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

Lorién López-Villellas[1], Carl Christian Kjelgaard Mikkelsen[2], Juan José Galano-Frutos[3], Santiago Marco-Sola[1], Jesús Alastruey-Benedé[4*], Pablo Ibáñez[4], Miquel Moretó[1,5], Javier Sancho[3], Pablo García-Risueño[6ᵽ].

1 Barcelona Supercomputing Center, Barcelona, Spain.
2 Department of Computing Science and HPC2N, Umeå University, Sweden.
3 Department of Biochemistry, Molecular and Cellular Biology, Universidad de Zaragoza, 50009 Zaragoza, Spain / Biocomputation and Complex Systems Physics Institute (BIFI), Universidad de Zaragoza, 50018 Zaragoza, Spain.
4 Departamento de Informática e Ingeniería de Sistemas - Aragón Institute for Engineering Research (I3A), Universidad de Zaragoza, Zaragoza, Spain.
5 Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
6 Independent scholar.

* jalastru@unizar.es ᵽ risueno@unizar.es

Abstract

In molecular dynamics simulations we can often increase the time step by imposing constraints on internal degrees of freedom, such as 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. In this article we analyse the impact of the accuracy of the constraint solver. We present ILVES-PC, an algorithm for imposing constraints on proteins accurately and efficiently.

ILVES-PC solves the same system of differential algebraic equations as the celebrated SHAKE algorithm, but uses Newton's method for solving the nonlinear constraint equations. It solves the necessary linear systems of equations using a specialised linear solver that utilises the molecular structure. ILVES-PC can rapidly solve the nonlinear constraint equations to nearly the limit of machine precision. This eliminates the spurious forces introduced to simulations through the very common use of inaccurate approximations. 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, we find that ILVES-PC is more accurate than the P-LINCS algorithm. Our work is a proof-of-concept of the utility of software designed specifically for the simulation of polymers.

Author summary

Molecular dynamics simulates the time evolution of molecular systems. It has become a tool of extraordinary importance for e.g. understanding biological processes and designing drugs and catalysts. This article presents an algorithm for computing the forces needed to impose constraints in molecular dynamics, i.e., the *constraint forces*; moreover, it analyses the effect of the accuracy of the constraint solver. Presently, it is customary to calculate the constraint forces with a relative error that that is not

tiny. This is due to the high computational cost associated with the available software. Accurate calculations are possible, but they are very time-consuming. The algorithm that we present solves this problem: it computes the constraint forces accurately and efficiently. Our work will improve the accuracy and reliability of molecular dynamics simulations beyond the present state-of-the-art. The results that we present are also a proof-of-concept that special-purpose code can increase the performance of software for the simulation of polymers. The algorithm is implemented into a popular molecular simulation package, and is now available for the research community.

1 Introduction

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 the 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 AlphaFold [10]. The availability of 3D structures of proteins provided by AlphaFold 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.

2 Background and Motivation

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 $\|x\|$ 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

$$\boldsymbol{q} = (\boldsymbol{q}_1^T, \boldsymbol{q}_2^T, \ldots, \boldsymbol{q}_m^T)^T\,; \quad \boldsymbol{v} = (\boldsymbol{v}_1^T, \boldsymbol{v}_2^T, \ldots, \boldsymbol{v}_m^T)^T\,; \quad \boldsymbol{f} = (\boldsymbol{f}_1^T, \boldsymbol{f}_2^T, \ldots, \boldsymbol{f}_m^T)^T \quad . \tag{3}$$

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 [12]. 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 [13]. 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 [14–23]. Their main objective has been the reduction of the time-to-solution of the constraint equations. The most popular algorithms are SHAKE [24] and (P-)LINCS [25]. 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

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

It is known that SHAKE is second order accurate in the time step [12]. 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 [26] 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 [25, 27, 28]. Therefore, solving the constraints to the limit of machine precision is time-consuming. Indeed, as we shall demonstrate, the time spent by SHAKE can easily exceed 50% of the total execution time in realistic simulations; other research works also indicate severe performance drops (23%) when LINCS is accurately solved [78]. Thus, it would be useful to have software that solves the constraint equations accurately and rapidly. To this end, we have developed and implemented an algorithm called *ILVES* [29, 30] that avoids coarse-grain approximations and calculates the constraint forces accurately. We expect that avoiding coarse approximations will also produce a solver that is capable of finding solutions when the atomic displacements are more abrupt, e.g., during simulations run at high temperatures or with large time steps (like those used in Brownian dynamics [31] calculations), or when constraints on bond angles are also imposed. For instance, in Ref. [28] it was shown that 7200 distinct simulations all produced matrices for which the expansion used by LINCS does not converge and SHAKE had to be used instead. However, the SHAKE algorithm is commonly described as inherently serial [32], which makes it inefficient for the –presently ubiquitous– parallel computations. Since the parallel algorithm ILVES-PC solves the exact same equations as the sequential algorithm SHAKE, we expect that ILVES-PC will be able to solve more problems than LINCS. To sum up, ILVES combines the features of efficiency, parallelism, accuracy, and reliability.

We emphasize that several authors have already applied Newton's method for solving nonlinear equations in the context of constrained molecular dynamics. M-SHAKE [20] treats the linear systems as dense and solves them using Gaussian elimination. 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 [17] utilizes the linear structure of alkanes to achieve a time complexity of $O(n)$ by computing an LU factorization of a tridiagonal matrix rather than a fully dense matrix. The authors of the papers [21, 33] all approximate the relevant matrices using a 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 solution can be accelerated 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

$$\boldsymbol{x}_{l+1} = \boldsymbol{x}_l - \boldsymbol{F}(\boldsymbol{x}_l)^{-1}\boldsymbol{f}(\boldsymbol{x}_l)\,, \tag{14}$$

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. In practice, we should never explicitly invert the matrix $\boldsymbol{F}(\boldsymbol{x}_l)$, instead we should compute the correction $\boldsymbol{F}(\boldsymbol{x}_l)^{-1}\boldsymbol{f}(\boldsymbol{x}_l)$ by *solving* the linear system

$$\boldsymbol{F}(\boldsymbol{x}_l)\boldsymbol{z}_l = \boldsymbol{f}(\boldsymbol{x}_l) \tag{15}$$

with respect to $\boldsymbol{z}_l$. We emphasize this point by restating Newton's method as the iteration

$$\boldsymbol{F}(\boldsymbol{x}_l)\boldsymbol{z}_l = \boldsymbol{f}(\boldsymbol{x}_l), \tag{16}$$

$$\boldsymbol{x}_{l+1} = \boldsymbol{x}_l - \boldsymbol{z}_l. \tag{17}$$

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

$$\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{y}) = -h^2 \boldsymbol{G}(\boldsymbol{x})\boldsymbol{M}^{-1}\boldsymbol{G}(\boldsymbol{y})^T\,. \tag{18}$$

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

$$\boldsymbol{A}(\boldsymbol{\phi}_k(\boldsymbol{\lambda}_{k,l}), \boldsymbol{q}_k)\boldsymbol{z}_{k,l} = \boldsymbol{g}(\boldsymbol{\phi}(\boldsymbol{\lambda}_{k,l})) \tag{19}$$

$$\boldsymbol{\lambda}_{k,l+1} = \boldsymbol{\lambda}_{k,l} - \boldsymbol{z}_{k,l}\,, \tag{20}$$

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.

We mention in passing that the matrix $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{y})$ is structurally symmetric and that the matrix $\boldsymbol{A}(\boldsymbol{\phi}_k(\boldsymbol{\lambda}_{k,l}), \boldsymbol{q}_k)$ is close to the symmetric matrix $\boldsymbol{A}(\boldsymbol{q}_k, \boldsymbol{q}_k)$, simply because

$$\boldsymbol{\phi}_k(\boldsymbol{\lambda}) = \boldsymbol{q}_k + O(h), \quad h \to 0, \quad h > 0. \tag{21}$$

This is the observation that was utilized by the authors of the papers [21, 33].

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. ILVES' design is expected to provide accurate solutions when imposing any kind of constraints, either on bond lengths, bond angles or dihedral angles, due to the fact that the coordinate matrix $\boldsymbol{A}$ is banded (regardless of the kind of the constrained degrees of freedom) in biological molecules. Constraining degrees of freedom other than bond lengths using flexible constraints [34] may be an appropriate method to increase the time step of the simulation.

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

$$g_i(\boldsymbol{q}) = 0 \tag{22}$$

where

$$g_i(\boldsymbol{q}) = \frac{1}{2}\left(\sigma_i^2 - \|\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i}\|^2\right)\,. \tag{23}$$

A direct calculation establishes that

$$\frac{\partial g_i}{\partial \boldsymbol{q}_c} = (\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i})\left(\delta_{b_i,c} - \delta_{a_i,c}\right)\,. \tag{24}$$

Here δ_i is Kronecker's delta, i.e,

$$\delta_{ij} = \begin{cases} 1 & i = j\,, \\ 0 & i \neq j\,. \end{cases} \tag{25}$$

In particular, we observe that

$$\frac{\partial g_i}{\partial \boldsymbol{q}_c} = \boldsymbol{0}, \quad c \notin \{a_i, b_i\}\,. \tag{26}$$

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

$$a_{ij} = (\boldsymbol{x}_{a_i}^T - \boldsymbol{x}_{b_i}^T)(\boldsymbol{y}_{a_j} - \boldsymbol{y}_{b_j})w_{ij}\,, \tag{27}$$

where the weight w_{ij} is given by

$$w_{ij} = \begin{cases} 0 & \{a_i, b_i\} \cap \{a_j, b_j\} = \emptyset, \\ \frac{1}{m_c}(\delta_{a_i,a_j} + \delta_{b_i,b_j} - \delta_{a_i,b_j} - \delta_{b_i,a_j}) & \{a_i, b_i\} \cap \{a_j, b_j\} = \{c\}\,, \\ \frac{1}{m_{a_i}} + \frac{1}{m_{b_i}} & \{a_i, b_i\} \cap \{a_j, b_j\} = \{a_i, b_i\}\,. \end{cases} \tag{28}$$

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 [35] 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 [36] 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 [37], are a bit more demanding ($\tau = 10^{-5}$) [38]. Notwithstanding, tighter satisfaction of the constraints is sometimes imposed, frequently in simulations performed in the NVE (microcanonical) ensemble [28, 38–48], though also in simulations with a thermostat [28, 49–54] (NVT, NPT). For example, in references [28, 38–54] the authors set a tolerance τ (`shake_tol`) for the constraints between 10^{-7} and 10^{-10} (except Ref. [41], which uses $\tau = 10^{-12}$).

Imposing constraints introduces a source of drift in the energy of the analysed system [40, 41], with the size of the drift increasing with the errors in the satisfaction of constraints. One of the principal reasons for performing accurate constraint calculations is to reduce this drift. This is especially important in simulations in the NVE ensemble, where the conservation of the energy is an explicit requisite. Accurate constraints may be also imposed to accurately calculate sought quantities [55, 56] or to avoid undesired effects –like spurious phase transitions [39]–; authors have also reported that improving simulation accuracy eliminated the flying ice cube effect [52].

When conducting simulations using a thermostat, say, in NVT or NPT ensembles, it is not customary to solve the constraint equations accurately, but this is not necessarily the correct approach. Certainly, the energy is not conserved for canonical or NPT ensembles, but there exists a *conserved quantity* (sometimes called *conserved energy*) which is analogous to the conserved energy of the microcanonical (NVE) ensemble. The derivation of the equations of the thermostat (e.g. Nosé-Hoover [57, 58], V-rescale [59]) implies that the conserved quantity is indeed conserved; otherwise, the thermostat equations that are assumed to hold do indeed not hold. There is no reason to believe that the conserved quantity will be conserved if the constraint equations are not solved to the limit of machine precision.

A small drift of the energy does not necessarily indicate that the MD simulation is accurate and reliable. Since different sources of inaccuracy can generate drifts with different signs, large errors in the dynamics can be masked by small drifts resulting from contributions that almost cancel each other. Generally speaking, a large drift of the energy is likely indicating that the dynamics is distorted and the simulation is not reliable, but the converse is not true (a small drift does not necessarily imply that the dynamics is not distorted) [60, 61]. An accurate MD should try to reduce all the sources of errors in the dynamics, hence solve constraints to the maximum affordable accuracy. Otherwise, the calculated quantities will not be reliable. Moreover, small drifts may be misleading, hinting that the simulation is accurate, when it is not.

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

$$\hat{\boldsymbol{v}}_{k+1/2} = \boldsymbol{v}_{k-1/2} + h\boldsymbol{M}^{-1}\left(\boldsymbol{f}(\boldsymbol{q}_k) - \boldsymbol{G}(\boldsymbol{q}_k)^T\hat{\boldsymbol{\lambda}}_k\right), \tag{29}$$

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

$$\boldsymbol{v}_{k+1/2} = \hat{\boldsymbol{v}}_{k+1/2} - h\boldsymbol{M}^{-1}\boldsymbol{G}(\boldsymbol{q}_k)^T(\boldsymbol{\lambda}_k - \hat{\boldsymbol{\lambda}}_k). \tag{30}$$

From the equations above we conclude that errors in the calculation of the Lagrange multipliers $\boldsymbol{\lambda}_n$ are mathematical equivalent to the actions of an external force, i.e, the term $-\boldsymbol{G}(\boldsymbol{q}_k)^T(\boldsymbol{\lambda}_k - \hat{\boldsymbol{\lambda}}_k)$. Due to its random nature, this force does not need to be a conservative force. Hence, if the objective is to simulate an *isolated* system, it is critical that we solve the constraint equations accurately.

The distortions of the bond lengths that arise from solving of the constraint equations inaccurately have non-zero average, i.e., the noise is not white, and are highly correlated with their previous values (see results and discussion below and in the S1 File). This is equivalent to using bond lengths which differ from the ones specified by the force fields (whose calibration is highly optimised to accurately describe chemical phenomena), and to using different bond lengths at different times during the simulation. Due to the accumulation of errors throughout many time steps and to the chaotic nature of simulations, the introduced spurious force may distort the dynamics in unpredicted manners, thus reducing the reliability of the calculated quantities. In contrast, if constraints are satisfied allowing a maximum relative error of e.g. 10^{-10} instead of the customary 10^{-4}, then the scale of the distortions drops by a factor of 10^6, see equation (30) and the surrounding paragraph. This significantly reduces the expected impact of spurious forces.

The points stated above indicate that an accurate satisfaction of the constraints is necessary for an appropriate simulation, where the ensemble is respected and the disruptions of the system's dynamics are reduced.

3 ILVES-PC: ILVES for peptide chains

3.1 Fundamentals

The only difference between ILVES and SHAKE is how we solve the nonlinear constraint equations; ILVES solves the same system of differential algebraic equations as SHAKE. ILVES uses Newton's method to solve the nonlinear equations; the involved linear (*linearised*) systems (19) are solved using a direct solver (Gauss-Jordan elimination), which has linear ($\mathcal{O}(n)$, being n the number of constraints) scaling due to the sparsity of matrix $\boldsymbol{A}$. Such sparsity arises from the fact that one atom cannot be covalently bonded to many others. This guarantees the sparsity of $\boldsymbol{A}$ for biological molecules, and

hence makes the ILVES algorithm suitable for constraining all kinds of internal degrees of freedom, including bond and dihedral angles. The $\boldsymbol{A}$ matrix of relevant bio-polymers can be defined so that it is banded (or nearly banded, with a few nonzero entries outside the band), which enables a special efficiency when solving the system [62]. ILVES-PC relies on a code (*compiled code* [30]) 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 their performance for 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 atoms a and b are bonded. The graph is undirected because $(a, b) \in E$ if and only $(b, a) \in E$. 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 (19), has the same structure as the adjacency matrix of the bond-graph of 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 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 [30]. By choosing the bond numbering of each amino acid we can reduce the fill-in during the factorization of the matrix. Fill-in are entries that are exactly zero in the original matrix, but become non-zero during the factorization process. We can

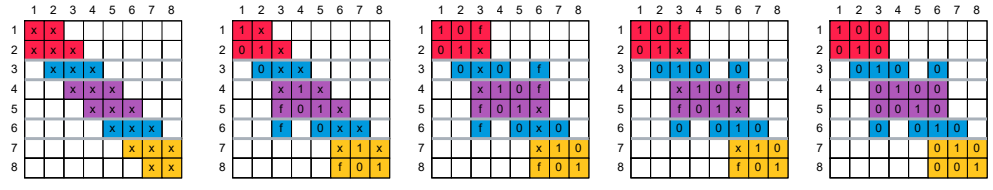

Fig 1. Steps to apply Gauss-Jordan elimination to a banded matrix in parallel using three threads via the Schur complement method. The entries of the matrix are represented with and x and the fill-ins with an f. The magenta, purple and yellow entries are private to threads, while blue entries are shared between threads.

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 [63].

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 [36]. 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 [64]. 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

$$\forall i \in \{1, 2, \ldots, n\} \; : \; \frac{1}{2} \left| \frac{\|\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i}\|^2 - \sigma_i^2}{\sigma_i^2} \right| < \tau. \tag{31}$$

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

$$\frac{1}{2} \left(\frac{\|\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i}\|^2 - \sigma_i^2}{\sigma_i^2} \right) \approx \frac{\|\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i}\| - \sigma_i}{\sigma_i} \tag{32}$$

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 processors 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.

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

4.2 Simulations

Four proteins have been simulated in this study, namely, ubiquitin, COVID-19 main protease, human SSU processome and barnase. A summary of their features can be found in Table 2. In order to assess the performance of the protein-specific constraint solver (ILVES-PC) presented in this article, simulations of the first three proteins were performed. They cover 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.

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	[65]
COVID-19 main protease	304	4645	4981	5R7Y	[66]
Human SSU processome	2722	40802	41209	7MQA bundle 3 chain F	[67]
Barnase	110	1700	1721	1A2P	[68]

The atomic coordinates of these proteins were taken from the RCSB Protein Data Bank (`www.rcsb.org`, see the PDB codes in Table 2).

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) [69]. 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 [70], 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 [71]. One (for the evaluation of performance with ubiquitin, COVID-19 main protease and human SSU processome) or three (for the evaluation of accuracy with barnase and ubiquitin) 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 [72]). 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 [59] 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) [59] and the Berendsen barostat used to set the pressure to 1 atm (coupling strength parameter $\tau_p = 2.0$ ps) [73]. 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 [74] (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 accuracy (NPT or NVE), the thermostat, barostat and the rest of 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 were used. Production runs launched for calculation of performance consisted in 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 calculation of the simulations accuracy (energy drift analysis of barnase and ubiquitin) 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 of the total energy (in NVE simulations) at every 1000 steps (0.2

ps) was obtained. These numbers were used to calculate the energy drift.

As general parameters of the simulations the Verlet cutoff-scheme algorithm [75, 76] was used for van der Waals interactions and the PME method [77] for electrostatic interactions, both with a radius cutoff of 1.2 nm as recommended by the authors of the CHARMM36 force field [69]. A radius cutoff of 1.0 nm was also set in simulations used for performance calculations. Velocities correction due to the thermostat were 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 accuracy 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 S1 File). 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 the energy drift has been done by taking also into account the Verlet buffer size for the pair-list neighboring search. The pair-list neighboring 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 $\cdot$ ps) to permit a lower level of drift than that allowed for GROMACS' default value ($5 \cdot 10^{-3}$ kJ/(mol $\cdot$ ps)). This way, it is possible to display the drift effect due to constraints more clearly. Data presented at the Fig 1 of the S1 File also include results obtained for a `Verlet-buffer-tolerance` of $5 \cdot 10^{-3}$ kJ/(mol $\cdot$ ps).

5 Results and discussion

In this section, 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 (31) 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 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. The time evolution of the energy followed regular straight lines in large enough time scales; hence we calculated the drift as the slope of the energy-vs-time relationship [60] 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.

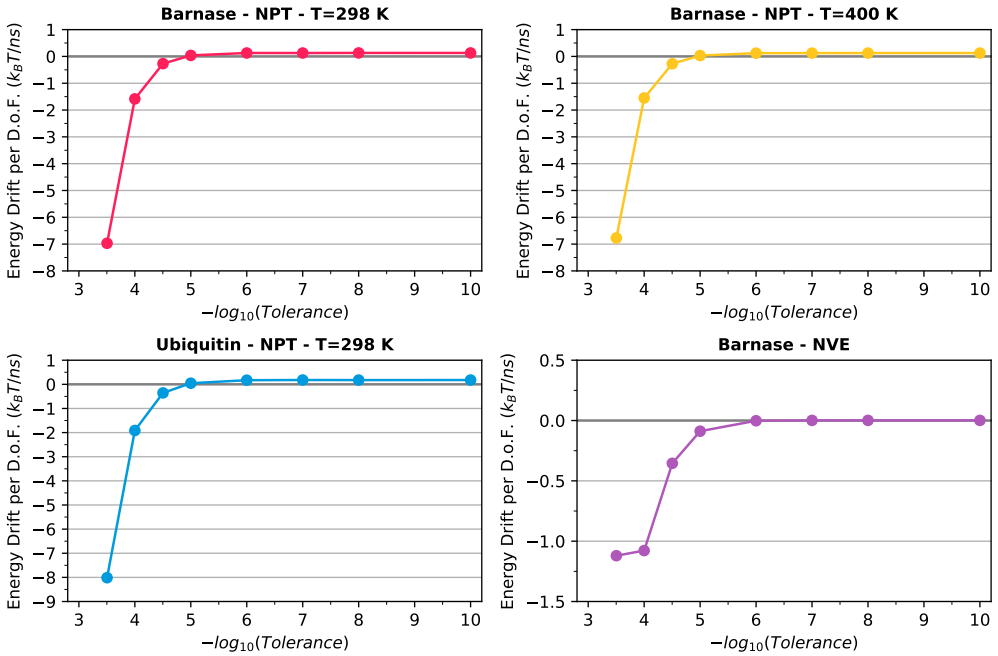

Fig 2. Drifts of the conserved energy of the thermostat (NPT ensemble) and of the energy (NVE ensemble) per degree of freedom as a function of the tolerance in satisfying the constraints. Top: barnase in the NPT ensemble at T=298 K (left) and T=400 K (right); Bottom: ubiquitin in the NPT ensemble at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

The results of the drift calculations are presented in Fig 2 [79]. 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 [47–49] to acceptable levels. Fig 2 suggests that one should choose a constraint tolerance τ which is at least as small at $\tau = 10^{-6}$.

As a 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 [78].

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 at the top and center panels 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 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:

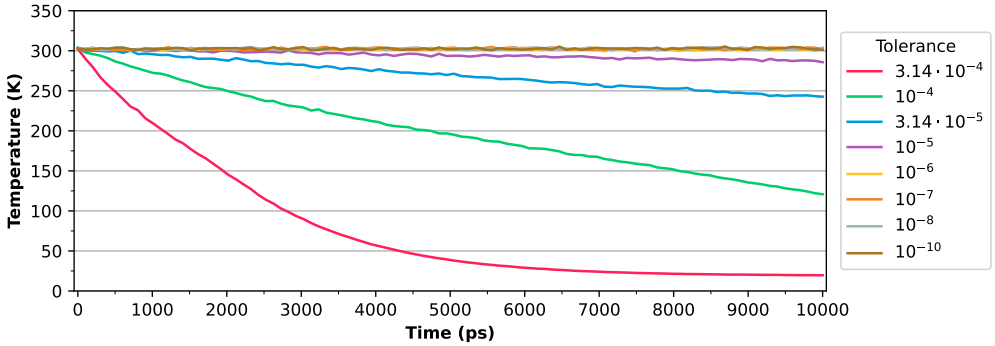

Fig 3. Temperature evolution over time of barnase in the NVE ensemble using different tolerances in satisfying the constraints.

$$\bar{d}(t) := n^{-1} \sum_{i=1}^{n} d_i(\boldsymbol{q}(t)) \,; \qquad d_{\mathrm{Max}}(t) := \max d_i(\boldsymbol{q}(t)) \,; \quad d_{\mathrm{min}}(t) := \min d_i(\boldsymbol{q}(t)) \tag{33}$$

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

$$d_i(\boldsymbol{q}) := \frac{\|\boldsymbol{q}_{a_i} - \boldsymbol{q}_{b_i}\| - \sigma_i}{\sigma_i} \tag{34}$$

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 for the tolerance of SHAKE, which indicates that the solver usually does not provide solutions much more accurate than that required for the user [80]. 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 $\tau = 10^{-10}$ (central panel in Fig 4). However, as expected, the size of the distortions is six orders of magnitude lower than those obtained for $\tau = 10^{-4}$ (top panel 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 at the top and central panels 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 S1 File).

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.

The bottom panel 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 the most recurrent trend in all the

bonds, see Figs 7 to 14 of the S1 File. Some other examples of this plot –grouped per bonds of the same nature– are presented in Figs 7 to 14 of the S1 File, and additional details thereof are pointed out there.

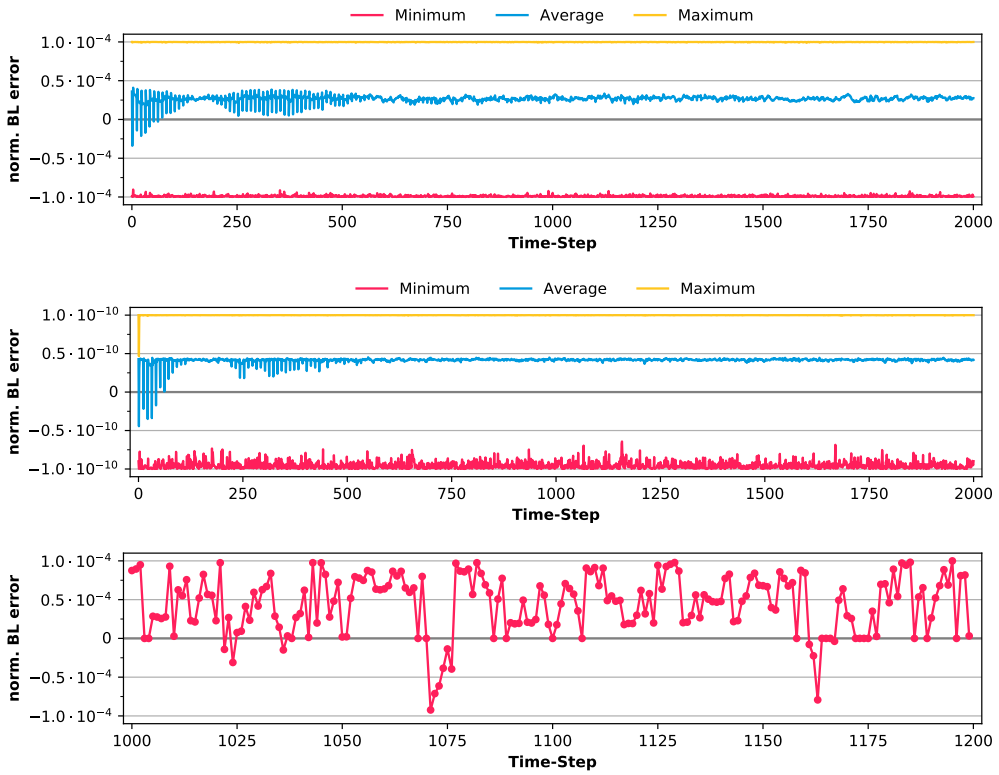

Fig 4. Normalised error over time for the bond lengths in MD runs of ubiquitin with SHAKE. The top and central panels show the average, maximum and minimum values of the normalised error for all the constrained bonds (all-bonds). The top panel corresponds to the tolerance of $\tau = 10^{-4}$ while the central one corresponds to $\tau = 10^{-10}$. The bottom panel shows a representative normalised error profile obtained for a constrained bond.

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 labeled 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) [79]. 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 simulations, i.e., $\tau = 10^{-4}, 10^{-6}, \ldots, 10^{-12}$.

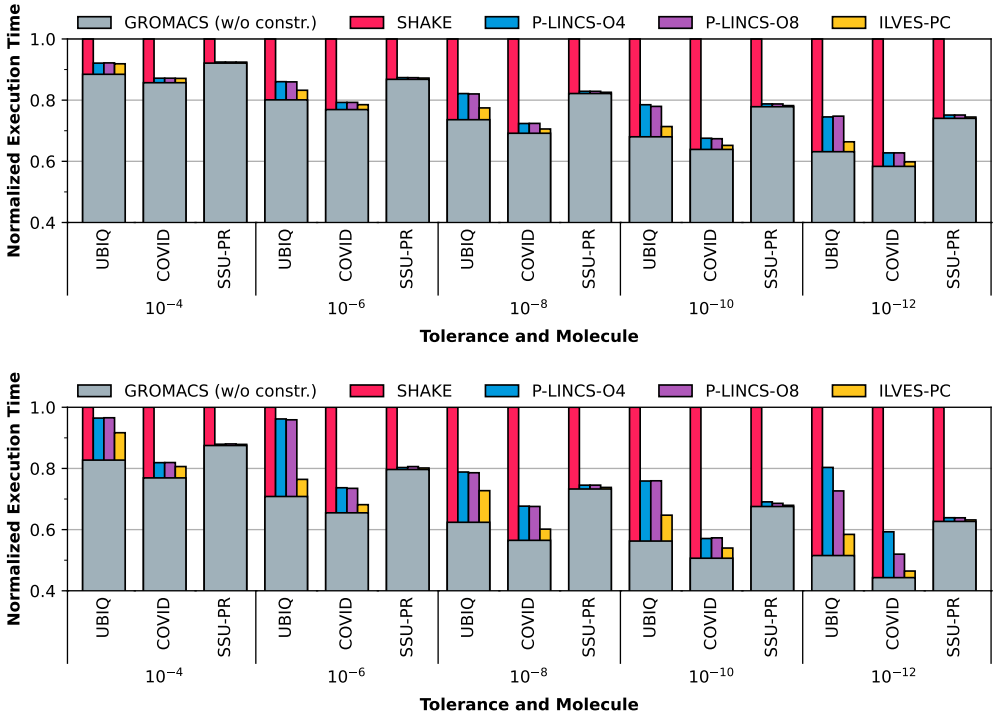

Fig 5. Normalized execution times for different accuracy limits (tolerances) and three different proteins. The height of the thick grey bars represents the execution time of GROMACS excluding the constraints computations. The height of the magenta, blue, purple, and yellow bars represent the time required by the different constraint solvers. Top: parallel execution with 24 threads; bottom: 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.

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 result that do not converge [81]. 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 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.

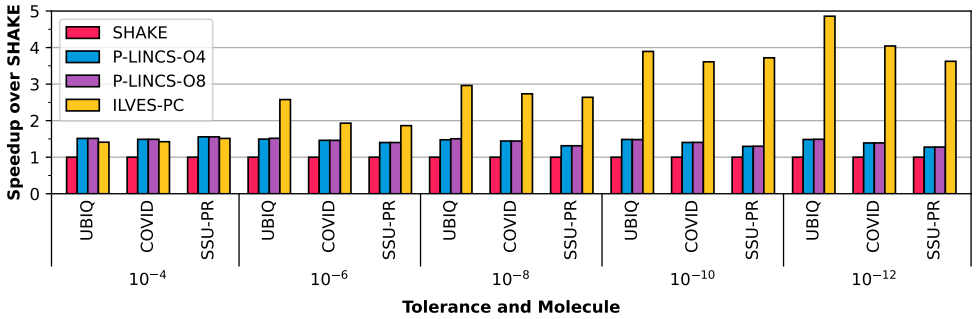

Fig 6. Single-thread speedup over the SHAKE algorithm of ILVES-PC and P-LINCS for different tolerances and test cases.

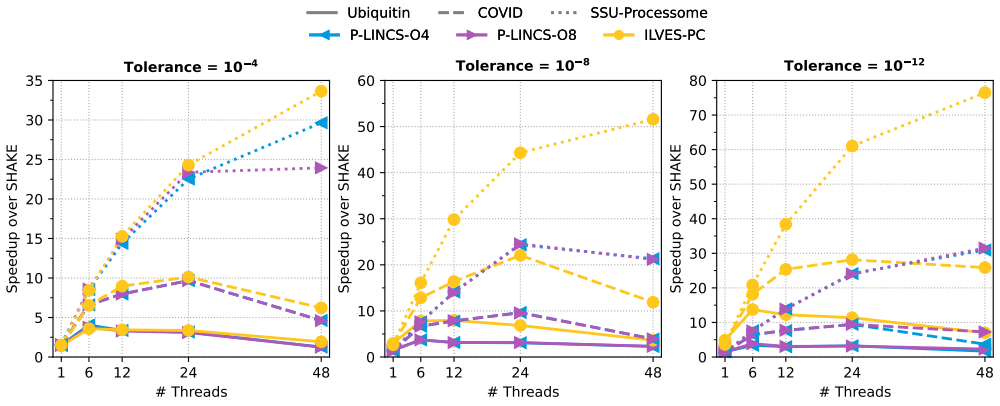

Fig 7. Multi-threaded speedup of ILVES-PC and P-LINCS over the SHAKE algorithm for different tolerances and test cases. Note the different y-axis scale of each plot.

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 require algorithms that scale well. It is wasteful to assigning 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 [79]. 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

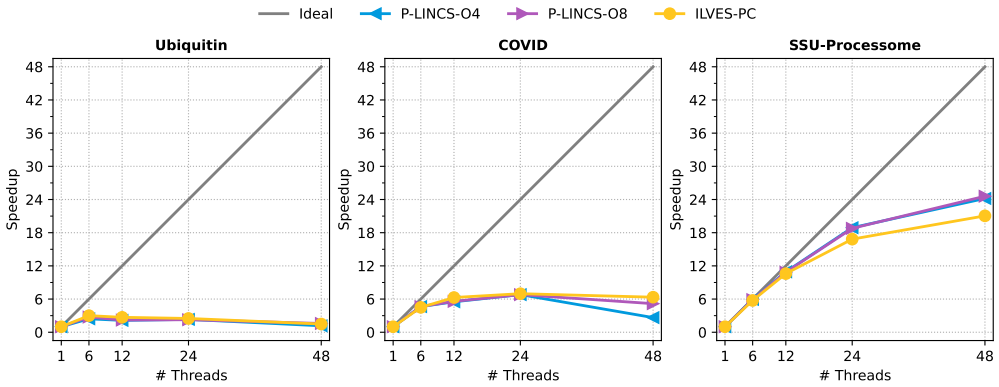

Fig 8. Speedup over serial execution of P-LINCS and ILVES-PC for different test cases and numbers of threads.

maximum speedups over SHAKE of 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 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 parallel scalability of P-LINCS and ILVES using different numbers of threads setting the tolerance to $\tau = 10^{-12}$ [79]. 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 the computational effort compared with low accuracy simulations.

Future stages of this project include tackling the case of imposing constraints on H-bonds, the use of inexact Newton methods –such as symmetric approximations of the Jacobian– in order to achieve 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.

Acknowledgments

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

References

1. Hlawacek G, Puschnig P, Frank P, Winkler A, Ambrosch-Draxl C, Teichert C. Characterization of Step-Edge Barriers in Organic Thin-Film. Science. 2008;321(5885):108–111.

2. Gossens C, Tavernelli I, Rothlisberger U. Rational Design of Organo-Ruthenium Anticancer Compounds. CHIMIA International Journal for Chemistry. 2005;59(3):81–84.

3. Andreano E, Piccini G, Licastro D, Casalino L, Johnson NV, Paciello I, et al. SARS-CoV-2 escape a highly neutralizing COVID-19 convalescent plasma. Proc Natl Acad Sci USA. 2021;118(36).

4. Bhardwaj VK, Singh R, Sharma J, Rajendran V, Purohit R, Kumar S. Identification of bioactive molecules from tea plant as SARS-CoV-2 main protease inhibitors. Journal of Biomolecular Structure and Dynamics. 2021;39(10):3449 – 3458.

5. Leimkuhler B, Reich S. Simulating Hamiltonian dynamics. 1st ed. Cambridge University Press - Cambridge Monographs on Applied and Computational Mathematics; 2004.

6. Karplus M. Dynamics of folded proteins. Abstracts of papers of the American Chemical Society. 1978;175(Mar):70.

7. García-Risueño P, Ibáñez PE. A review of high performance computing foundations for scientists. International Journal of Modern Physics C. 2012;23:1230001.

8. Brini E, Simmerling C, Dill K. Protein storytelling through physics. Science. 2020;370(6520):eaaz3041.

9. Shaw DE, Adams PJ, Azaria A, Bank JA, Batson B, Bell A, et al. Anton 3: twenty microseconds of molecular dynamics simulation before lunch. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis; 2021. p. 1–11.

10. Jumper J, Evans R, Pritzel A, Green T, et al. Highly accurate protein structure prediction with AlphaFold. Nature. 2021;596(7873):583–589.

11. Galano-Frutos JJ, García-Cebollada H, Sancho J. Molecular dynamics simulations for genetic interpretation in protein coding regions: where we are, where to go and when. Briefings in Bioinformatics. 2019;22(1):3–19. doi:10.1093/bib/bbz146.

12. Hairer E, Wanner G, Lubich C. Geometric Numerical Integration. Springer; 2006.

13. Mazur A. Hierarchy of Fast Motions in Protein Dynamics. J Phys Chem B. 1998;102:473.

14. Edberg R, Evans DJ, Morris GP. Constrained Molecular Dynamics: Simulations of liquid alkanes with a new algorithm. J Chem Phys. 1986;84:6933–6939.

15. Andersen HC. Rattle: A "velocity" version of the SHAKE algorithm for Molecular Dynamics calculations. J Comput Phys. 1983;52:24–34.

16. Gonnet P, Walther JH, Koumoutsakos P. θ-SHAKE: An extension to SHAKE for the explicit treatment of angular constraints. Comp Phys Commun. 2009;180:360–364.

17. Bailey AG, Lowe CP, Sutton AP. Efficient constraint dynamics using MILC SHAKE. J Comput Phys. 2008;227:8949–8959.

18. Miyamoto S, Kollman PA. Settle: An analytical version of the SHAKE and RATTLE algorithm for rigid water models. J Comput Chem. 1992;13(8):952–962.

19. Forester TR, Smith W. SHAKE, Rattle, and Roll: Efficient Constraint Algorithms for Linked Rigid Bodies. J Comput Chem. 1997;19:102–111.

20. Krautler V, Van Gunsteren WF, Hunenberger PH. A fast SHAKE Algorithm to Solve Distance Constraint Equations for Small Molecules in Molecular Dynamics Simulations. J Comput Chem. 2001;22:501–508.

21. Weinbach Y, Elber R. Revisiting and parallelizing SHAKE. J Comput Phys. 2005;209:193–206.

22. Gonnet P. P-SHAKE: a quadratically convergent SHAKE in O(n-squared). J Chem Phys. 2006;220:740–750.

23. Bailey AG, Lowe CP. MILCH SHAKE: An efficient method for constraint dynamics applied to alkanes. J Comput Chem. 2009;30:2485–2493.

24. Ryckaert JP, Ciccotti G, Berendsen HJC. Numerical integration of the Cartesian equations of motion of a system with constraints: Molecular dynamics of n-alkanes. J Comput Phys. 1977;23:327–341.

25. Hess B. P-LINCS: A parallel linear constraint solver for molecular simulation. J Chem Theory Comput. 2008;4:116–122.

26. Moré JJ. Nonlinear Generalizations of Matrix Diagonal Dominance with Application to Gauss–Seidel Iterations. SIAM Journal on Numerical Analysis. 1972;9(2):357–378. doi:10.1137/0709035.

27. Hess B, Bekker H, Berendsen HJC, Fraaije JGEM. LINCS: A Linear constraint solver for molecular simulations. J Comput Chem. 1997;18:1463–1472.

28. Broadbent RJ, Spencer JS, Mostofi AA, Sutton AP. Accelerated simulations of aromatic polymers: application to polyether ether ketone (PEEK). Molecular Physics. 2014;112(20):2672–2680.

29. García-Risueño P, Echenique P, Alonso JL. Exact and efficient calculation of Lagrange multipliers in constrained biological polymers: Proteins and nucleic acids as example cases. J Comput Chem. 2011;32(14):3039–3046.

30. Kjelgaard-Mikkelsen CC, Alastruey-Benedé J, Ibáñez Marín P, García-Risueño P. Accelerating sparse arithmetic in the context of Newton's method for small molecules with bond constraints. Proceedings of the 11$^{\text{th}}$ International Conference on Parallel Processing and Applied Mathematics (PPAM). 2016;I:160–171.

31. Abraham M, Berk Hess B, van der Spoel D, Lindahl E. GROMACS user manual version 5.0.4. http://www.gromacs.org/Documentation/Manual; 2014.

32. Several articles on the *SHAKE parallelization* have been published [21, 33, 82]. However, the method of solving the equations is conceptually different from the original SHAKE algorithm. Hence we consider it a different method in its own right.

33. Elber R, Ruymgaart AP, Hess B. SHAKE parallelization. Eur Phys J Spec Top. 2011;200:211–223.

34. Leimkuhler B. J., Reich, S., Skeel, R. D. Integration Methods for Molecular Dynamics. Mathematical approaches to biomolecular structure and dynamics, Springer 1996;161–185.

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

36. van der Spoel D, Lindahl E, Hess B, Groenhof G, Mark AE, Berendsen HJC. GROMACS: fast, flexible and free. J Comput Chem. 2005;26:1701–1719.

37. Pearlman DA, Case DA, Caldwell JW, Ross WR, Cheatham III TE, DeBolt S, et al. AMBER, a computer program for applying molecular mechanics, normal mode analysis, Molecular Dynamics and free energy calculations to elucidate the structures and energies of molecules. Comp Phys Commun. 1995;91:1–41.

38. Vo QN, Hawkins CA, Dang LX, Nilsson M, Nguyen HD. Computational Study of Molecular Structure and Self-Association of Tri-n-butyl Phosphates in n-Dodecane. The Journal of Physical Chemistry B. 2015;119(4):1588–1597.

39. Chiu SW, Clark M, Subramaniam S, Jakobsson E. Collective motion artifacts arising in long-duration Molecular Dynamics simulations. Journal of Computational Chemistry. 2000;21(2):121–131.

40. Riniker S, van Gunsteren WF. A simple, efficient polarizable coarse-grained water model for Molecular Dynamics simulations. The Journal of chemical physics. 2011;134(8):084110.

41. Ruymgaart AP, Cardenas AE, Elber R. MOIL-opt: Energy-Conserving Molecular Dynamics on a GPU/CPU system. J Chem Theory Comput. 2011;7(10):3072–3082.

42. Chodera JD, Swope WC, Pitera JW, Seok C, Dill KA. Use of the Weighted Histogram Analysis Method for the Analysis of Simulated and Parallel Tempering Simulations. J Chem Theory Comput. 2007;3(1):26–41.

43. Chodera JD, Swope WC, Pitera JW, Dill KA. Long-Time Protein Folding Dynamics from Short-Time MD. Multiscale Modeling & Simulation. 2006;5(4):1214–1226.

44. Panteva MT, Giambasu GM, York DM. Comparison of structural, thermodynamic, kinetic and mass transport properties of Mg(2+) ion models commonly used in biomolecular simulations. Journal of Computational Chemistry. 2015;15(36):970–82.

45. Chen F, Kerr R, Forsyth M. Cation effect on small phosphonium based ionic liquid electrolytes with high concentrations of lithium salt. The Journal of Chemical Physics. 2018;148(19):193813.

46. Das S, Baghel VS, Roy S, Kumar R. A Molecular Dynamics study of model SI clathrate hydrates: the effect of guest size and guest-water interaction on decomposition kinetics. Phys Chem Chem Phys. 2015;17:9509–9518.

47. Roest D, Ballone P, Bedeaux D, Kjelstrup S. MD simulations of metal/molten alkali carbonate interfaces. J Phys Chem C. 2017;121(33):17827–17847.

48. Ottochian A, Ricca C, Labat F, Adamo C. Molecular Dynamics simulations of a lithium/sodium carbonate mixture. Journal of molecular modeling. 2016;22(3):1–8.

49. Hu H, Wanga F. The liquid-vapor equilibria of TIP4P/2005 and BLYPSP-4F water models determined through direct simulations of the liquid-vapor interface. J Chem Phys. 2015;142(21):214507.

50. Turner M, Mutter ST, Deeth RJ, Platts JA. Ligand field Molecular Dynamics simulation of Pt (II)-phenanthroline binding to N-terminal fragment of amyloid-β peptide. PLOS One. 2018;13(3):e0193668.

51. III TEC, Cieplak P, Kollman PA. A Modified Version of the Cornell et al. Force Field with Improved Sugar Pucker Phases and Helical Repeat. Journal of Biomolecular Structure and Dynamics. 1999;16(4):845–862.

52. Martinez JM, Elmroth SKC, Kloo L. Influence of Sodium Ions on the Dynamics and Structure of Single-Stranded DNA Oligomers: A Molecular Dynamics Study. Journal of the American Chemical Society. 2001;123(49):12279–12289.

53. Galindo-Murillo R, Robertson JC, Zgarbová M, Sponer J, Otyepka M, Jurečka P, et al. Assessing the Current State of Amber Force Field Modifications for DNA. Journal of Chemical Theory and Computation. 2016;12(8):4114–4127.

54. Hancock SP, Ghane T, Cascio D, Rohs R, Di Felice R, Johnson RC. Control of DNA minor groove width and Fis protein binding by the purine 2-amino group. Nucleic acids research. 2013;41(13):6750–6760.

55. Hess B, van der Spoel D, Abraham MJ, Lindahl E. On The Importance of Accurate Algorithms for Reliable Molecular Dynamics Simulations. ChemRxiv. Cambridge: Cambridge Open Engage; 2019.

56. Gonçalves YMH, Senac C, Fuchs PFJ, Hünenberger PH, Horta BAC. Influence of the Treatment of Nonbonded Interactions on the Thermodynamic and Transport Properties of Pure Liquids Calculated Using the 2016H66 Force Field. Journal of Chemical Theory and Computation. 2019;15(3):1806–1826.

57. Nosé S. A unified formulation of the constant temperature Molecular Dynamics methods. J Chem Phys. 1984;81(1):511–519.

58. Hoover WG. Canonical dynamics: Equilibrium phase-space distributions. Physical review A. 1985;31(3):1695.

59. Bussi G, Zykova-Timan T, Parrinello M. Isothermal-isobaric Molecular Dynamics using stochastic velocity rescaling. J Chem Phys. 2009;130(7):074101.

60. Eastman P, Pande VS. Energy Conservation as a Measure of Simulation Accuracy. bioRxiv. 2016;doi:10.1101/083055.

61. *If some sources produce a positive systematic drift and others produce a negative systematic drift, they will tend to cancel out, leading to an artificially low drift on long time scales. Increasing the error can actually decrease the energy drift and make the simulation appear more accurate than it really is.* [60].

62. García-Risueño P, Echenique P. Linearly scaling direct method for accurately inverting sparse banded matrices. J Phys A: Math and Theor. 2012;45:065204.

63. Golub GH, Van Loan CF, editors. Matrix Computations. 2nd ed. Baltimore and London: The Johns Hopkins University Press; 1993.

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

65. Vijay-Kumar S, Bugg CE, Cook WJ. Structure of ubiquitin refined at 1.8 Å resolution. Journal of Molecular Biology. 1987;194(3):531–544.

66. Douangamath A, Fearon D, Gehrtz P, Krojer T, Lukacik P, Owen CD, et al. Crystallographic and electrophilic fragment screening of the SARS-CoV-2 main protease. Nat Comm. 2020;11(1):1–11.

67. Singh S, Vanden Broeck A, Miller L, Chaker-Margot M, Klinge S. Nucleolar maturation of the human small subunit processome. Science. 2021;373(6560):eabj5338.

68. Martin C, Richard V, Salem M, Hartley R, Mauguen Y. Refinement and structural analysis of Barnase at 1.5 Å resolution. Acta Crystallographica Section D: Biological Crystallography. 1999;55(2):386–398.

69. Huang J, MacKerell Jr AD. CHARMM36 all-atom additive protein force field: Validation based on comparison to NMR data. J Comp Chem. 2013;34(25):2135–2145.

70. Jorgensen WL, Chandrasekhar J, Madura JD, Impey RW, Klein ML. Comparison of simple potential functions for simulating liquid water. J Chem Phys. 1983;79(2):926–935.

71. Haug E, Arora J, Matsui K. A steepest-descent method for optimization of mechanical systems. Journal of Optimization Theory and Applications. 1976;19(3):401–424.

72. 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.

73. Berendsen HJ, van Postma J, van Gunsteren WF, DiNola A, Haak JR. Molecular dynamics with coupling to an external bath. J Chem Phys. 1984;81(8):3684–3690.

74. Parrinello M, Rahman A. Polymorphic transitions in single crystals: A new Molecular Dynamics method. Journal of Applied physics. 1981;52(12):7182–7190.

75. Verlet L. Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. Phys Rev. 1967;159:98–103.

76. Páll S, Hess B. A flexible algorithm for calculating pair interactions on SIMD architectures. Comp Phys Comm. 2013;184(12):2641–2650.

77. Essmann U, Perera L, Berkowitz ML, Darden T, Lee H, Pedersen LG. A smooth particle mesh Ewald method. J Chem Phys. 1995;103(19):8577–8593.

78. Thallmair S, Javanainen M, Fábián B, Martinez-Seara H, Marrink, SJ. Non-converged Constraints Cause Artificial Temperature Gradients in Lipid Bilayer Simulations. The Journal of Physical Chemistry B. 2021;33:9537–9546.

79. The data corresponding to this figure is presented at the S1 File.

80. 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 to 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 use a method with linear convergence, then increase in accuracy would have been much smaller, so when we terminate we are not that far below the threshold.

81. 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.

82. Ruymgaart AP, Elber R. Revisiting Molecular Dynamics on a CPU/GPU system: Water kernel and SHAKE parallelization. Journal of Chemical Theory and Computation. 2012;8(11):4624–4636.

Supporting Information

S1File. Supporting Information of *Accurate and efficient constrained molecular dynamics of polymers through Newton's method and special purpose code.*