OXFORD

Briefings in Bioinformatics, 19(5), 2018, 853–862

doi: 10.1093/bib/bbx006
Advance Access Publication Date: 18 February 2017
Paper

Molecular dynamics recipes for genome research

Tommaso Biagini*, Giovanni Chillemi*, Gianluigi Mazzoccoli, Alessandro Grottesi, Caterina Fusilli, Daniele Capocefalo, Stefano Castellana, Angelo Luigi Vescovi and Tommaso Mazza

Corresponding author. Tommaso Mazza, IRCCS Casa Sollievo della Sofferenza, Bioinformatics unit, viale Regina Margherita 261, 00198 Rome, Italy. Tel.: +39 06 44160526; Fax: +39 06 44160548; E-mail: t.mazza@css-mendel.it
* These authors contributed equally to this work.

Abstract

Molecular dynamics (MD) simulation allows one to predict the time evolution of a system of interacting particles. It is widely used in physics, chemistry and biology to address specific questions about the structural properties and dynamical mechanisms of model systems. MD earned a great success in genome research, as it proved to be beneficial in sorting pathogenic from neutral genomic mutations. Considering their computational requirements, simulations are commonly performed on high performance computing (HPC) devices, which are generally expensive and hard to administer. However, variables like the software tool used for modeling and simulation or the size of the molecule under investigation might make one hardware type or configuration more advantageous than another or even make the commodity hardware definitely suitable for MD studies. This work aims to shed lights on this aspect.

Key words: molecular dynamics; GPU computing; bioinformatics; genomics

Introduction

Molecular dynamics (MD) simulation is a leading tool of theoretical scientists for the study of the time-dependent motion of atoms and molecules. It is widely used in physics, chemistry and biology to confirm hypotheses and to help answering specific questions about the structural properties and dynamical mechanisms of model systems.

The first simulation of a protein in vacuum was published by Martin Karplus in 1977 [1], while molecules of water were explicitly added to the simulation environment only 11 years later

Tommaso Biagini holds a Master degree in Bioinformatics and a PhD in Cellular and Molecular Biology. His research interests include classical molecular dynamics methods and the oncogenomics.
Giovanni Chillemi holds a PhD in Biochemistry and Molecular Biology and the Italian national scientific qualifications as Associate Professor in Molecular Biology and Biochemistry. He is a Team Leader of the Cineca Specialistic User Support group.
Gianluigi Mazzoccoli is a member of the American Physiological Society and member of the European Biological Rhythm Society.
Alessandro Grottesi holds a Bachelor degree in Biology and a PhD in Biophysics. He is interested in Molecular Dynamics software and Bioinformatics applications. He is a member of the Cineca Specialistic User Support group.
Caterina Fusilli holds a Master degree in Statistics for Biomedicine and a PhD in Economics and Statistics. Her main research topics include shape analysis, cluster analysis and finite mixture models.
Daniele Capocefalo holds a Master degree in Bioinformatics. He is a PhD student in Life Sciences at 'La Sapienza' University of Rome. His research interests include regulatory networks analysis and next-generation sequencing (NGS) data analysis.
Stefano Castellana holds a Master degree in Cellular and Molecular Biology and a PhD in Genetics and Molecular Evolution. His research interests include molecular evolution and NGS data analysis.
Angelo Luigi Vescovi is Scientific Director of the IRCCS Casa Sollievo della Sofferenza, owner of the biotech StemGen, member of the scientific committee of Revert Onlus and professor at Bicocca University of Milan. His research interests include stem cell biology, regenerative medicine and innovative therapies.
Tommaso Mazza holds a Master degree in Computer Science Engineering and a PhD in Computer Science and Biomedical Engineering. His main research focus is on cancer genomics and biomarker discovery through unconventional software and hardware solutions. He leads the Bioinformatics Unit at CSS.
Submitted: 7 October 2016; **Received (in revised form):** 20 December 2016

by Michael Levitt [2]. In 2013, Karplus and Levitt won the Nobel Prize in Chemistry for their pioneering studies on molecular models of biological systems. Analogously, nucleic acids were modeled in vacuum nearly in the same years [3, 4], and in the more realistic aqueous environment right after [5]. In the following years, another important type of environment was modeled: the lipid bilayer [6, 7], essential to simulate membrane proteins [8]. Nowadays, size and length of standard MD simulations have grown till tens of millions of atoms and micro/milliseconds, even though sporadic examples exist that overcome these limits [9, 10].

MD earned a great success in biology. Its success was boosted by the growing availability of genomic information, such as point mutations, provided by increasingly powerful next-generation sequencing and microarray-based platforms. Whole-exome sequencing, among the others, is one of the most popular high-throughput sequencing strategy. It spans roughly 30 Mb [1], i.e. 30 million nucleotides making the coding part of the genome. Large population studies conducted on hundreds of thousands of human exomes taught that about 12–15 000 true single-nucleotide variants (SNVs) can usually be found in every exome, and that their frequencies may change according to the ethnical groups.

Irrespective of the rarity (<1% of frequency in a considered population of individuals, in general) or recurrence of a variant, the assessment of the physiological effects of coding variants is currently a matter of debate, especially when these regard the human being. The first step of this process is the novelty check in a number of SNVs databases. Historically, the Single Nucleotide Polymorphism Database (dbSNP) [11] is the reference repository. Hundreds of independent institutions or consortia stably feed dbSNP with variants of different types (e.g. SNVs, short indels and microsatellites). The consequence is that >150 million variants are currently recorded in dbSNP ver. 149. This number is intended to rapidly grow. Similar databases exist: ESP, containing data of ∼6500 exomes (NHLBI Exome Sequencing Project, Seattle, WA; accessed on August 2016), Exome Aggregation Consortium, with ∼65 000 exomes (Cambridge, MA; accessed on August 2016), Kaviar Genomic Variant Database, with ∼64 000 exomes (accessed on August 2016) [12], and the Haplotype Reference Consortium, with ∼65 000 haplotypes (accessed on August 2016). Generally, but not absolutely, only novel or low-frequent variants are considered as candidate disease-causing mutations. These are usually still too many to be all validated *in vitro* or *in vivo* and, then, their number needs to be further purged of low-priority variants.

Numerous scientists devoted a significant portion of their research to design new algorithms and to develop new tools capable of sorting mutations by their harmfulness. This task was mainly pursued by algorithms borrowed from comparative genomics, structural biology, molecular evolution, biochemistry, cell physiology and pharmacology. These accounted for information like the sequence alignments, peptide ternary or quaternary structures, biochemical properties of molecules, their evolutionary dynamics and sequence conservation through species, to calculate, on-the-fly, the pathogenicity scores of genomic mutations. Dozens of such software packages exist. A few others added the possibility to browse or download precomputed estimations of pathogenicity of all human nuclear [13] and mitochondrial [14] non-synonymous and splice-site variants. Unfortunately, it is common knowledge that none of them correctly predicts the pathogenicity of all variants analyzed in any considered context. Surprisingly, even permuting the results of any subset of software does not increase the detection sensitivity over the top performer alone [15].

The generally suboptimal performance of these predictors pushed the development of other tools that were based on structural and functional considerations. These can be grouped into three subcategories: those that base their predictions on structural information, like Struct2Net [16]; those that base on domain information, available from, e.g. Pfam [17] or Uniprot [18], and on protein interaction evidences, like STRING [19]; those that simulate the temporal trajectories or dynamics of the constituting particles of molecular models. Irrespective of the ways a molecular model is obtained, this last category makes use of two distinct classes of methods.

The classical MD methods process molecules as if they were elementary objects, like soft balls, which represent atoms, and elastic sticks that join pairs of atoms. The Newton's second law rules their overall dynamics. From the knowledge of the initial positions, velocities and individual forces of the atoms in the system, it is possible to calculate their trajectories, namely, the variation over time of their positions, velocities and accelerations. Quantum MD is a significant evolution of the classical MD. It focuses on the interactions between atoms and electrons only, without resorting to experimental data, and accounts for forces that are inferred by theoretical calculations of the electronic structures and that make the system's dynamics evolve over time. Quantum MD was successfully used to tackle several biological issues [20, 21], although it is known to suffer from the important drawback of being highly computational demanding. For this reason, the considered models are usually much simplified or, simply, naturally small. Additionally, hybrid quantum mechanics/molecular mechanics simulations were introduced to model complex systems, while keeping the number of quantum mechanical variables at a minimum. A limit of this approach is the choice of the QM region [22], while it is advantageous when chemical reactions are involved. The balance between advantages and detail loss pushed scientists to make preponderantly use of classical MD over the two.

In this perspective, we focus on tools that implement the classical MD methods and will apply some representatives on official benchmark models and on (un)conventional hardware equipments. These are real-life models, as they come from ongoing research projects or published articles, and differ in size (small, medium and large) and molecules type (protein, RNA and lipids). The aim is to help researchers to assess the feasibility of their MD analyses with the hardware in hand, as well as to make decisions on which hardware to acquire to get the best compromise in terms of cost/performance.

The MD workflow

A standard MD workflow (Figure 1) starts from the selection of a three-dimensional (3D) model, which sticks particle names to 3D coordinates. Depending on the availability of high-resolution experimental data from X-ray crystallography or nuclear magnetic resonance experiments, which are usually stored in the Protein Data Bank (PDB) [23], this step can be simple or really harsh. Resolution is not the only factor to consider when choosing a starting model of a system. The presence and length of unresolved loops, for example, can strongly undermine the goodness of a model.

In the unfortunate case when a model is not available, this is usually inferred by means of two different categories of prediction methods: homology [24, 25] and *ab initio* [26, 27] modeling. The former builds a target model referring to a second

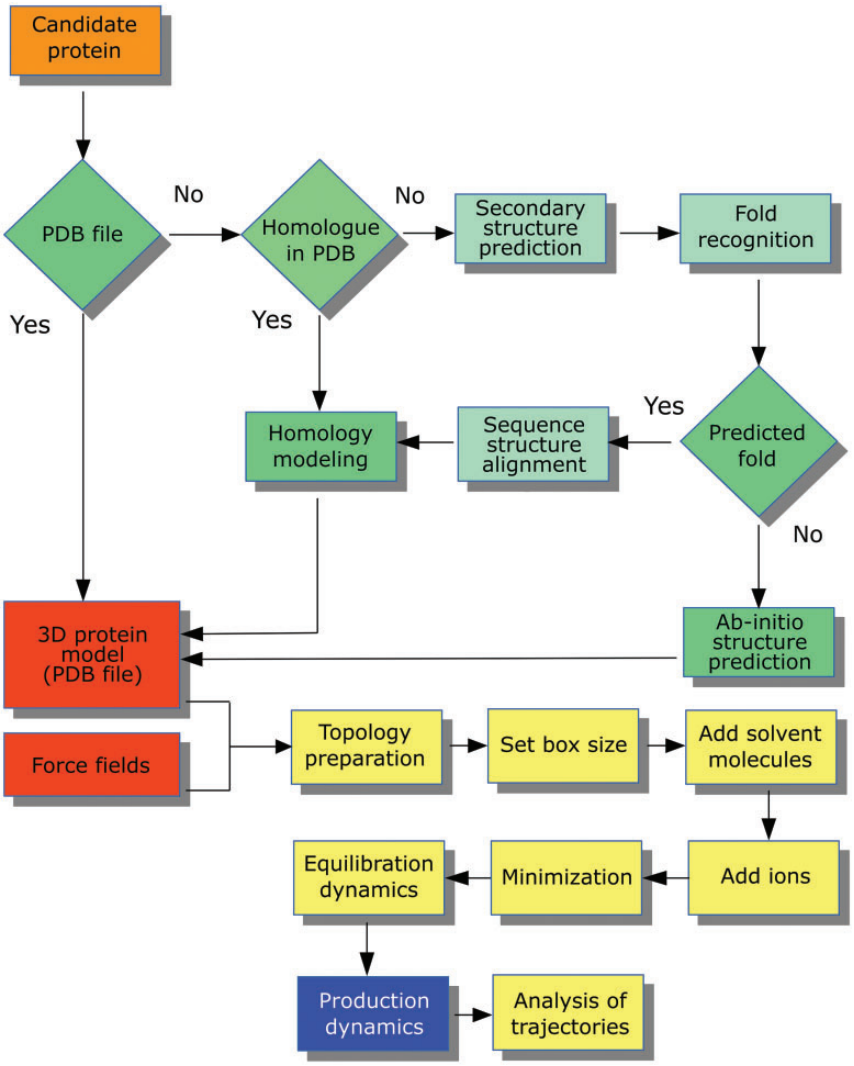

Figure 1. MD workflow.

homologous protein as a template. It can be subclassified in homology modeling (or comparative modeling) and protein threading (or fold recognition). Both are template-based methods, and there are no rigorous boundaries between them in terms of prediction techniques. The former is used whenever homologous proteins of targets are available. Protein threading is used when only a fold-level homology is found. This operating logic is implemented in several software packages: MODELLER [28], I-TASSER (Iterative Threading ASSEmbly Refinement) [29], SWISS-MODEL [30], Phyre2 [31] and MOE (Molecular Operating Environment) (Web address: https://www.chemcomp.com/MOE-Molecular_Operating_Environment.htm). *Ab initio* modeling is usually opted for whenever even a template is not available. In this case, a model is built *de-novo* according to physical laws rather than solved structures. Generally, these last methods are highly computational demanding and are, thus, used to simulate small proteins. Relevant representative softwares are QUARK [32] and ROSETTA [33].

Once obtained the 3D structure, the next step is that of creating a virtual box containing the molecule of interest. Often, the box is filled of solvent and ions, at a specified concentration, with the aim to mimic the total charge of the environment and, eventually, to neutralize it. Ions are approximated as charged point particles and do not possess more detailed electronic structures, which might be relevant for ion-binding sites and transition metals ligated to proteins. However, when using the NaCl molecule, a simple point charge representation is usually sufficient.

Coordinates of the starting model, together with the atoms constituting solvent and ions and with force fields, make the topology of a system. A force field is defined by a mathematical expression that describes the structural properties and vibrational spectra of a system. Force fields are obtained by parameterizing the potential functions from *ab initio* or semiempirical quantum mechanical calculations or by experimental data. Force fields for classical MD, even when obtained by QM calculations, are not physical quantities. Therefore, a force field is only valid for a molecule in the framework for which it was developed, and its reliability should be evaluated based on the reproducibility of experimental data.

A topology file defines which atom is connected with which other and through which kind of chemical bond. Thus, the final topology of a system defines bonds between two atoms, angles among three and dihedrals, when four atoms define an angle between two intersecting planes.

Once defined the topology, a minimization step requires finding an arrangement of the atoms in space that guarantees that the net interatomic forces acting on each particle are as close to zero as possible. The actual aim of this step is to find the closest local, thus not global, minimum energy configuration. To reach a minimum, the system must undergo several iteration cycles, in the order of thousands, where coordinates of atoms are iteratively updated until convergence. Each iteration is also known as a minimization step. There are three major protocols for minimization: steepest descent [34], conjugate gradient [35] and Newton–Raphson [36]. The former method is computationally inefficient, although it is deemed robust and reliable. It is commonly used during the first iteration cycles when the model is far from the minimum configuration. The second method uses information from previous iterations to determine the currently optimal direction toward the minimum. The reiteration of this procedure causes a drastic improvement in performance near the minimum. The latter method assumes parabolic potential energy, and in the ideal case where all potentials are exactly parabolic, it determines the minimum in a single step. The computational improvement is dramatic near the minimum, but when systems are far from the minimum, the algorithm could diverge toward high-energy structures.

A minimized system undergoes equilibration. This stage is required because the input structure is typically not within the equilibrium phase space of the simulation conditions. For a few iterations, the system is weakly coupled to a heath bath, i.e. a system of unlimited thermal capacity, in which the temperature remains constantly in contact with the system of interest, whose temperature is gradually increased until reaching the desired value. Given a typical time step of 1 fs, the equilibration step lasts at least 5 ps (5000 time steps), but more frequently 10 or 20 ps.

The dynamics of a model is here ready to be simulated. The classical name of this step is *production dynamics* (Figure 1). We will particularly focus on this task the hereafter.

Materials and methods

MD software

This study made use of Amber 16 [37], NAMD version 2.11 [38] and GROMACS 5.1.2 [39] for the production dynamics step, as they are the most used tools in the analysis of biomolecular systems.

The Amber Molecular Dynamics Package includes different software modules that allow to set up, perform and analyze MD simulations. The name Amber refers to a family of classical force fields of natural biomolecules and to the software tools designed to parameterize nonstandard complex molecules. Amber runs on the most widely used Unix platforms and compilers and is written in Fortran 90 and C languages. Today, it is distributed in two parts: Ambertools and Amber. The former is free of charge, and its components are mostly released under the GNU General Public License (GPL). Amber is available under a Site License Agreement, which includes the source code, currently priced at US $500 for noncommercial and US $20 000 for commercial organizations. One of the main programs included in Amber is LEaP. This takes in input force fields and atom coordinates as PDB files (.pdb), to produce the files necessary for the preparation of the topology. There are two versions of this program, a graphical interface called xleap and a command-line version called tleap. Amber consists of three MD simulation tools: Sander, Pmemd and Pmemd.cuda. Sander is the central parallel simulation program and provides algorithms for energy minimization and MD; PMEMD is a re-implementation of Sander, which minimizes the exchange of data between processors and thus performs better if used in parallel by >8–16 computing processors; Pmemd.cuda is an extension of the Fortran implementation of PMEMD, with calls to specific CUDA kernels for graphics processing unit (GPU) acceleration.

NAMD is a parallel MD software developed using the Charm++ parallel programming model. It is known to simulate large systems, in the order of millions of particles, efficiently. Source code and binaries are available freely for noncommercial purposes. It makes use of the psfgen plug-in to build topologies starting from PDB input files. NAMD works with the force fields of Amber and CHARMM and 'understands' their parameters and file formats. It is coupled with the molecular graphics program VMD, which provides tools tailored to NAMD, in addition to standard visualization functions. NAMD and VMD use GPUs to accelerate many computationally demanding functions.

GROMACS is a free, open-source MD software package, designed for the simulation of proteins, lipids and nucleic acids. It is released under the GNU LGPL since version 4.6. It is equipped with a rich library of trajectory analysis tools and scripts for the conversion of input file formats. It supports a variety of force fields and is known for its extreme flexibility and versatility. GROMACS can be executed in parallel on central processing unit (CPUs) and GPUs using the Message Passing Interface (MPI) or threads. Since version 4.6, all available compute resources can be exploited by a hybrid acceleration method that allows the use of multiple CPUs and GPUs, simultaneously. Under a non-GPL license, GROMACS is used in the Folding@home distributed computing project for the simulation of protein folding.

Biomolecular benchmark systems

Each MD tool was used to analyze three test cases, which mainly differed by size, i.e. atoms count, and molecular type.

The smallest system (~60 000 atoms) is a protein, named SUFU negative regulator of hedgehog signaling, which is bound in PDB to a member of the GLI family zinc finger (PDB ID: 4BLD; [40]; Figure 2A). The system was modeled from scratch following the steps described in the workflow in Figure 1 and simulated with the three MD tools. In detail, the topology of this system was built using Amber tleap. The resulting model was embedded in a box, extending up to 12 Å and solvated using the TIP3P water model. tleap was also used to add counter ions and thus to neutralize the overall charge of the model. The topology, consisting of 59 140 atoms, was suitable also for NAMD, being this compatible with the topology file formats generated by Amber. We needed, instead, to build a dedicated topology for GROMACS. To do that, we followed the workflow described in Supplementary File 1. The resulting topology was made up of 60 852 atoms. Both topologies of the small system were firstly energy minimized, then equilibrated for ~5 ns, by a time step of 1 fs. Finally, the equilibrated systems were simulated for 50 ps, by a time step of 2 fs (25 000 steps).

Medium (~100 000 atoms) and large (~1 000 000 atoms) systems were chosen instead, among the benchmark models, already minimized and equilibrated, available from the Web sites of the considered MD tools.

The selected medium systems were (i) one of the serine proteases of the coagulation system, the Factor IX protein NPT (isothermal–isobaric ensemble) in TIP3P water (90 906 atoms) for Amber (Figure 2C); (ii) the apolipoprotein A-I, which is the

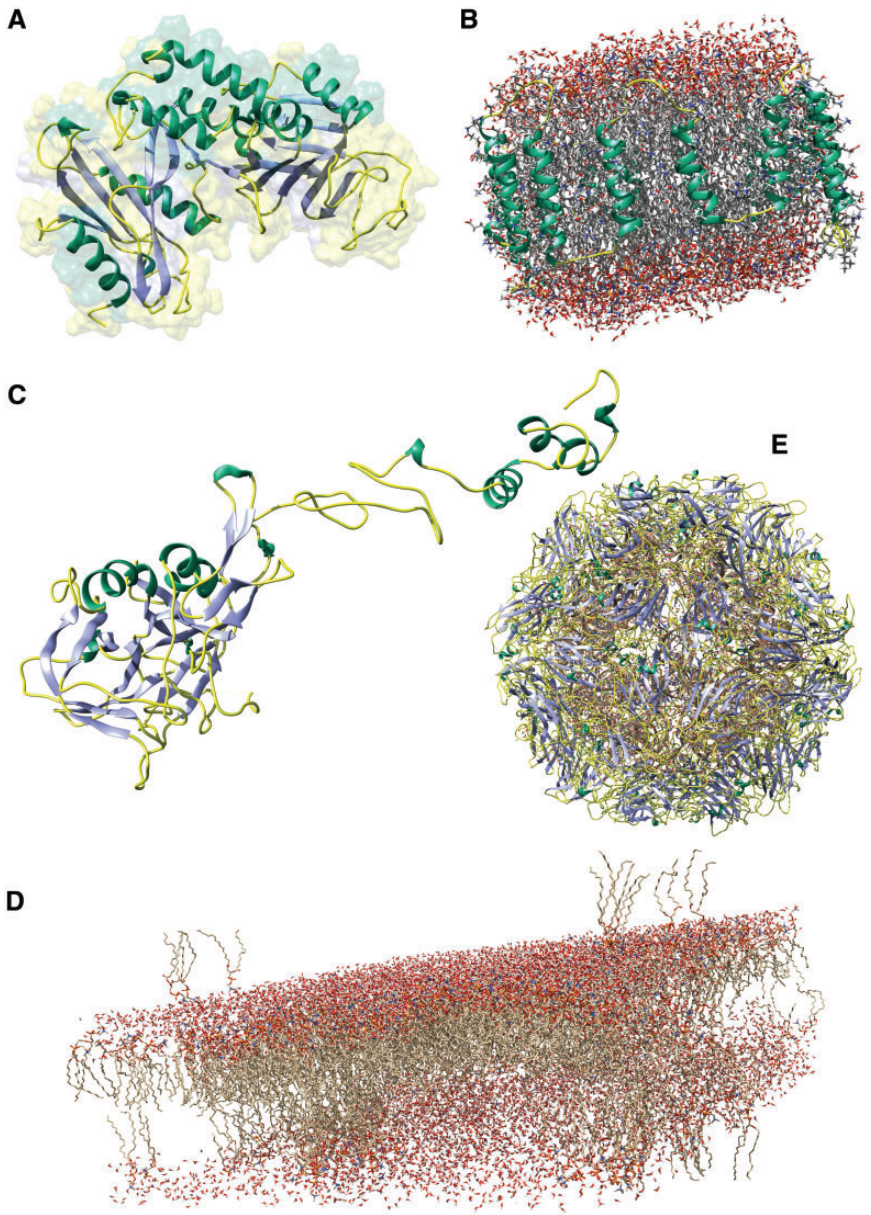

Figure 2. 3D structure of (**A**) human suppressor of fused (SUFU)-GLI3p complex, (**B**) factor IX protein, (**C**) apolipoprotein A-I, (**D**) hydrated mixed DPPC bilayer lipid and (**E**) STMV generated with UCSF Chimera.

primary protein constituent of HDL, solvated in explicit water (92 224 atoms) for NAMD (Figure 2B); (iii) the hydrated mixed 1,2-dipalmitoyl-sn-glycero-3-phosphocholine (DPPC) bilayer lipid (121 856 atoms), downloaded from ftp://ftp.gromacs.org/pub/benchmarks/gmxbench-3.0.tar.gz, for GROMACS (Figure 2D).

The large system was the satellite tobacco mosaic virus (STMV) for all three MD tools. The Amber model was made of 1 067 095 atoms and taken from http://ambermd.org/Amber16_Benchmark_Suite.tar.bz2. The NAMD model (1 066 628 atoms) was downloaded from http://www.ks.uiuc.edu/Research/namd/performance.html. The GROMACS topology was retrieved from [41] and included 30 proteins (i.e. half a STMV capsid) and some RNA fragments. Initial coordinates were taken from PDB (PDB ID: 1A34). The model was solvated in 275 415 molecules of water, 887 $Na+$ and 827 $Cl-$. The total number of atoms was 914 499 (Figure 2E). Nonbonded settings for all systems were set to be comparable, as specified in the Supplementary File 2.

Measures of performance

For each system, we measured the wall-clock time (WCT), i.e. the amount of elapsed time at task completion. For each system, the WCT of the slowest simulation of all performed simulations on CPU and GPU and on any number of computing CPU cores/GPUs, was called $t_{slowest}$. The WCT obtained by N computing CPU cores or GPUs was specified as t_N. We plotted speedup-like charts (here referred to as speedup for simplicity), according to the formula $t_{slowest}/t_N$.

Computational resources

MD simulations were run on a cluster of three computing nodes, each equipped with 4× AMD Opteron® processor 6172 @2.1GHz, with 12 CPU cores and 256 GB of RAM, for 144 working cores. Nodes are interconnected over InfiniBand through Mellanox/Intel host bus adapters and network switches. MD simulations were also run

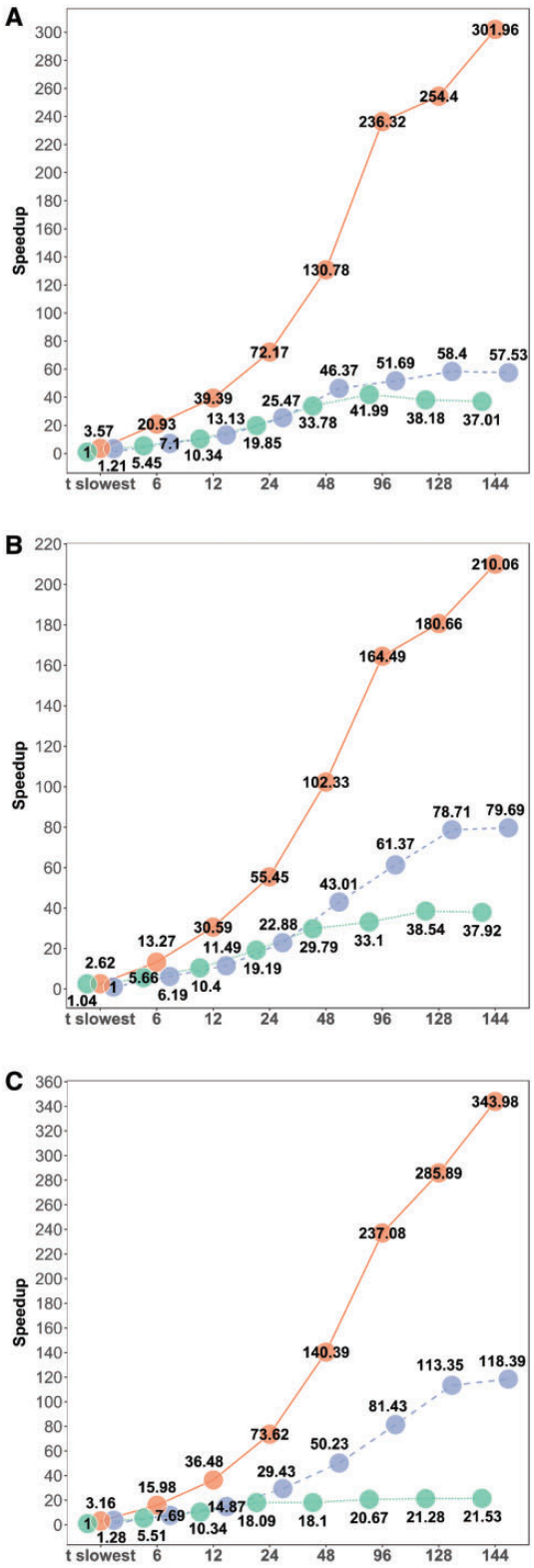

Figure 3. Speedup plots for the small (A), medium (B) and large (C) systems simulated on CPU nodes. Amber with dotted line, NAMD with dashed line and GROMACS with plain line.

on a 2X Intel Xeon E5620 @2.40GHz workstation (2 processors, 4 computing cores and 16 threads, with hyper-threading enabled), equipped with 12 GB of RAM and 4× NVIDIA Tesla C2070 GPUs.

To make this study repeatable, command lines, parameters and configuration files were made available in Supplementary File 2.

Results

Production dynamics performance

Each system was simulated three times. WCTs varied <5% through each triplet. All benchmarks presented here refer to the best of the three, in terms of WCTs. Simulations were run on 1–144 AMD CPU cores, as well as on 1–16 CPU cores of an Intel workstation (hyper-threading enabled) and from one to four GPUs. The slowest simulation, i.e. $t_{slowest}$, was achieved by Amber with the small and large systems and by NAMD with the medium system, when simulating each system with one AMD computing core.

Although the benchmark systems were carefully chosen and analyzed by state-of-the-art methods, a detailed comparison of the performance of these MD tools was not only out of our scope, but in this case, not even possible in a rigorous manner. All considerations reported in the paragraphs below concern commodity hardware, which was intentionally chosen for this study. High-cost and more performing hardware could overturn our results. Furthermore, important biases may also originate from the different molecular types of the medium systems, as well as from the fact that each tool supports some specific features that are not provided by the others as, for example, the way to integrate the time steps and to control the temperature and pressure. The only two variables considered in this work are the size of the systems (small, medium and large), and the size and type of processing units [small to medium sized conventional (CPU) or unconventional (GPU) hardware architectures]. With this in mind, we reasoned in terms of scalability of one over the other.

CPU-based computation

Figure 3 reports speedups for small (A), medium (B) and large systems (C), relative to the simulation runs on CPU cores.

GROMACS was the best performer on a single core with the small (3.57×), medium (2.62×) and large (3.16×) systems, as well as on all other CPU configurations. The serial execution of NAMD run faster than that of Amber for the small and large systems and was the slowest for the medium system. The speedup of Amber grew up until 96 and 128 cores, significantly, with the small and medium system, before reaching a plateau. For the large system, a plateau was reached earlier with 24 cores.

The NAMD code confirmed to be at ease with large systems run on big HPC infrastructures. It is not surprising, in fact, that NAMD did not perform as well with the small system as it did with the large one. It continued scaling until 144 cores and potentially even more with the addition of further CPU cores. The speedup with the small system increased until 48 cores, before stopping scaling soon later. However, performance slightly improved until 128 cores and definitely decreased with 144 cores.

GPU-based computation

The same systems were simulated on a GPU- and Intel CPU-equipped workstation with hyper-threading enabled, i.e. able to spawn until 16 parallel tasks. GPU cards were set up to communicate by MPI, and one MPI process was systematically mapped to one GPU. Figure 4 reports speedup values for small (A), medium (B) and large (C) systems. Amber performed best with all three systems, regardless of the number of GPUs used. The best

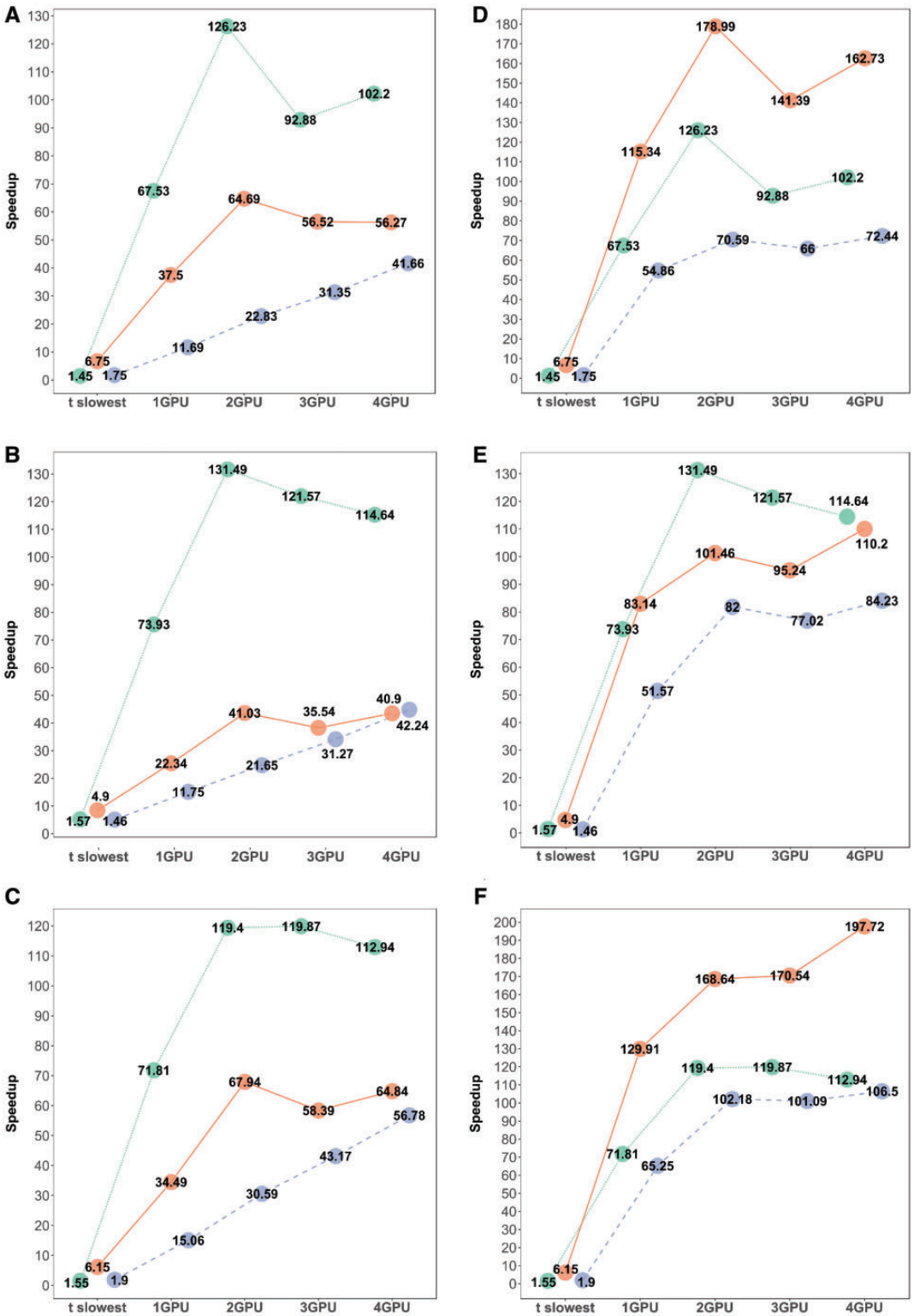

Figure 4. Speedup plots for the small (**A**), medium (**B**) and large (**C**) systems simulated on GPU nodes and mapping one MPI process to one GPU. Small (**D**), medium (**E**) and large (**F**) systems simulated on GPU nodes with optimization enabled for NAMD and GROMACS. Amber with dotted line, NAMD with dashed line and GROMACS with plain line.

performance was achieved with two GPUs with the medium system (131.49X). The use of the other two was penalizing because the motherboard hosting the four GPU cards only allows a two-way peer-to-peer communication at full 16× speed. The speedup achieved with only one GPU outperformed systematically the best records of the other tools obtained with any other configuration and with any system. GROMACS stopped scaling when run on more than two GPUs, and reached a seamless

Table 1. WCTs obtained with NAMD, with multi-threading enabled

HW configuration	Small system	Medium system	Large system
1 GPU			
1 thread	2283.94	2856.59	31 901.20
2 threads	1162.26	1548.38	15 563.51
4 threads	669.36	821.65	8850.89
8 threads	**486.51**	**650.86**	**7365.68**
16 threads	513.27	669.05	7 618 89
2 GPUs			
2 threads	1169.16	1550.55	15 710.72
4 threads	637.22	805.01	8404.94
8 threads	402.43	489.83	5242.70
16 threads	**378.13**	**409.34**	**4703.44**
3 GPUs			
3 threads	851.30	1073.30	11 132.77
6 threads	485.15	584.28	6339.45
12 threads	470.80	567.95	5653.58
15 threads	**404.42**	**435.77**	**4754.21**
4 GPUs			
4 threads	640.76	794.65	8464.19
8 threads	395.24	475.96	5025.46
16 threads	**388.05**	**398.47**	**4512.60**

Note. WCTs are expressed in seconds. The best performances are highlighted in bold.

plateau. NAMD, instead, scaled constantly, as already verified with the CPUs.

GROMACS and NAMD provide also some optimization options that better exploit the GPUs when they are underused. In detail, NAMD is able to map MPI processes to multiple CPU cores, and thus to process several GPU kernels on the same GPU, concurrently. GROMACS further allows hybrid parallelism, namely it can delegate the computation of the OpenMP multi-threaded bonded forces and PME long-range electrostatic calculations to the CPU cores, while dealing with all nonbonded forces by the GPUs. Amber just uses a single core per GPU and does not provide any optimization strategy.

Figure 4D–F reports benchmarks for the same systems, with optimization enabled for GROMACS and NAMD. The values reported for these two tools represent the best speedups achieved with the best combination of number of threads and GPUs. GROMACS inspected the hardware available at run time and did its best to make fairly efficient use of the whole node. NAMD was finely tuned, and, generally, it obtained the best performance with the maximum available threads (cf. Table 1).

These plots confirm that the optimization strategies were generally beneficial. In terms of scalability, GROMACS achieved the best performance with four GPUs and with the large system. Amber continued to be the best with only two GPUs and with the medium system. NAMD was generally slower than the others, obtaining its best performance (speedup of 72.44 for the small system, 84.23 and 106.49 for the medium and large systems, respectively) when exploiting all available hardware components, i.e. 16 threads and 4 GPUs (eight threads and four GPUs with the small system).

OpenCL versus CUDA

We have compared CUDA with an alternative general-purpose parallel programming paradigm, OpenCL, on our systems. To this end, we have ran GROMACS, which supports both paradigms, on the small, medium and large systems, using one to four GPUs. As expected, all simulations performed well with the CUDA compiled version of GROMACS that exhibited systematic lower WCTs as compared with those obtained with the OpenCL compiled counterpart. The benchmark data are reported in Table 2. In particular, CUDA simulations run 1.2–1.5 times faster than the OpenCL ones for all systems. This is mainly due to the different implementations of the off-load computation of the forces and energies that are generally more efficient on CUDA-based kernels.

AMD versus Intel

After disabling the hyper-threading on the GPU workstation, we directly compared the performance of AMD and Intel processors. In detail, all simulations were performed using one, two, four and eight CPU cores. All MD codes were generally run more efficiently on the Intel hardware, thereby exhibiting a speedup increase that ranged from 1.37× to 2.18×, as compared with the AMD WCTs (Supplementary Table S1).

DISCUSSION

Our results suggest that Amber performs differently on CPU and GPU. Its performance is in line with those of GROMACS and NAMD, when run on devices with low core counts, like common workstations. However, its speedup is lower than that of the other codes. Instead, it greatly benefits from the GPU acceleration. It is interesting to note that the best speedup obtained with the CPU cores by Amber was exceeded with only one GPU for all three systems. Two GPUs working together made Amber the best performer with the medium system and the second best with the small and large systems. The use of additional two GPUs did not contribute to the increase of the speedups. As anticipated, this is mainly due to the fact that Amber 16 uses peer-to-peer communication to provide optimal multi-GPU scaling, and no standard motherboards exist that support more than two-way peer-to-peer at full 16× speed, included our

Table 2. WCTs obtained with the OpenCL and CUDA-compiled versions of GROMACS

HW configuration	OpenCL	CUDA
Small system		
1 GPU	370.017	231.404
2 GPUs	232.635	149.124
3 GPUs	217.334	188.775
4 GPUs	196.346	164.016
Medium system		
1 GPU	580.22	403.72
2 GPUs	411.92	330.80
3 GPUs	439.39	352.40
4 GPUs	359.96	304.57
Large system		
1 GPU	5471.30	3699.39
2 GPUs	3804.81	2849.80
3 GPUs	3281.93	2817.97
4 GPUs	3057.13	2430.60

Note. WCTs are expressed in seconds.

workstation. The Amber team faced this issue in collaboration with the Exxact Corporation. They designed a hardware solution that hosted up to 10 GPUs on a single PCI Express root hub. This allowed the GPUs to communicate with each other without having to pass through the CPU chipset, thereby reducing the communication latency. The PCI Express is then reserved to I/O operations only. With this setting, the best possible performance can be achieved when running different jobs on different GPUs. Another important advantage of this configuration is that the computing performance on GPU is not more constrained by the CPU characteristics and, then, even commodity hardware hosting GPU cards can be exploited. A more obvious, but not less important, benefit is that all individual calculations run on as many GPUs can be run without interfering with each other's performance.

The scalability of NAMD on CPU cores was higher on the large systems. However, its performance improved significantly with the use of more CPU cores. NAMD benefited of the use of GPU accelerators, especially when each GPU was shared by multiple CPU cores. In this setting, NAMD automatically and equally distributed threads among the available GPUs on a node. All three systems were simulated faster when eight threads were spawned for one GPU. When using more than one GPU, the optimal solution was that of spawning all available threads, i.e. 16. The speedup of NAMD obtained with four GPUs was in line with that obtained with 48 CPU cores on all three systems and was achieved with all four GPUs working together, when one process was mapped to any one GPU. Mapping all available processes to the GPUs made the performances to exceed those obtained with 144 CPUs. The results of this study is that NAMD may greatly benefit from modern technology and that its scalability is good up to the maximum number of nodes. This is confirmed by some official benchmarks available at http://www.hpc.cineca.it/content/namd-benchmark where the maximum number of CPU cores is 2000.

GROMACS shined for its extremely high scalability capacity with all systems on CPU cores. On the other hand, it did not stand out when run on GPUs. For example, the speedups achieved by four GPUs with all systems were close to those obtained with 48 CPU cores and even lower than those obtained with 96, 128 and 144 CPU cores. Its hybrid parallel configuration (OpenMP/MPI) performed systematically better than the single-threaded setup. However, this should not be considered as a rule of thumb. The fact that the hybrid configuration allowed GROMACS to efficiently distribute the computational work on the CPU cores used in this study might not be always true, as MPI and OpenMP might exhibit different parallel scaling behaviors on other hardware infrastructures and MD systems.

Take-home messages

Amber is generally a good choice for the simulation of small/medium systems and particularly when only commodity hardware is available or when the hardware resources are limited. The use of one, or better, two GPUs generally allows achieving high performances.

GROMACS behaves well on small systems, when run on GPUs. However, it excels with big systems when high numbers of CPU cores are available.

NAMD is the second-best option when running on CPUs. It is the leader in flexibility, as it 'understands' all available topology file formats. Contrary to Amber, it seems to benefit from modern CPU computing technology and to significantly improve its performance on devices equipped with high numbers of CPUs.

Key Points

- The assessment of the putative pathogenicity of genomic mutations is a critical task.
- Dozens of software packages exist that predict the putative pathogenicity of genomic variants.
- MD simulation is a leading tool of theoretical scientists for the study of the time-dependent motion of atoms and molecules.
- GPU computing is the use of GPUs together with CPUs to accelerate domain-specific applications.
- MD simulations are run on GPU or CPU hardware for the discovery of harmful mutations.

Supplementary Data

Supplementary data are available online at http://bib.oxfordjournals.org/.

Funding

This work was supported by a grant awarded to A.L. Vescovi by "Associazione Italiana per la Ricerca sul Cancro" (AIRC) (IG-14368) and by a grant awarded to Tommaso Mazza by Ricerca Corrente 2016 granted by the Italian Ministry of Health and by the 5x1000 voluntary contributions.

References

1. McCammon JA, Gelin BR, Karplus M. Dynamics of folded proteins. *Nature* 1977;**267**:585–90.
2. Levitt M, Sharon R. Accurate simulation of protein dynamics in solution. *Proc Natl Acad Sci USA* 1988;**85**:7557–61.
3. Levitt M. How many base-pairs per turn does DNA have in solution and in chromatin? Some theoretical calculations. *Proc Natl Acad Sci USA* 1978;**75**:640–4.
4. Zhurkin VB, Lysov YP, Florentiev VL, *et al*. Torsional flexibility of B-DNA as revealed by conformational analysis. *Nucleic Acids Res* 1982;**10**:1811–30.
5. Westhof E. Water: an integral part of nucleic acid structure. *Annu Rev Biophys Biophys Chem* 1988;**17**:125–44.

6. Pastor RW. Molecular dynamics and Monte Carlo simulations of lipid bilayers. *Curr Opin Struct Biol* 1994;**4**:486–92.
7. Tieleman DP, Marrink SJ, Berendsen HJ. A computer perspective of membranes: molecular dynamics studies of lipid bilayer systems. *Biochim Biophys Acta* 1997;**1331**:235–70.
8. Domene C, Bond PJ, Sansom MS. Membrane protein simulations: ion channels and bacterial outer membrane proteins. *Adv Protein Chem* 2003;**66**:159–93.
9. Hu XH, Hong L, Smith MD, *et al*. The dynamics of single protein molecules is non-equilibrium and self-similar over thirteen decades in time. *Nat Phys* 2016;**12**:171–4.
10. Perilla JR, Goh BC, Cassidy CK, *et al*. Molecular dynamics simulations of large macromolecular complexes. *Curr Opin Struct Biol* 2015;**31**:64–74.
11. Sherry ST, Ward MH, Kholodov M, *et al*. dbSNP: the NCBI database of genetic variation. *Nucleic Acids Res* 2001;**29**:308–11.
12. Glusman G, Caballero J, Mauldin DE, *et al*. Kaviar: an accessible system for testing SNV novelty. *Bioinformatics* 2011;**27**:3216–17.
13. Liu X, Jian X, Boerwinkle E. dbNSFP: a lightweight database of human nonsynonymous SNPs and their functional predictions. *Hum Mutat* 2011;**32**:894–9.
14. Castellana S, Ronai J, Mazza T. MitImpact: an exhaustive collection of pre-computed pathogenicity predictions of human mitochondrial non-synonymous variants. *Hum Mutat* 2015;**36**:E2413–22.
15. Castellana S, Mazza T. Congruency in the prediction of pathogenic missense mutations: state-of-the-art web-based tools. *Brief Bioinform* 2013;**14**:448–59.
16. Hosur R, Peng J, Vinayagam A, *et al*. A computational framework for boosting confidence in high-throughput protein-protein interaction datasets. *Genome Biol* 2012;**13**:R76.
17. Finn RD, Coggill P, Eberhardt RY, *et al*. The Pfam protein families database: towards a more sustainable future. *Nucleic Acids Res* 2016;**44**:D279–85.
18. UniProt C. UniProt: a hub for protein information. *Nucleic Acids Res* 2015;**43**:D204–12.
19. Franceschini A, Szklarczyk D, Frankild S, *et al*. STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res* 2013;**41**:D808–15.
20. Guo X, Yuan H, An B, *et al*. Ultrafast excited-state deactivation of 9-methylhypoxanthine in aqueous solution: a QM/MM MD study. *J Chem Phys* 2016;**144**:154306.
21. Vianello R, Domene C, Mavri J. The use of multiscale molecular simulations in understanding a relationship between the structure and function of biological systems of the brain: the application to monoamine oxidase enzymes. *Front Neurosci* 2016;**10**:327.
22. Kulik HJ, Zhang JY, Klinman JP, *et al*. How large should the QM region be in QM/MM calculations? The case of Catechol O-Methyltransferase. *J Phys Chem B* 2016;**120**:11381–94.
23. Berman HM, Westbrook J, Feng Z, *et al*. The Protein Data Bank. *Nucleic Acids Res* 2000;**28**:235–42.
24. Zhang L, Zhu HW, Wang QA, *et al*. Homology modeling, molecular dynamic simulation and docking studies of cyclin dependent kinase 1. *J Mol Model* 2011;**17**:219–26.
25. Xiao JF, Li ZS, Sun M, *et al*. Homology modeling and molecular dynamics study of GSK3/SHAGGY-like kinase. *Comput Biol Chem* 2004;**28**:179–88.
26. Zhang Y. Progress and challenges in protein structure prediction. *Curr Opin Struct Biol* 2008;**18**:342–8.
27. Moult J, Fidelis K, Kryshtafovych A, *et al*. Critical assessment of methods of protein structure prediction (CASP)–round x. *Proteins* 2014;**82(Suppl 2)**:1–6.
28. Webb B, Sali A. Comparative protein structure modeling using MODELLER. *Curr Protoc Bioinformatics* 2016;**54**:5.6.1–5.6.37.
29. Yang J, Yan R, Roy A, *et al*. The I-TASSER suite: protein structure and function prediction. *Nat Methods* 2015;**12**:7–8.
30. Schwede T, Kopp J, Guex N, *et al*. SWISS-MODEL: an automated protein homology-modeling server. *Nucleic Acids Res* 2003;**31**:3381–5.
31. Kelley LA, Mezulis S, Yates CM, *et al*. The Phyre2 web portal for protein modeling, prediction and analysis. *Nat Protoc* 2015;**10**:845–58.
32. Xu D, Zhang Y. Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field. *Proteins* 2012;**80**:1715–35.
33. Raman S, Vernon R, Thompson J, *et al*. Structure prediction for CASP8 with all-atom refinement using Rosetta. *Proteins* 2009;**77(Suppl 9)**:89–99.
34. Petrova SS, Solov'ev AD. The origin of the method of steepest descent. *Hist Math* 1997;**24**:361–75.
35. Dongarra JJ, Duff IS, Sorensen DC, *et al*. *Solving Linear Systems on Vector and Shared Memory Computers*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1991.
36. Ermer O. Calculation of molecular properties using force fields. Applications in organic chemistry. In: *Bonding Forces*. Springer Berlin Heidelberg, 2005, 161–211.
37. Case DA, Cheatham TE, III, Darden T, *et al*. The Amber biomolecular simulation programs. *J Comput Chem* 2005;**26**:1668–88.
38. Phillips JC, Braun R, Wang W, *et al*. Scalable molecular dynamics with NAMD. *J Comput Chem* 2005;**26**:1781–802.
39. Abraham MJ, Murtola T, Schulz R, *et al*. GROMACS: high performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* 2015;**1**:19–25.
40. Cherry AL, Finta C, Karlstrom M, *et al*. Structural basis of SUFU-GLI interaction in human Hedgehog signalling regulation. *Acta Crystallogr D Biol Crystallogr* 2013;**69**:2563–79.
41. Larsson DS, van der Spoel D. Screening for the location of RNA using the chloride ion distribution in simulations of virus capsids. *J Chem Theory Comput* 2012;**8**:2474–83.