Proposing new configurations in Monte Carlo simulations using truncated Markov Chains

Joël Mabillard, Isha Malhotra, and Bortolo Matteo Mogネット

Center for Nonlinear Phenomena and Complex Systems, Code Postal 231, Université Libre de Bruxelles, Boulevard du Triomphe, 1050 Brussels, Belgium

We propose a new Monte Carlo method in which trial configurations are generated by iterating a finite number of times an initial guess, sampled from a prior distribution, using Markov chains. Like in path-sampling Monte Carlo, we define an extended configurational space comprising all possible trajectories obtained by iterating the prior distribution by a given number of Markov updates. By enforcing detailed balance conditions between trajectories in the extended configurational space, we validate two prototypical schemes that sample desired target distributions while generating new configurations through truncated Markov Chains. As compared to existing path-sampling methods, trajectories are generated using non-local moves based on Bayesian sampling. Such a complication guarantees reasonable acceptances in the setting considered by this work. Using representative systems, we identify the important factors controlling the quality of the sampling of this type of algorithms. Considering the problem of sampling polymers with fixed endpoints, truncated Markov Chains can outperform an existing algorithm in over-stretched conditions. Applications of the proposed methodology range from the design of new generative models to the improvement of the portability of Monte Carlo algorithms, like Configurational Bias schemes, relying on exact sampling of a subset of terms entering the Hamiltonian while generating new configurations.

I. INTRODUCTION

Markov Chain Monte Carlo (MCMC) methods are portable algorithms universally employed to sample probability functions in high-dimensional spaces \cite{13}. In MCMC methods, Markov Chains (MCs) are used to generate new configurations, \( y \), from the current state of the system, \( x \), using transition matrices \( P(x \to y) \) that usually read as follows (for continuum configurational spaces)

\[
P(x \to y) = P_{\text{gen}}(x \to y) \cdot \text{acc}(P)(x \to y) + \delta(x - y)
\]

where \( P_{\text{gen}} \) is used to generate the proposed new configuration, \( y \), and \( \text{acc}(P) \) is the probability of accepting it. For a given \( P_{\text{gen}} \), the corresponding acceptance is usually chosen to fulfill the detailed balance condition

\[
P_T(x) \cdot P(x \to y) = P_T(y) \cdot P(y \to x),
\]

where \( P_T \) is the probability density function of the target distribution to be sampled. Except for studies breaking the microscopic reversibility condition, Eq. (2), to increase the diffusivity of the chain in the configurational space \cite{4, 8}, many developments have focused on designing suitable \( P_{\text{gen}} \) leading to a high acceptance and a fast decorrelation between the configurations visited by the MC \cite{1, 2}, for instance, through the development of cluster moves, e.g. \cite{7, 8}.

Specific Monte Carlo methods, such as the Configurational Bias MCMC \cite{9, 11}, require performing exact sampling of a subset of degrees of freedom on the fly, e.g., by generating polymer segments following given torsional and bending potentials. This pre-sampling task is usually addressed using ad hoc, system-dependent static methods \cite{12, 14}. In this work, we introduce a generative method that leverages the portability of dynamic Monte Carlo in sampling arbitrary target distributions. We generate trial configurations (in the following \( y_n \)) by evolving states \( y_0 \) sampled from a prior distribution, with a probability density function \( P_0 \), using a second Markov Chain, with transition matrix \( M(x \to y) \), truncated after \( n \) iterations. In particular, the probability of generating a particular trajectory \( y_{0,n} \equiv (y_0, y_1, \ldots, y_n) \) reads as \( P_{\text{gen}}(y_{0,n}) = P_0(y_0) P_{\text{gen}}(y_{0,n}|y_0) \) with

\[
P_{\text{gen}}(y_{0,n}|y_0) = \prod_{i=0}^{n-1} M(y_i \to y_{i+1}).
\]

As compared to hybrid MCMC algorithms \cite{17, 19} based on symplectic integrators \cite{20, 21}, using truncated Markov Chains to propose new configurations is tricky as Markov transfer matrices compress volumes in configurational space. The work of Crooks \cite{22}, based on Jarzynski’s results \cite{23, 24}, shows how such configurational compression could be expressed in terms of the heat \( Q \) exchanged with the thermostat coupled to \( M \) during the generation of \( y_{0,n} \)

\[
\frac{P_{\text{gen}}(y_{0,n}|y_0)}{P_{\text{gen}}(y_{0,n}|y_n)} = e^{-\beta Q},
\]

where \( y_{n,0} \) is the reversed path \( y_{n,0} = (y_n, y_{n-1}, \ldots, y_0) \). In the setting considered here, in which \( y_0 \) is evolved by \( M \) from \( P_0 \) towards the target.
probability density $P_T$, the algorithm of Crooks is not directly applicable. Indeed, following Eq. (3) new configurations are generated from scratch starting from $y_0$ stochastically sampled from $P_0$. The configurational space is compressed along the old path $x_{0,n}$ and $y_{0,n}$ cannot be correlated (as in Eq. [3]), unless in specific cases. It follows that it is not possible to build a general acceptance rule based on $P_{gen}(y_{0,n})$, $P_{gen}(x_{0,n})$, $P_T(y_n)$, and $P_T(x_n)$.

In this work, we address this problem and develop MC methods that generate representative trial configurations using truncated MCs (tMCs) evolved from a prior distribution with probability density function $P_0$. Acceptance rates are calculated from detailed balance conditions between pairs of trajectories as done in path sampling MC [26,33]. In Sec. II B and Sec. II D we present and validate two possible implementations (Algorithm A and B). In Sec. II E we compare our methodology with existing path sampling schemes [25–33]. In Sec. II F we highlight the potentialities of the method by sampling polymers with fixed endpoints. Instead, in Sec. III we identify the limits of the methodology but also discuss some directions for improvement. Finally, in Sec. IV we summarise our results.

II. PRESENTATION OF THE ALGORITHMS

A. Generating new configurations

We first detail how we generate trajectories $y_{0,n}$, $y_{0,n} = (y_0, y_1, \cdots, y_n)$, from which we propose the new configuration $y = y_n$ (Fig. 1 left). The starting configuration $y_0$ is extracted from a prior distribution using the probability density function $P_0$ which we assume is simple enough to be sampled stochastically. The configuration $y_i (i \geq 1)$ is selected from $y_{i−1}$ with probability $M(y_{i−1} \to y_i)$ satisfying the detailed balance condition

$$P_T^0(y_{i−1})M(y_{i−1} \to y_i) = P_T^0(y_i)M(y_i \to y_{i−1}). \quad (5)$$

$P_T^0$ and $P_T$, the latter being the probability density function of the distribution to be sampled, Eq. (4), are treated as independent functions, although in most of the examples of this work we set $P_T^0(x) = P_T(x)$. As in Eq. (1), for a given $y_{i−1}$ we sample $M(y_{i−1} \to y_i)$ by first proposing a new configuration, $K_{i−1}$, using a transformation (e.g. a local translation) $K_i$, and then accepting it with a rate $\text{acc}^{(M)}(y_{i−1} \to y_i)\text{K}_{i−1}$ that enforces the detailed balance condition, Eq. (5). The transformation $K_i$ conserves volumes in the configurational space and is sampled from a generic distribution $\mu(K_i)$. We do not put any restriction on $\mu(K)$ to the extent that the ergodicity of $P$ (see Eq. (1)) is guaranteed. In particular, the condition $\mu(K) = \mu(K^{-1})$ is not required. Given the $n$ proposed transformations, $K = (K_1, K_2, \cdots, K_n)$ and the starting configuration $y_0$, $y_i$ is calculated as follows

$$y_i = \prod_{j=1}^{i} (K_j)^{n_j} y_0 \quad (6)$$

where $n_j = 0$ or 1, respectively, if the transformation $K_j$ is rejected or accepted, according to $\text{acc}^{(M)}(y_{j−1} \to K_j y_{j−1})$.

There are $2^n$ (possibly degenerate) final configurations $y_n$ identified by the set of acceptances $\eta = (\eta_1, \eta_2, \cdots, \eta_n)$. The probability of generating the proposed new configuration $y = y_n$ for a given $K$ is then

$$P_{gen}(y_{0,n} | K) = P_0(y_0) \prod_{i=1}^{n} f_{K_i, \eta_i}(y_{i−1} \to y_i) \quad (7)$$

with

$$f_{K_i, \eta_i}(y_{i−1} \to y_i) = \begin{cases} \text{acc}^{(M)}(y_{i−1} \to y_i) & \text{if } \eta_i = 1 \\ 1 - \text{acc}^{(M)}(y_{i−1} \to y_i) & \text{if } \eta_i = 0 \end{cases} \quad (8)$$

where $y_i$ is calculated using Eq. (6). In practice, $P_{gen}(y_{0,n} | K)$ is computed directly while generating $y_{0,n}$, as described in Appendix A. The probability $P_{gen}(y_{0,n} | K)$ cannot be directly identified with $P_{gen}(x \to y)$ in Eqs. (1) and (2) as the latter quantity includes the contributions of all trajectories terminating in $y$ as obtained by all possible choices of $K$. As done in path sampling MC [26,33], we will introduce two partition functions ($Z^{(A)}$ and $Z^{(B)}$) assigning a statistical weight to each trajectory $(y_0, y_1, \cdots, y_n)$ defined on an extended configurational space. Probability distributions defined over the extended configurational space, along with $P_{gen}(y_{0,n} | K)$, allow writing detailed-balance conditions between trajectories. Note that new trajectories are generated from scratch (i.e., $P_{gen}(y_{0,n} | K)$ does not depend on the current path $x$). This property is pivotal in applications as the one studied in Sec. III. Moreover, in Sec. III E we sketch out an alternative method in which new paths are constructed starting from the existing one. The latter method will allow studying systems in which $P_0$ cannot be sampled stochastically.

B. Algorithm A

We define the following partition function defined over the ensemble of all possible trajectories $x_{0,n} = (x_0, x_1, \cdots, x_n)$

$$Z^{(A)} = \int dx_0 \prod_{i=1}^{n} \sum_{\eta_i \in \{0,1\}} d\mu(K_i) P_T(x_n) \quad (9)$$

with $x_n = \prod_{i=1}^{n} (K_i)^{\eta_i} x_0$. Given the set of transformations $K$, $x_i$ (with $i > 0$) is uniquely determined by $x_0$ and the set of acceptances $\eta$. Since it is convenient to treat the physical variable $x_n$ as an independent variable, we will identify a configuration in the extended space with
FIG. 1. (a) Schematic representation of the method for a two-dimensional model (see Appendix B 2 and [34] for details of the model and simulation parameters). Dashed and solid lines represent the level lines, respectively, of the prior, $P_0$, and target probability density functions, $P_T$. A proposed path $y_{0:n}$ is generated by accepting or rejecting a series of local displacements $K_i$ (in the panel the second displacement, $K_2$, is rejected) starting from $y_0$ distributed as $P_0$. Similarly, for a given state $x \equiv x_n$, the method reconstructs an extended configuration by accepting or rejecting the series of reverse transitions $(K_{n-1}^1, K_{n-2}^1, \ldots, K_0^1)$. In the panel, Algorithm A accepts $K_{n-1}^1$ and $K_{n-2}^1$, while Algorithm B accepts $K_{n-1}^1$ and $K_{n-2}^1$. (b) Full lines are marginal distributions of the target distribution of panel a. Algorithm A and B properly sample $P_T$ (symbols). Dotted and dashed lines are the distributions of $y_n$ with $n = 10$ and $n = 4$, respectively. (c) Generation of the new path $y_{0:n}$ (left). Reconstruction of the trajectory ending in the old state $x$ with Algorithm A (center) and Algorithm B (right). $\eta_i = 1$ ($\eta_i = 0$) if the update $K_i$ or $K_i^{-1}$ is accepted (rejected) when generating the new or old configuration, respectively. When generating $y_n$ (left) $\eta_i$ is calculated using acceptance rates. When reconstructing the old state (center and right), Algorithm A and B calculate $\eta$ differently (see text). In panels a and c $n = 4$, while in panels a and b the scale bar represents the unit length.

$x_n$ and $\eta_i$, and calculate $x_0$ by inverting Eq. (6). The Jacobian of the change of variables $\{x_0, \eta\} \rightarrow \{x_n, \eta_i\}$ is equal to 1 if $K$ conserves volumes in configurational space. This change of variables leads to the following expression for $Z^{(A)}$

$$Z^{(A)} = \int dx_n \left[ \prod_{i=1}^n \sum_{\eta_i \in (0,1)} d\mu(K_i) \right] P_T(x_n), \quad (10)$$

From the partition function, we observe that the marginal distribution of the physical variable $x_n$ is given by $P_T$. Therefore sampling trajectories according to $Z^{(A)}$ provides configurations $x_n$ distributed as $P_T(x_n)$. Moreover, for a given $x_n$ and $K$, all $\eta_i$'s are uniformly distributed with $\text{Prob}(\eta_i = 0) = \text{Prob}(\eta_i = 1) = 1/2$. This

is an important simplification as compared to other path-sampling methods [26, 27]. In our method, given the actual configuration $x = x_n$, we can sample an ensemble of transformations $K$ and reconstruct a trajectory $x_{0:n}$ distributed as in $Z^{(A)}$ (conditionally to $x_n$), as explained below. Among other advantages, this enhances sampling of $Z^{(A)}$ given that paths are generated from scratch. Given the old configuration $x$ in the physical space, we identify the final state of the corresponding path $x_{0:n}$, $x_n$, with $x$ and sample $n$ uniformly distributed $\eta_i$. We reconstruct $x_i$, for $i = n - 1$ to 0, by inverting Eq. (6),
using iteratively  \( x_i = (K_{i+1})^{-\eta_i} x_{i+1} \) such that

\[
x_i = \left[ \prod_{j=i+1}^{n} K_j^{-\eta_j} \right] x_n.
\]

The process is illustrated in Fig. 1; center. We stress that in Eq. (6), the \( \eta_i \)'s are determined with an acceptance test whereas in Eq. (11) they are sampled with a uniform distribution. Having obtained the ensemble \( x_{0:n} \equiv (x_0, \cdots, x_n = x) \), we calculate the generating probability \( P_{\text{gen}}(x_{0:n}|K) \) computed using Eq. (7). \( P_{\text{gen}}(x_{0:n}|K) \) is the probability of generating \( x_{0:n} \) using the tMC described in the previous section. The new configuration is then accepted with probability \( \text{acc} = F(z_A) \), e.g., \( F(z_A) = \min[1, z_A] \) or \( F(z_A) = z_A/(1 + z_A) \) when using, respectively, the Metropolis or the heat-bath acceptance with

\[
z_A = \frac{P_{\text{gen}}(x_{0:n}|K) P_T(y_n)}{P_{\text{gen}}(y_{0:n}|K) P_T(x_n)}.
\]

A summary of the algorithm with more details on the computation of \( P_{\text{gen}}(x_{0:n}|K) \) is given in Appendix A.2. Notice that the possibility of reconstructing \( x_{0:n} \) from \( x_n \), using the set \( K \) used to generate the new configuration \( y_n \), is a key property of the method. Indeed, as required by ergodicity, the new configuration is generated using a set \( K \) which does not coincide with the set of transformations used to originally generate the old configuration.

### C. Two-dimensional Model

To illustrate the method and verify the validity of the algorithm, we consider the two-dimensional system of Fig. 1a. The prior distribution, with probability density function \( P_0 \) from which \( y_0 \) is sampled, is a two-dimensional Gaussian distribution. The probability density function \( P_T \) related to the target distribution is multimodal as depicted in Fig. 1b. The analytic expressions of \( P_0 \) and \( P_T \) are reported in Appendix B.2. Importantly, we challenge the algorithm by choosing a prior probability density function \( P_0 \) not overlapping with \( P_T \). Each transformation \( K_i \) attempts to displace the two dimensional state within a square with size equal to 2. The results of Fig. 1b have been obtained with \( 2 \cdot 10^6 \) iterations. This figure shows that Algorithm A properly samples the target probability density function \( P_T \).

Despite the fact that when \( P_T(x) = P_T^{(M)}(x) \), \( y_n \) samples \( P_T \) for \( n \to \infty \), arbitrarily big values of \( n \) do not improve the quality of the sampling. In particular, the acceptance, for a given \( \mu(K) \), is non-monotonous in \( n \), as shown in Fig. 2 2D Algorithm A. The poor performance of Algorithm A at large values of \( n \) is because the ensembles of replicas \( x_{0:n} \) are random walks that do not tempt to descend the landscape provided by \( P_T^{(M)} \). For the same reason, \( x_0 \) may land outside of the range of typical configurations distributed as \( P_0 \), as shown with the position of \( x_0^A \) in Fig. 1a. It is important to stress that \( x_{0:n} \) and \( y_{0:n} \) are distributed differently. In particular, \( y_{0:n} \) is determined by the growing procedure with \( P_0 \) and \( M \) in Eq. (9), while \( x_{0:n} \) by the extended partition functions \( Z^{(A)} \) in Eq. (10). These considerations will in general result in small values of \( P_{\text{gen}}(x_{0:n}|K) \), leading to small \( z_A \) as defined in Eq. (12) and, therefore, small acceptances. This analysis is supported by Fig. 3 “A old” and “new”, comparing the distributions of the averaged probabilities of generating new and old configurations. In particular, for the new configurations (“new”) we have

\[
\hat{P}_{\text{gen}} = \int d\mu(K) d\eta_0 P_0(\eta_0) P_{\text{gen}}(y_{0:n}|K)
\]

with the acceptances \( \eta \) calculated as in Eq. (7). Instead, for the old configurations we have (“old A”)

\[
\bar{P}_{\text{gen}} = \int d\mu(K) dx_n P_T(x_n) P_{\text{gen}}(x_{0:n}|K)
\]

where the \( \eta \) has been sampled uniformly as described above and in Appendix A.2. We notice that rejected states could be used to improve the quality of the sampling.

### D. Algorithm B

To alleviate the problem of low values of \( P_{\text{gen}}(x_{0:n}|K) \) as compared to \( P_{\text{gen}}(y_{0:n}|K) \), we modify the extended partition function in Eq. (10). Following the conclusions
of the previous section, we aim at increasing the overlap between the distribution of \( x_{0,n} \) and \( y_{0,n} \). Specifically, we constrain the distribution of an equilibrium configuration \( x_{0,n} \) for a given \( n \) and \( K \) to \( P_{\text{gen}}(x_{0,n}|K) \) given by Eq. (7). This observation leads to the following partition function in the extended space

\[
Z^{(B)} = \int dx \prod_{i=1}^{n} \sum_{\eta_i \in \{0,1\}} \left[ \text{det}(K) f_{K,\eta_i}(x_{i-1} \rightarrow x_i) \right] P_0(x_0) \prod_{i}(K,x_i) P_T(x_n),
\]

where each trajectory \( x_{0,n} \) ending in \( x_n = x \) is weighted by its generating probability \( P_{\text{gen}}(x_{0,n}|K) \). \( J \) is a bias term that constrains the marginal distribution of the physical variable \( x_n \) to be equal to \( P_T(x_n) \). This observation implies that the distribution of \( x_{0,n} \) and \( y_{0,n} \) are still not overlapping, given that \( y_n \) is not distributed as \( P_T^{(M)} \), as a consequence of truncating the Markov Chain \( M \) after \( n \) iterations. The same conclusion follows from Fig. 3 when comparing “new” with “B old”.

The expression of \( J \) reads as follows

\[
J(K,x_n) = \left[ \sum_{\eta \in \{0,1\}^n} P_{\text{gen}}(x_{0,n}|K) \delta_{z_{0,n},x_n} \right]^{-1}
\]

where we sum over all \( 2^n \) paths (identified by a given set of acceptances \( \eta \)) that for a given set of \( K \) end in \( x_n \). In particular, the trajectory \( x_{0,n} = (z_0, \cdots, z_n = x_n) \) is constructed inverting Eq. (6), \( z_{i-1} = K^{-1}_{i-1} z_i \). A comparison with existing literature [28,22] employing path-sampling partition functions similar to Eq. (15) will be presented in Sec. IIE.

As done in Algorithm A, for a given \( x_n \) and \( K \), we need to reconstruct the path \( x_{0,n} \) in the extended space by sampling Eq. (16). For a given \( K \), we consider the tree made of the \( 2^n \) possible trajectories leading to \( x_n \) as shown in Fig. 1, right. We then select a trajectory \( x_{0,n} \) by sampling \( P_{\text{gen}}(x_{0,n}|K) \) using Bayes’ theorem. In practice, given \( x_i \), we choose \( x_{i-1} \) (and therefore \( \eta_i \)) among \( x_i \) and \( K^{-1}_{i-1} z_i \) with probability \( (\eta_i = 0, 1) \)

\[
P(K^{-1}_{i-1} z_i | x_i) = \frac{f_{K,\eta_i}(K^{-1}_{i-1} z_i \rightarrow x_i) P_{\text{gen}}(x_{i-1}|K_{i-1}^{-1} z_i)}{P_1(x_i)}
\]

(17)

where \( P_1(x_i) \) is calculated recursively

\[
P_1(x_i) = f_{K,\delta}(x_i \rightarrow x_i) P_{1-1}(x_i) + f_{K,1}(K^{-1}_{i-1} z_i \rightarrow x_i) P_{1-1}(K^{-1}_{i-1} z_i)
\]

(18)

where \( f_{K,\eta_i} \) have been defined in Eq. (8). \( P_1(x_i) \) is the probability to visit the state \( x_i \) when sampling layer \( i \) at a given \( (\eta_{i+1}, \eta_{i+2}, \cdots, \eta_n) \), \( K \) and \( x_n \) [30]. In particular, we have that \( P_n(x_n) = J(K,x_n)^{-1} \). We report details on the calculation of \( P_1 \) and the \( \eta_i \)’s in Appendix A.3. Importantly, the calculation of \( P_1 \) limits the algorithm to small values of \( n \) given the necessity of enumerating \( 2^n \) states.

After sampling the \( \eta_i \)’s with the iterative use of Eq. (17), it can be shown that the probability of selecting \( x_{0,n} \) is still given by Eq. (7). The new configuration is then accepted with probability \( \text{acc}(P) = F(z_B) \)

\[
z_B = J(K,y_n) P_T(y_n) = P_n(x_n) P_T(y_n) / P_B(y_n) P_T(x_n)
\]

(19)

Interestingly, \( z_B \) is not a function of \( P_{\text{gen}}(x_{0,n}|K) \) and \( P_{\text{gen}}(y_{0,n}|K) \) as they cancel the corresponding terms appearing in the asymptotic distribution of the extended space in Eq. (15).

In Fig. 1 we verify that Algorithm B is not biased in reproducing the target distribution with probability density function \( P_T \). Moreover, Fig. 2 “2D Algorithm B,” shows how acceptance now increases with \( n \) due to higher values of \( P_{\text{gen}} \), as can be seen by comparing A and B in Fig. 3. In this figure, \( P_{\text{gen}} \) for Algorithm B is calculated using Eq. (14) with the acceptances \( \eta_i \)’s obtained from Eq. (17). A summary of the algorithm and more details on the computation of \( P_1 \) are given in Appendix A.3.

As already observed, Algorithm A and B are not peculiar to the use of transition matrices \( M \) having as asymptotic state \( P_T \), as for the two-dimensional example discussed above. This property is crucial in cases where the evaluation of \( P_T \) is computationally expensive, e.g. when \( P_T \) is a multibody function, and could be approximated by a less complex function [11]. As a proof of principle, in Appendix C.1 we describe and validate using the 2D system of Fig. 1 the case in which \( P_T^{(M)} \) is constant.

E. Comparison with other path-sampling methods

Path-sampling Monte Carlo has been used to calculate free energies [28,31]. Based on Jarzynski’s results [23]
Notice that Fig. 4, instead, using a notation similar to Eqs. (22) and (25), the difference in free energy between two systems (21), the difference in free energy between two systems with Hamiltonian \( \mathcal{H}_0 \) (left and center) and in the current study (right). \( x_{0,n} \) and \( x_{n,0} \) represent trajectories generated using, respectively, the forward and backward protocol (22). (b) (left) Generation of a new path using back-shooting from an updated \( x_i(y_i) \). \( y_{n,0} \) and \( y_{0,n} \) are paths obtained by iterating, respectively \( n-i \) and \( i \) times the forward and backward protocol. (right) Proposed method which does not rely on back-shooting to generate new paths.

As explained in Sec. [ID], the motivation for choosing an extended partition function as in Eqs. (15) and (23) is to maximize overlapping between the distributions of generated and equilibrium trajectories \( (y_{0,n} \) and \( x_{0,n} \), using the notation of Sec. [ID]). In this respect, a key difference between our algorithms and existing path-sampling MC protocols is that we never propose new configurations by an inverse-protocol (back-shooting). A typical move to propose new configurations using back–shooting is shown in Fig. 4, left [26, 27]. Here, new trajectories are generated by a local update of one of the configurations belonging to the current path \( x_i \rightarrow y_i \), followed by propagating \( y_i \) to \( y_n \) using the forward map and \( y_i \) to \( y_0 \) using the backward dynamics (back-shooting). In our general setting, in which we instantaneously drive the system from \( \mathcal{H}_0 \) to \( \mathcal{H}_T \), back-shooting is very inefficient in sampling \( Z_B \). Indeed, considering the case in which \( P_T \) is not overlapping with \( P_0 \) (Fig. 4, left), back-shooting updates would simply relax \( y_i \), as in the forward move, towards the typical values of \( P_T \). It follows that the distribution of \( y_0 \) will resemble more \( P_T \) than \( P_0 \). For the system of Fig. 4, back-shooting will be outperformed even by a random reconstruction of the path (as in algorithm A, Sec. [IB]) as in the latter case \( y_0 \) is generated from a random walk which is not constrained by \( P_T \).

It is worth pointing out that our method does not necessarily require the ability to sample the prior \( P_0 \) statically (i.e. generating new configurations distributed as \( P_0 \) from scratch). A possible algorithm generating new paths \( y_{0,n} \) from the current configuration \( x_{0,n} \) is illustrated in Fig. 4, right. Starting from a physical configuration \( x_n \), we reconstruct the most likely path \( x_{0,n} \) (as done in Sec. [ID]). We then propose a new path \( y_{0,n} \) (and therefore a new configuration \( y_n \)) by a forward propagation of \( x_0 \) identified with \( y_0 \). This algorithm generates more correlated configurations (as compared to the method presented in Sec. [ID]) but with a higher acceptance given that, in this case, \( y_0 \) is asymptotically distributed as in the extended partition function. It should also be stressed that the dynamic scheme of Fig. 4, right, cannot be used in applications where tMCs are used to pre–sample a subset of the degrees of freedom of the system. This is illustrated in the example presented in the next section.

$$\int dx_1 \cdots dx_{n-1} P_{\text{gen}}(x_{0,n}|x_n) \neq 1. \quad (26)$$
posited \(j - 1\) monomer \((r_{\text{new}}^{\text{new}} \equiv r_{0} \text{ if } j = 1)\). A truncated Markov Chain \(M^{(j)}\), attempting to displace a monomer within a cube of size 4, is then used to evolve \(y_{0}^{(j)}\) to \(y_{1}^{(j)}\). The asymptotic state visited by \(M^{(j)}\), \(P_{T}^{(M^{(j)})}\), is taken equal to \(P_{T}^{(M^{(j)})}(y^{(j)}) \sim P_{G}(y_{0}^{(j)}, r_{NT})P_{0}(y_{0}^{(j)})\), where \(P_{G}\) is a guiding function biasing the chain’s growth towards the fixed end monomer \(\mathbf{1} [14]\). \(P_{T}\) is chosen as the end-to-end distance distribution of a chain segment of length \(NT - j\) with unconstrained end-to-end distance.

FIG. 5. (a) Schematic of the truncated Markov Chain strategy employed to sample polymers with fixed endpoints. (b) Comparison of the acceptance rates obtained by Configurational Bias MC (with \(n = 1000\) trials) and Algorithm B (with \(n = 10\)) for a chain with 6 and 11 monomers (\(NT = 5\) and 11) as a function of the end-to-end distance \(d\). Inset: Asymptotic distributions of the \(i^\text{th}\) monomer along the stretching direction \(r_{x,i}\) (with \(i = 1, 5, 9,\) and \(NT = 10\)) sampled by Algorithm B compared with the expected distribution (solid lines).

III. SAMPLING POLYMERS WITH FIXED ENDPOINTS

In this section, we develop a method to generate chains with fixed endpoints based on Algorithm B introduced in Sec. [14]. This problem underlies efficient sampling of configurational volumes in polymer systems and is usually addressed using Configurational Biased Monte Carlo (CBMC) simulations [14]. We consider a chain with \(NT + 1\) monomers located at positions \(r_{i}, i = 0, \ldots, NT\) and endpoints fixed at a distance equal to \(d\), such that \(|r_{NT} - r_{0}| = d\). Neighboring monomers interact via a harmonic potential \(V_{0}\). The configurational energy of the system is given by

\[
U_{HS} = \sum_{i=1}^{NT} V_{0}(r_{i}, r_{i-1}).
\]

More details on the system are provided in Appendix B. As in CBMC, new configurations are generated one monomer at a time. We use \(NT - 1\) different truncated Markov Chains, \((M^{(1)}, \ldots, M^{(NT-1)})\), to sequentially generate new monomers, \(r_{\text{new}}^{\text{new}} \equiv y_{i}^{(i)}\) with \(i = 1, \ldots, NT - 1\). A proposal of monomer \(j\) at position \(y_{n}^{(j)}\), see Fig. 3a, is selected by first sampling \(y_{0}^{(j)}\) distributed as

\[
P_{0}(y_{0}^{(j)}) = \exp(-V_{0}(y_{0}^{(j)}, r_{\text{new}}^{\text{new}}_{j-1})),
\]

where \(r_{\text{new}}^{\text{new}}_{j-1}\) is the proposed \(j - 1\) monomer. As in CBMC, new configurations are generated one at a time, and the computational complexity of the CBMC with \(k = 1000\) is comparable to that of Algorithm B with a tMC truncated after \(n = 10\) iterations, as employed Fig. 5b. We stress that a more efficient CBMC
algorithm would generate trial segments distributed as $cP_T P_0$, where $c$ is a normalization constant. However, sampling $P_T P_0$ would require developing system-specific sampling procedures [14]. On the other hand, the truncated Markov Chain method can readily be employed for any type of potentials (including bending and torsional terms). In that sense, truncated Markov Chain algorithms are more portable.

In that sense, truncated Markov Chain algorithms are more portable.

![Image](https://via.placeholder.com/150)

**FIG. 6.** Simulated (symbols) and expected distribution (full line) of the dihedral angle $(\omega_{12})$ of molecules with 3 branches (2-methylpropane, panel a) and 4 branches (2,2-dimethylpropane, panel b). Dotted and dashed lines are the distributions of $\eta_n$ with $n = 10$ and $n = 4$, respectively. The legend is as in Fig. 1. Details of the models and simulation parameters are reported in Sec. B 4 and in [34].

**IV. LIMITATIONS OF THE CURRENT ALGORITHM**

In this section we highlight how the proposed method may be limited by a poor overlapping between the distribution of $\eta_n$ and the target probability density function $P_T$. That is the case in systems in which a single Markov Chain is used to propose multiple degrees of freedom in one time (notice that in Sec. II different monomers are proposed using different $M$). We consider branched molecules which are test systems in Configurational Bias MCMC developments [12–16]. We consider 2-methylpropane (BM2) and 2,2-dimethylpropane (BM3) molecules constituted, respectively, by three and four branches as shown in Fig. 6. The length of the branches is fixed while different branches interact through stiff bending terms $U_{bend}$. More details on the model are given in Appendix B 4. We fix the branch 0 and label the branches that are sampled from 1 to $n_b$, where $n_b = 2$ for BM2 or 3 for BM3. The target probability density function is given as

$$P_T = c \prod_{i<j} \exp[-\beta U_{bend}(\theta_{ij})],$$

where $\beta = 1/(k_B T)$, $k_B$ is the Boltzmann constant, $T$ the temperature, $\theta_{ij}$ is the angle between the branches $i$ and $j$, and $c$ a normalization constant. We choose the following prior probability density function

$$P_0 = c_0 \prod_{i>0} \exp[-\beta U_{bend}(\theta_{0i})],$$

where $c_0$ is the normalization factor. The trial moves, $K_i$, employed by the Markov Chain $M$ act as follows. One of the $n_b$ dynamic branches is chosen with uniform probability and rotated by a random angle chosen within $-\pi/2$ and $\pi/2$ around a random unit vector centered at the center of the molecule. For each point of Fig. 6 we performed $10^6$ iterations of the algorithm [34].

In Fig. 6 we test the algorithm by sampling the dihedral angle, $\omega_{ij}$, defined as the angle between the plane spanned by the branches 0 and $i$ and the plane spanned by the branches 0 and $j$. As expected, the proposed methods can reproduce the target distribution. However, the acceptance rates could be quite small, especially in the case of the 4-branches molecule, even when using the expensive Algorithm B, as reported in Fig. 2. Fig. 6 shows how the distributions of $\eta_n$ for the highest value of $n$ considered are still far from $P_T$. This observation unveils a possible limitation of the method: In high dimensions, larger values of $n$ are required to generate acceptable configurations. In favor of the proposed method, one should notice that most of the proposed configurations were rejected due to a completely agnostic selection of the proposed updates, $K_i$. Instead, as in hybrid MCMC, one could envisage using symplectic transformations to generate updates driving far from equilibrium states into the basins of $P_T$ in few updates. Investigations in this direction will be addressed in future efforts.

**V. CONCLUSIONS**

As compared to static methods, such as Von Neumann rejection schemes, dynamic Monte Carlo algorithms sample target distributions only asymptotically. On the other hand, dynamic schemes are more portable, they do not require knowing features of $P_T$ which are necessary to implement an efficient rejection scheme, and suitable to sample high-dimensional spaces. In this paper, we have shown how truncated Markov chains could be used to propose new configurations in Monte Carlo simulations. As done in path sampling MC, we have defined an extended space comprising all the trajectories generated by $n$ iterations of a Markov transformation and derived two prototypical algorithms (named A and B). As compared to existing path sampling methods, we sample trial states starting from a prior distribution. As proven by the results presented in Sec. III this setting is particularly useful to methods requiring static sampling of a subset of degrees of freedom on the fly [37]. In the short term, the proposed technique will support MC algorithms capable of coarsening degrees of freedom on
ACKNOWLEDGEMENTS

We thank an anonymous Referee for insightful suggestions and for bringing to our attention relevant literature about path sampling MC. We thank Alessandro Bevilacqua for early discussions on the sampling of stretched harmonic chains which motivated the developments of the proposed methodology. This paper is dedicated to his memory. Financial support was provided by the Université Libre de Bruxelles (ULB) and an A.R.C grant of the Fédération Wallonie-Bruxelles. Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CECI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11.

Appendix A: Numerical recipes

We provide the implementation of the algorithms introduced in this work. In particular, we detail the probability of generating an old trajectory, \( P_{\text{gen}}(x_{0,n}|K) \) in Eq. (7), and the total probability of generating a configuration \( x_n \), \( P_n(x_n) \) in Eq. (15). While calculating \( P_n(x_n) \) we also sample the acceptances \( \eta_i \)'s defining the trajectory \( x_{0,n} \) in the extended space, ending in the old configuration, \( x \). These are used to calculate \( P_{\text{gen}} \), “B old”, in Fig. 9.

1. Generate \( y \)

Flow chart to generate the new path \( y_{0,n} = (y_0, y_1, \ldots, y_n) \) and compute \( P_{\text{gen}}(y_{0,n}|K) \):

(i) Sample \( y_0 \) from the probability density function \( P_0 \) of the prior distribution:

(ii) Generate the set of transformations \( K = (K_1, \ldots, K_n) \):

(iii) Initialize \( P_{\text{gen}}(y_{0,n}|K), P_{\text{gen}}(y_{0,n}|K) = P_0(y_0); \)

(iv) Iterate from \( i = 1 \) to \( n \):

- Generate \( y_i^{\text{trial}} = K_i(y_{i-1}); \)
- Calculate the acceptance rate for \( y_i^{\text{trial}} \) using \( P_T^{(M)} \);
- If accepted, \( y_i = y_i^{\text{trial}} \), \( P_{\text{gen}} \rightarrow P_{\text{gen}} \cdot \text{acc}^{(M)}(y_{i-1} \rightarrow y_i^{\text{trial}}); \)
- Else, \( y_i = y_{i-1}, \ P_{\text{gen}}(y_{0,n}|K) \rightarrow P_{\text{gen}}(y_{0,n}|K) \cdot [1 - \text{acc}^{(M)}(y_{i-1} \rightarrow y_i^{\text{trial}})]; \)

(v) Return \( y_{0,n}, P_{\text{gen}}(y_{0,n}|K), K \).

2. Algorithm A

Flow chart to calculate the acceptance rate starting from the current configuration \( x \) (in the physical space):

(i) Generate the new configuration \( y \) and the associate path \( y_{0,n} \), along with the set of transformations \( K \) and the generating probability \( P_{\text{gen}}(y_{0,n}|K) \) with the steps described in Sec. A 1;

(ii) Sample \( n \) random acceptances, \( \eta_i \)'s, where \( \eta_i = 0 \) or 1 with equal probability;

(iii) Construct \( x_{0,n} \) by setting \( x_n = x \) and iterating \( x_i = K_i^{-\eta_i+1}(x_{i+1}) \), for \( i = n - 1, \ldots, 0; \)

(iv) Compute the generating probability \( P_{\text{gen}}(x_{0,n}|K) \) using Eq. (7)

\[
P_{\text{gen}}(x_{0,n}|K) = P_0(x_0) \prod_{j=1}^{n} f_{K_j, \eta_j}(x_{j-1}, x_j) \quad (A1)
\]

with \( f_{K_j, \eta_j} \) defined as in Eq. (8);

(v) Calculate the acceptance rate with \( z_A \) given by Eq. (12).

3. Algorithm B

Flow chart to calculate the acceptance rate starting from the current configuration \( x \) (in the physical space):

(i) Generate \( y \) and the set \( K \) with the steps described in Sec. A 1. Note that \( P_{\text{gen}}(y_{0,n}|K) \) is not needed;

(ii) Compute \( P_n(x) \) and \( P_n(y) \) using \( K \) and the method described below;

(iii) Calculate the acceptance rate with \( z^B \) given by Eq. (19).

Flow chart to calculate \( P_n(x_n) \) (or \( P_n(y_n) \)) for a given set of transformations \( K \):

(i) Initialise a \( 2^n \) dimensional vector \( (v_\alpha, \alpha = 0, \ldots, 2^n - 1) \) with the list of states at layer 0, \( x_0^{(0)} \), leading to \( x_n \) with a combination of displacements in \( K \), corresponding to the states flagged with \( x_0^B \) in Fig. 9, right. In particular, if \( b_1 b_2 \cdots b_n \) is the
binary representation of $m = 2^n + \alpha$, we calculate $x_0^{(a)}$ as

$$x_0^{(a)} = (K_1^{-b_1}) \cdots (K_n^{-b_n}) x_n$$  \hspace{1cm} (A2)

(ii) Initialise $P_0$ and $J_0$, where $P_0 = J_0 = P_0(v_0)$, with $P_0$ the prior probability.

(iii) Iterate $P_n$ and $J_n$ n times using the following procedure. At the $i$th iteration, update the components of $P_n$ and $J_n$ with $\alpha = p \cdot 2^i$, with $p = 0, 1, \ldots, 2^n - 1$ using Eqs. (17):

$$J_{p,2i} \left\{ \begin{array}{l} J_{p,2i}^f K_{i,0}(v_{p,2i} \rightarrow v_{p,2i}) \text{ Prob } = 1 - \gamma \\ J_{p,2i}^f K_{i,1}(v_{p,2i} \rightarrow v_{p,2i}) \text{ Prob } = \gamma \\ \end{array} \right.$$  \hspace{1cm} (A3)

where $v_{p,2i} = K_i v_{p,2i+2\cdots-i}$ and

$$\gamma = \frac{P_{p,2i}^f K_{i,0}(v_{p,2i} \rightarrow v_{p,2i})}{V}$$  \hspace{1cm} (A4)

$$V = P_{p,2i}^f K_{i,0}(v_{p,2i} \rightarrow v_{p,2i}) + P_{p,2i}^f K_{i,1}(v_{p,2i} \rightarrow v_{p,2i})$$

(iv) After the $n$th iteration, the value of $P_n(x_n)$ and $P_{gen}(x_{0,n}(k))$ are found as $P_n(x_n) = P_0$ and $P_{gen}(x_{0,n}(k)) = J_0$.

**Appendix B: Prior and target distributions employed**

1. **Notations and conventions**

Throughout this work, we consider a $d$-dimensional Gaussian distribution with a probability density function given by

$$f_N(x; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\}$$  \hspace{1cm} (B1)

where $\mu$ is the $d$-dimensional mean vector and $\Sigma$ the covariance matrix.

2. **Two-dimensional system (Fig. 1)**

The prior distribution is a two-dimensional Gaussian distribution with a probability density function

$$P_0(v) = f_N(v; 0, \Sigma_0),$$  \hspace{1cm} (B2)

with a covariance matrix given by $\Sigma_0 = \text{diag}(\sigma_1^2, \sigma_2^2)$ where $\sigma_1 = \sigma_2 = 0.6$. We use a multimodal target distribution given by the sum of three Gaussian distributions with a probability density function

$$P_T(v) = \frac{1}{2} \sum_{i=1}^{3} f_N(v; \mu_i, \Sigma_i)$$  \hspace{1cm} (B3)

where the inverse of the covariance matrix $\Sigma_i$ is given by

$$\Sigma_i^{-1} = \frac{1}{\sigma_{i,1}^2 \sigma_{i,2}^2 (1 - \rho_i^2)} \begin{pmatrix} \sigma_{i,2}^2 & -\rho_{i,1} \sigma_{i,2} \\ -\rho_{i,2} \sigma_{i,1} & \sigma_{i,1}^2 \end{pmatrix},$$  \hspace{1cm} (B4)

with

$$\mu_1 = (-2, -2), \quad \rho_1 = 0.7,$$

$$\sigma_{1,1} = 0.5, \quad \sigma_{1,2} = 0.5,$$

$$\mu_2 = (-1, -2), \quad \rho_2 = 0,$$

$$\sigma_{2,1} = 0.3, \quad \sigma_{2,2} = 0.6,$$

$$\mu_3 = (-2, 2), \quad \rho_3 = 0.3,$$

$$\sigma_{3,1} = 0.6, \quad \sigma_{3,2} = 0.3.$$

Prior and target distributions are shown on Fig. 1.

3. **The harmonic chain system (Sec. III)**

In the system of Sec. III, neighboring monomers interact via a harmonic potential, $V_0$, which reads as follows

$$V_0(r_i, r_{i-1}) = \frac{1}{2} |r_i - r_{i-1}|^2. \quad \hspace{1cm} (B5)$$

The prior distributions employed to select $y_0^{(j)}$ ($j = 1, \cdots, N_T - 1$) has a probability density function given by

$$P_0(y_0^{(j)}) = f_N(y_0^{(j)}; \rho_{y_{j-1}}^{\text{new}}, 1).$$  \hspace{1cm} (B6)

The asymptotic state visited by $M^{(j)}$, $P_T^{(M^{(j)})}$, is taken equal to

$$P_T^{(M^{(j)})}(y^{(j)}) = \frac{P_G(y^{(j)}, r_{N_T}) P_0(y^{(j)})}{\int d y^{(j)} P_G(y^{(j)}, r_{N_T}) P_0(y^{(j))}}, \quad \hspace{1cm} (B7)$$

with

$$P_G(y_0^{(j)}, r_{N_T}) = f_N(y_0^{(j)}; r_{N_T}, \sqrt{N_T - j} \cdot 1). \quad \hspace{1cm} (B8)$$

4. **Branched molecules (Fig. 6)**

We consider molecules constituted by $n_b + 1$ branches ($n_b = 2$ and 3, see Fig. 6). $P_T$ reads as follows

$$P_T(\theta_0, \cdots, \theta_{n_b}, \theta_{12}, \cdots, \theta_{(n_b-1)n_b}) = \frac{1}{N_T} \exp \left\{ -\beta \sum_{i=0}^{n_b} \sum_{j>i}^{n_b} U_{\text{bend}}(\theta_{ij}) \right\}, \quad \hspace{1cm} (B9)$$

where $\beta = 1/(k_BT)$, $k_B$ is the Boltzmann constant, $T$ the temperature, $\theta_{ij}$ is the angle between the branches $i$ and $j$, and $N_T$ a normalization constant. The bending potential $U_{\text{bend}}(\theta_{ij})$ is defined as

$$U_{\text{bend}}(\theta) = \frac{1}{2} k_B (\theta - \theta_0)^2, \quad \hspace{1cm} (B10)$$
where $k_0$ and $\theta_0$ are parameters of the system (see below). We choose the following probability density function for the prior distribution

$$P_0(\theta_{01}, \ldots, \theta_{0n}) = \frac{1}{N_0} \exp \left\{ -\beta \sum_{i=1}^{n_b} U_{\text{bend}}(\theta_{0i}) \right\},$$

(B11)

where $N_0$ is the normalization factor. In Sec. [IV] we considered 2-methylpropane molecules with parameters

$$n_b = 2,$$  \hspace{1cm}  (B12)
$$T = 300 \text{ K},$$  \hspace{1cm}  (B13)
$$\theta_0 = 112 \text{ (deg)},$$  \hspace{1cm}  (B14)
$$k_0/k_B = 62500 \text{ K},$$  \hspace{1cm}  (B15)

as well as 2,2-dimethylpropane with parameters

$$n_b = 3,$$  \hspace{1cm}  (B16)
$$T = 300 \text{ K},$$  \hspace{1cm}  (B17)
$$\theta_0 = 109.47 \text{ (deg)},$$  \hspace{1cm}  (B18)
$$k_0/k_B = 62500 \text{ K}.$$  \hspace{1cm}  (B19)

The dihedral angle $\omega_{ij}$ (see Fig. 6) is the angle between the plane spanned by the branches 0 and $i$ and the plane spanned by the branches 0 and $j$. This angle is obtained from the bending angles $\theta_{0i}$, $\theta_{0j}$ and $\theta_{ij}$ as

$$\cos(\omega_{ij}) = \frac{\cos(\theta_{ij}) - \cos(\theta_{0i}) \cos(\theta_{0j})}{\sin(\theta_{0i}) \sin(\theta_{0j})}.$$  \hspace{1cm}  (B20)

### Appendix C: Truncated Markov Chain sampling

with $P_T \neq P_T^{(M)}$

We consider a special case of Algorithm A and B where the new configurations, $y_n$, are generated using acceptances $\eta_i$’s uniformly distributed, $\text{Prob}(\eta_i = 1) = \text{Prob}(\eta_i = 0) = 1/2$. In other terms, proposed updates of $y_i$ with $i = 0, \ldots, n - 1$ are accepted with probability

$$1/2 \ (f_{K_{x0}} = f_{K_{x1}} = 1/2).$$

The probability of generating a path leading to $y_n$, $y_{0,n} = (y_0, \ldots, y_n)$ is then

$$P_{\text{gen}}(y_{0,n}|K) = P_0(y_0) \frac{1}{2^n}.$$  \hspace{1cm}  (C1)

In this case, the Markov chain $M$ would asymptotically sample the constant distribution, with probability density function $P_T^{(M)} \sim 1$. Contrary to the choice made in the example of Fig. 2 and 3 (namely $P_T = P_T^{(M)}$), in this section the distribution of the new configurations do not attempt to reproduce the target distribution, with probability density function $P_T$. Below, we consider Algorithms A’ and B’, as special cases of Algorithms A and B respectively, where $P_{\text{gen}}$ is given by Eq. (C1). Using the 2D system of Fig. 6, we prove that the Algorithms A’ and B’ are not biased.

### 1. Algorithm A’

Given $x_n$ and $K$, the probability of generating the old configuration $x_{0,n} = (x_0, \ldots, x_n)$ is given by

$$P_{\text{gen}}(x_{0,n}|K) = P_0(x_0) \frac{1}{2^n},$$  \hspace{1cm}  (C2)

using Eq. (C1) and where

$$x_0 = \prod_{j=1}^{n} K_{xj}^{-\eta_j} x_n.$$  \hspace{1cm}  (C3)

The new configuration is then accepted with probability $F(z_{A'})$ with

$$z_{A'} = \frac{P_{\text{gen}}(x_{0,n}|K)}{P_{\text{gen}}(y_{0,n}|K)} \frac{P_T(y_n)}{P_T(x_n)} = \frac{P_0(x_0)}{P_0(y_0)} \frac{P_T(y_n)}{P_T(x_n)}.$$  \hspace{1cm}  (C4)

using Eqs. (12) and (C2). Fig. 7 shows that Algorithm A’ is not biased. We observe from Fig. 7 that the acceptance rate is reduced compared to Algorithm A for small values of $n$, which is explained by a poorer overlapping between the distribution of the proposed new configurations, dashed and dotted lines in Fig. 7, with $P_T$ as compared to Algorithm A as shown on Fig. 1b. For larger
values of $n$, Algorithm $A'$ outperforms Algorithm A as in the latter case $P_{gen}(x_{0,n}|K)/P_{gen}(y_{0,n}|K)$ is smaller.

2. Algorithm $B'$

If $f_{K_i,\eta_i} = 1/2$, $P_n(y_n)$ becomes as follows

$$P_n(y_n) = \sum_{i=1}^{2^n} P_0(y_{0,i}) \frac{1}{2^n}, \quad \text{(C5)}$$

where $y_{0,i}$ with $i \in (1, 2^n)$ are the $2^n$ states obtained from $y_n$

$$y_{0,i} = \prod_{j=1}^{n} (K_j)^{-\eta_j^i} y_n, \quad \text{(C6)}$$

for a set of acceptance $\eta' = (\eta_1^i, \cdots, \eta_n^i)$. There are $2^n$ of such sets. Similarly for the old configuration $x$ we obtain

$$P_n(x_n) = \frac{\sum_{i=1}^{2^n} P_0(x_{0,i}) \frac{1}{2^n}}{\sum_{i=1}^{2^n} P_0(x_{0,i}) P_T(x_{0,i})} \prod_{j=1}^{n} (K_j)^{-\eta_j^i} x_n. \quad \text{(C7)}$$

The new configuration is accepted with probability $\text{acc}^{(P)} = F(z_B')$ with (see Eq. (19))

$$z_B' = \frac{P_n(x_n) P_T(y_n)}{P_n(y_n) P_T(x_n)} = \frac{\sum_{i=1}^{2^n} P_0(x_{0,i}) P_T(y_{0,i})}{\sum_{i=1}^{2^n} P_0(x_{0,i}) P_T(x_{0,i})}. \quad \text{(C8)}$$

We can verify that the conclusions made for Algorithm $A'$ also apply here. Fig. 7 shows that the Algorithm $B'$ is not biased but, as shown in Fig. 7, has a lower acceptance rate compared to Algorithm B.

The results of Fig. 7 have been obtained with $2 \cdot 10^6$ iterations, except in the $n = 10$ case, Algorithm $B'$ for which we generated $10^8$ new configurations.

[1] D. Frenkel and B. Smit. *Understanding molecular simulation: from algorithms to applications*, volume 1. Elsevier, 2001.
[2] D. P. Landau and K. Binder. *A guide to Monte Carlo simulations in statistical physics*. Cambridge University Press, 2014.
[3] W. Krauth. *Statistical mechanics: algorithms and computations*, volume 13. Oxford University Press, 2006.
[4] E. P. Bernard, W. Krauth, and D. B. Wilson. Event-chain monte carlo algorithms for hard-sphere systems. *Phys. Rev. E*, 80(5):056704, 2009.
[5] P. Diaconis, S. Holmes, and R. M. Neal. Analysis of a nonreversible markov chain sampler. *Ann. Appl. Probab.*, pages 726–752, 2000.
[6] M. Michel, A. Durmus, and S. Sénécal. Forward event-chain monte carlo: Fast sampling by randomness control in irreversible markov chains. *J. Comput. Graph. Stat.*, 29(4):689–702, 2020.
[7] R. H. Swendsen and J.-S. Wang. Nonuniversal critical dynamics in monte carlo simulations. *Phys. Rev. Lett.*, 58(2):86, 1987.
[8] J. Liu and E. Luijten. Rejection-free geometric cluster algorithm for complex fluids. *Phys. Rev. Lett.*, 92(3):035504, 2004.
[9] J. I. Siepmann and D. Frenkel. Configurational bias monte carlo: a new sampling scheme for flexible chains. *Mol. Phys.*, 75(1):59–70, 1992.
[10] J. J. de Pablo, M Laso, and U. W. Suter. Simulation of polyethylene above and below the melting point. *J. Chem. Phys.*, 96(3):2395–2403, 1992.
[11] G. Mooij and D. Frenkel. The overlapping distribution method to compute chemical potentials of chain molecules. *J. Phys.: Condens. Matter*, 6:3879, 1994.
[12] T. J. H. Vlugt, M. G. Martin, B. Smit, J. I. Siepmann, and R. Krishna. Improving the efficiency of the configurational-bias monte carlo algorithm. *Mol. Phys.*, 94(4):727–733, 1998.
[13] M. G. Martin and J. I. Siepmann. Novel configurational-bias monte carlo method for branched molecules. Transferable potentials for phase equilibria. 2. united-atom description of branched alkanes. *J. Phys. Chem. B*, 103(21):4508–4517, 1999.
[14] C. D. Wick and J. I. Siepmann. Self-adapting fixed-end-point configurational-bias monte carlo method for the regrowth of interior segments of chain molecules with strong intramolecular interactions. *Macromolecules*, 33:7207, 2000.
[15] A. Sepehri, T. D. Loeffler, and B. Chen. Improving the efficiency of configurational-bias monte carlo: A jacobian–
gaussian scheme for generating bending angle trials for linear and branched molecules. J. Chem. Theory Comput., 13(4):1577–1583, 2017.

[16] A. Sepehri, T. D. Loeffler, and B. Chen. Improving the efficiency of configurational-bias monte carlo: Extension of the jacobian–gaussian scheme to interior sections of cyclic and polymeric molecules. J. Chem. Theory Comput., 13(9):4043–4053, 2017.

[17] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. Phys. Lett. B, 195(2):216–222, 1987.

[18] B. Mehlig, D. W. Heermann, and B. M. Forrest. Exact langevin algorithms. Mol. Phys., 76(6):1347–1357, 1992.

[19] B. Mehlig, D. W. Heermann, and B. M. Forrest. Hybrid monte carlo method for condensed-matter systems. Phys. Rev. B, 45(2):679, 1992.

[20] M. B. B. J. M. Tuckerman, B. J. Berne, and G. J. Martyna. Reversible multiple time scale molecular dynamics. J. Chem. Phys., 97(3):1990–2001, 1992.

[21] M. Creutz. Microcanonical monte carlo simulation. Phys. Rev. Lett., 50(19):1411, 1983.

[22] G. E. Crooks. Nonequilibrium measurements of free energy differences for microscopically reversible markovian systems. J. Stat. Phys., 90(5):1481–1487, 1998.

[23] C. Jarzynski. Nonequilibrium equality for free energy differences. Phys. Rev. Lett., 78(14):2600, 1997.

[24] C. Jarzynski. Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach. Phys. Rev. E, 56(5):5018, 1997.

[25] L. R. Pratt. A statistical method for identifying transition states in high dimensional problems. J. Chem. Phys., 85(9):5045–5048, 1986.

[26] C. Dellago, P. G. Bolhuis, F. Csajka, and D. Chandler. Transition path sampling and the calculation of rate constants. J. Chem. Phys., 108(5):1964–1977, 1998.

[27] P. G. Bolhuis, D. Chandler, C. Dellago, and P. L. Geissler. Transition path sampling: Throwing ropes over rough mountain passes, in the dark. Annu. Rev. Phys. Chem., 53(1):291–318, 2002.

[28] F. M. Ytreberg and D. M. Zuckerman. Single-ensemble nonequilibrium path-sampling estimates of free energy differences. J. Chem. Phys., 120(23):10876–10879, 2004.

[29] F. M. Ytreberg and D. M. Zuckerman. Erratum: “single-ensemble nonequilibrium path-sampling estimates of free energy differences” [j. chem. phys. 120, 10876 (2004)]. J. Chem. Phys., 121(10):5022–5023, 2004.

[30] M. Athènes. A path-sampling scheme for computing thermodynamic properties of a many-body system in a generalized ensemble. Eur. Phys. J. B, 38(4):651–663, 2004.

[31] G. Adjanor and M. Athènes. Gibbs free-energy estimates from direct path-sampling computations. J. Chem. Phys., 123(23):234104, 2005.

[32] G. Adjanor, M. Athènes, and J. M. Rodgers. Waste-recycling monte carlo with optimal estimates: Application to free energy calculations in alloys. J. Chem. Phys., 135(4):044127, 2011.

[33] M. Athènes and M.-C. Marinica. Free energy reconstruction from steered dynamics without post-processing. J. Comp. Phys., 229(19):7129–7146, 2010.

[34] The codes used for the simulations are publicly available at [https://github.com/jo-mab/TruncatedMC](https://github.com/jo-mab/TruncatedMC).

[35] D. Frenkel. Speed-up of monte carlo simulations by sampling of rejected states. Proc. Natl. Acad. Sci. U.S.A, 101(51):17571–17575, 2004.

[36] Notice that some within the $2^n-1$ possible states at layer $i$ may coincide. That is often the case when considering discrete systems. In that case, $P_i(y_i)$ is the probability of sampling $y_i$ divided by the multiplicity of $y_i$ at a given $(K_j)_{j=1}^n$ and $x_n$.

[37] B. Oyarzú and B. M. Mognetti. Programming configurational changes in systems of functionalised polymers using reversible intramolecular linkages. Mol. Phys., 116(21-22):2927–2941, 2018.

[38] Miranda Holmes-Cerfon. Simulating sticky particles: A monte carlo method to sample a stratification. The Journal of Chemical Physics, 153(16):164112, 2020.

[39] D. Frenkel, K. J. Schrenk, and S. Martiniani. Monte carlo sampling for stochastic weight functions. Proc. Natl. Acad. Sci. U.S.A, 114(27):6924–6929, 2017.

[40] F. Noé, S. Olsson, J. Kühler, and H. Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. Science, 365(6457), 2019.

[41] We employ the routine available at [https://www.geeksforgeeks.org/python-slicing-extract-k-bits-given-position](https://www.geeksforgeeks.org/python-slicing-extract-k-bits-given-position).