Computer Physics Communications 239 (2019) 311–324

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

gen.parRep: A first implementation of the Generalized Parallel Replica dynamics for the long time simulation of metastable biochemical systems[☆]

Florent Hédin[*], Tony Lelièvre

CERMICS, École des Ponts–ParisTech, 6 et 8 avenue Blaise Pascal, Cité Descartes – Champs-sur-Marne, F-77455 Marne-la-Vallée Cedex 2, France
Inria Centre de Recherche Paris Rocquencourt, 2 rue Simone Iff, F-75589 Paris, France

ARTICLE INFO

Article history:
Received 13 August 2018
Received in revised form 14 December 2018
Accepted 7 January 2019
Available online 25 January 2019

Keywords:
Molecular simulations
Langevin dynamics
Quasi-stationary distributions
Metastability
Protein–Ligand dissociation
High performance computing

ABSTRACT

Metastability is one of the major encountered obstacles when performing long molecular dynamics simulations, and many methods were developed to address this challenge. The “Parallel Replica”(ParRep) dynamics is known for allowing to simulate very long trajectories of metastable Langevin dynamics in the materials science community, but it relies on assumptions that can hardly be transposed to the world of biochemical simulations. The later developed “Generalized ParRep” variant solves those issues, but it was not applied to significant systems of interest so far.

In this article, we present the program *gen.parRep*, the first publicly available implementation of the Generalized Parallel Replica method (BSD 3-Clause license), targeting frequently encountered metastable biochemical systems, such as conformational equilibria or dissociation of protein–ligand complexes. It will be shown that the resulting C++ implementation exhibits a strong linear scalability, providing up to 70% of the maximum possible speedup on several hundreds of CPUs.

Program summary
Program Title: gen.parRep
Program Files doi: http://dx.doi.org/10.17632/dgp8wdgtgt.1
Licensing provisions: BSD 3-clause
Programming language: C++ (mostly), C and Lua
Nature of problem: Molecular dynamics simulations of chemical and biological systems usually encounter the problem of metastability, because of the timescale separation between the time discretization step used for dynamics and the usual mean time between conformational changes. The use of Accelerated dynamics [1] methods is usually necessary in order to address this challenge.
Solution method: The Generalized Parallel Replica method [2] accelerates the exit from metastable states, providing a linear speedup of N, N being the number of replicas of the system running in parallel. This C++ implementation, the first available so far, exhibits a strong linear scaling on hundreds of CPUs, therefore ready for production studies on High Performance Computing (HPC) machines.
Additional comments: Git repository: https://gitlab.inria.fr/parallel-replica/gen.parRep
References:
[1] Lelièvre, T., Eur. Phys. J. Spec. Top. (2015) 224, DOI: 10.1140/epjst/e2015-02420-1
[2] Binder, A. et al., Journal of Computational Physics (2015) 284; DOI: 10.1016/j.jcp.2015.01.002

1. Introduction

Molecular dynamics (MD) simulations are nowadays of a common use for simulating large and complex biological or chemical systems [1]: the continuous increase of the available computing power, together with the development of stable and accurate deterministic or stochastic sampling strategies, made possible the emergence of computer based, *in silico* drug design strategies [2–4]. However, a commonly encountered obstacle while performing MD simulation is the timescale separation between the fastest

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author at: CERMICS, École des Ponts–ParisTech, 6 et 8 avenue Blaise Pascal, Cité Descartes – Champs-sur-Marne, F-77455 Marne-la-Vallée Cedex 2, France.

E-mail addresses: florent.hedin@enpc.fr (F. Hédin), tony.lelievre@enpc.fr (T. Lelièvre).

https://doi.org/10.1016/j.cpc.2019.01.005
0010-4655/© 2019 Elsevier B.V. All rights reserved.

conformational changes – usually vibrations occurring at the femtoseconds (fs) level – and the slowest one, occurring from the nanosecond (ns) to second (or more) timescale; one may use various coarse grained [5,6] approaches in order to reach such large simulation time, however this usually implies to sacrifice the accurate description of fast processes, such as non-bonded donor–acceptor interactions, playing a key role in biological interactions [7,8]. The existence of *metastable* regions in the configurational space, separated by high potential energy or entropy barriers, is the main origin of this timescale separation, and the simulation time required for observing a transition from such a region to another one can quickly become intractable by the use of direct numerical simulations.

A large number of methods were developed to address the challenge of metastability in MD simulations. When it is assumed that both the starting and the ending metastable regions (let us denote them by A and B) are known, one can consider that most of the methods fall within one of the two following categories: *local search methods* start from an initial guess path connecting A and B, and will optimize it until convergence to an optimal path, for example characterized by a minimal potential or free energy profile: the nudged elastic band method [9], the string method [10], the max flux approach [11], the weighted ensemble methods [12,13], or the transition path sampling method [14] (which is actually a path sampling method starting from the initial guess, but not an optimization method); the second category consists in *global search methods* where the ensemble of all the possible paths between A and B is sampled without any initial guess, and it includes adaptive multilevel splitting methods [15–18], transition interface sampling [19], forward flux sampling [20] or milestoning techniques [21–27].

A.F. Voter and coworkers also proposed another class of methods, the *Accelerated Dynamics methods* [28–32]: the Parallel Replica (ParRep) method (and the derived ParSplice algorithm) [33–36], the hyperdynamics method [37,38], and the temperature accelerated dynamics method [39]. They all rely on the Transition State theory and kinetic Monte Carlo models, and the aim of these algorithms is to efficiently generate the succession of jumps between metastable regions in a statistically consistent way compared to the reference Langevin dynamics.

The ParRep method was later formalized [40], and it was shown that the notion of quasi-stationary distribution (QSD) [41,42] is the mathematical foundation at its heart: this revealed one of the possible weaknesses of the algorithm, where it is assumed that the (user defined) time required for converging to the QSD is the same for all the metastable regions. While this assumption may be reasonable for materials science, this cannot be transposed to the chemical configurational space, where the large variety of possible interactions and steric exclusions usually result in a rough energy landscape, characterized by both an extremely large number of energy minima, and the presence of super basins of attraction (usually referred to as "funnels"). The Generalized Parallel Replica (Gen. ParRep) [43] method addresses this issue by estimating during the simulation if convergence to the QSD is obtained; however, while this can possibly extend the range of application of ParRep to any biochemical system which can be studied via MD, no implementation has been designed and released so far.

This article describes the first publicly available implementation of Gen. ParRep, specially targeting metastable biochemical systems. After a description of the methods in Section 2, the novelty of the software implementation is detailed in Section 3; two study cases are later investigated in Section 4, the conformational equilibrium of the alanine dipeptide (Section 4.1), and the dissociation of the protein–ligand complex FKBP–DMSO (Section 4.2). It will be shown in both cases that the Gen. ParRep algorithm can be used for accurately sampling the state-to-state dynamics, and in particular the state-to-state transition times; furthermore evidences that the software exhibits a strong linear scaling will be reported: when running over hundreds of CPUs, one gets speedups of up to 70% of the maximum possible linear speedup.

2. Methods

2.1. Langevin dynamics

Let us consider a stochastic process $X_t = (q_t, p_t)_{t\geq 0} \in \mathbb{R}^{d\times d}$ ($\mathbb{R}^{d\times d}$ representing the *phase space*), where q_t and p_t denote the positions and momenta of the $d/3$ particles at time t. The stochastic process X_t follows the Langevin dynamics:

$$\begin{cases} dq_t = M^{-1} p_t \, dt \\ dp_t = -\nabla V(q_t)\, dt - \gamma M^{-1} p_t \, dt + \sqrt{2\gamma\beta^{-1}} dW_t \end{cases} \tag{1}$$

where $\beta = \frac{1}{k_B T}$ is the inverse temperature, M is the mass matrix, $V : \mathbb{R}^d \to \mathbb{R}$ is a function associating to a given configuration q a potential energy $V(q)$, $\gamma > 0$ is the damping parameter, and W_t a d-dimensional Brownian motion.

The Langevin dynamics on the d-dimensional potential energy surface V is likely to consist in a succession of "entry then exit" events from wells (or groups of wells) progressively discovered by the process X_t, and one can expect that the time spent within a well before it hops to another one will be far more large that the discretization timestep dt: it is therefore necessary to design an alternative approach to the computationally expansive direct simulation in order to address this problem of *metastability*.

2.2. States and metastability

Let us introduce the ensemble of metastable states $\mathcal{S} = \{S_1, \ldots, S_n\}$. These are typically defined in terms of positions only (and not velocities). In the original ParRep algorithm [33,35], these states are defined as the basins of attraction of the local minima of V for the gradient descent $\dot{q} = -\nabla V(q)$: this leads to a partition of the state space. One important output of the mathematical analysis performed in Ref. [40] is that (i) the metastable states can be defined arbitrarily, the only prerequisite being that for most of the visits in one of those, the exit time will be much larger than the convergence time to the local equilibrium within the state (the so-called Quasi Stationary Distribution), and (ii) the algorithm can be applied even if these metastable states do not define a partition of the state space: in this work we propose to define them as disjoint subsets, using *collective variables* or *reaction coordinates* [44,45], chosen a priori in order to correspond to a few given metastable conformations of the molecular system of interest; the topological definition of the states will be discussed for each system of interest in Section 4.

Let $\Omega \in \mathcal{S}$ be a given state: we define

$$\tau = \inf\{t \geq 0 \mid X_t \notin \Omega\}$$

to be the first exit time from Ω (for a given initial condition $X_0 \in \Omega$), and

$$X_\tau \in \partial\Omega$$

to be the corresponding exit configuration (first hitting point on the boundary $\partial\Omega$): the goal of the various Parallel Replica (ParRep) [33,35,43] based methods (but also of other accelerated dynamics methods) is to sample efficiently the values (τ, X_τ) from the unknown exit distribution associated to each state Ω.

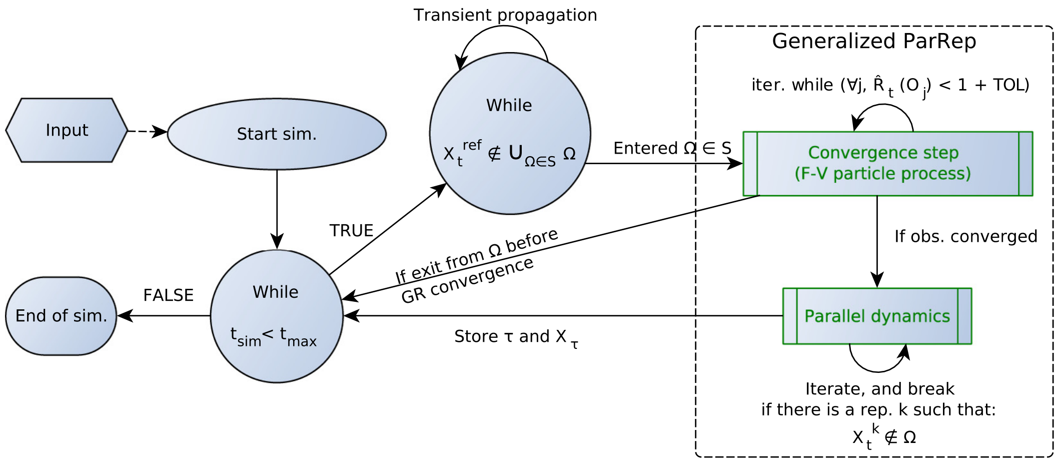


Fig. 1. Diagram view of the generalized ParRep [43] algorithm. After setup, the first step (Transient propagation) is to iterate the reference walker until X_t enters a defined state Ω (if the states define a partition of the configuration space, this first part of the algorithm is not required); then the Gen. ParRep procedure (right frame) is executed, starting with the Convergence step, until either: (i) the reference walker exits before convergence of the Gelman–Rubin (G–R) statistics is observed, or (ii) convergence of the G–R observables is obtained before the reference walker exits. In the later case simulation proceeds to the Parallel dynamics step, until an exit event from Ω is observed, generating a sample of (τ, X_τ). After the parallel phase (or if X_t exited Ω before convergence), the reference walker performs once again the Transient propagation procedure, until entering a valid state Ω, and the Gen. ParRep procedure is iterated once again. This is repeated until the total simulation time t_{sim} reaches a user defined value $t_{\max}$, where the program stops. The two frames colored in green are parts of the algorithm fully exploiting the N available CPU cores.

2.3. Quasi-stationary distribution (QSD)

Recent mathematical analyses showed [40] that the *quasi-stationary distribution* (QSD) [41,42] is an essential ingredient of the above mentioned accelerated dynamics methods. Let ν be a probability measure with support in Ω: ν is a QSD if and only if, for any $A \subset \Omega$ and $t \geq 0$:

$$\nu(A) = \mathbb{P}^{\nu}\left[X_t \in A \mid t < \tau\right]$$

where $\mathbb{P}^{\nu}$ indicates that the initial condition X_0 is distributed according to ν. This means that ν is a QSD if, for all t, when X_0 is distributed according to ν, the law of X_t conditionally to the fact that $(X_s)_{0\leq s\leq t}$ remains in the state Ω is still ν.

The QSD satisfies the following properties which will be of critical importance (see Refs. [40,43] for detailed proofs):

1. Existence and uniqueness of ν: the QSD is the unique long time limit ($t \to +\infty$) of the distribution of X_t, conditioned to starting and staying in Ω up to time t;
2. if X_0 is distributed according to the QSD ν, then the first sampled exit time τ is independent of the first sampled exit configuration X_τ;
3. if X_0 is distributed according to the QSD ν, sampled values of the first exit time τ are exponentially distributed: $\mathbb{P}^{\nu}(\tau > t) = e^{-\lambda t}$, where $\lambda = \frac{1}{\mathbb{E}^{\nu}(\tau)}$.

2.4. The generalized ParRep method

Having introduced the concepts of states and QSD, it is now possible to detail the Generalized Parallel Replica [43] (Gen. ParRep) method. In the following, it is assumed that different metastable states $\mathcal{S} = \{S_1, \ldots, S_n\}$ are defined, either by partitioning the whole configuration space, or by defining disjoint subsets of $\mathbb{R}^d$, and Ω denotes any member of $\mathcal{S}$. It is also assumed that at least N CPU cores are available in order to propagate simultaneously N replicas of the system in parallel.

As stated above, the aim of accelerated dynamics methods is to quickly sample values of (τ, X_τ) (respectively the first exit time from a metastable $\Omega \in \mathcal{S}$ and the first hitting point on the boundary $\partial\Omega$): in the case of ParRep methods, detailed information about how the process evolves within each state Ω is discarded, and in return exit events can be generated N times faster (a linear speedup is achieved), which is of particular interest when considering computations performed on High Performance Computing (HPC) machines, where thousands of CPUs can be used at once by a single simulation.

In the following, let $t_{\text{sim}} \geq 0$ be the simulation clock, corresponding to the physical time (i.e. a multiple of the time step dt), and let $X^{\text{ref}}_{t_{\text{sim}}}$ be the configuration of the system at time t_{sim} (where ref indicates the *reference walker*, the first replica). The method is implemented as a three steps procedure, repeated as the process diffuses from one state to another, until a total simulation time $t_{\max}$ is reached (see Fig. 1 for a diagram representation):

1. Transient propagation: if the set $\mathcal{S}$ is not a partition of the whole configuration space, it might be that $X^{\text{ref}}_{t_{\text{sim}}}$ is outside of any known state: therefore the process has to be propagated for a time t_{reach} until it reaches a metastable state Ω (note that t_{reach} is expected to be much smaller than the typical exit times from the states in $\mathcal{S}$, at least if the states definitions encompass accurately the metastable domains). After this step, the simulation time is updated as $t_{\text{sim}} \leftarrow t_{\text{sim}} + t_{\text{reach}}$.
2. Convergence step: a Fleming–Viot (F–V) particle process is launched to estimate the convergence time to the QSD. If the reference walker leaves Ω before the convergence time to the QSD, one goes back to step 1. If not, one proceeds to the Parallel dynamics step.
3. Parallel dynamics step: N replica are propagated independently in parallel, until one exits the state Ω. The corresponding exit time τ is calculated (more details below) and is saved together with the exit configuration X_τ; then the program proceeds to a new Transient propagation.

In terms of wall-clock time, the computational gain of this algorithm compared to a direct numerical simulation comes from the parallel dynamics step, which allows to generate a sample of the exit event (τ, X_τ) in a wall-clock time N times smaller than for the direct numerical simulation.

In the following the Convergence step and Parallel dynamics step will be detailed.

2.4.1. Convergence step: Fleming–Viot process and Gelman–Rubin convergence diagnostic

The Fleming–Viot (F–V) process [46,47] is a branching and interacting particle process, used for simulating the law of the random variable X_t conditioned to $\{\tau > t\}$. As a consequence, an estimate of $t_{\text{F-V}}$ – the F–V convergence time – can be obtained by assessing the convergence to a stationary state of the F–V process, and when this convergence is observed, one obtains samples (approximately) distributed according to the QSD. For a detailed description with illustrations, we refer to the dedicated section from Ref. [43].

Let us first consider N i.i.d. initial conditions X_0^k $(k \in \{1, \dots, N\})$; the procedure summarizes as follows: a *reference walker* X_t^{ref} (namely the replica numbered $k = 1$) explores Ω driven by the Langevin equation (1): at the same time the other replicas (the *F–V workers*) perform the following tasks:

1. the F–V workers evolve independently according to Eq. (1) within Ω, each of them regularly collecting the instant values of several *observables*; until one of them, e.g. X^i, exits;
2. the process i that exits is discarded, and replaced by a copy of one of the other F–V workers (*survivors*), randomly drawn with uniform probability among the survivors: this is called a *F–V branching*;
3. the survivors and the newly branched processes evolve and collect values, going back to 1, until convergence is reached for each observable (convergence will be defined below using the Gelman–Rubin diagnostic).

However, if at any moment the reference walker X^{ref} leaves Ω before the F–V process has converged, all the F–V walkers replicas are killed, and a new Transient propagation is initiated.

The observables are properties of interest which are expected to characterize the convergence to equilibrium of the F–V particle process within each state Ω: they can have a physical meaning (e.g. based on the potential V, or the momenta p), or be any type of distance/topological measure (for instance derived from the collective variables used for designing the sets in $\mathcal{S}$).

The convergence of the observables is assessed using the Gelman–Rubin (G–R) statistics [48,49]: let $\mathcal{O} : \Omega \to \mathbb{R}$ be some observable, and let

$$\bar{\mathcal{O}}_t^k \equiv t^{-1} \int_0^t \mathcal{O}(X_s^k)\, ds$$

$$\bar{\mathcal{O}}_t \equiv \frac{1}{N} \sum_{k=1}^{N} \bar{\mathcal{O}}_t^k = \frac{1}{N} \sum_{k=1}^{N} t^{-1} \int_0^t \mathcal{O}(X_s^k)\, ds \tag{2}$$

be the average of an observable along each trajectory ($\bar{\mathcal{O}}_t^k$) and the average of the observable along all trajectories $\bar{\mathcal{O}}_t$. The statistic of interest for the observable $\mathcal{O}$ is defined by:

$$\hat{R}_t(\mathcal{O}) = \frac{\frac{1}{N}\sum_{k=1}^{N} t^{-1} \int_0^t (\mathcal{O}(X_s^k) - \bar{\mathcal{O}}_t)^2 ds}{\frac{1}{N}\sum_{k=1}^{N} t^{-1} \int_0^t (\mathcal{O}(X_s^k) - \bar{\mathcal{O}}_t^k)^2 ds} \tag{3}$$

Note that $\hat{R}_t(\mathcal{O}) \geq 1$, and as the F–V workers' trajectories explore Ω, $\hat{R}_t(\mathcal{O})$ converges to 1 as t goes to infinity.

The time required for the F–V particle process to converge is denoted by $t_{\text{F-V}}$ and defined by:

$$t_{\text{F-V}} = \inf\left\{t \geq 0 \mid \hat{R}_t(\mathcal{O}_j) < 1 + \text{TOL},\ \forall j\right\} \tag{4}$$

i.e. it is the time required for obtaining a ratio $\hat{R}_t(\mathcal{O})$ less than $1 + \text{TOL}$ for each of the observables $\mathcal{O}$ (where $\text{TOL} > 0$ is a user defined stopping criterion).

After a successful Convergence step, the simulation clock is updated as follows:

$$t_{\text{sim}} \leftarrow t_{\text{sim}} + t_{\text{F-V}}$$

and one proceeds to the Parallel dynamics step; in case the reference walker left Ω before the convergence time $t_{\text{F-V}}$, the simulation clock time is updated as follows:

$$t_{\text{sim}} \leftarrow t_{\text{sim}} + t_{\text{ref}}$$

where t_{ref} is the amount of simulation time the reference walker spent within Ω before an exit event was observed, and one proceeds to a new Transient propagation step.

Note that because of the small value of the timestep dt, usually chosen between 0.5 and 2 fs, one does not expect to observe large fluctuations of the observables between two consecutive times t and $t + dt$: it therefore makes sense to accumulate the values of the observables less frequently, say with a period $t_{\text{G-R}}$, satisfying $dt < t_{\text{G-R}} \ll t_{\text{F-V}}$.

Likewise, the test to check whether an exit from Ω occurred is only performed with period t_{check}, with typically $t_{\text{G-R}} < t_{\text{check}} \ll t_{\text{F-V}}$.

2.4.2. *Parallel dynamics step*

The N samples obtained after the Convergence step are used as initial conditions; then the N replicas are propagated following Eq. (1) with independent driving Brownian motions.

Let $t_{\text{para}} = 0$ be the simulation time spent in the Parallel dynamic step, until the first exit event is observed; let t_{check} be a simulation time interval (multiple of dt) at which one tests if an exit event occurred, and let M count how many times this test was performed before an exit event occurred; finally let

$$k = \min \arg \min_{n \in \{1,\dots,N\}} t_{\text{para}}^n$$

be the index of the first replica for which an exit event occurred: then it was shown [50] that the exit time τ can be sampled as:

$$\tau = [N(M-1) + k]\, t_{\text{check}}. \tag{5}$$

The simulation clock is updated as:

$$t_{\text{sim}} \leftarrow t_{\text{sim}} + \tau. \tag{6}$$

A new Transient propagation can therefore be initiated, using as new initial condition the exit point X_τ^k of the first replica which exited.

2.4.3. *Differences with the original ParRep algorithm*

The Gen. ParRep algorithm differs from the original ParRep algorithm (as described in Refs. [33,34]) on several points:

- The original ParRep algorithm has originally been introduced on a partitioned configuration space, usually defining states as the basins of attraction of the local minima of the potential energy function, thus implying regular gradient descent on V. This makes the state identification simple and unambiguous for systems characterized by a smooth potential energy landscape where minima are separated by high energy barriers; however biochemical systems are usually characterized by rough and funneled energy landscapes, where conformation changes usually involve numerous transitions over local minima separated by low energy barrier.
- The original ParRep implementations require the user to define two parameters, the decorrelation time t_{corr} and the dephasing time t_{phase}. The decorrelation time t_{corr} is used to assess the convergence to the QSD for the reference walker: if it stays in a state Ω for a time t_{corr} it is assumed to be distributed according to the QSD. Likewise, the dephasing time t_{phase} is used to sample the QSD before the Parallel dynamics step starts: in the so-called dephasing step, each of the N replicas is propagated within the state Ω, and its end point is kept as a sample of the QSD if it stayed within the state Ω for a time t_{phase}. Once again, this approach appears hardly compatible with biochemical systems, as it is impossible to define ubiquitous values of t_{corr} and t_{phase} appropriate for all the possible local minima and all initial conditions within the states.

Those two limitations are addressed by the implementation of the Gen. ParRep algorithm described in this article: while permitted, partition of the configuration space is not enforced, and the user has total control on how to define the states; this allows for instance to merge multiple local minima together in order to define a metastable state accurately englobing a funnel of the PES.

Furthermore the use of the F–V particle process during the Convergence step releases the user from providing a priori estimates of the time required for converging to the QSD, as $t_{\text{F–V}}$ is estimated on the fly based on the convergence of the observables, the only requirements being to provide meaningful observables and a tolerance level.

3. Software implementation

In Section 4 we will present results obtained with our current implementation of the Generalized ParRep algorithm: it consists in a newly written C++ program, *gen.parRep*, available free of charge (see https://gitlab.inria.fr/parallel-replica/gen.parRep) and released under an open-source BSD 3-clause licensing. We aimed at providing an easy to use, versatile and performance oriented implementation, focusing on the study of metastability encountered when studying chemical and biochemical systems. Note that while the original ParRep method is also implemented and available in our new software, we will not present any result for it, as we focus on the novelty of Gen. ParRep.

In the following paragraphs, the critical requirements for developing such a code are detailed, together with details on the technical solutions adopted in order to address them.

3.1. Distributed computing capabilities

The replica-based approach of the ParRep algorithms naturally suggests that the parallelization is achieved by using a distributed computing approach: an obvious choice nowadays is to use the Message Passing Interface (MPI) [51] standardized protocol, for which various high performance computing (HPC) implementations are available [52,53].

Each of the N replicas corresponds to a *MPI task*: each task will use P CPU cores, P being at least 1 and at most all the cores available on a given machine (a *MPI node*). Therefore each computing node will execute 1 or more replicas, each performing the dynamics on P cores.

Regularly, *messages* of arbitrary size are exchanged between the replicas, which can be classified in two categories:

- *point-to-point* communications involve two replicas and are usually inexpensive as long as the amount of data sent remains relatively small: one example is the branching and cloning operation of the F–V algorithm, where an exiting F–V worker will copy the $X_t = (q_t, p_t)$ configurations plus the history of all the $\mathcal{O}$ observables from another F–V worker.
- *collective* communications involve the full ensemble of the N replicas and are likely to be time consuming, and are therefore used with care: they include *barriers* for keeping the replicas synchronized and *broadcasting* operations where a replica sends its configuration $X_t = (q_t, p_t)$ to the $(N-1)$ others (for example to be used as an initial condition for the next F–V iteration).

Furthermore, communications can either be *blocking* or *non-blocking*, the later allowing the developer to interleave communications and computations in order to hide latency. To provide an efficient Gen. ParRep implementation, the use of barriers and collective communications have been reduced to the minimum possible, and non-blocking variants of those were used whenever possible.

3.2. MD engine

One requires an efficient Molecular Dynamics (MD) engine, capable of performing the dynamics of Eq. (1): the minimal requirement is to have access to one code block which, when executed, will realize one or more discretization steps of size dt, and which internally takes care of the evaluation of the potential $V(q)$ and its gradient (usually analytically calculated). A read and write access to the internal configuration $X_t = (q_t, p_t)$ of each replica is also required for performing the exchanges.

In order to study large systems, one also expects: full support of commonly used force-fields, availability of modern optimizations such as the Particle Mesh Ewald [54], Reaction Field [55,56], or Cell-Linked Lists [57–60] methods, for an efficient evaluation of non-bonded interactions. As mentioned in the previous paragraph one can decide to provide $P \geq 1$ CPU cores to each of the N replicas, therefore a shared memory parallelization capability for the MD engine is encouraged.

For the current implementation it was decided to use the OpenMM 7 library; [61] OpenMM is a high performance, free of charge and open source toolkit for performing molecular simulations, which can be used either as a software library on which to build a program, or directly as an application (via python scripting): the later is used for preparing the molecular systems before simulation, accepting force-field and configuration files from various origins (CHARMM [62], AMBER [63], GROMACS [64], NAMD [65], …), while the library mode provides a direct and simplified access to the MD engine from the C++ application.

3.3. Definition of the states $\mathcal{S}$

While technical aspects as parallelization and efficiency of the MD engine are important, the Generalized ParRep critically relies on an efficient definition of the set of states $\mathcal{S}$. As stated in Section 2.2 this implementation focuses on applications where the states are a priori defined using either atomic coordinates or more elaborated collective variables: it is thus necessary to provide a way to define the states *online*, e.g. using a scripting language interfaced with the core C++ methods in order to have access to atomic properties.

It was decided to use the Lua [66] language: it is a fast, lightweight, easy to learn, embeddable and dynamically typed scripting language. The user input required for running the ParRep algorithms is written to an input Lua file, together with all the codes and variables for (i) defining the states, (ii) checking if an exit event is observed, and (iii) monitor the G–R statistics. The Sol2 [67] library (embedded within the C++ program's source code) takes care of parsing the input file at initialization, and it dynamically maps the user-defined code to C++ functions. The core code is therefore *state agnostic* as it never exactly knows how a state has been defined: indeed the whole implementation will only call the following: (i) a function returning a true/false boolean value indicating if ($X_t \notin \mathcal{S}$) (always true in case of a partitioned configuration space, possibly false otherwise); (ii) another function returning a true/false boolean value indicating if ($X_t \notin \Omega$) where Ω is the last visited state, being called every time it is required to check if an exit event occurred; (iii) and a few functions (one per user defined observable) monitoring the G–R observables $\mathcal{O}$ returning a real value to be accumulated and used in Eqs. (2)–(4). Fig. 2 exemplifies the Lua code checking if an exit event has occurred.

For further increased performance it is possible to use the LuaJIT implementation [68] where the Lua code is compiled to machine code during parsing: this allows performance close to what would be obtained by defining the states based on compiled code

Finally, the Lua layer can act as an intermediate proxy between the C++ Gen. ParRep code and any other external library, providing

```lua
-- this function is mandatory and called from C++, program will fail if not defined
--  it should take no arguments
--  it should return a boolean : true in case the dynamics left the state, false otherwise
function check_state_left()

  -- access coordinates of the 2 atoms defining group index_dist_1
  local d1_h_x,d1_h_y,d1_h_z = get_coordinates(index_dist_1[1]+1)
  local d1_o_x,d1_o_y,d1_o_z = get_coordinates(index_dist_1[2]+1)

  -- access the com of the 6-ring and the S atom defining group index_dist_2
  local d2_ring_x,d2_ring_y,d2_ring_z = get_COM_idxs(table.slice(index_dist_2,1,6))
  local d2_s_x,d2_s_y,d2_s_z = get_coordinates(index_dist_2[7]+1)

  local d1 = 10.0*math.sqrt( (d1_h_x-d1_o_x)^2    + (d1_h_y-d1_o_y)^2    + (d1_h_z-d1_o_z)^2 )
  local d2 = 10.0*math.sqrt( (d2_ring_x-d2_s_x)^2 + (d2_ring_y-d2_s_y)^2 + (d2_ring_z-d2_s_z)^2 )

  local newState = currState

  -- we require BOTH d1 and d2 to be strictly larger than b_to_u for having an unbound state
  if(d1>b_to_u_d1 and d2>b_to_u_d2) then
    currState = 'u'
  else
    currState = 'b'
  end

  print('Distances d1,d2 are : ',d1,' and ',d2)

  print('DMSO state appears to be :',currState)

  if (newState == currState) then
    return false
  else
    return true
  end

end
```

Fig. 2. Example of a Lua function written by the user within the input file (corresponding to the validation system presented in Section 4.2), and called from the C++ program. This function is called every time the algorithm checks whether an exit event from the current metastable state Ω happened, either during the Convergence Step or the Parallel dynamics step. The variables `index_dist_1` and `index_dist_2` are simply tables of atomic indices defined earlier in the input file, and indices used in the state definition (see Fig. 9(b)). Functions `get_coordinates(sele)` and `get_COM_idxs(sele)` are bindings to the C++ code which respectively retrieve atomic coordinates, and calculate the center of mass, for a given set of atomic indices `sele`. This versatile procedure gives to the user a lot of flexibility for: (i) defining the metastable states and (ii) detecting exits, as it does not require any modification of the compiled code.

the possibility to define states and observables using external software pieces: one can for example imagine to use tools such as Colvars [44] or PLUMED [45], providing access to an extensive ready to use collection of collective variables definitions.

3.4. *On the choice of $t_{\mathrm{G-R}}$ and t_{check}*

As previously mentioned in Section 2.4 it is not necessary to check at each integration of t if $(X_t \notin \Omega)$ (parallel step) or $(X_t^{\mathrm{ref}} \notin \Omega)$ (F–V step) as one expects that the exit time is much larger than dt.

And likewise, while it is important to regularly gather the value of the G–R observables $\mathcal{O}$ in order to obtain convergence of Eq. (3), it is expected that they will not differ that much between time t and $t + dt$: hence it is interesting to choose $t_{\mathrm{G-R}} > dt$.

While the values should be fine tuned for each system, based on our experience we ended up with the following rule of thumb: one can take $t_{\mathrm{G-R}}$ to be 5 to 100 times the value of dt, and t_{check} to be 500 to 2000 times dt. This should be adjusted depending on: (i) the amount of calculations involved in the process of defining the state and the G–R observables in the input script, and (ii) the size of the system; for a large solvated protein, if the states and observables only involve distance measures on a few atoms, then the time required for performing the dynamics will be comparatively much larger and relatively small values of $t_{\mathrm{G-R}} \approx 10$ dt and $t_{\mathrm{check}} \approx 250$ dt can be selected; however, if it involves tracking the length of several dozens of hydrogen distances, or counting native contacts, a wise approach would be to choose $t_{\mathrm{G-R}} \approx 50$ dt and $t_{\mathrm{check}} \approx 1000$ dt.

Furthermore, it should be emphasized that while Eq. (5) is mathematically valid for any values of t_{check} (in the sense that it indeed samples the exit time of the sub-sampled Markov chain $(X_{kt_{\mathrm{check}}})_{k\in\mathbb{N}}$), if t_{check} is taken too large, one may miss an exit event if the process re-enters the same state Ω during the time interval $t \to t + t_{\mathrm{check}}$; however one may argue that such cases may denote a poor definition of the states, and that for states exhibiting strong metastability this should not be an issue.

4. Results and discussion

Now that both the algorithm and the software implementation of the Generalized ParRep (in the following denoted as “Gen. ParRep”) have been extensively discussed, let us consider two applications: the first validates the implementation and consists in a study of the conformational equilibrium of alanine dipeptide (Section 4.1), while the second investigates the dissociation of the FKBP–DMSO protein–ligand complex (Section 4.2).

In the following, when reporting estimated values of the average exit time $\mathbb{E}(\tau)$ from a metastable state we will consider the sample average $\bar{\tau}$ over n samples $\{\tau_1, \ldots, \tau_n\}$ as

$$\bar{\tau} = \frac{1}{n}\sum_{i=1}^{n} \tau_i.$$

Furthermore the $1 - \alpha$ confidence interval for those (close to) exponentially distributed samples is:

$$\frac{2n\bar{\tau}}{\chi^2_{1-\frac{\alpha}{2},2n}} < \mathbb{E}(\tau) < \frac{2n\bar{\tau}}{\chi^2_{\frac{\alpha}{2},2n}}$$

where $\chi^2_{q,\nu}$ is the value of the quantile function of the χ^2 distribution with ν degrees of freedom at level q; in the following we chose $\alpha = 0.05$ and therefore report the 95% confidence interval.

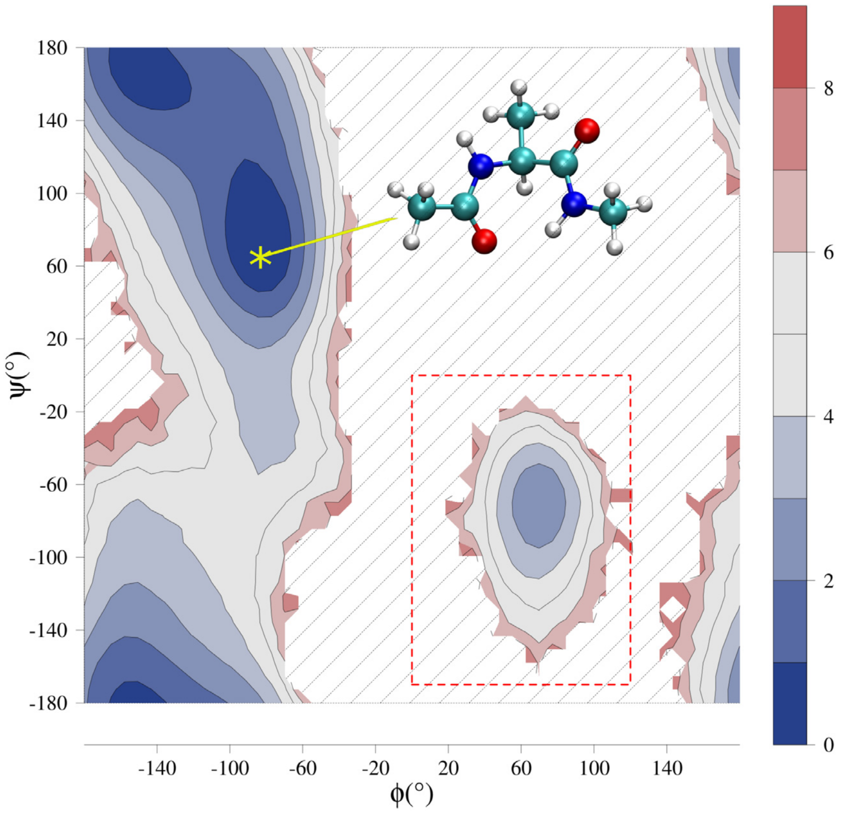


Fig. 3. Alanine dipeptide: definition of ParRep domains based on a free energy surface (color coded, in kcal/mol, dashed oblique lines correspond to unsampled areas), constructed from the long MD reference simulation. The red rectangle corresponding to $\phi \in [0; 120]$ and $\psi \in [-170; 0]$ is used as a threshold defining the C_{7ax} state (see Section 4.1 for details). The yellow cross corresponds to the starting configuration for either Gen. ParRep or serial MD simulations.

4.1. Conformational equilibrium of the alanine dipeptide

The blocked alanine dipeptide (Ac-Ala-N-H-Me) has been used as a validation system for computational studies of conformational equilibria, and energy landscape reconstruction and analysis [69–76]. The dipeptide contains several notable structural features, including the two (ϕ, ψ) dihedral angles, NH- and CO-groups capable of H-bond formation, and a methyl group attached to the C_α atom. One suitable way to visualize the conformations and the transitions between them is to draw an energy surface as a Ramachandran plot [77]: when studied *in vacuo* the following two metastable states are clearly identified: (i) C_{7eq} for $(\phi, \psi) \sim (-75°, 100°)$, and (ii) C_{7ax} for $(\phi, \psi) \sim (60°, -60°)$.

In the following we estimate the mean first passage time $\tau_{C_{7eq} \to C_{7ax}}$ between the two metastable states, using the Gen. ParRep algorithm; accuracy of the method is compared to one long serial Langevin dynamics, as the low complexity of this system allows direct numerical simulation of numerous transition events; finally the influence of some of the Gen. ParRep parameters is also evaluated.

4.1.1. MD setup

The initial configuration of the dipeptide is $(\phi, \psi) = (-81.0°, 70.0°)$, i.e. within the most populated area of the C_{7eq} state (see the yellow mark in Fig. 3 together with a representation of the corresponding conformation). The OpenMM system was configured as follows: the CHARMM 22 all-atoms for proteins and lipids force-field including CMAP corrections [78,79] was used; dynamics was performed using a Langevin integrator (time-step of $dt = 2$ fs, friction of $\gamma = 2$ ps^{-1}), thermostated at a temperature of $T = 300$ K; the non-bonded interactions were evaluated using a non-periodic cutoff scheme up to a distance of 1.6 nm; and bonds involving hydrogens are constrained to a value of $\pm 10^{-3}$% of their original distance.

4.1.2. Gen. ParRep setup

The procedure for defining the states may have to be adapted for each force-field and in the following we assume the use of the aforementioned CHARMM22 force-field. Fig. 3 is a Ramachandran plot based free energy surface built from preliminary serial Langevin MD simulation: it illustrates how the ParRep states were a priori defined.

One can see that in the upper left quarter of the plot two close stable conformations indeed coexist, separated by a low energetic barrier of 1 to 2 kcal/mol, which is comparable to the product k_BT: hence it was decided to combine those two minima together, as they do not constitute alone a valid metastable target for applying the ParRep method (Refs. [70,74,76] indeed suggest that the transition between those two wells is of 2.7, 3.0 and 4.05 ps, respectively). Therefore the conformational equilibrium of the dipeptide is modeled using a two states definition:

1. The C_{7ax} ParRep state corresponds to the well for which $\phi > 0°$ and $\psi < 0°$, i.e. the lower right quarter of Fig. 3; it was decided to model this state using the following rectangular domain:

   $$\phi \in [0; 120] \text{ and } \psi \in [-170; 0]$$

   represented as a red rectangle in Fig. 3.
2. The C_{7eq} ParRep state consists in the set of all configurations not falling within the red rectangle: it is therefore the complement of the state C_{7ax}.

Therefore this setup corresponds to a two states partition of the configuration space projected onto a Ramachandran plot.

Concerning the Fleming–Viot procedure, four Gelman–Rubin observables $\mathcal{O}$ are considered for tracking the convergence to the QSD: the total potential energy $V(q)$, the kinetic energy $K(p) =$

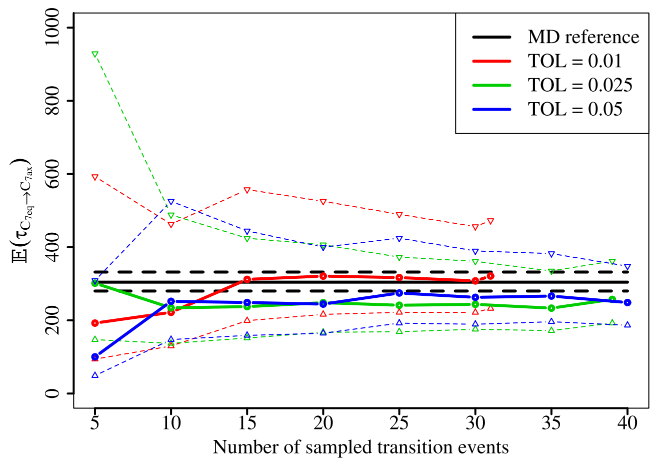


Fig. 4. Convergence of $\mathbb{E}(\tau_{C_{7eq}\to C_{7ax}})$ (plain lines) to the MD reference (black lines), for Gen. ParRep sampled values (red, green and blue lines), when considering only the first m of the $n = \{31, 39, 40\}$ samples (abscissa), for TOL levels of respectively $\{0.01, 0.025, 0.05\}$. Dashed lines correspond to the 95% confidence interval; The number of Gen. ParRep replicas was fixed at $N = 224$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

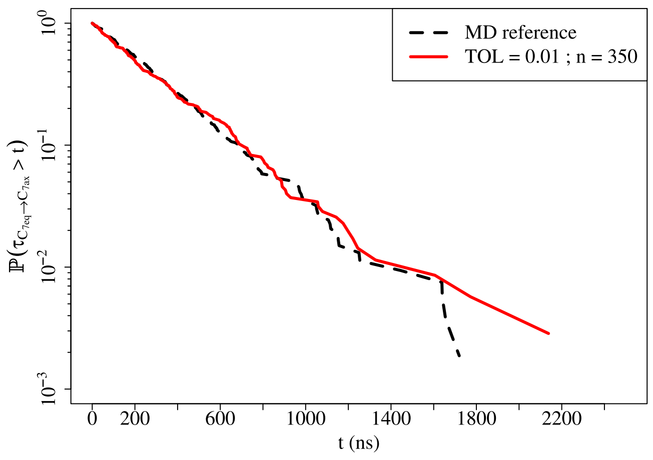


Fig. 5. Distribution of Gen. ParRep sampled values of $\tau_{C_{7eq}\to C_{7ax}}$ for TOL = 0.01, for $N = 224$, and a number of τ values generated $n = 350$.

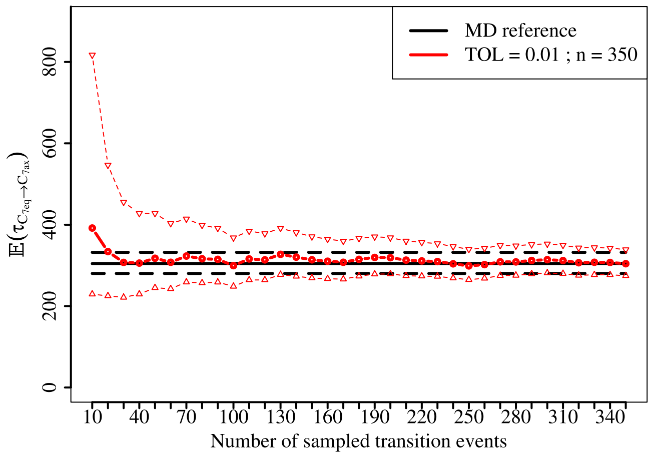


Fig. 6. Convergence of $\mathbb{E}(\tau_{C_{7eq}\to C_{7ax}})$ (solid lines) for Gen. ParRep sampled values from Fig. 5, when considering only the first m sampled values among $n = 350$ (abscissa). Dashed lines correspond to the 95% confidence interval.

$\frac{1}{2}p^T M^{-1} p$, and the value of the ϕ and ψ dihedral angles acting here as collective variables. The tolerance criterion TOL is fixed per simulation to a given value which is the same for each observable (the influence of TOL is investigated below for a range of values). The value of t_{G-R} (accumulation of observables) was set to $10 \times dt$ (i.e. 20 fs) as the observables are not computationally expansive to calculate, and the test $(X_t^{\text{ref}} \notin \Omega)$ is performed at $t_{\text{check}} = 250 \times dt$ (i.e. 0.5 ps) during the Convergence step, but at $t_{\text{check}} = 2500 \times dt$ during the Parallel dynamics step, which corresponds to 5 ps, in order to maximize the CPU time spent in the Langevin dynamics.

4.1.3. Discussion

In the following the distribution of the Gen. ParRep sampled values $\tau_{C_{7eq}\to C_{7ax}}$ is compared to results obtained when performing a long reference dynamics (denoted as *reference MD* in the following), consisting in a Langevin dynamics simulation of a total length of 162 —s. As a result, 533 $\tau_{C_{7eq}\to C_{7ax}}$ events were sampled, and $\mathbb{E}(\tau_{\text{MDref}}) = 304.47$ ns is obtained, while the confidence interval is $280.19 \text{ ns} < \mathbb{E}(\tau_{\text{MDref}}) < 332.07$ ns (see Table 1 for a summary); in Ref. [74] the authors estimate $\tau_{C_{7eq}\to C_{7ax}}$ to be of 353 ns (no provided error estimate), in agreement with this value.

First, the possibility to obtain an accurate estimate of $\tau_{C_{7eq}\to C_{7ax}}$ by generating a relatively small number n of samples is investigated: the number of Gen. ParRep replicas was set to $N = 224$, and the number n of samples of $\tau_{C_{7eq}\to C_{7ax}}$ generated was $\{31, 39, 40\}$ for respective tolerance levels of TOL $= \{0.01, 0.025, 0.05\}$ (red, green and blue solid lines; less samples were collected for TOL = 0.01 because of the higher computational effort required for lower tolerance values).

The convergence to the MD reference (black lines) can be visualized in Fig. 4 where $\mathbb{E}(\tau_{C_{7eq}\to C_{7ax}})$ (solid lines) and the corresponding confidence interval (dashed lines) are given for the three ParRep simulations, when considering only the subset of the first m sampled values; the red line (TOL = 0.01) quickly converges to the same distribution than the MD reference, while higher values of TOL appear to converge to a different distribution underestimating $\mathbb{E}(\tau)$ (see also numerical values in Table 1): this suggests that convergence to the QSD is only obtained for a value of TOL = 0.01 when studying the $C_{7eq} \to C_{7ax}$ transition.

Now that a value of TOL = 0.01 appears to be accurate enough, one can collect more samples n in order to verify that the distribution of the exit times converges to the one obtained with the reference MD simulation. In Fig. 5, the distribution of $n = 350$ samples generated for a level of TOL = 0.01 and using $N = 224$ is illustrated: this was done by building an empirical *complementary cumulative distribution function* (in the following referred to as *ccdf*) using the n samples, and it provides an estimate of the probability that $\tau_{C_{7eq}\to C_{7ax}}$ is higher than a given value t (i.e. $\mathbb{P}(\tau_{C_{7eq}\to C_{7ax}} > t)$), with by definition $\mathbb{P}(\tau_{C_{7eq}\to C_{7ax}} > 0) = 1$). One can see from Fig. 5 that the Gen. ParRep and MD distributions are in really good agreement for $t \in [0; 1500]$ ns, where a quasi linear function $t \mapsto \ln \mathbb{P}(\tau_{C_{7eq}\to C_{7ax}} > t)$ (i.e. exponential law) is observed (we ignore the area for $t > 1500$ ns, i.e. the low probability tail of the distribution corresponding to large values of $\tau_{C_{7eq}\to C_{7ax}}$, where the MD simulation lacks samples for performing a meaningful comparison). This observation is confirmed by looking at Fig. 6: the convergence of Gen. ParRep samples to the MD reference is observed, both for the mean value and the confidence interval, for increasing values of n.

4.1.4. Distribution of the F–V estimated value t_{F-V}

One last interesting quantity to collect is the time t_{F-V} necessary for the convergence of the observables $\mathcal{O}$ defined by Eq. (4) (see Section 2.4 for more details). Fig. 7 provides the histogram distribution of t_{F-V} for two of the datasets from Fig. 4, together

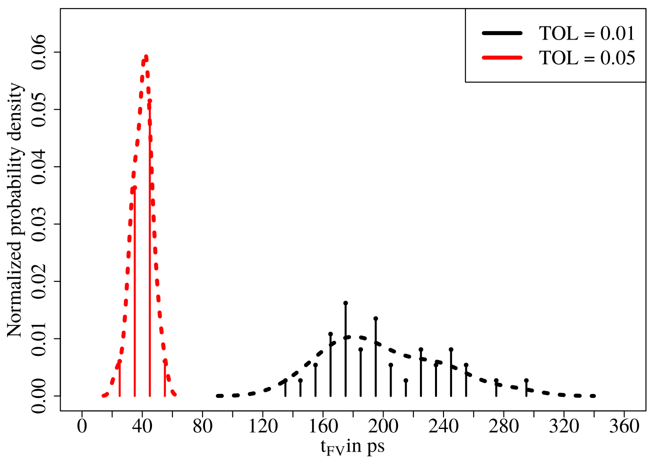


Fig. 7. Histogram distribution (vertical lines) of t_{F-V}, obtained for two different tolerance levels of TOL = {0.01, 0.05} (two datasets from Fig. 4): t_{F-V} corresponds to the simulation time before one assumes that the samples are distributed according to the QSD (see Section 2.4). The dashed lines correspond to a Kernel Density Estimation [80,81] smoothing.

with a Kernel Density Estimate smoothing [80,81] (dashed lines). For a tolerance of 0.05 one observes that $t_{F-V} \approx 40$ ps and appears to follow a normal distribution; for TOL = 0.01 it seems that the distribution is bimodal, with a major mode at $t_{F-V} \approx 180$ ps and a minor mode at $t_{F-V} \approx 240$ ps. While it is difficult to argue how the distribution of t_{F-V} ideally looks like, one should remember that the C_{7eq} is defined as the large funnel on the left ($\phi < 0°$) side of Fig. 3, and that therefore it encompasses the whole range of the possible ψ values, meaning that ψ will be the slowest observable to converge; it is thus expected that the value of t_{F-V} will be large for conservative tolerance levels (TOL $\to$ 0), and that depending on how the F–V workers randomly diffused on the (ϕ, ψ) surface, its distribution will be broad, possibly multimodal. Hence the dispersion for TOL = 0.01 in Fig. 7 appears coherent, while the homogeneous distribution for TOL = 0.05 probably indicates that the F–V workers did not diffuse far enough from their starting point in C_{7eq}: they are therefore still distant from what the QSD would be, and this explains why $\mathbb{E}(\tau_{C_{7eq}\to C_{7ax}})$ never converged to the result obtained by direct numerical simulation when TOL = 0.05.

Fig. 7 emphasizes one of the main advantages of the Gen. ParRep algorithm versus the original method, i.e. the fact that t_{F-V} is calculated on the fly, and thus adjusted to the initial condition within the state, whereas the original algorithm required a fixed user defined value after which it was assumed that the QSD was reached: indeed, one can see that for TOL = 0.01 (which seems necessary to be sufficiently close to the exact QSD), the distribution of t_{F-V} is spread over an interval going from 120 to 300 ps; it is therefore obvious that choosing a priori a decorrelation time of 120 ps would result in a bias as this value appears to be far below the time it takes to reach the QSD for some initial conditions; and on the contrary choosing a decorrelation time of 300 ps would ensure quasi-convergence to the QSD for most of the initial conditions, but at the cost of an unnecessary long (and then costly) decorrelation step for some of the initial conditions.

4.1.5. Performance

The last point to discuss concerns the performance of the Gen. ParRep method and particularly the speedup compared with the reference serial Langevin dynamics.

In Table 2 benchmarking data is reported for the simulations from Fig. 4; the fifth column reports the calculated effective speedup which is compared to the maximum possible speedup

Table 1
Summary of the estimated value of $\mathbb{E}(\tau_{C_{7eq}\to C_{7ax}})$ and of the corresponding 95% confidence interval for data presented in Figs. 4–6. The Gen. ParRep results (N = 224) appear to converge accurately for a value of TOL = 0.01.

Method	n	N	TOL	$\mathbb{E}(\tau)$ (ns)	Confidence interval (ns)
MD ref.	533	–	–	**304.47**	(280.19, 332.07)
Gen. ParRep	40	224	0.05	248.72	(186.60, 348.14)
Gen. ParRep	39	224	0.025	257.37	(192.45, 361.94)
Gen. ParRep	31	224	0.01	321.26	(232.54, 472.83)
Gen. ParRep	350	224	0.01	**304.22**	(274.70, 338.78)

Table 2
Benchmarking data for the three datasets from Fig. 4 (N = 224, n = {31, 39, 40} and TOL = {0.01, 0.025, 0.05}). Each replica runs on P = 1 CPU cores. The wall-clock time (column 2) is taken as the time elapsed from the beginning to end of the execution of the program, it includes both computations and communications time. The speed (ns/day, column 4) is obtained by dividing the total simulation clock t_{sim} (column 3) by the value of the wall-clock time (in days). The effective speedup (column 5) corresponds to column 4 divided by the performance of a serial MD reference (evaluated as 921 ns/day by independent tests on the same architecture). The ratio between the effective and maximum possible speedup (by definition equal to N = 224) is given as a percentage in column 6: a theoretical value of 100% would correspond to the maximum possible speedup. .

TOL	WT (s)	t_{sim} (ns)	Speed (ns/day)	Eff. speedup	(Eff./Max.)
0.01	6015	10 008	143 752	156	70%
0.025	5239	10 103	166 609	181	80%
0.05	4973	10 032	174 296	189	84%

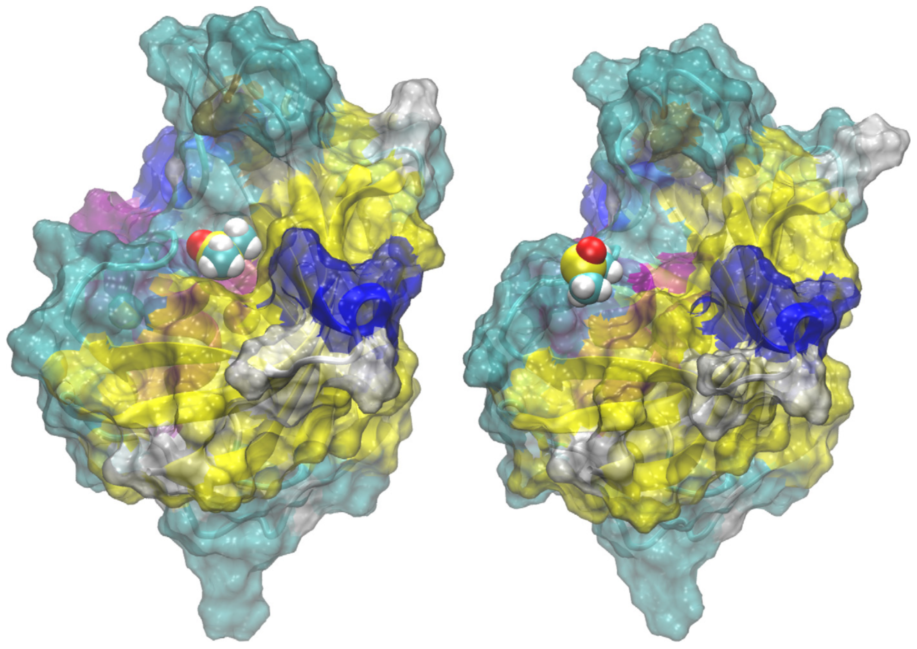
N = 224; the sixth column reports the ratio between the effective speedup (see table's caption for methodology) and the maximum possible speedup (hence a value of 100% would indicate a perfect linear speedup).

One can see that for a large tolerance of 0.05 a value of 189 is obtained, i.e. 84% of the maximum possible value; and for a more conservative tolerance criterion of 0.01 this falls to 156 i.e. 70% of N = 224: this illustrates the cost of an accurate convergence step which, as seen in the previous section, is the key for obtaining accurate results.

Considering the reduced size of the system (22 atoms) and the fact that during the F–V procedure the MD engine code is interrupted every 10 $\times$ dt for collecting the value of the G–R observables, this speedup is an impressive result; although slightly higher values may be obtained by tuning further the values of t_{G-R} and t_{check}, there is probably little space for optimization for such a small test-case system: therefore a more detailed performance analysis will be performed in the next subsection for the protein–ligand system.

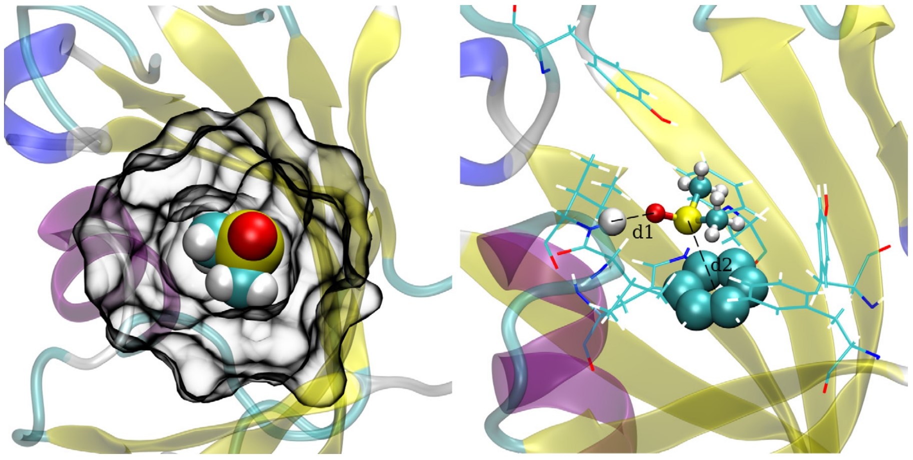
4.2. Dissociation of the FKBP–DMSO protein–ligand system

After validation of the Gen. ParRep algorithm on the alanine dipeptide, we would like to demonstrate its efficiency on a larger protein–ligand system: the aim is to sample the dissociation time τ_{off} between the *bound* and *unbound* states of the FKBP–DMSO complex (see Fig. 8). The FKBP protein (also known as the FK506 binding protein) has a role in the folding of other proteins containing proline residues [82]; in the human body the FKBP12 protein binds to the tacrolimus molecule (and derivatives), an immunosuppressant drug used to reduce organ rejection after an organ transplant [83]. Because of this important role, both experimental studies [84] and molecular dynamics simulations [85–87] were performed for evaluating the affinity of the FKBP protein to multiple ligands; these include the DMSO (Dimethyl-sulfoxide), a small molecule with anti-inflammatory, antioxidant and analgesic activities [88], and often used in topical treatments because of its membrane-penetrating ability, which enhances the diffusion of other substances through the skin [89].



(a) Bound state (b) Unbound state

Fig. 8. Illustration of the FKBP–DMSO complex, corresponding to the RCSB-PDB entry "1D7H": on the left the undissociated ("bound") state used as starting configuration for all the simulations; on the right the target dissociated ("unbound") state characterized by a τ_{off} dissociation time.



(a) DMSO in its binding cavity (b) Tracked distances

Fig. 9. On the left a closer view of the DMSO ligand in its binding cavity: favorable interactions between the O or S atoms and surface residues, together with the little available space around the ligand, are responsible for metastability. On the right, surrounding residues within the cavity are represented: in order to detect the dissociation event two distances are tracked for defining the bound state (see the Gen. ParRep setup paragraph for details).

4.2.1. *MD setup*

The initial configuration was taken from the RCSB-PDB entry "1D7H"; the AmberTools17 [63] software suite was used for setting up an implicit solvent input configuration (using the OBC [90] model II): first, parameters for the DMSO ligand were retrieved from the GAFF [91] force-field using the antechamber program; then parameters for the protein are taken from the ff14SB [92] force-field; dynamics was performed using a Langevin integrator (time-step of $dt = 2$ fs, friction of $\gamma = 2$ ps^{-1}), thermostated at a temperature of $T = 310$ K; the non-bonded interactions were evaluated using a non-periodic cutoff scheme up to a distance of 1.6 nm; and bonds involving hydrogens are constrained to a value of $\pm 10^{-3}$% of their original distance.

Before running ParRep simulations, the system was equilibrated for 1.0 ns, with the DMSO's center of mass being position-constrained within 0.36 nm of its original crystallographic position (force constant of 50 kJ/mol/nm^2).

4.2.2. *Gen. ParRep setup*

For defining the ParRep states, we used the following procedure, inspired from Refs. [85,87]: a closer view at the ligand binding cavity (see Fig. 9(a)) reveals a dense packing with only little available space around the ligand, and one expects that the sulfur and oxygen atoms will interact favorably via non-bonded interactions with the surface of the protein; when observing in detail the residues surrounding the DMSO (see Fig. 9(b) corresponding to the RCSB-PDB structure obtained from X-ray diffraction [84]), one can

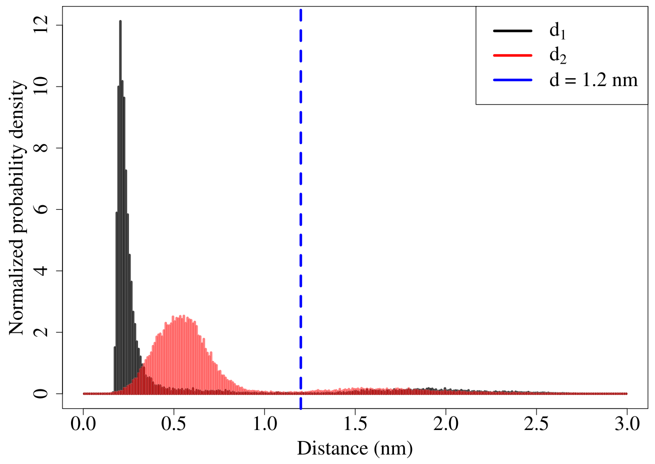


Fig. 10. Histogram distribution of the two d_1 and d_2 distances as defined in Fig. 9(b), for 30 ns of plain Langevin dynamics.

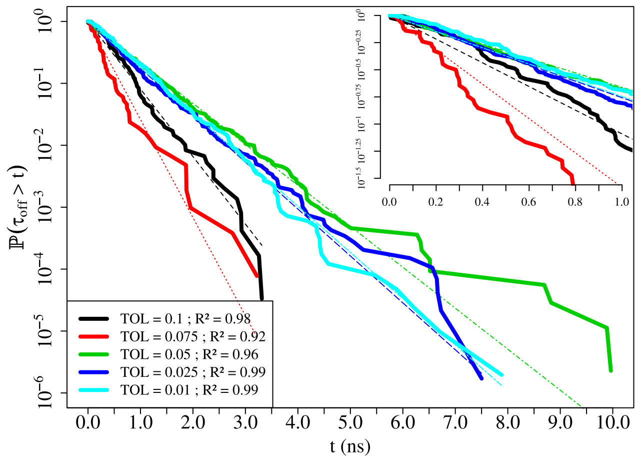


Fig. 11. Distribution of the dissociation time τ_{off} for the complex FKBP–DMSO, estimated using the Gen. ParRep method, at different levels of tolerance TOL $=\{0.1, 0.075, 0.05, 0.025, 0.01\}$. The top-right inset is a zoom for $t < 1.0$ ns, and dashed straight lines (for which the R^2 coefficient of determination is given) denote the expected quasi-exponential distribution for large t: $\ln \mathbb{P}(\tau_{\text{off}} > t) = -t/\mathbb{E}(\tau_{\text{off}})$. See Table 3 for quantitative values of: $\mathbb{E}(\tau_{\text{off}})$, N, n and the 95% confidence interval.

see favorable interaction of the O atom with residue ILE-56 and of the S atom with residue TRP-59.

Hence we used for defining the Gen. ParRep metastable "bound state" (denoted by b) a criterion based on the distances d_1 and d_2 as illustrated in Fig. 9(b): d_1 corresponds to the distance between ligand's oxygen and the hydrogen amide of residue ILE-56; and d_2 corresponds to the distance between ligand's sulfur and the center of mass of the carbons forming the ring of residue TRP-59. The DMSO is considered to be in the b state when any of d_1 or d_2 is less than 1.2 nm, and the "unbound state" (denoted by u) is simply defined as configurations where both distances are larger than 1.2 nm.

One may wonder whether this distance threshold $d < 1.2$ nm has a physical meaning: Fig. 10 shows a histogram distribution of the two distances d_1 and d_2, for a 30 ns Langevin dynamics, it appears that the threshold of 1.2 nm corresponds to rarely sampled configurations, far enough from the top of the two distributions ($\approx$0.25 and $\approx$0.55 nm, respectively), but still closer than the distance range around $d \geq 1.4$ nm, corresponding to unbound states. This threshold therefore appears to approximately correspond to a boundary between the b and u states.

Concerning the Fleming–Viot procedure, once again 4 observables were selected in order to track convergence to the QSD: the two first are the aforementioned distances d_1 and d_2; the third one is the distance between the center of mass of the DMSO and the center of mass of the protein; the last one is the root mean square velocity of the DMSO ligand. The levels of TOL were set to the same value for each of the observables, and this value will be the main Gen. ParRep parameter discussed below. The value of $t_{\text{G-R}}$ was set to $50 \times dt$, and the test ($X_t \notin \Omega$) is performed with a period $t_{\text{check}} = 1000 \times dt$, both during the Convergence step and the Parallel dynamics step.

4.2.3. Discussion

In the following we will compare the Gen. ParRep results to Ref. [87], where the authors performed long explicit water MD simulations using the CHARMM 27 force-field and where a value of $\mathbb{E}(\tau_{\text{off}}) = 2.2$ ns is reported.

In Fig. 11 the distribution of τ_{off} for tolerance values of TOL $=\{0.1, 0.075, 0.05, 0.025, 0.01\}$ is shown (details are available in Table 3): first, a moderate number of transition events ($n < 100$) was generated for each level of TOL, and a first estimate of $\mathbb{E}(\tau_{\text{off}})$ was calculated: as the results for TOL $=\{0.1, 0.075\}$ appeared to be far from the results obtained for more stringent tolerances, they were not further considered; then for the three remaining tolerance levels (TOL $=\{0.05, 0.025, 0.01\}$), extended simulations were performed which permitted to obtain $n = \{282, 320, 301\}$ samples of τ_{off}, thus providing estimates $\mathbb{E}(\tau_{\text{off}})$ of respectively 1.51, 1.32 and 1.34 ns with a confidence interval of approximately $\pm$0.3 ns (see Table 3); the convergence of the corresponding simulations can be observed in Fig. 12.

It should first be noted that levels of TOL larger than 0.05 have to be avoided for this system (and, from our experience, for any application in general) as they will systematically produce biased results: it is indeed not possible to generate initial conditions distributed according to the QSD by using such a loose tolerance criterion, especially when the definition of the state Ω involves a large number of degrees of freedom.

For tolerance levels smaller or equal to 0.05, the confidence intervals mostly overlap, as one can see in Fig. 12: for TOL $= 0.01$ or 0.025 (respectively the cyan and dark blue lines) the two estimated values of 1.32 and 1.34 ns are almost identical, for TOL $= 0.05$ (the green line) the estimated value of $\mathbb{E}(\tau_{\text{off}})$ is 1.51, a slightly higher value. A closer look at the upper and lower bounds of the confidence intervals shows a constant overlap – of decreasing width – around 1.35 ns, which is the value of $\mathbb{E}(\tau_{\text{off}})$ for TOL $= 0.01$ or 0.025 (1.34 and 1.32 ns), therefore suggesting that, once again, strict tolerance criteria provide the most accurate estimate; this is confirmed by a qualitative (solid vs dashed lines) and quantitative (coefficient R^2) look at Fig. 11, where one can see that the two lowest values of TOL follow the more accurately the quasi-exponential distribution (however, one should remember that the distribution is not expected to be exactly exponential especially for small values of τ_{off}, as exit events of the reference walker happening before the end of the Convergence step are not guaranteed to be exponentially distributed as the QSD was not yet reached).

In the aforementioned Ref. [87] the estimate of $\mathbb{E}(\tau_{\text{off}})$ is 2.2 ns (no confidence interval provided): this can be considered to be in a reasonable agreement with our value of 1.34 ns obtained for TOL $=$ 0.01 and where the 95% confidence interval is (1.20, 1.51) ns, considering that the force-field was different, and that the current study uses an implicit solvent while the reference used explicit water molecules.

4.2.4. Performance and convergence to the QSD

Because the FKBP–DMSO system is much more representative of a typical research application than the alanine dipeptide, it is of

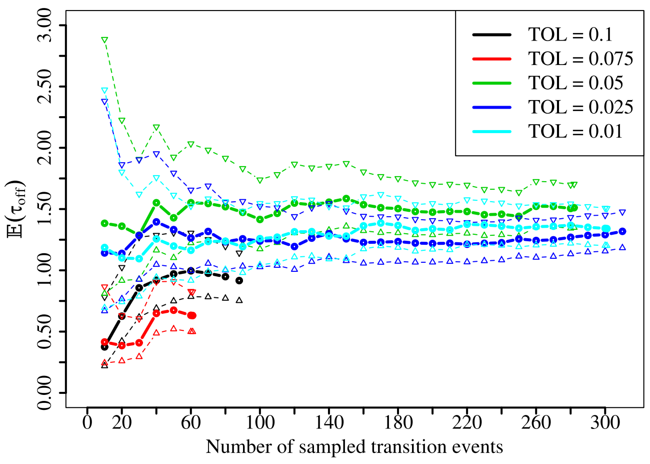


Fig. 12. Convergence of $\mathbb{E}(\tau_{\mathrm{off}})$ for the datasets from Fig. 11 and Table 3. Dashed lines correspond to the 95% confidence interval.

Table 3
Summary of the estimated value of $\mathbb{E}(\tau_{\mathrm{off}})$ and of the corresponding 95% confidence interval for data presented in Figs. 11 and 12.

TOL	N	n	$\mathbb{E}(\tau_{\mathrm{off}})$ (ns)	Confidence interval (ns)
0.1	112	88	0.92	(0.75, 1.14)
0.075	112	61	0.63	(0.50, 0.83)
0.05	140	282	1.51	(1.35, 1.70)
0.025	140	320	1.32	(1.19, 1.48)
0.01	140	301	1.34	(1.20, 1.51)

Table 4
Benchmarking data for three of the datasets from Fig. 11 and Table 3 (corresponding to $N = 140$, TOL $= \{0.01, 0.025, 0.05\}$): each replica used $P = 4$ CPUs cores, and the equivalent speed of a reference Langevin dynamics on those same 4 cores is 5.15 ns/day; see Table 2 for the methodology.

TOL	WT (s)	t_{sim} (ns)	Speed (ns/day)	Eff. speedup	(Eff./Max.)
0.01	85 142	403.5	409.4	79.5	56.8%
0.025	79 574	457.6	496.8	96.5	68.9%
0.05	84 455	482.2	493.4	95.8	68.4%

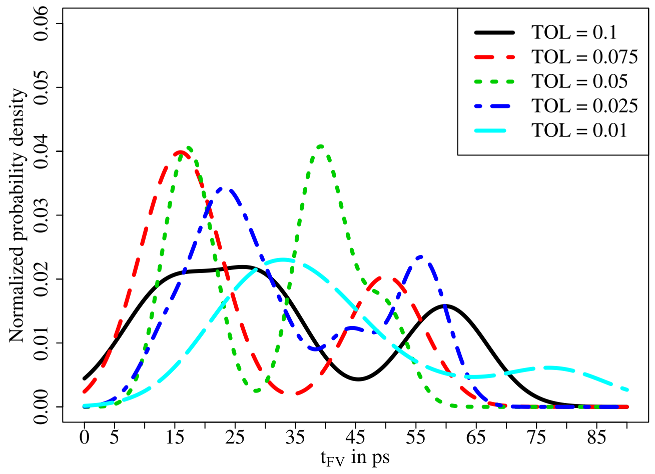


Fig. 13. Kernel Density Estimation of the distribution of $t_{\mathrm{F-V}}$, obtained for tolerance levels of TOL $= \{0.1, 0.075, 0.05, 0.025, 0.01\}$ (datasets from Fig. 11 and Table 3).

an utmost interest to provide an accurate estimate of the performances: for this, we provide in Table 4 benchmarking data (following the same methodology established for Table 2): the three datasets correspond to the samples for TOL $= \{0.01, 0.025, 0.05\}$ from Fig. 11; the number of replica $N = 140$ corresponds to the maximum speedup, while column 5 reports the calculated effective speedup; the ratio given in column 6 gives an idea of the efficiency of the implementation for studying this medium size protein–ligand system.

One can see that the effective speedup is close to 97 (i.e. ≈69% of the maximum $N = 140$) for tolerances of 0.025 and 0.05; for the stricter tolerance level of 0.01 the speedup falls to ≈80 (i.e. ≈57% of the maximum $N = 140$) which illustrates the computational cost one has to pay for an increased accuracy. Once again the ability to obtain a speedup between 57 and 69% of the maximum possible denotes the parallel efficiency of the Gen. ParRep implementation on a production system, and is an extremely promising achievement towards future studies of larger biochemical systems.

The distribution of $t_{\mathrm{F-V}}$ is illustrated in Fig. 13 for all the considered levels of tolerance: all distributions appear to be multimodal, revealing that multiple sub-states are likely to be found within the surrounding cavity definition of the bound state (this was suggested in earlier studies such as Ref. [86]); for the larger levels of TOL the multimodality is particularly visible with two well defined peaks, resulting in an average $t_{\mathrm{F-V}}$ falling between the peaks, around ≈25 ps; however for low tolerance such as 0.01 a broad distribution is observed, and the average $t_{\mathrm{F-V}}$ goes to ≈50 ps.

This emphasizes once again how difficult it would be to fix a priori a value for $t_{\mathrm{F-V}}$ (as required by the original ParRep implementations), as this value would be inappropriate for some of the initial conditions; the ability of the Gen. ParRep algorithm to automatically determine a value of $t_{\mathrm{F-V}}$ appropriate for the current initial condition therefore appears to be a major advantage of the method when investigating protein–ligand complexes' dissociation.

5. Conclusion and outlook

In this article, we detailed a new implementation of the Gen. ParRep algorithm, developed with the aim of facilitating the study of biochemical systems exhibiting strong metastability. After detailing the methods and the software implementation in Sections 2 and 3, a validation (Section 4) with two systems of increasing complexity was discussed.

In Section 4.1, it was shown that the Gen. ParRep method can accurately sample the transition time $\tau_{C_{7eq}\to C_{7ax}}$ characterizing the conformational equilibrium of alanine dipeptide in vacuum: for sufficiently small levels of tolerance (e.g. TOL $= 0.01$), the estimation converges to what was obtained from a long reference Langevin dynamics (see Table 1 for a summary, and Figs. 3–7). Results also appeared to compare favorably to previously published studies [74]. Finally it was also shown that the implementation of the algorithm proves to be scalable as one can obtain ≈80% of the maximum possible speedup (see Table 2).

The second application consisted in the study of the dissociation of the FKBP–DMSO complex (Section 4.2), a protein–ligand system of larger size, much more representative of typical metastable problems encountered in computational biology or chemistry. The goal was to obtain an accurate estimate of the average time $\mathbb{E}(\tau_{\mathrm{off}})$ required for observing a dissociation of the complex, with comparison to previous computational studies [85–87]. It was shown (see Table 3 and associated Figs. 8–13) that a simple definition of the bound and unbound states based on a two distances threshold can provide an accurate estimate of $\mathbb{E}(\tau_{\mathrm{off}})$, once again when tolerance levels TOL < 0.05 are used: a value of $1.32 < \mathbb{E}(\tau_{\mathrm{off}}) < 1.51$ is found using the Gen. ParRep method, the value of 1.34 ns for TOL $= 0.01$ appearing to be the most accurate, and this compares relatively well to Ref. [87] where a value of 2.2 ns was found using a different force-field and an explicit solvent. The algorithm was also benchmarked for the FKBP–DMSO system (see Table 4), and it was shown that one can maintain performances up to 60%–70% of the maximum possible speedup on 560 CPU cores, which definitely makes this new Gen. ParRep ready for production on large scale HPC machines.

From the two studies performed in this article it ought to be remembered that, beyond the accurate definition of the states $\mathcal{S}$, the choice of the tolerance level is the main parameter influencing the accuracy of the results: a value of $\text{TOL} = 0.01$ appears to be the most reasonable choice, confirming previous observations on smaller systems [43]. Furthermore, the algorithm provides an accurate estimate of the time $t_{\text{F-V}}$ required for approximating the QSD depending on the initial condition within the state, and it was shown (see Figs. 7 and 13) that $t_{\text{F-V}}$ is distributed over a large interval of time: in such a case the a priori choice of a fixed value decorrelation time approximating $t_{\text{F-V}}$ (as it was done in the original ParRep implementations) is non-obvious, and the use of the Gen. ParRep method is justified.

As an outlook, it has to be emphasized that there is still, of course, place for improvement of the software implementation: the authors would like to make the program compatible with other MD engines; tests are currently being performed where replicas are distributed over General-Purpose computing units (GPGPUs) in order to consider applications to larger systems; and preliminary simulations are being performed on a HPC Cloud Computing platform, on which a user could easily use thousands of replicas.

The authors are also currently studying more advanced biochemical metastable problems, including larger protein–ligand systems in explicit water where the states consist in a set of disjoint cavities inside a protein.

Acknowledgments

- This work is supported by the European Research Council under the European Union's Seventh Framework Program (FP/2007-2013)/ERC Grant Agreement number 614492.
- The authors acknowledge the "Maison de la Simulation" (and particularly Matthieu Haefele and Julien Derouillat) for helpful discussion and advices concerning the optimization of the software.
- This work was granted access to the HPC resources of the CINES under the GENCI allocations 2017-AP010710193, 2017-AP010710245 and 2017-A0030710275.
- FH acknowledges the Institute for Pure and Applied Mathematics (IPAM) at the University of California Los Angeles (UCLA) for participation in the long program "Complex High-Dimensional Energy Landscapes", its organizers, participants, and particularly members of the "Accelerated dynamics and benchmarking" working group for discussion and remarks on this work.
- The authors benefited from the scientific environment of the "Laboratoire International Associé" between the "Centre National de la Recherche Scientifique" and the "University of Illinois at Urbana-Champaign".

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cpc.2019.01.005.

References

[1] A. Hospital, J.R. Goñi, M. Orozco, J.L. Gelpi, Adv. Appl. Bioinform. Chem. 8 (2015) 37–47, http://dx.doi.org/10.2147/AABC.S70333.

[2] M. De Vivo, M. Masetti, G. Bottegoni, A. Cavalli, J. Med. Chem. 59 (9) (2016) 4035–4061, http://dx.doi.org/10.1021/acs.jmedchem.5b01684.

[3] V. Lounnas, T. Ritschel, J. Kelder, R. McGuire, R.P. Bywater, N. Foloppe, Comput. Struct. Biotechnol. J. 5 (6) (2013) e201302011, http://dx.doi.org/10.5936/csbj.201302011, URL http://www.sciencedirect.com/science/article/pii/S2001037014600398.

[4] G. Sliwoski, S. Kothiwale, J. Meiler, E.W. Lowe, Pharmacol. Rev. 66 (1) (2014) 334–395, http://dx.doi.org/10.1124/pr.112.007336.

[5] H.I. Ingólfsson, C.A. Lopez, J.J. Uusitalo, D.H. de Jong, S.M. Gopal, X. Periole, S.J. Marrink, WIREs. Comput. Mol. Sci. 4 (3) (2014) 225–248, http://dx.doi.org/10.1002/wcms.1169.

[6] S. Kmiecik, D. Gront, M. Kolinski, L. Wieteska, A.E. Dawid, A. Kolinski, Chem. Rev. 116 (14) (2016) 7898–7936, http://dx.doi.org/10.1021/acs.chemrev.6b00163.

[7] B. Kuhn, P. Mohr, M. Stahl, J. Med. Chem. 53 (6) (2010) 2601–2611, http://dx.doi.org/10.1021/jm100087s.

[8] C. Bissantz, B. Kuhn, M. Stahl, J. Med. Chem. 53 (14) (2010) 5061–5084, http://dx.doi.org/10.1021/jm100112j.

[9] H. Jónsson, G. Mills, K.W. Jacobsen, Classical and Quantum Dynamics in Condensed Phase Simulations, World Scientific, 2011, pp. 385–404, http://dx.doi.org/10.1142/9789812839664_0016, URL http://www.worldscientific.com/doi/abs/10.1142/9789812839664_0016.

[10] W. E, W. Ren, E. Vanden-Eijnden, Phys. Rev. B 66 (5) (2002) 052301, http://dx.doi.org/10.1103/PhysRevB.66.052301.

[11] R. Zhao, J. Shen, R.D. Skeel, J. Chem. Theory Comput. 6 (8) (2010) 2411–2423, http://dx.doi.org/10.1021/ct900689m.

[12] G.A. Huber, S. Kim, Biophys. J. 70 (1) (1996) 97, http://dx.doi.org/10.1016/S0006-3495(96)79552-8.

[13] M.C. Zwier, J.L. Adelman, J.W. Kaus, A.J. Pratt, K.F. Wong, N.B. Rego, E. Suárez, S. Lettieri, D.W. Wang, M. Grabe, D.M. Zuckerman, L.T. Chong, J. Chem. Theory Comput. 11 (2) (2015) 800–809, http://dx.doi.org/10.1021/ct5010615.

[14] C. Dellago, P.G. Bolhuis, D. Chandler, J. Chem. Phys. 110 (14) (1999) 6617–6625, http://dx.doi.org/10.1063/1.478569.

[15] F. Cérou, A. Guyader, Stoch. Anal. Appl. 25 (2) (2007) 417–443, http://dx.doi.org/10.1080/07362990601139628.

[16] F. Cérou, A. Guyader, T. Lelièvre, D. Pommier, J. Chem. Phys. 134 (5) (2011) 054108, http://dx.doi.org/10.1063/1.3518708.

[17] C.-E. Bréhier, T. Lelièvre, M. Rousset, ESAIM Probab. Stat. 19 (2015) 361–394, http://dx.doi.org/10.1051/ps/2014029.

[18] I. Teo, C.G. Mayne, K. Schulten, T. Lelièvre, J. Chem. Theory Comput. 12 (6) (2016) 2983–2989, http://dx.doi.org/10.1021/acs.jctc.6b00277.

[19] T.S. van Erp, P.G. Bolhuis, J. Comput. Phys. 205 (1) (2005) 157–181, http://dx.doi.org/10.1016/j.jcp.2004.11.003.

[20] R.J. Allen, C. Valeriani, P.R.t. Wolde, J. Phys.: Condens. Matter 21 (46) (2009) 463102, http://dx.doi.org/10.1088/0953-8984/21/46/463102.

[21] A.K. Faradjian, R. Elber, J. Chem. Phys. 120 (23) (2004) 10880–10889, http://dx.doi.org/10.1063/1.1738640.

[22] L. Maragliano, E. Vanden-Eijnden, B. Roux, J. Chem. Theory Comput. 5 (10) (2009) 2589–2594, http://dx.doi.org/10.1021/ct900279z.

[23] C. Schütte, F. Noé, J. Lu, M. Sarich, E. Vanden-Eijnden, J. Chem. Phys. 134 (20) (2011) 204105, http://dx.doi.org/10.1063/1.3590108.

[24] J.M. Bello-Rivas, R. Elber, J. Comput. Chem. 37 (6) (2016) 602–613, http://dx.doi.org/10.1002/jcc.24039.

[25] D. Aristoff, J.M. Bello-Rivas, R. Elber, Multiscale Model. Simul. (2016) http://dx.doi.org/10.1137/15M102157X.

[26] G. Grazioli, I. Andricioaei, J. Chem. Phys. 149 (8) (2018) 084103, http://dx.doi.org/10.1063/1.5029954.

[27] G. Grazioli, I. Andricioaei, J. Chem. Phys. 149 (8) (2018) 084104, http://dx.doi.org/10.1063/1.5037482.

[28] A.F. Voter, M.R. Sørensen, MRS Online Proc. Libr. Arch. 538 (1998) http://dx.doi.org/10.1557/PROC-538-427.

[29] D. Perez, B.P. Uberuaga, Y. Shim, J.G. Amar, A.F. Voter, in: R.A. Wheeler (Ed.), Annual Reports in Computational Chemistry, Vol. 5, Elsevier, 2009, pp. 79–98, http://dx.doi.org/10.1016/S1574-1400(09)00504-0, URL http://www.sciencedirect.com/science/article/pii/S1574140009005040.

[30] T. Lelièvre, Eur. Phys. J. Spec. Top. 224 (12) (2015) 2429–2444, http://dx.doi.org/10.1140/epjst/e2015-02420-1.

[31] T. Lelièvre, G. Stoltz, Acta Numer. 25 (2016) 681–880, http://dx.doi.org/10.1017/S0962492916000039.

[32] T. Lelièvre, Mathematical Foundations of Accelerated Molecular Dynamics Methods, in: W. Andreoni, S. Yip (Eds.), Handbook of Materials Modeling, Springer, Cham, 2018, URL https://link.springer.com/referenceworkentry/10.1007/978-3-319-42913-7_27-1.

[33] A.F. Voter, Phys. Rev. B 57 (1998) R13985–R13988, http://dx.doi.org/10.1103/PhysRevB.57.R13985, URL https://link.aps.org/doi/10.1103/PhysRevB.57.R13985.

[34] O. Kum, B.M. Dickson, S.J. Stuart, B.P. Uberuaga, A.F. Voter, J. Chem. Phys. 121 (20) (2004) 9808–9819, http://dx.doi.org/10.1063/1.1807823.

[35] D. Perez, E.D. Cubuk, A. Waterland, E. Kaxiras, A.F. Voter, J. Chem. Theory Comput. 12 (1) (2016) 18–28, http://dx.doi.org/10.1021/acs.jctc.5b00916.

[36] D. Perez, ParSplice, version 1, version 00, 2017, URL https://www.osti.gov//servlets/purl/1338199.

[37] A.F. Voter, J. Chem. Phys. 106 (11) (1997) 4665–4677, http://dx.doi.org/10.1063/1.473503.

[38] A.F. Voter, Phys. Rev. Lett. 78 (20) (1997) 3908–3911, http://dx.doi.org/10.1103/PhysRevLett.78.3908.
[39] M.R. Sørensen, A.F. Voter, J. Chem. Phys. 112 (21) (2000) 9599–9606, http://dx.doi.org/10.1063/1.481576.
[40] C.L. Bris, T. Lelièvre, M. Luskin, D. Perez, Monte Carlo Methods Appl. 18 (2) (2012) 119–146, http://dx.doi.org/10.1515/mcma-2012-0003, arXiv:1105.4636.
[41] P.A. Ferrari, H. Kesten, S. Martinez, P. Picco, Ann. Probab. 23 (2) (1995) 501–521, URL http://www.jstor.org/stable/2244994.
[42] E. van Doorn, P. Pollett, Quasi-stationary distributions, Memorandum / Department of Applied Mathematics, no. 1945, University of Twente, Department of Applied Mathematics, 2011, URL https://research.utwente.nl/en/publications/quasi-stationary-distributions.
[43] A. Binder, T. Lelièvre, G. Simpson, J. Comput. Phys. 284 (2015) 595–616, http://dx.doi.org/10.1016/j.jcp.2015.01.002, arXiv:1404.6191.
[44] G. Fiorin, M.L. Klein, J. Hénin, Mol. Phys. 111 (22-23, SI) (2013) 3345–3362, http://dx.doi.org/10.1080/00268976.2013.813594.
[45] G.A. Tribello, M. Bonomi, D. Branduardi, C. Camilloni, G. Bussi, Comput. Phys. Comm. 185 (2) (2014) 604–613, http://dx.doi.org/10.1016/j.cpc.2013.09.018, URL http://www.sciencedirect.com/science/article/pii/S0010465513003196.
[46] W.H. Fleming, M. Viot, Indiana Univ. Math. J. 28 (5) (1979) 817–843, URL http://www.jstor.org/stable/24892583.
[47] A. Asselah, P.A. Ferrari, P. Groisman, J. Appl. Probab. 48 (2) (2011) 322–332, http://dx.doi.org/10.1239/jap/1308662630.
[48] A. Gelman, D.B. Rubin, Statist. Sci. 7 (4) (1992) 457–472, URL http://www.jstor.org/stable/2246093.
[49] S.P. Brooks, A. Gelman, J. Comput. Graph. Statist. 7 (4) (1998) 434–455, http://dx.doi.org/10.1080/10618600.1998.10474787, URL http://www.tandfonline.com/doi/abs/10.1080/10618600.1998.10474787.
[50] D. Aristoff, T. Lelièvre, G. Simpson, Appl. Math. Res. eXpress 2014 (2) (2014) 332–352, http://dx.doi.org/10.1093/amrx/abu005.
[51] MPI: a message-passing interface standard, Tech. rep., University of Tennessee, The MPI forum, Knoxville, TN, USA, 1994.
[52] E. Gabriel, G.E. Fagg, G. Bosilca, T. Angskun, J.J. Dongarra, J.M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R.H. Castain, D.J. Daniel, R.L. Graham, T.S. Woodall, Proceedings, 11th European PVM/MPI Users' Group Meeting, Budapest, Hungary, 2004, pp. 97–104.
[53] W. Gropp, Proceedings of the 9th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, Springer-Verlag, London, UK, UK, 2002, p. 7, URL http://dl.acm.org/citation.cfm?id=648139.749473.
[54] T. Darden, D. York, L. Pedersen, J. Chem. Phys. 98 (12) (1993) 10089–10092, http://dx.doi.org/10.1063/1.464397.
[55] F.S. Lee, A. Warshel, J. Chem. Phys. 97 (5) (1992) 3100–3107, http://dx.doi.org/10.1063/1.462997.
[56] I.G. Tironi, R. Sperb, P.E. Smith, W.F. van Gunsteren, J. Chem. Phys. 102 (13) (1995) 5451–5459, http://dx.doi.org/10.1063/1.469273.
[57] W. Mattson, B.M. Rice, Comput. Phys. Comm. 119 (2) (1999) 135–148, http://dx.doi.org/10.1016/S0010-4655(98)00203-3.
[58] Z. Yao, J.S. Wang, G.R. Liu, M. Cheng, Comput. Phys. Comm. 161 (1) (2004) 27–35, http://dx.doi.org/10.1016/j.cpc.2004.04.004.
[59] T.N. Heinz, P.H. Hünenberger, J. Comput. Chem. 25 (12) (2004) 1474–1486, http://dx.doi.org/10.1002/jcc.20071.
[60] P. Gonnet, J. Comput. Chem. 28 (2) (2007) 570–573, http://dx.doi.org/10.1002/jcc.20563.
[61] P. Eastman, J. Swails, J.D. Chodera, R.T. McGibbon, Y. Zhao, K.A. Beauchamp, L.-P. Wang, A.C. Simmonett, M.P. Harrigan, C.D. Stern, R.P. Wiewiora, B.R. Brooks, V.S. Pande, PLoS Comput. Biol. 13 (7) (2017) 1–17, http://dx.doi.org/10.1371/journal.pcbi.1005659.
[62] B.R. Brooks, I. C. L. Brooks, A.D. MacKerell Jr., L. Nilsson, R.J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A.R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R.W. Pastor, C.B. Post, J.Z. Pu, M. Schaefer, B. Tidor, R.M. Venable, H.L. Woodcock, X. Wu, W. Yang, D.M. York, M. Karplus, J. Comput. Chem. 30 (10) (2009) 1545–1614, http://dx.doi.org/10.1002/jcc.21287.
[63] D.A. Case, D.S. Cerutti, T.E. Cheatham, T.A. Darden, R.E. Duke, T.J. Giese, H. Gohlke, A.W. Goetz, D. Greene, N. Homeyer, S. Izadi, A. Kovalenko, T.S. Lee, S. LeGrand, P. Li, C. Lin, J. Liu, T. Luchko, R. Luo, D. Mermelstein, K.M. Merz, G. Monard, H. Nguyen, I. Omelyan, A. Onufriev, F. Pan, R. Qi, D.R. Roe, A. Roitberg, C. Sagui, C.L. Simmerling, W.M. Botello-Smith, J. Swails, R.C. Walker, J. Wang, R.M. Wolf, X. Wu, L. Xiao, D.M. York, P.A. Kollman, Amber16 and amberTools17, Software, University of California, San Francisco, 2017, http://ambermd.org/.
[64] M.J. Abraham, T. Murtola, R. Schulz, S. Páll, J.C. Smith, B. Hess, E. Lindahl, SoftwareX 1–2 (2015) 19–25, http://dx.doi.org/10.1016/j.softx.2015.06.001.
[65] J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kalé, K. Schulten, J. Comput. Chem. 26 (16) (2005) 1781–1802, http://dx.doi.org/10.1002/jcc.20289.
[66] R. Ierusalimschy, L.H. de Figueiredo, W.C. Filho, Softw. - Pract. Exp. 26 (6) (1996) 635–652, http://dx.doi.org/10.1002/(SICI)1097-024X(199606)26:6<635::AID-SPE26>3.0.CO;2-P.
[67] ThePhD, Sol2: a C++ <-> lua API wrapper, Software, version 2.17.5, 2017, URL https://github.com/ThePhD/sol2.
[68] M. Pall, LuaJIT, a just-in-time compiler (JIT) for the Lua programming language, Software, version 2.0.5, 2017 URL http://luajit.org/.
[69] J. Apostolakis, P. Ferrara, A. Caflisch, J. Chem. Phys. 110 (4) (1999) 2099–2108, http://dx.doi.org/10.1063/1.477819.
[70] H.M. Chun, C.E. Padilla, D.N. Chin, M. Watanabe, V.I. Karlov, H.E. Alper, K. Soosaar, K.B. Blair, O.M. Becker, L.S.D. Caves, R. Nagle, D.N. Haney, B.L. Farmer, J. Comput. Chem. 21 (3) (2000) 159–184, http://dx.doi.org/10.1002/(SICI)1096-987X(200002)21:33.0.CO;2-J.
[71] W.C. Swope, J.W. Pitera, F. Suits, M. Pitman, M. Eleftheriou, B.G. Fitch, R.S. Germain, A. Rayshubski, T.J.C. Ward, Y. Zhestkov, R. Zhou, J. Phys. Chem. B 108 (21) (2004) 6582–6594, http://dx.doi.org/10.1021/jp037422q.
[72] W. Ren, E. Vanden-Eijnden, P. Maragakis, W. E, J. Chem. Phys. 123 (13) (2005) 134109, http://dx.doi.org/10.1063/1.2013256.
[73] H. Jang, T.B. Woolf, J. Comput. Chem. 27 (11) (2006) 1136–1141, http://dx.doi.org/10.1002/jcc.20444.
[74] B. Strodel, D.J. Wales, Chem. Phys. Lett. 466 (4) (2008) 105–115, http://dx.doi.org/10.1016/j.cplett.2008.10.085.
[75] P.T. Vedell, Z. Wu, Int. J. Numer. Anal. Model. 10 (4) (2013) 920–942, URL http://www.math.ualberta.ca/ijnam/Volume-10-2013/No-4-13/2013-04-11.pdf.
[76] C. Velez-Vega, E.E. Borrero, F.A. Escobedo, J. Chem. Phys. 130 (22) (2009) 225101, http://dx.doi.org/10.1063/1.3147465.
[77] G.N. Ramachandran, C. Ramakrishnan, V. Sasisekharan, J. Mol. Biol. 7 (1) (1963) 95–99, http://dx.doi.org/10.1016/S0022-2836(63)80023-6.
[78] A.D. MacKerell, D. Bashford, M. Bellott, R.L. Dunbrack, J.D. Evanseck, M.J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F.T.K. Lau, C. Mattos, S. Michnick, T. Ngo, D.T. Nguyen, B. Prodhom, W.E. Reiher, B. Roux, M. Schlenkrich, J.C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiórkiewicz-Kuczera, D. Yin, M. Karplus, J. Phys. Chem. B 102 (18) (1998) 3586–3616, http://dx.doi.org/10.1021/jp973084f, PMID: 24889800.
[79] A.D. Mackerell, M. Feig, C.L. Brooks, J. Comput. Chem. 25 (11) (2004) 1400–1415, http://dx.doi.org/10.1002/jcc.20065.
[80] M. Rosenblatt, Ann. Math. Stat. 27 (3) (1956) 832–837, http://dx.doi.org/10.1214/aoms/1177728190.
[81] E. Parzen, Ann. Math. Stat. 33 (3) (1962) 1065–1076, http://dx.doi.org/10.1214/aoms/1177704472.
[82] J.J. Siekierka, S.H.Y. Hung, M. Poe, C.S. Lin, N.H. Sigal, Nature 341 (6244) (1989) 755, http://dx.doi.org/10.1038/341755a0.
[83] T. Wang, P. Donahoe, A. Zervos, Science 265 (5172) (1994) 674–676, http://dx.doi.org/10.1126/science.7518616.
[84] P. Burkhard, P. Taylor, M.D. Walkinshaw, J. Mol. Biol. 295 (4) (2000) 953–962, http://dx.doi.org/10.1006/jmbi.1999.3411, URL http://www.sciencedirect.com/science/article/pii/S0022283699934113.
[85] D. Huang, A. Caflisch, PLoS Comput. Biol. 7 (2) (2011) e1002002, http://dx.doi.org/10.1371/journal.pcbi.1002002.
[86] D. Huang, A. Caflisch, ChemMedChem 6 (9) (2011) 1578–1580, http://dx.doi.org/10.1002/cmdc.201100237.
[87] M. Xu, A. Caflisch, P. Hamm, J. Chem. Theory Comput. 12 (3) (2016) 1393–1399, http://dx.doi.org/10.1021/acs.jctc.5b01052.
[88] G. Smith, A.L. Bertone, C. Kaeding, E.J. Simmons, S. Apostoles, Am. J. Vet. Res. 59 (9) (1998) 1149–1152, URL https://www.ncbi.nlm.nih.gov/pubmed/9736394.
[89] Z. Malik, G. Kostenich, L. Roitman, B. Ehrenberg, A. Orenstein, J. Photochem. Photobiol., B 28 (3) (1995) 213–218, http://dx.doi.org/10.1016/1011-1344(95)07117-K.
[90] A. Onufriev, D. Bashford, D.A. Case, Proteins: Structure, Function, and Bioinformatics 55 (2) (2004) 383–394, http://dx.doi.org/10.1002/prot.20033.
[91] J. Wang, R.M. Wolf, J.W. Caldwell, P.A. Kollman, D.A. Case, J. Comput. Chem. 25 (9) (2004) 1157–1174, http://dx.doi.org/10.1002/jcc.20035.
[92] J.A. Maier, C. Martinez, K. Kasavajhala, L. Wickstrom, K.E. Hauser, C. Simmerling, J. Chem. Theory Comput. 11 (8) (2015) 3696–3713, http://dx.doi.org/10.1021/acs.jctc.5b00255.