SBMOpenMM: A builder of Structure-Based Models for OpenMM

Martin Floor,[a,b,‡] Kengjie Li,[d] Miquel Estévez-Gay,[e] Luis Agulló,[c] Pau Marc Muñoz,[a] Jenn K. Hwang,[d] Sílvia Osuna,[e,f] Jordi Villà-Freixa[a,b]

[a]Department of Basic Sciences, Faculty of Medicine and Health Sciences, Universitat Internacional de Catalunya, 08195 Sant Cugat del Vallès, Spain
[b]Department of Biosciences, Faculty of Sciences and Technology, Universitat de Vic-Universitat Central de Catalunya, 08500 Vic, Spain
[c]Faculty of Medicine, Universitat de Vic - Universitat Central de Catalunya, 08500 Vic, Spain
[d]Warshel Institute of Computational Biology, The Chinese University of Hong Kong, 518172 Shenzhen, China
[e]CompBioLab group, Institut de Química Computacional i Catàlisi (IQCC) and Departament de Química, Universitat de Girona, 17071 Girona, Spain
[f]ICREA, Pg. Lluís Companys 23, 08010 Barcelona, Spain

[‡]Corresponding author. Email: mfloor@uic.es

Abstract

Molecular dynamics (MD) simulations have become a standard tool to correlate the structure and function of biomolecules, and significant advances have been made in the study of proteins and their complexes. A major drawback of conventional MD simulations is the difficulty and cost of obtaining converged results, especially when exploring potential energy surfaces containing considerable energy barriers. This limits the wide use of MD calculations to determine the thermodynamic properties of biomolecular processes. Indeed, this is true when one considers the conformational entropy of such processes, which is ultimately a critical component in assessing the simulations' convergence. Alternatively, a wide range of Structure-Based Models (SBMs) has been used in the literature to unravel the basic mechanisms of biomolecular dynamics. These models introduce simplifications that focus on the relevant aspects of the physical process under study. Because of this, SBMs incorporate the need to modify force field definition and parameters to target specific biophysical simulations. Here we introduce SBMOpenMM, a Python library to build force fields for SBMs, that uses the OpenMM framework to create and run SBM simulations. The code is flexible, user-friendly, and profits from the high customizability and performance provided by the OpenMM platform.

Background

Proteins are structurally fluctuating macromolecular systems that perform the vast majority of functions of biological cells. Protein dynamics have different timescale manifestations related to various physical events, ranging from fast local flexibility to slower collective motions.[1] The movements of protein atoms ultimately govern the kinetics and thermodynamics of folding, binding, catalysis, among other biochemical activities.[2–4]

Molecular dynamics (MD) has become an essential tool for the computational study of biological polymers. Detailed physical observations can be directly obtained that otherwise are not directly accessible by traditional wet-lab experimental studies. Nonetheless, the computational cost for running MD simulations still hinders its application for many relevant processes with significant activation barriers. Many efforts have been devoted to enhance conformational space sampling by improving searching algorithms,[5] simplifying force fields (for a recent review, see reference [6]), and accelerating algorithm execution through code optimization and parallelization.[7]

Among several MD techniques, dating back to the pioneering work of Warshel and Levitt,[8,9] Structure-Based Models (SBMs) emerged as a simplified methodology for studying protein folding dynamics.[10,11] SBMs are based on simplifying assumptions rooted in energy landscape theory for protein folding that capture essential physical aspects of natural proteins and bridge the gap between physical complexity and computational efficiency. These simplifications allow us to obtain a reasonable kinetic and equilibrium characterization of protein systems to be studied within the framework of statistical mechanics.

SBMs idealize the energy landscape based on two main ideas. First, natural proteins have evolved to avoid kinetic traps along the folding coordinate; this prevents the formation of long-lived folding intermediaries containing non-native interactions that could frustrate the folding reaction. Secondly, protein evolution has maximized the energy gap between the natively folded state and other competing configurations; these focused simplifications increase the number of protein molecules that populate the native basin and remove the excess of kinetic frustration.[12,13] SBMs exploit these facts by focusing on the native structure geometry and contact information, exploring a conformational landscape entirely based on the native topology constraints. In this way, one can separate energetic from topological contributions, showing that essential features of protein folding are mainly encoded by the organization of the folded state.[14] Following the same concept, SBM simulations have also been used to study protein-protein binding mechanisms or ligand-induced conformational changes. Additionally, multi-basin SBM potentials have served to study other complex mechanisms in protein dynamics, such as folding to competing native configurations, conformational shifts, domain swapping, knotting, or drastic structural rearrangements between functional and dysfunctional conformations. For a review on SBM applications, see reference [15], and for a review on the evolution of SBM models and their varied types of implementations for specific biophysical problems, see reference [16].

Easy customization of force field terms and parameters is typically a challenge for molecular simulation packages. Because SBM is an ongoing practical and meaningful research methodology, it would greatly benefit from a fast, accessible, flexible, and easy-to-set-up implementation that facilitates simulation and force field experimentation. To this end, we have developed SBMOpenMM, a Python library, to set up custom SBMs using the OpenMM toolkit framework for MD simulations.[17] The Python code is flexible, user-friendly, and profits from the high customizability and performance provided by the OpenMM platform. We hope this will foster further research in the field in an open-source collaborative manner.

We validated the library by running several simulations to show that the derived dynamic profiles match the structures experimental crystallographic temperature factors. Additionally, we exemplify its use simulating the two-step folding process of an 87-residues small protein.

Implementation

The SBMOpenMM software addresses the problem of compiling custom force fields for studying protein systems using SBM simulations. Currently, the program has built-in default models from popular SBM implementations, including single- and multi-basin approaches. However, the program structure serves as a base to construct further SBM implementations by providing an open-source library in which new force field terms can be easily added and adapted to tackle different biophysical questions.

SBMOpenMM uses two files as input: a Protein Data Bank (PDB) format file for the protein structure and a contact file for specifying the native contact interactions (see Section SI I.A for detail on how to obtain this file). The topology and the set of bonded interactions are defined from the structure's connectivity. These include definitions of atoms participating in bonds, angles, and proper, improper, and planar torsions, together with their equilibrium distances. Likewise, nonbonded interactions are defined from the atomic pairs listed in the contact file, and their equilibrium distances are calculated from the input structure. This connectivity information, together with the set of force constant parameters, is used by the library to construct the relevant forces and system objects, directly employed by the OpenMM MD engine to start a simulation.[17]

The SBMOpenMM library contains methods to define the SBM force field composition and set default or custom models and parameters. To this end, the Python library is composed of three major classes (see also Figure S1):

- ➔ *`geometry`*
- ➔ *`system`*
- ➔ *`models`*

The first class, *`geometry`*, contains methods for calculating geometrical parameters of the input structure, including equilibrium distances of contacts, bonds, angles, and torsional degrees of freedom.

The *`system`* class contains all the SBM system definitions, including the connectivity, initial coordinates, force field parameters, and the OpenMM force objects. Several attributes and methods inside this class allow customizing the forces and parameters in the SBM force field before creating an OpenMM system object.

Finally, the *models* class automatizes the generation of default SBMOpenMM *system*-class objects containing the default force field parameters to create coarse-grained or all-atom SBMs, or their corresponding multi-basin versions, ready to be simulated with the OpenMM engine.[17] This class works as a container of current pre-defined SBMs, and it is the place for future SBM implementations.

A simple Python code to run a default all-atom SBM simulation with OpenMM is:

```
import sbmOpenMM
from simtk.openmm.app import *
from simtk.openmm import *
from simtk.unit import *

sbmAA = sbmOpenMM.models.getAllAtomModel(pdb_file, contact_file)

integrator = LangevinIntegrator(temperature*kelvin, 1.0/picosecond,
0.002*picoseconds)
simulation = Simulation(sbmAA.topology, sbmAA.system, integrator)
simulation.context.setPositions(sbmAA.positions)

simulation.reporters.append(DCDReporter('AAModel_traj.dcd', 10000))
sbmReporter = sbmOpenMM.reporter.sbmReporter('AAModel_energy.data',
                                          100,          step=True,
                                          potentialEnergy=True,
                                          temperature=True,
                                          sbmObject=sbmAA)

simulation.reporters.append(sbmReporter)

simulation.step(n_steps)
```

Here, the *sbmAA* object, created with SBMOpenMM, contains the SBM "topology," "system," and "positions" attributes that are to be passed to the OpenMM library.[17] After the *sbmAA* object is created, the usual steps for running simulations with OpenMM follow. First, an OpenMM *integrator* object is created and loaded into an OpenMM *simulation* object. Here, we pass the *topology* and OpenMM *system* attributes contained in the *sbmAA* object. Then, the initial positions in the *sbmAA* object are passed to the `simulation` OpenMM `context` attribute. In the example, to write simulation data into files, we define two reporter classes; a trajectory reporter (an OpenMM *DCDReporter* class) and a state data reporter (`StateDataReporter` class) for storing energies and other simulation parameters. The `sbmReporter` is an inherited class of the OpenMM `StateDataReporter` that allows communicating the SBM energies if an SBM object (here *`sbmAA`*) is passed to it with the `sbmObject` option. Finally, the last line in the code will advance the simulation for *`n_steps`* number of steps, thus starting the MD simulation.

Similarly, for creating a coarse-grained alpha-carbon (CA) SBM, the SBM `system` class definition changes to:

```
sbmCA = sbmOpenMM.models.getCAModel(pdb_file, contact_file)
```

The general workflow for setting up a custom SBM simulation with SBMOpenMM is shown in Figure 1 (for details, see Section S-I). Detailed steps for setting a custom SBM force field are given in Section S-II.A in the Supplementary material. Also, for a description of other pre-defined models in the `models` class, see section S-II.B.

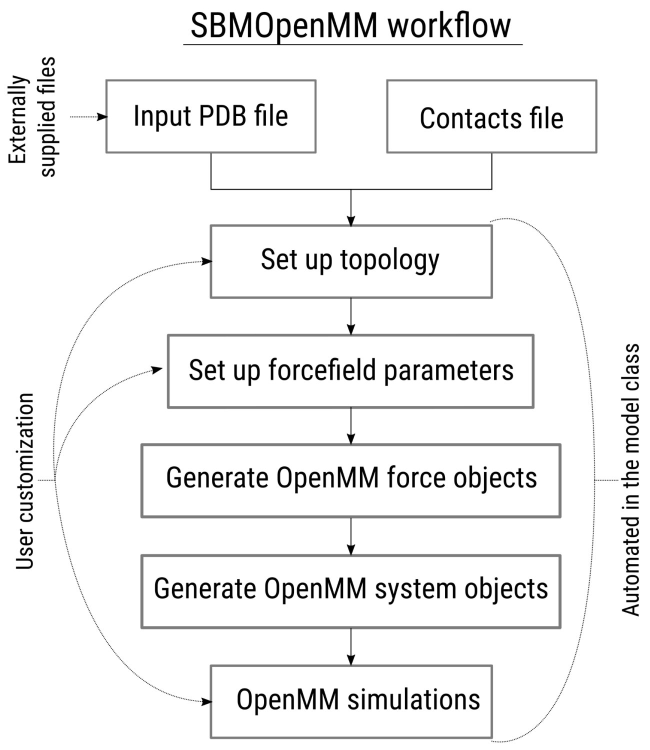
SBMOpenMM workflow
Externally supplied files
Input PDB file
Contacts file
Set up topology
Set up forcefield parameters
Generate OpenMM force objects
Generate OpenMM system objects
OpenMM simulations
User customization
Automated in the model class

Figure 1. Workflow to set up a simulation with SBMOpenMM. Only a PDB and a contact file are needed for running SBM simulations. The library contains automated methods to set up the geometric and force field parameters and generate the force objects that will act during the simulation. When all the parameters and forces are ready, SBMOpenMM generates an OpenMM system class object for running MD simulations. Most of these steps can be user-customized, including forces; however, to make a faster deployment of models, default SBMs can be called directly from the *model* class.

The `dumpStructure()` method inside the SBM `system` class can be used to generate PDB format files containing only the atoms in the SBM system during the trajectory:

```
sbmCA.dumpStructure('ca_output_file.pdb')
```

The `dumpForceFieldData` and `loadForcefieldFromFile` methods inside the SBM `system` class can be used to track or manually modify force field parameters. These methods can also be helpful when importing other SBMs into this library.

For further examples and additional information on creating and customizing SBM force fields with SBMOpenMM, please refer to the supplementary information and the hosting website's documentation and tutorials (see Data availability section).

Exploring SBM simulations for diverse protein systems

To examine the dynamic information produced by the SBMOpenMM library, we tested the code in a dataset of 110 single-chain protein structures (Table S1). The dataset is structurally diverse, representing a range of protein sizes and topologies (Figure S2). Several simulations were carried out to estimate the temperatures at which the folding transitions (T_f) occur. Dynamic profiles (calculated as root-mean-square fluctuations) at different relative temperatures were compared to the corresponding crystallographic temperature factors to determine correlation values (Figure S3). At temperatures lower than the T_f, average correlations are high (0.66); however, they start to decrease significantly at temperatures above $0.85T_f$. At temperatures above the T_f, the systems are mainly unfolded (Figure S4), and correlations decrease significantly (see Section S-III for details on the simulations, dataset compilation, and further discussion). This result shows that SBMOpenMM can produce dynamic profiles with good correlations to experimental information for studying folded configurations and consistent behavior for studying folded to unfolded transitions in protein systems.

Example of use

To portray the SBMOpenMM library's usage, we ran folding simulations for the monomeric structure of the FoxP1 DNA-binding domain.[18] Figure 2 shows five out of the fifteen folding/unfolding equilibrium replicas ran for this system, using an all-atom SBM at the folding temperature of the system (see Sections SI-III.C and IV.A for details, including the rest of the replicas). The native contacts employed to define the

SBM were calculated using the shadow contact algorithm.[19] The code for running the simulations is identical to the one shown in the Implementation section above. The simulations show different numbers of folding/unfolding transitions, ranging from one to eight, indicating independence on the course of the phase space sampled among them (Figure 2 and S1).

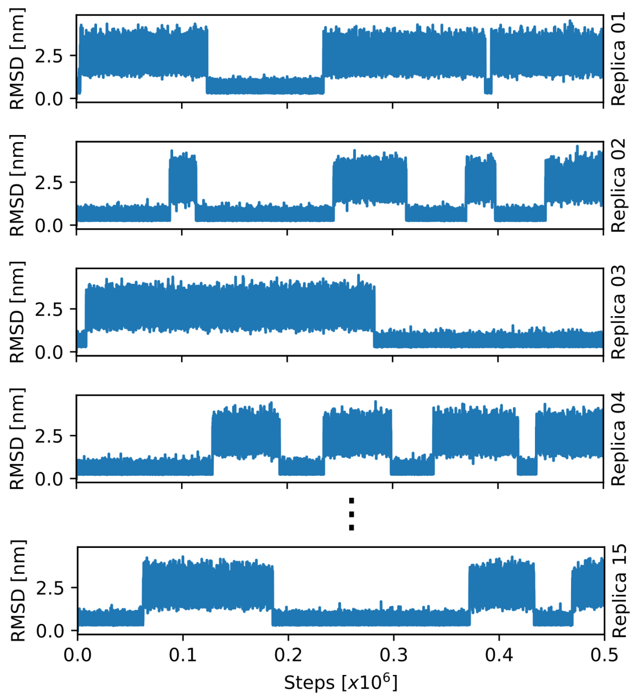

Figure 2. FoxP all-atom SBM folding simulations. Five of the fifteen SBM replicas at the folding temperature of the SBM system are shown. Each plot shows the evolution of the root-mean-square deviation (RMSD) to the native structure in nanometers. High and low RMSD values represent the time spent by the trajectory at the unfolded and folded configurations, respectively.

We analyzed the simulated trajectories using the Markov-State Model (MSM) framework implemented in the PyEMMA package[20] (for details on MSM construction and validation, see Section SI-IV.B). The system free energy was projected onto the two slowest time-lagged independent component analysis (TICA) dimensions[21] (Figure 3B). The simulation reproduces the experimental folding mechanism of FoxP1 as a two-state folder.[22]

To follow the unfolding mechanism progression, we estimated the probability of contact formation at the folded, transition-state (TS), and unfolded regions (Figure 3C). While the folded configuration retains most of the native contacts, it shows diminished interactions for the contacts made by the N- and C-terminal residues (Figure 3C, left matrix “b” triangle). Most native contacts are already lost at the TS; however, contacts made between strands 1, 2, and 3 hold the structure and are the last to be broken (Figure 3C, right matrix “c” triangle).

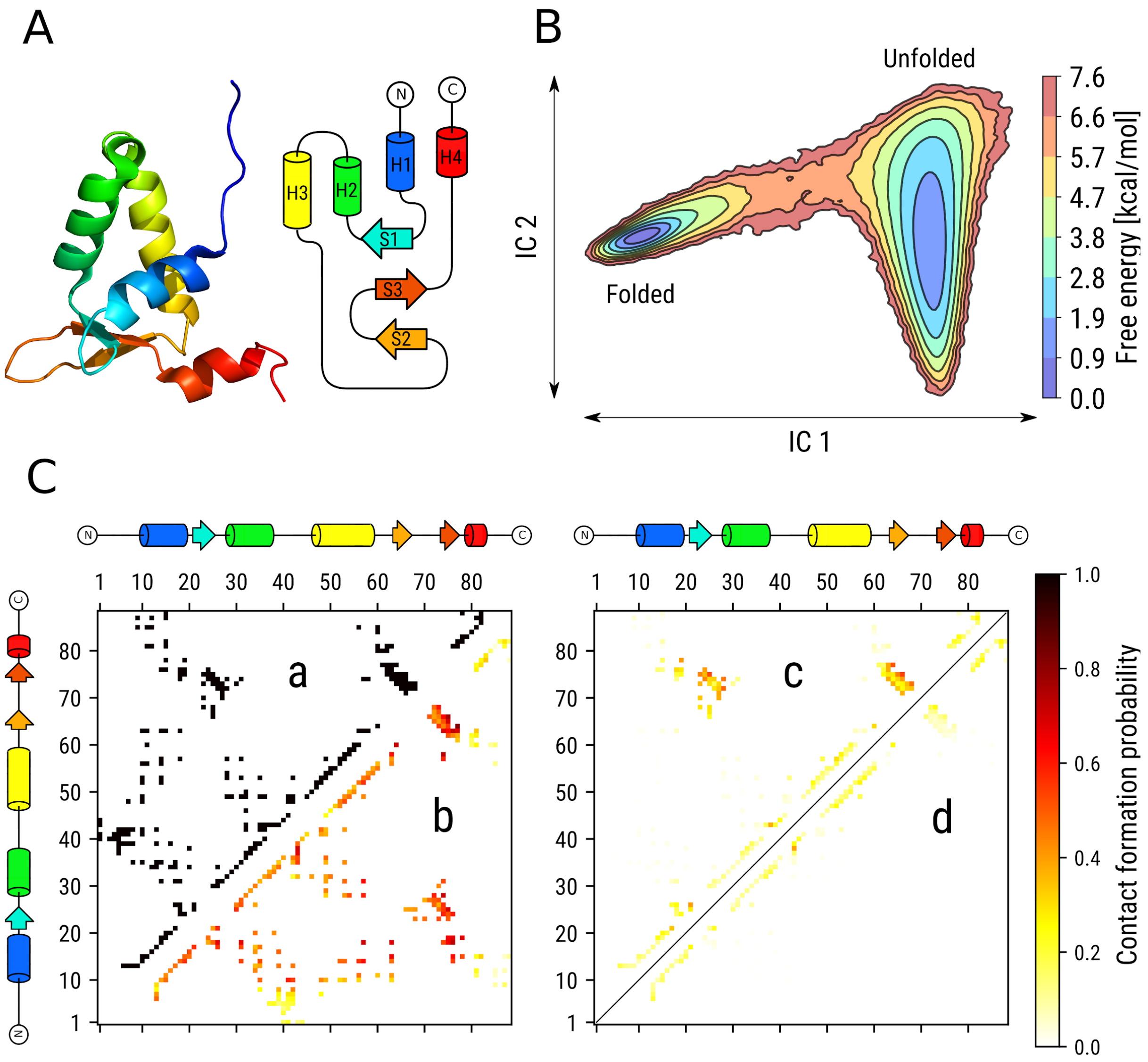

Figure 3. Analysis of the FoxP1 SBM folding simulation. (A) Tertiary structure and topological arrangement of the FoxP1 domain. Loops in this topology diagram are not in scale with their sequence length. (B) Free energy profile projected onto the two slowest time-lagged independent component analysis (TICA) coordinates (IC1 and IC2) based on native contact distances. FoxP1 folds in a two-state mechanism, clearly visible as two characteristic minima; the folded state (left) and the unfolded state (right). (C) FoxP1 tertiary structure per-residue contact map (a); and contact formation probability at the folded state (b), transition state (c), and unfolded state (d) for the SBM folding simulation.

Discussion

Due to the drastic reduction of the number of computed interactions (i.e., non-native and explicit solvent interactions are not taken into account), SBM simulations converge extraordinarily fast compared to standard MD, allowing, for example, simulating the folding of a small protein in equilibrium conditions by using only modern personal computer hardware, but also allowing to explore other biomolecular processes involving considerable energy barriers. Even though SBMs simplifications can overlook relevant pathways that include non-native interactions or explicit solvent definitions, an idealized study can serve many purposes before moving to more complex and definitely more expensive representations.

SBMs are used to understand the topological restrictions that a specific fold imposes on the protein's dynamical behavior. By later, including different or additional force field terms, deviations from this idealized model can be attributed to the specific physical parameters incorporated in the simulation. To this end, we have presented in this article a high-level language (Python) implementation of SBMs using OpenMM as the engine for molecular dynamic simulations. Our tool provides a platform to deploy implementations of SBMs, facilitating the programmatic development of force fields within an efficient and community-based molecular simulation package.

We have portrayed the use of the SBMOpenMM library in a study of the protein folding of a small-protein system, from the conformational sampling to the calculation of the free-energy surface and its kinetic hierarchization using a Markov state model analysis. SBMs can serve as an initial step before moving into simulations employing state-of-the-art MD force fields following adaptive sampling techniques to study complex biomolecular phenomena.[23] In such approaches, relevant conformations, carefully extracted from the simplified simulation, can be used as seeds to optimize the phase-space sampling by more costly simulations, making SBM a relevant and practical tool to conduct complex theoretical research in protein biophysics.[9,24]

Data and Software Availability

The SBMOpenMM library is publicly available at https://github.com/CompBiochBiophLab/sbm-openmm under MIT license. The package is written in Python 3, and its dependencies include OpenMM 7.0 (http://openmm.org/) and NumPy 1.15 (https://numpy.org/) or above. The simulations' analysis was carried out using the MDTraj 1.9 (https://mdtraj.org/) and PyEMMA 2.5 (http://emma-project.org/) packages, while graphics were generated with Matplotlib 3.3 (https://matplotlib.org/) and Seaborn 0.11 (https://seaborn.pydata.org/) Python libraries.

Associated Content

Supporting Information Available: The supporting file contains a more detailed explanation of how the folding SBM molecular dynamics of FoxP1 were run and how the Markov State Model Analysis was carried out. The structure, force field parameters, and simulated data for this system are available in a public repository (https://doi.org/10.34810/data31). The supporting file also contains a description of the implemented force objects in the SBMOpenMM library and the available default SBMs.

Author Contributions

MF and KJL developed the SBMOpenMM package. MF, MEG, and SO carried out the MSM analysis. LA, JKH, and JVF provided planning and guidance for the project. PM provided testing of the package. All authors contributed to write, read, and approve the final version of the manuscript. MF was a major contributor in writing the manuscript.

Corresponding author

Correspondence to Martin Floor (mfloor@uic.es).

Competing interests

The authors declare that they have no competing interests.

Funding

A UVic-UCC Ph.D. fellowship funded the work by MF. JVF, MF, LA, and PM thank Generalitat de Catalunya for the consolidated group BMS (2017 SGR-901). The SBMOpenMM software has been developed within the BIO2017-83650-P project, funded by the Spanish Subprograma Estatal de Generación del Conocimiento. MEG and SO thank Generalitat de Catalunya for the emerging group CompBioLab (2017 SGR-1707) and Spanish MINECO for project PGC2018-102192-B-I00. SO is grateful for the European Research Council (ERC) funding under the European Union's Horizon 2020 research and innovation program (ERC-2015-StG-679001).

References

1. Berendsen, H. Collective protein dynamics in relation to function. *Current Opinion in Structural Biology* vol. 10 165–169 (2000).
2. Kern, D. & Zuiderweg, E. R. P. The role of dynamics in allosteric regulation. *Current Opinion in Structural Biology* vol. 13 748–757 (2003).
3. Grant, B. J., Gorfe, A. A. & McCammon, J. A. Large conformational changes in proteins: signaling and other functions. *Curr. Opin. Struct. Biol.* **20**, 142–147 (2010).

4. Villà, J. & Warshel, A. Energetics and Dynamics of Enzymatic Reactions. *The Journal of Physical Chemistry B* vol. 105 7887–7907 (2001).
5. Yang, Y. I., Shao, Q., Zhang, J., Yang, L. & Gao, Y. Q. Enhanced sampling in molecular dynamics. *J. Chem. Phys.* **151**, 070902 (2019).
6. Blaszczyk, M. *et al.* Protein Structure Prediction Using Coarse-Grained Models. *Springer Series on Bio- and Neurosystems* 27–59 (2019) doi:10.1007/978-3-319-95843-9_2.
7. Rovigatti, L., Sulc, P., Reguly, I. Z. & Romano, F. A comparison between parallelization approaches in molecular dynamics simulations on GPUs. *J. Comput. Chem.* **36**, 1–8 (2015).
8. Levitt, M. & Warshel, A. Computer simulation of protein folding. *Nature* vol. 253 694–698 (1975).
9. Fan, Z. Z., Hwang, J.-K. & Warshel, A. Using simplified protein representation as a reference potential for all-atom calculations of folding free energy. *Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta)* vol. 103 77–80 (1999).
10. Taketomi, H., Ueda, Y. & Gō, N. STUDIES ON PROTEIN FOLDING, UNFOLDING AND FLUCTUATIONS BY COMPUTER SIMULATION: I. The effect of specific amino acid sequence represented by specific inter-unit interactions. *Int. J. Pept. Protein Res.* **7**, 445–459 (1975).
11. Nymeyer, H., García, A. E. & Onuchic, J. N. Folding funnels and frustration in off-lattice minimalist protein landscapes. *Proc. Natl. Acad. Sci. U. S. A.* **95**, 5921–5928 (1998).
12. Bryngelson, J. D. & Wolynes, P. G. Spin glasses and the statistical mechanics of protein folding. *Proceedings of the National Academy of Sciences* vol. 84 7524–7528 (1987).
13. Bryngelson, J. D. & Wolynes, P. G. Intermediates and barrier crossing in a random energy model (with applications to protein folding). *The Journal of Physical Chemistry*

vol. 93 6902–6915 (1989).

14. Clementi, C., Nymeyer, H. & Onuchic, J. N. Topological and energetic factors: what determines the structural details of the transition state ensemble and 'en-route' intermediates for protein folding? An investigation for small globular proteins. *J. Mol. Biol.* **298**, 937–953 (2000).
15. Noel, J. K. & Onuchic, J. N. The Many Faces of Structure-Based Potentials: From Protein Folding Landscapes to Structural Characterization of Complex Biomolecules. *Computational Modeling of Biological Systems* 31–54 (2012) doi:10.1007/978-1-4614-2146-7_2.
16. Takada, S. Gō model revisited. *Biophys Physicobiol* **16**, 248–255 (2019).
17. Eastman, P. *et al.* OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS Comput. Biol.* **13**, e1005659 (2017).
18. Chu, Y. P., Chang, C. H., Shiu, J. H. & Chang, Y. T. Solution structure and backbone dynamics of the DNA‐binding domain of FOXP1: Insight into its domain swapping and DNA binding. *Proteins* (2011).
19. Noel, J. K., Whitford, P. C. & Onuchic, J. N. The shadow map: a general contact definition for capturing the dynamics of biomolecular folding and function. *J. Phys. Chem. B* **116**, 8692–8702 (2012).
20. Scherer, M. K. *et al.* PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. *J. Chem. Theory Comput.* **11**, 5525–5542 (2015).
21. Pérez-Hernández, G., Paul, F., Giorgino, T., De Fabritiis, G. & Noé, F. Identification of slow molecular order parameters for Markov model construction. *J. Chem. Phys.* **139**, 015102 (2013).
22. Medina, E. *et al.* Three-Dimensional Domain Swapping Changes the Folding Mechanism of the Forkhead Domain of FoxP1. *Biophys. J.* **110**, 2349–2360 (2016).
23. Hruska, E., Abella, J. R., Nüske, F., Kavraki, L. E. & Clementi, C. Quantitative

comparison of adaptive sampling methods for protein dynamics. *J. Chem. Phys.* **149**, 244119 (2018).

24. Avila, C. L., Drechsel, N. J. D., Alcantara, R. & Villa-Freixa, J. Multiscale Molecular Dynamics of Protein Aggregation. *Current Protein & Peptide Science* vol. 12 221–234 (2011).