J. Comp. Chem. 16:1192--1209 (1995)

Algorithms for Constrained Molecular Dynamics

Eric Barth
Courant Institute of Mathematical Sciences,
New York University
251 Mercer Street
New York, NY 10012

Krzysztof Kuczera
Departments of Chemistry and Biochemistry,
University of Kansas
2010 Malott Hall
Lawrence, KS 66045

Benedict Leimkuhler
Department of Mathematics,
University of Kansas
405 Snow Hall
Lawrence, KS 66045

Robert D. Skeel
Department of Computer Science,
University of Illinois, Urbana-Champaign
1304 West Springfield Avenue,
Urbana, Illinois 61801-2987

ALGORITHMS FOR CONSTRAINED MOLECULAR DYNAMICS

ERIC BARTH*, KRZYSZTOF KUCZERA , BENEDICT LEIMKUHLER† AND ROBERT D. SKEEL‡

Abstract. In molecular dynamics simulations, the fastest components of the potential field impose severe restrictions on the stability and hence the speed of computational methods. One possibility for treating this problem is to replace the fastest components with algebraic length constraints. In this paper, the resulting systems of mixed differential and algebraic equations are studied. Commonly used discretization schemes for constrained Hamiltonian systems are discussed. The form of the nonlinear equations is examined in detail and used to give convergence results for the traditional nonlinear solution technique *SHAKE iteration* and for a modification based on *Successive OverRelaxation* (SOR). A simple adaptive algorithm for finding the optimal relaxation parameter is presented. Alternative direct methods using sparse matrix techniques are discussed. Numerical results are given for the new techniques, implemented in the molecular modeling software package **CHARMM**, showing as much as twofold improvement over SHAKE iteration.

Key words. molecular dynamics, computer simulation, constraints, numerical algorithms, sparse matrix methods

1. Introduction. In molecular dynamics, the length of timestep for numerically integrating the equations of motion is dictated by the contributions to the force vector which maintain pairs of atoms near some equilibrium distance. The imposition of algebraic constraints that fix these lengths removes the associated rapid vibrational modes, enabling the use of longer timesteps without substantially altering important physical characteristics of the motion [1]. Although we treat only length constraints in the present work, constrained techniques are also of interest for conformational search and conformational free energy simulations [2]. In [3] the SHAKE iteration was described for solving the nonlinear equations at each timestep of a constrained version of the Verlet discretization, and a similar scheme was proposed in [4] for use with the RATTLE discretization.

We describe in §2 the equations of motion for the molecule with algebraic constraints and discuss these two discretizations. In §3 we examine the form of the nonlinear equations and discuss the existence of solutions. We describe the SHAKE iteration in a matrix formulation, and show its equivalence to the nonlinear Gauss-Seidel-Newton iteration [5]. This leads to a convergence result and theoretical rate of convergence for the SHAKE iteration as well as a more general scheme based on *Successive OverRelaxation* (SOR). Experiments with molecules of interest show that SOR improves the performance of SHAKE iteration by a factor of two or more for certain choices of relaxation parameter, at no additional cost. We present an algorithm for effective choice of SOR relaxation parameter ω. It has often been suggested (e.g., [6]) that the SHAKE iteration is an efficient alternative to matrix methods for numerical solution of the nonlinear equations encountered in molecular dynamics, but little discussion of these methods is found in the literature. In §4 we present several matrix methods for the nonlinear systems which utilize sparse direct matrix techniques. We

* The work of this author was supported by a fellowship from the Kansas Institute for Theoretical and Computational Science and by NSF grant DMS-9303223.

† The work of this author was supported in part by NSF grant DMS-9303223.

‡ The work of this author was supported in part by NIH grant P41RR05969 and by DOE/NSF grant DE-FG02-91-ER25099/DMS-9304268.

present numerical results which show that, in some instances, the matrix techniques proposed here perform faster than iterative schemes like SHAKE iteration and SOR. §5 contains the results of numerical experiments comparing the performance of the various methods on several molecules of interest. In appendices, we describe the implementation of the proposed methods in the molecular dynamics software package **CHARMM** [7] and compare the trajectories generated by the constraint techniques.

2. Background. The Newtonian equations of motion for a system of N particles interacting in potential field $V = V(q)$ are a Hamiltonian system:

$$M\ddot{q} = -\nabla_q V(q). \tag{1}$$

Here $q \in \mathbf{R}^{3N}$ is the vector of cartesian particle positions, and M is a $3N$-dimensional positive diagonal mass matrix of the form

$$M = \mathrm{diag}(m_1, m_1, m_1, m_2, m_2, m_2, \ldots, m_N, m_N, m_N).$$

Particle systems of the form (1) arise frequently in physics, chemistry and biology. An important application is to model dynamics at the molecular level. For this application, the potential field might be written [8]

$$V(q) = V_b(q) + V_\theta(q) + V_\phi(q) + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \psi_{ij}(|q_i - q_j|), \tag{2}$$

where q_i and q_j in $\mathbf{R}^3$ denote the position vectors of the ith and jth atoms, ψ represents terms due to nonbonded interactions and $|\cdot|$ denotes Euclidean distance in $\mathbf{R}^3$. $V_b(q)$ is the component of the potential due to deformations of chemical bonds between certain atom pairs. A bond α between atoms i and j is usually described by a harmonic term of the form

$$\frac{K_\alpha}{2}(|q_i - q_j| - L_\alpha)^2, \tag{3}$$

where L_α is the equilibrium length of the bond, and K_α is the force constant of the bond. Terms V_θ and V_ϕ for angles and *dihedrals* (torsions) are also present. It is well known that the force constants associated with the bond terms are substantially larger than those for angles or dihedrals [9]. Although we seek to simulate the long term conformational changes of a molecule, not the rapid vibrations of the bonds, the fast bond components necessitate very short timesteps in the numerical integration.

Following [10] we impose algebraic constraint equations

$$g_\alpha(q) = \frac{1}{2}(|q_i - q_j|^2 - {L_\alpha}^2) = 0$$

that freeze the bond lengths, thereby removing the associated rapid vibrational modes. Thus we are concerned with efficient numerical discretization of the equations of motion (1) subject to m holonomic (position) constraints:

$$M\ddot{q} = -\nabla_q V(q) - g'(q)^t \lambda \tag{4}$$

$$0 = g(q). \tag{5}$$

Here $g : \mathbf{R}^{3N} \to \mathbf{R}^m$, g' is the matrix of partial derivatives with respect to position q, and $\lambda \in \mathbf{R}^m$ is a vector of time-dependent Lagrange multipliers. These equations

form a system of differential-algebraic equations (DAEs) of index three: three differentiations of (5) with respect to time are required reduce the equations to a system of ordinary differential equations [11]. The solution manifold underlying (4)–(5) is

$$\mathcal{M} = \{(q,p)|g(q) = 0, g'(q)M^{-1}p = 0\}.$$

The *hidden* constraint $g'(q)M^{-1}p = 0$ is obtained through time differentiation of the position constraint.

A computational approach for unconstrained problems that remains the basis for modern molecular dynamics simulation is a discretization that is often referred to as the *Verlet method* [12] when applied in molecular dynamics. When rewritten as a one-step discretization incorporating half-steps in the momenta,

$$\begin{aligned} q_{n+1} &= q_n + hM^{-1}p_{n+1/2} \\ p_{n+1/2} &= p_n - (h/2)\nabla_q V(q_n) \\ p_{n+1} &= p_{n+1/2} - (h/2)\nabla_q V(q_{n+1}) \end{aligned}$$

it is called the *leap-frog method.*[1] Due to its evident simplicity and efficiency, Verlet/leap-frog continues to be popular in molecular dynamics and is the most commonly used option for dynamical simulations in the important molecular dynamics software package **CHARMM** [7]. It is known that the method is symplectic i.e., conserves the wedge product $dq \wedge dp$ of differentials [14]. The property of being symplectic duplicates a corresponding property for the true flow map of a Hamiltonian system, and may explain the good performance of the method for long time interval simulations [15]. In particular, the property of being symplectic implies the Liouville property of conservation of volume in phase space.

The Verlet method was adapted to allow for bond constraints by Ryckaert et al [3], and the resulting discretization scheme is referred to as the Verlet method with SHAKE-type constraints, or simply SHAKE.[2] An alternative velocity-level formulation for the constrained case, RATTLE, was proposed by Andersen [4]. For $n = 0$,

one can infer from (7) and (9) that

$$p_{1/2} = p_0 - (h/2)\nabla_q V(q_0) - (h/2)g'(q_0)^t\lambda_0.$$

At each step of a discretization like RATTLE or SHAKE, a system of nonlinear algebraic equations must be solved. Assuming that the nonlinear equations are solved exactly at each step, SHAKE and RATTLE are globally second order accurate. The two methods are equivalent at timesteps in position and at half steps in momenta and, moreover, the RATTLE method is a symplectic discretization. SHAKE is essentially symplectic [16] in that the wedge product is preserved, but the computed solution does not preserve the hidden constraint $g'(q)M^{-1}p = 0$. Symplectic discretization schemes

[1] The velocity-level leap-frog formulation enjoys improved stability compared to the position-level scheme, see [13]. Other formulations of the underlying Verlet method include velocity leap-frog and velocity Verlet.

[2] To avoid possible ambiguity, we will refer to the iterative solution technique as *SHAKE iteration* to distinguish it from the constrained Verlet discretization SHAKE.

SHAKE	(6)	q_{n+1}	$=$	$q_n + hM^{-1}p_{n+1/2}$
	(7)	$p_{n+1/2}$	$=$	$p_{n-1/2} - h\nabla_q V(q_n) - hg'(q_n)^t\lambda_n$
	(8)	0	$=$	$g(q_{n+1})$
	(9)	p_n	$=$	$(p_{n+1/2} + p_{n-1/2})/2$
RATTLE	(10)	q_{n+1}	$=$	$q_n + hM^{-1}p_{n+1/2}$
	(11)	$p_{n+1/2}$	$=$	$p_{n-1/2} - h\nabla_q V(q_n) - hg'(q_n)^t\lambda_n$
	(12)	0	$=$	$g(q_{n+1})$
	(13)	p_n	$=$	$(p_{n+1/2} + p_{n-1/2})/2 - (h/2)g'(q_n)^t\mu_n$
	(14)	0	$=$	$g'(q_n)M^{-1}p_n$

for ordinary differential equations are discussed in [17], and symplectic methods for constrained problems are discussed in [18]. The SHAKE and RATTLE discretizations have been generalized to families of higher-order schemes by Jay [19] using multistage partitioned Runge-Kutta methods and by Reich [20] through concatenation of steps with appropriately chosen stepsizes (following the ideas of Yoshida [21]). Regardless of which discretization is used, it is necessary to solve the nonlinear equations at each step accurately in order to retain the desirable theoretical properties.

3. Length constraints and nonlinear equations.

3.1. Existence of solutions to the discrete equations. Substituting (7) into (6) leads to the equations

$$\begin{aligned} q_{n+1} &= q_n + hM^{-1}(p_{n-1/2} - h\nabla_q V(q_n) - hg'(q_n)^t\lambda_n) \\ 0 &= g(q_{n+1}). \end{aligned}$$

Denoting $Q \equiv q_{n+1}$, $G \equiv g'(q_n)$, and $\Lambda \equiv h^2\lambda_n$, we can write the nonlinear system as

$$g(\bar{Q} - M^{-1}G^t\Lambda) = 0, \tag{15}$$

where

$$\bar{Q} \equiv q_n + hM^{-1}\left(p_{n-1/2} - h\nabla_q V(q_n)\right) \tag{16}$$

represents the result of an unconstrained step of size h.

The behavior of numerical methods depends on nonsingularity of the derivative matrix

$$\begin{aligned} R(\Lambda) &\equiv \partial_\Lambda\, g(\bar{Q} - M^{-1}G^t\Lambda) && (17) \\ &= g'(\bar{Q} - M^{-1}G^t\Lambda)M^{-1}G^t. && (18) \end{aligned}$$

The structure of this matrix is discussed in §3.2.

Equation (15) might easily have multiple solutions for Λ or none at all. When solutions exist, they are not unique. For m length constraints we can expect perhaps

2^m solutions. For the example of a pair of atoms with unit mass joined by a length constraint, several possibilities are shown in Fig. 1a,b,c. The positions of the atoms at the previous integration step and after the unconstrained step are shown, with dashed lines for the latter. From (15) we see that a solution Q is obtained by moving the coordinates of $\bar{Q}$ in the directions given by G^t, the gradient of the constraints evaluated at the previous integration step. The magnitudes of the moves are proportional to the entries of Λ. For a single length constraint, these gradient directions are simply the positive and negative vector between the particles. Hence the atoms are moved in opposite directions with equal magnitude. Fig. 1a shows a situation in which no solution exists, while the matrix $R(0)$ is nonsingular. If the unconstrained step takes the atoms outside this cylinder of solutions, a smaller timestep h must be used.

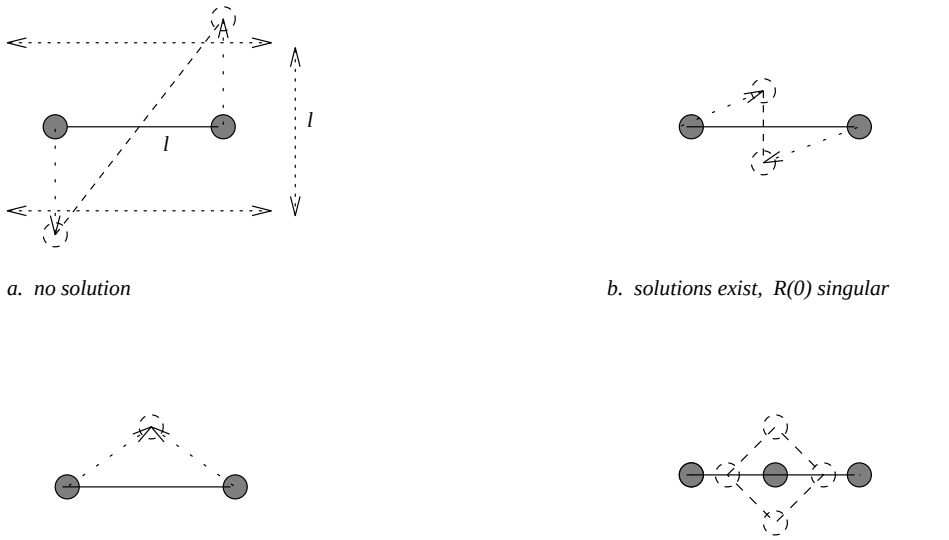


FIG. 1. *Existence of solutions for simple length constraint models.*

Solutions may exist when:

1. $g'(\bar{Q})M^{-1}G^t$ is singular, even though both $g'(\bar{Q})$ and G have full rank. In Fig. 1b the bonds from the previous and unconstrained steps are perpendicular.
2. $g'(\bar{Q})$ is rank deficient. In Fig. 1c the two atoms have the same position after the unconstrained step.
3. G is rank deficient. Fig. 1d illustrates this case with a closed loop of four atoms and four length constraints of equal length. At the previous timestep, the positions of two atoms are identical, leading to dependent constraints. In practice, occurrences of dependent constraints might be 'stepped over' by the numerical integrator, but many iterations of the nonlinear solver may be required, and the gradient directions are ill conditioned. This situation can be remedied only by a reformulation of the problem to avoid dependent constraints. It can be shown that G always has full rank when the constraints form no closed loops.

It is perhaps typical that there are 2^m solutions. In order to specify the physically correct solution we could use some continuation process [22, 23] on $q_{n+\theta}$ as θ varies

from 0 to 1. We choose

$$q_{n+\theta} = q_n + \theta h(P - hM^{-1}G^t\Lambda), \tag{19}$$

where $P = M^{-1}(p_{n-1/2} - h\nabla_q V(q_n))$, $G = g'(q_n)$, and $\Lambda = \lambda_n$. Continuation uniquely defines $\Lambda = \Lambda(\theta)$ unless bifurcation occurs for $0 \leq \theta \leq 1$. It is sufficient, it can be shown, that $\partial_\Lambda g(q_{n+\theta})$ be of full rank for $0 \leq \theta \leq 1$.

3.2. Form of the nonlinear equations. To study the structure of the matrix R in (18), we first consider a related symmetric matrix

$$\hat{R}(q) \equiv g'(q)M^{-1}g'(q)^t,$$

in which the two occurrences of the constraint gradient are evaluated at the same q. This is analogous to the well known B-matrices from the molecular vibration theory of Wilson [24] for the case of bond length internal coordinates. Evidently, the matrix $\hat{R}(q)$ is positive definite when the constraints are not dependent. The entries of this matrix are also nearly constant along solutions for the macromolecular modeling problem. This can be seen using techniques from [24]. As an alternative, we first decompose the matrix using the *stamps* of VLSI circuit simulation [25]. Consider length constraints of the form

$$|q_l - q_r|^2 = L^2.$$

We will view stamps as $N \times N$ block matrices, with 3×3 blocks. With this convention, for fixed indices l and r, $1 \leq l, r \leq N$, the associated stamp matrix S has the following nonzero elements: $s_{ll} = s_{rr} = I_3$; $s_{lr} = s_{rl} = -I_3$ where I_3 denotes the 3×3 identity matrix. Although the directionality is unimportant, we refer to the two indices of a stamp as the left and right. For each bond k, we have a corresponding stamp S_k.

A constraint of length L_k can be written

$$\frac{1}{2}(q^t S_k q - L_k^2) = 0$$

and the matrix of partial derivatives is given by

$$g'(q) = \begin{bmatrix} q^t S_1 \\ q^t S_2 \\ \vdots \\ q^t S_m \end{bmatrix}.$$

$\hat{R}(q)$ can also be decomposed using stamps:

$$\hat{R}(q) = g'(q)M^{-1}g'(q)^t = \begin{bmatrix} q^t S_1 M^{-1} S_1 q & q^t S_1 M^{-1} S_2 q & cldots & q^t S_1 M^{-1} S_m q \\ q^t S_2 M^{-1} S_1 q & q^t S_2 M^{-1} S_2 q & cldots & q^t S_2 M^{-1} S_m q \\ \vdots & \vdots & \ddots & \vdots \\ q^t S_m M^{-1} S_1 q & q^t S_m M^{-1} S_2 q & \ddots & q^t S_m M^{-1} S_m q \end{bmatrix}.$$

Here we have used the symmetry of the stamps. We denote the left and right indices associated with stamp k by $l(k)$ and $r(k)$, respectively. The individual elements of the

matrix can be evaluated as follows. On the diagonal, we have

$$\begin{aligned} q^t S_k M^{-1} S_k q &= \frac{1}{m_{l(k)}}(q_{l(k)} - q_{r(k)})^t(q_{l(k)} - q_{r(k)}) + \frac{1}{m_{r(k)}}(q_{r(k)} - q_{l(k)})^t(q_{r(k)} - q_{l(k)}) \\ &= \left(\frac{1}{m_{l(k)}} + \frac{1}{m_{r(k)}}\right)|q_{l(k)} - q_{r(k)}|^2. \end{aligned}$$

The off-diagonal elements are nonzero only when the corresponding pair of edges shares a common vertex. For example, if $r(i) = l(j)$, then we have

$$\begin{aligned} q^t S_i M^{-1} S_j q &= \frac{1}{m_{l(j)}}(q_{l(i)} - q_{r(i)})^t(q_{l(j)} - q_{r(j)}) \\ &= -\frac{1}{m_{l(j)}}|q_{l(i)} - q_{r(i)}||q_{l(j)} - q_{r(j)}| \cos\theta_{ij} \end{aligned}$$

where θ_{ij} represents the angle between the vectors $q_{l(i)} - q_{r(i)}$ and $q_{r(j)} - q_{l(j)}$. Hence the entries of $\hat{R}$ depend upon the distance between pairs of atoms involved in length constraints and the angles θ_{ij} between these constraints.

If we have correctly solved the length constraints, we know that $|q_{l(k)} - q_{r(k)}| = L_k$ and is constant throughout the integration. (We require, of course, that the constraints be satisfied by the initial conditions. In a practical implementation, this should be checked and corrected if needed.) Let $\hat{R} = \{\rho_{ij}\}$. If q satisfies the constraints, then

$$\rho_{kk} = L_k^2\left(\frac{1}{m_{l(k)}} + \frac{1}{m_{r(k)}}\right),$$

while for $j \neq i$, if edges i and j share vertex α, then

$$\rho_{ij} = \frac{1}{m_\alpha} L_i L_j \cos\theta_{ij}$$

where θ_{ij} is the angle between edges i and j measured at vertex α. For the molecular dynamics problem the potential will contain, in addition to terms like (3), corresponding angle terms which inhibit variation in the θ_{ij} from their reference values θ_{ij}^0. Hence, variation in the entries of $\hat{R}$ is likewise inhibited.

The structure of the nonzero off-diagonal elements can be stated graph theoretically. It is natural to consider the graph structure of a molecule by taking the atoms as nodes and chemical bonds (or in our case length constraints) as edges. Nodes of the graph are adjacent exactly when the corresponding atoms are a bonded pair. The adjacency structure of a graph is commonly represented by an *adjacency matrix*, which has nonzero entries corresponding to adjacent pairs of nodes. The dual concept is to consider pairs of edges in the graph to be adjacent if they share a common node. The adjacency matrix for this dual graph describes the nonzero structure of R . We use these ideas in §3.3 in regard to the convergence of the SHAKE iteration, and again in §4.1 where we consider sparse matrix techniques.

Having examined the structure of the symmetric matrix $\hat{R}$, we now must consider

$$R = g' M^{-1} G^t$$

with G, as before, evaluated at the coordinates of the previous timestep, and g' evaluated at the unconstrained step $\bar{Q}$ plus some correction. The nonzero structure is

unchanged (i.e., R is *structurally symmetric*). By the methods above it can be seen that

$$R_{ii} = \tilde{L}_i L_i \left(\frac{1}{m_{l(i)}} + \frac{1}{m_{r(i)}} \right) \cos \phi_i,$$

while for $j \neq i$, if edges i and j share vertex α, then

$$R_{ij} = \frac{1}{m_\alpha} \tilde{L}_i L_j \cos \tilde{\theta}_{ij}.$$

Here $\tilde{L}_i$ is the current length of the bond, ϕ_i is the angle (with appropriate sign) formed by the current and previous bond i, and $\tilde{\theta}_{ij}$ is the angle between bond i at the previous step and the current bond j. It can be seen from (16) that

$$\bar{Q} = q_n + \mathcal{O}(h),$$

and hence

$$\begin{aligned} \tilde{L}_k &= L_k + \mathcal{O}(h) \\ \cos \tilde{\theta}_{ij} &= \cos \theta_{ij} + \mathcal{O}(h) \end{aligned}$$

with the result that R is an $\mathcal{O}(h)$ perturbation of the symmetric matrix $\hat{R}$. Note finally that for a solution Q^* of the nonlinear system, $\tilde{L}_i = L_i$.

The form of the nonlinear system can be shown to yield important matrix properties for many examples in macromolecular modeling. Assuming $\theta_{ij} \geq \frac{\pi}{2} \ \ \forall i, j$, we have

$$R_{ii} > 0 \ \ \forall i, \qquad R_{ij} \leq 0 \ \ i \neq j. \tag{20}$$

Molecular structure considerations imply that the interbond angles θ_{ij} often satisfy this condition, as in *bovine pancreatic trypsin inhibitor* BPTI [1]. The three-constraint water molecule is a notable exception, with angles much less than $\frac{\pi}{2}$. In that case it can be shown that for sufficiently small h, the matrix R is strictly diagonally dominant:

$$\sum_{j \neq i} |R_{ij}| < |R_{ii}| \qquad \forall i. \tag{21}$$

For the C_{60} molecule, both conditions are met. These properties will be used below in the convergence analysis of iterative solvers for the nonlinear equations.

3.3. The SHAKE iteration for the nonlinear equations. The system of nonlinear equations from §3.2 can be written in terms of the individual unknowns as

$$0 = g(\bar{Q} - M^{-1} \sum_{i=1}^{m} \Lambda_i G_i^t), \tag{22}$$

where Λ^i is the ith component of Λ, and $G_i^t = \nabla_q g_i(q_n)$. In the SHAKE iteration the individual constraint equations are linearized and solved sequentially:

$$Q_i^k = Q_{i-1}^k - M^{-1} G_i^t \Delta \Lambda_i^k \tag{23}$$

$$0 = g_i(Q_{i-1}^k) + g_i'(Q_{i-1}^k)(Q_i^k - Q_{i-1}^k). \tag{24}$$

Here the constraints are indexed by i and the sweeps through the list of constraints by k. Notice that equations (23)–(24) reduce to the scalar equation

$$\Delta\Lambda_i^k = \frac{g_i(Q_{i-1}^k)}{g_i'(Q_{i-1}^k)M^{-1}G_i^t}. \tag{25}$$

Following a successful iteration of K sweeps, we will have

$$Q_m^K = \bar{Q} - M^{-1}\sum_{i=1}^{m} G_i^t \sum_{k=1}^{K} \Delta\Lambda_i^k$$

and $|g_i(Q_i^K)| < \epsilon$ for a specified tolerance ϵ. [3] The process of modifying the unconstrained step to obey the constraints is known as *coordinate resetting*.

We discuss the convergence of the SHAKE iteration by showing that it fits into the framework of Ortega and Rheinboldt [5]. Define

$$f_i(\Lambda) = g_i(\bar{Q} - M^{-1}G^t\Lambda).$$

We want Λ so that $f_i(\Lambda) = 0$ for $i = 1, \ldots, m$. The nonlinear Gauss-Seidel iteration consists of solving

$$f_i(\Lambda_1^k, \ldots, \Lambda_{i-1}^k, \Lambda_i^k, \Lambda_{i+1}^{k-1}, \ldots, \Lambda_l^{k-1}) = 0.$$

at each iterate for Λ_i^k . Each of the scalar nonlinear equations might be solved using j iterations of Newton's method, which is the so called j-step Gauss-Seidel-Newton (GSN) method.

If we apply *1-step* GSN to the equation (22), we obtain

$$\Lambda_i^k = \Lambda_i^{k-1} - \frac{g_i(\bar{Q} - M^{-1}G^t\Lambda_i^{k-1})}{-\partial_i g_i(\bar{Q} - M^{-1}G^t\Lambda_i^{k-1})}$$

or

$$\Lambda_i^k = \Lambda_i^{k-1} - \frac{g_i(\bar{Q} - M^{-1}G^t\Lambda_i^{k-1})}{-g_i'(\bar{Q} - M^{-1}G^t\Lambda_i^{k-1})M^{-1}G_i^t}$$

or

$$\Delta\Lambda_i^k = \frac{g_i(Q_{i-1}^k)}{g_i'(Q_{i-1}^k)M^{-1}G_i^t}$$

which is precisely SHAKE iteration (25).

It is shown in [5] that this iteration converges locally to a solution Q^* when the linear Gauss-Seidel iteration applied to

$$R = g'(Q^*)M^{-1}G^t \tag{26}$$

converges, with the same convergence rate. In particular, it can be shown [26] that linear Gauss-Seidel converges when applied to a matrix with property (21) as for three-constraint water molecules and C_{60}. Convergence is also assured for *M-matrices*, an

[3] Note: this does not insure that $|g_i(Q_m^K)| < \epsilon$, $i \neq m$. In principle, we should pass once more through the constraints, merely checking that this condition holds. This check is not typically performed.

important subclass of matrices with property (20). An M-matrix R is characterized by property (20) and the existence of a positive scalar α such that $R = \alpha I - E$ where all entries of E are nonnegative and $\alpha > \rho(E)$, the spectral radius of E. We have numerically determined the matrix $\hat{R}$ associated with BPTI (and by continuity, also the matrix R for small enough h) to be an M-matrix with α equal to the largest diagonal entry $\hat{R}_{ii}$.

3.4. Nonlinear SOR iteration. In the SHAKE iteration, an initial approximation is gradually corrected until the solution is obtained. By exaggerating each individual correction by some factor ω

$$\Lambda_i^k = \Lambda_i^{k-1} - \omega \frac{f_i(\Lambda_{i-1}^k)}{\partial_i f_i(\Lambda_{i-1}^k)}$$

we can usually hasten convergence. This is *successive overrelaxation* (SOR). One wishes to find an ω which is optimal for rapid convergence of the iteration.

As with the nonlinear Gauss-Seidel-Newton iteration, it is shown in [5] that SOR-Newton converges locally to a solution Q^* with the same convergence rate as linear SOR iteration applied to $R(Q^*)$. With judicious choice of ω, we loose none of the robustness of SHAKE, and stand to gain considerably in terms of computational work. Fig. 2 gives results of a series of short MATLAB [27] simulations of a simplified *"buckminsterfullerene"* [28] using starting coordinates from the MATLAB demonstration routine `bucky.m`, angle force constants normalized to one, and with a stringent convergence criteria ($\epsilon \approx 10^{-12}$). It is seen that values of ω can be found which improve the performance of SHAKE iteration ($\omega = 1$) dramatically. While this example with its rich connectivity structure is not representative in molecular dynamics, we choose it to challenge the various techniques considered here.

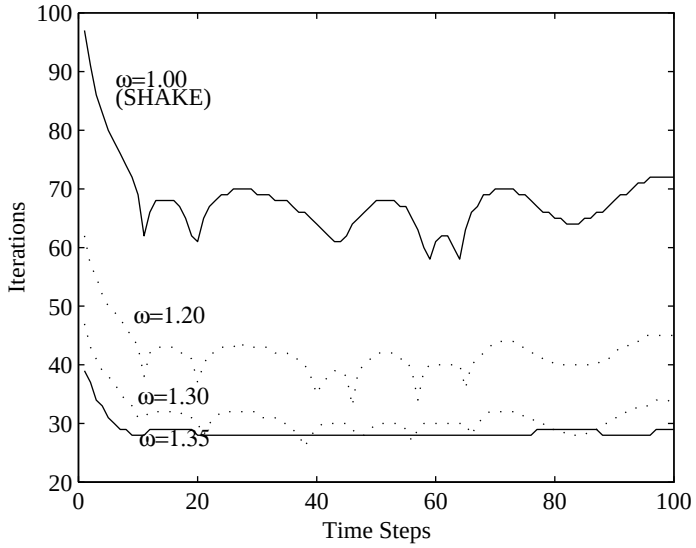


FIG. 2. *Iteration counts for SHAKE ($\omega = 1$), and several other ω values.*

3.5. An adaptive algorithm for the SOR parameter. Conditions on the coefficient matrix have been found [26, 29] which make possible the *a priori* computation of the optimal relaxation factor $\bar{\omega}$. However, the matrices from the present molecular

dynamics application almost never satisfy these conditions [30]. In this case, since we repeatedly solve nonlinear systems whose form does not change from step to step, we can use the behavior of the iteration during the early stages of the integration to find a good value of the relaxation parameter via iterative improvement:

Adaptive Relaxation Algorithm

initialize:

$\omega_0 := 1$

$\gamma_0 := 0$

$\Delta := \Delta_0$

for $k = 1, 2, \ldots$ (timesteps)

integrate and perform SHAKE/SOR iteration with relaxation parameter ω_k

compute number of iterations γ_k (see item 3 below)

if $\gamma_k > \gamma_{k-1}$ **then**

$\Delta := -\frac{\Delta}{2}$

end if

$\omega_{k+1} := \omega_k + \Delta$

end for .

Practical Considerations:

1. With $\omega_0 = 1$, we should choose $\Delta_0 > 0$. In our current implementation, $\Delta_0 = 0.1$.
2. Beginning with $\omega_0 = 1$ the relaxation parameter is increased by $\Delta = \Delta_0$ at each timestep until convergence speed of SHAKE/SOR iteration, as measured by convergence factor γ_k, no longer improves. i.e., further change of ω in the current search direction will not increase speed of convergence. Assigning $\Delta = -\frac{\Delta}{2}$ refines and reverses the direction of the search.
3. The convergence factor γ_k could also be taken as the average of the ratios $\|\Delta\Lambda^k\|/\|\Delta\Lambda^{k-1}\|$.
4. The algorithm continues to modify Δ until $\omega + \Delta$ is numerically indistinguishable from ω. A possible improvement would include a mechanism by which Δ grows automatically if convergence becomes slow for the "converged" value ω.

The behavior of the algorithm for the "buckminsterfullerene" example is illustrated by Fig. 3.

4. Newton-like iterations. A more general family of methods based on iteratively improving the multipliers Λ can be described by the general form

$$\begin{aligned} \Lambda_0 &= 0 \\ Q_k &= \bar{Q} - M^{-1}G^t\Lambda_k, \qquad k \geq 1 \\ \Lambda_{k+1} &= \Lambda_k + R_k g(Q_k), \qquad k \geq 0 \end{aligned}$$

where, for $k \geq 1$, we propose the choices

I. $R_k = (g'(Q_k)M^{-1}G^t)^{-1}$. This is the conventional Newton-Raphson iteration (NIP).

II. $\hat{R} = (GM^{-1}G^t)^{-1}$. This is what we might call *Symmetric Newton iteration.* Here the symmetric matrix $\hat{R}$ is constant throughout the iteration. Further motivation for this method comes from the observation in §3.2 that $\hat{R}$ is nearly constant over the course of the numerical integration, and hence rarely requires update and refactorization. We distinguish between symmetric newton iteration (SYMM), in which $\hat{R}$ is updated at each timestep, and *adaptive*

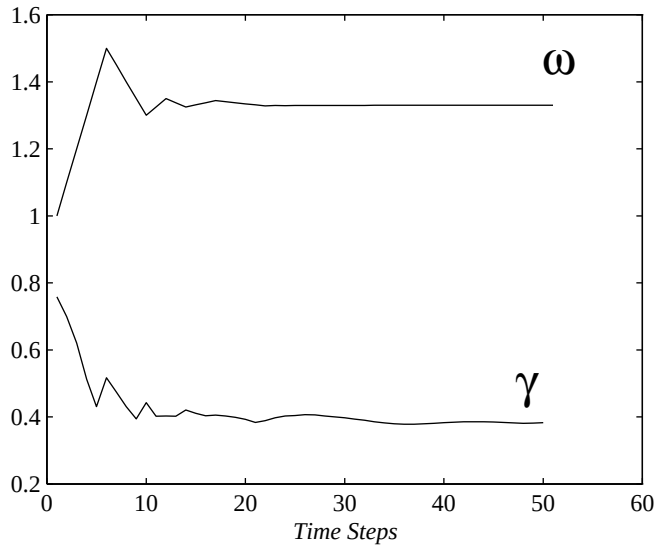


FIG. 3. *Performance of the adaptive relaxation algorithm with* $\gamma = \|\Delta\Lambda^k\|/\|\Delta\Lambda^{k-1}\|$.

symmetric newton iteration (SNIP), in which the matrix $\hat{R}$ is updated only when convergence becomes slow. In both SNIP and SYMM, we require that the constraints be satisfied in order to construct the initial matrix $\hat{R}$. For initial conditions which fail to satisfy the constraints, we use an alternative method (NIP or SHAKE) for one timestep, then proceed with SNIP/SYMM. These methods can be shown to be convergent by standard theory for full rank G and small enough h. It is important to emphasize that we solve the same system of nonlinear equations as in SHAKE, with the same convergence criteria as SHAKE. Further, these methods are equally robust as SHAKE. We present these alternative numerical methods in an attempt to achieve a more efficient implementation of (6)—(9).

4.1. Sparse matrix techniques. In the system of linear of equations $Ax = b$, the coefficient matrix A is said to be sparse if it has relatively few nonzero entries. The task of a sparse linear solver is to order the equations so that the process of matrix factorization produces additional nonzeros (called *fill-in*) in a way which can be conveniently handled with regard to additional computer storage overhead and computational work. For sparse matrices, the structure of the nonzero elements and the fill-in produced by factorization can often be neatly characterized in terms of the adjacency structure of an associated graph. We have already seen that the structure of a molecule can naturally be represented by a graph and that the structure of the dual graph determines the structure of the nonlinear equations which must be solved in order to satisfy the constraints at each timestep.

For method I above, we utilize the nonsymmetric sparse solver MA28 [31]. In this code, the nonzero structure is entered and manipulated in the so called *general sparse format.* Three vectors are required to represent this structure: an integer vector of IRN row indices, an integer vector JCN of column indices and a real vector VAL containing the matrix entries. The rows and columns of A are reordered according to the *Markowitz criterion* [32]. Let r_i and c_j give the number of nonzeros in row i and column j respectively. At each stage of Gaussian elimination, a new pivot entry a_{ij} is

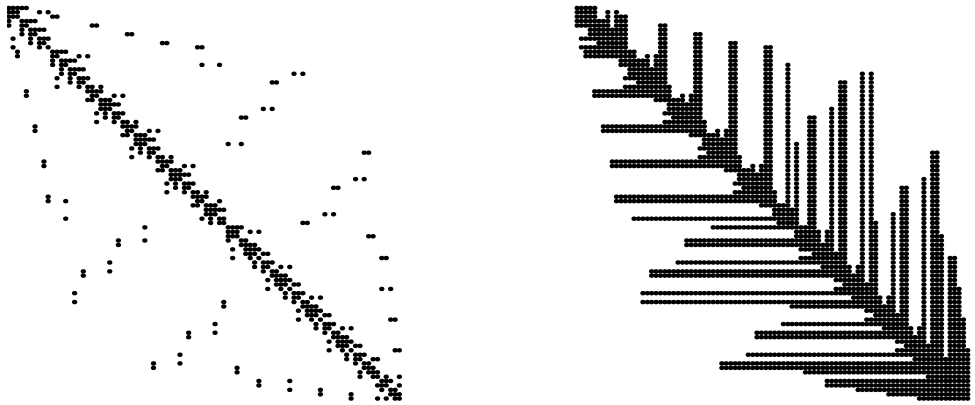
to be chosen which

1. is not too small numerically

2. minimizes $(r_i - 1)(c_j - 1)$.

The Markowitz criterion produces a pivot entry which approximately minimizes the fill-in at each step of Gaussian elimination. For the MD application, this ordering need be chosen only once, since the structure of the problem remains unchanged throughout the integration. For our problem, this ordering scheme has several weaknesses: it does not exploit structural stability and it compromises the quest for minimal fill-in by pivotting for numerical stability, which we suspect is not an issue here.

For method II we use the symmetric SPARSPAK code [33]. The nonzeros are stored in an *compressed* sparse data structure. The adjacency structure is contained in the integer array NADJ and ADJNCY , where NADJ(i) contains the number of bonds adjacent to bond i and ADJNCY(1:NADJ(1)) contains the indices of the bonds adjacent to bond 1, ADJNCY(NADJ(1)+1:NADJ(2)) contains the indices of the bonds adjacent to bond 2, etc. This information is used to order the equations according to the *minimum degree* scheme [32] which is the symmetric version of Markowitz ordering. In the symmetric positive definite case ordering can proceed without regard to numerical values of the entries. After reordering, the adjacency information is used to construct the compressed data structure: an integer vector XLNZ contains pointers to the first entry in each row, a real vector DIAG contains the diagonal entries. and a real vector LNZ contains all entries of the original matrix and space required for fill-in. Of particular importance for molecular dynamics applications, these ordering schemes produce no fill-in when applied to molecules with tree structure [33], such as BPTI in the absence of disulfide bonds. The reduction in fill-in gained from these schemes is exemplified by Figs. 4 and 5. Fig. 4 shows the sparsity structure of the matrix R for the C_{60} molecule with all 90 bonds constrained, before and after factorization with the equations ordered without regard to fill-in. For this natural ordering of the bond constraints, the factorization produces considerable growth in the number of nonzero entries. Fig. 5 shows the sparsity structure for the matrix before and after factorization, with the bond constraints reordered by the minimum degree scheme.



450 nonzero entries 2332 nonzero entries

FIG. 4. *Original ordering and fill-in , C_{60}.*

FIG. 5. *Minimum degree ordering and fill-in, C_{60}.*

5. Numerical experiments. The performance of the nonlinear equation solvers discussed above is illustrated by a series of short molecular dynamics runs for three systems: (i) a box of water molecules, (ii) bovine pancreatic trypsin inhibitor (BPTI, a small protein), and (iii) "buckminsterfullerene", a model of the C_{60} molecule with all bonds assumed equivalent. Systems (i) and (ii) are typical for molecular dynamics simulations in chemistry and biology [34, 8], while C_{60} and other fullerenes are currently attracting significant interest in materials science [35, 36, 37] and biology [38]. It is important to test the different constraint methods on such realistic examples.

The simulation results are presented below. In all cases, the nonbonded interactions were calculated using point charges and van der Waals parameters from the CHARMM Version 22 parameter set. A 12 Å nonbonded cutoff distance was employed, with a switching function between 10 and 12 Å for van der Waals terms and a shift function at 12 Å for electrostatics, in order to eliminate discontinuities due to the cutoff [7]. All simulations were performed using the program CHARMM Version 22, modified to enable alternative distance constraint schemes as described in Appendix A.

5.1. Box of water molecules. A truncated cubic cell [39] of size 28.01 Å containing 367 water molecules was simulated with periodic boundary (N,V,E) conditions [39], using the TIP3P three–center model of Jorgensen [40]. The internal motions of each molecule were eliminated by introducing three constraints — two for the O—H bonds and one for the H···H distance, resulting in 1101 constraints altogether. An alternative algorithm for this model (SETTLE) was described in [41].

Starting from a structure previously equilibrated at 300 K with SHAKE constraints, a 10 ps simulation was performed with timestep of 0.002 ps using the various length constraint methods.

Results of the experiments are summarized in Table I. SHAKE tolerance 10^{-6} was used. (The nonlinear solver iterates until constraint error, in norm, is less than SHAKE tolerance.) Time spent preprocessing sparse data structures is given in the rows labeled "setup". The rows labeled "reset" give the total time spent solving the nonlinear equations throughout the simulation. The number of iterations for each method is given in the rows marked "its". The CPU time reported by CHARMM for

the entire simulation is given by "total". For SOR, the final relaxation parameter ω from the adaptive relaxation algorithm is given. For this example about 7% of the total computer time was spent on the SHAKE iteration, which is a small but non-negligible fraction. Thus, for solvated systems, exploring more efficient alternatives to the SHAKE algorithm could lead to some savings in computer time. SOR required less than 3% of the total time. We have seen that the entries of the matrix $\hat{R}$ vary with the angles between constraints. In this case, these angles are fixed, leading to identical iteration counts between SYMM, where $\hat{R}$ is updated at each integration step, and SNIP, where no updates are required. The SHAKE iteration can be seen to perform slowly compared to all other methods; SOR and SNIP/SYMM provide more than twofold speedup.

	SHAKE	SOR	SNIP	NIP
setup	0.08 s.	0.07 s.	6.95 s.	6.58 s.
reset	41.8 m.	17.4 m.	16.1 m.	33.4 m.
total	10.5 h.	10.1 h.	10.1 h	10.3 h.
its	21.7	9.91	5.0	2.0
omega	1.00	1.24		

TABLE I
Comparison of SHAKE, SOR, SNIP, and NIP for three-constraint water molecule.

5.2. BPTI. BPTI is a small protein of 58 amino acid residues. Simulations were performed on an all-hydrogen model consisting of 898 atoms in vacuum, using the CHARMM Version 22 all-hydrogen protein parameter set. The starting structure was taken from the Brookhaven Protein Data Bank [42, 43] from the file pdb4pti.ent [44]. After the positions of hydrogen atoms were generated [45], the coordinates were energy-minimized with 500 steps of the Adopted Basis Newton-Raphson method [7], subject to isotropic harmonic constraints on heavy atoms [46]. The unconstrained system was then heated from 0 to 300 K by 20 K increments during 5 ps (5000 steps of 1 fs), and equilibrated at 300 K for a further 5 ps. The final coordinates from this procedure, together with a random sample of velocities from a Maxwell–Boltzmann distribution at 300 K, were used to start a series of 500 fs simulations with various timesteps and constraint methods. Lengths of all 918 bonds were constrained. Fig. 6 shows the sparsity structure of the matrix R to be tightly banded. The outlying entries correspond to the three disulfide bonds in BPTI [47].

A detailed analysis of the different BPTI trajectories is given in Appendix B. The results indicate that all the constraint methods used here yield essentially the same trajectories, and that the deviations between trajectories are determined primarily by the value of the constraint tolerance parameter. Additionally, relations between total energy fluctuations and timestep were determined, which are useful for choosing the value of the timestep giving a desired accuracy of energy conservation.

Table II gives results for dynamics runs with timesteps of $h = 0.0005, 0.001, 0.002,$ 0.004 picoseconds, using SHAKE tolerance 10^{-6}. Rows labeled "setup" refer to the preprocessing time for the sparse matrix data structures. The total time spent solving the nonlinear equations (coordinate resetting) in the course of the simulation is given in the rows labeled "reset". The total CPU time reported by CHARMM is given by "total". The average number of iterations required per timestep is given in the rows

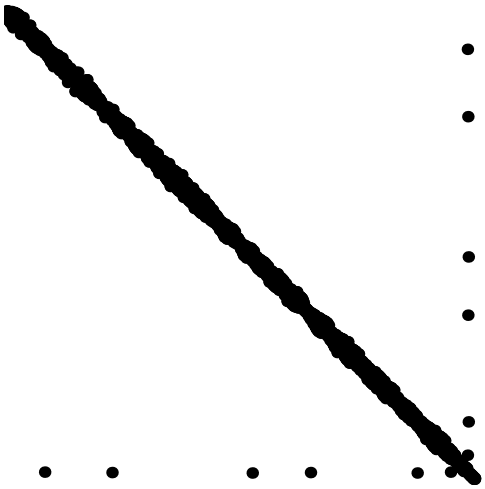


FIG. 6. *Sparsity structure for BPTI with disulfide bonds.*

labeled "its". For the SOR schemes, the final relaxation parameter ω is given. The column labeled "SOR+" gives results of the SOR iteration with adaptive relaxation algorithm and initial approximate solutions obtained from the Lagrange multipliers from the previous step. The success of this strategy depends on the timestep size, with diminished performance for larger steps. When improvements are realized, the time savings are mitigated due to the extra computation required to accumulate the Lagrange multipliers during the SOR iteration for use at the next integration step. Due to the nearly tree-like structure of BPTI, very little fill-in is introduced. This results in fast matrix factorization. Consequently SYMM, in which the number of iterations is kept low by refactorization at every timestep, performs well — slightly faster than SNIP where matrix updates were not performed. NIP gives the results for the Newton-Raphson iteration. Notice that half the number of iterations of SNIP are required in the latter case, yet more time was spent in coordinate resetting by a factor of between 2 and 3. The fraction of computation time spent on coordinate resetting with the SHAKE iteration was substantial for this system — approximately 17 percent.

5.3. "Buckminsterfullerene". We also simulated "buckminsterfullerene", a simplified model of the C_{60} molecule in which all bonds have been assumed equivalent. The potential energy parameters were chosen to be intermediate between those for aliphatic and aromatic carbons in the CHARMM Version 22 all-hydrogen protein parameter set. In all of the 90 bond energy terms of form $\frac{1}{2}k_b(b-b_0)^2$, we used k_b = 560.0 kcal/(mol·Å^2) and b_0 = 1.44 Å . In all of the 180 angle energy terms of form $\frac{1}{2}k_\theta(\theta-\theta_0)^2$, we used k_θ = 140.0 kcal/(mol·rad^2) and $\theta_0 = 120.0°$. The van der Waals interatomic interactions were described by a Lennard-Jones function of distance R: $\epsilon\left[(R_0/R)^{12} - 2\,(R_0/R)^6\right]$ with R_0=3.9848 Å and ϵ = 0.07 kcal/mol. Starting from a set of points on a unit sphere generated by the algorithm `bucky.m` from the numeric and visualization software package MATLAB [27], the coordinates were energy minimized using 1000 steps of the steepest descent algorithm, followed by a phase of heating to 300 K and equilibration. The final coordinates from this procedure,

timestep		SHAKE	SOR	SOR+	SNIP	SYMM	NIP
0.0005	its	19.0	9.25	7.89	3.58	2.03	1.01
	reset	4.62 m.	2.25 m.	2.18 m.	2.53 m.	2.52 m.	4.36 m.
	total	30.6 m.	28.2 m.	28.1 m.	28.7 m.	28.5 m.	30.4 m.
	setup	0.07 s.	0.07 s.	0.05 s.	6.39 s.	6.28 s.	6.07 s.
	omega	1.00	1.17	1.01			
0.001	its	21.02	10.4	7.80	4.95	3.01	2.10
	reset	2.8 m.	1.34 m.	1.19 m.	1.70 m.	1.53 m.	4.25 m.
	total	16.3 m.	14.8 m.	14.6 m.	15.3 m.	15.1 m.	17.8 m.
	setup	0.07 s.	0.07 s.	0.07 s.	6.38 s.	6.13 s.	5.98 s.
	omega	1.00	1.19	1.26			
0.002	its	22.1	11.7	9.86	6.20	3.96	2.02
	reset	1.59 m.	0.785 m.	0.735 m.	1.05 m.	0.933 m.	2.13 m.
	total	8.86 m.	8.02 m.	7.97 m.	8.47 m.	8.34 m.	9.50 m.
	setup	0.07 s.	0.07 s.	0.05 s.	6.32 s.	6.38 s.	6.17 s.
	omega	1.00	1.20	1.15			
0.004	its	23.6	15.3	11.8	7.78	5.28	2.96
	reset	0.872 m.	0.463 m.	0.50 m.	0.636 m.	0.560 m.	1.56 m.
	total	4.98 m.	3.76 m.	3.80 m.	4.82 m.	4.75 m.	5.78 m.
	setup	0.05 s.	0.07 s.	0.05 s.	6.40 s.	6.22 s.	5.98 s.
	omega	1.00	1.14	1.22			

TABLE II

Comparison of SHAKE,SOR,SNIP,SYMM, and NIP for BPTI , tol=10^{-6}.

together with a random sample of velocities from a Maxwell–Boltzmann distribution at 300 K, were used to start a series of 0.5 ps simulations with different timesteps and constraint methods. 90 length constraints were imposed. Fig. 4 shows the sparsity structure of the matrix R. The adjacency structure in the bonds can be seen to broaden the bandwidth, compared to the matrix for BPTI.

Table III gives results for 5 fs dynamics runs with timesteps of $h = 0.0005, 0.001$, $0.002, 0.004$ picoseconds, using SHAKE tolerance 10^{-6}. Increased connectivity in this example leads to increased fill-in during matrix factorization and hence more costly matrix factorizations. For this reason SNIP, which avoids repeated factorizations, performs faster than SYMM at each choice of timestep. For larger timesteps, SNIP gives the fastest coordinate resetting among all methods considered here. For the largest timestep, SOR gives no improvement over SHAKE iteration. In this example, nearly half of the total computation time was spent on coordinate resetting with the SHAKE iteration. It would thus be advantageous to use sparse matrix methods to implement bond constraints in systems with complicated bonding topologies.

6. Conclusion. In this paper we have studied systems of nonlinear equations which arise at each step of a popular discretization scheme in constrained molecular dynamics. When solutions of these equations exist they are not unique, but the desired solution can be defined uniquely by analytical continuation. We have pointed out that SHAKE iteration, the standard solver for such equations, is equivalent to a nonlinear Gauss-Seidel iteration and hence convergent for sufficiently small timesteps.

timestep		SHAKE	SOR	SNIP	SYMM	NIP
	its	10.3	8.78	2.00	2.00	1.00
	reset	13.1 s.	11.7 s.	12.9 s.	27.2 s.	51.9 s.
0.0005	total	38.4 s.	37.22 s.	38.6 s.	52.6 s.	1.29 m.
	setup	0.00 s.	0.00 s.	0.08 s.	0.12 s.	0.60 s.
	omega	1.00	1.17			
	its	14.1	13.3	2.97	2.01	1.01
	reset	10.3 s.	9.2 s.	9.07 s.	14.4 s.	26.0 s.
0.001	total	24.8 s.	23.3 s.	23.5 s.	28.7 s.	40.7 s.
	setup	0.00 s.	0.00 s.	0.15 s.	0.10 s.	0.58 s.
	omega	1.00	1.05			
	its	18.3	17.1	3.10	2.24	2.00
	reset	7.90 s.	7.18 s.	4.67 s.	7.35 s.	24.7 s.
0.002	total	16.3 s.	16.1 s.	13.4 s.	15.9 s.	33.7 s.
	setup	0.00 s.	0.00 s.	0.12 s.	0.12 s.	0.60 s.
	omega	1.00	1.05			
	its	23.0	23.4	4.06	3.04	2.04
	reset	4.68 s.	5.53 s.	3.15 s.	4.27 s.	13.4 s.
0.004	total	10.6 s.	11.1 s.	9.07 s.	10.8 s.	19.5 s.
	setup	0.00 s.	0.00 s.	0.12 s.	0.10 s.	0.60 s.
	omega	1.00	.98			

TABLE III
Comparison of SHAKE,SOR,SNIP,SYMM and NIP for C_{60}, tol = 10^{-6}.

In constrained molecular dynamics simulations, substantial computer time is spent on the SHAKE iteration. Further, SHAKE and other iterative methods are not particularly amenable to vector or parallel treatments. Hence we were lead to explore more efficient alternatives to this method. Successive overrelaxation is a natural improvement of Gauss-Seidel which exaggerates the correction at each iteration by some relaxation factor. We presented an adaptive relaxation algorithm which iteratively determines the optimal relaxation factor over the course of the integration. Experiments have shown SOR with the adaptive algorithm can offer up to twofold speed-up over to SHAKE iteration and compares favorably to all methods considered here.

Several matrix methods for the nonlinear equations were discussed: Newton iteration implemented with sparse matrix techniques and symmetric approximate Newton methods (SYMM and SNIP) formulated to avoid matrix updates and refactorizations by exploiting the special structure of the nonlinear equations. The symmetric Newton methods were found to be much faster than the SHAKE iteration in all cases, and were in some instances the fastest among all methods.

Acknowledgements. The authors would like to thank Martin Karplus for providing us with the unpublished CHARMM version 22 parameter set, and Ralph Byers for many helpful conversations. The calculations were performed on an IBM RS6000-550 workstation at the Departments of Chemistry and Biochemistry, University of Kansas. We would like to acknowledge the Kansas Institute for Theoretical and Computational Science for fostering the interdisciplinary collaboration which made this work possible.

REFERENCES

[1] W.F. van Gunsteren and M. Karplus. *Macromolecules*, 15:1528–1544, 1982.

[2] D. Tobias and C. Brooks. *J. Chem. Phys.*, 89:5115–5127, 1988.

[3] J.P. Ryckaert, G. Ciccotti, and H.J.C. Berendsen. *J. Comp. Phys*, 23:327–341, 1977.

[4] H.C. Andersen. *J. Comp. Phys.*, 52:24–34, 1983.

[5] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, 1970.

[6] J.P. Ryckaert. In C. R. A. et al. Catlow, editor, *Computer Modeling of Fluids Polymers and Solids*, pages 189–201.

[7] B. R. Brooks, R. Bruccoleri, B. Olafson, D. States, S. Swaminathan, and M. Karplus. *J. Comp. Chem.*, 4:187–217, 1983.

[8] C. L. Brooks III, M. Karplus, and B. M. Pettitt. *Proteins: A Theoretical Perspective of Dynamics, Structure, and Thermodynamics.* John Wiley & Sons, New York, 1988.

[9] W.F. van Gunsteren and H.J.C. Berendsen. *Molecular Physics*, 34:1311–1327, 1977.

[10] J.P. Ryckaert. *Molecular Physics*, 55:549–556, 1985.

[11] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations.* North-Holland, 1989.

[12] L. Verlet. *Phys. Rev.*, 159:98–103, 1967.

[13] E. Hairer, S. Norsett, and G. Wanner. *Solving Ordinary Differential Equations*, volume 1. Springer, 1989.

[14] V.I. Arnold. *Mathematical Methods of Classical Mechanics*, volume 60 of *Springer Graduate Texts in Mathematics.* Springer-Verlag, 1975.

[15] D.I. Okunbor and R.D. Skeel. *J. Comp. Chem.*, 15:72, 1994.

[16] B. Leimkuhler and R.D. Skeel. *J. Comp. Phys.* to appear.

[17] J.M. Sanz-Serna. *Acta Numerica*, 1:243–286, 1992.

[18] B. Leimkuhler and S. Reich. to appear.

[19] L. Jay. Technical report, Université de Genève, 1993.

[20] S. Reich. Technical Report 93-13, University of British Columbia, 1993.

[21] H. Yoshida. *Phys. Lett. A.*, 150:262–268, 1990.

[22] F. Ficken. *Comm. Pure Appl. Math.*, 14:435–456, 1951.

[23] J.H. Avila. *SIAM J. Num. Anal.*, 11:102–122, 1974.

[24] E. B. Wilson Jr., J. C. Decius, and P.C. Cross. *Molecular Vibrations.* McGraw-Hill, New York, 1955.

[25] K. Singal and J. Vlach. In A.E. Ruehli, editor, *Circuit Analysis, Simulation and Design Vol. 3*, Amsterdam, 1986. Elsevier, North-Holland.

[26] R. Varga. *Matrix Iterative Analysis.* Prentice-Hall, 1962.

[27] MATLAB. High-performance numeric computation and visualization software., 1992. The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760.

[28] H. Kroto. *Science*, 242:1139–1145, 1988.

[29] D.M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, 1971.

[30] E. Barth. PhD thesis, University of Kansas, 1994. In preparation.

[31] I.S. Duff. Technical Report AERE R8730, HMSO, 1986.

[32] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices.* Oxford, 1986.

[33] A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive-Definite Systems.* Prentice-Hall, 1981.

[34] W. F. van Gunsteren and H. J. C. Berendsen. *Angew. Chem. Int. Ed. Engl.*, 29:992–1023, 1990.

[35] *Nature*, 366:123, 1993.

[36] P. J. Fagan et al. *Science*, 262:404, 1994.

[37] *Energy and Fuels*, 7:685, 1993.

[38] S. H. Friedman et al. *J. Am. Chem. Soc.*, 115:6506 & 6510, 1993.

[39] M. P. Allen and D. J. Tildesley. *Computer Simulations of Liquids.* Oxford Science Publications, London, 1987.

[40] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein. *J. Chem. Phys.*, 79:926–935, 1983.

[41] S. Miyamoto and P.A. Kollman. *J. Comp. Phys.*, 52:24–34, 1983.

[42] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer Jr., M.D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. *J. Mol. Biol.*, 112:535–542, 1977.

[43] E. E. Abola, F. C. Bernstein, S. H. Bryant, T. F. Koetzle, and J. Weng. In F. H. Allen, G. Berg-

erhoff, , and R. Sievers, editors, *Crystallographic Databases - Information Content, Software Systems, Scientific Applications*, pages 107–132. Data Commission of the International Union of Crystallography, Bonn/Cambridge/Chester, 1987.
[44] R. Huber, D. Kukla, A. Ruehlmann, O. Epp, H. Formanek, J. Deisenhofer, and W. Steigemann. *Brookhaven Protein Data Bank*, 1982.
[45] A. T. Brunger and M. Karplus. *Protein*, 4:148–156, 1988.
[46] B. R. Bruccoleri and M. Karplus. *J. Comput. Chem.*, 7:165–175, 1986.
[47] T. E. Creighton. *Proteins. Structures and Molecular Properties.* W. H. Freeman, New York, 1993.
[48] J. L. Lebowitz, J. K. Percus, and L. Verlet. *Phys. Rev.*, 153:250–254, 1967.

A. Appendix: Modifications to CHARMM. The modifications to the nonlinear equations solver described in this paper have been implemented in CHARMM. These methods can be selected on the SHAKE command line:

`SHAKE BOND`	SHAKE iteration
`SHAKE SOR BOND`	SOR iteration
`SHAKE NIP BOND`	Newton iteration
`SHAKE SNIP BOND`	Adaptive Symmetric Newton iteration
`SHAKE SYMM BOND`	Symmetric Newton Iteration

For SOR, only slight modification to the code is involved. Several global variables are added to the common block `SHAKE.fcm` . Parameter `oldNITER` contains the iteration count from the previous timestep for comparison with the present value `NITER`. `OMEGA` and `DOMEGA` contain the current relaxation parameter and the next change to be made to `OMEGA` in convergence speed continues to improve. Updates of `OMEGA` occur in the last few lines of the subroutine `SHAKEA`.

A number of new arrays are required for the implementation of the sparse matrix methods. Memory is allocated from the `HEAP` using the `ALLHP` command in the subroutine `SHKSET`. We list the variable names and size for each method in Tables IV and V. The parameter `NCONST` gives the number of constraints to be imposed.

Array Name	Array Size	Type
IVECT	10*NCONST	integ4
JVECT	10*NCONST	integ4
IKEEP	5*NCONST	integ4
IW	8*NCONST	integ4
W	NCONST	real8
RHS	NCONST	real8
VEC2	NCONST	real8
IRN	5*NCONST	integ4
JCN	5*NCONST	integ4
A	5*NCONST	real8

TABLE IV
NIP storage overhead.

The symmetric SPARSPAK code requires roughly half the storage of its nonsymmetric counterpart. The array containing the values of the matrix entries for the symmetric code has length `MAXLNZ+1`, where the parameter `MAXLNZ` is computed dur-

Array Name	Array Size	Type
IVECT	10*NCONST	integ4
JVECT	10*NCONST	integ4
IKEEP	5*NCONST	integ4
IW	8*NCONST	integ4
W	NCONST	real8
RHS	NCONST	real8
VEC2	NCONST	real8
IRN	5*NCONST	integ4
JCN	5*NCONST	integ4
A	5*NCONST	real8

TABLE V
SYMM/SNIP storage overhead.

ing the symbolic factorization routine `SMBFCT`. `MAXLNZ` is typically 5 to 8 times larger than `NCONST`.

In the nonsymmetric code, the length of `IRN`, `JCN`, and `A` is chosen somewhat arbitrarily. These arrays must of course be sufficiently large to contain all nonzero entries after fill-in, but extra length enables the code to run more efficiently by reducing the need for memory swapping [31].

When one of the matrix methods is selected by the SHAKE command, the bond list `SHKAPR` to used to determine the adjacency structure of the dual graph. This is accomplished by a call to the routine `bondaj` in the nonsymmetric case, and the pair of routines `fnxadj` and `fnadje` in the symmetric case.

In addition to factoring and solving systems of linear equations, the sparse codes also must evaluate the constraint equations, update the coefficient matrix R of (26) and reset the coordinates as in (22). The constraint equations are evaluated at each iteration by the subroutine `g.f` in a straightforward way. The coefficient matrix is updated when necessary by the routines `mdij.f` and `asymij.f` for the symmetric and nonsymmetric cases respectively. The bond adjacency structure discussed above is used so that R can be evaluated without the need to form the matrices g', M, or G^t explicitly. Coordinate resetting

$$q_{new} = q_{old} + G^t \Delta\Lambda$$

is accomplished at each iteration by the routine `ggaxpy.f` which also makes use of the bond structure to avoid the explicit formation of G^t.

B. Appendix: Comparison of constrained trajectories for BPTI. For the case of BPTI we have performed detailed comparisons of MD trajectories generated by the different constraint algorithms. In our comparisons we used the following quantities: root-mean-square (rms) deviations between final coordinates from the trajectories, average total energy and total energy rms fluctuations, average temperature and temperature rms fluctuations. For two molecular structures a and b, the rms deviation is defined as:

$$RMSD_{ab} = \left(\frac{1}{N} \sum_{i=1}^{N} [(\vec{r}_i)_a - (\vec{r}_i)_b]^2 \right)^{1/2}, \tag{27}$$

where N is the number of atoms and $(\vec{r}_i)_a$ and $(\vec{r}_i)_b$ are the positions of atom i in structures a and b, respectively. This quantity informs us about the average difference between two sets of atomic coordinates of a molecular system; the RMSD is zero for identical structures. The total energy is a fundamental quantity in molecular simulations; it is conserved along the exact solution to the equations of motion. Differences in average values of total energy indicate that trajectories explore different regions of phase space, while the magnitudes of the total energy fluctuations inform about the deviation between the numerical and exact solutions to Newton's equations. The temperature in a molecular dynamics simulation is defined as

$$T = \frac{1}{fk} \sum_{i=1}^{N} m_i v_i^2, \tag{28}$$

where m_i and v_i are the mass and velocity of atom i, respectively, k is the Boltzmann constant and f is the number of degrees of freedom. For an N-atomic molecule in vacuum, for which we remove the translational motion of the center of mass and the angular momentum relative to the center of mass, and on which m constraints have been imposed, $f = 3N - m - 6$. Trajectories in which atoms move with different velocities will exhibit different values of average temperatures and temperature fluctuations. Temperature fluctuations in constant energy simulations are related to heat capacities [48].

To obtain a better understanding of the constraint algorithms we compare results of a series of 0.512 ps simulations starting from coordinates and velocities prepared as described in the § 5 under the following conditions: (i) unconstrained simulations with timesteps h = 0.5, 1 and 2 fs, (ii) simulations with only heavy atom-hydrogen bonds (X—H, X=C,N,O,S) constrained with h = 0.5, 1, 2 and 4 fs and constraint tolerance of 10^{-10}, denoted below by X—H, and (iii) simulations with all 918 bonds of BPTI (including the three disulfide bonds) constrained with h = 0.5, 1, 2 and 4 fs, with a constraint tolerance of 10^{-10}, denoted by ALL. Stable trajectories could not be generated for unconstrained simulations with $h \geq 4$ fs and constrained ones with $h \geq 8$ fs.

Differences in the trajectories can arise from two sources — simulation conditions and algorithm performance. We expect deviations between the unconstrained and constrained trajectories, since they correspond to different effective Hamiltonians and different numbers of degrees of freedom. Additionally, trajectories started from the same coordinates and velocities with different timesteps will differ. Thus, in order to examine the dependence of trajectories on the algorithm we have to compare simulations using the different constraint schemes and the same integration timestep.

The average temperatures and temperature fluctuations in the fully constrained simulations (ALL) were 303.8 ± 6.6 K, 303.1 ± 6.5 K, 300.1 ± 6.4 K and 288.3 ± 6.9 K, for timesteps h = 0.5, 1, 2, and 4 fs respectively. In the X—H simulations the corresponding values were 301.1 ± 6.2 K, 300.1 ± 6.1 K, 296.5 ± 6.2 K and 282.4 ± 6.1 K, for timesteps h = 0.5, 1, 2, and 4 fs respectively while in the unconstrained simulations values of 303.2 ± 5.3 K, 300.8 ± 5.6 K, and 291.8 ± 5.6 K, for timesteps h = 0.5, 1, and 2 fs respectively. For all the tested bond constraint algorithms (SHAKE, SOR, NIP, SNIP and SYMM) at a given value of h, the average temperatures and temperature fluctuations were the same within a relative error of 10^{-4} for ALL; for X—H all the available digits were identical, giving $\sim 10^{-6}$ as the upper limit of the differences. The higher values of temperature fluctuations in the constrained simulations

agree with the expected trend that lowering the number of degrees of freedom will decrease heat capacities and increase kinetic energy fluctuations [48]. The temperature fluctuations found here correspond to kinetic energy fluctuations of $\sim$ 10 kcal/mol in the unconstrained and 11-12 kcal/mol in the constrained simulations. The ratio of fluctuations of the total to the kinetic energy is usually used as a criterion of numerical stability/accuracy of MD simulations. These ratios were about 0.002 (0.004), 0.01 (0.02), 0.04 (0.07) and 0.2 (0.3) in the constrained ALL (X—H) simulations with h = 0.5, 1, 2, and 4 fs, respectively, and were 0.01, 0.06 and 0.4 in the unconstrained simulations with h = 0.5, 1, and 2 fs, respectively.

The average total energies and total energy fluctuations in the constrained ALL simulations were -170.92 $\pm$ 0.03, -170.81 $\pm$ 0.11, -170.04 $\pm$ 0.44 and -159.09 $\pm$ 2.66 kcal/mol, for timesteps h = 0.5, 1, 2, and 4 fs, respectively. In the X—H simulations the corresponding results were -41.64 $\pm$ 0.04, -41.54 $\pm$ 0.18, -40.53 $\pm$ 0.75 and -22.01 $\pm$ 3.79 kcal/mol, for timesteps h = 0.5, 1, 2, and 4 fs, respectively. For all the tested bond constraint algorithms (SHAKE, SOR, NIP, SNIP and SYMM) and at a given value of h, the average total energies and total energy fluctuations were the same within a relative error of 10^{-4} in the ALL simulations; in X—H all the available digits were identical, giving about 10^{-6} as the upper limit of the differences. For comparison, the corresponding values for the unconstrained simulations were 105.27 $\pm$ 0.13, 106.19 $\pm$ 0.64, and 120.19 $\pm$ 3.93, for timesteps h = 0.5, 1, and 4 fs, respectively. Successively eliminating the high frequency bond motions thus leads to improved energy conservation for a given timestep h. Fig. 7 shows the dependence of total energy fluctuations on time step in the constrained and unconstrained simulations. In all cases the dependence of energy fluctuations on h is approximately quadratic for small values of h.

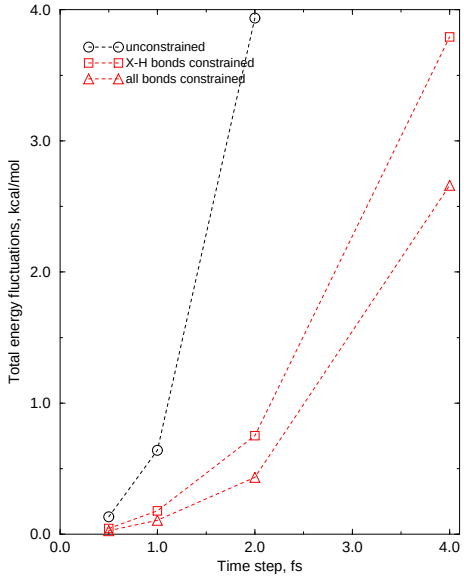


FIG. 7. *Total energy fluctuations in 512 fs simulations of BPTI.*

To further compare the performance of the different bond constraint algorithms, we have calculated the rms deviations between several classes of final structures from the 0.512 ps simulations. Comparison of the final coordinates from the constrained simulation using SHAKE with those of the other constrained algorithms (SOR, NIP, SNIP and SYMM), for a given integration timestep, yields RMSD values in the range

of 10^{-5}–10^{-6} Å in the ALL and 10^{-9}–10^{-10} Å in the X—H simulations, for all of the timesteps h = 0.5, 1, 2, and 4 fs employed. The corresponding trajectories are thus quite similar. Furthermore, the deviations between the trajectories appear to be determined primarily by the value of the constraint tolerance parameter, which was 10^{-6} for ALL and 10^{-10} for X—H. Fig. 8 shows the time evolution of the coordinate rms deviations between corresponding structures from simulations with all bonds constrained using SHAKE and SOR algorithms with all bonds constrained. Comparisons of other pairs of constraint methods yield qualitatively similar results.

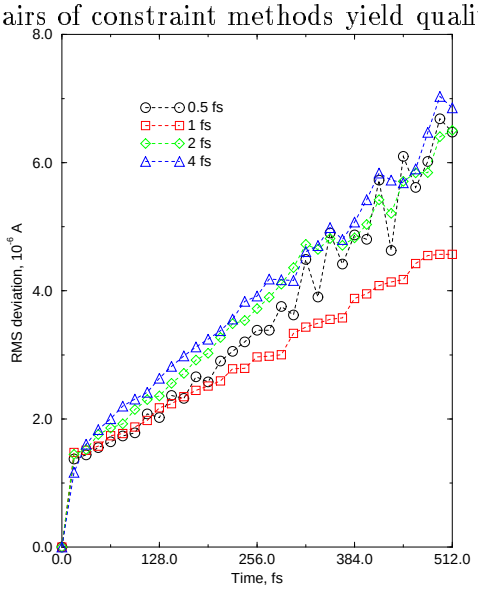


FIG. 8. *RMS deviations between SHAKE and SOR trajectory frames.*

RMS deviations between the final coordinates from the constrained and corresponding (i.e., with same h) unconstrained trajectories are about 0.3 Å . For each of the constrained methods, rms deviations between the final coordinates from simulations with h = 0.5 fs and those with the longer timesteps were 0.02 (0.11), 0.08 (0.09) and 0.12 (0.13) Å , in ALL (X—H) for h = 1, 2, and 4 fs, respectively.

In conclusion, analysis of the BPTI trajectories shows that all the considered bond constraint algorithms lead to practically identical trajectories, as measured by coordinate rms deviations, total energy and kinetic energy averages and fluctuations. The deviation between trajectories generated using different methods appear to be primarily determined by the value of the constraint tolerance parameter. Our numerical results confirm that adding bond constraints decreases total energy fluctuations (i.e., improves energy conservation) for a given timestep and increases temperature/kinetic energy fluctuations. Also, the ratios of total to kinetic energy fluctuations given here may be used to tailor timestep values to a given level of accuracy in both unconstrained and constrained simulations.

The longer term behavior of the different constraint algorithms can be analyzed on the example of the water box trajectories (Section 5.1). The trajectories consisted of 5000 steps of molecular dynamics using the Verlet algorithm and a 2 fs time step; all were started from the same structure equilibrated using SHAKE constraints and the same set of atomic velocities. The average total energies and their fluctuations were -3100.9 ± 1.0, -3101.4 ± 0.6, -3102.0 ± 0.5 and -3100.9 ± 1.47 kcal/mol for the SHAKE, SOR, SNIP and NIP algorithms, respectively, all with a constraint tolerance of 10^{-6}. The corresponding average temperatures were 307.8 ± 7.6 , 306.7 ± 7.6, 306.9 ± 7.8

and 307.0 ± 7.7. The basic descriptors of the different trajectories agree within their fluctuations; we can thus conclude that the four methods lead to essentially the same trajectories on the 10 ps time scale. It is encouraging that SOR and SNIP showed about a factor of two improved total energy conservation over SHAKE. To determine with certainty whether this is a general feature of the methods or a special case, further numerical experiments are needed. The ratios of fluctuations of total to kinetic energy were 0.06, 0.03, 0.03 and .10 for the SHAKE, SOR, SNIP and NIP algorithms, respectively. The values for SHAKE, SOR and SNIP are quite similar to the ratio of 0.04 found in all tested methods for 0.512 ps trajectories of BPTI with all bonds constrained and $h = 2$ fs.