Journal of Computational Physics **172,** 188–197 (2001)
doi:10.1006/jcph.2001.6819, available online at http://www.idealibrary.com on IDEAL®

A Generalized Non-iterative Matrix Method for Constraint Molecular Dynamics Simulations

Makoto Yoneya

Yokoyama Nano-structured Liquid Crystal Project, ERATO, Japan Science and Technology Corporation, 5-9-9 Tokodai, Ttsukuba, Ibaraki 300-2635, Japan
E-mail: yoneya@nanolc.jst.go.jp

Received April 6, 2000; revised April 6, 2001

A generalization of the previously proposed non-iterative matrix method (NIMM) for constraint molecular dynamics simulations is presented. The resultant generalized version of NIMM (gNIMM) makes possible the constraint force calculation with exactly the same procedure (subroutine) for various "Verlet equivalent" integration schemes with the same fourth order of the time step as the constraint error order. This gNIMM needs only the latest available configuration information and does not need extra memory and restart file size to store the one-step earlier configuration as in a previous extension of NIMM by J. Slusher and P. Cummings (1996, *Mol. Sim.* **18**, 213). The method is tested by simulations with systems of fullerenes, a protein, etc., and is three to five times faster than SHAKE for solving the constraints even in such rather large molecule simulations. © 2001 Academic Press

Key Words: constraint; SHAKE; Verlet method; bond length; constraind dynamics; molecular dynamics simulation; molecular simulation.

1. INTRODUCTION

Constraint dynamics places constraints on some intramolecular degrees of freedom, thus allowing integration time steps larger than those possible in ordinary dynamics. Since the introduction of the algorithm SHAKE [1] in the late 1970s, constraint dynamics simulations have been done by mostly applying this SHAKE algorithm. Although application of SHAKE is generally successful, efforts have been made to improve the constraint algorithm in its computational efficiency and parallelization. Some examples are the EEM method based on Gauss's principle of least constraint proposed by Edberg *et al.* [2]; LINCS [3], which is similar to the EEM, but solves the same non-linear constraint problem as SHAKE instead of derivatives of the constraint as in the EEM; and NIMM (non-iterative matrix method) by Yoneya *et al.* [4]. NIMM was recently extended by Slusher and Cummings [5] from its original combinations with Verlet and leap-frog to combinations with velocity Verlet and Beeman time integration schemes.

In this article, we present a more generalized extension of NIMM for so-called "Verlet equivalent" time integration schemes. We evaluate its computational efficiency and robustness compared to SHAKE with test simulations of fullerenes, a protein, rigid planar benzenes, etc.

2. CONSTRAINT ALGORITHMS

In this section, we show the basic idea of NIMM in comparison with SHAKE. We discuss only the bond length constraint because the other constraints, e.g., bond angle, are formally the same as that of bond length.

2.1. *Constraint with SHAKE*

Let us assume there are N bonds in a molecule. $\mathbf{x}_i, \mathbf{x}_j$ are the coordinates of two atoms which form the nth bond of this molecule and $\mathbf{d}_n$ is its fixed bond length. The total constraint condition of a molecule can be expressed as

$$\chi_n(t) \equiv \mathbf{r}_n^2(t) - \mathbf{d}_n^2 = 0 \quad (n = 1 - N),$$

where

$$\mathbf{r}_n(t) \equiv \mathbf{x}_i(t) - \mathbf{x}_j(t).$$

In SHAKE, the atomic coordinates are iteratively reset to fulfill the following equation as a result:

$$\chi_n(t + \Delta t) + O(\varepsilon) = 0. \tag{1}$$

The point of SHAKE is that the constraint force is calculated to fulfill the constraint condition (with user-specified error tolerance ε) at not the current, but the next time step $t + \Delta t$. It is known, that the time integration with constraint force which is calculated to fulfill the constraint condition at the current time step results in large drifts of constrained bond lengths [1]. This is because the time integration scheme itself has a limited accuracy and the accumulation of integration error causes the drifts. SHAKE effectively suppresses this drift by the point above. The method proposed by Edberg *et al.* [2] has no such error suppressing function itself and then it needs an independent error handler.

The constraint force $\boldsymbol{f}_i^c$ is expressed as follows using the Lagrange multiplier λ_n:

$$\boldsymbol{f}_i^c(t) = \frac{1}{2}\sum_{n=1}^{N} \lambda_n(t)\nabla_{r_i(t)}\chi_n(t) = \sum_{n=1}^{N} \lambda_n(t)\boldsymbol{r}_i(t). \tag{2}$$

Equation (2) means that the calculation of constraint force is equivalent to the calculation of the Lagrange multipliers. If we use this constraint force equation (2) in the constraint condition (1), it results in a non-linear (quadratic) equation about λ_n and then, needs an iterative procedure to solve λ_n. This is the reason why SHAKE becomes an iterative method.

2.2. *NIMM and Its Extension*

NIMM is a constraint algorithm which can only be combined with (up to) fourth error order time integration schemes, e.g., various "Verlet equivalent" time integration schemes.

The difference between NIMM and SHAKE is the equation which is solved in SHAKE (Eq. (1)) and the equation which is solved in NIMM,

$$\chi_n(t+\Delta t) + O(\Delta t^4) = 0; \tag{3}$$

i.e., the difference between them is simply the specification of the constraint errors. This means that the point which suppresses the constraint error accumulation is considered to be the same in both SHAKE and NIMM. In SHAKE, the constraint error tolerance ε is a user-specified parameter. On the other hand, the constraint error order $O(\Delta t^4)$ in NIMM is the error order of the combined time integration scheme. In constraint dynamics, these error orders of the constraint condition and time integration are not independent as in SHAKE, but related to each other. It is shown that the error orders of the constraint condition and time integration should be on the same order with the simple analysis in the original presentation of NIMM [4]. The major difference when solving Eq. (3) instead of Eq. (1) to obtain the Lagrange multipliers is the former can be solved non-iteratively because of the quadratic term about λ_n is omitted as $O(\Delta t^4)$ terms, in contrast to the latter which must be solved by the iterative procedure. This is the reason why NIMM becomes non-iterative and it must be combined with (up to) fourth error order time integration schemes.

In the original presentation of NIMM [4], the combinations with Verlet and the leap-frog time integration scheme were explained. Neither evaluates the atomic coordinates and velocities at the same timing and this causes inconvenience in some applications. Slusher and Cummings [5] extended NIMM, combining it with the velocity Verlet and Beeman time integration scheme in which the coordinate and velocity are evaluated at the same timing. Their actual algorithm is shown with an example of the velocity Verlet in

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \dot{\mathbf{x}}(t)\Delta t + \frac{1}{2}\ddot{\mathbf{x}}(t)\Delta t \tag{4}$$

$$\dot{\mathbf{x}}(t+\Delta t) = \dot{\mathbf{x}}(t) + \frac{1}{2}\{\ddot{\mathbf{x}}(t) + \ddot{\mathbf{x}}(t+\Delta t)\}\Delta t^2. \tag{5}$$

Slusher and Cummings proposed the following procedure which evaluates $\chi_n(t+2\Delta t)$ with an equation obtained by substituting Eq. (5) into Eq. (4) and shifting one time step Δt:

$$\begin{aligned}\chi_n(t+2\Delta t) + O(\Delta t^4) \;=\; & \{\mathbf{r}_n(t+\Delta t) + \dot{\mathbf{r}}_n(t)\Delta t\}^2 - \mathbf{d}_n^2 + \{\mathbf{r}_n(t+\Delta t) \\ & + \dot{\mathbf{r}}_n(t)\Delta t\}\cdot\{\ddot{\mathbf{r}}_n(t) + 2\,\ddot{\mathbf{r}}_n(t+\Delta t)\}\Delta t^2 = 0.\end{aligned} \tag{6}$$

Equation (6) utilizes one time step earlier values $\dot{\mathbf{r}}_n(t)$, $\ddot{\mathbf{r}}_n(t)$ which are usually not available at the timing of the constraint force evaluation because these are overwritten with the corresponding latest values. As a result, we need extra memory area for these earlier step values to calculate the constraint force using Eq. (6). On the other hand, the original NIMM for the leap-frog only utilizes the latest available values $\mathbf{r}_n(t)$, $\dot{\mathbf{r}}_n(t-\Delta t)$, $\ddot{\mathbf{r}}_n^{nc}(t)$ at the timing of the constraint force evaluation. Analogously, when we restart the constraint dynamics run from the previous configuration which is stored in a restart file, the file with the same contents as the non-constraint run, i.e., $\mathbf{x}(t)$, $\dot{\mathbf{x}}(t-\Delta t)$ is enough in the original

NIMM, whereas the file should additionally contain the one step earlier values in the case of Eq. (6).

3. GENERALIZED NIMM

3.1. *Alternative Extension of NIMM*

As in the previous section, the NIMM extension for the velocity Verlet by Slusher and Cummings needs extra memory and restart file size, which the original NIMM for the leap-frog does not. This problem can be solved with an alternative extension based on systematic generalization of NIMM in the following.

Let us begin with that the "Verlet equivalent" schemes are derived from the original Verlet scheme with the velocity definitions [6]

$$\dot{\mathbf{x}}(t + (\alpha - 1/2)\Delta t) \equiv \frac{\mathbf{x}(t + \alpha\Delta t) - \mathbf{x}(t + (\alpha - 1)\Delta t)}{\Delta t}. \tag{7}$$

The corresponding rewritten Verlet form is

$$\dot{\mathbf{x}}(t + (\alpha - 1/2)\Delta t) = \dot{\mathbf{x}}(t + (\alpha - 3/2)\Delta t) + \ddot{\mathbf{x}}(t + (\alpha - 1)\Delta t)\Delta t + O(\Delta t^3). \tag{8}$$

The cases $\alpha = 1$, $3/2$, and 2 correspond to the leap-frog, the position Verlet [8], and the velocity Verlet integration scheme, respectively. From these equations, we get

$$\begin{aligned}\mathbf{x}(t + \alpha\Delta t) &= \mathbf{x}(t + (\alpha - 1)\Delta t) + \dot{\mathbf{x}}(t + (\alpha - 3/2)\Delta t)\Delta t \\ &\quad + \ddot{\mathbf{x}}(t + (\alpha - 1)\Delta t)\Delta t^2 + O(\Delta t^4).\end{aligned}$$

This means the error order of the coordinate integration is the same as that of original Verlet, i.e., $O(\Delta t^4)$. We will show the corresponding NIMM formulation in the following.

First, the velocity Verlet procedures for two time steps are listed in a different form with Eqs. (4), (5) as follows:

$$\begin{aligned}\dot{\mathbf{x}}(t + \Delta t/2) &= \dot{\mathbf{x}}(t) + \ddot{\mathbf{x}}(t)(\Delta t/2) \\ \mathbf{x}(t + \Delta t) &= \mathbf{x}(t) + \dot{\mathbf{x}}(t + \Delta t/2)\Delta t \\ \dot{\mathbf{x}}(t + \Delta t) &= \dot{\mathbf{x}}(t + \Delta t/2) + \ddot{\mathbf{x}}(t + \Delta t)(\Delta t/2) \\ \dot{\mathbf{x}}(t + (3/2)\Delta t) &= \dot{\mathbf{x}}(t + \Delta t) + \ddot{\mathbf{x}}(t + \Delta t)(\Delta t/2) \\ \mathbf{x}(t + 2\Delta t) &= \mathbf{x}(t + \Delta t) + \dot{\mathbf{x}}(t + (3/2)\Delta t)\Delta t \\ \dot{\mathbf{x}}(t + 2\Delta t) &= \dot{\mathbf{x}}(t + (3/2)\Delta t) + \ddot{\mathbf{x}}(t + (3/2)\Delta t)(\Delta t/2).\end{aligned}$$

By substituting the first equation of the second step into the last equation of the first step, we obtain the following:

$$\dot{\mathbf{x}}(t + (3/2)\Delta t) = \dot{\mathbf{x}}(t + \Delta t/2) + \ddot{\mathbf{x}}(t + \Delta t)\Delta t \quad (+O(\Delta t^3)).$$

This is Eq. (8) with $\alpha = 2$ and Eq. (7) with $\alpha = 2$ corresponds the second equation of the second step. Substituting the above into this second equation of the second step gives

$$\mathbf{x}(t + 2\Delta t) = \mathbf{x}(t + \Delta t) + \dot{\mathbf{x}}(t + \Delta t/2)\Delta t + \ddot{\mathbf{x}}(t + \Delta t)\Delta t^2 \quad (+O(\Delta t^4)).$$

Evaluating $\chi_n(t+2\Delta t)$ with this equation makes the constraint condition for $t+2\Delta t$

$$\chi_n(t+2\Delta t)+O(\Delta t^4)=\mathbf{q}_n^2(t+\Delta t)-\mathbf{d}_n^2+2\mathbf{q}_n(t+\Delta t)\cdot\ddot{\mathbf{r}}(t+\Delta t)\Delta t^2=0, \tag{9}$$

where $\mathbf{q}_n(t+\Delta t)$ is defined as

$$\mathbf{q}_n(t+\Delta t)\equiv\mathbf{r}_n(t+\Delta t)+\dot{\mathbf{r}}_n(t+\Delta t/2)\Delta t.$$

Equation (9) utilizes only the latest available values at the timing of the constraint force evaluation (between the second and the third equations of the first time step), and does not require the one step earlier values as in Eq. (6).

Next, the restart file size is discussed. Usually, the restart file contains atomic coordinates and velocities and these are actually $\mathbf{x}(t+\Delta t)$ and $\dot{\mathbf{x}}(t+\Delta t)$ in the case of the two-step expression of the velocity Verlet above. In this case, at the first step after restart, $\dot{\mathbf{x}}(t+(3/2)\Delta t)$ cannot be calculated due to a lack of $\ddot{\mathbf{x}}(t+\Delta t)$ which includes constraint force. To solve it, the values contained in the restart file should be the latest values which can be calculated at the last time step. In the case of the velocity Verlet, $\mathbf{x}(t+\Delta t)$ and $\dot{\mathbf{x}}(t+(3/2)\Delta t)$ should be stored as a restart file and a continuous run becomes possible using such a restart file without extra restart file size.

3.2. *Generalization of NIMM*

In the previous section, the NIMM was extended to the combinations with the velocity Verlet. Extension to the combinations with the position Verlet is also possible in a similar manner. We found that the equations to obtain the constraint force take the general form

$$\begin{aligned}\chi_n(t+\alpha\Delta t)+O(\Delta t^4)&=\mathbf{q}_n^2(t+(\alpha-1)\Delta t)-\mathbf{d}_n^2\\&\quad+2\mathbf{q}_n(t+(\alpha-1)\Delta t)\cdot\ddot{\mathbf{r}}(t+(\alpha-1)\Delta t)\Delta t^2=0 \qquad (10)\\ \mathbf{q}_n(t+(\alpha-1)\Delta t)&\equiv\mathbf{r}_n(t+(\alpha-1)\Delta t)+\dot{\mathbf{r}}_n(t+(\alpha-3/2)\Delta t)\Delta t,\end{aligned}$$

where

$$\begin{aligned}\alpha&=1 && \textit{for Verlet using } \dot{\boldsymbol{r}}_n(t-\Delta t/2)\Delta t=\boldsymbol{r}_n(t)-\boldsymbol{r}_n(t-\Delta t)\\&=1 && \textit{for leap-frog}\\&=3/2 && \textit{for position Verlet}\\&=2 && \textit{for velocity Verlet.}\end{aligned}$$

This means exactly the same subroutine (to solve Eq. (10)) can be used to calculate constraint force for all these "Verlet equivalent" time integration schemes (with $O(\Delta t^4)$ constraint error order). Corresponding constraint force is obtained by Eq. (2) as follows:

$$\begin{aligned}f_i^c(t+(\alpha-1)\Delta t)&=\frac{1}{2}\sum_{n=1}^{N}\lambda_n(t+(\alpha-1)\Delta t)\nabla_{r_i(t+(\alpha-1)\Delta t)}\chi_n(t+(\alpha-1)\Delta t)\\&=\sum_{n=1}^{N}\lambda_n(t+(\alpha-1)\Delta t)\boldsymbol{r}_n(t+(\alpha-1)\Delta t).\end{aligned} \tag{11}$$

Here, the Lagrange multiplier $\lambda_n(t + (\alpha - 1)\Delta t)$ is actually obtained by Eq. (10), i.e., not $\chi_n(t + (\alpha - 1)\Delta t) + O(\Delta t^4) = 0$ but $\chi_n(t + \alpha\Delta t) + O(\Delta t^4) = 0$, and then can be considered as $\lambda'_n(t + \alpha\Delta t)$. Corresponding to this, if we evaluate χ_n in Eq. (11) with Eq. (10), we get

$$\begin{aligned} f_i^c(t + (\alpha - 1)\Delta t) &= \frac{1}{2}\sum_{n=1}^{N} \lambda'_n(t + \alpha\Delta t)\frac{\partial \boldsymbol{q}_i}{\partial \boldsymbol{r}_i}\nabla_{q_i(t+(\alpha-1)\Delta t)}\chi_n(t + \alpha\Delta t) \\ &= \sum_{n=1}^{N} \lambda'_n(t + \alpha\Delta t)\{\boldsymbol{q}_n(t + (\alpha - 1)\Delta t) + \ddot{\boldsymbol{r}}(t + (\alpha - 1)\Delta t)\Delta t^2\}. \end{aligned} \tag{12}$$

Here, the $\ddot{\boldsymbol{r}}(t + (\alpha - 1)\Delta t)\Delta t^2$ term of the above equation is omitted as $O(\Delta t^4)$ term in the calculation of the Lagrange multipliers with Eq. (10). In the combination of Eqs. (12) and (10), the matrix to solve the Lagrange multipliers becomes symmetric, whereas the corresponding matrix is asymmetric if we combinate Eqs. (11) and (10). Symmetric matrix solvers are generally faster than asymmetric ones and for this reason, the former combination is practically preferable in gNIMM.

4. TEST SIMULATION RESULTS

4.1. *Error Orders*

In the previous section, NIMM was generalized to various "Verlet equivalent" time integration schemes. In this section, we checked this generalized NIMM (gNIMM) with simple test simulations. The tested system was a 32 *n*-butane molecule (only bond constraint was applied) system under a periodic boundary condition which is the same as in the test of the original NIMM paper. Figure 1 shows the dependency of R.M.S. constraint error ($\langle\chi_n^2\rangle^{1/2}$ evaluated with Eq. (10) and $\langle\ \rangle$ stands for average over bonds and time steps) on time integration step Δt for various "Verlet equivalent" schemes with the same subroutine (results of the original NIMM are also shown).

The constraint errors are the value at the timing of the constraint force calculations of each integration scheme, e.g., $\chi(t + \Delta t)$ for the leap-frog, $\chi(t + (3/2)\Delta t)$ for the position Verlet. The gNIMM constraint errors of all these "Verlet equivalent" integration scheme are indistinguishable and all $O(\Delta t^4)$.

4.2. *Timing Results*

Then, to compare NIMM with SHAKE in its computational efficiency, we did test simulations of various molecular systems.

The first one was a system of 64 liquid crystal 5CB (4-*n*-pentyl-4′-cyanobiphenyl) molecules [9] (total: 1216 atoms including united atoms) under a periodic boundary condition at isotropic states. A time step of 2fs was used in a total of 200 steps of constant energy simulations with bond length constraint. The timing results were obtained by Hewlett Packard HP9000/735 workstation and they are summarized in Table 1 with the corresponding R.M.S. constraint errors.

In the constraint force calculation part, the CPU time of gNIMM is almost the same as that of NIMM and about five times faster than SHAKE (note that the current implementation

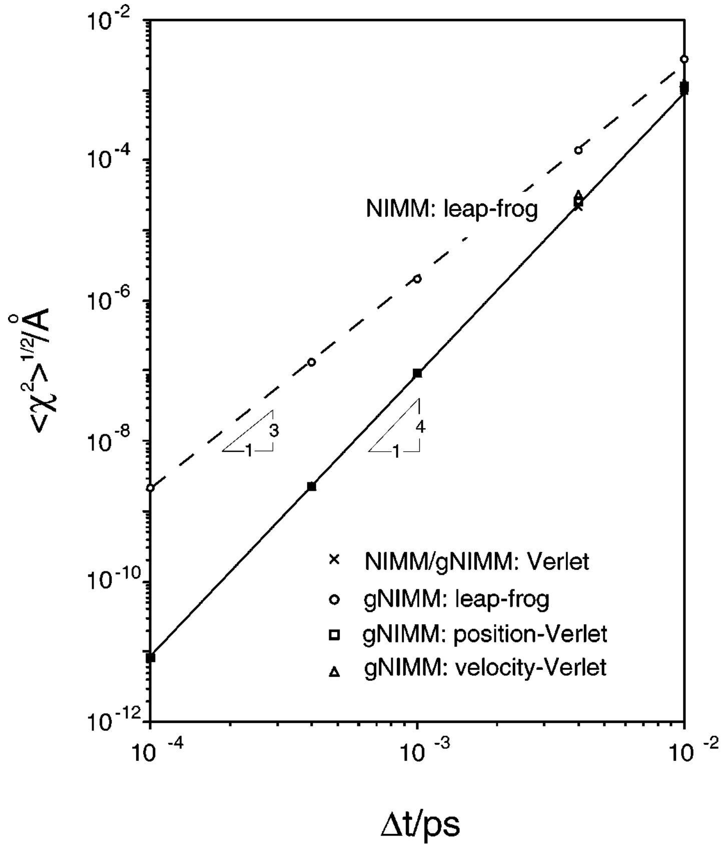

FIG. 1. Constraint error dependency on time step size.

of these constraint routines is not extensively tuned). In these test simulations, R.M.S. constraint errors and energy fluctuations are the same order of magnitude for all methods. The CPU times of gNIMM with different time integration schemes are negligibly different because all were calculated with exactly the same subroutines.

The extended run (with gNIMM and leap-frog) up to 50,000 time steps on this 5CB system was done to check the long-term stability. In Fig. 2, we plot the instantaneous fluctuations of the energy about the initial value, i.e., $\Delta E(t)/E(0) = (E(t) - E(0))/E(0)$ [8], with the corresponding result with SHAKE. The total energy was conserved with 0.5% fluctuation and R.M.S. constraint errors were stably kept on the order of 10^{-5} Å.

TABLE 1
CPU Seconds for the 5CB System (200 Steps)

Const. method	Integration method	Const. CPU	Total CPU	Const. error $\langle\chi_n^2\rangle^{1/2}$
SHAKE	leap-frog	7.38	67.9	3.5×10^{-5}
NIMM	leap-frog	1.20	63.7	2.5×10^{-5}
gNIMM	leap-frog	1.41	62.8	2.0×10^{-5}
gNIMM	pos. Verlet	1.38	66.0	1.8×10^{-5}
gNIMM	vel. Verlet	1.43	63.0	2.5×10^{-5}

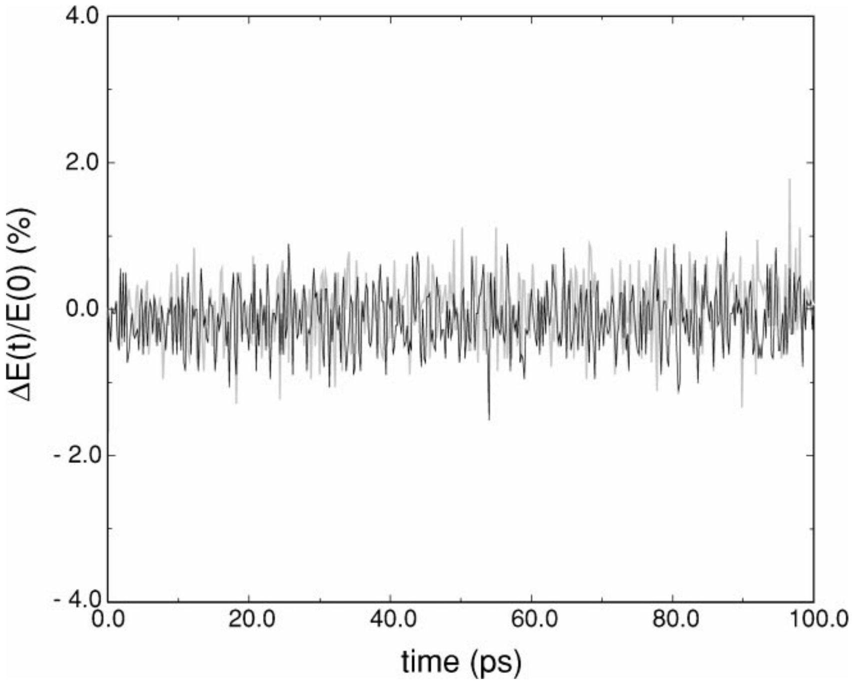

FIG. 2. Instantaneous fluctuations of the energy for gNIMM/leap-frog (black line) and SHAKE (gray line).

Next, we run a test with a rather large molecular system, e.g., the 32-fullerene C60 molecules [10], under a periodic boundary condition and a protein BPTI (Bovine Pancreatic Trypsin Inhibitor [11]) *in vacuo*. The time step of 2fs was used and only bond lengths were constrained in both C60 and BPTI molecules, but C60 actually became rigid with this constraint. Corresponding timing results (only the case with leap-frog) are listed in Tables 2, and 3, respectively.

As in the tables above, gNIMM (again almost the same as NIMM) is 4.5 and 3.2 times faster than SHAKE (with the same level of constraint errors) in the C60 and BPTI systems, respectively. It has been reported that NIMM is eight times faster than SHAKE with a water molecule system [12], so the computational efficiency of NIMM/gNIMM relative to SHAKE might be higher in smaller molecules. However, NIMM/gNIMM is proved to be faster than SHAKE even with rather large molecular systems, i.e., C60 and BPTI. Moreover, NIMM/gNIMM can be easily accelerated by using a tuned ready-made matrix solver for a specific machine.

We already showed (Fig. 1) that NIMM/gNIMM will work with a rather large time step size (with sacrifice of the constraint error which is consistent with the time integration error

TABLE 2
CPU Seconds for 32 Mols. of C60 (100 Steps)

Method	Const. CPU	Total CPU	$\langle \chi_n^2 \rangle^{1/2}$
SHAKE	28.0	524	1.7×10^{-5}
NIMM	5.45	502	3.3×10^{-7}
gNIMM	6.22	503	2.8×10^{-7}

TABLE 3
CPU Seconds for a BPTI (200 Steps)

Method	Const. CPU	Total CPU	$\langle \chi_n^2 \rangle^{1/2}$
SHAKE	5.30	155	8.8×10^{-6}
NIMM	1.40	154	5.3×10^{-6}
gNIMM	1.67	149	2.5×10^{-6}

of the atomic coordinates) with which SHAKE will not converge. To look at an another aspect of robustness of gNIMM, we tested it with a system of 32 molecules of planar rigid benzene. The system has been reported (Takeuchi [13]) as one in which SHAKE will fail. Benzene is modeled with six united atoms, and all six bond lengths and three extra pseudo-bond lengths (these equally divide the benzene hexagon into six triangles) are additionally constrained to make benzene planar rigid. Actually, SHAKE fails within a few tens of time steps even with a small time step of 0.04fs, whereas gNIMM did not fail after five thousand steps (we did not check the actual limit). The gNIMM has been applied to the simulations of various kinds of molecules (mainly liquid crystals [14]) up to more than a million time integration steps without any numerical instability.

So far, we have stated the positive aspects of the gNIMM, but there are still subjects for further discussion. For example, the geometric properties of time integrators have been shown to be important in the long term stability of simulations [15]. It has been shown that, when the iteration is carried to ideal convergence (i.e., $\chi_n(t + \Delta t) = 0$), SHAKE in combination with Verlet is symplectic and time reversible [16]. The non-ideal convergence of the gNIMM (i.e., $\chi_n(t + \Delta t) + O(\Delta t^4) = 0$) possibly breaks this symplecticness, but as yet this is not clear. That is one of the subjects for a future study.

ACKNOWLEDGMENT

We thank Dr. Munetaka Takeuchi of Fujitsu, Ltd. for helpful discussions about the behavior of SHAKE with the planar rigid benzene.

REFERENCES

1. J. P. Ryckaert, G. Ciccotti, and H. J. C. Berendsen, Numerical integration of the Cartesian equation of motion of a system with constraints, Molecular dynamics of *n*-alkanes, *J. Comput. Phys.* **23**, 327 (1977).
2. R. Edberg, D. J. Evans, and G. P. Morriss, Constrained molecular dynamics: Simulations of liquid alkanes with a new algorithm, *J. Chem. Phys.* **84**, 6933 (1986).
3. B. Hess, H. Bekker, H. Berendsen, and J. Fraaije, LINCS: A linear constraint solver for molecular simulations, *J. Comput. Chem.* **18**, 1463 (1997).
4. M. Yoneya, H. Berendsen, and K. Hirasawa, A non-iterative matrix method for constraint molecular dynamics simulations, *Mol. Simul.* **13**, 395 (1994).
5. J. Slusher and P. Cummings, Non-iterative constraint dynamics using velocity-explicit Verlet methods, *Mol. Sim.* **18**, 213 (1996).
6. D. Frenkel and B. Smit, *Understanding Molecular Simulation* (Academic Press, San Diego, 1996), Chap. 4, p. 65.
7. H. Berendsen and W. van Gunsteren, Practical algorithms for dynamics simulations, in *Molecular Dynamics Simulations of Statistical Mechanics Systems* edited by G. Ciccotti and W. Hoover (North-Holland, Amsterdam, 1986), p. 43.

8. M. Tuckerman and B. Berne, Reversible multiple time scale molecular dynamics, *J. Chem. Phys.* **97**, 1990 (1992).
9. S. J. Picken, W. F. van Gunsteren, P. T. van Duijen, and W. H. de Jew, A molecular dynamics study of the nematic phase of 4-*n*-pentyl-4′-cyanumberbiphenyl, *Liq. Cryst.* **6**, 357 (1989).
10. S. Itoh, S. Ihara, and J. Kitakawa, Toroidal form of carbon C360, *Phys. Rev. B* **47**, 1703 (1993).
11. W. F. van Gunsteren and H. J. C. Berendsen, Algorithms for macromolecular dynamics and constraint dynamics, *Mol. Phys.* **34**, 1311 (1977).
12. M. Yoneya and T. Ouchi, Molecular dynamics simulations on shared-memory multiple processor computers, *Molec. Simul.* **15**, 273 (1995).
13. M. Takeuchi, personal communication (1998).
14. M. Yoneya and H. Yokoyama, Chirality induction from chiral molecules to adsorbed monolayers, *J. Chem. Phys.* **114**, 9532 (2001).
15. D. Okunbor and R. Skeel, Explicit canonical methods for Hamiltonian systems, *Math. Comput.* **59**, 439 (1992).
16. B. Leimkuhler and R. Skeel, Symplectic numerical integrators in constrained Hamiltonian systems, *J. Comput. Phys.* **112**, 117 (1994).