PHYSICS COMPUTING

INTRODUCTION

Computing is the third mode of research in physics, which bridges the gap between analytical theory and laboratory experiment (1). Experiments search for patterns in complex natural phenomena. Theories encode the discovered patterns into mathematical equations that provide predictive laws for the behavior of nature. Computer simulations solve these equations numerically in their full complexity, where analytical solutions are prohibitive because of a large number of degrees of freedom, nonlinearity, or lack of symmetry. In computer simulations, environments are controlled perfectly and extreme conditions are accessible beyond the scope of laboratory experiments.

Setting up a computer simulation involves several design steps (2): 1) Formulate a mathematical model to describe the physical phenomenon of interest; 2) discretize the model, which often consists of continuous differential or integral equations (i.e., calculus), into algebraic forms in order to allow for a numerical solution on digital computers; 3) select numerical algorithms to solve the algebraic equations efficiently; 4) translate the algorithms into a set of instructions, which constitute a computer program; and 5) perform a computer experiment by executing the program on a computer. Parallel computing has become an essential part of high-performance computer simulations that require massive computations. Basic ingredients of physics computing thus include mathematical models, numerical algorithms, and parallel computing.

MATHEMATICAL MODELS IN PHYSICS

Particle Versus Field Models

Mathematical models in physics are either the particle type or the field type (2). Particle models trace the motion of many interacting particles (Fig. 1). An example is Newton's second law of motion in classic mechanics, where a physical system consisting of N particles is represented by a set of coordinates $\mathbf{r}_k = (x_k, y_k, z_k) \,|\, k = 1,\ldots, N$ (2–4). This law is formulated as coupled ordinary differential equations

$$m_k \frac{d^2}{dt^2}\mathbf{r}_k = -\frac{\partial}{\partial \mathbf{r}_k} V(\mathbf{r}^N) \tag{1}$$

where m_k is the mass of the kth particle, $\mathbf{r}^N = (\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N)$, and V is the interparticle potential energy. Molecular dynamics (MD) simulations (2–4) follow the particle trajectories $\mathbf{r}_k(t)$ by integrating equation 1 numerically with respect to time t.

Field models deal with functions extending over the space. For example, the above classic mechanical law is not valid on the atomic scale and must be replaced by the quantum mechanical law. The dynamics of N quantum-mechanical particles is described by a parabolic partial differential equation called the Schrödinger equation (5,6)

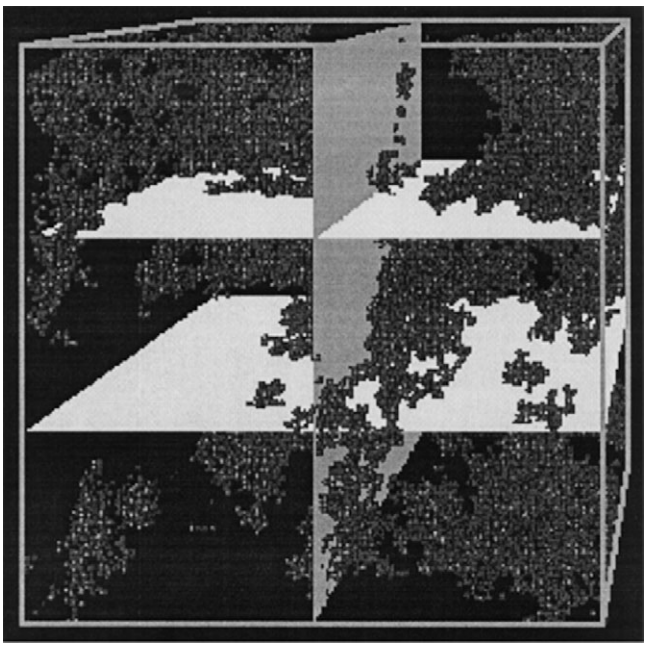

Figure 1. A particle model. The spheres represent silicon and oxygen atoms in a porous silicon dioxide material. The planes represent partition boundaries used to map the physical system onto processors in a parallel computer.

$$i\hbar\frac{\partial}{\partial t}\psi(\mathbf{r}^N, t) = [-\sum_{k=1}^{N}\frac{\hbar^2}{2m_k}\nabla_k^2 + V(\mathbf{r}^N)]\psi(\mathbf{r}^N, t) \tag{2}$$

where $i = \sqrt{-1}$, $= 1.05 \times 10^{-34}$ Js is the Plank constant, $\nabla_k^2 = \partial^2/\partial x_k^2 + \partial^2/\partial y_k^2 + \partial^2/\partial z_k^2$ is the Laplacian operator, and $\psi(\mathbf{r}^N, t)$ is a complex-valued wave function. The square $|\psi(\mathbf{r}^N, t)|^2$ of the wave function is proportional to the probability to find the N particles at positions $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$. Quantum dynamical (QD) simulations integrate equation 2 numerically with respect to t (5, 6).

Another example of a field model is Maxwell equations, which describe the behavior of electromagnetic fields (see MAXWELL EQUATIONS). With a time-independent charge distribution $\rho(\mathbf{r})$, Maxwell equations are reduced to Poisson's equation (5,6)

$$\nabla^2\phi(\mathbf{r}) = -\frac{\rho(\mathbf{r})}{\epsilon} \tag{3}$$

for the electrostatic potential $\phi(\mathbf{r})$, where ϵ is the permittivity (see PERMITTIVITY). Other commonly used field models in physics include those of elasticity and fluid mechanics.

Discretization

The above mathematical models are formulated as differential equations, where physical variables have continuous values. To perform computer simulations, these continuous laws must be cast into discrete algebraic forms, which are amenable to numerical solution on a digital computer. In a particle simulation, only the time variable must be discretized, so that the physical system is sampled at times $t_n = n\ \Delta t (n = 1, 2, 3, \ldots)$. For example, the most common

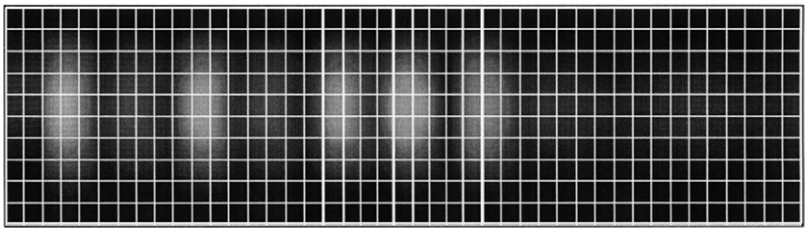

Figure 2. Finite-difference discretization of a field model. An electron wave function in a two-dimensional strip is sampled on regular grids

discretized form (2–4) of equation 1 is

$$m_k \frac{\mathbf{r}_k(t_{n+1}) - 2\mathbf{r}_k(t_n) + \mathbf{r}_k(t_{n-1})}{\Delta t^2} = \mathbf{F}_k(\mathbf{r}^N(t_n)) \qquad (4)$$

where $\mathbf{F}_k = -\partial V/\partial \mathbf{r}_k$ is the force acting on the kth particle.

In field simulations, spatial variables are also discretized, so that field values are sampled on a finite set of points. In the finite difference method (2), the continuum of space is replaced with a grid of points that are separated by spatial distances Δx, Δy, and Δz in the x, y, and z directions, respectively (Fig. 2). The continuous derivatives in the equations are then replaced with algebraic expressions involving the finite quantities, Δx, Δy, Δz, and Δt. However, many physical problems do not allow the use of a regular grid because of complex geometry. These problems are better discretized by the finite element method (7), in which a complex domain is divided into a mesh of geometrically simple subdomains called finite elements (Fig. 3). The original equations, expressed in terms of derivatives, are transformed into algebraic equations in each element. These discrete element equations are then combined to form a system equation for the entire domain.

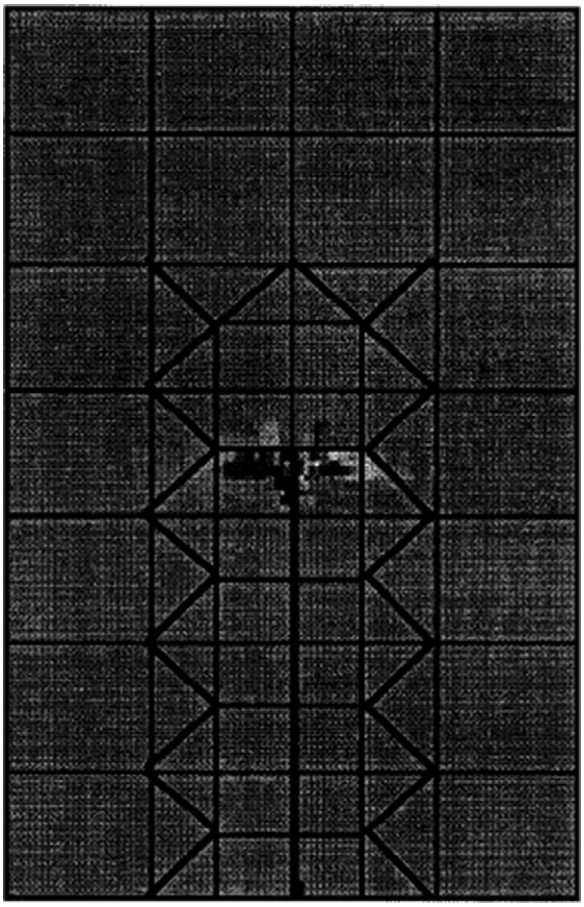

Figure 3. Finite-element discretization of a field model. A shear stress field in a two-dimensional strip with a crack is sampled using finite elements.

Deterministic Versus Stochastic Simulations

Computer simulations are either deterministic or stochastic. Deterministic simulations usually deal with mathematical initial value problems; i.e., differential equations such as equations 1 and 2 are integrated forward in time starting with some initial configuration. Stochastic simulations use random numbers to provide approximate solutions to large-scale problems, whereas deterministic solutions are intractable.

Stochastic approaches are often used to compute the equilibrium properties of matter based on the law of statistical mechanics. For a classic N-particle system described by equation 1, the thermal average of any physical quantity $A(\mathbf{r}^N)$ is given by a multidimensional quadrature (3–5)

$$\langle A \rangle = \int d\mathbf{r}^N P(\mathbf{r}^N) A(\mathbf{r}^N) \qquad (5)$$

where

$$P(\mathbf{r}^N) = \frac{exp(-V(\mathbf{r}^N)/k_B T)}{\int d\mathbf{r}^N exp(-V(\mathbf{r}^N)/k_B T)} \qquad (6)$$

is a probability function. In equation 6, $k_B = 1.38 \times 10^{-23}$ JK^{-1} is Boltzmann's constant and T is the temperature. Direct numerical quadrature of equation 5 requires computations that scale exponentially with N, and it is intractable for large N. Monte Carlo (MC) method uses random sampling of abscissas to evaluate equation 5 approximately (3–5). In the Metropolis algorithm, a Markov chain (see MARKOV PROCESSES) is used to achieve an importance sampling, i.e., generating a sequence of random states such that each state occurs with the probability $P(\mathbf{r}N)$.

Stochastic approaches are also used to study the ground state of a quantum many-particle system. The ground state is a stationary solution $\psi(\mathbf{r}^N, t) = \psi(\mathbf{r}^N) exp(-iEt/\hbar)$ of equation 2 with the lowest energy E, and it can be projected out by integrating equation 2 for long imaginary time $\tau = it$ (3). The diffusion MC approach solves the imaginary-time Schrödinger equation by generating a random walk (3). The wave function is represented by an ensemble of

randomly generated configurations $\mathbf{r}N$. Each configuration $\mathbf{r}^N$ is displaced randomly according to the diffusion law and is duplicated or deleted according to its weight. Quantum problems are alternatively solved by another stochastic method called Green's function MC (8).

Another quantum MC method called path-integral MC calculates the thermal properties of a quantum-mechanical system. Using the path-integral formulation of the thermal density matrix, the simulation of a quantum-mechanical N-particle system is reduced to that of a classic N-molecule system with each molecule consisting of S atoms (3). The classic MC to solve equation 5 is then used to simulate the quantum system. This formulation becomes exact when the number of discretization points S approaches infinity.

The above examples manifest the quantum mechanical law of nature that a single quantum state is equivalent to an ensemble of classic states, which has a deep consequence in computer science. Although the positions and velocities of a classic N-particle system are encoded with $6N$ numbers, the amount of information contained in a quantum N-particle system $\psi(\mathbf{r}^N)$ grows exponentially with N. Consequently, although a classic computer can only encode a 0 or 1, a quantum computer can encode a weighted superposition of 0 and 1 using a single information unit called qubit (9). Therefore, a quantum computer working on an n-qubit register can perform 2^n calculations with one operation. This "quantum parallelism" allows a quantum computer to solve certain hard computational problems exponentially faster than any classic computers (9).

Hybrid Models

Often a single mathematical model is not sufficient to describe a physical phenomenon because of the wide range of length and time scales involved in the phenomenon. Hybrid physical models introduce multiple levels of abstraction to capture the essential physics across the length and time scales. For example, the quasi-continuum model uses direct atomistic calculations based on the MD method to provide inputs to the finite element analysis of continuum mechanics (10). Molecular dynamics can also be combined with first-principles electronic structure calculations to describe the breaking and formation of chemical bonds during macroscopic material processes (11). Enormous saving in computing is achieved by treating only the reactive portion of the system quantum mechanically and treating the rest, the environment, classically (12). Furthermore, hierarchical simulation models (13–15) seamlessly combine finite-element method, MD simulations with chemically reactive and nonreactive interatomic potentials, and quantum-mechanical simulation.

Numerical Algorithms

To write a simulation program, solutions to discretized algebraic models must be translated to a sequence of computer instructions based on some numerical algorithms. For example, the commonly used velocity–Verlet algorithm translates equation 4 into iterated operations of the following time-stepping procedures (3):

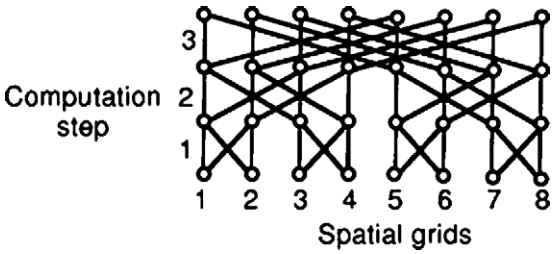

Figure 4. Computation pattern involved in the one-dimensional FFT.

1. Compute the forces $\mathbf{F}_k(t_n)$ as a function of $\mathbf{r}^N(t_n)$.
2. Update the velocities $\mathbf{v}_k \leftarrow \mathbf{v}_k(t_n) + \mathbf{F}_k(t_n)\Delta t/2m_k$ of the particles.
3. Obtain the new particle positions $\mathbf{r}_k(t_{n+1}) \leftarrow \mathbf{r}_k(t_n) + \mathbf{v}_k\Delta t$.
4. Compute the new forces $\mathbf{F}_k(t_{n+1})$ as a function of $\mathbf{r}^N(t_{n+1})$.
5. Obtain the new velocities $\mathbf{v}_k(t_{n+1}) \leftarrow \mathbf{v}_k + \mathbf{F}_k(t_{n+1})\Delta t/2m_k$.

Efficient algorithms are key to extending the scope of physics computing to larger spatial and temporal scales that are otherwise impossible to be simulated. As summarized below, these algorithms often use multiresolutions in both space and time.

Algorithms for Extending Spatial Scales

Solving Poisson's equation (Eq. 3) using a spatial grid is required for the simulation of various systems such as plasmas (see PLASMA ELECTROMAGNETIC WAVE PROPAGATION) and semiconductor devices. A primitive algorithm solves Poisson's equation on M grid points using $O(M^2)$ operations. In 1963, Hockney developed a fast direct solver for Poisson's equation, which requires only $O(M \log M)$ operations (2). This fast Poisson solver was based on the so-called fast Fourier transform (FFT) algorithm (2, 6), and it immediately made multidimensional plasma simulations feasible. The FFT in one spatial dimension consists of $\log_2 M$ computation steps, with each step operating on $M/2$ pairs of grid points (Fig. 4). Multidimensional FFT is achieved as a successive application of one-dimensional FFT for each spatial dimension (6).

The solution of Poisson's equation has been further revolutionized by the introduction of the multigrid method (MGM) by Brandt (6, 16). The MGM is based on the simple idea that slowly varying long-wavelength components of $\phi(\mathbf{r})$ can be accurately represented on a coarser grid. By employing hierarchical grids with coarser spacings (Fig. 5), the MGM solves Poisson's equation with $O(M)$ operations.

Another computationally intensive problem is the calculation of the electrostatic energy for N charged particles

$$V(\mathbf{r}^N) = \sum_{j=1}^{N-1} \sum_{k=j+1}^{N} \frac{Z_j Z_k}{\|\mathbf{r}_j - \mathbf{r}_k\|} \tag{7}$$

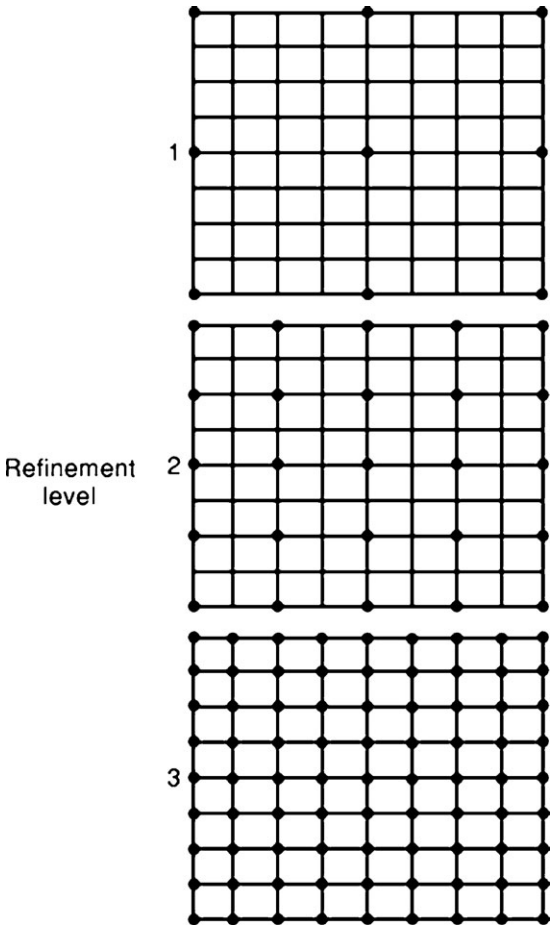

Figure 5. Hierarchical grids used in the MGM. A grid at the lth level is generated by decimating even rows and columns from the finer grid at the $(l+1)$th level.

where Z_j is the charge of the jth particle. Equation 7 also describes the gravitational energy of N point masses, where Z_j denotes the mass of the jth particle. Direct evaluation of equation 7 requires $O(N^2)$ operations. In 1987, Greengard and Rokhlin discovered an $O(N)$ algorithm called the fast multipole method (FMM) (17). This algorithm enabled million-to-billion particle astrophysical (18) and materials (15, 19) simulations. The FMM groups distant particles together and treats them collectively. Hierarchical grouping is facilitated by recursively dividing the physical system into smaller cells, generating a tree data structure (17). The root of the tree is at level 0, and it corresponds to the entire simulation box. A parent cell at level l is decomposed into $2 \times 2 \times 2$ children cells of equal volume at level $l+1$. The FMM uses the truncated multipole expansion (20) and the local Taylor expansion of the electrostatic potential field. By computing both expansions recursively for the hierarchy of cells (Fig. 6), the Coulomb potential is computed with $O(N)$ operations.

In simulations of bulk materials, periodic boundary conditions are often imposed, in which images of the simulation system are repeated infinitely to avoid the surface effects (3, 4). The conditionally convergent sum of the electrostatic energy caused by all the image charges is conventionally carried out by the Ewald summation technique (3, 21). This technique splits equation 7 into two convergent sums in the real and Fourier spaces, and computes both with $O(N3/2)$ operations. Various approaches have been proposed to incorporate periodic boundary conditions to the $O(N)$ FMM (22) as well as to compute various physical quantities such as stress tensor using the FMM (23).

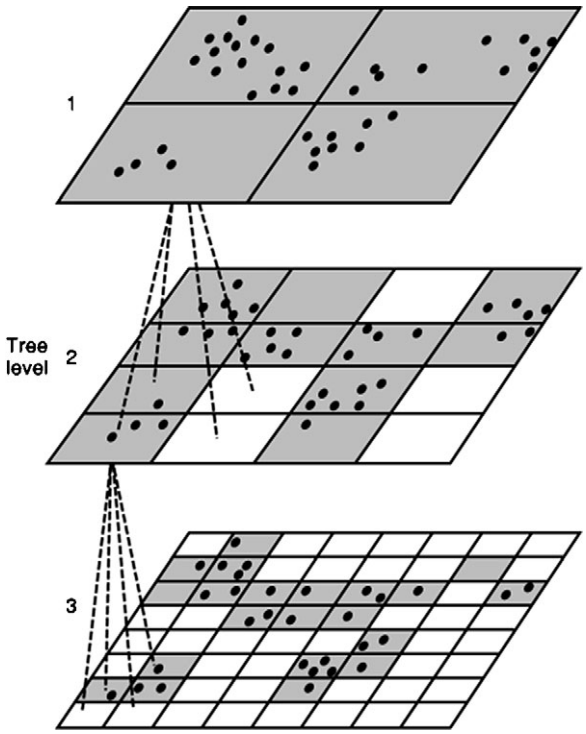

Figure 6. Hierarchical tree structure in the FMM for a two-dimensional system. A parent cell at the lth level is composed of four children cells at the $(l+1)$th level. The parent–child relation is illustrated by dashed lines. Particles are represented by circles.

Computationally more demanding is the exponentially complex quantum many-body problem, for which several algorithms have been designed to provide approximate solutions. The density functional theory (DFT) reduces the complexity by solving many one-electron problems self-consistently, instead of one many-electron problem (24). The DFT problem can be formulated as the minimization of an energy functional $V(\mathbf{r}N, \psi^{N\mathrm{el}})$, with respect to electronic wave functions (or Kohn–Sham orbitals) $\psi^{N\mathrm{el}}(\mathbf{r}) = \psi_n(\mathbf{r}) \mid n = 1,\ldots,N_{\mathrm{el}}$, subject to orthonormality constraints (N_{el} is the number of wave functions on the order of N) (25). Based on a data-locality principle called quantum nearsightedness (26), several $O(N)$ DFT algorithms have been designed, for which the computational cost scales linearly with the number of electrons (27). An example is a divide-and-conquer DFT algorithm (28), which has attained controlled error bounds, robust convergence properties, and energy conservation during MD simulation, to make large DFT-based MD simulation practical (29).

Algorithms for Extending Time Scales

The discrete time step Δt in MD simulations must be chosen sufficiently small such that the fastest characteristic

oscillations of the simulated system are accurately represented. However, many important physical processes are slow and are characterized by time scales that are many orders-of-magnitude larger than the Δt thus determined. Dynamic systems involving a wide range of time scales are called "stiff" (6). Molecular dynamics simulations of stiff systems require many iteration steps, and this severely restricts the applicability of the simulation.

Various approaches have been developed for long-time MD simulations of stiff systems. One approach introduces constraints to freeze high-frequency modes. This enables the use of a larger Δt, assuming that the high-frequency modes are unimportant for global conformational changes (30). In the subspace dynamics approach, low-frequency modes are systematically selected by diagonalizing the dynamical matrix of the system (31). Another approach called the multiple time scale (MTS) method uses different Δt for different force components to reduce the number of force evaluations (32, 33). Ordinary differential equations can be integrated using a large Δt by implicit-integration schemes (34). For example, the implicit Euler integrator is a low-pass filter that selects motions with eigenfrequencies less than $1/\Delta t$. The frozen fast motions can be integrated separately by normal-mode analysis (35, 36). Recently, more accurate implicit integrators have been proposed, which are symplectic (34). Symplectic integrators conserve the phase-space volume, and this symplecticness is essential for the long-time stability of orbitals (34).

Many long-time physical processes, such as thermally activated diffusion of an atom in a solid, occur as a sequence of rare events. Such processes have traditionally been treated with the transition state theory (37), which assumes that successive events are uncorrelated. Various schemes have been developed to accelerate the sampling of rare events without invoking such an assumption (38).

PARALLEL COMPUTING

Parallel computing technology (39) has extended the scope of computer simulations in terms of simulated system size and has become an essential part of physics computing. To perform parallel computer simulations efficiently, however, algorithms developed for serial computers must often be modified.

For example, let us consider the time-dependent Schrödinger equation (Eq. 2), for one particle. This equation is formally solved as

$$\psi(\mathbf{r}, t + \Delta t) = exp(-i\hat{H}\Delta t/\hbar)\psi(\mathbf{r}, t) \tag{8}$$

where the Hamiltonian operator is

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{r}) = \hat{K} + \hat{V} \tag{9}$$

For a one-dimensional problem on M grid points, the $\hat{V}$ operator is a diagonal $M \times M$ matrix. On the other hand, the $\hat{K}$ matrix is tridiagonal (6). A conventional solution to equation 2 on serial computers is based on the split-operator approach (40), in which equation 8 is approximated as

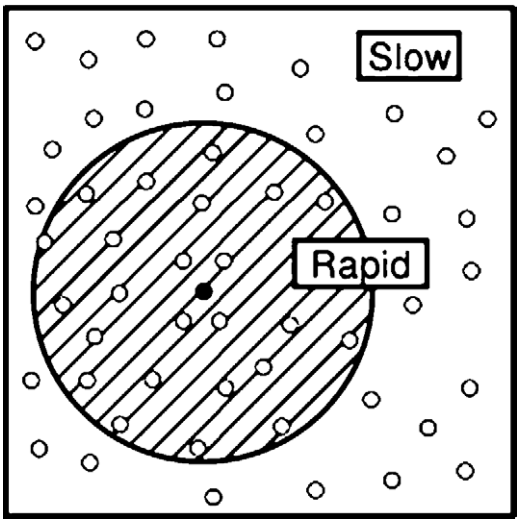

Figure 7. Schematic representation of the MTS scheme. The force on a particle (solid circle) are due to the near (open circles within the hatched area) and far (the other open circles) particles.

$$exp(-i\hat{H}\Delta t/\hbar) = exp(-i\hat{V}\Delta t/2\hbar)exp(-i\hat{K}\Delta t/\hbar)exp(-i\hat{V}\Delta t/2\hbar) + O(\Delta t^3) \tag{10}$$

The $exp(-i\hat{V}\Delta t/2\hbar)$ matrix is multiplied to the wave function vector with $O(M)$ operations because it is diagonal. On the other hand, the $exp(-i\hat{K}\Delta t/\hbar)$ matrix is diagonal in the Fourier space. The conventional spectral method (SM) uses the FFT to transform the wave function alternately between the real and the Fourier spaces, so that both matrices are multiplied in a diagonal form (40). The SM requires $O(M \log M)$ operations because it uses the FFT. On a parallel computer, however, the SM is not an optimal algorithm. By assigning the value of a wave function at each grid point to a node in an array of M processors, the SM algorithm requires $O(\log M)$ time to solve the problem. Also the SM involves considerable communication because of the butterfly communication pattern of the FFT algorithm shown in Fig. 4.

An algorithm called the space-splitting method (SSM) has been developed (41) to solve the time-dependent Schrödinger equation efficiently on a parallel computer (42). The SSM is based on the decomposition of the tridiagonal $\hat{H}$ matrix into direct sums of 2×2 matrices. This decomposition provides an explicit scheme to propagate wave functions in time with $O(M)$ operations. On M parallel processors, the execution time of the SSM algorithm is $O(1)$. In addition, the operations in the SSM are local, requiring only nearest-neighbor grids points to be communicated with little communication overhead (Fig. 8).

Spatial Decomposition

Parallel computing requires decomposing the computation to subtasks and mapping them to processors. For MD simulations, the divide-and-conquer strategy based on spatial decomposition is commonly used (19, 43). The total volume of the system is divided into P subsystems of equal volume,

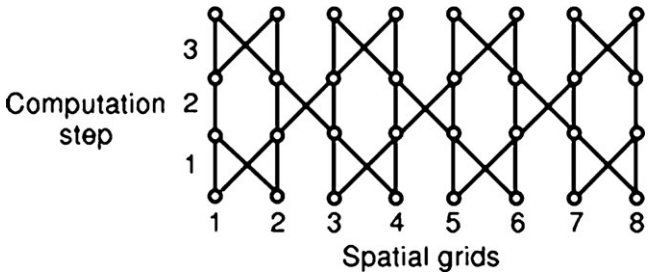

Figure 8. Computation pattern involved in the one-dimensional SSM.

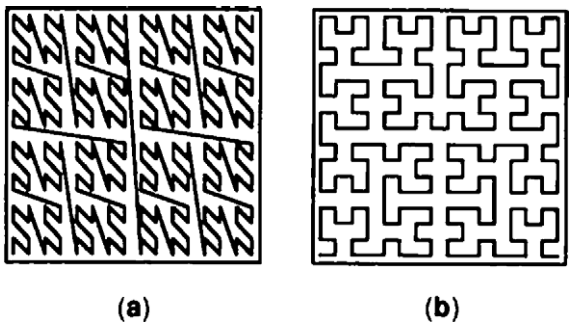

Figure 9. Space-filling curve in two dimensions based on (a) the Z curve and (b) the Hilbert curve.

and each subsystem is assigned to a node in an array of P processors (Fig. 1). The data associated with particles of a subsystem are assigned to the corresponding processor. To calculate the force on a particle in a subsystem, the coordinates of the particles in the boundaries of 26 neighbor subsystems must be copied from the corresponding processor. In the actual code, the message passing to the 26 neighbor nodes is completed in six steps by sending the boundary-particle information to east, west, north, south, and up and down neighbor nodes sequentially (19). The corner and edge boundary particles are copied to proper neighbor processors by forwarding some received boundary particles to other neighbor nodes (19). After updating the particle positions from the time stepping procedure, some particle may move out of its subsystem. These particles, which moved out of a subsystem, are migrated to the proper neighbor nodes. With the spatial decomposition, the computation scales as N/P while communication scales as $(N/P)^{2/3}$.

The implementation of the FMM on parallel computers has been studied extensively (18–23). Suppose the processors are logically organized as a three-dimensional array of $P_x \times P_y \times P_z$. For deeper tree levels, $l \geq \log_2[\max(P_x, P_y, P_z)]$, the calculation of the multipoles is local to each processor, so that the computation scales with N/P. For lower levels, however, the number of cells becomes smaller than the number of processors. Consequently many processors become idle or alternatively they duplicate the same computation, and this computation overhead scales as $\log P$. By making decomposition to be coarse grained ($N >> P$), this $\log P$ overhead can be made negligible.

Load Balancing

Many MD simulations are characterized by irregular atomic distribution (Fig. 1). One practical problem in simulating such irregular systems on parallel computers is that of load imbalance. Suppose that we partition the simulation system into subsystems of equal volume according to the three-dimensional array of processors. Because of the irregular distribution of atoms, this uniform spatial decomposition results in unequal partition of workloads among processors. As a result the parallel efficiency is degraded significantly.

Various approaches have been developed for load balancing such dynamic irregular problems on parallel computers (44). For example, recursive coordinate bisection is one of the widely used methods (44). The load-balancing problem can also be formulated in terms of the more general graph-partitioning problem. Spectral partitioning methods use the lowest nontrivial eigenvectors of the Laplacian matrix of a graph to produce a high-quality partition (44, 45). Multilevel algorithms have been combined with the spectral method to reduce the computational cost (45, 46). By constructing successive coarse approximations of the original graph, these multilevel spectral methods solve static problems efficiently where the cost to perform load balancing is tolerated.

In irregular dynamic simulations, the need for repeated repartitioning necessitates low-overhead load balancers. Most successful among dynamic load balancing schemes are the methods based on spacefilling curves (47). These methods map three-dimensional grid points to a recursively defined self-similar curve, which conserves spatial locality between successive points (Fig. 9). Particles are sorted in a one-dimensional array according to their positions on this space-filling curve, and the array is partitioned into consecutive subarrays of equal size. In a dynamic load-balancer, the partition can be refined incrementally during a simulation based on the load-diffusion concept (48). Another load-balancing scheme uses adaptive curvilinear coordinates to represent partition boundaries (49). Workloads are partitioned with a uniform three-dimensional mesh in the curvilinear coordinate system, which results in curved partition boundaries in the Euclidean space (Fig. 10). The optimal coordinate system is determined to minimize the load imbalance and communication costs. Wavelets allow compact representation of the curved partition boundaries and accordingly speed up the minimization procedure (49).

SUPPORTING TECHNOLOGIES

Although multiresolution algorithms and parallel computing described above are the key enabling technologies for high-performance physics computing, other supporting technologies are also essential for successful computer simulations. These include the management of large and distributed data sets, three-dimensional visualization of multivariate data sets, and knowledge discovery from these data sets (50, 51). For example, hierarchical spatial data structures, a probabilistic approach, and parallel and distributed computing technologies have been combined

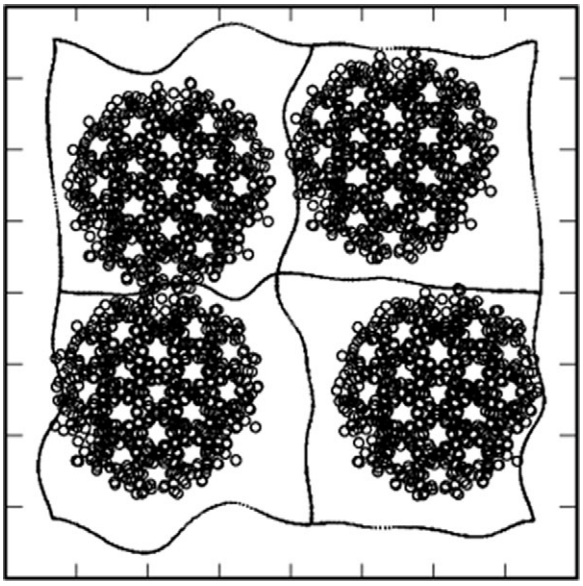

Figure 10. Curved partition boundaries in the Euclidean space used in the curvilinear-coordinate load balancing scheme. Circles represent particles in an MD simulation, and solid curves represent partition boundaries.

to visualize a billion-particle data set interactively in an immersive three-dimensional visualization environment (52). The massive visualization system has been integrated with graph algorithms to automatically discover topological patterns in million-to-billion atom chemical bond networks (53). Parallel and distributed computing technologies have been advanced, so that a Grid (54) of geographically distributed parallel computers can be used to solve challenging scientific problems (55, 56). Valuable information on these topics is found in journals specializing in computational science and engineering techniques (see the Further Reading section).

BIBLIOGRAPHY

1. Emmott, S.; Rison, S. *Towards 2020 Science*; Microsoft Research: Cambridge, UK, 2006.
2. Hockney, R. W.; Eastwood, J. W. *Computer Simulation Using Particles*; Adam Hilger: Bristol, UK, 1988.
3. Allen, M. P.; Tildesley, D. J. *Computer Simulation of Liquids*; Oxford University Press: Oxford, UK, 1987.
4. Frenkel, D.; Smit, B. *Understanding Molecular Simulation*, 2nd ed.; Academic Press: San Diego, CA, 2001.
5. Koonin, S.; Meredith, D. C. *Computational Physics*; Addison-Wesley: Redwood City, CA, 1990.
6. Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; Vetterling, W. T. *Numerical Recipes*, 2nd ed.; Cambridge University Press: Cambridge, UK, 1992.
7. Cook, R. D.; Malkus, D. S.; Plesha, M. E. *Concepts and Applications of Finite Element Analysis*, 3rd ed.; John Wiley & Sons: New York, NY, 1989.
8. Ceperley, D. M.; Kalos, M. H. Quantum many-body problems. In *Monte Carlo Methods in Statistical Physics*, 2nd ed.;Binder, K., Ed.; Springer: New York, NY, 1986.
9. Nielsen, M. A.; Chuang, I. L. *Quantum Computation and Quantum Information*; Cambridge University Press: Cambridge, UK, 2000.
10. Tadmor, E. B.; Phillips, R.; Ortiz, M. Mixed Atomistic and Continuum Models of Deformation in Solids. *Langmuir* 1996, **12**, pp 4529–4534.
11. Car, R.; Parrinello, M. Unified Approach for Molecular Dynamics and Density-Functional Theory. *Phys. Rev. Lett.* 1985, **55**, pp 2471–2474.
12. Warshel, A; Levitt, M. Theoretical Studies of Enzymic Reactions. *J. Mol. Biol.* 1976, **103**, pp 227–249.
13. Broughton, J. Q.; Abraham, F. F.; Bernstein, N.; Kaxiras, E. Concurrent Coupling of Length Scales: Methodology and Application. *Phys. Rev. B* 1999, **60**, pp 2391–2403.
14. Ogata, S.; Lidorikis, E.; Shimojo, F.; Nakano, A.; Vashishta, P.; Kalia, R. K. Hybrid Finite-Element/Molecular-Dynamics/Electronic-Density-Functional Approach to Materials Simulations on Parallel Computers. *Comput. Phys. Commun.* 2001, **138**, pp 143–154.
15. Nakano, A.; Kalia, R. K.; Nomura, K.; Sharma, A.; Vashishta, P.; Shimojo, F.; van Duin, A. C. T.; Goddard III, W. A.; Biswas, R.; Srivastava, D.; Yang, L. H. De Novo Ultrascale Atomistic Simulations on High-End Parallel Supercomputers. *Int. J. High Performance Comput. Appl.* 2007. In press.
16. Brandt, A. Multi-Level Adaptive Solutions to Boundary-Value Problems. *Math. Comput.* 1977, **31**, pp 333–390.
17. Greengard, L.; Rokhlin, V. A Fast Algorithm for Particle Simulations. *J. Comput. Phys.* 1987, **73**, pp 325–348.
18. Salmon, J. K.; Warren, M. S. Skeletons from the Treecode Closet. *J. Comp. Phys.* 1994, **111**, pp 136–155.
19. Nakano, A.; Kalia, R. K.; Vashishta, P. Multiresolution Molecular Dynamics Algorithm for Realistic Materials Modeling on Parallel Computers. *Comput. Phys. Commun.* 1994, **83**, pp 197–214.
20. Jackson, J. D. *Classical Electrodynamics*, 2nd ed. John Wiley & Sons: New York, 1975.
21. de Leeuw, S. W.; Perram, J. W.; Smith, E. R. Simulation of Electrostatic Systems in Periodic Boundary Conditions. I. Lattice Sums and Dielectric Constant, *Proc. Roy. Soc. Lond. A* 1980, **373**, pp 27–56.
22. Toukmaji, A. Y.; and Board, J. A. Ewald Summation Techniques in Perspective: A Survey. *Comput. Phys. Commun.* 1996, **95**, pp 73–92.
23. Ogata, S.; Campbell, T. J.; Kalia, R. K.; Nakano, A.; Vashishta, P.; Vemparala, S. Scalable and Portable Implementation of the Fast Multipole Method on Parallel Computers. *Comput. Phys. Commun.* 2003, **153**, 445–461.
24. Hohenberg, P.; Kohn, W. Inhomogeneous Electron Gas. *Phys. Rev.* 1964, **136**, pp B864–B871.
25. Kohn, W.; Sham, L. J. Self-Consistent Equations Including Exchange and Correlation Effects. *Phys. Rev.* 1965, **140**, pp A1133–A1138.
26. Kohn, W. Density Functional and Density Matrix Method Scaling Linearly with the Number of Atoms. *Phys. Rev. Lett.* 1996, **76**, pp 3168–3171.
27. Goedecker, S. Linear Scaling Electronic Structure Methods. *Rev. Mod. Phys.*, 1999, **71**, pp 1085–1123.
28. Yang, W. Direct Calculation of Electron Density in Density-Functional Theory. *Phys. Rev. Lett.* 1991, **66**, 1438–1441.
29. Shimojo, F.; Kalia, R. K.; Nakano, A.; Vashishta, P. Embedded Divide-and-Conquer Algorithm on Hierarchical Real-Space Grids: Parallel Molecular Dynamics Simulation Based

on Linear-Scaling Density Functional Theory. *Comput. Phys. Commun.* 2005, **167**, pp 151–164.

30. Ryckaert, J. P.; Ciccotti, G.; Berendsen, J. C. Numerical Integration of the Cartesian Equations of Motion of a System with Constraints: Molecular Dynamics of N-alkanes *J. Comput. Phys.* 1977, **23**, pp 327–341.
31. Space, B.; Rabitz, H.; Askar, A. Long Time Scale Molecular Dynamics Subspace Integration Method Applied to Anharmonic Crystals and Glasses. *J. Chem. Phys.* 1993, **99**, pp 9070–9079.
32. Streett, W. B.; Tildesley, D. J.; Saville, G. Multiple Time Step Method in Molecular Dynamics. *Mol. Phys.* 1978, **35**, pp 639–648.
33. Tuckerman, M. E.; Berne, B. J.; Martyna, G. J. Reversible Multiple Time Scale Molecular Dynamics. *J. Chem. Phys.* 1992, **97**, pp 1990–2001.
34. Skeel, R. D.; Zhang, G.; Schlick, T. A Family of Symplectic Integrators. *SIAM J. Sci. Comput.* 1997, **18**, pp 203–222.
35. Zhang, G.; Schlick, T. LIN: A New Algorithm to Simulate the Dynamics of Biomolecules by Combining Implicit-Integration and Normal Mode Techniques. *J. Comput. Chem.* 1993, **14**, pp 1212–1233.
36. Nakano, A. Fuzzy Clustering Approach to Hierarchical Molecular Dynamics Simulation of Multiscale Materials Phenomena. *Comput. Phys. Commun.* 1997, **105**, pp 139–150.
37. Truhlar, D. G.; Garrett, B. C.; Klippenstein, S. J. Current Status of Transition-State Theory. *J. Phys. Chem.* 1996, **100**, pp 12771–12800.
38. Voter, A. F.; Montalenti, F.; Germann, T. C. Extending the Time Scale in Atomistic Simulation of Materials. *Annu. Rev. Mater. Res.* 2002, **32**, pp 321–346.
39. Kumar, V.; Grama, A.; Gupta, A.; Karypis, G. *Introduction to Parallel Computing*, 2nd ed.; Addison-Wesley: Harlow, UK, 2003.
40. Feit, M. D.; Fleck, J. A.; Steiger, A. Solution of the Schrödinger Equation by a Spectral Method. *J. Comput. Phys.* 1982, **47**, pp 412–433.
41. de Raedt, H. Product Formula Algorithms for Solving the Time-Dependent Schrödinger Equation. *Comput. Phys. Rep.* 1987, **7**, pp 1–72.
42. Nakano, A.; Vashishta, P.; Kalia, R. K. Massively Parallel Algorithms for Computational Nanoelectronics Based on Quantum Molecular Dynamics. *Comput. Phys. Commun.* 1994, **83**, pp 181–196.
43. Rapaport, D. C. *The Art of Molecular Dynamics Simulation*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2004.
44. Fox, G. C.; Williams, R. D.; Messina, P. C. *Parallel Computing Works*; Morgan Kaufmann: San Francisco, CA:, 1994.
45. Barnard, S. T.; Simon, H. D. Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems. *Concurrency* 1994, **6**, pp 101–117.
46. Hendrickson, B.; Leland, R. An Improved Spectral Load Balancing Method. *Proc. Sixth SIAM Conf. Parallel Processing for Scientific Computing*; SIAM, Philadelphia, PA, 1993;pp. 953–961.
47. Kaddoura, M.; Ou, C.-W.; Ranka, S. Partitioning Unstructured Computational Graphs for Nonuniform and Adaptive Environments. *IEEE Parallel Distrib. Tech.* 1996, **3**, pp 63–69.
48. Cybenko, G. Dynamic Load Balancing for Distributed Memory Multiprocessors. *J. Parallel Distrib. Comput.* 1989, **7**, pp 279–301.
49. Nakano, A. Multiresolution Load Balancing in Curved Space: The Wavelet Representation. *Concurrency: Practice Exper.* 1999, **11**, pp 343–353.
50. Zabusky, N. J. Computational Synergetics. *Phys. Today* 1984, **37**, pp 36–46.
51. Chen, J. X.; Nakano, A. High-Dimensional Data Acquisition, Computing, and Visualization. *IEEE Comput Sci Eng.* 2003, **5**, pp 12–13.
52. Sharma, A.; Nakano, A.; Kalia, R. K.; Vashishta, P.; Kodiyalam, S.; Miller, P.; Zhao, W.; Liu, X.; Campbell, T. J.; Haas, A. Immersive and Interactive Exploration of Billion-Atom Systems. *Presence: Teleoperators Virtual Environ.* 2003, **12**, pp 85–95.
53. Zhang, C.; Bansal, B.; Branicio, P. S.; Kalia, R. K.; Nakano, A.; Sharma, A.; Vashishta, P. Collision-Free Spatial Hash Functions for Structural Analysis of Billion-Vertex Chemical Bond Networks. *Comput. Phys. Commun.* 2006, **175**, pp 339–347.
54. Foster, I.; Kesselman, C. *The Grid 2: Blueprint for a New Computing Infrastructure*; Morgan Kaufmann: San Francisco, CA, 2003.
55. Shirts, M.; Pande, V. S. Computing—Screen Savers of the World Unite. *Science* 2000, **290**, pp 1903–1904.
56. Takemiya, H.; Tanaka, Y.; Sekiguchi, S.; Ogata, S.; Kalia, R. K.; Nakano, A.; Vashishta, P. Sustainable Adaptive Grid Supercomputing: Multiscale Simulation of Semiconductor Processing across the Pacific; *Proc. of Supercomputing 2006 (SC06)*; IEEE Computer Society: Los Alamitos, CA, 2006.

Reading List

Computer Physics Communications; Elsevier: Amsterdam.

Journal of Computational Physics; Academic Press: New York.

IEEE Computational Science & Engineering; IEEE Computer Society: Los Alamitos, CA.

AIICHIRO NAKANO
Department of Computer Science University of Southern California, 3651 Watt Way, VHE 610, Los Angeles, CA 90089–0242