Parallel algorithm for spin and spin-lattice dynamics simulations

Pui-Wai Ma and C. H. Woo[*]

Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong, SAR, China

(Received 28 November 2008; published 6 April 2009)

To control numerical errors accumulated over tens of millions of time steps during the integration of a set of highly coupled equations of motion is not a trivial task. In this paper, we propose a parallel algorithm for spin dynamics and the newly developed spin-lattice dynamics simulation [P. W. Ma *et al.*, Phys. Rev. B **78**, 024434 (2008)]. The algorithm is successfully tested in both types of dynamic calculations involving a million spins. It shows good stability and numerical accuracy over millions of time steps ($\sim$1 ns). The scheme is based on the second-order Suzuki-Trotter decomposition (STD). The usage can avoid numerical energy dissipation despite the trajectory and machine errors. The mathematical base of the symplecticity, for properly decomposed evolution operators, is presented. Due to the noncommutative nature of the spin in the present STD scheme, a unique parallel algorithm is needed. The efficiency and stability are tested. It can attain six to seven times speed up when eight threads are used. The run time per time step is linearly proportional to the system size.

DOI: 10.1103/PhysRevE.79.046703 PACS number(s): 05.10.−a, 02.60.Cb, 95.75.Pq, 75.10.−b

I. INTRODUCTION

Spin-lattice dynamics (SLD) simulation [1] has recently been shown to be a viable means of including magnetic effects in the atomistic study of properties of ferromagnetic materials. To control the accumulation of numerical errors to an acceptable level over tens of millions of time steps is essential but nontrivial. Indeed, Hatano and Suzuki [2] showed that perturbational integration schemes (e.g., the standard predictor-corrector method) that are commonly implemented in molecular dynamics (MD) or spin dynamics (SD) simulations did not have sufficient accuracy for energy conservation, particularly for long simulation times, due to the substantial accumulation of numerical errors. Omelyan *et al.* [3–5] and Tsai *et al.* [6,7] investigated the application of symplectic integration algorithms for the spin and the lattice degrees of freedom, respectively. Both were based on the second-order Suzuki-Trotter decomposition (STD) scheme. We note that the symplecticity (i.e., the ability to preserve the phase space volume which is the energy in the present case) of STD has only been mathematically proven for MD simulations [2]. To consider the case for SD and SLD is one of the objectives in the present paper.

In large-scale atomistic simulations that involve millions of atoms and tens of millions of time steps, experience with conventional MD has shown that a parallel algorithm [8–13] is essential. While parallel programming can be readily performed with STD when the spin system is not involved [13], its implementation in the spin system is not straightforward and has not been reported. The situation is even more difficult when the spin and the atom coordinates are coupled in a case like SLD. In this paper, the viability of a proposed algorithm that may be adopted in such cases is investigated. The designed algorithm has been tested in a computer system with share memory architecture (e.g., using OPENMP). It is found to be capable of achieving a gain in speed of six to seven times with eight threads, while the run time per time step still maintains its linearity with the system size.

This paper is to supplement our earlier paper [1] in which the SLD is proposed to provide a viable tool to investigate the physics of ferromagnetic materials, particularly near and during the ferromagnetic transition, via large-scale dynamical atomistic simulations. Due to limited space, details crucial to the viability of the scheme have not been discussed. What is considered in this paper is not just parallel programming nor STD but the method of doing parallel programming on the STD. This is particularly important within the context of SLD, in which the dynamics of both the spin and the lattice systems has to be integrated simultaneously. This is a crucial step in the practical implementation of the SLD scheme. Thus, if symplectic integration is not used, one may have to face a long list of serious problems related to nonconservation of energy through an integration scheme involving tens of millions of time steps. At the same time, without a parallel algorithm, large-scale SLD simulations are impracticable. Within our knowledge, no viable method capable of tackling this duel problem has been written in the literature.

II. SYMPLECTICITY

The SLD simulation is structured similar to a conventional MD simulation, but with the intrinsic (classical) spin degrees of freedom coupled to the lattice subsystem via the exchange function. The effective classical Hamiltonian is given by [1,14]

$$H = \sum_i \frac{\mathbf{p}_i^2}{2m_i} + U(\{\mathbf{R}_k\}) + \frac{1}{2}\sum_{i,j} J_{ij}(\{\mathbf{R}_k\})(1 - \mathbf{e}_i \cdot \mathbf{e}_j), \quad (1)$$

where the $\mathbf{e}_i$ is a unit vector of the atomic spin direction. U is the potential energy of the atomic system $\{\mathbf{R}_k\}$ with perfect spin collinearity, i.e., when $\mathbf{e}_i \cdot \mathbf{e}_j = 1$ for any pair of atoms i and j. J_{ij} is the exchange coupling function, which also depends on the atomic configuration. The last term in Eq. (1) accounts for the energy associated with a loss of collinearity, i.e., $\mathbf{e}_i \cdot \mathbf{e}_j < 1$. The corresponding Hamiltonian equations of

[*]Corresponding author. chung.woo@polyu.edu.hk

motion can be derived using Poisson brackets [1] and are given by

$$\frac{d\mathbf{R}_k}{dt} = \frac{\partial H}{\partial \mathbf{p}_k} = \frac{\mathbf{p}_k}{m_k},$$

$$\frac{d\mathbf{p}_k}{dt} = -\frac{\partial H}{\partial \mathbf{R}_k} = -\frac{\partial U}{\partial \mathbf{R}_k} + \frac{1}{2}\sum_{i,j}\frac{\partial J_{ij}}{\partial \mathbf{R}_k}(1 - \mathbf{e}_i \cdot \mathbf{e}_j),$$

$$\Pi_k \frac{d\mathbf{e}_k}{dt} = \mathbf{e}_k \times \mathbf{H}_k, \tag{2}$$

where $\mathbf{H}_k = \Sigma_i J_{ik}\mathbf{e}_i$ and $\Pi_k = \hbar S_k = \hbar M_k / g\mu_B$. The g factor is taken as a positive quantity. M_k is the magnitude of an atomic magnetic moment. $\mathbf{H}_k$ is the effective exchange vector field acting on spin k. Π_k plays the role of moment of inertia for the dynamics of angular motion of the spin vector. The above equation shows that the dynamics of the lattice and spin subsystems are explicitly coupled through the dependence of J_{ij} on the atomic positions $\{\mathbf{R}_k\}$ via the gradient term $\partial J_{ij}/\partial \mathbf{R}_k$ in the second equation in Eq. (2) and the J_{ij} term in the third equation in Eq. (2).

The set of SLD equation has to be solved simultaneously by time integration. A suitable algorithm is needed to enhance the overall accuracy and performance. Within a small time step Δt, we may assume that there is a Hamiltonian operator $\mathcal{H}$ that operates on the generalized coordinate $\mathbf{x}$, satisfying the equation of motion,

$$\frac{d\mathbf{x}}{dt} = \mathcal{H}\mathbf{x}, \tag{3}$$

where $\mathbf{x}$ is an element of the direct sum of the coordinate space R, momentum space P, and spin-direction space S, i.e., $\mathbf{x} \in \mathsf{R} \oplus \mathsf{P} \oplus \mathsf{S}$. Referring to Eq. (2), $\mathcal{H}$ can be decomposed into the sum of three operators, representing the atomic force $\mathcal{F}$, atomic momentum $\mathcal{P}$, and spin precession $\mathcal{S}$, defined as

$$\mathcal{F}\text{:}\mathsf{R} \oplus \mathsf{S} \to \mathsf{P} \text{ such that } \mathcal{F}\mathbf{x} \equiv -\frac{\partial U}{\partial \mathbf{R}} + \frac{1}{2}\sum_{i,j}\frac{\partial J_{ij}}{\partial \mathbf{R}}(1 - \mathbf{e}_i \cdot \mathbf{e}_j),$$

$$\mathcal{P}\text{:}\mathsf{P} \to \mathsf{R} \quad \text{such that } \mathcal{P}\mathbf{x} \equiv \frac{\mathbf{p}}{m},$$

$$\mathcal{S}\text{:}\mathsf{R} \oplus \mathsf{S} \to \mathsf{S} \quad \text{such that } \mathcal{S}\mathbf{x} \equiv \mathbf{e} \times \sum_i J_i \mathbf{e}_i. \tag{4}$$

In this equation, $\mathcal{F}$ relates an element in $\mathsf{R} \oplus \mathsf{S}$ to an element in P; its operation leaves the elements in R and S unchanged. Similarly, the operation of $\mathcal{P}$ relates an element in P to one in R, leaving the element in P unchanged. The operation of $\mathcal{S}$ is the *only* one that changes the elements in S which it operates on. The treatment of $\mathcal{S}$ has to be achieved in a self-consistent way and will be considered as we proceed. Using these three operators, the equations of motion [Eq. (2)] can be written as

$$\mathcal{H}\mathbf{x} \equiv (\mathcal{P} + \mathcal{F} + \mathcal{S})\mathbf{x} \tag{5}$$

and

$$\frac{d}{dt}\begin{pmatrix}\mathbf{R}\\ \mathbf{p}\\ \mathbf{e}\end{pmatrix} = (\mathcal{P} + \mathcal{F} + \mathcal{S})\mathbf{x} = \begin{pmatrix} \dfrac{\mathbf{p}}{m} \\ -\dfrac{\partial U}{\partial \mathbf{R}} + \dfrac{1}{2}\displaystyle\sum_{i,j}\dfrac{\partial J_{ij}}{\partial \mathbf{R}}(1 - \mathbf{e}_i \cdot \mathbf{e}_j) \\ \mathbf{e} \times \displaystyle\sum_i J_i \mathbf{e}_i \end{pmatrix}. \tag{6}$$

If we assume that $\mathcal{H}$ is constant within a small time interval between time t and $t+\Delta t$, the solution of Eq. (3) can be formally written as

$$\mathbf{x}(t + \Delta t) = e^{\mathcal{H}\Delta t}\mathbf{x}(t). \tag{7}$$

According to STD [2],

$$e^{\mathcal{H}\Delta t}\mathbf{x} = e^{(\mathcal{P}+\mathcal{F}+\mathcal{S})\Delta t}\mathbf{x} = e^{\mathcal{S}(\Delta t/2)}e^{(\mathcal{F}+\mathcal{P})\Delta t}e^{\mathcal{S}(\Delta t/2)}\mathbf{x} + O(\Delta t^3). \tag{8}$$

Physically, when one deals with the operation of $(\mathcal{F}+\mathcal{P})$, only variables related to the lattice, i.e., $\mathbf{R}$ and $\mathbf{p}$, are affected but not the spin directions $\{\mathbf{e}_k\}$. In other words, $\{\mathbf{e}_k\}$ are frozen during the operation of $(\mathcal{F}+\mathcal{P})$, and the second equation in Eq. (2) can be written as

$$\frac{d\mathbf{p}_k}{dt} = -\frac{\partial U}{\partial \mathbf{R}_k} + \frac{1}{2}\sum_{i,j}\frac{\partial J_{ij}}{\partial \mathbf{R}_k}(1 - \mathbf{e}_i \cdot \mathbf{e}_j) \equiv -\frac{\partial U'}{\partial \mathbf{R}_k}. \tag{9}$$

With frozen spins, Eq. (2) then reduces to the equation of motion of conventional MD, for which the symplecticity of the STD has already been proven [2].

The operation of the *spin precession operators* $\mathcal{S}$ only affects the elements of the spin subspace and does not affect the lattice that can be considered frozen during the evolution of $\mathbf{x}$ under the operation of $\mathcal{S}$. Yet, the operation of $\mathcal{S}$ is complex to implement because the effective field encountered by a particular spin is a self-consistent field determined by all the neighboring spins. Indeed, the third equation in Eq. (2) constitutes a system of coupled first-order differential equations, and instead of a single equation like the case of the coordinates and momenta, the spin precession operators $\mathcal{S}$ in Eq. (4) bring changes to itself via its action on S. It is impractical to solve for the spin precessions of a reasonably large simulation system taking into account its self-consistent nature. We can circumvent the problem by decomposing the spin subsystem further into a series of single spin precession, an action that is *not* needed for the lattice subsystem. By using the second-order STD, one can transform this messy problem into a series of single spin rotations. However, the trade-off is an $O(\Delta t^3)$ trajectory error. Thus, the spin directions are further decomposed into a direct sum of subspaces, each of which only contains the spin direction of a single atom. The spin precession operator $\mathcal{S}$ can then be written as the sum of a series of single spin precession operators [3], i.e., $\mathcal{S} = (\mathcal{S}_1 + \mathcal{S}_2 + \ldots + \mathcal{S}_{N-1} + \mathcal{S}_N)$, so that in the STD scheme we may write

$$e^{\mathcal{S}\tau}\mathbf{x} = e^{\mathcal{S}_1(\tau/2)}e^{\mathcal{S}_2(\tau/2)}\cdots e^{\mathcal{S}_{N-1}(\tau/2)} \times e^{\mathcal{S}_N\tau}e^{\mathcal{S}_{N-1}(\tau/2)}\cdots e^{\mathcal{S}_2(\tau/2)}e^{\mathcal{S}_1(\tau/2)}\mathbf{x} + O(\tau^3). \quad (10)$$

Since the rotation of a single spin is analytically solvable [15] involving no numerical dissipation of the total energy, this leads to the symplecticity of the method.

III. PARALLEL ALGORITHM

Parallel algorithms of MD are usually designed around three types of decomposition schemes, i.e., atomic, force, and spatial [8,12], designed to redistribute the workload over a set of processors or threads to speed up the computation by performing independent procedures simultaneously. Trobec *et al.* [13] reviewed several symplectic integration methods for MD simulations with parallel algorithms. Although the leap-frog-Verlet (LFV) and split integration symplectic (SIS) methods are different on how the Hamiltonian is split, both of them belong to the second-order STD. The LFV (or velocity Verlet) algorithm is indeed equivalent to the STD used in MD cases, with solution [2]

$$\mathbf{x}(t+\Delta t) = e^{\mathcal{H}\Delta t}\mathbf{x}(t) = e^{\mathcal{F}\Delta t/2}e^{\mathcal{P}\Delta t}e^{\mathcal{F}\Delta t/2}\mathbf{x}(t) + O(\Delta t^3) \quad (11)$$

for the operator $\mathcal{H}=(\mathcal{F}+\mathcal{P})$ and $\mathbf{x}$ is the generalized coordinates. Within a single time step, these three evolutionary steps have to be processed sequentially. Thus, a parallel algorithm of any of the three types of decompositions can only be implemented within each evolutionary step [13].

This type of algorithm is effective only when each evolution operation may involve with a sufficiently large number of variables simultaneously. However, in SLD simulations, the evolution of the spin has to be treated sequentially. This situation may be most easily visualized in the pure spin dynamics example of a one-dimensional (1D) spin chain system with ten spins, as depicted schematically in the top half of Fig. 1. The evolution of the spin chain from time t to $t+\Delta t$ in the STD scheme can be written as

$$\mathbf{e}(t+\Delta t) = e^{\mathcal{S}_1\Delta t/2}e^{\mathcal{S}_2\Delta t/2}\cdots e^{\mathcal{S}_{10}\Delta t}\cdots e^{\mathcal{S}_2\Delta t/2}e^{\mathcal{S}_1\Delta t/2}\mathbf{e}(t) + O(\Delta t^3), \quad (12)$$

where $e^{\mathcal{S}_k\Delta t}$ is the evolution operator of $\mathbf{e}_k(t)$ over the time interval $t\to t+\Delta t$ and $\mathbf{e}(t)=\{\mathbf{e}_k(t)\}$ is the system of unit atomic spin vectors. In Eq. (12), the evolution of the spin system is processed from spin 1 to 10, then reversely from spin 10 to 1. For simplicity, it may be represented as (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1). In other words, the evolution of spin 1 depends on spin 2, the evolution of which depends on spin 3 up to spin 10. It is impossible to do any parallel programming under such circumstances. Omelyan *et al.* [4] proposed an iterative method on the spin precession that is suitable for parallel programming. However, its perturbative nature is a lethal cause to the nonconservation of energy when a large number of time steps are involved even though the error is only $O(\Delta t^3)$.

In most computer simulations, a cutoff distance for the interatomic potential is used to account for screening effects and to cut down on unnecessary computation. The dynamics of an atom and its spin thus only depend on those in a defined neighborhood and are independent of those outside this neighborhood. In other words, the evolution of an atom and its spin only depends on information from a *selected number* of atoms to which it is linked rather than from the *entire* system.

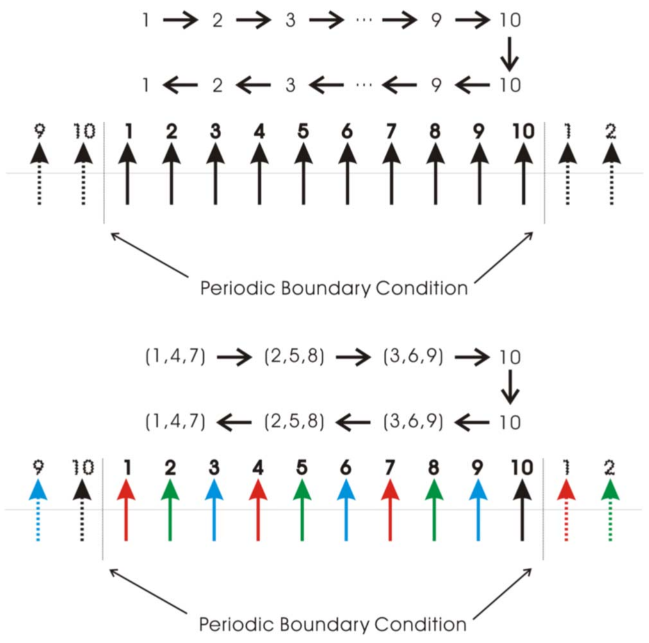

FIG. 1. (Color online) Schematic picture of 1D spin chain containing ten spins, where the Suzuki-Trotter decomposition is (top) sequential and (bottom) parallelize.

For the 1D spin chain, for example, if the exchange coupling is assumed to extend to the second nearest neighbors only, the evolution of spin 1 is independent of spins 4 and 7 (the bottom of Fig. 1). In addition, according to STD,

$$\begin{aligned}e^{(A+B+C)\tau}\mathbf{x}(t) &= e^{A\tau/2}e^{B\tau/2}e^{C\tau}e^{B\tau/2}e^{A\tau/2}\mathbf{x}(t) + O(\tau^3)\\ &= e^{B\tau/2}e^{A\tau/2}e^{C\tau}e^{A\tau/2}e^{B\tau/2}\mathbf{x}(t) + O(\tau^3)\\ &= e^{C\tau/2}e^{B\tau/2}e^{A\tau}e^{B\tau/2}e^{C\tau/2}\mathbf{x}(t) + O(\tau^3). \quad (13)\end{aligned}$$

Thus, the order of the operation is immaterial if they are interchanged according to Eq. (13). This property allows us to rearrange $(1,2,\ldots,10,\ldots,2,1)$ into (1, 4, 7, 2, 5, 8, 3, 6, 9, 10, 9, 6, 3, 8, 5, 2, 7, 4, 1) or ({1,4,7}, {2,5,8}, {3,6,9}, 10, {3,6,9}, {2,5,8}, {1,4,7}). Because atoms 1, 4, and 7 in {1,4,7} are independent, they evolve independently and can be processed simultaneously in different threads, followed by the group {2,5,8}, then {3,6,9}, and so on. For the remaining atom 10, it is a group on its own. We may use this approach to formulate a parallel algorithm in which atoms are collected into groups within which members are all independent and can therefore be processed simultaneously in different threads. Of course, it is important that before starting any calculation on a specific group, one must ensure that the calculation on the previous group is completed. In other words, a "barrier" directive in the programming code is needed.

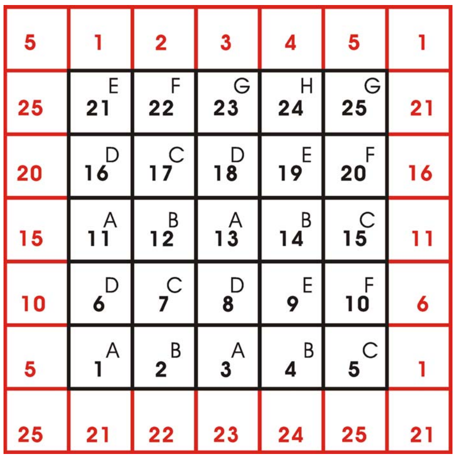
5 1 2 3 4 5 1
25 E 21 F 22 G 23 H 24 G 25 21
20 D 16 C 17 D 18 E 19 F 20 16
15 A 11 B 12 A 13 B 14 C 15 11
10 D 6 C 7 D 8 E 9 F 10 6
5 A 1 B 2 A 3 B 4 C 5 1
25 21 22 23 24 25 21

FIG. 2. (Color online) A two-dimensional system is cut into 25 link cells with periodic boundary condition. Cells are allocated into group A to group H.

In two-dimensional (2D) and three-dimensional (3D) cases, although one may still use the atomic delinking approach, a spatial delinking approach may be more convenient. As in conventional MD, the simulation box is usually divided into many smaller boxes, called link cells [16]. Since the edge of each cell is larger than the cutoff distance(s) of interatomic potential and/or exchange coupling, the forces on a particular atom or spin can only come from atoms within its own cell or surrounding cells. The implementation of link cells greatly enhances the computation efficiency by reducing redundant calculations. Figure 2 is an illustration of a 2D system divided into 25 link cells with periodic boundary condition. Treating each cell as a subsystem, they may be grouped into groups A–H, so that any atom in a member cell of a group is independent of any atom in any other cell of the same group. As a result, cells in the same group can be processed simultaneously in different threads. For example, cells {1,3,11,13} constitute group A, {2,4,12,14} group B, etc. Then, cells 1, 3, 11, and 13 can be processed concurrently. The working sequence (1,2,…,25…,2,1) can be recast into a different one (A,B,…,H,…,B,A), i.e.,

$$e^{L_1(\Delta t/2)}e^{L_2(\Delta t/2)}\cdots e^{L_{25}\Delta t}\cdots e^{L_2(\Delta t/2)}e^{L_1(\Delta t/2)}\mathbf{x}(t)$$
$$\rightarrow e^{L_A(\Delta t/2)}e^{L_B(\Delta t/2)}\cdots e^{L_H\Delta t}\cdots e^{L_B(\Delta t/2)}e^{L_A(\Delta t/2)}\mathbf{x}(t). \tag{14}$$

It is to be noted that spins within a link cell have to be processed sequentially according to STD. The 3D case is similar to 2D case, but the number of cells in a group is increased to 26. In practice, a subprogram can be written to allocate members into groups automatically.

Essentially, the parallel algorithm works to rearrange the order of the evolutionary operations, without violating the validity of STD. Then, one can gather them into groups in which the cells are independent, so that in each group evolutionary step all the cells in the group can be processed simultaneously in parallel. Of course, the evolutionary operation of different groups still has to be treated sequentially.

The integrity of the algorithm is established by checking with Bernstein's conditions [17] and Lamport's sequential consistency model [18]. Bernstein's condition states that when there are two program fragments P_i and P_j, with input $I_{i(j)}$ and output $O_{i(j)}$ variables, they can be calculated simultaneously if $I_{i(j)} \cap O_{j(i)} = \emptyset$ and $O_i \cap O_j = \emptyset$. It means that the same memory cannot be shared between different threads, except for inputs only. Otherwise, it causes race condition. Since each operator only needs information on itself and its surroundings, they do not need data from other members of the same group nor alter any of their properties. Therefore, it satisfies Bernstein's condition. Moreover, working on the same group members in sequential or parallel should be the same, as all processors are just calculating according to the same set of equations of motion, and each calculation is independent of the output of its group member. It satisfies Lamport's sequential consistency model.

Indeed, the foregoing spatial decomposition scheme can also be applied to MD simulations. In this case, the use of protective locks [18] on memory (e.g., the C/C++ directive "#pragma omp atomic" in OPENMP), which may slow down the program in the evaluation of forces, can be saved. Since forces are calculated in pairs (Newton's third law), the outputs do not only modify the data within a particular link cell, they also modify those in surrounding. The present method cancels the need of a lock within its own link cell. Surely, there is trade-off between this and the load balance. From the foregoing example, one may note that idle processors are unavoidable.

IV. EFFICIENCY AND STABILITY

In the following, the efficiency of the proposed algorithm will be evaluated. For this purpose, simulations are performed according to the SLD equations of motion [i.e., Eq. (2)] in a microcanonical (*NVE*) ensemble of ferromagnetic iron [1]. We used the Dudarev-Derlet potential [19] as the potential U. An *ad hoc* pairwise J_{ij} is obtained by fitting to *ab initio* data [1]. In each case, atoms are initially placed in a regular bcc lattice in [100] directions with lattice constant $a=2.8665$ Å. Five different lattice configurations are used. They are all cubes with edge sizes of 20a, 30a, 40a, 50a, and 80a, with numbers of atoms of 16 000, 54 000, 128 000, 250 000, and 1 024 000, respectively. The velocities of all the atoms are initially set to zero. All the spins in the left-hand side of the simulation cell are initialized to point upward whereas those on the right-hand side point downward, making a 180° domain wall structures. Periodic boundary conditions are applied along x, y, and z. The edge of each link cell is larger than the cutoff distance of the interatomic potential and exchange function for spin-spin interaction. The same spatial decomposition schemes as mentioned before are applied to the spin and lattice subsystems.

The actual run time of the foregoing model (i.e., $\mathcal{H}=\mathcal{F}+\mathcal{P}+\mathcal{S}$) and the one with only the spin part (i.e., $\mathcal{H}=\mathcal{S}$ only; a fixed lattice system) is recorded for 10 000 time steps. A computer with two Intel xeon quad-core X5355 CPU and 4 Gbytes random access memory (RAM) is used. With a total of eight cores, only the efficiencies of one to eight threads

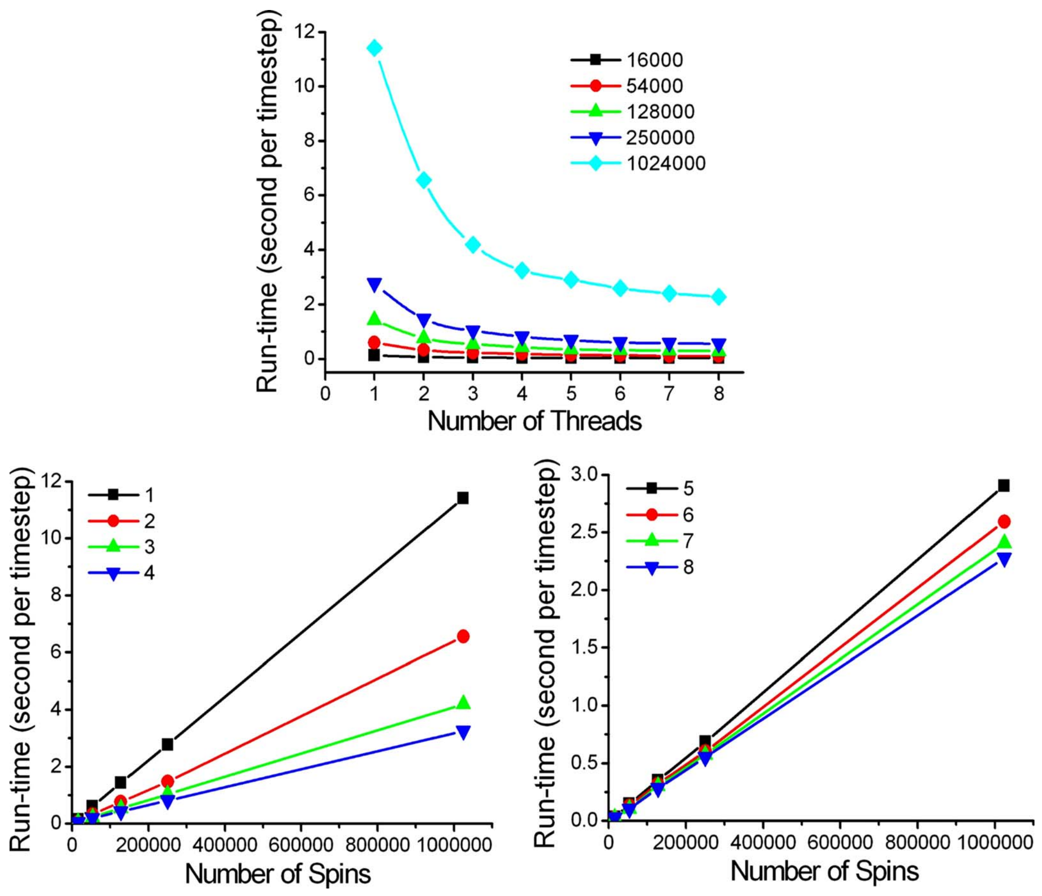

FIG. 3. (Color online) The actual run time of spin-only system (top) versus number of threads with 16 000, 54 000, 128 000, 250 000, and 1 024 000 spins, (bottom left) versus number of spins with threads from 1 to 4, and (bottom right) versus number of spins with threads from 5 to 8.

are measured. Scientific Linux 5.0 and Intel C++ compiler 10.0 with OPENMP package are used. The efficiency is defined as [20]

$$\text{eff}(n) = \frac{T(1)}{nT(n)}, \tag{15}$$

where eff(n) is the efficiency and $T(n)$ is the run time for n threads. The term efficiency, of course, does not relate directly to the actual run time.

Figures 3 and 4 show the run time and efficiency of the spin-only subsystem, while Figs. 5 and 6 show the case in which a coupled spin and lattice subsystem is treated. In both cases, the run time decreases with an increasing number of threads, independent of the size of the systems (ranging from 10 000 to over 10^6 spins and atoms). However, they tend to saturate at about eight threads due to Amdahl's law [21], which states that because of the sequential component, a parallelized program cannot be speeded up more than n times.

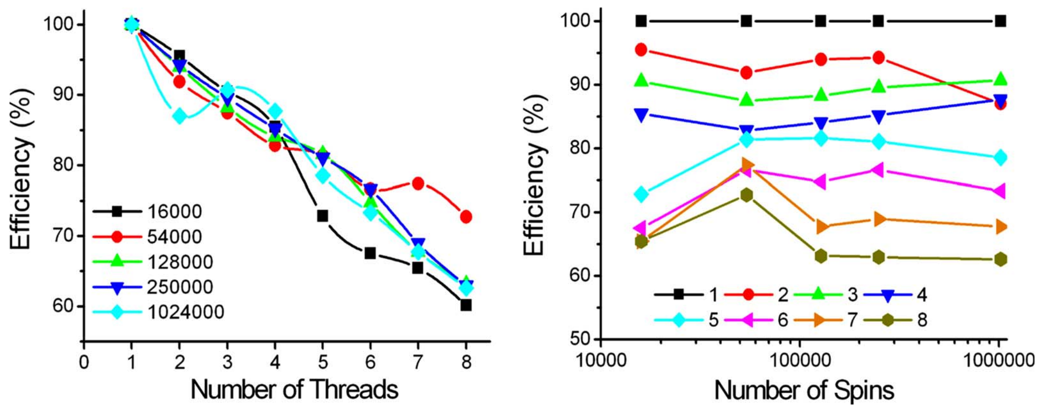

FIG. 4. (Color online) The efficiency of spin-only system (left) versus number of threads with 16 000, 54 000, 128 000, 250 000, and 1 024 000 spins and (right) versus number of spins with threads from 1 to 8.

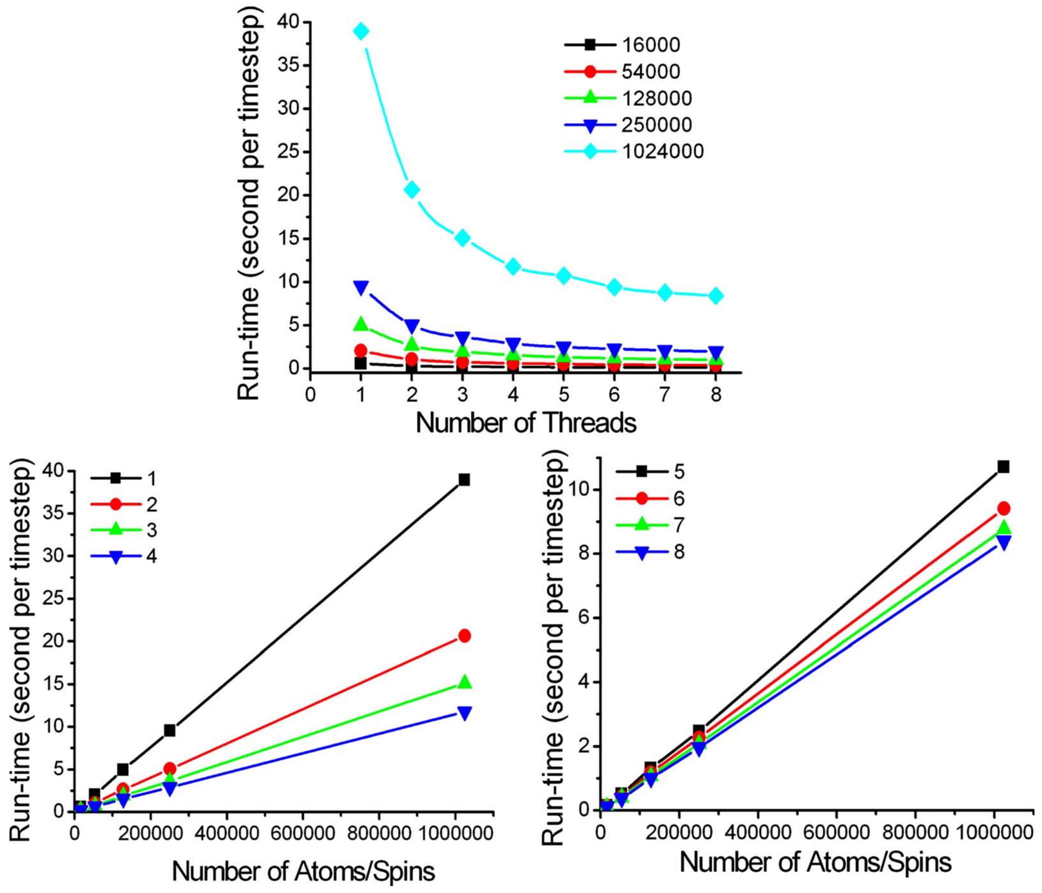

FIG. 5. (Color online) The actual run time of spin-lattice system (top) versus number of threads with 16 000, 54 000, 128 000, 250 000, and 1 024 000 spins, (bottom left) versus number of spins with threads from 1 to 4, and (bottom right) versus number of spins with threads from 5 to 8.

Of course, one can always improve on the efficiency by fine tuning the program, such as eliminating unnecessary sequential components. Well designed hardware architecture or an expensive compiler may also help. However, this is out of the scope of this paper.

Next, we checked whether the run times appear to increase linearly with the systems' size or not. Because of the order-*N* nature of the link-cell system, the total time spent on force calculations increases only linearly with the number of atoms. When eight threads are used, a 1 024 000 atoms system takes about 2.3 s/time step for the spin-only system and 8.4 s/time step for the spin-lattice systems. If one uses a time step of 1 fs, it takes about 2.7 and 9.7 days, respectively, to do a simulation of 100 ps.

As can be seen in Fig. 4, the efficiency generally decreases as the number of threads increases due to Amdahl's law. However, exceptions do exist because of a higher proportion of idling processors, e.g., the efficiency using three

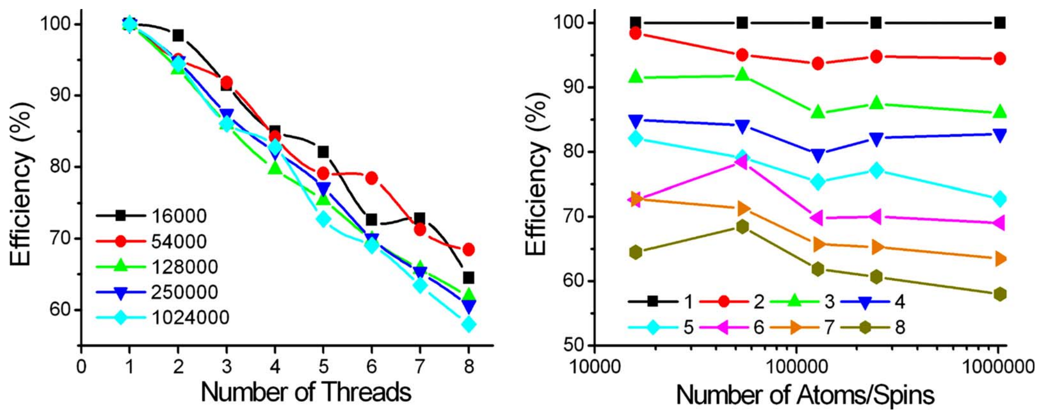

FIG. 6. (Color online) The efficiency of spin and lattice coupled system (left) versus number of threads with 16 000, 54 000, 128 000, 250 000, and 1 024 000 spins and (right) versus number of spins with threads from 1 to 8.

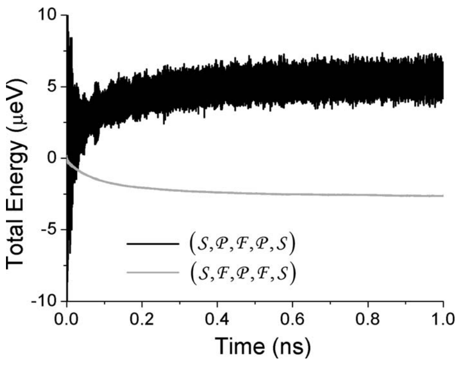

FIG. 7. The total energy per atom versus time for a microcanonical ensemble simulation of the dynamical relaxation of a 180° domain wall. The energy is subtracted by their initial values. Two algorithms, the $(\mathcal{S},\mathcal{F},\mathcal{P},\mathcal{F},\mathcal{S})$ and the $(\mathcal{S},\mathcal{P},\mathcal{F},\mathcal{P},\mathcal{S})$, in both of which the parallel algorithm for STD is incorporated, are compared.

threads is higher than that using two threads in the case of 1 024 000 spins. This can be understood as follows. Using the potential cutoff, the system was automatically separated into 166 375 link cells. They were put into groups so that any member in a group is not a neighbor of any other member, i.e., based on the same principle as the 2D case (Fig. 2). A spin-only system with 1 024 000 spins (or 166 375 link cells) was separated into 18 groups—8 groups with 19 683 members (or link cells), 2 groups with 2783 members, 2 groups with 1431 members, 2 groups with 207 members, 1 group with 31 members, 1 group with 30 members, and 2 groups with 4 members. Suppose the computer time involved with each link cell is about equal, then the idle processor time of each group can be calculated from the group size modulo the number of threads, resulting in 15 units of idle processor time for the two-thread case and 7 units for the three-thread case. Of course, there are other factors that dictate the processing speed, such as the computer architecture, operating system, processor, memory, etc. Moreover, since link cells are not required to have the same number of atoms, load balancing is another important issue in this kind of spatial decomposition algorithm.

As a whole, the efficiency decreases roughly linearly from 100% to about 60%–70%, as the number of threads used increases from 1 to 8. Compared to the efficiency of some other parallel algorithms, such as certain force decomposition method [10], which can attain 98.88%, the current method are certainly inferior. However, one must remember that we are dealing with an extreme case of STD, in which most parallel processing schemes do not apply.

Numerical stability is another important factor to consider. Figure 7 shows that the total energy per atom of the 180° domain wall system with 54 000 atoms and spins as a function of computation time using, respectively, the $(\mathcal{S},\mathcal{F},\mathcal{P},\mathcal{F},\mathcal{S})$ and the $(\mathcal{S},\mathcal{P},\mathcal{F},\mathcal{P},\mathcal{S})$ algorithms, with the proposed parallel algorithm already incorporated. The energy is subtracted by its initial value. The $\mathcal{F}$, $\mathcal{P}$, and $\mathcal{S}$ are the force, momentum, and spin precession operators, respectively, as we defined in Eq. (4). $(\mathcal{S},\mathcal{F},\mathcal{P},\mathcal{F},\mathcal{S})$ and $(\mathcal{S},\mathcal{P},\mathcal{F},\mathcal{P},\mathcal{S})$ refer to the sequence of operations in the STD. For example, $(\mathcal{S},\mathcal{F},\mathcal{P},\mathcal{F},\mathcal{S})$ means $e^{\mathcal{H}\Delta t} \to e^{\mathcal{S}(\Delta t/2)}e^{\mathcal{F}(\Delta t/2)}e^{\mathcal{P}\Delta t}e^{\mathcal{F}(\Delta t/2)}e^{\mathcal{S}(\Delta t/2)}$. The total energy per atom remains constant with error within the order of 10^{-5} eV and sometimes even smaller than 10^{-6} eV. This demonstrates the stability of our procedure for long runs ($\sim$1 ns). Furthermore, the speed of the $(\mathcal{S},\mathcal{P},\mathcal{F},\mathcal{P},\mathcal{S})$ SLD algorithm is found to be approximately half the speed of the velocity Verlet algorithm when compared with conventional MD. Moreover, the $(\mathcal{S},\mathcal{F},\mathcal{P},\mathcal{F},\mathcal{S})$ algorithm is approximately 60% slower than the $(\mathcal{S},\mathcal{P},\mathcal{F},\mathcal{P},\mathcal{S})$ algorithm. Since evaluating forces includes the time consuming calculations of the potential, it is obvious that performance can be much improved by calculating it once, instead of twice, per cycle.

V. CONCLUSION

A parallel algorithm is developed for SD and SLD simulations. Since the spin precession operator produces changes in itself, the set of first-order differential equations has to be solved via a complex self-consistent procedure. By using the second-order STD, one can avoid the complexity by transforming this problem into a series of single spin rotations. Since the rotation of a single spin is analytically solvable involving no numerical dissipation of the total energy, inclusion of the spin subsystem does not affect the symplecticity of the method, which has been proven in the MD case. At first sight, each evolutionary step for the spin subsystem has to be processed sequentially, one spin at a time, and a parallel algorithm appears difficult. We have shown that, by arranging the order of the evolution operations in groups and making sure they are independent of each other within a group using the cutoff distance, a parallel algorithm for STD can be designed. The efficiency is about 60%–70% when using eight threads. The run time per time step is linearly proportional to the system size. Such algorithms can also be implemented on conventional MD and SD programs, without handling complicated mathematics. It can be applied to very large system ($>10^6$ atoms and/or spins). It also shows good stability and energy conservation in long runs of 10^6 time steps. No iteration or further approximation is needed within our algorithm. Only the trajectory error with $O(\Delta t^3)$ from the STD remains. The proposed method is not mutually exclusive to other decomposition methods. For example, one can use spatial decomposition twice, which gives two layers of decomposition. The first layer can be distributed within the cluster by message passing interface (MPI) and the second layer is distributed within the machine by OPENMP.

ACKNOWLEDGMENTS

This work was supported by the Hong Kong RGC under Grants No. 530507 and No. 532008. We would also like to thank Dr. S. L. Dudarev for helpful discussions.

[1] P. W. Ma, C. H. Woo, and S. L. Dudarev, Phys. Rev. B **78**, 024434 (2008); in *Electron Microscopy and Multiscale Modeling*, edited by A. S. Avilov, S. L. Dudarev, and L. D. Marks, AIP Conf. Proc. No. 999 (AIP, New York, 2008), p. 134.
[2] N. Hatano and M. Suzuki, Lect. Notes Phys. **679**, 37 (2005).
[3] I. P. Omelyan, I. M. Mryglod, and R. Folk, Phys. Rev. Lett. **86**, 898 (2001).
[4] I. P. Omelyan, I. M. Mryglod, and R. Folk, Phys. Rev. E **64**, 016105 (2001).
[5] I. P. Omelyan, I. M. Mryglod, and R. Folk, Phys. Rev. E **66**, 026701 (2002).
[6] S. H. Tsai, M. Krech, and D. P. Landau, Braz. J. Phys. **34**, 382 (2004).
[7] S. H. Tsai, H. K. Lee, and D. P. Landau, Am. J. Phys. **73**, 615 (2005).
[8] S. J. Plimpton and B. A. Hendrickson, in *Materials Theory and Modeling*, MRS Symposia Proceedings No. 291 (Materials Research Society, Pittsburgh, 1992), p. 37.
[9] S. J. Plimpton, J. Comput. Phys. **117**, 1 (1995).
[10] R. Murty and D. Okunbor, Parallel Comput. **25**, 217 (1999).
[11] A. Fijany, T. Cağinm, A. Jaramillo-Botero, and W. Goddard III, Adv. Eng. Software **29**, 441 (1998).
[12] G. S. Heffelfinger, Comput. Phys. Commun. **128**, 219 (2000).
[13] R. Trobec, F. Merzel, and D. Janežič, J. Chem. Inf. Comput. Sci. **37**, 1055 (1997).
[14] S. L. Dudarev and P. M. Derlet, J. Comput.-Aided Mater. Des. **14**, 129 (2007).
[15] J. B. Marion and S. T. Thornton, *Classical Dynamics of Particles and Systems*, 4th ed. (Harcourt Brace Jovanovitch, San Diego, 1995).
[16] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford University Press, New York, 1987).
[17] A. J. Bernstein, IEEE Trans. Electron. Comput. **EC-15**, 757 (1966).
[18] Leslie Lamport, IEEE Trans. Comput. **C-28**, 690 (1979).
[19] S. L. Dudarev and P. M. Derlet, J. Phys.: Condens. Matter **17**, 7097 (2005).
[20] L. R. Scott, T. Clark, and B. Bagheri, *Scientific Parallel Computing* (Princeton University Press, Princeton, NJ, 2005).
[21] G. Amdahl, AFIPS Conf. Proc. **30**, 483 (1967).