Computer Physics Communications 288 (2023) 108742

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Accurate and efficient constrained molecular dynamics of polymers using Newton's method and special purpose code ☆

Lorién López-Villellas [a,1], Carl Christian Kjelgaard Mikkelsen [b], Juan José Galano-Frutos [c], Santiago Marco-Sola [a,d], Jesús Alastruey-Benedé [e,*], Pablo Ibáñez [e], Miquel Moretó [a,d], Javier Sancho [c], Pablo García-Risueño [2]

[a] Barcelona Supercomputing Center, Barcelona, Spain
[b] Department of Computing Science and HPC2N, Umeå, Sweden
[c] Department of Biochemistry, Molecular and Cellular Biology / Biocomputation and Complex Systems Physics Institute (BIFI), Universidad de Zaragoza, Zaragoza, Spain
[d] Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
[e] Departamento de Informática e Ingeniería de Sistemas / Aragón Institute for Engineering Research (I3A), Universidad de Zaragoza, Zaragoza, Spain

ARTICLE INFO

Article history:
Received 22 December 2022
Received in revised form 21 March 2023
Accepted 24 March 2023
Available online 29 March 2023

Keywords:
Molecular dynamics
Constraint algorithms
Non-linear equations
Newton's method
SHAKE
LINCS

ABSTRACT

In molecular dynamics simulations we can often increase the time step by imposing constraints on bond lengths and bond angles. This allows us to extend the length of the time interval and therefore the range of physical phenomena that we can afford to simulate. We examine the existing algorithms and software for solving nonlinear constraint equations in parallel and we explain why it is necessary to advance the state-of-the-art. We present ILVES-PC, a new algorithm for imposing bond constraints on proteins accurately and efficiently. It solves the same system of differential algebraic equations as the celebrated SHAKE algorithm, but ILVES-PC solves the nonlinear constraint equations using Newton's method rather than the nonlinear Gauss-Seidel method. Moreover, ILVES-PC solves the necessary linear systems using a specialized linear solver that exploits the structure of the protein. ILVES-PC can rapidly solve constraint equations as accurately as the hardware will allow. The run-time of ILVES-PC is proportional to the number of constraints. We have integrated ILVES-PC into GROMACS and simulated proteins of different sizes. Compared with SHAKE, we have achieved speedups of up to 4.9× in single-threaded executions and up to 76× in shared-memory multi-threaded executions. Moreover, ILVES-PC is more accurate than P-LINCS algorithm. Our work is a proof-of-concept of the utility of software designed specifically for the simulation of polymers.

1. Introduction and motivation

Molecular simulation is a powerful research tool for scientific and technological purposes. It is applied to a wide range of problems in chemistry and biology, such as the development of novel materials [1] or biomedicines, e.g., for fighting cancer [2] and infectious diseases, like the SARS-CoV-2 [3,4]. One of the most widely used techniques for molecular simulations is molecular dynamics (MD) [5,6], which calculates the time evolution of molecular systems subject to Newton's equations, thus enabling the calculation of a variety of quantities whose measurement in laboratories is frequently either difficult or unfeasible. The impact of molecular simulation is expected to increase greatly due to the continuous improvement of available computational capabilities [7] and calculation methods [8]. Among the former, we highlight the successive generations of the Anton supercomputers [9]; among the latter, the solution of the protein folding problem by AlphaFold2 [10]. The availability of 3D structures of proteins provided by AlphaFold2 will probably boost their simulations, e.g., for analysing their capabilities as catalysts or medicines or for a more accurate interpretation of the effect of mutations on the phenotype [11]. The availability of accurate and efficient methods for such simulations does hereby acquire a novel boost.

☆ The review of this paper was arranged by Prof. W. Jong.

* Corresponding author.
E-mail addresses: jalastru@unizar.es (J. Alastruey-Benedé), risueno@unizar.es (P. García-Risueño).
[1] Now at Departamento de Informática e Ingeniería de Sistemas / Aragón Institute for Engineering Research (I3A), Universidad de Zaragoza.
[2] Independent scholar.

https://doi.org/10.1016/j.cpc.2023.108742

In molecular dynamics the time step can be incremented through the imposition of constraints on internal degrees of freedom, which raises the total simulated time for equal computational effort. Imposing a constraint on a bond between two atoms consists in fixing a constant distance between these atoms equal to the length of the bond. This enables researchers to simulate a wider collection of phenomena of interest, which frequently take place at large time scales.

Imposing constraints on all the bonds of a molecule is a difficult problem to parallelize because the constraints introduce dependencies between every pair of atoms. Some molecular dynamics packages limit the imposition of constraints to small clusters of atoms (LAMMPS [12]) or to bonds involving Hydrogen atoms only (NAMD [13]). This limitation implies that either the goal of increasing the simulation time step is forgone or a larger error is tolerated for bonds for which no constraints have been imposed. On the other hand, tools like GROMACS allow to impose bond constraints between all the bonds of the molecule and even angle constraints.

In this article we consider the unfortunate effects of solving the constraint equations inaccurately and we present a constraint algorithm that is not only accurate but also very fast. We expect that the new algorithm will allow for simulations that are more accurate and faster than the current state-of-the-art. In addition, our article provides a proof-of-concept of the utility of software that is designed specifically for the simulations of polymers. Our method is expected to cross disciplinary boundaries because it can be applied to numerous problems in physics, chemistry and biology.

We shall now explain why it is necessary to advance the state-of-the-art for the problem of solving nonlinear constraint equations in parallel. To this end, we ask our readers to consider their favourite simulation of molecular dynamics with constraints. It is extremely likely that the majority of the run-time is spent *outside* the constraint solver. It is therefore possible to argue that improving the constraint solver serves no purpose because the reduction of the run-time will be relatively low. We shall now examine the flaws of this argument. The argument assumes that the constraint solver will always converge. The successful application of the LINCS and P-LINCS algorithms hinges on the rapid convergence of a specific power series. There exist molecules for which the series diverges [14,15]. The argument also assumes that the chosen simulation is representative of *all* future simulations. This cannot be true. In particular, it is likely that we will continue to extend the length of our simulations. At this point we have to contemplate the quality of our simulations. Are they a faithful representation of the underlying physics? Unless the constraint equations are solved exactly, they will be introducing an increasing distortion in the total mechanical energy. It is an experimental fact that the rate of the energy drift is an increasing function of the tolerance used by the constraint solver. It is also an experimental fact that the energy drift tends to increase with the length of the time interval. In short, if the length of the interval is long enough and if the tolerance is large enough, then the violation of the underlying physics can be arbitrarily large [16,17]. In this case, we have to choose between two options. We either prove that the conclusions drawn from the simulation are valid or we reduce the issue as much as possible by solving the constraint equations as accurately as the hardware will allow. We fear that the first option is impossible in general, but we believe that the second option is practical for general polymers. In this paper we take the first major step by considering proteins.

The paper is organized as follows. Section 2 motivates the need for a constraint algorithm to achieve high performance and ensure constraint enforcement to a high degree of accuracy. In Section 3 we propose ILVES-PC, a constraint algorithm that performs calculations of constraint forces for proteins made of the 20 proteinogenic amino acids. Section 4 describes our experimental methodology. In Section 5 we present the evaluation of ILVES-PC, which has been integrated into GROMACS. Finally, Section 6 outlines our conclusions and future work.

2. Background

2.1. Notation

All vectors are written using bold lowercase letters. All vectors are column vectors by default. When we need a row vector, then we shall explicitly transpose a column vector. The Euclidean norm of a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ is written as $\|\boldsymbol{x}\|$ and is the nonnegative number $\|\boldsymbol{x}\|$ given by $\|\boldsymbol{x}\|^2 = \sum_{j=1}^{n} x_j^2$. All matrices are written using bold uppercase letters. If the function $\boldsymbol{f} = (f_1, f_2, \ldots, f_n)^T : \mathbb{R}^n \to \mathbb{R}^n$ is differentiable, then the Jacobian $\boldsymbol{F}(\boldsymbol{x})$ of $\boldsymbol{f}$ at the point $\boldsymbol{x} \in \mathbb{R}^n$ is the matrix $\boldsymbol{A} = [a_{ij}] \in \mathbb{R}^{n\times n}$ given by

$$a_{ij} = \frac{\partial f_i}{\partial x_j}(\boldsymbol{x}). \tag{1}$$

We shall now define the notation used to describe a system of m atoms. Let $m_i > 0$ denote mass of the ith atom and let $\boldsymbol{m}_i \in \mathbb{R}^3$ and the diagonal mass matrix $\boldsymbol{M} \in \mathbb{R}^{3m\times 3m}$ be given by

$$\boldsymbol{m}_i = (m_i, m_i, m_i)^T, \quad \boldsymbol{M} = \operatorname{diag}(\boldsymbol{m}_1^T, \boldsymbol{m}_2^T, \ldots, \boldsymbol{m}_n^T)\,. \tag{2}$$

In addition, let $\boldsymbol{q}_i, \boldsymbol{v}_i, \boldsymbol{f}_i \in \mathbb{R}^3$ denote the position of, the velocity of, and the force acting on the ith atom, and let $\boldsymbol{q} \in \mathbb{R}^{3m}$, $\boldsymbol{v} \in \mathbb{R}^{3m}$, and $\boldsymbol{f} \in \mathbb{R}^{3m}$ be given by

$$\begin{aligned} &\boldsymbol{q} = (\boldsymbol{q}_1^T, \boldsymbol{q}_2^T, \ldots, \boldsymbol{q}_m^T)^T,\ \boldsymbol{v} = (\boldsymbol{v}_1^T, \boldsymbol{v}_2^T, \ldots, \boldsymbol{v}_m^T)^T, \\ &\boldsymbol{f} = (\boldsymbol{f}_1^T, \boldsymbol{f}_2^T, \ldots, \boldsymbol{f}_m^T)^T. \end{aligned} \tag{3}$$

2.2. Fundamentals of constrained molecular dynamics

By Newton's second law, the equations of motion of a system of m atoms are

$$\boldsymbol{q}'(t) = \boldsymbol{v}(t), \tag{4}$$

$$\boldsymbol{M}\boldsymbol{v}'(t) = \boldsymbol{f}(t)\,, \tag{5}$$

where the prime indicates differentiation with respect to the time t. In nature, the motion of atoms is continuous in time; however, a computer simulation customarily uses a sequence of discrete time steps. The standard algorithm for this problem is the velocity Verlet-algorithm [18]. It is well-known that certain motions such as bond stretching, bond bending, and torsional vibrations are all periodic with characteristic frequencies that depend on the atoms involved [19]. It is generally accepted that in order to accurately resolve periodic motion one needs at least five time steps per period. Hence the fastest vibration imposes a limitation on the maximum time step that can be used and this limits the length of the time interval one can afford to simulate. In order to simulate phenomena with a longer duration it is customary to constrain the fastest degrees of freedom. Let n denote the number of constraints. Mathematically, the problem consists of solving the following system of differential algebraic equations

$$\begin{aligned} &\boldsymbol{q}'(t) = \boldsymbol{v}(t)\,, \\ &\boldsymbol{M}\boldsymbol{v}'(t) = \boldsymbol{f}(t) - \boldsymbol{G}(\boldsymbol{q}(t))^T\boldsymbol{\lambda}(t)\,, \\ &\boldsymbol{g}(\boldsymbol{q}(t)) = \boldsymbol{0}\,, \end{aligned} \tag{6}$$

with respect to $\boldsymbol{q}$, $\boldsymbol{v}$, and $\boldsymbol{\lambda}$. Here $\boldsymbol{g} : \mathbb{R}^{3m} \to \mathbb{R}^n$ is the constraint function, i.e.,

$$\boldsymbol{g} = (g_1, g_2, \ldots, g_n)^T, \tag{7}$$

where $g_i : \mathbb{R}^{3m} \to \mathbb{R}$ is the ith constraint function and $\boldsymbol{G}(\boldsymbol{q}) \in \mathbb{R}^{n\times 3m}$ is the Jacobian of $\boldsymbol{g}$ at the point $\boldsymbol{q}$. The vector $-\boldsymbol{G}(\boldsymbol{q}(t))^T\boldsymbol{\lambda}(t)$ is the *constraint force*.

2.3. Constrained MD solvers

Numerous algorithms for constrained molecular dynamics have been proposed [20–29]. Their main objective has been the reduction of the time-to-solution of the constraint equations. The most popular algorithms are SHAKE [30] and (P-)LINCS [15]. The SHAKE algorithm solves the system of differential algebraic equations (6) using a pair of staggered uniform grids with *fixed* time step $h > 0$. SHAKE's equations take the form:

$$\boldsymbol{v}_{k+1/2} = \boldsymbol{v}_{k-1/2} + h\boldsymbol{M}^{-1}\left(\boldsymbol{f}(\boldsymbol{q}_k) - \boldsymbol{G}(\boldsymbol{q}_k)^T\boldsymbol{\lambda}_k\right), \tag{8}$$

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + h\boldsymbol{v}_{k+1/2}, \tag{9}$$

$$\boldsymbol{g}(\boldsymbol{q}_{k+1}) = \boldsymbol{0}. \tag{10}$$

Here $\boldsymbol{q}_k \approx \boldsymbol{q}(t_k)$ and $\boldsymbol{v}_{k+\frac{1}{2}} \approx \boldsymbol{v}(t_{k+\frac{1}{2}})$, where $t_k = kh$ and $t_{k+\frac{1}{2}} = (k+1/2)h$. Equation (10) is a nonlinear equation for the unknown Lagrange multiplier $\boldsymbol{\lambda}_k$, namely

$$\boldsymbol{g}(\boldsymbol{\phi}_k(\boldsymbol{\lambda})) = \boldsymbol{0}, \tag{11}$$

where $\boldsymbol{\phi}_k$ is the function given by

$$\boldsymbol{\phi}_k(\boldsymbol{\lambda}) = \boldsymbol{q}_k + h(\boldsymbol{v}_{k-\frac{1}{2}} + h\boldsymbol{M}^{-1}(\boldsymbol{f}(\boldsymbol{q}_k) - \boldsymbol{G}(\boldsymbol{q}_k)^T\boldsymbol{\lambda})). \tag{12}$$

It is known that SHAKE is second order accurate in the time step [18]. The original SHAKE algorithm solved the constraint equations using the nonlinear Gauss-Seidel method, which converges locally and linearly subject to certain mild conditions (see [31] and the references therein). The LINCS and P-LINCS algorithms use a truncated Neumann-series to approximate the solution of the relevant linear systems. The Neumann-series converges linearly at best and there are physically relevant cases for which it does not converge at all [14,15,32]. Therefore, solving the constraints to the limit of machine precision is currently a time-consuming process.

Many authors have already sought to apply Newton's method for solving nonlinear equations in the context of constrained molecular dynamics. M-SHAKE [26] treats the linear systems as dense and solves them using Gaussian elimination with partial pivoting. This approach is limited to small molecules because the time complexity for computing an LU factorization of a dense matrix of dimension n is $O(n^3)$. MILC-SHAKE [23] computes an LU factorization of a tridiagonal matrix rather than a fully dense matrix. MILC-SHAKE and the related algorithm MILCH-SHAKE [29] achieve a low, linearly-scaling time complexity, but are applicable only to linear chains and n-alkanes, respectively.

The authors of the papers [27,33] all approximate the relevant matrices using matrices that are symmetric positive semi-definite and apply the conjugate gradient (CG) algorithm to solve these systems. The main advantages of this approach are twofold: the simplicity of the parallelisation of the CG algorithm, and the potential for accelerating the process using a preconditioner. The disadvantage of this approach is the difficulty of finding a preconditioner whose quality can be guaranteed mathematically.

We shall now describe how Newton's method can be applied in the context of molecular dynamics with constraints. We begin by stating the method in the case of a general nonlinear equation. Let $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ be a differential function and consider the problem of solving

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0} \tag{13}$$

with respect to $\boldsymbol{x} \in \mathbb{R}^n$. If the Jacobian $\boldsymbol{F}$ of $\boldsymbol{f}$ is nonsingular, then Newton's method is defined and takes the form

$$\boldsymbol{F}(\boldsymbol{x}_l)\boldsymbol{z}_l = \boldsymbol{f}(\boldsymbol{x}_l), \tag{14}$$

$$\boldsymbol{x}_{l+1} = \boldsymbol{x}_l - \boldsymbol{z}_l, \tag{15}$$

where the initial value $\boldsymbol{x}_0$ must be chosen by the user. In general, we expect that Newton's method will converge locally to a zero of $\boldsymbol{f}$ and that the convergence will be quadratic.

We now return to the nonlinear constraint equation (10). To this end, we introduce the matrix function $\boldsymbol{A} : \mathbb{R}^{3m} \times \mathbb{R}^{3m} \to \mathbb{R}^{n\times n}$ given by

$$\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{y}) = -h^2\boldsymbol{G}(\boldsymbol{x})\boldsymbol{M}^{-1}\boldsymbol{G}(\boldsymbol{y})^T. \tag{16}$$

Then Newton's method for the Lagrange multiplier $\boldsymbol{\lambda}_k$ is given by

$$\boldsymbol{A}(\boldsymbol{\phi}_k(\boldsymbol{\lambda}_{k,l}), \boldsymbol{q}_k)\boldsymbol{z}_{k,l} = \boldsymbol{g}(\boldsymbol{\phi}_k(\boldsymbol{\lambda}_{k,l})) \tag{17}$$

$$\boldsymbol{\lambda}_{k,l+1} = \boldsymbol{\lambda}_{k,l} - \boldsymbol{z}_{k,l}, \tag{18}$$

where the initial value $\boldsymbol{\lambda}_{k,0}$ must be chosen by the user. The simple choice of $\boldsymbol{\lambda}_{k,0} = \boldsymbol{0}$ is the de-facto standard choice.

2.4. Bond constraints

We now limit the discussion to general bond-length constraints. Note that constraints on bond angles are commonly enforced by constraining distances between two atoms. Our objective is to present a formula for the entries of the matrix $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{y})$. Let the ith bond have length $\sigma_i > 0$ and let a_i and b_i denote the indices of the two bonded atoms. Then the ith constraint can be written as

$$g_i(\boldsymbol{q}) = 0 \tag{19}$$

where

$$g_i(\boldsymbol{q}) = \frac{1}{2}\left(\sigma_i^2 - \|\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i}\|^2\right). \tag{20}$$

A direct calculation establishes that

$$\frac{\partial g_i}{\partial \boldsymbol{q}_c} = (\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i})\left(\delta_{b_i,c} - \delta_{a_i,c}\right). \tag{21}$$

Here δ_i is Kronecker's delta, i.e.,

$$\delta_{ij} = \begin{cases} 1 & i = j, \\ 0 & i \neq j. \end{cases} \tag{22}$$

In particular, we observe that

$$\frac{\partial g_i}{\partial \boldsymbol{q}_c} = \boldsymbol{0}, \quad c \notin \{a_i, b_i\}. \tag{23}$$

Now let i and j denote the indices of two bonds. There are exactly 3 distinct possibilities:

1. The two bonds have no atoms in common.
2. The two bonds share a single atom.
3. The two bonds are identical, i.e., $i = j$.

Let bond i link atoms a_i and b_i and let bond j link atoms a_j and b_j. The (i, j)th entry of the matrix $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{y})$ is given by the weighted inner-product

$$a_{ij} = (\boldsymbol{x}_{a_i}^T - \boldsymbol{x}_{b_i}^T)(\boldsymbol{y}_{a_j} - \boldsymbol{y}_{b_j})w_{ij}, \tag{24}$$

where the weight w_{ij} is given by

$$w_{ij} = \begin{cases} 0 & \{a_i, b_i\} \cap \{a_j, b_j\} = \emptyset, \\ \frac{1}{m_c}(\delta_{a_i,a_j} + \delta_{b_i,b_j} - \delta_{a_i,b_j} - \delta_{b_i,a_j}) & \{a_i, b_i\} \cap \{a_j, b_j\} = \{c\}, \\ \frac{1}{m_{a_i}} + \frac{1}{m_{b_i}} & \{a_i, b_i\} \cap \{a_j, b_j\} = \{a_i, b_i\}. \end{cases} \tag{25}$$

When the matrix $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{y})$ represents bond constraints for a real molecule, then it is necessarily quite sparse. Consider the row corresponding to the bond between a pair of atoms with valence r_1 and r_2. For this row of the matrix $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{y})$, there can be at most $r_1 + r_2 - 1$ nonzero entries, regardless of the overall size of the molecule.

The bond lengths σ_i are normally constant[3] during MD simulations and their values are set by the force fields used.

2.5. The importance of solving constraints accurately

Presently, MD simulations usually accept a large relative error when solving the constraint equations. The default value of GROMACS [35] for the maximum allowed relative error for satisfying any constraint with the SHAKE algorithm (`shake_tol`) is $\tau = 10^{-4}$. Other MD packages, like Amber [36], are a bit more demanding ($\tau = 10^{-5}$) [37]. Notwithstanding, tighter satisfaction of the constraints is sometimes imposed, frequently in simulations performed in the NVE (microcanonical) ensemble [38–42,37,43–47,32], though also in simulations with a thermostat [32,48–53] (NVT, NPT). For example, in references [32,38–42,37,43–53] the authors set a tolerance τ (`shake_tol`) for the constraints between 10^{-7} and 10^{-10} (except Ref. [40], which uses $\tau = 10^{-12}$).

It is important to appreciate the consequences of solving the constraint equations inaccurately. Let $\hat{\boldsymbol{\lambda}}_k$ denote the *computed* value of the Lagrange multiplier $\boldsymbol{\lambda}_k$; then the *computed* value of $\boldsymbol{v}_{k+1/2}$ cannot be more accurate than

$$\hat{\boldsymbol{v}}_{k+1/2} = \boldsymbol{v}_{k-1/2} + h\boldsymbol{M}^{-1}\left(\boldsymbol{f}(\boldsymbol{q}_k) - \boldsymbol{G}(\boldsymbol{q}_k)^T\hat{\boldsymbol{\lambda}}_k\right), \tag{26}$$

in which case the exact value $\boldsymbol{v}_{k+1/2}$ satisfies

$$\boldsymbol{v}_{k+1/2} = \hat{\boldsymbol{v}}_{k+1/2} - h\boldsymbol{M}^{-1}\boldsymbol{G}(\boldsymbol{q}_k)^T(\boldsymbol{\lambda}_k - \hat{\boldsymbol{\lambda}}_k). \tag{27}$$

We see that the term $-\boldsymbol{G}(\boldsymbol{q}_k)^T(\boldsymbol{\lambda}_k - \hat{\boldsymbol{\lambda}}_k)$ is mathematically equivalent to an external force. There is no reason to expect that this force is conservative. Therefore, if the objective is to simulate an *isolated* system, then it is critical that we solve the constraint equations as accurately as possible. In fact, it is well-known that if the constraint equations are not solved accurately then the total mechanical energy of the system will not be conserved. We are left with two options. We either attempt to prove that any conclusions drawn from the simulation are valid or we reduce the issue as much as possible by solving the constraint equations as accurately as the hardware will allow. Our ultimate goal is to reduce the computational burden associated with the second choice to the point where it becomes the default choice.

3. ILVES-PC: ILVES for peptide chains

3.1. Fundamentals

We have developed and implemented an algorithm called *ILVES* [54], [55] that avoids coarse-grain approximations and calculates the constraint forces accurately in an efficient manner. ILVES solves the same system of differential algebraic equations as SHAKE, the difference with SHAKE is how it solves the nonlinear constraint equations: ILVES uses Newton's method to this end. The involved linear (*linearised*) systems (17) are solved using a direct solver (Gauss-Jordan elimination). In the context of biological polymers, the time complexity of this procedure is $O(n)$, where n is the number of constraints. This linear scaling is due to linear topology of bio-polymers. The $\boldsymbol{A}$ matrix for bio-polymers can be defined so that it is either banded or nearly banded, with a few nonzero entries outside the band, which enables a special efficiency when solving the system [56]. ILVES-PC relies on a code (*compiled code* [55]) which is specifically designed to be efficient for known structures, such as the amino acid residues that make up peptides and proteins.

3.2. Implementation

The implementation of the ILVES algorithm presented in this document, called ILVES-PC, is specifically designed to compute the constraint forces for proteins. Developing code for given types of molecules is the extension to software of a concept which has already proven to be very successful with hardware. The Anton supercomputers were conceived to perform MD simulations of proteins and other biological molecules [9]. Their specific design greatly enhances the performance of these systems compared with general-purpose computers. The algorithm we present in this article also utilises specific features of widely simulated systems in order to increase the performance compared with general-purpose algorithms.

Peptide chains (peptides and proteins) are polymers of variable length formed by repeating blocks of atoms. They are a subset of biological polymers (also including e.g. nucleic acids and polysaccharides), which are just a subset of chemical polymers (which could benefit from the approach presented in this article). Each of the building blocks of a peptide chain (*residues*) has a given connectivity pattern, which is found essentially unchanged in biological molecules (occasionally further atoms can be attached to the protein, e.g. in glycoproteins, or the protein structure can get modified, e.g. at the chromophore of the Green Fluorescent Protein; in addition, hydrogen atoms in carbon rings of side chains can lie in alternative positions). However, 20 given –*proteinogenic*– amino acid connectivities largely dominate the structure of peptides and proteins.

We can make an abstraction of a peptide chain as a graph $G = (V, E)$. The vertices $V = \{1, 2, 3 \ldots, n\}$ represent the atoms and the edges $E \subseteq V \times V$ represent the bonds. Specifically, we have $(a, b) \in E$ if and only if atoms a and b are bonded. The graph is undirected because $(a, b) \in E$ if and only if $(b, a) \in E$. From G we can build a new graph $L(G)$ where each vertex represents a bond and two bonds are connected if and only if they share an atom. The graph $L(G)$ is known as the line-graph of G.

The coefficient matrix of the linear system solved by ILVES-PC, i.e., the matrix $\boldsymbol{A}$ of equation (17), has the same structure as the adjacency matrix of the line graph $L(G)$ of the graph G that represents the peptide chain. Since the peptide chain is composed of less than 30 different building blocks, the main features of the matrix $\boldsymbol{A}$ can be described using less than 30 different submatrices. These matrices correspond to the proteinogenic amino acids, some of them having slightly different configurations due to different locations of hydrogen atoms. In truth, we need a few more subroutines to account for, say, the beginning and the end of the chain. We have written subroutines for solving the linear subsystems corresponding to each of these submatrices. To solve the entire linear system, we iterate over the subsystems of the linear system, calling the required subroutine. We ensure that submatrices corresponding to identical amino acids have the same structure by always

[3] Algorithms that impose *flexible constraints* and allow, say, the bond lengths to vary over time, exist [34], but to date are less frequently used.

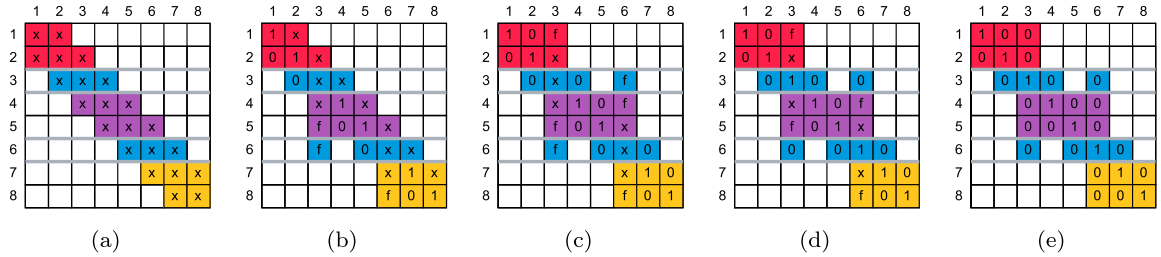


Fig. 1. Steps to apply Gauss-Jordan elimination to a banded matrix in parallel using three threads via the Schur complement method. The entries of the matrix are represented with and x and the fill-ins with an f. The magenta, purple and yellow entries are private to threads, while blue entries are shared between threads. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

numbering the bonds of each amino acid in the same order. This allows us to generate loop-free subroutines that store the relevant data contiguously in memory and do direct memory access instead of relying on the auxiliary data structures and the indirect memory access patterns that are so typical of direct solvers for sparse linear systems. These ideas are all further adaptations of the compiled code approach utilised in [55]. By choosing the bond numbering of each amino acid we can reduce the fill-in during the factorization of the matrix. Fill-ins are entries that are exactly zero in the original matrix, but become non-zero during the factorization process. We can work with peptide chains with any bond numbering. Before simulating a new protein for the very first time, we first explore the topology to identify the individual amino acid residues. Then, we renumber the bonds to match the numbering required by our specific implementation. The structural information can be saved and recycled if further simulations are required.

ILVES-PC exploits modern processors' computational resources by assigning a subset of amino acids of a peptide chain to different hardware threads. We ensure that each thread is assigned a similar number of matrix elements. To concurrently apply Gauss-Jordan elimination to the submatrices corresponding to different amino acids, we rely on the Schur complement method [57].

Consider the example presented in Fig. 1. We want to make the elimination at the banded coordinate matrix of (a) –which exemplifies $\boldsymbol{A}$– using three threads (magenta, purple, and yellow). First (b), each of the threads works with its own *private* submatrix, trying to fill the lower left-hand (subdiagonal) corner with zeros and the diagonal with ones. The threads also update the shared (blue) rows using mutual exclusion (mutex) locks. They repeat this step (c) in the upper-hand corner of their submatrix. These two steps may produce fill-ins. Then, the master thread processes the submatrix comprised of the shared (blue) rows (d), while the other two threads wait until this step is completed. Finally (e), all threads work in parallel to clean the fill-ins produced in steps (b) and (c). In the case of ILVES-PC, the thread private data corresponds to constraints within a given amino acid, while the shared rows correspond to the constraint that connects two amino acids (which corresponds to a peptide bond).

While a general-purpose implementation of ILVES would almost certainly rely mainly on coarse-grain synchronization mechanisms such as thread barriers, the precise knowledge of the structure of the linear system corresponding to proteins allows us to use very fine-grain synchronization mechanisms. We use lightweight mutex locks to protect the shared rows of the linear system from data races, so that each mutex involves only a pair of threads. Similarly, during the update phase at the end of each Newton step, the positions and velocities of each atom are protected by individual mutex locks.

4. Materials and methods

We have integrated ILVES-PC into the popular GROMACS molecular simulation package [35]. Our solver can be used as an alternative to SHAKE and P-LINCS when solving the bond constraints of proteins (only consisting of amino acid residues) without disulfide bonds.[4] In addition, we have extended the code of P-LINCS to accept a tolerance $\tau > 0$ for the satisfaction constraints on all bonds (all-bonds), as in SHAKE and ILVES-PC, so that the three solvers can be compared on an equal footing. All the tested algorithms iterate until

$$\forall i \in \{1, 2, \ldots, n\} \; : \; \frac{1}{2} \left| \frac{\|\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i}\|^2 - \sigma_i^2}{\sigma_i^2} \right| < \tau, \tag{28}$$

where the ith bond is between atoms a_i and b_i. It is straightforward to verify that

$$\frac{1}{2} \left(\frac{\|\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i}\|^2 - \sigma_i^2}{\sigma_i^2} \right) \approx \frac{\|\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i}\| - \sigma_i}{\sigma_i} \tag{29}$$

is a good approximation when the ith constraint equation is almost satisfied, i.e., when $\|\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i}\| \approx \sigma_i$ is a good approximation. It follows that τ is a good approximation of an upper bound for the largest relative error for the bond lengths.

As already mentioned, P-LINCS uses a truncated Neumann series to approximate the solution; then P-LINCS applies an iterative correction phase. The accuracy of the expansion is determined by its order (`lincs_order`), and the accuracy of the correction phase is determined by the number of iterations (`lincs_iter`). Both the order of the expansion and the number of iterations of the correction phase are set at GROMACS' startup and do not change throughout a given simulation. With our modification, P-LINCS keeps iterating in the correcting phase until all constraints are solved with the specified tolerance (`shake_tol`). We found that the additional calculations due to this modification (i.e. the calculations to check the degree of constraint satisfaction) typically increase P-LINCS' execution time by 4% to 9%.

4.1. Experimental setup

For the experiments carried out in this work we have used our modified version of GROMACS 2020.1 in double precision (`-DGMX_DOUBLE=on`) compiled using GCC-10.1.0. All simulations were performed on a computer with two Intel Xeon Platinum 8160 CPUs. Each processor has 24 physical cores. The main characteristics of our computer are presented in Table 1. We have used a GROMACS OpenMP version for multi-thread executions.

4.2. Simulations

Four proteins have been simulated in this study, namely, ubiquitin, barnase, COVID-19 main protease, and human SSU processome. A summary of their features can be found in Table 2. The

[4] Copies of the code –under development– are available on request by writing to the authors.

Table 1
Main features of our computer.

Processor	2 × Intel Xeon Platinum 8160
Cores	2 × 24
AVX-512 FMA units	2 (per core)
L1 cache (I, D)	8-way 32 KiB (per core)
L2 cache	16-way 1024 KiB (per core)
LLC	11-way 33 MiB (shared)
Main Memory	96 GiB DDR4 (12 × 8 GB 2667 MHz DIMM), 6 channels
OS	SUSE Linux Enterprise Server 12 SP2, kernel 4.4.120-92.70-default

Table 2
Proteins simulated in this article. The number of amino acid residues, atoms and constraints (all-bonds), the PDB codes and articles of reference are included.

Name	# resid.	# atoms	# constr.	Code	Reference
Ubiquitin	76	1231	1237	1UBQ	[58]
Barnase	110	1700	1721	1A2P	[59]
COVID-19 main protease	304	4645	4981	5R7Y	[60]
Human SSU processome	2722	40802	41209	7MQA bundle 3 chain F	[61]

atomic coordinates of these proteins were taken from the RCSB Protein Data Bank (www.rcsb.org, see the PDB codes in Table 2). Ubiquitin and barnase were used to study the connection between the physics of the simulation (energy and temperature) and the tolerance of the constraint solver. Ubiquitin, COVID-19, and SSU processome were used to evaluate and compare the performance of SHAKE, P-LINCS, and ILVES-PC, covering a wide range of protein sizes: 76 to 2722 amino acid residues —SSU processome is unusually large and was mainly chosen for testing parallel scalability.

Our modified version of GROMACS was used to run and analyse the simulations, which were set up with the CHARMM36 force field including CGenFF version 4.1 (last update on March 28, 2019) [62]. The ionizable residues of selected proteins were protonated in all the cases as default (pH 7) in GROMACS, and after adding explicitly TIP3P water molecules [63], chloride or sodium counterions were added to neutralize the systems. A truncated dodecahedral was chosen as the simulation box, and the minimum distance between the protein and the box edge was set to 1 nm. Periodic boundary conditions (PBC) were imposed. A minimization phase of the solvated systems was performed (maximum of 20000 steps) with the steepest descent algorithm [64]. One (for the evaluation of performance with ubiquitin, COVID-19 main protease and human SSU processome) or three (for the evaluation of physics with ubiquitin and barnase) simulation replicas were launched under each setting, enabling the averaging of results in the latter case. Systems were gradually heated through a heating ladder that allowed to increase the temperature tier by tier in a number of NVT steps (50 K every 50 ns; 1 fs time step) until reaching the target temperature (either 298 or 400 K[5]). Three consecutive steps for system equilibration followed. The first one (NVT) ran for 100 ps with restraints imposed on the heavy atoms (protein and water) and the V-rescale thermostat [65] used to keep the targeted temperature (coupling strength parameter $\tau_T = 0.1$ ps). The second equilibration step (NPT) ran for 100 ps (2 fs time step) without any restraint, with the V-rescale thermostat ($\tau_T = 0.1$ ps) [65] and the Berendsen barostat used to set the pressure to 1 atm (coupling strength parameter $\tau_p = 2.0$ ps) [66]. The third equilibration step (NPT) ran for 200 ps (2 fs time step) using the V-rescale thermostat ($\tau_T = 0.1$ ps) and the Parrinello-Rahman barostat [67] (1 atm; $\tau_p = 2.0$ ps). The configuration reached after these steps was the starting point for different production runs carried out in either the NPT or NVE thermodynamic ensembles.

For calculations of performance (NPT) and physics (NPT or NVE), the thermostat, barostat and the rest of the parameters were set up in the production phase as done in the third equilibration step, except for the NVE simulations (used only for ubiquitin and barnase), where neither thermostat nor barostat was used. Production runs launched for calculation of performance consisted of 50k steps (2 fs time step for ubiquitin and COVID-19 main protease) or 10k steps (2 fs time step for human SSU-processome). On the other hand, production runs for the calculation of physic consisted of half a million ($5 \cdot 10^5$) steps (2 fs time step, for a total simulation time of 10 ns). A record of the conserved energy values (in NPT simulations) or the total energy (in NVE simulations) and the temperature was obtained every 1000 steps (0.2 ps). These numbers were used to calculate the energy drift and the temperature evolution over time.

As general parameters of the simulations, the Verlet cutoff-scheme algorithm [68,69] was used for van der Waals interactions and the PME method [70] for electrostatic interactions, both with a radius cutoff of 1.2 nm, as recommended by the authors of the CHARMM36 force field [62]. A radius cutoff of 1.0 nm was set in simulations used for performance calculations. Velocities' correction due to the thermostat was done every 10 timesteps (default values of GROMACS).

For the tests performed to assess simulation performance, tolerance values (τ) of 10^{-4}, 10^{-6}, 10^{-8}, 10^{-10} and 10^{-12} have been used to constraint all bonds. The constraint algorithms tested include SHAKE, the modified version of P-LINCS and ILVES-PC. In the case of P-LINCS, also the `lincs_order` parameter has been tested and values of 4 and 8 are combined with the tolerance values listed above. In simulations carried out to analyse the physics of the system, the SHAKE algorithm was tested with constraints both on all-bonds and on bonds connecting with hydrogen atoms (H-bonds; results obtained with these constraints appear in Fig. 1 of the supplementary material). Tolerance (τ) values of $3.1423 \cdot 10^{-4}$, 10^{-4}, $3.1423 \cdot 10^{-5}$, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, and 10^{-10} have been tested.

The analysis of physics has been done by taking also into account the Verlet buffer size for the pair-list neighbouring search. The pair-list neighbouring search is considered one of the two most important sources of energy drift in MD simulations (the other one being that related to the constraints algorithm). The GROMACS parameter used to establish the Verlet buffer size (`Verlet-buffer-tolerance`) was set here to $5 \cdot 10^{-5}$ kJ/(mol · ps) to permit a lower level of drift than that allowed by GROMACS' default value ($5 \cdot 10^{-3}$ kJ/(mol · ps)). This way, it is possible to display the drift effect due to constraints more clearly. Data presented in Fig. 1 of the supplementary material also include results obtained for a `Verlet-buffer-tolerance` of $5 \cdot 10^{-3}$ kJ/(mol · ps).

[5] Simulations at high temperatures of biomolecules, e.g. proteins and DNA, are not infrequent. For instance, studies by molecular dynamics of protein folding and stability often increase the temperature over the protein mid-denaturation point to speedup the conformational exploration of the energy landscape.

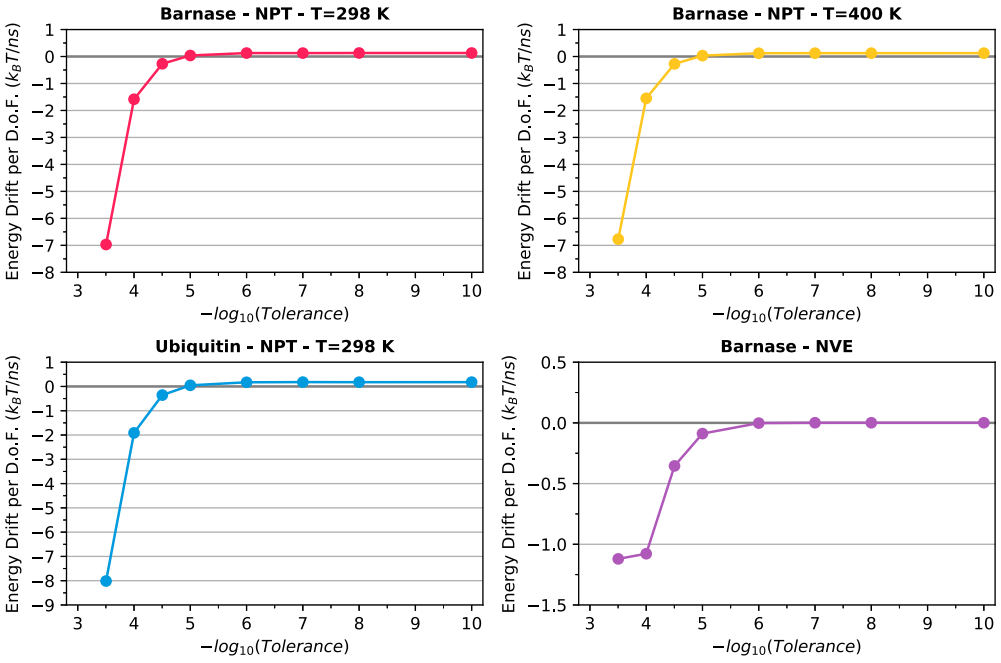


Fig. 2. Drifts of the conserved energy of the thermostat (NPT ensemble) and of the energy (NVE ensemble) per degree of freedom as a function of the tolerance in satisfying the constraints. Top: barnase in the NPT ensemble at T=298 K (left) and T=400 K (right); Bottom: ubiquitin in the NPT ensemble at T=298 K (left) and barnase in the NVE ensemble (right).

5. Results and discussion

In this section, the results of the test calculations are summarised. In the first subsection, an analysis of the reliability of the simulation as a function of the accuracy in satisfying constraints is presented, whereas an analysis of the efficiency (performance) of the calculations is displayed in the second subsection.

5.1. Physics

One of our goals is to understand the connection between the energy drift in MD simulations of proteins and the tolerance τ of the constraint algorithm (see equation (28) for the definition of τ). For this purpose, we have simulated ubiquitin and barnase in the NVE and NPT ensembles (with a V-rescale thermostat in the latter), using SHAKE to impose constraints. We choose SHAKE because the relatively *slow* and steady convergence of this algorithm ensures that the largest relative error for the bond lengths is essentially equal to the tolerance τ of the constraint algorithm.

Conversely, since ILVES-PC uses Newton's method the convergence is likely to be quadratic and there is no guarantee that the iteration will terminate with a relative error that is just *slightly* lower than the tolerance. In particular, if the current approximation has a relative error in the k-th iteration equal to e.g. $\text{Æ}_k = 1.1 \cdot 10^{-6}$ then it will not satisfy the accuracy goal represented by, say, tolerance $\tau = 10^{-6}$, but the next approximation is expected to have a relative error is significantly smaller than τ simply because $\text{Æ}_{k+1} = \mathcal{O}(\text{Æ}_k^2) \simeq 10^{-12}$. Therefore, SHAKE is preferable when the goal is to study the impact of the constraint tolerance on the conservation of energy.

In the calculations performed for our analysis of the effect of the accuracy the time evolution of the energy followed regular straight lines in large enough time scales. We thus calculated the *drift* as the slope of the energy-vs-time relationship [16] divided by the number of degrees of freedom of the system N_{df}. We defined the latter as: $N_{idf} = 3m - n - 6$, where m is the number of atoms in the protein, n is the number of imposed constraints, and the -6 term accounts for rotations and translations. In the NPT simulations the temperature was set to 298 and to 400 K (this latter only for barnase), whereas the pressure was set to 1 atmosphere. To convert Joule per mol to units of k_BT we have divided by 2477.7 if T=298 K (it includes the NVE simulations, where no T is imposed during the production stage, but 298 K was initially set), and divided by 3325.8 if T=400 K. The time step used for all the simulations was 2 fs.

The results of the drift calculations are presented in Fig. 2.[6] It is observed that the energy drift decays rapidly as the tolerance τ is reduced. The results presented in Fig. 2 are consistent with the literature, where $\tau = 10^{-7}$ is used to reduce the energy drift [48, 46,47] to acceptable levels. Fig. 2 suggests that one should choose a constraint tolerance τ which is at least as small as $\tau = 10^{-6}$.

As further evidence of the importance of using small values of τ, in Fig. 3 it is observed that higher tolerances in NVE simulations led to significant drift of the initial temperature set for the simulations. Conversely, if we use $\tau \leq 10^{-6}$ the temperature is approximately preserved over the analysed time range. Other authors have also found that the low accuracy in solving the constraints gives rise to inaccurate temperatures, and that the default parameters of LINCS lead to non-converged results [17].

The non-negligible drifts presented in Fig. 2 suggest that the mechanics of the whole molecule have distorted. Here, the dynamics has been examined on a finer level by measuring the actual bond lengths, whose value is expected to be frozen since constraints on all bonds are imposed, as a function of time. Our primary goal is to answer the questions: *Do the distortions of the bond lengths have zero average? Are the distortions of the bond lengths time-correlated?*

To delve into these questions, an analysis of bond length error (actual *vs.* expected value) has been performed for all the constrained bonds in our simulations. Plots (a) and (b) of Fig. 4 are displayed as an example and correspond to 2000 time steps (1 time step = 2 fs) from an NPT production trajectory of ubiquitin

[6] The data corresponding to this figure is presented in the supplementary material.

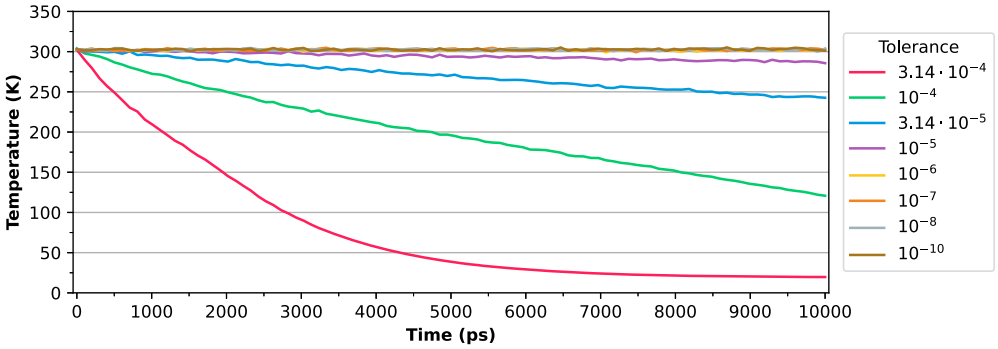


Fig. 3. Temperature evolution over time of barnase in the NVE ensemble using different tolerances in satisfying the constraints.

run with the SHAKE constraint solver with tolerances of $\tau = 10^{-4}$ and $\tau = 10^{-10}$, respectively. These plots include the *average* ($\bar{d}$), *maximum* (d_{Max}) and *minimum* (d_{min}) bond length errors over time obtained for all the constrained bonds. These parameters are defined as:

$$\bar{d}(t) := n^{-1} \sum_{i=1}^{n} d_i(\boldsymbol{q}(t)); \quad d_{\text{Max}}(t) := \max d_i(\boldsymbol{q}(t)) \qquad (30)$$
$$d_{\text{min}}(t) := \min d_i(\boldsymbol{q}(t))$$

where d_i is the relative error for the ith bond length, i.e.,

$$d_i(\boldsymbol{q}) := \frac{\|\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i}\| - \sigma_i}{\sigma_i} \qquad (31)$$

and n is the number of constraints.

The maximum and minimum values, d_{Max} and d_{min} (yellow and red lines, respectively), are approximately the established value of SHAKE's tolerance, which indicates that the solver usually does not provide solutions much more accurate than that required by the user.[7] The average $\bar{d}(t)$ (blue line) is positive along the time, which implies that the bond lengths are, on average, longer than expected. Qualitatively, the patterns of the error plots persist if the tolerance of the constraint solver is changed, e.g. to $\tau = 10^{-10}$ (plot (b) in Fig. 4). However, as expected, the size of the distortions is six orders of magnitude lower than those obtained for $\tau = 10^{-4}$ (plot (a) in Fig. 4). This element confirms that a tighter tolerance for satisfying the constraints largely reduces the bond length distortions in the simulated system.

The bond length distortions presented in the plots (a) and (b) of Fig. 4 show a Pearson correlation with their immediately previous value of 0.41 ± 0.12 (average $\pm$ SD, see Fig. 15 of the supplementary material).

Similar plots have been obtained for one of the simulations of ubiquitin run with the P-LINCS solver (`lincs_order`=4, `lincs_iter`=2, and the rest of parameters identical to those used in the simulation presented in Fig. 4) and presented in Fig. 5 of the SI File. The normalised bond length error plot presenting the largest values among all the constrained bonds is depicted in the top panel of SI Fig. 5. The average distortion (bottom panel) obtained over time for all the constrained bonds (blue line) with P-LINCS mounts to $2.17 \cdot 10^{-5}$, a lower value than that obtained for ubiquitin simulated with SHAKE ($2.67 \cdot 10^{-5}$). An interesting element one can point out from this analysis is that the bond length error profiles obtained with SHAKE present much sharper and chaotic peaks than those obtained with P-LINCS. As a result of this, a higher temporal correlation between the normalised errors and their previous value is observed with P-LINCS.

Plot (c) of Fig. 4 shows a representative normalised bond length error profile for a given constrained bond. The most important feature one can notice there is that the error is almost exclusively positive. This is the most recurrent trend in all the bonds, see Figs. 7 to 14 of the supplementary material.

5.2. Performance

We have evaluated the performance of the state-of-the-art constraint solvers (SHAKE and P-LINCS) and of the Newton method-based solver presented in this article (ILVES-PC), imposing constraints on all bond lengths. The modified version of P-LINCS (which stops iterating when the largest relative error for the bond lengths is less than the specified tolerance τ) was used, and two values of its matrix expansion parameter (`lincs_order`) were assessed: 4 (the default) and 8 (a commonly used value). Results obtained for these tests are labelled in the figures as P-LINCS-O4 and P-LINCS-O8, respectively. Other input parameters of the simulations were kept at their default values in GROMACS. Simulations were carried out for ubiquitin, COVID-19 main protease and human SSU processome in the NPT ensemble (298 K and 1 atm). The results are presented in Figs. 5, 6, 7, and 8.

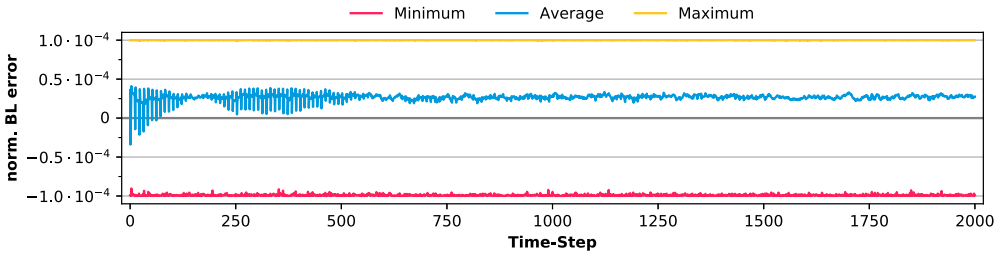
Fig. 5 presents the total (parallel, wall-clock) execution time. Stacked bars show the fraction of the time which is spent solving the constraints imposed on the protein (not on the solvent) in the production phase of the simulations and the rest of the execution time (gray background).[8] The text labels (UBIQ, COVID and SSU-PR) along the x- axis of Fig. 5 represent the three analysed proteins (ubiquitin, COVID-19 main protease and the –atypically large– human SSU processome, respectively). The numerical labels along the x-axis represent the largest acceptable relative error for a constraint in each simulation, i.e., $\tau = 10^{-4}, 10^{-6}, \ldots, 10^{-12}$.

The value $\tau = 10^{-4}$ is the GROMACS default value; we omit larger values of τ because, as presented in the previous section, they produce results that do not converge.[9] We found that $\tau = 10^{-12}$ is near the lowest relative error that can be consistently achieved during our simulations. This should not be perceived as a

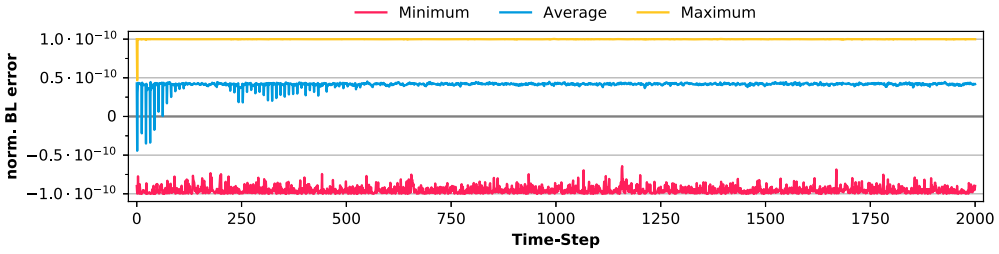
[7] This can become clearer with an example. Suppose the tolerance is $\tau = 10^{-4}$ and the current error is 10^{-3}. If we do one Newton step then we expect the new error to be 10^{-6} because of the quadratic convergence. We terminate with an error that is much smaller than the tolerance. Had we used a method with linear convergence, then the increase in accuracy would have been much smaller, so when we terminate we are not that far below the threshold.

[8] The data corresponding to this figure is presented at the supplementary material.

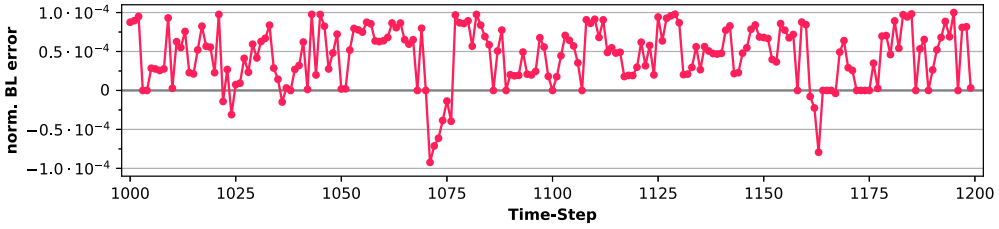
[9] Here (as well as in some other statements throughout this article) the word *converged* indicates that a tighter constraint tolerance leads to values of quantities from the simulation which strongly –in a non-negligible manner– differ from the corresponding values obtained using a loose tolerance. Hence *non-converged* does not mean that the iterative algorithm (e.g. SHAKE) was unable to find a solution to the equations.



(a) All-bonds, tolerance $= 10^{-4}$



(b) All-bonds, tolerance $= 10^{-10}$



(c) Single-bond, tolerance $= 10^{-4}$

Fig. 4. Normalised error over time for the bond lengths in MD runs of ubiquitin with SHAKE. Panels (a) and (b) show the average, maximum and minimum values of the normalised error for all the constrained bonds (all-bonds) using a tolerance of $\tau = 10^{-4}$ and $\tau = 10^{-10}$ respectively. Panel (c) shows a representative normalised error profile obtained for a constrained bond using a tolerance of $\tau = 10^{-4}$.

general result as this value depends on the specific equations being solved and the floating point number system used.

The results presented in Fig. 5 indicate that the performance of SHAKE, whose implementations are most commonly serial –in contrast to P-LINCS and ILVES-PC–, is the worst among the analysed solvers. This implies that SHAKE requires higher ratios of the total execution time, which is aggravated by increasing the number of threads. This is a natural consequence of Amdahl's Law [7], that is, the performance improvement obtained by parallel execution is limited by the sequential fraction of the application. This problem is likely to be exacerbated by the continuous increase in the number of cores in the processors. ILVES-PC performs better than the state-of-the-art in nearly all the analysed cases, and its relative advantage increases as higher accuracy is demanded. Moreover, since the computation time is similar for high accuracy (e.g. $\tau = 10^{-12}$) and low accuracy (e.g. $\tau = 10^{-4}$), accurate calculations become affordable using ILVES-PC.

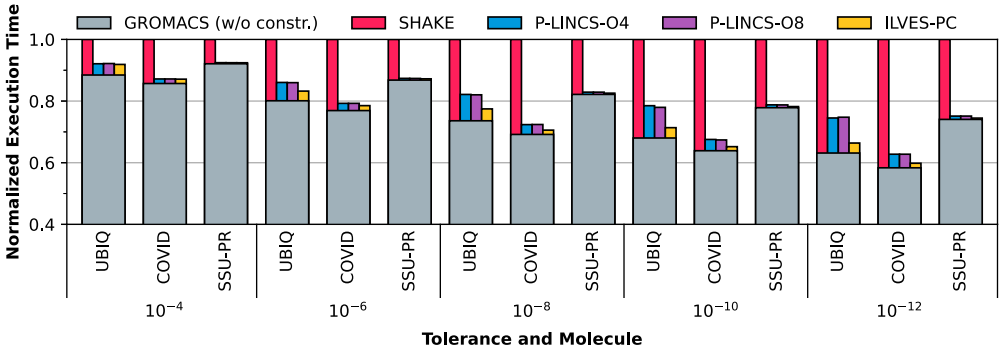
It is entirely possible to view Fig. 5 and conclude that it is a waste of time to improve the quality of parallel constraint solvers. After all, the majority of the execution time is spent outside the constraint solver, so why should we bother improving the constraint solver? This line of reasoning ignores two key points:

1. It is easy to question the conclusions drawn from inaccurate simulations that show a significant violation of the fundamental principle of conservation of energy. High accuracy is costly unless the constraint solver is parallel.
2. The efficient use of large computers requires algorithms that scale well. It is wasteful to assign 48 cores to a problem if the speedup is only 4. This precludes the use of algorithms that scale poorly.

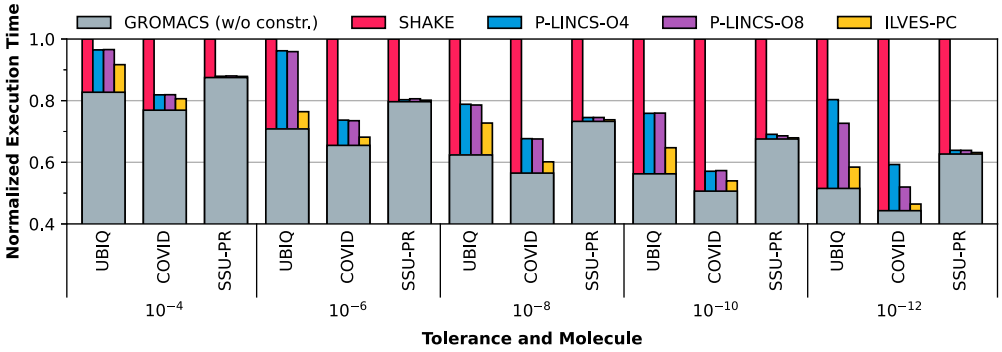
Fig. 6 shows the single-threaded speedup of P-LINCS and ILVES-PC compared with SHAKE.[10] Both P-LINCS and ILVES-PC show a speedup of approximately 1.30 over SHAKE using the GROMACS default tolerance ($\tau = 10^{-4}$). While P-LINCS preserves a speedup over SHAKE of between 1.25 and 1.5 for the whole range of tolerances, ILVES-PC's speedup with respect to both SHAKE and P-LINCS increases as the tolerance τ decreases, achieving an average speedup over SHAKE of 4.2 at the smallest value of τ.

We have also executed P-LINCS and ILVES-PC using different thread counts. Fig. 7 presents the multi-threaded speedup of P-LINCS and ILVES-PC over SHAKE for three different tolerances. The parallel speedups over SHAKE of both P-LINCS and ILVES are similar for all thread counts using GROMACS' default tolerance ($\tau = 10^{-4}$), achieving maximum speedups over SHAKE of 4×, 10×, and 34× for the three executed test cases. As we decrease the tolerance, ILVES-PC speedups over SHAKE dramatically increase while P-LINCS shows similar speedups for the whole range of tolerances. At the minimum tolerance ($\tau = 10^{-12}$), ILVES-PC achieves

[10] The data corresponding to this figure is presented in the supplementary material.



(a) 24 threads



(b) 48 threads

Fig. 5. Normalized execution times for different accuracy limits (tolerances) and three different proteins. The height of the thick grey bars represents the execution time of GROMACS excluding the constraints computations. The height of the magenta, blue, purple, and yellow bars represents the time required by the different constraint solvers. Panel (a): parallel execution with 24 threads; panel (b): ibid. with 48 threads. Note that the y-axis starts at $y = 0.4$ rather than $y = 0$. This has been done to emphasize the differences between the constraint solvers.

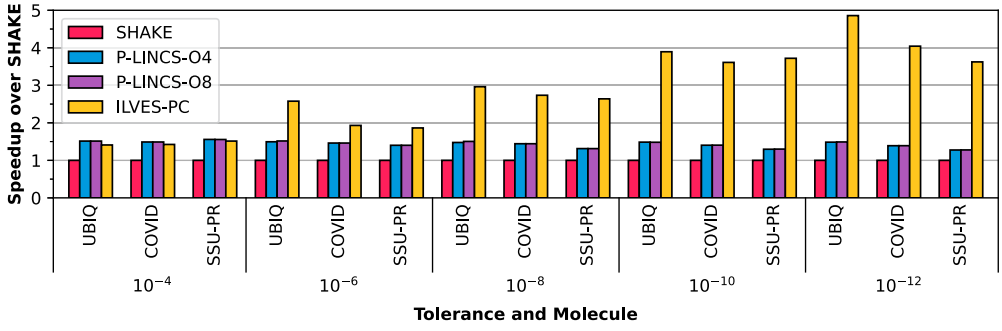


Fig. 6. Single-thread speedup over the SHAKE algorithm of ILVES-PC and P-LINCS for different tolerances and test cases.

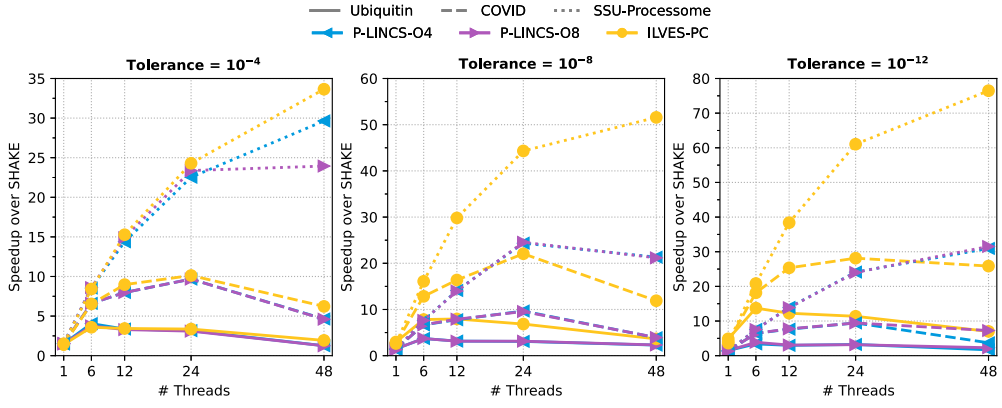


Fig. 7. Multi-threaded speedup of ILVES-PC and P-LINCS over the SHAKE algorithm for different tolerances and test cases. Note the different y-axis scale of each plot.

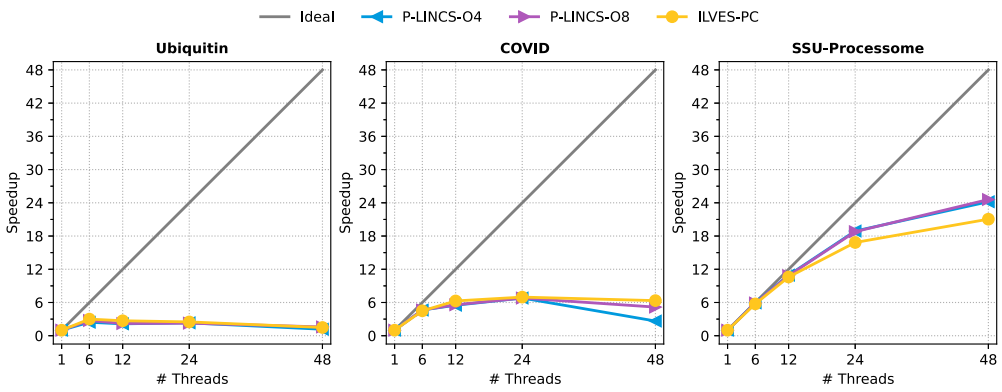


Fig. 8. Speedup over serial execution of P-LINCS and ILVES-PC for different test cases and numbers of threads.

maximum speedups over SHAKE of 13×, 28×, and 76× for the three analyzed proteins. This is not surprising. ILVES-PC is based on Newton's method which normally has quadratic convergence, while there is no reason to expect more than linear convergence from P-LINCS. Fig. 8 shows the speedup over serial execution of P-LINCS and ILVES using different numbers of threads setting the tolerance to $\tau = 10^{-12}$.[11] Nearly identical results were obtained for the whole range of tolerances. The scalability of both solvers is similar for all three test cases. ILVES-PC performs slightly better than P-LINCS for the two representative proteins. P-LINCS and ILVES-PC require regular synchronization between threads, and the size of their parallel tasks depends on the number of bonds of the molecule. Hence the size of the test case is important for scalability. If the molecule is sufficiently small and the number of cores is sufficiently large, then the parallel overhead will dominate and no solver can be efficient. Conversely, if the molecule is sufficiently large compared with the number of cores, then good parallel performance is not theoretically impossible.

6. Conclusions and future work

The constraint solver introduced in this article demonstrates that it is possible to conduct efficient parallel simulations of polymers by utilising the chemical structure. We have also presented arguments supporting that constraint equations must be solved accurately, including the fact that the default configuration of popular MD packages leads to situations where the constraint solver does not lead to converged results.

The introduced algorithm is well-suited for accurate calculations. It is faster than state-of-the-art constraint algorithms for most of the analysed cases and equally fast for all other cases. The use of Newton's method ensures that we can reach high accuracy with a small increase in computational effort compared with low-accuracy simulations.

Future stages of this project may include angular constraints and flexible constraints [34] which could make it possible to increase the time step even further. ILVES could also use inexact Newton methods, such as a symmetric approximation of the Jacobian, to achieve even higher computational efficiency. The extension of ILVES to nucleic acids (ILVES-NA), MPI parallelization, SIMD vectorization as well as a general version of ILVES which can calculate constraints in every molecule are all natural steps.

[11] The data corresponding to this figure is presented at the supplementary material.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Science and Innovation MCIN/AEI/10.13039/501100011033 (contracts PID2019-107255GB-C21, PID2019-105660RB-C21, and PID2019-107293GB-I00), by the Generalitat de Catalunya (contract 2017-SGR-1328), by the Gobierno de Aragón (E45_20R T58_20R research groups), and by Lenovo-BSC Contract-Framework Contract (2020). Santiago Marco was supported by the Agencia Estatal de Investigación (Spain) under Juan de la Cierva fellowship grant IJC2020-045916-I. Miquel Moretó was partially supported by the Agencia Estatal de Investigación (Spain) under Ramón y Cajal fellowship number RYC-2016-21104. Carl Christian Kjelgaard Mikkelsen is supported by eSSENCE, a collaborative e-Science programme funded by the Swedish Research Council within the framework of the strategic research areas designated by the Swedish Government. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

We want to thank Jesús Asín (Universidad de Zaragoza) and Benjamin Dalton (Freie Universität Berlin) for our interesting discussion.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cpc.2023.108742.

References

[1] G. Hlawacek, P. Puschnig, P. Frank, A. Winkler, C. Ambrosch-Draxl, C. Teichert, Science 321 (2008) 108–111.
[2] C. Gossens, I. Tavernelli, U. Rothlisberger, CHIMIA Int. J. Chem. 59 (2005) 81–84.
[3] E. Andreano, G. Piccini, D. Licastro, L. Casalino, N.V. Johnson, I. Paciello, S. Dal Monego, E. Pantano, N. Manganaro, A. Manenti, R. Manna, E. Casa, I. Hyseni, L. Benincasa, E. Montomoli, R.E. Amaro, J.S. McLellan, R. Rappuoli, Proc. Natl. Acad. Sci. USA 118 (2021).
[4] V.K. Bhardwaj, R. Singh, J. Sharma, V. Rajendran, R. Purohit, S. Kumar, J. Biomol. Struct. Dyn. 39 (2021) 3449–3458.
[5] B. Leimkuhler, S. Reich, Simulating Hamiltonian Dynamics, 1st ed., Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2004.

[6] M. Karplus, Abstr. Pap.—Am. Chem. Soc. 175 (1978) 70.
[7] P. García-Risueño, P.E. Ibáñez, Int. J. Mod. Phys. C 23 (2012) 1230001.
[8] E. Brini, C. Simmerling, K. Dill, Science 370 (2020) eaaz3041.
[9] D.E. Shaw, P.J. Adams, A. Azaria, J.A. Bank, B. Batson, A. Bell, M. Bergdorf, J. Bhatt, J.A. Butts, T. Correia, et al., in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2021, pp. 1–11.
[10] J. Jumper, R. Evans, A. Pritzel, T. Green, et al., Nature 596 (2021) 583–589.
[11] J.J. Galano-Frutos, H. García-Cebollada, J. Sancho, Brief. Bioinform. 22 (2019) 3–19.
[12] LAMMPS user manual, https://docs.lammps.org/fix_shake.html, 2022.
[13] NAMD user manual, https://www.ks.uiuc.edu/Research/namd/2.14/ug/node29.html, 2020.
[14] B. Hess, H. Bekker, H.J.C. Berendsen, J.G.E.M. Fraaije, J. Comput. Chem. 18 (1997) 1463–1472.
[15] B. Hess, J. Chem. Theory Comput. 4 (2008) 116–122.
[16] P. Eastman, V.S. Pande, Energy conservation as a measure of simulation accuracy, bioRxiv, 2016.
[17] S. Thallmair, M. Javanainen, B. Fábián, H. Martinez-Seara, S.J. Marrink, J. Phys. Chem. B 125 (2021) 9537–9546.
[18] E. Hairer, G. Wanner, C. Lubich, Geometric Numerical Integration, Springer, 2006.
[19] A. Mazur, J. Phys. Chem. B 102 (1998) 473.
[20] R. Edberg, D.J. Evans, G.P. Morris, J. Chem. Phys. 84 (1986) 6933–6939.
[21] H.C. Andersen, J. Comput. Phys. 52 (1983) 24–34.
[22] P. Gonnet, J.H. Walther, P. Koumoutsakos, Comput. Phys. Commun. 180 (2009) 360–364.
[23] A.G. Bailey, C.P. Lowe, A.P. Sutton, J. Comput. Phys. 227 (2008) 8949–8959.
[24] S. Miyamoto, P.A. Kollman, J. Comput. Chem. 13 (1992) 952–962.
[25] T.R. Forester, W. Smith, J. Comput. Chem. 19 (1997) 102–111.
[26] V. Krautler, W.F. Van Gunsteren, P.H. Hunenberger, J. Comput. Chem. 22 (2001) 501–508.
[27] Y. Weinbach, R. Elber, J. Comput. Phys. 209 (2005) 193–206.
[28] P. Gonnet, J. Chem. Phys. 220 (2006) 740.
[29] A.G. Bailey, C.P. Lowe, J. Comput. Chem. 30 (2009) 2485–2493.
[30] J.P. Ryckaert, G. Ciccotti, H.J.C. Berendsen, J. Comput. Phys. 23 (1977) 327–341.
[31] J.J. Moré, SIAM J. Numer. Anal. 9 (1972) 357–378.
[32] R.J. Broadbent, J.S. Spencer, A.A. Mostofi, A.P. Sutton, Mol. Phys. 112 (2014) 2672–2680.
[33] R. Elber, A. Ruymgaart, B. Hess, Eur. Phys. J. Spec. Top. 200 (2011) 211–223.
[34] B.J. Leimkuhler, S. Reich, R.D. Skeel, in: J.P. Mesirov, K. Schulten (Eds.), Mathematical Approaches to Biomolecular Structure and Dynamics, Springer, 1996.
[35] D. van der Spoel, E. Lindahl, B. Hess, G. Groenhof, A.E. Mark, H.J.C. Berendsen, J. Comput. Chem. 26 (2005) 1701–1719.
[36] D.A. Pearlman, D.A. Case, J.W. Caldwell, W.R. Ross, T.E. Cheatham III, S. DeBolt, D. Ferguson, G. Seibel, P. Kollman, Comput. Phys. Commun. 91 (1995) 1–41.
[37] Q.N. Vo, C.A. Hawkins, L.X. Dang, M. Nilsson, H.D. Nguyen, J. Phys. Chem. B 119 (2015) 1588–1597.
[38] S.-W. Chiu, M. Clark, S. Subramaniam, E. Jakobsson, J. Comput. Chem. 21 (2000) 121–131.
[39] S. Riniker, W.F. van Gunsteren, J. Chem. Phys. 134 (2011) 084110.
[40] A.P. Ruymgaart, A.E. Cardenas, R. Elber, J. Chem. Theory Comput. 7 (2011) 3072–3082.
[41] J.D. Chodera, W.C. Swope, J.W. Pitera, C. Seok, K.A. Dill, J. Chem. Theory Comput. 3 (2007) 26–41.
[42] J.D. Chodera, W.C. Swope, J.W. Pitera, K.A. Dill, Multiscale Model. Simul. 5 (2006) 1214–1226.
[43] M.T. Panteva, G.M. Giambasu, D.M. York, J. Comput. Chem. 15 (2015) 970–982.
[44] F. Chen, R. Kerr, M. Forsyth, J. Chem. Phys. 148 (2018) 193813.
[45] S. Das, V.S. Baghel, S. Roy, R. Kumar, Phys. Chem. Chem. Phys. 17 (2015) 9509–9518.
[46] D. Roest, P. Ballone, D. Bedeaux, S. Kjelstrup, J. Phys. Chem. C 121 (2017) 17827–17847.
[47] A. Ottochian, C. Ricca, F. Labat, C. Adamo, J. Mol. Model. 22 (2016) 1–8.
[48] H. Hu, F. Wanga, J. Chem. Phys. 142 (2015) 214507.
[49] M. Turner, S.T. Mutter, R.J. Deeth, J.A. Platts, PLoS ONE 13 (2018) e0193668.
[50] T.E. C. III, P. Cieplak, P.A. Kollman, J. Biomol. Struct. Dyn. 16 (1999) 845–862.
[51] J.M. Martinez, S.K.C. Elmroth, L. Kloo, J. Am. Chem. Soc. 123 (2001) 12279–12289.
[52] R. Galindo-Murillo, J.C. Robertson, M. Zgarbová, J. Šponer, M. Otyepka, P. Jurečka, T.E. Cheatham, J. Chem. Theory Comput. 12 (2016) 4114–4127, PMID: 27300587.
[53] S.P. Hancock, T. Ghane, D. Cascio, R. Rohs, R. Di Felice, R.C. Johnson, Nucleic Acids Res. 41 (2013) 6750–6760.
[54] P. García-Risueño, P. Echenique, J.L. Alonso, J. Comput. Chem. 32 (2011) 3039–3046.
[55] C.C.K. Mikkelsen, J. Alastruey-Benedé, P. Ibáñez Marín, P. García-Risueño, in: Proceedings of the 11th International Conference on Parallel Processing and Applied Mathematics (PPAM) I, 2016, pp. 160–171.
[56] P. García-Risueño, P. Echenique, J. Phys. A, Math. Theor. 45 (2012) 065204.
[57] G.H. Golub, C.F. Van Loan (Eds.), Matrix Computations, 2nd ed., The Johns Hopkins University Press, Baltimore and London, 1993.
[58] S. Vijay-Kumar, C.E. Bugg, W.J. Cook, J. Mol. Biol. 194 (1987) 531–544.
[59] C. Martin, V. Richard, M. Salem, R. Hartley, Y. Mauguen, Acta Crystallogr., D Biol. Crystallogr. 55 (1999) 386–398.
[60] A. Douangamath, D. Fearon, P. Gehrtz, T. Krojer, P. Lukacik, C.D. Owen, E. Resnick, C. Strain-Damerell, A. Aimon, et al., Nat. Commun. 11 (2020) 1–11.
[61] S. Singh, A. Vanden Broeck, L. Miller, M. Chaker-Margot, S. Klinge, Science 373 (2021) eabj5338.
[62] J. Huang, A.D. MacKerell Jr, J. Comput. Chem. 34 (2013) 2135–2145.
[63] W.L. Jorgensen, J. Chandrasekhar, J.D. Madura, R.W. Impey, M.L. Klein, J. Chem. Phys. 79 (1983) 926–935.
[64] E. Haug, J. Arora, K. Matsui, J. Optim. Theory Appl. 19 (1976) 401–424.
[65] G. Bussi, T. Zykova-Timan, M. Parrinello, J. Chem. Phys. 130 (2009) 074101.
[66] H.J. Berendsen, J. van Postma, W.F. van Gunsteren, A. DiNola, J.R. Haak, J. Chem. Phys. 81 (1984) 3684–3690.
[67] M. Parrinello, A. Rahman, J. Appl. Phys. 52 (1981) 7182–7190.
[68] L. Verlet, Phys. Rev. 159 (1967) 98–103.
[69] S. Páll, B. Hess, Comput. Phys. Commun. 184 (2013) 2641–2650.
[70] U. Essmann, L. Perera, M.L. Berkowitz, T. Darden, H. Lee, L.G. Pedersen, J. Chem. Phys. 103 (1995) 8577–8593.