, $f_z^{\mathrm{vs}}(\beta)$ are given by

$$\begin{pmatrix} f^{\mathrm{vs}}(\beta) \\ f_z^{\mathrm{vs}}(\beta) \end{pmatrix} \approx \frac{8}{\pi^2} \begin{pmatrix} I_P^{\mathrm{vs}}(\beta) \\ I_Q^{\mathrm{vs}}(\beta) \end{pmatrix} \qquad \text{in d.d. regime} \tag{6.52}$$

$$\begin{pmatrix} f^{\mathrm{vs}}(\beta) \\ f_z^{\mathrm{vs}}(\beta) \end{pmatrix} \approx \begin{pmatrix} 3.0 \\ 1.1(h_{z,\max}/R^{\mathrm{vs}})^2 \end{pmatrix} \qquad \text{in s.d. regime} \tag{6.53}$$

using functions $I_P^{\mathrm{vs}}(\beta)$, $I_Q^{\mathrm{vs}}(\beta)$ defined in Ohtsuki et al. (2002) and approximated by Ormel et al. (2010) as

$$I_P^{\mathrm{vs}}(\beta) \approx \quad 2.439 - 8.242\exp(-3.396\beta) \,, \tag{6.54}$$

$$I_Q^{\mathrm{vs}}(\beta) \approx -0.459 + 3.807\exp(-2.931\beta) \,. \tag{6.55}$$

6.1.4 Dynamical friction

With regard to dynamical friction, the same distinction between the *superescape regime* ($v_{\mathrm{esc}} \leqslant w$), the *dispersion-dominated (d.d.) regime* ($2.5\, v_{\mathrm{h}} \leqslant w \leqslant v_{\mathrm{esc}}$), and the *shear-dominated (s.d.) regime* ($w \leqslant 2.5\, v_{\mathrm{h}}$) is made. Dynamical friction also does not operate in the superescape regime. For the other regimes, the interaction radii are defined to be

$$R_{r,jk}^{\mathrm{df,min}} = 0 \,, \quad R_{r,jk} = R^{\mathrm{df},0} \tag{6.56}$$

$$R_{z,jk}^{\mathrm{df,min}} = 0 \,, \quad R_{z,jk} = R^{\mathrm{df},0} \,, \tag{6.57}$$

where

$$R^{\mathrm{df},0} := \begin{cases} R_{\mathrm{s}} \dfrac{v_{\mathrm{esc}}^2}{w^2} & \text{in d.d. regime} \\ 2.5 R_{\mathrm{h}} & \text{in s.d. regime,} \end{cases} \tag{6.58}$$

[2]Note that Ormel et al. (2010), Eq. (B24) is missing a factor of Ω_{K}.

and where the interaction timescale τ_{jk}^{df} is given by

$$\left(\tau_{jk}^{\mathrm{df}}\right)^{-1} = \frac{v_{\mathrm{a}}}{a} \,, \tag{6.59}$$

that is, no calibration factors are applied. The approach velocity v_{a} is given by

$$v_{\mathrm{a}} = \begin{cases} 2h_{x,\max}\,\Omega_{\mathrm{K}} & \text{in d.d. regime} \\ 3.2v_{\mathrm{h}} & \text{in s.d. regime.} \end{cases} \tag{6.60}$$

Following Ormel et al. (2010), (B12a,b), we take dynamical friction to be an elastic collision and, by averaging over head-on and tail-on collisions, obtain the average velocity changes

$$\begin{aligned} \Delta(v_M^2)^{\mathrm{df}} &= -\frac{4m}{(M+m)^2}\left(Mv_M^2 - mv_m^2\right) \\ \Delta(v_m^2)^{\mathrm{df}} &= \frac{4M}{(M+m)^2}\left(Mv_M^2 - mv_m^2\right) \,, \end{aligned} \tag{6.61}$$

which amount to a change in rms eccentricities and rms inclinations as

$$\begin{pmatrix} (\Delta e^2)_{jk}^{\mathrm{df}} \\ (\Delta \sin^2 i)_{jk}^{\mathrm{df}} \end{pmatrix} = \frac{4m_k}{(m_j+m_k)^2}\left[-m_j\begin{pmatrix} \left\langle e_j^2 \right\rangle \\ \left\langle \sin^2 i_j \right\rangle \end{pmatrix} + m_k \begin{pmatrix} \left\langle e_k^2 \right\rangle \\ \left\langle \sin^2 i_k \right\rangle \end{pmatrix}\right] \,. \tag{6.62}$$

6.1.5 Collisions

The dynamics of collision are slightly different depending on the velocity regime. We distinguish between the *superescape regime*, $v_{\mathrm{esc}} \leqslant w$; the *dispersion-dominated regime*, $2.5\,v_{\mathrm{h}} \leqslant w \leqslant v_{\mathrm{esc}}$; and the *Hill regime*, $w \leqslant v_{\mathrm{h}}$.[3]

By comparing with Ormel et al. (2010), (B4), we find an interaction timescale τ_{jk}^{vs} to be given by

$$\left(\tau_{jk}^{\mathrm{vs}}\right)^{-1} = \begin{cases} \dfrac{2h_{z,\max}}{a}\Omega_{\mathrm{K}}, & \text{in superescape or d.d. regime} \\ f_{\mathrm{hit}}\,\dfrac{3.2v_h}{a}, & \text{in Hill regime,} \end{cases} \tag{6.63}$$

where the probability that a projectile that has entered the Hill sphere of the target actually collides with it is approximated by

$$f_{\mathrm{hit}} \approx A_3 \times \frac{b_{\mathrm{col}}}{R_{\mathrm{h}}} \times \min\left\{1; \frac{b_{\mathrm{col}}}{h_{z,\max}}\right\} \,, \tag{6.64}$$

$$b_{\mathrm{col}} := R_{\mathrm{s}}\frac{v_{\mathrm{esc}}}{2.5v_{\mathrm{h}}} \approx \sqrt{R_{\mathrm{s}}R_{\mathrm{h}}} \tag{6.65}$$

[3] It is unclear from Ormel et al. (2010) whether the omission of the range $v_{\mathrm{h}} \leqslant w \leqslant 2.5v_{\mathrm{h}}$ is intentional.

as suggested in Ormel et al. (2010), Eq. (B9). $A_3 \approx 2.9$ is the 3-body calibration factor.[4] The interaction radii for collisions are

$$R_{r,jk}^{\mathrm{coll,min}} = \begin{cases} 0 & \text{in superescape or d.d. regime} \\ 1.7R_{\mathrm{h}} & \text{in s.d. regime,} \end{cases} \tag{6.66}$$

$$R_{r,jk}^{\mathrm{coll}} = R^{\mathrm{coll}} \,, \tag{6.67}$$

$$R_{z,jk}^{\mathrm{coll,min}} = 0 \,, \tag{6.68}$$

$$R_{z,jk}^{\mathrm{coll}} = R^{\mathrm{coll}} \,, \tag{6.69}$$

where

$$R^{\mathrm{coll}} = \begin{cases} R_{\mathrm{s}}\sqrt{A_1 + A_2\dfrac{v_{\mathrm{esc}}^2}{w^2}} & \text{in superescape or d.d. regime} \\ 2.5R_{\mathrm{h}} & \text{in Hill regime} \end{cases} \tag{6.70}$$

with the calibration factors $A_1 = 0.90$, $A_2 = 1.5$. Unlike Ormel et al. (2010), we follow the prescription in Greenzweig and Lissauer (1992, Eq. (17)) and use the gravitational focussing factor (Eq. (6.70)) instead of explicitly distinguishing between the superescape regime and the dispersion-dominated regime.

If the collision leads to coagulation, we infer from local conservation of momentum that the new velocity of the coagulated particle will be

$$\mathbf{v}_j' = \frac{m_j\mathbf{v}_j + m_k\mathbf{v}_k}{m_j + m_k} \,, \tag{6.71}$$

and hence obtain a change of rms eccentricities and rms inclinations

$$\begin{pmatrix} (\Delta e^2)_{jk}^{\mathrm{vs}} \\ (\Delta \sin^2 i)_{jk}^{\mathrm{vs}} \end{pmatrix} = \frac{m_k}{(m_j + m_k)^2}\left[-(2m_j + m_k)\begin{pmatrix} \left\langle e_j^2 \right\rangle \\ \left\langle \sin^2 i_j \right\rangle \end{pmatrix} + m_k \begin{pmatrix} \left\langle e_k^2 \right\rangle \\ \left\langle \sin^2 i_k \right\rangle \end{pmatrix}\right] , \tag{6.72}$$

equivalent to the velocity change for inelastic collisions given in Ormel et al. (2010, Eqs. (B14a,b)).[5]

In a realistic collision model, not all colliding bodies will hit and stick. Depending on the impact velocity, the porosity, and the masses of the two bodies, partial fragmentation, or cratering, is a likely outcome. Unlike for coagulation, we cannot simply infer the velocities of the fragments because only the total momentum is conserved. Lacking a more sophisticated model for the distribution of fragment velocities, we therefore make the simplest possible assumption that the rms velocities of all the bodies resulting from a collision matches the updated rms velocities as per Eqs. (6.72).

[4] Note that, in Ormel et al. (2010), the calibration factor A_3 erroneously appears in both Eqs. (B8) and (B9), whereas it presumably should appear only in Eq. (B9).

[5] Note that Eq. (B14b) of Ormel et al. (2010) mentions v when it supposedly means to refer to v_m.

6.1.6 Differences from the original model

In this section we have formally introduced Ormel's geometrical interaction model for stochastic collision, viscous stirring, and dynamical friction processes. Notably, we deviate from the model of Ormel et al. (2010) and its implementation in a few regards.

First, we simulate viscous stirring and dynamical friction events as Monte Carlo events, which has become feasible thanks to the increased efficiency of the bucketing scheme. However, as mentioned in Sect. 4.6.3.2, the mutual changes in rms velocities inflicted by the close encounter of two particles are often minuscule, we use the boosting technique not only for collisions but also for viscous stirring and dynamical friction.

Another notable discrepancy is that we do not set up a grid of 'zones', which is how the radial resolution limit is implemented in Ormel et al. (2010) (where the zone width Δa corresponds to $R_{\min}$ in our formalism). Instead, we have modified the definition of the scale length in Eq. (6.1) to enforce the resolution limits, but only for stochastic particles, not for self-representing bodies. The multi-zone setup of Ormel et al. (2010) not only introduces a resolution limit but presumably also helps in speeding up the interaction rate updating for the representative bodies. For the bucketing scheme, a comparable computational advantage is attained by means of sub-bucketing as discussed in Sect. 4.5. Although radial sub-bucketing bears some similarities to radial zones, it is a detail of the computational scheme and thus, unlike the multi-zone approach, is imperceptible by the physical model being simulated.

We also differ in our definition of the planar impact velocity in Eq. (6.37), which takes into account systematic velocity differences caused by different drift speeds, and we also adopt a different definition of the 'velocities of the lighter particle' v_m and $v_{z,m}$ (Eqs. (6.33–6.34)). We do not know if our reconstruction of the geometric filling factors in Sect. 6.1.1 are exactly equivalent to the geometric filling factors used by Ormel et al. (2010) because the authors have not provided an exact definition. More slight discrepancies may exist, but we believe that our adaptation is otherwise very close to the original model.

6.2 Fragmentation and the RPMC method

The treatment of fragmentation in the RPMC method was first discussed in Sect. 3.4.4.3, where a rudimentary approach for including a fragmentation model had been proposed. Citing Bukhari Syed et al. (2017), we had argued that the mass of the largest remaining fragment continuously decreases with increasing energy of the kinetic impact. This relation blends well with the simple collision model of Ormel et al. (2010), which was previously referenced in Sect. 4.6.3.1 of Chapter 4. In Ormel's model, a certain fraction $f_{\rm frag}$ of the combined mass of the colliding particles is converted to fragments, which are assumed to be of uniform size $R_{\rm frag}$ for the sake of simplicity. Because the fragment size is usually assumed small enough to be effectively gas-bound by means of very short stopping times, rms velocities are irrelevant for fragments. The fraction $f_{\rm frag}$ is a function of the mass of the colliding bodies j and k and of the impact velocity $v_{\rm a}$,

$$f_{\rm frag} = \frac{\epsilon E_{\rm col}}{\frac{1}{2}\left(m_j + m_k\right) v_{\rm esc}^2} \,, \tag{6.73}$$

with the kinetic energy of the impact given by

$$E_{\mathrm{col}} = \frac{1}{2}\frac{m_j m_k}{m_j + m_k} v_{\mathrm{a}}^2 \,. \tag{6.74}$$

The *effective coefficient of restitution* ϵ is a free parameter that can be chosen to account for the inelastic collision characteristics of loosely bound ('rubble pile') material.

Being agnostic to the individual characteristics of the sampled particles, the original RPMC method with its equal-weight sampling adapts well to collision processes that include fragmentation. In accordance with the 'atomistic' picture, the properties of the representative particle are chosen anew from the weighted distribution of particle properties after every collision (Eq. (3.30)). Combining boosting with a fragmentation model is possible as well, and the procedure suggested in Sect. 3.4.4.3 may serve as an approximation. However, there are two intricacies worthy of further consideration.

Firstly, a fragmentation model comprising a fractional loss of mass, as embodied by Eq. (6.73), is not fully done justice with the simple linear mass transfer model represented by Eqs. (3.58,3.59). With this model, the resulting mass of the largest cohesive body after β collisions would be

$$m_{\mathrm{co}} = (1 - \beta f_{\mathrm{frag}})\, m_j + (1 - f_{\mathrm{frag}})\, \beta\, m_k \,, \tag{6.75}$$

where we abbreviated the boost factor as $\beta \equiv \beta_{jk}$. In other words, the combined mass of the β impactors is added at once before the fraction of fragmented mass is subtracted from it. This, however, neglects the 'compound interest' effect: according to the collision model, fragments are produced at every impact, and thus the contribution of the impactors to the total mass of fragments must scale approximately quadratically with β.

The second difficulty is more severe and pertains the generalisation of the RPMC method developed in Chapter 3. Interactions in the few-particles regime may involve individually resolved bodies; if a part of their mass is converted to fragments, we would expect to retain them as individual bodies. But the mass of the fragments must also be represented to ensure that overall mass is conserved. At the same time, adding a representative particle to the simulation every time a collision produces fragments would cause the number of representative particles n to grow excessively as soon there is the slightest chance of fragmentation.

To address both problems, in this section we outline a more systematic means of incorporating a fragmentation model in a RPMC simulation.

6.2.1 Boosted fragmentation

For a slightly more general fragmentation model, let us assume that, given two particles j, k undergoing a collision, we can compute the total mass fraction f_{frag} that is converted to fragments, while the remainder of the combined mass $(m_j + m_k)$ remains in one cohesive body. f_{frag} is assumed to be relatively insensitive to small changes to the mass m_j, implying that it can be assumed constant throughout a series of collision events grouped together for the purpose of boosting. We impose no particular distribution of mass fragments $n_{jk}^{\mathrm{coll}}(\mathbf{q})$.

Let a representative particle j now undergo a sequence of collision events with particles from swarm k. Let $(m_j^{(n)})_n$ denote the ensuing sequence of masses of the cohesive body, with $m_j^{(0)} \equiv m_j$ being the initial mass. Our fragmentation model then implies the iterative relation

$$m_j^{(n+1)} = f_{\mathrm{co}} \left(m_j^{(n)} + m_k \right) \ , \tag{6.76}$$

where $f_{\mathrm{co}} \coloneqq (1 - f_{\mathrm{frag}})$ is the total mass fraction remaining in one cohesive body in each collision. The iterative relation can be collapsed to a non-iterative expression $m_j^{(\beta)}$ by using a geometric series,

$$m_{\mathrm{co}} \coloneqq m_j^{(\beta)} = (f_{\mathrm{co}})^\beta m_j + f_{\mathrm{co}} \frac{(f_{\mathrm{co}})^\beta - 1}{f_{\mathrm{co}} - 1} m_k \ . \tag{6.77}$$

In the context of boosting, we can always assume that $f_{\mathrm{frag}} \ll 1$ because, if the total mass fraction converted to fragments in a single collision was significant, no boosting would be needed in the first place, and above mass relation would not be exercised beyond $\beta = 1$. But if $f_{\mathrm{frag}} \ll 1$, f_{co} will be numerically ill-conditioned when using a conventional floating-point representation. To illustrate, let $f_{\mathrm{frag}} \equiv A \times 2^B$ be represented with a normalised mantissa A, $\frac{1}{2} < A \leqslant 1$, and an exponent B in a base-2 floating-point format. Representing f_{co} in the same normalised manner, $f_{\mathrm{co}} \equiv C \times 2^D$, we find that $C = (1 - A \times 2^B)$ and $D = 0$ because $f_{\mathrm{co}} \sim 1$. Therefore, the entire dynamic range captured by the exponent of f_{frag} must be accommodated in the mantissa of f_{co}. If B is too small, the subtraction in $C = (1 - A \times 2^B)$ will underflow, yielding $C = 1$. As a consequence, Eq. (6.77) is not numerically viable. We therefore approximate the expression with a series expansion to first order in f_{frag}:

$$m_{\mathrm{co}} \approx (1 - \beta\, f_{\mathrm{frag}})\, m_j + \left(1 - \frac{\beta + 1}{2} f_{\mathrm{frag}} \right) \beta\, m_k \ , \tag{6.78}$$

finding a significant difference from Eq. (6.75).

6.2.2 Fragmentation and the boost factor

The initial estimate for the boost factor was given in Sect. 3.4.4, Eq. (3.46) as

$$\beta^0 \coloneqq q_m \frac{m_j}{m_k} \tag{6.79}$$

and later refined for fragmentation in Eq. (3.57). Regarding this refined estimate we point out that, in the case of bouncing, $m_{\mathrm{co}} = m_j$, the transferred mass δm is 0, rendering the estimate ill-defined. To avoid this problem, we instead propose that the boost factor be chosen such that the total mass transferred from swarm k to representative particle j does not exceed the mass accumulation threshold, $\beta m_k \lesssim q_m m_j$. The original estimate (Eqs. (3.46) and (6.79)) therefore remains a suitable estimate even when considering fragmentation. However, for the first-order approximation in Eq. (6.78) to remain sufficiently accurate, we additionally want to ensure that $\beta\, f_{\mathrm{frag}} \lesssim q_m$, and we thus impose the constraint

$$\beta \leqslant \frac{q_m}{f_{\mathrm{frag}}} \ . \tag{6.80}$$

In Sect. 3.4.4.2, we had introduced constraints (Eqs. (3.48) and (3.50)) for the boost factor β designed to favour an outcome where the swarm particle count made a precision landing on $N_j' = N_\mathrm{th}$ if admitted by the other constraints. This was intended to ease the splitting up of swarms into individual self-representing particles at the particle regime threshold N_th. However, these constraints are detrimental for outcomes other than hit-and-stick collisions. First, they are not easy to adapt to our generalised fragmentation model. Eqs. (3.48) and (3.50) were derived by inverting the function $m_\mathrm{co}(\beta)$ for β, which worked reasonably well without fragmentation where $m_\mathrm{co}(\beta) = m_j + \beta m_k$. However, even with the first-order approximation of Eq. (6.78), the dependence of $m_\mathrm{co}(\beta)$ on β is quadratic, and its inverse would again need to be approximated to avoid subsequent numerical issues. For simplicity, we therefore abandon the two constraints for particles that undergo partial fragmentation, allowing an average relative deviation from the individual particle mass by a factor of $q_m/2$ upon swarm split-up.

6.2.3 Fragmentation in the few-particles regime

Although fragmentation naturally fits in the framework of the original RPMC method, it is unclear how to proceed if partial fragmentation occurs in the few-particles regime, that is, in an interaction between a few-particles swarm and a many-particles swarm or in an interaction between two few-particles swarms, both entailing transfer of mass between swarms. Our choice is determined by three requirements: the total mass shall always be conserved; the part of individually represented bodies that is not converted to fragments shall remain represented individually; and the number of representative particles shall not grow excessively.

As far as coagulation was concerned, the extended RPMC method had been formulated such that, even though it was designed to split up swarms when their particle count dropped below N_th, it could still operate correctly without the split-up if the individual representation of bodies in few-particle swarms was not deemed necessary. For runaway growth, individual representation is necessary, but this may not be the case for other applications; as an example, the few-particles regime of the RPMC method without the splitting bears some similarity to the super-droplet method of Shima et al. (2009) originally developed for atmospheric cloud microphysics. However, in our treatment of coagulation and fragmentation, allowing few-particles swarms k with $N_k > 1$, besides being unfavourable for any physical scenario with runaway growth characteristics, would make it difficult to avoid adding new representative particles in collisions between few-particles swarms. Therefore, in the following treatment we impose that swarms k whose swarm particle count falls below N_th be always split up into N_th individual self-representing particles. For a few-particles swarm k, we can thus assume $N_k = 1$ henceforth.

In an interaction between a representative particle j with $N_j > N_\mathrm{th}$ with a many-particles swarm k, only the properties of representative particle j can change. Therefore, if the collision between particle j with a particle from swarm k happens to produce fragments, the new mass of representative particle j will be sampled from the fragment mass distribution as per Eq. (3.30).

Let us now consider the interaction of two self-representing particles j and k. In this case, no boosting occurs, $\beta = 1$, and hence Eq. (6.78) degenerates to

$$m_{\mathrm{co}} = (1 - f_{\mathrm{frag}})\,(m_j + m_k) \ . \tag{6.81}$$

Representative particle j will represent the combined mass. If $f_{\mathrm{frag}} = 0$, the two particles coagulate fully; in this case, representative particle k may be removed from the simulation. If $f_{\mathrm{frag}} > 0$, the new mass of representative particle k will be sampled from the weighted distribution of fragment masses. Because fragments tend to be light, the number of particles in the new swarm N_k' will probably be larger than N_{th}, rendering swarm k a many-particles swarm. Only in the unlikely event that $N_k' \leqslant N_{\mathrm{th}}$ does swarm k need to be split up into individual self-representing particles.

Next, we consider the interaction of a self-representing particle j with a particle from many-particles swarm k. The interaction produces a cohesive body of mass m_{co} and a total mass of fragments

$$M_{\mathrm{frag}} = m_j + \beta m_k - m_{\mathrm{co}} \ , \tag{6.82}$$

leaving behind a swarm of particles of mass m_k with a total mass of

$$M_{\mathrm{rem}} = M_k - \beta m_k \ . \tag{6.83}$$

Here, representative particle j is chosen to represent the cohesive mass m_{co}. A new representative particle k is then sampled from the totality of the $(N_k - \beta)$ old swarm particles and the distribution of fragments. The probability that k keeps representing the remains of the original swarm are

$$P = \frac{M_{\mathrm{rem}}}{M_{\mathrm{rem}} + M_{\mathrm{frag}}} \ . \tag{6.84}$$

The procedure laid out here has proven adequate for simulating collision processes that exhibit partial fragmentation. A test run with Ormel's model with and without partial fragmentation has been conducted in Sect. 4.6.3.3, both running with comparable efficiency (cf. Fig. 4.15).

6.3 Gas drag model

Particles in a protoplanetary disk are subject to gas drag forces that depend on their gas-relative velocity. In the simple axisymmetric gas model introduced in Chapter 1, the gas is assumed to orbit the star with a tangential velocity that differs from the velocity of an undisturbed test body on a circular orbit. The gas drag force alters the orbital positions, the (rms) eccentricities, and the (rms) inclinations of representative particles, with particularly dramatic effect for pebble-sized objects as discussed in Sect. 2.2.7. In this section, we describe how the gas drag force can be taken into account for particles of different masses.

6.3.1 Estimating the Stokes number

For a given object on a circular orbit with semimajor axis a and Stokes number St, an expression for the gas-relative velocity in a co-rotating frame was given in Eq. (2.61). This is sufficient if the drag timescale, represented by the Stokes number St, does not depend on the drift velocity, as would be the case in the Epstein regime. However, if the particle operates in one of the Stokes regimes, the drag timescale depends on the frame-relative gas velocity $\mathbf{v}_{\mathrm{gas}}$, and Eq. (2.60) becomes a non-linear system of equations with no obvious analytical solution. This difficulty was first acknowledged in Sect. 4 of Weidenschilling (1977a), who applied some algebraic transformations, established a quadratic relation between u_x and u_y, and then proposed a method to solve the transformed set of equations numerically. However, we found it much simpler to interpret Eq. (2.61) as a prescription of an iterative method,

$$\begin{aligned}
\mathrm{St}^{(0)} &:= 1 \,, \\
\mathbf{u}^{(n+1)} &:= \frac{1}{1 + \left(\mathrm{St}^{(n)}\right)^2} \begin{pmatrix} 1 & 2\,\mathrm{St}^{(n)} \\ -\frac{1}{2}\,\mathrm{St}^{(n)} & 1 \end{pmatrix} \mathbf{v}_{\mathrm{gas}} \,, \\
\mathrm{St}^{(n+1)} &:= \mathrm{St}\Big(\big|\mathbf{v}_{\mathrm{gas}} - \mathbf{u}^{(n+1)}\big|\Big) \,,
\end{aligned} \tag{6.85}$$

where $\mathrm{St}(\Delta v)$ denotes the Stokes number of a particle with gas-relative velocity Δv as per Eqs. (2.51–2.55). Although we did not attempt to prove the convergence of this method formally, it was found to converge to sufficient accuracy within three or four iterations for all parameter combinations relevant to our purpose. We use this method to estimate the Stokes numbers of the particles in the simulation, neglecting eccentricity and inclination for this purpose.

6.3.2 Gas drag regimes

To account for gas drag, particles simulated with N-body calculations may be directly subjected to the gas drag force $\mathbf{F}_{\mathrm{drag}}$ defined in Sect. 2.2.6. However, for orbits with moderate eccentricities and inclinations it would also be sufficient to impose equilibrium conditions and to impose the damping relations of Inaba et al. (2001) which were introduced in Sect. 2.2.7. These damping relations are also applicable for representative particles for which only rms eccentricity and rms inclination are tracked. However, the differential equation in Eq. (2.63) is not applicable for $\mathrm{St} \ll 1$, as can be seen by computing the limit for $\mathrm{St} \to 0$. Noting that the damping process of (rms) eccentricities and (rms) inclinations can be quantified by damping timescales,

$$t_e := e \left| \frac{\mathrm{d}e}{\mathrm{d}t} \right|^{-1} , \qquad t_{\langle e^2 \rangle} := \langle e^2 \rangle \left| \frac{\mathrm{d}\langle e^2 \rangle}{\mathrm{d}t} \right|^{-1} , \tag{6.86}$$

$$t_i := \sin i \left| \frac{\mathrm{d}\sin i}{\mathrm{d}t} \right|^{-1} , \quad t_{\langle i^2 \rangle} := \langle \sin^2 i \rangle \left| \frac{\mathrm{d}\langle \sin^2 i \rangle}{\mathrm{d}t} \right|^{-1} , \tag{6.87}$$

we argue that, when studying secular disk dynamics, quantities damped on timescales shorter than the timescale of an orbital revolution can be neglected. By equating above damping timescales with Ω_{K}^{-1}, we thus find that (rms) eccentricities and (rms) inclinations are negligible for bodies with $\mathrm{St} \lesssim 1$. Therefore, for bodies with $\mathrm{St} \lesssim 1$, we

explicitly force $e \leftarrow 0$ and $i \leftarrow 0$, or $\langle e^2 \rangle \leftarrow 0$ and $\langle \sin^2 i \rangle \leftarrow 0$, respectively. Without eccentricity and inclination, the radial drift of the body may be simulated with the simpler relation of Eq. (2.62) that, unlike Eq. (2.63), is applicable for all values of St.

Turbulent stirring, governed by Eq. (2.69), is a consequence of gas density fluctuations and operates at all mass scales. Mathematically, this stirring relation effectuates non-zero rms eccentricities for all bodies, which seems to be in conflict with our pragmatic choice of forcing $\langle e^2 \rangle \leftarrow 0$. It can be argued that secular orbital parameters such as eccentricity and inclination become meaningless for particles with stopping times shorter than their orbital period. Nevertheless, turbulent stirring remains a relevant influence for light particles with short stopping times. But rather than generating an (rms) eccentricity, it leads to a radial and vertical 'spread' in the distribution of dust particles. The damping induced by gas drag counteracts the turbulent stirring and effectuates a vertical settling of particles. By equating the timescales for the settling and stirring processes, Birnstiel et al. (2010) derived the *dust scale height*

$$h_\mathrm{d} = H_p \cdot \min\left\{1, \sqrt{\frac{\alpha}{\min\{\mathrm{St}, 1/2\} \cdot \left(1 + \mathrm{St}^2\right)}}\right\} \,. \tag{6.88}$$

For particles with $\mathrm{St} \lesssim 1$, the scale height h_z is thus determined by the dust scale height h_d, not by the relation given in Eq. (6.2).

6.4 A hybrid of deterministic and stochastic models

In many gravitational N-body codes, a particle may operate in one of two modes. An *active* particle exerts gravitational force on all other particles, whereas a *passive* particle does not. Passive particles 'feel' the gravitational pull of active particles, but they do not themselves pull other particles, effectively operating as massless test particles. A passive particle also does not participate in collisions. Although kinetically equivalent to a massless particle as far as the N-body calculations are concerned, a passive particle need not be massless as long as its gravitational influence is accounted for by means other than direct N-body calculations. Because only active particles exert gravitational forces through the N-body code, an implementation may choose to manage active and passive particles separately, thus allowing for a more efficient implementation. Ideally, the computational cost per N-body timestep then scales with $n \cdot n_\mathrm{active}$, where n is the total number of N-body (active and passive) particles in the simulation, and n_active is the number of active particles.

This distinction between active and passive particles naturally enables the embedding of a stochastic representative particle method. In such a design, the representative particle of a many-particles swarm also participates in the N-body simulation as a passive particle, while individual self-representing bodies in the stochastic simulation are identified with active N-body particles. Thereby, all particles can undergo radial redistribution through gravitational scattering, and gap opening by protoplanets can be modelled correctly. Any gravitational encounters and collisions with the swarms of (passive) representative particles are mediated by stochastic means, using geometrical surrogate models such as Ormel's model of collisions, viscous stirring and dynamical friction presented in Sect. 6.1. Assuming that only a small number of oligarchs emerge from run-

effectuating / affected	gas	dust $\mathrm{St}_j \lesssim \alpha$	pebble $\mathrm{St}_j \sim 1$ …	planetesimal $\mathrm{St}_j \gg 1$ $N_j \gg 1$	embryo … $N_j = 1$	planet …
gas	gas model					gap model
dust	coupled					
pebble	drift					
planetesimal	drift + damping; turbulence		viscous stirring & dynamical friction		N-body	
embryo						
planet						

Table 6.1: Overview of kinetic interactions between different constituents of the protoplanetary disk.
A static axisymmetric gas model was proposed in Sect. 2.2 and augmented with a simple planetary gap model in Sect. 2.2.5. 'Drift' refers to the radial drift according to the simple model in Eq. (2.62), whereas 'drift + damping' designates the combined drift and damping model in Eqs. (2.63–2.67). 'Turbulence' refers to the turbulent stirring model in Eq. (2.69). 'N-body' implies that the corresponding interactions are mediated through direct N-body forces. Viscous stirring and dynamical friction as per Ormel's model are described in Sect. 6.1.

effectuating / affected	gas	dust $\mathrm{St}_j \lesssim \alpha$	pebble $\mathrm{St}_j \sim 1$ …	planetesimal $\mathrm{St}_j \gg 1$ $N_j \gg 1$	embryo … $N_j = 1$	planet …
gas						
dust						
pebble			SI; stochastic			
planetesimal				stochastic		
embryo					N-body	
planet						

Table 6.2: Overview of collisional and mass-accreting interactions between different constituents of the protoplanetary disk.
'SI' refers to an effective model for generating planetesimals from gravitational collapse as mentioned in Sect. 6.4. Stochastic collisions are handled by Ormel's model described in Sect. 6.1.

away growth, $n_{\mathrm{active}} \ll n$, the computational cost of the N-body calculations thus remains manageable even if the tail end of the mass distribution is highly resolved.

Ormel's model of dynamical heating only comprises the exchange of kinetic and potential energy during close encounters and thus neglects larger-scale gravitational effects, that is, dust self-gravity. With the average dust-to-gas ratio being $\sim 1:100$, we would expect the long-range gravitational influence of dust to be dwarfed by the gravity exerted by the gas, which would have to be considered by a more sophisticated gas model. However, the local density of dust may be much higher under planet-forming conditions and specifically in dust traps, and may contribute to the formation of plan-

etesimals through gravitational collapse (Johansen et al., 2007), which would be very expensive to simulate. Instead, an effective model for planetesimal formation as proposed by Drążkowska and Dullemond (2014); Drążkowska et al. (2016) could be used, similar to what has been done by Lau et al. (2022). Such a model would continuously convert dust and pebbles to planetesimals if a density threshold is exceeded.

In Tables 6.1 and 6.2 we give a visual overview of the interactions between gas and various types of solids in a protoplanetary disk according to the proposed hybrid simulation method. The tables show quite clearly that several types of interactions are completely neglected. Specifically, dust is assumed to couple to the gas, and its contribution to gravitational deflections of other solid particles is ignored. The proposed model also ignores how the gas disk density is affected gravitationally by solid bodies, with the exception of planets whose influence is represented in the axisymmetric planetary gap model. Despite not participating in dynamical heating processes, dust particles can be accreted by other solid particles, and thereby contribute directly and indirectly to protoplanetary growth. In Ormel's model, there is no mutual coagulation of dust grains; but even though the role of dust coagulation may be less relevant at the stage where planetesimals abound, an end-to-end model of planet formation has to start with dust and hence needs to simulate the dust evolution as well. The evolution of dust is often simulated with grid-based methods (e.g. Birnstiel et al., 2010; Stammler and Birnstiel, 2022). Monte Carlo methods have been used as well to model dust growth (Zsom et al., 2010, 2011b; Krijt et al., 2015), but their scalability is constrained by their high computational cost. We believe that the bucketing scheme renders Monte Carlo methods a more competitive choice for implementing dust coagulation models.

6.5 Summary

This chapter documents our effort to assemble a holistic simulation for the purpose of studying growth processes in protoplanetary dust traps. To this end, the extended RPMC method has been used to simulate collision and dynamical heating interactions stochastically as per Ormel's model, which was elaborated in Sect. 6.1. The extended RPMC method had only rudimentary provisions for collision outcomes other than 'hit-and-stick' coagulation; this was rectified in Sect. 6.2, where a viable representation of fragmentation models for the extended RPMC method was developed. Ormel's model and the refined fragmentation model have been used to test the bucketing scheme with a 'realistic' scenario in Sect. 4.6.3. In a sense, Sects. 6.1 and 6.2 thus are physical supplements to the technical Chapter 4, providing the comprehensive details required for reproducing our results.

Going beyond the representative statistical simulation, Sect. 6.3 introduces an external operator representing a simple gas disk model with provisions for planet-induced pressure bumps. Sect. 6.4 then discusses how a representative particle simulation can be combined with an N-body simulation, which is crucial for correctly reproducing the orbital dynamics of individual bodies, such as planets or planetary embryos, and for simulating radial scattering and redistribution which are not accounted for by the geometrical interaction models used for the stochastic representative particle simulation. Although we have built a prototype implementation of such a hybrid stochastic and de-

terministic simulation, this effort is incomplete, and several challenges still have to be addressed before our code can be used effectively. Here we briefly discuss some of these challenges.

As discussed in Sect. 4.7.2, the bucketing scheme, like any computational scheme, would greatly benefit from a parallel implementation. The bucketing scheme mainly comprises two types of computation: updating the bucket–bucket interaction rate bounds, and sampling interaction events. The former is easily parallelised, but we found an efficient parallelisation of the latter very difficult to achieve. As proposed in Sect. 4.7.2.1, the rejection sampling process might be parallelised by sampling not only the next event but the next P events concurrently, with $P \in \mathbb{N}_+$ being the degree of parallelism. Because sampling potential events is relatively cheap, we also have to compute the likelihood of rejection concurrently in order to benefit from parallelisation. However, this is not trivial because events ultimately have to be simulated in order. An interaction event might alter the properties of the interacting entities, which may cause their interaction rates to change. As a consequence, the likelihood of rejection of future events may change as well. Rejection likelihoods already computed for future events may then have to be discarded, which would render our parallel scheme inefficient. At first glance, this problem may seem negligible because we observed relatively low rates of event acceptance in realistic simulations ($\sim 1\%$, cf. Fig. 4.16); a rejected potential event cannot alter entity properties, and if $\sim 99\%$ of potential events ended up being rejected, one would expect that future event samplings rarely need to be discarded. However, this picture changes if we augment the stochastic representative particle simulation with an external operator, such as the gas drag forces discussed in Sect. 6.3, and with an N-body code, as proposed in Sect. 6.4. A continuous external operator inflicts continuous changes upon all particles affected by it; regardless of the average rate of acceptance, a projection of future event rejection likelihoods is difficult if external operators can modify particle properties at any time. Despite these difficulties, we believe that the sampling process can still be parallelised effectively even in the presence of external operators since the alteration of particle properties does not strictly necessitate the re-evaluation of all projected future potential events involving the particle. Because particle properties will usually undergo small changes, the likelihood of acceptance will often be similar. The overlap between the old and the new acceptance likelihood can be quantified, allowing to retain many event samplings, discarding only some of them while interspersing them with additional potential events. We have begun to explore this strategy with a prototype implementation, noting however that its completion must be left to future work.

Appendix

6.A Local and non-local interactions

In Eq. (C1) of Ormel et al. (2010, Appendix C), the general interaction rate between any two particles from swarms j and k is given as

$$\lambda_{jk} = \int \mathrm{d}^3x\, n_j(\mathbf{x}) n_k(\mathbf{x}) \sigma(\mathbf{x}) v_\mathrm{a} \ , \tag{6.89}$$

where $n_i(\mathbf{x})$ is the *number density distribution* of the particles in swarm i, $\sigma(\mathbf{x}) = \pi R_x R_z$ is the interaction cross-section composed of the geometric interaction radii R_x and R_z, and v_a is the approach velocity. The interaction rate is then decomposed into *filling factors*:

$$\lambda_{jk} = \frac{\pi v_\mathrm{a} N_j N_k}{4(2\pi a)} \phi_x \phi_z \ , \tag{6.90}$$

$$\phi_x \coloneqq \int \mathrm{d}b\, P(b) \bar{P}_\mathrm{int}(b) \ , \tag{6.91}$$

$$\phi_z \coloneqq \int \mathrm{d}z \frac{2R_z}{(2h_j)(2h_k)} \ , \tag{6.92}$$

where the refined version of the filling factor we give in Eq. (6.91) was 'reverse-engineered' from Fig. 22 of Ormel et al. (2010). N_j is the number of particles in swarm j, a is the orbital radius of interaction (presumably an average between the orbital radii of representative particles j and k), and $h_j = r_j(\langle \sin^2 i_j^2 \rangle)^{1/2}$ is the scale height. The distance distribution $P(b)$ is given in Ormel et al. (2010, Eq. (C7)) as

$$P(b) = \int \mathrm{d}x P_j(x) P_k(x+b) \tag{6.93}$$

where $P_j(x)$ is the normalised radial particle density distribution of swarm j. The characteristic function $\bar{P}_\mathrm{int}(b)$ used in Eq. (6.91) determines whether two particles at radial distance b can interact; it can be defined as

$$\bar{P}_\mathrm{int}(b) = \begin{cases} 1 & \text{if } R_\mathrm{int}^\mathrm{min} \leqslant |b| \leqslant R_\mathrm{int} \\ 0 & \text{otherwise,} \end{cases} \tag{6.94}$$

thus formally stating that two particles can interact if their orbital distance is not smaller than $R_\mathrm{int}^\mathrm{min}$ and not greater than R_int. With the simple geometric assumption that the orbital radii of particles in swarm j are distributed homogeneously in an interval of orbital radii $r \in [r_j - h_{x,j}, r_j + h_{x,j}]$, the radial particle density distribution can then be expressed in terms of a characteristic function,

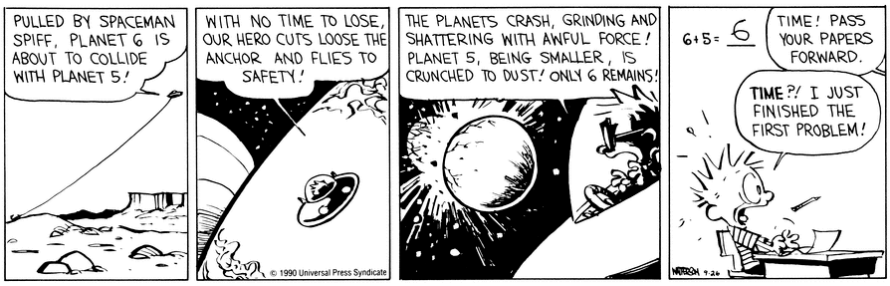
$$\begin{aligned} P_j(x) &\coloneqq \left[\int \mathrm{d}x' \bar{P}_j(r') \right]^{-1} \bar{P}_j(r) \\ &= (2h_{x,j})^{-1} \bar{P}_j(r) \ , \end{aligned} \tag{6.95}$$

where the characteristic function is given as

$$\bar{P}_j(x) = \begin{cases} 1 & \text{if } |x - r_j| \leqslant h_{x,j} \\ 0 & \text{otherwise.} \end{cases} \tag{6.96}$$

Particles interact through gravitation, slightly or severely bending their respective trajectories, or by collision. Gravitational interaction is non-local, that is, particles can exert gravitational force onto each other despite having a non-negligible distance; and even collisions, which are necessarily local, require consideration of the non-local gravitational focussing effect (e.g. Armitage, 2017, §III.B.1). However, Eq. (6.89) only defines a local interaction rate. This implies the assumption that $\sqrt{|\sigma|} \ll V^{1/3}$, or equivalently, that $R_x \ll h_x$ and $R_z \ll h_z$, that is, the interaction lengths are assumed to be smaller than the extents of the population volume by orders of magnitude. These assumptions clearly do not hold for massive runaway bodies whose gravitational reach may greatly exceed their scale length $h_x = re$ and their scale height $h_z = r \sin i$. Despite starting their considerations with a local interaction rate, Ormel et al. (2010) do not actually assume locality; they take precautions to allow for $R_x \sim h_x$ and $R_z \sim h_z$ specifically, and their decomposition of the interaction rate, which we reproduced in Eq. (6.90), is consistent even for non-local interactions. We note, however, that Eq. (6.90) cannot be derived from the local definition of the interaction rate in Eq. (6.89), thus suspecting that the authors have in fact worked with a non-local definition of the interaction rate but have refrained from elaborating on it in their publication.

Summary 7



Bill Watterson, *Calvin and Hobbes*, September 26, 1990 (Andrews McMeel Syndication)

This work was motivated by the desire to study protoplanetary growth processes in planet-induced dust traps, a scenario that is challenging to simulate. Although gravitational interactions among lighter bodies such as planetesimals can be adequately modelled with statistical means, they may also undergo gravitational scattering by planets or runaway bodies and should therefore be subject to N-body calculations as well. However, due to the great number of planetesimals needed to form a planet, individual representation is impractical. At the same time, the eponymous dust component of the solid material in a dust trap, which is aerodynamically coupled to the gas disk, needs to be considered as it may significantly contribute to the growth of planetesimals.

Grid-based methods, which have proven successful for the simulation of dust growth, are inadequate for runaway growth processes, and particle-based simulation methods that allow for a natural transition from representative to individual bodies would therefore be preferable. However, the representative particle methods developed thus far faced several challenges, such as the necessity to maintain a balance of sampling weights in the methods of Ormel and Spaans (2008) and Shima et al. (2009), or an effective upper bound to the mass of individual representative particles in the RPMC method of Zsom and Dullemond (2008). Another impediment to the adoption of particle-based stochastic methods is their high computational cost.

In Chapter 3, a stochastic simulation method has been developed which, though being generally applicable to representative particle simulations, is adequate specifically for the simulation of runaway particle growth as may occur in a dust trap. To this end, the RPMC method of Zsom and Dullemond (2008) has been extended such that unrestricted growth is possible, and the method can now correctly handle individual particles such as runaway bodies that emerge from a representatively sampled system of abundant particles. Besides conducting the usual numerical tests, we have proven the statistical correctness of the RPMC method by analytical means for a generalised two-component test case, hoping to dispel remaining doubts with regard to the soundness of the method.

A novel computational scheme for stochastic interactions among representative particles has then been devised in Chapter 4. By using a dynamically updated grouping of 'buckets' together with a hybrid sampling scheme, the proposed computational scheme has been demonstrated to have linear or better scaling characteristics with regard to the sampling resolution, which is a substantial improvement over the quadratic scaling characteristics of the traditional computational scheme.

In order to use the bucketing scheme, a routine for computing upper bounds to the interaction rates between groups of particles is required. Though near-trivial for simple test kernels, the manual implementation of such a routine for realistic interaction kernels would be tedious and error-prone. In Chapter 5 we have therefore developed a generic programming paradigm for interval arithmetic that enables a programmer to implement a numerical routine in such a way that it may be used either as a direct computation or as a bounds estimation routine.

We have put our method and computational scheme to the test by adopting the interval-aware programming paradigm in our implementation of the interaction rate routines of a spatially resolved statistical model for the evolution of planetesimals. The detailed interaction model, taken from Ormel et al. (2010), has been described in Chapter 6, along with a proposed design for a hybrid deterministic and stochastic simulation that combines the extended RPMC method with an N-body code.

7.1 Discussion and outlook

Although our work was motivated by astrophysical research interests, this thesis resides within the domain of computer science and engineering, and its contributions to the field of astrophysics are methodical in nature. As a representative particle-based method with support for the emergence of individual self-representing particles, the RPMC method lends itself to being complemented with an N-body code. Also, with the increased efficiency of the bucketing scheme, the RPMC method may be a compelling alternative to grid-based methods such as DustPy (Stammler and Birnstiel, 2022) for simulating the growth of dust particles that precedes the formation of planetesimals. Even though the construction of such a hybrid of deterministic and stochastic particle-based methods and the study of possible emergence of planets in dust traps must be left to future work, we believe that this thesis makes a tangible contribution towards such a hybrid simulation method.

Another implementation of a hybrid code has been put forth with LIPAD (Levison et al., 2012), which is a hybrid particle-based stochastic and deterministic code that bears many similarities to our vision of a hybrid simulation method. Unfortunately, the source code of LIPAD has not been made public, so our insights into the inner workings or the computational demands of LIPAD are limited. However, we surmise that our refinement of the RPMC method with regard to the 'regime transition' from representative to individual particles may be beneficial for LIPAD, which suffers from slight accuracy issues with regard to the onset of runaway growth (cf. Levison et al., 2012, Fig. 7); and the bucketing scheme may be generally useful in simulating the stochastic interactions much more efficiently, allowing for a higher overall resolution.

Regardless of our extension, the RPMC method retains the general disadvantage of particle-based methods that it is inefficient under stiff equilibrium conditions where different processes cancel each other. As remarked in Sect. 3.7.1, the 'regime transition' characteristic of our extended method may rather aggravate the problem, leading to an explosion in the number of representative particles and thereby dramatically increasing the computational cost. For simulating steady-state systems, a grid-based method may thus be more appropriate. Although we found no evidence of an exploding number of representative particles in our own simulation of a protoplanetary growth model with fragmentation, it would be reasonable to obviate the problem by further amending the RPMC method with a merging prescription for similar representative particles as done by, for instance, Krijt et al. (2015, §3.4).

Interval arithmetic has rarely been employed to speed up rejection sampling. One such attempt was ventured by Sainudiin and York (2009), who combined a rejection sampling scheme with a sampling estimator built upon interval arithmetic. The authors point out that the interval bounds obtained for the target shape function needs to be tight, and the dimension of the parameter space needs to be low, for their scheme to be efficient. As illustrated by Fig. 4.16, our independently developed method similarly suffers from low acceptance rates (of order $\sim 1\%$) when used with complicated interaction rate functions defined on a high-dimensional property space, requiring a highly resolved grid of buckets and significant constructive effort in crafting an efficiently interval-aware interaction rate function to even get this far. However, even with its low sampling efficiency, the bucketing scheme was found to be a substantial improvement over the traditional sampling scheme, owed mostly to the reduced costs of the interaction rate updating.

Among the technical advancements conceivable for the bucketing scheme, a parallel implementation strategy is the most striking one. Modern computers use massively parallel designs, and hence parallel execution is paramount for efficiency. Although we believe that the bucketing scheme leaves ample opportunities for parallelisation, designing an economical concurrent rejection sampling scheme is not a trivial endeavour.

The bucketing scheme was devised for the purpose of implementing the extended RPMC method more efficiently, but we emphasise that it generally applies to the simulation of compound Poisson point processes, and it may also find use as an alternative to adaptive rejection sampling techniques. Likewise, even though the paradigm of interval-aware programming was developed for easier application of the bucketing scheme, we

believe that it can be more generally useful, for instance in applying arithmetic error analysis to existing numerical routines.

7.2 Conclusion

With the extended RPMC method and the bucketing scheme presented in this work, a representative particle method is now available that can simulate interaction processes spanning a high dynamic range of masses while retaining a balance of sampling weights. With considerable but not prohibitive effort, numerical routines for computing particle interaction rates can be made interval-aware, thus allowing the use of the bucketing scheme whose computational demands scale linearly, as opposed to quadratically, with the number of representative particles.

The extended RPMC method is particularly suited for the simulation of runaway growth processes in protoplanetary disks. With the efficiency of the bucketing scheme, very high resolution can be achieved, and the simulation of highly interactive processes such as the dynamical heating of planetesimals with a Monte Carlo method becomes affordable. We believe that the extended RPMC method is well-suited for being combined with an N-body simulation, and that the ensuing hybrid simulation will be useful for studying the growth dynamics in a planet-induced dust trap, retaining high resolution at moderate cost while correctly describing gap opening and radial redistribution of particles through close gravitational encounters.

List of Acronyms 8

ALMA Atacama Large Millimeter/submillimeter Array 2

IEEE Institute of Electrical and Electronics Engineers

MMSN Minimum Mass Solar Nebula 2

NaN not a number 143

NaR not a real 140

rms root mean square 76

RP Representative Particle

RPMC Representative Particle Monte Carlo 6

Bibliography

Isao Adachi, Chushiro Hayashi, and Kiyoshi Nakazawa. The Gas Drag Effect on the Elliptic Motion of a Solid Body in the Primordial Solar Nebula. *Progress of Theoretical Physics*, 56(6):1756–1771, December 1976. ISSN 0033-068X. doi: 10.1143/PTP.56.1756. URL https://doi.org/10.1143/PTP.56.1756.

Götz Alefeld and Jürgen Herzberger. *Introduction to Interval Computation*. Academic Press, December 2012. ISBN 978-0-08-091636-1. Google-Books-ID: rUsX5x0OqUcC.

ALMA Partnership, C. L. Brogan, L. M. Pérez, T. R. Hunter, W. R. F. Dent, A. S. Hales, R. E. Hills, S. Corder, E. B. Fomalont, C. Vlahakis, Y. Asaki, D. Barkats, A. Hirota, J. A. Hodge, C. M. V. Impellizzeri, R. Kneissl, E. Liuzzo, R. Lucas, N. Marcelino, S. Matsushita, K. Nakanishi, N. Phillips, A. M. S. Richards, I. Toledo, R. Aladro, D. Broguiere, J. R. Cortes, P. C. Cortes, D. Espada, F. Galarza, D. Garcia-Appadoo, L. Guzman-Ramirez, E. M. Humphreys, T. Jung, S. Kameno, R. A. Laing, S. Leon, G. Marconi, A. Mignano, B. Nikolic, L. A. Nyman, M. Radiszcz, A. Remijan, J. A. Rodón, T. Sawada, S. Takahashi, R. P. J. Tilanus, B. Vila Vilaro, L. C. Watson, T. Wiklind, E. Akiyama, E. Chapillon, I. de Gregorio-Monsalvo, J. Di Francesco, F. Gueth, A. Kawamura, C. F. Lee, Q. Nguyen Luong, J. Mangum, V. Pietu, P. Sanhueza, K. Saigo, S. Takakuwa, C. Ubach, T. van Kempen, A. Wootten, A. Castro-Carrizo, H. Francke, J. Gallardo, J. Garcia, S. Gonzalez, T. Hill, T. Kaminski, Y. Kurono, H. Y. Liu, C. Lopez, F. Morales, K. Plarre, G. Schieven, L. Testi, L. Videla, E. Villard, P. Andreani, J. E. Hibbard, and K. Tatematsu. The 2014 ALMA Long Baseline Campaign: First Results from High Angular Resolution Observations toward the HL Tau Region. *The Astrophysical Journal*, 808:L3, July 2015. ISSN 0004-637X. doi: 10.1088/2041-8205/808/1/L3. URL https://ui.adsabs.harvard.edu/abs/2015ApJ...808L...3A. ADS Bibcode: 2015ApJ...808L...3A.

Sean M. Andrews, Jane Huang, Laura M. Pérez, Andrea Isella, Cornelis P. Dullemond, Nicolás T. Kurtovic, Viviana V. Guzmán, John M. Carpenter, David J. Wilner, Shangjia Zhang, Zhaohuan Zhu, Tilman Birnstiel, Xue-Ning Bai, Myriam Benisty, A. Meredith Hughes, Karin I. Öberg, and Luca Ricci. The Disk Substructures at High Angular Resolution Project (DSHARP). I. Motivation, Sample, Calibration, and Overview. *The Astrophysical Journal*, 869(2):L41, December 2018. ISSN 2041-8205. doi: 10.3847/2041-8213/aaf741. URL https://doi.org/10.3847%2F2041-8213%2Faaf741. Publisher: American Astronomical Society.

Philip J. Armitage. Lecture notes on the formation and early evolution of planetary systems, February 2017. URL http://arxiv.org/abs/astro-ph/0701485. arXiv:astro-ph/0701485 version: 6.

Jaehan Bae, Richard Teague, Sean M. Andrews, Myriam Benisty, Stefano Facchini, Maria Galloway-Sprietsma, Ryan A. Loomis, Yuri Aikawa, Felipe Alarcón, Edwin Bergin, Jennifer B. Bergner, Alice S. Booth, Gianni Cataldi, L. Ilsedore Cleeves, Ian Czekala, Viviana V. Guzmán, Jane Huang, John D. Ilee, Nicolas T. Kurtovic, Charles J. Law, Romane Le Gal, Yao Liu, Feng Long, François Ménard, Karin I. Öberg, Laura M. Pérez, Chunhua Qi, Kamber R. Schwarz, Anibal Sierra, Catherine Walsh, David J. Wilner, and Ke Zhang. Molecules with ALMA at Planet-forming Scales (MAPS): A Circumplanetary Disk Candidate in Molecular-line Emission in the AS 209 Disk. *The Astrophysical Journal*, 934:L20, August 2022. ISSN 0004-637X. doi: 10.3847/2041-8213/ac7fa3. URL https://ui.adsabs.harvard.edu/abs/2022ApJ...934L..20B. ADS Bibcode: 2022ApJ...934L..20B.

Jaehan Bae, Andrea Isella, Zhaohuan Zhu, Rebecca Martin, Satoshi Okuzumi, and Scott Suriano. Structured Distributions of Gas and Solids in Protoplanetary Disks, January 2023. URL http://arxiv.org/abs/2210.13314. arXiv:2210.13314 [astro-ph].

Steven A. Balbus and John F. Hawley. A Powerful Local Shear Instability in Weakly Magnetized Disks. I. Linear Analysis. *The Astrophysical Journal*, 376:214, July 1991. ISSN 0004-637X. doi: 10.1086/170270. URL https://ui.adsabs.harvard.edu/abs/1991ApJ...376..214B. ADS Bibcode: 1991ApJ...376..214B.

Edwin A. Bergin, L. Ilsedore Cleeves, Uma Gorti, Ke Zhang, Geoffrey A. Blake, Joel D. Green, Sean M. Andrews, Neal J. Evans, II, Thomas Henning, Karin Öberg, Klaus Pontoppidan, Chunhua Qi, Colette Salyk, and Ewine F. van Dishoeck. An old disk still capable of forming a planetary system. *Nature*, 493:644–646, January 2013. ISSN 0028-0836. doi: 10.1038/nature11805. URL https://ui.adsabs.harvard.edu/abs/2013Natur.493..644B. ADS Bibcode: 2013Natur.493..644B.

Lia Marta Bernabò, Diego Turrini, Leonardo Testi, Francesco Marzari, and Danai Polychroni. Dust Resurgence in Protoplanetary Disks Due to Planetesimal-Planet Interactions. *The Astrophysical Journal*, 927:L22, March 2022. ISSN 0004-637X. doi: 10.3847/2041-8213/ac574e. URL https://ui.adsabs.harvard.edu/abs/2022ApJ...927L..22B. ADS Bibcode: 2022ApJ...927L..22B.

M. Beutel and C. P. Dullemond. An improved Representative Particle Monte Carlo method for the simulation of particle growth. *Astronomy & Astrophysics*, 670:A134, February 2023. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/202244955. URL https://www.aanda.org/10.1051/0004-6361/202244955.

M. Beutel, C. P. Dullemond, and R. Strzodka. Efficient simulation of stochastic interactions among representative Monte Carlo particles. *Astronomy & Astrophysics*, in press.

Moritz Beutel. intervals: Simple C++ library for interval arithmetic, 2022. URL https://github.com/mbeutel/intervals.

Moritz Beutel and Robert Strzodka. A Paradigm for Interval-Aware Programming. In John Gustafson, Siew Hoon Leong, and Marek Michalewicz, editors, *Next Generation Arithmetic*, Lecture Notes in Computer Science, pages 38–60, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-32180-1. doi: 10.1007/978-3-031-32180-1_3.

T. Birnstiel, C. P. Dullemond, and F. Brauer. Gas- and dust evolution in protoplanetary disks. *Astronomy and Astrophysics*, 513:A79, April 2010. ISSN 0004-6361. doi: 10.1051/0004-6361/200913731. URL http://adsabs.harvard.edu/abs/2010A%26A...513A..79B.

T. Birnstiel, M. Fang, and A. Johansen. Dust Evolution and the Formation of Planetesimals. *Space Science Reviews*, 205:41–75, December 2016. ISSN 0038-6308. doi: 10.1007/s11214-016-0256-1. URL https://ui.adsabs.harvard.edu/abs/2016SSRv..205...41B. ADS Bibcode: 2016SSRv..205...41B.

Bertram Bitsch, Michiel Lambrechts, and Anders Johansen. The growth of planets by pebble accretion in evolving protoplanetary discs. *Astronomy and Astrophysics*, 582: A112, October 2015. ISSN 0004-6361. doi: 10.1051/0004-6361/201526463. URL https://ui.adsabs.harvard.edu/abs/2015A&A...582A.112B/abstract.

Jürgen Blum. Dust Evolution in Protoplanetary Discs and the Formation of Planetesimals. What Have We Learned from Laboratory Experiments? *Space Science Reviews*, 214:52, March 2018. ISSN 0038-6308. doi: 10.1007/s11214-018-0486-5. URL https://ui.adsabs.harvard.edu/abs/2018SSRv..214...52B. ADS Bibcode: 2018SSRv..214...52B.

F. Brauer, C. P. Dullemond, and Th Henning. Coagulation, fragmentation and radial motion of solid particles in protoplanetary disks. *Astronomy & Astrophysics*, 480(3):859–877, March 2008. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361:20077759. URL https://www.aanda.org/articles/aa/abs/2008/12/aa7759-07/aa7759-07.html. Number: 3 Publisher: EDP Sciences.

S. Brdar and A. Seifert. McSnow: A Monte-Carlo Particle Model for Riming and Aggregation of Ice Particles in a Multidimensional Microphysical Phase Space. *Journal of Advances in Modeling Earth Systems*, 10:187–206, January 2018. doi: 10.1002/2017MS001167. URL https://ui.adsabs.harvard.edu/abs/2018JAMES..10..187B. ADS Bibcode: 2018JAMES..10..187B.

Hervé Brönnimann, Guillaume Melquiond, and Sylvain Pion. The design of the Boost interval arithmetic library. *Theoretical Computer Science*, 351(1):111–118, February 2006. ISSN 0304-3975. doi: 10.1016/j.tcs.2005.09.062.

M. Bukhari Syed, J. Blum, K. Wahlberg Jansson, and A. Johansen. The Role of Pebble Fragmentation in Planetesimal Formation. I. Experimental Study. *The Astrophysical Journal*, 834:145, January 2017. ISSN 0004-637X. doi: 10.3847/1538-4357/834/2/145. URL https://ui.adsabs.harvard.edu/abs/2017ApJ...834..145B. ADS Bibcode: 2017ApJ...834..145B.

Daniel Carrera, Anders Johansen, and Melvyn B. Davies. How to form planetesimals from mm-sized chondrules and chondrule aggregates. *Astronomy and Astrophysics*, 579:A43, July 2015. ISSN 0004-6361. doi: 10.1051/0004-6361/

201425120. URL https://ui.adsabs.harvard.edu/abs/2015A&A...579A..43C. ADS Bibcode: 2015A&A...579A..43C.

Daniel Carrera, Jacob B. Simon, Rixin Li, Katherine A. Kretke, and Hubert Klahr. Protoplanetary Disk Rings as Sites for Planetesimal Formation. *The Astronomical Journal*, 161:96, February 2021. ISSN 0004-6256. doi: 10.3847/1538-3881/abd4d9. URL https://ui.adsabs.harvard.edu/abs/2021AJ....161...96C. ADS Bibcode: 2021AJ....161...96C.

John Chambers. A semi-analytic model for oligarchic growth. *Icarus*, 180(2):496–513, February 2006. ISSN 0019-1035. doi: 10.1016/j.icarus.2005.10.017. URL http://www.sciencedirect.com/science/article/pii/S0019103505004215.

Bruce Christianson and Maurice Cox. Automatic Propagation of Uncertainties. In Martin Bücker, George Corliss, Uwe Naumann, Paul Hovland, and Boyana Norris, editors, *Automatic Differentiation: Applications, Theory, and Implementations*, Lecture Notes in Computational Science and Engineering, pages 47–58, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-28438-3. doi: 10.1007/3-540-28438-9_4.

Thayne Currie, Kellen Lawson, Glenn Schneider, Wladimir Lyra, John Wisniewski, Carol Grady, Olivier Guyon, Motohide Tamura, Takayuki Kotani, Hajime Kawahara, Timothy Brandt, Taichi Uyama, Takayuki Muto, Ruobing Dong, Tomoyuki Kudo, Jun Hashimoto, Misato Fukagawa, Kevin Wagner, Julien Lozi, Jeffrey Chilcote, Taylor Tobin, Tyler Groff, Kimberly Ward-Duong, William Januszewski, Barnaby Norris, Peter Tuthill, Nienke van der Marel, Michael Sitko, Vincent Deo, Sebastien Vievard, Nemanja Jovanovic, Frantz Martinache, and Nour Skaf. Images of embedded Jovian planet formation at a wide separation around AB Aurigae. *Nature Astronomy*, 6:751–759, April 2022. ISSN 2397-3366. doi: 10.1038/s41550-022-01634-x. URL https://ui.adsabs.harvard.edu/abs/2022NatAs...6..751C. ADS Bibcode: 2022NatAs...6..751C.

Luiz Henrique de Figueiredo and Jorge Stolfi. Affine Arithmetic: Concepts and Applications. *Numerical Algorithms*, 37(1):147–158, December 2004. ISSN 1572-9265. doi: 10.1023/B:NUMA.0000049462.70970.b6.

Ruobing Dong, Zhaohuan Zhu, and Barbara Whitney. Observational Signatures of Planets in Protoplanetary Disks I. Gaps Opened by Single and Multiple Young Planets in Disks. *The Astrophysical Journal*, 809:93, August 2015. ISSN 0004-637X. doi: 10.1088/0004-637X/809/1/93. URL https://ui.adsabs.harvard.edu/abs/2015ApJ...809...93D. ADS Bibcode: 2015ApJ...809...93D.

J. Drążkowska and C. P. Dullemond. Can dust coagulation trigger streaming instability? *Astronomy & Astrophysics, Volume 572, id.A78, <NUMPAGES>12</NUMPAGES> pp.*, 572:A78, December 2014. ISSN 0004-6361. doi: 10.1051/0004-6361/201424809. URL https://ui.adsabs.harvard.edu/abs/2014A%26A...572A..78D/abstract.

J. Drążkowska, F. Windmark, and C. P. Dullemond. Planetesimal formation via sweep-up growth at the inner edge of dead zones. *Astronomy and Astrophysics*, 556:A37, August 2013. ISSN 0004-6361. doi: 10.1051/0004-6361/201321566. URL https://ui.adsabs.harvard.edu/abs/2013A&A...556A..37D/abstract.

J. Drążkowska, F. Windmark, and C. P. Dullemond. Modeling dust growth in protoplanetary disks: The breakthrough case. *Astronomy & Astrophysics*, 567:A38, July 2014. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201423708. URL https://www.aanda.org/articles/aa/abs/2014/07/aa23708-14/aa23708-14.html. Publisher: EDP Sciences.

J. Drążkowska, Y. Alibert, and B. Moore. Close-in planetesimal formation by pile-up of drifting pebbles. *Astronomy & Astrophysics*, 594:A105, October 2016. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201628983. URL https://www.aanda.org/articles/aa/abs/2016/10/aa28983-16/aa28983-16.html. Publisher: EDP Sciences.

Joanna Drążkowska, Shengtai Li, Til Birnstiel, Sebastian M. Stammler, and Hui Li. Including Dust Coagulation in Hydrodynamic Models of Protoplanetary Disks: Dust Evolution in the Vicinity of a Jupiter-mass Planet. *The Astrophysical Journal*, 885:91, November 2019. ISSN 0004-637X. doi: 10.3847/1538-4357/ab46b7. URL https://ui.adsabs.harvard.edu/abs/2019ApJ...885...91D. ADS Bibcode: 2019ApJ...885...91D.

Joanna Drążkowska, Bertram Bitsch, Michiel Lambrechts, Gijs D. Mulders, Daniel Harsono, Allona Vazan, Beibei Liu, Chris W. Ormel, Katherine Kretke, and Alessandro Morbidelli. Planet Formation Theory in the Era of ALMA and Kepler: from Pebbles to Exoplanets, October 2022. URL http://arxiv.org/abs/2203.09759. arXiv:2203.09759 [astro-ph].

C. P. Dullemond and C. Dominik. Dust coagulation in protoplanetary disks: A rapid depletion of small grains. *Astronomy & Astrophysics*, 434(3):971–986, May 2005. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361:20042080. URL https://www.aanda.org/articles/aa/abs/2005/18/aa2080/aa2080.html. Number: 3 Publisher: EDP Sciences.

Cornelis P Dullemond. Theoretical Models of the Structure of Protoplanetary Disks. *Lecture at workshop in Les Houches*, page 15, 2013.

Cornelis P. Dullemond, Tilman Birnstiel, Jane Huang, Nicolás T. Kurtovic, Sean M. Andrews, Viviana V. Guzmán, Laura M. Pérez, Andrea Isella, Zhaohuan Zhu, Myriam Benisty, David J. Wilner, Xue-Ning Bai, John M. Carpenter, Shangjia Zhang, and Luca Ricci. The Disk Substructures at High Angular Resolution Project (DSHARP). VI. Dust Trapping in Thin-ringed Protoplanetary Disks. *The Astrophysical Journal*, 869 (2):L46, December 2018. ISSN 2041-8205. doi: 10.3847/2041-8213/aaf742. URL https://doi.org/10.3847%2F2041-8213%2Faaf742. Publisher: American Astronomical Society.

Piotr Dziekan and Hanna Pawlowska. Stochastic coalescence in Lagrangian cloud microphysics. *Atmospheric Chemistry and Physics*, 17(22):13509–13520, November 2017. ISSN 1680-7316. doi: 10.5194/acp-17-13509-2017. URL https://acp.copernicus.org/articles/17/13509/2017/. Publisher: Copernicus GmbH.

Alexandre Emsenhuber, Christoph Mordasini, Remo Burn, Yann Alibert, Willy Benz, and Erik Asphaug. The New Generation Planetary Population Synthesis (NGPPS). I. Bern global model of planet formation and evolution, model tests, and emerging planetary systems. *Astronomy & Astrophysics, Volume 656, id.A69,*

<*NUMPAGES*>44</*NUMPAGES*> *pp.*, 656:A69, December 2021. ISSN 0004-6361. doi: 10.1051/0004-6361/202038553. URL https://ui.adsabs.harvard.edu/abs/2021A%26A...656A..69E/abstract.

Paul S. Epstein. On the Resistance Experienced by Spheres in their Motion through Gases. *Physical Review*, 23:710–733, June 1924. ISSN 1536-6065. doi: 10.1103/PhysRev.23.710. URL https://ui.adsabs.harvard.edu/abs/1924PhRv...23..710E. ADS Bibcode: 1924PhRv...23..710E.

Linn E. J. Eriksson, Thomas Ronnet, and Anders Johansen. The fate of planetesimals formed at planetary gap edges. *Astronomy & Astrophysics*, 648:A112, April 2021. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/202039889. URL https://www.aanda.org/articles/aa/abs/2021/04/aa39889-20/aa39889-20.html. Publisher: EDP Sciences.

Daniel T. Gillespie. An Exact Method for Numerically Simulating the Stochastic Coalescence Process in a Cloud. *Journal of Atmospheric Sciences*, 32:1977–1989, October 1975. ISSN 0022-4928. doi: 10.1175/1520-0469(1975)032<1977:AEMFNS>2.0.CO;2. URL https://ui.adsabs.harvard.edu/abs/1975JAtS...32.1977G. ADS Bibcode: 1975JAtS...32.1977G.

P. Goldreich and S. Tremaine. The excitation and evolution of density waves. *The Astrophysical Journal*, 222:850–858, June 1978. ISSN 0004-637X. doi: 10.1086/156203. URL https://ui.adsabs.harvard.edu/abs/1978ApJ...222..850G. ADS Bibcode: 1978ApJ...222..850G.

P. Goldreich and S. Tremaine. The excitation of density waves at the Lindblad and corotation resonances by an external potential. *The Astrophysical Journal*, 233:857–871, November 1979. ISSN 0004-637X. doi: 10.1086/157448. URL https://ui.adsabs.harvard.edu/abs/1979ApJ...233..857G. ADS Bibcode: 1979ApJ...233..857G.

Peter Goldreich, Yoram Lithwick, and Re'em Sari. Planet Formation by Coagulation: A Focus on Uranus and Neptune. *Annual Review of Astronomy and Astrophysics*, 42:549–601, September 2004. ISSN 0066-4146. doi: 10.1146/annurev.astro.42.053102.134004. URL https://ui.adsabs.harvard.edu/abs/2004ARA&A..42..549G. ADS Bibcode: 2004ARA&A..42..549G.

J. F. Gonzalez, G. Laibe, and S. T. Maddison. Self-induced dust traps: overcoming planet formation barriers. *Monthly Notices of the Royal Astronomical Society*, 467:1984–1996, May 2017. ISSN 0035-8711. doi: 10.1093/mnras/stx016. URL https://ui.adsabs.harvard.edu/abs/2017MNRAS.467.1984G. ADS Bibcode: 2017MNRAS.467.1984G.

Frédéric Goualard. Fast and Correct SIMD Algorithms for Interval Arithmetic. page 10, 2009.

Frédéric Goualard. GAOL (Not Just Another Interval Library), 2015. URL https://frederic.goualard.net/#research-software-gaol.

Wojciech W. Grabowski, Hugh Morrison, Shin-Ichiro Shima, Gustavo C. Abade, Piotr Dziekan, and Hanna Pawlowska. Modeling of Cloud Microphysics: Can We Do Better? *Bulletin of the American Meteorological Society*, 100(4):655–672, April 2019. ISSN 0003-0007, 1520-0477. doi: 10.1175/BAMS-D-18-0005.1. URL https://journals.ametsoc.org/view/journals/bams/100/4/bams-d-18-0005.1.xml. Publisher: American Meteorological Society Section: Bulletin of the American Meteorological Society.

Richard Greenberg, John F. Wacker, William K. Hartmann, and Clark R. Chapman. Planetesimals to planets: Numerical simulation of collisional evolution. *Icarus*, 35: 1–26, July 1978. ISSN 0019-1035. doi: 10.1016/0019-1035(78)90057-X. URL https://ui.adsabs.harvard.edu/abs/1978Icar...35....1G. ADS Bibcode: 1978Icar...35....1G.

Yuval Greenzweig and Jack J. Lissauer. Accretion rates of protoplanets: II. Gaussian distributions of planetesimal velocities. *Icarus*, 100(2):440–463, December 1992. ISSN 0019-1035. doi: 10.1016/0019-1035(92)90110-S. URL https://www.sciencedirect.com/science/article/pii/001910359290110S.

John L. Gustafson. *The End of Error: Unum Computing*. Chapman and Hall/CRC, New York, February 2017a. ISBN 978-1-315-16153-2. doi: 10.1201/9781315161532.

John L Gustafson. Posit Arithmetic, October 2017b.

John L Gustafson and Isaac Yonemoto. Beating Floating Point at its Own Game: Posit Arithmetic. *Supercomputing Frontiers and Innovations*, 4(2):16, 2017.

C. Güttler, J. Blum, A. Zsom, C. W. Ormel, and C. P. Dullemond. The outcome of protoplanetary dust growth: pebbles, boulders, or planetesimals? - I. Mapping the zoo of laboratory collision experiments. *Astronomy & Astrophysics*, 513:A56, April 2010. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/200912852. URL https://www.aanda.org/articles/aa/abs/2010/05/aa12852-09/aa12852-09.html. Publisher: EDP Sciences.

S. Y. Haffert, A. J. Bohn, J. de Boer, I. A. G. Snellen, J. Brinchmann, J. H. Girard, C. U. Keller, and R. Bacon. Two accreting protoplanets around the young star PDS 70. *Nature Astronomy*, 3:749–754, June 2019. ISSN 2397-3366. doi: 10.1038/s41550-019-0780-5. URL https://ui.adsabs.harvard.edu/abs/2019NatAs...3..749H. ADS Bibcode: 2019NatAs...3..749H.

C. Hayashi. Formation of the planets. 93:113–126, January 1981a. URL https://ui.adsabs.harvard.edu/abs/1981IAUS...93..113H. Conference Name: Fundamental Problems in the Theory of Stellar Evolution ADS Bibcode: 1981IAUS...93..113H.

C. Hayashi. Structure of the Solar Nebula, Growth and Decay of Magnetic Fields and Effects of Magnetic and Turbulent Viscosities on the Nebula. *Progress of Theoretical Physics Supplement*, 70:35–53, January 1981b. ISSN 0375-9687. doi: 10.1143/PTPS.70.35. URL https://ui.adsabs.harvard.edu/abs/1981PThPS..70...35H. ADS Bibcode: 1981PThPS..70...35H.

T. Hickey, Q. Ju, and M. H. Van Emden. Interval arithmetic: From principles to implementation. *Journal of the ACM*, 48(5):1038–1068, September 2001. ISSN 0004-5411. doi: 10.1145/502102.502106.

Robin J. Hogan. Fast Reverse-Mode Automatic Differentiation using Expression Templates in C++. *ACM Transactions on Mathematical Software*, 40(4):1–16, June 2014. ISSN 0098-3500, 1557-7295. doi: 10.1145/2560359.

Jane Huang, Sean M. Andrews, Cornelis P. Dullemond, Andrea Isella, Laura M. Pérez, Viviana V. Guzmán, Karin I. Öberg, Zhaohuan Zhu, Shangjia Zhang, Xue-Ning Bai, Myriam Benisty, Tilman Birnstiel, John M. Carpenter, A. Meredith Hughes, Luca Ricci, Erik Weaver, and David J. Wilner. The Disk Substructures at High Angular Resolution Project (DSHARP). II. Characteristics of Annular Substructures. *The Astrophysical Journal*, 869(2):L42, December 2018. ISSN 2041-8205. doi: 10.3847/2041-8213/aaf740. URL https://doi.org/10.3847%2F2041-8213%2Faaf740. Publisher: American Astronomical Society.

Shigeru Ida. Stirring and dynamical friction rates of planetesimals in the solar gravitational field. *Icarus*, 88(1):129–145, November 1990. ISSN 0019-1035. doi: 10.1016/0019-1035(90)90182-9. URL http://www.sciencedirect.com/science/article/pii/0019103590901829.

Shigeru Ida and Junichiro Makino. N-body simulation of gravitational interaction between planetesimals and a protoplanet. II. Dynamical friction. *Icarus*, 98:28–37, July 1992a. ISSN 0019-1035. doi: 10.1016/0019-1035(92)90203-J. URL https://ui.adsabs.harvard.edu/abs/1992Icar...98...28I. ADS Bibcode: 1992Icar...98...28I.

Shigeru Ida and Junichiro Makino. N-Body simulation of gravitational interaction between planetesimals and a protoplanet. I. velocity distribution of planetesimals. *Icarus*, 96:107–120, March 1992b. ISSN 0019-1035. doi: 10.1016/0019-1035(92)90008-U. URL https://ui.adsabs.harvard.edu/abs/1992Icar...96..107I. ADS Bibcode: 1992Icar...96..107I.

Shigeru Ida and Junichiro Makino. Scattering of Planetesimals by a Protoplanet: Slowing Down of Runaway Growth. *Icarus*, 106(1):210–227, November 1993. ISSN 0019-1035. doi: 10.1006/icar.1993.1167. URL http://www.sciencedirect.com/science/article/pii/S001910358371167X.

Shigeru Ida, Tristan Guillot, and Alessandro Morbidelli. Accretion and Destruction of Planetesimals in Turbulent Disks. *The Astrophysical Journal*, 686:1292–1301, October 2008. ISSN 0004-637X. doi: 10.1086/591903. URL https://ui.adsabs.harvard.edu/abs/2008ApJ...686.1292I. ADS Bibcode: 2008ApJ...686.1292I.

IEEE Computer Society. IEEE Standard for Interval Arithmetic. *IEEE Std 1788-2015*, pages 1–97, June 2015. doi: 10.1109/IEEESTD.2015.7140721.

IEEE Computer Society. IEEE Standard for Interval Arithmetic (Simplified). *IEEE Std 1788.1-2017*, pages 1–38, January 2018. doi: 10.1109/IEEESTD.2018.8277144. Conference Name: IEEE Std 1788.1-2017.

IEEE Computer Society. IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pages 1–84, July 2019. doi: 10.1109/IEEESTD.2019.8766229.

Satoshi Inaba, Hidekazu Tanaka, Kiyoshi Nakazawa, George W. Wetherill, and Eiichiro Kokubo. High-Accuracy Statistical Simulation of Planetary Accretion: II. Comparison with N-Body Simulation. *Icarus*, 149(1):235–250, January 2001. ISSN 0019-1035. doi: 10.1006/icar.2000.6533. URL https://www.sciencedirect.com/science/article/pii/S0019103500965333.

A. Johansen, A. N. Youdin, and Y. Lithwick. Adding particle collisions to the formation of asteroids and Kuiper belt objects via streaming instabilities. *Astronomy and Astrophysics*, 537:A125, January 2012. ISSN 0004-6361. doi: 10.1051/0004-6361/201117701. URL https://ui.adsabs.harvard.edu/abs/2012A&A...537A.125J. ADS Bibcode: 2012A&A...537A.125J.

Anders Johansen, Jeffrey S. Oishi, Mordecai-Mark Mac Low, Hubert Klahr, Thomas Henning, and Andrew Youdin. Rapid planetesimal formation in turbulent circumstellar disks. *Nature*, 448:1022–1025, August 2007. ISSN 0028-0836. doi: 10.1038/nature06086. URL https://ui.adsabs.harvard.edu/abs/2007Natur.448.1022J. ADS Bibcode: 2007Natur.448.1022J.

Anders Johansen, Andrew Youdin, and Mordecai-Mark Mac Low. Particle Clumping and Planetesimal Formation Depend Strongly on Metallicity. *The Astrophysical Journal*, 704:L75–L79, October 2009. ISSN 0004-637X. doi: 10.1088/0004-637X/704/2/L75. URL https://ui.adsabs.harvard.edu/abs/2009ApJ...704L..75J. ADS Bibcode: 2009ApJ...704L..75J.

K. D. Kanagawa, H. Tanaka, T. Muto, T. Tanigawa, and T. Takeuchi. Formation of a disc gap induced by a planet: effect of the deviation from Keplerian disc rotation. *Monthly Notices of the Royal Astronomical Society*, 448:994–1006, March 2015. ISSN 0035-8711. doi: 10.1093/mnras/stv025. URL https://ui.adsabs.harvard.edu/abs/2015MNRAS.448..994K. ADS Bibcode: 2015MNRAS.448..994K.

Kazuhiro D. Kanagawa, Hidekazu Tanaka, Takayuki Muto, and Takayuki Tanigawa. Modelling of deep gaps created by giant planets in protoplanetary disks. *Publications of the Astronomical Society of Japan*, 69:97, December 2017. ISSN 0004-6264. doi: 10.1093/pasj/psx114. URL https://ui.adsabs.harvard.edu/abs/2017PASJ...69...97K. ADS Bibcode: 2017PASJ...69...97K.

Kazuhiro D. Kanagawa, Hideko Nomura, Takashi Tsukagoshi, Takayuki Muto, and Ryohei Kawabe. Model of a Gap Formed by a Planet with Fast Inward Migration. *The Astrophysical Journal*, 892:83, April 2020. ISSN 0004-637X. doi: 10.3847/1538-4357/ab781e. URL https://ui.adsabs.harvard.edu/abs/2020ApJ...892...83K. ADS Bibcode: 2020ApJ...892...83K.

Immanuel Kant. *Allgemeine Naturgeschichte und Theorie des Himmels, oder Versuch von der Verfassung und dem mechanischen Ursprunge des ganzen Weltgebäudes nach Newtonischen Grundsätzen abgehandelt.* Johann Friederich Petersen, Königsberg and Leipzig, 1755.

P. Karpiński and J. McDonald. A high-performance portable abstract interface for explicit SIMD vectorization. In *Proceedings of the 8th International Workshop on Programming Models and Applications for Multicores and Manycores*, PMAM'17, pages 21–28, New York, NY, USA, February 2017. Association for Computing Machinery. ISBN 978-1-4503-4883-6. doi: 10.1145/3026937.3026939.

R. B. Kearfott. Interval Computations: Introduction, Uses, and Resources. page 23, 2000.

R. Baker Kearfott and V. Kreinovich. *Applications of Interval Computations*. Springer Science & Business Media, December 2013. ISBN 978-1-4613-3440-8. Google-Books-ID: ONDgBwAAQBAJ.

M. Keppler, M. Benisty, A. Müller, Th. Henning, R. van Boekel, F. Cantalloube, C. Ginski, R. G. van Holstein, A. L. Maire, A. Pohl, M. Samland, H. Avenhaus, J. L. Baudino, A. Boccaletti, J. de Boer, M. Bonnefoy, G. Chauvin, S. Desidera, M. Langlois, C. Lazzoni, G. D. Marleau, C. Mordasini, N. Pawellek, T. Stolker, A. Vigan, A. Zurlo, T. Birnstiel, W. Brandner, M. Feldt, M. Flock, J. Girard, R. Gratton, J. Hagelberg, A. Isella, M. Janson, A. Juhasz, J. Kemmer, Q. Kral, A. M. Lagrange, R. Launhardt, A. Matter, F. Ménard, J. Milli, P. Mollière, J. Olofsson, L. Pérez, P. Pinilla, C. Pinte, S. P. Quanz, T. Schmidt, S. Udry, Z. Wahhaj, J. P. Williams, E. Buenzli, M. Cudel, C. Dominik, R. Galicher, M. Kasper, J. Lannier, D. Mesa, D. Mouillet, S. Peretti, C. Perrot, G. Salter, E. Sissa, F. Wildi, L. Abe, J. Antichi, J. C. Augereau, A. Baruffolo, P. Baudoz, A. Bazzon, J. L. Beuzit, P. Blanchard, S. S. Brems, T. Buey, V. De Caprio, M. Carbillet, M. Carle, E. Cascone, A. Cheetham, R. Claudi, A. Costille, A. Delboulbé, K. Dohlen, D. Fantinel, P. Feautrier, T. Fusco, E. Giro, L. Gluck, C. Gry, N. Hubin, E. Hugot, M. Jaquet, D. Le Mignant, M. Llored, F. Madec, Y. Magnard, P. Martinez, D. Maurel, M. Meyer, O. Möller-Nilsson, T. Moulin, L. Mugnier, A. Origné, A. Pavlov, D. Perret, C. Petit, J. Pragt, P. Puget, P. Rabou, J. Ramos, F. Rigal, S. Rochat, R. Roelfsema, G. Rousset, A. Roux, B. Salasnich, J. F. Sauvage, A. Sevin, C. Soenke, E. Stadler, M. Suarez, M. Turatto, and L. Weber. Discovery of a planetary-mass companion within the gap of the transition disk around PDS 70. *Astronomy and Astrophysics*, 617:A44, September 2018. ISSN 0004-6361. doi: 10.1051/0004-6361/201832957. URL `https://ui.adsabs.harvard.edu/abs/2018A&A...617A..44K`. ADS Bibcode: 2018A&A...617A..44K.

Oliver Keszöcze, Marcel Brand, Michael Witterauf, Christian Heidorn, and Jürgen Teich. Aarith: an arbitrary precision number library. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, SAC '21, pages 529–534, New York, NY, USA, April 2021. Association for Computing Machinery. ISBN 978-1-4503-8104-8. doi: 10.1145/3412841.3442085.

Hubert Klahr and Andreas Schreiber. Turbulence Sets the Length Scale for Planetesimal Formation: Local 2D Simulations of Streaming Instability and Planetesimal Formation. *The Astrophysical Journal*, 901:54, September 2020. ISSN 0004-637X. doi: 10.3847/1538-4357/abac58. URL `https://ui.adsabs.harvard.edu/abs/2020ApJ...901...54K`. ADS Bibcode: 2020ApJ...901...54K.

Eiichiro Kokubo and Shigeru Ida. On Runaway Growth of Planetesimals. *Icarus*, 123:180–191, September 1996. ISSN 0019-1035. doi: 10.1006/icar.1996.

0148. URL https://ui.adsabs.harvard.edu/abs/1996Icar..123..180K. ADS Bibcode: 1996Icar..123..180K.

Eiichiro Kokubo and Shigeru Ida. Oligarchic Growth of Protoplanets. *Icarus*, 131:171–178, January 1998. ISSN 0019-1035. doi: 10.1006/icar.1997.5840. URL https://ui.adsabs.harvard.edu/abs/1998Icar..131..171K. ADS Bibcode: 1998Icar..131..171K.

Eiichiro Kokubo and Shigeru Ida. Formation of Protoplanets from Planetesimals in the Solar Nebula. *Icarus*, 143(1):15–27, January 2000. ISSN 0019-1035. doi: 10.1006/icar.1999.6237. URL http://www.sciencedirect.com/science/article/pii/S0019103599962371.

Eiichiro Kokubo and Shigeru Ida. Formation of Protoplanet Systems and Diversity of Planetary Systems. *The Astrophysical Journal*, 581(1):666, December 2002. ISSN 0004-637X. doi: 10.1086/344105. URL https://iopscience.iop.org/article/10.1086/344105/meta. Publisher: IOP Publishing.

Kaitlin Kratter and Giuseppe Lodato. Gravitational Instabilities in Circumstellar Disks. *Annual Review of Astronomy and Astrophysics*, 54:271–311, September 2016. ISSN 0066-4146. doi: 10.1146/annurev-astro-081915-023307. URL https://ui.adsabs.harvard.edu/abs/2016ARA&A..54..271K. ADS Bibcode: 2016ARA&A..54..271K.

S. Krijt, C. W. Ormel, C. Dominik, and A. G. G. M. Tielens. Erosion and the limits to planetesimal growth. *Astronomy & Astrophysics*, 574:A83, February 2015. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201425222. URL https://www.aanda.org/articles/aa/abs/2015/02/aa25222-14/aa25222-14.html. Publisher: EDP Sciences.

Sebastiaan Krijt and Fred J. Ciesla. Dust Diffusion and Settling in the Presence of Collisions: Trapping (sub)micron Grains in the Midplane. *The Astrophysical Journal*, 822:111, May 2016. ISSN 0004-637X. doi: 10.3847/0004-637X/822/2/111. URL https://ui.adsabs.harvard.edu/abs/2016ApJ...822..111K. ADS Bibcode: 2016ApJ...822..111K.

Sebastiaan Krijt, Fred J. Ciesla, and Edwin A. Bergin. Tracing Water Vapor and Ice During Dust Growth. *The Astrophysical Journal*, 833:285, December 2016. ISSN 0004-637X. doi: 10.3847/1538-4357/833/2/285. URL https://ui.adsabs.harvard.edu/abs/2016ApJ...833..285K. ADS Bibcode: 2016ApJ...833..285K.

H. H. Ku. Notes on the Use of Propagation of Error Formulas, 1966. URL https://doi.org/10.6028%2Fjres.070c.025.

M. Lambrechts and A. Johansen. Rapid growth of gas-giant cores by pebble accretion. *Astronomy & Astrophysics*, 544:A32, August 2012. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201219127. URL https://www.aanda.org/articles/aa/abs/2012/08/aa19127-12/aa19127-12.html. Publisher: EDP Sciences.

Tommy Chi Ho Lau, Joanna Drążkowska, Sebastian M. Stammler, Tilman Birnstiel, and Cornelis P. Dullemond. Rapid formation of massive planetary cores in a pressure bump. *Astronomy and Astrophysics*, 668:A170, December 2022. ISSN 0004-6361. doi: 10.1051/0004-6361/202244864. URL https://ui.adsabs.harvard.edu/abs/2022A&A...668A.170L. ADS Bibcode: 2022A&A...668A.170L.

Gregory Laughlin, Peter Bodenheimer, and Fred C. Adams. The Core Accretion Model Predicts Few Jovian-Mass Planets Orbiting Red Dwarfs. *The Astrophysical Journal*, 612:L73–L76, September 2004. ISSN 0004-637X. doi: 10.1086/424384. URL https://ui.adsabs.harvard.edu/abs/2004ApJ...612L..73L. ADS Bibcode: 2004ApJ...612L..73L.

Justin Le. Automatic Propagation of Uncertainty with AD, May 2016. URL https://blog.jle.im/entry/automatic-propagation-of-uncertainty-with-ad.html.

Harold F. Levison, Martin J. Duncan, and Edward Thommes. A LAGRANGIAN INTEGRATOR FOR PLANETARY ACCRETION AND DYNAMICS (LIPAD). *The Astronomical Journal*, 144(4):119, September 2012. ISSN 1538-3881. doi: 10.1088/0004-6256/144/4/119. URL https://doi.org/10.1088%2F0004-6256%2F144%2F4%2F119. Publisher: IOP Publishing.

Rixin Li, Andrew N. Youdin, and Jacob B. Simon. Demographics of Planetesimals Formed by the Streaming Instability. *The Astrophysical Journal*, 885:69, November 2019. ISSN 0004-637X. doi: 10.3847/1538-4357/ab480d. URL https://ui.adsabs.harvard.edu/abs/2019ApJ...885...69L. ADS Bibcode: 2019ApJ...885...69L.

Beibei Liu, Chris W. Ormel, and Anders Johansen. Growth after the streaming instability. From planetesimal accretion to pebble accretion. *Astronomy and Astrophysics*, 624:A114, April 2019. ISSN 0004-6361. doi: 10.1051/0004-6361/201834174. URL https://ui.adsabs.harvard.edu/abs/2019A&A...624A.114L. ADS Bibcode: 2019A&A...624A.114L.

R. V. E. Lovelace, H. Li, S. A. Colgate, and A. F. Nelson. Rossby Wave Instability of Keplerian Accretion Disks. *The Astrophysical Journal*, 513:805–810, March 1999. ISSN 0004-637X. doi: 10.1086/306900. URL https://ui.adsabs.harvard.edu/abs/1999ApJ...513..805L. ADS Bibcode: 1999ApJ...513..805L.

Johan Mabille and Sylvain Corlay. xsimd: C++ wrappers for SIMD intrinsics and parallelized, optimized mathematical functions, 2022. URL https://github.com/xtensor-stack/xsimd.

Scott Meyers. *More Effective C++: 35 New Ways to Improve Your Programs and Designs, PDF Version*. Pearson Education, December 1995. ISBN 978-0-13-279747-4.

R. E. Moore. Interval Analysis, 1966.

Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, January 2009. ISBN 978-0-89871-669-6 978-0-89871-771-6. doi: 10.1137/1.9780898717716.

NVIDIA. CUDA C++ Programming Guide. page 454, 2022. URL https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

C. R. O'dell, Zheng Wen, and Xihai Hu. Discovery of New Objects in the Orion Nebula on HST Images: Shocks, Compact Sources, and Protoplanetary Disks. *The Astrophysical Journal*, 410:696, June 1993. ISSN 0004-637X. doi: 10.1086/

172786. URL https://ui.adsabs.harvard.edu/abs/1993ApJ...410..696O. ADS Bibcode: 1993ApJ...410..696O.

Masahiro Ogihara, Shigeru Ida, and Alessandro Morbidelli. Accretion of terrestrial planets from oligarchs in a turbulent disk. *Icarus*, 188:522–534, June 2007. ISSN 0019-1035. doi: 10.1016/j.icarus.2006.12.006. URL https://ui.adsabs.harvard.edu/abs/2007Icar..188..522O. ADS Bibcode: 2007Icar..188..522O.

Keiji Ohtsuki, Yoshitsugu Nakagawa, and Kiyoshi Nakazawa. Artificial acceleration in accumulation due to coarse mass-coordinate divisions in numerical simulation. *Icarus*, 83(1):205–215, January 1990. ISSN 0019-1035. doi: 10.1016/0019-1035(90)90015-2. URL https://www.sciencedirect.com/science/article/pii/0019103590900152.

Keiji Ohtsuki, Glen R. Stewart, and Shigeru Ida. Evolution of Planetesimal Velocities Based on Three-Body Orbital Integrations and Growth of Protoplanets. *Icarus*, 155 (2):436–453, February 2002. ISSN 0019-1035. doi: 10.1006/icar.2001.6741. URL https://www.sciencedirect.com/science/article/pii/S0019103501967417.

Satoshi Okuzumi, Hidekazu Tanaka, and Masa-aki Sakagami. Numerical Modeling of the Coagulation and Porosity Evolution of Dust Aggregates. *The Astrophysical Journal*, 707:1247–1263, December 2009. ISSN 0004-637X. doi: 10.1088/0004-637X/707/2/1247. URL https://ui.adsabs.harvard.edu/abs/2009ApJ...707.1247O. ADS Bibcode: 2009ApJ...707.1247O.

Satoshi Okuzumi, Hidekazu Tanaka, Hiroshi Kobayashi, and Koji Wada. Rapid Coagulation of Porous Dust Aggregates outside the Snow Line: A Pathway to Successful Icy Planetesimal Formation. *The Astrophysical Journal*, 752:106, June 2012. ISSN 0004-637X. doi: 10.1088/0004-637X/752/2/106. URL https://ui.adsabs.harvard.edu/abs/2012ApJ...752..106O. ADS Bibcode: 2012ApJ...752..106O.

C. W. Ormel and H. H. Klahr. The effect of gas drag on the growth of protoplanets - Analytical expressions for the accretion of small bodies in laminar disks. *Astronomy & Astrophysics*, 520:A43, September 2010. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201014903. URL https://www.aanda.org/articles/aa/abs/2010/12/aa14903-10/aa14903-10.html. Publisher: EDP Sciences.

C. W. Ormel and M. Spaans. Monte Carlo Simulation of Particle Interactions at High Dynamic Range: Advancing beyond the Googol. *The Astrophysical Journal*, 684:1291–1309, September 2008. ISSN 0004-637X. doi: 10.1086/590052. URL http://adsabs.harvard.edu/abs/2008ApJ...684.1291O.

C. W. Ormel, M. Spaans, and A. G. G. M. Tielens. Dust coagulation in protoplanetary disks: porosity matters. *Astronomy and Astrophysics, Volume 461, Issue 1, January I 2007, pp.215-232*, 461(1):215, January 2007. ISSN 0004-6361. doi: 10.1051/0004-6361:20065949. URL https://ui.adsabs.harvard.edu/abs/2007A%26A..461..215O/abstract.

C. W. Ormel, C. P. Dullemond, and M. Spaans. Accretion among preplanetary bodies: The many faces of runaway growth. *Icarus*, 210(1):507–538, November 2010. ISSN

0019-1035. doi: 10.1016/j.icarus.2010.06.011. URL http://www.sciencedirect.com/science/article/pii/S0019103510002332.

Chris W. Ormel, Beibei Liu, and Djoeke Schoonenberg. Formation of TRAPPIST-1 and other compact systems. *Astronomy and Astrophysics*, 604:A1, July 2017. ISSN 0004-6361. doi: 10.1051/0004-6361/201730826. URL https://ui.adsabs.harvard.edu/abs/2017A&A...604A...10. ADS Bibcode: 2017A&A...604A...10.

Matt Pharr and William R. Mark. ispc: A SPMD compiler for high-performance CPU programming. In *2012 Innovative Parallel Computing (InPar)*, pages 1–13, May 2012. doi: 10.1109/InPar.2012.6339601.

P. Pinilla, T. Birnstiel, L. Ricci, C. P. Dullemond, A. L. Uribe, L. Testi, and A. Natta. Trapping dust particles in the outer regions of protoplanetary disks. *Astronomy and Astrophysics*, 538:A114, February 2012. ISSN 0004-6361. doi: 10.1051/0004-6361/201118204. URL https://ui.adsabs.harvard.edu/abs/2012A&A...538A.114P/abstract.

Posit Working Group. Standard for Posit Arithmetic (2022). *Posit Working Group*, March 2022.

R. R. Rafikov. Planet Migration and Gap Formation by Tidally Induced Shocks. *The Astrophysical Journal*, 572:566–579, June 2002. ISSN 0004-637X. doi: 10.1086/340228. URL https://ui.adsabs.harvard.edu/abs/2002ApJ...572..566R. ADS Bibcode: 2002ApJ...572..566R.

W. K. M. Rice, G. Lodato, J. E. Pringle, P. J. Armitage, and I. A. Bonnell. Accelerated planetesimal growth in self-gravitating protoplanetary discs. *Monthly Notices of the Royal Astronomical Society*, 355:543–552, December 2004. ISSN 0035-8711. doi: 10.1111/j.1365-2966.2004.08339.x. URL https://ui.adsabs.harvard.edu/abs/2004MNRAS.355..543R. ADS Bibcode: 2004MNRAS.355..543R.

Sheldon M. Ross. *Introduction to Probability Models (Eleventh Edition)*. Academic Press, Boston, January 2014. ISBN 978-0-12-407948-9. doi: 10.1016/B978-0-12-407948-9.00013-X. URL https://doi.org/10.1016/C2012-0-03564-8.

V. S. Safronov. The Growth of TerrestriaL Planets. *Problems of Cosmogeny*, 6:71, April 1964. URL https://ui.adsabs.harvard.edu/abs/1964PrCmg...6...71S. ADS Bibcode: 1964PrCmg...6...71S.

V. S. Safronov. *Evolution of the protoplanetary cloud and formation of the earth and planets.* January 1972. URL https://ui.adsabs.harvard.edu/abs/1972epcf.book.....S. Publication Title: Evolution of the protoplanetary cloud and formation of the earth and planets ADS Bibcode: 1972epcf.book.....S.

Viktor Sergeevich Safronov. *Evoliutsiia doplanetnogo oblaka.* January 1969. URL https://ui.adsabs.harvard.edu/abs/1969edo..book.....S. Publication Title: 1969 ADS Bibcode: 1969edo..book.....S.

Raazesh Sainudiin and Thomas York. Auto-validating von Neumann rejection sampling from small phylogenetic tree spaces. *Algorithms for molecular biology: AMB*, 4:1, January 2009. ISSN 1748-7188. doi: 10.1186/1748-7188-4-1.

Aaron David Schneider and Bertram Bitsch. How drifting and evaporating pebbles shape giant planets. II. Volatiles and refractories in atmospheres. *Astronomy & Astrophysics, Volume 654, id.A72, <NUMPAGES>11</NUMPAGES> pp.*, 654:A72, October 2021. ISSN 0004-6361. doi: 10.1051/0004-6361/202141096. URL https://ui.adsabs.harvard.edu/abs/2021A%26A...654A..72S/abstract.

Aaron David Schneider and Bertram Bitsch. How drifting and evaporating pebbles shape giant planets (Corrigendum). *Astronomy & Astrophysics, Volume 659, id.C3, <NUMPAGES>3</NUMPAGES> pp.*, 659:C3, March 2022. ISSN 0004-6361. doi: 10.1051/0004-6361/202141096e. URL https://ui.adsabs.harvard.edu/abs/2022A%26A...659C...3S/abstract.

Andreas Schärtl. Unums and Posits: A Replacement for IEEE 754 Floating Point? *M.Sc. thesis*, page 111, 2021.

Minoru Sekiya and Isamu K. Onishi. Two Key Parameters Controlling Particle Clumping Caused by Streaming Instability in the Dead-zone Dust Layer of a Protoplanetary Disk. *The Astrophysical Journal*, 860:140, June 2018. ISSN 0004-637X. doi: 10.3847/1538-4357/aac4a7. URL https://ui.adsabs.harvard.edu/abs/2018ApJ...860..140S. ADS Bibcode: 2018ApJ...860..140S.

N. I. Shakura and R. A. Sunyaev. Black Holes in Binary Systems: Observational Appearances. 55:155, January 1973a. URL https://ui.adsabs.harvard.edu/abs/1973IAUS...55..155S. Conference Name: X- and Gamma-Ray Astronomy ADS Bibcode: 1973IAUS...55..155S.

N. I. Shakura and R. A. Sunyaev. Black holes in binary systems. Observational appearance. *Astronomy and Astrophysics*, 24:337–355, January 1973b. ISSN 0004-6361. URL https://ui.adsabs.harvard.edu/abs/1973A&A....24..337S. ADS Bibcode: 1973A&A....24..337S.

S. Shima, K. Kusano, A. Kawano, T. Sugiyama, and S. Kawahara. The super-droplet method for the numerical simulation of clouds and precipitation: a particle-based and probabilistic microphysics model coupled with a non-hydrostatic model. *Quarterly Journal of the Royal Meteorological Society*, 135(642):1307–1320, 2009. ISSN 1477-870X. doi: 10.1002/qj.441. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/qj.441. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qj.441.

Jacob B. Simon, Philip J. Armitage, Rixin Li, and Andrew N. Youdin. The Mass and Size Distribution of Planetesimals Formed by the Streaming Instability. I. The Role of Self-gravity. *The Astrophysical Journal*, 822:55, May 2016. ISSN 0004-637X. doi: 10.3847/0004-637X/822/1/55. URL https://ui.adsabs.harvard.edu/abs/2016ApJ...822...55S. ADS Bibcode: 2016ApJ...822...55S.

M. V. Smoluchowski. Three reports about Diffusion, Brownian Molecule movement and Coagulation of Colloidal Particles. *Physikalisch Zeitschrift*, 17:557–571, 1916. URL https://ci.nii.ac.jp/naid/10026664191/.

Sebastian M. Stammler and Tilman Birnstiel. DustPy: A Python Package for Dust Evolution in Protoplanetary Disks. *The Astrophysical Journal*, 935(1):35, August 2022.

ISSN 0004-637X. doi: 10.3847/1538-4357/ac7d58. URL https://dx.doi.org/10.3847/1538-4357/ac7d58. Publisher: The American Astronomical Society.

Sebastian M. Stammler, Joanna Drążkowska, Til Birnstiel, Hubert Klahr, Cornelis P. Dullemond, and Sean M. Andrews. The DSHARP Rings: Evidence of Ongoing Planetesimal Formation? *The Astrophysical Journal*, 884(1):L5, October 2019. ISSN 2041-8205. doi: 10.3847/2041-8213/ab4423. URL https://doi.org/10.3847%2F2041-8213%2Fab4423. Publisher: American Astronomical Society.

Sebastian Markus Stammler, Tilman Birnstiel, Olja Panić, Cornelis Petrus Dullemond, and Carsten Dominik. Redistribution of CO at the location of the CO ice line in evolving gas and dust disks. *Astronomy and Astrophysics*, 600:A140, April 2017. ISSN 0004-6361. doi: 10.1051/0004-6361/201629041. URL https://ui.adsabs.harvard.edu/abs/2017A&A...600A.140S. ADS Bibcode: 2017A&A...600A.140S.

Glen R. Stewart and Shigeru Ida. Velocity Evolution of Planetesimals: Unified Analytical Formulas and Comparisons with N-Body Simulations. *Icarus*, 143(1):28–44, January 2000. ISSN 0019-1035. doi: 10.1006/icar.1999.6242. URL http://www.sciencedirect.com/science/article/pii/S0019103599962425.

Sun Microsystems, Inc. C++ Interval Arithmetic Programming Reference, 2001. Forte Developer 6 update 2 (Sun WorkShop 6 update 2).

Hidekazu Tanaka, Taku Takeuchi, and William R. Ward. Three-Dimensional Interaction between a Planet and an Isothermal Gaseous Disk. I. Corotation and Lindblad Torques and Planet Migration. *The Astrophysical Journal*, 565:1257–1274, February 2002. ISSN 0004-637X. doi: 10.1086/324713. URL https://ui.adsabs.harvard.edu/abs/2002ApJ...565.1257T. ADS Bibcode: 2002ApJ...565.1257T.

Hidekazu Tanaka, Youhei Himeno, and Shigeru Ida. Dust Growth and Settling in Protoplanetary Disks and Disk Spectral Energy Distributions. I. Laminar Disks. *The Astrophysical Journal*, 625:414–426, May 2005. ISSN 0004-637X. doi: 10.1086/429658. URL https://ui.adsabs.harvard.edu/abs/2005ApJ...625..414T. ADS Bibcode: 2005ApJ...625..414T.

Richard Teague, Jaehan Bae, Edwin A. Bergin, Tilman Birnstiel, and Daniel Foreman-Mackey. A Kinematical Detection of Two Embedded Jupiter-mass Planets in HD 163296. *The Astrophysical Journal*, 860:L12, June 2018. ISSN 0004-637X. doi: 10.3847/2041-8213/aac6d7. URL https://ui.adsabs.harvard.edu/abs/2018ApJ...860L..12T. ADS Bibcode: 2018ApJ...860L..12T.

L. Trapman, A. Miotello, M. Kama, E. F. van Dishoeck, and S. Bruderer. Far-infrared HD emission as a measure of protoplanetary disk mass. *Astronomy and Astrophysics*, 605:A69, September 2017. ISSN 0004-6361. doi: 10.1051/0004-6361/201630308. URL https://ui.adsabs.harvard.edu/abs/2017A&A...605A..69T. ADS Bibcode: 2017A&A...605A..69T.

B. A. Trubnikov. Solution of the coagulation equations in the case of a bilinear coefficient of adhesion of particles. *Sov. Phys. Dokl.*, 16:124–126, 1971. URL https://ci.nii.ac.jp/naid/10006214919/.

D. Turrini, F. Marzari, D. Polychroni, and L. Testi. Dust-to-gas Ratio Resurgence in Circumstellar Disks Due to the Formation of Giant Planets: The Case of HD 163296. *The Astrophysical Journal*, 877:50, May 2019. ISSN 0004-637X. doi: 10.3847/1538-4357/ab18f5. URL https://ui.adsabs.harvard.edu/abs/2019ApJ...877...50T. ADS Bibcode: 2019ApJ...877...50T.

Orkan M. Umurhan, Paul R. Estrada, and Jeffrey N. Cuzzi. Streaming Instability in Turbulent Protoplanetary Disks. *The Astrophysical Journal*, 895:4, May 2020. ISSN 0004-637X. doi: 10.3847/1538-4357/ab899d. URL https://ui.adsabs.harvard.edu/abs/2020ApJ...895....4U. ADS Bibcode: 2020ApJ...895....4U.

Simon Unterstrasser, Fabian Hoffmann, and Marion Lerch. Collection/aggregation algorithms in Lagrangian cloud microphysical models: rigorous evaluation in box model simulations. *Geoscientific Model Development*, 10(4):1521–1548, April 2017. ISSN 1991-959X. doi: 10.5194/gmd-10-1521-2017. URL https://gmd.copernicus.org/articles/10/1521/2017/. Publisher: Copernicus GmbH.

V. Urpin and A. Brandenburg. Magnetic and vertical shear instabilities in accretion discs. *Monthly Notices of the Royal Astronomical Society*, 294:399–406, March 1998. ISSN 0035-8711. doi: 10.1046/j.1365-8711.1998.01118.x. URL https://ui.adsabs.harvard.edu/abs/1998MNRAS.294..399U. ADS Bibcode: 1998MNRAS.294..399U.

Maike Voelkel. Testing the Kanagawa Models in Connection with Dust Traps Using Hydrodynamic Simulations. *B.Sc. thesis*, 2022.

Oliver Voelkel, Rogerio Deienno, Katherine Kretke, and Hubert Klahr. Linking planetary embryo formation to planetesimal formation - II. The effect of pebble accretion in the terrestrial planet zone. *Astronomy & Astrophysics*, 645:A132, January 2021a. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/202039245. URL https://www.aanda.org/articles/aa/abs/2021/01/aa39245-20/aa39245-20.html. Publisher: EDP Sciences.

Oliver Voelkel, Rogerio Deienno, Katherine Kretke, and Hubert Klahr. Linking planetary embryo formation to planetesimal formation - I. The effect of the planetesimal surface density in the terrestrial planet zone. *Astronomy & Astrophysics*, 645:A131, January 2021b. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/202039214. URL https://www.aanda.org/articles/aa/abs/2021/01/aa39214-20/aa39214-20.html. Publisher: EDP Sciences.

Karl Wahlberg Jansson and Anders Johansen. Formation of pebble-pile planetesimals. *Astronomy and Astrophysics*, 570:A47, October 2014. ISSN 0004-6361. doi: 10.1051/0004-6361/201424369. URL https://ui.adsabs.harvard.edu/abs/2014A&A...570A..47W. ADS Bibcode: 2014A&A...570A..47W.

Karl Wahlberg Jansson, Anders Johansen, Mohtashim Bukhari Syed, and Jürgen Blum. The Role of Pebble Fragmentation in Planetesimal Formation. II. Numerical Simulations. *The Astrophysical Journal*, 835:109, January 2017. ISSN 0004-637X. doi: 10.3847/1538-4357/835/1/109. URL https://ui.adsabs.harvard.edu/abs/2017ApJ...835..109W. ADS Bibcode: 2017ApJ...835..109W.

S. Watanabe, M. Arakawa, M. Hirabayashi, S. Sugita, W. F. Bottke, and P. Michel. Exploration-Based Reconstruction of Planetesimals. 534:993, July 2023. URL https://ui.adsabs.harvard.edu/abs/2023ASPC..534..993W. Conference Name: Astronomical Society of the Pacific Conference Series ADS Bibcode: 2023ASPC..534..993W.

S. J. Weidenschilling. Aerodynamics of solid bodies in the solar nebula. *Monthly Notices of the Royal Astronomical Society*, 180(2):57–70, September 1977a. ISSN 0035-8711. doi: 10.1093/mnras/180.2.57. URL https://doi.org/10.1093/mnras/180.2.57.

S. J. Weidenschilling. The distribution of mass in the planetary system and solar nebula. *Astrophysics and Space Science*, 51(1):153–158, September 1977b. ISSN 1572-946X. doi: 10.1007/BF00642464. URL https://doi.org/10.1007/BF00642464.

S. J. Weidenschilling. Dust to planetesimals: Settling and coagulation in the solar nebula. *Icarus*, 44:172–189, October 1980. ISSN 0019-1035. doi: 10.1016/0019-1035(80)90064-0. URL https://ui.adsabs.harvard.edu/abs/1980Icar...44..172W. ADS Bibcode: 1980Icar...44..172W.

S. J. Weidenschilling. The Origin of Comets in the Solar Nebula: A Unified Model. *Icarus*, 127:290–306, June 1997. ISSN 0019-1035. doi: 10.1006/icar.1997.5712. URL http://adsabs.harvard.edu/abs/1997Icar..127..290W.

Stuart J. Weidenschilling. Stirring of a planetesimal swarm: The role of distant encounters. *Icarus*, 80(1):179–188, July 1989. ISSN 00191035. doi: 10.1016/0019-1035(89)90166-8. URL https://linkinghub.elsevier.com/retrieve/pii/0019103589901668.

G. W. Wetherill and G. R. Stewart. Accumulation of a swarm of small planetesimals. *Icarus*, 77:330–357, February 1989. ISSN 0019-1035. doi: 10.1016/0019-1035(89)90093-6. URL https://ui.adsabs.harvard.edu/abs/1989Icar...77..330W. ADS Bibcode: 1989Icar...77..330W.

G. W. Wetherill and G. R. Stewart. Formation of Planetary Embryos: Effects of Fragmentation, Low Relative Velocity, and Independent Variation of Eccentricity and Inclination. *Icarus*, 106:190–209, November 1993. ISSN 0019-1035. doi: 10.1006/icar.1993.1166. URL https://ui.adsabs.harvard.edu/abs/1993Icar..106..190W. ADS Bibcode: 1993Icar..106..190W.

George W. Wetherill. Comparison of analytical and physical modeling of planetesimal accumulation. *Icarus*, 88(2):336–354, December 1990. ISSN 0019-1035. doi: 10.1016/0019-1035(90)90086-O. URL https://www.sciencedirect.com/science/article/pii/001910359090086O.

F. L. Whipple. Radial Pressure in the Solar Nebula as Affecting the Motions of Planetesimals. *NASA Special Publication*, 319:355, January 1973. URL https://ui.adsabs.harvard.edu/abs/1973NASSP.319..355W. ADS Bibcode: 1973NASSP.319..355W.

F. Windmark, T. Birnstiel, C. Güttler, J. Blum, C. P. Dullemond, and Th Henning. Planetesimal formation by sweep-up: how the bouncing barrier can be beneficial to growth. *Astronomy & Astrophysics*, 540:A73, April 2012. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201118475. URL https://www.aanda.org/articles/aa/abs/2012/04/aa18475-11/aa18475-11.html. Publisher: EDP Sciences.

A. Wolszczan and D. A. Frail. A planetary system around the millisecond pulsar PSR1257 + 12. *Nature*, 355(6356):145–147, January 1992. ISSN 1476-4687. doi: 10.1038/355145a0. URL https://www.nature.com/articles/355145a0. Number: 6356 Publisher: Nature Publishing Group.

Gerhard Wurm and Jens Teiser. Understanding planet formation using microgravity experiments. *Nature Reviews Physics*, 3(6):405–421, June 2021. ISSN 2522-5820. doi: 10.1038/s42254-021-00312-7. URL https://www.nature.com/articles/s42254-021-00312-7. Number: 6 Publisher: Nature Publishing Group.

Tetsuo Yamamoto, Toshihiko Kadono, and Koji Wada. An Examination of Collisional Growth of Silicate Dust in Protoplanetary Disks. *The Astrophysical Journal*, 783:L36, March 2014. ISSN 0004-637X. doi: 10.1088/2041-8205/783/2/L36. URL https://ui.adsabs.harvard.edu/abs/2014ApJ...783L..36Y. ADS Bibcode: 2014ApJ...783L..36Y.

Chao-Chin Yang, Anders Johansen, and Daniel Carrera. Concentrating small particles in protoplanetary disks through the streaming instability. *Astronomy and Astrophysics*, 606:A80, October 2017. ISSN 0004-6361. doi: 10.1051/0004-6361/201630106. URL https://ui.adsabs.harvard.edu/abs/2017A&A...606A..80Y. ADS Bibcode: 2017A&A...606A..80Y.

Andrew N. Youdin and Jeremy Goodman. Streaming Instabilities in Protoplanetary Disks. *The Astrophysical Journal*, 620:459–469, February 2005. ISSN 0004-637X. doi: 10.1086/426895. URL https://ui.adsabs.harvard.edu/abs/2005ApJ...620..459Y. ADS Bibcode: 2005ApJ...620..459Y.

Ke Zhang, Geoffrey A. Blake, and Edwin A. Bergin. Evidence of Fast Pebble Growth Near Condensation Fronts in the HL Tau Protoplanetary Disk. *The Astrophysical Journal*, 806:L7, June 2015. ISSN 0004-637X. doi: 10.1088/2041-8205/806/1/L7. URL https://ui.adsabs.harvard.edu/abs/2015ApJ...806L...7Z. ADS Bibcode: 2015ApJ...806L...7Z.

A. Zsom and C. P. Dullemond. A representative particle approach to coagulation and fragmentation of dust aggregates and fluid droplets. *Astronomy & Astrophysics*, 489(2):931–941, October 2008. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361:200809921. URL https://www.aanda.org/articles/aa/abs/2008/38/aa09921-08/aa09921-08.html. Number: 2 Publisher: EDP Sciences.

A. Zsom, C. W. Ormel, C. Güttler, J. Blum, and C. P. Dullemond. The outcome of protoplanetary dust growth: pebbles, boulders, or planetesimals? - II. Introducing the bouncing barrier. *Astronomy & Astrophysics*, 513:A57, April 2010. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/200912976. URL https://www.aanda.org/articles/aa/abs/2010/05/aa12976-09/aa12976-09.html. Publisher: EDP Sciences.

A. Zsom, C. W. Ormel, C. P. Dullemond, and T. Henning. The outcome of protoplanetary dust growth: pebbles, boulders, or planetesimals? - III. Sedimentation driven coagulation inside the snowline. *Astronomy & Astrophysics*, 534:A73, October 2011a. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201116515. URL https://www.aanda.org/articles/aa/abs/2011/10/aa16515-11/aa16515-11.html. Publisher: EDP Sciences.

A. Zsom, Zs Sándor, and C. P. Dullemond. The first stages of planet formation in binary systems: how far can dust coagulation proceed? *Astronomy & Astrophysics*, 527:A10, March 2011b. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201015434. URL https://www.aanda.org/articles/aa/abs/2011/03/aa15434-10/aa15434-10.html. Publisher: EDP Sciences.

This document was typeset using the XeLaTeX typesetting system, the *memoir* class created by Peter Wilson, and a template created by Federico Maggi and improved by Vel. The body text is set with Aldus Nova in 11pt. Other fonts include Palatino for titles, Consolas for monospace type, Futura for sans-serif type, DejaVu Sans for labelling in figures, and Segoe UI for labelling in illustrations. Vector graphics were prepared with Microsoft PowerPoint and with the Asymptote vector graphics language.