Self-assembly via anisotropic interactions

Modeling association kinetics of patchy particle systems and self-assembly induced by critical Casimir forces

Newton, A.C.

Publication date
2017
Document Version
Other version
License
Other

Link to publication

2 Computational methods

2.1 Ensemble averaging

In statistical thermodynamics a system in the canonical ensemble consists of N particles in a volume V at a temperature T, where a configuration of all particles is denoted as $\mathbf{x} = \{\mathbf{r_1}, \mathbf{r_2}, ...\mathbf{r_N}\}$ with $\mathbf{r}_i \in \mathbb{R}^3$ the coordinates of the ith particle. If the particles interact with a pair potential, $U(\mathbf{x})$, the equilibrium probability of a configuration is given by the Boltzmann weight:

$$P(\mathbf{x}) = \frac{\exp\left(-\beta U(\mathbf{x})\right)}{Z} \tag{2.1}$$

where β is $1/k_B T$ and the partition function $Z = \int d\mathbf{x} \exp\left(-\beta U(\mathbf{x})\right)$ normalizes the relative probability, where $\int d\mathbf{x}$ is a proper integral over every configuration $\mathbf{x}$. In computer simulation averages can be calculated based on the coordinates of particles. The ensemble average of an observable, $\mathcal{O}(\mathbf{x})$, is given by:

$$\langle \mathcal{O} \rangle = \frac{\int d\mathbf{x} \mathcal{O}(\mathbf{x}) \exp\left(-\beta U(\mathbf{x})\right)}{\int d\mathbf{x} \exp\left(-\beta U(\mathbf{x})\right)} \tag{2.2}$$

The explicit evaluation of the integral is typically not feasible for systems with a large number of particle due to the huge number of configurations possible. The same problem underlies the calculation of the free energy, from which all thermodynamic properties can be determined:

$$F = -k_B T \log Z \tag{2.3}$$

A simple Monte Carlo integration which calculates Eq. 2.2 by generating random configurations of particles, calculating the corresponding Boltzmann weight and adding to the running average is extremely inefficient in getting correct results, as it is extraordinarily more likely to generate configurations with a very low weight than not.

Instead of generating configurations with uniform probability, configurations could be generated proportional to their Boltzmann weight. As such, configurations with a high Boltzmann weight would simply occur more frequently during sampling than configurations with a low weight. In doing so, the statistical weight of each configuration is then taken into account in the generation probability and therefore ensemble averages can be calculated as unweighted averages:

$$\langle \mathcal{O} \rangle \approx \frac{1}{m} \sum_{k=1}^{m} \mathcal{O}(\mathbf{x}_k) \tag{2.4}$$

where on the right hand side the estimate of the integral as we can compute it in computer simulations is shown when generating m configurations and computing the value of $\mathcal{O}$ for configuration $\mathbf{x}_k$, which is now generated with a probability proportional to its Boltzmann weight.

Unfortunately, we can not use the same method to calculate F as it explicitly depends on the phase space volume. More advanced techniques need to be employed, such as thermodynamic integration, umbrella sampling, *etc.* [48].

Below we discuss two techniques which ensure the generation of new configurations with the proper relative Boltzmann weight. One is doing a Monte Carlo simulation according to the Metropolis algorithm and the other is using Molecular dynamics.

2.2 Monte Carlo

In the Metropolis Monte Carlo (MC) technique new configurations are not generated randomly, but are generated according to the relative Boltzmann weight between old and new configurations. In practice this means that starting from a given configuration of particles, $\mathbf{x}_o$, a trial move is performed which changes the system to a new configuration, $\mathbf{x}_n$. This trial move is rejected or accepted on the basis that it should preserve the equilibrium distribution once it is reached. Usually, an even stronger condition, detailed balance, is imposed:

$$p(\mathbf{x}_o)T(\mathbf{x}_o \to \mathbf{x}_n) = p(\mathbf{x}_n)T(\mathbf{x}_n \to \mathbf{x}_o) \tag{2.5}$$

This process is iterated m times, where after m iterations a sufficient number of configurations is collected. In doing so, we ensure that the most relevant configurations are sampled and from this ensemble of configurations the correct equilibrium properties are calculated according to Eq. 2.4.

Monte Carlo trial moves

Several Monte Carlo moves are used in the following chapters to sample configuration space [48]. As the particles are anisotropic in shape and/or potential, translational and rotational moves are used. In a **translation move** we translate a randomly chosen particle, i, by a random shift in the range of $[-\delta r_{max}; \delta r_{max}]$. The probability of accepting the move is:

$$P_{acc}(\mathbf{x}_o \to \mathbf{x}_n) = \min\left[1, \exp\left(-\beta \Delta E\right)\right] \tag{2.6}$$

where $\Delta E = E(n) - E(o)$ is the energy difference between the new and old configuration.

The **rotation move** is a bit more involved as we need a way to change the orientation of particles randomly. Rotations of objects could be done via the three Euler angles. However, when converting the Euler angles to rotation matrices, they would also require the use of trigonometric functions which are expensive and not numerically stable. Therefore, we make use of quaternions to represent the orientation. Quaternions are an extension to complex numbers first described by William Rowan Hamilton in 1843 and take the form:

$$\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k \tag{2.7}$$

where the complex numbers, i, j and k, follow the relation:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{2.8}$$

Importantly, if we only use unit length quaternions we can represent quaternions as rotations. In three dimensions, any rotation of an object can be seen as a single rotation around an unit axis $\hat{u}$ over an angle θ. Quaternions can encode this exact rotation as follows:

$$\mathbf{q} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ \hat{u}_x \sin\left(\frac{\theta}{2}\right) \\ \hat{u}_y \sin\left(\frac{\theta}{2}\right) \\ \hat{u}_z \sin\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{2.9}$$

In this case, we also make use of trigonometric functions. It is, however, less than using Euler angles. To actually rotate the rigid object we could convert the quaternion into a rotation matrix without the use of trigonometric equations as follows:

$$\mathbf{A} = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \tag{2.10}$$

and rotate a vector $\hat{\mathbf{p}}$ to $\hat{\mathbf{p}}'$ using $\hat{\mathbf{p}}' = \mathbf{A}\hat{\mathbf{p}}$. Alternatively, we can directly use the quaternion and rotate the vector $\hat{\mathbf{p}}$ by $\hat{\mathbf{p}}' = \hat{\mathbf{q}}\hat{\mathbf{p}}\hat{\mathbf{q}}^{-1}$, where $\hat{\mathbf{q}}^{-1}$ is the complex conjugate of $\hat{\mathbf{q}}$. Note that the vector $\hat{\mathbf{p}}$ first needs to be converted to a quaternion: $\hat{\mathbf{p}} = (0, \hat{p}_x, \hat{p}_y, \hat{p}_z)$. A random unit quaternion which uniformly samples a sphere can be generated by randomly choosing an angle θ in range of $[0; \delta\theta_{max}]$ and a random unit vector $\hat{\mathbf{u}}$. A rotational move is then performed by again choosing a random particle, choosing a random orientation and accepting the move according to Eq. 2.6.

To ensure that configurations decorrelate and the samples become independent, δr and $\delta\theta$ should be chosen carefully. However, as systems studied here have a low density, two versions of the rotational and translational moves are used, one with no limit to δr or $\delta\theta$ and one where the acceptance ratio is somewhere between 0.3 and 0.7.

These two moves only change the old configuration to the new one, by changing only a single particle. If strongly binding particles are considered clusters of different sizes are formed quickly. However, the subsequent relaxation of these clusters is limited by only using single particle moves. In order to get further relaxation, we also employ moves where not only entire clusters but also parts of clusters are translated and rotated. In this thesis the cluster move was used as explained in Ref. [90].

2.3 Molecular dynamics for colloidal particles

Molecular Dynamics is a simulation technique where uncle Isaac Newton's equations of motion are solved:

$$\vec{F} = m\ddot{x} \tag{2.11}$$

where $\vec{F} = -\partial U/\partial r$. Typically, this would entail the calculation of forces between each and every particle in the system. However, we make use of effective potentials where we have integrated out the degrees of freedom of the solvent particles. To describe Brownian motion which colloidal particles exhibit due to the now 'disappeared' solvent molecules, the Langevin equation can be used:

$$\vec{F} - \gamma\dot{x} + \sqrt{2\gamma k_B T}\zeta = m\ddot{x} \tag{2.12}$$

where the friction term, γ, is related to the diffusion constant via the Einstein relation, $D = \gamma k_B T$. The random collisions with the solvent are captured by the random force, ζ, which satisfies the following relations:

$$\begin{aligned} \langle\zeta(t)\rangle &= 0 \\ \langle\zeta(t)\zeta(t')\rangle &= \delta(t-t') \end{aligned} \tag{2.13}$$

which is referred to as white noise. Here we further assume that the inertia of particles is negligible relative to the friction and random forces, *i.e.* the over-damped limit, $m\ddot{x} \approx 0$. This leads to the following equation for over-damped Langevin dynamics or Brownian Dynamics (BD):

$$\beta D\vec{F} + \sqrt{2D}\zeta = \dot{x} \tag{2.14}$$

Eq. 2.14 can easily be integrated for isotropic particles, however, for patchy particles both the position and orientation need to be evolved over time. An elementary algorithm developed for rigid patchy particles that updates the orientation of particles by the use of quaternions is described in Ref. [91]. If BD is used, some potentials, especially when anisotropic are not possible or difficult to convert to forces. A list of expressions for the calculation of forces and torques is given in Ref. [92]. However, in this thesis Dynamic Monte Carlo is used to evolve particles in time.

Dynamic Monte Carlo

Dynamic Monte Carlo (DMC) simulations can be used to study dynamics of simple systems, because in the limit of very small displacements and when no unphysical moves are used, it is equivalent to Brownian dynamics (BD) [74, 93, 94]. DMC is, however, easier to implement in the case of patchy particles, especially in the case of step potentials, lattice models, 'reactive' patches or other exotic potentials from which the derivative of the energy is hard to calculate. A disadvantage of DMC is that collective motion is suppressed if anything but infinitesimal steps are being used. However, if collective motion is not important then this is no problem. Additionally, recent DMC methods also incorporate cluster moves in a dynamically correct way, which makes hierarchical assembly dynamics also possible, in exchange for a more complex algorithm [74, 93].

In molecular simulation we make use of reduced units, also for time. In BD a time-step is related to physical time via the diffusion constant at infinite dilution, D_T^0:

$$\Delta t = \frac{\Delta t^{BD} \sigma^2}{D_T^0} \tag{2.15}$$

where σ is the diameter of the particle. Therefore, to convert time-steps from a Brownian Dynamics simulation to physical time, the diffusion constant of the particles need to be known. In DMC there is no integration cycle with a well defined time-step, but MC cycles with a given set of displacements. In what follows, a MC cycle will consist of on average N translations and N rotations, N being the number of particles. We need to connect the displacements in a MC cycle to translational and rotational diffusion constants, and thus Brownian time. To achieve this, we should obtain the 'translational time', Δt_{trans}, and 'rotational time', Δt_{rot}, per MC cycle.

A translation move is performed by randomly choosing a shift in position in a range of $[-\delta r, \delta r]$ for each Cartesian axis. We start the derivation by using the relation between mean square displacement and the unit of time at infinite dilution, D_T^0:

$$\frac{\langle \Delta r^2 \rangle}{\Delta t} = 6 D_T^0 \tag{2.16}$$

The mean square displacement in a single MC cycle is given by:

$$\frac{\langle \Delta r^2 \rangle}{cycle} = \frac{3}{2\delta r} \int_{-\delta r}^{\delta r} r^2 a_{trans}(r) dr \approx \delta r^2 \bar{a}_{trans} \tag{2.17}$$

where $a_{trans}(\delta r)$ is the acceptance ratio for translation. Combining equation 2.16 and 2.17 we get the translational time corresponding to a MC cycle:

$$\frac{\Delta t_{trans}}{cycle} \approx \frac{\delta r^2 \bar{a}_{trans}}{6 D_T^0} \tag{2.18}$$

A rotational move is performed by choosing a random unit vector and a random angle between $[0, \delta\theta]$. Just like in Eq. 2.16 the requirement is:

$$\frac{\langle \Delta\theta^2 \rangle}{\Delta t} = 6D_R^0 \tag{2.19}$$

where θ is measured in radians. The mean squared rotational displacement (mrd) per cycle:

$$\frac{\langle \Delta\theta^2 \rangle}{cycle} = \frac{1}{\delta\theta} \int_0^{\delta\theta} a_{rot}(\theta)\theta^2 d\theta \approx \frac{\bar{a}_{rot}\delta\theta^2}{3} \tag{2.20}$$

The rotational time per cycle is then given by:

$$\frac{\Delta t_{rot}}{cycle} = \frac{\delta\theta^2 \bar{a}_{rot}}{18D_R^0} \tag{2.21}$$

The argument of Romano [94] to bootstrap the algorithm is by saying that in order for both 'translational and rotational time' to flow evenly, Δt_{trans} and Δt_{rot} per MC cycle have to be equal on average (plus having a high acceptance probability):

$$\frac{\Delta t_{trans}}{cycle} = \frac{\Delta t_{rot}}{cycle} \tag{2.22}$$

Using Eq. 2.21 and 2.18 and rearranging, we find that the ratio between δr and $\delta\theta$ should be:

$$\frac{\delta r}{\delta\theta} = \sigma \sqrt{\frac{\bar{a}_{rot}}{\bar{a}_{trans}} \frac{f_{SE}^2 D_T}{3D_R}} \tag{2.23}$$

In Ref. [94], it is shown that for DMC to correspond to BD, it is necessary that $\bar{a}_{rot} > 0.7$ and $\bar{a}_{trans} > 0.7$. For spherically shaped particles, the Stokes-Einstein relation between translational and rotational diffusion constants can be used, $D_R^0 = 3D_T^0/\sigma^2$. However, a constant, f_{SE}, is added to the ratio that can take non-Stokes-Einstein conditions into account:

During the simulation several diagnostics should be checked. For instance the ratio $\langle \Delta\theta^2 \rangle / \langle \Delta r^2 \rangle$ should be equal to the state dependent ratio D_r/D_t in the diffusive limit. In Ref. [94] it is shown that this ratio is the same for MD and DMC for several packing fractions.

If one wants to compare simulations to experiments, D_t^0 and D_r^0 have to be known a priori from experiments. In this way the reduced unit of time in simulations can be converted or the experimental values can be converted to Brownian units.

2.4 Structural and dynamical properties

Both MC and MD (DMC) can be used to obtain thermodynamic properties of the system by collecting a statistically relevant number of configurations for systems of anisotropic particles. Here, we discuss radial distribution functions which need special attention when considering anisotropic particles, and how to calculate the second virial coefficient for anisotropic particles which gives information about when particles tend to aggregate into clusters.

Radial distribution functions

The radial distribution function or pair correlation function, $g(r)$, gives the probability of observing a particle at a distance r from a reference particle. Therefore, it characterizes the local structure around a particle. The $g(r)$ can be measured experimentally via either neutron and X-ray scattering for molecular fluids or light-scattering for colloidal suspensions. Furthermore, $g(r)$ is important for many theories of simple liquids. For isotropic particles and in the low density limit, $g(r)$ is related to the pair potential $U(r)$ via a simple Boltzmann inversion: $\beta U(r) = -\log(g(r)$. Therefore, measuring $g(r)$ in experiment can give direct information on the microscopic interactions between particles.

However, for anisotropic particles this relation naturally does not hold. For this purpose, we adopt the approach explained in Eq. 1.2. Here a particle is considered to be a rigid framework made up of different spheres, comparable to how atoms form a molecule. A radial distribution is then defined where the trivial neighbor contributions, the contributions from the same 'molecule', are ignored. This is called the site-site radial distribution and will be further explained in chapter 3. The site-site radial distribution is typically used for molecular liquids, however, in chapter 3 we use this correlation function to characterize the structure of anisotropically shaped colloids.

Second virial coefficient

Particles interacting with a relatively short ranged isotropic interaction obey a generalised law of correspondence states (GLCS) which states that thermodynamic properties of these type of systems are insensitive to details of the potential, but are only dependent on the density and the second virial coefficient B_2. A condition for this phenomenon to arise is that each bond contributes independently and equally to the partition function which is usually true for a short ranged interaction, [51]. Conveniently, it is shown that GLCS is also true for particles with anisotropic interactions [51]. In general, when $B_2 < -1.5$ the system will aggregate [95]. Sometimes B_2 can be calculated analytically such as in Ref. [52]. However, it can also be calculated numerically. The general definition for the virial coefficient in three dimensions without an assumption about the potential is:

$$B_2 = \frac{1}{2}\int_0^{2\pi}\int_0^{\pi}\int_0^{r_c} d\phi d\theta dR R^2 \sin(\theta)\left[1 - e^{\beta U_{eff}(R)}\right] \tag{2.24}$$

However, for anisotropic particles we also need to integrate over every possible orientation:

$$\begin{aligned} B_2 = \frac{1}{2}\int_0^{2\pi}\int_0^{\pi}\int_0^{r_c} & d\phi d\theta dR R^2 \sin(\theta) \\ \left[1 - \int_0^{\pi}\int_0^{2\pi}\int_0^{\pi}\int_0^{2\pi} \right. & \left. d\alpha_1 d\beta_1 d\alpha_2 d\beta_2 \sin(\alpha_1)\sin(\alpha_2) e^{\beta U(\Omega_1,\Omega_2,R)} (4\pi)^{-2}\right] \end{aligned} \tag{2.25}$$

where the Euler angles α_i and β_i properly define every possible orientation of both particles. Note that symmetry arguments can reduce the number of integrations necessary, *e.g.* in the case of anisotropic but linear particles.

Diffusion constants of patchy particles

From MC we can only obtain static properties such as the quantities described above. However, from MD or DMC we can also calculate dynamic transport properties related to the diffusion of particles. When considering anisotropic particles, both translational and rotational diffusion constants are important quantities. Note that D_T and D_R are input parameters in Brownian dynamics, but we need to check whether the implemented dynamics is consistent.

Translational diffusion constant

To calculate the translational diffusion constant, the mean square displacement (msd) can be measured. The msd grows in time without bound as follows:

$$\left\langle \Delta r^2(\Delta t)\right\rangle = \left\langle [r(t+\Delta t) - r(t)]^2 \right\rangle \tag{2.26}$$

where $r(t)$ is the position of the particle at time t. If we take the limit:

$$\lim_{\Delta t\to\infty} \left\langle \Delta r^2(\Delta t)\right\rangle = 2dD_T\Delta t \tag{2.27}$$

where d is the number of dimensions. It follows that the translational self-diffusion constant D_T can be easily calculated by taking the slope of the msd in the long time limit. Note that for particles with anisotropic shape, each cartesian component in the reference frame can show different diffusion constants which can be taken into account by using a diffusion tensor with differing diagonal components.

D_T can in principle also be calculated from the velocity autocorrelation function (vacf) using linear response theory via the Green-Kubo relation:

$$D_T = \int_0^\infty d\tau \left\langle v_x(\tau)v_x(0)\right\rangle \tag{2.28}$$

However, as particles considered in this thesis have no inertia and thus no velocity because they are considered to be completely over-damped (Brownian), the vacf can not be calculated.

Rotational diffusion constant

The rotational diffusion constant, D_R, can be calculated by measuring the mean square angular displacement, msad, which quantifies the rotational motion of the unit vector $\hat{u}$ that defines the orientation of the particle. Rotation of a particle in a time difference Δt can be envisioned as the rotation of $\hat{u}$ by an angle $\theta_{rot} = \arccos\left(\hat{u}(t)\cdot\hat{u}(t+\Delta t)\right)$ around a rotation vector defined as the cross product $u_{rot} = \hat{u}(t)\times\hat{u}(t+\Delta t)$.

If simply the angle is used to construct the msad, the msad will be bound as particles will eventually have rotated full circle towards the initial orientation $\hat{u}(0)$, which would seem as it has returned to the initial orientation. This would limit the time over which one can measure the msad. To measure a msad which is unbound just like the msd described above, a more involved calculation is necessary. A rotational displacement is defined, $\delta\phi(t)$, whose magnitude is given by θ_{rot} and direction is given by u_{rot} [96]. The total angular displacement is given by:

$$\phi(t) = \int_0^t \delta\phi(t')dt' \tag{2.29}$$

The unbound msad is then given by:

$$\left\langle \Delta\phi^2(\Delta t) \right\rangle = \left\langle [\phi(t+\Delta t) - \phi(t)]^2 \right\rangle \tag{2.30}$$

The long time limit is as follows:

$$\lim_{\Delta t \to \infty} \left\langle \Delta\phi^2(\Delta t) \right\rangle = 4D_R \Delta t \tag{2.31}$$

from which it follows similarly to D_T, that D_R can be calculated by taking the slope of the msad even when measured over long times.

Digression :
Note that in Eq. 2.19 $d = 3$, whereas in Eq. 2.31 $d = 2$. It depends on how rotation is actually measured. Rotation can be measured by the diffusion of a rotating unit vector $\hat{u}$ which is described by a two-dimensional diffusion on a spherical plane and it leads to a pre-factor of 4 as in Eq. 2.31. However, if rotation is measured by the angular displacement we need three dimensions as we have three independent Euler angles to consider, which leads to a pre-factor of 6 as in Eq. 2.19.

2.5 Simulating self-assembly rare events

Rare events are transitions between stable regions of phase space that are extremely infrequent due to the presence of large free energy barriers. Typical examples in chemical physics are the folding of proteins, homogeneous nucleation or chemical reactions. In self-assembly of patchy particles, transitions between stable states where particles are clustered can also be viewed as rare events. Due to the reduction of binding volume for patchy particles relative to isotropic particles, high binding energies are necessary for stable clusters, which causes unbinding events to be rare. Moreover, due to the small binding volume, the binding transition of patchy particles also encounter significant entropic barriers.

Although the ground-state is known to be an ordered structure, a system can still become kinetically trapped in states where particles are clustered into disordered aggregates. Therefore, knowing how the dynamics and the interactions of particles affect the association or dissociation rate constants and transition between intermediate states becomes important, besides when the thermodynamic phase behavior is known.

However, obtaining accurate (un)binding rate constants is often difficult in simulations due to the same high free energy barriers that naturally arise in strongly bound particles. When brute-force simulations are used, a majority of the simulation time is wasted inside the stable state which does not give information about the transitions. To alleviate this problem of separation of timescales between simulating the particle dynamics and the macroscopic binding rate constants, advanced path sampling techniques have been developed that bias the sampling of reactive pathways.

In this section we will discuss what path sampling method is used in this thesis and technical details concerning the calculation of rate constants and path densities.

From TPS to SRTIS

A method developed to generate reactive pathways without biasing the dynamics such as umbrella sampling or metadynamics, is Transition Path Sampling (TPS) [88, 97–99]. In TPS reactive pathways are generated through a Monte Carlo scheme which samples the reactive path ensemble between two or multiple states. Similarly to conventional Monte Carlo, where configurations are visited in proportion to the Boltzmann distribution, TPS samples reactive pathways according to their proper weight. The advantage of TPS and related methods is that the entire unbiased reactive path ensemble can be sampled, from which the mechanism of otherwise elusive reactive pathways can be analyzed in a statistically meaningful way. Moreover, a reaction coordinate is not needed, only a definition of states, *i.e.* the basins of attraction of a reactant and a product. In principle, also the reaction rate constant can be calculated via TPS.

After the development of TPS, many advances has been made in the field of path sampling methods. Transition Interface Sampling (TIS) was introduced as a more efficient approach to calculate the rate constant between states. It does so by defining interfaces around states through an order parameter that gives an indication on the progress of the reaction [100]. The rate constant calculation is based on the effective positive flux through these dividing surfaces. In TIS, path ensembles are simulated for each interface separately. These paths no longer only need to be reactive, but can also consist of $A \to A$ paths. Moreover, in the RETIS approach the concept of replica exchange was introduced, by allowing neighbouring interfaces to swap pathways if both pathways obey the conditions imposed by the interfaces [101–103]. This greatly increased the convergence of the path simulations as pathways can decorrelate quicker. A disadvantage of RETIS however, is that the number of interfaces can easily increase to an enormous amount when for instance multiple states need to be defined, limiting the practical implementation as all interfaces need to be sampled simultaneously. Recently, the Single Replica Transition Interface Sampling (SRTIS) method was developed by Du and Bolhuis [104] where, similar to simulated tempering, only a single replica walks through all the interfaces set by the multiple state TIS network, in contrast to RETIS where each interface is sampled individually and simultaneously. As such, the large increase in the number of interfaces is not as much of a problem, as there is only one single replica in memory. In the following chapters, the SRTIS method is used to sample path space for different patchy particle systems.

Paths, interfaces and indicator functions

Before continuing on to explain what path moves are actually used and how the rate constant is calculated, it is convenient to introduce how a path is conceptualised in path sampling.

A path is thought of as a discretized sequence of of configurations, $\mathbf{x}^L = [x_0, x_1....x_L]$, where x_i are phase space points usually defined by coordinates $\mathbf{r}$ and momenta $\mathbf{p}$ of the N-particle system. However, in following chapters the momenta are left out due to the use of DMC. Each configuration x_k is separated by a time Δt, such that the total time duration of the path $\mathcal{T} = L\Delta t$.

An unbiased trajectory $\mathbf{x}^L$ has probability:

$$P[\mathbf{x}^L] = \rho(x_0) \prod_{i=0}^{L-1} p\left(x_i \to x_{i+1}\right) \tag{2.32}$$

where $\rho(x_0)$ is the steady state distribution for the first configuration of the path, $p\left(x_i \to x_{i+1}\right)$ is the Markov probability to move from x_i to x_{i+1}. Akin to the Boltzmann probability for configurations, also the path probability is normalized by a partition function:

$$Z = \int \mathcal{D}\mathbf{x}^L P[\mathbf{x}^L] \tag{2.33}$$

where $\mathcal{D}\mathbf{x}^L$ defines a suitable integral over all paths.

In the multiple state TIS framework we define a set $\mathbf{S}$ of M states. Each state I has its own interface set $\mathbf{\Lambda}_I$ of $m+1$ interfaces, Λ_I^i, defined through an order parameter $\lambda_I(x)$, where the boundary of interface i of state I is set by λ_I^i. The boundary of a state is denoted as λ_I^0. No interfaces of the same set are allowed to intersect. However, interfaces that belong to different states of course are allowed to overlap.

In order to define the path probability for TIS path ensembles it is convenient to introduce indicator functions which define when a path belongs to the ensemble of Λ_I^i:

$$h_I^i[\mathbf{x}^L] = \begin{cases} 1 & \text{if } x_0 \in I \wedge x_L \in S \wedge \\ & \forall\{j|0<j<L\} : x_j \notin S \wedge \\ & \exists\{j|0<j<L\} : \lambda(x_j) > \lambda_I^i \\ 0 & \text{otherwise} \end{cases} \tag{2.34}$$

where the third line indicates that there has to exists a slice x_j which crosses interface Λ_I^i. We then define the path probability to observe a path $\mathbf{x}^L$ in replica i as:

$$\mathcal{P}_{\Lambda_I^i} = h_I^i[\mathbf{x}^L] P[\mathbf{x}^L]/Z \tag{2.35}$$

In this thesis states are defined through the topology of the structure formed, number of bonds and energy of the system. Interfaces are defined based on the energy of the system which ensures that paths are biased both radially and orientationally away from stable states, which avoids hysteresis.

Path MC moves

The main path sampling move is the **shooting move**, where usually from a (randomly) chosen time slice of the current path, a new path is generated. Due to the stochastic nature of the dynamics the newly generated path will sample a different part of path space. Here a time slice is not chosen randomly, but we use constrained one-way shooting from the current interface, λ_I^i where the new path is always accepted as long as it ends up in a stable state [103]. The acceptance probability of the shooting move is:

$$P_{acc}\left[\mathbf{x}^L(o) \to \mathbf{x}^L(n)\right] = h_I^i[\mathbf{x}^L] \tag{2.36}$$

Note that the usual TIS length criterion $\min\left[1, \frac{L^0}{L^n}\right]$ does not appear in Eq. 2.36. This term should be used when the generation probability of a shooting point is uniform along the path, because then the generation probability is dependent on the length of the current and new path.

In order to walk through replicas, a **replica swap** is used, where an attempt is made to change the sampling from the current interface to a neighbouring interface which is only possible when the path crosses both interfaces. Naturally, as stable states are strong attractors, this would lead to oversampling near the state and undersampling of any pathways that lead far away from state I. Therefore, we employ a Wang Landau bias on the replica swapping probability to enforce uniform sampling between replicas [104, 105]:

$$P_{acc}(\mathbf{x}^L; \lambda_I^i \to \lambda_I^j) = h_I^j[\mathbf{x}^L] \min\left[1, \frac{g(\lambda_I^i)}{g(\lambda_I^j)}\right] \tag{2.37}$$

where $g(\lambda_I^i)$ is the density of paths, which is updated upon visiting Λ_I^i via a scale-factor f_{WL} which is set to an arbitrary value at the beginning of the simulation. When during the simulation all replicas have been sampled uniformly within a certain threshold, f_{WL} is halved until it has converged to a sufficiently low number [104, 105]. For a converged TIS simulation, $g(\lambda_I)$ should become proportional to the crossing probability, because to obtain uniform sampling over each interface, the acceptance in Eq. 2.37 should be biased with the ratio of the naturally occurring probability for pathways, which is the ratio of crossing probabilities between the interfaces to be swapped (see also section 2.5).

To allow exchange between paths starting from different states, a **state-swap** move is employed. The current initial state is changed to a different state and the direction of the path is reversed which is only possible when the path connects

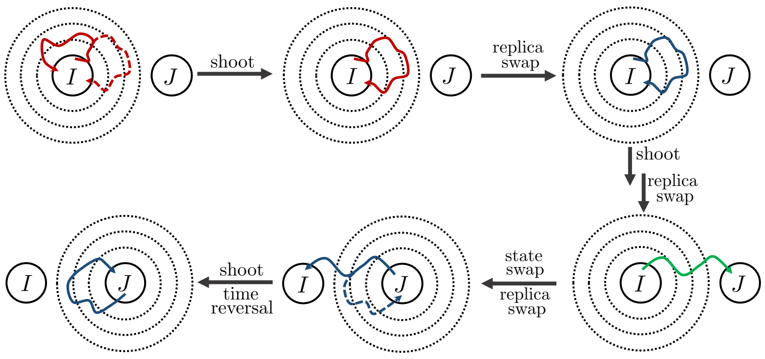

Figure 2.1: A schematic demonstrating the SRTIS path moves. From top left to top center: a shooting move is attempted which shoots the current path from the current interface Λ_I^1 and eventually also crosses interface Λ_I^2. From top center to top right: as the current path now also crosses Λ_I^2, we can swap the current replica Λ_I^1 with Λ_I^2 and start collecting pathways for this path ensemble. From top right to bottom right: after a shooting attempt and a replica swap, a reactive path is generated which connects states I and J. From bottom right to bottom center: a state swap is performed which reverses the direction of the path. Also the current interface set is changed from $\mathbf{\Lambda_I}$ to $\mathbf{\Lambda_J}$. From here a replica swap can be performed according to the Wang-Landau bias, $\frac{g(\lambda_J^3)}{g(\lambda_J^2)}$. From bottom center to bottom left: again a shooting move is performed, and subsequently a time-reversal which changes the direction of the path.

state $I \to J$ with $J \neq I$. The acceptance probability for the state swap is:

$$P_{acc}(\mathbf{x}^L; \lambda_I^i \to \lambda_J^j) = h_J^j[\overleftarrow{\mathbf{x}}^L] \min\left[1, \mathcal{N}\frac{g(\lambda_I^i)}{g(\lambda_J^j)}\right] \tag{2.38}$$

where $\overleftarrow{\mathbf{x}}^L$ is denoted to indicate $\mathbf{x}^L$ in reverse order, and $\mathcal{N}$ is unity if a state swap is only performed between the same two interfaces λ_I^k and λ_J^k, and it is the fraction of the number of replicas of states, $\frac{m_J}{m_I}$ if all interfaces between states are allowed to swap. Although not commonly used as in most studies all states have the same number of interfaces, especially when states are nested within interfaces, it can be advantageous to allow all interfaces to state swap. Note that $g(\lambda_I^i)$ is also used for the state swap. This ensures that each replica, across all states is sampled uniformly.

To randomize within the state, we also employ the so-called **minus move** [102]. In the minus move, a path is generated that begins at the interface boundary of Λ_I^1 and is subsequently evolved within the stable state I instead of away from it. First of all, this makes sure different exits out of state I are sampled. Second, these paths can be used to calculate the flux out of state I as shown in Eq. 2.41.

Moreover, in order to achieve further decorrelation between pathways, we also use the **time-reversal** move where the order of the path is reversed. Acceptance probability for this move is simple due to the fact that microscopic reversibility ensures that both directions of paths are equally probable:

$$P_{acc}(\lambda_I^i; \mathbf{x}^L \rightarrow \overleftarrow{\mathbf{x}}^L]) = h_I^i[\overleftarrow{\mathbf{x}}^L] \tag{2.39}$$

which indicates that only $I \rightarrow I$ paths are accepted. In Fig. 2.1 a graphical summary is presented of all the moves. Note that other path moves are also possible, but these moves are able to efficiently sample the entire path ensemble for self-assembling systems.

Digression:
Although TIS as described above and used in the following chapters is a very efficient method to obtain the entire path ensemble between all states, it might not be the best method in general for studying self-assembly. The main reason is that self-assembly intrinsically consist of two different types of processes. One is association where the energy decreases due to bond formation and the other is dissociation where energy increases due to bond breaking. In the TIS setup described here, we only use one set of interfaces for each state defined by the energy of the system that can only bias the generation of pathways in one direction of the reaction coordinate, up or down in energy. Therefore, it is not possible to effectively bias the generation of pathways corresponding to both types of processes. An obvious solution to this setback, is that each state gets two or more sets of interfaces, each corresponding to a distinct process. Multiple Interface Set TIS (MISTIS) is designed to handle different sets of interfaces per state [106]. Specifically for self-

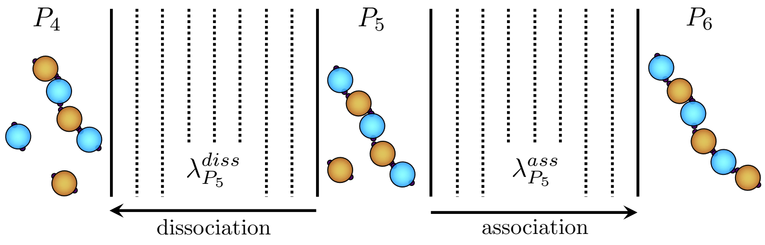

Figure 2.2: An example system where MISTIS might be better suited for than SRTIS with a single set of interfaces. Solid lines are the boundaries of stable state definitions and dashed lines are interfaces. Here a system, residing in a state where the largest cluster is made of 5 particles, P_5, can transition via dissociation towards, P_4 or via association towards P_6. Having two sets of interfaces, one that generates pathways for dissociation and one that generates pathways for association could work more efficiently.

assembly transitions, two different interfaces can be defined. One for association, $\lambda_A^{ass} = \min[r_{cut} - d_{ij}]$ where d_{ij} are all distances between possible binding sites and r_{cut} is the distance where two binding sites are considered to be completely unbound, and where the min function returns the minimum of all pairs. Another interface set would be for dissociation, $\lambda_A^{diss} = \min[d_{ij}]$. Now, in contrast to the TIS setup described above, when the system resides in an intermediate state, both associating and dissociating pathways can be generated, see Fig. 2.2. Note that the rate constant calculation is a bit more involved than TIS with one interface set [106].

Rate equation

One of the main reasons to perform a TIS simulation, is that one is interested in calculating rate constants. From a TIS calculation, the rate constant can be obtained via [100]:

$$k_{IJ} = \phi_I P(\lambda_J^0|\lambda_I^1) \tag{2.40}$$

where the flux, ϕ_I, is defined as the number of positive crossings per unit of time out of the stable state I through the first interface λ_I^1 and $P(\lambda_J^0|\lambda_I^1)$ is the crossing probability from the first interface of state I to the first interface of state J, which can be factorized as $P(\lambda_J^0|\lambda_I^1) = P(\lambda_I^m|\lambda_I^1)P(\lambda_J^0|\lambda_I^m)$. The flux is calculated on the fly as follows:

$$\phi_I = \left(\langle\tau^0\rangle + \langle\tau^1\rangle\right)^{-1} \tag{2.41}$$

where $\langle\tau^0\rangle$ is the average path-length in the minus interface and $\langle\tau^1\rangle$ is the average path-length from the first interface. Typically, $P(\lambda_J^0|\lambda_I^m)$ can also be calculated on the fly from the outermost interface:

$$P(\lambda_J^0|\lambda_I^m) = \frac{n_{IJ}(\lambda_I^m)}{\sum_J n_{IJ}(\lambda_I^m)} \tag{2.42}$$

where $n_{IJ}(\lambda_I^m)$ is the number of pathways from I to J for the outermost interface λ_I^m and the sum is over all states. Note that both ϕ_I and $P(\lambda_{0J}|\lambda_{mI})$ could also efficiently be calculated from a brute force MD run. The crossing probabilities, $P(\lambda_{mI}|\lambda_{1I})$, are usually very small in the case of rare events and therefore, difficult to obtain via brute-force MD. However, because in TIS crossing histograms are accumulated for each interface, we can obtain good statistics on the crossing probability by performing histogram reweighting (WHAM) on the individual crossing probabilities for every interface which is described next.

Crossing probability and WHAM

Crossing histograms can be constructed by monitoring the maximum order parameter value a path has reached away from state A as follows:

$$P_A(\lambda|\lambda^i) = \int \mathcal{D}\mathbf{x}^L \mathcal{P}_{\Lambda_A^i}[\mathbf{x}^L]\mathcal{H}(\lambda_{max}[\mathbf{x}^L] - \lambda) \tag{2.43}$$

where $\mathcal{P}_{\Lambda_A^i}[\mathbf{x}^L]$ is the probability of a path, $\mathbf{x}^L$, when sampling pathways in replica i, and $\mathcal{H}(x)$ is the Heaviside step function.

In order to obtain the full unbiased crossing histogram, each individual crossing histogram obtained by sampling pathways for each replica Λ_A^i has to be combined and given its proper weight. This is done via WHAM and the combined crossing probability is defined as follows:

$$P_A(\lambda|\lambda_A^1) = \sum_{i=1}^{n=1} \bar{w}_A^i \mathcal{H}(\lambda_A^{i+1} - \lambda)\mathcal{H}(\lambda - \lambda_A^i) \sum_{j=1}^{i} P_A(\lambda|\lambda_A^j) \tag{2.44}$$

where $\bar{w}_A^i$ are given by:

$$\bar{w}_A^i = \frac{1}{\sum_{j=1}^{i}(w_A^j)^{-1}} \tag{2.45}$$

where w_A^j are the optimized WHAM weights for each interface crossing histogram.
Digression :
Now, Eq. 2.45 looks at first sight a bit magical. Why wouldn't each interface be given the weight w_A^i which comes directly from WHAM? It is basically constructed such, that it avoids over counting of paths.

However, the following simple example might give some more comfort. Imagine a state A with three interfaces. Imagine the WHAM weights for each interface are $w_A^1 = 1.0$, $w_A^2 = 0.5$ and $w_A^3 = 0.1$, which according to Eq. 2.45 would lead to $\bar{w}_A^1 = 1.0$, $\bar{w}_A^2 = \frac{1}{3}$ and $\bar{w}_A^3 = \frac{1}{13}$. If we perform a TIS simulation for this system and generate a thousand pathways for each replica, we could make the following histogram which gives the total number of pathways starting from replica Λ^i (row) and has crossed replica Λ^j (column) with $j \geq i$, $n_{IJ}(\lambda)$, which would resemble the continuous crossing histogram as in Eq. 2.43:

$n_{IJ}(\lambda)$	Λ^1	Λ^2	Λ^3
Λ^1	1000	500	100
Λ^2		1000	200
Λ^3			1000
Total	1000	1500	1300

As the distribution of pathways from Λ^1 represents the distribution of the natural ensemble, we want the total number of pathways given in the last row after reweighting to match the distribution of Λ^1. If we multiply the weights $\bar{w}_A^i$ with $n_{IJ}(\lambda)$ we obtain precisely that!

However, in the above histogram we did not give a path a particular weight as a path can belong to all three replicas. It seems more intuitive and clean to assign a path one certain weight, *i.e.* it belongs to only one replica. To assign pathways only one certain weight we histogram the pathways differently (which will also be used in the Reweighted Path Ensemble below). We will create a histogram where a path is only assigned to a replica which it has maximally crossed, $n_{IJ}(\lambda^{max})$:

$n_{IJ}(\lambda^{max})$	Λ^1	Λ^2	Λ^3
Λ^1	500	400	100
Λ^2		800	200
Λ^3			1000
Total	500	1200	1300

Mindbogglingly, the same weights $\bar{w}_A^i$ multiplied with $\sum_{I\in S} n_{IJ}(\lambda^{max})$ also returns the distribution of Λ^1! Note that this reweighting does not depend on the fact that we have uniformly sampled across each replica, as w_A^i would scale accordingly, which becomes important for SRTIS. A more formal proof is given in Ref. [107].

Reweighted path ensemble

In SRTIS we obtain the Wang-Landau biased path ensemble. To reweight the path ensemble to match it to the natural path ensemble, *i.e.* give each sampled pathway its proper weight, we can use the crossing probabilities obtained from WHAM [107]. The reweighted path ensemble (RPE) can be calculated as follows:

$$\mathcal{P}[\mathbf{x}^L] = \sum_{I\in S} c_I \left[w_I^1 \mathcal{P}^-_{\Lambda_I^1}[\mathbf{x}^L] + \sum_{j=1}^{n-1} \mathcal{P}_{\Lambda_I^j}[\mathbf{x}^L] W_I^j[\mathbf{x}^L] \right], \tag{2.46}$$

where $\mathcal{P}^-_{\Lambda_I^1}[\mathbf{x}^L]$ is the probability to sample path $\mathbf{x}^L$ in the minus interface, $\mathcal{P}_{\Lambda_I^j}[\mathbf{x}^L]$ is the probability to obtain path $\mathbf{x}^L$ while sampling Λ_I^j, $W_I^j[\mathbf{x}^L] = \sum_{i=1}^{n-1} \bar{w}_I^i \mathcal{H}_I^i[\mathbf{x}^L]$ where $\mathcal{H}_I^i[\mathbf{x}^L]$ is used to select the correct weight $\bar{w}_I^i$ for a path that has its maximum λ between interface j and $j+1$: $\mathcal{H}_I^i[\mathbf{x}^L] = \mathcal{H}(\lambda_{max}[\mathbf{x}^L] - \lambda_i)\mathcal{H}(\lambda_{i+1} - \lambda_{max}[\mathbf{x}^L])$.

In SRTIS we strive to sample all interfaces across all states uniformly, which is also done via the WL bias, $g(\lambda_I^i)$ in the state swap move. To get the proper weight between states, we use the constants c_I. If we would not use any bias for the state swap, and only allow swaps between replicas Λ_I^m and Λ_J^m, we would obtain the natural ensemble between these two interfaces, meaning these two interfaces should have the same weight. We can achieve this by scaling each weight $w_I^{i,nat} = w_I^i / w_I^m$, which sets the weight of the outermost interface of each state, w_I^m, to the same value. If no bias is used in state swap, c_I would not be necessary. However, we do. For this reason, we should after scaling all weights with w_I^m, also scale with $g(\lambda_R^m)/g(\lambda_I^m)$ where R is a reference state. Therefore, $c_I = \frac{g(\lambda_I^m)}{w_I^m g(\lambda_R^m)}$. Note that c_I can also be obtained by matching rate constants method described in Ref. [103]. Via the RPE we can calculate the free energy landscape, path densities and reactive currents.

Rate calculation nested states

The rate constant calculation for the systems with only two states is given by Eq. 2.40. In a multiple state system, where states can be nested in between interfaces of other states, Eq. 2.40 is not valid anymore as it assumes that all transitions from I cross the outermost interface λ_I^m, which is not necessarily the case for systems which are nested in between interfaces. If Eq. 2.40 is used for states nested within interfaces, many transitions could be missed in the rate constant calculation, see Fig. 2.3 for example. One can circumvent this problem by calculating the rate constant via the path-type numbers introduced in ref. [108]. A path-type number is defined as $n^i_{IJ}(\lambda^k_I)$, which is the number of paths in replica i joining states I and J that have crossed at maximum interface λ^k_I. The superscript indicates that the paths should obey the condition of replica i in the ensemble. Because we have set the maximum interface, we can reweight these numbers with the WHAM weights obtained with the reweighting of the crossing probability as follows:

$$\tilde{n}_{IJ}(\lambda^k_I) = \bar{w}^k_I \sum_{i=1}^{m} n^i_{IJ}(\lambda^k_I), \tag{2.47}$$

where $\bar{w}^k_I = \left(\sum_l^k \frac{1}{w^l_I}\right)^{-1}$ is the WHAM weight for paths that have crossed interface λ^k_I at maximum (note that also the path-type numbers themselves can be reweighted, however, this is more difficult and should be the same as the weights obtained via the crossing probability anyway). Now we have the reweighted num-

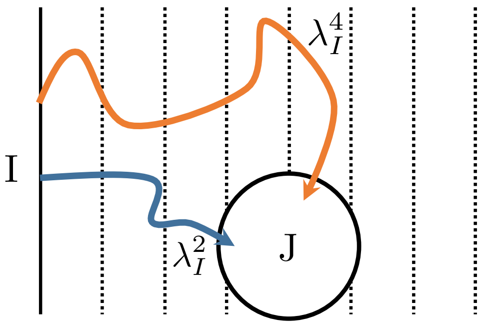

Figure 2.3: When a state J is nested within the interfaces of a state I, reactive pathways can have different path-types, λ^k_I. The rate constant equation in Eq. 2.40 assumes that every reactive pathway crosses each interface of I and hence, certain transitions would not be taken into account. Therefore, Eq. 2.49 is the more general rate equation.

ber of paths that join state I with state J that have crossed interface λ^k_I. Subsequently summing over all k replicas gives the reweighted number of paths coming from state I and ending in state J:

$$\tilde{n}_{IJ} = \sum_{k=1}^{m} \tilde{n}_{IJ}(\lambda^k_I). \tag{2.48}$$

Because the Wang-Landau scheme biases the simulation to sample all states equally via the state-swap bias, the path-numbers for each state need to be corrected for this bias. In an unbiased ensemble there are as many reactive IJ path as JI paths. Therefore, we split the obtained path-type matrix, $\tilde{n}_{IJ}$, into M matrices and symmetrize the Ith matrix: $\tilde{n}_{JI} = \tilde{n}_{IJ}$ and setting all other entries of the Ith matrix to zero, resulting in M different matrices with only a nonzero Ith row and a nonzero Ith column. Subsequently, all M matrices are joined via WHAM giving the individual weights for each state (these weights can also be used to calculate the coefficients, c_I, in RPE). This leads to a $M \times M$ transition matrix, $\tilde{n}^*_{IJ}$. Normalizing the matrix with the total numbers of paths going out of a state $\sum_J \tilde{n}^*_{IJ}$ and multiplying with the flux gives the generalized rate matrix for multiple state systems:

$$k_{IJ} = \phi_I \frac{\tilde{n}^*_{IJ}}{\sum_{J \in M} \tilde{n}^*_{IJ}}. \tag{2.49}$$

Free energy landscape

From the RPE we can calculate the free energy landscape for a chosen set of collective variables simply by projection:

$$\begin{aligned} F(\mathbf{q}) &= -k_B T \log p(\mathbf{q}) + C \\ p(\mathbf{q}) &= C \int \mathcal{D}\mathbf{x}^L \mathcal{P}[\mathbf{x}^L] \sum_{k=0}^{L} \delta[\mathbf{q}(\mathbf{x}_k) - \mathbf{q}], \end{aligned} \tag{2.50}$$

where C is an arbitrary constant, $\mathbf{q}(\mathbf{x}_k)$ are the collective variables at time step $\mathbf{x}_k$ and $\mathcal{P}[\mathbf{x}^L]$ is the reweighted path probability as given in Eq. 2.46.

One could histogram all the free energies after the simulation using saved pathways after one has obtained the proper weights. However, as every path that has reached a certain maximum interface based on λ_{max} is reweighted with the same weight, it is convenient to histogram on the fly for each interface separately, and subsequently reweight and sum all the histograms. As such, paths do not necessarily need to be stored for subsequent analysis.

Reactive path density

In Eq. 2.50, $p(\mathbf{q})$ gives the probability of each configuration, $\mathbf{q}(x_k)$. However, the mechanism of a process can sometimes be obscured by the dominance of certain metastable configurations. To get more insight into the mechanism, the reactive path density is useful. We can define two different path densities, per transition or per state.

The reactive path density for each transition separately is defined as:

$$n^r_{AB}(\mathbf{q}) = \int \mathcal{D}\mathbf{x}^L \mathcal{P}[\mathbf{x}^L] h_A(x_0) h_B(x_L) h_{\mathbf{q}}(\mathbf{x}^L). \tag{2.51}$$

where $h_A(x_0)h_B(x_L)$ picks out all the reactive trajectories between the appropriate states A and B and $h_{\mathbf{q}}(\mathbf{x}^L)$ is unity if the path visits $\mathbf{q}$.

The reactive path density out of state A is defined as:

$$n^r_A(\mathbf{q}) = \int \mathcal{D}\mathbf{x}^L \mathcal{P}[\mathbf{x}^L] h_A(x_0) h_{\mathbf{q}}(\mathbf{x}^L). \tag{2.52}$$

where now $h_A(x_0)$ picks out all reactive trajectories out of state A. The additional information that this type of path density offers, is that also the relative probability of reactive pathways out of state A can be seen.

Note that a path density does not add up to unity as the density is not normalized by the number of configurations used to construct the final histogram, but normalized by the number of paths.

Reactive path current

In self-assembly certain configurations are dead ends, which are still projected in $n_r(\mathbf{q})$. We can also average out these dead ends by calculating the reactive path current. The reactive path current is defined as follows [109]:

$$J_{BU}(\mathbf{q}) = C \int \mathcal{D}\mathbf{x}^L \mathcal{P}[\mathbf{x}^L] h_A(x_0) h_B(x_L) \sum_{k=0}^{L} \delta[\mathbf{q}(\mathbf{x}_k) - \mathbf{q}] \dot{\mathbf{q}}(\mathbf{x}_k) \tag{2.53}$$

where $\dot{\mathbf{q}}(x_k)$ is the estimated velocity of $\mathbf{q}$: $\dot{\mathbf{q}}(x_k) \approx \frac{\mathbf{q}(x_k+1)-\mathbf{q}(x_k-1)}{2\Delta t}$. This current has the same properties as the reactive path density, except for the fact that dead ends are averaged out, which results in the mean direction a path will follow.

Transition Path Theory

From SRTIS we can obtain the full rate matrix, $\mathbf{K}$, between states. However, essential questions important to understanding self-assembly are not answered directly by looking at $\mathbf{K}$ [110]. How can patchy particles with different initial configurations find the correct ground state? Through which sequence of intermediate states do the building blocks pass through towards the final structure before possible unbinding? Are there multiple routes possible? What is the overall rate constant of the process if all possible routes are considered? Transition Path Theory (TPT) is a convenient framework designed to help answer these questions. [109, 110]. Similarly to TIS, TPT requires state space to be separated in states which specify the (self-assembly) process. Therefore, it seems very suited to use TPT from TIS results. In what follows we describe a transition from A to B with possible intermediate states I along the way.

Committor

The starting point in TPT analyses in this thesis usually start from the committor, q_i^+, defined as the probability that the system, currently in i, will reach B before it reaches A. The commitment probabilities are computed from the transition matrix, $\mathbf{T}$, where T_{ij} gives the probability to reach j from i within a certain lag time, τ by solving the following linear set of equations:

$$q_i^+ = \sum_{k \in B} T_{ik} + \sum_{k \in I} T_{ik} q_k^+ \tag{2.54}$$

with the committor $q_A^+ = 0$ and $q_B^+ = 1$ as boundary conditions. We get the transition matrix from the rate matrix obtained via TIS by $\mathbf{T} = \exp(\mathbf{K}\tau)$.

We can also define the reverse committor probability $q_i^- = 1 - q_i^+$, which is the probability to reach A before B when currently in i.

Flux through states

As discussed above the question through which sequence of intermediate states do the building blocks pass through towards the final structure can also be deduced from TPT. For this purpose we can calculate the fluxes, f_{ij}, from which the dominant sequence of states during self-assembly can be deduced. The effective flux f_{ij} is given by:

$$f_{ij} = \pi_i q_i^- T_{ij} q_j^+ \tag{2.55}$$

where π_i is the equilibrium population of state i. Note that this definition of f_{ij} still contains all recrossings between intermediate states. To get rid of these recrossings, we can consider the net flux which will give a more clear picture of the sequence of states taken in the overall process:

$$f_{ij}^+ = \max[0, f_{ij} - f_{ji}] \tag{2.56}$$

Note that detailed balance dictates that the net flux for the reverse process is given by the transpose: $f_{ji}^- = (f_{ij}^+)^{\mathbf{T}}$.

Overall rate constant

TPT not only gives information about the mechanism of the process, one can also calculate the overall rate constant between states considering all possible routes between A and B. The overall rate constant is given by:

$$k_{AB} = \frac{\sum_{J \neq A} \pi_A T_{AJ} q_J^+}{\tau \sum_{J=0}^{M} \pi_J q_J^-} \tag{2.57}$$

where the denominator is necessary because the numerator does not take into account the probability of being in a forward $A \to B$ transition. Therefore, the overall rate constant is divided by the probability that the system while being in any state $\mathbf{I}$ was last in A and not in B.