PyRID: A Brownian dynamics simulator for reacting and interacting particles written in python

Moritz F. P. Becker

August 2022

Georg-August-University Göttingen

Contents

1 Introduction

1.1 The cell environment

Cells are crowded and inhomogeneous. Its components are constantly undergoing transitions between different states via biochemical reactions. They are translocated actively or via diffusion while their movement is restricted by the shape of cell compartments. Biomolecules move either in 3d within the cell on a 2d surface within cell membranes. New proteins are constantly synthesized while others are broken down. Interactions between molecules drive phase separation and protein aggregation. This list of processes governing cells could go on. Depending on the research question at hand, we would like to be able to account for any of these features in a model.

As an example, synaptic plasticity at excitatory synapses is mediated mainly by a change in AMPA receptor number at the postsynaptic density (PSD) (Fig. 1.1). Of the different forms of synaptic plasticity that have been discovered, long-term potentiation (LTP) is the most noted (Fig. 1.2). Therefore, it is of great interest to understand how AMPARs are trafficked. However, this question is not easily answered as AMPARs have a multitude of interaction partners which are important for AMPAR targeting.

The transmembrane domain of AMPARs forms a complex with various auxiliary proteins such as Stargazin. These auxiliary proteins interact with scaffolding proteins in the intracellular space of the postsynapse such as PSD95. In addition, trans-synaptic complexes composed, e.g., of neuroligin and neurexin bind to PS95 and presynaptic proteins such as N-cadherin are known to interact with AMPARs [Baranovic, 2021, Tanaka et al., 2012]. The many proteins at synapses result in a crowded environment. In addition, the synaptic cleft only measures 20-30 nm in width. As such, effects of molecular crowding are accompanied by geometric restrictions.

As mentioned above, the trafficking of transmembrane proteins such as AMPARs that move on a 2d surface while inside the membrane is governed also by their interaction with intracellular scaffolding proteins that move in 3d space such as PSD95. In order to fully understand AMPAR trafficking we may also want to investigate the behaviour of the intracellular interaction partners. In a series of experiments, [Zeng et al., 2016, 2018, 2019] have shown that the PSD proteins PSD95, Shank, SynGAP and Homer undergo

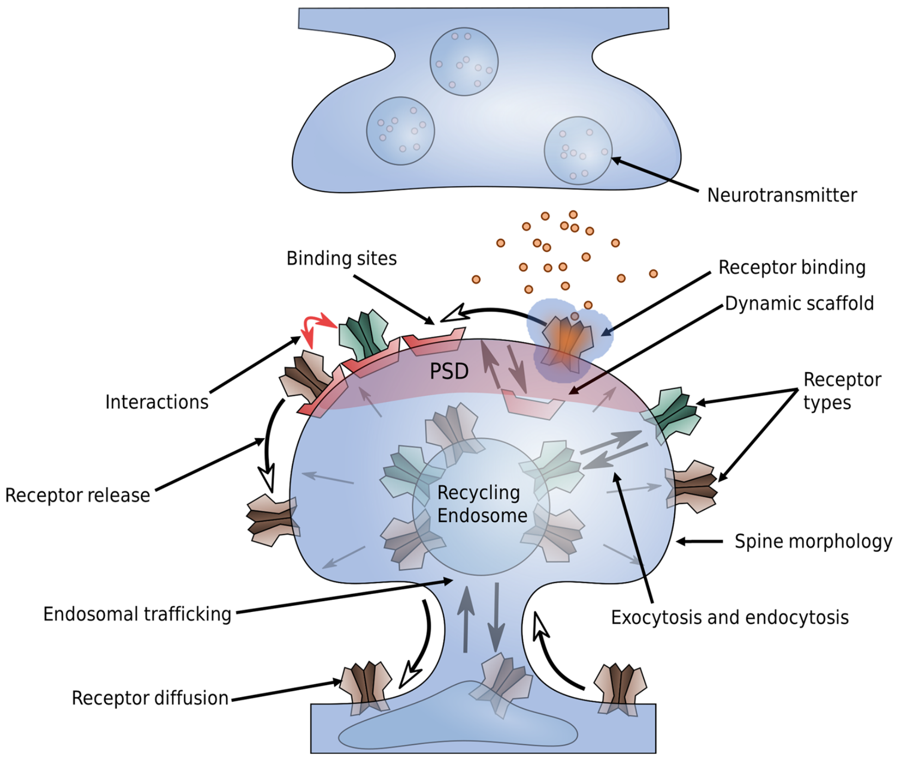


Fig 1.1. AMPAR trafficking at the synapse. Whereas LTP, LTD but also homeostatic synaptic plasticity involve many different biomolecular processes, many of them target at some point the AMPA receptor, which mediates synaptic transmission at excitatory synapses. The trafficking of AMPARs can be related to various processes such as spine growth and receptor binding that are depicted in this figure.

liquid-liquid phase separation (LLPS) in vitro, raising the assumption that also in vivo the PSD may form via a similar mechanism. Indeed, many proteins have been found to form liquid like condensates and this topic has gained a lot of attention in recent years as LLPs has many properties that seem to be important for cell functions in that it enables compartmentalization in the presence of constant protein exchange [Banani et al., 2017]. Indeed, experimental studies have found that the PSD is not entirely rigid but able to change its structure and composition under control conditions and in response to synaptic plasticity [Wegner et al., 2018, Bosch et al., 2014, Hruska et al., 2018]. This raises several questions, e.g, of how stable are PSD condensates, and under what conditions do they form? Also, how does the interaction with transmembrane proteins shape the composition, structure and stability of such condensates?

Importantly, erroneous factors that bias LLPS can lead to pathological conditions via protein aggregation. As such, LLPS might have implications for diseases such as Alzheimer, ALS, and cancer [Lu et al., 2021, Molliex et al., 2015, Wegmann et al., 2018]. A better understanding of phase separation is therefore of great interest in general as well as a possible mechanism involved in synaptic transmission and plasticity. The geometric

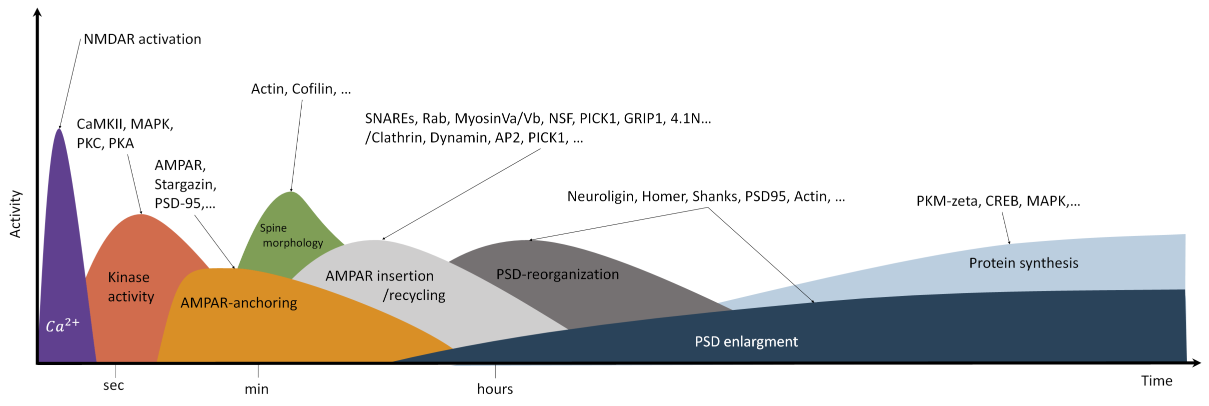


Fig 1.2. Processes involved in synaptic plasticity (long-term potentiation). Long-term potentiation involves several different processes on different time- and spatial scales. High frequency stimulation and strong postsynaptic depolarization result in the influx of Ca^{2+}. Calcium elevation occurs on the ms timescale and occurs locally in the dendrite and at dendritic spines [Frick et al., 2004]. Subsequently, calcium ions activate different kinases such as CaMKII, PKC and PKA. CaMKII in particular is essential for LTP induction. The activation of CaMKII is transient and occurs locally at the dendritic spine [Lisman et al., 2012, Lee et al., 2009]. The increased kinase activity triggers many subsequent processes. For example, phosphorylation of AMPAR subunits, PSD scaffolding proteins and auxiliary proteins such as Stargazin cause binding of additional AMPARs at the postsynaptic density [Opazo et al., 2012, Penn et al., 2017, Huganir and Nicoll, 2013, MacGillavry et al., 2011]. In addition, induction of LTP triggers spine growth, which also depends, among others, on CaMKII activation, F-actin elevation and cofilin [Fukazawa et al., 2003, Okamoto et al., 2009, Matsuzaki et al., 2004]. LTP and spine growth are accompanied by enhanced protein recycling, endocytosis and exocytosis [Park et al., 2004, 2006, Patterson et al., 2010]. Maintenance and expression of late LTP depends on the synthesis of new proteins [Abraham and Williams, 2008, Bramham, 2008]. Protein synthesis, actin dynamics and translocation result in the reorganization and growth of the PSD [Kerr and Blanpied, 2012, Bosch et al., 2014, Meyer et al., 2014, Araki et al., 2015, Hruska et al., 2018].

restrictions at the synaptic cleft are also of great importance for experimental procedures. Fluorescent labeling of proteins often involve antibodies that can be 10-20 nm in size, impairing AMPAR movement inside and in the vicinity of the synaptic cleft [Lee et al., 2017]. However, the effects of crowding and of large probes at the synaptic cleft have not been investigated apart from [Lee et al., 2017] such that older experimental results may need to be reevaluated. Computer models could help to better understand these effects and to better interpret experimental data.

To summarize, their exist several open questions of which here only very few have been addressed:

1. How is receptor trafficking influenced by different crowded environments, by the interaction with other molecules and by synapse geometry.

2. How do PSD composition and structure influence receptor distributions and vice versa.
3. Under what conditions do the PSD proteins undergo phase separation and how stable is the PSD? How is the phase behaviour of the PSD shaped by the interaction with transmembrane proteins?
4. ...

If we want to understand synaptic transmission and plasticity in more detail, we should take into account the structure of the PSD, the spine morphology and multivalent protein-protein interactions and we need to ask how these factor might influence receptor distribution and trafficking. This becomes important especially in the light of diseases. Synaptic signaling, neuronal growth etc. are robust also because their exists some redundancy in protein function. However, sometimes a small bias can result in erroneous functioning. To understand when and under what conditions these may arise we need to study not only the single proteins but we need to put these into an environment where they can interact in large populations. For this, however, we need powerful and especially flexible tools that can be relatively quickly adapted to target a new research question. In the following chapter I will introduce PyRID, a new tool for reaction diffusion simulations of interacting particles.

1.2 Modeling approaches

One big problem that arises not only in terms of modeling synaptic plasticity but for many cell processes in general are the different time and spatial scales at which these processes occur as well as the large number of complex molecules involved. A number of cell processes can be simulated on the required time scale by using simplifications. Various methods utilizing different approximations have therefore been developed (Fig. 1.3). For example, large signaling pathways can be formulated in terms of systems of ODEs, also termed reaction rate equations in this setting, or in terms of chemical master equations which can be solved efficiently using stochastic simulation algorithms such as the Gillespie SSA [Smolen et al., 2012, Johnson et al., 2021]. However, this approach neglects most aspects of the cellular environment, e.g. by assuming a well mixed, homogeneous system. Also, intrinsic time delays are not captured out of the box but can be accounted for to some degree by a multi-compartment description where molecules hop between compartments. Therefore, it is often required to add spatial dimensions. This can be done, using voxel- (3D) or lattice- (2D) based methods, i.e. we solve our chemical master equation per voxel and simulate diffusion by means of molecules hopping between voxels. However, the approach does assume that within each voxel the system is well mixed and that the voxel size is much larger than the size of the molecules. Therefore, The voxel based approach is mainly useful for large scale simulations but breaks down for smaller systems. System scales above μm and on time scales of minutes are feasible. In this regard, the synapse and sub-synaptic structures such as the PSD have to be considered as small systems. Similarly, the ODE approach can be extended to partial differential equations to include spatial dimensions. However, the PDE approach underlies similar limitations and is not well suited for small scale simulations with just a few hundred molecules. For models of the synapse and sub-synaptic structures, particle based approaches are therefore much better suited. A review on the above discussed approaches can also be found in [Johnson et al., 2021]. There exist many particle-based simulation methods. All atom molecular dynamics simulations model the system of interest in great detail, however are not suitable to investigate processes on larger scales, i.e., consisting of hundreds to thousands of proteins and that take place on a micrometer scale (size of the synapse). As such, coarse-graining approaches are necessary where groups of atoms are approximated by single beads. The level of coarse-graining can range from single amino acids to arbitrary reductions of the polypeptide chain of the proteins [Kmiecik et al., 2016, Dignon et al., 2019]. Mesoscopic molecular dynamics takes this approach to the extreme and allows for the simulation of thousands of proteins. Instead of representing proteins per atom or per amino sequence, each protein is reduced to a minimal representation of its excluded volume, i.e. an approximation of its average rigid shape, and the interaction sites that are of interest in a specific scenario, which may be represented by a single particle. Thereby, in the minimal case, molecules are represented as patchy particles [Espinosa et al., 2019]. Tools by which either of the above approaches can be implemented are, e.g. LAMMPS, Gromacs or HooMD. Coarse graining is a large field of research itself and many problems such as defining a proper force field have to be solved. However, a discussion on this topic is way beyond the scope of this work. A review on coarse grained protein models can be found ,e.g., in [Kmiecik et al., 2016]. Whereas mesoscopic molecular dynamics allows for the simulation of the interaction of hundreds of proteins, reaction kinetics are, however, usually neglected. The particle based reaction diffusion approach simulates the Brownian dynamics of molecules and also includes stochastic simulation algorithms for uni- and biomolecular reactions [Kerr et al., 2008, Anderson et al., 2020]. However, most particle-based reaction diffusion models represent molecules as points and neglect force fields and pairwise particle interactions. Therefore, whereas this simplification allow for even larger simulations than the mesoscopic MD approach with 10 thousands of molecules and time scales on the order of $\mu s - s$, it is often unrealistic, especially in crowded environments. Popular tools for particle-based reaction diffusion simulations are, e.g., MCell (CellBlender) and Smoldyn. The only particle-based reaction diffusion simulator that accounts for particle interactions via potential energy functions is ReaDDy. However, ReaDDy does out of the box not support simulations in arbitrary 3D geometries and is also not well suited for mesoscopic modeling as defined above as it does not support rigid bead models of molecules. Therefore, I here introduce a new tool named PyRID, which is a Python based simulator for reaction-diffusion models of interacting particles. PyRID runs about as fast as ReaDDy, is flexible and modifiable and supports 3D mesh geometries, surface diffusion, rigid bead models and many types of uni and bimolecular reactions. A comparison of features between PyRID, ReaDDy, MCell and Smoldyn is shown in Fig. 1.4.

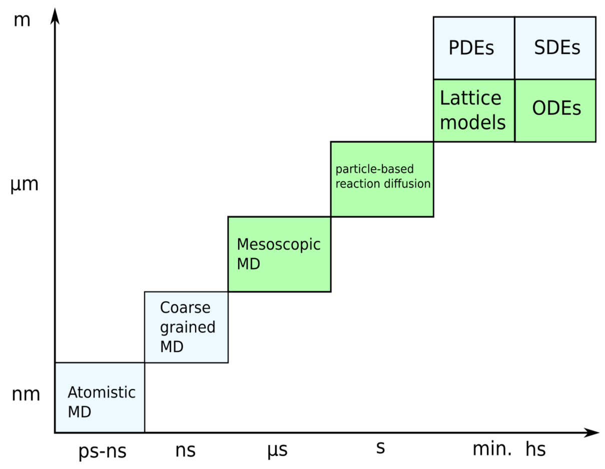
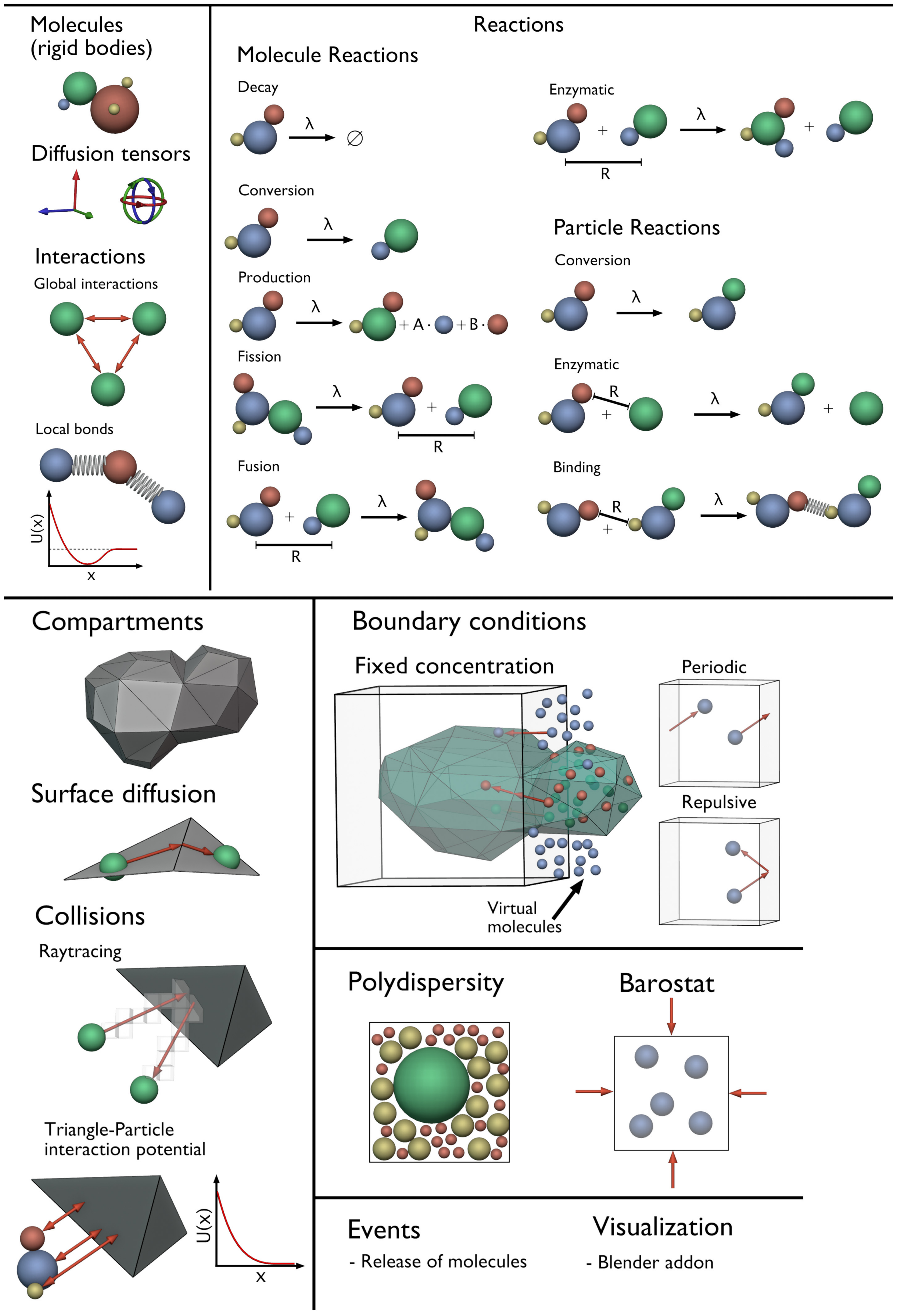


Fig 1.3. Modeling approaches. Independent of whether one models the individual signaling pathways or processes such as exocytosis and spine growth, the question of the method is important. As discussed in the text, all-atom molecular dynamics are in principle capable of replicating all the processes we are interested in but are by far too computationally expensive. Such, only simulations on the nm spatial and the ns time-scale are possible. Therefore, coarse graining and mean field approaches are necessary. These, however, come at the cost of of neglecting many details. At the extreme, ordinary differential equations neglect any spatial and stochastic properties and are therefore suited to model processes on very long time scales. Intermediate approaches such as particle based reaction diffusion simulations or mesoscopic molecular dynamics are able to simulate biochemical reactions and molecular interactions by rate based approaches and coarse grained force fields. Which method to use strongly depends on the scientific question.

Features	PyRID	ReaDDy	MCell	Smoldyn
Reactions	🟢 Very Good (zeroth order, unimolecular, bimolecular, arbitrary number of products, compartment specific, different reaction paths)	🟢 Very Good (not compartment specific however!)	🟢 Excellent (Integration with BioNetGen)	🟢 Excellent (Integration with BioNetGen)
Reaction accuracy	🟡 Volume: Good (Not exact close to boundary, reversible fusion reactions of interacting particles do not obey detailed balance) 🟡 Surface: Good (euclidian distance only)	🟢 Volume: Very Good (Not exact close to boundary, reversible fusion reactions obey detailed balance) 🟡 Surface: Good (euclidian distance only)	🟢 Volume: Very Good, 🟢 Surface: Very Good	🟢 Volume: Very Good, 🟢 Surface: Very Good
Diffusion	🟢 Anisotropic translational and rotaional diffusion, integrated diffusion tensors calculation	🟡 Isotropic translational diffusion	🟡 Isotropic translational diffusion	🟡 Anisotropic translational diffusion
Molecular structure	🟢 Molecules modeled explicitly (by interaction potential and /or rigid bodies).	🟡 Molecules modeled explicitly (only by interaction potential).	🔴 Indirectly by internal state variables (only point particles).	🔴 Indirectly by internal state variables (spherical particle approximation).
Surfaces	🟢 Arbitrary surfaces (triangulated mesh, supports obj. format)	🟡 Only via external potentials (Box and Sphere)	🟢 Arbitrary surfaces triangulated mesh, blender interface)	🟢 Arbitrary surfaces (6 elementary shapes, custom format)
Interactions	🟢 Selection of several pair-potentials, custom potentials can be added easily.	🟢 Selection of 4 potentials, custom potentials require altering C++ source code.	🔴 No Interactions	🔴/ 🟡 Excluded volume approximation for spheres.
Boundary Conditions	🟢 Periodic, Repulsive, Fixed concentration	🟡 Periodic, Repulsive	🟢 Periodic, Repulsive, Fixed concentration	🟢 Periodic, Repulsive, Fixed concentration
Polydispersity	🟢 Efficient simulation of polydisperse system by the use of a hierarchical grid data structure	🟡 Polydisperse systems result in performance drop.	Does not apply	Does not apply
API	🟢 Python	🟢 Python	🟢 Blender GUI, Python	🟢 Python, Text based
Modifiability	🟢 Excellent (All source code in python, little dependencies)	🟡 Ok (Requires changing C++ source code)	🟡 Ok (Requires changing C++ source code)	🟡 Ok (Requires changing C++ source code, Libsmoldyn API)

Fig 1.4. Feature comparison. Please note that this feature comparison is not complete and biased towards PyRID as only the main features of PyRID are compared to the other tools. Each of the tools mentioned here have some unique abilities and features that are not necessarily supported by the other tools or PyRID. However, to do an all-encompassing comparison would go beyond the scope of this work.

PyRID: A Brownian dynamics simulator for reacting and interacting particles written in python.



2 Theory and Methods

In this section, I will introduce and discuss the main methods used in PyRID. I start in section 2.1 by introducing the scheme by which bead molecules are represented. Followed by the derivation of an algorithm for the propagation of translational and predominantly rotational diffusion. The rotational and translational mobility tensors dictate the translational and rotational motion of anisotropic rigid bodies. Therefore, in section 2.3 I outline the calculation of the mobility tensors based on a modified Oseen tensor [Carrasco and de la Torre, 1999a]. One of the main features of PyRID that distinguishes it from other molecular dynamics tools such as LAMMPS, Gromacs and HooMD is the ability to simulate arbitrary unimolecular and bimolecular reactions using stochastic simulation algorithms. In section 2.7 I describe how these reactions are evaluated in PyRID. Another notable feature of PyRID is its ability to restrict the motion of molecules to complex compartment geometries represented by triangulated meshes. Section 2.5 gives a brief overview of how compartment collisions and surface diffusion are handled. The remainder of this chapter discusses some additional methods and algorithms: Distribution of molecules in mesh volumes and on surfaces, fast data structures for molecular dynamics simulations, fixed concentration boundary conditions and barostat/pressure calculation for rigid bead model systems.

2.1 Rigid bead molecules

Proteins and other molecules are not point like particles. Especially the interactions between proteins are not accurately described by isotropic energy potentials. Instead, the physical properties of bio-molecular systems emerge from an-isotropic multivalent interactions [Dignon et al., 2018, Espinosa et al., 2019]. Protein-protein interaction can be accurately simulated in all-atom molecular dynamics simulations. However, even modern computers and algorithms are not efficient enough to simulate systems with more than a few molecules on time scales relevant for processes such as protein assembly and LLPS. Therefore, coarse graining methods are needed [Tozzini, 2005]. Rigid bead models are a method of minimal coarse graining that have some important benefits. Strong and short ranged interactions between atoms are replaced by a rigid bead topology. This allows for integration time steps several orders larger than in all-atom simulations when atoms within the molecule are held together by an energy potential. Usually, the beads of a rigid bead model do not represent single atoms but pattern the geometry of the molecule of interest [de la Torre, 2001], significantly reducing the overall number of particles that need to be simulated. In addition, experimentally or theoretically estimated diffusion tensors can be used to accurately describe the diffusive motion of molecules. Importantly, multivalent protein-protein interactions can be described by patches located on the bead model surface. On the downside, the properties of coarse grained model systems strongly depend on the choice of interaction potentials and other model parameters. The estimation of these model parameters is fairly involved and is out of the scope of this work.

The position of each bead i of molecule j can be characterized by

$$\boldsymbol{R}_i(t) = \boldsymbol{R}_i^{local}(t) + \boldsymbol{R}_j^{O}(t) \tag{2.1}$$

where

$$\boldsymbol{R}_i^{local}(t) = \boldsymbol{A}_j(t) \cdot \boldsymbol{X}_i^{local}. \tag{2.2}$$

Here $\boldsymbol{X}_i^{local}$ are the coordinates of bead i in the local reference frame, and $\boldsymbol{A}(t)$ and $\boldsymbol{R}_j^{O}(t)$ are the rotation matrix and center of diffusion of molecule j in the lab reference frame respectively. The center of diffusion propagates in response to external forces $\boldsymbol{F}(t)$ exerted, e.g., by particle-particle interactions or an external force field, and due to hydrodynamic interactions and collisions of the beads with solvent molecules (Brownian motion). Thereby, the total force $\boldsymbol{F}(t)$ acting on the molecules' center of diffusion is the sum of all forces $\boldsymbol{f}_i(t)$ acting on the individual beads:

$$\boldsymbol{F}(t) = \sum_{i=1}^{N_{beads}} \cdot \boldsymbol{f}_i(t), \tag{2.3}$$

where N_{beads} is the total number of beads contained in the molecule.

2.2 Propagation of translational and angular motion

The motion of an isolated rigid bead molecule j in solution can be described in terms of the Langevin equation for translational and rotational motion. Note that we are always considering isolated molecules in dispersion and do not account for the hydrodynamic interaction between molecules as this is computationally very expensive $(O(N^2) - O(N^3))$ [Geyer and Winter, 2009, Długosz and Trylska, 2011]. In the most general case the Langevin equation for translational and rotational motion reads [Ermak and McCammon, 1978, Dickinson et al., 1985, Jones and Pusey, 1991]:

$$m\frac{d^2\boldsymbol{r}_j(t)}{dt^2} = \boldsymbol{F}_j - \Big(\boldsymbol{\Xi}^{tt}\frac{d\boldsymbol{r}_j}{dt} + \boldsymbol{\Xi}^{tr}\frac{d\boldsymbol{\phi}_j}{dt}\Big) + \boldsymbol{R}^t \tag{2.4}$$

$$\frac{d}{dt}\Big(I\frac{d\boldsymbol{\phi}_j(t)}{dt}\Big) = \boldsymbol{T}_j - \Big(\boldsymbol{\Xi}^{rr}\frac{d\boldsymbol{\phi}_j}{dt} + \boldsymbol{\Xi}^{rt}\frac{d\boldsymbol{r}_j}{dt}\Big) + \boldsymbol{R}^r, \tag{2.5}$$

where $\boldsymbol{r}_j(t)$ is the position of the molecule center and $\boldsymbol{\phi}_j(t)$ the rotation angle. $\boldsymbol{F}_j$ is the total force exerted on molecule j and $\boldsymbol{T}_j$ is the torque. $\boldsymbol{R}^t$ and $\boldsymbol{R}^r$ describe the random, erratic movement of the molecule due to collisions with the solvent molecules where

$$\langle \boldsymbol{R}^a(t)\rangle = 0 \tag{2.6}$$

$$\langle \boldsymbol{R}^a(t)\boldsymbol{R}^b(t')\rangle = 2k_BT\boldsymbol{\Xi}^{ab}_{ij}\delta(t-t'), \tag{2.7}$$

with $a, b \in \{t, r\}$. Here, $\boldsymbol{\Xi}^{tt}, \boldsymbol{\Xi}^{rr}, \boldsymbol{\Xi}^{tr}, \boldsymbol{\Xi}^{rt}$ are the translational, rotational and translation-rotation coupling friction tensors of the rigid body in the lab frame. Also, $\boldsymbol{\Xi}^{ab} = k_BT(\boldsymbol{D}^{-1})^{ab}$ (Einstein relation). Due to the translation-rotation coupling, the equations for rotation and translation are not independent. For low-mass particles, such as molecules, and for long enough time intervals, the acceleration of the molecules can be neglected in the description of the diffusion process. As such it is convenient to describe the motion of molecules by overdamped Langevin dynamics also called Brownian motion where $I\frac{d^2\phi_j(t)}{dt^2} = m\frac{d^2x_j(t)}{dt^2} = 0$:

$$\frac{d\boldsymbol{r}_j(t)}{dt} = \boldsymbol{M}^{tt}_j\boldsymbol{F}_j + \boldsymbol{M}^{tr}_j\boldsymbol{T}_j + \boldsymbol{S}^t \tag{2.8}$$

$$\frac{d\boldsymbol{\phi}_j(t)}{dt} = \boldsymbol{M}^{rr}_j\boldsymbol{T}_j + \boldsymbol{M}^{rt}_j\boldsymbol{F}_j + \boldsymbol{S}^r. \tag{2.9}$$

with

$$\langle \boldsymbol{S}^a(t)\rangle = 0 \tag{2.10}$$

$$\langle \boldsymbol{S}^a(t)\boldsymbol{S}^b(t')\rangle = 2k_BT\boldsymbol{M}^{ab}_{ij}\delta(t-t'), \tag{2.11}$$

where $\boldsymbol{M}^{tt}, \boldsymbol{M}^{rr}, \boldsymbol{M}^{tr}, \boldsymbol{M}^{rt}$ are the translational, rotational and translation-rotation coupling mobility tensors of the rigid body in the lab frame and $\boldsymbol{M}^{ab} = \frac{\boldsymbol{D}^{ab}}{k_BT}$. Also $\boldsymbol{M}^{rt} = \boldsymbol{M}^{tr,T}$. In most cases, the effect of the translation-rotation coupling on the molecular dynamics is negligible. However, translation-rotation coupling increases the complexity of the propagation algorithm for the translation and rotation vectors. Therefore, in the following, we will consider translation and rotation as being independent. In this case, the propagator for the Cartesian coordinates as well as the orientation angle can be formulated as [Ilie et al., 2015]

$$\boldsymbol{r}_j(t) = \boldsymbol{r}_j(t-\Delta t) + \boldsymbol{A}_j \boldsymbol{M}_j^{tt,b} \boldsymbol{A}_j^T \boldsymbol{F}_j \Delta t + \boldsymbol{A}_j \sqrt{2\boldsymbol{M}_j^{tt,b} k_B T}\, \boldsymbol{W}^t(\Delta t) \tag{2.12}$$

$$\boldsymbol{\phi}_j(t) = \boldsymbol{\phi}_j(t-\Delta t) + \boldsymbol{A}_j \boldsymbol{M}_j^{rr,b} \boldsymbol{A}_j^T \boldsymbol{T}_j \Delta t + \boldsymbol{A}_j \sqrt{2\boldsymbol{M}_j^{rr,b} k_B T}\, \boldsymbol{W}^r(\Delta t). \tag{2.13}$$

Here, $\boldsymbol{W}(\Delta t)$ is a 3-dimensional Wiener process, i.e. $\boldsymbol{W}(t+\Delta t) - \boldsymbol{W}(t) \sim \mathcal{N}(0, \Delta t)$, which can be argued from the central limit theorem and the assumption that the forces of the solvent molecules act with equal probability from all directions. The superscript b indicates that the mobility tensors $\boldsymbol{M}^{ab,b}$ are given in terms of the body/local frame of the molecule, which is much more convenient when we talk about the propagation algorithm. In this context, $\boldsymbol{A}_j$ is the rotation matrix of molecule j. One problem with the rotational equation of motion is that several issues arise depending on how rotations are represented. Propagating the rotation in terms of Euler angles, e.g., will result in numerical drift and singularities [Baraff, 2001, Ilie et al., 2016]. Therefore, especially in computer graphics, it is standard to represent rotations in unit quaternions, which is much more stable and has fewer issues in general. An algorithm for the rotation propagator based on quaternions can, for example, be found in [Ilie et al., 2015]. In the following, I will introduce a more concise derivation of the very same algorithm.

2.2.1 Quaternion propagator

The orientation/rotation of the molecule can be described by a unit quaternion $\boldsymbol{q} = q_0 + i\, q_1 + j\, q_2 + k\, q_3$ where $\boldsymbol{q}^2 = \sum_{i=0}^3 q_i^2 = 1$. Quaternions can be thought of as an extension to complex numbers and were introduced in 1844 by Sir William Rowan Hamilton [Hamilton, 1844]. The rotation quaternion $\boldsymbol{q}(t)$ propagates in response to the torque $\boldsymbol{T}_i(t) = \boldsymbol{F}_i(t) \times \boldsymbol{r}_{ij}$ exerted by the external forces, where $\boldsymbol{r}_{ij}$ is the distance vector between bead i and the center of diffusion of molecule j. The rotation matrix can be represented in terms of rotation quaternions by [Baraff, 2001]:

$$\boldsymbol{A} = \begin{pmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 1-2(q_1^2+q_2^2) \end{pmatrix}, \tag{2.14}$$

The goal is to derive a propagator for the rotation quaternion. A well-established connection between the angular velocity and the unit quaternion velocity is [Baraff, 2001]:

$$\frac{d\boldsymbol{q}}{dt} = \frac{1}{2}\frac{d\boldsymbol{\phi}}{dt}\boldsymbol{q} = \boldsymbol{B}\frac{d\boldsymbol{\phi}}{dt} \tag{2.15}$$

where

$$\boldsymbol{B} = \frac{1}{2}\begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{pmatrix}. \tag{2.16}$$

Inserting 2.13 into 2.15, we get:

$$\boldsymbol{q}_j(t) = \boldsymbol{q}_j(t-\Delta t) + \boldsymbol{B}_j\boldsymbol{A}_j\boldsymbol{M}_j^{rr,b}\boldsymbol{A}_j^T\boldsymbol{T}_j\Delta t + \boldsymbol{B}_j\boldsymbol{A}_j\sqrt{2\boldsymbol{M}_j^{rr,b}k_BT}\,\boldsymbol{W}^r(\Delta t). \tag{2.17}$$

The factor $\boldsymbol{BA}$ can, however, be simplified:

$$\begin{aligned}\boldsymbol{BA} =& \frac{1}{2}\begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{pmatrix}\begin{pmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 1-2(q_1^2+q_2^2) \end{pmatrix} \\ =& \frac{1}{2}\begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3(1-2q^2) & q_2(2q^2-1) \\ q_3(2q^2-1) & q_0 & q1(1-2q^2) \\ q_2(1-2q^2) & q_1(2q^2-1) & q_0 \end{pmatrix} \\ =& \frac{1}{2}\begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{pmatrix},\end{aligned} \tag{2.18}$$

where $q^2 = q_0^2+q_1^2+q_2^2+q_3^2 = 1$. For the quaternion to accurately represent the rotation, we need to ensure that it keeps its unit length. However, due to the finite time step in simulations, the quaternion will diverge from unit length over time. Thus, it is necessary to frequently re-normalize the quaternion. Ilie et al. [2015] point out that re-normalization will introduce a bias by changing the sampled phase space distribution. Thereby, it is more appropriate to introduce a constraint force using the method of undetermined Lagrange multipliers as is used in molecular dynamics algorithms such as SHAKE. However, for integration time steps used in practice, I found the error introduced by re-normalization to be negligible. Validation of the above algorithms are presented in section 3.1.

2.3 Mobility tensor for rigid bead models

In order simulate the motion of molecules with the algorithms introduced above, we need to calculate the molecule's diffusion tensor. Diffusion tensors have also been estimated experimentally [Niethammer et al., 2006] or using molecular dynamics simulations [Chevrot et al., 2013]. However, for many proteins, the diffusion tensor is unknown. Therefore, it would often be more convenient to calculate the diffusion tensor directly from the coarse-grained representation of a molecule in terms of the rigid bead model. Pioneering work in this direction has been done by Bloomfield et al. [1967] and Torre and Bloomfield [1977a]. In the following I will only present the main results that are needed for the calculation of the rigid bead model diffusion tensor. For the interested reader, a more in depth introduction can be found in 5.1 Appendix A.

In general, the mobility and/or diffusion tensor of an anisotropic rigid body can be calculated from the inverse of the rigid body's friction supermatrix [Carrasco and de la Torre, 1999a]:

$$\begin{pmatrix} \boldsymbol{M}^{tt} & \boldsymbol{M}^{tr,T} \\ \boldsymbol{M}^{rt} & \boldsymbol{M}^{rr} \end{pmatrix} = \frac{1}{k_B T} \begin{pmatrix} \boldsymbol{D}^{tt} & \boldsymbol{D}^{tr,T} \\ \boldsymbol{D}^{rt} & \boldsymbol{D}^{rr} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Xi}^{tt} & \boldsymbol{\Xi}^{tr,T} \\ \boldsymbol{\Xi}^{rt} & \boldsymbol{\Xi}^{rr} \end{pmatrix}^{-1}. \tag{2.19}$$

Therefore, the main challenge lies in deriving an expression for the translational, rotational and translation-rotation coupling tensors of the friction super matrix $\boldsymbol{\Xi}^{tt}, \boldsymbol{\Xi}^{rr}, \boldsymbol{\Xi}^{tr} = \boldsymbol{\Xi}^{rt,T}$. PyRID uses a method based on a modified Oseen tensor [Torre and Bloomfield, 1977a, Carrasco and de la Torre, 1999a] to account for the hydrodynamic interaction between the beads of a rigid bead molecule in a first order approximation [Carrasco and de la Torre, 1999b]. For a rigid molecule consisting of N different beads, the friction tensors read

$$\begin{aligned}
\boldsymbol{\Xi}^{tt} &= \sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{\xi}_{ij}^{tt} \\
\boldsymbol{\Xi}_O^{tr} &= \sum_{i=1}^{N}\sum_{j=1}^{N} (-\boldsymbol{\xi}_{ij}^{tt} \cdot \boldsymbol{A}_j + \boldsymbol{\xi}_{ij}^{tr}) \\
\boldsymbol{\Xi}_O^{rt} &= \sum_{i=1}^{N}\sum_{j=1}^{N} (\boldsymbol{A}_j \cdot \boldsymbol{\xi}_{ij}^{tt} + \boldsymbol{\xi}_{ij}^{rt}) \\
\boldsymbol{\Xi}_O^{rr} &= \sum_{i=1}^{N}\sum_{j=1}^{N} (\boldsymbol{\xi}_{ij}^{rr} - \boldsymbol{\xi}_{ij}^{rt} \cdot \boldsymbol{A}_j + \boldsymbol{A}_i \cdot \boldsymbol{\xi}_{ij}^{tr} - \boldsymbol{A}_i \cdot \boldsymbol{\xi}_{ij}^{tt} \boldsymbol{A}_j)
\end{aligned} \quad . \tag{2.20}$$

Here $\boldsymbol{A}$ is given by

$$\boldsymbol{A}_i = \begin{pmatrix} 0 & -z_i & y_i \\ z_i & 0 & -x_i \\ -y_i & x_i & 0 \end{pmatrix} \tag{2.21}$$

with $\boldsymbol{r}_i = x_i e_x + y_i e_y + z_i e_z$ being the position vector of bead i in the molecule's local reference frame. $\boldsymbol{\xi}^{ab}, a, b \in \{t, r\}$ are the translational, rotational and translation-rotation coupling friction tensors of the system of N freely diffusing beads. $\boldsymbol{\xi}$, are calculated from the inverse of the mobility supermatrix [Carrasco and de la Torre, 1999b]:

$$\begin{pmatrix} \boldsymbol{\xi}^{tt} & \boldsymbol{\xi}^{tr} \\ \boldsymbol{\xi}^{rt} & \boldsymbol{\xi}^{rr} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mu}^{tt} & \boldsymbol{\mu}^{tr} \\ \boldsymbol{\mu}^{rt} & \boldsymbol{\mu}^{rr} \end{pmatrix}^{-1} \tag{2.22}$$

Here $\boldsymbol{\xi}^{ab}$ are of dimension (3Nx3N), forming the friction supermatrix of dimension (6N,6N). $\boldsymbol{\mu}^{ab}$ are the (3Nx3N) dimensional elements of the mobility supermatrix. The translational mobility tensor $\boldsymbol{\mu}^{tt}$ for a system of different sized beads is, in first order approximation, given by [Carrasco and de la Torre, 1999b]:

$$\begin{aligned}
\boldsymbol{\mu}_{ij}^{tt} =& \delta_{ij}(6\pi\eta_0\sigma_i)^{-1}\boldsymbol{I} + (1-\delta_{ij})(8\pi\eta_0 r_{ij}^{-1})(\boldsymbol{I} + \boldsymbol{P}_{ij}) \\
&+ (8\pi\eta_0 r_{ij}^{-3})(\sigma_i^2 + \sigma_j^2)(\boldsymbol{I} - 3\boldsymbol{P}_{ij}),
\end{aligned} \tag{2.23}$$

where $\boldsymbol{P}_{ij} = \left(\boldsymbol{I} + \frac{\boldsymbol{r}\otimes\boldsymbol{r}}{r^2}\right)$, η_0 is the fluid viscosity and r_{ij} is the distance vector between bead i and bead j. $\boldsymbol{I}$ is the identity matrix. The mobility tensor for rotation, however, not accounting for the bead radii in the hydrodynamic interaction term, is given by [Carrasco and de la Torre, 1999b]:

$$\begin{aligned}\boldsymbol{\mu}_{ij}^{rr} =&\delta_{ij}(8\pi\eta_0\sigma_i^3)^{-1}\boldsymbol{I}\\ &+(1-\delta_{ij})(16\pi\eta_0 r_{ij}^3)^{-1}(3\boldsymbol{P}_{ij}-\boldsymbol{I}).\end{aligned} \tag{2.24}$$

In this formulation, there is still a correction for the bead radii missing. This correction consists of adding $6\eta_0 V_m \boldsymbol{I}$ to the diagonal components of the rotational friction tensor Ξ_O^{rr}, where V_m is the total volume of the rigid bead molecule [de la Torre and Rodes, 1983, Carrasco and de la Torre, 1999b]. The rotation-translation coupling is given by [Carrasco and de la Torre, 1999b]:

$$\boldsymbol{\mu}_{ij}^{rt} = (1-\delta_{ij})(8\pi\eta_0 r_{ij}^2)^{-1}\boldsymbol{\epsilon}\hat{\boldsymbol{r}}_{ij}, \tag{2.25}$$

where ϵ is the Levi-Civita symbol with [de la Torre et al., 2007]

$$\epsilon\cdot\boldsymbol{r}_{ij} = \begin{pmatrix} 0 & z_{ij} & -y_{ij} \\ -z_{ij} & 0 & x_{ij} \\ y_{ij} & -x_{ij} & 0 \end{pmatrix}. \tag{2.26}$$

$\boldsymbol{\mu}^{tt}, \boldsymbol{\mu}^{rr}, \boldsymbol{\mu}^{rt}$ describe the mobility of a multi-sphere system with hydrodynamic interactions. From the above follows that we need to calculate the inverse of a supermatrix twice, once in equation 5.11 and once in equation 2.20. A super Matrix $\boldsymbol{M} = [[\boldsymbol{M}_1, \boldsymbol{M}_2], [\boldsymbol{M}_3, \boldsymbol{M}_4]]$ is invertible, if both the diagonal blocks, $\boldsymbol{M}_1$ and $\boldsymbol{M}_4$ are invertible The inverse of a (2x2) supermatrix can be calculated by [Varadarajan, 2004], [Deligne and Morgan, 1996]:

$$\begin{aligned}\boldsymbol{T}_1 &= (\boldsymbol{M}_1 - \boldsymbol{M}_2\boldsymbol{M}_4^{-1}\boldsymbol{M}_3)^{-1}\\ \boldsymbol{T}_2 &= -\boldsymbol{M}_1^{-1}\boldsymbol{M}_2(\boldsymbol{M}_4 - \boldsymbol{M}_3\boldsymbol{M}_1^{-1}\boldsymbol{M}_2)^{-1}\\ \boldsymbol{T}_3 &= -\boldsymbol{M}_4^{-1}\boldsymbol{M}_3(\boldsymbol{M}_1 - \boldsymbol{M}_2\boldsymbol{M}_4^{-1}\boldsymbol{M}_3)^{-1}\\ \boldsymbol{T}_4 &= (\boldsymbol{M}_4 - \boldsymbol{M}_3\boldsymbol{M}_1^{-1}\boldsymbol{M}_2)^{-1}\end{aligned} \tag{2.27}$$

2.4 Center of Diffusion

One problem that arises with the above description is that we have not yet formulated an expression for the center of diffusion of the rigid bead molecule. For a rigid body immersed in a fluid, the force and torque act at the body's center of diffusion [Harvey and de la Torre, 1980], which, in general, is different from the center of mass except for spherically symmetric molecules. The center of diffusion can, however, be calculated from a diffusion tensor referring to an arbitrary origin by [Carrasco and de la Torre, 1999a]

$$\begin{aligned}\boldsymbol{r}_{OD} &= \begin{pmatrix} x_{OD} \\ y_{OD} \\ z_{OD} \end{pmatrix} \\ &= \begin{pmatrix} D_{rr}^{yy} + D_{rr}^{zz} & -D_{rr}^{xy} & -D_{rr}^{xz} \\ -D_{rr}^{xy} & D_{rr}^{xx} + D_{rr}^{zz} & -D_{rr}^{yz} \\ -D_{rr}^{xz} & -D_{rr}^{yz} & D_{rr}^{yy} + D_{rr}^{xx} \end{pmatrix}^{-1} \begin{pmatrix} D_{tr}^{zy} - D_{tr}^{yz} \\ D_{tr}^{xz} - D_{tr}^{zx} \\ D_{tr}^{yx} - D_{tr}^{xy} \end{pmatrix}. \end{aligned} \tag{2.28}$$

2.5 Compartments

Compartmentalization plays an important role in cell processes. Therefore, we would like to be able to restrict diffusion and reactions to the volume and surface of arbitrarily shaped compartments. There exist different methods to restrict the motion of particles to a confined region. One option is to pattern the boundary of the compartment with particles such that they interact with the particle inside the compartment via a repulsive interaction potential. This method, however, has several downsides. On the one hand, one needs many particles to pattern the surface, which makes this method highly inefficient. On the other hand this method does not support surface diffusion in a straight forward way. Another approach would be to add external potentials/force fields that restrict the motion of particles either to the volume or the surface of a compartment. This method is used in ReaDDy [Hoffmann et al., 2019]. However, complex geometries/compartment shapes are more difficult to establish. A third approach is to represent the compartment geometry by triangulated meshes as is done, e.g. in MCell [Kerr et al., 2008]. This approach has several benefits over alternative approaches, such as representing compartments by force fields or other particles. Triangulated meshes are heavily used in computer graphics. Therefore, a large number of highly optimized algorithms exist. Also, triangulated meshes are very well suited to represent complex compartment geometries. In the following, I will introduce how PyRID handles surface diffusion and collisions of particles with compartment surfaces.

2.5.1 Triangulated meshes

A triangulated mesh surface is described by a set of N vertices. These vertices are combined to sets of n vertices that form the mesh faces. In our case, each face is a triangle ($n = 3$) determined by three vertices $\boldsymbol{p}_i, \boldsymbol{p}_j$ and $\boldsymbol{p}_k$. The order in which these vertices are sorted per triangle determines the orientation of the triangle normal vector. The normal vector of the triangle plane is given by $\boldsymbol{n} = (\boldsymbol{p}_1 - \boldsymbol{p}_0) \times (\boldsymbol{p}_2 - \boldsymbol{p}_0)$. In the following, I will write the three vertices of a triangle as $\boldsymbol{p}_0, \boldsymbol{p}_1$ and $\boldsymbol{p}_2$ and vertices are always sorted in counter clockwise order (Fig. 2.1 B). Thereby, the normal vector of a triangle points outside the mesh compartment. In PyRID, a compartment is defined by a triangulated manifold mesh, which is a mesh without holes and disconnected vertices or edges, i.e. it has no gaps and separates the space on the inside of the compartment from the space outside [Shirley et al., 2009]. As one vertex is shared by at minimum three triangles it is most convenient to store meshes in a shared vertex mesh data structure [Shirley et al., 2009] where an array with all vertex position vectors is kept as well as an array holding for each triangle the indices of the three vertices that make up the triangle (Fig. 2.1 A).

2.5.2 Volume molecules

The collision response of a molecule with the mesh is calculated in two different ways. For large rigid bead molecules, each triangle exerts a repulsive force on the individual beads; for small, isotropic molecules or atoms, a ray tracing algorithm is used.

Contact forces

Contact detection generally consists of two phases, 1) neighbor searching and 2) contact resolution. Contact detection and update of contact forces can become fairly involved, depending on the required accuracy, the surface complexity, the type of geometries involved, and whether frictional forces need to be accounted for. Contact resolution of the more complex type is found primarily in discrete element method simulations [Hu et al., 2013]. Here, however, we will not require exact accuracy but instead use a simple but, as I think, sufficiently accurate approach. A bead i is said to be in contact with a mesh element j (which can be a vertex, edge, or face) if the minimum distance r_{ij} is smaller than the bead radius. In this case, a repulsive force is exerted on the bead:

$$U_{wall,i} = \sum_{j}^{N} \frac{k}{2}(r_{ij} - d)^2 \Omega_{ij}. \tag{2.29}$$

, where k is the force constant, d is the bead radius, and N is the number of faces that are in contact with bead i. In general, Ω_{ij} accounts for the amount of overlap of bead i with mesh face j. However, calculation of Ω_{ij} is computationally expensive. Therefore, we here use a simple approximation $\Omega_{ij} = 1/N$ with $N = |\mathcal{F}|$ where $\mathcal{F}$ is the set of all faces the bead is in contact with. Thereby, we assume that the bead overlaps by the same amount with each mesh element and only account for overlaps with faces as valid contacts but not edges or vertices. If $\mathcal{F} = \emptyset$, only the distance to the closest mesh element is used to calculate the repulsive force, which in this case is either an edge or a vertex.

To calculate the distance between the bead and a triangle, PyRID uses the "Point to Triangle" algorithm by Eberly [2001].

Ray tracing

Contact force calculations are disadvantageous for small, spherical molecules because they require a very small integration time step. Here, ray tracing is more convenient as it works independently of the chosen integration time step. In this approach, which is similar to the contact detection used in MCell Kerr et al. [2008], the displacement vector $\boldsymbol{\Delta R}$ of the molecule is traced through the simulation volume and collisions with the compartment boundary (the mesh) are resolved via reflection.

$$\boldsymbol{\Delta R}_{refl} = \boldsymbol{\Delta R} - 2(\boldsymbol{\Delta R} \cdot \hat{\boldsymbol{n}})\hat{\boldsymbol{n}}, \tag{2.30}$$

where $\hat{\boldsymbol{n}}$ is the normal vector of the triangle face. Collision tests are done using the "Fast Voxel Traversal Algorithm for Ray Tracing" introduced by Amanatides and Woo [1987].

2.5.3 Surface molecules

Surface molecules laterally diffuse within the mesh surface and can represent any transmembrane molecules such as receptors. Here, I take a similar approach to MCell. Thereby, a molecule diffuses in the plane of a triangle until it crosses a triangle edge. In this case, the molecule's displacement vector ΔR is advanced until that edge and then rotated into the plane of the neighboring triangle where the rotation axis is given by the shared triangle edge. Thereby, the molecule will move in a strait line on the mesh surface (Figure 2.1 C-E). This method is equivalent to unfolding the triangles over the shared edge such that they end up in a common tangent space, i.e. such that they are co-planar, advancing the position vector, and folding/rotating back. From the latter method it becomes intuitively clear that the molecule will indeed move in a straight line on the mesh surface. In the following I will introduce the details of the method sketched above.

Surface ray marching

First, we need to be able to detect if a triangle edge has been crossed, and to which neighbouring triangle this edge belongs. Therefore, in addition to the triangle and vertex data, for each triangle, the vertex indices of the three triangle edges are kept in an array (Fig. 2.1A). Edges are sorted in counter clockwise order. Also, for each of the three edges the index of the corresponding neighbouring triangle is kept in a separate array for fast lookup (Fig. 2.1A).

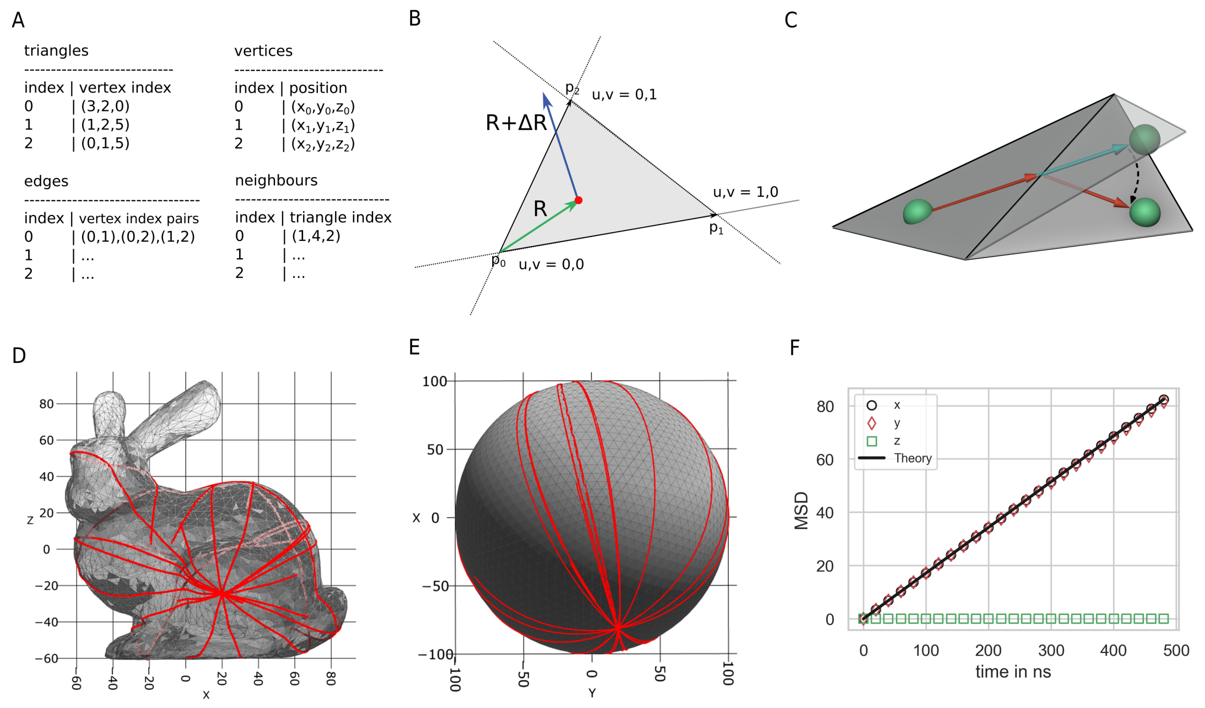


Fig 2.1. Mesh compartments and surface molecules. **(A)** PyRID uses triangulated meshes to represent compartments. These are kept in a shared vertex mesh data structure (top left, right) [Shirley et al., 2009]. In addition, for neighbour search, two array that hold for each triangle the vertex indices of the three triangle edges and the triangle indices of the three triangle neighbours are used. **(A)** Triangle vertices belonging to a triangle are ordered counterclockwise, as are edges. For in triangle and edge intersection tests barycentric triangle coordinates are used. **(A)** Visualization of mesh surface ray marching. If a molecule (green sphere) crosses a triangle edge, its displacement vector is advanced to the corresponding edge and then rotated into the plane of the neighboring triangle. **D,E** By the ray marching method described in the text, molecules follow a geodesic paths on the mesh surface. **F** The mean squared displacement of diffusing surface molecules is in agreement with theory.

The triangle edge intersection test can be made efficient by the use of barycentric coordinates. Let $\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2$ be the three vertices of a triangle. Also, the vertices are numbered in counter clockwise order and the triangle origin is at $\boldsymbol{p}_0$. Then, the center of the molecule R_0 can be described in barycentric coordinates by

$$\boldsymbol{R}_0 = \boldsymbol{p}_0 + u(\boldsymbol{p}_1 - \boldsymbol{p}_0) + v(\boldsymbol{p}_2 - \boldsymbol{p}_0), \tag{2.31}$$

and the molecule displacement vector by

$$\boldsymbol{\Delta R} = du(\boldsymbol{p}_1 - \boldsymbol{p}_0) + dv(\boldsymbol{p}_2 - \boldsymbol{p}_0), \tag{2.32}$$

Efficient algorithms to compute the barycentric coordinates u and v can, e.g., be found in [Ericson, 2004]. The triangle edges are sorted in counter clockwise order, starting from the triangle origin $\boldsymbol{p}_0$. As such, we are on the line $\boldsymbol{p}_0 + u(\boldsymbol{p}_1 - \boldsymbol{p}_0)$ (edge 0) if $v = 0$, on the line $\boldsymbol{p}_0 + v(\boldsymbol{p}_2 - \boldsymbol{p}_0)$ (edge 2) if $u = 0$ and on the line $u\boldsymbol{p}_1 + v\boldsymbol{p}_2$ (edge 1) if $u + v = 1$. Thereby, the edge intersection test comes down to solving

$$\begin{aligned} &u + t_1 \cdot du = 0 \\ &v + t_0 \cdot dv = 0 \\ &(u + t_2 \cdot du) + (v + t_2 \cdot dv) = 1, \end{aligned} \tag{2.33}$$

where t_i with $i \in \{0, 1, 2\}$ is the distances to the respective edge i along the displacement vector. We find that the intersections occur at

$$\begin{aligned} t_1 &= -\frac{u}{du} \text{ (edge 1)} \\ t_0 &= -\frac{v}{dv} \text{ (edge 0)} \\ t_2 &= \frac{1 - u - v}{du + dv} \text{ (edge 2)}. \end{aligned} \tag{2.34}$$

To determine with which edge $\boldsymbol{R} + \boldsymbol{\Delta R}$ intersects first, we simply need to check for the smallest positive value of t_i. Afterward, we advance $\boldsymbol{R}$ to the intersecting edge, reduce $\boldsymbol{\Delta R}$ by the corresponding distance traveled and transform $\boldsymbol{R}$ to the local coordinate frame of the neighboring triangle. At last, $\boldsymbol{\Delta R}$ is rotated into the plane of the neighboring triangle. This can be done efficiently using Rodrigues' rotation formula

$$\Delta \boldsymbol{R}_{rot} = \Delta \boldsymbol{R} \cos(\phi) + (\boldsymbol{a}_n \times \Delta \boldsymbol{R}) \sin(\phi) + \boldsymbol{a}_n(\boldsymbol{a}_n \cdot \Delta \boldsymbol{R})(1 - \cos(\phi)), \tag{2.35}$$

where

$$\begin{aligned} \cos(\phi) &= \frac{\hat{\boldsymbol{n}}_1 \cdot \hat{\boldsymbol{n}}_2}{|\hat{\boldsymbol{n}}_1||\hat{\boldsymbol{n}}_2|} \\ \sin(\phi) &= \frac{\hat{\boldsymbol{n}}_1 \times \hat{\boldsymbol{n}}_2}{|\hat{\boldsymbol{n}}_1||\hat{\boldsymbol{n}}_2|} \end{aligned}. \tag{2.36}$$

Here, $\hat{\boldsymbol{n}}_1$ and $\hat{\boldsymbol{n}}_2$ are the normal vectors of the two neighboring triangles. As PyRID supports anisotropic rigid bead molecules, the orientation of the molecule needs to be updated as well for each triangle that is crossed. It is not sufficient, however, to rotate the molecule only after it has reached its final position, because the final orientation depends on the exact path that is taken (in case multiple triangles are crossed) and not only on the normal vector/orientation of the target triangle plane. The rotation quaternion is given by:

$$\boldsymbol{q} = \cos(\phi/2) + \boldsymbol{a}_n \sin(\phi/2), \tag{2.37}$$

where $\sin(\phi/2)$ and $\cos(\phi/2)$ can be calculated from the half-angle formulas for sine and cosine such that the $\cos(\phi)$ and $\sin(\phi)$ that were calculated to rotate ΔR can be reused. The molecule's orientation quaternion is than propagated by quaternion multiplication. The procedure is stopped if $\boldsymbol{R}_0 + \Delta \boldsymbol{R}$ end up inside the triangle the molecule is currently located on, i.e. if $0 <= u <= 1, 0 <= v <= 1, u + v <= 1$.

Integrating the equation of motion

Because in PyRID the mobility of each molecule is described by the mobility tensor in the local frame instead of a scalar mobility coefficient, integrating the equation of motion for surface molecules becomes straight forward. Here, we can simply skip the z components in the integration scheme (Eqs.2.12, 2.17). Otherwise, we would need to calculate the tangent external and Brownian force vectors.

2.6 Boundary Conditions

PyRID supports three kinds of boundary conditions:

1. Periodic,
2. repulsive and
3. fixed concentration boundary conditions.

Repulsive boundary conditions are handled either by a repulsive interaction potential or via ray tracing, depending on the molecule type (see section 2.5.2). For periodic boundary conditions, the minimal image convention is applied (Fig. 3.3 C). Thereby, each particle only interacts with the closest image of the other particles in the system. Note, however, that the box size must not become too small, otherwise particles start to interact with themselves. As periodic and repulsive boundary conditions are very common, I will, in the following only introduce the fixed concentration boundary conditions in more detail, which is a feature unique to PyRID.

2.6.1 Fixed concentration boundary conditions

Fixed concentration boundary conditions couple the simulation box to a particle bath. Thereby, we can simulate, e.g., a sub-region within a larger system without the need to simulate the dynamics of the molecules outside simulation box directly. Instead, molecules that are outside the simulation box are treated as 'virtual' molecules that only become part of the simulation if they cross the simulation box border. In PyRID it is possible to have mesh compartments intersect with the simulation box boundary. Molecules then enter and exit the simulation across the intersection surface or the intersection line in the case of surface molecules.

Each iteration of a simulation, the expected number of hits between a molecule type and simulation box borders are calculated. The number of hits depends on the outside concentration of the molecule, the diffusion coefficient and the border surface area. The average number of volume molecules that hit a boundary of area A from one side within a time step Δt can be calculated from the molecule concentration C and the average distance a diffusing molecule travels normal to a plane l_n within Δt [Kerr et al., 2008]:

$$N_{hits} = \frac{A l_n}{2C}, \tag{2.38}$$

where

$$l_n = \sqrt{\frac{4D\Delta t}{\pi}}. \tag{2.39}$$

Here $D = Tr(\boldsymbol{D}^{tt,b})/3$ is the scalar translational diffusion coefficient. For surface molecules $D = Tr(\boldsymbol{D}^{tt,b}_{xy})/2$ and

$$N_{hits} = \frac{L l_n}{2C}, \tag{2.40}$$

where L is the length of the boundary edge. The boundary crossing of molecules can be described as a Poisson process. As such, the number of molecules that cross the boundary each time step is drawn from a Poisson distribution with a rate N_{hits}.

The normalized distance that a crossing molecule ends up away from the plane/boundary follows distribution [Kerr et al., 2008]:

$$P(d\tilde{x}) = 1 - e^{-d\tilde{x}^2} + \sqrt{\pi} * dx * \mathrm{erfc}(d\tilde{x}) \tag{2.41}$$

The distance vector normal to the plane after the crossing can then be calculated from the diffusion length constant λ and the plane's normal vector $\hat{\boldsymbol{n}}$ by $d\boldsymbol{x} = \lambda\, d\tilde{x}\, \hat{\boldsymbol{n}} = \sqrt{4Dt}\, d\tilde{x}\, \hat{\boldsymbol{n}}$.

In the case that a molecule enters the simulation box close to another boundary, e.g. of a mesh compartment, we may also want to account for the distance traveled parallel to the plane in order to correctly resolve collision with the mesh. However, currently PyRID does not account for this. For small integration time steps and meshes that are further than $\sqrt{4Dt}$ away from the simulation box border, the error introduced should, however, be negligible.

Now that the number of molecules and their distance away from the plane are determined, the molecules are distributed in the simulation box. Since the diffusion along each dimension is independent we can simply pick a random point uniformly distributed on the respective plane. For triangulated mesh surfaces, triangles are picked randomly, weighted by their area. Sampling a uniformly distributed random point in a triangle is done by [Osada et al., 2002]

$$P(\boldsymbol{r}) = (1 - \sqrt{\mu_1}) * \boldsymbol{p}_0 + (\sqrt{\mu_1} * (1 - \mu_2)) * \boldsymbol{p}_1 + (\mu_2 * \sqrt{\mu_1}) * \boldsymbol{p}_2, \tag{2.42}$$

where μ_1, μ_2 are random numbers between 0 and 1. $\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2$ are the three vertices of the triangle.

Note that, in general, any interactions between the virtual molecules are not accounted for. Therefore, fixed concentration boundary conditions only result in the same inside and outside concentrations if no molecular interactions are simulated.

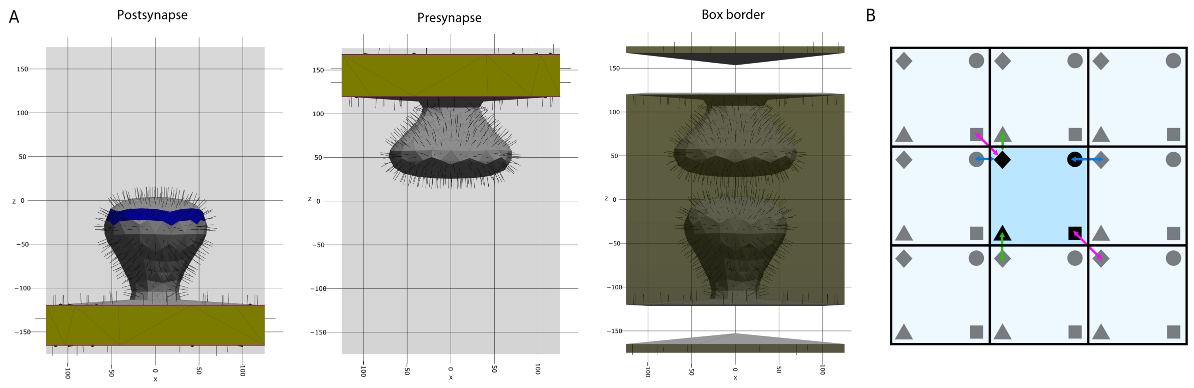


Fig 2.2. Boundaries. A (left, middle) In PyRID, the user can define different face groups. Face groups can be used, e.g., to distribute molecules on specific regions of the mesh surface (blue). When a compartment intersects with the simulation box, the intersecting triangles are assigned to a transparent class (yellow), as are the corresponding edges that intersect with the boundary (purple lines). If boundary conditions are set to "fixed concentration" transparent triangles and edges act as absorbing boundaries but in addition release new molecules into the simulation volume. (Right) The same is the case for those parts of the simulation box border that is not intersecting with one of the compartments. **B** For periodic boundary conditions, PyRID follows the minimal image convention, i.e. a particle (black marker) only interacts (colored arrows) with the closest image (grey marker) of the other particles in the system.

2.7 Reactions

In this section, methods to simulate reactions between proteins and other molecules are introduced. Reactions include, for example, post-translational modifications such as phosphorylation or ubiquitination or binding of ligands and ATP. This list could be continued. In PyRID, reactions are described on several different levels. As a result of rigid bead molecules consisting of one or several subunits, reactions can be defined either on the molecule level or on the bead/particle level. In addition, reactions are categorized into bi-molecular (second order) and uni-molecular (first order) and zero order reactions. Each uni- and bimolecular reaction can consist of several different reaction paths, each belonging to a different reaction type (for an overview see Fig. 2.3). Uni-molecular reactions are divided into the following categories:

1. decay reactions,
2. fission reactions,
3. conversion reactions.

Decay reactions account for the degradation of proteins whereas fission reactions can be used to describe ligand unbinding but also, e.g., the disassembly of protein complexes or even the flux of ions in response to ion channel opening. Conversion reactions on the other hand may be used to describe different folded protein states that change the protein properties, post-translational modifications but also binding and unbinding reactions in the case where we do not need to model the ligands explicitly (which is the case, e.g. if we can assume an infinite ligand pool). Bi-molecular reactions are divided into

1. fusion reactions,
2. enzymatic reactions.
3. binding reactions

Fusion reactions can, e.g., describe protein complex formation or ligand binding.

As mentioned above, each uni- and bi-molecular reaction can consist of one or several reaction paths. This is motivated by the minimal coarse-graining approach we take. Two proteins, e.g., can have different sites by which they interact. However, these are not necessarily represented in the rigid bead model. Similarly, a protein may convert to one of a larger set of possible states. And again, the list could be continued. In the following sections I will describe the methods by which reactions are executed in PyRID in more detail.

2.7.1 Unimolecular reactions

Unimolecular reactions include fission, conversion, and decay reactions. These can be efficiently simulated using a variant of the Gillespie Stochastic Simulation Algorithm (SSA) [Erban et al., 2007, Gillespie, 1977]. Thereby, the time point of the next reaction is

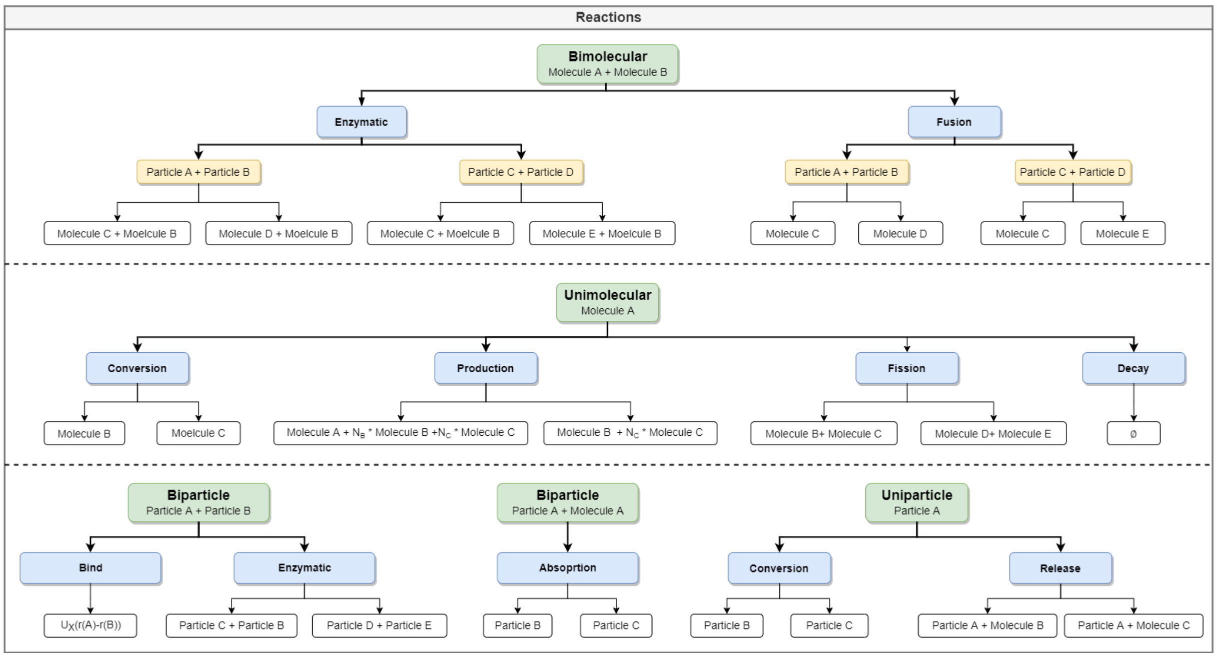


Fig 2.3. Reactions graph. PyRID supports various kinds of bimolecular and unimolecular reactions. The trees give an overview about the possible reactions and reaction paths. Bimolecular reactions are always associated with one or several particle pairs. A reaction can always have one or several possible reaction products by having different reaction paths.

sampled from the probability distribution of expected molecule lifetimes, assuming that in between two time points no interfering event occurs. An interfering event could, e.g., be a bi-molecular reaction. The naive way of simulating uni-molecular reactions would be to check each time step whether the reaction will occur depending on its reaction rate. The Gillespie SSA has the benefit of being exact (partially true since the simulation evolves in finite, discrete time steps) and far more efficient, because we only need to evaluate a reaction once and not each time step. For a single molecule having n possible reaction paths each with a reaction rate k_i, let $k_t = \sum_i^n k_i$ be the total reaction rate. Let $\rho(\tau)d\tau$ be the probability that the next reaction occurs within $[t+\tau, t+\tau+d\tau)$, which can be split into $g(\tau)$, the probability that no reaction occurs within $[t, t+\tau)$ and probability that a reaction occurs within the time interval $d\tau$, which is given by $k_t d\tau$. Thereby,

$$\rho(\tau)d\tau = g(\tau)k_t d\tau, \tag{2.43}$$

where $g(\tau) = e^{-k_t\tau}$ [Erban et al., 2007]. From the above equation we find $P(\tau) = 1 - e^{-k_t\tau}$ by integration. To sample from this distribution, we can use the inverse distribution function.

$$\tau = P^{-1}(U) \tag{2.44}$$

where U is uniformly distributed in $(0,1)$. From $U = P(\tau) = 1 - e^{-k_t \tau}$, we find $P^{-1}(U) = \frac{-log(1-U)}{k_t}$. Since U is uniformly distributed in $(0,1)$, so is $1-U$. Thereby, we can draw the time point of the next reaction from:

$$\tau = \frac{1}{k_t} \ln \left[\frac{1}{U} \right], \tag{2.45}$$

With the above method, we accurately sample from the distribution of expected molecule lifetimes $\rho(\tau) = k_t e^{-k_t \tau}$.

At the time point of the reaction, we can sample from the set of reaction paths by a weighted random choice algorithm. Therefore, we compare a second random number, uniformly distributed in $(0, k_t)$, with the cumulative set of reaction rates $(k_1, k_1+k_2, ..., k_t)$. The comparison can be made efficiently via a bisection algorithm.

Particle and molecule reactions

Because in PyRID, molecules are represented by rigid bead models, uni-molecular reactions can occur either on the molecule level or on the particle level. As such, if a conversion or decay reaction is defined on a molecule, executing the reaction will exchange the complete rigid bead molecule by a product molecule, or, in the case of a decay reaction, will remove the complete molecule from the simulation. On the other hand, if the reactions are defined on a particle/bead type, only the particle will be affected. Whereas decay and conversion reactions are handled very similar for molecules and particles, fission reactions are handled slightly different. Therefore, PyRID offers three types of fission reactions:

1. fission reactions,
2. production reactions,
3. release reactions.

Standard fission reactions can only be defined on the molecule level and are executed similar to ReaDDy [Hoffmann et al., 2019]. Here, the number of product molecules is limited to two. In the case where educt and products are volume molecules, the product molecules are placed within a sphere of radius $R_{fission}$. Therefore, an orientation vector $\boldsymbol{d}$ uniformly distributed in the rotation space with a length $<= R_{fission}$ is sampled. The two products are then placed according to

$$\begin{aligned} \boldsymbol{r}_1 &= \boldsymbol{r}_0 + w_1 \boldsymbol{d}, \\ \boldsymbol{r}_2 &= \boldsymbol{r}_0 - w_2 \boldsymbol{d}, \end{aligned} \tag{2.46}$$

where $\boldsymbol{r}_0$ is the center of the educt molecule. By default $w_1 = w_2 = 0.5$. However, for different sized educts one may choose w_1 and w_2 proportional to the molecules diffusion coefficient or the diffusion length constant. If the educt and product molecules are surface molecules, the procedure is equivalent except that the direction vector is sampled from a disc on the mesh surface instead of from a sphere. If the educt is a surface molecule but the product a volume molecule, in addition to the sphere radius, the direction needs to be defined, .i.e whether the product is placed inside or outside the compartment. In both cases, the direction vector is not sampled from the full rotation space but only within the half-sphere cut by the triangle plane. Also, whenever a mesh compartment is present in the simulation, a ray tracing algorithm is used to resolve any collisions of the products' direction vectors with the mesh.

production reactions: In addition to the standard fission reaction, PyRID supports reactions with more than two products, which are here called production reactions, because an educt molecule "produces" a number of product molecules. This type of reaction can, e.g., be used to simulate the influx of ions into a compartment via an ion channel. The procedure by which the reaction is executed is very similar to the fission reaction. However, here, the educt molecule is preserved but may change its type. Also, for each product molecule, a separate direction vector within a sphere of radius R_{prod} is sampled. Collisions with the mesh are handled as before, however, collisions between the product molecules are not resolved.

release reaction: PyRID also allows for a fission type reaction to be defined on particles, which is called a release reaction. Release reactions are limited to one particle and one molecule product. When a release reaction is executed, the particle is converted to the product particle type while releasing a product molecule either into the simulation volume or the surface of a mesh compartment. The latter is only possible if the rigid bead molecule the educt particle belongs to is also a surface molecule. Release reactions can, e.g., be used to simulate the release of a ligand from a specific binding site of a rigid bead molecule. The release reaction is introduced as the inverse of the particle absorption reaction (see next section on bi-molecular reactions).

2.7.2 Bi-molecular reactions

Bi-molecular reactions cannot be evaluated the same way as uni-molecular reactions since we cannot sample from the corresponding probability space as we have done for the uni-molecular reactions, because we do not know when two molecules meet in advance. Here, we use a reaction scheme introduced by Doi [1976], which is also used in the Brownian dynamics simulation tool ReaDDy [Schöneberg and Noé, 2013, Hoffmann et al., 2019]. In this scheme, two molecules can only react if the inter-molecular distance $|\boldsymbol{r}_{ij}|$ is below a

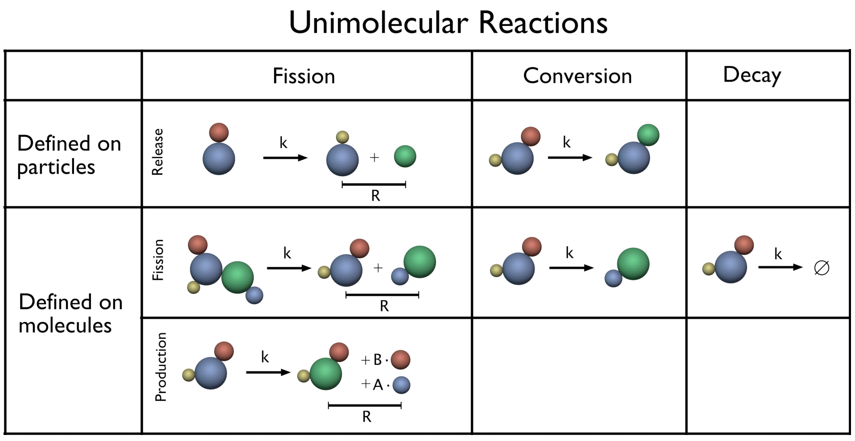


Fig 2.4. Unimolecular reactions. Unimolecular reactions can be either defined on a particle or on a molecule type. Their exist in total three different unimolecular reaction type categories: fission, conversion and decay. For details see text.

reaction radius R_{react}. The probability of having at least one reaction is then given by

$$p = 1 - \exp\Big(-\sum_i^n k_i \Delta t\Big), \tag{2.47}$$

where n is the number of reaction paths. Here, we assume that the time step Δt is so small that the molecule can only undergo one reaction. As such, the accuracy of the simulation strongly depends on the proportion between the reaction rate and the time step Δt. If $k_t \cdot \Delta t > 0.1$, PyRID will print out a warning. As for uni-molecular reactions, each bi-molecular reaction can contain several reaction paths, each of which can be of a different bi-molecular reaction type. PyRID supports the following bi-molecular reactions:

1. fusion reactions,
    - molecule fusion,
    - particle-molecule absorption,
2. enzymatic reactions (defined on molecules or particles),
3. binding reactions

Molecule fusion reactions are defined on molecule pairs. The product molecule is always placed relative to the position of the first educt. Thereby, in PyRID, the order in which the educts of a reaction are set is important. For example, for a fusion reaction $\mathrm{A} + \mathrm{B} \longrightarrow \mathrm{C}$ the product is placed at $\boldsymbol{R}_A + \omega \Delta \boldsymbol{R}$, where $\boldsymbol{R}_A$ is the origin of molecule A, $\Delta \boldsymbol{R}$ is the distance vector between A and B and ω is a weight factor. For a $\mathrm{B} + \mathrm{A} \longrightarrow \mathrm{C}$, the product is placed at $\boldsymbol{R}_B + \omega \Delta \boldsymbol{R}$. By default, $\omega = 0.5$ such that the product is placed in the middle between the educt and the order does not matter. However, for $\omega \neq 0.5$, the order in which the educts have been set determines where the product is placed.

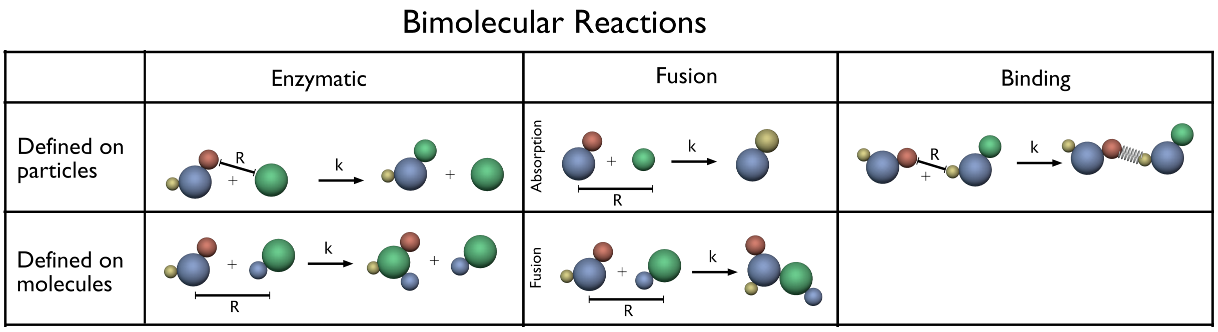


Fig 2.5. Bimolecular reactions. Bimolecular reactions can be either defined on a particle or on a molecule type. Their exist in total three different bimolecular reaction type categories: enzymatic, fusion and binding. For details see text.

In addition to the fusion reaction, PyRID offers the *particle-molecule absorption reaction*, which is also a reaction of the fusion type. However, here a molecule is absorbed by the bead/particle of another molecule. The molecule is thereby removed from the simulation and the absorbing particle is converted to a different type.

Binding reactions are defined between two particle/bead types and handled similar to fusion and enzymatic reactions except that, if the reaction was successful, an energy potential between the two educt particles is introduced such that these interact with each other. Upon binding, the beads can change their respective type. Also, a bead can only be bound to one partner particle at a time. Bonds can be either persistent or breakable. In the latter case, the bond is removed as soon as the inter-particle distance crosses an unbinding threshold. Similarly, unbinding reactions can be introduced by means of a conversion reaction as bonds are removed if a particle or the corresponding rigid bead molecule are converted to a different type.

Reactions between surface molecules

As for volume molecules, molecules that reside on the surface/in the membrane of a compartment react with each other if the inter-particle distance is below the reaction radius. However, PyRID only computes the euclidean distance between particles. Therefore, however, surface reactions are only accurate if the local surface curvature is large compared to the reaction radius. Accurate calculation of the geodesic distance on mesh surfaces is computationally very expensive. Algorithms that allow for relatively fast approximations of geodesic distances and shortest paths such as the Dijkstra's algorithm often only provide good approximations for point far away from the source. Therefore, the benefit of implementing such algorithms is questionable as reaction radii are on the order of the molecule size and thereby usually small compared to the mesh size. However, much progress has been made in this field [Polthier and Schmies, 2006, Crane et al., 2017, Trettner et al., 2021].

2.8 Potentials

PyRID supports any pairwise, short ranged interaction potential and external potentials. The force is given by

$$\boldsymbol{F}_i = \nabla_i \left(\sum U_{ext}(\boldsymbol{r}_i) + \sum_{i \neq j} U_{pair}(\boldsymbol{r}_i, \boldsymbol{r}_j) \right) \tag{2.48}$$

PyRID comes with a selection of pairwise interaction potentials. PyRID does not support methods such as Ewald summation and pair interaction potentials need to be short ranged, i.e., they need to have a cutoff distance.

In the following I list the functions currently implemented in PyRID. However, any short ranged, pair-wise interaction potential can be easily added using python.

Weak piecewise harmonic potential

The very same interaction potential is also used in ReaDDy [Hoffmann et al., 2019].

$$U_{ha}(r) = \begin{cases} \frac{1}{2}k(r-(d_1+d_2))^2 - h, & \text{if } r < (d_1+d_2), \\ \frac{h}{2}(\frac{r_c-(d_1+d_2)}{2})^{-2}(r-(d_1+d_2))^2 - h, & \text{if } d \leq r < d + \frac{r_c-(d_1+d_2)}{2}, \\ -\frac{h}{2}(\frac{r_c-(d_1+d_2)}{2})^{-2}(r-r_c)^2, & \text{if } d + \frac{r_c-(d_1+d_2)}{2} \leq r < r_c, \\ 0, & \text{otherwise} \end{cases} \tag{2.49}$$

Harmonic repulsion potential

The very same interaction potential is also used in ReaDDy [Hoffmann et al., 2019].

$$U(r) = \begin{cases} \frac{\kappa}{2}(r-\sigma)^2, \text{ if } r \leq \sigma \\ 0, \text{ otherwise }, \end{cases} \tag{2.50}$$

Continuous Square-Well (CSW) potential

The Continuous Square-Well (CSW) potential has been introduced in [Espinosa et al., 2014].

$$U_{CSW}(r) = -\frac{\epsilon_{CSW}}{2}\Big[1 - \tanh\Big(\frac{r-r_w}{\alpha}\Big)\Big]. \tag{2.51}$$

Pseudo Hard Sphere (PHS) potential

The Pseudo Hard Sphere (PHS) potential has been introduced in [Jover et al., 2012].

$$U_{HS} = \begin{cases} \lambda_r (\frac{\lambda_r}{\lambda_a})^{\lambda_a} \epsilon_R [(\frac{\sigma}{r})^{\lambda_r} - (\frac{\sigma}{r})^{\lambda_a}] + \epsilon_R, \text{ if } r < (\frac{\lambda_r}{\lambda_a})\sigma \\ 0, \text{ if } r < (\frac{\lambda_r}{\lambda_a})\sigma, \end{cases} \tag{2.52}$$

2.9 Observables

PyRID can sample several different system properties:

1. Energy
2. Pressure
3. Virial
4. Virial tensor
5. Volume
6. Molecule number
7. Bonds
8. Reactions
9. Position
10. Orientation
11. Force
12. Torque
13. Radial distribution function

Each observable (except the volume) is sampled per molecule type or molecule/particle pair in the case of bimolecular reactions and bonds. In addition, values can be sampled in a step-wise or binned fashion. Binning is especially useful when sampling reactions as one is usually interested in the total number of reactions that occurred with a time interval and not in the number of reactions that occurred at a specific point in time. In the following I briefly describe how the radial distribution function and the pressure are calculated in PyRID.

2.9.1 Radial distribution function

The radial distribution function is given by

$$g(\boldsymbol{r}) = \frac{V_{box}}{N_i N_j} \left\langle \sum_{i \neq j} \delta(\boldsymbol{r} - (\boldsymbol{r}_i - \boldsymbol{r}_j)) \right\rangle = \frac{V_{box}}{N_i N_j} \frac{1}{V(\boldsymbol{r})} \sum_{i \neq j} \delta(\boldsymbol{r} - (\boldsymbol{r}_i - \boldsymbol{r}_j)) \tag{2.53}$$

where $V(\boldsymbol{r}) = \frac{4}{3}\pi(r - \Delta r)^3$ with Δr being the sampling bin size. V_{box} is the volume of the simulation box and N_i and N_j are the total number of molecule types i and j respectively.

2.9.2 Pressure

The pressure can be calculated from the virial or the the viral tensor. However, when calculating the pressure for a system of rigid bodies/ rigid bead molecules, we need to be careful how to calculate the virial tensor. Taking the inter-particle distances will result in the wrong pressure. Instead, one needs to calculate the molecular virial [Glaser et al., 2020], by taking the pairwise distance between the center of diffusion of the respective molecule pairs:

$$P_{mol} = P_{mol}^{kin} + \frac{1}{6V} \sum_{i=1}^{N} \sum_{j\neq}^{N} \langle \boldsymbol{F}_{ij} \cdot (\boldsymbol{R}_i - \boldsymbol{R}_j) \rangle, \tag{2.54}$$

where V is the total volume of the simulation box, $\boldsymbol{F}_{ij}$ is the force on particle i exerted by particle j and $\boldsymbol{R}_i, \boldsymbol{R}_j$ are the center of diffusion of the rigid body molecules, not the center of mass of particles i and j! In Brownian dynamics simulations, $P_{mol}^{kin} = N_{mol} k_B T$, where N_{mol} is the number of molecules. Also, the origin of molecules is represented by the center of diffusion around which the molecule rotates about, which is not the center of mass [Harvey and de la Torre, 1980]. The net frictional force and torque act through the center of diffusion. This is because when doing Brownian dynamics (and equaly for Langevin dynamics), we do account for the surrounding fluid. Different parts of the molecule will therefore interact with each other via hydrodynamic interactions/coupling. As a result, the center of the molecule (around which the molecule rotates in response to external forces) is not the same as the center of mass, which assumes no such interactions (the molecule sites in empty space). However, for symmetric molecules, the center of mass and the center of diffusion are the same.

2.10 Berendsen barostat

It is sometimes desirable to be able to do simulations in the NPT ensemble, e.g., in preparation steps to release the system from stresses. This can become necessary, e.g. when computing inter-facial properties of fluids or computing phase diagrams via direct coexistence methods [Espinosa et al., 2019, 2020, Muller et al., 2020]. The Berendsen barostat [Berendsen et al., 1984] is simple to implement and results in the correct target density of the system, however, it does not sample from the correct statistical ensemble distribution as pressure fluctuations are usually too small. By scaling the inter-particle distances, the Berendsen barostat changes the virial and thereby the system pressure. Per time step, the molecule coordinates and simulation box length are scaled by a factor μ that is given by:

$$\mu = (1 - \Delta t/\tau_P(P_0 - P))^{1/3}, \tag{2.55}$$

where τ_P is the coupling time constant, P_0 the target pressure. The above equation applies to an isotropic system. For a anisotropic system the equation can be generalized by substituting P with the pressure tensor. In the case of a rectangular simulation box, all tensor remain diagonal and application of the anisotropic barostat is trivial.

2.11 Distribution of molecules

2.11.1 Volume molecules

The distribution of molecules in the simulation volume becomes a special problem when we have mesh compartments and account for the excluded volume of the molecules. A standard approach from molecular dynamics first loosely distributes the molecules in the simulation box and then shrinks the simulation volume until a target density is reached. This approach could be transferred to a system with mesh compartments. However, here, we might also care about the compartment size. As such, we would need to choose a larger than target compartment size and shrink it until we reach the target size. If the density is too large, we may randomly delete molecules until the target density is also reached. A second approach would be to utilize the Metropolis Monte Carlo method [Allen and Tildesley, 2017] to distribute the molecules. However, this approach is more time-consuming. A third approach, which is the one we use in PyRID, uses a so-called Poisson-Disc sampling algorithm [Bridson, 2007]. This approach has the benefit of being computationally efficient and relatively simple to implement. It, however, has the disadvantage of not reaching densities above 30% and is only well suited for approximately spherical molecules. To distribute highly aspherical molecules, currently, the only useful method that works well with PyRID is to distribute the molecules using Monte-Carlo sampling and then resolve overlaps via a soft repulsive interaction potential. If no mesh compartments are used, one may also use the Berendsen barostat at high pressure to drive the system to a high density state. The Poison-disc sampling algorithm consists of 3 steps. 1) A grid is initialized, where the cell size is set to $r/\sqrt{3}$. 2) A sample point is created and inserted into a list of active elements. 3) While the active list is not empty, new random points around the annulus (r-2r) of the active sample points are created. If no other sample points exist within the radius r, the new sample point is accepted and inserted into the grid and the active list. If, after k trials, no new sample point is found, the active sample point is removed from the active list. For PyRID, this algorithm has been extended to account for polydisperse particle distributions.

2.11.2 Surface molecules

The distribution of molecules on the surface of a mesh compartment is a little more involved. Here, we utilize an algorithm introduced by Corsini et al. [2012]:

1. Generate a sample pool S by uniformly distributing points on the mesh surface.
2. Divide space into cells and count the number of samples in each cell.
3. Randomly select a cell weighted by the number of active samples in each cell (active sample: sample that is not yet occupied or deleted).
4. Randomly select a sample from the selected cell.
5. Randomly choose a particle type of radius R_i (weighted by the relative number of each type we want to distribute).
6. Check whether the distance of the selected sample to the neighboring samples that are already occupied is larger or equal to Ri+Rj.
7. If True, accept the sample and add the molecule type and position to an occupied sample list. Next, delete all other samples within radius Ri, as these won't ever become occupied anyway.
8. Update the number count of samples for the current cell.
9. While the desired number of molecules is not reached, return to 3. However, set a maximum number of trials.
10. If there are no active samples left before we reach the desired molecule number and the maximum number of trials, generate a new sample pool.

PyRID also allows the user to assign individual mesh triangles to a group and thereby define surface regions on which to distribute molecules. Example results for the distribution of volume and surface molecules using the above described methods are shown in Fig. 2.6.

2.12 Fast algorithms for Brownian dynamics of reacting and interacting particles

PyRID is written entirely in the programming language python. To make the simulations run efficiently, PyRID heavily relies on jit compilation using Numba. In addition,

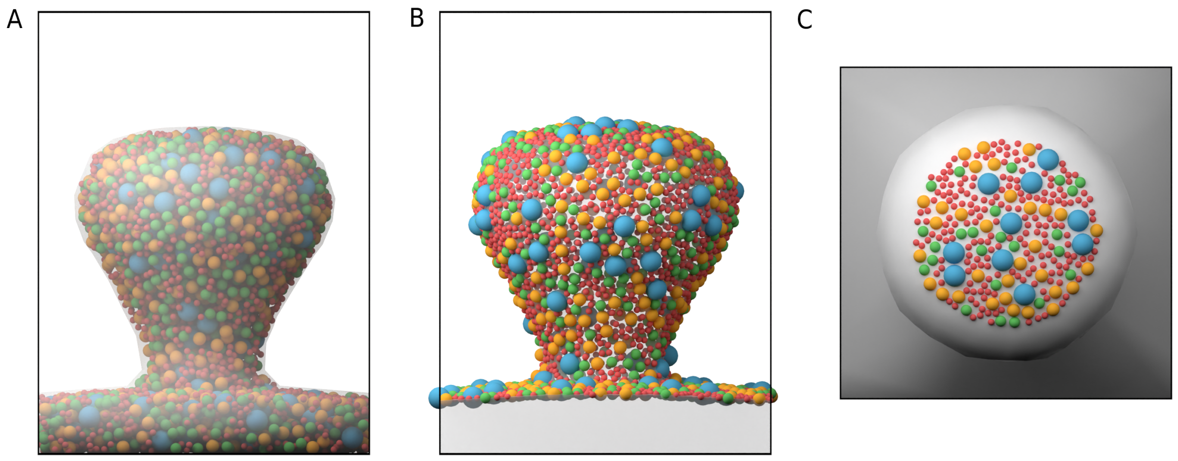


Fig 2.6. Poisson Disc Sampling of polydisperse spheres. (A) Example distribution for three different sized particle types confined to the volume of a mesh compartment . **(B)** Poisson Disc sampling for surface molecules.**(C)** Poisson Disc sampling for surface molecules but restricted to a surface region that is defined by a triangle face group.

PyRID uses a data-oriented design and specific dynamic array data structures to keep track of molecules and their reactions efficiently. For this important part of the PyRID implementation to not remain elusive, I will introduce the main data structures that make PyRID run efficiently in this section. A very nice introduction/overview to the kind of data structures used here has been written by Niklas Gray[1].

2.12.1 Dynamic arrays in PyRID

In PyRID, molecules and particles constantly enter or leave the system due to reactions and other events. Therefore, we need a data structure that can efficiently handle this constant change in the number of objects we need to keep track of in our simulation. The same holds true for the molecular reactions occurring at each time step. These need to be listed and evaluated efficiently. Fortunately, variants of dynamic array data structures are tailored for such tasks, of which we use two kinds, the tightly packed dynamic array and the dynamic array with holes.

[1]https://web.archive.org/web/20220517145529/https://ourmachinery.com/post/data-structures-part-1-bulk-data/
https://web.archive.org/web/20220314011542/https://ourmachinery.com/post/data-structures-part-2-indices/
https://web.archive.org/web/20220517134710/https://ourmachinery.com/post/data-structures-part-3-arrays-of-arrays/
https://www.gamedeveloper.com/programming/data-structures-part-1-bulk-data

The tightly packed dynamic array (dense array)

A tightly packed dynamic array is a dynamic array (similar to lists in python or vectors in C++) where elements can be quickly deleted via a pop and swap mechanism (Fig. 2.7). The problem with standard numpy arrays but also lists and C++ vectors is that deletion of elements is very expensive. For example, if we want to delete an element at index m of a numpy array of size n, numpy would create a new array that is one element smaller and copies all the data from the original array to the new array. Also, if we want to increase the size of a numpy array by appending an element, again, a new array will need to be created, and all data needs to be copied. This is extremely computationally expensive. One way to create a dynamic array (and python lists work in that way) is to not increase the array size each time an element is added but increase the array size by some multiplicative factor (usually 2). This consumes more memory but saves us from creating new arrays all the time. Now we simply need to keep track of the number of elements in the array (the length of the array) and the actual capacity, which can be much larger. One straightforward method to delete elements from the array is just to take the last element of the array and copy its data to wherever we want to delete an element (swapping). Next, we pop out the last element by decreasing the array length by 1. We call this type of array a ‘tightly packed array’ because it keeps the array tightly packed. One issue with this method is that elements move around. Thereby, to find an element by its original insertion index, we need to keep track of where elements move. One can easily solve this issue by keeping a second array that saves for each index the current location in the tightly packed array.

The dynamic array with holes

To store molecules and particles, we use a dynamic array with holes (Fig. 2.7). A dynamic array with holes is an array where elements can be quickly deleted by creating ‘holes’ in the array. These holes are tracked via a free linked list. The array with holes has the benefit over the ‘tightly packed array’ that elements keep their original index because they are not shifted/swapped at any point due to deletion of other elements. This makes accessing elements by index a bit faster compared to the other approach. However, if the number of holes is large, i.e. the array is sparse, this approach is not very cache friendly. Also, iterating over the elements in the array becomes more complex because we need to skip the holes. Therefore, we add a second array, which is a tightly packed array, that saves the indices of all the occupied slots in the array with holes (alternatively, we could add another linked list that connects all occupied slots). We can then iterate over all elements in the holes array by iterating over the tightly packed array. Keep in mind, however, that the order is not preserved in the tightly packed array, since, whenever we delete an element from the holes array, we also need to delete this element from the dense array by the pop and swap mechanism. As such, this method does not work well if we need to iterate over a sorted array. In that case, one should use a free linked list approach for iteration. As with the tightly packed dynamic array, the array size is increased by a multiplicative factor of 2 as soon as the capacity limit is reached.

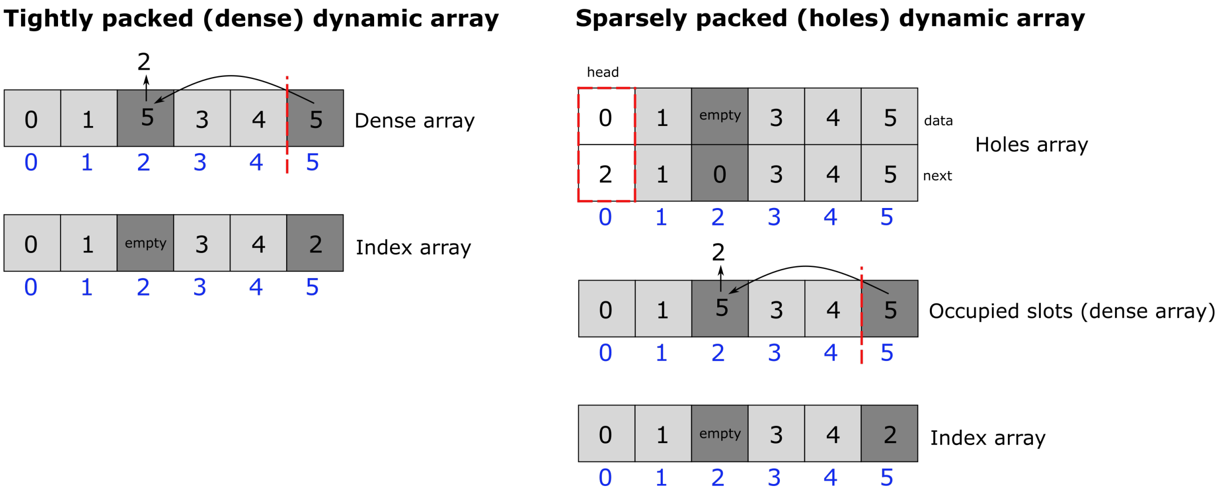

Tightly packed (dense) dynamic array
Sparsely packed (holes) dynamic array
Dense array
Index array
head
data
next
Holes array
Occupied slots (dense array)
Index array
empty

Fig 2.7. Dynamic arrays. (A) Tightly packed dynamic array. A tightly packed dynamic array uses a pop and swap mechanism to delete single elements. New elements are appended. Thereby, however, elements change their position. To find elements, a second, sparsely packed dynamic array is introduced that keeps track of the array index of each element. **(B)** In sparsely packed dynamic arrays, elements are deleted by creating holes. Holes are kept track of by a free linked list. Sparsely packed dynamic arrays have the benefit of keeping the position of each element inside the array fixed, which makes finding elements easy. However, this comes at the expense of more memory usage. Also, iteration over a sparse dynamic array is not straight forward as we must skip the holes in the array. Their exist different methods to iterate over such an array. In PyRID, a second, densely packed array is introduced that keeps the indices of all occupied slots. However, in order to be able to delete an element, we now also need third, sparse array, that keeps track of the element indices in the densely packed array.

Dynamic arrays used for reaction handling

The data structure we need to organize the reactions is a little bit more complex than a simple dense, dynamic array or one with holes, as is used to keep track of all the rigid body molecules and particles in the system. Instead a combination of different dynamic arrays and a hash table is used (Fig. 2.10). Let me motivate this: Our data structure needs to be able to do four things as efficient as possible:

1. Add reactions,
2. Delete single reactions,

3. Delete all reactions a certain particle participates in,
4. Return a random reaction from the list.

We need to be able to delete a single reaction whenever this reaction is not successful. We need to delete all reactions of a particle whenever a reaction was successful because, in this case, the particle is no longer available since it either got deleted or changed its type (except in the case where the particle participates as an enzyme). We need to be able to request a random reaction from the list because processing the reactions in the order they occur in the list would introduce a bias[2]. Points 1. and 2. could be easily established with a simple dynamic array. However, point 3 is a bit more complicated but can be solved with a doubly free linked list embedded into a dynamic array with holes. This doubly linked list connects all reactions of a particle. To find the starting point (the head) of a linked list within the array for a certain particle, we save the head in a hash table (python dictionary). A doubly linked list is necessary because we need to be able to delete a single reaction (of index k) from the linked list (point 2). As such, we need to be able to reconnect the linked list's 'chain'. Therefore, we need to know the element in the linked list that pointed to k (previous element) in addition to the element/reaction k points to (next element). Another problem that needs to be solved is that a reaction can be linked to at maximum two educts. Therefore, each next and previous pointer needs to be 4-dimensional: We need one integer for each educt to save the next (previous) reaction index and another integer 0,1 to keep track of whether in the next (previous) reaction, the particle is the first or the second educt, because this may change from reaction to reaction! Since the dynamic array, the doubly linked list is embedded, in has holes, picking a random reaction from the list becomes another issue. This can, however, easily be solved by adding another dynamic array (tightly packed), which keeps the indices of all the reactions that are left in a tightly packed format. Picking a random reaction is then as easy as drawing a uniformly distributed random integer between 0 and n, where n is the length of the dense array.

2.13 Polydispersity

A problem that needs to be addresses, especially when using minimal coarse-graining approaches with low granularity is polydispersity of particle radii. One extreme example

[2]Reactions of particles with a low index are added to the reaction list first, because particle distances are evaluated in the order particles occur in the corresponding list (which is, at least in the beginning, in ascending order, when the indices of particles have not yet been swapped around a lot). Thereby particle one would always have a higher chance of having a successful reaction when competing with other particles.

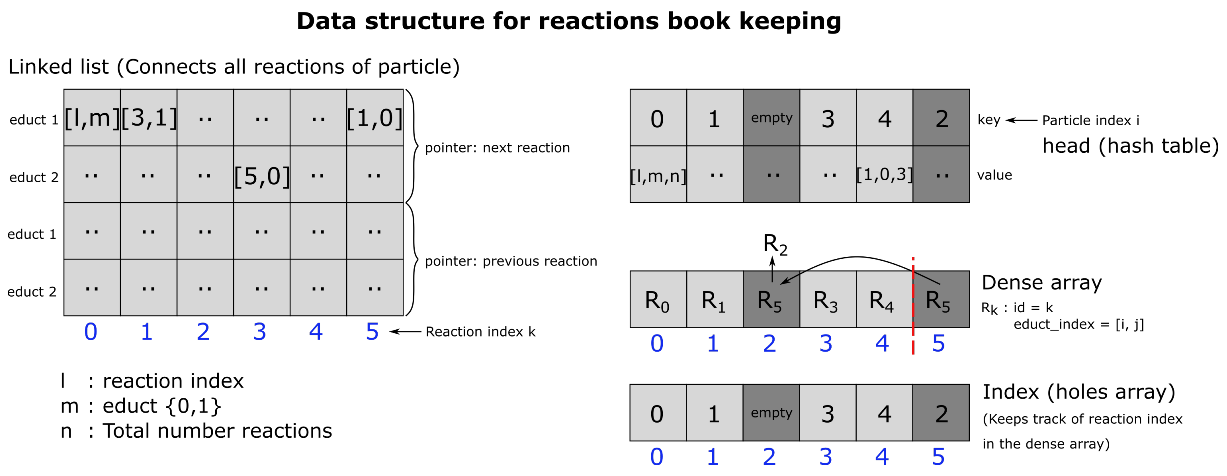


Fig 2.8. Dynamic array for reactions. The dynamic array that keeps track of all the reactions that need to be executed within a simulation time step consists a free doubly linked list, a hash table (python dictionary), one densely packed and one sparsely packed dynamic array.

would, e.g., be a simulation where proteins and synaptic vesicles are represented by single particles. In this case classical linked cell list algorithms become highly inefficient.

The computationally most expensive part in molecular dynamics simulations is usually the calculation of the pairwise interaction forces, because to calculate these, we need to determine the distance between particles. When doing this in the most naive way, i.e. for each particle i we iterate over all the other particles j in the system and calculate the distance, the computation time will increase quadratic with the number of particles in the system ($O(N^2)$). Even if we take into account Newtons third law, this will decrease the number of computations ($O(\frac{1}{2}N(N-1))$), but the computational complexity still is quadratic in N. A straight forward method to significantly improve the situation is the linked cell list approach [Allen and Tildesley, 2017] (pp.195: 5.3.2 Cell structures and linked lists) where the simulation box is divided into $n \times n \times n$ cells. The number of cells must be chosen such that the side length of the cells in each dimension $s = L/n$, where L is the simulation box length, is greater than the maximum cutoff radius for the pairwise molecular interactions (and in our case also the bimolecular reaction radii). This will decrease the computational complexity to $O(14N\rho s^3)$, where ρ is the molecule density (assuming a mostly homogeneous distribution of molecules). Thereby, the computation time increase rather linear with N instead of quadratic (N^2).

However, one problem with this method is that it does not efficiently handle polydisperse particle size distributions. This becomes a problem when doing minimal coarse graining of proteins and other structures we find in the cell such as vesicles. As mentioned above, n should be chosen such that the cell length is greater than the maximum cutoff radius. If we would like to simulate, e.g. proteins ($r \approx 2 - 5nm$) in the presence of synaptic vesicles ($r \approx 20 - 30nm$), the cells become way larger than necessary from the perspective of the small proteins.

One way to approach this problem would be, to choose a small cell size (e.g. based on the smallest cutoff radius) and just iterate not just over the nearest neighbour cells but over as many cells such that the cutoff radius of the larger proteins is covered. This approach has a big disadvantages: Whereas for the smaller particles we actually reduce the number of unnecessary distance calculations, we do not for the larger particles as we iterate over all particles, also those, which are far beyond the actual interaction radius. We can, however, still take advantage of Newton's 3rd law. For this, we only do the distance calculation if the radius of particle i is larger than the radius of particle j. If the radii are equal, we only do the calculation if index i is smaller than index j.

A much better approach has been introduced by Ogarko and Luding [2012] that makes use of a so called hierarchical grid. This approach is the one I use in PyRID. In the hierarchical grid approach, each particle is assigned to a different cell grid depending on its cutoff radius, i.e. the grid consists of different levels or hierarchies, each having a different cell size. This has the downside of taking up more memory, however, it drastically reduces the number of distance calculations we need to do for the larger particles and also takes advantage of Newtons third law, enabling polydisperse system simulations with almost no performance loss. The algorithm for the distance checks works as follows [Ogarko and Luding, 2012]:

1. Iterate over all particles
2. Assign each particle to a cell on their respective level in the hierarchical grid
3. Iterate over all particles once more
4. Do a distance check for the nearest neighbour cells on the level the current particle sites in. This is done using the classical linked cell list algorithm.
5. Afterwards, do a cross-level search. For this, distance checks are only done on lower hierarchy levels, i.e. on levels with smaller particle sizes than the current one. This way, we do not need to double check the same particle pair (Newtons 3rd law). However, in this step, we will have to iterate over potentially many empty cells.

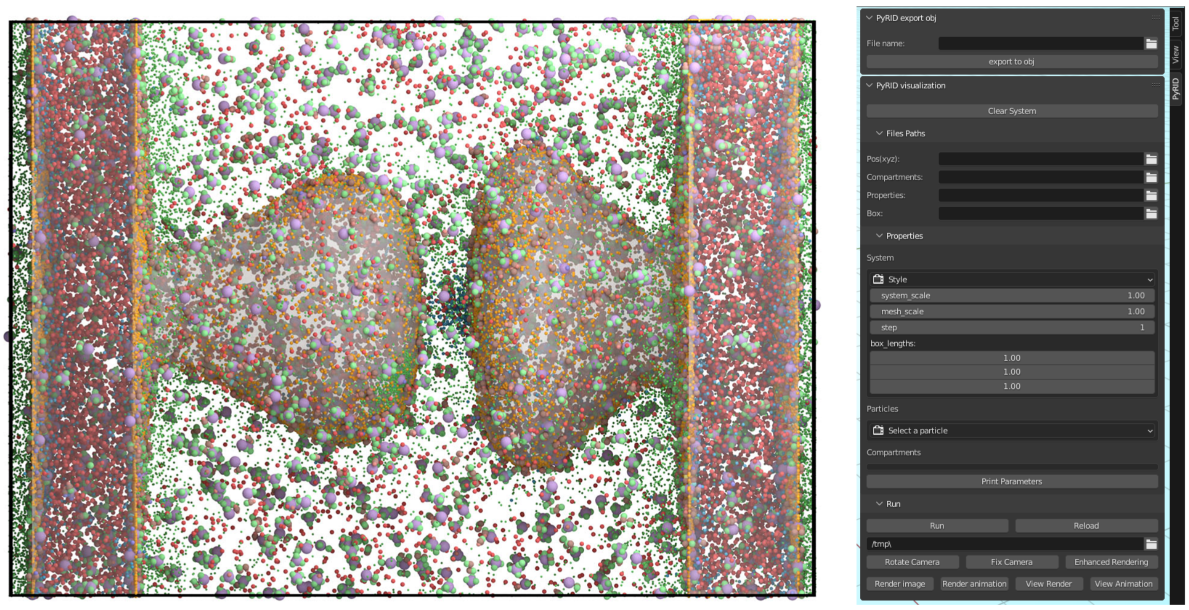
2.14 Simulation loop

At the beginning of each iteration, PyRID updates the position of all particles. Next, PyRID determines the particle pair distances and based on that calculates the forces and adds reactions to the reactions list. Then, the reactions are performed. Thereby, new particles can enter the system, either by a fusion or a fission reactions. In principle, the inter-particle distances as well as the forces would need to be updated. However, to save computation time this step is skipped. The only exceptions are binding reactions, for which the force is calculated at the time of reaction execution. Thereby, in the beginning of the next iteration, the reaction products diffuse freely as they did not experience any forces yet and the trajectory of particles that have been in the system before also is not altered due to the presence of the new molecules. Also, for molecules that where converted to a different type the forces that are taken into account for the upcoming position update are those that the molecule experienced before the type change. This is of course an approximation but saves us from updating the particle distances twice. For small integration time steps the error introduced by this scheme should be small. Also, the product molecule placement itself does already not resolve any collisions and other interactions with nearby molecules. Therefore, a very similar error is introduced anyway since resolving collisions would be unfeasible and take up too much computation time. For fission reactions the error will be very similar, no matter whether forces are updated or not. For fusion reactions the the situation is slightly different because the educts will leave behind an empty space that will be filled with new molecules. But, also here, the error should be small for small time steps, which need to be considered anyhow as soon as interactions are introduced. A problem will always arise as soon as the system is dense and many products enter the system at ones. In this case, the integration time step should be decreased such that only a few products enter the system.

An alternative to the above approach would be to calculate all the forces and add all the reactions to the reactions list at the beginning of each iteration. Next we would update the molecule positions and only then evaluate the reactions. This approach has the benefit that for the product molecules, the forces will be evaluated directly after they have been placed. However, because the products are only placed after the molecule positions are updated, this results in a bias, especially for bimolecular reactions. The first approach has the benefit that it does not introduce any bias in the case where we do not consider interactions. Therefore, this is the approach I took with PyRID. In contrast, ReaDDy does update the neighbouring list for the molecules twice per iteration. Once to evaluate the reactions and another time to update the forces. However, in the worst case this could increase computation time by a factor of two if the update of inter-molecular distances is not optimized in some way. With the approach I took, forces and reactions are evaluated in one loop over all particle pairs in the neighbouring list and pair distances are calculated only once. One could try to optimize the update of pair distances after the reactions have been executed, since the pair distances between most particles stays the same and we only need to consider those particles that left or entered the system. However, this will be future work.

2.15 Visualization

For visualization, I have developed a blender addon for PyRID. In addition, PyRID can visualize the different reactions using graphs. Graph visualization in PyRID is build upon the pyvis library.



system_scale 1.00
mesh_scale 1.00
step 1
box_lengths:
1.00
1.00
1.00
/tmp\

Fig 2.9. Visualization of moelcule trajectories with PyRIDs Blender addon. Left: Example visualization with 50.000 particles. Right: GUI of the Blender addon.

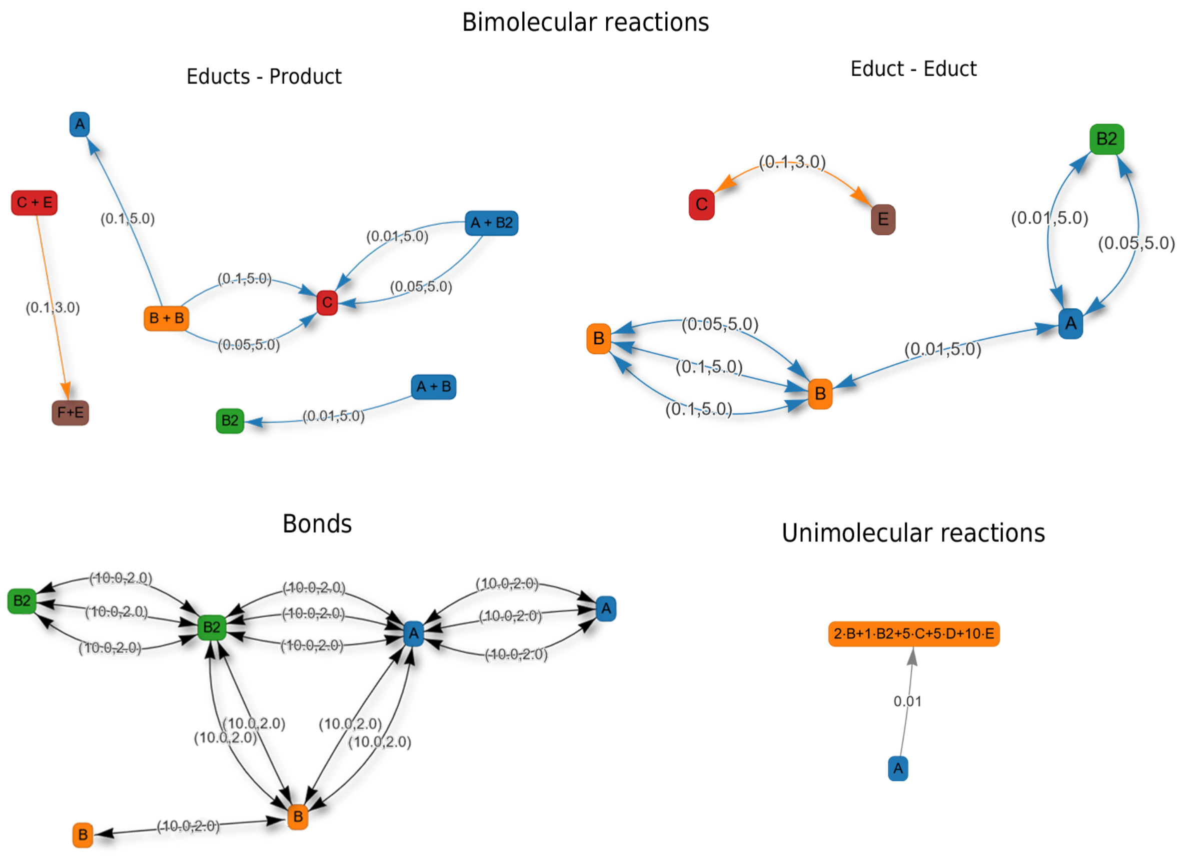


Fig 2.10. **Examples of different reaction graphs.**

3 Results and validation

3.1 Anisotropic diffusion

To validate the implementation of the algorithms for translational and rotationional diffusion introduced in 2.2 I use the same example as in [Ilie et al., 2015]. Here, the translational and rotational diffusion tensors do not represent any specific molecule:

$$D_{tt} = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.4 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \frac{nm^2}{ns}, \tag{3.1}$$

$$D_{rr} = \begin{pmatrix} 0.005 & 0 & 0 \\ 0 & 0.04 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \frac{rad^2}{ns}, \tag{3.2}$$

In order to validate the algorithm, the mean squared displacement (MSD) and rotational time correlation are compared with theory. The mean squared displacement (MSD) is given by

$$MSD = \langle |\boldsymbol{x}(t+\Delta t) - \boldsymbol{x}(t)|^2 \rangle \tag{3.3}$$

The rotational time correlation function is given by [de la Torre et al., 1999]:

$$MSD = \frac{3}{2}\langle (\hat{\boldsymbol{n}}(t+\Delta t)\hat{\boldsymbol{n}}(t))^2 \rangle - \frac{1}{2}, \tag{3.4}$$

where $\hat{\boldsymbol{n}}(t)$ is some unitary vector that describes the current orientation of the molecule at time point t. Fig. 3.1 compares the simulation results to the theoretical prediction, which, for the rotational time correlation function, is given by a multi-exponential decay function [de la Torre et al., 1999]:

$$P_{2,l}(t) = \sum_{i=1}^{5} a_{i,l} exp(-t/\tau_i), \tag{3.5}$$

where $l \in 1, 2, 3$. The relaxation times are given by

$$\begin{aligned}
\tau_1 &= (6D - 2\Delta)^{-1} \\
\tau_2 &= (3D - D_1^{rr,b})^{-1} \\
\tau_3 &= (3D - D_2^{rr,b})^{-1} \\
\tau_4 &= (3D - D_3^{rr,b})^{-1} \\
\tau_5 &= (6D - 2\Delta)^{-1}.
\end{aligned} \tag{3.6}$$

$D_1^{rr,b}, D_2^{rr,b}, D_3^{rr,b}$ are the eigenvalues of the rotational diffusion tensor $\boldsymbol{D}^{rr,b}$ in the molecule frame and D is the scalar rotational diffusion coefficient given by $D = \frac{Tr(\boldsymbol{D}^{rr,b})}{3}$. Parameter Δ is given by

$$\Delta = \sqrt{((D_1^{rr,b})^2 + (D_2^{rr,b})^2 + (D_3^{rr,b})^2 - D_1^{rr,b}D_2^{rr,b} - D_1^{rr,b}D_3^{rr,b} - D_2^{rr,b}D_3^{rr,b})} \tag{3.7}$$

The amplitudes of the individual exponential decays are given by

$$\begin{aligned}
a_{1,l} &= \frac{3}{4}(F + G) \\
a_{2,l} &= 3\hat{n}_{l,2}^2\hat{n}_{l,3}^2 \\
a_{3,l} &= 3\hat{n}_{l,1}^2\hat{n}_{l,3}^2 \\
a_{4,l} &= 3\hat{n}_{l,1}^2\hat{n}_{l,2}^2 \\
a_{5,l} &= \frac{3}{4}(F - G),
\end{aligned} \tag{3.8}$$

with $F = -\frac{1}{3} + \sum_{k=1}^{3} \hat{n}_k^4$ and $G = \frac{1}{\Delta}\Big(- D + \sum_{k=1}^{3} D_k^{rr,b}\Big[\hat{n}_k^4 + 2\hat{n}_m^2\hat{n}_n^2\Big]\Big)$, where $m, n \in \{1, 2, 3\} - \{k\}$.

If we choose the normal vectors of each axis $\hat{\boldsymbol{n}}_l$ such that these are identical with the basis vectors of the local frame, i.e. $\hat{\boldsymbol{u}}_1 = \boldsymbol{e}_x = [1, 0, 0]$, $\hat{\boldsymbol{u}}_2 = \boldsymbol{e}_y = [0, 1, 0]$, $\hat{\boldsymbol{u}}_3 = \boldsymbol{e}_z = [0, 0, 1]$, $a_2 - a_3$ vanish such that we end up with a double exponential decay (Fig. 3.1 B).

Fig. 3.1 shows that the rotation and translation propagators result in the correct mean squared distribution and rotational time correlation.

3.2 Diffusion tensor of igG3

The methods outlined in section 2.3 have, at least to my knowledge, only been implemented in the freely available tool Hydro++. The source code for Hydro++ is, however,

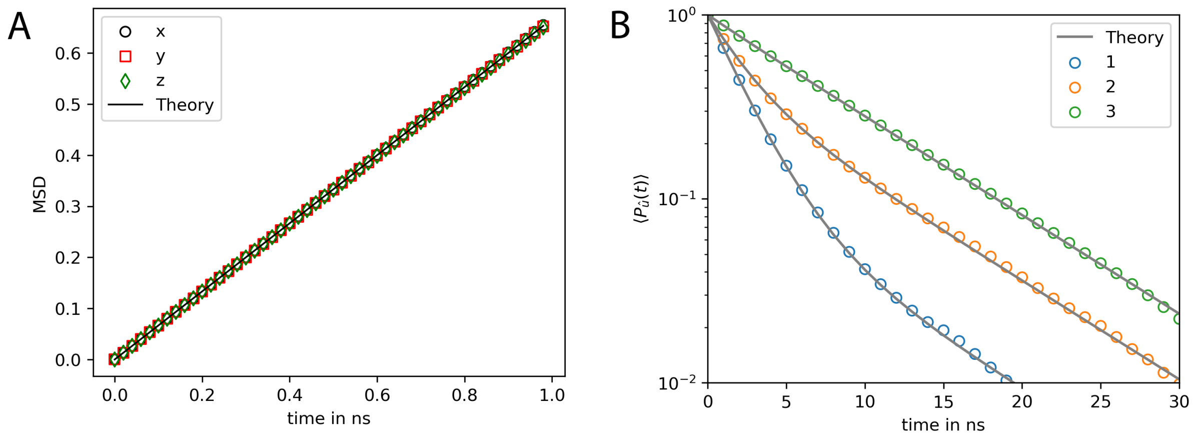


Fig 3.1. MSD and rotational relaxation times of a rigid bead molecule matches the theoretical prediction. (A) Mean squared displacement (MSD) of the rigid bead molecule computed with PyRID. The displacement in each dimension (colored markers) is in very good agreement with the theory (black line). **(B)** The rotational relaxation of the rigid bead molecule is also in close agreement with the theory (gray lines, Eqs.3.5-3.8) for each of the the rotation axes (colored markers).

not publicly available. To efficiently set up a system of rigid bead molecules, the method has now also been implemented directly into PyRID. The implementation is tested against Hydro++ using a model of the protein igG3 that comes with the documentation of Hydro++. The results are in good agreement at up to 4 digits (Table 3.1). The slight difference is probably due to numerical errors that accumulate when numerically inverting the large supermatrices.

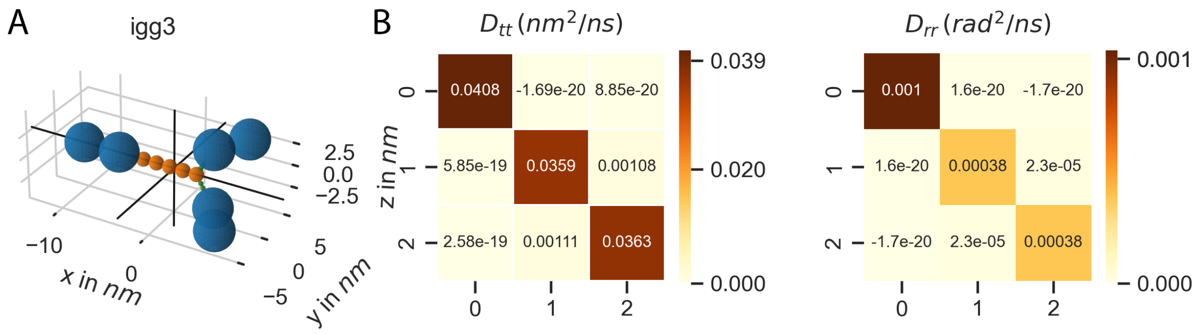


Fig 3.2. The diffusion tensor of igG3 calculated with PyRID. (A) Rigid bead molecule representation of igG3 as found in de la Torre and Ortega [2013]. The black cross marks the center of diffusion. **(B)** Translational and rotational diffusion tensor of igG3. A comparison of the result from PyRID with those of the Hydro++ suite can be found in table 3.1.

Table 3.1. Translational and rotational diffusion tensors of the IgG3 rigid bead model. Here, the result from PyRID is compared to the result gained from the Hydro++ suite. We find small deviations originating from numerical errors that build up mainly during the super-matrix inversion calculations.

	D_{tt}			D_{rr}		
Hydro++	0.0407	0.0	0.0	1.03e-03	0.0	0.0
	0.0	0.03581	0.00109	0.0	3.80e-04	2.00e-05
	0.0	0.00109	0.0363	0.0	2.00e-05	3.80e-04
PyRID	0.04077	0.0	0.0	1.04e-03	0.0	0.0
	0.0	0.03586	0.00108	0.0	3.80e-04	2.00e-05
	0.0	0.00111	0.03634	0.0	2.00e-05	3.80e-04

3.3 Fixed concentration boundary

As mentioned in the methods chapter, fixed concentration boundary conditions couple the simulation box to a particle bath. Thereby, we can simulate, e.g., a sub-region within a larger system without the need to simulate the dynamics of the molecules outside simulation box directly. As an example system we take a 3d model of synapse. The post- and presynaptic spine are both contained inside the simulation volume whereas dendrite and axon are cutoff at the simulation box border (Fig. 3.3 A). We define three molecular species: Species A diffuses in the volume outside the spines (in the extracellular space), species B is located inside the postsynaptic spine and species C on the surface (within the membrane) of the postsynaptic spine. All species consist of a single particle with radius $2\,nm$. The diffusion coefficient is calculated from the the Einstein relation where the temperature is set to $293.15\,K$. The viscosity is set to $1\,mPa \cdot s$ and the time step to $10\,ns$. The simulation box size is set to $250\,nm \cdot 250\,nm \cdot 350\,nm$. At the beginning there are no molecules inside the simulation box. However, the outside concentration of each species is set to 1000 molecules per total volume or total surface area respectively. Thereby, there should be 1000 molecules of each species in the volume and on the surface of each compartment as soon as the system has reached its equilibrium state. Indeed, after about $0.5\,ms$ the system has reached equilibrium and the number of each species fluctuates around the number 1000 (Fig. 3.3 B). As one would expect, species A fills the simulation volume the fastest as the border area is the largest. Species B and C which are located in the volume and on the surfaces of the postsynaptic compartment fill the simulation volume at about the same rate.

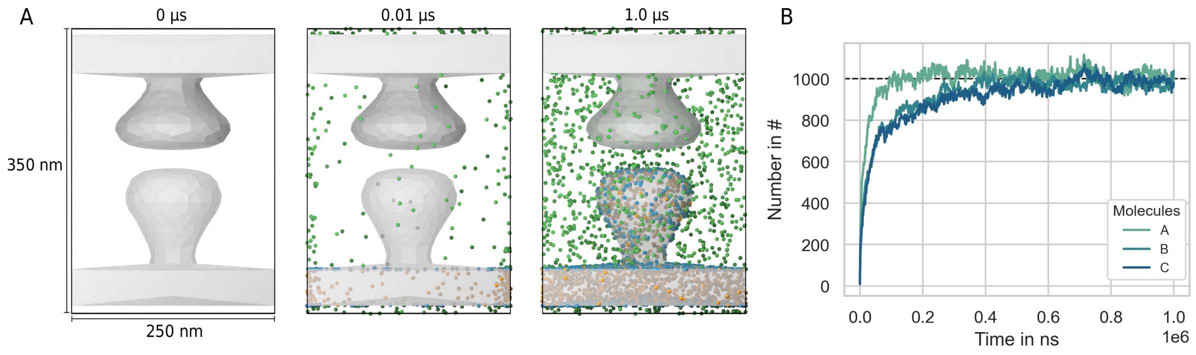


Fig 3.3. Fixed concentration boundary conditions result in the system approaching a target molecule concentration per compartment. (A) We start with an empty scene (left). However, because the molecule concentration of virtual molecules outside the simulation box is above zero, surface and volume molecules enter the system via the boundary (middle). After around 500 ns, the molecule concentration inside the simulation box reaches the target concentration **(A)** right, **(B)**.

3.4 Choosing the right reaction rate and radius

As described in [Schöneberg and Noé, 2013], the reaction radius R_{react} may be interpreted as the distance at which two particles can no longer be treated as moving independently, because there interactions becomes significant. Furthermore, Schöneberg and Noé [2013] suggest that the length scale of electrostatic interactions can be used to define R_{react}. In general, the reaction radius should not be so large that in dense settings molecules would react with a partner that is not among the nearest neighbours. However, R_{react} should also not be smaller than the average change in the distance between molecules, which is given by $\lambda_{AB} = \sqrt{4(D^t_A + D^t_B)\Delta t}$, where D^t_A and D^t_B are the translational diffusion constants of two molecular species A and B. Otherwise, a molecule might pass many reaction partners in between two time steps where the bi-molecular reactions are not evaluated [Erban and Chapman, 2009]. However, even if $\lambda_{AB} \approx R_{react}$ the system would still correctly reproduce the deterministic rate equation description of the reaction kinetics. Of course, in any case, R_{react} should not be chosen smaller than the radius of excluded volume of the molecule species in the presence of repulsive interactions. A description of the reaction kinetics in terms of a system of differential equations assumes a well mixed system. Therefore, the simulation results are also only directly comparable with the ODE approach, if the reactions are reaction rate limited, not diffusion limited such that the system has enough time to equilibrate in between reactions. Let us take a very simple example where A + B $\longrightarrow$ C. If the reaction kinetics are rate limited, the reaction products do not have enough time to mix with the rest of the system. Thereby, regions of low educt concentration evolve where reactions had occurred, while in the regions where no reactions occurred yet, the concentration of educts stays approximately same as in the beginning. Therefore, for the remaining educts in the system, the probability of encounter stays approximately the same. In contrast, if we assume a well stirred system, the concentration of educts would globaly decrease in time, lowering the probability of educt encounters. Therefore, the reaction kinetics are sped up in the stochastic simulation compared to the ode approach (Fig. 3.4). Interestingly, Schöneberg and Noé [2013] found exactly the opposite effect, as the reaction kinetics where slowed down in the stochastic simulation. The reason for this discrepancy in the results is unclear. However, I simulated the very same system in ReaDDy and got the same result as with PyRID.

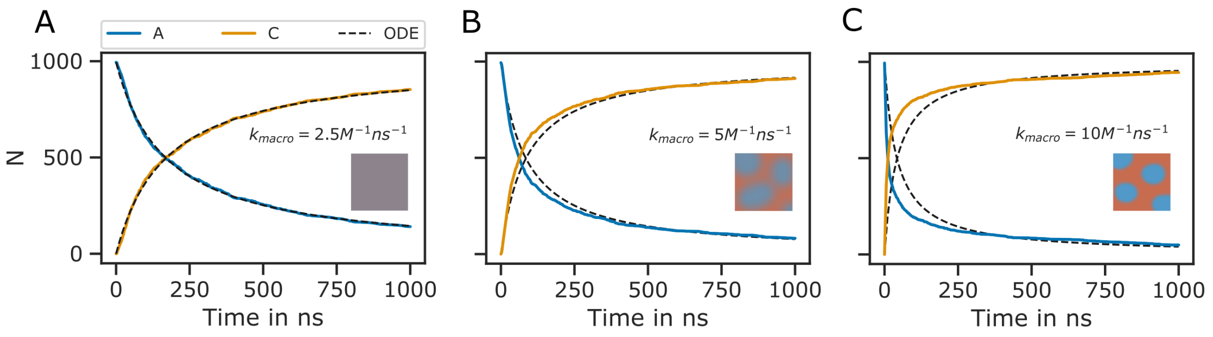


Fig 3.4. Diffusion limited bi-molecular reactions are not accurately described by ODEs. Shown is the minimal system $\mathrm{A} + \mathrm{B} \xrightarrow{\mathrm{k_1}} \mathrm{C}$ with $R_{react} = 4.5nm$ and $\sigma_A = 3nm$, $\sigma_B = 4.5nm$, $\sigma_C = 3.12nm$. The same system has been used for validation of ReaDDy in [Schöneberg and Noé, 2013]. The ODE approach to the description of the reaction kinetics assumes a well mixed system. If the reaction rate is small, the system has enough time to equilibrate in between reactions and the ODE approach (black dotted lines) and the particle-based SSA approach (colored lines) match (**A**). As the reaction rate increases (**B**-**C**) this is no longer the case, as the system is no longer well mixed at any point in time. Here, the system can be divided into regions of high and low educt concentrations (depicted by the small insets). Thereby, at the onset, the reaction kinetics in the stochastic simulation are faster than predicted by the ODE approach (**B**, **C**). However, when a critical mass of educts have reacted, the slow diffusion has an opposite effect on the reaction kinetics as the probability of isolated single educts to collide becomes lower than in the well mixed case. The slow down effect is especially prominent in **B**, **C** at around 500 ns. The reaction kinetics are therefore better described by two exponential functions instead of one.

Given a reaction radius R_{react}, we would like to know at what reaction rate k_t a simulation would match an experimentally measured macroscopic reaction rate k^{macro}. For two non-interacting molecule species A and B with translational diffusion constants D^t_A and D^t_B and $\lambda_{AB} << R_{react}$, k_{macro} is given by [Erban and Chapman, 2009]

$$k_{macro} = 4\pi(D^t_A + D^t_B) \left[R_{react} - \sqrt{\frac{D^t_A + D^t_B}{k_t}} \tanh\left(R_{react} \sqrt{\frac{k_t}{D^t_A + D^t_B}} \right) \right] \tag{3.9}$$

Equation 3.9 can solved numerically for k_t. Also, if the $k_t \rightarrow \infty$, 3.9 simplifies to the Smoluchowski equation where we can express the reaction radius in terms of the macroscopic reaction rate [Erban and Chapman, 2009]:

$$R_{react} = \frac{k_{macro}}{4\pi(D^t_A + D^t_B)} \tag{3.10}$$

In the limit where $k_t << \frac{D^t_A + D^t_B}{R^2_{react}}$, Eq. 3.9 can be Taylor expanded and simplifies to [Erban and Chapman, 2009]:

$$k_t = \frac{k_{macro}}{4/3\pi R^3_{react}} \tag{3.11}$$

The above equations are, however, only valid in the case where molecules are represented by single particles and also only in 3 dimensions. PyRID has a build in method to calculate the reaction rates and radii based on equation 3.9.

3.5 Bi-molecular reactions between rigid bead molecules

The representation of molecules by single particles neglects the complex structure of molecules. Bi-molecular reactions between proteins can occur via different reaction sites. Therefore, also here, the isotropic picture breaks down. PyRID enables the simulation of reactions between complex molecules having different reaction sites. Different reaction sites are represented by beads/patches that are part of the rigid bead molecules topology. Similar to uni-molecular reactions, bi-molecular reactions can be defined on particles or molecules. However, because PyRID only computes the distances between the particles in the system, also reactions that are defined on the molecule level need to be linked to a particle type pair. If the the two particles are within the reaction distance and if the reaction is successful, the reaction itself will, however, be executed on the respective molecule types. As an example, we again consider the simple system $\mathrm{A + B \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C}$. However, molecules A and B are each represented by two beads a_1, a_2 and b_1, b_2. Also, we add another reaction path $\mathrm{A + B \xrightarrow{k_2} D}$. We now may define reactions for different pair permutations of the available beads:

$$\begin{aligned}
\mathrm{A(a_1) + B(b_1)} &\xrightarrow{\mathrm{k_1, R_1}} \mathrm{C} \\
\mathrm{A(a_1) + B(b_1)} &\xrightarrow{\mathrm{k_2, R_2}} \mathrm{D} \\
\mathrm{A(a_1) + B(b_2)} &\xrightarrow{\mathrm{k_3, R_3}} \mathrm{C} \\
\mathrm{A(a_2) + B(b_2)} &\xrightarrow{\mathrm{k_4, R_4}} \mathrm{C}
\end{aligned} \tag{3.12}$$

where k_i are the microscopic reaction rates and R_i the reaction radii. For better visualization, also see figure 3.5 A and B. As such, molecules A and B can undergo fusion to molecule C via three pathways, defined by three bead pairs $(a_1, b_1), (a_1, b_2), (a_2, b_2)$. Whereas for the particle pairs (a_1, b_2) and (a_2, b_2) only one reaction pathway is defined respectively, for the particle pair (a_1, b_1) a second reaction path has been defined for the fusion of molecules A and B to molecule C. We may also describe this system in terms of a system of ODEs:

$$\begin{aligned}
\frac{dA}{dt} &= -(k^1_{macro} + k^3_{macro} + k^4_{macro})AB - k^2_{macro}AB + k^{-1}_{macro}C \\
\frac{dB}{dt} &= -(k^1_{macro} + k^3_{macro} + k^4_{macro})AB - k^2_{macro}AB + k^{-1}_{macro}C \\
\frac{dC}{dt} &= (k^1_{macro} + k^3_{macro} + k^4_{macro})AB - k^{-1}_{macro}C \\
\frac{dD}{dt} &= k^2_{macro}AB
\end{aligned} \tag{3.13}$$

The macroscopic rate constants k^i_{macro} can be calculated from Eq. 3.9. Note, however, that for more complex molecules Eq. 3.9 does not hold true, because we would also need to take into account the rotational motion of the molecule in addition to the translational diffusion constant that describes the motion of the molecule center. In our example, the bead motion is, however, close enough to that of a single spherical particle such that the results from the Brownian dynamics simulation are in close agreement with the ODE formulation (Fig. 3.5 C).

At this point one might argue that there is only little to no benefit of the rigid bead model description over other SSA schemes. And in principle that is true. Systems such as the above could also be modeled using single particle Brownian dynamics or even ODEs. However, if we take into account the excluded volume of the molecules by introducing a repulsive interactions between the beads, the reaction kinetics differ from the ODE solution (Fig. 3.5 D). The bead radii are chosen equal to the reaction radius, where $\sigma_{a_1} = 2.0nm$, $\sigma_{a_2} = 1.5nm$, $\sigma_{b_1} = 2.0nm$, $\sigma_{b_2} = 3.0nm$. Thereby, the molecules react upon contact. For such simple molecules one could, however, neglect the bead topology and approximate the molecules by single beads with repulsive interactions and get a very similar result. For more complex molecules where the reaction volumes are much more anisotropic, one would, however, expect a larger deviation from the repulsive sphere approximation. The benefits of the rigid bead model approach become more important when we consider binding reactions.

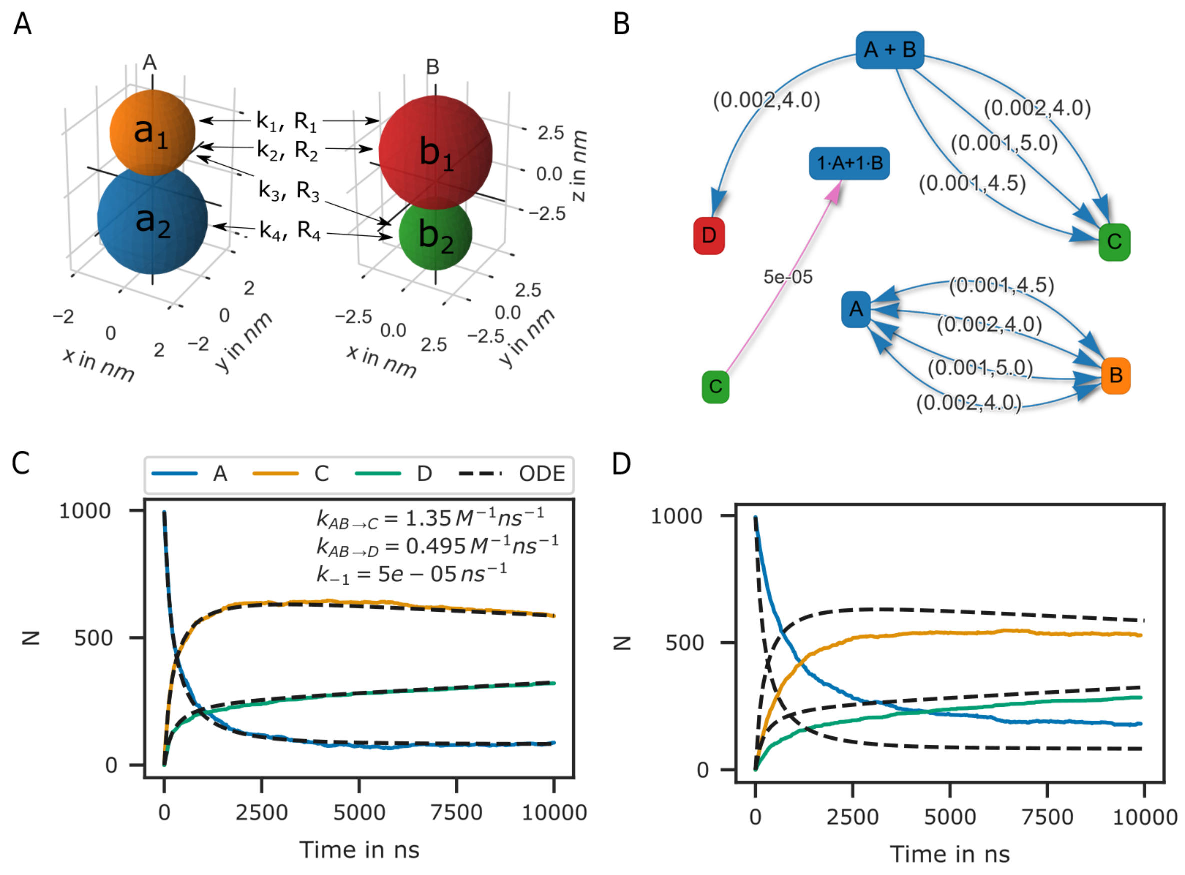


Fig 3.5. Bi-molecular reaction between two rigid bead molecules.(A) Depiction of the two rigid bead molecules and the different reactions defined on their respective particles/beads. **(B)** Reaction graphs showing the different reaction paths for the fusion reactions $\mathrm{A} + \mathrm{B} \longrightarrow \mathrm{C}$ and $\mathrm{A} + \mathrm{B} \longrightarrow \mathrm{D}$ as well as the fission reaction $\mathrm{C} \longrightarrow \mathrm{A} + \mathrm{B}$. The lower right graph simply depicts the different reaction paths between the two educts A and B without specifying the products. In total there are 4 paths (Eq. 3.12). **(C)** If not accounting for any repulsive interaction between molecules A and B, the simulation results are in good agreement with the ODE description (Eq. 3.13). **(D)** However, if we account for the excluded volume of the molecules by a repulsive interaction potential, the results of the two approaches (particle dynamics and ODE description) differ.

3.6 Reactions between surface molecules

As a model, let us consider a four component system and implement a simple autocatalytic reaction scheme. The system consists of a freely diffusing transmembrane molecule U. In addition, we add a second, freely diffusing, surface molecule P. Let U and P form a complex B via a fusion reaction:

$$\mathrm{U} + \mathrm{P} \xrightarrow{\mathrm{k_{on}}} \mathrm{B} \tag{3.14}$$

The reaction rates are set to $k_{on} = 1e - 5ns^{-1}$ and $R_{react} = 4nm$. In addition, we add an enzymatic/katalytic reaction:

$$\mathrm{B} + \mathrm{P} \xrightarrow{\mathrm{k_{enz}}} \mathrm{B} + \mathrm{P'}. \tag{3.15}$$

We also account for a reverse reaction where

$$\mathrm{P'} \xrightarrow{\mathrm{k_{-enz}}} \mathrm{P}. \tag{3.16}$$

Here, $k_{enz} = 1e - 3ns^{-1}$, $k_{-enz} = 5e - 5ns^{-1}$ and $R_{react} = 4nm$. The reaction product P' has a much higher binding affinity for U:

$$\mathrm{U} + \mathrm{P'} \xrightarrow{\mathrm{k_{on}}'} \mathrm{B}, \tag{3.17}$$

with $k'_{on} = 1e - 2ns^{-1}$ and $R_{react} = 4nm$ (note that $k'_{on} >> k_{on}$). The break up of the complex is accounted for by a fission reaction

$$\mathrm{B} \xrightarrow{\mathrm{k_{off}}} \mathrm{P} + \mathrm{U}. \tag{3.18}$$

As expected from an autocatalytic reaction, the product B follows a sigmoid function (Fig. 3.6). We may compare the simulation result to the corresponding ODE description. The above system expressed in terms of a system of ODEs reads

$$\begin{aligned}
\frac{dU}{dt} &= -U\,P\,k_{macro}^{on} - U\,P'\,k_{macro}^{on\prime} + B\,k_{macro}^{off} \\
\frac{dB}{dt} &= U\,P\,k_{macro}^{on} + U\,P'\,k_{macro}^{on\prime} - B\,k_{macro}^{off} \\
\frac{dP}{dt} &= -U\,P\,k_{macro}^{on} - P\,B\,k_{macro}^{enz} + P'\,k_{macro}^{-enz} + B\,k_{macro}^{off} \\
\frac{dP'}{dt} &= B\,P\,k_{macro}^{enz} - P'\,k_{macro}^{-enz} - U\,P_{a}ct\,k_{macro}^{on\prime}
\end{aligned} \tag{3.19}$$

However, equation 3.9 is only valid in the 3D case and a solution for the 2D case is difficult to derive as the rate constant is concentration dependent. A closed form analytical expression has not yet been derived for the Doi scheme [Erban and Chapman, 2009, Galanti et al., 2019, Crank, 1980, Berg, 1984]. A more in depth discussion on this topic and theoretical results for the Smoluchowski theory can be found in [Yogurtcu and Johnson, 2015]. However, for the current system the simulation result can be matched using a constant reaction rate $k_{macro}^{2D} = k_{macro}/5.6$ despite the decay in molecule density over time (Fig. 3.6A). A closed form expression for k_{macro} is extremely useful when setting up a reaction diffusion simulation. Even if results do not match exactly, the ODE approach can help to choose the correct parameters for a particle-based simulation that might take several order longer than solving the system of ODEs. Whereas the ODE description is useful in many regards, we usually decide to do a particle based simulation because we are interested in settings that are not well mixed, or where interactions between molecules play a role.

3.6.1 Toy model of the PSD

As an example, where the particle-based approach becomes essential, we may transfer the autocatalytic system from above to a simplified model of the postsynapse. In our new setting, P, P' and B no longer diffuse but are fixed to a region that we interpret as being the PSD. In addition, a 3d mesh of a postsynaptic spine is introduced to the simulation and species U enters the simulation volume where the extrasynaptic region intersects the simulation box via a fixed concentration boundary. P now represents a receptor binding site, U the freely diffusing receptors and B the bound receptor or an occupied binding site. In this adapted system the autocatalytic reaction scheme results in receptor clustering (Fig. 3.6B). Note that whereas the reaction $\mathrm{B} + \mathrm{P} \xrightarrow{\mathrm{k_{enz}}} \mathrm{B} + \mathrm{P}'$ is implemented as an enzymatic reaction in PyRID, the physical interpretation could be very different. For example, the conversion of $\mathrm{P} \longrightarrow \mathrm{P}'$ could occur only indirectly via the complex B and by a local signaling pathway that includes other molecules that we do not model here explicitly. Important is only that this pathway is triggered by B and that it is locally restricted for receptor clusters to evolve.

3.7 Hard sphere fluid

A hard sphere fluid is very useful for validation as there exist analytic expressions for the radial distribution function but also for the pressure.

3.7.1 Radial distribution function

Figure 3.7 shows the radial distribution function for a hard sphere fluid that is modelled using the harmonic repulsive interaction potential (Eq. 2.50). The sphere diameter is set to $1\,nm$. The simulation result is is in good agreement with a closed-form analytical expressions of the hard sphere radial distribution function [Trokhymchuk et al., 2005] (Fig. 3.7 B). The analytical expression for the radial distribution function is too long to be presented here. The interested reader is referred to [Trokhymchuk et al., 2005].

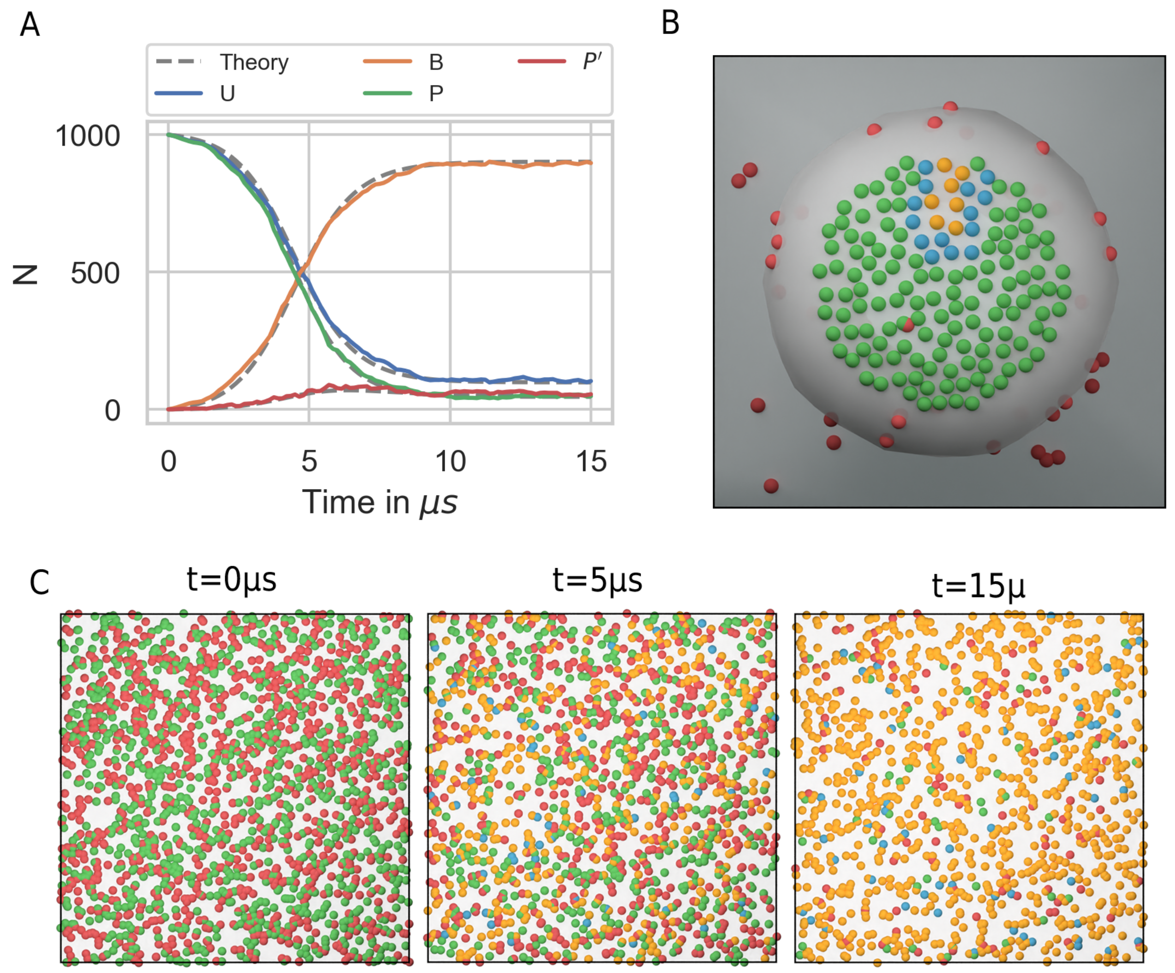


Fig 3.6. Autocatalytic reaction diffusion system in 2D. (A) Number of the different molecular species evolving according to the reactions defined by equations 3.14-3.18. The simulation results are matched by the ODE description by fitting the macroscopic reaction rates (dotted grey lines). **(B)** Toy model of the PSD. Using the reaction scheme defined by equations 3.14-3.18 but fixing the position of species P, P' and B we observe the formation of species cluster (U in red, P in green, P' in blue and B in yellow). **(C)** Evolution of the autocatalytic reaction system shown in **(A)** at different points in time (U in red, P in green, P' in blue and B in yellow).

3.7.2 Pressure

We can use a hard sphere fluid for validation of the pressure calculation. For a hard sphere fluid, an analytical expression for the pressure is given in terms of the radial distribution function at contact and the second virial coefficient [Tao et al., 1992]:

$$p = \rho k_B T + \rho^2 k_B T b g(\sigma^+), \tag{3.20}$$

where σ is the hard-sphere diameter, ρ the number density and $b = (2\pi/3)\sigma^3$ the second virial coefficient. The radial distribution function at contact can be approximated by the

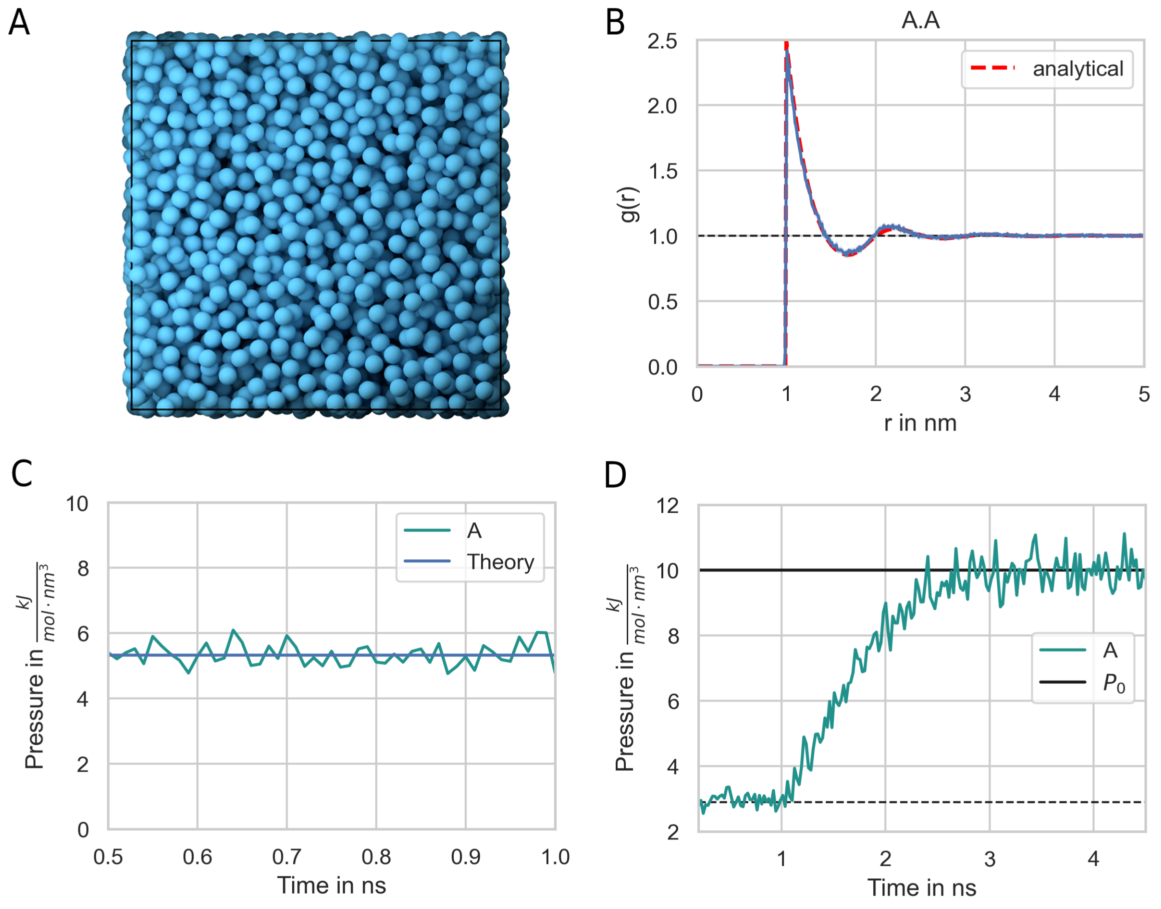


Fig 3.7. Hard-sphere radial distribution function. (A) The system is set up with a packing fraction of $\eta = 0.3$. The particle diameter is set to 1 nm and pair interactions occur via a harmonic repulsive potential. **(B)** The resulting radial distribution function (blue line) is in close agreement with theoretical prediction (red line). **(C)** The pressure of the hard-sphere fluid obtained from simulations is also in close agreement with theory [Trokhymchuk et al., 2005]. (D) A hard-sphere fluid NPT ensemble simulation. From time 0.5 ns, the Berendsen barostat is activated and drives the system to the target pressure $P_0 = 10\,\mathrm{kJ/(mol\,nm^3)} = 16.6\,\mathrm{MPa} = 166\,\mathrm{bar}$.

solution to the Percus-Yevick equation [Hansen Jean-Pierre, 2013]:

$$g_{PY}(\sigma) = \frac{1 + \eta/2}{(1-\eta)^2}, \tag{3.21}$$

where $\eta = (\pi/6)\rho\sigma^3$ is the packing fraction. The pressure obtained from the simulation of a hard sphere fluid is in close agreement with this theoretical result (Fig. 3.7 **C**). In addition, the system does reach the target pressure using the Berendsen barostat (Fig. 3.7 D)

3.8 LLPs of Patchy Particles

Liquid-liquid phase separation (LLPS) gained a lot of interest in recent years as more experimental evidence has been gathered that many cell structures are formed by LLPS. LLPS is a compelling mechanism as it might answer, how cells are able to organize in the presence of a crowded environment with thousands of molecular species [Banani et al., 2017]. Examples include nucleoli, Cajal bodies, stress granules but also the PSD [Zeng et al., 2016]. In an number of papers Zeng et al. have shown that many of the proteins found in the PSD are able to phase separate [Zeng et al., 2016, 2018, 2019]. We would like to better understand the phase behaviour of the PSD as this might have an impact especially on the expression of late phase LTP. PSD substructures change in morphology within half an hour or stay rigid for many hours [Wegner et al., 2018], indicating that the PSD might switch back and forth between crystalline, gel like and liquid states. Another study has shown that synaptic nanomodules, including the PSD are reallocated and change in size in response to synaptic plasticity induction [Hruska et al., 2018, Bosch et al., 2014]. The issue that arises when investigating the phase behaviour of complex molecules is that even modern computers are able to only simulate 10-20 small proteins [Espinosa et al., 2020]. Therefore, coarse graining methods are needed. With models that describe the disordered region of proteins on the level of amino-sequences simulations with a few hundred copy numbers are already feasible [Dignon et al., 2018, Espinosa et al., 2020]. A minimal coarse graining approach represents proteins by patchy particles where the multivalent interaction sites of the proteins are modeled by attractive patches whereas the excluded volume is represented by a core particle with repulsive interactions. Espinosa et al. [2020] have used such a model to investigate the stability and composition of biomolecular condensates. PyRID is well suited for simulations of patchy particles. For validation I here reproduce one of the results from [Espinosa et al., 2020]. In their work, patches interact via an attractive square well interaction potential [Espinosa et al., 2014]:

$$U_{CSW}(r) = -\frac{\epsilon_{CSW}}{2}\Big[1 - \tanh\Big(\frac{r - r_w}{\alpha}\Big)\Big], \tag{3.22}$$

where $\alpha = 0.01\sigma$ with σ being the hard sphere radius. The core particles interact via a pseudo hard sphere potential [Jover et al., 2012]:

$$U_{HS} = \begin{cases} \lambda_r(\frac{\lambda_r}{\lambda_a})^{\lambda_a}\epsilon_R[(\frac{\sigma}{r})^{\lambda_r} - (\frac{\sigma}{r})^{\lambda_a}] + \epsilon_R, \text{ if } r < (\frac{\lambda_r}{\lambda_a})\sigma \\ 0, \text{ if } r < (\frac{\lambda_r}{\lambda_a})\sigma, \end{cases} \tag{3.23}$$

where $\lambda_a = 49$ and $\lambda_r = 50$. ϵ_R is the energy constant, r_w is the radius of the attractive well and α determines the steepness of the potential well edge. To ensure that each patch does at maximum interact with one other patch at any time, in [Espinosa et al., 2020] r_w

has been set to 0.12σ. Here, I did the same, however, note that thanks to the ability to define binding reactions in PyRID we could in principle also choose a larger radius for the attractive interaction potential. To compute the phase diagram/the coexistence curve for a patchy particle fluid, [Espinosa et al., 2020] used the direct coexistence method. The system is initialized at a volume density of ≈ 0.3 in a cubic box with 2000 patchy particles and at a temperature of $179.71K$. Periodic boundary conditions are used. The integration time step was set to $2.5ps$. A small integration time step is necessary due to the very short and steep attractive interaction between patches. Note that for Brownian dynamics simulations one would ideally use a weaker, soft interaction potential. Also, for such small integration time steps, the Brownian assumption is not necessarily valid anymore, e.i. the diffusive motion is not accurately described by a Markov process. However, we will see that, nevertheless, the results from [Espinosa et al., 2020] can be reproduced fairly accurately using the Brownian dynamics approach. In the following I briefly describe the direct coexistence method as used in [Espinosa et al., 2020]. In a first step, the patchy particle fluid is equilibrated in an NPT simulation at zero pressure and an energy constant ϵ_{CSW} that is high enough to ensure phase separation. Thereby, the value of ϵ_{CSW} depends on the system temperature and the patchy particle valency. As a rule of thumb, $\frac{k_B T}{\epsilon_{CSW}}$ should be smaller than 0.1. For the equilibration phase I used the highest value that is given in table 3.2 for the different valencies respectively. After the equilibration phase the simulation box is elongated along the x-axis by a factor of 3. Thereby, a two phase system is created with infinite dense and dilute sheets. The elongated system is then simulated in the NVT ensemble for various different values of ϵ_{CSW} (see table 3.2). The simulation is continued until the system reaches a new equilibrium, which was the case after $\approx 2e7$ steps at approximately $120\,it/s$. Thereby, a single simulation took $\approx 2\,days$. In total 33 such simulations, 11 for each of the three valency cases, were executed on a high compute cluster. In a final step, a concentration profile is sampled, from which the volume fraction of the dense and dilute phase are estimated [Espinosa et al., 2019]. I found that the coexistence curves acquired with PyRID were in good agreement with [Espinosa et al., 2020] (Fig. 3.8). However, for the 5-valency case, Espinosa et al. [2020] found a slightly higher volume fraction in the dense phase close to the critical point. Also, Espinosa et al. [2020] found that the coexistence curve shows minimum below the critical for the 4-valence case, which I did not observe. The reason could lie in inaccuracies that are a result of to the Brownian approximation. More probable is, however, that the choice of the thermostat is responsible for the discrepancy as Espinosa et al. [2020] used a Nosé-Hoover thermostat instead of a Langevin thermostat. However, I would argue that a Langevin thermostat, or in this case overdamped langevin dynamics/Brownian dynamics, represent the dilute phase more accurately as it accounts for the interaction with the solvent molecules.

Table 3.2. Parameters for the patchy particle LLPS simulation.

valency	ϵ_{CSW} in $\frac{kJ}{mol}$
3 sites	$14.5 - 23.3$
4 sites	$12.0 - 20.0$
5 sites	$10.5 - 16.0$

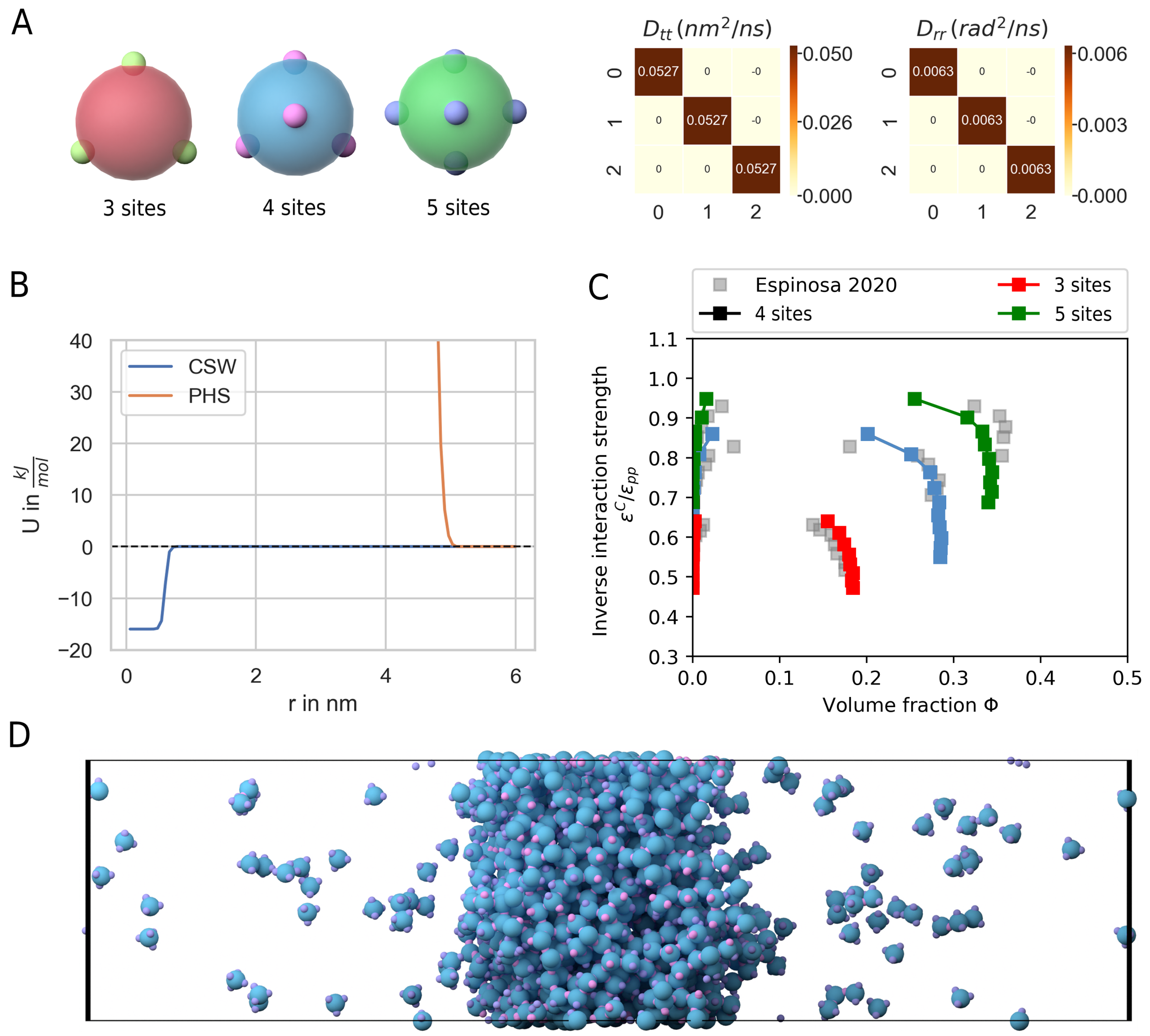


Fig 3.8. LLPS of patchy particles. (A) Patchy particles with 3, 4 and 5 sites. Left: The translational and rotational diffusion tensor. **(B)** Graph of the continuous square-well potential (CSW) used for the attractive patches and the pseudo hard sphere potential (PHS) used for the core particle. **(C)** Coexistence curves for the 3, 4 and 5 sided patchy particle systems and comparison with the results from Espinosa et al. [2020]. **(D)** Side view showing the dilute and dense phase for the 4-sided patchy particle system.

3.9 Benchmarks

To benchmark PyRID I directly compare it to ReaDDy. As a benchmark test I will therefore use the same that has been used in [Hoffmann et al., 2019]. The system consists of the molecule types A, B and C with radii $1.5\,nm$, $3.0\,nm$, and $3.12\,nm$. The viscosity is set to $1.0 mPa \cdot s$, which is the value for water at about 293 Kelvin (20°C). The molecules all interact via a harmonic repulsive potential [Hoffmann et al., 2019]:

$$U(r) = \begin{cases} \frac{\kappa}{2}(r-\sigma)^2, \text{ if } r \leq \sigma \\ 0, \text{ otherwise }, \end{cases} \tag{3.24}$$

where the force constant $\kappa = 10kJ/mol$. The interaction distance σ is given by the radii of the interacting molecule pair. In addition, the molecules take part in the reaction $\mathrm{A} + \mathrm{B} \underset{\mathrm{k_{-1}}}{\overset{\mathrm{k_1}}{\rightleftharpoons}} \mathrm{C}$. The rate for the fusion reaction is $k_1 = 0.001 ns^{-1}$ and the reaction radius $R_{react} = 4.5nm$. The fission reaction rate is set to $k_{-1} = 5 \cdot 10^{-5} ns^{-1}$ and the dissociation radius is set equal to R_{react}. The benchmark is carried out for different values of the total initial molecule number N_{tot}, with $N_A = N_{tot}/4$, $N_B = N_{tot}/4$, $N_C = N_{tot}/2$. The number density is, however, kept constant at $\rho_{tot} = 0.00341nm^{-3}$ by scaling the simulation box accordingly. Simulations are carried out for $300ns$ with an integration time step of $0.1ns$. The result of the performance test is shown in figure 3.9 B. For particle numbers between 1.000 and 10.000, the computation time per particle update stays approximately constant at $1.25\mu s$, which corresponds to about 800.000 particle updates per second. For particle numbers above 10.000, the performance starts to drop slightly (Fig. 3.9 B, blue line). The benchmark test has been performed on a machine with an Intel Core i5-9300H with 2.4 GHz and 24 GB DDR4 RAM. Interestingly, PyRID always performed better than ReaDDy for this benchmark test (Fig. 3.9 B, yellow line). Also, ReaDDy scaled less linear for large particle numbers than PyRID. Shown are the results for ReaDDy ran on the sequential kernel. In addition, I performed the benchmark test for the parallel kernel but the results were always worse. However, in [Hoffmann et al., 2019], where the same benchmark test has been used, ReaDDy scaled much better and there was almost no performance drop even at 100.000 particles for the sequential kernel (Fig. 3.9 B, green line). Also, performance increased significantly when using the parallel kernel (down to $\approx 0.5\mu s$). The performance has been tested on a slightly faster but comparable machine with an Intel Core i7 6850K processor at 3.8GHz and 32GB DDR4 RAM. The faster machine is probably the cause for the better performance at particle numbers below 10.000 particles in comparison with my results. However, I can only speculate why ReaDDys' scaling behavior for large particle numbers is much less linear in my benchmark test and why multi-threading only let to a performance loss. The reason might be that in [Hoffmann et al., 2019] ReaDDy was compiled for their benchmark system whereas I used the binaries distributed by the developers behind ReaDDy. Nonetheless, the benchmark test shows that PyRIDs performance is very much comparable with ReaDDy, and at least in certain situation PyRID can even outperform ReaDDy. Thereby, for a system with 10^4 particles, PyRID is able to perform at $\approx 80it/s$ and $\approx 7 \cdot 10^6/day$. At an integration time step of $1ns$, therefore, $7ms$ per day can be simulated on medium machine.

3.9.1 Polydispersity

As mentioned in the methods chapter, PyRID uses a hierarchical to efficiently handle polydispersity. As a test, a two component system is used. Both components consist of a single particle. Component A has a radius of $10\,nm$, component B has a radius of $2.5\,nm$. The simulation box measures $75\,nm \cdot 75\,nm \cdot 75\,nm$. The simulation volume is densely packed with both components such that we reach a volume fraction of 52%. The simulation ran for $1e4$ steps. When not using the hierarchical grid approach but the classical linked cell list algorithm, PyRID only reaches about 80000 particle updates per second (pu/s) on average (Fig. 3.9 A). However, when using the hierarchical grid, more than 500000 pu/s are reached (Fig. 3.9 A). If instead of the two component system we only simulate a one component system, PyRID also only reaches about 500000 pu/s (Fig. 3.9 A). Thereby, PyRID performs similar independent of whether the system is mono- or polydisperse.

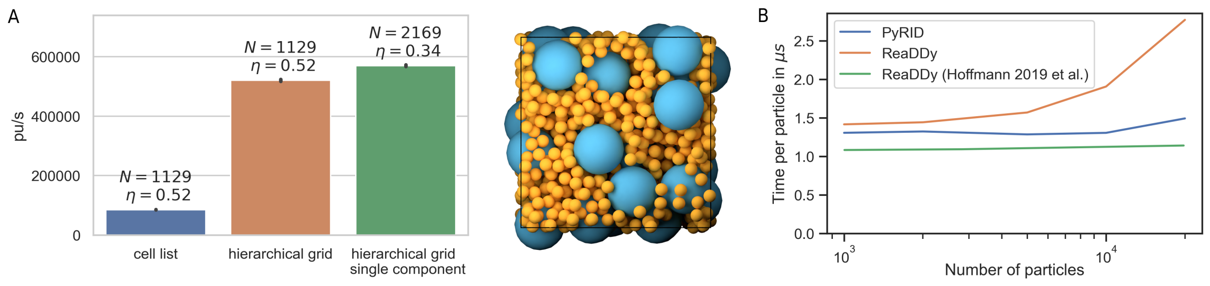


Fig 3.9. Performance test of the hierarchical grid approach. (A) Performance hierarchical grid. **(B)** Performance comparison between PyRID and ReaDDy. On a benchmark system with an Intel Core i5-9300H with 2.4 GHz and 24 GB DDR4 RAM, PyRID (blue line) outperforms ReaDDy (yellow). However, Hoffmann et al. [2019] obtained a better performance and especially scaling for ReaDDy on a different machine with an Intel Core i7 6850K processor at 3.8GHz and 32GB DDR4 RAM (green line).

4 Discussion

PyRID is a fast and flexible tool for particle-based reaction diffusion simulations with pair-interactions. However, a challenge remains in that PyRID is not able simulate processes that take several seconds or even minutes. However, in cell biology we find many processes that act on such time scales. This is true for signaling processes, for self-assembly processes, e.g., of clathrin, and for protein trafficking. Other tools that follow a similar approach such as ReaDDy [Hoffmann et al., 2019, Schöneberg and Noé, 2013] also do not provide methods that would enable simulations on such long time scales. Tools such a MCell [Kerr et al., 2008] and Smoldyn [Andrews, 2016] enable simulations on larger time scales but do not resolve molecular structure and are not able to simulate protein binding and assembly. However, alternative reaction-rate based approaches have been developed that account for molecule structure, binding, and diffusion. A prominent example is NERDSS [Varga et al., 2020] that is able to resolve fast binding reactions as well as processes on large time and- spatial scales. In NERDSS, molecules are represented by rigid bodies, similar to PyRID. Also, excluded volume is accounted for by rejection sampling. However, NERDSS avoids any energy interaction functions but instead molecules ”snap” into place in a predefined way when a binding reaction is executed. Thereby, assembly processes can be simulated very efficiently. Still, since the resulting assembly has a predefined form NERDSS is not able make predictions about the structure of protein assemblies. Also, binding reactions are not orientation dependent which can result in unrealistic binding events, whereas in PyRID orientation dependence is accounted for by construction. Also, with NERDSS, one relies on reaction rates that have been measured either in experiment or by molecular dynamics simulations to describe assembly and disassembly processes. However, also with PyRID such processes can not accurately be modeled without exactly specifying the energy functions of the binding interaction which can even be harder than estimating rates. At last, due to the lack of interaction forces in NERDSS, any physical properties that are derived from the interaction forces or the energy functions can not be computed and flexible chains of beads or molecules are not supported. Still, such solely rate based approaches are very promising if one is interested in the kinetics of complex assembly processes and could also be a valuable addition to PyRID. In principle the PyRID framework would allow rigid body assembly growth so this could be a useful future extension. However, as indicated above, there exist many settings in which we need to include energy functions and where, as a result, there is a an upper limit for the integration time step that we can choose. We can slightly shift this threshold towards larger time steps by using different approximations to the intermolecular interaction energy functions but even then we are restricted to integration time steps $\leq 1ns$. As such we need to speed up computation, e.g. by parallelization. For very large system simulations many molecular dynamics tools such as LAMMPS support parallel implementations of their algorithms for the message passing interface standard (MPI). However, here, we only gain a benefit in speed for large systems as message passing otherwise becomes a bottle neck. For intermediate sized system such as those that we want to simulate with PyRID containing 10.000-100.000 particles, algorithms that run on the GPU are much more promising. A good example is the molecular dynamics tool HooMD [Anderson et al., 2020] that is optimized for the GPU and that can reach speed ups of up to two order of magnitude compared to a single CPU and more than one order of magnitude compared to a modern multi-core CPU [Anderson et al., 2020]. As such, bringing particle-based reaction diffusion simulations to the GPU could be the key for simulations on time scales of even minutes. The question remains to what degree the required algorithms and data structures can be efficiently ported to the GPU. However, this is beyond the scope of this work. At last, machine learning long found its way into MD simulations and is used in coarse graining, molecular kinetics and more [Noé et al., 2020].

4.1 On hydrodynamic interactions

As mentioned above, PyRID does not account for hydrodynamic interactions between molecules because, in this case, the kind of simulations for which PyRID has been developed would become unfeasible. Here, the 6Nx6N diffusion tensor of the entire system is needed to propagate the molecule positions. As the molecule positions change each time step, this diffusion tensor needs to be recalculated each iteration. A discussion on this topic in terms of many particle simulations can also be found in [Geyer, 2011]. A new algorithm that scales $O(N^2)$ has been introduced by [Geyer and Winter, 2009] making larger simulations with hydrodynamic interactions more feasible.

4.2 Limitations of the Brownian dynamics approach

Brownian dynamics simulations come with some limitations that one should consider [Snook, 2007]: In BD, only time steps are considered that are much longer than the velocity relaxation time $\tau_{rel} = \frac{m}{\gamma} = \frac{2\rho r^2}{9\eta}$. Due to the strong damping forces in a viscous fluid the kinetic energy of large molecules rapidly dissipates. Thereby, the erratic movement of the molecules in between two time steps is memory less and can be described as a Markov process. However, when accounting for interactions between molecules, the integration time step must also not be too small such that forces stay approximately constant in one time step. This becomes a problem for small interacting molecules or atoms, where the time step needs to be chosen small enough to resolve the interactions but large enough for the the approximation of over-damped kinetics. Thereby, if the molecules are of similar size as the solvent molecules, BD may not correctly describe the dynamics. Winter and Geyer [2009] introduced a Langevin integration scheme that enables the accurate simulation of small molecules. Here, we will however use the well established BD scheme introduced by Ermak and McCammon [1978]. Another thing to keep in mind is that BD is only applicable for Newtonian fluids. Also, since the details of the interaction between solvent particles and bead particles are neglected, simulations of molecule aggregation may not be correctly described. However, in the case where aggregation is dominated by the interaction between the proteins, the latter may be negligible.

5 Appendix

5.1 Appendix A

Since the method goes beyond what is found in most textbooks, I will give an introduction to the method in the following. However, I will not derive the methods in detail since this has been done in various publications [Torre and Bloomfield, 1977a,b, Carrasco and de la Torre, 1999a,b, de la Torre et al., 2007].

5.1.1 The Oseen tensor and hydrodynamic interaction between beads

The Oseen tensor has first been introduced by Oseen in 1927 (for reference also see [Dhont, 1996]). The Oseen tensor emerges from the solution of the Stokes equations (linearization of the Navier-Stokes equations) for the flow velocity field in case of a force acting on a point-like particle ($\boldsymbol{F}(\boldsymbol{r}) = \boldsymbol{F}_0\delta(\boldsymbol{r} - \boldsymbol{r}_p)$) which is immersed in a viscous liquid. In this case, the solution to the Stokes equation can be written as a linear transformation (due to its linearity, any solution to the Stokes equation has to be a linear transformation):

$$\boldsymbol{v}(\boldsymbol{r}) = \boldsymbol{T}(\boldsymbol{r} - \boldsymbol{r}_p) \cdot \boldsymbol{F}, \tag{5.1}$$

where $\boldsymbol{r}_p$ is the Cartesian coordinate vector of the point-like particle. $\boldsymbol{T}$ is called the hydrodynamic interaction tensor, Oseen tensor or Green's function of the Stoke's equations. The above solution is also called Stokeslet [Oseen, 1927]:

$$\boldsymbol{T}(\boldsymbol{r}) = \frac{1}{8\pi\eta r} \cdot \Big(\boldsymbol{I} + \frac{\boldsymbol{r} \otimes \boldsymbol{r}}{r^2}\Big), \tag{5.2}$$

where η is the fluid viscosity and $\otimes$ is the outer product and $\boldsymbol{I}$ is the identity matrix. Thereby, $\boldsymbol{T}$ relates the fluid flow velocity at some point $\boldsymbol{r}$ to a force acting at another point $\boldsymbol{r}_p$ in the fluid. As mentioned above, the mobility matrix of a system of dispersed subunits/beads can be related to the Oseen tensor. The mobility $\boldsymbol{\mu}$ is defined as the ratio of a particle‘s drift velocity and the applied force; thereby, the Oseen tensor represents an approximation for the hydrodynamic interaction part of the mobility matrix. Bloomfield et al. [1967] first introduced a formulation of the translational mobility tensor for a system of multiple dispersed beads using the Oseen tensor to describe the hydrodynamic interaction between the beads, and by assigning each bead its friction coefficient $\xi_i = 6\pi\eta_0\sigma_i$ [Carrasco and de la Torre, 1999a]:

$$\begin{aligned}\boldsymbol{\mu}_{ij}^{tt} =& \delta_{ij}(6\pi\eta_0\sigma_i)^{-1}\boldsymbol{I} \\ &+ (1-\delta_i j)(8\pi\eta_0 r_{ij})^{-1} \\ &\left(\boldsymbol{I} + \frac{\boldsymbol{r}\otimes\boldsymbol{r}}{r^2}\right)\end{aligned} \tag{5.3}$$

Here, the first term is just the mobility coefficient of a single particle with radius σ_i in the absence of any other beads. The second term is the Oseen tensor. However, since the Oseen tensor only considers the distance between the bead centers but neglects their finite radius σ_i Torre and Bloomfield [1977a] established a correction to the Oseen tensor for nonidentical spheres (also see de la Torre et al. [2007]):

$$\boldsymbol{T}_{ij} = \frac{1}{8\pi\eta r}\cdot\left(\boldsymbol{I} + \frac{\boldsymbol{r}_{ij}\otimes\boldsymbol{r}_{ij}}{r_{ij}^2} + \frac{\sigma_i+\sigma_j}{r_{ij}^2}\left(\frac{1}{3}\boldsymbol{I} - \frac{\boldsymbol{r}_{ij}\otimes\boldsymbol{r}_{ij}}{r_{ij}^2}\right)\right), \tag{5.4}$$

The corrected friction tensor then reads [Carrasco and de la Torre, 1999b]:

$$\begin{aligned}\boldsymbol{\mu}_{ij}^{tt} =& \delta_{ij}(6\pi\eta_0\sigma_i)^{-1}\boldsymbol{I} + (1-\delta_{ij})(8\pi\eta_0 r_{ij}^{-1})(\boldsymbol{I}+\boldsymbol{P}_{ij}) \\ &+ (8\pi\eta_0 r_{ij}^{-3})(\sigma_i^2+\sigma_j^2)(\boldsymbol{I}-3\boldsymbol{P}_{ij}),\end{aligned} \tag{5.5}$$

where $\boldsymbol{P}_{ij} = \left(\boldsymbol{I} + \frac{\boldsymbol{r}\otimes\boldsymbol{r}}{r^2}\right)$. The mobility tensor for rotation, however, not correcting for the bead radii, is [Carrasco and de la Torre, 1999b]:

$$\begin{aligned}\boldsymbol{\mu}_{ij}^{rr} =& \delta_{ij}(8\pi\eta_0\sigma_i^3)^{-1}\boldsymbol{I} \\ &+ (1-\delta_{ij})(16\pi\eta_0 r_{ij}^3)^{-1}(3\boldsymbol{P}_{ij}-\boldsymbol{I}).\end{aligned} \tag{5.6}$$

Here, again, the first term is just the rotational mobility of the single bead and the second term accounts for the hydrodynamic interactions. In this formulation, there is still a correction for the bead radii missing. This correction consists of adding $6\eta_0 V_m\boldsymbol{I}$ to the diagonal components of the rotational friction tensor Ξ_O^{rr}, where V_m is the total volume of the rigid bead molecule [de la Torre and Rodes, 1983, Carrasco and de la Torre, 1999b].

The rotation-translation coupling is given by [Carrasco and de la Torre, 1999b]:

$$\boldsymbol{\mu}_{ij}^{rt} = (1 - \delta_{ij})(8\pi\eta_0 r_{ij}^2)^{-1}\boldsymbol{\epsilon}\hat{\boldsymbol{r}}_{ij}, \tag{5.7}$$

where $\boldsymbol{\epsilon}$ is the Levi-Civita tensor. $\boldsymbol{\mu}^{tt}, \boldsymbol{\mu}^{rr}, \boldsymbol{\mu}^{rt}$ describe the mobility of a multi-sphere system with hydrodynamic interactions. The above can be extended to account for rigid bead molecules [Carrasco and de la Torre, 1999b] as outlined in the next section.

5.1.2 The friction tensor for rigid bead molecules

Here, we closely follow [Carrasco and de la Torre, 1999b]. To get an expression for the friction tensor of a rigid bead molecule, we start by considering a system of N free spherical beads in a fluid with viscosity η_0. Each sphere laterally moves at some velocity $\boldsymbol{u}_i$ and rotates with some angular velocity $\boldsymbol{\omega}_i$. The spheres will experience a frictional force and torque $\boldsymbol{F}_i, \boldsymbol{T}_i$. In the non-inertial regime (Stokes regime), the relationship between the force/torque and the velocities is linear:

$$\boldsymbol{F}_i = \sum_{j=1}^{N} \boldsymbol{\xi}_{ij}^{tt} \cdot \boldsymbol{u}_j + \boldsymbol{\xi}_{ij}^{tr} \cdot \boldsymbol{\omega}_j \tag{5.8}$$

$$\boldsymbol{T}_i = \sum_{j=1}^{N} \boldsymbol{\xi}_{ij}^{rt} \cdot \boldsymbol{u}_j + \boldsymbol{\xi}_{ij}^{rr} \cdot \boldsymbol{\omega}_j. \tag{5.9}$$

The $\boldsymbol{\xi}_{ij}^{ab}, a, b \in \{t, r\}$ are the (3x3) friction matrices, connecting the amount of friction a particle i experiences due to the presence of particle j moving through the fluid at velocities $\boldsymbol{u}_j, \boldsymbol{\omega}_j$. We may rewrite Eqs. 5.8, 5.9 in matrix form as:

$$\begin{pmatrix} \boldsymbol{F} \\ \boldsymbol{T} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\xi}^{tt} & \boldsymbol{\xi}^{tr} \\ \boldsymbol{\xi}^{rt} & \boldsymbol{\xi}^{rr} \end{pmatrix} \begin{pmatrix} \boldsymbol{U} \\ \boldsymbol{W} \end{pmatrix}, \tag{5.10}$$

where $\boldsymbol{F} = (\boldsymbol{F}_1, ..., \boldsymbol{F}_N)^T$, $T = (\boldsymbol{T}_1, ..., \boldsymbol{T}_N)^T$ and $\boldsymbol{U} = (\boldsymbol{u}_1, ..., \boldsymbol{u}_N)^T$, $W = (\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_N)^T$. Here $\boldsymbol{\xi}^{ab}, a, b \in \{t, r\}$ are of dimension (3Nx3N), forming the friction supermatrix of dimension (6N,6N). The inverted friction supermatrix is the mobility supermatrix.

$$\begin{pmatrix} \boldsymbol{\mu}^{tt} & \boldsymbol{\mu}^{tr} \\ \boldsymbol{\mu}^{rt} & \boldsymbol{\mu}^{rr} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\xi}^{tt} & \boldsymbol{\xi}^{tr} \\ \boldsymbol{\xi}^{rt} & \boldsymbol{\xi}^{rr} \end{pmatrix}^{-1} \tag{5.11}$$

Next, we consider not a system of N free beads, but a rigid bead model, i.e., the beads are rigidly connected. Thereby, all beads move together with some translational velocity $\boldsymbol{u}_O$. Let the body's frame of reference lie at the center of diffusion of the bead model $\boldsymbol{r}_O$ and let $\boldsymbol{\omega}$ be the angular velocity of the rigid bead model. Then, in addition to the translational velocity of the molecule's center, each bead experiences a translation velocity due to the rotation $\boldsymbol{\omega} \times \boldsymbol{r}_i$, where $\boldsymbol{r}_i$ is the position vector from the molecules origin $\boldsymbol{r}_O$ (in the body frame of reference). Thereby, the total velocity is:

$$\boldsymbol{u}_i = \boldsymbol{u}_O + \boldsymbol{\omega} \times \boldsymbol{r}_i \tag{5.12}$$

The force that a single bead experiences due to the movement of all the other beads is:

$$\boldsymbol{F}_i = \sum_{j=1}^{N} \boldsymbol{\xi}_{ij}^{tt} \cdot (\boldsymbol{u}_O + \boldsymbol{\omega} \times \boldsymbol{r}_j) + \boldsymbol{\xi}_{ij}^{tr} \cdot \boldsymbol{\omega}, \tag{5.13}$$

and the torque that a single bead experiences due to the movement of all the other beads is:

$$\boldsymbol{T}_{P,i} = \sum_{j=1}^{N} \boldsymbol{\xi}_{ij}^{rt} \cdot (\boldsymbol{u}_O + \boldsymbol{\omega} \times \boldsymbol{r}_j) + \boldsymbol{\xi}_{ij}^{rr} \cdot \boldsymbol{\omega}. \tag{5.14}$$

From these expressions, we get the total force acting at the rigid body origin by summation over all beads:

$$\boldsymbol{F} = \sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{\xi}_{ij}^{tt} \cdot (\boldsymbol{u}_O + \boldsymbol{\omega} \times \boldsymbol{r}_j) + \boldsymbol{\xi}_{ij}^{tr} \cdot \boldsymbol{\omega} \tag{5.15}$$

For the total torque, however, we get an extra term. $\boldsymbol{T}_{P,i}$ is only the torque acting on bead i relative to it's center, i.e., the center of the sphere. Thereby, this only describes the amount of rotation bead i would experience around its center due to the movement of all the other beads. However, the force $\boldsymbol{F}_i$ acting on bead i due to the movement of the other beads also results in a torque with which bead i acts on the rigid bead models center $\boldsymbol{r}_O$:

$$\boldsymbol{r}_i \times \boldsymbol{F}_i = \boldsymbol{r}_i \times \Big(\sum_{j}^{N} \boldsymbol{\xi}_{ij}^{tt} (\boldsymbol{u}_O + \boldsymbol{\omega} \times \boldsymbol{r}_j) + \boldsymbol{\xi}_{ij}^{tr} \boldsymbol{\omega} \Big) \tag{5.16}$$

Thereby, the total torque acting on the rigid bead model's origin is:

$$\boldsymbol{T}_O = \sum_{i}^{N} \boldsymbol{T}_{P,i} + \boldsymbol{r}_i \times \boldsymbol{F}_i = \sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{\xi}_{ij}^{rt} \cdot (\boldsymbol{u}_O + \boldsymbol{\omega} \times \boldsymbol{r}_j) + \boldsymbol{\xi}_{ij}^{rr} \cdot \boldsymbol{\omega} + \boldsymbol{r}_i \times \Big(\boldsymbol{\xi_{ij}}^{tt} (\boldsymbol{u}_O + \boldsymbol{\omega} \times \boldsymbol{r}_j) + \boldsymbol{\xi}_{ij}^{tr} \omega \Big). \tag{5.17}$$

The above can be transformed into a general expression in simpler matrix form. For this, a little trick can be used to get rid of the cross product by turning $\boldsymbol{\omega} \times \boldsymbol{r}$ into the dot product $-\boldsymbol{A} \cdot \boldsymbol{\omega}$ (note: the sign changed, because of the anticommutativity of the cross product). After some rearranging, we end up with:

$$\boldsymbol{F} = \Big(\sum_{i=1}^{N}\sum_{j=1}^{N}\boldsymbol{\xi}_{ij}^{tt}\Big)\cdot\boldsymbol{u}_O + \Big(\sum_{i=1}^{N}\sum_{j=1}^{N} -\boldsymbol{\xi}_{ij}^{tt}\cdot\boldsymbol{A}_j + \boldsymbol{\xi}_{ij}^{tr}\Big)\cdot\boldsymbol{\omega} \tag{5.18}$$

$$\boldsymbol{T} = \Big(\sum_{i=1}^{N}\sum_{j=1}^{N}\boldsymbol{\xi}_{ij}^{rt} + A_i\boldsymbol{\xi}_{ij}^{tt}\Big)\cdot\boldsymbol{u}_O + \Big(\sum_{i=1}^{N}\sum_{j=1}^{N}\boldsymbol{\xi}_{ij}^{rt}\cdot\boldsymbol{A}_j + \boldsymbol{\xi}_{ij}^{rr} - A_i\boldsymbol{\xi}_{ij}^{tt}A_j + A_i\boldsymbol{\xi}_{ij}^{tr}\Big)\cdot\boldsymbol{\omega}. \tag{5.19}$$

If we now write this in matrix form, similar to the free bead example from above, we get:

$$\begin{pmatrix}\boldsymbol{F}\\ \boldsymbol{T}_O\end{pmatrix} = \begin{pmatrix}\boldsymbol{\Xi}^{tt} & \boldsymbol{\Xi}^{tr}\\ \boldsymbol{\Xi}^{rt} & \boldsymbol{\Xi}^{rr}\end{pmatrix}\begin{pmatrix}\boldsymbol{u}_O\\ \boldsymbol{\omega}\end{pmatrix}, \tag{5.20}$$

Where we call $\boldsymbol{\Xi}$ the friction tensor of the rigid bead molecule [Carrasco and de la Torre, 1999b] :

$$\begin{aligned}
\boldsymbol{\Xi}^{tt} &= \sum_{i=1}^{N}\sum_{j=1}^{N}\boldsymbol{\xi}_{ij}^{tt}\\
\boldsymbol{\Xi}_O^{tr} &= \sum_{i=1}^{N}\sum_{j=1}^{N}(-\boldsymbol{\xi}_{ij}^{tt}\cdot\boldsymbol{A}_j + \boldsymbol{\xi}_{ij}^{tr})\\
\boldsymbol{\Xi}_O^{rt} &= \sum_{i=1}^{N}\sum_{j=1}^{N}(\boldsymbol{A}_j\cdot\boldsymbol{\xi}_{ij}^{tt} + \boldsymbol{\xi}_{ij}^{rt})\\
\boldsymbol{\Xi}_O^{rr} &= \sum_{i=1}^{N}\sum_{j=1}^{N}(\boldsymbol{\xi}_{ij}^{rr} - \boldsymbol{\xi}_{ij}^{rt}\cdot\boldsymbol{A}_j + \boldsymbol{A}_i\cdot\boldsymbol{\xi}_{ij}^{tr} - \boldsymbol{A}_i\cdot\boldsymbol{\xi}_{ij}^{tt}\boldsymbol{A}_j)
\end{aligned} \tag{5.21}$$

The $\boldsymbol{\xi}$, are calculated from the inverse of the mobility supermatrix (Eq. 5.11).

A super Matrix $\boldsymbol{M} = [[\boldsymbol{M}_1, \boldsymbol{M}_2], [\boldsymbol{M}_3, \boldsymbol{M}_4]]$ is invertible, if both the diagonal blocks, $\boldsymbol{M}_1$ and $\boldsymbol{M}_4$ are invertible The inverse of a (2x2) supermatrix can be calculated by [Varadarajan, 2004], [Deligne and Morgan, 1996]:

$$\begin{aligned}
\boldsymbol{T}_1 &= (\boldsymbol{M}_1 - \boldsymbol{M}_2\boldsymbol{M}_4^{-1}\boldsymbol{M}_3)^{-1}\\
\boldsymbol{T}_2 &= -\boldsymbol{M}_1^{-1}\boldsymbol{M}_2(\boldsymbol{M}_4 - \boldsymbol{M}_3\boldsymbol{M}_1^{-1}\boldsymbol{M}_2)^{-1}\\
\boldsymbol{T}_3 &= -\boldsymbol{M}_4^{-1}\boldsymbol{M}_3(\boldsymbol{M}_1 - \boldsymbol{M}_2\boldsymbol{M}_4^{-1}\boldsymbol{M}_3)^{-1}\\
\boldsymbol{T}_4 &= (\boldsymbol{M}_4 - \boldsymbol{M}_3\boldsymbol{M}_1^{-1}\boldsymbol{M}_2)^{-1}
\end{aligned} \tag{5.22}$$

Bibliography

Jelena Baranovic. AMPA receptors in the synapse: Very little space and even less time. *Neuropharmacology*, 196:108711, sep 2021. doi: 10.1016/j.neuropharm.2021.108711.

Hiroki Tanaka, Naoyuki Miyazaki, Kyoko Matoba, Terukazu Nogi, Kenji Iwasaki, and Junichi Takagi. Higher-order architecture of cell adhesion mediated by polymorphic synaptic adhesion molecules neurexin and neuroligin. *Cell Reports*, 2(1):101–110, jul 2012. doi: 10.1016/j.celrep.2012.06.009.

Andreas Frick, Jeffrey Magee, and Daniel Johnston. LTP is accompanied by an enhanced local excitability of pyramidal neuron dendrites. *Nature Neuroscience*, 7(2):126–135, jan 2004. doi: 10.1038/nn1178.

John Lisman, Ryohei Yasuda, and Sridhar Raghavachari. Mechanisms of CaMKII action in long-term potentiation. *Nature Reviews Neuroscience*, 13(3):169–182, feb 2012. doi: 10.1038/nrn3192. URL `https://doi.org/10.1038/nrn3192`.

Seok-Jin R. Lee, Yasmin Escobedo-Lozoya, Erzsebet M. Szatmari, and Ryohei Yasuda. Activation of CaMKII in single dendritic spines during long-term potentiation. *Nature*, 458(7236):299–304, mar 2009. doi: 10.1038/nature07842. URL `https://doi.org/10.1038/nature07842`.

Patricio Opazo, Matthieu Sainlos, and Daniel Choquet. Regulation of AMPA receptor surface diffusion by PSD-95 slots. *Current Opinion in Neurobiology*, 22(3):453–460, jun 2012. doi: 10.1016/j.conb.2011.10.010. URL `https://doi.org/10.1016/j.conb.2011.10.010`.

A. C. Penn, C. L. Zhang, F. Georges, L. Royer, C. Breillat, E. Hosy, J. D. Petersen, Y. Humeau, and D. Choquet. Hippocampal LTP and contextual learning require surface diffusion of AMPA receptors. *Nature*, 549(7672):384–388, sep 2017. doi: 10.1038/nature23658. URL `https://doi.org/10.1038/nature23658`.

Richard L. Huganir and Roger A. Nicoll. AMPARs and synaptic plasticity: The last 25 years. *Neuron*, 80(3):704–717, oct 2013. doi: 10.1016/j.neuron.2013.10.025. URL `https://doi.org/10.1016/j.neuron.2013.10.025`.

Harold D. MacGillavry, Justin M. Kerr, and Thomas A. Blanpied. Lateral organization of the postsynaptic density. *Molecular and Cellular Neuroscience*, 48(4):321–331, dec 2011. doi: 10.1016/j.mcn.2011.09.001.

Yugo Fukazawa, Yoshito Saitoh, Fumiko Ozawa, Yasuhiko Ohta, Kensaku Mizuno, and Kaoru Inokuchi. Hippocampal LTP is accompanied by enhanced f-actin content within the dendritic spine that is essential for late LTP maintenance in vivo. *Neuron*, 38(3): 447–460, may 2003. doi: 10.1016/s0896-6273(03)00206-x.

Kenichi Okamoto, Miquel Bosch, and Yasunori Hayashi. The roles of CaMKII and f-actin in the structural plasticity of dendritic spines: A potential molecular identity of a synaptic tag? *Physiology*, 24(6):357–366, dec 2009. doi: 10.1152/physiol.00029.2009.

Masanori Matsuzaki, Naoki Honkura, Graham C. R. Ellis-Davies, and Haruo Kasai. Structural basis of long-term potentiation in single dendritic spines. *Nature*, 429(6993): 761–766, jun 2004. doi: 10.1038/nature02617. URL `https://doi.org/10.1038/nature02617`.

Mikyoung Park, Esther C. Penick, Jeffrey G. Edwards, Julie A. Kauer, and Michael D. Ehlers. Recycling endosomes supply AMPA receptors for LTP. *Science*, 305(5692): 1972–1975, sep 2004. doi: 10.1126/science.1102026.

Mikyoung Park, Jennifer M. Salgado, Linnaea Ostroff, Thomas D. Helton, Camenzind G. Robinson, Kristen M. Harris, and Michael D. Ehlers. Plasticity-induced growth of dendritic spines by exocytic trafficking from recycling endosomes. *Neuron*, 52(5):817–830, dec 2006. doi: 10.1016/j.neuron.2006.09.040.

M. A. Patterson, E. M. Szatmari, and R. Yasuda. AMPA receptors are exocytosed in stimulated spines and adjacent dendrites in a ras-ERK-dependent manner during long-term potentiation. *Proceedings of the National Academy of Sciences*, 107(36):15951–15956, aug 2010. doi: 10.1073/pnas.0913875107. URL `https://doi.org/10.1073/pnas.0913875107`.

Wickliffe C. Abraham and Joanna M. Williams. LTP maintenance and its protein synthesis-dependence. *Neurobiology of Learning and Memory*, 89(3):260–268, mar 2008. doi: 10.1016/j.nlm.2007.10.001.

Clive R Bramham. Local protein synthesis, actin dynamics, and LTP consolidation. *Current Opinion in Neurobiology*, 18(5):524–531, oct 2008. doi: 10.1016/j.conb.2008.09.013.

J. M. Kerr and T. A. Blanpied. Subsynaptic AMPA receptor distribution is acutely regulated by actin-driven reorganization of the postsynaptic density. *Journal of Neuroscience*, 32(2):658–673, jan 2012. doi: 10.1523/jneurosci.2927-11.2012.

Miquel Bosch, Jorge Castro, Takeo Saneyoshi, Hitomi Matsuno, Mriganka Sur, and Yasunori Hayashi. Structural and molecular remodeling of dendritic spine substructures during long-term potentiation. *Neuron*, 82(2):444–459, apr 2014. doi: 10.1016/j.neuron.2014.03.021. URL `https://doi.org/10.1016/j.neuron.2014.03.021`.

Daniel Meyer, Tobias Bonhoeffer, and Volker Scheuss. Balance and stability of synaptic structures during synaptic plasticity. *Neuron*, 82(2):430–443, apr 2014. doi: 10.1016/j.neuron.2014.02.031. URL `https://doi.org/10.1016/j.neuron.2014.02.031`.

Yoichi Araki, Menglong Zeng, Mingjie Zhang, and Richard L. Huganir. Rapid dispersion of SynGAP from synaptic spines triggers AMPA receptor insertion and spine enlargement during LTP. *Neuron*, 85(1):173–189, jan 2015. doi: 10.1016/j.neuron.2014.12.023.

Martin Hruska, Nathan Henderson, Sylvain J. Le Marchand, Haani Jafri, and Matthew B. Dalva. Synaptic nanomodules underlie the organization and plasticity of spine synapses. *Nature Neuroscience*, 21(5):671–682, apr 2018. doi: 10.1038/s41593-018-0138-9. URL `https://doi.org/10.1038/s41593-018-0138-9`.

Menglong Zeng, Yuan Shang, Yoichi Araki, Tingfeng Guo, Richard L. Huganir, and Mingjie Zhang. Phase transition in postsynaptic densities underlies formation of synaptic complexes and synaptic plasticity. *Cell*, 166(5):1163–1175.e12, aug 2016. doi: 10.1016/j.cell.2016.07.008.

Menglong Zeng, Xudong Chen, Dongshi Guan, Jia Xu, Haowei Wu, Penger Tong, and Mingjie Zhang. Reconstituted postsynaptic density as a molecular platform for understanding synapse formation and plasticity. *Cell*, 174(5):1172–1187.e16, aug 2018. doi: 10.1016/j.cell.2018.06.047.

Menglong Zeng, Javier Díaz-Alonso, Fei Ye, Xudong Chen, Jia Xu, Zeyang Ji, Roger A. Nicoll, and Mingjie Zhang. Phase separation-mediated TARP/MAGUK complex condensation and AMPA receptor synaptic transmission. *Neuron*, 104(3):529–543.e6, nov 2019. doi: 10.1016/j.neuron.2019.08.001.

Salman F. Banani, Hyun O. Lee, Anthony A. Hyman, and Michael K. Rosen. Biomolecular condensates: organizers of cellular biochemistry. *Nature Reviews Molecular Cell Biology*, 18(5):285–298, feb 2017. doi: 10.1038/nrm.2017.7.

Waja Wegner, Alexander C. Mott, Seth G. N. Grant, Heinz Steffens, and Katrin I. Willig. In vivo STED microscopy visualizes PSD95 sub-structures and morphological changes over several hours in the mouse visual cortex. *Scientific Reports*, 8(1), jan 2018. doi: 10.1038/s41598-017-18640-z.

Jiahua Lu, Junjie Qian, Zhentian Xu, Shengyong Yin, Lin Zhou, Shusen Zheng, and Wu Zhang. Emerging roles of liquid–liquid phase separation in cancer: From protein aggregation to immune-associated signaling. *Frontiers in Cell and Developmental Biology*, 9, jun 2021. doi: 10.3389/fcell.2021.631486.

Amandine Molliex, Jamshid Temirov, Jihun Lee, Maura Coughlin, Anderson P. Kanagaraj, Hong Joo Kim, Tanja Mittag, and J. Paul Taylor. Phase separation by low complexity domains promotes stress granule assembly and drives pathological fibrillization. *Cell*, 163(1):123–133, sep 2015. doi: 10.1016/j.cell.2015.09.015.

Susanne Wegmann, Bahareh Eftekharzadeh, Katharina Tepper, Katarzyna M Zoltowska, Rachel E Bennett, Simon Dujardin, Pawel R Laskowski, Danny MacKenzie, Tarun Kamath, Caitlin Commins, Charles Vanderburg, Allyson D Roe, Zhanyun Fan, Amandine M Molliex, Amayra Hernandez-Vega, Daniel Muller, Anthony A Hyman, Eckhard Mandelkow, J Paul Taylor, and Bradley T Hyman. Tau protein liquid–liquid phase separation can initiate tau aggregation. *The EMBO Journal*, 37(7), feb 2018. doi: 10.15252/embj.201798049.

Sang Hak Lee, Chaoyi Jin, En Cai, Pinghua Ge, Yuji Ishitsuka, Kai Wen Teng, Andre A de Thomaz, Duncan Nall, Murat Baday, Okunola Jeyifous, Daniel Demonte, Christopher M Dundas, Sheldon Park, Jary Y Delgado, William N Green, and Paul R Selvin. Super-resolution imaging of synaptic and extra-synaptic AMPA receptors with different-sized fluorescent probes. *eLife*, 6, jul 2017. doi: 10.7554/elife.27744. URL `https://doi.org/10.7554/elife.27744`.

Paul Smolen, Douglas A. Baxter, and John H. Byrne. Molecular constraints on synaptic tagging and maintenance of long-term potentiation: A predictive model. *PLOS Computational Biology*, 8(8):e1002620, aug 2012. doi: 10.1371/journal.pcbi.1002620.

M. E. Johnson, A. Chen, J. R. Faeder, P. Henning, I. I. Moraru, M. Meier-Schellersheim, R. F. Murphy, T. Prüstel, J. A. Theriot, and A. M. Uhrmacher. Quantifying the roles of space and stochasticity in computer simulations for cell biology and cellular biochemistry. *Molecular Biology of the Cell*, 32(2):186–210, jan 2021. doi: 10.1091/mbc.e20-08-0530.

Sebastian Kmiecik, Dominik Gront, Michal Kolinski, Lukasz Wieteska, Aleksandra Elzbieta Dawid, and Andrzej Kolinski. Coarse-grained protein models and their applications. *Chemical Reviews*, 116(14):7898–7936, jun 2016. doi: 10.1021/acs.chemrev.6b00163.

Gregory L Dignon, Wenwei Zheng, and Jeetain Mittal. Simulation methods for liquid–liquid phase separation of disordered proteins. *Current Opinion in Chemical Engineering*, 23:92–98, mar 2019. doi: 10.1016/j.coche.2019.03.004.

Jorge R. Espinosa, Adiran Garaizar, Carlos Vega, Daan Frenkel, and Rosana Collepardo-Guevara. Breakdown of the law of rectilinear diameter and related surprises in the liquid-vapor coexistence in systems of patchy particles. *The Journal of Chemical Physics*, 150(22):224510, jun 2019. doi: 10.1063/1.5098551.

Rex A. Kerr, Thomas M. Bartol, Boris Kaminsky, Markus Dittrich, Jen-Chien Jack Chang, Scott B. Baden, Terrence J. Sejnowski, and Joel R. Stiles. Fast monte carlo simulation methods for biological reaction-diffusion systems in solution and on surfaces. *SIAM Journal on Scientific Computing*, 30(6):3126–3149, jan 2008. doi: 10.1137/070692017.

Joshua A. Anderson, Jens Glaser, and Sharon C. Glotzer. HOOMD-blue: A python package for high-performance molecular dynamics and hard particle monte carlo simulations. *Computational Materials Science*, 173:109363, feb 2020. doi: 10.1016/j.commatsci.2019.109363.

Beatriz Carrasco and Jose Garcia de la Torre. Hydrodynamic properties of rigid particles: Comparison of different modeling and computational procedures. *Biophysical Journal*, 76(6):3044–3057, jun 1999a. doi: 10.1016/s0006-3495(99)77457-6.

Gregory L. Dignon, Wenwei Zheng, Young C. Kim, Robert B. Best, and Jeetain Mittal. Sequence determinants of protein phase behavior from a coarse-grained model. *PLOS Computational Biology*, 14(1):e1005941, jan 2018. doi: 10.1371/journal.pcbi.1005941.

Valentina Tozzini. Coarse-grained models for proteins. *Current Opinion in Structural Biology*, 15(2):144–150, apr 2005. doi: 10.1016/j.sbi.2005.02.005.

J Garcia de la Torre. Building hydrodynamic bead–shell models for rigid bioparticles of arbitrary shape. *Biophysical Chemistry*, 94(3):265–274, December 2001. doi: 10.1016/s0301-4622(01)00244-7.

Tihamér Geyer and Uwe Winter. An o(n2) approximation for hydrodynamic interactions in brownian dynamics simulations. *The Journal of Chemical Physics*, 130(11):114905, mar 2009. doi: 10.1063/1.3089668.

Maciej Długosz and Joanna Trylska. Diffusion in crowded biological environments: applications of brownian dynamics. *BMC Biophysics*, 4(1), mar 2011. doi: 10.1186/2046-1682-4-3.

Donald L. Ermak and J. A. McCammon. Brownian dynamics with hydrodynamic interactions. *The Journal of Chemical Physics*, 69(4):1352–1360, aug 1978. doi: 10.1063/1.436761.

Eric Dickinson, Stuart A. Allison, and J. Andrew McCammon. Brownian dynamics with rotation–translation coupling. *J. Chem. Soc., Faraday Trans. 2*, 81(4):591–601, 1985. doi: 10.1039/f29858100591.

R B Jones and P N Pusey. Dynamics of suspended colloidal spheres. *Annual Review of Physical Chemistry*, 42(1):137–169, oct 1991. doi: 10.1146/annurev.pc.42.100191.001033.

Ioana M. Ilie, Wim J. Briels, and Wouter K. den Otter. An elementary singularity-free rotational brownian dynamics algorithm for anisotropic particles. *The Journal of Chemical Physics*, 142(11):114103, mar 2015. doi: 10.1063/1.4914322.

David Baraff. Physically based modeling: Rigid body simulation. *SIGGRAPH Course Notes*, 2001. doi: 10.1145/97880.97881.

Ioana M. Ilie, Wouter K. den Otter, and Wim J. Briels. A coarse grained protein model with internal degrees of freedom. application to α-synuclein aggregation. *The Journal of Chemical Physics*, 144(8):085103, feb 2016. doi: 10.1063/1.4942115.

William Rowan Hamilton. II. on quaternions or on a new system of imaginaries in algebra. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 25(163):10–13, jul 1844. doi: 10.1080/14786444408644923.

Marc Niethammer, Raul San Jose Estepar, Sylvain Bouix, Martha Shenton, and Carl-Fredrik Westin. On diffusion tensor estimation. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, aug 2006. doi: 10.1109/iembs.2006.259826.

Guillaume Chevrot, Konrad Hinsen, and Gerald R. Kneller. Model-free simulation approach to molecular diffusion tensors. *The Journal of Chemical Physics*, 139(15):154110, oct 2013. doi: 10.1063/1.4823996.

V. Bloomfield, W. O. Dalton, and K. E. Van Holde. Frictional coefficients of multisubunit structures. i. theory. *Biopolymers*, 5(2):135–148, feb 1967. doi: 10.1002/bip.1967.360050202.

Jose Garcia De La Torre and Victor A. Bloomfield. Hydrodynamic properties of macromolecular complexes. i. translation. *Biopolymers*, 16(8):1747–1763, aug 1977a. doi: 10.1002/bip.1977.360160811.

B. Carrasco and J. Garcia de la Torre. Improved hydrodynamic interaction in macromolecular bead models. *The Journal of Chemical Physics*, 111(10):4817–4826, sep 1999b. doi: 10.1063/1.479743.

José García de la Torre and Vicente Rodes. Effects from bead size and hydrodynamic interactions on the translational and rotational coefficients of macromolecular bead models. *The Journal of Chemical Physics*, 79(5):2454–2460, sep 1983. doi: 10.1063/1.446054.

J. García de la Torre, G. del Rio Echenique, and A. Ortega. Improved calculation of rotational diffusion and intrinsic viscosity of bead models for macromolecules and nanoparticles. *The Journal of Physical Chemistry B*, 111(5):955–961, jan 2007. doi: 10.1021/jp0647941.

V. Varadarajan. *Supersymmetry for Mathematicians: An Introduction.* American Mathematical Society, jul 2004. doi: 10.1090/cln/011.

P Deligne and J Morgan. *Notes on supersymmetry following Bernstein. Quantum fields and strings; a course for mathematicians*, volume 1, pages 41–96. Amer. Math. Soc, Princeton, NJ; Providence, RI, 1996.

Steven Harvey and Jose Garcia de la Torre. Coordinate systems for modeling the hydrodynamic resistance and diffusion coefficients of irregularly shaped rigid macromolecules. *Macromolecules*, 13(4):960–964, jul 1980. doi: 10.1021/ma60076a037.

Moritz Hoffmann, Christoph Fröhner, and Frank Noé. ReaDDy 2: Fast and flexible software framework for interacting-particle reaction dynamics. *PLOS Computational Biology*, 15(2):e1006830, feb 2019. doi: 10.1371/journal.pcbi.1006830.

Peter Shirley, Michael Ashikhmin, and Steve Marschner. *Fundamentals of Computer Graphics.* A K Peters/CRC Press, jul 2009. doi: 10.1201/9781439865521.

L. Hu, G.M. Hu, Z.Q. Fang, and Y. Zhang. A new algorithm for contact detection between spherical particle and triangulated mesh boundary in discrete element method simulations. *International Journal for Numerical Methods in Engineering*, 94(8):787–804, mar 2013. doi: 10.1002/nme.4487.

David Eberly. *3D Game Engine Design.* Morgan Kaufmann Publishers, 2001.

John Amanatides and Andrew Woo. A fast voxel traversal algorithm for ray tracing. In *In Eurographics '87*, pages 3–10, 1987.

Christer Ericson. *Real-Time Collision Detection*. CRC Press, dec 2004. doi: 10.1201/b14581.

Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4):807–832, oct 2002. doi: 10.1145/571647.571648.

Radek Erban, Jonathan Chapman, and Philip Maini. A practical guide to stochastic simulations of reaction-diffusion processes, 2007.

Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, dec 1977. doi: 10.1021/j100540a008.

M Doi. Stochastic theory of diffusion-controlled reaction. *Journal of Physics A: Mathematical and General*, 9(9):1479–1495, sep 1976. doi: 10.1088/0305-4470/9/9/009.

Johannes Schöneberg and Frank Noé. ReaDDy - a software for particle-based reaction-diffusion dynamics in crowded cellular environments. *PLoS ONE*, 8(9):e74261, sep 2013. doi: 10.1371/journal.pone.0074261.

Konrad Polthier and Markus Schmies. Straightest geodesics on polyhedral surfaces. In *ACM SIGGRAPH 2006 Courses on - SIGGRAPH '06*. ACM Press, 2006. doi: 10.1145/1185657.1185664.

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. The heat method for distance computation. *Communications of the ACM*, 60(11):90–99, oct 2017. doi: 10.1145/3131280.

P. Trettner, D. Bommes, and L. Kobbelt. Geodesic distance computation via virtual source propagation. *Computer Graphics Forum*, 40(5):247–260, aug 2021. doi: 10.1111/cgf.14371.

J. R. Espinosa, C. Vega, and E. Sanz. The mold integration method for the calculation of the crystal-fluid interfacial free energy from simulations. *The Journal of Chemical Physics*, 141(13):134709, oct 2014. doi: 10.1063/1.4896621.

J. Jover, A. J. Haslam, A. Galindo, G. Jackson, and E. A. Müller. Pseudo hard-sphere potential for use in continuous molecular-dynamics simulation of spherical and chain molecules. *The Journal of Chemical Physics*, 137(14):144505, oct 2012. doi: 10.1063/1.4754275.

Jens Glaser, Xun Zha, Joshua A. Anderson, Sharon C. Glotzer, and Alex Travesset. Pressure in rigid body molecular dynamics. *Computational Materials Science*, 173: 109430, feb 2020. doi: 10.1016/j.commatsci.2019.109430.

Jorge R. Espinosa, Jerelle A. Joseph, Ignacio Sanchez-Burgos, Adiran Garaizar, Daan Frenkel, and Rosana Collepardo-Guevara. Liquid network connectivity regulates the stability and composition of biomolecular condensates with many components. *Proceedings of the National Academy of Sciences*, 117(24):13238–13247, jun 2020. doi: 10.1073/pnas.1917569117.

Erich A. Muller, Åsmund Ervik, and Andrés Mejía. A guide to computing interfacial properties of fluids from molecular simulations [article v1.0]. *Living Journal of Computational Molecular Science*, 2(1), 2020. doi: 10.33011/livecoms.2.1.21385.

H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. Molecular dynamics with coupling to an external bath. *The Journal of Chemical Physics*, 81(8):3684–3690, oct 1984. doi: 10.1063/1.448118.

Michael P. Allen and Dominic J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, nov 2017. doi: 10.1093/oso/9780198803195.001.0001.

Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 sketches on - SIGGRAPH '07*. ACM Press, 2007. doi: 10.1145/1278780.1278807.

M. Corsini, P. Cignoni, and R. Scopigno. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):914–924, jun 2012. doi: 10.1109/tvcg.2012.34.

V. Ogarko and S. Luding. A fast multilevel algorithm for contact detection of arbitrarily polydisperse objects. *Computer Physics Communications*, 183(4):931–936, apr 2012. doi: 10.1016/j.cpc.2011.12.019.

J. García de la Torre, Stephen E. Harding, and Beatriz Carrasco. Calculation of NMR relaxation, covolume, and scattering-related properties of bead models using the SOLPRO computer program. *European Biophysics Journal*, 28(2):119–132, jan 1999. doi: 10.1007/s002490050191.

José García de la Torre and Álvaro Ortega. HYDRO suite of computer programs for solution properties of rigid macromolecules. In *Encyclopedia of Biophysics*, pages 1002–1006. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-16712-6_291.

Radek Erban and S Jonathan Chapman. Stochastic modelling of reaction–diffusion processes: algorithms for bimolecular reactions. *Physical Biology*, 6(4):046001, aug 2009. doi: 10.1088/1478-3975/6/4/046001.

Marta Galanti, Duccio Fanelli, Sergey D. Traytak, and Francesco Piazza. Diffusion to capture and the concept of diffusive interactions. In *Chemical Kinetics*, pages 321–352. WORLD SCIENTIFIC (EUROPE), sep 2019. doi: 10.1142/9781786347015_0014.

John Crank. *The Mathematics of Diffusion*. Oxford University Press, USA, 1980. ISBN 9780198534112.

Howard C. Berg. *Random Walks in Biology*. Princeton University Press, jan 1984. doi: 10.1515/9781400820023.

Osman N. Yogurtcu and Margaret E. Johnson. Theory of bi-molecular association dynamics in 2d for accurate model and experimental parameterization of binding rates. *The Journal of Chemical Physics*, 143(8):084117, aug 2015. doi: 10.1063/1.4929390.

Andrij Trokhymchuk, Ivo Nezbeda, Jan Jirsák, and Douglas Henderson. Hard-sphere radial distribution function again. *The Journal of Chemical Physics*, 123(2):024501, jul 2005. doi: 10.1063/1.1979488.

Fu-Ming Tao, Yuhua Song, and E. A. Mason. Derivative of the hard-sphere radial distribution function at contact. *Physical Review A*, 46(12):8007–8008, dec 1992. doi: 10.1103/physreva.46.8007.

McDonald Ian R Hansen Jean-Pierre. *Theory of Simple Liquids*. Elsevier, 2013. doi: 10.1016/c2010-0-66723-x.

Steven S Andrews. Smoldyn: particle-based simulation with rule-based modeling, improved molecular interaction and a library interface. *Bioinformatics*, 33(5):710–717, dec 2016. doi: 10.1093/bioinformatics/btw700.

Matthew J. Varga, Yiben Fu, Spencer Loggia, Osman N. Yogurtcu, and Margaret E. Johnson. NERDSS: A nonequilibrium simulator for multibody self-assembly at the cellular scale. *Biophysical Journal*, 118(12):3026–3040, jun 2020. doi: 10.1016/j.bpj.2020.05.002.

Frank Noé, Alexandre Tkatchenko, Klaus-Robert Müller, and Cecilia Clementi. Machine learning for molecular simulation. *Annual Review of Physical Chemistry*, 71(1):361–390, apr 2020. doi: 10.1146/annurev-physchem-042018-052331.

Tihamér Geyer. Many-particle brownian and langevin dynamics simulations with the brownmove package. *BMC Biophysics*, 4(1), apr 2011. doi: 10.1186/2046-1682-4-7.

Ian Snook. *The Langevin and Generalised Langevin Approach to the Dynamics of Atomic, Polymeric and Colloidal Systems*. Elsevier, 2007. doi: 10.1016/b978-0-444-52129-3.x5000-7.

Uwe Winter and Tihamer Geyer. Coarse grained simulations of a small peptide: Effects of finite damping and hydrodynamic interactions. *The Journal of Chemical Physics*, 131(10):104102, 2009. doi: 10.1063/1.3216573.

Jose Garcia De La Torre and Victor A. Bloomfield. Hydrodynamics of macromolecular complexes. II. rotation. *Biopolymers*, 16(8):1765–1778, aug 1977b. doi: 10.1002/bip.1977.360160812.

J K G Dhont. *An Introduction to Dynamics of Colloids*. Elsevier, 1996. doi: 10.1016/s1383-7303(96)x8001-3.

C. W. Oseen. *Neuere methoden und ergebnisse in der Hydrodynamik*. Akadem. Verlagsges. Leipzig, 1927.