This is the peer reviewed version of the following article:
Sieradzan A., Sans-duñó J., Lubecka E., Czaplewski C., Lipska A., Leszczyński H., Ocetkiewicz K., Proficz J., Czarnul P., Krawczyk H., Liwo A., Optimization of parallel implementation of UNRES package for coarse-grained simulations to treat large proteins, JOURNAL OF COMPUTATIONAL CHEMISTRY, Vol. 44, iss. 4 (2023), pp. 602-625,
which has been published in final form at https://doi.org/10.1002/jcc.27026. This article may be used for non-commercial purposes in accordance with Wiley Terms and Conditions for Use of Self-Archived Versions. This article may not be enhanced, enriched or otherwise transformed into a derivative work, without express permission from Wiley or by statutory rights under applicable legislation. Copyright notices must not be removed, obscured or modified. The article must be linked to Wiley's version of record on Wiley Online Library and any embedding, framing or otherwise making available the article or pages thereof by third parties from platforms, services and websites other than Wiley Online Library must be prohibited.

Optimization of parallel implementation of UNRES package for coarse-grained simulations to treat large proteins

Adam K. Sieradzan[*†], Jordi Sans-Duñó[‡], Emilia A. Lubecka, Cezary Czaplewski[*†], Agnieszka G. Lipska[*†], Henryk Leszczyński[¶‖], Krzysztof M. Ocetkiewicz[†], Jerzy Proficz[†], Paweł Czarnul, Henryk Krawczyk[†§], Adam Liwo[*†]

March 17, 2023

Abstract

We report major algorithmic improvements of the UNRES package for physics-based coarse-grained simulations of proteins. These include (i) introduction of interaction lists to optimize computations, (ii) transforming the inertia matrix to a pentadiagonal form to reduce computing and memory requirements, (iii) removing explicit angles and dihedral angles from energy expressions and recoding the most time-consuming energy/force terms to minimize the number of operations and to improve numerical stability, (iv) using OpenMP to parallelize those sections of the code for which distributed-memory parallelization involves unfavorable computing/communication time ratio, and (v) careful memory management to minimize simultaneous access of distant memory sections. The new code enables us to run molecular dynamics simulations of protein systems with size exceeding 100,000 amino-acid residues, reaching over 1 ns/day (1 μs/day in all-atom timescale) with 24 cores for proteins of this size. Parallel performance of the code and comparison of its performance with that of AMBER, GROMACS and MARTINI 3 is presented.

■ Keywords: Coarse graining, UNRES, molecular dynamics, parallel algorithms, OpenMP.

[*]Faculty of Chemistry, University of Gdańsk, Fahrenheit Union of Universities in Gdańsk, ul. Wita Stwosza 63, 80-308 Gdańsk, Poland

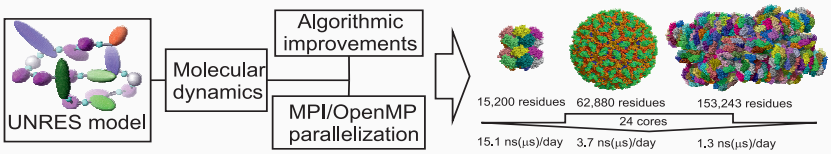
[†]Centre of Informatics Tri-city Academic Supercomputer and Network (CI TASK), Gdańsk University of Technology, Fahrenheit Union of Universities in Gdańsk, ul. G. Narutowicza 11/12, 80-233 Gdańsk, Poland

[‡]Department of Chemistry, University of Lleida, Av. Alcade Rovira Roure, 191, 25198 Lleida, Spain

[§]Faculty of Electronics, Telecommunication and Informatics, Gdańsk University of Technology, Fahrenheit Union of Universities in Gdańsk, ul. G. Narutowicza 11/12, 80-233 Gdańsk, Poland

[¶]Faculty of Mathematics, Physics and Informatics, University of Gdańsk, Fahrenheit Union of Universities in Gdańsk, Wita Stwosza 57, 80-308 Gdańsk, Poland

[‖]Deceased, September 30, 2021



Molecular dynamics with the optimized and efficiently parallelized implementation of the highly-reduced physics-based UNRES model of polypeptide chains enables us to reach, with moderate computer resources, several nanosecond/day of MD time, for protein systems with size over 100,000 residues, which translates to microsecond/day real time given a 1,000 fold faster occurrence of events in the simulations with the UNRES model compared to the all-atom representation.

MOST WIEDZY Downloaded from mostwiedzy.pl

INTRODUCTION

Molecular simulations, especially molecular dynamics (MD), are now established methodologies with which to study the structure, properties, dynamics, and transformations of biological macromolecules and assemblies, polymers, crystals, glasses, and other systems[1–5]. In biological applications, the simulations are already capable of treating viruses and cell organelles and it is very likely that their scope will soon be extended to the minimal bacterial cell[6]. Efficient algorithms for all-atom molecular dynamics and dedicated hardware, namely the ANTON supercomputer of the D.E. Shaw group[7,8] have been developed. With these resources, millisecond-scale all-atom simulations of small proteins and microsecond-scale simulations of entire viruses or cell fragments[9] are now feasible. However, all-atom simulations have not yet reached the time scale of biological events for larger systems. Consequently, coarse-grained (CG) models, in which each interaction site comprises several atoms, are a plausible means to overcome the time- and size-scale problem[10–18]. A number of coarse-grained models have been developed, including AWSEM[19], OpenAWSEM[20], CABS[10], the residue-level coarse-grained model recently implemented in the GENESIS MD software[21] by Tan and colleagues[22] (which will hereafter be referred to as the GENESIS CG model), MARTINI[23,24], SIRAH[25], and UNRES[26,27]. Of those, MARTINI is most widely used, because of is generality and possibility of automatic coarse-graining.

Switching from the all-atom to a coarse-grained representation of a system results in omitting the fast-moving degrees of freedom corresponding to uncorrelated motions involving only few atoms (e.g., bond vibrations), to preserve only correlated motions of larger parts of a system. Consequently, the time scale of coarse-grained dynamics is accelerated by several orders of magnitude with respect to that of all-atom dynamics[28–30]. Moreover, omitting the fast motions from the model makes it much easier for a researcher to extract important events from simulations. The cost of the calculation of energy and forces is also reduced with respect to all-atom representation, owing to fewer interaction sites. However, a coarser representation also implies reduced accuracy of results, which decreases with the extent of coarse graining.

As the extent of coarse graining increases, it also becomes more and more problematic

MOST WIEDZY Downloaded from mostwiedzy.pl

to keep a straightforward and easy to implement scheme, in which each extended atom is centered in a point and its distribution of mass and the interaction potentials are centrosymmetric. Introducing axially-symmetric sites is a sensible solution, which has been implemented first by Berne and Pechukas[31] to coarse-grain liquid-crystal molecules, which was further modified by Gay and Berne[32] to become the well-known family of the Gay-Berne anisotropic potentials.

The axial symmetry of interaction sites appears a natural choice when polymer chains are considered. Polymer-chain units can then be linked by virtual bonds, between which the backbone sites are located, while the "side groups" are attached to the main chain with virtual bonds. The main degrees of freedom to average over are the angles for rotation of the groups of atoms constituting extended sites about the respective virtual bonds. The solvent is considered explicitly as extended solvent atoms or superatoms comprising a number of solvent molecules (as in MARTINI[23]) or is averaged over to be absorbed in the mean-field interaction potentials.

Recently, using the axial-symmetry ansatz, and taking advantage of our earlier derivation[33] of coarse-grained effective energy functions as Kubo cluster-cumulant[34] expansions of the respective potentials of mean force, we developed a mathematical formalism for the construction of scale-consistent expressions in coarse-grained force fields[35,36]. Owing to this formalism, expressions for pairwise and multibody interaction potentials that exhibit correct dependence on site orientation and on local chain geometry can be derived[35–37]. We implemented this formalism and its earlier versions in the derivation of the effective energy expressions in the coarse-grained UNRES model of polypeptide chains developed in our laboratory[26,27]. Owing to these expressions, UNRES is capable of quite accurate modeling of the structure, dynamics, and thermodynamics of proteins, despite heavy coarse graining (only two interaction sites per amino-acid residue)[26,27]. The formalism of UNRES has been extended to nucleic acids[38] and polysaccharides[26].

Assuming the axial symmetry of the interacting polymer beads implies that the variables present in the equations of motion are not necessarily identified with the Cartesian coordinates of the site centers. In our first implementation of molecular dynamics with the UNRES model of polypeptide chains[39], we used virtual-bond vectors as variables, which

MOST WIEDZY Downloaded from mostwiedzy.pl

resulted in the presence of a full, albeit constant, inertia matrix in the equations of motion. Consequently, the memory requirements and the number of operations necessary to solve the equations of motion grew with the square of problem size. In this work, we use the anchor points of the interaction sites as variables, which results in a pentadiagonal (five-band) inertia matrix and, therefore, linear growth of memory and computing-time requirements with system size.

In biomolecular computations the non-bonded interactions between the atoms that are separated by more than the selected distance cut-off are omitted and the particle mesh Ewald (PME) scheme is used to handle electrostatic interactions[40] to speed up computations. In most CG force fields, for example in MARTINI[41], a cut-off is used on all interactions. Lists of interactions that are within the cut-off are constructed; these lists are updated only every given number of MD steps. With the interaction lists, the cost of computations grows linearly or as $n \log n$ (where n is the number of interacting sites) when the PME is used. The cut-off-based calculation schemes in coarse-grained simulations are constantly being improved[42]. We made an attempt at introducing the cut-off on long-range interactions in UNRES in our previous work[43]. However, the site-site distances were evaluated every MD step, this resulting in the still quadratic growth of the cost of computations with system size, albeit with a smaller coefficient compared to the situation in which the interaction energies of all pairs are computed. In this work, we have introduced interaction lists and carefully selected the cut-off distance to achieve the best compromise between computing cost and accuracy.

Even though coarse graining heavily reduces the computation time compared to all-atom schemes[28], parallelization of the code is a necessity for larger systems. There are currently several application programming interfaces (APIs) that can be used for parallelizing computation-intensive C/Fortran applications, mainly depending on target system type (distributed or shared memory) and computing device type (CPU or GPU)[44]. For distributed-memory parallelization the Message Passing Interface (MPI)[45] is typically used for communication among processes of a parallel application that run on various nodes. Shared-memory parallelization is accomplished by multithreading with synchronization using OpenMP[46], Pthreads[47], OpenCL[48], with multi-core CPUs and CUDA[49], OpenCL[48], OpenACC[50] on GPUs. All these means are extensively used in the codes developed for

MOST WIEDZY Downloaded from mostwiedzy.pl

molecular simulations such as AMBER[51], GROMACS[52], LAMMPS[53], DESMOND[54], and TINKER[55].

In our earlier work[56,57], we parallelized UNRES with MPI, by designing a two-grain parallelization scheme. The tasks were divided into coarse-grain and fine-grain ones. Each coarse-grain task was handling a single MD trajectory. The communication between these tasks was necessary only upon ending the job or every 10,000–20,000 MD steps when replica-exchange simulations were carried out. Each coarse-grain task was divided into fine-grain subtasks, each of those computing its assigned part of energy and forces of a single conformation. Because relatively small systems were considered at that time, no cut-off on long-range interactions was introduced and, therefore, the pairwise interactions could be partitioned between the fine-grain tasks at the beginning of a run. Very good scalability was achieved, which amounted to about 70 % parallel efficiency with 256 coarse-grain tasks, each divided into 64 fine-grain tasks (16,384 tasks total) with IBM Blue Gene, for a 600-residue protein[56,57].

Extensive code optimization is essential for the development of high-performance simulation software. It includes the minimization of the amount of costly (e.g. floating point) operations, using caches for proper data partitioning, using vectorization, proper mapping of threads onto cores (thread affinity), the minimization of false sharing in case of multiple threads accessing locations in the same cache line, proper data representation, data alignment, loop unrolling, (dynamic) load balancing, the minimization of recurrent memory allocation and data copying, and the minimization of thread creation overheads. In this work we followed the above guidelines and achieved major improvements of the UNRES software through (i) introducing interaction lists, together with computation- and memory-saving construction and management of these, (ii) transformation of the inertia matrix to a penta-diagonal (five-band) form to reduce memory requirements, (iii) removing explicit angles and dihedral angles from energy expressions and recoding the most time-consuming energy/force terms to minimize the number of operations and to improve numerical stability, (iv) the use of OpenMP to parallelize those sections of the code for which distributed-memory parallelization can additionally benefit from multithreading within computing nodes, and (v) careful memory management to minimize simultaneous access of distant memory sections.

MOST WIEDZY Downloaded from mostwiedzy.pl

It should be noted that the solutions designed here are not specific for UNRES but apply to any model of polymer chains with axially-symmetric interaction sites.

The paper is organized as follows. In the Methodology section, we describe the UNRES model of polypeptide chains, the equations of motions of the CG molecular dynamics with UNRES along with the selection of variables that leads to the most economic five-band structure of the inertia matrix, and solving the equations of motion. Next we describe the components of the energy function and forces, the computation of which was optimized in this work and the solutions applied in this optimization. Then we describe the construction and management of the interaction lists, including the selection of the optimal cut-off distance. We conclude the section with the description of the parallelization solutions applied, which involves the use of both MPI and OpenMP, as well as profiling and optimization of the code. In the Results section, we report the functional tests of the new UNRES code, memory and CPU time scaling with system size, as well as the scalability of parallel implementation. We also compare the timing of the optimized UNRES with those of the MARTINI 3[24], OpenAWSEM[20], and GENESIS[22] coarse-grained packages and the AMBER[51] and GROMACS[52] all-atom MD packages. Lastly, we address the extension of UNRES time-scale compared to that of all-atom simulations. We conclude the paper with outlining further algorithmic developments of the UNRES package and its applications.

METHODOLOGY

UNRES model of polypeptide chains

In the UNRES model[26,27,33,35,37,58], a polypeptide chain is represented by a sequence of α-carbon (C^α) atoms linked with virtual bonds, with peptide groups (p) located strictly (not just by imposing the relevant restraints) halfway between the consecutive C^αs and united side chains attached to the C^αs with the $C^\alpha \cdots SC$ virtual bonds. Only the ps and the SCs are interaction sites, while the C^αs assist in chain-geometry definition; these points will hereafter be referred to as the *anchor points*. The positions of the glycine and dummy-residue side chains coincide with those of the anchor points. The equilibrium length of the backbone

MOST WIEDZY Downloaded from mostwiedzy.pl

$C^{\alpha} \cdots C^{\alpha}$ virtual bonds equals 3.8 Å, which corresponds to the *trans* peptide group, while the equilibrium $C^{\alpha} \cdots SC$ bond lengths depend on side-chain kind[59,60]. The N- and C-terminal chain-blocking groups are represented by placing a glycine residue on the respective end, while the unblocked chain ends are represented by dummy residues, which have no mass and do not take part in interactions. Multichain systems are technically represented by single chains, with two dummy residues placed between the preceding and the succeeding chain. The model is illustrated in Figure 1. It should be noted that, although we discuss the UNRES model only in this paper, the considerations of this and the next section are also valid for any model of polymer chains, where the backbone interaction sites are located between (not necessarily halfway) anchor points and side groups are attached to them. Extension is also possible to branched polymers.

The UNRES energy function is expressed by Equation (1).

$$\begin{aligned} U &= w_{SC} \sum_{i<j} U_{SC_i SC_j} + w_{SCp} \sum_{i \neq j} U_{SC_i p_j} + w_{pp}^{VDW} \sum_{i<j-1} U_{p_i p_j}^{VDW} + w_{pp}^{el} f_2(T) \sum_{i<j-1} U_{p_i p_j}^{el} \\ &+ w_{tor} f_2(T) \sum_i U_{tor}(\gamma_i, \theta_i, \theta_{i+1}) + w_b \sum_i U_b(\theta_i) + w_{rot} \sum_i U_{rot}(\theta_i, \hat{\mathbf{r}}_{SC_i}) \\ &+ w_{bond} \sum_i U_{bond}(d_i) + w_{ssbond} \sum_i U_{ssbond}(d_i^{SS}) \\ &+ w_{corr}^{(3)} f_3(T) \sum_{i<j-1} U_{corr;ij}^{(3)} + w_{turn}^{(3)} f_3(T) \sum_i U_{turn;i}^{(3)} \end{aligned} \tag{1}$$

where the terms $U_{SC_i SC_j}$ are sidechain-sidechain interaction energies represented by the modified Gay-Berne potentials[58], $U_{SC_i p_j}$ are excluded-volume potentials that prevent the collapse of the united side chains on the backbone, $U_{p_i p_j}^{VDW}$ (with spherical symmetry) and $U_{p_i p_j}^{el}$ (with axial symmetry) are the non-bonded and mean-field-electrostatic interaction potentials of united peptide groups, U_{bond}, are the bond-deformation potentials, U_b and U_{tor} are the backbone-virtual-bond-angle and the backbone-virtual-bond-torsional potentials, respectively, θ_i and γ_i denoting the virtual-bond- and virtual-bond-dihedral angles, respectively (Figure 1), U_{rot} are the side-chain-rotamer potentials, in which $\hat{\mathbf{r}}_{SC_i}$ denotes the local coordinates of the unit vector pointing from C_i^{α} to SC_i, while $U_{corr}^{(3)}$ and $U_{turn}^{(3)}$ are multibody terms that account for the coupling of the backbone-local and backbone-electrostatic interactions[33,35]. U_{ssbond} denotes the terms that account for the energetics of disulfide bonds,

MOST WIEDZY Downloaded from mostwiedzy.pl

including their formation and breaking[61,62]. Because the energy terms have been discussed in detail in our earlier work[26,27,35,58,59] and the respective formulas consume substantial space, we do not show detailed formulas here but will bring the optimized formulas for U_b, U_{tor}, $U_{corr}^{(3)}$ and the respective gradient components in section "Removing explicit angles from energy expressions".

The factors $f_n(T)$ account for the dependence of the force-field terms that correspond to higher-order terms in the Kubo cluster-cumulant expansion on temperature[63], as given by Equation (2).

$$f_n(T) = \frac{\ln\left[\exp(1) + \exp(-1)\right]}{\ln\left\{\exp\left[(T/T_\circ)^{n-1}\right] + \exp\left[-(T/T_\circ)^{n-1}\right]\right\}} \tag{2}$$

where $T_\circ = 300$ K.

The ws are the weights of the energy terms and have been determined, along with some other parameters, by maximum-likelihood calibration of the force field[37]. The variant of UNRES used in this work has been termed NEWCT-9P[37].

The U_b, U_{tor}, $U_{corr}^{(3)}$, and $U_{turn}^{(3)}$ terms are the scale-consistent terms derived in our earlier work[35]; in particular, the torsional and correlation terms depend on both virtual-bond-dihedral angles γ and virtual-bond angles θ. Because the solvent degrees of freedom belong to the set of secondary variables that are averaged out when passing to the coarse-grained representation[33,35], the interactions involving the solvent are implicit in the effective energy function; they are mainly contained in the $U_{SC_iSC_j}$ terms[33].

As can be seen from Equation (1), all long-range energy terms, including the multibody terms, are technically pairwise. The multibody terms ($U_{corr}^{(3)}$ and $U_{turn}^{(3)}$) depend not only on the positions and orientation of the sites but also on the local geometry of the surrounding chain segment[33,35–37].

Coarse-grained molecular dynamics and its extensions with UNRES

Selection of variables

Because the UNRES interaction sites have axial and not spherical symmetry (cf. section "UNRES model of polypeptide chains"), their coordinates must comprise both the posi-

MOST WIEDZY Downloaded from mostwiedzy.pl

tion and direction. In our previous work[28,39], we used the $C^\alpha \cdots C^\alpha$ ($\mathbf{dC}_0, \mathbf{dC}_1, \mathbf{dC}_2, \ldots, \mathbf{dC}_{n-1}$) and $C^\alpha \cdots SC$ ($\mathbf{dX}_2, \mathbf{dX}_3, \ldots, \mathbf{dX}_{n-1}$) virtual-bond vectors, n being the total number of residues (including the dummy residues) in the system under study, $\mathbf{dC}_0$ denoting the position of C_1^α. For glycine and dummy groups, the corresponding $\mathbf{dX}$s were absent, because the positions of the respective side chains coincide with those of the C^α anchor points. These vectors define the interaction-site axes and the positions of site centers can easily be calculated by summing over the virtual-bond vectors[39]. However, such an approach results in a full square symmetric constant inertia matrix, the storage of which requires memory growing with the square of system size, the time to compute forces from accelerations also growing quadratically with system size. In our recent work[64] on using the force-matching method to obtain a variant of UNRES compatible with all-atom MD, we changed the variables to the Cartesian coordinates of the C^α anchor points ($\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n$) and side-chain centers ($\mathbf{X}_2, \mathbf{X}_3, \ldots, \mathbf{X}_{n-1}$). With these coordinates, the positions of the peptide-group centers ($\mathbf{p}$), those of the side-chain centers ($\mathbf{SC}$), and the virtual-bond vectors are defined by Equations (3–6)[64].

$$\mathbf{p}_i = \frac{1}{2}\left(\mathbf{C}_i + \mathbf{C}_{i+1}\right), i = 1, 2, \ldots n-1;\ i, i+1 \text{ not a dummy residue} \tag{3}$$

$$\mathbf{SC}_i = \begin{cases} \mathbf{X}_i & \text{if } i \text{ is not glycine} \\ \mathrm{C}_i & \text{otherwise} \end{cases} \tag{4}$$

$$\mathbf{dC}_i = \mathbf{C}_{i+1} - \mathbf{C}_i \tag{5}$$

$$\mathbf{dX}_i = \mathbf{X}_i - \mathbf{C}_i \tag{6}$$

We arrange the C^α-atom and side-chain-center coordinates into the $3m$-dimensional vector $\mathbf{q}$ defined by Equation (7).

$$\mathbf{q} = \begin{pmatrix} \mathbf{q}_x \\ \mathbf{q}_y \\ \mathbf{q}_z \end{pmatrix} \quad \mathbf{q}_\xi = \begin{pmatrix} \mathbf{C}_{\xi 1} \\ \mathbf{C}_{\xi 2} \\ [\mathbf{X}_{\xi 2}^T] \\ \vdots \\ \mathbf{C}_{\xi n} \end{pmatrix} \tag{7}$$

MOST WIEDZY Downloaded from mostwiedzy.pl

where $\xi = x$, y or z and $[\mathbf{X}_{\xi i}]$ denotes the coordinates of the side chain of the ith residue. The square bracket around the symbol indicates that the side-chain coordinates are equal to the C^{α} coordinates if residue i is a glycine or a dummy residue. The dimension of each sub-vector is equal to $m = n - n_{Gly} - n_D$, n_{Gly} and n_D denoting the number of glycine and dummy residues, respectively. We array all site coordinates into a single vector $\mathbf{R}$ defined by Equation (8).

$$\mathbf{R} = \begin{pmatrix} \mathbf{r}_x \\ \mathbf{r}_y \\ \mathbf{r}_z \end{pmatrix} \quad \mathbf{r}_\xi = \begin{pmatrix} [\mathbf{p}_{\xi 1}] \\ [\mathbf{SC}_{\xi 1}] \\ \mathbf{p}_{\xi 2} \\ \mathbf{SC}_{\xi 2} \\ \vdots \\ [\mathbf{p}_{\xi n}] \\ [\mathbf{SC}_{\xi n}] \end{pmatrix} \tag{8}$$

With this notation, Equations (3–6) can be written as Equation (9).

$$\mathbf{R} = \mathbf{A}\mathbf{q} \tag{9}$$

where $\mathbf{A}$ is the matrix that relates the generalized coordinates to the site coordinates. Its elements are implicitly defined by Equations (3–6). It should be noted that, for a multichain system, $\mathbf{A}$ consists of disjoint blocks, each corresponding to one chain.

Equations of motion

The Langevin equations of motions have a similar form as those derived in our earlier work[39] and are given by Equation (10). It should be noted that, because different chains are not bonded, the equations of motion for each chain are not coupled through the inertia matrix but only through the potential forces. Likewise, the equations of motion for the x, y, and z coordinates are not coupled through the inertia matrix but only through the potential forces.

$$\mathbf{G}_I \ddot{\mathbf{q}}_{I\xi} = -\nabla_{\mathbf{q}_{I\xi}} U - \mathbf{A}_I^T \mathbf{\Gamma}_I \mathbf{A} \dot{\mathbf{q}}_{I\xi} + \mathbf{A}_I^T \mathbf{F}_{I\xi}^r, \; I = 1, 2, \ldots \; N_c; \; \xi = x, y, z \tag{10}$$

MOST WIEDZY Downloaded from mostwiedzy.pl

where $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ denote the velocities and accelerations of the generalized coordinates [defined by Equation (7)] $\mathbf{G}_I$ is the inertia matrix for chain I, $\mathbf{\Gamma}_I$ is the diagonal matrix of site friction coefficients for chain I, $\mathbf{F}_I^r$ are the random forces acting on the sites of chain I, and N_c is the number of chains. Instead of explicit introduction of the friction and stochastic forces, the Berendsen[65], Nosé-Hoover[66] or Nosé-Poincaré[67] thermostats can also be used, as implemented in our earlier work[28,68] to provide thermostating.

The inertia matrix $\mathbf{G}_I$ is a pentadiagonal (five-band) matrix of the Ith chain. The diagonal, first-band off-diagonal, and second-band off-diagonal elements are defined by Equations (11–15). For clarity sake, we drop the chain index I. We denote the index of the starting anchor point of the chain by s, the indices of the C^α atoms as α and those of the side-chain centers as β.

$$G_{ss} = \frac{1}{4}m_p + I_p + m_{SC_s} \tag{11}$$

$$G_{\alpha\alpha} = \frac{1}{2}m_p + 2I_p + m_{SC_\alpha} \tag{12}$$

$$G_{\beta\beta} = m_{SC_\beta} + I_{SC_\beta} \tag{13}$$

$$G_{\alpha,\alpha+1} = G_{\alpha+1,\alpha} = \begin{cases} \frac{1}{4}m_p - I_p & \text{if residue } \alpha \text{ is Gly} \\ -I_{SC_\alpha} & \text{otherwise} \end{cases} \tag{14}$$

$$G_{\alpha,\alpha+2} = G_{\alpha+2,\alpha} = \begin{cases} 0 & \text{if residue } \alpha \text{ is Gly} \\ \frac{1}{4}m_p - I_p & \text{otherwise} \end{cases} \tag{15}$$

where m_p is the mass of the peptide group, m_{SC_α} is the mass of the side chain corresponding to index α; if a residue is glycine, m_{SC} is the mass of glycine's central methylene group, while $I_p = \frac{1}{12}m_p$, $I_{SC_i} = \frac{1}{3}m_{SC_i}$ are the normalized moments of inertia of a peptide group and a side chain, respectively, computed with the assumption that their masses are uniformly distributed along the virtual-bond axes[39].

The friction coefficients of the sites are calculated by using the Stokes' law, as in our previous work[28] [Equation (16)]

$$\gamma_i = 6\pi(r_i + r_{wat})\eta_{wat}f \tag{16}$$

MOST WIEDZY Downloaded from mostwiedzy.pl

where γ_i is the friction coefficient of site i, r_i is the effective radius of site i, r_{wat} is the effective radius of water (taken in this study as 1.4 Å), η_{wat} is the viscosity of water equal to 0.8904 cPoise at 298 K, and $f < 1$ is a scaling factor introduced to speed up the calculations; usually $f = 0.01$ is assumed.

As in our earlier work[28], the stochastic forces acting on the sites are calculated following the fluctuation-dissipation theorem, from Equation (17).

$$\mathbf{F}_i^r = \sqrt{\frac{2\gamma_i RT}{\delta t}}\mathbf{N}(0,1) \tag{17}$$

where $\mathbf{F}_i^r$ is the vector of random forces acting on site i, γ_i is the friction coefficient of this site [defined by Equation (16)], R is the universal gas constant, T is the absolute temperature, δt is the integration time step, and $\mathbf{N}(0,1)$ is the 3-dimensional vector whose components are sampled independently for a normal distribution with zero mean and unit variance. The balance between the stochastic and the friction forces provides a thermostat that maintains the average temperature at the pre-set value.

Solving the equations of motion

We use the variant of the velocity-Verlet algorithm[69] developed in our earlier work[28,39] to solve the equations of motion. To solve the equations system given by Equation (10), we implemented the Fortran 77 `fdisy` subroutine (subroutine F 4.12.2) described in Ref. 70, designed for solving linear equations systems with symmetric five-band matrices. For multichain systems, a separate system of linear equations is solved for each chain. The five-band matrix is stored in form of three vectors: the diagonal (DM), defined by Equations (11–13), the first off-diagonal (DU1), defined by Equation (14), and the second off-diagonal (DU2), defined by Equation (15). Therefore, the memory requirements grow linearly with system size. Likewise, the number of operations in solving the system of equations by `fdisy` grows linearly with chain length.

The initial structures read from the PDB or randomly generated require local energy minimization before starting MD. In this implementation, we adapted the Limited Memory Broyden Fletcher Goldfarb Shanno (LBFGS)[71] quasi Newton minimizer from the TINKER package[55]. Owing to the fact that the approximation to the inverse of the Hessian is con-

MOST WIEDZY Downloaded from mostwiedzy.pl

structed on the fly, the memory requirements of this algorithm scale linearly with the number of variables (roughly 6 times the number of residues). This feature enables us to treat large systems while preserving the Q-superlinear convergence, which is characteristic of quasi-Newton minimizers. In the previous implementations of UNRES, we used the quasi-Newton Secant Unconstrained Minimization Solver (SUMSL)[72], the memory requirements of which scale with the square of the number of variables, due to storing the whole lower triangle of the approximation to the inverse of the Hessian matrix.

As pointed out in our earlier work[39], the diagonalization of the inertia matrix is needed to set the initial velocities corresponding to a given temperature. The memory requirements of this step change from square to linear in system size when passing from virtual-bond vectors to the coordinates of the C^{α} and SC centers.

Periodic boundary conditions

To enable us to treat multichain systems we apply periodic boundary conditions introduced into UNRES in our previous work[43]. In our implementation, the periodic box is cuboidal in general, usually cubic. We apply the minimum-image convention, setting the box-side length at more than twice the cut-off distance.

Extensions of UNRES MD

To search the conformational space more extensively (e.g., in determining the conformational ensembles of proteins), we apply the replica exchange molecular dynamics (REMD)[73] and multiplexed replica exchange molecular dynamics (MREMD)[74]. In REMD, m trajectories (replicas) are run independently, each at a different temperature. Every M time steps (iterations), attempts at exchanging temperatures between the neighboring replicas are made according to the Boltzmann criterion. As a result, a trajectory stuck in a local high-energy minimum and run at a low temperature gets a higher temperature that enables this trajectory to overcome energy barriers and land in a lower-energy basin, while that having a low energy but run at a high temperature gets a lower temperature, enabling it to explore the low-energy basin to find even lower-energy conformations. MREMD is a multiplexed variant of REMD, in which multiple trajectories are run at a given temperature. Both extensions of MD have

MOST WIEDZY Downloaded from mostwiedzy.pl

been implemented in UNRES[63,75].

Removing explicit angles from energy expressions

As opposed to all-atom force fields, in which the bond-angle terms depend explicitly on bond angles and the torsional terms depend on the cosines (and, sometimes, sines) of the dihedral angles, the scale-consistent energy expressions for the UNRES bond-angle (U_b) and torsional (U_{tor}) terms contain only the trigonometric functions of the bond angles and the sines and cosines of multiples of the torsion angles multiplied by the reciprocating powers of the adjacent virtual-bond angles, respectively[35]. Especially the second feature removes the problem of indefiniteness of the torsional potentials when the virtual bonds involved become collinear, as the scale-consistent torsional potentials become zero in such a case. Likewise, the local parts of the scale-consistent correlation terms ($U_{corr}^{(3)}$ and $U_{turn}^{(3)}$) are expressed in the same way as the torsional angles, avoiding indefiniteness for collinear fragments of the chain. Consequently, in the new version of UNRES we replaced explicit angles in the UNRES energy expressions with the scalar and cross products of the virtual-bond vectors, as detailed in the subsections below. This modification improved the numerical stability of and accelerated the evaluation of energy and energy gradient.

Virtual-bond-angle terms (U_b)

The virtual-bond-valence energy terms (U_b) in Equation (1) depend only on the cosines of the angles θ (cf. Refs. 35 and 37). Hence, U_b can be generally expressed by Equation (18).

$$U_b(\theta_i) = U_b(\cos\theta_i) = U_b(-\mathbf{d\hat{C}}_{i-1} \circ \mathbf{d\hat{C}}_i) \tag{18}$$

where $\mathbf{d\hat{C}}_i = \frac{1}{||\mathbf{dC}_i||}\mathbf{dC}_i$ is the virtual-bond vector normalized to unit length and

$$\cos\theta_i = -\mathbf{d\hat{C}}_{i-1} \circ \mathbf{d\hat{C}}_i \tag{19}$$

The gradient in $\mathbf{d\hat{C}}_i$ and $\mathbf{d\hat{C}}_{i+1}$ is expressed by Equation (20).

MOST WIEDZY Downloaded from mostwiedzy.pl

$$\nabla_{\mathbf{dC}_k} U_b(\theta_i) = \frac{dU_b}{d\cos\theta_i} \nabla_{\mathbf{dC}_k} \cos\theta_i,\ k = i-1, i \tag{20}$$

$$\nabla_{\mathbf{dC}_{i-1}} \cos\theta_i = -\frac{1}{||\mathbf{dC}_{i-1}||} \left(\mathbf{d\hat{C}}_{i-1}^T + \cos\theta_i \mathbf{d\hat{C}}_i^T \right) \tag{21}$$

$$\nabla_{\mathbf{dC}_i} \cos\theta_i = -\frac{1}{||\mathbf{dC}_i||} \left(\mathbf{d\hat{C}}_i^T + \cos\theta_i \mathbf{d\hat{C}}_{i-1}^T \right) \tag{22}$$

Virtual-bond-dihedral-angle terms (U_{tor})

The scale-consistent expressions for the torsional energy involve expressions of the form given by Equations (23) and (24)[35].

$$\phi_i^{(n)} = (\sin\theta_i \sin\theta_{i+1})^n \cos n\gamma_i \tag{23}$$

$$\psi_i^{(n)} = (\sin\theta_i \sin\theta_{i+1})^n \sin n\gamma_i \tag{24}$$

In the present UNRES, only up to second-order terms are used[35,37]. Compared to the commonly applied expressions for the torsional energy, which involve $\cos n\gamma$ and $\sin n\gamma$, the expressions given by Equations (23) and (24) have the advantage that they tend to 0 as the respective fragment becomes linear, which makes the dihedral angle undefined. To compute $\phi_i^{(n)}$ and $\psi_i^{(n)}$ and their derivatives without having to evaluate $\cos n\gamma$ and $\sin n\gamma$, we developed modified Tschebyshev polynomials of the first (Θ) and the second (Υ) kind, respectively, as given by Equations (25–27) and (28–30), respectively. In the equations below, $x = \phi_i^{(1)} = \sin\theta_i \sin\theta_{i+1} \cos\gamma_i$, $y = \psi_i^{(1)} = \sin\theta_i \sin\theta_{i+1} \sin\gamma_i$, $t = \sin\theta_i \sin\theta_{i+1}$.

$$\begin{aligned} \Theta_0(x,t) &= 1 \\ \Theta_1(x,t) &= x \\ \Theta_n(x,t) &= 2\Theta_1(x,t)\Theta_{n-1}(x,t) - \Theta_{n-2}(x,t)t^2 \end{aligned} \tag{25}$$

MOST WIEDZY Downloaded from mostwiedzy.pl

$$\begin{aligned}
\frac{\partial \Theta_0}{\partial x} &= 0 \\
\frac{\partial \Theta_1}{\partial x} &= 1 \\
\frac{\partial \Theta_n}{\partial x} &= 2\Theta_{n-1} + 2\Theta_1 \frac{\partial \Theta_{n-1}}{\partial x} - \frac{\partial \Theta_{n-2}}{\partial x} t^2
\end{aligned} \tag{26}$$

$$\begin{aligned}
\frac{\partial \Theta_0}{\partial t} &= 0 \\
\frac{\partial \Theta_1}{\partial t} &= 0 \\
\frac{\partial \Theta_n}{\partial t} &= 2\Theta_1 \frac{\partial \Theta_{n-1}}{\partial t} - \frac{\partial \Theta_{n-2}}{\partial t} t^2 - 2\Theta_{n-2} t
\end{aligned} \tag{27}$$

$$\begin{aligned}
\Upsilon_0(x,t) &= 1 \\
\Upsilon_1(x,t) &= 2x \\
\Upsilon_n(x,t) &= \Upsilon_1(x,t)\Upsilon_{n-1}(x,t) - \Upsilon_{n-2}(x,t) t^2
\end{aligned} \tag{28}$$

$$\begin{aligned}
\frac{\partial \Upsilon_0}{\partial x} &= 0 \\
\frac{\partial \Upsilon_1}{\partial x} &= 2 \\
\frac{\partial \Upsilon_n}{\partial x} &= 2\Upsilon_{n-1} + \Upsilon_1 \frac{\partial \Upsilon_{n-1}}{\partial x} - \frac{\partial \Upsilon_{n-2}}{\partial x} t^2
\end{aligned} \tag{29}$$

$$\begin{aligned}
\frac{\partial \Upsilon_0}{\partial t} &= 0 \\
\frac{\partial \Upsilon_1}{\partial t} &= 0 \\
\frac{\partial \Upsilon_n}{\partial t} &= \Upsilon_1 \frac{\partial \Upsilon_{n-1}}{\partial t} - \frac{\partial \Upsilon_{n-2}}{\partial t} t^2 - 2\Upsilon_{n-2} t
\end{aligned} \tag{30}$$

x and y can be computed directly from the virtual-bond vectors, as given by Equations (31) and (32).

MOST WIEDZY Downloaded from mostwiedzy.pl

$$x = \sin\theta_i \sin\theta_{i+1} \cos\gamma_i = -\mathbf{d\hat{C}}_{i-1} \circ \mathbf{d\hat{C}}_{i+1} + \cos\theta_i \cos\theta_{i+1} \quad (31)$$

$$y = \sin\theta_i \sin\theta_{i+1} \sin\gamma_i = \left(\mathbf{d\hat{C}}_{i-1} \times \mathbf{d\hat{C}}_i\right) \circ \mathbf{d\hat{C}}_{i+1} \quad (32)$$

With the above definitions, the expressions for $\phi^{(n)}$ and $\psi^{(n)}$ are given by Equations (33) and (34), respectively.

$$\phi_i^{(n)} = \Theta_n(\sin\theta_i \sin\theta_{i+1} \cos\gamma_i, \sin\theta_i \sin\theta_{i+1}) \quad (33)$$

$$\psi_i^{(n)} = \sin\theta_i \sin\theta_{i+1} \sin\gamma_i \times \Upsilon_{n-1}(\sin\theta_i \sin\theta_{i+1} \cos\gamma_i, \sin\theta_i \sin\theta_{i+1}) \quad (34)$$

The derivatives of x and y and, thereby those of U_{tor}, in the virtual-bond vectors, can be calculated by using Equations (35 – 40).

$$\nabla_{\mathbf{dC}_{i-1}} x = -\frac{1}{||\mathbf{dC}_{i-1}||}\left[\mathbf{d\hat{C}}_{i+1}^T - (\mathbf{d\hat{C}}_{i-1} \circ \mathbf{d\hat{C}}_{i+1})\mathbf{d\hat{C}}_{i-1}^T\right] + \cos\theta_{i+1} \nabla_{\mathbf{dC}_{i-1}} \cos\theta_{i-1} \quad (35)$$

$$\nabla_{\mathbf{dC}_i} x = \cos\theta_{i+1} \nabla_{\mathbf{dC}_i} \cos\theta_{i-1} + \cos\theta_{i-1} \nabla_{\mathbf{dC}_i} \cos\theta_{i+1} \quad (36)$$

$$\nabla_{\mathbf{dC}_{i+1}} x = -\frac{1}{||\mathbf{dC}_{i+1}||}\left[\mathbf{d\hat{C}}_{i-1}^T - (\mathbf{d\hat{C}}_{i-1} \circ \mathbf{d\hat{C}}_{i+1})\mathbf{d\hat{C}}_{i+1}^T\right] + \cos\theta_{i-1} \nabla_{\mathbf{dC}_{i+1}} \cos\theta_{i+1} \quad (37)$$

$$\nabla_{\mathbf{dC}_{i-1}} y = \frac{1}{||\mathbf{dC}_{i-1}||}(\mathbf{d\hat{C}}_i \times \mathbf{d\hat{C}}_{i+1} - y\mathbf{d\hat{C}_{i-1}})^T \quad (38)$$

$$\nabla_{\mathbf{dC}_i} y = -\frac{1}{||\mathbf{dC}_i||}(\mathbf{d\hat{C}_{i-1}} \times \mathbf{d\hat{C}_{i+1}} + y\mathbf{d\hat{C}_i})^T \quad (39)$$

$$\nabla_{\mathbf{dC}_{i+1}} y = \frac{1}{||\mathbf{dC}_{i+1}||}(\mathbf{d\hat{C}_{i-1}} \times \mathbf{d\hat{C}_i} - y\mathbf{d\hat{C}_{i+1}})^T \quad (40)$$

The gradient in virtual-bond vectors can easily be transformed into the gradient in Cartesian coordinates by using Equations (41) and (42).

$$\nabla_{\mathbf{C}_i} = \nabla_{\mathbf{dC}_{i-1}} - \nabla_{\mathbf{dC}_i} - \nabla_{\mathbf{dX}_i} \quad (41)$$

$$\nabla_{\mathbf{X}_i} = \nabla_{\mathbf{dX}_i} \quad (42)$$

Correlation terms ($U_{corr}^{(3)}$)

The terms $U_{corr}^{(3)}$, which account for the coupling of the backbone-local and backbone-electrostatic interactions also depend on the products of the cosines or sines of the virtual-bond-dihedral

MOST WIEDZY Downloaded from mostwiedzy.pl

and the sines of the neighboring virtual-bond angles[35,37]. These are expressed by Equation (43)[35,37].

$$U^{(3)}_{corr;ij} = \frac{s_{el;i+1,j+1}}{R^3_{ij}} \left[\boldsymbol{\mu}_i \circ \boldsymbol{\mu}_j - 3(\boldsymbol{\mu}_i \circ \hat{\mathbf{R}}_{ij})(\boldsymbol{\mu}_j \circ \hat{\mathbf{R}}_{ij}) \right] \tag{43}$$

where $s_{el;i+1,j+1} = \sqrt{B_{p_{i+1},p_{j+1}}}$ of Equation (5) of Ref. 76 absorbs the magnitude of the dipole moments of the interacting peptide groups (it depends on whether the residues at positions $i+1$ and $j+1$ are regular residues or proline residues[77]) and the effective dielectric constants, R_{ij} is the distance between peptide group i (located between C^α_i and C^α_{i+1}) and peptide group j (located between C^α_j and C^α_{j+1}), $\hat{\mathbf{R}}_{ij}$ is the unit vector pointing from peptide group i to peptide group j, and $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ are fictitious dipole moments, which depend on the coefficients of the second-order Fourier expansion of the energy surface of a terminally-blocked residue[35,37], which are defined by Equation (44).

$$\boldsymbol{\mu}_k = b_{21;k}\hat{\mathbf{y}}^R_k + b_{22;k}\hat{\mathbf{z}}^R_k + b_{11;k+1}\hat{\mathbf{y}}^L_{k+1} + b_{12;k+1}\hat{\mathbf{z}}^L_{k+1} \tag{44}$$

where the coefficients $b_{lm;k},\ k = 1,2,\ l = 1,2$ are the above-mentioned coefficients of the expansion of the energy surface of the terminally-blocked residue with index k, defined by Equation (45)[35,37], while $\hat{\mathbf{y}}^L_k$, $\hat{\mathbf{z}}^L_k$, $\hat{\mathbf{y}}^R_k$, and $\hat{\mathbf{y}}^R_k$ are the y and z axes of the right-handed coordinate systems of residue i whose x axes are vectors $-\mathbf{d}\hat{\mathbf{C}}_{k-1}$ and $\mathbf{d}\hat{\mathbf{C}}_k$, respectively, and the y axes are in the $\mathrm{C}^\alpha_{i-1}\cdots\mathrm{C}^\alpha_i$ and $\mathrm{C}^\alpha_i\cdots\mathrm{C}^\alpha_{i+1}$ planes, respectively, as defined by Equations (46–49).

$$b_{ij;k} = b_{ij;k}(\theta_k) = \left[b^{(0)}_{ij;k} + b^{(1)}_{ij;k}\cos\theta_k + b^{(2)}_{ij;k}(\cos\theta_k)^2 \right] \sin\theta_k = b'_{ij;k}\sin\theta_k,\ i = 1,2;\ j = 1,2 \tag{45}$$

The coefficients $b^{(0-2)}_{ij;k}$ in Equation (45) depend on the kind of residue with index k. In the present UNRES we define three kinds of residues regarding local interactions: glycine, proline, and alanine, alanine comprising all residues except for glycine and proline. The coefficients for D-amino-acid residues are obtained by changing the sign of the $b_{2j;k}$ coefficients[35,37].

MOST WIEDZY Downloaded from mostwiedzy.pl

$$\hat{\mathbf{y}}_k^L = \frac{1}{\sin\theta_k}\left(\mathbf{d}\hat{\mathbf{C}}_{k-1} + \cos\theta_k \mathbf{d}\hat{\mathbf{C}}_k\right) \quad (46)$$

$$\hat{\mathbf{z}}_k^L = \frac{1}{\sin\theta_k}\mathbf{d}\hat{\mathbf{C}}_k \times \mathbf{d}\hat{\mathbf{C}}_{k-1} \quad (47)$$

$$\hat{\mathbf{y}}_k^R = \frac{1}{\sin\theta_k}\left(\mathbf{d}\hat{\mathbf{C}}_k + \cos\theta_k \mathbf{d}\hat{\mathbf{C}}_{k-1}\right) \quad (48)$$

$$\hat{\mathbf{z}}_k^R = \frac{1}{\sin\theta_k}\mathbf{d}\hat{\mathbf{C}}_{k-1} \times \mathbf{d}\hat{\mathbf{C}}_k = -\hat{\mathbf{z}}_k^L \quad (49)$$

It should be noted that the terms $\sin\theta_k$ in Equation (45) cancel with those in the denominators in Equations (46–49), when combined in Equation (44). The final folded expression for the fictitious dipole moment of peptide group k, μ_k is given by Equation (50).

$$\begin{aligned}\boldsymbol{\mu}_i = {} & b'_{21;i}\mathbf{d}\hat{\mathbf{C}}_{i-1} + b'_{11;i+1}\mathbf{d}\hat{\mathbf{C}}_{i+1} + \left(b'_{21;i}\cos\theta_{i-1} + b'_{11;i+1}\cos\theta_i\right)\mathbf{d}\hat{\mathbf{C}}_i \\ & -b'_{22;i}\mathbf{d}\hat{\mathbf{C}}_{i-1} \times \mathbf{d}\hat{\mathbf{C}}_i - b'_{12;i+1}\mathbf{d}\hat{\mathbf{C}}_i \times \mathbf{d}\hat{\mathbf{C}}_{i+1} \end{aligned} \quad (50)$$

In contrast to the expression derived in our earlier work[35,37], Equation (50) involves no explicit dependence on the virtual-bond-dihedral angle γ_i and expresses $\boldsymbol{\mu}_i$ in the global coordinate system. In our earlier work[35,37] the dipole moments were expressed in local coordinate systems of the respective residues, which required the calculation of the scalar products of the unit vectors of the coordinate systems to compute the energy and transformation of forces from the local to the global coordinate system for each pair of interacting peptide groups, this incurring additional cost. The new formulas are not only more stable numerically but reduce the number of operations by about 25 % with respect to the old formulas.

To calculate the gradient of $U_{corr}^{(3)}$, in this work we use Equations (51) and (52), respectively. Equation (51) expresses the radial derivatives that depend explicitly on C^α-atom coordinates, while Equation (52) expresses the derivatives in virtual-bond vectors. The derivatives in virtual-bond vector can easily be converted into those in C^α coordinates by using Equation (41).

$$\nabla_{\mathbf{C}_k} U_{corr;ij}^{(3)} = \frac{3}{2}\frac{U_{corr;ij}^{(3)}}{R_{ij}}\hat{\mathbf{R}}_{ij}^T, k = i, i+1 \quad (51)$$

$$\nabla_{\mathbf{dC}_k} U_{corr;ij}^{(3)} = \frac{s_{el;i+1,j+1}}{R_{ij}^3}\left[\boldsymbol{\mu}_j - 3\left(\boldsymbol{\mu}_j \circ \hat{\mathbf{R}}_{ij}\right)\hat{\mathbf{R}}_{ij}\right]^T \nabla_{\mathbf{dC}_k}\boldsymbol{\mu}_i, k = i-1, i, i+1 \quad (52)$$

MOST WIEDZY Downloaded from mostwiedzy.pl

The derivatives corresponding to site j can easily be obtained by exchanging j with i. In Equation (51), we used the relationship between the C^{α}- and peptide-group coordinates [Equation (3)]. The matrix of the derivatives of the dipole moment $\nabla_{\mathbf{dC}_k}\boldsymbol{\mu}_i$ is given by Equation (53).

$$\begin{aligned}\nabla_{\mathbf{dC}_k}\boldsymbol{\mu}_i &= b'_{21;i}\nabla_{\mathbf{dC}_k}\mathbf{d\hat{C}}_{i-1} + b'_{11;i+1}\nabla_{\mathbf{dC}_k}\mathbf{d\hat{C}}_{i+1} + \mathbf{d\hat{C}}_i\left(b'_{21;i}\nabla_{\mathbf{dC}_k}\cos\theta_{i-1} + b'_{11;i+1}\nabla_{\mathbf{dC}_k}\cos\theta_i\right) \\ &+ \left(b'_{21;i}\cos\theta_{i-1} + b'_{11;i+1}\cos\theta_i\right)\nabla_{\mathbf{dC}_k}\mathbf{d\hat{C}}_i \\ &-b'_{22;i}\nabla_{\mathbf{dC}_k}\left(\mathbf{d\hat{C}}_{i-1}\times\mathbf{d\hat{C}}_i\right) - b'_{12;i+1}\nabla_{\mathbf{dC}_k}\left(\mathbf{d\hat{C}}_i\times\mathbf{d\hat{C}}_{i+1}\right), k = i-1, i, i+1 \quad (53)\end{aligned}$$

The gradients of the cosines of the virtual-bond angles (θ) are expressed by Equations (21) and (22), respectively, while the other gradients are defined by Equations (54) and (55), respectively.

$$\nabla_{\mathbf{dC}_i}\mathbf{d\hat{C}}_i = \mathbf{I} - \mathbf{d\hat{C}}_i\mathbf{d\hat{C}}_i^T \quad (54)$$

$$\nabla_{\mathbf{dC}_i}\left(\mathbf{d\hat{C}}_i\times\mathbf{d\hat{C}}_{i+1}\right) = \begin{pmatrix} 0 & d\hat{C}_{i+1,z} & -d\hat{C}_{i+1,y} \\ -d\hat{C}_{i+1,z} & 0 & d\hat{C}_{i+1,x} \\ d\hat{C}_{i+1,y} & -d\hat{C}_{i+1,x} & 0 \end{pmatrix} - \left(\mathbf{d\hat{C}}_i\times\mathbf{d\hat{C}}_{i+1}\right)\mathbf{d\hat{C}}_i^T \quad (55)$$

where $\mathbf{I}$ is the unit matrix of dimension 3. It should be noted that $\nabla_{\mathbf{dC}_k}\mathbf{d\hat{C}}_i = 0$ for $k \neq i$ and $\nabla_{\mathbf{dC}_i}(\mathbf{d\hat{C}}_i\times\mathbf{d\hat{C}}_{i+1}) = 0$ if $k \neq i$ and $k \neq i+1$. The derivatives of $\mathbf{d\hat{C}}_i\times\mathbf{d\hat{C}}_{i+1}$ in the components of $\mathbf{dC}_{i+1}$ can also be obtained from Equation (55) by exchanging the indices i and $i+1$ and changing all signs in that Equation.

It should be noted that, with the algorithm for the computation of $U_{corr}^{(3)}$ outlined above, the bulk of computations is done on the "per residue" and not "per residue pair" basis, this involving linear growth of the computational cost with system size. Once the dipole moments and their derivatives are computed, the evaluation of energy requires calculating the peptide-group distance (R_{ij}) and the respective unit vector ($\hat{\mathbf{R}}_{ij}$), this involving a total of 3 multiplications, one square root, and 3 division, then 4 multiplications and 1 division to compute the energy, and a total of 18 multiplications and 9 divisions to compute gradient components. On the other hand, due to large amount data that would have to be transmitted,

MOST WIEDZY Downloaded from mostwiedzy.pl

the computation of the dipole moments and their derivatives, as well as that of the quantities needed for the computation of the virtual-bond and virtual-bond-dihedral angle energies, cannot be parallelized on the distributed-memory basis, this resulting in reduced parallel efficiency of energy and gradient computation in case many cores are used. Therefore, we parallelized this part with OpenMP, in the shared-memory mode.

Handling long-range interactions

In our earlier work on the parallelization of UNRES[56], we considered proteins with size less than 1,000 amino-acid residues, which enabled us to consider all long-range interactions at the coarse-grained level. The list of interacting pairs was, therefore, fixed and could be distributed between parallel tasks at the beginning of a run. In this work, to treat large proteins, we applied cut-off on long-range energy terms [$U_{SC_iSC_j}$, $U_{SC_ip_j}$, $U_{p_ip_j}^{VDW}$, $U_{p_ip_j}^{el}$, and $U_{corr}^{(3)}$ in Equation (1)]. Because no explicit charge-charge interactions are considered in the present UNRES, we did not use PME at this time, although we plan to do so in the future in order to treat $U_{corr}^{(3)}$, which are on the borderline between the slow- and fast-decaying energy contributions. It should be noted that the long-range multibody terms in the scale-consistent UNRES ($U_{corr}^{(3)}$) depend only on the distance between two peptide groups, their multibody component coming from the coupling with the local interactions within the neighboring residues [cf. section "Correlation terms ($U_{corr}^{(3)}$)"]. Thus, they are technically pairwise. We set the respective $U_{X_iY_j} = 0$ (where X_i and Y_j denote the ith interaction site of type X and the jth interaction site of type Y, respectively) if the site-site distance $r_{X_iY_j} > r_{cut}$, r_{cut} being the cut-off distance determined to keep energy error reasonably small and apply a smoothing function, $s(r)$, introduced in r-RESPA[78] and used in our earlier work[43] in the transition region to achieve continuous forces [Equation (56)].

$$s(r) = \begin{cases} 1 & r < r_{cut} - \lambda \\ 1 + \gamma^2(2\gamma - 3) & r_{cut} - \lambda \le r < r_{cut} \\ 0 & r \ge r_{cut} \end{cases} \tag{56}$$

$$\gamma = \frac{r - (r_{cut} - \lambda)}{\lambda} \tag{57}$$

MOST WIEDZY Downloaded from mostwiedzy.pl

We set $U_{X_iY_j}(r) \leftarrow s(r)U_{X_iY_j}(r)$ and $\lambda = 0.3$.

In the following subsections we describe the choice of an optimal cut-off distance and the construction and management the interaction list.

Determining the cut-off distance

To determine the cut-off distance, we generated diverse conformations of the Aβ_{40} pentamer by performing multiplexed replica exchange molecular dynamics (MREMD). The starting structure was formed by extended chains stacked on top of each other and separated by 20 Å. We ran 1,000,000 simulation steps with the 0.489 fs time step at 20 temperatures equal to 250 K, 260 K, 270 K, 280 K, 285 K, 290 K, 295 K, 300 K, 305 K, 310 K, 315 K, 320 K, 330 K, 340 K, 350 K, 360 K, 370 K, 380 K, 390 K, and 400 K, respectively (20 trajectories). Replicas were exchanged every 10,000 steps and snapshots were saved at every replica exchange, giving a total of 2,000 conformations. The energies of all conformations were recalculated with r_{cut} of 20 Å, 25 Å, and 30 Å. A plot of the distributions of the differences between the energies calculated with cut-off and without cut-off is shown in Figure S1 of the Supplementary Material. The average differences and standard deviations of energy for the three cut-off radii are $\overline{\Delta E_{20}} = -0.45$ kcal/mol, $\sigma_{E,20} = 0.25$ kcal/mol, $\overline{\Delta E_{25}} = -0.03$ kcal/mol, $\sigma_{E,25} = 0.12$ kcal/mol, and $\overline{\Delta E_{30}} = 0.01$ kcal/mol, $\sigma_{E,30} = 0.05$ kcal/mol, respectively. Given a small average difference between the energy calculated without and with applying the cut-off and small standard deviations of the energy difference, we selected $r_{cut} = 25$ Å.

Construction and management of interaction lists

In all-atom simulations with explicit solvent the space is nearly uniformly filled with particles (atoms), thus enabling us to distribute the particles between processes or threads in a parallel implementation of the code, by using the domain-decomposition algorithms[79–83]. Conversely, in a coarse-grained system with implicit solvent the particles (CG interaction sites) do not fill the space uniformly. Consequently, as in our earlier parallel implementation of UNRES[56], we make explicit lists of pairs of interacting sites, which we term the *interaction lists*. To construct the interaction lists, we took an approach combining the cut-off distance, cell index method[79,80] and Verlet neighbor list method[81], together with a few tweaks. Similar

MOST WIEDZY Downloaded from mostwiedzy.pl

approaches, which have been coined a name of pairwise Verlet lists, can be found in the literature[82,83]. It should be noted that, in our approach, the cells play only an auxiliary role by enabling us to exclude the majority of pairs that are out of the cut-off distance by checking cell coordinates and without having to calculate all site-site distances.

We build three lists of interactions, one for the SC–SC, one for the SC–p interactions, and one for all interactions between the peptide groups ($U_{p_ip_j}^{VDW}$, $U_{p_ip_j}^{el}$, and $U_{corr}^{(3)}$). At the construction time, a given list contains the pairs of sites of given kind(s) that lie within a distance of $r_{cut} + \delta$, where δ is the cut-off (or Verlet) buffer. In our implementation, $r_{cut} = 25$ Å (see the preceding subsection) and we set $\delta = 0.5$ Å. The purpose of introducing δ is to enable us to use an interaction list for a number of MD steps and its value has been determined as a compromise between the number of MD steps during which a given list can be used on one hand and not including too many pairs that are separated by more than the cut-off distance on the other hand. With the current settings, there are 5 % pairs beyond the cut-off that need to be processed, while the same lists of interactions can be used every 50 MD steps on average.

To determine whether the interaction lists need to be rebuilt, we track the movement of every particle. The lists are rebuilt if any particle moves by more than half of the Verlet buffer, because this means that some pairs that had been beyond the $r_{cut} + \delta$ at list-construction time could become closer than r_{cut}. This approach is similar to the one taken in LAMMPS[83].

Rebuilding the interaction list begins with assigning particles to cells with each side equal to or slightly greater than the extended cut-off length. Such a choice of cell size guarantees that any two interaction sites that are within the considered distance occupy either the same or neighboring grid cells. If a periodic-box side is not an integer multiple of this cut-off length, the cells are enlarged accordingly to fully cover the entire simulation space with an integral number of cells.

For every cell, a list is constructed, which holds those particles that are potentially in the range of a particle within this cell. Once a particle has been assigned to its home cell (based on its coordinates), it is added to its home cell's list and to the lists of those of the 26 neighboring cells, which are sufficiently close to this particle. The latter condition is checked

MOST WIEDZY Downloaded from mostwiedzy.pl

by computing the distance of the particle to the closest point of the neighbor cell. If this distance does not exceed the extended cut-off distance, the particle is added to the list of the neighboring cell, otherwise it is not (see Figure 2). While this approach multiplies lists' memory usage up to 27-fold, only the particles from the home-cell list of a given particle need to be considered to find the particles which are within the extended cut-off distance from it. These neighbors can be enumerated in a sequential order, which is important for further parallelization (see subsection "Parallelization"). If every particle appeared only in its home cell's list, the 27 lists would have to be merged and sorted to obtain such a sequential list of neighbors.

Once the grid is fully built, the cell list of every particle is iterated over. At each iteration, the distance of the particle from a given neighbor is calculated and the pair consisting of the particle and the neighbor is appended to the interaction list if they are not farther from each other than the cut-off distance. The cost of the above computations scales linearly with system size, although with quite a large constant, except for the final step.

Each of the three interaction lists is stored in a 2D array. The first element of each row is the index of the first particle in a pair, while the second and the third elements define the range of the index of the second particle in the pair. For example, if the interactions involving pairs $(1,2),(1,3),\ldots\ (1,10),(1,20),(1,21),\ldots\ (1,30)$ have been enumerated (the interactions of particle 1 with particles 11 through 19 being excluded because by applying cut-off), the two consecutive rows of the interaction array are $1,2,10$ and $1,20,30$. This form of storage requires much less memory than explicit storage of all pairs of the particles that are within the cut-off distance.

The construction of the interaction lists is a good target for parallelization and has been parallelized with OpenMP and MPI in our current implementation of UNRES.

Parallelization

Parallelization strategy and tools

We apply the two-grain parallelization scheme of the first parallel implementation of UNRES[56], which is depicted in Figure 3A. At the *coarse-grain level*, the tasks assigned to a

MOST WIEDZY Downloaded from mostwiedzy.pl

parallel job are divided into coarse-grain (CG) tasks, each handling an MD trajectory in an (M)REMD or a multi-trajectory canonical MD run or other multi-CG-task runs (e.g. evaluation or minimization of the energy of multiple input structures). In an (M)REMD run, the CG tasks are synchronized every pre-set number of MD steps (cf. section "Extensions of UNRES MD"), in the other types of runs they are synchronized only at the end of a run.

The CG-level parallelization is accomplished with MPI[45] exclusively. Thus, each CG task corresponds to an MPI process. All CG processes have their dedicated communicator (CG_COMM). Synchronization is governed by the assigned master CG task, which also does its share of computations. For the non-(M)REMD run types, the master CG task only serves to synchronize the termination of all tasks to end the parallel job smoothly.

At the *fine-grain level*, each CG task is divided into fine grain (FG) tasks, each handling a single MD trajectory, energy minimization or energy evaluation. This parallelization is also done with MPI. The FG tasks that correspond to CG task n have their dedicated communicator (FG_COMMn). The master FG task does its share of computations, divides the work between the slave tasks, collects the energy and energy-gradient contributions from the slave tasks, synchronizes the slave tasks, updates the coordinates and velocities in MD steps, does energy minimization, and also does the I/O operations. The FG slave tasks only do their share of energy and energy-gradient evaluation.

In this work, we have extended the earlier parallel implementation of UNRES[56] by splitting the FG tasks into OpenMP[46] threads. With a single FG task per conformation, fine-grain parallelization is done using OpenMP only, while hybrid (MPI and OpenMP) parallelization takes place when the number of FG tasks is greater than 1. OpenMP is more efficient with shared-memory machines, because all threads of a process parallelized with OpenMP live in the same address space, thereby minimizing communication which, as opposed to MPI, does not involve the operating-system layer. Consequently, there is no substantial communication overhead. This feature has enabled us to parallelize not only the pairwise energy computation but also the computation of the arrays and vectors needed for U_b, U_{tor}, U_{rot} and $U_{corr}^{(3)}$. These calculations require a remarkable fraction of computing time and cannot be parallelized with MPI due to the need of significant data transmission. On the other hand, the downside of OpenMP is that it is possible only within a single physical

MOST WIEDZY Downloaded from mostwiedzy.pl

node of a supercomputer.

Parallelization with MPI

For MPI parallelization, we use essentially the same scheme as tools as in our earlier work[56] except that only those pairwise interactions which are in the interaction lists are considered. Collective communication is used extensively at both the CG and the FG level. In particular, the coordinates are transmitted to the slave FG processes with MPI_Bcast and MPI_Reduce is used to collect the energy and energy-gradient contributions. The interaction lists are partitioned between the MPI processes subject to the condition of load balance; subsequently each chunk is partitioned into OpenMP threads, if applicable, as shown in Figure 3B. A scheme of work and communication between the CG and FG tasks is shown in Figure S2 of the Supplementary Material.

Parallelization with OpenMP

As mentioned, in the current implementation of UNRES, the FG processes are split into OpenMP threads, thus enabling us to take advantage of shared memory. We also utilize the instruction-level parallelism termed SIMD (Single Instruction Multiple Data) to parallelize the most time-consuming loops. The OpenMP parallelism, however, carries a risk that a number of threads could interfere with each other when trying to write to the common address at the same time, namely while updating the energy, energy-gradient components, and other quantities in a common accumulator, unless the threads are synchronized at every update, this incurring severe loss of performance. Therefore, in our current OpenMP implementation of UNRES, every thread has its own private copy of every accumulator where it gathers its result. After completing the loop, all accumulators are summed to produce the final result. This approach requires more memory but no coordination of accesses between threads and, thus, is much more performant.

Every thread runs multiple SIMD lanes that also must be coordinated to avoid updating the same memory locations simultaneously. This problem was solved by grouping together interactions. A group consists of a set of interactions between a single particle (the "source") and a number of its neighbors. For such a group, the computations can be safely vectorized,

MOST WIEDZY Downloaded from mostwiedzy.pl

because they do not involve overlapping updates. All neighbors are distinct from each other and also from the source. The updates to the source can be gathered in a separate accumulator that is added to the remaining results only after the loop completes, as shown in Algorithm 1.

Algorithm 1 Handling concurrent updates to the common particle of interactions in a group. $accumulator\_for\_u$ is summed with OpenMP reduction clause.

1: $u = first\_of\_group$
2: $accumulator\_for\_u = 0$
3: **for** $i = 1, number\_of\_pairs\_in\_group$ **do** ▷ **OpenMP parallelized**
4: $v = second\_in\_group(i)$
5: $contribution = calculate\_contribution(u, v)$
6: $accumulator\_for\_u = accumulator\_for\_u + contribution$ ▷ **OpenMP reduction**
7: $accumulator(v, my\_thread) = accumulator(v, my\_thread) + contribution$
8: **end for**
9: $accumulator(u, my\_thread) = accumulator(u, my\_thread) + accumulator\_for\_u$

The first index of the accumulator corresponds to the index of the respective contribution to the energy gradient; if the accumulator stores the respective chunk of an energy contribution, there is no first index.

The OpenMP dynamic scheduler is also used to balance the load during building the interaction lists. Specifically, interaction sites are grouped in batches of 16 consecutive elements. Threads repeatedly take the next free batch and determine, if interactions between sites in the batch and all other sites are within a given threshold. The value of 16 was chosen as a balance between conflicting goals. On the one hand, the number of batches should be significantly greater than the number of threads so the processing time differences between batches are not as impactful. A batch must be large enough to minimize the relative overhead of work allocation and thread management. Results are then merged and sorted to create a deterministic order of interactions, independent of the number of threads or the batch scheduling sequence. In this step, every thread has an equal number of interactions to process and every interaction requires the same amount of work. Finally, the resulting list of

MOST WIEDZY Downloaded from mostwiedzy.pl

interactions is equally partitioned for the threads to handle in the next computational steps.

Another issue with the SIMD vectorization is that a conditional execution flow is very difficult for a compiler to parallelize. Such a code often ends up with computing both conditional branches and then selecting the result based on a given condition. If one of these paths of computations is costly but rarely used, it can noticeably impact performance. This is the case of the SC-p interactions, which only require the powers of the square of the site-site distance but, with the smoothing function [Equation (56)] which is needed for the pairs that are between the regular and the extended cut-off, the square root is also required. However, the square of the distance is sufficient to check whether the sites involved are within this range. Similarly, for all interactions, only the square of the distance must be checked to determine whether the sites are within the extended cut-off.

Given the above considerations, we rewrote the code to provide different paths for different computations. A given list of interactions is first split into equal-sized lists for OpenMP threads. Every thread then passes its list through a scheduler. Its job is to verify that the interaction is within the cut-off limit, by calculating the square of the Euclidean distance (see section "Profiling"). Then, if the interaction has not been rejected, it is scheduled to one of different queues, based on the particles of the pair and their distance. The pairs to be processed by the same code path, which contain the same source particle are gathered in one queue, termed the *sequential queue*, those that break the sequence but still need the same code path and keep the same source particle are gathered in a *non-sequential queue*, while the pairs requiring different execution paths are gathered in other, dedicated queues. When a queue fills up or a pair with another source particle was reached, the data gathered in a queue are passed to a proper computing function (Figure 4).

When running parallel jobs with OpenMP, it should be kept in mind that the NUMA (Non Uniform Memory Access) topology is not covered. This can result in some latency when OpenMP threads encompass more than a single physical processor. For example, when running the current implementation of UNRES on a machine with 2 12-core processors/node, we observed up to 20 % longer run times of the 1-process 24-threads per process runs compared to the 2-process, 12-threads per process setup, while running the code on a node with 2 12-core CPUs, until the number of such out-of-NUMA-node memory accesses

MOST WIEDZY Downloaded from mostwiedzy.pl

were significantly reduced.

Hardware implementation of the parallel code

The new UNRES code was implemented on the Tryton Linux cluster located in the Academic Computer Center in Gdańsk, TASK, with the peak performance of 1.792 PFLOPS. The cluster consists of 1607 servers with a total of 3214 Intel Xeon Processors E5 v3 @ 2,3 GHz, 12-core (Haswell), 128/256 GB RAM DDR4 memory per server, InfiniBand FDR 56 Gb/s network, fat-tree topology, Mellanox switches. The development environment consisted of two reserved nodes of Tryton supercomputer and Intel Parallel Studio XE Cluster edition 2019 toolkit, providing the Fortran compiler (`ifort` version 19.0.5.281 20190815) both OpenMP and MPI runtime (version 2019 Update 5 Build 20190806).

Profiling

We used an in-house built profiler, which is based on Linux performance events API, and the Intel VTune Profiler[84] to monitor the performance of UNRES and to detect the bottlenecks in the code. These two tools have different area of expertise. The in-house profiler was designed as a quick-and-dirty alternative for a Linux `perf` utility, and a research platform for gathering and correlating performance data from jobs run on a supercomputer. It was used for sampling the executable during the run and precisely pinpointing the hot spots. On the other hand, the VTune Profiler was used to gather more coarse-grained information about the code, but also of a much wider scope.

With the profiling tools described above, we identified and eliminated a number of important bottlenecks in the code. The first one was confining particles to the periodic box and selecting, for a pair of particles, their images that have the shortest distance (wrapping). The use of the floating point `mod` function seems to be a natural choice; however, its cost is remarkable given the fact that, for wrapping, it has to be called for every pair within the cut-off. Therefore, the `floor` function was used to confine the particles to the periodic box and a series of additions and subtractions was used for wrapping. Another improvement was introducing explicit code of the Gay-Berne potentials with fixed "12" and "6" exponents.

MOST WIEDZY Downloaded from mostwiedzy.pl

Although these are the values that are exclusively used at present in UNRES, the exponents are read from the respective parameter file to give the developers some room for modification, should there be a need to change the exponents to make the potential less or more steep. Another improvement was the computation of the square of the distance only instead of the distance itself to check if an interaction is within the extended cut-off and to evaluate the bulk of $U_{SC_i p_j}$. Finally, the profiler enabled us to identify all sections of the code in which multiple arrays were zeroed out in the same loop which, for large arrays, implies consecutive writings to distant memory elements, incurring significant delays.

Memory requirements

As can be gathered from the description of the modifications of the UNRES package discussed in the subsections above, the memory requirements of the present code scale linearly with the system size, thus enabling us to treat very large protein systems. The total memory requirements are about $6 \times 100 \times n$ double precision words for the storage of coordinates, gradient components and auxiliary quantities for energy and gradient calculations and, $2 \times (6 \times n - 3)$ double-precision words for the storage of the five-band inertia and friction matrices and their inverses, $9 \times 52 \times n$, $n \times number\_of\_CG\_tasks \times 6$ double precision words for storing the information from all trajectories by the master CG processes in (M)REMD runs, and $9 \times 52 \times n$ integer words for the storage of the interaction lists, where n is the number of residues. The largest protein system considered in this work has over 150,000 amino-acid residues and the largest number of trajectories in our MREMD simulations was 96 (usually 48 are run). Altogether, about 1 GB of memory is required for a protein of this size. This analysis does not include auxiliary bioinformatics and sparse experimental data, which occupy about 3.2 GB of memory for proteins with this size; however, these data are rarely used in full amount for such big systems. Altogether, the memory requirements are very moderate.

The main reason for favorable memory scaling was changing the coordinates in the equations of motion from virtual-bond vectors to C^α atoms and SC centers, this resulting in the reduction of the inertia matrix from a symmetric square matrix [which requires storing $2n(2n+1)/2$ elements] to a symmetric five-band matrix (which requires storing only $6n - 3$

MOST WIEDZY Downloaded from mostwiedzy.pl

elements). It can easily be realized that a 150,000 residue system would be impossible to treat, because storing the inertia matrix alone would require over 160 GB of RAM even in single precision, given the fact that also the inverse of the inertia matrix has to be computed. With Langevin dynamics, these memory requirements are doubled because of the necessity to store the friction matrix. With the modification, only 0.033 GB of memory is needed for the storage of the inertia matrix and its inverse in double precision, the same amount being required for the friction matrix.

RESULTS AND DISCUSSION

Functional tests

To assess the correctness of the new code (after changing the variables to C^{α} and SC coordinates, introducing interaction lists and optimizing energy expressions), we carried out tests of energy conservation in the NVE mode (constant number of particles, constant volume, and constant energy) and temperature conservation in the NVT mode (constant number of particles, constant volume and constant temperature), using gramicidin D (a small dimer with chains containing both L- and D-amino-acid residues, tightly intertwined into a right-handed double helix; PDB code: 1AL4), as a benchmark. The calculations were run starting from the energy-minimized experimental structure.

The plots of the total energy for the time steps 0.978 fs, 4.89 fs, and 9.78 fs, 1,000,000 time steps run, are shown in Figure S3A of the Supplementary Material. It can be seen from the Figure that, the total energy oscillates about the "shadow Hamiltonian" value, the oscillations increasing with time-step length; however no drift and no jumps of the total energy are observed. The standard deviations of the total energy over the trajectory are 0.0009, 0.024, and 0.10 kcal/mol for the time step of 0.978 fs, 4.89 fs, and 9.78 fs, respectively, compared to those of the kinetic or potential energy, which range from 2.62 to 3.67 kcal/mol (panel B of the Figure). The differences between the minimum and maximum total-energy values are 0.009, 0.268, and 1.12 kcal/mol for the three time-step lengths. Consequently, the MD algorithm appears to be stable even with a nearly 10 fs time step, while the 4.89 fs time

MOST WIEDZY Downloaded from mostwiedzy.pl

step assessed in our earlier work (which reported the first MD-enabled version of UNRES)[39] as a recommended time-step length without using the time-split algorithms appears to be a safe value. Additionally, it can be seen from Figure S3B of the Supplementary Material that the kinetic and potential energies get swapped after 380,000 steps. Even though this change is not accompanied by any major structural changes (the radius of gyration of the dimer stays around 9 Å), the fact that the total energy does not drift or jump in the swapping region is an additional proof of the stability of the algorithm.

It should also be noted that, in the versions of UNRES/MD prior to the current one, we did observe occasional energy jumps in NVE trajectories, which was caused by instability of the energy expressions that included explicit torsional angles, removed in the current work owing to the introduction of the formulas described in section "Removing explicit angles from energy expressions". Further increase of the time step can be achieved with the time-split algorithms, one of which, based on the velocity-Verlet integration scheme has been implemented in UNRES/MD[39,85]. Implementation of the time-split algorithms in the optimized UNRES/MD code will be a subject of our further research.

To test temperature conservation, we carried out the calculations for the gramicidin D system with the Berendsen thermostat[65] (rescaling velocities) and in the Langevin mode (adding explicit friction and stochastic forces). The coupling constant in Berendsen calculations was set at $\tau = 48.9$ fs and the stochastic and friction forces in Langevin-dynamics calculations were scaled down by the factor of 0.01, as in our earlier work[28]. In both calculations 1,000,000 steps with 4.89 fs length were run, the thermostat temperature being $T = 300$ K. The graphs of temperature distribution with superposed theoretical distribution resulting from the Maxwell-Boltzmann law, are shown in Figure S3C of the Supplementary Material. The histograms were constructed from the trajectories after skipping the initial 100,000 step section, the momentary temperatures being sampled every 100 steps (a total of 9,000 snapshots). The temperature distribution resulting from the Langevin-mode calculations approximates very well the Maxwell-Boltzmann distribution [Equation (58)], while that resulting from Berendsen-mode simulations is too narrow, which is characteristic of the weak-coupling (WC) thermostat family, which includes the Berendsen thermostat. The momentary temperatures averaged over the 9,000 snapshots were 300.0 and 302.5 for the

MOST WIEDZY Downloaded from mostwiedzy.pl

Berendsen and the Langevin simulation, respectively, thus being equal or close to the bath temperature (300 K).

$$P(\overline{T}) = \frac{\left(\frac{g}{2T}\right)^{\frac{g}{2}}}{\Gamma(\frac{g}{2})} \overline{T}^{\frac{g-2}{2}} \exp\left(-\frac{g\overline{T}}{2T}\right) \tag{58}$$

where $\overline{T} = E_k/gR$ is the momentary temperature (with E_k being the kinetic energy and R being the universal gas constant), T is the bath temperature, g is the number of the degrees of freedom, and Γ is the Euler function.

Performance tests

The performance of a parallel application running in a high-performance computing (HPC) system can be assessed in several ways: the evaluation of the observed processing performance (e.g. in FLOPS), memory bandwidth vs. theoretical performance of a compute device such as CPU/GPU or memory specs as well as code scalability evaluated as speed-up (s), i.e., the ratio of the 1 core (or node/CPU/GPU) runtime divided by the n core (or node/CPU/GPU) runtime or parallel efficiency (η) computed as speed-up divided by n. These quantities are defined by Equations (59) and (60), respectively.

$$s = \frac{t(1)}{t(n)} \tag{59}$$

$$\eta = \frac{s(n)}{n} = \frac{t(1)}{n \times t(n)} \tag{60}$$

where $t(n)$ is the code-execution time with n cores.

In this section we describe the scalability of the new optimized UNRES code in single- and multi-trajectory runs, which correspond to fine-grain and coarse- and fine-grain parallelization runs (Figure 3) and compare the timing of the optimized UNRES code developed in this work with that of the previous versions and with that of all-atom simulations.

To assess the performance of UNRES, we selected 9 proteins with size from 109 to 153,243 amino-acid residues. The PDB IDs and other characteristics of these proteins are collected in Table 1.

MOST WIEDZY Downloaded from mostwiedzy.pl

In what follows, the old scale-consistent parallel UNRES code with the force field parameterized in our earlier work with 9 training proteins[37] is referred to as UNRES-9P. The non-optimized code obtained by changing the variables to C^{α} and SC Cartesian coordinates, which results is a five-band inertia matrix, and with interaction lists enabled is referred to as UNRES-9P-5D. These two versions of the UNRES code are parallelized with MPI only. The optimized UNRES-9P-5D code is referred to as UNRES-9P-5D(o). In this code, MPI, OpenMP, and mixed fine-grain parallelization has been implemented.

Scalability tests

Single-trajectory runs. For all proteins of Table 1, we carried out single-trajectory canonical-simulation runs with the Berendsen thermostat and time step of 0.489 fs. The number of time steps varied from 100 to 10,000, depending on system size and the number of cores. We analyzed only the non-setup time, i.e., the time spent on running MD, excluding data reading, generation of initial velocities, and other initial computations. For production runs, these computations constitute a negligible fraction of the total wall-clock time. For runs shorter than 10,000 steps, the non-setup time was scaled to correspond to 10,000 steps. For a given protein, all runs were started from the same coordinates and velocities obtained by energy-minimizing the starting structure (the experimental structure for all proteins except H1081, in which we used our CASP14 model[86], as its experimental structure is not available yet) and carrying out a 1000-step canonical MD run. Each run was repeated three times and the average non-setup time taken for further processing. In these calculations we used the fully optimized UNRES-9P-5D(o) version.

The speedup and efficiency plots obtained with using MPI parallelization and OpenMP or hybrid parallelization are shown in panels A – D of Figure 5, respectively. Detailed timings with splitting between the list construction and the evaluation of the site-site interaction energies ($U_{SC_iSC_j}$, $U_{SC_ip_j}$ and $U_{p_ip_j}$) are summarized in Table S1 of the Supplementary Material.

It can be seen from the Figure that OpenMP or hybrid parallelization results in better scalability. Inspection of the contributions to the computing time (Table S1) shows that the difference is caused by list construction, the evaluation of $U_{p_ip_j}$ and $U_{corr}^{(3)}$, local energy

MOST WIEDZY Downloaded from mostwiedzy.pl

components and the summation of gradient and calculating accelerations from forces. Some differences are also observed for the updates of coordinates and related quantities and velocities in MD steps; however, these calculations do not contribute much to the non-setup time. For proteins with size bigger than about 100 residues, there are only small differences in the ratios of the times for the evaluation of $U_{SC_iSC_j}$ or $U_{SC_ip_j}$ to non-setup time.

The above observations are illustrated in Figure 6, in which the ratios of the non-setup time and the times for list-construction and for the evaluation of the energy components obtained with 24 cores (the entire node) in two OpenMP modes: one with 24 threads (panel A of the Figure) and one with 2 FG processes (panel B of the Figure), each split into 12 threads, to those obtained with 24 MPI processes are shown. It can be seen that the differences are the biggest for energy and gradient-component summation (8 times or more) then for list construction (up to 3.5 times with 24 and up to 3 times with 2×12 OpenMP threads), and last is $U_{p_ip_j} + U^{(3)}_{corr}$ (up to 1.5 times). In summary, for bigger proteins, the total non-setup time is up to about 3 times greater for the calculations with 24 or 2×12 threads compared to those with 24 MPI processes. It can also be noted that the ratios of all contributions to the execution time in MPI and OpenMP/hybrid calculations grow with protein size to reach an asymptote.

It can also be seen that for the smallest protein (5OMT, 109 residues), the calculations with 24 MPI processes result in shorter execution times compared to those with 24 OpenMP threads and take about or a shorter time than those with 2 MPI processes split into 12 OpenMP threads each. With 24 threads, the computations of all components, except for list construction and gradient summation, take longer compared to that of the calculations with 24 MPI processes. The reason for unfavorable timing for OpenMP for small protein is a large overhead of initializing and management of the threads compared to the gain from using them.

Analysis of single-trajectory timings. An analysis of the relative contributions of the non-setup time mentioned above (Figure 7) helps us to understand the reason for the benefits from using shared-memory parallelization. List construction and calculation of $U_{p_ip_j} + U^{(3)}_{corr}$ are the dominant contributions to the non-setup time for which shared-memory calculations

MOST WIEDZY Downloaded from mostwiedzy.pl

are faster. With 24 MPI processes, these contributions require about 28 % and about 17 % of non-setup time, respectively, for the largest system. The longer time required for list construction with MPI parallelization results from extensive message passing between the processes while working on the load-balanced partition of the groups of interactions (Figure 3B). The partition of groups of interactions between OpenMP threads uses shared memory, which causes very low communication overhead, thus resulting in faster list construction. It can also be seen from Figure 7 that, while the contribution of the list-construction time to the non-setup time decreases with protein size, it remains nearly constant for systems with size bigger than 3,143 residues (half of 4R3O).

The reason for longer execution time of the $U_{p_ipj} + U_{corr}^{(3)}$ contributions with MPI compared to OpenMP is that, with cut-off on long-range interactions applied, the number of these interactions grows nearly linearly with the number of residues, as does the number of operations required to compute the quantities dependent on single-residue or adjacent-residue geometry that occur in the expressions for $U_{corr}^{(3)}$ (section "Correlation terms ($U_{corr}^{(3)}$)") and their derivatives. Thus, with many FG processes, the single-residue-quantity computations become a remarkable fraction of total computational effort. These computations have not been parallelized with MPI, because of large amount of data to be transmitted (which kills the gains from distributed-memory parallelization), but have been parallelized with OpenMP.

The next contribution that benefits from shared-memory parallelization is the summation of gradient components and computing accelerations, which takes about 14 % of non-setup time for the largest system for MPI parallelization. This contribution decreases to a few percent and takes up to 16-fold shorter (Figure 6) with OpenMP. One reason for the much longer execution time of this step with MPI is a large amount of communication when summing the gradient contributions from different processes (which uses shared memory with OpenMP) and parallelization of the transformation of some of some of the gradient contributions from virtual-bond vectors to C^{α} and SC-center coordinates, which can be accomplished efficiently only using shared memory. The overhead of energy and gradient-component calculations with MPI amounts to about 7 % for the largest system and is reduced about 15 fold for the largest system when using OpenMP. This overhead includes

MOST WIEDZY Downloaded from mostwiedzy.pl

the preparation for energy and gradient calculations (computing the positions of the peptide groups, wrapping the sites in the central box, zeroing the gradient components), which are worth parallelizing only with shared memory, summation of energy contributions from different MPI tasks, and MPI task synchronization.

The evaluation of local energy and energy-gradient components (U_{bond}, U_b, U_{tor}, and U_{rot}) amounts to about 12 % of the non-setup time for the largest systems and takes about 15 times longer with MPI than with OpenMP. The reason for this difference is that this contribution includes the computation of the scalar and cross-products and their derivatives described in sections "Virtual-bond-dihedral-angle terms (U_{tor})i" and "Virtual-bond-dihedral-angle terms (U_{tor})" (it should be noted that these quantities are also used for computing $U_{corr}^{(3)}$). These computations have not been parallelized with MPI but have been with OpenMP. Finally, the greater fraction of velocity-Verlet step processing in MPI calculations results from the contribution from coordinate updating and the calculation of normalized virtual-bond vectors, which have been parallelized only with OpenMP.

Analysis of hybrid-parallelization performance. While using OpenMP exclusively or in addition to MPI results in shorter non-setup times for all but the smallest systems, it is remarkable that, for the number of residues up to 3,143 (half of 4R3O), using 12 OpenMP cores gives a greater speed-up, compared to using 24 OpenMP cores, while the tendency is reversed for larger proteins. These differences are visualized in more detail in the form of bar plots of non-setup time per 10,000 MD steps shown in Figure 8. A more detailed plot that shows the differences of the non-setup times for all number of cores and parallel-run modes executed in this work is presented in Figure S4 of the Supplementary Material. As pointed out at the end of section "Parallelization with OpenMP", the use of 12 OpenMP threads on the hardware architecture with 2×12-core processors has the advantage of caching all data processed by OpenMP threads. The benefit from this is greater than the loss of using two FG MPI processes instead of one. However, for larger systems, the amount of data becomes too large to be cached and the threads have to use RAM, which is slower, this removing the advantage of caching and, consequently, making the use of 1 FG MPI process split into 24 threads a better choice. Qualitatively the same difference are observed for calculations with

MOST WIEDZY Downloaded from mostwiedzy.pl

2 nodes (2 MPI processes, each split into 24 OpenMP threads and 4 MPI processes, each split into 12 OpenMP threads, respectively).

It can be seen from Figure 6C, in which the ratio of the non-setup time contributions obtained with 2 MPI processes, each split into 12 OpenMP threads to that obtained with 24 OpenMP threads, that the list-construction contribution is the main reason for switching the order of non-setup times obtained in the two modes of calculation. The list-construction time starts to be greater for the 2×12 mode for 3,143 residues (half of 4R3O) while, starting from the whole 4R3O system (6,268 residues), the non-setup time becomes greater with the 2×12 mode. Compared to the change of the ratio of the list-construction time, those of the gradient/Lagrangian handling and local-energy calculations are much bigger; however, those contributions to the non-setup time are much smaller than that from list construction (Figure 7).

Multiple-trajectory MREMD runs. Because replica exchange involves synchronization of coarse-grained tasks (Figure 3A), the parallel performance of the calculations run in this mode need not be as high as that for single-trajectory runs. Therefore, we tested the parallel performance of MREMD calculations for the following three of the test proteins: 5HKQ (263 residues), 2SY1 (1458 residues), and H1081 (15,200 residues) (see Table 1 for the information of these proteins). For each of these proteins, we ran from 2 to 96 replicas, exchanged every 1,000 MD steps, the total length being 10,000 steps (10 exchanges). The time step was set at 4.89 fs and the Berendsen thermostat with the coupling parameter of $\tau = 48.9$ fs was used. For 5HKQ and 2SY1, each trajectory was run with 1, 4, 12, 24, and 2×12 cores dedicated to fine-grain tasks, pure OpenMP mode used in all but the last series of runs, in which 2 MPI tasks each split into 12 OpenMP threads were set. For H1081, additionally, runs with 2×24 (2 nodes) and 4×12 fine-grain cores were also carried out. Thus, the total number of cores dedicated to all (coarse-grain and fine-grain tasks) was from 2 to 4608. The results are shown as non-setup times and efficiency in Figure 9A-E, in which the non-setup times and efficiencies for single-trajectory runs (no exchanges) are also shown for reference. It can be seen that the non-setup times and efficiencies are nearly constant for the number of trajectories up to 24, some minor increase of the non-setup time and decrease of efficiency being observed

MOST WIEDZY Downloaded from mostwiedzy.pl

starting from 48 trajectories. Consequently, the coarse-grain task synchronization has a negligible effect on the parallel performance of the code. As observed in single-trajectory runs, organizing the fine-grain tasks per a 24-core node into 2 MPI tasks, each split into 12 OpenMP threads gives a better performance for the smallest protein considered in this section (5HKQ) but not for the larger proteins (2SY1 and H1081), for which allocating all cores of a node to OpenMP threads gives better results.

Timing of UNRES and other MD codes

The timings, measured as non-setup time per MD step and achievable simulation length (ns/day) of UNRES-9P, UNRES-9P-5D, and UNRES-9P-5D(o) (with or without the 25 Å cut-off on long-range interactions and MPI, OpenMP or hybrid parallelization), all-atom explicit-water GROMACS 2020[52] code (which uses MPI and OpenMP), all-atom implicit-water AMBER 20[51] (Generalized Born Surface Area, GBSA model[87,88]), and MARTINI 3[24] coarse-grained model with explicit coarse-grained solvent (built on the GROMACS platform) are plotted in number of amino-acid residues in panels A and B, respectively, of Figure 10. The 11 Å cut-off on non-bonded interactions and particle-mesh Ewald summation on electrostatic interactions was used in the explicit-solvent calculations with GROMACS. A 25 Å cut-off on all long-range interactions, including electrostatic interactions, was applied in the implicit-solvent AMBER calculations. All calculations were carried out on a single 24-core node; consequently exclusively OpenMP parallelization was used in the GROMACS and AMBER calculations and in the OpenMP-enabled UNRES calculations.

As can be seen from Figure 10, the previous, NEWCT-9P version of UNRES exhibits the poorest performance, with per-MD-step CPU time growing quadratically with system size. After optimization, the CPU time dropped by a factor of 10 when the optimized code with no cut-off on nonbonded interactions was run with MPI and about 20 when it was run with OpenMP. The NEWCT-9P-5D and NEWCT-9P-5D(o) execution times corresponding to runs with a 25 Å cut-off exhibit asymptotic linear dependence on protein size, the optimized code with MPI and OpenMP being 5 and 20 times faster, respectively. As can be seen from Figure S5 of the Supplementary Material, the change of the dependence of the non-setup time on protein size from quadratic to linear corresponds to that of the number of long-range

MOST WIEDZY Downloaded from mostwiedzy.pl

interactions before and after applying the 25 Å cut-off. AMBER with implicit water is slower compared to all versions of UNRES due to large cut-off, which results from the inability of applying the particle mesh Ewald scheme[40] to implicit water simulations. Depending on system, GROMACS is 6 – 12 times slower than the OpenMP NEWCT-9P-5D(o) version of the UNRES code. The CG MARTINI 3 code is, on the other hand, about 10 times faster than UNRES for large proteins. One of the reasons is that the UNRES energy function is more complicated than the MARTINI energy function, which has only radial pair-interaction potential and a smaller cut-off of 11 Å is applied (as opposed to the 25 Å cut-off in UNRES). Therefore, even though more centers are involved in the MARTINI 3 model, the energy function is not as expensive as in UNRES. There is a room to improve UNRES in this regard, by alternating the cut-off depending on the kinds of interacting sites. The present 25 Å cut-off has to handle both small and large sites and is, therefore, overestimated. The other reason is that MARTINI 3 benefits from the long-time development of GROMACS, thus implementing more developed algorithms, which are difficult to translate to coarse-grained models with axial symmetry such as UNRES. On the other hand, the benefit from using a more complex energy function in UNRES is that it can fold proteins in the *ab initio* mode, which is not possible with MARTINI 3, which requires imposing secondary-structure restraints.

We also compared the UNRES timings with those of OpenAWSEM[20] and the GENESIS CG model[22] presented in the references cited for similar systems. The AWSEM model has all-atom backbone and united single-center side chains, while 1 interaction site per residue, centered on the C^{α} atom is present in the GENESIS CG model. Both models implement centrosymmetric site-site interaction potential. According to the data in Ref. 20, OpenAWSEM requires 200 CPU hours per 4,000,000 MD steps with the Intel Xeon CPU E5-2650 v2 processor or 8 CPU hrs with the Nvidia V100 GPU accelerator for the 4QQW protein (3274 residues). For a protein with similar size (2 chains of 4R3O; 3,143 residues) UNRES requires 78.7 CPU hours with a single core of the Intel Xeon Processors E5 v3 @ 2,3 GHz or 6.3 hours with all 24 cores and using OpenMP (we have not yet implemented the UNRES code on GPU). The GENESIS CG model can run 2,200,000 MD steps/day for a system containing 120 copies of the DPS protein (a total of 222,360 residues) with 8 MPI

MOST WIEDZY Downloaded from mostwiedzy.pl

processes, each split into 5 threads with the Intel Xeon Gold-6148 (2.4GHz) processor[22], compared with 5Y6P (153,243 residues) 340,828 MD steps/day with UNRES run with 4MPI processes, 12 thread each (the optimal setup). Thus, UNRES is faster than OpenAWSEM but slower than the GENESIS CG code. This feature results from the presence of a greater number of centers in AWSEM (all-atom backbone) and only 1 center per residue in the GENESIS CG model compared to 2 sites with axially-symmetric potentials in UNRES. It should be noted that, as opposed to UNRES and AWSEM, the GENESIS CG model has not been used in *ab initio* folding of proteins[22].

Relationship between UNRES and all-atom time scale

In addition to faster calculations with UNRES compared to those with the all-atom model, it should be noted that the events (e.g., formation of secondary structure, folding, and docking) occur faster compared to all-atom simulations. This feature of coarse-grained simulations results from the elimination of the secondary degrees of freedom in coarse-grained approaches, which results in a large reduction of free-energy barriers to conformational transitions[28,29,89,90]. In order to estimate this speed-up, we simulated the folding of a variant of tryptophan cage (a small 20-residue protein; PDB: 2JOF), which folds and unfolds repeatedly in both UNRES and all-atom simulations. The folding of this protein was studied by Lindorff-Larsen et al.[91] by all-atom simulations with explicit water run on the ANTON supercomputer[7]. The total length of simulations was 208 μs. The average folding and unfolding times obtained from these all-atom simulations were $\tau_f = 14$ μs and $\tau_u = 3$ μs, respectively.

We ran 24 trajectories with UNRES, each lasting 0.8 μs (19.2 μs total), using the same 2 fs time step as in Ref. 91, in the Langevin dynamics mode with water viscosity scaled by the factor of 0.01 at a temperature T = 273 K, at which the ratio of folding to unfolding time was comparable to that obtained in Ref. 91. We used a variant of the UNRES force field developed in our recent work[64] by applying force matching to the NEWCT-9P scale-consistent force field, to make the forces compatible with the average forces obtained from the all-atom trajectories of the standard tryptophan cage (PDB: 1L2Y) simulated with AMBER. Therefore, the selected variant of the UNRES force seems to be most closely related to the

MOST WIEDZY Downloaded from mostwiedzy.pl

all-atom force field and can be considered its smoothed version.

We performed the same analysis as in Ref. 91 to determine the folding and unfolding time (see Supplementary Material for details), obtaining the average folding and unfolding times of $\tau_f^{UNRES} = 0.018$ μs and $\tau_u^{UNRES} = 0.004$ μs, respectively. These values are averaged over 600 folding and unfolding events. This gives the ratio of the times of folding and unfolding events in the all-atom and UNRES simulations of 778 and 750, respectively. Consequently, it can be concluded that the events that occur in a time unit of UNRES simulations require over 700 times longer all-atom simulations with the same time step. In view of the fact that the standard UNRES time step (4.89 fs) is about 2.5 times longer than that available in all-atom simulations (2 fs) and that UNRES simulations run 6 – 12 times faster than the all-atom simulations (after applying cut-off on long-range interactions in both simulation types; Figure 10), the UNRES simulations can cover at least 1000-fold longer time scale compared to the all-atom simulations. Detailed research of the relationship between the UNRES and all-atom time scale, which requires using force matching with more training proteins to produce a variant of the UNRES force field related to all-atom force fields (and, consequently, compatible with them) will be the subject of our further research.

CONCLUSIONS

Molecular simulations with coarse-grained models offer a tremendous advantage over the all-atom approaches regarding the speed of computations, owing to the presence of fewer interaction sites[4,10–18]. However, the coarse-grained models derived rigorously on the physical basis, with effective energy surfaces strictly linked to all-atom energy surfaces, pose a number of challenges to the developers, the greatest one being the non-spherical symmetry of the interacting sites and the presence of coupling terms in the expressions for effective potential energy[18]. Site anisotropy and the coupling contributions become essential with extensive coarse graining for the resulting CG model to have good predictive capacity[35]. Thus, even though the number of interacting sites is much smaller in the CG models, the energy expressions are more complex and, consequently, more expensive to evaluate compared to those of the all-atom models. Moreover, site anisotropy results in the appearance of non-

MOST WIEDZY Downloaded from mostwiedzy.pl

diagonal inertia tensors in the equations of motion. These features of coarse-grained models also make the memory management more difficult. Finally, the force fields and algorithms for all-atom MD have a much longer history than the CG ones (which often mimic their all-atom counterparts) and are, therefore, much more developed than the CG approaches both regarding the advancement of parameterization and bench-marking, parallelization, and hardware implementation[1–5].

In this work, we heavily optimized the implementation of the coarse-grained UNRES model of proteins[26,27], which includes site anisotropy and coupling terms in the effective potential energy, by bringing the expressions for the effective energy and energy gradient to the most optimal and numerically stable forms. In particular, we eliminated explicit angles and explicit dihedral angles from the equations, calculating only the scalar and cross products of the respective virtual-bond vectors that occur in the energy expressions. We implemented cut-off on long-range interactions and interaction lists, with which the computation time grows linearly with system size, as in all-atom calculations. By changing the variables, we transformed the inertia matrix to the pentadiagonal form, this resulting in reduced memory and computation cost in solving the equations of motion.

We also took advantage of both distributed- and shared-memory architectures, thus enabling efficient parallelization of the construction of the interaction list (which involves substantial communication between processes/threads), the computation of the quantities dependent on single residue that are needed in the calculations of local and coupling energy terms, as well as other tasks requiring data exchange. These computations are not as time-consuming as the site-site interaction energies, however, with high degree of parallelization of the latter those do count. Distributed-memory parallelization of pre-computing single-residue quantities is not an option because, as opposed to site-site energy evaluation, sizeable data are generated. With shared memory, the latter feature is not a problem. With shared-memory parallelization, we reached the efficiency of over 60 % and about 50 % with 24 and 48 cores, respectively, for proteins with sizes over 5,000 residues. With 24 cores, we are able to reach over 1 ns/day computing time for a system with over 100,000 residues. This is 6-12 times more compared to the highly optimized GROMACS[52] code and, apart from faster per-MD-step computations, the fact that the time step in UNRES can be about

MOST WIEDZY Downloaded from mostwiedzy.pl

2.5 times longer than in all-atom MD without violating the integration-algorithm stability also contributes to this feature. Moreover, due to the elimination of the fine-grain degrees of freedom the events occur about 1,000 faster than in all-atom simulations. Therefore, UNRES is able to reach effectively 1 μs/day with 24 cores.

Further improvements of the UNRES code involve porting the most computation-intensive parts to GPU, optimizing the cut-off to make it dependent on site size, and optimizing the time-split algorithm of numerical solution of the equations of motion, which was introduced in our earlier work[85], which makes it possible to use longer time steps. This work is underway in our laboratory.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Centre, grants UMO-2021/40/Q-/ST4/00035 (to AL), UMO-2017/26/M/ST4/00044 (to CC), and UMO-2017/27/B/ST4/00926 (to AKS). For the purpose of Open Access, the author has applied a CC-BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.
This work was partially supported by Pomorskie Voivodeship Regional Operational Program for 2014-2020, grant RPPM.01.02.00-22-0001/17, CK STOS.
This work was partially supported by the infrastructure provided by European Funds - Smart Growth in a grant: EuroHPC PL - National Supercomputing Infrastructure for EuroHPC, POIR.04.0200-00-D014/20-00.
Computational resources were provided by the Centre of Informatics – Tricity Academic Supercomputer & Network (CI TASK) in Gdańsk.

DATA AVAILABILITY STATEMENT

All scalability data are in Table S1 of the Supporting Information. The optimized UNRES source code, installation instructions, and benchmarks are available from https://projects.task.gda.pl/eurohpcpl-public/unres and from https://unres.pl under the GPL v3 license.

MOST WIEDZY Downloaded from mostwiedzy.pl

References

1. D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications* (Academic Press, New York, 2000).

2. S. A. Adcock and J. A. McCammon, Chem. Rev. **106**, 1589 (2006).

3. H. A. Scheraga, M. Khalili, and A. Liwo, Annu. Rev. Phys. Chem. **58**, 57 (2007).

4. G. J. A. Sevink, J. A. Liwo, P. Asinari, D. MacKernan, G. Milano, and I. Pagonabarraga, J. Chem. Phys. **153**, 100901 (2020).

5. C. L. Brooks III, D. A. Case, S. Plimpton, B. Roux, D. van derSpoel, and E. Tajkhorshid, J. Chem. Phys. **154**, 100401 (2021).

6. M. Maritan, L. Autin, J. Karr, M. W. Covert, A. J. Olson, and D. S. Goodsell, J. Mol. Biol. **434**, 167351 (2022).

7. D. E. Shaw, M. M. Deneroff, R. O. Dror, J. S. Kuskin, R. H. Larson, J. K. Salmon, C. Young, B. Batson, K. J. Bowers, J. C. Chao, et al., Commun. ACM **51**, 91 (2008).

8. K. Lindorff-Larsen, N. Trbovic, P. Maragakis, S. Piana, and D. E. Shaw, J. Am. Chem. Soc. **134**, 3787 (2012).

9. D. S. D. Larsson, L. Liljas, and D. van der Spoel, PLoS Comput. Biol. **8**, e1002502 (2012).

10. A. Kolinski, Acta Biochim. Pol. **51**, 349 (2004).

11. G. Voth, in *Coarse-Graining of Condensed Phase and Biomolecular Systems*, edited by G. Voth (CRC Press, Taylor & Francis Group, 2008), chap. 1, pp. 1–4, 1st ed.

12. V. Tozzini, Quart. Rev. Biophys. **43**, 333 (2010).

13. S. Kmiecik, D. Gront, M. Koliński, L. Wieteska, A. Dawid, and A. Koliński, Chem. Rev. **116**, 7898 (2016).

MOST WIEDZY Downloaded from mostwiedzy.pl

14. G. A. Papoian, *Coarse-Grained Modeling of Biomolecules* (CRC Press, 2017), ISBN 9781466576063.

15. N. Singh and W. Li, Int. J. Mol. Sci. **20**, 3774 (2019).

16. T. Sun, V. Minhas, N. Korolev, A. Mirzoev, A. P. Lyubartsev, and L. Nordensiöld, Front. Biomol. Sci. **8**, 645527 (2021).

17. M. Giulini, M. Rigoli, G. Mattioti, R. Menichetti, T. Tarenzi, R. Florentini, and R. Potestio, Front. Mol. Biosci. **8**, 676976 (2021).

18. A. Liwo, A. K. Sieradzan, A. S. Karczyńska, E. A. Lubecka, S. A. Samsonov, C. Czaplewski, P. Krupa, and M. Mozolewska, in *Practical Aspects of Computational Chemistry V*, edited by M. S. Leszczynski (Springer, 2021), chap. 2, pp. 31–69.

19. A. Davtyan, N. P. Schafer, W. Zheng, C. Clementi, P. G. Wolynes, and G. A. Papoian, J. Phys. Chem. B **116**, 8494 (2012).

20. W. Lu, C. Bueno, N. P. Schafer, J. Moller, S. Jin, X. Chen, M. Chen, X. Gu, A. Davtyan, J. J. de Pablo, et al., PLoS Comput. Biol. **17**, e1008308 (2021).

21. J. Jung, T. Mori, C. Kobayashi, Y. Matsunaga, T. Yoda, M. Feig, and Y. Sugita, WIREs Comput. Mol. Sci. **5**, 310 (2015).

22. C. Tan, J. Jung, C. Kobayashi, D. U. La Torre, S. Takada, and Y. Sugita, PLoS Comput. Biol. **18**, e1009578 (2022).

23. S. J. Marrink and D. P. Tieleman, Chem. Soc. Rev. **42**, 6801 (2013).

24. S. J. Marrink, L. Monticelli, M. N. Melo, R. Alessandri, D. P. Tieleman, and P. C. T. Souza, WIREs Comput. Mol. Sci. **Early View**, e1620 (2022).

25. L. Darré, M. R. Machado, A. F. Brandner, H. C. González, S. Ferreira, and S. Pantano, J. Chem. Theory Comput. **11**, 723 (2015).

26. A. Liwo, M. Baranowski, C. Czaplewski, E. Gołaś, Y. He, D. Jagieła, P. Krupa, M. Maciejczyk, M. Makowski, M. A. Mozolewska, et al., J. Mol. Model. **20**, 2306 (2014).

MOST WIEDZY Downloaded from mostwiedzy.pl

27. A. K. Sieradzan, C. Czaplewski, P. Krupa, M. A. Mozolewska, A. Karczyńska, A. Lipska, E. A. Lubecka, E. Gołaś, T. Wirecki, M. Makowski, et al., in *Methods in Molecular Biology*, edited by V. Muñoz (Springer New York, 2022), vol. 2376, chap. 23, pp. 399–416.

28. M. Khalili, A. Liwo, A. Jagielska, and H. A. Scheraga, J. Phys. Chem. B **109**, 13798 (2005).

29. V. Klippenstein, M. Tripathy, G. Jung, F. Schmid, and N. F. A. van der Vegt, J. Phys. Chem. B **125**, 4931 (2021).

30. Y. Han, J. Jin, and G. A. Voth, J. Chem. Phys. **154**, 084122 (2021).

31. B. Berne and P. Pechukas, J. Chem. Phys. **56**, 4213 (1972).

32. J. G. Gay and B. J. Berne, J. Chem. Phys. **74**, 3316 (1981).

33. A. Liwo, C. Czaplewski, J. Pillardy, and H. A. Scheraga, J. Chem. Phys. **115**, 2323 (2001).

34. R. Kubo, J. Phys. Soc. Japan **17**, 1100 (1962).

35. A. K. Sieradzan, M. Makowski, A. Augustynowicz, and A. Liwo, J. Chem. Phys. **146**, 124106 (2017).

36. A. Liwo, C. Czaplewski, A. K. Sieradzan, E. A. Lubecka, A. G. Lipska, Ł. Golon, A. Karczyńka, P. Krupa, M. A. Mozolewska, M. Makowski, et al., in *Progress in molecular biology and translational science. Computational Approaches for Understanding Dynamical Systems: Protein Folding and Assembly*, edited by B. Strodel and B. Barz (Academic Press, London, 2020), vol. 170, chap. 2, pp. 73–122.

37. A. Liwo, A. K. Sieradzan, A. G. Lipska, C. Czaplewski, I. Joung, W. Żmudzińska, A. Hałabis, and S. Ołdziej, J. Chem. Phys. **150**, 155104 (2019).

38. Y. He, M. Maciejczyk, S. Ołdziej, H. A. Scheraga, and A. Liwo, Phys. Rev. Lett. **110**, 098101 (2013).

MOST WIEDZY Downloaded from mostwiedzy.pl

39. M. Khalili, A. Liwo, F. Rakowski, P. Grochowski, and H. A. Scheraga, J. Phys. Chem. B **109**, 13785 (2005).

40. M. Crowley, T. Darden, T. Cheatham, and D. Deerfield, J. Supercomput. **11**, 255 (1997).

41. S. J. Marrink, H. J. Risselada, S. Yefimov, D. P. Tieleman, and A. H. de Vries, J. Phys. Chem. B **111**, 7812 (2007).

42. D. H. de Jong, S. Baoukina, H. I. Ingólfsson, and S. J. Marrink, Comput. Phys. Commun. **199**, 1 (2016).

43. A. K. Sieradzan, J. Comput. Chem. **36**, 940 (2015).

44. P. Czarnul, *Parallel Programming for Modern High Performance Computing Systems* (CRC Press, Taylor & Francis Group, 2018).

45. W. Gropp, E. Lusk, and A. Skjellum, *Parallel libraries* (The MIT Press, Cambridge Massachussetts, London, 1999), chap. 6, pp. 157–193, 2nd ed.

46. T. Mattson, Y. He, and A. Koniges, *The OpenMP Common Core: Making OpenMP Simple Again*, Scientific and Engineering Computation (MIT Press, 2019), ISBN 9780262538862.

47. B. Nichols, D. Buttlar, and J. P. Farrell, *Pthreads programming - a POSIX standard for better multiprocessing.* (O'Reilly, 1996), ISBN 978-1-56592-115-3.

48. A. Munshi, B. Gaster, T. Mattson, J. Fung, and D. Ginsburg, *OpenCL Programming Guide*, Graphics programming (Addison-Wesley, 2012), ISBN 9780321749642.

49. J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming* (Addison-Wesley Professional, 2010), 1st ed., ISBN 0131387685.

50. G. Juckeland and S. Chandrasekaran, *OpenACC for Programmers: Concepts and Strategies* (Pearson Education, Inc., 2017), ISBN 978-0134694283.

51. R. Salomon-Ferrer, D. A. Case, and R. C. Walker, WIREs Comput. Mol. Sci. **3**, 198 (2013).

MOST WIEDZY Downloaded from mostwiedzy.pl

52. S. Páll, A. Zhmurov, P. Bauer, M. Abraham, M. Lundborg, A. Gray, B. Hess, and E. Lindahl, J. Chem. Phys. **153**, 134110 (2020).

53. S. Plimpton, J. Comput. Phys. **117**, 1 (1995).

54. K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, et al., ACM/IEEE SC 2006 Conference (SC.06) pp. 43–43 (2006).

55. J. A. Rackers, Z. Wang, C. Lu, M. L. Laury, L. Lagardere, M. J. Schnieders, J.-P. Piquemal, P. Ren, and J. W. Ponder, J. Chem. Theory Comput. **14**, 5273 (2018).

56. A. Liwo, S. Ołdziej, C. Czaplewski, D. S. Kleinerman, P. Blood, and H. A. Scheraga, J. Chem. Theory Comput. **6**, 583 (2010).

57. E. Lubecka, A. Sieradzan, C. Czaplewski, P. Krupa, and A. Liwo, Supercomput. Front. Innov. **5**, 63 (2018).

58. A. Liwo, S. Ołdziej, M. R. Pincus, R. J. Wawak, S. Rackovsky, and H. A. Scheraga, J. Comput. Chem. **18**, 849 (1997).

59. U. Kozłowska, A. Liwo, and H. A. Scheraga, J. Comput. Chem. **31**, 1143 (2010).

60. U. Kozłowska, G. G. Maisuradze, A. Liwo, and H. A. Scheraga, J. Comput. Chem. **31**, 1154 (2010).

61. M. Chinchio, C. Czaplewski, A. Liwo, S. Ołdziej, and H. A. Scheraga, J. Chem. Theory Comput. **3**, 1236 (2007).

62. P. Krupa, A. K. Sieradzan, M. A. Mozolewska, H. Li, A. Liwo, and H. A. Scheraga, J. Chem. Theory Comput. **13**, 5721 (2017).

63. A. Liwo, M. Khalili, C. Czaplewski, S. Kalinowski, S. Ołdziej, K. Wachucik, and H. A. Scheraga, J. Phys. Chem. B **111**, 260 (2007).

64. A. Liwo and C. Czaplewski, J. Chem. Phys. **152**, 054902 (2020).

MOST WIEDZY Downloaded from mostwiedzy.pl

65. H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, J. Chem. Phys. **81**, 3684 (1984).

66. W. G. Hoover, Phys. Rev. A **31**, 1695 (1985).

67. S. Nosé, J. Phys. Soc. Jpn. **70**, 75 (2001).

68. D. S. Kleinerman, C. Czaplewski, A. Liwo, and H. A. Scheraga, J. Chem. Phys. **128**, 245103 (2008).

69. W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson, J. Chem. Phys. **76**, 637 (1982).

70. G. Engeln-Müllges, F. Uhlig, M. Schon, and F. Uhlig, *Numerical Algorithms with Fortran* (Springer-Verlag Berlin and Heidelberg, 2013).

71. D. C. Lui and J. Nocedal, Math. Program. **45**, 503 (1989).

72. D. M. Gay, ACM Trans. Math. Software **9**, 503 (1983).

73. U. H. E. Hansmann and Y. Okamoto, Physica A **212**, 415 (1994).

74. Y. M. Rhee and V. S. Pande, Biophys. J. **84**, 775 (2003).

75. C. Czaplewski, S. Kalinowski, A. Liwo, and H. A. Scheraga, J. Chem. Theory Comput. **5**, 627 (2009).

76. A. Liwo, M. R. Pincus, R. J. Wawak, S. Rackovsky, and H. A. Scheraga, Protein Sci. **2**, 1715 (1993).

77. A. Liwo, S. Ołdziej, C. Czaplewski, U. Kozłowska, and H. A. Scheraga, J. Phys. Chem. B **108**, 9421 (2004).

78. M. Tuckerman, B. J. Berne, and G. J. Martyna, J. Chem. Phys. **97**, 1990 (1992).

79. B. Quentrec and C. Brot, J. Comput. Phys. **13**, 430 (1973).

80. R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (Taylor & Francis, Inc., USA, 1988), ISBN 0852743920.

MOST WIEDZY Downloaded from mostwiedzy.pl

81. S. M. Thompson, CCP5 Quaterly p. 20 (1983).

82. P. Gonnet, J. Comput. Chem. **33**, 76 (2012).

83. A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, et al., Comput. Phys. Commun. **271**, 108171 (2022).

84. J. Reinders, *VTune Performance Analyzer Essentials: Measurement and Tuning Techniques for Software Developers* (Intel Press, 2005), ISBN 0974364959.

85. F. Rakowski, P. Grochowski, B. Lesyng, A. Liwo, and H. A. Scheraga, J. Chem. Phys. **125**, 204107 (2006).

86. A. Antoniak, I. Biskupek, K. K. Bojarski, C. Czaplewski, A. Giełdoń, M. Kogut, M. M. Kogut, P. Krupa, A. G. Lipska, A. Liwo, et al., J. Mol. Graph. Model. **108**, 108008 (2021).

87. J. Mongan, C. Simmerling, J. A. McCammon, D. A. Case, and A. Onufriev, J. Chem. Theory Comput. **3**, 156 (2007).

88. H. Huang and C. Simmerling, J. Chem. Theory Comput. **14**, 5797 (2018).

89. R. Zwanzig, *Nonequilibrium statistical mechanics* (Oxford University Press, New York, 2001).

90. T. Kinjo and S. Hyodo, Phys. Rev. E **75**, 051109 (2007).

91. K. Lindorff-Larsen, S. Piana, R. O. Dror, and D. E. Shaw, Science **334**, 517 (2011).

MOST WIEDZY Downloaded from mostwiedzy.pl

Figure 1: A scheme of polypeptide-chain representation in the UNRES model. The variables are the Cartesian coordinates of the α-carbon (C^α) atoms and those of side-chain centers (SC). The interaction sites are the peptide groups (p) located halfway between the consecutive C^αs and the side chains. These sites interact via axially-symmetric and not centrosymmetric potentials, the axes being the $C^\alpha \cdots C^\alpha$ (**dC**) and $C^\alpha \cdots SC$ (**dX**) virtual-bond vectors. It should be noted that the coordinates of the peptide-group centers need to be calculated from the C^α coordinates, which implies a non-diagonal inertia matrix of the system. The backbone-virtual-bond (θ_i) and the backbone-virtual-bond-dihedral angle (γ_i) ascribed to the ith residue are also indicated.

Figure 2: Illustration of assignment of two selected particles (labelled A and B, respectively) to grid cells neighboring their home grid cell. For clarity, only two dimensions are shown and the grid is equal in the parallel and perpendicular direction. The size of the grid is equal to the extended cut-off adjusted to divide each side of the simulation box into equal segments. A circle with radius equal to grid dimension is drawn around each particle. Particle B has all neighbor cells within the cut-off range and, consequently, is assigned to its home cell and to all neighboring cells. Conversely, the circle around particle A does not extend to the top-right cell (filled gray) and, therefore, this particle is beyond the cut-off distance from any particle of the gray cell. Consequently, particle B is not assigned to the list of particles of the gray cell.

MOST WIEDZY Downloaded from mostwiedzy.pl

Figure 3: (A) General parallelization scheme of UNRES. The upper diagram shows the MPI-level parallelization, with n coarse-grain (CG) tasks, each spanning m fine-grain (FG) tasks. The CG MPI processes (with the assigned CG_COMM communicator) supervise the processing of an MD trajectory or an energy evaluation/minimization job. The master process (with rank 0 in MPI_COMM_WORLD), doing its share of computation, governs the whole parallel job. Each fine-grain task has a dedicated communicator (FG_COMM). The CG task is simultaneously the master of the respective FG task. All FG processes (both master and slave) do energy and energy-gradient evaluation; the master process additionally governs the calculations (gathering the contributions from the slave processors, doing MD steps or energy minimization, doing I/O operations). The numbers in parentheses show the ranks of the CG/FG processes; the first number is the absolute rank in MPI_COMM_WORLD, the second number is the rank in CG_COMM or FG_COMM, respectively. The bottom diagram shows the division of a given FG process into OpenMP threads. (B) A scheme of partitioning of the blocks of interactions between FG processes and OpenMP threads. The pairs of interacting particles are stored in NG groups, each with the same first (source) particle in the pair. For each group, the index of the source particle (i), and the indices of the first ($j^{(s)}$) and the last ($j^{(e)}$) particle interacting with it are stored. The groups are combined into blocks assigned to the subsequent FG task ($\mathrm{FG}_1 \ldots \mathrm{FG}_m$). The consecutive groups are assigned to tasks subject to the load-balance condition. The blocks corresponding to a process can further be split between different OpenMP threads.

MOST WIEDZY Downloaded from mostwiedzy.pl

Figure 4: An example of an interaction scheduler in action. A list of interaction ranges is passed through the scheduler. In this example, these are the interactions between particle 2 and particles $4, 5, 6, 7, 9, 10, 11$. Assume that the distance between particles 4 and 10 is between $r_{cut} - \lambda$ and r_{cut} so that the smoothing function [Equation (56)] has to be triggered. The scheduler will handle the interactions in order, assigning the pair of $(2, 4)$ to the smoothing queue first. Next, the pairs $(2, 5)$, $(2, 6)$, $(2, 7)$ will be put in a sequential queue. The next pair, $(2, 9)$, breaks the sequence and will be moved to the non-sequential queue. Next, pair $(2, 10)$ will be put into the smoothing queue and pair $(2, 11)$ into the non-sequential one. After scheduling, the interactions within each queue can be computed with instruction level-parallelism without any conditional branches.

Figure 5: Plots of the speedups (A and C) and efficiencies (B and D) for canonical single-trajectory UNRES/MD simulations using MPI libraries (A and B) and OpenMP or hybrid OpenMP/MPI parallelization (C and D) for proteins with different sizes. The graphs are labelled with protein codes (see Table 1). For 24 cores (single node with two CPUs, 12 cores each) the graphs corresponding to OpenMP (panel B) are labelled with 1×24 and hybrid OpenMP/MPI are labelled with 2×12, indicating two MPI processes each split into 12 OpenMP threads. For 48 cores (two nodes) the two versions of hybrid OpenMP/MPI setups are labelled with 2×24 (24 OpenMP threads on each node) and 4×12 (two MPI processes each split into 12 OpenMP threads on each node). The optimized UNRES-9P-5D(o) code was used. Each calculation was repeated 3 times and the average values are shown. The error bars are smaller than symbol sizes.

MOST WIEDZY Downloaded from mostwiedzy.pl

Figure 6: Bar plots of the ratios of the contributions to the non-setup time and to the total non-setup time of single 10,000-step canonical MD runs carried out with 24 cores and three MPI/OpenMP modes. (A) $t(24\times1)/t(1\times24)$, (B) $t(24\times1)/t(2\times12)$, (C) $t(2\times12)/t(1\times24)$, where $t(24\times1)$, $t(1\times24)$, and $t(2\times12)$ denote the times corresponding to the runs carried out with 24 MPI processes, 24 OpenMP threads, and 2 MPI processes, each split into 12 OpenMP threads, respectively. The respective contributions are labelled as follows: 'lists': list construction, 'ene prep/sum': preparation for energy and energy-gradient calculations (which include putting the particles into the simulation box, coordinate and energy-term weight broadcast to slave processes, and zeroing the gradient components) and the reduction of energy contributions from different processes, 'SC-SC': the evaluation of $U_{SC_iSC_j}$ energy/gradient components, 'SC-p': the evaluation of $U_{SC_ip_j}$ energy/gradient components, 'p-p+corr': the evaluation of $U_{p_ip_i}$ and $U^{(3)}_{corr}$ energy/gradient components, 'local': the evaluation of local-energy/gradient components (U_{bond}, U_b, U_{tor}, and U_{rot}), 'grad/lagr': summation of gradient components and calculating accelerations.

Figure 7: Stacked-bar plots of the contributions to the non-setup time for the runs with 24 MPI processes (24×1), with 2 MPI processes, each split into 12 OpenMP thread (2×12), and 24 OpenMP threads (1×24). The contributions are labelled as in Figure 6. Additionally, 'VV' stands for updating the coordinates and velocities in velocity-Verlet steps and computing normalized virtual-bond vectors.

Figure 8: Bar plot of nonsetup times per 10,000 MD steps using OpenMP and hybrid OpenMP/MPI parallelization in number of residues. As in Figure 5B, the label $n\times m$ stands for a calculation carried out with n MPI processes, each split into m OpenMP threads. Each calculation was repeated 3 times and the average values with error bars (from minimum to maximum) are shown. The error bars are shown on the top of each bar. The optimized UNRES-9P-5D(o) code was used.

MOST WIEDZY Downloaded from mostwiedzy.pl

Figure 9: Plots of nonsetup times (A, C, and E) and efficiencies (B, D, and F) in 10,000-step MREMD simulations, for the 5HKQ (263 residues; A and B), 5SY1 (729 residues; C and D), and H1081 (15,200 residues; E and F) proteins. The number of trajectories ranges from 1 (reference MD simulation) to 96. Different points correspond to calculations carried out with 1, 4, 12, and 24 OpenMP threads per core and 2 MPI processes split into 12 OpenMP threads each, respectively, as indicated by the respective labels. The maximum number of cores used was 48×96=4608 (for 96 MREMD trajectories of H1081, each run with 48 cores). Each run was repeated 3 times and the average values and error bars (from minimum to maximum) are shown for each point.

MOST WIEDZY Downloaded from mostwiedzy.pl

Figure 10: Plots of (A) nonsetup times per 1 MD step and (B) the per-day (24 hrs of non-setup time) trajectory length (ns/day) in number of residues obtained with a single 24-core processor in UNRES and all-atom calculations of proteins with various sizes. The logarithmic scale is used on both axes. The respective graphs are labelled according to the types of calculations as follows: NEWCT-9P: the old scale-consistent version of UNRES prior to this work (MPI only, no cut-off on long-range interactions), NEWCT-9P-5D: the preliminary version of UNRES with variables changed to C^{α} and SC coordinates described in this work prior to code optimization (no cut-off and 25 Å cut-off, MPI only), NEWCT-9P-5D(o): the optimized NEWCT-9P-5D code (no cut-off and 25 Å cut-off, MPI only or MPI and OpenMP), MARTINI 3, (cut-off 11 Å, explicit coarse-grained water), GROMACS: explicit-water all-atom-simulation times simulations with cut-off 8 Å using GROMACS 2020[52], AMBER: all-atom simulations with AMBER 20[51] with implicit water, GBSA model[87,88], 25 Å cut-off. The speedups and per-day times of the runs carried out with no cut-off (dashed lines) exhibit the quadratic ($speedup \propto number\_of\_residues^2$) and inverse-quadratic dependence on number of residues ($traj\_length\_per\_day \propto 1/number\_of\_residues^2$), respectively, while, for larger proteins, those of the runs carried out with cut-off (solid lines) are proportional to the number of residues ($speedup \propto number\_of\_residues$) or to the inverse of the number of residues ($traj\_length\_per\_day \propto 1/number\_of\_residues$), respectively. The simulations with UNRES were run with a 5 fs time step except the ones where the 10 fs time step is indicated explicitly at the respective graph and those with MARTINI 3 were carried out with a 20 fs time step. All-atom simulations were carried out with a 2 fs time step.

MOST WIEDZY Downloaded from mostwiedzy.pl

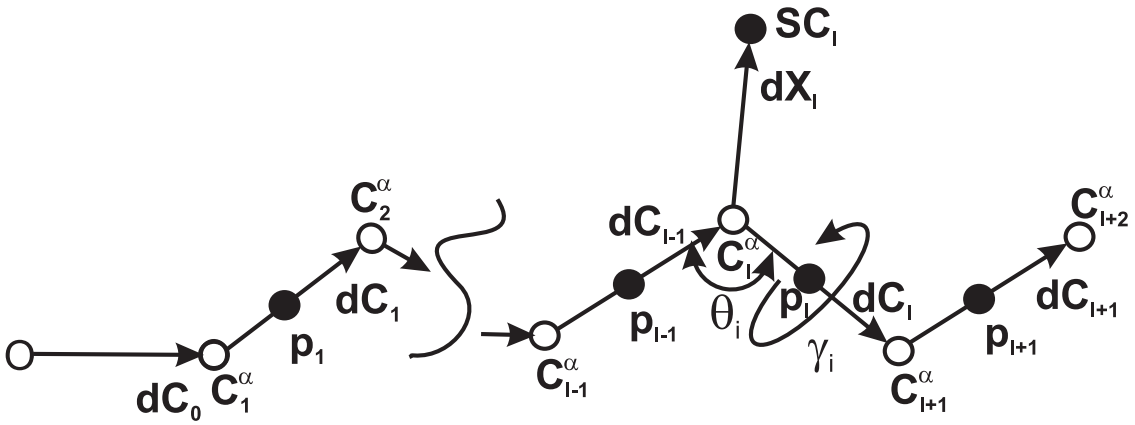


Figure 1
A. K. Sieradzan, J. Sans-Duñó, E. A. Lubecka, C. Czaplewski, A. G. Lipska, H. Leszczyński, K. M. Ocetkiewicz, J. Proficz, P. Czarnul, H. Krawczyk, A. Liwo
J. Comput. Chem.

MOST WIEDZY Downloaded from mostwiedzy.pl

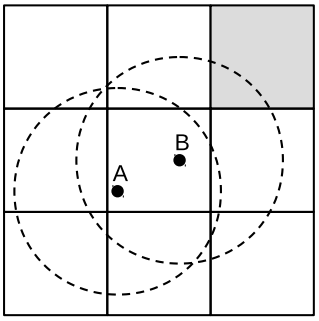


Figure 2
A. K. Sieradzan, J. Sans-Duñó, E. A. Lubecka, C. Czaplewski, A. G. Lipska, H. Leszczyński, K. M. Ocetkiewicz, J. Proficz, P. Czarnul, H. Krawczyk, A. Liwo
J. Comput. Chem.

MOST WIEDZY Downloaded from mostwiedzy.pl

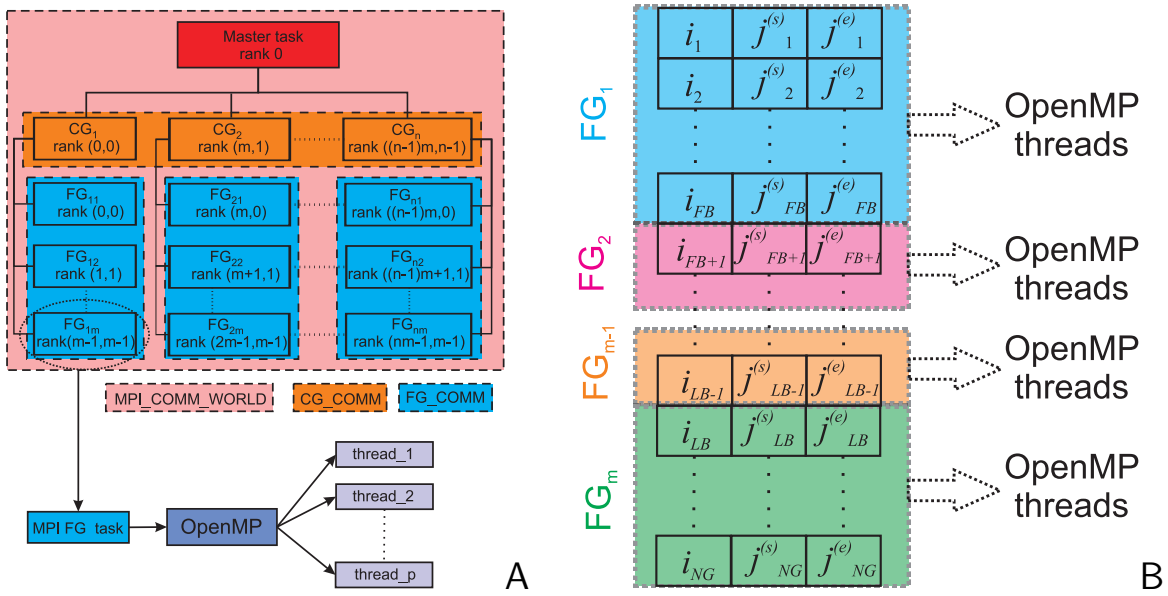


Figure 3
A. K. Sieradzan, J. Sans-Duñó, E. A. Lubecka, C. Czaplewski, A. G. Lipska, H. Leszczyński, K. M. Ocetkiewicz, J. Proficz, P. Czarnul, H. Krawczyk, A. Liwo
J. Comput. Chem.

MOST WIEDZY Downloaded from mostwiedzy.pl

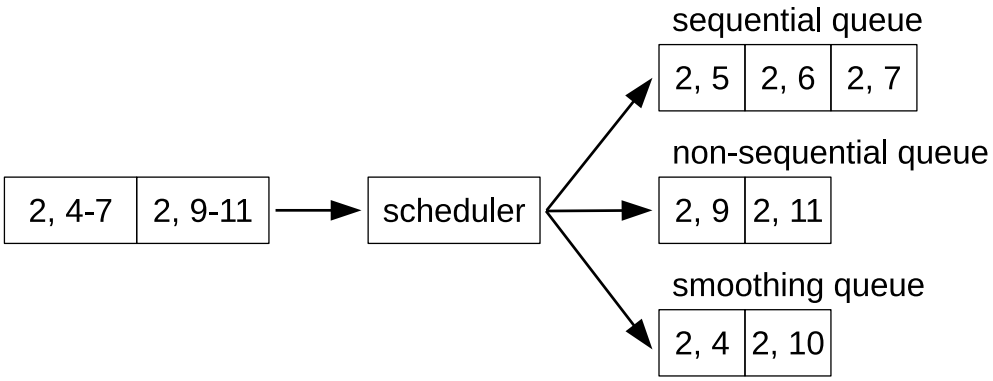


Figure 4
A. K. Sieradzan, J. Sans-Duñó, E. A. Lubecka, C. Czaplewski, A. G. Lipska, H. Leszczyński, K. M. Ocetkiewicz, J. Proficz, P. Czarnul, H. Krawczyk, A. Liwo
J. Comput. Chem.

MOST WIEDZY Downloaded from mostwiedzy.pl

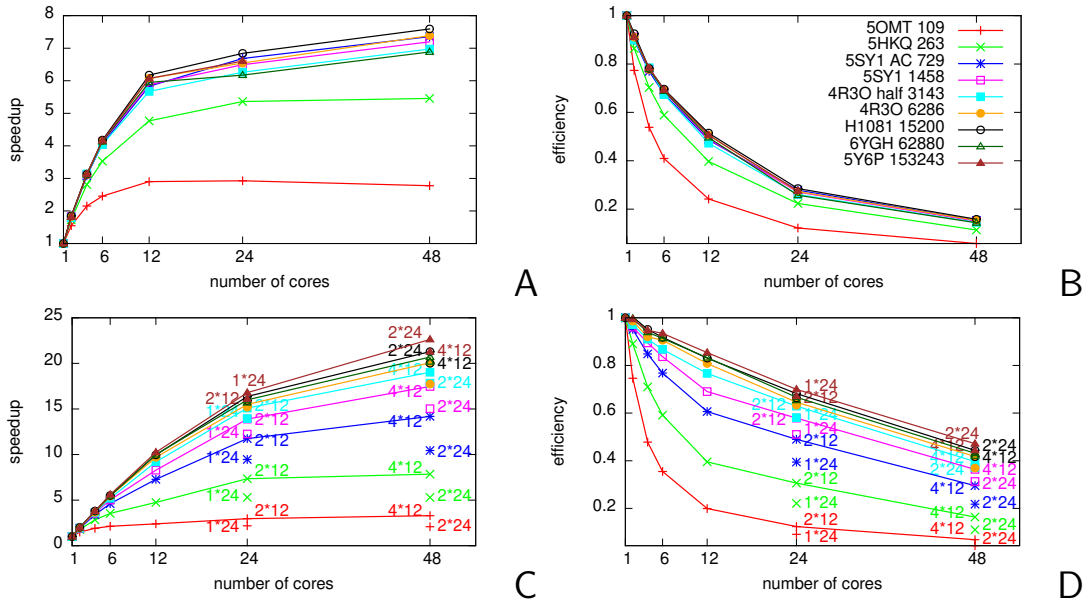


Figure 5
A. K. Sieradzan, J. Sans-Duñó, E. A. Lubecka, C. Czaplewski, A. G. Lipska, H. Leszczyński, K. M. Ocetkiewicz, J. Proficz, P. Czarnul, H. Krawczyk, A. Liwo
J. Comput. Chem.

MOST WIEDZY Downloaded from mostwiedzy.pl

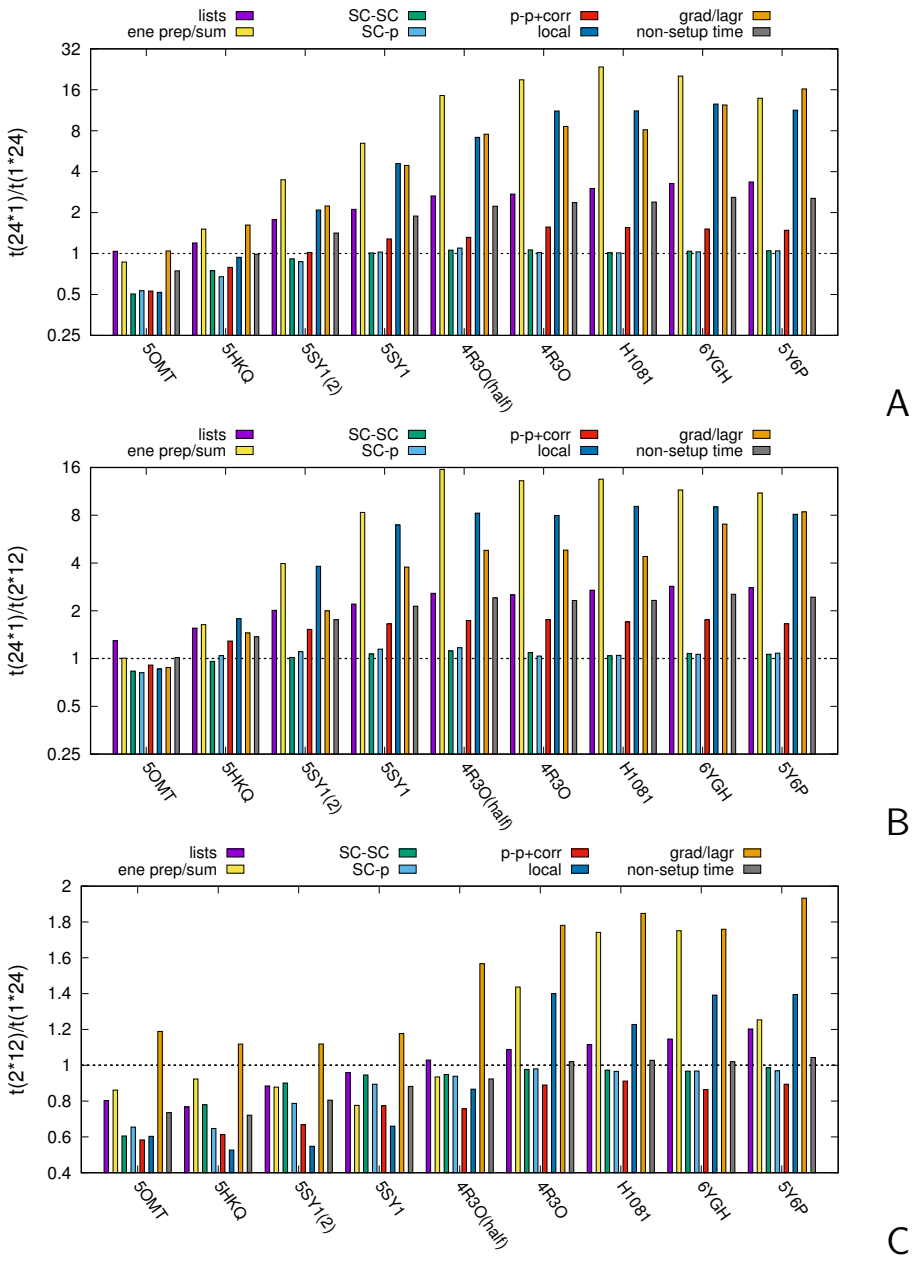


Figure 6
A. K. Sieradzan, J. Sans-Duñó, E. A. Lubecka, C. Czaplewski, A. G. Lipska, H. Leszczyński, K. M. Ocetkiewicz, J. Proficz, P. Czarnul, H. Krawczyk, A. Liwo
J. Comput. Chem.

MOST WIEDZY Downloaded from mostwiedzy.pl

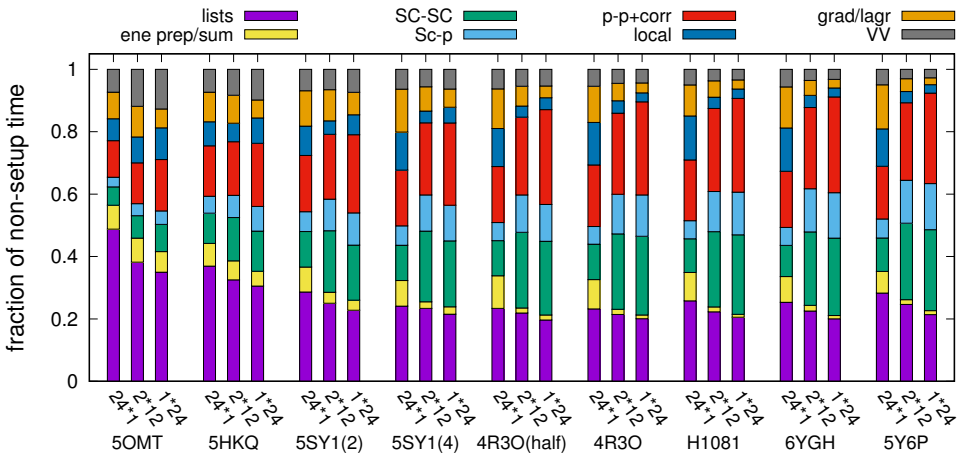


Figure 7
A. K. Sieradzan, J. Sans-Duñó, E. A. Lubecka, C. Czaplewski, A. G. Lipska, H. Leszczyński, K. M. Ocetkiewicz, J. Proficz, P. Czarnul, H. Krawczyk, A. Liwo
J. Comput. Chem.

MOST WIEDZY Downloaded from mostwiedzy.pl

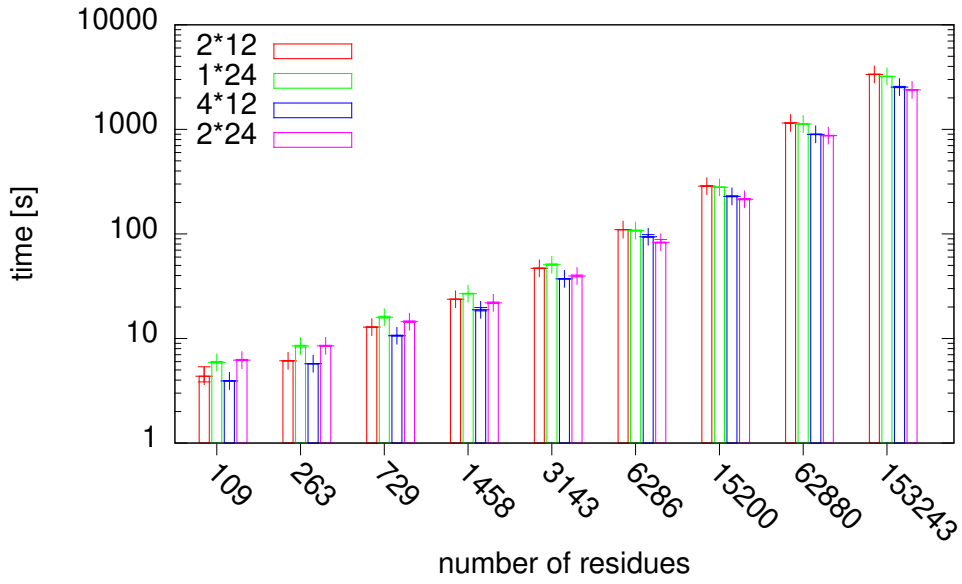


Figure 8
A. K. Sieradzan, J. Sans-Duñó, E. A. Lubecka, C. Czaplewski, A. G. Lipska, H. Leszczyński, K. M. Ocetkiewicz, J. Proficz, P. Czarnul, H. Krawczyk, A. Liwo
J. Comput. Chem.

MOST WIEDZY Downloaded from mostwiedzy.pl

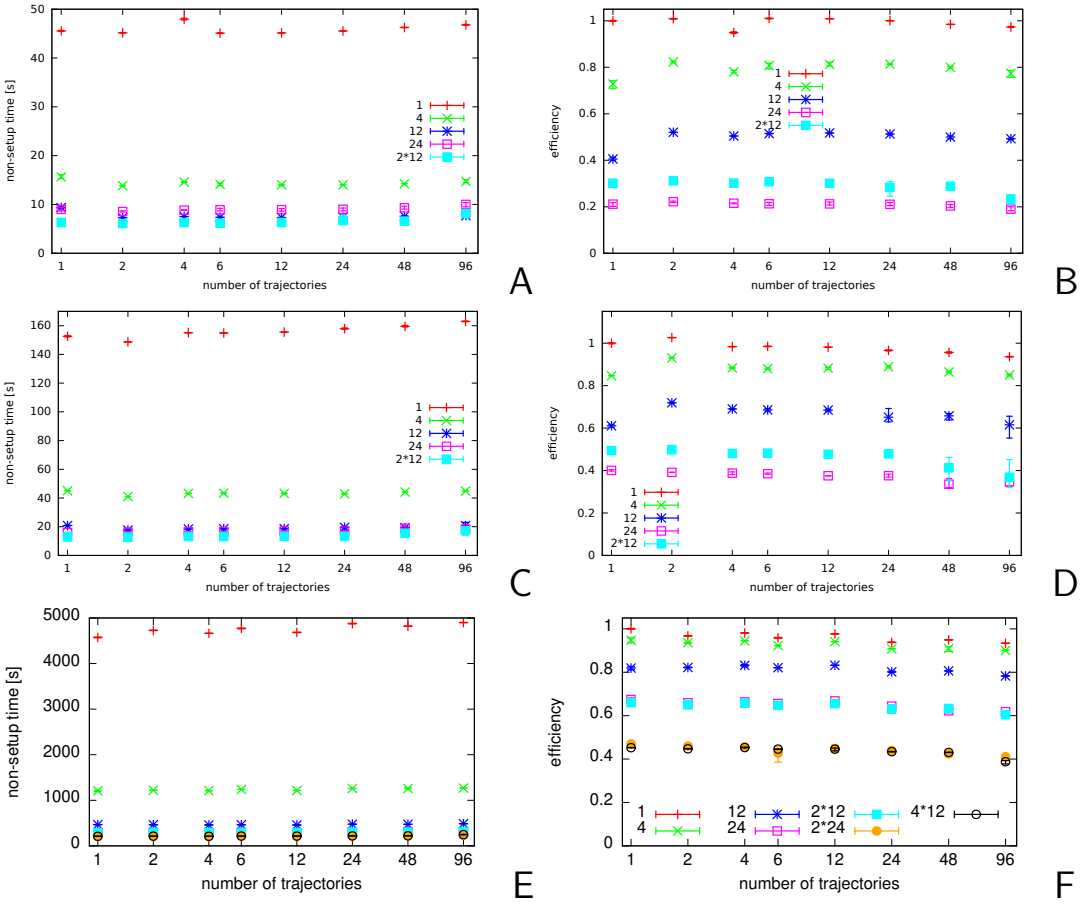


Figure 9
A. K. Sieradzan, J. Sans-Duñó, E. A. Lubecka, C. Czaplewski, A. G. Lipska, H. Leszczyński, K. M. Ocetkiewicz, J. Proficz, P. Czarnul, H. Krawczyk, A. Liwo
J. Comput. Chem.

MOST WIEDZY Downloaded from mostwiedzy.pl

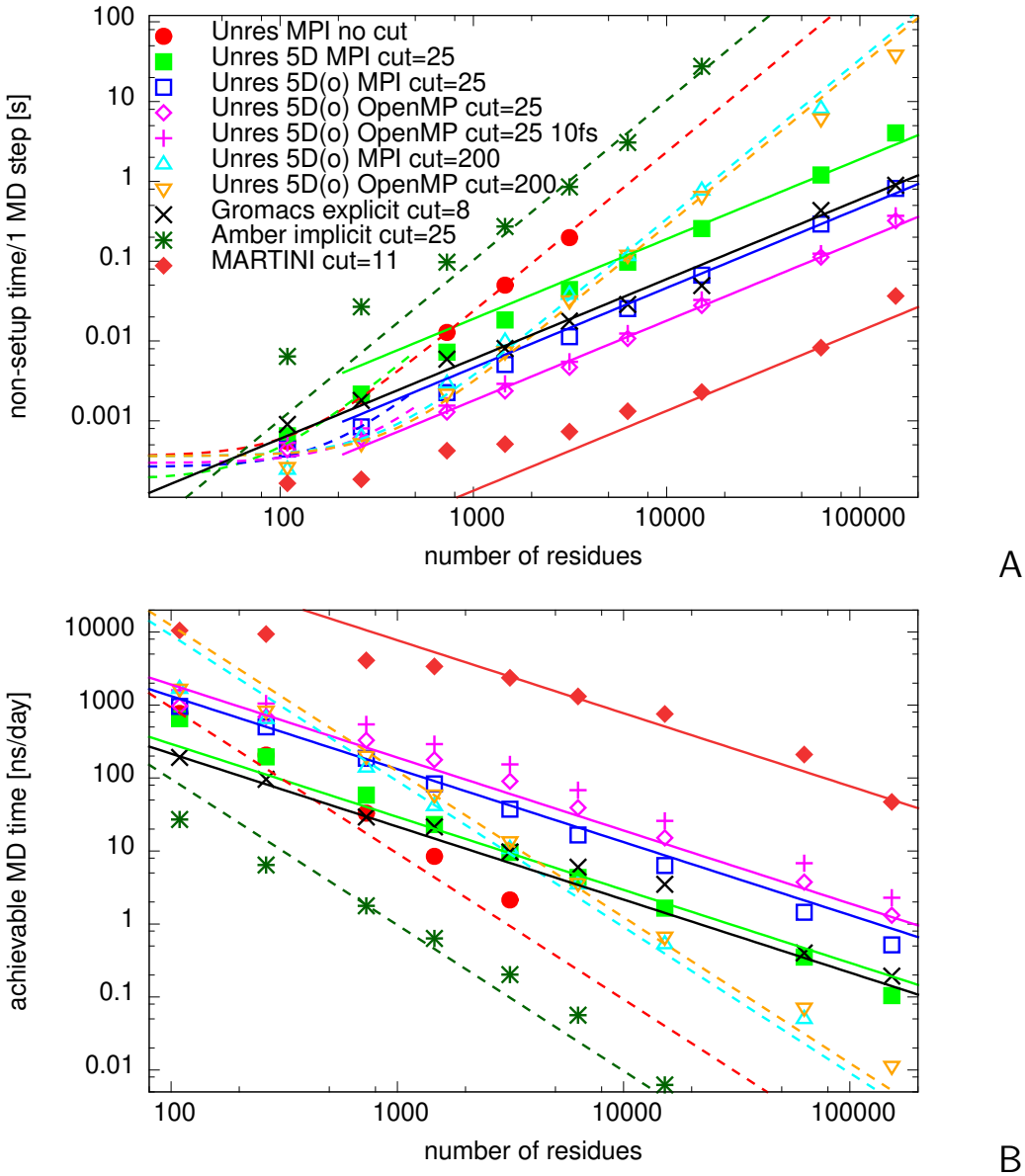


Figure 10
A. K. Sieradzan, J. Sans-Duñó, E. A. Lubecka, C. Czaplewski, A. G. Lipska, H. Leszczyński, K. M. Ocetkiewicz, J. Proficz, P. Czarnul, H. Krawczyk, A. Liwo
J. Comput. Chem.

MOST WIEDZY Downloaded from mostwiedzy.pl

Table 1: The proteins used for benchmark

Description	PDB code	number of residues	number of atoms with implicit solvent	number of atoms with explicit solvent
endonuclease NucB	5OMT	109	1640	15172
CDI complex, 2 chains	5HKQ	263	4114	32762
STRA6 receptor, 2 chains	5SY1	729	11733	114739
STRA6 receptor, 4 chains	5SY1	1458	23466	160334
human proteasome 20S, 14 chains	4R3O	3143	48913	344178
human proteasome 20S, 28 chains	4R3O	6286	97826	568791
bacterial arginine decarboxylase, H1081 CASP14 target, 20 chains	-	15200	237960	959732
Duck hepatitis B virus, 240 chains	6YGH	62880	1029120	5904390
the phycobilisome from the red alga, 862 chains	5Y6P	153243	2283236	15702431

MOST WIEDZY Downloaded from mostwiedzy.pl

Optimization of parallel implementation of UNRES package for coarse-grained simulations to treat large proteins

Adam K. Sieradzan[*†], Jordi Sans-Duñó[‡], Emilia A. Lubecka[§], Cezary Czaplewski[*†], Agnieszka G. Lipska[*†], Henryk Leszczyński[¶‖], Krzysztof M. Ocetkiewicz[†], Jerzy Proficz[†], Paweł Czarnul[§], Henryk Krawczyk[†§], Adam Liwo[*†]

June 14, 2022

[*]Faculty of Chemistry, University of Gdańsk, Fahrenheit Union in Universities of Gdańsk, ul. Wita Stwosza 63, 80-308 Gdańsk, Poland

[†]Centre of Informatics Tri-city Academic Supercomputer and Network (CI TASK), Gdańsk University of Technology, Fahrenheit Union of Universities in Gdańsk, ul. G. Narutowicza 11/12, 80-233 Gdańsk, Poland

[‡]Department of Chemistry, University of Lleida, Av. Alcade Rovira Roure, 191, 25198 Lleida, Spain

[§]Faculty of Electronics, Telecommunication and Informatics, Gdańsk University of Technology, Fahrenheit Union of Universities in Gdańsk, ul. G. Narutowicza 11/12, 80-233 Gdańsk, Poland

[¶]Faculty of Mathematics, Physics and Informatics, University of Gdańsk, Fahrenheit Union of Universities in Gdańsk, Wita Stwosza 57, 80-308 Gdańsk, Poland

[‖]Deceased, September 30, 2021

MOST WIEDZY Downloaded from mostwiedzy.pl

Table S1: Non-setup times and their components (in seconds) for 10,000-step single-trajectory canonical MD runs with the optimized NEWCT-9P-5D(o) variant of UNRES. The 25 Å cutoff on long-range interactions has been applied.

$m * nt^a$	non-setup^b [min,max]		ene prep/sum^c [min,max]		lists^d [min,max]		SC-SC^e [min,max]		SC-p^f [min,max]		p-p+corr^g [min,max]		local^h [min,max]		grad/lagr^i [min,max]	
5OMT (109 residues)																
1*1	12.9	[12.9, 13.0]	0.190	[0.187,0.182]	2.41	[2.41, 2.42]	3.85	[3.83, 3.88]	1.78	[1.78, 1.78]	3.48	[3.45, 3.49]	0.784	[0.790,0.782]	0.133	[0.130,0.129]
1*2	8.67	[8.48, 8.76]	0.275	[0.252,0.280]	1.97	[1.95, 1.99]	2.02	[2.01, 2.02]	0.941	[0.937,0.944]	2.03	[1.99, 2.05]	0.715	[0.674,0.737]	0.255	[0.236,0.264]
2*1	8.36	[8.35, 8.37]	0.289	[0.301,0.277]	2.18	[2.18, 2.19]	1.95	[1.94, 1.97]	0.926	[0.924,0.927]	1.89	[1.89, 1.90]	0.540	[0.540,0.531]	0.267	[0.264,0.268]
1*4	6.76	[6.71, 6.82]	0.300	[0.300,0.298]	1.88	[1.86, 1.89]	1.14	[1.14, 1.15]	0.542	[0.539,0.546]	1.43	[1.42, 1.45]	0.643	[0.642,0.653]	0.326	[0.324,0.328]
2*24	6.21	[6.13, 6.30]	0.538	[0.554,0.541]	2.18	[2.16, 2.19]	0.398	[0.391,0.412]	0.213	[0.210,0.218]	0.918	[0.907,0.924]	0.567	[0.549,0.593]	0.640	[0.639,0.628]
1*6	6.08	[5.96, 6.14]	0.338	[0.325,0.341]	1.75	[1.74, 1.76]	0.878	[0.871,0.885]	0.414	[0.410,0.419]	1.22	[1.20, 1.23]	0.584	[0.562,0.594]	0.320	[0.303,0.331]
4*1	6.01	[6.00, 6.01]	0.379	[0.379,0.381]	2.05	[2.05, 2.05]	0.989	[0.989,0.990]	0.491	[0.490,0.493]	1.11	[1.10, 1.11]	0.404	[0.403,0.402]	0.282	[0.281,0.281]
1*24	5.92	[5.82, 6.02]	0.390	[0.421,0.370]	2.07	[2.06, 2.10]	0.517	[0.511,0.521]	0.255	[0.244,0.261]	0.977	[0.922,1.007]	0.600	[0.585,0.609]	0.360	[0.365,0.360]
1*12	5.40	[5.26, 5.48]	0.342	[0.345,0.348]	1.70	[1.68, 1.72]	0.594	[0.583,0.600]	0.290	[0.285,0.294]	0.932	[0.906,0.958]	0.573	[0.547,0.576]	0.342	[0.330,0.345]
6*1	5.26	[5.26, 5.27]	0.416	[0.416,0.414]	2.03	[2.03, 2.03]	0.671	[0.670,0.674]	0.337	[0.336,0.338]	0.845	[0.844,0.845]	0.354	[0.355,0.354]	0.304	[0.302,0.305]
12*1	4.46	[4.45, 4.46]	0.305	[0.314,0.297]	1.98	[1.98, 1.99]	0.409	[0.400,0.428]	0.199	[0.199,0.200]	0.620	[0.613,0.626]	0.308	[0.306,0.306]	0.324	[0.329,0.316]
24*1	4.42	[4.38, 4.48]	0.338	[0.330,0.353]	2.15	[2.14, 2.19]	0.261	[0.257,0.264]	0.136	[0.134,0.138]	0.518	[0.512,0.525]	0.311	[0.317,0.312]	0.376	[0.377,0.370]
2*12	4.36	[3.85, 5.36]	0.336	[0.274,0.460]	1.66	[1.59, 1.81]	0.313	[0.277,0.382]	0.167	[0.149,0.201]	0.570	[0.505,0.700]	0.362	[0.282,0.521]	0.428	[0.378,0.524]
4*12	3.93	[3.92, 3.95]	0.381	[0.385,0.377]	1.63	[1.62, 1.63]	0.200	[0.197,0.203]	0.114	[0.114,0.114]	0.441	[0.439,0.443]	0.268	[0.266,0.271]	0.490	[0.490,0.496]
48*1	4.66	[4.66, 4.66]	0.539	[0.539,0.539]	2.20	[2.20, 2.20]	0.182	[0.182,0.182]	0.100	[0.100,0.100]	0.460	[0.460,0.460]	0.301	[0.301,0.301]	0.539	[0.539,0.539]
5HKQ (263 residues)																
1*1	45.0	[45.0, 45.0]	0.385	[0.392,0.380]	5.73	[5.73, 5.74]	15.8	[15.8, 15.8]	7.40	[7.39, 7.40]	13.0	[13.0, 13.0]	1.78	[1.82, 1.74]	0.334	[0.332,0.334]
2*1	26.0	[26.0, 26.1]	0.989	[0.977,0.992]	4.32	[4.32, 4.32]	7.83	[7.82, 7.83]	3.77	[3.77, 3.77]	6.82	[6.81, 6.83]	1.15	[1.16, 1.15]	0.550	[0.547,0.549]
1*2	25.3	[25.2, 25.3]	0.375	[0.394,0.362]	3.88	[3.87, 3.89]	8.10	[8.09, 8.11]	3.78	[3.78, 3.78]	6.82	[6.79, 6.86]	1.26	[1.27, 1.25]	0.403	[0.402,0.402]
4*1	16.0	[16.0, 16.0]	0.799	[0.799,0.798]	3.45	[3.45, 3.45]	3.97	[3.96, 3.97]	1.90	[1.90, 1.90]	3.83	[3.82, 3.85]	0.842	[0.847,0.836]	0.593	[0.595,0.594]
1*4	15.8	[15.6, 16.0]	0.341	[0.316,0.349]	2.85	[2.82, 2.87]	4.39	[4.38, 4.40]	1.98	[1.97, 1.99]	4.24	[4.22, 4.26]	0.960	[0.881,0.973]	0.453	[0.423,0.469]
6*1	12.7	[12.7, 12.8]	0.600	[0.615,0.580]	3.21	[3.20, 3.21]	2.77	[2.76, 2.77]	1.34	[1.33, 1.34]	2.88	[2.87, 2.89]	0.738	[0.737,0.739]	0.630	[0.630,0.630]
1*6	12.7	[12.6, 12.8]	0.395	[0.435,0.378]	2.44	[2.43, 2.46]	3.05	[2.99, 3.09]	1.40	[1.39, 1.40]	3.29	[3.17, 3.38]	0.920	[0.947,0.929]	0.477	[0.480,0.491]
1*12	9.49	[9.41, 9.55]	0.395	[0.399,0.397]	2.21	[2.20, 2.22]	1.67	[1.65, 1.68]	0.877	[0.870,0.886]	2.31	[2.29, 2.32]	0.788	[0.776,0.806]	0.486	[0.474,0.491]
12*1	9.44	[9.44, 9.44]	0.564	[0.575,0.547]	2.90	[2.89, 2.90]	1.47	[1.47, 1.48]	0.744	[0.744,0.745]	1.85	[1.85, 1.86]	0.637	[0.636,0.638]	0.675	[0.678,0.669]
2*24	8.50	[8.41, 8.62]	0.631	[0.693,0.601]	2.65	[2.63, 2.66]	0.752	[0.736,0.766]	0.540	[0.516,0.554]	1.48	[1.43, 1.53]	0.674	[0.657,0.709]	0.885	[0.887,0.879]
1*24	8.49	[8.28, 8.62]	0.405	[0.404,0.391]	2.59	[2.58, 2.60]	1.09	[1.07, 1.11]	0.672	[0.638,0.702]	1.72	[1.67, 1.77]	0.691	[0.652,0.710]	0.490	[0.473,0.497]
24*1	8.39	[8.38, 8.40]	0.612	[0.615,0.612]	3.10	[3.09, 3.10]	0.816	[0.813,0.819]	0.454	[0.451,0.458]	1.36	[1.35, 1.36]	0.648	[0.650,0.646]	0.794	[0.793,0.790]
2*12	6.12	[6.12, 6.13]	0.374	[0.385,0.358]	1.99	[1.99, 1.99]	0.850	[0.845,0.857]	0.435	[0.434,0.436]	1.05	[1.05, 1.05]	0.364	[0.360,0.368]	0.548	[0.547,0.549]
4*12	5.73	[5.72, 5.75]	0.444	[0.438,0.452]	2.04	[2.04, 2.04]	0.519	[0.516,0.521]	0.290	[0.289,0.291]	0.806	[0.805,0.808]	0.331	[0.332,0.333]	0.757	[0.762,0.756]
48*1	8.24	[8.24, 8.24]	0.810	[0.810,0.810]	3.20	[3.20, 3.20]	0.484	[0.484,0.484]	0.303	[0.303,0.303]	1.11	[1.11, 1.11]	0.624	[0.624,0.624]	1.07	[1.07, 1.07]
5SY1 (2 chains) (729 residues)																
1*1	151	[151, 151]	0.974	[0.968,0.965]	20.4	[20.4, 20.5]	53.6	[53.5, 54.0]	25.8	[25.8, 25.9]	42.8	[42.7, 42.8]	4.88	[4.90, 4.83]	1.02	[1.02, 1.03]
2*1	84.0	[83.9, 84.1]	1.15	[1.10, 1.26]	14.3	[14.3, 14.3]	26.9	[26.9, 26.9]	13.1	[13.1, 13.1]	22.5	[22.5, 22.6]	3.12	[3.16, 3.06]	1.44	[1.44, 1.43]
1*2	79.3	[79.3, 79.4]	0.719	[0.722,0.716]	11.9	[11.9, 11.9]	27.0	[26.9, 27.0]	13.2	[13.1, 13.2]	21.5	[21.4, 21.5]	2.80	[2.90, 2.70]	1.16	[1.16, 1.16]
4*1	49.0	[49.0, 49.1]	1.07	[1.09, 1.00]	9.83	[9.81, 9.86]	13.6	[13.6, 13.6]	6.69	[6.68, 6.71]	12.4	[12.4, 12.5]	2.26	[2.26, 2.25]	1.69	[1.70, 1.69]
1*4	44.5	[44.3, 44.6]	0.524	[0.513,0.512]	7.03	[7.03, 7.04]	13.8	[13.8, 13.9]	6.74	[6.73, 6.75]	12.5	[12.4, 12.6]	1.73	[1.73, 1.68]	1.09	[1.08, 1.10]
6*1	37.2	[37.2, 37.3]	1.06	[1.07, 1.07]	8.04	[8.03, 8.05]	9.24	[9.23, 9.25]	4.57	[4.57, 4.58]	9.04	[9.02, 9.06]	1.97	[1.97, 1.95]	1.84	[1.84, 1.85]
1*6	32.8	[32.5, 32.9]	0.526	[0.549,0.513]	5.37	[5.36, 5.38]	9.46	[9.41, 9.52]	4.60	[4.59, 4.60]	9.20	[8.94, 9.35]	1.44	[1.46, 1.38]	1.11	[1.09, 1.12]
12*1	25.9	[25.9, 25.9]	1.27	[1.27, 1.25]	6.63	[6.62, 6.63]	4.76	[4.76, 4.76]	2.44	[2.43, 2.44]	5.61	[5.58, 5.64]	1.73	[1.73, 1.73]	2.00	[2.01, 1.99]
24*1	22.6	[22.5, 22.6]	1.80	[1.80, 1.79]	6.47	[6.45, 6.47]	2.57	[2.57, 2.58]	1.44	[1.42, 1.45]	4.07	[4.05, 4.09]	2.13	[2.16, 2.11]	2.56	[2.56, 2.57]
1*12	20.7	[20.5, 20.9]	0.497	[0.479,0.495]	3.87	[3.85, 3.88]	4.97	[4.95, 5.00]	2.52	[2.52, 2.53]	5.46	[5.37, 5.52]	1.19	[1.16, 1.21]	1.12	[1.09, 1.14]
1*24	16.0	[15.7, 16.3]	0.517	[0.526,0.545]	3.63	[3.60, 3.66]	2.81	[2.79, 2.84]	1.65	[1.60, 1.71]	4.00	[3.92, 4.09]	1.02	[1.00, 1.06]	1.15	[1.10, 1.20]
2*24	14.5	[14.2, 14.7]	0.903	[0.866,0.938]	3.67	[3.61, 3.76]	1.66	[1.65, 1.66]	1.10	[1.08, 1.12]	3.09	[3.07, 3.12]	0.845	[0.843,0.842]	1.90	[1.79, 1.98]
2*12	12.8	[12.8, 12.9]	0.454	[0.437,0.465]	3.21	[3.21, 3.22]	2.53	[2.52, 2.54]	1.30	[1.30, 1.30]	2.67	[2.67, 2.67]	0.558	[0.568,0.552]	1.28	[1.28, 1.28]
4*12	10.6	[10.6, 10.8]	0.593	[0.563,0.642]	2.96	[2.95, 2.99]	1.40	[1.40, 1.40]	0.748	[0.746,0.749]	1.84	[1.83, 1.84]	0.469	[0.470,0.471]	1.73	[1.71, 1.74]
48*1	20.5	[20.5, 20.5]	2.11	[2.11, 2.11]	6.26	[6.26, 6.26]	1.43	[1.43, 1.43]	0.915	[0.915,0.915]	3.20	[3.20, 3.20]	2.06	[2.06, 2.06]	2.91	[2.91, 2.91]

MOST WIEDZY Downloaded from mostwiedzy.pl

5SY1 (1458 residues)

$m*nt$[a]	non-setup[b]	[min,max]	ene prep/sum[c]	[min,max]	lists[d]	[min,max]	SC-SC[e]	[min,max]	SC-p[f]	[min,max]	p-p+corr[g]	[min,max]	local[h]	[min,max]	grad/lagr[i]	[min,max]
1*1	329	[329, 329]	1.76	[1.78, 1.73]	51.0	[51.0, 51.0]	114	[114, 115]	55.7	[55.7, 55.8]	91.6	[91.3, 91.8]	9.69	[9.89, 9.48]	2.26	[2.25, 2.26]
2*1	180	[180, 181]	2.44	[2.13, 2.76]	33.0	[33.0, 33.0]	57.2	[57.1, 57.2]	28.0	[28.0, 28.1]	47.8	[47.8, 47.9]	6.21	[6.19, 6.19]	2.96	[2.96, 2.95]
1*2	171	[171, 172]	1.28	[1.28, 1.25]	28.1	[28.1, 28.1]	58.3	[58.1, 58.7]	28.4	[28.4, 28.4]	46.0	[45.8, 46.4]	5.20	[5.22, 4.65]	1.75	[1.75, 1.74]
4*1	105	[105, 106]	2.70	[2.63, 2.78]	22.3	[22.3, 22.3]	29.1	[29.1, 29.1]	14.5	[14.5, 14.5]	26.2	[26.1, 26.2]	4.45	[4.48, 4.40]	3.50	[3.49, 3.49]
1*4	92.0	[91.7, 92.3]	0.768	[0.807,0.578]	15.5	[15.4, 15.6]	29.4	[29.3, 29.4]	14.3	[14.3, 14.4]	26.1	[25.9, 26.6]	2.83	[3.03, 2.59]	1.46	[1.44, 1.49]
6*1	79.9	[79.8, 80.0]	2.43	[2.40, 2.52]	18.2	[18.2, 18.2]	19.9	[19.9, 19.9]	9.85	[9.85, 9.85]	19.0	[19.0, 19.1]	3.88	[3.88, 3.87]	3.80	[3.81, 3.80]
1*6	65.6	[65.6, 65.6]	0.712	[0.736,0.698]	11.2	[11.2, 11.2]	19.8	[19.8, 19.9]	9.82	[9.80, 9.85]	18.8	[18.8, 18.9]	2.19	[2.22, 2.12]	1.44	[1.47, 1.43]
12*1	56.0	[56.0, 56.1]	2.74	[2.77, 2.72]	14.1	[14.1, 14.1]	10.3	[10.3, 10.4]	5.31	[5.30, 5.33]	12.0	[12.0, 12.0]	4.04	[4.06, 4.05]	4.56	[4.55, 4.58]
24*1	50.7	[50.3, 51.3]	4.17	[4.03, 4.32]	12.2	[12.1, 12.3]	5.74	[5.71, 5.78]	3.15	[3.14, 3.16]	9.07	[9.03, 9.14]	6.18	[6.17, 6.21]	6.98	[6.86, 7.11]
1*12	39.7	[39.4, 40.0]	0.608	[0.604,0.555]	7.29	[7.27, 7.32]	10.6	[10.5, 10.8]	5.25	[5.24, 5.25]	11.3	[11.2, 11.4]	1.63	[1.60, 1.63]	1.45	[1.43, 1.47]
1*24	26.9	[26.7, 27.1]	0.644	[0.626,0.656]	5.77	[5.76, 5.79]	5.68	[5.65, 5.72]	3.07	[3.04, 3.09]	7.09	[6.98, 7.22]	1.35	[1.42, 1.32]	1.57	[1.53, 1.59]
2*12	23.7	[23.7, 23.7]	0.500	[0.497,0.489]	5.54	[5.53, 5.55]	5.37	[5.37, 5.38]	2.74	[2.74, 2.75]	5.49	[5.48, 5.49]	0.890	[0.900,0.883]	1.85	[1.85, 1.85]
2*24	21.9	[21.6, 22.3]	1.01	[1.08, 1.07]	5.32	[5.21, 5.42]	3.16	[3.13, 3.19]	1.91	[1.89, 1.93]	4.92	[4.81, 5.00]	1.14	[1.12, 1.14]	2.57	[2.50, 2.66]
4*12	18.8	[18.4, 19.7]	1.22	[0.92, 1.82]	4.99	[4.89, 5.19]	2.82	[2.82, 2.82]	1.51	[1.51, 1.51]	3.54	[3.54, 3.54]	0.720	[0.716,0.723]	2.53	[2.49, 2.61]
48*1	45.8	[45.8, 45.8]	4.43	[4.43, 4.43]	11.2	[11.2, 11.2]	3.57	[3.57, 3.57]	2.08	[2.08, 2.08]	7.24	[7.24, 7.24]	6.22	[6.22, 6.22]	7.59	[7.59, 7.59]

4R3O (half) (3,143 residues)

$m*nt$[a]	non-setup[b]	[min,max]	ene prep/sum[c]	[min,max]	lists[d]	[min,max]	SC-SC[e]	[min,max]	SC-p[f]	[min,max]	p-p+corr[g]	[min,max]	local[h]	[min,max]	grad/lagr[i]	[min,max]
1*1	708	[707, 708]	3.68	[3.74, 3.62]	107	[107, 107]	249	[249, 249]	118	[118, 118]	199	[199, 199]	20.6	[20.6, 20.2]	4.82	[4.82, 4.82]
2*1	394	[394, 394]	3.77	[3.85, 3.76]	70.8	[70.7, 70.8]	127	[127, 127]	60.1	[60.1, 60.3]	106	[106, 106]	13.2	[13.3, 12.9]	6.25	[6.25, 6.26]
1*2	364	[363, 367]	2.20	[2.27, 2.14]	57.7	[57.6, 57.7]	126	[126, 127]	59.9	[59.8, 59.9]	99.9	[98.8,101.8]	10.7	[10.7, 10.6]	3.49	[3.49, 3.50]
4*1	225	[225, 226]	4.65	[4.66, 4.31]	47.3	[47.3, 47.4]	63.2	[62.9, 63.8]	30.3	[30.3, 30.3]	56.8	[56.7, 56.8]	9.71	[9.80, 9.57]	7.25	[7.24, 7.28]
1*4	194	[191, 196]	1.41	[1.37, 1.43]	31.0	[30.9, 31.0]	64.1	[64.0, 64.1]	30.3	[30.3, 30.3]	56.3	[53.6, 57.8]	5.71	[5.73, 5.71]	2.45	[2.38, 2.46]
6*1	175	[175, 176]	5.28	[5.20, 5.48]	39.6	[39.5, 39.7]	43.0	[42.8, 43.2]	20.9	[20.9, 20.9]	41.8	[41.8, 41.9]	9.43	[9.37, 9.37]	8.71	[8.68, 8.72]
1*6	136	[135, 136]	1.16	[1.15, 1.18]	22.2	[22.2, 22.2]	43.2	[42.8, 43.6]	20.5	[20.4, 20.5]	40.0	[39.7, 40.3]	4.06	[4.06, 3.57]	2.10	[2.09, 2.11]
12*1	125	[125, 125]	6.25	[6.35, 6.20]	30.2	[30.1, 30.2]	22.3	[22.3, 22.4]	11.3	[11.3, 11.4]	26.7	[26.7, 26.8]	10.2	[10.2, 10.1]	11.0	[11.0, 11.0]
24*1	113	[112, 114]	11.8	[12.4, 11.7]	26.4	[26.4, 26.5]	12.7	[12.7, 12.8]	6.55	[6.50, 6.62]	20.3	[19.8, 20.6]	13.8	[13.3, 14.0]	14.4	[14.3, 14.5]
1*12	76.9	[76.5, 77.2]	0.860	[0.899,0.788]	13.3	[13.2, 13.4]	22.1	[22.0, 22.3]	10.7	[10.7, 10.7]	22.9	[22.7, 23.2]	2.57	[2.52, 2.25]	1.94	[1.92, 1.94]
1*24	50.7	[50.2, 51.6]	0.813	[0.814,0.912]	9.96	[9.92, 9.99]	12.0	[12.0, 12.0]	5.97	[5.93, 6.04]	15.4	[15.1, 16.1]	1.92	[1.92, 1.93]	1.91	[1.87, 1.98]
2*12	46.9	[46.6, 47.2]	0.760	[0.730,0.778]	10.3	[10.2, 10.3]	11.4	[11.3, 11.4]	5.61	[5.59, 5.63]	11.7	[11.6, 11.8]	1.67	[1.67, 1.65]	2.99	[2.96, 3.01]
2*24	39.7	[38.9, 40.5]	1.46	[1.33, 1.64]	9.69	[9.59, 9.79]	6.54	[6.46, 6.65]	3.40	[3.36, 3.44]	10.1	[9.9, 10.3]	1.76	[1.74, 1.73]	3.67	[3.55, 3.88]
4*12	37.2	[37.1, 37.2]	0.933	[0.968,0.913]	10.3	[10.3, 10.3]	5.99	[5.98, 6.00]	3.16	[3.15, 3.17]	7.76	[7.75, 7.78]	1.35	[1.35, 1.35]	4.80	[4.79, 4.80]
48*1	101	[101, 101]	11.4	[11.4, 11.4]	24.1	[24.1, 24.1]	7.96	[7.96, 7.96]	4.14	[4.14, 4.14]	17.4	[17.4, 17.4]	13.5	[13.5, 13.5]	15.3	[15.3, 15.3]

4R3O (6,286 residues)

$m*nt$[a]	non-setup[b]	[min,max]	ene prep/sum[c]	[min,max]	lists[d]	[min,max]	SC-SC[e]	[min,max]	SC-p[f]	[min,max]	p-p+corr[g]	[min,max]	local[h]	[min,max]	grad/lagr[i]	[min,max]
1*1	1668	[1667, 1670]	7.89	[7.84, 7.77]	281	[281, 281]	580	[580, 581]	278	[277, 278]	457	[457, 458]	41.4	[41.9, 41.3]	11.0	[10.9, 10.9]
2*1	915	[914, 918]	7.87	[7.83, 7.90]	183	[183, 183]	293	[292, 295]	140	[140, 140]	239	[239, 239]	26.6	[27.1, 26.6]	13.6	[13.5, 13.6]
1*2	846	[845, 847]	4.23	[4.68, 3.77]	148	[148, 148]	294	[292, 296]	140	[140, 140]	224	[224, 225]	21.0	[21.6, 19.8]	6.61	[6.61, 6.60]
4*1	536	[536, 537]	14.4	[15.2, 13.3]	121	[121, 121]	146	[146, 147]	70.9	[70.7, 71.1]	130	[130, 131]	22.3	[22.2, 22.3]	18.0	[17.9, 18.1]
1*4	453	[452, 456]	2.39	[2.39, 2.44]	78.6	[78.3, 78.9]	149	[148, 149]	71.2	[71.0, 71.3]	132	[130, 133]	10.9	[11.3, 10.7]	4.53	[4.54, 4.56]
6*1	401	[400, 402]	14.0	[14.0, 13.9]	94.7	[94.6, 94.7]	99.0	[98.5, 99.9]	48.1	[47.9, 48.3]	94.0	[93.9, 94.0]	20.5	[20.7, 20.3]	17.8	[17.8, 17.8]
1*6	307	[303, 313]	1.94	[1.95, 1.86]	55.0	[54.9, 55.0]	98.9	[98.4, 99.7]	47.9	[47.9, 48.0]	86.8	[83.3, 91.9]	7.58	[7.58, 7.62]	3.87	[3.87, 3.90]
4*12	94.1	[92.9, 98.6]	2.93	[2.40, 6.00]	22.0	[21.9, 22.5]	14.2	[14.1, 14.3]	7.87	[7.83, 7.93]	19.6	[19.4, 19.7]	3.84	[3.83, 3.86]	15.2	[15.0, 15.4]
12*1	275	[274, 275]	16.1	[16.3, 15.6]	69.7	[69.6, 69.7]	51.4	[51.3, 51.4]	25.5	[25.4, 25.6]	59.1	[58.8, 59.4]	19.8	[19.8, 20.0]	20.1	[20.1, 20.1]
24*1	255	[254, 256]	24.0	[23.6, 23.9]	59.2	[59.1, 59.2]	28.9	[28.9, 28.9]	14.5	[14.4, 14.5]	50.3	[50.2, 50.5]	34.7	[34.6, 35.1]	29.6	[29.5, 29.7]
2*24	83.2	[81.7, 88.4]	2.50	[3.63, 5.30]	19.4	[19.1, 20.0]	14.8	[14.5, 15.6]	8.34	[8.15, 8.51]	21.6	[20.1, 22.5]	3.00	[2.89, 2.54]	6.44	[6.35, 6.68]
1*12	172	[171, 173]	1.35	[1.37, 1.33]	31.6	[31.6, 31.6]	50.6	[50.3, 50.9]	24.8	[24.8, 24.8]	51.3	[50.4, 52.1]	4.46	[4.41, 4.10]	3.31	[3.28, 3.34]
2*12	110	[110, 110]	1.82	[1.71, 2.02]	23.5	[23.5, 23.6]	26.5	[26.5, 26.6]	13.9	[13.9, 14.0]	28.6	[28.4, 28.6]	4.36	[4.45, 4.41]	6.13	[6.11, 6.15]
1*24	108	[106, 109]	1.27	[1.27, 1.28]	21.6	[21.6, 21.7]	27.2	[27.0, 27.4]	14.2	[14.0, 14.4]	32.1	[31.4, 32.8]	3.11	[3.03, 3.16]	3.44	[3.40, 3.51]
48*1	226	[226, 226]	24.6	[24.6, 24.6]	52.6	[52.6, 52.6]	16.8	[16.8, 16.8]	8.91	[8.91, 8.91]	42.1	[42.1, 42.1]	33.4	[33.4, 33.4]	30.7	[30.7, 30.7]

MOST WIEDZY Downloaded from mostwiedzy.pl

H1081 (15,200 residues)

$m*nt^a$	non-setup^b [min,max]		ene prep/sum^c [min,max]		lists^d [min,max]		SC-SC^e [min,max]		SC-p^f [min,max]		p-p+corr^g [min,max]		local^h [min,max]		grad/lagr^i [min,max]	
1*1	4571	[4529, 4645]	25.3	[25.4, 25.3]	850	[849, 850]	1568	[1531, 1640]	745	[745, 746]	1209	[1207, 1211]	112	[111, 110]	33.3	[33.1, 33.6]
2*1	2470	[2468, 2471]	24.5	[25.3, 24.2]	529	[529, 529]	769	[769, 770]	376	[376, 378]	629	[628, 630]	72.2	[72.3, 70.0]	40.0	[40.1, 40.0]
1*2	2277	[2275, 2281]	11.6	[12.4, 11.7]	446	[445, 446]	768	[768, 769]	374	[374, 374]	587	[586, 589]	53.0	[52.8, 53.2]	18.4	[18.3, 18.6]
4*1	1462	[1460, 1465]	31.4	[32.7, 30.0]	374	[374, 374]	392	[390, 395]	193	[193, 194]	345	[344, 346]	54.5	[54.4, 54.3]	42.9	[42.9, 43.0]
1*4	1201	[1193, 1212]	6.74	[7.49, 5.96]	234	[233, 234]	389	[388, 390]	190	[189, 190]	329	[322, 340]	27.9	[28.0, 27.4]	11.3	[11.3, 11.3]
6*1	1094	[1094, 1095]	38.8	[39.1, 38.5]	295	[295, 295]	261	[261, 262]	130	[130, 130]	247	[246, 247]	48.6	[48.7, 48.4]	44.6	[44.6, 44.6]
1*6	830	[822, 844]	4.81	[4.97, 4.85]	162	[162, 162]	262	[261, 263]	128	[127, 128]	233	[226, 244]	19.9	[20.5, 20.0]	9.85	[9.68,10.00]
12*1	741	[739, 742]	42.3	[42.1, 42.1]	210	[209, 210]	134	[134, 134]	68.0	[67.9, 68.0]	153	[152, 153]	55.2	[55.2, 55.2]	49.0	[48.8, 49.1]
24*1	668	[661, 672]	60.7	[60.0, 60.0]	172	[171, 173]	72.3	[72.0, 72.6]	38.6	[38.4, 38.7]	130	[128, 132]	94.3	[92.7, 95.6]	66.4	[65.2, 67.0]
1*12	459	[457, 460]	3.32	[3.41, 3.37]	91.2	[91.1, 91.2]	133	[133, 134]	66.7	[66.6, 66.7]	134	[133, 135]	11.6	[12.7, 11.1]	8.45	[8.39, 8.54]
48*1	601	[599, 602]	60.7	[62.7, 58.7]	153	[152, 155]	40.9	[40.8, 41.1]	23.7	[23.4, 24.0]	107	[106, 107]	96.2	[96.1, 96.3]	81.7	[81.2, 82.3]
2*12	287	[286, 288]	4.50	[4.37, 4.61]	63.9	[63.9, 64.0]	69.3	[69.2, 69.5]	36.9	[36.8, 37.0]	76.5	[76.0, 77.1]	10.3	[10.3, 10.5]	15.1	[15.1, 15.1]
1*24	279	[278, 282]	2.58	[2.64, 2.72]	57.3	[57.2, 57.5]	71.2	[70.6, 71.7]	38.2	[38.0, 38.4]	84.0	[82.8, 85.3]	8.43	[8.90, 8.50]	8.17	[8.12, 8.30]
4*12	228	[227, 231]	5.72	[5.86, 5.36]	58.0	[57.4, 59.1]	37.0	[37.0, 37.1]	21.2	[21.2, 21.3]	52.6	[52.3, 52.9]	8.92	[8.80, 9.05]	29.8	[29.5, 30.8]
2*24	215	[212, 218]	3.85	[3.55, 3.91]	47.6	[47.2, 48.3]	38.5	[38.4, 38.7]	22.2	[22.1, 22.4]	57.2	[56.4, 57.9]	7.25	[7.15, 6.79]	25.0	[24.6, 25.9]

6YGH (62,880 residues)

$m*nt^a$	non-setup^b [min,max]		ene prep/sum^c [min,max]		lists^d [min,max]		SC-SC^e [min,max]		SC-p^f [min,max]		p-p+corr^g [min,max]		local^h [min,max]		grad/lagr^i [min,max]	
1*1	18042	[18032,18060]	163	[163, 161]	3538	[3537, 3538]	5979	[5976, 5981]	2890	[2886, 2899]	4730	[4720, 4741]	476	[483, 474]	149	[149, 148]
2*1	9893	[9887, 9901]	144	[145, 144]	2211	[2209, 2215]	3005	[3004, 3005]	1458	[1457, 1458]	2472	[2466, 2476]	306	[308, 304]	177	[176, 177]
1*2	9046	[9041, 9053]	67.8	[67.9, 67.7]	1842	[1841, 1842]	3003	[3001, 3005]	1450	[1450, 1450]	2293	[2289, 2298]	232	[234, 230]	84.7	[84.6, 84.9]
4*1	5781	[5779, 5785]	150	[149, 153]	1479	[1478, 1479]	1516	[1514, 1520]	742	[742, 742]	1350	[1349, 1353]	235	[238, 231]	187	[187, 187]
1*4	4802	[4793, 4815]	35.0	[35.2, 33.7]	960	[959, 961]	1527	[1521, 1539]	735	[734, 737]	1323	[1317, 1328]	120	[124, 115]	52.2	[52.5, 52.0]
6*1	4369	[4356, 4376]	155	[150, 156]	1191	[1190, 1191]	1015	[1014, 1017]	504	[503, 505]	977	[975, 978]	214	[211, 216]	192	[191, 192]
1*6	3292	[3270, 3303]	24.5	[25.0, 23.4]	667	[666, 667]	1020	[1017, 1024]	493	[492, 494]	923	[904, 936]	82.0	[82.3, 74.9]	41.1	[41.2, 41.2]
12*1	3032	[3025, 3047]	159	[159, 160]	886	[885, 887]	524	[523, 524]	269	[269, 269]	619	[617, 621]	235	[233, 242]	208	[207, 211]
24*1	2923	[2891, 2942]	242	[233, 247]	740	[738, 741]	291	[290, 292]	169	[166, 170]	525	[520, 528]	406	[399, 410]	385	[380, 388]
1*12	1805	[1803, 1808]	14.8	[14.9, 14.2]	371	[371, 372]	520	[516, 521]	257	[256, 257]	528	[524, 530]	45.6	[49.7, 44.9]	32.7	[33.0, 32.3]
48*1	2594	[2567, 2621]	372	[379, 364]	633	[626, 639]	157	[155, 159]	95.8	[95.4, 96.2]	381	[373, 390]	387	[380, 394]	418	[406, 430]
2*12	1150	[1149, 1151]	21.0	[21.3, 20.9]	259	[259, 259]	271	[270, 271]	159	[159, 159]	300	[299, 300]	45.0	[46.1, 44.2]	54.8	[54.6, 55.1]
1*24	1128	[1124, 1131]	12.0	[12.3, 12.1]	226	[226, 226]	280	[278, 281]	164	[164, 165]	346	[345, 348]	32.3	[32.3, 32.9]	31.1	[31.2, 31.4]
4*12	895	[890, 902]	22.6	[23.8, 21.1]	220	[218, 222]	146	[146, 147]	101	[101, 101]	210	[210, 211]	38.1	[38.6, 37.6]	103	[100, 106]
2*24	872	[866, 877]	17.4	[17.7, 14.2]	184	[183, 184]	156	[152, 161]	106	[105, 108]	246	[245, 247]	26.4	[28.6, 26.6]	89.5	[89.0, 89.5]

5Y6P (153,243 residues)

$m*nt^a$	non-setup^b [min,max]		ene prep/sum^c [min,max]		lists^d [min,max]		SC-SC^e [min,max]		SC-p^f [min,max]		p-p+corr^g [min,max]		local^h [min,max]		grad/lagr^i [min,max]	
1*1	53979	[53843,54091]	398	[396, 381]	10922	[10914,10934]	18019	[17994,18031]	8786	[8751, 8809]	14017	[13958,14100]	1166	[1161, 1159]	374	[372, 377]
2*1	29623	[29604,29657]	363	[366, 366]	7105	[7100, 7112]	9030	[9025, 9040]	4404	[4401, 4410]	7211	[7200, 7219]	752	[756, 753]	453	[453, 453]
1*2	27153	[26936,27588]	179	[184, 177]	5656	[5653, 5658]	9225	[9011, 9650]	4403	[4397, 4407]	6722	[6719, 6725]	573	[575, 574]	212	[212, 213]
4*1	17209	[17193,17232]	384	[379, 397]	4833	[4831, 4836]	4540	[4537, 4545]	2235	[2235, 2235]	3871	[3868, 3874]	571	[570, 568]	468	[467, 468]
1*4	14239	[14189,14271]	92.0	[96.8, 87.2]	2950	[2947, 2952]	4601	[4586, 4623]	2217	[2214, 2219]	3837	[3799, 3858]	291	[296, 282]	129	[129, 128]
6*1	12974	[12901,13111]	457	[388, 590]	3880	[3878, 3885]	3049	[3047, 3051]	1514	[1513, 1516]	2768	[2767, 2768]	518	[519, 515]	475	[476, 475]
1*6	9624	[9531, 9726]	63.9	[65.7, 56.8]	2040	[2037, 2045]	3048	[3030, 3063]	1482	[1480, 1484]	2587	[2514, 2675]	200	[199, 199]	101	[102, 101]
12*1	8893	[8880, 8912]	393	[390, 403]	2825	[2822, 2830]	1568	[1567, 1569]	802	[801, 804]	1702	[1699, 1705]	587	[589, 584]	678	[673, 681]
24*1	8176	[8085, 8221]	570	[556, 574]	2311	[2306, 2314]	875	[872, 877]	498	[494, 503]	1383	[1367, 1394]	978	[956, 988]	1158	[1133, 1169]
1*12	5268	[5243, 5292]	39.0	[41.0, 37.0]	1136	[1135, 1137]	1560	[1556, 1566]	781	[781, 781]	1473	[1442, 1500]	113	[122, 107]	76.6	[77.1, 76.3]
2*12	3355	[3342, 3378]	51.6	[52.8, 48.6]	827	[827, 827]	822	[806, 849]	461	[459, 463]	834	[834, 835]	121	[125, 116]	138	[138, 138]
1*24	3215	[3202, 3222]	41.2	[41.3, 41.3]	688	[687, 688]	834	[824, 844]	476	[473, 478]	933	[929, 938]	86.4	[87.3, 75.9]	71.3	[71.3, 72.7]
4*12	2535	[2507, 2581]	79.8	[63.3,112.1]	701	[699, 703]	430	[429, 430]	280	[279, 280]	568	[566, 571]	106	[104, 107]	249	[246, 255]
2*24	2385	[2379, 2390]	54.9	[53.1, 53.3]	570	[570, 570]	443	[441, 446]	297	[292, 300]	627	[624, 629]	72.6	[77.4, 69.8]	213	[214, 214]
48*1	7015	[7015, 7015]	842	[842, 842]	1899	[1899, 1899]	451	[451, 451]	264	[264, 264]	987	[987, 987]	974	[974, 974]	1223	[1223, 1223]

[a] $m*nt$ in the first column stands for m FG (MPI) processes each split into nt OpenMP threads.

[b] Total non-setup time.

[c] The time spent on preparation for energy and energy-gradient calculations (which include putting the particles into the simulation box, coordinate and energy-term weight broadcast to slave processes, and zeroing the gradient components) and the reduction of energy

MOST WIEDZY Downloaded from mostwiedzy.pl

contributions from different processes.
[d]The time for the construction of the interaction lists.
[e]The time for the calculation of the $U_{SC_jSC_j}$ contributions to energy and energy gradient.
[f]The time for the calculation of the $U_{SC_ip_j}$ contributions to energy and energy gradient.
[g]The time for the calculation of the $U_{p_ip_j}$ and $U^{(3)}_{corr}$ contributions to energy and energy gradient.
[h]The time for the calculations local (U_{bond}, U_b, U_{tor} and U_{rot}) contributions to energy and energy gradient.
[i]The time for gradient-component summation and transformation and calculating accelerations from the energy gradient.

MOST WIEDZY Downloaded from mostwiedzy.pl

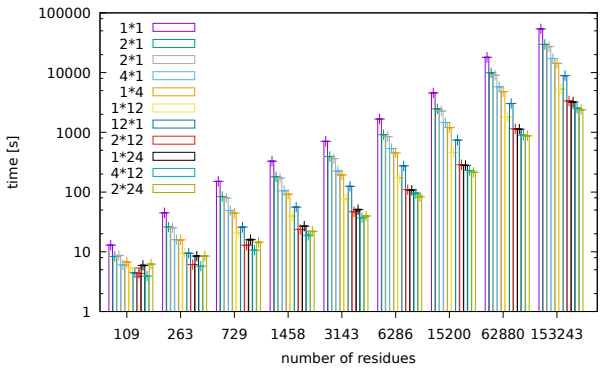


Figure S1: Dependence of the non-setup time for 10,000 steps of single-trajectory canonical MD simulations on protein size for various total number of cores and their partition between MPI processes and OpenMP threads.

MOST WIEDZY Downloaded from mostwiedzy.pl

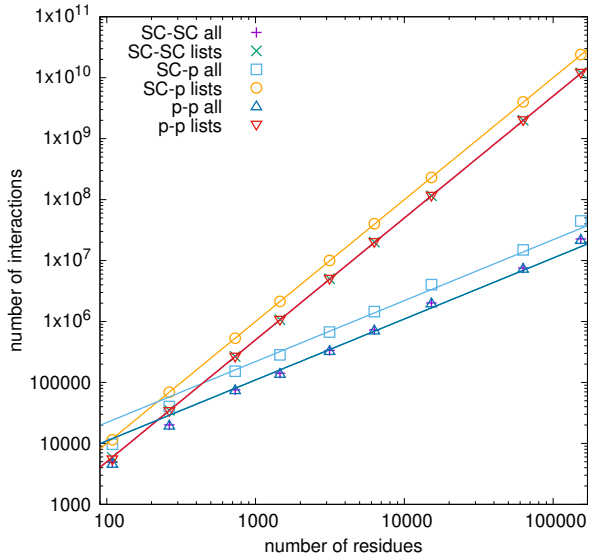


Figure S2: Dependence of the numbers of long-range SC-SC, SC-p, and p-p interactions on the number of residues without cut-off and with the 25 Å cut-off.

MOST WIEDZY Downloaded from mostwiedzy.pl

Details of the simulations and data analysis of the 2JOF miniprotein

The variant of the tryptophan cage studied by Lindorff-Larsen et al.[1] (PDB:2JOF) was simulated with the force-matched variant of the scale-consistent UNRES force field[2], which was calibrated with the regular tryptophan cage (PDB:1L2Y). First, the protocol of *ab initio* prediction of protein structures[3,4], which is based on extensive MREMD simulations, was run to determine if the force field folds this protein. A production MREMD run 12 four-plexed trajectories at 12 temperatures ranging from 260 K to 370 K (48 trajectories total) was carried out for 20,000,000 4.89 fs steps, as in Ref. 4. For this small protein, the duration of the run was sufficient to achieve convergence. The structure closest to the mean of the most populated cluster had C^{α}-RMSD = 2.73 Å from the experimental structure (Figure S3). Consequently, the force field can be considered to be good for studying the folding of 2JOF.

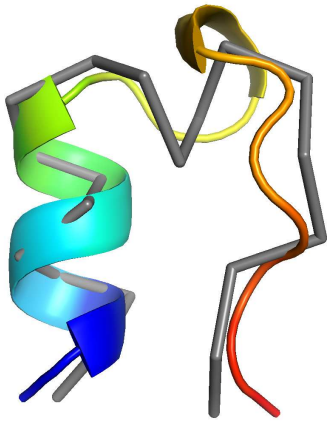


Figure S3: Superposition of the structure closest to the mean structure of the lowest-free-energy cluster of the variant of tryptophan cage obtained in MREMD simulations (gray C^{α}-trace) on the experimental 2JOF structure (ribbon colored blue to red from the N- to the C-terminus). The C^{α}-RMSD is 2.73 Å.

To compare the UNRES and all-atom folding/unfolding times, we subsequently ran 4 series of simulations, 24 canonical Langevin-dynamics trajectories each, at temperatures 268 K, 273 K, 278 K, and 288 K, respectively, to find the temperature at which the ratio of the folding to unfolding time was about the same as that in Ref. 1. Water viscosity was scaled by a factor of 0.01, as in our earlier work[5]. Each trajectory was started from a fully extended conformation. We took the same time step of 2 fs as in Ref. 1 and the total number of steps was 400,000,000 (0.8 μs), which amounts to the total simulation time of 19.2 μs for each series of simulations (each of the four temperatures). Snapshots were recorded every 1,000 MD steps (2 ps).

To find the folding and unfolding times, we carried out the same analysis as in Ref. 1 except that we used the RMSD from the experimental structure and not the fraction of the native contacts. First, the RMSD time series were smoothed by computing the moving averages with period of 100 ps (50 snapshots). Then the smoothed RMSD was analyzed starting from the first to the last point. If the initial RMSD was below the "folded" threshold,

MOST WIEDZY Downloaded from mostwiedzy.pl

Table S2: Mean folding ($\langle\tau_f\rangle$) and unfolding ($\langle\tau_u\rangle$) times and their ratios of the 2JOF miniprotein obtained in canonical simulations at four temperatures.

T[K]	$\langle\tau_f\rangle[\mu s]$	$\langle\tau_u\rangle$ $[\mu s]$	$\langle\tau_f\rangle/\langle\tau_u\rangle$
268	0.0192	0.00797	2.4
273	0.0178	0.00446	4.0
278	0.0283	0.00326	8.7
288	0.0428	0.00143	29.9

set at 2.8 Å, the point was flagged '1' and this flag continued until the RMSD increased to more than the "unfolded" threshold, set at 4.2 Å, this point being flagged '0'. The flag '0' continued until the RMSD decreased below 2.8 Å. If the first RMSD was above 2.8 Å, its flag was set at 0. Then the same analysis was carried out from the last to the first RMSD, yielding the second set of flags. At each point, the final flag was computed as an arithmetic average of the forward and the backward flag. A final flag of 0 indicated an unfolded, a flag of 1 a folded, and a flag of 1/2 a transition structure. Sample plots of moving averages of the RMSD for all trajectories at T =273 K, in which the "folded", "unfolded", and "transition" sections are colored differently are shown in Figure S4. It can be seen that the protein repeatedly folded and unfolded in all trajectories.

Subsequently, the unfolding (τ_u) and folding times (τ_f) were as the periods of residence in the folded and unfolded state, respectively. The distributions of these time for T =273 K are shown in Figure S5. It can be seen that the distributions do have dominant maxima but are not unimodal, which suggests heterogeneous folding and unfolding kinetics. The distribution of τ_f is shifted to the right with respect to that of τ_u. The distributions of the folding and unfolding times at the three remaining temperatures exhibit the same feature. The mean values of the folding and unfolding times are and their ratios at the four temperatures considered are summarized in Table S2.

It can be seen from the Table that the $\langle\tau_u\rangle$ (which is the residence time of the system in the folded state) steadily decreases with temperature, while $\langle\tau_f\rangle$ (which is the residence time of the system in the unfolded state) increases with temperature except for a small drop from 268 K to 273 K. This behavior is understandable, because the increase of temperature results in reducing the probability of the folded state. At T = 273 K, the ratio of the mean folding and unfolding times is about 4.0, which is close to the ratio of $14/3 = 4.7$ found in the all-atom simulations found in Ref. 1. Consequently, we selected the values obtained in simulations carried out at T = 273 K to compare the UNRES and all-atom folding/unfolding times.

MOST WIEDZY Downloaded from mostwiedzy.pl

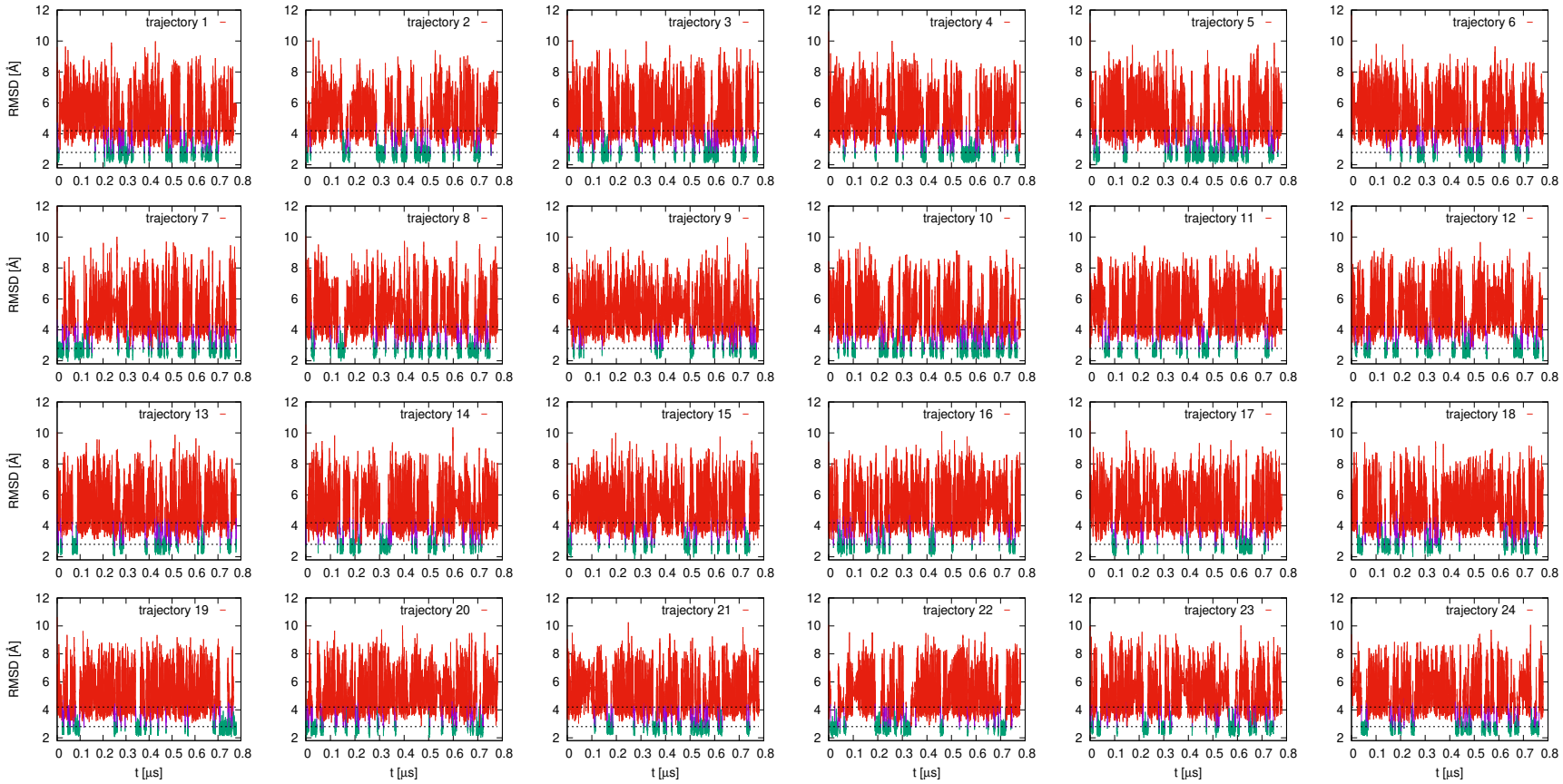


Figure S4: Plots of the moving-average-smoothed C^{α}-RMSD of the simulated structures of the variant of the 2JOF variant of the tryptophan cage from the experimental 2JOF structure in simulation time simulated at T = 273 K. The red, purple, and green sections of the plots correspond to the unfolded, transition, and folded structures of the miniprotein and the horizontal dashed lines have been drawn at the "folded" (2.8 Å) and "unfolded" (4.2 Å) RMSD threshold. The RMSD cut-off values are shown as dashed lines.

MOST WIEDZY Downloaded from mostwiedzy.pl

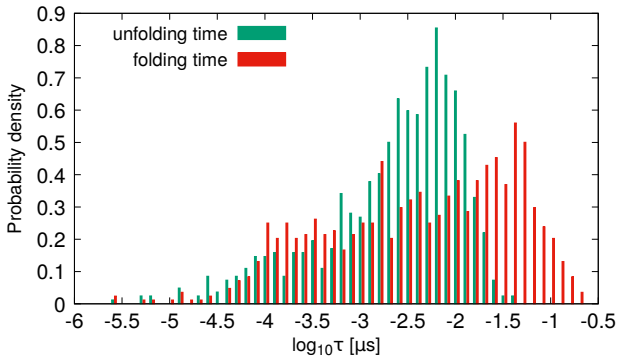


Figure S5: Histograms of the distributions of the unfolding and folding times of the 2JOF variant of the tryptophan cage obtained in simulations run at T = 273 K.

References

1. K. Lindorff-Larsen, S. Piana, R. O. Dror, and D. E. Shaw, Science **334**, 517 (2011).

2. A. Liwo, A. K. Sieradzan, A. S. Karczyńska, E. A. Lubecka, S. A. Samsonov, C. Czaplewski, P. Krupa, and M. Mozolewska, in *Practical Aspects of Computational Chemistry V*, edited by M. S. Leszczynski (Springer, 2021), chap. 2, pp. 31–69.

3. P. Krupa, M. A. Mozolewska, M. Wiśniewska, Y. Yin, Y. He, A. K. Sieradzan, R. Ganzynkowicz, A. G. Lipska, A. Karczyńska, M. Ślusarz, et al., Bioinformatics **32**, 3270 (2016).

4. A. Antoniak, I. Biskupek, K. K. Bojarski, C. Czaplewski, A. Giełdoń, M. Kogut, M. M. Kogut, P. Krupa, A. G. Lipska, A. Liwo, et al., J. Mol. Graph. Model. **108**, 108008 (2021).

5. M. Khalili, A. Liwo, A. Jagielska, and H. A. Scheraga, J. Phys. Chem. B **109**, 13798 (2005).

MOST WIEDZY Downloaded from mostwiedzy.pl