Parallel Constrained Molecular Dynamics

Pierre-Eric Bernard, Olivier Coulaud

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Parallel Constrained Molecular Dynamics

Pierre-Eric Bernard — Olivier Coulaud

N° 3868

Janvier 2000

THÈME 4

ISSN 0249-6399 ISRN INRIA/RR--3868--FR+ENG

Parallel Constrained Molecular Dynamics

Pierre-Eric Bernard*, Olivier Coulaud

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Numath

Rapport de recherche n° 3868 — Janvier 2000 — 14 pages

Abstract: A Block version of the shake method for heavy atoms simulation in biological systems is presented in this paper. The method solves successively, independent blocks of constraints of small size by a Newton method. This algorithm is implemented in TAKAKAW, an efficient parallel molecular dynamics code. This method has been tested on a small system and on an ionic canal of 67671 atoms.

Key-words: Molecular Dynamics, Differential Algebraic System, Shake method, Constraints, Parallelism, Newton method.

* This work was performed as a part of SIMBIO, a research project on molecular simulation supported by Inria.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Un algorithme parallèle pour une dynamique moléculaire contrainte

Résumé : Nous présentons ici une méthode shake par bloc pour des simulations «atomes lourds» dans des sytèmes biologiques. Cette méthode résoud successivement des blocs indépendants de contraintes de petites tailles par une méthode de Newton. Cet algorithme est implémenté dans TAKAKAW un code parallèle de dynamique moléculaire. La méthode est testée sur un petit système puis sur un canal ionique de 67671 atomes.

Mots-clés : dynamique moléculaire, equations différentielles algébriques, contraintes, parallélisme, méthode de newton, algorithme shake.

1 Introduction

In classical molecular dynamics the motion of atoms in a system of molecules is described by the Newton's second law (cf. [1], [7]). In cartesian coordinates the equations are

$$M\frac{d^2q(t)}{dt^2} = F(q,t) = -\nabla V(q(t)) \tag{1}$$

where $q(t) \in \mathbb{R}^{3N}$ describes the position of atoms, N is the number of atoms, V is an empirical potential energy function and M is a diagonal matrix of atomic masses containing each mass replicated thrice.

The potential energy may be divided in two parts. The first one concerns the non-bonded interaction between atoms modeling van der Waals and electrostatic interactions and the second corresponds to intra-molecular geometrical potential linked to the skeleton of the molecule (bonds, angles, ...). In our application, the potential energy has the form

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

where q_i is the position of the ith atom, Ψ_{ij} represents the non-bonded pairwise potential including the van der Waals potential and Coulombic potential, $|.|$ denotes the Euclidean distance. The first three potentials on the right hand side of equation (2) are potentials linked to the structure of the molecules. In this paper, the term of interest is the bonds potential defined by

$$V_b(q) = \tfrac{1}{2}\sum_{k=1}^{m} K_k^b(|q_{r(k)} - q_{l(k)}| - L_k)^2 \tag{3}$$

where m is the number of bonds, L_k is the equilibrium length of the kth bond and K_k^b is the force constant, $l(k), r(k)$ are the number of the two atoms involved in the kth bond. Potential V_θ is also a harmonic potential and it depends on the angle between two bonds linking three atoms. V_ϕ is the sum of a potential of torsion along a bond based on the dihedral angle and a harmonic potential also based on the dihedral angle. The force constants of the harmonic bond potentials are an order of magnitude higher than the potentials based on angles. So the highest vibration modes are generated by the bond potentials.

These potentials are highly nonlinear, oscillatory and they differ by their strength and time scale. The dynamic of biological molecules is highly oscillatory and very complex, even chaotic. In molecular simulations, we are interested in generating acceptable statistical trajectory and sampling the configurational space over long time periods. It is not necessary to follow the exact trajectories. The conformational change occurs in a continuum manner over a few pico-second to seconds time scale, for example, the folding of proteins occurs in about a second. The Newton equations (1) form a Hamiltonian system with the Hamiltonian $H(v,q) = 1/2v^T Mv + V(q)$, where v is the velocities vector of q, so the equations can be written

$$\begin{cases} \dfrac{dq}{dt} &= v \\ M\dfrac{dv}{dt} &= -\nabla_q V(q) \end{cases} \tag{4}$$

To solve this Hamiltonian system, the best algorithms for long time integration are the symplectic integrators [10]. With these methods, we integrate naturally in the micro-canonical system (NVE) which means the number of atoms, the volume and the energy (here the Hamiltonian) are constant.

One of the most popular algorithm for Molecular Dynamics (MD) is the following leapfrog method

$$\begin{cases} q_{n+1} = q_n + \Delta t(v_{n-1/2} - \Delta t M^{-1}\nabla V(q_n)) \\ v_{n+1/2} = \dfrac{q_{n+1} - q_{n-1}}{\Delta t} \end{cases} \tag{5}$$

where Δt is the time-step, q_n the vector of positions of atoms at time $n\Delta t$, $v_{n+1/2}$, the vector of their velocity at time $(n+1/2)\Delta t$, $V(q_n)$ is the potential evaluated at position q_n.

Since this method is explicit the time-step is limited by the highest frequencies of oscillation in the system. For example, since the frequency of vibration of C-H bond is roughly about $0.9\ 10^{14}\ Hz$, we have to impose a time-step of around $1fs = 10^{-15}\ s$. As we have said before, for long simulations, we are not interested in following exactly the oscillating trajectories, but only the "mean" trajectories. Many methods are available to solve this high frequency problems such as implicit methods, LI method and constraint methods (for a review of these methods see [12]). In MD, we suppress the highest oscillations and to do this we introduce constraints in the system whereby the system becomes a Differential Algebraic System (DAS). Due to the magnitude of the spring constants, the bond oscillations generate the highest frequency. We decompose the potential V as $V(q) = U(q) + V_b(q)$, and introduce the Lagrange multipliers $\Lambda = (\lambda_1, ..., \lambda_m)$ as well as m length constraints $g = (g_1, ..., g_m)$ of the form

$$g_i(q) = \tfrac{1}{2}(|q_{r(i)} - q_{l(i)}|^2 - L_i^2) = 0. \tag{6}$$

Then the constrained Hamiltonian system becomes

$$\begin{cases} \dfrac{dq}{dt} &= v \\ M\dfrac{dv}{dt} &= -\nabla_q U(q) - g_q(q)^T\Lambda \\ g(q) &= 0 \end{cases} \tag{7}$$

where $G(q) = g_q(q) = [\dfrac{\partial g_i(q)}{\partial q_j}] \in \mathbb{R}^{m\mathrm{x}3N}$ with $m \leq N$ is the Jacobian matrix. The system (7) has $N-m$ degrees of freedom. The algebraic constraint g_k suppresses the associated frequency of vibration of the two atoms involved in the kth bond. The constraint g_i is a holonomic constraint, and the system is a DAS of index 3. If we differentiate the constraints twice, we obtain the hidden constraint $G(q)v = 0$ and Λ is implicitly defined by

$$R(q)\Lambda = G(q)M^{-1}G^T(q)\Lambda = g_{qq}\, v\, v - G(q)M^{-1}\nabla_q U$$

where g_{qq} is the hessian of g.

2 Constraint method

For solving (7) we apply the shake algorithm defined by

$$\begin{cases} q_{n+1} = q_n + \Delta t v_{n+1/2} \\ v_{n+1/2} = v_{n-1/2} - \Delta t[M^{-1}\nabla_q U(q_n) + g_q(q_n)^T \Lambda_n] \\ g(q_{n+1}) = 0 \end{cases} \tag{8}$$

This algorithm does not satisfy the hidden constraint on the velocity. If we wish to impose it, we need to use the Rattle method. In this case, we impose the two constraints and then we have two nonlinear systems to solve. From a simulation point of view the trajectories in the two algorithms are equivalent [8].

The shake method described by (8) is a reversible and symplectic method. For the rest of the paper, we introduce the new variable $\Lambda' = \Delta t^2 \Lambda$. Moreover, for the convenience of the reader we rename Λ' by Λ.

2.1 Classical approach

The right way to compute the new position in equations (8) [1] is to compute first the position of the atoms by the leapfrog method (5)

$$\bar{q} := q_n + \Delta t M^{-1}[V_{n-1/2} - \Delta t \nabla_q U(q_n)], \tag{9}$$

and then we compute a correct position satisfying the constraints by solving the nonlinear equation

$$0 = F(\Lambda) = (F_1(\Lambda), \ldots, F_m(\Lambda)) := g(\bar{q} - M^{-1} g_q(q_n)^T \Lambda). \tag{10}$$

There are two classes of methods to solve the nonlinear system (10). They can be characterized as "local" and "global" methods. In the local methods, we satisfy constrains successively one by one as in the classical methods of Gauss-Seidel Newton and SOR-Newton. On the other hand, global methods as Newton, consider all constraints and directly solve the nonlinear system by using sparse matrix technics to solve the system. In the next two subsections we describe rapidly Gauss-Seidel and Newton methods. If the reader is interested in more details we refer to [2], [12].

2.1.1 Gauss-Seidel Newton method

This is the most commonly used method to solve nonlinear equations (10). The idea is to solve iteratively, one after another, each of the constraints [16]. This is why we introduce F_i the ith component of F which satisfies

$$F_i(\Lambda) = g_i(\bar{q} - M^{-1} G^T(q^n)\Lambda),$$

and then we solve

$$f_i(\lambda_i^k) = F_i(\lambda_1^k, \lambda_2^k, \ldots, \lambda_{i-1}^k, \lambda_i^k, \lambda_{i+1}^{k-1}, \ldots, \lambda_m^{k-1}) = 0, \tag{11}$$

by Newton method. This gives

$$\lambda_i^k = \lambda_i^{k-1} - \frac{f_i(\bar{q} - M^{-1}G^T(q^n)\Lambda^{k-1})}{\partial f_i(\bar{q} - M^{-1}G^T(q^n)\Lambda^{k-1})},$$

where $\Lambda = (\lambda_1, ..., \lambda_m)$ and $\partial f_i(q) = -G_i(q)M^{-1}G_i(q^n)$.

One iteration of the Gauss-Seidel method on the ith constraint gives the following Newton Algorithm

$$\begin{aligned}
\Delta\lambda_i &= -\frac{m_{r(i)}m_{l(i)}}{m_{r(i)} + m_{l(i)}} \quad \frac{g_i(q^{n,k-1})}{< q_{r(i)}^{n,k-1} - q_{l(i)}^{n,k-1}, q_{r(i)}^n - q_{l(i)}^n >} \\
\lambda_{i+1} &= \lambda_i - \Delta\lambda_i \\
q_{r(i)}^{n,k} &= q_{r(i)}^{n,k-1} - \frac{\Delta\lambda_i}{m_{r(i)}}(q_{r(i)}^n - q_{l(i)}^n) \\
q_{l(i)}^{n,k} &= q_{l(i)}^{n,k-1} + \frac{\Delta\lambda_i}{m_{l(i)}}(q_{r(i)}^n - q_{l(i)}^n)
\end{aligned}$$

where q_j^n is the position of the jth atom at time $n\Delta t$ and $q_j^{n,k}$ its position at iteration k of the newton process.

The asymptotic speed of convergence of the method is given by the spectral radius of the matrix $(D + L)^{-1}U$ where $R = G(q^*)M^{-1}G^T(q^n) = L + D + U$ and q^* is the limit of the iterations (see [16]). It is well known that the convergence of Gauss-Seidel is slow. It could be accelerated by an over-relaxation process, but there is no criteria to choose automatically the optimal parameter of relaxation. Nevertheless, we can introduce an adaptative process to find a good approximation of that parameter during the first iterations of the method.

2.1.2 Newton method

The second class of methods to solve (10) uses sparse matrix technics. If we use the Newton method the algorithm is

$$\begin{cases} (\partial_\Lambda F)\delta\Lambda = F(\Lambda^k) \\ \Lambda^{k+1} = \Lambda^k - \delta\Lambda \\ q^{n,k+1} = q^{n,k} - M^{-1}G(q_n)^T\delta\Lambda \end{cases} \tag{12}$$

where $q^{n,0} = \overline{q}$ is the position from the leapfrog scheme which corresponds to $\Lambda = 0$, $\partial_\Lambda F$ is the Jacobian matrix defined by

$$\partial_\Lambda F = -G(q^{n,k})M^{-1}G^T(q^n). \tag{13}$$

We then need to solve a non symmetric linear system at each iteration of the Newton method. Symmetric Newton methods (see [2]) or more efficient Newton-like method [17] can also be used.

2.1.3 Block Newton method

We numerate the bonds of the molecular system molecule by molecule, for example, we begin by the bonds in the protein followed by the bonds in the solvent a molecule at a time. So we obtain

the following structure of the Jacobian matrix

$$R = \left(\begin{array}{c|ccc} \mathrm{R}_0 & & & \\ \hline & R_1 & & \\ & & R_2 & \\ & & & \cdots \\ & & & R_M \end{array} \right) \tag{14}$$

where R_0 is a sparse matrix corresponding to the bonds of the protein, and R_i for $i > 0$, is a full small matrix corresponding of the bonds of th ith molecule of solvent. Due to the structure of the matrix R it is easy to see that we have independent sets of constraints. If we decompose the two vectors of size m, $\delta\Lambda = (\delta\Lambda_0, \delta\Lambda_1, ..., \delta\Lambda_M)$ and $\Lambda = (\Lambda_0, \Lambda_1, ..., \Lambda_M)$ then the system $R\delta\Lambda = H = H(\Lambda)$ can be rewritten

$$R\delta\Lambda = \left(\begin{array}{c} R_0\delta\Lambda_0 \\ R_1\delta\Lambda_1 \\ \cdots \\ R_M\delta\Lambda_M \end{array} \right) = H = \left(\begin{array}{c} F_0(\Lambda_0) \\ F_1(\Lambda_1) \\ \cdots \\ F_M(\Lambda_M) \end{array} \right). \tag{15}$$

We have $M + 1$ independent systems to solve $R_i\delta\Lambda_i = F_i(\Lambda_i)$, so the idea is to solve them in parallel.

2.2 Shake on Hydrogen Atoms

2.2.1 Description

In many simulations it is sufficient to constrain only the bonds between heavy atoms like carbon, oxygen, nitrogen, ... and the light atom i.e. hydrogen. Such a simulation is called a heavy atoms simulation. In this case, the complete matrix R is a block diagonal matrix, and the size of the block is at most 4. The maximum size is attained for the molecule CH4, but it is not considered in biological simulation, where the maximum size is only 3 (TIP3P water molecule [14]). For different sizes of blocks we represent in Figure 1 and 2 examples of the configuration of atoms involved in the block. In biological system, type 1 in Figure 2 never appears when

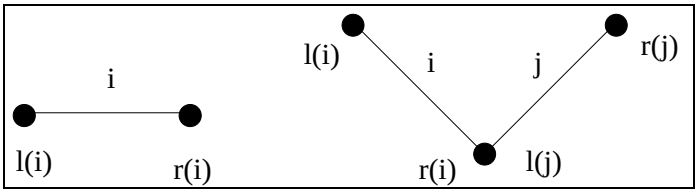

Figure 1: Atoms configurations with one and two bonds

we constrain only the light atom (Hydrogen). The second type appears in the protein when one atom is shared by three bonds typically -CH3 group. The third configuration represents the TIP3P water molecule where we have a set of cyclic bonds corresponding to the two O-H bonds and also the H-H bond. In this case the additional constraint freezes the angle oscillations so that the molecule behaves like a rigid molecule.

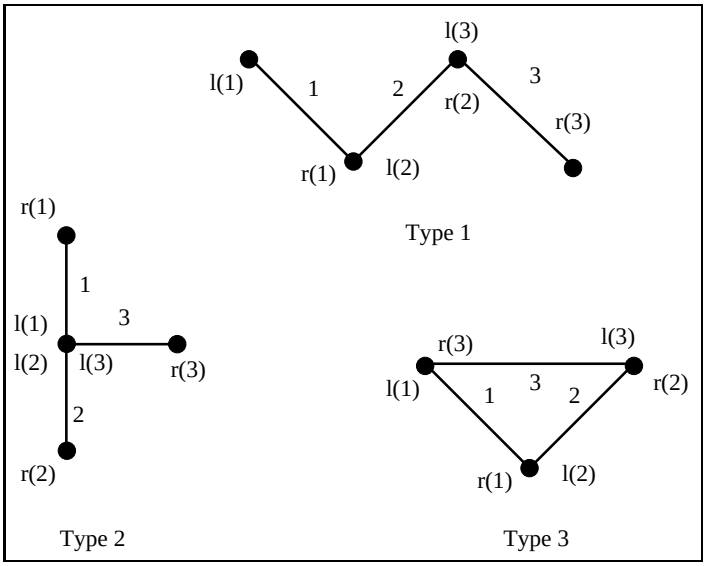

Figure 2: Atoms Configurations with 3 bonds

The bond potential is now decomposed into two parts, the potential for bonds between two heavy atoms denoted by V_b^{X-X}, and the V_b^{X-H} is the potential for bonds between an heavy atom and hydrogen atom (H) as follows

$$V_b(q) = V_b^{X-X}(q) + V_b^{X-H}(q).$$

The global potential V is now decomposed as $V(q) = U(q) + V_b^{X-H}(q)$ and we apply the shake algorithm on $V_b^{X-H}(q)$. In the constrained bonds there are in some simulations (for example with TIP3P water molecule) some bonds between two hydrogen atoms.

2.2.2 A mixed approach implementation

We have implemented the shake algorithm in TAKAKAW, a parallel code designed to handle large proteins in biology ([4], [5]). TAKAKAW simulates large systems of atoms with geometrical forces (Xplor potential) and van der Waals, Coulomb forces with the cutoff approximation. The parallel code has been developed by using the parallel environment Athapascan[1][3] based on multi-threading. It employs both the communication library MPI [9] and a kernel of posix threads [15].

The principle of the code is based on a spatial decomposition, also called the link-cell method [13], which corresponds to a geometric decomposition of the domain. Each part of the physical domain is assigned to a processor, the main characteristic is that atoms are not assigned to a given processor, but they are allowed to move from one processor to its neighbors.

As atoms can pass to another processor during the simulation we attach to a heavy atom all hydrogen atoms involved through a bond with it. This means that hydrogen atoms are placed on the same processor as the heavy atom to which it is bonded and, the heavy atoms drive the displacement of all associated hydrogen atoms. So, when a heavy atom changes processor

[1] ATHAPASCAN is developed inside the project Apache in Grenoble, the research project is supported by Inria

we also move the hydrogen atoms bonded to it. Hence we have **no communication** between processors during the implementation of shake algorithm and there is also **no synchronization** at this level. So, the influence of the shake algorithm, based on blocks of constraints, on the efficiency of the parallel code depends on a good balancing of constraints on the processors. In fact the placement of the constraints on the processors is induced by the placement of the boxes on those processors. Several algorithm of placement are available in Takakaw : random, LPTF , BPR and BPRF based on the work performed in the evaluation of non bonded interactions.

The initial allocation is computed by an algorithm based on a recursive bisection (BPR and BPRF) for distributing the boxes onto the processors. As we have already mentioned, the domain has been decomposed into cubic boxes whose side is larger or equal to the cut-off radius. The computational load of non-bonded interactions is subdivided into tasks associated with boxes: the tasks for computation of non-bonded interactions for each box and for each pair of neighbor boxes. The recursive bisection method allows to capture the trade-off between a good repartition of the number of atoms (which is proportional to local computations) and a relatively small communication cost [11]. For BPRF we have a finest decomposition, i.e. not smooth interface between the set of boxes, which induces a better load-balancing. The LPTF is a pure load balancing strategy which does not take into account any communication: a greedy algorithm based on the well-known LPT rule (which stands for Largest Processing Time first [6]). We refer to [5] for more details.

Due to (15), each processor has a set of independent constraints which are solved by either the Gauss-Seidel method Newton or the method as we will see now. We consider two methods, the first is the classical Gauss-Seidel Newton denoted by GSN. The second is called Block-Newton (BN). The idea here, is to solve by the newton algorithm the blocks of constraints. To do this, all the constraints sharing one atom are regrouped in a block. The block types are sorted and enumerated as described in figures 1 and 2. Then we solve iteratively all the independent blocks by Newton method (12).

3 Numerical results

Numerical results are presented on two molecular systems in order to compare the two methods GSN and NB described above. The first system is small with 3625 atoms, the second is an ionic canal of 67 598 atoms. All the computations are done on the Origin200 with R10000 at 195 MHz processors.

3.1 A small system : the bacteriorhodopsin

This system is a small structure composed of a protein of 3544 atoms and 27 molecules of water in vacuum. We have 1887 constraints which correspond to 1184 blocks, the different types of blocks are given in Table 1. To compute the non-bonded forces we use a cutoff of 10Å, this

Table 1: Type of the constraints

type 1	type 2	type 3	constraints
705	255	224	1 887

induces 72 boxes of size 12Å distributed on the processors. All the results shown in Tab 2 to 4 are from simulations on one processor.

Table 2 presents the overhead for the different Shake methods when the tolerance for the constraints, ϵ, is fixed to 10^{-6} this means that

$$max\{|g_i(r)| \; for \; i = 1, \ldots, m\} \leq \epsilon. \tag{16}$$

In the Table 3, we present the average time for one iteration with different values of the tolerance

Table 2: Average time in seconds for one iteration

Leapfrog	GSN	NB
0.255	0.274	0.258

for the constraints. The time step is of 5.0 10^{-4} ps and we integrate during 100 time-steps. As

Table 3: Average time in second for one iteration, $\delta t = 5.0\ 10^{-4}$

ϵ	1.0 10^{-4}	5.0 10^{-6}	1.0 10^{-9}	1.0 10^{-12}
GSN	0.267	0.278	0.295	0.311
NB	0.258	0.257	0.258	0.257

we can see, the Block Newton method is faster. This is due to the fact we solve directly all the dependent constraints. As we can see, the choice of the tolerance parameter is not important in the Block Newton algorithm. The last Table for this system, show the influence of the time step

Table 4: Average time in second for one iteration

Δt	5.0 10^{-4}	1.0 10^{-3}	4.0 10^{-3}
GSN	0.263	0.278	0.291
NB	0.257	0.257	0.259

on the average iteration time. In Table 4 the tolerance for the constraints is 10^{-6}.

The next two tables show the influence of our strategy of placement of the atoms on the parallelism. Now the simulation is performed over 1000 steps and the structures are updated each 100 steps. This mean that during each update, data associated with each of the atom including constraints which has moved accross to an adjacent box which may be attached to another processor has to be communicated to the new processor. The tolerance for the constraints defined by (16) is fixed to 10^{-7}. In Tab 5 we see the influence of our new strategy of placement, on the numbers of atoms, on the constraints per processors and the mean time of one iteration. For Leapfrog algorithm [5] have shown that BPRF is the best static placement for the data which gives a good load balancing followed by LPTF. If we consider BPRF placement, table 5 shows that there is no influence of the shake algorithm on the load balancing due to the modification of the strategy of placement of atoms into boxes. In the next table, we show the influence of the shake newton block algorithm on the speedup. The speedup is defined by the ration of the sequential time over the time in parallel.

3.2 A large system : KCSA tetrameric potassium canal

The second system is a membrane, the calculations are done in a box of size 96 Å by 96 Å by 84 Å, under periodic boundary conditions. The system is constituted by 67 671 atoms and 67 509

Table 5: Influence of the placement

Placement		Proc. 1	Proc. 2	Proc. 3	time in sec.
RANDOM	Number of constraints	693	436	758	
	Number of atoms	1305	852	1468	
	Number of atoms without Shake	1308	858	1459	
	Average time of one iteration				0.331
BPR	Number of constraints	494	959	434	
	Number of atoms	974	1793	858	
	Number of atoms without Shake	982	1788	855	
	Average time of one iteration				0.300
LPTF	Number of constraints	659	630	598	
	Number of atoms	1227	1231	1167	
	Number of atoms without Shake	1212	1215	1198	
	Average time of one iteration				0.274
BPRF	Number of constraints	588	701	598	
	Number of atoms	1097	1366	1162	
	Number of atoms without Shake	1099	1363	1163	
	Average time of one iteration				0.265

Table 6: comparison of Speedup between leapfrog and Shake

Number of processor	1	4	8
Leapfrog	1	3.71	6.58
Shake GS	1	3.70	6.33
Shake NB	1	3.74	6.23

bonds. We have 53 954 constraints which correspond to 21 256 blocks, the different types of blocks are described in Table 7.

Table 7: Type of the constraints

type 1	type 2	type 3	constraints
1 435	6 944	12 877	53 954

In all simulations of this section the cutoff for the non-bonded forces is 10 Å, this induces 384 boxes of size 12 Å distributed on the processors with the BPTF placement. The tolerance for the constraints is fixed to 10^{-7}.

In Table 8, we consider an adimensional time-step of 4.0 10^{-3} ps (4.0 femtosecond) and we integrate only on 20 time-steps. In this case, for large Δt the Block Newton is really interesting

Table 8: Average time in second of one iteration

GSN	NB
0.381	0.276

because firstly the initial value given by (9) is not close to the solution. Secondly we have a large number of constraints of type 3 for which we solve the constraints faster. The figure 3 shows us

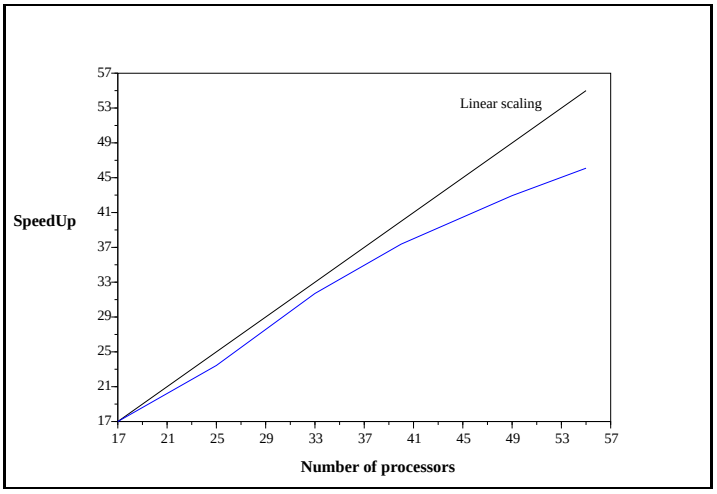

Figure 3: SpeedUp

the speedup of the methods. For this simulation we integrate on 150 time steps and we update the position of atoms, the constraints, ... on all the processors every 20 iterations. The time step is one picosecond ($1.0\ 10^{-3}$). Because the system is large, we begin the simulation on 17 processors to suppress the influence of the cache on the speedup. If this is not done, we have a super linear speedup until 55 processors.

In conclusion, the block Newton method is really interesting to solve quickly and with a great precision the system of blocks of constraints. Our parallel implementation of the shake algorithm gives nearly the same speedup as by the code without shake.

4 Appendix

Coefficients of Matrix R for block of three constraints Consider constraint of the form

$$g_i(q) = \tfrac{1}{2}(|q_{r(i)} - q_{l(i)}|^2 - L_i^2) = 0$$

then the Jacobian of the ith constraint is

$$\nabla g_i(q) = \left((q_{r(i)} - q_{l(i)})\delta_{r(i),j} - (q_{r(i)} - q_{l(i)})\delta_{l(i),j}\right)_{j=1..N}.$$

The coefficients of matrix $\partial_\Lambda F = -R$ defined in (13) of the linear system for $i, j = 1..3$ are given by

$$R_{i,j} = v_{rl}^i \overline{v}_{rl}^j \left(\tfrac{1}{m_{r(i)}}\delta_{r(i),r(j)} - \tfrac{1}{m_{r(i)}}\delta_{r(i),l(j)} - \tfrac{1}{m_{l(i)}}\delta_{l(i),r(j)} + \tfrac{1}{m_{l(i)}}\delta_{l(i),l(j)}\right)$$

where $v_{rl}^i = q_{r(i)} - q_{l(i)}$ and $\overline{v}_{rl}^i = q_{r(i)}^n - q_{l(i)}^n$. If we denote $\mu_i = \frac{1}{m_{r(i)}} + \frac{1}{m_{l(i)}}$, then the diagonal coefficients are

$$R_{i,i} = \mu_i v_{rl}^i \overline{v}_{rl}^i \quad for \quad i = 1..3.$$

Now, we define explicitly the coefficients of the matrix R for type 2 and type 3 for block of three constraints. For type 2 the matrix R are given by

$$R_{type2} = \begin{pmatrix} \mu_0 v_{rl}^0 \overline{v}_{rl}^0 & \frac{1}{m} v_{rl}^0 . \overline{v}_{rl}^1 & \frac{1}{m} v_{rl}^0 . \overline{v}_{rl}^2 \\ \frac{1}{m} v_{rl}^1 . \overline{v}_{rl}^0 & \mu_1 v_{rl}^1 . \overline{v}_{rl}^1 & \frac{1}{m} v_{rl}^1 . \overline{v}_{rl}^2 \\ \frac{1}{m} v_{rl}^2 . \overline{v}_{rl}^0 & \frac{1}{m} v_{rl}^2 . \overline{v}_{rl}^1 & \mu_2 v_{rl}^2 \overline{v}_{rl}^2 \end{pmatrix},$$

where $m = m_{l(1)} = m_{l(2)} = m_{l(3)}$ is the mass of the heavy atom connected to the three bonds. The matrix R for type 3 writes

$$R_{type3} = \begin{pmatrix} \mu_0 v_{rl}^0 . \overline{v}_{rl}^0 & -\frac{1}{m_{r(0)}} v_{rl}^0 . \overline{v}_{rl}^1 & -\frac{1}{m_{r(0)}} v_{rl}^0 . \overline{v}_{rl}^2 \\ -\frac{1}{m_{l(1)}} v_{rl}^1 . \overline{v}_{rl}^0 & \mu_1 v_{rl}^1 . \overline{v}_{rl}^1 & -\frac{1}{m_{r(1)}} v_{rl}^1 . \overline{v}_{rl}^2 \\ -\frac{1}{m_{r(2)}} v_{rl}^2 . \overline{v}_{rl}^0 & -\frac{1}{m_{l(2)}} v_{rl}^2 . \overline{v}_{rl}^1 & \mu_2 v_{rl}^2 . \overline{v}_{rl}^2 \end{pmatrix}.$$

Acknowledgement. We want to thank the *Centre Charles Hermite* research center, for allowing us to access to their 64 processors Origin 2000.

References

[1] M.P. Allen and D.J. Tildesley, *Computer Simulation of liquids*, Clarendon Press Oxford 1987.

[2] E. Barth, K. Kuczera, B. Leimkuhler and R. Skeel, Algorithms for constraints molecular dynamics, Journal of Computational Chemistry, Vol 16, No 10, 1192-1209 1995.

[3] J. Briat, I. Ginzburg, M.Pasin, and B. Plateau, Athapascan Runtime : Efficiency for Irregular in Proceedings of the Europar'97 Conference, Passau, Germany, pp 590–599, Aug, 1997

[4] P.E. Bernard, Parallélisation et multiprogrammation pour une application irégulière de dynamique moléculaire opérationnelle, Thèse de l'Institut National Polytechnique de Grenoble, Oct. 1997.

[5] P.E. Bernard, T. Gautier and D. Trystram, Large Scale Simulation of Parallel Molecular Dynamics , in Proceedings of Second Merged Symposium IPPS/SPDP 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing, pp 638–644, april, 1999.

[6] R.L. Graham, Bounds for Certain Multiprocessor Anomalies, Bell System Tech J., vol 45, pp 1563–1581, 1966.

[7] D. Frenkel and B. Smit, *Understanding Molecular Simulation From Algorithms to Applications*, Academic Press, 1996.

[8] B. Leimkuhler and R.D. Sekel, Symplectic Numerical Integrators in Constrained Hamiltonian System, Journal of Computational Physics, vol 112, pp 117-125, 1994.

[9] W. gropp, E. Lusk and A. Skjellum, *Using MPI - Portable Parallel Programming with Message Passing Interface* MIT press, 1994.

[10] J.M. Sanz-Serna, Symplectic integrators for Hamiltonian problems: an overview, Acta Numerica),pp 241-286, 1992.

[11] LR.J. Lipton, D.J. Rose, and R. Tarjan Endre, Generalized nested dissection, Stanford University, Department of Computer Science Technical Report CS-TR-77-645, 1995.

[12] L. Petzold, L. Jay, J. Yen, Numerical solution of highly oscillatory ordinary differential equations, Acta Numerica,pp 437-487, 1997.

[13] S. Plimpton, Fast Parallel Algorithms for Short-Range Molecular dynamics, Journal of Computational Physics, 117, pp 1-19, 1995.

[14] W.L. Jorgensen, J. Chandrasekhar, J. Madura, R.W. Impey and M.L. Klein, Comparison of simple potential functions for simulating liquid water, J. Chem. Phys.vol 79, No 2, 1983.

[15] D.R. Butenhof, *Programming with posix threads*, Addison-Wesley, Professional Computing Series, 1997.

[16] J.M. Ortega and C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*, Academic Press, New-York 70.

[17] J. Yen and L. Petzold, an efficient newton-type iteration for the numerical solution of highly oscillatory constrained multibody dynamic systems, Siam J. Sci. Comput. vol 19; no 5, pp 1513-1534, 1998.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399