OMMProtocol: A command line application to launch molecular dynamics simulations with OpenMM

Jaime Rodríguez-Guerra Pedregal,[1] Lur Alonso-Cotchico,[1] Lorea Velasco-Carneros,[2] and Jean-Didier Maréchal.[1*]*

[1] InsiliChem, Departament de Química, Universitat Autònoma de Barcelona, 08193 Barcelona, Spain.
[2] Biofisika Institute (CSIC, UPV/EHU), Department of Biochemistry and Molecular Biology, Universidad del País Vasco / Euskal Herriko Unibertsitatea, 48940 Leioa, Spain

ABSTRACT

OpenMM is a free and GPU-accelerated Molecular Dynamics (MD) engine written as a layered and reusable library. This approach allows maximum flexibility to configure MD simulations and develop new molecular mechanics (MM) methods. However, this powerful versatility comes at a cost: the user is expected to write Python scripts to run a simulation. OMMProtocol aims to fill this gap by stitching OpenMM and additional third-party modules together, providing an easy way to create an input file to configure a full multi-stage simulation protocol, from minimization to equilibration and production. OMMProtocol is LGPL-licensed and freely available at https://github.com/insilichem/ommprotocol.

Introduction

Structural biology and physical chemistry have been using computer simulation for decades now. One of the most popular methods is Molecular Mechanics (MM) and its maximum applicative expression, classical Molecular Dynamics (MD). Nowadays, there are numerous well-established software suites to perform MD simulations like Amber,[1] CHARMM,[2] Gromacs,[3] NAMD[4] or TINKER.[5] However, novel packages keep appearing. Those novel implementations mainly take advantage of new hardware improvements and more particularly the availability of general purpose graphic processing units (GPGPUs) at consumer level prices, which allow significant speedup of the software performance through massive parallelization at a cheap cost. While the traditional actors of the MD field have been implementing GPU acceleration for years now, a new project has been attracting significant attention: the OpenMM toolkit.[6]

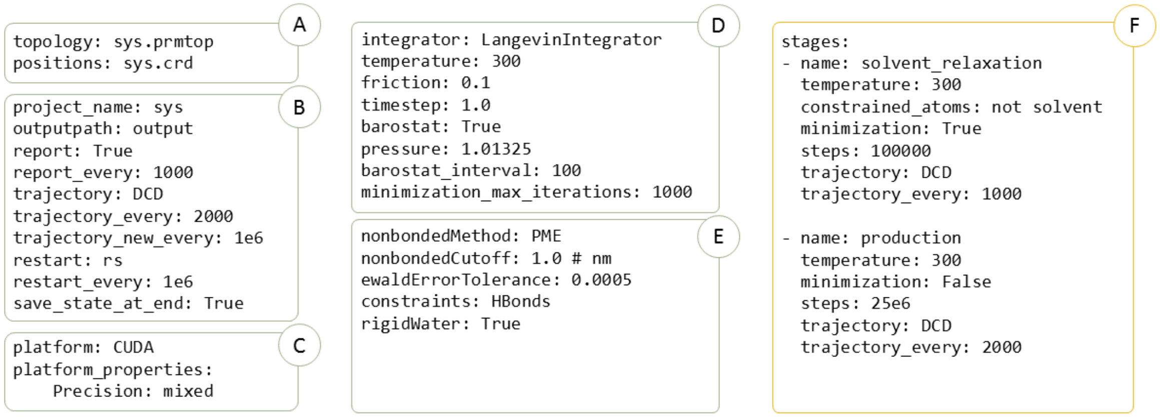
Built for both multiplatform performance and development flexibility, OpenMM provides an open-source layered library that allows easy reutilization in external software. It also counts with a thriving environment of satellite packages, like MDTraj[7] for trajectory analysis, ParmEd[8] for advanced handling of structures and parameters or PDBFixer[9] for structure sanitization. Despite its versatility, OpenMM lacks a command-line executable. While it is true that OpenMM contains a high-level Python interface to setup MD simulations, it also expects the user to write or modify Python scripts.[10] In fact, the OpenMM team offers some tools to ease the script preparation process, like an online input creator[11] or the more recent *openmm-setup*[12] local web app. However, they still require running a Python script. The project *openmm-cmd*[13] does offer a command-line interface, but it was last updated in 2014 and only wraps the OpenMM *app* module with command line arguments.

Here, we present OMMProtocol, a Python 2.7/3.5+ application which combines OpenMM, MDTraj, ParmEd and openmoltools in a straightforward executable to easily configure and deploy OpenMM-based MD simulations.

Methods

OMMProtocol is built around three main ideas: (1) Only a single, readable file should be needed to easily setup a full, GPU-accelerated, reproducible MD simulation without programming skills; (2) Users from other MD suites should be able to use it without disrupting their existing workflows; (3) Reported data should be well-labeled and properly named without user intervention.

The protocol file. The name OMMProtocol stands for OpenMM Protocol. This means that, in addition to the initial conditions of the structure (topology, coordinates, velocities, periodic box vectors…) and the treatment of the simulated universe (forcefield parameters, integrators, temperature, barostat...), the user can specify the different stages of the protocol, like minimization, equilibration, simulated annealing, or production. Each stage can override most of the global parameters (number of steps, temperature, constrained atoms if any, format of the output files…), which confers maximum flexibility when designing a protocol. Since each protocol is a plain-text YAML[11] file, the same file can be shared and edited for other calculations: only the structure files (topology and positions, most of the time) need to be modified (see figure 1). All the available keys are reported in the accompanying Technical Documentation. For advanced users, the Jinja[12] templating engine has been implemented, which can greatly simplify protocols involving highly-similar stages (equilibration, for example).



```
topology: sys.prmtop
positions: sys.crd
```

```
project_name: sys
outputpath: output
report: True
report_every: 1000
trajectory: DCD
trajectory_every: 2000
trajectory_new_every: 1e6
restart: rs
restart_every: 1e6
save_state_at_end: True
```

```
platform: CUDA
platform_properties:
    Precision: mixed
```

```
integrator: LangevinIntegrator
temperature: 300
friction: 0.1
timestep: 1.0
barostat: True
pressure: 1.01325
barostat_interval: 100
minimization_max_iterations: 1000
```

```
nonbondedMethod: PME
nonbondedCutoff: 1.0 # nm
ewaldErrorTolerance: 0.0005
constraints: HBonds
rigidWater: True
```

```
stages:
- name: solvent_relaxation
  temperature: 300
  constrained_atoms: not solvent
  minimization: True
  steps: 100000
  trajectory: DCD
  trajectory_every: 1000

- name: production
  temperature: 300
  minimization: False
  steps: 25e6
  trajectory: DCD
  trajectory_every: 2000
```

Figure 1: OMMProtocol files are formatted in YAML. Configuration keys can be specified in any order, but they have been grouped in this figure for convenience. Section A contains the structural data of the system to be simulated: the topology key is always required. Section B groups options related to file output. Section C controls the hardware to be used. Section D and E specify the conditions of the simulation. Finally, section E lists all the stages to be simulated in this protocol. Each entry, marked with a starting dash, can override any of the global options specified in sections B-E. Usually, only constraints, minimization, temperature and simulated steps will be modified here, since every other parameter is normally constant during the full protocol.

Input and output compatibility. In addition to PDB/PDBx files, OpenMM can open files coming from Amber, Gromacs, Desmond and CHARMM suites. While these formats are very different in syntax and contents, OMMProtocol provides a single interface to handle them all with a clear order of precedence (see figure 2). The core of this interface is the topology container, on top of which the user can choose to load coordinates, velocities, box vectors, or forcefield parameters, if needed. Additionally, if the default OpenMM compatibility is not enough, ParmEd's automated loaders will be called to parse the file into something that OpenMM can understand. When it comes to write results, OpenMM is compatible with several types of trajectory and checkpoint formats. OMMProtocol integrates those found in ParmEd and MDTraj and adds two custom ones. To handle this diversity, the possible output files are categorized in trajectory, restart and log reports. For more details, please see ESI tables 1 and 2. All the generated files are named after the protocol name and originating stage for easy identification during the analysis.

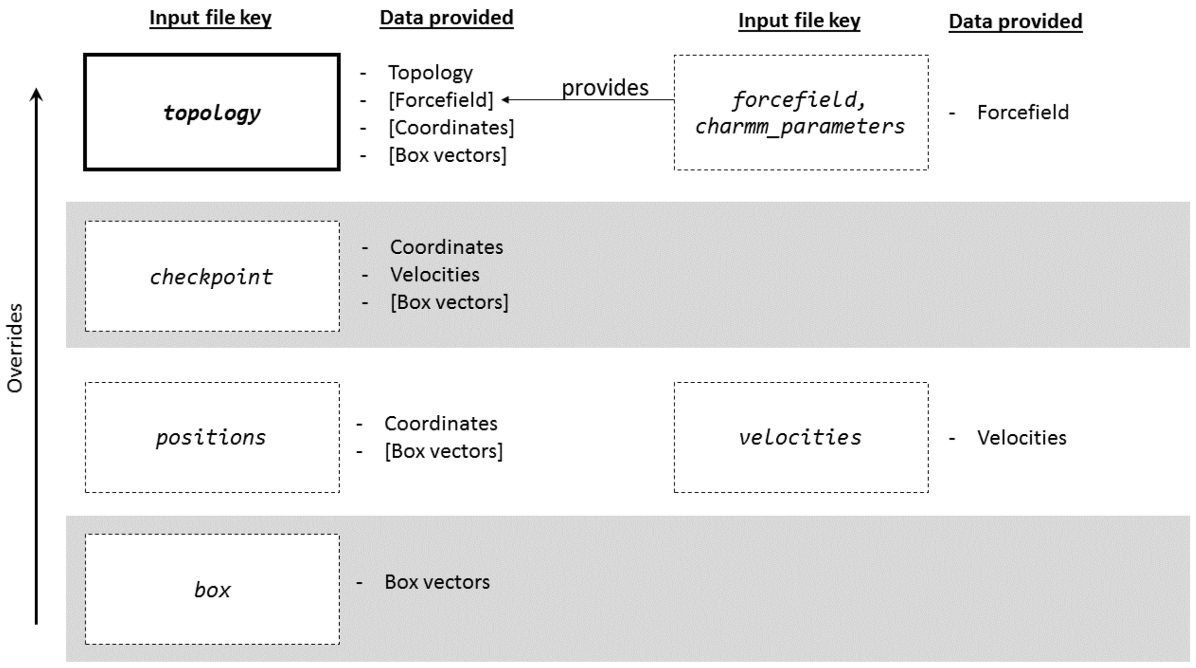


Figure 2: Order of precedence of input files. In OMMProtocol, three main data are required to run a simulation: topology, coordinates and forcefield parameters. The main file is specified with the topology key. In addition to the topology information, this file can also provide coordinates and box vectors (like in PDB files) or forcefield parameters (like Amber's PRMTOP). If one of the three main data is missing, it must be provided in its separate, corresponding key. This strategy can also be used to override information provided by files specified in lower categories of precedence. For example, if a PDB file is chosen as the topology source, its original coordinates can be overridden with a different PDB file in the positions key.

Constraints and restraints. Custom forces are one of the most popular OpenMM features. However, applying a given force to a subset of atoms is not very user-friendly and one must provide the specific atom indices as enumerated in the topology file. Users would expect to use simple keywords like "protein", "solvent" or "backbone" to select parts of the system. OMMProtocol uses MDTraj's domain-specific language (DSL) to apply constraints, positional restraints or distance-based restraints to a subset of atoms.

Results

OMMProtocol can be easily installed through *conda* packages and self-contained wizards in Windows, MacOS and Linux, which will provide the needed commands. To launch a simulation, the user just needs to run *ommprotocol simulation.yaml* from the terminal, where *simulation.yaml* is the input file. Additionally, basic trajectory analysis can be performed afterwards with the accompanying *ommanalyze* executable, if desired. More details can be found in the accompanying Supporting Information.

Conclusions

With OMMProtocol, both beginners and experts can start to benefit from the stellar OpenMM performance right away, without worrying about programming skills or implementation details. As a result, unattended simulations can be set to reach longer timescales in less time, both for the computation itself and the input preparation. Nowadays it is routinely used in our research group, where several publications have been directly benefitted.[14,15] It is LGPL-licensed and freely available in GitHub (https://github.com/insilichem/ommprotocol).

ASSOCIATED CONTENT

Supporting Information. The following files are available free of charge.

- ommprotocol-supporting (PDF). Supplementary details on OMMProtocol implementation and usage.
- ommprotocol-technical-documentation (PDF). Installation instructions and Python API documentation.
- ommprotocol-src (ZIP). Snapshot of the source code at the moment of submission.

AUTHOR INFORMATION

Corresponding Author

* jaime.rodriguezguerra@uab.cat, jeandidier.marechal@uab.cat.

Author Contributions

The manuscript was written through contributions of all authors. All authors have given approval to the final version of the manuscript. The software was written by J. R-G. P. and tested by all authors.

Funding Sources

We are thankful for the support given by the Spanish MINECO (project CTQ2017-87889-P) and the Generalitat de Catalunya (project 2017-SGR-1323). Support of COST Action CM1306 is kindly acknowledged. Research grants to JRGP and LAC (Generalitat de Catalunya & European Social Fund, 2017FI_B2_00168 & 2017FI_B2_00064), and LVC (UPV-EHU, PIF17/22) are also acknowledged.

Conflict of Interest: none declared.

ACKNOWLEDGMENTS

We want to thank Elisabeth Ortega-Carrasco and Sergi Pérez Labernia for helping introduce GPU computing in our research group. Jason Swails, Peter Eastman and John Chodera are also acknowledged for their help and assistance on the questions arisen during the development of OMMProtocol.

ABBREVIATIONS

MM, molecular mechanics; MD, molecular dynamics; GPGPU, general purpose graphic processing unit; GPU, graphic processing unit; DSL, domain-specific language; LGPL, lesser general public license; ESI, electronic supporting information

REFERENCES

(1) Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. An overview of the Amber biomolecular simulation package. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2013**, *3*, 198–210.

(2) Brooks, B. R.; Brooks, C. L.; Mackerell, A. D.; Nilsson, L.; Petrella, R. J.; Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch, S.; Caflisch, A.; Caves, L.; Cui, Q.; Dinner, A. R.; Feig, M.; Fischer, S.; Gao, J.; Hodoscek, M.; Im, W.; Kuczera, K.; Lazaridis, T.; Ma, J.; Ovchinnikov, V.; Paci, E.; Pastor, R. W.; Post, C. B.; Pu, J. Z.; Schaefer, M.; Tidor, B.; Venable, R. M.; Woodcock, H. L.; Wu, X.; Yang, W.; York, D. M.; Karplus, M. CHARMM: the biomolecular simulation program. *J. Comput. Chem.* **2009**, *30*, 1545–1614.

(3) Abraham, M. J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J. C.; Hess, B.; Lindahl, E. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* **2015**, *1–2*, 19–25.

(4) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kalé, L.; Schulten, K. *Journal of Computational Chemistry*. 2005, 1781–1802.

(5) Pappu, R. V.; Hart, R. K.; Ponder, J. W. Tinker: a package for molecular dynamics simulation. *J. Phys. Chem. B* **1988**, *102*, 9725–9742.

(6) Eastman, P.; Swails, J.; Chodera, J. D.; McGibbon, R. T.; Zhao, Y.; Beauchamp, K. A.; Wang, L.-P.; Simmonett, A. C.; Harrigan, M. P.; Stern, C. D.; Wiewiora, R. P.; Brooks, B. R.; Pande, V. S. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Comput. Biol.* **2017**, *13*, e1005659.

(7) McGibbon, R. T.; Beauchamp, K. A.; Harrigan, M. P.; Klein, C.; Swails, J. M.; Hernández, C. X.; Schwantes, C. R.; Wang, L.-P.; Lane, T. J.; Pande, V. S. MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. *Biophys. J.* **2015**, *109*, 1528–1532.

(8) Swails, J. M. ParmEd http://parmed.github.io (accessed Feb 13, 2018).

(9) Eastman, P. pdbfixer https://github.com/pandegroup/pdbfixer (accessed Feb 13, 2018).

(10) Beauchamp, K.; Bruns, C.; Chodera, J.; Eastman, P.; Friedrichs, M.; Ku, J. P.; Markland, T.; Pande, V.; Radmer, R.; Sherman, M.; Swails, J.; Wang, L.-P. 2. The OpenMM Application Layer: Introduction http://docs.openmm.org/latest/userguide/application.html (accessed Feb 13, 2018).

(11) McGibbon, R. T. OpenMM Builder http://builder.openmm.org (accessed Feb 13, 2018).

(12) Eastman, P. OpenMM Setup https://github.com/pandegroup/openmm-setup (accessed Feb 13, 2018).

(13) McGibbon, R. T. OpenMM CMD https://github.com/rmcgibbo/openmm-cmd (accessed Feb 14, 2018).

(14) Drienovská, I.; Alonso-Cotchico, L.; Vidossich, P.; Lledós, A.; Maréchal, J.-D.; Roelfes, G. Design of an enantioselective artificial metallo-hydratase enzyme containing an unnatural metal-binding amino acid. *Chem. Sci.* **2017**, *8*, 7228–7235.

(15) Villarino, L.; Splan, K.; Reddem, E.; Gutiérrez de Souza, C.; Alonso-Cotchico, L.; Lledós, A.; Maréchal, J.-D.; Thunnissen, A.-M.; Roelfes, G. An Artificial Heme Enzyme for Cyclopropanation Reactions. *Angew. Chemie* **2018**.