Report

Trimer Flip Monte Carlo (TFMC): *an algorithm to equilibrate glassy trimers*

Nikita ALLAGLO
under the supervision of Camille Scalliet
Original project of Ludovic Berthier and Romain Simon (not published yet)

Summarizing report for my Numerical Project

Master 2 Sciences de la Matière

January 22nd, 2025

I. INTRODUCTION

Glass, ubiquitous as it is in our daily life, stands as one of the drosophilias of research in disordered systems. A deep microscopic understanding of the glassy state of matter remains indeed a challenge for condensed matter physicists [1]. Aside from theoretical perspectives, experimental and computational studies are crucial to shed light on the topic. Computer approaches especially because they provide a complete freedom with the model, the system size, and more. However, the dynamics of glassy systems features increasingly slow dynamics as temperature decreases. This inevitably limits both experiments and simulations so that smarter protocols must be developed [2].

A recent algorithm, Swap Monte Carlo (SMC), produced a breakthrough with a speed-up of the dynamics by several decades (40 orders of magnitude in 2 dimensional systems!) [3, 4]. As a matter of fact, since thermalization to very low temperatures became straightforward, unprecedented studies could be achieved [5, 6]. Ever since, even faster algorithms based upon SMC have been developed [7]. The particle-swap algorithm has nonetheless only been implemented on very specific atomistic models featuring polydispersity [3]. For even more complex liquids, for example polymer melts exhibiting an intrinsic length scale, reaching the glass transition yet remains a challenge. Thus, the framework of SMC remains globally limited.

In this spirit, the aim of this project was to **generalize the SMC algorithm to glassy 3-dimensional trimer mixtures**. SMC was originally studied for 2-dimensional polydisperse systems during my master 1 internship. For assessment purposes, we strongly suggest the reader to check upon the genesis of the project in Appendix A. Aside from the scientific interest, in the context of the *scientific software development* class, another objective was to **create a robust, sharable and well-documentated package for the community**. The interested reader will find the project's companion-website here.

The present report is organized as follows. We firstly give a brief background to glassy systems and their simulations in Section II. We then present our Trimer Flip Monte Carlo (TFMC) algorithm and the model at hand in Section III. Directly next comes a review of TFMC's capabilities in Section IV. Our conclusion terminates the report in Section V.

II. SIMULATING GLASSY SYSTEMS: BACKGROUND

A. Glasses and supercooled liquids

The simplest way to make glass is to cool a material from its liquid state quickly enough. If the transition to the crystal is avoided, particle motion inside the supercooled liquid slows dramatically with decreasing temperature. When the temperature decreases further, the time scale for molecular rearrangements becomes so long that the system falls out of equilibrium with respect to the supercooled liquid state. The material stops flowing and becomes a glass. The *slowness* of glasses is characterised by the structural relaxation time τ_α. In terms of molecular organization, glassy materials appear to be the continuation of the liquid state. Particle configurations closely resemble the disordered structure of the supercooled liquid, as shown in Figure 1.

However, glasses do not easily deform, and thus they constitute a solid form of matter despite not featuring any kind of periodicity. Glasses are also fascinating materials from a fundamental perspective because they represent nonequilibrium disordered states of matter. Understanding them in the context of thermodynamic phase transitions is an exciting challenge that has yet to be solved [1].

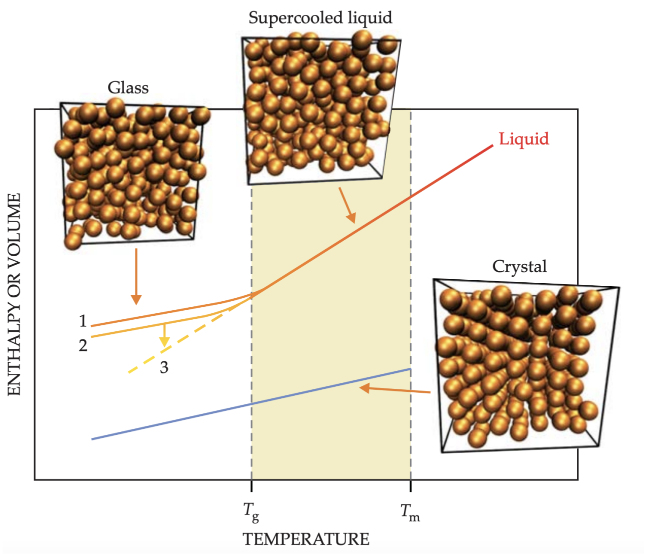

FIG. 1. Volume-Temperature diagram of a cooled liquid. Below T_m, if crystallization is avoided, the liquid is supercooled. Below a different temperature T_g, glass will be formed. Slower cooling produces a denser glass (glass 2 and 3 compared to 1). Reprinted from Ref. [8].

B. Relaxation times

A standard way to portray the timescales of glasses relies on Arrhenius plots [9]: $\log \tau_\alpha$ vs $1/T$ as it is plotted on Fig. 2 (a). Ordinary thermodynamic processes obey the *Arrhenius equation*:

$$\tau_\alpha \sim \tilde{\tau} \exp\left(\frac{\Delta}{k_B T}\right), \tag{1}$$

with Δ some typical constant activation energy and $\tilde{\tau}$ some time unit. For supercooled liquids, an Arrhenius form fits at high temperatures down to some onset temperature T_o as can be seen on Fig. 2 (a). Below T_o, Δ starts developing a T-dependence and the relaxation time grows sharper than an Arrhenius law: it is *super-Arrhenius*. The relaxation time at T_o, noted τ_o is therefore taken as a reference time since it is the point at which the dynamics starts to deviate from (1). The glass transition temperature T_g is empirically defined when $\tau_\alpha(T_g) \sim 10^{12}\tau_o$ [9].

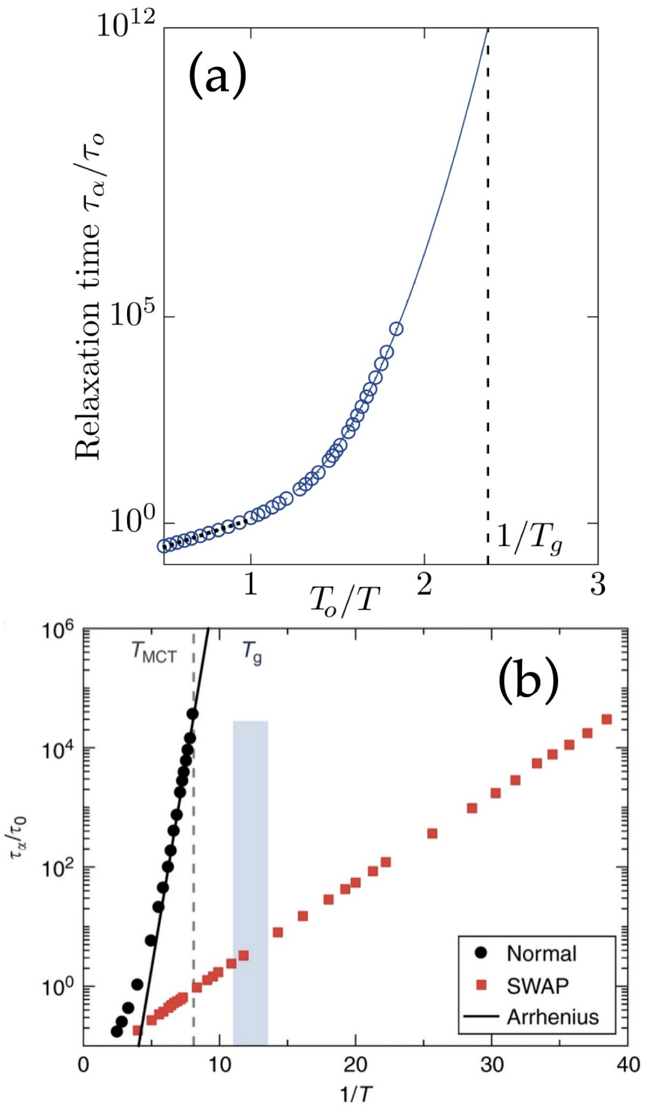

FIG. 2. Arrhenius plots of relaxation times. (a) Pedagogical Arrhenius plot. The temperature has been rescaled by T_o, the onset temperature defined as the point where τ_α deviates from Arrhenius' law (1): the later fits (black dashed line) up to $T_o/T = 1$. T_g is the experimental glass transition temperature: $\tau_\alpha(T_g)$ is the associated relaxation times. Its value is typically $10^{12}\tau_o$. (b) Comparison of Swap and Normal Monte Carlo relaxation times for 2-dimensional polydisperse systems. SWAP can equilibrate systems down to $T \approx 0.3T_g$, where the Arrhenius fit gives $\tau_\alpha^{\text{Normal}} = 10^{46}\tau_o$.

C. Computational approaches for glassy materials

1. Generalities

Computer simulations play a fundamental role in understanding glassy physics, as demonstrated by the abundance of algorithms still being developed [10]. The latter can be clustered into two families [11]. First are molecular dynamics (MD) methods, involving numerical solutions to the equations of motion of the particles via the Newtonian/Hamiltonian formulation. Then, are Monte Carlo (MC) methods which are more targeted towards configuration samplings. Despite their fundamental difference in nature, both methods lead to equivalent dynamical behaviour in the supercooled liquid regime [12]. Therefore, both methods are similarly slow to probe the regime $T \lesssim T_g$. Indeed, simulating the dynamics up to $10^6\tau_o$ (still very far from the glass transition) takes around a week of computation for simple atomistic models.

To tackle this issue, a smart modification of the standard MC method recently allowed to gain several decades in the relaxation time [3]. This is the particle-swap Monte Carlo method, which consists in adding particle-swap moves to standard particle-displacement moves. Indeed, Fig. 2 (b) presents the Arrhenius plots of swap and normal Monte Carlo dynamics for a two-dimensional model. At the glass-transition temperature, the time to fully relax the structure takes 10^{12} time units with MC simulations, while it only takes of order ten time units with SWAP MC. Despite all these achievemnts, SWAP MC's framework remains rigid in the sense that it is highly dependent on the model.

2. *Glassy polymers*

There has built up a considerable understanding of polymer behavior based on the use of highly simplified models ("coarse-grain"). The latter provide insightful results but they are limited in the sense that they do not provide a direct connection between monomer level structure and bulk properties. Atomistic level simulations offer new opportunities for making such predictions ab initio. Using only a knowledge of interatomic forces together with the assumptions of classical mechanics it is possible, at least in principle, to give a complete atomic-level description of a polymer system. Our objective is therefore to simulate off-lattice polymers.

Because SMC is non-local, trying to naively transpose it to polymers would be impossible. A polymer having an intrinsic length and a specific geometry, there is little chance it could "fit" elsewhere. In highly packed glassy melts, they are even further caged. An algorithm inspired by SMC for polymers would therefore require a swap move at the monomer level. As a first step into that direction, we considered simple triangle-shaped ternary trimers. We excluded dimer mixtures as it has been previously showed that binary dispersity does not drastically improve the dynamics compared to standard MC [3]. Up next, we present the *Trimer Flip Monte Carlo* (TFMC) algorithm that generalizes SMC to a novel glass-former model.

III. TFMC: ALGORITHM AND MODELS

A. Trimer glass-forming model

The system of interest involves $M = 1000$ 3-dimensional triangular molecules as sketched on Fig. 3. Each molecule is

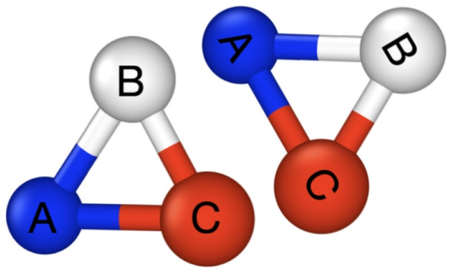

FIG. 3. Representation of trimers with three different diameters. Between atoms of a single molecule, both FENE (3) and WCA potential (2) apply. Between the two molecules, atoms B and A and C and C are subjected to the WCA potential. Figure from Romain Simon.

composed of three particles A, B, C of equal mass $m = 1$ and of diameters $\sigma_A = 0.9$, $\sigma_B = 1.0$, $\sigma_C = 1.1$. A configuration $\mathscr{C}(t)$ as time t is characterized by a set of $N = 3M = 3000$ positions vectors $\{\boldsymbol{r}_i(t)\}$ associated to a set of diameters $\{\sigma_i(t)\}$ each drawn from $\{\sigma_A, \sigma_B, \sigma_C\}$. Particles fill the simulation volume with density $\rho = 1.2$. We employ periodic boundary conditions with the minimum image conventions in a cubic box of size $L = (N/\rho)^{1/3}$. The interactions between particles (both intra-molecular and inter-molecular) are governed by a purely repulsive Weeks-Chandler-Andersen (WCA) potential:

$$U_{\mathrm{WCA}}(\boldsymbol{r}_i, \boldsymbol{r}_j) = 4\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6} + \frac{1}{4}\right), \tag{2}$$

with cutoff at $r_{ij} = 2^{1/6}\sigma_{ij}$ where $r_{ij} = |\boldsymbol{r}_i - \boldsymbol{r}_j|$ and $\sigma_{ij} = (\sigma_i + \sigma_j)/2$.

For bounded particles within the same molecule, we use a purely attractive *Finitely Extensible Nonlinear Elastic* (FENE) potential:

$$U_{\mathrm{FENE}}(\boldsymbol{r}_i, \boldsymbol{r}_j) = -0.5 k_{ij} \bar{R}_{ij} \ln\left(1 - \left(\frac{r_{ij}}{\bar{R}_{ij}}\right)^2\right), \tag{3}$$

where $k_{ij} = 30/\sigma_{ij}^2$ and $\bar{R}_{ij} = 1.5\sigma_{ij}$. At short range, the potential is effectively harmonic and diverges logarithmically at the maximum bond length. This form of interactions was previously used in Ref. [**?**] to investigate out-of-equilibrium aging phenomena in glassy polymers of length 10. For each particle, the associated energy therefore reads

$$U(\mathscr{C}, i) = \sum_{j | r_{ij} < 2^{1/6}\sigma_{ij}} U_{\mathrm{WCA}}(r_{ij}) + \sum_{j|\mathrm{bonds}} U_{\mathrm{FENE}}(r_{ij}) \tag{4}$$

so that the total system energy can be expressed as

$$U_{\mathrm{tot}}(\mathscr{C}) = \frac{1}{2}\sum_i U(\mathscr{C}, i). \tag{5}$$

B. *Monte Carlo sampling strategies*

1. *Standard Monte Carlo method*

The goal of the standard Monte Carlo (MC) method is to sample the phase space according to the Boltzmann distribution. To do so, the algorithm proceeds by picking randomly a particle and trying to displace it by a random small vector. In order to sample equilibrium configurations from the Boltzmann distribution at temperature T, the moves are required to obey the principle of detailed balance. To ensure this, we employ the Metropolis criterion: the probability $\mathrm{P}_{\mathrm{acc}}$ of accepting a given move reads

$$\mathrm{P}_{\mathrm{acc}} = \min\{1, \exp(-\beta\Delta E)\}, \tag{6}$$

where $\beta = 1/k_{\mathrm{B}}T$ and $\Delta E = E' - E$ is the difference in system's potential energy after E' and before E the displacement move. Displacement moves being local, ΔE can be computed efficiently (without looping over $O(N^2)$ terms with (5)) using (4). If the move decreases the potential energy, $\Delta E < 0$, the move is always accepted. Conversely, if the move increases the potential energy, it is accepted with probability $\exp(-\beta\Delta E)$. We implement this by generating a random number $r \in [0,1]$. If $r < \exp(-\beta\Delta E)$ the move is accepted and the particle is displaced. If not, the particle stays at its original position.

2. *The* flip *move*

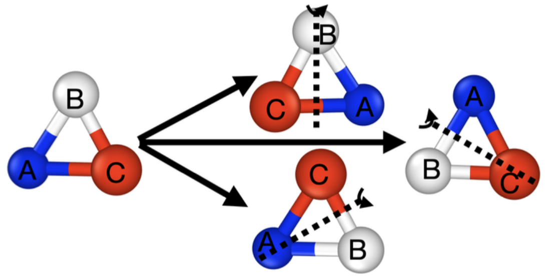

FIG. 4. Sketch of a *flip* move. After randomly selecting the molecule on the left, one of the three *flipped* molecules is accepted or rejected according to Metropolis criterion. Figure from Romain Simon.

In the flip MC algorithm, with a probability of p_{flip}, two out of three particles in a randomly selected molecule are chosen, and their diameters are exchanged. This operation is equivalent to a 180 degrees rotation of the molecule around the axis passing through its center of mass and the unselected particle. Therefore, three possible axes of rotation can be chosen, as shown in Fig. 4. With a probability of $1 - p_{\mathrm{flip}}$, the standard Monte Carlo move is performed, where a randomly selected particle is translated. Since the molecule is flexible and bond lengths are not fixed, this move effectively captures both rotational and translational motions of the molecule. Although these particle-swap (or equivalently trimer-flip) moves are unphysical, we still use the Metropolis criterion to accept or reject the moves, ensuring that the configurations are at equilibrium [13].

A full MC step of the simulation is defined as N consecutive trials of either flip or displacement and is representative of one time step.

We insist that only the diameters of the particles are exchanged. The indices of the particles remain the same over a swap move. This implies that the total displacement of a particle is meaningful: the particle-swap does not result in arbitrarily large jumps. The particle stays at the same position x, y, z, and simply inflates or deflates.

C. Physical observables

To discuss upon the glass-forming ability of our model, we characterize single-particle dynamics through the mean-squared displacement (MSD) and the self-part of the intermediate scattering function (F_s). The MSD is defined as

$$\mathrm{MSD}(\mathscr{C}(t), \mathscr{C}(t_w)) = \left\langle \frac{1}{N}\sum_i |\boldsymbol{r}_i(t) - \boldsymbol{r}_i(t_w)|^2 \right\rangle, \tag{7}$$

where the brackets indicate an average over independent configurations (trajectories and initial configurations) and as before $\mathscr{C}(t)$ denotes the configuration at time t and $\mathscr{C}(t_w)$ the configuration at time t_w. Structural relaxation is estimated with F_s:

$$F_s(q, \mathscr{C}(t), \mathscr{C}(t_w)) = \left\langle \frac{1}{N}\sum_i \cos(\boldsymbol{q}\cdot(\boldsymbol{r}_i(t) - \boldsymbol{r}_i(t_w))) \right\rangle, \tag{8}$$

where the wavevector $\boldsymbol{q}$ is typically evaluated at the first peak of the static structure factor $q \approx 2\pi/\sigma_{\mathrm{max}}$. We also average (8) on multiple orientations of the wavevector $\boldsymbol{q}$ to improve statistical precision. The structural relaxation time τ_α is then defined as the value at which $F_s(\tau_\alpha) = 1/e$. One shall finally keep in mind that these observables only provide a partial characterization of the glassiness of the system. As a matter of fact, one should systematically check on demixing and crystallization with the structure factor and the sixfold bond-orientational order parameter [3]. We did not have time to perform such measurements as we consider to have already worked on a vast quantity of aspects - especially computational ones. Now that we have reviewed the principles of the `TFMC` algorithm, we discuss in the next section the optimization of the code.

D. Optimization of `TFMC`

1. *Verlet-lists*

When implementing off-lattice MC algorithms, it is natural that there will be a huge number of iterations through *for* loops. In the present context, FENE pair-potential is computationally inexpensive as the bonds remain throughout the whole simulation. For the WCA pair-potential, it is not as straightforward. A way to drastically reduce computational time here is to include in the code a list of neighbours for each particle in the system. It is much more computationally efficient to calculate the inter-particle potentials between a particle and its neighbours rather than to compute this with all N particles in the system. The neighbourhood of a particle i is defined as all particles j which verify:

$$r_{ij} \leq 2^{1/6}\sigma_{\max} + r_{\text{skin}}, \tag{9}$$

where $2^{1/6}\sigma_{\max}$ is the maximum radius at which particles are capable of interacting immediately with i (they lie within the cut-off of U_{WCA} [14], whereas r_{skin} includes a set of nearby particles which can potentially interact later, as the system evolves. Furthermore, the computing time for this step can be cut in half by using the fact that if particle i has a neighbour j, then conversely, j has a neighbour i. The neighbour list requires updating after a certain number of MC steps, because over this period, a particle initially outside of the list will enter the neighbourhood of the relevant particle. After time t, the particles will move a maximum distance of $\Delta r_{\max}(t)$. The condition for updating the neighbour list is therefore met if the maximum displacement of the particles exceeds $r_{\text{skin}}/2$. This condition is checked after every sweep and the value of r_{skin} is optimised for the shortest computing time. There will hence be a trade-off between computationally expensive, but infrequent, updates of the neighbour list and iterating through a smaller list of particles which are considered neighbours. For the model defined above, we have found the optimal rskin to be 0.7.

2. *Efficiency of flip moves*

There are very few parameters that can be adjusted to further optimize TFMC. Actually, only two supplementary parameters remain: the maximum displacement per move and the flip probability. About the first one, the speedup in terms of relaxation time is not as considerable. For the sake of completeness, we shall mention that the optimal value was found to be 0.17. On the contrary, p_{flip} is crucial to optimize in that regards as it was established for SMC [3]. There are two obvious limiting cases. For $p_{\text{flip}} = 0$, one recovers the dynamics of a standard Monte Carlo simulation without flip moves. For $p_{\text{flip}} = 1$ only flip moves are attempted; the particle positions are never updated. The optimal choice for p_{flip} is thus the one that minimizes the structural relaxation time τ_α of the system with respect to p_{flip}. We have performed equilibrium measurements of the structural relaxation time (extracted from the self-part of the intermediate scattering function) for a configuration equilibrated at $T = 1.2$. At this temperature, both standard and flip monte carlo managed to equilibrate the system. On Fig. 5, we observe that the relaxation time is minimized for $p_{\text{flip}} \approx 0.2$. We have kept this value for all temperatures as we have checked that the speedup was always optimal (when standard monte carlo manages to equilibrate) for this value.

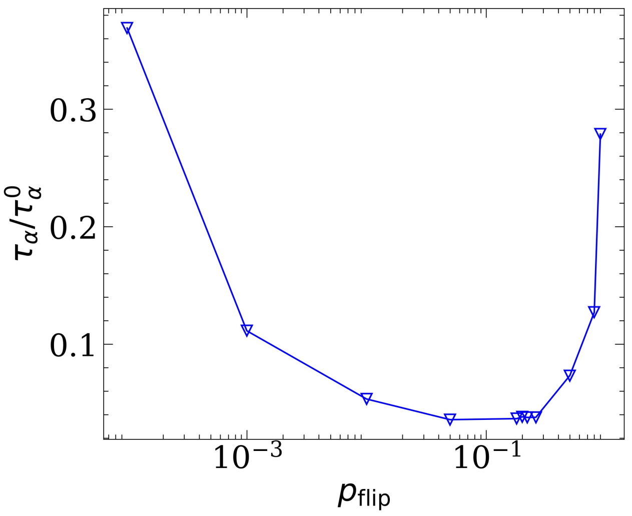

FIG. 5. Structural relaxation time (normalized by the relaxation time found with standard monte carlo, ie $p_{\text{flip}} = 0$) at $T = 1.2$ for different values of p_{flip}. We find that for p_{flip} close to 0.2, the relaxation time is minimized. At this temperature, the speedup is not so dramatic as TFMC is "only" 20 times faster then standard MC.

E. TFMC package

This section will be quite short as a dedicated documentation website for my code is available here.. TFMC was written in C++ as an inspiration of my original M1 project. Not only is the model at hand completely different (add a third dimension and then trimers), we completely rethought the structure of the code by eliminating global arrays for C++ dedicated objects. All the utilities gathered during the scientific software development classes, despite being originally learnt in Python, were now applied in C++. CMakeLists files, header files and dealing with external libraries, testing frameworks, continuous integration, style guidelines, and much more were first learnt and then thoroughly implemented. My knowledge of git was even further developed even if I was already quite accustomed to the later before the classes. The code is not yet made public as it is originally a work of Ludovic Berthier and Romain Simon that is not yet published. It can however be made available upon reasonsable request.

The reader now being familiar with the TFMC algorithm and our implementation of the latter, we will discuss in the next section the performances and achievements of our package.

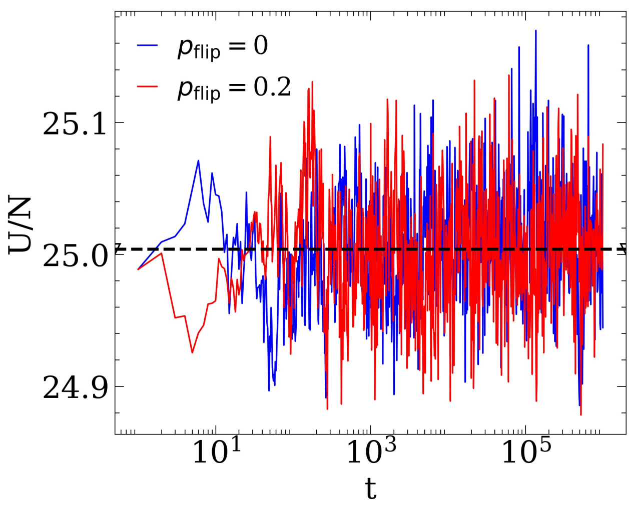

FIG. 6. Evolution of the potential energy per particle for standard and flip monte carlo algorithms running at $T = 2$ from a configuration equilibrated at the same temperature. We observe an analogous time-average for both algorithms and the latter coincides with the one calculated by LAMMPS (black line)

IV. RESULTS: HOW DOES TFMC PERFORM ?

A. Benchmarking the code and equilibration procedures

Whether using standard or flip Monte Carlo, the procedure to monitor equilibration is identical. Simulations start from a configuration (provided to us by our supervisor) equilibrated at a sufficiently high temperature $T_0 = 2$ with a Molecular Dynamics software (LAMMPS). Then, they consist in an instantaneous quench to the target temperature T. We then compute on logarithmically-spaced snapshots the total potential energy and all dynamical observables introduced in Section III C. We thus monitor the potential energy per particle to ensure convergence within thermal fluctuations. We therefore tried to benchmark our code by inspecting the behaviour maintaining the temprature at $T = 2$. We observe on Fig. 6 that both standard and flip monte carlo algorithms return the same time-averaged energy. On top of that, this energy coincides with the one returned by LAMMPS for the exact same model. Then, we look at the self-part of the intermediate scattering function. For this quantity, we check, within statistical fluctuations, both the absence of aging and the complete decorrelation to zero at long times. As specified in the previous section, dynamical observables depend on both the actual configuration but also a certain reference configuration that serves as a starting point for the dynamics. In order to observe aging, we need to wait for the system to relax a certain amount of time so that a new decorrelation cycle is considered. Aging is precisely the phenomenon observed when the system relaxes slowlier if we wait some time in-between decorrelation cycles: it is literally *aging*. At times large enough, aging eventually disappears and we conclude that the system is *thermalized*. In that spirit, we have checked that NO aging was visible for both standard and flip MC for the provided LAMMPS configuration. We decided not to include these results here as real aging plots are covered in the Section IV B 2. As a conclusion to this benchmark run, we can say that our code is correctly implemented.

B. Temperature quenches

Starting from the same equilibrium configuration at $T = 2$ we performed monte carlo simulations with and without the flip move for different temperature quenches. We did not really sought to investigate the equilibrium regime but rather the difference between both methods in reaching equilibrium.

1. *Thermalization*

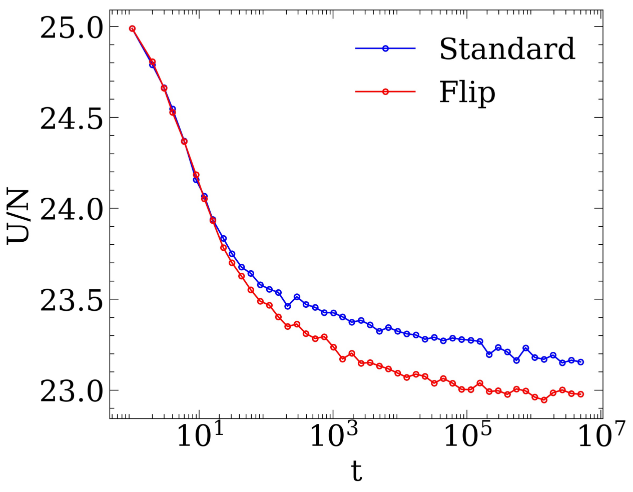

FIG. 7. Evolution of the potential energy per particle for standard and flip monte carlo algorithms following a quench to $T = 0.9$ from $T = 2$.

We first simply compared the potential energy between both methods in an identical run consisting of a quench to $T = 0.9$ from $T = 2$. On Fig. 7, we present the evolution of the potential energy along the run. FTMC seems to have reached equilibrium as the potential energy starts to fluctuates from $t = 300000$ (on the sole observation of the potential energy). On the other hand, standard MC returns an energy decreasing slowlier and slowlier quite above FTMC. This proves that the configuration returned by standard MC are not the true equilibrium states as lower-energies states exist and are obtained by FTMC.

2. *Aging phenomena*

To compare further the differences in thermalization between standard and flip MC, we simulated quenches to $T =$

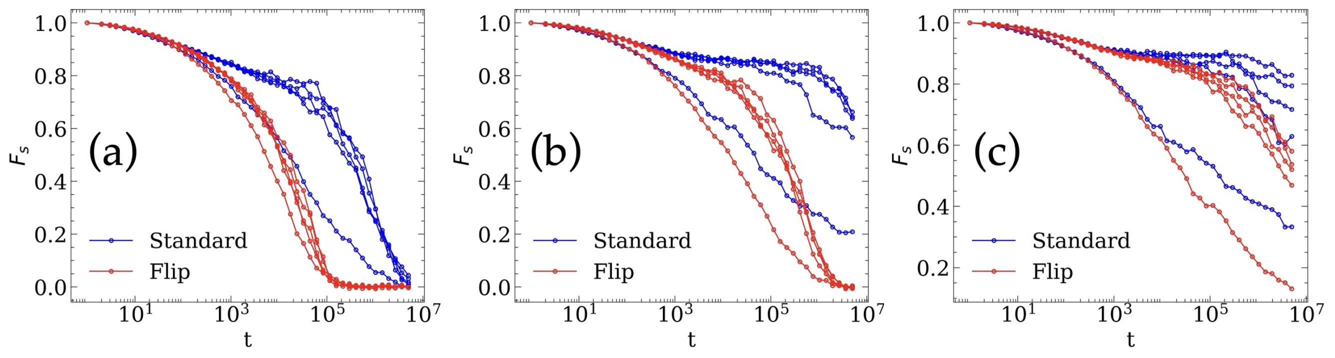

FIG. 8. Aging phenomena at temperatures (a) 1.2, (b) 0.9 and (c) 0.7 for TFMC and standard MC. For each subplot, each curve of the same color represent a distinct t_w, in other words the waiting time before decorrelation is measured. The faster curve represents $t_w = 0$, the one just next on the right $t_w = 1000000 \times 1$, the one next once again $t_w = 1000000 \times 2 = 2000000$. 5 decorrelation cycles were thus measured of each temperature and each algorithm.

$1.2, 0.9, 0.7$ and measured decorrelation cycles of the self-part of the intermediate scattering function on Fig. 8. (a) At $T = 1.2$, aging is not strong for both methods since the curves overlap quite fast. (b) At $T = 0.9$, aging is quite strong between the first and second cycle but eventually all curves overlap in the Flip case (red). For MC (blue), it seems like for each cycle, decorrelation happens even slower. (c) Finally at $T = 0.7$, even TFMC (red) is out-of-equilibrium: the more you go on the right, the slower F_s decorrelates. Nonetheless, the flip still manages to significantly decorrelates, in opposite of standard MC (blue) that gives the impression of eventually reaching a plateau at even longer waiting times.

At this point, we have seen that not only does TFMC converges to lower energies than the standard monte carlo algorithm for low temperatures, it also reaches equilibrium because of the absence of aging at long times. We can therefore use TFMC as a tool to reach equilibrium at temperatures inaccessible to displacements-only monte carlo.

C. Equilibrium dynamics

Following the procedure of equilibration detailed in Section IV A, we equilibrated temperatures from 1.9 down to 0.8 using the TFMC algorithm. We then computed dynamical observables in equilibrium (cf. Section III C) using 5 independent trajectories (random seeds) for both trimer flip and standard displacement monte carlo.

1. *Dynamical observables*

We observe in Fig. 9 the evolution of dynamical observables with different temperatures and both methods. For F_s (a), flip monte carlo manages to structurally relax the system for every temperature. The normal algorithm fails to even show signs of structural relaxation: The analysis is similar for the MSD (b). While at short times particles naturally exhibit ballistic behaviour and eventually remains caged, with flips, they all manage to escape their cage as is concluded from the diffusive behaviour for every temperature at long times. In the standard case, particles eventually manage to diffuse away at $T = 1.2$ but they remain completely caged as the MSD has arrested for lower temperatures.

2. *Relaxation times*

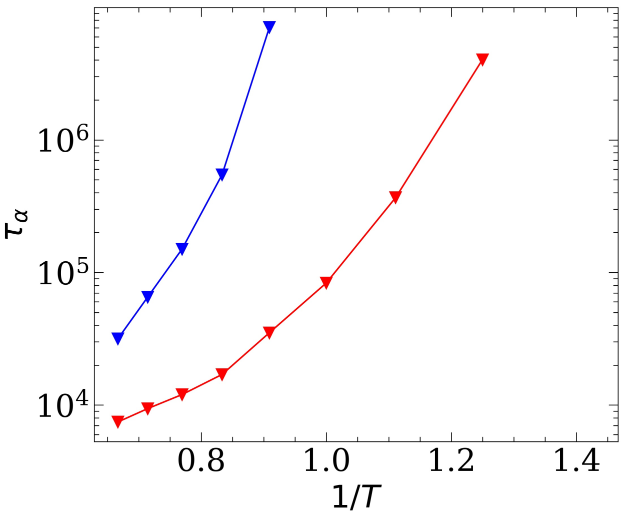

FIG. 10. Structural relaxation times estimated from F_s in equilibrium for TFMC and MC. Red points are for TFMC and blue for MC.

Using F_s in Fig. 9, we extracted the structural relaxation times for each method and gathered them under an Arrhenius plot in Fig. 10. Without flips, we could only extract the relaxation time down to $T = 1.1$ being limited by computing times. For this temperature, TFMC produced a speedup of around 100. For even lower temperatures, the speedup is bigger. On the practical aspect, TFMC managed to equilibrate all the temperatures between 1.9 and 0.8 in matter of a few hours (on a computing cluster) for the last tempera-

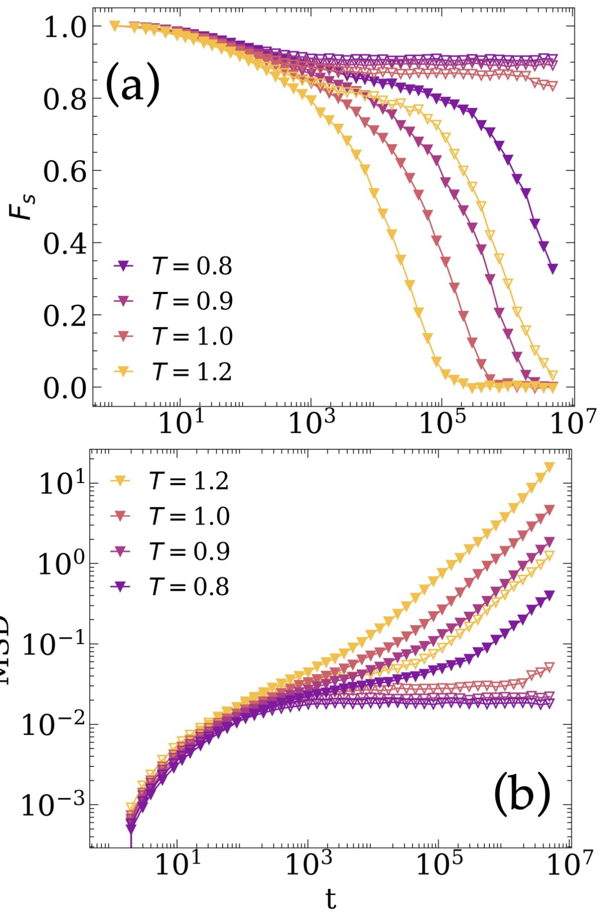

FIG. 9. (a) Self-part of the intermediate scattering function and (b) MSD in equilibrium at various temperatures. The filled symbols represent TFMC with $p_{\text{flip}} = 0.2$ and the empty ones are standard monte carlo calculations. (a) We observe structural relaxation for all plotted temperatures with flips, while normal Monte Carlo yields a plateau for every temperature except $T = 1.2$. (b) At short times, all temperatures and methods present ballistic behaviour and at longer times, TFMC captures diffusive behaviour. On the contrary, we observe a plateau for every temperature except the highest for displacements-only monte carlo.

ture. Its standard counterpart reached equilibrium at the lowest temperature of 1.3 in about ten hours (with the same resources) for this temperature. This technical inquiry actually enables us to better understand the correspondance between relaxation times and computational resources. From rough extrapolations, one expects the standard monte carlo to undergo a glass transition ($\tau_\alpha \sim \tau_o 10^{12}$ where $\tau_o \sim 10^5$ because we can approximately identify the onset temperature as the temperature where the relaxation time growth becomes nonlinear in an Arrhenius plot) at a temperature above 0.8. This quick yet non-rigorous extrapolation therefore suggests that TFMC can reach the glass transition temperature of the standard MC method.

V. CONCLUSION

In this work, we have presented the Trimer Flip Monte Carlo (TFMC) algorithm, specifically designed to investigate the glass-forming behavior of a model system composed of triangular molecules. By incorporating both translational and rotational moves, the TFMC algorithm captures a broad range of dynamics within the system, contributing to its efficient equilibration. Through the implementation of optimized Monte Carlo strategies, including the use of flip moves, we have significantly improved the performance of the model. The flip move, which allows for the exchange of diameters between particles in a molecule, plays a crucial role in enhancing the dynamics and accelerating the relaxation process. We have determined that the optimal flip probability, p_{flip} is around 0.2, offering the best balance between efficiency and accuracy in modeling the system's relaxation dynamics.

The results of our simulations, including the analysis of mean-squared displacement (MSD) and intermediate scattering functions (F_s), offer a thorough characterization of the system's dynamics and structural relaxation. Despite this, we acknowledge that the characterization of glassiness in such systems requires further investigations. The `TFMC` package, developed in C++, serves as a comprehensive and efficient tool for simulating complex molecular dynamics in glass-forming systems. While the code is not yet publicly available due to its basis in unpublished work, it represents a significant step forward in the development of efficient Monte Carlo simulation techniques for glassy materials.

In conclusion, the TFMC method provides a powerful framework for simulating the dynamics of complex systems, with the potential for broad applications in the study of glass transition phenomena and related areas. Future work will focus on extending the capabilities of the algorithm and exploring the detailed structural and dynamic properties of glass-forming systems under various conditions.

[1] L. Berthier and G. Biroli, *Theoretical perspective on the glass transition and amorphous materials*, Reviews of Modern Physics **83**, 587 (2011).
[2] J.-L. Barrat and L. Berthier, *Computer simulations of the glass transition and glassy materials*, Comptes Rendus. Physique **24**, 57 (2023).
[3] A. Ninarello, L. Berthier, and D. Coslovich, *Models and Algorithms for the Next Generation of Glass Transition Studies*, Physical Review X **7**, 021039 (2017).
[4] L. Berthier, E. Flenner, C. J. Fullerton, C. Scalliet, and M. Singh, *Efficient swap algorithms for molecular dynamics simulations of equilibrium supercooled liquids*, Journal of Statistical Mechanics: Theory and Experiment **2019**, 064004 (2019).
[5] L. Berthier, P. Charbonneau, A. Ninarello, M. Ozawa, and S. Yaida, *Zero-temperature glass transition in two dimensions*, Nature Communications **10**, 1508 (2019).
[6] C. Scalliet, B. Guiselin, and L. Berthier, *Thirty Milliseconds*

in the Life of a Supercooled Liquid, Physical Review X **12**, 041028 (2022).

[7] F. Ghimenti, L. Berthier, and F. van Wijland, *Irreversible Monte Carlo Algorithms for Hard Disk Glasses: From Event-Chain to Collective Swaps*, Physical Review Letters **133**, 10.1103/physrevlett.133.028202 (2024).

[8] L. Berthier and M. D. Ediger, *Facets of glass physics*, Physics Today **69**, 40 (2016).

[9] L. Berthier, *Glass transition and amorphous solids*, Beg Rohu Summer school (2019) .

[10] L. Berthier, F. Ghimenti, and F. van Wijland, *Monte Carlo simulations of glass-forming liquids beyond Metropolis*, The Journal of Chemical Physics **161**, 10.1063/5.0225978 (2024).

[11] M. P. Allen and D. J. Tildesley, Computer Simulation of Liquids (Oxford University Press, 2017).

[12] L. Berthier and W. Kob, *The Monte Carlo dynamics of a binary Lennard-Jones glass-forming mixture*, Journal of Physics: Condensed Matter **19**, 205130 (2007).

[13] T. S. Grigera and G. Parisi, *Fast Monte Carlo algorithm for supercooled soft spheres*, Physical Review E **63**, 045102 (2001).

[14] This is not exact since WCA's cut-off is formally $2^{1/6}\sigma_{ij}$. In other words, the maximal value is $2^{1/6}(\sigma_i+\sigma_{\max})/2$. We have overrated the maximum radius so that the average number of neighbours per particle remains constant.

Appendix A: Work initially done in M1 internship

During my master 1 internship, I studied the swap monte carlo algorithm for 2-dimensional polydisperse systems. I was at the moment only a beginner in C++ and in that regards, the progress I made ever since are abundant. The objective of the internship was to determine the origin of the speedup coming from SMC. After writing the code, I was therefore extremely focused on analyzing displacement vectors patterns, movies of particle's evolutions; characterizing collective correlations, etc. The speedup was not at the heart of the project since the main interest was to explain the dynamical behaviours that SMC gives rise to. As I will maybe work with the same team for my PhD, and because I'm particularly fond of the topics related to computational studies of glasses, my supervisor proposed me this project that is already the project of a PhD student of Ludovic Berthier: Romain Simon. The latter gave me all the details about his model and my task was to implement them in C++. This was an occasion for me to put into practice the scientific software development classes and to develop deeper my abilities for compiled languages.