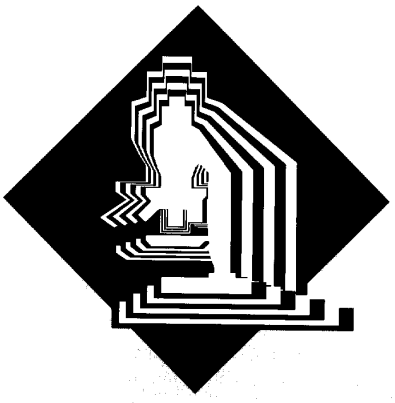

Modeling Biomolecules: Larger Scales, Longer Durations

John A. Board Jr.
Duke University

Laxmikant V. Kalé, Klaus Schulten, and Robert D. Skeel
University of Illinois at Urbana-Champaign

Tamar Schlick
New York University and Howard Hughes Medical Institute

The molecules of life are big, complex, and dynamic. Collaborating in multidisciplinary groups that draw on chemistry, physics, mathematics, and other fields, computational scientists are beginning to simulate systems of large biomolecules interacting in time. Obstacles are formidable; the potential benefit, vast.

THE FAST GROWTH OF MOLECULAR MODELING AS A research tool in biology and medicine has been tightly coupled to the advent of the supercomputer and to advances in applied and computational mathematics over the past decade. Three features characterize the progress made to date: bigger molecular systems described in atomic detail, longer simulation time scales, and more realistic representations of interatomic forces. With these improvements, molecular modeling by computer has given us many insights into the relationship between structure and function of biopolymers and drugs.[1,2] Researchers now find it indispensable for structure refinement.[3]

Still, the state of the art in molecular modeling leaves much room for more progress:

- Simulations of biopolymers must be extended from the current few thousand atoms to systems of 100,000 or more atoms.
- The time scale of simulations in molecular dynamics must reach beyond the present nanosecond horizon to describe longer processes of biologically relevant duration. Examples are substrate binding, enzyme reactions, and the folding of proteins into their native form.
- Descriptions of interatomic forces must be improved to include such factors as atomic polarizabilities and to combine molecular modeling and quantum chemical calculations.

1070-9924/94/$4.00 © 1994 IEEE

Molecular dynamics simulations must speed up by several orders of magnitude to allow the increases in system size and time scale that researchers need. To achieve this speed and to improve the quality of force-field representations, scientists will exploit advances in processor speed and in numerical and parallel algorithms. The increasingly important role of molecular modeling in biology and medicine has already led to the formation of multidisciplinary teams to provide the needed knowledge of hardware, software, mathematics, and science.

Our groups are actively developing computer programs with improved schemes for numerical integration, parallelization, and efficient and scalable evaluation of electrostatic force fields. These programs are targeted for a variety of scalable shared-memory and distributed-memory machines, including networks of workstations. Describing this current research and its background will give readers a taste of the challenges ahead in computational biology.

Modeling macromolecules

Computational structural biology has been confined so far to the study of only the smallest functional units in living cells: small and medium-sized proteins or protein complexes, segments of DNA, minute patches of membranes, and biomolecules solvated by minuscule droplets of water. Most simulations have dealt with less than 10,000 atoms. To be more realistic, simulations of molecular dynamics need to include more of the natural environment, like water or membranes, that surrounds the molecule in question. This often increases system size to about 100,000 atoms. Moreover many biological functions, even simple ones, involve aggregates of biopolymers; for example, polymers with proteins as subunits. To describe supramolecular structures like complexes of regulatory proteins with DNA, the aggregations of proteins that form muscle strands, or protein coats protecting the genetic material of a virus, requires simulations of up to a million atoms.

Multidisciplinary teams are combining knowledge of hardware, software, mathematics, and science.

Experimentalists have had dramatic success deciphering the structures of some large-scale biomolecular systems observationally. Examples from the past 10 years include the myosin–actin complex in muscle strands, the F_1 fraction of ATPase (which synthesizes the important cellular energy source ATP or converts it into a proton gradient across the cell wall), the TATA-box binding protein (which plays a crucial role in gene expression), and the protein coats of virus particles. This observational progress in determining certain structures opens up opportunities for computationally determining how those structures function.

On the other hand, many important biopolymer structures are inherently too flexible and disordered to be solved by available observational means such as crystallography. Solving their structures will likely require large-scale modeling efforts in concert with observation. Computational structural biologists eventually hope to predict entire structures from first principles. The structure of a protein, for instance, should be computable from its sequence of amino acids. The principles linking amino acid sequences or, equivalently, gene sequences and protein structures are often referred to as the second part of the genetic code. Once these principles are cast into computer programs—if that is achievable at all—the rapidly increasing database on the genomes of humans and other organisms could be used for structure prediction. The opportunities resulting from this would be far-reaching.

This goal may long remain elusive. Meanwhile, though, researchers will try to model particular proteins which they judge to be amenable to computation. They will also begin to design proteins with new properties, for example, enzymes that can remediate toxic materials.

At present molecular modeling is carried out mainly in batch-job mode. Researchers thus face long delays between having an idea for structure building and being able to view the resulting structures. The process often involves extremely lengthy searches for optimal geometries, made longer by the lack of interactive programs that would let the researcher guide the search. The design of new biopolymer structures would benefit tremendously from interactive simulations. This would require, however, much greater simulation speed, realized through better algorithms and the pooling of parallel processors. Such advances need to be combined with visualization and analysis tools.

A few examples will illustrate the progress now being made in macromolecular modeling.

Simulating membranes and membrane proteins

Biopolymers function in two environments, water and membranes, which in turn affect structure. Water has been studied extensively

and we have suitable, if not perfect, models of it to use in biomolecular simulations. Membranes, however, have resisted modeling attempts. Why? Suppose we want a patch of membrane to serve as an environment for an embedded protein. If we are modeling the protein bacteriorhodopsin embedded in a membrane of 1-palmitoyl-2-oleoyl-*sn*-glycero-3-phosphotidylcholine, or POPC (Figure 1), we need to model a cube of membrane measuring about 100 angstroms on each side. Moreover, the properties of lipid bilayers of membranes are determined by solvation and long-range electrostatic interactions. This means that we must add water and that the Coulomb forces of attraction and repulsion between charged particles cannot be cut off. As a result, membrane studies require simulations of large size (N = 30,000 atoms) and long duration (t = 1 nanosecond). In the world of molecules, where vibrations (and, hence, time steps for integrating the equations of motion) are measured in femtoseconds or picoseconds, a nanosecond is actually quite a while.

The first simulation of a lipid bilayer patch by one of our research groups (at Illinois) covered a total time span of 263 picoseconds and consisted of 200 molecules of POPC on an 85 × 100–angstrom rectangle, with 5,483 water molecules covering the lipid head groups. The system was equilibrated in both the gel and the liquid-crystal phases. Structural and dynamic properties like order parameter profiles for nuclear magnetic resonance, distribution of molecular groups, and self-diffusion coefficients were determined.[4] The simulations included over 25,000 atoms, ran on a self-built MIMD-type parallel computer with 60 processors, and required about two years of uninterrupted computing. We have extended the simulations to a patch of a DLPE (dilauryl-phosphatidylethanolamine) membrane with 32,000 atoms. In the DLPE simulation we determined, among other things, the potential experienced by charged particles when crossing the membrane. This potential depends crucially on the structure and distribution of water at the membrane surfaces and requires a faithful description of all electrostatic forces in the system. The calculation became feasible only when we incorporated a method called the fast multipole algorithm into our programs, since this approach yields only an $O(N)$ computational complexity.

Presently, the DLPE-membrane model is used to study the function of phospholipase A_2. This enzyme associates with membranes and digests the lipid molecules by cleaving their *sn*-2 ester bonds. It occurs in the venom of bees and rattlesnakes, but also plays a role in the digestive tract and in body fluids. Human synovial phospholipase A_2 has attracted pharmacological interest since it is involved in inflammation processes. Our present goal is to understand how phospholipase A_2 binds to membrane surfaces and to explain how the binding dramatically enhances the enzyme's effectiveness.

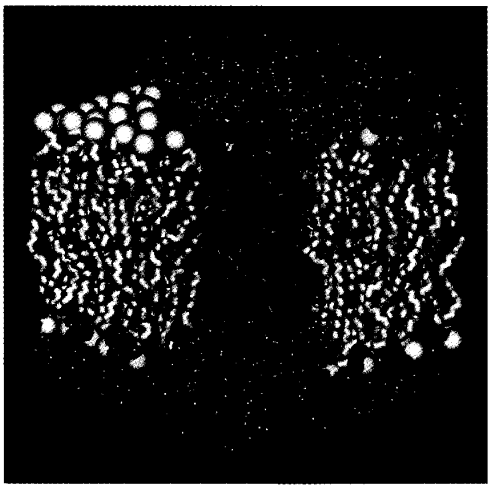

Figure 1. The protein bacteriorhodopsin embedded in a patch of a POPC membrane. (Courtesy Helmut Heller)

Protein–DNA interactions

Hormone receptors are proteins that interact with DNA to regulate the transcription of genes into proteins. In 1991 the structure of a key component of the glucocorticoid hormone receptor (GR) complexed with DNA was solved.[5]

This protein belongs to a class of nuclear receptors that has been implicated in a variety of human cancers of reproductive tissues. The GR *dimerizes*, or joins with another GR molecule, when it binds to DNA; each monomer subunit of the receptor forms specific interactions in what is called the major groove of DNA, and can recognize local DNA sequences.

Our simulation of this system (see Figure 2) included two GR protein units, a DNA segment, and an ellipsoid of water surrounding the proteins and DNA; altogether about 13,000 atoms. Tom Bishop in our group at Illinois did the simulations on a 64-Transputer T805 parallel machine (Parsytec's GC–el64). The simulations started from the structure previously determined[5] and mutated the DNA into a biologically more relevant sequence. This modeling revealed how the proteins "read" local DNA sequences and that the proteins bend the DNA and unwind the DNA helix to a significant degree. The simulations pave the way to study other regulatory proteins involved in ma-

lignancies. A prime candidate for further study is the protein p53, implicated in a very wide class of human cancers.

Muscle strand

The primordial dynamic system in higher organisms is muscle. Its smallest functional unit, the sarcomer, consists of actin and myosin filaments gliding past each other to contract or extend the muscle. Figure 3 presents one component of muscle fibers, the F-actin strand, which provides the scaffolding along which the myosin filaments move.[6] As Figure 3 shows, the F-actin strand consists of many units, the so-called G-actin proteins. Presently, Willy Wriggers in our group at Illinois carries out simulations of isolated G-actin and of G-actins that form a repeat unit (about 40,000 atoms) of the F-actin strand. The goal is to understand the aggregation of G-actin into F-actin and the ensuing flexibility of F-actin strands during muscle action. This figure illustrates a new role for biopolymer simulations, namely the modeling of biomolecules that take on new structure and properties when joined together in aggregates. Another protein aggregate we study is the coat of poliovirus (see below). Protein polymerization may be a key factor in mad cow disease, a poorly understood disorder of the central nervous system. This disease is of great concern because of its possible wide impact on public health. Molecular dynamics simulations can contribute to our understanding of it.

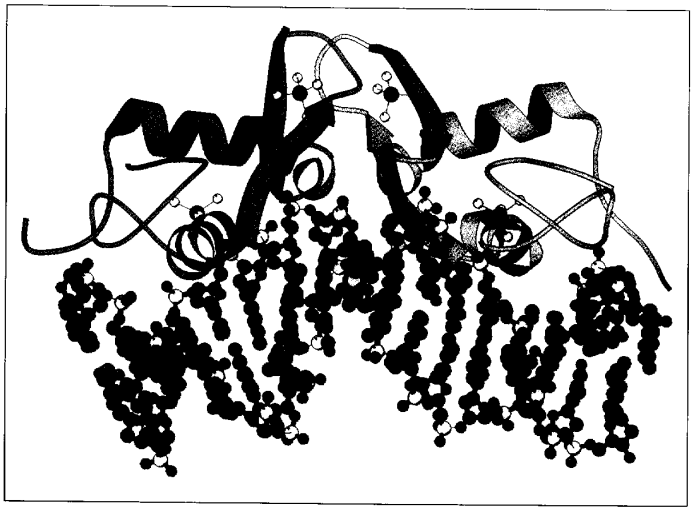

Figure 2. Simulation of the DNA-binding domain of a dual-molecule glucocorticoid receptor (upper) complexed with DNA (lower). The receptor bends and unwinds the DNA, which may help explain its function. (Arrows denote certain sequences of amino acids.) (Courtesy Tom Bishop)

Simulating how viruses attack

The determination of the atomic structures of the coats of virus particles is a triumph of modern crystallography. The most intensely studied viruses belong to the picornavirus family and are connected with poliomyelitis, the common cold, hepatitis A, and foot-and-mouth disease. Crystallographers have provided nearly complete structures of the coats, or capsids, for several picornaviruses and related viruses. Knowing these structures makes possible exciting molecular dynamics studies of how viruses infect the body, what determines the level of virulence, and how infections can be prevented through vaccination or drugs. However, remaining structural uncertainties and system size pose enormous technical difficulties for modeling. The 60 subunits forming the coat of polioviruses encompass 399,000 mostly well resolved atoms, to which about 200,000 polar hydrogen atoms must be added. Water must also be added, since solvent effects are expected to play important roles. Altogether these require ambitious simulations of over one million atoms; recently our group embarked on the first phase of this project.

Supercoiled DNA: Energetics and dynamics

The folding and knotting of topologically circular DNA is an important aspect of many fundamental biological processes such as replication and recombination. Its study spans biology, chemistry, and mathematics, and forms a computational challenge for modelers.[7] *Supercoiling* is a higher level of folding of the DNA that involves bending and twisting about the global helix itself. Little in detail is known about its structural and dynamical aspects, much less about the associated enzymes that affect DNA topology. Thus, systematic data are critically needed to interpret the effects of supercoiling on fundamental biological functions. Simulation work is challenging because the systems are very large—thousands of base pairs—and the time scales are very long. One of us (Schlick), working with W.K. Olson, has devised a model that macroscopically represents DNA as essentially a charged elastic ribbon, and combines this with large-time-step simulations to capture global folding and many interesting related processes.[8] These include the knotting of DNA, salt effects on DNA mobility, and the profound effects of solvent on the thermal fluctuations of DNA (see Figure 4). These studies add detailed information to low-resolution experimental data obtained by gel electrophoresis, electron microscopy, knotting recombination experiments,

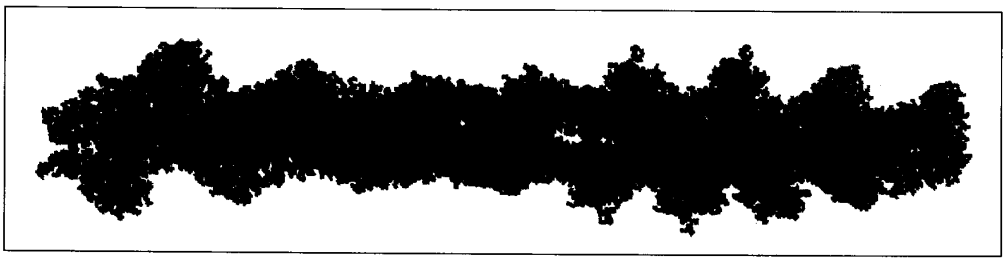

Figure 3. Simulations help explain the flexibility of muscle strands. This piece of the F-actin filament of muscle contains 17 "repeat units" (different colors) of a protein G-actin, together about 40,000 atoms. X-ray fiber diffraction supplied the basic structure; computers refined it.[6] (Courtesy Willy Wriggers)

and light scattering, and suggest new lines of experimentation.

However, much progress must be made in modeling to extend such observations to the all-atom level necessary for investigating many important details of protein-DNA interaction. This extension is a focus in our collaboration. We are working to determine an all-atom structure for several dozen nucleotides. The results will be plugged into the macroscopic supercoiled-DNA model in the form of a rigid restraint on a part of the curve that is later subjected to relaxation under given topological strain. Such improved modeling and long simulations should provide further insight into the topological and geometrical changes induced by proteins, as well as the covalent binding of various chemical mutagens that alter superhelicity and therefore biological activity.

Long-time integration methods

We have emphasized that biomolecular models simulate not just a *structure* but a *process*. Life is not static, nor are the molecules that make it up, so computer simulations of them must model changes over time. Molecular dynamics, or MD, is a useful tool for sampling a range of biomolecular structures and obtaining insight into how they change over the durations relevant in the microscopic domain. The idea is to solve Newton's second law of motion

$$M\dot{V}(t) = -\nabla E(X(t)), \quad \dot{X}(t) = V(t) \qquad (1)$$

for the atomic positions X and velocities V as a function of time. Above, the dots denote differentiation with respect to time, M is a diagonal mass matrix, and $E(X)$ is the potential energy. The function $E(X)$ is obtained by fitting parameters semiempirically to thermodynamic and structural properties of representative molecules while incorporating some quantum-mechanical calculations. This function models bonded interactions, electrostatic forces, and the van der Waals attraction and repulsion. The initial positions typically come from observational data, such as those available from determined crystal structures, and initial velocities are assigned randomly to yield a Boltzmann distribution.

From MD, structural and dynamical information can be obtained for many interesting quantities, such as the fluctuations of dihedral angles between sets of bonded atoms, average energies, conformational distributions, large-scale protein bending and other collective motions of subgroups, rates of conformational change, structural rearrangements due to solvent, and correlations between various geometric parameters. However, MD suffers from today's limited simulation times. Therefore, the sampling is typically far from completely representative, and it is difficult to have a global perspective on

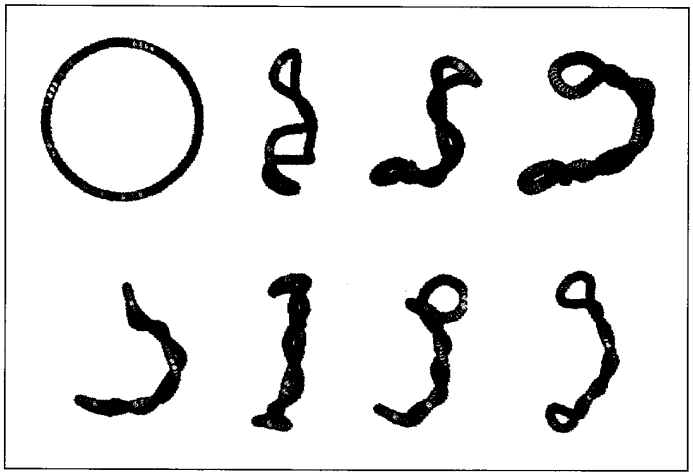

Figure 4. Supercoiling: snapshots from a 2,000-base-pair DNA simulation. Selected frames from a DNA trajectory model the relaxation of closed circular DNA in standard solvent and sodium salt conditions from the circle, which is torsionally stressed, to the interwound structure, which is at a potential-energy minimum.

many measurements and observations. Progress is still required to extend our results beyond the nanosecond horizon to capture the many realizable conformational states and transitional pathways of biomolecules.

For these problems, the trajectories depend sensitively on the initial conditions and the parameters. Experiments show that the size of a perturbation doubles every picosecond or so until it reaches the molecular scale.[9] Thus, the kind of information obtained from a molecular dynamics simulation is statistical in nature. In particular, it involves energetically favorable changes in and fluctuations about the structure of the biomolecule.

New algorithms show promise for lengthening the time step in molecular dynamics.

In many situations, such as the modeling of a protein without explicit solvent molecules, it is very important to include effects of the environment. This is most easily done by adding simple terms for frictional and random forces to the systematic force described in the Newton equation. This results in the Langevin equation, which is widely used in biomolecular simulations to mimic a thermal reservoir.[10] This equation takes the form

$$M\dot{V}(t) = -\nabla E(X(t)) - \gamma MV(t) + R(t),$$
$$\dot{X}(t) = V(t)$$

where R is a random force satisfying the conditions

$$\langle R(t)\rangle = 0, \quad \langle R(t)R(t')^T\rangle = 2\gamma k_B T M\delta(t-t')$$

Here γ is the damping constant, k_B is Boltzmann's constant, T is the temperature, δ is the Dirac delta symbol, and angle brackets denote the expected (that is, average) value.

The standard procedures for numerically solving the Newtonian equations of motion (Equation 1) are *explicit*.[1,9] While simple to formulate, explicit schemes severely restrict the size of the integration time step: Δt must be at least as small as the most rapid vibrational mode. This generally limits Δt to the femtosecond (10^{-15} s) range and the trajectory time to the picosecond to nanosecond (10^{-12} to 10^{-9} s) range. Since key conformational changes—rotations about chemical bonds, for instance—occur on time scales of anywhere from a picosecond to one hundred seconds, many enhanced sampling techniques combining Monte Carlo, high-temperature MD, free-energy, global optimization, and other methods have been developed. In addition, researchers have devoted considerable effort to devising methods that allow a longer integration time step to be used. However, several formidable difficulties arise: (1) reality of the results, if some local resolution is sacrificed, (2) computational cost of suitable long-time-step schemes, and (3) estimates of accuracy.

In MD we are dealing with nontraditional ways of assessing accuracy. Appropriate mathematical theories of error analysis and long-time stability are not yet available for ordinary differential equations of the type that arise in chemistry—multidimensional, highly nonlinear, frequently chaotic systems. Most error analyses are performed on systems for which accurate trajectories are easy to calculate. From a practical point of view, in MD we would like to merge both qualitative and quantitative behavior of biological systems. Therefore, we can imagine that different models, in combination with different integration or propagation methods, could be designed to address different aspects of dynamical problems for macromolecules. Essentially, the appropriateness of an algorithm should be tightly coupled to the scientific questions that are being asked of it. Some compromise might therefore be necessary between, on one hand, modeling the detailed dynamics of a system correctly but missing its global features and, on the other, obtaining a more global view of conformation space at the expense of detailed resolution.

Most popular among standard integration methods for the Newton equation is the Verlet (or leapfrog) scheme. It has excellent long-time behavior, because it is stable but nondamping for linear problems—if Δt is suitably restricted. In addition the Verlet scheme is *symplectic* (or canonical).[11] This attractive area-preserving property of phase space is a feature possessed by the dynamics of Hamiltonian systems, and has recently been explored by mathematicians and physicists in a variety of applications, including molecular dynamics.[12] While symplectic integrators may be implicit or explicit, most computational chemists have adopted the Verlet explicit method because of its simplicity. In explicit methods, the solution at the end of the time step is obtained by performing operations on quantities available at the beginning of each time step. For example, in one form of the Verlet method we begin a step at time $n\Delta t$ with positions X^n, velocities V^n, and forces $F^n = -\nabla E(X^n)$ and advance the solution as follows:

$$V^{n+1/2} = V^n + \tfrac{1}{2}\Delta t \boldsymbol{M}^{-1} F^n$$
$$X^{n+1} = X^n + \Delta t V^{n+1/2}$$
$$F^{n+1} = -\nabla E\left(X^{n+1}\right)$$
$$V^{n+1} = V^{n+1/2} + \tfrac{1}{2}\Delta t \boldsymbol{M}^{-1} F^{n+1}$$

With implicit integrators, the final solutions are functions of both the initial and final variables for positions, velocities, and so forth. Therefore, coupled nonlinear equations must usually be solved to determine the results at each time step. Generally, the explicit versions involve very simple algorithms that use little memory, while implicit methods involve more complex algorithms but may be more suitable for treating systems with disparate time-scale dynamics.

To reduce the severity of the requirement of very short time steps to keep the Verlet algorithm stable, the favored approach has long been a constrained formulation for the bond-length and possibly bond-angle motion. However, this increases the time step only modestly. Another viable approach, variously called multiple time steps, distance classes, or multiple time scales, uses different time steps for different interactions. These methods certainly provide added speedup but do not fundamentally attack the more global problem of enhanced sampling and much longer time frames.

Numerical analysts typically resort to implicit numerical schemes to handle problems with multiple time scales. For example, if we replace the explicit force evaluation $F^{n+1} = -\nabla E(X^{n+1})$ in the Verlet method by

$$F^{n+1} = -\nabla E\left(X^{n+1} + \tfrac{1}{4}(\Delta t)^2 \boldsymbol{M}^{-1} F^{n+1}\right)$$

we get an implicit symplectic method, which can be shown to be equivalent to the popular implicit midpoint method. It has the generally high computational cost of solving coupled nonlinear equations for the unknown forces, eased somewhat by the fact that the Jacobian matrix is symmetric and, for reasonably small Δt, positive definite. In fact, the nonlinear system can be formulated as a minimization problem. The value of such a method is that for linear problems, at least, it is stable but nondamping regardless of how large Δt is—in fact, the implicit midpoint method exactly conserves total energy for linear problems.

However, the implicit midpoint method is only marginally stable. When it is applied to mildly nonlinear dynamics problems in structural mechanics, instabilities have been observed unless the time step is severely limited. And generally, there is some concern that longer time steps are prone to instability. Therefore, it may be necessary to adjust the MD model to counteract numerical artifacts. In particular, it may be desirable to increase the damping constant γ, especially if we are using no damping at all.

Moreover, the success of implicit schemes with large Δt often relies on the assumption that the rapidly varying motion does not contribute significantly to the major motion of interest. This assumption is false in molecular systems, where vibrational modes are intimately coupled and the cooperative motion among the correlated vibrational modes may require energy transfer. Indeed, the freezing of the bond-angle motion was found long ago to have severe effects on the overall dynamics of molecular systems.

Code for exact Coulomb forces is easy to write; expensive to run.

One possible way to exploit a highly stable implicit scheme is to combine it with another procedure that more adequately resolves the high-frequency motion. Our idea in the Langevin/implicit-Euler/normal-modes method, or LIN,[13,14] is to obtain an approximation to the solution which captures most of the oscillations within that time step by using normal-mode analysis based on a linearization of the Langevin dynamics equations with an approximate Hessian $\boldsymbol{H}_b$. We then correct it with a large-time-step integration by the implicit (or backward) Euler scheme, which is one of the simplest implicit integration schemes with high stability.

In this way, the rapidly varying motion (X_b, V_b) is captured by the normal-mode (NM) analysis, while the slowly varying motion (Z, W) is resolved by the correction.

Numerical tests with LIN have been performed for butane and for deoxycytidine. To make the minimization problem efficient, a truncated-Newton method is used.[15] Current efforts focus on development of LIN for biomolecular applications. This involves careful implementation of the NM details. Since the approximate Hessian in the linearized Langevin equation is system-dependent, systematic procedures for protein and nucleic acid systems must be developed regarding the choice of the approximation of $\boldsymbol{H}_b$ in the linearized Langevin

equation, the frequency of the normal-mode decomposition, and efficient implementations for large systems.

Rapid evaluation of electrostatics

As we just described, researchers are finding ways to reduce the *number* of force evaluations necessary by using information in the best way. What are we to do, though, when *each* set of force evaluations still takes an inordinately long time?

Among the many types of forces present in simulations of biomolecules, the *nonbonded* forces (Coulomb and van der Waals) are both the easiest to characterize and the most time-consuming to compute. It is trivial to write code to solve exactly for the Coulomb force (and the van der Waals force to a very good approximation) by summing over all nonbonded pairs of atoms in the system, but it is costly to evaluate these forces. The runtime of this *direct summation* procedure grows with the square of the number of atoms. Employing this technique on large numbers of particles requires enormous simulation time, even on large, dedicated, parallel machines. The direct summation method is limited to relatively small systems with at most a few tens of thousands of atoms.

An alternative to direct summation that is common to MD programs today is to approximate the Coulomb force by truncating its effect at a certain radius (typically 12 angstroms or so). The force evaluation complexity then grows only linearly with the size of the system, but accuracy is greatly reduced.

A class of new algorithms developed over the last decade combines the best features of both methods. The best *multipole-accelerated algorithms* and related methods have runtimes that grow linearly with the size of the system, while still including contributions from all atoms. Perhaps the best known of these algorithms is the *fast multipole algorithm* of Greengard and Rokhlin;[16] the tree code of Barnes and Hut shares many features with the FMA.[17]

Such algorithms approximate the force that a group of distant particles exerts on a particle P by a single function representing the entire distant group, instead of by having P interact individually with each far-away particle (see Figure 5). The multipole-accelerated algorithms, including the FMA, use a truncated multipole expansion to represent the distant group of atoms.

The multipole expansion of a distant group of particles is given by

$$\Phi(\mathbf{r}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \frac{M_l^m}{r^{l+1}} Y_l^m(\theta,\phi), \text{ where} \quad (2)$$

$$M_l^m = \sum_{i=1}^{k} q_i \rho_i^l Y_l^{*m}(\alpha_i,\beta_i) \quad (3)$$

For a given distant box (shaded box in Figure 5) containing k particles, the position of the ith particle is given in spherical coordinates by $(\rho_i, \alpha_i, \beta_i)$. $\Phi(\mathbf{r})$ represents the aggregate potential field due to all k particles at any point $\mathbf{r} = (r, \theta, \phi)$ outside the given box. The infinite multipole expansion describes the potential field exactly. For practical implementations, rigorous error bounds have been derived to show how the accuracy of the force and potential depend on the number of terms retained. The series is summed over the legal values of the order l ($0 \le l \le \infty$) and degree m ($-l \le m \le l$) of the spherical harmonic function $Y_l^m(\theta, \phi)$.

The various multipole-accelerated algorithms differ essentially only in the details of determining which groups of particles are sufficiently "well separated" to interact via the series approximation. In most cases, spatial decomposition is used to separate the simulation region into increasingly fine subregions; various rules are used to determine which subregions meet the "well separated" criteria. Some of the rules have been shown to result in linear-time algorithms for evaluating the Coulomb force to arbitrarily high accuracy. For all the algorithms, some particles (typically those in adjacent subregions at the finest level of spatial decomposition) fail the well-separatedness test; the potential and forces between these nearby atoms

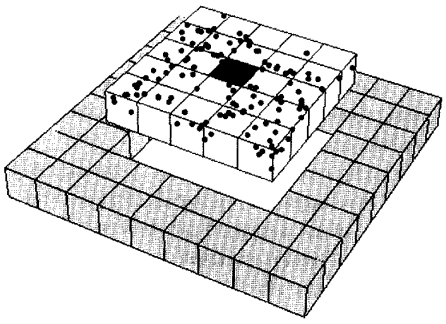

Figure 5. Multipole-accelerated algorithms allow a particle P in the central black box to interact with distant boxes (shaded) via truncated multipole expansions. Interactions with particles in nearby (white) boxes must be calculated individually.

are computed explicitly (white boxes in Figure 5).

All of the multipole-accelerated algorithms are much faster than all-pairs summation. In Figure 6, which shows uniprocessor timings on a fast workstation (Hewlett-Packard 735/125), three algorithmic variants are compared with direct (all-pairs) summation. The PMTA curve is for the parallel multipole tree algorithm, a method very similar to that of Barnes and Hut. The "enhanced PMTA" hybrid scheme combines features of the FMA and PMTA. These algorithms are described in more detail elsewhere.[17] Parameters for the three schemes have been adjusted to give similar accuracy, five to six significant figures in computations of potential and four to five significant figures in force.

We can speed up force evaluations further by parallelizing the FMA code. The granularity of the FMA is such that it can run successfully on a wide range of parallel machine types, from networked workstations to tightly coupled machines such as the Cray T3D and Kendall Square KSR1. Our parallel approach exploits spatial decomposition, which is an integral part of the FMA. We have run the code on 128 processors, and scaling indicates that for large simulations we can use up to 512 processors with reasonable efficiency.

Using an FFT formulation for the multipole manipulations further accelerates the potential and force computations. This approach recognizes that evaluating the appropriate multipole expansion expressions resembles a convolution operation on the arrays of coefficients of the expansions. The curves in Figure 6 reflect this enhancement. Additional speed can be had from observing that the potential (and force) between groups of particles separated by great distances changes very slowly with time, while the potential between relatively nearby groups of particles changes much faster. This suggests that all potentials and forces need not be updated at each time step. An initial implementation of this "hierarchical time step" idea indicates that it can save at least an order of magnitude in execution time.

High-performance implementations

Running simulations of big molecular systems interacting over long durations with accurately calculated forces takes a lot of computing power. Many of the algorithms involved in biomolecular modeling are challenging from the point of view of parallel implementations that efficiently scale to large parallel machines with hundreds of processors. We are working on some of these challenges in a project we call NAMD.

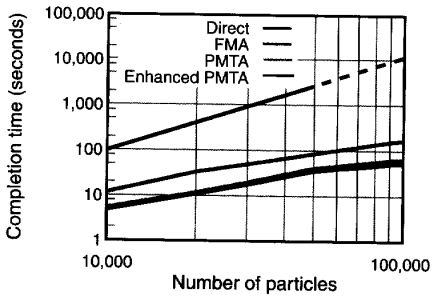

Figure 6. The "enhanced PMTA" multipole-accelerated algorithm can compute the interactions between 100,000 particles in 50 seconds, compared with about 10,000 seconds for "direct" (all-pairs) summation, on a fast workstation.

Currently available parallel computers with hundreds of processors provide tens of gigaflops in performance. Within the next 2 to 3 years, computers with thousands of microprocessors may reach teraflops performance. However, there remain significant hurdles to jump before this power can be harnessed effectively. A major one is development of software. Parallel computer architectures are so diverse that programs written for one machine are not easily ported to another. Parallel programming is also more difficult because of issues such as *load balancing*—which tasks should be assigned to each processor; *scheduling*—the sequence in which a processor should execute its tasks; and *synchronization*—how to coordinate the work of the different processors. Finally, reusing independently written modules is difficult because the module interfaces are too complex in the parallel context.

To ensure that the software we develop for biomolecular modeling is portable, we are writing two versions of NAMD, one in Charm++[18] and one in PVM. Charm++, a parallel language developed at the University of Illinois, is message-driven, object-oriented, and portable. Object orientation facilitates the modular development of programs for MD while message-driven execution boosts efficiency by automatically adapting the runtime schedule to tolerate communication latencies. PVM, the "parallel virtual machine," supports portable message-passing primitives. It is being used in many applications, and is a commonly supported system in new offerings from IBM, DEC, and Cray Research. Using both PVM and Charm++ allows us to cover a larger set of parallel machines, and Charm++ provides a greater degree of flexibility for developing complex algorithms. MPI, the emerging Message Passing Interface standard, is likely to subsume the functionality of PVM,

and if adopted, will be more widely supported than it is now.

Charm

Charm is C with a few syntactic extensions for parallelism; Charm++ is to Charm as C++ is to C. Programs written with Charm can run efficiently without change on any MIMD machine. The system currently runs on the iPSC/860, Paragon, Ncube/2, CM-5, IBM SP1, Sequent Symmetry, Encore MultiMax, and a network of workstations. Charm provides a suite of dynamic load-balancing strategies and five specific modes of information sharing. Each mode is supported by a uniform set of primitives, which are implemented differently on different parallel machines. A graphics-based "expert" performance analysis tool has been developed for Charm.

Scalability guides us: programs should run on machines with a few to a few hundred processors.

Charm is a stable, robust system that is being used for applications such as CFD, VLSI CAD, and operations research in addition to computer science. It is particularly useful for our project because of its support for irregular computations and modularity. The modularity support is crucial: modules for various energy calculations, numerical algorithms, and dynamical schemes can all be separated cleanly in Charm. Thus we shall have a test bed in which various pieces, developed relatively independently by different subgroups, can be plugged in and experimented with easily.

One useful Charm feature is message-driven execution: A program consists of many processes on each processor; each process may be waiting for more than one message at a time, and a process is scheduled for execution only when there is a message for it. This technique tolerates the communication latency and unpredictability of remote response times. With message-driven execution, one can invoke multiple library modules concurrently on one processor so that the idle time in one module can be utilized for computations in another. In contrast, in traditional message-passing programs, such overlapping often requires breaking the library abstractions and merging the modules. We recently demonstrated the advantages of message-driven execution in a "concurrent reduction" kernel. The example we used is abstracted and modified from the nonbonded force calculation in a parallelized version of a molecular mechanics code. Each processor has an array A of size N in which to store forces, where N is the total number of atoms in the system being simulated. Each processor is also assigned a subset of pairs of atoms. The processor computes the forces between each assigned pair, and adds them to the forces being accumulated in array A. The computation thus requires each processor to compute the values of the elements of its own array and to compute the global sum of the arrays across all processors. Thus, the ith element of A on every processor after the operation is the sum of the ith elements computed by each processor. Global summing is expensive, particularly for a large array.

One can divide the array A into k parts, and in a loop, compute each partition and call the reduction library for each segment separately. A traditional SPMD (single-program multiple-data) message-passing program would have to transfer full control to the reduction library. Thus the substantial idle time during the reduction cannot be used effectively. A message-driven formulation, on the other hand, can start a reduction for a partition concurrently while it is computing the next partition. With this strategy, multiple reduction operations overlap with each other and with computations of later partitions. Figure 7 shows the completion time of the traditional and message-driven implementation of this example. Reduction time does not increase with the number of processors, because of the pipelining-like effect obtained by message-driven execution.

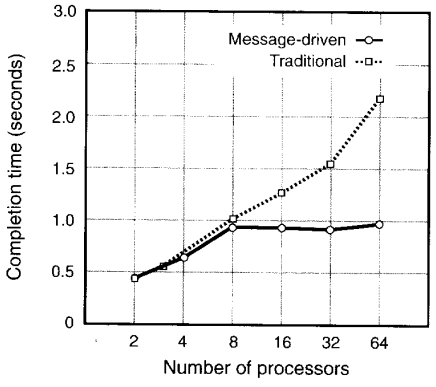

Figure 7. Message-driven execution greatly speeds completion time for concurrent reductions. Here the number of atoms N = 40,960; the array is divided into k = 160 partitions.

Parallelization strategies

Even after a parallel algorithm is chosen, many different parallel implementation strategies are possible. The algorithm specifies the set of computational actions, while the implementation strategies involve specifying the data distribution, assigning computational actions to processors, and scheduling or sequencing these actions. These choices have a significant impact on the performance of a parallel program.

Our choice of parallel algorithms and parallelization strategies is guided by scalability: the programs should run on machines with a few to a few hundred processors. Ideally, we should be able to simulate systems of larger numbers of atoms in the same amount of wall-clock time "simply" by using a proportionately larger number of processors, so that systems consisting of over a million atoms become as feasible as systems of a few thousand atoms. Also, memory scalability should be ensured by keeping the memory requirements of a processor proportional to the number of atoms per processor.

Three ways of partitioning work into separate processes are as follows: a fixed assignment of atoms to a process, a fixed set of forces to a process, or a fixed region of space to a process. The last of these we call spatial decomposition, and on physical grounds it might be expected to reduce the overall communication load enough to render the program scalable. This is the decomposition strategy we have adopted, following the example of a molecular dynamics program called PMD written by Andreas Windemuth, now at Columbia University.

For large systems it is appropriate to use FMA and other tree-structured algorithms. However, these algorithms have an irregular, unpredictable structure. Such irregularities can be handled by using the following mechanisms, which are available in Charm:

- Message-driven execution can be used to tolerate latencies and can exploit dynamic schedules that adapt to runtime conditions.
- Dynamic load-balancing strategies can deal with computational loads that change over time.
- Priorities can be used to focus execution on critical paths.

We should not close without remarking on another critical component of the biomolecular modeling enterprise—the hardware. It is difficult to forecast the shape of large-scale computing in the future. At present, networked very high performance workstations are attractive both because they are relatively affordable and because a research group like ours can have exclusive use of the machines and thereby get excellent performance in terms of wall-clock time. We are developing the NAMD project, for instance, on a cluster of twelve HP 735/125 workstations connected with an ATM, or asynchronous transfer mode, switch. ATM technology is expected to evolve into a low-cost, high-performance protocol because of its wide adoption for communications.

The development of computer technology has allowed computational chemists and biologists to understand biochemical processes of increasing size and time scales. Researchers studied condensed systems of several hundred atoms in the 1960s, modeled biomolecules with hundreds of atoms in the 1970s, and simulated biopolymers with several thousand atoms in the 1980s. Each increase in scale opened up qualitatively new computational research areas—the theory of elementary reactions, of liquids and polymers, of enzyme catalysis. Combining networks of workstations and efficient parallel algorithms now lets us model, in the 1990s, *biopolymer systems* on a large scale and for long times. This will open new avenues for research in molecular biomedicine, namely, the study of interactions among large biopolymers such as proteins and DNA. The new research will contribute to our understanding of the molecular architecture and control of biological cells, and of disease and its prevention, to a much greater extent than ever before. ◆

Acknowledgments

Our research is supported by the NSF (Advanced Scientific Computing, Computational and Theoretical Chemistry, and Computational Biology divisions) and ARPA Grand Challenge Group Collaboration Award ASC-9318159. We thank our laboratory collaborators Tom Bishop, Bill Blanke, Andrew Dalke, Philippe Derreumaux, Bill Elliot, Dan Gray, Attila Gursoy, Ziyad Hakura, Bill Humphrey, Hongmei Jian, Constantine Kreatsoulas, Chris Lambert, Ilya Logunov, Margaret Mandziuk, Mark Nelson, Gomathi Ramachandran, Bill Rankin, Sebastian Reich, Qing Sheng, Nour Toukmaji, Willy Wriggers, Dong Xu, Guihua Zhang, and Feng Zhou. Gila Budescu and Edna Shavit performed crucial organizational work. We thank Jan Hermans for many valuable consultations.

References

1. J.A. McCammon and S.C. Harvey, *Dynamics of Proteins and Nucleic Acids*, Cambridge Univ. Press, Cambridge, 1987.
2. T.J. Perun and C.L. Propst, eds., *Computer-Aided Drug*

Design, Methods and Applications, Marcel Dekker, New York, 1989.
3. A. Brünger, J. Kuriyan, and M. Karplus, "Crystallographic *R* Factor Refinement by Molecular Dynamics," *Science*, Vol. 235, 1987, pp. 458–460.
4. H. Heller, M. Schaefer, and K. Schulten, "Molecular Dynamics Simulation of a Bilayer of 200 Lipids in the Gel and in the Liquid-Crystal Phases," *J. Physical Chemistry*, Vol. 97, No. 31, 1993, pp. 8,343–8,360.
5. B.F. Luisi et al., "Crystallographic Analysis of the Interaction for the Glucocorticoid Receptor with DNA," *Nature*, Vol. 352, 1991, pp. 497–505.
6. M. Lorenz, D. Popp, and K.C. Holmes, "Refinement of the F-Actin Model Against X-Ray Fiber Diffraction Data by the Use of a Directed Mutation Algorithm," *J. Molecular Biology*, Vol. 234, 1993, pp. 826–836.
7. N.R. Cozzarelli and J.C. Wang, eds., *DNA Topology and Its Biological Effects*, Cold Spring Harbor Laboratory Press, Cold Spring Harbor, N.Y., 1990.
8. T. Schlick, B. Li, and W.K. Olson, "The Influence of Salt on the Energetics and Dynamics of Supercoiled DNA," to be published in *Biophysical J.*, 1994.
9. M.P. Allen and D.J. Tildesley, *Computer Simulation of Liquids*, Oxford Univ. Press, New York, 1987.
10. R.W. Pastor, "Techniques and Applications of Langevin Dynamics Simulations," in G.R. Luckhurst and C.A. Veracini, eds., *The Molecular Dynamics of Liquid Crystals*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1994, pp. 85–138.
11. J.M. Sanz-Serna and M.P. Calvo, *Numerical Hamiltonian Problems*, Chapman and Hall, London, 1994.
12. D. Okunbor and R.D. Skeel, "Canonical Numerical Methods for Molecular Dynamics Simulations," *J. Computational Chemistry*, Vol. 15, 1994, pp. 72–79.
13. G. Zhang and T. Schlick, "LIN: A New Algorithm to Simulate the Dynamics of Biomolecules by Combining Implicit-Integration and Normal Mode Techniques," *J. Computational Chemistry*, Vol. 14, 1993, pp. 1,212–1,233.
14. G. Zhang and T. Schlick, "The Langevin/Implicit-Euler/Normal-Mode Scheme for Molecular Dynamics at Large Time Steps," *J. Chemical Physics*, Vol. 101, 1994, pp. 4,995–5,012.
15. T. Schlick and A. Fogelson, "TNPACK—a Truncated Newton Minimization Package for Large-Scale Problems: I. Algorithm and Usage," *ACM Trans. Math. Software*, Vol. 18, 1992, pp. 46–111.
16. L. Greengard, "Fast Algorithms for Classical Physics," *Science*, Vol. 265, Aug. 12, 1994, pp. 909–914.
17. J.A. Board Jr. et al., "Scalable Variants of Multipole-Accelerated Algorithms for Molecular Dynamics Applications," to be published in *Proc. Seventh SIAM Conf. Parallel Processing for Scientific Computing*, SIAM, Philadelphia, Pa., 1995.
18. L.V. Kalé, "The Chare Kernel Parallel Programming Language and System," *Proc. 1990 Int'l Conf. Parallel Processing*, Vol. 2, Pennsylvania State Univ. Press, University Park, Pa., 1990, pp. 17–25.

John A. Board Jr. is associate professor of electrical and computer engineering and of computer science at Duke University, where he has been since 1987. His research interests include development of new algorithms for scientific applications and implementation of same on parallel and distributed computing systems. He received a BSE and an MS in electrical engineering from Duke University and a DPhil degree in theoretical physics from Oxford University, where he was a Rhodes Scholar. He is a member of IEEE, ACM, and the American Physical Society.

Laxmikant V. Kalé received a BTech degree in electronics engineering from Benares Hindu University, Varanasi, India, an ME in computer science from the Indian Institute of Science in Bangalore, and a PhD in computer science from the State University of New York, Stony Brook. He spent two years at the Tata Institute of Fundamental Research, and in 1985 went to the University of Illinois at Urbana-Champaign, where he is now associate professor of computer science. His research interests include parallel programming tools, performance feedback and debugging, scientific and engineering applications of large-scale parallel computing, and parallel processing for AI. Kalé is a member of the IEEE, IEEE Computer Society, and ACM.

Klaus Schulten earned his PhD in chemical physics at Harvard University in 1974. He then spent six years as a research assistant at the Max Planck Institute for Biophysical Chemistry in Göttingen, Germany, and eight years as associate professor of physics at the Technical University of Munich. Since 1988 he is professor of physics, chemistry, biophysics, and electrical and computer engineering at the University of Illinois at Urbana-Champaign. He directs the Theoretical Biophysics Group at the Beckman Institute at Illinois. Schulten's research interests focus on theoretical and computational physics and biology.

Robert D. Skeel is professor of computer science at the University of Illinois at Urbana-Champaign (Department of Computer Science and Beckman Institute), where he has been since 1973. His research interest is in computational methods for biomolecular modeling. He received a BSc in applied mathematics from the University of Alberta, an MSc in mathematics from the University of Toronto, and a PhD in computing science again from Alberta. He is a member of SIAM and of the editorial board for *SIAM Journal on Scientific Computing*.

Tamar Schlick is associate professor of chemistry and mathematics at New York University (Chemistry Department and the Courant Institute of Mathematical Sciences), and associate investigator at the Howard Hughes Medical Institute. Her interests include molecular dynamics algorithms, large-scale nonlinear optimization, and macromolecular (particularly DNA) simulations. She earned her BS in mathematics at Wayne State University and her MS and PhD in applied mathematics at the Courant Institute. She is a member of SIAM, AMS, AWM, ACS, and the Biophysical Society. Schlick serves on the National Research Council panel on mathematical challenges in computational chemistry, the AMS–SIAM committee on applied mathematics, and the editorial board of Oxford's New Series in Scientific Computing.

Readers may reach the authors in care of Skeel at the University of Illinois, Department of Computer Science, 1304 West Springfield Avenue, Urbana, IL 61801-2987; e-mail, skeel@cs.uiuc.edu. E-mail addresses of the other authors are jab@ee.duke.edu, kale@cs.uiuc.edu, kschulte@ks.uiuc.edu, and schlick@nyu.edu.