PHYSICAL REVIEW B VOLUME 57, NUMBER 22 1 JUNE 1998-II

Parallel replica method for dynamics of infrequent events

Arthur F. Voter
Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545
(Received 8 January 1998)

Although molecular-dynamics simulations can be parallelized effectively to treat large systems (10^6–10^8 atoms), to date the power of parallel computers has not been harnessed to make analogous gains in *time* scale. I present a simple approach for infrequent-event systems that extends the time scale with high parallel efficiency. Integrating a replica of the system independently on each processor until the first transition occurs gives the correct transition-time distribution, and hence the correct dynamics. I obtain $>90\%$ efficiency simulating Cu(100) surface vacancy diffusion on 15 processors. [S0163-1829(98)51420-8]

With the increasing prominence and availability of multiprocessor computers, recasting problems in a form amenable to parallel solution is becoming a critical step in effective scientific computation. For dynamical systems, assigning a region of physical space to each processor is an efficient way to extend the accessible size scale. Using this approach, molecular-dynamics (MD) simulations as large as 10^6-10^8 atoms are now practical.[1,2] Unfortunately, using the same algorithm on a small system (e.g., $<10^3$ atoms), with the goal of applying the parallel power to the time scale, is inefficient because the algorithm becomes communication bound.

For some classical systems, the long-time dynamics are characterized by extended residence times in a potential basin, with an occasional transition to a new basin. These ''infrequent-event'' processes are common in many fields of current interest. Examples include diffusion and reorganization processes on a surface during film growth, vacancy diffusion at a grain boundary, annealing of a damaged region of crystal after ion-implantation, and diffusion of an absorbate through a zeolite.

A feature of infrequent-event systems is the separability of the two characteristic time scales: the brief duration of a transition event and the long waiting time between events. Direct MD integration of the long waiting period is often unnecessary because transition-state theory (TST) can be employed to compute the rate constant directly, provided the dividing surface for the reaction is known. Sometimes, however, the available reaction mechanisms are not easily determined due to the complexity of the system. In these cases, it would be extremely useful to investigate the system behavior with a direct simulation method such as MD, as it requires no advanced knowledge of the available pathways. However, MD is currently limited to nanoseconds. It is thus of great interest to develop ways to extend the MD time scale, and research efforts along these lines are underway.[3,4]

The purpose of this paper is to show that the properties of an infrequent-event system can be exploited in a different way to develop an efficient parallel approach to the dynamics. For a system in which successive transitions are uncorrelated (the usual case for diffusive processes in materials), running a number of independent trajectories in parallel gives the exact dynamical evolution from state to state. For a system with correlated crossing events, the state-to-state sequence is still correct, and the associated time scale has small, controllable errors. With this approach, the power of parallel processing can be applied to make substantial extensions in the MD time scale for small infrequent-event systems. For example, in the Cu(100) simulation discussed below, a 15-processor implementation gives a 14-fold increase in simulation time per wall-clock time. Moreover, this method can be combined with other methods for extending the MD time scale, such as the recently presented hyperdynamics method,[4] giving a multiplicative effect in the time scale gain. I present a derivation and demonstration for an atomistic system, but the generality of the approach should make it useful in a variety of applications.

Consider a classical, canonical system of N atoms vibrating in a basin of the $3N$-dimensional potential energy surface. It is assumed that the dynamical exploration of this basin is ergodic. Available to this system are a number (n_{esc}) of possible escape routes, each corresponding to a section of the total dividing surface bounding this state. Eventually the trajectory finds a point on the dividing surface and passes through it to another state. Following this initial crossing (the primary event), there is a period of time (τ_{corr}) during which the system remembers how it entered the new state, and there may be dynamically correlated surface crossings that return the system to the original state or send it on to another state.[5,6] τ_{corr} is system dependent; for processes of primary interest here, such as bulk or surface diffusion, it is typically a few Einstein vibrational periods ($\sim$1 ps). At times greater than τ_{corr} after the primary event, by definition, the trajectory has no memory of how it arrived in the new state. The probability of finding a particular escape pathway from the new state is now unbiased, in the sense that it depends only on the properties of the dividing surface for that pathway. Because the average time until the next escape is much greater than the system memory time (τ_{corr}), many independent attempts are made to find an escape path during the ergodic exploration of the basin. The success probability per unit time is thus a constant, creating a first-order process. Defining k_{tot} as the total rate constant for finding the next escape path from this state, the probability distribution for the waiting time before the next primary crossing event is given by[7]

$$p(t)=k_{tot}\exp(-k_{tot}t). \tag{1}$$

In a system that exhibits no correlated crossing events, k_{tot} is exactly the TST rate constant (k^{TST}). k^{TST} is defined as the outgoing flux through the total dividing surface, an equilibrium property of the canonical system.[8,9] In the more general case, in which correlated crossings occur, $k_{tot}<k^{TST}$, because some phase-space points on the dividing surface are unavailable for a primary crossing event; the system passes through those points only during the subsequent correlated crossing events.[9,5] Either way, k_{tot} is a well-defined quantity, which (if desired) can be calculated exactly if the whole dividing surface is known.[5]

The dynamical evolution of the system can be summarized in the following way. Assuming that at least τ_{corr} has passed since the trajectory entered the present state, Eq. (1) gives the exact waiting time distribution until the next escape crossing occurs. When it does occur, during a transient time τ_{corr} from the primary event, the trajectory may execute additional state-to-state transitions correlated with the primary crossing, after which it thermalizes in some state (perhaps the original one), and the cycle begins again with the value of k_{tot} for the new state. I now derive the parallel-replica method, which is surprisingly simple.

I first assume that when a transition occurs, it can be detected. This may not be possible for all systems, but is required for implementing this method. One approach is to interrupt the MD simulation periodically to perform a steepest-descent or conjugate-gradient minimization, leading the system towards the minimum of its current potential basin. Even with a partially converged minimization, comparison of the geometry to that of a previous minimization will signal when a new basin has been entered. This requires no advanced knowledge of the nature of the transition. By choosing the number of MD steps between quench interruptions much larger than the number of steepest-descent steps, this transition monitoring need not excessively slow the simulation.

Now consider simulating M replicas of this same system on M different processors. Each replica starts in the same state, but with a different initial condition for the trajectory, so that the replicas are statistically independent. Assume for the moment that all M processors are equivalent and run at the same speed. Define S as the summed speed of all the processors, relative to the speed of processor number 1; $S=M$ in the present case. The key point is that this set of M replicas acts the same as a supersystem with Mn_{esc} escape paths; i.e., it is equivalent to a physical system where M replicas have been placed side by side. Modifying Eq. (1) to account for the increased number of escape paths and the new total escape rate (Sk_{tot}) gives the escape-time probability distribution for this supersystem as a function of t_1, the trajectory time on processor number 1,

$$p_{super}(t_1)=Sk_{tot}\exp(-Sk_{tot}t_1). \tag{2}$$

This is the probability distribution for the time until the next primary event occurs on *any* of the processors. At a given point in time, the accumulated simulation time summed over all the replica trajectories is related to t_1 by

$$t_{sum}=St_1. \tag{3}$$

Inserting this relation into Eq. (2), and recognizing that $(1/S)p(t/S)dt=p(t)dt$ for any probability distribution p, gives

$$p_{super}(t_{sum})=k_{tot}\exp(-k_{tot}t_{sum}). \tag{4}$$

Comparison of Eq. (4) with Eq. (1) shows that running M independent replica simulations and defining the time (t_{sum}) as in Eq. (3) *results in the correct probability distribution for the escape time from this state*. After a transition occurs, the value for k_{tot} changes,[10] but again (after integrating for a period of at least τ_{corr} on a single processor) the parallel trajectories give the correct waiting time probability distribution for the new state. Moreover, the parallelization has no effect on the relative probabilities of the different possible escape paths. Thus, if the parallel-replica trajectories are monitored continuously for transitions, *both the sequence of states and the transition times in this parallel simulation are indistinguishable from a simulation on a single processor*. This is true even if the processors run at different, time-varying speeds, as shown below. Using intermittent transition checks (rather than continuous monitoring) introduces controllable errors in the transition times if there are recrossing events. This is also discussed below.

The steps in the parallel-replica simulation procedure are as follows: (1) The current configuration of the system is replicated on M processors. (2) A minimization is performed to generate a reference configuration for transition checks. (3) On each processor, after a momentum randomization stage to eliminate correlations with other replicas, a classical trajectory is integrated. A thermostat is used to control the temperature. (4) Each replica trajectory is monitored for a transition event by performing a quench after each Δt_{block} of integration time. When one processor (i) detects an event, all processors are notified to stop. The time of this primary transition can be refined to arbitrary precision if desired. (5) The simulation clock is advanced by t_{sum}, the sum of the trajectory times accumulated by all M replicas since the beginning of step (3). (6) On one processor, replica i is integrated forward for a prechosen time ($\Delta t_{corr}\geqslant\tau_{corr}$), during which new transitions may occur. The simulation clock is advanced by Δt_{corr}. (7) Replica i becomes the new configuration of the system. (8) Go to step (1).

In this procedure, an exact mapping between the system configuration and the simulation clock can only be made at the time of each transition. Information on finer time scales, e.g., regarding vibrational behavior, can be obtained from any of the individual replica simulations. The procedure is efficiently parallel if the typical escape time is much larger than $\Delta t_{corr}S$ and if the wall-clock time between transitions is much greater than the communication time required in steps (1) and (4) when a transition occurs. Between transitions, no interprocessor communication is required. An appealing feature is that if Δt_{corr} is chosen too conservatively (larger than necessary), the dynamics are still correct, because Eq. (4) is valid for the remaining time before the next escape, regardless of when the parallelization begins.

I now consider the effect of the intermittent transition checks (noninfinitessimal Δt_{block}). For a TST-obeying system, the dynamics are still exact provided that when a transition is detected, the most recent integration block is reana-

lyzed to pinpoint the time of transition. If errors of the order of $\Delta t_{block}/2$ are acceptable, the transition time can be taken as halfway through the most recent integration block. Provided that $\tau_{esc} \gg \Delta t_{block}$, this gives an error whose average is negligible. For systems with correlated events, there is one additional source of error in the transition times. A recrossing event (in which the trajectory quickly reenters the state it just exited) that occurs within one integration block will be invisible to the transition monitoring. Ideally, at the instant of the first escape (on processor i), the other M-1 trajectories should be stopped while the correlated event takes place. When the trajectory reenters the state at a time $\Delta t_{recross}$ later, the other M-1 trajectories can be restarted immediately without biasing the dynamics, assuming that this is the end of the correlated sequence. Each hidden recrossing thus causes t_{sum} to accumulate $(S-1)\Delta t_{recross}$ more time than it should. This is a controllable error. The larger $\Delta t_{recross}$ is, the more often the recrossing event will cross over an integration block boundary rather than being hidden. By counting these events and measuring their durations, an estimate of the number and effect of the hidden recrossings can be easily made.

In general, the processor speeds may be inequivalent, having speeds relative to processor number one given by $\{s_i;\ i=1,2,\ldots,M;\ s_1=1\}$, and a summed speed of

$$S=\sum_i^M s_i . \tag{5}$$

The supersystem now consists of a set of M systems where the rates within each system i have been scaled by s_i. Because the total escape rate is given by $\Sigma_i^M s_i k_{tot}=Sk_{tot}$, the proof that the parallel simulation gives the correct waiting time distribution proceeds just as in Eqs. (2) through (4), using the new definition for S in Eq. (5).

Finally, I consider the case in which the processor speeds are not constant in time. Due to the characteristics of a first-order process, at any point in time (t_p) at which a transition has not yet occurred, the future probability distribution for the waiting time measured from t_p is the same normalized exponential function [Eq. (4)] as when t was measured from $t=0$. If the processor speeds $\{s_i\}$ change abruptly at $t=t_p$, the (exact same) probability distribution for the parallel trajectories can be rederived using the new set of speeds. As this hypothetical rederivation can be applied as often as necessary, the parallel-replica method is valid for arbitrary fluctuations in the individual processor speeds.

As a demonstration, I apply the parallel-replica method to the diffusion of a surface vacancy on the Cu(100) surface at $T=500$ K. Although the real power of this method lies in applications to systems where the pathways are numerous and unanticipated (and preliminary tests indicate it works well in those situations), proving the dynamics are correct requires a simple system. This case was intentionally constructed to have only one reaction pathway, so that the escape-time probability distribution could be computed and compared to the exact result. The copper interaction was described using an embedded atom method (EAM)[11] interatomic potential, fit following the procedure described in Ref. 12, with $a_0=3.615$ Å, $E_{coh}=3.54$ eV, and $B=1.419\times10^{12}$ erg/cm^3, resulting in the parameters

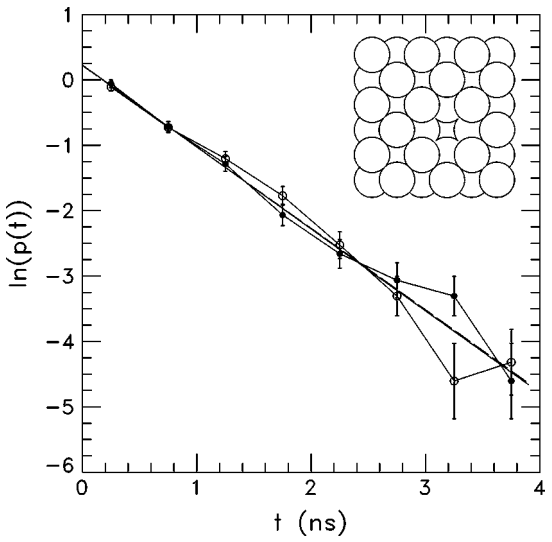

FIG. 1. Escape-time probability distribution for surface vacancy diffusion on Cu(100), obtained from a single trajectory (open circles, dashed line) and from the parallel-replica method (filled circles, solid line). Error bars are one standard deviation and normalization is for time expressed in ns. The inset shows a surface-normal view of the simulation cell.

$D_M=0.7366$ eV, $R_M=2.325$ Å, $\alpha_M=1.919$ Å^{-1}, $\beta_M=4.043$ Å^{-1}, and $r_{cut}=4.961$ Å. The simulation cell (see Fig. 1 inset) consisted of five layers of atoms, 18 atoms per layer (17 in the top layer), with periodic boundary conditions parallel to the surface. The substrate-exchange pathway was suppressed by restricting the number of moving layers to two, allowing only hopping events. The equations of motion were integrated using a Langevin-Verlet algorithm[13] with a time step of 2×10^{-15} s and a Langevin coupling constant of 2×10^{11} s^{-1}. Both Δt_{block} and Δt_{corr} were set to 2×10^{-12} s. The parallelization was implemented on 16 processors using a message-passing interface. A master-slave configuration was chosen for these initial tests, giving $M=15$.

To decorrelate the replica trajectories, the momenta were randomized (by drawing from a $T=500$ K Maxwellian distribution) every 1.0 ps for the first 10 ps. Transitions were not prevented from occurring during this period, although none did. Alternatively, the randomization stage could be performed before $t_i=0$, with rejection of any transition attempts during this period. These two methods gave indistinguishable escape-time distribution functions.

Primary transition checks were performed using a fixed-ratio steepest descent (SD) minimization (one gradient evaluation per step). A transition was declared if, after 100 SD steps, any atom still deviated from the reference configuration obtained in step (2) by more than 1.0 Å. Each minimization was terminated after as few steps as possible. When a transition was detected, the transition time was taken to be halfway through the previous integration block. The wall-clock time at the instant assigned to the transition was communicated to the other processors (via the master) to allow calculation of their precise accumulated trajectory time at the time of the transition, nullifying any communication lag effect. (For very simple systems, such as a 2D model potential, this had a significant effect.) On the processor that detected the transition, the trajectory was continued for an additional

time Δt_{corr} (2 ps), after which this configuration was passed to the master processor and then distributed to all slave processors to start the procedure again.

Figure 1 shows the distribution of escape times (defined as the time between primary crossings) from 600 events in the parallel-replica simulation. Also shown is the distribution from a 600-event single-processor trajectory, performed using the same procedure, but with $M=1$ and no momentum randomizations. The two distributions agree, exhibiting the expected exponential behavior. The average escape times are also in excellent agreement (parallel: 0.798 ± 0.03 ns, single processor: 0.803 ± 0.03 ns), and were used to construct the lines in Fig. 1, by equating k_{tot} to the inverse of the average escape time.

Correlated events were analyzed by comparing the successive minimized reference geometries. Of the 600 transitions in the parallel simulation, there were 7 recrossings, 28 double displacements, and 2 more complicated events. The numbers for the single-processor trajectory were similar: 3, 31, and 1, respectively. These results are consistent with the correlated events observed in a set of half-trajectories that were performed to determine τ_{corr} for this system. There, 100 trajectories, each 10 ps in length, were initiated at the saddle-plane dividing surface. All correlated events ceased within 1.5 ps, and most were over within the first 0.75 ps.

Running the master process on one of the slave processors (reducing the total number of processors involved from 16 to 15) made no discernible difference in the simulation speed. The simulation achieved a parallel efficiency of 92%, relative to 15 times the speed of a single, uninterrupted trajectory. Using $\Delta t_{block}=6$ ps and lowering T to 450 K (decreasing k_{tot} by a factor of ~ 3.5) increased the efficiency to 96%. In general, the method becomes more efficient for systems with more complicated potential functions (reducing the relative time spent on communication) and less efficient as τ_{corr} increases or as the number of SD steps required to detect a transition increases.

In conclusion, for small infrequent-event systems the parallel-replica approach offers an efficient alternative to the standard algorithm for parallelizing MD simulations. Its generality and ease of implementation should make it useful in a variety of situations.

I am grateful to Antonio Redondo and Lawrence Pratt for stimulating and helpful discussions, and to Wolfgang Windl for a critical reading of the manuscript. Parallel computations were performed on the Loki machine (16 Intel Pentium Pro processors) in the Theoretical Division at Los Alamos; assistance from Michael Warren is gratefully acknowledged. This work was supported by the Department of Energy, Office of Basic Energy Sciences.

[1] S. J. Zhou, D. M. Beazley, P. S. Lomdahl, and B. L. Holian, Phys. Rev. Lett. **78**, 479 (1997).

[2] F. F. Abraham, IEEE Comput. Sci. Eng. **4**, 66 (1997).

[3] T. Schlick, E. Barth, and M. Mandziuk, Annu. Rev. Biophys. Biomol. Struct. **26**, 181 (1997), and references therein.

[4] A. F. Voter, J. Chem. Phys. **106**, 4665 (1997); Phys. Rev. Lett. **78**, 3908 (1997).

[5] A. F. Voter and J. D. Doll, J. Chem. Phys. **82**, 80 (1985).

[6] D. Chandler, J. Chem. Phys. **68**, 2959 (1978); J. A. Montgomery, Jr., D. Chandler, and B. J. Berne, *ibid.* **70**, 4056 (1979).

[7] D. T. Gillespie, J. Comp. Physiol. **22**, 403 (1976).

[8] R. Marcelin, Ann. Physique **3**, 120 (1915).

[9] J. B. Anderson, Adv. Chem. Phys. **91**, 381 (1995).

[10] Note that k_{tot} need not be calculated in the simulation.

[11] M. S. Daw and M. I. Baskes, Phys. Rev. B **29**, 6443 (1984).

[12] A. F. Voter, in *Intermetallic Compounds: Principles and Practice*, edited by J. H. Westbrook and R. L. Fleischer (John Wiley and Sons, Ltd., New York, 1995), Vol. 1, p. 77.

[13] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford, New York, 1987), p. 263, Eq. (9.24).