UNIVERSITY OF CALIFORNIA SAN DIEGO

Forging Pathways to Enable Multiscale Modeling of Cellular Scale Phenomena

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Chemistry

by

Christopher Ting-Kuang Lee

Committee in charge:

Professor Rommie E. Amaro, Co-Chair
Professor J. Andrew McCammon, Co-Chair
Professor Michael Gilson
Professor Michael Holst
Professor Katja Lindenberg
Professor Francesco Paesani

2019

The dissertation of Christopher Ting-Kuang Lee is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Co-Chair

Co-Chair

University of California San Diego

2019

DEDICATION

To my parents Serena, and Tsengdar: for the endless support and encouragement.

To my sister Gloria: for challenging me to be my best.

To Jim, and Tanya: for sharing their home and welcoming me.

To Linda, and Cam: without your influence I am not sure I would be the scientist I am today.

To Bryn: for the courage to pursue my dreams in spite of my doubts. As a wise man once wrote, “for keeping me human as I become a scientist.”

EPIGRAPH

We will always have STEM with us. Some things will drop out of the public eye and will go away, but there will always be science, engineering and technology. And there will always, always be mathematics. Everything is physics and math.

—Katherine Johnson

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGEMENTS

A PhD is an important recognition from the scientific community and institution signifying the success and qualifications of an individual to pursue creative and original research. This success is not spontaneous but typically builds upon many years of academic and other preparation. I thank all of the innumerable mentors who have influenced me throughout the years. Especially Professors Linda Columbus, Cameron Mura, and Michael Shirts.

I would like to acknowledge my advisors Rommie Amaro and Andy McCammon. They have created a stimulating and rich yet flexible research environment which has afforded me the ability to explore my personal research interests with their full support. Through the highs and lows of our work, they have guided me and pushed me to grow in all aspects of my life. Their constant encouragement has allowed me to flourish in ways I never thought possible. During my graduate tenure, perhaps most importantly, they have entrusted me with the role of mentoring many students across all levels of academic preparation. They have enabled my passion for sharing the pursuit of science with others. I will treasure this gift, thank you.

Rob Swift was a close mentor during the first two years of my degree. He helped lay the foundations for my graduate success. Thanks for sharing your humor, scientific rigor, and skepticism which have forged me into the scientist I am today. I also thank my committee members Katja Lindenberg, Mike Holst, Francesco Paesani, and Mike Gilson. You were tough but fair. Thanks for always pushing me to think critically.

Special thanks to all of the other current and former members of the Amaro and McCammon groups. I have realized that listing all of your many contributions to my life and work could take me an additional 5 years!

I am also thankful for various sources of funding and computing resources which supported my research. I especially thank the UCSD Molecular Biophysics Training Program for also providing me with a community of valued colleagues.

Chapter 2, in full, is a modified reprint of the material as it appears in “Lee, C. T.; Amaro, R. E. *Exascale Computing: A New Dawn for Computational Biology.* Comput. Sci. Eng. 2018, 20 (5), 18–25.”.

The dissertation author was the primary investigator and author of this work.

Chapter 3, in full, is a modified reprint of the material as it appears in "Lee, C. T.; Comer, J.; Herndon, C.; Leung, N.; Pavlova, A.; Swift, R. V.; Tung, C.; Rowley, C. N.; Amaro, R. E.; Chipot, C.; Wang, Y.; Gumbart J. C. *Simulation-Based Approaches for Determining Membrane Permeability of Small Compounds.* J. Chem. Inf. Model. 2016, 56 (4), 721–733." The dissertation author was the primary investigator and author of this work.

Chapter 4, in full, is a modified reprint of the material as it appears in "Votapka, L. W.†; Lee, C. T.†; Amaro, R. E. *Two Relations to Estimate Membrane Permeability Using Milestoning.* J. Phys. Chem. B 2016, 120 (33), 8606–8616." The dissertation author was a primary coinvestigator and author of this work.

Chapter 5, in full, is a modified reprint of the material as it appears in "Jagger, B. R.; Lee, C. T.; Amaro, R. E. *Quantitative Ranking of Ligand Binding Kinetics with a Multiscale Milestoning Simulation Approach.* J. Phys. Chem. Lett. 2018, 9 (17), 4941–4948." The dissertation author was an investigator, supervisor, and coauthor of this work.

Chapter 6, in full, is a modified reprint of the material as it appears in "Wagner, J. R.†; Lee, C. T.†; Durrant, J. D.; Malmstrom, R. D.; Feher, V. A.; Amaro, R. E. *Emerging Computational Methods for the Rational Discovery of Allosteric Drugs.* Chem. Rev. 2016, 116 (11), 6370–6390.". The dissertation author was a primary coinvestigator and author of this work.

Chapter 7, in full, has been submitted for publication and is presented as it may appear in "Taylor, B. C.; Lee, C. T.; Amaro, R. E. *Structural Basis for Ligand Modulation of the CCR2 Conformational Landscape.* PNAS." The dissertation author was an investigator, supervisor, and coauthor of this work.

Chapter 8, in full, has been submitted for publication and is presented as it may appear in "Lee, C. T.†; Moody, J. B.†; Amaro, R. E.; McCammon, J. A.; Holst, M. *The Implementation of the Colored Abstract Simplicial Complex and Its Application to Mesh Generation.* Trans. Math. Soft." The dissertation author was a primary coinvestigator and author of this work.

VITA

2010-2011	Undergraduate Research Fellow Shirts Lab, Department of Chemical Engineering, University of Virginia
2011	Bachelor of Science in Chemistry University of Virginia
2011	Bachelors of Arts in Computer Science University of Virginia
2011	Graduate Research Fellow Columbus and Mura Labs, Department of Chemistry, University of Virginia
2013	Master of Chemistry With a Specialization in Biochemistry University of Virginia
2013-2019	Graduate Research Fellow Amaro and McCammon Labs, University of California San Diego
2019	Doctor of Philosophy in Chemistry University of California San Diego

PUBLICATIONS

Malmstrom, R. D.; **Lee, C. T.**; Van Wart, A. T.; Amaro, R. E. *Application of Molecular-Dynamics Based Markov State Models to Functional Proteins.* J. Chem. Theory Comput. 2014, 10 (7), 2648–2657.

Gray, C.; Price, C. W.; **Lee, C. T.**; Dewald, A. H.; Cline, M. A.; McAnany, C. E.; Columbus, L.; Mura, C. *Known Structure, Unknown Function: An Inquiry-Based Undergraduate Biochemistry Laboratory Course.* Biochem. Mol. Biol. Educ. 2015, 43 (4), 245–262.

Wagner, J. R.[†]; **Lee, C. T.**[†]; Durrant, J. D.; Malmstrom, R. D.; Feher, V. A.; Amaro, R. E. *Emerging Computational Methods for the Rational Discovery of Allosteric Drugs.* Chem. Rev. 2016, 116 (11), 6370–6390.

Votapka, L. W.[†]; **Lee, C. T.**[†]; Amaro, R. E. *Two Relations to Estimate Membrane Permeability Using Milestoning.* J. Phys. Chem. B 2016, 120 (33), 8606–8616.

Lee, C. T.; Comer, J.; Herndon, C.; Leung, N.; Pavlova, A.; Swift, R. V.; Tung, C.; Rowley, C. N.; Amaro, R. E.; Chipot, C.; Wang, Y.; Gumbart J. C. *Simulation-Based Approaches for Determining Membrane Permeability of Small Compounds.* J. Chem. Inf. Model. 2016, 56 (4), 721–733.

Lee, C. T.; Amaro, R. E. *Exascale Computing: A New Dawn for Computational Biology.* Comput. Sci. Eng. 2018, 20 (5), 18–25.

Jagger, B. R.; **Lee, C. T.**; Amaro, R. E. *Quantitative Ranking of Ligand Binding Kinetics with a Multiscale Milestoning Simulation Approach.* J. Phys. Chem. Lett. 2018, 9 (17), 4941–4948.

Taylor, B. C.; **Lee, C. T.**; Amaro, R. E. *Structural Basis for Ligand Modulation of the CCR2 Conformational Landscape.* bioRxiv 2018, 392068. *Submitted*

Lee, C. T.†; Moody, J. B.†; Amaro, R. E.; McCammon, J. A.; Holst, M. *The Implementation of the Colored Abstract Simplicial Complex and Its Application to Mesh Generation.* 2018. arXiv: 1807.01417. *Submitted*

Lee, C. T.†; Laughlin, J. G.†, Angliviel de La Beaumelle, N.; Amaro, R. E.; McCammon, J. A.; Ramamoorthi, R.; Holst, M. J.; Rangamani, P. *GAMer 2: A System for 3D Mesh Processing of Cellular Electron Micrographs.* bioRxiv 2019. *Submitted*

ABSTRACT OF THE DISSERTATION

Forging Pathways to Enable Multiscale Modeling of Cellular Scale Phenomena

by

Christopher Ting-Kuang Lee

Doctor of Philosophy in Chemistry

University of California San Diego, 2019

Professor Rommie E. Amaro, Co-Chair
Professor J. Andrew McCammon, Co-Chair

The dream of any biologist is to have an "Asimovian nanosubmarine" to allow the observations of molecular scale events first hand. While such a device does not currently exist, biologists and chemists can instead employ multiscale modeling techniques to investigate molecular behaviors. By integrating kinetic parameters, structural data, and computation, it is possible to animate cellular scenes. While the construction of such a model may seem simple, the task requires the assimilation of parameters from literature, and the generation of computable representations of cellular structures. Where parameters are missing, computational approaches such as molecular dynamics can be used to supplement by predicting values of interest. In this dissertation, I describe my efforts in computing important physical properties such as passive membrane permeability of drug molecules, and binding/unbinding kinetics using molecular dynamics and a variety of enhanced sampling strategies. Beyond these transport properties, the conformational dynamics of proteins is critical to understand problems such as drug efficacy and protein function. I present a review of emerging computational methods to rationally design allosteric drugs, and a study using Markov state modeling theory to represent the differential dynamics of C-C chemokine receptor type 2 with and without drugs bound. To address the computer representation of cellular structures, I present an implementation of a general simplicial complex (mesh triangulations) data structure suitable for mesh generation applications. Collectively the work presented in this dissertation address the barriers hindering the development of physical models of whole cells on multiple fronts.

Chapter 1

Introduction: Why Model Biology?

The goal of biology is to understand life to an extent that one can predict the eventual behaviors of a living system under or in the absence of a stimulus or perturbation. Although many debate the utility of quantitative modeling in answering this question, I will abstain from such discussion and focus on the benefits of modeling as I view them. The development of a whole cell or organism model has become a hot topic in modern biology[1]. As biologists uncover the mysteries which make life possible, they unveil new complexities[2]. These complex and non-linear behaviors are not amenable to scrutiny via traditional research methods which are driven by classical hypotheses (i.e., by varying one parameter at a time and observing and characterizing the results to form conclusions). Simple experimental models, while elegant, test isolated model behaviors in simplified systems which are often not generalizable. In order to derive a general conclusion, many expensive experiments must be run to explore the solution space at large potential cost.

One strategy to reduce the experimental cost burden is to shift towards inexpensive mathematical modeling. Mathematical models in biology are increasing in popularity, and come in a wide variety of flavors[3]. For example, there are information and data driven models as well as models based on the fundamentals of physical principles. In this dissertation, I will focus on the development of methods in support of the latter. By codifying our knowledge of the physical world into a model, we can gain many benefits. First, physical models can serve as a glue to reconcile the results of individual experiments under isolated conditions. Cleverly designed mathematical models are also amenable to computed solutions using numerical methods. Combining knowledge and computation, inexpensive parameter sweeps that eliminate physical impossibilities and guide experimental efforts can serve to reduce the experimental cost of biology.

Physical models can also produce other benefits for science. During the design and validation phase of model construction, when a physical model fails to reproduce experimental results, this is a litmus test for the limitations of our current understanding. By studying the assumptions of the model and querying the differences between prediction and experiment, scientists can pinpoint possible knowledge gaps. Once a physical model is sufficiently validated, due to the fundamental basis of transferable physical properties, the models can also be used to generate prospective predictions. For example, in an inspiring study, Hodgkin and Huxley created a famous model describing the propagation of an action potential along a giant squid axon[4]. Although the model was based on empirical fitting of electrophysiology data, many speculated upon the physical implications of the mathematics. One term in particular, predicting the presence of four voltage-sensitive gates in the sodium ion channel, was later experimentally confirmed by crystallographic structure which resolved the tetrameric channel structure[5].

We live in an exciting time, where there is a convergence of emerging structural data, legacy experimental parameters, and computational power. In order to understand life, we must embrace the complexity of biology. The work described in this dissertation seeks to address several challenges which hinder the routine application of large systems biology models. This includes the computational prediction of various biological rates such as diffusion, permeability, and binding/unbinding, strategies to develop models of individual protein dynamics, and the capacity to generate multiscale geometric models suitable for biological modeling.

1.1 Overview of Chapter Contents

To reiterate and highlight the opportune convergence of computing and biological data, Chapter 2 discusses the potential applications and gains afforded by future exascale supercomputers to biology. This chapter provides some perspective on the role that computers can play to solve various biological problems.

We speculate upon not only the contributions of computation to basic science, but also upon the impact of computation and technology on translational fields such as personalized medicine. In this chapter, we also address and frame some of the aforementioned challenges to future modeling efforts with solutions described in the subsequent chapters.

Owing to the complexity of biology, a complete systems model will require the definition of many rates. Due to the basic nature of many of these measurements, the experimental work to measure these values may not be publishable and is therefore difficult to source. Although robotics which automate assay development and screening are amenable to some tasks, the complexity of experimental setups for measuring kinetic properties can often hinder accurate and high-throughput measurement. To augment the existing data and measurements, molecular simulations such as Molecular Dynamics (MD) can be employed to generate estimates[6, 7]. MD effectively integrates the classical equations of motion for an atomistic system. Given that the typical timescales of simulation are typically much shorter than the timescales of the process of interest, brute force MD simulations can rarely capture the dynamics of interest with sufficient statistics. Instead, using strategies derived from statistical mechanics, many enhanced sampling strategies have been developed to facilitate the estimation of classically intractable properties[8, 9]. In Chapter 3, we compare the efficiency of four enhanced sampling strategies to compute drug passive membrane permeability. We find that there are many orthogonal degrees of freedom which can hinder the convergence of these calculations and recommend some best practices for future work.

While studying membrane permeability, I had many stimulating discussions with my colleague Lane Votapka on the topic. During our overlap in the group, Lane was working on the development of a software tool which facilitates the use of milestoning to compute various rates. Milestoning is another enhanced sampling strategy where, instead of observing the full transition from start to finish of a process, we collect statistics of transitions along many small segments of the full pathway and stitch them together in postprocessing[10–16]. The application of milestoning theory to permeability calculations was not straightforward since the natural currency of milestoning is transition probabilities, while the expressions to compute permeability required potentials of mean force. To solve this, Lane derived a new relation to compute permeability values using transition probabilities from milestoning which we validated using a toy model inspired by my prior work on permeability. This research is described in Chapter 4.

After learning about the virtues of the milestoning method, the group started thinking about possible applications of milestoning to compute other rates of interest. By chance, my colleague Ben Jagger and I met Professor Chia-en Chang at a local conference where she was presenting preliminary results describing brute-force MD calculations of host-guest binding kinetics and thermodynamics[17]. This dataset which contained several ligands, experimentally determined kinetics, and brute-force MD was the ideal system to benchmark the use of SEEKR[16]. Chia-en kindly shared the dataset with us and working with Ben, we predicted the binding kinetics of the system using milestoning. The results of this work are described in Chapter 5 along with some best practices to monitor the convergence of computed rate estimates which we developed along the way.

Also of interest for biological simulations is the dynamics of protein movement. Not only are the dynamics of proteins critical to their biological function, but protein dynamics also influence phenomena such as drug binding. In Chapter 6, we present a review of many different strategies to use computers to inform the rational design of allosteric drugs. Previously I had worked with Robert Malmstrom on a paper describing the applications of Markov state models to model protein kinase A dynamics with and without cyclic AMP[18]. During this project, we started thinking about the use of MSMs to model perturbations to the conformational ensembles of protein targets by drugs and using the information to support drug discovery efforts. Later working with Bryn Taylor, we investigated the impact of drug binding to GPCR CCR2. The generation and description of our CCR2 MSMs is presented in Chapter 7.

Another important aspect of cell modeling is the generation of computable geometries to represent cellular scenes. Meshes are commonly used in engineering fields as a general geometric representation. The use of meshes is desirable since they are often also compatible with simulation methods such as finite elements. However, the numerical behavior of algorithms is often limited by mesh conditioning. To support the development of robust mesh generation codes for biological geometries, working with John Moody, we developed the Colored Abstract Simplicial Complex library which is described in Chapter 8.

In summary, the work in this dissertation explores the use of various modeling strategies to predict values of interest or to generate geometric meshes in support of future multiscale modeling efforts.

References

(1) Roberts, E. *Curr. Opin. Struct. Biol.* **2014**, *25*, 86–91.

(2) Check Hayden, E. *Nature* **2010**, *464*, 664–7.

(3) Gunawardena, J. Models in biology: 'Accurate descriptions of our pathetic thinking'., 2014.

(4) Hodgkin, A. L.; Huxley, A. F. *Bull. Math. Biol.* **1990**, *52*, 25–71.

(5) Sigg, D. *J. Gen. Physiol.* **2014**, *144*, 7–26.

(6) Leach, A. R. *Pearson Educ. EMA* **2001**, DOI: 10.1016/S0097-8485(96)00029-0.

(7) Durrant, J. D.; McCammon, J. A. *BMC Biol.* **2011**, *9*, 71.

(8) *Free Energy Calculations*; Chipot, C., Pohorille, A., Eds.; Springer Series in Chemical Physics, Vol. 86; Springer Berlin Heidelberg: Berlin, Heidelberg, 2007.

(9) Tuckerman, M., *Statistical Mechanics: Theory and Molecular Simulation*; OUP Oxford: Oxford, 2010, p 720.

(10) Faradjian, A. K.; Elber, R. *J. Chem. Phys.* **2004**, *120*, 10880–9.

(11) Májek, P.; Elber, R. *J. Chem. Theory Comput.* **2010**, *6*, 1805–1817.

(12) Vanden-Eijnden, E.; Venturoli, M.; Ciccotti, G.; Elber, R. *J. Chem. Phys.* **2008**, *129*, 174102.

(13) Kirmizialtin, S.; Elber, R. *J. Phys. Chem. A* **2011**, *115*, 6137–6148.

(14) Votapka, L. W.; Amaro, R. E. *PLoS Comput. Biol.* **2015**, *11*, e1004381.

(15) Bello-Rivas, J. M.; Elber, R. *J. Chem. Phys.* **2015**, *142*, 094102.

(16) Votapka, L. W.; Jagger, B. R.; Heyneman, A. L.; Amaro, R. E. *J. Phys. Chem. B* **2017**, *121*, 3597–3606.

(17) Tang, Z.; Chang, C.-e. A. *J. Chem. Theory Comput.* **2018**, *14*, 303–318.

(18) Malmstrom, R. D.; Kornev, A. P.; Taylor, S. S.; Amaro, R. E. *Nat. Commun.* **2015**, *6*, 7588.

Chapter 2

Exascale Computing: A New Dawn for Computational Biology

2.1 Abstract

As biologists discover and learn to embrace the complexity of biological systems, computational data analysis and modeling have become critical for furthering our understanding. Exascale computing will enable the development of new predictive multiscale models, transforming how we study the behaviors of organisms and ecosystems, ultimately leading to new innovations and discoveries.

2.2 Introduction

Computers have completely revolutionized the way we study and understand the complexity and multiscale nature of biology. From simulating the bond making and breaking of enzyme catalysis to computing protein sequence alignments, nearly every aspect of modern biology has some level of computation involved. There has been a clear trend throughout history of algorithmic and hardware advances creating new opportunities for biologists. In the early days, computers were used to accelerate difficult tasks, such as the determination of structures from X-ray diffraction data, as well as to piece together full gene sequences from sequence fragments. This spurred an era of uncovering the molecular basis of biology. Since then, drastic improvements in computing power and automation have enabled the rapid creation of large structural and sequence datasets such as the Protein Data Bank and the Human Genome Project[1, 2]. At the same time, owing to the development of numerical integration techniques and improvements in our understanding of chemical and biological physics, scientists began developing complicated models and simulations of various phenomena. Without the driving force of computation, our understanding of biology simply would not be as advanced as it is today. Even the latest structural biology "revolution," cryoelectron microscopy (CryoEM), is only possible because of the ability to store and process massive imaging datasets using computers[3]. The marriage of computation and biology has completely changed our understanding of life and revolutionized medicine. This trend where innovations in biological research piggyback off advances in computation will continue. Efforts such as the Exascale Computing Project (ECP) spearheaded by the National Strategic Computing Initiative (NSCI) promise to reduce the time to deployment of the next generation of exascale supercomputing facilities, and we predict that this will have a transformative effect on computational biology.

The ultimate goal of computational biology is to develop the ability to predict, control, and design the function of organisms and biomes. We seek to understand, through the discovery and characterization of networks of biochemical and chemical linkages/reactions defining metabolism and other relevant chemistries, how perturbations such as the introduction of a drug or changing environmental conditions will affect an organism or population. Despite decades of experimental parameter collection, our ability to model across spatiotemporal scales to predict emergent system behaviors remains unrealized. Existing efforts have been confounded by both the need to account for phenomena that occur over 15 and 10 orders of temporal and spatial magnitude, respectively, as well as the need to accurately account for physiological or genetic variability in the population. Combating these factors, biologists have turned toward increasingly quantitative and data-driven methods to identify anomalies and query behaviors of interest. Through a combination of clever experimental design and technological innovation, experiments such as sequencing, structural imaging, and molecular functional assays are being automated. The trends of the past several years indicate that the output data rates and data sizes of these technologies are growing at an exponential rate. To convert these raw data streams into useful information such as genes and genomes, protein and cellular ultrastructures, and chemical activities, computationally intensive algorithms that require high memory and disk space are employed. To keep up with the flood of data, in the future, biologists will need access to suitable new computing resources such as exascale high-performance computing (HPC). Future research into algorithms that scale into exascale will enable the development of data assimilation workflows, and strategies for interconnecting different simulation modalities to enable predictive multiscale systems biology.

Exascale computing refers to the development of computing systems that are 1,000 times faster than existing petaflop machines, capable of the sustained delivery of at least 1 exaflop (10^{18} calculations per second). Note that while FLOPs were historically the biggest cost in HPC, in today's post-Moore's law era, the primary design constraint will be system power usage. In our current technology, the cost of data movement is far greater than compute capability. In essence, exascale HPC will likely provide exponential gains in concurrency and parallelism with only modest gains in memory scaling. Nevertheless, these resources, coupled with other compute modalities, will help computational biologists address several of the fundamental challenges we face on a routine basis:

- insufficient model/analysis throughput;
- insufficient simulation domain or detail;
- incomplete model physics;
- lack of uncertainty quantification; and
- lack of workflows that are able to reproducibly and robustly integrate simulation, experimental data, and analytics.

By tackling these roadblocks, exascale computing will create new opportunities to develop more realistic simulations and derive new insights from large datasets. Such advances stand to transform what we understand about the -omics fields (genomics, metagenomics, proteomics, metabolomics, and so on), evolutionary biology, personalized medicine, and drug discovery.

2.3 Keeping Pace with Data

Improvements in technology and automation have led to the creation of enormous streams of experimental data. For example, advances such as next-generation sequencing have driven down the costs of sequencing whole genomes to an affordable level. Efforts are already underway to deep sequence millions of human genomes[4], while others seek to sequence thousands of non-human organisms across all domains of life[5, 6]. What comes out of the sequencer is not an ordered genome sequence but rather the sequences of billions of short fragments (multiple terabytes). Biologists must then use complex genome assembly algorithms, which compare and align these fragments to assemble the complete genome. This is a classical bioinformatics task that can require up to several terabytes of memory[7]. Although the genome assembly problem traditionally necessitates a tightly coupled (HPC) compute infrastructure due to the algorithmic communication load[8], some have developed MapReduce-based algorithms to run in the cloud[9]. Additional research in the future will be necessary to determine whether the flexibility of elastic cloud computing at the sacrifice of increased network communication latency will be preferred by the community.

Once the genome is assembled, biologists can use computationally driven comparative analysis to correlate mutations with diseases and to track the evolutionary (phylogenetic) history of life. Here the computation typically requires pairwise comparisons between genomes or genes. Therefore, the computational effort scales quadratically with respect to the number of available sequences. As the number of complete genomes grows exponentially, exascale computing and other compute modalities can provide the necessary throughput to support these analyses. A philosophical question arises as to whether to store the data in a single resource or distribute it in the cloud. Benefits of localization include reduced network traffic and potentially superior dataset security and provenance. On the other hand, cloud solutions provide superior flexibility for sporadic compute loads. Biologists must work together to develop best practices to protect private patient data as well as to track the software and hardware versions in data processing steps.

Beyond this, other important questions remain. Predicting and annotating the functions of genes in genomic and metagenomic sequences will be an ongoing challenge. With sufficient annotation coverage, the metabolism of new organisms can be automatically modeled to predict its functional behaviors. Given the extreme dimensionality of biochemical networks, throughput from exascale computing will be necessary to perform the parameter estimation and optimization of the generated metabolic model. This could lead to the discovery of significant genes that encode proteins that might be useful in biomanufacturing and other industries. Genomics and metagenomics are just two of the many -omics fields that will benefit from exascale resources. In each field, significant discoveries are likely hidden within the datasets. Exascale computing will be key to accessing these discoveries, potentially creating entirely new fields of study.

Another source of biological big data stems from continually improving imaging modalities. This includes both medical image sources such as MRI and CT, as well as those employed in life sciences research efforts. Emerging methods such as serial blockface scanning electron microscopy (EM), where the face of a resin embedded tissue block is sequentially imaged using a scanning electron microscope as thin veneers of tissue are removed from the face using an ultramicrotome, will provide unprecedentedly high-resolution 3D views into the ultrastructure of tissue samples[10]. With automation, in a single run, serial block EM machines can collect data for days at a time resulting in image datasets that are each petabytes in size. Once these datasets are developed, the myriad tissue components in the images must be "segmented" (that is, traced and labeled) for subsequent analysis (see Figures 2.1(a)–(d)). Owing to the huge number of collected images, human-driven manual segmentation will be impossible for future datasets. Computational biologists will need to develop methods, likely similar to deep-learning-based computer vision algorithms, to automatically segment tissue features. Furthermore, as the throughput and resolution of imaging increases, our ability to move and interact with the image data will be reduced. To minimize movement of data, imaging instrumentation can be directly connected to upcoming HPC resources for both processing and visualization. Exascale computers will allow biologists to use large and complex neural network architectures, which require high memory to process the data with minimal down sampling.

In addition to the benefits of increased computational throughput and parallelism, biological big data will also benefit from the many other architectural improvements necessary to support exascale. Improvements to memory architectures and interconnects might lead to faster data accessibility. When coupled with the increased throughput, exascale technology will support new advanced interactive analysis for large data sets (for example, in the image data segmentation and refinement example given above).

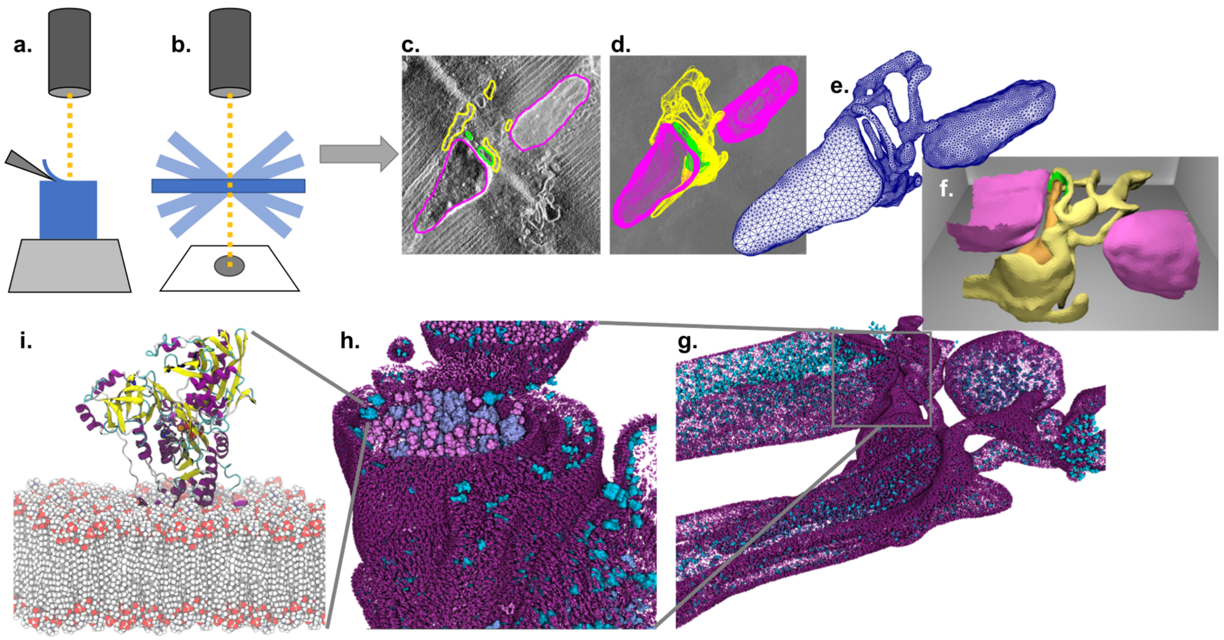


Figure 2.1: Exascale computing will aid the process of bringing to life the multiscale modeling of complex molecular scenes. Starting from volume microscopy modalities such as (a) serial block-face scanning electron microscopy and (b) electron tomography, biologists can gain unparalleled views into the cellular ultrastructure. (c) Biological features in the collected images are segmented and visualized as (d) stacks of contours. (e) Using mesh generation algorithms, computable representations of the geometries can be generated. (f) Boundaries of a scene are marked to represent specific boundary conditions. (g) Using cellPACK, the mesh geometry can be filled with molecular-level details. (h) Zooming in, many different diverse membrane and cytosolic proteins interact within this subcellular space. (i) Individual membrane and cytosolic proteins can be modeled using atomistic techniques such as molecular dynamics and Brownian dynamics to derive relevant model parameters. The computed parameters from these isolated models can be integrated up into models of the cellular scene.

Additionally, new workflows and strategies for data handling and provenance tracking will help address the data storage crisis many labs are facing today. As datasets grow in size, biologists might turn to on-the-fly processing, similar to what is done in the high-energy physics community, to extract parameters of interest during runtime, discarding raw data instead of saving it for postprocessing.

2.4 Supercharging Biomolecular Simulations

As we uncover new protein sequences, there will be a growing need to determine their structures. Protein structures are helpful to biologists for inferring protein function as well as for predicting protein-protein interactions that are critical for system feedback and regulation. Moreover, structures are integral for modern structure-based drug discovery efforts. Due to the extreme cost and difficulty of experimental structural genomics pipelines, computational methods for predicting protein folds can be used as a supplement. Protein structure prediction methods such as Rosetta use a Monte Carlo algorithm to generate folding configurations that are then evaluated for energetic stability[11]. Because proteins are long polymers of amino acids, each with several degrees of freedom, the fold space of proteins is astronomical. Many computations are necessary to enumerate enough folding configurations to generate reliable structure predictions. Large-scale efforts for automated structure prediction, such as the Human Proteome Project and Microbiome Immunity Project driven by IBM's World Community Grid ($\sim$1 PFLOPS), are already ongoing. Protein structure prediction methods tend to be CPU intense with limited storage and I/O requirements. Exascale computers, along with other compute modalities, can provide the throughput necessary to keep pace with the incoming surge of new sequences to fold.

With the maturation of protein structure prediction methods, the engineering and design of completely new (de novo) functional proteins should be possible. Since proteins are naturally "designed" through evolution, and evolution proceeds by incremental mutation and selection, naturally occurring proteins do not span sequence space. By inferring the timeline of mutation acquisition, scientists can actually trace back the evolution of protein families. There remains a huge set of protein sequences that are not represented by these families[11]. Using computational protein structure prediction methods, scientists can sample this unknown space of protein sequences and structures, which might hold solutions to challenges in modern biology, medicine, and nanotechnology. Again, the concurrency afforded by exascale resources in tandem with other compute modalities will allow biologists to enumerate new protein sequences, increasing innovation in de novo protein design and decrease time to market for discoveries.

Exascale will also benefit other modeling efforts across all scales in computational biology and chemistry. One method, virtual screening, predicts the binding strengths and orientations of libraries of candidate drug molecules to targets of interest. With increased parallelism, researchers will be able to dock increasingly large and diverse sets of compounds, accelerating drug discovery. For applications such as biomolecular simulations, which typically integrate sets of differential equations to generate trajectories of how systems move in time, exascale computing enables the generation of longer simulations and more replicates, improving statistics. One popular method, molecular dynamics (MD), is used to classically simulate atomistic models of macromolecules. MD is often used to identify biological mechanisms of enzyme function, reveal druggable cryptic pockets, or provide computational assays to predict biophysical properties such as drug binding[12–14]. To obtain sufficient MD statistics, scientists model extremely idealized versions of biological systems. By introducing additional model complexity, the amount of sampling must also increase. In most cases, conventional Boltzmann sampling will be too inefficient. Instead, biologists will turn to interconnected ensemble-based methods to increase the simulation's coverage. Concurrency provided by exascale computing will allow for the development of superior ensemble methods, enabling biologists to sample additional complexity.

Using these methods, scientists will be able to increase the model detail to study the ramifications of glycosylations and other post-translational modifications as well as the impact of physiologically relevant bilayer compositions or polarizable water molecules on drug binding, for example. Moreover, with exascale computing, biologists will be able to move from simulating the dynamics of individual proteins toward the modeling of many proteins or macromolecular complexes as they exist in vivo. Owing to the weak scaling properties of MD, with the appropriate exascale machines and improvements in algorithms, simulations of up to billions of atoms will become tractable. Such efforts are aided by new data integration frameworks such as cellPACK[15] that are able to build molecular scale models of complex biological scenes. Exascale computing architectures will play an indispensable role in our ability to "bring these scenes to life" with physics-based simulation (see Figures 2.1(e)–(i)).

2.5 Big Data, Multiscale Modeling, and Systems Biology

The ultimate challenge of computational biology is the ability to predict how an organism will behave in response to some stimulus. With sufficiently accurate gene annotations, it will be possible to reconstruct and simulate a whole microorganism's metabolic network. With such a model, scientists will be able to predict the conditions at which an organism can proliferate. This will require repeated simulation of the model while scanning various environmental parameters for both benchmarking and subsequent prediction following a new perturbation. Owing to the large number of metabolic processes in a single organism (including, for example, influences of microorganisms that might reside within a larger organism, similar to the gut microbiome within humans) this task will clearly necessitate the use of exascale (and beyond) resources. Through these metabolic modeling efforts, we might discover how to harness organisms to perform difficult jobs such as detoxifying hazardous environmental waste.

Other biological predictions will require more complicated modeling approaches. For example, consider the drug design challenge in which computational biologists seek to maximize a compound's therapeutic effect while minimizing side effects[16]. At the smallest scales, quantum mechanical (QM), electronic structure methods can predict the electronic state of electrons and nuclei in molecules. This, at least in theory, provides a strategy for computing all molecular properties with high accuracy, allowing researchers to investigate chemical reactivity. However, due to the limits of computational tractability and the enormous spatiotemporal span of biological problems, no single method is suitable for describing all phenomena. QM calculations are tractable for only tens to thousands of atoms depending upon the level of theory modeled. When faced with the challenge of optimizing the reactivity and specificity of covalent enzyme inhibitors or predicting the rate of enzymatic drug degradation, the drug-enzyme system alone might be hundreds of thousands of atoms in size, far beyond the practical size limitations of QM. To achieve sufficient system sampling, researchers must use other methods, such as the aforementioned MD. Notably, MD employs semi-empirical molecular mechanics (MM) force fields that classically represent molecules, neglecting the electronic structure, and cannot probe chemical reactivity. To resolve this, researchers have developed multiscale hybrid QM/MM approaches in which small regions of high importance (for example, catalytic regions) are modeled using QM while the remainder is represented using MM[17]. This kind of approach can bring, for example, a representation of chemical reactivity to MD models of complex subcellular scenes. Concurrency at the exascale will be necessary to efficiently couple the different simulation modalities into one integrated simulation.

Multiscale methods, which couple different levels of theory, provide adaptive simulation resolution while balancing practicality. To successfully model biological processes spanning from the atomic to cellular, organ, and, ultimately, organism scales many multiscale coupling schemes must be developed. Returning to our drug discovery example, it is often important to model the binding rates of drugs to proteins. This can be computed using a hybrid MD and Brownian dynamics approach[18, 19]. Similarly the systemic distribution of drug in the bloodstream can be modeled using computational fluid dynamics

(CFD). The rates of increase and decrease in circulating drug concentration due to gastric and cellular absorption respectively can be computed using detailed MD simulations to describe the permeation of drugs across bilayers. MD derived rates can then be propagated into systemic CFD models improving the realism of our model. Again, it is evident that, due to the large number of model subsystems that require tight integration along with the need to execute parameter scans for optimizing unknowns, the concurrency provided by exascale will be required to efficiently couple future multiscale simulations.

Exascale computing will also pave the way toward personalized medicine. As medical imaging modalities improve, there are other opportunities, such as in cardiac[20, 21] or cancer[22] modeling, where realistic patient geometries can be coupled with emerging multiscale systems models, described previously, to predict personalized results[21]. Let us consider the case of predicting potential cardiac drug interactions: first, from a patient's genetic sequence we can uncover the mutations to key ion channels responsible for maintaining normal cardiac rhythm. Using molecular modeling, we can study the effect of an introduced drug on the behavior of each channel. The drug-induced perturbations of channel behavior can then be coupled to a cellular action potential model. From an MRI image, we can obtain a 3D picture of the patient's heart geometry. Using algorithms from computer vision and machine learning, we can segment the image and create a computer model of the patient's heart. Propagating results from the cellular action potential model to a tissue model applied on the patient's heart geometry, personalized predictions of drug-cardiac interactions can be realized. Critically, the computational throughput of exascale computing will be necessary to produce predictions such as this in a diagnostically relevant timeframe. By combining structural datasets and experimental measurements with various modeling regimes, scientists can construct new models to predict system-level changes to small perturbations for all manner of problems.

2.6 Conclusion

In the coming years, discoveries in biology will be driven by the convergence of improved and increasing experimental data, algorithmic advances, and computational power. Exascale computing will be a necessary development to support the ongoing research efforts of computational biologists. From keeping pace with the rate of experimental data generation to enabling new multiscale multiphysics models of complex biological processes, computational science will continue to evolve and, ultimately, transform biology from an observational to a quantitative science. Continued investments in computing infrastructure will speed up the rate of discovery for life science researchers, thus accelerating discoveries with huge potential impacts in diverse markets and creating new opportunities for improved human health and well-being.

2.7 Acknowledgements

Chapter 2, in full, is a modified reprint of the material as it appears in “Lee, C. T.; Amaro, R. E. *Exascale Computing: A New Dawn for Computational Biology.* Comput. Sci. Eng. 2018, 20 (5), 18–25.”. The dissertation author was the primary investigator and author of this paper.

The authors would like to thank Bryn C. Taylor for critical comments, and Sophia P. Hirakis for assistance in figure preparation. Funding and support was provided by the National Science Foundation through XSEDE supercomputer resources provided via TG-CHE060073N to REA. Funding and support from the National Biomedical Computation Resource was provided through NIH P41 GM103426. CTL was supported in part by a NIH Molecular Biophysics Training Grant T32 GM008326.

References

(1) Berman, H. M. *Nucleic Acids Res.* **2000**, *28*, 235–242.

(2) International Human Genome Sequencing Consortium *Nature* **2004**, *431*, 931–945.

(3) Callaway, E. *Nature* **2015**, *525*, 172–174.

(4) Kaiser, J. *Science (80-.).* **2016**, DOI: 10.1126/science.aaf4108.

(5) Pennisi, E. *Science (80-.).* **2017**, DOI: 10.1126/science.aal0824.

(6) Normile, D. *Science (80-.).* **2017**, DOI: 10.1126/science.aan7165.

(7) Nystedt, B.; Street, N. R.; Wetterbom, A.; Zuccolo, A.; Lin, Y.-C.; Scofield, D. G.; Vezzi, F.; Delhomme, N.; Giacomello, S.; Alexeyenko, A.; Vicedomini, R.; Sahlin, K.; Sherwood, E.; Elfstrand, M.; Gramzow, L.; Holmberg, K.; Hällman, J.; Keech, O.; Klasson, L.; Koriabine, M.; Kucukoglu, M.; Käller, M.; Luthman, J.; Lysholm, F.; Niittylä, T.; Olson, Å.; Rilakovic, N.; Ritland, C.; Rosselló, J. A.; Sena, J.; Svensson, T.; Talavera-López, C.; Theißen, G.; Tuominen, H.; Vanneste, K.; Wu, Z.-Q.; Zhang, B.; Zerbe, P.; Arvestad, L.; Bhalerao, R.; Bohlmann, J.; Bousquet, J.; Garcia Gil, R.; Hvidsten, T. R.; de Jong, P.; MacKay, J.; Morgante, M.; Ritland, K.; Sundberg, B.; Lee Thompson, S.; Van de Peer, Y.; Andersson, B.; Nilsson, O.; Ingvarsson, P. K.; Lundeberg, J.; Jansson, S. *Nature* **2013**, *497*, 579–584.

(8) Spjuth, O.; Bongcam-Rudloff, E.; Dahlberg, J.; Dahlö, M.; Kallio, A.; Pireddu, L.; Vezzi, F.; Korpelainen, E. *Gigascience* **2016**, *5*, 26.

(9) O'Driscoll, A.; Daugelaite, J.; Sleator, R. D. *J. Biomed. Inform.* **2013**, *46*, 774–781.

(10) Denk, W.; Horstmann, H. *PLoS Biol.* **2004**, *2*, ed. by Kristen M. Harris, e329.

(11) Huang, P.-S.; Boyken, S. E.; Baker, D. *Nature* **2016**, *537*, 320–327.

(12) Liu, X.; Shi, D.; Zhou, S.; Liu, H.; Liu, H.; Yao, X. *Expert Opin. Drug Discov.* **2018**, *13*, 23–37.

(13) Durrant, J. D.; McCammon, J. A. *BMC Biol.* **2011**, *9*, 71.

(14) De Vivo, M.; Masetti, M.; Bottegoni, G.; Cavalli, A. *J. Med. Chem.* **2016**, *59*, 4035–4061.

(15) Johnson, G. T.; Autin, L.; Al-Alusi, M.; Goodsell, D. S.; Sanner, M. F.; Olson, A. J. *Nat. Methods* **2015**, *12*, 85–91.

(16) Amaro, R. E.; Mulholland, A. J. *Nat Rev Chem* **2018**, *2*, 0148.

(17) Van der Kamp, M. W.; Mulholland, A. J. *Biochemistry* **2013**, *52*, 2708–2728.

(18) Votapka, L. W.; Amaro, R. E. *PLoS Comput. Biol.* **2015**, *11*, e1004381.

(19) Votapka, L. W.; Jagger, B. R.; Heyneman, A. L.; Amaro, R. E. *J. Phys. Chem. B* **2017**, *121*, 3597–3606.

(20) Kayvanpour, E.; Mansi, T.; Sedaghat-Hamedani, F.; Amr, A.; Neumann, D.; Georgescu, B.; Seegerer, P.; Kamen, A.; Haas, J.; Frese, K. S.; Irawati, M.; Wirsz, E.; King, V.; Buss, S.; Mereles, D.; Zitron, E.; Keller, A.; Katus, H. A.; Comaniciu, D.; Meder, B. *PLoS One* **2015**, *10*, ed. by Berger, T., e0134869.

(21) Winslow, R. L.; Trayanova, N.; Geman, D.; Miller, M. I. *Sci. Transl. Med.* **2012**, *4*, 158rv11–158rv11.

(22) Deisboeck, T. S.; Wang, Z.; Macklin, P.; Cristini, V. *Annu. Rev. Biomed. Eng.* **2011**, *13*, 127–155.

Chapter 3

Simulation-Based Approaches for Determining Membrane Permeability of Small Compounds

3.1 Abstract

Predicting the rate of nonfacilitated permeation of solutes across lipid bilayers is important to drug design, toxicology, and signaling. These rates can be estimated using molecular dynamics simulations combined with the inhomogeneous solubility-diffusion model, which requires calculation of the potential of mean force and position-dependent diffusivity of the solute along the transmembrane axis. In this paper, we assess the efficiency and accuracy of several methods for the calculation of the permeability of a model DMPC bilayer to urea, benzoic acid, and codeine. We compare umbrella sampling, replica exchange umbrella sampling, adaptive biasing forces, and multiple-walker adaptive biasing forces for the calculation of the transmembrane PMF. No definitive advantage for any of these methods in their ability to predict the permeability coefficient P_m was found, provided that a sufficiently long equilibration is performed. For diffusivities, a Bayesian inference method was compared to a Generalized Langevin method, both being sensitive to chosen parameters and the slow relaxation of membrane defects. Agreement within

1.5 log units of the computed P_m with experiment is found for all permeants and methods. Remaining discrepancies can likely be attributed to limitations of the force field as well as slowly relaxing collective movements within the lipid environment. Numerical calculations based on model profiles show that P_m can be reliably estimated from only a few data points, leading to recommendations for calculating P_m from simulations.

3.2 Introduction

Cells are the basic unit of life. An essential feature of cells is their encapsulating phospholipid membrane. Due to the hydrophobic effect [1, 2], individual phospholipids do not diffuse and tumble randomly. Instead they form a bilayer structure with polar phosphate head groups on each side, facing the bulk solvent, with the apolar lipid tails forming a hydrophobic slab in between. This densely packed structure serves two primary purposes: (1) to contain and protect cellular machinery from the harsh external environment, and (2) to maintain ionic gradients to later harvest as energy. Lipid bilayers are an effective barrier to passive diffusion of ions and hydrophilic small molecules, such as carbohydrates, but many molecules can permeate bilayers through passive diffusion at rates that depend on bilayer composition and properties of the permeating solute. The semi-permeable nature of the membrane results in an effective "selectivity," where small apolar compounds can cross the membrane at appreciable rates. In contrast to transmembrane ion channels and transporters that are carefully controlled by the cell, passive selectivity is not actively regulated, but instead arises intrinsically from the forces and fluctuations present across the membrane environment. Despite the enormous importance of passive permeability to basic cell function, a detailed mechanistic understanding of this phenomenon has yet to be achieved.

Estimation of passive permeation rates is of key importance, primarily for the delivery of candidate drugs to intracellular targets, as well as for later excretion of metabolites. For example, in 1991, $\sim$40% of all attrition of drug candidates was related to adverse pharmacokinetic (PK) and bioavailability results [3]. PK attrition rates have since been reduced to about 10% [4] primarily by high-throughput experimental measures of permeability such as the parallel artificial membrane permeability assay (PAMPA) [5, 6] and the cell-based CaCo-2 assay [7, 8]. Although these empirical methods have become a mainstay in industry, they provide little to no insight into the biophysics of membrane permeation. To gain rational insight, assay results can be used to inform linear response models such as the quantitative structure permeability relationship (QSPR) [9, 10]. Due to the nature of training models, QSPR exhibits mediocre predictive performance when compared across a broad range of experimental test sets [11, 12]. Despite advances in these technologies, neither experimental nor QSPR methods provide detailed atomistic insight into the permeation process.

To gain atomistic insight into the passive permeability process, physics-based methods, such as molecular dynamics (MD), have become increasingly popular. While the application of MD to passive permeability is alluring, broad adoption of the method is limited by several major outstanding challenges. First, there is much debate on the ability of current force fields to reproduce system thermodynamics and kinetics correctly [13–15]. Second, the computational and human time burden is large; calculating the permeability of individual compounds can require thousands of CPU-years and months of work by an experienced researcher. Thus, in order to bring MD-based methods for passive permeability into broad practice, systematic studies addressing force field accuracy and computational efficiency of potential methods are warranted. Given the plethora of new experimental results, compound permeability is an ideal benchmark system for both computational free-energy and kinetics calculations that exist.

Passive membrane permeability has traditionally been studied using the homogeneous solubility-diffusivity model [16]. Later work incorporating the heterogeneous nature of lipid bilayers led to the development of the inhomogeneous solubility-diffusion model [17, 18]. The inhomogeneous solubility model is derived from the steady-state flux and assumes equilibrium across the membrane. Mathematically the potential of mean force (PMF), $W(z)$, and local diffusivity coefficient, $D(z)$, are related to the resistivity, R, and permeability, P, via the equation

$$R = \frac{1}{P} = \int_{z_1}^{z_2} \frac{\exp[\beta W(z)]}{D(z)} dz, \tag{3.1}$$

where β is the thermodynamic beta ($\beta = 1/k_B T$), and z is a collective variable describing the relative position of the solute along the transmembrane axis. The integration bounds, z_1 and z_2, are points along this axis on opposing sides of the membrane. Both $W(z)$ and $D(z)$ can be estimated from MD simulations, provided that all z's are well sampled. Due to the nature of Boltzmann sampling, conventional MD is not ideal for sampling rare transition states. In order to obtain sufficient sampling of transition states, various importance sampling techniques have been developed. Some examples include umbrella sampling (US) [19], adaptive biasing force (ABF) [20–23], metadynamics [24], and the Wang–Landau algorithm [25]. In general, many methods can be implemented with multiple replicas (RE) or walkers (MW) to further enhance sampling [26].

MD simulations have long been applied to study the mechanisms and rates of permeability; for a review please refer to Ref. 27. Many works have used various forms of US [12, 28–41], adaptive biasing force [42, 43], metadynamics [44], and kinetic master equations [45]. Passive membrane permeability can also be calculated without the use of the solubility-diffusion equation via methods such as milestoning or directional milestoning [46–49]. For example, milestoning has been used to determine the permeability of water as well as blocked tryptophan [50, 51].

While the aforementioned studies have examined the membrane permeability of individual solutes, there has not yet been a systematic examination of what the most effective method is for calculating $W(z)$ and $D(z)$. In particular, accurate calculations of $W(z)$ are difficult because of slowly converging orthogonal degrees of freedom, namely those related to membrane distortion and relaxation [52, 53]. Umbrella sampling (US) is the canonical methodological approach, in which the solute is restrained at regular intervals along z. The effect of the restraints are analytically removed from the probability distributions calculated from the US simulations. These distributions are then combined into a single PMF covering the complete interval of the coordinate, employing a post-treatment analysis, e.g., the weighted histogram analysis method. Replica-exchange US improves the sampling ergodicity of US simulations by attempting periodic exchanges between neighboring replicas. Conversely, the adaptive biasing force (ABF) algorithm adjusts the biasing force over time to sample the coordinate uniformly, whereas multiple walker ABF (MW-ABF) extends this further by spawning additional concurrent simulations in poorly explored regions of the coordinate.

In the present work, we systematically compare four methods: US, REUS, ABF, and MW-ABF, by calculating the permeability of urea, benzoic acid and codeine through a DMPC bilayer.

3.3 Results and Discussion

Below, we report the membrane permeability coefficients (P_m) of urea, benzoic acid, and codeine computed with four different methods, namely, US, REUS, ABF and MW-ABF. We also present a detailed analysis of two methods for the computation of diffusivity, namely, a Bayesian inference method and a Generalized Langevin method. While we did not test metadynamics, another common free-energy method, it has recently been favorably compared with US for water-membrane partitioning, nonetheless while suffering the same shortcomings [54]. Initial states, i.e., positions of the permeant within the bilayer, were generated using 100-ns steered MD simulations [55, 56] in which the permeant is pulled from one side of the membrane to the other (see Methods).

The three permeants studied here, shown in Fig.3.1, are chosen for their diverse chemical properties: their molecular weights range from 60 g/mol (urea) to 299 g/mol (codeine); their hydrophilicity, as measured by the octanol:water partition coefficient, differs by over two orders of magnitude; and most importantly, the experimentally determined P_m of the three permeants span five orders of magnitude (Table 1), making them a relatively robust test set for P_m calculation via different methods. Of the three permeants, benzoic acid is the only permeant that exhibits a formal charge at pH=7. Nonetheless, only its neutral form is considered in our calculation. This treatment is consistent with the corresponding experimental protocol [57], where fluxes at several pH values are measured to determine the 'intrinsic permeability' corresponding to the un-ionized form of a given molecule. The underlying assumption of such a protocol, i.e., only the non-ionized form of benzoic acid contributes significantly to P_m, has been confirmed experimentally [57].

3.3.1 Computed vs. experimental P_m

The computed logP_m of the three permeants are listed in Table 1, along with the corresponding experimental references obtained from egg phosphatidylcholine bilayers (urea [58], codeine [59] and benzoic acid [57]). From Table 1, it is clear that the majority of computed P_m values exceed the corresponding experimental data by 1–2 orders of magnitude. For comparison, we define $\Delta\log P_m = \log P_m^{\mathrm{com}} - \log P_m^{\mathrm{exp}}$, where $\log P_m^{\mathrm{com}}$ and $\log P_m^{\mathrm{exp}}$ are the computed and experimental logP_m, respectively. Negative $\Delta\log P_m$ values are only observed for urea, the most hydrophilic of the three permeants. Results obtained with different

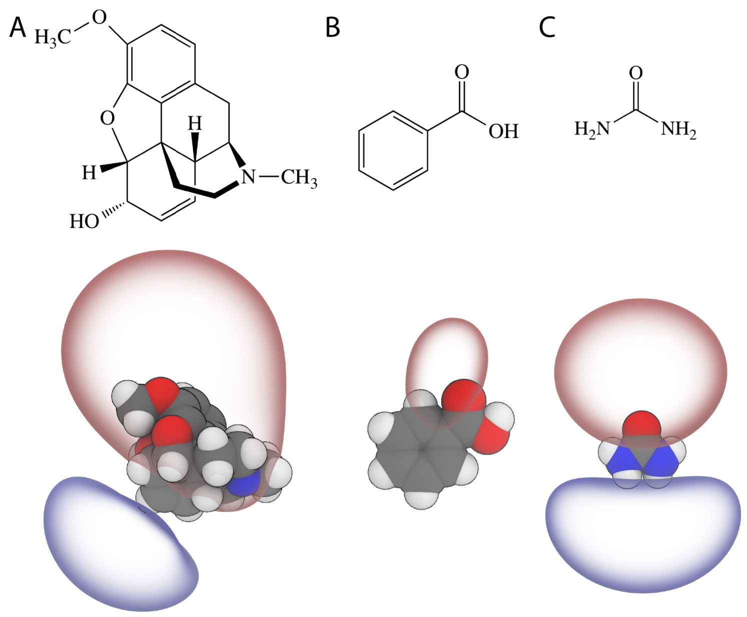


Figure 3.1: Three permeants tested shown as 2D schematics (top) and 3D structures rendered with the ±5 kT/e electrostatic potential isosurfaces using the assigned CGenFF charges (bottom). (A) codeine. (B) Benzoic acid (neutral). (C) Urea.

methods also show the largest discrepancy for urea: the US and REUS methods underestimate $\log P_m$ by 0.87 and 0.51, whereas ABF and MW-ABF overestimate it by 0.71 and 1.14, respectively. For benzoic acid and codeine, all the methods overestimate $\log P_m$ by 0.71 to 1.67.

Table 3.1: $\log P_m$ of the three permeants examined in this study. The unit of P_m is cm/s. Experimental values are obtained from 59, 58 and 57. For each computed $\log P_m$, the length of the simulation used in the computation is shown in parenthesis.

	Urea	Benzoic Acid	Codeine
Experiment	-5.4	-0.26	-0.85
US	-6.27 (2.8 μs)	0.45 (1.4 μs)	0.03 (1.4 μs)
REUS	-5.91 (4.3 μs)	1.17 (1.4 μs)	0.64 (1.4 μs)
ABF	-4.69 (0.9 μs)	1.16 (0.9 μs)	0.82 (2.7 μs)
MW-ABF	-4.26 (1.4 μs)	0.81 (0.7 μs)	0.18 (1.1 μs)

Fig. 3.2 also reveals that with the possible exception of urea, increasing simulation time does not significantly improve agreement with experiment. Even for urea, the computed P_m tends to plateau once the total simulation time exceeds 1 μs. It is unlikely that P_m has converged, however, but requires much

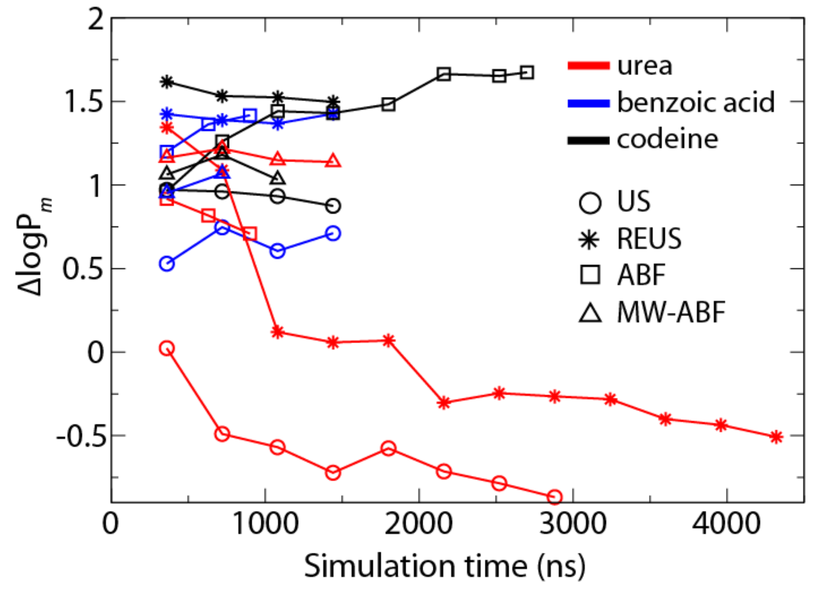


Figure 3.2: $\Delta \log P_m$ of urea (red), benzoic acid (blue) and codeine (black) as a function of simulation time. Results obtained with different methods are indicated using the following symbols: US (circle), REUS (star), ABF (square), MW-ABF (triangle).

longer time scales (milliseconds) to sample very slow membrane reorganization processes [52].

3.3.2 Potentials of Mean Force

In the solubility-diffusion model, the PMF is a critical component of the permeability (see Eq. 3.1). Given its exponential weighting, the PMF may even be considered the greatest contributor to the permeability, making its accurate calculation of paramount importance. To determine the best approach for calculating the PMF, we have compared four methods mentioned earlier and investigated the appropriate balance between equilibration time and sampling time.

The PMFs were determined by bringing the permeant across the entire membrane and then symmetrizing the resulting profile, *i.e.*, the profiles were adjusted to be identical on either side of the membrane center and to both start and end at 0. The resulting *symmetrized* PMFs for all three permeants are shown in Fig. 3.3A-C. Each plot shows varying levels of disagreement between the methods, although no method is consistently different. For example, while for codeine, ABF and US are in agreement and MW-ABF is the most discrepant, for benzoic acid, MW-ABF and US are in agreement but different from REUS and ABF. Thus, it is not apparent from these three permeants that any one method for determining

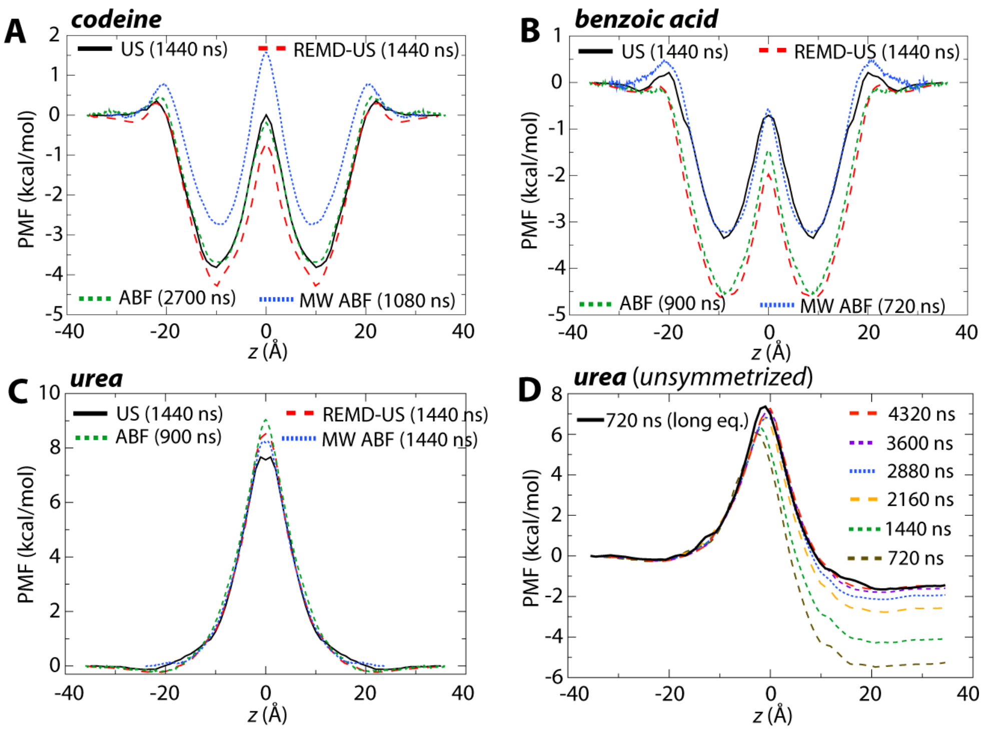


Figure 3.3: Potentials of Mean Force (PMFs) for the permeants (A) codeine, (B) benzoic acid, and (C) urea. In each case, four PMFs from each of the methods are given: US (black, solid), REUS (red, large dashes), ABF (green, medium dashes) and MW-ABF (blue, small dashes). All PMFs have been symmetrized about the membrane center. (D) Convergence of the unsymmetrized PMF for urea. PMFs determined using REUS for a total simulation time of 720 ns – 4.3 μs are shown as various colored, dashed lines. The PMF determined after 500 ns of total equilibration and 720 ns of sampling is also shown (solid black line) and is comparable to that from 4.3 μs of sampling with less equilibration. See Fig. 3.7 for the unsymmetrized PMFs for codeine and benzoic acid.

the PMF converges more rapidly than the others.

The original profiles, prior to symmetrization, reveal a disturbing degree of asymmetry, and therefore accumulated error. The evolving *unsymmetrized* profiles for urea are shown in Fig. 3.3D and for codeine and benzoic acid in Fig. 3.7. After 720 ns (10 ns/window), the asymmetry between the two end points is $\sim$5 kcal/mol. This asymmetry decreases at a rate of only 1 kcal/mol per 720 ns, and still persists at $\sim$1.5 kcal/mol after over 4 μs of sampling. Trajectory snapshots from US of codeine bowing the interfacial phosphates as well as urea coordinating waters at the membrane core are shown in Fig. 3.8 and S3, respectively, highlighting potentially slow converging orthogonal degrees of freedom.

One reason for the growing error in the PMF as the permeant goes through the membrane is the residual disturbance to the membrane structure from the steered MD used to generate the starting states. The time blocked probability distributions of headgroup phosphates and water in the vicinity of urea (shown in Fig. 3.10) support the existence of non-equilibrium artifacts. One way to ameliorate this disturbance is longer equilibration of the starting states. As a test of this idea, selected starting states for urea were equilibrated for a particular amount of time, dependent on their distance from the membrane center, namely 100 ns at the center; 50 ns at $\pm$5 Å, $\pm$10 Å, and $\pm$15 Å; 25 ns at $\pm$20 Å; 15 ns at $\pm$25 Å; and 10 ns at $\pm$30 Å, giving a total of 500 ns of equilibration. Intermediate windows at each ångstrom were then generated from these equilibrated states. The unsymmetrized PMF resulting from 720 ns of sampling with REUS is shown as the solid black line in Fig. 3.3D. It is immediately apparent that this newly generated PMF and its asymmetry after only 720 ns are comparable to those after 4.3 μs of sampling without sufficient equilibration. Thus, starting states should be sufficiently equilibrated in order to avoid artifacts resulting from their initial setup, in agreement with earlier recommendations by Paloncýová et al. [38].

In the preceding calculations, we determined the PMF for the entire permeation process, i.e., from $z = 36$ Å to -36 Å, after which the resulting PMF was symmetrized about the membrane center. However, noting the expectation of a symmetric PMF, for many published cases the PMF is only determined from $z = 36$ to 0 Å [31, 60, 61]. Given a finite amount of time for sampling, one may ask which is better — to sample the full range of permeation for time t or to sample half of the range for time $2t$ and then mirror it across the membrane center? These two possibilities are compared in Fig. 3.11. When considering the *unsymmetrized* PMF from 10 ns/window of the full range (10 ns $\times$ 72 windows), the maximum value of 10.0 kcal/mol is a full 1.8 kcal/mol greater than the (presumably) converged result from 60 ns/window (8.3 kcal/mol). For 20 ns/window over half of the range (20 ns $\times$ 36 windows), the maximum in the PMF is 9.4 kcal/mol, apparently a better result than simulating over the full range. However, once the result over the full range is symmetrized, the peak shifts to 8.0 kcal/mol, significantly closer to the final result than the half-range peak. Thus, the combination of simulating over the entire range along with the requirement of symmetrization appears to produce a more accurate result than simulating over half of the range for twice as long.

3.4 Diffusivities

The second key component of the solubility-diffusion model is the position-dependent diffusivity, $D(z)$. However, some commonly used methods for estimating diffusivity from simulations are not applicable to the heterogeneous membrane environment. One of the most common means of calculating the diffusion coefficient of a solute dissolved in a liquid is the Einstein–Smoluchowski equation. In the one-dimensional long-time limit, this equation relates the diffusivity, $D(z)$ to the mean square deviation of the position of the solute,

$$D(z) = \frac{\langle |z(t) - z(0)|^2 \rangle}{2t}. \tag{3.2}$$

This relationship is only valid for solutes undergoing a random walk in a homogeneous liquid and offers a very poor approximation of the true diffusivity in a membrane, where most solutes encounter free-energy barriers with heights greater than $k_{\mathrm{B}}T$. For similar reasons, estimates based on a Green–Kubo relation of the velocity are expected to be equally poor [62].

Marrink and Berendsen calculated the diffusivity profile for the permeation of water using the force autocorrelation function, [18, 60]

$$D(z) = \frac{(k_{\mathrm{B}}T)^2}{\displaystyle\int_0^\infty \langle \Delta F_z(z,t) \Delta F_z(z,0) \rangle \mathrm{d}t}. \tag{3.3}$$

This method requires that the solute be constrained to a point z on the coordinate, which makes it relatively difficult to apply because the equations of motion of the MD integration must be modified to impose the constraint. [62, 63] As a result, it is far more common to perform simulations where the solute is simply restrained to remain near a given value of z with a biasing potential. As the solubility-diffusion model requires the determination of $W(z)$ and $D(z)$ over the full bilayer, it would be preferable for the method used to provide both of these profiles from one set of simulations.

In the following sections, we present two strategies for calculating $D(z)$ for the permeation of a solute across a lipid bilayer using biased MD simulations. The first is based on the generalized Langevin equation for a harmonic oscillator. The second employs Bayesian inferences on the likelihood of the observed dynamics of the solute.

3.4.1 The Generalized Langevin method

The generalized Langevin equation provides straightforward methods to calculate position dependent diffusion coefficients from restrained MD simulations. In these methods, the solute is restrained by a harmonic potential so that it oscillates at a point along the coordinate. The solute can now be described as a harmonic oscillator undergoing Langevin dynamics, where the remainder of the system serves as the frictional bath for the solute. Implicitly, describing the system as a harmonic oscillator requires the restraining potential to be dominant over the underlying free energy surface, i.e., the latter is effectively a perturbation on the former. Otherwise, the assumption that the bilayer serves only as a frictional bath to the oscillating solute may not be valid. Values of the spring constant sufficiently large to justify this assumption also tend to be too large for umbrella sampling simulations, meaning that it may not be possible to use the same simulation to calculate $W(z)$ and $D(z)$.

Once a time series of the z position of the solute is collected, the diffusion coefficient for that point can be calculated from the position or velocity autocorrelation functions (ACF and VACF, respectively). These methods were originated by Berne and coworkers for the calculation of reaction rates [64], and elaborated by Woolf and Roux to calculate position dependent diffusion coefficients. [65] In their approach, the diffusion coefficient is calculated from the VACF. This approach requires the numerical Laplace transform of the VACF for several values of the transform parameter s and extrapolation to the limit of $s = 0$. Hummer proposed a simpler method to calculate diffusion coefficients from harmonically restrained simulations [66] in which the diffusion coefficient is calculated directly from the integral of the ACF, C_{zz}, of z and the variance of z,

$$D(z = \langle z \rangle) = \frac{\mathrm{var}(z)^2}{\displaystyle\int_0^\infty C_{zz}(t)\,\mathrm{d}t}. \tag{3.4}$$

This method is attractive because it is simple to impose a harmonic restraint on a solute and save a time series of the z-position of this trajectory in most MD codes. It also avoids the need for multiple numerical Laplace transforms of the VACF. The ACF can be calculated directly from this time series, [67]

$$C_{zz}(t) = \langle \delta z(0) \delta z(t) \rangle = \frac{1}{n_{\mathrm{samples}}} \sum_{i=0}^{n_{\mathrm{samples}}} \delta z(i) \delta z(t+i) \tag{3.5}$$

where $\delta z(t) = z(t) - \langle z \rangle$. Our code for calculating the ACF from a NAMD [68] time series is provided in the SI. Although this method is a straightforward way to calculate membrane diffusion coefficient profiles, there are several practical issues associated with its use. We illustrate these issues by presenting the ACFs calculated from a simulation of urea restrained at three positions in the model bilayer system: $z = 0$ Å, $z = 10$ Å, and $z = 36$ Å (see Fig. 3.4).

Correlation functions typically require extensive sampling to achieve convergence, particularly for long correlation times. Heterogeneity of the bilayer environment can cause the ACF calculated from different simulations at the same z-reference value to be significantly discrepant, a particularly serious for hydrogen-bonding solutes that remain partially coordinated by water molecules inside the membrane such as urea (see Fig. S3). This issue can be addressed in part by performing several simulations, with a long equilibration period for each. The correlation functions can then be calculated from long time series, e.g., > 1 ns, collected from each of these simulations.

The general features of the ACFs can be interpreted based on the analytical solution to the autocorrelation function of a harmonic oscillator undergoing Langevin dynamics, [69]

$$C_{zz}(t) = \mathrm{var}(z) e^{-\gamma(\bar{\omega})t/2\mu} [\cos(\Omega t) + \frac{\gamma(\bar{\omega})}{2\mu\Omega} \sin(\Omega t)] \tag{3.6}$$

where γ is the friction coefficient, $\bar{\omega}$ is the renormalized frequency of the oscillator, and μ is its reduced

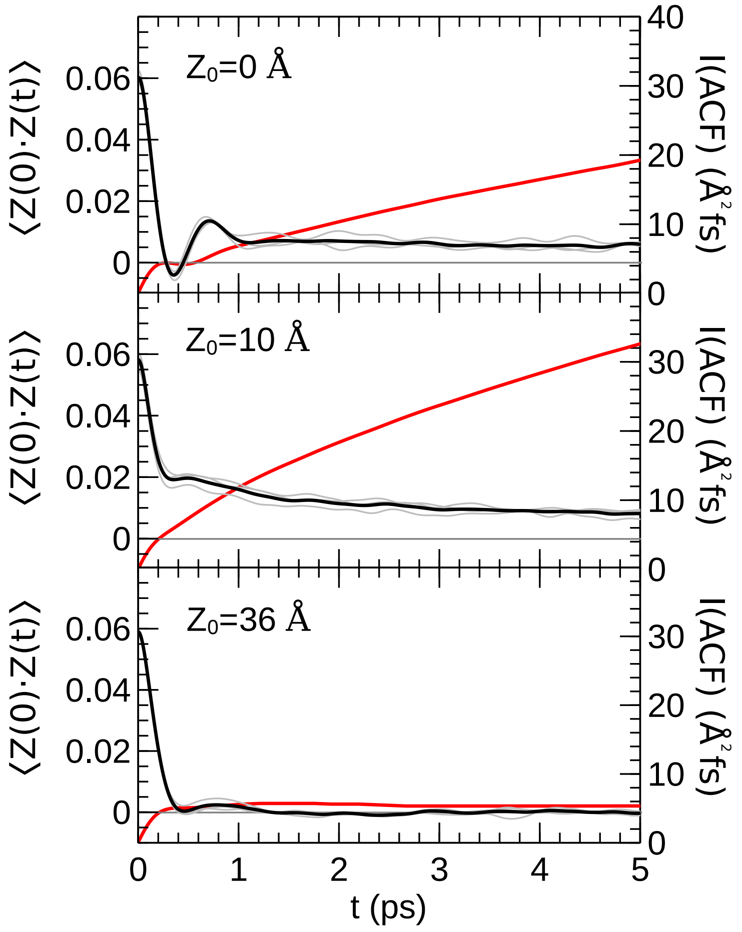


Figure 3.4: Time autocorrelation function of urea in the model DMPC bilayer restrained at various values of z_0 using a harmonic potential ($U_{\mathrm{rest}} = \frac{1}{2}k(z - z_0)^2$, $k = 10$ kcal/(mol Å ^{2})). The black lines are the cumulative ACF calculated from three 1-ns simulations. The ACF from each 1-ns trajectory alone is presented in grey. The red lines (secondary axis) show the integral of the ACF over the interval $[0,t]$.

mass. γ determines the rate of decay of the ACF. The expected behavior of the ACF based on Eq. 3.6 is that for damped periodic oscillations; however, if the friction coefficient, γ, is high, the ACF may decay to zero before there are any significant oscillations. This decay is apparent in the ACF when the solute is restrained at $z = 36$ Å (bulk water; see Fig. 3.4), which decays to zero within 0.5 ps, but then has a short second peak extending to 1.2 ps. The calculation of the diffusion coefficient formally requires the integration of the ACF in Eq. 3.4 over the interval $[0,\infty]$, but the ACF will decay to zero within $\sim$ 2 ps in most fluid environments. Even if extensive sampling has been performed, there can be significant noise in the ACF at long times. Hummer noted that this noise causes the calculated diffusion coefficients to be sensitive to the interval of integration. [66] The g_wham code [70], one of the tools provided with Gromacs, limits the contribution of noise by cutting off the integration when the ACF drops below a threshold value of $0.05 \times \text{var}(z)$. This cutoff is not appropriate in all instances because the ACF can go through multiple oscillations before converging to zero. Nevertheless, it can provide reasonably accurate results if the autocorrelation function decays rapidly (i.e., in a high friction regime).

Compared to bulk water, the autocorrelation functions in the lipid tail region are much slower to converge. The ACFs from simulations with $z_0 = 10$ Å and $z_0 = 0$ Å only decay to 8 and 13% of their initial values in 5 ps, respectively. A consequence of the failure to converge to zero is that the integrated ACF increases almost linearly, causing the calculated diffusion coefficients to be sensitive to the interval over which the ACF is integrated. Typically, this sensitivity will cause the integrated ACF to be larger than it should be, so that the calculated diffusion coefficient is erroneously underestimated.

The slow convergence of the ACF is consistent with the work of Neale et al., [52] which showed that the convergence of US simulations of solute permeation into bilayers can require μs-ms-length simulations. Long correlation times are due to slow diffusion of the lipid tails, variations in the hydration of the solute, and inhomogeneities in the bilayer interface that form over very long time scales. [71] While US simulations of the bilayer PMF can achieve convergence by running for a very long time, the lack of ergodicity in sampling is inconsistent with the underlying model of harmonic oscillator in frictional bath that is used to derive Eq. 3.4.

The issues noted above for calculating diffusivity in the membrane are typically not significant for small solutes like water, [39, 41] but they are severe for larger, more complex ones, such as urea, which are more prone to have long-time correlations. Under such circumstances, the Generalized Langevin method is not appropriate for calculating $D(z)$. To determine if convergence issues are significant for a given system, the ACF should be calculated and plotted for several z-positions in the bilayer. If the ACF has not decayed to values near zero at long time scales (e.g., 5 ps), the Generalized Langevin method is probably not appropriate for calculating the position-dependent diffusivity for that solute.

In the Generalized Langevin approach, a prerequisite noted above is that the force constant of the harmonic restraint should be sufficiently large to render the underlying free-energy surface a small perturbation on the harmonic potential. The force constant used for the US simulations, 1.5 kcal/mol/Å ^{2}, may be too small for application of this approach. To test its applicability, we ran additional 2-ns simulations for each window for urea using a much larger restraining force constant of 10 kcal/mol/Å ^{2}. However, as shown in Fig. 3.12, the differences between the $D(z)$ profiles for the two cases are practically negligible at nearly all points. At a few data points in the membrane interior, urea under a higher restraint gives a diffusivity as much as twice that of urea under a lower one. Yet, because the permeability depends linearly on $D(z)$ (see Eq. 3.1), but exponentially on $W(z)$, even a factor of two in the diffusivity across the entire permeation pathway contributes only $\sim$0.3 to $\log P_m$. Thus, for permeability calculations, it is acceptable to use the same simulations for calculation of both $W(z)$ and $D(z)$.

3.4.2 The Bayesian inference method

A fundamentally different approach for the determination of position-dependent diffusivities employs Bayesian inferences to reconcile thermodynamics and kinetics [66, 72, 73]. This approach is especially suited for calculations within lipid membranes because no assumptions are made regarding the form of the free-energy landscape on which diffusion occurs. Furthermore, such an approach is compatible with a wide variety of biasing schemes used in MD, and thus, allows the practitioner more flexibility in simulation design. This approach may be viewed as the inverse solution of the Smoluchowski equation, which yields consistent estimates of $W(z)$ and $D(z)$, given the trajectory obtained from biased simulations. These biases may also be time-dependent, as is the case for the metadynamics and ABF algorithms. Since the latter free-energy calculation algorithms are designed for the accurate description of $W(z)$, one may use it as a consistency check of the solution provided by the Bayesian inference method. Alternatively,

$W(z)$ as determined by a free-energy algorithm can serve as a prior in the Bayesian scheme, increasing the reliability of $D(z)$ in situations where statistics are poor. In a nutshell, the Bayesian scheme uses as parameters the values of the transition coordinate, z, along the trajectory, together with the force, $F(z,t)$, which is the sum of the time-dependent bias and the intrinsic system force, equal to $-\nabla W(z)$. Under the stringent assumption of a diffusive regime, the motion is propagated using a discretized Brownian integrator,

$$\Delta z = z_2(t) - z_1(t) = \beta D(z_1)F(z_1,t_1)\Delta t + \nabla D(z_1)\Delta t + \sqrt{2D(z_1)\Delta t}\; g(t) \tag{3.7}$$

where $\beta = (k_B T)^{-1}$, $\Delta t = t_2 - t_1$ and $g(t)$ is a Gaussian white noise of zero mean and variance equal to unity. Propagation along the transition coordinate can be recast as the sum of a drift and a noise term, i.e., $\Delta z = \mu + \sigma g(t)$, where μ and σ^2, the variance, are defined by,

$$\begin{cases} \mu & = & \beta D(z_1)F(z_1,t_1)\Delta t + \nabla D(z_1)\Delta t \\ \sigma^2 & = & 2D(z_1)\Delta t \end{cases} \tag{3.8}$$

Propagation of motion obeys a Gaussian-distributed probability of observing the transition from z_1 at time t_1 to z_2 at time t_2,

$$P[(z_2,t_2|z_1,t_1)|F(z,t),D(z)] = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\Delta z - \mu)^2}{2\sigma^2}\right) \tag{3.9}$$

which assumes the system is in the overdamped Langevin dynamics regime and satisfies the fluctuation-dissipation theorem. It is worth noting that in contrast with related schemes, the present formalism, embodied in Eq. 3.7, features a gradient term, $\nabla D(z_1)\Delta t$, which has been shown to improve the accuracy of the predicted intrinsic system force, or gradient of the PMF. See the SI for more details on applying the scheme to MD simulations.

As shown in Fig. 3.4, the motion of permeants within the membrane is complex and correlations in such motion are much longer in the membrane environment than in solutions (at least several picoseconds). These long correlation times complicate calculating the diffusivity by most available methods, including the Bayesian inference method described herein. A crucial component of this scheme is the time step, Δt. The correlations shown in Fig. 3.4 violate the assumptions of overdamped Brownian motion leading to

Eq. 3.7; thus, we found that Δt values of a few picoseconds yield substantial overestimates of the diffusivity. Δt should be chosen to be much larger than any correlation time of the motion; however, there is also an upper bound on Δt because the discretization implicit in Eq. 3.7 requires only small changes in $F(z_1, t_1)$ and $D(z)$ over the duration of Δt. Errors due to the violation of this requirement are apparent when $W(z)$ as predicted by the Bayesian scheme diverges substantially from that obtained by the free-energy calculation technique (here, ABF). For the permeants examined in the present work, we found $\Delta t = 32$ ps still gives consistent $W(z)$ functions while minimizing errors due to correlation.

3.4.3 Comparison of the two approaches

As is clear from Eq. 3.1, $\log P_m$ is much more sensitive to $W(z)$ than to $D(z)$. However, as shown in Fig. 3.5, there are some notable differences in the results of the Generalized Langevin and Bayesian inference approaches, which have an appreciable effect on the predicted permeability. For the four methods used to determine the PMF, the Generalized Langevin approach for calculating $D(z)$ was used for US and REUS, while the Bayesian inference method was used for ABF and MW-ABF. Under the conditions described above, i.e. the restraint strength used in the US and REUS simulations and the Δt chosen for the Bayesian scheme, we obtain smaller diffusivity values using the Generalized Langevin approach than using the Bayesian approach. These differences are most notable near $z = 0$, where the value of $D(z)$ most influences the permeability for urea due to the maximum of $W(z)$. For example, the much larger diffusivity of urea near the center of the membrane as determined by the Bayesian scheme compared to the Generalized Langevin scheme (see Fig. 3.5) results in the ABF and MW-ABF methods having larger $\log P_m$ values by more than an order of magnitude, despite the fact that the height of the free energy barrier calculated by ABF is significantly larger than that for the other methods (see Fig. 3.3). The generally larger $D(z)$ values given by the Bayesian scheme are likely due to the fact that the scheme used in the present work is limited to relatively small values of Δt, a consequence of the Gaussian approximation for the probability profile (Eq. 3.9). We have found that the motion of the permeants is not strictly diffusive at the Δt values used here and that the diffusivity appears to decrease continuously with Δt. We are currently working to improve the Bayesian scheme to overcome the limitations on Δt, which will be addressed in subsequent work. Hence, although it plays a less dramatic role in Eq. 3.1 than $W(z)$, $D(z)$ is also highly influential and the approach to calculating it must be carefully considered.

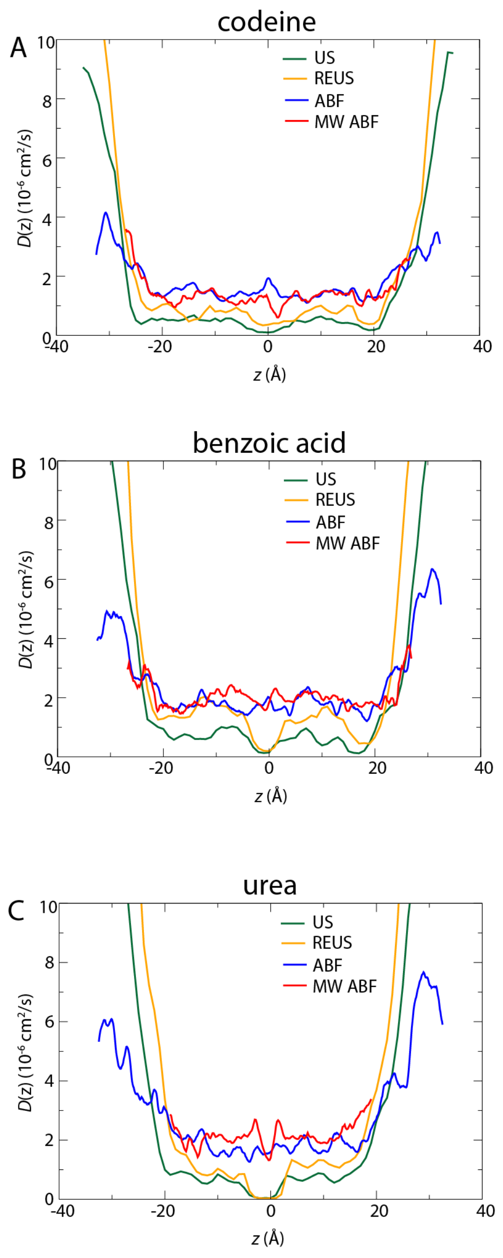


Figure 3.5: Diffusivity profiles for each permeant, (A) codeine, (B) benzoic acid, and (C) urea. In green and orange are the umbrella-based methods, US and REUS, for which the Generalized Langevin approach was used to determine $D(z)$. In blue and red are the ABF-based methods, for which the Bayesian scheme was used.

3.5 Comparison of methods and tolerance to error

Results presented up to this point do not favor any one method over another. For example, while REMD-US is more accurate for urea, it is less so for the other two permeants (see Table 3.1). To examine if the rate of convergence depends on the method employed, we have plotted for each method and permeant $\log P_m$ as a function of simulation time in Fig. 3.2. For codeine and benzoic acid, the variance in $\log P_m$ for a given method is quite small, no more than 0.25 log units, even between 360 ns and almost 3 μs, suggesting that as much as 90% of the simulation time invested was unnecessary. On the other hand, there is a clear effect of time on $\log P_m$ for urea, with it changing by as much as two log units over time. The downward trend for urea's $\log P_m$ is an effect of the shrinking peak in the PMF (Fig. 3.3C). A similar trend for codeine or benzoic acid is likely not observed because the peaks in their PMFs, i.e., those parts above 0 that contribute most significantly to $\log P_m$, are almost non-existent. Regardless, for codeine and benzoic acid, the disagreement with experiment is notable in part because the simulated values are consistently greater by 0.5 to 1.5 log units.

After examining the PMFs and diffusivities in great detail, it is useful to consider their individual contributions to the permeability and, thus, how accurate each of them ought to be to give $\log P_m$ at a chosen level of accuracy. To explore how these two parameters contribute to the permeability, a program to generate arbitrary PMFs and diffusivities was written. This program creates smooth profiles based on input values of the PMF and diffusivity at the interfacial region and at the center of the membrane (see Methods for more details). We first used the program to test molecules with a single barrier (or valley) at the membrane center, varying the barrier's height and width. As seen in Fig. 3.6A, the contribution of the width is negligible, while the height of the barrier dominates $\log P_m$. For positive PMF values at the center, there is roughly a correspondence of 2 kcal/mol to one log unit for $\log P_m$. Changing the diffusivity by a factor of 10 (range of 10^{-6} to 10^{-5} cm^2/s) has a similar effect of one log unit, as expected from the linear dependence of the permeability on $D(z)$ (see Fig. 3.6B). Nearly identical behavior was observed when two barriers were placed at the interfacial region, with only the height and not the position of the barriers contributing (see Fig. 3.13).

While permeants for which the PMF barrier is pronounced have their $\log P_m$ values dominated by

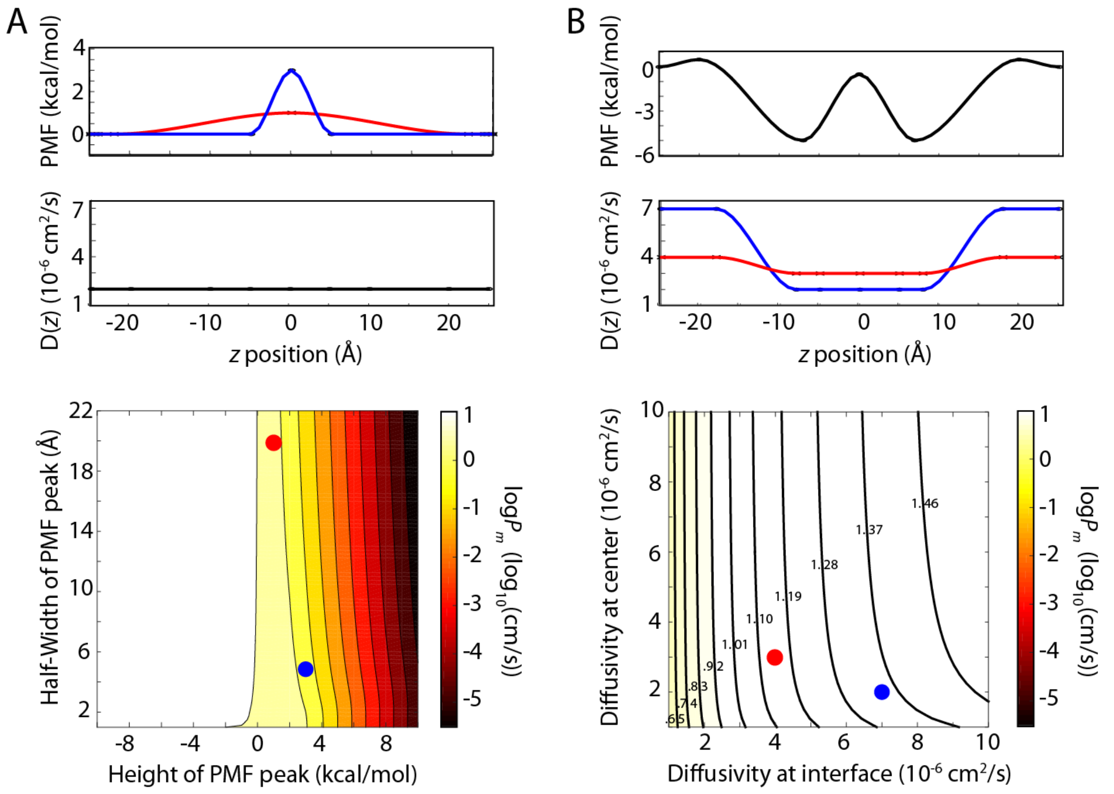


Figure 3.6: logP_m as a function of modeled input. In both parts, the input PMF and $D(z)$ are shown on top and the dependence of logP_m on them on bottom. In each contour plot, the red and blue circles correspond to the red and blue PMF or diffusivity above. (A) logP_m as a function of PMF barrier height and width. $D(z)$ is held constant. (B) logP_m as a function of diffusivity profile. The PMF is held constant. $D(z)$ is varied at $z = \pm 20$ Å (interface) and at $z = 0$ Å (center). The range of logP_m values is 0.55 to 1.55, with contour values given in the figure.

the barrier height, others may exhibit more subtle dependencies. To test this possibility, a PMF in which the barriers are less than 1 kcal/mol at the interfacial regions of the membrane, akin to the PMFs for codeine or benzoic acid, was created and the diffusivity profile varied. As shown in Fig. 3.6B, the diffusivity at the PMF barriers contributes to $\log P_m$, but that at the core (a minimum in the PMF) does not. As a final test of the minimal amount of information necessary to calculate $\log P_m$, we generated PMFs and diffusivities to match those from simulation based solely on the PMF value at the peak(s) and $D(z)$ at that position. Comparison of each generated profile to the computed ones is presented in Fig. 3.14. The obtained $\log P_m$ values are 0.31 (benzoic acid), 0.44 (codeine), and -5.82 (urea). While urea is identical to the value estimated with REUS (see Table 3.1), the other two permeants are slightly below their computed values (difference of 0.85 and 0.63, respectively), although they are closer to experimental values. Regardless, our models demonstrate that $\log P_m$ can be obtained from surprisingly few data points to a high degree of accuracy.

In lieu of determining the full PMF, which can require microseconds of simulation, one can use alternative methods to sample the critical points, i.e., at the interface and/or at the center. As a proof-of-concept, we ran alchemical free energy perturbation (FEP) to calculate the solvation free energy of urea in the membrane core and in water. We found $\Delta G = 9.26$ kcal/mol ($\sim$1–1.5 kcal/mol higher than predicted by our PMFs in Fig. 3.3) and $D(0) = 1.25 \times 10^{-6}$ cm^2/s. When combined with a very simple interpolated PMF curve that goes smoothly to 0 at $z = \pm 20$ Å, we find $\log P_m = -5.26$. This is within < 3% of the experimental value, despite requiring significantly less simulation time: 200 ns for FEP vs. at least 1 μs for the full-PMF approach. However, one loses insight into the details of the permeation process when using FEP over a PMF-based method.

3.6 Conclusions

We have computed the membrane permeability to three compounds using a variety of simulation-based methods, namely US and ABF, along with their multiple-copy variants, REUS and MW-ABF, respectively. These three compounds, codeine, benzoic acid, and urea, span a range of chemical properties and, most importantly, permeabilities (see Table 3.1). All simulation methods were able to predict the permeability within one log unit typically, except in a few cases that were off by 1.5 (see Fig. 3.2). Interestingly, of the four methods tested, none stood out as unequivocally better than the others. The root-mean-square error in log units for each method was 0.821 (US), 1.23 (REUS), 1.33 (ABF), and 1.08 (MW-ABF).

Because of their similar performances, no one simulation method is recommended over another. However, regardless of the method chosen, certain procedures can improve convergence. It was found that simulation over the entire range, i.e., from one side of the membrane to the other, followed by symmetrization of the resulting PMF converged much faster than simulating over just half the range, i.e., from one side to the membrane center (see Fig. 3.11). Additionally, the states used to seed the windows for production sampling should be well equilibrated. Methods such as SMD used to produce these initial states induce significant non-equilibrium perturbations to the membrane that may take hundreds of nanoseconds or more to equilibrate. Alternatives to SMD include building the membrane *de novo* around the permeant at varying depths [74] or introducing the permeant in a perturbative fashion at different values of z. Regardless of the method, equilibration should be carried out for as long as is feasible (50-100 ns/window at least, depending on membrane depth), particularly when the permeant's stability relies upon the spontaneous formation of membrane defects and penetration of water.

Two methods for calculating the diffusivity were explored. The first, more commonly used Generalized Langevin method, based on restraining the solute and measuring the correlation of the system forces acting on it, was combined with the PMFs in the US and REUS; the second, the Bayesian inference method, was combined with PMFs from ABF and MW-ABF. Both methods present a number of subtleties that prevent a straightforward calculation of $D(z)$, e.g., a proper choice of the force constant in the Generalized Langevin method or the discretization time step in Bayesian inference. Furthermore, they are both plagued by long correlation times that are not amenable to the usual MD time scales. Despite these caveats, they produced diffusivity profiles in rough agreement, although those from the Bayesian inference method are consistently slightly higher in the membrane than those from the Generalized Langevin method (see Fig. 3.5).

Because the solubility-diffusion model relies on two position-dependent quantities, the PMF and the diffusivity, it is natural to assume they must be calculated to determine the permeability. However, as demonstrated in the section "Comparison of methods and tolerance to error", only a few data points contribute significantly to it. Therefore, permeability can be estimated from a handful of parameters, namely the values of $W(z)$ and $D(z)$ at the membrane/water interface and at the membrane center, with the remainder interpolated from an expected smooth topology. Going even further, extrapolating from just a single value each of the PMF and the diffusivity at the membrane center for urea (see Fig. 3.6A) produced a $\log P_m$ almost identical to the experimental value. Thus, we recommend that sampling be focused on critical regions where barriers are expected, e.g., the membrane core and/or interfacial region.

Taken together, the results show the robustness of a variety of MD-based methods in calculating membrane permeability for small molecules. In particular, all methods appear to converge on sub-μs time scales, although the determined $\log P_m$ values are nearly all 0.5 to 1.5 log units above the experimental values. This consistent overestimation is in agreement with a previous MD study from 2004 [31], despite using 10-100$\times$ longer simulations in the current study. The membranes compared, DMPC in simulation with egg PC in experiment, are not identical, which may contribute to the discrepancy. Another possible reason could be slow membrane reorganization that occurs on a time scale even an additional 2-3 orders of magnitude greater [52]. The lack of polarizability is another tempting possibility, given the vastly different environments experienced by the permeating molecule [39]. Similarly, the most commonly used force fields (including CGenFF used in this study) are primarily parameterized to reproduce phenomena obtained in aqueous environments. Thus, it may be unrealistic to expect that solute phenomena within the non-polar membrane environment is as well represented. Therefore, further exploration of the force-field effects within lipid-like or non-polar environments are warranted. In particular, inclusion of explicit polarizability may improve accuracy, although at a computational cost of 2-6$\times$ [75, 76]. However, given the lack of convergence of the four methods to a single value for each permeant, all of which use the same force field, it cannot be known a priori to what degree, if any, changes to the force field will improve agreement with experiment. Additionally, the assumptions of the solubility-diffusion model, such as the reliance on a single reaction coordinate and that permeation obeys classical diffusion, may need to be challenged to obtain improved quantitative agreement with experimental measures [45, 77, 78].

Finally, we draw attention to the potential benefit of our results to QSPR models. We have shown that only one or two points in the PMF and diffusivity contribute significantly to the permeability.

Thus, calculations for a single permeant focusing on these critical points could be done in hours on a sufficiently fast cluster. Furthermore, the location of these points, typically at the membrane center or at the water/membrane interface, suggest reduced systems that could represent them reasonably well, e.g., octanol for the interfacial region. Combined with improved force fields, fast calculations on such reduced systems could augment the molecular descriptors used to design QSPR models, similar to the "membrane-interaction QSAR" approach pioneered by Hopfinger et al. [79, 80].

3.7 Methods

3.7.1 System preparation

We constructed a model membrane bilayer consisting of pure dimyristoyl phosphatidylcholine (DMPC) using the CHARMM-GUI membrane builder [81]. The membrane consists of 64 lipids per leaflet solvated by 30-Å water pads on either side. The number of lipids was selected to provide balance between the accuracy of the free energy calculations and the bilayer size [82]. All simulations were run using TIP3P water [83] and CHARMM36 lipid parameters [84]. Previously it has been identified that CHARMM36 is a good choice for small molecule permeability calculations [13]. Small-molecule parameters came from the CGenFF force field [85]. In the case of codeine, which is not part of the standard CGenFF distribution, compatible parameters were obtained from the Paramchem webserver [86, 87].

The membrane-water system was minimized and equilibrated in multiple stages. During the first stage, the membrane was constrained and water was allowed to minimize for 5,000 steps. During the second stage, the membrane was allowed to minimize for 5,000 steps and the water was constrained. During the third stage, both the water and membrane were minimized for 10,000 steps in the absence of constraints. Following minimization, 10 ns of NPT equilibration was carried out using a Lowe-Andersen thermostat [88, 89] and Langevin barostat [90] at a temperature of 298.15 and pressure of 1 atm. The Lowe-Andersen cutoff and coupling rate was set to 2.7 Å and 50 ps^{-1}, respectively. The barostat period and decay times were set to 100 and 50 fs respectively. A 2-fs time step was used. The SETTLE algorithm [91] was used to constrain all covalent bonds to hydrogen atoms. The long-range cutoff was set to 12 Å, and short-range non-bonded and bonded interactions were calculated every time step. Long-range electrostatics were calculated using the Particle Mesh Ewald method [92] every two time steps. All minimization, equilibration, and production dynamics were carried out using the NAMD molecular dynamics engine [68].

Steered MD [55, 56] simulations were used to pull each permeant through the membrane at a speed of 0.7 Å/ns (100 ns in total). Coordinates of the system with the permeant at equally spaced locations over the permeation pathway were extracted from the trajectories and used as initial states for all subsequent simulations.

3.7.2 Umbrella Sampling

The reaction coordinate was defined as the z-component of the distance separating the center of mass of lipid phosphorous atoms and the heavy atoms of the permeant. For each of the permeants, a total of 71 windows spaced 1Å apart were used with a biasing harmonic constraint with a strength of 1.5 kcal/mol/Å ^{2} using the collective variables module of NAMD [93]. Each window was run for a total of at least 20 ns, totaling 1.42 μs of sampling per permeant (40 ns/window and 2.84 μs overall in the case of urea). The resulting biased probability distributions were then reweighted using the weighted histogram model (WHAM) [94–96], implemented in the g_wham software [70], to obtain the PMF. The local diffusivity was estimated using the Hummer positional autocorrelation extension of the Woolf-Roux estimator [65, 66]

$$D(z) = \frac{\langle \delta z^2 \rangle^2}{\int_0^\infty \langle \delta z(t) \delta z(0) \rangle dt}, \tag{3.10}$$

where $\delta z(t) = z(t) - \langle z \rangle$. For the actual numerical calculation of $D(z)$ the integrated autocorrelation times with a sigma of 0.1 was obtained from g_wham. We performed a linear interpolation of the resulting PMF and $D(z)$ values at 1-Å intervals. The results were used to numerically integrate the inhomogeneous solubility-diffusion equation, Eq. 3.1.

3.7.3 Replica-exchange umbrella sampling

For Hamiltonian replica-exchange US (REUS), the same parameters as plain US were used, namely the window spacing and force constant. One additional window in bulk water was added, bringing the total to 72, in order to distribute evenly across available resources. Window exchanges were attempted every 2 ps. Exchange ratios varied between 20 and 30%, which is above a minimum threshold of 10% [97]. Simulations were run for 20 ns for codeine and benzoic acid and 60 ns for urea.

3.7.4 Adaptive biasing force

For each permeant, ABF calculations were performed on nine 12-Å windows, centered at z = -30, -22, -14, -6, 0, 6, 14, 22, 30 Å, respectively. In order to prevent the permeant from leaving the boundary of a window, a wall force constant of 20 kcal/mol/Å ^{2} was used. ABF forces were collected with a bin width of 0.1 Å. A minimum of 500 samples were collected in each bin prior to applying the biasing force. For urea and benzoic acid, each ABF window was simulated for 100 ns, rendering a total of 0.9 μs of sampling. For codeine, each ABF window was simulated for 300 ns, producing a total of 2.7 μs of sampling. The final PMFs were obtained by integrating the gradient force along the reaction coordinate via NAMD. The resulting PMFs were then symmetrized by averaging of data in the +z and -z directions.

3.7.5 Multiple-walker ABF

The underlying idea of multiple-walker ABF is to explore simultaneously different portions of the transition coordinate in multiple replicas of the simulation system. Multiple-walker ABF with N walkers consists of N separate simulation systems, each including only a single permeant. The only interaction between the separate systems is indirect: the force samples accumulated by all walkers up to the current time are combined to give the biasing force experienced by each. However, as the calculation converges, the biasing force approaches a fixed profile and even this indirect interaction between the walkers becomes negligible. The role of multiple-walker ABF is to circumvent possible non-ergodicity scenarios, wherein hidden barriers in orthogonal space hamper diffusion along the transition coordinate. The primary limitation of multiple-walker ABF lies in the possibility that certain walkers become trapped in basins of the free-energy landscape, thereby diminishing the efficiency of the algorithm. To overcome this limitation, an extension of the approach, based on Darwinian selection, eliminates the least effective walkers, i.e., the walkers present in already well-sampled portions of the transition coordinate, while replicating the most effective ones [98, 99]. An implementation of the multiple-walker adaptive biasing force algorithm is available in the popular molecular dynamics program NAMD [68], relying on its replica-communication infrastructure. For the different free-energy calculations reported herein, eight walkers were used. Force buffers were synchronized every 5,000 molecular dynamics steps.

3.7.6 Matlab-generated PMFs and $D(z)$ profiles

The phospholipid bilayer may be regarded as juxtaposed regions with distinct properties and characteristics approximately related to the lipid headgroup and tail densities. [18]. At the bulk water-membrane interface, the headgroup density is low. Next, progressing farther into the membrane, one encounters a strongly hydrophilic and high headgroup density region, then a strongly hydrophobic and high tail density region, and finally a low tail density region at the core of the membrane. The regions of highest PMF contribute the most to the permeability, as implied by Eq. 3.1. With these regions in mind, a rough estimate of the permeability may be determined strictly from analysis of the region of greatest PMF and interpolation to the other regions (the PMF always begins at ends at zero in the bulk water). We combined the simplified membrane model with this interpolation assumption in a Matlab program (provided in the SI). With values for the PMF and diffusivity provided at up to five points, at the center and the two interfacial regions as well as two more intermediate points, this program performs a piecewise Hermite polynomial fit to emulate a PMF and diffusivity landscape, which is then integrated according to Eq. 3.1. By varying the parameters in this script, we can determine the contribution of particular characteristics of the solute-membrane interaction to the resulting permeability.

3.8 Acknowledgements

Chapter 3, in full, is a modified reprint of the material as it appears in "Lee, C. T.; Comer, J.; Herndon, C.; Leung, N.; Pavlova, A.; Swift, R. V.; Tung, C.; Rowley, C. N.; Amaro, R. E.; Chipot, C.; Wang, Y.; Gumbart J. C. *Simulation-Based Approaches for Determining Membrane Permeability of Small Compounds.* J. Chem. Inf. Model. 2016, 56 (4), 721–733." The dissertation author was the primary investigator and author of this work.

CTL acknowledges support from the NIH Molecular Biophysics Training Program (T32-GM008326). JC was supported by Kansas Bioscience Authority funds to the Institute of Computational

Comparative Medicine (ICCM) at Kansas State University and to the Nanotechnology Innovation Center of Kansas State University (NICKS). JCG acknowledges a CAREER award from the National Science Foundation (MCB-1452464), a Burroughs Wellcome Fund Collaborative Research Travel Grant and supercomputing resources provided by the National Institute for Computational Sciences at Oak Ridge National Lab through the Extreme Science and Engineering Discovery Environment (XSEDE), supported by NSF grant number OCI-1053575. CNR thanks NSERC of Canada for funding through the Discovery Grant program (Application 418505-2012) and Compute Canada (RAPI: djk-615-ab) for supercomputing resources. YW acknowledges Project 21403183 supported by National Natural Science Foundation of China. REA acknowledges a NIH Director's New Innovator Award DP2 OD007237, the National Biomedical Computation Resource (NBCR) NIH P41 GM103426, and supercomputing resources provided by XSEDE (NSF TG-CHE060073). REA is a co-founder of Actavalon, Inc.

3.9 Supplementary Information

3.9.1 Application of the Bayesian scheme to MD

To infer from the trajectory the intrinsic system force and the diffusivity, the latter are represented by cubic interpolants, which admit a continuous first derivative. These interpolants are evaluated at equally spaced values, z_i, of the transition coordinate. Under these premises, the position-dependent diffusivity writes,

$$D(z) = a_i^{(0)} + a_i^{(1)} \left(\frac{z - z_i}{h}\right) + a_i^{(2)} \left(\frac{z - z_i}{h}\right)^2 + a_i^{(3)} \left(\frac{z - z_i}{h}\right)^3 \tag{3.11}$$

where $a_i^{(0)}$, $a_i^{(1)}$, $a_i^{(2)}$ and $a_i^{(3)}$ are functions of a_{i-1}, a_i, a_{i+1} and a_{i+2}, that is a set of parameters $\{a_i\}$. The intrinsic system force, which depends on z, can be expressed in a similar fashion, as a function of a set of parameters $\{b_i\}$. The probability $P[(z_{k+1}, t_{k+1} | z_k, t_k) | F(z,t), D(z)]$ can be interpreted as the likelihood of observing the transition from z_k at time t_k to z_{k+1} at time t_{k+1} under the action of a time-dependent bias, $F_{\text{bias},k}$, given the sets of parameters $\{a_i\}$ and $\{b_i\}$. z_k and $F_{\text{bias},k}$ are data readily available from the trajectory. By virtue of Bayes' theorem, the posterior probability of the parameters $\{a_i\}$ and $\{b_i\}$, given z_k

and $F_{\text{bias},k}$, can be expressed as,

$$P(\{a_i\},\{b_i\}|z_k,F_{\text{bias},k}) = P(z_k,F_{\text{bias},k}|\{a_i\},\{b_i\}) \times P_{\text{prior}}(\{a_i\},\{b_i\}) \qquad (3.12)$$

The prior probability, $P_{\text{prior}}(\{a_i\},\{b_i\})$, of the sought parameters can be of different nature — it embraces a number contributions ensuring that $\{a_i\}$ be scale invariant, that the intrinsic system force obtained from $\{b_i\}$ do not depart significantly from that measured in the biased simulation, and that $D(z)$ be smooth over an appreciable length scale.

The diffusivity and the intrinsic system force — or the potential of mean force, are subsequently determined by means of a Metropolis-Hastings algorithm, which generates a Markovian chain of states characterized by consecutive values of the parameters $\{a_i\}$ and $\{b_i\}$. We refer the reader to reference 73 for the computational details of the algorithm.

3.9.2 Qualitative Analysis

In some of the MD trajectory snapshots, we observed conditions in accord with that of others. Shown in Fig 3.8, is the distortion of the bilayer when codeine is constrained at some distance away from the bilayer. Similarly, in Fig 3.9, is shown water molecules traversing the bilayer to solubilize the charged urea, a result previously also described by Cardenas et al[50]. In an attempt to quantify the prevalence of membrane deformation and water finger patterns, an MDAnalysis script[100] was used to bin the positions of phosphates and waters (*i.e.,* those within 5 Å in the x and y directions). The probability distributions for several time blocks from US urea are shown in Fig. 3.10. The choice of phosphate atoms as the reference defining bilayer normal is not optimal due to the substantial rearrangement of phospholipids. Due to the small membrane size, the asymmetry induced by the crossing permeant on headgroup locations will lead to a moving center of membrane reference influencing results.

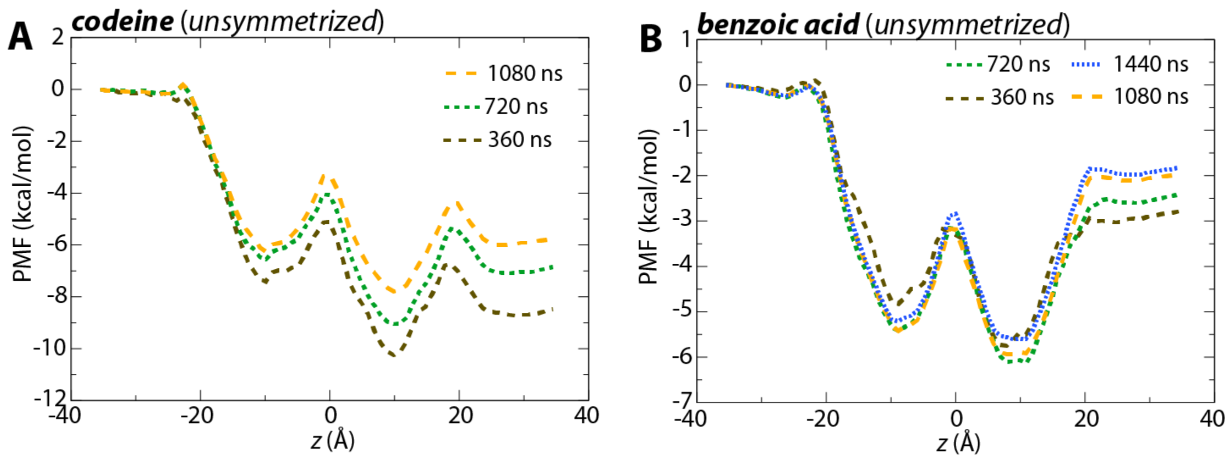


Figure 3.7: Unsymmetrized PMFs for (A) codeine and (B) benzoic acid from REUS.

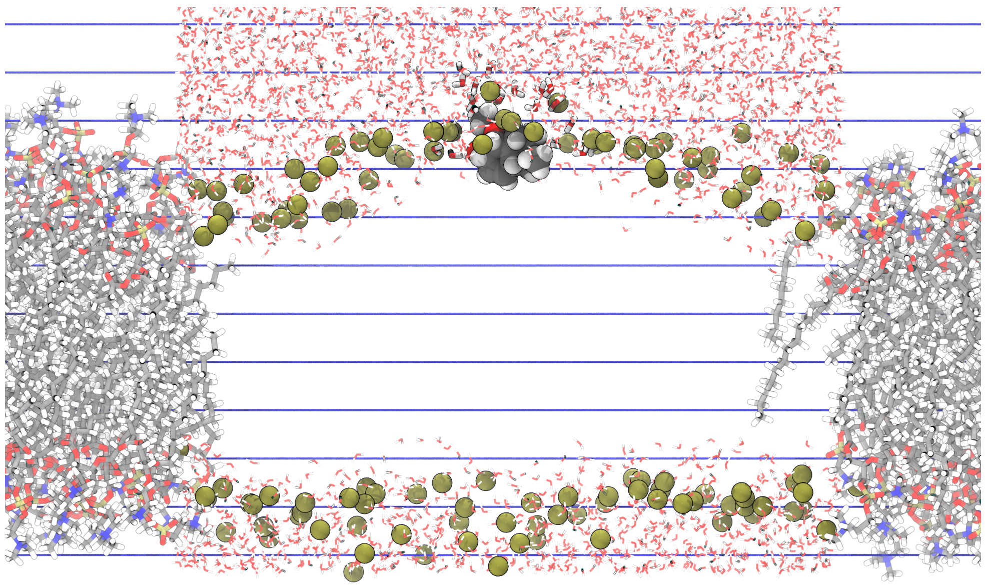


Figure 3.8: A VMD rendered snapshot from the US of codeine constrained at $z = -20$. The lipids in center have been hidden, and lateral periodic images shown. Phosphates are shown as yellow spheres illustrating the bowing of the top leaflet to solubilize the hydrophobic rings. Water molecules within 5 Å of codeine are shown as licorice, while all other waters are lines.

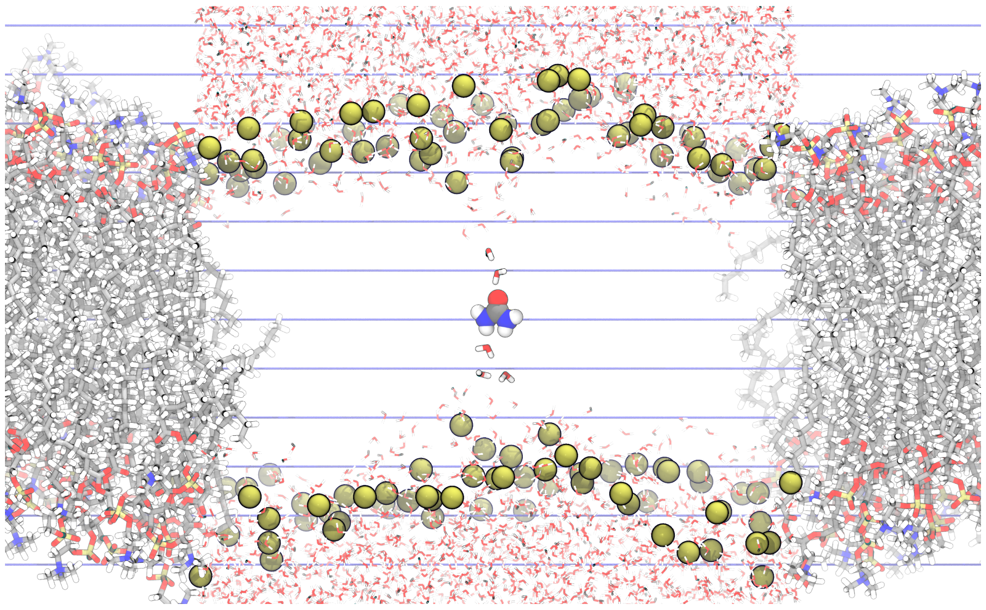


Figure 3.9: A VMD rendered snapshot from the US of urea constrained at $z = 0$. The lipids in center have been hidden, and lateral periodic images shown. Phosphates are shown as yellow spheres. Water molecules within 5Å of urea are shown as licorice.

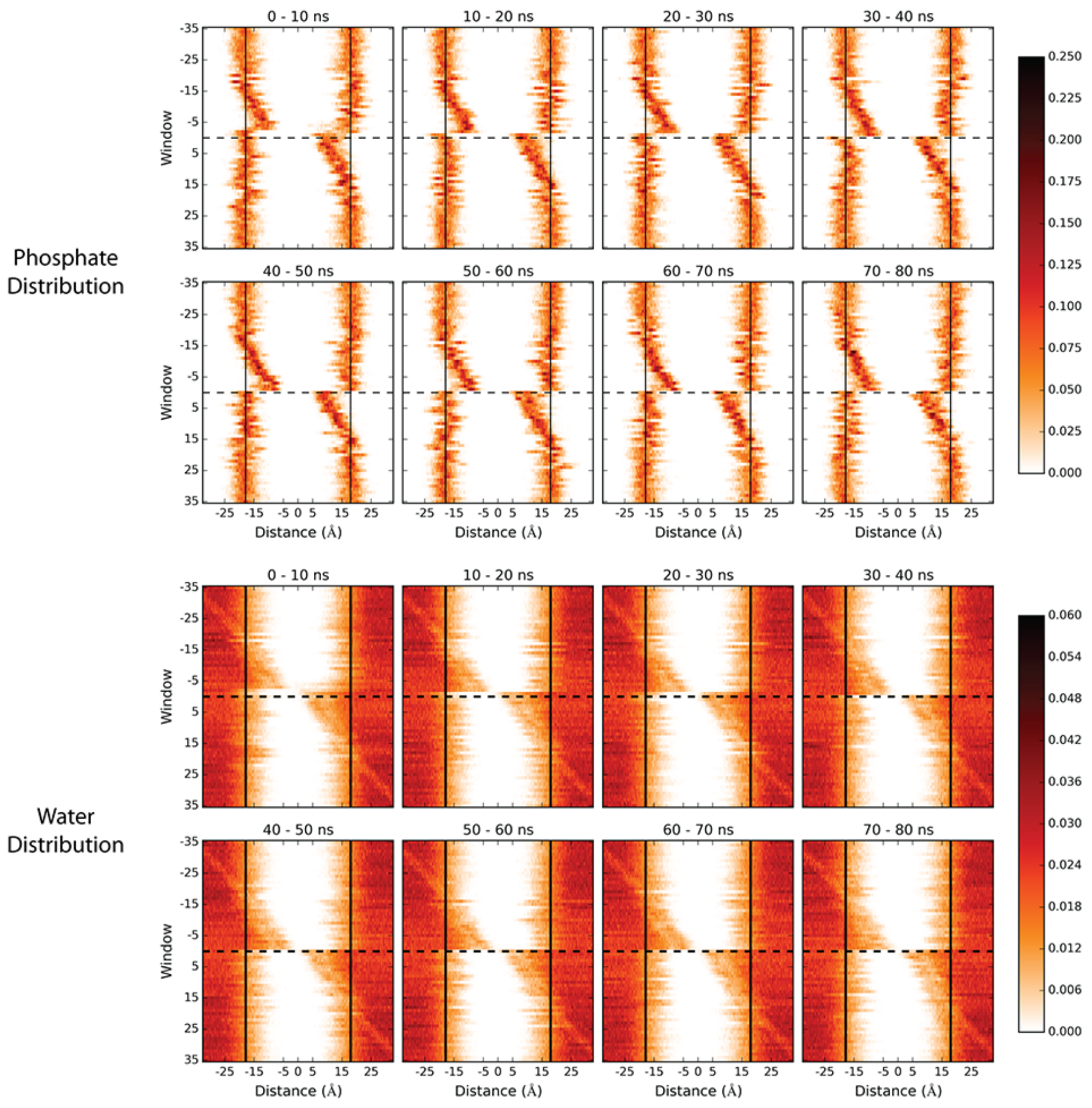


Figure 3.10: The normalized probability density of phosphates and water molecules within 5 Å in the x and y dimensions of urea are shown as a heatmap. The window reflects the z-value reference of the US harmonic constraint. A dotted line is placed at window $z = 0$ for visual reference. Residual non-equilibrium effects can be seen as deviations of the membrane divot from $z = 0$. It appears that both phosphate and water distributions approach the middle after 30 ns of sampling. The vertical black lines highlight the DMPC average phosphate position at equilibrium without permeant. Note the diagonal observed from the upper left to bottom right of the water distribution is not an artifact. The constrained permeant occupies this position and displaces some waters.

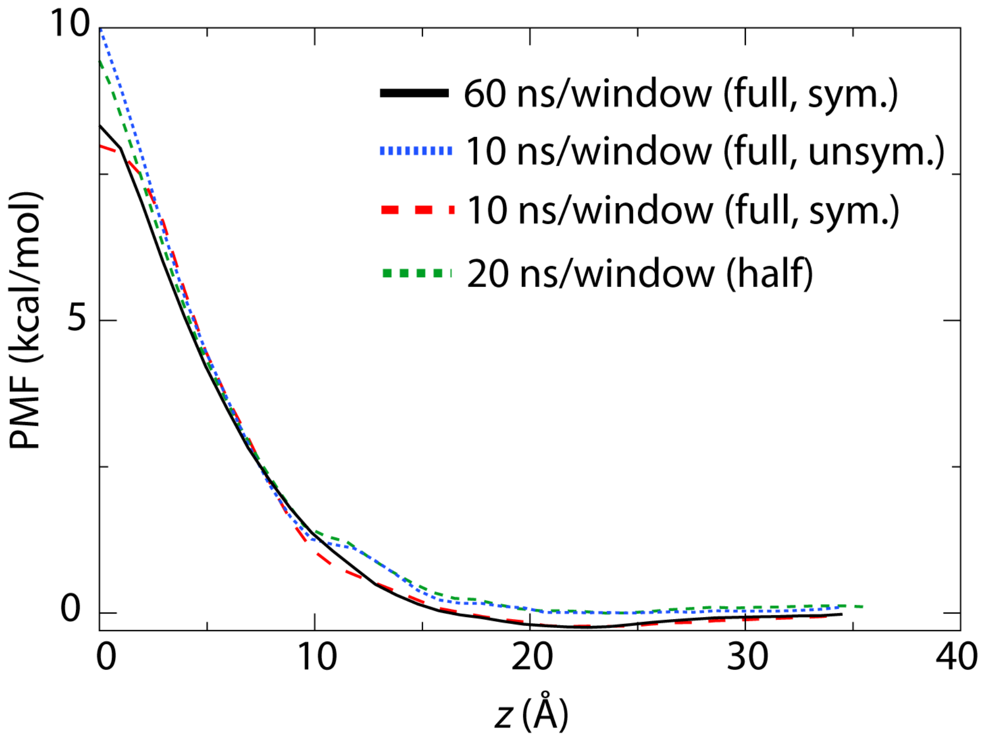


Figure 3.11: PMFs for urea calculated using REUS. The PMFs are shown for only the range *z*=(0,36). The term “full” represents that the REUS calculation was performed for 72 windows spanning the entire bilayer, whereas “half” means only 36 windows spanning the range from the bulk to the membrane center were used. “sym.” or “unsym.” refers to whether or not the PMFs for the full range were symmetrized.

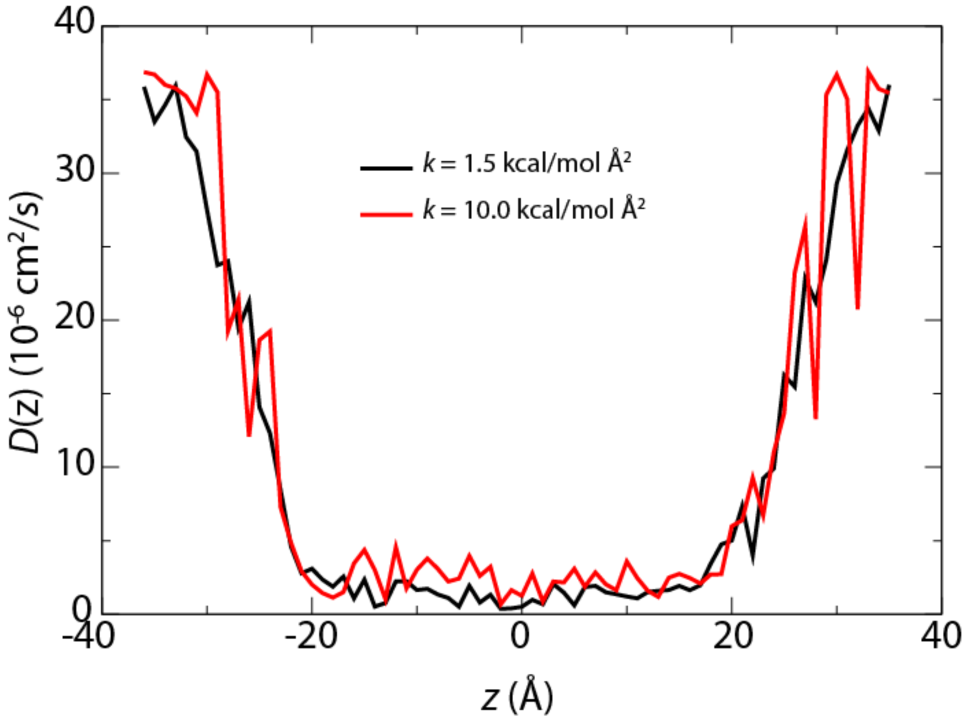
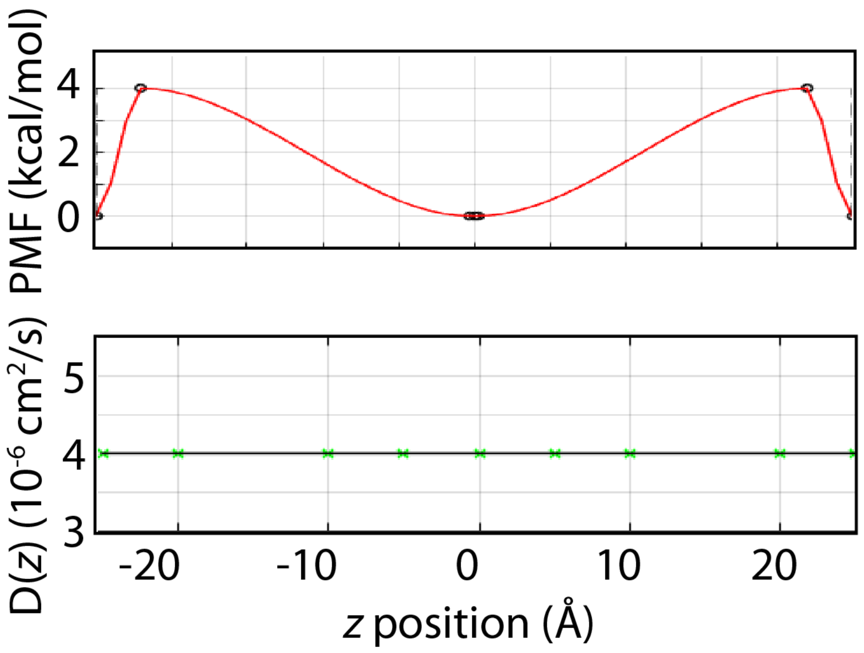


Figure 3.12: Diffusivity profiles for urea calculated using the Generalized Langevin method with different force constants. The black line is for $k = 1.5$ kcal/mol·Å ^{2}, calculated from 1-Å-spaced windows run for 10-ns each, i.e., those also used for the PMF calculation. The red line was computed from 2-ns simulations in each window with a force constant $k = 10$ kcal/mol·Å ^{2}.



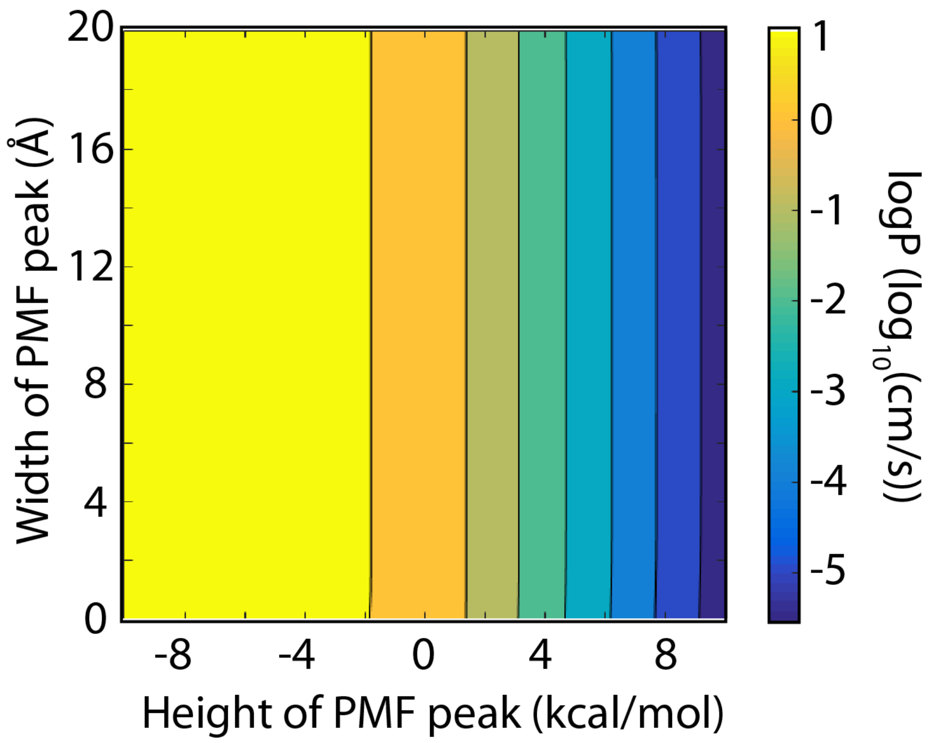


Figure 3.13: LogP$_m$ as a function of modeled input. The input PMF and $D(z)$ are shown on top and the dependence of logP$_m$ on bottom. The height and width of the two peaks in the PMF at the membrane-water interfaces are varied while $D(z)$ is held constant.

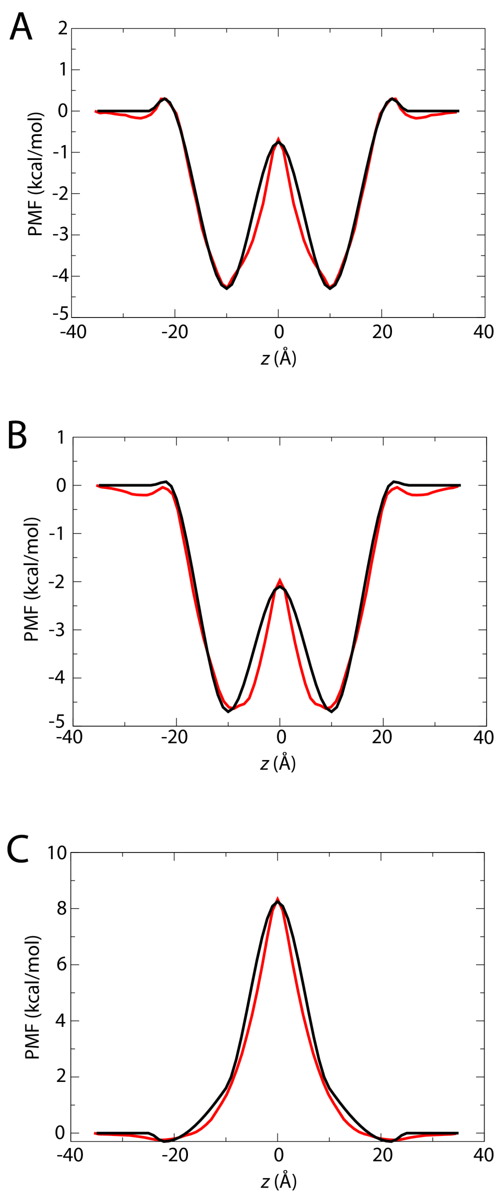


Figure 3.14: PMFs modeled from 3-5 data points at extrema (black) compared to those computed from REUS (red). (A) codeine. (B) benzoic acid. (C) urea.

3.9.3 Code for calculating autocorrelation functions and diffusivity from NAMD traj files

```
usage is ./correlation trajfile fieldnumber

#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <cstdlib>
#include <stdlib.h>
#include <vector>
#include <iterator>
#include <algorithm>

using namespace std;

double *calcCorrelation(double *y, int nSamples, int nCorr)
{
  double *corr=new double[nCorr];
  int *norm=new int[nCorr];
  int min;
  int t;
  int ttoMax;

  for(int i=0;i<nCorr;++i)
    {
      corr[i]=0.0;
      norm[i]=0;
    }

  for(int i=0;i<nSamples;++i)
    {
      ttoMax=nSamples;

      if(i+nCorr<nSamples)
    ttoMax=i+nCorr;

      for(int j=i;j<ttoMax;++j)
    {
      t=j-i;
      corr[t]+=y[i]*y[j];
      norm[t]++;
    }
    }

  for(int i=0;i<nCorr;++i)
```

```
    {
      corr[i]=corr[i]/norm[i];
    }

  delete[] norm;

  return(corr);
}

double variance(double *y, int nSamples)
{
  double v2=0.0;
  for(int i=0;i<nSamples;++i)
    v2+=y[i]*y[i];
  //  cout << "variance << " << v2 << endl;
  v2/=nSamples;
  return(v2);
}

void subtract_average(double *y, int nSamples)
{
  double avg=0.0;

  for(int i=0;i<nSamples;++i)
    avg+=y[i];

  avg/=nSamples;

  for(int i=0;i<nSamples;++i)
    y[i]-=avg;
}

double integrateCorr(double *acf, int nCorr, double timestep)
{
  double I=0;

  for(int i=0;i<nCorr-1;++i)
    {
      I+=0.5*(acf[i]+acf[i+1])*timestep;
    }
  return(I);
}

int countLines(char *fname)
{
  string line;
  int numSamples=0;
  ifstream datafile(fname);

  while(getline(datafile,line))
    {
      if(line.at(0)!='#')
    ++numSamples;
    }

  datafile.close();

  return(numSamples);
}

vector<double> readSeries(char *fname, int &numSamples, int field)
{
  ifstream datafile(fname);
  vector<double> series;
  double *timeSeries;
  int i=0;
  string line;
  istringstream iss;
  int begin;


  if(field==1)
    begin=15;
  else if(field==2)
    begin=37;
  else
    begin=61;

  numSamples=0;

  while(getline(datafile,line))
    {
      //      cout << line << endl;
      if(line.at(0)!='#')
    {
      string str2=line.substr(begin,23);
      series.push_back(atof(str2.c_str()));
      ++numSamples;
    }
    }

  return(series);
```

```
}

int main(int argc, char *argv[])
{
  int nCorr=10000;
  vector<double> series;
  double *acf, *timeSeries;
  double var, I;
  char *fname;
  double timestep=2.0;
  int field=1;
  int numSamples;

  if(argc<1)
    return(1);

  fname=argv[1];

  if(argc>1)
    field=atoi(argv[2]);
  //  int numSamples=countLines(fname)-1;

  series=readSeries(fname, numSamples, field);
  timeSeries=&series[0];

  numSamples=numSamples-1;

  subtract_average(timeSeries, numSamples);
  acf=calcCorrelation(timeSeries, numSamples, nCorr);
  var=variance(timeSeries, numSamples);

  I=integrateCorr(acf, nCorr, timestep);

  cout << "I = " << I << endl;
  cout << "var = " << var << endl;

  cout << "D = " << var*var/I << " A2/fs " << endl;

  cout << "D = " << var*var/I*0.1 << " cm2/s " << endl;

  for(int i=0;i<10;++i)
    cout << i << " " << acf[i] << endl;
  delete[] acf;
  //  series.earse();
}
```

3.9.4 Matlab code for generation of modeled PMF and *D*(*z*) profiles

```
function cA = permeability4(bound,zz,W1,W2,W3,a,b,D1,D2,D3,D4,aD,bD,cD)

% bound: distance from center of membrane to bounds (in Angstroms)
% zz: distance from center to edge of membrane
% W1: PMF at boundary
% W2: PMF between boundary and core
% W3: PMF at core
% a,b: locations of W1,W2
% D1,D2,D3,D4: diffusivity. Input variables are in x*10^-6 (cm^2/s)
% aD,bD,cD: locations for D inputs

%% parameters
%R = 8.3144622./(4184);%(kcal/mol*K)
width = 2.*zz;
beta = 310.15/300*0.5962;
%D = 2e-6;%(cm^2/s)
n = .0001; %integration step size)
bounds = -bound:n:bound;
inds = 1:length(bounds);
integrand = [];
Ma = -zz;%leftmost membrane barrier

%% Creating Diffusion Model
        xD =[inds(bounds==Ma),...
        inds(bounds==-aD),...
        inds(bounds==-bD),...
        inds(bounds==-cD),...
        inds(bounds==0),...
        inds(bounds==cD),...
        inds(bounds==bD),...
        inds(bounds==aD),...
        inds(bounds==(Ma+width))];

yD = [D1 D1 D2 D3 D4 D3 D2 D1 D1];% set values for input W points
xxD = xD(1):1/n:xD(end);% make a dense field of locations for curve fit
D = interp1(xD,yD,xxD,'pchip');
xx2D = xxD.*n - bound;%readjust xx to regular format
xx3D = -bound:bound;
LenD = (length(xx3D)-length(xx2D))/2;
%difference between bound and membrane
xxD = [(min(xx2D)-LenD):(min(xx2D)-1) xx2D (max(xx2D)+1):(max(xx2D)+LenD)];
%make location field from a to b
D = [(zeros(1,LenD)+D(1)) D (zeros(1,LenD)+D(1))];
%outer membrane region has W=0
```

```
D = D * 10^-6;%(cm^2/s)
D = D * 10^16;%(A^2/s)

%% Creating PMF Model
    % set up locations for input W points
        x = [inds(bounds==Ma),...
        inds(bounds==Ma)+1,...
        inds(bounds==-a),...
        inds(bounds==-b),...
        inds(bounds==0),...
        inds(bounds==b),...
        inds(bounds==a),...
        inds(bounds==(Ma+width))-1,...
        inds(bounds==(Ma+width))];

y = [0 0 W1 W2 W3 W2 W1 0 0];% set values for input W points
xx = x(1):1/n:x(end);% make a dense field of locations for curve fit
W = interp1(x,y,xx,'pchip');
xx2 = xx.*n - bound;%readjust xx to regular format
xx3 = -bound:bound;
Len = (length(xx3)-length(xx2))/2;%difference between bound and membrane
xx = [(min(xx2)-Len):(min(xx2)-1) xx2 (max(xx2)+1):(max(xx2)+Len)];
%make location field from a to b
W = [zeros(1,Len) W zeros(1,Len)]; %outer membrane region has W=0

    for j = 1:length(xx)
        integrand(j) = exp(W(j)./beta)./D(j);%(s/A^2)
    end

%% Getting Resistance
integrand = integrand .* 10.^16;%(s/cm^2)
integral = trapz(xx.*10.^-8,integrand);%(s/cm)
integral = integral(end);%resistance (s/cm)

%% Plotting
    %PMF vs position
clf
subplot(211)
hold on
plot(x*n - bound,y,'ko',xx,W,'r','LineWidth',1.5)
plot(linspace(Ma,Ma,length(min(W):.01:max(W))),min(W):.01:max(W), \
                   'k--','LineWidth',1.009);
plot(linspace(Ma+width,Ma+width,length(min(W):.01:max(W))), \
                   min(W):.01:max(W),'k--','LineWidth',1.009);
xlabel('location (angstroms)','FontSize',15)
ylabel('PMF (kcal/mol)','FontSize',15)
```

```
title('PMF','FontSize',40)
axis([-bound,bound,min(W)-1,max(W)+1]);
grid on

%     %resistance vs position
% subplot(312)
% plot(xx,integrand,'b','LineWidth',1.005)%(s/cm^2)
% xlabel('location (angstroms)','FontSize',15)
% ylabel('Resistance per cm (s/cm^2)','FontSize',15)
% title('Resistance')
%     if min(integrand)~=max(integrand)&&max(integrand)>1e6;
%         axis([-bound bound 0 2e7]);
%     elseif min(integrand)~=max(integrand)&&max(integrand)>1e5;
%         axis([-bound bound 0 2e6]);
%     else
%         axis([-bound bound 0 max(integrand)]);
%     end
% grid on

subplot(212)
plot(xD*n-bound,yD.*10^-6,'gx',xxD,D.*10^-16,'k','lineWidth',1.5)
xlabel('location (angstroms)','FontSize',15)
ylabel('Diffusivity (cm^2/s)','FontSize',15)
title('Diffusivity','FontSize',40)
%axis([-bound,bound,min(D.*10^-16),max(D.*10^-16)]);
grid on

%% Cell Array
cA = cell(1);
cA{1,1} = 'Permeation (cm/s)';
cA{2,1} = 'log(P)';
cA{1,2} = 1./integral(1);
cA{2,2} = log10(1./integral(1));
end
```

References

(1) Tanford, C. *Proc. Natl. Acad. Sci. USA* **1979**, *76*, 4175–4176.

(2) Tanford, C., *the Hydrophobic Effect: Formation of Micelles and Biological Membranes*; John Wiley & Sons Inc.: New York, 1973.

(3) Kola, I.; Landis, J. *Nat. Rev. Drug Discov.* **2004**, *3*, 711–715.

(4) Tsaioun, K.; Bottlaender, M.; Mabondzo, A. *BMC Neurol.* **2009**, *9 Suppl 1*, S1.

(5) Kansy, M.; Senner, F.; Gubernator, K. *J. Med. Chem.* **1998**, *41*, 1007–1010.

(6) Avdeef, A. *Expert Opin. Drug Metab. Toxicol.* **2005**, *1*, 325–342.

(7) Artursson, P.; Palm, K.; Luthman, K. *Adv. Drug Deliv. Rev.* **2012**, *64*, 280–289.

(8) Van Breemen, R. B.; Li, Y. *Expert Opin. Drug Metab. Toxicol.* **2005**, *1*, 175–185.

(9) Hansch, C. *Drug Metab. Rev.* **1972**, *1*, 1–14.

(10) Hansch, C. *Acc. Chem. Res.* **1969**, *2*, 232–239.

(11) Stouch, T. R.; Kenyon, J. R.; Johnson, S. R.; Chen, X.-Q.; Doweyko, A.; Li, Y. *J. Comput. Aided Mol. Des.* **2003**, *17*, 83–92.

(12) Swift, R. V.; Amaro, R. E. *Chem. Biol. Drug Des.* **2013**, *81*, 61–71.

(13) Paloncýová, M.; Fabre, G.; DeVane, R. H.; Trouillas, P.; Berka, K.; Otyepka, M. *J. Chem. Theory Comput.* **2014**, *10*, 4143–4151.

(14) Wang, L.; Wu, Y.; Deng, Y.; Kim, B.; Pierce, L.; Krilov, G.; Lupyan, D.; Robinson, S.; Dahlgren, M. K.; Greenwood, J.; Romero, D. L.; Masse, C.; Knight, J. L.; Steinbrecher, T.; Beuming, T.; Damm, W.; Harder, E.; Sherman, W.; Brewer, M.; Wester, R.; Murcko, M.; Frye, L.; Farid, R.; Lin, T.; Mobley, D. L.; Jorgensen, W. L.; Berne, B. J.; Friesner, R. A.; Abel, R. *J. Am. Chem. Soc.* **2015**, *137*, 2695–2703.

(15) Vitalini, F.; Mey, A. S. J. S.; Noé, F.; Keller, B. G.; Vitalini, F.; Mey, A. S. J. S.; Noé, F.; Keller, B. G. *J. Chem. Phys.* **2015**, *142*, 084101.

(16) Finkelstein, A.; Cass, A. *J. Gen. Physiol.* **1968**, *52*, 145–172.

(17) Diamond, J. M.; Katz, Y. *J. Membr. Biol.* **1974**, *17*, 121–154.

(18) Marrink, S.-J.; Berendsen, H. J. C. *J. Phys. Chem.* **1994**, *98*, 4155–4168.

(19) Torrie, G.; Valleau, J. *J. Comput. Phys.* **1977**, *23*, 187–199.

(20) Rodriguez-Gomez, D.; Darve, E.; Pohorille, A. *J. Chem. Phys.* **2004**, *120*, 3563–3578.

(21) Darve, E.; Pohorille, A. *J. Chem. Phys.* **2001**, *115*, 9169.

(22) Darve, E.; Rodriguez-Gomez, D.; Pohorille, A. *J. Chem. Phys.* **2008**, *128*, 144120.

(23) Wei, C.; Pohorille, A. *J. Phys. Chem. B* **2011**, *115*, 3681–3688.

(24) Laio, A.; Parrinello, M. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 12562–12566.

(25) Wang, F.; Landau, D. *Phys. Rev. Lett.* **2001**, *86*, 2050–2053.

(26) Swendsen, R. H.; Wang, J. S. *Phys. Rev. Lett.* **1986**, *57*, 2607–2609.

(27) Xiang, T.-X.; Anderson, B. D. *Adv. Drug Deliv. Rev.* **2006**, *58*, 1357–1378.

(28) Wilson, M. A.; Pohorille, A. *J. Am. Chem. Soc.* **1996**, *118*, 6580–6587.

(29) Grossfield, A.; Woolf, T. B. *Langmuir* **2002**, *18*, 198–210.

(30) Ulander, J.; Haymet, A. D. J. *Biophys. J.* **2003**, 3475–3484.

(31) Bemporad, D.; Essex, J. W.; Luttmann, C. *J. Phys. Chem. B* **2004**, *108*, 4875–4884.

(32) MacCallum, J. L.; Bennett, W. F. D.; Tieleman, D. P. *J. Gen. Physiol.* **2007**, *129*, 371–377.

(33) Johansson, A. C. V.; Lindahl, E. *Proteins* **2008**, *70*, 1332–1344.

(34) MacCallum, J. L.; Bennett, W. F. D.; Tieleman, D. P. *Biophys. J.* **2008**, *94*, 3393–4304.

(35) Bauer, B. A.; Lucas, T. R.; Meninger, D. J.; Patel, S. *Chem. Phys. Lett.* **2011**, *508*, 289–294.

(36) Tejwani, R. W.; Davis, M. E.; Anderson, B. D.; Stouch, T. R. *J. Pharm. Sci.* **2011**, *100*, 2136–2146.

(37) MacCallum, J. L.; Bennett, W. F. D.; Tieleman, D. P. *Biophys. J.* **2011**, *101*, 110–117.

(38) Paloncýová, M.; Berka, K.; Otyepka, M. *J. Chem. Theory Comput.* **2012**, *8*, 1200–1211.

(39) Riahi, S.; Rowley, C. N. *J. Am. Chem. Soc.* **2014**, *136*, 15111–15113.

(40) Carpenter, T. S.; Kirshner, D. A.; Lau, E. Y.; Wong, S. E.; Nilmeier, J. P.; Lightstone, F. C. *Biophys. J.* **2014**, *107*, 630–641.

(41) Issack, B. B.; Peslherbe, G. H. *J. Phys. Chem. B* **2015**, *119*, 9391–9400.

(42) Bemporad, D.; Luttmann, C.; Essex, J. W. *Biochim. Biophys. Acta - Biomembr.* **2005**, *1718*, 1–21.

(43) Comer, J.; Schulten, K.; Chipot, C. *J. Chem. Theory Comput.* **2014**, *10*, 554–564.

(44) Ghaemi, Z.; Minozzi, M.; Carloni, P.; Laio, A. *J. Phys. Chem. B* **2012**, *116*, 8714–8721.

(45) Parisio, G.; Stocchero, M.; Ferrarini, A. *J. Chem. Theory Comput.* **2013**, *9*, 5236–5246.

(46) Kirmizialtin, S.; Elber, R. *J. Phys. Chem. A* **2011**, *115*, 6137–6148.

(47) Vanden-Eijnden, E.; Venturoli, M.; Ciccotti, G.; Elber, R. *J. Chem. Phys.* **2008**, *129*, 174102.

(48) Májek, P.; Elber, R. *J. Chem. Theory Comput.* **2010**, *6*, 1805–1817.

(49) Bello-Rivas, J. M.; Elber, R. *J. Chem. Phys.* **2015**, *142*, 094102.

(50) Cardenas, A. E.; Jas, G. S.; DeLeon, K. Y.; Hegefeld, W. A.; Kuczera, K.; Elber, R. *J. Phys. Chem. B* **2012**, *116*, 2739–2750.

(51) Cardenas, A. E.; Elber, R. *J. Chem. Phys.* **2014**, *141*, 054101.

(52) Neale, C.; Bennett, W. F. D.; Tieleman, D. P.; Pomès, R. *J. Chem. Theory Comput.* **2011**, *7*, 4175–4188.

(53) Neale, C.; Hsu, J. C. Y.; Yip, C. M.; Pomès, R. *Biophys. J.* **2014**, *106*, L29–L31.

(54) Bochicchio, D.; Panizon, E.; Ferrando, R.; Monticelli, L.; Rossi, G. *J. Chem. Phys.* **2015**, *143*, 144108.

(55) Izrailev, S.; Stepaniants, S.; Balsera, M.; Oono, Y.; Schulten, K. *Biophys. J.* **1997**, *72*, 1568–1581.

(56) Sotomayor, M.; Schulten, K. *Science* **2007**, *316*, 1144–1148.

(57) Walter, A.; Gutknecht, J. *J. Membr. Biol.* **1984**, *77*, 255–264.

(58) Gallucci, E.; Micelli, S.; Lippe, C. *Arch. Int. Physiol. Biochim.* **1971**, *79*, 881–887.

(59) Orbach, E.; Finkelstein, A. *J. Gen. Physiol.* **1980**, *75*, 427–436.

(60) Marrink, S. J. S.; Berendsen, H. H. J. C. *J. Phys. Chem.* **1996**, *3654*, 16729–16738.

(61) Holland, B. W.; Gray, C. G.; Tomberli, B. *Phys. Rev. E* **2012**, *86*, 036707.

(62) Mamonov, A. B.; Kurnikova, M. G.; Coalson, R. D. *Biophys. Chem.* **2006**, *124*, 268–278.

(63) Wilson, M. A.; Pohorille, A.; Pratt, L. R. *J. Chem. Phys.* **1985**, *83*, 5832.

(64) Berne, B. J.; Borkovec, M.; Straub, J. E. *J. Phys. Chem.* **1988**, *92*, 3711–3725.

(65) Woolf, T. B.; Roux, B. *J. Am. Chem. Soc.* **1994**, *116*, 5916–5926.

(66) Hummer, G. *New J. Phys.* **2005**, *7*, 34–34.

(67) Allen, M. P.; Tildesley, D. J., *Computer Simulation of Liquids*; Clarendon Press: 1989, p 385.

(68) C. Phillips, J.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; D. Skeel, R.; Kale, L.; Schulten, K. *J. Comput. Chem.* **2005**, *26*, 1781–1802.

(69) Tuckerman, M., *Statistical Mechanics: Theory and Molecular Simulation*; OUP Oxford: Oxford, 2010, p 720.

(70) Hub, J. S.; de Groot, B. L.; van der Spoel, D. *J. Chem. Theory Comput.* **2010**, *6*, 3713–3720.

(71) Neale, C.; Madill, C.; Rauscher, S.; Pomès, R. *J. Chem. Theory Comput.* **2013**, *9*, 3686–3703.

(72) Türkcan, S.; Alexandrou, A.; Masson, J.-B. *Biophys. J.* **2012**, *102*, 2288–2298.

(73) Comer, J.; Chipot, C.; González-Nilo, F. D. *J. Chem. Theory Comput.* **2013**, *9*, 876–882.

(74) Dorairaj, S.; Allen, T. W. *Proc. Natl. Acad. Sci. USA* **2007**, *104*, 4943–4948.

(75) Chowdhary, J.; Harder, E.; Lopes, P. E. M.; Huang, L.; MacKerell Jr., A. D.; Roux, B. *J. Phys. Chem. B* **2013**, *117*, 9142–9160.

(76) Wang, L.-P.; Head-Gordon, T.; Ponder, J. W.; Ren, P.; Chodera, J. D.; Eastman, P. K.; Martinez, T. J.; Pande, V. S. *J. Phys. Chem. B* **2013**, *117*, 9956–9972.

(77) Orsi, M.; Essex, J. W. *Soft Mat.* **2010**, *6*, 3797–3808.

(78) Comer, J.; Schulten, K.; Chipot, C. *J. Chem. Theory Comput.* **2014**, *10*, 2710–2718.

(79) Kulkarni, A. S.; Hopfinger, A. J. *Pharm. Res.* **1999**, *16*, 1244–1252.

(80) Tseng, Y. J.; Hopfinger, A. J.; Esposito, E. X. *J. Comput. Aided Mol. Des.* **2012**, *26*, 39–43.

(81) Wu, E. L.; Cheng, X.; Jo, S.; Rui, H.; Song, K. C.; Dávila-Contreras, E. M.; Qi, Y.; Lee, J.; Monje-Galvan, V.; Venable, R. M.; Klauda, J. B.; Im, W. *J. Comput. Chem.* **2014**, *35*, 1997–2004.

(82) Hu, Y.; Ou, S.; Patel, S. *J. Phys. Chem. B* **2013**, *117*, 11641–11653.

(83) Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. *J. Chem. Phys.* **1983**, *79*, 926.

(84) Klauda, J. B.; Venable, R. M.; Freites, J. A.; O'Connor, J. W.; Tobias, D. J.; Mondragon-Ramirez, C.; Vorobyov, I.; MacKerell, A. D.; Pastor, R. W. *J. Phys. Chem. B* **2010**, 7830–7843.

(85) Vanommeslaeghe, K.; Hatcher, E.; Acharya, C.; Kundu, S.; Zhong, S.; Shim, J.; Darian, E.; Guvench, O.; Lopes, P.; Vorobyov, I.; Mackerell, A. D. *J. Comput. Chem.* **2010**, *31*, 671–690.

(86) Vanommeslaeghe, K.; Raman, E. P.; MacKerell, A. D. *J. Chem. Inf. Model.* **2012**, *52*, 3155–3168.

(87) Vanommeslaeghe, K.; MacKerell, A. D. *J. Chem. Inf. Model.* **2012**, *52*, 3144–3154.

(88) Koopman, E. A.; Lowe, C. P. *J. Chem. Phys.* **2006**, *124*, 204103.

(89) Lowe, C. P. *Europhys. Lett.* **1999**, *47*, 145–151.

(90) Feller, S. E.; Zhang, Y.; Pastor, R. W.; Brooks, B. R. *J. Chem. Phys.* **1995**, *103*, 4613.

(91) Miyamoto, S.; Kollman, P. A. *J. Comput. Chem.* **1992**, *13*, 952–962.

(92) Darden, T.; York, D.; Pedersen, L. *J. Chem. Phys.* **1993**, *98*, 10089.

(93) Hénin, J.; Forin, G.; Chipot, C.; Klein, M. L. *J. Chem. Theory Comput.* **2010**, *6*, 35–47.

(94) Kumar, S.; Rosenberg, J. M.; Bouzida, D.; Swendsen, R. H.; Kollman, P. A. *J. Comput. Chem.* **1992**, *13*, 1011–1021.

(95) Tan, Z.; Gallicchio, E.; Lapelosa, M.; Levy, R. M. *J. Chem. Phys.* **2012**, *136*, 144102.

(96) Chodera, J. D.; Swope, W. C.; Pitera, J. W.; Seok, C.; Dill, K. A. *J. Chem. Theory Comput.* **2007**, *3*, 26–41.

(97) Sugita, Y.; Okamoto, Y. *Chem. Phys. Lett.* **1999**, *314*, 141–151.

(98) Minoukadeh, K.; Chipot, C.; Lelièvre, T. *J. Chem. Theory Comput.* **2010**, *6*, 1008–1017.

(99) Comer, J.; Phillips, J. C.; Schulten, K.; Chipot, C. *J. Chem. Theory Comput.* **2014**, *10*, 5276–5285.

(100) Michaud-Agrawal, N.; Denning, E. J.; Woolf, T. B.; Beckstein, O. *J. Comput. Chem.* **2011**, *32*, 2319–2327.

Chapter 4

Milestoning Permeability

4.1 Abstract

Prediction of passive permeation rates of solutes across lipid bilayers is important to drug design, toxicology and other biological processes such as signaling. The inhomogeneous solubility-diffusion (ISD) equation is traditionally used to relate the position-dependent potential of mean force and diffusivity to the permeability coefficient. The ISD equation is derived via the Smoluchowski equation and assumes over-damped system dynamics. It has been suggested that the complex membrane environment may exhibit more complicated damping conditions. Here we derive a variant of the inhomogeneous solubility diffusion equation as a function of the mean first passage time (MFPT) and show how milestoning, a method that can estimate kinetic quantities of interest, can be used to estimate the MFPT of membrane crossing and, by extension, the permeability coefficient. We further describe a second scheme, agnostic to the damping condition, to estimate the permeability coefficient from milestoning results or other methods that compute a probability of membrane crossing. The derived relationships are tested using a 1-D Langevin dynamics toy system and confirmed that the presented theoretical methods can be used to estimate permeabilities given simulation and milestoning results.

4.2 Introduction

Passive transport of solutes across membranes is of key importance for processes such as the uptake and excretion of drug compounds. Poor permeability can cause unfavorable pharmacokinetics or bioavailability leading to drug attrition[1]. From this perspective of drug design, methods which provide molecular level insight into the permeation process are of high interest. As such, many physical models of permeability have been developed; these methods are recently reviewed[2].

Mathematically, the permeability coefficient P of a solute is described by relating the flux per unit area, J, across a bilayer and the concentration gradient, Δu, as

$$P = \frac{J}{\Delta u}. \tag{4.1}$$

From this relation, assuming over-damped conditions and accounting for drift caused by the potential of mean force (PMF) we get the inhomogeneous solubility-diffusion equation (ISD)[3–7]:

$$P = \frac{1}{R} = \left[\int_{z_1}^{z_2} \frac{\exp(\beta W(z))}{D(z)} dz \right]^{-1}, \tag{4.2}$$

where R is the resistivity, $\beta = 1/k_B T$ where T is the temperature, k_B is Boltzmann's constant, $W(z)$ is the PMF, $D(z)$ is the local diffusivity constant and z_1, z_2 are two positions just outside of opposing sides of the membrane. Estimates of $W(z)$ and $D(z)$ can be predicted using molecular dynamics (MD) simulations[2, 6, 7]. While the ISD has been applied successfully in many studies[2, 5, 8–22], recent work by various groups has suggested that the dynamics of crowded environments, such as inside the bilayer, may exhibit more complicated behaviors[2, 23]. The dynamics inside of the bilayer has not, to the best of our knowledge, been probed in detail[2].

The theory of milestoning is a promising new approach to study molecular processes by employing many unbiased short independent simulations in subregions of phase space to predict thermodynamic and kinetic properties of interest. Salient descriptions of milestoning theory can be found in Refs. 242526. Previously, milestoning has been applied to the permeability problem by Cardenas et al., studying the permeation of block tryptophan across a membrane[27]. Cardenas et al. later studied the permeation of small peptide NATA across a DOPC bilayer using milestoning. In their later work, they proposed a scheme to estimate the permeability coefficient[28, 29]. This scheme is comprised of three steps involving: 1) estimating the diffusivity of the small molecule in bulk, 2) the measurement of the steady state flux from bulk into the membrane, 3) estimation of permeant crossing probability. We will show that the additional calculations in step 2 are not necessary and the permeability can be computed directly from the milestoning results and bulk diffusivity.

In this work we derive two methods for obtaining permeability coefficients based on milestoning results. The first derivation is a new formulation of the ISD relating the MFPT to the permeability coefficient P assuming over-damped dynamics. The second derivation makes no assumptions about system damping, and relates P to the transition probability of crossing a membrane over diffusing to a previous position in bulk solvent. We evaluate the compatibility of using either approach depending on damping dynamics using a 1-D Langevin dynamics system across various $W(z)$ and $D(z)$ profiles, showing that the new relations yield solutions in good accord with previous results.

4.3 Theory

4.3.1 Relating permeability to mean first passage time via the Smoluchowski equation

The physical description of permeability can be drawn as a membrane separating donor and acceptor compartments (Fig. 4.1). Assuming that the dynamics are overdamped, permeation across a membrane can be modeled as a one-dimensional process using the 1-D Smoluchowski equation

$$\frac{\partial u(z,t)}{\partial t} = -\frac{\partial J(z,t)}{\partial z} = \frac{\partial}{\partial z}\left(D(z)\frac{\partial u(z,t)}{\partial z} - v(z,t)\cdot u(z,t) \right), \tag{4.3}$$

where $u(z,t)$ is the probability distribution at position z and time t, $v(z,t)$ is the drift velocity, and $J(z,t)$ is the infinitesimal flux. A particle travels through a viscous medium at velocity $v = F(z)/\gamma$, where F is the force on the particle and γ is the friction coefficient. Using the Smoluchowski-Einstein relation $\gamma^{-1} = \beta D$, assuming steady-state conditions, and applying that $F = -\frac{dW(z)}{dz}$, we can convert the partial

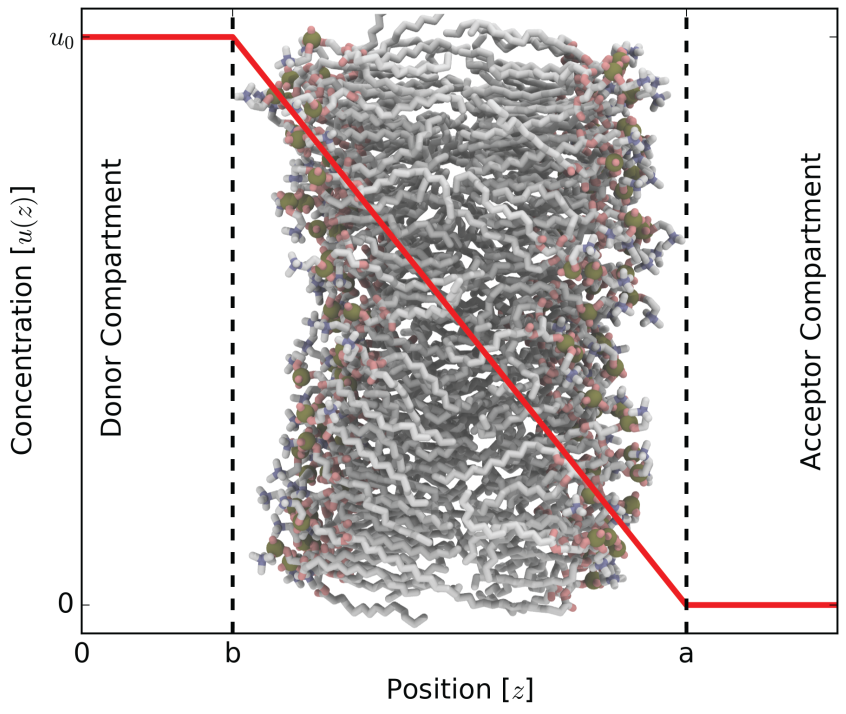


Figure 4.1: Schematic of membrane permeability showing the concentration of permeant with respect to position. We define that the concentration at $z = b$ is u_0 and $z = a$ is 0. Assuming over-damped dynamics along with these boundary conditions, we get the inhomogeneous solubility-diffusion equation and its relation to passage time.

differential equation into an ordinary differential equation

$$0 = \frac{d}{dz}\left(D(z)\left[\frac{du(z)}{dz} + \frac{d\beta W(z)}{dz}u(z)\right]\right). \tag{4.4}$$

Integrating and multiplying by $\frac{e^{\beta W(z)}}{D(z)}$, we get

$$e^{\beta W(z)}\frac{A_1}{D(z)} = \frac{du(z)}{dz}e^{\beta W(z)} + u(z)\frac{d\beta W(z)}{dz}e^{\beta W(z)}, \tag{4.5}$$

where A_1 is a constant. The RHS can be condensed into a single differential

$$\frac{A_1}{D(z)}e^{\beta W(z)} = \frac{d}{dz}\left(u(z)e^{\beta W(z)}\right). \tag{4.6}$$

By integrating, we can obtain a general expression for the steady-state concentration

$$u(z) = A_1 e^{-\beta W(z)}\left(\int_0^z \frac{e^{\beta W(z')}}{D(z')}dz'\right) + A_2 e^{-\beta W(z)}. \tag{4.7}$$

We now apply the boundary conditions shown in Fig. 4.1. Assuming that the acceptor compartment is a sink, $u(a) = 0$, and that in bulk $W(a) = 0$,

$$0 = A_1\left(\int_0^a \frac{e^{\beta W(z')}}{D(z')}dz'\right) + A_2. \tag{4.8}$$

Solving for A_2

$$A_2 = -A_1\int_0^a \frac{e^{\beta W(z')}}{D(z')}dz'. \tag{4.9}$$

Substituting Eq. 4.9 into Eq. 4.7, we get a difference of two integrals. Combining them into a single integral and setting the concentration of the donor compartment at position b to that of bulk u_0

$$u(b) = u_0 = A_1 e^{-\beta W(b)}\left(\int_a^b \frac{e^{\beta W(z')}}{D(z')}dz'\right). \tag{4.10}$$

We can solve for the constant A_1

$$A_1 = \frac{u_0}{\int_a^b \frac{e^{\beta W(z')}}{D(z')} dz'} = -\frac{u_0}{\int_b^a \frac{e^{\beta W(z')}}{D(z')} dz'}. \tag{4.11}$$

Combining Eqs. 4.11, 4.9, 4.7, the concentration $u(z)$ can be expressed according to

$$u(z) = \frac{-u_0 e^{-\beta W(z)} \left(\int_a^z \frac{e^{\beta W(z')}}{D(z')} dz' \right)}{\int_b^a \frac{e^{\beta W(z'')}}{D(z'')} dz''}. \tag{4.12}$$

Returning to the calculation of the stationary flux, substituting Eq. 4.5 into Eq. 4.3, and recognizing that the flux is just the negative of our first constant of integration, we get that

$$J = -A_1 = \frac{u_0}{\int_b^a \frac{e^{\beta W(z')}}{D(z')} dz'}. \tag{4.13}$$

Substituting into Eq. 4.1 and assuming that $\Delta u = u_0$, we get the ISD

$$P = \left[\int_b^a \frac{e^{\beta W(z')}}{D(z')} dz' \right]^{-1} \tag{4.14}$$

Mean First Passage Time

The MFPT $\langle t \rangle$ of a diffusional encounter can be generally expressed as[30]

$$\langle t \rangle = \frac{\int_v u dV}{\sum_{i=1}^m \int_{A_i} J(A_i) dA_i}, \tag{4.15}$$

here V is the total volume of the system in question, m is the total number of isolated sinks, and A_i is the total area of the ith sink. In our 1-D case, the expression is simplified:

$$\langle t \rangle = \frac{\int_b^a u(z') dz'}{J(a)}. \tag{4.16}$$

Taking the integral of Eq. 4.12, and reversing the order of integration in the numerator, we get

$$\int_b^a u(z)dz = \int_b^a \frac{u_0 e^{-\beta W(z)} \left[\int_z^a \frac{e^{\beta W(z')}}{D(z')} dz'\right]}{\int_b^a \frac{e^{\beta W(z'')}}{D(z'')} dz''} dz. \tag{4.17}$$

Noting the equivalence of $\int_b^a \int_z^a f(x)dxdz = \int_b^a \int_b^z f(x)dxdz$ for symmetrical functions over $z|b < z < a$ and dividing by $J(a) = -A_1$ we get

$$\langle t \rangle = -\int_b^a e^{-\beta W(z)} \left[\int_b^z \frac{e^{\beta W(z')}}{D(z')} dz'\right] dz. \tag{4.18}$$

The MFPT can be related to the permeability coefficient by integrating Eq. 4.18 by parts

$$\langle t \rangle = \left[\int_b^a e^{-\beta W(z)} dz\right] \left[\int_b^a \frac{e^{\beta W(z)}}{D(z)} dz\right] - \underbrace{\int_b^a \frac{e^{\beta W(z)}}{D(z)} \left[\int_b^z e^{-\beta W(z')} dz'\right] dz}_{\langle t \rangle}, \tag{4.19}$$

where the second term on the RHS has been shown to be equal to the MFPT by others[10, 27, 31, 32] and can be derived by applying Fubini's theorem. Rewriting in terms of P and $\langle t \rangle$

$$\langle t \rangle = \frac{1}{P} \int_b^a e^{-\beta W(z)} dz - \langle t \rangle \tag{4.20}$$

and rearranging we get the permeability as it relates to $\langle t \rangle$

$$P = \frac{\int_b^a e^{-\beta W(z)} dz}{2\langle t \rangle}. \tag{4.21}$$

This is formally equivalent to the ISD and obviates the calculation of the local diffusivity, requiring only the local PMF (or equivalently, the local stationary concentration of the permeant) and the MFPT.

4.3.2 Permeability from milestoning using crossing probability

Our second permeability derivation in this section is very similar to that used in Brownian dynamics (BD) simulations that make use of the Northrup-Allison-McCammon (NAM) algorithm[33]. As in BD simulations, the *b*-surface is the starting position of particles in the simulations. Conversely, the *q*-surface and a-surface are absorbing boundaries during the simulations. The quantity ρ is the proportion of permeants that start at the b-surface and cross the membrane to touch the a-surface and not "escape" to the q-surface. Here, we attempt to be consistent with notation used in the NAM derivation. The only exception is that NAM uses a β to represent the probability of crossing the a-surface, but we use the symbol ρ instead to avoid confusion with the thermodynamic β used in the previous section. This probability ρ can be found using milestoning.

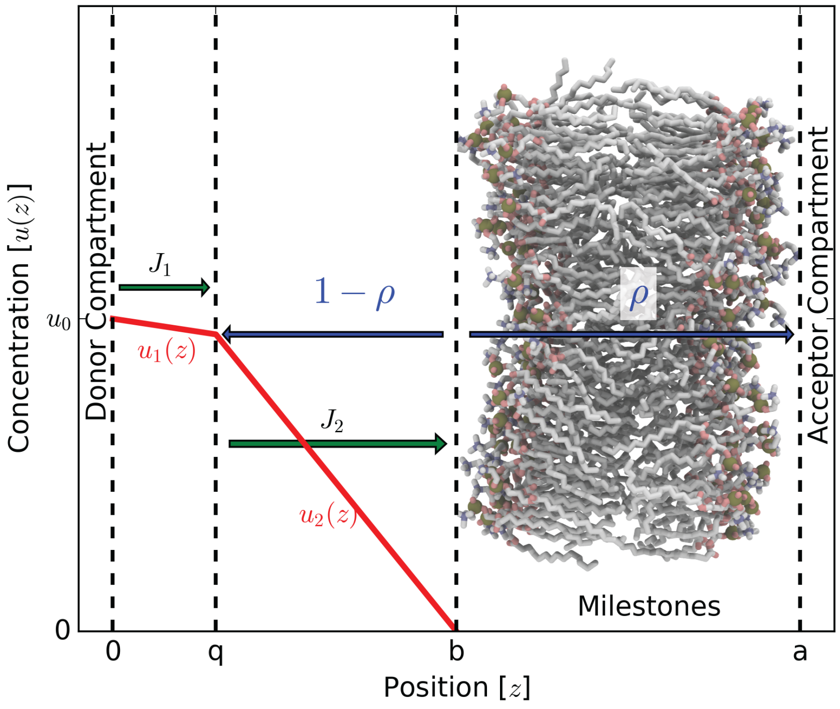


Figure 4.2: Shown here is the milestoning system setup. The stationary concentrations in the bulk u_1 and u_2 are subject to boundary conditions at $z = 0$ and $z = b$. A permeant that hits the b-surface has a probability of ρ to cross the membrane and a probability $1-\rho$ to return to the q-surface. These boundaries and domains are not necessarily drawn to scale.

We define one additional surface in bulk solution on the donor side of the membrane: 0-surface (placed at $z = 0$ for mathematical convenience). Note that $a > b > q > 0$. A steady-state Smoluchowski problem (Fig. 4.2) is set up in the bulk domain between the 0-surface and the b-surface, where the concentration at the 0-surface boundary is held at u_0 and the concentration at the b-surface is held at 0. Any flux across the b-surface has a probability equal to ρ that it will be taken across the membrane and

$1-\rho$ to return to the q-surface. Under this scheme, the permeability from the 0-surface to a-surface can be found according to

$$P_{0\to a} = J(a)/u_0, \tag{4.22}$$

where u_0 is the difference between the concentration at the 0-surface and the concentration at the a-surface, and $J(a)$ is the flux across the a-surface. Since we assume steady-state conditions, the flux across the a-surface must be equal to the flux across the 0-surface

$$P_{0\to a} = J(0)/u_0. \tag{4.23}$$

Due to the fact that the region between the 0-surface and b-surface exists in bulk solution, we assume that the PMF is constant and the dynamics over-damped, and we can use a 1-D Smoluchowski equation for a continuum description of this region. Although the q-surface is an absorbing boundary during the simulations and milestoning calculations, the q-surface will become a source boundary in the continuum Smoluchowski calculations for the region where $0 < z < b$. Here, it is useful to split this domain into two pieces: $0 < z < q$, $q < z < b$. We then solve a system of two Smoluchowski equations:

Domain 1:

$$0 = -\frac{\partial J_1(z)}{\partial z} = D\frac{d^2 u_1}{dz^2} \tag{4.24a}$$

$$u_1 = u_1(z), \text{where } 0 < z < q \tag{4.24b}$$

$$u_1(0) = u_0 \tag{4.24c}$$

Domain 2:

$$0 = -\frac{\partial J_2(z)}{\partial z} = D\frac{d^2 u_2}{dz^2} \tag{4.24d}$$

$$u_2 = u_2(z); \text{where } q < z < b \tag{4.24e}$$

$$u_2(b) = 0 \tag{4.24f}$$

Domain 1 and 2:

$$u_1(q) = u_2(q) \tag{4.24g}$$

$$J_2(q) - J_1(q) = J_2(q) \cdot (1-\rho) \tag{4.24h}$$

where D is the diffusion coefficient of the particle in bulk water. The last equation, Eq. 4.24h, assumes that the flux through the 0-surface (J_1) is less than the flux going through the b-surface (J_2) because some of that flow comes back and appears once again on the q-surface. Since anything that goes through the b-surface has a probability of $1-\rho$ to return to the q-surface, the flow out of the q-surface is what flows through the b-surface (J_2) multiplied by $1-\rho$. This is equal to the difference between J_2 and J_1. We find a general solution to Eqs. 4.24a and 4.24d by integrating twice.

$$u_1 = c_1 z + d_1 \tag{4.25a}$$

$$u_2 = c_2 z + d_2 \tag{4.25b}$$

where c_1, c_2, d_1 and d_2 are constants of integration. By Eqs. 4.24a, 4.24d we know that $J = D(-du/dz)$, and therefore $J_1 = -Dc_1$ and $J_2 = -Dc_2$ and thus Eq. 4.24h simplifies to $c_1 = \rho c_2$. Upon applying the boundary conditions from Eqs. 4.24c, 4.24f to Eq. 4.25 we get that $d_1 = u_0$ and $d_2 = -c_1 b/\rho$. Substituting the above relations and Eq. 4.25 into 4.24g and solving for c_1 we get

$$c_1 = \frac{-J_1}{D} = \frac{-u_0\rho}{b-(1-\rho)q}. \tag{4.26}$$

Rearranging for the flux, J_1, and dividing by u_0, we get the permeability from the 0-surface to the a-surface

$$P_{0\to a} = \frac{D\rho}{b-(1-\rho)q}. \tag{4.27}$$

Now we remove the contribution to the permeability by the bulk layer, $0 < z < b$, to obtain the true permeability across the membrane $P_{b\to a}$. This is done most effectively by working with the resistivity,

$R = 1/P$. Expressing the resistivities across the domain as a sum of resistivities over subdomains

$$\frac{1}{P_{0\to a}} = R_{0\to a} = R_{0\to b} + R_{b\to a}. \tag{4.28}$$

Assuming a constant PMF and diffusivity in bulk, $0 < z < b$, the resistivity in that domain can be expressed as

$$R_{0\to b} = \frac{b}{D}, \tag{4.29}$$

obtained by solving the Smoluchowski equation for a flat PMF and diffusion profile. Plugging Eq. 4.29 into Eq. 4.28 and inverting we get the true membrane permeability as related to the crossing probability ρ

$$P_{b\to a} = \frac{D\rho}{(b-q)(1-\rho)}. \tag{4.30}$$

For subsequent discussion we refer to Eq. 4.21 as the MFPT in ISD relation (MFPT-ISD), and Eq. 4.30 as the permeability based on crossing probability (PBCP).

4.4 Methods

Langevin Dynamics Model: To demonstrate the validity of the above relations we have developed a 1-D Langevin dynamics model to generate sample trajectories over a user specified PMF and viscosity profile. All code used has been made available through GitHub at https://github.com/ctlee/langevin-milestoning. The profiles are represented using interpolated cubic Hermite polynomials shown in Fig 4.3. Four different systems are considered, flat, small-barrier, urea-like barrier and codeine-like. The flat system emulated diffusion in bulk with viscosity chosen to match water. For the small-barrier system, an arbitrary barrier height was chosen such that brute force computation could yield successful crossings. Meanwhile the urea- and codeine-like systems are interpolated from profiles from Ref. 34. To facilitate convergence of calculations, the viscosity of the membrane environment is set to 0.005 $kgm^{-1}s^{-1}$. Each of the membrane simulations employ the same viscosity profiles. The hydrodynamic radius and mass of the particle were arbitarily chosen for the flat and small-barrier cases while the urea and codeine cases use molecular weight and a radius as predicted by software `HydroPro`[35]. The parameters used in each case are shown in Table 4.1. The diffusivity along the profile, D, can be calculated from the Stokes-Einstein relation

$$D = \frac{k_B T}{6\pi\eta R_{hyd}} = \frac{k_B T}{c}, \tag{4.31}$$

where η is the viscosity, R_{hyd} the hydrodynamic radius, and c the friction coefficient. Notably, new systems can be generated with any user-specified PMF, viscosity, mass and hydrodynamic radius parameters of interest.

We integrate the Langevin equation using a velocity-Verlet-like integrator by Grønbech-Jensen–Farago[36] to generate sample trajectories

$$q(t+dt) = q(t) + bdtv(t) + \frac{bdt^2}{2m} f(t) + \frac{bdt}{2m}\Gamma(t+dt), \tag{4.32}$$

$$v(t+dt) = av(t) + \frac{dt}{2m}[af(t) + f(t+dt)] + \frac{b}{m}\Gamma(t+dt), \tag{4.33}$$

$$b \equiv \frac{1}{1+\frac{cdt}{2m}}, \tag{4.34}$$

$$a \equiv \frac{1-\frac{cdt}{2m}}{1+\frac{cdt}{2m}}, \tag{4.35}$$

where q is the position, t is the current time, c the friction coefficient, b and a are unitless constants, f is the force, v the velocity, and m the particle mass. This integrator was derived by integrating the noise term at a small time step dt. The new term, Γ, is Gaussian distributed and has the following properties:

$$\langle \Gamma(t+dt) \rangle = 0, \tag{4.36}$$

$$\langle \Gamma(t)\Gamma(t') \rangle = 2ck_B T dt \delta(t,t'). \tag{4.37}$$

The quality of random number generators (RNG) to sample Γ is shown in Fig. 4.6. In this work, the aforementioned drag coefficient, c, is estimated by Stokes law,

$$F_d = \underbrace{6\pi\eta R_{hyd}}_{c} v, \tag{4.38}$$

where F_d is the frictional force. For additional information regarding the implementation details of the

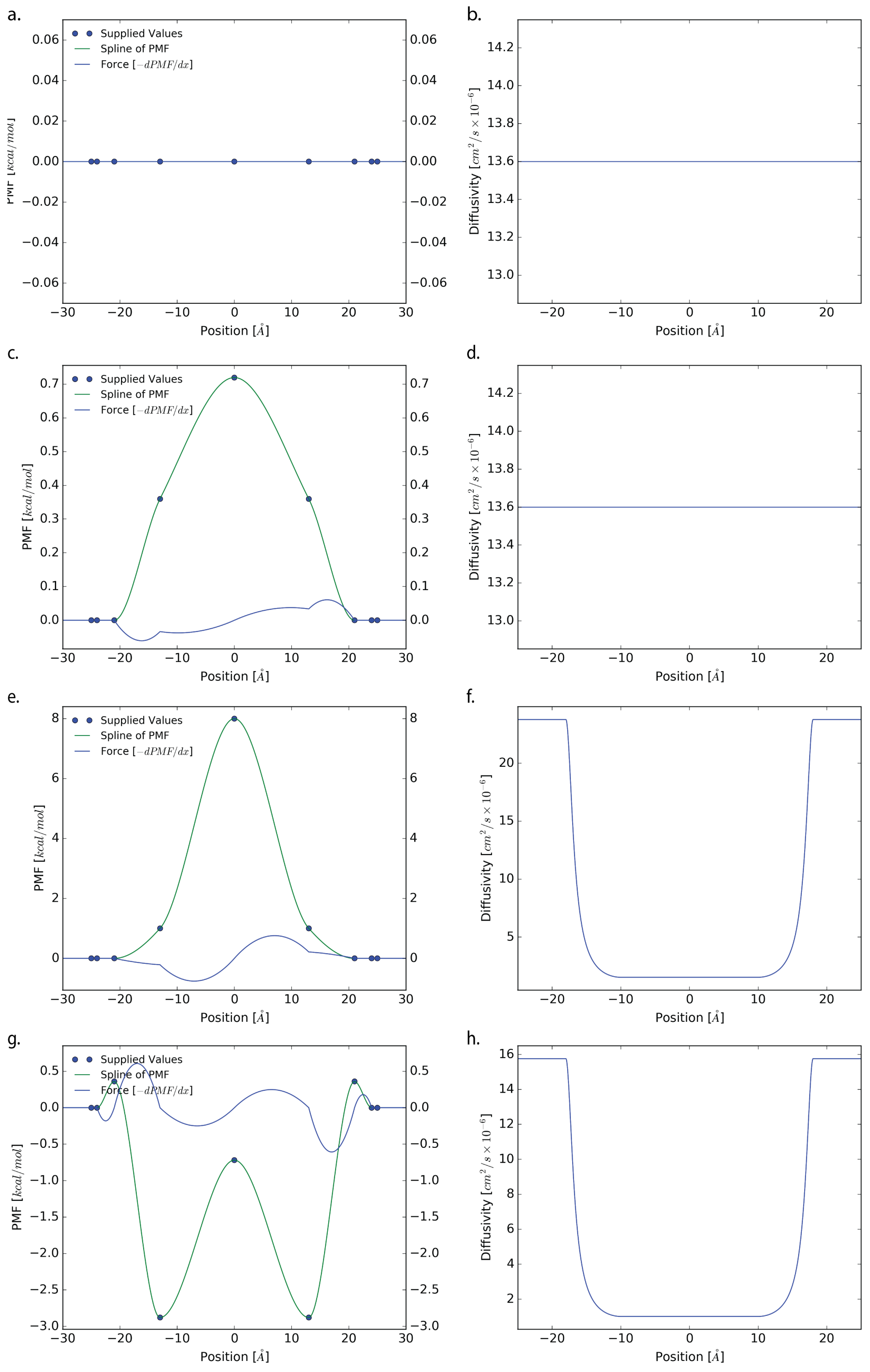


Figure 4.3: The PMF and diffusivity profiles used for each system. Each profile is represented using a piecewise cubic Hermite polynomial interpolated from a set of user specified values.

Langevin dynamics engine please refer to the SI.

Table 4.1: Parameter inputs to the Langevin dynamics simulations for each case.

	Flat	Small Barrier	Urea	Codeine
R_{hyd} (Å)	5	5	2.86	4.32
Mass (g/mol)	20	20	60	299
T (K)	298	298	298	298
Brute dt (s)	$1{\times}10^{-12}$	$1{\times}10^{-12}$	$1{\times}10^{-12}$	$1{\times}10^{-12}$
Milestone dt (s)	$1{\times}10^{-15}$	$1{\times}10^{-15}$	$1{\times}10^{-15}$	$1{\times}10^{-15}$

Using the above dynamics engine, sample trajectories under various conditions can be generated. Here the permeability is calculated via several methods: 1) theoretical ISD, 2) theoretical MFPT-ISD, 3) brute forced MFPT-ISD, 4) milestoning MFPT-ISD, 5) brute PBCP, and 6) milestoning PBCP. Methods 1, 2 are directly evaluated numerically from the interpolated profiles using the relations in the Theory section. Methods 3–6 employ brute force or milestoning calculations to generate sample trajectories which are post-processed to yield the permeability coefficient. The strategy for brute force and milestoning simulations are described below.

4.4.1 Brute Force Sampling and Statistics

Brute force attempts to sample the membrane crossing event via a single continuous trajectory were attempted as follows: Trajectories to sample $\langle t \rangle$ and ρ were intiated at one end of the membrane $z = -25$ Å, the a-surface. Trajectories to sample the MFPT employed reflecting boundary conditions at the start to prevent the particle from diffusing away into bulk. Effectively trajectories which passed the reflecting boundary at the a-surface, were place back across the boundary and velocity reversed. In contrast, the simulations to obtain ρ have an absorbing boundary placed at $z = -26$ Å, corresponding to the so-called q-surface. Trajectories that crossed the q-surface were halted and counted as non-crossing events.

For all brute force trajectories, an absorbing boundary was placed at $z = 25$ Å. Upon crossing this terminal boundary, the simulations are halted and either the crossing time or the event recorded for the

MFTP, and ρ sampling cases respectively. The crossing probability was found according to

$$\rho = \frac{n_b}{n_b + n_q}, \tag{4.39}$$

where n_b and n_q are the number of $b-$ or q-surface crossing events observed. The FPT distribution often takes the form of a Poisson or exponential distribution. We attempt to quantify the quality of the MFPT estimate by computing the root mean square error (RMSE) of the population and confidence interval by bootstrapping. The RMSE is defined as

$$\sigma_{\langle t \rangle} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(t_i - \langle t \rangle)^2}, \tag{4.40}$$

where t_i is the time that trajectory i took before touching the absorbing boundary and N is the total number of observations. We also report the 95% confidence intervals of both the MFPT and the crossing probability. This is estimated by bootstrapping using the Bias-Corrected and Accelerated method and percentile method for the MFPT and crossing probabilities respectively with 10,000 resamples[37, 38].

4.4.2 Milestoning

In this section we explain our experimental approach and include a brief description of milestoning theory. Milestones are spaced at 1 Å increments across the bilayer ($[-25, 25]$), and in the case of ρ estimation, an additional milestone was placed at $z = -26$ Å. Trajectories were initiated with a position starting at each milestone, M_i, with a random velocity sampled from the Maxwell-Boltzmann distribution. Two stages are needed to obtain milestoning statistics. According to formal milestoning theory, we must determine the first hitting point distribution (FHPD) on M_i. To find the FHPD, trajectories are started at milestone M_i and run until they cross either another milestone (M_{i+1}, M_{i-1}) or the originating milestone (M_i). If the trajectory is self-crossing then the simulation is halted and statistics discarded. Conversely, if the trajectory crosses an adjacent milestone first then this trajectory is a member of the FHPD on milestone M_i. Due to the microscopic time-reversiblity of classical mechanics, the reverse of this trajectory is a valid simulation of the crossing from ($M_{i+1}, M_{i-1} \rightarrow M_i$). Thus we term this stage the "reverse" phase. Next we begin the "forward" phase where each member of the FHPD is restarted at M_i, but with the initial velocity vectors reversed. These trajectories propagate until they cross an adjacent milestone, M_{i+1} or M_{i-1}. Self crossing events of the starting milestone M_i are ignored in the forward phase.

The forward phase provides all the transition and incubation time statistics that will be used in the milestoning analysis. As the milestones are crossed, each trajectory is counted, and the time that each trajectory took is also tracked. A transition kernel matrix $\mathbf{K}$ is then generated, whose elements K_{ij} represent the transition probability that a system starting at milestone M_i will transition to milestone M_j

$$K_{ij} = \frac{n_{i\to j}}{\sum_{k=1}^{N} n_{i\to k}}, \tag{4.41}$$

where $n_{i\to j}$ represents the total number of trajectories that cross from milestone i to milestone j, where N is the index of the final milestone. Similarly, a lifetime or incubation time vector τ is also generated, whose elements τ_i represent the average time that a system spends in milestone i

$$\tau_i = \frac{\sum_{l=1}^{n_i} t_l}{n_i}, \tag{4.42}$$

where t_l is the time that trajectory l takes to reach an adjacent milestone, and n_i is the total number of trajectories starting from milestone i.

Using $\mathbf{K}$ and τ, we can compute thermodynamic quantities such as the stationary flux ($\mathbf{q_{stat}}$) and the stationary probability ($\mathbf{p_{stat}}$), as well as the MFPT ($\langle t\rangle$), a kinetic quantity. First, we compute the stationary probability using the following eigenvalue equation

$$\mathbf{K}\mathbf{q_{stat}} = \mathbf{q_{stat}}. \tag{4.43}$$

Note that before solving Eq. 4.43, we typically have to modify $\mathbf{K}$ to have the correct boundary conditions. For our membrane system, since the two terminal milestones at each end of the membrane have no additional milestones beyond them, it does not make sense to simulate from them. We can fill in reasonable transition probabilities and lifetimes using the following scheme: Since each terminal milestone only has one adjacent milestone, we can set the transition probability to transition to the singular neighbor or the other side of the membrane with equal probability. These are periodic boundary conditions. That is, for a set of milestones $\mathbf{M} = M_1, M_2, \ldots, M_N$

$$K_{1i} = \begin{cases} 0.5 & \text{if } i = 2 \\ 0.5 & \text{if } i = N \\ 0 & \text{if } i \neq 2, N \end{cases}, \tag{4.44}$$

and

$$K_{Ni} = \begin{cases} 0.5 & \text{if } i = N-1 \\ 0.5 & \text{if } i = 1 \\ 0 & \text{if } i \neq 1, N-1 \end{cases}, \tag{4.45}$$

where N is the index of the last milestone and i is a milestone index. Splitting the transition probabilities equally makes sense because these milestones are in the bulk solvent, where the PMF is flat. For the lagtime vector τ, given that the terminal milestones are in bulk solvent, the lagtime is defined as follows

$$\tau_0 = \tau_N = \frac{\Delta x^2}{2D}, \tag{4.46}$$

where Δx is the cartesian distance to the adjacent milestone, and D is the diffusivity coefficient in bulk. With these changes, we can estimate the stationary probability according to

$$p_{stat,i} = q_{stat,i} \cdot \tau_i, \tag{4.47}$$

Where we take the element-wise product over all milestone indeces i.

The stationary probability distribution, $\mathbf{p_{stat}}$, can be used to obtain an estimate of the underlying PMF, W_i, at milestone i accordingly

$$W_i = -\beta^{-1} \log(p_{stat,i}/p_{stat,1}). \tag{4.48}$$

We use this calculated PMF to estimate the integral in Eq. 4.21 by performing a numerical integration.

To find the MFPT needed to estimate the permeability in Eq. 4.21, we must modify the transition kernel $\mathbf{K}$ once again. This time, we must make milestone M_N a "draining" state by setting all $K_{N,k} = 0, \forall k$. We also must modify transitions at milestone M_1 to transition only to its neighbor, milestone M_2:

$$K_{1i} = \begin{cases} 1 & \text{if } i = 2 \\ 0 & \text{if } i \neq 2 \end{cases}. \tag{4.49}$$

To get the MFPT, we then solve the following equation

$$\langle t \rangle = \mathbf{p_0} \cdot (\mathbf{I} - \mathbf{K})^{-1} \cdot \tau, \tag{4.50}$$

where $\mathbf{p_0}$ is a starting distribution of probabilities along the milestone. We choose $p_{0,i}$ to be 1 if $i = 1$, and choose $p_{0,i}$ to be 0 otherwise.

Finally, we obtain the crossing probability ρ for Eq. 4.30 by further modifying the transition kernel $\mathbf{K}$. Recalling that the computation of ρ requires an additional milestone at q-surface, which we assign index $i = 0$. We simulate starting from the other interior milestones. The first and last milestones become "sink" states that capture probability and never let it escape, thus; no simulations start from sink states, only end on them. Sink states are defined by setting

$$K_{0i} = \begin{cases} 1 & \text{if } i = 1 \\ 0 & \text{otherwise} \end{cases}, \tag{4.51}$$

and

$$K_{Ni} = \begin{cases} 1 & \text{if } i = N \\ 0 & \text{otherwise} \end{cases}. \tag{4.52}$$

Once again solving Eq. 4.43 we obtain a stationary flux vector, $\mathbf{q_{stat}}$, which represents the proportion of systems that would have partitioned between the sink states at the ends of the membrane. The elements of the vector $\mathbf{q_{stat}}$, must be normalized such that all its elements sum to one for this calculation, however, since $\mathbf{K}$ is a Markov matrix, the eigenvector should already be normalized. Since milestone M_N represents the a-surface, ρ in Eq. 4.30 can be extracted from the last value in $\mathbf{q_{stat}}$,

$$\rho = q_{stat,N}. \tag{4.53}$$

The error is sampled according to the scheme found in Ref. 39. For all milestoning values, we resample 1000 times and report the standard deviation of each measurable.

4.5 Results and Discussion

Below we report the computed permeability coefficients and MFPTs according to the six methods: 1) the ISD, 2) ISD from MFPT, 3) ISD with brute forced MFPT, 4) milestoning estimated $\langle t \rangle$in ISD, 5) brute force with no assumptions, and 6) milestoning with no assumptions in Table 4.2. Note again that the brute forced MFPT is performed with a reflecting boundary condition at the starting position while no such condition is applied for the brute force crossing probabilities. This explains the reported brute force MFPT without any crossings observed for the urea system.

The distribution of the mean first passage times is shown in Fig. 4.4. Due to the highly tailed nature of the MFPT, it is difficult to estimate it from brute force simulations. Milestoning greatly aids in the computation of the first passage times by reducing the total distance between points of interest. Using clever statistics, milestoning theory is capable of reconstructing the MFPT from many short simulations. For all cases, we obtain MFPTs in good agreement with the theoretical results. Deviations in both the brute force and milestoning cases are likely caused by insufficient sampling. Sampling convergence for the milestoning can be visualized by deviations in the milestoning-derived PMF from the actual PMF.

Reconstructed PMF profiles from milestoning are shown in Fig. 4.5. Here we see that the estimated potentials of mean force are relatively converged. Discrepancies are more prevalent along $-18 < z < 18$ where the membrane exists. We suspect that the increased viscosity leads to additional self-crossing events during the reverse phase thus hindering the overall sampling. The total amount of collected statistics per milestone for each system is shown in Fig. 4.7. With milestoning, we can easily adaptively sample regions of poor convergence.

By comparing and utilizing the two equations derived, Eq. 4.21 and Eq. 4.30, the difference

Table 4.2: The computed MFPT and permeability coefficients according to each method. Reported error bars for the MFPT corresponds to the RMSE while the error for the crossing probability is from bootstrapping. Error in the brute force log permeability is estimated by the mean log(P) difference of the high and low confidence intervals. [†] We observed no brute force crossing transitions for the urea system. The values in parentheses represent the number of forward-phase trajectories sampled in each case.

	Flat	**Small Barrier**	**Urea**	**Codeine**
Theoretical MFPT (s)	9.19×10^{-9}	1.11×10^{-8}	1.99×10^{-3}	2.89×10^{-7}
Brute Forced MFPT (s)	$9. \pm 7. \times10^{-9}$ (5100)	$1.1 \pm 1.0\times10^{-8}$ (2500)	$3. \pm 3.\times10^{-4}$ (1694)	$10. \pm 9.\times10^{-7}$ (1600)
Milestoning MFPT (s)	$1.15 \pm 0.05\times10^{-8}$	$1.35 \pm 0.04\times10^{-8}$	$1.9 \pm 0.3\times10^{-3}$	$4.0 \pm 0.3\times10^{-7}$
Crossing Probability (ρ)				
Brute Force	$2.7 \pm 0.8\times10^{-2}$ (1722)	$1.0 \pm 0.4\times10^{-2}$ (2500)	–[†] (1627)	$7. \pm 1.\times10^{-2}$ (1340)
Milestoning	$1.7 \pm 0.1\times10^{-2}$	$9.8 \pm 0.5\times10^{-3}$	$3.1 \pm 0.4\times10^{-8}$	$2.2 \pm 0.3\times10^{-2}$
Permeability log(cm/s)				
1) ISD	1.43	1.14	-4.33	1.44
2) MFPT-ISD	1.43	1.14	-4.33	1.44
3) Brute Force MFPT-ISD	1.43 ± 0.01	1.14 ± 0.01	-3.49 ± 0.02	0.91 ± 0.02
4) Milestoning MFPT-ISD	1.33 ± 0.03	1.06 ± 0.02	-4.3 ± 0.09	1.35 ± 0.06
5) Brute Force PBCP	1.6 ± 0.1	1.1 ± 0.2	–	2.04 ± 0.09
6) Milestoning PBCP	1.37 ± 0.03	1.13 ± 0.02	-4.13 ± 0.06	1.54 ± 0.04

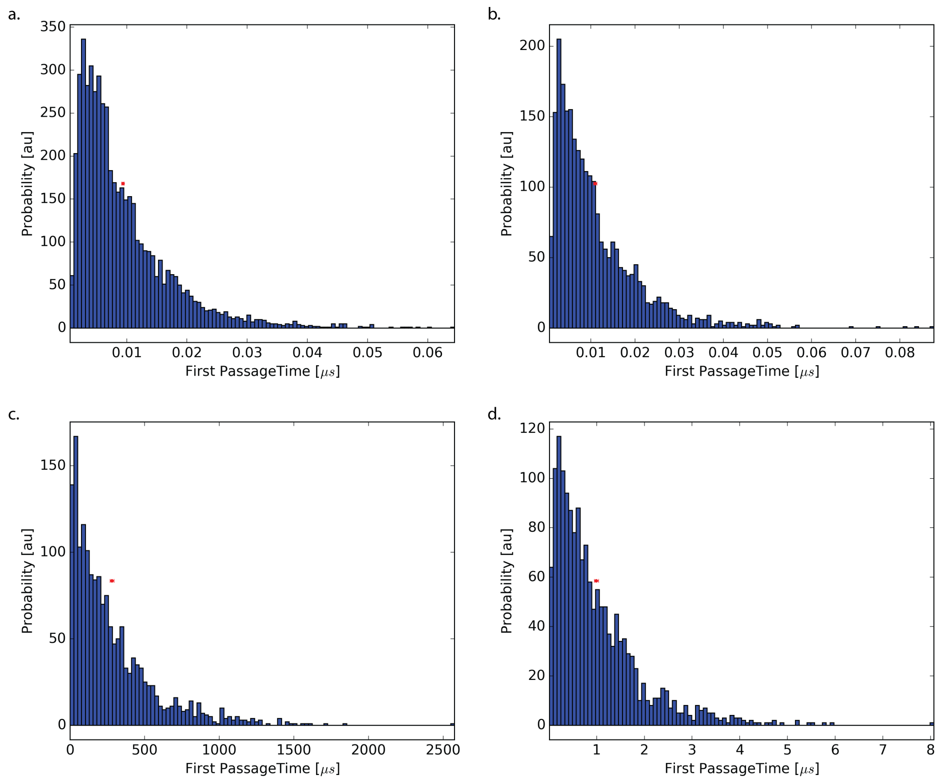


Figure 4.4: Shown here are the distributions of the brute force MFPTs for systems (a) flat, (b) small-barrier, (c) urea, and (d) codeine. Due the presence of large barriers in the urea and codeine cases, the sampling is limited. The red points and error bars indicate the mean and boostrapped 95% confidence intervals of the MFPT.

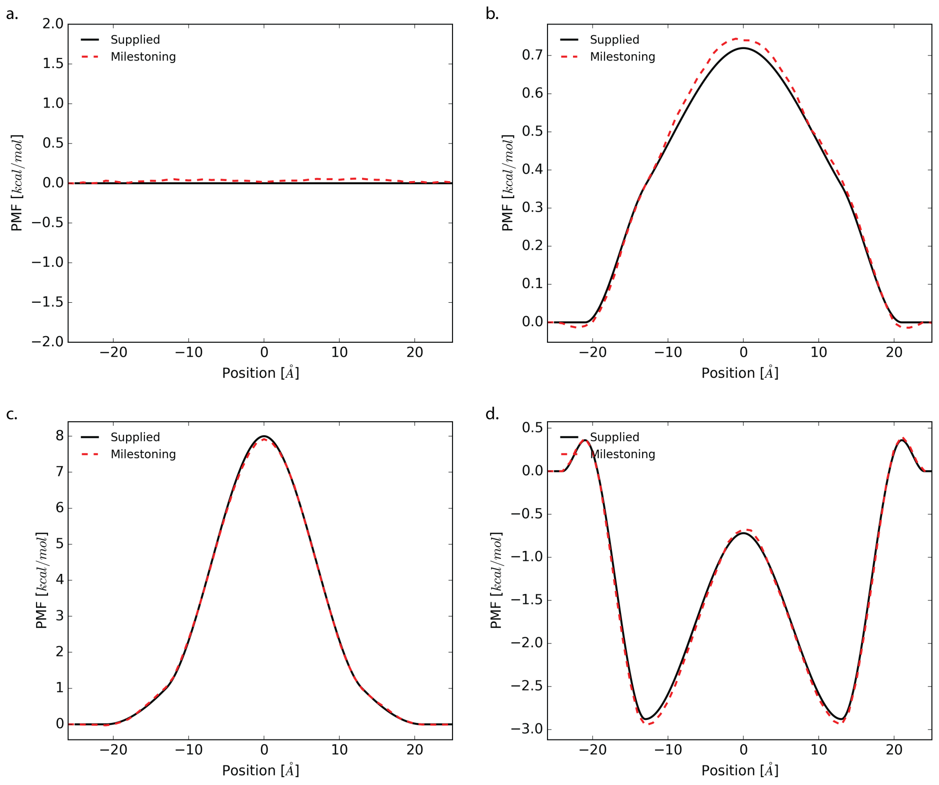


Figure 4.5: PMF profiles of each system: (a) flat, (b) small-barrier, (c) urea, and (d) codeine as estimated by milestoning, dotted red. The actual PMF is shown in black.

in the derivations must be considered for proper analysis and application. Eq. 4.21 was derived using the Smoluchowski equation, which assumes that the underlying system dynamics is overdamped. This assumption is also used in the widely employed ISD, Eq. 4.2. Our relation of the MFPT to the permeability does not require the estimation of a position dependent diffusion tensor. While there are many methods for calculation of the local diffusivity, these methods suffer largely from convergence issues[34, 40]. The theory of milestoning does not compute diffusion coefficient directly, but rather estimates rates of passage, probability, and flux. Using Eq. 4.21, the MFPT from milestoning can be used to compute permeability coefficients along with the stationary probability. To the best of our knowledge, this formulation of the ISD using the MFPT has not been shown previously.

As aforementioned, the assumption that the dynamics inside of the bilayer are over-damped is largely untested. Several instances of anomalous diffusion have been observed in crowded environments among many others[23, 41–44]. Similarly, the dynamics of water near hydrophobic surfaces is often complicated[45]. Thus we present a formula relating the permeability to the crossing probabilities, ρ, which makes no assumptions about the dampedness. While the Smoluchowski equation is used in the derivation of Eq. 4.30, it was only used in the regions containing bulk solvent. Remaining transport properties are encapsulated by the probability ρ, which, when calculated from milestoning, makes no assumptions about the dynamics.

Our scheme is similar to that proposed by Cardenas et. al.[28, 29] excepting that we obviate the need to compute the stationary flux across the first milestone by Langevin dynamics simulations. In Eq. 4.30, the flux across the first milestone is captured by our system of two Smoluchowski equations. Therefore, we eliminate a potentially cumbersome, albeit inexpensive, step in the estimation of the permeability coefficient.

We also demonstrated the utility and accuracy of both of the derived equations using a 1-D Langevin dynamics model with various PMF and diffusion profiles. In the case of the flat profile, the results of the simulations, both brute-force and milestoning, can be directly compared with theoretically derived results. For all cases, the theoretical ISD and MFPT-ISD were the same. This is not suprising, given that both are derived from equivalent expressions. Comparing the brute forced MFPT-ISD to the theoretical for the flat profile, we see that they are also comparable within error. In contrast, the flat milestoning PBCP yields a permeability coefficient that was smaller than predicted by the ISD. This discrepancy could be an example of how dampedness plays a role in the difference between the true and predicted permeability coefficients. Since the molecule simulated was not perfectly overdamped, there may be artifacts from conservation of momentum in Langevin dynamics, the formula using ρ may have captured the influence of that effect. Considering that an overdamped integrator would likely have yielded permeabilities that were more consistent between the two equations, one may question why we used the more general Langevin integrator in our test simulations. While an overdamped simulation integrator would have allowed us to apply Eq. 21 more accurately, we wanted to closely reproduce the situation that would be encountered in practical application of these equations, for instance, during an all-atom MD simulation with milestones across a bilayer, where an assumption of overdamped diffusion would not necessarily hold. For this reason, we also wanted to observe how different the predicted permeabilities would be for the same trajectories using the two different formulas. We anticipate that additional investigations of various permeabilities using all-atom molecular dynamics with milestoning may yield helpful insight about the structural characteristics of the bilayer by comparing the permeabilities estimated using Eqs. 21, 30.

The two drug-like molecules that we simulated, urea and codeine, provide insight about the potential usefulness of using milestoning to approach practical permeability problems. We obtain quantitatively accurate results for both urea and codeine. Unfortunately, no crossing events for urea were observed by brute force despite substantial efforts. The effect of increased sampling on transition probabilities is explored using bootstrapping in Fig. 4.8. Due to the presence of large transition barriers, the brute force calculation of permeability is intractable, excepting potentially the use of hyperspecialized hardware such as Anton[46]. Milestoning theory provides a framework for the estimation of membrane permeability using the MFPT in combination with transition probabilities via either the MFPT-ISD or the PBCP.

4.6 Conclusions

We have derived two useful new relations for the calculation of permeability: first, a relation of the ISD to the MFPT assuming over-damped dynamics; second, we derive the PBCP which is agnostic to the underlying diffusional dynamics. Using simulations of our toy systems, we have demonstrated the utility and validity of each method. By comparing results obtained by the two methods, one can gain insight into the deviations of the permeability coefficient when the over-damped assumption is not valid. We anticipate that both approaches will be useful to the computational biophysical community when estimating permeability coefficients across membranes, particularly in the case when milestoning is being used in combination with molecular simulation to determine permeabilities.

4.7 Acknowledgements

Chapter 4, in full, is a modified reprint of the material as it appears in “Votapka, L. W.†; Lee, C. T.†; Amaro, R. E. *Two Relations to Estimate Membrane Permeability Using Milestoning.* J. Phys. Chem. B 2016, 120 (33), 8606–8616.” The dissertation author was a primary coinvestigator and author of this work.

LWV acknowledges support from the National Science Foundation Graduate Research Fellowship Program (DGE-1144086). CTL acknowledges support from the NIH Molecular Biophysics Training Program (T32-GM008326). REA acknowledges a NIH Directors New Innovator Award DP2 OD007237, the National Biomedical Computation Resource (NBCR) NIH P41 GM103426, and supercomputing resources provided by XSEDE (NSF TG-CHE060073). REA is a co-founder of Actavalon, Inc. We also thank Gary Huber for the correction in the permeability of Eq. 4.30 due to the resistivity of the bulk water. We would like to thank Rob Swift for insightful discussions on statistics and permeation physics. We also thank Professor Andy McCammon for his tireless mentorship, his impeccable theoretical and computational contributions, and for his unsurpassed vision in leading the field for all these years.

4.8 Supplementary Information

4.8.1 Langevin Dynamics Engine

We have developed a software package which performs Langevin dynamics calculations on a user specified PMF and diffusivity profile. The engine is implemented using the Python and C programming languages. To run a simulation the user specifies a PMF and viscosity profile as well as the hydrodynamic radius and mass.

The PMF profile is generated using a piecewise cubic Hermite polynomial (PCHIP) from the following user-specified values: `dz`, the distance from the center to the edge of the membrane; `W1`, `W2`, and `W3`, the PMF at the interface, between the interface/core, and at the core of the membrane respectively; as well as `a` and `b` which correspond to the location of the water bilayer interface and interface/core. We interpolate the points using `scipy.interpolate`. Conveniently the PCHIP algorithm allows for both fast computation of the value at any point. Furthermore we can precalculate the first derivative, the negative value of which is the force, as another spline for fast evaluation. The viscosity profile is computed in a similar fashion though requiring additional inputs: `dz`, the distance from the center to edge of the membrane; `d1`, `d2`, `d3`, and `d4`, the viscocities in bulk, the interface, interface/core, and core respectively; and `a`, `b`, `c` which specify the location of the interface, interface/core, and core respectively.

The Grønbach-Jensen–Farago integrator is implemented in both Python and C. To generate Gaussian random number in C, we employ the use of the PCG random number generator (RNG) [47] to sample random integers. These random integers are converted to random uniform floating point numbers by using a bit stream algorithm by Taylor R. Campbell. Uniform floats are then transformed using the Box-Muller algorithm to yield a gaussian random number. The qualities of our C random number generator are compared to that of python library `numpy` in Fig. 4.6. The numpy and C RNGs are seeded off entropy from `/dev/urandom`. The C integrator is wrapped using the Python/C API allowing it to be called from Python which handles the system setup, configuration, and analysis.

4.8.2 Milestoning Sampling and Efficiency

The total number of statistics aggregated during milestoning for each system is shown in Fig. 4.7. Notably, the statistics consist of only the trajectories which were members of the first hitting point distribution (FHPD). To quantify the efficiency of identifying FHPD members we ran a quick test of the accept rate of the milestone at $z = 0$ with adjacent milestones at $z = -1, 1$ for 10000 random members with a timestep of 1×10^{-15} s. We find that the typical rejection rate is upwards of $\approx$90% (9000) during the reverse phase. While most rejections come about from self-crossing events occurring in the first several timesteps, and thus contribute little to the total computer time, the number of trajectories required to generate a well-explored FHPD is very large, making the compute time of rejections non-trivial. None of

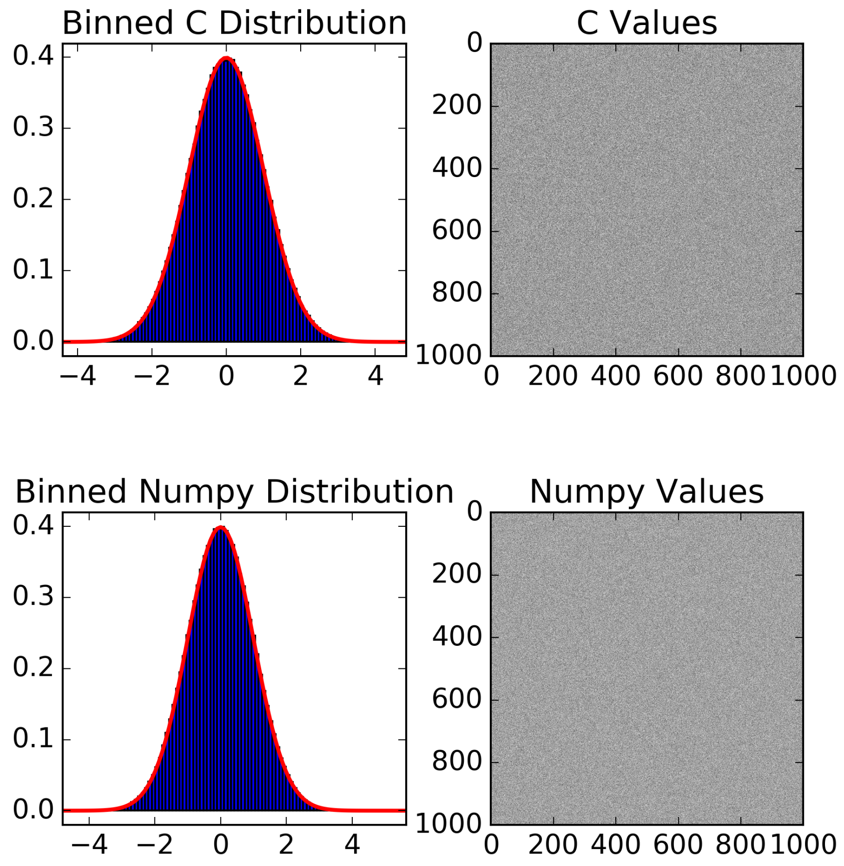


Figure 4.6: The binned probability distribution of both C and `numpy` generated gaussian random numbers are shown in comparison with an ideal Gaussian with zero mean and unit variance. On the right, a greyscale plot of the sequence of random numbers indicates, by visual inspection, that they are not correlated and there are no repeat sequences.

the reported statistics account for the number of rejected trajectories.

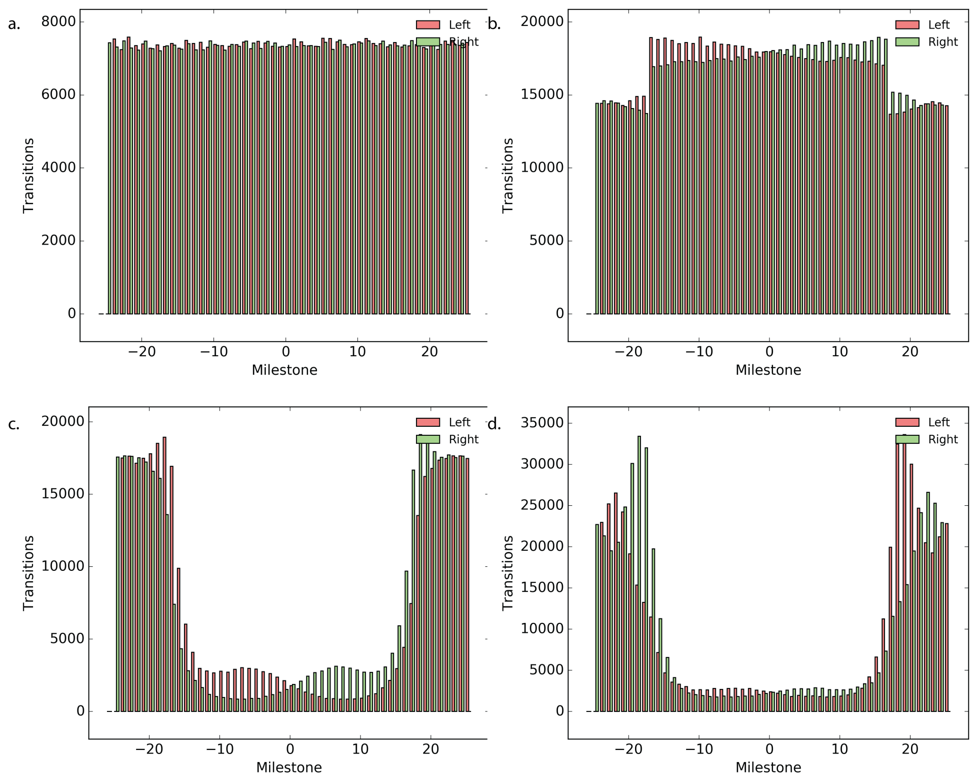


Figure 4.7: Shown here are the total number of forward and reverse trajectories collected for systems (a) flat, (b) small-barrier, (c) urea, and (d) codeine.

4.8.3 Sampling Convergence of Crossing Probabilities

To quantify the impact of sampling on our confidence in crossing probability, we compute the 95% confidence of $\rho = 0.5$ with respect to a total number of observations. Confidence intervals are estimated using the percentile boostrap over 10,000 resamples. The results are shown in Fig. 4.8. Even at $N = 200$ the confidence interval is still quite large at 0.07. The calculation of ρ is a binomial sampling problem, thus the 95% confidence interval should follow the following solution,

$$CI(95) = 2\rho/\sqrt{N-1}. \tag{4.54}$$

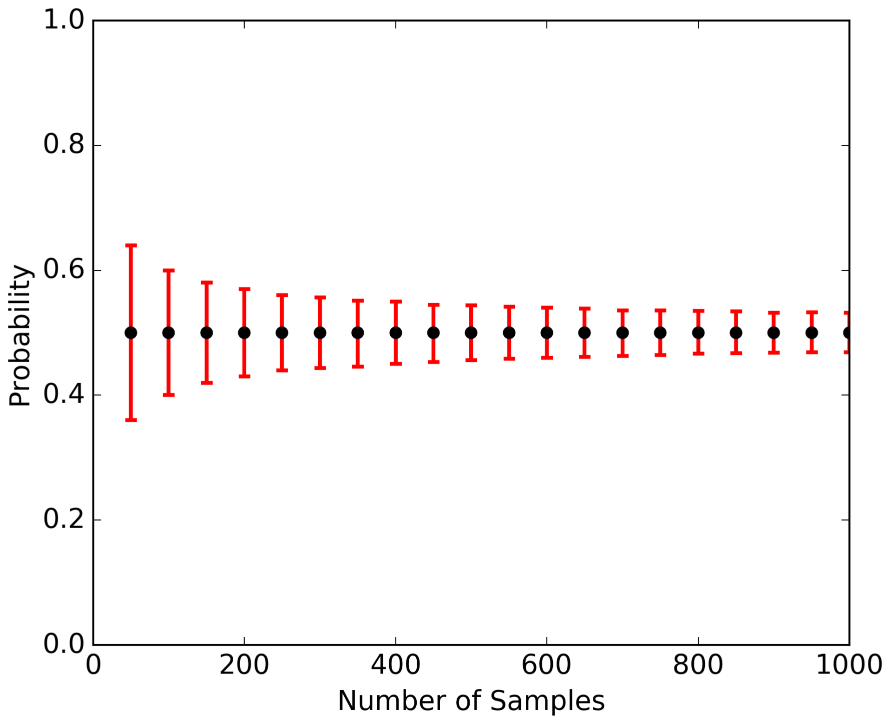


Figure 4.8: The 95% confidence intervals of a 50-50 crossing probability with respect to the total number of observations.

References

(1) Kola, I.; Landis, J. *Nat. Rev. Drug Discov.* **2004**, *3*, 711–715.

(2) Swift, R. V.; Amaro, R. E. *Chem. Biol. Drug Des.* **2013**, *81*, 61–71.

(3) Marrink, S.-J.; Berendsen, H. J. C. *J. Phys. Chem.* **1994**, *98*, 4155–4168.

(4) Woolf, T. B.; Roux, B. *J. Am. Chem. Soc.* **1994**, *116*, 5916–5926.

(5) Bemporad, D.; Essex, J. W.; Luttmann, C. *J. Phys. Chem. B* **2004**, *108*, 4875–4884.

(6) Xiang, T.-X.; Anderson, B. D. *Adv. Drug Deliv. Rev.* **2006**, *58*, 1357–1378.

(7) Awoonor-Williams, E.; Rowley, C. N. *Biochim. Biophys. Acta - Biomembr.* **2016**, *1858*, 1672–1687.

(8) Wilson, M. A.; Pohorille, A. *J. Am. Chem. Soc.* **1996**, *118*, 6580–6587.

(9) Grossfield, A.; Woolf, T. B. *Langmuir* **2002**, *18*, 198–210.

(10) Ulander, J.; Haymet, A. D. J. *Biophys. J.* **2003**, 3475–3484.

(11) MacCallum, J. L.; Bennett, W. F. D.; Tieleman, D. P. *J. Gen. Physiol.* **2007**, *129*, 371–377.

(12) Johansson, A. C. V.; Lindahl, E. *Proteins* **2008**, *70*, 1332–1344.

(13) MacCallum, J. L.; Bennett, W. F. D.; Tieleman, D. P. *Biophys. J.* **2008**, *94*, 3393–4304.

(14) Bauer, B. A.; Lucas, T. R.; Meninger, D. J.; Patel, S. *Chem. Phys. Lett.* **2011**, *508*, 289–294.

(15) Tejwani, R. W.; Davis, M. E.; Anderson, B. D.; Stouch, T. R. *J. Pharm. Sci.* **2011**, *100*, 2136–2146.

(16) MacCallum, J. L.; Bennett, W. F. D.; Tieleman, D. P. *Biophys. J.* **2011**, *101*, 110–117.

(17) Paloncýová, M.; Berka, K.; Otyepka, M. *J. Chem. Theory Comput.* **2012**, *8*, 1200–1211.

(18) Riahi, S.; Rowley, C. N. *J. Am. Chem. Soc.* **2014**, *136*, 15111–15113.

(19) Carpenter, T. S.; Kirshner, D. A.; Lau, E. Y.; Wong, S. E.; Nilmeier, J. P.; Lightstone, F. C. *Biophys. J.* **2014**, *107*, 630–641.

(20) Issack, B. B.; Peslherbe, G. H. *J. Phys. Chem. B* **2015**, *119*, 9391–9400.

(21) Bemporad, D.; Luttmann, C.; Essex, J. W. *Biochim. Biophys. Acta - Biomembr.* **2005**, *1718*, 1–21.

(22) Comer, J.; Schulten, K.; Chipot, C. *J. Chem. Theory Comput.* **2014**, *10*, 554–564.

(23) McGuffee, S. R.; Elcock, A. H. *PLoS Comput. Biol.* **2010**, *6*, e1000694.

(24) Faradjian, A. K.; Elber, R. *J. Chem. Phys.* **2004**, *120*, 10880–9.

(25) Májek, P.; Elber, R. *J. Chem. Theory Comput.* **2010**, *6*, 1805–1817.

(26) Kirmizialtin, S.; Elber, R. *J. Phys. Chem. A* **2011**, *115*, 6137–6148.

(27) Cardenas, A. E.; Jas, G. S.; DeLeon, K. Y.; Hegefeld, W. A.; Kuczera, K.; Elber, R. *J. Phys. Chem. B* **2012**, *116*, 2739–2750.

(28) Cardenas, A. E.; Elber, R. *Mol. Phys.* **2013**, *111*, 3565–3578.

(29) Cardenas, A. E.; Elber, R. *J. Chem. Phys.* **2014**, *141*, 054101.

(30) Hardt, S. L. *Bull. Math. Biol.* **1981**, *43*, 89–99.

(31) Schulten, K. *J. Chem. Phys.* **1981**, *74*, 4426.

(32) Nadler, W.; Schulten, K. *J. Chem. Phys.* **1985**, *82*, 151.

(33) Northrup, S. H.; Allison, S. A.; McCammon, J. A. *J. Chem. Phys.* **1984**, *80*, 1517.

(34) Lee, C. T.; Comer, J.; Herndon, C.; Leung, N.; Pavlova, A.; Swift, R. V.; Tung, C.; Rowley, C. N.; Amaro, R. E.; Chipot, C.; Wang, Y.; Gumbart, J. C. *J. Chem. Inf. Model.* **2016**, *56*, 721–733.

(35) Ortega, A.; Amorós, D.; Garcia de la Torre, J. *Biophys. J.* **2011**, *101*, 892–8.

(36) Grønbech-Jensen, N.; Farago, O. *Mol. Phys.* **2013**, *111*, 983–991.

(37) Efron, B. *J. Am. Stat. Assoc.* **1987**, *82*, 171–185.

(38) Efron, B.; Tibshirani, R. J., *an Introduction to the Bootstrap*; Chapman and Hall: London, 1993.

(39) Votapka, L. W.; Amaro, R. E. *PLoS Comput. Biol.* **2015**, *11*, e1004381.

(40) Mamonov, A. B.; Kurnikova, M. G.; Coalson, R. D. *Biophys. Chem.* **2006**, *124*, 268–278.

(41) Sokolov, I. M. *Soft Matter* **2012**, *8*, 9043.

(42) Banks, D. S.; Fradin, C. *Biophys. J.* **2005**, *89*, 2960–71.

(43) Regner, B. M.; Vučinić, D.; Domnisoru, C.; Bartol, T. M.; Hetzer, M. W.; Tartakovsky, D. M.; Sejnowski, T. J. *Biophys. J.* **2013**, *104*, 1652–60.

(44) Berezhkovskii, A. M.; Dagdug, L.; Bezrukov, S. M. *Biophys. J.* **2014**, *106*, L09–11.

(45) Setny, P.; Baron, R.; Kekenes-Huskey, P. M.; McCammon, J. A.; Dzubiella, J. *Proc. Natl. Acad. Sci. USA* **2013**, *110*, 1197–202.

(46) Shaw, D. E.; Chao, J. C.; Eastwood, M. P.; Gagliardo, J.; Grossman, J. P.; Ho, C. R.; Lerardi, D. J.; Kolossváry, I.; Klepeis, J. L.; Layman, T.; McLeavey, C.; Deneroff, M. M.; Moraes, M. A.; Mueller, R.; Priest, E. C.; Shan, Y.; Spengler, J.; Theobald, M.; Towles, B.; Wang, S. C.; Dror, R. O.; Kuskin, J. S.; Larson, R. H.; Salmon, J. K.; Young, C.; Batson, B.; Bowers, K. J. *Commun. ACM* **2008**, *51*, 91.

(47) O'Neill, M. E. *PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation*; tech. rep. HMC-CS-2014-0905; Claremont, CA: Harvey Mudd College, 2014.

Chapter 5

Quantitative Ranking of Ligand Binding Kinetics with a Multiscale Milestoning Simulation Approach

5.1 Abstract

Efficient prediction and ranking of small molecule binders by their kinetic (k_{on} and k_{off}) and thermodynamic (ΔG) properties can be a valuable metric for drug lead optimization, as these quantities are often indicators of *in vivo* efficacy. We have previously described a hybrid molecular dynamics, Brownian dynamics, and milestoning model, Simulation Enabled Estimation of Kinetic Rates (SEEKR), that can predict k_{on}'s, k_{off}'s, and ΔG's. Here we demonstrate the effectiveness of this approach for ranking a series of seven small molecule compounds for the model system, β-cyclodextrin, based on predicted k_{on}'s and k_{off}'s. We compare our results using SEEKR to experimentally determined rates as well as rates calculated using long-timescale molecular dynamics simulations and show that SEEKR can effectively rank the compounds by k_{off} and ΔG with reduced computational cost. We also provide a discussion of convergence properties and sensitivities of calculations with SEEKR to establish "best practices" for its future use.

5.2 Results and Conclusions

Molecular binding processes are ubiquitous in biology and serve as the fundamental basis for biological complexity. For the drug discovery community, engineering pharmacologically active small molecules is of particular importance. Traditionally, the paradigm for lead optimization is to select for leads with the greatest affinity for a protein, or other, target of interest. However, recent evidence suggests that the kinetics of binding may also be a useful metric for lead selection. It is now thought that both residence times and association rates are key determinants of *in vivo* efficacy for many drugs[1–5]. Similar to computational predictions of binding thermodynamics, molecular simulations can be used to compute binding kinetics[6–8]. Methods such as Brownian Dynamics (BD) have been used effectively for estimating molecular association rates[9–12]. Molecular dynamics (MD) simulations which explicitly represent all atoms and forces, can also be used to predict binding kinetics. Due to significantly increased model complexity, MD is limited by sampling. Nevertheless, owing to software improvements and the development of commodity hardware such as GPUs and specialty hardware such as Anton[13, 14], "brute force" calculation of binding kinetics with MD is now a possibility[15–19]. To improve upon "brute force" sampling statistics, many sampling strategies employ force biases or other statistical mechanical techniques to predict both association and dissociation rates of many systems. This includes methods such as: Markov State Models[20–23], metadynamics[24–26], milestoning[27–32], and other techniques[33–39].

Our previous work uses a multiscale MD, BD, and Milestoning approach for the calculation of both association and dissociation rates of receptor-ligand complexes[40, 41]. Our implementation, "Simulation Enabled Estimation of Kinetic Rates" (SEEKR) is a freely available software package[1] that automates the preparation, simulation, and analysis of these multiscale milestoning calculations using existing softwares: NAMD[42] for MD simulations and BrownDye[12] for BD simulations. Milestoning theory provides the glue for the multiscale scheme by providing a strategy to subdivide, simulate, and subsequently statistically reconnect small regions of simulation space called "milestones"[27, 40, 41, 43–52] This approach reduces the compute time required to simulate transition events, is embarrassingly parallel, and is agnostic to the simulation modality used. This allows us to use atomically detailed, yet computationally expensive,

[1]https://amarolab.ucsd.edu/seekr

fully flexible MD simulations in milestones near the binding site where these interactions are critical for understanding the binding and unbinding, and BD simulations far from the binding site where rigid body dynamics provides a sufficient description at significantly reduced computational cost. For a more thorough description of milestoning theory and the calculation of kinetic quantities, such as k_{on} and k_{off}, we refer the reader to the existing literature.[27, 40, 41, 43–52]

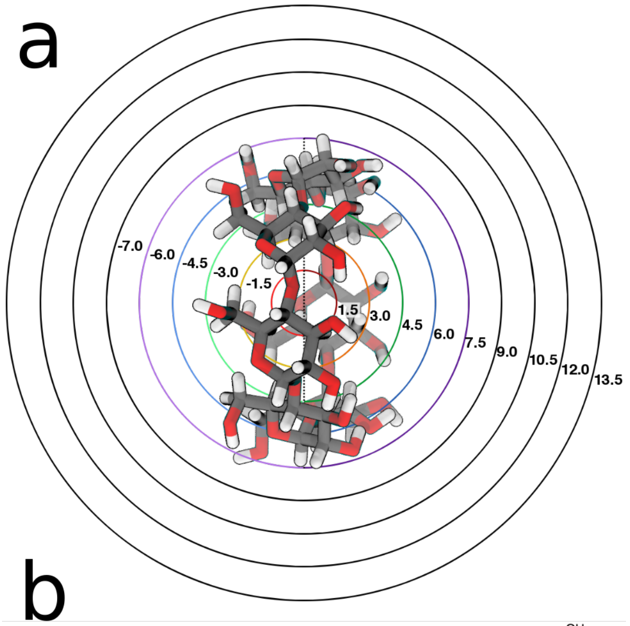
The effectiveness of the SEEKR scheme for the calculation of k_{on} and k_{off} values has been demonstrated for multiple protein-ligand systems[40, 41]. However, it has not yet been used for rank ordering sets of compounds by kinetic (k_{on} and k_{off}) and thermodynamic (ΔG) values, as would be done in pharmaceutical discovery settings. Here we use SEEKR to estimate k_{on}, k_{off}, and ΔG for a model host-guest system, β-cyclodextrin with seven ligands representing diverse chemical groups (Fig. 5.1), using two forcefields for β-cyclodextrin GAFF[53, 54] and Q4MD[55]. We compare the SEEKR estimates with previously published "brute force" (long-timescale) MD predictions[19] and experimental results[56–61]. Using this model system we examine both the accuracy and efficiency of SEEKR compared to long-timescale MD. We further explore the reduction in computational effort required for SEEKR estimates as well as discuss the convergence properties and sensitivity of SEEKR calculations to establish "best practices" for its future use.

SEEKR calculations and the long timescale MD simulations struggle to reproduce both the values and rank ordering of the experimentally determined k_{on}'s (Fig. 5.2).

However, similar qualitative results are seen with the SEEKR calculations and long timescale MD calculations using the same forcefield. On rates calculated using Q4MD are approximately one order of magnitude faster than experimental rates, while the GAFF forcefield produces rates closer to the experimental values, differing by approximately a factor of 3 or less.

Both methods fail to effectively order the ligands by increasing k_{on}, as demonstrated by low or negative Kendall and Spearman rank correlation coefficients. As the values of all the experimental rates have limited variability (all within half an order of magnitude), the sensitivity of the methods as well as the errors associated with the calculations and experiments makes differentiation and ordering challenging.

Unlike the experimental k_{on}'s, k_{off}'s for the seven guest molecules span multiple orders of magnitude, making them a better target for ranking the compounds with SEEKR. Again, off rates calculated with



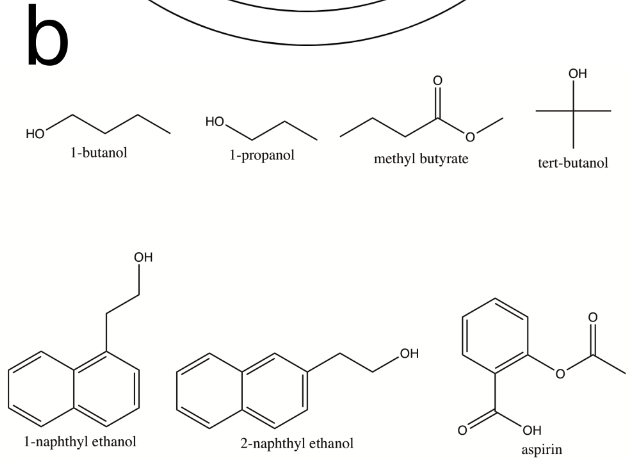
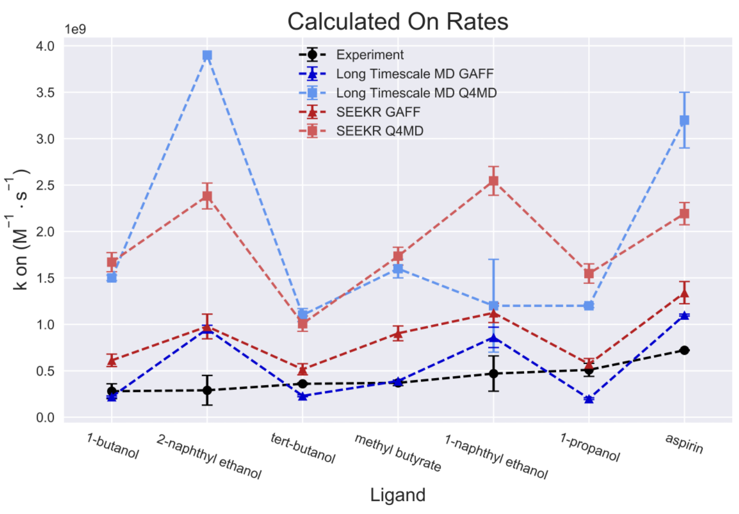
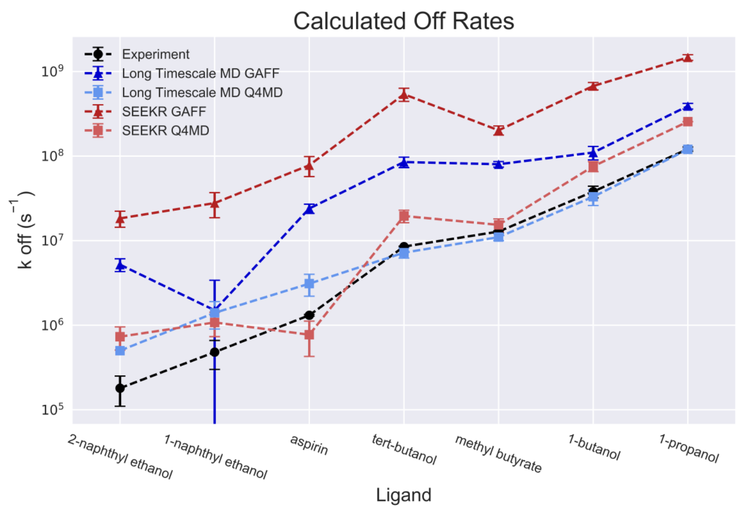


Figure 5.1: a) β-cyclodextrin with milestones spaced at 1.5 Å increments and b) the seven ligands used in this study. The top four ligands are known to bind more weakly while the bottom three are known to bind more tightly.



Method	**Kendall**	**Spearman**
SEEKR GAFF	$-0.20\ \pm 0.31$	$-0.29\ \pm 0.40$
SEEKR Q4MD	$0.14\ \pm 0.29$	$0.14\ \pm 0.38$
Long Timescale MD GAFF	$0.24\ \pm 0.26$	$0.25\ \pm 0.29$
Long Timescale MD Q4MD	$0.00\ \pm 0.28$	$-0.05\ \pm 0.37$

Figure 5.2: a) Experimental and calculated on rates for SEEKR GAFF and Q4MD forcefields as well as long timescale MD with both forcefields. b) Calculated rank correlation coefficients. Errors are determined with a bootstrapping analysis.

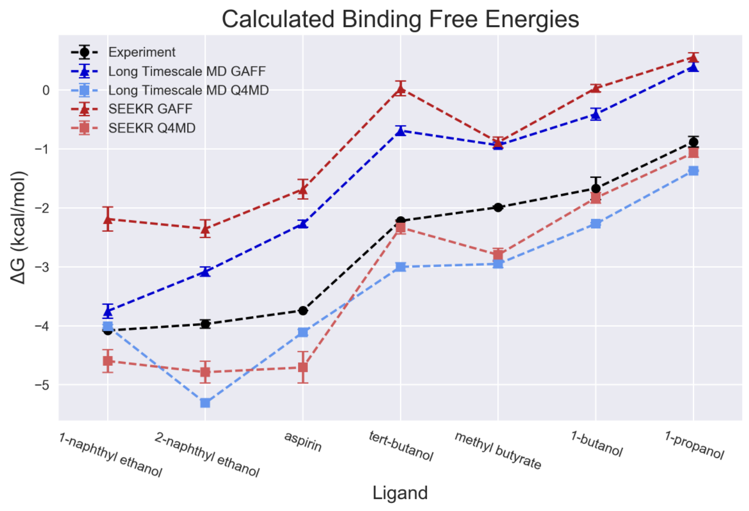


Method	Kendall	Spearman
SEEKR GAFF	0.90 $\pm$0.06	0.96 $\pm$0.04
SEEKR Q4MD	0.81 $\pm$0.09	0.93 $\pm$0.05
Long Timescale MD GAFF	0.81 $\pm$0.09	0.93 $\pm$0.04
Long Timescale MD Q4MD	1.00 $\pm$0.05	1.00 $\pm$0.03

Figure 5.3: a) Experimental and calculated off rates for SEEKR GAFF and Q4MD forcefields as well as long timescale MD with both forcefields. b) Calculated rank correlation coefficients. Errors are determined with a bootstrapping analysis.

SEEKR are in good agreement with the long timescale MD simulations using the same forcefield (Fig. 5.3). Rates calculated using the GAFF forcefield are consistently faster than experiment by approximately one order of magnitude.

This trend is seen in both the long timescale MD and SEEKR, but is more pronounced in the SEEKR calculations. The Q4MD forcefield, however, more accurately reproduces the magnitude of the experimental values with both SEEKR and long timescale MD. SEEKR calculations with both Q4MD and GAFF forcefields were effective for ranking the compounds by increasing off rates, as evidenced by high rank correlation values. The smaller magnitudes of the Q4MD values potentially contribute to this forcefield's difficulty to differentiate between compounds with similar rates, where the larger values associated with the GAFF forcefield allow for more variability in the rate value without changing the overall ordering. Both the GAFF and Q4MD forcefields successfully differentiate the three tighter binding



Method	Kendall	Spearman
SEEKR GAFF	0.88 ±0.08	0.96 ±0.05
SEEKR Q4MD	0.73 ±0.10	0.89 ±0.06
Long Timescale MD GAFF	0.90 ±0.07	0.96 ±0.04
Long Timescale MD Q4MD	0.87 ±0.11	0.94 ±0.06

Figure 5.4: a) Experimental and calculated binding free energies for SEEKR GAFF and Q4MD forcefields as well as long timescale MD with both forcefields. b) Calculated rank correlation coefficients. Errors are determined with a bootstrapping analysis.

compounds from the four weaker binding, with the tighter binding compounds all having slower off rates and a difference of one order of magnitude between the fastest tightly binding compound and the slowest weakly binding compound. This suggests that SEEKR could be useful for identifying and separating long residence time ligands from shorter residence time ligands and then further discriminating the compounds through ranking by k_{off}.

An additional benefit of kinetics calculations with SEEKR is that binding free energies can also be obtained from the same simulations (Fig. 5.4).

Binding free energies calculated using the rate constants are most heavily influenced by the k_{off} for these ligands, as this value is more variable, where the k_{on}'s for all ligands are more similar. Therefore, similar trends are observed for the calculated binding free energies as were observed for the off rates. Binding free energies can also be calculated using the stationary probabilities for each milestone, rather than the rate constants, and produce similar results. The GAFF forcefield consistently underestimates the binding free energies in both SEEKR and the long timescale MD, resulting from the consistent underestimation of the magnitudes of the k_{off}'s. The magnitudes of the binding free energies calculated using Q4MD are in much better agreement with the experimental values, differing by 1 kcal or less. SEEKR with both Q4MD and GAFF successfully differentiates the three known tighter binding compounds from the four weaker binding compounds. SEEKR can also further discriminate ligands by its effective ranking by binding free energies, demonstrated by high rank correlation values.

A key aspect of future development of the SEEKR software is the systematic development of methodological best-practices as well as the elucidation of the sensitivity of calculated kinetic parameters to various SEEKR input conditions. While it is possible to determine the kinetics for small systems like β-cyclodextrin using conventional long timescale MD simulations, increasing system size soon makes this inefficient or even impossible. The convergence of k_{on} and k_{off} were assessed by calculating the rate constants as a function of the reversal trajectory number at increasing intervals of 50 reversal numbers (with 10 trajectories initiated for each reversal number). The reversal number is a direct measure of the equilibrium simulation length, as reversals are initiated from evenly spaced configurations of the equilibrium distribution. In general, both the on and off rates appear converged in less than the maximum number of reversals available. Approximately half the total reversals (4000 of 8000) were sufficient for obtaining reasonably converged rate constants. This suggests that the total simulation cost to obtain a similar result could be as little as 2 μs per ligand, rather than 3.8 μs.

Convergence of the rate constant is a highly complicated quantity dependent on the transition probabilities as well as the incubation times obtained from each milestone. Therefore, a more detailed analysis of the convergence of these quantities on a per milestone level can provide further insight into the overall convergence of a rate calculation within SEEKR. Fig. 5.5a,b shows the convergence of k_{on} and k_{off}, respectively, as a function of the number of reversals launched for the representative system of Q4MD β-cyclodextrin with aspirin. Reversal number is directly related to the length of equilibrium sampling, the current bottleneck of a SEEKR calculation. While both values appear to converge in fewer than the maximum number of reversals, the dramatic change in k_{off} after reversal number 400 is of note. Further analysis of the per-milestone transition counts (Fig. 5.5c) and incubation times (Fig. 5.5d) revealed that this change in k_{off} was due to poor initial sampling of the -1.5 Å milestone, which once sampled decreased the overall k_{off}.

This was only observed for the aspirin ligand, one of the bulkiest ligands, where it was extremely unlikely to observe transitions outward from the primary face due to steric effects. Evaluation of these convergence properties on a per milestone basis is a valuable diagnostic tool; identifying which milestones contribute most to the mean first passage time (and therefore k_{on} and k_{off}) and milestones where the ligand spends only a short time, providing detailed molecular insight into the binding and unbinding processes. Furthermore, this analysis is also useful during the simulation process, as the convergence of each milestone can be assessed "on the fly" and individual simulations can be terminated or extended accordingly for each milestone.

We also explore the sensitivity of the calculated rate constants to the milestoning model construction. In particular, the appropriate spacing of milestones is critical for the calculation. Milestones must not be spaced so close such that the velocity of the system cannot decorrelate between transitions [45, 46]. This assumption is typically valid for molecular dynamics simulations, as velocities typically decorrelate on the subpicosecond timescale [46]. However, if milestones are spaced far apart, transitions will require much longer simulations and milestioning sampling efficiency is lost.

For our systems, the incubation times of all milestones are on the order of multiple picoseconds or greater, which is longer than the sub-picosecond timescale typically necessary for decorrelation[46]. When the milestone spacing was doubled to 3 Å, simulation efficiency was dramatically reduced, such that few to no transitions between milestones were observed, precluding the calculation of rate constants. These observations suggest that the 1.5 Å spacing used in our simulations was appropriate for the calculation of the desired kinetic parameters.

Our milestoning model differentiates the two faces of the cyclodextrin ring and therefore defines two bound states, corresponding to each face. Investigation into the effect of this on the resulting rate constants revealed that it had only minimal effects. When the two bound states were combined into a single milestone, only small changes to the rate were observed, within the error of both calculations. Furthermore, when the the two faces were not differentiated with unique milestones, minimal change in the calculated rate constants was observed.

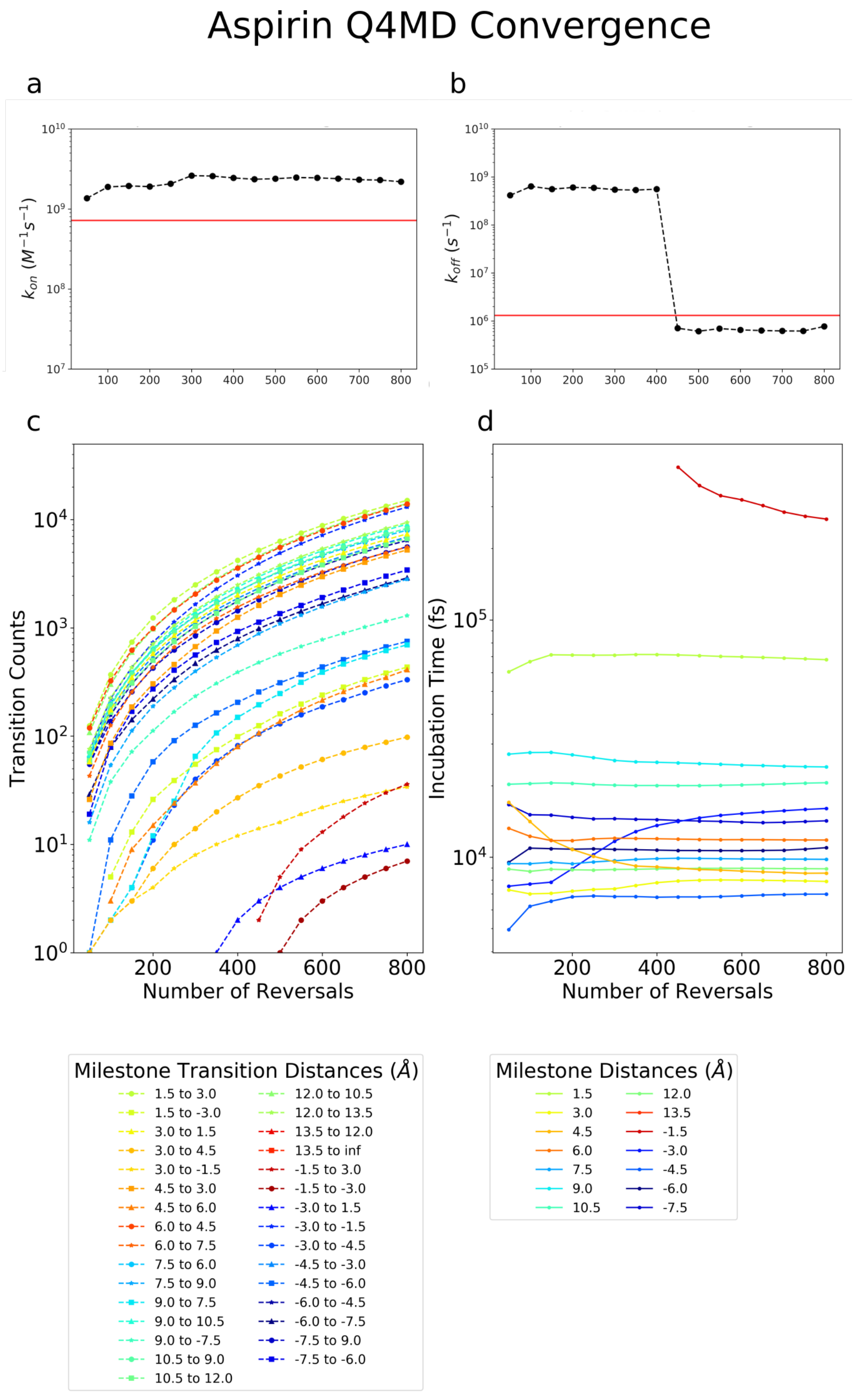


Figure 5.5: Convergence analysis for a representative ligand, aspirin, and β-cyclodextrin with the Q4MD forcefield. Convergence of a) k_{on}, b) k_{off}, c) transition counts, and d) incubation times for each milestone are plotted as a function of the number of reversals launched. Reversal number is directly related to the length of equilibrium sampling, as reversals were launched at 2 ns intervals from the equilibrium trajectory.

It is also important to note that our milestoning model did not explicitly resolve the ligand orientation in any way, and therefore any ligand orientational sampling was achieved entirely through simulation. This resulted in some ligand orientations being unsampled in the deepest milestones where the orientation was sterically restricted to the starting conformation on that milestone. While this is a limitation that will be addressed in future developments of SEEKR, it also highlights that a relatively simplistic model was able effectively calculate kinetic parameters with good agreement to experimental values.

The simplicity of this model has many advantages. The bound state is defined naturally as the innermost milestone and all other milestones can be defined at the same time, including what defines the unbound state. The long timescale MD employed a more empirical definition of the bound state where the ligand was only considered bound when the COM of the ligand was within 7.5 Å of the COM of the β-cyclodextrin for at least 1.0 ns. Similarly the ligand was considered unbound when it left this 7.5 Å bound state for at least 1.0 ns. With the SEEKR approach, minimal prior knowledge of the system is required, as binding and unbinding are determined only from the milestone surfaces. No time cutoff is required, as short excursions that do not result in full binding and unbinding events are captured naturally in the milestoning model. The simplicity of SEEKR milestoning calculation setup, in conjunction with the ability to monitor convergence for each milestone and terminate simulations accordingly, makes this approach well-suited for calculations with multiple ligands as would be necessary in a drug discovery setting.

We present the first successful ranking of a set of seven guest molecules with the β-cyclodextrin host using the SEEKR hybrid MD/BD/Milestoning approach. SEEKR effectively reproduces both the magnitudes and rankings of the experimental off rates[56–61] and binding free energies, two quantities of interest in typical drug discovery campaigns[1–4]. SEEKR also successfully differentiates the known longer residence time and tighter binding compounds from the weaker binding compounds. Our results are also in good agreement with previously conducted long timescale MD simulations for the same set of ligands[19]. In particular SEEKR and long timescale MD simulations using the same forcefield (GAFF or Q4MD) exhibited similar deviations from the experimental values, with GAFF producing consistently faster off rates than experiment and Q4MD producing consistently faster on rates. In general both methods and both forcefields struggled to reproduce the experimental on rate ranking, as all ligands had very similar k_{on}'s. The SEEKR method requires less simulation time (3.8 μs per ligand) than the long timescale MD approach (4.5 - 11 μs per ligand). Furthermore, convergence analysis of the SEEKR calculations suggests that comparable results could be achieved with as little as 2 μs per ligand. In addition, SEEKR's milestoning approach makes these calculations highly parallel, as the simulations on each milestone are completely independent from all other milestones. We also provide an analysis of the sensitivity of the SEEKR calculations to various factors such as sampling, milestone spacing, and the construction of the milestoning model with the intention of putting forth "best practices" for the use of SEEKR. SEEKR's effectiveness at ranking compounds for this small model system suggest that it is well suited for ranking compounds of more complex protein-ligand and protein-drug systems, where the efficiency and enhanced sampling advantages of our multiscale MD/BD/milestoning approach will be more apparent.

5.3 Computational Methods

GAFF[53, 54] forcefield parameters for the seven guest molecule along with both GAFF and Q4MD-CD[55] parameterizations of β-cyclodextrin were obtained from Tang and Chang[19]. For comparison we use identical structures and parameterizations as those used in their study. These initial structures were used by the SEEKR software for preparation of the milestoning simulations. The preparation procedure was the same for each of the seven guest molecules and followed standard SEEKR protocols[41]. All systems were solvated with TIP3P waters[62]. All BD simulations were performed using the BrownDye software package[12]. Electrostatic potentials of the host and guest molecules used as inputs for the BD simulation were calculated with APBS version 1.4[63]. A modified version of NAMD 2.12 was used for all MD simulations[42]. For all 13 milestones in the MD region, the standard SEEKR procedure for minimization, equilibration and simulation was followed. In total, 2.6 μs of equilibrium sampling (160 ns for 16 milestones) were used and approximately 570 ns of FHPD sampling for a total of 3.2 μs of simulation used in the milestoning model. The total cost per ligand (including simulation discarded for equilibration) was therefore $\sim$3.8 μs.

5.4 Acknowledgements

Chapter 5, in full, is a modified reprint of the material as it appears in “Jagger, B. R.; Lee, C. T.; Amaro, R. E. *Quantitative Ranking of Ligand Binding Kinetics with a Multiscale Milestoning Simulation Approach.* J. Phys. Chem. Lett. 2018, 9 (17), 4941–4948.” The dissertation author was an investigator, supervisor, and coauthor of this work.

We thank Zhiye Tang and Chia-en Chang for sharing structures and parameters as well as helpful discussions. This work is funded in part by the Director’s New Innovator Award Program NIH DP2-OD007237, the National Biomedical Computation Resource (NBCR) NIH P41-GM103426, and the National Science Foundation through XSEDE supercomputing resources provided via TG-CHE060073 to R.E.A. B.R.J. and C.T.L. also acknowledge support from the NIH Molecular Biophysics Training Program (T32-GM008326).

5.5 Supplemental Information

5.5.1 Methods

System Preparation

GAFF[53, 54] forcefield parameters for the seven guest molecule along with both GAFF and Q4MD-CD[55] parameterizations of β-cyclodextrin were obtained from Tang and Chang[19]. For comparison we use identical structure and parameterizations as those used in their study. These initial structures were used by the SEEKR software for preparation of the milestoning simulations. The preparation procedure was the same for each of the seven guest molecules and followed standard SEEKR protocols[41]. All systems were solvated with TIP3P waters[62].

Preparation of milestoning simulations with SEEKR The bound state of the host-guest complex was defined as the center of mass (COM) of the β-cyclodextrin. The guest molecule was considered to be bound when its COM was within 1.5 Å of the bound state coordinates. From this bound state, spherical milestones were defined in increasing 1.5 Å increments from 1.5 Å to 13.5 Å. Furthermore, spherical milestones of radius 7.5 Å and less, were divided into two half-spheres to better capture the asymmetries between the two faces. When the ligand is less than 7.5 Å away from the bound state, it is trapped on a particular face due to the size of the host molecule. For milestone distances greater than 7.5 Å, the ligand was found to freely sample both faces, and therefore a single spherical milestone was sufficient for sampling host-guest interactions. In total, 14 unique milestones were defined. In practice, this was achieved via post-processing the simulations on each face to identify any trajectories that crossed from one face to the other, and modifying the transitions accordingly in the milestoning model. In addition, two simulations were conducted for the outermost milestones (one with the ligand initiated on each face), and these were then combined into a single milestone (with double the sampling) for milestones that were not restricted to a particular face.

The first 13 milestones correspond to the MD region, while the 14th and outermost milestone corresponds to the BD region. The standard SEEKR preparation protocol[41] was then used to generate the coordinate, parameter, and simulation files necessary for a milestoning calculation. For each of the MD milestones, a copy of the apo β-cyclodextrin structure was generated and the guest molecule was then placed at the appropriate radius from the bound state coordinates. Any water molecules that clashed with the guest molecule were removed. The guest distribution for the BD milestone was constructed by first running a conventional BD simulation where trajectories terminated at the appropriate distance for the milestone surface (13.5 Å).

Simulation

MD Simulations A modified version of NAMD 2.12 was used for all MD simulations[42]. For all 13 milestones in the MD region, the standard SEEKR procedure for minimization, equilibration and simulation was followed. First, 5000 steps of minimization were performed to allow for relaxation, particularly of solvent, around the newly placed guest molecule. Further relaxation of the solvent was achieved by a series of 2 ps heating simulations that gradually increased the temperature from 298 K to 350 K and then cooled back to 298 K. Host and guest atoms were constrained during these heating simulations to ensure that the guest remained on the appropriate milestone surface. To obtain the equilibrium distribution of the guest molecule on each milestone, 200 ns of constant volume simulation was performed. A $90\,\mathrm{kcal}\cdot\mathrm{mol}^{-1}\cdot\mathring{\mathrm{A}}^{-2}$ harmonic restraint was used to hold the COM of the guest molecule at the appropriate distance from the binding site. To minimize any bias of the arbitrary guest starting conformation, the first 40 ns of each simulation were discarded and therefore were not included as part of the equilibrium distribution. From these trajectories, position and velocity configurations were selected every 0.2 ns, resulting in a total of 800 configurations per milestone. To obtain the first hitting point distribution (FHPD) of each milestone, 10 independent and unrestrained simulations were initiated from these equilibrium configurations. Each simulation was propagated backwards in time by reversing its velocity at constant energy and volume (a total of 8000 reversals for each milestone). Only trajectories that struck an adjacent milestone before recrossing the milestone on which they originated were included as part of the FHPD for that milestone. To obtain the transition probabilities and times necessary for the calculation of kinetic parameters, all members of the FHPD were brought back to their starting position and velocity and new unrestrained simulations were then initiated from each configuration. These simulations were propagated forward in time at constant energy and volume. Once a simulation crossed its starting milestone again, it was monitored for crossing of adjacent milestones. When an adjacent milestone was crossed, the simulation was stopped and the transition and incubation time were recorded. Although more of the equilibrium simulation trajectories could likely have been used without biasing the results based on the starting conformation, the 8000 reversal trajectories that resulted from the 160 ns used were more than sufficient to sample the transitions between the milestones, resulting in hundreds of observed transitions between each milestone. In total, 2.6 μs of equilibrium sampling (160 ns for 16 milestones) were used and approximately 570 ns of FHPD sampling for a total of 3.2 μs of simulation used in the milestoning model. The total cost per ligand (including simulation discarded for equilibration) was therefore $\sim$3.8 μs.

BD Simulations All BD simulations were performed using the BrownDye software package[12]. Electrostatic potentials of the host and guest molecules used as inputs for the BD simulation were calculated with APBS version 1.4[63]. To match experimental conditions, APBS calculations and BD simulations were carried out with a solvent dielectric of 78, a solute dielectric of 2, and zero ionic concentration. An initial series of simulations were initiated from the b-surface, a sphere that encloses the entire host molecule and has sufficient radius for the guest molecule to be situated in bulk solvent such that forces between the host and guest are centrosymmetric. 10^6 independent BD simulations were initiated from random points on the b-surface and and were propagated until the guest either contacted the outermost milestone (13.5 Å) or escaped. Trajectories that successfully contacted the 13.5 Å milestone were used as the FHPD for this milestone. Another series of 10^6 BD simulations were initiated from the FHPD and propagated until contacting the second-outermost milestone (12.0 Å) or escaping to the q-surface. This procedure is automated by SEEKR.

Milestoning Calculations

Statistics from all milestones in the MD and BD regions were extracted using SEEKR and combined to construct a transition kernel as well as an incubation time vector. These are the two key quantities for the calculation of kinetic parameters in milestoning theory.[43] As described previously, a post-simulation analysis was performed to account for ligand transitions between the two faces of the β-cyclodextrin ring for milestones of radius 7.5 Å or less. The analysis portion of SEEKR was then used to compute the desired kinetic quantities, k_{on} and k_{off}, as well as the free energy of binding ΔG_{bind}.

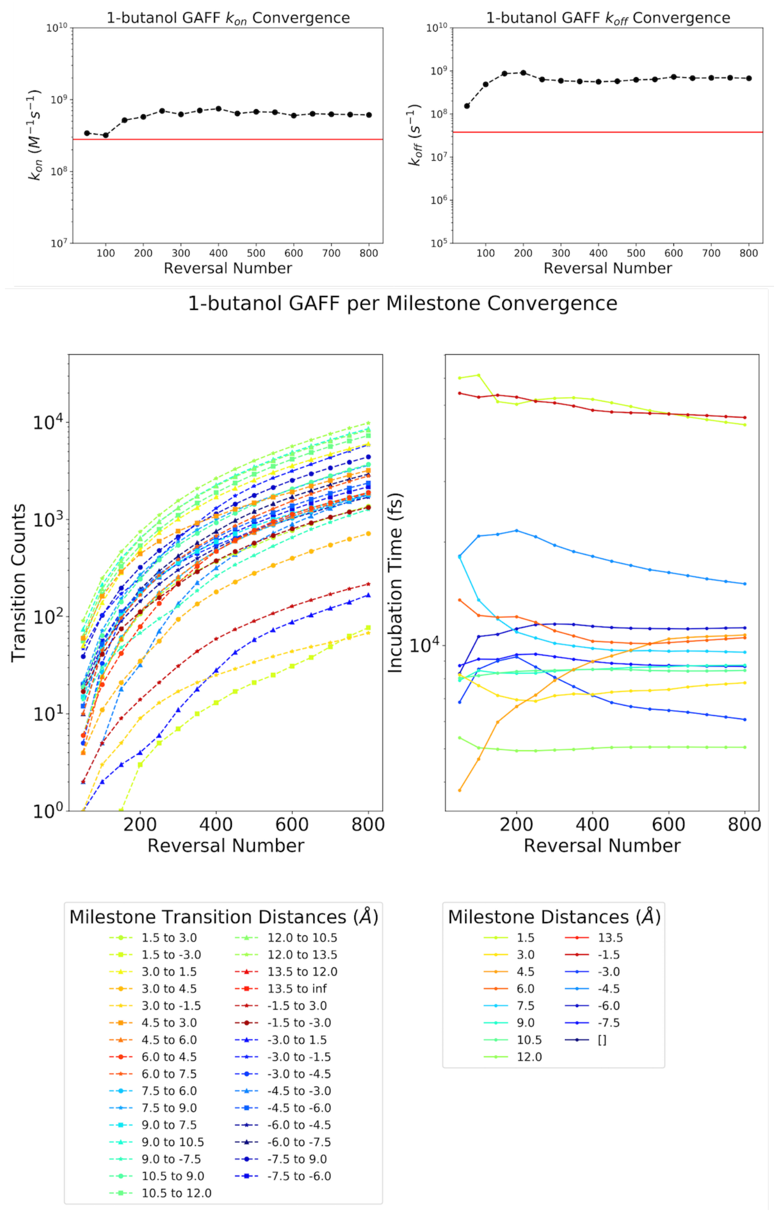


Figure 5.6: 1-butanol GAFF per milestone convergence plot

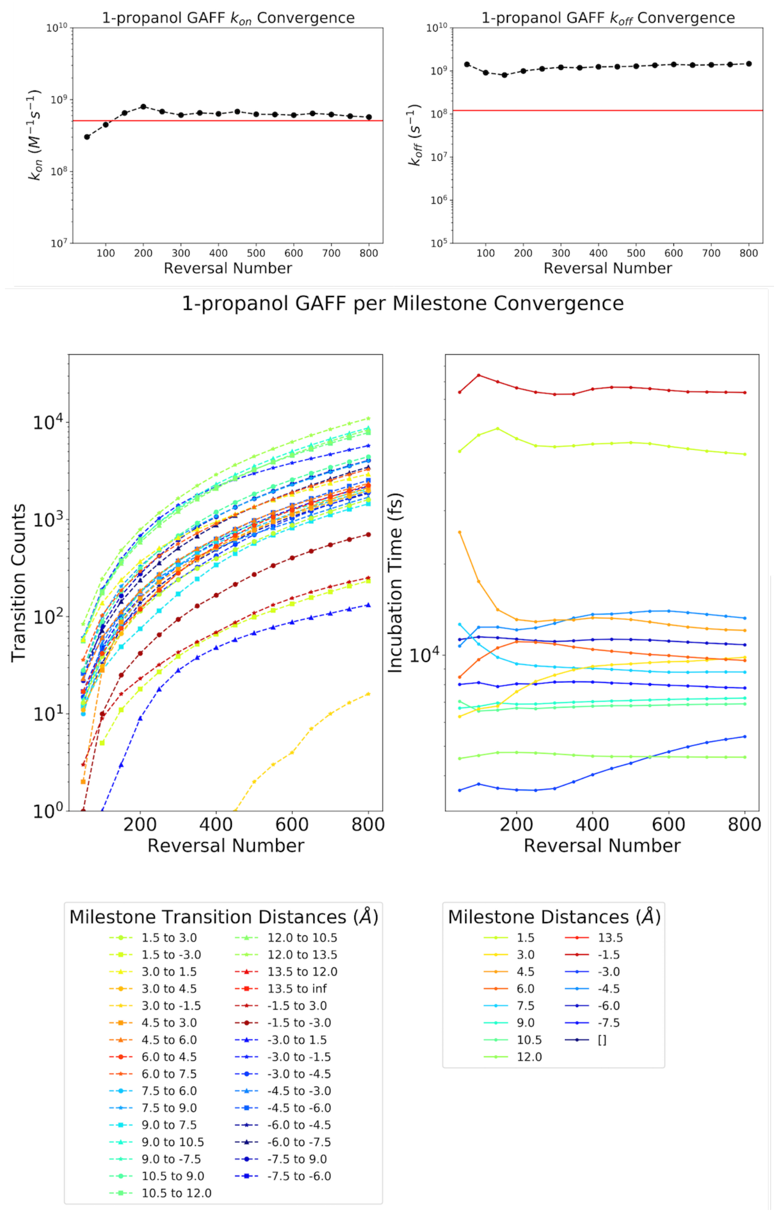


Figure 5.7: 1-propanol GAFF per milestone convergence plot

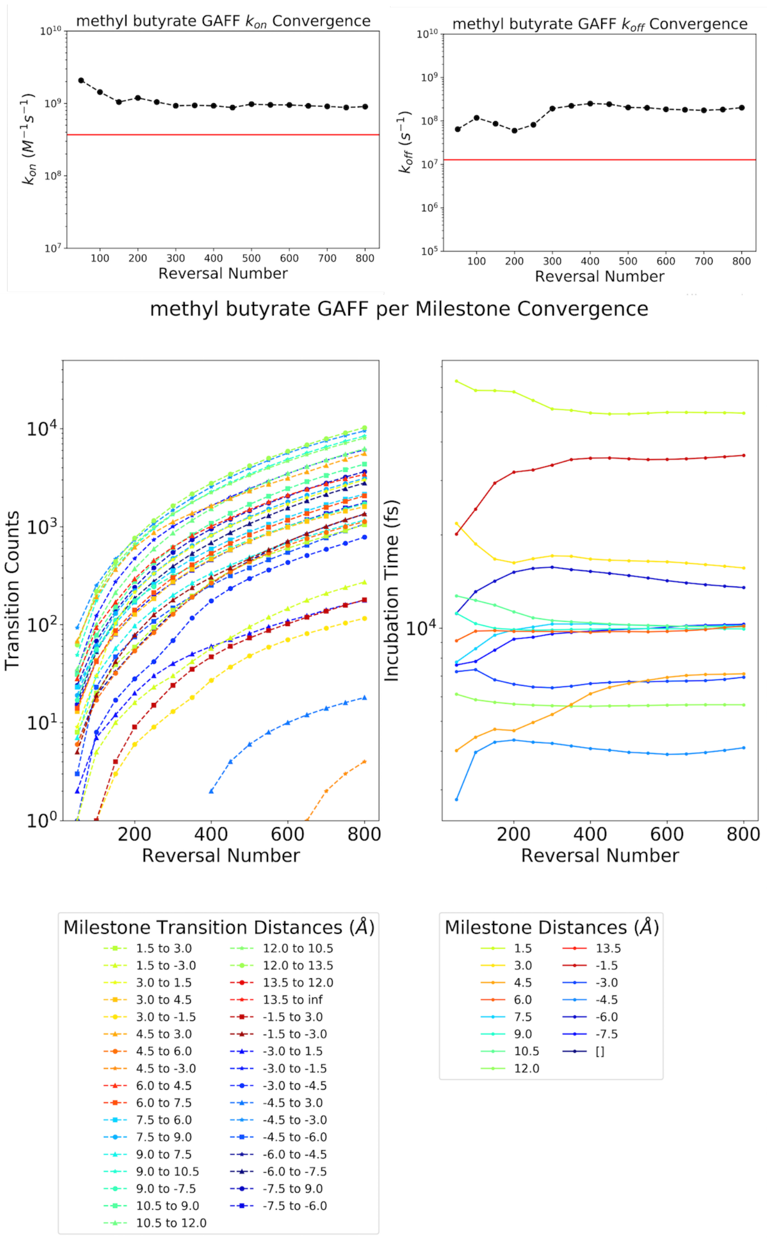


Figure 5.8: methyl butyrate GAFF per milestone convergence plot

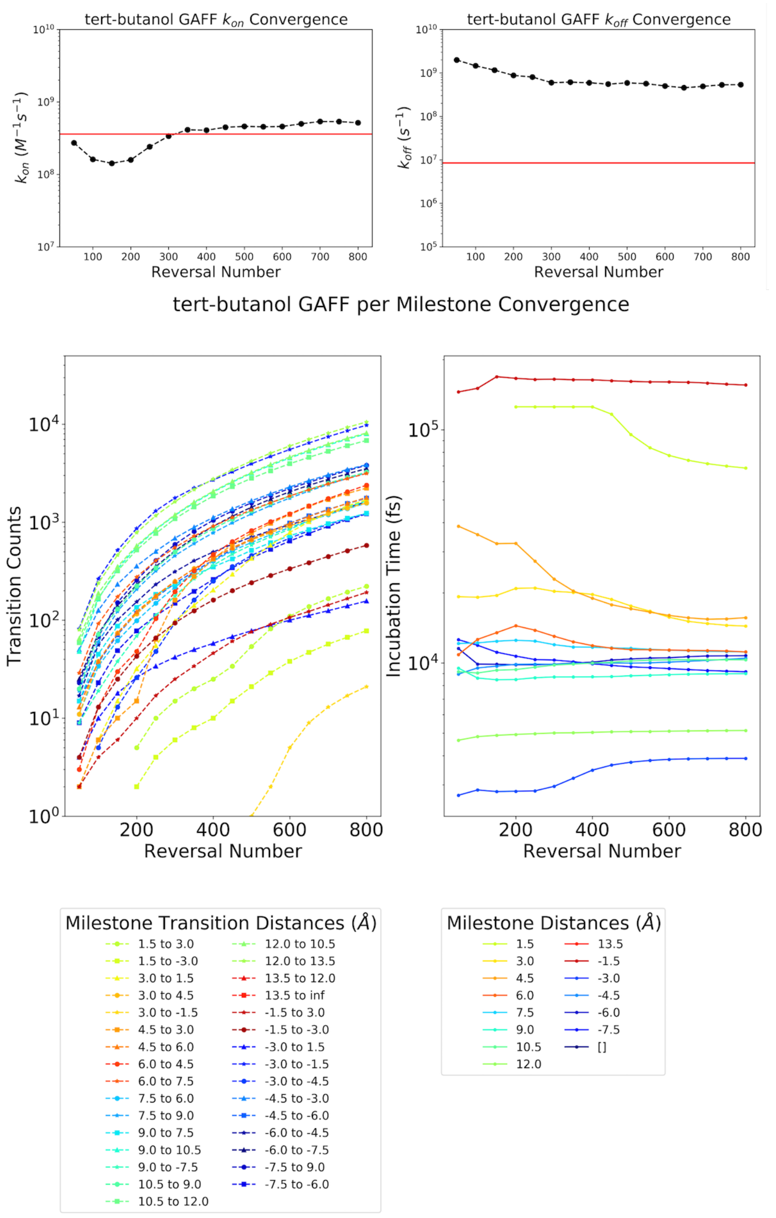


Figure 5.9: tert butanol GAFF per milestone convergence plot

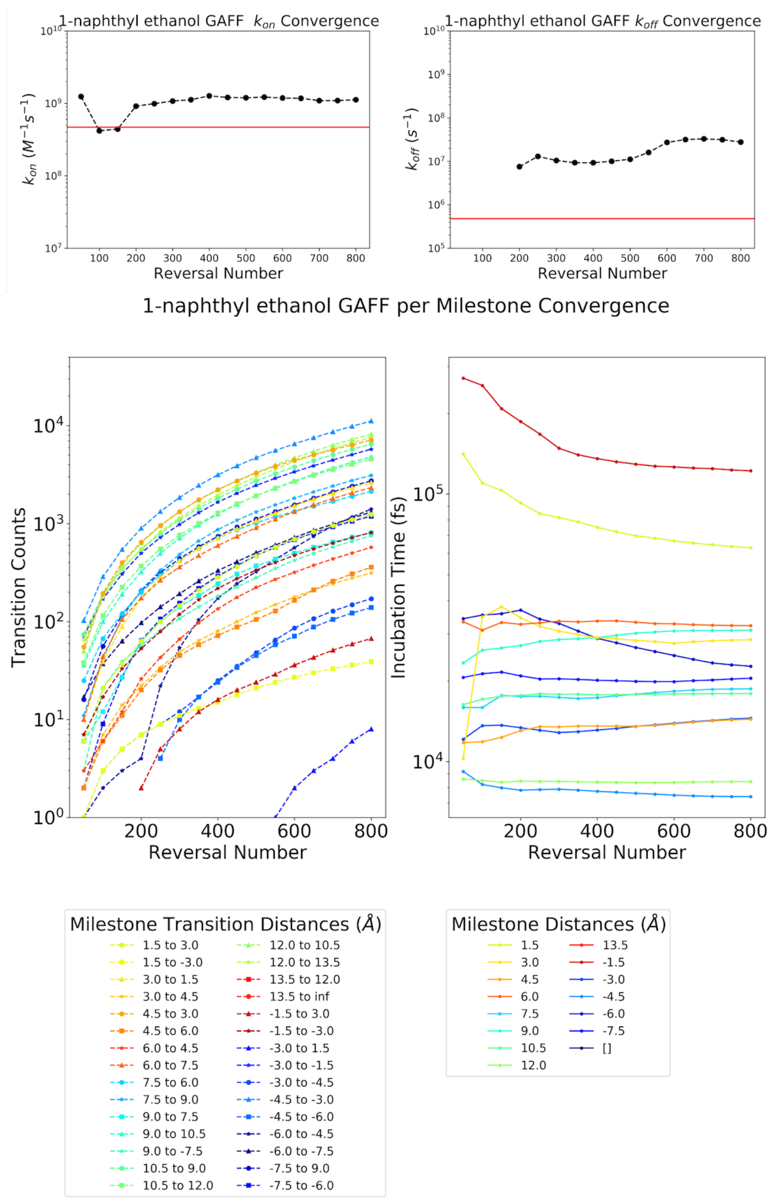


Figure 5.10: 1-naphthyl ethanol GAFF per milestone convergence plot

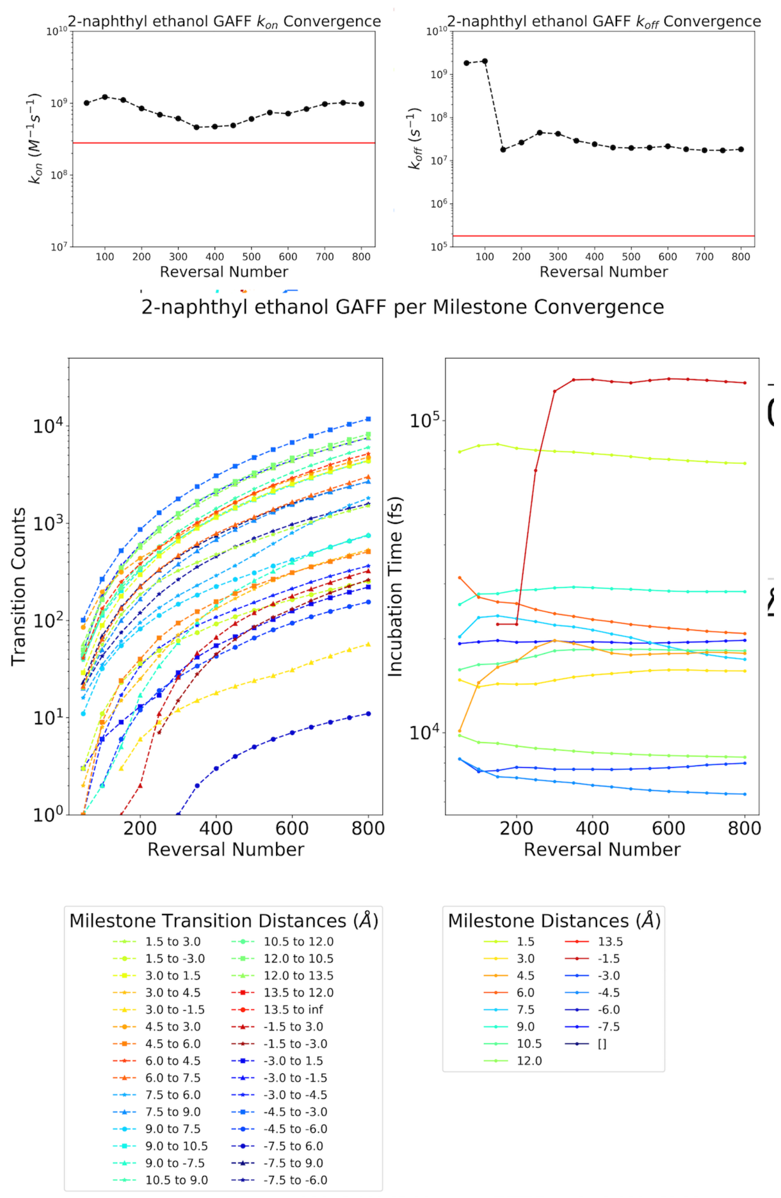


Figure 5.11: 2-naphthyl ethanol GAFF per milestone convergence plot

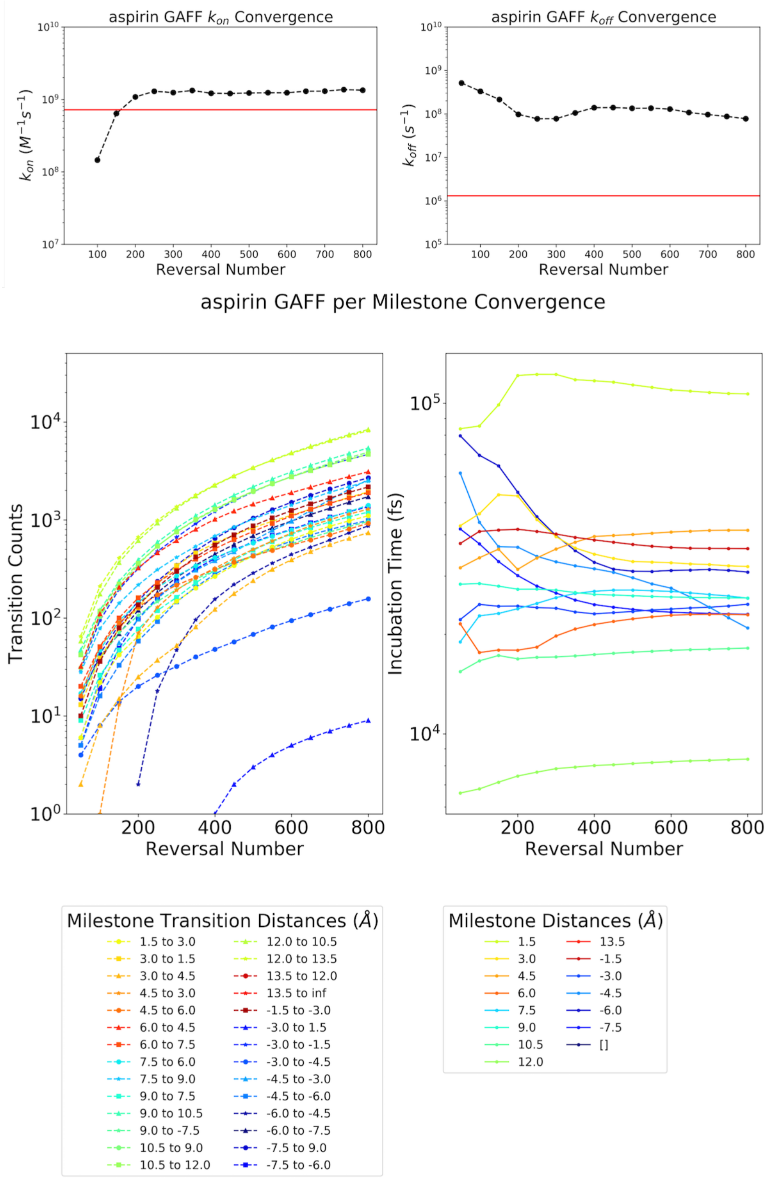


Figure 5.12: aspirin GAFF per milestone convergence plot

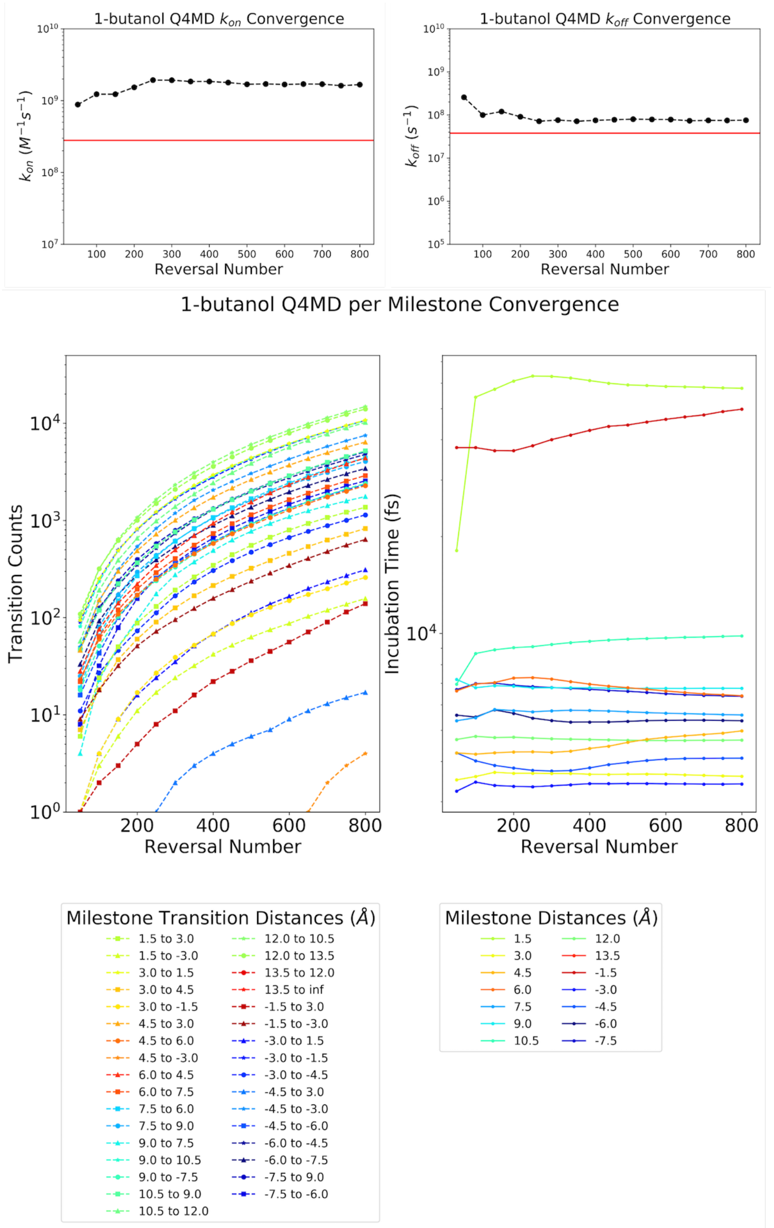


Figure 5.13: 1-butanol Q4MD per milestone convergence plot

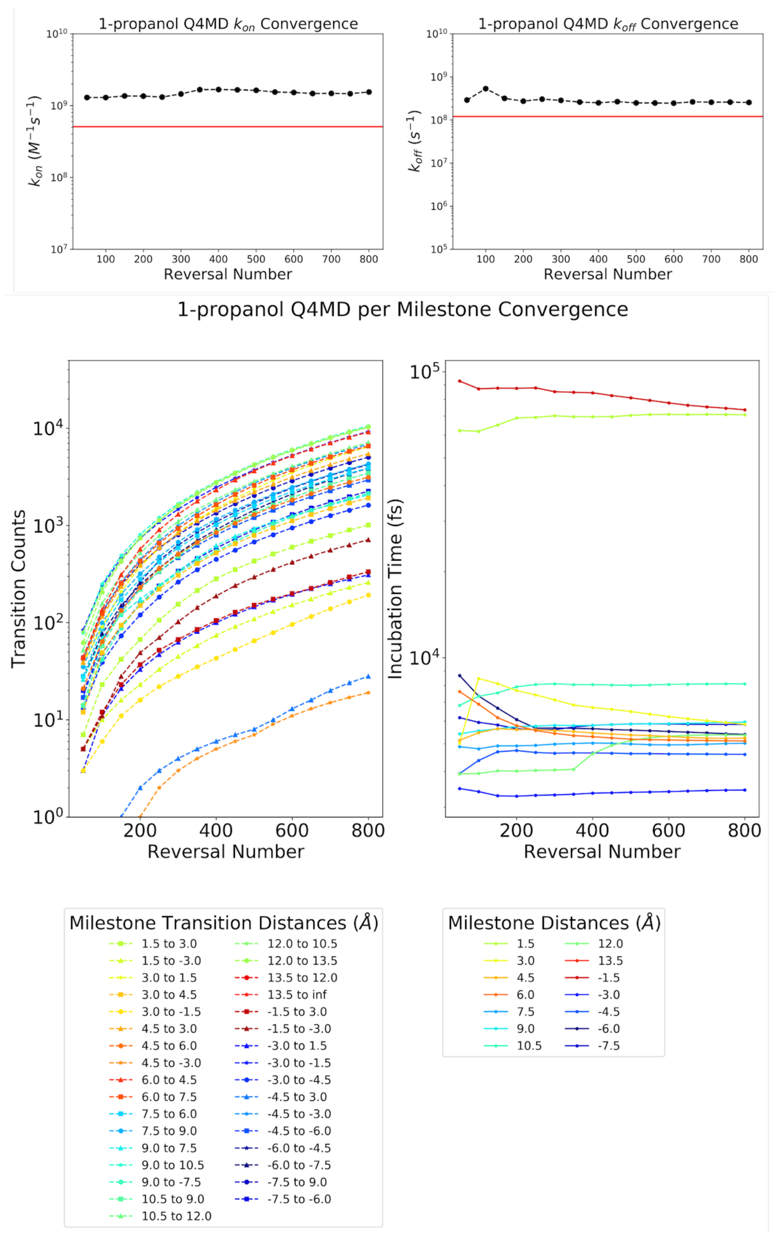


Figure 5.14: 1-propanol Q4MD per milestone convergence plot

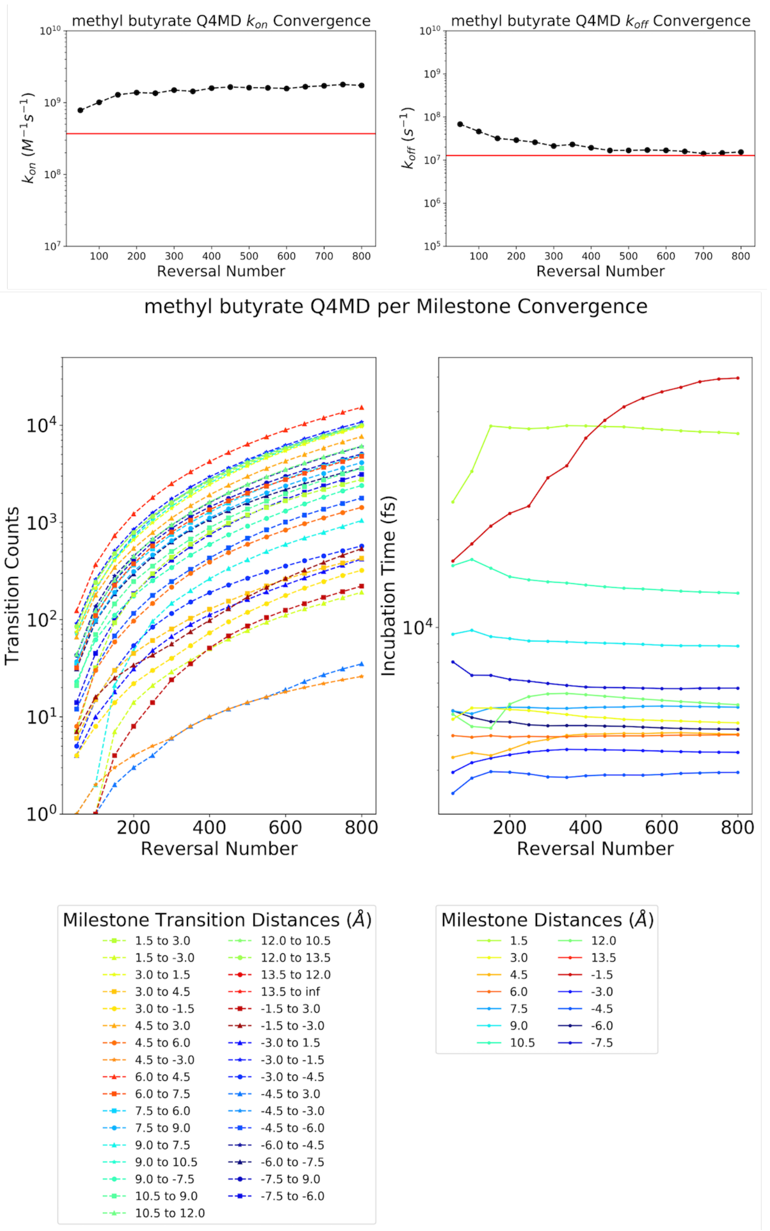


Figure 5.15: methyl butyrate Q4MD per milestone convergence plot

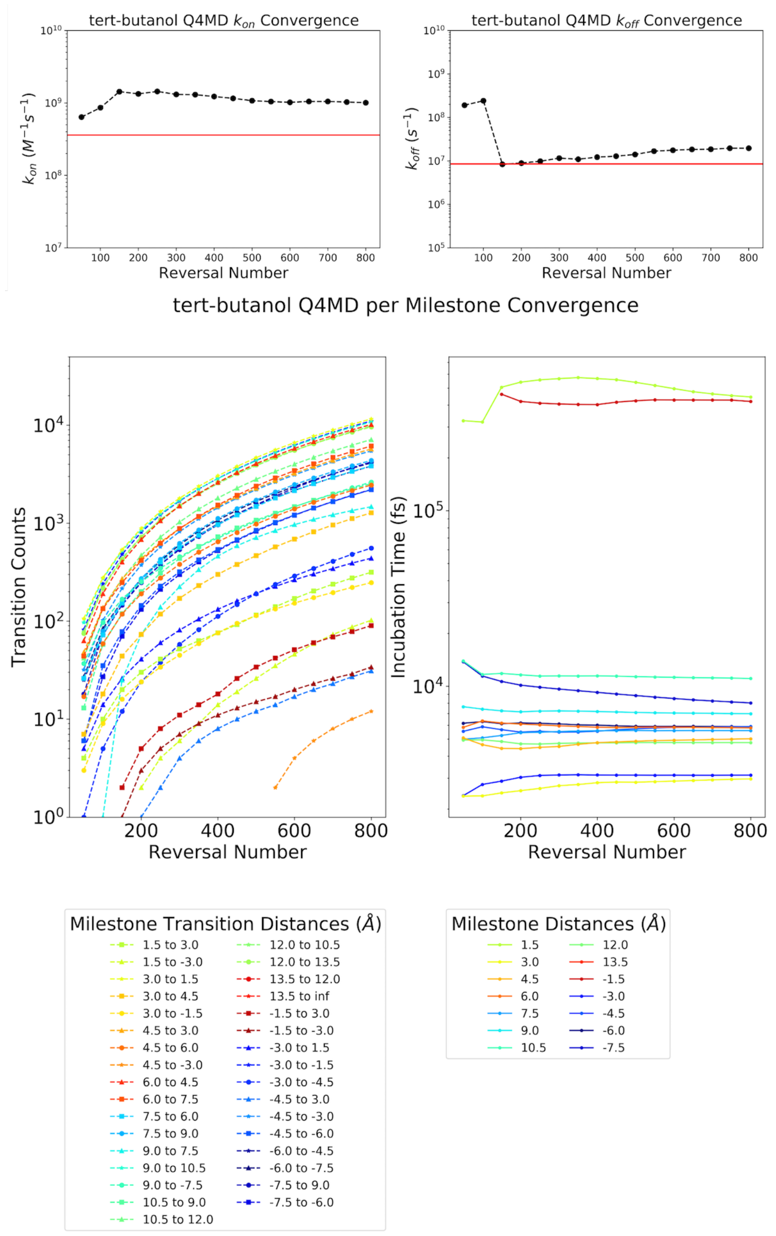


Figure 5.16: tert butanol Q4MD per milestone convergence plot

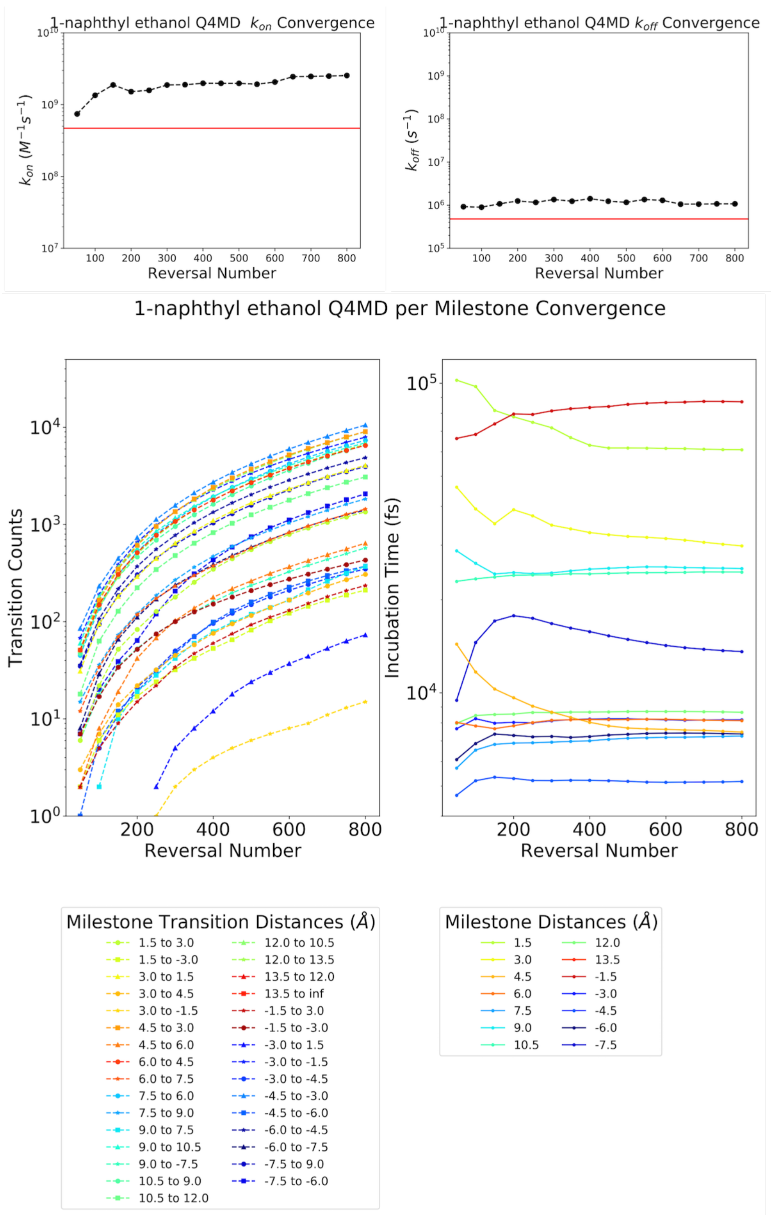


Figure 5.17: 1-naphthyl ethanol Q4MD per milestone convergence plot

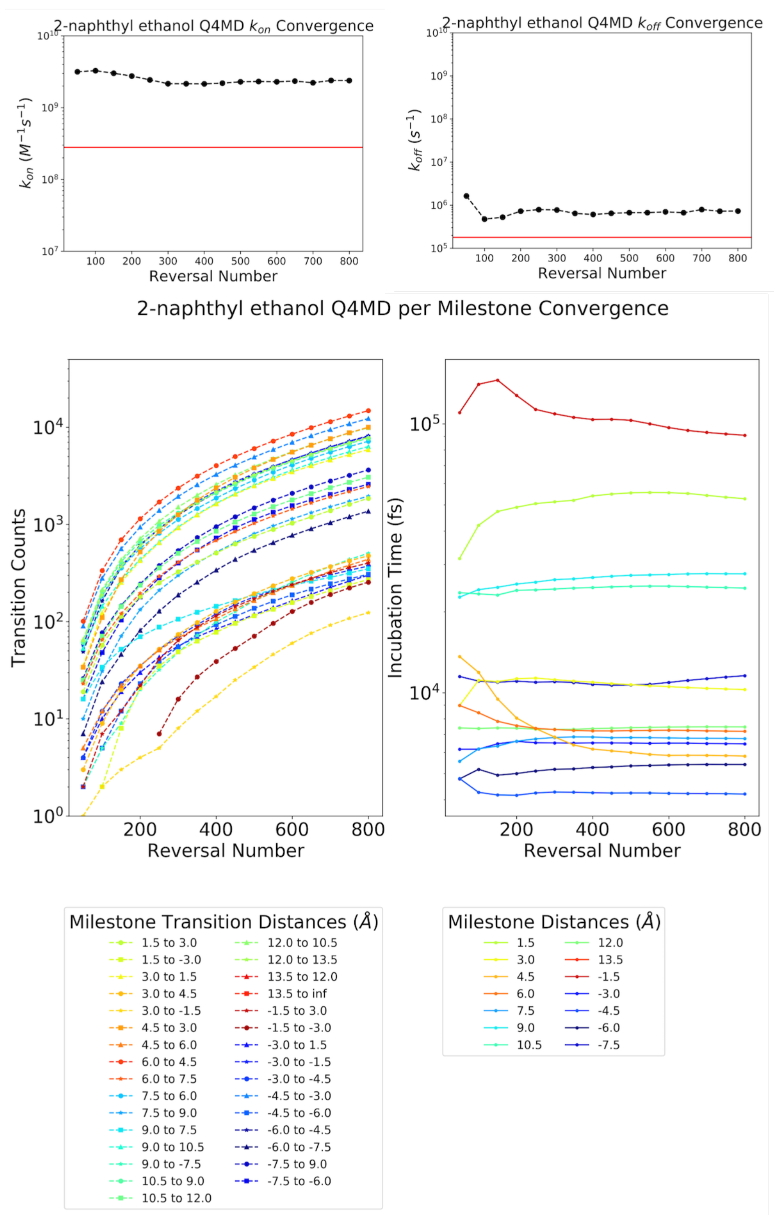


Figure 5.18: 2-naphthyl ethanol Q4MD per milestone convergence plot

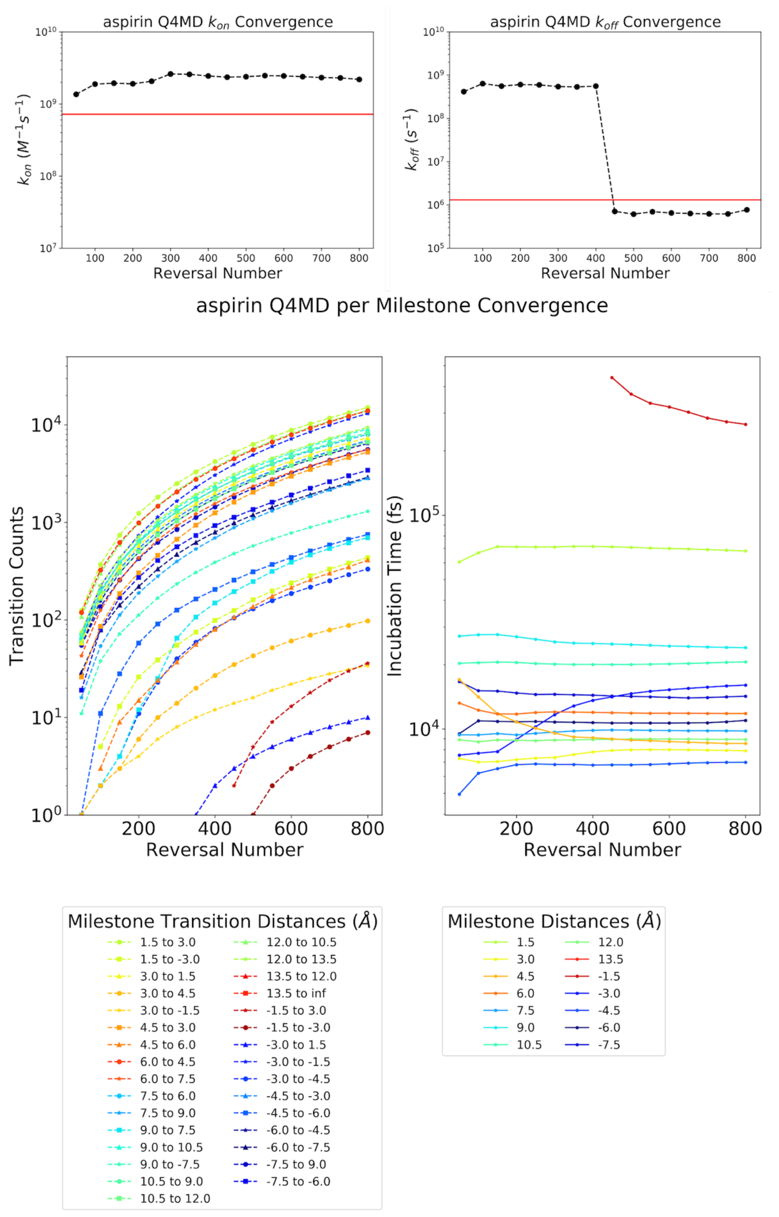


Figure 5.19: aspirin Q4MD per milestone convergence plot

References

(1) Schuetz, D. A.; de Witte, W. E. A.; Wong, Y. C.; Knasmueller, B.; Richter, L.; Kokh, D. B.; Sadiq, S. K.; Bosma, R.; Nederpelt, I.; Heitman, L. H.; Segala, E.; Amaral, M.; Guo, D.; Andres, D.; Georgi, V.; Stoddart, L. A.; Hill, S.; Cooke, R. M.; De Graaf, C.; Leurs, R.; Frech, M.; Wade, R. C.; de Lange, E. C. M.; IJzerman, A. P.; Müller-Fahrnow, A.; Ecker, G. F. *Drug Discov. Today* **2017**, *22*, 896–911.

(2) Lu, H.; Tonge, P. J. *Curr. Opin. Chem. Biol.* **2010**, *14*, 467–474.

(3) Copeland, R. A.; Pompliano, D. L.; Meek, T. D. *Nat. Rev. Drug Discov.* **2006**, *5*, 730–739.

(4) Copeland, R. A. *Nat. Rev. Drug Discov.* **2016**, *15*, 87–95.

(5) Swinney, D. C. *Nat. Rev. Drug Discov.* **2004**, *3*, 801–808.

(6) De Vivo, M.; Masetti, M.; Bottegoni, G.; Cavalli, A. *J. Med. Chem.* **2016**, *59*, 4035–4061.

(7) Amaro, R. E.; Mulholland, A. J. *Nat Rev Chem* **2018**, *2*, 0148.

(8) Bruce, N. J.; Ganotra, G. K.; Kokh, D. B.; Sadiq, S. K.; Wade, R. C. *Curr. Opin. Struct. Biol.* **2018**, *49*, 1–10.

(9) Northrup, S. H.; Allison, S. A.; McCammon, J. A. *J. Chem. Phys.* **1984**, *80*, 1517.

(10) McCammon, J. A.; Northrup, S. H.; Allison, S. A. *J. Phys. Chem.* **1986**, *90*, 3901–3905.

(11) Zhou, H. *J. Chem. Phys.* **1990**, *92*, 3092–3095.

(12) Huber, G. A.; McCammon, J. A. *Comput. Phys. Commun.* **2010**, *181*, 1896–1905.

(13) Shaw, D. E.; Bowers, K. J.; Chow, E.; Eastwood, M. P.; Ierardi, D. J.; Klepeis, J. L.; Kuskin, J. S.; Larson, R. H.; Lindorff-Larsen, K.; Maragakis, P.; Moraes, M. A.; Dror, R. O.; Piana, S.; Shan, Y.; Towles, B.; Salmon, J. K.; Grossman, J. P.; Mackenzie, K. M.; Bank, J. A.; Young, C.; Deneroff, M. M.; Batson, B. In *Proc. Conf. High Perform. Comput. Networking, Storage Anal. - SC '09*, ACM Press: New York, New York, USA, 2009, p 1.

(14) Shaw, D. E.; Grossman, J. P.; Bank, J. A.; Batson, B.; Butts, J. A.; Chao, J. C.; Deneroff, M. M.; Dror, R. O.; Even, A.; Fenton, C. H.; Forte, A.; Gagliardo, J.; Gill, G.; Greskamp, B.; Ho, C. R.; Ierardi, D. J.; Iserovich, L.; Kuskin, J. S.; Larson, R. H.; Layman, T.; Lee, L. S.; Lerer, A. K.; Li, C.; Killebrew, D.; Mackenzie, K. M.; Mok, S. Y. H.; Moraes, M. A.; Mueller, R.; Nociolo, L. J.; Peticolas, J. L.; Quan, T.; Ramot, D.; Salmon, J. K.; Scarpazza, D. P.; Ben Schafer, U.; Siddique, N.; Snyder, C. W.; Spengler, J.; Tang, P. T. P.; Theobald, M.; Toma, H.; Towles, B.; Vitale, B.; Wang, S. C.; Young, C. In *Int. Conf. High Perform. Comput. Networking, Storage Anal. SC*, IEEE: 2014; Vol. 2015-Janua, pp 41–53.

(15) Shan, Y.; Kim, E. T.; Eastwood, M. P.; Dror, R. O.; Seeliger, M. A.; Shaw, D. E. *J. Am. Chem. Soc.* **2011**, *133*, 9181–9183.

(16) Shan, Y.; Eastwood, M. P.; Zhang, X.; Kim, E. T.; Arkhipov, A.; Dror, R. O.; Jumper, J.; Kuriyan, J.; Shaw, D. E. *Cell* **2012**, *149*, 860–870.

(17) Dror, R. O.; Pan, A. C.; Arlow, D. H.; Borhani, D. W.; Maragakis, P.; Shan, Y.; Xu, H.; Shaw, D. E. *Proc. Natl. Acad. Sci.* **2011**, *108*, 13118–13123.

(18) Pan, A. C.; Borhani, D. W.; Dror, R. O.; Shaw, D. E. *Drug Discov. Today* **2013**, *18*, 667–673.

(19) Tang, Z.; Chang, C.-e. A. *J. Chem. Theory Comput.* **2018**, *14*, 303–318.

(20) Buch, I.; Giorgino, T.; De Fabritiis, G. *Proc. Natl. Acad. Sci.* **2011**, *108*, 10184–10189.

(21) Plattner, N.; Noé, F. *Nat. Commun.* **2015**, *6*, 7653.

(22) Wu, H.; Paul, F.; Wehmeyer, C.; Noé, F. *Proc. Natl. Acad. Sci.* **2016**, *113*, E3221–E3230.

(23) Doerr, S.; de Fabritiis, G. *J. Chem. Theory ...* **2014**, *10*, 2064–2069.

(24) Mollica, L.; Theret, I.; Antoine, M.; Perron-Sierra, F.; Charton, Y.; Fourquez, J.-M.; Wierzbicki, M.; Boutin, J. A.; Ferry, G.; Decherchi, S.; Bottegoni, G.; Ducrot, P.; Cavalli, A. *J. Med. Chem.* **2016**, *59*, 7167–7176.

(25) Tiwary, P.; Limongelli, V.; Salvalaglio, M.; Parrinello, M. *Proc. Natl. Acad. Sci.* **2015**, *112*, E386–E391.

(26) Casasnovas, R.; Limongelli, V.; Tiwary, P.; Carloni, P.; Parrinello, M. *J. Am. Chem. Soc.* **2017**, *139*, 4780–4788.

(27) Elber, R. *Q. Rev. Biophys.* **2017**, *50*, e8.

(28) Ma, P.; Cardenas, A. E.; Chaudhari, M. I.; Elber, R.; Rempe, S. B. *J. Am. Chem. Soc.* **2017**, jacs.7b07419.

(29) Kirmizialtin, S.; Nguyen, V.; Johnson, K. A.; Elber, R. *Structure* **2012**, *20*, 618–627.

(30) Yu, T.-Q.; Lapelosa, M.; Vanden-Eijnden, E.; Abrams, C. F. *J. Am. Chem. Soc.* **2015**, *137*, 3041–3050.

(31) Bucci, A.; Yu, T.-Q.; Vanden-Eijnden, E.; Abrams, C. F. *J. Chem. Theory Comput.* **2016**, *12*, 2964–2972.

(32) Ma, W.; Schulten, K. *J. Am. Chem. Soc.* **2015**, *137*, 3031–3040.

(33) Teo, I.; Mayne, C. G.; Schulten, K.; Lelièvre, T. *J. Chem. Theory Comput.* **2016**, acs.jctc.6b00277.

(34) Dickson, A.; Lotz, S. D. *J. Phys. Chem. B* **2016**, *120*, 5377–5385.

(35) Dickson, A.; Lotz, S. D. *Biophys. J.* **2017**, *112*, 620–629.

(36) Lotz, S. D.; Dickson, A. *J. Am. Chem. Soc.* **2018**, jacs.7b08572.

(37) Chiu, S. H.; Xie, L. *J. Chem. Inf. Model.* **2016**, *56*, 1164–1174.

(38) Wong, C. F. *J. Comput. Chem.* **2018**, DOI: 10.1002/jcc.25201.

(39) Tran, D. P.; Takemura, K.; Kuwata, K.; Kitao, A. *J. Chem. Theory Comput.* **2018**, *14*, 404–417.

(40) Votapka, L. W.; Amaro, R. E. *PLoS Comput. Biol.* **2015**, *11*, e1004381.

(41) Votapka, L. W.; Jagger, B. R.; Heyneman, A. L.; Amaro, R. E. *J. Phys. Chem. B* **2017**, *121*, 3597–3606.

(42) C. Phillips, J.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; D. Skeel, R.; Kale, L.; Schulten, K. *J. Comput. Chem.* **2005**, *26*, 1781–1802.

(43) Faradjian, A. K.; Elber, R. *J. Chem. Phys.* **2004**, *120*, 10880–9.

(44) Shalloway, D.; Faradjian, A. K. *J. Chem. Phys.* **2006**, *124*, 054112.

(45) West, A. M. A.; Elber, R.; Shalloway, D. *J. Chem. Phys.* **2007**, *126*, 145104.

(46) Vanden-Eijnden, E.; Venturoli, M.; Ciccotti, G.; Elber, R. *J. Chem. Phys.* **2008**, *129*, 174102.

(47) Vanden-Eijnden, E.; Venturoli, M. *J. Chem. Phys.* **2009**, *130*, 194101.

(48) Májek, P.; Elber, R. *J. Chem. Theory Comput.* **2010**, *6*, 1805–1817.

(49) Kirmizialtin, S.; Elber, R. *J. Phys. Chem. A* **2011**, *115*, 6137–6148.

(50) Cardenas, A. E.; Elber, R. *Mol. Phys.* **2013**, *111*, 3565–3578.

(51) Cardenas, A. E.; Shrestha, R.; Webb, L. J.; Elber, R. *J. Phys. Chem. B* **2015**, *119*, 6412–6420.

(52) Bello-Rivas, J. M.; Elber, R. *J. Chem. Phys.* **2015**, *142*, 094102.

(53) Wang, J.; Wolf, R. M.; Caldwell, J. W.; Kollman, P. A.; Case, D. A. *J. Comput. Chem.* **2004**, *25*, 1157–1174.

(54) Wang, J.; Wang, W.; Kollman, P. A.; Case, D. A. *J. Mol. Graph. Model.* **2006**, *25*, 247–260.

(55) Cézard, C.; Trivelli, X.; Aubry, F.; Djedaïni-Pilard, F.; Dupradeau, F.-Y. *Phys. Chem. Chem. Phys.* **2011**, *13*, 15103.

(56) Fukahori, T.; Nishikawa, S.; Yamaguchi, K. *Bull. Chem. Soc. Jpn.* **2004**, *77*, 2193–2198.

(57) Fukahori, T.; Kondo, M.; Nishikawa, S. *J. Phys. Chem. B* **2006**, *110*, 4487–4491.

(58) Nishikawa, S.; Fukahori, T.; Ishikawa, K. *J. Phys. Chem. A* **2002**, *106*, 3029–3033.

(59) Nishikawa, S.; Kondo, M. *J. Phys. Chem. B* **2006**, *110*, 26143–26147.

(60) Rekharsky, M. V.; Inoue, Y. *Chem. Rev.* **1998**, *98*, 1875–1918.

(61) Barros, T. C.; Stefaniak, K.; Holzwarth, J. F.; Bohne, C. *J. Phys. Chem. A* **1998**, *102*, 5639–5651.

(62) Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. *J. Chem. Phys.* **1983**, *79*, 926–935.

(63) Baker, N. A.; Sept, D.; Joseph, S.; Holst, M. J.; McCammon, J. A. *Proc. Natl. Acad. Sci.* **2001**, *98*, 10037–10041.

Chapter 6

Emerging Computational Methods for the Rational Discovery of Allosteric Drugs

6.1 Abstract

Allosteric drug development holds promise for delivering medicines that are more selective and less toxic than those that target orthosteric sites. To date, the discovery of allosteric binding sites and lead compounds has been mostly serendipitous, through high-throughput screening. Over the last decade, structural data has become readily available for larger protein systems and more membrane protein classes (e.g., GPCRs and ion channels), which are common allosteric drug targets. In parallel, improved simulation methods now provide better atomistic understanding of the protein dynamics and cooperative motions that are critical to allosteric mechanisms. As a result of these advances, the field of predictive allosteric drug development is now on the cusp of a new era of rational structure-based computational methods. Here, we review algorithms that predict allosteric sites based on sequence data and molecular dynamics simulations, describe tools that assess the druggability of these pockets, and discuss how Markov state models and topology analyses provide insight into the relationship between protein dynamics and allosteric drug binding. In each section, we first provide an overview of the various method classes before describing relevant algorithms and software packages.

6.2 Review Motivation and Organization

To date, most allosteric drugs have been discovered through high-throughput screening. But growing databases of biomolecular structure and sequence data, in conjunction with increases in computing power and improvements in predictive algorithms, are enabling the rational de novo design of allosteric drugs. Given the large number of published algorithms for predicting allosteric mechanisms, it can be difficult to select the most appropriate method for a given target. This review serves as an introduction for those who wish to use computational techniques to develop allosteric drugs.

After a broad overview of allosteric drug discovery, this review is divided into three sections. First, we discuss bioinformatics and molecular-dynamics methods to identify allosterically important sequence positions. Second, we summarize the computational methods to predict druggable pockets at these functionally relevant sites. Finally, we describe how Markov state models and topological analyses can tie these single sequence sites to global protein function and dynamics.

6.3 Introduction

Allosteric drugs offer a number of advantages that make them desirable as drug candidates. Allosteric effectors, by definition, alter protein activity by binding to a site distinct from the orthosteric pocket. Because allosteric sites are typically less evolutionarily conserved, allosteric drugs can be highly selective, even among other members of the same protein family.[1–8] In some cases, allosteric sites are so unique among proteins that an effector is said to have “absolute subtype specificity."[2, 3, 9, 10]

Allosteric modulators may have spatiotemporal specificity. For example, they can be active only in the presence of the endogenous ligand, thus restricting their effect to certain tissues at certain times,[2–4, 9, 11] which may slow desensitization.[10, 12]

Allosteric effectors are generally saturable, meaning that they have a maximal effect that does not necessarily correspond to complete inhibition or activation.[2, 4–6, 8, 10, 11] This saturability enables safer dosing. For example, if the maximal effect is an 80% reduction in signaling, overdosing will not fully eliminate an essential signal.[1, 2, 4]

Other advantages can include noncompetitive inhibition (i.e., drug activity cannot be “overwhelmed" by high concentrations of the endogenous ligand) and pathway- or substrate-specific modulation, which reduces unwanted activity by specifically targeting a single protein function.[2, 4, 10] For example, if a protein is involved in multiple pathways, an allosteric effector may impact the activity of each pathway differently depending on the systems-biology context. If a protein acts on multiple substrates, the impact on activity may depend on the biological context.

Despite many potential advantages of allosteric therapeutics, it has been challenging to identify predictive approaches to discovering allosteric drugs. In recent decades, the pharmaceutical industry has favored more traditional targets for three primary reasons: the relative ease of assay development around orthosteric sites; access to high-throughput, high-resolution X-ray crystallography; and advances in ligand- and receptor-based computational methods to optimize ligand-binding affinity at a substrate-competitive site. This structure-based approach is thought to significantly reduce the time and cost of hit-to-lead and lead-to-drug development by reducing the number of compounds that need be synthesized.[13, 14] Work by Doman et al. comparing computer-aided drug discovery (CADD) and high-throughput screening (HTS) reported that the two methods had hit rates of 35% and 0.021%, respectively.[15]

In contrast, allosteric drugs are uniquely challenging from a rational drug-design perspective. Because experimental assays typically measure orthosteric function rather than ligand binding at the allosteric site, efficient development of allosteric drugs requires that the complex structure-activity relationships (SARs) governing both binding affinity and allosteric activity be considered simultaneously.[5, 10, 16] Further, allosteric sites are less likely to be evolutionarily conserved. While this enables increased subtype specificity, it also increases the chances of evolved resistance[5, 9, 10] and can complicate testing in evolutionarily distant animal models.[10]

Additionally, allosteric effectors are particularly susceptible to “mode switching," where relatively minor chemical changes can drastically affect ligand efficacy.[10, 16] Structurally similar drug metabolites, therefore, may have varying and unpredictable distributions and allosteric effects.[10, 16] Optimizing allosteric modes of action requires methods that are very different than those used in orthosteric drug discovery.[5]

Multifunctional allosteric proteins are particularly challenging. While drug designers may desire to target a single protein function, an allosteric effector may also alter other functions, hindering a full mechanistic understanding of the pharmacology.[10] Also, the benefits of spatiotemporal specificity are lost if the distribution of the endogenous ligand changes with progression of the disease state.[10] Finally, assessment of the limited number of known allosteric pockets indicates that they are generally shallow[5] and present flat SAR.[10] These structural features similarly challenge existing rational drug-discovery paradigms and the general practice of developing selective compounds by optimizing affinity.

Despite these challenges, allosteric drug discovery has gained momentum recently due to a number of developments.[10] First, several allosteric drugs across a broad range of pharmacological target classes have been rationally designed,[17–23] encouraging pursuit of others, as evidenced by the number of allosteric drugs currently in clinical trials.[24] The recent elucidation of new membrane protein crystal structures for GPCRs[25, 26] and ion channels[27, 28] have assisted in the structure-based design approach to these successes. Finally, advances in our understanding of allosteric mechanisms have supported development of additional rational design strategies (see below).

Our understanding of allosteric mechanisms has advanced considerably since the initial conception of Monod, Wyman, and Changeux.[29] Modern models of allostery consider conformational ensembles.[5, 8, 9, 30–43] This revised view supports newly established and emerging computational advances that comprehensively map conformational landscapes and predict communication between allosteric and orthosteric sites. For example, the physical mechanisms of allostery generally alter the entropic and enthalpic factors that define the conformational landscape and, therefore, govern protein function.[5, 8, 9] The observed correlation between allosteric modulation and protein structural dynamics is varied: Major conformational rearrangements occur in some cases, as compared with subtle shifts in conformational populations in others.[4–6, 9, 10, 35, 37, 44] An excellent metaphor for these phenomena are Kornev and Taylor's classification of “domino" versus “violin" models of allosteric signal transduction.[45] (Figure 1.) Further, allosteric signals are transmitted through a range of structural motifs, from rigid core regions to flexible linkers.[46] Allostery may occur through essential residues along a single allosteric path[47] or through many weak pathways connecting one site to another, acting in concert.[48] As such, it is not surprising that many of the allostery-prediction methods discussed in this review (some of which do not use structure/geometry information at all) in practice may identify non-contiguous groups of residues as being allosterically linked. Such predictions should not immediately be assumed to be wrong but rather may indicate a non-“domino" model of allostery.

Recent work has also revealed that protein allostery is not merely a transition between two discrete protein conformations, as initially thought, but rather a shift in the equilibrium populations of many conformations, induced by effector binding.[31, 35, 38, 41, 42, 51, 52] It is becoming increasingly clear that the kinetics of these transitions define the mechanisms of allostery.[38, 51] Empowered with these new understandings and advances in molecular simulation (in terms of speed of calculation and improved methodologies), the era of allosteric drug discovery is now on the cusp of radical advancement.

6.3.1 Emerging Rational Design Principles

So-called tried-and-true “design principles" are still being developed. However, a few general principles have begun to emerge. For example, many argue that it is insufficient to design a ligand that merely binds to an allosteric site; rather, the effector must make contact with certain key binding-pocket atoms to have the desired effect.[5, 10] These key atoms can often be identified through mutagenesis experiments and crystallographic studies of other allosteric ligands.

A promising set of design principles is encapsulated in the “allo-network" strategy, a rational approach that adopts two simultaneous but orthogonal approaches to ligand design.[6] On the protein-structure level, the primary focus is to target a single protein function or an interaction with a single partner. On the signaling-pathway level, the “allo-network" strategy suggests targeting less-connected upstream proteins instead of the more direct, though potentially highly connected, signaling proteins themselves. When applied to early-stage design, the allo-network method is predicted to increase the likelihood that a given allosteric effector will proceed through the drug-approval process.[6, 12, 55]

Several published examples for recently approved allosteric drugs serve to illustrate the current state-of-the art for emerging allosteric drug design principles. They also represent the significant advances that have been made to utilize structure-based methods for challenging druggable sites such as protein-protein interfaces and for membrane proteins such as ion channels. The available details of the discovery and optimization of these compounds do not include the methods discussed in this review, however they highlight where these predictive techniques could contribute to the allosteric design process.

In 2011, Gilmartin et al. of GlaxoSmithKline reported the discovery of a pharmacokinetically-

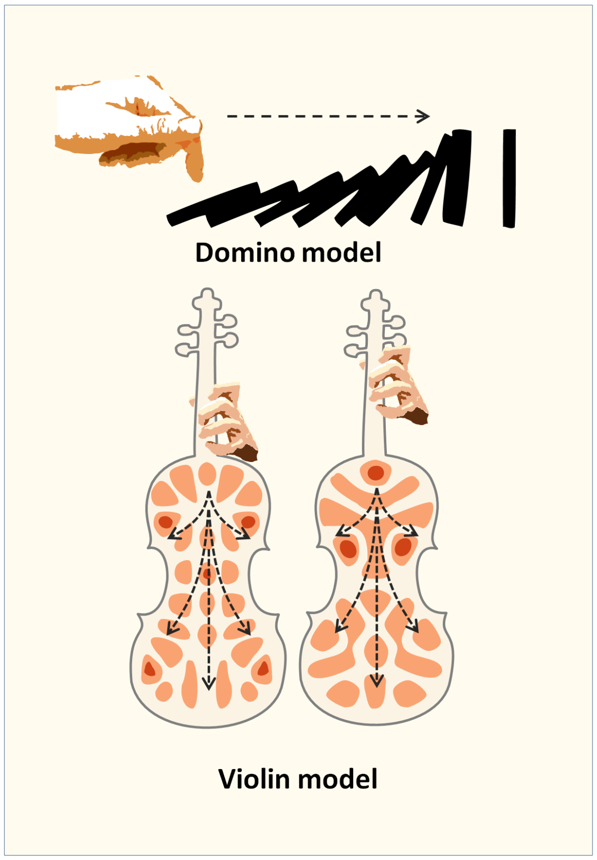


Figure 6.1: When a small molecule binds to the allosteric site of a protein, information is transferred through the protein molecule to its active site. Two different methods of transmission can be defined. The first mechanism, here defined as the “domino model", is a sequential set of events propagating linearly from the allosteric site to the active site. Binding of the effector triggers local structural changes that sequentially propagate via a single pathway to the active site. It was suggested that this mechanism is applicable for the PDZ domain family[49], G protein-coupled receptors, the chymotrypsin class of serine proteases, and hemoglobin[50]. The second mechanism, defined here conceptually as a “violin model", is based on vibration pattern changes inside the protein. In a violin its pitch can be changed by a slight movement of the violin player’s finger on the fingerboard. Information about the finger movement is, thus, transferred throughout the whole body of the violin with no specific pathway for the signal transduction. By analogy, protein allosteric site is a fingerboard of the protein and a small signaling molecule is the player’s finger. If a protein is in a particular vibration mode, it is possible to suggest that binding a small effector molecule to a specific site can change this mode. The signal, thus, will be spread throughout the whole protein including its active site. The “domino model" is a reliable way to transfer information in a macro world, but on a molecular level, with significant thermal motions of the protein, this mechanism will be prone to random triggering of the domino chain reaction, creating noise in the signaling system. Thermal motions in case of the “violin model" do not hinder the transduction. In fact, the permanent motion of the molecule is a prerequisite for this mechanism. Reproduced with permission from 45. Copyright 2015 Cell Trends in Biological Sciences

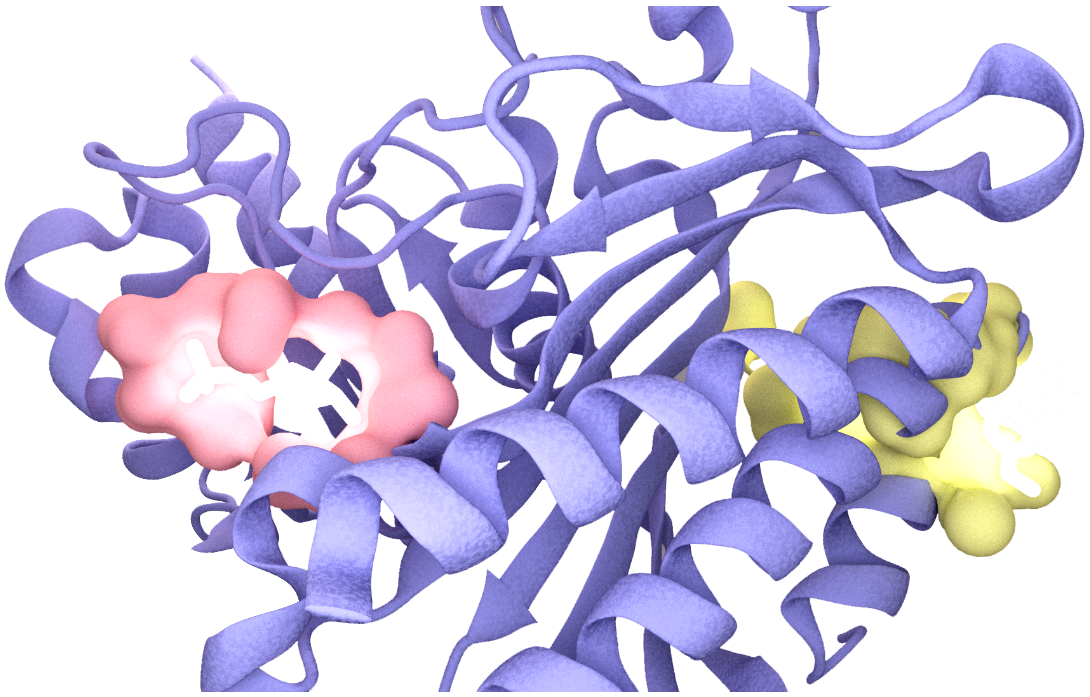


Figure 6.2: The allosteric protein fructose 1,6-bisphosphatase, shown for illustration. Orthosteric and allosteric pockets (yellow and red, respectively) are bound to an endogenous ligand and an allosteric effector, respectively. Note that the allosteric site is distant from the orthosteric site such that there is no overlap between the bound poses of the allosteric and orthosteric ligands. Despite the distance between them, the allosteric effector measurably modifies the enzymatic activity at the orthosteric site. Illustration derived from PDB IDs 2Y5K[53] and 3IFC.[54]

optimized allosteric MEK inhibitor, GSK1120212.[56] The first generation inhibitor was discovered by high throughput screening [57] and the subsequent ternary crystal structure showed the allosteric pocket to be adjacent to the orthosteric ATP-bound site.[58] By 2012 there were fourteen allosteric MEK1/2 inhibitors in clinical trials,[59] because it was recognized that an inhibitor developed for this allosteric pocket afforded two very unique opportunities to avoid adverse clinical effects; the high doses required to compete against 1mM cellular ATP concentrations and inhibition of closely related ATP-binding sites in other kinases. The unique efficacy properties of GSK1120212 highlight both the opportunity and challenge of allosteric drug design. Gilmartin et al. report that although some other MEK allosteric drugs demonstrate inhibition of the ERK1/2 pathway *in vitro*, this has not translated into efficacy in patients.[56] GSK1120212 has since been approved in the U.S. under the name Trametinib for treatment of metastatic melanoma caused by the V600E mutation.

In 2012, Saalau-Bethell et al. of Astex reported the discovery of allosteric inhibitors for the HCV NS3 protein [17]. These inhibitors produce an allosteric effect by binding at an interdomain interface and stabilizing a pre-existing autoinhibited state of the protein. The original discovery of the allosteric site was accomplished using a fragment-based HTS technique, followed by optimization using X-ray crystallography and structure-based SAR. The authors discuss their experience with a few confounding factors in the allosteric design process, namely the need to use the full-length protein in their screening construct to observe the exerted allosteric effect, and the subsequent directed evolution study of resistance mutations that could occur at the allosteric site.

Hackos et al. of Genentech published on the discovery of positive allosteric modulators (PAMs) for GluN2A-containing NMDA receptors in 2016 [60]. PAMs are allosteric ligands which increase the effect of the endogenous signalling molecule and do not cause a change in its absence. The allosteric site in this case was at a protein-protein interface, and was discovered using HTS. Subsequent medicinal chemistry efforts then optimized the early hit molecule. The authors note that the validation of this allosteric site was reinforced by its similarity to an analogous allosteric site in AMPA receptors, but that the NMDA receptor site has elements of asymmetry that the AMPA receptor site did not. In comparing two similar compounds, GNE-6901 and GNE-8324, the authors make comments that indicate evidence of mode switching or a shallow SAR landscape, and they further characterize the details of the allosteric mechanism using mutagenesis experiments.

In summary these examples demonstrate allosteric drug discovery can be successful at protein sites often considered to be undruggable. It is apparent that these successes can be further built upon through computational methods that allow for rational rather than serendipitous HTS discovery of new allosteric binding sites and a deeper understanding of allosteric mechanisms that overcome design challenges such as mode switching.

6.4 Computational Methods for Studying Allostery

6.4.1 Protein-Sequence Analysis Methods

Introduction

Protein-sequence analysis is a useful tool to detect and characterize allosteric pathways and pockets. Here, we classify sequence-based methods into two groups: 1) **“single site"** methods, which produce a list of individual functional sequence positions; and 2) **“coupled site"** methods, which produce a list of groups comprised of two or more sequence positions that appear to be functionally linked based on their coevolution.

All sequence-based analysis methods share some challenges. These challenges include how to: select and aggregate clean, relevant sequences as input; interpret the output; and integrate sequence-analysis results with other forms of data. Determining the biological meaning of a strong signal is also problematic. While many analysis methods identify evolutionarily important residues, the specific biological role of these residues cannot be inferred without additional knowledge. For example, it is difficult to determine, based on sequence alone, whether an evolutionarily significant residue plays an allosteric role, or whether its role is related to another essential process (e.g., substrate binding, maintaining protein structure, etc.).[61, 62] Indeed, it is likely that a given residue serves multiple purposes simultaneously.

Input sequence selection and alignment also present challenges. Most techniques require many sequences to establish statistical significance. To obtain the required number of sequences, researchers often lower the stringency of their search parameters, resulting in alignments that contain sequences with lower similarity or incomplete coverage of the original query. While some analysis methods manage to detect meaningful coevolution over a wide range of multiple sequence alignment (MSA) conservation and noise levels, others are more susceptible to messy data.[63, 64] For a more complete discussion of these topics and how they affect coevolution analysis methods, readers are directed to an excellent recent review by Juan et al.[65]

Single-Site Evolutionary Analysis Methods

By our definition, single-site evolutionary analysis methods return a list of predicted functional sequence positions but do not suggest specific linkages between sites. Once a researcher has constructed an MSA, the conservation or phylogenetic relevance of each column can be used to infer the evolutionary importance of each sequence position. This importance is sometimes a hallmark of thermodynamically critical residues that participate in allostery. Though single-site methods only return a list of single high-scoring sequence positions, the inner workings of some single-site methods are based on the aggregate or correlated behaviors of multiple sequence positions (e.g., to determine baseline residue probabilities within a multiple-sequence alignment or construct a phylogenetic tree).

Single-site methods for detecting allostery are advantageous because they lack much of the noise often associated with correlation analysis. These analyses are also appealing because of their simplicity: There are usually fewer parameters to set, and the results can be visualized directly by highlighting key residues on a 3D protein structure.

Single-Position Entropy. Shannon entropy, one of the simplest nontrivial sequence-analysis metrics,[66] was used widely in early works to identify conserved sequence positions for drug-design or mutagenesis experiments.[67] Similar in form to thermodynamic entropy from statistical mechanics, Shannon entropy measures the population diversity of residues at a single MSA position. It is also central to mutual information (MI), a popular coupled-pair metric. The MI of two sequence positions is defined as the sum of the individual position entropies, minus the entropy of the positions considered jointly. While we not cover the mathematical details of these methods here, interested readers are directed to previous articles on these topics.[64, 68]

Shannon entropy does not consider amino acid similarity (e.g., in the Shannon-entropy framework, a leucine-to-isoleucine mutation is considered mathematically equivalent to a leucine-to-arginine mutation). Other entropy measures, such as the relative Shannon entropy (also called the Kullback-Leibler Divergence (KLD)[69] and the von Neumann entropy,[70, 71] attempt to overcome this limitation and, as a result, may be more useful in the search for allosteric sites. Relative Shannon entropy/KLD accounts for some measure of the protein's chemical environment by considering each mutation with respect to the background amino-acid frequencies calculated from the MSA. This analysis may be particularly useful when searching for allosteric sites in proteins that reside in membranes or other noncytosolic compartments, where background residue probabilities or mutational preferences may be biased due to different biochemical contexts. In contrast, von Neumann entropy, a concept borrowed from quantum statistical mechanics, is calculated using amino-acid similarity matrices. Identifying an optimal amino-acid similarity metric is nontrivial and may well depend on the nature of the system (e.g., in a well-packed protein, residue size may be a sensitive metric, whereas surface-site comparison may require the user to prioritize charge). In a recent publication describing these types of entropy, Johansson and Toh explored how the two metrics can be mixed to detect enzyme active sites with maximum sensitivity.[64]

Zhang et al. constructed a variety of new analysis methods in 2008 by combining Shannon or von Neumann entropy, phylogenetic tree structure, and a novel gap-treatment approach.[70] In benchmarking their method, they compared their results to Evolutionary Trace and ConSurf (discussed in greater detail below). Two of their hybrid approaches outperformed all other techniques in detecting significant residues across a variety of proteins: the Improved Zoom method, which incorporates a tree breakdown of subalignments, and the Physiochemical Similarity Zoom method, which extends the Improved Zoom method with von Neumann entropy and tree-branch-size normalization.

Evolutionary Trace. Lichtarge, Bourne, and Cohen pioneered the evolutionary trace (ET) method. The approach has become quite popular, largely because the algorithm is intuitive and its results are readily visualizable.[72] ET aligns a number of sequences and constructs a phylogenetic tree, then monitors the conservation of sequence positions at major tree branching points. By slicing the tree at different similarity cutoffs, the algorithm extracts the cluster-defining sequence positions. The evolutionary significance of these sequence positions is implied by their conservation in the sequences beyond the next branch. In their first paper,[72] the authors demonstrated that ET can detect functionally important sites in SH2, SH3, and DNA-binding domains. Work has since been published on ET validation, parameter optimization, and best-use practices.[73]

In a method often referred to as “Difference-ET," the user runs ET on two related proteins and considers differences in the high-ranking residues and their scores. The sequence positions with strongly varying scores may suggest specificity determinants or differences in allosteric and/or orthosteric mechanisms. Notably, Difference-ET has been used in the study of GPCR specificity.[74–76]

To better account for varying rates of evolution in different subtrees and correlated mutations, in 2004 Mihalek et al. developed real-valued ET.[77] This method incorporates entropy information into the standard ET framework. This work also introduces the zoom ET method (not related to Improved Zoom, above), which adds higher weight to sequences that are more similar to the protein of interest. In the introductory work, they used real-valued and zoom ET to detect the functional residues in a kinase domain, then compared the performance of both methods to regular (integer-valued) ET and entropy. Given unpruned sequence data sets, the real-valued ET and zoom ET methods outperformed the others by a significant margin. In contrast, integer-valued ET prevailed in most respects when pruned data was available. An automated web server is available to perform real-valued ET calculations, generate reports, and visualize results (http://mammoth.bcm.tmc.edu/ETserver.html).[78]

H2r(s). In 2008, Merkl et al. introduced a method called H2r that serves as a segue between single-site and coupled-site approaches.[79] H2r generates a mutual-information matrix for an MSA, then discards all but the strongest detected coupled pairs. For each sequence position k, the method returns $conn(k)$, the number of top-ranked pairs that include k. Initial work proved that H2r can successfully detect functionally significant residues across a range of proteins. More recently, H2rs, an improved version of H2r, has been released.[80] This method modifies the original by using von Neumann instead of Shannon entropy and performing more detailed checks for statistical significance. H2rs is available as a web server and a stand-alone application at http://www-bioinf.uni-regensburg.de/.

Coupled-Site Evolutionary Analysis Methods

Second-order sequence analysis detects residue pairs that mutate in concert more frequently than would be expected given random genetic events. Coevolving residue pairs are assumed to be functionally linked, often because they serve essential roles in allostery or structural integrity.

The immediate output of second-order allostery analysis is a list of residue pairs with associated correlation strengths. Combining these individual pairwise correlations into a single picture of the entire protein is a separate task. On the most basic level, the strongest correlations that include a residue or site of interest can suggest thermodynamic coupling to other sites, possibly related to allostery. More complex analyses use hierarchical clustering or principal component analysis to analyze these linkages and uncover strongly linked networks of coevolving residues.

Basic Coupled-Site Analyses. Several simple yet reliable residue-coupling analyses have maintained a presence in the literature over the past decades. These basic approaches are advantageous because they are easier to understand and have been shown to score consistently well in a wide range of tests. However, they may fail to detect correlations in more complex cases.[81, 82] Though more complex methods exist, many of these basic methods still appear as analysis options in coevolution-detection software packages and web servers. In this review, we focus on a few that are still widely used.

Mutual Information. Mutual Information (MI) is one of the most straightforward and long-lived coupling metrics. The MI between two sequence positions is defined as the sum of the Shannon entropies of both positions, minus their joint entropy. Due to its simplicity and favorable mathematical properties, MI analysis is the basis for a number of more complex coevolution methods. However, MI does present certain shortcomings. For example, uncorrelated pairs of high-entropy sequence positions are likely to have a higher MI than uncorrelated pairs of low-entropy positions.[83, 84] To compensate for this and other shortcomings, various software packages have implemented a number of mathematical corrections to MI.[85–89] Further, methods to estimate baseline values for correlation (e.g., resampling or sequence shuffling) can improve MI analysis.[84, 90, 91]

Another relatively direct coevolution metric, the McLachlan-Based Substitution Correlation

(McBASC),[92] looks for similar patterns of variation in the columns of an MSA, weighting for residue similarity using the McLachlan scoring matrix.[93] Analogous methods can be constructed using different substitution matrices, but McBASC continues to be a popular choice in the literature.[63, 84, 94]

In 2002, Kass and Horovitz[95] analyzed the GroEL complex using a chi-squared test to detect significant residue coevolution in an MSA. The analysis suggested intra- and inter-chain contact pairs and has continued to appear in the literature under the name “Observed Minus Expected Squared" (OMES).[63, 81, 82, 84, 96]

Statistical Coupling Analysis. The Statistical Coupling Analysis (SCA) method developed by Lockless and Ranganathan is perhaps the most widely used sequence-based method for allostery prediction.[49] SCA draws an analogy to statistical physics by calculating a “coupling energy" between each sequence-position pair. The original SCA method computes a conservation value for each sequence position i in an MSA, applies one of several types of perturbation to another position j (depending on the SCA version,[97]) and finally recalculates the conservation at position i for the sequences that satisfy the perturbation. By calculating the change in individual and joint conservation over a variety of perturbations, SCA establishes a “coupling energy" that indicates the evolutionary coupling of positions i and j.

The output of the SCA method is an $N \times N$ matrix of coupling energies, where N is the number of sequence positions in the alignment. In early work, the researchers manually identified strongly coupled residue pairs that included one functional member (per experiment). More recent versions of SCA have grouped this matrix into meaningful clusters of coevolving residues using hierarchical or spectral clustering.[50]

Refinements of SCA have achieved improved statistical properties by resampling the original distribution.[98] In a 2011 paper, SCA was effectively used to engineer a light-sensitive LOV2 domain onto the surface of DHFR at a location that SCA had identified as energetically linked to the enzyme active site. Some variants of the resulting protein chimera were found to have acquired light-dependent activity.[99]

Further work has used SCA to design artificial WW domains.[100, 101] In 2011, an SCA analysis of antigen 85C from *Mycobacterium tuberculosis* suggested new sites that potentially could be exploited in drug design.[102] A number of projects have also demonstrated how SCA can be used to target mutations that affect protein function.[103–109]

Inspired by earlier work on the Sequence Correlation Entropy (SCE) method,[110] Dima and Thirumalai published an SCA variant in 2006.[111] This variant controls for specific protein composition by calculating the background probability that a given amino acid will be present at a random sequence position. This probability is determined by considering only the sequences being analyzed, as opposed to all sequences in the SWISS-PROT database.[112, 113] Further, they borrowed a coupled two-way clustering procedure from gene-sequence analysis to define the sectors.[114] In validating this method, the authors analyzed the PDZ, GPCR, and lectin families of proteins and were able to quantitatively predict functional residues, which were in agreement with experimental findings.

Explicit Likelihood of Subset Co-variation. As mentioned above, SCA is a “perturbation-based" method in which correlation is established by excluding certain sequences from an MSA and monitoring how entropies change. Another popular perturbation-based method was published in 2003 by Dekker et al.[115] This method, Explicit Likelihood of Subset Co-variation (ELSC), relies on similar principles but returns correlation scores in the form of probabilities rather than statistical energies. ELSC was shown to be superior to SCA in contact prediction when tested on a range of protein families. It has since been implemented on web servers[116, 117] and has been a popular benchmark method in the literature.[81, 82, 94, 96]

Direct Coupling Analysis. In 2009, Weigt et al. proposed a mutual-information-based method called Direct Coupling Analysis (DCA) that disentangles directly interacting residues from large networks of indirectly coupled sequence positions.[118] While this method is typically used in structure prediction to identify spatially adjacent sequence positions, it may find application in the study of short-range allosteric interactions. A more efficient implementation of the DCA method, known as “mean field" (as opposed to the original “message-passing" implementation) was published in 2011.[119] Both introductory papers show that DCA is a robust predictor of both intra- and inter-protein contacts and that it can hint at the existence of unobserved protein conformations. Related work has shown that DCA can be used in conjunction with structural models to generate predictive models of protein complexes,[120–122] determine the sequence positions that contribute to protein-interaction specificity,[123] and describe the conformational ensembles of proteins in crystallographic or near-crystallographic states.[124] A web server and software package are available to perform DCA analysis at http://dca.rice.edu/portal/dca/home.

PSICOV. PSICOV is another popular contact-detection method that may find productive use in the study of allostery.[125] Mathematically, PSICOV relies on an estimated inverse of the MSA covariance matrix, which acts as a matrix of correlations between all sequence-position pairs that inherently controls for the variations in all other positions. PSICOV was successful at predicting contacting protein residues based on MSA data. The code has been published online at http://bioinfadmin.cs.ucl.ac.uk/downloads/PSICOV/.

Recurrence Quantification Analysis. Recurrence Quantification Analysis (RQA), another second-order sequence-analysis technique, is best used when much is already known about the mechanism under investigation (e.g., physiochemical amino-acid properties such as charge or hydrophobicity are known to drive the allostery). RQA itself is a general method in nonlinear dynamics[126]: In the context of protein sequences, it considers a scalar-value vector that represents some property of a given sequence. In introductory work by Zbilut et al.,[127] the method was used to properly classify 56 TEM-1/β-lactamase mutants with impaired function based on their hydrophobicity profiles. Further RQA work used hydrophobicity scores to classify proteins as allosteric or nonallosteric,[128] study p53 mutants,[129] and reveal interaction partners in viral-envelope proteins.[130]

In 2005, Colafranceschi et al. investigated the effect of choosing different physicochemical amino-acid descriptors and changing the numerical parameters of the RQA algorithm.[131] More recently, a comparison method based on RQA measurements, known as cross-RQA, effectively detected protein allostery.[132] Interested readers are directed to a review by Zbilut-Webber, which provides examples of RQA applied to a range of computational biology problems.[133]

Comparative Analyses. Some work has been done to competitively benchmark the performance of these methods. In 2004, Fodor and Aldritch compared OMES, MI, SCA, and McBASC in a variety of tests. In short, the study found that performance is largely dependent on the way that different methods determine background residue probabilities and handle positional conservation.[63] A follow-up study investigated how effectively coevolution analysis finds thermodynamically linked residue pairs.[62] In general, spatially contiguous linked pairs were detected, but long-range couplings did not agree with experiments.

In 2010, Brown and Brown introduced a new pair-scoring method, called Z-scored-product Normalized Mutual Information (ZNMI), and compared it to the accuracy and reproducibility of MI, two versions of SCA, OMES, and ELSC.[82] The authors presented a thorough meta-analysis of method performance and the impact of input-parameter selection. Though none of the tools tested was particularly powerful, ZNMI was the most robust prediction tool. Brown and Brown also found that the use of multiple subalignments produced more accurate and reproducible results.

A comparative analysis of SCA and DCA revealed that the top 35 “sectons" found via spectral clustering of the DCA matrix corresponded to pairs, triplets, and quadruplets of spatially contiguous residues.[134] In contrast, a similar analysis of the SCA matrix produced spatially adjacent clusters of many residues each. These different results validate the stated goals of each method: DCA aims to find contacting pairs, whereas SCA aims to find potentially distant groups that are thermodynamically linked in a certain function.

In 2014, Pelé et al. investigated seven coevolution analysis methods to find the hallmark covarying pairs in GPCR alignments.[81] They considered three variants of MI, McLachlan Based Substitution Correlation, SCA, ELSC, and OMES. OMES and ELSC were the most robust methods for finding the residues responsible for subfamily divergence. Their article also included an insightful discussion of the methods.

Mao et al. published a comparative analysis in 2015.[135] Their study tests OMES, two variants of MI, SCA, PSICOV, and DI, and finds that PSICOV and DI are best at identifying contacting residues. OMES and MIp excel at removing false positives from the lists of predicted contacts. While the authors focused on detecting inter- and intramolecular contacts, their analysis also provided useful insights to guide the productive use of each method. For example, all methods benefit from repeatedly shuffling the MSA and rerunning the analyses in order to provide a baseline and remove false positives. Finally, the authors found that the consensus of DI and PSICOV provides a more robust prediction of contacting residues than any single prediction method alone. The software used to perform this analysis is available through the

ProDy Evol program (http://prody.csb.pitt.edu/evol/).[136]

In the course of introducing new types of MI analysis (dbZPX2, dgbZPX2, and nbZPX2) and evaluating the effectiveness of MSA simulation (a topic beyond the scope of this paper), Ackerman et al. in 2012 compared many different coevolution analyses in their ability to predict contacting residue pairs.[96] These comparisons found that the “new" methods (the ZPX2 family, DCA, and log(R) (not discussed here)), were significantly superior to the “old" methods (OMES, McBASC, ELSC, and SCA).

Web Servers. Several web servers perform and visualize sequence analyses. Given a PDB code, Contact Map WebViewer (CMWeb)[117] automatically constructs an MSA and visualizes a variety of coevolution analyses: mutual information, SCA, ELSC, OMES, and an early method presented by Gobel et al.[137] The same server can also compare the results of these methods to user-uploaded data (e.g., results the user obtained using some other type of analysis). The CMWeb server can be accessed at http://cmweb.enzim.hu/.

The Coevolution Analysis of Protein Residues server hosted by the Gerstein Lab[116] (http://coevolution.gersteinlab.org/coevolution/) can perform a large number of the coupled-site analyses presented in this review, including SCA, ELSC, MI, and McBASC-type methods employing different scoring matrices. The server can also validate the results of these methods in predicting residue distances in a crystal structure.

MISTIC (Mutual Information Server to Infer Coevolution) is an automated web server that accepts user-submitted MSAs or collects them from PFAM.[83] MISTIC uses a corrected form of MI to infer coevolving pairs and offers several analysis methods that combine structure and coevolution.[88] It can be accessed at http://mistic.leloir.org.ar/.

CAPS (Coevolution Analysis using Protein Sequences) is a unique algorithm that combines phylogenetic, 3D, and MSA data to predict coupled sequence positions.[138, 139] Versions 1 and 2 are hosted on web servers at http://bioinf.gen.tcd.ie/caps/ and http://caps.tcd.ie/, respectively.

The Interprotein-COrrelated Mutations Server (I-COMS, http://i-coms.leloir.org.ar/index.php) focuses on detecting contacts at protein-protein interfaces, though it can also return intra-chain correlations.[140] The server automatically builds alignments; performs MI, DCA, or PSICOV analysis; generates visualizations of the results; and allows users to download data taken from various points in the data-collection and analysis workflow.

In 2012, Jeong and Kim published a study describing a close MI variant.[141] They employed an automated workflow to control for various types of noise in sequence alignments, using the MSA sequence profile to establish prior knowledge about the protein. While the authors only studied a few MI variants, they stressed that their profile-based method could be extended to more complex analysis techniques. Their approach, Correlated Mutation Analysis Tool (CMAT), is available on a web server at http://binfolab12.kaist.ac.kr/cmat/.

Access to ConSurf, a single-site detection method similar to ET, is available at http://consurf.tau.ac.il/.[142–147]

Software. The ProDy Python package, referred to above, can compute a variety of coevolution metrics.[135, 148, 149] In 2014, Skjærven et al.[150] released the latest version of the powerful Bio3D R package for protein structure and sequence analysis.[151] While it focuses on structural analysis, the package can compute Shannon entropy and offers useful functions to create and manipulate sequence alignments. Also in 2014, Li et al.[152] published the CorMut software package for R, which computes MI, a metric called the “Jaccard index,"[153] and the conditional selection pressure metric K_a/K_s.[154]

Table 6.1: A table of selected coevolution web servers/software packages and their capabilities

	Web server	Downloadable	Generates MSA	Shannon Entropy	Relative Shannon/KLD	MI	SCA	DCA	PSICOV	OMES	ELSC	Notes	URL
CMWeb [137]	X	X	X			X	X			X	X	Also computes an early method from Gobel et al.	http://cmweb.enzim.hu/
MISTIC [83]	X				X	X						Calculates a corrected version of MI	http://mistic.leloir.org.ar/
CMAT [141]	X	X	X			X						Many noise-filtering parameters to customize; returns MI, MIp, and MIc	http://binfolab12.kaist.ac.kr/cmat/
I-COMS [140]	X		X			X		X	X			Offers two variants of DCA	http://i-coms.leloir.org.ar/
ET [78]	X	X										Runs Evolutionary Trace	http://mammoth.bcm.tmc.edu/ ETserver.html
Coevolution Analysis of Protein Residues [116]	X	X				X	X			X	X	Also computes similarity matrix-based methods (including McBASC), Chi Square, and Quartets coevolution metrics	http://coevolution.gersteinlab.org/
ConSurf [147]	X		X									Performs a single-site type of analysis similar to ET	http://consurf.tau.ac.il/
DCA [118, 119]	X	X										Returns pairwise DI	http://dca.rice.edu/portal/dca/home
CAPS [138, 139]	X	X										Runs a nonstandard coevolution analysis technique	http://caps.tcd.ie/
H2r [79]	X											Runs a nonstandard single-site coevolution analysis technique	http://www-bioinf.uni-regensburg.de/
H2rs [80]		X										Runs a nonstandard single-site coevolution analysis technique	http://www-bioinf.uni-regensburg.de/
Bio3D [150]		X		X								R package; useful for creating and modifying sequence alignments	http://thegrantlab.org/bio3d/
CorMut [152]		X				X						R package. Also computes Ka/Ks and Jaccard index	https://www.bioconductor.org/packages/ release/bioc/html/-CorMut.html
ProDy [148, 149]		X		X		X	X			X		Python package; can compute DI and mutual information correction/normalization	http://prody.csb.pitt.edu/evol/

6.4.2 Simulation Methods / Correlated Motions

Molecular Dynamics and Monte Carlo Methods

Protein allostery is intimately tied to protein dynamics. Allosteric effectors communicate with orthosteric pockets by altering protein dynamics, either through large-scale structural changes or more subtle changes in correlated residue motions. A proper understanding of the allosteric mechanism is impossible unless one fully appreciates the underlying dynamic conformational flexibility and energetic landscape.

Computational methods to characterize receptor flexibility include molecular dynamics (MD) and Monte Carlo simulations. Briefly, MD simulations represent molecular systems as beads (e.g., atoms) connected by springs (e.g., bonds). Newton's equations of motion are numerically integrated to propagate the dynamics of this classically formulated system.[155–158] Simulation time is typically discretized into steps of one or two femtoseconds, as required to accurately simulate the fastest atomic motions (i.e., hydrogen bond stretching). At each step, the potential energy of the system is calculated by considering the energies associated with bonded atoms (e.g., due to bond stiffness, angle bending, torsion rotation, etc.) and nonbonded atom pairs (e.g., due to electrostatic and van der Waals forces). The forces on each atom, calculated from the negative gradient of the potential energy, are then used to update the atomic positions at each step. In a related method, called "accelerated MD" (aMD), [159–164] the underlying potential energy landscape is modified by raising the energy wells. These modifications facilitate transitions between states that do not normally occur on the time scales of traditional MD simulations. Interested readers are directed to recent reviews that describe MD simulations in the context of drug discovery.[165–168]

In contrast, Monte-Carlo-based methods take a stochastic approach to conformational sampling.[169, 170] At each Monte Carlo step, a Markov-chain procedure is used to generate a slightly modified system conformation. This new conformation is then accepted or rejected at random, biased by the relative potential energy of the new conformation as compared to that of the previous step (the so-called "Metropolis criterion."[169]) As these simulations are stochastic, they do not typically explore conformational space via the same time-dependent path traced in an MD simulation. Indeed, a key limitation of the Monte Carlo method is that the time information connecting the states is lost. Nevertheless, Monte

Carlo simulations are widely used because they tend to more efficiently sample system conformations that are statistically representative of the equilibrium state. A recently published book by Landau and Binder describes the technique in detail.[171]

Network Representations and Protein Allostery

As mentioned above, allosteric signals can be transmitted through large-scale conformational changes or subtle shifts in the correlated motions/interactions of individual residues.[31, 35] Accessing the time scales required to observe many large-scale conformational changes is presently computationally intractable using unbiased simulation-based methods,[172, 173] but these simulations are well suited to the study of allosteric signals that are transmitted via fast, local fluctuations (i.e., nanosecond-scale changes in the coordinated motions of adjacent residues).

Distilling the dynamics sampled by these simulation-based methods into a simple network representation of protein motions often facilitates allosteric analysis.[174–180] Rather than tracking the position of every atom, each protein constituent (e.g., amino acid) is represented as a single node (located, for example, at the residue center-of-mass). Each pair of nodes is connected by an edge whose length is inversely proportional to the degree of interdependence between their motions, such that nodes connected by short edges are highly interdependent. Different metrics of “interdependence" have been used, including metrics based on correlated motions (e.g., mutual information,[181, 182] fluctuations in atomic positions,[183], etc.) or the number of specific noncovalent interactions.[183]

Once a network representation of protein dynamics has been constructed, various network-analysis techniques (described below) can identify important pathways of communication between distant residues that may contribute to allostery. Interested readers are directed to a recent review by Atilgan et al. for more detailed information.[184]

Dynamical Network Analysis

A number of simulation-analysis techniques consider the interdependence of individual residues along an allosteric path or paths. In 2008, Bradley et al. analyzed molecular dynamics simulations of *Escherichia coli* NikR, a transcriptional repressor of the NikABCDE nickel permease, using a novel network-space clustering approach.[183] Two types of residue-residue interdependence were considered, based on fluctuations in residue motions and the number of polar contacts between residue-pair members. Bradley et al. first identified sets of residues whose motions and/or contact counts were correlated with residues in both allosteric and orthosteric pockets. These sets were then clustered using the “unweighted pair group method with arithmetic mean" (UPGMA), a hierarchical agglomerative approach that groups residue sets according to the degree of similarity in their correlation patterns. The authors ultimately identified residue sets likely to mediate communication between the NikR allosteric Ni^{2+}-binding pocket and the orthosteric DNA-binding pocket.

Others have pursued orthogonal methods to represent and analyze protein dynamics *in silico*. In 2009, McClendon et al. introduced MutInf, a novel MD-analysis technique for identifying statistically significant correlated motions.[181] Unlike other techniques, MutInf uses an internal coordinate scheme. Rather than considering the positional coordinates of residue atoms directly, it quantifies correlated motions using a mutual-information metric derived from residue torsion angles. This approach makes MutInf particularly well suited to study allosteric signals transmitted through the gear-like correlated motions of side-chain rotamers. Using multiple short simulations, the authors were able to identify statistically significant correlations. Applying the technique to interleukin-2 provided a viable theory of allosteric transmission involving a population-shift-mediated signal transmitted between the allosteric and orthosteric pockets via a hydrophobic core. More recent work has incorporated the Kullback-Leibler divergence metric to improve the sensitivity of MutInf analysis.[182]

A paper by Van Wart et al., published in 2012, studied imidazole glycerol phosphate synthase (IGPS), an important enzyme in the histidine biosynthetic pathway of plants, fungi, and microbes.[178] Following a molecular dynamics simulation of the enzyme, they built a representative network of residue interactions as inferred by correlated motions. The edges connecting physically distant protein residues were ignored to emphasize correlations resulting from immediate physical interactions. The single shortest path (in network space) between the allosteric and orthosteric sites was then identified. Two of the residues along this path had been shown previously by experiment to be involved in allostery.

Van Wart et al. later expanded on their work by creating the freely available Weighted Implementation of Suboptimal Paths (WISP) program.[185] WISP can identify not only the single optimal path connecting allosteric and orthosteric sites, but also multiple other near-optimal paths. While allosteric signaling may occur through a single optimal path in some cases, for many proteins it is likely the summed (or perhaps synergistic) effect of signaling through multiple near-optimal pathways.

While several efficient algorithms exist to calculate the single optimal path between two nodes in a network space (e.g., the Floyd–Warshall algorithm, Dijkstra's algorithm, etc.), calculating near-optimal pathways is far more computation intensive. WISP uses several methods to first simplify the network representation of system dynamics. First, the edges between physically distant nodes are removed regardless of the interdependence of their motions, as in van Wart's original paper. Second, all nodes that cannot possibly participate in optimal or near-optimal pathways of allosteric communication are deleted. For each node, Dijkstra's algorithm is used to calculate the shortest path connecting the source node, the node being considered (n_i), and the sink node. If this shortest path is still too long (in network space) to be an optimal or near-optimal pathway, no other path passing through n_i can be optimal or near-optimal. Therefore, n_i is deleted from the network. After thus greatly simplifying the network, it becomes computationally tractable to calculate notable pathways of allosteric communication between source and sink residues using a recursive, bidirectional approach.

In 2014, LeVine et al. introduced a novel analysis framework, called N-body Information Theory (NbIT), that uses multivariate mutual information to measure residue interdependence. This technique is unique in that it considers not only the interdependence (mutual information) between two residues, but also the mutual information between *groups* of up to *N* residues. Specifically, a metric called "coordination information" is used to measure the degree to which the correlated motions of a set of residues are shared with another residue not in the set (i.e., the "contribution of a site to all possible *n*-body correlations with another site"). The authors use this technique to analyze MD simulations of the bacterial transporter LeuT and identify many of the same residues previously found through experiment.[186]

Community Analysis

A number of simulation-analysis techniques expand on calculations of residue-residue interdependence to consider how highly interconnected (and often rigid) *communities* of residues interact with one another. In this view, allostery is achieved when largely isolated communities of highly interconnected residue nodes (“small-world networks") engage in long-range interactions. For example, in 2012, Rivalta et al. studied the allosteric mechanism of PRFAR binding to the IGPS heterodimer.[187] After simulating the system in both the absence and presence of the allosteric modulator, they calculated the mutual information between the positional vectors of all residue-residue pairs. Ignoring the correlations between nodes that were not frequently physically adjacent (≤ 5.0 Å for at least 75% of the simulation), the authors next used the Girvan-Newmann algorithm to identify communities of highly interconnected nodes. This analysis provides a powerful illustration of how an allosteric modulator might manipulate entire protein “modules" to transmit a signal. Supported by NMR experiments, the authors ultimately concluded that PRFAR binding decreases the correlated motions at sideL of the heterodimer and induces conformational changes in sideR that ultimately impact the hydrogen-bond network near the active site.[187] Sethi et al. used a similar technique in 2008 to study allostery in tRNA:protein complexes.[174]

The use of accelerated MD (aMD) simulations in this same mutual-information/community-analysis workflow has also proved fruitful. aMD simulations are particularly helpful when the allosteric mechanism being studied occurs on time scales that are longer than those accessible through traditional MD. For example, in 2012, Gasper et al. used a mutual-information/community-analysis technique to study α-thrombin.[175] Ultimately, their results suggested that α-thrombin may have two allosteric pathways. One occurs on slower time scales that might not have been accessible were it not for the aMD. Others have used aMD to similarly study allosteric effects in the M2 muscarinic receptor.[188]

In 2014, Blacklock and Verkhivker used a similar technique to analyze MD trajectories of the Hsp90 chaperone in various functional states.[189] To verify the existence of communities in this system, they first calculated force constants for each residue, corresponding to the energy cost of residue displacement over the course of the equilibrium simulation. As residues with high force constants are typically rigid, community boundaries (e.g., the boundary between a rigid module and a flexible inter-domain hinge site) often correspond to locations where these constants change abruptly. In fact, the “hotspot" regions did correspond to residues known experimentally to be important hinge sites or sites of dimerization.

To further characterize this community-based organization, the authors next used protein-structure network analysis to build a network representation of Hsp90 dynamics. They calculated cross-correlation matrices from the MD simulations and identified residues with high degrees of “centrality" (the number of interacting residues) and “betweenness" (the frequency with which a given node lies along the shortest path in network space between two other nodes). Based on this analysis, the authors concluded that Hsp allosteric communication is mediated by highly stable central nodes, surrounded by residues with high degrees of betweenness that may “shield" the more critical residues from the random Brownian/thermal motions common to all proteins.

6.4.3 Pocket Detection

Introduction

Simple pocket-finding techniques can be useful to identify potential allosteric sites. One straightforward approach is to consider any identified pocket other than the orthosteric to be allosteric. Pocket-detecting algorithms can be used to analyze static crystal structures. However, coupling these algorithms with techniques that account for pocket dynamics is often useful. Some allosteric pocket conformations are extremely rare in the absence of the allosteric effector itself. These “cryptic pockets" are not readily evident in static apo- or ligand-bound crystal structures, but they may be predictively sampled with molecular dynamics simulations.[190–194] Computational methods for pocket detection can be classified as geometry, knowledge, and energy based. We describe these three types of methods next.

Geometry-Based Pocket Detection

Geometry-based methods identify pockets by considering only the positions of the receptor atoms themselves. These methods are well suited to situations where speed of computation is critical (e.g., when pocket detection is applied to conformations extracted from entire MD trajectories) or where pockets are particularly well defined. On the other hand, accuracy in pocket detection suffers when pockets are shallow, partially collapsed, or highly flexible. And pocket ranking by simple geometric metrics (e.g., volume) does not always correlate strongly with ligand binding affinities or druggability because geometry-based methods do not consider the chemical properties of a given pocket.[195–205]

Many geometry-based algorithms have been described in the literature. They can be classified into three subcategories: sphere based, α-shape based,[206] and grid based. Sphere-based methods, such as PASS,[198] PHECOM,[207] POCASA,[208] and SURFNET,[209] first cover protein surfaces with spheres. Each sphere is evaluated and eliminated if judged unlikely to occupy a surface cavity. The locations of the remaining spheres identify likely binding pockets.

In contrast, methods such as APROPOS,[210] CAST,[211] CASTp,[212] and Fpocket,[199] use α-shape schemes to identify protein pockets. In brief, an α-shape is like a convex hull, except that it potentially follows the surface of the source points more closely than a convex hull. The degree to which it deviates from the convex hull is controlled by the α value (for more details, see Edelsbrunner et al.[213]). APROPOS compares multiple α-shape representations of a protein at different values of α.[210] Pockets are identified by subtracting “higher-resolution" α-shapes from “lower-resolution" shapes. In contrast, CAST generates a Voronoi diagram based on protein-atom locations, removes any Voronoi edges and vertices that fall completely outside the receptor, and identifies pockets as collections of empty triangles (for details, see[211]). Fpocket uses a Voronoi tessellation to calculate receptor α-spheres (i.e., spheres that include four protein atoms on their surface but no protein atoms in their interior).[199] When clustered, these spheres tend to congregate at potential pocket sites.

Grid-based geometric methods for pocket detection include GHECOM,[214] LIGSITE,[215] LIGSITE^{CSC},[216] POCKET,[217] Pocket-Picker,[218] PocketFinder,[219] Q-SiteFinder,[220] and POVME/POVME Pocket ID.[195, 221] These techniques consider points spaced along a grid that encompasses the protein receptor. Each point is evaluated for the likelihood of pocket occupancy, and high-scoring points are clustered to identify binding pockets, whether orthosteric or allosteric. Various metrics are used to evaluate the points, depending on the algorithm. They include the presence of steric clashes with the protein, detected protein-solvent-protein events,[215, 217] buriedness indices,[218] predicted methyl-probe interaction energies,[219, 220] etc.

As a representative example of the geometry-based class, consider POVME/POVME Pocket ID, a grid-based pocket-finding algorithm.[195, 221] POVME Pocket ID identifies surface and internal cavities by first creating a low-resolution 3D grid of equidistant points that encompasses the entire protein. Points are removed if they are physically near any protein heavy atom or fall outside the convex hull defined by the protein α-carbons. The volumes corresponding to the remaining low-resolution points, which tend to encompass protein pockets, are then replaced with smaller, higher-resolution 3D grids, and the same point-eliminating process is repeated. Finally, any remaining high-resolution points that have fewer than a user-defined number of neighbors are eliminated iteratively until no such points remain, and stretches of contiguous points are grouped together into distinct pockets.

The locations of these identified pockets can be fed into the POVME program, which calculates pocket metrics such as volume and shape. The latest version of POVME can efficiently analyze entire MD trajectories, allowing the user to monitor pocket changes over time and identify sampled conformations that are geometrically distinct. This approach is useful for identifying cryptic pockets that only occasionally manifest themselves over the course of an MD simulation.

Knowledge-Based Pocket Detection

Knowledge-based algorithms draw on large structural and genomic databases to determine binding-pocket locations based on sequence and overall structure rather than pocket shape. Algorithms such as 3DLigandSite,[222] FINDSITE,[223] and Pocketome[224] infer these locations by querying existing databases for ligand-bound proteins that are structurally similar to the target protein. These algorithms work particularly well when the target protein has a highly conserved binding site. Other methods, such as FRpred, use sequence conservation, in conjunction with supporting methods like surface-residue-prediction algorithms, to identify surface residues that are likely biologically important.[225] Finally, some hybrid methods, such as the evolutionary-trace method,[226] ConSeq,[142] and ConCavity,[227] combine sequence and structural homology to identify likely binding pockets. The “Single-Site Evolutionary Analysis Methods" section above describes methods like these in further detail.

Energy-Based Pocket Detection

Energy-based methods identify binding sites by evaluating whether or not a given protein region interacts favorably with docked small organic probes. These methods are particularly useful to detect a given pocket *and* assess its druggability. This additional information does come at a cost; energy-based methods are more computationally demanding and so do not scale well, limiting their applicability when analyzing large-scale structural datasets or MD trajectories.

Methods such as SiteMap[202] and SITEHOUND[228] use single probes (typically methane or water molecules) to assess the likelihood of small-molecule/protein binding. In contrast, methods such as FTMap,[203, 229, 230] GRID,[231] and MCSS[232] consider multiple chemically diverse probes and so identify not only pockets, but druggable hot spots that are more likely to function as orthosteric or allosteric binding sites.

FTMap[203, 229, 230] in particular warrants further discussion. FTMap is the computational equivalent of the Multiple Solvent Crystal Structures (MSCS) technique.[233, 234] Rather than superimposing multiple crystal structures obtained by soaking a protein in 6-8 different organic solvents, FTMap computationally docks 16 distinct organic probes into a user-provided protein model. The docked positions of these probes are then clustered, and the clusters are ranked by their Boltzmann-averaged energies. In an alternate implementation called FTSite,[235] adjacent, mutually accessible clusters are combined into “sites" that are then ranked by the number of nonbonded contacts between the protein and the corresponding probes. Both these techniques can be applied to multiple structures, whether derived from X-ray crystallography or simulation, to potentially identify cryptic sites that are not always evident when single structures are considered. FTMap can be used to identify possible allosteric sites. For example, Ivetac et al. used it to detect five potentially druggable sites on β_1AR and β_2AR.[236, 237] It has also been used to evaluate the druggability of p53 cryptic pockets, which were subsequently used to prospectively discover novel reactivation compounds.[192]

6.4.4 Markov State Models

Introduction

Over the past 10 years, Markov state models (MSMs) have been used increasingly to analyze molecular dynamics simulations, suggesting that they are already promising tools to study allostery in a drug-discovery context.[194, 238–242] In brief, a Markov state model is a stochastic kinetic model that describes the probability of transitioning between discrete states at a fixed time interval.[238, 243, 244] These models are required to have the Markovian property (i.e., thatvthe probability of transitioning between discrete states is independent of previous transitions).

By clustering protein structures extracted from an MD trajectory, it is possible to identify discrete conformational states for use in MSMs.[238, 243, 244] Transitions between conformational clusters observed over the course of an MD trajectory are tallied, and the MSM is then built from the transition probabilities between these distinct clusters. As they are built solely on these transition probabilities, MSMs can draw upon multiple trajectories, enabling efficient sampling of the entire conformational space through many independent simulations carried out in parallel. MSMs can be built with the software package MSMBuilder, available at http://msmbuilder.org/latest/.[243] EMMA[245], available at http://www.emma-project.org/latest/, is another useful building tool. Because independent trajectories are aggregated as a post-processing step, the simulations can be carried out on a variety of computational resources, including independent desktop machines, local clusters, or high-performance computing resources. A recent example carried out by Kohlhoff et al. used tens of thousands of trajectories carried out on Google's exascale cloud computing resources. By subsequently "knitting together" the simulation data within a MSM framework, the researchers elucidated the activation pathway of GPCR $\beta_2 AR$.[246]

MSM/MD analysis provides access to the thermodynamic, kinetic, and structural characteristics of the protein conformational ensemble (i.e., a robust description of the free-energy landscape of the protein).[238, 243–245, 247, 248] The thermodynamics of the various conformational states can be calculated from the equilibrium distribution. It is also possible to resolve the transition kinetics between individual states, the concerted or principal protein motions, metastable states, and the transition pathways between discrete states.[243, 245, 247] Lastly, the MD trajectories also provide representative receptor snapshots for use in structure-based drug design.[248]

MSMs and Drug Discovery

MD-based MSMs were originally developed to study protein folding,[239, 249–254] but they can also be used to study molecular phenomena that are directly applicable to allostery and drug discovery, including cryptic site identification,[191, 194, 246] protein-ligand interactions,[255, 256] and protein function.[51, 246, 249, 257]

MSMs have been used to study binding-site conformational heterogeneity, revealing conformations that are not readily evident in any crystal structure. In 2012, Bowman and co-workers used MSMs to identify and validate cryptic allosteric sites in TEM-1 β-lactamase, interleukin-2, and RNase H.[191, 194]

Similarly, in a recent study of GPCR conformational transitions, Kohlhoff et al. identified intermediate conformations that they used successfully in a subsequent virtual-screening campaign.[246] Shukla and co-workers also identified a key conformational intermediate in the Src kinase activation pathway that could be used as a target for drug discovery.[258] While traditional molecular dynamics simulations have been used to identify cryptic allosteric sites and elucidate binding-site conformational diversity, Markov state models can characterize the probability and dynamics of these conformations, permitting rational design against dynamic pockets.

Researchers have also employed MSMs to gain insights into ligand-protein binding that are critical to both allosteric and orthosteric structure-based drug discovery. Buch et al. used MSMs derived from multiple independent MD simulations to accurately determine the k_{on} and binding free energy of benzamidine to trypsin.[255] Building on that work, Tiwary et al. used MSMs in conjunction with metadynamics, a variant of molecular dynamics, to determine the k_{off} for the same system.[256] While this approach is currently too computationally demanding for virtual screening, it has identified transitory sites important for binding and shed new light on the interactions that govern ligand-binding kinetics.

MSMs also provide useful insights into how a protein conformational ensemble begets function and transmits allosteric signals. Most studies have focused on the transition between two functional states. In this context, MSMs have been used to study Src kinase,[258] the β_2 adrenergic receptor (β_2AR),[246] and bacterial response regulator NtrC.[52, 257] Malmstrom et al. recently used this same technique to study allostery in the cyclic-nucleotide binding domain (CBD) of protein kinase A.[51] By building and simulating models of the CBD, with and without bound cAMP, the study provided insight into how ligand binding modulates the protein conformational ensemble.

As it is challenging to determine the impact of an effector molecule on the conformational ensemble, most MSM studies to date have not examined allostery directly. Nevertheless, two relevant themes emerge from existing studies. First, kinetics play an important role in the transition between functional states.[51, 52, 257] This is best shown in our study of the PKA CBD, where the effector molecule cAMP significantly modulates the kinetics of the transition from the active to the inactive states of the protein, while simultaneously leaving the reverse transition between the inactive and active states unmodified.[51] This modulation seems to be regulated by transient interactions between the effector molecule and the protein.

Second, recent MSM studies suggest that there is not one but many pathways between functional states. Studies of β_2AR,[246] NtrC,[52, 257] and the PKA CBD[51] each revealed multiple transition pathways that arose from transient interactions within the system, suggesting a probabilistic model of allostery in addition to one that describes allostery in terms of concerted pathways.[257] Studies like these can provide useful insights applicable to structure-based drug discovery targeting transitional conformational states.

MSMs can provide significant insights into protein-ligand interactions and allostery, but building these models is computationally demanding, limiting their use as screening tools. This weakness aside, MSMs do provide a strong foundation for rational drug discovery and enhanced understanding of allosteric mechanisms.

6.5 Energy Landscape and Topological Analyses

6.5.1 Introduction

Like MSMs, energy-landscape theories were originally developed to study protein folding. But they too can be applied to the study of allosteric mechanisms and protein dynamical modes. Per statistical mechanics, the underlying free-energy landscape of a system determines the probability of observing a particular configuration. Any topological analysis aimed at understanding allostery must be based on a thermodynamic understanding of the impact that effector binding has on this energy landscape.

6.5.2 Protein Frustration

It is helpful to view allosteric binding as an event that perturbs the conformational exchange, or folding pathways, of a protein. The principles of statistical mechanics applied to polymers in solution have provided much insight into the protein-folding process. As proteins fold, the number of native contacts increases, and the entropy of the polymer decreases. Folding is impossible when the required conformational transitions are occluded by large free-energy barriers or wells. For example, a large local minima along the folding pathway could force a protein to undergo a glass transition, leading to a kinetically trapped state. Typical folding must be governed by the “principle of minimum frustration" (i.e., the folding pathway must be funnel shaped and relatively smooth). [259–264]

In terms of allostery and energy-landscape theory, a positive allosteric modulator can be seen as a compound that increases the likelihood of the binding state by deepening the binding-state well or decreasing the depth of competing wells. On the other hand, a negative allosteric modulator may shift the location of the binding-site well, shallow out the well, or increase the depth of competing wells. Using these guiding principles, heuristics can be designed to identify each of the above scenarios.

Ferreiro et al. have developed a heuristic to pick out locally frustrated residues.[265, 266] While the overall folding process obeys the principle of minimum frustration, this does not preclude the possibility of having small local barriers that must be overcome to fold. To detect these local frustrations, the authors systematically perturbed the amino-acid sequence and compared the perturbed total energy with that of the native state. This comparison provided the effective stabilization energy associated with each amino-acid pair. If a given native-state interaction is highly stabilizing (i.e., the interaction energy is favorable compared to mutants), it is said to be “minimally frustrated." On the other hand, if a pair of residues is sufficiently destabilizing (i.e., the interaction energy is weak compared with that of the mutants), it is “frustrated." Frustrated residues can be thought of as weak points in the global structure, contributing to barriers that may introduce slow degrees of freedom. As such, they may be involved in ligand binding or allostery. Jenik et al. have implemented this heuristic as a web server http://www.frustratometer.tk/.[267]

Weinkam et al. used a dual-topology Gõ model[268] to survey the free-energy landscape of the calmodulin-GFP Ca^{2+} sensor protein, the maltose binding protein, and the CSL transcription factor. They introduced a truncated Gaussian distance term to the soft-sphere atom overlap term implemented in MODELLER,[269–272] resulting in either one basin corresponding to the bound/unbound distances or two corresponding to both, depending on the pairwise distance and cutoff. Subsequent MD simulations of the resulting model were analyzed using their pseudocorrelation map algorithm.[273] They found that their model reproduces allosteric motions and sheds insight into the macroscopic mechanism.

6.5.3 Normal Mode Analysis

Protein dynamics and subsequent perturbations from binding events can also be studied using normal mode analysis (NMA).[274–276] NMA assumes that the potential energy landscape in the vicinity of a minimized atomic structure is approximately harmonic. This simplifying assumption allows for diagonalization of the Hessian matrix. Solving the eigenvalue problem produces the eigenvectors (movement direction) and eigenvalues (vibrational frequencies). NMA is known to reproduce the slow degrees of freedom of protein motion well.[274, 277] Applying additional constraints to simulate the impact of an effector on binding-site residues can yield new fundamental modes that may provide insight into the allosteric mechanism.

Ming and Wall applied normal mode analysis to investigate the effect of tri-*N*-acetyl-D-glucosamine upon lysozyme.[278] They assumed that greater perturbation to the conformational distribution corresponds to increased likelihood that a binding event will occur at a regulatory or active site. The degree of perturbation is monitored by comparing the distributions predicted by NMA of the protein with and without the ligand bound, via the Kullback-Leibler Divergence. In this work, Ming and Wall generated many arbitrary poses and investigated the allosteric potential, defined by the KLD. They showed that sites with large KLD correspond with identified functional lysozyme residues.

The use of normal mode analysis removes the need to integrate the equations of motion. However, for the harmonic approximations to be accurate, the starting structure needs to be in a local minima. An initial molecular mechanics minimization can ensure that proteins of interest are near what might be seen in their native environments (e.g., a GPCR embedded in a lipid bilayer with heterogeneous composition). However, alternate options include simplified interaction hamiltonians such as those offered by elastic network models (ENMs).

Elastic Network Models. The complexity and computational cost of simulating detailed atomic potentials inspired development of simpler ENMs in which uniform harmonic potentials are used to model all interactions.[279–281] A popular ENM is the Gaussian Network Model (GNM), wherein neighboring residues are connected by virtual springs to create a network/graph of interactions. Here the interatomic potential energy, U, of the system can be expressed as,

$$U = \frac{\gamma}{2}\left[\sum_{i,j}^{N} \delta R_i \Gamma \delta R_j\right], \tag{6.1}$$

where γ is the uniform force constant of the harmonic springs, δR is the N-dimensional column vector of the instantaneous fluctuations of specific atoms, and Γ is the $N \times N$ Kirchoff (connectivity) matrix of residue interactions given by,

$$\Gamma_{ij} = \begin{cases} -1, & \text{if } i \neq j \text{and } R_{ij} \leq R_c \\ 0, & \text{if } i \neq j \text{and } R_{ij} > R_c \\ -\sum_{i,i \neq j}^{N} \Gamma_{ij}, & \text{if } i = j \end{cases} \tag{6.2}$$

where R_c is a residue cutoff distance.

Ming and Wall have developed a backbone-enhanced elastic network model (BENM) in which the interactions between connected residues are scaled by an additional factor. Using an objective function defined by the KLD of the marginal distribution of all-atom and C_α atomic coordinates, the authors showed that BENM can reproduce the atomistic mean-squared displacement for bovine pancreatic trypsin inhibitor (BPTI).[282] To consider the effects of allostery, the authors again used the KLD of local conformational distributions before and after ligand binding.[282]

To locate allosteric binding sites, Su et al. recently identified allosterically coupled regions in a GNM using a thermodynamic method.[283] All neighboring residues (represented by their corresponding C_α atoms) were connected by virtual harmonic springs of equivalent strength. A spherical probe then sampled the protein surface to locate the potential effector interaction points. Additional springs were attached between the effector point and the local residues in the protein-effector model. The free-energy difference ($\Delta\Delta G$) between the protein-ligand and protein-effector-ligand systems was calculated. Regions with large $\Delta\Delta G$ values were assumed to be important for the allosteric mechanism. Using this method, Su et al. studied the Hsp70 and the GluA2 AMPA receptors. These allosteric predictions corresponded to experimental results.[283]

Berezhovsky et al. used a similar coarse-grained network-interaction model to predict the location of allosteric sites.[284] To identify potential pockets for effector/substrate binding, the authors generated a “residue interaction graph" (RIG), with residues as nodes and interatomic contacts as edges. They then evaluated the “local closeness," defined as the number of total neighboring residues weighted by $1/r$, where r is the inter-residue distance.[285] Regions with high local closeness were likely to be in pockets and, thus, were putative binding sites. The extent to which each binding site was coupled to the intrinsic protein motions was then explored by estimating the strain (so-called “binding leverage") on ligand-protein contacts under deformations described by NMA. A ligand that bound to a site with high binding leverage could restrict protein dynamics.[286]

The binding leverage describes the extent to which changes in binding sites and protein conformational degrees of freedom are coupled; it does not specifically predict allosterically linked sites. To predict allostery, Mitternacht and Berezovsky introduced “leverage coupling," which selects for sites that are strongly coupled to the same degree of freedom.[286] The mode associated with the most binding leverage is considered most relevant, based on the assumption that there is one dominant motion promoting allostery. These methods have been implemented in the online SPACER server http://allostery.bii.a-star.edu.sg/.[287]

6.6 Conclusions

Over the past decade, advances in computing power and predictive algorithms coupled with the increased availability of structural and biochemical data have revealed new opportunities for rational design of allosteric drugs. The emergence of novel computational approaches to describe and predict allosteric phenomena across a range of scales, from the coordinated atomic movements in a single receptor molecule to complex allosteric signaling networks, is ushering in a new era wherein computational methods can be used to prospectively predict, discover, and characterize allosteric sites and effector molecules. Within the context of a drug-discovery program, such approaches hold the potential for developing drugs with increased specificity and selectivity, as well as the ability to gain new and more comprehensive understanding of old targets. For example, the convergence of advances in (i) theoretical MSM-based frameworks and MSM building software, (ii) community MD codes that can achieve $>$ 100 ns/day sampling for realistic sized systems on single gaming/commodity GPU processors, and (iii) pocket and druggable site-detection algorithms now make it possible for researchers even in industrial settings, with fast-paced timelines and stringent quality standards, to apply these approaches to drug targets already in their arsenals. The application of these methods to kinases and GPCRs seems particularly worthwhile, given the existence of assays and structural data, and the challenges faced by existing drug candidates in the clinic.

6.7 Acknowledgements

Chapter 6, in full, is a modified reprint of the material as it appears in “Wagner, J. R.†; Lee, C. T.†; Durrant, J. D.; Malmstrom, R. D.; Feher, V. A.; Amaro, R. E. *Emerging Computational Methods for the Rational Discovery of Allosteric Drugs.* Chem. Rev. 2016, 116 (11), 6370–6390.”. The dissertation author was a primary coinvestigator and author of this work.

This work was funded in part by the Director’s New Innovator Award Program (NIH DP2 OD007237 to REA) and the National Science Foundation through XSEDE supercomputer resources provided via TG-CHE060073N to REA. Funding and support from the National Biomedical Computation Resource was provided through NIH P41 GM103426. JRW and CTL are supported by a NIH Molecular Biophysics Training Grant T32 GM008326. Additional resources from the UC San Diego Center for Drug Discovery Innovation and the Keck II Computing and Visualization facility are gratefully acknowledged.

References

(1) Christopoulos, A.; May, L.; Avlani, V.; Sexton, P. *Biochem. Soc. Trans.* **2004**, *32*, 873–877.

(2) Groebe, D. R. *Drug Discov. Today* **2006**, *11*, 632–639.

(3) May, L. T.; Leach, K.; Sexton, P. M.; Christopoulos, A. *Annu. Rev. Pharmacol. Toxicol.* **2007**, *47*, 1–51.

(4) Kenakin, T. P. *J. Biomol. Screen.* **2010**, *15*, 119–130.

(5) Nussinov, R.; Tsai, C.-J. *Curr. Pharm. Des.* **2012**, *18*, 1311–1316.

(6) Szilagyi, A.; Nussinov, R.; Csermely, P. *Curr. Top. Med. Chem.* **2013**, *13*, 64–77.

(7) Grover, A. K. *Med. Princ. Pract.* **2013**, *22*, 418–426.

(8) Ma, B.; Nussinov, R. *Curr. Pharm. Des.* **2014**, *20*, 1293–1301.

(9) Gunasekaran, K.; Ma, B.; Nussinov, R. *Proteins* **2004**, *57*, 433–43.

(10) Wenthur, C. J.; Gentry, P. R.; Mathews, T. P.; Lindsley, C. W. *Annu. Rev. Pharmacol. Toxicol.* **2014**, *54*, 165–184.

(11) Groebe, D. R. *Drug Discov. Today* **2009**, *14*, 41–49.

(12) Csermely, P.; Nussinov, R.; Szilagyi, A. *Curr. Top. Med. Chem.* **2013**, *13*, 2–4.

(13) Muchmore, S. W.; Hajduk, P. J. *Curr. Opin. Drug Discov. Devel.* **2003**, *6*, 544–9.

(14) Jorgensen, W. L. *Science (80-.).* **2004**, *303*, 1813–1818.

(15) Doman, T. N.; McGovern, S. L.; Witherbee, B. J.; Kasten, T. P.; Kurumbail, R.; Stallings, W. C.; Connolly, D. T.; Shoichet, B. K. *J. Med. Chem.* **2002**, *45*, 2213–2221.

(16) Wood, M. R.; Hopkins, C. R.; Brogan, J. T.; Conn, P. J.; Lindsley, C. W. *Biochemistry* **2011**, *50*, 2403–2410.

(17) Saalau-bethell, S. M.; Woodhead, A. J.; Chessari, G.; Carr, M. G.; Coyle, J.; Graham, B.; Hiscock, S. D.; Murray, C. W.; Pathuri, P.; Rich, S. J.; Richardson, C. J.; Williams, P. A.; Jhoti, H. *Nat. Chem. Biol.* **2012**, *8*, 920–925.

(18) Rawal, R.; Murugesan, V.; Katti, S. *Curr. Med. Chem.* **2012**, *19*, 5364–5380.

(19) Reynolds, C.; de Koning, C. B.; Pelly, S. C.; van Otterlo, W. a. L.; Bode, M. L. *Chem. Soc. Rev.* **2012**, *41*, 4657.

(20) Gentry, P. R.; Sexton, P. M.; Christopoulos, A. *J. Biol. Chem.* **2015**, *290*, 19478–19488.

(21) Regan, M. C.; Romero-Hernandez, A.; Furukawa, H. *Curr. Opin. Struct. Biol.* **2015**, *33*, 68–75.

(22) Rutkowski, P.; Lugowska, I.; Kosela-Paterczyk, H.; Kozak, K. *Onco. Targets. Ther.* **2015**, *8*, 2251.

(23) Meng, H.; McClendon, C. L.; Dai, Z.; Li, K.; Zhang, X.; He, S.; Shang, E.; Liu, Y.; Lai, L. *J. Med. Chem.* **2015**, DOI: 10.1021/acs.jmedchem.5b01011.

(24) Wu, P.; Nielsen, T. E.; Clausen, M. H. *Trends Pharmacol. Sci.* **2015**, *36*, 422–439.

(25) Jazayeri, A.; Dias, J. M.; Marshall, F. H. *J. Biol. Chem.* **2015**, *290*, 19489–19495.

(26) Tautermann, C. S. *Bioorg. Med. Chem. Lett.* **2014**, *24*, 4073–4079.

(27) Karakas, E.; Furukawa, H. *Science (80-.).* **2014**, *344*, 992–997.

(28) Sciara, G.; Mancia, F. *Curr. Opin. Struct. Biol.* **2012**, *22*, 476–481.

(29) Monod, J.; Wyman, J.; Changeux, J.-P. *J. Mol. Biol.* **1965**, *12*, 88–118.

(30) Weber, G. *Biochemistry* **1972**, *11*, 864–878.

(31) Cooper, A.; Dryden, D. T. *Eur. Biophys. J.* **1984**, *11*, 103–9.

(32) Jardetzky, O. *Prog. Biophys. Mol. Biol.* **1996**, *65*, 171–219.

(33) Kumar, S.; Ma, B.; Tsai, C.-J.; Sinha, N.; Nussinov, R. *Protein Sci.* **2008**, *9*, 10–19.

(34) Kern, D.; Zuiderweg, E. R. *Curr. Opin. Struct. Biol.* **2003**, *13*, 748–757.

(35) Tsai, C.-J.; del Sol, A.; Nussinov, R. *J. Mol. Biol.* **2008**, *378*, 1–11.

(36) Boehr, D. D.; Nussinov, R.; Wright, P. E. *Nat. Chem. Biol.* **2009**, *5*, 789–96.

(37) Kar, G.; Keskin, O.; Gursoy, A.; Nussinov, R. *Curr. Opin. Pharmacol.* **2010**, *10*, 715–722.

(38) Zhou, H.-X. *Biophys. J.* **2010**, *98*, L15–L17.

(39) Nussinov, R.; Ma, B.; Tsai, C. J. *Biophys. Chem.* **2014**, *186*, 22–30.

(40) Changeux, J.-P. *Annu. Rev. Biophys.* **2012**, *41*, 103–133.

(41) Greives, N.; Zhou, H.-X. *Proc. Natl. Acad. Sci.* **2014**, *111*, 10197–202.

(42) Guo, J.; Pang, X.; Zhou, H.-X. *Structure* **2015**, *23*, 237–247.

(43) Motlagh, H. N.; Wrabl, J. O.; Li, J.; Hilser, V. J. *Nature* **2014**, *508*, 331–339.

(44) Tsai, C.-J.; del Sol, A.; Nussinov, R. *Mol. Biosyst.* **2009**, *5*, 207.

(45) Kornev, A. P.; Taylor, S. S. *Trends Biochem. Sci.* **2015**, *40*, 628–647.

(46) Ma, B.; Tsai, C.-J.; Haliloğlu, T.; Nussinov, R. *Structure* **2011**, *19*, 907–917.

(47) Del Sol, A.; Fujihashi, H.; Amoros, D.; Nussinov, R. *Mol. Syst. Biol.* **2006**, *2*, 2006.0019.

(48) Del Sol, A.; Tsai, C.-J.; Ma, B.; Nussinov, R. *Structure* **2009**, *17*, 1042–1050.

(49) Lockless, S. W. *Science (80-.).* **1999**, *286*, 295–299.

(50) Süel, G. M.; Lockless, S. W.; Wall, M. A.; Ranganathan, R. *Nat. Struct. Biol.* **2003**, *10*, 59–69.

(51) Malmstrom, R. D.; Kornev, A. P.; Taylor, S. S.; Amaro, R. E. *Nat. Commun.* **2015**, *6*, 7588.

(52) Pontiggia, F.; Pachov, D.; Clarkson, M.; Villali, J.; Hagan, M.; Pande, V.; Kern, D. *Nat. Commun.* **2015**, *6*, 7284.

(53) Hebeisen, P.; Haap, W.; Kuhn, B.; Mohr, P.; Wessel, H. P.; Zutter, U.; Kirchner, S.; Ruf, A.; Benz, J.; Joseph, C.; Alvarez-Sánchez, R.; Gubler, M.; Schott, B.; Benardeau, A.; Tozzo, E.; Kitas, E. *Bioorg. Med. Chem. Lett.* **2011**, *21*, 3237–3242.

(54) Zarzycki, M.; Kołodziejczyk, R.; Maciaszczyk-Dziubinska, E.; Wysocki, R.; Jaskolski, M.; Dzugaj, A. *Acta Crystallogr. Sect. D Biol. Crystallogr.* **2011**, *67*, 1028–1034.

(55) Nussinov, R.; Tsai, C.-J.; Csermely, P. *Trends Pharmacol. Sci.* **2011**, *32*, 686–693.

(56) Gilmartin, A. G.; Bleam, M. R.; Groy, A.; Moss, K. G.; Minthorn, E. A.; Kulkarni, S. G.; Rominger, C. M.; Erskine, S.; Fisher, K. E.; Yang, J.; Zappacosta, F.; Annan, R.; Sutton, D.; Laquerre, S. G. *Clin. Cancer Res.* **2011**, *17*, 989–1000.

(57) Sebolt-Leopold, J. S.; Dudley, D. T.; Herrera, R.; Van Becelaere, K.; Wiland, a.; Gowan, R. C.; Tecle, H.; Barrett, S. D.; Bridges, a.; Przybranowski, S.; Leopold, W. R.; Saltiel, a. R. *Nat. Med.* **1999**, *5*, 810–816.

(58) Ohren, J. F.; Chen, H.; Pavlovsky, A.; Whitehead, C.; Zhang, E.; Kuffa, P.; Yan, C.; McConnell, P.; Spessard, C.; Banotai, C.; Mueller, W. T.; Delaney, A.; Omer, C.; Sebolt-Leopold, J.; Dudley, D. T.; Leung, I. K.; Flamme, C.; Warmus, J.; Kaufman, M.; Barrett, S.; Tecle, H.; Hasemann, C. a. *Nat. Struct. Mol. Biol.* **2004**, *11*, 1192–1197.

(59) Zhao, Y.; Adjei, A. A. *Nat. Publ. Gr.* **2014**, *11*, 385–400.

(60) Hackos, D. H.; Lupardus, P. J.; Grand, T.; Sheng, M.; Zhou, Q.; Hanson, J. E.; Hackos, D. H.; Lupardus, P. J.; Grand, T.; Chen, Y.; Wang, T.-m.; Reynen, P. *Neuron* **2016**, *89*, 1–17.

(61) Talavera, D.; Lovell, S. C.; Whelan, S. *Mol. Biol. Evol.* **2015**, *32*, 2456–2468.

(62) Fodor, A. A.; Aldrich, R. W. *J. Biol. Chem.* **2004**, *279*, 19046–19050.

(63) Fodor, A. A.; Aldrich, R. W. *Proteins Struct. Funct. Bioinforma.* **2004**, *56*, 211–221.

(64) Johansson, F.; Toh, H. *J. Bioinform. Comput. Biol.* **2010**, *8*, 809–23.

(65) De Juan, D.; Pazos, F.; Valencia, A. *Nat. Rev. Genet.* **2013**, *14*, 249–261.

(66) Shannon, C. E. *Bell Syst. Tech. J.* **1948**, *27*, 379–423.

(67) Shenkin, P. S.; Erman, B.; Mastrandrea, L. D. *Proteins Struct. Funct. Genet.* **1991**, *11*, 297–313.

(68) Vinga, S. *Brief. Bioinform.* **2014**, *15*, 376–389.

(69) Kullback, S.; Leibler, R. A. *Ann. Math. Stat.* **1951**, *22*, 79–86.

(70) Zhang, S.-W.; Zhang, Y.-L.; Pan, Q.; Cheng, Y.-M.; Chou, K.-C. *Amino Acids* **2008**, *35*, 495–501.

(71) Nussinov, R.; Tsai, C.-J. *Cell* **2013**, *153*, 293–305.

(72) Lichtarge, O.; Bourne, H. R.; Cohen, F. E. *J. Mol. Biol.* **1996**, *257*, 342–58.

(73) Madabushi, S.; Gross, A. K.; Philippi, A.; Meng, E. C.; Wensel, T. G.; Lichtarge, O. *J. Biol. Chem.* **2004**, *279*, 8126–8132.

(74) Raviscioni, M.; He, Q.; Salicru, E. M.; Smith, C. L.; Lichtarge, O. *Proteins Struct. Funct. Bioinforma.* **2006**, *64*, 1046–1057.

(75) Rodriguez, G. J.; Yao, R.; Lichtarge, O.; Wensel, T. G. *Proc. Natl. Acad. Sci.* **2010**, *107*, 7787–7792.

(76) Madabushi, S.; Gross, A. K.; Philippi, A.; Meng, E. C.; Wensel, T. G.; Lichtarge, O. *J. Biol. Chem.* **2004**, *279*, 8126–8132.

(77) Mihalek, I.; Reš, I.; Lichtarge, O. *J. Mol. Biol.* **2004**, *336*, 1265–1282.

(78) Mihalek, I.; Res, I.; Lichtarge, O. *Bioinformatics* **2006**, *22*, 1656–1657.

(79) Merkl, R.; Zwick, M. *BMC Bioinformatics* **2008**, *9*, 151.

(80) Janda, J.-O.; Popal, A.; Bauer, J.; Busch, M.; Klocke, M.; Spitzer, W.; Keller, J.; Merkl, R. *BMC Bioinformatics* **2014**, *15*, 118.

(81) Pelé, J.; Moreau, M.; Abdi, H.; Rodien, P.; Castel, H.; Chabbert, M. *Proteins Struct. Funct. Bioinforma.* **2014**, *82*, 2141–2156.

(82) Brown, C. A.; Brown, K. S. *PLoS One* **2010**, *5*, ed. by Rattray, M., e10779.

(83) Simonetti, F. L.; Teppa, E.; Chernomoretz, A.; Nielsen, M.; Marino Buslje, C. *Nucleic Acids Res.* **2013**, *41*, W8–W14.

(84) Horner, D. S.; Pirovano, W.; Pesole, G. *Brief. Bioinform.* **2007**, *9*, 46–56.

(85) Gao, H.; Dou, Y.; Yang, J.; Wang, J. *BMC Bioinformatics* **2011**, *12*, 206.

(86) Dunn, S.; Wahl, L.; Gloor, G. *Bioinformatics* **2008**, *24*, 333–340.

(87) Wollenberg, K. R.; Atchley, W. R. *Proc. Natl. Acad. Sci.* **2000**, *97*, 3288–3291.

(88) Buslje, C. M.; Santos, J.; Delfino, J. M.; Nielsen, M. *Bioinformatics* **2009**, *25*, 1125–1131.

(89) Clark, G. W.; Ackerman, S. H.; Tillier, E. R.; Gatti, D. L. *BMC Bioinformatics* **2014**, *15*, 157.

(90) Martin, L. C.; Gloor, G. B.; Dunn, S. D.; Wahl, L. M. *Bioinformatics* **2005**, *21*, 4116–4124.

(91) Crooks, G. E.; Wolfe, J.; Brenner, S. E. *Proteins Struct. Funct. Genet.* **2004**, *57*, 804–810.

(92) Olmea, O.; Valencia, A. *Fold. Des.* **1997**, *2*, S25–S32.

(93) McLachlan, A. *J. Mol. Biol.* **1971**, *61*, 409–424.

(94) Halperin, I.; Wolfson, H.; Nussinov, R. *Proteins Struct. Funct. Bioinforma.* **2006**, *63*, 832–845.

(95) Kass, I.; Horovitz, A. *Proteins Struct. Funct. Genet.* **2002**, *48*, 611–617.

(96) Ackerman, S. H.; Tillier, E. R.; Gatti, D. L. *PLoS One* **2012**, *7*, ed. by Tramontano, A., e47108.

(97) Ranganathan, R.; Rivoire, O. *Note 109: A summary of SCA calculations*; tech. rep.; 2012, pp 1–11.

(98) Halabi, N.; Rivoire, O.; Leibler, S.; Ranganathan, R. *Cell* **2009**, *138*, 774–786.

(99) Reynolds, K. A.; McLaughlin, R. N.; Ranganathan, R. *Cell* **2011**, *147*, 1564–1575.

(100) Russ, W. P.; Lowery, D. M.; Mishra, P.; Yaffe, M. B.; Ranganathan, R. *Nature* **2005**, *437*, 579–583.

(101) Socolich, M.; Lockless, S. W.; Russ, W. P.; Lee, H.; Gardner, K. H.; Ranganathan, R. *Nature* **2005**, *437*, 512–518.

(102) Baths, V.; Roy, U. *J. Biomed. Res.* **2011**, *25*, 165–169.

(103) McLaughlin Jr, R. N.; Poelwijk, F. J.; Raman, A.; Gosal, W. S.; Ranganathan, R. *Nature* **2012**, *491*, 138–142.

(104) Peterson, F. C.; Penkert, R. R.; Volkman, B. F.; Prehoda, K. E. *Mol. Cell* **2004**, *13*, 665–676.

(105) Hatley, M. E.; Lockless, S. W.; Gibson, S. K.; Gilman, A. G.; Ranganathan, R. *Proc. Natl. Acad. Sci.* **2003**, *100*, 14445–14450.

(106) Shulman, A. I.; Larson, C.; Mangelsdorf, D. J.; Ranganathan, R. *Cell* **2004**, *116*, 417–429.

(107) Smock, R. G.; Rivoire, O.; Russ, W. P.; Swain, J. F.; Leibler, S.; Ranganathan, R.; Gierasch, L. M. *Mol. Syst. Biol.* **2010**, *6*, 414.

(108) Chi, C. N.; Elfstrom, L.; Shi, Y.; Snall, T.; Engstrom, A.; Jemth, P. *Proc. Natl. Acad. Sci.* **2008**, *105*, 4679–4684.

(109) Fuentes, E. J.; Der, C. J.; Lee, A. L. *J. Mol. Biol.* **2004**, *335*, 1105–15.

(110) Dima, R. I.; Thirumalai, D. *Bioinformatics* **2004**, *20*, 2345–2354.

(111) Dima, R. I. *Protein Sci.* **2006**, *15*, 258–268.

(112) O'Donovan, C. *Brief. Bioinform.* **2002**, *3*, 275–284.

(113) Boeckmann, B. *Nucleic Acids Res.* **2003**, *31*, 365–370.

(114) Getz, G.; Levine, E.; Domany, E. *Proc. Natl. Acad. Sci.* **2000**, *97*, 12079–12084.

(115) Dekker, J. P.; Fodor, A.; Aldrich, R. W.; Yellen, G. *Bioinformatics* **2004**, *20*, 1565–1572.

(116) Yip, K. Y.; Patel, P.; Kim, P. M.; Engelman, D. M.; McDermott, D.; Gerstein, M. *Bioinformatics* **2008**, *24*, 290–292.

(117) Kozma, D.; Simon, I.; Tusnady, G. E. *Nucleic Acids Res.* **2012**, *40*, W329–W333.

(118) Weigt, M.; White, R. A.; Szurmant, H.; Hoch, J. A.; Hwa, T. *Proc. Natl. Acad. Sci.* **2009**, *106*, 67–72.

(119) Morcos, F.; Pagnani, A.; Lunt, B.; Bertolino, A.; Marks, D. S.; Sander, C.; Zecchina, R.; Onuchic, J. N.; Hwa, T.; Weigt, M. *Proc. Natl. Acad. Sci.* **2011**, *108*, E1293–E1301.

(120) Schug, A.; Weigt, M.; Onuchic, J. N.; Hwa, T.; Szurmant, H. *Proc. Natl. Acad. Sci.* **2009**, *106*, 22124–22129.

(121) Sulkowska, J. I.; Morcos, F.; Weigt, M.; Hwa, T.; Onuchic, J. N. *Proc. Natl. Acad. Sci.* **2012**, *109*, 10340–10345.

(122) Szurmant, H.; Hoch, J. A. *Mol. Microbiol.* **2013**, *87*, 707–712.

(123) Procaccini, A.; Lunt, B.; Szurmant, H.; Hwa, T.; Weigt, M. *PLoS One* **2011**, *6*, ed. by Rattray, M., e19729.

(124) Morcos, F.; Jana, B.; Hwa, T.; Onuchic, J. N. *Proc. Natl. Acad. Sci.* **2013**, *110*, 20533–20538.

(125) Jones, D. T.; Buchan, D. W. A.; Cozzetto, D.; Pontil, M. *Bioinformatics* **2012**, *28*, 184–190.

(126) Eckmann, J.-P.; Kamphorst, S. O.; Ruelle, D. *Europhys. Lett.* **1987**, *4*, 973–977.

(127) Zbilut, J. P.; Giuliani, A.; Webber, C. L.; Colosimo, A. *Protein Eng. Des. Sel.* **1998**, *11*, 87–93.

(128) Namboodiri, S.; Verma, C.; Dhar, P. K.; Giuliani, A.; Nair, A. S. *Syst. Synth. Biol.* **2010**, *4*, 271–280.

(129) Porrello, A.; Soddu, S.; Zbilut, J. P.; Crescenzi, M.; Giuliani, A. *Proteins Struct. Funct. Bioinforma.* **2004**, *55*, 743–755.

(130) Giuliani, A.; Tomasi, M. *Proteins Struct. Funct. Genet.* **2002**, *46*, 171–176.

(131) Colafranceschi, M.; Colosimo, A.; Zbilut, J. P.; Uversky, V. N.; Giuliani, A. *J. Chem. Inf. Model.* **2005**, *45*, 183–189.

(132) Namboodiri, S.; Giuliani, A.; Nair, A. S.; Dhar, P. K. *J. Theor. Biol.* **2012**, *304*, 211–8.

(133) Zbilut, J. P.; Sirabella, P.; Giuliani, A.; Manetti, C.; Colosimo, A.; Webber, C. L. *Cell Biochem. Biophys.* **2002**, *36*, 67–88.

(134) Rivoire, O. *Phys. Rev. Lett.* **2013**, *110*, 178102.

(135) Mao, W.; Kaya, C.; Dutta, A.; Horovitz, A.; Bahar, I. *Bioinformatics* **2015**, *31*, 1929–1937.

(136) Bakan, A.; Meireles, L. M.; Bahar, I. *Bioinformatics* **2011**, *27*, 1575–1577.

(137) Göbel, U.; Sander, C.; Schneider, R.; Valencia, A. *Proteins Struct. Funct. Genet.* **1994**, *18*, 309–317.

(138) Fares, M. A. *Genetics* **2006**, *173*, 9–23.

(139) Fares, M. A.; McNally, D. *Bioinformatics* **2006**, *22*, 2821–2822.

(140) Iserte, J.; Simonetti, F. L.; Zea, D. J.; Teppa, E.; Marino-Buslje, C. *Nucleic Acids Res.* **2015**, *43*, W320–W325.

(141) Jeong, C.-S.; Kim, D. *Protein Eng. Des. Sel.* **2012**, *25*, 705–713.

(142) Berezin, C.; Glaser, F.; Rosenberg, J.; Paz, I.; Pupko, T.; Fariselli, P.; Casadio, R.; Ben-Tal, N. *Bioinformatics* **2004**, *20*, 1322–1324.

(143) Glaser, F.; Pupko, T.; Paz, I.; Bell, R. E.; Bechor-Shental, D.; Martz, E.; Ben-Tal, N. *Bioinformatics* **2003**, *19*, 163–164.

(144) Landau, M.; Mayrose, I.; Rosenberg, Y.; Glaser, F.; Martz, E.; Pupko, T.; Ben-Tal, N. *Nucleic Acids Res.* **2005**, *33*, W299–W302.

(145) Armon, A.; Graur, D.; Ben-Tal, N. *J. Mol. Biol.* **2001**, *307*, 447–63.

(146) Ashkenazy, H.; Erez, E.; Martz, E.; Pupko, T.; Ben-Tal, N. *Nucleic Acids Res.* **2010**, *38*, W529–W533.

(147) Celniker, G.; Nimrod, G.; Ashkenazy, H.; Glaser, F.; Martz, E.; Mayrose, I.; Pupko, T.; Ben-Tal, N. *Isr. J. Chem.* **2013**, *53*, 199–206.

(148) Liu, Y.; Gierasch, L. M.; Bahar, I. *PLoS Comput. Biol.* **2010**, *6*, ed. by Rost, B., e1000931.

(149) Liu, Y.; Bahar, I. *Mol. Biol. Evol.* **2012**, *29*, 2253–2263.

(150) Skjærven, L.; Yao, X.-Q.; Scarabelli, G.; Grant, B. J. *BMC Bioinformatics* **2014**, *15*, 399.

(151) Grant, B. J.; Rodrigues, A. P. C.; ElSawy, K. M.; McCammon, J. A.; Caves, L. S. D. *Bioinformatics* **2006**, *22*, 2695–2696.

(152) Li, Z.; Huang, Y.; Ouyang, Y.; Jiao, Y.; Xing, H.; Liao, L.; Jiang, S.; Shao, Y.; Ma, L. *Bioinformatics* **2014**, *30*, 2073–2075.

(153) Rhee, S.-Y.; Liu, T. F.; Holmes, S. P.; Shafer, R. W. *PLoS Comput. Biol.* **2007**, *3*, e87.

(154) Hurst, L. D. *Trends Genet.* **2002**, *18*, 486–487.

(155) Alder, B. J.; Wainwright, T. E. *J. Chem. Phys.* **1959**, *31*, 459.

(156) Rahman, A. *Phys. Rev.* **1964**, *136*, A405–A411.

(157) Karplus, M.; McCammon, J. A. *Nat. Struct. Biol.* **2002**, *9*, 646–652.

(158) Adcock, S. A.; McCammon, J. A. *Chem. Rev.* **2006**, *106*, 1589–615.

(159) Hamelberg, D.; Mongan, J.; McCammon, J. A. *J. Chem. Phys.* **2004**, *120*, 11919.

(160) Wang, Y.; Harrison, C. B.; Schulten, K.; McCammon, J. A. *Comput. Sci. Discov.* **2011**, *4*, 015002.

(161) Miao, Y.; Feher, V. A.; McCammon, J. A. *J. Chem. Theory Comput.* **2015**, *11*, 3584–3595.

(162) Miao, Y.; Feixas, F.; Eun, C.; McCammon, J. A. *J. Comput. Chem.* **2015**, *36*, 1536–1549.

(163) Pierce, L. C.; Salomon-Ferrer, R.; Augusto F. de Oliveira, C.; McCammon, J. A.; Walker, R. C. *J. Chem. Theory Comput.* **2012**, *8*, 2997–3002.

(164) Lindert, S.; Bucher, D.; Eastman, P.; Pande, V.; McCammon, J. A. *J. Chem. Theory Comput.* **2013**, *9*, 4684–4691.

(165) Amaro, R. E.; Baron, R.; McCammon, J. A. *J. Comput. Aided. Mol. Des.* **2008**, *22*, 693–705.

(166) Amaro, R.; Li, W. *Curr. Top. Med. Chem.* **2010**, *10*, 3–13.

(167) Durrant, J. D.; McCammon, J. A. *BMC Biol.* **2011**, *9*, 71.

(168) E. Nichols, S.; V. Swift, R.; E. Amaro, R. *Curr. Top. Med. Chem.* **2012**, *12*, 2002–2012.

(169) Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; Teller, E. *J. Chem. Phys.* **1953**, *21*, 1087.

(170) Hastings, W. K. *Biometrika* **1970**, *57*, 97–109.

(171) Landau, D. P.; Binder, K., *A Guide to Monte Carlo Simulations in Statistical Physics*; Cambridge University Press: Cambridge, 2009.

(172) Zwier, M. C.; Chong, L. T. *Curr. Opin. Pharmacol.* **2010**, *10*, 745–752.

(173) Noé, F. *Biophys. J.* **2015**, *108*, 228–229.

(174) Sethi, A.; Eargle, J.; Black, A. A.; Luthey-Schulten, Z. *Proc. Natl. Acad. Sci.* **2009**, *106*, 6620–6625.

(175) Gasper, P. M.; Fuglestad, B.; Komives, E. A.; Markwick, P. R. L.; McCammon, J. A. *Proc. Natl. Acad. Sci.* **2012**, *109*, 21216–22.

(176) Freddolino, P. L.; Gardner, K. H.; Schulten, K. *Photochem. Photobiol. Sci.* **2013**, *12*, 1158.

(177) Yan, W.; Zhou, J.; Sun, M.; Chen, J.; Hu, G.; Shen, B. *Amino Acids* **2014**, *46*, 1419–1439.

(178) VanWart, A. T.; Eargle, J.; Luthey-Schulten, Z.; Amaro, R. E. *J. Chem. Theory Comput.* **2012**, *8*, 2949–2961.

(179) Ghosh, A.; Vishveshwara, S. *Proc. Natl. Acad. Sci.* **2007**, *104*, 15711–15716.

(180) Collier, G.; Ortiz, V. *Arch. Biochem. Biophys.* **2013**, *538*, 6–15.

(181) McClendon, C. L.; Friedland, G.; Mobley, D. L.; Amirkhani, H.; Jacobson, M. P. *J. Chem. Theory Comput.* **2009**, *5*, 2486–2502.

(182) McClendon, C. L.; Hua, L.; Barreiro, G.; Jacobson, M. P. *J. Chem. Theory Comput.* **2012**, *8*, 2115–2126.

(183) Bradley, M. J.; Chivers, P. T.; Baker, N. A. *J. Mol. Biol.* **2008**, *378*, 1155–1173.

(184) Atilgan, C.; Okan, O. B.; Atilgan, A. R. *Annu. Rev. Biophys.* **2012**, *41*, 205–225.

(185) Van Wart, A. T.; Durrant, J.; Votapka, L.; Amaro, R. E. *J. Chem. Theory Comput.* **2014**, *10*, 511–517.

(186) LeVine, M. V.; Weinstein, H. *PLoS Comput. Biol.* **2014**, *10*, ed. by Schlessinger, A., e1003603.

(187) Rivalta, I.; Sultan, M. M.; Lee, N.-S.; Manley, G. A.; Loria, J. P.; Batista, V. S. *Proc. Natl. Acad. Sci.* **2012**, *109*, E1428–(1) Rivalta, I., Sultan, M. M., Lee, N.–S.

(188) Miao, Y.; Nichols, S. E.; Gasper, P. M.; Metzger, V. T.; McCammon, J. A. *Proc. Natl. Acad. Sci.* **2013**, *110*, 10982–7.

(189) Blacklock, K.; Verkhivker, G. M. *PLoS Comput. Biol.* **2014**, *10*, e1003679.

(190) Frembgen-Kesner, T.; Elcock, A. H. *J. Mol. Biol.* **2006**, *359*, 202–214.

(191) Bowman, G. R.; Geissler, P. L. *Proc. Natl. Acad. Sci.* **2012**, *109*, 11681–11686.

(192) Wassman, C. D.; Baronio, R.; Demir, Ö.; Wallentine, B. D.; Chen, C.-K.; Hall, L. V.; Salehi, F.; Lin, D.-W.; Chung, B. P.; Hatfield, G. W.; Richard Chamberlin, A.; Luecke, H.; Lathrop, R. H.; Kaiser, P.; Amaro, R. E. *Nat. Commun.* **2013**, *4*, 1407.

(193) Nair, P. C.; Miners, J. O. *Silico Pharmacol.* **2014**, *2*, 4.

(194) Bowman, G. R.; Bolin, E. R.; Hart, K. M.; Maguire, B. C.; Marqusee, S. *Proc. Natl. Acad. Sci.* **2015**, *112*, 2734–2739.

(195) Durrant, J. D.; De Oliveira, C. A. F.; McCammon, J. A. *J. Mol. Graph. Model.* **2011**, *29*, 773–776.

(196) Chovancova, E.; Pavelka, A.; Benes, P.; Strnad, O.; Brezovsky, J.; Kozlikova, B.; Gora, A.; Sustr, V.; Klvana, M.; Medek, P.; Biedermannova, L.; Sochor, J.; Damborsky, J. *PLoS Comput. Biol.* **2012**, *8*, ed. by Prlic, A., e1002708.

(197) Eyrisch, S.; Helms, V. *J. Med. Chem.* **2007**, *50*, 3457–3464.

(198) Brady, G. P.; Stouten, P. F. *J. Comput. Aided. Mol. Des.* **2000**, *14*, 383–401.

(199) Le Guilloux, V.; Schmidtke, P.; Tuffery, P. *BMC Bioinformatics* **2009**, *10*, 168.

(200) Schmidtke, P.; Bidon-Chanal, A.; Luque, F. J.; Barril, X. *Bioinformatics* **2011**, *27*, 3276–3285.

(201) Halgren, T. *Chem. Biol. Drug Des.* **2007**, *69*, 146–148.

(202) Halgren, T. A. *J. Chem. Inf. Model.* **2009**, *49*, 377–389.

(203) Brenke, R.; Kozakov, D.; Chuang, G.-Y.; Beglov, D.; Hall, D.; Landon, M. R.; Mattos, C.; Vajda, S. *Bioinformatics* **2009**, *25*, 621–627.

(204) Votapka, L.; Amaro, R. E. *Bioinformatics* **2013**, *29*, 393–394.

(205) Zheng, X.; Gan, L.; Wang, E.; Wang, J. *AAPS J.* **2013**, *15*, 228–241.

(206) Edelsbrunner, H.; Mücke, E. P. *ACM Trans. Graph.* **1994**, *13*, 43–72.

(207) Kawabata, T.; Go, N. *Proteins Struct. Funct. Bioinforma.* **2007**, *68*, 516–529.

(208) Yu, J.; Zhou, Y.; Tanaka, I.; Yao, M. *Bioinformatics* **2010**, *26*, 46–52.

(209) Laskowski, R. A. *J. Mol. Graph.* **1995**, *13*, 323–330.

(210) Peters, K. P.; Fauck, J.; Frömmel, C. *J. Mol. Biol.* **1996**, *256*, 201–213.

(211) Liang, J.; Woodward, C.; Edelsbrunner, H. *Protein Sci.* **1998**, *7*, 1884–1897.

(212) Binkowski, T. A. *Nucleic Acids Res.* **2003**, *31*, 3352–3355.

(213) Edelsbrunner, H.; Kirkpatrick, D.; Seidel, R. *IEEE Trans. Inf. Theory* **1983**, *29*, 551–559.

(214) Kawabata, T. *Proteins Struct. Funct. Bioinforma.* **2010**, *78*, 1195–1211.

(215) Hendlich, M.; Rippmann, F.; Barnickel, G. *J. Mol. Graph. Model.* **1997**, *15*, 359–63, 389.

(216) Huang, B.; Schroeder, M. *BMC Struct. Biol.* **2006**, *6*, 19.

(217) Levitt, D. G.; Banaszak, L. J. *J. Mol. Graph.* **1992**, *10*, 229–34.

(218) Weisel, M.; Proschak, E.; Schneider, G. *Chem. Cent. J.* **2007**, *1*, 7.

(219) An, J. *Mol. Cell. Proteomics* **2005**, *4*, 752–761.

(220) Laurie, A. T. R.; Jackson, R. M. *Bioinformatics* **2005**, *21*, 1908–1916.

(221) Durrant, J. D.; Votapka, L.; Sørensen, J.; Amaro, R. E. *J. Chem. Theory Comput.* **2014**, *10*, 5047–5056.

(222) Wass, M. N.; Kelley, L. A.; Sternberg, M. J. E. *Nucleic Acids Res.* **2010**, *38*, W469–W473.

(223) Brylinski, M.; Skolnick, J. *Proc. Natl. Acad. Sci.* **2008**, *105*, 129–134.

(224) Kufareva, I.; Ilatovskiy, A. V.; Abagyan, R. *Nucleic Acids Res.* **2012**, *40*, D535–D540.

(225) Fischer, J. D.; Mayer, C. E.; Soding, J. *Bioinformatics* **2008**, *24*, 613–620.

(226) Lichtarge, O.; Sowa, M. E. *Curr. Opin. Struct. Biol.* **2002**, *12*, 21–27.

(227) Capra, J. A.; Laskowski, R. A.; Thornton, J. M.; Singh, M.; Funkhouser, T. A. *PLoS Comput. Biol.* **2009**, *5*, ed. by Lengauer, T., e1000585.

(228) Hernandez, M.; Ghersi, D.; Sanchez, R. *Nucleic Acids Res.* **2009**, *37*, W413–W416.

(229) Kozakov, D.; Hall, D. R.; Chuang, G.-Y.; Cencic, R.; Brenke, R.; Grove, L. E.; Beglov, D.; Pelletier, J.; Whitty, A.; Vajda, S. *Proc. Natl. Acad. Sci.* **2011**, *108*, 13528–13533.

(230) Kozakov, D.; Grove, L. E.; Hall, D. R.; Bohnuud, T.; Mottarella, S. E.; Luo, L.; Xia, B.; Beglov, D.; Vajda, S. *Nat. Protoc.* **2015**, *10*, 733–755.

(231) Goodford, P. J. *J. Med. Chem.* **1985**, *28*, 849–857.

(232) Miranker, A.; Karplus, M. *Proteins* **1991**, *11*, 29–34.

(233) Allen, K. N.; Bellamacina, C. R.; Ding, X.; Jeffery, C. J.; Mattos, C.; Petsko, G. A.; Ringe, D. *J. Phys. Chem.* **1996**, *100*, 2605–2611.

(234) Mattos, C.; Ringe, D. *Nat. Biotechnol.* **1996**, *14*, 595–9.

(235) Ngan, C. H.; Bohnuud, T.; Mottarella, S. E.; Beglov, D.; Villar, E. A.; Hall, D. R.; Kozakov, D.; Vajda, S. *Nucleic Acids Res.* **2012**, *40*, W271–W275.

(236) Ivetac, A.; Andrew McCammon, J. *Chem. Biol. Drug Des.* **2010**, *76*, no–no.

(237) Ivetac, A.; McCammon, J. A. In *Methods Mol. Biol.* 2012; Vol. 819, pp 3–12.

(238) Pande, V. S.; Beauchamp, K.; Bowman, G. R. *Methods* **2010**, *52*, 99–105.

(239) Schwantes, C. R.; McGibbon, R. T.; Pande, V. S. *J. Chem. Phys.* **2014**, *141*, 090901.

(240) Chodera, J. D.; Noé, F. *Curr. Opin. Struct. Biol.* **2014**, *25*, 135–144.

(241) Malmstrom, R. D.; Lee, C. T.; Van Wart, A. T.; Amaro, R. E. *J. Chem. Theory Comput.* **2014**, *10*, 2648–2657.

(242) Shukla, D.; Hernández, C. X.; Weber, J. K.; Pande, V. S. *Acc. Chem. Res.* **2015**, *48*, 414–422.

(243) Beauchamp, K. A.; Bowman, G. R.; Lane, T. J.; Maibaum, L.; Haque, I. S.; Pande, V. S. *J. Chem. Theory Comput.* **2011**, *7*, 3412–3419.

(244) Prinz, J.-H.; Wu, H.; Sarich, M.; Keller, B.; Senne, M.; Held, M.; Chodera, J. D.; Schütte, C.; Noe, F. *J. Chem. Phys.* **2011**, *134*, 174105.

(245) Senne, M.; Trendelkamp-Schroer, B.; Mey, A. S.; Schütte, C.; Noé, F. *J. Chem. Theory Comput.* **2012**, *8*, 2223–2238.

(246) Kohlhoff, K. J.; Shukla, D.; Lawrenz, M.; Bowman, G. R.; Konerding, D. E.; Belov, D.; Altman, R. B.; Pande, V. S. *Nat. Chem.* **2013**, *6*, 15–21.

(247) Prinz, J.-H.; Keller, B.; Noé, F. *Phys. Chem. Chem. Phys.* **2011**, *13*, 16912.

(248) Cronkite-Ratcliff, B.; Pande, V. *Bioinformatics* **2013**, *29*, 950–952.

(249) Lane, T. J.; Bowman, G. R.; Beauchamp, K.; Voelz, V. a.; Pande, V. S. *J. Am. Chem. Soc.* **2011**, *133*, 18413–18419.

(250) Bowman, G. R.; Ensign, D. L.; Pande, V. S. *J. Chem. Theory Comput.* **2010**, *6*, 787–794.

(251) Lapidus, L. J.; Acharya, S.; Schwantes, C. R.; Wu, L.; Shukla, D.; King, M.; DeCamp, S. J.; Pande, V. S. *Biophys. J.* **2014**, *107*, 947–955.

(252) Li, S.; Xiong, B.; Xu, Y.; Lu, T.; Luo, X.; Luo, C.; Shen, J.; Chen, K.; Zheng, M.; Jiang, H. *J. Chem. Theory Comput.* **2014**, *10*, 2255–2264.

(253) Schor, M.; Mey, A. S. J. S.; Noé, F.; MacPhee, C. E. *J. Phys. Chem. Lett.* **2015**, *6*, 1076–1081.

(254) Voelz, V. A.; Bowman, G. R.; Beauchamp, K.; Pande, V. S. *J. Am. Chem. Soc.* **2010**, *132*, 1526–1528.

(255) Buch, I.; Giorgino, T.; De Fabritiis, G. *Proc. Natl. Acad. Sci.* **2011**, *108*, 10184–10189.

(256) Tiwary, P.; Limongelli, V.; Salvalaglio, M.; Parrinello, M. *Proc. Natl. Acad. Sci.* **2015**, *112*, E386–E391.

(257) Vanatta, D. K.; Shukla, D.; Lawrenz, M.; Pande, V. S. *Nat. Commun.* **2015**, *6*, 7283.

(258) Shukla, D.; Meng, Y.; Roux, B.; Pande, V. S. *Nat. Commun.* **2014**, *5*, DOI: 10.1038/ncomms4397.

(259) Bryngelson, J. D.; Wolynes, P. G. *Proc. Natl. Acad. Sci.* **1987**, *84*, 7524–7528.

(260) Wolynes, P.; Onuchic, J.; Thirumalai, D. *Science (80-.).* **1995**, *267*, 1619–1620.

(261) Onuchic, J. N.; Socci, N. D.; Luthey-Schulten, Z.; Wolynes, P. G. *Fold. Des.* **1996**, *1*, 441–450.

(262) Onuchic, J. N.; Luthey-Schulten, Z.; Wolynes, P. G. *Annu. Rev. Phys. Chem.* **1997**, *48*, 545–600.

(263) Wolynes, P. G. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **2005**, *363*, 453–467.

(264) Karplus, M. *Nat. Chem. Biol.* **2011**, *7*, 401–4.

(265) Ferreiro, D. U.; Hegler, J. A.; Komives, E. A.; Wolynes, P. G. *Proc. Natl. Acad. Sci.* **2007**, *104*, 19819–24.

(266) Ferreiro, D. U.; Hegler, J. A.; Komives, E. A.; Wolynes, P. G. *Proc. Natl. Acad. Sci.* **2011**, *108*, 3499–3503.

(267) Jenik, M.; Parra, R. G.; Radusky, L. G.; Turjanski, A.; Wolynes, P. G.; Ferreiro, D. U. *Nucleic Acids Res.* **2012**, *40*, W348–51.

(268) Gō, N. *Adv. Biophys.* **1984**, *18*, 149–164.

(269) Sali, A.; Blundell, T. L. *J. Mol. Biol.* **1993**, *234*, 779–815.

(270) Marti-Renom, M. A.; Stuart, A. C.; Fiser, A.; Sánchez, R.; Melo, F.; Šali, A. *Annu. Rev. Biophys. Biomol. Struct.* **2000**, *29*, 291–325.

(271) Fiser, A.; Do, R. K. G.; Šali, A. *Protein Sci.* **2000**, *9*, 1753–1773.

(272) Eswar, N.; Webb, B.; Marti-Renom, M. A.; Madhusudhan, M.; Eramian, D.; Shen, M.-Y.; Pieper, U.; Sali, A. In *Curr. Protoc. Protein Sci.* John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2007; Vol. Chapter 2, Unit 2.9.

(273) Weinkam, P.; Pons, J.; Sali, A. *Proc. Natl. Acad. Sci.* **2012**, *109*, 4875–80.

(274) Bahar, I.; Rader, A. *Curr. Opin. Struct. Biol.* **2005**, *15*, 586–592.

(275) Zheng, W.; Brooks, B. R.; Thirumalai, D. *Proc. Natl. Acad. Sci.* **2006**, *103*, 7664–7669.

(276) Zheng, W.; Brooks, B.; Thirumalai, D. *Curr. Protein Pept. Sci.* **2009**, *10*, 128–132.

(277) Ma, J. *Structure* **2005**, *13*, 373–380.

(278) Ming, D.; Wall, M. E. *Proteins* **2005**, *59*, 697–707.

(279) Tirion, M. M. *Phys. Rev. Lett.* **1996**, *77*, 1905–1908.

(280) Bahar, I.; Atilgan, A. R.; Erman, B. *Fold. Des.* **1997**, *2*, 173–181.

(281) Haliloglu, T.; Bahar, I.; Erman, B. *Phys. Rev. Lett.* **1997**, *79*, 3090–3093.

(282) Ming, D.; Wall, M. E. *Phys. Rev. Lett.* **2005**, *95*, 198103.

(283) Su, J. G.; Qi, L. S.; Li, C. H.; Zhu, Y. Y.; Du, H. J.; Hou, Y. X.; Hao, R.; Wang, J. H. *Phys. Rev. E* **2014**, *90*, 022719.

(284) Berezovsky, I. N. *Biochim. Biophys. Acta - Proteins Proteomics* **2013**, *1834*, 830–835.

(285) Mitternacht, S.; Berezovsky, I. N. *Protein Eng. Des. Sel.* **2011**, *24*, 405–409.

(286) Mitternacht, S.; Berezovsky, I. N. *PLoS Comput. Biol.* **2011**, *7*, ed. by Lengauer, T., e1002148.

(287) Goncearenco, A.; Mitternacht, S.; Yong, T.; Eisenhaber, B.; Eisenhaber, F.; Berezovsky, I. N. *Nucleic Acids Res.* **2013**, *41*, W266–W272.

Chapter 7

Structural Basis for Ligand Modulation of the CCR2 Conformational Landscape

7.1 Abstract

CC Chemokine Receptor 2 (CCR2) is a part of the chemokine receptor family, an important class of therapeutic targets. These class A G-protein coupled receptors (GPCRs) are involved in mammalian signaling pathways and control cell migration toward endogenous CC chemokine ligands, named for the adjacent cysteine motif on their N-terminus. Chemokine receptors and their associated ligands are involved in a wide range of diseases and thus have become important drug targets. CCR2, in particular, promotes the metastasis of cancer cells and is also implicated in autoimmunity driven type-1 diabetes, diabetic nephropathy, multiple sclerosis, asthma, atherosclerosis, neuropathic pain, and rheumatoid arthritis. Although promising, CCR2 antagonists have been largely unsuccessful to date. Here, we investigate the effect of an orthosteric and an allosteric antagonist on CCR2 dynamics by coupling long timescale molecular dynamics simulations with Markov-state model theory. We find that the antagonists shift CCR2 into several stable inactive conformations that are distinct from the crystal structure conformation and disrupt a continuous internal water and sodium ion pathway, preventing transitions to an active-like state. Several metastable conformations present a cryptic drug binding pocket near the allosteric site that may be

amenable to targeting with small molecules. Without antagonists, the apo dynamics reveal intermediate conformations along the activation pathway that provide insight into the basal dynamics of CCR2, and may also be useful for future drug design.

7.2 Introduction

The signaling axis of CCR2 and its endogenous ligand, CCL2, is a notable therapeutic target due to its association with numerous diseases, including cancer, autoimmunity driven type-1 diabetes, diabetic nephropathy, multiple sclerosis, asthma, atherosclerosis, neuropathic pain, and rheumatoid arthritis[1–3]. Despite much effort that has been devoted to clinical and pre-clinical trials, a successful antagonist has yet to be developed[4–7]. Prior to a full-length crystal structure of CCR2, several studies used homology modeling and docking to gain insights into the structure and dynamics of the protein and its associated ligands or small molecule drugs[8–10]. However, recently CCR2 was crystallized for the first time[11], opening up new opportunities for rational drug design.

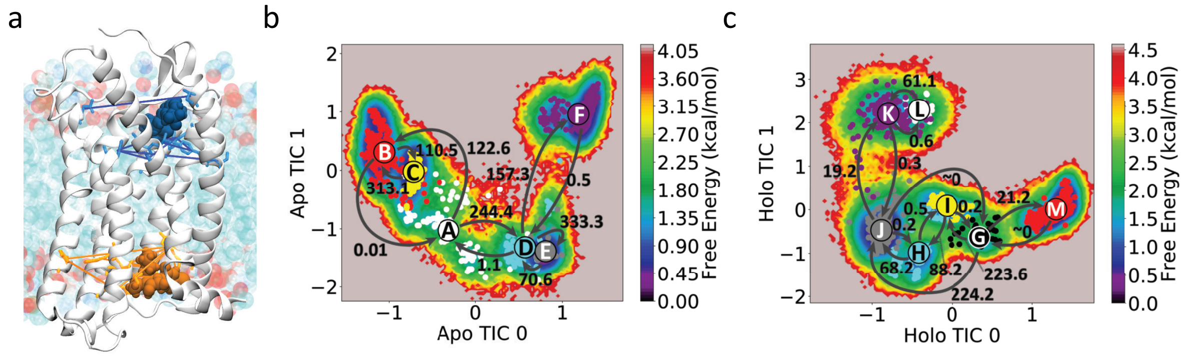


Figure 7.1: MD simulations of CCR2 in a lipid bilayer were performed apo and holo CCR2. A) Sets of residue pairs surrounding the two ligand binding pockets were used with TICA (SI Appendix). The protein is shown in white cartoon. Lipids are teal, red, and blue. The orthosteric and allosteric ligands are shown in blue and orange, respectively, with inter-residue pair distances denoted by similarly colored lines. The free energy and Maximum-Likelihood HMMs of B) apo and C) holo CCR2, projected onto the first two TICA components. Coarse-grained states are labeled and colored. Transition rates between macrostates are represented by arrows reported in units of ms^{-1}.

As with most GPCRs, chemokine receptors transmit signals across cell membranes by means of extracellular ligand and intracellular G-protein binding. Distinct conformational states of the receptor are necessary for chemokine/ligand binding, G-protein binding, activation, inactivation, and signal transmission

[12–14]. GPCRs are no longer considered to be simple on/off molecular switches – instead, they assume a wide range of conformational states, including ligand-specific states, intermediate states, and states that allow for basal (apo) signaling without ligands bound [13, 15–21]. Ligands and small molecule drugs may shift the equilibrium of the receptor's conformational states towards particular states. Effective small molecule antagonists that inhibit CCR2 signaling, potentially by shifting the receptor equilibrium toward inactive conformational states, are desired for treatment of diseases that involve the CCR2/CCL2 axis. Key challenges are to characterize the basal dynamics of CCR2 and to understand how current antagonistic small molecule drugs modulate these dynamics. While crystal structures provide valuable snapshots of proteins and protein complexes, they lack the ability to reveal dynamics at the atomic level. Starting with the newly resolved crystal structure of CCR2 (PDB ID: 5T1A) we performed multi-microsecond all-atom explicitly solvated molecular dynamics (MD) simulations of the receptor in a lipid bilayer in unbound (apo) and dual-antagonist-bound (holo) states (Fig. 7.1). The two antagonists were co-crystallized with CCR2: the orthosteric antagonist, BMS-681, and the allosteric antagonist, CCR2-RA-[R]. Each system was simulated in triplicate on Anton 2 [22] for a total of 260 microseconds (Table 7.1 and Fig. 7.5).

While long timescale (tens of microseconds) simulations are useful for analyzing sequential conformational changes, simulations are generally unable to directly probe timescales of biological interest (milliseconds - seconds)[23]. One way to bridge this timescale gap is to couple MD simulations with Markov state model theory[24–32] (MSM, described in SI Materials and Methods). Integrating MD simulations with MSMs allowed us to extend the reach of simulated timescales, and identify key differences in the conformational ensembles and dominant slow motions of apo and holo CCR2 (Fig. 7.1). We find that the antagonists disrupt a continuous internal water and sodium ion pathway preventing transitions to an active-like state, and shift CCR2 into several stable states that are distinct from the crystal structure conformation, three of which present a cryptic druggable pocket. Without antagonists, intermediate conformations with active-state conformational signatures shed light on the apo dynamics of CCR2 and may also be useful for future drug design.

7.3 Results and Discussion

To compare the conformational landscapes of apo and holo CCR2 we ran all-atom MD simulations totalling 260 μs on Anton 2[22]. For one MD replicate of the holo system, we observed the orthosteric drug dissociate from the protein. Analyzing the conformations before and after ligand dissociation yields a first glimpse of the allosteric effect of the remaining antagonist on the protein dynamics, and provides a starting point for future rounds of adaptive sampling to obtain robust dissociation statistics (not pursued here). In order to extend the analysis beyond a dissociation event and connect to longer timescale phenomena, MSMs were constructed from the trajectories (Fig. 7.1B,C). The variational approach for conformation dynamics[33], specifically Time-structure-based independent component analysis (TICA)[34, 35] was used to perform dimensionality reduction for the MSMs and identify the features and collective variables (time-structure based independent components (TICs)) that best represent the dominant slow motions. The MSMs create human interpretable models that we use to interrogate the conformational and kinetic differences between the two ensembles to derive new understandings about the mechanisms underlying effects of CCR2 antagonism. Further methodological details are provided in Methods and in the SI Appendix.

7.3.1 Comparison of the CCR2 conformational ensemble with other class A GPCRs

We compare representative states from the apo and holo conformational ensemble with other class A GPCRs to establish similarities within the class. We find that the states of apo CCR2 have conformational signatures found in the active or intermediately-active states of GPCRs, suggesting that these states are on a pathway toward activation. Holo CCR2 diverges from the crystal structure to form distinct states that expose putative drug binding pockets and reveal the effect of antagonists on receptor dynamics. The most populated holo macrostate, J, is not representative of the crystal structure as it deviates 10.8 Å from the crystal structure conformation (Fig. 7.1C).

We evaluate the metastable states by comparing helical conformational signatures and conserved groups of structurally neighboring amino acids called 'microswitches'. These include NPxxY (Tyr 305$^{7.53}$), DRY (Arg 138$^{3.50}$), Tyr 222$^{5.58}$, sets of residues in the orthesteric and allosteric binding sites, and the chemokine and G-protein binding pockets to the inactive crystal structure of CCR2 that we used in this study (PDB ID: 5T1A), to an intermediately-active crystal structure of a class A GPCR, A_{2A}AR(PDB ID: 2YDO[36]; 25% sequence identity to CCR2), the active crystal structure of a class A GPCR, US28 (PDB ID: 4XT3[37]; 30% sequence identity to CCR2), and three other chemokine receptors: CCR5 (PDB ID: 4MBS[38], CCR9 (PDB ID: 5LWE[39]), and CXCR4 (PDB ID: 4RWS[40]). Signatures of an active GPCR state include: 1) the inward shift of the intracellular part of helix VII toward the helical bundle, 2) the outward shift of the intracellular part of helix VI in concert with helix V, 3) the upward shift and lateral movement of helix III, and 4) the rearrangements of conserved microswitches [15]. According to these metrics, the starting crystal structure of CCR2 is in an inactive conformation[11], the crystal structure of A_{2A}AR is an an intermediately-active conformation, and the crystal structure of US28 is in the active conformation.

1) Apo macrostates show an active-like inward shift of the intracellular part of helix VII toward the helical bundle

All of the apo macrostates exhibit an active state hallmark (Fig. 7.2A): the intracellular end of helix VII tilts slightly inward toward the center of the helical bundle. More prominently, the extracellular end of helix VII tilts outward, resembling the active conformation of US28. The holo macrostates show the opposite: the intracellular end of helix VII tilts slightly outward and the extracellular end of helix VII tilts inward, remaining in the crystal structure conformation.

2) Holo macrostates, not apo, show an active-like outward shift of the intracellular part of helix VI in concert with helix V

Helix V and VI in the apo macrostates are not in an active conformation. Instead, it is the holo macrostates that have the intracellular end of helix V and VI tilting outward to resemble the active conformation (Fig. 7.2C,D), suggesting that neither apo nor holo macrostates are in an exclusively inactive or active conformational state, despite starting from a particularly inactive crystal structure. Due to this outward motion of helix VI, holo macrostates K and L exhibit a more open G-protein binding site compared to holo macrostate G which is more closed (Fig. 7.1B,C). The RMSF of the allosteric ligand is larger in

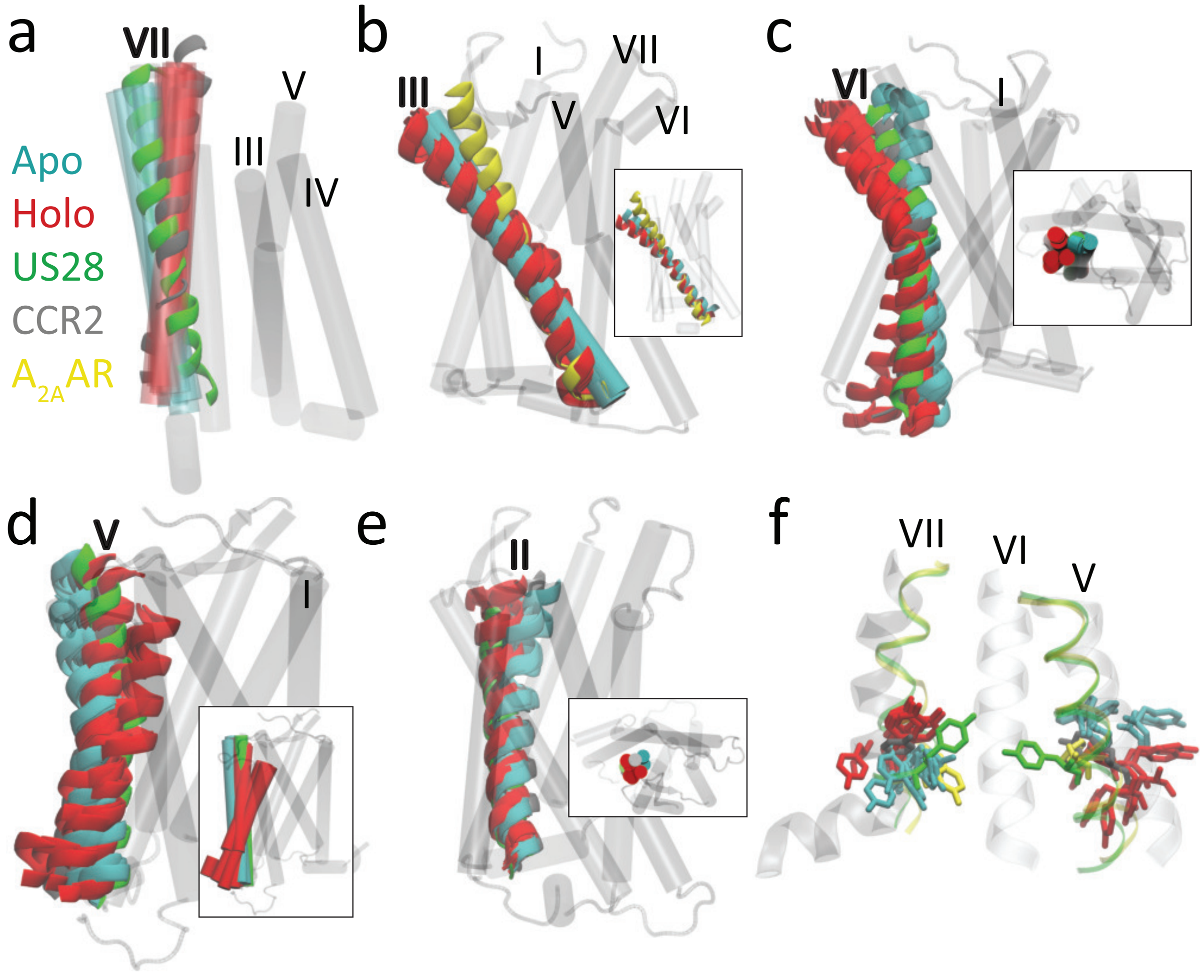


Figure 7.2: Apo and holo macrostates are compared to the active crystal structure of US28 (green, PDB ID 4XT3) or the active crystal structure of $A_{2A}AR$ (yellow, PDB ID 2YDO) and the inactive crystal structure of CCR2 (grey, PDB ID 5T1A). A) Helix VII of apo macrostates resemble the active conformation of US28; holo macrostates resemble the CCR2 crystal structure. B) The conformation of helix III in apo macrostates subtly resemble active $A_{2A}AR$; holo macrostates are tilted away from the center of the helical bundle. C,D) Helix V and VI of apo macrostates straighten or tilt in toward the center of the protein, similar to the active conformation of US28 and the inactive CCR2 crystal structure; helix V and VI of several holo macrostates tilt away from the binding sites, accessing more active-like conformations than the apo macrostates or even the active state of US28. E) Helix II in the apo macrostates shifts inward; in the holo macrostates it shifts outward. F) In licorice are conserved motifs TYR $305^{7.53}$ and TYR $222^{5.58}$. All six apo metastable state assume a new conformation for TYR $305^{7.53}$, pointing intracellulary and in a similar conformation to active $A_{2A}AR$. Six out of the seven holo metastable states have TYR $305^{7.53}$ in the same conformation as the equilibrated crystal structure. Post-ligand-dissociation holo state L assumes a new position of TYR $305^{7.53}$, more similar to the dominant apo conformation. Apo metastable states sample a narrower range of conformations for TYR $222^{5.58}$ than holo.

macrostates K and L, indicating that the inactive (inward) conformation of helix VI may play a critical role in stabilization of the allosteric ligand (Fig. 7.6).

3) Apo macrostates show an active-like upward shift and lateral movement of helix III

Apo macrostates also resemble the active conformation by the slight upward shift of helix III; unlike holo macrostates, which remain in a position similar to the inactive crystal structure (Fig. 7.2B).

4) The rearrangements of conserved microswitches suggest that apo macrostates resemble active states, and holo macrostates resemble inactive states

a) NPxxY motif (Tyr 305$^{7.53}$). In the inactive conformation of GPCRs, Tyr 305$^{7.53}$ points towards helices I, II, or VIII (in CCR2, it points toward II), and in the active state Tyr 305$^{7.53}$ points toward middle axis of helical bundle[15]. Each apo macrostate shows Tyr 305$^{7.53}$ pointing downward into the intracellular (G-protein) binding pocket (Fig. 7.2F). This positioning of Tyr 305$^{7.53}$ matches the intermediately-active conformation of A_{2A}AR, which also points down. It does not match the active conformation in US28 that points up, and is also distinct from the inactive crystal structure of CCR2. In six out of the seven the holo macrostates, Tyr 305$^{7.53}$ is stabilized in the inactive state and matches the inactive CCR2 crystal structure conformation as well as the inactive chemokine receptor crystal structures of CCR5 and CCR9.

The holo macrostate in which Tyr 305$^{7.53}$ is not stabilized in the inactive conformation is accessed after the orthosteric ligand dissociates (State L, Fig. 7.1C, 7.2F); the allosteric pocket residues rearrange and Tyr 305$^{7.53}$ assumes a downward conformation similar to the apo states and CXCR4. These concerted events may indicate allosteric cross-talk between the chemokine binding site and the G-protein binding site.

b) The microswitch residue Trp 256$^{6.48}$, and the interaction of the DRY motif (Arg 138$^{3.50}$) with Tyr 222$^{5.58}$. Apo and holo macrostates both maintain the same chi angle of the conserved microswitch residue Trp 256$^{6.48}$ which describes an active GPCR when it switches from gauche to trans conformation and facilitates the interaction of Tyr 222$^{5.58}$ and Tyr 305$^{7.53}$. In the CCR2 crystal structure and the crystal structures of chemokine receptors CCR5 and CXCR4, Trp 256$^{6.48}$ points upward and extends into the helical core. Each apo macrostate shows Trp 256$^{6.48}$ in a single conformation pointing toward helix III

(Fig. 7.7). In the holo macrostates, Trp 256$^{6.48}$ access three distinct conformations: one resembling the crystal structure but with the helix shifted slightly outward from the helical core, another that laterally twists toward helix V, and one conformation that points down into the helical core toward the G-protein binding site (Fig. 7.7). This third conformation, in which the orthosteric ligand has dissociated, represented by the holo macrostate L, further suggests cross-talk between the chemokine binding site and the rest of the protein.

The interaction of these two tyrosines and Arg 138$^{3.50}$ also characterizes an active state GPCR [41]. In the inactive crystal structure of CCR2, Tyr 222$^{5.58}$ points toward lipids, sterically blocked by Phe 246$^{6.38}$ from interaction with Arg 138$^{3.50}$ and Tyr 305$^{7.53}$ [11]. In apo macrostates, Tyr 222$^{5.58}$ remains pointed toward the lipids, never swiveling around to interact with Arg 138$^{3.50}$ or Tyr 305$^{7.53}$ as occurs in activated GPCR states (Fig. 7.2F). Holo macrostates actually show increased range of motion of Tyr 222$^{5.58}$, diverging from the crystal structure and stabilizing in unique intermediate conformations. The steric obstruction from Phe 246$^{6.38}$ is alleviated in both apo and holo macrostates, as Phe 246$^{6.38}$ swings outward and points toward the lipids. The conformations of these microswitch residues indicate that both apo and holo macrostates are sampling different conformations.

5) Formation of continuous water pathway suggests movement of apo towards activation

Internal water molecules, which may influence conformational changes in GPCRs by interfering with hydrogen bonding networks of the receptor's backbone and side chains, are postulated to be an integral part of receptor activation in GPCRs[42–45]. Work in other GPCRs has additionally shown that activation can allow water and sodium ion flow through GPCR core[46]. Furthermore, it has been shown that the activation of GPCRs is voltage sensitive[47]. Our simulations enable the direct visualization of water and sodium ion density in both CCR2 ensembles.

A continuous internal water pathway forms in apo CCR2 (Fig. 7.3A). The antagonists disrupt this water pathway, slowing the rate of water entry to and egress from the protein core (Fig. 7.3A). An analysis of the water occupancy per residue (Fig. 7.8A) indicates that several of the high water occupancy residues (e.g. Asp 36$^{1.26}$, Ser 50$^{1.40}$, Glu 235$^{6.27}$, Lys 236$^{6.28}$, Glu 310$^{8.48}$, Lys 311$^{8.49}$) may be exposed to more water in the apo simulations than in the holo simulations simply because the ligands have been removed

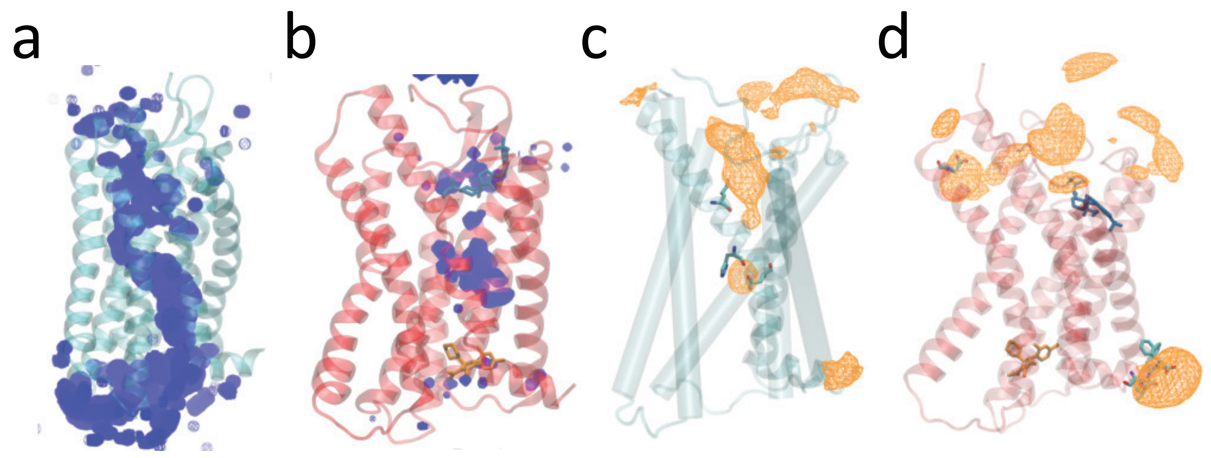


Figure 7.3: Ligands disrupt a continuous internal water and sodium ion pathway. Average water density over a 50 microsecond simulation of A) apo (teal) and B) holo (red). The orthosteric ligand is shown in blue and the allosteric ligand is shown in orange. Total average sodium ion density in C) apo and D) holo. Highest occupancy residues are depicted in cyan licorice and plotted in Fig. 7.8.

and the water has access to the binding pockets. The other residues (e.g., Asp 78$^{2.40}$, Tyr 80$^{2.42}$, Asp 88$^{2.50}$, Leu 92$^{2.54}$, Ile 93$^{2.55}$, Gly 127$^{3.39}$, Ile 128$^{3.40}$, Glu 291$^{7.39}$, and Phe 312$^{8.50}$) reside in the protein core, along the continuous pathway (Fig. 7.3A).

Class A GPCRs possess a conserved sodium binding site at Asp$^{2.50}$ corresponding to Asp 88 in CCR2[48]. The role of sodium is thought to contribute to the mechanism of receptor activation[49–51]. In particular, dynamics of activation were previously hypothesized to impinge upon the sodium binding pocket, eventually leading to ion permeation from the sodium binding site into the cytosol[50]. A sodium ion occupancy per residue analysis (7.3C, Fig. 7.8B) indicates that, while no sodium permeation events into the cytosol were observed in the apo trajectories, ions interact with sodium binding site residues Asp 88$^{2.50}$, Glu 291$^{7.39}$, and His 297$^{7.45}$. In holo CCR2, sodium does not interact with binding site residues, preventing the possibility of a permeation event (7.3D, Fig. 7.8B).

7.4 Effects of antagonist binding on CCR2 dynamics

Comparisons of the apo and holo MSMs elucidate the effects of antagonists on CCR2 dynamics. Notably, apo relaxation timescales are an order of magnitude less than holo relaxation timescales (Table 7.2), indicating that the antagonists greatly perturb CCR2 dynamics. The motions described by apo and holo TIC 0 represent the most striking difference between the two systems' dynamics. In the apo MSM, the inter-residue distances most closely correlated with apo TIC 0 are all a part of the allosteric (G-protein) binding pocket, whereas in the holo MSM, the inter-residue distances most closely correlated with holo TIC 0 are all a part of the orthosteric (chemokine) binding pocket (Fig. 7.9).

Apo TIC 1 represents the flipping of Trp $98^{2.60}$ into the orthosteric drug binding site (Fig. 7.11A,B, 7.9C). In the crystal structure Trp $98^{2.60}$ packs with the tri-substituted cyclohexane of the orthosteric antagonist, BMS-681[11]. Without the presence of this ligand, Trp $98^{2.60}$ assumes three distinct positions. The Trp $98^{2.60}$ conformation in the cluster at the neutral TIC (boxed in yellow, Fig. 7.11A,B) most closely resembles the conformation of Trp $98^{2.60}$ in the active GPCR US28, which is shifted slightly up and in towards the helical core in comparison to the CCR2 crystal structure. The two other conformations are found at the extreme ends of apo TIC 1 in densely populated free energy wells. Of these two conformations, state F assumes the most dramatic conformation and protrudes into the chemokine binding site (Fig. 7.10). In the holo macrostates there is markedly less intrusion into the binding pocket due to the presence of the ligand.

Holo TIC 1 represents the concerted movement of 5 pairs of residues in the orthosteric ligand binding site during orthosteric ligand dissociation (Fig. 7.11C,D, Fig. 7.9D). The separation projected in the first two TICs (Fig. 7.11D) is divided into clusters of frames that occur before (white clusters), during (grey), and after (black) dissociation. The residue pairs identified by TICA that contribute to holo TIC 1 and this ligand dissociation (Fig. 7.9D) were confirmed by analyzing the original simulation data. The key changes are the change in distance between Tyr $49^{1.39}$ - Thr $292^{7.40}$, Trp $98^{2.60}$ - Tyr $120^{3.32}$, Ser $50^{1.40}$ - Tyr $259^{6.51}$, and the chi angle of Glu $291^{7.39}$. Notably, four of these residues (Tyr $49^{1.39}$, Trp $98^{2.60}$, Tyr $120^{3.32}$, and Thr $292^{7.40}$) are not only involved in binding to the co-crystallized orthosteric antagonist BMS-681 and/or CCL2 binding, but are also critical for GPCR activation [52, 53].

The positioning of the orthosteric ligand and the conformation of Trp $98^{2.60}$ are closely linked (Fig. 7.11C, Fig. 7.12). After ligand dissociation, in holo states K and L (purple and white, respectively), Trp $98^{2.60}$ turns towards helix III, bending slightly inward toward the chemokine binding site. Prior to ligand dissociation, Trp $98^{2.60}$ has two distinct conformations. In the first conformation (states I and G, yellow and black), the ligand positions itself between helices I and VII, in the same conformation as the crystal structure. Trp $98^{2.60}$ is constrained in a downward position, pointing intracellularly, also resembling the CCR2 crystal structure conformation and the crystal structure of chemokine receptor CCR5. In the second conformation (States H and J, cyan and grey), Trp $98^{2.60}$ flips up and out of the binding pocket, pointing extracellularly, and the ligand moves between helices I and II. This conformation of Trp $98^{2.60}$ more closely resembles CCR9 (Fig. 7.13). The third conformation of Trp $98^{2.60}$ is found in State M (red), and is the most prominent position of the residue as it extends deeper into the chemokine binding site toward helix III. In this case, the ligand interacts with helices II, IV, and V, and there are no transitions from this state to a dissociated state.

As in the apo MSM, the absence of the orthosteric ligand causes a shift in the position of Trp $98^{2.60}$. In the holo simulations shown in Fig. 7.14, the dissociation event is preceded by a doubling of the distance between Trp $98^{2.60}$ and Tyr $120^{3.32}$, and 3 μs after dissociation the distance returns to its previous 0.4 nm. This increase in distance may be required for the ligand to begin the process of dissociating. Another drastic change during the dissociation event is the switch of Glu $291^{7.39}$ from a constrained chi angle of -50 to -100 degrees to an unconstrained chi angle (Fig.7.15). After dissociation, this angle more closely resembles the conformation in all apo simulations. Glu $291^{7.39}$ is a key mediator of many CCR2 antagonists[54], but there is no direct interaction between Glu $291^{7.39}$ and the orthosteric antagonist in the CCR2 crystal structure[40]. That the conformation of Glu $291^{7.39}$ switches after dissociation suggests that Glu $291^{7.39}$ is involved in ligand stabilization despite not directly interacting with the ligand.

In the CCR2 crystal structure, there is a hydrogen bond between Tyr $49^{1.39}$ and Thr $292^{7.40}$. The gamma-lactam secondary exocyclic amine of the orthosteric ligand forms a hydrogen bond with the hydroxyl of Thr $292^{7.40}$, and the carbonyl oxygen of the gamma-lactam forms a hydrogen bond with Tyr $49^{1.39}$. During simulation, the distance between Tyr $49^{1.39}$ and Thr $292^{7.40}$ remains stable until 3 μs after the ligand dissociates, when it begins fluctuating (Fig. 7.16). This suggests that the orthosteric ligand dissociation breaks the hydrogen bond between these key ligand binding residues. This motion is captured in the holo MSM: the separation of the two residues is exemplified between States H (pre-dissociation) and L (post-dissociation) in Fig. 7.16.

Finally, the faster dominant motions (TICs 2, 3, 4) in the holo MSM consist of rearrangements in the allosteric ligand binding site, suggesting that an allosteric rearrangement must first happen in order for the orthosteric ligand to dissociate. Further evidence for this is the observed correlated motion of the downward flip of the conserved residue Tyr $305^{7.53}$ in the G-protein binding site with the dissociation of the orthosteric ligand from the chemokine binding site.

Overall, holo macrostates show more helical tilting and binding site expansion, which increases the solvent-accessible surface area (SASA) when compared to the crystal structure and the apo macrostates. However, the apo simulations overall have greater residue fluctuation, suggesting that the antagonist ligands dampen CCR2 dynamics (Fig. 7.17).

7.4.1 Opening of a cryptic druggable pocket

A dramatic expansion of the extracellular (chemokine) binding site is exhibited in the holo macrostates. The expansion is caused by a pronounced outward tilting of helix VI and slight outward tilting of helix II in the holo macrostates, whereas the apo macrostates show the opposite, with a slight inward tilting of both helices VI and II (Fig. 7.2C,E). The intracellular (G-protein) binding site also enlarges in the holo macrostates due to the outward shift of the intracellular ends of helices V and VI, but remains obstructed in all apo and holo states. In the crystal structure, this obstruction occurs by the interaction of Arg $138^{3.50}$ with Asp $137^{3.49}$ and with Thr $77^{2.39}$ [11], which are maintained throughout all the simulations. The outward movement of the intracellular end of helix VI and the movement of helix V toward helix VI in states L, J, and H in the holo MSM create a putative site for novel allosteric antagonists; this pocket also transiently appears in the apo simulations (Fig. 7.4). Computational solvent mapping[55] of this novel site indicates that the pocket presents surfaces that are amenable to ligand binding due to its ability to bind clusters of multiple different drug-like probes (Fig. 7.4C). The pocket can be accessed through the lipid bilayer between helices IV and V, or from the G-protein binding site, as a deeper extension of the current allosteric binding site of CCR2-RA-[R], and may be useful for rational drug design or modification of current antagonists.

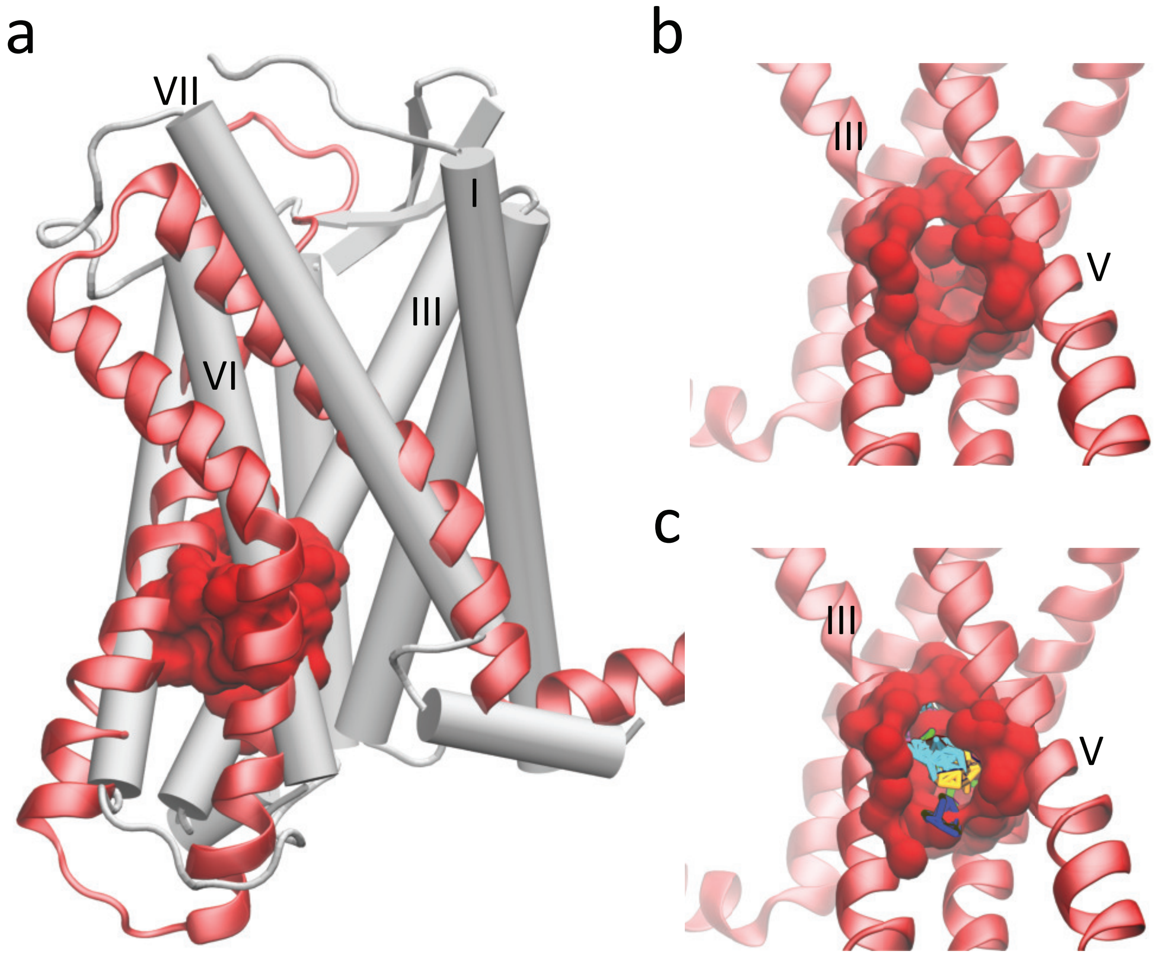


Figure 7.4: A putative allosteric drug binding pocket is revealed by three holo macrostates. A) A comparison of the CCR2 crystal structure (white cartoon) with helices V, VI, and VII (red new cartoon) of one holo macrostate. The pocket is shown in red surface. B) A closer view of the pocket from the other side of the protein, between helices III and V. C) Small organic probes used for computational fragment mapping are multicolored.

7.5 Conclusions

To characterize the basal dynamics of CCR2 and understand how small molecule antagonists modulate these dynamics, we coupled long timescale atomic simulations and MSM theory to compare the metastable states accessed by apo and holo CCR2 in its native membrane-embedded form.

Antagonists perturb CCR2 dynamics and kinetics, and are associated with distinct residue rearrangement and key motions. Several intermediate states reveal a novel cryptic binding site that could be targeted with small molecule inhibitors. In a previous study[56], cryptic pockets predicted with MSM theory have been experimentally confirmed and suggest that this methodology can successfully be used to guide drug discovery efforts.

Without antagonists, CCR2 is able to access other distinct metastable states that are likely sampling along an activation pathway. These intermediate states inform on the basal dynamics of CCR2 and may be useful for modification of previously unsuccessful drugs.

7.6 Abridged Methods

See the SI Appendix for full Materials and Methods. MD trajectories and MSM construction scripts will be available for download.

7.6.1 System Preparation and Molecular Dynamics Simulations

Two systems were simulated for a total of 260 μs: CCR2 holo, with both co-crystallized antagonist ligands bound, and CCR2 apo, without ligands bound. CCR2-RA-[R] and BMS 681[11] were removed to build the apo system. Each all-atom system is embedded in a POPC bilayer, explicitly solvated with TIP3P, and simulated with 150mM NaCl, at pH 7.4, at 310K and 1 bar. The initial coordinates were taken from the experimental crystal structure[11].

7.6.2 Building the Markov State Models

The MSMs were built with PyEMMA version 2.5.4 [57] and selected based on implied timescale plots (Fig. 7.18) and Chapman-Kolmogorov tests (Figs. 7.19 and 7.20) and coarse-grained with hidden

Markov models (HMMs). Representative structures were selected from each macrostate by taking the centroid of the most populated microstate (Figs. 7.21 to 7.23)

7.7 Acknowledgements

Chapter 7, in full, has been submitted for publication as it may appear in “Taylor, B. C.; Lee, C. T.; Amaro, R. E. *Structural Basis for Ligand Modulation of the CCR2 Conformational Landscape.* PNAS.” The dissertation author was an investigator, supervisor, and coauthor of this work.

The authors thank Tracy Handel and Irina Kufareva for their valuable input and initial modification of CCR2 coordinates. We thank Frank Noe and Cecilia Clementi for their useful discussions regarding MSM construction. We also thank the organizers and participants of the 2018 Workshop on Free energy Methods, Kinetics and Markov State Models in Drug Design for helpful input. This work was supported in part by the Director’s New Innovator Award Program NIH DP2-OD007237, the National Biomedical Computation Resource (NBCR) NIH P41-GM103426, and the National Science Foundation through XSEDE supercomputing resources provided via TG-CHE060073 to R.E.A. C.T.L. also acknowledges support from the NIH Molecular Biophysics Training Program (T32-GM008326). Anton 2 computer time was provided by the Pittsburgh Supercomputing Center (PSC) through Grant R01GM116961 from the National Institutes of Health. The Anton 2 machine at PSC was generously made available by D.E. Shaw Research.

B.C.T, C.T.L, and R.E.A. designed research; B.C.T. performed the research; B.C.T. and C.T.L. analyzed the results; and B.C.T., C.T.L, and R.E.A. wrote the paper. The authors declare the following competing financial interest(s): REA has equity interest in, and is a cofounder and scientific advisor of, Actavalon, Inc.

7.8 SI Appendix

7.8.1 Methods

System Preparation

Two systems were simulated: holo CCR2, with both co-crystallized antagonist ligands bound, and apo CCR2, without the ligands bound. Each all-atom system is embedded in a biologically similar POPC bilayer and explicitly solvated with TIP3P. The initial coordinates were taken from the experimental crystal structure[11] and simulated for 50ns MD simulations on local resources before simulation on Anton2. All simulations are in a POPC lipid bilayer and cubic water box with 150mM NaCl, at pH 7.4, at 310K and 1 bar.

The two small molecules CCR2-RA-[R][11, 58] and BMS 681[11, 59] were deleted to build the apo system. Both systems were protonated at pH 7.4 in Maestro-integrated PROPKA. A POPC lipid bilayer was added to each system and solvated with TIP3 waters and 0.15 M NaCl using CHARMM-GUI[60]. The small molecules were parameterized with CGenFF[61]. System coordinates were parameterized with the CHARMM36[62] force field. No restraints were added.

Modification of CCR2 coordinates

The coordinates for CCR2 were taken from the experimental crystal structure [11] and modified to build the apo (unbound) and holo (dual-antagonist-bound) systems. For both systems, the T4 Lysozyme was removed from the crystal structure and intracellular loop 3 and part of the ECL3 and N terminus was constructed. For ICL3, a peptide containing residues 223:243 was built ab-initio, the backbones of residues 223:231 and 236:243 and the side chains of residues 223:226 and 241:243 were tethered to their respective positions, the receptor represented as a set of potential grid maps representing vw, el, hb, and sf "potentials", and the peptide was sampled in these maps. For NT/ECL3, the protocol is similar except that 2 separate peptides are built (31:41 and 276:285), a disulfide bond is imposed between 21 and 277, and the entire thing is sampled as above. There are several zero-occupancy side chains whose conformations are predicted as a part of this simulation.

Best scoring conformations of the two fragments are merged with the rest of receptor coordinates and the system is minimized in its full-atom representation: first by exhaustively sampling polar rotatable hydrogens, then by minimizing the side-chain conformers, then by Monte-Carlo sampling of side-chain conformers, then by minimizing everything. During these steps, harmonic restraints of gradually decreasing strength are imposed between the model and either the X-ray coordinates or the best prediction conformations of the built regions. Towards the end of the optimization, the restraints were released almost entirely and the complex remained stable. This was done in the presence of ligands. Zn ion and water molecules were removed.

Molecular Dynamics Simulations

Both systems were minimized and equilibrated using the GPU version of AMBER12. The systems were minimized at NPT for 15,000 total steps and were equilibrated for 2 sequential 25 nanosecond runs. The systems were then simulated for 50 nanoseconds in the NPT ensemble at 310K and 1 bar with 2-fs time-step and particle mesh Ewald electrostatic approximation. The additional replicates were made from the final production output by simulating for three additional nanoseconds to scramble the input velocities.

MD simulations on Anton2 were performed on the final coordinates from the short GPU-enabled AMBER12 simulations. The Anton 2 simulations were run in the NPT ensemble, using Anton's Nose-Hoover thermostat-barostat, at 310K and 1 bar with a 2.5-fs time-step and particle mesh Ewald electrostatic approximation. The two systems were simulated for an aggregate total of 260 microseconds (Table 7.1, Fig. 7.5).

- Ensemble and Constraints: After minimization, equilibration, and initial production runs, both simulations were run as NPT ensembles using Anton's Multigrator framework. No constraints were used in the simulations.

- Boundaries: The fully solvated systems are cubic with X, Y, Z unit lengths of 72 Å, 72 Å, and 103 Å respectively.

- Force Fields: CHARMM36 force field with TIP3P waters; small molecule ligands were parameterized with CGenFF.

- Atom Count and Types: Each system contains 55,000 atoms. The systems are composed of the protein, POPC lipids, water, small molecule drugs, Na^+, and Cl^-.

Trajectory Preparation

MD trajectories were processed using VMD[63]. All frames were aligned to the first frame using all residues of the protein. The frame rate for trajectories was 240ps, the standard for Anton2 simulations. The trajectories were converted into NAMD's .dcd trajectory format for analysis with PyEMMA[57], MSMBuilder[64], and in-house scripts.

Markov State Models

A Markov State Model (MSM) is a time-dependent master equation that describes the probability of transitioning between discrete states at a fixed time interval. These models are required to have the Markovian property (i.e., the probability of transitioning between discrete states is independent of previous transitions). By clustering protein structures extracted from an MD trajectory, discrete conformational states can be identified for use in MSMs[31, 64–66]. Transitions between conformational clusters observed over the course of an MD trajectory are tallied, and the MSM is then built from the transition probabilities between these distinct clusters. MSM/MD analysis provides access to the thermodynamic, kinetic, and structural characteristics of the protein conformational ensemble (i.e., a robust description of the free-energy landscape of the protein)[31, 64–68]. The thermodynamics of the various conformational states can be calculated from the equilibrium distribution. It is also possible to resolve the transition kinetics between individual states, the concerted or principal protein motions, metastable states, and the transition pathways between discrete states[64, 66, 67]. Lastly, the source molecular dynamics simulations provide representative cluster structures for use in structure-based drug design[68].

Building the Markov-State Models

We used time-structure independent component analysis (TICA)[34, 35, 69] starting with all pairwise inter-residue distances to perform dimensionality reduction and identify the features and collective variables (time-structure based independent components (TICs)) that best represent the dominant slow motions in the apo and holo simulations. To reduce the number of pairwise distances to a manageable number, we employed an iterative TICA feature selection approach. First, TICA was run on a curated starting set of hundreds of distances between transmembrane helices. Features with low TIC correlation were then removed from the set and TICA run on the resultant new basis. In this iterative fashion, we selected 22 representative features, listed below. These resultant features are the 22 sets of distances between residues in the orthosteric and allosteric ligand-binding pockets (Fig. 1A).

The apo and holo systems were clustered separately using K-means. Two MSMs were built: one MSM was built on the apo data and a second MSM was built on the holo data. In each case, the data was projected separately into TICA space and the trajectory frames were clustered using K-means clustering implemented in PyEMMA[57]. The MSMs were selected based on implied timescale plots (Fig. 7.18) and the Chapman-Kolmogorov test[31] was used to test the consistency between the MSMs and the MD simulations (Fig. 7.19, 7.20). The apo MSM has a lag time of 14.4 nanoseconds and 665 clusters; and the holo MSM has a lag time of 48 nanoseconds and 790 clusters. The MSMs were coarse-grained using hidden Markov models (HMMs)[70] to identify metastable macrostates and transitions between those states. The apo MSM is coarse-grained into six macrostates; the holo MSM into seven macrostates (Fig. 1B,C). A representative structure for each macrostate was selected by taking the centroid of the most populated microstate, using MSMBuilder[64] (Fig. 7.21, 7.22, 7.23).

Set of 22 Features:

- Ile 40, Asn 199
- Tyr 222, Arg 138
- Tyr 305, Arg 138
- Tyr 305, Tyr 222
- Tyr 49, Thr 292
- Tyr 120, Glu 291
- Tyr 120, Tyr 259

- Glu 291, Tyr 259
- Tyr 49, Trp 98
- Trp 98, Tyr 120
- Trp 98, Glu 291
- Trp 98, Thr 292
- Tyr 49, Tyr 259
- Phe 246, Leu 81
- Ile 245, Leu 134
- Ile 245, Leu 81
- Val 244, Tyr 305
- Tyr 305, Leu 81
- Tyr 305, Val 63
- Tyr 305, Leu 134
- Tyr 305, Leu 67
- Leu 67, Val 244

Table 7.1: Description and setup of simulated CCR2 systems.

	Apo			Holo		
Simulation Number	1	2	3	1	2	3
Simulation Time in microseconds	50	50	10	50	50	50
Ligands Bound	None			BMS-681 and CCR2-RA-[R]		
Number of Atoms	53,097			53,077		
Membrane Lipids	POPC					
Water Model	TIP3P					
Force Field	CHARMM36 FF					
Box Dimensions	72 Å, 72 Å,103 Å					

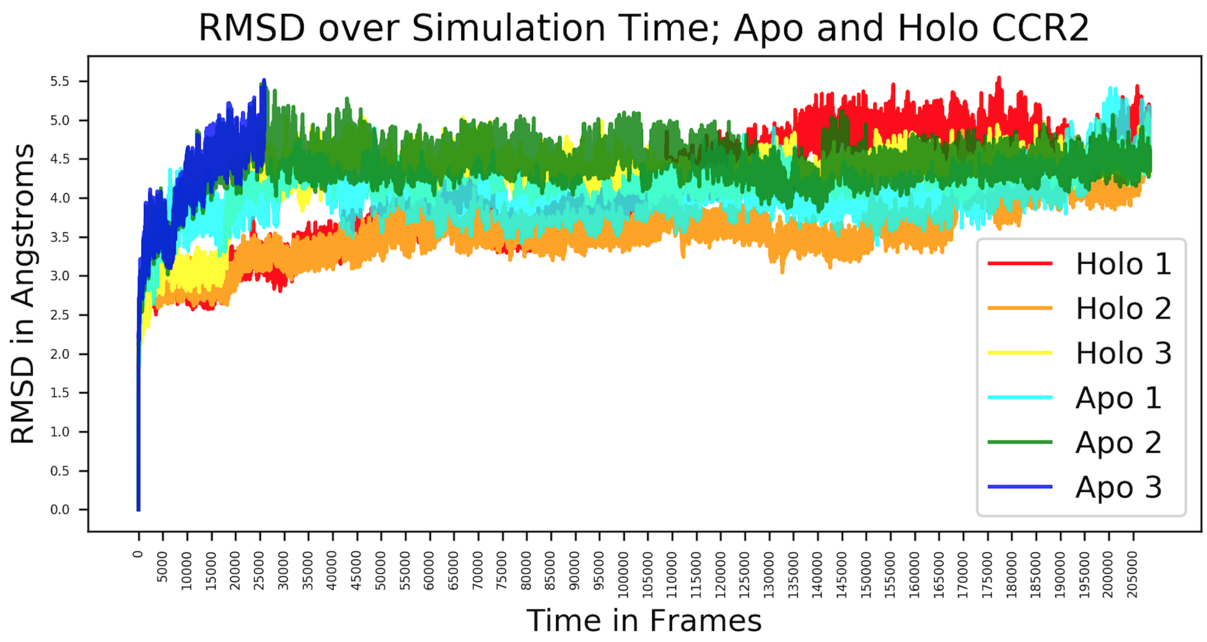


Figure 7.5: RMSD plot of all trajectories over simulation time.

Table 7.2: Relaxation timescales for the apo and holo MSMs. Units are in microseconds.

Apo	321.6	57.7	11.1	9.0	4.2	2.8	2.4	1.7	1.3	1.3
Holo	2246.9	286.8	84.9	59.1	44.3	21.6	8.2	7.7	4.4	3.9

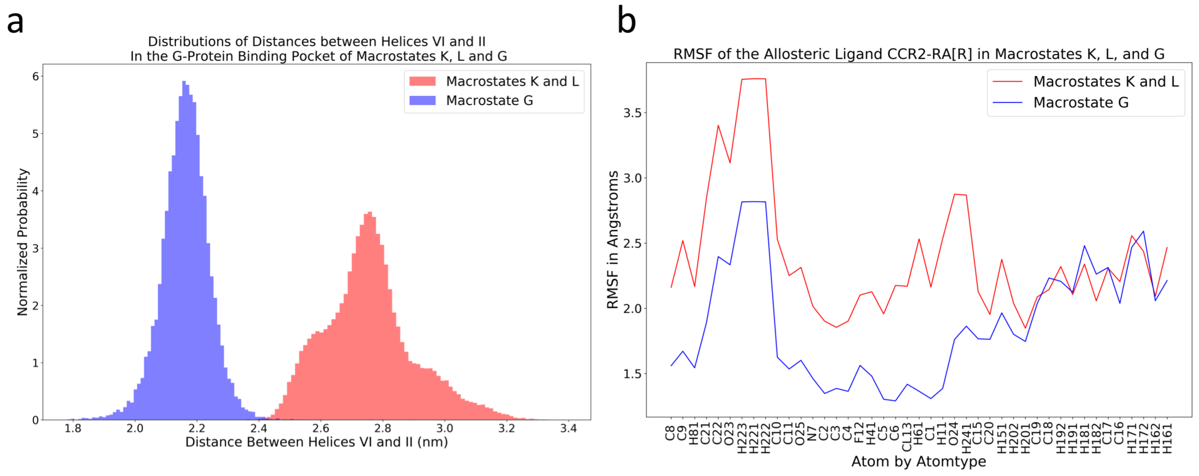


Figure 7.6: The RMSF of the allosteric ligand is larger in macrostates with more open G-protein binding sites. A) Distribution of distances in macrostates K, L and G between residues Gly 224 and Asp 78 in the intracellular regions of helices VI and II. B) RMSF of the allosteric ligand in holo macrostates K, L, and G.

a

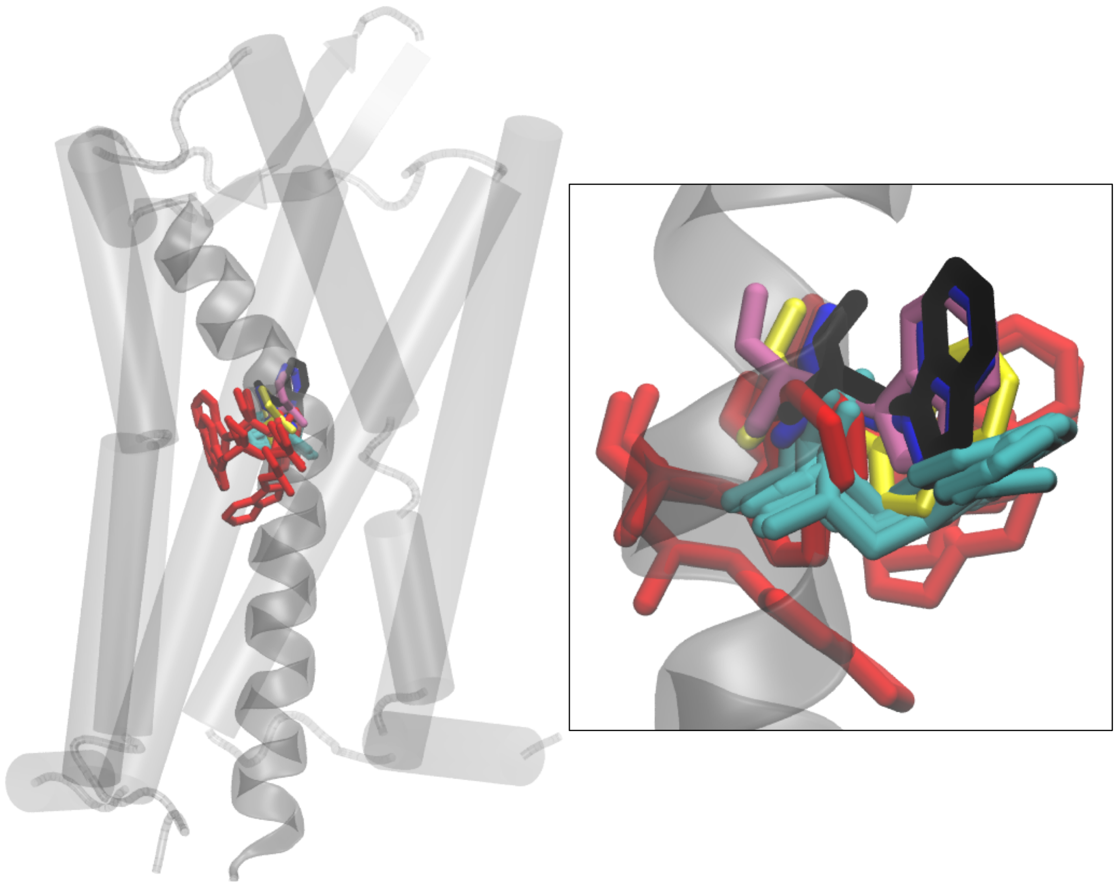
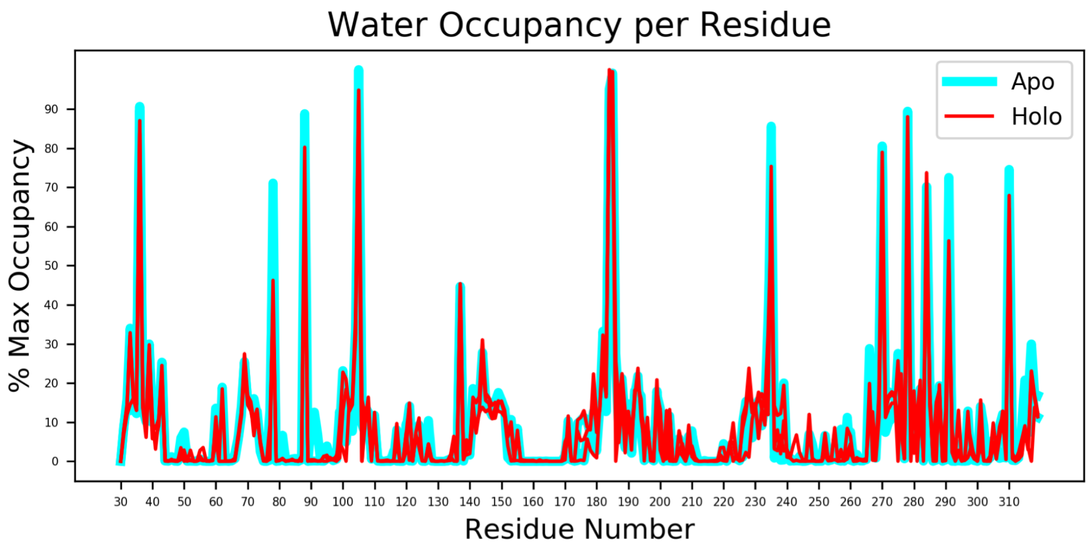


Figure 7.7: Trp 256$^{6.48}$ compared to the apo and holo macrostates and the crystal structures of CCR2 (PDB ID: 5T1A; grey, black) CCR5 (PDB ID: 4MBS; blue) and CXCR4 (PDB ID: 4RWS; mauve). A) Each apo macrostate shows Trp 256$^{6.48}$ in a single conformation pointing toward helix III. In the holo macrostates, Trp 256$^{6.48}$ access three distinct conformations: one resembling the crystal structure but with the helix shifted slightly outward from the helical core, another that laterally twists toward helix V, and one conformation that points down into the helical core toward the G-protein binding site.

a



b

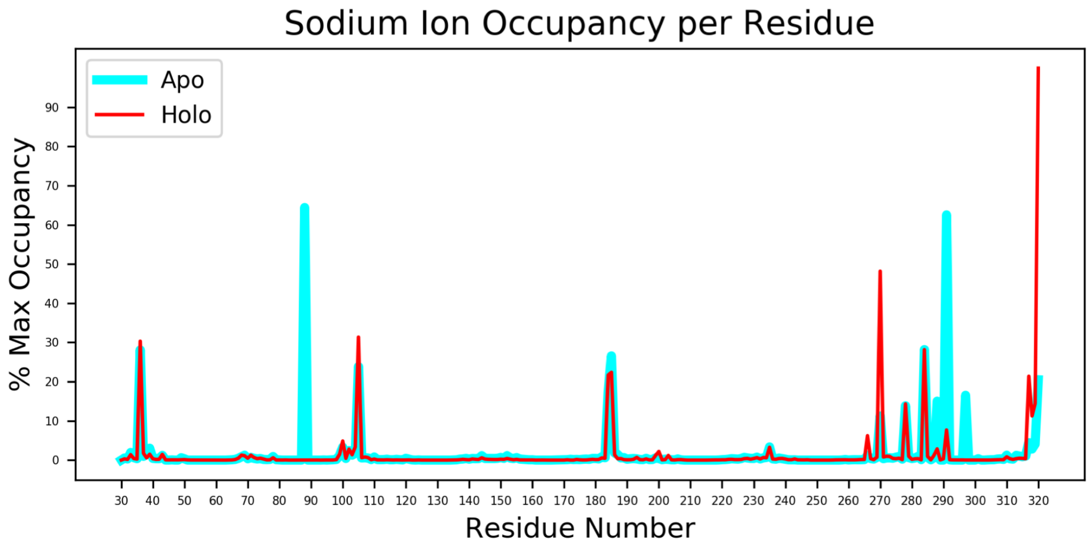


Figure 7.8: Ligands disrupt a continuous internal water and sodium ion pathway. A) Water occupancy per residue in all apo (teal) and holo (red) simulations. B) Sodium ion occupancy per residue in all apo and holo simulations.

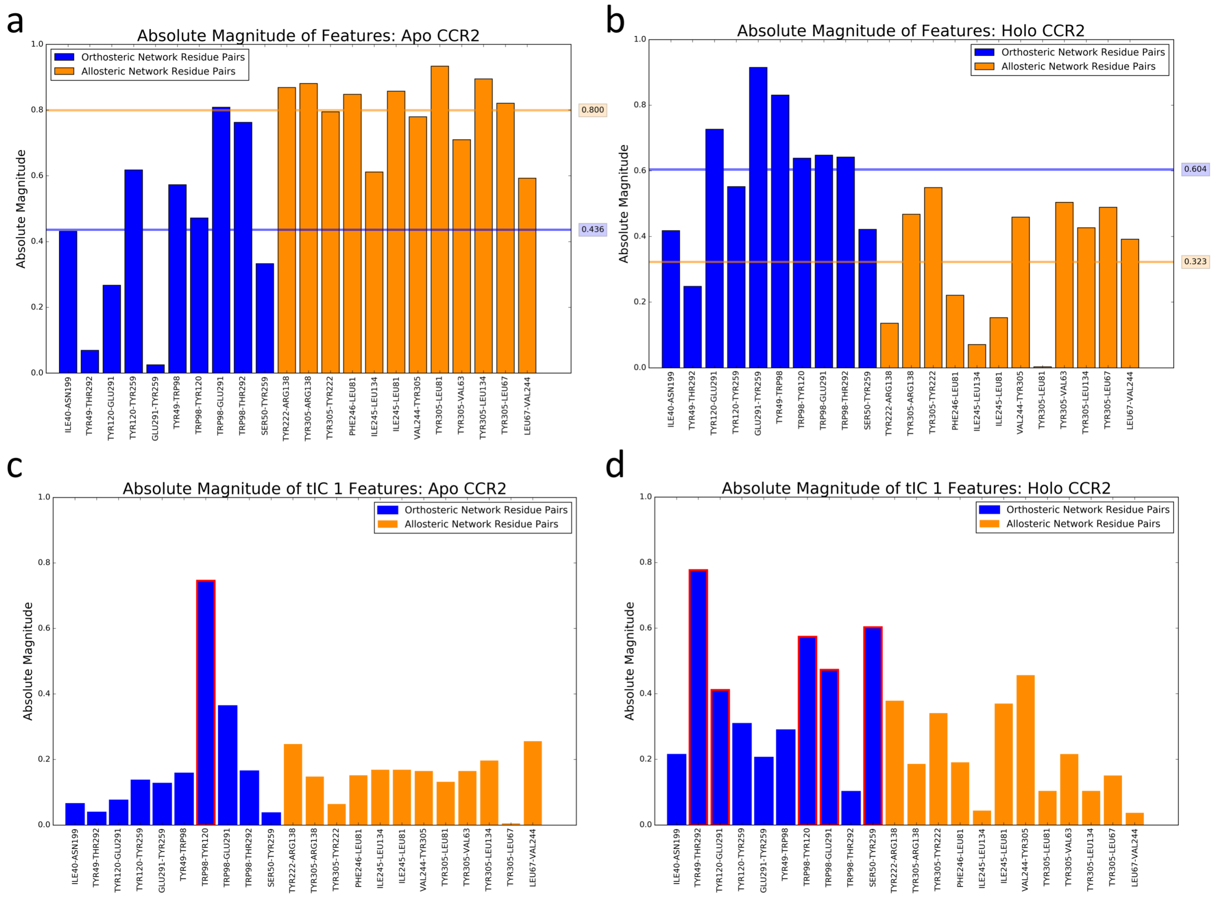


Figure 7.9: The absolute magnitude of each input feature in A) apo TIC 0, B) holo TIC 0, C) apo TIC 1, and D) holo TIC 1. Each bar represents the absolute magnitude of one inter-residue distance. Blue bars are distances between residue pairs in the orthosteric pocket, and orange bars are distances between residue pairs in the allosteric pocket.

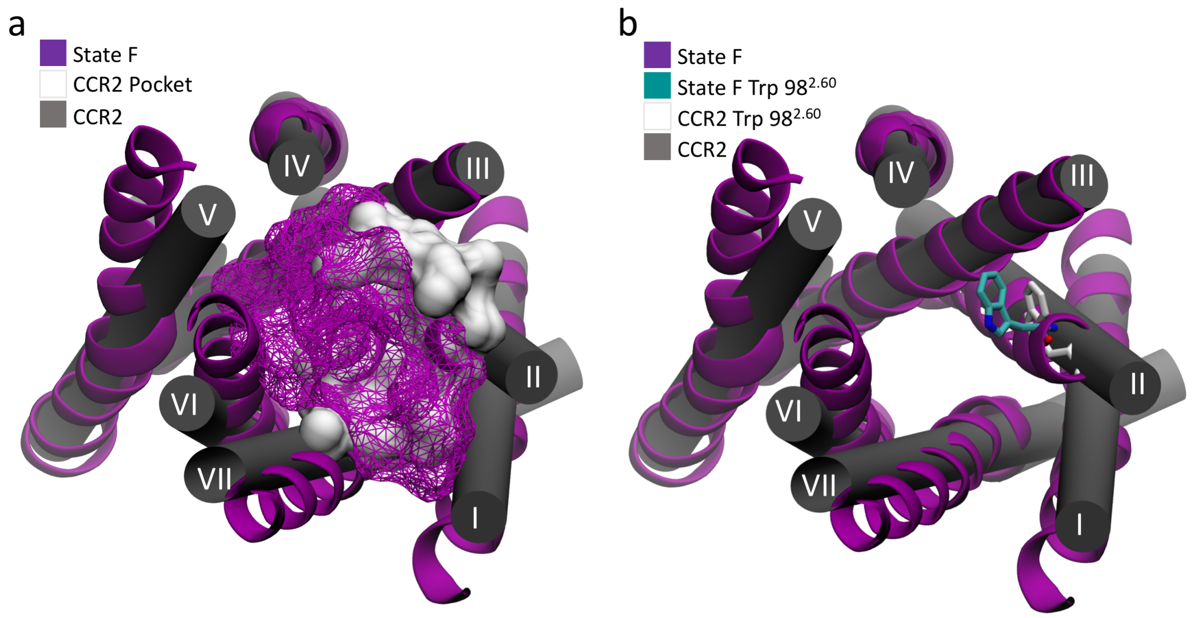


Figure 7.10: A) The shape of the chemokine binding site of apo state F in comparison to the crystal structure. Without the ligand, the binding site expands and rotates toward helices IV, V, and VI, and extends between helices I and VII. B) The conformations of Trp98$^{2.60}$ in apo state F and the crystal structure. Trp98$^{2.60}$ protrudes into the pocket in the absence of ligands.

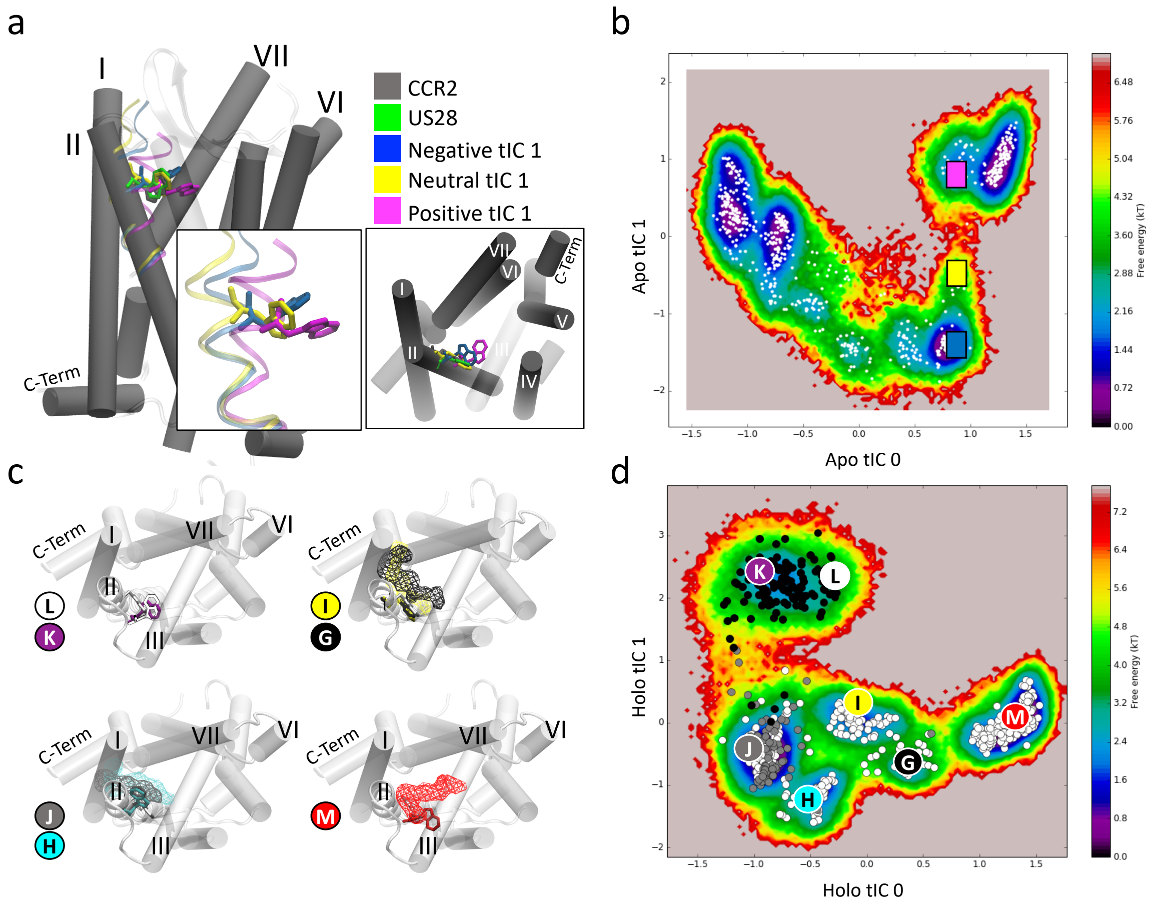


Figure 7.11: A) In apo CCR2, TIC 1 represents Trp $98^{2.60}$ in three distinct positions. In gray is the crystal structure; in green is the active crystal structure of US28; in blue and yellow are transitions, and in magenta is the most dramatic conformation. Each conformation is plotted on the free energy in TICA space in B). C) In holo CCR2, the positioning of the orthosteric ligand and the conformation of Trp $98^{2.60}$ is closely linked. Shown in light silver cartoon is CCR2 5T1A; Trp $98^{2.60}$ is displayed as purple in state K, white in state L, gray in state J, cyan in state H, yellow in state I, black in state G, and red in state M. D) Holo CCR2. White circles are clusters of frames before any ligand dissociation. Grey circles are clusters of frames during the event. Black circles are clusters of frames after the event.

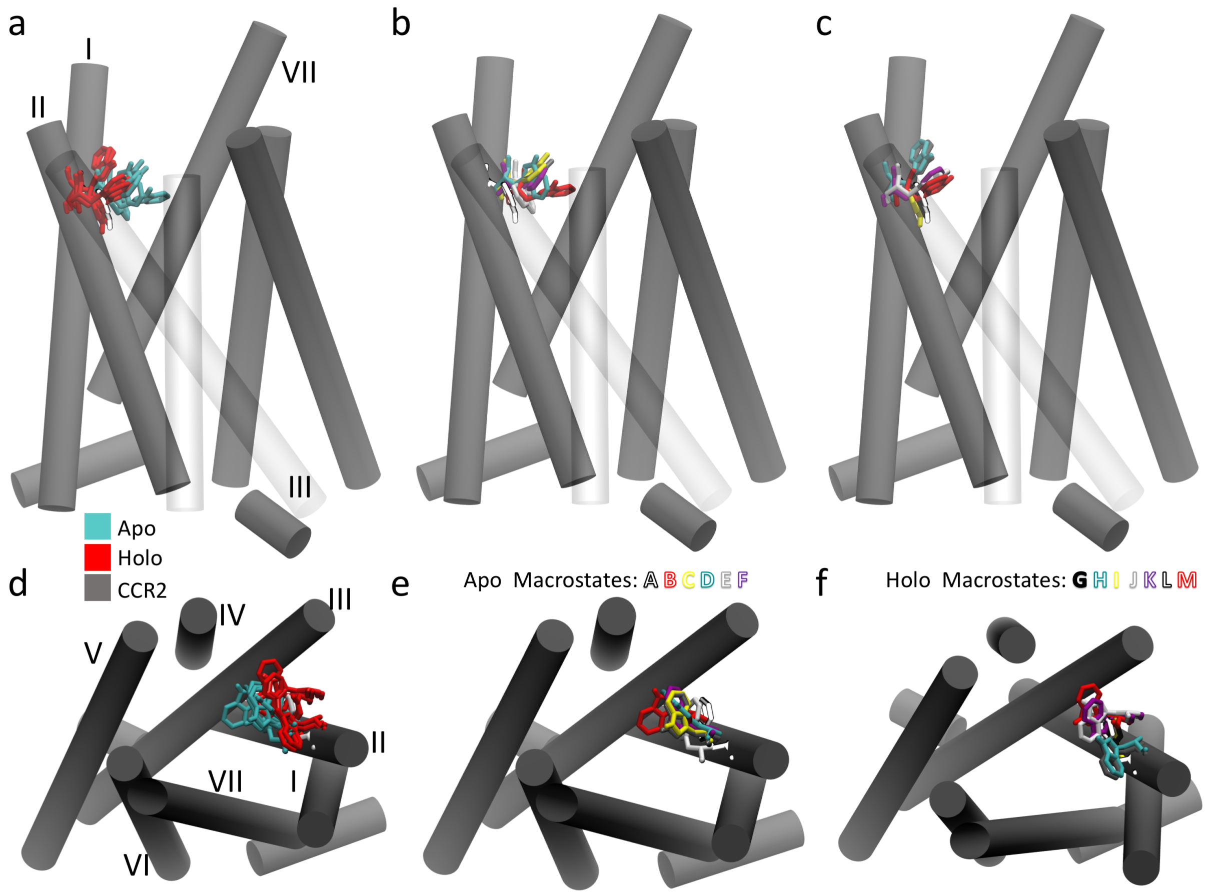


Figure 7.12: Trp 98$^{2.60}$ extends farther into the chemokine binding site in apo CCR2 than in the holo crystal structure and holo macrostates. A side view of the CCR2 crystal structure (grey) and Trp 98$^{2.60}$ (bright white) compared to A) the apo (cyan) and holo (red) macrostate conformations of Trp 98$^{2.60}$ B) the holo conformations, and C) the apo conformations. The extracellular-to-intracellular view of A)-C).

a

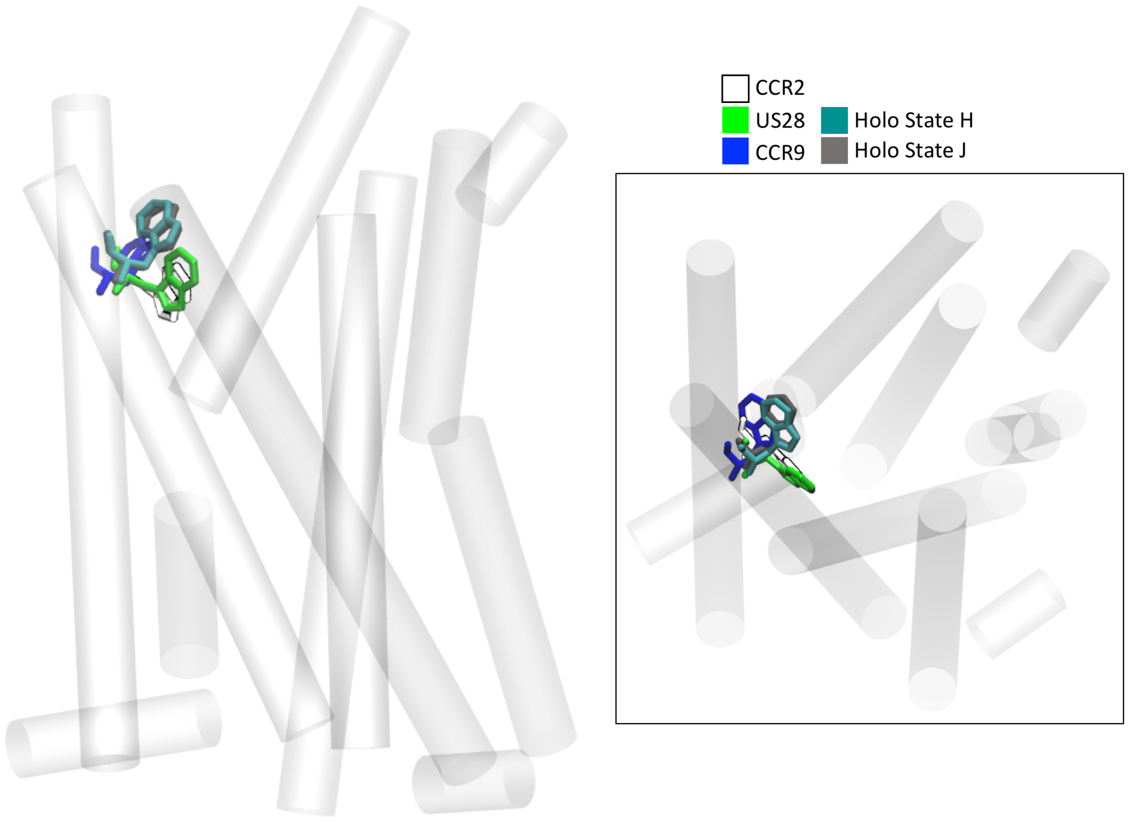


Figure 7.13: A) Trp $98^{2.60}$ in holo states H and J (cyan and grey) compared to CCR9 (dark blue) and the CCR2 crystal structure (white). Inset depicts the extracellular-to-intracellular view of A).

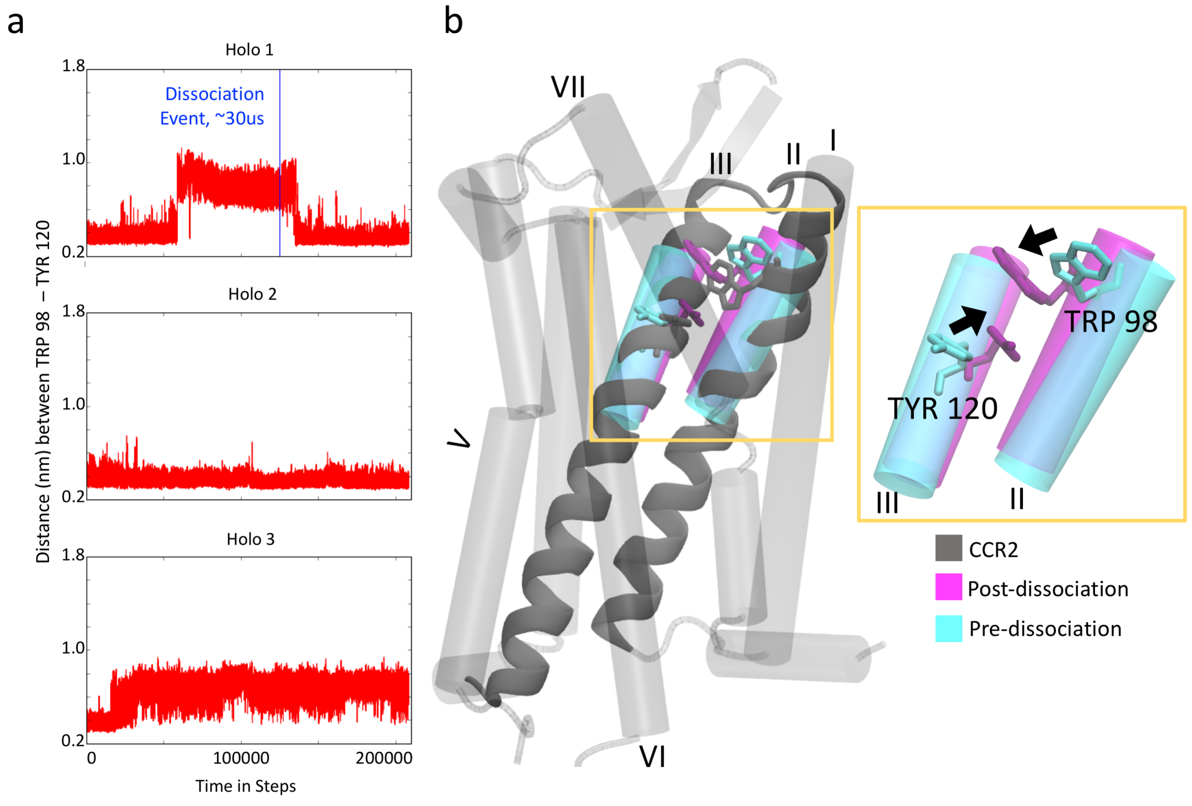


Figure 7.14: A) Distance between residues Trp $98^{2.60}$ and Tyr $120^{3.32}$ over simulation time. Dissociation event noted by blue line. B) Conformation of CCR2 before (teal) and after (purple) ligand dissociation.

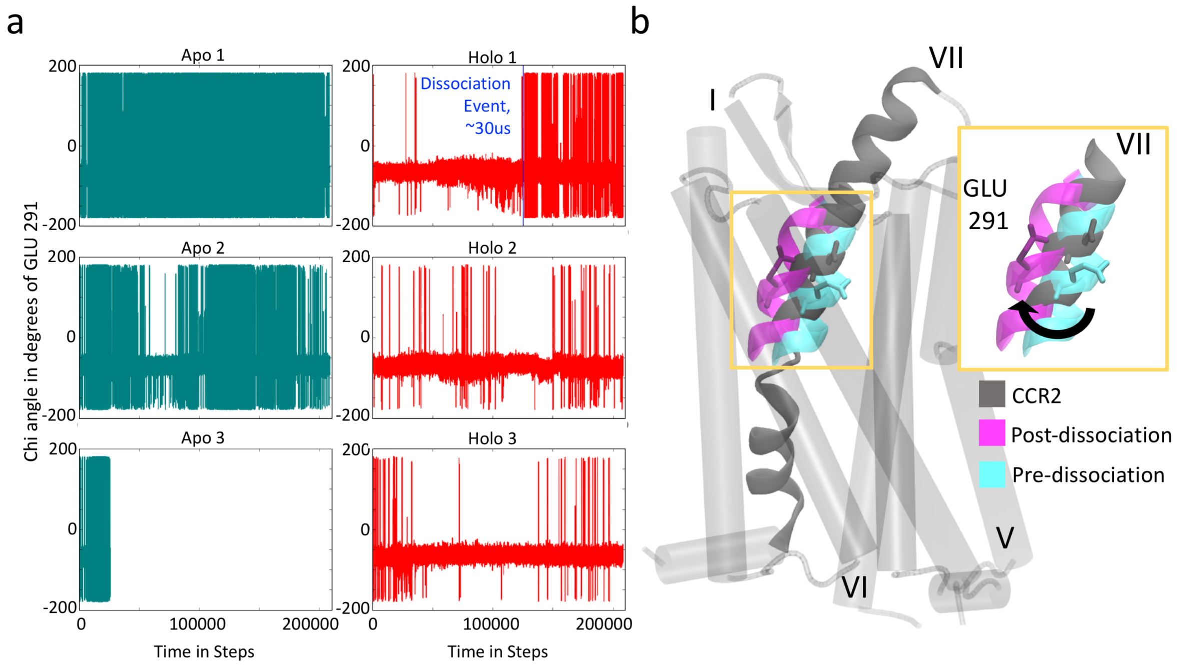


Figure 7.15: A) Chi angle of Glu 291 over simulation time. Dissociation event noted by blue line. B) Conformation of CCR2 before (teal) and after (purple) ligand dissociation.

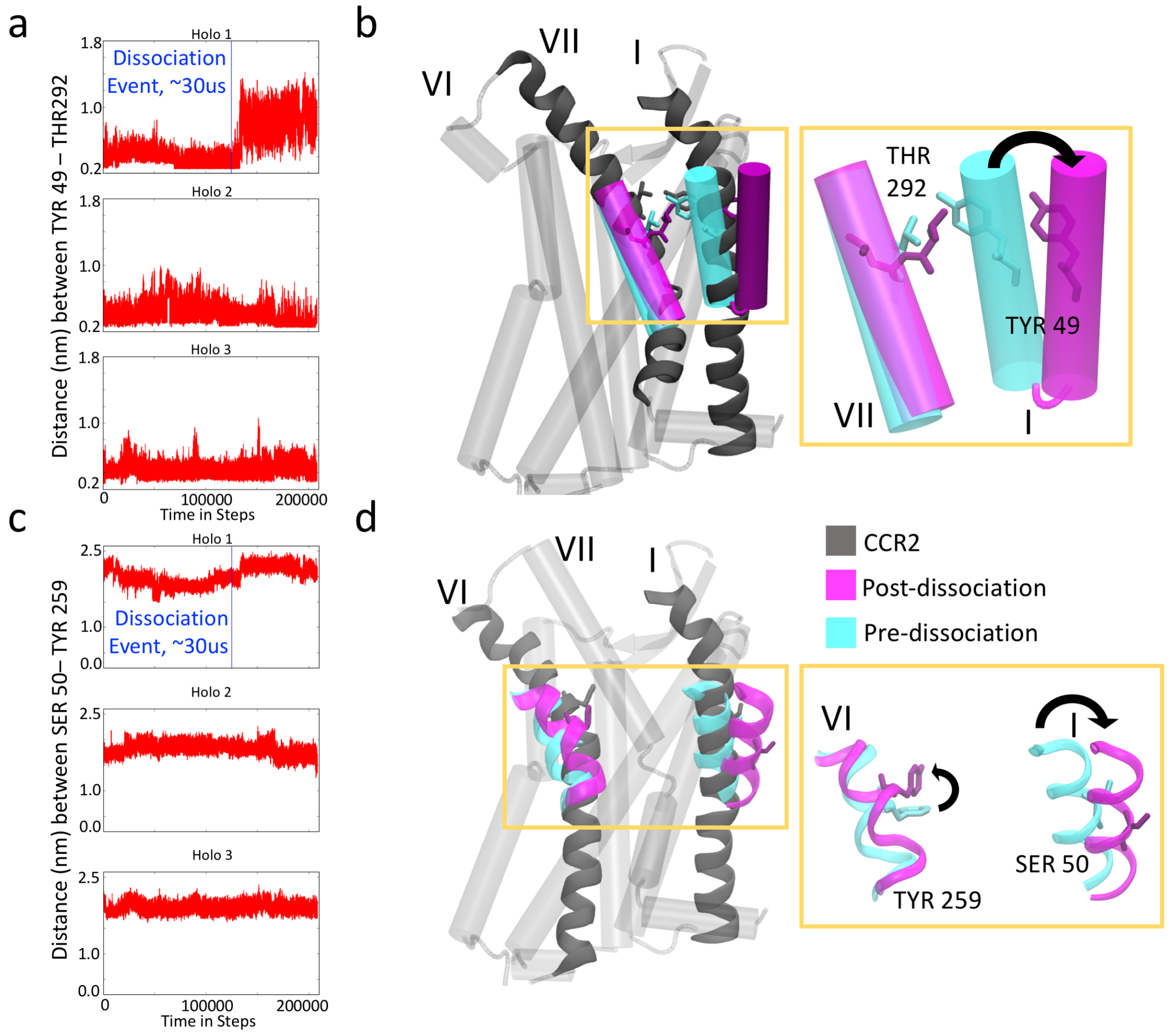
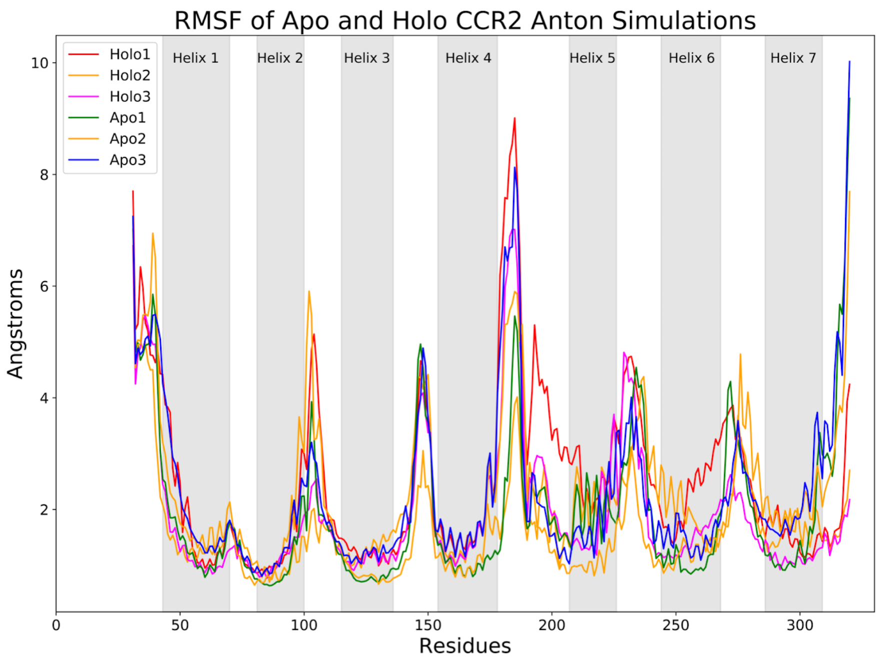


Figure 7.16: Orthosteric ligand dissociation breaks the hydrogen bond between key ligand binding residues. A) Distance between residues TYR 49$^{1.39}$ - Thr 292 over simulation time. Dissociation event noted by blue line. B) Conformation of CCR2 before (teal) and after (purple) ligand dissociation. C) The distance between Ser 50$^{1.40}$ and Tyr 259$^{6.51}$ over simulation time. This distance is also a contributor to holo TIC 1, and shows the same outward movement of helix I. There is a slight decrease in distance between the residue pair, followed by the same lag time of 3 μs, and finally an increase in distance as the extracellular end of helix I bends away from the helical bundle. D) Conformation of CCR2 before (teal) and after (purple) ligand dissociation.

a



b

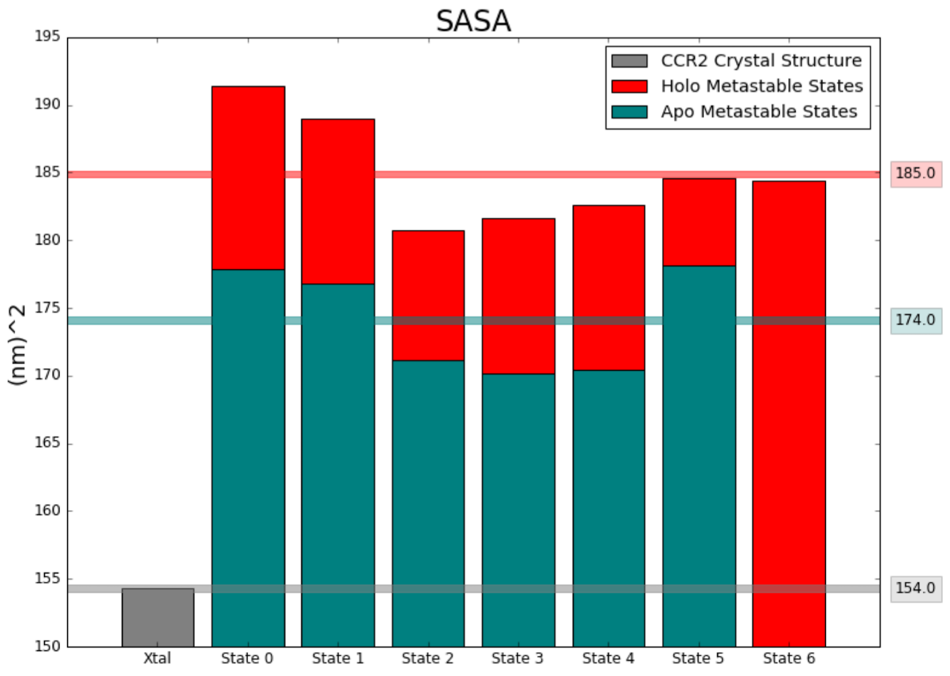


Figure 7.17: A) RMSF of each residue for individual simulations of apo and holo CCR2. B) SASA for each metastable macrostate of apo and holo, compared to the CCR2 crystal structure.

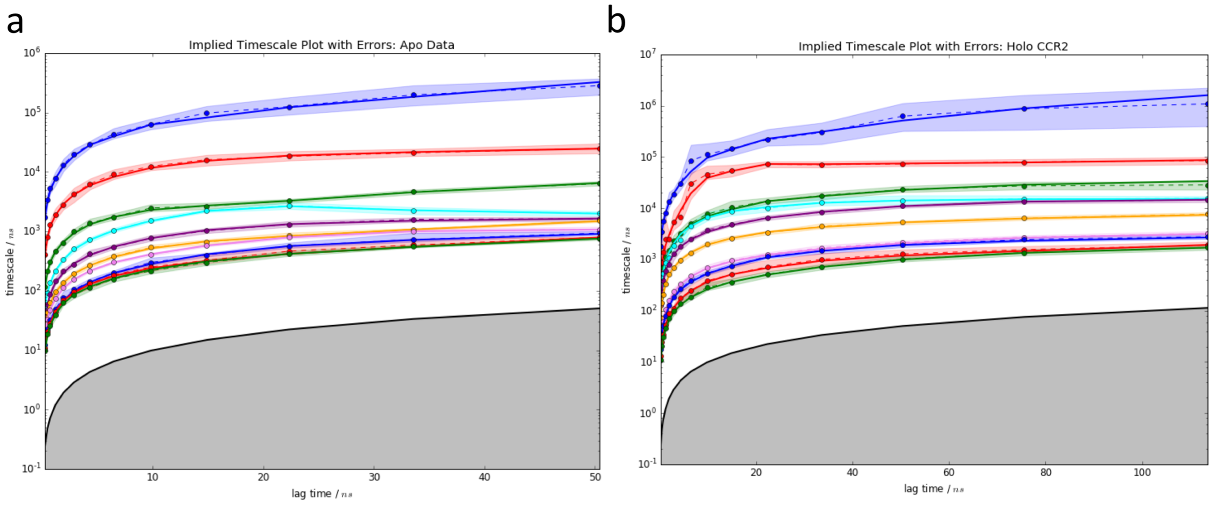


Figure 7.18: Implied timescale plots for A) apo and B) holo CCR2.

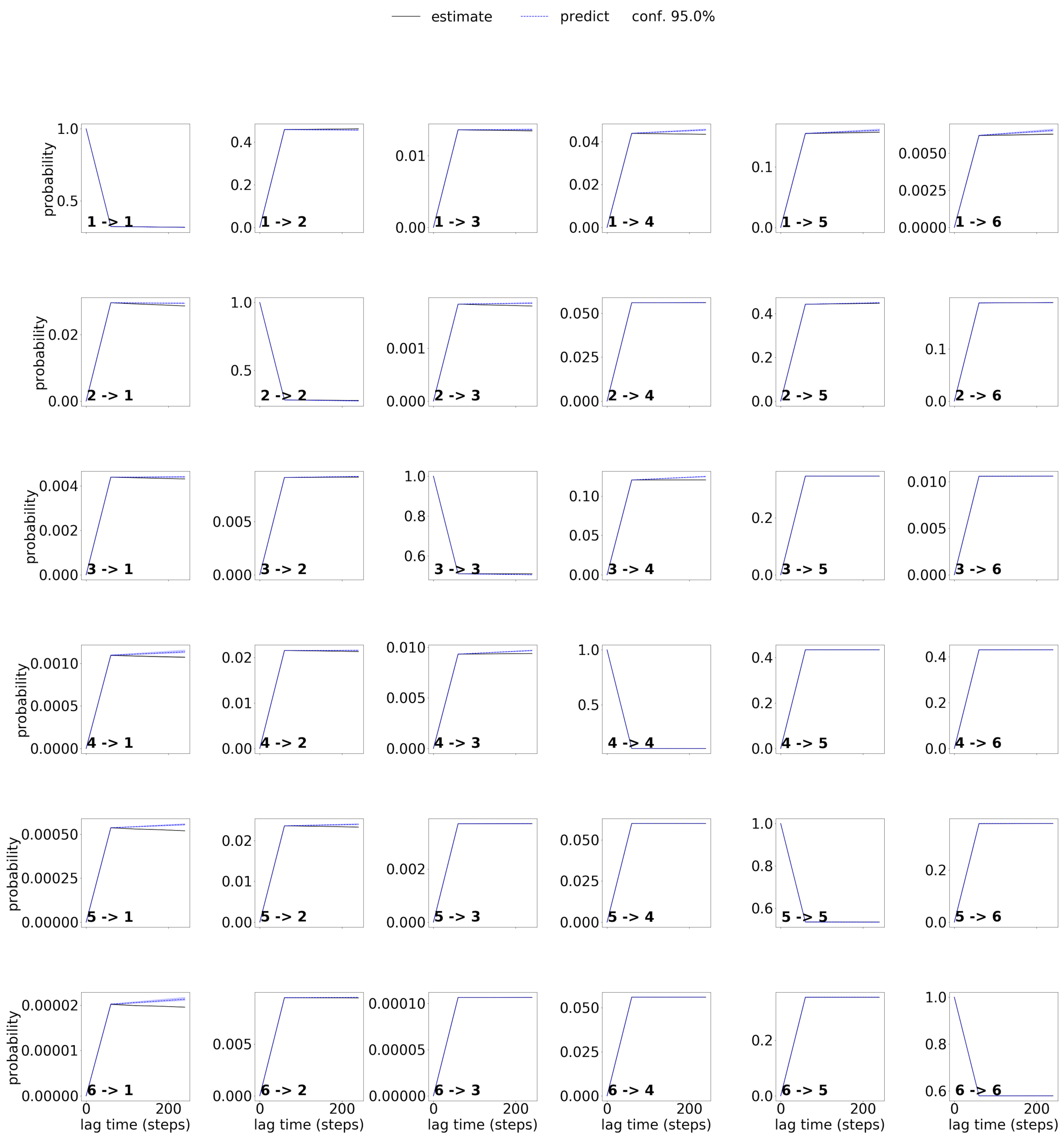


Figure 7.19: Chapman-Kolmogorov test for apo CCR2 MSM.

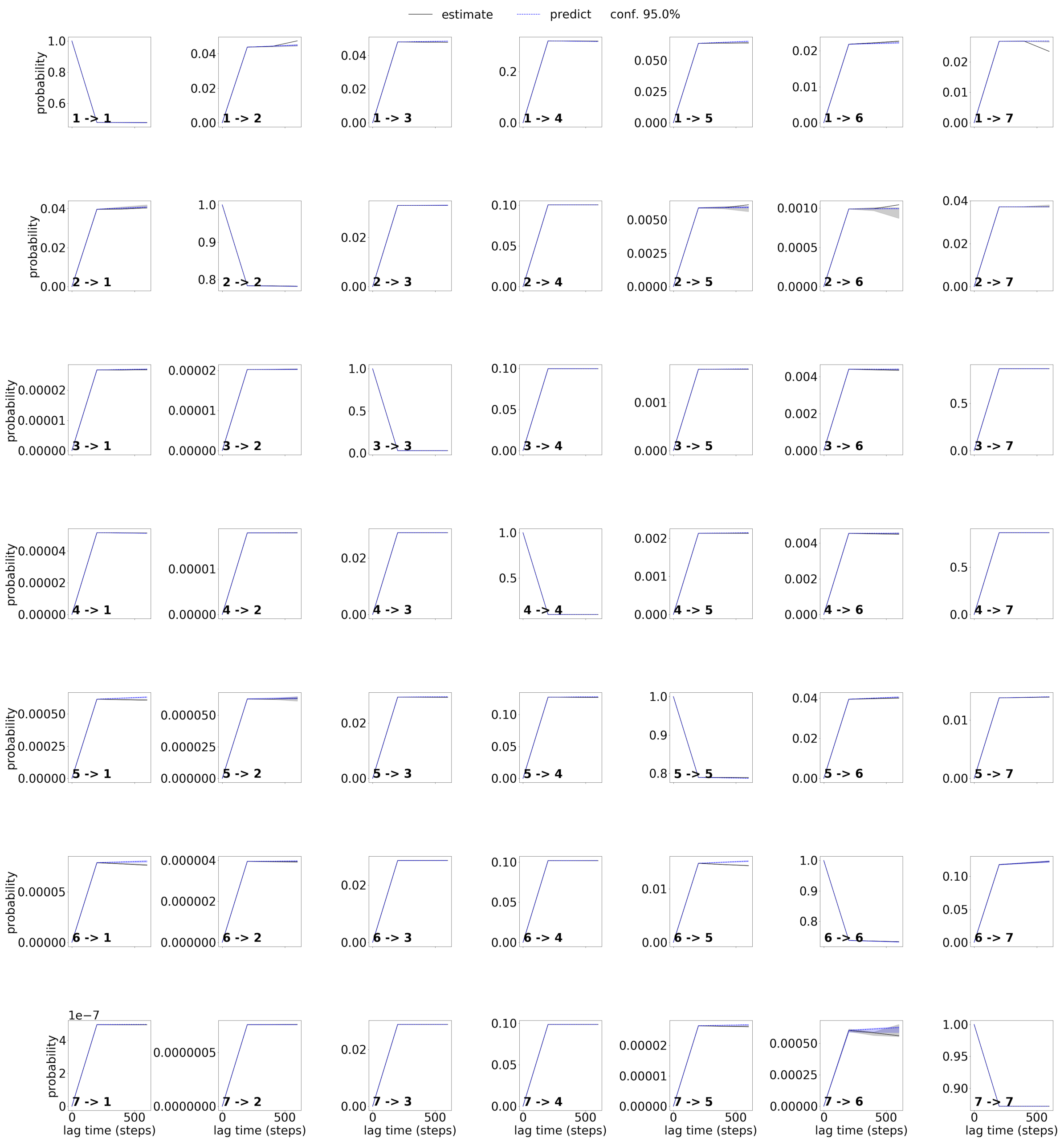


Figure 7.20: Chapman-Kolmogorov test for holo CCR2 MSM.

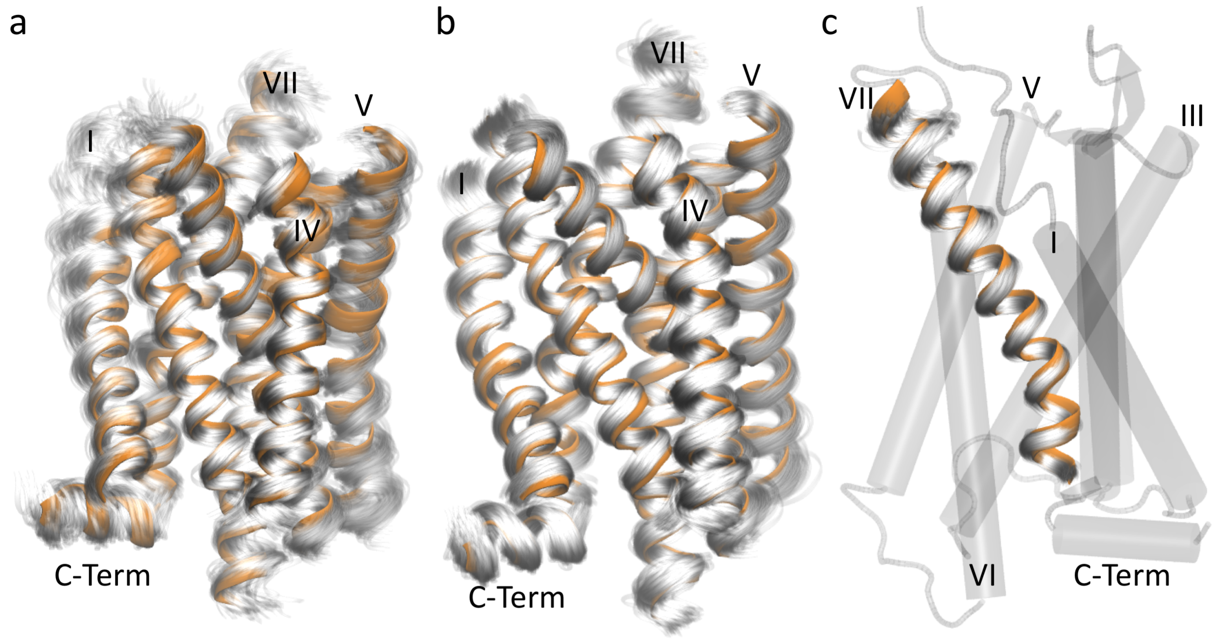


Figure 7.21: The centroid of apo macrostate A, in orange, is compared to A) the 214 other microstate centroids in white (average RMSD of alpha helices from representative structure: 2.01 Å, standard deviation 0.50 Å), and B) the 1,024 frames from its microstate in white (average RMSD of alpha helices from centroid: 0.872 Å, standard deviation 0.139 Å). C) Helix VII of the same microstate (average RMSD from centroid: 1.035 Å, standard deviation 0.259 Å.)

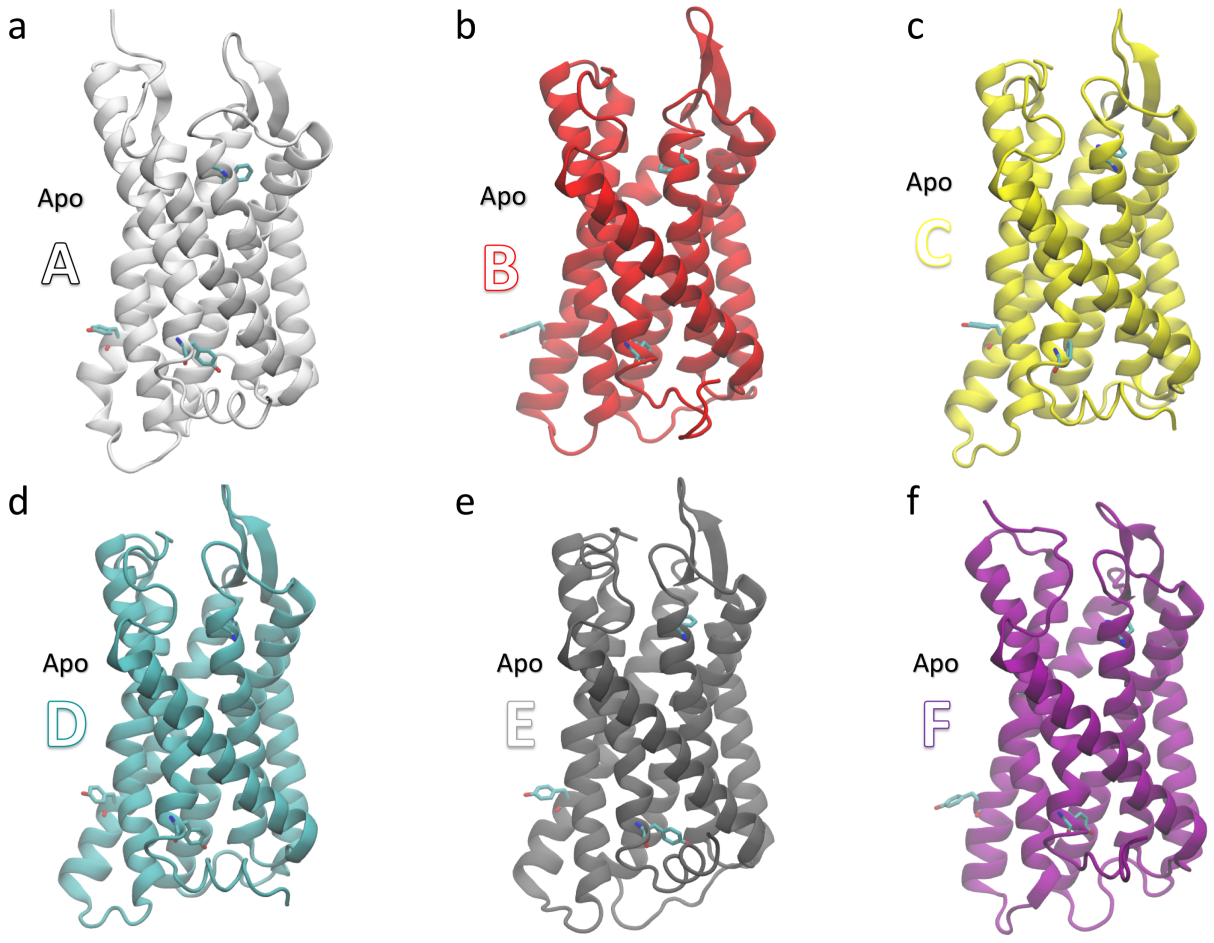


Figure 7.22: Visualization of apo macrostate structures.

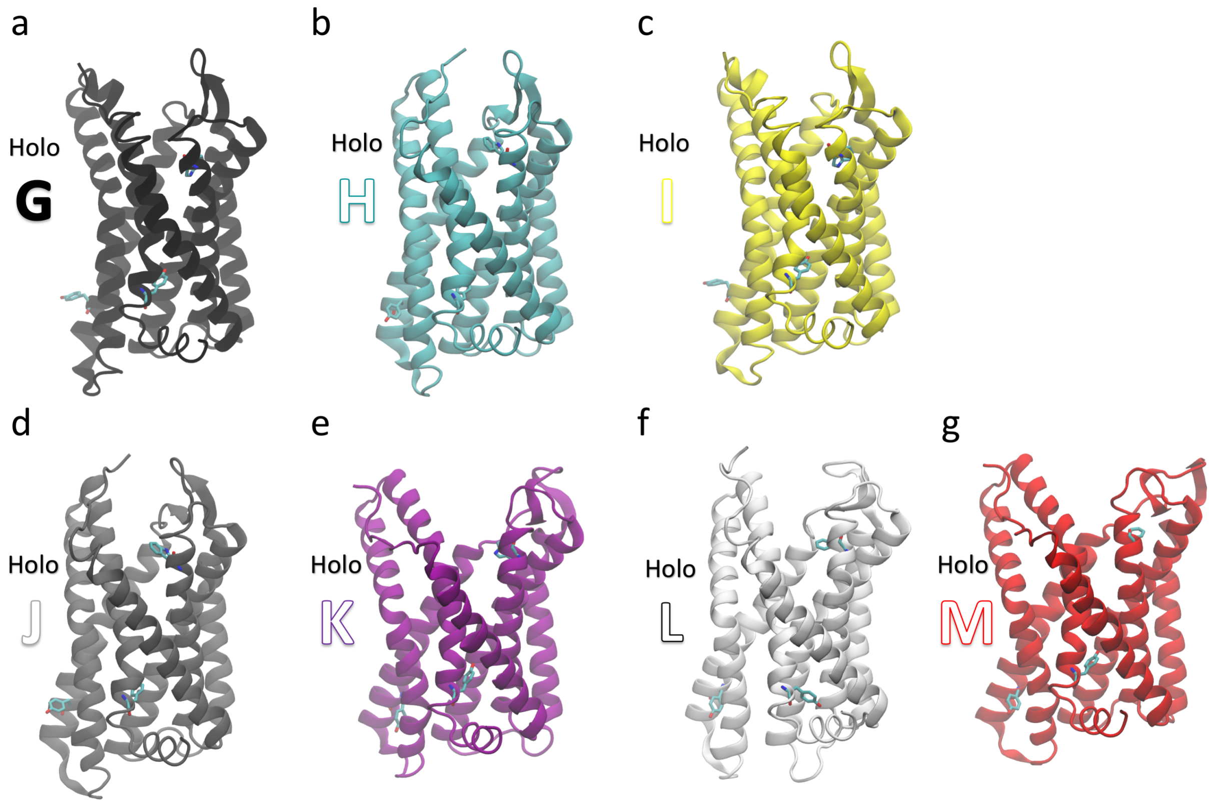


Figure 7.23: Visualization of holo macrostate structures.

References

(1) Ben-Baruch, A. *Cancer and Metastasis Reviews* **2006**, *25*, 357–371.

(2) O'Connor, T.; Borsig, L.; Heikenwalder, M. *Endocrine, Metabolic and Immune Disorders - Drug Targets* **2015**, *15*, 105–118.

(3) Solomon, M.; Balasa, B.; Sarvetnick, N. *Autoimmunity* **2010**, *43*, 156–163.

(4) Scholten, D. J.; Canals, M.; Mussang, D.; Roumen, L.; Smit, M.; Wijtmans, M.; de Graaf, C.; Vischer, H.; Leurs, R. *Br. J. Pharmacol* **2012**, *165*, 1617–1643.

(5) Lim, S.; Yuzhalin, A.; Gordon-Weeks, A.; Muschel, R. *Oncotarget* **2016**, *7*, 28697–710.

(6) Solari, R.; Pease, J. E.; Begg, M. *Eur. J. Pharmacol.* **2015**, *746*, 363–367.

(7) Horuk, R. *Nature reviews. Drug discovery* **2009**, *8*, 23–33.

(8) Shahlaei, M.; Fassihi, A.; Papaleo, E.; Pourfarzam, M. *Chemical biology & drug design* **2013**, *82*, 534–545.

(9) Chavan, S.; Pawar, S.; Singh, R.; Sobhia, M. E. *Molecular Diversity* **2012**, *16*, 401–413.

(10) Kothandan, G.; Gadhe, C. G.; Cho, S. J. *PloS one* **2012**, *7*, e32864.

(11) Zheng, Y.; Qin, L.; Zacarías, N. V. O.; de Vries, H.; Han, G. W.; Gustavsson, M.; Dabros, M.; Zhao, C.; Cherney, R. J.; Carter, P.; Stamos, D.; Abagyan, R.; Cherezov, V.; Stevens, R. C.; IJzerman, A. P.; Heitman, L. H.; Tebben, A.; Kufareva, I.; Handel, T. M. *Nature* **2016**, *540*, 458–461.

(12) Latorraca, N. R.; Venkatakrishnan, A. J.; Dror, R. O. *Chemical Reviews* **2017**, *117*, 139–155.

(13) Venkatakrishnan, A. J.; Deupi, X.; Lebon, G.; Tate, C. G.; Schertler, G. F.; Babu, M. M. *Nature* **2013**, *494*, 185–194.

(14) Zhang, Q.; Zhou, M.; Zhao, L.; Jiang, H.; Yang, H. *Biochemistry* **2018**, null.

(15) Katritch, V.; Cherezov, V.; Stevens, R. *Annual review of pharmacology and toxicology* **2013**, *53*, 531–556.

(16) Manglik, A.; Kim, T. H.; Masureel, M.; Altenbach, C.; Yang, Z. Y.; Hilger, D.; Lerch, M. T.; Kobilka, T. S.; Thian, F. S.; Hubbell, W. L.; Prosser, R. S.; Kobilka, B. K. *Cell* **2015**, *162*, 1431–1431.

(17) Katritch, V.; Cherezov, V.; Stevens, R. C. *Trends Pharmacol. Sci.* **2012**, *33*, 17–27.

(18) Malik, R. U.; Ritt, M.; DeVree, B. T.; Neubig, R. R.; Sunahara, R. K.; Sivaramakrishnan, S. *J. Biol. Chem.* **2013**, *288*, 17167–17178.

(19) Yao, X. J.; Velez Ruiz, G.; Whorton, M. R.; Rasmussen, S. G. F.; DeVree, B. T.; Deupi, X.; Sunahara, R. K.; Kobilka, B. *Proc. Natl. Acad. Sci. U. S. A.* **2009**, *106*, 9501–9506.

(20) Nygaard, R.; Zou, Y. Z.; Dror, R. O.; Mildorf, T. J.; Arlow, D. H.; Manglik, A.; Pan, A. C.; Liu, C. W.; Fung, J. J.; Bokoch, M. P.; Thian, F. S.; Kobilka, T. S.; Shaw, D. E.; Mueller, L.; Prosser, R. S.; Kobilka, B. K. *Cell* **2013**, *152*, 532–542.

(21) Bockenhauer, S.; Furstenberg, A.; Yao, X. J.; Kobilka, B. K.; Moerner, W. E. *J. Phys. Chem. B.* **2011**, *115*, 13328–13338.

(22) Shaw, D. E.; Grossman, J. P.; Bank, J. A.; Batson, B.; Butts, J. A.; Chao, J. C.; Deneroff, M. M.; Dror, R. O.; Even, A.; Fenton, C. H.; Forte, A.; Gagliardo, J.; Gill, G.; Greskamp, B.; Ho, C. R.; Ierardi, D. J.; Iserovich, L.; Kuskin, J. S.; Larson, R. H.; Layman, T.; Lee, L. S.; Lerer, A. K.; Li, C.; Killebrew, D.; Mackenzie, K. M.; Mok, S. Y. H.; Moraes, M. A.; Mueller, R.; Nociolo, L. J.; Peticolas, J. L.; Quan, T.; Ramot, D.; Salmon, J. K.; Scarpazza, D. P.; Ben Schafer, U.; Siddique, N.; Snyder, C. W.; Spengler, J.; Tang, P. T. P.; Theobald, M.; Toma, H.; Towles, B.; Vitale, B.; Wang, S. C.; Young, C. In *Int. Conf. High Perform. Comput. Networking, Storage Anal. SC*, IEEE: 2014; Vol. 2015-Janua, pp 41–53.

(23) Bowman, G. R.; Voelz, V. a.; Pande, V. S. *Current Opinion in Structural Biology* **2011**, *21*, 4–11.

(24) Swope, W. C.; Pitera, J. W. *J. Phys. Chem. B* **2004**, *108*, 6571–6581.

(25) Singhal, N.; Snow, C. D.; Pande, V. S. *Journal of Chemical Physics* **2004**, DOI: 10.1063/1.1738647.

(26) Malmstrom, R. D.; Lee, C. T.; Van Wart, A. T.; Amaro, R. E. *J. Chem. Theory Comput.* **2014**, *10*, 2648–2657.

(27) Amaro, R. E.; Mulholland, A. J. *Nat Rev Chem* **2018**, *2*, 0148.

(28) Amaro, R. E.; Baudry, J.; Chodera, J.; Demir, Ö.; McCammon, A.; Miao, Y.; Smith, J. C. *Biophysical Journal* **2018**, *114*, 2271–2278.

(29) Bowman, G. R.; Pande, V. S.; Noe, F., *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation*; Springer Netherlands: 2014; Vol. 797.

(30) Noé, F.; Horenko, I.; Schütte, C.; Smith, J. C. *The Journal of Chemical Physics* **2007**, *126*, 155102.

(31) Prinz, J. H.; Wu, H.; Sarich, M.; Keller, B.; Senne, M.; Held, M.; Chodera, J. D.; Schtte, C.; No??, F. *Journal of Chemical Physics* **2011**, *134*, DOI: 10.1063/1.3565032.

(32) SchÃŒtte, C.; Fischer, A.; Huisinga, W.; Deuflhard, P. *Journal of Computational Physics* **1999**, *151*, 146–168.

(33) Noé, F.; Nüske, F. *Multiscale Modeling & Simulation* **2013**, *11*, 635–655.

(34) Schwantes, C. R.; Pande, V. S. *J. Chem. Theory Comput.* **2013**, *9*, 2000–2009.

(35) Perez-Hernandez, G.; Paul, F.; Giorgino, T.; De Fabritiis, G.; Noe, F. *J. Chem. Phys.* **2013**, *139*, 015102.

(36) Lebon, G.; Warne, T.; Edwards, P. C.; Bennett, K.; Langmead, C. J.; Leslie, A. G. W.; Tate, C. G. *Nature* **2011**, *474*, 521–525.

(37) Burg, J. S.; Ingram, J. R.; Venkatakrishnan, A. J.; Jude, K. M.; Dukkipati, A.; Feinberg, E. N.; Angelini, A.; Waghray, D.; Dror, R. O.; Ploegh, H. L.; Garcia, K. C. *Science* **2015**, *347*, 1113–1117.

(38) Tan, Q.; Zhu, Y.; Li, J.; Chen, Z.; Han, G. W.; Kufareva, I.; Li, T.; Ma, L.; Fenalti, G.; Li, J.; Zhang, W.; Xie, X.; Yang, H.; Jiang, H.; Cherezov, V.; Liu, H.; Stevens, R. C.; Zhao, Q.; Wu, B. *Science* **2013**, *341*, 1387–1390.

(39) Oswald, C.; Rappas, M.; Kean, J.; Doré, A. S.; Errey, J. C.; Bennett, K.; Deflorian, F.; Christopher, J. A.; Jazayeri, A.; Mason, J. S.; Congreve, M.; Cooke, R. M.; Marshall, F. H. *Nature* **2016**, *540*, 462–465.

(40) Qin, L.; Kufareva, I.; Holden, L. G.; Wang, C.; Zheng, Y.; Zhao, C.; Fenalti, G.; Wu, H.; Han, G. W.; Cherezov, V.; Abagyan, R.; Stevens, R. C.; Handel, T. M. *Science* **2015**, *347*, 1117–1122.

(41) Caliman, A.; Swift, S.; Wang, Y.; Miao, Y.; McCammon, J. *Protein Science: A Publication of the Protein Society* **2015**, *24*, 1004–1012.

(42) Jastrzebska, B.; Palczewski, K.; Golczak, M. *The Journal of biological chemistry* **2011**, *286*, 18930–7.

(43) Angel, T. E.; Chance, M. R.; Palczewski, K. *Proceedings of the National Academy of Sciences of the United States of America* **2009**, *106*, 8555–8560.

(44) Choe, H.-W.; Kim, Y. J.; Park, J. H.; Morizumi, T.; Pai, E. F.; Krauß, N.; Hofmann, K. P.; Scheerer, P.; Ernst, O. P. *Nature* **2011**, *471*, 651–655.

(45) Huang, W.; Manglik, A.; Venkatakrishnan, A. J.; Laeremans, T.; Feinberg, E. N.; Sanborn, A. L.; Kato, H. E.; Livingston, K. E.; Thorsen, T. S.; Kling, R. C.; Granier, S.; Gmeiner, P.; Husbands, S. M.; Traynor, J. R.; Weis, W. I.; Steyaert, J.; Dror, R. O.; Kobilka, B. K. *Nature* **2015**, *524*, 315–321.

(46) Yuan, S.; Filipek, S.; Palczewski, K.; Vogel, H. *Nature communications* **2014**, *5*, 4733.

(47) Rinne, A.; Birk, A.; Bünemann, M. *Proceedings of the National Academy of Sciences of the United States of America* **2013**, *110*, 1536–41.

(48) Katritch, V.; Fenalti, G.; Abola, E. E.; Roth, B. L.; Cherezov, V.; Stevens, R. C. *Trends in biochemical sciences* **2014**, *39*, 233–44.

(49) Yuan, S.; Vogel, H.; Filipek, S. *Angewandte Chemie International Edition* **2013**, *52*, 10112–10115.

(50) Vickery, O. N.; Carvalheda, C. A.; Zaidi, S. A.; Pisliakov, A. V.; Katritch, V.; Zachariae, U. *Structure* **2018**, *26*, 171–180.

(51) Miao, Y.; Feixas, F.; Eun, C.; McCammon, J. A. *J. Comput. Chem.* **2015**, *36*, 1536–1549.

(52) Berkhout, T. A.; Blaney, F. E.; Bridges, A. M.; Cooper, D. G.; Forbes, I. T.; Gribble, A. D.; Groot, P. H. E.; Hardy, A.; Ife, R. J.; Kaur, R.; Moores, K. E.; Shillito, H.; Willetts, J.; Witherington, J. *Journal of medicinal chemistry* **2003**, *46*, 4070–86.

(53) Hall, S. E.; Mao, A.; Nicolaidou, V.; Finelli, M.; Wise, E. L.; Nedjai, B.; Kanjanapangka, J.; Harirchian, P.; Chen, D.; Selchau, V.; Ribeiro, S.; Schyler, S.; Pease, J. E.; Horuk, R.; Vaidehi, N. *Molecular pharmacology* **2009**, *75*, 1325–36.

(54) Cherney, R.; Nelson, D.; Lo, Y.; Yang, G.; Scherle, P.; Jezak, H.; Solomon, K.; Carter, P.; Decicco, C. *Bioorg Med Chem Lett.* **2008**, *18*, 5063–5.

(55) Kozakov, D.; Grove, L.; Hall, D.; Bohnuud, T.; Mottarella, S.; Luo, L.; Xia, B.; Beglov, D.; Vajda, S. *Nature Protocols* **2015**, *10*, 733–755.

(56) Bowman, G. R.; Bolin, E. R.; Hart, K. M.; Maguire, B. C.; Marqusee, S. *Proceedings of the National Academy of Sciences of the United States of America* **2015**, *112*, 2734–2739.

(57) Scherer, M. K.; Trendelkamp-Schroer, B.; Paul, F.; Pérez-Hernández, G.; Hoffmann, M.; Plattner, N.; Wehmeyer, C.; Prinz, J. H.; Noé, F. *Journal of Chemical Theory and Computation* **2015**, *11*, 5525–5542.

(58) Dasse, O. A.; Evans, J. L.; Zhai, H.-X.; Zou, D.; Kintigh, J. T.; Chan, F.; Hamilton, K.; Hill, E.; Eckman, J. B.; Higgins, P. J.; Volosov, A.; Collart, P.; Nicolas, J.-M.; Kondru, R.; Schwartz, C. *Letters in Drug Design and Discovery* **2007**, *4*, 263–271.

(59) Carter, P. H.; Brown, G. D.; Cherney, R. J.; Batt, D. G.; Chen, J.; Clark, C. M.; Cvijic, M. E.; Duncia, J. V.; Ko, S. S.; Mandlekar, S.; Mo, R.; Nelson, D. J.; Pang, J.; Rose, A. V.; Santella, J. B.; Tebben, A. J.; Traeger, S. C.; Xu, S.; Zhao, Q.; Barrish, J. C. *ACS Med Chem Lett.* **2015**, *6*, PMID: 25893046, 439–444.

(60) Jo, S.; Kim, T.; Iyer, V. G.; Im, W. *Journal of Computational Chemistry* **2008**, *29*, 1859–1865.

(61) Vanommeslaeghe, K.; Hatcher, E.; Acharya, C.; Kundu, S.; Zhong, S.; Shim, J.; Darian, E.; Guvench, O.; Lopes, P.; Vorobyov, I.; Mackerell, A. D. *J. Comput. Chem.* **2010**, *31*, 671–690.

(62) Huang, J.; Mackerell, A. D. *Journal of Computational Chemistry* **2013**, *34*, 2135–2145.

(63) Humphrey W. Dalke, A.; Schulten, K. *J. Molec. Graphics* **1996**, *14*, 33–38.

(64) Beauchamp, K. A.; Bowman, G. R.; Lane, T. J.; Maibaum, L.; Haque, I. S.; Pande, V. S. *Journal of Chemical Theory and Computation* **2011**, *7*, 3412–3419.

(65) Pande, V. S.; Beauchamp, K.; Bowman, G. R. Everything you wanted to know about Markov State Models but were afraid to ask., 2010.

(66) Prinz, J.-H.; Keller, B.; Noe, F. *Phys. Chem. Chem. Phys.* **2011**, *13*, 16912–16927.

(67) Senne, M.; Trendelkamp-Schroer, B.; Mey, A. S.; Schütte, C.; Noé, F. *J. Chem. Theory Comput.* **2012**, *8*, 2223–2238.

(68) Cronkite-Ratcliff, B.; Pande, V. *Bioinformatics* **2013**, *29*, 950–952.

(69) Nüske, F.; Keller, B. G.; Pérez-Hernández, G.; Mey, A. S. J. S.; Noé, F. *Journal of Chemical Theory and Computation* **2014**, *10*, PMID: 26580382, 1739–1752.

(70) Noé, F.; Wu, H.; Prinz, J. H.; Plattner, N. *J. Chem. Phys.* **2013**, *139*, DOI: 10.1063/1.4828816.

Chapter 8

The Implementation of the Colored Abstract Simplicial Complex and its Application to Mesh Generation

8.1 Abstract

We introduce CASC: a new, modern, and header-only C++ library which provides a data structure to represent arbitrary dimension abstract simplicial complexes (ASC) with user-defined classes stored directly on the simplices at each dimension. This is accomplished by using the latest C++ language features including variadic template parameters introduced in C++11 and automatic function return type deduction from C++14. Effectively CASC decouples the representation of the topology from the interactions of user data. We present the innovations and design principles of the data structure and related algorithms. This includes a meta-data aware decimation algorithm which is general for collapsing simplices of any dimension. We also present an example application of this library to represent an orientable surface mesh.

8.2 Introduction

For problems in computational topology and geometry, it is often beneficial to use simple building blocks to represent complicated shapes. A popular block is the simplex, or the generalization of a triangle in any dimension. Due to the ease of manipulation and the coplanar property of triangles, triangulations have become commonplace in fields such as geometric modeling and visualization as well as topological analysis. Discretizations are also used for efficient solving of Partial Differential Equations (PDE). The use of meshes has become increasingly popular even in the fields of computational biology and medicine[1].

As methods in structural biology improve and new datasets become available, there is interest in integrating experimental and structural data to build new predictive computer models[2]. A key barrier that modelers face is the generation of multi-scale, computable, geometric models from noisy datasets such as those from Electron Tomography (ET)[3]. This is typically achieved in at least two steps: (1) segmentation of relevant features, and (2) approximation of the geometry using meshes. Subsequently, numerical techniques such as Finite Elements Modeling or Monte Carlo can be used to investigate the transport and localization of molecules of interest.

While many have studied mesh generation in the fields of engineering and animation, few methods are suitable for biological datasets. This is largely due to noise introduced by limits in image resolution or contrast. Even while using state-of-the-art segmentation algorithms for ET datasets there are often unresolved or missed features. Due to these issues, the generated meshes often have holes and other non-manifolds which must be resolved prior to mathematical modeling. Another challenge is the interpretation of a voxel valued segmentation. The conversion of zig-zag boundaries into a mesh can lead to other problems such as extremely high aspect ratio triangles or, in general, poorly conditioned elements[3]. To remedy this, various smoothing and decimation algorithms must also be applied prior to simulation.

Previous work by us and others have introduced a meshing tool for biological models, `GAMer`, for building 3D tetrahedral meshes which obey internal and external constraints, such as matching embedding and/or enclosing molecular surfaces. It also provides the ability to use various mesh improvement algorithms for volume and surface meshes[4, 5]. `GAMer` uses the `Tetgen` library as the primary tetrahedral volume generator[6]. While the algorithms are sound, the specific implementation is prone to segmentation faults even for simple meshes. Careful analysis of the code has identified that the data structures used for the representation of the mesh is primarily at fault. This article will focus entirely on the representation of topology in very complex mesh generation codes. We note that the algorithms which handle geometric issues like shape regularity and local adaptivity are well understood[7, 8], among others. Similarly there is a large body of literature related to local mesh refinement and decimation[9, 10]. Our innovations serve to enable the implementation of these algorithms in the most general and robust way.

`GAMer` currently employs a neighbor list data structure which tracks the adjacency and orientation of simplices. Neighbor lists are quick to construct, however the representation of non-manifolds often leads to code instability. Algorithms must check for aberrant cases creating substantial overhead. We note that while the need to gracefully represent 2D and 3D non-manifolds for ET applications drove our initial focus, we are also interested in mesh generation in higher dimensions with applications to: numerical general relativity (3D+1)[11, 12], computational geometric analysis (nD)[13], phase space simulations (6D), and arbitrary collective variable spaces in molecular modeling for enhanced sampling[14]. We therefore chose following requirements for a mesh data structure to serve as design goals:

- General and capable of representing non-manifold, mixed dimensional, oriented and non-oriented meshes in arbitrary dimensions.
- Support for inline and flexible data storage. In some applications, data must be associated with the topology. For example, problems in general relativity typically require the storage of metric tensors on all simplex dimensions.
- Support for intuitive and simple manipulations and traversals.

Here we describe the development of a scalable *colored abstract simplicial complex* data structure called CASC. Simplices are stored as nodes on a Hasse diagram. For ease of traversal all adjacency is stored at the node level. An additional data object can be stored at each node which is typed according to the simplex dimension at compile time. This means that, for example, for a mesh the 0-simplices can be assigned a vertex type while the 2-simplices can store some material property instead. Typing of each *k*-simplex is achieved using variadic templates introduced in C++11. CASC thus provides a natural separation between the combinatorics represented by the ASC from the underlying data types at each simplex dimension and their interactions. In §8.3 we briefly define an ASC and some relevant definitions followed by the introduction of the CASC data structure and it's construction in §8.4. We then demonstrate the use of CASC to represent a surface mesh and compute vertex tangents in §8.6.

8.2.1 Related Work

Although many data structures to represent simplicial complexes have been developed, to the best of our knowledge there currently exists no data structure which supports meshes of arbitrary dimension with user-selected typed data stored directly on each simplex. A full review of all existing data structures is beyond the scope of this work, however we highlight several representative examples. Many data structures such as the half-edge and doubly-connected edge list among others are restricted to the representation of two-manifolds only[15]. Other data structures such as SIG[16], IS[17], IA*[18], SimplexTree[19], AHF[20, 21], LincearCellComplex, and dD Triangulations[22] support or can be extended to represent arbitrary dimensional simplicial complexes. However, their current implementations either do not consider the storage of data beyond possibly embedding, or do not support inline storage of user data. LinearCell-Complex from CGAL supports only a linear geometrical embedding[23]. AHF implemented in MOAB uses separate arrays of data which are then referenced using a handle[20, 21]. In addition addition to the limitations of data storage, some make assumptions limiting their generality. dD Triangulations, for example, assumes that a simplicial complex is pure and therefore does not support the representation of mixed dimensional complexes[22].

8.3 Background – Abstract Simplicial Complexes

An Abstract Simplicial Complex (ASC) is a combinatorial structure which can be used to represent the connectivity of a simplicial mesh, independent of any geometric information. More formally, the definition of an ASC is as follows.

Definition 8.3.1. Given a vertex set V, an *abstract simplicial complex* $\mathscr{F}$ of V is a set of subsets of V with the following property: for every set $X \in \mathscr{F}$, every subset $Y \subset X$ is also a member of $\mathscr{F}$.

The sets $s \in \mathscr{F}$ are called a simplex or face of $\mathscr{F}$; similarly a face X is said to be a face of simplex

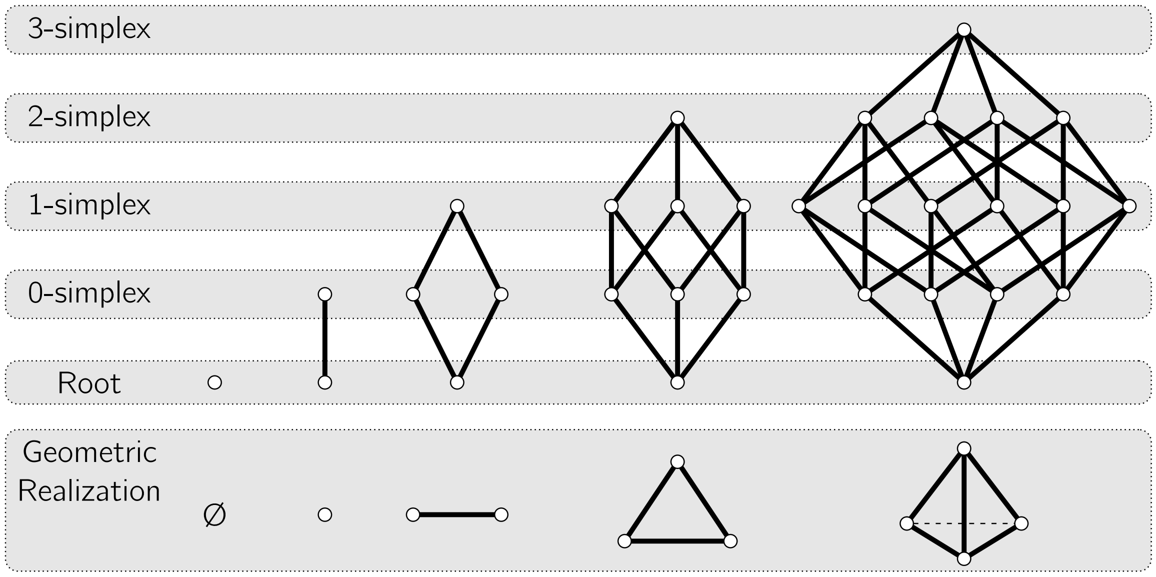


Figure 8.1: Hasse diagrams of several Abstract Simplicial Complexes and a geometric realization from left to right: the empty set, a vertex, an edge, a triangle, a tetrahedron.

s if $X \subset s$. Since X is a face of s, s is a coface of X. Each simplex has a dimension characterized by $\dim s = |s| - 1$, where $|s|$ is the cardinality of set s. A simplex of $\dim s = k$ is also called a k-simplex. The dimension of the complex, $\dim(\mathscr{F})$, is defined by the largest dimension of any member face. Simplices of the largest dimension, $\dim(\mathscr{F})$ are referred to as the facets of the complex.

If one simplex is a face of another, they are incident. Every face of a k-simplex s with dimension $(k-1)$ is called a boundary face while each incident face with dimension $(k+1)$ is a coboundary face. Two k-simplices, f and s are considered adjacent if they share a common boundary face, or coboundary face. The boundary of simplex s, ∂s, is the sum of the boundary faces.

Having introduced the concept of an ASC, we can also define several operations useful when dealing with ASCs. A subcomplex is a subset that is a simplicial complex itself. The Closure (Cl) of a simplex, f, or some set of simplices $F \subseteq \mathscr{F}$ is the smallest simplicial subcomplex of $\mathscr{F}$ that contains F:

$$\mathrm{Cl}(f) = \{s \in \mathscr{F} \mid s \subseteq f\}; \qquad \mathrm{Cl}(F) = \bigcup_{f \in F} \mathrm{Cl}(f) \quad \text{(closure)}. \tag{8.1}$$

It is often useful to consider the local neighborhood of a simplex. The Star (St) of a simplex f is the set of all simplices that contain f:

$$\mathrm{St}(f) = \{s \in \mathscr{F} \mid f \subseteq s\}; \qquad \mathrm{St}(F) = \bigcup_{f \in F} \mathrm{St}(f) \quad \text{(star)}. \tag{8.2}$$

The Link (Lk) of f consists of all faces of simplices in the closed star of f that do not intersect f:

$$\mathrm{Lk}(f) = \{s \in \mathrm{Cl} \circ \mathrm{St}(f) \mid s \cap f = \emptyset\} = \mathrm{Cl} \circ \mathrm{St}(f) - \mathrm{St} \circ \mathrm{Cl}(f) \quad \text{(link)}. \tag{8.3}$$

For some algorithms, it is often useful to iterate over the set of all vertices or edges etc. We use the following notation for the horizontal "level" of an abstract simplicial complex.

$$\mathrm{Lvl}_k(\mathscr{F}) = \big\{s \in \mathscr{F} \mid \dim s = k\big\} \tag{8.4}$$

A subcomplex which contains all simplices $s \in \mathscr{F}$ where $\dim(s) \leq k$ is the k-skeleton of $\mathscr{F}$:

$$\mathscr{F}_k = \mathrm{Cl} \circ \mathrm{Lvl}_k(\mathscr{F}) = \bigcup_{i \leq k} \mathrm{Lvl}_i(\mathscr{F}). \tag{8.5}$$

By Definition 8.3.1, an ASC forms a partially ordered set, or poset. Posets are frequently represented by a Hasse diagram, a directed acyclic graph, where nodes represent sets, and edges denote set membership. Several example simplicial complexes and their corresponding Hasse diagrams are shown in Fig. 8.1. Colloquially we will use *up* and *down* to refer to the boundary and coboundary of a simplex respectively. In Hasse diagrams, we follow a convention that simplices shown graphically on the same horizontal level have the same simplex dimension. Furthermore, simplices of greater dimension are drawn above lesser simplices.

8.4 Colored Abstract Simplicial Complex

In this section we introduce the CASC data structure and its implementation. For a given simplicial complex, each simplex is represented by a node (`asc_Node`) in the Hasse diagram, and defined by a set of keys corresponding to the vertices which comprise the simplex. Note that we use node to refer to objects

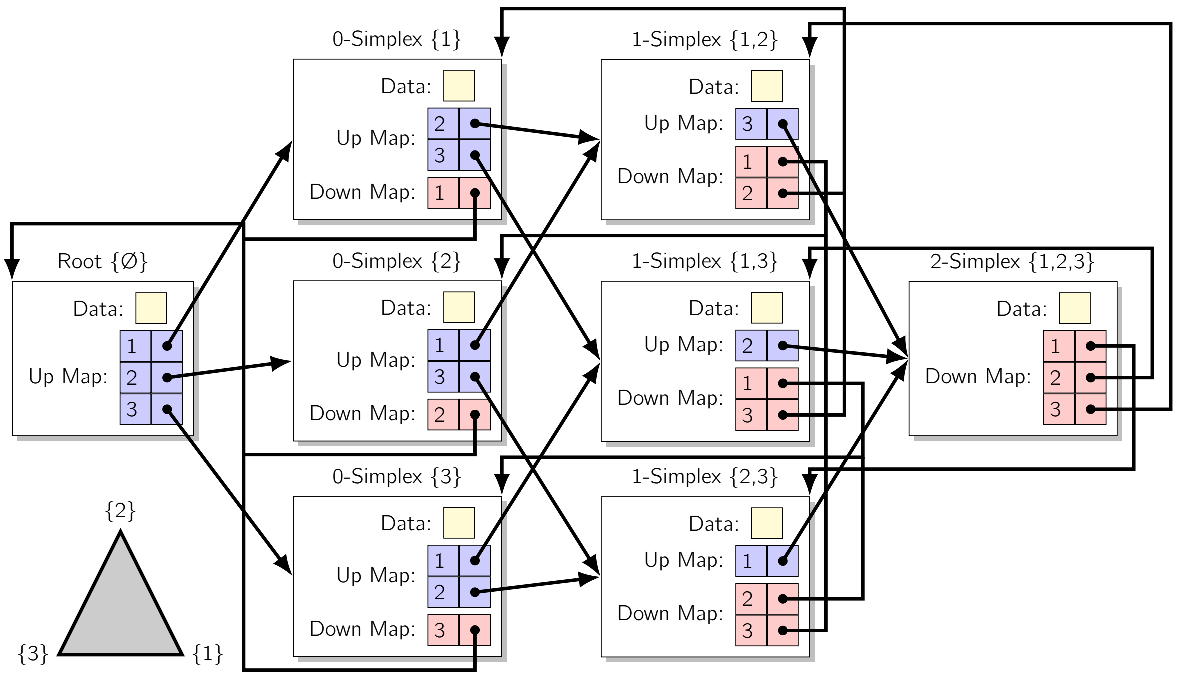


Figure 8.2: Data structure diagram of the resulting CASC for a triangle. Each simplex is represented as a node containing a dictionary up and/or down which maps the vertex index to a pointer to the next simplex. Data can be stored at each node with type determined at compile time. Effectively each level can contain different meta-data as defined by the user, separating the interactions of user data from the representation of topology.

in CASC Hasse diagram and not 0-simplices. Instead, 0-simplex are referred to as the vertices of the mesh. Furthermore we refer to the $\varnothing$-simplex or -1-simplex as the root simplex interchangeably. When a node is instantiated, we assign it a unique Integer Internal Identifier (iID) for use in the development of CASC algorithms. The iID is constant and never exposed to the end-user except for debugging purposes. Instead nodes can be referenced by the user using the `SimplexID` which acts as a convenience wrapper around an `asc_Node*`, providing additional support for move semantics for fast data access. All topological relations (i.e., edges of the Hasse diagram) are stored in each node as a dictionary which maps user specified keys to `SimplexID`s up and down. An example data structure diagram of triangle {1,2,3} is shown in Fig. 8.2. Based upon this example, if a user has the `SimplexID` of 1-simplex $\{1,2\}$ and wishes to get 2-simplex $\{1,2,3\}$, they can look in the `Up` dictionary of `SimplexID`$\{1,2\}$ for key 3 which maps to a `SimplexID`$\{1,2,3\}$. The vertices which constitute each simplex are not stored directly, but can be accessed by aggregating all keys in `Down`.

We note that while the representation of all topological relations is redundant and may not be memory optimal, it vastly simplifies the traversals across the complex. Furthermore, the associate algorithms and innovations using variable typing are general and thus compatible with other more condensed representations.

8.4.1 Variable Typing Per Simplex Dimension

We achieve coloring by allowing user-defined data to be stored at each node. The typical challenge for strongly typed languages such as C++ is that the types must be defined at compile time. Typical implementations would either hard code the type to be stored at each level or use a runtime generic type, such as `void*`. However, each of these have drawbacks. For the former, this requires writing a new node data structure for every simplicial complex we may wish to represent. For the latter, using `void*` adds an extra pointer dereference which defeats cache locality and may lead to code instability. Another possible implementation might be to require users of the library to derive their data types from a common class through inheritance. This solution puts an unnecessary burden on users who may have preexisting class libraries, or simply wish to store a built in type, such as an `int`. Furthermore, under the inheritance scheme, changes to the underlying container may require users to update their derived classes. To avoid this cumbersome step, we have employed the use of variadic templates introduced in C++11 to allow for unpacking and assignment of data types. The user specifies the types to be stored at each level in a list of templates to the object constructor, see Fig. 8.3.

The variadic templating allows CASC to represent complexes of any user-defined dimension. To specify an N-simplicial complex, CASC requires the definition of an index/key type followed by $N+1$ data types and N edge types. The first data type provided after the key type corresponds to data stored on the $\varnothing$-simplex which can be thought of as global metadata. For example, suppose we have a 2-simplicial complex intended for visualization and wish to store locations of vertices and colors of faces. A suitable ASC can be constructed using the following template command:

```
auto mesh = AbstractSimplicialComplex<int, void, Vertex, void, Color>()
```

If we now wish to represent a tetrahedral mesh, instead of constructing a new data structure, we can simply adjust the command:

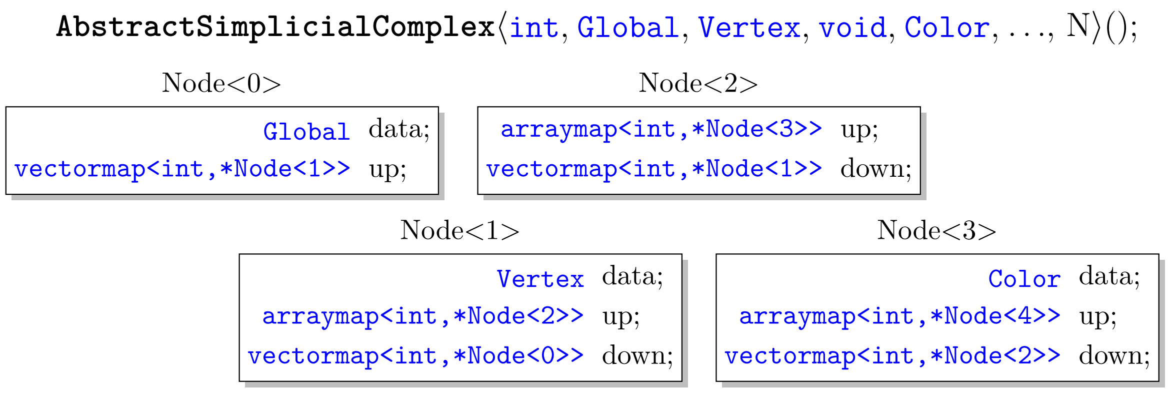

AbstractSimplicialComplex⟨int, Global, Vertex, void, Color, ..., N⟩();
Node<0>
Global data;
vectormap<int,*Node<1>> up;
Node<2>
arraymap<int,*Node<3>> up;
vectormap<int,*Node<1>> down;
Node<1>
Vertex data;
arraymap<int,*Node<2>> up;
vectormap<int,*Node<0>> down;
Node<3>
Color data;
arraymap<int,*Node<4>> up;
vectormap<int,*Node<2>> down;

Figure 8.3: Template arguments for CASC are unpacked and assigned to nodes accordingly. The first argument "`int`" is the key type for labeling vertices while the following arguments define node data types. Notably, `Node<2>` does not allocate memory for data as the corresponding template argument was "`void`". By passing in additional types, simplicial complexes of higher dimensions are instantiated.

```
auto mesh = AbstractSimplicialComplex<int, void, Vertex, void, Color, Density>()
```

In both cases, the first template argument is the key type for referring to vertices followed by the data type for each *k*-simplex. Supposing that the user does not wish to store data on any given level, by passing "`void`" as the template argument, the compiler will optimize the node data type and no memory will be allocated to store data. In both cases, the 0- and 1-simplices will have no data.

By using variadic templates, we allow the user to specify both the dimension of the simplicial complex as well as the types stored at each level. Because the type deduction is performed at compile time, there is no runtime performance impact on user codes. There is, however, some additional code complexity introduced. If the user wishes to retrieve the data stored in a simplicial complex, they must know what level they are accessing at compile time. A consequence is that the exposed identifier object, `SimplexID`, is templated on the integral level, so that types can be deduced. This does not present a problem for simple use cases, such as:

```
// Create alias for 2-simplicial complex
using ASC = AbstractSimplicialComplex<int,int,int,int>;
ASC mesh = ASC();        // construct the mesh object
mesh.insert<2>({1,2},5); // insert edge {1,2} with data 5
ASC::SimplexID<2> s = mesh.get_simplex<2>({1,2}); // get the edge
// NOTE: the type is templated on the integral level
```

```
std::cout << *s << std::endl; // prints data "5"
```

Listing 8.1: Example instantiation of 2-simplicial complex using CASC.

However, when implementing algorithms intended to be generic on any simplicial complex, templated code must be written. We discuss the implementation of several such algorithms in the following section.

8.5 Implemented Algorithms

The following algorithms are provided with the CASC library:

- Basic Operations
  - Creating and deleting simplices (`insert`/`remove`)
  - Searching and traversing topological relations (`GetSimplexUp`/`GetSimplexDown`)
- Traversals
  - By level (`get_level`)
  - By adjacency (`neighbors_up`/`neighbors_down`)
  - Traversals across multiple node types (Visitor Design Pattern/double dispatch)
- Complex Operations
  - Star/Closure/Link
  - Meta-Data Aware Decimation

8.5.1 Basic Operations

Creating and deleting simplices

Since the CASC data structure maintains every simplex in the complex and all topological relations, inserting a k-simplex, s, into the complex means ensuring the existence of, and possibly creating, $\mathscr{O}(2^k)$ nodes and $\mathscr{O}(k \cdot 2^{k-1})$ edges (see derivation in SI). Fortunately, the combinatorial nature of simplicial complexes allows this to be performed recursively. A generalized recursive insertion operation for any dimensional complex and user specified types, is described in Algorithm 8.1. The insertion algorithm defines an insertion order such that all dependent simplices exist in the complex prior to the insertion of the next simplex.

As an illustrative example of the template code used in this library, Algorithm 8.1 is rewritten in C++ template function-like pseudocode shown in Algorithm 8.6. While the templated code is more complicated, it provides many optimizations. For example, since the looping and recursion are performed at compile time, for any k-simplex we wish to insert, any modern compiler should optimize the code into a series of `insertNode()` calls; the `setupForLoop()` and `forLoop()` function calls can be completely eliminated. As a result, the optimized templated code will exhibit superior run time performance. To illustrate the insertion operation, a graphical representation of inserting tetrahedron {1,2,3,4} by Algorithm 8.6 is shown in Fig. 8.4, and step-by-step in Fig. S8.6. In the example, new simplex $root \cup v$ is added sequentially to the complex and any missing topological relations are found by traversing the faces of *root* and backfilling.

The removal of any simplex is also performed using a recursive template function. When removing simplex s, in order to maintain the property of being a simplicial complex, all cofaces of s or $f \in \mathrm{St}(s)$ must also be removed along with any boundary/coboundary relations of f. The implemented removal algorithm traverses up the complex and removes simplex s and all cofaces of s level by level.

ALGORITHM 8.1: Pseudocode description of the simplex insertion algorithm in CASC.

```
Input: keys[n]: Indices of n simplices to describe new simplex s
rootSimplex: The simplex to insert relative to (most commonly root)
Output: The new simplex s
Function insert(keys[n], rootSimplex)
{
    for (i = 0; i < n; i++)
    {
        newSimplex = createSimplex($rootSimplex \cup keys[i]$)
        if (i > 0) /* Recurse to insert sub-simplices                    */
        {
            /* Pass only the first part of keys up to index i           */
            return insert(keys[0:i], newSimplex)
        }
        else /* Terminal conditional                                     */
        {
            return newSimplex
        }
    }
}
```

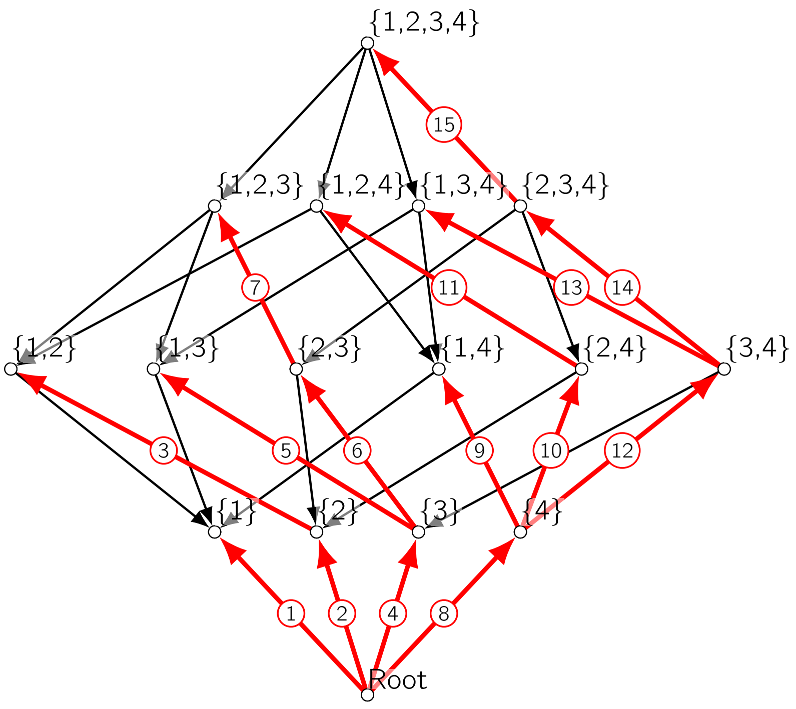


Figure 8.4: Recursive insertion of tetrahedral simplex {1,2,3,4}. The order of node insertions is represented by the numbered red arrows. When each node is created, the black arrows to parent simplices are created by backfilling.

Searching and traversing topological relations

The algorithms for retrieving a simplex as well as for basic traversals from one simplex to another across the data structure are the same. Given a starting simplex, and an array of keys up, the new simplex can be found recursively by Algorithm 8.2; The annotated code used is shown in §8.9.3. Traversals from one simplex to another require a key lookup followed by a pointer dereference and therefore occur in approximately constant time ($\mathscr{O}(1)$). Since all topological relations are stored, the traversal order across the array of keys does not matter. The same algorithm can be applied going down in dimension. For the retrieval of an arbitrary simplex, we start the search up from the root node of the complex.

ALGORITHM 8.2: Pseudocode for searching for a new simplex in CASC.

Input: *root*: the simplex to start the search from.
keys[*n*]: the relative name of the desired simplex *s*
Output: The simplex *s*

```
Function getSimplexUp(root, keys[n], index)
{
    if (index < n)
    {
        Find keys[index] in root.up /* Look for the edge up */
        if (Edge up is found)
        {
            return getSimplexUp(root ∪ keys[index], keys, index+1)
        }
        else /* Edge keys[index] doesn't exist */
        {
            return null pointer
        }
    }
    else {  return root  } /* Terminal condition */
}
```

8.5.2 Traversals

Thus far, we have presented algorithms for the creation of a simplicial complex as well as the basic traversal across faces and cofaces. For many applications, other traversals, such as by adjacency, may be more useful. We present several built-in traversal algorithms as well as the visitor design pattern for complicated operations.

By level

It is often useful to have a traversal over all simplices of the same level. For example, iterating across all vertices to compute a center of mass. To support this in an efficient fashion, simplices of the same dimension are stored in a level specific map of iIDs to node pointers. Notably, the map for each level is instantiated with the correct user specified node type with respect to level at compile time. To achieve this, we again use variadic templates to generate a tuple of maps, where each tuple element corresponds to the map for a specific level's node type.

Since `asc_node`s are templated on the integral level, we can use a template type map to map an integral sequence to the node pointer type,

$$tuple\langle 1,2,3,\ldots\rangle \xrightarrow{Node\langle k\rangle *} tuple\langle Node\langle 1\rangle *, Node\langle 2\rangle *, Node\langle 3\rangle *,\ldots\rangle,$$

producing a tuple of integrally typed node pointers. Subsequently, we can map again to generate a tuple of maps,

$$tuple\langle Node\langle 1\rangle *, Node\langle 2\rangle *,\ldots\rangle \xrightarrow{map<int,T>} tuple\langle map\langle int, Node\langle 1\rangle *\rangle, map\langle int, Node\langle 2\rangle *\rangle,\ldots\rangle.$$

By using this variadic template mapping strategy we now have the correct typenames assigned. Any level of the tuple can be accessed by getting the integral level using functions in the C++ standard library. Variations of this mapping strategy are also used to construct the `SimplexSet` and `SimplexMap` structures below.

For end users, the implementation details are entirely abstracted away. Continuing from the example above, iteration over all vertices of simplicial complex, `mesh`, can be performed using the provided iterator adaptors as follows.

```
// Deduces the type of nid = ASC::SimplexID<1>
for(auto nid : mesh.get_level_id<1>()){
        std::cout << nid << std::endl;
}
```

Listing 8.2: Example use of iterator adaptors for traversal across vertices of mesh.

The function `get_level_id<k>()` retrieves level k from the tuple and returns an iterable range across the corresponding map.

By adjacency

Many geometric algorithms operate on the local neighborhood of a given simplex. Unlike other data structures such as the halfedge, CASC does not store the notion of the next simplex. Instead, adjacency is identified by searching for simplices with shared faces or cofaces in the complex. The algorithm for finding neighbors with shared faces is shown in Algorithm 8.3. We note that the set of simplices with

ALGORITHM 8.3: Pseudocode to get the neighbors of a simplex, s, by inspecting the faces of s in CASC.

Input: s: The simplex to get the neighbors of.
Output: List of neighbors

```
Function getNeighborsUp($s$) /* Get the neighbors */
{
    Create an empty list of neighbors
    /* Follow all coboundary relations up from s. */
    for ($a \in s.up$)
    {
        SimplexID id = $s \cup a$
        for ($b \in id$) /* Go down from id */
        {
            if ($id \setminus b \neq s$) { Add $id \setminus b$ to neighbors } /* Do not add self to neighbors */
        }
    }
    return neighbors
}
```

shared faces may be different than the set of simplices with shared cofaces. Both adjacency definitions have been implemented and we leave it to the end user to select the relevant function. Once a neighbor list has been aggregated, it can be traversed using standard methods. While the additional adjacency lookup step is extra in comparison to other data structures, in many cases, the generation of neighbor lists need only be done once and cached. The trade off is that CASC offers facile manipulations of the topology without having to worry about reorganizing neighbor pointers.

Traversals over multiple node types.

When performing more complicated traversals, such as iterating over the star of a simplex, multiple node types may be encountered. In order to avoid typename comparison based branch statements, we have implemented visitor design pattern-based breadth first searches (BFS). The visitor design pattern refers to a double dispatch strategy where a traversal function takes a visitor functor which implements an overloaded `visit()` function. At each node visited, the traversal function will call `visit()` on the current node. Since the functor overloads `visit()` per node type, the compiler can deduce which visit function to call. Example pseudocode is shown in Listing 8.3. This double dispatch strategy, eliminates the need for extensive runtime typename comparisons, and enables easy traversals over multiple node types. We provide breadth first traversals up and down the complex from a set of simplices. These visitor traversals are used extensively in the complex operations described below.

```
template <typename Complex>
struct Visitor{
        template <std::size_t k>
        using Simplex = typename Complex::template SimplexID<k>;
        // General template prototype
        template <std::size_t level>
        bool visit(Complex& F, Simplex<level> s){
                return true;
        }
        // Specialization for vertices
        bool visit(Complex& F, Simplex<1> s){
                s.position = s.position*2;
                return true;
        }
        // Specialization for faces
        bool visit(Complex& F, Simplex<3> s){
                s.color = 'green';
                return true;
        }
};

void BFS(ASC& mesh, Visitor&& v){
        // ... traversal code
        v.visit(mesh, currentSimplex);
        // NOTE: visit is overloaded and called based on function prototype.
}

void main(){
        // ... define simplicial complex traits for a surface mesh
        ASC mesh = ASC(); // construct the mesh object
        // ... insert some simplices etc.
        BFS(mesh, Visitor());
        }
```

```
}
```

Listing 8.3: Example pseudocode of double dispatch to traverse the complex while scaling the mesh by 2 and coloring the faces green.

8.5.3 Complex Operations

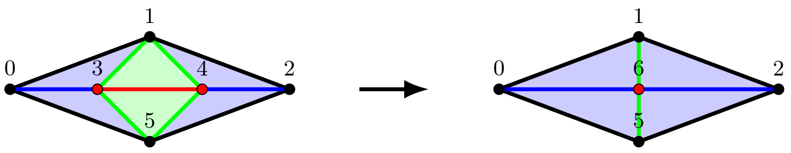
Star/Closure/Link

The star, link, and closure can be computed using the visitor breadth first traversals to collect simplices. These operations typically produce a set of simplices spanning multiple simplex dimensions, and thus simplex typenames, which cannot be stored in a traditional C++ set. We have implemented a multi-set data structure called the `SimplexSet`, which is effectively a tuple of typed sets corresponding to each level. The `SimplexSet` is constructed using the same mapping strategy as the tuple of maps used for the iteration across levels. For convenience, we provide functions for typical set operations such as insertion, removal, search, union, intersection, and difference. Using a combination of the star and closure functions with `SimplexSet` difference we can get the link by Eq. 8.3.

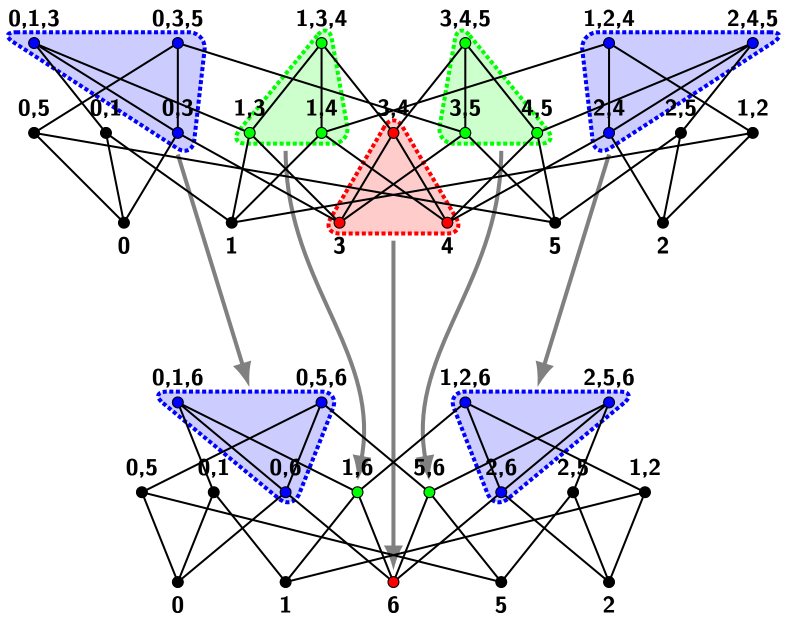
Meta-Data Aware Decimation

We have implemented a general decimation algorithm which operates by collapsing simplices of higher dimensions into a vertex. While edge collapses for 2-manifolds are well studied, a general dimensional collapse is useful for decimating higher-dimensional meshes used to solve PDEs such as those encountered in general relativity. Since simplices are being removed from the complex, user data may be lost. Our implementation is meta-data aware and allows the user to specify what data to keep post decimation. This is achieved by using a recursive algorithm to produce a map of removed simplices to new simplices. The user can use this mapping to define a function which maps the original stored data to the post decimation topology. This decimation strategy is implemented as an inplace operation yielding decimated mesh containing data mapped according to a user specified callback function.

This decimation algorithm is a generalization of an edge collapse operation to arbitrary dimensions. It is formally defined as follows:



(a) A geometric realization of the complex before and after decimation.



(b) Explicitly drawn out Hasse diagrams for the constructed example where the TOP is before, and BOTTOM is after decimation. Grey arrows mark the relationships between sets of simplices before and after. Because there is always a mapping, users can define strategies to manage the stored data.

Figure 8.5: The example decimation of edge $s = \{3,4\}$ in a constructed example.

Definition 8.5.1. Given simplicial complex $\mathscr{F}$, simplex to decimate $s \in \mathscr{F}$, vertex set V of $\mathscr{F}$, and new vertex $p \notin V$, we define function,

$$\varphi(f) = \begin{cases} f & \text{if } f \cap s = \varnothing \\ p \cup (f \setminus s) & \text{if } f \cap s \neq \varnothing \end{cases}, \tag{8.6}$$

where f is any simplex $f \in \mathscr{F}$. We define the decimation of $\mathscr{F}$ by replacing s with p as $\varphi(\mathscr{F})$.

Note that decimation under this definition is not guaranteed to preserve the topology, as can by seen by decimating any edge or face of a tetrahedron.

Decimation of a simplicial complex must result in a valid triangulation. Here we show that decimation by Definition 8.5.1 produces a valid abstract simplicial complex.

Theorem 8.5.1. *$\varphi(\mathscr{F})$ is an abstract simplicial complex.*

Proof. Given simplices y and x, let $y \in \varphi(\mathscr{F})$ and let $x \subset y$. We will show that $x \in \varphi(\mathscr{F})$. There are two cases for $y \cap p$, as they can be disjoint or intersecting.

Considering the disjoint case where $y \cap p = \varnothing$. This implies that $y \in \mathscr{F}$ and $y \cap s = \varnothing$. Since $\mathscr{F}$ is a simplicial complex, $y \in \mathscr{F}$ implies that $x \in \mathscr{F}$. Furthermore since $y \cap s = \varnothing$ and $x \subset y$, then $x \cap s = \varnothing$ implying that $\varphi(x) = x$ and thus $x \in \varphi(\mathscr{F})$.

Alternately in the intersecting case where $y \cap p \neq \varnothing$, that is $y \cap p = p$. Since $x \subset y$ there are two sub-cases where x either contains p or does not.

Supposing that $x \cap p = \varnothing$. Then $x \subset (f \setminus s)$ implying that $x \in \mathscr{F}$ and $x \cap s = \varnothing$. Therefore by the disjoint case, $\varphi(x) = x$ implies that $x \in \varphi(\mathscr{F})$.

Supposing that $x \cap p = p$. We can rewrite any such simplex x as $x = w \cup p$ where $w \cap p = \varnothing$. Furthermore we can write that $y = x \cup r$ where $x \cap r = \varnothing$ and $r \cap p = \varnothing$ and thus $y = w \cup p \cup r$. There exists some set q such that $f = w \cup r \cup q$ and $q \subseteq s$ such that $\varphi(f) = p \cup ((w \cup r \cup q) \setminus s) = y$ Since $f \in \mathscr{F}$ then $w \cup q \in \mathscr{F}$. Therefore $\varphi(w \cup q) = p \cup ((w \cup q) \setminus s) = p \cup w = x$ and thus $x \in \varphi(\mathscr{F})$.

For all cases and sub-cases we have shown that $x \in \varphi(\mathscr{F})$ therefore $\varphi(\mathscr{F})$ is an abstract simplicial complex. □

A pseudocode implementation for this decimation is provided in Algorithm 8.4. Given some simplicial complex $\mathscr{F}$ and simplex $s \in \mathscr{F}$ to decimate, this algorithm works in four steps. First, we compute the complete neighborhood, *nbhd* $= \mathrm{St}(\mathrm{Cl}(s))$, of s. Simplices not in the complete neighborhood will be invariant under φ and are ignored. Next, we use a nested set of breadth first searches to walk over the complete neighborhood and compute $p \cup f \setminus s$ for each simplex in the neighborhood. The results are inserted into a `SimplexMap` which maps $\varphi(f)$ to a `SimplexSet` of all f which map to $\varphi(f)$. Third, we iterate over the `SimplexMap` and run the user defined callback on each mapping to generate a list of

ALGORITHM 8.4: Pseudocode to decimate a simplex by collapsing it into a vertex in CASC.

Input: F: the simplicial complex;
s: the simplex to decimate;
clbk: a callback function to handle the data mapping
Output: Simplicial complex with simplex collapsed according to Def. 8.5.1

```
Simplex np = F.newVertex() /* Create a dummy new vertex to map to */
SimplexSet N /* For the complete neighborhood */
SimplexDataSet data /* Data structure to store (simplex name, simplex data) pairs */
SimplexMap simplexMap /* Data structure to store New Simplex -> SimplexSet map */

/* Get the complete neighborhood (all simplices which are associated with $s$). */
for (vertex $v$ of $s$)
{
    BFSup (simplex $i \in \mathrm{St}(v)$)
    {
        if ($j \notin N$) { N.insert (j) }
    }
}

/* Backup the complete neighborhood.  These simplices will be destroyed eventually. */
SimplexSet doomed = N
/* Generate the before-after mapping */
BFSdown (simplex $i \in \mathrm{Cl}(s)$) /* MainVisitor */
{
    BFSup (simplex $j \in N \cap \mathrm{St}(i)$) /* InnerVisitor */
    {
        /* $i$ maps to $np$ so we need to connect $j$ to $np$ instead */
        Name newName = $np \cup j \setminus i$
        SimplexSet grab
        BFSdown (simplex $k \in N \cap \mathrm{Cl}(j)$) /* GrabVisitor */
        {
            /* Grab dependent simplices which have not been grabbed yet.  Grabbed simplices will
             map to simplex newName */
            N.remove($k$)
            grab.insert($k$)
        }
        if (newName ∉ simplexMap) {  simplexMap.insert(pair(newName, grab))  }
        else {  simplexMap[newName].insert(grab)  }
    }
}

for ($\{newName, grabbed\} \in simplexMap$)
{
    /* Run the user's callback to map simplex data. */
    DataType mappedData = (*clbk)(F, name($j$), newName, grabbed)
    data.insert({newName, mappedData}) /* Insert a pair containing new simplex name and data */
}
/* Iterate over the complete neighborhood and remove simplices */
performRemoval(F, doomed)
/* Iterate over data and append mapped simplices and data */
performInsertion(F, data)
```

Table 8.1: Traversal order of the visitors for the decimation shown in Figure 8.5a.

Order	`MainVisitor`	`InnerVisitor`	`GrabVisitor`	Maps to
1	{3,4}	{3,4}	{3,4}, {3}, {4}	{6}
2	{3,4}	{1,3,4}	{1,3,4}, {1,3}, {1,4}	{1,6}
3	{3,4}	{3,4,5}	{3,4,5}, {3,5}, {4,5}	{3,6}
4	{3}	{0,3}	{0,3}	{0,6}
5	{3}	{0,1,3}	{0,1,3}	{0,1,6}
6	{3}	{0,3,5}	{0,3,5}	{0,5,6}
7	{4}	{2,4}	{3,5}	{5,6}
8	{4}	{1,2,4}	{1,2,4}	{1,2,6}
9	{4}	{2,4,5}	{2,4,5}	{2,5,6}

new simplices and associated mapped data stored in `SimplexDataSet`. Finally, the algorithm removes all simplices in the complete neighborhood and inserts the new mapped simplices.

An example application of this decimation operation is shown in Figure 8.5. In Figure 8.5a we show the geometric realization of the complex before and after the decimation of simplex {3,4}. In Figure 8.5b we show the detailed Hasse diagrams for the constructed example. Note that there are two possible mapping situations. In one case, $f \in \mathrm{St}(\mathrm{Cl}(s)) \cap \mathrm{Cl}(\mathrm{St}(s))$, groups of simplices are merged. In the other case, simplices $f \in \mathrm{Cl}(\mathrm{St}(s)) \setminus \mathrm{St}(\mathrm{Cl}(s))$ only need to be reconnected to the new merged simplices. By carefully choosing the traversal order, some optimizations can be made.

We apply the decimation on the constructed example shown in Figure 8.5 and show the order of operations with respect to the current visited simplex for each visitor function in Table 8.1. Starting out, `MainVisitor` and `InnerVisitor` will visit {3,4}. At this point, `GrabVisitor` will search BFS down from {3,4} to grab the set {{3,4}, {3}, {4}} and remove it from the neighborhood, eliminating some future calculations at {3} and {4}. All simplices in this set will map to new simplex {6} after decimation. Continuing upwards, `InnerVisitor` will find simplex {1,3,4} and `GrabVisitor` will grab set {{1,3,4}, {1,3}, {1,4}}. Again this set of simplices will map to common simplex {1,6} post decimation. A similar case occurs with simplex {3,4,5}. At this point, all simplices in $\mathrm{St}(\mathrm{Cl}(s)) \cap \mathrm{Cl}(\mathrm{St}(s))$ have been visited and removed from the neighborhood and `MainVisitor` continues BFS down and finds {3} and calls BFS up (`InnerVisitor`). Note that since simplex {3} has already been grabbed, `InnerVisitor` will continue upwards and find {0,3}. Looking down there are no simplices which are faces of {0,3} in the neighborhood. So on and so forth.

To reiterate, `GrabVisitor` grabs the set of simplices which will be mapped to a common simplex. We show here that the order in which simplices are grabbed by Algorithm 8.4 will preserve that all simplices $f = w \cup q$ where $q \subseteq s$ will map to $\varphi(f) = w \cup p$.

When visiting any simplex $f \supsetneq q$ where $q \subseteq s$ and q corresponds to simplices visited by `MainVisitor`. We can write f as $f = w \cup q$ where the sets w and q are disjoint. Looking down from f all simplices fall into two cases: $g = v \cup q$ where $\varnothing \subseteq v \subsetneq w$ or $h = w \cup t$ where $t \subsetneq q$. All simplices of form g, at worst case, will been grabbed while `InnerVisitor` proceeded BFS up from q. Remaining simplices h can be grouped with f and correctly mapped to $w \cup p$.

We note that in some non-manifold cases `GrabVisitor` will not always grab set members in one visit. Supposing that we removed simplex {1,3,4} from the constructed example, in this case, `InnerVisitor` cannot visit {1,3,4} and simplices {1,3} and {1,4} will not be grouped. Instead {1,3} and {1,4} will be found individually when `MainVisitor` visits {3} then {4}. To catch this case and correctly map {1,3} and {1,4} to {1,6}, we use a `SimplexMap` to aggregate all maps prior to proceeding. We note that in all cases starting with a valid simplicial complex, this implementation of the general collapse of simplex s visits each member in $\mathrm{St}(\mathrm{Cl}(s))$ and maps according to Def. 8.5.1 producing a valid simplicial complex. There is no guarantee that the result will have the same topological type as the pre-decimated mesh. The preservation of the topological type under decimation is often a desirable trait. We will show how to verify the Link Condition for when edge collapse of a 2-manifold will preserve the topological type in §8.6.1.

8.6 Surface Mesh Application Example

CASC is a general simplicial complex data structure which is suitable for use in mesh manipulation and processing. For example, we can use CASC as the underlying representation for an orientable surface mesh. Using a predefined `Vertex` class which is wrapped around a tensor library, and a class `Orientable` which wraps an integer, we can easily create a surface mesh embedded in $\mathbb{R}^3$.

```
using Vector = tensor<double,3,1>;
struct Vertex {
        Vector position;
        // ... other helpful vertex functions;
};
struct Orientable {
        int orientation;
};
struct complex_traits
{
    using KeyType = int;
    using NodeTypes = util::type_holder<void,Vertex,void,Orientable>;
    using EdgeTypes = util::type_holder<Orientable,Orientable,Orientable>;
};
using SurfaceMesh = simplicial_complex<complex_traits>;
```

In this case, 1-simplices will store a `Vertex` type while faces and all edges will store `Orientable`. Using `SurfaceMesh` we can easily create functions to load or write common mesh filetypes such as OFF as shown in the included library examples.

ALGORITHM 8.5: Pseudocode to define the orientation of a topological relation in CASC.

Input: Simplices a and b where $b = a \cup v$ and v is a vertex.
Output: The orientation of edge $a \to b$

```
int orient = 1
for (Vertex u : a) /* For each vertex u in simplex a */
{
    if (v>a) { orient *= -1; }
}
return orient;
```

We can define a boundary morphism which applies on an ordered k-simplex,

$$\partial_i^k([a_0,\ldots,a_{k-1}]) = (-1)^i([a_0,\ldots,a_{k-1}] \setminus \{a_i\}), \tag{8.7}$$

where $a_i < a_{i+1}$. Using Algorithm 8.5, we can apply this morphism to assign a ± 1 orientation to each topological relation in the complex. Subsequently, for orientable manifolds, we can compute orientations of faces f_1 and f_2 which share edge e such that,

$$\text{Orient}(e_1) \cdot \text{Orient}(f_1) + \text{Orient}(e_2) \cdot \text{Orient}(f_2) = 0, \tag{8.8}$$

where e_1 and e_2 correspond to the edge up from e to f_1 and f_2 respectively. Doing so, we create an oriented simplicial complex.

Supposing that we wish to compute the tangent of a vertex as defined by the weighted average tangent of incident faces. This is equivalent to computing the oriented wedge products of each incident face. This can be written generally as,

$$\begin{aligned}\text{Tangent}(v) &= \frac{1}{N}\sum_{i=0}^{N} \text{Orient}(f_i) \cdot (\partial_j(f_i) \wedge \partial_k(f_i)) &(8.9)\\ &= \frac{1}{N}\sum_{i=0}^{N} \text{Orient}(f_i) \cdot \frac{1}{2}(e_{i,j} \otimes e_{i,k} - e_{i,k} \otimes e_{i,j}), &(8.10)\end{aligned}$$

where N is the number of incident faces, f_i is incident face i, j and k are indices of vertex members of f_i not equal to v, and $e_{i,j} = \partial_j(f_i)$. This can be easily computed using a templated recursive function.

```
// Terminal case
auto getTangentH(const SurfaceMesh& mesh,
                const Vector& origin,
                SurfaceMesh::SimplexID<SurfaceMesh::topLevel> curr){
    return (*curr).orientation;
}


template <std::size_t level, std::size_t dimension>
auto getTangentH(const SurfaceMesh& mesh,
                const Vector& origin,
                SurfaceMesh::SimplexID<level> curr){
    tensor<double, 3, SurfaceMesh::topLevel - level> rval;
    auto cover = mesh.get_cover(curr); // Lookup coboundary relations
    for(auto v : cover){
        auto edge = *mesh.get_edge_up(curr, v); // Get the edge object
        const auto& vtx = (*mesh.get_simplex_up({v})).position; // Vertex v
        auto next = mesh.get_simplex_up(curr,v); // Simplex curr union v
        rval += edge.orientation * (v-origin) * getTangentH(mesh, origin, next);
    }
    return rval/cover.size();
}
```

This demonstrates the ease using the CASC library as an underlying simplicial complex representation. Using the provided API, it is easy to traverse the complex to perform any computations.

8.6.1 Preservation of Topology Type of Surface Mesh Under Edge Decimation by Contraction

Supposing we wish to decimate a surface mesh by edge contraction under Def. 8.5.1 without changing the topology of the complex. This can be verified by checking the Link Condition, defined with proof from Edelsbrunner[24], stating,

Lemma 8.6.1. Let F be a triangulation of a 2-manifold. The contraction of $ab \in F$ preserves the topological type if and only if $\mathrm{Lk}(a) \cap \mathrm{Lk}(b) = \mathrm{Lk}(ab)$.

Revisiting the example from Fig. 8.5a, we can construct the topology and check the Link Condition using operations supported by the CASC library.

```
// Construct the topology
SurfaceMesh mesh;
mesh.insert({0,1,3});
mesh.insert({0,3,5});
mesh.insert({1,3,4});
mesh.insert({3,4,5});
mesh.insert({1,2,4});
mesh.insert({2,4,5});

// Get simplices and edge
SimplexSet<SurfaceMesh> A,B,AB, AcapB;
auto ab = mesh.get_simplex_up({3,4});
auto a = mesh.get_simplex_up({3});
auto b = mesh.get_simplex_up({4});

// Compute the links of each
getLink(mesh, a, A);
getLink(mesh, b, B);
getLink(mesh, ab, AB);

// Link(a): Simplices {1}, {4}, {5}, {1,4}, {0,1}, {0,5}, {4,5}
std::cout << A << std::endl;
// Link(b): Simplices {5}, {2}, {3}, {1}, {3,5}, {1,3}, {1,2}, {2,5}
std::cout << B << std::endl;
// Link(AB): Simplices {1}, {5}
std::cout << AB << std::endl;

set_intersection(A, B, AcapB);
// Link(a) \cap Link(b): Simplices {1}, {5}
std::cout << AcapB << std::endl;

// Check the Link Condition
std::cout << (ab == acapb) << std::endl; // Evaluates to true
```

The `getLink()` function utilizes a series of visitor breadth first traversals down then up the complex, collecting the set of simplex into a `SimplexSet` object. Then using set operations provided by `SimplexSet` the set difference and equality comparison are performed. This example highlights the simplicity, clarity, and transparency of using the CASC library.

8.7 Conclusions

CASC provides a general simplicial complex data structure which allows the storage of user defined types at each simplex level. The library comes with a full-featured API providing common simplicial complex operations, as well as support for complex traversals using a visitor. We also provide a meta-data aware decimation algorithm which allows users to collapse simplices of any dimension while preserving data according to a user defined mapping function. Our implementation of CASC using a strongly-typed language is only possible due to recent innovations in language tools. The CASC API abstracts away most of the complicated templating, allowing it to be both modern and easy to use. We anticipate that CASC will not only be of use for the ET community but microscopy and modeling as a whole along with other fields like applied mathematics and CAD.

One limitation is the ease of extending to other languages. The generality of CASC is reliant upon the C++ compiler. Sacrificing this, specific realizations of CASC can be wrapped using tools like SWIG for use in other languages. Another limitation of CASC is the memory efficiency. The current Hasse diagram based implementation was selected for the sake of transparency, ease of traversal, and manipulation. Optimizations to the memory efficiency of CASC can be made by employing more compact representations. The variadic template approach we use to attach user data to simplices is compatible with data structures which explicitly represent all simplices but only a subset of topological relations. This includes data structures such as SimplexTree[19], IS[17], and SIG[16] among others. Other compressed data structures which skip levels of low importance using implicit nodes are not compatible with the current CASC implementation. Skipped levels would need to be implemented as exceptions to the combinatorial variadic rules. Similarly, although CASC in its current form is restricted to the representation of simplicial complexes, the combinatorial strategy can be easily adapted to support other regular polytopes by changing the boundary relation storage rules. In the future we hope to incorporate parallelism into the CASC library. A copy of CASC along with online documentation can found on GitHub https://github.com/ctlee/casc.

8.8 Acknowledgements

Chapter 8, in full, has been submitted for publication and appears as it may appear in “Lee, C. T.†; Moody, J. B.†; Amaro, R. E.; McCammon, J. A.; Holst, M. *The Implementation of the Colored Abstract Simplicial Complex and Its Application to Mesh Generation.* Trans. Math. Soft.” The dissertation author was a primary coinvestigator and author of this work.

This work is supported by the National Institutes of Health under grant numbers P41-GM103426 and RO1-GM31749. CTL also acknowledges support from the NIH Molecular Biophysics Training Grant T32-GM008326. MJH was supported in part by the National Science Foundation under awards

DMS-CM1620366 and DMS-FRG1262982.

8.9 Supplementary Information

8.9.1 Derivation of Worse Case Insertion Performance

The number of combinations at each level l for a k-simplex is given by the binomial theorem, $\binom{k}{l}$. In the worst case, where the simplicial complex is empty, the total number of nodes created for inserting a k-simplex is $\sum_{l=0}^{k}\binom{k}{l} = 2^k$. The number of edges to represent all topological relations is then given by $\sum_{l=0}^{k} l\binom{k}{l} = k \cdot 2^{k-1}$.

8.9.2 Templated Insertion Algorithm

Pseudocode for the algorithms presented in this manuscript have been vastly simplified in order to facilitate understanding. For example Algorithm 8.1, while the non-templated version is appears straightforward, it is impossible to be implemented in C++ directly, due to several typing related issues. First, the function prototype for `insert()` requires the `rootSimplex` as the second argument. Simplices at different levels have different types and `insert()` must be overloaded. Similarly the variable `newSimplex` and function `createSimplex()` must know the type of simplex which will be created at compile time.

The actual implementation uses variadic templates to resolve the typing issues. As an example, templated pseudocode for simplex insertion (Algorithm 8.1) is shown in Algorithm 8.6. Not only does the templated code automatically build the correct overloaded functions, but it provides many optimizations.

The step-by-step insertion of tetrahedron {1,2,3,4} is shown in Figure 8.6. Numbered red lines correspond to `newNode` and `root` in function `insertNode()`. Skinny black lines are the topological relations inserted by `backfill()`.

ALGORITHM 8.6: Templated pseudocode implementation of Algorithm 8.1 in CASC.

Input: *keys*[n]: Indices of n simplices to describe new simplex s,
$\mathscr{F}$: simplicial complex
Output: The new simplex s

```
/* User function to insert simplex {keys}                                        */
Function insert<n>(keys[n]){
    return setupForLoop<0, n>(root, keys) /* 'root' is the root node              */
}

// The following are private library functions...
/* Array slice operation. Algorithm 8.1: keys[0:i]                                */
Function setupForLoop<level, n>(root, keys){ /* General template                  */
    return forLoop<level, n, n>(root, keys) /* Setup the recursive for loop     */
}
Function setupForLoop<level, 0>(root, keys){ /* Terminal condition $n = 0$         */
    return root
}

/* Templated for loop. Algorithm 8.1: for (i = 0; i < n; i++)                     */
Function forLoop<level, antistep, n>(root, keys){ /* For loop for antistep        */
    /* $n - antistep$ defines next key to add to root                              */
    insertNode<level, n-antistep>(root, keys)
    return forLoop<level, antistep-1, n>(root, keys)
}
Function forLoop<level, 1, n>(root, keys){ /* Stop when antistep = 1             */
    return insertNode<level, n-1>(root, keys)
}

Function insertNode<level, n>(root, keys){ /* Insert a new node                   */
    v = keys[n]
    if ($root \cup v \in \mathscr{F}$){
        newNode= root.up[v]
    }
    else{ /* Add simplex $root \cup v$                                               */
        newNode = createSimplex<n>() /* Create a new node, n-simplex newNode     */
        newNode.down[v] = root /* Connect boundary relation                  */
        root.up[v] = newNode /* Connect coboundary relation                  */
        backfill (root, newNode, v)/* Backfill other topological relations     */
    }
    /* Recurse to insert any cofaces of newNode. Algorithm 8.1: insert(keys[0:i],
       newSimplex)                                                                 */
    return setupForLoop<level+1, n>(newNode, keys)
}

Function backfill<level>(root, newNode, value){ /* Backfilling pointers to other parents */
    for (currentNode in root.down){
        childNode = currentNode.up[value] /* Get simplex $currentNode \cup value$       */
        newNode.down[value] = child /* Connect boundary relation              */
        child.up[value] = newNode /* Connect coboundary relation              */
    }
}
```

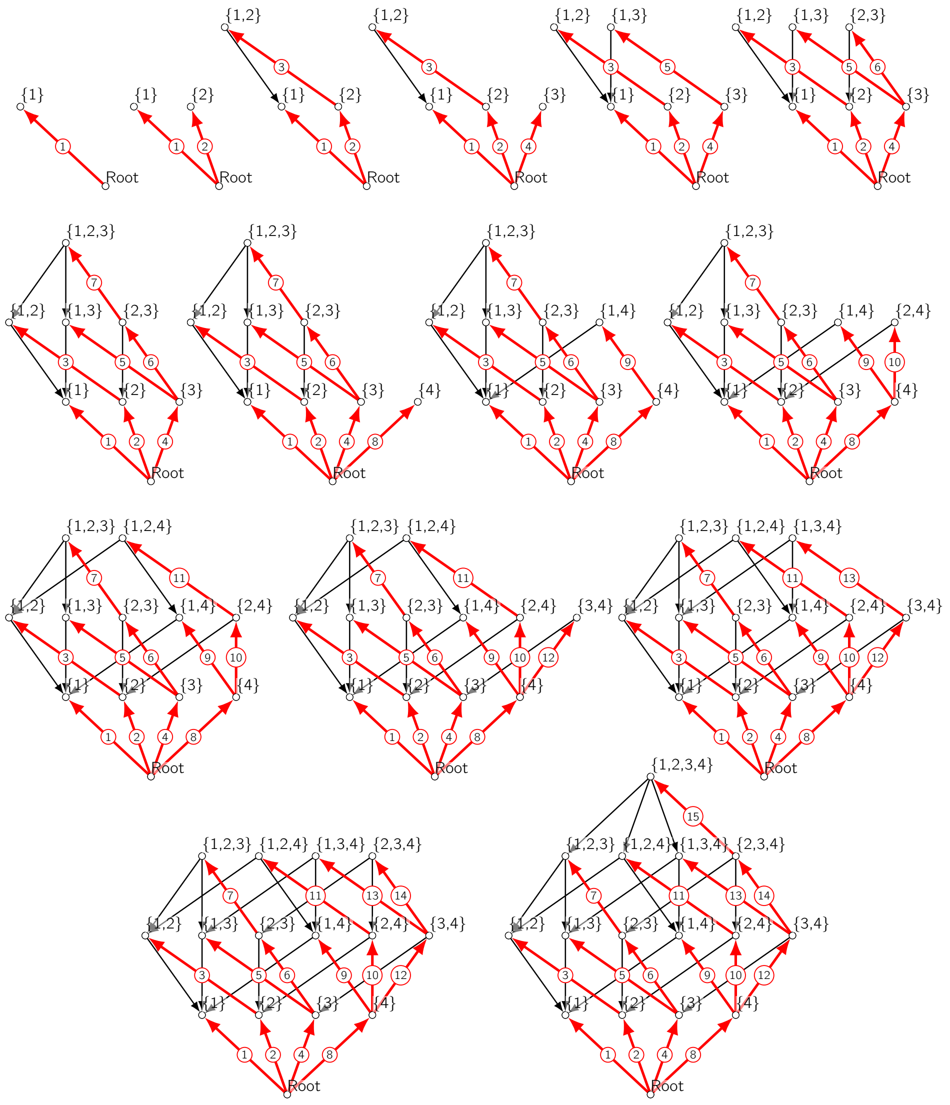


Figure 8.6: The Hasse diagrams for the step-by-step insertion of tetrahedron {1,2,3,4} by Algorithm 8.1. Red lines represent the order of creation for each simplex. The skinny black lines represent where connections to parent simplices are backfilled.

8.9.3 Code for Getting Neighbors by Adjacency

The C++ code for collecting the set of *k*-simplices sharing a common coface with simplex *s*. A function call to `neighbors_up()` calls the following code which serves primarily to help the compiler deduce the dimension, *k*, of *s*.

```
template <class Complex, class SimplexID, class InsertIter>
void neighbors_up(Complex &F, SimplexID s, InsertIter iter)
{
    neighbors_up<Complex, SimplexID::level, InsertIter>(F, s, iter);
}
```

With the simplex dimension determined, we call an overloaded function which defines the operation for a *k*-simplex.

```
template <class Complex, std::size_t level, class InsertIter>
void neighbors_up(
        Complex &F,
        typename Complex::template SimplexID<level> s,
        InsertIter iter)
{
    for (auto a : F.get_cover(s))
    {
        auto id = F.get_simplex_up(s, a);
        for (auto b : F.get_name(id))
        {
            auto nbor = F.get_simplex_down(id, b);
            if (nbor != s)
            {
                *iter++ = nbor;
            }
        }
    }
}
```

Neighbors of *s* are pushed into an insert iterator provided by the user. In this fashion, depending upon the container type the iterator corresponds to, the user can specify whether or not duplicate simplices are returned (`std::vector`) or not (`std::set`).

References

(1) Zhang, Y., *Geometric Modeling and Mesh Generation from Scanned Images*; Chapman & Hall/CRC Mathematical and Computational Imaging Sciences Series; Chapman and Hall/CRC: 2016.

(2) Roberts, E. *Curr. Opin. Struct. Biol.* **2014**, *25*, 86–91.

(3) Yu, Z.; Holst, M. J.; Hayashi, T.; Bajaj, C. L.; Ellisman, M. H.; McCammon, J. A.; Hoshijima, M. *J. Struct. Biol.* **2008**, *164*, 304–313.

(4) Gao, Z.; Yu, Z.; Holst, M. *Graph. Models* **2013**, *75*, 23–38.

(5) Yu, Z.; Holst, M. J.; Andrew McCammon, J. *Finite Elem. Anal. Des.* **2008**, *44*, 715–723.

(6) Si, H. *ACM Trans. Math. Softw.* **2015**, *41*, 1–36.

(7) Babuška, I.; Aziz, A. K. *SIAM J. Numer. Anal.* **1976**, *13*, 214–226.

(8) Liu, A.; Joe, B. *BIT* **1994**, *34*, 268–287.

(9) Bank, R. E.; Sherman, A. H.; Weiser, A. In *Sci. Comput. Appl. Methematics Comput. to Phys. Sci.* Stepleman, R. S., Ed.; North-Holland: 1983, pp 3–17.

(10) Bank, R. E.; Xu, J. *Numer. Math.* **1996**, *73*, 1–36.

(11) Holst, M.; Sarbach, O.; Tiglio, M.; Vallisneri, M. *Bull. Amer. Math. Soc.* **2016**, *53*, 513–554.

(12) Regge, T. *Nuovo Cim.* **1961**, *19*, 558–571.

(13) Holst, M.; Tiee, C. *Journal of Computational Mathematics* **2018**, *36*, 792–832.

(14) Vanden-Eijnden, E.; Venturoli, M. *J. Chem. Phys.* **2009**, *130*, 194101.

(15) De Floriani, L.; Hui, A. In *Proc. Third Eurographics Symp. Geom. Process.* Eurographics Association: Vienna, 2005, p 119.

(16) De Floriani, L.; Greenfieldboyce, D.; Hui, A. In *Proc. 2004 Eurographics/ACM SIGGRAPH Symp. Geom. Process. - SGP '04*, ACM Press: New York, New York, USA, 2004, p 83.

(17) De Floriani, L.; Hui, A.; Panozzo, D.; Canino, D. In *Proc. 19th Int. Meshing Roundtable*; Springer Berlin Heidelberg: Berlin, Heidelberg, 2010, pp 403–420.

(18) Canino, D.; De Floriani, L.; Weiss, K. *Comput. Graph.* **2011**, *35*, 747–753.

(19) Boissonnat, J.-D.; Maria, C. *Algorithmica* **2014**, *70*, 406–427.

(20) Dyedov, V.; Ray, N.; Einstein, D.; Jiao, X.; Tautges, T. J. *Eng. Comput.* **2015**, *31*, 389–404.

(21) Ray, N.; Grindeanu, I.; Zhao, X.; Mahadevan, V.; Jiao, X. *Procedia Eng.* **2015**, *124*, 291–303.

(22) Boissonnat, J.-D.; Devillers, O.; Hornus, S. In *Annu. Symp. Comput. Geom.* Aarhus, Denmark, 2009, pp 208–216.

(23) CGAL, Computational Geometry Algorithms Library.

(24) Edelsbrunner, H., *Geometry and Topology for Mesh Generation*; Cambridge University Press: Cambridge, 2001.