JCTC Journal of Chemical Theory and Computation

P-LINCS: A Parallel Linear Constraint Solver for Molecular Simulation

Berk Hess*

Max-Planck Institute for Polymer Research, Ackermannweg 10, D-55128 Mainz, Germany

Received August 9, 2007

Abstract: By removing the fastest degrees of freedom, constraints allow for an increase of the time step in molecular simulations. In the last decade parallel simulations have become commonplace. However, up till now efficient parallel constraint algorithms have not been used with domain decomposition. In this paper the parallel linear constraint solver (P-LINCS) is presented, which allows the constraining of all bonds in macromolecules. Additionally the energy conservation properties of (P-)LINCS are assessed in view of improvements in the accuracy of uncoupled angle constraints and integration in single precision.

I. Introduction

In classical molecular simulation methods, such as molecular dynamics (MD), the time step is limited by the fastest motions, which are bond oscillations. These oscillations have a relatively high frequency and low amplitude. By replacing at least the bond vibrations involving hydrogen atoms by holonomic constraints the time step in molecular simulations can be increased by roughly a factor of 4. Constraints are often considered a more faithful representation of the physical behavior of bond vibrations which are almost exclusively in their vibrational ground state.

Constraints can be added to the Hamiltonian using Lagrange multipliers. When time is discretized the linear equations for the Lagrange multipliers become nonlinear. In the past decades several algorithms have appeared to solve these equations. The first algorithm was SHAKE,[1] an iterative method for use with a leapfrog integrator. The equivalent for the velocity-Verlet integrator is called RATTLE.[2] Because of their iterative nature these algorithms do not lend them selves well for parallelization. In the simplest approach[3] communication is required at each iteration. For a molecule with all bonds constrained and a time step of 2 fs this leads to around 10 iterations and communication steps per MD time step, which is a much higher communication load than that of the other parts of the MD algorithm. In principle one could use the strategy for parallelization that will be described in this paper for the LINCS (linear constraint solver) algorithm[4] also for iterative methods. But the problem with that is that the number of iterations is not known a priori, and, therefore, the data that need to be communicated are not known when the domain decomposition is (re)made.

To avoid the issue of nonlinearity, the problem can be reduced to a linear matrix equation if the second derivatives of the constraint equations are set to zero. However, in a finite discretization scheme corrections are necessary to achieve accuracy and stability. Several methods have been proposed based on this linearization that have been termed promising for parallel simulations.[5−8] But none of these methods has been widely used, because the inherently unstable algorithms require some (periodically applied) corrections.

Since there are currently no practical parallel constraints algorithms, molecular simulation packages do not allow for constraints to cross node boundaries. For codes using domain decomposition this means that in practice only bonds involving hydrogens can be constrained, as such bonds only couple locally to one heavy atom. For particle or force decomposition codes, such as GROMACS 3.3,[9] it means that molecules with all bonds constrained cannot be split over processor boundaries. For a protein in water one can then not parallelize efficiently over more than a few processors.

Without constraints the fastest motions in molecular simulations are bond vibrations involving hydrogens, with a period of about 10 fs. When these bonds are constrained, the time step can be doubled, since the next fastest motion, bond vibrations involving only heavy atoms, have the fastest

* Corresponding author e-mail: hessb@mpip-mainz.mpg.de.

mode with a period of about 20 fs. How big the time step can actually be is difficult to determine. A nice, but deceptive property of Verlet type integrators, which are used by all major simulation packages, is that vibrations in harmonic potentials are integrated with exact energy conservation. Thus energy conservation is independent of the time step. However, the effective temperature at which the ensemble for harmonic modes is generated increases with the time step. For 20 and 10 time steps per oscillation the effective temperature is 3% and 14% too high, respectively. When such harmonic modes are not considered important, their temperature increase can be ignored, but in that case constraining them is better, since this avoids instabilities.

Additionally replacing bonds involving heavy atoms by constraints does not allow for an increase in time step, since also angle vibrations involving hydrogens have the shortest period of 20 fs. In the GROMACS package these vibrations can also be removed by replacing most hydrogen atoms by virtual interaction sites and constraining C−O−H angles.[10] Since then the fastest remaining modes have a period of about 45 fs, and this allows for an increase in a time step of slightly more than a factor of 2. Recently it was decided to implement domain decomposition into the GROMACS package, and, therefore, a parallel constraint algorithm is required, otherwise a factor of 2 in performance would be lost.

A decade ago the LINCS (linear constraint solver) algorithm was introduced.[4] This algorithm builds on the same linear approximation stated above but improves upon earlier algorithms in three ways. First a term is added to the equations which is analytically zero but for the discretized version leads to a completely stable algorithm. Second, iterations are applied to capture the nonlinear effects (i.e., bond rotations); under most circumstances a single iteration suffices. Third, the matrix is inverted efficiently using a series expansion, which leads to a bounded range of couplings between constraints, equal to the expansion order. The algorithm can be used for any type of integrator. In the original paper it is presented for the leapfrog integrator. But it is ideally suited for projecting out components of velocities or forces, a linear problem for which no iterations are required. The LINCS algorithm is the standard constraint algorithm in the GROMACS package, next to the SETTLE algorithm[11] which is only used for water molecules. Recently a "matrix-version'' of the SHAKE algorithm has been developed,[12] which is parallelized with particle decomposition. This algorithm solves the same matrix equations as those of LINCS but uses a conjugate gradient solver. Therefore it does not have the exactly bounded coupling range that makes LINCS suitable for use in domain decomposition.

The original paper hinted that LINCS is easy to parallelize. That is what will be shown in the following, although in a slightly different way than originally stated. Additionally the energy conservation properties of LINCS are shown and compared with SHAKE in the light of an improvement for uncoupled angle constraints and recent improvements in integration accuracy.

II. The LINCS Algorithm

A concise description of the LINCS algorithm will now be presented, and the full derivation can be found in the LINCS paper.[4]

Consider a system of N particles, with positions given by a $3N$ vector $\mathbf{r}(t)$. The equations of motion are given by Newton's law

$$\frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}t^2} = \mathbf{M}^{-1}\mathbf{f} \tag{1}$$

where $\mathbf{f}$ is the $3N$ force vector and $\mathbf{M}$ is a $3N \times 3N$ diagonal matrix, containing the masses of the particles. In general a system is constrained by K time-independent constraint equations

$$g_i(\mathbf{r}) = 0 \quad i = 1,..., K \tag{2}$$

The constrained system can still be described by $3N$ second-order differential equations in Cartesian coordinates.[13,14] The constraints will be applied according to the principal of least action.[15] In this approach the constraints are added as a zero term to the potential $\mathbf{V}(\mathbf{r})$, multiplied by Lagrange multipliers $\lambda_i(t)$

$$-\mathbf{M}\frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}t^2} = \frac{\partial}{\partial \mathbf{r}}(\mathbf{V} - \lambda\cdot\mathbf{g}) \tag{3}$$

A new notation is introduced for the gradient matrix of the constraint equations which appears on the right-hand side of the equation

$$B_{hi} = \frac{\partial g_h}{\partial r_i} \tag{4}$$

Note that $\mathbf{B}$ is a $K \times 3N$ matrix, and it contains the directions of the constraints. Equation 3 can now be simplified to give

$$-\mathbf{M}\frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}t^2} + \mathbf{B}^T\lambda + \mathbf{f} = 0 \tag{5}$$

The equations can be solved for λ to give the constrained equations of motion

$$\frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}t^2} = (\mathbf{I} - \mathbf{TB})\mathbf{M}^{-1}\mathbf{f} - \mathbf{T}\frac{\mathrm{d}\mathbf{B}}{\mathrm{d}t}\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} \tag{6}$$

where $\mathbf{T} = \mathbf{M}^{-1}\mathbf{B}^T(\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T)^{-1}$. The projection matrix $\mathbf{I} - \mathbf{TB}$ projects out the components of a vector in the directions of the constraints, $\mathbf{M}^{-1}\mathbf{f}$ is the vector of unconstrained second derivatives, and $\mathbf{T}$ is a $3N \times K$ matrix that transforms motions in the constrained coordinates into motions in Cartesian coordinates, without changing the equations of motion of the unconstrained coordinates. The last term in (6) represents centripetal forces caused by rotating bonds. If the constraints are satisfied in the starting configuration, the linear differential eq 6 will conserve the constraints. The nonlinearity arises when eq 6 is discretized.

For holonomic constraints the constraint equations can be chosen as

$$g_i(\mathbf{r}_n) = |\mathbf{r}_{n,i_1} - \mathbf{r}_{n,i_2}| - d_i = 0 \quad i = 1,..., K \tag{7}$$

where d_i is the reference distance between atoms i_1 and i_2. The first step in the LINCS algorithm for a leapfrog integrator gives the linear, "zeroth iteration'' correction

$$\mathbf{r}_{n+1}^{0} = (\mathbf{I} - \mathbf{T}_n\mathbf{B}_n)\mathbf{r}_{n+1}^{*} + \mathbf{T}_n\mathbf{d} \qquad (8)$$

where $\mathbf{r}_{n+1}^{*}$ is the unconstrained updated configuration:

$$\mathbf{r}_{n+1}^{*} = \mathbf{r}_n + \Delta t\mathbf{v}_{n-1/2} + (\Delta t)^2\mathbf{M}^{-1}\mathbf{f}_n \qquad (9)$$

This algorithm is already stable but does not capture the nonlinear effect of the lengthening of constraints due to rotation. To correct for this, iterations are applied:

$$\mathbf{r}_{n+1}^{z+1} = (\mathbf{I} - \mathbf{T}_n\mathbf{B}_n)\mathbf{r}_{n+1}^{z} + \mathbf{T}_n\mathbf{p}^{z} \qquad (10)$$

The projected lengths $\mathbf{p}^z$ are chosen by assuming that the observed displacements in iteration z perpendicular to the constraint direction in step n are (nearly) correct

$$p_i^z = \sqrt{2d_i^{\,2} - (l_i^z)^2} \qquad (11)$$

where l_i^z is the slightly too long a distance in configuration $\mathbf{r}_{n+1}^{z}$. For most systems a single iteration provides sufficient accuracy. Quantitative results will be shown later. For projecting out constraint components of other quantities, such as velocities or forces, only the linear projection is required:

$$\mathbf{v}_n = (\mathbf{I} - \mathbf{T}_n\mathbf{B}_n)\mathbf{v}_n^{*} \qquad (12)$$

The main computational issue is the matrix inversion required to obtain $\mathbf{T}$. This is simplified by left and right multiplying the constraint coupling matrix $\mathbf{B}_n\mathbf{M}^{-1}\mathbf{B}_n^{T}$ with a diagonal $K \times K$ matrix $\mathbf{S}$ containing the inverse square root of the diagonal of the coupling matrix:

$$\mathrm{S} = \mathrm{Diag}\left(\sqrt{\frac{1}{m_{1_1}} + \frac{1}{m_{1_2}}}, ..., \sqrt{\frac{1}{m_{K_1}} + \frac{1}{m_{K_2}}}\right) \qquad (13)$$

The conversion goes as follows:

$$(\mathbf{B}_n\mathbf{M}^{-1}\mathbf{B}_n^{T})^{-1} = \mathbf{S}\mathbf{S}^{-1}(\mathbf{B}_n\mathbf{M}^{-1}\mathbf{B}_n^{T})^{-1}\mathbf{S}^{-1}\mathbf{S} = \mathbf{S}(\mathbf{S}\mathbf{B}_n\mathbf{M}^{-1}\mathbf{B}_n^{T}\mathbf{S})^{-1}\mathbf{S} \equiv \mathbf{S}(\mathbf{I} - \mathbf{A}_n)^{-1}\mathbf{S} \qquad (14)$$

The matrix $\mathbf{A}_n$ is symmetric and sparse and has zeros on the diagonal. Thus a series expansion can be used to calculate the inverse:

$$(\mathbf{I} - \mathbf{A}_n)^{-1} = \mathbf{I} + \mathbf{A}_n + \mathbf{A}_n^{\,2} + \mathbf{A}_n^{\,3} + ... \qquad (15)$$

The inversion only converges when the absolute values of all the eigenvalues of $\mathbf{A}_n$ are smaller than one. For calculating the expansion only matrix-vector multiplications are required. Since A is very sparse, only the nonzero elements should be stored, and the multiplications are computationally cheap.

Nearly all bonds in molecular simulations are sp^3 or sp^2 hybridized, which leads to a cosine of the angle between bonds of $-1/3$ and $-1/2$, respectively. The off-diagonal elements of A are given by this cosine times a mass factor which is between 0.5 and 1. This results in a maximum eigenvalue of A of 0.6$-$0.7, which means that the inversion always converges. The effective eigenvalue for typical bond distortions in MD simulations is around 0.4. Thus for each term in the expansion, the projection becomes more accurate by a factor of 0.4. The accuracy of each iteration can be less though, since it is limited by the guess for the projection of that iteration. The practical range for the expansion order is between 4 and 8. For large angle-constrained molecules eigenvalues larger than one occur, and therefore a different inversion method is required.

Nonconnected angle constraints appear in methyl, NH_3^+, and COH groups when angle vibrations involving hydrogens are removed.[10] Such individual angle constraints produce large eigenvalues with a localized eigenvector in matrix A. Especially the rigid COH group is problematic with a largest eigenvalue of 0.7, which is significantly larger than the effective eigenvalues of 0.4 for bond constraints. This imbalance means that the convergence of the expansion (15) can be limited by a few angle constraints. To avoid computational overhead due to angle constraints, the expansion can be extended for some couplings only

$$(\mathbf{I} - \mathbf{A}_n)^{-1} \approx \mathbf{I} + \mathbf{A}_n + ... + \mathbf{A}_n^{N_i} + (\mathbf{A}_n^{*} + ... + \mathbf{A}_n^{*N_i})\mathbf{A}_n^{N_i} \qquad (16)$$

where N_i is the normal order of the expansion, $\mathbf{A}^*$ only contains the elements of $\mathbf{A}$ that couple constraints within rigid triangles, and all other elements are zero. In this manner the accuracy of angle constraints comes close to that of the other constraints, while the series of matrix vector multiplications required for determining the expansion only needs to be extended for a few constraint couplings.

The last point is how the constraints need to be applied to derivatives of the coordinates, namely the velocity and the forces. In GROMACS, as in most other simulations packages, the velocities were determined from the difference between the new and the old constrained positions. This can lead to inaccurate integration in single precision, since the increment of the coordinates can be very small. For the leapfrog integrator, eq 12 applied to the half step velocity would provide a more accurate solution. But recently it has been shown that one can also directly use the Lagrange multipliers.[16] For the LINCS algorithm these are already calculated, and the velocity correction then reads

$$\mathbf{v}_{n+1/2} = \mathbf{v}_{n+1/2}^{*} + \frac{1}{\Delta t}M^{-1}\mathbf{B}_n^{T}\lambda \qquad (17)$$

Similarly the contribution of the constraints to the virial can be determined from the constraint forces which in the derivation above are given by the Lagrange multipliers. For the virial the inner product of the distance with the constraint forces $\mathbf{f}_c$ is required; this is simply $\mathbf{d}\cdot\lambda$. The full tensor can be obtained by using the outer products of the constraint directions with themselves.

III. The P-LINCS Algorithm

The inversion through a series expansion provides a nice physical picture. The coupling matrix A gives the direct coupling between bonds. The matrix A^2 gives the coupling between bonds separated by one bond and also the back-coupling of bonds to themselves. The matrix A^3 gives the

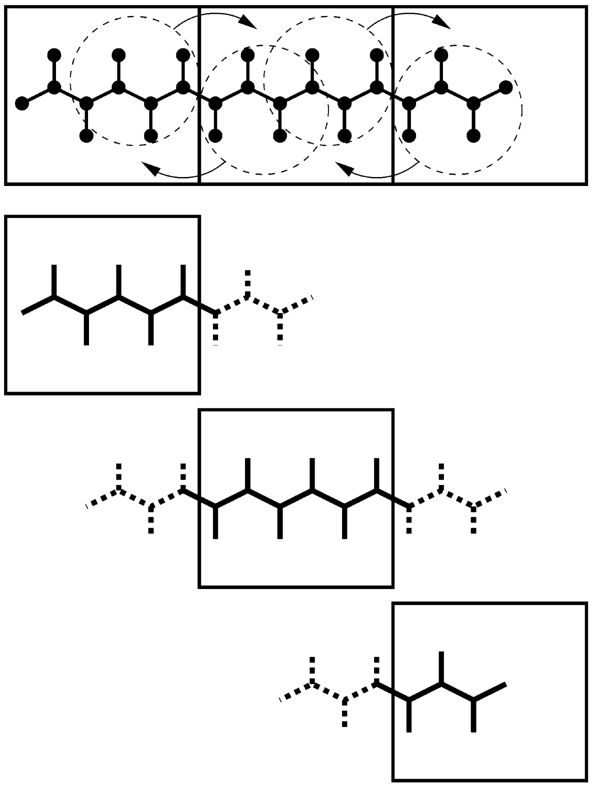

Figure 1. Example of the parallel setup of P-LINCS with one molecule split over three domain decomposition cells, using a matrix expansion order of 3. The top part shows which atom coordinates need to be communicated to which cells. The bottom parts show the local constraints (solid) and the nonlocal constraints (dashed) for each of the three cells.

coupling between bonds separated by two bonds, etc. This means that with this inversion method bonds do not influence each other when they are separated by more bonds than the maximum order in the expansion. This fact can be used to parallelize the algorithm.

When domain decomposition is used, each domain can apply the LINCS algorithm not only to the local constraints but also to some constraints involving only atoms of the neighboring domains. In the original paper it was suggested to communicate the atoms for the extra constraints for all iterations at once. But it turns out to be more practical, and often also more efficient, to communicate before each iteration. Then in addition to the atoms of constraints that cross the borders between domains, atoms that are separated by maximally the number of bonds equal to the order of the expansion need to be communicated (see Figure 1). When these coordinates are present, a LINCS iteration can be applied to local plus extra constraints. After such an iteration, the local constraints and the ones crossing the border will be in an identical state to a nonparallel version of the algorithm. The extra, nonlocal constraints will differ, since they have not felt the influence of some coupled constraints, but this does not matter, since in the end state they do not influence the local atoms. Note that for determining the constraint contribution to the virial, one should take care that constraints that cross cell boundaries are not double counted. The additional terms in the expansion (16) for angle constraints can interfere with the exact correspondence of P-LINCS and LINCS results but only when two or more triangles of constraints have a constraint in common. However, this is not an issue, since for such cases the matrix expansion converges too slowly or not at all. The procedure for removing angle vibrations of hydrogens[10] does not introduce coupled triangles of constraints.

Before each iteration the updated nonlocal coordinates need to be communicated. This leads to a total number of communication steps of one plus the number of iterations. No communication is required after the last iteration. Since the number of iterations is usually one, the number of communication steps is two. By doing a communication step for each iteration an expansion order of up to 6 can be used, since the number of bonds in an all-trans chain that fits in a domain decomposition cell size of 1 nm (a typical minimum value) is 7. When required, extra communication steps can be added for atoms in more distant cells. The same procedure can be used for constraining velocities or forces; there only one communication step is required.

The final result of the P-LINCS algorithm is identical to that of the LINCS algorithm, save for numerical rounding differences. In the implementation of P-LINCS in the upcoming 4.0 version of the GROMACS package, the coupling matrix is always stored in the same order, which leads to binary identical results to those of LINCS.

When the required extra pieces of molecule(s) are not longer than the smallest dimension of a domain decomposition cell, one cell needs to communicate with at most 26 other cells with full 3D domain decomposition. The communication can be performed in 3 steps of pairs of communication calls. In the P-LINCS implementation in GROMACS the constraint communication setup is redetermined every time the decomposition changes, which is usually every 5−10 integration steps. Every cell has a list of all the constraints in the whole system. Each cell can then determine of which nonlocal atoms it needs the coordinates for connected constraints that cross the cell boundaries. The list of required atoms is first sent one cell forward and backward in the *x* direction. Atoms in the received list that are locally present are marked, and the rest, in addition to the locally required atoms, are sent forward and backward one cell in the *y* direction. The same procedure is repeated for the *z* direction. In the opposite order and direction the cells send and accumulate the found atom indices. Each cell can then determine if all required atoms have been found. Before each LINCS iteration the last part of the procedure is repeated but then with the coordinates instead of the atom indices. This results in a maximum of 6 communication steps per iteration. For a machine with two-way network connections the forward and backward calls can be overlapped. The required bandwidth will be quite low compared to the latency. The passing of coordinates through other cells leads to little overhead, since these are usually small in number, and often these coordinates are also required by the cells they pass through.

IV. Benchmarks

It is difficult to assess the accuracy of MD simulations of biomolecular systems. Ideally one would want to check how accurate the generated ensemble is, but for proteins and even for peptides of more than a few amino acids it is computationally infeasible to sample the full phase space. A constraint algorithm should, of course, accurately set the constraint lengths. But how accurate is accurate enough, a relative error of 10^{-4}, or maybe 10^{-6}? An easy quantitative check is the accuracy of energy conservation. But also for this quantity is it difficult to judge what value is required. For microcanonical simulations one can easily determine how much drift in the total energy one can allow over the total simulation time. But most simulations are performed in the canonical or constant-NPT ensembles. Here the thermostat will effectively compensate for energy changes due to (small) integration errors. How much energy drift can be allowed is therefore unclear. Also one should keep in mind that Verlet type integrators perfectly conserve energy for harmonic potentials for any and therefore also unrealistically large, integration step size. This means that for the accuracy of especially simulations without bond constraints one cannot rely on energy conservation as a measure of integration accuracy.

It is clear that a good algorithm should be able to reach any energy conservation value required by the user. To demonstrate this for LINCS, we simulated the actin-binding domain of villin headpiece (36 residues) with the OPLS all-atom force field.[17] To avoid cutoff artifacts all simulations were performed in vacuo without cutoffs. To ensure the same conditions for all LINCS accuracies, canonical simulations at 300 K were performed using a Nose-Hoover thermostat, implemented in GROMACS with a reversible leapfrog integrator.[18] The period of the temperature oscillations was set to 2 ps. The energy conservation accuracy is obtained from the drift of the conserved energy quantity in Nose-Hoover dynamics. The results for LINCS and SHAKE for simulations of 1 ns are shown in Table 1. One can see that with a logarithmic increase in computational effort the accuracy of LINCS can be increased to a finally unmeasurable drift over 1 ns in double precision. The amount of drift can be compared to another common source of drift, namely cutoff artifacts. Often plain cutoffs are use for nonbonded interactions, mainly for reasons of computational efficiency. A plain cutoff for the Lennard-Jones interactions of 0.9−1.1 nm with a neighbor-list update interval between 10 and 20 fs introduces energy into the system at a rate of of 1−10 k_BT/ns per degree of freedom. Reaction-field electrostatics produces 1 order of magnitude more drift.

The amount of time spent in the constraint algorithm is relatively high in this system. For a protein in solvent the relative time for LINCS will be a factor of 2−4 lower. In single precision it does not make sense to increase the order of matrix expansion above 6, since all further terms in the matrix are beyond the numerical precision compared to the first terms. Already for the case with two iterations and expansion order 6 the energy drift is unmeasurable in single precision, whereas in double precision there is a clear, although very small, negative drift. This is because in single precision numerical rounding errors cancel the small analytical error of the LINCS algorithm. The effect of the old way of constraining the velocities, using the changes in coordinates, can only be observed for these same settings. LINCS is computationally slightly more efficient than SHAKE, but this difference can probably be overcome by using overrelaxation.[19] When the original LINCS paper was written, a decade ago, LINCS was a factor of 4 faster than SHAKE. This difference has become much smaller, because modern processors can more efficiently execute code with conditional statements such as SHAKE.

Table 1. Accuracy of the LINCS Algorithm Applied to Villin with a Time Step of 2 fs in Single and Double Precision as a Function of the Number of Iterations N_i and the Order of the Expansion, in Terms of the Relative Root-Mean-Square Deviation (RMSD) of the Constraint Lengths and the Drift of the Conserved Energy in k_BT per Degree of Freedom[a]

	prec.	N_i	order	tol., 10^{-4}	RMSD, 10^{-4}	energy drift, ns^{-1}	time, ms	time, %
LINCS	single	1	4		0.27	−1.59	0.14	3.9
	single	1	6		0.11	−0.34	0.17	4.8
	single	2	6		0.02	0.00	0.24	6.5
	double	2	6		0.012	−0.03	0.30	4.8
	double	2	8		0.003	0.00	0.35	5.6
LINCS	single	1	4		0.27	−1.60	0.15	3.9
old	single	1	6		0.11	−0.35	0.17	4.8
v corr.	single	2	6		0.02	−0.08	0.24	6.5
SHAKE	single			1.00		−2.06	0.15	4.3
	single			0.10		−0.06	0.22	6.1
	double			0.10		−0.14	0.28	4.6
	double			0.01		−0.01	0.39	6.2

[a] The last column shows the CPU time spent in the constraint algorithm per step and as a percentage of the total run time. For comparison LINCS with the old, inaccurate velocity correction and SHAKE with difference relative constraint tolerances are also shown. Benchmarks were performed on one core of an Intel 2.4 GHz Core 2 CPU.

Table 2. As Table 1, but Only for LINCS in Single Precision for Villin with Virtual Interaction Sites

Δt	N_i	order	RMSD, 10^{-4}	energy drift, ns^{-1}	time, ms	time, %
2	1	4	0.22	−1.26	0.10	2.5
2	1	6	0.08	−0.23	0.13	3.7
4	1	6	0.31	−1.67	0.13	3.7
4	2	6	0.03	−0.15	0.18	5.1
5	1	6	0.50	−3.07	0.13	3.7
5	2	6	0.05	−0.26	0.18	5.1

A clear advantage of LINCS over SHAKE is its stability at large time steps. To show the performance of LINCS with large time steps villin was simulated with all hydrogens replaced by virtual sites or angle constraints[10] with time steps of 2, 4, and 5 fs. The results are shown in Table 2. For maintaining the constraint and integration accuracy with increasing time step, the order of the expansion and/or the number of LINCS iterations need to be increased. This also increases the computational effort slightly, but LINCS still takes a negligible amount of the total run time.

To illustrate the performance of P-LINCS, T4-lysozyme in a rectangular box of $5.2 \times 6.7 \times 5.2$ nm ^{3} with 5000 SPC water molecules[20] and 8 Cl^- ions was simulated with

Table 3. Performance of P-LINCs on Lysozyme in Water (See Text for Details) without and with Virtual Sites as a Function of the Time Step (Δt) and the Number of Domain Decomposition Cells, except for pd4 Which Is Particle Decomposition over 4 Processors[a]

					constraint time (ms)					speed (ns/day)				
virtual sites	Δt, fs	N_i	order	RMSD, 10^{-4}	1	pd4	4	16	32	1	pd4	4	16	32
no	2	1	4	0.21	2.4	2.4	3.8	5.7	8.8	3.0	10.2	11.8	41	70
yes	2	1	4	0.26	2.3	2.3	3.3	4.9	7.6	3.0	10.2	11.8	40	69
yes	2	1	6	0.07	2.5	2.5	3.6	5.9	9.0	3.0	10.2	11.6	40	67
yes	4	1	6	0.29	2.5	2.5	3.6	5.9	9.0	5.3	18.1	21.2	70	117
yes	5	1	6	0.47	2.5	2.5	3.6	5.9	9.0	6.3	21.5	25.1	83	137

[a] RMSD is the relative deviation of the constraint lengths, constraint time gives the time used by the constraint algorithms (LINCS/P-LINCS plus settle for water) in one integration step summed over all the processors, and speed is the simulation time per day. Benchmarks were performed on a 2.2 GHz AMD64 cluster with Infiniband interconnects.

the GROMOS 53a6 force-field[21] with united aliphatic carbons. To create a computationally demanding test case for P-LINCS, a plain cutoff of 1.1 nm was used for the Lennard-Jones interaction and the reaction-field electrostatics, with a neighbor-list update every 20 fs. These settings do not lead to very good energy conservation. Tapered cutoffs and/or particle-mesh Ewald electrostatics provide much better energy conservation, but are, in the GROMACS package, computationally roughly twice as expensive, since tabulated instead of analytical potentials need to be used. This will lower the relative computational cost of P-LINCS and is therefore a less demanding benchmark for P-LINCS.

Timings for P-LINCS only and the complete simulation, without and with virtual sites, are shown in Table 3 using a preliminary version of the GROMACS 4.0 package with load-balanced domain decomposition. The order of the expansion for P-LINCS needs to be adjusted with increasing time step, which leads to a marginally higher computational cost. The time for the constraint algorithm includes the time used by SETTLE for the water molecules. This combination of algorithms does not cause load imbalance, since SETTLE does not communicate and is done after P-LINCS. Processors that have a lot of water molecules to constrain have less protein constraints, and the two algorithms roughly balance out during the last LINCS iteration. One can see that the time for P-LINCS increases with an increasing number of processors/cells. This is not due to the extra constraints across the cell boundaries but nearly only due to communication. When going from 4 to 32 processors the domain decomposition goes from 1D to 3D, requiring from 1 to 3 communication steps per iteration. Since the time for a communication step in P-LINCS, which is latency bound, stays nearly constant, the P-LINCS time increases with the dimensionality of the domain decomposition. However, even on 32 processors P-LINCS still only takes 10% of the computation time. This number will halve when more accurate cutoff schemes are employed, even when an extra P-LINCS iteration is used for more accuracy. For comparison LINCS results are shown with particle decomposition on 4 processors, which is slightly slower than domain decomposition. Particle decomposition without parallel constraints is limited to 4 processors, since already on 5 processors there is a load imbalance when one processor needs to take care of the whole protein. It should also be noted that with a time step of 5 fs the neighbor list is updated every 4 steps, which is costly. In practice one would increase the neighbor-list cutoff and decrease the neighbor-list update frequency to improve perform ace. This was not done here to keep the simulations comparable.

As mentioned before, the P-LINCS communication is latency limited. For the lysozyme system with a matrix expansion order of 6, the number of atoms to be communicated between the different cell boundaries varies between 0 and 330 for 4 cells and between 0 and 130 for 16 or 32 cells. This means that in single precision the largest message is 4000 bytes for 4 cells and 1600 bytes for 16 or 32 cells. With an all-atom force field there are twice as many constraints, and, therefore, the numbers double. With hydrogen angle vibrations removed the numbers are twice as small, also for an all-atom force-field, since most constraints involving hydrogens dissappear. For such message sizes the communication is latency limited on any system. The message size depends mainly on the cell size. As the cell size increases, the message size also increases. Thus for large cells the communication may no longer be latency limited. But since the computational cost for the forces and constraints is proportional to the volume of the cell and the number of atoms for constraint communication proportional to its surface, the P-LINCS communication will never be a limiting factor.

V. Conclusions

P-LINCS is a parallel constraint algorithm which gives (binary) identical results to the nonparallel algorithm LINCS. It therefore has the same high stability. Its implementation on top of LINCS is straightforward, requiring only the coding of some bookkeeping and communication. The treatment of uncoupled angle constraints has been improved compared to the original version of the LINCS algorithm. The computational cost of P-LINCS increases with the dimensionality of the domain decomposition grid, since the communication is latency limited. But the time spent in P-LINCS is negligible on the total run time. The computational cost increases logarithmically with the required accuracy. For a typical protein simulation with the GROMACS package the total cost of P-LINCS with full 3D domain decomposition is between 4% and 10%, depending on the treatment of the electrostatics. In other packages this could be significantly less, since in GROMACS the force calculation is very efficient due to assembly SSE/SSE2 force loops.

Acknowledgment. The author thanks David van der Spoel and Erik Lindahl for stimulating discussions.

References

(1) Ryckaert, J. P.; Ciccotti, G.; Berendsen, H. J. C. *J. Comput. Phys.* **1977**, *23*, 327.

(2) Andersen, H. *J. Comput. Phys.* **1983**, *52*, 24.

(3) Brown, D.; Clarke, J. H. R.; Okuda, M.; Yamazaki, T. *Comput. Phys. Comm.* **1994**, *83* (1), 1.

(4) Hess, B.; Bekker, H.; Berendsen, H. J. C.; Fraaije, J. G. E. M. *J. Comput. Chem.* **1997**, *18*, 1463.

(5) Edberg, R.; Evans, D. J.; Morriss, G. P. *J. Chem. Phys.* **1986**, *84*, 6933.

(6) Baranyai, A.; Evans, D. J. *Mol. Phys.* **1990**, *70*, 53.

(7) Yoneya, M.; Berendsen, H. J. C.; Hirasawa, K. *Mol. Simul.* **1994**, *13*, 395.

(8) Slusher, J. T.; Cummings, P. T. *Mol. Sim.* **1996**, *18*, 213.

(9) van der Spoel, D.; Lindahl, E.; Hess, B.; Groenhof, G.; Mark, A. E.; Berendsen, H. J. C. *J. Comput. Chem.* **2005**, *26*, 1701.

(10) Feenstra, K. A.; Hess, B.; Berendsen, H. J. C. *J. Comput. Chem.* **1999**, *20*, 786.

(11) Miyamoto, S.; Kollman, P. A. *J. Comput. Chem.* **1992**, *13*, 952.

(12) Weinbach, Y.; Elber, R. *J. Comput. Phys.* **2005**, *209*, 193.

(13) de Leeuw, S. W.; Perram, J. W.; Petersen, H. G. *J. Stat. Phys.* **1990**, *61*, 1203.

(14) Bekker, H. Molecular Dynamics Simulation Methods Revised, Ph.D. Thesis, University of Groningen, The Netherlands, 1996.

(15) Landau, L.; Lifshitz, E. *Mechanics;* Pergamon Press: Oxford, 1961.

(16) Lippert, R. A.; Bowers, K. J.; Dror, R. O.; Eastwood, M. P.; Gregersen, B. A.; Klepeis, J. L.; Kolossvary, I.; Shaw, D. E. *J. Chem. Phys.* **2007**, *126*, 046101.

(17) Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J. *J. Am. Chem. Soc.* **1996**, *118*, 11225.

(18) Holian, B. L.; Voter, A. F.; Ravelo, R. *Phys. Rev. E* **1995**, *52* (3), 2338.

(19) Barth, E.; Kuczera, K.; Leimkuhler, B.; Skeel, R. D. *J. Comput. Chem.* **1996**, *16*, 1192.

(20) Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; Hermans, J. In *Intermolecular Forces*; Pullman, B., Ed.; D. Reidel Publishing Company: Dordrecht, 1981; pp 331−342.

(21) Oostenbrink, C.; Villa, A.; Mark, A. E.; van Gunsteren, W. F. *J. Comput. Chem.* **2004**, *25*, 1656.

CT700200B