IT 11 048
Examensarbete 30 hp
Juni 2011

Accurate and Efficient Solution of the Smoluchowski Equation

Mohammad Hossein Bani-Hashemian

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Accurate and Efficient Solution of the Smoluchowski Equation

Mohammad Hossein Bani-Hashemian

The probability density function (PDF) of the relative position of molecules diffusing independently in three dimensional space according to Brownian motion and reacting with a certain probability when any two of them collide is given by the Smoluchowski equation. The PDF is used to sample particle positions in simulations of reaction-diffusion processes by particle-based simulation methods, like Green's Function Reaction Dynamics (GFRD) proposed by van Zon and ten Wolde. The GFRD algorithm is an event-driven algorithm, allowing the use of longer time steps, which is particularly efficient for simulating chemical reactions at low concentration in molecular biology. This study is based on the improved version of the GFRD algorithm developed by S. Hellander and P. Lötstedt, where the applicability of the algorithm is increased and computing the PDFs is simplified by using an operator splitting approach. The main idea is to split the spatial differential operator of the Smoluchowski equation into a radial part and an angular part, resulting in two one-dimensional time-dependent partial differential equations (PDEs) to be solved independently and sequentially. These equations can be solved analytically but the solutions are complicated and computationally expensive to evaluate. This thesis intends to compare the accuracy and efficiency of sampling the radial distance between two molecules and the relative angular position of the two molecules from directly evaluated exact PDFs or their finite difference approximations and interpolating the positions from precomputed tabulated data.

Handledare: Per Lötstedt / Stefan Hellander
Ämnesgranskare: Michael Thuné
Examinator: Anders Jansson
IT 11 048
Tryckt av: Reprocentralen ITC

Acknowledgements

My sincerest appreciation goes to my supervisor, Professor Per Lötstedt, and my assistant supervisor, Stefan Hellander, for giving me the opportunity to work with them and write this thesis in their research project. I am so grateful to them for their kind guidance, trust and especially the freedom I was granted to explore my own ideas throughout this work. Our weekly meetings and discussions were always inspiring and a great experience for me.

I would like to thank Professor Michael Thuné not only for accepting to review this manuscript and his invaluable comments, but also for being such an excellent, caring programme coordinator.

I give my warmest thanks to my family, my father and brothers, for their consistent support, patience and always having faith in me. I dedicate this thesis to them and the loving memory of my mother.

Contents

1 Introduction

Investigating the dynamical evolution of particles diffusing independently in three dimensional space according to Brownian motion has long been a topic of interest, particularly when there is some probability that any pair of them can bind reversibly, forming a new compound, when they come into contact or collide. A naturally occurring example of this phenomenon can be observed in *diffusion-influenced* biochemical reactions involving few reactant molecules in living cells. In 1917, Smoluchowski derived a time-dependent PDE, known as the *Smoluchowski equation*, that describes the time evolution of the probability density function (PDF) of the relative position of sphere-shaped particles performing a Brownian motion in a potential field, with boundary conditions in terms of the *association* and *dissociation rates* [11].

Many algorithms have been developed to perform computer simulations to get a better insight into the diffusive motion of reactants at different spatial and temporal scales [3]. At a *microscopic* level, where instead of the variation over time of the concentrations of chemical components (*macroscopic* level) or the dynamical behaviour of clusters of molecules as a whole (*mesoscopic* level), we focus on local particle-particle interactions around each individual molecule, MCell [12], ChemCell [8], Cell++ [10] and Smoldyn (Smoluchowski Dynamics) [1] are among the widely used simulators. These simulators carry out *particle-based stochastic simulations* based on *time-driven* algorithms, where the simulation time is advanced in predetermined fixed increments; as time is advanced, all the events (association-dissociation reactions) within the time step are simulated. On the contrary, in an *event-driven* simulator, time is advanced by the occurrence of events, i.e. the length of time advancement does not need to be small and constant and varies depending on when reactions occur. The main advantage of employing an event-driven technique in simulation is to avoid time steps with no reactions or ‘‘empty” time steps, by taking longer time steps which leads to significant improvement in computational efficiency of the simulation. This approach is of greater applicability in simulating chemical reactions at low concentration, where a time-driven simulator may have to simulate the environment at empty time steps more frequently.

An event-driven algorithm has been proposed by van Zon and ten Wolde, called *Green's Function Reaction Dynamics* (*GFRD*) [14, 15]. In the GFRD algorithm, by generating sufficiently small time steps, i.e. the time steps until the next collision or reaction, a complex multi-particle reaction-diffusion system is decomposed into simple subsystems: freely diffusing single molecules; monomolecular (dissociation) reactions; and bimolecular (association) reactions. At each time step, the states of the chemical subsystems are updated independently from one another using the Green's function or the probability density function solution of the Smoluchowski equation [14, 2].

Inspired by the GFRD algorithm, Hellander and Lötstedt developed a new event-driven algorithm [4], in which more flexibility to handle various types of reactions is achieved and the computations that are performed in every simulation time interval to sample new positions of molecules in subsystems according to the PDF solution of

the Smoluchowski equation, are simplified by splitting the PDE into a radial part and an angular part via an *operator splitting* or *fractional step* scheme. Each part is a one dimensional time-dependent PDE, whose analytical solution is known [6, 2]. The solutions have quite complicated analytical expressions in terms of infinite series or special functions making them intractable and computationally expensive. As an alternative approach, numerical methods can be employed to solve the PDEs.

The aim of this thesis is to directly evaluate the exact analytical PDFs for the radial distance between two molecules and the relative angular position of the two molecules, implement a finite difference method to approximate the analytical solution in the radial case, and implement position sampling techniques using the analytical (numerical) solutions or via linear interpolation between the entries of precomputed lookup tables in MATLAB and compare the sampling approaches with respect to accuracy and efficiency.

The remainder of the thesis is outlined as follows: In Section 2, the Smoluchowski equation, which we are going to solve in this work is introduced. In Section 3, first we explain how an operator splitting technique can be applied to reduce the dimension of the problem by decomposing the Smoluchowski equation into two simpler PDEs: one in the radial coordinate and the other in the angular coordinate. Afterwards, we present and develop different methods, both analytical and numerical, to solve the radial and angular parts of the Smoluchowski equation and draw random samples from the solutions. In Section 4, the accuracy and efficiency of the sampling techniques are investigated through some comparison experiments. We then present the algorithm for simulating reaction-diffusion processes using the best particle position sampling approach in Section 5 and as an example simulate a system of a pair of two molecules and verify the correctness of the algorithm. In the final section we provide a conclusion.

2 Smoluchowski Equation: A model for the diffusion dynamics of a pair of reactant molecules

Consider a reversible association/dissociation reaction of the form

$$A + B \underset{k_d}{\overset{k_a}{\rightleftharpoons}} C\,, \tag{2.1}$$

where k_a and k_d are the *association* and *dissociation rates*, respectively. We assume that the molecules are sphere-shaped and molecule A is positioned fixed at the origin of a spherical coordinate system (r, θ, ϕ) and molecule B is free to move randomly in space and time by diffusion with a *diffusion coefficient* D, which is defined to be the sum of the diffusion coefficients of A and B.

Depending on the value of k_a, A and B may immediately react with each other to form molecule C, whenever their distance is equal to the *reaction radius* σ; and backreaction may also occur with the rate k_d, which results in dissociation of C into A and B, separated by the *dissociation distance* σ. For simplicity, we further assume that the potential energy of the interaction is negligible.

Let $\mathbf{r} = (r, \theta, \phi)^{\mathbf{T}}$ represent the position vector of a point in the above mentioned coordinate system, then the time dependent conditional probability density (PDF) of the molecule B to stay at position $\mathbf{r}$ at time t, if it starts at position $\mathbf{r}_0$ at time $t_0 < t$, $p_{\mathbf{r}}(\mathbf{r}, t|\mathbf{r}_0, t_0)$, satisfies the *Smoluchowski Equation*

$$\partial_t p_{\mathbf{r}} = D\left(\frac{\partial^2 p_{\mathbf{r}}}{\partial r^2} + \frac{2}{r}\frac{\partial p_{\mathbf{r}}}{\partial r} + \frac{1}{r^2 \sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta \frac{\partial p_{\mathbf{r}}}{\partial \theta}\right) + \frac{1}{r^2 \sin^2\theta}\frac{\partial^2 p_{\mathbf{r}}}{\partial \phi^2}\right), \tag{2.2}$$

with initial condition

$$p_r(\mathbf{r}, t_0|\mathbf{r}_0, t_0) = \delta(\mathbf{r} - \mathbf{r}_0)\,, \tag{2.3}$$

and boundary conditions

$$\lim_{r\to\infty} p_{\mathbf{r}}(r, \theta, \phi, t|\mathbf{r}_0, t_0) = 0\,, \tag{2.4}$$

$$4\pi\sigma^2 D \frac{\partial p_{\mathbf{r}}}{\partial r}\bigg|_{r=\sigma} = k_a p_{\mathbf{r}}(r = \sigma, t|\mathbf{r}_0, t_0) - k_d(1 - S(t|\mathbf{r}_0, t_0))\,, \tag{2.5}$$

where S is the *survival probability*, or with other words the probability that the system remains unchanged and no reaction occurs between the molecules until time t and is defined as follows:

$$S(t|\mathbf{r}_0, t_0) = 1 - p_r(*, t|\mathbf{r}_0, t_0)\,; \tag{2.6}$$

$p_r(*, t|\mathbf{r}_0, t_0)$ denotes the probability of the molecules being bound together (in the form of molecule C) at time t and is computed via

$$p_{\mathbf{r}}(*, t|\mathbf{r}_0, t_0) = \int_{t_0}^{t} 4\pi\sigma^2 D \frac{\partial p_{\mathbf{r}}(\mathbf{r}, \tau|\mathbf{r}_0, t_0)}{\partial r}\bigg|_{r=\sigma} d\tau\,. \tag{2.7}$$

The survival probability S can be equivalently expressed by the following formula that we occasionally use

$$S(t|\mathbf{r}_0, t_0) = \int_\sigma^\infty 4\pi\rho^2 p_{\mathbf{r}}(\rho, t|\mathbf{r}_0, t_0)\, d\rho\,. \tag{2.8}$$

Note that in the case of irreversible reaction, $k_d = 0$, the boundary condition (2.5) is simplified as

$$4\pi\sigma^2 D \frac{\partial p_{\mathbf{r}}}{\partial r}\bigg|_{r=\sigma} = k_a p_{\mathbf{r}}(r = \sigma, t|\mathbf{r}_0, t_0)\,, \tag{2.9}$$

that for $k_a = 0$ turns out to be a homogeneous Neumann boundary condition at $r = \sigma$ and as $k_a \to \infty$, we have $p_{\mathbf{r}}(r = \sigma, t|\mathbf{r}_0, t_0) = 0$.

3 Solution of the Smoluchowski equation and position sampling techniques

Our approach to solve the time-dependent PDE (2.2), subject to the initial condition (2.3) and boundary conditions (2.4)-(2.5), is based on a *first order operator splitting* or *fractional step* technique [1], where we decompose the differential operator in the right hand side of equation (2.2) into two parts :

(i) a *radial* part, with derivatives only in the radial direction:

$$\partial_t p_r = D\left(\frac{\partial^2 p_r}{\partial r^2} + \frac{2}{r}\frac{\partial p_r}{\partial r}\right), \tag{3.1}$$

with initial condition

$$p_r(r, t_0|r_0, t_0) = \delta(r - r_0)\,, \tag{3.2}$$

and boundary conditions

$$\lim_{r\to\infty} p_r(r, t_0|r_0, t_0) = 0\,, \tag{3.3}$$

$$4\pi\sigma^2 D\frac{\partial p_r}{\partial r}\bigg|_{r=\sigma} = k_a p_r(\sigma, t|r_0, t_0) - k_d(1 - S(t|r_0, t_0))\,, \tag{3.4}$$

where

$$S(t|r_0, t_0) = 1 - \int_{t_0}^{t} 4\pi\sigma^2 D\frac{\partial p_r(r, \tau|r_0, t_0)}{\partial r}\bigg|_{r=\sigma} d\tau\,; \tag{3.5}$$

and

(ii) an *angular* part, with derivatives in the polar and azimuthal directions:

$$\partial_t p_\theta = \frac{D}{r^2}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial p_\theta}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2 p_\theta}{\partial\phi^2}\right), \tag{3.6}$$

with $\theta \in [0, \pi]$, $\phi \in [0, 2\pi)$ and initial condition

$$p_\theta(\theta, \phi, t_0|r, 0, \phi_0, t_0) = \delta(\theta)\,. \tag{3.7}$$

Each of the parts or subproblems can be treated separately, either analytically or numerically, which we will discuss in the following sections. Once the solution is known for all subproblems, the solution of (2.2) can be approximated by choosing a time step, Δt, and time integrating the sub-problems sequentially to advance to the next time step until a specified time is reached.

[1] Interested readers are referred to [4] for a *second order Strang splitting* method.

3.1 Solution and sampling in the radial direction

3.1.1 Analytical solution

In [6], Kim and Shin derived the following solution to Eqs. (3.1) - (3.4) based on the Green's function solution of Eq. (3.1) for the *reflecting boundaries* ($k_a = 0$ and $k_d = 0$) presented in Carslaw and Jaeger [2] :

$$\begin{aligned} p_r(r,t|r_0,t_0)4\pi r r_0\sqrt{D} = \frac{1}{\sqrt{4\pi t}}\Bigg\{ &\exp\left(-\frac{(r-r_0)^2}{4Dt}\right) + \exp\left(-\frac{(r+r_0-2\sigma)^2}{4Dt}\right)\Bigg\} \\ &+ \frac{\alpha(\gamma+\alpha)(\alpha+\beta)}{(\gamma-\alpha)(\alpha-\beta)}\, W\left(\frac{r+r_0-2\sigma}{\sqrt{4Dt}}, \alpha\sqrt{t}\right) \\ &+ \frac{\beta(\alpha+\beta)(\beta+\gamma)}{(\alpha-\beta)(\beta-\gamma)}\, W\left(\frac{r+r_0-2\sigma}{\sqrt{4Dt}}, \beta\sqrt{t}\right) \\ &+ \frac{\gamma(\beta+\gamma)(\gamma+\alpha)}{(\beta-\gamma)(\gamma-\alpha)}\, W\left(\frac{r+r_0-2\sigma}{\sqrt{4Dt}}, \gamma\sqrt{t}\right) . \end{aligned} \tag{3.8}$$

In this expression,

$$W(a,b) := \exp(2ab+b^2)\,\mathrm{erfc}(a+b)\,, \tag{3.9}$$

where $\mathrm{erfc}(x) = 1-\mathrm{erf}(x)$ is the *complementary error function*, with $\mathrm{erf}(x)$ being the *error function*, defined by $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp(-u^2)du$; and α, β and γ satisfy the following relations :

$$\alpha+\beta+\gamma = \left(1+\frac{k_a}{k_D}\right)\frac{\sqrt{D}}{\sigma}\,, \tag{3.10a}$$

$$\alpha\beta+\beta\gamma+\gamma\alpha = k_d\,, \tag{3.10b}$$

$$\alpha\beta\gamma = k_d\frac{\sqrt{D}}{\sigma}\,, \tag{3.10c}$$

where $k_D := 4\pi\sigma D$.

The Green's function solution for the final bound state, $p_r(*,t|r_0,t_0)$, is given by:

$$\begin{aligned} p_r(*,t|r_0,t_0) = \frac{k_a}{4\pi r_0\sigma\sqrt{D}}\Bigg\{ &\frac{\alpha}{(\gamma-\alpha)(\alpha-\beta)}\, W\left(\frac{r_0-\sigma}{\sqrt{4Dt}}, \alpha\sqrt{t}\right) \\ &+ \frac{\beta}{(\alpha-\beta)(\beta-\gamma)}\, W\left(\frac{r_0-\sigma}{\sqrt{4Dt}}, \beta\sqrt{t}\right) \\ &+ \frac{\gamma}{(\beta-\gamma)(\gamma-\alpha)}\, W\left(\frac{r_0-\sigma}{\sqrt{4Dt}}, \gamma\sqrt{t}\right)\Bigg\}, \end{aligned} \tag{3.11}$$

which can be used to obtain the exact survival probability via the following relation

$$S(t|r_0,t_0) = 1 - p_r(*,t|r_0,t_0)\,. \tag{3.12}$$

At a given time t, let us denote the (conditional) *cumulative distribution function* (CDF) for the probability density of particle position in the radial direction as

$$F_{p_r}(r,t|r_0,t_0) = \int_\sigma^r 4\pi\rho^2 p_r(\rho,t|r_0,t_0)\,d\rho\,, \tag{3.13}$$

then it can be shown [7] that

$$\begin{aligned} &F_{p_r}(r,t|r_0,t_0) + p_r(*,t|r_0,t_0) \\ &= -\frac{\sqrt{Dt}}{r_0\sqrt{\pi}}\left\{\exp\left(\frac{(r-r_0)^2}{4Dt}\right) - \exp\left(-\frac{(r+r_0-2\sigma)^2}{4Dt}\right)\right\} \\ &\quad + \frac{1}{2}\left\{\operatorname{erf}\left(\frac{r-r_0}{\sqrt{4Dt}}\right) + \operatorname{erf}\left(\frac{r+r_0-2\sigma}{\sqrt{4Dt}}\right)\right\} \\ &\quad + \frac{1}{r_0}\left(r-\frac{\sqrt{D}}{\alpha}\right)\frac{(\gamma+\alpha)(\alpha+\beta)}{(\gamma-\alpha)(\alpha-\beta)}\,W\left(\frac{r+r_0-2\sigma}{\sqrt{4Dt}},\alpha\sqrt{t}\right) \\ &\quad + \frac{1}{r_0}\left(r-\frac{\sqrt{D}}{\beta}\right)\frac{(\alpha+\beta)(\beta+\gamma)}{(\alpha-\beta)(\beta-\gamma)}\,W\left(\frac{r+r_0-2\sigma}{\sqrt{4Dt}},\beta\sqrt{t}\right) \\ &\quad + \frac{1}{r_0}\left(r-\frac{\sqrt{D}}{\gamma}\right)\frac{(\beta+\gamma)(\gamma+\alpha)}{(\beta-\gamma)(\gamma-\alpha)}\,W\left(\frac{r+r_0-2\sigma}{\sqrt{4Dt}},\gamma\sqrt{t}\right)\,. \end{aligned} \tag{3.14}$$

Assuming that the chemical reaction is irreversible, i.e. that $k_d = 0$, the above results are consistent with the well-known analytical solutions given in [2] :

$$\begin{aligned} p_{\rm irr}(r,t|r_0,t_0)4\pi r r_0\sqrt{D} &= \frac{1}{\sqrt{4\pi t}}\left\{\exp\left(-\frac{(r-r_0)^2}{4Dt}\right) + \exp\left(-\frac{(r+r_0-2\sigma)^2}{4Dt}\right)\right\} \\ &\quad - \alpha_{\rm irr}\,W\left(\frac{r+r_0-2\sigma}{\sqrt{4Dt}},\alpha_{\rm irr}\sqrt{t}\right)\,, \end{aligned} \tag{3.15}$$

and

$$S(t|r_0,t_0) = 1 - \left(\frac{\sigma}{r_0}\right)\frac{k_a}{k_a+k_D}\left\{\operatorname{erfc}\left(\frac{r_0-\sigma}{\sqrt{4Dt}}\right) - W\left(\frac{r_0-\sigma}{\sqrt{4Dt}},\alpha_{\rm irr}\sqrt{t}\right)\right\}, \tag{3.16}$$

where as before $k_D = 4\pi\sigma D$ and $\alpha_{\rm irr} := \left(1+\frac{k_a}{k_D}\right)\frac{\sqrt{D}}{\sigma}$.

In spite of a rather complicated form of the analytical PDF in (3.8) and the exact CDF in (3.14) the formulas can be easily implemented in MATLAB. We just need to take the fact into account that $-\alpha$, $-\beta$ and $-\gamma$ are the three roots of the cubic equation

$$\sigma x^3 + \sqrt{D}\left(1+\frac{k_a}{k_D}\right)x^2 + \sigma k_d x + \sqrt{D}\,k_d = 0\,, \tag{3.17}$$

that appears while proving (3.8) (see [6]). Note that, generally, α, β and γ can be complex numbers but the sum of the last three terms of (3.8), (3.11) and (3.14) is a real number.

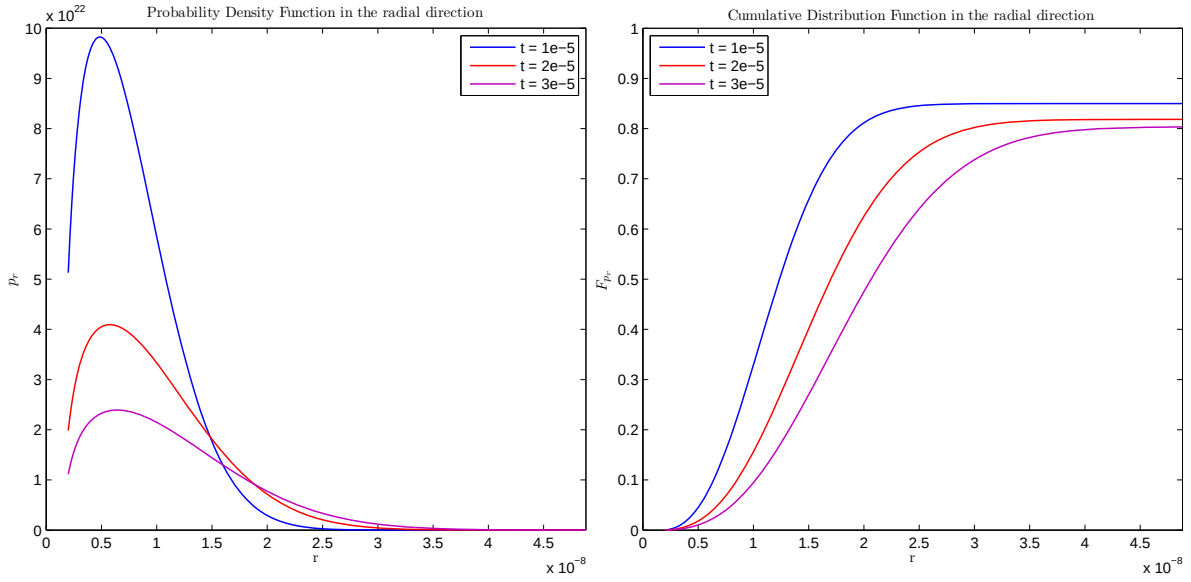
One issue in the implementation is that there exists no built-in function in MATLAB for computing the complementary error function, erfc, with a complex argument. One way to evaluate the complementary error function is through the so called *Faddeeva function* defined as $w(z) = e^{-z^2}\mathrm{erfc}(-iz)$, implemented as the MATLAB function `FADDEEVA.m` available at the MATLAB Central File Exchange [5]. Furthermore, for sufficiently large numbers, either real or complex, we use the asymptotic expansion [13]:

$$\mathrm{erfc}(z) \sim \frac{e^{-z^2}}{\sqrt{\pi}\, z} \sum_{m=0}^{\infty} (-1)^m \frac{1 \cdot 3 \cdot 5 \cdots (2m-1)}{(2z^2)^m}, \tag{3.18}$$

to avoid obtaining intermediate values.

The following figure represents the PDF and the CDF of the relative position of two reactant molecules, one of which is fixed at the origin, in the radial direction at different time instances. The slope of the CDF plot at any point r corresponds to the probability density of finding molecules at the distance r from the fixed molecule multiplied by $4\pi r^2$, as it follows from (3.13), and the difference between the maximum CDF and 1 indicates how frequently the molecules that are involved in the reaction can be found in the bound state. It can be inferred from Figure 1 that as we proceed in time or equivalently as we pick larger time steps, it is more probable that the molecules stay farther away from each other and the probability to be in the bound state decreases.

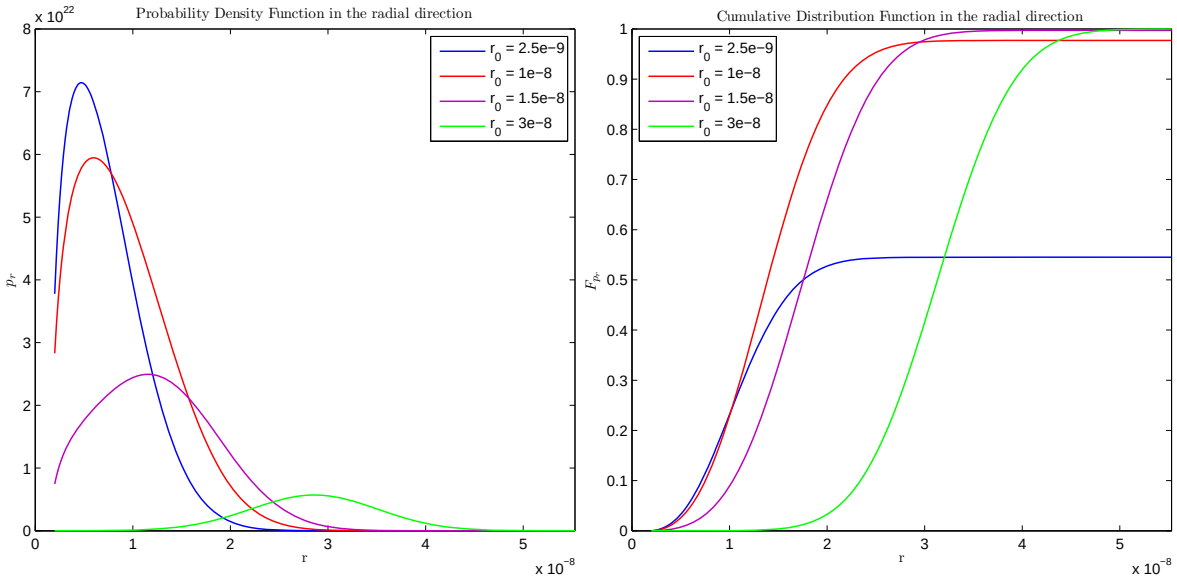
Figure 1: The PDF and CDF in the radial direction at different time steps. The model parameters are $\mathtt{k_a} = 10^{-19}$, $\mathtt{k_d} = 3$, $\mathtt{D} = 2 \times 10^{-12}$, $\sigma = 2 \times 10^{-9}$, $\mathtt{r_0} = 5 \times 10^{-9}$. These parameters are typical in applications in molecular biology in the simulations in [4].



In Figure 1, one can also see that the peak of the probability density function occurs in a small neighbourhood of r_0 (in the example $r_0 = 5 \times 10^{-9}$), which means that the

freely moving particle is more frequently distributed in a radial distance close to its initial radial position. So, as Figure 2 illustrates well, for a fixed time step t, as r_0 is increased, the concentration of molecules at the reaction radius is significantly decreased, hence fewer reactions are expected to take place.

Figure 2: The PDF and CDF in the radial direction for different values of r_0. The model parameters are $\mathtt{k_a} = 10^{-19}$, $\mathtt{k_d} = 3$, $\mathtt{D} = 2 \times 10^{-12}$, $\sigma = 2 \times 10^{-9}$, $\mathtt{t} = 10^{-5}$.



For given r_0 and t, having a discrete approximation to the CDF defined on a finite number of points in the r direction, we can sample radial distances distributed randomly according to the corresponding PDF via the following algorithm, which is essentially the same as the *inverse transform sampling algorithm* (see e.g. [9]) but with slight differences:

Data: cdf, the vector of CDF values evaluated at a grid of points in the r direction
Result: a radial distance r_ξ sampled from the probability density function p_r

$\xi \leftarrow$ *a uniformly distributed random number on the interval* $[0, 1]$;
if $\xi \leqslant$ *the maximum CDF value* **then**
  $k \leftarrow$ *index of the largest CDF value that is* $\leqslant \xi$;
  compute r_ξ by linear interpolation using two interpolating points $(\mathsf{cdf}(k), r_k)$ and $(\mathsf{cdf}(k+1), r_{k+1})$;
else
  $r_\xi \leftarrow \sigma$ `(* molecules are in the bound state, reaction will occur*)`
end

Algorithm 1: Algorithm to sample radial distances from the PDF p_r

Before we continue with constructing a numerical solution let us take a look at the

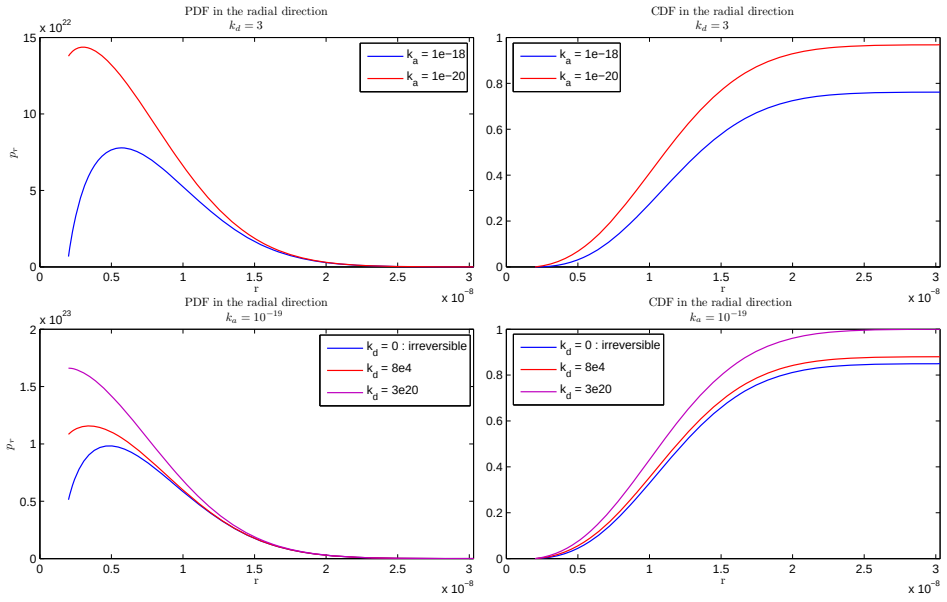
effects of the model parameters k_a and k_d on the graph of the solution.

The association rate, k_a, and the dissociation rate, k_d, mainly have influence on the boundary condition (3.4) determining the value of the first derivative of the probability density function with respect to r at the boundary $r = \sigma$. Note that in fact this derivative describes the rate of change in density of free molecules around the bound state.

If we fix the dissociation rate, then the higher the association rate is, the more likely molecules will react when they are close enough to each other, as can be seen in the top pair of plots in Figure 3.

In general, a higher dissociation rate leads to a more rapid dissociation, so if two molecules reach the distance where the reaction is expected to occur i.e. σ, the possibility that the reaction begins is low. Now if we increase k_d while k_a is fixed, individual molecules can move away from one another more easily and have more chance to escape from getting trapped in the bound state, and molecules that are already bound tend to dissociate into separated individuals; see the two plots[2] in the bottom row of Figure 3.

Figure 3: The PDF and CDF in the radial direction for different values of association and dissociation rates, in the first row of plots $k_d = 3$ and in the second row k_a is fixed to 10^{-19}. The other model parameters are $\mathtt{D} = 2 \times 10^{-12}$, $\sigma = 2 \times 10^{-9}$, $\mathtt{r_0} = 5 \times 10^{-9}$, $\mathtt{t} = 1 \times 10^{-5}$.



3.1.2 Numerical solution

To solve the subproblem (3.1) - (3.4) numerically, we use the *Crank-Nicolson* finite difference scheme because of its stability and second order accuracy in time and space.

[2]Since no analytical formula for the CDF for irreversible reactions was available to us, we instead used the formula (3.14) and set k_d equal to a very small positive number like `1.00e-020`.

Consider a non-uniform spatial grid that contains the initial radial position $r_\circ$ as a grid point[3] and subdivide the time interval $[0, T]$ into N subintervals of equal length $\Delta t = \frac{T}{N}$. Replacing the time and spatial derivatives in the equation (3.1) by the following approximations:

$$\frac{\partial p_r}{\partial t}(r_j, t^n) \approx \frac{p_j^{n+1} - p_j^n}{\Delta t}\,, \tag{3.19a}$$

$$\frac{\partial p_r}{\partial r}(r_j, t^n) \approx \frac{h_j}{h_{j+1}(h_j + h_{j+1})} p_{j+1}^n + \frac{h_{j+1} - h_j}{h_j h_{j+1}} p_j^n - \frac{h_{j+1}}{h_j(h_j + h_{j+1})} p_{j-1}^n\,, \tag{3.19b}$$

$$\frac{\partial^2 p_r}{\partial r^2}(r_j, t^n) \approx \frac{2}{h_{j+1}(h_j + h_{j+1})}(p_{j+1}^n - p_j^n) - \frac{2}{h_j(h_j + h_{j+1})}(p_j^n - p_{j-1}^n)\,, \tag{3.19c}$$

where $h_j = r_j - r_{j-1}$, the Crank-Nicolson approximation for (3.1) after rearranging terms can be formulated as

$$l_j p_{j-1}^{n+1} + m_j p_j^{n+1} + u_j p_{j+1}^{n+1} = l'_j p_{j-1}^n + m'_j p_j^n + u'_j p_{j+1}^n\,, \tag{3.20}$$

where

$$l_j = \tfrac{D\,\Delta t}{h_j(h_j+h_{j+1})}\left(-1 + \tfrac{h_{j+1}}{r_j}\right)\,, \tag{3.21a}$$

$$m_j = 1 + D\,\Delta t\left(\tfrac{1}{h_{j+1}(h_j+h_{j+1})} - \tfrac{h_{j+1}-h_j}{r_j h_j h_{j+1}} + \tfrac{1}{h_j(h_j+h_{j+1})}\right)\,, \tag{3.21b}$$

$$u_j = -\tfrac{D\,\Delta t}{h_{j+1}(h_j+h_{j+1})}\left(1 + \tfrac{h_j}{r_j}\right)\,, \tag{3.21c}$$

$$l'_j = -l_j\,, \tag{3.21d}$$

$$m'_j = 1 - D\,\Delta t\left(\tfrac{1}{h_{j+1}(h_j+h_{j+1})} - \tfrac{h_{j+1}-h_j}{r_j h_j h_{j+1}} + \tfrac{1}{h_j(h_j+h_{j+1})}\right)\,, \tag{3.21e}$$

$$u'_j = -u_j\,, \tag{3.21f}$$

and the initial and boundary conditions are imposed in the following way:

The initial condition (3.2) gives $p_i^1 = \frac{1}{h_i}$, for the first time step, where i is the index of $r_\circ$ in the spatial grid.

For practical reasons, the boundary condition at infinity (3.3) must be replaced by the same condition at a suitably large finite distance $r_{\max}$. In order to reduce the effects of the approximate boundary condition, the value of $r_{\max}$ has to be large enough to ensure that the solution p_r has already decayed to significantly small values at the boundary. From (3.8), one can say that p_r is nearly Gaussian with a mean value of $r_\circ$ and a standard deviation of $\sqrt{2DT}$. Under this assumption, $r_{\max} = r_\circ + 4\sqrt{2DT}$, the 99.99 % upper confidence bound for such a normal (Gaussian) distribution, could be a suitable choice. Then for each computational time step, the boundary condition (3.3) is expressed as

[3]In this subsection, to prevent the initial radial position from being mistaken with r_0, the first grid point in the r direction which is equal to σ, we use the Fraktur font for the 0 subscript that indicates the position at the initial time.

$p_M^n = 0$, where M is the last grid index in the r direction (i.e. $r_M = r_{\max}$).

In addition, the boundary condition (3.4) can be approximated by

$$4\pi\sigma^2 D\,\frac{p_2^n - p_1^n}{h_2} \approx k_a p_1^n - k_d(1 - S(t^n))\,, \qquad 1 \leqslant n \leqslant N+1\,, \tag{3.22}$$

from which we have

$$p_1^n = \frac{1}{4\pi\sigma^2 D + k_a h_2}\left(4\pi\sigma^2 D\,p_2^n + h_2 k_d(1 - S(t^n))\right)\,, \qquad 1 \leqslant n \leqslant N+1\,. \tag{3.23}$$

In the above equation, the survival probability at the n^{th} time step, $S(t^n)$, can be either computed via the analytical formula (3.12) or evaluated approximately by applying the trapezoidal rule to the truncation of the infinite integral in Eq. (2.8) at $r_{\max}$, using the values of p_r at the spatial grid points obtained from the previous time step. Here we use the latter to have an entirely numerical solution.

Finally, the numerical integration of the integral (3.13) in the domain $[\sigma, r_{\max}]$ by the trapezoidal rule using the values p_j^{N+1}, $1 \leqslant j \leqslant M$, yields a sufficiently accurate approximation of the cumulative distribution function. By using the CDF data and applying Algorithm 1, one can generate random radial positions that are distributed according to the probability density function p_r.

The most critical requirement for implementing the numerical scheme is that $r_\circ$ must be one of the discretization points in the r direction, due to its presence in the initial condition. In order to have such a grid in space, we first discretize the subinterval $[\sigma, r_\circ]$ into M_{left} points, then using the same step size we discretize the subinterval $[r_\circ, r_{\max}]$[4] [5].

When the initial position $r_\circ$ is very close to the reaction radius σ, a major problem arises: the numerical solution exhibits decaying oscillatory behaviour around $r_\circ$. To overcome the oscillation we need to take more time steps increasing the cost of the computation. Because of this problem and also the inevitable difficulty of applying the numerical scheme to the case where $r_\circ = \sigma$, the numerical method is not recommended to solve the system (3.1) - (3.4) when the molecules are located initially in or too close to the bound state.

Now assume that $r_\circ$ is close to σ but not that close that leads to the above mentioned problem, then, as it can be noticed in the figures in the previous subsection, the slope of the PDF plot in a small neighbourhood of $r = \sigma$ is very steep. In this case more grid points are required in the region for accurate solution. This grid can be constructed by

[4] Depending on whether $r_{\max}$ is divisible by the step size $\Delta r = \frac{r_\circ - \sigma}{M_{\text{left}} - 1}$ or not, the length of the last grid interval $[r_{M-1}, r_M]$ could be equal to or smaller than Δr.

[5] When the distance between $r_\circ$ and σ is very small compared to the length of the interval $[r_\circ, r_{\max}]$, this idea of discretization may generate too many grid points. If this is the case, we can simply consider σ as the first grid point and divide $[r_\circ, r_{\max}]$ into a set of $M - 1$ grid points.

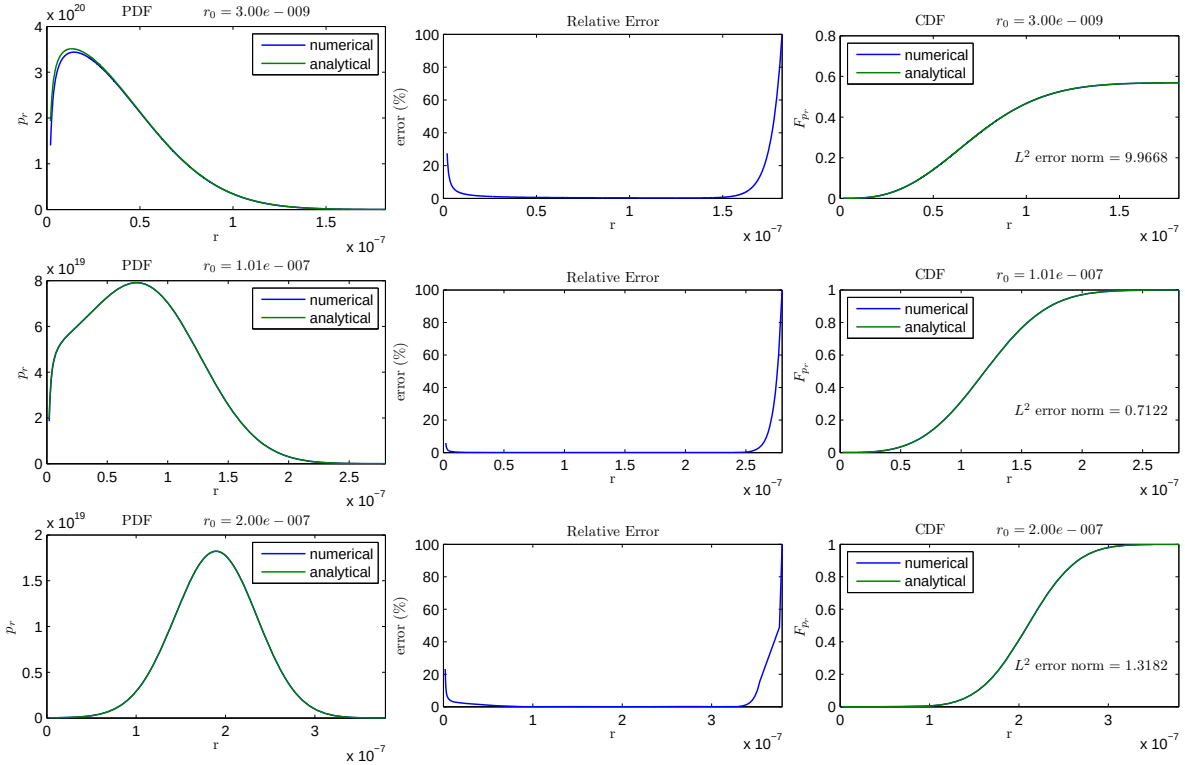
repeatedly bisecting the former grid intervals close to σ until a sufficiently fine mesh is obtained. This technique can also be used for small values of T, where we have the same situation. Needless to say, performing an adaptive grid generation algorithm is the best way to treat this kind of situation, but that needs more study to be done. Here we use the simple technique that we explained above.

Figure 4 compares the PDF and CDF obtained from the numerical method with the analytical solution for three different values of r_0. The plots in the middle column show the relative error (in percent) in computing the PDF. For each value of r_0, the L^2 error norm computed via

$$\texttt{error} = \sqrt{\sum_{j=2}^{M} 4\pi r_j^2 h_j \left(p_j^{N+1} - p_r(r_j)\right)^2} \tag{3.24}$$

is shown as a text on the CDF plot. As can be seen, the relative error in the right tail of the numerical PDFs can be extremely large but the error is not of the main concern as it has no significant influence on the corresponding CDFs (the L^2 error norm is reasonably small).

Figure 4: Comparison of the numerical PDF and CDF with the exact analytical ones for three different values of r_0: `3e-9`, `1.01e-7` and `2e-7` . The plots in the middle column show the percent error in PDFs and the texts on the CDF plots indicate the corresponding L^2 error norms. In this example $\texttt{k}_\texttt{a} = 10^{-19}$, $\texttt{k}_\texttt{d} = 2.9894$, $\texttt{D} = 2 \times 10^{-12}$, $\sigma = 2 \times 10^{-9}$, $\texttt{T} = 5 \times 10^{-4}$.

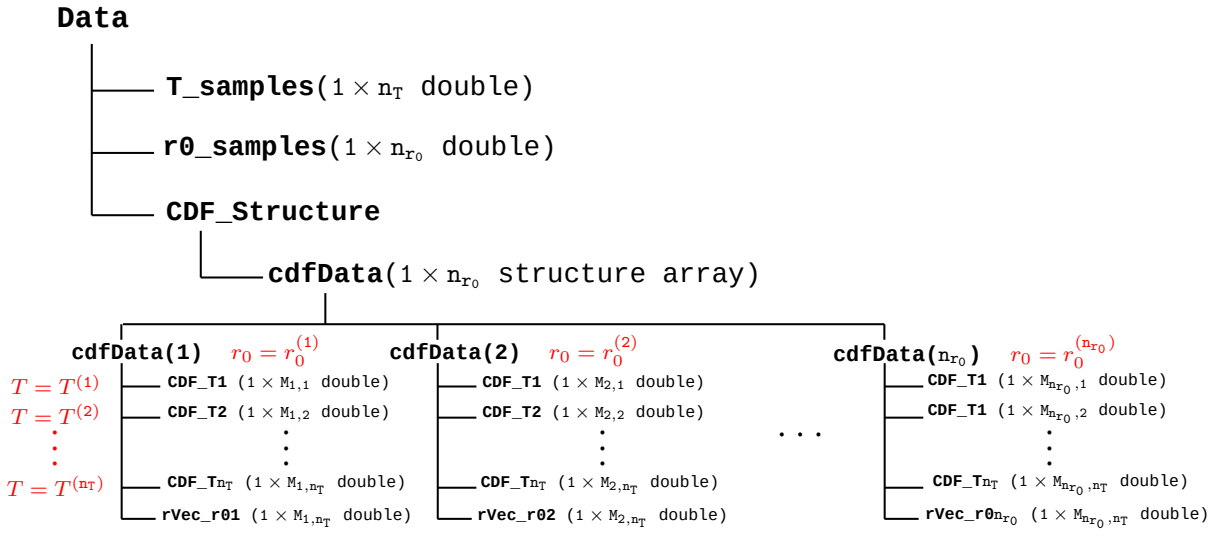


3.1.3 Interpolation in a precomputed lookup table

Another way to randomly draw samples from the PDF solution of the radial part of the Smoluchowski equation, Eqs. (3.1) - (3.4), for any given pair of values of initial position r_0 and time instance T, is to use linear interpolation between precomputed CDF values stored in a 3D lookup table. For fixed model parameters k_a, k_d, σ and D, this method involves two stages: creating the lookup table and performing interpolation between the table's data points.

To create the table, first, from reasonable ranges, a finite number of (not necessarily equally spaced) distinct values of r_0 and T are sampled, say $\mathtt{n_{r_0}}$ values for r_0 and $\mathtt{n_T}$ values for T, and sorted by ascending order. Then the cumulative distribution function is computed for every $\mathtt{n_{r_0}} \times \mathtt{n_T}$ possible combination of the r_0 and T values using either of the two methods explained in the previous subsections. These CDF values form the entries or nodes of the lookup table and are stored on disk in a data structure as illustrated schematically in Figure 5 below. Let `i` index the r_0 samples and `j` index the T samples, assigned respectively to the variables `r0_samples` and `T_samples`, then in this figure, $\mathtt{M_{i,j}}$ denotes the total number of grid points in space that are used in computing the CDF when $r_0 = r_0^{(\mathtt{i})}$ and $T = T^{(\mathtt{j})}$; and every horizontal branch in the diagram indicates a *field* of a *structure* (data type).

Figure 5: The data structure of a 3D lookup table whose entries are the CDF values corresponding to the PDF solutions of the radial part of the Smoluchowski equation for $\mathtt{n_T}$ different time instances T, $\mathtt{n_{r_0}}$ different initial positions r_0 and fixed model parameters k_a, k_d, σ and D.



Such a data structure has a great flexibility for storing vectors of CDF values of different lengths, which is essential when the whole data or a portion of it is computed via the numerical method. If the analytical method is to be employed to obtain *all* the CDF data, then for any sampled values of r_0 and T, a fixed number of spatial grid points

M can be used, i.e. $\mathtt{M}_{\mathtt{i},\mathtt{j}} = M$, where $1 \leq \mathtt{i} \leq \mathtt{n}_{\mathtt{r}_0}$ and $1 \leq \mathtt{j} \leq \mathtt{n}_{\mathtt{T}}$, because there will be no constraint on how the space domain is discretized and the larger number of grid points we take, the smoother results we get. In that case, our data could be stored in a (preallocated) three dimensional matrix, but remember that in the numerical method $\mathtt{M}_{\mathtt{i},\mathtt{j}}$ depends on the grid refinement needed to be performed to reduce the error and the choice of $r_{\max}$, which itself depends on $r_0^{(\mathtt{i})}$ and $T^{(\mathtt{j})}$ as $r_{\max} = r_0^{(\mathtt{i})} + 4\sqrt{2DT^{(\mathtt{j})}}$, so $\mathtt{M}_{\mathtt{i},\mathtt{j}}$ varies and can not be determined prior to the computation.

For interpolating between the entries of the table of CDF values, beside the CDF data, we need the sets of the grid points in space that were used in computing those values, which may make the size of the data set to be stored very huge. The amount of data can be considerably reduced by being consistent in choosing the method for computing the CDF for every $T^{(\mathtt{j})}$, $1 \leq \mathtt{j} \leq \mathtt{n}_{\mathtt{T}}$, while r_0 is fixed, because then storing only $\mathtt{n}_{\mathtt{r}_0}$ sets of grid points in r (that in Figure 5 are assigned to the variables identified by names starting with `rVec_r0`) will cover our need.

Having the tabulated data and following the three steps of Algorithm 2, one can generate a random radial position, sampled from the probability density function $p_r(r, \tau|\rho, 0)$ for any arbitrary pair of values $\rho \in [r_0^{(1)}, r_0^{(\mathtt{n}_{\mathtt{r}_0})}]$ and $\tau \in [T^{(1)}, T^{(\mathtt{n}_{\mathtt{T}})}]$:

Data: a lookup table whose entries are the CDF values precomputed for every possible pair of r_0 and T values taken from the sets r0_samples and T_samples
input : arbitrary initial position ρ and time instance τ from reasonable ranges and ξ: a uniformly distributed random number on the interval $[0, 1]$
output: a radial position r_ξ sampled from the probability density solution p_r when $r_0 = \rho$ and $t = \tau$

begin
  step 1 : retrieve the required data from the table ;
  step 2 : find the interpolation region ;
  step 3 : perform a trilinear interpolation ;
end

Algorithm 2: Algorithm for sampling radial distances from the PDF solution p_r for arbitrary values of r_0 and T via linear interpolation in a precomputed lookup table.

In the first step of Algorithm 3, based on the values of ρ and τ, we access the part of the data, where the the points of the CDF corresponding to $p_r(r, \tau|\rho)$ are expected to be located and retrieve all the four surrounding CDF vectors with their corresponding spatial discretization data. Note that when both ρ and τ are among the sampled r_0 and T values, then the CDF is already in the table, in which case we could simply retrieve that CDF from the table and apply Algorithm 1 on page 9 (see Algorithm 3 on the following page).

In the second step, we find the table cell containing ξ from the cumulative distribution function of interest. Visualizing the table cell provides a better insight on how this step

```
   (* step 1 :  retrieve the required data from the table  *)
 1 if both ρ and τ are among the sampled r_0 and T values then
 2 |   retrieve the corresponding CDF vector from the table ;
 3 |   apply Algorithm 1 to compte r_ξ using the CDF vector ;
 4 |   return r_ξ ;
 5 end
 6 i ← index of the largest r_0 sample in r0_samples that is ⩽ ρ ;
 7 j ← index of the largest T sample in T_samples that is ⩽ τ ;
 8 r0p ← r0_samples(i) ;        r0n ← r0_samples(i + 1) ;
 9 Tp ← T_samples(j) ;          Tn ← T_samples(j + 1) ;
10 from cdfData(i) retrieve the whole vectors CDF_Tj, CDF_Tj+1 and rVec_r0i
   cdfpp ← CDF_Tj;             cdfpn ← CDF_Tj+1; and
   rp ← rVec_r0i ;
11 from cdfData(i + 1) retrieve the whole vectors CDF_Tj, CDF_Tj+1 and rVec_r0i+1
   cdfnp ← CDF_Tj;             cdfnn ← CDF_Tj+1; and
   rn ← rVec_r0i+1 ;
```

Algorithm 3: The first step of Algorithm 2 on the previous page

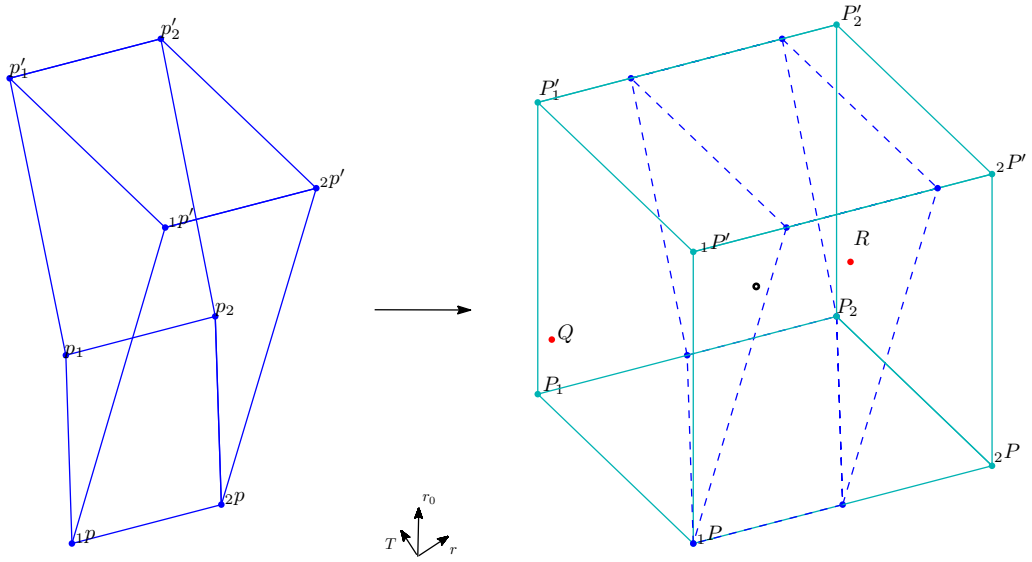
works.

In Figure 6 on the facing page, the panel to the left represents such a cell in one of the worst cases. As we can see, it can have a very irregular shape, which is due to the difference in the curvature of the graphs of the four neighbouring CDFs, a part of which forms four sides of the cell. The only feasible interpolation technique for computing r_ξ is via doing two bilinear interpolations between the vertices of the faces ${}_1p_1p'p'_1p_1$ and ${}_2p_2p'p'_2p_2$ of the cell followed by one linear interpolation (a trilinear interpolation), but the vertices do not have the same radial coordinate r. One way to handle this problem is to interpolate inside a regularly cube-shaped region circumscribing the original cell (as the right hand side figure in Figure 6 illustrates), because then the new vertices ${}_1P$,${}_1P'$, P'_1, and P_1 have the same r coordinate equal to $r^{(1)} := \min\left\{r_{{}_1p}, r_{p_1}, r_{{}_1p'}, r_{p'_1}\right\}$, and the same situation holds for ${}_2P$,${}_2P'$, P'_2, and P_2 where $r = r^{(2)} := \max\left\{r_{{}_2p}, r_{p_2}, r_{{}_2p'}, r_{p'_2}\right\}$. The CDF values at these new points are also needed, which if they are not available in the table, can be calculated by linear interpolation (see Algorithm 4 on the next page).

Now, that all the conditions for performing the bilinear interpolations are satisfied, we find two points Q and R that are *normally* located on the two sides of the target point, in the sense that they all belong to the same CDF and ξ is between the values of the cumulative distribution function at Q and R, or $F_Q \leq \xi \leq F_R$[6]. Considering Q and R as interpolation points and through a simple linear interpolation, r_ξ is obtained (see Algorithm 5 on page 18).

[6]If this is not the case, r_ξ is computed via linear *extrapolation*.

Figure 6: A cell from the 3D lookup table whose entries are the CDF values obtained from numerical method. This cell contains a data point with value $\xi = 0.9595$, the result from interpolation is shown for $\rho =$ `8.8962e-008` and $\tau =$ `1.2558e-004`, and the model parameters are fixed as $\mathtt{k_a} = 10^{-19}$, $\mathtt{k_d} = 2.9894$, $\mathtt{D} = 2 \times 10^{-12}$, $\sigma = 2 \times 10^{-9}$.



(* step 2 : find the interpolation region *)
$k \leftarrow$ *index of the largest CDF value in* cdfpp *that is* $\leqslant \xi$;
$l \leftarrow$ *index of the largest CDF value in* cdfpn *that is* $\leqslant \xi$;
$k' \leftarrow$ *index of the largest CDF value in* cdfnp *that is* $\leqslant \xi$;
$l' \leftarrow$ *index of the largest CDF value in* cdfnn *that is* $\leqslant \xi$;
if $k \geq$ *number of elements of* cdfpp *or* $l \geq$ *number of elements of* cdfpn *or* $k' \geq$ *number of elements of* cdfnp *or* $l' \geq$ *number of elements of* cdfnn **then**
  $r_\xi \leftarrow \sigma$; (* molecules are in the bound state *)
  return r_ξ ;
end
find the vertices of the LUT cell which contains the target point with CDF value ξ:
$_1p : (\mathsf{r0p}, \mathsf{Tp}, \mathsf{rp}(k); \mathsf{cdfpn}(k))$, $_2p : (\mathsf{r0p}, \mathsf{Tp}, \mathsf{rp}(k+1); \mathsf{cdfpn}(k+1))$,
$_1p' : (\mathsf{r0n}, \mathsf{Tp}, \mathsf{rn}(k'); \mathsf{cdfpn}(k'))$, $_2p' : (\mathsf{r0n}, \mathsf{Tp}, \mathsf{rn}(k'+1); \mathsf{cdfpn}(k'+1))$,
$p_1 : (\mathsf{r0p}, \mathsf{Tn}, \mathsf{rp}(l); \mathsf{cdfpn}(l))$, $p_2 : (\mathsf{r0p}, \mathsf{Tn}, \mathsf{rp}(l+1); \mathsf{cdfpn}(l+1))$,
$p'_1 : (\mathsf{r0n}, \mathsf{Tn}, \mathsf{rn}(l'); \mathsf{cdfpn}(l'))$, $p'_2 : (\mathsf{r0n}, \mathsf{Tn}, \mathsf{rn}(l'+1); \mathsf{cdfpn}(l'+1))$;
find the vertices of a regularly cube-shaped region circumscribing the table cell :
$_1P : (\mathsf{r0p}, \mathsf{Tp}, r^{(1)}; F_{_1P})$, $_2P : (\mathsf{r0p}, \mathsf{Tp}, r^{(2)}; F_{_2P})$,
$_1P' : (\mathsf{r0n}, \mathsf{Tp}, r^{(1)}; F_{_1P'})$, $_2P' : (\mathsf{r0n}, \mathsf{Tp}, r^{(2)}; F_{_2P'})$,
$P_1 : (\mathsf{r0p}, \mathsf{Tn}, r^{(1)}; F_{P_1})$, $P_2 : (\mathsf{r0p}, \mathsf{Tn}, r^{(2)}; F_{P_2})$,
$P'_1 : (\mathsf{r0n}, \mathsf{Tn}, r^{(1)}; F_{P'_1})$, $P'_2 : (\mathsf{r0n}, \mathsf{Tn}, r^{(2)}; F_{P'_2})$;

Algorithm 4: The second step of Algorithm 2 on page 15.

(* step 3 : perform interpolation *)

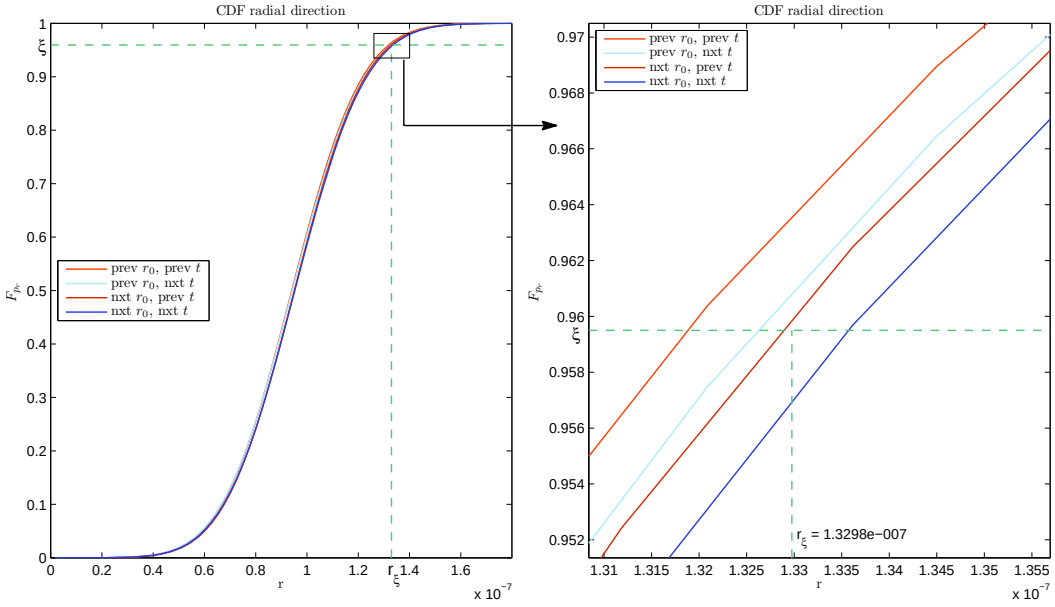
22 compute F_Q the CDF at Q with $r_0 = \rho$ and $T = \tau$, located on the face ${}_1P_1P'P'_1P_1$, via bilinear interpolation using the vertices of the face as interpolation points ;

23 compute F_R the CDF at R with $r_0 = \rho$ and $T = \tau$, located on the face ${}_2P_2P'P'_2P_2$, via bilinear interpolation using the vertices of the face as interpolation points ;

24 compute r_ξ through linear interpolation using two interpolation points $(F_Q, r^{(1)})$ and $(F_R, r^{(2)})$

Algorithm 5: The third step of Algorithm 2 on page 15.

Figure 7: In this figure the result shown in Figure 6 on the preceding page is represented on the CDF plots. The figure to the right is a zoom in to the region indicated by a rectangle on the left hand side figure.



As we shall show by an example, beside the cost in space for data storage, this method has an initial cost in time for generating the data.

Here we create two lookup tables using 200×200 pairs of sampled r_0 and T values, one of which stores the computed exact (analytical) CDF values and the other one stores the numerical values. For fixed r_0 and T, we used the same spatial discretization for both analytical and numerical methods and to accelerate the computations the number of computational time steps in the numerical scheme is changed according to a piecewise constant function of r_0[7]. The information about the generated data and timing results are shown in two listings on the following page.

[7]The purpose is to avoid unnecessary computations when r_0 is large. The larger r_0 is, the fewer time steps are used.

listing 1: Information about the stored data

```
model parameters :
        sigma = 2.0000e-009
        D = 2.0000e-012
        k_a = 1.0000e-019
        k_d = 2.989436788648693
number of r0 samples : 200    from  [ 3.0000e-009 , 2.0609e-007 ]
number of T samples  : 200    from  [ 1.0000e-008 , 7.5928e-004 ]

size of the largest CDF vector  = 1 x 2008 double = 128512 bits = 15.69 kilobytes
size of the smallest CDF vector = 1 x 130  double = 8320   bits = 1.02  kilobytes

Total number of CDF data points in the table = 7248471
Total number of grid (in r) data points in the table = 45682
Total size of the stored data (in any of the two tables) = 52.6904 MB ( 55249896 bytes )
```

In the above listing, the maximum and minimum size of the CDF vectors in the tables are also given to show that retrieving CDF vectors from the tables in the first step of Algorithm 2, as described in detail in Algorithm 3, is not an expensive process.

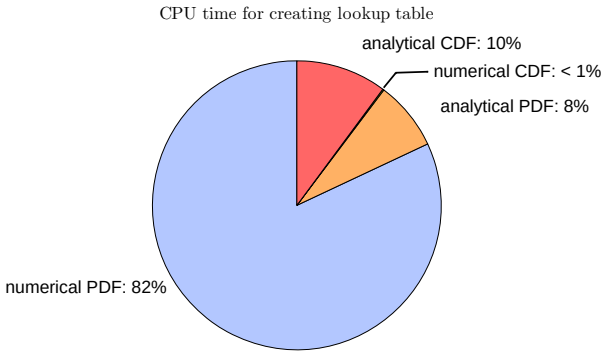
listing 2: CPU time spent on creating the example tables

```
Total elapsed CPU time for main computations = 1.12e+003 seconds = 18.71 minutes

Elapsed time to compute numerical solution   : 9.20e+002 seconds = 15.34 minutes
Elapsed time to compute analytical solution  : 8.68e+001 seconds = 1.45 minutes
Elapsed time to compute numerical CDF        : 1.72e+000 seconds = 0.03 minutes
Elapsed time to compute analytical CDF       : 1.14e+002 seconds = 1.90 minutes
```

The pie chart in Figure 8 displays what portion (in percent) of the total execution time for creating the above described lookup tables was spent on any part of the computations. From the chart, it can be clearly noticed that the fastest way to create a CDF lookup table is to use the analytical method to obtain the PDFs and then compute their corresponding CDFs using numerical integration.

Figure 8: This pie chart shows the elapsed CPU time for creating two lookup tables using analytical and numerical methods. the model parameters are $\mathtt{k_a} = 10^{-19}$, $\mathtt{k_d} = 2.9894$, $\mathtt{D} = 2 \times 10^{-12}$, $\sigma = 2 \times 10^{-9}$.



3.2 Solution and sampling in the angular direction

Since applying numerical schemes to solve the sub-problem (3.6)-(3.7) involves special difficulties due to the presence of mathematical singularities in the polar direction, here we attempt two approaches: analytical method and linear interpolation in a lookup table.

3.2.1 Analytical solution

The analytical solution of the angular part of the Smoluchowski equation, Eqs. (3.6)-(3.7), is given in the form of an infinite series as (see [16])

$$p_\theta(\theta, t|r, 0, \phi_0, t_0) = \sum_{l=0}^{\infty} \frac{2l+1}{4\pi r^2} \exp\left(-l(l+1)\frac{D(t-t_0)}{r^2}\right) P_l(\cos\theta)\,, \tag{3.25}$$

where P_l denotes the l^{th}-degree Legendre polynomial, and the corresponding cumulative distribution function (CDF), F_{p_θ}, is given by

$$F_{p_\theta}(\theta, t|r, 0, \phi_0, t_0) = \int_0^\theta p_\theta(\vartheta, t|r, 0, \phi_0, t_0)\, 2\pi r^2 \sin\vartheta\, d\vartheta\,. \tag{3.26}$$

Note that for a fixed radial distance r, the conditional PDF (3.25) is a function of the polar angle θ and time t, and independent of the azimuthal angle ϕ.

The convergence of the series in (3.25) can be verified analytically using standard techniques and the rate of convergence is mainly influenced by the value of the parameter $\hat{t} := \frac{D(t-t_0)}{r^2}$.

For practical computations, at some level, the infinite series (3.25) is required to be truncated to a finite summation. A possible choice of the level of truncation can be the first positive integer $l_{\max}$, for which the absolute value of the remainder term

$$\mathcal{R} := \sum_{l=l_{\max}+1}^{\infty} (2l+1) \exp\left(-l(l+1)\hat{t}\right) P_l(\cos\theta) \tag{3.27}$$

becomes very small. Since $|P_l(x)| \le 1$, for any $x \in [-1, 1]$ and every positive integer l, we have

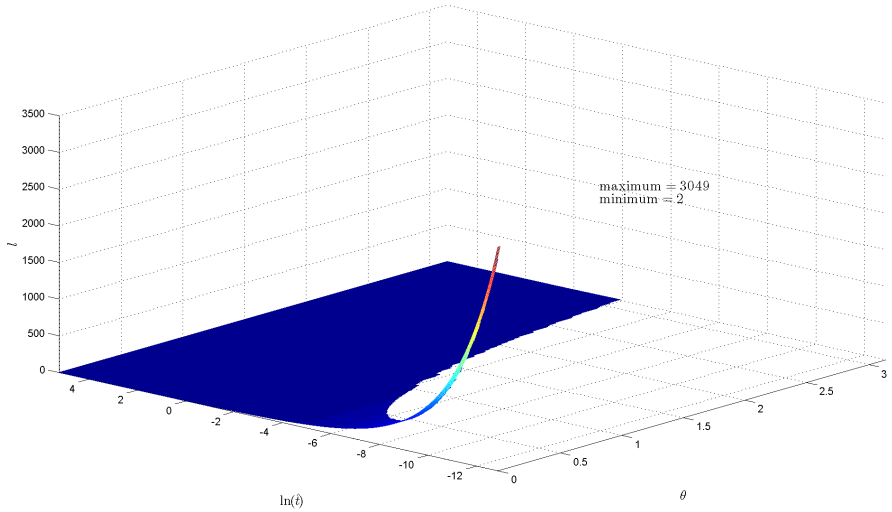
$$\begin{aligned} |\mathcal{R}| &\le \max_\theta |P_l(\cos\theta)| \sum_{l=l_{\max}+1}^{\infty} (2l+1) \exp\left(-l(l+1)\hat{t}\right) \\ &\le \sum_{l=l_{\max}+1}^{\infty} (2l+1) \exp\left(-l(l+1)\hat{t}\right)\,. \end{aligned} \tag{3.28}$$

Approximating the last summation in (3.28) by an infinite integral and setting the value of the integral less than or equal to a very small positive real number ϵ, gives the following criterion

$$-(l_{\max}+1)(l_{\max}+2)\hat{t} \le \ln(\hat{t}\epsilon)\,, \tag{3.29}$$

for truncating (3.25) at a proper level.

Figure 9: This figure plots $l = l_{\max} + 1$, the number of terms of the infinite series (3.25) that are sufficient to reach convergence within the stopping criterion $\epsilon = 10^{-5}$, as a function of the polar angle θ and the natural logarithm of the parameter $\hat{t}$.



In the above figure, the total number of terms needed to compute a reasonably accurate approximation of the infinite series, i.e. $l_{\max} + 1$, is plotted against θ and the natural logarithm of $\hat{t}$. As we can see for small values of $\hat{t}$, it is very difficult to reach convergence (in the worst case 3049 terms are needed), but when $\hat{t}$ is large the series converges pretty fast (after only 2 terms). In this example, a second stopping criterion different from (3.28) is also imposed that we shall define in the following paragraph.

Note that the criterion (3.28) is expressed in terms of the parameter $\hat{t}$, so, for a fixed value of $\hat{t}$, we can estimate the probability density function p_θ at a number of grid points (that are not necessarily equally spaced from each other) in the polar angle domain $[0, \pi]$, but the truncation error, caused by approximating the infinite series (3.25) by a finite sum, results in unwanted oscillations in the PDF graph, especially for small $\hat{t}$ and a larger θ where the PDF reaches a constant level. We can avoid this oscillation as well as unnecessary computations by introducing another stopping criterion[8] depending on θ and test if the value of the PDF has stopped changing, relying on the fact that the probability density function p_θ is a non-increasing function away from $\theta = 0$. In Figure 9, the reason that computations are not performed for some values of θ is that the stopping criterion is met.

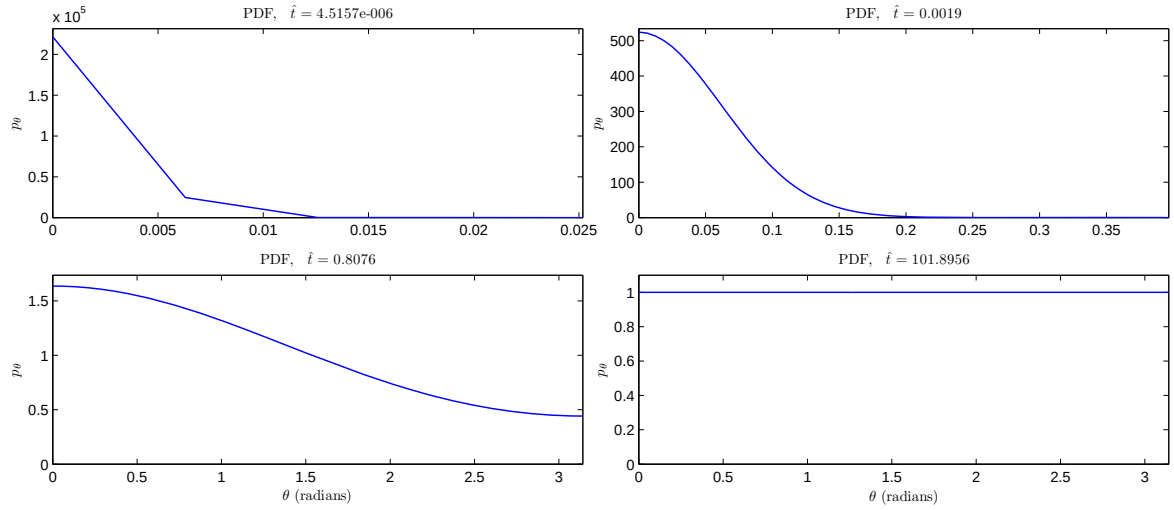
[8] An alternative stopping criterion could be

$$\left| \frac{a_{l_{\max}}}{s_{l_{\max}}} \right| \leq \epsilon' \,,$$

where ϵ' is a small positive real number, $a_{l_{\max}}$ is the $l_{\max}$-th term of the sequence, that the series (3.25) is defined as the sum over its terms, and $s_{l_{\max}}$ denotes the sum of the series after performing the truncation at the index $l_{\max}$.

The PDF for four different values of $\hat{t}$ are plotted in Figure 10. It can be inferred from the figure that for large values of $\hat{t}$, p_θ becomes uniform, that means as time proceeds or the radial distance between molecules gets smaller, molecules choose their new position in the polar direction according to a uniform distribution.

Figure 10: the PDF in the polar direction for different values of $\hat{t}$, here $\epsilon =$ `1e-5`



Having the PDF p_θ obtained as described above, the corresponding cumulative distribution function (3.26) can be evaluated numerically using the trapezoidal rule and hence an angular position of a molecule can be sampled via the following algorithm[9]:

Data: cdf, the vector of CDF values evaluated at a grid of points in the θ direction
Result: a polar position θ_ξ sampled from the probability density function p_θ; and a uniformly distributed random azimuthal angle ϕ

1 $\xi \leftarrow$ *a uniformly distributed random number on the interval* $[0, 1]$;
2 **if** $\xi >$ *the maximum CDF value, that might become true due to the expected computational error in computing the CDF* **then**
3 $\quad \xi \leftarrow \xi * \max(\mathsf{cdf})$; `(* rescale ξ *)`
4 **end**
5 $k \leftarrow$ *index of the largest CDF value that is* $\leqslant \xi$;
6 compute θ_ξ by linear interpolation using two interpolating points $(\mathsf{cdf}(k), \theta_k)$ and $(\mathsf{cdf}(k+1), \theta_{k+1})$;
7 $\phi \leftarrow$ *a uniformly distributed random number on the interval* $[0, 2\pi)$

Algorithm 6: Algorithm to draw random angular positions according to the solution of the angular part of the Smoluchowski equation.

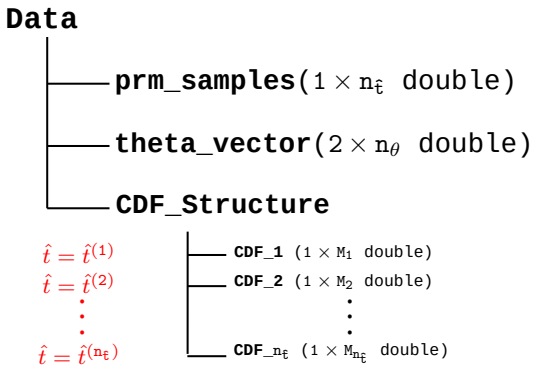
[9] Algorithm 6 is basically similar to the inverse transform sampling algorithm, but differs in some details.

3.2.2 Interpolation in a precomputed lookup table

The second approach that we employ to sample polar positions according to the PDF solution of the angular part of the Smoluchowski equation (3.6)-(3.7) when the diffusion coefficient D is fixed and the current radial position r and the time step[10] $\Delta t = t - t_0$ are given, is to perform linear interpolation between precomputed CDF values tabulated in a 2D lookup table. This method consists of two stages: creating the lookup table and linear interpolation between the entries of the table.

The procedure of creating the CDF lookup table is as follows: Assume that we have a set of $\mathtt{n_t}$ preselected (not necessarily equally spaced) values for the parameter $\hat{t}$ in a suitable range, that are sorted in increasing order and indexed by `i`. For every member of this set and on a fixed grid consisting of $\mathtt{n}_\theta$ points in the polar direction, we compute the corresponding cumulative distribution function via the method described in the previous subsection. The lookup table is then constructed by storing the obtained CDF values in memory for later access. Depending on the value of $\hat{t}$ and the stopping criteria that we use in the calculations, the $\mathtt{n_t}$ vectors of the precomputed CDF values may have different lengths (of at most $\mathtt{n}_\theta$) forcing us to use a data structure similar to what we proposed in subsection 3.1.3 for data storage. The data structure is illustrated in Figure 11 below. For the purpose of interpolation, together with the CDF data, we need to store the sampled $\hat{t}$ values (`prm_samples`) and the grid of polar angles (`theta_vector`) used in the computations.

Figure 11: The data structure of a 2D lookup table whose entries are the CDF values corresponding to the PDF solutions of the angular part of the Smoluchowski equation for $\mathtt{n_t}$ different values of the parameter $\hat{t}$. At most $\mathtt{n}_\theta$ number of grid points in the polar direction, stored in the vector `theta_vector`, are used for computing the CDF values. In the diagram every horizontal branch indicates a field of a structure and $\mathtt{M_i}$ denotes the length of the vector of CDF values `CDF_i` and $\mathtt{M_i} \leq \mathtt{n}_\theta$.



After creating the table, for any arbitrary value of r and Δt, for which $\hat{t}^{(1)} \leq \frac{D\Delta t}{r^2} \leq \hat{t}^{(\mathtt{n_t})}$, we can sample a random polar position from the corresponding probability density function,

[10] Here by time step we mean the time passed from the last update, this should not be mistaken with the computational time step used in the numerical method.

to which and its CDF we refer as target, by Algorithm 7 :

Data: a lookup table whose entries are the CDF values precomputed for every $\hat{t}$ value taken from the set prm_samples
input : arbitrary radial position ρ and time span τ from reasonable ranges; and ξ a uniformly distributed random number on the interval $[0, 1]$
output: a polar position θ_ξ sampled from the probability density solution p_θ when $r = \rho$ and $t - t_0 = \tau$

begin
  step 1 : retrieve the required data from the table ;
  step 2 : find the interpolation region ;
  step 3 : perform an irregular bilinear interpolation
end

Algorithm 7: Algorithm for sampling polar position from the PDF solution p_θ for arbitrary values of r and Δt via linear interpolation in a precomputed lookup table.

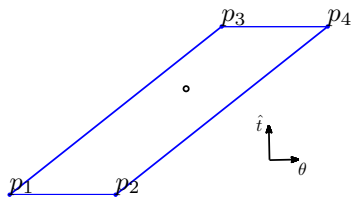
The algorithm has three steps starting with accessing the required data and retrieving it from the table (Algorithm 8). By required data we mean the CDFs that are on both sides of the target CDF (`cdfp` and `cdfn`), provided that it is not already in the table.

```
   (* step 1 :  retrieve the required data from the table *)
 1 τ̂ ← Dτ/ρ² ;
 2 parameter ← Data · prm_samples ;
 3 if τ̂ ∈ parameter then
 4 |   retrieve the corresponding CDF vector from the table ;
 5 |   apply Algorithm 6 to compute θ_ξ using the CDF vector ;
 6 |   return θ_ξ ;
 7 end
 8 theta ← Data · theta_vector ;
 9 i ← index of the largest t̂ sample in parameter that is ⩽ τ̂ ;
10 from CDF_Structure retrieve the whole vector CDF_i     and cdfp ← CDF_i ;
11 from CDF_Structure retrieve the whole vector CDF_i+1 and cdfn ← CDF_i+1 ;
```

Algorithm 8: The first step of Algorithm 7

Remember that the CDF vectors stored in the table are discrete approximations of their corresponding analytical CDFs. The second step (Algorithm 9 on the next page) then deals with finding four elements in the two CDF vectors, obtained from the previous step, that are positioned on the two sides of a point on the corresponding CDFs, whose value is equal to a randomly generated uniformly distributed number $\xi \in [0, 1]$. The four data nodes form a *parallelogram-shaped* region around the point on the target CDF with the CDF value ξ. This is the region, where the interpolation should be performed to compute θ_ξ.

Figure 12: A cell from a 2D CDF lookup table $\xi = 0.6787$, $\rho =$ `3.4601e-008` and $\tau =$ `1.0473e-005`, and the model parameter D is fixed as $\mathtt{D} = 2 \times 10^{-12}$.



(* step 2 : find the interpolation region *)
$k \leftarrow$ *index of the largest CDF value in* cdfp *that is* $\leqslant \xi$;
$l \leftarrow$ *index of the largest CDF value in* cdfn *that is* $\leqslant \xi$;
if $k \geq$ *number of elements of* cdfp *or* $l \geq$ *number of elements of* cdfn **then**
  $\xi \leftarrow \xi * \min\{\max(\mathsf{cdfp}), \max(\mathsf{cdfn})\}$; (* rescale ξ *)
end
find the vertices of the LUT cell which contains the target point with CDF value ξ:
$p_1 : (\mathsf{parameter}(i), \mathsf{theta}(k); \mathsf{cdfp}(k))$,
$p_2 : (\mathsf{parameter}(i), \mathsf{theta}(k+1); \mathsf{cdfp}(k+1))$,
$p_3 : (\mathsf{parameter}(i+1), \mathsf{theta}(l); \mathsf{cdfn}(l))$,
$p_4 : (\mathsf{parameter}(i+1), \mathsf{theta}(l+1); \mathsf{cdfn}(l+1))$;

Algorithm 9: The second step of Algorithm 7 on the facing page.

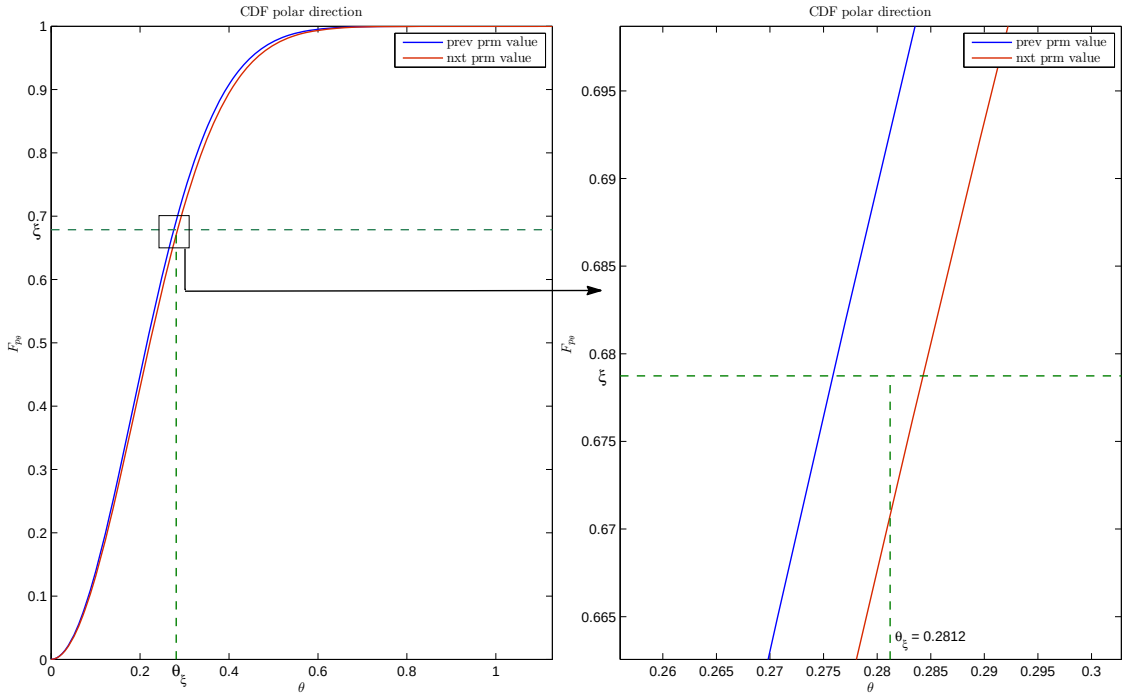
Since we have four interpolation points, p_1 and p_2 on `cdfp` and p_3 and p_4 on `cdfn` (see Figure 12 above), the interpolation will be naturally of *bilinear* type, but for a standard bilinear interpolation we need a regularly rectangular-shaped interpolation region, which is not the case here. Here, our interpolation region is often like a parallelogram due to the fact that as $\hat{t}$ increases, the CDF graph skews to the right. So we have to perform an *irregular bilinear interpolation*, as described in Algorithm 10 :

(* step 3 : perform an irregular bilinear interpolation *)
$v \leftarrow (\hat{\tau} - \mathsf{parameter}(i))/(\mathsf{parameter}(i+1) - \mathsf{parameter}(i))$;
$\mathtt{temp1} \leftarrow \xi - \mathsf{cdfp}(k) + (\mathsf{cdfp}(k) - \mathsf{cdfp}(l)) * v$;
$\mathtt{temp2} \leftarrow \mathsf{cdfp}(k+1) - \mathsf{cdfp}(k) + (\mathsf{cdfp}(k) - \mathsf{cdfp}(k+1) + \mathsf{cdfn}(l) - \mathsf{cdfn}(l+1)) * v$;
$u \leftarrow \mathtt{temp1}/\mathtt{temp2}$;
$\theta_\xi \leftarrow (\mathsf{theta}(k+1) - \mathsf{theta}(k)) * u + (\mathsf{theta}(l) - \mathsf{theta}(k)) * v + \mathsf{theta}(k)$

Algorithm 10: The third step of Algorithm 7.

Like the similar method we discussed in Section 3.1.3 for the radial part of the Smoluchowski equation, for the angular part of the equation, the time needed to generate the lookup tables and the space required for storing them are the main concerns, though not of the same level of seriousness. Here, the CDF lookup tables can be generated much

Figure 13: The result from interpolation shown in Figure 12 on the previous page on the CDF plots. The figure to the right is a zoom in to the region indicated by a rectangle on the left hand side figure.



faster and occupy a much smaller amount of space. Listing 3 below provides information about the size of the data stored in a table constructed using the CDFs computed for 300 different values of the parameter $\hat{t}$ and the CPU time spent for creating the table[11].

listing 3: Information about the stored data and the CPU time for generating the data

```
model parameters :
        D = 2.0000e-012
number of t_tilde samples ( t_tilde = ln(t_hat) ): 300   from  [ -12.8522 , 5.2287 ]
maximum number of grid points in theta direction : 500

size of the largest CDF vector  = 1 x 500 double = 32000 bits = 3.91 kilobytes
size of the smallest CDF vector = 1 x 4   double = 256   bits = 32   bytes
Total number of CDF data points in the table = 76862

Total size of the stored data = 482.97 KB ( 494564 bytes )
Total elapsed CPU time for main computations = 16.7223 seconds
```

[11] While creating such a table one should be aware of the poor performance of the MATLAB's built-in function `legendre.m` designed for computing the values of the Legendre functions. This issue can be resolved by computing the values using the Legendre recursive formulas.

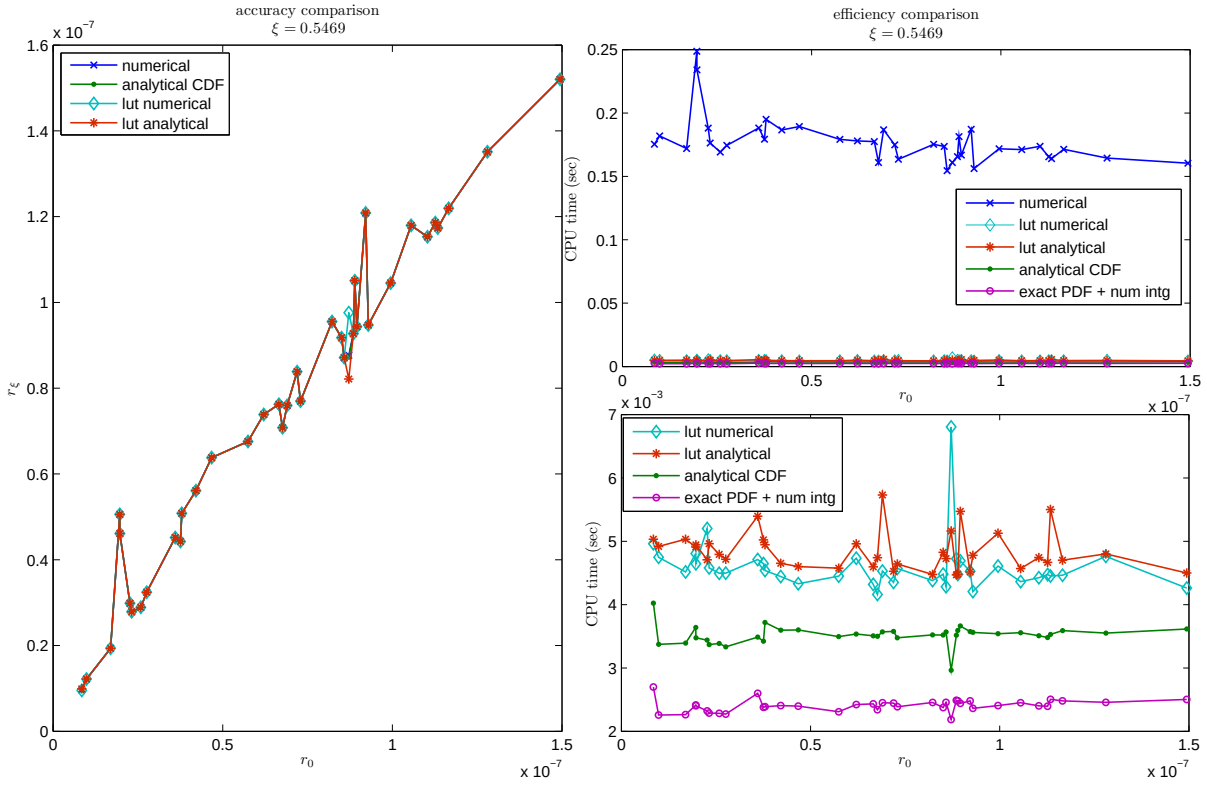
4 Comparison of the different sampling methods

In this section, we compare all the sampling methods presented in Section 3 with respect to accuracy and efficiency, using 92225 distinct pairs of values for (initial) radial position (r_0) r and time step (instance) Δt (T), collected from an accurate simulation using the methods in [4], as reference values. For convenience we denote the set of the pairs of values by `PoV`. The model parameters are: $k_a = 10^{-19}$, $k_d = 2.9894$, $D = 2 \times 10^{-12}$ and $\sigma = 2 \times 10^{-9}$. To check the accuracy, we assumed that the analytical methods yield the most accurate results, although some approximation errors are involved. We performed all the experiments in MATLAB v7.10 (R2010a) on a laptop PC with Intel Core 2 Duo T7200 processor (2.00GHz, 4MB L2 Cache) and 2GB RAM running Windows 7 (64 bit).

4.1 Comparison in the radial direction

In Section 3.1, we presented three different approaches to sample particle positions from

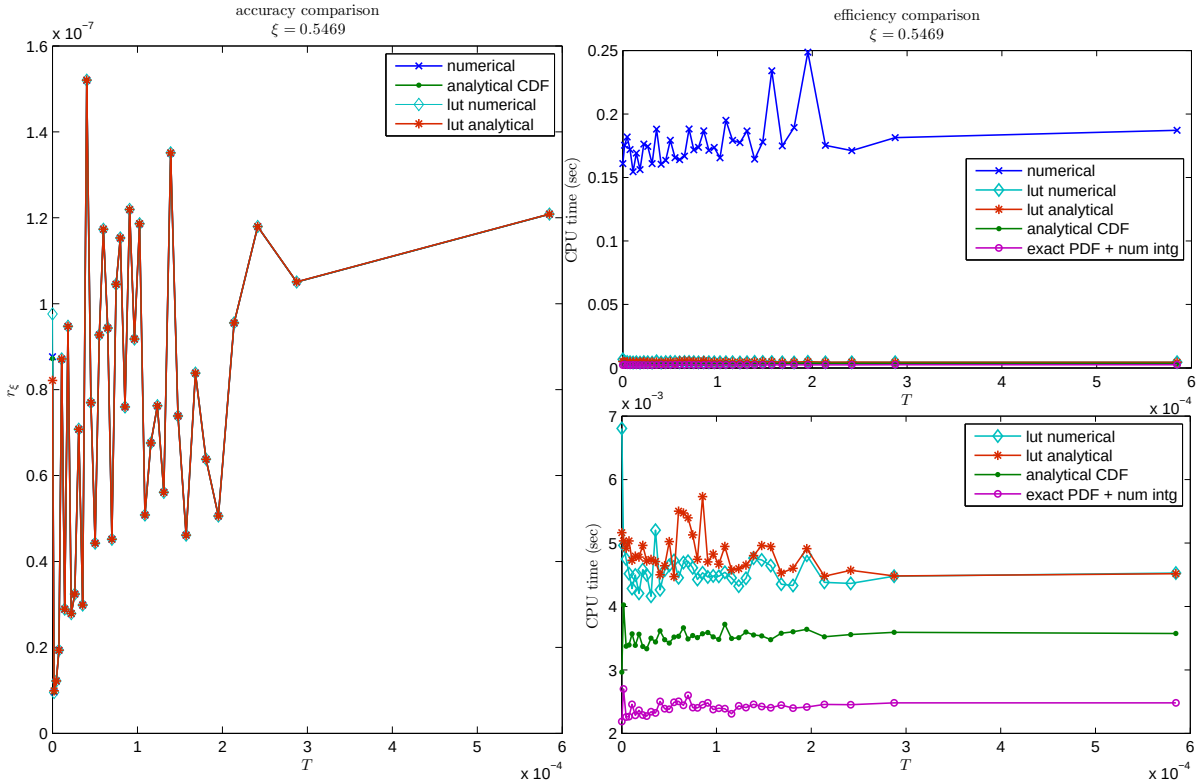
Figure 14: Comparison of the four different sampling methods in the radial direction with respect to accuracy and efficiency for 38 different pairs of r_0 and T values. In this figure, for every method, the computed r_ξ (left) and the run-time required to obtain the solution (right) are plotted against r_0. The plot in the bottom-right corner is a zoom in to the lower part of the plot in the top-right side of the figure. Here we fixed $\xi = 0.5469$ and each run is repeated up to 1000 times for every r_0 and T values.



the PDF solution of the Smoluchowski equation in the radial direction: analytical method, numerical method and linear interpolation between precomputed lookup table data points (LUT-interpolation method from now on for short).

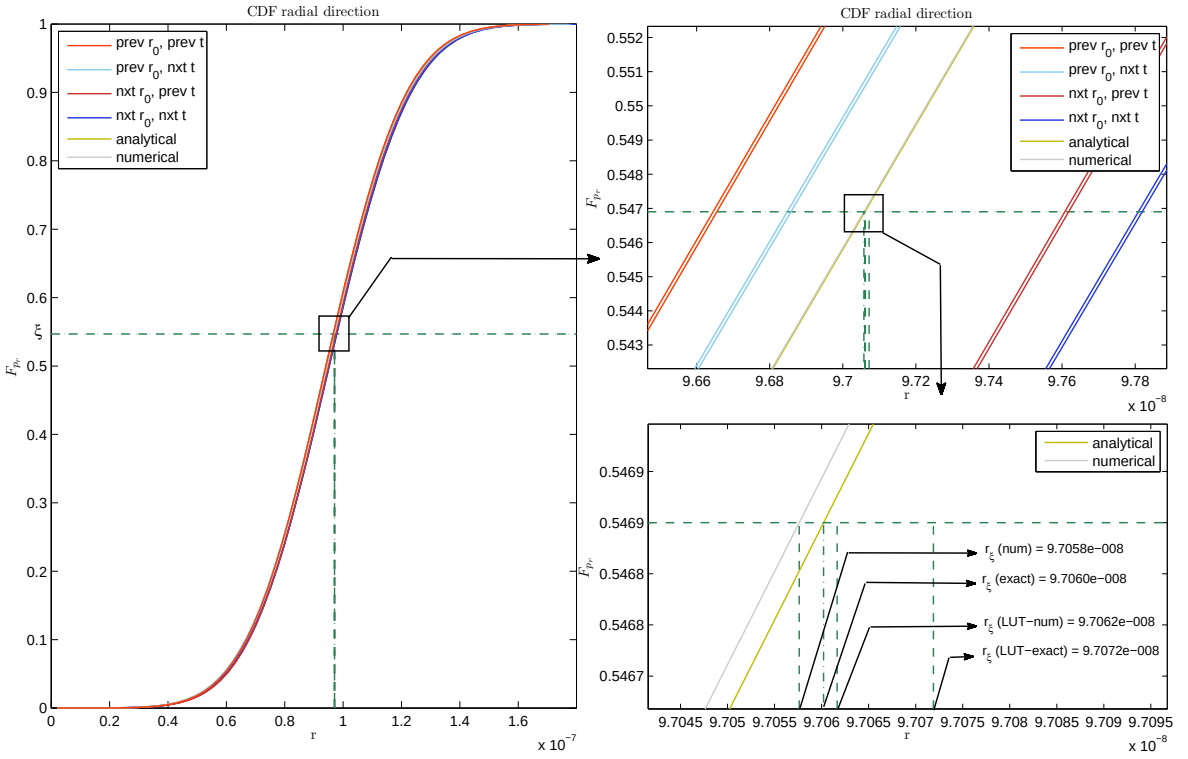
To get an overall feeling for how accurate and efficient the methods are, Figure 14 on the preceding page and Figure 15 plot the r_ξ computed by any of the methods and the CPU time required to obtain the result as a function of r_0 or T, for 38 different pairs of values of r_0 and T and a fixed randomly generated target CDF value ξ, uniformly distributed in the range $[0, 1]$. In Figure 16 on the facing page, as an example, we can take a closer look at the computed r_ξ values for one of the 38 pairs. From these figures, one can observe that all the methods have a good accuracy and the analytical method is the most efficient method and the numerical method is the least efficient one. We shall discuss and validate this conclusion in the following sections.

Figure 15: Comparison of the four different sampling methods in the radial direction with respect to accuracy and efficiency for 38 different pairs of r_0 and T values. In this figure, for every method, the computed r_ξ (left) and the run-time required to obtain the solution (right) are plotted against T. The plot in the bottom-right corner is a zoom in to the lower part of the plot in the top-right side of the figure. Here we fixed $\xi = 0.5469$ and each run is repeated up to 1000 times for every r_0 and T values.



Note that here we perform the experiments for two versions of the LUT-interpolation

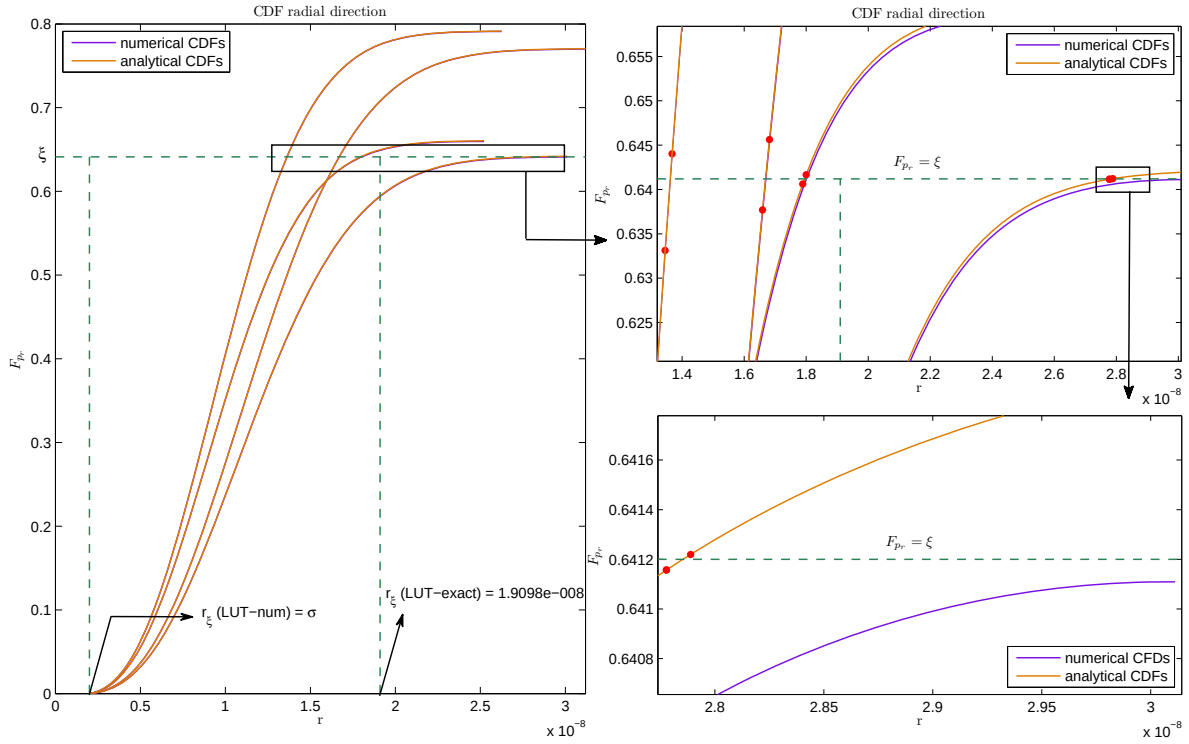
Figure 16: Comparison of the different methods for $r_0 =$ `8.8962e-008`, $T =$ `1.2558e-004` and $\xi = 0.5469$. In this figure, five pairs of solid lines can be noticed. In any of the pairs, represented by a unique color, the line to the right is the plot of an analytical CDF for the values indicated in the legend and the one to the left is the plot of its numerical approximation. The pair in the middle consists of the analytical and numerical CDFs for the given r_0 and T. In this figure the rectangles indicate the regions on the plots where we zoom in.



method: one with a lookup table whose entries are obtained from the numerical solution and the other one with a lookup table that stores the exact CDF data, since there is a special case, where the results from the two versions of the method can differ very much. To see the source of the difference, we need to recall that in the LUT-interpolation method, or Algorithm 2 on page 15, before we can proceed to the last step, i.e. interpolating between the entries of the table, we first need to find the table cell containing the target data point. Otherwise the output of the method/algorithm will be $r_\xi = \sigma$, indicating that molecules are in the bound state. Plotting the cumulative distribution functions that are in the ''neighbourhood'' of the target CDF, one can show the eight vertices of the table cell are actually positioned two-by-two on both sides of the four points, where the horizontal line $F_{p_r} = \xi$ intersects the graphs of the CDFs (see the top right figure in Figure 17 on the following page), so if $F_{p_r} = \xi$ does not intersect at least one of the CDF graphs (in other words, if ξ is greater than the maximum value of at least one of the CDFs), we fail in finding the table cell and hence get $r_\xi = \sigma$ as the result. The problem is that sometimes this only happens when we use a table with numerical data values, as is shown in Figure 17, because normally at the regions close to the right end of the r

domain, the numerical approximation of a CDF is a bit smaller than the exact values due to computational errors.

Figure 17: A special case where we get different results from the LUT-interpolation method, depending on whether we use a lookup table with numerical values or a lookup table with exact CDF data. In this example $\xi = 0.6412$, $r_0 =$ `3.9583e-009` and $T =$ `9.5604e-006`. On the top right figure the eight red dots represent the vertices of the table cell containing the target data point. In this figure the rectangles indicate the regions on the plots where we zoom in.

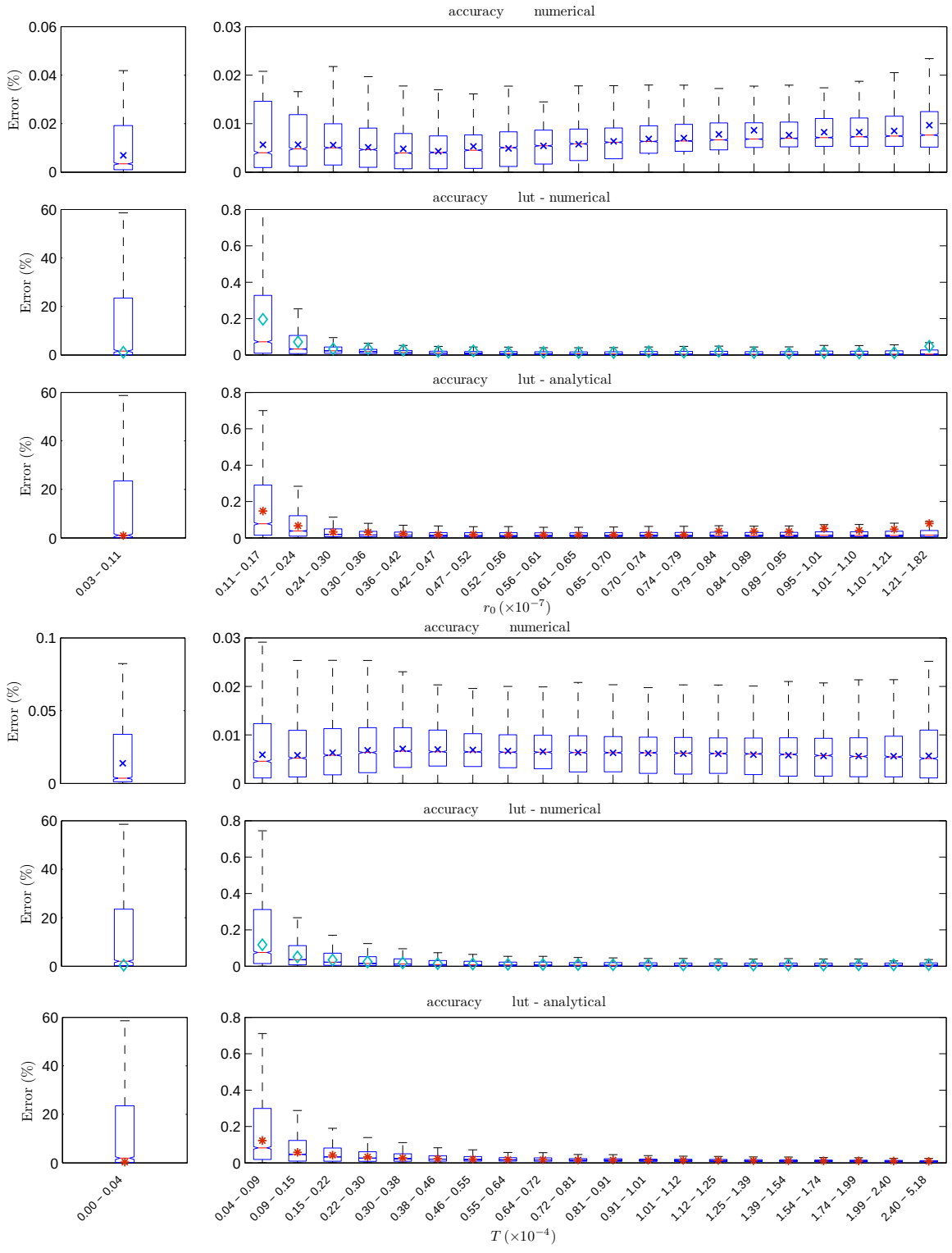


4.1.1 Accuracy comparison

We begin by comparing the sampling methods with respect to accuracy. The comparison is made by applying all the methods over the whole set of values, `PoV`[12], while ξ is fixed, and calculating the percent relative error in computing r_ξ for any member of the set. To investigate the obtained error data, we split the data into 21 groups according to the r_0 or T values and utilize *box-plots* (*box-and-whisker plots*), as demonstrated in Figure 18. In these plots, each box encloses the middle 80% of the data, i.e. the top and bottom edges of the box are the 90$^{\text{th}}$ and 10$^{\text{th}}$ percentiles of the data, respectively (hence the length of the box indicates the spread of the data); the line across the box represents the median and the cross/diamond/asterisk within the box indicates the mean value. The *whiskers*

[12]We exclude those very few members of the set, whose r_0 values are very close to σ, because then we are unable to use the numerical method.

Figure 18: Box-plots of percent error in computing r_ξ using different sampling methods versus r_0 (the top three plots) and T (the bottom three plots). The results obtained from the analytical method are used to estimate the errors. The data are grouped according to the value of r_0 and T, for which the solution is computed. $\xi = 0.5469$ and the experiment is performed for 91735 distinct pairs of values for r_0 and T (for every pair computations are repeated up to 10 times).



(dashed T-bars) display the highest and the lowest data points not considered to be outliers.

From the box-plots Figure 18 and with the help of Figure 19, we can tell that although the accuracy of the numerical method depends on the mesh size used in computations but with an average relative error less than 0.01% it is more accurate than the LUT-interpolation method.

Despite the above fact, however, the LUT-interpolation method performs quite well: From the figures, first, it seems that there is no much difference between the two versions of the LUT-interpolation method. One can conclude that the problem with the LUT-interpolation method emerges for small values of r_0 or T, where a wide variability (between 0.05% up to 20%) can be noticed in the relative error in the results obtained from the method. Often this is not a big concern, because the error data in the corresponding sub-ranges of r_0 or T values are significantly *positively* or *right skewed* (because the median is closer to the lower end of the box and the upper whisker is longer than the bottom one), meaning that, some, but only a few, large errors occurred in the sub-ranges, which also explains the small average relative errors there. In the other sub-ranges (groups), the average error is desirably very small i.e. less than 0.2%.

Figure 19: The average percent error in computing r_ξ within 21 subranges of values for r_0 (top) and T (bottom).

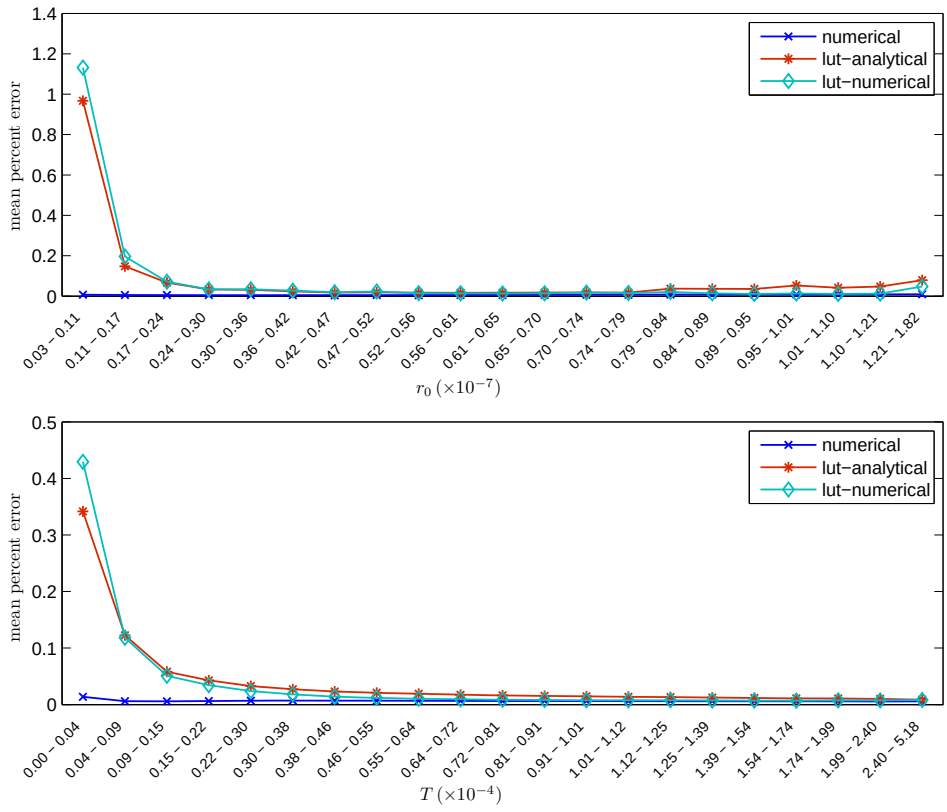


Figure 20: This figure shows how transforming a table cell into a regularly cube-shaped interpolation region, as the second step of the LUT-interpolation method, may cause an error. Here $r_0 =$ `2.3640e-006`, $T =$ `5.0000e-009` and $\xi = 0.5469$.

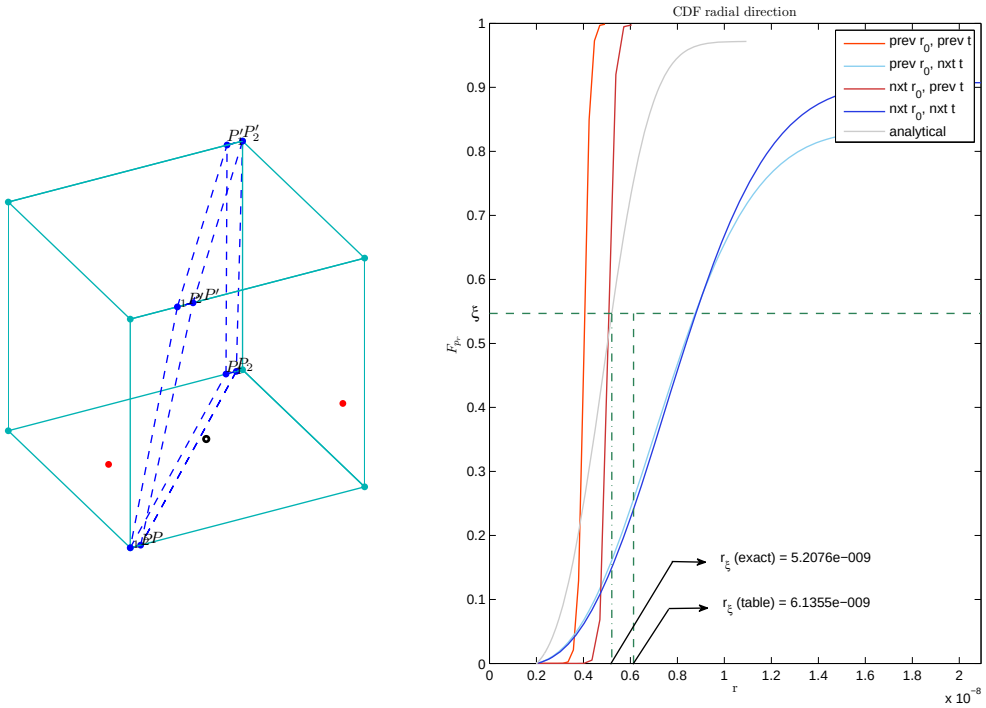
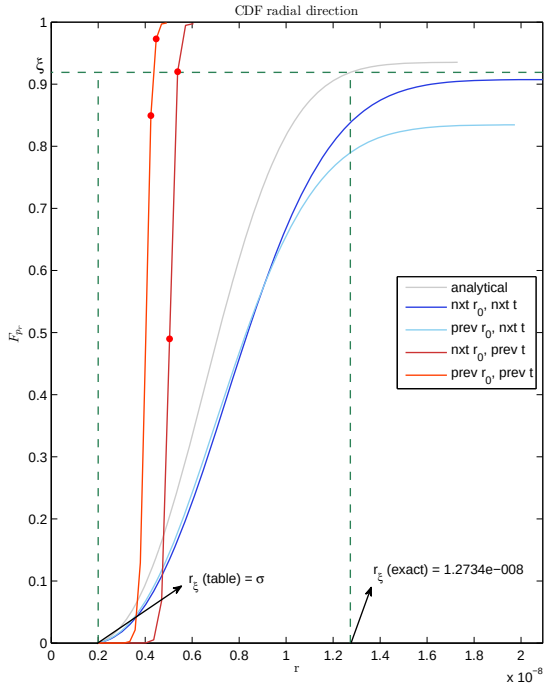


Figure 21: Another source of error in the LUT-interpolation method: no table cell can be found that contains ξ. Here $r_0 =$ `4.2768e-009`, $T =$ `6.8825e-007` and $\xi = 0.9190$.



A word of warning is required here: the LUT-interpolation method is very sensitive to large errors even though they rarely occur. As we explained above, when molecules are very close to each other or the time step is very short, the interpolation error may go far beyond the acceptable range and even accumulate if the simulation iterates over several small time steps or if during simulation the molecules frequently tend to bounce toward each other. This issue can be conclusively resolved by using high-resolution lookup tables, specifically those containing more CDFs for small initial particle positions and small time intervals. In addition to interpolation error, there are other sources of error that influence the accuracy of the LUT-interpolation method: error in computing the entries of a lookup table; the error introduced by transforming the original region of interpolation in the lookup table (table cell) into a regularly cube-shaped region (see Figure 20); and false detection of the bound state, which occurs when the table cell containing the target CDF value ξ can not be found (see Figure 21, this situation and the reason why it arises is similar to the case shown in Figure 17 on page 30).

4.1.2 Efficiency comparison

Measuring the required CPU time to compute every single solution that we used in our accuracy experiment in the previous section and drawing the box-plots of the timing results versus the same sub-ranges of r_0 and T values reveal that the numerical method, with an average run time of 0.1756 seconds, is noticeably less efficient than the other methods (see Figures 22 and 23). It can be inferred from the plots that among the sampling methods the analytical method[13], which yields the sampled position, on average, within 2.34 milliseconds, is the fastest one, approximately 75 times faster than the numerical method and 1.7 times faster than the LUT-interpolation method (with 4.2 milliseconds average CPU time per run).

Note that the speed of the numerical method is highly dependent on the grid size in the spatial domain and the total number of computational time steps used to obtain the numerical probability density function. Remember that when the initial position r_0 is close to the reaction radius σ, we have to use more grid points in the radial direction to have an accurate solution, leading to unwanted oscillations in the PDF solution that we can not avoid unless we take many more time steps, which is why the numerical method is slower in the very first sub-ranges of values for r_0. We need to mention again that the numerical method could become faster if the spatial grid is refined adaptively based on a local error estimation.

In Figures 22 and 23, one can observe that the LUT-interpolation method is a bit faster for larger values of r_0 or T, which is due to the fact that in these cases, the four vectors of CDF values, needed to get retrieved from the lookup table in the first step of the method, are relatively smaller in size, making the data retrieval faster, though not significantly.

[13]In the analytical method, CDFs can be calculated either directly using the analytical formula or by integrating the exact PDFs numerically. We assume that the accuracy of the two variations of the analytical method are the same but the latter is faster than the former with mean CPU time 3.42 ms.

Figure 22: Box-plots of the CPU time required to compute r_ξ using different sampling methods versus r_0. The data are grouped according to the value of r_0, for which the solution is computed (top); the average of the estimated times in each group is represented in the two figures at the bottom. The figure at the bottom right corner is a zoom in of the lower part of the figure to its left. Here $\xi = 0.5469$ and the experiment is performed for 91735 distinct pairs of values of r_0 and T (for every pair computations are repeated up to 10 times).

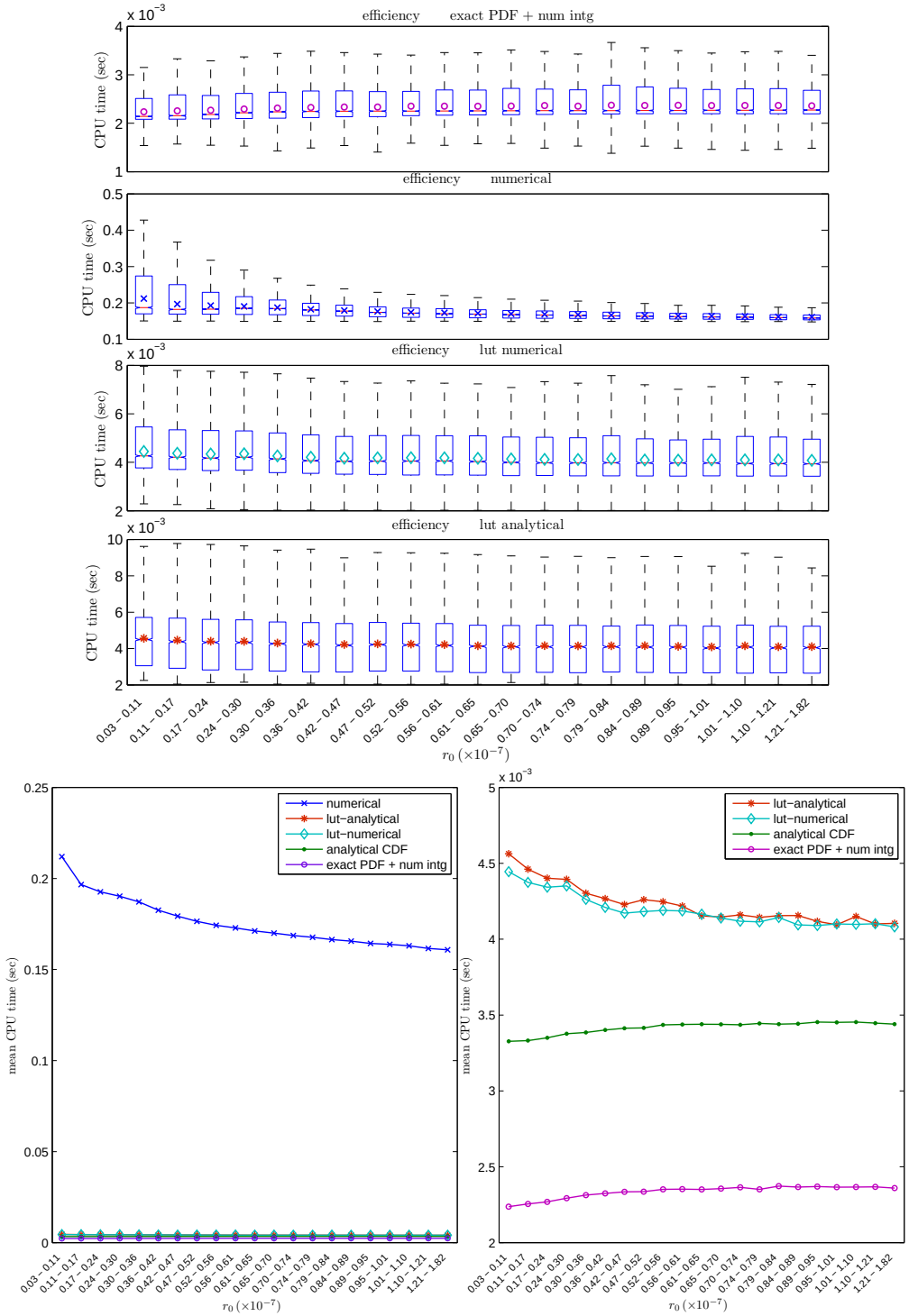
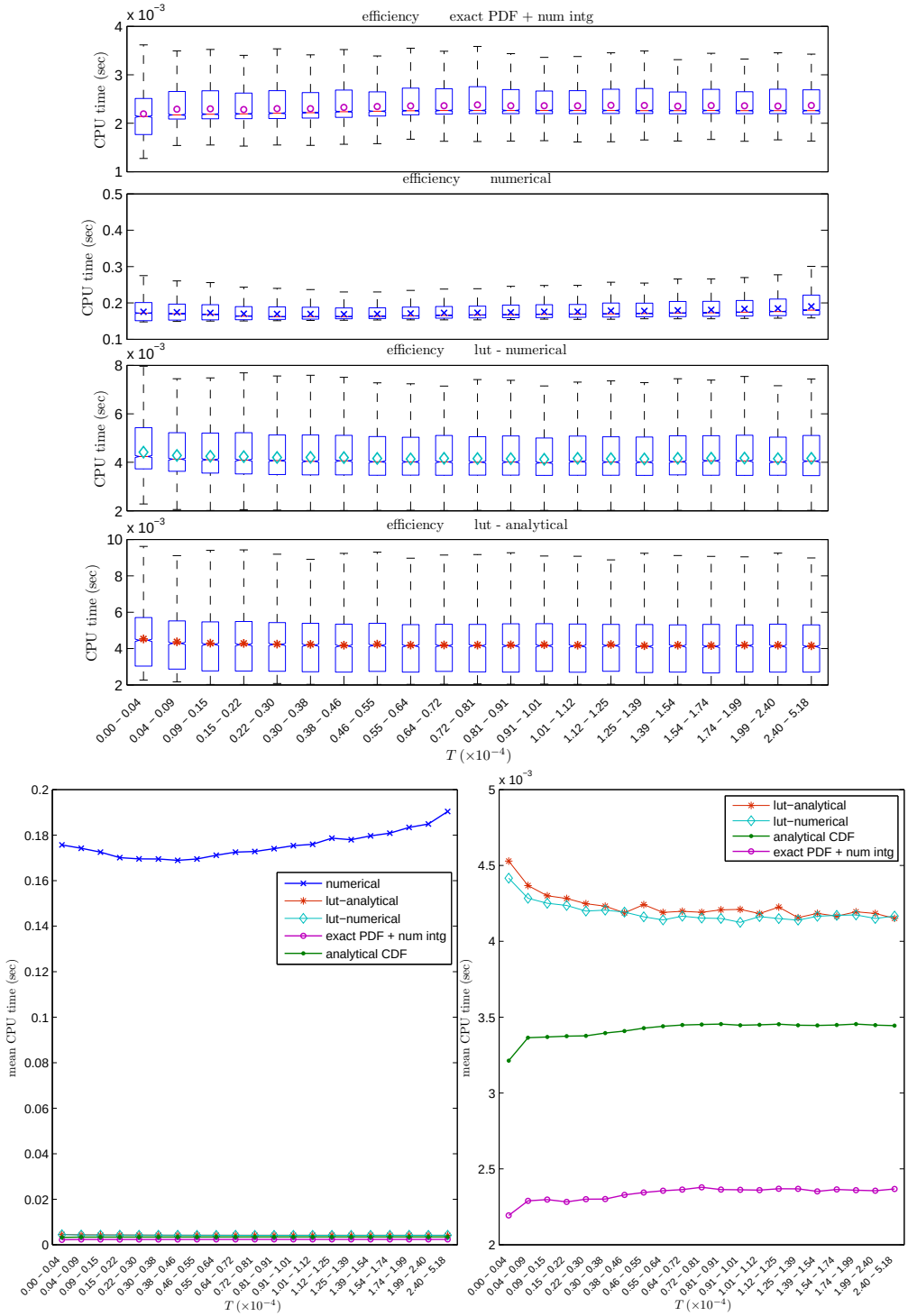


Figure 23: Box-plots of the CPU time required to compute r_ξ using different sampling methods versus T. The data are grouped according to the value of T, for which the solution is computed (top); the average of the estimated times in each group is represented in the two figures at the bottom. The figure at the bottom right corner is a zoom in of the lower part of the figure to its left. Here $\xi = 0.5469$ and the experiment is performed for 91735 distinct pairs of values of r_0 and T (for every pair computations are repeated up to 10 times).

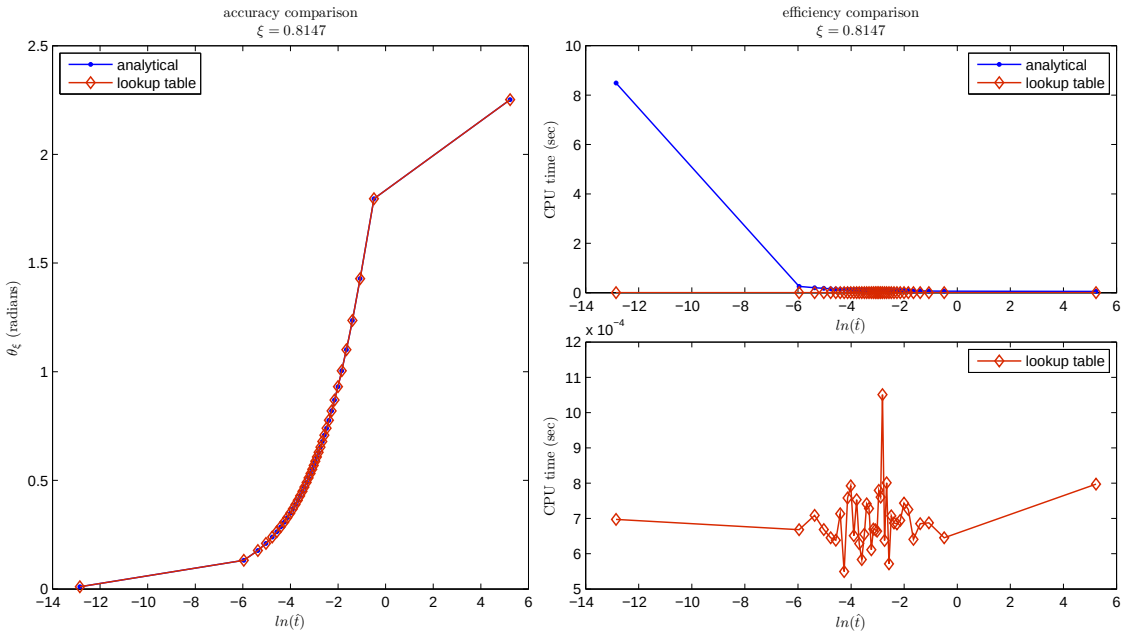


4.2 Comparison in the angular direction

To draw random angular positions according to the solution of the angular part of the Smoluchowski equation, in Section 3.2, we proposed two approaches: analytical method and interpolation between the CDF values precomputed and stored in a lookup table (LUT-interpolation method).

Figure 24 represents the solutions θ_ξ, computed by any of the above sampling methods, for 38 different values of the parameter $\hat{t}$ and a fixed uniformly distributed random number $\xi \in [0, 1]$, as a target CDF value, as well as the CPU time taken by the methods to evaluate the results, and Figure 25 on the following page provides a closer look at one of the solutions on the graphs of the cumulative distribution functions. From the figures, we can draw a primary conclusion that the accuracy of the LUT-interpolation method is high and its efficiency is better than the analytical method. We will support this hypothesis by some accuracy experiments and efficiency tests.

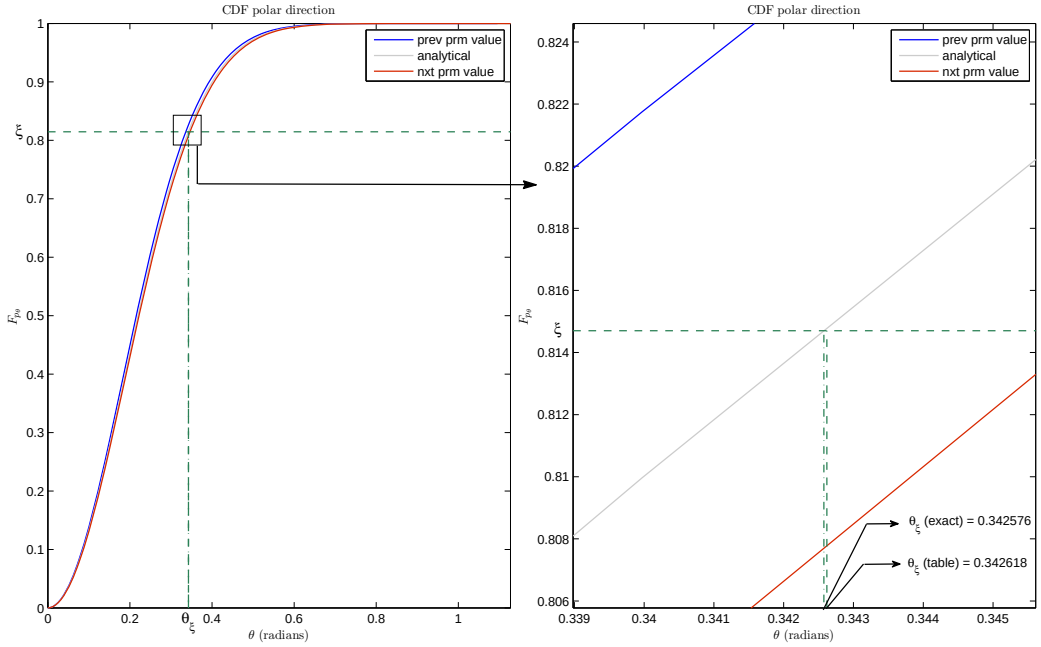
Figure 24: Comparison of the two sampling methods in the angular direction with respect to accuracy and efficiency for 38 different values of the parameter $\hat{t}$. In this figure, for every method, the computed θ_ξ (left) and the run-time required to obtain the solution (right) are plotted against the natural logarithm of $\hat{t}$. The plot in the bottom-right corner is a zoom in to the lower part of the plot in the top-right side of the figure. Here we fixed $\xi = 0.8147$ and each run is repeated up to 1000 times for every $\hat{t}$.



4.2.1 Accuracy comparison

The accuracy of the LUT-interpolation method can be estimated by comparisons with exact solutions obtained by the analytical method. Here we fix ξ and calculate 92225

Figure 25: Comparison of the different methods for the angular part of the Smoluchowski equation. The figure to the right is a zoom in to the region indicated by a rectangle on the left-hand side figure. In this example $\xi = 0.8147$, $r =$ `3.4601e-008` and $\Delta t =$ `1.0473e-005`

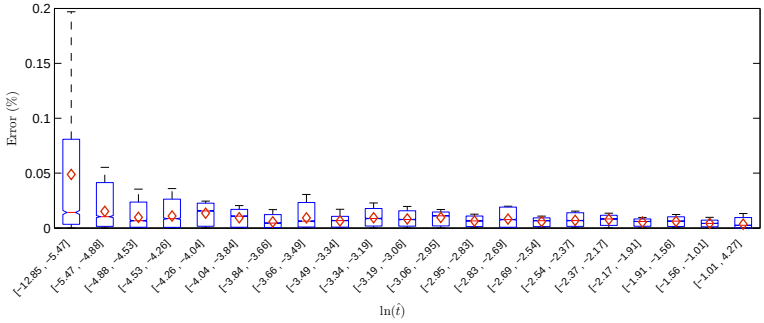


distinct values for the parameter $\hat{t}$, using all the pairs of values of Δt and r in the set `PoV` and via the definition formula $\hat{t} = \frac{D\Delta t}{r^2}$. Then for every value of $\hat{t}$, we compute θ_ξ using both sampling techniques and the relative error in the result obtained from the LUT-interpolation method.

As we did before, to study the error data we partition the whole range of value of the natural logarithm of $\hat{t}$ into 21 sub-ranges and group our error data values based on the sub-ranges and draw a box-plot, as shown in Figure 26 on the next page. In the box-plot, it can be easily seen that the accuracy of the LUT-interpolation method is very high (the average relative error is 0.01%), moreover, an example of the dependency of the accuracy on the "resolution" of the precomputed lookup table can be observed: The accuracy of the LUT-interpolation method slightly deteriorates when $\hat{t}$ is small, because the resolution of the CDF vectors[14] on the two sides of the target CDF is quite low. Hence the interpolation region, whose vertices are located on the neighbouring CDFs, is relatively extensive, which results in a larger interpolation error. Note that for small values of $\hat{t}$, the slope of the cumulative distribution function p_θ at $\theta = 0$ is very large, i.e. the function rises from zero to its maximum (approximately equal to 1) very quickly. Thus, to obtain a smooth (high-resolution) CDF, a finer grid in the polar direction, especially close to $\theta = 0$ is needed, when the parameter value $\hat{t}$ is very small.

[14]By resolution of CDF vectors, we mean the number of grid points in the polar direction, at which the cumulative distribution functions are evaluated.

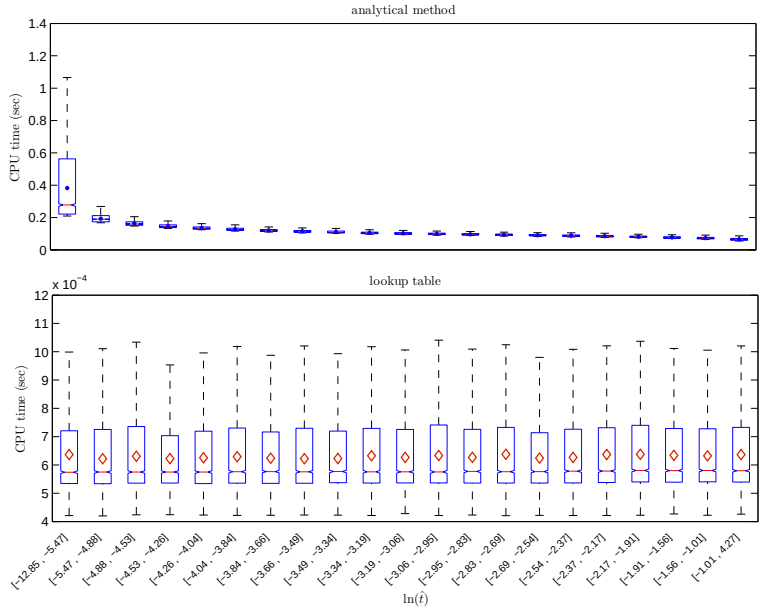
Figure 26: Box-plot of percentage error in computing θ_ξ using LUT-interpolation methods versus the natural logarithm of the parameter $\hat{t}$. Here $\xi = 0.8147$ and the experiment is performed for 92225 distinct values of $\hat{t}$, for every value of $\hat{t}$ computations are repeated up to 10 times.



4.2.2 Efficiency comparison

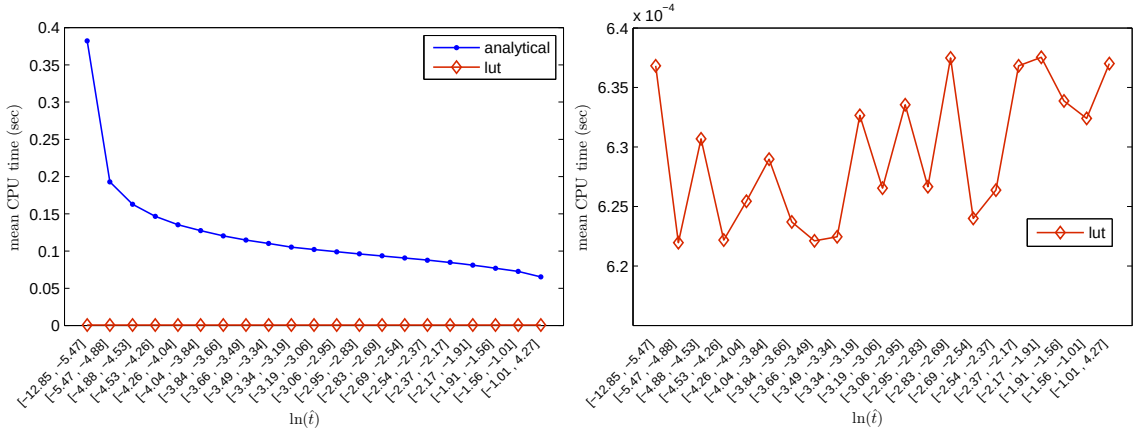
In order to compare the efficiency of the analytical method with the LUT-interpolation method, we measure the CPU time consumed by any of the methods to compute the solutions used in the above accuracy analysis and use box-plots to summarize the timing data. In Figures 27 and 28, at one glance, one can clearly see that the LUT-interpolation method is much more efficient than the analytical method (the average runtime of the LUT-interpolation method is 629 microseconds).

Figure 27: Box-plots of the CPU time required to compute θ_ξ using the analytical method and the LUT-interpolation method versus $\ln(\hat{t})$.



In Section 3.2.1, we discussed that for small values of the parameter $\hat{t}$, evaluating the probability density function p_θ via its analytical formula (3.25) can be computationally very challenging, as too many terms of the infinite series in the formula have to be taken to reach the convergence. As Figure 9 on page 21 illustrates well, the problem becomes less severe for larger values of $\hat{t}$ explaining the gradual decrease in the mean CPU time taken by the analytical method as the value of $\hat{t}$ increases.

Figure 28: The average CPU times for obtaining results in 21 sub-ranges of the values for $\hat{t}$. The right-hand side figure is a zoom in to the lower part of the figure to the left.



5 Simulation of a system of a pair of molecules

Based on the accuracy and efficiency analysis presented in the previous section, the best technique to generate particle positions in simulating reaction-diffusion processes via the event-driven algorithm by Hellander and Lötstedt [4], is sampling radial positions by the analytical method and polar positions using linear interpolation from precomputed tabulated CDF data. Hence, the algorithm can be outlined as follows:

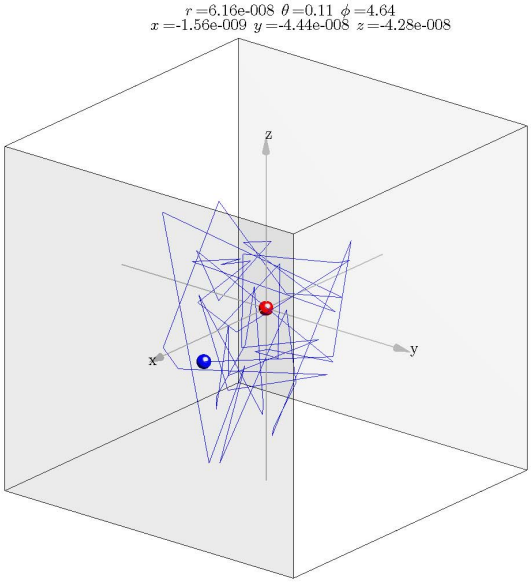
Data: a 2D lookup table storing F_{p_θ}

choose randomly the initial radial position $r_0 \geq \sigma$;
$B \leftarrow [e_1, e_2, e_3]$; `(* bases of the initial coordinate system *)`
$\theta \leftarrow 0$ and $\phi \leftarrow 0$;
repeat
  set up the *rotation matrix* $\mathcal{R}$ to rotate the current frame $\{b_1, b_2, b_3\}$, being the columns of the basis matrix B, by ϕ radians about b_2 and θ radians about b_3 ;
  $B' \leftarrow \mathcal{R} \cdot B$; `(* the basis matrix after rotation *)`
  $P \leftarrow (r_0, 0, 0)$; `(* position in the rotated frame *)`
  $P_{std} \leftarrow P \cdot B'$; `(* position in the standard Cartesian system *)`
  if *the freely moving particle leaves the simulation box* **then**
    compute P'_{std}, the new position in the standard frame, based on the type of the boundaries of the simulation region (i.e. reflective or periodic);
    set up the rotation matrix $\mathcal{R}_\alpha$ to perform rotation about the cross product of $\overrightarrow{OP_{std}}$ and $\overrightarrow{OP'_{std}}$ by α radians, the angle between the vectors ;
    $B' \leftarrow \mathcal{R}_\alpha \cdot B'$; `(* rotate the current frame again *)`
    $P_{std} \leftarrow P'_{std}$ and $r_0 \leftarrow \|\overrightarrow{OP_{std}}\|$; `(* update the position and` r_0 `*)`
  end
  $\Delta t \leftarrow \max\left\{\frac{r_0^2}{Dc}, \frac{d^2}{Dc}\right\}$, where c is a fixed constant and d is the minimum distance of the free particle from the boundaries ; `(* the time step, during which molecules react either with each other or with the boundary *)`
  ξ_1 and $\xi_2 \leftarrow$ *uniformly distributed random numbers on the interval* $[0, 1]$;
  evaluate the exact p_r for r_0 and Δt, compute its CDF, F_{p_r}, via numerical integration and calculate the radial distance r_{ξ_1} using ξ_1 and *Algorithm 1*;
  calculate the polar position θ via *Algorithm 7* using ξ_2, r_{ξ_1} and Δt as input ;
  $\phi \leftarrow$ *a uniformly distributed random number on the interval* $[0, 2\pi)$;
  if *a reaction occurs* **then**
    sample the next dissociation time according to $\mathrm{Exp}\left(\frac{1}{k_d}\right)$
  end
  $r_0 \leftarrow r_{\xi_1}$; `(* update` r_0 `*)`
  $B \leftarrow B'$
until *a predetermined final time is reached* ;

Algorithm 11: Algorithm for simulating a reaction-diffusion process involving two reactant molecules.

Using the above algorithm, we simulate a reaction-diffusion process involving two reactant molecules A and B with a diffusion coefficient $D = 2 \times 10^{-12}$ and reaction radius $\sigma = 2 \times 10^{-9}$, in two cases: a reversible reaction with $k_a = 10^{-19}$, $k_d = 2.9894$ and *free diffusion* (association and dissociation rates equal to zero). In our simulations molecule A is fixed at the origin.

Figure 29: a snapshot of a simulation of the reaction-diffusion process $A + B \underset{k_d}{\overset{k_a}{\rightleftharpoons}} C$, with $k_a = 10^{-19}$, $k_d = 2.9894$, $D = 2 \times 10^{-12}$ and $\sigma = 2 \times 10^{-9}$.



In order to check the correctness of the simulations, for each case, we repeat the simulation 1000 times and plot the normalized histograms for the obtained particle positions.

For the reversible reaction, we separately compare the spread of the molecules in the radial and polar directions with the corresponding analytical PDFs, p_r and p_θ (see Figure 30) and for the free diffusion case we make a comparison with the numerical solution of the *diffusion equation*

$$\frac{\partial p}{\partial t} = D\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2}\right), \tag{5.1}$$

subject to a Dirac initial condition at the origin and reflecting boundary conditions $\partial_x p = \partial_y p = \partial_z p = 0$ (see Figure 31). In both cases a quite nice agreement can be observed.

Figure 30: This figure verifies the LUT-interpolation method in the radial (left) and polar (right) directions for a reversible chemical process

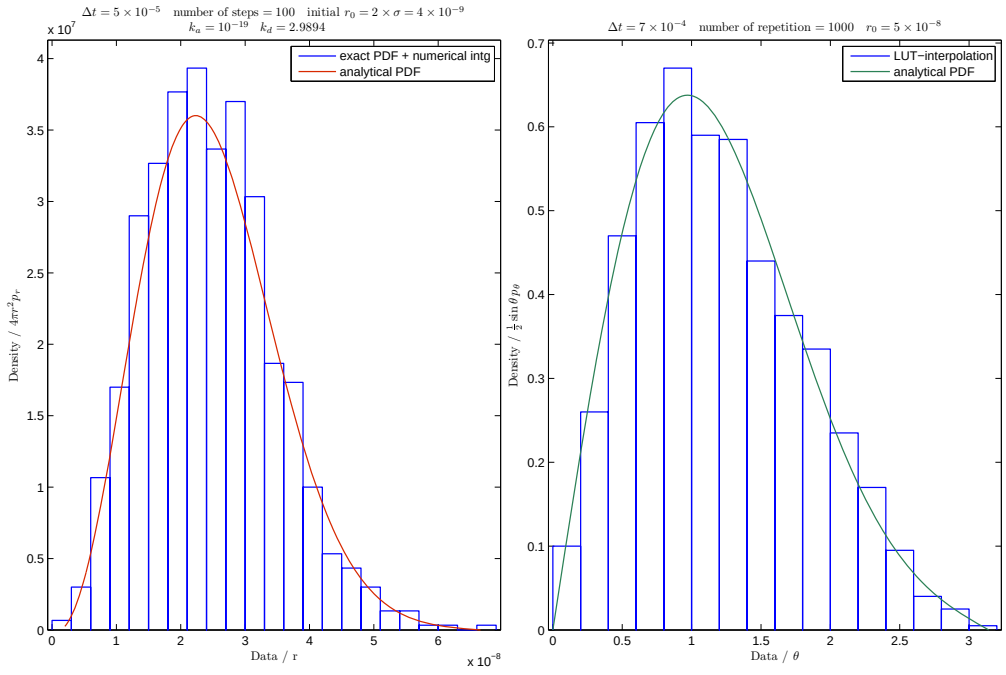
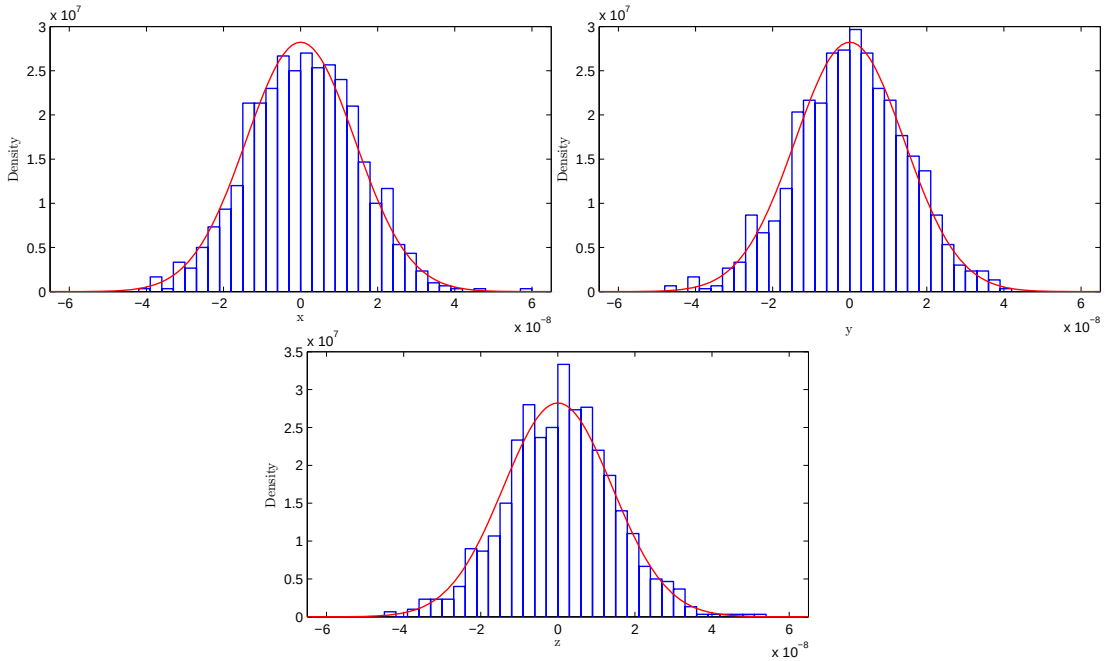


Figure 31: comparison of the probability density for the relative position of the freely diffusing molecules in the x, y, and z directions at $T = 5 \times 10^{-5}$ obtained from a simulation with the numerical solution of the diffusion equation (5.1)



6 Summary and conclusion

The GFRD algorithm [14, 15], ideally suited to simulate biochemical reaction-diffusion processes due to its event-driven time advance approach, can be considerably improved and simplified by applying an operator splitting technique in solving the Smoluchowski equation in every simulation time step (see Hellander and Lötstedt [4]). Like the original algorithm, a complex many-body system is reduced to a set of one- and two-body subsystems that are treated independently, i.e. the new positions of molecules in the subsystems are randomly sampled from the Green's function or the PDF solution of the Smoluchowski equation. The computations for a pair of molecules are simplified by splitting the equation into two subproblems that are solved separately and sequentially. The subproblems are in fact one-dimensional time-dependent PDEs, one of which gives the probability density of the radial distance between two molecules, and the other one gives the PDF for the relative polar position of the two molecules. The azimuthal position is sampled from a uniform distribution.

The focus of this study is to solve the radial and angular parts of the Smoluchowski equation, modelling the Brownian motion of a system of a pair of reactant molecules, both analytically and numerically, and comparing the accuracy and efficiency of different sampling algorithms to draw random samples from the solutions. The need for solving the equations numerically is justified by the fact that the known analytical solutions are complicated and usually computationally expensive to calculate.

We apply two methods to solve the Smoluchowski equation in the radial direction, an analytical method and a Crank-Nicolson finite difference scheme; and present and discuss three different techniques to sample according to the PDF solutions of the governing PDE, using the analytical CDF of the distribution, the CDF obtained via a numerical integration method and a numerical procedure based on linear interpolation between the entries of a 3D lookup table storing precomputed (analytical and/or numerical) CDFs corresponding to PDF solutions for a set of preselected model parameter values.

In the polar direction, we implement the analytical solution, as well as two approaches to sample from the solution: using the numerically computed CDF of the distribution and tabulating pre-evaluated CDFs in a 2D lookup table and performing linear interpolation between the table's data points.

For generating radial positions, evaluating the analytical PDF directly and sampling from its corresponding CDF, which is computed via a numerical integration scheme, is the most efficient technique. We assume that the results obtained from the method have the highest accuracy in spite of the fact that the accuracy is influenced by some error introduced by truncating infinite sums and numerical approximations performed for practical purposes. In the polar direction, sampling from lookup tables through linear interpolation is definitely more efficient. The accuracy of the method is very good, although dependent on the resolution of the table. In general, using lookup tables is recommended but it depends on the application and the amount of time that the

simulation will run, as the method has initial cost in time for creating the tables and cost in space for data storage. On the other hand, if several molecular species are interacting with each other in the biochemical system to be simulated, for any combination of the model parameter diffusion coefficient, different lookup tables need to be constructed. Furthermore, since tables are finite data sets, limitations are imposed on the longest distance that two molecules can move apart and the longest time step that can be taken during simulation.

References

[1] S. S. Andrews and D. Bray, *Stochastic Simulation of Chemical Reactions with Spatial Resolution and Single Molecule Detail*, Phys. Biol., 1 (2004), pp. 137--151.

[2] H. S. Carslaw and J. C. Jaeger, *Conduction of Heat in Solids*, Oxford University Press, Oxford, second ed., 1959.

[3] M. Dobrzyński, J. V. Rodríguez, J. A. Kaandorp, and J. G. Blom, *Computational Methods for Diffusion-Influenced Biochemical Reactions*, Bioinformatics, 23 (2007), pp. 1969--1977.

[4] S. Hellander and P. Lötstedt, *Flexible Single Molecule Simulation of Reaction-Diffusion Processes*, J. Comput. Phys., 230 (2011), pp. 3948--3965.

[5] T. Ikuma, ***FADDEEVA.m**: Faddeeva function (FFT based).* http://www.mathworks.com/matlabcentral/fileexchange/22207-faddeeva-function-fft-based , Feb 2009.

[6] H. Kim and K. J. Shin, *Exact Solution of the Reversible Diffusion-Influenced Reaction for an Isolated Pair in Three Dimensions*, Phys. Rev. Lett., 82 (1999), pp. 1578--1581.

[7] H. Kim, K. J. Shin, and M. Yang, *Dynamic Correlation Effect in Reversible Diffusion-Influenced Reactions: Brownian Dynamics Simulation in Three Dimensions*, J. Chem. Phys., 111 (1999), pp. 1068--1075.

[8] S. J. Plimpton and A. Slepoy, *Microbial Cell Modeling via Reacting Diffusive Particles*, Journal of Physics: Conference Series, 16 (2005), p. 305.

[9] C. P. Robert and G. Casella, *Monte Carlo statistical methods*, Springer Texts in Statistics, Springer-Verlag, New York, second ed., 2004.

[10] C. Sanford, M. L. K. Yip, C. White, and J. Parkinson, *Cell++ - Simulating Biochemical Pathways*, Bioinformatics, 22 (2006), pp. 2918--2925.

[11] M. Smoluchowski, *Versuch einer mathematischen Theorie der Koagulationskinetik kolloider Lösungen*, Z. Phys. Chem., 92 (1917), pp. 129--168.

[12] J. R. Stiles and T. M. Bartol, *Monte Carlo Methods for Simulating Realistic Synaptic Microphysiology Using MCell*, in Computational Neuroscience: Realistic Modelling for Experimentalists, E. D. Schutter, ed., CRC Press, 2001, pp. 87--127.

[13] N. M. Temme, *Error Functions, Dawson's and Fresnel Integrals*, in NIST Digital Library of Mathematical Functions (DLMF), National Institute of Standard and Technology (NIST), May 2010. *Available at:* http://dlmf.nist.gov/7.12#i.

[14] J. S. van Zon and P. R. ten Wolde, *Green's-Function Reaction Dynamics: A Particle-Based Approach for Simulating Biochemical Networks in Time and Space*, J. Chem. Phys., 123 (2005), p. 234910.

[15] ——, *Simulating Biochemical Networks at the Particle Level and in Time and Space: Green's-Function Reaction Dynamics*, Phys. Rev. Lett., 94 (2005), p. 128103.

[16] K. Yosida, *Brownian Motion on the Surface of the 3-Sphere*, Ann. Math. Statistics, 20 (1949), pp. 292--296.