Methods for Simulation and Calculation of Diffusion

Dissertation zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim
Fachbereich Phyik
der Johann Wolfgang Goethe-Universität
in Frankfurt am Main

von
Max Linke
aus Gera

Frankfurt am Main 2019
(D 30)

Diese Arbeit wurde vom Fachbereich Physik
der Johann Wolfgang Goethe-Universität als Dissertation angenommen.

Dekan: Prof. Dr. Michael Lang

1. Gutachter: Prof. Dr. Gerhard Hummer

2. Gutachter: Prof. Dr. Achilleas Frangakis

Datum der Disputation: 13.1.2020

Die vorliegende Dissertation wurde in der Zeit vom Januar 2015 bis Mai 2019 am Max Planck Institut für Biophysik in der Abteilung Theoretische Biophysik angefertigt.

Contents

Zusammenfassung

Das diffuse Verhalten von Makromolekülen in Lösung ist ein Schlüsselfaktor in der Kinetik von makromolekularen Bindungs und Anordnungs Prozessen, und wichtig in der theoretischen Beschreibung vieler Experimente. Experimente an Proteinlösungen mit hohen Dichten haben ergeben, dass sich die Diffusionsdynamik stärker verlangsamt als von der kolloidalen Theorie für nicht interagierende harte Kugeln erwartet. Es hat sich auch gezeigt, dass die Rotationsdiffusionsanisotropie bei hohen Dichten größer ist als in verdünnten Lösungen. Proteinlösungen bei hohen Dichten sind eine komplexe Flüssigkeit, die sich von der in der hydrodynamischen Theorie verwendeten Annahme der einfachen Flüssigkeit unterscheidet. Daher sind Methoden zur genauen Berechnung des Translations- und Rotationsdiffusionstensors sowie Simulationsalgorithmen zur Untersuchung von Lösungen mit hoher Dichte wichtig.

Simulationen stellen ein leistungsstarkes Werkzeug zur Untersuchung der Diffusion in komplexen Flüssigkeiten dar. Sie können verwendet werden, um die makroskopischen und mikroskopischen Auswirkungen komplexer Flüssigkeiten auf das Diffusionsverhalten zu untersuchen.

Es existieren bereits einfache Algorithmen, um Diffusion zu simulieren und um die Diffusionskoeffizienten aus Simulationen zu bestimmen. Die translatorische Diffusion von Molekülen in einfachen und komplexen Flüssigkeiten kann mit hoher Genauigkeit aus Simulationen bestimmt werden. Dies ist bei der Rotationsdiffusion noch nicht der Fall. Bestehende Algorithmen zur Berechnung der Rotationsdiffusionskoeffizienten aus Simulationen machen Annahmen über die Form des Proteins oder funktionieren nur für das kurzzeit Verhalten. Für die Simulation des diffusen Verhaltens von Makromolekülen gibt es heute zwei Möglichkeiten. Ein atomistischen Integrator mit expliziten Lösungsmittelmolekülen oder Coarse-Grained (CG) Simulationen mit impliziten Lösungsmittel. CG Simulationen mit impliziten Lösungsmitteln werden auch als Brownsche-Dynamik (BD) Simulationen bezeichnet. Für die CG Simulationen wird häufig der Ermak-McCammon-Algorithmus verwendet, um die zugrunde liegende Langevin-Gleichung zu lösen. Der Algorithmus ist eine Erweiterung des Euler-Maruyama-Integrators um Translation und Rotation in drei Dimensionen. Dieser Algorithmus bildet die Gleichgewichtswahrscheinlichkeit nur für kurze Zeitschritte korrekt ab und der Integrationsfehler hängt linear vom Zeitschritt ab. Es hat sich gezeigt, dass Monte Carlo basierte Algorithmen bei entsprechender Parametrisierung BD für die translationale Dynamik erzeugen können. Der

Vorteil eines Monte Carlo basierten Algorithmus besteht darin, dass er die korrekte Gleichgewichtsverteilung unabhängig vom gewählten Zeitschritt reproduziert. Dies ermöglicht im Gegenzug größere Zeitschritte in Simulationen zu benutzen. Das Ziel dieser Arbeit ist die Entwicklung neuer Methoden zur genauen Bestimmung des Rotationsdiffusionskoeffizienten aus Simulationen und die Erweiterung bestehender Monte-Carlo-Algorithmen um Rotationsdynamik.

Das erste Projekt beschäftigt sich mit der Frage, wie man die Rotationsdiffusionskoeffizienten aus Simulationen genau bestimmen kann. Wir entwickeln eine Quaternionen basierten Algorithmus zur Berechnung des Rotationsdiffusionstensors aus Simulationen und eine Theorie für die Auswirkungen von periodischen Randbedingungen (PBC) auf den Rotationsdiffusionskoeffizienten in atomistischen Simulationen.

Unser Verfahren zur Berechnung von Rotationsdiffusionskoeffizienten basiert auf den Quaternion-Kovarianzen von Favro für ein frei rotierendes starres Molekül. Die Kovarianzen sind nur im Hauptachsen System (PCS) des Rotationsdiffusionstensors gültig. Die Kovarianzen können für ein beliebiges Referenzkoordinatensystem (RCS), zum Beispiel einer Simulation, verallgemeinert werden, wenn man die Hauptachsen des Rotationsdiffusionstensors im RCS kennt. Wir zeigen, dass keine Vorkenntnisse über den Diffusionstensor und seine Hauptachsen erforderlich sind, um die generalisierten Kovarianzen aus Simulationen mit gängigen RMSD-Verfahren zu berechnen. Wir entwickeln zwei Methoden, um die aus Simulationen berechneten Kovarianzen an unsere verallgemeinerten Gleichungen zu fitten. In der ersten Methode minimieren wir die Summe der quadrierten Fehlerabweichungen zwischen Modell- und Simulationsdaten. Für diese sechsdimensionale Optimierung verwenden wir einen Simulated Annealing Algorithmus. Alternativ kann der Rotationsdiffusionstensor auch aus einer Eigenwertzerlegung der Kovarianz nach der Integration bestimmt werden. Um die Auswirkungen von Rauschen in der Integration zu minimieren, wenden wir zunächst eine Laplace-Transformation an, um die Kovarianzen zu glätten. Für rauschfreie Daten sollte der resultierende Rotationsdiffusionskoeffizient unabhängig vom Wert der Laplace-Variable sein. In der Praxis werden jedoch die besten Ergebnisse mit einem Wert erzielt, der nahe der inversen Autokorrelationzeit der Drehbewegung ist.

Der Fit der Quaternionen-Kovarianzen geht davon aus, dass die Simulationsbox unendlich groß ist und dass nichts den hydrodynamischen Fluss beeinflusst. Aus praktischen Gründen verwenden Simulationen oft periodische Randbedingungen (PBC), um ein sich unendlich wiederholendes System anstelle eines unendlich verdünnten Systems zu simulieren. PBC verlangen, dass der Nettofluss über die Grenzen der Simulationsbox hinweg Null ist, was den hydrodynamischen Fluss stark einschränkt. Es hat sich gezeigt, dass PBC die translatorische Diffusion bei kleinen

Simulationsboxen verlangsamen. Mit unserem Algorithmus untersuchen wir, ob dieser Effekt auch bei der Rotationsdiffusion vorhanden ist. Darüber hinaus entwickeln wir eine Theorie über die Wirkung von PBC auf die Rotationsdiffusion. Wir modellieren die hydrodynamische Strömung um eine Kugel in einem unendlichen periodischen Gitter und leiten ein einfaches sphärisches Couette-Modell ab, um die Volumenabhängigkeit des Rotationsdiffusionstensors zu beschreiben. Unsere Theorie sagt voraus, dass die PBC auch die Rotationsdiffusion für kleine Simulationsboxen verlangsamt, wenn auch nicht so stark wie bei der Translationsdiffusion. Die Reduzierung der Rotationsdiffusion hängt nur vom Boxenvolumen ab und nicht von der Form des Makromoleküls oder der Form der Simulationsbox. Das heißt, unsere Theorie kann verwendet werden, um den Rotationsdiffusionskoeffizienten in einer unendlichen Verdünnung aus Simulationen unter PBC zu berechnen.

Die Berechnung von Kovarianzen aus Simulationsdaten ist ein bekannt schwieriges Problem und molekulardynamische Simulationen sind sehr rechenintensiv. Um den Fehler zu minimieren, verwenden wir daher immer alle verfügbaren Simulationsdaten, um die Quaternionen Kovarianzen zu berechnen. Um den Fehler der Kovarianzen abzuschätzen, verwenden wir reine Rotation BD Simulationen mit unserem geschätzten Diffusionstensor als Input. Wir leiten den Rotations-Algorithmus aus den Quaternionen Kovarianzen ab.

Um unsere Theorie der Wirkung von PBC auf die Rotationsdiffusion und unsere Methode zur Bestimmung des Rotationsdiffusionstensors zu testen wenden wir sie auf voll atomistische Molekulardynamik (MD) Simulationen an und vergleichen sie mit hydrodynamischen Theorie- und Literaturwerten. Wir simulieren Pferdeherzmyoglobin und B-DNA. Die B-DNA hat eine zylindrische Form und dient als Modell für ein Makromolekül mit einer großen Anisotropie. Wir simulieren Myoglobin in einer $\approx$ 150 mM NaCl-Lösung mit GROMACS 5 in einer rhombischen Dodekaeder-Box bei fünf verschiedenen Boxvolumen, von 206 nm^3 bis 516 nm^3. Wir simulieren B-DNA in einer $\approx$ 150 mM NaCl-Lösung mit GROMACS 2016 in einer kubischen Box bei drei verschiedenen Boxvolumen, von 233 nm^3 bis 580 nm^3. Für jedes Makromolekül und jede Boxgröße haben wir mindestens fünf Simulationen mit einer Dauer von 1 μs bis 3 μs durchgeführt. Die Fehler der Kovarianzen wurden durch wiederholte rein rotierende BD Simulationen mit unserem gefitten Diffusionstensor als Input ermittelt. Die Variation der Hauptachsen wurde mit Hilfe der Frobenius-Norm zwischen Rotationsdiffusionstensoren, von Fits der BD Simulationen im Vergleich zu unserer Fit der MD-Simulationsdaten, bestimmt. Die berechneten Rotationsdiffusionskoeffizienten für Myoglobin und B-DNA stimmen sowohl mit der hydrodynamischen Theorie als auch mit den Experimenten hervorragend vereinbar. Unsere Schätzung der harmonischen mittleren Relaxationszeit, beider Moleküle, liegt innerhalb des Fehlers der Literaturwerte. Aus den Simulatio-

nen bei verschiedenen Simulationvolumen zeigen wir auch, dass die Rotationsdiffusion mit abnehmenden Simulationsvolumen verlangsamt wird. Unter Verwendung der Stokes-Einstein-Relation ist unsere Schätzung des Rotationsdiffusionstensors bei unendlicher Verdünnung in ausgezeichneter Übereinstimmung mit Literaturwerten. Auch zuvor veröffentlichte Simulationsdaten von GB3 und Ubiquitin bei unterschiedlichen Simulationsvolumen, Formen der Simulationsbox und Wassermodellen folgen unseren Vorhersagen.

Im zweiten Projekt dieser Arbeit entwickeln wir einen neuartigen BD Algorithmus, der die Gleichgewichtsverteilung unabhängig vom gewählten Zeitschritt reproduziert. Unser Algorithmus ist eine Erweiterung bestehender Monte-Carlo-basierter Algorithmen für isotrope Partikel um Rotationsdynamik. Außerdem entwickeln wir eine Erweiterung für positionsabhängige Translationsdiffusions. Im weiteren Verlauf der Arbeit nennen wir diese Algorithmen Brownsche Dynamik Monte-Carlo (BDMC). Wir entwickeln zusätzlich eine Generalisierung, um BDMC Algorithmen für verschiedene Akzeptanz- und Übergangswahrscheinlichkeiten abzuleiten.

Für die Translationsfunktion hat die Übergangswahrscheinlichkeit einen freien Parameter a. Der Parameter a ist so gewählt, dass die mittlere quadratische Verschiebung (MSD) eines einzelnen Monte-Carlo-Schrittes und der MSD von BD nach einer Zeit Δt übereinstimmen. Daher definiert der Translationsdiffusionskoeffizient und der Zeitschritt Δt den Parameter a. Für anisotrope Makromoleküle sind die Translationsbewegungen nur im PCS des translatorischen Diffusionstensors definiert. Das Äquivalent zu dem MSD für Rotationen sind die Kovarianzen von Favro. Daher konstruieren wir unsere Drehbewegungen im Quaternionenraum. Die kleinen Rotationen im Quaternionenraum werden durch die Wahl eines Zufallsvektors innerhalb der Einheitskugel erzeugt. Der entsprechende Vektor repräsentiert die Drehachsen und die Länge des Vektors bestimmt den Drehwinkel. Der Zufallsvektor wird um drei Variablen a_i $i = (1, 2, 3)$ skaliert. Eine Variable für jede Achse. Die drei Variablen a_i sind so gewählt, dass die Quaternion-Kovarianzen nach einem Schritt gleich den erwarteten Quaternion-Kovarianzen für einen gegebenen Rotationsdiffusionstensor und Zeitschritt Δt sind. Der Einfachheit halber verwenden wir eine Kurzzeit Approximation der Quaternionen Kovarianzen im PCS des Rotationsdiffusionstensors. Für eine Simulation anisotroper Makromoleküle werden die Rotations- und Translationsbewegungen in den jeweiligen PCS gezogen. Um die rotierenden und translatorischen Monte-Carlo-Schritte zu kombinieren, muss man beide in einem gemeinsamen RCS anwenden. In einer Simulation müssen wir daher die Orientierung im RCS mit Bezug auf das jeweilige PCS des translatorischen und rotatorischen Diffusionstensors speichern. In einer einzigen Monte-Carlo-Bewegung muss eine Translations- und Rotationsbewegung durchgeführt werden. Um ein detailliertes Gleichgewicht zu gewährleisten, muss die Reihenfolge, in der Translation und Ro-

tation angewendet werden, zufällig gewählt werden. Für Multipartikelsimulationen muss der Monte-Carlo-Sweep alle Partikel einmal berühren, um die gesamte Simulation in einem Zeitschritt voranzutreiben. Um ein detailliertes Gleichgewicht zu gewährleisten, muss die Reihenfolge, in der die Partikel bewegt werden, zufällig festgelegt werden.

Bisherige BDMC Algorithmen verwendeten nur die Metropolis-Hastings Akzeptanzfunktion und eine Gleichverteilung als Übersetzungswahrscheinlichkeit. Metropolis-Hastings ist nicht die einzige Akzeptanzfunktion, die detailliertes Gleichgewicht erfüllt. Wir berechnen den Fehler des ersten, zweiten und dritten Moments der Gleichgewichtsverteilung in Abhängigkeit vom Parameter a der Übersetzungswahrscheinlichkeit für ein eindimensionales Teilchen unter konstanter Kraft. Mit Metropolis-Hastings sind die ersten drei Momente genau auf die dritte Ordnung. Die Glauber-Akzeptanzfunktion, eine gängige Alternative zu Metropolis-Hastings, ist eine Ordnung genauer als Metropolis-Hastings. Wir entwickeln auch eine neue Akzeptanzfunktion, die in den ersten drei Momenten bis zur sechsten Ordnung genau ist.

Viele Probleme in der statistischen Mechanik erfordern positionsabhängige Diffusionskoeffizienten. Wir verwenden eine Gaußsche Übersetzungswahrscheinlichkeit, die von einem Parameter a und dem positionsabhängigen Diffusionskoeffizienten abhängt, um ein neues effektives Potenzial V zu definieren. Das Potenzial V hängt nur von der Übersetzungswahrscheinlichkeit ab. Es kann daher mit jeder Akzeptanzfunktion verwendet werden.

Wir testen unsere Algorithmen an Spielzeugmodellen des harmonischen Oszillators und vergleichen sie mit den Euler-Maruyama und BAOAB Algorithmen. Unsere BDMC Algorithmen sind in der Lage, die korrekte Gleichgewichtsverteilung unabhängig vom gewählten Integrationszeitschritt zu reproduzieren. Die berechneten Autokorrelationszeiten der Energie sind auch in guter Übereinstimmung mit den theoretischen Werten unter Verwendung akzeptabler Zeitschritte. Der kleinste Fehler in der Autokorrelationszeit erreichen wir mit der uniforme Übersetzungsverteilung und unser eigenen Akzeptanzfunktion. Der Fehler der Autokorrelationszeit skaliert annähernd mit einem Faktor zwei. Das heißt, wenn der Zeitschritt halbiert wird, reduziert sich der Fehler um den Faktor vier. Sie schlägt sogar traditionelle Integratoren wie den Euler-Maruyama und BAOAB. Der größte Fehler wird mit einer Gaußschen Übersetzungsverteilung und der Akzeptanzfunktion Metropolis-Hastings erzeugt. Für die positionsabhängige Diffusion transformieren wir den harmonischen Oszillator mit konstantem Diffusionskoeffizienten in ein Koordinatensystem mit einem positionsabhängigen Diffusionskoeffizienten. In diesem Beispiel können unsere BDMC-Algorithmen die Gleichgewichtsverteilung unabhängig vom gewählten Zeitschritt reproduzieren. Die Autokorrelationszeit hat jedoch einen größeren Fehler für die BDMC Algorithmen als für den Euler-Maruyama und BAOAB Integrator.

Der Fehler skaliert mit einem Faktor 0.8 für unsere Algorithmen.

Wir haben auch die Bindungsraten und Dissoziationskonstanten für die UIM-1-Ubiquitin- und CUE-Ubiquitin-Komplexe berechnet. Als Kraftfeld benutzen wir das KimHummer (KH) CG Proteinmodell. Das KH Kraftfeld ersetzt Aminosäuren durch einzelne Interaktionsstellen an der C_α -Position und verwendet starre Körper für die Proteine. Wir haben beide Komplexe für ein breites Spektrum von Zeitschritten von 0.05 ps bis 100 ps simuliert. Um zu bestimmen, ob ein Komplex gebunden oder ungebunden ist, verwenden wir Transition-Based-Assignment (TBA) auf einem zweidimensionalen Reaktionsweg, der aus der Gesamtenergie und dem Mindestabstand zwischen zwei beliebigen Aminosäuren der beiden Reaktionspartner besteht. In TBA definieren wir eine Übergangsregion, in der zunächst unklar ist, ob ein Komplex gebunden oder ungebunden ist. Ein Filter wird dann verwendet, um basierend auf der Geschichte der Trajektorie zu entscheiden, ob eine Konfiguration in der Übergangszone gebunden oder ungebunden ist. Die Simulationen werden unter PBC durchgeführt, bei denen die Dissoziationskonstante Simulationsboxvolumen abhängig ist. Daher haben wir jeden Komplex bei mehreren Volumen simuliert, um die Dissoziationskonstante bei unendlichem Volumen zu erhalten. Für die Dissoziationskonstante sind unsere Schätzungen aus Simulationen in ausgezeichneter Übereinstimmung mit Experimenten. Diese Ergebnisse sind unabhängig, wenn wir die Dissoziationskonstanten aus den Bindungsraten oder der zu bindenden Wahrscheinlichkeit berechnen. Unsere geschätzten On-Raten von $\approx 10^{10}$ sind überraschend schnell. Es sollte jedoch beachtet werden, dass wir erwarten, dass die Raten schneller sind als in einem reinen atomaren Kraftfeld, da das KH Kraftfeld viel glatter ist. In Anbetracht der Vielzahl von Funktionen, die Ubiquitin in der Zelle ausführt, wäre es außerdem von Vorteil, wenn es eine schnelle On-Rate hat.

Im letzten Projekt dieser Arbeit beschreiben wir die Implementierung einer neuen Monte-Carlo-Engine die auf dem KH Proteinmodell basiert. Die Monte-Carlo-Engine ist in zwei Programme aufgeteilt: Complexes++ und pycomplexes. Das erste ist ein allgemeines Werkzeug für Monte-Carlo-Simulationen von CG Proteinmodellen, geschrieben in C++14. pycomplexes ist eine in Python geschriebene Toolbox, die Eingabedateien für Complexes++ mit dem KH Modell erzeugt. Diese Trennung ermöglicht die Monte-Carlo-Engine allgemein und flexibel zu halten. In Complexes++ sind zwei thermodynamischen Ensembles, die NVT und NIIT, zusammen mit Replica-Exchange Monte-Carlo implementiert. Die Algorithmen für den Replikataustausch sind für beide Ensembles und für Hamiltonian-Replica-Exchange implementiert. Der zuvor beschriebene neue BDMC Algorithmus ist ebenfalls implementiert. Das KH Modell kann kurze flexible Aminosäureketten und größere starre Aminosäureketten beschreiben. Die flexiblen Aminosäureketten existieren, um ein realistisches Modell eines Linker von mehrere starre Strukturen zu haben. Das Originalmodell für die flexiblen Ketten wurde mit einer Energiefunktion implementiert, die denen von voll atomistischen Kraftfeldern ähnelt. Wir er-

setzen die flexiblen Gliederketten durch ein effektives Potential basierend auf dem Gaußschen Polymerkettenmodell. Wir haben auch einen Algorithmus zum Erzeugen repräsentativer Konfigurationen für die flexiblen Ketten nach einer Simulation entwickelt. Complexes++ verwendet OpenMP und MPI sowie gängige Algorithmen in der Molekulardynamik um auf Laptops und Supercomputern optimal zu laufen. Am Ende zeigen Benchmarks verschiedenster Systeme mit bis zu einer Million Aminosäuren.

Zusammenfassend lässt sich sagen, dass wir in dieser Arbeit zeigen, wie man den vollständigen anisotropen Rotationsdiffusionstensor aus Simulation genau abschätzen kann, wobei finite Größeneffekte berücksichtigt werden. Wir entwickeln auch einen neuartigen Brownsche-Dynamik-Algorithmus basierend auf Monte-Carlo-Algorithmen. Der Brownsche-Dynamik-Algorithmus ist in einer neuen Monte-Carlo-Engine für Coarse-Grained Proteinmodelle implementiert, die als Complexes++ bezeichnet wird.

Summary

The diffusive behavior of macromolecules in solution is a key factor in the kinetics of macromolecular binding and assembly, and in the theoretical description of many experiments. Experiments on high-density protein solutions have found that a slow down of the diffusion dynamics is larger than expected from colloidal theory for non-interaction hard-spheres. It has also been shown that the rotational diffusion anisotropy in high-density protein solutions is larger than in dilute ones. High-density protein solutions are a complex fluid that is different from the neat fluid assumption used in the hydrodynamic theory. It is therefore important to have methods to accurately calculate the translational and rotational diffusion tensor from simulations as well as simulation algorithms to explore high-density solutions.

Simulations provide a powerful tool to study diffusion in complex fluids. They can be used to study the macroscopic and microscopic effects of complex fluids on the diffusive behavior. There has been already a lot of work done to accurately simulate diffusion and to determine the diffusion coefficients from simulations.

The translational diffusion of molecules in simple and complex liquids can be determined with high accuracy from simulations. This is not yet the case for rotational diffusion. Existing algorithms to calculate the rotational diffusion coefficients from simulations make assumptions about the shape of the protein or only work at short times. For the simulation of diffusive behavior of macromolecules two options exist today. An all-atom integrator with explicit solvent molecules or coarse-grained (CG) simulations with an implicit solvent. CG simulations of dynamic behavior with implicit solvent are also called Brownian dynamics (BD) simulations. For the CG simulations the Ermak-McCammon algorithm is often used to solve the underlying Langevin equation. The algorithm is an extension of the Euler-Maruyama integrator to include translation and rotation in three dimensions. This algorithm only correctly reproduces the equilibrium probability for short time-steps and the error depends linearly on the time-step. It has been shown that Monte Carlo based algorithms can produce BD for translational dynamics, when appropriately parametrized. The advantage of Monte Carlo based algorithm is that they will reproduce the correct equilibrium distribution independent of the chosen time-step. This in return allows choosing larger time-steps in simulations. The aim of this thesis is to develop novel methods to accurately determine the rotational diffusion coefficient from simulations and extend existing Monte Carlo algorithms to include rotational dynamics.

The first project addresses the question of how to accurately determine the rotational diffusion coefficients from simulations. We develop a quaternion based method to calculate the rotational diffusion tensor from simulations and a theory for the effects of periodic boundary conditions (PBC) on the rotational diffusion coefficient in simulations.

Our method for calculating rotational diffusion coefficients is based on the quaternion covariances from Favro for a freely rotating rigid molecule. The covariances as formulated by Favro are only valid in the principal coordinate system (PCS) of the rotation diffusion tensor. The covariances can be generalized for an arbitrary reference coordinate system (RCS), i.e., a simulation, given the principle axes of the rotational diffusion tensor in the RCS. We show that no prior knowledge of the diffusion tensor and its principal axes is required to calculate the generalized covariances from simulations using common root-mean-square distance (RMSD) procedures. We develop two methods to fit the covariances calculated from simulations to our generalized equations to fit the rotational diffusion tensor. In the first method we minimize the sum of the squared error deviations between model and simulation data. For this six dimensional optimization we use a simulated annealing algorithm. Alternatively the rotational diffusion tensor can also be determined from a eigenvalue decomposition of covariance after integration. To minimize the effects of sampling noise in the integration we first apply a Laplace-transformation to smooth the covariances at large times. For ideal sampling the resulting rotational diffusion coefficient should be independent of the value of the Laplace variable. In practice, however, the best results are achieved using a value close to the inverse autocorrelation time of the rotational motion.

The fit of the quaternion covariances assume that the simulation box is infinitely large and nothing influences the hydrodynamic flow. However, simulations often use PBC, to simulate an infinitely repeating system instead of an infinitely diluted system, for practical reasons. The PBC require that the net flux of across simulation box boundaries is zero, severely limiting the hydrodynamic flow. It has been shown that PBC slow down translational diffusion for small box sizes. We use our algorithm to investigate if this effect is also present in rotational diffusion. In addition, we develop a theory of the effect of PBC on rotational diffusion. We model the hydrodynamic flow around a sphere in an infinite periodic grid and derive a simple spherical Couette-model to describe the volume dependency of the rotational diffusion tensor. Our theory predicts that PBC also slow down rotational diffusion for small boxes albeit not as much as for translation diffusion. The reduction in rotational diffusion only depends on the box volume and not the shape of the macromolecule or simulation box. Our theory can be used to calculate the rotational diffusion coefficient in an infinite dilution from simulations under PBC.

Calculating covariances from simulation data is a notoriously difficult problem and molecular dynamics simulations are very compute intensive. To minimize the error we, therefore, always use all available simulation data to calculate quaternion covariances. To estimate the error of the covariances we use purely rotational BD simulations with our estimated diffusion tensor as input. We derive the rotational BD algorithm from the quaternion covariances.

To test our theory of the effect of PBC on the rotational diffusion and our method for the determination of the rotational diffusion tensor we apply it to all-atom molecular dynamics (MD) simulations and compare against hydrodynamic theory and literature

values. We simulated horse heart myoglobin and B-DNA in neat solutions. The B-DNA has a cylindrical shape and serves as a model of a macromolecule with large anisotropy. We simulated myoglobin in an $\approx 150\,\mathrm{mM}$ NaCl solution using GROMACS 5 in a rhombic dodecahedron box at five different box volumes, from $206\,\mathrm{nm}^3$ to $516\,\mathrm{nm}^3$. We simulated B-DNA in an $\approx 150\,\mathrm{mM}$ NaCl solution with GROMACS 2016 in a cubic box at three different box volumes, from $233\,\mathrm{nm}^3$ to $580\,\mathrm{nm}^3$. For each macromolecule and box size we ran a minimum of five simulations each between $1\,\mu\mathrm{s}$ to $3\,\mu\mathrm{s}$ long. The errors of the covariances have been determined using repeated purely rotational BD simulations with our fit as input. The variation of the principal axes have been determined using the Frobenius norm between two rotational diffusion tensors, from fits of to the BD simulations compared with our fit of the MD simulation data. The calculated rotational diffusion coefficients for both myoglobin and B-DNA are in excellent agreement both with hydrodynamic theory and experiments. For both our estimate of the harmonic mean relaxation time is within the error of literature values. From the simulations at different box sizes we can also show that the rotational diffusion is indeed slowed down with decreasing box size. Using the Stokes-Einstein relation our estimate of the rotational diffusion tensor at infinite dilution is in excellent agreement with experimental values. Also previously published simulation data of the third IgG-binding domain of Protein G (GB3) and ubiquitin at different box volumes, shapes and water models follows our predictions.

In the second project of this thesis, we develop a novel BD algorithm that reproduces the equilibrium distribution independent of the chosen time-step. Our algorithm is an extension of existing Monte Carlo based algorithms for isotropic particles. We also developed an extension for position-dependent translational diffusion. In the remainder of the thesis, we call these methods Brownian dynamics Monte Carlo (BDMC) algorithms. We develop a framework to derive BDMC algorithms for different acceptance functions and transition probabilities.

The quaternion covariances from Favro can be used to add rotational moves to BDMC algorithms. For the translational moves, the transition probability has one free parameter a. The parameter a is chosen such that the mean square displacement (MSD) of a single Monte Carlo step and the MSD of BD after a time Δt are equal. Therefore, the translation diffusion coefficient and the time-step Δt define a. For anisotropic macromolecules the translational moves are only defined in the PCS of the translational diffusion tensor. The equivalent of the MSD for rotations are the covariances of Favro. Hence, we construct our rotational moves in quaternion space. The small rotations in quaternion space are generated by choosing a random vector within the unit-sphere. The corresponding unit-vector represents the rotation axes and the length of the vector encodes the rotation angle. The random vector is scaled by three variables $a_i,\ i = (1, 2, 3)$. One for each axes. The three variables a_i are chosen such that quaternion covariances after one step equals the expected quaternion covariances for a given rotational diffusion tensor and time-step Δt. For simplicity we use a short-time approximation of the quaternion covariances in the PCS of the rotational diffusion tensor. For a simulation of anisotropic macromolecules

the rotational and translational moves are drawn in the respective PCS. To combine the rotational and translational Monte Carlo steps one has to apply both in a common RCS. In a simulation we therefore need to add book-keeping steps that store the orientation in the RCS with reference to the respective PCS of the translational and rotational diffusion tensor. For a Monte Carlo step to represent an increment of time both a translation and rotation move have to be applied. To ensure detailed balance the order in which to apply translation and rotation is randomly selected. For multi-particle simulations the Monte Carlo sweep has to touch all particles exactly once to advance the whole simulation by a time-step. To ensure detailed balance the order in which particles are moved is randomized.

Previous BDMC algorithms only used the Metropolis-Hastings acceptance function and a uniform distribution as translation probability. The Metropolis-Hastings acceptance function is not the only function that fulfills detailed balance. We calculate the error of the first, second and third moment of the equilibrium distribution depending on the parameter a of the translation probability for a one-dimensional particle under constant force. Using Metropolis-Hastings the first three moments are accurate to order three. The Glauber acceptance function, a common alternative to Metropolis-Hastings, is one order more accurate than the Metropolis-Hastings acceptance function. We also develop a novel acceptance function that is accurate up to sixth order in the first three moments.

Many problems in statistical mechanics require position-dependent diffusion coefficients. We use a Gaussian translation probability that depends on a parameter a and the position-dependent diffusion coefficient to define a new effective potential V. The potential V only depends on the translation probability. It can, therefore, be used with any acceptance function.

We test our algorithms on toy models of the harmonic oscillator and compare it to the Euler-Maruyama and BAOAB algorithms. Our BDMC algorithms are able to reproduce the correct equilibrium distribution independent of the chosen integration time-step. The calculated autocorrelation times of the energy are also in good agreement with theory values at reasonable time-steps. The smallest error in the autocorrelation time is achieved with the uniform translation distribution and our own acceptance function. The error of the autocorrelation time than scales approximately with a factor two. Meaning when the time-step is halfed the error reduces by a factor four. It even beats traditional integrators like the Euler-Maruyama and BAOAB. The largest error is produced with a Gaussian translation distribution and the Metropolis-Hastings acceptance function. For the position dependent diffusion we transform the harmonic oscillator with constant diffusion coefficient into a coordinate system with a position-dependent diffusion coefficient. In this toy example our BDMC algorithms can reproduce the equilibrium distribution independent of the chosen time-step. However, the autocorrelation time has a larger error for the BDMC algorithms than for the Euler-Maruyama and BAOAB integrator. The error scales with a factor 0.8 for our algorithms.

We also calculated the binding rates and dissociation constant for the UIM-1-ubiquitin

and CUE-ubiquitin complexes. As a forcefield, we use the Kim-Hummer (KH) CG protein model. The KH forcefield replaces amino acids with single interaction sites at the C_α position and uses rigid bodies for proteins. We simulated both complexes for a wide range of time-steps from 0.05 ps to 100 ps. To determine if a complex is bound or unbound we use transition based assignment (TBA) on the two-dimensional reaction path consisting of the total energy and the minimal distance between any two amino acids of the two reaction partners. In TBA we define a transition region in which it is initially unclear if a complex is bound or unbound. A filter is then used to decide based on the history of the trajectory if a configuration in the transition region is bound or unbound. The simulations are done under PBC for which the dissociation constant is box volume dependent. We also calculate the dissociation constant from the binding rates to double check our rates. Therefore, we simulated each complex at multiple box sizes to get the dissociation constant at infinite box size. For the dissociation constant, our estimates from simulations are in excellent agreement with experiments. These results are independent if we calculate the dissociation constants from the binding rates or the probability to be bound. Our estimated on-rates of $\approx 10^{10}$ are surprisingly fast. It should be noted though that we expect the rates to be faster than in an all-atom forcefield because the KH forcefield is much smoother. In addition considering the multitude of functions, ubiquitin performs in the cell it would be beneficial for it to have a fast on-rate.

In the last project of this thesis, we describe the implementation of a new Monte Carlo engine that is based on the KH protein model. The Monte Carlo engine is split into two programs Complexes++ and pycomplexes. The first is a general tool for Monte Carlo simulations of CG protein models written in C++14. pycomplexes is a toolbox written in Python that generates input files for Complexes++ using the KH model. This separation allows the Monte Carlo engine to be extensible and flexible while maintaining sensible defaults to quickly set up a simulation. In Complexes++ two thermodynamic ensembles, the NVT and NIIT, are implemented together with replica exchange Monte Carlo for enhanced sampling. The replica exchange algorithms are implemented for both ensembles and for Hamiltonian replica exchange. The previously described novel BDMC algorithm is also implemented. The KH model can describe short flexible amino acid chains and larger rigid structures. The flexible amino acid chains exist to have a realistic model of a linker for multiple rigid structures. The original model for the flexible chains was implemented with an energy function similar to all-atom forcefields. We replace the flexible linker chains with an effective potential based on the Gaussian polymer chain model. We also developed an algorithm to generate representative configurations for the flexible chains after a simulation. The Complexes++ program is optimized to run well on laptops and supercomputer, using OpenMP and MPI, and uses established algorithms so that the total run-time scales linearly with the number of beads. In the end, we show benchmarks of a variety of different systems with up to one million amino acids.

In summary, in this thesis we show how to accurately estimate the full anisotropic rotational diffusion tensor from simulation, taking into account finite-size effects. We also

develop a novel Brownian dynamics algorithm based on the Monte Carlo method. The Brownian dynamics algorithm is implemented in a new Monte Carlo engine for coarse-grained protein models called Complexes++.

Chapter 1

Introduction

The diffusive behavior of macromolecules in solution is a key factor in the kinetics of macromolecular binding and assembly [86, 99, 137, 140, 186]. The association of two proteins is an ubiquitous event in cells. Often one of the reaction partners is free to move through the intra- and extracellular environment and must find its partner through diffusion. When the post-diffusional binding is much faster than the diffusional association the reaction is diffusion limited. Reactions with an association rate larger than 10^5 $M^{-1}s^{-1}$ are diffusion limited [5, 182], see Figure 1.1. Reactions with a lower association rate are limited by conformational change.

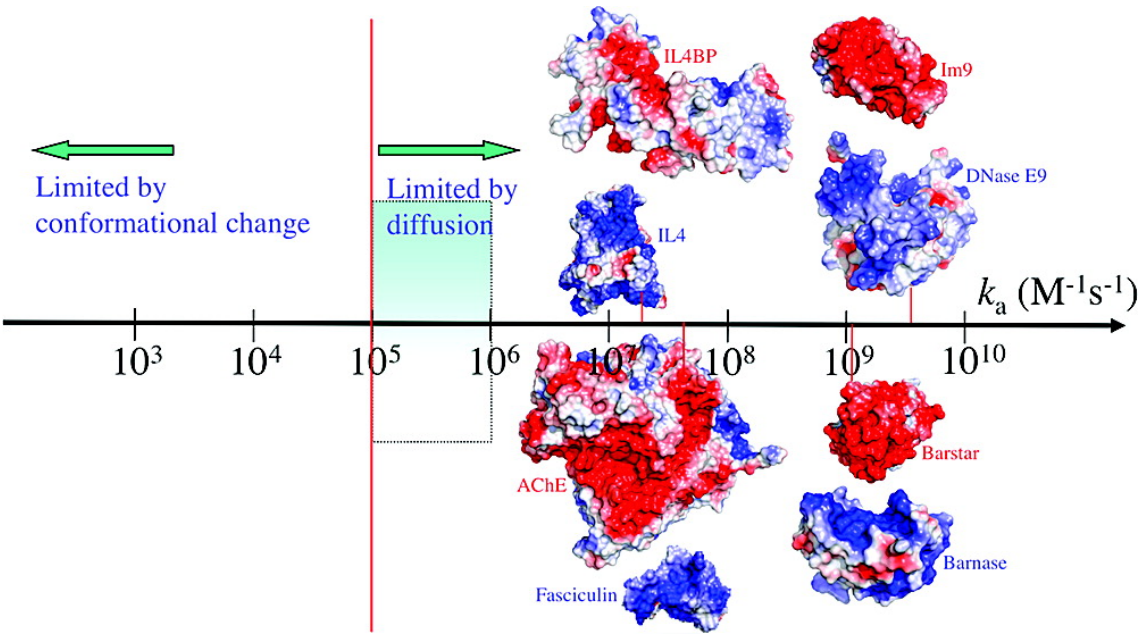

Figure 1.1: Spectrum of association rate constants. The red vertical line marks the start of the diffusion controlled regime. (Reprinted with permission from ref. [5] and [182], Copyright 2008 Wiley Interscience.)

Diffusion also features prominently in the theoretical description of many experiments. For example rotational dynamics are used in light scattering [168], nuclear magnetic resonance (NMR) [23, 26, 45, 61, 110, 116, 206, 224, 229, 230], fluorescence anisotropy decay and fluorescence resonance energy transfer (FRET) [83, 107, 118, 201], or dielectric spectroscopy [231]. Therefore, it is important to accurately calculate diffusive dynamics, and

simulate diffusion at large length- and time-scales to analyze experiments more accurately and better understand biological processes.

Recent NMR experiments on high-density solutions of proteins [164, 165] show a larger rotational diffusion anisotropy compared to proteins in dilute solutions, consistent with the behavior expected from steric clashes seen in molecular dynamics (MD) simulations [130, 133, 238]. Experiments on crowded protein solutions have found a reduction in protein self-diffusion by up to 20% [166]. The reduction in self-diffusion has been predicted by colloidal theory of non-interacting hard spheres [215]. However, experimental evidence and simulations show that the colloidal model underestimates the slowdown of diffusion in high density protein solutions [138, 223, 238]. These experiments show the need to develop a method for the accurate determination of diffusion coefficients in simulations of complex liquids.

Diffusion is closely linked to Brownian motion. Diffusion is generally used to describe the net movement of molecules due to a chemical gradient. When the chemical gradient is zero there is no net movement. However the individual molecules are still randomly moving. This is called Brownian motion. Einstein showed in his 1905 paper [48] that Brownian motion can be described as the random motion of molecules suspended in a liquid. His theory showed how Avogrado's number is connected to the mean square displacement (MSD). Using a different derivation Langevin [112, 152] obtained the same equation as Einstein for Avogrado's number. Later Perrin proofed experimentally that Einsteins theory of Brownian motion is correct by calculating Avogrado's number [155]. The work of Einstein only concerned translational diffusion. Perrin later worked on rotational diffusion [153, 154]. His theory was later extended by Furry [57], and Favro [51].

All-atom and coarse-grained (CG) simulations have long been used to study diffusional behavior [9, 39, 43, 46, 59, 68, 79, 123, 130, 133, 138, 141, 219, 221–223, 228, 232, 236, 238]. Simulations are well suited to investigate diffusion as they allow to study macroscopic effects [130, 221–223, 236], like protein concentration, as well as microscopic effects [9, 138, 238], like protein-protein interactions, influencing diffusive behavior. Translational diffusion coefficients can be determined with high accuracy from simulations of simple and complex fluids [46, 68, 219, 221, 222, 236]. Determining the rotational diffusion tensor of a molecule or molecular assembly from MD trajectories is non trivial. Current methods to calculate rotational diffusion tensors using the second order rotational correlation function have focused on isotropic macromolecules or macromolecules with small anisotropies [192, 232]. The focus of recent quaternion-based algorithms has also been on small anisotropy or on estimates of the full anisotropic rotational diffusion tensor from the rotational dynamics at short times [31, 32]. The method of Wang and Case has been extended by Roe and Cheatham [163] to calculate fully anisotropic rotational diffusion.

Simulations of neat fluids and lipid membranes [219, 221, 222] are known to suffer from large finite-size effects due to the commonly used periodic boundary conditions (PBC). This effect is caused by the severe constraints on the hydrodynamic flow due to the PBC. PBC requires that the net flux at the boundary of the system is zero. The hydrodynamic

correction is determined by the difference between the Stokes friction in an infinite system and in a system under PBC. For a particle in a neat fluid, the hydrodynamic correction predicts that the translational diffusion coefficient is underestimated. The correction depends only on the box size and shape and not on the shape or diffusion coefficient of the particle [46, 221, 236]. We expect that rotational diffusion will similarly be affected by the PBC. However, it is an open question if the effect is as strong as for translational diffusion.

glsMD simulations currently allow to study molecular systems on times-scales up to tens of μs and are limited in size to the nm range. The largest all-atom simulation today consisted of 1/10th of the volume of the *Mycoplasma Genitalium* cell contained 100 million atoms and was run for 20 ns [238]. The longest continuous simulations today reach the millisecond range [185]. However, they require specialized hardware [184, 185]. It is possible to acquire insights into dynamics on the millisecond range and longer using commodity hardware with Markov state models [101, 159, 187], master equation models [80, 194–196], and transition path sampling [80, 140]. These models require prior insight into the system and are computationally intensive. Protein-interaction can be studied on the milliseconds to second time-scale and on a μm length-scale using Brownian dynamics (BD) simulations. They allow to generate continuous trajectories that are multiple milliseconds long. BD simulations have been used to study proteins in crowded environments [43, 130, 132], to understand how shape influences diffusion [9], to simulate an accurate environment of the cytoplasm [130], to explore shape and binding site effects on the binding kinetics [139], and te determine binding rates and constants for arbitrary protein complexes [59, 120, 160, 180, 182, 239]. The Ermak-McCammon [49] algorithm is commonly used to integrate the Langevin equation for anisotropic rigid bodies. The algorithm is an extension of the Euler-Maruyama [128] algorithm to translation and rotation in three dimensions. It is known that the error of the Euler-Maruyama algorithm depends linearly on the chosen time-step [204]. It has been shown that Monte Carlo simulations generate dynamics that are similar to BD. The trial distribution is chosen carefully to reproduce BD for a given diffusion coefficient [95, 104, 169, 179]. Monte Carlo based algorithm have the advantage that they reproduce the equilibrium distribution independent of the chosen time-step. Current algorithms describe only translational motion and are therefore limited to isotropic particles. To use Monte Carlo based algorithms for anisotropic particles rotational moves have to be included as well.

BD algorithms are often used in conjunction with CG protein models. The CG models used with BD algorithms represent amino acids as single interaction site, called beads, and replace the solvent molecules with a continuum model [43, 59, 96, 130, 133]. The hydrodynamic interactions of the solvent with the simulated macromolecules are described by the diffusion coefficients. The diffusion coefficients are often predetermined using hydrodynamic theory. The theory to calculate the diffusion coefficients assumes the macromolecules are rigid and therefore the macromolecules can be treated as rigid bodies, further reducing the computational effort. It should be noted that CG protein models, e.g. MARTINI, exist that lump together multiple atoms into one interaction site but keep

the explicit solvent molecules and internal dynamics [126, 127]. In this thesis we use the Complexes protein model [96]. This model is known as the Kim-Hummer (KH) model in the literature. In this thesis we will only refer to the KH-model when referring to the specific parametrization developed by Kim and Hummer [96], for the definition of the energy function we refer to it as the Complexes protein model.

In this thesis we estimate the full anisotropic rotational diffusion coefficients from fully atomistic MD simulations, and the finite-size effects of rotational diffusion under PBC in Chapter 3. To simulate BD with accurate equilibrium distributions we will develop a novel Monte Carlo based algorithm in Chapter 4. Our algorithm uses quaternions for the rotation moves to accurately simulate rotation dynamics. The new algorithm will be implemented in a new program called Complexes++, see Chapter 5. In Complexes++ we implemented the Complexes protein model [96] together with a new algorithm for the flexible domains. Each chapter will include example applications of the methods that have been developed. Initially, we will present the simulation methods used in this thesis.

Chapter 2

Fundamentals

In this chapter we will describe the essential established concepts that lay the foundation of our work.

2.1 Forcefields

There exist many different forcefields to describe biomolecules. They vary in the level of detail, the potential energy term, and how they are parameterized. Forcefields like CHARMM36 [20, 121, 122] and Amber99sb*-idln [17, 78, 113, 225] are all-atom forcefields and provide a very high level of detail in classical simulations. However, due to the high level of detail, these forcefields are computationally expensive to evaluate. Therefore they are well suited for problems on short timescales, up to hundreds of microseconds, e.g. conformational change [156]. For protein-protein binding problems or membrane remodeling CG forcefields like the Complexes model [96] and MARTINI [126, 127] are better suited. The Complexes model is also known as the Kim-Hummer (KH) model in the literature. This model replaces amino acids with single beads centered at the C_α atom. The MARTINI model uses a 4:1 mapping where it replaces four atoms in an all-atom description with a single MARTINI atom. Even though CG forcefields use a less detailed description of the biomolecule they have a lower overall error due to better sampling of conformation space, see Figure 2.1. The choice between an all-atom and a CG forcefield depends on the biological question and the available computation time. In this work the Amber99sb*-idln and AMBER Parmbsc1 [87] forcefields for proteins and DNA, respectively, and the Complexes model have been used.

2.1.1 All-Atom Forcefields

All-atom forcefields allow an accurate description of the dynamic and thermodynamics of biomolecules using classical mechanics. Approximating the motion of nuclei of biomolecules with classical mechanics means that there is no unique forcefield. As we only use AMBER forcefields for our all-atom simulations we will explain it in more detail. The forcefields from the AMBER family [17, 35, 78, 87, 113, 225] consist of terms for bonded

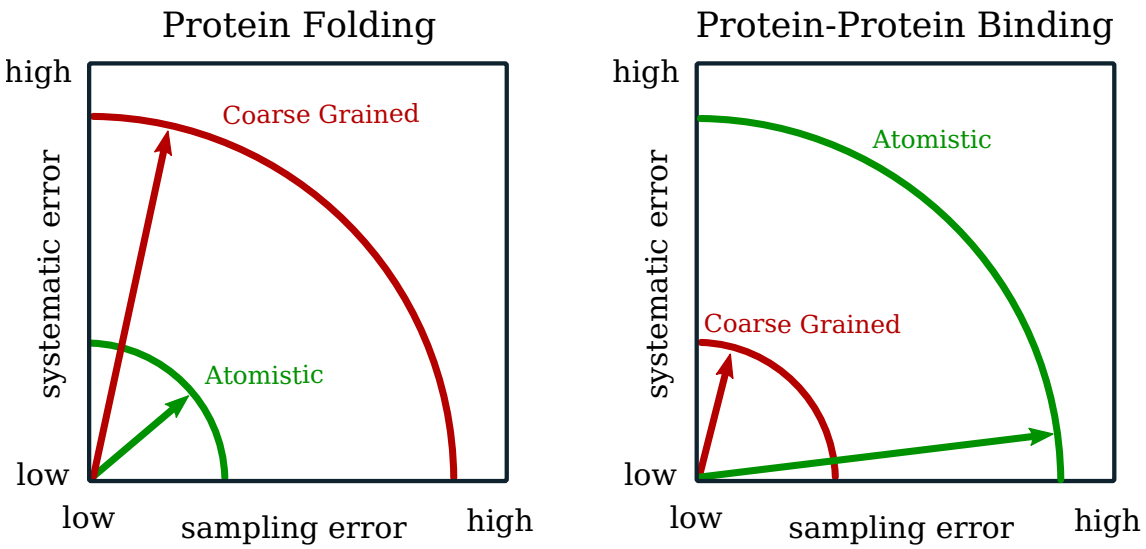

Figure 2.1: Sketch of sampling and systematic error for protein folding and protein-protein for forcefields with different accuracy. A smaller circle corresponds to a lower total error. The systematic error is dominated by the forcefield and energy surface. The sampling error is dominated by the available computation time. (Based on an image by Jürgen Köfinger.)

and pairwise non-bonded interactions [35, 225]

$$U(\vec{x}) = \underbrace{U_{\text{angles}} + U_{\text{dihedrals}} + U_{\text{bond}}}_{\text{bonded interactions}} + \underbrace{U_{\text{Lennard Jones}} + U_{\text{Coulomb}}}_{\text{nonbonded interactions}}. \tag{2.1}$$

To model the fluctuating angle θ between neighboring bonded atoms a harmonic angle potential is used

$$U_{\text{angles}} = \sum_{\text{angles}} K_\theta(\theta - \theta_{\text{eq}})^2, \tag{2.2}$$

with θ_{eq} the equilibrium angle, and K_θ spring constant.

The dihedral angle potential between four consecutive bonded atoms is described by a periodic function

$$U_{\text{dihedrals}} = \sum_{\text{dihedrals}} \frac{V_n}{2}[1 + \cos(n\phi - \gamma)]. \tag{2.3}$$

The integer n here gives the periodicity of the potential, and V_n the associated barrier height. The value of n and V_n depends on the atom types for the atoms used to calculate the dihedral. There may be more than one n, V_n pair for the same atom sequence. The results for each pair are summed up. Eq 2.3 is also used to model planar topologies using so called improper dihedrals.

Bonded interations are modeled using a harmonic spring

$$U_{\text{bonds}} = \sum_{\text{bonds}} K_r (r - r_{\text{eq}})^2, \tag{2.4}$$

with the spring constant K_r. The potentials are usually so stiff that the resulting bond vibrations are unrealistic. Bond vibrations barely couple with other degrees of freedom and are not of interest for most phenomena studied by MD simulations. Therefore the bonds are often constrained to the equilibrium bond distance in simulations.

The Lennard Jones (LJ) term consists of the attractive van der Waals and Pauli repulsion terms. The van der Waals interactions are known to scale well with r^{-6}. The Pauli repulsion are calculated as r^{-12} for computational efficiency.

$$U_{\text{Lennard Jones}} = \sum_{i<j} 4\epsilon_{ij} \left[\underbrace{\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12}}_{\text{Pauli Repulsion}} - \underbrace{\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}}_{\text{van der Waals attraction}} \right] \tag{2.5}$$

The parameters ϵ_{ij} and σ_{ij} depend on pairs of atom types. The LJ potential is fully determined by position of its minimum, located at a distance $2^{1/6}\sigma_{ij}$, and the value at the minimum $-\epsilon_{ij}$. The values for σ and ϵ are determined for single atom types during parametrization. Both are combined using arithmetic or geometric averages to calculate σ_{ij} and ϵ_{ij}. In the AMBER family forcefields the LJ interaction are only evaluated for atoms in different molecules or for atoms separated by at least three bonds in the same molecule.

The electrostatic interactions are computed using the Coulomb potential

$$U_{\text{Coulomb}} = \sum_{i<j} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}, \tag{2.6}$$

with q_i the charge and ϵ_0 the vacuum permittivity. The electrostatic interaction are long-ranged and cannot be cut-off. When using PBC the particle mesh Ewald summation [38, 50] can be used to calculate eq 2.6. The Ewald summation decomposes the summation of pairwise interactions in a periodic system into a real space and a Fourier part. The charges can be evaluated in Fourier space in a grid, resulting in computational efficiency of $\mathcal{O}(n \log n)$, with n being the total number of particles.

2.1.2 Complexes Forcefield

The Complexes model is a coarse-grained forcefield that describes proteins and large macromolecular structures at three hierarchical levels [96]. The Complexes model is known as the KH model in the literature. Beads are the first level. Beads are interaction sites that are used to evaluate potentials and represent a single amino acid, centered on the C_α atoms. The second level are domains. Domains are collections of beads that define how the bead positions are propagated in a simulation. The Complexes model has rigid and

flexible domains. The third level are topologies. A topology is a collection of connected domains. Typically rigid domains and flexible domains are connected to model a loosely connected protein complex. In this thesis, the general expression of the forcefield will be referred to as the Complexes model, when specific values for forcefield parameters are given we refer to them explicitly, e.g. as the KH-model. In the following, the bead model and pair potential for different beads and different domains will be explained.

Beads

Beads are the interaction sites at which the forcefield and additional restraint potentials are evaluated. In the Complexes model, amino acids are modeled as single beads centered on the C_α atoms. As energy function, a LJ like potential is used for effective interactions of native and non-native contacts and a Coulomb term with an exponential screening term for the electrostatics. The potential energies are by convention calculated in units of $k_\mathrm{B}T$, with $T = 300\,\mathrm{K}$ as the reference temperature.

The LJ like potential U_{LJ}, between beads i and j, consists of four different branches to model attractive and repulsive interactions

$$U_{LJ}(r, \sigma_{ij}, \epsilon_{ij}) = \begin{cases} 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] & \text{if } \epsilon_{ij} < 0 \\ 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] + 2\epsilon_{ij} & \text{if } \epsilon_{ij} > 0 \text{ and } r < 2^{1/6}\sigma_{ij} \\ -4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] & \text{if } \epsilon_{ij} > 0 \text{ and } r > 2^{1/6}\sigma_{ij} \\ .01\left(\frac{\sigma_{ij}}{r}\right)^{12} & \text{if } \epsilon_{ij} = 0, \end{cases} \tag{2.7}$$

with r the distance between the beads, the bead type pair parameters σ_{ij} and ϵ_{ij} for the contact distance and interaction energy, respectively. For $\epsilon_{ij} < 0$ this is the standard LJ potential, see Figure 2.2. For $\epsilon_{ij} > 0$ this potential is purely repulsive, see Figure 2.2. In the case of $\epsilon_{ij} = 0$ the potential is a hard wall slightly shorter than the LJ minimum of $2^{1/6}\sigma_{ij}$ to avoid overlaps if additional potentials are attractive and have singularities at $r = 0$, for example electrostatic potentials. At contact $r = \sigma_{ij}$ this potential gives equal contributions from attractive and repulsive pairs with opposite sign. The parameters ϵ_{ij} are derived from the knowledge-based statistical contact potentials e_{ij} by Miyazawa and Jernigan [136]. The contact distances σ_{ij} are determined as weighted average $\sigma_{ij} = (\sigma_i + \sigma_j)/2$ from the individual amino acids diameters, Table 2.1. Note that in the original paper these values have been incorrectly labeled as radii [96].

Table 2.1: Van-der-Waals diameters of amino acids in Å[96].

Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile
5.0	6.6	5.7	5.6	5.5	6.0	5.9	4.5	6.1	6.2

Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
6.2	6.4	6.2	6.4	5.6	5.2	5.6	6.8	6.5	5.9

The Miyazawa and Jernigan (MJ) contact potentials have to be scaled to account for

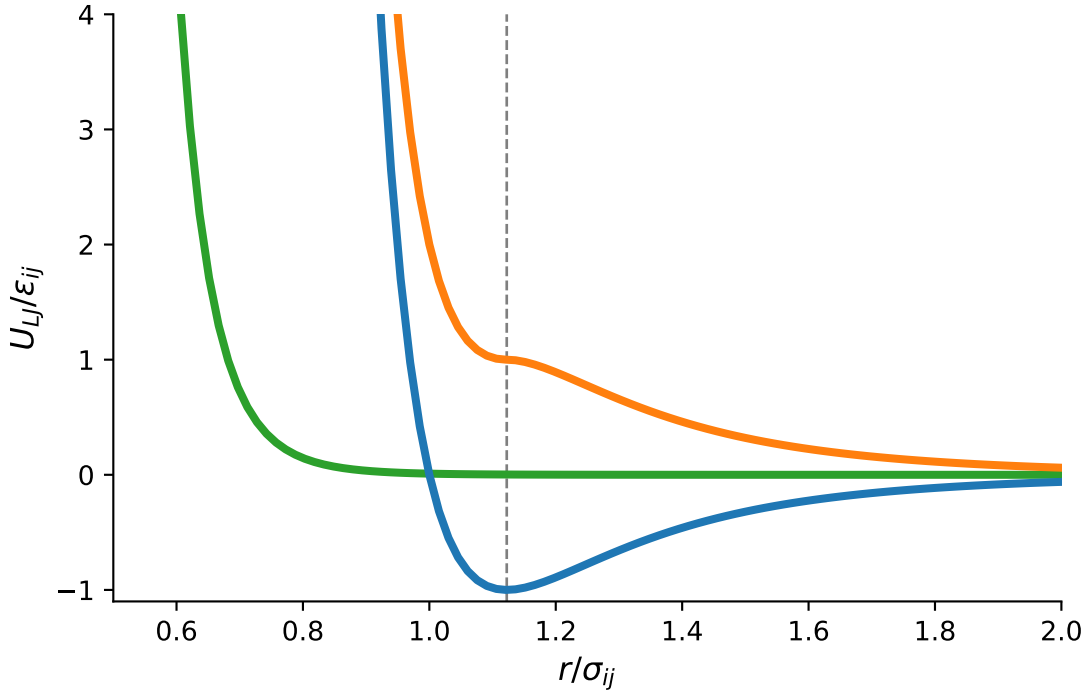

Figure 2.2: Modified LJ potential, eq 2.7, used in the Complexes model in reduced units. The attractive branch $\epsilon_{ij} < 0$ is shown in blue, the repulsive part $\epsilon_{ij} > 0$ is shown in orange and the branch for $\epsilon_{ij} = 0$ in green. The gray dashed line shows the minima at $2^{1/6}$ of the attractive branch.

the added electrostatic interactions and the preference of residue-residue to residue-solvent interactions have to be balanced. In the Complexes model, this is done by scaling with a parameter λ for the electrostatic interaction and shifting the interaction with a parameter e_0 for the residue-residue to residue-solvent interactions with

$$\epsilon_{ij} = \lambda(e_{ij} - e_0). \tag{2.8}$$

For the KH model the values $\lambda = 0.159$ and $e_0 = -2.27\,\mathrm{k_B T}$ have been used, based on parametrizations to reproduce the experimentally determined second virial coefficient of hen egg lysozyme and the dissociation constant K_d of the UIM-1-ubiquitin complex [96]. In recent years there have been efforts to optimize the ϵ_{ij} parameters to reproduce the binding affinity of a large variety of weak binding protein complexes [90].

The electrostatic potential consists of Coulomb interactions with an ionic-screening term

$$U_{EL}(r) = \underbrace{\frac{q_i q_j e^2}{4\pi\epsilon_0 D_{el} r}}_{\text{coulomb}} \underbrace{\exp\left(\frac{-r}{\xi}\right)}_{\text{ionic-screening}} \frac{1}{k_B T}, \tag{2.9}$$

with e the elementary charge, ϵ_0 the vacuum permittivity, D_{el} the dielectric constant and ξ the Debye-length. The scaling factor of $1/k_B T$ is used to convert the electrostatic energy

into units of $k_{\mathrm{B}}T$. Bead charges are set according to the amino acid type corresponding to a pH of 7. Arginine and lysine are charged with $+e$, histidine with $+\frac{1}{2}e$, due to its isoelectric point and aspartate and glutamine with $-e$. Charges for other amino acids are set to 0. The ionic-screening term is used to set the salt concentration of the environment using the Debye length

$$\xi = \sqrt{\frac{\epsilon_r \epsilon_0 k_{\mathrm{B}} T}{e^2 N_A 2I}}, \tag{2.10}$$

where ϵ_r the absolute permittivity, and I the ionic strength. For a typical salt concentration of 100 mM NaCl ξ is about 10 Å.

Domains and Topologies

Proteins and multiprotein complexes consist of multiple units that are connected and behave as a single unit for a short time period. Parts of the complex are rigid, like α-helices and β-sheets can be considered to be rigid to study protein association, while others can be unstructured, loops connecting rigid domains. Domains are used to appropriately model the different parts in a complex. To model the two different described behaviors the Complexes model has rigid and flexible domains. The domains in a complex are connected using a harmonic potential

$$U_{\mathrm{Bond}}(x) = \frac{1}{2}k(x - x_0)^2, \tag{2.11}$$

where $k = 378\,\mathrm{kcal\,mol^{-1}\,\AA^{-2}}$ [94] is the spring constant and $x_0 = 3.81\,\mathrm{\AA}$ [17] is the C_α - C_α distance. The complex that is build by connecting domains is called a topology. The total energy in the Complexes model is not a straight forward sum over all bead pairs. Instead one has to sum over the pairs of

$$U_{\mathrm{Total}} = \sum_{A,B \in \text{Domain Pairs}} \left[U_{\mathrm{Inner}} + \sum_{i \in A, j \in B} U_{LJ}(|r_i - r_j|) + U_{EL}(|r_i - r_j) \right] + \sum_{\text{connections}} U_{\mathrm{Bond}}. \tag{2.12}$$

Rigid Domain Rigid domains are the simplest form of a domain and the most versatile at the same time. As the name suggests in a rigid domain the internal coordinates of the beads in the domain do not change over time. Rigid domains are so versatile because they can be used to model very different things. The obvious cases are rigid protein parts like an α-helix or a β-sheet. The inner energy of rigid domains is zero, $U_{\mathrm{Inner}} = 0$. While not described in the original Complexes model it is possible to describe a rigid domain at an even coarser level by grouping together amino acids. Using such a CG description requires finding new forcefield parameters for the interactions but this versatility makes it possible to set up simulations that incorporate experimental data with an appropriately modeled experimental uncertainties and prior knowledge.

Flexible Domains The flexible domains are modeled as bead polymers. The inner energy consists of the LJ eq 2.7, electrostatic potential eq 2.9 between bead pairs more than three beads apart, and contributions from bond stretching potentials for pseudobonds, angle potentials for pseudoangles and torsion potentials for pseudotorsion

$$U_{\text{Inner}} = \sum_{|i-j|\geq 4} U_{LJ} + U_{EL} + \sum_{\text{bonds}} U_{\text{bond}} + \sum_{\text{angles}} U_{\text{angle}} + \sum_{\text{torsionangles}} U_{\text{torsion}}. \tag{2.13}$$

The bond potential is a harmonic potential

$$U_{\text{bond}} = \frac{1}{2}k(r - r_0)^2, \tag{2.14}$$

with r the $C_\alpha - C_\alpha$ distance, $r_0 = 3.81\,\text{Å}$ the reference distance and $k = 378\,\text{kcal}\,\text{mol}^{-1}\text{Å}^{-2}$ the spring constant [94]. The pseudoangle potentials is a double well potential [19]

$$\exp[-\gamma U_{\text{angle}}(\theta)] = \exp[-\gamma(k_\alpha(\theta - \theta_\alpha) + \epsilon_\alpha)] + \exp[-\gamma k_\beta(\theta - \theta_\beta)^2], \tag{2.15}$$

where θ is the angle between $C_\alpha - C_\alpha - C_\alpha$, the constants are $\gamma = 0.1\,\text{mol}\,\text{kcal}^{-1}$, $\epsilon_\alpha = 4.3\,\text{kcal}\,\text{mol}^{-1}$, $\theta_\alpha = 1.6\,\text{rad}$, $\theta_\beta = 2.27\,\text{rad}$, $k_\alpha = 106.4\,\text{kcal}\,\text{mol}^{-1}\text{rad}^{-2}$ and $k_\beta = 23.6\,\text{kcal}\,\text{mol}^{-1}\text{rad}^{-2}$. This potential accounts for the helical and extended pseudoangles. The torsion potential is given by [94]

$$U_{\text{torsion}}(\phi) = \sum_{n=1}^{4}[1 + \cos(n\phi - \delta_n)]V_n, \tag{2.16}$$

where ϕ is the torsion angle of the middle two beads in $C_\alpha - C_\alpha - C_\alpha - C_\alpha$. The constants δ_n and V_n are taken from Karanicolas and Brooks [94].

2.2 Molecular Dynamics

In molecular dynamics (MD) simulation the time evolution of Newton's equations of motion

$$\vec{F}(\vec{x}) = \vec{a} \cdot m = \ddot{\vec{x}} \cdot m \tag{2.17}$$

are evaluated, with m the mass, $\vec{a}$ the acceleration, $\vec{x}$ the position, and the forces $\vec{F}$ as the negative gradient of the potential energy $U(x)$

$$\vec{F}(\vec{x}) = -\nabla U(\vec{x}). \tag{2.18}$$

There exist several algorithms for the numerical integration of eq 2.17. In this work we use the leap frog integrator [73, 217]

$$\vec{x}_{i+1} = \vec{x}_i + \vec{v}_i dt + \frac{1}{2}\vec{a}_i \Delta t^2 \tag{2.19}$$

$$\vec{v}_{i+1} = \vec{v}_i + \frac{1}{2}(\vec{a}_i + \vec{a}_{i+1})\Delta t, \tag{2.20}$$

with $\vec{x}_i, \vec{v}_i, \vec{a}_i$ the coordinates, velocity and acceleration at step i, respectively. Other commonly used integrators are the Verlet [220] and Velocity-Verlet [200] integrator. All of these algorithms are symplectic integrators [212]. While this class of integrators does not preserve the Hamiltonian exactly, there exists a "shadow" Hamiltonian that is preserved and remains close to the original Hamiltonian. Here close mean that for $\Delta t \to 0$ the "shadow" Hamiltonian approaches the real Hamiltonian [60, 190, 210, 237]. Other commonly used integrators like the Runge-Kutta integrators [105, 176] do not have this property.

A straightforward integration of the Hamiltonian with eq 2.19 will yield an ensemble in the microcanonical NVE ensemble. Experimental conditions are best described by the isothermal-isobaric NPT ensemble. Therefore we need to couple our system to a heat and pressure bath using a thermostat and barostat, respectively. For a thermostat, we use the velocity re-scaling algorithm [29]. This thermostat is based on the Berendsen thermostat [16]. The Berendsen thermostat re-scales to impulses to match a target temperature T_0

$$\frac{\mathrm{d}\vec{p}_i}{\mathrm{d}t} = \sum_j^N F_{ij}\left(|\vec{q}_i - \vec{q}_j|\right) - \frac{\vec{p}_i}{\tau_T}\left[\alpha - 1\right], \tag{2.21}$$

with a scaling factor $\alpha = K_0/K$, K_0 the kinetic energy at T_0, K kinetic energy at the current temperature, and τ_T a relaxation time constant. Eq 2.21 enforces the total kinetic energy to be equal to the kinetic energy at T_0. In the velocity re-scaling algorithm α is not constant during the simulation, instead

$$\alpha = \frac{K_t}{K} \tag{2.22}$$

with K_t being chosen at each time-step from the canonical equilibrium distribution for the kinetic energy

$$P(K_t)dK_t \propto K_t^{\left(\frac{N_f}{2}-1\right)} e^{-\beta K_t} dK_t, \tag{2.23}$$

with N_f the number of degrees of freedom, and $\beta = 1/k_\mathrm{B}T$ the inverse temperature. To generate the values K_t we could either sample directly from eq 2.23 or we can evolve the kintetic energy in time using a stochastic integrator

$$dK = (K_0 - K)\frac{dt}{\tau_T} + 2\sqrt{\frac{KK_t}{N_f}}\frac{dW}{\sqrt{\tau_t}}, \tag{2.24}$$

with dW Wiener noise. During a simulation before re-scaling the velocities we calculate the current kinetic energy by evolving it in time using eq 2.24, and calculating the scaling factor α. Therefore using the velocity re-scaling algorithm we have added the current temperature as an additional degree of freedom. Other commonly used thermostats are the Andersen thermostat [6] and Nosé-Hoover chains [76, 142].

For pressure coupling, we are using the Parrinello-Rahman algorithm [151]. Like the velocity re-scaling algorithm, it adds an additional degrees of freedom, the volume and shape of the simulation box. The first algorithm to introduce the volume as a free variable was developed by Andersen [6] and later generalized to anisotropic volume fluctuations by Parrinello and Rahman [151]. The pressure tensor $\boldsymbol{P}$ in MD simulations is calculated using the second virial theorem

$$\boldsymbol{P} = \frac{2}{V}(\boldsymbol{E_{kin}} - \Xi) \tag{2.25}$$

with

$$\boldsymbol{E_{kin}} = \frac{1}{2}\sum_{i}^{N} m_i \vec{v}_i \vec{v}_i^T \text{ and} \tag{2.26}$$

$$\Xi = -\frac{1}{2}\sum_{i<j} r_{ij} F(r_{ij}). \tag{2.27}$$

The pressure depends on the pairwise distance r_{ij} and velicities $\vec{v}_i$. MD simulations often use a finite simulation box and PBC, see Section 2.5, in which the distance between two particles changes when the simulation box changes. The simulation box is described by three vectors $\vec{a}, \vec{b}, \vec{c}$, that can be written as a matrix $\boldsymbol{H} = (\vec{a}, \vec{b}, \vec{c})$. The position $\vec{r}_i$ of an atom is now written using a vector $\vec{s}_i = (\xi_i, \eta_i, \zeta_i)^T$ and $\boldsymbol{H}$ as

$$\vec{r}_i = \boldsymbol{H}\vec{s}_i = \xi_i\vec{a} + \eta_i\vec{b} + \zeta_i\vec{c} \tag{2.28}$$

with $0 \leq \xi_i, \eta_i, \zeta_i \leq 1$. The square of the pairwise distance is

$$r_{ij}^2 = (\vec{s}_i - \vec{s}_j)\,\boldsymbol{G}\,(\vec{s}_i - \vec{s}_j)\,. \tag{2.29}$$

with $\boldsymbol{G} = \boldsymbol{H}^T\boldsymbol{H}$. Therefore, changes in the simulation box $\boldsymbol{H}$ result in a change of pressure. Parrinello and Rahman [151] derived equations of motion for $\vec{s}$ and $\boldsymbol{H}$

$$\ddot{\vec{s}}_i = -\sum_{i\neq j} m_i^{-1}\frac{\dot{V}(\vec{s}_i)}{r_{ij}}(\vec{s}_i - \vec{s}_j) - \boldsymbol{G}^{-1}\dot{\boldsymbol{G}}\dot{\vec{s}}_i, \tag{2.30}$$

$$\ddot{\boldsymbol{H}} = V\boldsymbol{W}^{-1}\dot{\boldsymbol{H}}^{-1}(P - P_{\text{ref}}) \tag{2.31}$$

with P_{ref} the reference pressure, $\boldsymbol{W}$ a matrix parameter that determines the strength of the pressure coupling, and V the volume of the simulation box. This barostat modifies

the Hamiltonian

$$\mathscr{H} = E_{\text{pot}} + E_{\text{kin}} + \sum_i P_{ii}V + \sum_{ij} \frac{1}{2} W_{ij} \left(\dot{H}_{ij}\right)^2. \tag{2.32}$$

The last term constributes a constant of $9/2k_{\text{B}}T$ at equilibrium. For a large number of atoms $N \gg 1$ the Hamiltonian is equal to the enthalpy

$$\mathscr{H} \approx E_{\text{pot}} + E_{\text{kin}} + \sum_i P_{ii}V \tag{2.33}$$

$$= E + PV. \tag{2.34}$$

Therefore, the equations of motion, eq 2.30, generate an NPT ensemble.

The largest possible time-step Δt that can be chosen in a simulation depends on the fastest motion in the system. In macromolecules, the fastest motion is the vibration of atomic bonds which is on the order of femtoseconds. Typical biological processes examined with MD are on the order of nanoseconds to tens of microseconds. On those timescales, the bonds are essentially stiff. Therefore it is beneficial to constrain atomic bonds and use a larger time-step. Δt is chosen between 2 and 4 fs. SHAKE [177] was the first developed constraint algorithm for MD. It acts as an additional term on the positions in a Velocity-Verlet integrator. It was later changed to act on the velocities to be easier usable with barostats and thermostats. This algorithm is called RATTLE [7]. In this thesis, we use the LINear Constraint Solver (LINCS) algorithm [71, 72]. The LINCS algorithm is up to four times faster than SHAKE and can be easily parallelized.

2.3 Brownian Dynamics

Brownian dynamics (BD) describes the random motion of a macromolecule suspended in a liquid. BD can be described by extending Newtons equations of motion with a friction force $-\gamma\dot{x}$, to account for the jostling of the liquid particles with the protein, and a random force $R(t)$, to account for the occasional high-velocity collision [243],

$$m\ddot{x}(t) = -\nabla U(x) - \gamma\dot{x} + \sqrt{2\gamma k_{\text{B}}T}R(t). \tag{2.35}$$

Eq 2.35 is known as the Langevin equation. For proteins the friction term is often small enough to be neglected. To estimate how small the friction term needs to be we use the Fourier transform of eq 2.35

$$(m\omega - \gamma)\omega\tilde{x} = -\tilde{U}(x) + \tilde{R}. \tag{2.36}$$

Eq 2.36 shows that the friction term becomes negligible if $m\omega \gg \gamma$. To estimate when this is the case for proteins we use the Einstein relation $\gamma = \frac{k_{\text{B}}T}{D}$ [48], order of magnitude values for $k_{\text{B}}T = 10^{-21}\text{kgm}^2\text{s}^{-2}$ at room temperature, the diffusion constant of water in

water $D = 10^{-9}\mathrm{m}^2\mathrm{s}^{-1}$ [218] and the atomic mass $m = 10^{-27}\mathrm{kg}$. We find that $m\omega = \gamma$ for

$$\frac{1}{\omega} \approx 1\,\mathrm{fs}. \tag{2.37}$$

Therefore for processes that occur at a timescale $\gg 1\,\mathrm{fs}$ the friction term is negligible and we can use the overdamped Langevin equation

$$\dot{x}(t) = -\nabla U(x)\frac{D}{k_\mathrm{B}T} + \sqrt{2D}R(t). \tag{2.38}$$

The time evolution of the probability $P(x,t)$ that a particle is at position x at time t for a particle that follows eq 2.38 can be described with a Fokker-Planck equation

$$\frac{\partial}{\partial t}P(x,t) = -\frac{\partial}{\partial x}\left[-\nabla U(x)P(x,t)\right] + D\frac{\partial^2}{\partial x^2}P(x,t). \tag{2.39}$$

Eq 2.39 can only be solved for special cases. Let us consider the case of a constant drift $-\nabla U(x) = -\nu x$, with force constant ν, boundary conditions $\lim_{x\to\pm\infty} = 0$, and initial conditions $P(x,0) = \delta(x - x_0)$. The corresponding solution of eq 2.39 is

$$P(x,t) = \frac{1}{\sqrt{2Dt}}\exp\left[-\frac{(x - x_0 - \nu t)^2}{2Dt}\right]. \tag{2.40}$$

This is a moving and broadening Gaussian profile.

Simulations using eq 2.35 or eq 2.38 are also called BD simulations. In BD simulations the solute is implicitly modeled through the diffusion coefficient. Therefore BD are computationally less expensive than MD simulations. A widely used integrator for the Langevin equation is the Euler-Maruyama integrator [128]

$$x_{i+1} = x_i + F(x_i)\frac{D}{k_\mathrm{B}T}dt + \sqrt{2D\Delta t}R_t \tag{2.41}$$

with R_t being Gaussian random number of unit variance and zero mean. A recently discovered simple extension of Euler-Maruyama integrator is the BAOAB integrator [111]

$$x_{i+1} = x_i + F(x_i)\frac{D}{k_\mathrm{B}T}dt + \sqrt{2D\Delta t}\frac{1}{2}(R_t + R_{t-1}). \tag{2.42}$$

The BAOAB integrator numerically more stable than the Euler-Maruyama and reproduces the correct equilibrium distribution for many problems. Note that both integrators can be extended to N dimensions in the special case that the $N \times N$ translation diffusion tensor is diagonal.

An extension of the Euler-Maruyama integrator to 3 dimensions for rotational and translational diffusion is the Ermak-McCammon [49] integrator. The displacement vector $\Delta\vec{x}_i$ for a particle i in a system of N particles is

$$\Delta\vec{x}_i = dt\sum_{j=1}^{N}\left(\frac{\partial \boldsymbol{D}_{T,ij}}{\partial\vec{x}_j} + \frac{\boldsymbol{D}_{T,ij}}{k_\mathrm{B}T}\cdot\vec{F}_i\right) + \vec{R}_i, \tag{2.43}$$

with $\boldsymbol{D}_{T,ij}$ the 3×3 hydrodynamic diffusion tensor between particles i and j, $\vec{F}_i$ is the systematic force acting on particle i, and $\boldsymbol{R}_i$ is the random displacement vector for particle i samples at each simulation step from a Gaussian distribution of zero mean and covariance $\langle \boldsymbol{R}_i \boldsymbol{R}_j \rangle = 2\boldsymbol{D}_{T,ij}dt$ for all i, j. The rotation dynamics of particle i are described by a similar equation, in which the translational diffusion tensor is replaced by the rotational tensor and the forces are replaced by torques [59].

2.4 Monte Carlo Simulations

This introduction to Monte Carlo simulations loosely follows the derivation from "Statistical Mechanics: Theory and Molecular Simulation" by Mark Tuckerman [212].

An expectation value I of an observable $f(x)$ in equilibrium statistical mechanics is defined using ensemble averages

$$I = \int \mathrm{d}x \; f(x)\pi(x), \tag{2.44}$$

where x is a $3N$-dimensional vector, $\pi(x)$ is a probability density function, and N the number of particles. We can estimate the expectation value using

$$I_M = \frac{1}{M} \sum_{i=1}^{M} f(x_i), \tag{2.45}$$

where $x_1, \ldots, x_M$ are M random vectors sampled from $\pi(x)$. The central limit theorem guarantees that for a sufficiently large M both values are equal [212]

$$\lim_{M \to \infty} I_M = I. \tag{2.46}$$

The random vectors x_i are often not drawn directly from $\pi(x)$ because it is computationally too expensive. Two less computationally expensive alternatives to compute eq 2.44 are importance sampling and Markov chain Monte Carlo (MCMC). Here we will only explain MCMC.

In importance sampling instead of sampling from $\pi(x)$ we sample from a distribution $h(x)$ by rewriting eq 2.44 as

$$I = \int \mathrm{d}x \; \left[\frac{f(x)\pi(x)}{h(x)} \right] h(x) \tag{2.47}$$

$$= \int \mathrm{d}x \; \psi(x)h(x). \tag{2.48}$$

$h(x)$ is called the importance function. The importance function should be chosen so that it is easier to sample than $\pi(x)$.

In MCMC the random vectors x_i are generated sequentially $x_1 \to x_2 \to \cdots \to x_M$ with a rule to generate x_{i+1} from x_i. Such a sequence of vectors is called a Markov chain. Let $R(x|y)$ be the rule to generate a vector x from a given vector y. In physical systems,

$R(x|y)$ is the probability to move from a state x to a state y. For $R(x|y)$ to sample the desired equilibrium distribution $\pi(x)$ it is sufficient that the detailed balance condition

$$R(x|y)\pi(y) = R(y|x)\pi(x). \tag{2.49}$$

is fulfilled. We will later proof why this condition is sufficient to guarantee that we sample from $\pi(x)$. We now have to find expressions for $R(x|y)$. To do so, we define

$$R(x|y) = T(x|y)A(x|y), \tag{2.50}$$

with $T(x|y)$ the probability to generate a move from y to x and $A(x|y)$ the probability to accept this move. $T(x|y)$ has to fulfill the normalization condition

$$1 = \int \mathrm{d}x T(x|y) \tag{2.51}$$

and be symmetric

$$T(x|y) = T(y|x). \tag{2.52}$$

Substituting eq 2.50 into eq 2.49 we obtain

$$A(x|y) = \frac{\pi(x)}{\pi(y)} A(y|x). \tag{2.53}$$

Eq 2.53 defines the condition that $A(x|y)$ has to fulfill so that detailed balance, eq 2.49, is preserved. One solution of eq 2.53 is [134]

$$A(x|y) = \min\left[1, \frac{\pi(x)}{\pi(y)}\right] = \min\left[1, e^{-\Delta U/k_\mathrm{B}T}\right], \tag{2.54}$$

with $\pi(x) \propto e^{-U(x)/k_\mathrm{B}T}$ the Boltzmann distribution, and $\Delta U = U(x) - U(y)$ the energy difference between state x and y. Eq 2.54 is known in the literature as the Metropolis-function. The algorithm to generate the Markov chain is as follows. Generate trial move x'_{i+1} according to $T(x_{i+1}|x_i)$ and evaluate $e^{-(U(x'_{i+1})-U(x_i))/k_\mathrm{B}T}$. If $e^{-(U(x'_{i+1})-U(x_i))/k_\mathrm{B}T} \geq 1$ than the move is accepted with probability 1. If, however $e^{-(U(x'_{i+1})-U(x_i))/k_\mathrm{B}T} < 1$ a random number $\xi \in [0, 1)$ is drawn. If $e^{-(U(x'_{i+1})-U(x_i))/k_\mathrm{B}T} > \xi$ the move is accepted otherwise it is rejected. Note that when a move is rejected than $x_{i+1} = x_i$ and the old state is recorded again in the Markov chain. The Metropolis algorithm was later generalized by Hastings for none symmetric trial probabilities $T(x|y) \neq T(y|x)$ [69].

Every point x_i in the Markov chain has a probability π_i associated with it. We will use eq 2.49 to show that

$$\lim_{i\to\infty} \pi_i(x) = \pi(x). \tag{2.55}$$

We are using a recursive proof that if $\pi_i(x) = \pi(x)$ for one i this is also true for π_{i+1}. π_{i+1}

is a combination of the probability to make a step from any y to x and the probability to have a step to any y from x rejected

$$\pi_{i+1}(x) = \int \mathrm{d}y\ A(x|y)T(x|y)\pi_i(x) + \pi_i(x) \int \mathrm{d}y\ [1 - A(y|x)]T(y|x). \tag{2.56}$$

Assuming that there is at least one i for which $\pi(x) = \pi_i(x)$ we show that $\pi(x)$ is a fixed point of eq 2.56

$$\pi_{i+1}(x) = \int \mathrm{d}y\ A(x|y)T(x|y)\pi(x) + \pi(x) \int \mathrm{d}y[1 - A(y|x)]T(y|x) \tag{2.57}$$

$$= \int \mathrm{d}y\ \left[\pi(x)T(y|x) + \underbrace{A(x|y)T(x|y)\pi(x) - A(y|x)T(y|x)\pi(y)}_{=0}\right] \tag{2.58}$$

$$= \pi(x) \int \mathrm{d}y\ T(y|x) \tag{2.59}$$

$$= \pi(x) \tag{2.60}$$

For the third equality we used eqs 2.49 and 2.50, and for the fourth we used eq 2.51. Therefore random vectors from the Markov chain $x_1 \to x_2 \to \cdots \to x_n$ are sampled from $\pi(x)$.

Monte Carlo simulations do not naturally have a time associated with the generated configurations. By convention, the "time" in Monte Carlo simulations is counted in units sweeps. A sweep consists of N trial moves if N is the number of particles in the simulation. The N particles that are moved are chosen at random, therefore a single particle can be moved multiple times during a sweep.

2.5 Periodic Boundary Conditions

Simulations of biomolecules are often done using periodic boundary conditions (PBC) [55, 212]. Under PBC all atoms are placed inside of a unit cell, which is surrounded by translated, and space-filling copies of itself. These copies are commonly known as periodic images. If an atom crosses the boundary it is replaced by an equivalent atom from a periodic image on the opposite boundary, see Figure 2.3. The unit cell of the simulation can either be a cuboid, truncated octahedron,or a rhombic dodecahedron. Cuboid unit cells are easy to implement efficiently but the rhombic dodecahedron only has 71 % of the volume of a cube with the same image distance. Saving about 29% of computation time when simulating spherical molecules with explicit solvent. The distance between any two atoms is determined using the minimum image convention (MIC). The MIC states that the distance between any two atoms is the minimum distance between any of the neighboring images, Figure 2.3. PBC has known finite size effects for some hydrodynamic observables like the translational and rotational diffusion coefficient, that can be minimized by enlarging the box [46, 115, 221, 222, 236].

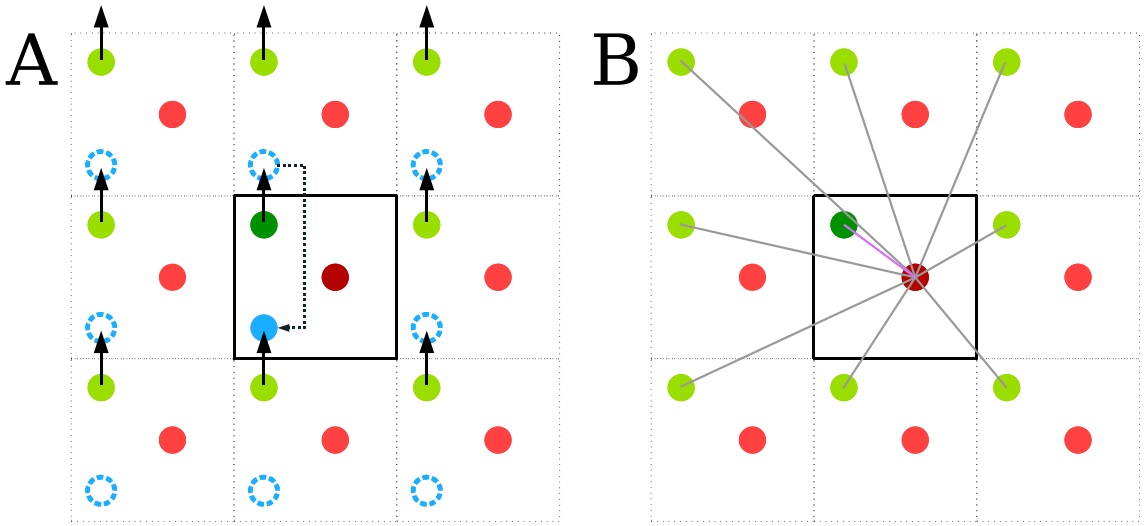

Figure 2.3: Simulation box containing two atoms (red and green) is shown in the center with its eight periodic images (lighter shades) in 2D. A) The green atom is moved across the box boundary and replaced by the atom from the periodic image below the simulation box. The new position of the atom is shown in blue outline for the periodic images and a solid circle for the new position in the simulation box. B) All distances between green and red atom and the respective neighboring images. The purple line is the minimum image distance. Image adapted from wikipedia and licensed under CC share alike.

2.6 Replica Exchange Simulations

Single trajectories often do require a long time to sample a large conformation space. To sample as much of conformation space as possible practitioners often employ enhanced sampling schemes. The simplest scheme is to run multiple simulations in parallel with identical initial configurations but different initial conditions, e.g., different seeds for the random number generator (RNG). Because the simulation run in parallel the set of simulation covers a larger amount of phase space in the same time compared to a single simulation. In the last decades many more sophisticated enhanced sampling schemes have been developed. In this thesis we will use Replica Exchange Monte Carlo (REMC) [15, 197, 199, 212]. Other popular schemes include metadynamics [106], generalized simulated annealing [211], and Bayesian inference of ensembles [82].

Replica exchange simulations have been shown to enhance sampling in biological systems [150, 170]. In replica exchange simulations M independent simulations of a system, which are further referred to as replicas, are run simultaneously and coordinates between systems are exchanged periodically. The replicas are identical except for a physical control variable like the temperature or pressure. The assumption of replica exchange is that by tuning the physical control parameter high energy barriers can be easier overcome in some replicas (e.g. high temperatures) and through the periodic exchange also other replicas (e.g. low temperatures) are able to sample states with high energy barriers between them. For example, if the temperature is the control parameter and T_1 is the temperature at

which the equilibrium distribution is to be sampled than higher temperature replicas can easily cross barriers in the potential energy landscape. The low-temperature replicas in Figure 2.4 only sample the energy minima, while the high-temperature replicas are able to cross barriers and explore the full phase space.

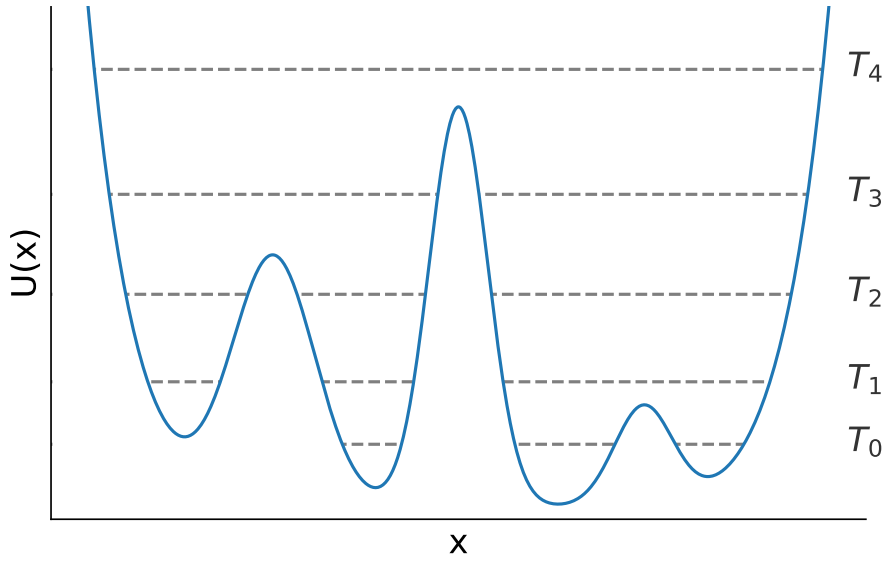

Figure 2.4: Schematic of the different energy levels accessible at different temperature using REMC. The temperatures in the schematic are increasing $T_i < T_{i+1}$. At low temperatures the simulations are confined to the minima, and at higher temperatures the full phase space can be explored.

In a replica exchange simulation, the individual replicas are propagated in either with an MD or a Monte Carlo algorithm. Periodically a pair of replicas is selected and a move is attempted. To derive the trial move and acceptance function for temperature replica exchange let M be the number of replicas with temperatures $T_M > T_{M-1} > \cdots > T_1$, and $r_{(1)}, \ldots, r_{(M)}$ the configurations of the replicas. The probability for a configuration r_i is

$$f_j\left(r_i\right) = \frac{\exp\left(-\beta_i U(r_i)\right)}{Q(N, V, T_i)}, \tag{2.61}$$

where β_i is the inverse temperature of replica i, Q is the partition function, N the number of degrees of freedom and V the volume. During an exchange attempt between replicas i and j the positions are exchanged $(r_i, r_j) \to (\tilde{r}_i, \tilde{r}_j)$, where $\tilde{r}_i = r_j$ and $\tilde{r}_j = r_i$. This move is symmetric

$$T(r_i|r_j) = T(r_j|r_j). \tag{2.62}$$

Therefore the acceptance function becomes [66]

$$A(r_i|r_j) = \min\left[1, \frac{f_i(r_j)f_j(r_i)}{f_i(r_i)f_j(r_j)}\right] \tag{2.63}$$

$$= \min\left[1, e^{(\beta_i-\beta_j)[U(r_i)-U(r_j)]}\right]. \tag{2.64}$$

Besides temperature replica exchange more replica exchange protocols have been developed. Two commonly used once are pressure replica exchange [144] and Hamiltonian replica exchange [28]. For pressure replica exchange the acceptance function is

$$A(r_i|r_j) = \min\left(1, \exp\left((\beta_i-\beta_j)(U(r_i)-U(r_j)) + (\beta_i P_i - \beta_j P_j)(V_i - V_j)\right)\right), \tag{2.65}$$

with P_i the pressure of replica i and V_i the volume of replica i. For Hamiltonian replica exchange the acceptance function is

$$A(r_i|r_j) = \min\left(1, \exp\left(\beta_i(U_i(r_i) - U_i(r_j)) + \beta_j(U_j(r_j) - U_j(r_i))\right)\right), \tag{2.66}$$

with U_i the energy function of replica i. In Hamiltonian replica exchange the potential energy function is changed from U_A to U_B using a one dimensional scaling factor λ, for example

$$U_i(r) = \lambda_i U_A(r) + (1-\lambda_i)U_B(r). \tag{2.67}$$

The descriptions of the replica exchange algorithms do not specify which replicas are exchanged in an attempt. To increase the probability to accept an exchange we only attempt to exchange neighboring replicas [28]. An alternative would be to randomly select a pair i, j, see Figure 2.5 for how configurations are exchanged between replicas in both schemes. During an exchange, not all neighboring pairs are attempted to exchange, this is because of the exchange between replica i and $i+1$ also depends on replica $i-1$. Therefore attempts are only done between even pairs when the attempt number is even and odd pairs otherwise. A pair $i, i+1$ is called even/odd if i is even and vice versa.

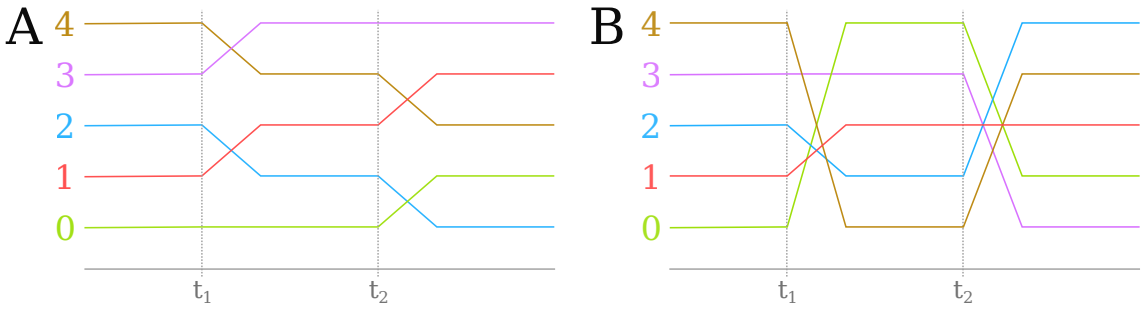

Figure 2.5: Two different exchange protocols for replica exchange simulations. A shows the odd/even pair protocol, where only neighboring replicas are exchanged. B shows the all to all protocol, where exchanges are attempted between random replicas. t_1 and t_2 are exchange attempts.

2.7 Rotational Dynamics

Based on the work of Fury [57] explicit expressions for the rotational dynamics of rigid bodies using Cayley-Klein parameters [64] have been derived by Favro [51]. In this thesis we are describing the rotational motion of a rigid body using quaternions or, more precisely, column vectors of the so-called Euler parameters [64, 119], $\boldsymbol{q} = (q_0, q_1, q_2, q_3)^T$, with $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$ and T indicating the matrix transpose. Quaternions are closely related to Caley-Klein parameters and allow use to directly copy the expressions derived by Favro. The quaternion coefficients are related to the rotation axes $\vec{v}$ and angle ϕ through $\boldsymbol{q} = (\cos(\phi/2), \sin(\phi/2)\vec{v})$. A quaternion $\boldsymbol{q} = (1, \epsilon, 0, 0)$ with $\epsilon \to 0$ therefore corresponds to an infinitesimal rotation about the x axis by an angle $\epsilon/2$. Quaternions can be directly converted into rotation matrices [64]

$$\boldsymbol{Q} = \begin{bmatrix} 1 - 2({q_2}^2 + {q_3}^2) & 2(q_0 q_3 - q_1 q_2) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & 1 - 2({q_1}^2 + {q_3}^2) & 2(q_0 q_1 - q_2 q_3) \\ 2(q_0 q_2 - q_1 q_3) & 2(q_2 q_3 + q_0 q_1) & 1 - 2({q_1}^2 + {q_2}^2) \end{bmatrix}. \tag{2.68}$$

The rotational dynamics of a freely rotating rigid body are completely determined by its rotational diffusion tensor. Let the quaternion $\boldsymbol{q}(t)$ describe the orientation of the rigid body in the principal coordinate system (PCS) as a function of time t. In the PCS, the rotational diffusion tensor is diagonal, $\boldsymbol{D_R} = \mathrm{diag}(D_{R,1}, D_{R,2}, D_{R,3})$. For rotational diffusion of a rigid body that is aligned with the PCS at time 0, $\boldsymbol{q}(0) = (1, 0, 0, 0)^T$, Favro [51] derived explicit expressions for the average quaternion covariance coefficients $\langle q_i(t) q_j(t) \rangle$ as a function of time,

$$\langle q_0^2(t) \rangle = \frac{1}{4}\left(1 + e^{-3D_R t}\left(e^{D_{R,1}t} + e^{D_{R,2}t} + e^{D_{R,3}t}\right)\right) \tag{2.69}$$

$$\langle q_1^2(t) \rangle = \frac{1}{4}\left(1 + e^{-3D_R t}\left(e^{D_{R,1}t} - e^{D_{R,3}t} - e^{D_{R,2}t}\right)\right) \tag{2.70}$$

$$\langle q_2^2(t) \rangle = \frac{1}{4}\left(1 + e^{-3D_R t}\left(e^{D_{R,2}t} - e^{D_{R,1}t} - e^{D_{R,3}t}\right)\right) \tag{2.71}$$

$$\langle q_3^2(t) \rangle = \frac{1}{4}\left(1 + e^{-3D_R t}\left(e^{D_{R,3}t} - e^{D_{R,2}t} - e^{D_{R,1}t}\right)\right) \tag{2.72}$$

$$\langle q_i(t) q_j(t) \rangle = 0 \quad \text{for } i \neq j, \tag{2.73}$$

where $D_R = \mathrm{Tr}\boldsymbol{D_R}/3 = (D_{R,1} + D_{R,2} + D_{R,3})/3$. The average $\langle \cdots \rangle$ is over repeated initializations of the stochastic trajectories or, equivalently, different starting points on a long equilibrium trajectory, appropriately rotated into the PCS. Note that Favro derived eqs 2.69 to 2.73 using the Caley-Klein parameters, which have a close relationship to quaternions. Therefore we adopted Favro's results to quaternions.

In the theoretical description of experiments the rotation correlation functions $C_i(t) = \langle P_i(\vec{v}(t) \cdot \vec{v}(0)) \rangle = \langle P_i(\cos(\theta(t))) \rangle$ are commonly used, which can be calculated from eqs 2.69 to 2.73. Here P_i is the Legendre polynomial of order i, $\vec{v}(t) = (v_1(t), v_2(t), v_3(t))^T$ is a unit vector, $\vec{v} \cdot \vec{v} = 1$, associated with a rigid molecule as a function of time t. The dot product

$\vec{v}(t) \cdot \vec{v}(0) = \vec{v}^T(t)\vec{v}(0)$ is the cosine of the angle this vector has rotated as a function of t. For P_1 and P_2 analytical results are known by the work of Favro [51] and Woessner [229].

$$\langle P_1(\vec{v}(t) \cdot \vec{v}(0))\rangle = \left(v_1^2 e^{D_{R,1}t} + v_2^2 e^{D_{R,2}t} + v_3^2 e^{D_{R,3}t}\right) e^{-3D_R t} \tag{2.74}$$

and

$$\begin{aligned}\langle P_2(\vec{v}(t) \cdot \vec{v}(0))\rangle = 3 \sum_i v_{i+1}^2 v_{i+2}^2 e^{-3(D_R + D_{R,i})t} \\ + \left(1 - 3(v_1^2 v_2^2 + v_2^2 v_3^2 + v_3^2 v_1^2)\right) e^{-6D_R t} \cosh(2t\Delta) \\ - \frac{\sum_i D_{R,i}\left(1 - 3v_i^4 - 6v_{i+1}^2 v_{i+2}^2\right) e^{-6D_R t} \sinh(2t\Delta)}{2\Delta},\end{aligned} \tag{2.75}$$

with $D_{R,i}, i \in [1,2,3]$ the elements of the diagonal rotation diffusion tensor,

$$D_R = \frac{1}{3}(D_{R,1} + D_{R,2} + D_{R,3}), \text{ and} \tag{2.76}$$

$$\Delta = \left(D_{R,1}^2 + D_{R,2}^2 + D_{R,3}^2 - D_{R,1}D_{R,2} - D_{R,1}D_{R,3} - D_{R,2}D_{R,3}\right)^{1/2}. \tag{2.77}$$

For an isotropic molecule, $D_{R,1} = D_{R,2} = D_{R,3} = D_R$, eqs 2.74 and 2.75 simplify to

$$\langle P_1(\vec{v}(t) \cdot \vec{v}(0))\rangle = \exp(-2D_R t), \tag{2.78}$$

and

$$\langle P_2(\vec{v}(t) \cdot \vec{v}(0))\rangle = \exp(-6D_R t), \tag{2.79}$$

respectively. Let $\boldsymbol{U}(t)$ be the rotation matrix that describes the orientation of the molecule, with $\boldsymbol{U}(0) = \boldsymbol{I}$, and $\vec{v}(t)$ a unit vector attached to the rigid molecule. Then $\vec{v}(t) = \boldsymbol{U}(t)\vec{v}(0)$ and the expectation value becomes

$$\langle P_1(\vec{v}(t) \cdot \vec{v}(0))\rangle = \vec{v}^T(0)\langle \boldsymbol{U}(t)\rangle \vec{v}(0). \tag{2.80}$$

If one uses polar angles to represent the unit vector, $v_1 = \cos\phi \sin\theta$, $v_2 = \sin\phi \sin\theta$, and $v_3 = \cos\theta$, and eq 2.68 to write the rotation matrix in terms of quaternions. By averaging uniformly over all orientations of $\vec{v}(0)$, using $u_0^2 + u_1^2 + u_2^2 + u_3^2 = 1$ and eq 2.69, one finds

$$\begin{aligned}\langle\langle P_1(\vec{v}(t) \cdot \vec{v}(0))\rangle\rangle &= \frac{1}{4\pi}\int_0^\pi \sin\theta \, \mathrm{d}\theta \int_0^{2\pi} \mathrm{d}\phi \, \vec{v}^T(0)\langle \boldsymbol{U}(t)\rangle \vec{v}(0) \\ &= \left\langle 1 - \frac{4}{3}(u_1^2 + u_2^2 + u_3^2)\right\rangle \\ &= \left\langle \frac{1}{3}(4u_0^2(t) - 1)\right\rangle \\ &= \frac{1}{3}\left(e^{D_1 t} + e^{D_2 t} + e^{D_3 t}\right) e^{-3Dt}\end{aligned} \tag{2.81}$$

where $\langle\langle \ldots \rangle\rangle$ indicates an average over the orientation of $\vec{v}(0)$ in the reference frame of the

molecule and over repeated Brownian dynamics trajectories. For an isotropic molecule, $D_1 = D_2 = D_3 = D$, eq 2.81 yields the expected single-exponential decay eq 2.78. Integrating eq 2.81, we obtain

$$\begin{aligned}\tau_1 &= \int_0^\infty \mathrm{d}t \langle P_1(\vec{v}(t)\cdot\vec{v}(0))\rangle = \int_0^\infty \mathrm{d}t \left\langle 4u_0^2(t) - 1 \right\rangle /3 \\ &= \frac{1}{3}\left(\frac{1}{D_{R,1}+D_{R,2}} + \frac{1}{D_{R,2}+D_{R,3}} + \frac{1}{D_{R,3}+D_{R,1}}\right).\end{aligned} \tag{2.82}$$

For the second-order correlation function $P_2(\cos\theta)$ averaged isotropically over orientations of $\vec{v}$, one finds

$$\begin{aligned}\langle\langle P_2(\vec{v}(t)\cdot\vec{v}(0))\rangle\rangle &= \frac{1}{5}\left\langle 16u_0^4(t) - 12u_0^2(t) + 1\right\rangle \\ &= \frac{1}{5}\Big(e^{-3(D+D_1)t} + e^{-3(D+D_2)t} \\ &\quad + e^{-3(D+D_3)t} + 2e^{-6Dt}\cosh(2t\Delta)\Big).\end{aligned} \tag{2.83}$$

using eqs 2.69 to 2.73, 3.13 and 3.14. For an isotropic molecule one recovers the expected single-exponential decay eq 2.79. Integrating eq 2.83, we obtain the rotational correlation time

$$\begin{aligned}\tau_2 &= \int_0^\infty \mathrm{d}t \,\langle\langle P_2(\vec{v}(t)\cdot\vec{v}(0))\rangle\rangle \\ &= \frac{1}{15}\left(\frac{1}{D+D_1} + \frac{1}{D+D_2} + \frac{1}{D+D_3}\right. \\ &\quad \left. + \frac{D_1+D_2+D_3}{D_1D_2+D_2D_3+D_3D_1}\right)\end{aligned} \tag{2.84}$$

for an arbitrary rotation diffusion tensor. In the limit of isotropic diffusion eq 2.79 can be used to estimate the hydrodynamic radius using

$$R_h = (3k_\mathrm{B}T\tau_2/4\pi\eta)^{1/3}, \tag{2.85}$$

which follows from $\tau_2 = 1/(6D)$ and $D = k_\mathrm{B}T(8\pi\eta {R_h}^3)^{-1}$, with η the viscosity.

In experiments, the rotational correlation time is often reported in terms of the harmonic mean relaxation time [209]

$$\tau_c = \frac{1}{6D} = \frac{1}{2(D_1+D_2+D_3)}. \tag{2.86}$$

To first order in the differences between the D_i and $D = \mathrm{Tr}\boldsymbol{D}/3$, τ_c agrees with the correlation time τ_2 of $P_2(\vec{v}^T(t)\cdot\vec{v}(0))$, eq 2.84.

Chapter 3

Accurate Calculation of Rotation Dynamics

3.1 Introduction

In this chapter, we will develop a novel algorithm to calculate fully anisotropic rotational diffusion tensors from simulations. The algorithm is based on fitting the time-dependent covariances (eqs 2.69 to 2.73) that fully describe the rotational motion of a free rigid body. The quaternion covariances can be directly calculated from simulations. The rotational diffusion tensor is determined by a fit to the time-dependent quaternion covariances, or directly by Laplace transformation and matrix diagonalization. To quantify uncertainties in the fit, we derive analytical expressions for the uncertainties and compare them with Brownian dynamics simulations of anisotropic rotational diffusion. Using our algorithm we also show that rotational diffusion depends on simulation box size when using PBC similar to translational diffusion (eq 3.23). We remove the box-size dependence of the rotational diffusion coefficients by adding a hydrodynamic correction term $k_{\mathrm{B}}T/6\eta V$ to the apparent diffusion coefficient in the periodic box. We test the fitting algorithm and finite-size correction on all-atom MD simulations of horse heart myoglobin and a B-DNA dodecamer at various box sizes.

3.2 Theory

3.2.1 Quaternion Covariance in a Reference Coordinate System.

The quaternion covariances from Favro (eqs 2.69 to 2.73) are only valid in the PCS of the rotational diffusion tensor. The PCS is not commonly known. For a quaternion $\boldsymbol{u}(t)$ that describes the orientation in an arbitrary reference coordinate system (RCS) the covariance matrix $[\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle]_{ij} = \langle u_i(t)u_j(t)\rangle$ is in general not diagonal. However there exists a transformation $\boldsymbol{V}$

$$\boldsymbol{q}(t) = \boldsymbol{V}\boldsymbol{u}(t), \tag{3.1}$$

to rotate the RCS into the PCS, so that the covariance matrix of $\boldsymbol{q}(t)$ is diagonal. Here, $\boldsymbol{V}\boldsymbol{u}(t)$ is the matrix product of a 4×4 matrix representing a member of the group SO(4)

with a 4-component vector. Because $\boldsymbol{V}$ corresponds to a rotation in three dimensions, it can be written as

$$\boldsymbol{V} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & & & \\ 0 & & \boldsymbol{R} & \\ 0 & & & \end{bmatrix}, \tag{3.2}$$

where $\boldsymbol{R}$ is a 3×3 rotation matrix representing an element of the group SO(3). Both the $\boldsymbol{V}$ and $\boldsymbol{R}$ matrices are orthogonal with determinant one,

$$\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^T = \mathbf{1} \text{ and} \tag{3.3}$$

$$\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}, \tag{3.4}$$

with $\mathbf{1}$ and $\boldsymbol{I}$ the 4×4 and 3×3 identity matrices, respectively. Using eq 3.1, the covariance matrix coefficients can be written as

$$\langle \boldsymbol{q}(t)\boldsymbol{q}^T(t)\rangle = \boldsymbol{V}\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle \boldsymbol{V}^T. \tag{3.5}$$

$\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle$ is symmetric by construction, $\langle \boldsymbol{q}(t)\boldsymbol{q}^T(t)\rangle$ is diagonal according to eqs 2.69 to 2.73, and $\boldsymbol{V}$ is an element of SO(4) by definition. Therefore, $\boldsymbol{V}^T$ is the matrix of eigenvectors of $\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle$, *i.e.*,

$$\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle \boldsymbol{V}^T = \boldsymbol{V}^T\boldsymbol{\Lambda}(t) \tag{3.6}$$

with $\boldsymbol{\Lambda}(t) = \mathrm{diag}[\lambda_0(t), \lambda_1(t), \lambda_2(t), \lambda_3(t)] = \langle \boldsymbol{q}(t)\boldsymbol{q}^T(t)\rangle$ the diagonal matrix of corresponding eigenvalues. The transformation $\boldsymbol{V}$ thus rotates the RCS into the PCS.

For an ideal Brownian rotor, for which eqs 2.69 to 2.73 hold exactly in the PCS, a matrix of type $\boldsymbol{V}$ is expected to diagonalizes the full covariance matrix. By contrast, if Brownian rotational diffusion is only an approximation to a more complex dynamics, or simply because of sampling noise, the $\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle$ matrices require the more general form of an SO(4) transformation for full diagonalization. To limit the transformation to SO(3) rotations, we impose the form of $\boldsymbol{V}$ according to eq 3.2 and diagonalize only the 3×3 covariance matrix $\boldsymbol{C}(t)$ with elements $C_{ij}(t) = \langle u_i(t)u_j(t)\rangle$ $(i = 1, 2, 3)$, *i.e.*, ignoring the

elements of the covariance matrix involving the "privileged" $u_0(t)$. Obtaining

$$\boldsymbol{V}\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle \boldsymbol{V}^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & & & \\ 0 & & \boldsymbol{R} & \\ 0 & & & \end{bmatrix} \begin{bmatrix} C_{00}(t) & \boldsymbol{c}_0(t)^T \\ \boldsymbol{c}_0(t) & \boldsymbol{C}(t) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & & & \\ 0 & & \boldsymbol{R}^T & \\ 0 & & & \end{bmatrix} \tag{3.7}$$

$$= \begin{bmatrix} C_{00}(t) & \boldsymbol{c}_0^T(t)\boldsymbol{R}^T \\ \boldsymbol{R}\boldsymbol{c}_0(t) & \boldsymbol{R}\boldsymbol{C}(t)\boldsymbol{R}^T \end{bmatrix},$$

where $\boldsymbol{R}^T$ is the matrix of orthonormal eigenvectors of $\boldsymbol{C}(t)$. For ideal Brownian rotations, the eigenvalues follow eqs 2.70 to 2.72, and $\boldsymbol{R}$ will be independent of time t. Since $\boldsymbol{R}^T$ rotates a vector from the PCS into the RCS,

$$\boldsymbol{R^T}\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{R^T}\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \text{ and } \boldsymbol{R^T}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

are the principal axes in the RCS. Therefore, the row vectors in the matrix $\boldsymbol{R}$ of eigenvectors of $\boldsymbol{C}(t)$ give the principal axes.

3.2.2 Calculating Rotational Covariance Matrix from Simulation Trajectories.

Common tools for superposition of two molecular structures provide rotation matrices. Therefore, it is convenient to express the coefficients $u_i(t)u_j(t)$ of the covariance matrix $\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle$ in terms of a rotation matrix. Let $\boldsymbol{U}$ be the root-mean-square distance (RMSD) superposition matrix that rotates the macromolecule of interest into the (predefined) reference orientation. By inverting the relationship, eq 2.68, between quaternions

and rotation matrices we arrive at explicit expressions for the covariance coefficients.

$$\begin{aligned}
u_0u_0 &= (1 + U_{11} + U_{22} + U_{33})/4 \\
u_0u_1 &= (U_{23} - U_{32})/4 \\
u_0u_2 &= (U_{31} - U_{13})/4 \\
u_0u_3 &= (U_{12} - U_{21})/4 \\
u_1u_1 &= (1 + U_{11} - U_{22} - U_{33})/4 \\
u_1u_2 &= (U_{12} + U_{21})/4 \\
u_1u_3 &= (U_{13} + U_{31})/4 \\
u_2u_2 &= (1 - U_{11} + U_{22} - U_{33})/4 \\
u_2u_3 &= (U_{23} + U_{32})/4 \\
u_3u_3 &= (1 - U_{11} - U_{22} + U_{33})/4.
\end{aligned} \tag{3.8}$$

The remaining coefficients follow from the symmetry of the covariance matrix. Note we have to use the transpose of $\boldsymbol{Q}$ in eq 2.68 due to the definition of $\boldsymbol{R}$ in eq 3.1.

We calculate $\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle$ from a single long MD trajectory by calculating all orientations of the trajectory with respect to a reference structure and averaging the rotations. To obtain the average covariance matrix $\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle$ we have to determine the orientation with respect to a structure at time τ that is different from the initial orientation (*i.e.*, into the RCS). Let $\boldsymbol{S}(\tau + t)$ be the 3×3 rotation matrix that rotates the macromolecular structure at time $\tau + t$ into optimal superposition with the reference. $\boldsymbol{S}(\tau + t)$ can be broken up into a rotation $\boldsymbol{S}(\tau)$ to move to the RCS at time τ and another rotation $\boldsymbol{U}(t;\tau)$ to then establish optimal superposition at time $t + \tau$,

$$\boldsymbol{S}(\tau + t) = \boldsymbol{U}(t;\tau)\boldsymbol{S}(\tau). \tag{3.9}$$

Because $\boldsymbol{S}\boldsymbol{S}^T = \boldsymbol{S}^T\boldsymbol{S} = \boldsymbol{I}$, we can solve for the rotation matrix $\boldsymbol{U}(t;\tau)$ describing the orientation of the molecule at time $\tau + t$ with $t \geq 0$, starting in the RCS at time τ,

$$\boldsymbol{U}(t;\tau) = \boldsymbol{S}(\tau + t)\boldsymbol{S}^T(\tau). \tag{3.10}$$

The sums of coefficients $U_{ij}(t;\tau)$ in eq 3.8 provide the coefficients $u_i(t;\tau)u_j(t;\tau)$ at time t for the trajectory segment starting at time τ. The average time-dependent covariance matrix at time t is obtained from a trajectory by repeating this procedure for all N_τ starting times τ and averaging over them,

$$\langle u_i(t)u_j(t)\rangle = \frac{1}{N_\tau}\sum_{\tau}^{N_\tau} u_i(t;\tau)u_j(t;\tau), \tag{3.11}$$

where N_τ is the number of snapshots up to time τ.

3.2.3 Estimating the Rotational Diffusion Tensor from Trajectories

To obtain the optimal rotational diffusion tensor we use a least-squares fit to the covariance matrix $\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle$ obtained from simulations. The statistical variances $s_{ij}^2(t)$ of $\langle \boldsymbol{u}(t)\boldsymbol{u}^T(t)\rangle$ are given by

$$s_{ij}^2(t) = \langle u_i^2(t)u_j^2(t)\rangle - \langle u_i^2(t)\rangle\langle u_j^2(t)\rangle. \tag{3.12}$$

Estimates of the fourth moments $\langle u_i^2(t)u_j^2(t)\rangle$ from trajectory data are problematic because they are themselves subject to large uncertainties. Therefore, we replace the fourth moments with analytic expressions $\langle u_i^2(t)u_j^2(t)\rangle_{\mathrm{id}}$ for ideal rotational diffusion with given diffusion coefficients and PCS. For the fourth-order correlations, Favro [51] obtained

$$\begin{aligned}\langle q_1^4(t)\rangle = \frac{1}{8}\Bigg(1 &+ \frac{3}{2}e^{-3D_R t}\left(e^{D_{R,1}t} - e^{D_{R,2}t} - e^{D_{R,3}t}\right)\\ &+ \frac{1}{2}e^{-3D_R t}\left(e^{-3D_{R,1}t} - e^{-3D_{R,2}t} - e^{-3D_{R,3}t}\right)\\ &+ e^{-6D_R t}\cosh(2t\Delta)\Bigg)\end{aligned} \tag{3.13}$$

and

$$\begin{aligned}\langle q_1^2(t)q_2^2(t)\rangle = \frac{1}{8}\Bigg(\frac{1}{3} &- \frac{1}{2}e^{-3D_R t}\left(e^{D_{R,3}t} - e^{-3D_{R,3}t}\right)\\ &+ e^{-6D_R t}\Big(\Delta^{-1}(D_{R,3} - D_R)\sinh(2t\Delta)\\ &- \frac{1}{3}\cosh(2t\Delta)\Big)\Bigg),\end{aligned} \tag{3.14}$$

where

$$\Delta = \left(D_{R,1}^2 + D_{R,2}^2 + D_{R,3}^2 - D_{R,1}D_{R,2} - D_{R,1}D_{R,3} - D_{R,2}D_{R,3}\right)^{1/2}. \tag{3.15}$$

The other $\langle q_i^2(t)q_j^2(t)\rangle$ can be found by permutations of the indices. Averages over products of odd powers of the q_i vanish due to symmetry. We transform $\langle q_i^2(t)q_j^2(t)\rangle$ from the PCS into the RCS with $\boldsymbol{R}$ in eq 3.7

$$\begin{aligned}\langle u_i^2(t)u_j^2(t)\rangle_{\mathrm{id}} &= \left\langle \left(\sum_{k=1}^{3} R_{k,i}q_k\right)\left(\sum_{l=1}^{3} R_{l,i}q_l\right)\left(\sum_{m=1}^{3} R_{m,j}q_m\right)\left(\sum_{n=1}^{3} R_{n,j}q_n\right)\right\rangle\\ &= \sum_{k=1}^{3}\langle q_i^2(t)q_j^2(t)\rangle R_{ki}^2 R_{kj}^2 +\\ &\quad \sum_{m<n}^{3}\langle q_i^2(t)q_j^2(t)\rangle(R_{mi}^2R_{ni}^2 + R_{mj}^2R_{nj}^2 + 4R_{mi}R_{ni}R_{mj}R_{nj}).\end{aligned} \tag{3.16}$$

The sum of the squared deviations between model and simulation data, scaled by the inverse of the expected errors, becomes

$$\chi^2 = \sum_t \sum_{\substack{i,j=1 \\ i\leq j}}^{3} \frac{\left(\left[\boldsymbol{R}^T\langle \boldsymbol{q}(t)\boldsymbol{q}^T(t)\rangle \boldsymbol{R}\right]_{ij} - \langle u_i(t)u_j(t)\rangle\right)^2}{s_{ij}^2(t)} \tag{3.17}$$

$$= \sum_t \sum_{\substack{i,j=1 \\ i\leq j}}^{3} \frac{\left(\left[\boldsymbol{R}^T\langle \boldsymbol{q}(t)\boldsymbol{q}^T(t)\rangle \boldsymbol{R}\right]_{ij} - \langle u_i(t)u_j(t)\rangle\right)^2}{\langle u_i^2(t)u_j^2(t)\rangle - \langle u_i^2(t)\rangle\langle u_j^2(t)\rangle_{\mathrm{id}}}, \tag{3.18}$$

where the first sum is over the different lag times t used for analysis and the product of $\boldsymbol{q}$ quaternions involves only elements 1 to 3, excluding q_0. We then determine the full diffusion tensor by minimizing eq 3.17 with respect to the diagonal elements $D_{R,1}, D_{R,2}$ and $D_{R,3}$, and the rotation matrix $\boldsymbol{R}$ defining the PCS. In this work we use simulated annealing for this six-dimensional optimization. This approach ignores possible effects of nonideal rotational diffusion.

As an alternative way to calculate the rotational diffusion tensor we use the Laplace transform $\boldsymbol{\tau}'$ of the covariance matrix coefficients

$$\begin{aligned}\tau'_{ij}(s) &= 4\int_0^\infty \mathrm{d}t\ \left(\langle u_i(t)u_j(t)\rangle - \langle u_i(\infty)u_j(\infty)\rangle\right) e^{-st} \\ &= \int_0^\infty \mathrm{d}t\ \left(4\langle u_i(t)u_j(t)\rangle - \delta_{ij}\right) e^{-st}\end{aligned} \tag{3.19}$$

With $i,j \in [1,2,3]$. Diagonalizing $\boldsymbol{\tau}'(s)$ yields three eigenvalues $\mu_1(s) \geq \mu_2(s) \geq \mu_3(s)$ and a matrix of eigenvectors $\boldsymbol{R}^T$ that corresponds to the transposed rotation matrix in eq 3.7 and thus contains the principal axes of the rotational diffusion tensor as column vectors. We use eqs 2.69 to 2.73 to calculate the diffusion coefficients $D_{R,1}, D_{R,2}$ and $D_{R,3}$ from the s dependent eigenvalues $\mu_i(s)$

$$D_{R,1} = -\frac{1}{\mu_1+\mu_2} + \frac{1}{\mu_2+\mu_3} - \frac{1}{\mu_3+\mu_1} - \frac{s}{2} \tag{3.20}$$

$$D_{R,2} = -\frac{1}{\mu_1+\mu_2} - \frac{1}{\mu_2+\mu_3} + \frac{1}{\mu_3+\mu_1} - \frac{s}{2} \tag{3.21}$$

$$D_{R,3} = +\frac{1}{\mu_1+\mu_2} - \frac{1}{\mu_2+\mu_3} - \frac{1}{\mu_3+\mu_1} - \frac{s}{2}. \tag{3.22}$$

We expect the right-hand side of eqs 3.20 to 3.22 to be independent of s for ideal rotational dynamics without noise. In practice these assumptions are generally not fulfilled. Therefore, we use the reciprocal of characteristic time for rotational diffusion $s \approx 1/\tau_1$ to estimate $D_{R,1}, D_{R,2}$ and $D_{R,3}$. τ_1 is independent of the RCS and can calculate by integrating $u_0^2(t)$, see eq 2.82.

3.2.4 Finite-Size Effects in Rotational Diffusion

PBC are a severe constraint on the hydrodynamic flow. For translational diffusion in an cubic simulation box of length L we can determine the apparent diffusion coefficient D_{PDC} using [46, 236]

$$D_{\mathrm{PBC}} = D_0 - \frac{2.837}{k_{\mathrm{B}}T/6\pi\eta L}, \tag{3.23}$$

where, D_0 is the infinite box-size diffusion coeffient, η is the shear viscosity, k_{B} is the Boltzmann constant, and T the absolute temperature. The apparent diffusion coefficient is reduced because the velocity field of the hydrodynamic flow generated by a particle moving in a periodically replicated box has to satisfy PBC. We illustrate this effect for

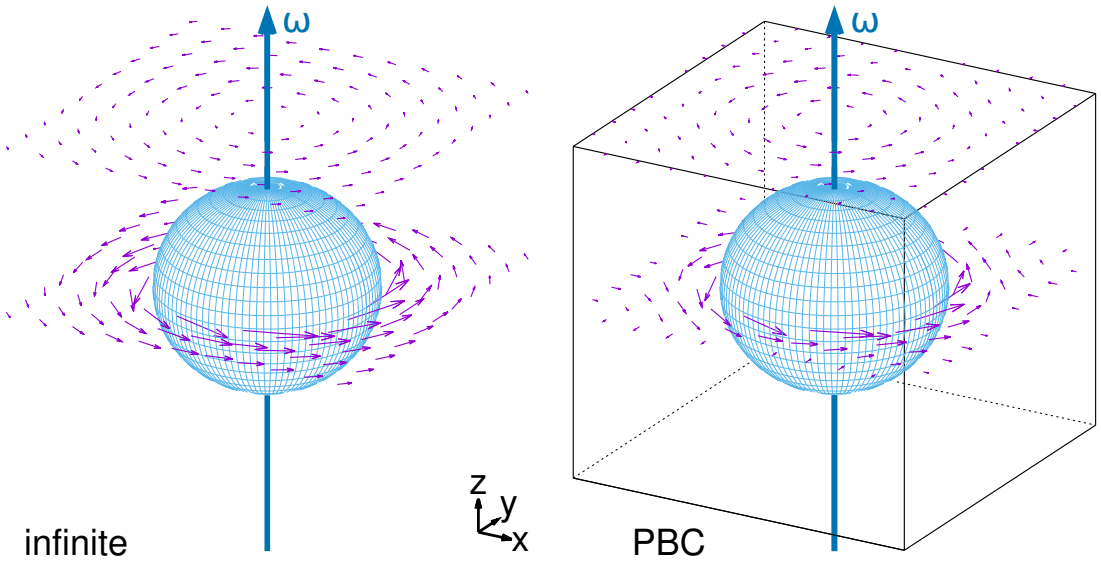

Figure 3.1: Effect of PBC on the hydrodynamic flow around a rotating sphere. (Left) Unrestricted rotational flow in infinite space. (Right) Fluid flow in a cubic box of length L under PBC. The sphere of radius $R_h = 0.3L$ rotates at constant angular velocity ω around the z axis. The velocity field of the surrounding fluid is shown as arrows at the mid-plane, $z = 0$, and at $z = L/2$. The unrestricted velocity field is the "rotlet" $\mathbf{v}(\mathbf{r}) = (R_h/r)^3\,\boldsymbol{\omega} \times \mathbf{r}$ where $\boldsymbol{\omega} = (0, 0, \omega)^T$. The vertical blue arrow indicates the axis of rotation $\boldsymbol{\omega}$. For visualization under PBC, the simple-cubic lattice sum of rotlets evaluated using Ewald summation [4] is shown, not the full solution of the Stokes equation for no-slip boundary conditions on the surface of the rotating sphere [242].

a hydrodynamic flow generated by a rotating particle in a periodically replicated box in Figure 3.1, there we compare the hydrodynamic flow around a sphere rotating about the z axis in infinite space and in a cube under PBC. Under PBC the tangential flow vanishes on the four faces at $x = \pm L/2$ and $y = \pm L/2$ that are aligned with the rotation axis, which amounts to a no-slip-like boundary condition. We therefore expect additional friction for macromolecules in tight simulation boxes. It has been shown by Zuzovsky et al. [242] that for lattices of rotating spheres with no-slip boundary condition the reciprocal of

the rotational friction coefficient ζ^r depends, to the lowest order, linearly on the volume fraction,

$$1/\zeta^r = 1 - v_{\mathrm{mol}}/V, \tag{3.24}$$

where $v_{\mathrm{mol}} = 4\pi {R_h}^3/3$ is the hydrodynamic volume expressed in terms of the hydrodynamic radius R_h of the rotating sphere. For this Zuzovsky et al. [242] generalized Hasimoto's [68] calculation and performed detailed analytical and numerical calculations for simple cubic (sc), face-centered cubic (fcc), and body-centered cubic (bcc) lattices of rotating spheres [75, 157, 242]. The Wigner-Seitz cells of these three Bravais lattices correspond to simple cubic (sc), rhombic dodecahedral (fcc), and truncated octahedral (bcc) simulation boxes, respectively.

This dependence of the rotational friction on the relative volume of a rotating sphere can be recovered from a simple hydrodynamic model of spherical Couette flow between two counter-rotating concentric spheres with no-slip boundary conditions. Adapting Fushiki's approach for translational diffusion [58], we replace the simulation box volume by a Wigner-Seitz sphere of equal volume, *i.e.*, with radius $R = (3V/4\pi)^{1/3}$. In the Stokes regime of laminar flow [67], ζ^r of the inner sphere of radius R_h increases by a factor $(1 - (R_h/R)^3)^{-1}$ over the infinite-system value obtained in the limit $R \to \infty$. This result agrees with the detailed calculations [75, 157, 242] to lowest order in the volume fraction. Assuming the same box-size dependence of the rotational diffusion coefficient on the rotational friction coefficient also for aspherical particles simulated under PBC, and $D_R = k_{\mathrm{B}}T/\zeta^r$, the expected rotational diffusion coefficient $D_{R,\mathrm{PBC}}$ observed in a simulation depends on box volume as

$$D_{R,\mathrm{PBC}} = \left(1 - \frac{v_{\mathrm{mol}}}{V}\right) D_{R,0} = \left(1 - \frac{4\pi {R_h}^3}{3V}\right) D_{R,0} \tag{3.25}$$

where $D_{R,0}$ is the infinite-system rotational diffusion coefficient and $v_{\mathrm{mol}} = 4\pi {R_h}^3/3$ is the effective volume of the rotating particle. In this model R_h is the hydrodynamic radius of the macromolecule. For large, near-spherical macromolecules, the rotational diffusion coefficient is expected to satisfy the Stokes-Einstein relation, $D_{R,0} = k_{\mathrm{B}}T(8\pi\eta {R_h}^3)^{-1}$, which further simplifies the box-size dependence, eq 3.25, to

$$D_{R,\mathrm{PBC}} = \left(1 - \frac{4\pi {R_h}^3}{3V}\right) \frac{k_{\mathrm{B}}T}{8\pi\eta {R_h}^3} = D_{R,0} - \frac{k_{\mathrm{B}}T}{6\eta V}. \tag{3.26}$$

Eq 3.25 shows that the apparent rotational diffusion coefficient depends linearly on the inverse box volume V^{-1} with intercept $D_{R,0}$, and eq 3.26 shows that the slope depends solely on temperature and viscosity.

3.2.5 Rotational Brownian Dynamics Algorithm

To verify the calculated rotational diffusion tensors we test whether the values from our simulations are within the expected distribution. To calculate the expected distribution of rotational diffusion coefficients and rotation axes we use BD simulations of the rotational dynamics. To develop a BD algorithm we consider the short time approximation of the covariance coefficients in the PCS

$$\langle q_i^2(t)\rangle = \frac{1}{2}D_{R,i}t + \mathcal{O}(t^2), \tag{3.27}$$

for $i = 1, 2, 3$, and $\langle q_i(t)q_j(t)\rangle = 0$ for $i \neq j$. The short time approximation for the privileged q_0 is

$$\langle q_0(\Delta t)^2\rangle = 1 - \frac{(D_{R,1} + D_{R,2} + D_{R,3})\Delta t}{2} + \mathcal{O}(t^2). \tag{3.28}$$

Akin to purely translational BD algorithms we set

$$\tilde{q}_i(\Delta t) = \left(\frac{D_{R,i}\Delta t}{2}\right)^{1/2} g_i \tag{3.29}$$

for $i = 1, 2, 3$, with g_i uncorrelated Gaussian random numbers of zero mean and unit variance, but keep the privileged q_0 as

$$\tilde{q}_0(\Delta t) = \left(1 - \frac{(D_{R,1} + D_{R,2} + D_{R,3})\Delta t}{2}\right)^{1/2} \tag{3.30}$$

ensuring that on average $\tilde{\boldsymbol{q}}$ is normalized to one on first order in time

$$\left\langle \sum_{i=0}^{3} \left(\tilde{q}_i(\Delta t)\right)^2 \right\rangle \approx 1 - \frac{\Delta t^2}{4}\left(D_{R,1}^2 + D_{R,2}^2 + D_{R,3}^2 + D_{R,1}D_{R,2} + D_{R,2}D_{R,3} + D_{R,3}D_{R,1}\right). \tag{3.31}$$

To guarantee that the normalization condition is fulfilled at every time-step the quaternions $\tilde{\boldsymbol{q}}(\Delta t)$ are normalized

$$q_i'(\Delta t) = \tilde{q}_i(\Delta t)\left(\sum_{i=0}^{3}\left(\tilde{q}_i(\Delta t)\right)^2\right)^{-1/2}. \tag{3.32}$$

Lastly the quaternion $\boldsymbol{q}'(\Delta t)$ has to be multiplied with the quaternion at time t to advance the trajectory

$$\boldsymbol{q}(t + \Delta t) = \boldsymbol{q}(t)\boldsymbol{q}'(\Delta t). \tag{3.33}$$

As initial orientation we choose $\boldsymbol{q}(0) = (1, 0, 0, 0)^T$ to follow the rotational motion of the molecule in the PCS at time $t = 0$. The quaternion product $\boldsymbol{q}(t)\boldsymbol{q}'(\Delta t)$ is defined as

$$\begin{aligned} q_0'' &= q_0' q_0 - q_1' q_1 - q_2' q_2 - q_3' q_3 \\ q_1'' &= q_1' q_0 + q_0' q_1 + q_3' q_2 - q_2' q_3 \\ q_2'' &= q_2' q_0 - q_3' q_1 + q_0' q_2 + q_1' q_3 \\ q_3'' &= q_3' q_0 + q_2' q_1 - q_1' q_2 + q_0' q_3 \end{aligned} \tag{3.34}$$

with $\boldsymbol{q}'' = \boldsymbol{q}(t + \Delta t)$ being the new orientation, $\boldsymbol{q}' = \boldsymbol{q}'(\Delta t)$ the change in orientation, and $\boldsymbol{q} = \boldsymbol{q}(t)$ the old orientation at each time-step.

3.3 Methods

3.3.1 Anisotropic Tetrahedron

To test the fitting algorithm we created a 500 ns rotational Brownian dynamics trajectory of a tetrahedron with a rotational diffusion tensor of $\boldsymbol{D} = \mathrm{diag}(20, 10, 5)$ ns^{-1}. We used the rotational Brownian dynamics algorithm described in Section 3.2.5 with an integration time-step of $\Delta t = 10$ fs to generate a trajectory in orientational space. We then applied each orientation to the tetrahedron and stored the resulting 10^5 structures in a trajectory file.

3.3.2 Molecular Dynamics Simulations of the Dickerson-Drew DNA Dodecamer

We calculated the rotational diffusion tensor of the Dickerson-Drew dodecamer B-DNA double helix (PDB code 1DUF [208]). As input, we used multiple all-atom molecular dynamics trajectories at different box volumes. The simulations were run with GROMACS 2016.3 [3] using the AMBER Parmbsc1 forcefield [87] in a cubic box with TIP3P water molecules [89] and ≈150 mM NaCl [91] at a temperature [29] of 298.15 K and a pressure [151] of 1 bar, at a time-step of 2 fs. Exact initial box volumes and trajectory lengths are listed in Table 3.1.

Set	V [nm^3]	L [nm]	d [nm]	Trajectory Length [μs]
B_1	233	6.162	1	5 × 1.4
B_2	365	7.147	1.5	5 × 1.9
B_3	580	8.343	2	10 × 1.0

Table 3.1: Different simulation box sizes, corresponding simulation times, and set names for B-DNA. All trajectories at the same box size have the same length. L is the edge-length of the cubic ($V = L^3$) boxes. d is the shortest distance between the solute to any box edge.

3.3.3 Molecular Dynamics Simulations of Myoglobin

We calculated the rotational diffusion tensor of CO-bound horse-heart myoglobin (PDB code 1DWR [33]). As input, we used multiple all-atom molecular dynamics trajectories at different volumes. The simulations were done with GROMACS 5.1.4 [3] using the Amber99sb*-idln forcefield [17, 78, 113] and the Giammona-Case force field for the CO-bound heme [62] in a rhombic dodecahedron box with water molecules [89] and ≈150 mM NaCl [92] at a temperature [29] of 293 K and a pressure [151] of 1 bar, at a time-step of 2 fs. Exact initial box volumes and trajectory lengths are listed in Table 3.2.

Set	V [nm^3]	L [nm]	d [nm]	Trajectory Length [μs]
M_1	206	4.064	0.8	5×1.4
M_2	240	4.274	1	10×2.3
M_3	293	4.565	1.25	5×1.1
M_4	366	4.920	1.5	5×2.4
M_5	516	5.516	2	5×2.9

Table 3.2: Different simulation box sizes, corresponding simulation times, and set names for myoglobin. All trajectories at the same box size have the same length. L is the edge-length of the rhombic dodecahedral box ($V = 16L^3/3\sqrt{3}$). d is the shortest distance between the solute to any box edge.

3.3.4 Hydrodynamics

We compared the MD rotational diffusion tensors of B-DNA and myoglobin to corresponding hydrodynamic estimates using HYDROPRO [147]. We used atomic-level models with a hydrodynamic bead radius of 0.29 nm for the heavy atoms of B-DNA and myoglobin. For the B-DNA we used a viscosity for TIP3P water of 0.321(16) mPa s at 298.15 K [218]. We did correct the viscosity to account for the 150 mM NaCl. For myoglobin we used a viscosity for TIP3P water of 0.326(16) mPa s at 293 K. To obtain the viscosity at 293 K we used a linear extrapolation of the temperature-dependent viscosity of TIP3P water reported by Mao and Zhang [125]. To account for the variability of the molecules we calculated the rotational diffusion tensor as an average, over an ensemble of structures selected from the MD simulations of B-DNA and myoglobin.

3.3.5 Calculating the Covariance Matrix

To calculate the orientations with respect to the RCS, we used the qcprot RMSD alignment algorithm [117, 205] implemented in MDAnalysis [65, 135]. As a reference structure, we used the average structure in the simulations, which for B-DNA was obtained by RMSD aligning the inner five base pairs and averaging the simulation structures iteratively to convergence. The first frame was used for the initial alignment. For myoglobin, we used the C_α backbone atoms in the recursive RMSD alignment. To ensure that the fitted rotational diffusion tensor is independent of the RCS, as defined by the reference structure

in the RMSD alignment, we verified that the results did not change when different frames of the trajectory were used to define the RCS. The references structure was determined from the simulations in simulation set B_1 for B-DNA and simulating set M_2 for myoglobin.

3.3.6 Fitting the Covariance Matrix

We used a simulated annealing algorithm to minimize χ^2 in eq 3.17. The annealing algorithm alternated between optimizing the rotational diffusion coefficients D_i and the PCS every 20 iterations. The Monte Carlo move widths in the rotational diffusion coefficients were decremented in 0.005 % steps from 5 % to a minimum of 0.5 % of the current value during cooling. To search for the optimal orientation, the PCS was rotated about a random axis by a random angle whose range was reduced from 90° at the beginning to a minimum of 1° in 0.5° steps during cooling. For the system studied here, 5000 iteration cycles were found to lead to well-converged results. To find the global minimum, we used the minimum of 10 independent annealing runs. The annealing algorithm was implemented Python [54] with time-critical functions like eq 3.17 using Numba [108] and NumPy [146] for optimal performance. The implementation of the simulated annealing algorithm has been published in the *pydiffusion* Python package [114].

3.3.7 Error Estimates

To estimate the statistical error of the rotational diffusion tensor, we used 100 rotational BD simulations. The simulations were of equal length as the corresponding MD simulations and used the best-fit rotational diffusion tensor as input. From each Brownian dynamics simulation, we calculated the time-dependent quaternion covariance matrices, and from these determined the rotational diffusion tensors. The rotational BD algorithm has been implemented in Cython [12]. The implementation has been published in the *pydiffusion* Python package [114].

From these simulations, we also estimated the number N of independent samples. The factor N should bring the variances $\hat{s}^2_{ij}(t)$ of repeated estimates of the quaternion covariances into agreement with the analytical estimates $s^2_{ij}(t)$ for ideal diffusion, as given in eq 3.12,

$$N\hat{s}^2_{ij}(t) \approx s^2_{ij}(t). \tag{3.35}$$

Applying this relation to the variance $\hat{s}^2_{ij}(t)$ determined from rotational BD simulation allows us to estimate N.

To compare two rotational diffusion tensors, $\boldsymbol{D}$ and $\boldsymbol{D'}$, we used the Frobenius norm,

$$\sigma = |\boldsymbol{D} - \boldsymbol{D'}| = \left(\sum_{i,j} \left(D_{ij} - D'_{ij} \right)^2 \right)^{1/2}. \tag{3.36}$$

As a test statistic, we used the normalized distribution $p(\sigma)$ of σ values obtained for the difference between the input and fitted rotational diffusion tensors in the rotational

BD simulations. The complement of the cumulative distribution, $\int_{\sigma}^{\infty} \mathrm{d}\sigma'\, p(\sigma')$, provides an estimate of the probability of observing a deviation as large as σ or larger purely by chance.

3.4 Results and Discussion

3.4.1 Anisotropic Tetrahedron

To test if we can determine the correct rotational diffusion tensor for ideal Brownian rotation we used a tetrahedron trajectory, generated with known rotational diffusion tensor of high anisotropy $\boldsymbol{D} = \mathrm{diag}(20, 10, 5)$. The largest error of the estimated rotational diffusion coefficient is 4 % in the slowest axes, Table 3.3. Therefore our annealing algorithm can recover anisotropic diffusion coefficients from simulation data.

	Input	Fit
D_1	20	20.51
D_2	10	9.96
D_3	5	5.03

Table 3.3: Comparison of input rotational diffusion coefficients used in a rotational Brownian dynamics simulation of a tetrahedral molecule, and the rotational diffusion tensor estimated from a fit to the resulting 500 ns trajectory.

3.4.2 Quaternion Covariances and Rotational Correlation Function

Figures 3.2 and 3.3 show the raw quaternion covariances in the RCS. For the B-DNA, Figure 3.2, the diagonal covariance coefficients $\langle q_i(t)^2 \rangle$ are similar for all box volumes in the first 5 ns. At later times differences become visible. The off-diagonal elements are similar only for the first 3 ns. The overall shapes of the off-diagonal elements are similar though. The off-diagonal elements contain information about the rotation axes, therefore we expect this similarity because we chose the same reference structure for all simulations. For myoglobin, Figure 3.3 the covariance coefficients diverge earlier. There is also no clear trend at short times for the off-diagonal elements. Suggesting that our chosen reference structure was already closely orientated to the PCS and that the differences are due to noise.

Figure 3.4 shows the correlation functions $\langle\langle P_1(\cos(\theta(t)))\rangle\rangle$. The average P_1 correlation function was calculated directly from the variance of the scalar quaternion component q_0, eq 2.80. Both for myoglobin and B-DNA, increasing the box size accelerates the rotational dynamics.

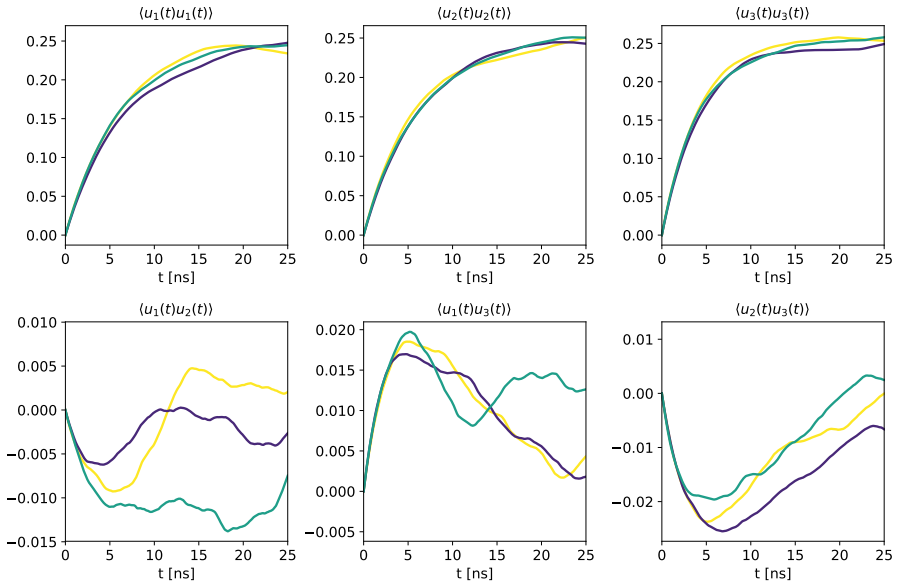

Figure 3.2: Quaternion covariances of B-DNA from MD simulations at three different volumes of the periodic boundary box. Colors increase in brightness from the smallest volume 233 nm^3 (dark blue) to the largest volume 580 nm^3 (yellow).

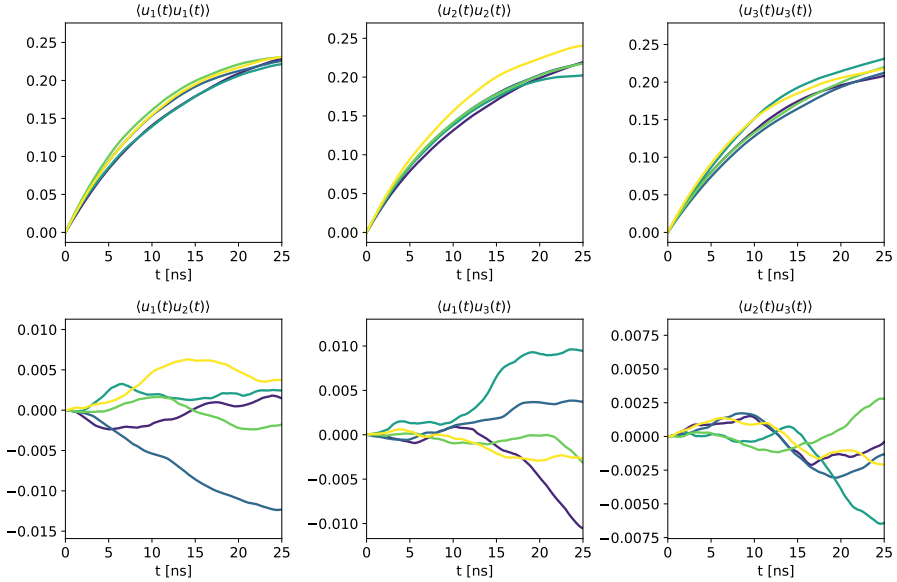

Figure 3.3: Quaternion covariances of myoglobin from MD simulations at five different volumes of the periodic boundary box. Colors increase in brightness from the smallest volume 206 nm^3 (dark blue) to the largest volume 516 nm^3 (yellow).

3.4.3 Rotational Diffusion Tensor

A fit of the quaternion covariances together with the hydrodynamic estimate and comparison of the rotation axes is shown in Figure 3.5. We fitted the average quaternion covariance obtained from from 0 to 24.9 ns, sampled at 5 ps intervals. We used simulation set B_1 and M_2 for B-DNA and myoglobin, respectively. Fitting the time-dependent

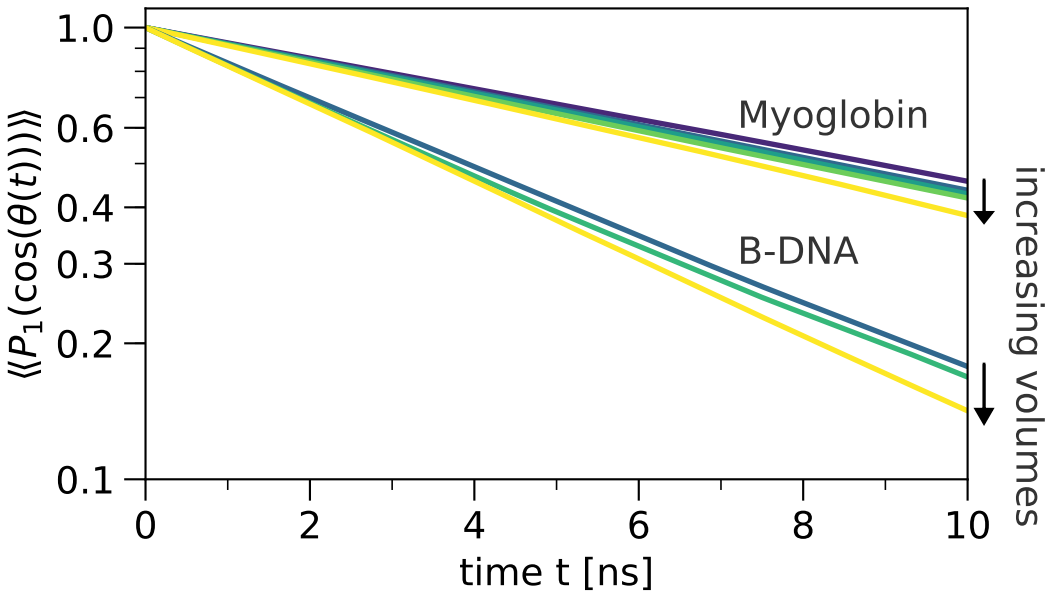

Figure 3.4: Orientational correlation function $\langle\langle P_1(\cos(\theta(t)))\rangle\rangle$ for myoglobin and B-DNA averaged over all MD trajectories and orientations of the reference vector for small (dark blue) to large simulation boxes (yellow).

quaternion covariance matrices led to an improvement of χ^2 for myoglobin from $\chi^2 = 67$ for the hydrodynamic rotational diffusion tensor to $\chi^2 = 11$ for the fitted rotational tensor. For B-DNA we also see an improvement, from $\chi^2 = 13$ for the hydrodynamic rotational diffusion tensor to $\chi^2 = 7$ for the fitted rotational diffusion tensor. The components $\langle u_i(t)u_j(t)\rangle$ calculated for the best-fit rotational diffusion tensor agree very well with the covariances calculated from the MD trajectories, see Figs. 3.5 and 3.7. The fitted rotational diffusion coefficients D_i are remarkably close to the hydrodynamic estimates, see Figure 3.6. The principal rotation axes from simulation and hydrodynamics calculation agree very well, see Figure 3.5 A and B. All three rotation axes are in good correspondence, with hydrodynamic axes 2 and 3 being nearly degenerate for B-DNA and all three axes being nearly degenerate for myoglobin. The main axis 1 of rotation points along the long axis of the DNA and agrees with experiment [23].

The correlation time τ_c eq 2.86 of the Dickerson-Drew dodecamer from experiment [45] is 5 ns, which is in excellent agreement with our simulation results of $\tau_c = 4.8$ ns after scaling by a factor of 2.7, the ratio of the viscosity of water and TIP3P water. The corresponding hydrodynamic estimate is 4.6 ns. The simulation estimate for the P_2 correlation time is $\tau_2 = 4.6$ ns. An experimental harmonic mean relaxation time [224] of $\tau_c = 9.7(3)$ ns for myoglobin compares well with the 10.4 ns obtained both from hydrodynamics and simulations, after scaling by a factor of 2.7 to account for the lower viscosity of TIP3P water. Considering the combined uncertainties in the experiment, the viscosity

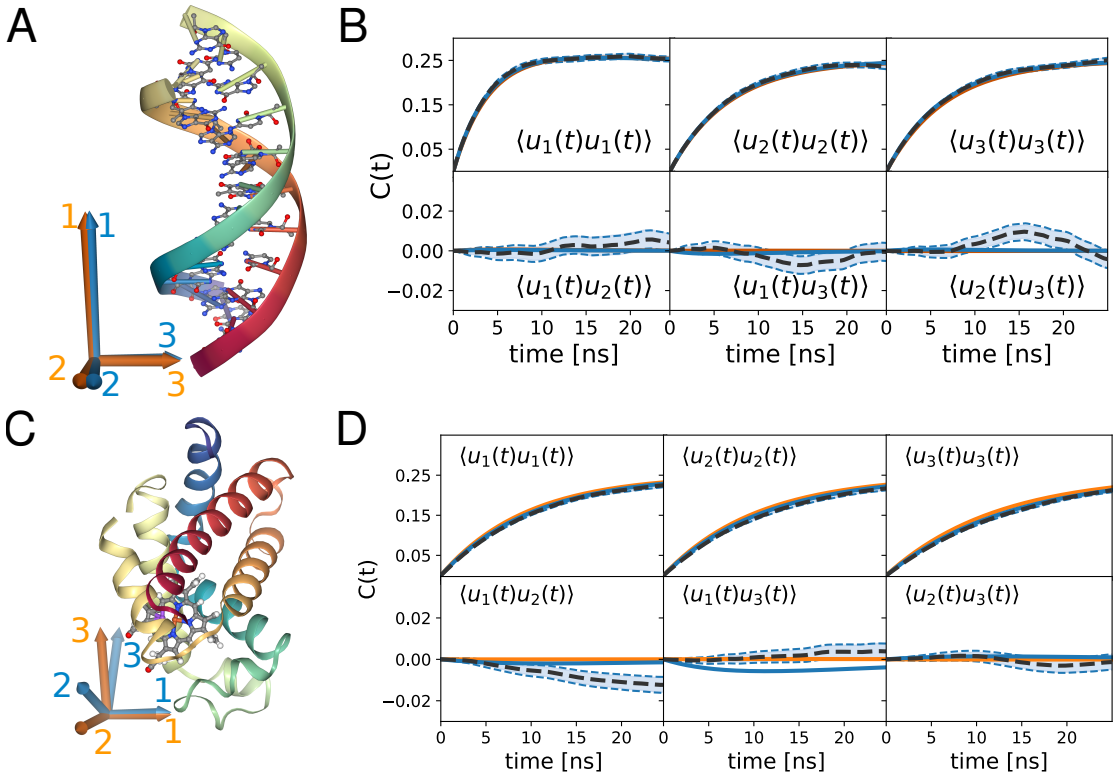

Figure 3.5: Rotational diffusion tensors of B-DNA and myoglobin obtained from MD simulation trajectories compared to hydrodynamic calculations. Average structure of B-DNA (A) and myoglobin (C) with principal axes of the rotational diffusion tensor, scaled by the respective principal values D_i of the rotational diffusion tensor, from hydrodynamic estimates (orange) and least-χ^2 fit (blue). Numbers indicate the order of the axes. Time-dependent covariance matrix coefficients of B-DNA (B) and myoglobin (D) from simulations (black dashed lines), a numerical fit (blue), and the HYDROPRO [147] hydrodynamic estimate (orange). The expected standard error of the mean based on the Brownian dynamics simulations using the best-fit model as input is shown as light blue shaded area. The dashed blue line is the theoretical expected standard error of the mean for the best-fit tensor based on eq 3.35, with an independent sample size estimated as $N \approx 2000$ and $N \approx 2400$ for B-DNA and myoglobin, respectively. Data shown here is from simulating set B_1 for B-DNA and M_2 for myoglobin.

correction, and the fit of the rotational diffusion tensor, the agreement is excellent for both molecules. Note that we do not correct the shear-viscosity for the 150 mM NaCl present in the simulation.

The rotational motion of the B-DNA is well described by a cylindrical rotational diffusion tensor. Using a fit with an imposed cylindrical symmetry, $D_2 = D_3$, does not lead to significant changes in χ^2 for the B-DNA. We estimated a diffusion anisotropy $2D_1/(D_2 + D_3)$ of 1.87(7). This is slightly lower than the experimental value of 2.1(4) but still within the combined uncertainties of simulation and experiment. The corresponding hydrodynamic estimate is 1.86. The error of the diffusion anisotropy estimate from our simulations has been calculated from one hundred Brownian dynamics simulations of the

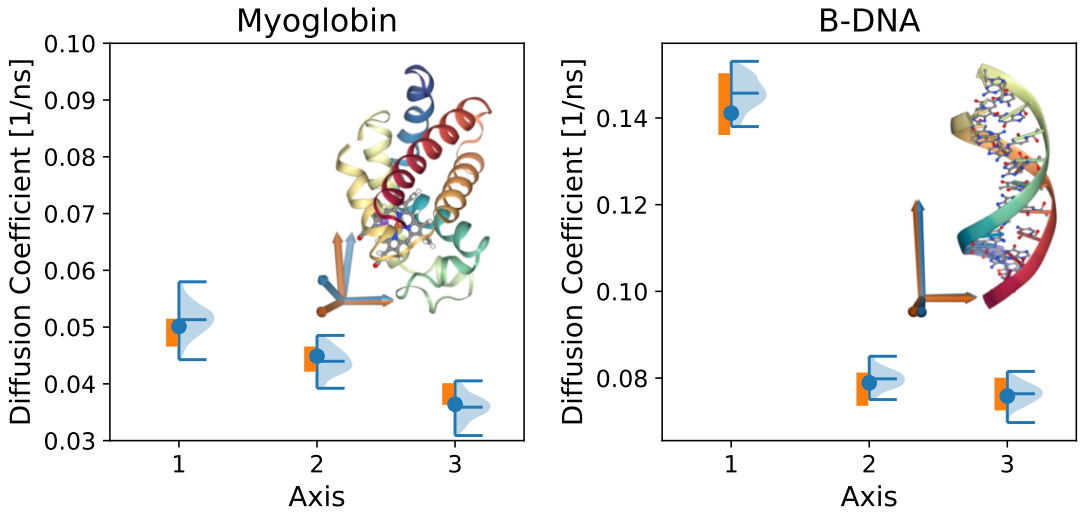

Figure 3.6: Principal values D_i of the rotational diffusion tensor for myoglobin (left) and B-DNA (right) from a least-χ^2 fit to the observed quaternion covariance matrices (blue dots). The distribution of rotational diffusion coefficients from 100 Brownian dynamics simulations, using the best-fit rotational diffusion tensor as input, is shown as blue shaded area with markers for the minimum, maximum, and mean values. The ranges of the hydrodynamic estimates based on the viscosity of pure TIP3P water with a 5 % error are shown as orange bars. In the respective upper right corners myoglobin and B-DNA are shown aligned with the PCS together with the rotation axes as identified from fits to the quaternion covariance (blue) and hydrodynamic estimates (orange). Note that the plotted rotational diffusion coefficients have not been corrected for the low TIP3P water viscosity. Data shown here is from simulating the set B_1 for B-DNA and M_2 for myoglobin.

same length as the MD simulations.

3.4.4 Goodness of Fit

To access the goodness of fit we use the rotational BD simulations to calculate the residuals of the covariances. The residuals are rarely larger than one standard deviation both for B-DNA and myoglobin and agree well with the analytical result, see Figure 3.7. As shown in Figure 3.7, diagonal and off-diagonal elements of the covariance matrix have errors of the same magnitude at fixed time t. We scaled the analytical estimate of the standard errors of the mean of the covariance matrix, eq 3.12, by the effective sample size N according to eq 3.35. We estimate $N \approx 2000$ and $N \approx 2400$, for B-DNA and myloglobin, respectively, as shown in Figure 3.7. The corresponding time per sample is $10 \times 1\,\mu\text{s}/2000 = 5\,\text{ns}$ and $10 \times 2.3\,\mu\text{s}/2400 = 9.6\,\text{ns}$ for B-DNA and myoglobin, respectively. This corresponds well with experimental rotational correlation times.

The Frobenius norm, eq 3.36, of the different rotational diffusion tensors, see Figure 3.8A, is a more quantitative comparison that takes into account exchanges in the order of the principal values. As a reference, the distribution of the norm σ of the differ-

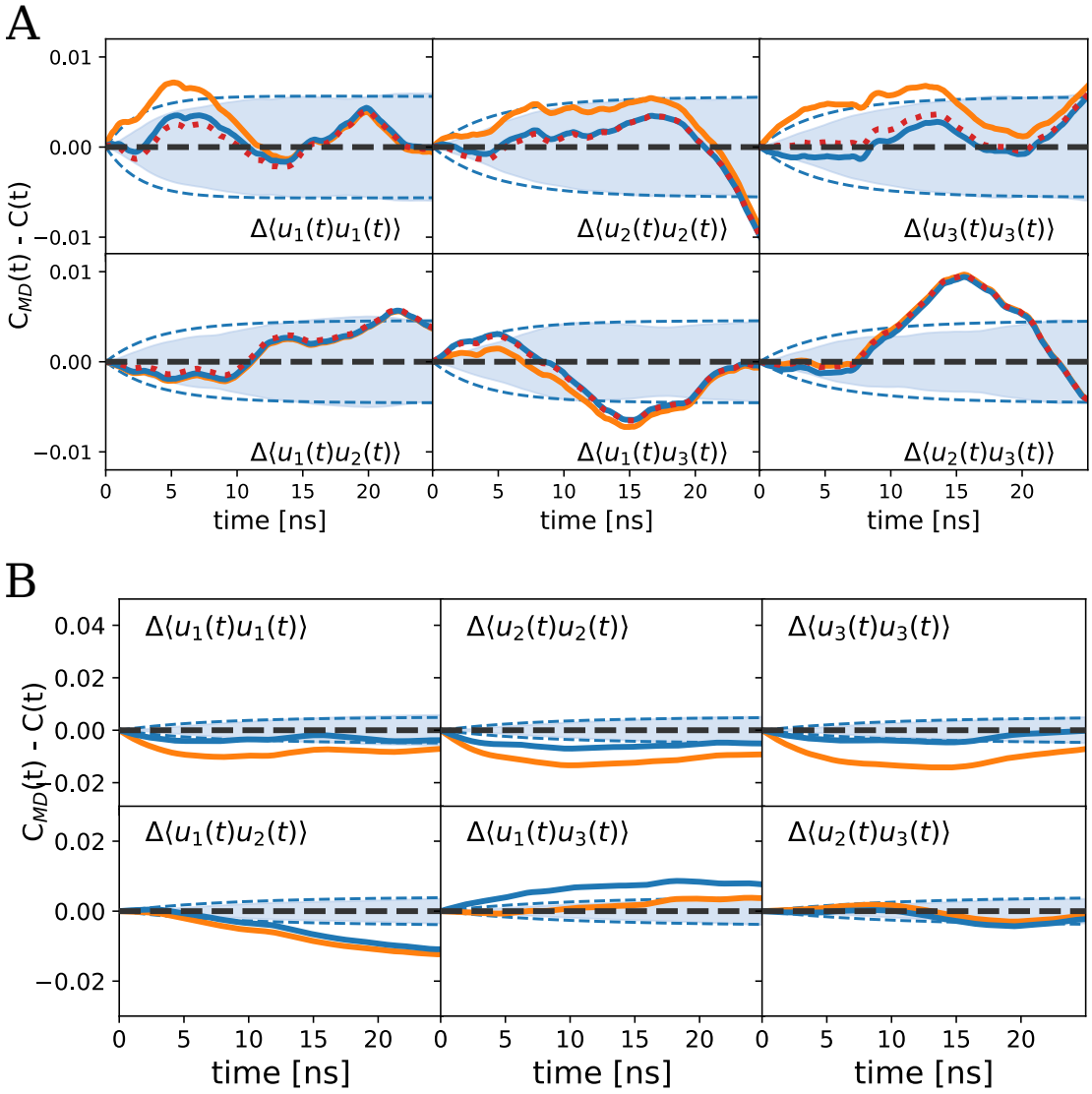

Figure 3.7: Residuals of the quaternion covariance matrix coefficients for the least-χ^2 fit (solid blue lines) and the hydrodynamics estimate (solid orange lines) for B-DNA (A) and myoglobin (B). The expected standard errors of the mean based on the best-fit model are shown as dashed blue lines, after scaling by $1/N$, where N is the estimated effective sample size. The standard errors of the mean from rotational Brownian dynamics simulations are shown in light blue shading. For B-DNA the residuals for a fit of a cylindrical rotational diffusion tensor are shown as a red dotted line.

ence between input and fitted rotational diffusion tensors was calculated from repeated rotational BD simulations of equal lengths to the MD simulations. Using this reference, the probability to obtain, purely by chance, a value of σ that exceeds the Frobenius norm of the difference between the hydrodynamic and fitted rotational diffusion tensors is 47 % for B-DNA and 23 % for myoglobin. We compared the χ^2 values of the fit of the rotational diffusion model to the rotational BD simulations with the χ^2 value from the MD simulations in Figure 3.8B. With the χ^2 value of the fit well within the distribution for

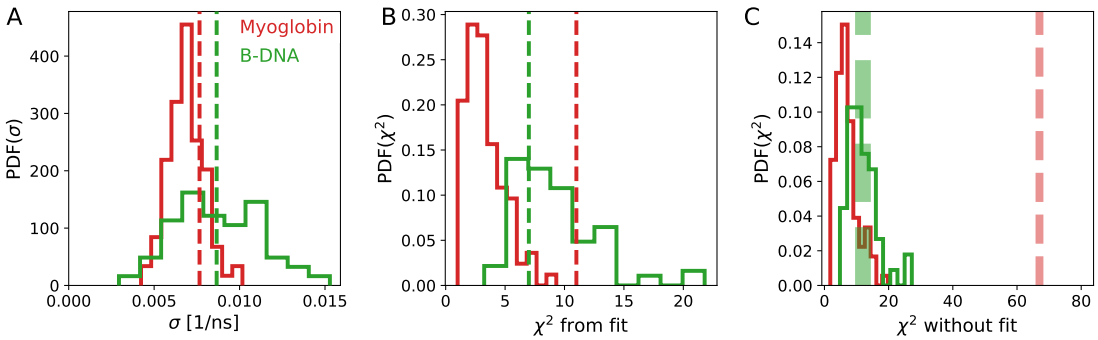

Figure 3.8: Goodness of fit. (A) Distribution of the Frobenius norm σ of difference tensors between the best-fit rotational diffusion tensor and the fitted tensors from repeated rotational BD simulations, which used the best-fit MD rotational diffusion tensor as input. The dashed vertical line indicates the corresponding Frobenius norm between the best-fit tensor and the hydrodynamic estimate. (B) Distribution of χ^2 values from fits to repeated rotational BD simulations. For reference, the dashed vertical line shows the χ^2 value of the fit to the molecular dynamics simulations. (C) Distribution of χ^2 value without a fit, as obtained from the repeated rotational BD simulations. For reference, the dashed vertical line shows the χ^2 value of the hydrodynamic estimate. Its width reflects uncertainties in the viscosity of pure TIP3P water of 5 %.

B-DNA, the rotational dynamics in the MD simulations is well explained by the best-fit rotational diffusion tensor. For myoglobin, the χ^2 value of the fit is slightly outside of the distribution. Because myoglobin has a nearly isotropic diffusion tensor all three principal axes are nearly degenerate and it is much more likely that our results are influenced by noise. Therefore we conclude that the simulations are well explained by the best-fit rotational diffusion tensor.

Figure 3.8C compares the distribution of χ^2 values without fitting the rotational diffusion model. For B-DNA this plot shows that the χ^2 value of the hydrodynamic, despite being larger than that of the fit, is indeed in the expected range. This finding is consistent with the results for the Frobenius norm in Figure 3.8B. For myoglobin Figure 3.8C shows that the hydrodynamic model is different from the MD fitted rotational diffusion tensor. This discrepancy can also be seen in Figure 3.5C where the second and third rotation axes are visibly different from the hydrodynamic model. Given the low anisotropy, and therefore near degeneracy of the principal axes, for myoglobin, it is to be expected that exact determination of the rotational diffusion coefficient together with the principal axes is difficult.

3.4.5 Rotational Diffusion Tensor from Laplace Transform

Figure 3.9 shows the dependence of the principal values D_1, D_2, and D_3 on s in the range from 1 to 1/(200 ns) calculated from integrating the Laplace transform eqs 3.19 to 3.22. For the B-DNA, all three coefficients converge to values close to those of the least-χ^2 fit.

For myoglobin only D_1 and D_2 converge to the least-χ^2 fit. The second coefficient the Laplace transform predicts a value of 12 % less than the least-χ^2 fit.

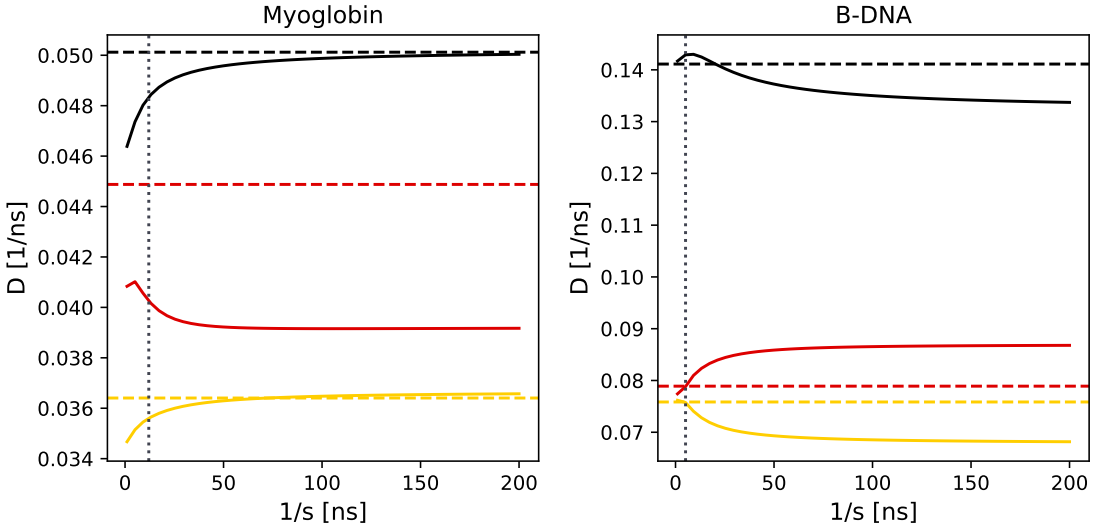

Figure 3.9: Principal values D_i of rotational diffusion tensor of myoglobin (left) and B-DNA (right) calculated directly from the Laplace transforms of the covariance matrix elements, eqs 3.19 to 3.22, for different values of s. The coefficients are colored according to their axes (black for D_1, red for D_2 and gold for D_3). The horizontal dashed lines show the least-χ^2 fit and the vertical black dashed line indicates $s = 1/\tau_1$.

3.4.6 Boxsize Dependency

From the rotational diffusion tensors calculated for B-DNA and myoglobin we calculated the invariant mean rotational diffusion coefficient $D_{R,\mathrm{PBC}} = (D_{R,1}+D_{R,2}+D_{R,3})/3$ for each box size, Table 3.4. Figure 3.10 shows the dependence of the mean rotational diffusion coefficients of B-DNA and myoglobin on the inverse box volume. For both systems the apparent rotational diffusion coefficient $D_{R,\mathrm{PBC}}$ decreases linearly with $1/V$. Also shown are single-parameter fits of eq 3.26. All fitted values including errors are listed in Table 3.4.

The hydrodynamic radius R_h is sufficient to explain the values of $D_{R,\mathrm{PBC}}$ and the dependence on the box volume. For myoglobin and the B-DNA, we obtained values of $R_h = 2.15(3)\,\mathrm{nm}$ and $R_h = 1.72(3)\,\mathrm{nm}$, respectively. These calculated values of R_h are in excellent agreement with the NMR experimental values of $2.12(2)\,\mathrm{nm}$ for horse-heart myoglobin [224, 227] and 1.71 nm for the Drew-Dickerson B-DNA dodecamer [45]. We calculated R_h from second-order rotational correlation times τ_2 using eq 2.85 with water viscosities of $\eta = 1.0016\,\mathrm{mPa\,s}$ and $\eta = 0.975\,\mathrm{mPa\,s}$ at 20 °C and at 21 °C, respectively.

We can remove the box volume dependency by adding $k_\mathrm{B}T/6\eta$, according to eq 3.26, to the mean rotational diffusion coefficients. For myoglobin, the correction removes the box-size dependence. For B-DNA, the data suggest a residual box size dependence, albeit within the statistical errors. A possible cause for the small deviations is the near-cylindrical

	V [nm^3]	$D_{R,1}$	$D_{R,2}$	$D_{R,3}$	D_{PBC}	$D_{R,0}$
B-DNA	233	0.129(3)	0.073(2)	0.066(2)	0.089(2)	0.098(1)
	365	0.136(4)	0.077(3)	0.071(2)	0.095(3)	0.101(2)
	580	0.143(3)	0.078(2)	0.076(2)	0.099(3)	0.103(2)
	∞^b	0.142(3)	0.082(2)	0.077(2)		0.100(3)
Myoglobin	206	0.041(1)	0.038(1)	0.038(1)	0.039(1)	0.049(1)
	240	0.047(1)	0.040(1)	0.035(1)	0.0414(5)	0.0499(5)
	293	0.0453(13)	0.041(1)	0.039(1)	0.042(1)	0.049(1)
	366	0.051(1)	0.042(1)	0.038(1)	0.043(1)	0.0495(7)
	516	0.0506(6)	0.0469(6)	0.0452(6)	0.0476(6)	0.0516(7)
	∞^b	0.055(1)	0.049(1)	0.045(1)		0.0500(4)

Table 3.4: Rotational diffusion coefficients of B-DNA and myoglobin for different simulation box volumes. Eigenvalues $D_{R,\mathrm{PBC},i}$ of uncorrected rotational diffusion tensors of myoglobin and B-DNA, and mean rotational diffusion coefficients before and after correction with eq 3.26 , $D_{R,\mathrm{PBC}}$ and $D_{R,0}$, respectively. Comparisons to experiment require corrections for the low viscosity of TIP3P water, $D^W_{R,0} = D_{R,o}\eta^{\mathrm{TIP3P}}/\eta^W$, with $\eta^{\mathrm{TIP3P}} \approx 0.32\,\mathrm{mPa\,s}$[218] and $\eta^W \approx 0.9\,\mathrm{mPa\,s}$. All rotational diffusion coefficients are given in units of ns^{-1}. Numbers in parentheses indicate uncertainties in the last significant digit (standard error of the mean). Extrapolation to infinite box size was done by a one-parameter fit of the intercept $D_{R,0}$ in eq 3.26.

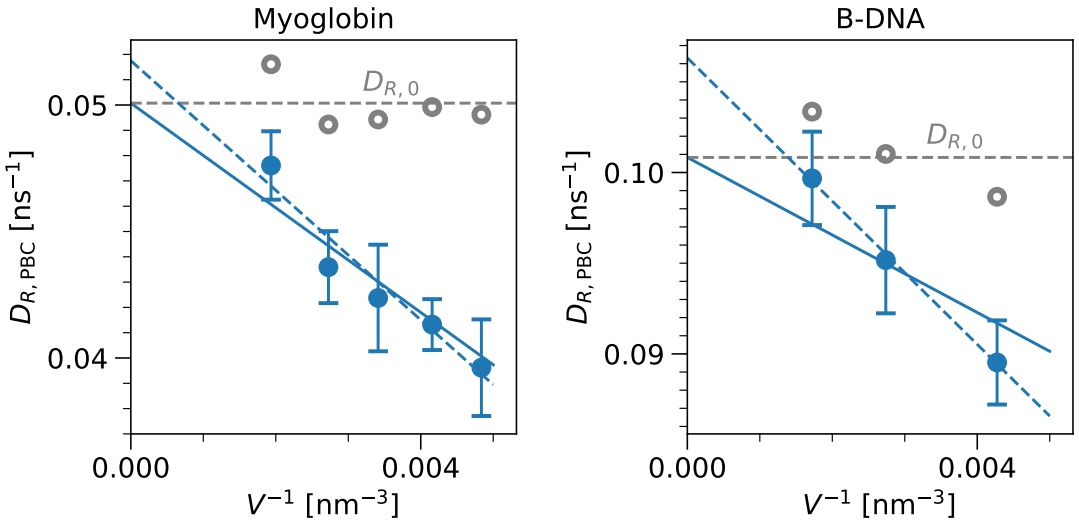

Figure 3.10: Mean rotational diffusion coefficients $D_{R,\mathrm{PBC}}$ from MD simulations of myoglobin and B-DNA as a function of the inverse box volume. Results obtained from the quaternion covariances averaged over all trajectories are shown as blue filled circles. Error bars indicate SEM, as estimated from repeated BD simulations using the best-fit model as input. Two-parameter (R_h, $D_{R,0}$) straight-line fits of eq 3.25 are shown as blue dashed lines. One-parameter ($D_{R,0}$) fits of eq 3.26 are shown as blue solid lines. Individual estimates of the infinite-system value $D_{R,0}$ from eq 3.26 are shown as open gray circles. Global estimates of $D_{R,0}$ from eq 3.26 are shown as horizontal gray dashed lines.

shape of B-DNA, which deviates noticeably from the sphere assumed in our hydrodynamic model.

To investigate the dependence on molecule shape, Figure 3.11 shows the eigenvalues $D_{R,\mathrm{PBC},i}$ $(i = 1, 2, 3)$ of the rotational diffusion tensors. The eigenvalues are in good agreement with eq 3.26, *i.e.*, $D_{R,i} \approx D_{R,\mathrm{PBC},i} + k_\mathrm{B}T/6\eta V$, albeit with larger scatter than the mean rotational diffusion coefficients in Figure 3.10.

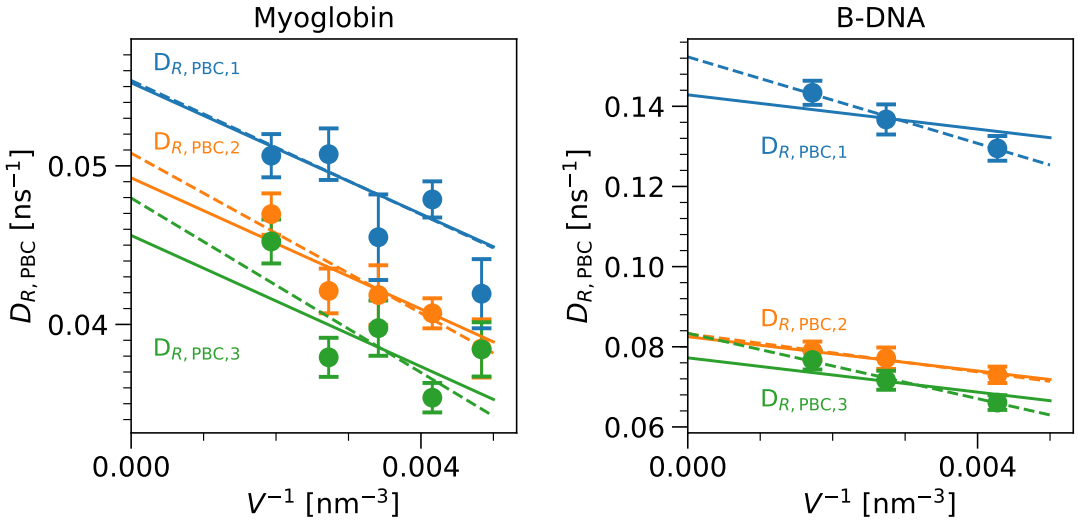

Figure 3.11: Inverse box-volume dependence of the eigenvalues $D_{R,\mathrm{PBC},i}$ of the rotational diffusion tensors of myoglobin and B-DNA. Values of $D_{R,\mathrm{PBC},i}$ are shown as filled circles with the SEM as error bars. Two-parameter (R_h, $D_{R,0}$) straight-line fits of eq 3.25 are shown as dashed lines. Single-parameter ($D_{R,0}$) fits of eq 3.26 are shown as solid lines.

The simulation results published by Takemura and Kitao for the third IgG-binding domain of Protein G (GB3) [203] and ubiquitin [203] at different box volumes and with different water models also follow our predictions eq 3.26. The rotational diffusion coefficients of GB3 were calculated from the simulation data as $D_R = 1/6\tau_2$, values for τ_2 are listed in Table 5 of ref 203. The mean rotational diffusion coefficient $D_{R,\mathrm{PBC}}$ of ubiquitin for different box volumes and water model was calculated from τ_2 with data from Takemura and Kitao [202]. When using a solvent viscosity of bulk water value of 0.85 mPa s, because the SPC/E$_b$ water model developed by Takemura and Kitao [203] captures the experimental translational diffusion coefficient, eq 3.26 can explain the simulation results of GB3 without fitting, Figure 3.12. For the experimental rotational diffusion coefficient, we used a value measured by NMR [61] and corrected for D_2O and temperature [232]. Ubiquitin [202] with three different water models follow the hydrodynamic model with a single hydrodynamic radius independent of water model as well, Figure 3.12. The absolute values of $D_{R,\mathrm{PBC}}$ for the different water models can be explained by a single R_h value using eq 3.26 with $D_{R,0,\mathrm{wat}} = k_\mathrm{B}T(8\pi\eta_\mathrm{wat}{R_h}^3)^{-1}$. For the SPC, SPC/E, and TIP3P water models, we used viscosities of $\eta_\mathrm{wat} = 0.43$ mPa s [70], 0.73 mPa s[218], and 0.32 mPa s [218], respectively. The hydrodynamic radius $R_h = 1.56(1)$ nm from our fit is in the range of

the experimental values of 1.57 nm from size-exclusion chromatography [214] and 1.65 nm from NMR [109, 207]. Note that the ubiquitin simulation data [202] include variations not only in box size, but also in box shape, from cubic to rectangular.

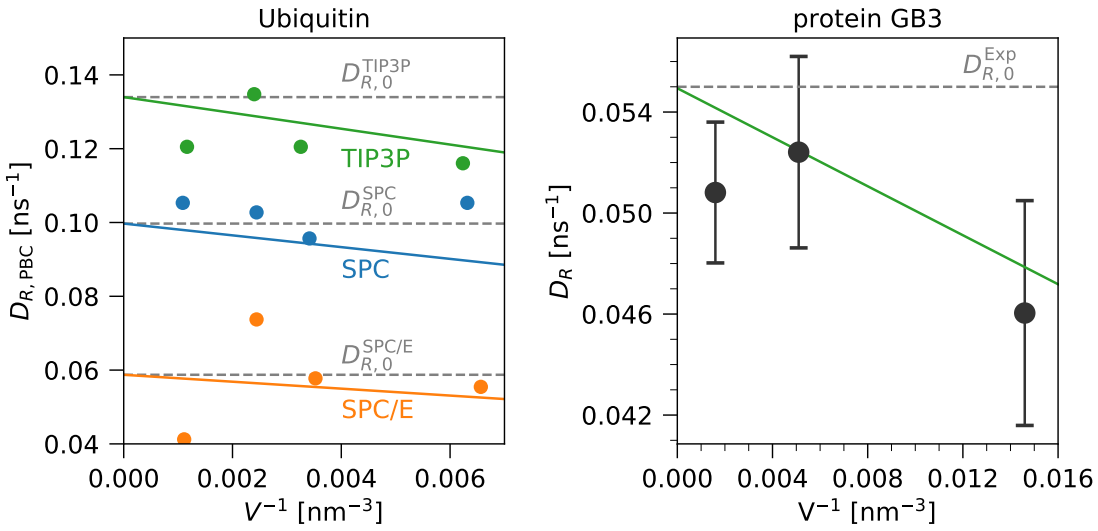

Figure 3.12: (Left) Mean rotational diffusion coefficient $D_{R,\mathrm{PBC}}$ of ubiquitin for different box volumes and water models. Data taken from Takemura and Kitao [202]. The solid lines are the result of a global single-parameter fit of R_h using eq 3.26. (Right) Rotational diffusion coefficients for protein GB3 calculated by Takemura and Kitao [203] (symbols). The predictions (solid green line) of the hydrodynamic model eq 3.26 using as input, without fit, the NMR value of D_R [61, 232] (horizontal dashed line).

3.5 Conclusion

In this chapter, we presented an algorithm to calculate the rotational diffusion tensor and principal axes for arbitrarily shaped bodies from molecular dynamics simulations, and showed that rotational diffusion is effected by PBC.

We demonstrated using MD simulation trajectories that our is able to reproduce experimental values and prediction from hydrodynamic theory for neat solvents. By formulating the problems in terms of quaternions the expressions we used are relatively simple and compact. The rotational diffusion tensor can be calculated from trajectory data using a straightforward χ^2 minimization or a using a Laplace transform. In practice, the straightforward χ^2 minimization led to an accurate and robust rotational diffusion tensor estimates. For the myoglobin and B-DNA test cases, the results are in excellent agreement with hydrodynamics calculations using HYDROPRO [147] and with experiment [23, 45, 224]. Therefore we can indeed estimate fully asymmetric rotational diffusion tensors from MD simulation trajectories. Since the algorithm is based on the theory of rigid-body motion, its application is not limited to all-atom simulations but it can be

applied to Brownian dynamics simulations and other CG simulations that include dynamics information [130, 133]. Our algorithm will also work in regimes where rotational and translational diffusion are coupled. A description in terms of the 6×6 diffusion matrix is not required because even for coupled rotation and translation the Green's function of the rotational dynamics alone, in an isotropic and homogeneous medium, reduces to that of the ideal Brownian rotor [13, 103].

Analysis of our simulations at different box volumes as well as literature values showed that rotational diffusion is affected by PBC. In the simplest form, eq 3.26, the infinite-system value is obtained by adding $k_{\mathrm{B}}T/6\eta V$ to the simulation estimate. We also developed a correction eq 3.26 that is independent of the size of the molecule. Our model assumes that the simulation box is spherical in shape. We expect that for rectangular boxes our simple correction is not correct anymore, similar to translational diffusion [221]. The same is true for highly anisotropic molecules. We noticed for the anisotropic B-DNA that our model still gives a good qualitative description of the behavior but our fit is not as good as for myoglobin. For myoglobin in a box of typical size filled with TIP3P water, the relative correction is about 20 % of the infinite-system value. Including finite-size corrections thus becomes important when MD simulations are used for quantitative comparisons to experiments sensitive to rotational dynamics, such as NMR or fluorescence. We showed here that MD simulations, after finite-size correction, produce hydrodynamic radii in excellent agreement with experiment.

Chapter 4

Brownian Dynamics Monte Carlo Integrators

4.1 Introduction

Langevin integrators for BD, like the Euler-Maruyama integrator eq 2.41, do not sample the equilibrium distribution. Markov chain Monte Carlo (MCMC) algorithms sample the equilibrium distribution by design and have been shown to simulate BD for isotropic particles [95, 104, 169, 179]. The MCMC algorithms adopted for BD will be called Brownian dynamics Monte Carlo (BDMC) for the remainder of this thesis. BDMC algorithms utilize that repeated convolutions of compact distributions tend towards Gaussians, according to the central limit theorem. To connect MCMC simulations with BD one has to match the width of the emerging Gaussian to that of the diffusion process. Therefore, the BDMC algorithm reproduces the correct equilibrium distribution *and* dynamics.

In this chapter, we extend the existing BDMC algorithms to include rotations and position-dependent diffusion. Adding rotational BD allows BDMC algorithms to be used with arbitrarily shaped molecules. Similar to translational BD we construct a rotational move in quaternion space that is connected to the quaternion covariances, eqs 2.69 to 2.73. We show that rotational BD can be reproduced in MCMC simulations using our rotational trial move. We will also look at the effect of different acceptance function for the Monte Carlo algorithm with respect to the dynamics, and extend the BDMC algorithms to simulate systems with position-dependent diffusion constants.

We will show two possible applications of our algorithm. The first is an application for a harmonic oscillator toy model with and without position-dependent diffusion coefficient. The second system is to calculate the binding rates of the UIM-1-ubiquitin and CUE-ubiquitin complexes using the KH protein model [96].

4.2 Theory

4.2.1 Brownian Dynamic Monte Carlo

Monte Carlo simulations dynamics correspond to to BD if the step-size is chosen correctly [95, 104, 169, 179]. To find the correct step-size for a given trial probability $T(x|y)$, to generate a step from y to x, and acceptance function $A(x|y)$, to accept a trial step, we

look at the mean square displacement (MSD) of a single Monte Carlo step under constant force $\Delta U = -Fx$. For an arbitrary trial function and acceptance function, the MSD at any point y is given by

$$MSD = \left\langle (x - \langle x \rangle)^2 \right\rangle \tag{4.1}$$

$$= \int dx\ x^2 \left[T(x|y)A(x|y) + \delta(x-y)(1 - A(x|y)) \right], \tag{4.2}$$

with $\delta(x)$ Dirac's delta function. The term $\delta(x-y)(1-A(x|y))$ represents the probability that the step was rejected. Using a uniform box distribution as the trial probability, with width $2a$,

$$T(x|y) = \begin{cases} \frac{1}{2a} & \text{if } |x-y| \leq a \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

and the Metropolis acceptance function for sampling from the Boltzmann distribution [134]

$$A(x|y) = \min\left[1, \exp\left(-\frac{U(x) - U(y)}{k_{\mathrm{B}}T}\right)\right], \tag{4.4}$$

with $U(x)$ the potential energy at position x. Without loss of generality we assume $y = 0$. Then the MSD, eq 4.1, simplifies to

$$MSD = \int_{-a}^{a} \mathrm{d}x\ x^2 \frac{1}{2a} \min\left[1, \exp\left(\frac{Fx}{k_{\mathrm{B}}T}\right)\right]$$

$$= \frac{a^2}{3} + \mathcal{O}(a^3). \tag{4.5}$$

In the second equality the integral has been approximated for small step-sizes $a \ll 1$. The MSD of a Brownian process in one dimension with a diffusion coefficient D and a time-step dt is [48]

$$MSD(dt) = 2D_T dt, \tag{4.6}$$

with D_T the one dimensional translational diffusion constant. Replacing the left hand side of eq 4.6 with the results of eq 4.5 gives an expression for the step-size depending on the diffusion coefficient and time-step

$$a = \sqrt{6D_T dt}. \tag{4.7}$$

This result can be extended to three dimensions in the coordinate system in which the translational diffusion tensor is diagonal.

This derivation is only valid for translational moves and dynamics. For rotations the quaternion covariance of Fravo [51] fully determines the diffusive behavior of rigid bodies. To connect rotational trials moves to the quaternion covariances we construct a

rotational move in quaternion space. From the BD rotational diffusion algorithm developed in Chapter 3 it is known that small rotations in quaternion space can be generated using

$$\tilde{q}_i(dt) = a_i g_i, \ i = 1, 2, 3 \tag{4.8}$$

$$\tilde{q}_0(dt) = \left(1 - \sum_{i=1}^{3} a_i^2\right)^{1/2}, \tag{4.9}$$

with g_i being uncorrelated Gaussian random numbers of zero mean and unit variance. This algorithm ensures that on average the generated quaternions are normalized

$$\left\langle \sum_{i=0}^{3} \left(\tilde{q}_i(dt)\right)^2 \right\rangle \approx 1. \tag{4.10}$$

It is recommended to enforce the normalization condition for every generated quaternion

$$q_i'(dt) = \tilde{q}_i(dt) \left(\sum_{i=0}^{3} \tilde{q}_i(dt)^2\right)^{-1/2}. \tag{4.11}$$

A similar algorithm is possible if g_i are uniformly distributed random numbers between -1 and 1. In this case, we replace eq 4.9 with

$$\tilde{q}_0(dt) = \left(1 - \frac{1}{3}\sum_{i=1}^{3} a_i^2\right)^{1/2}. \tag{4.12}$$

It should be noted that in both cases there exists an upper limit for a_i such that $\tilde{q}_0(dt)$ is not an imaginary number. The quaternion covariance, eqs 2.69 to 2.73, can be approximated at short times $dt \ll 1$

$$\langle q_0^2(dt)\rangle = 1 - \frac{dt}{2}(D_{R,1} + D_{R,2} + D_{R,3}) + \mathcal{O}(dt^2) \tag{4.13}$$

$$\langle q_i^2(dt)\rangle = \frac{1}{2} D_{R,i} dt + \mathcal{O}(dt^2) \quad \text{for } i = 1, 2, 3 \tag{4.14}$$

$$\langle q_i(dt) q_j(dt)\rangle = 0 \quad \text{for } i \neq j. \tag{4.15}$$

This approximation, eq 4.13, corresponds to the trial move for rotations with Gaussian random numbers and $a_i = (D_{R,i}t/2)^{1/2}$. For the rotation trial move with uniform random numbers the step-size is

$$a_i = \left(\frac{3}{2} D_{R,i} dt\right)^{1/2}. \tag{4.16}$$

A possible improvement of the previous algorithms would be to use the full quaternion covariance to calculate the step-width for any time-step t. In Figure 4.1 we compare the full quaternion covariance with its short time approximation for an isotropic rotation diffusion tensor $D_R = [0.1, 0.1, 0.1]1/\mathrm{s}$. For a time-step $dt = 0.1\,\mathrm{s}$ the error of the approximation is about 1 %. Cautiously one would choose a time-step one or two orders of magnitude below

the value of the rotational diffusion coefficients. Eq 4.13 is also computationally cheaper than using the full quaternion covariance which would require the expensive calculations of three exponential functions.

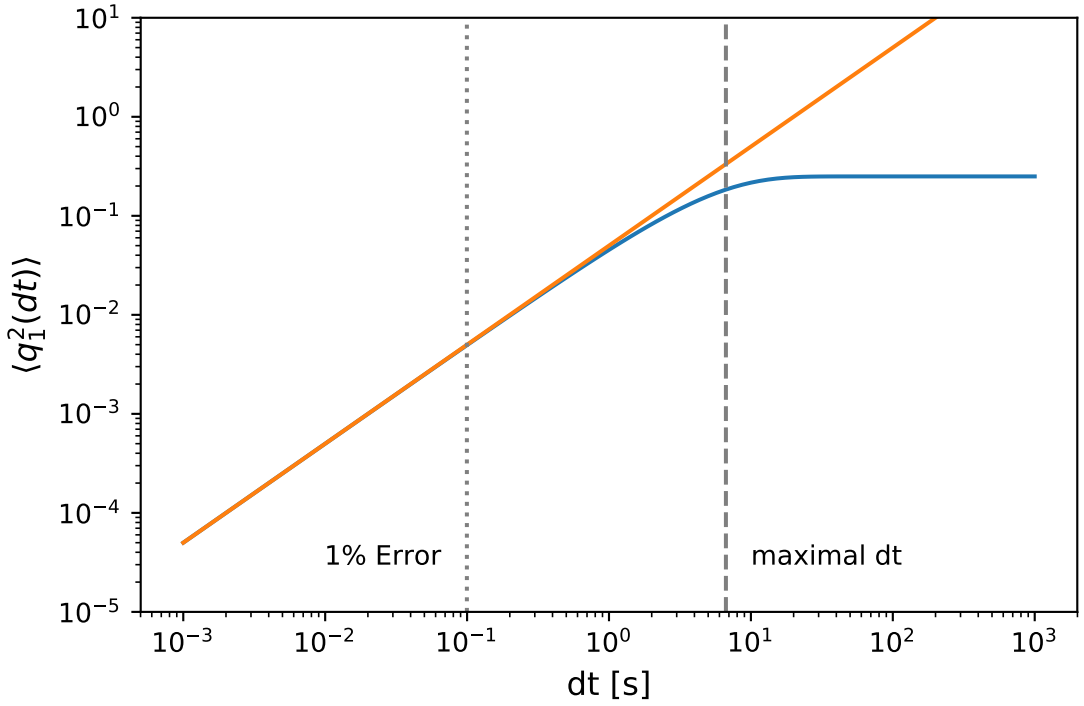

Figure 4.1: Quaternion Covariance (blue) and the short time approximation (orange) for $\langle q_1^2(dt) \rangle$ with $D_R = [0.1, 0.1, 0.1]1/\text{s}$. Also shown are the time-step at which the difference between the exact solution and the approximation is 1%, and the largest usable time-step so that $\tilde{q}_0(dt) \in \mathbb{R}$.

4.2.2 Combined Translation and Rotation Brownian Dynamics Monte Carlo

To simulate translation and rotation together we have to consider that the our derived algorithms are only valid in the respective PCS, i.e. the respective diffusion tensors are diagonal. Therefore, to simulate both the translation and rotation trial steps have to be rotated from their respective PCSinto a shared RCS and the implementation has to include book-keeping to remember the current orientation in respect to the PCS at start of the simulation. To explain the necessary book-keeping we introduce:

- $\boldsymbol{q}_t$ is the quaternion describing the orientation in the rotation diffusion PCS at time t.
- $\boldsymbol{Q}_t$ is a rotation matrix describing the orientation in the rotation diffusion PCS at time t.

- $\boldsymbol{Q}_t^{\mathrm{RCS}}$ is a rotation matrix describing the orientation in the RCS at time t.
- $\boldsymbol{M}^R$ is the transformation from the rotation diffusion PCS to the RCS at time 0.
- $\boldsymbol{M}^T$ is the transform from translation diffusion PCS to the RCS at time 0.

For a simulation $\boldsymbol{M}^R$ and $\boldsymbol{M}^T$ have to be known beforehand, $\boldsymbol{Q}_t$ has to be continuously updated, and $\boldsymbol{Q}_t^{\mathrm{ref}}$ can be calculated

$$\boldsymbol{Q}_t^{\mathrm{RCS}} = \boldsymbol{M}^R \boldsymbol{Q}_t \left(\boldsymbol{M}^R\right)^T. \tag{4.17}$$

In the translation step the displacement vector dx' is calculated in the translation PCS and converted into the RCS before being applied

$$dx = \boldsymbol{Q}_t^{\mathrm{RCS}} \boldsymbol{M}^T dx'. \tag{4.18}$$

The displacement can be directly applied to the current coordinates. In the rotation step the small rotation $d\boldsymbol{q}$ is calculated in the rotation PCS and the orientation in the rotation PCS is updated $\boldsymbol{q}_t = \boldsymbol{q}_{t-1} d\boldsymbol{q}$. Afterwards the new rotation is converted into the RCS using eq 4.17. To minimize numerical errors applying the new orientation it is beneficial rotate the coordinates from the initial positions, centered at the origin, with $\boldsymbol{Q}_t^{\mathrm{RCS}}$ and then translate them by the current center of geometry.

Special care has to be taken to ensure that BDMC simulations fulfill detailed balance and that sweeps correspond to a time-step. During a trial move in a BDMC simulation, the protein has to be rotated and translated. To ensure detailed balance we randomly choose with equal probability if the protein is first rotated or translated. For a Monte Carlo sweep to be counted as a single time-step a sweep has to generate a trial move for all N proteins. To ensure detailed balance we randomly choose with equal probability the order in which proteins are updated.

4.2.3 Brownian Dynamics Monte Carlo with Gaussian Trial probabilities

Because Langevin integrators use the Gaussian distribution to generate random displacements an obvious improvement to the Brownian dynamics Monte Carlo algorithm could be to use a Gaussian distribution for the trial probability

$$T(x|y) = \left(2\pi a^2\right)^{-1/2} e^{-\frac{(x-y)^2}{2a^2}}, \tag{4.19}$$

here a is the width of the Gaussian distribution. With this trial probability the MSD of a single Monte Carlo trial step, eq 4.1, is

$$MSD = a^2. \tag{4.20}$$

Therefore the relationship between step-size, diffusion coefficient and time-step for translation is

$$a = \sqrt{2D_T dt} \tag{4.21}$$

and for rotation

$$a_i = \sqrt{\frac{D_{R,i} dt}{2}} \tag{4.22}$$

4.2.4 Alternative Acceptance Function

So far we only considered the Metropolis acceptance function. The question arises if the Metropolis acceptance function is the optimal choice. An alternative to the Metropolis acceptance function is the Glauber acceptance function [63]

$$A(x|y) = \frac{1}{1 + e^{\frac{U(y)-U(x)}{k_B T}}}. \tag{4.23}$$

This acceptance function is also known as the Barker rule [10] in the literature.

As a measure for the accuracy of the simulation we look at the first, second and third moment of the probability distribution for a Monte Carlo step under constant force. For BD with diffusion coefficient D starting from $x = 0$ and evolving for time dt under constant force F, these moments are $\langle x \rangle = DFdt$, $\langle (x - \langle x \rangle)^2 \rangle = 2Ddt$, and $\langle (x - \langle x \rangle)^3 \rangle = 0$. The general equation for the n-th moment is

$$\langle (x - \langle x \rangle)^n \rangle = \int dx \; x^n A(-Fx|0) T(x|0), \tag{4.24}$$

Our assumption is that a better reproduction of the moments will also yield a better description of the dynamics. The error of dynamic observables, like the MSD and quaternion covariance is determined by the fourth order moments. Therefore a better accuracy in higher order moments should yield a faster convergence of dynamic observables. We use a uniform box distribution as the trial probability eq 4.3 and want to know the accuracy of the moments with respect to the step-size a. For Metropolis, eq 4.4, we obtain,

$$\mu = \langle x \rangle = -\frac{Fa^2}{6} + \frac{F^2 a^3}{16} + \mathcal{O}(a^4) \tag{4.25}$$

$$\langle (x - \mu)^2 \rangle = \frac{a^2}{3} - \frac{Fa^3}{8} + \mathcal{O}(a^4) \tag{4.26}$$

$$\langle (x - \mu)^3 \rangle = \frac{Fa^4}{15} + \mathcal{O}(a^5). \tag{4.27}$$

Here the second central moment is correct to order $\mathcal{O}(a^3)$ and the third central moment

is correct to order $\mathcal{O}(a^4)$. For Glauber, eq 4.23, we obtain,

$$\mu = \langle x \rangle = \frac{Fa^2}{12} - \frac{F^3a^4}{240} + \mathcal{O}(a^5) \tag{4.28}$$

$$\langle (x-\mu)^2 \rangle = \frac{a^2}{6} - \frac{F^2a^4}{144} + \mathcal{O}(a^5) \tag{4.29}$$

$$\langle (x-\mu)^3 \rangle = \frac{Fa^4}{120} + \mathcal{O}(a^5). \tag{4.30}$$

Where both the second and third central moment are correct to order $\mathcal{O}(a^4)$.

Metropolis and Glauber are not the only possible acceptance function that fulfill detailed balance

$$A(x|y) = \frac{\pi(x)}{\pi(y)} A(y|x). \tag{4.31}$$

For convenience we use $\Delta U = U(x) - U(y)$ as the variable for the acceptance function $A(\Delta U)$, the Boltzmann distribution as equilibrium probability $\pi(x) = 1/Z \exp(-\frac{U(x)}{k_\mathrm{B}T})$, with Z the partition function, and rewrite equation 4.31 as

$$A(\Delta U) = e^{-\beta \Delta U} A(-\Delta U) \tag{4.32}$$

$$A(\Delta U) e^{-1/2 \beta \Delta U} = e^{-1/2 \beta \Delta U} A(-\Delta U). \tag{4.33}$$

The Ansatz

$$A(\Delta U) = \left(\frac{h_2}{2} \Delta U^2 + h_0 \right) e^{-\frac{\beta \Delta U}{2}}, \tag{4.34}$$

with free parameters h_0 and h_2, fulfills condition eq 2.53 and therefore preserves detailed balance. If we approximate the resulting moments for $a \ll 1$ we obtain,

$$\mu = \langle x \rangle = a^2 \frac{F h_0}{6} + a^4 F^3 \left(\frac{h_0}{240} + \frac{h_2}{20} \right) + \mathcal{O}(a^6) \tag{4.35}$$

$$\langle (x-\mu)^2 \rangle = a^2 \frac{h0}{3} + a^4 F^2 \left(\frac{h_2}{10} - \frac{h_0^2}{36} + \frac{h_0}{40} \right) + \mathcal{O}(a^6) \tag{4.36}$$

$$\langle (x-\mu)^3 \rangle = a^4 F \left(\frac{h_0}{10} - \frac{h_0^2}{6} \right) + \mathcal{O}(a^6). \tag{4.37}$$

To achieve an accuracy to order $\mathcal{O}(a^6)$ the free parameters have to be chosen so that all fourth order terms are zero. The only non trivial solution is $h_0 = 3/5$ and $h_2 = -1/20$. Leading to an acceptance function

$$A(\Delta U) = \left(\frac{3}{5} - \frac{\Delta U^2}{40} \right) e^{-\frac{\Delta U}{2}}. \tag{4.38}$$

Because this solution can have values larger than 1 or smaller than zero, Figure 4.2, the

probability has to be clamped to stay between zero and one

$$A(\Delta U) = \begin{cases} 1 & \text{for } \Delta U < U' \\ \left(-\frac{\Delta U^2}{40} + \frac{3}{5}\right) e^{-\frac{\Delta U}{2}} & \text{for } |\Delta U| < |U'| \\ e^{-\Delta U} & \text{otherwise} \end{cases}, \tag{4.39}$$

with $U' = -1.13124207139993$ the solution of $\left(-\frac{\Delta U^2}{40} + \frac{3}{5}\right) e^{-\frac{\Delta U}{2}} = 1$. To ensure that detailed balance is fulfilled the function applied to clamp eq 4.38 has to fulfill detailed balance as well and it has to be clamped symmetrically around zero. Note that we have here clamped using the Metropolis acceptance function.

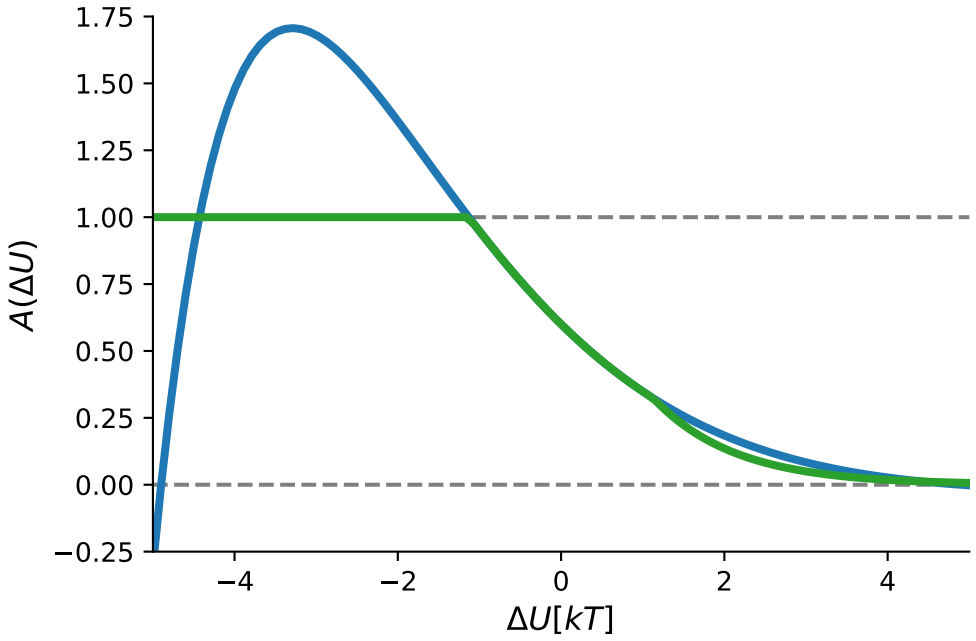

Figure 4.2: Optimized acceptance function eq 4.38 (blue) and the clamped probability eq 4.39 (green).

In the derivation of eq 4.39 we used a uniform distribution for the trial probability. The question arises if the Gaussian distribution can also be used to derive a similar acceptance function. Using a Gaussian probability for the first three moments we obtain

$$\mu = \langle x \rangle = a^2 \frac{F h_0}{2} + a^4 F^3 \left(\frac{h_0}{16} + \frac{3h_2}{4}\right) + \mathcal{O}(a^6) \tag{4.40}$$

$$\langle (x-\mu)^2 \rangle = a^2 h_0 + a^4 F^2 \left(\frac{3h_2}{2} - \frac{h_0^2}{4} + \frac{3h_0}{8}\right) + \mathcal{O}(a^6) \tag{4.41}$$

$$\langle (x-\mu)^3 \rangle = \frac{3h_0}{2} F a^4 (-h_0 + 1) + \mathcal{O}(a^6) \tag{4.42}$$

with a the standard deviation of the normal distribution. The only non trivial solution to

remove the fourth order terms is $h_0 = 1$ and $h_2 = -1/12$

$$A(\Delta U) = \left(-\frac{\Delta U^2}{24} + 1\right) e^{-\frac{\Delta U}{2}}. \tag{4.43}$$

This equation also has the problem that it can have values below zero and above one. Unfortunately, eq 4.43 is equal one at $\Delta U = 0$ making it impossible clamp the probability symmetrically around zero. It is therefore not possible to construct an explicit optimized acceptance function for a Gaussian trial probability. Note that eq 4.39 can still be used with a Gaussian trial probability.

For both the Glauber, eq 4.23 and the derived optimal acceptance function eq 4.39 the approximations for the second moment can be used to determine the step-size a of the box distribution to reproduce BD for translation, and rotation Table 4.1.

Translation	Metropolis	Glauber	Optimal
Uniform	$\sqrt{6Ddt}$	$\sqrt{12Ddt}$	$\sqrt{10Ddt}$
Gaussian	$\sqrt{2Ddt}$	$\sqrt{4Ddt}$	$\sqrt{\frac{10}{3}Ddt}$
Rotation	Metropolis	Glauber	Optimal
Uniform	$\sqrt{\frac{3}{2}D_{R,i}dt}$	$\sqrt{3D_{R,i}dt}$	$\sqrt{\frac{5}{2}D_{R,i}dt}$
Gaussian	$\sqrt{\frac{1}{2}D_{R,i}dt}$	$\sqrt{D_{R,i}dt}$	$\sqrt{\frac{5}{6}D_{R,i}dt}$

Table 4.1: MC trial-move width a for translation and rotation and different acceptance functions. Trial displacements are uniformly distributed in the interval $[-a, a]$.

4.2.5 Effective Time-Step

In the derivation of the BDMC algorithm we assumed that trial moves are small and therefore $\Delta U \approx 0$. A side effect of this assumption is that the trial move size is optimized for an average acceptance rate of $\mathrm{acc}_{\mathrm{ideal}} = A(\Delta U = 0)$. This assumption is not necessarily true for large trial moves, yielding a distortion of the dynamics. We suggest to calculate dynamic observables with an effective time-step dt_e

$$dt_e = dt \frac{\mathrm{acc}_{\mathrm{sim}}}{\mathrm{acc}_{\mathrm{ideal}}}, \tag{4.44}$$

where $\mathrm{acc}_{\mathrm{sim}}$ is the observed acceptance rate. For the Metropolis acceptance function and translational diffusion, it has been shown that re-scaling the time-step dt by $\mathrm{acc}_{\mathrm{sim}}/\mathrm{acc}_{\mathrm{ideal}}$ improves the accuracy of dynamic observables [179].

4.2.6 Position Dependent Diffusion Brownian Dynamic Monte Carlo

For some applications of BD it is useful to use a position dependent diffusion coefficient [81]. In the derivation of Monte Carlo from the detailed balance condition, eq 2.49, with the separation of the transition probability into a probability to accept a step $A(x|y)$ and a probability to generate a step $T(x|y)$ we find that the acceptance function $A(x|y)$ and $A(y|x)$ are connected via [69]

$$A(x|y) = r(x|y)A(y|x) \text{ with} \tag{4.45}$$

$$r(x|y) = \frac{T(y|x)\pi(x)}{T(x|y)\pi(y)}. \tag{4.46}$$

If we make the Ansatz that $T(x|y)$ is a Gaussian probability with the variance depending on a position dependent diffusion coefficient $D(x)$

$$T(x|y) = (4\pi a D(y))^{-1/2} e^{-\frac{(x-y)^2}{4aD(y)}}, \tag{4.47}$$

with a the step-size. Then we get

$$r(x|y) = \left(\frac{D(y)}{D(x)}\right)^{1/2} e^{-\beta[U(x)-U(y)]-\frac{(x-y)^2}{4aD(y)}+\frac{(x-y)^2}{4aD(x)}} \tag{4.48}$$

$$= \exp\left[-\beta[U(x)-U(y)] + \frac{1}{2}\log\left(\frac{D(y)}{D(x)}\right) + \frac{(x-y)^2}{4a}\left(\frac{1}{D(y)} - \frac{1}{D(x)}\right)\right] \tag{4.49}$$

$$= \exp\left[V(x,y,a)\right]. \tag{4.50}$$

$V(x,y,a)$ is a new effective potential that can be used with the previously defined acceptance function, instead of $-\Delta U/k_{\mathrm{B}}T$. Because $V(x,y,a)$ is derived from a general expression independent of the acceptance function it fulfills detailed balance.

To connect the step-size a to the chosen time-step dt the same approach as before can be used. We calculate the MSD for a single Monte Carlo step at position x and compare it with the expected analytical result at x. The resulting values for the step-size are shown in Table 4.2. Here the step-size is only depending on the time-step because the local diffusion coefficient is part of the trial probability.

	Metropolis	Glauber	Optimal
a	1	2	5/3

Table 4.2: Move width a to choose for BDMC with the position-dependent diffusion and Gaussian trial moves, eq 4.47, for different acceptance functions.

This algorithm can be generalized to dimensions larger than one if we work in the coordinate system in which the diffusion tensor is diagonal. In n dimensions, we make

multivariate trial moves

$$T(\boldsymbol{x}|\boldsymbol{y}) = (4\pi dt)^{-n/2}|\boldsymbol{D}(\boldsymbol{y})|^{-1/2}\exp\left(-\frac{\boldsymbol{z}^T(\boldsymbol{D}(\boldsymbol{y}))^{-1}\boldsymbol{z}}{4dt}\right) \tag{4.51}$$

where $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{y}$ and $\boldsymbol{D}(\boldsymbol{y})$ is the covariance matrix and $|\boldsymbol{D}|$ is the determinant of the local diffusion tensor. To make a move in n dimensions we first evaluate $\boldsymbol{D}(\boldsymbol{y})$ to determine the diffusion coefficient along each of the principal directions and then make Gaussian trial moves in each of the principal directions according to eq 4.47 with the parameter a chosen according to Table 4.2

4.3 Methods

4.3.1 Harmonic Oscillator

As a simple toy model to test the BDMC algorithms we use a harmonic oscillator $U(x) = \frac{1}{2}kx^2$ with spring constant k. For comparison to our algorithm we use the Euler-Maruyama eq 2.41 , and BAOAB eq 2.42 integrators. For simplicity the values of D, k and $k_\mathrm{B}T$ have been set to one.

To evaluate how well the equilibrium distribution is reproduced, we compare the simulation result to the analytical MSD,

$$MSD(t) = \frac{k_\mathrm{B}T}{k}\left[1 - e^{-\frac{2kD}{k_\mathrm{B}T}t}\right] \tag{4.52}$$

The MSD has been calculated for different time-steps from 0.0001 s to 0.25 s at $t = 10$ s. For each time-step and algorithm 100000 simulations have been performed.

To evaluate how well the dynamics are reproduced, we calculated the autocorrelation time τ of the positions

$$ACF_x(t) = e^{-t/\tau} \tag{4.53}$$

$$\tau = \frac{k_\mathrm{B}T}{Dk}. \tag{4.54}$$

To estimate the autocorrelation time we averaged the autocorrelation function (ACF) of forty trajectories each 1000 s long, and fitted a straight line to the logarithm of the ACF using a least squares fit. This procedure was repeated fifty times to obtain a mean, and error of the mean. To asses the systematic deviation of different algorithms from the expected solution τ_0 we fit the calculated difference $\tau - \tau_0$ with respect to the time-step to

$$\tau - \tau_0 = \alpha dt^\gamma \tag{4.55}$$

with the fit parameter α and γ. For our system $\tau_0 = 1$.

Our BDMC algorithms, the Euler-Maruyama, and BAOAB integrators have been im-

plemented in julia v0.6 [21].

4.3.2 Position Dependent Diffusion

To validate the position dependent diffusion algorithm and the effective potential $V(x, y, a)$ we transformed the harmonic oscillator into a new coordinate z, where D is position dependent, with the following transformation

$$z(x) = 2\exp\left(\frac{x}{2}\right) \tag{4.56}$$

$$x(z) = 2\log\left(\frac{z}{2}\right). \tag{4.57}$$

The potential and diffusion coefficient in z can be determined with [18]

$$U(z) = U(x(z)) + k_{\mathrm{B}}T\ln|dz/dx| \tag{4.58}$$

$$D(z) = D(x(z))(dz/dx)^2, \tag{4.59}$$

yielding

$$D(z) = \frac{z^2}{4} \tag{4.60}$$

$$U(z) = 2\log\left(\frac{2}{z}\right)^2 + \log\left(\frac{z}{2}\right). \tag{4.61}$$

Because the transformation preserves the dynamics we will perform the simulations in z-space and analyze the MSD and autocorrelation time after transforming the trajectories back into x. After the back transformation the simulation procedures to calculate the MSD and autocorrelation time are the same as for the harmonic oscillator. Because the diffusion is position dependent the Euler-Maruyama integrator changes to [178]

$$x_{i+1} = x_i + \left[F(x_i)D(x_i) + D'(x_i)\right]dt + \sqrt{2D(x_i)dtk_{\mathrm{B}}T}R_t, \tag{4.62}$$

with $D'(x)$ being the derivative of the diffusion coefficient. We adjusted the BAOAB integrator in the same way as the Euler-Maruyama integrator by adding the derivative of the diffusion coefficient

$$x_{i+1} = x_i + \left[F(x_i)D(x_i) + \frac{3}{4}D'(x_i)\right]dt + \sqrt{2D(x_i)dtk_{\mathrm{B}}T}\frac{1}{2}(R_t + R_{t-1}), \tag{4.63}$$

We determined the factor $\frac{3}{4}$ for the derivative of D empirically by minimizing the error of the equilibrium probability for a wide range of time-steps. This algorithm is called "xBAOAB" in this thesis.

The BDMC algorithms for position-dependent diffusion, and the Euler-Maruyama and BAOAB integrator have been implemented in julia v0.6 [21].

4.3.3 Binding of the Vps27 UIM-1-ubiquitin complex

We looked at the binding kinetics of the Vps27 UIM-1-ubiquitin complex. For the simulations we took the PDB structure 1Q0W [198] and placed the two proteins into a cubic box, separated by a distance of 75 Å. We used 6 different box sizes with edge-lengths 125, 150, 175, 200, 225 and 300 Å. The diffusion tensor, used in the simulation, was calculated with HYDROPRO [147] for each protein using the 'shell' model, with the recommended bead radii of 4.8 Å, using a temperature of 300 K and a viscosity of 1 mPa s. Translation diffusion coefficient of UIM-1 in the reference frame of the crystal struture

$$D_T = \begin{bmatrix} 1.9 \times 10^{-1} & 1.3 \times 10^{-2} & 6.4 \times 10^{-4} \\ 1.3 \times 10^{-2} & 1.7 \times 10^{-1} & 1.1 \times 10^{-3} \\ 6.5 \times 10^{-4} & 1.1 \times 10^{-3} & 1.6 \times 10^{-1} \end{bmatrix} \mathrm{nm^2 ns^{-1}}. \tag{4.64}$$

Rotational diffusion coefficient of UIM-1 in the reference frame of the crystal struture

$$D_R = \begin{bmatrix} 1.2 \times 10^{-1} & 4.4 \times 10^{-2} & 4.5 \times 10^{-3} \\ 4.4 \times 10^{-2} & 7.4 \times 10^{-2} & 2.6 \times 10^{-3} \\ 4.5 \times 10^{-3} & 2.6 \times 10^{-3} & 4.9 \times 10^{-2} \end{bmatrix} \mathrm{ns^{-1}} \tag{4.65}$$

The translation diffusion coefficient of ubiquitin in the reference frame of the crystal struture

$$D_T = \begin{bmatrix} 1.3 \times 10^{-1} & 3.8 \times 10^{-3} & -5.5 \times 10^{-4} \\ 3.8 \times 10^{-3} & 1.3 \times 10^{-1} & -4.3 \times 10^{-4} \\ -5.4 \times 10^{-4} & -4.5 \times 10^{-4} & 1.3 \times 10^{-1} \end{bmatrix} \mathrm{nm^2 ns^{-1}} \tag{4.66}$$

Rotational diffusion coefficient of ubiquitin in the reference frame of the crystal struture

$$D_R = \begin{bmatrix} 3.8 \times 10^{-2} & 4.6 \times 10^{-3} & -4.4 \times 10^{-4} \\ 4.6 \times 10^{-3} & 3.0 \times 10^{-2} & -4.7 \times 10^{-4} \\ -4.4 \times 10^{-4} & -4.7 \times 10^{-4} & 2.8 \times 10^{-2} \end{bmatrix} \mathrm{ns^{-1}} \tag{4.67}$$

As forcefield we used the KH model [96] with the complexes-pp simulation engine. Simulations have been performed with a uniform trial probability, and all three acceptance function, Metropolis, Glauber, and Dynamic, and with six different integration time-steps of 0.05, 0.1, 1, 10, 20, and 100 ps. For each unique combination of integration time-step, acceptance function, and box size we ran 100 individual simulations each 8 μs long at a temperature of 300 K and a Debye length of 10 Å corresponding to a salt concentration of 100 mM NaCl. Structures were recorded every 400 ps. For the simulations with a 300 Å edge-length we simulated for 40 μs and only with a time-step of 1 ps or larger. All simulations start from the same initial configuration with different random number seeds. The first 80 ns of each simulation is equilibration and is discarded from further analysis. To test the influence of electrostatic interaction on the binding rates we repeated all simulations described above also with the Debye length set to 1 Å.

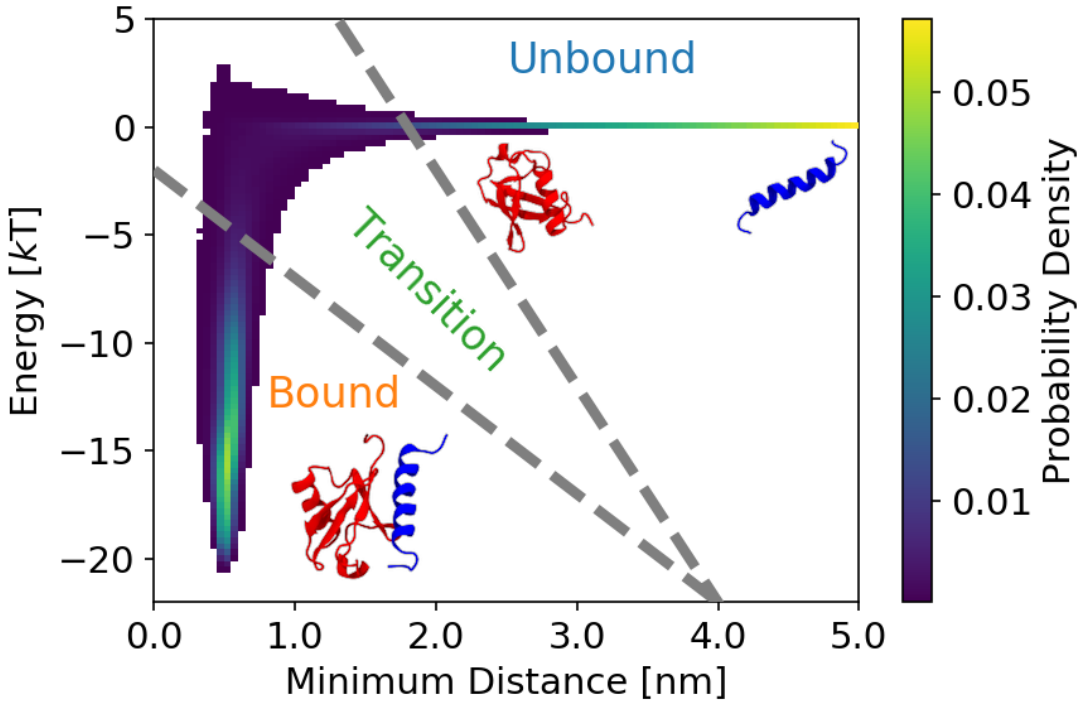

Figure 4.3: Probability density of the UIM-1-ubiquitin complex for the minimal distance between two C_α atoms and the total energy. The bound, unbound and transition region are marked as well.

The bound and unbound state for the UIM-1-ubiquitin complex are defined using the potential energy and the minimum distance between two C_α atoms, Figure 4.3. To reduce noise due do frequent recrossing at state boundaries we use transition based assignment (TBA) [27]. In TBA an additional transition region is defined where it is initially not clear in which state the system is. All time continuous paths in the transition region are initially assumed to be transition paths. A transition path connects two states without visiting another in between. In an assignment step, the conformations on the transition paths are assigned to their respective states. The first half is assigned to the initial state and the second half to the final state. Any path in the transition region that is not a transition path will be assigned to the initial state. See Figure 4.4 for a sketch representation of the algorithm. The state assignment is shown in Figure 4.3. Note that this definition makes no distinction between specifically and non-specifically bound states.

Using the state assignments after applying the TBA filter we calculate the rates k_{ij} from state i to j using the observed transitions N_{ij}. Detailed balance, and microscopic time reversibility, require that for a long trajectory the transition matrix is symmetric. To enforce detailed balance, we symmetrize the matrix of transitions $N_{ij}^{\text{sym}} = 1/2(N_{ij} + N_{ji})$. Let T_i be the total time the system spends in state i. The rates are calculated as

$$k_{ij} = \frac{N_{ij}^{\text{sym}}}{T_i} \tag{4.68}$$

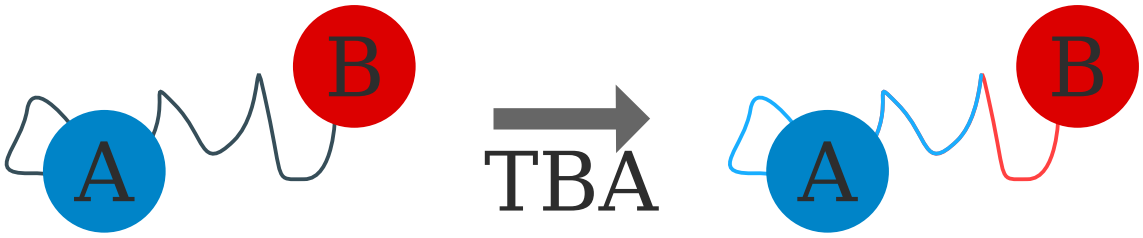

Figure 4.4: Example of state assignment of conformations on a transition path with TBA. The two states are called A and B with their corresponding regions colored blue and red. Anything between is the transition region. The left side shows candidate transition paths before TBA. The right side shows the state assignment along the candidate transition paths after TBA.

for $i \neq j$ and

$$k_{ii} = -\sum_{\substack{j=1 \\ j \neq i}}^{N} k_{ij} \tag{4.69}$$

for $i = j$. This estimator has been recently shown to be the maximum likelihood estimator for k_{ij} [195]. We estimate the error of the rates using bootstrapping with fifty samples from the one hundred trajectories for each unique combination of time-step and acceptance function. Each bootstrap sample contains fifty trajectories. To convert the pseudo first-order on-rate into k_{on} we use eq 4.71, and eq 4.44 to correct for the effective time-step.

The binding kinetics of the two-state model, with the reactants A and B, follow a pseudo first order rate equation

$$A + B \underset{k_{\text{off}}}{\overset{k'_{\text{on}}}{\rightleftharpoons}} AB, \tag{4.70}$$

where k_{off} is the off-rate, k'_{on} is a pseudo first-order on-rate, and AB is the bound state. The true on-rate can be determined with [149]

$$k_{\text{on}} = \frac{k'_{\text{on}}}{[B]} = k'_{\text{on}} \frac{V N_A}{p_A}, \tag{4.71}$$

with $[B]$ denoting the concentration of reactant B, N_A the Avogadro constant, $V = (V_{\text{Box}} - V_{\text{ex}})$ the volume accessible to the unbound components, V_{Box} the simulation box volume, V_{ex} the excluded volume of the bound state, and p_A the probability to be unbound. Here we approximated the concentration $[B] = \frac{p_B}{N_A V}$ using the activity of B. To determine V_{ex} and the dissociation constant K_d we can use that the fraction bound p_{AB} is given by

$$p_{AB} = \frac{[A]}{[A] + K_d} \tag{4.72}$$

$$= \frac{-p_{AB} + 1}{K_d N_A V - p_{AB} + 1}. \tag{4.73}$$

In the second equality we used $[A] = \frac{1-p_{AB}}{N_A V}$. This equation can be solved for p_{AB}

$$p_{AB} = \frac{K_d N_A \left(V_{\text{Box}} - V_{\text{ex}}\right)}{2} - \frac{\sqrt{K_d N_A \left(V_{\text{Box}} - V_{\text{ex}}\right)\left(K_d N_A \left(V_{\text{Box}} - V_{\text{ex}}\right) + 4\right)}}{2} + 1. \quad (4.74)$$

Now we can determine K_d and V_{ex} from a global fit of eq 4.74 to the observed values of p_{AB} at different box volumes. An alternative is to use the concentrations of the reactants and products, and the rates

$$K_d = \frac{(1-p_{AB})^2}{p_{AB}} \frac{1}{N_A V} \quad (4.75)$$

$$= \frac{k_{\text{off}}}{k_{\text{on}}}. \quad (4.76)$$

It should be noted that due to the definition of k_{on} and k_{off} calculating K_d from the fraction bound and rates yield exactly the same results. Therefore eq 4.75 can be used to test if the rates have been calculated correctly.

To check the if the K_d values are persevered by our BDMC algorithms we also set up a normal Monte Carlo simulation for each simulation box size with all three acceptance functions. The rotation angle was chosen uniformly between $-\pi$ and π, the translation was chosen uniformly between $-3\,\text{Å}$ and $3\,\text{Å}$, 20000 configurations have been recorded with every fourth simulation frame saved to file. For the simulation with a box edge-length of $300\,\text{Å}$ 100000 configurations have been recorded. For this comparison, every structure with an energy below $-2\,\text{kT}$ was counted as bound [96]. To estimate an error of any equilibrium observable value the same bootstrapping procedure as for the rate calculation was used.

The trajectories have been analyzed with scripts building on the following Python libraries: MDAnalysis,[65, 135] NumPy,[216] SciPy,[88] datreant,[44] pandas,[131] and matplotlib[84].

4.3.4 Binding of the CUE-ubiquitin complex

We simulated the binding kinetics of the CUE-ubiquitin complex. For the simulations we took the PDB structure 1OTR [93] and placed the two proteins into a cubic box, separated by a distance of 75 Å. We used 6 different box sizes with edge-lengths 125, 150, 175, 200, 225 and 300 Å. The diffusion tensor, used in the simulation, was calculated with HYDROPRO [147] for each protein using the 'shell' model, with the recommended bead radii of 4.8 Å, using a temperature of 300K and a viscosity of $1\,\text{mPa}\,\text{s}$. For the translation diffusion coefficient of CUE in the reference frame of the crystal struture we obtain

$$D_T = \begin{bmatrix} 5.9 \times 10^{-2} & -1.2 \times 10^{-2} & 9.5 \times 10^{-5} \\ -1.2 \times 10^{-2} & 4.0 \times 10^{-2} & -8.4 \times 10^{-5} \\ 9.4 \times 10^{-5} & -8.4 \times 10^{-5} & 3.5 \times 10^{-2} \end{bmatrix} \text{nm}^2\text{ns}^{-1}. \quad (4.77)$$

For the rotational diffusion coefficient of CUE in the reference frame of the crystal struture we obtain

$$D_R = \begin{bmatrix} 3.9 \times 10^{-2} & 4.6 \times 10^{-4} & 6.0 \times 10^{-3} \\ 4.6 \times 10^{-4} & 3.1 \times 10^{-2} & 2.7 \times 10^{-4} \\ 6.0 \times 10^{-3} & 2.7 \times 10^{-4} & 3.4 \times 10^{-2} \end{bmatrix} \text{ns}^{-1} \tag{4.78}$$

For the translation diffusion coefficient of ubiquitin in the reference frame of the crystal struture we obtain

$$D_T = \begin{bmatrix} 1.5 \times 10^{-1} & -7.8 \times 10^{-3} & 8.3 \times 10^{-4} \\ -7.8 \times 10^{-3} & 1.4 \times 10^{-1} & -4.2 \times 10^{-5} \\ 8.3 \times 10^{-4} & -3.6 \times 10^{-5} & 1.4 \times 10^{-1} \end{bmatrix} \text{nm}^2\text{ns}^{-1} \tag{4.79}$$

For the rotational diffusion coefficient of ubiquitin in the reference frame of the crystal struture we obtain

$$D_R = \begin{bmatrix} 1.4 \times 10^{-1} & 7.4 \times 10^{-4} & 5.2 \times 10^{-3} \\ 7.3 \times 10^{-4} & 1.3 \times 10^{-1} & 3.8 \times 10^{-4} \\ 5.2 \times 10^{-3} & 3.7 \times 10^{-4} & 1.3 \times 10^{-1} \end{bmatrix} \text{ns}^{-1} \tag{4.80}$$

Other simulation parameters and the analysis are the same as for the UIM-1-ubiquitin complex. We use the total energy of the system and the minimal distance between any two C_α atoms of CUE and ubiquitin to define transition-, unbound- and bound-states, identical to the UIM-1-ubiquitin complex, see Figure 4.5.

4.4 Results

4.4.1 Harmonic Oscillator

In the case of a harmonic oscillator the BDMC algorithms and the BAOAB integrator are able to correctly reproduce the MSD, using the Euler-Maruyama integrator the MSD grows linearly with the time-step, see Figure 4.6A. For time-steps $dt \leq 0.01$ all algorithm reproduce the correct MSD. With exception of the Metropolis acceptance function all algorithms predict the correct autocorrelation time for $dt < 0.005$. The BDMC algorithms using the Metropolis acceptance function converge to the correct autocorrelation time for time-steps $dt < 0.0001$. At larger time-steps all algorithms show systematic deviations from the expected value, Figure 4.6B. The Euler-Maruyama and BAOAB methods have the smallest drift. The BDMC algorithm with the smallest drift uses the optimal acceptance function and a uniform trial function. The version with the largest error uses the Metropolis acceptance function and the Gaussian trial probability. Re-scaling the autocorrelation time with eq 4.44 yields a large improvement of the autocorrelation times, Figure 4.6C. The BDMC with uniform trial probability now yields results as good as the BAOAB integrator for large time-steps as well.

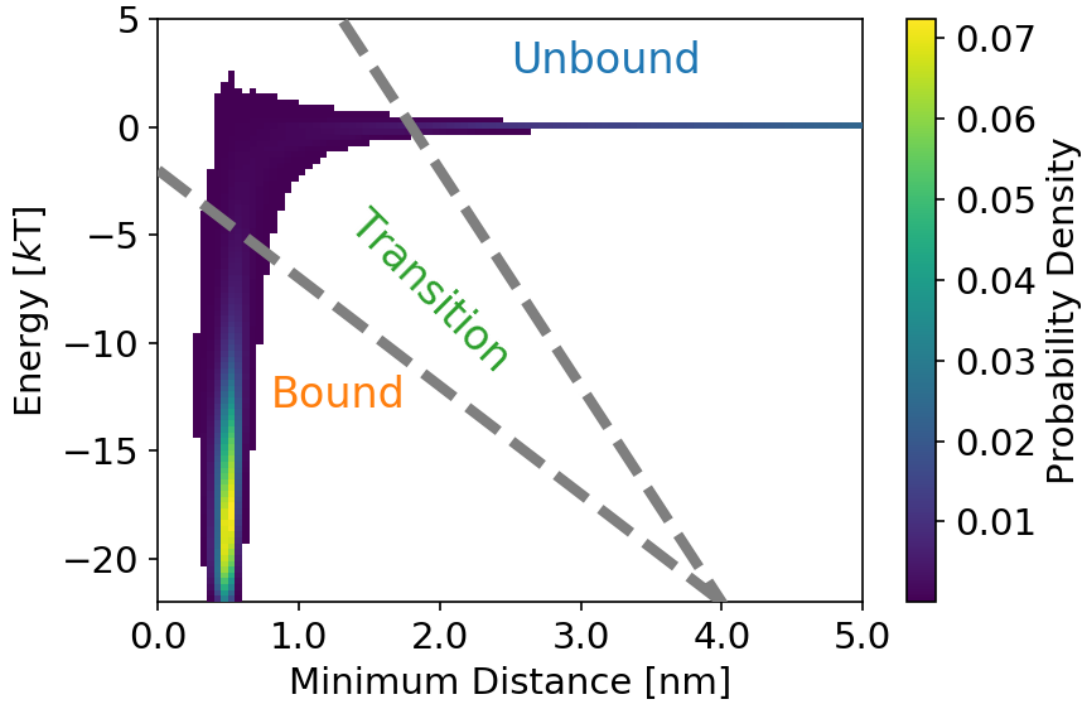

Figure 4.5: Probability density of the CUE-ubiquitin complex for the minimal distance between two C_α atoms and the total energy.

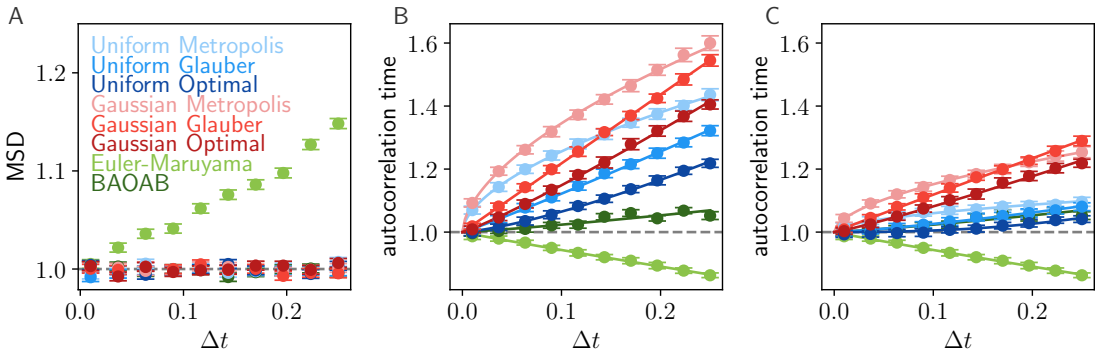

Figure 4.6: MSD (left), autocorrelation time (middle) and autocorrelation time scaled by ratio of acceptance rate to ideal acceptance rate, see eq 4.44, (right) of the harmonic oscillator calculated with BDMC algorithm using different acceptance function and trial probabilities, Euler-Maruyama, and BAOAB integrator. The BDMC algorithms using a Gaussian trial probability and uniform trial probability are shown in red and blue shades, respectively. For the BDMC algorithms three difference acceptance function, Metropolis, Glauber, and the optimal have been used. Error bars denote standard error of the mean. Lines for the autocorrelation time show a fit to eq 4.55. The autocorrelation time was scaled using eq 4.44.

From the simulation results of the autocorrelation functions we fit eq 4.55 to estimate the accuracy of the BDMC algorithms. γ indicates the order of the error $\tau - \tau_0 \approx \mathcal{O}(dt^\gamma)$, see Table 4.3. The fit has been performed using all simulations from $dt = 0.0001$ to

$dt = 0.25$. The Euler-Maruyama and BAOAB integrators have a value of $\gamma \approx 1$. Using the Metropolis acceptance function we get $\gamma \approx 1/2$ independent of the trial probability before and after correcting the autocorrelation times. For the optimal acceptance function with the uniform trial function γ changes from 1 to 2, after correcting the autocorrelation times. The pre-factor α is significantly smaller though leading to an overall smaller error. Other BDMC algorithms have $\gamma \approx 1$ before and after correcting the autocorrelation times. The pre-factor α is decreased for all BDMC algorithms after correction of the autocorrelation times.

$T(x\|y)$	$A(x\|y)$	α	γ	α'	γ'
Uniform	Metropolis	0.94(1)	0.56(1)	0.20(1)	0.53(1)
	Glauber	1.41(5)	1.06(5)	0.39(5)	1.14(5)
	Optimal	1.40(8)	1.32(8)	0.83(58)	2.12(58)
Gaussian	Metropolis	1.32(1)	0.58(1)	0.55(1)	0.56(1)
	Glauber	2.18(5)	1.00(5)	1.12(4)	0.97(4)
	Optimal	1.95(7)	1.11(7)	1.05(6)	1.11(6)
Euler-Maruyama		0.50(3)	0.95(3)		
BAOAB		0.27(7)	1.01(7)		

Table 4.3: Parameter of the fit to eq 4.55 for the autocorrelation times. All available data was used for the fit. Values in parentheses denote the standard deviation of the fit parameter. α' and γ' denote the fit values before scaling the autocorrelation time using eq 4.44.

4.4.2 Position Dependent Diffusion

For the position dependent diffusion of the harmonic oscillator the MSD in the back mapped system, eq 4.60, is correctly reproduced with any BDMC algorithm, Figure 4.7. The xBAOAB integrator now has noticeable deviations from the expected MSD for $dt > 0.1$. The Euler-Maruyama integrator performs the worst again to reproduce the MSD. It can only predict the correct MSD for $dt < 0.01$. The autocorrelation-time can be reproduced by all algorithms for sufficiently small dt. The BDMC algorithms experience a larger deviation for large time-steps than the xBAOAB and Euler-Maruyama integrators. For the Euler-Maruyama integrator calculating a autocorrelation time was impossible for time-steps larger than $dt = 0.17$ due to numerical instabilities. For the BDMC algorithms the autocorrelation times are systematically larger than the expected value for a time-step of $dt = 0.01$ and larger. With a time-step of $dt = 0.001$ the BDMC algorithms are also able to reproduce the expected autocorrelation time. Re-scaling the autocorrelation time with eq 4.44 yields again a large improvement.

Estimating the overall error for the position dependent diffusion from a fit of the autocorrelation times to eq 4.55 shows that the γ values are very similar before and after correcting the autocorrelation times for the BDMC algorithms, Table 4.4. Correcting the autocorrelation times results in a reduction of the pre-factor α for all BDMC algorithms.

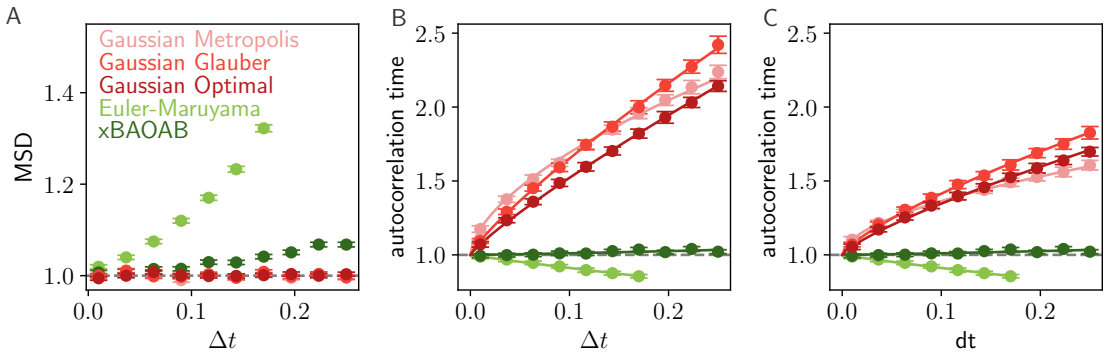

Figure 4.7: MSD (left) and autocorrelation time (middle) and autocorrelation time scaled by ratio of acceptance rate to ideal acceptance rate, see eq 4.44, (right) of the position dependent diffusion model eq 4.60 after conversion to x for different acceptance function using the BDMC algorithm, and Euler-Maruyama, and xBAOAB integrators. The position dependent BDMC algorithms using a Gaussian trial probability are shown in red shades. Error bars denote the standard error of the mean. Lines for the autocorrelation time show a fit to eq 4.55.

The Euler-Maruyama integrator has again a value of $\gamma \approx 1$. For the xBAOAB integrator we obtain $\gamma \approx 1/2$.

$T(x\|y)$	$A(x\|y)$	α	γ	α'	γ'
Gaussian	Metropolis	1.28(1)	0.54(1)	2.73(4)	0.60(4)
	Glauber	2.36(4)	0.76(4)	4.45(5)	0.83(5)
	Optimal	2.10(6)	0.78(6)	3.76(6)	0.86(6)
Euler-Maruyama		0.73(4)	0.92(4)		
xBAOAB		0.07(2)	0.63(2)		

Table 4.4: Parameter of the fit to eq 4.55 for the autocorrelation times. All available data was used for the fit. Values in parentheses denote the standard deviation of the fit parameter. α' and γ' denote the fit values before scaling the autocorrelation time using eq 4.44.

4.4.3 Binding of the Vps27 UIM-1-ubiquitin complex

We simulated the UIM-1-ubiquitin complex for different time-steps, box sizes, and acceptance functions, and calculated the rates and dissociation constants. The lifetime distribution of the bound and unbound state in the UIM-1-ubiquitin complex follows a single exponential decay after applying the TBA filter, Figure 4.8. The transition times are much shorter than the lifetimes of the unbound and bound states. It is noticeable that before the TBA filter is applied the bound and unbound state experience an unusually large amount of short lived states. These short lived states are an indication for frequent recrossing. The TBA filter removes the nonphysical amount of short lived states and the resulting lifetime distributions look similar to a waiting time distributions $\sim \lambda e^{-\lambda t}$, with λ the inverse of the half-life time. For the largest time-step, unusually long lived states start

to appear. These long lived states are an indicator that our assumption $\Delta U \approx 0$ is not true any longer. Therefore we need to use an effective time-step,eq 4.44, when analyzing the rates.

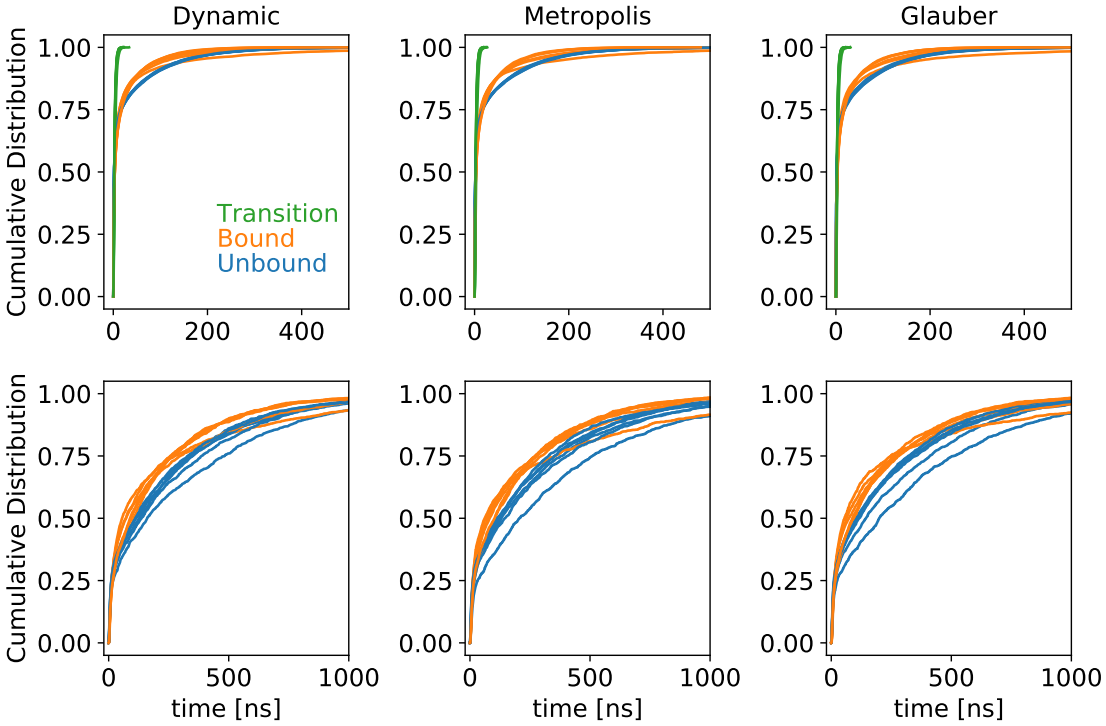

Figure 4.8: The cumulative distribution function of the lifetimes of the bound, and unbound states before (top) and after (bottom) applying the TBA filter for different time-steps and acceptance functions (rows). The distribution of transition times is shown in green. The box size was fixed at 150 Å.

The on- and off-rates are remarkably independent of the integration time-step, see Figure 4.9. For the on-rate more than three orders of magnitude change in the time-step, from 50 fs to 50 000 fs, results in a difference of on-rates of 12 % for the Metropolis and Glauber acceptance function and 50 % for the Dynamic acceptance function. For the off-rate, the increase is 50 % independent of the acceptance function. For small time-steps, the calculated rates using different acceptance functions agree within the error. For the on-rate differences between the acceptance functions exist for time-steps $dt > 10$ ps. The on- and off-rates are stable with respect to the simulation box size, Figure 4.10. The pseudo on-rate experiences a decrease with increasing box-size as expected because it takes longer for the two diffusing reactants to come together. When we disable electrostatic interactions the on-rate decreases by one order of magnitude and the off-rate increases by one order of magnitude, Figure 4.11. Electrostatic interactions therefore play an important role in the formation of the bound state. The fast on-rates we observed before are therefore a result of a electrostatically guided binding process.

To determine K_d we fitted eq 4.74 to the fraction bound averaged over acceptance function and time-steps for different box volumes, Figure 4.12. The fit yields $K_d = 536(21)\,\mu$M

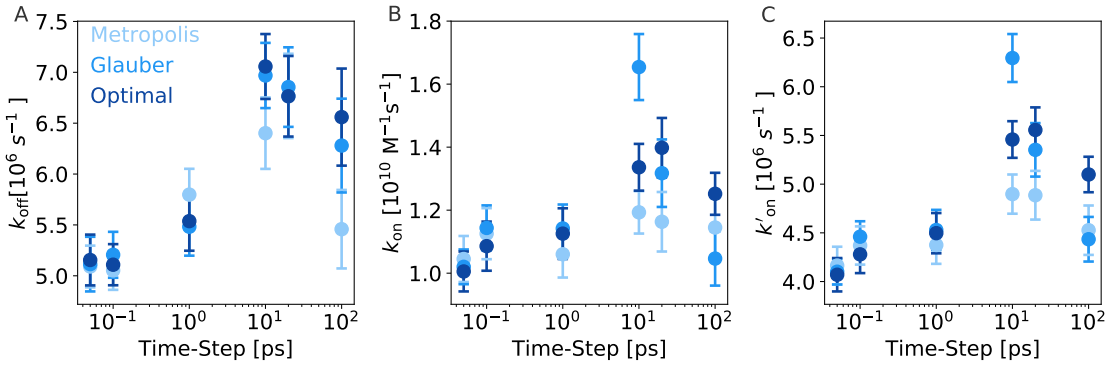

Figure 4.9: UIM-1-ubiquitin off-rates (left), on-rates (middle), and pseudo on-rates (right) for different time-steps and acceptance functions (Metropolis: pentagon, Glauber: square, Optimal: diamond). The box size was fixed at 150 Å. Error bars denote the standard error of the mean.

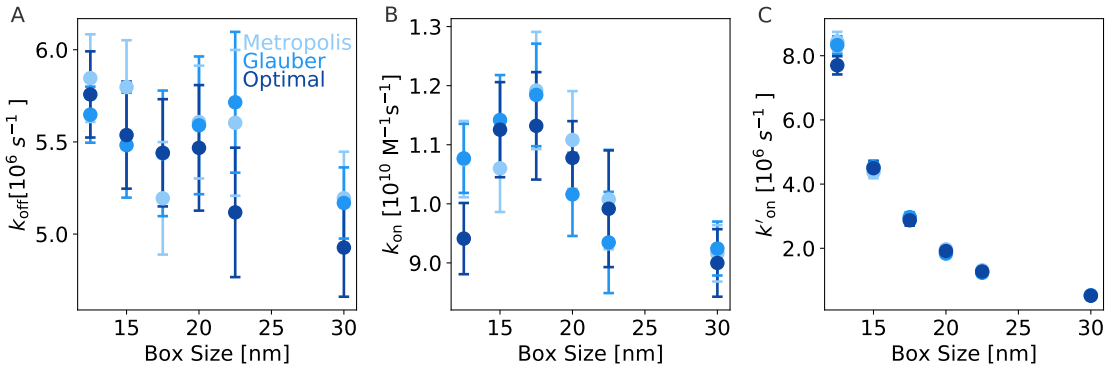

Figure 4.10: UIM-1-ubiquitin off-rates (left), on-rates (middle), and pseudo on-rates (right) for different box sizes and acceptance functions. The time-step was fixed at $dt = 50\,\text{fs}$. Error bars denote the standard error of the mean.

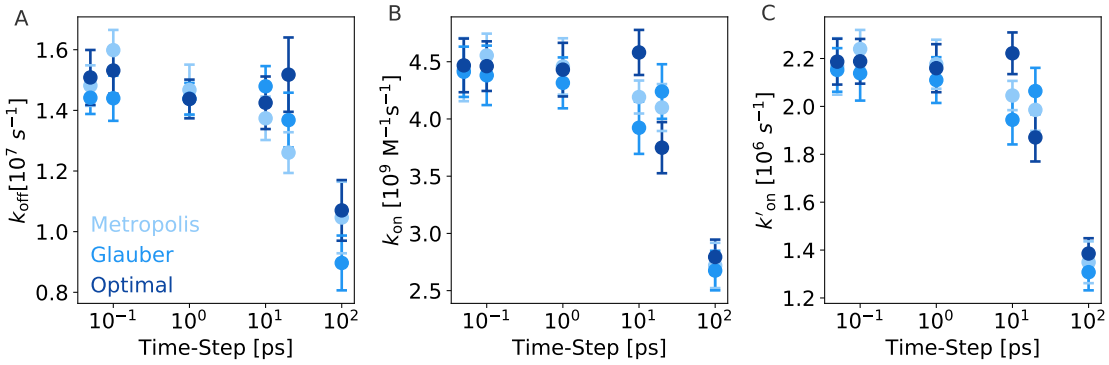

Figure 4.11: Results for binding simulations without electrostatic interactions. Off-rates (left), on-rates (middle), and pseudo on-rates (right) for different time-steps and acceptance functions. The box size was fixed at 150 Å. Error bars denote the standard error of the mean.

and $V_{ex} = 1.07(7) \times 10^{-21}$ Liter. This value agrees well with almost all fits of eq 4.74 to individual simulations using different time-steps and acceptance function, Figure 4.12 right.

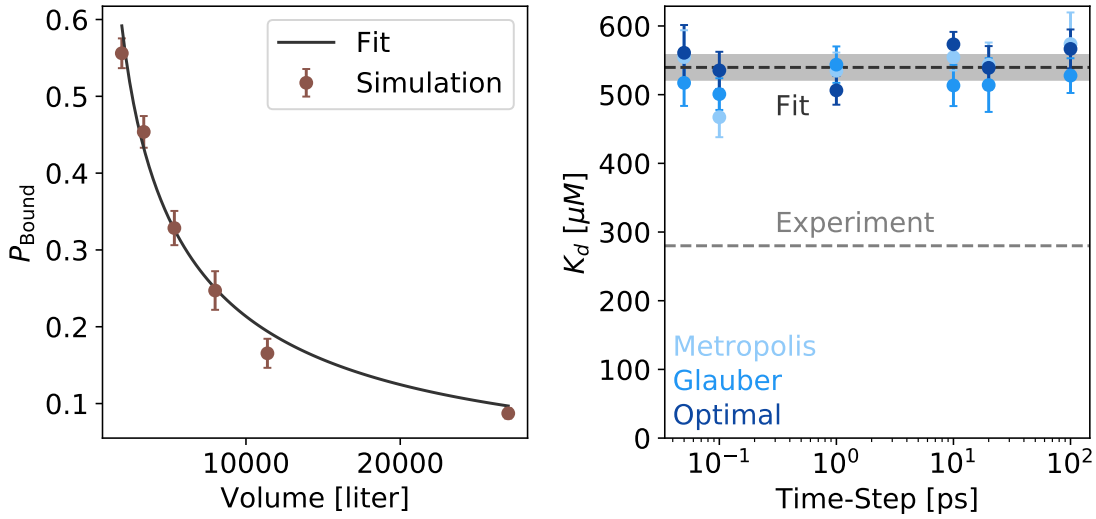

Figure 4.12: (Left) Fraction bound averaged over acceptance function and time-steps for different simulation box volumes fitted to eq 4.74. (Right) K_d values from fits of simulations for time-step sizes for each acceptance function. The experimental value [198] is shown as dashed gray line and global fit of K_d black dotted line with the standard deviation of the fit as shaded area.

Our value is of by a factor ~2 from the experimental value of 280 μM [198]. The K_d value can also be calculated as an equilibrium probability from a single simulation using eq 4.75. In Figure 4.13 the K_d values are shown calculated directly from the probabilities and rates. For all used acceptance functions the K_d values agree well with the global fit. Therefore the BDMC algorithm also preserves the equilibrium for the UIM-1-ubiquitin complex.

We used three properties to check if the BDMC algorithms preserve equilibrium properties: the energy fluctuations, K_d, and the probability p_{AB} to be bound, Figure 4.14. For all observables and acceptance functions the observables agree within the error margin. For this analysis the criteria for a configuration to be bound was solely based on the total energy of the system.

4.4.4 Binding of the CUE-ubiquitin complex

We simulated the CUE-ubiquitin complex for different time-steps, box sizes, and acceptance functions, and calculated the rates and dissociation constants. The on- and off-rates of the CUE-ubiquitin complex are again remarkably constant with regard to changes in integration time-step and box size, see Figs. 4.15 and 4.16. Over all tested time-steps the calculated rates are on the same order of magnitude. Increasing time-step by three orders of magnitude results increase by a factor 2-2.5 for all rates. The effect of the box size of the rates is even less except for the pseudo first-order on-rate k'_{on} that is expected to be box size dependent. Similar to the UIM-1-ubiquitin complex the choice of acceptance function has a minimal influence on the rates.

We also calculated the equilibrium dissociation constant, see Figure 4.17. The fit of

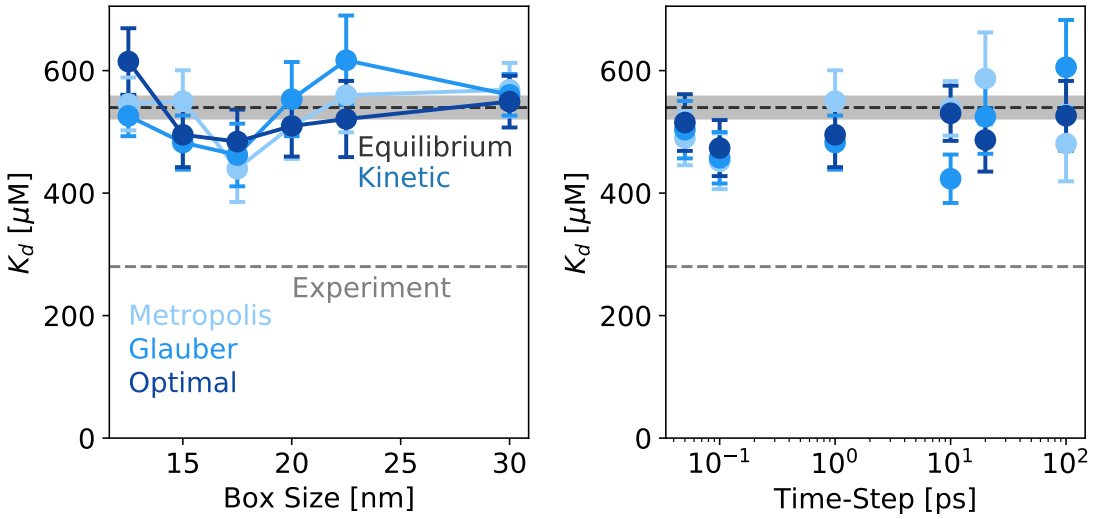

Figure 4.13: UIM-1-ubiquitin dissociation constant K_d calculated from the rates (left) and the equilibrium probabilities (right) for different time-steps and acceptance functions. The box size was fixed at 150 Å. Error bars denote the standard error of the mean. Experimental value of $K_d = 280\,\mu$M [198] and the value of the global fit are also shown. The shaded area indicates the standard deviation of the global fit.

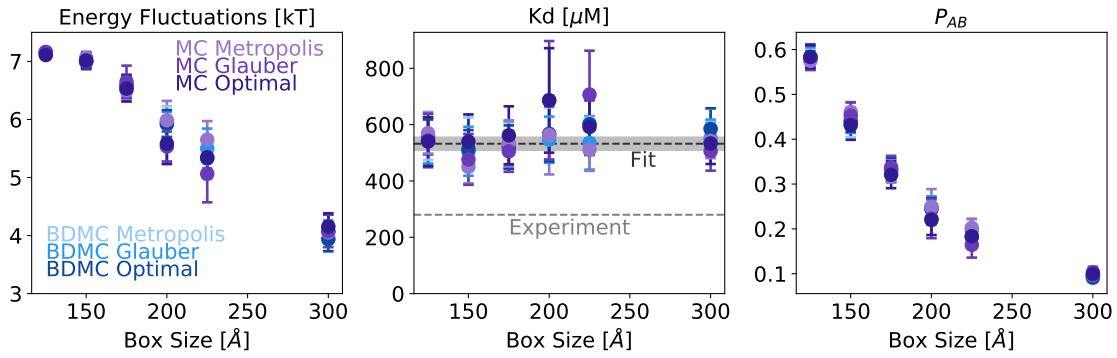

Figure 4.14: Energy fluctuations, dissociation constant K_d, and probability p_{AB} to be bound for the Brownian dynamics Monte Carlo (blue shades) and Monte Carlo (purple shades) for different acceptance functions and simulation box sizes. Errorbars show the standard deviation from bootstrap samples. Experimental value of $K_d = 280\,\mu$M [198] is shown with the global fit estimate and the shaded area indicates the standard deviation of the fit.

eq 4.74 yields $K_d = 120(11)\,\mu$M and an excluded volume off $V_{\text{ex}} = 1.5(1)\times10^{-21}$ Liter. This is close to the experimental value of $K_d = 160\,\mu$M [93]. The K_d values are independent of the time-step as expected, Figure 4.17 (right).

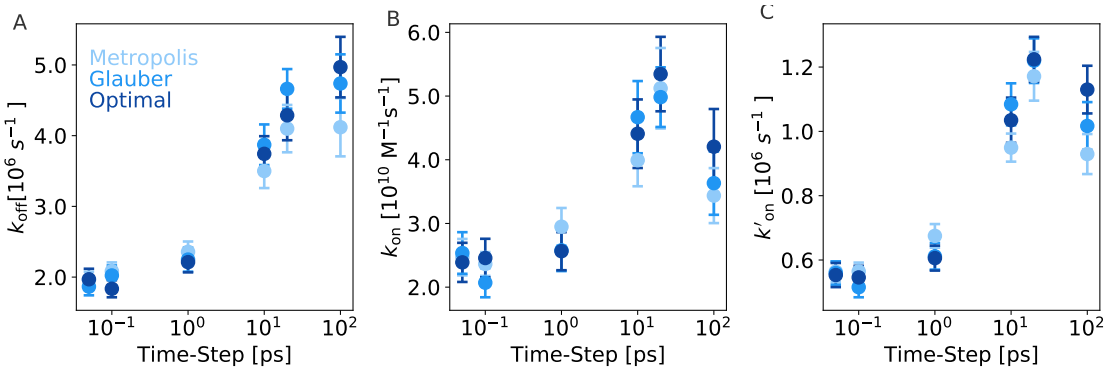

Figure 4.15: CUE-ubiquitin off-rates (left), on-rates (middle), and pseudo on-rates (right) for different time-steps and acceptance functions. The box edge length was 150 Å. Error bars denote the standard error of the mean.

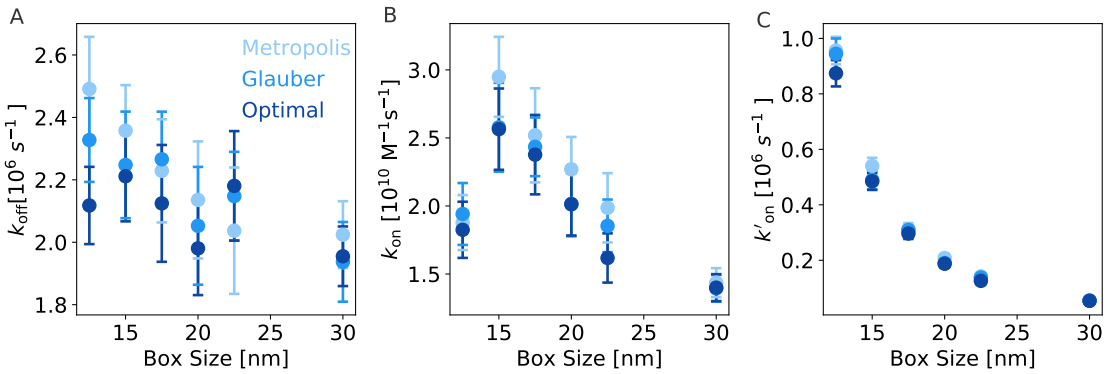

Figure 4.16: CUE-uiquitin off-rates (left), on-rates (middle), and pseudo on-rates (right) for different box sizes and acceptance functions. The time-step was fixed at $dt = 50$ fs. Error bars denote the standard error of the mean.

4.5 Conclusion

We have generalized the existing BDMC algorithm for anisotropic particles, different acceptance function, and trial-move probabilities. Our tests using a harmonic oscillator show that dynamic observables agree well with common Langevin integrators. Compared to the Langevin integrators the BDMC algorithms do not suffer an error in the equilibrium distribution for large time-steps. We developed a new acceptance function that is optimized to reduce the error in of higher moments in the probability distribution of a Brownian particle under constant force. Using our new acceptance function we are able to more accurately reproduce dynamical properties of the harmonic oscillator than with the Metropolis and Glauber acceptance function. We showed how the BDMC method can also be extended for position-dependent diffusion. For the position-dependent diffusion the error in dynamic observables is larger than for common Langevin integrators, however the BDMC algorithms can also be used if gradients of the energy or the position dependent diffusion coefficients are not known. We have also implemented the BDMC algorithm for anisotropic particles for a CG protein model. Our results show that the dissociation con-

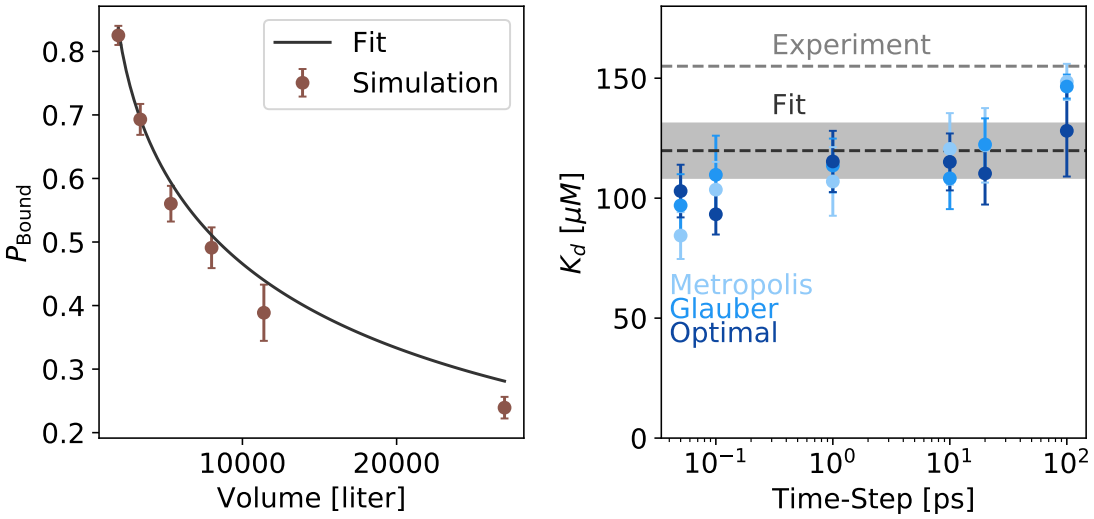

Figure 4.17: CUE-ubiquitin dissociation constant K_d calculated from the fraction bound (left) and binding rates (right). Error bars denote the standard error of the mean. Experimental value of $K_d = 160\,\mu$M [93] and the value of the global fit to eq 4.74 are also shown. The shaded area indicates the standard deviation of the global fit. The box edge length was 150 Å.

stant, an equilibrium property, calculated from on and off-rates agrees well with estimates calculated from the equilibrium density. It should be noted that with the standard KH model our on-rates are high. The rates we calculated are likely higher than they would be in an all-atom forcefield simulations due to the smoothened landscape. The rates are also reduced when disabling electrostatic interactions indicating that the binding process is guided by them. The estimates of on- and off-rates are remarkably stable with regard to the chosen time-step. A disadvantage of BDMC algorithms is that only local moves are possible and the widths of different move types have to be properly balanced to produce proper BD. It should be noted that results with the Metropolis acceptance function show that the dynamics of isotropic particles can be calculated with existing Monte Carlo programs if the parameters of the trial move are chosen correspondingly.

Chapter 5

Complexes++: A General Monte Carlo Engine

5.1 Introduction

Complexes is a hierarchical CG protein model. It has been previously implemented with a Monte Carlo engine to generate structures of protein complexes [96] and is commonly refereed to as the KH model in the literature. Since the original publication, the forcefield has been used in a large variety of applications, including studies of the binding kinetics of the HIV-1 capsid proteins [183, 241], the kinetic behavior of proteins in crowded environments [42, 97, 98, 167], the folding of knotted protein [2, 36, 37, 188, 189, 191], enhancing the structural resolution of experiments [11, 30, 100, 173, 174], to study protein-protein interactions [124, 145, 162], protein design [233], docking [53], and multi-enzyme complexes [52, 77, 172, 175]. The model has also been extended to simulate intrinsically disordered proteins and ligand-ligand phase separation of disordered proteins [42].

Originally the Complexes forcefield was implemented in FORTRAN77. In the meantime parts of the forcefield have been ported to run on GPU [213] and implemented in LAMMPS [77, 158]. Here we present a new implementation of the simulation engine, called "Complexes++", written in C++14, and a helper application, called "pycomplexes", written in Python, to set up and visualize simulations. It implements the original hierarchical model and adds several new features, including additional functional forms interaction potentials, easier simulation setup, and more thermodynamic ensembles among others.

5.2 Gaussian Chain Flexible Domain

A compelling feature of the original Complexes implementation [96] was the explicit treatment of flexible chains in protein complexes. These chains served as an anchor to link rigid domains together. In the remainder of this thesis we call a flexible protein chain a linker. In the original paper [96] a peptide chain model using bond, angle and dihedral potentials similar to MD forcefields [78] has been used. The advantage of such a model is that amino acids are modeled explicitly and can interact with the rigid domains. A drawback of using this model with a Monte Carlo scheme is that movements of the whole chain are small. As a result the chain diffuses slowly through configuration space and the overall diffusion of attached rigid domains is limited by the linker instead of the translation and rotation

step-size chosen for the rigid domains. While such an explicit model can work for smaller protein complexes [173] it makes simulations of larger complexes [100] difficult.

A linker model that has a limited influence on the diffusion of the rigid domains would be better suited to efficiently sample configuration space. It has been shown that for a linker only the length is important and not the exact dynamics [175]. We therefore assume that the linker only has to hold two rigid domains close and has no other functional purpose. We replace the explicit peptide chain with a potential of mean force (PMF) that only depends on the distance between the two rigid domains and the number of amino acids in the linker, i.e., the linker length. This PMF potential acts as a restraint potential ensuring that the distance distribution between two rigid domains is physically correct. Because no explicit beads are involved the diffusion of the rigid domains is not influenced by the linker. We use a Gaussian chain polymer model to develop a suitable PMF. Gaussian chains have been previously used to model flexible linkers in FRET experiments [143].

In a Gaussian chain the beads are point particles connected by harmonic springs. The average distance b between two beads determines the spring constant. In one dimension the probability for two beads i and j to have a distance distance r_{ij} is

$$P(r_{ij}) = \left(\frac{3}{2\pi\langle R_{ij}^2\rangle}\right)^{3/2} \exp\left(-\frac{3r_{ij}^2}{2\langle R_{ij}^2\rangle}\right), r_{ij} > 0, \tag{5.1}$$

For a three dimensional chain the distances $R_{ij} = |\vec{r}_j - \vec{r}_i|$ between two beads, irrespective of the direction, is distributed according to [234]

$$P(R_{ij}) = 4\pi R_{ij}^2 \left(\frac{3}{2\pi\langle R_{ij}^2\rangle}\right)^{3/2} \exp\left(-\frac{3R_{ij}^2}{2\langle R_{ij}^2\rangle}\right), R_{ij} > 0, \tag{5.2}$$

where $\langle R_{ij}^2\rangle = b^2(j-i)$ is the mean squared distances between beads i and j. The factor $4\pi R_{ij}^2 dR$ is the volume element of a shell with width dR and radius R_{ij}. The average end-to-end distance for a Gaussian chain with N beads is

$$\sqrt{\langle R_{1N}^2\rangle} = \sqrt{N}b. \tag{5.3}$$

To construct a PMF for the Gaussian chain model we look at the exponential term in eq 5.1. It is similar to the Boltzmann distribution for a harmonic oscillator with spring constant $3/(b^2(j-i))$

$$V(\vec{r}_i, \vec{r}_j) = \frac{3}{2b^2}\frac{1}{(j-i)}|\vec{r}_i - \vec{r}_j|^2 \approx -\ln(P(r_{i-j})). \tag{5.4}$$

From this the PMF between the ends of a Gaussian chain of length N follows as

$$PMF(\vec{r}_1, \vec{r}_N) = \frac{3}{2b^2}\frac{1}{N-1}|\vec{r}_1 - \vec{r}_N|^2. \tag{5.5}$$

To compare the distribution of end-to-end distances obtained from eq 5.5 we run a Monte Carlo simulation for a linker of length $N = 200$ and a bond length $b = 3.81\,\text{Å}$ [19], see

Figure 5.1. The distribution of end-to-end distances created using the PMF agrees well with the expected distribution of the Gaussian polymer model. The agreement of the PMF with the expected distribution is independent of the length N. This PMF has been previously used to model proteins and ribonucleic acid (RNA) [85].

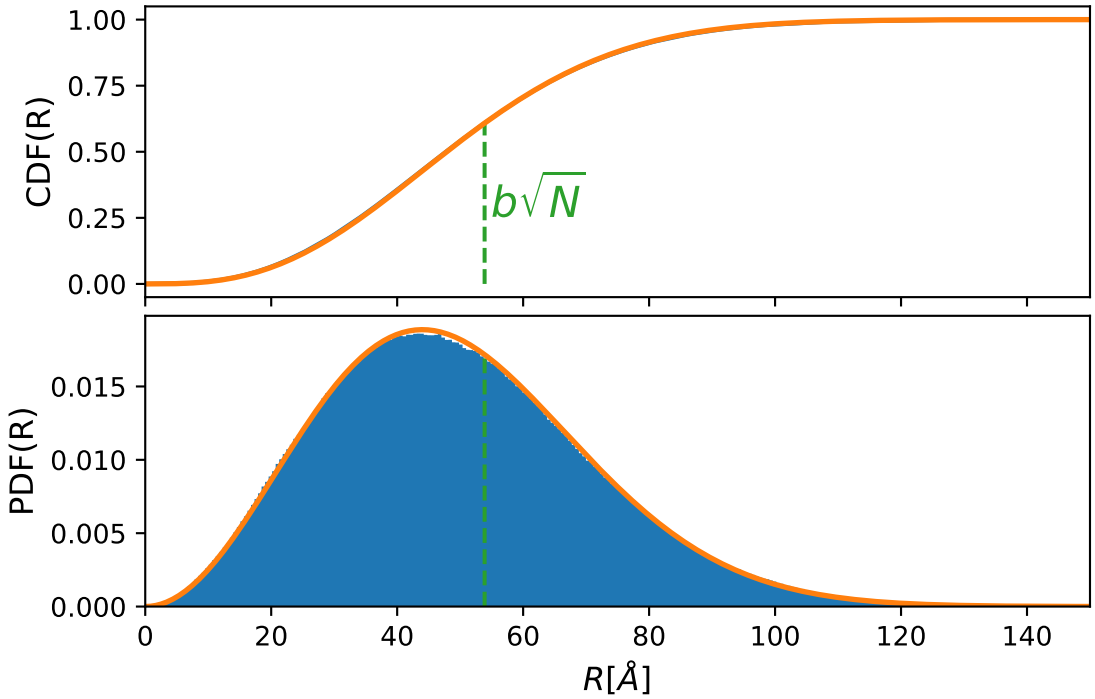

Figure 5.1: Cumulative distribution function (top) and probability density function (bottom) of the end-to-end distance R for a Gaussian chain of length $N = 200$ and bond length $b = 3.81$ Å. The distributions for the ideal Gaussian chain are shown in orange. Results from an ensemble of 1 million distances obtained from a Monte Carlo simulation using eq 5.5 are shown in blue. The value of the mean end-to-end distance, eq 5.3, is marked in green.

The final PMF that we use is between two beads of the two connected rigid domains. These two beads have to be added to the length N of the linker. Therefore the final PMF for a linker of length N is

$$PMF(\vec{r}_0, \vec{r}_{N+1}) = \frac{3}{2b^2}\frac{1}{N+1}(\vec{r}_0 - \vec{r}_{N+1})^2, \tag{5.6}$$

with $\vec{r}_0$ and $\vec{r}_{N+1}$ being the two beads of the rigid domains that are connected by the linker.

Generating Explicit Beads for Linker Model

While the PMF eq 5.5 is good for fast exploration of phase space it does not provide explicit positions for linker beads. However, some applications, like the comparison of simulations to small angle x-ray scattering (SAXS) measurements [100, 173] for example, require to

have explicit beads for the linker domains. We now describe an iterative algorithm to generate positions for the linker beads given fixed positions for the first and last bead of the linker. Given the positions of bead N and 1 a single bead $N-1$ can be generated

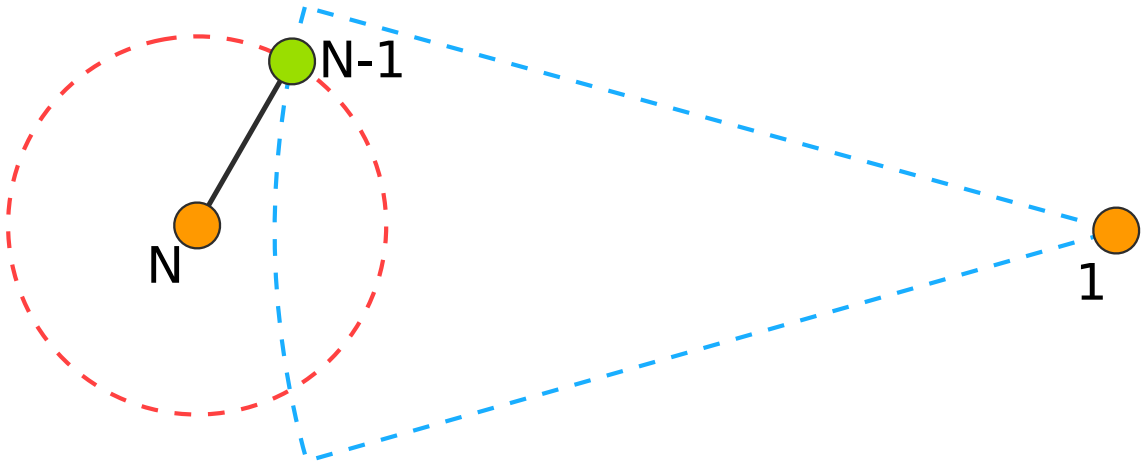

Figure 5.2: Example of distances drawn for a new bead $N-1$ (green) between to fixed endpoints (orange) of a Gaussian linker of length N in two dimensions. The red circle is the distance between the beads N and $N-1$. This distance is randomly chosen from eq 5.2. The bead $N-1$ can be placed anywhere on this circle. The distance between bead 1 and $N-1$ now has to be chosen so that circle (indicated as blue cone) drawn around bead 1 intersects with the first circle (red). In two dimensions this restricts the positions of the new bead to the two intersection points. In three dimensions it would be restricted to a circle.

with the following algorithm:

1. Randomly choose a distance d_{start} between bead N and $N-1$ distributed according to eq 5.2. This distance defines a sphere around bead N on which bead $N-1$ will be placed. (Figure 5.2, red circle)

2. Choose a distance d_{end} between bead 1 and $N-1$ so that the sphere around bead 1 intersects with the sphere calculated in step 1. (Figure 5.2, blue cone)

3. Choose a random point on the intersection of the red and blue sphere to place the bead $N-1$ (Figure 5.2, green bead). In three dimensions the intersection is a circle so a random angle θ has to be drawn.

To grow the next bead, with index $N-2$, simply repeat the algorithm with bead $N-1$ as new starting point to choose the random distance $R_{N-1,N-2}$. Repeat this algorithm until all beads are generated. This algorithm gives the three distances between the beads and orientation that uniquely determine where a new bead should be placed. To calculate the actual coordinates in the coordinate system of the simulation we use the following algorithm.

1. Determine the axis $\vec{z'}$ along the vector $\vec{r}_N - \vec{r}_1$.

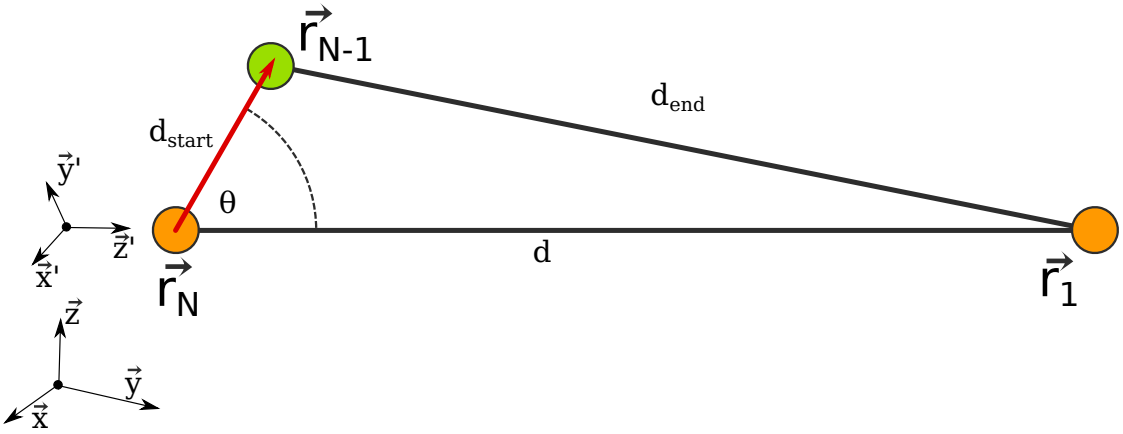

Figure 5.3: Schematic for calculating the position of bead $\vec{r}_{N-1}$ given the positions of $\vec{r}_N$, $\vec{r}_1$, the distances d, d_{start}, d_{end} and an angle ϕ (not shown). The coordinate system $(\vec{x}, \vec{y}, \vec{z})$ is our reference coordinate system. The coordinate system $(\vec{x'}, \vec{y'}, \vec{z'})$ is used to calculate $\vec{r}_{N-1}$.

2. Determine perpendicular axes $\vec{x'} = \left(\frac{-z'_1 - z'_2}{z'_0}, 1, 1\right)^T$ and normalize. Permutate in elements of $\vec{x'}$ if $z'_0 = 0$.
3. Determine $\vec{y'} = \vec{x'} \times \vec{z'}$, with $\times$ the cross product.
4. Determine angle ϕ using the law of cosines $\cos(\phi) = \frac{d^2_{\text{start}} + d^2 - d^2_{\text{end}}}{2 d_{\text{start}} d}$, with $d = |\vec{r}_n - \vec{r}_1|$ see Figure 5.3.
5. Calculate $\vec{r}_{N-1}$ in the coordinate system spanned by $(\vec{x'}, \vec{y'}, \vec{z'})$ from the spherical coordinates given by $(d_{\text{start}}, \theta, \phi)$, see Figure 5.3.
6. Convert $\vec{r}_{N-1}$ into reference coordinate system $(\vec{x}, \vec{y}, \vec{z})$.

Linker configurations generated using the above two algorithms will include overlaps between neighboring beads, see Figure 5.4. To avoid overlaps and generate more extended configurations it is sufficient to add overlap checks in step 1 and 3 in the first algorithm algorithm and redraw distances if two beads are to close to each other.

Relaxation of Gaussian Polymer Chain

The structures produced by the Gaussian chain growing algorithms are not physical. The distances between beads vary by a standard deviation of 1 Å with a mean of 5 Å in a single chain when the chain is grown with overlap checks. The fluctuation in typical protein structures is less than a hundredth of an Å with a mean distance of 3.81 Å [19]. For the generation of more physical bead coordinates, it is, therefore, necessary to relax the structures generated by the previous algorithm. For the relaxation, the linker energy eq 2.13 of the KH forcefield [96] can be used in combination with a Monte Carlo algorithm. As acceptance function we use Metropolis eq 2.54 with a temperature of 300 K. For trial moves the position of individual beads is changed. The start and end bead are

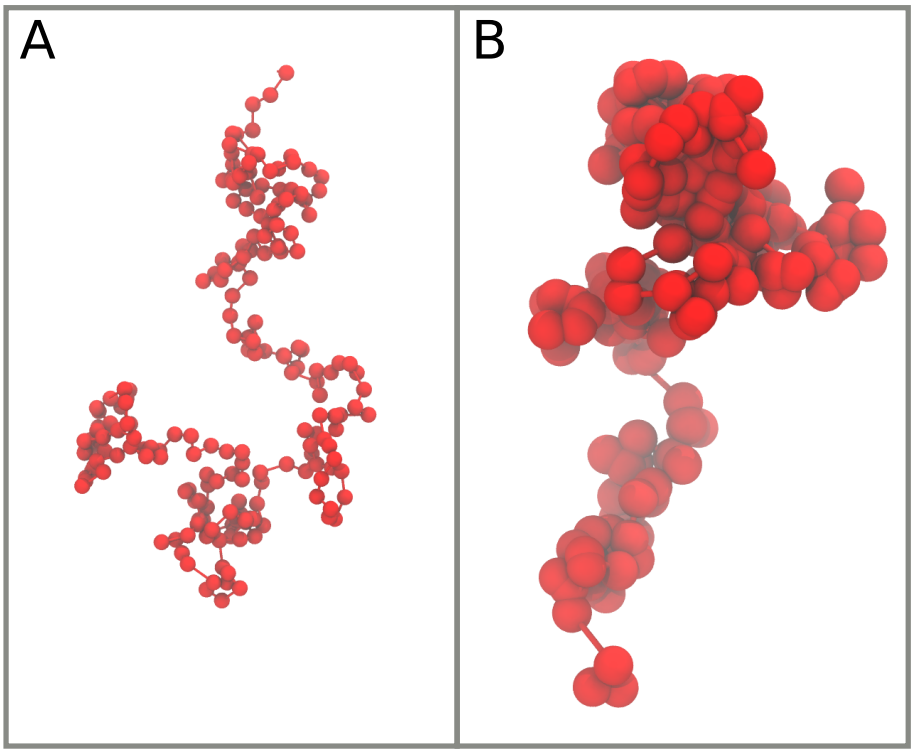

Figure 5.4: Gaussian chain with (left) and without (right) overlap check. Example configuration for a alanine chain of length 200 with a bond-length of b =3.81 Å. All beads are drawn with a diameter of 3.81 Å. Both configurations have the same start and end bead.

treated as fixed. For a linker of length N the probability to pick a bead is uniform between all $N-2$ beads that are allowed to move. A sweep consists of $N-2$ trial moves. Note we only do displacements of the beads and no angular or dihedral trial moves. Because the structure is only supposed to be relaxed it is not necessary to generate structures from an equilibrium distribution and therefore detailed balance does not need to be strictly preserved. Therefore, we adjust the step-width after each sweep to achieve a target acceptance ratio of 30 %. If after a sweep the acceptance ratio is larger than 30 % the step-width is increased by 10 % and decreased if the acceptance ratio is below 30 %.

Energy contributions for each term contributing to the total energy of the linker from a relaxation run for a chain of 200 beads is shown in Figure 5.5. In the beginning, the energy is dominated by the bond potential, this is due to the fact that average bond-length is larger than 3.81 Å. The bond lengths are fully relaxed after around 200 sweeps. The angle potentials start to relax around sweep 50 when the bond energy has already dropped by a factor of two. The angle potential is fully relaxed after around 1000 sweeps. The last potential to relax are the torsion angles. After about 500 sweeps the torsion angles start to see a more pronounced decrease after 3000 sweeps when the other two potentials haven been fully relaxed. The energy difference in the torsion potential from the beginning of the simulation to the final structure is significantly less than for the other two terms in the energy. There are two contributions to this behavior: The starting structures generated

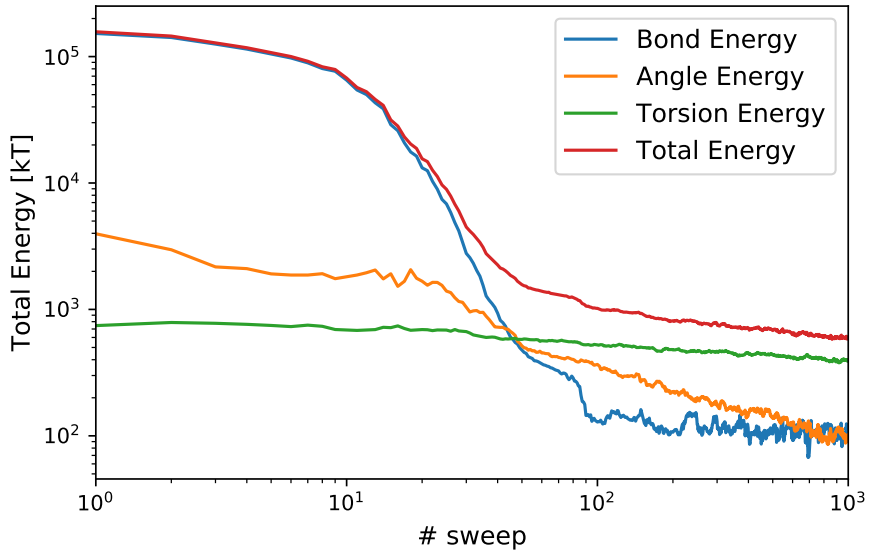

Figure 5.5: Energies of a nonoverlapping Gaussian chain during relaxation. The chain is 200 beads long and the initial structure was generated using the Gaussian chain model with no overlaps. The acceptance rate during the Monte Carlo simulation was set to target 30 %. The bond energy is shown in blue, the angle energy in orange, the torsion energy in green and the total energy in red.

by the Gaussian chain algorithm trapped in a local minima and the simple displacement trial moves of the beads are not efficient for relaxing this potential function.

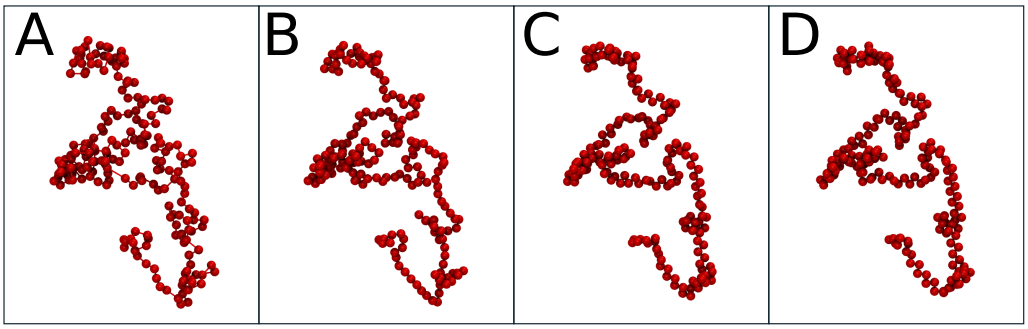

Figure 5.6: Linker configuration before relaxation (A) and after using only the bond potential (B), using the bond and angle potential (C), and using the bond, angle and torsion potential (D). All relaxation runs used the same initial structure (A).

The structures generated by relaxing an initial configuration from the Gaussian chain algorithm can be seen in Figure 5.6. For comparison, single structures have been generated with the full potential, only the bond potential, and the bond and angle potential. The structures have all been generated from the same initial structure and relaxation runs where 1000 sweeps long. The bond potential alone has the biggest visual influence on the structure by achieving a more uniform bond distance. The addition of the angle potential also gives a visual improvement. Differentiating the bond and angle potential structure from the structure with the full potential is difficult as both are very similar. The full

potential adds correct torsion angles in comparison to the other structures.

Comparison of Unfolded Proteins and Linker Model

To understand how well the model describes unfolded protein regions we will compare the radius of gyration R_G, as a measure of compactness, of our model with experimental data. For unfolded proteins the R_G has been determined experimentally in dependence on the protein length with denaturants [102]

$$\langle R_G \rangle = R_0 N^{\nu}, \tag{5.7}$$

with $R_0 = 1.927^{+0.271}_{-0.238}$Å and $\nu = 0.598 \pm 0.028$. The radius of gyration of the Gaussian chain model is

$$\langle R_G \rangle = \sqrt{\frac{1}{6}\frac{N(N+2)}{N+1}}b \approx \sqrt{\frac{1}{6}}N^{1/2}b. \tag{5.8}$$

This scaling behavior is slightly different with $\nu = 0.5$ and $R_0 = 1.555\,$Å. Therefore it is unlikely that the Gaussian polymer without overlap checks reproduces the R_G values of a denatured protein for any number of beads. To compare the R_G scaling behavior of the

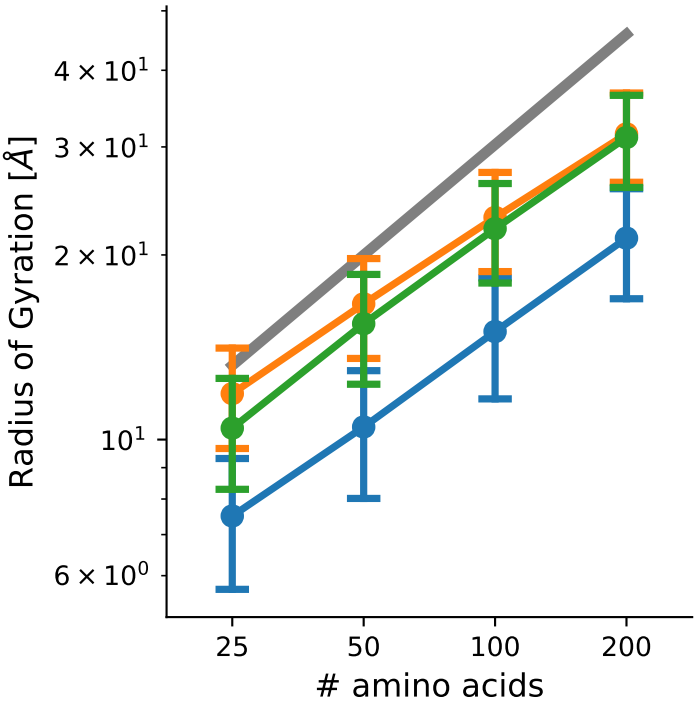

Figure 5.7: Radius of Gyration computed for a linker with different number of amino acids. For each length 1000 structures have been generated with overlap check (orange), without (blue), and full relaxation (green). Error bars denote the standard deviation. The experimental R_G scaling law for denatured proteins [102] is shown as gray line.

linker growth algorithm to the scaling of denatured proteins we generate 1000 different structures for a linker of 25 to 200 beads length. To get the values of a truly free chain for each simulation, the linker was padded with 25 beads in the front and end so that only the middle N beads have been used to calculate the R_G. The bond-length was set to 3.81 Å [19]. Start and end points have been placed at the optimal end-to-end distance, eq 5.3. Results are shown in Figure 5.7. As anticipated without an overlap check the R_G values are systematically different. But with overlap checks enabled, the Gaussian polymer is within one standard of the experimental values if $N < 50$.

It should be noted that for intrinsically disordered proteins it has been shown that the R_G with denaturants is larger than of the protein observed in natural conditions [24, 56, 193, 240]. Therefore our Gaussian polymer model is a good enough description for the flexible domains. A similar model has been employed to study intrinsically disordered proteins [42].

5.3 Implementation

The Complexes model described in Section 2.1.2 has been implemented in a C++14 program Complexes++ and a Python tool pycomplexes. Complexes++ implements the Monte Carlo engine and pycomplexes is a helper library and command line interface (CLI) tool to setup simulations and visualize them, see Figure 5.8. The Monte Carlo integrator accepts input files in the CPLX format and configuration files for simulations. The split enforces that a well defined file format exists that uniquely defines a simulation. We have decided to use the YAML standard [14] for the CPLX files. A library to write YAML files exists for many programming languages allowing easier integration into existing workflow without forcing a specific programming language onto the user unlike other tools [47, 181]. In this chapter, the Monte Carlo engine and helper tools are described separately.

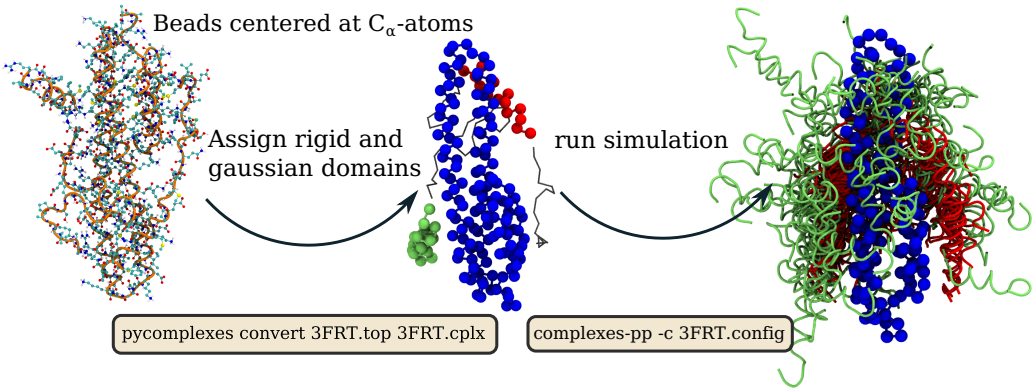

Figure 5.8: Example use case how to go from a single known structure to an ensemble of structures using Complexes++. To prepare the simulation, domain types have to be assigned to amino acids and a CPLX file has to be generated.

5.3.1 Monte Carlo Engine Complexes++

Complexes++ is a Monte Carlo simulation engine implementing the Complexes protein model. The program is written in modern C++14 with a focus on being extensible. The program contains an extensible Monte Carlo engine, the modified LJ potential, eq 2.7 and other pair-potentials, the Monte Carlo movements for different domain types, and a cell-list algorithm [55] to speed up the energy evaluations. It has been developed in collaboration with Berenger Bramas from the Max Planck Computing and Data Facility.

Flexible and Extensible Core Algorithms

The core algorithms of the Monte Carlo engine have the ability to add new trial moves for domains and to add new Monte Carlo algorithms for different statistical ensembles or enhanced sampling techniques. This extensibility is achieved through a combination of run-time polymorphism and use of modern C++14. With run-time polymorphism multiple data types have the same application programming interface (API) and can be used interchangeably. In C++ a data type can either be a class definition or a builtin type like floating point numbers. Run-time polymorphism means that the exact type stored in a variable is only known when the program is executed. Using run-time polymorphism leads to a flexible implementation because it allows writing the core algorithms, like the Monte Carlo integration scheme, independent of the exact domain type chosen for a simulation. It also leads to an extensible implementation because new data-types can directly be used in the code as long as they have the same API. Therefore allowing to add new features with minimal changes to the code. In Complexes++, this technique is used to implement domain trial moves, Monte Carlo algorithms, evaluation of pair-potentials, and a cell-list algorithm.

```
class AbstractMcAlgo {
  // setting the method as '=0' tells C++ that this method has no
  // implementation in the abstract class virtual void
  sweep() = 0;
};

class NVT : public AbstractMcAlgo { void sweep() { // ... } };

class NPT : public AbstractMcAlgo { void sweep() { // ... } };

void monteCarlo(std::unique_ptr<AbstractMcAlgo> algo, int nsweeps) {
  for (int i = 0; i < nsweeps; ++i) {
    algo->sweep();
  }
}
```

Listing 1: Extensible interface for a Monte Carlo algorithm. The comment "..." indicates other implementation details.

In C++ run-time polymorphism is implemented through abstract classes and inheritance. An abstract class does not define exact implementations for all its methods. A method is a function of a class. These methods are called virtual in C++ and define a common API that is shared by all classes that inherit from the abstract class. In Complexes++ abstract classes are marked with the preposition "Abstract". Any class that inherits from an abstract class only has to implement the methods for which no definition exists. C++ will enforce at compile time that the methods marked as virtual without an implementation are defined in the inherited class. As an example take the simplified extensible implementation of the Monte Carlo algorithm, see Listing 1. The abstract class `AbstractMcAlgo` defines a virtual method `sweep` that has no implementation. The method `sweep` is used by the `monteCarlo` function to update all domains by one sweep. Specific Monte Carlo algorithms can now be implemented by inheriting from `AbstractMcAlgo` and defining the sweep. Therefore, allowing to define two Monte Carlo algorithms for different thermodynamic ensembles and have the `monteCarlo` function be independent of the used ensemble. One common concern of run-time polymorphism is that it has a performance penalty during runtime. The most performance critical part of Complexes++ is the energy evaluation. We ensure that this penalty is minimal through careful benchmarks and design of the cell-list algorithm and the pair-potential API.

The most interesting abstract classes to add new features to Complexes++ are `AbstractDomain`, `AbstractPairKernel`, `AbstractConnection`, `AbstractInteractionGrid` and `AbstractMcAlgo`. See Figure 5.9 for a schematic how these classes interact with each other during a simulation.

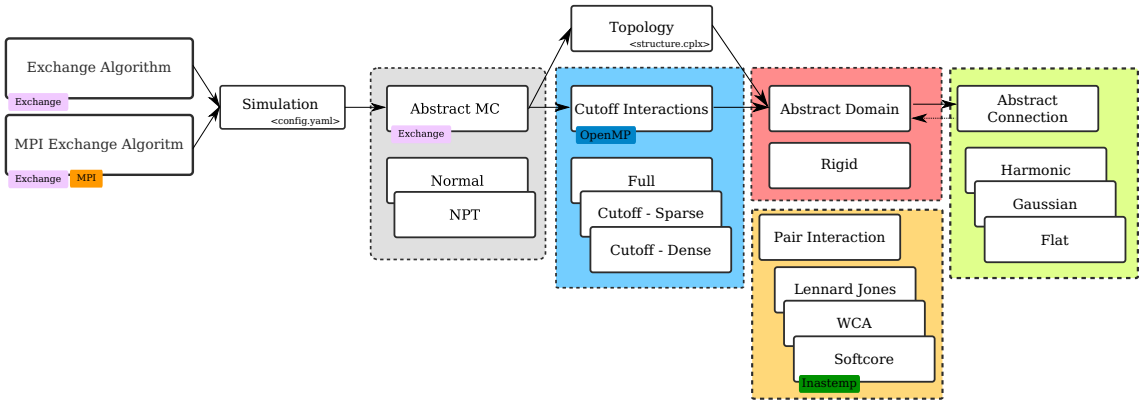

Figure 5.9: Schematic of the classes used in the Complexes++ program and how they interact with each other (arrows). Abstract classes and their explicit implementations are shown as colored boxes. Nested boxes show different implementations of an abstract class. Names are as in the code.

Monte Carlo Algorithm

The implementation of Monte Carlo algorithm in Complexes++ follows closely the example shown in Listing 1. The main logic to run a Monte Carlo simulation is implemented in `AbstractMcAlgo` with a virtual method called `sweep` to implement a sweep. At the time

of writing Complexes++ implements sweep functions for the NVT and NΠT ensemble, where Π is the osmotic pressure, [55] and an algorithm for BDMC, Chapter 4. In the remainder of this thesis we will refer to the NΠT ensemble as the NPT ensemble. In addition three different acceptance functions, Metropolis [134], Glauber [63] and Optimal, Chapter 4, have been implemented as well. All three Monte Carlo algorithms work with all acceptance functions.

The sweeps are defined differently for all three Monte Carlo algorithms. In the NVT ensemble domains are chosen randomly from a uniform distribution. If N is the number of domains than a sweep will make N trial moves and select a random domain with probability $1/N$ for each trial. In the NPT ensemble trial moves are generated for the domains and the volume. To account for the additional volume move a sweep consists of $N + 1$ trial moves. At each trial, the probability to make a volume move is $1/(N + 1)$. If a domain move was chosen the domain to be moved is chosen randomly with a $1/N$ probability. For the volume moves the domains are re-scaled so that the centroid of the domain is scaled by $\sqrt[3]{(V + dV)/V}$. In the BDMC sweep, trial moves are guaranteed to update every domain to ensure all domains are propagated in time. The order in which domains are updated is randomized every sweep.

Boundary Conditions

Complexes++ implements PBC [55] for the centroids of all domains. Under these conditions, the centroid of all domains is placed inside of the unit-cell. Complexes++ only implements rectangular unit-cells. Because Complexes++ keeps the centroid of a domain within the box it can happen that some beads are outside of the box.

Complexes++ determines the distance between two beads via the minimum image convention (MIC) [55]. Complexes++ uses an efficient implementation to calculate the minimum image distance, Algorithm 1. This algorithm is an extension of a fast algorithm [40], where the while-loop ensures it works for any distance, i.e, if a trial move displaces a domain by more than one periodic image. To keep the number of iterations of the while-loop as small as possible Complexes++ ensures that the centroid of a domain are inside the unit-cell.

Replica Exchange Algorithms

Complexes++ implements replica exchange algorithms (see Section 2.6 for enhanced sampling). Replica simulations require that multiple simulations are started. Complexes++ requires that every replica is started in its own folder. Multiple simulations can be started with the `-multidir` flag and a list of the folders containing individual replicas. The `-multidir` option alone only tells Complexes++ to simulate multiple replicas. To activate the exchange the two flags `-replex` and `-replex-accept` have to be used as well. `-replex` states after how many sweeps an exchange should be attempted. `-replex-accept` stets the exchange function to be used to change between the NVT, NΠT ensembles and Hamil-

ALGORITHM 1: Algorithm to efficiently convert the distance between two beads to the minimum image distance it domain centers are inside of the simulation box.

```
1  function mic(distance[3], box[3])
2      for i=0; i<3; ++i do
3          while distance[i] > .5 * box[i] do
4              distance[i] -= box[i]
5          end
6          while distance[i] < .5 * box[i] do
7              distance[i] += box[i]
8          end
9      end
10     return distance;
```

tonian replica exchange. In Complexes++ exchange attempts are done with neighboring replicas using the odd-even scheme.

The replica exchange simulations are also implemented in an extensible manner. Because the different replica exchange algorithms implemented require different variables to be changed between the replicas, Complexes++ only exchanges the coordinates of the beads and box dimensions during an exchange. The currently implemented algorithms only require to change the acceptance function such that new replica simulation protocols like Bayesian ensemble refinement [82] can be added easily.

Random Numbers

Complexes++ uses pseudo random numbers to generate trial moves and accept moves. As a pseudo RNG it is using the Mersenne Twister [129] implementation in the C++ standard library. The Mersenne Twister is a reasonable choice for Monte Carlo simulations and produced better results than previously employed linear congruent generators [34]. Because the RNG cannot be safely used in a multi-threaded program like Complexes++ it uses a different instance of the RNG for each thread. The individual RNG instances are seeded from random numbers from an initial RNG instance during setup. To achieve reproduceability Complexes++ requires the user to provide an explicit seed for the initial RNG. This requirement ensures that two runs with the same input are identical on the same computation node.

Pair-Interactions Potentials

In Complexes++ pair-potentials are called pair-kernels following a common terminology used in computer science. The parameters of all available pair-kernels are given in the forcefield class. The parameters σ_{ij} and ϵ_{ij} are set according to the respective bead types and shared between the pair-kernels. During a simulation pair-kernels can be chosen for each individual domain type pair present in the simulation, allowing to fine tune the interactions. The LJ like potential eq 2.7 in combination with the electrostatic potential

eq 2.9, and several other potentials have been implemented. One additional potential is the Weeks-Chandler-Anderson (WCA) potential [226]

$$U_{\mathrm{WCA}}(r,\sigma_{ij},\epsilon_{ij}) = \begin{cases} -\epsilon_{ij} & \text{if } r < 2^{1/6}\sigma_{ij} \\ 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] & \text{if } \epsilon_{ij} > 0. \end{cases} \tag{5.9}$$

A smoothened LJ potential that smoothly decays to zero, with a cutoff distance b, is implemented with the following smoothing term

$$F_{\mathrm{smooth}}(r,a,b) = \begin{cases} 1 & \text{if } r/\sigma_{ij} < a \\ 0 & \text{if } r/\sigma_{ij} > b \\ \frac{(b^2-(r/\sigma_{ij})^2)^2(b^2+2(r/\sigma_{ij})^2-3a^2)}{(b-a)^3} & \text{otherwise ,} \end{cases} \tag{5.10}$$

with $a = 1.4\,\text{Å}$ and $b = 1.8\,\text{Å}$ being the bound in which the potential decays to 0. The value for a and b are hard coded. A purely repulsive potential with

$$U_{\mathrm{repulsive}} = \left(\frac{\sigma_{ij}}{r}\right)^{12} \tag{5.11}$$

is also implemented. We also added a soft-core potential [8] that allows to tune how soft the beads are and therefore if they can overlap. The soft-core potential is a modification of the LJ like potential without the hard core,

$$U_{\mathrm{SC}}(r_{ij}) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}^6}{\alpha\sigma_{ij}^6 + (r_{ij}-s)^6}\right)^2 - \left(\frac{\sigma_{ij}^6}{\alpha\sigma_{ij}^6 + (r_{ij}-s)^6}\right)\right], \tag{5.12}$$

with α the parameter to tune the softness of the beads, and $s = \left(\sqrt[6]{2} - \sqrt[6]{2-\alpha}\right)\sigma_{ij}$ a shift parameter to ensure that the minimum is always at $\sqrt[6]{2}$ independent of α. The other branches of the LJ like potential can be obtained by applying the same modifications. α can be changed in the range of zero to one, with $\alpha = 1$ allowing full overlap of the beads as $U_{\mathrm{SC}}(r=0) = 0$ and $\alpha = 0$ recovering the LJ like potential $U_{\mathrm{SC}}(r=0) = \infty$, eq 2.7. Because this potential explicitly allows overlaps the electrostatics potential also has to be changed to remove the divergence at $r = 0$. For this we use the potential between two Gaussian charge distributions [235]

$$U_{\mathrm{el}}(r_{ij}) = \frac{q_i q_j}{4\pi\epsilon_0 D}\frac{\operatorname{erf}\left(r_{ij}\sqrt{\lambda_{ij}}\right)}{r_{ij}}\exp\left(-\frac{r_{ij}}{\zeta}\right)\frac{1}{k_{\mathrm{B}}T} \tag{5.13}$$

with $\lambda_{ij} = \frac{\lambda_i\lambda_j}{\lambda_i+\lambda_j}$ and λ_i the charge radius of bead i. The charge radii are specified in a forcefield for every bead type. In the standard Complexes++ forcefield all radii are set to one.

The different pair-kernels are implemented with a common abstract base class `AbstractPairKernel`. The LJ like and WCA potential have been implemented using Inastemp, a portable single instruction multiple data (SIMD) library [25].

Bead and Domain Implementation

No specific bead class exists in Complexes++, instead, domains contain all information specific to the beads as several lists. This design scheme follows the "struct of arrays" pattern [1] used to optimize memory access. For the beads a domain stores the coordinates in a `m_xyz` field, the charges in `m_charges` and the bead types in `m_beads`. These three fields are the only information needed to evaluate the potential energy with the pair-potential functions. Domains are also implemented with run-time polymorphism in an AbstractDomain class that contains information about the beads and coordinates. Derived classes only need to implement trial moves.

In addition to bead information, a domain also contains a unique id and a type id that are defined at runtime. The type id is used to find the corresponding pair-kernel when evaluating the energy with a different domain. Choosing a pair-kernel for domain type pairs and using a struct of arrays pattern allows evaluating the pair-kernel for groups of beads. This pattern can be efficiently implemented using Inastemp [25]. Because the evaluation of the pair-kernel is the most expensive calculation of Complexes++ this pattern ensures that Complexes++ has good performance while using run-time polymorphism. The type ids are also used to create domains with the same type of move but different parametrizations for it. For example, the rigid domains have two parameters for translation and rotation. Having the final parametrization set at runtime allows creating two distinct rigid domain types for different proteins. This flexibility is useful for simulations with a mixture of small and large domains. The translation and rotation of larger domains can be chosen smaller to increase the acceptance rate for their moves and small domains can be set to large translation and rotation. To achieve an optimal phase space exploration rate.

In Complexes++ two domain types are implemented, a rigid type and a Brownian type. For the Monte Carlo moves the rigid domains can be translated by an arbitrary vector, each component is chosen randomly from the range $[-a, +a]$, with a the maximal displacement. The rotations are generated by choosing a random rotation axes in the unit cube and a random rotation angle [55]. This causes that rotation axes along the edges are more likely chosen. Because the move still obeys detailed balance it does not affect generated ensembles. In each trial move the probability to make a translation or a rotation move is one half. The Brownian domains implement the translation and rotation algorithm for the BDMC, Chapter 4.

Flexible Linkers and Connection Potentials

As described in Section 5.2 the flexible linker domains are replaced with effective potentials from a Gaussian chain polymer model. For this Complexes++ implements connection potentials in a `Connection` class. A connection contains the ids of the two domains that are connected and the corresponding bead ids in the domains. For each domain, a list of connections is stored in the class. So if domains i and j are connected both contain

the same connection. This duplication of data makes it easier to find the corresponding connections for a domain during the energy calculation. This compromise was chosen as it is expected that the number of connections is significantly smaller the number of beads. Currently implemented connection potentials are a flat potential (zero everywhere), a harmonic potential, and a Gaussian chain PMF potential, eq 5.6.

Cell-List Algorithm

The other important part of the performance of Complexes++ is the cell-list algorithm [55] used to reduce the number of interactions needed to evaluate. Cell-list algorithms reduce the computation time to calculate the full energy from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ with N being the number of beads in a simulation. Because beads linked together as domains the algorithm stores for every cell continues intervals of beads that are in the corresponding cell. Every bead in a domain is given a unique id defined by the order in which they appear in the input files. The interval will therefore only store the id of the first bead entering the cell and the length of beads in the cell. For domains that are larger than a single cell it can happen that two different intervals are in a cell, see the red domain in Figure 5.10 that has two intervals in the cell (1, 4). In that case, the cell will store two intervals for the same domain. An interval is stored in the `CoInterval` class, Listing 2. Each cell stores a list of CoInterval instances. As an example of the data stored for a list take the cell (1, 4) containing the red domain in Figure 5.10. Our implementation is different from conventional cell-

```
class CoCell {
  // The list of intervals inside the current
  cell std::vector<CoInterval> m_intervals;
  // ...
};

class CoInterval {
  // The domain related to the current interval
  int m_domainId;
  // The position of the first element of the element-list
  int m_beginingOfInterval;
  // The number of elements in the current interval
  int m_nbElementsInInterval;
  // ...
};
```

Listing 2: Definition of the CoCell and CoInterval class used in the cell list algorithm implemented in Complexes++. The comment "..." indicates other implementation details.

list implementations in standard MD codes. In conventional implementations the linkage between beads is not considered. We however explicitly store information about the linkage of beads (i.e. belonging to the same domain) to reduce the computational complexity of

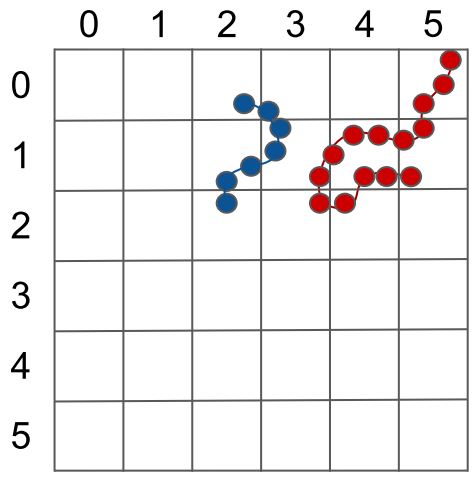

Figure 5.10: Example configuration for two domains (blue and red) in a 2D grid. Numbers on the top and left are used to index cells.

the energy evaluation. After moving a domain we can immediately calculate on which cells we have to evaluate the energy and which beads are affected.

To calculate the complete potential energy between all domains we iterate over all cells in the cell-list, see Algorithm 2. For each cell we then iterate over the list of CoIntervals. During a Monte Carlo move only a single domain will be moved. We therefore do not need to recalculate the complete potential energy. Instead we need iterate over the cells occupied by the moved domain and update the energy difference appropriately, replacing the outer most loop in Algorithm 2 with a loop over only the cell currently occupied by the domain. The last step further reduces the computational effort to calculate energies during a simulation.

The interface to the cell-list algorithm to calculate pair-interactions has also been implemented without references to the underlying algorithm and can be exchanged with different algorithms. In Complexes++, there are two data-structures available for the cutoff grid. The *dense* data structure is allocated cell objects for all cells in a simulation box and offers efficient computation of neighboring cells. The memory consumption of this data structure scales with $(L_{\text{Box}}/L_{\text{Cell}})^3$, where L_{Box} is the edge-length of the simulation box and L_{Cell} is the edge-length of a cell, see Figure 5.11. This data structure is optimal for dense simulations. The *sparse* data structure uses a hash-map to only allocate cells that are occupied by beads. The memory consumption of this data structure scales with the number of beads in a simulation and is independent of the box size. It is suited for sparse simulations. The high-level interface is also agnostic to the underlying cell-list algorithm allowing to chose a completely different algorithm if desired.

Task-based parallelism

Monte Carlo algorithms have two parts which can be parallelized, the energy calculation and the trial move generation. The trial move generation in Complexes++ is not computationally expensive and a minuscule amount of time is spend on it. We therefore optimize

ALGORITHM 2: Cell list algorithm to calculate all pair-interactions.

```
1  function computeEnergy(Cells[K], Domains[M])
2      energy = 0
3      // Compute energy (particle to particle interactions)
4      for cell in Cells do
5          for target in cell do
6              for source in cell do
7                  if source.domainId ≠ target.domainId then
8                      target_dom = Domains[target.domainId]
9                      source_dom = Domains[source.domainId]
10                     energy += kernel(target_dom, source_dom, target, source)
11                 end
12             end
13             for nb_cell in cell.neighbors() do
14                 for source in nb_cell do
15                     if source.domainId ≠ target.domainId then
16                         target_dom = Domains[target.domainId]
17                         source_dom = Domains[source.domainId]
18                         energy += kernel(target_dom, source_dom, target,
                               source)
19                     end
20                 end
21             end
22         end
23     end
24     return energy
```

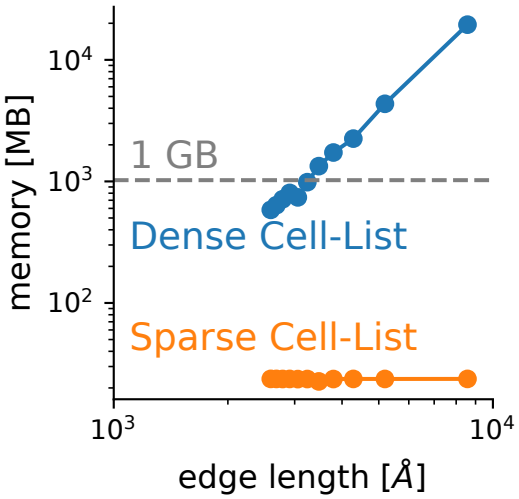

Figure 5.11: Memory consumption of the dense (blue) and sparse (orange) cell-list data-structures with a constant number of beads and constant size of the simulation box. The cutoff is set to 12 Å. The gray line marks one gigabyte.

the energy calculation for the single replica simulations. In replica exchange Monte Carlo all replicas can be executed in parallel with few synchronization points to exchange co-

ordinates. Ideally, the single replica and replica exchange simulation can use the same parallelization scheme. It has to be taken into account that replicas can be unbalanced or change during a simulation if a phase transition occurs (from gas to liquid, for example). Another requirement was that it should be possible to always use the maximum number of available central processing unit (CPU) cores independent of the number of replicas. Therefore a single replica can use multiple threads and in the case that more replicas than available threads are simulated the different replicas have a scheduler queue.

In Complexes++ this problem is solved using the task API in OpenMP [22]. Tasks are generated based on either domains or cell-lists (if activated), meaning tasks are small and plentiful. At the beginning of the energy calculation, the number of tasks is distributed equally to the available threads. If a thread finishes early it can steal tasks from other threads. Allowing threads to be always busy and balancing load when different replicas have different work loads. The replica exchange simulations can run in parallel. To make better use of available resources on common HPC clusters we have implemented an MPI version to spread replicas to multiple nodes.

CPLX File Format

The input for Complexes++ simulation is stored in a CPLX file similar to the tpr files in GROMACS. The CPLX file contains all information about the domains and pair-kernels to simulate, with the exception of the parameters for the Monte Carlo algorithm. The CPLX file format is based on YAML [14] and divided into four sections *box, definitions*, *topologies*, and *forcefield*.

The *box* section contains the dimensions of the rectangular simulation box as a YAML list. The length in each dimension is given in Å.

The *definitions* section defines the type of domains that are used in the simulation and the pair-kernels used to calculate the energy between domains. The *definitions* section is separated into two sub sections to define the final domain-types and their interactions between them, called *domains* and *pair-interactions* respectively. The *domains* section contains a dictionary with the entry names being the domain names and the definition for the domain. A definition contains the type of move the domain can make, either rigid or Brownian, and the parameters for the move. Allowing to define several rigid bodies in a simulation that have different rotation and translation parameters. The definitions can be used for systems with a mixture of large and small rigid bodies to model the corresponding mobility. The *pair-interaction* section defines which pair-interaction potential to choose for each combination of domain types defined in *domains*. In the definition, one can define more domain types than will be used in a simulation. Allowing to create standard definitions (like the KH forcefield) for different applications. See Listing 3 for an example definition using two domain types **A** and **EM**. Here the **EM** type is set to not move at all and interacts with **A** domains using the WCA potential. This definition could be used to fit any domain of type **A** into a domain defined by **EM**, e.g., electron densities from cryo electron microscopy experiments.

```
definitions:
  domains:
    A:
      defaults: {rotation: 2, translation: 1}
      move: rigid
   EM:
      defaults: {rotation: 0, translation: 0}
   move: rigid
pair-interaction:
  - domain-type-pair: [A, EM] function: WCA
  - domain-type-pair: [A, A] function: LJH
  - domain-type-pair: [EM, EM] function: None
```

Listing 3: Complexes++ domain definitions for a simulation with two domain types **A** and **EM**. Both domain types move as rigid bodies but they have different pair-interaction potentials with each other.

The *topologies* section defines the actual domains that are used in the simulation. It consists of a YAML list of topology entries. Each topology entry contains domain definitions and connections between domains of the same topology. A domain contains the following fields: beads, chain-ids, charges, coordinates, nbeads, type, mc-moves, meta-data, name. Here the type field specifies the domain type, which has to be defined in the definitions section. Domains have to be numbered consecutively and uniquely across all topologies.

The *forcefield* section contains definitions for the available bead types as well as the energy and diameter pair-parameters.

5.3.2 Preprocessor pycomplexes

pycomplexes is a Python library and CLI program that includes several tools to help setup simulations for Complexes++ and analyze them later. The CPLX format accepted by Complexes++ is versatile complex and gives the user a lot of freedom in setting up simulations. A lot of simulation setups do not need to leverage the full flexibility of Complexes++ and therefore pycomplexes includes a tool called `convert` that takes as input a simplified format, called a TOP file, for describing simulations and generating CPLX files from it. As part of the conversion, the script will automatically choose charges and interaction energies based on amino acid type. As interaction energies, the user can choose either the KH model or the unaltered MJ model. The charges are set according to the KH model in both cases. The interaction potential is chosen based on domain type, so far allowed are rigid, gaussian and brownian. The rigid and brownian domain type use the modified Lennard Jones potential, eq 2.7. For the gaussian domains positions of the C_α beads will be stored in the CPLX as rigid domains that do not move and a connection potential, eq 5.5, will be used between the two rigid domains connected by the gaussian domain. To define a domain in the TOP file the type and a selection of beads have to be

```
box: [100, 100, 100]
topology:
  protein-name:
    coordinate-file: structure.pdb
    move: true
    domains:
      A:
        type: rigid
        selection: 'namd CA and segid A'
      link:
        type: gaussian
        selection: 'name CA and segid L'
        start_connection: [A, 'segid A and resid 10']
        end_connection: [B, 'segid B and resid 10']
      B:
        type: rigid
        selection: 'name CA and segid B'
```

Listing 4: TOP file for a simulation for two rigid domains connected by a Gaussian domain. All selections are written in the atom selection language used by MDAnalysis.

specified. For the gaussian domain, in addition, the beads of the rigid domains connected by the gaussian domain have to be specified as well. The convert script could ignore known beads for a gaussian domain in the structure but keeping the information in the CPLX file allows to generate explicit linker positions in the post processing, i.e., with the `addlinker` tool.

The structures and selections are read and parsed using MDAnalysis [65, 135]. In the molecular dynamics community there exists a large variety of structure file formats and they are often not well defined or popular programs write and accept ill formatted files, MDAnalysis helps to read a large variety of different formats with an easy to understand atom selection language.

In addition to the `convert` command pycomplexes also include a variety of other commands to help the user setup and visualize simulations. As of the time of writing the other implemented commands are

- `equilibration` Update coordinates stored in a CPLX from a trajectory.
- `forcefield` Convert a forcefield using scaling parameters λ and e_0, see eq 2.8.
- `visualize` Create VMD scripts from simulations.
- `demux` generates input files for the GROMACS tool `trjcat` to generate time-continuous trajectories from replica exchange simulations.
- `addlinker` generates explicit linker conformations for a simulation with Gaussian linkers.

Add Explicit Linker Beads in Post-Processing

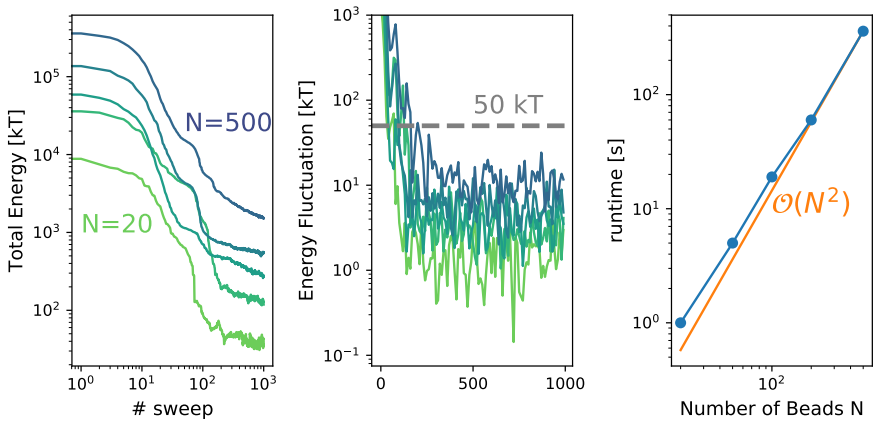

Figure 5.12: (Left) Total energy for a linker chain of varying length, between 20 and 500 beads. (Middle) Energy fluctuations from 10 sweeps during the simulation. Fluctuations are calculated as standard deviation. (Right) Runtime for the linker relaxation depending on the linker length. The orange line marks quadratic scaling behavior in this log-log plot.

The Complexes++ Monte Carlo engine treats the linker domains as harmonic springs and does not calculate explicit positions for them. The calculation of explicit bead positions can be done with the `addlinker` command of pycomplexes. The linker beads are generated as explained in Section 5.2. To quantify the current implementation we have generated and relaxes linkers for 5 different numbers of beads [20, 50, 100, 200, 500]. Interestingly the number of sweeps needed to find a relaxed structure seems to be independent of the length of the linker, Figure 5.12 (Left). This allows to use a general limit on the number of sweeps for relaxation runs independently on the linker length. Looking at the fluctuations in the energy between sweeps we notice that the energy has stabilized after around 100 sweeps for all linkers, Figure 5.12 (Middle). The magnitude of fluctuations is again independent of the linker length, therefore the fluctuations can be used to implement an adaptive stop criterion for the relaxation. Here we determined the fluctuations as the standard deviation for the potential energy of ten sweeps. Suggesting from these simulations would be to stop after the fluctuations are below 50 kT. The current implementation of the linker relaxation evaluates the full potential energy for a move, leading to a quadratic run-time for the relaxation algorithm, Figure 5.12 (Right).

5.3.3 Benchmarks

For the benchmarks we use three different structures, Table 5.1. The structures differ in size, from small and compact to large and elongated, to show how Complexes++ performs with respect to number of threads in relation to protein size and cell-size. For all benchmarks, if not further specified, the volume density N/V is set to 0.1 , the number of proteins in a simulation is 128 , the cell-size is set to 12 Å, the number of sweeps is 5000,

Name	R [Å]	N	Reference
6LYZ	22	129	Diamond [41]
1HE8	48	749	Pacold et al. [148]
4HPQ	167	1282	Ragusa et al. [161]

Table 5.1: Properties of test structures for benchmarks. N is the number of amino acids and R is the radius of the smallest sphere encompassing the structure.

the temperature is set to 300 K, the Metropolis acceptance function is used, the Debye length is 10 Å, the dielectric constant is 80 and we use the KH model.

The first benchmark demonstrates the scaling behavior of the cell-list algorithm with increasing number of beads using a single thread, Figure 5.13. The number of protein copies in the simulation was varied between 2 and 1024. The largest simulation contained more than one million beads. For all three studies structures, the runtime increases linearly with the number of proteins in the simulation. If the runtime is scaled by the number of beads in the simulation it becomes clear that the overall performance is correlated with the number of beads and that the scaling is linear and independent of domain size.

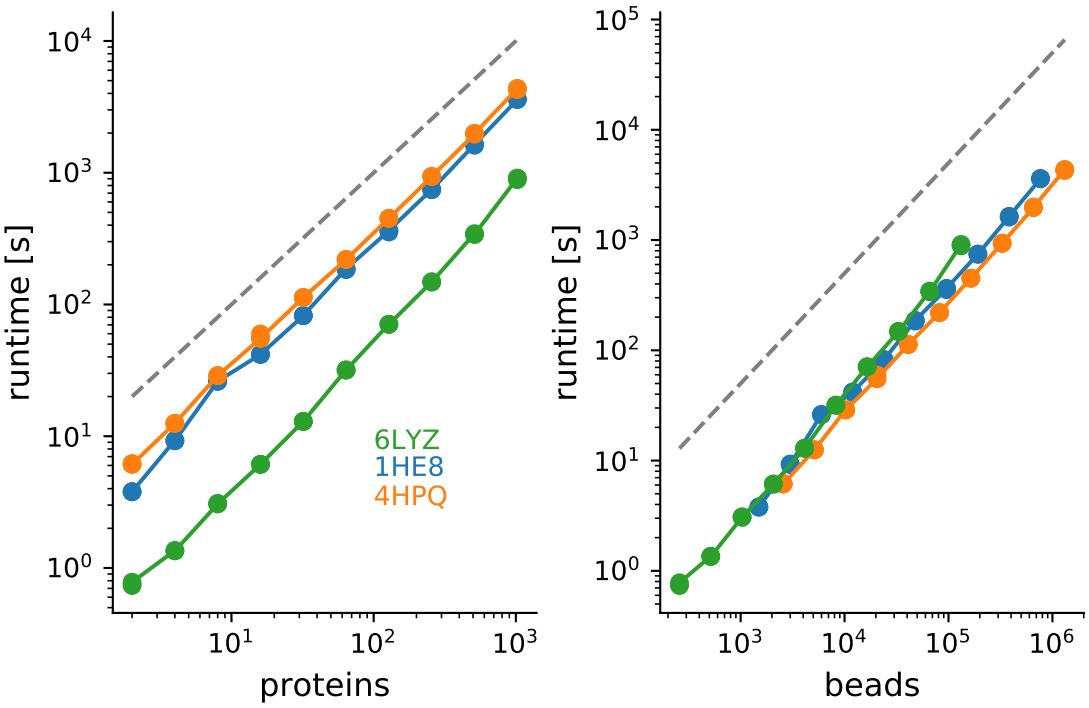

Figure 5.13: Scaling behavior of Complexes++ with increasing number of proteins (left) or number or beads (right) in a simulation. The dashed gray line indicates linear growth in this log-log plot. All simulations have been performed using a single thread.

The performance only scales well with the number of used threads for large structures and a small number of threads, as expected due to the use of a cell-list algorithm to minimize the number of energy evaluations. In Figure 5.14 it can be seen that the performance only increases up to around eight threads independent of the number of proteins in the

simulation. When we normalize the runtime with respect to a single thread, Figure 5.15,

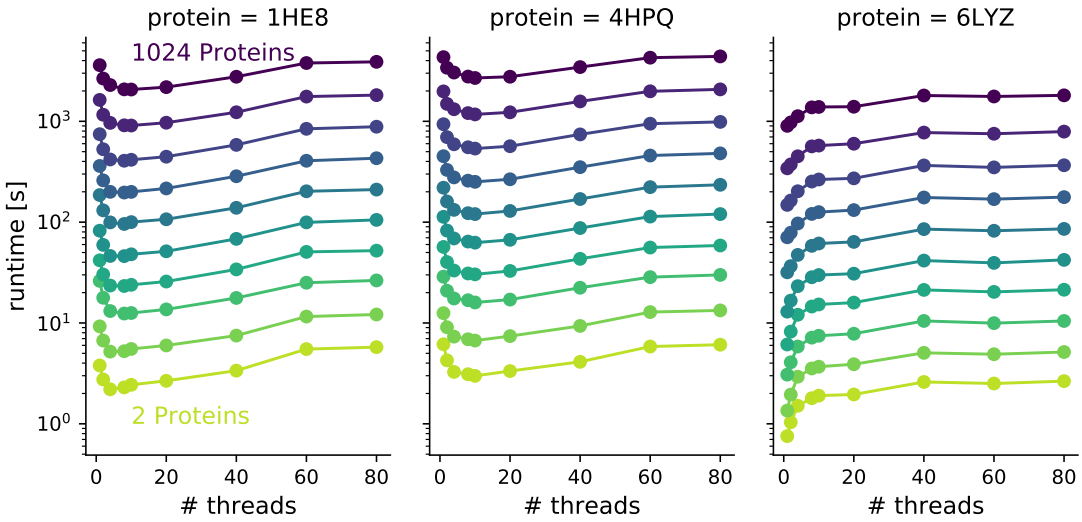

Figure 5.14: Runtime of single replica simulations using an increasing number of threads. The dashed lines mark one minute and one hour.

it becomes clear that for small proteins like 6LYZ using more than one thread will result in a performance degradation. For the two larger proteins we see a doubling of performance when using eight threads per replica compared to single thread runs. When more threads are used the performance is degrading. As a rule of thumb from these simulations the

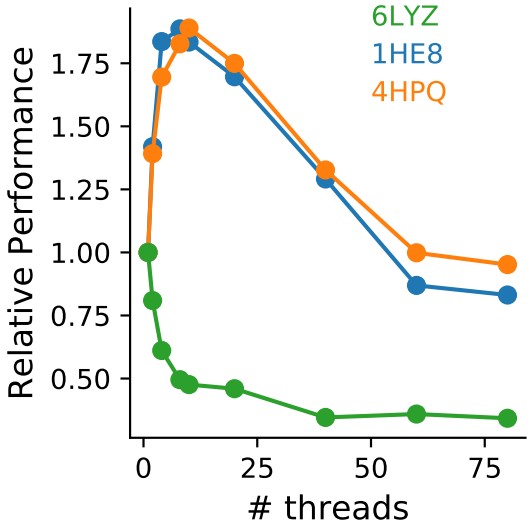

Figure 5.15: Relative performance of Complexes++ with increasing number of threads. The performance has been scaled with respect to the single threaded performance.

standard for simulations should be one thread per replica as an initial choice, especially for small systems. It can be beneficial to allocate more threads per replica but using a full computation node for a single replica is likely to result in less than optimal performance. Note that these results are for single-replica simulations only. When running

multiple-replica simulations Complexes++ can make use of more threads and nodes. For multiple-replica simulations the performance can be optimized when taking care that the number of threads per replica is around eight in our homogeneous benchmark simulations, see Figure 5.16.

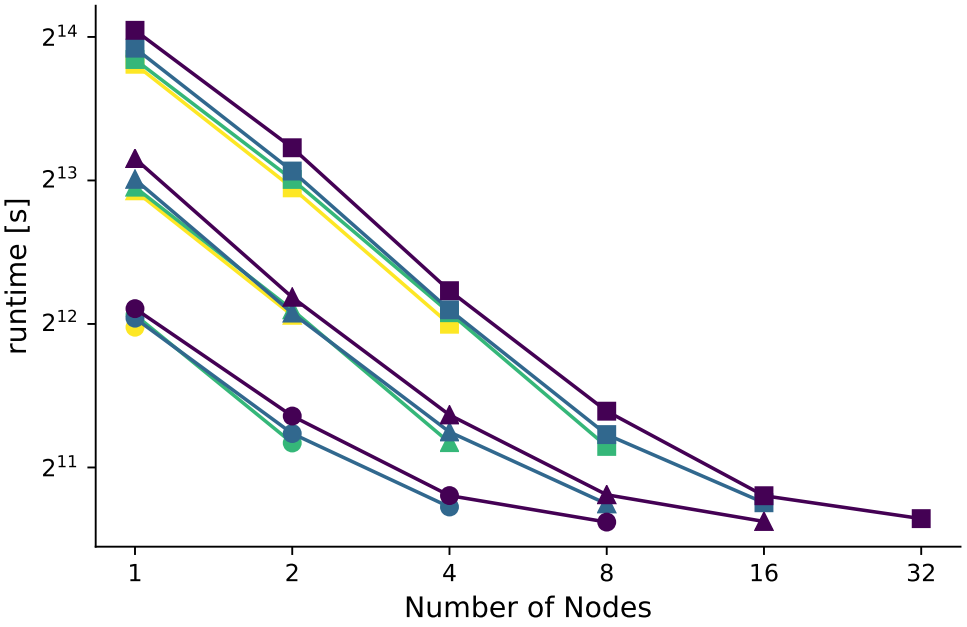

Figure 5.16: MPI execution of replica exchange simulations of the 4HPQ structure using 1 - 32 nodes. The number of threads per replica is varied from one (yellow) to eight (dark blue) in powers of two. Three sets of simulations with a different number of replicas, 64 (cricle), 128 (triangle), and 256 (square), have been run.

Another aspect for single-replica performance besides the number of used threads is the cell-list data structure. For a single replica with 128 proteins the choice of cell-list data structure has a significant impact on runtime, Figure 5.17. In general the dense data-structure has a better performance than sparse, with the exception for the 4HPQ structure at a density of 0.004 with 10 threads, and 6LYZ with a density of 0.13 and 40 threads. When comparing the relative performance between sparse and dense the dense data structure is around 30-40% faster than sparse for large structures and around 10% for small structures, Figure 5.18

The memory consumption between the different data structures varies greatly though, Figure 5.19. With the sparse option, the cell-list is usually in the order of a few megabytes, while dense can consume up to two 20 gigabytes in our testing, Figure 5.19. The choice of cell-list data-structure should be taken into account when setting up multiple-replica simulations on the same node to keep the memory requirement of all replicas below the available memory on the node, i.e. the user has trade-off if he using the sparse cell-list with more replicas will result in better sampling of configuration space than using the dense cell-list at the same wall time.

We have also run simulations with a heterogeneous mixture of 41 different proteins.

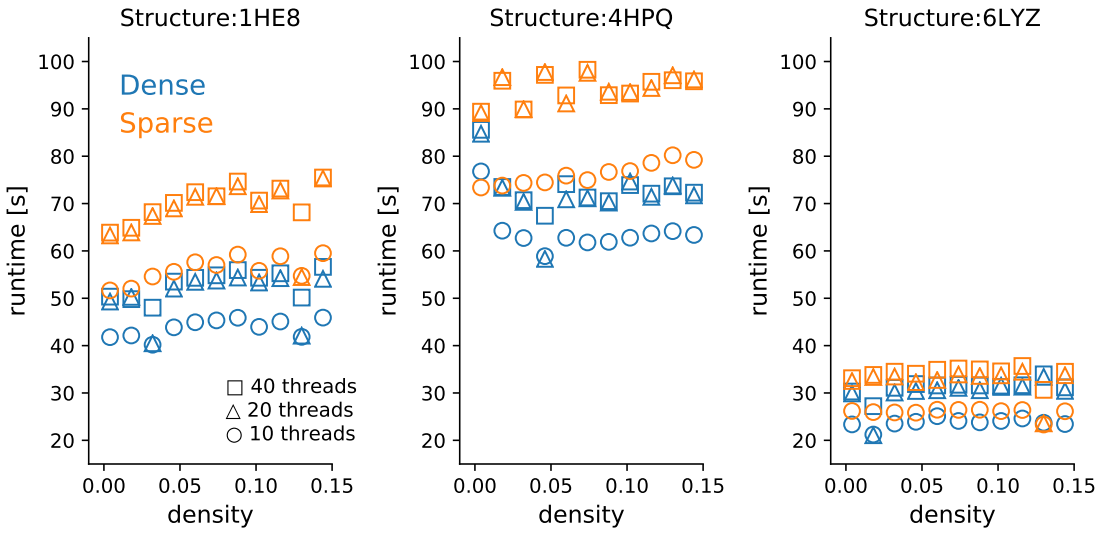

Figure 5.17: Total runtime of single-replica simulations for the dense (blue) and sparse (orange) cell-list data structures at different volume densities N/V. The number of threads has been varied between 10 and 40.

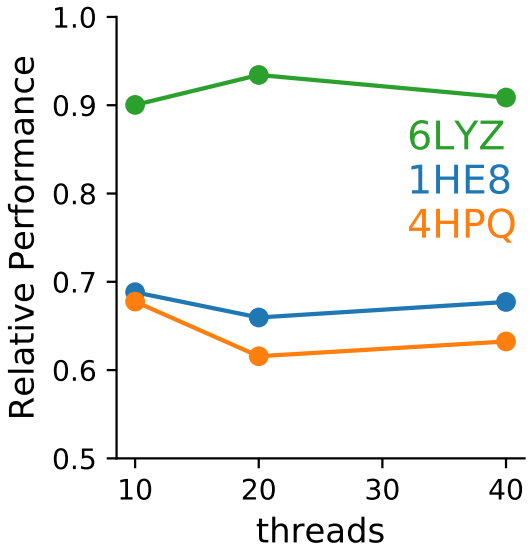

Figure 5.18: Relative performance of the sparse data-structure to the dense data-structure for the cell-list algorithm. A value below one means the dense data-structure is faster.

The proteins and their quantities have been taken from McGuffee and Elcock [130], we only selected proteins that could be directly parsed from the Protein Data Bank (PDB), i.e., no alternative locations for residues, and that only contain residues included in the KH model. The system used a box edge-length of 400 nm and a total of 611 proteins. The cell-list box-length was set to 3 nm. For this system, Complexes++ can make more efficient use of multi-threading. The peak-performance is at 20 threads with a speed-up of ~ 7. Optimal scaling is only achieved up to 10 threads, Figure 5.20.

Single replica and MPI benchmarks have been performed on the CORBA cluster of the

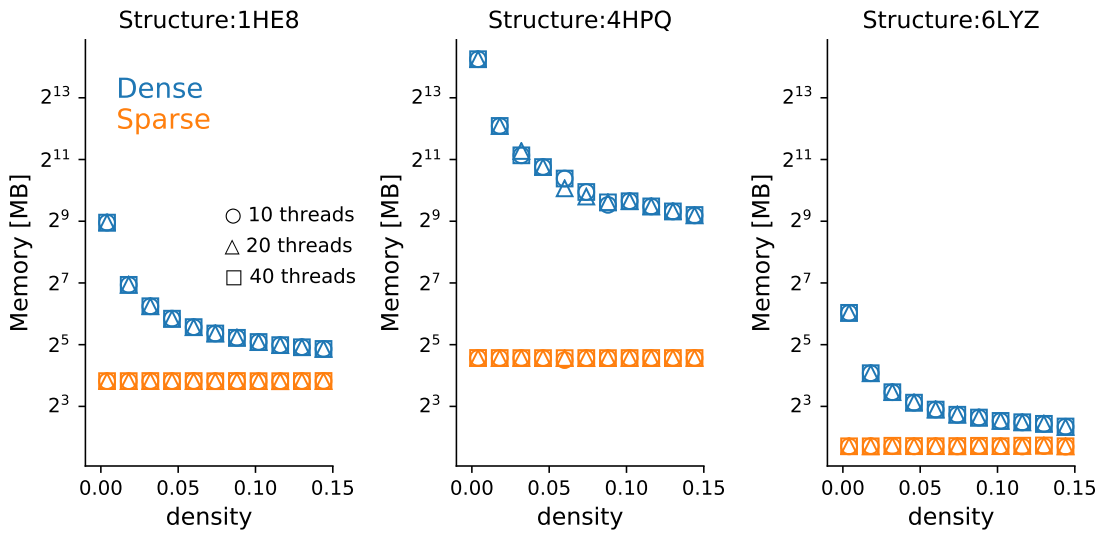

Figure 5.19: Total memory consumption of the cell-list for the dense (blue) and sparse (orange) data structures at different volume densities N/V. The number of threads has been varied between 10 and 40.

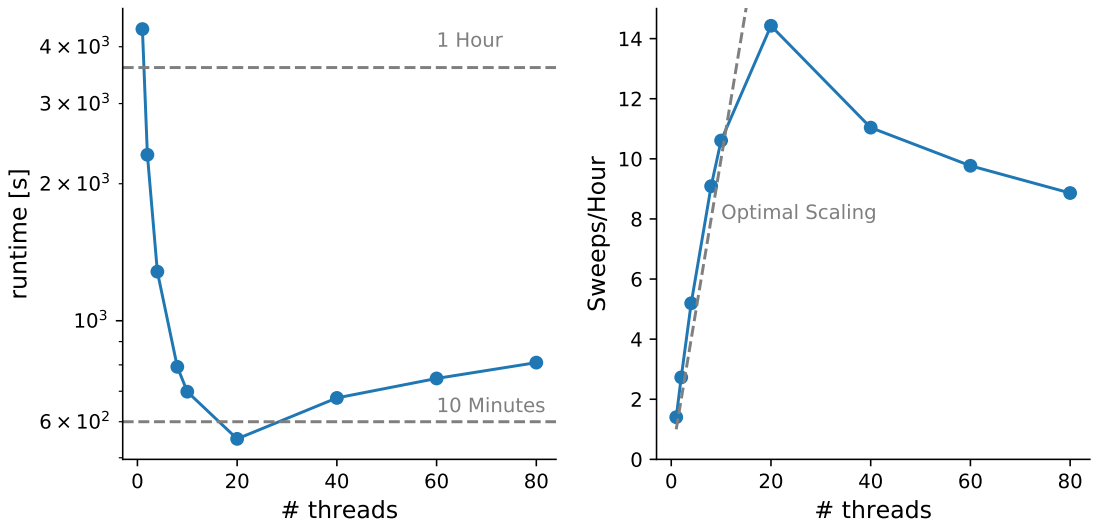

Figure 5.20: (Left) Runtime of heterogeneous protein mixture for a different number of threads. (Right) Number of sweeps per hour for a different number of threads.

MPG. A node consists of two Intel Xeon "Skylake" processors with 20 cores @ 2.4 Ghz per processor, and 96-192 GB of memory. Cell-list algorithm benchmarks have been performed on the DRACO cluster of the MPG. A node consists of two Intel Xeon E5-2698 processors with 16 cores @ 2.3 Ghz per processor, and 128 Gigabyte of memory.

5.4 Summary

Complexes++ is a new Monte-Carlo simulation engine with an efficient and extensible implementation of the original KH forcefield model. We ensured in development that it can run efficiently on a laptop as well as on a high-performance cluster. We have replaced the polymer model for flexible domains with a Gaussian chain model to efficiently sample configuration space of connected rigid domains. To generate linkers we have also added a post-processing tool. We extended the original KH model by adding new potential energy terms to choose, an NPT ensemble, and different replica exchange algorithms. Defining a simulation for Complexes++ is a non-trivial process. Due to the complexity of the systems of interest. The program gives the user a lot of control over the simulation. To enable an easier simulation setup for less experienced users we developed the pycomplexes helper tool.

Chapter 6

Conclusion & Outlook

In this thesis, we developed novel methods to understand the role of diffusion in molecular systems. In Chapter 3 we showed that the quaternion covariance equations from Favro can be directly fitted to simulation data without further assumptions, allowing us to fit anisotropic rotation diffusion tensors more accurate. We found that similar to translation diffusion the rotation diffusion tensor is influenced by PBC and derived a correction. Using our algorithm and box-size correction facilitates an accurate comparison of experiments and simulations. Similar to the work of Hoffmann et al. [74] such comparisons can be used to improve existing forcefields. Recent simulations of high density protein solutions have shown that the translation and rotational diffusion coefficient are slowed down more then expected from non-interacting hard-sphere colloidal theory [223]. This effect is due to protein-protein interactions [138]. It would be interesting to test whether a simple reduction of the diffusion tensor at high densities in CG simulations is enough to capture all effects observed in all-atom simulations. Because rotational diffusion is used in the theoretical description of many experiments it would also be possible to use accurate estimates of the rotational diffusion from MD simulations to improve experimental results. Because our algorithm works with MD simulations and does not not rely on neat solvent behavior it can be used to study macromolecules in complex liquids, like dense protein solutions of a single protein type or a mixture similar to the environment in the cell. In combination, our results enable us to compare MD simulations to a broader range of experiments and to investigate the diffusion behavior of protein solutions with complex solvents.

In Chapter 4 we have developed a novel BD algorithm that is based on the Monte Carlo method. It builds upon the known similarity of Monte Carlo simulation to Brownian dynamics for isotropic particles. We generalized the derivation of the correct step-size to link Monte Carlo to a time-step and diffusion coefficient to arbitrary acceptance and trial function. We developed a novel acceptance function to reduce the error in dynamical observables using a BDMC algorithm. We could show that our algorithm can be extended to problems with position-dependent diffusion coefficients, a phenomenon that often occurs in the analysis of protein dynamics in reduced coordinates. Lastly, we included rotational moves for arbitrarily shaped macromolecules. In combination these extensions mean that the BDMC algorithm can be applied for many different BD problems. Because the BDMC algorithm only evaluates potentials it can yield better performance and be applied to

problems where force calculation is computationally expensive. For our one-dimensional model systems problems the BAOAB integrator showed exceptional accuracy even at large time-steps. The BAOAB integrator is a simple modification of the Euler-Maruyama integrator that could also be applied to the Ermak-McCammon integrator. One should test if such a modification would make traditional BD simulations stable with larger time-steps. So far we have tested our algorithm for one dimensional model systems or two particle systems. It would be interesting to see how well the algorithm works for many particle systems. We assume that it will achieve better performance for high density protein solutions than the traditional Ermak-McCammon integrator [49]. Our results are further evidence that the Monte Carlo method can be used to determine dynamic properties in addition to equilibrium properties. The framework we developed to correlate diffusion with the trial step size can be used for different system and acceptance functions to extend existing Monte Carlo algorithms and engines to include dynamic calculations.

In Chapter 5 we developed Complexes++, a new Monte-Carlo simulation engine. Complexes++ implements the KH forcefield, a hierarchical CG protein model, and the new BDMC algorithms and several enhanced sampling routines. Great care was taken to ensure that the engine is optimized to run efficiently on laptops and high-performance cluster systems. The program allows great flexibility in setting up different simulations and is at the same time easy to use. To achieve both goals two programs have been developed in the end. The general purpose Monte Carlo engine Complexes++ and the Python program pycomplexes. The later is targeted to be easy to use and sets sensible defaults for a simulation. The addition of new interaction potentials and connection potentials will allow to tackle new scientific questions and perform large-scale multi complex simulation [130, 171, 238]. Because of the good scaling of complexes it would be interesting to study how well the KH model describes dense protein solution that appear in cells. For the BD simulations it is known that the diffusion coefficients have to be scaled with the density [132]. However, most existing theories are based on non-sticky polymer model. An interesting addition to the code would be a new enhanced sampling algorithms like BioEN [82]. The KH model has been extensively used and extended in the literature and we are sure that the new code will allow the scientific community to make use of the model and for appropriating it to develop new simulation protocols.

Bibliography

[1] https://en.wikipedia.org/wiki/AOS_and_SOA.

[2] S. a Beccara, T. Škrbić, R. Covino, C. Micheletti, and P. Faccioli. Folding Pathways of a Knotted Protein with a Realistic Atomistic Force Field. *PLoS Comput. Biol.*, 9(3), 2013. ISSN 1553734X. doi: 10.1371/journal.pcbi.1003002.

[3] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl. GROMACS : High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *Softw. X*, 2:19–25, 2015. doi: 10.1016/j.softx.2015.06.001.

[4] L. af Klinteberg. Ewald summation for the rotlet singularity of Stokes flow. *arXiv:1603.07467 [physics.flu-dyn]*, 2016.

[5] R. Alsallaq and H. X. Zhou. Electrostatic rate enhancement and transient complex of protein-protein association. *Proteins Struct. Funct. Genet.*, 71(1):320–335, 2008. ISSN 08873585. doi: 10.1002/prot.21679.

[6] H. C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *J. Chem. Phys.*, 72:2384, 1980.

[7] H. C. Andersen. Rattle: A "velocity" version of the shake algorithm for molecular dynamics calculations. *J. Comput. Phys.*, 52(1):24–34, 1983.

[8] I. Antes. DynaDock: A new molecular dynamics-based algorithm for protein-peptide docking including receptor flexibility. *Proteins Struct. Funct. Bioinforma.*, 78(5): 1084–1104, 2009. doi: 10.1002/prot.22629.

[9] J. Balbo, P. Mereghetti, D.-P. Herten, and R. C. Wade. The Shape of Protein Crowders is a Major Determinant of Protein Diffusion. *Biophys. J.*, 104(7):1576–1584, 2013. ISSN 00063495. doi: 10.1016/j.bpj.2013.02.041.

[10] A. A. Barker. Monte Carlo Calculations of the Radial Distribution Functions for a Proton-Electron Plasma. *Aust. J. Phys.*, 18(2):119–133, 1965.

[11] A. Baumlova, D. Chalupska, B. Rozycki, M. Jovic, E. Wisniewski, M. Klima, A. Dubankova, D. P. Kloer, R. Nencka, T. Balla, and E. Boura. The crystal structure of the phosphatidylinositol 4-kinase II. *EMBO Rep.*, 15(10):1085–1092, 2014. ISSN 1469-221X. doi: 10.15252/embr.201438841.

[12] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Comput. Sci. Eng.*, 13:31–39, 2011. ISSN 15219615. doi: 10.1109/MCSE.2010.118.

[13] I. S. Beloborodov, V. Y. Orekhov, and A. S. Arseniev. Effect of Coupling between Rotational and Translational Brownian Motions on NMR Spin Relaxation: Consideration Using Green Function of Rigid Body Diffusion. *J. Magn. Reson.*, 132(2): 328–329, 1998. ISSN 1090-7807. doi: http://dx.doi.org/10.1006/jmre.1998.1395.

[14] O. Ben-Kiki, C. Evans, and I. döt Net. http://yaml.org/spec/1.2/spec.html, 2018.

[15] C. H. Bennett. Efficient estimation of free energy differences from Monte Carlo data. *J. Comput. Phys.*, 22(2):245–268, 1976.

[16] H. J. Berendsen, J. P. Postma, W. F. Van Gunsteren, A. Dinola, and J. R. Haak. Molecular dynamics with coupling to an external bath. *J. Chem. Phys.*, (81):3684, 1984.

[17] R. B. Best and G. Hummer. Optimized Molecular Dynamics Force Fields Applied to the Helix-Coil Transition of Polypeptides - The Journal of Physical Chemistry B (ACS Publications). *J Phys Chem B*, 113:9004–9015, 2009. ISSN 1520-6106. doi: 10.1021/jp901540t.

[18] R. B. Best and G. Hummer. Coordinate-dependent diffusion in protein folding. *Proc. Natl. Acad. Sci.*, 107(3):1088–1093, 2010. ISSN 0027-8424. doi: 10.1073/pnas.0910390107.

[19] R. B. Best, Y. G. Chen, and G. Hummer. Slow protein conformational dynamics from multiple experimental structures: The helix/sheet transition of Arc repressor. *Structure*, 13:1755–1763, 2005. ISSN 09692126. doi: 10.1016/j.str.2005.08.009.

[20] R. B. Best, X. Zhu, J. Shim, P. E. Lopes, J. Mittal, M. Feig, and A. D. MacKerell. Optimization of the additive CHARMM all-atom protein force field targeting improved sampling of the backbone ϕ, ψ and side-chain χ1and χ2Dihedral Angles. *J. Chem. Theory Comput.*, 8(9):3257–3273, 2012. ISSN 15499626. doi: 10.1021/ct300400x.

[21] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Rev.*, 59(1):65–98, 2017. ISSN 0036-1445. doi: 10.1137/141000671.

[22] O. A. R. Board. OpenMP Application Program Interface Version 3.0, 2013. URL `https://www.openmp.org/wp-content/uploads/OpenMP4.0.0.pdf`.

[23] J. Boisbouvier, Z. Wu, A. Ono, M. Kainosho, and A. Bax. Rotational diffusion tensor of nucleic acids from 13C NMR relaxation. *J. Biomol. NMR*, 27:133–142, 2003. ISSN 0925-2738. doi: 10.1023/A:1024931619957.

[24] A. Borgia, W. Zheng, K. Buholzer, M. B. Borgia, A. Schüler, H. Hofmann, A. Soranno, D. Nettels, K. Gast, A. Grishaev, R. B. Best, and B. Schuler. Consistent View of Polypeptide Chain Expansion in Chemical Denaturants from Multiple Experimental Methods. *J. Am. Chem. Soc.*, 138(36):11714–11726, 2016. ISSN 15205126. doi: 10.1021/jacs.6b05917.

[25] B. Bramas. Inastemp : A Novel Intrinsics-as-Template Library for Portable SIMD-Vectorization. 2017:1–22, 2017. ISSN 10589244. doi: 10.1155/2017/5482468.

[26] R. Brüschweiler, L. Xiubei, and P. E. Wright. Long-range motional restrictions in a multidomain zinc-finger protein from anisotropic tumbling. *Science (80-.).*, 268 (5212):886–889, 1995.

[27] N.-V. Buchete and G. Hummer. Coarse Master Equations for Peptide Folding Dynamics. *J. Phys. Chem. B*, 112(19):6057–6069, 2008. ISSN 1520-6106. doi: 10.1021/jp0761665.

[28] G. Bussi. Hamiltonian replica exchange in GROMACS: a flexible implementation. *Mol. Phys.*, 112(3-4):379–384, 2014. ISSN 0026-8976. doi: 10.1080/00268976.2013.824126.

[29] G. Bussi, D. Donadio, and M. Parrinello. Canonical sampling through velocity rescaling. *J. Chem. Phys.*, 126(1):014101, 2007. ISSN 0021-9606. doi: 10.1063/1.2408420.

[30] D. Chalupska, A. Eisenreichova, B. Różycki, L. Rezabkova, J. Humpolickova, M. Klima, and E. Boura. Structural analysis of phosphatidylinositol 4-kinase IIIβ (PI4KB) – 14-3-3 protein complex reveals internal flexibility and explains 14-3-3 mediated protection from degradation in vitro. *J. Struct. Biol.*, 200(1):36–44, 2017. ISSN 10958657. doi: 10.1016/j.jsb.2017.08.006.

[31] P.-c. Chen, M. Hologne, and O. Walker. Computing the Rotational Diffusion of Biomolecules via Molecular Dynamics Simulation and Quaternion Orientations. *J. Phys. Chem. B*, 121(8):1812–1823, 2017. ISSN 1520-6106. doi: 10.1021/acs.jpcb.6b11703.

[32] G. Chevrot, K. Hinsen, and G. R. Kneller. Model-free simulation approach to molecular diffusion tensors. *J. Chem. Phys.*, 139(15):154110, 2013. ISSN 1089-7690. doi: 10.1063/1.4823996.

[33] K. Chu, J. Vojtchovský, B. H. McMahon, R. M. Sweet, J. Berendzen, and I. Schlichting. Structure of a ligand-binding intermediate in wild-type carbonmonoxy myoglobin. *Nature*, 403(6772):921–923, 2000. ISSN 0028-0836. doi: 10.1038/35002641.

[34] T. H. Click, A. Liu, and G. A. Kaminski. Quality of Random Number Generators Significantly Affects Results of Monte Carlo Simulations for Organic and Biological Systems. *J. Comput. Chem.*, 32:513–524, 2011. ISSN 1096-987X. doi: 10.1002/jcc.

[35] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *J. Am. Chem. Soc.*, 117(19):5179–5197, 1995. ISSN 0002-7863. doi: 10.1021/ja00124a002.

[36] R. Covino. *Investigating Protein Folding Pathways at Atomistic Resolution : from a Small Domain to a Knotted Protein.* Phd, UNIVERSITÀ DEGLI STUDI DI TRENTO, 2013.

[37] R. Covino, T. Skrbić, S. A. Beccara, P. Faccioli, and C. Micheletti. The role of non-native interactions in the folding of knotted proteins: insights from molecular dynamics simulations. *Biomolecules*, 4(1):1–19, 2014. ISSN 2218273X. doi: 10.3390/biom4010001.

[38] T. Darden, D. York, and L. Pedersen. Particle mesh Ewald: An N·log(N) method for Ewald sums in large systems. *J. Chem. Phys.*, 98(12):10089 – 10092, 1993. ISSN 00264806. doi: 10.1063/1.464397.

[39] M. E. Davis, J. D. Madura, B. A. Luty, J. A. McCammon, B. A. Lutty, and J. A. McCammon. Electrostatics and diffusion of molecules in solution: simulations with the University of Houston Brownian dynamics program. *Comput. Phys. Commun.*, 62(1-3):187–197, 1991. ISSN 0010-4655. doi: JCM.02209-06[pii]\r10.1128/JCM.02209-06.

[40] U. K. Deiters. Efficient coding of the minimum image convention. *Zeitschrift fur Phys. Chemie*, 227(2-3):345–352, 2013. ISSN 09429352. doi: 10.1524/zpch.2013.0311.

[41] R. Diamond. Real-space refinement of the structure of hen egg-white lysozyme. *J. Mol. Biol.*, 82(3):371–391, 1974.

[42] G. L. Dignon, W. Zheng, Y. C. Kim, R. B. Best, and J. Mittal. Sequence determinants of protein phase behavior from a coarse-grained model. *PLoS Comput. Biol.*, 14(1):1–23, 2018. ISSN 15537358. doi: 10.1371/journal.pcbi.1005941.

[43] M. Długosz and J. M. Antosiewicz. Evaluation of Proteins' Rotational Diffusion Coefficients from Simulations of Their Free Brownian Motion in Volume-Occupied Environments. *J. Chem. Theory Comput.*, 10:481–491, 2014. ISSN 1549-9618. doi: 10.1021/ct4008519.

[44] D. L. Dotson, S. L. Seyler, M. Linke, R. J. Gowers, and O. Beckstein. datreant: persistent, Pythonic trees for heterogeneous data. In *Proc. 15th Python Sci. Conf.*, pages 51–56, 2016. ISBN 5737309321776.

[45] E. Duchardt, L. Nilsson, and J. Schleucher. Cytosine ribose flexibility in DNA: A combined NMR 13C spin relaxation and molecular dynamics simulation study. *Nucleic Acids Res.*, 36(12):4211–4219, 2008. ISSN 03051048. doi: 10.1093/nar/gkn375.

[46] B. Dünweg and K. Kremer. Molecular dynamics simulation of a polymer chain in solution. *J. Chem. Phys.*, 99(9):6983, 1993. ISSN 00219606. doi: 10.1063/1.465445.

[47] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L. P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, and V. S. Pande. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS Comput. Biol.*, 13(7):1–17, 2017. ISSN 15537358. doi: 10.1371/journal.pcbi.1005659.

[48] A. Einstein. Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen. *Ann. Phys.*, 322 (8):549–560, 1905.

[49] D. L. Ermak and J. a. McCammon. Brownian dynamics with hydrodynamic interactions. *J. Chem. Phys.*, 69(4):1352–1360, 1978.

[50] P. P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Ann. Phys.*, 369(3):253—-287, 1921.

[51] L. D. Favro. Theory of the Rotational Brownian Motion of a Free Rigid Body. *Phys. Rev.*, 119(1):53–62, 1960. ISSN 0038-5697. doi: 10.1007/BF00815660.

[52] M. Fortoul, M. Bykhovskaia, and A. Jagota. Coarse-Grained Model of the SNARE Complex Shows that Quick Zippering Requires Partial Assembly. *bioRxiv*, 2018. doi: 10.1101/294181.

[53] N. Fortoul, P. Singh, C. Y. Hui, M. Bykhovskaia, and A. Jagota. Coarse-grained model of SNARE-mediated docking. *Biophys. J.*, 108(9):2258–2269, 2015. ISSN 15420086. doi: 10.1016/j.bpj.2015.03.053.

[54] P. S. Foundation. Python. URL `https://www.python.org/`.

[55] D. Frenkel and B. Smit. *Understanding molecular simulation: from algorithms to applications*, volume 1. Academic press, 2001.

[56] G. Fuertes, N. Banterle, K. M. Ruff, A. Chowdhury, D. Mercadante, C. Koehler, M. Kachala, G. Estrada Girona, S. Milles, A. Mishra, P. R. Onck, F. Gräter, S. Esteban-Martín, R. V. Pappu, D. I. Svergun, and E. A. Lemke. Decoupling of size and shape fluctuations in heteropolymeric sequences reconciles discrepancies in SAXS vs. FRET measurements. *Proc. Natl. Acad. Sci.*, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1704692114.

[57] W. H. Furry. Isotropic Rotational Brownian Motion. *Phys. Rev.*, 107(1), 1957.

[58] M. Fushiki. System size dependence of the diffusion coefficient in a simple liquid. *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.*, 68(2):6, 2003. ISSN 1063651X. doi: 10.1103/PhysRevE.68.021203.

[59] R. R. Gabdoulline and R. C. Wade. Simulation of the diffusional association of barnase and barstar. *Biophys. J.*, 72(5):1917–1929, 1997. ISSN 00063495. doi: 10.1016/S0006-3495(97)78838-6.

[60] J. Gans and D. Shalloway. Shadow mass and the relationship between velocity and momentum in symplectic numerical integration. *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.*, 61:4587, 2000. ISSN 1063651X. doi: 10.1103/PhysRevE.61.4587.

[61] R. Ghose, D. Fushman, and D. Cowburn. Determination of the rotational diffusion tensor of macromolecules in solution from nmr relaxation data with a combination of exact and approximate methods–application to the determination of interdomain orientation in multidomain proteins. *J. Magn. Reson.*, 149(2):204–17, 2001. ISSN 1090-7807. doi: 10.1006/jmre.2001.2295.

[62] D. A. Giammona. *An Examination of Conformational Flexibility in Porphyrins and Bulky-Ligand Binding in Myoglobin.* Ph.d. thesis, Univ. of California, Davis, 1984.

[63] R. J. Glauber. Time-Dependent Statistics of the Ising Model. *J. Math. Phys.*, 4(2):294, 1963. doi: 10.1063/1.1703954.

[64] H. Goldstein, C. Poole, and J. Safko. *Classical Mechanics.* 2007. ISBN 0201657023.

[65] R. J. Gowers, M. Linke, J. Barnoud, T. J. E. Reddy, M. N. Melo, S. L. Seyler, J. Domański, D. L. Dotson, S. Buchoux, I. M. Kenney, and O. Beckstein. MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. In *Proc. 15th Python Sci. Conf.*, pages 98–105, 2016.

[66] U. H. E. Hansmann. Parallel tempering algorithm for conformational studies of biological molecules. *Chem. Phys. Lett.*, 281(1-3):140–150, 1997. ISSN 00092614. doi: 10.1016/S0009-2614(97)01198-6.

[67] J. Happel and H. Brenner. *Low Reynolds Number Hydrodynamics.* 1965.

[68] H. Hasimoto. On the periodic fundamental solutions of the Stokes equations and their application to viscous flow past a cubic array of spheres. *J. Fluid Mech.*, 5(2):317–328, 1959. doi: 10.1017/S0022112059000222.

[69] W. K. Hastings. Biometrika Trust Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970.

[70] B. Hess. Determining the Shear Viscosity of Model Liquids From Molecular Dynamics Simulations. *J. Chem. Phys.*, 116(1):209–217, 2002. ISSN 00219606. doi: 10.1063/1.1421362.

[71] B. Hess. P-LINCS: A parallel linear constraint solver for molecular simulation. *J. Chem. Theory Comput.*, 4(1):116–122, 2008. ISSN 15499618. doi: 10.1021/ct700200b.

[72] B. Hess, H. Bekker, H. J. Berendsen, and J. G. E. M. Fraaije. LINCS: a linear constraint solver for molecular simulations. *J. Comput. Chem.*, 18(12):1463–1472, 1997.

[73] R. Hockney and J. Eastwood. Computer simulations using particles. *New York*, 1981.

[74] F. Hoffmann, M. Xue, L. V. Schäfer, and F. A. A. Mulder. Narrowing the gap between experimental and computational determination of methyl group dynamics in proteins. *Phys. Chem. Chem. Phys.*, pages 24577–24590, 2018. ISSN 1463-9076. doi: 10.1039/C8CP03915A.

[75] J. M. Hofman, H. J. H. Clercx, and P. P. Schram. Hydrodynamic interactions in colloidal crystals (II). Application to dense cubic and tetragonal arrays. *Phys. A Stat. Mech. its Appl.*, 268(3-4):353–390, 1999. ISSN 03784371. doi: 10.1016/S0378-4371(99)00053-9.

[76] W. G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Phys. Rev. A*, 31(3):1695–1697, 1985. ISSN 10502947. doi: 10.1103/PhysRevA.31.1695.

[77] B. G. Horan, D. Vavylonis, and Y. C. Kim. Computational modeling highlights disordered Formin Homology 1 domain's role in profilin-actin transfer. *bioRxiv*, 2018. doi: 10.1101/263566.

[78] V. Hornak, R. Abel, A. Okur, B. Strockbine, A. Roitberg, and C. Simmerling. Comparison of Multiple Amber Force Fields and Development of Improved Protein Backbone Parameters. *Proteins*, 65:712–725, 2006.

[79] G. A. Huber and J. A. McCammon. Browndye: A software package for Brownian dynamics. *Comput. Phys. Commun.*, 181(11):1896–1905, 2010. ISSN 00104655. doi: 10.1016/j.cpc.2010.07.022.

[80] G. Hummer. From transition paths to transition states and rate coefficients. *J. Chem. Phys.*, 120(2):516–523, jan 2004. ISSN 0021-9606. doi: 10.1063/1.1630572.

[81] G. Hummer. Position-dependent diffusion coefficients and free energies from Bayesian analysis of equilibrium and replica molecular dynamics simulations. *New J. Phys.*, 7(34):0–14, 2005. ISSN 13672630. doi: 10.1088/1367-2630/7/1/034.

[82] G. Hummer and J. Köfinger. Bayesian ensemble refinement by replica simulations and reweighting. *J. Chem. Phys.*, 143(24), 2015. ISSN 00219606. doi: 10.1063/1.4937786.

[83] G. Hummer and A. Szabo. Dynamics of the Orientational Factor in Fluorescence Resonance Energy Transfer. *J. Phys. Chem. B*, 121(15):3331–3339, 2017. ISSN 15205207. doi: 10.1021/acs.jpcb.6b08345.

[84] J. D. Hunter. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55.

[85] C. Hyeon, G. Morrison, and D. Thirumalai. Force-dependent hopping rates of RNA hairpins can be estimated from accurate measurement of the folding landscapes. *Proc. Natl. Acad. Sci.*, 105(28):9604–9609, 2008. ISSN 0027-8424. doi: 10.1073/pnas.0802484105.

[86] I. M. Ilie, W. K. den Otter, and W. J. Briels. Rotational Brownian Dynamics simulations of clathrin cage formation. *J. Chem. Phys.*, 141(6):065101, 2014. ISSN 0021-9606. doi: 10.1063/1.4891306.

[87] I. Ivani, P. D. Dans, A. Noy, A. Pérez, I. Faustino, A. Hospital, J. Walther, P. Andrio, R. Goñi, A. Balaceanu, G. Portella, F. Battistini, J. L. Gelpí, C. González, M. Vendruscolo, C. A. Laughton, S. A. Harris, D. A. Case, and M. Orozco. Parmbsc1: a refined force field for DNA simulations. *Nat. Methods*, 13(1):55–58, 2016. ISSN 1548-7105. doi: 10.1038/nmeth.3658.

[88] E. Jones, T. Oliphant, P. Peterson, and Others. SciPy: Open Source Scientific Tools for Python, 2001. URL `http://www.scipy.org/`.

[89] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein. Comparison of simple potential functions for simulating liquid water. *J. Chem. Phys.*, 79:926–935, 1983.

[90] A. Jost-Lopez. *Coarse-Grained Molecular Simulations of Weak Protein Complexes.* Master thesis, 2014.

[91] I. S. Joung and T. E. Cheatham III. Determination of alkali and halide monovalent ion parameters for use in explicitly solvated biomolecular simulations. *J. Phys. Chem. B*, 112:9020–9041, 2008. ISSN 15206106. doi: 10.1021/jp8001614.

[92] I. S. Joung and T. E. Cheatham III. Halide Ions Using Water-Model-Specific Ion Parameters. *J. Phys. Chem. B*, 113(40):13279–13290, 2009.

[93] R. S. Kang, C. M. Daniels, S. A. Francis, S. C. Shih, W. J. Salerno, L. Hicke, and I. Radhakrishnan. Solution structure of a CUE-ubiquitin complex reveals a conserved mode of ubiquitin binding. *Cell*, 113(5):621–630, 2003. ISSN 00928674. doi: 10.1016/S0092-8674(03)00362-3.

[94] J. Karanicolas and C. L. Brooks. The origins of asymmetry in the folding transition states of protein L and protein G. *Protein Sci.*, 11(10):2351–2361, 2002. ISSN 0961-8368. doi: 10.1110/ps.0205402.

[95] K. Kikuchi, M. Yoshida, T. Maekawa, and H. Watanabe. Metropolis Monte Carlo method as a numerical technique to solve the Fokker—Planck equation. *Chem. Phys. Lett.*, 185(3):335–338, 1991.

[96] Y. C. Kim and G. Hummer. Coarse-grained Models for Simulations of Multiprotein Complexes: Application to Ubiquitin Binding. *J. Mol. Biol.*, 375(5):1416–1433, 2008. ISSN 00222836. doi: 10.1016/j.jmb.2007.11.063.

[97] Y. C. Kim and J. Mittal. Crowding induced entropy-enthalpy compensation in protein association equilibria. *Phys. Rev. Lett.*, 110(20):1–5, 2013. ISSN 00319007. doi: 10.1103/PhysRevLett.110.208102.

[98] Y. C. Kim, R. B. Best, and J. Mittal. Macromolecular crowding effects on protein-protein binding affinity and specificity. *J. Chem. Phys.*, 133(20), 2010. ISSN 00219606. doi: 10.1063/1.3516589.

[99] H. C. Klein and U. S. Schwarz. Studying protein assembly with reversible Brownian dynamics of patchy particles. *J. Chem. Phys.*, 140(18):184112, 2014. ISSN 00219606. doi: 10.1063/1.4873708.

[100] J. Köfinger, M. J. Ragusa, I.-H. Lee, G. Hummer, and J. H. Hurley. Solution Structure of the Atg1 Complex: Implications for the Architecture of the Phagophore Assembly Site. *Structure*, 23(5):809–818, 2015. ISSN 09692126. doi: 10.1016/j.str.2015.02.012.

[101] K. J. Kohlhoff, D. Shukla, M. Lawrenz, G. R. Bowman, D. E. Konerding, D. Belov, R. B. Altman, and V. S. Pande. Cloud-based simulations on Google Exacycle reveal ligand modulation of GPCR activation pathways. *Nat. Chem.*, 6(1):15–21, 2014. ISSN 1755-4349. doi: 10.1038/nchem.1821.

[102] J. E. Kohn, I. S. Millett, J. Jacob, B. Zagrovic, T. M. Dillon, N. Cingel, R. S. Dothager, S. Seifert, P. Thiyagarajan, T. R. Sosnick, M. Z. Hasan, V. S. Pande, I. Ruczinski, S. Doniach, and K. W. Plaxco. Random-coil behavior and the dimensions of chemically unfolded proteins. *Proc. Natl. Acad. Sci. U. S. A.*, 101(34): 12491–6, 2004. ISSN 0027-8424. doi: 10.1073/pnas.0403643101.

[103] D. M. Korzhnev, M. Billeter, A. S. Arseniev, and V. Y. Orekhov. NMR studies of Brownian tumbling and internal motions in proteins. *Prog. Nucl. Magn. Reson. Spectrosc.*, 38(3):197–266, 2001. ISSN 00796565. doi: 10.1016/S0079-6565(00)00028-5.

[104] M. J. Kotelyanskii and U. W. Suter. A dynamic Monte Carlo method suitable for molecular simulations. *J. Chem. Phys.*, 96(7):5383, 1992. ISSN 00219606. doi: 10.1063/1.462723.

[105] M. W. Kutta. Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. *Zeitschrift für Math. und Phys.*, 46:435–453, 1901.

[106] A. Laio and M. Parrinello. Escaping free-energy minima. *Proc. Natl. Acad. Sci.*, 99 (20):12562–12566, 2002. ISSN 1091-6490. doi: 10.1073/pnas.202427399.

[107] J. R. Lakowicz. *Principles of Flourescence Spectroscopy*. Springer Science & Business Media, New York, second edition, 1999. ISBN 9781475730630.

[108] S. K. Lam, A. Pitrou, and S. Seibert. Numba: A LLVM-based python JIT compiler. *Proc. Second Work. LLVM Compil. Infrastruct. HPC - LLVM '15*, 2015. doi: 10.1145/2833157.2833162.

[109] A. L. Lee and A. J. Wand. Assessing potential bias in the determination of rotational correlation times of proteins by NMR relaxation. *J. Biomol. NMR*, 13(2):101–112, 1999. ISSN 09252738. doi: 10.1023/A:1008304220445.

[110] L. K. Lee, M. Rance, W. J. Chazin, and A. G. Palmer III. Rotational diffusion anisotropy of proteins from simultaneous analysis of 15 N and 13 C α nuclear spin relaxation. *J. Biomol. NMR*, 9(3):287–298, 1997.

[111] B. Leimkuhler and C. Matthews. Rational construction of stochastic numerical methods for molecular sampling. *Appl. Math. Res. eXpress*, 2013(1):34–56, 2012. ISSN 16871200. doi: 10.1093/amrx/abs010.

[112] D. S. Lemons and A. Gythiel. Paul Langevin's 1908 paper "On the Theory of Brownian Motion" ["Sur la théorie du mouvement brownien," C. R. Acad. Sci. (Paris) 146 , 530–533 (1908)]. *Am. J. Phys.*, 65(11):1079–1081, nov 1997. ISSN 0002-9505. doi: 10.1119/1.18725.

[113] K. Lindorff-Larsen, S. Piana, K. Palmo, P. Maragakis, J. L. Klepeis, R. O. Dror, and D. E. Shaw. Improved side-chain torsion potentials for the Amber ff99SB protein force field. *Proteins Struct. Funct. Bioinforma.*, 78(8):1950–1958, 2010. ISSN 08873585. doi: 10.1002/prot.22711.

[114] M. Linke and G. Hummer. pydiffusion, 2018. URL `https://github.com/bio-phys/pydiffusion`.

[115] M. Linke, J. Köfinger, and G. Hummer. Rotational Diffusion Depends on Box Size in Molecular Dynamics Simulations. *J. Phys. Chem. Lett.*, 9(11):2874–2878, 2018. ISSN 1948-7185. doi: 10.1021/acs.jpclett.8b01090.

[116] G. Lipari and A. Szabo. Model-Free Approach to the Interpretation of Nuclear Magnetic Resonance Relaxation in Macromolecules. 1. Theory and Range of Validity. *J. Am. Chem. Soc.*, 104(17):4546–4559, 1982.

[117] P. Liu, D. K. Agrafiotis, and D. L. Theobald. Fast Determination of the Optimal Rotational Matrix for Macromolecular Superpositions. *J. Comput. Chem.*, 31(7): 1561–1563, 2010. ISSN 1096-987X. doi: 10.1002/jcc.

[118] A. Loman, I. Gregor, C. Stutz, M. Mund, and J. Enderlein. Measuring rotational diffusion of macromolecules by fluorescence correlation spectroscopy. *Photoch. Photobio. Sci.*, 9(5):627–636, 2010. ISSN 1474-905X. doi: 10.1039/B9PP00029A.

[119] R. M. Lynden-Bell and A. J. Stone. Reorientational correlation functions, quaternions and wigner rotation matrices. *Mol. Simul.*, 3:271–281, 1989. ISSN 10290435. doi: 10.1080/08927028908031380.

[120] D. Maciej and J. M. Antosiewicz. Anisotropic Diffusion Effects on the Barnase−Barstar Encounter Kinetics. *J. Chem. Theory Comput.*, 9:1667–1677, 2013.

[121] A. D. MacKerell, D. Bashford, M. Bellott, R. L. Dunbrack, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiórkiewicz-Kuczera, D. Yin, and M. Karplus. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. *J. Phys. Chem. B*, 102(18): 3586–3616, 1998. ISSN 1520-6106. doi: 10.1021/jp973084f.

[122] A. D. Mackerell, M. Feig, and C. L. Brooks. Extending the Treatment of Backbone Energetics in Protein Force Fields: Limitations of Gas-Phase Quantum Mechanics in Reproducing Protein Conformational Distributions in Molecular Dynamics Simulation. *J. Comput. Chem.*, 25(11):1400–1415, 2004. ISSN 01928651. doi: 10.1002/jcc.20065.

[123] J. D. Madura, J. M. Briggs, R. C. Wade, M. E. Davis, B. A. Luty, A. Ilin, J. Antosiewicz, M. K. Gilson, B. Bagheri, L. R. Scott, and J. A. McCammon. Electrostatics and diffusion of molecules in solution: simulations with the University of Houston Brownian Dynamics program. *Comput. Phys. Commun.*, 91(1-3):57–95, 1995. ISSN 00104655. doi: 10.1016/0010-4655(95)00043-F.

[124] D. Malinverni, A. J. Lopez, P. De Los Rios, G. Hummer, and A. Barducci. Modeling Hsp70/Hsp40 interaction by multi-scale molecular simulations and coevolutionary sequence analysis. *Elife*, 6:1–20, 2017. ISSN 2050084X. doi: 10.7554/eLife.23471.

[125] Y. Mao and Y. Zhang. Thermal conductivity, shear viscosity and specific heat of rigid water models. *Chem. Phys. Lett.*, 542:37–41, 2012. ISSN 00092614. doi: 10.1016/j.cplett.2012.05.044.

[126] S. J. Marrink, A. H. de Vries, and A. E. Mark. Coarse Grained Model for Semiquantitative Lipid Simulations. *J. Phys. Chem. B*, 108(2):750–760, 2004. ISSN 1520-6106. doi: 10.1021/jp036508g.

[127] S. J. Marrink, H. J. Risselada, S. Yefimov, D. P. Tieleman, and A. H. De Vries. The MARTINI force field: Coarse grained model for biomolecular simulations. *J. Phys. Chem. B*, 111(27):7812–7824, 2007. ISSN 15206106. doi: 10.1021/jp071097f.

[128] G. Maruyama. Continuous Markov processes and stochastic equations. *Rend. del Circ. Mat. di Palermo*, 4(1):48–90, 1955.

[129] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8:3–30, 1998.

[130] S. R. McGuffee and A. H. Elcock. Diffusion, crowding & protein stability in a dynamic molecular model of the bacterial cytoplasm. *PLoS Comput. Biol.*, 6(3): e1000694, 2010. ISSN 1553734X. doi: 10.1371/journal.pcbi.1000694.

[131] W. McKinney. Data Structures for Statistical Computing in Python. *Proc. 9th Python Sci. Conf.*, 1697900(Scipy):51–56, 2010.

[132] P. Mereghetti and R. C. Wade. Atomic detail brownian dynamics simulations of concentrated protein solutions with a mean field treatment of hydrodynamic interactions. *J. Phys. Chem. B*, 116(29):8523–8533, 2012. ISSN 15205207. doi: 10.1021/jp212532h.

[133] P. Mereghetti, R. R. Gabdoulline, and R. C. Wade. Brownian dynamics simulation of protein solutions: Structural and dynamical properties. *Biophys. J.*, 99(11):3782–3791, 2010. ISSN 00063495. doi: 10.1016/j.bpj.2010.10.035.

[134] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, 21 (6):1087–1092, 1953. doi: doi:10.1063/1.1699114.

[135] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein. Software News and Updates MDAnalysis : A Toolkit for the Analysis of Molecular Dynamics Simulations. *J. Comput. Chem.*, 32(10):2319–2327, 2011. doi: 10.1002/jcc.

[136] S. Miyazawa and R. L. Jernigan. Residue – Residue Potentials with a Favorable Contact Pair Term and an Unfavorable High Packing Density Term, for Simulation and Threading. *J. Mol. Biol.*, 256(3):623–644, 1996. ISSN 00222836. doi: 10.1006/jmbi.1996.0114.

[137] M. J. Morelli and P. R. Ten Wolde. Reaction Brownian dynamics and the effect of spatial fluctuations on the gain of a push-pull network. *J. Chem. Phys.*, 129(5): 054112, 2008. ISSN 00219606. doi: 10.1063/1.2958287.

[138] G. Nawrocki, A. Karaboga, Y. Sugita, and M. Feig. Effect of Protein-Protein Interactions and Solvent Viscosity on the Rotational Diffusion of Proteins in

Crowded Environments. *Phys. Chem. Chem. Phys.*, 2018. ISSN 1463-9076. doi: 10.1039/C8CP06142D.

[139] S. H. Northrup and H. P. Erickson. Kinetics of protein-protein association explained by Brownian dynamics computer simulation. *Proc. Natl. Acad. Sci. U. S. A.*, 89(8): 3338–3342, 1992. ISSN 0027-8424. doi: 10.1073/pnas.89.8.3338.

[140] S. H. Northrup, S. A. Allison, and J. A. McCammon. Brownian dynamics simulation of diffusion-influenced bimolecular reactions. *J. Chem. Phys.*, 80(4):1517–1524, 1984.

[141] S. H. Northrup, J. O. Boles, and J. C. Reynolds. Brownian dynamics of cytochrome c and cytochrome c peroxidase association. *Science (80-.).*, 241(4861):67–70, 1988. ISSN 00368075. doi: 10.1126/science.2838904.

[142] S. Nosé. A unified formulation of the constant temperature molecular dynamics methods. *J. Chem. Phys.*, 81(1):511–519, 1984.

[143] E. P. O'Brien, G. Morrison, B. R. Brooks, and D. Thirumalai. How accurate are polymer models in the analysis of Förster resonance energy transfer experiments on proteins? *J. Chem. Phys.*, 130(12), 2009. ISSN 00219606. doi: 10.1063/1.3082151.

[144] T. Okabe, M. Kawata, Y. Okamoto, and M. Mikami. Replica-exchange Monte Carlo method for the isobaric-isothermal ensemble. *Chem. Phys. Lett.*, 335(5-6):435–439, 2001. ISSN 00092614. doi: 10.1016/S0009-2614(01)00055-0.

[145] K. I. Okazaki, T. Sato, and M. Takano. Temperature-enhanced association of proteins due to electrostatic interaction: A coarse-grained simulation of actin-myosin binding. *J. Am. Chem. Soc.*, 134(21):8918–8925, 2012. ISSN 00027863. doi: 10.1021/ja301447j.

[146] T. E. Oliphant. *A guide to NumPy*. Trelgol Publishing USA, 2006.

[147] A. Ortega, D. Amorós, and J. Garcia De La Torre. Prediction of Hydrodynamic and Other Solution Properties of Rigid Proteins from Atomic- and Residue-Level Models. *Biophys. J.*, 101:892–898, 2011. ISSN 00063495. doi: 10.1016/j.bpj.2011.06.046.

[148] M. E. Pacold, S. Suire, O. Perisic, S. Lara-Gonzalez, C. T. Davis, E. H. Walker, P. T. Hawkins, L. Stephens, J. F. Eccleston, and R. L. Williams. Crystal Structure and Functional Analysis of Ras Binding to Its Effector Phosphoinositide 3-Kinase γ. *Cell*, 103(6):931–944, 2000. ISSN 00928674. doi: 10.1016/S0092-8674(00)00196-3.

[149] A. G. Palmer III, C. D. Kroenke, and J. P. Loria. Nuclear magnetic resonance methods for quantifying microsecond-to-millisecond motions in biological macromolecules. *Methods Enzymol.*, 339(1997):204–238, 2001. ISSN 00766879. doi: 10.1016/S0076-6879(01)39315-1.

[150] A. C. Pan, T. M. Weinreich, S. Piana, and D. E. Shaw. Demonstrating an Order-of-Magnitude Sampling Enhancement in Molecular Dynamics Simulations of Complex Protein Systems. *J. Chem. Theory Comput.*, 12(3):1360–1367, 2016. ISSN 15499626. doi: 10.1021/acs.jctc.5b00913.

[151] M. Parrinello and A. Rahman. Polymorphic transitions in single crystals: A new molecular dynamics method. *J. Appl. Phys.*, 52(12):7182–7190, 1981. ISSN 00218979. doi: 10.1063/1.328693.

[152] Paul Langevin. Sur la théorie du mouvement brownien. *Comptes-rendus l'Académie des Sci.*, 1908. doi: 10.1119/1.18725.

[153] F. Perrin. Mouvement brownien d'un ellipsoide - I. Dispersion diélectrique pour des molécules ellipsoidales. *J. Phys. le Radium*, 5(10):497–511, 1934.

[154] F. Perrin. Mouvement Brownien d'un ellipsoide (II). Rotation libre et dépolarisation des fluorescences. Translation et diffusion de molécules ellipsoidales. *J. Phys. le Radium*, 7(1):1–11, 1936.

[155] J. Perrin. *Brownian Movement and Molecular Reality*. 1909.

[156] J. H. Peters and B. L. De Groot. Ubiquitin dynamics in complexes reveal molecular recognition mechanisms beyond induced fit and conformational selection. *PLoS Comput. Biol.*, 8(10):e1002704, 2012.

[157] R. J. Phillips, J. F. Brady, and G. Bossis. Hydrodynamic transport properties of hard-sphere dispersions. I. Suspensions of freely mobile particles. *Phys. Fluids*, 31 (12):3462, 1988. ISSN 00319171. doi: 10.1063/1.866914.

[158] S. Plimpton. Fast Parallel Algorithms for Short – Range Molecular Dynamics. *J. Comput. Phys.*, 117:1–19, 1995. ISSN 00219991. doi: 10.1006/jcph.1995.1039.

[159] J.-H. Prinz, H. Wu, M. Sarich, B. G. Keller, M. Senne, M. Held, J. D. Chodera, C. Schütte, and F. Noé. Markov models of molecular kinetics: Generation and validation. *J. Chem. Phys.*, 174105(17):174105, 2011. doi: 10.1063/1.3565032.

[160] S. Qin, X. Pang, and H.-X. Zhou. Automated prediction of protein association rate constants. *Structure*, 19(12):1744–1751, 2011. ISSN 09692126. doi: 10.1016/j.str.2011.10.015.

[161] M. J. Ragusa, R. E. Stanley, and J. H. Hurley. Architecture of the Atg17 Complex as a Scaffold for Autophagosome Biogenesis. *Cell*, 151(7):1501–1512, 2012. ISSN 0092-8674. doi: 10.1016/j.cell.2012.11.028.

[162] K. M. Ravikumar, W. Huang, and S. Yang. Coarse-grained simulations of protein-protein association: An energy landscape perspective. *Biophys. J.*, 103(4):837–845, 2012. ISSN 00063495. doi: 10.1016/j.bpj.2012.07.013.

[163] D. R. Roe and T. E. Cheatham. PTRAJ and CPPTRAJ: Software for Processing and Analysis of Molecular Dynamics Trajectory Data. *J. Chem. Theory Comput.*, 9(7):3084–3095, 2013. ISSN 1549-9618. doi: 10.1021/ct400341p.

[164] M. Roos, M. Hofmann, S. Link, M. Ott, J. Balbach, E. Rössler, K. Saalwächter, and A. Krushelnitsky. The "long tail" of the protein tumbling correlation function: observation by 1H NMR relaxometry in a wide frequency and concentration range. *J. Biomol. NMR*, 63:403–415, 2015. ISSN 0925-2738. doi: 10.1007/s10858-015-0001-1.

[165] M. Roos, M. Ott, M. Hofmann, S. Link, E. Rössler, J. Balbach, A. Krushelnitsky, and K. Saalwächter. Coupling and Decoupling of Rotational and Translational Diffusion of Proteins under Crowding Conditions. *J. Am. Chem. Soc.*, 138(32):10365–10372, 2016. ISSN 15205126. doi: 10.1021/jacs.6b06615.

[166] F. Roosen-Runge, M. Hennig, F. Zhang, R. M. J. Jacobs, M. Sztucki, H. Schober, T. Seydel, and F. Schreiber. Protein self-diffusion in crowded solutions. *Proc. Natl. Acad. Sci. U. S. A.*, 108(29):11815–20, 2011. ISSN 1091-6490. doi: 10.1073/pnas.1107287108.

[167] J. Rosen and Y. C. Kim. Modest Protein - Crowder Attractive Interactions Can Counteract Enhancement of Protein Association by Intermolecular Excluded Volume Interactions. *J. Phys. Chem. B*, I:2683–2689, 2011.

[168] S. B. Ross-Murphy. Dynamic Light Scattering. *Br. Polym. J.*, 9(2):177–177, jun 1977.

[169] P. Rossky, J. Doll, and H. Friedman. Brownian dynamics as smart Monte Carlo simulation. *J. Chem. Phys.*, 69(10):4628–4633, 1978.

[170] E. Rosta and G. Hummer. Error and efficiency of replica exchange molecular dynamics simulations. *J. Chem. Phys.*, 131(16), 2009. ISSN 00219606. doi: 10.1063/1.3249608.

[171] B. Różycki and E. Boura. Large, dynamic, multi-protein complexes: A challenge for structural biology. *J. Phys. Condens. Matter*, 26(46), 2014. ISSN 1361648X. doi: 10.1088/0953-8984/26/46/463103.

[172] B. Rozycki and M. Cieplak. Stiffness of the C-terminal disordered linker affects the geometry of the active site in endoglucanase Cel8A. *Mol. BioSyst. Mol. BioSyst*, 12 (I):1–2, 2016. ISSN 1463-9076. doi: 10.1039/C6MB00606J.

[173] B. Różycki, Y. C. Kim, and G. Hummer. SAXS Ensemble Refinement of ESCRT-III CHMP3 Conformational Transitions. *Structure*, 19(1):109–116, 2011. ISSN 09692126. doi: 10.1016/j.str.2010.10.006.

[174] B. Rózycki, M. Cieplak, and M. Czjzek. Large conformational fluctuations of the multi-domain xylanase Z of Clostridium thermocellum. *J. Struct. Biol.*, 191(1): 68–75, 2015. ISSN 10958657. doi: 10.1016/j.jsb.2015.05.004.

[175] B. Różycki, P.-A. Cazade, S. O'Mahony, D. Thompson, and M. Cieplak. The length but not the sequence of peptide linker modules exerts the primary influence on the conformations of protein domains in cellulosome multi-enzyme complexes. *Phys. Chem. Chem. Phys.*, 19(32):21414–21425, 2017. ISSN 1463-9076. doi: 10.1039/C7CP04114D.

[176] C. D. T. Runge. Ueber die numerische Auflösung von Differentialgleichungen. *Math. Ann.*, 46(2):167–178, 1895. ISSN 00255831. doi: 10.1007/BF01446807.

[177] J. P. Ryckaert, G. Ciccotti, and H. J. Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *J. Comput. Phys.*, 23(3):327–341, 1977.

[178] J. M. Sancho, M. S. Miguel, and D. Dürr. Adiabatic elimination for systems of Brownian particles with nonconstant damping coefficients. *J. Stat. Phys.*, 28(2): 291–305, 1982.

[179] E. Sanz and D. Marenduzzo. Dynamic Monte Carlo versus Brownian dynamics: A comparison for self- diffusion and crystallization in colloidal fluids. *J. Chem. Phys.*, 132(19):194102, 2010.

[180] J. Schluttig, C. B. Korn, and U. S. Schwarz. Role of anisotropy for protein-protein encounter. *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, 81(3):030902, 2010. ISSN 15393755. doi: 10.1103/PhysRevE.81.030902.

[181] J. Schöneberg and F. Noé. ReaDDy - a software for particle-based reaction-diffusion dynamics in crowded cellular environments. *PLoS One*, 8(9), 2013. ISSN 19326203. doi: 10.1371/journal.pone.0074261.

[182] G. Schreiber, G. Haran, and H.-X. Zhou. Fundamental Aspects of Protein−Protein Association Kinetics. *Chem. Rev.*, 109(3):839–860, 2009. ISSN 0009-2665. doi: 10.1021/cr800373w.

[183] H. Sha and F. Zhu. Parameter Optimization for Interaction between C-Terminal Domains of HIV-1 Capsid Protein. *J. Chem. Inf. Model.*, 57(5):1134–1141, 2017. ISSN 15205142. doi: 10.1021/acs.jcim.7b00011.

[184] D. E. Shaw, J. C. Chao, M. P. Eastwood, J. Gagliardo, J. P. Grossman, C. R. Ho, D. J. Lerardi, I. Kolossváry, J. L. Klepeis, T. Layman, C. McLeavey, M. M. Deneroff, M. A. Moraes, R. Mueller, E. C. Priest, Y. Shan, J. Spengler, M. Theobald, B. Towles, S. C. Wang, R. O. Dror, J. S. Kuskin, R. H. Larson, J. K. Salmon, C. Young, B. Batson, and K. J. Bowers. Anton, a special-purpose machine for

molecular dynamics simulation. *Commun. ACM*, 2008. ISSN 00010782. doi: 10.1145/1364782.1364802.

[185] D. E. Shaw, K. J. Bowers, E. Chow, M. P. Eastwood, D. J. Ierardi, J. L. Klepeis, J. S. Kuskin, R. H. Larson, K. Lindorff-Larsen, P. Maragakis, M. A. Moraes, R. O. Dror, S. Piana, Y. Shan, B. Towles, J. K. Salmon, J. P. Grossman, K. M. Mackenzie, J. A. Bank, C. Young, M. M. Deneroff, and B. Batson. Millisecond-scale molecular dynamics simulations on Anton. *Proc. Conf. High Perform. Comput. Networking, Storage Anal. - SC '09*, (c):1, 2009. ISSN 2167-4329. doi: 10.1145/1654059.1654126.

[186] D. Shoup and A. Szabo. Role of diffusion in ligand binding to macromolecules and cell-bound receptors. *Biophys. J.*, 40(1):33–39, 1982.

[187] C. Shutte. Conformational Dynamics : Modelling , Theory , Algorithm , and Application to Biomolecules Conformational Dynamics : odelling , Theory , Algorithm , and Application to Biomolecules. 18(July):1–139, 1999.

[188] A. Sirur and R. B. Best. Effects of interactions with the groel cavity on protein folding rates. *Biophys. J.*, 104(5):1098–1106, 2013. ISSN 00063495. doi: 10.1016/j.bpj.2013.01.034.

[189] A. Sirur, D. De Sancho, and R. B. Best. Markov state models of protein misfolding. *J. Chem. Phys.*, 144(7), 2016. ISSN 00219606. doi: 10.1063/1.4941579.

[190] R. D. Skeel and D. J. Hardy. Practical Construction of Modified Hamiltonians. *SIAM J. Sci. Comput.*, 23:1172, 2001. ISSN 1064-8275. doi: 10.1137/S106482750138318X.

[191] T. Skrbić, P. Faccioli, and C. Micheletti. The role of non-native interactions in the folding of knotted proteins: insights from molecular dynamics simulations. *PLoS Comput. Biol.*, 8(6):1–12, 2012. ISSN 2218273X. doi: 10.3390/biom4010001.

[192] P. E. Smith and W. F. van Gunsteren. Translational and rotational diffusion of proteins. *J. Mol. Biol.*, 236(2):629–636, 1994. ISSN 0022-2836. doi: 10.1006/jmbi.1994.1172.

[193] J. Song, G. N. Gomes, T. Shi, C. C. Gradinaru, and H. S. Chan. Conformational Heterogeneity and FRET Data Interpretation for Dimensions of Unfolded Proteins. *Biophys. J.*, 113(5):1012–1024, 2017. ISSN 15420086. doi: 10.1016/j.bpj.2017.07.023.

[194] S. Sriraman, I. G. Kevrekidis, and G. Hummer. Coarse master equation from Bayesian analysis of replica molecular dynamics simulations. *J. Phys. Chem. B*, 109(14):6479–6484, 2005. ISSN 15206106. doi: 10.1021/jp046448u.

[195] L. S. Stelzl and G. Hummer. Kinetics from Replica Exchange Molecular Dynamics Simulations. *J. Chem. Theory Comput.*, 13(8):3927–3935, 2017. ISSN 15499626. doi: 10.1021/acs.jctc.7b00372.

[196] L. S. Stelzl, A. Kells, E. Rosta, and G. Hummer. Dynamic Histogram Analysis To Determine Free Energies and Rates from Biased Simulations. *J. Chem. Theory Comput.*, 13(12):6328–6342, 2017. ISSN 15499626. doi: 10.1021/acs.jctc.7b00373.

[197] Y. Sugita and Y. Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chem. Phys. Lett.*, 314(1-2):141–151, 1999. ISSN 00092614. doi: 10.1016/S0009-2614(99)01123-9.

[198] K. A. Swanson, R. S. Kang, S. D. Stamenova, L. Hicke, and I. Radhakrishnan. Solution structure of Vps27 UIM-ubiquitin complex important for endosomal sorting and receptor downregulation. *EMBO J.*, 22(18):4597–4606, 2003. ISSN 1460-2075. doi: 10.1093/emboj/cdg471.

[199] R. Swendsen and J. Wang. Replica Monte Carlo simulation of spin glasses. *Phys. Rev. Lett.*, 57(21):2607–2609, 1986. ISSN 1079-7114. doi: 10.1103/PhysRevLett.57.2607.

[200] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *J. Chem. Phys.*, 76(1):637–649, 1982.

[201] A. Szabo. Theory of polarized fluorescent emission in uniaxial liquid crystals. *J. Chem. Phys.*, 72:4620–4626, 1980.

[202] K. Takemura and A. Kitao. Effects of Water Model and Simulation Box Size on Protein Diffusional Motions. *J. Phys. Chem. B*, 111(41):11870–11872, 2007. ISSN 1520-6106. doi: 10.1021/jp0756247.

[203] K. Takemura and A. Kitao. Water model tuning for improved reproduction of rotational diffusion and NMR spectral density. *J. Phys. Chem. B*, 116(22):6279–6287, 2012. ISSN 15205207. doi: 10.1021/jp301100g.

[204] D. Talay. Simulation and numerical analysis of stochastic differential systems : a review. Technical Report RR-1313, Inria, 1990.

[205] D. L. Theobald. Rapid calculation of RMSDs using a quaternion-based characteristic polynomial. *Acta Crystallogr. Sect. A Found. Crystallogr.*, A61:478–480, 2005. ISSN 0108-7673. doi: 10.1107/S0108767305015266.

[206] N. Tjandra, S. E. Feller, R. W. Pastor, and A. Bax. Rotational Diffusion Anisotropy of Human Ubiquitin from NMR Relaxation. *J. Am. Chem. Soc.*, 117(50):12562–12566, 1995.

[207] N. Tjandra, S. E. Feller, R. W. Pastor, and A. Bax. Rotational diffusion anisotropy of human ubiquitin from 15N NMR relaxation. *J. Am. Chem. Soc.*, 117(50):12562–12566, 1995. ISSN 0002-7863. doi: 10.1021/ja00155a020.

[208] N. Tjandra, S. I. Tate, A. Ono, M. Kainosho, and A. Bax. The NMR structure of a DNA dodecamer in an aqueous dilute liquid crystalline phase. *J. Am. Chem. Soc.*, 122(26):6190–6200, 2000. ISSN 00027863. doi: 10.1021/ja000324n.

[209] J. G. Torre de la, B. Carrasco, and S. E. Harding. SOLPRO : theory and computer program for the prediction of SOLution PROperties of rigid macromolecules and bioparticles. *Eur. Biophys. J.*, 25:361–372, 1997.

[210] S. Toxvaerd. Hamiltonians for discrete dynamics. *Phys. Rev. E*, 50:2274, 1994. ISSN 1063651X. doi: 10.1103/PhysRevE.50.2271.

[211] C. Tsallis and D. A. Stariolo. Generalized simulated annealing. *Phys. A Stat. Mech. its Appl.*, 233(1-2):395–406, 1996.

[212] M. Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation.* Oxford University Press, 2010.

[213] I. Tunbridge, R. B. Best, J. Gain, and M. M. Kuttel. Simulation of coarse-grained protein-protein interactions with graphics processing units. *J. Chem. Theory Comput.*, 6(11):3588–3600, 2010. ISSN 15499618. doi: 10.1021/ct1003884.

[214] V. N. Uversky. Use of Fast Protein Size-Exclusion Liquid Chromatography To Study the Unfolding of Proteins Which Denature through the Molten Globule. *Biochemistry*, 32(48):13288–13298, 1993. ISSN 15204995. doi: 10.1021/bi00211a042.

[215] A. Van Blaaderen, J. Peetermans, G. Maret, and J. K. Dhont. Long-time self-diffusion of spherical colloidal particles measured with fluorescence recovery after photobleaching. *J. Chem. Phys.*, 96(6):4591–4603, 1992. ISSN 00219606. doi: 10.1063/1.462795.

[216] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The NumPy array: A structure for efficient numerical computation. *Comput. Sci. Eng.*, 13(2):22–30, 2011. ISSN 15219615. doi: 10.1109/MCSE.2011.37.

[217] W. F. Van Gunsteren and H. J. C. Berendsen. A Leap-frog Algorithm for Stochastic Dynamics. *Mol. Simul.*, 1(3):173–185, 1988. ISSN 0892-7022. doi: 10.1080/08927028808080941.

[218] C. Vega and J. L. F. Abascal. Simulating water with rigid non-polarizable models: a general perspective. *Phys. Chem. Chem. Phys.*, 13(44):19663–20022, 2011. ISSN 1463-9076. doi: 10.1039/c1cp22168j.

[219] R. M. Venable, H. I. Ingólfsson, M. G. Lerner, B. S. Perrin, B. A. Camley, S. J. Marrink, F. L. Brown, and R. W. Pastor. Lipid and Peptide Diffusion in Bilayers: The Saffman-Delbrück Model and Periodic Boundary Conditions. *J. Phys. Chem. B*, 121(15):3443–3457, 2017. ISSN 15205207. doi: 10.1021/acs.jpcb.6b09111.

[220] L. Verlet. Computer Expermiments on Classical Fliuds I: Thermodynamical Properties of Lennard Jones Molecules. *Phys Rev.*, 159(1):98–103, 1967.

[221] M. Vögele and G. Hummer. Divergent Diffusion Coefficients in Simulations of Fluids and Lipid Membranes. *J. Phys. Chem. B*, 120(33):8722–8732, 2016. ISSN 15205207. doi: 10.1021/acs.jpcb.6b05102.

[222] M. Vögele, J. Köfinger, and G. Hummer. Hydrodynamics of Diffusion in Lipid Membrane Simulations. *Phys. Rev. Lett.*, 120(26):268104, 2018. ISSN 1079-7114. doi: 10.1103/PhysRevLett.120.268104.

[223] S. von Bülow, M. Siggel, M. Linke, and G. Hummer. Dynamic cluster formation determines viscosity and diffusion in dense protein solutions. *Proc. Nat. Acad. Sci. USA*, In Review, 2019.

[224] D. Wang, U. Kreutzer, Y. Chung, and T. Jue. Myoglobin and hemoglobin rotational diffusion in the cell. *Biophys. J.*, 73(5):2764–2770, 1997. ISSN 0006-3495. doi: 10.1016/S0006-3495(97)78305-X.

[225] J. Wang, P. Cieplak, and P. A. Kollman. How well does a restrained electrostatic potential (RESP) model perform in calculating conformational energies of organic and biological molecules? *J. Comput. Chem.*, 21(12):1049–1074, 2000.

[226] J. D. Weeks, D. Chandler, and H. C. Andersen. Role of repulsive forces in determining the equilibrium structure of simple liquids. *J. Chem. Phys.*, 54(12):5237–5247, 1971.

[227] D. K. Wilkins, S. B. Grimshaw, V. Receveur, C. M. Dobson, J. A. Jones, and L. J. Smith. Hydrodynamic Radii of Native and Denatured Proteins Measured by Pulse Field Gradient NMR Techniques. *Biochemistry*, 38(50):16424–16431, 1999. ISSN 00062960. doi: 10.1021/bi991765q.

[228] U. Winter and T. Geyer. Coarse grained simulations of a small peptide: Effects of finite damping and hydrodynamic interactions. *J. Chem. Phys.*, 131(10), 2009. ISSN 00219606. doi: 10.1063/1.3216573.

[229] D. E. Woessner. Nuclear Spin Relaxation in Ellipsoids Undergoing Rotational Brownian Motion. *J. Chem. Phys.*, 37(3):647–654, 1962. doi: 10.1063/1.1701390.

[230] D. E. Woessner. Spin Relaxation Processes in a Two-Proton System Undergoing Anisotropic Reorientation. *J. Chem. Phys.*, 36(1):1, 1962. doi: 10.1063/1.1732274.

[231] M. Wolf, R. Gulich, P. Lunkenheimer, and A. Loidl. Relaxation dynamics of a protein solution investigated by dielectric spectroscopy. *Biochim. Biophys. Acta (BBA)-Proteins Proteomics*, 1824(5):723–730, 2012. ISSN 15709639. doi: 10.1016/j.bbapap.2012.02.008.

[232] V. Wong and D. A. Case. Evaluating rotational diffusion from protein MD simulations. *J. Phys. Chem. B*, 112:6013–6024, 2008. ISSN 15206106. doi: 10.1021/jp0761564.

[233] S. Yadahalli, V. V. Hemanth Giri Rao, and S. Gosavi. Modeling Non-Native Interactions in Designed Proteins. *Isr. J. Chem.*, 576104, 2014. ISSN 00212148. doi: 10.1002/ijch.201400035.

[234] H. Yamakawa. *Modern Theory of Polymer Solutions*. Harper & Row, NewYork, 1971.

[235] D. R. Yarkony. *Modern electronic structure theory Part II*. World Scientific Publishing, Singapore, 1995.

[236] I.-C. Yeh and G. Hummer. System-Size Dependence of Diffusion Coefficients and Viscosities from Molecular Dynamics Simulations with Periodic Boundary Conditions In-Chul. *J. Phys. Chem. B*, 108(40):15873–15879, 2004. doi: 10.1021/acs.jctc.7b00372.

[237] H. Yoshida. Construction of higher order symplectic integrators. *Phys. Lett. A*, 150: 262, 1990. ISSN 03759601. doi: 10.1016/0375-9601(90)90092-3.

[238] I. Yu, T. Mori, T. Ando, R. Harada, J. Jung, Y. Sugita, and M. Feig. Biomolecular interactions modulate macromolecular structure and dynamics in atomistic model of a bacterial cytoplasm. *Elife*, 5:e19274, 2016. ISSN 2050-084X. doi: 10.7554/eLife.19274.

[239] X. Yu, M. Martinez, A. L. Gable, J. C. Fuller, N. J. Bruce, S. Richter, and R. C. Wade. webSDA: a web server to simulate macromolecular diffusional association. *Nucleic Acids Res.*, 43(W1):W220–W224, 2015. ISSN 0305-1048. doi: 10.1093/nar/gkv335.

[240] W. Zheng, A. Borgia, K. Buholzer, A. Grishaev, B. Schuler, and R. B. Best. Probing the Action of Chemical Denaturant on an Intrinsically Disordered Protein by Simulation and Experiment. *J. Am. Chem. Soc.*, 138(36):11702–11713, 2016. ISSN 15205126. doi: 10.1021/jacs.6b05443.

[241] F. Zhu and B. Chen. Monte Carlo Simulations of HIV Capsid Protein Homodimer. *J. Chem. Inf. Model.*, 55(7):1361–1368, 2015. ISSN 15205142. doi: 10.1021/acs.jcim.5b00126.

[242] M. Zuzovsky, P. M. Adler, and H. Brenner. Spatially periodic suspensions of convex particles in linear shear flows. III. Dilute arrays of spheres suspended in Newtonian fluids. *Phys. Fluids*, 26(7):1714, 1983.

[243] R. Zwanzig. *Nonequilibrium statistical mechanics*, volume 54. 2001. ISBN 0195140184.

List of Abbreviations

ACF	autocorrelation function
API	application programming interface
BD	Brownian dynamics
BDMC	Brownian dynamics Monte Carlo
CG	coarse-grained
CLI	command line interface
CPU	central processing unit
FRET	fluorescence resonance energy transfer
GB3	the third IgG-binding domain of Protein G
KH	Kim-Hummer
LJ	Lennard Jones
MCMC	Markov chain Monte Carlo
MD	molecular dynamics
MIC	minimum image convention
MJ	Miyazawa and Jernigan
MPI	message passing interface
MSD	mean square displacement
NMR	nuclear magnetic resonance
PBC	periodic boundary conditions
PCS	principal coordinate system
PDB	Protein Data Bank
PMF	potential of mean force
RCS	reference coordinate system
REMC	Replica Exchange Monte Carlo
RMSD	root-mean-square distance
RNA	ribonucleic acid
RNG	random number generator
SAXS	small angle x-ray scattering
SEM	standard error of the mean

SIMD	single instruction multiple data
TBA	transition based assignment
WCA	Weeks-Chandler-Anderson

List of Tables

List of Figures

List of Algorithms

List of Listings

Acknowledgement

I thank my advisor Gerhard Hummer to have given me the opportunity to work on the topic of diffusion in simulations. The last three and a half years have been a great joy for me in which I learned a lot about physics and myself. I'm very grateful for this. I'm also thankful for the invaluable help of Jürgen Köfinger. He has given me lots of insightful advice for all the big and small problems that I had in the last three and a half years. I also thankful to my many collaborators that have helped in writing the Complexes++ software package: Bramas Berenger, Johannes Schöneberg, Patrick Quoika, Benoit-Lofti Jacquelin, Klaus Reuter. I'm also grateful for my office mates Lukas Stelzl, Martin Vögele and Michael Gecht for many discussions about physics, programming and life. Last I thank my amazing wife Dorota for her continued support and encouragement. She shows me everyday that marrying her has been the best decision of my life.

Curriculum Vitae

Personal details

Name:	Max Linke
Date of birth:	16 Oktober 1988
Place of birth:	Gera

Education & research experience

Since 01/2015	Doctoral dissertation
	Johann Wolfgang Goethe-University, Frankfurt and
	Max Planck Institute of Biophysics,
	Methods for Simulation and Calculation of Diffusion
04/2012 - 09/2014	Master thesis
	Georg August University, Göttingen and
	Max Planck Institute for biophysical chemistry,
	Analysis of Protein Dynamics with Markov Models
08/2011 - 12/2012	Study abroad
	University of Vienna, Vienna, Austria
9/2011 - 09/2014	Master of Physics
	Georg August University, Göttingen
10/2008 - 06/2011	Bachelor Thesis
	Georg August University, Göttingen
	Quantum dot Tracking of EGF Receptor traffic
10/2008 - 06/2011	Bachelor Physics
	Georg August University, Göttingen
6/2008	Abitur
	Hilda Gymnasium, Pforzheim

Publikationen

Teile dieser Arbeit wurden bereits in folgenden Publikationen veröffentlicht:

M. Linke, J. Köfinger, G. Hummer. Fully Anisotropic Rotational Diffusion Tensor from Molecular Dynamics Simulations. *J. Phys. Chem. B* 2018
M. Linke, J. Köfinger, G. Hummer. Rotational Diffusion Depends on Box Size in Molecular Dynamics Simulations. *J. Phys. Chem. Lett.* 2018

Weiter Publikationen während der PhD Zeit:

R.J. Gowers, M. Linke, J. Barnoud, T. Reddy, M.N. Melo, S.L. Seyler, D.L. Dotson, J. Domanski and S. Buchoux, I.M. Kenney, O. Beckstein. MDAnalysis: a Python package for the rapid analysis of molecular dynamics simulations. *Proceedings of the 15th Python in Science Conference* 2016
D.L. Dotson, S.L. Seyler, M. Linke, R.J. Gowers, O. Beckstein. datreant: persistent, Pythonic trees for heterogeneous data. *Proceedings of the 15th Python in Science Conference* 2016
S.v. Bülow, M. Siggel, M. Linke, G. Hummer. Dynamic cluster formation determines viscosity and diffusion in dense protein solutions. *PNAS* 2019

Vorträge

Teile dieser Arbeit wurden in folgenden Vorträgen vorgestellt:

M. Linke, G. Hummer, Complexes++, Oliver Beckstein, Phoenix, AZ, USA, October 2017
M. Linke, J. Köfinger, G. Hummer, Rotational Diffusion in Simulations, BDBDB4, Heidelberg, Deutschland, April 2018

Posterpräsentationen

Teile dieser Arbeit wurden auf folgenden Konferenzen als Posterpräsentationen vorgestellt:

M. Linke, J. Köfinger, G. Hummer. Rotational Diffusion Tensor from Molecular Dynamics Simulations. Computer Simulation and Theory of Macromolecules, Hünfeld, Deutschland, April 2017.