SoftwareX 7 (2018) 70–73

Contents lists available at ScienceDirect

SoftwareX

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

Software update

Update 0.2 to "pysimm: A python package for simulation of molecular systems"

Alexander G. Demidov [a], Michael E. Fortunato [a], Coray M. Colina [a,b,*]

[a] Department of Chemistry, University of Florida, Gainesville FL 32611, United States
[b] Department of Materials Science and Engineering and Nuclear Engineering, University of Florida, Gainesville, FL 32611, United States

ARTICLE INFO

Article history:
Received 19 February 2018
Received in revised form 23 February 2018
Accepted 26 February 2018

Keywords:
Amorphous polymers
Molecular dynamics simulations
Monte Carlo simulations

ABSTRACT

An update to the pysimm Python molecular simulation API is presented. A major part of the update is the implementation of a new interface with CASSANDRA — a modern, versatile Monte Carlo molecular simulation program. Several significant improvements in the LAMMPS communication module that allow better and more versatile simulation setup are reported as well. An example of an application implementing iterative CASSANDRA–LAMMPS interaction is illustrated.

Code metadata

Current code version	v0.2
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2018_7
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	python2.7
Compilation requirements, operating environments & dependencies	Linux
If available Link to developer documentation/manual	https://pysimm.org/documentation/
Support email for questions	support@pysimm.org

1. Motivation and significance

The *pysimm* API [1] was developed recently to facilitate research in the field of computational materials chemistry, particularly simulations of amorphous polymeric systems. One general goal of the *pysimm* development was to provide easier access to different molecular simulation tools in a clear and uniform manner. This paper describes an update to the *pysimm* software that expands functionality to execute Monte Carlo (MC) molecular simulations. Additionally, having the functionality of molecular dynamics (MD) simulations, *pysimm* now can combine these methods and perform a variety of hybrid MC/MD simulations. For example, hybrid MC/MD simulations can be implemented as iterative cycles of MC and MD where the two methods work with shared molecular structures. Well known MC simulation software packages such as Cassandra [2], RASPA [3], or GPU-Optimized Monte Carlo (GOMC) [4], to the best of our knowledge, do not provide specific tools to communicate with MD software packages or perform the MD simulations directly. MD specialized packages such as Amber (and Ambertools) [5], GROMACS [6], and DL_POLY [7] often have abilities to perform specific MC-like (random or non-physical) moves during the MD simulations. Developers of the LAMMPS software package [8] recently added the possibility to perform Grand-Canonical Monte Carlo (GCMC) simulations. The update described in this work implements an interface that allows *pysimm* to communicate with the Cassandra molecular simulation software released in 2015 for MC simulations developed in the Maginn group [2]. It can perform simulations in a variety of ensembles (NVT, NPT, μVT), and uses advanced computationally efficient MC techniques. The latest version of the software (1.2) was released in November of 2017, and the *Cassandra* module of *pysimm* (**PyS**imm–**CaS**sandra, **PS–CS**) currently supports this version. The module is included in the distribution of the latest API version that can be cloned from the GitHub or alternatively downloaded from the project web-site (see Code metadata).

2. pysimm.Cassandra module

The interface of the PS–CS module is represented by 3 main classes (Fig. 1). The primary class of the module is called *Cassandra*,

DOI of original article: http://dx.doi.org/10.1016/j.softx.2016.12.002.
* Corresponding author at: Department of Chemistry, University of Florida, Gainesville, FL 32611, USA.
E-mail address: colina@chem.ufl.edu (C.M. Colina).

https://doi.org/10.1016/j.softx.2018.02.006
2352-7110/

which organizes all simulations, and interacts with the user and the Cassandra binary. The module assumes that the absolute path to the Cassandra binary (either serial or parallel versions) is set through the environment variable CASSANDRA_EXEC. Similar to the *pysimm.lmps* module, the *Cassandra* class has a task queue that is a list of different MC simulation objects. The simulation objects are added to the queue by the *add_simulation* template method of the *Cassandra* class. For example, the *Cassandra.add_gcmc* method is called if a GCMC simulation is required. Currently the module supports simulations in the canonical (NVT), grand-canonical (μVT) and isobaric–isothermal (NPT) molecular ensembles.

The simulation object will set up the correct Cassandra run by creating the simulation instructions file (INP) using either user provided or default simulation settings. Additionally, the molecular specific files (MCF and DAT) required by Cassandra for each user-defined molecular species are created. The functionality of parsing and writing the molecular specific files is delegated to the *cassandra.McSystem* class. This class represents the set of species that will be inserted/modified during the MC simulations (each species is a separate *pysimm.System* object). Note, that in the current implementation the PS–CS module does not support non-rigid MC simulations, which require CASSANDRA-specific molecule fragment files. However, if these fragment files are available, they can be provided to pysimm to perform non-rigid MC simulations.

The PS–CS module uses the same syntax found in the CASSANDRA input file. For example, the simulation temperature in the CASSANDRA input file is preceded by the tag "#Temperature Info". Correspondingly, the simulation temperature is set by the "Temperature_Info" key of the simulation settings dictionary in the Cassandra.GCMC object.

3. Improvements and additions to the LAMMPS module

Several improvements have been introduced to the pysimm–LAMMPS interface to increase the coverage of LAMMPS functionality and enhance the modularity of the object oriented representation of LAMMPS input. A new class was added to the pysimm–LAMMPS module: it defines groups of particles within the context of LAMMPS to allow users to apply integration to only a selection of particles. This was useful specifically when developing a hybrid MC/MD application during which a separate fix was applied to the frame and gas particles, respectively. This is a requirement for NPT simulations in LAMMPS that include rigid and non-rigid molecules.

Another class was introduced to define initialization settings for LAMMPS simulations. As an example of this, force field functional forms can be explicitly defined in this class. Alternatively, the name of a supported force field can be supplied, which implicitly defines a set of functional forms defined within the LAMMPS module itself. This is in an effort to ultimately remove LAMMPS specific syntax from the system module, and use force field names as a common language between the modules. If no explicit styles are defined, they are inferred based on the force field attribute of the System object passed when constructing the Simulation object.

4. Combined MC/MD application

The functionality of the PS–CS module and improvements in the *lmps* module enabled the development of a combined MC/MD application workflow. The application iteratively runs two consecutive simulations: first, GCMC simulations in Cassandra; and second, NPT molecular dynamics simulations in LAMMPS. The setup of all simulations and all data transfer is carried out by pysimm. To begin, the GCMC simulation will receive a matrix molecular system and the list of molecules to be inserted. After the GCMC simulation is finished, the MD simulation will run starting with the final configuration taken from GCMC. After MD is finished, the updated configuration of the matrix is passed to GCMC on the next iteration. This application is a single Python script that takes all necessary settings for both the MC and MD simulations as well as the description of the initial molecular system. The MC/MD application is included in the distribution of the new pysimm version.

The application can be used to simulate gas adsorption within a flexible matrix. Gas adsorption into flexible matrices is of crucial interest in current and future research [9]. Recently, adsorption simulations with flexible matrices have proven to be an important addition to earlier investigations, as they can show an increase of the matrix size (swelling) or evolution of the pore size distribution which results in changes to the adsorbate uptake and matrix selectivity properties [10,11]. Despite the interest in flexible frameworks, freely available molecular simulation software packages that allow the execution of combined MC/MD simulations are scarce at best.

5. Illustrative example: Swelling in a flexible framework

As an example of the MC/MD application, consider the adsorption of methane (CH_4) to a metal–organic framework (IRMOF-14) matrix. For CH_4 molecules, the TraPPE UA force field model was used [12]. The initial MOF structure (XYZ file) was obtained from the repository of the BTW-FF [13] project. The atoms in the structure were automatically assigned with parameters of the Dreiding force field [14] as it is shown in the *example 10* of the distribution repository. A total of five iterative steps of MC/MD were performed where each iteration had $3 \cdot 10^5$ steps of μVT Monte Carlo simulations followed by 0.1 ns of NPT molecular dynamic simulations. The simulation temperature was set to 300 K, and the pressure to 30 MPa, which is equivalent to a chemical potential of -22.5 kJ/mol using the ideal gas approximation.

Additionally, for comparison with hybrid MC/MD simulations, CH_4 adsorption was also performed using only GCMC. Fig. 2 shows the side-view orthographic projection of the simulated molecular system before and after the MC/MD (right) and only MC (left) were performed. From these figures, it can be observed that after the hybrid MC/MD simulations, the geometry of the MOF framework changed.

Though the difference in geometrical structures of the MOF after MC/MD and MC might look small, the MC/MD simulation shows a noticeably higher amount of adsorbed CH_4 molecules, as depicted in Fig. 3. In this figure the number of adsorbate molecules in MC (blue triangles) and MC/MD (red circles) simulations are presented. The average of the number of inserted molecules over the last 50% of each simulation are shown as bold horizontal lines. On average, MC/MD simulations results showed an increase of adsorbed methane molecules ($\sim$10%) compared to the MC simulations.

6. Conclusions

The update of the *pysimm* API provides new functionalities that include communication with the CASSANDRA software package for Monte Carlo molecular simulations and increased coverage of LAMMPS features. The update will be helpful for studies of adsorption in a variety of nanoporous materials, including polymers and organic molecules with intrinsic microporosity, hypercrosslinked polymers, porous aromatic frameworks, and many others. In general, *pysimm* can be used for any computational research that will benefit from fast and automated interaction between MC and MD molecular simulations. This update will be useful for educational purposes: as an illustration of common principles and differences of Monte Carlo and molecular dynamics simulations. The capabilities of the new *pysimm* module were illustrated in an application