Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 220 (2007) 740–750
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

P-SHAKE: A quadratically convergent SHAKE in $\mathcal{O}(n^2)$

Pedro Gonnet *

Institute of Computational Science, ETH Zürich, 8092 Zürich, Switzerland

Received 4 January 2006; received in revised form 19 May 2006; accepted 22 May 2006
Available online 24 October 2006

Abstract

An algorithm for solving arbitrary linear constraints in molecular dynamics simulations of rigid and semi-rigid molecules is presented. The algorithm – P-SHAKE – is a modified version of the SHAKE [J.-P. Ryckaert, G. Ciccotti, H.J.C. Berendsen, Numerical integration of the cartesian equations of motion of a system with constraints: Molecular dynamics of *n*-alkanes, J. Comput. Phys. 23 (1977) 327–341.] algorithm with a preconditioner applied which effectively de-couples the constraint equations. It achieves quadratic convergence, as does M-SHAKE [V. Kräutler, W.F. van Gunsteren, P.H. Hünenberger, A fast SHAKE algorithm to solve distance constraint equations for small molecules in molecular dynamics simulations. J. Comput. Chem. 22 (5) (2001) 501–508.], yet at a cost of only $\mathcal{O}(n^2)$ operations per iteration, as opposed to $\mathcal{O}(n^3)$ per iteration for M-SHAKE. The algorithm is applied to simulations of rigid water, DMSO, chlorophorm and non-rigid ethane and cyclohexane and is shown to be faster than M-SHAKE by up to a factor of three for relatively small error tolerances.

Keywords: Molecular dynamics; Simulations; Constraints; SHAKE; M-SHAKE; Lagrange multipliers; Newtion iteration; Preconditioning

1. Introduction

In molecular dynamics (MD) simulations the limiting factor for the time step is usually the fast motions of the system, such as the vibration of internal bonds. These vibrations usually occur on a much smaller time scale than the internal or external movement of the molecule and in most cases can be neglected. Therefore, constraining the fast internal motions (i.e. internal bonds) is an efficient way of increasing the simulation time step.

The most common way to constrain these internal motions is the SHAKE algorithm [16], in which the atomic coordinates within a molecule are iteratively and independently adjusted until all constraints are fulfilled up to a prescribed tolerance. RATTLE [1] solves the constraint equations in a similar fashion, adjusting also the atomic velocities and therefore provides a higher accuracy per iteration than SHAKE does. In the same category, both MSHAKE [12] and WIGGLE [13] solve the constraint equations using derived constraint forces, as opposed to positions and velocities in SHAKE and RATTLE respectively. Since for all these methods the constraint equations are solved for iteratively as if they were independent (i.e. not coupled), they converge slowly for tightly-coupled, rigid molecules.

* Tel.: +41 446324552.
E-mail address: gonnetp@inf.ethz.ch.

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2006.05.032

A different approach is presented in Ciccotti and Ryckaert [5] and implemented in both Barth et al. [2] and M-SHAKE [11], where the constraints are solved for as a non-linear system of equations using Newton's Method. This approach converges quadratically as opposed to linearly for SHAKE and similar algorithms, yet each iteration is substantially more expensive, since it involves the solution of a system of linear equations, which can only be done in $\mathcal{O}(n^3)$, as opposed to the $\mathcal{O}(n)$ operations per iteration in SHAKE.

Further efforts include SETTLE [15] and LINCS [10]. SETTLE solves the constraint equations for water or any other three-body rigid molecule analytically. This approach, however, is restricted to small molecules due to its complexity. LINCS was designed for very large molecules with weakly coupled constraints. The constraints are solved for by computing the projection of the constraint forces onto the force vector of the entire system. As stated in Hess et al. [10], however, LINCS is not suited for molecules with angular constraints since a matrix inversion, which is computed iteratively in the algorithm, becomes unstable and computationally expensive.

This paper presents P-SHAKE, an iterative method to solve linear constraint equations based on SHAKE, yet with a preconditioner applied to it such that it converges quadratically, as M-SHAKE does, yet at a much lower computational cost. The method is geared towards rigid molecules, yet it is also applicable to semi-rigid geometries or molecules with rigid sub-groups. Section 2 describes the algorithm and its derivation in detail. In Section 3, the algorithm is then tested against SHAKE, M-SHAKE and the procedure described in Ciccotti and Ryckaert [5] for three rigid solvent molecules and two semi-rigid molecules.

2. Method

The n_c linear distance constraints within a molecule can be written in the form of n_c constraint equations

$$\sigma_l^{(t)} = \left\| \mathbf{x}_{l\alpha}^{(t)} - \mathbf{x}_{l\beta}^{(t)} \right\|_2^2 - d_l^2 = 0, \quad l = 1 \ldots n_c, \tag{1}$$

where $\mathbf{x}_{l\alpha}$ and $\mathbf{x}_{l\beta}$ are the positions of the two particles involved in the constraint σ_l at time t and d_l is the prescribed inter-atomic distance. In the following we will always assume the 2-norm when the notation $\|\cdot\|$ is used.

The constraints can be enforced while integrating the equations of motion, using Lagrange's method of undetermined multipliers [16]. For each time-step, this is equivalent to finding the Lagrange multipliers λ_k, $k = 1 \ldots n_c$, such that the constrained particle positions

$$\mathbf{x}_i^{(t+\Delta t)} = \tilde{\mathbf{x}}_i^{(t+\Delta t)} + \sum_{k=1}^{n_c} \lambda_k \frac{\partial \sigma_k^{(t)}}{\partial \mathbf{x}_i^{(t)}} (\Delta t)^2 m_i^{-1}, \tag{2}$$

where $\tilde{\mathbf{x}}_i^{(t+\Delta t)}$ is the unconstrained particle position at time $t + \Delta t$ and m_i the mass of the particle i, satisfy all of the n_c constraints.

For simplicity, we write the weighted constraint gradients as

$$\mathbf{v}_{k,i}^{(t)} = \frac{\partial \sigma_k^{(t)}}{\partial \mathbf{x}_i^{(t)}} (\Delta t)^2 m_i^{-1} \tag{3}$$

which, inserted into Eq. (2), yields

$$\mathbf{x}_i^{(t+\Delta t)} = \tilde{\mathbf{x}}_i^{(t+\Delta t)} + \sum_{k=1}^{n_c} \lambda_k \mathbf{v}_{k,i}^{(t)} \tag{4}$$

which, in turn, inserted into the constraint Eq. (1), yields the constraint equation at time $t + \Delta t$

$$\begin{aligned} \sigma_l^{(t+\Delta t)} &= \left\| \tilde{\mathbf{x}}_{l\alpha}^{(t+\Delta t)} + \sum_{k=1}^{n_c} \lambda_k \mathbf{v}_{k,l\alpha}^{(t)} - \tilde{\mathbf{x}}_{l\beta}^{(t+\Delta t)} - \sum_{k=1}^{n_c} \lambda_k \mathbf{v}_{k,l\beta}^{(t)} \right\|^2 - d_l^2 \\ &= \left\| \tilde{\mathbf{x}}_{l\alpha}^{(t+\Delta t)} - \tilde{\mathbf{x}}_{l\beta}^{(t+\Delta t)} + \sum_{k=1}^{n_c} \lambda_k \left(\mathbf{v}_{k,l\alpha}^{(t)} - \mathbf{v}_{k,l\beta}^{(t)} \right) \right\|^2 - d_l^2. \end{aligned} \tag{5}$$

Using the notation

$$\delta\mathbf{x}_l = \tilde{\mathbf{x}}_{l\alpha}^{(t+\Delta t)} - \tilde{\mathbf{x}}_{l\beta}^{(t+\Delta t)} \tag{6}$$

and

$$\delta\mathbf{v}_{k,l} = \mathbf{v}_{k,l\alpha}^{(t)} - \mathbf{v}_{k,l\beta}^{(t)} \tag{7}$$

we can abbreviate Eq. (5) as

$$\sigma_l^{(t+\Delta t)} = \left\| \delta\mathbf{x}_l + \sum_{k=1}^{n_c} \lambda_k \delta\mathbf{v}_{k,l} \right\|^2 - d_l^2. \tag{8}$$

The σ_l (for simplicity we shall ignore the superscript) form a system of n_c non-linear equations in the λ_k which can be solved iteratively.

A simple approach, suggested in Ryckaert et al. [16] and referred to therein as the SHAKE algorithm, is to assume that the σ_l depend only on the respective λ_l (i.e. the equations are treated as if they were not coupled) and solve each σ_l for λ_l iteratively using Newton's Method:

$$\lambda_l = -\left.\frac{\sigma_l}{\partial\sigma_l/\partial\lambda_l}\right|_{\lambda_k=0, k=1\ldots n_c} = -\frac{\|\delta\mathbf{x}_l\| - d_l^2}{2\delta\mathbf{x}_l \cdot \delta\mathbf{v}_{l,l}}. \tag{9}$$

The particle positions are updated in each iteration using

$$\tilde{\mathbf{x}}_i \leftarrow \tilde{\mathbf{x}}_i + \sum_{k=1}^{n_c} \lambda_k \mathbf{v}_{k,i}. \tag{10}$$

In Eq. (9) we set all $\lambda_k = 0$ since we update the particle positions in each step and therefore reset the λ_k to their initial value 0. This effectively eliminates all terms linear in λ_k in σ_l and all terms quadratic in λ_k in both σ_l and $\partial\sigma_l/\partial\lambda_l$. This elimination is referred to as *linearization* in Ryckaert et al. [16].

The more straight-forward way of solving the system of non-linear equations is to use Newton's Method or a Quasi-Newton Method, i.e. the Chord Method [8].

Both methods involve computing the Jacobian of the vector of constraint equations $\boldsymbol{\sigma}$

$$\mathbf{J}_\sigma = \begin{pmatrix} \frac{\partial\sigma_1}{\partial\lambda_1} & \frac{\partial\sigma_1}{\partial\lambda_2} & \cdots & \frac{\partial\sigma_1}{\partial\lambda_{n_c}} \\ \frac{\partial\sigma_2}{\partial\lambda_1} & \frac{\partial\sigma_2}{\partial\lambda_2} & \cdots & \frac{\partial\sigma_2}{\partial\lambda_{n_c}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\sigma_{n_c}}{\partial\lambda_1} & \frac{\partial\sigma_{n_c}}{\partial\lambda_2} & \cdots & \frac{\partial\sigma_{n_c}}{\partial\lambda_{n_c}} \end{pmatrix}. \tag{11}$$

The λ_k are then computed by solving the system of linear equations

$$(\mathbf{J}_\sigma|_{\lambda_k=0})\boldsymbol{\lambda} = -\boldsymbol{\sigma}|_{\lambda_k=0} \tag{12}$$

and the unconstrained positions $\tilde{\mathbf{x}}_i$ are updated as in the SHAKE algorithm using Eq. (10).

This is repeated until

$$\max_{k=1\ldots n_c} |\sigma_k| < \tau,$$

where τ is the prescribed tolerance of the constraints. For the Chord Method, the Jacobian $\mathbf{J}_\sigma$ is computed only once in the first iteration and used in all following iterations thereafter [5,2][1]. For Newton's Method – as used in M-SHAKE [11][2] and Barth et al. [2] – the Jacobian is re-computed in every iteration.

[1] In their paper, Ciccotti and Ryckaert [5] explicitly refer to the iteration of their "matrix method" in Section 4.1.1. as "Newton-Raphson", yet at the top of page 380 they state "Note that while $\mathbf{A}^{-1}$ must be evaluated once per time step [...]", which leads the author to believe that in fact the Chord Method was used.

[2] In Kräutler et al. [11] Newton's Method is used rather inadvertently, and the authors make no explicit mention thereof in their derivation.

Although both approaches solve the coupled constraint equations and both have an asymptotic cost in $\mathcal{O}(n^3)$ associated with solving the system in Eq. (12) in the first iteration, Newton's Method converges quadratically whereas the Chord Method converges only linearly. The advantage of using the Chord Method is that the inverse or $\mathbf{J}_\sigma$ or a suitable decomposition thereof can be precomputed in the first iteration, therefore significantly accelerating the solution of the linear system of equations (Eq. (12)) in the following iterations, i.e. at a cost in $\mathcal{O}(n^2)$ vs. $\mathcal{O}(n^3)$.

Despite their increased cost compared to SHAKE, both methods significantly outperform SHAKE for tightly-coupled, rigid molecules since the number of iterations needed is drastically reduced by solving the *coupled* system of equations as opposed to the assumption of independence in SHAKE.

The error in the simpler approach (SHAKE, Eq. (9)) can be computed by inserting the computed λ_k (Eq. (9)) into the original constraint equations (Eq. (8)), thus obtaining

$$\sigma_l = \left\| \delta\mathbf{x}_l - \sum_{k=1}^{n_c} \frac{\|\delta\mathbf{x}_k\|^2 - d_k^2}{2\delta\mathbf{x}_k \cdot \delta\mathbf{v}_{k,k}} \delta\mathbf{v}_{k,l} \right\|^2 - d_l^2. \tag{13}$$

By replacing $\|\delta\mathbf{x}_l\|^2 - d_l^2$ with ε_l, the error of the lth constraint equation, we get

$$\begin{aligned} \sigma_l &= \left\| \delta\mathbf{x}_l - \sum_{k=1}^{n_c} \frac{\varepsilon_k}{2\delta\mathbf{x}_k \cdot \delta\mathbf{v}_{k,k}} \delta\mathbf{v}_{k,l} \right\|^2 - d_l^2 \\ &= \|\delta\mathbf{x}_l\|^2 - 2\delta\mathbf{x}_l \sum_{k=1}^{n_c} \frac{\varepsilon_k}{2\delta\mathbf{x}_k \cdot \delta\mathbf{v}_{k,k}} \delta\mathbf{v}_{k,l} + \sum_{k=1}^{n_c} \sum_{j=1}^{n_c} \varepsilon_k \varepsilon_j (\dots)^2 - d_l^2. \end{aligned} \tag{14}$$

Considering only the terms linear in the ε_k, we obtain, after some re-arrangement

$$\sigma_l = \underbrace{\|\delta\mathbf{x}_l\|^2 - d_l^2}_{=\varepsilon_l} - \sum_{k=1}^{n_c} \varepsilon_k \frac{\delta\mathbf{v}_{k,l} \cdot \delta\mathbf{x}_l}{\delta\mathbf{x}_k \cdot \delta\mathbf{v}_{k,k}} \tag{15}$$

We then separate the case $k = l$ from the sum and obtain

$$\sigma_l = \varepsilon_l - \varepsilon_l \underbrace{\frac{\delta\mathbf{x}_l \cdot \delta\mathbf{v}_{l,l}}{\delta\mathbf{x}_l \cdot \delta\mathbf{v}_{l,l}}}_{=1} - \sum_{k \neq l}^{n_c} \varepsilon_k \frac{\delta\mathbf{x}_l \cdot \delta\mathbf{v}_{k,l}}{\delta\mathbf{x}_k \cdot \delta\mathbf{v}_{k,k}} = -\sum_{k \neq l}^{n_c} \varepsilon_k \frac{\delta\mathbf{x}_l \cdot \delta\mathbf{v}_{k,l}}{\delta\mathbf{x}_k \cdot \delta\mathbf{v}_{k,k}} \tag{16}$$

which is the part of the error of σ_l linear in the ε_k, $k \neq l$, i.e. the error induced by ignoring the coupling of the equations. To effectively de-couple each equation σ_l, this term should be made 0.

Before we try to minimize Eq. (16), we first start with some observations regarding the constraint gradients $\mathbf{v}_{k,i}$. Given a molecule and a set of λ_k which satisfy the n_c constraint equations as in Eq. (4), we can rotate the molecule (i.e. the positions at time t and $t + \Delta t$) with any given orthogonal 3×3 matrix $\mathbf{R}$

$$\begin{aligned} \mathbf{R}\mathbf{x}_i^{(t+\Delta t)} &= \mathbf{R}\tilde{\mathbf{x}}_i^{(t+\Delta t)} + \sum_{k=1}^{n_c} \lambda_k \mathbf{R}\mathbf{v}_{k,i}^{(t)} = \mathbf{R}\left[\tilde{\mathbf{x}}_i^{(t+\Delta t)} + \sum_{k=1}^{n_c} \lambda_k \mathbf{v}_{k,i}^{(t)}\right] \\ \mathbf{x}_i^{(t+\Delta t)} &= \tilde{\mathbf{x}}_i^{(t+\Delta t)} + \sum_{k=1}^{n_c} \lambda_k \mathbf{v}_{k,i}^{(t)} \end{aligned} \tag{17}$$

and the λ_k will still satisfy the constraint equations. This also holds for translations along any vector $\mathbf{s}$

$$\begin{aligned} \mathbf{s} + \mathbf{x}_i^{(t+\Delta t)} &= \mathbf{s} + \tilde{\mathbf{x}}_i^{(t+\Delta t)} + \sum_{k=1}^{n_c} \lambda_k \mathbf{v}_{k,i}^{(t)} \\ \mathbf{x}_i^{(t+\Delta t)} &= \tilde{\mathbf{x}}_i^{(t+\Delta t)} + \sum_{k=1}^{n_c} \lambda_k \mathbf{v}_{k,i}^{(t)} \end{aligned} \tag{18}$$

since the gradients $\mathbf{v}_{k,i}^{(t)}$ are translationally independent and hence are not affected by the choice of $\mathbf{s}$ (Eq. (3)).

Given the matrix $\mathbf{V}$, the columns of which consist of the constraint gradients $\mathbf{v}_{k,i}$, and λ_k such that all constraint equations are satisfied, we can write the constrained particle positions (Eq. (4)) as (superscripts removed for clarity)

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{n_\mathrm{p}} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{x}}_1 \\ \tilde{\mathbf{x}}_2 \\ \vdots \\ \tilde{\mathbf{x}}_{n_\mathrm{p}} \end{pmatrix} + \begin{pmatrix} \mathbf{v}_{1,1} & \mathbf{v}_{2,1} & \cdots & \mathbf{v}_{n_\mathrm{c},1} \\ \mathbf{v}_{1,2} & \mathbf{v}_{2,2} & \cdots & \mathbf{v}_{n_\mathrm{c},2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}_{1,n_\mathrm{p}} & \mathbf{v}_{2,n_\mathrm{p}} & \cdots & \mathbf{v}_{n_\mathrm{c},n_\mathrm{p}} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{n_\mathrm{c}} \end{pmatrix}, \tag{19}$$

where n_p is the number of particles in the molecule, or, in matrix notation

$$\mathbf{X} = \widetilde{\mathbf{X}} + \mathbf{V}\boldsymbol{\lambda} \tag{20}$$

which we can write as

$$\mathbf{V}\boldsymbol{\lambda} = \mathbf{X} - \widetilde{\mathbf{X}}. \tag{21}$$

Given any invertible $n_\mathrm{c} \times n_\mathrm{c}$ matrix $\mathbf{A}$, we can re-write Eq. (21) as

$$(\mathbf{V}\mathbf{A})(\mathbf{A}^{-1}\boldsymbol{\lambda}) = \mathbf{X} - \widetilde{\mathbf{X}} \tag{22}$$

or

$$\underbrace{(\mathbf{V}\mathbf{A})}_{=\mathbf{V}^\star}\boldsymbol{\lambda}^\star = \mathbf{X} - \widetilde{\mathbf{X}} \tag{23}$$

which, since we are solving for $\boldsymbol{\lambda}$, is equivalent to the problem in Eq. (21). Therefore, using a linear recombination of the constraint gradients, as given by the non-singular matrix $\mathbf{A}$, does not affect the constraint problem. The $\lambda_k^\star$ are different than the original λ_k, but, given the different constraint gradients ($\mathbf{V}^\star$ vs. $\mathbf{V}$), both solve the constraint equations.

We now want to find the linear recombination of the constraint gradients such that the right-hand side of the linear error (Eq. (16)) is 0. The remaining error (Eq. (14)) should then only be quadratic in the ε_k and the iteration should converge quadratically. We start by re-writing the constraint gradients $\mathbf{v}_{k,i}$ as the linear recombination

$$\mathbf{v}_{k,i}^\star = \mathbf{v}_{k,i} + \sum_{j\neq i} a_{j,i}\mathbf{v}_{k,j}, \tag{24}$$

where $a_{j,i}$ are the entries of an $n_\mathrm{c} \times n_\mathrm{c}$ matrix $\mathbf{A}$ with unit diagonal elements. Inserting the recombined constraint gradients (Eq. (24)) into the $\delta\mathbf{v}_{k,i}$ (Eq. (7)) we obtain

$$\delta\mathbf{v}_{k,i}^\star = \mathbf{v}_{k,i\alpha}^\star - \mathbf{v}_{k,i\beta}^\star = \left[\mathbf{v}_{k,i\alpha} + \sum_{j\neq i\alpha} a_{j,i\alpha}\mathbf{v}_{k,j}\right] - \left[\mathbf{v}_{k,i\beta} + \sum_{j\neq i\beta} a_{j,i\beta}\mathbf{v}_{k,j}\right] = \delta\mathbf{v}_{k,i} + \sum_{j\neq i} a_{j,i}\delta\mathbf{v}_{k,j} \tag{25}$$

which we re-insert into the right-hand side of the linear residual (Eq. (16))

$$\sigma_l = \sum_{k\neq l}^{n_\mathrm{c}} \varepsilon_k \frac{\delta\mathbf{x}_l \cdot (\delta\mathbf{v}_{k,l} + \sum_{j\neq l} a_{j,l}\delta\mathbf{v}_{k,j})}{\delta\mathbf{x}_k \cdot (\delta\mathbf{v}_{k,k} + \sum_{j\neq k} a_{j,k}\delta\mathbf{v}_{k,j})}. \tag{26}$$

We will now solve the error in σ_l induced by each constraint $k \neq l$ separately. For this, it suffices to solve the nominator in Eq. (26):

$$\delta\mathbf{x}_l \cdot \left(\delta\mathbf{v}_{k,l} + \sum_{j\neq l} a_{j,l}\delta\mathbf{v}_{k,j}\right) = 0 \tag{27}$$

for all $k \neq l$. For convenience, we can re-write Eq. (27) as a linear sum of scalars

$$\sum_{j\neq l} a_{j,l}(\delta\mathbf{v}_{k,j} \cdot \delta\mathbf{x}_l) = -(\delta\mathbf{v}_{k,l} \cdot \delta\mathbf{x}_l) \tag{28}$$

which, for all $k \neq l$, is a system of $n_c - 1$ linear equations in the $n_c - 1$ unknowns $a_{j,l}$, $j \neq l$

$$\begin{pmatrix} r_{1,1} & \cdots & r_{1,l-1} & r_{1,l+1} & \cdots & r_{1,n_c} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ r_{l-1,1} & \cdots & r_{l-1,l-1} & r_{l-1,l+1} & \cdots & r_{l-1,n_c} \\ r_{l+1,1} & \cdots & r_{l+1,l-1} & r_{l+1,l+1} & \cdots & r_{l+1,n_c} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ r_{n_c,1} & \cdots & r_{n_c,l-1} & r_{n_c,l+1} & \cdots & r_{n_c,n_c} \end{pmatrix} \begin{pmatrix} a_{1,l} \\ \vdots \\ a_{l-1,l} \\ a_{l+1,l} \\ \vdots \\ a_{n_c,l} \end{pmatrix} = - \begin{pmatrix} r_{1,l} \\ \vdots \\ r_{l-1,l} \\ r_{l+1,l} \\ \vdots \\ r_{n_c,l} \end{pmatrix}, \tag{29}$$

where $r_{i,j} = \delta\mathbf{v}_{i,j} \cdot \delta\mathbf{x}_j$. Computing the $a_{i,l}$ for any given l in such a way, however, ignores the influence of the other columns of $\mathbf{A}$, namely the $a_{i,k}$, $k \neq l$. We must therefore compute the columns of $\mathbf{A}$ iteratively as described in Algorithm 1, where τ_A is the tolerance for the entries of $\mathbf{A}$. The resulting matrix $\mathbf{A}_0$ can then be applied to the constraint gradients matrix $\mathbf{V}$ directly.

Since solving n_c systems of linear equations in $n_c - 1$ unknowns is in $\mathcal{O}(n^4)$, this is hardly a practical way of improving the performance of the SHAKE algorithm. However, we need not do this at every time-step. In the case of rigid molecules, if instead of using the unconstrained pairwise distances $\delta\mathbf{x}_k^{(t)}$ for each molecule in each time-step we use the initial constrained positions

Algorithm 1

Compute $\mathbf{A}$ iteratively

1: Initialize $\mathbf{A}_0 \leftarrow \mathbf{I}_{n_c}$
2: **repeat**
3: Compute all $r_{i,j} \leftarrow \delta\mathbf{v}_{i,j} \cdot \delta\mathbf{x}_j$
4: Initialize $\mathbf{A} \leftarrow \mathbf{I}_{n_c}$
5: **for** each constraint σ_l **do**
6: Solve Eq. (29) for entries $a_{i,l}$, $i \neq l$ of $\mathbf{A}$
7: **end for**
8: $\mathbf{V} \leftarrow \mathbf{V}\mathbf{A}$
9: $\mathbf{A}_0 \leftarrow \mathbf{A}_0\mathbf{A}$
10: **until** $\max\|||\mathbf{A}||_\infty\| \leqslant \tau_A$

$\mathbf{x}_{k\alpha}^{(t_0)}$ and $\mathbf{x}_{k\beta}^{(t_0)}$ from a previous time-step, we only need to compute $\mathbf{A}_0$ once for each distinct rigid molecule type. The matrix $\mathbf{A}_0$ works for all molecules of the same type, since we have shown in Eqs. (17) and (18) that the solution of the constraint equations is indifferent to translations and rotations. The preconditioner $\mathbf{A}_0$ can be computed and stored offline to be passed to the simulation as a parameter. It must then only be applied to the constraint gradients $\mathbf{V}$ by a matrix–matrix multiplication

$$\mathbf{V} \leftarrow \mathbf{V}\mathbf{A}_0.$$

The convergence rate of this approach can be extracted from the linear error (Eq. (16)) by replacing the unconstrained particle positions $\tilde{\mathbf{x}}_i$ with the first-order approximation

$$\tilde{\mathbf{x}}_i \approx \mathbf{x}_i + \sum_{k=1}^{n_c} \varepsilon_k \mathbf{v}_{k,i} \tag{30}$$

which is valid if we assume that the particle movement is dominated by internal degrees of freedom, which is the case if constraints were applied in the first place.

Eq. (16) then breaks down as follows:

$$\sigma_l = -\sum_{k\neq l}^{n_c} \varepsilon_k \frac{(\mathbf{x}_{k\alpha} + \sum_j \varepsilon_j \mathbf{v}_{j,k\alpha} - \mathbf{x}_{k\beta} - \sum_j \varepsilon_j \mathbf{v}_{j,k\beta}) \cdot \delta\mathbf{v}_{k,l}}{(\mathbf{x}_{k\alpha} + \sum_j \varepsilon_j \mathbf{v}_{j,k\alpha} - \mathbf{x}_{k\beta} - \sum_j \varepsilon_j \mathbf{v}_{j,k\beta}) \cdot \delta\mathbf{v}_{k,k}} = -\sum_{k\neq l}^{n_c} \varepsilon_k \frac{\delta\mathbf{x}_k \cdot \delta\mathbf{v}_{k,l} + \sum_j \varepsilon_j \delta\mathbf{v}_{j,k} \cdot \delta\mathbf{v}_{k,l}}{(\delta\mathbf{x}_k + \sum_j \varepsilon_j \delta\mathbf{v}_{j,k}) \cdot \delta\mathbf{v}_{k,k}}$$

$$= -\sum_{k\neq l}^{n_c} \varepsilon_k \frac{\delta\mathbf{x}_k \cdot \delta\mathbf{v}_{k,l}}{(\delta\mathbf{x}_k + \sum_j \varepsilon_j \delta\mathbf{v}_{j,k}) \cdot \delta\mathbf{v}_{k,k}} - \sum_{k\neq l}^{n_c} \sum_{j=1}^{n_c} \varepsilon_k \varepsilon_j \frac{\delta\mathbf{v}_{j,k} \cdot \delta\mathbf{v}_{k,l}}{(\delta\mathbf{x}_k + \sum_j \varepsilon_j \delta\mathbf{v}_{j,k}) \cdot \delta\mathbf{v}_{k,k}}. \tag{31}$$

The first sum in Eq. (31) is 0 by construction (see Eq. (27)), and the second term is quadratic in the errors ε_k. Since all terms linear in the ε_k disappear, the method has quadratic convergence.

The complete algorithm, P-SHAKE, is shown in Algorithm 2. The only difference to the traditional SHAKE are the lines 1–3 and 6, where the preconditioner $\mathbf{A}_0$ is computed once (lines 1–3) and applied to the constraint gradients for each molecule thereafter (line 6).

Algorithm 2

Preconditioned SHAKE

```
 1: if t = t_0 then
 2:    Compute A_0 for each rigid molecule type
 3: end if
 4: for all rigid molecules do
 5:    Compute the constraint gradients V
 6:    V ← VA_0
 7:    loop
 8:       for each constraint k do
 9:          ε_k ← δx_k^2 − d_k^2
10:       end for
11:       if max_k|ε_k| > τ then
12:          Exit loop
13:       end if
14:       for each constraint k do
15:          λ_k ← −ε_k / (2δx_k · δv_{k,k})
16        end for
17:       for each particle i do
18:          for each constraint k do
19:             x_i ← x_i + λ_k v_{k,i}
20:          end for
21:       end for
22:    end loop
23: end for
```

The computational cost of the main loop (Lines 7–22) is dominated by the particle position update in Line 19, which is in $\mathcal{O}(n_p n_c)$. The main cost, however, is outside the main loop, namely in Line 6, where the matrix–matrix multiplication appears to have a cost of $\mathcal{O}(n_p n_c^2)$ and since in rigid molecules $n_p \sim n_c$, we would be in $\mathcal{O}(n_c^3)$ for all practical purposes.

The matrix $\mathbf{V}$, however, is sparse: each column k contains only two non-zero entries, namely the entries for the two particles $\mathbf{x}_{k\alpha}$ and $\mathbf{x}_{k\beta}$. If each column contains only two entries, then each row contains, on average, $\frac{2}{n_p} n_c$ entries. Each row is then multiplied with a column of $\mathbf{A}_0$, which, since we have only $\frac{2}{n_p} n_c$ non-zero entries, can be done at a cost in $\mathcal{O}(n_p^{-1} n_c^2)$. This is then done for all n_p rows for a total cost in $\mathcal{O}(n_c^2)$. The total cost of the algorithm is therefore in $\mathcal{O}(n_c^2)$.

Up to here we have only considered constraints in rigid molecules. In the case of molecules with rigid subgroups, no change in the algorithm is required. Although the orientation of the constraint gradients in the

matrix $\mathbf{V}$ change relative to each other between subgroups, no error is introduced since $\delta\mathbf{v}_{i,j} = 0$ for constraints i and j in disjoint sub-groups.

For semi-rigid molecules subject to only small internal movement, the preconditioner $\mathbf{A}_0$ still works. Although convergence is no longer guaranteed to be quadratic (Eq. (16) is no longer guaranteed to be 0), the iteration can be expected to converge at a much higher rate than the traditional uncoupled treatment (Eq. (9)), since the error in Eq. (16) will be considerably smaller.

For semi-rigid molecules subject to more significant conformational changes, a preconditioner $\mathbf{A}_0$ can be maintained for each individual molecule and be updated periodically to account for the changing molecular configuration. If the preconditioner $\mathbf{A}_0$ is still close to optimal, we can update it much more efficiently than in Algorithm 1 by solving Eq. (28) for each $a_{i,j}$, $i \neq j$ separately using

$$a_{i,j} = -\frac{\delta\mathbf{v}_{i,j} \cdot \delta\mathbf{x}_j}{\delta\mathbf{v}_{i,i} \cdot \delta\mathbf{x}_j}. \tag{32}$$

This update is described more formally in Algorithm 3. The update itself is in $\mathcal{O}(n_c^2)$, yet it needs only to be applied periodically and should reduce the number of iterations required for convergence.

Algorithm 3

Update $\mathbf{A}_0$

```
1:  Initialize A ← I_{n_c}
2:  for i from 1 to n_c do
3:     for j from 1 to n_c do
4:        if i ≠ j then
5:           A_{i,j} = -(δv_{i,j}·δx_j)/(δv_{i,i}·δx_j)
6:        end if
7:     end for
8:  end for
9:  V ← VA
10: A_0 ← A_0A
```

3. Results

To assess the performance of the four methods described – SHAKE, M-SHAKE, Ciccotti's Matrix Method (hereafter referred to as CMM) and P-SHAKE – molecular dynamics simulations were run with three different rigid solvents: water (SPC\E, Berendsen et al. [3], $n_c = 3$), dimethyl sulfoxide (DMSO, Liu and Müller-Plathe [14], $n_c = 6$) and chloroform ($CHCl_3$, Tironi and van Gunsteren [17], $n_c = 9$).

Additionally, simulations of semi-rigid ethane and cyclohexane were run. The ethane was modelled as an all-atom molecule using the AMBER force-field [6], yet with the two methyl end groups kept rigid using 6 constraints each ($n_c = 12$). The cyclohexane molecules were modelled as described in Faller et al. [7] with harmonic angular potentials, yet with all bond-lengths constrained ($n_c = 18$).

The three rigid solvents were simulated in cubic, periodic cells of edge length 3.166 nm, 5.009 nm and 5.2075 nm each, containing 1000 molecules at the densities of 0.997, 1.095 and 1.489 g cm^{-3} at 298 K for the water, DMSO and $CHCl_3$ respectively. The water simulations were run with a time-step of 2 fs and the DMSO and chloroform with a time-step of 5 fs. The ethane simulations were run in a cubic, periodic cell of edge length 4.295 nm containing 1000 molecules at 100 K with a time-step of 2 fs. The cyclohexane simulations were run in a cubic, periodic cell of edge length 5.6454 nm containing 1000 molecules at 298 K with a time-step of 2 fs. Electrostatic and Lennard-Jones interactions were truncated at 1.0 nm in all simulations.

All simulations were equilibrated for 10 000 steps while coupled to a heat bath using a coupling constant of 0.1 ps. All data (no. of iterations, timings, etc.) was averaged from the 1000 simulation time-steps following equilibration.

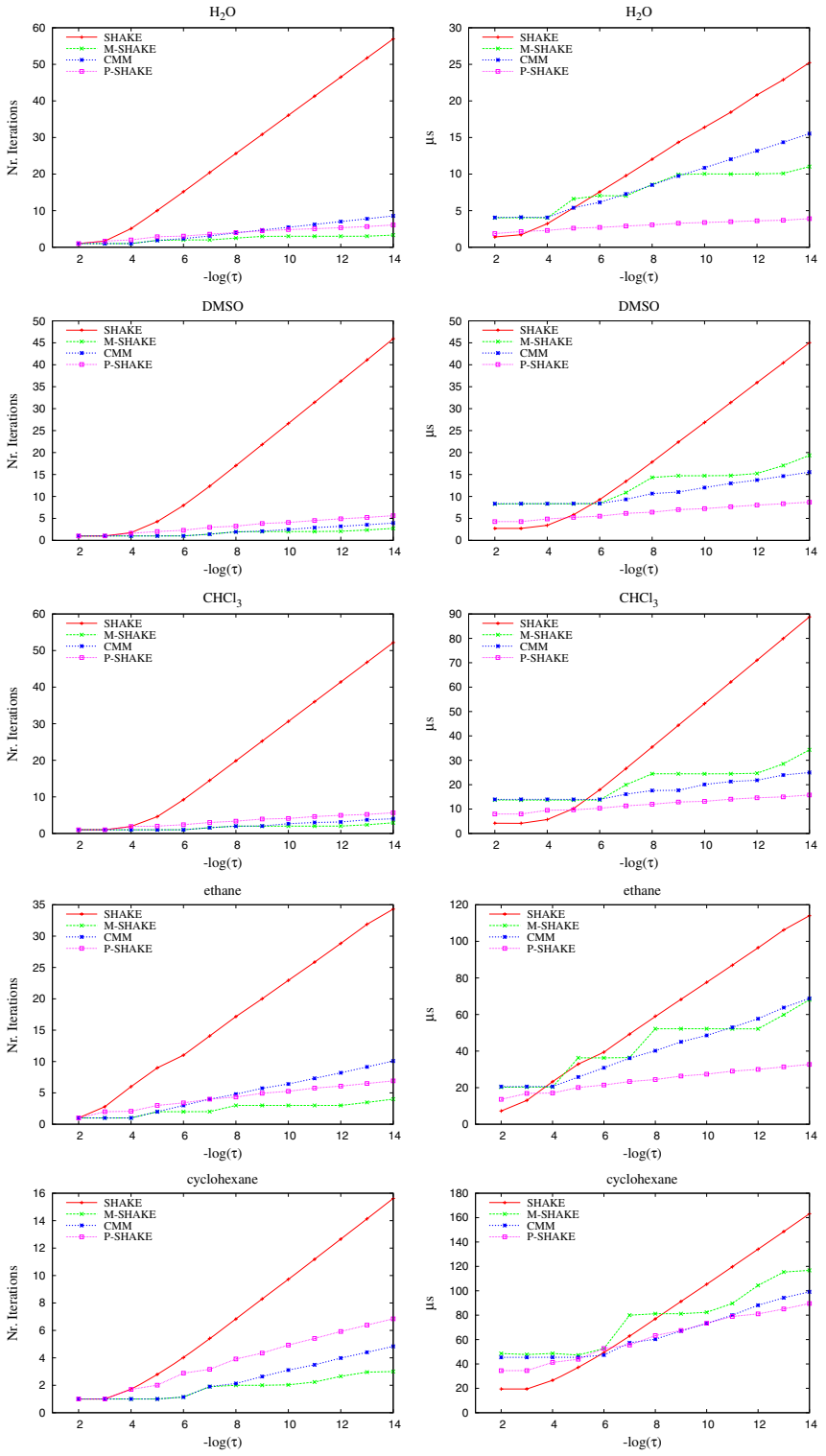

Fig. 1. Average number of iterations and average computation time for SHAKE, M-SHAKE, CMM and P-SHAKE for the solvents tested.

Table 1
Details for each algorithm for each molecule for $\tau = 10^{-10}$

Molecule	Method	No. of iterations	μs per mol	% total time
H_2O	SHAKE	36.08	16.40	10.28
	M-SHAKE	3.00	10.03	6.55
	CMM	5.47	10.86	7.05
	P-SHAKE	4.83	3.38	2.31
DMSO	SHAKE	26.62	26.86	27.59
	M-SHAKE	2.00	14.72	17.27
	CMM	2.49	12.02	14.57
	P-SHAKE	4.07	7.23	9.30
$CHCl_3$	SHAKE	30.63	53.23	29.37
	M-SHAKE	2.00	24.44	16.03
	CMM	2.67	20.09	13.56
	P-SHAKE	4.10	13.17	9.33
Ethane	SHAKE	22.95	77.64	41.82
	M-SHAKE	3.00	52.21	32.59
	CMM	6.42	48.55	31.01
	P-SHAKE	5.26	27.39	20.23
Cyclohexane	SHAKE	9.72	105.35	8.39
	M-SHAKE	2.03	82.38	6.68
	CMM	3.11	73.16	5.98
	P-SHAKE	4.93	73.36	6.00

All 4 algorithms were implemented in the FASTTUBE simulation software [18] and compiled with the Intel Fortran Compiler 9.0 using the Intel Math Kernel Library 7.2 BLAS and LAPACK routines. All simulations were run on an IBM T40p ThinkPad computer with an 1.6 GHz Intel Pentium M processor.

The SHAKE algorithm was implemented as first described in Ryckaert et al. [16]. Both M-SHAKE and CMM where implemented using an LU-factorization [9] of the Jacobian (Eq. (11)). For CMM the LU-factorization was computed only once and subsequently used to solve Eq. (12) for the different right-hand sides, as opposed to computing the inverse as suggested in Ciccotti and Ryckaert [5]. The factorization was computed using the LAPACK subroutine DGETRF and the system was then solved using the LAPACK subroutine DGETRS. P-SHAKE was implemented as described in Section 2. The sparse matrix–matrix multiplication was implemented by hand. The matrices $\mathbf{A}_0$ were pre-computed for each molecule type using MATLAB (MATLAB 7, The MathWorks Inc., Natick, MA, 2000). For the cyclohexane simulations, the preconditioner was computed for the relaxed configuration of the cyclohexane molecules. No update of the preconditioner was required.

The number of iterations and microseconds required on average to converge to a specified absolute tolerance τ were recorded for $\tau = 10^{-2}, 10^{-3}, \ldots, 10^{-14}$ and are shown in Fig. 1. Results for $\tau = 10^{-10}$ for each method and solvent are summarized in Table 1.

Since an LU-factorization is probably overkill for small systems (*i.e.* water with $n_c = 3$), a simulation was run using M-SHAKE with an explicit solution for the linear system of equations (Eq. (12)), computed with Maple [4] using the packages `linalg` and `codegen` for automatic code generation. To allow for a fair comparison, P-SHAKE was re-compiled for a fixed $n_c = 3$ and the sparse matrix–matrix multiplication $\mathbf{VA}_0$ was coded explicitly. For water molecules at $\tau = 10^{-14}$ M-SHAKE and P-SHAKE require on average 3.06 and 2.89 μs, respectively.

4. Conclusions

We have presented P-SHAKE, a linear constraint solver for rigid and semi-rigid molecules in molecular dynamics simulations. The algorithm is an extension of the classical SHAKE algorithm and differs only insofar, that the constraint gradients are linearly recombined such that the individual constraint equations become

linearly independent (i.e. the constraint equations are linearly de-coupled). The thus modified P-SHAKE converges quadratically, requiring far less iterations than SHAKE to achieve convergence for low tolerances.

Due to the low computational cost of the initial preconditioning and of each iteration (both in $\mathcal{O}(n^2)$) and its quadratic convergence, it significantly out-preforms both M-SHAKE and CMM for tightly coupled, rigid molecules and molecules with tightly coupled, rigid subgroups. This is due to the cost in $\mathcal{O}(n^3)$ per iteration required by M-SHAKE and the linear convergence of CMM. This asymptotic advantage is backed by the timings in Fig. 1, where the lower initial cost, the low cost per iteration and the quadratic convergence can be observed.

The algorithm preforms best for small rigid molecules or molecules with rigid sub-groups. For the semi-rigid molecules, P-SHAKE out-preforms M-SHAKE and CMM for ethane, whereas for cyclohexane the results are more or less equal.

The method can also be applied to semi-rigid molecules, as the simulations of ethane and cyclohexane demonstrate. In the case of ethane, since the rigid sub-groups are independent (i.e. they do not share any constraint), the preconditioner works as with any rigid molecule. In the case of cyclohexane, the internal movements of the molecule do not greatly affect the effectiveness of the preconditioner. Whenever large deformations occur, the preconditioner can be updated rather efficiently.

Acknowledgements

The author would like to thank Urs Zimmerli for the valuable discussions regarding constraint resolution and life in general and Oscar Chinellato for all things regarding linear algebra and notation as well as the group of Prof. P. Koumoutsakos for their support. The author also thanks the anonymous reviewers for their helpful comments.

References

[1] H. Andersen, RATTLE – a velocity version of the shake algorithm for molecular-dynamics calculations, J. Comput. Phys. 52 (1) (1983) 24–34.
[2] E. Barth, K. Kuczera, B. Leimkuhler, R.D. Skeel, Algorithms for constrained molecular dynamics, J. Comput. Chem. 16 (10) (1995) 1192–1209.
[3] H.J.C. Berendsen, J.R. Grigera, T.P. Straatsma, The missing term in effective pair potentials, J. Phys. Chem. 91 (1987) 6269–6271.
[4] B.W. Char, K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan, S.M. Watt, Maple V Language Reference Manual, Springer-Verlag, 1991.
[5] G. Ciccotti, J. Ryckaert, Molecular dynamics simulation of rigid molecules, Computer Physics Reports 4 (1986) 345–392.
[6] W.D. Cornell, P. Cieplak, C.I. Bayly, I.R. Gould, K.M. Merz Jr., D.M. Ferguson, D.C. Spellmeyer, T. Fox, J.W. Caldwell, P.A. Kollman, A second generation force field for the simulation of proteins, nucleic acids, and organic molecules, J. Am. Chem. Soc. 117 (1995) 5179–5197.
[7] R. Faller, H. Schmitz, O. Biermann, F. Müller-Plathe, Automatic parameterization of force fields for liquids by simplex optimization, J. Comput. Chem. 20 (10) (1999) 1009–1017.
[8] P.E. Gill, W. Murray, M.H. Wright, Numerical linear algebra and optimization, Addison-Wesley, Redwood City, California, 1991.
[9] G. Golub, C.F.V. Loan, Matrix Computations, The John Hopkins University Press, Baltimore, 1983.
[10] B. Hess, H. Bekker, H.J.C. Berendsen, J.G.E.M. Fraaije, LINCS: A linear constraint solver for molecular simulations, J. Comput. Chem. 18 (12) (1997) 1463–1472.
[11] V. Kräutler, W.F. van Gunsteren, P.H. Hünenberger, A fast SHAKE algorithm to solve distance constraint equations for small molecules in molecular dynamics simulations, J. Comput. Chem. 22 (5) (2001) 501–508.
[12] S. Lambrakos, J. Boris, E. Oran, I. Chandrasekhar, M. Nagumo, A modified SHAKE algorithm for maintaining rigid bonds in moleculardynamics simulations of large molecules, J. Comput. Phys. 85 (2) (1989) 473–486.
[13] S.-H. Lee, K. Palmo, S. Krimm, A new constrained molecular dynamics algorithm in cartesian coordinates, J. Comput. Phys. 210 (1) (2005) 171–182.
[14] H. Liu, F. Müller-Plathe, W.F. van Gunsteren, A force-field for liquid dimethyl-sulfoxide and physical-properties of liquid dimethylsulfoxide calculated using molecular-dynamics simulation, J. Am. Chem. Soc. 117 (15) (1995) 4363–4366.
[15] S. Miyamoto, P.A. Kollman, SETTLE: An analytical version of the SHAKE and RATTLE algorithm for rigid water models, J. Comput. Chem. 13 (8) (1992) 952–962.
[16] J.-P. Ryckaert, G. Ciccotti, H.J.C. Berendsen, Numerical integration of the cartesian equations of motion of a system with constraints: Molecular dynamics of n-alkanes, J. Comput. Phys. 23 (1977) 327–341.
[17] I. Tironi, W.F. van Gunsteren, A molecular dynamics simulation study of chloroform, Mol. Phys. 83 (1994) 381–403.
[18] T. Werder, J.H. Walther, R.L. Jaffe, T. Halicioglu, P. Koumoutsakos, On the water-graphite interaction for use in MD simulations of graphite and carbon nanotubes, J. Phys. Chem. B 107 (2003) 1345–1352.