

# EPSR++: An Open Source Empirical Potential Structure Refinement Neutron Data Analysis Framework Supporting Parallel Across Computer Cluster Nodes and GPU Hardware Acceleration

Changli Ma<sup>a,b</sup>, He Cheng<sup>a,b</sup>, Taisen Zuo<sup>a,b</sup>, Guisheng Jiao<sup>a,b</sup>, Zehua Han<sup>a,b,c</sup>

<sup>a</sup>*Institute of High Energy Physics, Chinese Academy of Sciences (CAS) Beijing 100049, China*

<sup>b</sup>*Dongguan Neutron Science Center, Dongguan 523803, China*

<sup>c</sup>*University of Chinese Academy of Sciences, Beijing 100049, China*

---

## Abstract

Empirical Potential Structure Refinement (EPSR) is a neutron scattering data analysis algorithm and software package. It was developed in the British Spallation Neutron Source (ISIS) Disordered Materials Group, aims at constructing most-probable all-atom structure of amorphous materials. In the past decades, it has been wildly used and obtained reliable results for scientists. However, it is programmed in Fortran and implement shared memory architecture with OpenMP, so it can only run in parallel within a PC or one node of a computer cluster. As a result, EPSR is good at simulating systems smaller than 50 thousand atoms with small molecules, but it is difficult to be used in a macromolecular or nano-particle system and it is very hard to define special molecules by users for their analysis. With the extensive construction of supercomputer clusters and the widespread use of GPU acceleration technology, it is necessary to update EPSR with distributed memory architecture and GPU acceleration supporting to improve its calculation speed. In this paper, an open source framework EPSR++ is introduced. It can be paralleled across nodes within a computer cluster and supports GPU acceleration. In addition, the framework is programmed in C++ object-oriented language, thereby users can define special simulation box, atoms, molecules and random motion patterns conveniently for their analysis. The framework was successfully tested with H<sub>2</sub>O standard samples, and it reconstructs correct microstructure. Parallel performance tests shows that GPU acceleration has a very significant effect.

**Keywords:** Neutron Diffraction, Neutron scattering, Empirical Potential Structure Refinement, EPSR, Reverse Monte Carlo, RMC, GPU, MPI, OpenMP,

C++

---

## PROGRAM SUMMARY/NEW VERSION PROGRAM SUMMARY

*Program Title:* EPSR++

*Licensing provisions:* GPLv2

*Programming language:* C++

*Nature of problem:* Neutron scattering technology is an irreplaceable method to construct the microstructure of experiment samples at atomic scale.[1][2] Empirical Potential Structure Refinement (EPSR) is a powerful and widely used data analysis algorithm and program.[3][4] Unfortunately, EPSR is programmed in Fortran and supports shared memory parallel within a multi-core CPU with OpenMP API. This feature restricts the calculation speed so that EPSR is suited to simulate a system with less than 50 thousand atoms. In addition, the procedure of oriented programming makes it very difficult to define a special molecule and corresponding random movement pattern by users, which is very important especially in analysis macromolecular samples, because of the unique features of every macromolecule. As these restricts, EPSR is cannot be used to analysis a large system with macromolecules and nanoparticles.

*Solution method:* The framework EPSR++ is in C++ object-oriented language. Almost all basic algorithm, functions and classes are designed in it, so users can define special simulation box, molecules, atoms for an analysis based on the multi inherit mechanism. The framework is based on distributed memory architecture to implement across nodes parallel within a computer cluster, and it supports GPU acceleration as well as shared memory parallel with OpenMP API.

*Additional comments including Restrictions and Unusual features:* EPSR++ can be compiled with “make” command in Linux OS. With editing Makefile, users can select which parallel acceleration method, i.e., MPI, OpenMP or GPU, is used according to their hardware and requirement. It defines the main program to implement EPSR, some important functions to calculate pair correlation function (PDF), neutron structure factor (NSF), etc. The framework defines some base classes to describe atoms, molecules and simulation box, while users can define special classes for their samples using inherit mechanism.

## References

- [1] Lefmann, Kim and Kirkensgaard, Jacob and Arleth, Lise and Lebech, Bente and Strobl, Markus and Source, European Spallation. Neutron Scattering : Theory, Instrumentation and Simulation. (2014)
- [2] Jackson, Aj. Introduction to Small-Angle Neutron Scattering and Neutron Reflectometry. (2008)

- [3] Soper, A K. Empirical potential Monte Carlo simulation of fluid structure. *Chemical Physics* 202 (1996) 295-306
- [4] Soper, A K. Empirical Potential Structure Refinement A Users Guide Version 25 February (2017)

## 1. Introduction

Neutron total scattering for disordered material is a powerful tool to study the most probable all-atom structure in an amorphous system so far. Since the first instrument, Total Scattering Spectrometer (TSS) which was developed at HELIOS in the 1970s, lots of important scientific problems have been solved by this kind of instruments, such as the structure both the high and low density of water and the observation of the heterogeneities in the mixed alcohol aqueous solutions.

The success of neutron total scattering is based on deuteration techniques. the neutron scattering pattern  $S(Q)$  is a weighted summation of Fourier transforms of all pair correlation function (PDF),  $g(r)$ , in a system containing  $M$  different atom types[1]:

$$S(Q) = \sum_{\alpha=1} c_{\alpha} b_{\alpha}^2 + \sum_{\alpha=1, \beta \geq \alpha} (2 - \delta_{\alpha\beta}) c_{\alpha} c_{\beta} b_{\alpha} b_{\beta} \left\{ 4\pi \rho \int_0^{\infty} r^2 (g_{\alpha\beta}(r) - 1) \frac{\sin(Qr)}{Qr} dr \right\} \quad (1)$$

where  $\sum_{\alpha} c_{\alpha} b_{\alpha}^2$  is a flat background, and  $(2 - \delta_{\alpha\beta}) c_{\alpha} c_{\beta} b_{\alpha} b_{\beta}$  is the weighted factor of different partial neutron structure factors.  $c_{\alpha/\beta}$  and  $b_{\alpha/\beta}$  are the molar ratio and neutron scattering length of atom (nuclide) type  $\alpha/\beta$  respectively. Therefore, there are  $M(M + 1)/2$  different  $g_{\alpha\beta}(r)$  in Eq.(1), and we must solve all of them first before the most probable all-atom structure is obtained. Fortunately, neutron scattering can employ deuteration techniques. Deuterate samples have almost the same atomic structure as its hydrogenate one, and each deuterated sample can generate different scattering patterns.

Computer simulation must be involved to solve the matrix of Eq.(1)s, because the number of deuterate samples is still smaller than  $M(M + 1)/2$ . A direct way to get the microstructure of samples is making reverse Fourier transform to  $S(Q)$ , which is named PDF method in tradition.[2][3] The results of the PDF method can be interpreted in terms of pair distribution function directly only in monatomic samples, such as an inert gas liquid. In addition, the PDF method can give a statistical average coordination number for one atom type around another, but can not give the direct visualization of the atomic structure of the amorphous

samples.[4][5] Compared with PDF method, EPSR[4][5] is a MC method and developed from Reverse Monte Carlo (RMC)[6, 7, 8, 9] for neutron data analysis to reconstruct the atomic structure of samples. Because EPSR can give a reliable and visualized atomic microstructure, it is widely used in recent years. But there are some defects in the program.

However, EPSR still has lots of defects, which limits its application. First, it is programmed in Fortran. Fortran can implement very fast calculation speed, but it is a procedure-oriented language. It is very difficult to define special atoms, molecules, special molecular or atomic movement, and it is impossible for users to add some new algorithm for their own. This restricts the flexibility of the software. Second, EPSR is paralleled in shared memory architecture[10] with OpenMP[11]. It cannot be paralleled across different nodes of a supercomputer cluster. This restricts the calculation speed and analysis system scale, such as macromolecular system. A macromolecule generally contains hundreds of atoms. To cover the whole ranges of scattering vectors, which characterize the whole conformation of macromolecules, the simulation box containing more than half million atoms should be larger than 10nm. And EPSR cannot run in such huge system. So parallel calculation is necessary.

Aimed at these shortages, object-oriented language C++ and a distributed memory architecture [10] API mpich2[12] are used to develop an open source framework EPSR++ to implement EPSR algorithm. In the framework, users can define new simulation box, atoms, molecules and movement model easily with the class multiple inheritance mechanism. With mpich2 API, the framework can be paralleled across nodes of a supercomputer cluster. In addition, GPU hardware acceleration is supported with CUDA, [13, 14, 15, 16, 17, 18], and this makes the program can take advantage more and more commonly used in GPU computing servers.

## 2. EPSR Principle and EPSR++ Algorithm Flow

### 2.1. A Brief Description of EPSR Principle

EPSR is essentially a Monte-Carlo energy minimization simulation method, but the difference from Metropolis Monte-Carlo is that the potential used in a simulation is from experiment data rather than from theory. In EPSR, the atomic potential used in MC simulation is divided into two categories *i.e.* “reference potential” (RP) and “empirical potential” (EMP).[4] RP is similar to that used in MD simulation, so the potential form and parameters can be obtained from all-atom MD force field such as OPLS[19, 20], AMBER[21], CHARMM[22, 23] and so

on.[4] In contrast, EMP has no fixed form and it is used to reflect the difference of neutron structure factor between experiment and MC simulation ( $\Delta S(Q)$ ). To be exact, EMP is the reverse Fourier transform of  $\Delta S(Q)$ . At present EPSR, EMP is expressed to a list of Poisson distribution in real space and their corresponding Fourier transforms in Q space. In short, RP is used to keep molecules with reasonable shape, while EMP is used as a feedback parameter to lead the MC simulation consistent progressively with diffraction experiment data. In EPSR simulation, RP alone is used in MC simulation at the beginning, and the potential change of system ( $\Delta U$ ) is used as selection criteria of molecule or atom random movement. When the simulation gets to equilibrium, EMP is introduced to fit  $\Delta S(Q)$  and added to RP to continue the simulation. When the MC simulation with updated potential get to equilibrium again, EPSR calculates EMP and updates simulation potential once more. Repeat in this way, until the EMP close to zero, *i.e.*  $\Delta S(Q)$  is very little.

## 2.2. EPSR++ Algorithm Flow

Algorithm flow of EPSR++ is shown in Fig.1. Details of some processes are described below:

- I. Define simulation box, molecules, atoms, select a parallel method, and re-compile the program. Three basic C++ classes are defined in the framework for users to define special classes or objects:
  - SimBox: It is used for generating a simulation box. It can be initialized with the molar mass of molecules, molecular numbers and density, etc. Users can define a spacial initial function of simulation model box.
  - Molecule: It is used to define molecules, intra-molecule potentials and their movements such as translation, rotation, etc. Users can define arbitrary molecule classes and try any special inter and intra molecule movements for their analysis. This class makes the program very flexible and friendly to uses.
  - Atom: It is used to define an atom type used in a simulation. Some basic properties of an atom such as atom name, element name, isotope name, coordinate position, neutron scattering length, etc are defined. The coordinate position is defined with Hep3Vector class of CLHEP[24] library because of Hep3Vector has abundant functions to perform transition, rotation, distance and angle calculation, etc. Users need to initialize atom objects in molecule objects rather than define any new atom classes.

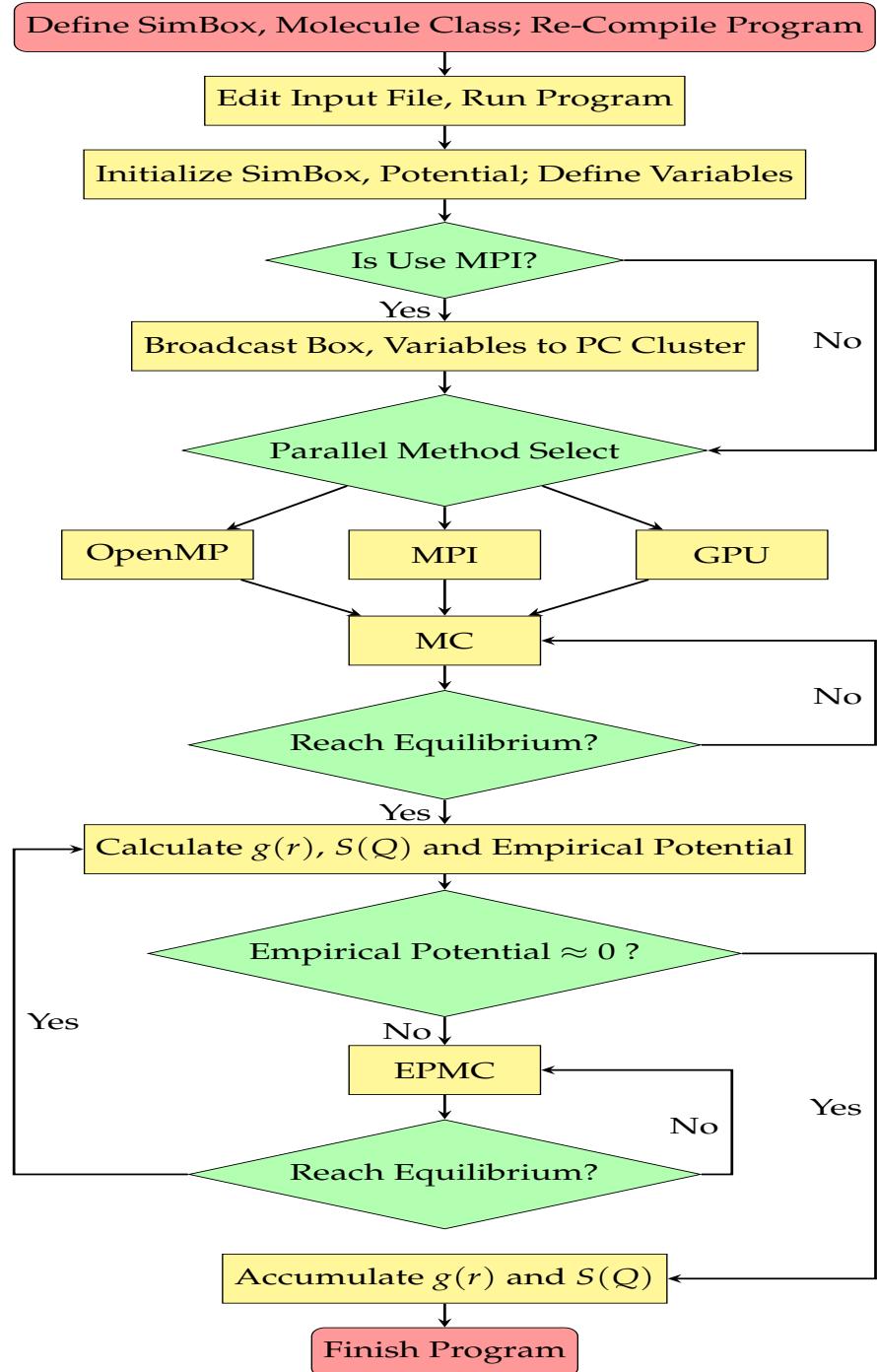


Fig. 1: The software arithmetic flow of EPSR++.

In addition, new atom potential and potential function can be added through editing the C++ head and source file **NDA\_parameter\_pot.h** and **NDA\_func\_pot.h**

After classes for special simulation are defined, users need to edit **Makefile** to select which parallel method will be used to accelerate the program *i.e.* ‘**-D USE\_MPI**’, ‘**-D USE\_OMP**’, ‘**-D USE\_GPU**’ compiler options mean employ mpich2 API to make the program parallel among different nodes of a computer cluster, use OpenMP API to make the program parallel among many cores of a CPU and enable GPU hardware acceleration respectively.

II. Edit Input File, Run Program: a text file as input needs to be offered to the program to initialize the simulation box and run parameters, such as the molar mass of molecules, molecular numbers, density, maximal movement step, MC equilibrium criteria, EPMC finish criteria, etc.

III. Initialize SimBox, Potential; Define Variables: Program initializes simulation box, reference potential arrays of all-atom type pairs with input parameters. The reference potential parameters are defined in a C++ head file, users can edit or define these parameters conveniently. Potential functions can also be defined by users, including L-J potential, electric field, potential truncation function.

If the program needs to be paralleled among different nodes of a computer cluster, the initialized simulation box and variables will be broadcast to different nodes with **MPI\_Bcast()** function of mpich2.

IV. MC: The program performs MC simulation with reference potentials until to equilibrium. 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 movement accept criterion. If  $\Delta U < 0$  the movement is accepted. If  $\Delta U > 0$ , the movement is accepted with a probability  $e^{-\Delta U/kT}$ .

The computation consuming of  $g(r)$  is in direct proportion to the second order of atom amount and it needs to be recalculated after every MC simulation step, so this step is the most calculation consuming part of the program. We designed three selectable parallel methods, *i.e.* OpenMP, MPI and GPU, to accelerate the calculation. Users could select a suitable method according to their hardware and requirements.

V. Calculate  $g(r)$ ,  $S(Q)$  and EMP: When MC gets to equilibrium, program calculate the difference neutron structure factor  $\Delta S(Q)$  between simulation  $S_{sim}(Q)$  and experiment  $S_{exp}(Q)$ . EMP is calculated by performing Fourier transition to  $\Delta S(Q)$ . Program add EMP and RP together as updated potential to perform MC simulation.

EPMC is very similar with MC except that: when the simulation gets to equilibrium, the program calculates EMP again, add it to the previous potential, and perform the simulation with the updated potential. Until EMP get to a very low value, the program stops to update simulation potential.

VI. Accumulate  $g(r)$  and  $S(Q)$ : When stopping update simulation potential, program will still perform EPMC to accumulate simulation data to improve statistics until obtain smooth  $g(r)$  and  $S(Q)$  curves. Except  $g(r)$  and  $S(Q)$ , program output a coordinate file include all atoms in text format as used in GROMACS (with the suffix of .gro)[25] or as used in LAMMPS (with the suffix of .xyz)[26], this file can be input to GROMACS or LAMMPS to calculate enthalpy, entropy, etc, and also can be visualized with visual molecular dynamics (VMD)[27]. Users can define, calculate and output any interesting variables which are related directly with the atomic structure of the sample by editing the source file.

### 3. Performance Test

Full hydrogen, full deuterated and half deuterated H<sub>2</sub>O samples are used to test the performance of the program about correctness and computation speed.

#### 3.1. Correctness

Based on SPCE[28] model and OPLS[19, 20] all-atom force field, a control sample about 200 thousand atoms (125Å) is generated with GRPMACS for testing the correctness of the program. The comparison of PDF distributions and neutron diffraction spectrum between MD sample and EPSR++simulation are shown in Fig. 2. As shown in the comparison, EPSR++simulation results are consisted with the MD truth of the control sample. Experiments neutron data of full hydrogen water, full deuterated water and half deuterated water distributed with EPSR and GudRun[29] in ISIS website are also used to test the program, and Fig.3 shows the EPSR++simulation results according to the experimental data. These show that the EPSR++program can reconstruct the atomic structure of experiment sample correctly based on neutron diffraction spectrum.

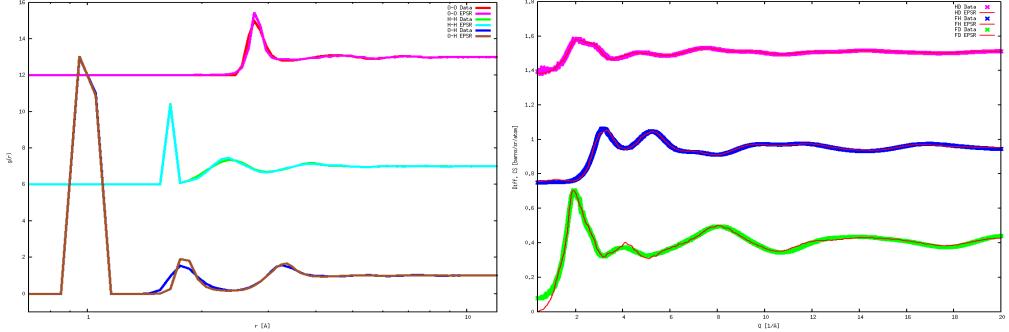


Fig. 2: Left: PDF comparison between EPSR++ and MD control sample. The curves from top to bottom are  $g(r)$  of O-O, H-H, O-H respectively. Right: neutron diffraction spectrum comparison between EPSR++ and MD control sample. The curves from top to bottom are  $S(Q)$  of half deuterated, full hydrogen and full deuterated samples. The solid lines are from the control samples, while the dashed lines are from EPSR++ simulation.

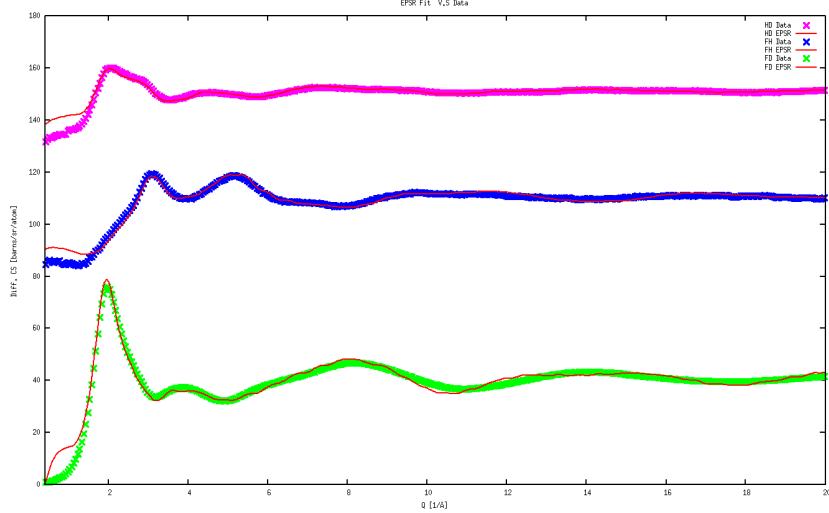


Fig. 3: Neutron diffraction spectrum comparison between EPSR and experiment sample. The curves from top to bottom are  $S(Q)$  of half deuterated, full hydrogen and full deuterated samples. The solid lines are from the control samples, while the dashed lines are from EPSR simulation.

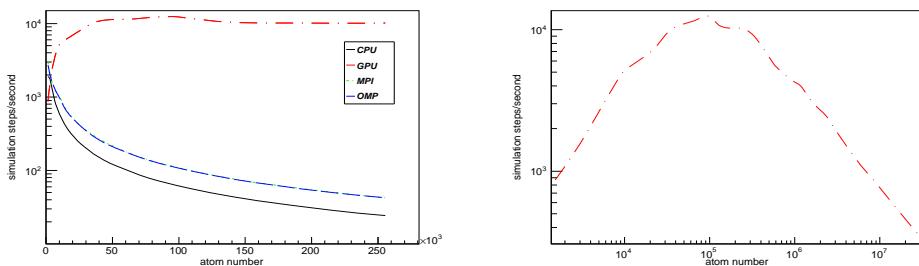
### 3.2. Computation Speed

In the program, three different parallel method (*i.e.* MPI, OpenMP, GPU) are provided to increase calculation velocity. A small computer cluster is used to test

the speed performance of different methods. The cluster uses CentOS 7.3 as the operating system and has 2 nodes. Each node has 2 Intel Xeon Scalable Gold 6126 CPU (2 Skylake-SP architecture, 12 Core, 24 Thread 2.6 GHz, Turbo 3.7 GHz 19.25 MB L3 Intel Smart Cache), 2 Nvidia Tesla V100 calculation GPU card and 128GB DDR4 ECC Registered Shared Memory. The two nodes are connected with InfiniBand (IB) connector (data transmission speed can up to 56Gb/s).

For implementing across nodes parallel, MPICH2 is used in the program. MPICH2 is based on message passing interface (MPI) standard and supports point-to-point and collective data communication among different nodes, so is very suited and high-efficiency in this program. For implementing shared memory multi-thread parallel within a computer or server node, Open Multi-Processing (OpenMP) is used in the C++ program. OpenMP syntax supports setting thread number dynamically and thread number cannot be known in advance under most conditions, so it is very convenient in programming. For supporting GPU hardware acceleration, CUDA toolkit is used in the program.

A speed comparison of these acceleration method is shown in Fig.4. and a more detailed quantitative comparison are listed in Tab.1. As shown in the figure and table, MPICH2 and OpenMP can improve the speed by a ratio of nodes or threads number; while GPU can provide a very excellent accelerate ratio. Most important of all, with GPU acceleration the program can simulate a system including more than 1 million atoms. This is an essential improvement because it makes the program can simulate a system larger than 220 Å, so that it can analyze samples with polymers in an all-atom model.



**Fig. 4:** Left: Calculation speed comparison among serial algorithm and different acceleration algorithms. Right: Calculation speed with GPU acceleration. X coordinate denotes atom number of simulation box, while Y coordinate denotes simulation step every second.

Table 1: Simulation speed with different parallel methods and serial algorithm.  
[steps/second]

atom num	$3 \times 10^3$	$3 \times 10^4$	$1 \times 10^5$	$2.5 \times 10^5$	$1.2 \times 10^6$	$3 \times 10^6$	$1 \times 10^7$	$3 \times 10^7$
GPU	1562	9507	12412	10189	3917	1963	757.0	340.3
MPI	2000	334.9	110.5	42.57	—	—	—	—
OMP	1989	327.5	109.4	42.44	—	—	—	—
CPU	1754	189.8	62.67	24.30	—	—	—	—

#### 4. Conclusion

An updated neutron diffraction data analysis software EPSR is re-programmed with object-oriented language C++. This makes the program very reflexible and friendly for users defining special molecules and MC random movement patterns. Potential functions and their corresponding parameters of the atomic force field can be modified or added through editing C++ head and source files. In addition, C++ is an easy-to-read high-level computer language, so users can try new algorithm and program flow to improve their analysis, and calculate and output any variables related with sample microstructure which are important to their analysis.

Except for parallel within a server node with OpenMP, parallel cross different nodes of a computer cluster and GPU hardware acceleration are supported with MPICH2 and CUDA, respectively. Especially with GPU acceleration, the calculation speed has a huge improvement so that the program has the capacity to analysis samples with macromolecules samples or nanoparticles using all-atom model.

Although the updated program is flexibility to users and has powerful calculation capacity, it was tested with very limited control samples and it is not a fully functional software package. So that the authors want to offer it as an open-source toolkit frame to all scientists. Users can contribute abundant new molecule classes, algorithm and analysis routines in the future to make the program more and more powerful.

#### 5. Acknowledgement

The authors thank the staff of the ISIS Disordered Materials Group for their patiently explanation to the principle of EPSR algorithm and providing experiment neutron diffraction data for program testing. This work is supported in part

by the National Key Research and Development Program of China under Contracts No. 2017YFA0403703.

---

## References

- [1] C. J. C. B. T. M. Willis, Computational Statistical Physics, no. Oxford University Press, 2017.
- [2] J. Xu, L. Mei, W. Yin, X. Wang, W. Cai, Z. Li, T. Bo, H. Chen, B. Wang, Y. Chen, Physical design of multipurpose physics neutron diffractometer for the CSNS, Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 927 (February) (2019) 161–168. doi:10.1016/j.nima.2019.02.011.
- [3] K. I. Shefer, Investigation of nanocatalysts using the PDF method, Materials Today: Proceedings 4 (11) (2017) 11360–11363. doi:10.1016/j.matpr.2017.09.009.  
URL <https://doi.org/10.1016/j.matpr.2017.09.009>
- [4] R. L. McGreevy, L. Pusztai, Reverse Monte Carlo Simulation: A New Technique for the Determination of Disordered Structures, Molecular Simulation 1 (6) (1988) 359–367. doi:10.1080/08927028808080958.
- [5] A. K. Soper, Empirical potential Monte Carlo simulation of fluid structure 202 (1996) 295–306.
- [6] M. A. T. Dove, M. A. G. Tucker, S. T. A. Wells, D. A. A. Keen, Reverse Monte Carlo methods 4 (March) (2002) 59–82.
- [7] R. L. McGreevy, L. Pusztai, Reverse Monte Carlo Simulation : A New Technique for the Determination of Disordered Structures 7022. doi:10.1080/08927028808080958.
- [8] M. A. T. Dove, M. A. G. Tucker, S. T. A. Wells, D. A. A. Keen, Reverse Monte Carlo methods 4 (March) (2002) 59–82.
- [9] J. C. Walter, G. T. Barkema, An introduction to Monte Carlo methods, Physica A: Statistical Mechanics and its Applications 418 (2015) 78–87. arXiv:1404.0209, doi:10.1016/j.physa.2014.06.014.  
URL <http://dx.doi.org/10.1016/j.physa.2014.06.014>

- [10] H. El-Rewini, M. Abd-El-Barr, Advanced Computer Architecture and Parallel Processing, 2005. doi:10.1002/0471478385.
- [11] OpenMP Architecture Review Board, OpenMP Application Program Interface, no. March. doi:10.1080/08905769008604595.
- [12] OpenMPI, Open MPI: A Message-Passing Interface Standard, Version 3.1, 2017. arXiv:a, doi:10.1021/ac800347j.  
URL <https://www.open-mpi.org/doc/v3.1/>
- [13] Z. Liping, G. Zhiqiang, C. Ruiwen, F. Yue, Z. Gaofeng, X. Benzhu, GPU-based efficient computation of power diagram, Computers & Graphics (xxxx). doi:10.1016/j.cag.2019.03.011.  
URL <https://linkinghub.elsevier.com/retrieve/pii/S0097849319300342>
- [14] M. Kiran Kumar, M. R. Abdel-Majeed, M. Annavaram, Efficient automatic parallelization of a single GPU program for a multiple GPU system, Integration (November). doi:10.1016/j.vlsi.2018.12.006.  
URL <https://doi.org/10.1016/j.vlsi.2018.12.006>
- [15] J. R. Cheng, M. Gen, Accelerating genetic algorithms with GPU computing: A selective overview, Computers and Industrial Engineering 128 (December 2018) (2019) 514–525. doi:10.1016/j.cie.2018.12.067.  
URL <https://doi.org/10.1016/j.cie.2018.12.067>
- [16] M. G. Awan, T. Eslami, F. Saeed, GPU-DAEMON: GPU algorithm design, data management & optimization template for array based big omics data, Computers in Biology and Medicine 101 (August) (2018) 163–173. doi:10.1016/j.compbiomed.2018.08.015.  
URL <https://doi.org/10.1016/j.compbiomed.2018.08.015>
- [17] Nvidia, Cuda C Best Practices Guide, Nvidia Corporation 7.5 (September) (2015) 85. doi:10.1086/676506.
- [18] NVIDIA, CUDA C Programming Guide (February). arXiv:0502092v1.  
URL <https://docs.nvidia.com/cuda/cuda-c-programming-guide/>

- [19] W. L. Jorgensen, J. Tirado-Rives, The opls potential functions for proteins. energy minimizations for crystals of cyclic peptides and crambin, *Journal of The American Chemical Society - J AM CHEM SOC* 110 (1988) 1657–1666. doi:10.1021/ja00214a001.
- [20] W. L Jorgensen, D. Maxwell, J. Tirado-Rives, Development and testing of the opls all-atom froce field on conformational energetics and properties of organic liquids, *J. Am. Chem. Soc.* 118 (1996) 11225–11236.
- [21] C. J. Dickson, B. D. Madej, Å. a. Skjevik, R. M. Betz, K. Teigen, I. R. Gould, R. C. Walker, Lipid14: The Amber Lipid Force Field - Supplementary Data, *Journal of Chemical Theory and Computation* 10 (2) (2014) 865–879. doi:10.1021/ct4010307.  
URL <http://pubs.acs.org/doi/abs/10.1021/ct4010307>
- [22] B. B.R. Brooks, C.L. Brooks, A.D. MacKerell Jr, L. Nilsson, R.J. Petrella, A. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, V. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, R. M. V. Ovchinnikov, E. Paci, R.W. Pastor, C.B. Post, J.Z. Pu, M. Schaefer, B. Tidor, H. L. Woodcock, X. Wu, W. Yang, D.M. York, M. Karplus, CHARMM: The Biomolecular Simulation Program 30 (10) (2010) 1545–1614. doi:10.1002/jcc.21287.CHARMM.
- [23] A. D. MacKerell, N. Banavali, N. Foloppe, Development and current status of the CHARMM force field for nucleic acids, *Biopolymers* 56 (4) (2000) 257–265. doi:10.1002/1097-0282(2000)56:4;257::AID-BIP10029;3.0.CO;2-W.
- [24] L. Lönnblad, CLHEP-a project for designing a C++ class library for high energy physics, *Computer Physics Communications* 84 (1-3) (1994) 307–316. doi:10.1016/0010-4655(94)90217-8.
- [25] S. Páll, R. Schulz, B. Hess, J. C. Smith, T. Murtola, M. J. Abraham, E. Lindahl, GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers, *SoftwareX* 1-2 (2015) 19–25. doi:10.1016/j.softx.2015.06.001.
- [26] S. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, *Journal of Computational Physics* 117 (6) (1997) 1–42. arXiv:nag.2347, doi:10.1006/jcph.1995.1039.

- [27] K. S. William Humphrey, Andrew Dalke, Vmd - visual molecular dynamics (1996). arXiv:arXiv:1503.05249v1, doi:10.1016/0263-7855(96)00018-5.
- [28] P. Mark, L. Nilsson, Structure and dynamics of the TIP3P, SPC, and SPC/E water models at 298 K, Journal of Physical Chemistry A 105 (43) (2001) 9954–9960. doi:10.1021/jp003020w.
- [29] A. K. Soper, GudrunN and GudrunX, 2012.