Comptes Rendus

Mécanique

Olga Gorynina, Frédéric Legoll, Tony Lelièvre and Danny Perez

Combining machine-learned and empirical force fields with the parareal algorithm: application to the diffusion of atomistic defects

The scientific legacy of Roland Glowinski / *L'héritage scientifique de Roland Glowinski*

Combining machine-learned and empirical force fields with the parareal algorithm: application to the diffusion of atomistic defects

Olga Gorynina [a, b], **Frédéric Legoll** [*, c, b], **Tony Lelièvre** [a, b] and **Danny Perez** [d]

[a] CERMICS, École des Ponts, Marne-La-Vallée, France

[b] MATHERIALS project-team, Inria, Paris, France

[c] Navier, École des Ponts, Univ Gustave Eiffel, CNRS, Marne-La-Vallée, France

[d] Theoretical Division T-1, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

E-mails: olga.gorynina@enpc.fr (O. Gorynina), frederic.legoll@enpc.fr (F. Legoll), tony.lelievre@enpc.fr (T. Lelièvre), danny_perez@lanl.gov (D. Perez)

Abstract. We numerically investigate an adaptive version of the parareal algorithm in the context of molecular dynamics. This adaptive variant has been originally introduced in [1]. We focus here on test cases of physical interest where the dynamics of the system is modelled by the Langevin equation and is simulated using the molecular dynamics software LAMMPS. In this work, the parareal algorithm uses a family of machine-learning spectral neighbor analysis potentials (SNAP) as fine, reference, potentials and embedded-atom method potentials (EAM) as coarse potentials. We consider a self-interstitial atom in a tungsten lattice and compute the average residence time of the system in metastable states. Our numerical results demonstrate significant computational gains using the adaptive parareal algorithm in comparison to a sequential integration of the Langevin dynamics. We also identify a large regime of numerical parameters for which statistical accuracy is reached without being a consequence of trajectorial accuracy.

* Corresponding author.

1. Introduction

This work is motivated by molecular dynamics (MD) simulations, where one often has to compute ensemble averages or dynamical quantities, which both involve averages over very long trajectories of stochastic dynamics (we refer e.g. to [2] for a general, mathematically oriented presentation of the MD context). Reducing the computational cost of these long time simulations, or at least the wall-clock time it takes to obtain such long trajectories, is thus of great practical interest. As conventional spatial parallelization schemes based on domain decomposition allow for larger system sizes but not longer simulation times [3], one alternative way to speed up such computations is to design accelerated MD approaches based on the parallelization of the problem in the temporal domain [4].

A popular parallel-in-time method for integrating ordinary differential equations is the parareal algorithm, which has been first introduced in [5]. Using a time domain decomposition, the algorithm aims at computing, in an iterative manner, an approximation of the exact solution of the dynamics (see Section 2.1 for a detailed description). The whole time domain is divided into several subintervals. At each iteration, the parareal algorithm utilizes a coarse solver to quickly step through the time domain by computing relatively cheap approximate solutions for all time subintervals, and then simultaneously refines all of these approximate solutions using an accurate fine solver which is applied *in parallel* over each time subinterval. Since the fine propagator corrections (which are expensive to compute) are applied concurrently over the subintervals (and not in a sequential manner from the initial until the final time), reduction in the associated wall-clock time is possible. In contrast, the coarse propagator is applied sequentially over the complete time interval, but its cost is often negligible when compared with the cost of the fine propagator.

In the MD context, it is convenient to consider parareal algorithms where the coarse and the fine propagators integrate dynamics based on different potential energies. More precisely, we assume here that both potential energies are written in terms of the same degrees of freedom (both phase spaces having thus the same dimension), and observe that most physically relevant coarse potentials are likely to have similar numerical-stability constraints as fine potentials (we refer to [6] for parareal variants adapted to slow-fast systems, for which the coarse propagator may integrate an effective system of smaller dimension, where fast degrees of freedom have been averaged out, and hence using a larger time-step; this is not an idea that we pursue here). We are thus going to use the same time-step to integrate both dynamics, the difference in cost stemming from the different complexity for evaluating the potential. In this setting, an adaptive version of the parareal algorithm has recently been introduced in [1]. It is shown there that this adaptive version leads to significantly improved gains (in comparison to the standard version of parareal) on some toy examples.

The main goal of this article is to investigate the performances of the adaptive parareal algorithm for realistic problems of physical interest. To that aim, we focus on the MD simulation of the diffusion of a self interstitial atom (SIA) in a body-centered cubic (BCC) tungsten lattice. The simulation is performed using the LAMMPS [7] molecular dynamics software (Large-scale Atomic/Molecular Massively Parallel Simulator), a software which is very broadly used within the materials science community. To model the tungsten lattice, we consider several interatomic potentials in two families: machine-learned spectral neighbor analysis potentials (SNAP) [8] and embedded-atom method potentials (EAM) [9].

This article is organized as follows. In Section 2, we briefly review the classical parareal algorithm and the adaptive version introduced in [1]. We also discuss the implementation of our method in LAMMPS, including some issues related to the fact that we use *as is* the time-stepping scheme implemented in LAMMPS (for the sake of having a non-intrusive implementation).

This raises some unexpected difficulties (some calculation details related to these questions are postponed until Appendix A). We next present our SIA simulations in Section 3. We describe the MD settings we have chosen and demonstrate the accuracy (both from a trajectorial and a statistical viewpoint) of the results obtained using the adaptive parareal algorithm in comparison to reference results obtained using a standard sequential integration of the dynamics. We also discuss in Section 3 the significant computational gains obtained when using the adaptive parareal algorithm (on our test cases, this gain varies between 3 and 20 depending on the choice of the coarse propagator and of the time-step).

2. Algorithm

The present work focuses on the parallelization in time of the Langevin dynamics

$$\begin{cases} dq(t) = p(t)\,dt, \\ dp(t) = -\nabla V(q(t))\,dt - \gamma\,p(t)\,dt + \sqrt{2\gamma\beta^{-1}}\,dW(t), \end{cases} \tag{1}$$

with initial condition (q_0, p_0). Here $(q(t), p(t)) \in \mathbb{R}^{3m} \times \mathbb{R}^{3m}$ are the positions and momenta at time $t \in [0, T]$ of m particles in the three-dimensional space, $V : \mathbb{R}^{3m} \to \mathbb{R}$ is the potential energy of the system, $\gamma > 0$ is the friction coefficient, β is (up to a multiplicative constant) the inverse temperature and $W(t)$ is a standard Brownian motion in dimension $3m$. In (1), we have set the mass of each particle to unity for simplicity, but the generalization to non-identity mass matrix is straightforward (and the numerical tests discussed in Section 3 are actually performed with the physically relevant values of the particle masses).

Let us introduce a uniform grid on the time interval $[0, T]$ with N time-steps of length $\Delta t = T/N$:

$$0 = t_0 < t_1 = \Delta t < \cdots < N\Delta t = t_N = T.$$

Because it will be useful in the context of the parareal algorithm, each time-step Δt is itself subdivided into L time-steps of length $\delta t = \Delta t / L$ for some $L \geq 1$.

The integrator for the Langevin dynamics (1) implemented in LAMMPS is a modification of the BBK scheme [10] with an effective force which takes into account the damping term and the fluctuation term associated to the white noise, in addition to the physical force $-\nabla V$. Let us consider $L + 1$ independent standard Gaussian random variables, denoted $G_0, G_1, \ldots, G_L$. The first iterate of the scheme has a particular expression:

$$\begin{cases} p_{1/2} = p_0 - \dfrac{\delta t}{2}\nabla V(q_0) - \dfrac{\delta t}{2}\gamma\,p_0 + \dfrac{1}{2}\sqrt{2\gamma\beta^{-1}\delta t}\,G_0, \\ q_1 = q_0 + \delta t\,p_{1/2}, \\ p_1 = p_{1/2} - \dfrac{\delta t}{2}\nabla V(q_1) - \dfrac{\delta t}{2}\gamma\,p_{1/2} + \dfrac{1}{2}\sqrt{2\gamma\beta^{-1}\delta t}\,G_1, \end{cases} \tag{2}$$

where we recall that (q_0, p_0) is the initial condition. The subsequent iterates (for $1 \leq \ell \leq L - 1$) are given by

$$\begin{cases} p_{\ell+1/2} = p_\ell - \dfrac{\delta t}{2}\nabla V(q_\ell) - \dfrac{\delta t}{2}\gamma\,p_{\ell-1/2} + \dfrac{1}{2}\sqrt{2\gamma\beta^{-1}\delta t}\,G_\ell, \\ q_{\ell+1} = q_\ell + \delta t\,p_{\ell+1/2}, \\ p_{\ell+1} = p_{\ell+1/2} - \dfrac{\delta t}{2}\nabla V(q_{\ell+1}) - \dfrac{\delta t}{2}\gamma\,p_{\ell+1/2} + \dfrac{1}{2}\sqrt{2\gamma\beta^{-1}\delta t}\,G_{\ell+1}. \end{cases} \tag{3}$$

The first iterate (2) differs from the next ones (3) in two ways. First, if we were to set $\ell = 0$ in the first line of (3), we would need to know $p_{-1/2}$ to compute $p_{1/2}$. However, $p_{-1/2}$ is of course not defined. It is replaced in the first line of (2) by p_0. Second, and more importantly, the random variable G_0 is only used in the first iteration, whereas all the other random variables $\{G_\ell\}_{1 \leq \ell \leq L-1}$

are used twice (G_ℓ is used in the last step of iterate $\ell - 1$ and in the first step of iterate ℓ). Stated otherwise, the first iterate (2) uses *two* independent random variables, whereas each subsequent iterate only uses *one* additional independent random variable. Since we are going to work with relatively small values of L in the sequel, this difference has far-reaching consequences, e.g. on the equilibrium kinetic temperature simulated by the numerical scheme, as will be discussed in details in Section 2.4.

Remark 1. The modification (2)–(3) of the BBK scheme is particularly well-suited to adapt to the Langevin equation an implementation of the standard Verlet algorithm used to integrate deterministic Hamiltonian dynamics. Indeed, as in a Verlet scheme, for $\ell \geq 1$, the total force (sum of force field, damping and fluctuation) to compute $p_{\ell+1/2}$ from p_ℓ in the first step of iterate ℓ is exactly the same as the total force to compute p_ℓ from $p_{\ell-1/2}$ in the last step of iterate $\ell - 1$. The total force hence only needs to be computed once per time-step.

2.1. *Parareal method*

To integrate (1) over a time interval of length Δt, we consider two propagators: $\mathscr{F}_{\Delta t}$ is a fine, expensive propagator which accurately approximates the exact solution of (1), and $\mathscr{C}_{\Delta t}$ is a coarse, less expensive propagator which is also less accurate. In the MD context, $\mathscr{F}_{\Delta t}$ and $\mathscr{C}_{\Delta t}$ are often integrators with a fixed discretization scheme and with the same time-step, but run on different potential energy landscapes. In what follows, and similarly to [1], the fine propagator $\mathscr{F}_{\Delta t}$ performs L iterations of (3) (with the time-step $\delta t = \Delta t / L$), where $V \equiv V_\mathscr{F}$ is the reference, accurate potential energy. The coarse propagator $\mathscr{C}_{\Delta t}$ also performs L iterations of (3) (with the same time-step δt), but where $V \equiv V_\mathscr{C}$ is now a coarse, approximate potential energy. As explained above, for both propagators $\mathscr{F}_{\Delta t}$ and $\mathscr{C}_{\Delta t}$, the first of these L iterates is given in the form of (2) instead of (3). We emphasize that, to ensure convergence of the parareal algorithm, both propagators $\mathscr{F}_{\Delta t}$ and $\mathscr{C}_{\Delta t}$ should use the *same* random variables. We denote by $\{G_{\ell,n}\}_{0 \leq \ell \leq L, 1 \leq n \leq N}$ the random variables used to propagate the system from the initial time to the final time $N\Delta t$.

The classical parareal method (as first introduced in [5] and reformulated in [11]) is an iterative, parallel-in-time algorithm, which computes the trajectory over $[0, T]$ by using domain (in time) decomposition. It proceeds as follows. Starting from the initial condition (q_0, p_0), the algorithm first performs a sequential coarse propagation to compute $\{(q_n^0, p_n^0)\}_{0 \leq n \leq N}$:

$$\left(q_{n+1}^0, p_{n+1}^0\right) = \mathscr{C}_{\Delta t}\left(q_n^0, p_n^0\right), \qquad \left(q_0^0, p_0^0\right) = (q_0, p_0). \tag{4}$$

Suppose now that we have at hand some numerical trajectory $\{(q_n^{k-1}, p_n^{k-1})\}_{0 \leq n \leq N}$, obtained at the previous iteration $k - 1$. The new parareal solution $\{(q_n^k, p_n^k)\}_{0 \leq n \leq N}$ is computed from the following scheme:

$$\left(q_{n+1}^k, p_{n+1}^k\right) = \mathscr{C}_{\Delta t}\left(q_n^k, p_n^k\right) + \mathscr{F}_{\Delta t}\left(q_n^{k-1}, p_n^{k-1}\right) - \mathscr{C}_{\Delta t}\left(q_n^{k-1}, p_n^{k-1}\right), \qquad \left(q_0^k, p_0^k\right) = (q_0, p_0). \tag{5}$$

We thus propagate the system (q_n^{k-1}, p_n^{k-1}), in parallel over the time-windows $[n\Delta t, (n+1)\Delta t]$, according to both the coarse and the fine propagators, thereby obtaining $\mathscr{F}_{\Delta t}(q_n^{k-1}, p_n^{k-1})$ and $\mathscr{C}_{\Delta t}(q_n^{k-1}, p_n^{k-1})$. We next compute, still in parallel, the jumps $\mathscr{F}_{\Delta t}(q_n^{k-1}, p_n^{k-1}) - \mathscr{C}_{\Delta t}(q_n^{k-1}, p_n^{k-1})$. We next perform a sequential propagation, from the initial condition and using the coarse propagator that we correct according to the precomputed jumps. We note that the fine solver is only used in the parallel part of the algorithm (the fine solver is applied in each interval $[n\Delta t, (n+1)\Delta t]$ independently of the other intervals), while the sequential part of the algorithm only calls the coarse propagator. The random numbers $\{G_{\ell,n}\}_{0 \leq \ell \leq L, 1 \leq n \leq N}$ which are used are the same at each parareal iteration k.

The parareal algorithm has been successfully applied to many problems. We refer to [12] for a reformulation of the algorithm in a more general setting that relates the parareal strategy to earlier time-parallel algorithms (see also [13, 14]). Several variants of the algorithm have been proposed for specific applications: multiscale-in-time problems (see e.g. [15, 16], [6] and [17]), Hamiltonian ODEs or PDEs [18, 19], stochastic differential equations [20–22], reservoir simulations [23], applications in fluid and solid mechanics [24, 25], to mention but a few. This work is based on the adaptive version of the algorithm that we introduced in [1]. We also wish to mention [26] for another work in the direction of designing adaptive variants of the parareal algorithm.

In the following, we denote by $\{(q_n^{\mathrm{ref}}, p_n^{\mathrm{ref}})\}_{n \geq 0}$ the reference trajectory obtained by a sequential use of the fine propagator $\mathscr{F}_{\Delta t}$ (and which is thus expensive to compute): $(q_{n+1}^{\mathrm{ref}}, p_{n+1}^{\mathrm{ref}}) = \mathscr{F}_{\Delta t}(q_n^{\mathrm{ref}}, p_n^{\mathrm{ref}})$. An important feature of the parareal method is that, on the interval $[0, N\Delta t]$, it is guaranteed to converge to this reference solution in at most $k = N$ iterations (see e.g. [11]): for any $k \geq N$, we have $(q_n^k, p_n^k) = (q_n^{\mathrm{ref}}, p_n^{\mathrm{ref}})$ for any $0 \leq n \leq N$. In practice, convergence is often reached much sooner, therefore providing computational gains (a noteworthy exception is the case of hyperbolic problems, where a larger number of iterations is often needed to reach convergence, as observed e.g. in [18, 19]; we also refer to the analysis in [1]).

Let us introduce the relative error between the reference trajectory $\{q_n^{\mathrm{ref}}\}_{0 \leq n \leq N}$ and the parareal trajectory $\{q_n^k\}_{0 \leq n \leq N}$ at the iteration k:

$$E_{\mathrm{ref}}\left(q^k, N\right) = \frac{\sum_{n=1}^{N} \left| q_n^{\mathrm{ref}} - q_n^k \right|}{\sum_{n=1}^{N} \left| q_n^{\mathrm{ref}} \right|}.$$

The error E_{ref} cannot be computed in practice, since we do not have access to the reference trajectory $\{q_n^{\mathrm{ref}}\}_{n \geq 0}$. Therefore, in order to monitor the convergence of the parareal method along the iterations, we introduce the relative error between two consecutive parareal trajectories:

$$E\left(q^{k-1}, q^k, N\right) = \frac{\sum_{n=1}^{N} \left| q_n^k - q_n^{k-1} \right|}{\sum_{n=1}^{N} \left| q_n^{k-1} \right|}. \tag{6}$$

As mentioned above, we have $q_n^k = q_n^{\mathrm{ref}}$ for any $k \geq n$, and thus $E(q^{k-1}, q^k, N) = 0$ for $k \geq N+1$.

In our numerical experiments discussed below, we proceed with the parareal iterations until the accuracy reaches the user-chosen threshold δ_{conv}, namely until $E(q^{k-1}, q^k, N) < \delta_{\mathrm{conv}}$. We denote by k_{conv} the number of parareal iterations required to reach this accuracy.

Remark 2. Even though the errors E_{ref} and E are not (because of their denominator) invariant by translation, this definition of error still makes sense in practice for the examples that we consider in this work. As explained below (see Section 3.1), we enforce periodic boundary conditions on the system, and we have checked that, along the various trajectories that we have considered, all the atoms of the system always stay in the same periodic cell. There is hence no significant global translation of the system, which remains close to the origin. We also note that, should need be, the definition of E_{ref} and E can easily be modified to be translation-invariant.

Remark 3. Note that other criteria of convergence can be envisioned. For example, one could imagine a criteria based on the statistical properties of the numerical solution $\{(q_n^k, p_n^k)\}_{0 \leq n \leq N}$. We do not address this question in this work.

The classical parareal algorithm, as described above, is presented as Algorithm 1.

Let us denote by C_f (resp. C_c) the cost of a single evaluation of the fine integrator $\mathscr{F}_{\Delta t}$ (resp. coarse integrator $\mathscr{C}_{\Delta t}$). Assuming that the communication time is zero, the wall-clock time cost of the parareal algorithm is $N C_c + k_{\mathrm{conv}}((C_f + C_c) + N C_c)$. In contrast, the wall-clock time cost of

---

Algorithm 1: Parareal algorithm

Numerical parameters: N, δ_{conv}

1 Compute $\left\{\left(q_n^{\text{cur}}, p_n^{\text{cur}}\right)\right\}_{0 \le n \le N}$: $\left(q_0^{\text{cur}}, p_0^{\text{cur}}\right) := (q_0, p_0)$ and $\left(q_{n+1}^{\text{cur}}, p_{n+1}^{\text{cur}}\right) := \mathscr{C}_{\Delta t}\left(q_n^{\text{cur}}, p_n^{\text{cur}}\right)$;

2 Set $k := 0$ and $\delta := 2\delta_{\text{conv}}$;

3 **while** $\delta \ge \delta_{\text{conv}}$ **do**

4 $k := k + 1$;

5   Define $\left\{\left(q_n^{\text{prev}}, p_n^{\text{prev}}\right)\right\}_{0 \le n \le N}$ as $\left(q_n^{\text{prev}}, p_n^{\text{prev}}\right) := \left(q_n^{\text{cur}}, p_n^{\text{cur}}\right)$ for any $0 \le n \le N$;

6   Compute $\{J_n\}_{0 \le n \le N-1}$ in parallel: $J_n := \mathscr{F}_{\Delta t}\left(q_n^{\text{prev}}, p_n^{\text{prev}}\right) - \mathscr{C}_{\Delta t}\left(q_n^{\text{prev}}, p_n^{\text{prev}}\right)$;

7   **for** $n \leftarrow 0$ **to** $N - 1$ **by** 1 **do**

8 $\left(q_{n+1}^{\text{cur}}, p_{n+1}^{\text{cur}}\right) := \mathscr{C}_{\Delta t}\left(q_n^{\text{cur}}, p_n^{\text{cur}}\right) + J_n$;

9   **end**

10   Compute the relative error $\delta = E\left(q^{\text{prev}}, q^{\text{cur}}, N\right)$;

11 **end**

Output of the algorithm: $\left\{\left(q_n^{\text{cur}}, p_n^{\text{cur}}\right)\right\}_{0 \le n \le N}$ and $k_{\text{conv}} := k$

---

a sequential propagation according to the fine propagator is NC_f. We are thus able to define the wall-clock gain of the parareal method as

$$\Gamma\left(\delta_{\text{conv}}, N\right) = \frac{NC_f}{NC_c + k_{\text{conv}}\left(\left(C_f + C_c\right) + NC_c\right)}.$$

If we additionally assume that the cost of the coarse propagations is negligible in comparison with the cost of the fine propagator (i.e. $NC_c \ll C_f$), we get $\Gamma(\delta_{\text{conv}}, N) = \Gamma^{\text{ideal}}(\delta_{\text{conv}}, N)$, where the ideal gain is defined by $\Gamma^{\text{ideal}}(\delta_{\text{conv}}, N) := \frac{N}{k_{\text{conv}}}$. Note that the total CPU effort spent by the parareal algorithm, which is equal to $NC_c + k_{\text{conv}}(N(C_f + C_c) + NC_c)$, is of course larger than the total CPU effort spent by a sequential fine integration (which is equal to NC_f). Beyond the wall-clock time gain provided by usual parallelism over processors, note that computational efficiency considerations can further increase the attractiveness of parareal. For example, MD simulations on small systems can be very inefficient on modern Graphical Processing Units (GPUs), given the extremely high level of data parallelism that can be accommodated by the hardware. In this case, parareal could be implemented to execute all the fine-grained steps simultaneously on the *same* GPU instead of requiring parallelization over multiple GPUs, which could significantly improve the efficiency of the calculation and lead to a decrease in the net computational effort required to carry out the simulation.

2.2. *Adaptive parareal method*

The work [1] has introduced an adaptive version of the parareal algorithm, motivated by the fact that, when applied to MD problems, the classical parareal algorithm suffers from various limitations (in particular, possible intermediate blow-up of the trajectory and lack of computational gain in the case of too long time horizons). The new approach introduced in [1] consists in adaptively dividing (on the basis of the relative error between two consecutive parareal trajectories) the computational domain $[0, N\Delta t]$ in several subdomains. For that, we have to revisit the definition of the error (6) for an arbitrary time-slab $[N_{\text{init}}\Delta t, N_{\text{final}}\Delta t]$, for some fixed $N_{\text{final}} \ge N_{\text{init}} \ge 0$. We naturally extend (6) as

$$E\left(q^{k-1}, q^k, N_{\text{init}}, N_{\text{final}}\right) = \frac{\sum_{n=N_{\text{init}}}^{N_{\text{final}}} \left|q_n^k - q_n^{k-1}\right|}{\sum_{n=N_{\text{init}}}^{N_{\text{final}}} \left|q_n^{k-1}\right|}.$$

The adaptive parareal method proceeds as follows (see Algorithm 2). We fix two parameters $\delta_{\mathrm{conv}} > 0$ (convergence parameter) and $\delta_{\mathrm{expl}} > 0$ (explosion threshold parameter, which satisfies $\delta_{\mathrm{expl}} > \delta_{\mathrm{conv}}$) and start by running the classical parareal algorithm on the whole time-slab $[0, T]$. As in the classical parareal method, for every parareal iteration k, we check whether the trajectory has reached convergence on the whole time slab, i.e. whether

$$E\left(q^{k-1}, q^k, 0, N\right) < \delta_{\mathrm{conv}}.$$

If this is the case then we stop the iterations. If this is not the case, then

- either $E(q^{k-1}, q^k, 0, N) \leq \delta_{\mathrm{expl}}$, and we then proceed with the next parareal iteration over the whole time-slab $[0, T]$.
- or $E(q^{k-1}, q^k, 0, N) > \delta_{\mathrm{expl}}$, which means that the relative error at the parareal iteration k is too large. We then give up on trying to reach convergence on the whole time-slab $[0, T]$ and decide to shorten it. In practice, we look for the smallest n such that $E(q^{k-1}, q^k, 0, n) > \delta_{\mathrm{expl}}$, denote it n^{cur} and shorten the original time-slab $[0, N\Delta t]$ to $[0, n^{\mathrm{cur}}\Delta t]$.

  We then proceed with the next parareal iterations on this new time-slab, that will possibly be further shortened, until the relative error E, on the shortened time-slab $[0, n^{\mathrm{cur}}\Delta t]$, becomes smaller than δ_{conv}. We have then reached convergence on the current time-slab and proceed with the subsequent part of the time range. We thus define the new (tentative) time-slab as $[n^{\mathrm{cur}}\Delta t, N\Delta t]$ and start again the adaptive parareal algorithm. This procedure is repeated until the final time T is reached.

The adaptive parareal method, as described above, is presented as Algorithm 2. We denote there by N_{slab} the number of time-slabs in which the whole time range $[0, N\Delta t]$ is eventually divided:

$$[0, N\Delta t] = \bigcup_{1 \leq i \leq N_{\mathrm{slab}}} \left[N_{\mathrm{init}}^i \Delta t, N_{\mathrm{final}}^i \Delta t \right],$$

with

$$N_{\mathrm{init}}^1 = 0, \quad N_{\mathrm{final}}^i = N_{\mathrm{init}}^{i+1} \quad \text{and} \quad N_{\mathrm{final}}^{N_{\mathrm{slab}}} = N.$$

For any $1 \leq i \leq N_{\mathrm{slab}}$, we denote by k_{conv}^i the number of parareal iterations required to reach convergence on $[N_{\mathrm{init}}^i \Delta t, N_{\mathrm{final}}^i \Delta t]$.

We now evaluate the cost of the adaptive algorithm, assuming that the communication time is zero. Each time-slab $[N_{\mathrm{init}}^i \Delta t, N_{\mathrm{final}}^i \Delta t]$ eventually identified by the algorithm has been determined in an iterative process. We denote by $[N_{\mathrm{init}}^i \Delta t, N_{\mathrm{final}}^{i,j} \Delta t]$ the time-slab considered at the jth iteration of that process (note that only the endpoint of the time-slab depends on j). Denoting m_i the number of iterations required to identify a sufficiently short time-slab such that convergence of the parareal iterations can be reached, we have that $\{N_{\mathrm{final}}^{i,j}\}_{1 \leq j \leq m_i}$ is a decreasing sequence with $N_{\mathrm{final}}^{i,1} = N$ and $N_{\mathrm{final}}^{i,m_i} = N_{\mathrm{final}}^i$. For any $1 \leq j < m_i$, the adaptive algorithm performs $k^{i,j}$ additional parareal iterations before realizing that the current time slab $[N_{\mathrm{init}}^i \Delta t, N_{\mathrm{final}}^{i,j} \Delta t]$ is too long. For $j = m_i$, the adaptive algorithm performs k^{i,m_i} additional parareal iterations before reaching convergence. The total number of iterations that have been performed to reach convergence on the ith time-slab is given by

$$k_{\mathrm{conv}}^i = \sum_{j=1}^{m_i} k^{i,j}.$$

The wall-clock time cost of the adaptive algorithm is

$$\mathrm{Cost} = \sum_{i=1}^{N_{\mathrm{slab}}} \left[\left(N - N_{\mathrm{init}}^i\right) C_c + \sum_{j=1}^{m_i} k^{i,j} \left(\left(C_f + C_c\right) + \left(N_{\mathrm{final}}^{i,j} - N_{\mathrm{init}}^i\right) C_c \right) \right]. \tag{7}$$

Algorithm 2: Adaptive parareal algorithm

    Numerical parameters: N, δ_{conv}, δ_{expl}

1   Set $N_{\text{init}} := 0$, $N_{\text{final}} := 0$, and $\delta := (\delta_{\text{conv}} + \delta_{\text{expl}})/2$;

2   Set $\left(q_0^{\text{cur}}, p_0^{\text{cur}}\right) := \left(q_0, p_0\right)$;

3   Set $N_{\text{slab}} := 0$;

4   **while** $N_{\text{final}} < N$ **do**

5   **if** $\delta < \delta_{\text{expl}}$ **then**

6 $N_{\text{init}} := N_{\text{final}}$; // initialization for a new time-slab

7 $N_{\text{final}} := N$;

8   Compute $\left\{\left(q_n^{\text{cur}}, p_n^{\text{cur}}\right)\right\}_{N_{\text{init}} \le n \le N_{\text{final}}}$: $\left(q_{n+1}^{\text{cur}}, p_{n+1}^{\text{cur}}\right) := \mathscr{C}_{\Delta t}\left(q_n^{\text{cur}}, p_n^{\text{cur}}\right)$ for all $N_{\text{init}} \le n \le N_{\text{final}} - 1$;

9 $N_{\text{slab}} := N_{\text{slab}} + 1$;

10 $k_{\text{conv}}^{N_{\text{slab}}} := 0$;

11   **end**

12   Set $\delta := (\delta_{\text{conv}} + \delta_{\text{expl}})/2$;

13   **while** $\delta \in \left[\delta_{\text{conv}}, \delta_{\text{expl}}\right]$ **do**

14   Define $\left\{\left(q_n^{\text{prev}}, p_n^{\text{prev}}\right)\right\}_{N_{\text{init}} \le n \le N_{\text{final}}}$ as $\left(q_n^{\text{prev}}, p_n^{\text{prev}}\right) := \left(q_n^{\text{cur}}, p_n^{\text{cur}}\right)$ for all $N_{\text{init}} \le n \le N_{\text{final}}$;

15   Compute $\{J_n\}_{N_{\text{init}} \le n \le N_{\text{final}} - 1}$ (in parallel): $J_n := \mathscr{F}_{\Delta t}\left(q_n^{\text{prev}}, p_n^{\text{prev}}\right) - \mathscr{C}_{\Delta t}\left(q_n^{\text{prev}}, p_n^{\text{prev}}\right)$;

16 $k_{\text{conv}}^{N_{\text{slab}}} := k_{\text{conv}}^{N_{\text{slab}}} + 1$;

17   **for** $n \leftarrow N_{\text{init}}$ **to** $N_{\text{final}} - 1$ **by** 1 **do**

18 $\left(q_{n+1}^{\text{cur}}, p_{n+1}^{\text{cur}}\right) := \mathscr{C}_{\Delta t}\left(q_n^{\text{cur}}, p_n^{\text{cur}}\right) + J_n$;

19   Update the relative error $\delta = E\left(q^{\text{prev}}, q^{\text{cur}}, N_{\text{init}}, n+1\right)$;

20   **if** $\delta > \delta_{\text{expl}}$ **then**

21 $N_{\text{final}} := n$;

22   break; // exit the for loop if condition satisfied; we also exit the while loop since δ is too large

23   **end**

24   **end**

25   **end**

26 **end**

    Output of the algorithm: $\left\{\left(q_n^{\text{cur}}, p_n^{\text{cur}}\right)\right\}_{0 \le n \le N}$, N_{slab}, $\left\{k_{\text{conv}}^i\right\}_{1 \le i \le N_{\text{slab}}}$

The first term corresponds to the coarse propagation on the initially proposed i^{th} time slab, namely $[N_{\text{init}}^i \Delta t, N \Delta t]$. The algorithm then proceeds with parareal iterations, on a slab which is possibly iteratively shortened. The wall-clock gain of the adaptive algorithm is

$$\Gamma_{\text{adapt}}\left(\delta_{\text{expl}}, \delta_{\text{conv}}, N\right) = \frac{N C_f}{\text{Cost}}. \tag{8}$$

If we additionally assume that the cost of the coarse propagator is negligible in comparison with the cost of the fine propagator (i.e. $N C_c \ll C_f$), we obtain $\Gamma_{\text{adapt}}(\delta_{\text{expl}}, \delta_{\text{conv}}, N) \approx \Gamma_{\text{adapt}}^{\text{ideal}}(\delta_{\text{expl}}, \delta_{\text{conv}}, N)$, where the ideal gain is defined by

$$\Gamma_{\text{adapt}}^{\text{ideal}} \left(\delta_{\text{expl}}, \delta_{\text{conv}}, N \right) := \frac{N}{\displaystyle\sum_{i=1}^{N_{\text{slab}}} k_{\text{conv}}^{i}}. \tag{9}$$

2.3. *Implementation in LAMMPS*

In order to provide a *non-intrusive* implementation of the Algorithms 1 and 2 in LAMMPS, we did not modify the source code of LAMMPS but proceeded as follows. The logic of the parareal algorithm is implemented in a so-called master code written in Python, while the force calculations and timestepping is carried out in LAMMPS. For each given (q_n^k, p_n^k), the Python code calls LAMMPS through an API in order to request an advance of the system (using either $V_{\mathscr{F}}$ or $V_{\mathscr{C}}$) over a time range Δt (by making L time-steps of length δt), thereby computing $\mathscr{F}_{\Delta t}(q_n^k, p_n^k)$ and $\mathscr{C}_{\Delta t}(q_n^k, p_n^k)$. The jumps $\mathscr{F}_{\Delta t}(q_n^k, p_n^k) - \mathscr{C}_{\Delta t}(q_n^k, p_n^k)$ are then computed by the master code. In the sequential part, Python first requests LAMMPS to compute $\mathscr{C}_{\Delta t}(q_n^{k+1}, p_n^{k+1})$ and then adds the jump to obtain $(q_{n+1}^{k+1}, p_{n+1}^{k+1})$. This implementation does not run the fine integrations in parallel, but it already allows us to monitor the performance of the algorithm in terms of required parareal iterations.

We mentioned previously that, in order to ensure convergence of the parareal algorithm, both propagators should use the same array of random variables $\{G_{\ell,n}\}_{0 \le \ell \le L}$ in the scheme (2)–(3), when propagating the system from time $n\Delta t$ to time $(n+1)\Delta t$. An ideal solution would be to first draw these random numbers and then to feed them to the coarse and fine propagators as required. However, in our implementation, the computation of $\mathscr{F}_{\Delta t}$ and $\mathscr{C}_{\Delta t}$ is performed within LAMMPS where there is no possibility to control the random increments used at each step of the scheme (2)–(3) (with $0 \le \ell \le L$). We can only control the seed of the random number generator.

To circumvent this difficulty, we proceed as follows to reach the final time $T = N\Delta t$:

- in Python, we draw a list of N random numbers that we denote $\{S_n\}_{1 \le n \le N}$ and that will be used as seeds by LAMMPS. Since LAMMPS expects the seed to be an integer number in a given range, we draw $\{S_n\}_{1 \le n \le N}$ as a random sequence of i.i.d. integers uniformly distributed within that range.
- when we enter LAMMPS to compute $\mathscr{F}_{\Delta t}$ or $\mathscr{C}_{\Delta t}$ to integrate the system from time $(n-1)\Delta t$ to $n\Delta t$ (at any parareal iteration k), we provide the LAMMPS random number generator with the seed S_n. On the basis of that seed, the random number generator of LAMMPS provides the Gaussian increments $G_{\ell,n}$ for any $0 \le \ell \le L$. Because the seed is the same for $\mathscr{F}_{\Delta t}$ and $\mathscr{C}_{\Delta t}$ and at all parareal iterations, both schemes make use of the same sequence of increments $\{G_{\ell,n}\}_{0 \le \ell \le L}$ (which of course remains the same at each parareal iteration).

We have made sure that the pseudo-random number generators of Python and of LAMMPS are of different type (by default, Python uses Mersenne Twister[1] while LAMMPS uses a Marsaglia random number generator[2]). The procedure that we have adopted, which is constrained by practical considerations, hence seems to be reasonable. We however note that a perfectly clean procedure would require to use parallel pseudo-random number generators (see e.g. [27]), in order to avoid correlations between the sequence of numbers used by LAMMPS on the different intervals $[(n-1)\Delta t, n\Delta t]$.

The Python code corresponding to Algorithm 2 can be found in the GitHub repository at

`https://github.com/OlgaGorynina/Parareal_MD`

---

[1]`https://docs.python.org/3/library/random.html`
[2]`https://docs.lammps.org/fix_gld.html`

2.4. *Kinetic temperature*

In our implementation of the parareal algorithm, LAMMPS is called N times from Python, each of these calls asking LAMMPS to perform L steps of the scheme (2)–(3). This procedure generates a trajectory over a time interval of total length $N\Delta t = NL\delta t$. We explain in this section that, surprisingly enough, this procedure is not equivalent to performing directly NL time-steps of (2)–(3), and actually introduces a bias in the observed kinetic temperature. This is because the first step of the scheme (namely (2)) reads differently from the subsequent ones (performed following (3)). In particular, one has to implement a carefully chosen temperature schedule in order to recover the correct kinetic temperature.

Consider the scheme which consists in making L steps of the BBK scheme (that is, we start with (2) and next perform $L-1$ steps of (3)). We show in Appendix A.1 (on the basis of analytical computations, see (18), which are confirmed by the numerical results of Table 5) that the equilibrium kinetic temperature reached by the scheme in the limit $N \to \infty$ is

$$K_{\text{eq}} = \beta^{-1}\left(1 - \frac{1}{2L}\right) + O(\delta t). \tag{10}$$

Of course, if L is very large, which is the standard regime in which LAMMPS is expected to be used, then K_{eq} is close to the target value β^{-1}, up to a time discretization error of the order of $O(\delta t)$. This explains why the scheme (2)–(3) is justified for MD simulations that are usually performed for long time horizons. But this is not our situation: we actually work with relatively small values of L (in practice, in the numerical experiments described in Section 3, we even work with $L = 1$). Therefore, we are in a situation where the kinetic temperature (10) is quite different from β^{-1}.

In order to guarantee that the scheme indeed reaches the correct equilibrium kinetic temperature, we propose to use a time-dependent temperature (which is indeed an option available in LAMMPS). In the first (resp. third) line of the time-integrator, instead of considering a fluctuating term of the form

$$\sqrt{2\gamma\beta^{-1}\delta t}\, G_\ell \ \left(\text{resp.} \sqrt{2\gamma\beta^{-1}\delta t}\, G_{\ell+1}\right)$$

as in (3), we use a term of the form

$$\sqrt{2\gamma\beta_\ell^{-1}\delta t}\, G_\ell \ \left(\text{resp.} \sqrt{2\gamma\beta_{\ell+1}^{-1}\delta t}\, G_{\ell+1}\right),$$

where the temperature thus depends on the iterate number. More precisely, we consider the following scheme (compare with (2)–(3)). The first iterate of the scheme is given by

$$\begin{cases} p_{1/2} = p_0 - \dfrac{\delta t}{2}\nabla V(q_0) - \dfrac{\delta t}{2}\gamma p_0 + \dfrac{1}{2}\sqrt{2\gamma\beta_0^{-1}\delta t}\, G_0, \\[2mm] q_1 = q_0 + \delta t\, p_{1/2}, \\[2mm] p_1 = p_{1/2} - \dfrac{\delta t}{2}\nabla V(q_1) - \dfrac{\delta t}{2}\gamma p_{1/2} + \dfrac{1}{2}\sqrt{2\gamma\beta_1^{-1}\delta t}\, G_1, \end{cases} \tag{11}$$

where we recall that (q_0, p_0) is the initial condition. The subsequent iterates (for $\ell \geq 1$) are given by

$$\begin{cases} p_{\ell+1/2} = p_\ell - \dfrac{\delta t}{2}\nabla V(q_\ell) - \dfrac{\delta t}{2}\gamma p_{\ell-1/2} + \dfrac{1}{2}\sqrt{2\gamma\beta_\ell^{-1}\delta t}\, G_\ell, \\[2mm] q_{\ell+1} = q_\ell + \delta t\, p_{\ell+1/2}, \\[2mm] p_{\ell+1} = p_{\ell+1/2} - \dfrac{\delta t}{2}\nabla V\left(q_{\ell+1}\right) - \dfrac{\delta t}{2}\gamma p_{\ell+1/2} + \dfrac{1}{2}\sqrt{2\gamma\beta_{\ell+1}^{-1}\delta t}\, G_{\ell+1}. \end{cases} \tag{12}$$

The first iterate (11) differs from the next ones (12) in the same two ways as (2) differs from (3). We note that the temperature schedule appears in the scheme (11)–(12) in such a way that the fluctuating force in the last step of the iterate ℓ is equal to the fluctuating force in the first step of

the iterate $\ell + 1$, thereby preserving the implementation of the scheme as a Verlet scheme with a force including the damping and the fluctuation terms, in addition to the force field term (see Remark 1).

As shown in Appendix A.2, for each value of L, several schedules $\{\beta_\ell\}_{0 \le \ell \le L}$ are possible in order to reach the correct equilibrium temperature. The only choice which is valid whatever L is given by

$$\beta_\ell^{-1} = C_\ell \, \beta^{-1} \quad \text{with } C_0 = 3 \text{ and } C_\ell = 1 \text{ for any } \ell \ge 1. \tag{13}$$

In the particular case when $L = 1$ (which is the only case we consider in the numerical experiments of Section 3), another possible choice (and this is the one we make in Section 3) is

$$\beta_0^{-1} = \beta_1^{-1} = 2\,\beta^{-1}. \tag{14}$$

Details about the implementation in LAMMPS of such schedules are provided in Remark 8 within Appendix A.2.

Remark 4. A strategy different from the one we have adopted here would be to allow ourselves to modify the source code of LAMMPS, in an *intrusive* fashion. In that case, it is of course possible to modify the time-scheme used to integrate the Langevin equation (1), and choose a numerical scheme which provides the correct kinetic temperature (up to discretization errors) for any value of L.

3. Numerical results

3.1. *MD settings*

We demonstrate the efficiency of the adaptive algorithm described above by simulating the diffusion of a self-interstitial atom (SIA) in a tungsten lattice. To do so, we consider a perfect periodic lattice of tungsten atoms and insert an additional tungsten atom. This extra atom can relax to a number of equilibrium positions. Because of thermal fluctuations, the interstitial atom hops from one equilibrium state to another one in a metastable fashion (see e.g. [2, 3] for a comprehensive description of the MD context). We choose to work with SIA because of relatively small activation energies for diffusion, in contrast with the diffusion of other lattice defects, such as vacancies for instance, for which the activation energies are much larger. Because the activation energy is small, we can afford to run trajectories where we observe *several* jumps, which makes it possible to make statistical analysis on these jumps. For instance, we can compute the mean transition time with a reasonable statistical accuracy. As mentioned above, we use LAMMPS to perform these molecular dynamics simulations, and use the adaptive parareal algorithm discussed above (with $L = 1$) to compute the trajectories.

We consider a system containing 129 tungsten atoms, forming a BCC lattice (except for the interstitial atom) with periodic boundary conditions. The temperature (equal to β^{-1} up to a multiplicative constant) is set to 2000 K and the damping parameter satisfies $\gamma^{-1} = 1$ ps. These values for temperature and damping parameter are used for all calculations in the current section. We consider two choices of time-step, $\delta t = 2$ fs and $\delta t = 0.5$ fs.

To compute the forces on the atoms, we consider two types of interatomic potentials:

- empirical force fields based on a physically informed parameterized expression; we use the Embedded-Atom Method (EAM) potential [9];
- machine-learned force fields using generic features as input to describe the chemical environment of each atom; we use Spectral Neighbor Analysis Potentials (SNAP) [8], where the parameters of the generic features are optimized using machine-learning techniques to reproduce (on some small configurations) the energies, forces, and stress

tensors obtained by ab-initio computations; these potentials are denoted SNAP-6, …, SNAP-205, depending on the number of features. The larger the number of features, the more accurate the potential is with respect to ab-initio results, but also the more computationally expensive it is.

Table 1 presents typical computational times required to perform 5000 iterations of the scheme (11)–(12) (in a purely sequential manner), with various interatomic potentials, as measured on a laptop computer equipped with Intel(R) Core(TM) i7-10610U at 1.80GHz. We see that, on average, SNAP-205 is 175 times (resp. 2600 times) more expensive than SNAP-6 (resp. EAM).

Table 1. Computational time (in seconds) required to perform (in a sequential manner) 5000 iterations of (11)–(12), with different interatomic potentials, on a standard laptop.

Potential	Comp. time
EAM	0.6923
SNAP-6	10.20
SNAP-15	22.65
SNAP-31	58.06
SNAP-56	151.2
SNAP-92	373.2
SNAP-141	861.2
SNAP-205	1787

For every simulation described below, we first initialize the system with the following equilibration procedure. We consider a sample containing the 128 tungsten atoms located on a BCC lattice. A self-interstitial tungsten atom is then inserted in the sample and we minimize the total energy of the 129 atoms system (to drive the system in an equilibrium position, i.e. a local minimum of the energy). We then perform 10,000 steps of (2)–(3), in a purely sequential manner and using the fine potential $V_{\mathscr{F}}$. The resulting thermalized configuration is our initial condition. Starting from there, we will next propagate the system for N steps, either sequentially using $V_{\mathscr{F}}$, or in a parareal manner using the fine potential $V_{\mathscr{F}}$ and a coarse potential $V_{\mathscr{C}}$. We use the Voronoi analysis of OVITO [28] to identify the location of the SIA, which allows us to estimate transition times between metastable states.

The reference solution $\{(q_n^{\mathrm{ref}}, p_n^{\mathrm{ref}})\}_{0 \le n \le N}$ is computed in a sequential manner with the SNAP-205 potential, which we denote $V_{\mathscr{F}}^{\mathrm{SNAP-205}}$. The corresponding reference average SIA residence time (that we compute on the basis of a single long trajectory of $N = 400,000$ time-steps) is 0.63568 ps.

We have at our disposal several potentials to be used as coarse potentials in the (adaptive) parareal algorithm. If we choose $V_{\mathscr{C}}$ to be close to $V_{\mathscr{F}}^{\mathrm{SNAP-205}}$, we may hope to need few iterations to reach convergence, but the discrepancy in term of cost between the fine and the coarse potential may be too small to observe any computational gain. From (7)–(8), we indeed know that the gain of the parareal algorithm crucially depends on the ratio C_f/C_c. In what follows, we consider two strategies for the (adaptive) parareal algorithm:

- Strategy I consists in using the SNAP-6 potential (denoted by $V_{\mathscr{C}}^{\mathrm{SNAP-6}}$) as coarse potential;
- Strategy II consists in using the EAM potential (denoted by $V_{\mathscr{C}}^{\mathrm{EAM}}$) as coarse potential.

In both strategies, the fine potential is the SNAP-205 potential $V_{\mathscr{F}}^{\mathrm{SNAP-205}}$. We thus have $C_f/C_c \sim$ 175 in the first case and $C_f/C_c \sim 2600$ in the second case (on an Intel(R) Core(TM) i7-10610U machine).

In the following sections, we investigate the accuracy of the parareal trajectories, first in a strong, trajectorial sense (in Section 3.2), second in a statistical sense (in Section 3.3). We conclude by discussing the observed computational gains (in Section 3.4).

3.2. *Convergence of adaptive trajectories*

The objective of this section is to check the trajectorial convergence of the adaptive parareal algorithm. We consider the physical system described above and perform a trajectory consisting of $N = 1500$ time-steps (after equilibration), either using only the reference potential $V_{\mathscr{F}}^{\text{SNAP}-205}$ or using the parareal algorithm. Along the trajectory, we register the number of time-steps (of length δt) that the system spends in a given well before hoping to another one. The results, which have been computed with the time-step $\delta t = 2$ fs and the explosion threshold $\delta_{\text{expl}} = 0.35$, are presented in Table 2.

The first line corresponds to the reference trajectory. The next two lines correspond to the parareal results using the strategy I, and the last two lines correspond to the parareal results using the strategy II (for two values of the convergence threshold δ_{conv}).

We see that, when we use $\delta_{\text{conv}} = 10^{-5}$, the parareal results (for strategy I and II) differ from the reference results. The parareal trajectory is not sufficiently close to the reference trajectory to obtain accurate results in terms of residence times (recall that the system is chaotic, so a small difference at some point of the trajectory may lead to a large difference in terms of the time spent in a metastable state). In contrast, when we set $\delta_{\text{conv}} = 10^{-10}$, we observe that both parareal strategies essentially give the same SIA residence times as the reference solution. In the case I (resp. case II), the first seven (resp. first six) residence times are exactly reproduced by the parareal trajectory. With a small enough value of δ_{conv}, it is thus possible to obtain convergence on a long time interval (the horizon T is sufficiently large to witness several exits of metastable states on $[0, T]$).

Note that we do not observe (and actually do not expect to observe) an exact trajectorial convergence in the limit $\delta_{\text{conv}} \to 0$, because of the inherent chaoticity of the trajectories: even when δ_{conv} is as small as 10^{-10}, the values of the last jumps are different between the parareal trajectories and the reference one, even though they are driven by exactly the same noise. This is often not important in practice since many commonly considered quantities of interest actually only depend on the law of the trajectories (statistical quantities) and not on the exact realization for a given noise. This is why we investigate the statistical accuracy of the parareal scheme in the next section.

3.3. *Statistical analyses*

The aim of MD simulations, especially those using stochastic equations of motion, is most often to generate statistically correct trajectories, in contrast to very accurately obtaining a trajectory corresponding to a specific random number sequence. Quantities of interest include thermodynamical averages (which are computed as time averages along the trajectory) or dynamical information in a mean sense, such as average residence times in metastable states.

In this section, we monitor the residence times of the SIA along a trajectory of $N = 100,000$ time-steps (computed with the time-step $\delta t = 2$ fs), and compute from the observed exit events an average residence time along with a confidence interval (in practice, 50 trajectories of 2,000 times-steps are computed).

We first consider the reference trajectory, computed only using the reference potential $V_{\mathscr{F}}^{\text{SNAP}-205}$. Along the N time-steps, we observe slightly more than 300 exit events. On Figure 1, we plot the average of the SIA residence times, as more and more transition events are taken into

Table 2. SIA residence times (in units of δt) for the reference and the parareal trajectories (for example, in the reference computations, the system spends 122 time-steps in the first well, then hops to a second well where it stays 23 time-steps, ...). We mark in bold the residence times of the parareal trajectories that exactly agree with those of the reference trajectory ($\delta t = 2$ fs and $\delta_{\text{expl}} = 0.35$).

|	SIA residence times for reference solution
	[**122**, **23**, **27**, **476**, **14**, **32**, **560**, 245]
δ_{conv}	SIA residence times for strategy I
10^{-5}	[273, 1226]
10^{-10}	[**122**, **23**, **27**, **476**, **14**, **32**, **560**, 217, 26, 2]
δ_{conv}	SIA residence times for strategy II
10^{-5}	[63, 27, 16, 36, 19, 34, 332, 972]
10^{-10}	[**122**, **23**, **27**, **476**, **14**, **32**, 575, 15, 28, 31, 156]

account to compute the average. We find that the average residence time is $T_{\text{avg}} = 320.26$, with the confidence interval [268.74; 371.78] (here and throughout the article, confidence intervals are computed in a way such that the corresponding expectation belongs to the confidence interval with a probability of 0.95). Since we work with the time-step $\delta t = 2$ fs, this corresponds to an average residence time of 640 fs, a time consistent with the one obtained in Section 3.1 (0.63568 ps) from a very long trajectory.
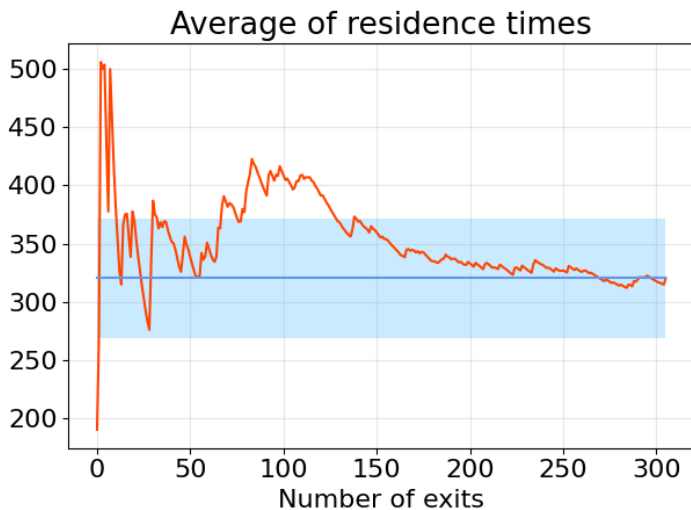
Figure 1. Reference results, in units of δt. The average residence time $T_{\text{avg}} = 320.26$ is shown by the dark blue line, and the confidence interval [268.74; 371.78] is represented in light blue.

We next repeat the same experiments, but with the EAM potential as reference potential. Results are shown on Figure 2. In this case, the mean residence time is $T_{\text{avg}} = 99.8$ (with the confidence interval [92.89; 106.71]). These results are very far from the reference solution results. We thus cannot rely on the EAM model to accurately predict the residence times. A coupling strategy (such as the parareal algorithm) is needed.

Remark 5. Note that, in this section, all the Gaussian increments that we use (i) for the reference computations, (ii) for the EAM simulations and (iii) for the various parareal simulations presented below are independent one from each other. Likewise, the initial conditions for the three types of computations are independent. We have made this choice in view of our aim to monitor the *statistical* accuracy of the EAM or parareal computations with respect to the reference computations. On the other hand, within one type of computations (e.g. parareal simulations for given $V_{\mathscr{C}}$ and δ_{conv}), we have of course used the same Gaussian increments for the two propagators $\mathscr{C}_{\Delta t}$ and $\mathscr{F}_{\Delta t}$ and at all parareal iterations.
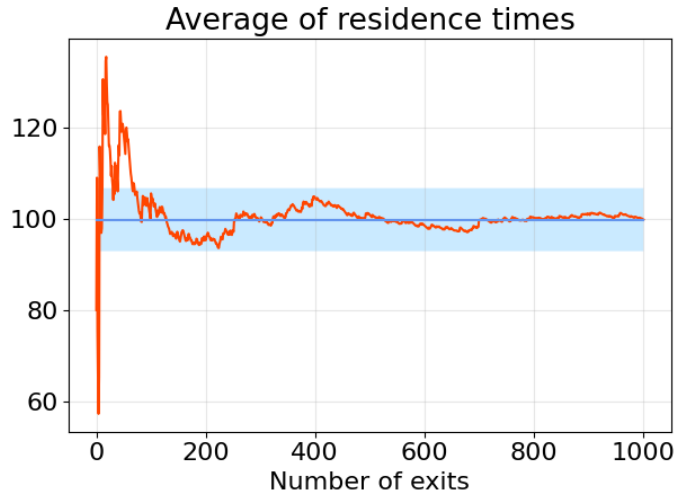

Figure 2. Results when modelling the system with the EAM potential, in units of δt. The average residence time $T_{\text{avg}} = 99.8$ is shown by the dark blue line, and the confidence interval $[92.89; 106.71]$ is represented in light blue.

We now consider parareal results obtained using the strategy II (coupling the fine potential $V_{\mathscr{F}}^{\text{SNAP}-205}$ with the coarse potential $V_{\mathscr{C}}^{\text{EAM}}$), with the explosion threshold $\delta_{\text{expl}} = 0.35$ and with various values of the convergence threshold δ_{conv} (for this test, we have not considered the strategy I because, as shown in Section 3.4 below, it provides smaller computational gains than the strategy II). Results obtained with $\delta_{\text{conv}} = 10^{-3}$ (resp. $\delta_{\text{conv}} = 10^{-5}$, $\delta_{\text{conv}} = 10^{-10}$) are shown on Figure 3 (resp. Figure 4, Figure 5). For these three values of δ_{conv}, we obtain confidence intervals for the mean residence time which overlap with the reference confidence interval obtained on Figure 1 (and that we have reproduced in orange on Figures 3, 4 and 5). The parareal results are thus statistically consistent with the reference results. This is true even in the case $\delta_{\text{conv}} = 10^{-3}$, which is a too large value to expect trajectorial convergence. We also note that, in the case $\delta_{\text{conv}} = 10^{-10}$, the statistical accuracy is *not* a consequence of a trajectorial convergence of the parareal trajectories to the reference trajectories. Indeed, as pointed out in Remark 5, the initial configurations and the random noises used in the reference computations differ from those used in the parareal computations (each converged parareal trajectory is thus different from any of the reference trajectories). Moreover, for a given sequence of random noises, we have observed in Section 3.2 that the parareal trajectory differs from the reference trajectory (computed with the fine potential) after roughly 10 jumps.

For the sake of completeness, we have also considered the aggressive choice $\delta_{\text{conv}} = 10^{-1}$ (results not shown). As could be expected, for this very large value of convergence threshold, the confidence intervals on residence times do not overlap at all and the parareal results are inaccurate.
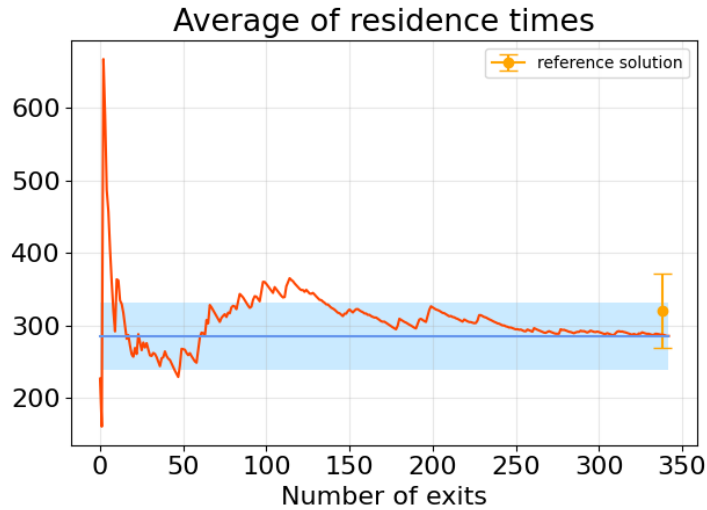
Figure 3. Parareal results (strategy II) with $\delta_{\mathrm{conv}} = 10^{-3}$, in units of δt. The average residence time $T_{\mathrm{avg}} = 285.714$ is shown by the dark blue line, and the confidence interval $[239.73; 331.7]$ is represented in light blue. We also show (in orange) the reference result (with its confidence interval).
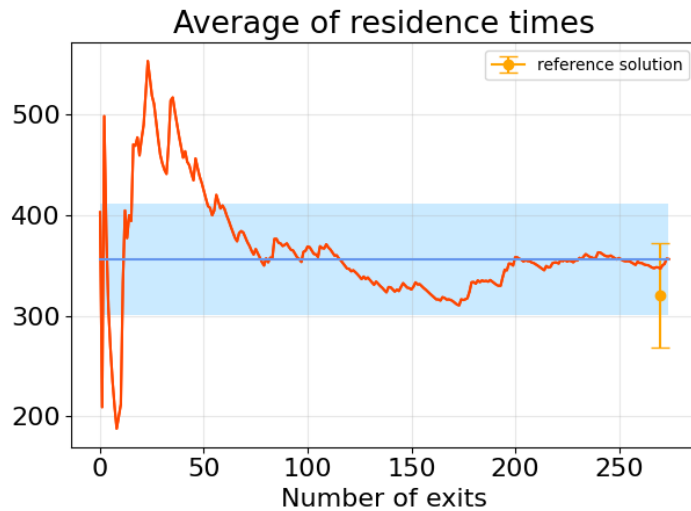
Figure 4. Parareal results (strategy II) with $\delta_{\mathrm{conv}} = 10^{-5}$, in units of δt. The average residence time $T_{\mathrm{avg}} = 356.36$ is shown by the dark blue line, and the confidence interval $[300.9; 411.83]$ is represented in light blue. We also show (in orange) the reference result (with its confidence interval).

3.4. *Computational gains*

We now investigate the wall-clock gains obtained using the adaptive parareal algorithm, for various values of the time-step δt, the convergence threshold δ_{conv} and the explosion threshold δ_{expl}. We fix the time horizon at $T = N\delta t$ with $N = 2000$. The gain is computed using (8) (we also compute the ideal gain defined by (9)), where the costs are those measured on a laptop computer (see Table 1).
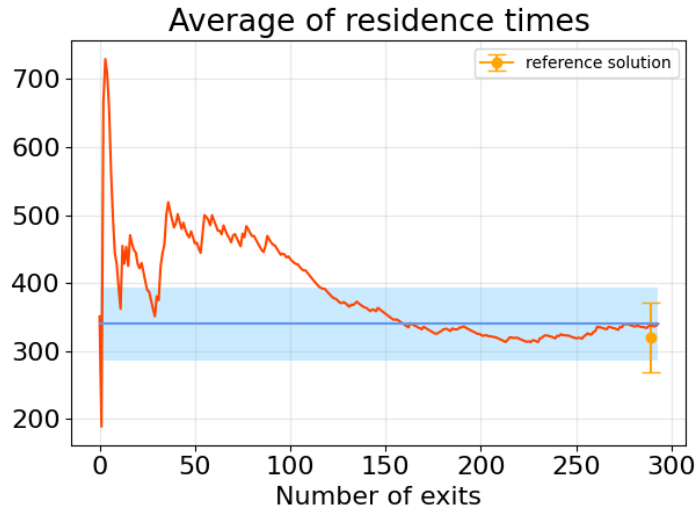
Figure 5. Parareal results (strategy II) with $\delta_{\mathrm{conv}} = 10^{-10}$, in units of δt. The average residence time $T_{\mathrm{avg}} = 340.14$ is shown by the dark blue line, and the confidence interval $[287.82; 393.45]$ is represented in light blue. We also show (in orange) the reference result (with its confidence interval).

We first consider the parareal strategy I, and collect in Table 3 the values of the gain. If we use the time-step $\delta t = 2$ fs, then the maximal value of the (actual) gain is equal to 2.95 and is attained for $\delta_{\mathrm{expl}} = 0.35$ and $\delta_{\mathrm{conv}} = 10^{-3}$. If we work with the smaller time-step $\delta t = 0.5$ fs, then the gain increases to 9.08 (attained for $\delta_{\mathrm{expl}} = 0.3$ and $\delta_{\mathrm{conv}} = 10^{-3}$). Table 4 collects the gains for the parareal strategy II. The gain reaches the value 5.18 (with the choice $\delta_{\mathrm{expl}} = 0.35$ and $\delta_{\mathrm{conv}} = 10^{-3}$) when using $\delta t = 2$ fs, and increases up to 19.05 (with the choice $\delta_{\mathrm{expl}} = 0.3$ and $\delta_{\mathrm{conv}} = 10^{-3}$) when using $\delta t = 0.5$ fs.

Overall, the gain obtained using strategy II is always larger than the one obtained using strategy I. This is expected since the ratio C_f / C_c is more than ten times larger in the case II. In addition, the gain is always larger when considering $\delta t = 0.5$ fs rather than $\delta t = 2$ fs.

As expected, the largest values of the gain are obtained when $\delta_{\mathrm{conv}} = 10^{-3}$ (if we decrease δ_{conv} to 10^{-5} or 10^{-10}, more parareal iterations are requested to achieve convergence). In terms of δ_{expl}, the gain seems to describe a "bell shape", in the sense that it increases when δ_{expl} is very small and decreases when δ_{expl} is too large. This is in agreement with the behavior observed in the previous work [1]: there exists a range of values of δ_{expl} for which the gain remains roughly constant.

We conclude this section by noticing that the gains obtained here are roughly similar to the gains reported in [1] for a Lennard–Jones cluster of 7 atoms in dimension two, although the system is here much more complex (in particular with a much larger dimensionality).

Remark 6. In the case of the strategy I, we have systematically observed that the choice $\delta_{\mathrm{expl}} = 0.4$ leads to a gain which is smaller than one. For some parameter choices (e.g. $\delta t = 0.5$ fs, $\delta_{\mathrm{expl}} = 0.4$ and $\delta_{\mathrm{conv}} = 10^{-10}$), we decided to stop the computations when it was obvious that the gain would be smaller than one, hence the void entries in Tables 3 and 4.

Still in the case of strategy I, we also note that the choice $\delta_{\mathrm{expl}} = 0.45$ (not reported in Table 3) may lead to unstable simulations. The system then explores regions of the phase space that are so unexpected (and so unphysical) that LAMMPS stops, declaring a `lost atom`.

Table 3. Wall-clock gain obtained using the adaptive parareal algorithm and strategy I (we use $V_{\mathscr{C}}^{\text{SNAP}-6}$ and $V_{\mathscr{F}}^{\text{SNAP}-205}$) on trajectories of length $N = 2000$. We have marked in bold the best results.

δt (in fs)	δ_{expl}	δ_{conv}	$\Gamma_{\text{adapt}}^{\text{ideal}}$	Γ_{adapt}	N_{slab}
0.5	0.15	10^{-10}	6.76	3.85	38
0.5	0.20	10^{-10}	7.75	4.63	26
0.5	0.25	10^{-10}	8.97	5.62	17
0.5	0.30	10^{-10}	10.27	**8.06**	11
0.5	0.35	10^{-10}	**11.63**	3.5	5
0.5	0.40	10^{-10}		<1	
0.5	0.15	10^{-5}	9.22	4.56	36
0.5	0.20	10^{-5}	10.7	5.51	27
0.5	0.25	10^{-5}	11.24	**6.05**	17
0.5	0.30	10^{-5}	10.99	4.44	9
0.5	0.35	10^{-5}	**12.35**	3.77	4
0.5	0.40	10^{-5}		<1	
0.5	0.15	10^{-3}	10.53	4.91	37
0.5	0.20	10^{-3}	10.99	5.39	26
0.5	0.25	10^{-3}	**13.25**	7.1	17
0.5	0.30	10^{-3}	13.07	**9.08**	9
0.5	0.35	10^{-3}	12.35	4.03	6
0.5	0.40	10^{-3}		<1	
2	0.15	10^{-10}	1.92	1.06	147
2	0.20	10^{-10}	2.24	1.75	102
2	0.25	10^{-10}	2.54	1.38	71
2	0.30	10^{-10}	2.93	2.26	42
2	0.35	10^{-10}	**3.32**	**2.49**	19
2	0.40	10^{-10}	1.66	0.55	5
2	0.15	10^{-5}	2.84	1.34	145
2	0.20	10^{-5}	3.07	1.67	100
2	0.25	10^{-5}	3.32	2.1	68
2	0.30	10^{-5}	3.51	2.55	39
2	0.35	10^{-5}	**3.67**	**2.89**	22
2	0.40	10^{-5}	2.21	0.81	7
2	0.15	10^{-3}	3.13	1.32	140
2	0.20	10^{-3}	3.33	1.76	102
2	0.25	10^{-3}	3.5	2.17	64
2	0.30	10^{-3}	3.54	2.55	41
2	0.35	10^{-3}	**3.68**	**2.95**	24
2	0.40	10^{-3}	2.4	0.9	9

Table 4. Wall-clock gain obtained using the adaptive parareal algorithm and strategy II (we use $V_{\mathscr{C}}^{\text{EAM}}$ and $V_{\mathscr{F}}^{\text{SNAP}-205}$) on trajectories of length $N = 2000$. We have marked in bold the best results.

δt (in fs)	δ_{expl}	δ_{conv}	$\Gamma_{\text{adapt}}^{\text{ideal}}$	Γ_{adapt}	N_{slab}
0.5	0.15	10^{-10}	9.9	9.21	23
0.5	0.20	10^{-10}	12.05	11.48	14
0.5	0.25	10^{-10}	13.33	12.88	9
0.5	0.30	10^{-10}	15.75	**15.31**	5
0.5	0.35	10^{-10}	**16.39**	14.64	2
0.5	0.40	10^{-10}		<1	
0.5	0.15	10^{-5}	14.93	13.99	22
0.5	0.20	10^{-5}	16.26	15.42	13
0.5	0.25	10^{-5}	17.24	16.77	9
0.5	0.30	10^{-5}	17.7	16.59	4
0.5	0.35	10^{-5}	**18.52**	**17.09**	2
0.5	0.40	10^{-5}		<1	
0.5	0.15	10^{-3}	16.13	14.88	22
0.5	0.20	10^{-3}	16.95	15.78	15
0.5	0.25	10^{-3}	17.54	16.81	9
0.5	0.30	10^{-3}	**20.0**	**19.05**	5
0.5	0.35	10^{-3}	17.7	15.84	2
0.5	0.40	10^{-3}		<1	
2	0.15	10^{-10}	2.71	2.58	89
2	0.20	10^{-10}	3.18	3.04	61
2	0.25	10^{-10}	3.7	3.56	40
2	0.30	10^{-10}	4.26	4.11	24
2	0.35	10^{-10}	**4.82**	**4.62**	13
2	0.40	10^{-10}	2.27	1.94	4
2	0.15	10^{-5}	4.39	4.06	87
2	0.20	10^{-5}	4.56	4.28	62
2	0.25	10^{-5}	4.9	4.66	41
2	0.30	10^{-5}	**5.28**	**5.06**	25
2	0.35	10^{-5}	5.22	4.98	14
2	0.40	10^{-5}	4.93	4.39	5
2	0.15	10^{-3}	4.73	4.34	90
2	0.20	10^{-3}	4.84	4.5	61
2	0.25	10^{-3}	5.06	4.8	38
2	0.30	10^{-3}	5.28	5.06	23
2	0.35	10^{-3}	**5.45**	**5.18**	13
2	0.40	10^{-3}	3.67	3.12	4

Appendix A. Kinetic temperature simulated in LAMMPS

We consider the scheme (2)–(3) with L time-steps, and assume that we repeat N times this loop. We show in Appendix A.1 that the equilibrium kinetic temperature K_{eq} provided by the scheme is different from the target value β^{-1}, and that it is significantly smaller than β^{-1} when L is small.

This motivates the introduction of the variant (11)–(12) (with the schedule (13) or (14)), that we show in Appendix A.2 to yield the correct equilibrium kinetic temperature. All the analytical derivations of this appendix are performed under the simplifying assumptions that the force field ∇V vanishes. The analytical conclusions are confirmed by numerical experiments performed on a realistic physical system modelled by a SNAP-56 potential energy.

A.1. *Stationary state of the kinetic temperature in the scheme* (2)–(3)

We consider the scheme (2)–(3) with a vanishing force field. We can assume without loss of generality that there is a single, one-dimensional particle: $(q, p) \in \mathbb{R} \times \mathbb{R}$. We define the kinetic temperature at step $\ell \in \{0, \dots, L\}$ as $K_\ell = \operatorname{Var} p_\ell$. We want to identify the equilibrium kinetic temperature K_{eq} of the scheme (2)–(3) with L steps. This is a value such that, if $\operatorname{Var} p_0 = K_{\text{eq}}$, then $\operatorname{Var} p_L = K_{\text{eq}}$. This is also, by ergodicity, the limit of $\operatorname{Var} p_{nL}$ when $n \to \infty$.

From the first and the third lines of (3), we compute that, for any $\ell \geq 1$,

$$
\begin{aligned}
p_{\ell+1/2} &= p_\ell - \frac{\delta t}{2} \gamma p_{\ell-1/2} + \frac{1}{2} \sqrt{2\gamma\beta^{-1}\delta t}\, G_\ell \\
&= \left[p_{\ell-1/2} - \frac{\delta t}{2} \gamma p_{\ell-1/2} + \frac{1}{2} \sqrt{2\gamma\beta^{-1}\delta t}\, G_\ell \right] - \frac{\delta t}{2} \gamma p_{\ell-1/2} + \frac{1}{2} \sqrt{2\gamma\beta^{-1}\delta t}\, G_\ell.
\end{aligned}
$$

We set

$$
\theta = \frac{1}{2} \gamma \beta^{-1} \delta t, \qquad \mu = 1 - \frac{1}{2} \gamma \delta t, \tag{15}
$$

and we therefore have, for any $\ell \geq 2$, that

$$
p_{\ell-1/2} = \left(1 - \gamma\delta t\right)^{\ell-1} p_{1/2} + 2\sqrt{\theta} \sum_{i=1}^{\ell-1} \left(1 - \gamma\delta t\right)^{i-1} G_{\ell-i}.
$$

Obviously, the above relation also holds for $\ell = 1$ (with the convention $\sum_{i=1}^{0} \cdot = 0$). Using the first line of (2), we thus deduce that, for any $\ell \geq 1$,

$$
p_{\ell-1/2} = \mu \left(1 - \gamma\delta t\right)^{\ell-1} p_0 + 2\sqrt{\theta} \sum_{i=1}^{\ell-1} \left(1 - \gamma\delta t\right)^{i-1} G_{\ell-i} + \sqrt{\theta} \left(1 - \gamma\delta t\right)^{\ell-1} G_0.
$$

Using now the last line of (2) and (3), we obtain that, for any $\ell \geq 1$,

$$
\begin{aligned}
p_\ell &= \mu p_{\ell-1/2} + \sqrt{\theta}\, G_\ell \\
&= \mu^2 \left(1 - \gamma\delta t\right)^{\ell-1} p_0 + 2\mu\sqrt{\theta} \sum_{i=1}^{\ell-1} \left(1 - \gamma\delta t\right)^{i-1} G_{\ell-i} + \mu\sqrt{\theta} \left(1 - \gamma\delta t\right)^{\ell-1} G_0 + \sqrt{\theta}\, G_\ell.
\end{aligned} \tag{16}
$$

Using that p_0 and all the G_j, $0 \leq j \leq \ell$, are independent and centered random variables and that $\operatorname{Var} G_j = 1$, we compute the expectation of p_ℓ^2 as

$$
\operatorname{Var} p_\ell = \mu^4 \left(1 - \gamma\delta t\right)^{2(\ell-1)} \operatorname{Var} p_0 + 4\mu^2\theta \sum_{i=1}^{\ell-1} \left(1 - \gamma\delta t\right)^{2(i-1)} + \mu^2\theta \left(1 - \gamma\delta t\right)^{2(\ell-1)} + \theta.
$$

Using that $\theta = O(\delta t)$, we have, at the leading order in the time-step, and for any $\ell \geq 1$, that

$$
\begin{aligned}
\operatorname{Var} p_\ell &= \left(1 - 2\ell\gamma\delta t\right) \operatorname{Var} p_0 + 4\theta(\ell - 1) + \theta + \theta + O\!\left(\delta t^2\right) \\
&= \left(1 - 2\ell\gamma\delta t\right) \operatorname{Var} p_0 + 4\theta\left(\ell - \frac{1}{2}\right) + O\!\left(\delta t^2\right).
\end{aligned} \tag{17}
$$

We are now in position to identify the equilibrium kinetic temperature K_{eq} of the scheme (2)–(3) with L steps. Indeed, inserting $\text{Var}\, p_L = \text{Var}\, p_0 = K_{eq}$ in (17), we obtain

$$2L\gamma\delta t K_{eq} = 4\theta\left(L - \frac{1}{2}\right) + O(\delta t^2) = 2\gamma\beta^{-1}\delta t\left(L - \frac{1}{2}\right) + O(\delta t^2),$$

and therefore

$$K_{eq} = \beta^{-1}\left(1 - \frac{1}{2L}\right) + O(\delta t). \tag{18}$$

We see that K_{eq} is always smaller than the target value β^{-1}, and that the difference is significant if L is small (think again of our choice $L = 1$, for which $K_{eq} = \beta^{-1}/2 + O(\delta t)$). At the intermediate stages, that is for p_ℓ with $1 \leq \ell \leq L-1$, the result is not better: inserting (18) in (17), we see that, for any $\ell \in \{1, \ldots, L\}$,

$$\text{Var}\, p_\ell = \left(1 - 2\ell\gamma\delta t\right)K_{eq} + 4\theta\left(\ell - \frac{1}{2}\right) + O(\delta t^2) = K_{eq} + \gamma\delta t\beta^{-1}\left(\frac{\ell}{L} - 1\right) + O(\delta t^2), \tag{19}$$

which is close to K_{eq} and thus significantly different from β^{-1} for small L.

In order to confirm the predictions of the above calculations in a more general setting (higher dimension and non-zero force field), we now turn to numerical experiments, performed using the SNAP-56 potential energy, for the same system as in Section 3 (128 tungsten atoms on a BCC lattice, this time without any defect). We consider the scheme (2)–(3) with L time-steps of length δt, and we iterate it N times, in order to reach the final time $N\Delta t$. The numerically computed values of the equilibrium kinetic temperature K_{eq} are shown in Table 5 for several choices of L, N and β.

The first two lines of Table 5 show that K_{eq} is indeed of the order of $\beta^{-1}/2$ when $L = 1$, as predicted by (18). If we set $L = 10$ and $\beta^{-1} = 300$, we expect from (18) to find $K_{eq} = 285$. A first simulation with $N = 20{,}000$ yields $K_{eq} \approx 280$ (see third line). The discrepancy with the theoretically predicted result decreases if N is increased to $N = 200{,}000$, as shown on the fourth line (we then expect to be closer to the ergodic limit). Finally, when $L = 100$ (fifth line), the difference between the computed equilibrium kinetic temperature and its target β^{-1} is negligible.

Table 5. Equilibrium kinetic temperature obtained using the scheme (2)–(3), for different values of L, N and target temperature β^{-1} (SNAP-56 potential energy, time-step $\delta t = 0.5$ fs, damping coefficient $\gamma^{-1} = 1$ ps).

$N \times L$	N	L	β^{-1}	K_{eq}
20,000	20,000	1	300	156.57
20,000	20,000	1	600	303.47
200,000	20,000	10	300	280.06
2,000,000	200,000	10	300	287.56
2,000,000	20,000	100	300	303.46

Remark 7. We have defined the numerical equilibrium kinetic temperature as the empirical variance of $\{p_{n,\ell}\}_{0 \leq n \leq N, 0 < \ell \leq L}$. We could alternatively have defined it as the empirical variance of $\{p_{n,L}\}_{0 \leq n \leq N}$. In all the test cases considered in Table 5, the first order correction in the right hand side of (19) satisfies

$$\left|\gamma\delta t\beta^{-1}\left(\frac{\ell}{L} - 1\right)\right| \leq 0.3,$$

and is thus negligible in comparison to the leading order term $K_{eq} = \beta^{-1}(1 - \frac{1}{2L})$ of (19). We thus expect the variances of $\{p_{n,\ell}\}_{0 \leq n \leq N, 0 < \ell \leq L}$ and of $\{p_{n,L}\}_{0 \leq n \leq N}$ to be close. The analytical result (18) corresponds to the theoretical variance of $\{p_{n,L}\}_{0 \leq n \leq N}$.

A.2. *Correction procedure*

In order to ensure that the kinetic temperature at every time-step of our integrator remains fixed to β^{-1}, we have introduced in Section 2.4 the variant (11)–(12) of (2)–(3). We are now going to show how to choose the temperature schedule in order to guarantee that the equilibrium kinetic temperature is indeed $K_{\mathrm{eq}} = \beta^{-1}$ at all the steps $\ell \in \{0, \ldots, L\}$.

Instead of working with β_ℓ, we work with the correction constants C_ℓ defined by

$$\beta_\ell^{-1} = C_\ell \, \beta^{-1}.$$

Similarly to (15), we set

$$\theta_\ell = \frac{1}{2} \gamma \beta_\ell^{-1} \delta t,$$

and similarly to (16), we have, for any $\ell \geq 1$, that

$$p_\ell = \mu^2 \left(1 - \gamma \delta t\right)^{\ell-1} p_0 + 2\mu \sum_{i=1}^{\ell-1} \left(1 - \gamma \delta t\right)^{i-1} \sqrt{\theta_{\ell-i}} \, G_{\ell-i} + \mu \sqrt{\theta_0} \left(1 - \gamma \delta t\right)^{\ell-1} G_0 + \sqrt{\theta_\ell} \, G_\ell.$$

We next compute the expectation of p_ℓ^2 as

$$\operatorname{Var} p_\ell = \mu^4 \left(1 - \gamma \delta t\right)^{2(\ell-1)} \operatorname{Var} p_0 + 4\mu^2 \sum_{i=1}^{\ell-1} \left(1 - \gamma \delta t\right)^{2(i-1)} \theta_{\ell-i} + \mu^2 \theta_0 \left(1 - \gamma \delta t\right)^{2(\ell-1)} + \theta_\ell.$$

Using that $\theta_j = O(\delta t)$ for any j, we have, at the leading order in the time-step, and for any $\ell \geq 1$, that

$$\operatorname{Var} p_\ell = \left(1 - 2\ell\gamma\delta t\right) \operatorname{Var} p_0 + 4 \sum_{i=1}^{\ell-1} \theta_i + \theta_0 + \theta_\ell + O\left(\delta t^2\right). \tag{20}$$

We now wish to choose $C_0, C_1, \ldots, C_L$ such that, if $\operatorname{Var} p_0 = \beta^{-1}$, then $\operatorname{Var} p_\ell = \beta^{-1}$ for any $1 \leq \ell \leq L$. We thus have $L+1$ unknowns for L equations.

Setting $\ell = 1$ in (20) and imposing $\operatorname{Var} p_0 = \operatorname{Var} p_1 = \beta^{-1}$ there, we get

$$2\gamma\delta t\beta^{-1} = \theta_0 + \theta_1 + O\left(\delta t^2\right) = \frac{1}{2}\gamma\beta^{-1}\delta t \, (C_0 + C_1) + O\left(\delta t^2\right),$$

which leads to enforcing

$$C_1 = 4 - C_0. \tag{21}$$

We next infer from (20) that, for any $\ell \geq 2$,

$$\operatorname{Var} p_\ell = \operatorname{Var} p_{\ell-1} - 2\gamma\delta t \operatorname{Var} p_0 + 4\theta_{\ell-1} + \theta_\ell - \theta_{\ell-1} + O\left(\delta t^2\right).$$

Imposing there that $\operatorname{Var} p_0 = \operatorname{Var} p_{\ell-1} = \operatorname{Var} p_\ell = \beta^{-1}$ leads to

$$2\gamma\delta t\beta^{-1} = 3\theta_{\ell-1} + \theta_\ell + O\left(\delta t^2\right) = \frac{1}{2}\gamma\beta^{-1}\delta t \, (3\,C_{\ell-1} + C_\ell) + O\left(\delta t^2\right),$$

which leads to enforcing

$$C_\ell = 4 - 3\,C_{\ell-1} \quad \text{for any } \ell \geq 2. \tag{22}$$

Collecting (21) and (22), we obtain

$$C_\ell = 1 - (-3)^\ell - (-3)^{\ell-1} C_0 \quad \text{for any } 1 \leq \ell \leq L,$$

from which we infer the simpler expression

$$C_\ell = 1 + 3^{\ell-1}(C_0 - 3) \text{ if } \ell \text{ is even}, \qquad C_\ell = 1 + 3^{\ell-1}(3 - C_0) \text{ if } \ell \text{ is odd}.$$

We now observe that not all choices of C_0 are admissible choices, since we have to ensure that $C_\ell > 0$ for any $0 \leq \ell \leq L$. If ℓ can take arbitrary large values, then the only possible choice is

$$C_0 = 3, \qquad C_\ell = 1 \text{ for any } \ell \geq 1. \tag{23}$$

This is the only choice which is robust with respect to L. In contrast, if L is fixed beforehand, several choices are possible. For instance, in the case $L = 1$, in addition to the choice $C_0 = 3$ and $C_1 = 1$, another possible choice is $C_0 = C_1 = 2$.

Remark 8. We have proceeded as follows to implement the scheme (11)–(12) in LAMMPS. In LAMMPS, the temperature schedule is defined in terms of the total amount of time (here, $n\Delta t + \ell\,\delta t$ for some n and ℓ) elapsed since the initial time $t = 0$. In contrast, in our scheme, the effective temperature β_ℓ^{-1} depends on ℓ but is independent of n. Our implementation relies on using the command `reset_timestep` of LAMMPS, which allows to reset the simulation clock to zero and that we call whenever $t = n\Delta t$ for some n. Thus, LAMMPS always makes use of the temperature at time 0, namely β_0^{-1}, when considering the first line of (11) to advance from time $n\Delta t$ to time $n\Delta t + \delta t$.

In order to check that the temperature schedules derived above indeed enforce the correct temperature in the system, we now turn to numerical experiments, which are performed with the same physical system as in Appendix A.1. We consider the scheme (11)–(12) with L time-steps of length δt, and we iterate it N times to reach the final time $N\Delta t$. The numerically computed value of the equilibrium kinetic temperature K_{eq} is shown in Table 6 for several choices of L, N and correction procedure.

In the first two lines of Table 6, we set $L = 1$ and consider two possible correction procedures, $C_0 = C_1 = 2$ in the first line and $C_0 = 3$, $C_1 = 1$ in the second line. We observe that both procedures lead to a numerical equilibrium kinetic temperature very close to its target (compare with the first line of Table 5). For larger values of L (here, $L = 10$; we have considered two values of N for the sake of comparison with Table 5), we only consider the robust choice (23), which leads to excellent results.

Table 6. Equilibrium kinetic temperature obtained using the corrected scheme (11)–(12), for different values of L and N and different correction procedures (SNAP-56 potential energy, target temperature $\beta^{-1} = 300$ K, time-step $\delta t = 0.5$ fs, damping coefficient $\gamma^{-1} = 1$ ps).

$N \times L$	N	L	correction procedure	K_{eq}
20,000	20,000	1	$C_0 = C_1 = 2$	303.38
20,000	20,000	1	$C_0 = 3$, $C_1 = 1$	296.87
200,000	20,000	10	$C_0 = 3$, $C_{[1,\dots,10]} = 1$	303.12
2,000,000	200,000	10	$C_0 = 3$, $C_{[1,\dots,10]} = 1$	303.85

Declaration of interests

Acknowledgments

References

[1] F. Legoll, T. Lelièvre, U. Sharma, "An adaptive parareal algorithm: application to the simulation of molecular dynamics trajectories", *SIAM J. Sci. Comput.* **44** (2022), no. 1, p. B146-B176.

[2] T. Lelièvre, M. Rousset, G. Stoltz, *Free Energy Computations. A mathematical perspective*, Imperial College Press, 2010.

[3] B. P. Uberuaga, D. Perez, "Computational methods for long-timescale atomistic simulations", in *Handbook of Materials Modeling: Method: Theory and Modeling* (W. Andreoni, S. Yip, eds.), Springer, 2020, p. 683-688.

[4] R. J. Zamora, D. Perez, E. Martinez, B. P. Uberuaga, A. F. Voter, "Accelerated molecular dynamics methods in a massively parallel world", in *Handbook of Materials Modeling: Methods: Theory and Modeling* (W. Andreoni, S. Yip, eds.), Springer, 2020, p. 745-772.

[5] J.-L. Lions, Y. Maday, G. Turinici, "Résolution d'EDP par un schéma en temps pararéel (A "parareal" in time discretization of PDE's)", *C. R. Acad. Sci. Paris Sér. I Math.* **332** (2001), no. 7, p. 661-668.

[6] F. Legoll, T. Lelièvre, G. Samaey, "A micro-macro parareal algorithm: application to singularly perturbed ordinary differential equations", *SIAM J. Sci. Comput.* **35** (2013), no. 4, p. A1951-A1986.

[7] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. R. Trott, S. J. Plimpton, "LAMMPS – a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales", *Comput. Phys. Commun.* **271** (2022), article no. 108171.

[8] A. P. Thompson, L. P. Swiler, C. R. Trott, S. M. Foiles, G. J. Tucker, "Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials", *J. Comput. Phys.* **285** (2015), p. 316-330.

[9] M. S. Daw, M. I. Baskes, "Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals", *Phys. Rev. B* **29** (1984), no. 12, p. 6443-6453.

[10] A. Brünger, C. L. Brooks III, M. Karplus, "Stochastic boundary conditions for molecular dynamics simulations of ST2 water", *Chem. Phys. Lett.* **105** (1984), no. 5, p. 495-500.

[11] G. Bal, Y. Maday, "A parareal time discretization for nonlinear PDE's with application to the pricing of an American put", in *Recent developments in domain decomposition methods* (L. F. Pavarino, A. Toselli, eds.), Lecture Notes in Computational Science and Engineering, vol. 23, Springer, 2002, p. 189-202.

[12] M. J. Gander, S. Vandewalle, "Analysis of the parareal time-parallel time-integration method", *SIAM J. Sci. Comput.* **29** (2007), p. 556-578.

[13] M. J. Gander, T. Lunet, D. Ruprecht, R. Speck, "A unified analysis framework for iterative parallel-in-time algorithms", *SIAM J. Sci. Comput.* **45** (2023), no. 5, p. A2275-A2303.

[14] M. J. Gander, "50 years of time parallel time integration", in *Multiple Shooting and Time Domain Decomposition Methods* (T. Carraro, M. Geiger, S. Körkel, R. Rannacher, eds.), Contributions in Mathematical and Computational Sciences, vol. 9, Springer, 2015, p. 69-114.

[15] A. Blouza, L. Boudin, S.-M. Kaber, "Parallel in time algorithms with reduction methods for solving chemical kinetics", *Commun. Appl. Math. Comput. Sci.* **5** (2010), no. 2, p. 241-263.

[16] Y. Maday, "Parareal in time algorithm for kinetic systems based on model reduction", in *High-dimensional partial differential equations in science and engineering* (A. Bandrauk, M. C. Delfour, C. Le Bris, eds.), CRM Proceedings & Lecture Notes, vol. 41, American Mathematical Society, 2007, p. 183-194.

[17] S. Engblom, "Parallel in time simulation of multiscale stochastic chemical kinetics", *Multiscale Model. Simul.* **8** (2009), p. 46-68.

[18] X. Dai, C. Le Bris, F. Legoll, Y. Maday, "Symmetric parareal algorithms for Hamiltonian systems", *ESAIM, Math. Model. Numer. Anal.* **47** (2013), no. 3, p. 717-742.

[19] X. Dai, Y. Maday, "Stable parareal in time method for first- and second-order hyperbolic systems", *SIAM J. Sci. Comput.* **35** (2013), no. 1, p. A52-A78.

[20] G. Bal, "Parallelization in time of (stochastic) ordinary differential equations", Preprint available at https://www.stat.uchicago.edu/~guillaumebal/PAPERS/paralleltime.pdf.

[21] G. Pagès, O. Pironneau, G. Sall, "The parareal algorithm for American options", *C. R. Acad. Sci. Paris Sér. I Math.* **354** (2016), no. 11, p. 1132-1138.

[22] F. Legoll, T. Lelièvre, K. Myerscough, G. Samaey, "Parareal computation of stochastic differential equations with time-scale separation: a numerical convergence study", *Comput. Vis. Sci.* **23** (2020), article no. 9.

[23] I. Garrido, M. Espedal, G. Fladmark, "A convergent algorithm for time parallelization applied to reservoir simulation", in *Domain decomposition methods in science and engineering* (R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, J. Xu, eds.), Lecture Notes in Computational Science and Engineering, vol. 40, Springer, 2005, p. 469-476.

[24] C. Farhat, M. Chandesris, "Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid–structure applications", *Int. J. Numer. Methods Eng.* **58** (2003), no. 9, p. 1397-1434.

[25] M. Gaja, O. Gorynina, "Parallel in time algorithms for nonlinear iterative methods", *ESAIM, Proc. Surv.* **63** (2018), p. 248-257.

[26] Y. Maday, O. Mula, "An adaptive parareal algorithm", *J. Comput. Appl. Math.* **377** (2020), article no. 112915.

[27] P. L'Ecuyer, D. Munger, B. Oreshkin, R. Simard, "Random numbers for parallel computers: Requirements and methods, with emphasis on GPUs", *Math. Comput. Simul.* **135** (2017), p. 3-17.

[28] A. Stukowski, "Visualization and analysis of atomistic simulation data with OVITO – the Open Visualization Tool", *Model. Simul. Mat. Sci. Eng.* **18** (2010), no. 1, article no. 015012.