ARTICLE

NeuDATool: an Open Source Neutron Data Analysis Tools, Supporting GPU Hardware Acceleration, and across-Computer Cluster Nodes Parallel

Chang-li Ma[a,b], He Cheng[a,b*], Tai-sen Zuo[a,b], Gui-sheng Jiao[a,b], Ze-hua Han[a,b], Hong Qin[a,b]

a. Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China
b. Spallation Neutron Source Science Center, Dongguan 523803, China

(Dated: Received on May 26, 2020; Accepted on June 21, 2020)

Empirical potential structure refinement is a neutron scattering data analysis algorithm and a software package. It was developed by the disordered materials group in the British spallation neutron source (ISIS) in 1980s, and aims to construct the most-probable atomic structures of disordered materials in the field of chemical physics. It has been extensively used during the past decades, and has generated reliable results. However, it implements a shared-memory architecture with open multi-processing (OpenMP). With the extensive construction of supercomputer clusters and the widespread use of graphics processing unit (GPU) acceleration technology, it is now possible to rebuild the EPSR with these techniques in the effort to improve its calculation speed. In this study, an open source framework NeuDATool is proposed. It is programmed in the object-oriented language C++, can be paralleled across nodes within a computer cluster, and supports GPU acceleration. The performance of NeuDATool has been tested with water and amorphous silica neutron scattering data. The test shows that the software can reconstruct the correct microstructure of the samples, and the calculation speed with GPU acceleration can increase by more than 400 times, compared with CPU serial algorithm at a simulation box that has about 100 thousand atoms. NeuDATool provides another choice to implement simulation in the (neutron) diffraction community, especially for experts who are familiar with C++ programming and want to define specific algorithms for their analysis.

Key words: Neutron diffraction, Neutron scattering, Empirical potential structure refinement, Graphics processing unit, C++

I. INTRODUCTION

Neutron total scattering for disordered materials is a powerful tool to study the most probable atomic structures of an amorphous system in the field of chemical physics. Since the introduction of the first total scattering spectrometer (TSS) [1], which was developed at HELIOS in the 1970s, numerous important scientific problems have been solved, including the elucidation of the structures of water at high- and low-densities, and the observation of the heterogeneities in mixed alcohol-aqueous solutions.

The success of neutron total scattering for disordered materials is based on isotope labeling techniques. The scattering profile $F(Q)$ is a weighted summation of the Fourier transforms of all the pair correlation functions (PDF),

$$\begin{aligned} F(Q) &= \sum_{\alpha=1}^{N} c_\alpha {b_\alpha}^2 + \sum_{\alpha=1,\beta\geq\alpha} (2-\delta_{\alpha\beta}) c_\alpha c_\beta b_\alpha b_\beta \\ &\cdot \left[4\pi\rho\int_0^\infty r^2(g_{\alpha\beta}(r)-1)\frac{\sin(Qr)}{Qr}\mathrm{d}r\right] \end{aligned} \quad (1)$$

where Q=$4\pi/\lambda\sin(\theta/2)$ is the scattering vector, λ is the neutron wave-length, θ is the scattering angle, $\sum\limits_\alpha c_\alpha {b_\alpha}^2$ is the self-scattering background, $(2-\delta_{\alpha\beta})c_\alpha c_\beta b_\alpha b_\beta$ are the weighted factors of different partial structural factors, $c_{\alpha/\beta}$ and $b_{\alpha/\beta}$ are the atom ratio and scattering length of atoms with types α/β, respectively. $g_{\alpha\beta}(r)$ is the PDF between atom types α and β. If the different atom type number is M in a sample, there are $M(M+1)/2$ different $g_{\alpha\beta}(r)$ functions in Eq.(1), and we must solve all of them first before the most probable atomic structure is obtained. It is thus difficult to reveal the most probable all-atom structure of disordered material, just according to one X-ray or neutron scattering curve. Fortunately, isotope labeling technique can solve the problem. Because isotoped samples have almost the same atomic structure as their counterpart,

*Author to whom correspondence should be addressed. E-mail: chenghe@ihep.ac.cn, chenghe@iccas.ac.cn each isotoped sample can generate a different scattering pattern.

All-atom model simulation, such as EPSR, is the common method to solve the matrix of Eq.(1), and reconstruct the atomic structure of the tested samples [2]. Because EPSR can provide a reliable and visualized atomic microstructure, it has been extensively used. EPSR is paralleled in shared memory API for parallel programming named Open Multi-Processing (OpenMP) [3, 4]. It cannot be paralleled across different nodes of a supercomputer cluster. This restricts the calculation speed and the analysis system scale, such as in the case of a macromolecular system. A macromolecule is a molecule with a large number of atoms. To cover the whole scattering vectors that characterize the configuration states of macromolecules in a typical total scattering instrument, the size of the simulation box composed of more than half a million atoms should be larger than 10 nm. Currently EPSR cannot be run in such a large system. Therefore, parallel calculations are necessary.

In light of this, the object-oriented language C++ and the compute unified device architecture (CUDA) [5, 6] are used to develop a toolkit NeuDATool. In NeuDATool, experts can define simulation boxes, atoms, molecules, and movement models easily using the class inheritance mechanism. The mechanism allows experts to define a class based on an existed class, and is easier to create new classes, too [7]. Graphics processing unit (GPU) hardware acceleration [8, 9] is supported by CUDA C, and this allows the program to take advantage of GPU computing servers. In addition, with the distributed memory architecture API message passing interface (MPI) mpich2 [3, 10], NeuDATool can be paralleled across nodes of a supercomputer cluster. It promotes the calculation speed.

II. NEUDATOOL ALGORITHMIC FLOW

At present, NeuDATool implements the basic analysis algorithm of EPSR [2, 11]. The principle of EPSR algorithm is briefly introduced in Appendix A. Similar to EPSR, the force field in NeuDATool is divided into a reference potential (RP) and an empirical potential (EMP): the reference potential takes the form of a combination of simple Lennard-Jones and electrostatic potential; while the EMP, which reflecting the experimental data, uses a combination of a list of Poisson functions. NeuDATool's algorithm flow is shown below, and the algorithm flow chart is in Appendix A.

I. Define atoms, molecules, and the simulation box with inheritance in C++. Three basic C++ classes have been designed to help users to define their special subclasses or objects:

Atom: It is used to define an atom type in a simulation. Some basic atomic properties, such as the atom name, element name, isotope name, coordinate position, neutron scattering length, are defined. In the class, the coordinate position of an atom is defined using the Hep3Vector class of the class library for high energy physics (CLHEP) [12, 13], because the Hep3Vector has abundant functions to perform transition, rotation, distance, and angle calculations. Users need to initialize atoms in molecular objects rather than defining new atom classes.

Molecule: It is used to define molecules, intramolecular potentials and their movements, such as translation, rotation, *etc.* Users need to define new molecular classes through inheritance, and can try special inter- and intramolecular movements for their analysis. This class makes the program flexible and user-friendly.

SimBox: It is used for generating a simulation box. Users need to define a subclass for their simulation. In the subclass, users only need to edit an initial function with defined molecules. Users can use any suitable algorithm to place the molecules in the model box for generating an initial conformation.

In order to facilitate users who are not familiar with C++ programming, the software provides functions from Jmol [14] and GROningen machine for chemical simulations (GROMACS) [15] to define molecules. Users need to provide the corresponding conformation text files, and the program will use these files to generate molecular C++ classes. The details are shown in Appendix B.

II. Recompile and run program.

III. MC: The program performs MC simulations with reference potentials until equilibrium is reached. The program moves molecules or atoms in sequence or randomly. The potential energy variation of the simulation box ($\Delta U{=}U_{\text{after}}{-}U_{\text{before}}$) is used as the movement acceptance criterion. If $\Delta U< 0$, the movement is accepted. If $\Delta U{>}0$, the movement is accepted with a probability $e^{-\Delta U/kT}$.

IV. Calculate $g(r)$, $F(Q)$, and EMP: When MC reaches equilibrium, the program calculates the difference of the neutron structural factor $\Delta F(Q)$ between the simulation $F_{\text{sim}}(Q)$ and experiment $F_{\text{exp}}(Q)$. EMP is calculated by applying the Fourier transform to $\Delta F(Q)$. The program adds the EMP and RP together as the updated potential to perform the EPMC simulation.

V. The empirical potential Monte Carlo (EPMC) simulation: When the simulation reaches equilibrium, the program calculates the EMP again, and adds it to the previous potential, then performs the simulation with the updated potential. When the maximum value of the $\Delta F(Q)$ reaches a low value which is set by the user, the program starts to accumulate simulation data. Users can also use other variables as the judgment conditions to start accumulating data, such as the maximum value of EMP, the χ^2 between $F_{\text{sim}}(Q)$ and $F_{\text{exp}}(Q)$, and so on.

VI. Accumulate $g(r)$ and $F(Q)$: The program continues to perform EPMC and accumulate simulation data to improve statistics. It will not be stopped until smoothed $g(r)$ and $F(Q)$ curves are obtained. The program outputs a coordinate file which includes all the atoms in a text format, as what is used in the GROMACS (with the suffix of .gro) [15], or in the large-scale atomic/molecular massively parallel simulator (LAMMPS) (with the suffix of .xyz) [16]. Accordingly, this file can be input to GROMACS or LAMMPS to calculate enthalpy, entropy, *etc.*, and also can be visualized by Visual molecular dynamics (VMD) [17, 18]. Experts can define, calculate, and output any interesting variables which are related directly with the atomic structure of the sample.

III. ACCELERATION METHOD

The computation consumptions of $\Delta N(r)$ and $N(r)$ are in approximate proportion to the atom number in the simulation box and its second order, respectively. In general, the amount of atoms in a simulation box is larger than ten thousand. $\Delta N(r)$ needs to be recalculated after every MC simulation step for ΔU calculation, while $N(r)$ needs to be recalculated after every MC/EPMC equilibrium for $g(r)$ and $F(Q)$ update. Thus, these algorithms represent the highest consumption of the program's calculation capacity. A graphics processing unit (GPU) has thousands of parallel threads, so it is a suitable candidate to accelerate the calculation of $\Delta N(r)$ and $N(r)$.

For the implementation of across nodes in a parallel configuration, mpich2 is used in the program [19]. Mpich2 is based on the MPI [10] standard and supports point-to-point and collective data communication among different nodes. Thus, it is more efficient in this program. For the implementation of shared memory multithread parallel configuration within a computer or server node, Open Multi-Processing (OpenMP) is used [4]. The OpenMP syntax supports the setting of a thread number dynamically, and a thread number cannot be known in advance in most cases. Thus, it is very convenient in programming. NeuDATool uses OpenMP and MPI API to distribute the calculations of $\Delta N(r)/N(r)$ to different GPU cards, which belong to different nodes in a computer cluster.

IV. PERFORMANCE TEST

Two kinds of neutron scattering data from literatures are used to test the correctness and computational speed of NeuDATool: (i) liquid water samples, *i.e.*, hydrogened, deuterated, and half deuterated water [20], (ii) amorphous SiO_2 sample [21].

A. Correctness test

To simulate water, we define a C++ class to describe its molecule H_2O by inheriting from the molecule basic class, which is predefined in the program. The molecule structure is maintained with harmonic oscillator potentials between every two atoms. Three different random movements *i.e.*, H_2O translation, H_2O rotation, and atoms (H or O) translation, are implemented in the simulation. The EMP (UEMP) of the atom type pairs of the sample are calculated [22],

$$U_j^{\mathrm{EMP}}(r)=\frac{1}{4\pi\rho}\sum_{i=1,M}w_{ji}^{-1}\int_0^{\infty}\Delta S_i(Q)\mathrm{e}^{-iQr}\mathrm{d}Q \quad (2)$$

where $U_j^{\mathrm{EMP}}(r)$ is the EMP of different atom type pairs, *i.e.*, O−O, O−H, and H−H in water; $\Delta S_i(Q)$ is the different neutron structure factor between experiment and simulation; w_{ji}^{-1} is the inverse of weighted factor matrix in Eq.(1). In water samples, the number of neutron scattering profiles and atom type pairs are the same, so we employ matrix Invert() function which is provided by CLHEP [12, 13] to calculate the inverse matrix.

In the simulation of SiO_2, we define two classes to describe Si and O as single atom molecules. Note that there is only one neutron scattering profile but three different atom type pairs (Si−Si, Si−O, and O−O, respectively). As what is done in EPSR [23], we enlarge the (1×3) weighted factor matrix, then use a Monte-Carlo method to calculate its pseudo-inverse matrix w'^{-1}. FIG. 1 shows the $\Delta F(Q)$ fitting results of water samples (left) and its corresponding EMP potential (right), respectively.

FIG. 2 demonstrates the NeuDATool fitting results of water and SiO_2 samples. They are consistent with the experiments. It confirms that the distributions of small molecules represent the most probable all-atom positions in the disordered material.

To verify the reliability of the simulation, we compare the PDF distribution of water from NeuDATool with other studies (FIG. 3). Liquid water has short-range-ordered tetrahedral structure. On average, 3.5 water molecules form hydrogen bonding with a center one. The NeuDATool fitting results are consistent with the report in Refs.[23, 24].

All of those prove NeuDAToool can reconstruct the atomic structures of experimental samples correctly.

B. Computational speed test

A small computer cluster is used to test the speed performance of different parallel methods. The cluster uses CentOS 7.3 as its operating system and has two nodes. Each node has two Intel Xeon Scalable Gold 6126 CPUs (2 SkylakeCSP architectures, 12 cores, 24 threads, 2.6 GHz, Turbo 3.7 GHz, and a 19.25 MB L3 Intel smart cache), two Nvidia Tesla V100 calculation GPU cards, and 128 GB double data rate (DDR4) error correcting code (ECC) registered shared memory. The two nodes are connected with an InfiniBand (IB) connector (data transmission speed can reach 56 Gb/s).

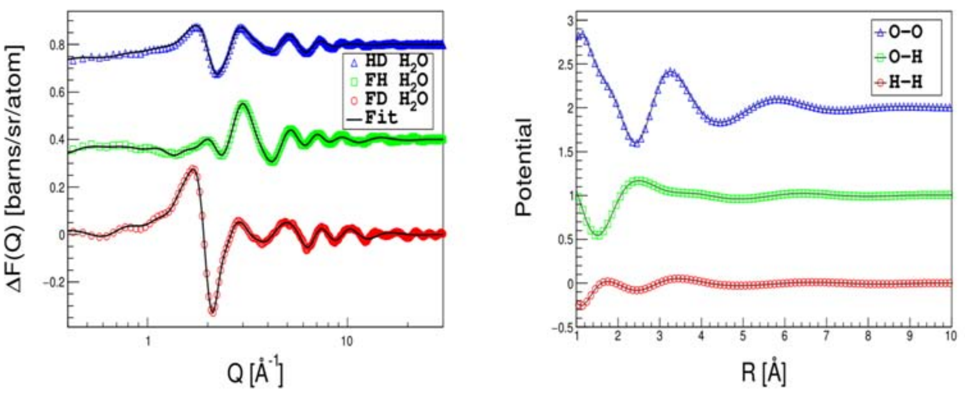

FIG. 1 The $\Delta F(Q)$ fitting results of water samples (HD, FH and FD H_2O mean half deuterated, fully hydrogenated and fully deuterated water, respectively.) with Poisson distribution (left) and the corresponding empirical potentials (right). The $\Delta F(Q)$ of FH and HD are shifted by 0.4 and 0.8, respectively. The EMP of O−H and O−O are shifted by 1.0 and 2.0, respectively.

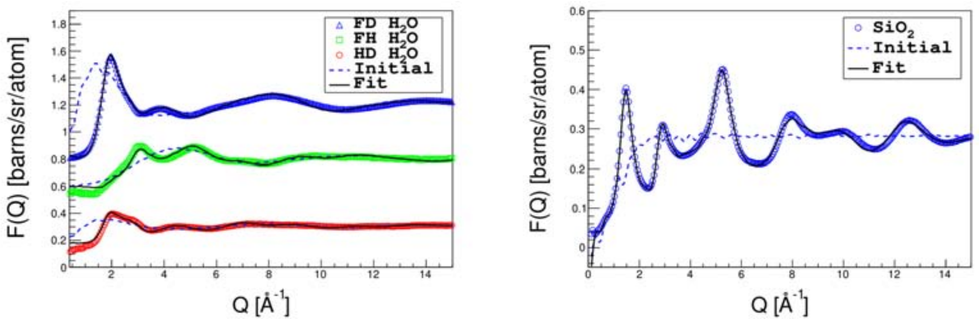

FIG. 2 Neutron scattering profiles and NeuDATool fitting results. Left: H_2O samples. Right: amorphous SiO_2 sample. The points denote the experiment data, while the solid lines are the NeuDATool simulations. The dashed lines denote the random initial simulation boxes.

A speed comparison of these acceleration methods is shown in FIG. 4, and a more detailed quantitative comparison is listed in Table I. As shown, GPU acceleration greatly improves the calculation speed. Especially for systems with more than 200,000 atoms, GPU computing speed is more than 400 times faster than serial CPU algorithm. Most importantly, with GPU acceleration, the program can simulate a system comprising>1 million atoms. This is an essential improvement because it allows the program to simulate systems larger than 200 Å, so that it may analyze macromolecules sample in the future.

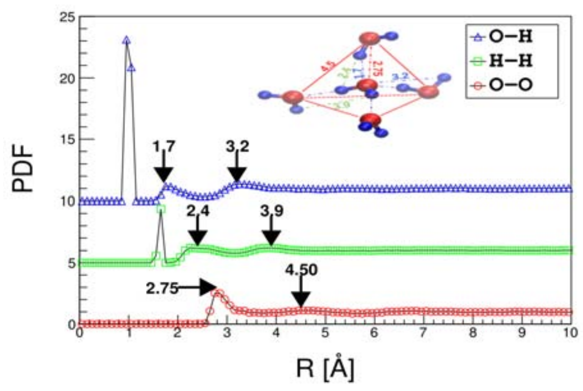

FIG. 3 PDF distributions of O−H, H−H, and O−O from NeuDATool simulation for water, and the inset is a cartoon of the liquid water structure [23−25].

V. CONCLUSION

The neutron scattering data analysis software NeuDATool is programmed with the object-oriented language C++. It makes the program flexible. Potential functions and the corresponding parameters of the atomic force field can be modified or added by editing the C++ head and source files. In addition, C++ is an easy-to-read, high-level computer language, so experts can try new algorithms and program flows to improve their analysis, and any important variables can be calculated. Nevertheless, the software also provides some functional modules so that users who are not familiar with C++ programming can define molecules, simulate boxes and set force field parameters through text files.

With the MPI API and CUDA architecture, cross-

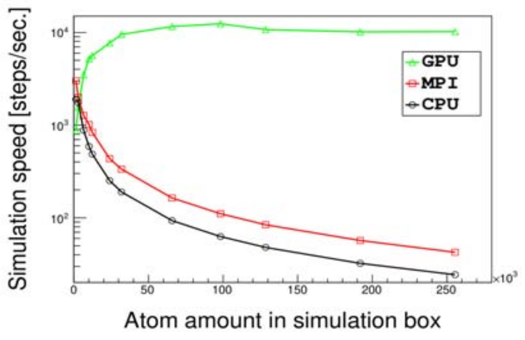

FIG. 4 Calculation speed comparison among serial algorithm, MPI parallel and GPU acceleration algorithms.

TABLE I Simulation speeds with GPU acceleration, MPI parallel and CPU serial algorithms in unit of steps/sec.

Atomic number	GPU	MPI	CPU
3×10^3	1562	2000	1754
3×10^4	9507	334.9	189.8
1×10^5	12412	110.5	62.67
2.5×10^5	10189	42.57	24.30
1.2×10^6	3917		
3×10^6	1963		
1×10^7	757.0		
3×10^7	340.3		

nodes parallel calculations within a computer cluster and GPU hardware acceleration are supported. Specifically, with GPU acceleration, the calculation speed is improved considerably, so the program has the capability to analyze disordered macromolecular samples and nanoparticles in the future.

Although the program is flexible for users and has a powerful calculation capacity, it was tested with a very limited number of samples. Accordingly, in its current form, it is not a fully functional software package. The authors aspire to release it as an open-source toolkit framework for interested scientists. In this sense, users will be able to contribute numerous new molecular classes, algorithms, and analysis routines to make the program more powerful in the future.

VI. ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China (No.2017YFA-0403703), and the National Natural Science Foundation of China (No.U1830205, No.21674020). The authors thank the staff of the ISIS Disordered Materials Group for the explanations of the principles of EPSR and for providing experimental neutron diffraction data for testing of the program.

Appendix A. NeuDATool simulation principle

NeuDATool implements the analysis algorithm of EPSR. EPSR is essentially a Monte-Carlo simulation method and evolved from Reverse Monte Carlo (RMC). Its difference from metropolis MC is based on the fact that its atomic potentials include neutron scattering data information [2, 11]. In NeuDATool, the atomic potential used in the MC simulation is divided into two categories *i.e.*, "reference potential (RP)" and "empirical potential (EMP)". RP uses Lennard-Jones potential and electrostatic potential, and the potential parameters can be obtained from all-atom MD force fields; RP is used to assign molecules with correct shapes and realize other constraints whose correctness has been approved in previous studies. By contrast, EMP is used to reflect the differences of neutron structural factors ($\Delta F(Q)$) between experiments and MC simulations. To be exact, EMP is the reverse Fourier transform of $\Delta F(Q)$ as the perturbation to the RP, and it is used to guide the simulation approach to scattering measurements. In the present form of NeuDATool, EMP is expressed as a list of Poisson distributions in real space and their corresponding Fourier transforms in Q space as in EPSR. In short, EMP as a feedback guides the simulation to get closer to the experimental data. Only RP is used in the MC simulation at the beginning, and the potential changes of the system (ΔU) are used as the selection criteria of molecules or atoms random movements. When the simulation reaches equilibrium, EMP is then introduced to fit $\Delta F(Q)$ and is added to RP to continue the simulation. When the MC simulation with updated potential reaches equilibrium again, NeuDATool calculates EMP and updates the simulation potential once more. This process is repeated until the maximum value of EMP is close to zero, *i.e.*, $\Delta F(Q)$ becomes very small.

Appendix B. How to create molecules from conformation files generated by Jmol and Gromacs

In order to facilitate users who are not familiar with C++ programming, we have designed a module that can define molecular C++ classes through conformation files generated by Jmol and Gromacs software. The molecular conformation file (suffixed with .jmol) generated by Jmol contains the three-dimensional coordinates and element names of all atoms in the molecule. The user only needs to edit the conformation file, *i.e.*, adds the atom names used in NeuDATool after the element names of the atoms, then calls function Initialize ByJmolFile ("file.jmol"), the software thus defines the molecular class automatically. Similar to this, the molecular conformation file (suffixed with .gro) generated by the molecular dynamics simulation software Gromacs also contains information such as the three-dimensional coordinates and the atom names of the atoms belonging to the molecule. The user only needs to call the function Initialize ByGroFile ("file.gro") function, and the software can automatically define the molecular classes needed in the simulation.

In terms of force field settings, users can also make it by writing text files. The format of the force field file is as follows: the first column of the first line is the number of atoms contained in the force field file, the second to the forth column are the force field parameters of the atoms, with the first column being the name of the atom, and the second column to the fourth column are the epsilon (kJ/mol), sigma (Å) of Lennard-Jones potential and electric charge (e), respectively. The user can call Set AtomType Text("potential.txt") function to set the force field parameters.

[1] R. Sinclair, D. Johnson, J. Dore, J. Clarke, and A. Wright, Nucl. Instru. Method. **117**, 445 (1974).

[2] R. L. Mcgreevy and L. Pusztai, Mol. Simulat. **1**, 359 (1988).

[3] E. R. Hesham and A. E. B. Mostafa, *Advanced Computer Architecture and Parallel Processing*, Hoboken, New Jersey: John Wiley & Sons, Inc., (2005).

[4] *The Openmp api Specification for Parallel Programming,* https://www.openmp.org/, accessed on 30 April 2020.

[5] *Cuda c Best Practices Guide*, https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html, accessed 30 April 2020.

[6] *Cuda c Programming Guide*, https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html, accessed on 30 April 2020.

[7] *C++ Inheritance*, https://www.tutorialspoint.com/cplusplus/cpp_inheritance.htm, accessed on 30 April 2020.

[8] Z. Liping, G. Zhiqiang, C. Ruiwen, F. Yue, Z. Gaofeng, and X. Benzhu, Comput. Graph. **80**, 29 (2019).

[9] M. G. Awan, T. Eslami, and F. Saeed, Comput. Biol. Med. **101**, 163 (2018).

[10] *Open mpi: Open Source High Performance Computing*, https://www.open-mpi.org/, accessed on 30 April 2020.

[11] A. K. Soper, Chem. Phys. **202**, 295 (1996).

[12] *CLHEP-a Class Library for High Energy Physics*, http://proj-clhep.web.cern.ch/proj-clhep/, accessed on 30 April 2020.

[13] L. Lonnblad, Comput. Phys. Commun. **84**, 307 (1994).

[14] *Jmol: an Open-Source Java Viewer for Chemical Structures in 3D*, http://jmol.sourceforge.net/, accessed on 30 April 2020.

[15] *Gromacs*, http://www.gromacs.org/, accessed on 30 April 2020.

[16] *Lammps Molecular Dynamics Simulator*, http://lammps.sandia.gov, accessed on 30 April 2020.

[17] *Vmd-Visual Molecular Dynamics*, http://www.ks.uiuc.edu/Research/vmd/, accessed on 30 April 2020.

[18] W. Humphrey, A. Dalke, and K. Schulten, J. Mol. Graph. Model. **14**, 33 (1996).

[19] *Mpich is a High Performance and Widely Portable Implementation of the Message Passing Interface (mpi) Standard*, http://www.mpich.org/, accessed on 30 April 2020.

[20] *Empirical Potential Structure Refinement*, https://www.isis.stfc.ac.uk/Pages/Empirical-Potential-Structure-Refinement.aspx, accessed 30 April 2020.

[21] *Gudrun-Routines for Reducing Total Scattering Data*, https://www. isis.stfc.ac.uk/Pages/Gudrun.aspx, accessed on 30 April 2020.

[22] *Empirical Potential Structure Refinement, a User's Guide*, version 25, (2017).

[23] I. M. Svishchev and P. G. Kusalik, J. Chem. Phys. **99**, 247 (1993).

[24] F. B. A. K. Soper and M. A. Ricci, J. Chem. Phys. **106**, 247 (1997).

[25] T. Head-Gordon and M. E. Johnson, Proc. Natl. Acad. Sci. USA **103**, 7973 (2006).