Computational Cost Optimization via Stochastic Kernel Sampling for the Numerical Integration of the Smoluchovski Coagulation Equation

Vincent C. Mader

June 20, 2023

Contents

List of Figures

List of Tables

1 Preface

1.1 Acknowledgements

1.2 Abstract

What is the topic of this thesis?

- We want to `test whether` the numerical `integration` of the Smoluchovski `coagulation equation` can be `sped up` via `stochastic sampling` of the kernel.
- Depending on how accurate one wants to model the involved processes, the integration of the coagulation equation can become quite costly (from a computational standpoint).
- This is due to the "high dimensionality" of the problem.
    - 3 dimensions of space (in principle)
    - 3 dimensions of mass (explain why -> fragmentation)
    - X dimensions of porosity
    - + maybe other attributes of the dust particles
- Therefore, the idea is that a big gain in computational performance could be gained by only including the "relevant" parts of the kernel into the integration. ("relevant" according to what criterion?)
- Finding out whether this works (and by `how much` the computational cost can be reduced) is the big goal of this thesis.

2 Definition of a simplified Model for Dust Coagulation

2.1 Ensemble Approach

2.1.1 QUES How can particle collisions be modeled?

["ensemble approach"]

[...]

- Particle collisions can be modeled using the Smoluchovski equation.

[...]

2.1.2 QUES What is a particle distribution function?

- Consider a distribution of particles.
- Let the particles have various different properties, e.g. position, velocity, and mass.
- Assume these properties to *not* be distributed homogenously among the particles.
- The distribution can then be described by a fitting *particle distribution function*

$$f(...) = ...$$

`QUES`: Continue writing this section.

- What is a particle distribution function *generally*?

2.1.3 QUES What is the dust particle distribution function?

`QUES`: Apply the definition of a general particle distribution function (from above) to the (special) case of a protoplanetary disk.

- We assumed the relevant properties to be
  - radial position in the disk (radial distance from star)
  - height above the midplane
  - particles mass
- These properties (in general) depend on time.

Therefore, we can write the distribution function for dust particles inside the modeled PPD as

$$f(r, z, m, t)$$

- NOTE:
  - After this section, focus (for now) on naive/simple particle *mass distribution function.*

2.1.4 QUES Assumptions needed for the (naive/simple) particle mass distribution.

Consider a distribution of dust particles with different masses m.

- Let them all have the same material density ρ_s (and other properties).
- Also, (for now) assume that their shape is always spherical.

Then, each dust particle is characterized solely by its mass, and hence its radius a:

$$m = \frac{4\pi}{3}\rho_s a^3$$

2.1.5 QUES What is the dust particle mass distribution function?

The particle mass distribution function $n(m)$ is defined in such a way that the integral

$$N = \int_0^\infty n(m)\ \mathrm{d}m$$

is the total number of dust particles per unit volume.

Following from that, the integral

$$\rho = \int_0^\infty m \cdot n(m)\ \mathrm{d}m$$

gives the total dust mass per unit volume, i.e. the dust density.

2.1.6 QUES What happens when particles collide? (from standpoint of particle mass distr.)

Collisions can be regarded as two-body collisions:

- The occurrence of three-body interactions is so tiny that it can be neglected entirely (negligible).

Outcomes:

1. bouncing
2. merging
3. fragmentation

Bouncing:

- Here: The only property we trace is the particle mass.
- Therefore, "nothing happens".

Merging:

- The two original particles with masses m_1 and m_2 "disappear".
- A single new particle with mass $m_{\text{tot}} = m_1 + m_2$ "appears".

Fragmentation:

- The two original particles with masses m_1 and m_2 also "disappear".
- But: A whole series/range of particles can appear, with a total mass of $m_{\text{tot}} = m_1 + m_2$

The input to a collision event are therefore always two bodies, but the output can be any number of particles.

2.2 The Dust Particle Mass Distribution Function

- Continuous
- Let $n(m)$ be the particle mass distribution function, i.e. the number of particles per unit volume that possess a mass m.
- Total number of particles per unit volume:

$$N = \int_0^\infty n(m)\ \mathrm{d}m$$

- Total mass per unit volume: (mass density)

$$\rho = \int_0^\infty m \cdot n(m)\ \mathrm{d}m$$

- Total disk mass: (correct?)

$$m_D = \int_0^{2\pi} \int_0^\infty \int_{-\infty}^\infty \rho\ \mathrm{d}\phi\ \mathrm{d}r\ \mathrm{d}z$$

2.2.1 Initialization

1. Dirac-delta Together with the normalization criterion from above, the Dirac-delta distribution δ_D is defined by the condition

$$\delta_D(x) = 0 \; \forall x \neq 0$$

Assuming all particles have a starting mass m_0, we can write the initial discrete mass distribution function as

$$N_i = \delta_D(m_i - m_0)$$

2. Gaussian The Gaussian distribution f_{Gauss} is given by

$$f_{\text{Gauss}}(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

Here, μ labels the "expectation value" around which the distribution is centered, and σ^2 is the variance distribution's variance ("width").

Assuming a Gaussian state in the beginning of the simulation, the initial discrete mass distribution function can then be written as

$$N_i = f_{\text{Gauss}}(m_i)$$

3. Flat (homogenous) Choose N_i in such a way that

$$N_i = C \quad \forall i$$

with a constant C.

The normalization condition leads to

$$C = \frac{1}{\mathcal{N}_m}$$

and thus also

$$N_i = \frac{1}{\mathcal{N}_m}$$

4. "True Flat" Choose N_i in such a way that

$$m_i \cdot N_i = C \quad \forall i \in [0, \mathcal{N}_m - 1) \cap \mathbb{N}$$

with a constant $C \in \mathbb{R}$.

This leads to

$$N_i = \frac{C}{m_i}$$

Before we can construct the discretized initial particle density distribution function though, we need to find out what C is. A constraint for this is given by the normalization condition

$$1 = \sum_{i=0}^{\mathcal{N}_m - 1} N_i$$

Plugging in and rearranging for C leads to

$$C = \frac{1}{\sum_{i=0}^{\mathcal{N}_m - 1} \frac{1}{m_i}}$$

2.2.2 Normalization

All of the following distributions (here labeled f) are normalized in such a way that the condition

$$\int_{-\infty}^{\infty} f(x)\ \mathrm{d}x = 1$$

holds. This then assures that the total disk mass

$$\sum_{i=0}^{\mathcal{N}_m - 1} N_i = 1$$

is equal to unity (bounded, not too large, easy conservation-checking). [Is this correct?]

1. NOTE (delete later)
   - x labels the values m_i of the discretized mass grid.
   - $f(x)$ labels the discretized particle mass distribution function

$$N_i = \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} n(m)\ \mathrm{d}m$$

2.3 The Smoluchovski Coagulation Equation

2.3.1 QUES What is the Smoluchovski coagulation equation?

The general time evolution of the particle mass distribution function due to collision can (by virtue of the two-body nature) be expressed using the Smoluchovski coagulation equation.

... background ... [1916 Smoluchovski]

The coagulation equation reads

$$\frac{\mathrm{d}n}{\mathrm{d}t}(m,t) = \int_0^\infty \int_0^\infty K(m,m',m'') \cdot n(m',t) \cdot n(m'',t)\ \mathrm{d}m'\ \mathrm{d}m''$$

Here, $K(m,m',m'')$ labels the "so-called" *coagulation kernel*.

This general form of the (temporal) evolution equation of the particle mass distribution function encompasses all possible collision outcomes (bouncing, merging, fragmentation).

The Smoluchowski coagulation equation is a (nonlinear?) integro-differential equation. It is used to model the evolution of system in which particles undergo aggregation.

2.3.2 QUES What is the coagulation kernel?

- NOTE:
  - Stay very general/"vague" here.
  - Details will follow in sections on stick-and-hit / fragmentation.
- First problem to look at:
  - simple case of Stick-and-Hit coagulation
- But before we do that, we need to talk about discretization...
  - discretization of mass axis
  - discretization of dust particle (mass) distribution function
  - discretization of kernel
  - discretization of time axis
- ...and about numerical integration of the Smoluchovski equation
  - for that, we need to talk about discretization of the time axis

2.3.3 Smoluchovski Coagulation Equation

A commonly-used method of formulating a general description for the temporal evolution of a protoplanetary disk's particle mass distribution is given by the Smoluchovski coagulation equation:

$$\frac{\partial n(m,t)}{\partial t} = \int_0^\infty \int_0^\infty K(m, m_1, m_2) \cdot n(m_1, t) \cdot n(m_2, t) \; \mathrm{d}m_1 \; \mathrm{d}m_2$$

Here, $K(m, m_1, m_2)$ labels the *coagulation kernel*, into which all information on the collisional dynamics is encoded. It summarizes all aspects of the collisions.

- integro-differential equation, population balance equation
- introduced by M.V. Smoluchovski in 1916
- can be used to determine the temporal evolution of the number density per mass $n(m)$ of particles as they coagulate.

3 Discretization of the Dust Coagulation Model

Due to the discrete nature of number representation on digital computers, it is necessary for numerical treatment to move away from a continuous representation. Instead, a discrete analogon has to be constructed for the mass grid, the evolution equations, as well as the coagulation kernel.

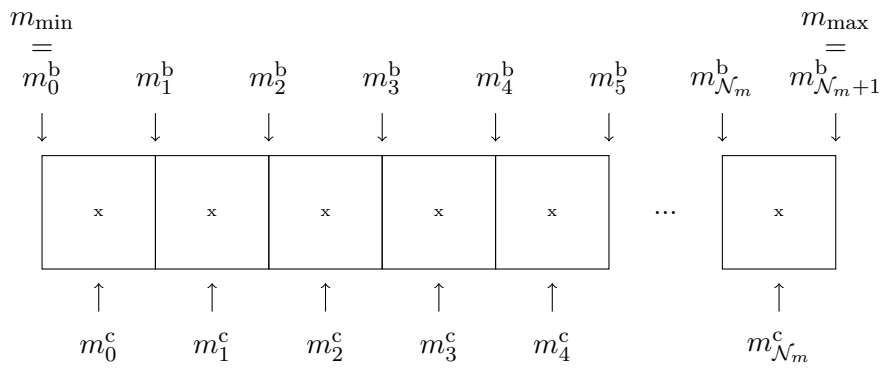
3.1 Discretization of the Mass Axis

3.1.1 ~~Introduction/General Idea~~

We will start with the discretization of the mass axis: Here, the goal is to partition the continuous range of masses present in the disk into a set of $\mathcal{N}_m \in \mathbb{N}$ intervals, which from now on will be referred to as mass grid "bins". Each of these bins can be uniquely characterized by an index $i \in [0, \mathcal{N}_m - 1]$, and its corresponding mass value m_i^{c}, situated at the center of the bin (ergo the superfix "c").

To derive an expression for these mass values, let us first define the upper and lower boundaries of the discrete mass grid. We shall label them $m_{\max}$ and $m_{\min}$, respectively. Now, consider a collection of $\mathcal{N}_m + 1$ appropriately spaced grid points m_i^{b}, which will serve as the bin boundaries.

Figure 1: Discretized mass axis.



3.1.2 Discretization of the Mass Axis using a Linear Scale

The spacing of these grid points of course depends heavily on the utilized scaling, which for simplicity shall be assumed to be linear at the moment. Later on, we will make use of a logarithmic scaling to assure a better representation of all masses along the wide range of mass values present in the disk.

Analogously to the bin centers, the boundary points can be labeled by an index $i \in [0, \mathcal{N}_m]$. The corresponding mass value m_i^{b} can be expressed as

$$m_i^{\mathrm{b}} = m_{\min} + (m_{\max} - m_{\min}) \cdot \frac{i}{\mathcal{N}_m}$$

To calculate the the mass values for the grid points at the center of a bin, all we need to do now is take the arithmetic mean of its two neighboring boundary values, i.e.:

$$m_i^{\mathrm{c}} = \frac{m_i^{\mathrm{b}} + m_{i+1}^{\mathrm{b}}}{2}$$

Thus, after having defined only the three numbers $\mathcal{N}_m$, $m_{\min}$, and $m_{\max}$, it is possible to interpolate the values of all mass grid points sitting on the boundaries and/or centers of the bins.

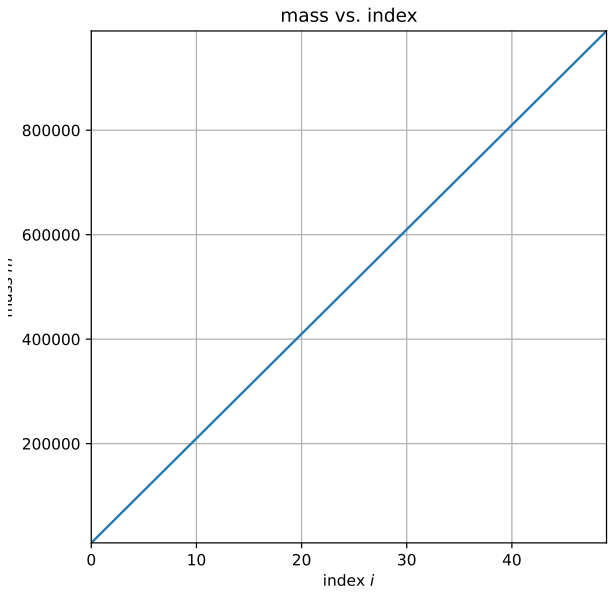
The inverse transformation from mass to index can be determined easily by rearranging the above relation. This leads to

$$i(m) = \mathcal{N}_m \cdot \frac{m - m_{\min}}{m_{\max} - m_{\min}}$$

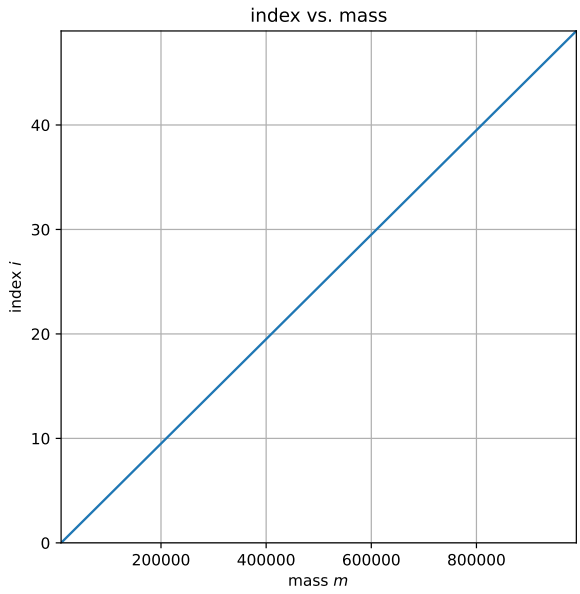
-> TODO Conversion: Mass -> Index

Sidenote: In the linearily scaled mass grid, the bin "width" is constant, and independent of the bin index. This will change once the switch to a logarithmic grid is made! For now though, it is given by

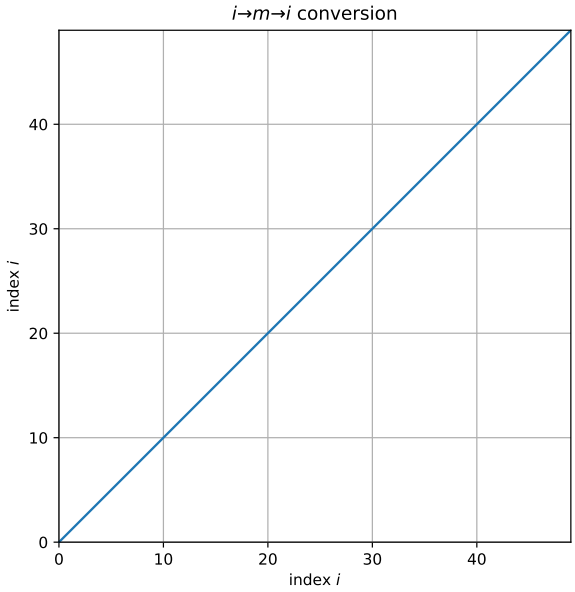
$$\Delta m = m^{\mathrm{b}}_{i+1} - m^{\mathrm{b}}_{i} = \frac{m_{\max} - m_{\min}}{\mathcal{N}_m}$$



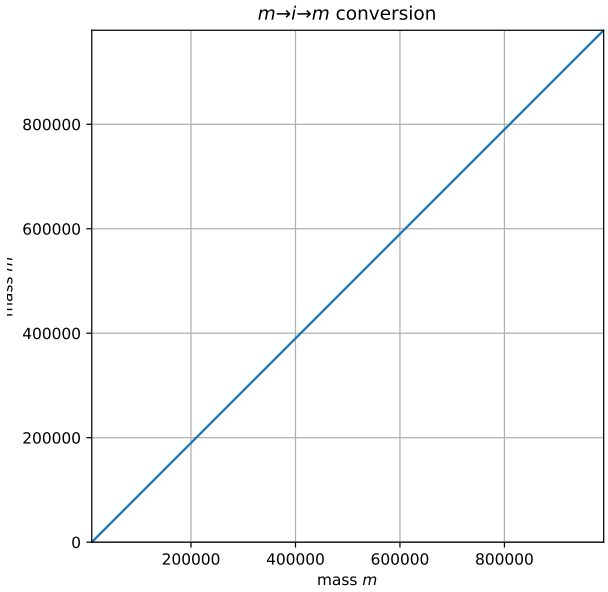
(a) CAPTION ...



(b) CAPTION ...



(c) CAPTION ...



(d) CAPTION ...

Figure 2: CAPTION ...

3.1.3 Discretization of the Mass Axis using a Logarithmic Scale

As before, let $\mathcal{N}_m$ be used to label the number of bins, and let the upper and lower grid boundaries be given by $m_{\min}$ and $m_{\max}$, respectively.

Once again, we first define an expression for the grid points sitting directly on the bin boundaries. Making use of an index $i \in [0, N_m]$, they can be uniquely identified and calculated via

$$m_i^{\mathrm{b}} = m_{\min} \cdot \left(\frac{m_{\max}}{m_{\min}} \right)^{i/\mathcal{N}_m}$$

To arrive at the mass values at the bin centers, we again take the mean. Here, it's not the arithmetic mean though, but the geometric mean, i.e.

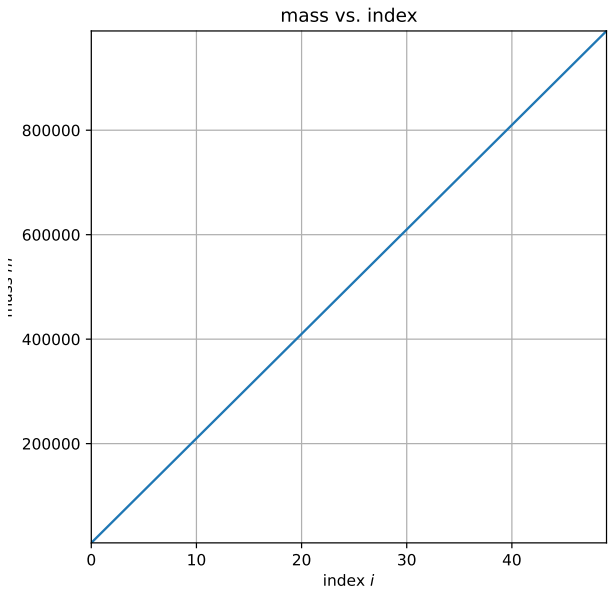
$$m_i^{\mathrm{c}} = \sqrt{m_i \cdot m_{i+1}}$$

In contrast to the linear grid, the bin "width", i.e. the additive offset from one bin to the next, is not the same for all bins in the logarithmic grid. Instead, what stays the same for is the *relative* mass increase from one bin to the next.

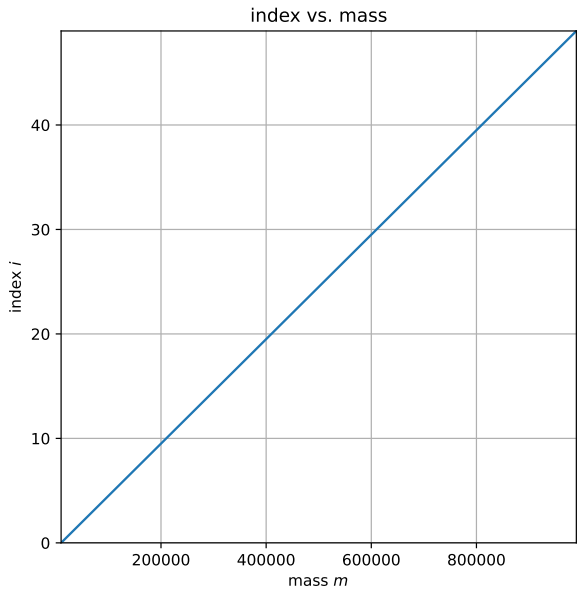
As above in the linear case, the inverse transformation can be arrived at by rearranging for i. This leads to

$$i(m) = \mathcal{N}_m \cdot \frac{\log(m) - \log(m_{\min})}{\log(m_{\max}) - \log(m_{\min})}$$

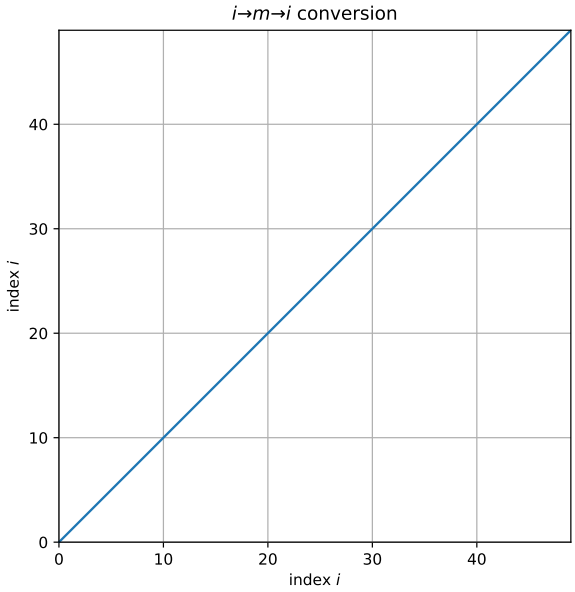
-> TODO Conversion: Mass -> Index



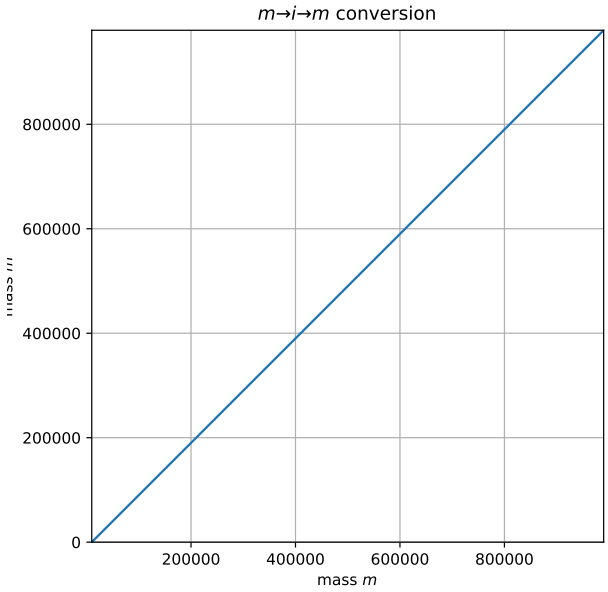
(a) CAPTION ...



(b) CAPTION ...



(c) CAPTION ...



(d) CAPTION ...

Figure 3: CAPTION ...

3.2 Discretization of the Particle Mass Distribution Function

3.2.1 QUES How to discretize the dust particle mass distribution function?

$$n_i := n(m_i)$$

To be more specific, the current time value should be included into the notation as well. We define:

$$n_{i,\xi} := n(m_i, t_\xi)$$

3.2.2 Discretization

Just as we discretized the range of masses, so we also have to discretize the particle mass distribution function. For this, we define:

$$n_i := n(m_i)$$

The number of particles per unit unit volume in bin i is then calculated by integration over the particle mass distribution function from the lower to the upper bin boundary:

$$N_i = \int_{m_{i-1/2}}^{m_{1+1/2}} n(m)\ \mathrm{d}m$$

The mass density ρ_i in the bin i is given by

$$\rho_i = \int_{m_{i-1/2}}^{m_{1+1/2}} m \cdot n(m)\ \mathrm{d}m$$

To arrive at the total mass density independent of particle mass, one has to sum over all bins:

$$\rho = \sum_{i=0}^{\mathcal{N}_m - 1} \rho_i$$

3.3 Discretization of the Smoluchovski Coagulation Equation

3.3.1 QUES How to discretize the Smoluchovski equation?

With $i, j \in \mathbb{N}$ "being indices":

$$\frac{\mathrm{d}n_k}{\mathrm{d}t} = \sum_{i=1}^{\mathcal{N}_m} \sum_{j=1}^{\mathcal{N}_m} K_{kij} \cdot n_i \cdot n_j$$

Equivalent formulation: [which one to use? I like the lower one <3]

$$\frac{\mathrm{d}n_k}{\mathrm{d}t} = \sum_{i=0}^{\mathcal{N}_m - 1} \sum_{j=0}^{\mathcal{N}_m - 1} K_{kij} \cdot n_i \cdot n_j$$

3.4 Initialization of the Particle Mass Distribution Function

3.4.1 QUES How to initialize the particle mass distribution function?

Now, let us turn our attention to the initialization of the particle mass distribution function.

Options:

1. Dirac-delta distribution
2. MRN distribution

What would be best?

- MRN is more physical than Dirac-delta. (see later chapter)
- It does not *really* make that much of a difference though.
- The information about the initialization is quickly lost anyways.
- Both 1. & 2. qualitatively lead to the same results (even if not quantitatively).

Observation:

- Using the MRN distribution leads to a significant slowdown of the integration. (minutes, instead of seconds)
- This might be because I chose $m_{\max}$ to be the largest mass grid bin.
  - Do not do this!
  - This is unphysical!
  - Instead use $\approx 0.5\mu$m
- Test whether this problem persists!

3.5 Integration of the Smoluchovski Equation

3.5.1 QUES How to discretize the time axis in a linear fashion?

Let the temporal domain be divided into $\mathcal{N}_t$ discrete "bins". Then, let ξ be the index used to uniquely identify each time step, i.e.

$$\xi \in \mathbb{N} \cap [0, \mathcal{N}_m - 1]$$

When discretizing the time axis in a linear way, this leads to

$$t_\xi = t_0 + \xi \cdot \Delta t$$

Here, t_0 labels the time value at the start of the simulation (can be set to $t_0 = 0$ without loss of generality), and Δt labels the size of each time step.

The step size needs to be chosen sufficiently small as to assure stability of the integration scheme.

[...CFL criterion...] [...implicit integration...]

3.5.2 QUES What is the difference between additive/multiplicative incrementation of time

- TODO: Move?

1. Additive time-steps
   - straightforward approach
   - incrementation of time via

$$t_{n+1} = t_n + \Delta t$$

2. Multiplicative time-steps
   - more appropriate here [explain why]

$$t_{n+1} = t_n \cdot q_t$$

3.5.3 QUES How to integrate the Smoluchovski equation using an explicit Euler scheme?

As above, we use the notation $n_{i,\xi}$ to label the dust particle mass distribution function evaluated for the mass m_i at a point in time t_ξ.

The goal now is to compute the distribution function's value after one additional step in time, i.e. at the time

$$t_{\xi+1} = t_\xi + \Delta t$$

This can be done by making use of an explicit Euler integration scheme [when?]. For a sufficiently small time step Δt, we can thus arrive at an approximate expression for the distribution function's updated value via the transformation

$$
\begin{aligned}
n_{k,\xi} &\to n_{k,\xi} + \Delta t \cdot \frac{\mathrm{d}n_{k,\xi}}{\mathrm{d}t} \\
&= n_{k,\xi} + \Delta t \cdot \sum_{i=0}^{\mathcal{N}_m - 1} \sum_{j=0}^{\mathcal{N}_m - 1} K_{kij} \cdot n_i \cdot n_j
\end{aligned}
$$

If the step size is chosen too large, the method will break down due to the insufficient stability properties of the explicit Euler integration scheme.

Instead, [. . . implicit Euler/Radau. . .] [see above]

3.5.4 QUES How to discretize the time axis in a logarithmic fashion?

As the temporal domain spans over multiple orders of magnitude for the processes involved in planet formation, it makes sense to discretize the time axis in a logarithmic fashion. [write this differently?]

[. . .]

3.5.5 QUES How to integrate the Smoluchovski equation using implicit schemes?

4 Construction of the Kernel for Simple Hit-and-Stick Coagulation

4.1 Introduction, Overview, & Goals for this Section

- Here, we only take a look at the simple case of...
  - ...simple *linear* mass axis discretization.
  - ...simple *Dirac-delta* initialization.
  - ...simple *explicit Euler* integration.

4.1.1 Construction of a Kernel for Hit-and-Stick Coagulation

Assuming simple hit-and-stick coagulation, the kernel can be written as

$$K(m, m_1, m_2) = \frac{1}{2}\big[\delta(m - m_1 - m_2) - \delta(m - m_1) - \delta(m - m_2)\big] \cdot R(m_1, m_2)$$

Here, the coefficient $R(m_1, m_2)$ labels the collision-and-merging rate.

4.2 Definition of the Kernel for simple Stick-and-Hit Coagulation

4.2.1 QUES How to implement stick-and-hit coagulation on a linear mass axis?

Let us first turn our attention to the simple case of stick-and-hit coagulation: Here, two particles with masses m_1 and m_2 collide, and subsequently merge into a single particle with combined mass $m_{\rm tot} = m_1 + m_2$.

Let $C(m_1, m_2)$ be the rate at which collisions occur between such a pair of particles. Then, let

$$R(m_1, m_2$$

label the rate of collision events that lead to a merging of the two particles into a single new one.

Of course, in actuality not every collision leads to a merge event. For simplicity, we will for now will assume that it does. As such, we can write

$$R(m_1, m_2) = C(m_1, m_2)$$

Later on, we will more carefully examine how the collision+sticking rate can be calculated.

[... also talk about definition of collision rate...] [... also talk about "reaction rates"...] [... difference between "collision", "collision+merge", "reaction" rates (fragmentation?)...]

[... collision+sticking rate, sticking probability, "Dust evolution w/ binning methods" eq.1.6-1.11...]

[... move the above elsewhere? -> "collisions" section?] [...]

A kernel describing the stick-and-hit coagulation process can be written as

$$K(m, m_1, m_2) = \frac{1}{2}\Big[\delta_D(m - m_1 - m_2) - \delta_D(m - m_1) - \delta_D(m - m_2)\Big] \cdot R(m_1, m_2)$$

Here, δ_D labels the Dirac delta function.

[...]

4.3 Numerical Integration with the Explicit Euler Scheme

4.3.1 QUES Why does the mass error increase over time? -> "Mass Outflow"

4.3.2 Hit-and-Stick Collision on Linear Mass Grid

4.3.3 Hit-and-Stick Collision on Logarithmic Mass Grid

5 Implementation of a Logarithmic Mass Axis Representation

5.1 Introduction, Overview, & Goals for this Section

5.2 Reasons for Making Use of a Logarithmic Representation

5.3 Challenges Associated with a Logarithmic Representation

5.4 Definition of the Logarithmic Mass Axis

5.4.1 QUES How to discretize the mass axis in a logarithmic fashion?

5.5 Implementation of the Kovetz-Olund algortihm

5.5.1 Reasons for the Necessity of the Kovetz-Olund Algorithm

Even in a highly simplified scenario, where only hit-and-stick coagulation is included, the definition of the kernel K_{kij} is not at all trivial. To assure both the consistency and accuracy of the algorithm, one has to take care of two separate problems, namely:

1. The conservation of mass *has* to be assured, otherwise the numerical solution can not be assumed to remain stable for long. In the case of stick-and-hit coagulation, this means that for every pair of colliding particles, a single new particle has to be created. At the same time, the two initial particles have to be removed from the distribution. During this process, the total mass should remain unaffected down to machine precision.

2. When using a logarithmically spaced grid for the discretized mass axis, it can not be assumed that after a collision of two dust particles with masses m_i and m_j the resulting particle will carry a mass $m_k = m_i + m_j$ whose value can be mapped trivially onto the grid. In general, the corresponding index will not be an integer, and instead lie between somewhere between the two neighboring grid points with indices k and $k+1$. Therefore, the result of the merging of m_i and m_j has to be divided in some sensible way between these two neighboring bins.

5.5.2 Implementation of the Kovetz-Olund Algorithm

An elegant way for solving the two problems listed above is given in the paper by [Kovetz & Olund, 1969], where they used the following procedure:

1. The stick-and-hit coagulation kernel is split into two parts. The first is the *gain* of particles in bin k due to the collision of particles from the bins i and j. The second is the *loss* of particles from bin k due to collisions of particles in bin k with particles from any other bin j.

   Using this separation into gain & loss, the dust particle mass distribution's temporal derivative can be expressed in the following form:

   $$\frac{\mathrm{d}n_k}{\mathrm{d}t} = \sum_{i,j} K_{kij}^{\mathrm{gain}} n_i n_j - \sum_j K_{kj}^{\mathrm{loss}} n_k n_j$$

   In other words, the total kernel [from above, cite eq.] can be written as

   $$K_{kij} = K_{kij}^{\mathrm{gain}} - K_{ij}^{\mathrm{loss}} \delta_{ki}$$

   Splitting the kernel like this into a gain & a loss term is a quite general approach, and can be used in more complex scenarios as well (including e.g. particle fragmentation processes).

2. For the scenario of pure hit-and-stick coagulation, a unique discretization of the kernel can be defined such that both the number of particles and the conservation of total mass are handled correctly. To do this, consider a pair of colliding particles with indices i and j. Then, let the index $\bar{k}$ be chosen in such a way that the condition

   $$m_{\bar{k}} \leq m_i + m_j < m_{\bar{k}+1}$$

   is satisfied.

3. As stated before, in hit-and-stick coagulation, a single new particle emerges for each pair of colliding particles. Using the definitions from above, this condition can be expressed as follows:

   $$K_{\bar{k},ij}^{\mathrm{gain}} + K_{\bar{k}+1,ij}^{\mathrm{gain}} \stackrel{!}{=} K_{ij}^{\mathrm{loss}}$$

4. The second condition is that of mass conservation, which can be written as:

$$m_{\bar{k}} K^{\text{gain}}_{\bar{k},ij} + m_{\bar{k}+1} K^{\text{gain}}_{\bar{k}+1,ij} \stackrel{!}{=} (m_i + m_j) K^{\text{loss}}_{ij}$$

5. Now, for the mapping of the resulting particle's mass onto two neighboring bins, let us define a parameter ε such that

$$K^{\text{gain}}_{\bar{k},ij} = K^{\text{loss}}_{ij} \cdot (1 - \varepsilon), \text{ and}$$
$$K^{\text{gain}}_{\bar{k}+1,ij} = K^{\text{loss}}_{ij} \cdot \varepsilon$$

This assures that [equation from pt 3] is satisfied. If we now plug this definition into [equation from pt 4] and solve for ε, we arrive at

$$\varepsilon := \frac{m_i + m_j - m_{\bar{k}}}{m_{\bar{k}+1} - m_{\bar{k}}}$$

This is the algorithm of [Kovetz & Olund (1969)], and it was also used in subsequent papers like [Brauer et al. (2008)] and [Birnstiel et al. (2010)].

5.6 Implementation of Near-Zero-Cancellation Handling for Coagulation

5.6.1 QUES What is Near-Zero-Cancellation and where can it occur in general?

When using floating-point numbers following the representation defined by the IEEE-754 standard, it can occur that

$$a + b = a \qquad \text{for} \qquad b \neq 0$$

Typically, this happens when

$$|b| < \varepsilon_m \cdot |a|$$

Here, ε_m labels the *machine precision*.

[Add: How big is ε_m for an f32, how big for an f64?]

5.6.2 QUES Where can Near-Zero-Cancellation occur in our model?

Let i and j once again be the indices used to label two colliding particles. Additionally, assume now that particle i is *much smaller* than particle j.

The detailed balance approach from above requires the removal of both the big and the small particle from the mass distribution, followed by the re-insertion of a new particle carrying the initial pair's combined mass. This new particle would then have a mass which is nearly identical to that of the bigger one of the original two particles, it would be only a tiny bit heavier.

In the approach defined above this would mean that $\bar{k} = j$, i.e. the resulting particle will reside in the same bin as the larger original one. Also, it would follow that $\varepsilon \ll 1$.

Let us now take a look at the particle mass distribution in the bin $\bar{k}$ and, more specifically, by how much it changes from one timestep to the next. For this particular pair of i and $\bar{k} = j$, we can write:

$$\frac{\mathrm{d}n_{\bar{k}}}{\mathrm{d}t} = K^{\text{gain}}_{\bar{k},i\bar{k}} n_i n_{\bar{k}} - K^{\text{loss}}_{\bar{k}i} n_i n_{\bar{k}}$$

Plugging in [equation from above] leads to

$$\frac{\mathrm{d}n_{\bar{k}}}{\mathrm{d}t} = (1-\varepsilon) K^{\text{loss}}_{\bar{k}i} n_i n_{\bar{k}} - K^{\text{loss}}_{\bar{k}i} n_i n_{\bar{k}}$$

Here, the two terms alsmost cancel each other out. What remains is a contribution which is proportional to ε.

If ε is small enough, the double-precision accuracy of the floating point representation will lead to breakdown of the method. [rewrite this sentence, copied almost exactly from Kees]

5.6.3 Handling Near-Zero-Cancellation for Stick-and-Hit Coagulation

It is relatively easy to identify the particle pairs (i, j) for which the scenario detailed above will occur. Let i (without loss of generality) be the index of the larger one of the two colliding masses. Cancellation may then occur when the resulting k is equal to j.

In that case, we carry out the subtraction in [previous equation] analytically, and write:

$$\frac{\mathrm{d}n_{\bar{k}}}{\mathrm{d}t} = -\varepsilon K^{\text{loss}}_{\bar{k}i} n_i n_{\bar{k}}$$

[Elaborate on this, see "Dust Evolution with Binning Methods"]

6 Inclusion of Fragmentation Processes into the Kernel

6.1 "First/Simple/Naive Attempt"

6.1.1 QUES How to implement naive fragmentation?

- "Pulverization"

6.2 The MRN Distribution

- introduced by Mathis-Rumpl-Nordsieck, 1977

6.2.1 QUES How to implement MRN distribution (for initialization)?

Particle mass distribution function is defined by the condition:

$$\rho = \int_{m_{\min}}^{m_{\max}} n(m) \cdot m \, \mathrm{d}m$$

Ansatz: power law with $q = -11/6$ (see 'sizedistributions.pdf')

$$n(m) \sim m^q$$

Choose constant A such that

$$n(m) = A \cdot m^q$$

Plug into the integral from above

$$\begin{aligned}
\rho &= \int A \cdot m^q \cdot m \, \mathrm{d}m \\
&= \int A \cdot m^{q+1} \, \mathrm{d}m \\
&= \frac{A}{q+2} \, m^{q+2} \Big|_{m_{\min}}^{m_{\max}} \\
&= \frac{A}{q+2} \, (m_{\max}^{q+2} - m_{\min}^{q+2})
\end{aligned}$$

Rearanging for A leads to

$$A = \frac{(q+2) \cdot \rho}{m_{\max}^{q+2} - m_{\min}^{q+2}}$$

Now we only need to choose values for $m_{\min}$ and $m_{\max}$.

- For the minimum mass, no lower limit exists. For it, the value corresponding to the smallest bin can be chosen [citation needed].
- For the maximum mass, an upper limit has to be chosen. It can be left as a free parameter to be set in 'config.toml', or set to something approximating 0.5 μm. [see 2004 Dullemond & Dominik] ^ max. in interstellar medium (use for $m_{\min}$?)

6.2.2 QUES How to implement MRN distribution for fragmentation?

Let m_i and m_j be the masses of two particles that are involved in a collision leading to fragmentation.

How do we model this?

- Distribute the mass $m_{\text{tot}} = m_i + m_j$ onto bins with $m < m_{\text{tot}}$.

- For this, use the MRN distribution $n(m) = Am^q$.
- For this, find the value of A.
- For this, use the formula from [Summarize: How to implement MRN distri].
- For this, find a value for $m_{\max}$.

How do we choose a value for $m_{\max}$?

- Can it be larger than $\max(m_i,\ m_j)$?
  - "Some merging, some fragmenting"
  - "1 + 1 → 1.5 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1"
- If...
  - ...yes: It can at most be equal to $m_i + m_j$
  - ...no: It can at most be equal to $\max(m_i, m_j)$
- In either case, it can't really be *equal* to either of the two values.
  - That would be stick-and-hit in case "yes", and bouncing in case "no".
  - It can at most be equal to the mass corresponding to the next-lower bin.

Answer:

- We don't really know.
- It doesn't *really* matter that much anyhow.
- One could specify the method in 'config.toml', or just use one of the methods.

6.2.3 QUES Test integration speed with MRN distribution enabled. Is there a slowdown?

6.3 "Implementation of Near-Zero-Cancellation Handling for Fragmentation" (?)

7 Definition of a the Dust Particle Reaction Rates

7.1 Definition of the Dust Particle Reaction Rates

7.2 Definition of the Dust Particle Collision Rates

7.2.1 QUES How to define the particle collision rate?

The collision rate for a pair of masses m_1 and m_2 can be written as

$$C(m_1, m_2, \Delta v) = \sigma(m_1, m_2) \cdot \langle |\vec{v}_1 - \vec{v}_2 \rangle$$

Here, $\sigma(m_1, m_2)$ labels the cross section for a collision event. The relative velocity between such two particles is given by the difference $\vec{v}_1 - \vec{v}_2$, its absolute value is $|\vec{v}_1 - \vec{v}_2|$, and $\langle |\vec{v}_1 - \vec{v}_2| \rangle$ is the expectation value of that quantity.

[Simplification was done here] [Elaborate: Give more precise formula]

In our notation:

$$C_{ij} = \sigma_{ij} \cdot \langle \Delta v_{ij} \rangle$$

7.3 Definition of the Collision Cross Section

7.3.1 QUES How to implement realistic collision cross sections?

The cross section for a collision of two particles i and j with radii a_i and a_j can be written as

$$\sigma_{ij} = \pi \cdot (a_i + a_j)^2$$

Here, the radii can be computed from the corresponding masses m_i and m_j from equation [Summarize: Assumptions needed for the (naive/simple) particle mass distribution.].

[Add plot here]

7.4 Definition of the Relative Particle Velocity

7.4.1 QUES How to implement realistic relative velocities between dust particles?

8 Calculation of the Relative Dust Particle Velocities

8.1 Definition of the Relative Velocity due to Radial Drift

$$\Delta u_{\mathrm{RD}} = \left| u_r(m_1) - u_r(m_2) \right|$$

dust radial velocity:

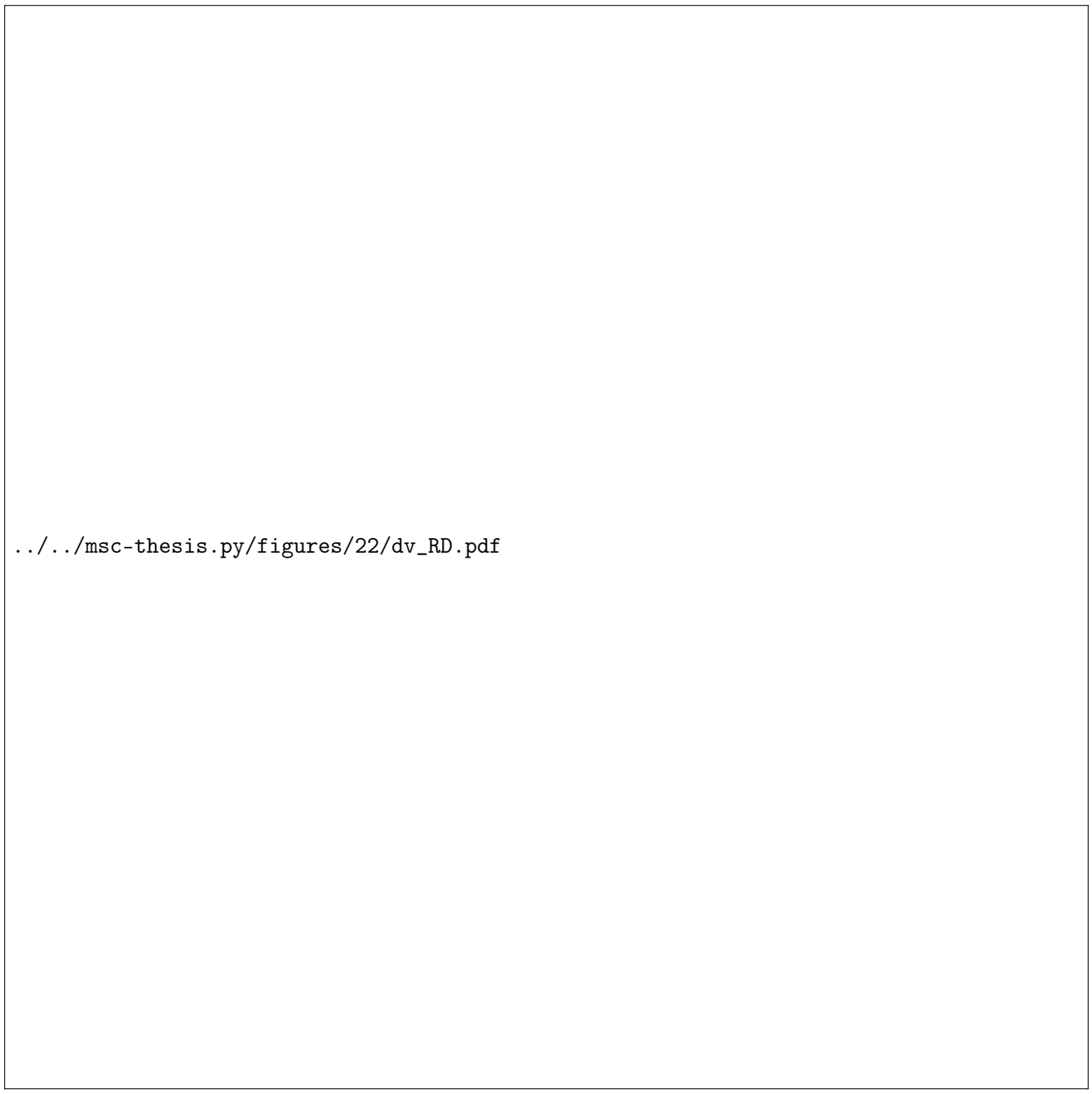
$$u_r = \frac{u_g}{1 + \mathrm{St}^2} - \frac{2u_n}{\mathrm{St} + \mathrm{St}^{-1}}$$

gas radial velocity:

$$u_g = -\frac{3}{\Sigma_g \sqrt{r}} \cdot \partial_r \left(\Sigma_g \nu_g \sqrt{r} \right)$$

particle maximum drift velocity:

$$u_n = -\frac{E_d}{2\rho_g \, \Omega_K} \cdot \frac{\partial P_g}{\partial r}$$

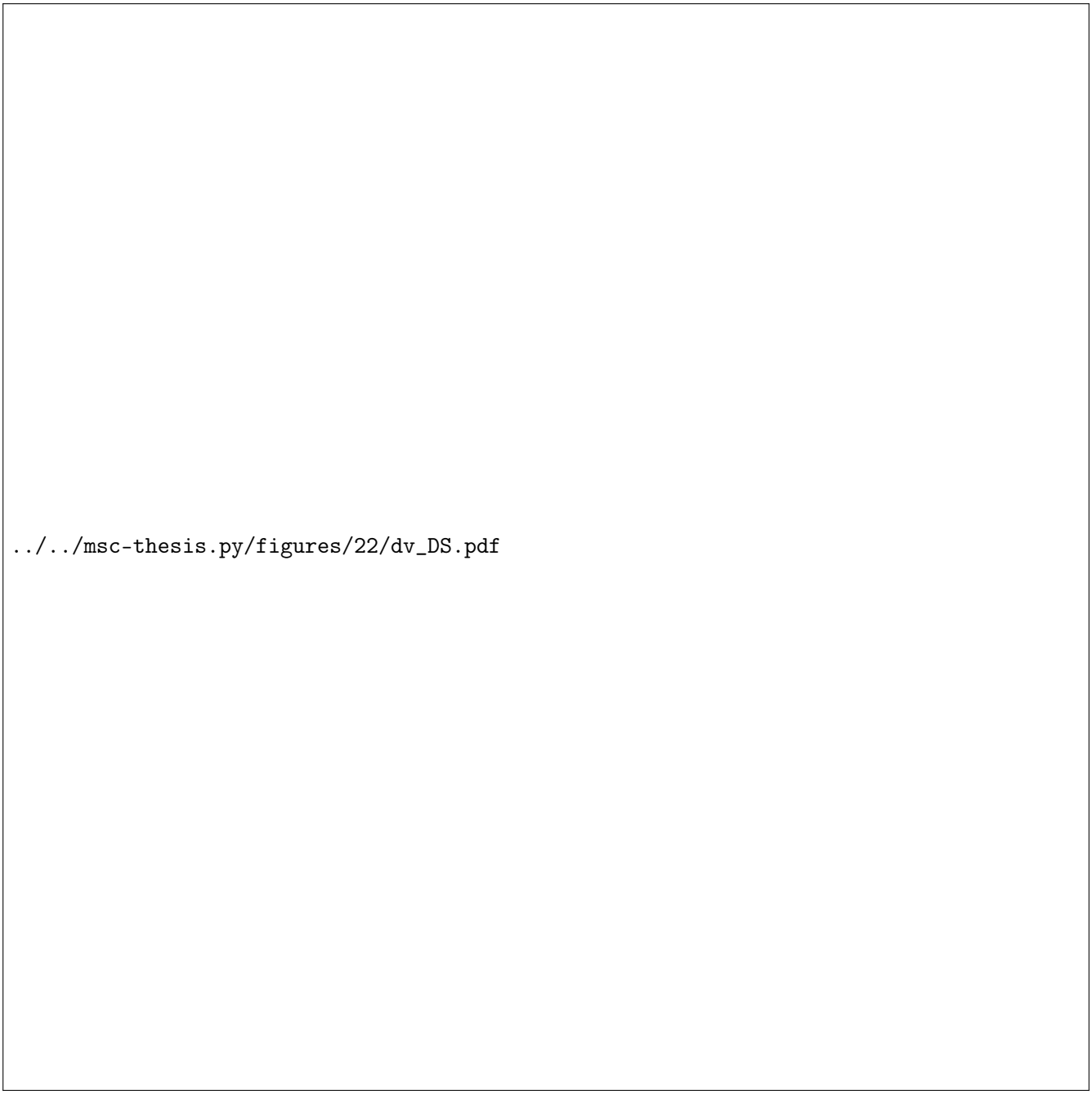

../../msc-thesis.py/figures/22/dv_RD.pdf

8.2 Definition of the Relative Velocity due to Differential Settling

$$\Delta u_{\mathrm{DS}} = \left| u_i - u_j \right|$$

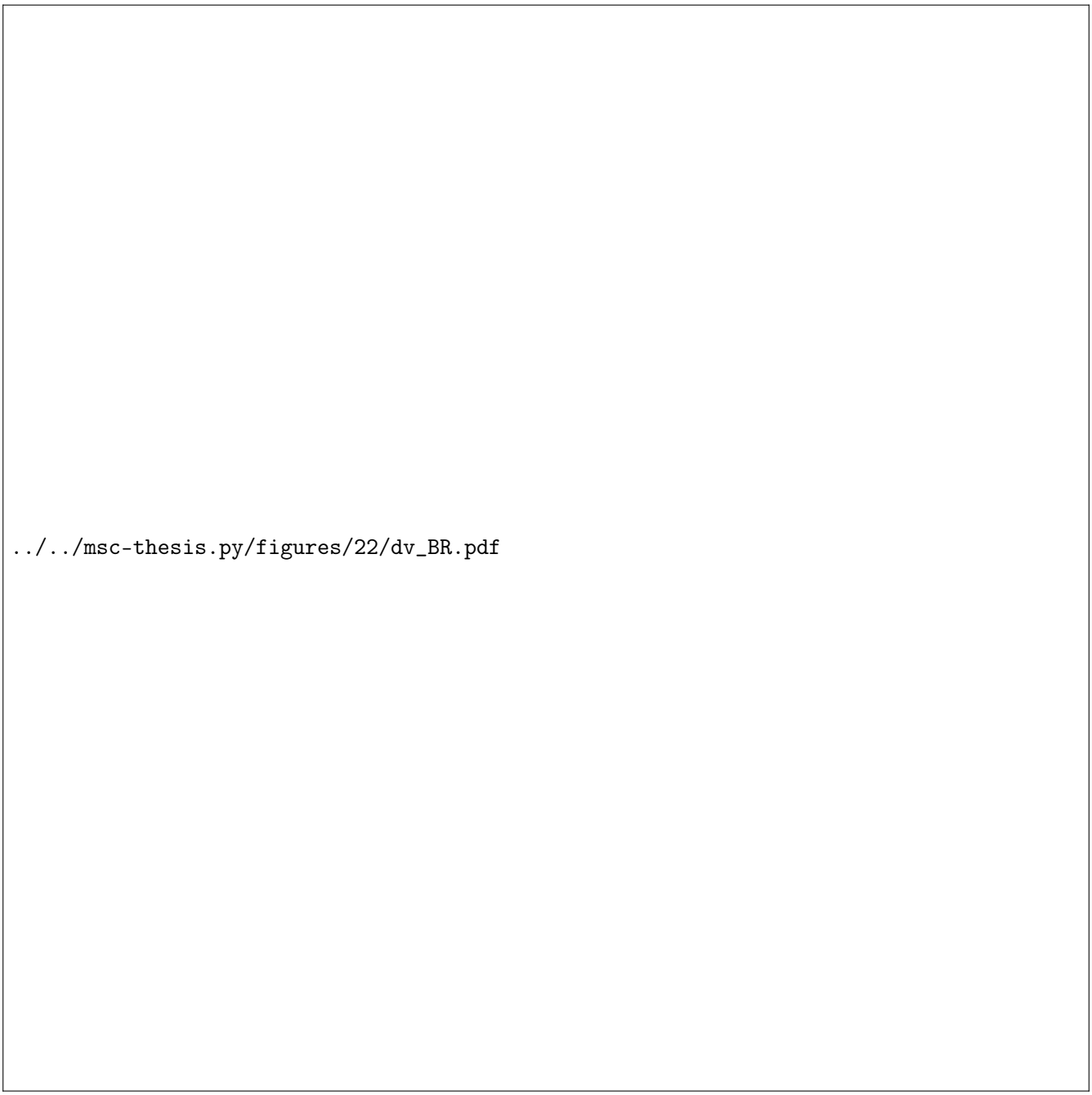
$$u_i = ts \cdot \Omega_K^2 \cdot z$$

$$\mathrm{St} = \Omega_K \cdot t_s$$

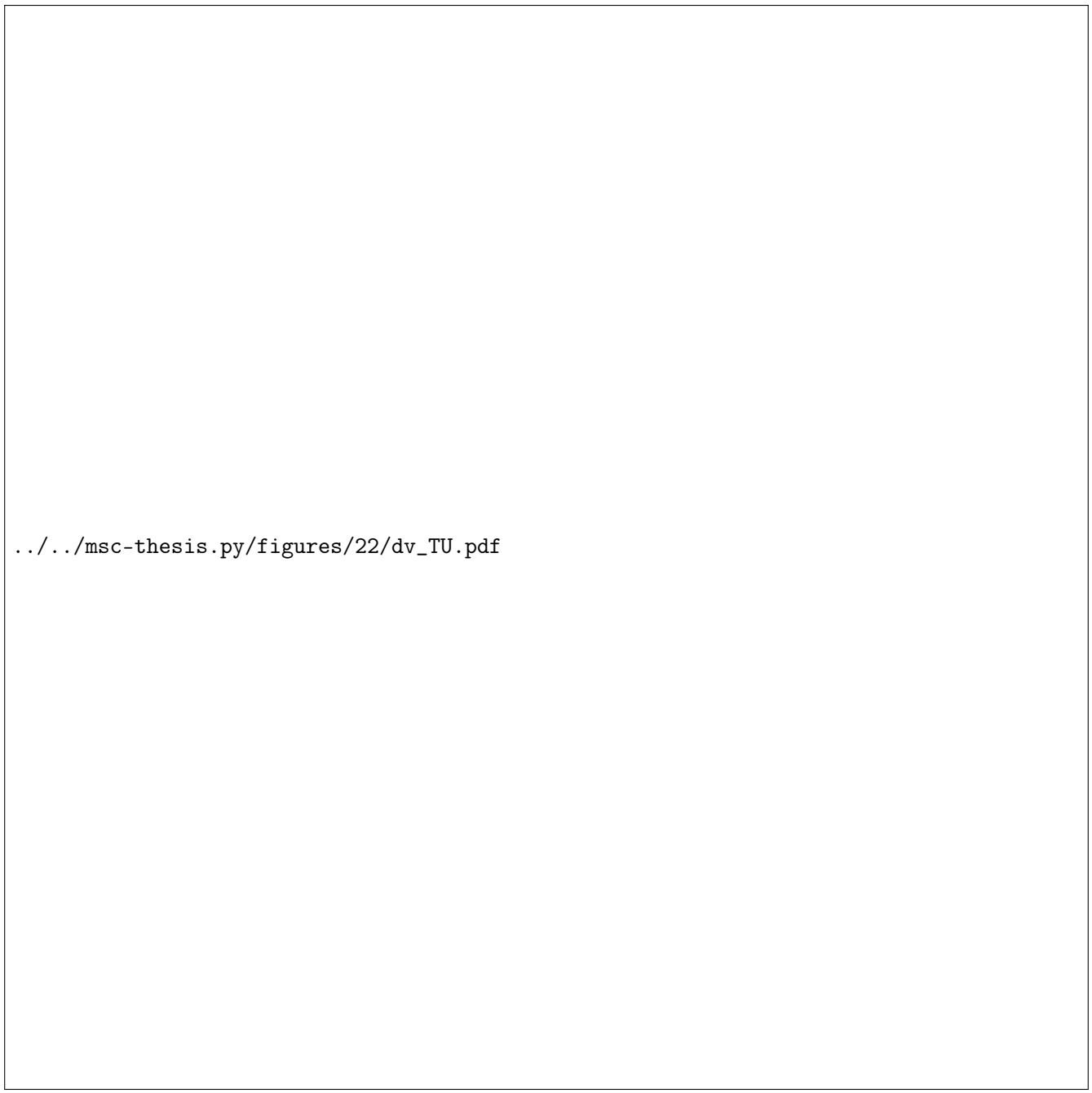


8.3 Definition of the Relative Velocity due to Brownian Motion

$$\begin{aligned}\Delta u_{ij}^{\mathrm{BR}} &= \sqrt{\frac{8k_BT}{\pi} \cdot \frac{m_i + m_j}{m_i \cdot m_j}} \\ &= \sqrt{\frac{8}{\pi\beta\mu_{ij}}}\end{aligned}$$

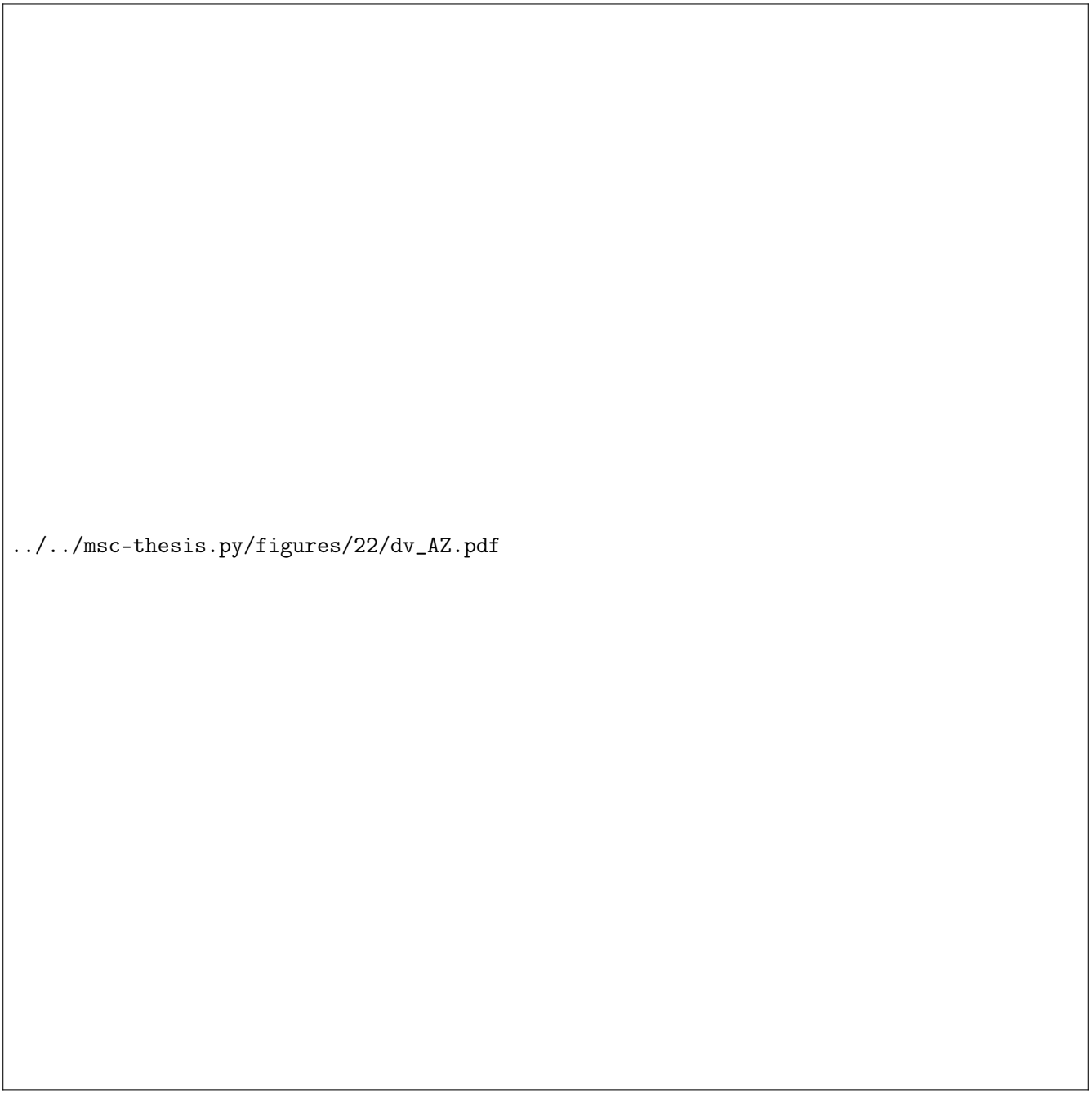


8.4 Definition of the Relative Velocity due to Turbulence



8.5 Definition of the Azimuthal Relative Velocity

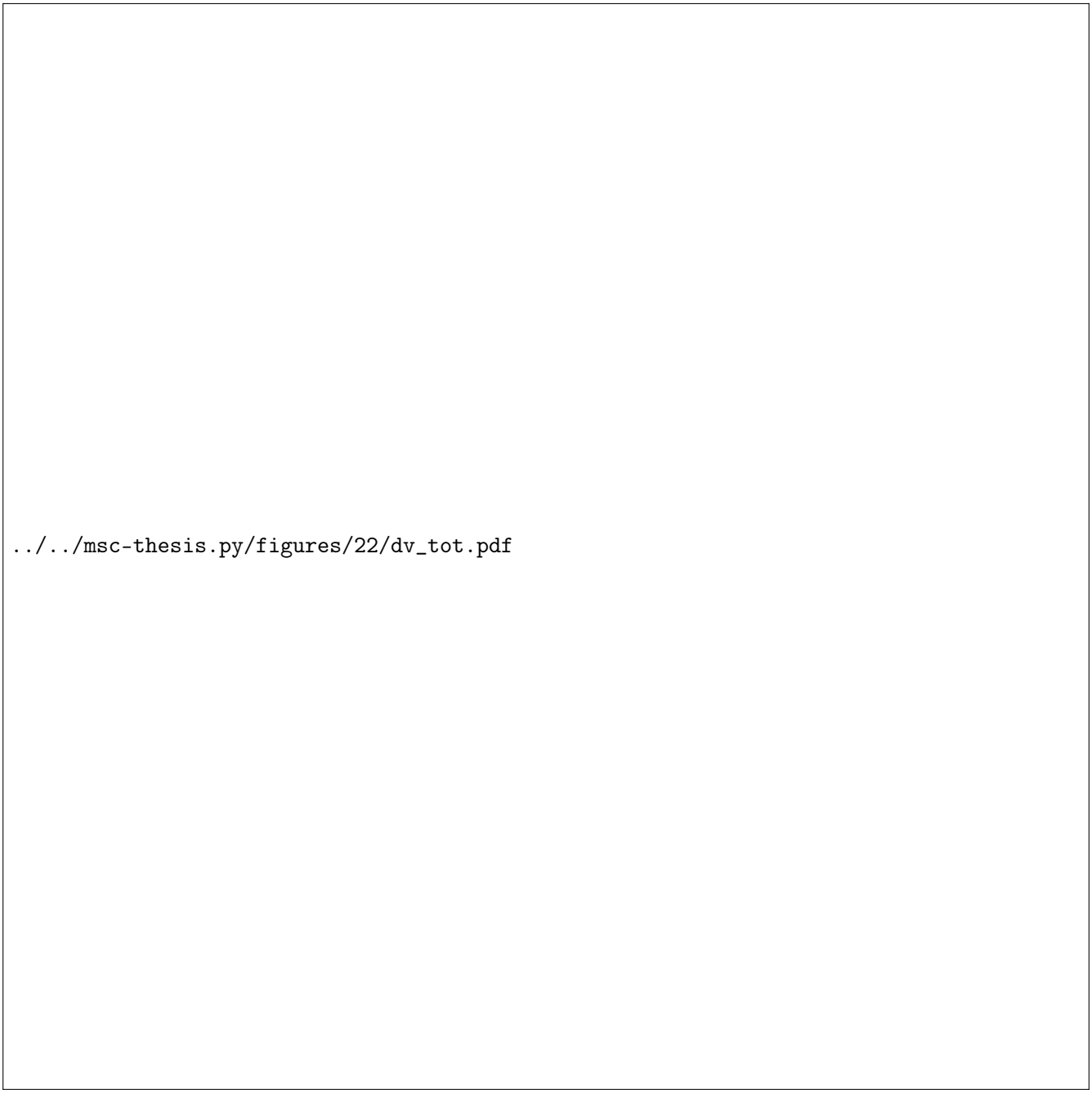
$$\Delta u_{ij} = \left| u_n \cdot \left(\frac{1}{1 + \mathrm{St}_i^2} - \frac{1}{1 + \mathrm{St}_j^2} \right) \right|$$



8.6 Total Relative Velocity

$$\Delta u_{ij} = \Delta u_{\mathrm{BR}} + \Delta u_{\mathrm{RD}} + \Delta u_{\mathrm{AZ}} + \Delta u_{\mathrm{TU}} + \Delta u_{\mathrm{DS}}$$

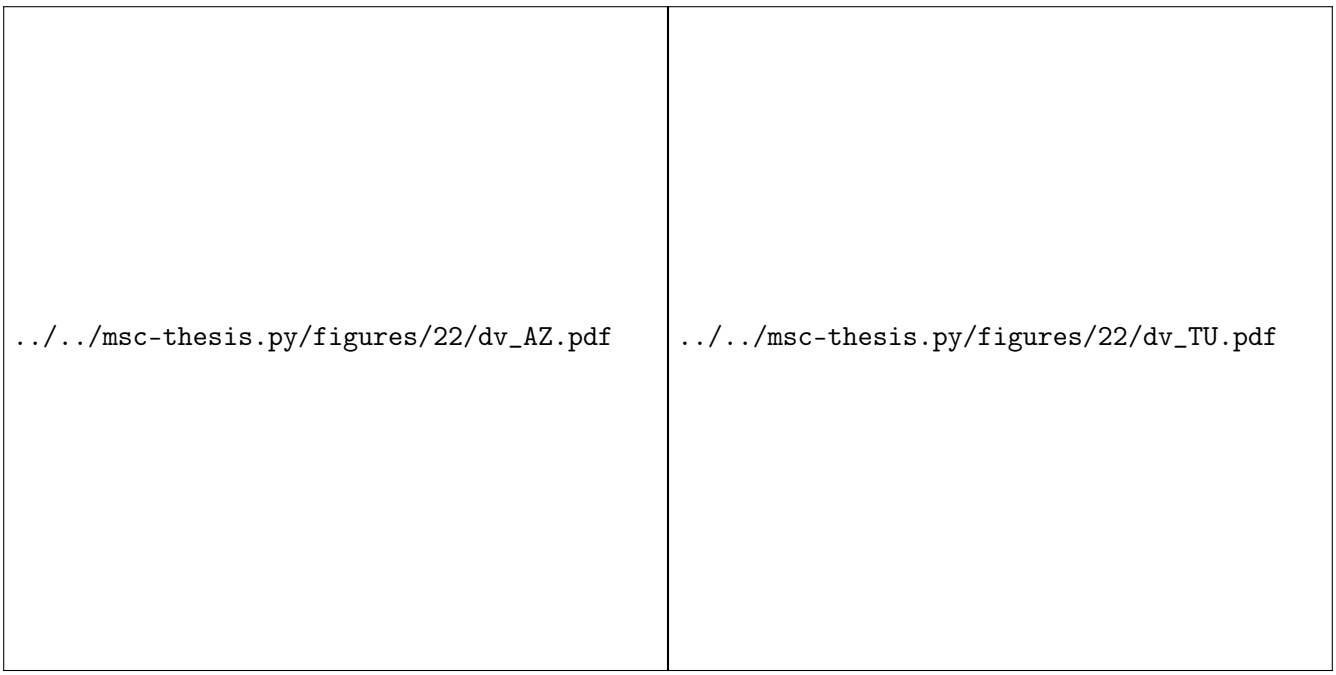
$$\Delta u_{ij} = \sqrt{\Delta u_{\mathrm{BR}}^2 + \Delta u_{\mathrm{RD}}^2 + \Delta u_{\mathrm{AZ}}^2 + \Delta u_{\mathrm{TU}}^2 + \Delta u_{\mathrm{DS}}^2}$$



9 Implemention of Stochastic Kernel Sampling

10 Summary of Results

11 Thesis Discussion & Outlook



(a) 1a (b) 1b

Figure 4: plots of....

12 Appendix

12.1 Glossary

12.2 Testing: Can I use *subfigures* for "multicol" LaTeX plots in Emacs org-mode?

- Yes! See below:

13 References