Check for updates

Received: 6 February 2023 | Revised: 21 April 2023 | Accepted: 27 April 2023

DOI: 10.1002/jcc.27132

SOFTWARE NOTE

Journal of Computational Chemistry WILEY

Enhanced sampling in explicit solvent by deep learning module in FSATOOL

Mincong Wu | Jun Liao | Zirui Shu | Changjun Chen

Biomolecular Physics and Modeling Group, School of Physics, Huazhong University of Science and Technology, Wuhan, Hubei, China

Correspondence
Changjun Chen, Biomolecular Physics and Modeling Group, School of Physics, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China.
Email: cjchen@hust.edu.cn

Funding information
National Natural Science Foundation of China

Abstract

FSATOOL is an integrated molecular simulation and data analysis program. Its old molecular dynamics engine only supports simulations in vacuum or implicit solvent. In this work, we implement the well-known smooth particle mesh Ewald method for simulations in explicit solvent. The new developed engine is runnable on both CPU and GPU. All the existed analysis modules in the program are compatible with the new engine. Moreover, we also build a complete deep learning module in FSATOOL. Based on the module, we further implement two useful trajectory analysis methods: state-free reversible VAMPnets and time-lagged autoencoder. They are good at searching the collective variables related to the conformational transitions of biomolecules. In FSATOOL, these collective variables can be further used to construct a bias potential for the enhanced sampling purpose. We introduce the implementation details of the methods and present their actual performances in FSATOOL by a few enhanced sampling simulations.

KEYWORDS

deep learning, enhanced sampling, molecular dynamics simulation, particle mesh Ewald method

1 | INTRODUCTION

Molecular dynamics simulation is widely used in the study of biomolecules, such as protein structure prediction,[1] protein folding,[2,3] protein–protein interaction,[4] and etc. In the simulation, the interactions between the solute and solvent molecules must be calculated precisely at every time step. At present, there are two kinds of widely-used solvent models. One is the implicit solvent model,[5] including the well-known generalized Born model like GBNeck model,[6] GBr6[7] and GBMV.[8] It considers the solvent as a continuous medium with a fixed dielectric constant. Due to the reduced degrees of freedom, the simulation work is simple and quite fast. However, it cannot present the details about how a particular water molecule moves around the solute in a dynamic way. Differently, the other solvent model contains more detailed information about the interactions between the solute and the solvent. It takes into account lots of solvent molecules explicitly in a periodic box. All the coordinates and velocities of the solvent atoms are updated as well as the solute atoms.

But, in comparison to the implicit solvent model, simulations with the explicit solvent model are very time-consuming. Most calculation quantities are spent on the long-range electrostatic interactions between the charged atoms in the periodic box. Specifically, the computation quantity scales as $O(N^2)$ for a system of N atoms. To decrease the calculation quantity, a classic Ewald summation method[9,10] has been proposed before. It replaces the real point charges by a set of smeared Gaussians centered on them. The charge densities become continuous in the space. This allows the long-range electrostatic energy to be calculated in the reciprocal space efficiently by the fast Fourier transform method (FFT). After considering the short-range interactions in the direct space and the self-energy corrections, one can have the whole electrostatic energy of the system. The Ewald summation method reduces the computation complexity to $O(N\log N)$. Actually, the calculation can be further accelerated by placing the point charges on evenly distributed grids instead of their real positions, like particle–particle–particle–mesh (P^3M),[11] particle mesh Ewald (PME),[12] and the smooth particle mesh Ewald (SPME)[13] method.

 © 2023 Wiley Periodicals LLC.

One year ago, we released a program named FSATOOL 2.0 for the simulation of biomolecules.[14] It contains various functional modules, including the molecular dynamics simulation module, enhanced sampling module, Markov state module and trajectory data analysis module. Users can use FSATOOL to do the simulation and analysis work by many methods, such as replica exchange molecular dynamics (REMD),[15] adaptively biased molecular dynamics (ABMD),[16] mixing REMD,[17] snapshot clustering (k-means,[18] k-medoids[19]), Markov state model analysis,[20–22] principal component analysis (PCA),[23] time-lagged or time-structure based independent component analysis method (TICA),[24,25] weighted histogram analysis method (WHAM)[26,27] and etc.

Previous FSATOOL has its own molecular dynamics engine that is runnable on both CPU and GPU.[14] But the simulation can only be performed in vacuum or implicit solvent. In this work, we implement the SPME explicit solvent method[13] to make the program more complete. Furthermore, to do the conformational analysis on the trajectory data, FSATOOL contains four dimension reduction methods, called principal component analysis (PCA),[23] time-lagged independent component analysis (TICA),[24,25] state-free reversible VAMPnets (SRV)[28] and time-lagged autoencoder (TAE).[29] The first two are traditional methods and the last two are deep learning-based methods developed in recent years. Compared to PCA and TICA, SRV and TAE are more suitable for the determination of nonlinear collective variables. PCA and TICA have been integrated into previous FSATOOL. In this work, we build a new deep learning module and implement SRV and TAE completely.

The upgraded FSATOOL has the following features.

First, the GPU-accelerated molecular dynamics simulation engine with SPME is completely written in Fortran and CUDA Fortran language. The installation is simple. It only needs three additional software, including the NVIDIA HPC SDK for the compilation of the source codes (https://developer.nvidia.com/hpc-sdk), FFTW for the fast Fourier transformation on CPU (http://fftw.org) and NetCDF for the recording of trajectory data in the molecular dynamics simulation (https://www.unidata.ucar.edu/software/netcdf).

Second, the new SPME simulation engine is made to be compatible with the existed enhanced sampling module. Users can do either the standard or enhanced sampling simulations in explicit solvent conveniently. The usage is similar to previous simulations in implicit solvent.[14] For example, a variable *task* must be provided in the input file to specify a specific enhanced sampling method. *task* = "temd" specifies the replica exchange molecular dynamics method,[15] *task* = "flood" specifies the ABMD[16] or mixing REMD method[17] and *task* = "smd" specifies the steered molecular dynamics method.[30] All the simulations support the sampling in the isothermal and isobaric ensembles.

At last, FSATOOL supports SRV[28] and TAE[29] deep learning methods for the trajectory data analysis work. It feeds the high-dimensional trajectory data into a neural network for training. When the training work is done, the dimensionality of the data is reduced to a set of representative collective variables (components). The new deep learning codes have been embedded into FSATOOL completely. Both the data training and prediction can be simply carried out by some related commands. Original trajectory data are readable by the new deep learning module. No data conversion steps are required at all. And most importantly, the reduced collective variables (components) can be further used to construct an adaptive bias potential in the subsequent enhanced sampling simulation. With the deep learning-based potential, one can search the conformational space of biomolecules in a more efficient way.

The whole paper is organized as follows. We first give a brief description on the SPME method,[13] especially its different energy terms. They are the core part in the implementation of SPME. Second, we introduce the existing approach to do an isothermal-isobaric simulation for a system in a rectangular or non-rectangular periodic box. Third, we discuss the basic ideas in the deep learning-based methods SRV[28] and TAE[29] and their applications in the calculation of the representative collective variables from a neural network. After then, we give the SPME calculation results by FSATOOL and compare them to another two popular programs, AMBER 18[31] and OpenMM 7.7.0.[32] These comparisons are done for the check of the validity of the SPME codes in FSATOOL. Finally, a set of accelerated molecular dynamics simulations are carried out in the explicit solvent. They present the actual performance of the upgraded FSATOOL.

2 | MATERIALS AND METHODS

2.1 | Energy components in Ewald summation

In a periodic box system of charged atoms, the whole electrostatic energy is expressed by the following formula:

$$E = \frac{1}{2}\sum_{i \neq j}\frac{q_i q_j}{r_{ij}} + \frac{1}{2}\sum_{\mathbf{n} \neq 0}\sum_{i,j}\frac{q_i q_j}{r_{ij} + \mathbf{n}} \tag{1}$$

where $\mathbf{n}$ is the periodic box index vector, q_i and q_j are the atomic charges and r_{ij} is the atom-to-atom distance. The formula indicates that the electrostatic energy depends on all the interactions between the charged atoms in the same central box (first term) and their images in different boxes (second term). There are too many terms in the formula that make the energy converge very slowly. So, the Ewald summation method[9,10] uses a set of continuous Gaussian functions to replace the original point charges. Specifically, every atom-atom electrostatic interaction is divided into two parts.[10]

$$E_{ij} = \frac{1}{2}\frac{q_i q_j}{r_{ij}}\varphi(r_{ij}) + \frac{1}{2}\frac{q_i q_j}{r_{ij}}\left[1 - \varphi(r_{ij})\right] \tag{2}$$

where i and j are atom indices. In the expression, the first term represents the short-range interaction and the second is the long-range interaction. $\varphi(r)$ is the complementary error function (erfc) with a constant β:

$$\varphi(r) = \frac{2}{\sqrt{\pi}}\int_{\beta r}^{\infty} e^{-t^2} dt \tag{3}$$

In practice, the short-range term is simply calculated in the direct space because it converges quickly on the distance. On the contrary,

1096987x, 2023, 22, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/jcc.27132 by Huazhong University Of Sci & Tech, Wiley Online Library on [17/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

the long-range term converges very slowly. To resolve this issue, all the long-range terms are collected together and placed into an integral[10]:

$$\begin{aligned}&\frac{1}{2}\sum_{i,j}\frac{q_iq_j}{r_{ij}}\left[1-\varphi\left(r_{ij}\right)\right]\\&=\frac{1}{2}\int\left[\sum_{i,j}q_iq_j\zeta\left(\mathbf{r}-\mathbf{r}_{ij}\right)-\sum_{i,j}q_iq_j\zeta\left(\mathbf{r}-\mathbf{r}_{ij}\right)\delta(\mathbf{r})\right]\frac{1-\varphi(r)}{r}d\mathbf{r}\\&=\frac{1}{2}\int\sum_{i,j}q_iq_j\zeta\left(\mathbf{r}-\mathbf{r}_{ij}\right)\frac{1-\varphi(r)}{r}d\mathbf{r}-\frac{1}{2}\int\sum_{i,j}q_iq_j\zeta\left(\mathbf{r}-\mathbf{r}_{ij}\right)\delta(\mathbf{r})\frac{1-\varphi(r)}{r}d\mathbf{r}\end{aligned}\tag{4}$$

where $\mathbf{r}_{ij}$ is the displacement vector from atom i to atom j. $\delta(\mathbf{r})$ is the Dirac delta function and $\zeta(\mathbf{r})$ is the Kronecker delta function. By taking a Fourier transform, the first term in the formula is deduced to the energy in the reciprocal space and the second term is the sum of the self-energy corrections for the point charges.

Finally, there are three energy terms in the Ewald sum, including the direct space term E_{dir}, the reciprocal space term E_{rec}, and the self-energy correction term E_{self}. All of them are listed below[13]:

$$\begin{aligned}E_{\text{dir}}&=\frac{1}{2}\sum_{\mathbf{n}}\sum_{i,j}\frac{q_iq_j\operatorname{erfc}\left(\beta\left|\mathbf{r}_j-\mathbf{r}_i+\mathbf{n}\right|\right)}{\left|\mathbf{r}_j-\mathbf{r}_i+\mathbf{n}\right|}\\E_{\text{rec}}&=\frac{1}{2\pi V}\sum_{\mathbf{m}\neq\mathbf{0}}\frac{e^{-\pi^2|\mathbf{m}|^2/\beta^2}}{|\mathbf{m}|^2}S(\mathbf{m})S(-\mathbf{m})\\E_{\text{self}}&=-\frac{\beta}{\sqrt{\pi}}\sum_{i=1}^{N}q_i^2\end{aligned}\tag{5}$$

where $\mathbf{r}_i$ and $\mathbf{r}_j$ are the coordinates of atoms, $\mathbf{m}$ is the reciprocal lattice vector, and $S(\mathbf{m})$ is the structure factor. $S(\mathbf{m})$ will be discussed in detail later.

2.2 | Short-range interactions in the direct space

As mentioned above, the calculation of the short-range energy term E_{dir} is completed in the direct space. This is almost the same as the implicit solvent model. However, because of large number of atoms in the system, these atom–atom interactions in the sum of E_{dir} are too many to compute. One cutoff distance R_c needs to be involved in the calculation. If it is shorter than one half of the periodic box length, there is only one term left for each atom–atom pair in the sum.

Moreover, to further reduce the calculation quantity on the distances between the atoms, one can collect all the close atom-atom pairs and put them into a large array called "neighbor list".[33] It is built on-the-fly in the simulation with an extended cutoff distance R_e[34]

$$R_e=R_c+R_b\tag{6}$$

In the formula, the buffer distance R_b is a constant. Generally, it is several times of the maximum movement distance of the atoms at a single time step. If any atom's movement exceeds the buffer distance, the whole neighbor list has to be rebuilt completely.

With a neighbor list, the energy and force calculation should be greatly reduced. However, during the building of the neighbor list, one still has to spend a lot of time to check the distances between all the atom-atom pairs. A common way to save the calculation time is dividing the whole periodic box into many smaller boxes by the extended cutoff distance R_e at the beginning of the simulation. Thus, for a single atom, its neighbors only exist in the central box containing the atom or other adjacent boxes. Specifically, if the small box's width is equivalent to or larger than R_e, the atom only needs to search for its neighbors in the 26 surrounding boxes as well as the central box (27 in total). In practice, to avoid the duplication in the force calculation, only 14 of them are necessary. Similarly, if the small box's width decreases to $R_e/2$, each atom must search 125 boxes to find all the neighbor atoms within the extended cutoff distance.

After building the neighbor list, the short-range electrostatic energy term E_{dir} can be calculated by the first formula in Equation (5) in a direct way. It should be noted that the complementary error function $erfc(x)$ in the formula is an integral with a large interval, which is hard to compute. On GPU, we use a simplified expression of $erfc(x)$.[35] But on CPU, it is still too slow. So, we divide the distance into bins and calculate the error function numerically by Taylor series as in AMBER.[31]

2.3 | Long-range interactions in the reciprocal space

As to the electrostatic energy in the reciprocal space, it has one key variable, structure factor $S(\mathbf{m})$[13]

$$S(\mathbf{m})=\sum_{i}^{N}q_ie^{2\pi i\mathbf{m}\cdot\mathbf{r}_i}\tag{7}$$

where N is the total number of charged atoms and $\mathbf{r}_i$ is the Cartesian coordinate of atom i. $\mathbf{m}$ is the reciprocal lattice vector. It is formulated by three integers m_α and three conjugate reciprocal vectors $\mathbf{a}_\alpha^*$ ($\alpha=1, 2, 3$)[13]

$$\mathbf{m}=\sum_{\alpha}m_\alpha\mathbf{a}_\alpha^*\tag{8}$$

With the conjugate reciprocal vectors, the Cartesian coordinates of atoms can be transformed into the scaled fractional coordinates[13]

$$u_\alpha=K_\alpha\mathbf{a}_\alpha^*\cdot\mathbf{r}_i\tag{9}$$

where K_α is the number of FFT grids along α-th lattice vector. After combining the above two expressions, the exponential in $S(\mathbf{m})$ is further rewritten into three independent factors as follows.[13]

$$e^{2\pi i\mathbf{m}\cdot\mathbf{r}_i}=e^{2\pi i\frac{m_1}{K_1}\cdot u_1}\cdot e^{2\pi i\frac{m_2}{K_2}\cdot u_2}\cdot e^{2\pi i\frac{m_3}{K_3}\cdot u_3}\tag{10}$$

And each factor can be approximately expressed by the B-spline method.[36] For example, the first factor is[13]

1096987x, 2023, 22, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/jcc.27132 by Huazhong University Of Sci & Tech, Wiley Online Library on [17/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

$$e^{2\pi i\frac{m_1}{K_1}u_1} \approx b_1(m_1)\sum_{k=-\infty}^{\infty} e^{\frac{2\pi i m_1 k}{K_1}} M_n(u_1-k) \tag{11}$$

where M_n is B-spline's basis function with order n and $b_1(m_1)$ equals

$$b_1(m_1) = 1/\sum_{j=1}^{n-1} e^{\frac{2\pi i m_1 j}{K_1}} M_n(j) \tag{12}$$

Similarly, all atomic charges q_i can also be spread onto the FFT grids in the periodic box by the B-spline method.[36] For example, in case of cubic B-spline, each atom spreads its charge onto four adjacent grids in each direction (64 grids in the three-dimensional space). And in turn, each grid charge is obtained by the combination of the surrounding atom charges:

$$Q(k_1,k_2,k_3) = \sum_{n_1,n_2,n_3=-\infty}^{\infty}\sum_{i=1}^{N} q_i M_4(u_{1i}-k_1-n_1K_1) M_4(u_{2i}-k_2-n_2K_2)M_4(u_{3i}-k_3-n_3K_3) \tag{13}$$

where k_1, k_2, and k_3 are the FFT grid indices along three directions, respectively, and K_1, K_2, and K_3 are their maximum numbers. Integers n_1, n_2, and n_3 are used to distinguish the image grid charges in different periodic boxes. Then a Fourier transform on the grid charge function Q is expressed by

$$F(Q)(m_1,m_2,m_3) = \sum_{k_1=0}^{K_1-1}\sum_{k_2=0}^{K_2-1}\sum_{k_3=0}^{K_3-1} Q(k_1,k_2,k_3)e^{2\pi i m_1 k_1/K_1}e^{2\pi i m_2 k_2/K_2}e^{2\pi i m_3 k_3/K_3} \tag{14}$$

Combining the above two equations together, it gives a simplified form of the structure factor $S(\mathbf{m})$

$$S(\mathbf{m}) = b_1(m_1)b_2(m_2)b_3(m_3)F(Q)(m_1,m_2,m_3) \tag{15}$$

After putting it into the reciprocal sum in Equation (5), the long-range electrostatic energy is derived in the reciprocal space finally[13]

$$E_{\text{rec}} = \frac{1}{2\pi V}\sum_{\mathbf{m}\neq 0}\frac{e^{-\pi^2|\mathbf{m}|^2/\beta^2}}{|\mathbf{m}|^2}|b_1(m_1)|^2\cdot|b_2(m_2)|^2\cdot|b_3(m_3)|^2 \cdot F(Q)(\mathbf{m})F(Q)(-\mathbf{m}) \tag{16}$$

For more details of SPME, readers are strongly referred to the reference paper.[13]

2.4 | Virial theorem and pressure coupling

In case of simulations in the isothermal-isobaric ensemble, we apply the Berendsen method[37] to control the pressure as well as in the temperature coupling procedure. It first calculates the instant pressure tensor by the virial theorem[37]:

$$\mathbf{P} = \frac{1}{V}\sum_i m_i\mathbf{v}_i\mathbf{v}_i^T + \sum_{i<j}\mathbf{r}_{ij}\mathbf{F}_{ij}^T \tag{17}$$

where $\mathbf{v}_i$ are the velocities of atoms and $\mathbf{F}_{ij}$ are the forces between them. For an isotropic system, the tensor is simplified to a scalar:

$$P = \frac{1}{3}\text{Tr}(\mathbf{P}) \tag{18}$$

As noted in reference 37, the calculation quantity of the virial can be further reduced if it is described by the centers of mass of the molecules.

With the instant pressure, one can control the pressure by scaling the coordinates of atoms by a factor[37]

$$\mu = \left(1-\frac{\kappa\Delta t(P_0-P)}{\tau_p}\right)^{\frac{1}{3}} \tag{19}$$

where κ is the isothermal compressibility of water and τ_p is a time constant for the pressure coupling in the simulation.

2.5 | Simulation in a nonrectangular periodic box

Besides the rectangular periodic box, there are many other periodic boxes in different shapes. For example, the truncated octahedral box has eight hexagonal faces and six square faces. All edges of the box are of equal length.

Unlike the rectangular box, the lattice vectors of a nonrectangular box are not orthogonal to each other. We have to transform all the atom coordinates into the fractional coordinates and find the closest atoms or images for the force calculation. The transformation matrix from the fractional coordinate to the Cartesian coordinate is[38]

$$\mathbf{a} = \begin{pmatrix} a & b\cdot\cos\gamma & c\cdot\cos\beta \\ 0 & b\cdot\sin\gamma & \dfrac{c\cdot\cos\alpha - c\cdot\cos\beta\cdot\cos\gamma}{\sin\gamma} \\ 0 & 0 & c\sqrt{\sin^2\beta - \left(\dfrac{\cos\alpha-\cos\beta\cdot\cos\gamma}{\sin\gamma}\right)^2} \end{pmatrix} \tag{20}$$

where α, β and γ are the angles between the axes describing the periodic box's geometric shape. Its inverse matrix $\mathbf{a}^*$ is used for the coordinate transformation in the reversed direction.

Compared to the rectangular periodic box, nonrectangular box require fewer solvent molecules to cover the same solute molecule. The calculation quantity is reduced correspondingly.

2.6 | GPU acceleration

GPU programming has been widely applied in many scientific software. As a powerful computation device, GPU has lots of stream processes to process the data at the same time. So, different to the

1096987x, 2023, 22, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/jcc.27132 by Huazhong University Of Sci & Tech, Wiley Online Library on [17/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

sequential calculation on CPU, the energy and force calculations on GPU are carried out in a parallel way inherently.

In FSATOOL, we use some highly efficient techniques that have been proposed before.[14,31,32] They are mainly listed as follows.

i. To ensure the spatial locality of neighbor atoms, a cell hash lookup table is constructed in the periodic box.[39] All the molecules are categorized by their indices in the hash table. And their atoms are sorted and divided into a set of 32-atom groups. Those atoms in the same group shall be close to each other in space.
ii. On GPU, any computation is executed simultaneously on a block of 32 threads, which is called a *warp*. For this reason, we divide the whole nonbonded interactions into many tiles as in OpenMM.[32] Each tile contains the interactions between 32 atoms in one group and 32 neighbor atoms distributed in the same or the rest groups. Thus, all the threads in a *warp* only need to process the data belonging to these 64 atoms (32 atoms for the same group), like their coordinates, charges and forces. This shall greatly improve the data accessing performance in the global memory.
iii. The GPU code in FSATOOL is written by CUDA Fortran in NVIDIA HPC SDK.[40] Atomic operation and shuffle operation are both applied in the energy and force calculation for the acceleration purpose. Atomic operation allows different threads to write to the same global array in a safe way. No large array is required any more for the storage of the temporary energy or force data from different threads. Shuffle operation allows one thread to transfer its data directly to another one in the same *warp*. This prevents the waste of the additional shared memory on GPU.

2.7 | Input parameters for SPME simulation

The basic usage of FSATOOL has been presented in the Supplementary Material of our pervious paper.[14] Readers can also find the reference manual on the webpage: https://gitee.com/chen_changjun/fsatool-2.0/wikis/pages. Here we only introduce the usage related to the new SPME simulation in the explicit solvent. To launch the simulation, FSATOOL requires the same namelist *sim* as that in the implicit solvent simulation. A template of the namelist is presented as follows

```
&sim
    ipb = 1, pressure0 = 2.0, taup = 1.0, discut = 9.0
    icuda = 1, lagfreq = 0, maxstep = 100, printcvstep = 10
    savetrajstep = 10,000,000, deltatime = 0.001, kelvin0 = 300.0, iseed = 0
```

Most parameters in the namelist have been explained before,[14] except the four new parameters on the first line. *ipb* specifies the sampling ensemble in the simulation. $ipb = 1$ is the isothermal-isovolumetric ensemble and $ipb = 2$ is the isothermal-isobaric ensemble. *pressure0* gives the reference pressure in the isothermal-isobaric simulation (in unit of bar). *taup* is the relaxation time of the coupling of the pressure (in unit of ps). *discut* is the cutoff distance in the force calculation (in unit of Å). As to the information about the periodic box, like the box angle and dimensions, they are provided in the topology file generated by the open-source AmberTools suite.[31]

Furthermore, compared to PME, Lennard-Jones PME (LJ-PME)[41,42] is a more accurate method to calculate the long-range van der Waals interactions between atoms in a periodic box. Its theory background and implementation details are presented in Supplementary Material (Section S1. Lennard–Jones PME calculation).

2.8 | Dimension reduction by deep learning methods

Deep learning methods have been widely applied in many molecular simulation studies. For example, Hernandez et al. provide a method called variational dynamics encoder (VDE).[43] It has a neural network with three parts: one encoder network, one variational layer and one decoder network. VDE is able to capture nontrivial dynamics in the protein folding process. Bhowmik et al. use a convolutional variational autoencoder (CVAE) to learn some latent features for proteins from the simulation trajectories.[44] And Brace et al. present a Stream-AI-MD method for the enhanced sampling purpose.[45] The method lets the simulation data stream continuously to a convolutional neural network. The network generates a latent space for the searching of the interesting states, which can be further used as new start points for the subsequent simulations.

In the new FSATOOL, we implement two deep learning-based methods to do the dimension reduction work, called time-lagged autoencoders (TAE)[29] and state-free reversible VAMPnets (SRV).[28] They are for the determination of the most relevant collective variables in a long trajectory.

TAE uses a symmetric neural network consisting of two parts: encoder E (from the input layer to the bottleneck layer) and decoder D (from the layer next to the bottleneck layer to the output layer). The encoder compresses the conformational information $\mathbf{x}_t$ at time t into a latent space (operation $E(\mathbf{x}_t)$) and the decoder reconstructs the time-lagged information at $t+\tau$ from the latent space (operation $D(E(\mathbf{x}_t))$). To minimize the regression error between the decoded information and the original information, one can use the following loss function[29]

$$L = \sum_t \left\| \mathbf{x}_{t+\tau} - D(E(\mathbf{x}_t)) \right\|^2 \tag{21}$$

The neural network of TAE contains the time propagation information from t to $t+\tau$. Reversely, the network can also be trained in a self-supervised way by the sampling data in the simulation. Moreover, during the time propagation, all the data go through an encoding-decoding process. The neurons in the bottleneck layer constitute a reduced space from the high dimensional conformational space.

The other method SRV does the dimension reduction work in a different way. As a variant of VAMPnets method,[46] SRV first encodes

1096987x, 2023, 22, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/jcc.27132 by Huazhong University Of Sci & Tech, Wiley Online Library on [17/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

the conformational information $\mathbf{x}_t$ at time t and time-lagged information $\mathbf{x}_{t+\tau}$ at $t+\tau$ into the low dimensional latent space independently by a Siamese neural network.[28] The encoded state functions in the latent space are $O(\mathbf{x}_t)$ and $O(\mathbf{x}_{t+\tau})$. In the latent space, the time propagation from $O(\mathbf{x}_t)$ to $O(\mathbf{x}_{t+\tau})$ is determined by a transfer operator $\mathbf{p}_\tau$. Generally, we are interested in those slow modes during the propagation, which correspond to the eigenfunctions (ψ_i) of $\mathbf{p}_\tau$ with large eigenvalues (λ_i). This requires to divide all the sampling data into a set of clusters and then construct a transition matrix.[47] To avoid the clustering operation, SRV applies a variational approach.[28] The basic theory is that any state function $u(\mathbf{x}_t)$ (orthogonal to the trivial eigenfunction ψ_0 of $\mathbf{p}_\tau$) can be represented by a linear expansion of nontrivial eigenfunctions (ψ_i) with expansion coefficients (a_i).[28]

$$u(\mathbf{x}_t) = \sum_{i\geq 1} a_i \psi_i(\mathbf{x}_t) \tag{22}$$

Now $u(\mathbf{x}_t)$ is put into the following expression of $\tilde{\lambda}$ (here $\langle * \rangle_\pi$ is the integration in the conformational space with the equilibrium distribution π).[28]

$$\begin{aligned}\tilde{\lambda} &= \frac{\langle u(\mathbf{x}_t)|u(\mathbf{x}_{t+\tau})\rangle_\pi}{\langle u(\mathbf{x}_t)|u(\mathbf{x}_t)\rangle_\pi} \\ &= \frac{\langle u(\mathbf{x}_t)|\mathbf{p}_\tau \circ u(\mathbf{x}_t)\rangle_\pi}{\langle u(\mathbf{x}_t)|u(\mathbf{x}_t)\rangle_\pi} \\ &= \frac{\left\langle \sum_{i\geq 1} a_i\psi_i(\mathbf{x}_t) \middle| \sum_{i\geq 1} a_i\lambda_i\psi_i(\mathbf{x}_t)\right\rangle_\pi}{\left\langle \sum_{i\geq 1} a_i\psi_i(\mathbf{x}_t) \middle| \sum_{i\geq 1} a_i\psi_i(\mathbf{x}_t)\right\rangle_\pi} \\ &= \frac{\sum_{i\geq 1}\lambda_i a_i^2}{\sum_{i\geq 1} a_i^2} \leq \frac{\sum_{i\geq 1}\lambda_1 a_i^2}{\sum_{i\geq 1} a_i^2} = \lambda_1\end{aligned} \tag{23}$$

If the state function $u(\mathbf{x}_t)$ happens to be the first nontrivial eigenfunction $\psi_1(\mathbf{x}_t)$, then $\tilde{\lambda}$ equals the first nontrivial eigenvalue λ_1. In another word, we can adjust the state function and maximize $\tilde{\lambda}$ to approach the eigenfunction $\psi_1(\mathbf{x}_t)$.

In SRV, the state function $u(\mathbf{x}_t)$ consists of a nonlinear basis set ($O_i(\mathbf{x}_t)$) from the outputs of the neural network.[28]

$$u(\mathbf{x}_t) = \sum_i c_i O_i(\mathbf{x}_t) \tag{24}$$

c_i are the expansion coefficients. According to the variational principle, all c_i can be determined by inserting the state function into the expression $\tilde{\lambda}$ in Equation (23) and maximizing the value. This becomes a generalized eigenvalue problem. It must be noted that all outputs ($O_i(\mathbf{x}_t)$) from the neural network are not known initially. The network should be trained with a loss function[28]

$$L = \sum_i g(\tilde{\lambda}_i) \tag{25}$$

here g is a monotonically decreasing function and i is the index of the eigenfunction. About SRV, readers are strongly referred to the reference paper for complete discussions.[28] When all eigenfunctions are found, they can be used as the Deep-TICA collective variables in the enhanced sampling simulations.[48]

Both of TAE and SRV have been implemented in the deep learning module in FSATOOL. Their usages are very simple. To build and train a neural network, one needs to run a command as follows.

```
mpirun – n 1 fsatool analysis deep – i deep.in
```

The last argument deep.in is an input file. It contains a namelist *deep* with all required parameters.

```
&deep
  task = "srvtrain", trajfile = "levelinfo_0.txt", networkfile = "network.txt"
  layers = 136, 50, 20, activation = "sigm", cvstart = 4, cvend = 139
  nepoch = 100, ntau = 1000, learnrate = 0.0001, batchsize = 10
  normalfactor = 1.0, normalmode = 1
```

here *task* is the job name, *task* = "srvtrain" lets FSATOOL do the training work by SRV method. Similarly, if *task* = "taetrain", the training work is done by TAE method. *trajfile* is the name of the trajectory file that is generated by the *sim* module in FSATOOL in a molecular dynamics simulation. *networkfile* is the file name to save all neural network parameters. *layers* = 136, 50, 20 determines the structure of the network. In this case, the network consists of three layers, including one input layer (136 neurons), one hidden layer (50 neurons) and one output layer (20 neurons). The parameter *activation* assigns the type of the activation function (in FSATOOL, it can be "sigm","tanh","relu"). *cvstart* and *cvend* are the start and end index of the collective variables recorded in the trajectory file. They serve as the inputs of the neural network. *nepoch* is the number of epochs in the training work. *ntau* is the lagged time steps. *learnrate* is the learning rate. *batchsize* is the batch size used by the minibatch gradient descent method.[49] The last two parameters, *normalfactor* and *normalmode*, are for the normalization of the input data. *normalmode* = 1 indicates that all input data are scaled into the interval [0, *normalfactor*] and *normalmode* = 2 scales the data into the interval [−*normalfactor*, *normalfactor*].

As to the prediction work, the command is the same as before. But the required parameters in the *deep* namelist are much fewer.

```
&deep
  task = "srvpredict", trajfile = "levelinfo_0.txt", networkfile = "network.txt" /
```

This is because that most required parameters have already been saved in the network file in the previous training work. Here *task* = "srvpredict" means the prediction is carried out by the SRV method and *task* = "taepredict" indicates the TAE method. When the prediction work is finished, FSATOOL generates a file named "predict.txt". In the file, all the data in the trajectory has been reduced to the low dimensional component space. The components can be either the hidden states of neurons in the bottleneck layer (TAE network) or outputs of the neurons in the last layer (SRV network).

1096987x, 2023, 22, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/jcc.27132 by Huazhong University Of Sci & Tech, Wiley Online Library on [17/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

In FSATOOL, the reduced collective variables (components) can be further used for the enhanced sampling purpose. The *sim* module in FSATOOL reads the parameters in the network file at the beginning and constructs an adaptive bias potential in the dimension reduced space. During the molecular dynamics simulation, the bias potential is updated on the fly to flatten out the barriers on the free energy landscape. More information about the usage of the deep learning module is given in the example files of FSATOOL.

The SRV training and enhanced sampling work are performed in two different calculations. But inspired by the Stream-AI-MD method in reference 45, they can be integrated together into one framework. The training data are forwarded to the enhanced sampling simulation and the unbiased sampling data come back for data training. The implementation and performance of the "streaming" method are presented in Supplementary Material (Section S2. Enhanced sampling by a "streaming" manner).

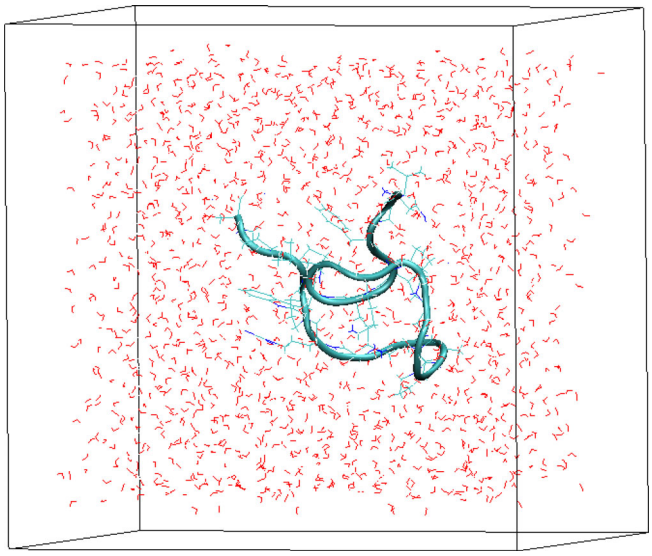

FIGURE 1 Prepared conformation of trpcage for the formal calculations in this work. The figure is produced by VMD.[60]

2.9 | Simulation setup

To present the performance of FSATOOL in the molecular dynamics simulation, we use a mini-protein trpcage[50] (PDB ID: 1L2Y) as a test model. Its amino acid sequence is NASN-LEU-TYR-ILE-GLN-TRP-LEU-LYS-ASP-GLY-GLY-PRO-SER-SER-GLY-ARG-PRO-PRO-PRO-CSER. The molecule's topology file is built by *tleap* in AmberTools[31] with FF14SBonlysc force field.[51] The initial conformation is a fully extended β-strand. It is first optimized in GBNeck2 implicit solvent[52] by the steepest descent method[53] for 500 steps and then optimized by the conjugate gradient method[54] for another 500 steps. Then the structure is further equilibrated at 300 K for 20 ns in the same implicit solvent. The temperature is controlled by Langevin dynamics.[55,56] The molecular dynamics integration method is leapfrog[57] and the time step is 2.0 fs. All covalent bonds involving hydrogen are constrained by SHAKE algorithm.[58] At the end of the equilibrium simulation, the molecule falls into a collapsed structure. We put the collapsed structure into a periodic box of TIP3P waters[59] with sodium and chloride ion concentration of approximately 0.1 M. The box size is 43.844 Å × 43.470 Å × 37.451 Å. At last, the whole system is further optimized for 10,000 steps by the steepest descent method[53] and 10,000 steps by the conjugate gradient method.[54] In the explicit solvent simulation, all atom-atom interactions are calculated with a cutoff of 10.0 Å. Other simulation parameters are the same as before. The prepared system is shown in Figure 1. It is used for the subsequent presentation work in this paper.

3 | RESULTS AND DISCUSSIONS

At the beginning, we compute the SPME[13] potential energy for the structure of trpcage in Figure 1 by FSATOOL and compare the data with AMBER 18[31] and OpenMM 7.7.0.[32] The computer hardware is Xeon E5-2620 (CPU) and GeForce RTX 2080 (GPU). All the data are presented in Table 1. For OpenMM, the total energy is calculated by its internal function "simulation.context.getState()" and all the individual energy components are calculated by an external function "parmed.openmm.energy_decomposition_system()" in ParmEd.[61] As we can see, all the individual energy components on both CPU and GPU, including the total energy, are basically consistent between these three programs. The slight differences on the numbers are because of different parameters used in the calculation (like the parameter β in Equation 3 and FFT grid number K_α in Equation 9). Overall, FSATOOL is able to produce the reasonable potential energies for a periodic system with the all-atom force field.

In the following, we begin to present the formal simulation results. We perform one 2000 ns molecular dynamics simulation at 300 K for trpcage in the explicit solvent. The sampling is enhanced by mixing REMD[17] in FSATOOL. It contains five different replicas. The first replica is a biased replica. It is simulated by ABMD[16] with an adaptive bias potential on two collective variables: C_α-RMSD and number of native contacts. The contacts are defined by atom–atom pairs whose distances in space are smaller than 6.0 Å and intervals on the amino acid sequence are larger than 7. The flooding time to update the bias potential is 100 ps. The second replica is an equilibrium replica that receives the state variables from the biased replica at a fixed time step (20 ns). It is simulated by conventional molecular dynamics without the bias potential. And the last three replicas are normal (unbiased) replicas to do the sampling in the canonical ensemble at the same temperature. Their state variables are updated from the equilibrium replica at a time period of 20 ns with the Metropolis criterion.[62] The details of the mixing REMD method are described in our previous paper.[17]

The free energy landscapes of the three unbiased replicas are presented in Figure 2. They are calculated by the traditional histogram method. It shows that all the unbiased replicas sample widely in the

1096987x, 2023, 22, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/jcc.27132 by Huazhong University Of Sci & Tech, Wiley Online Library on [17/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 1 Comparison of different energy components on CPU and GPU between different programs (unit: kcal/mol).

	CPU			GPU		
Energy	AMBER[31]	OpenMM[32]	FSATOOL	AMBER[31]	OpenMM[32]	FSATOOL
Bond	4171.257	4171.258	4171.257	4171.257	4171.258	4171.256
Angle	116.336	116.337	116.336	116.336	116.337	116.337
Dihedral	218.602	218.602	218.602	218.602	218.602	218.602
Elec14	845.253	−21424.642	845.252	845.253	−21424.646	845.252
VDW14	60.134		60.133	60.134		60.133
Elec	−26918.737		−26918.805	−26918.729		−26918.851
VDW	4589.550		4589.550	4589.545		4589.548
Total	−16917.607	−16918.451	−16917.674	−16917.604	−16918.443	−16917.722

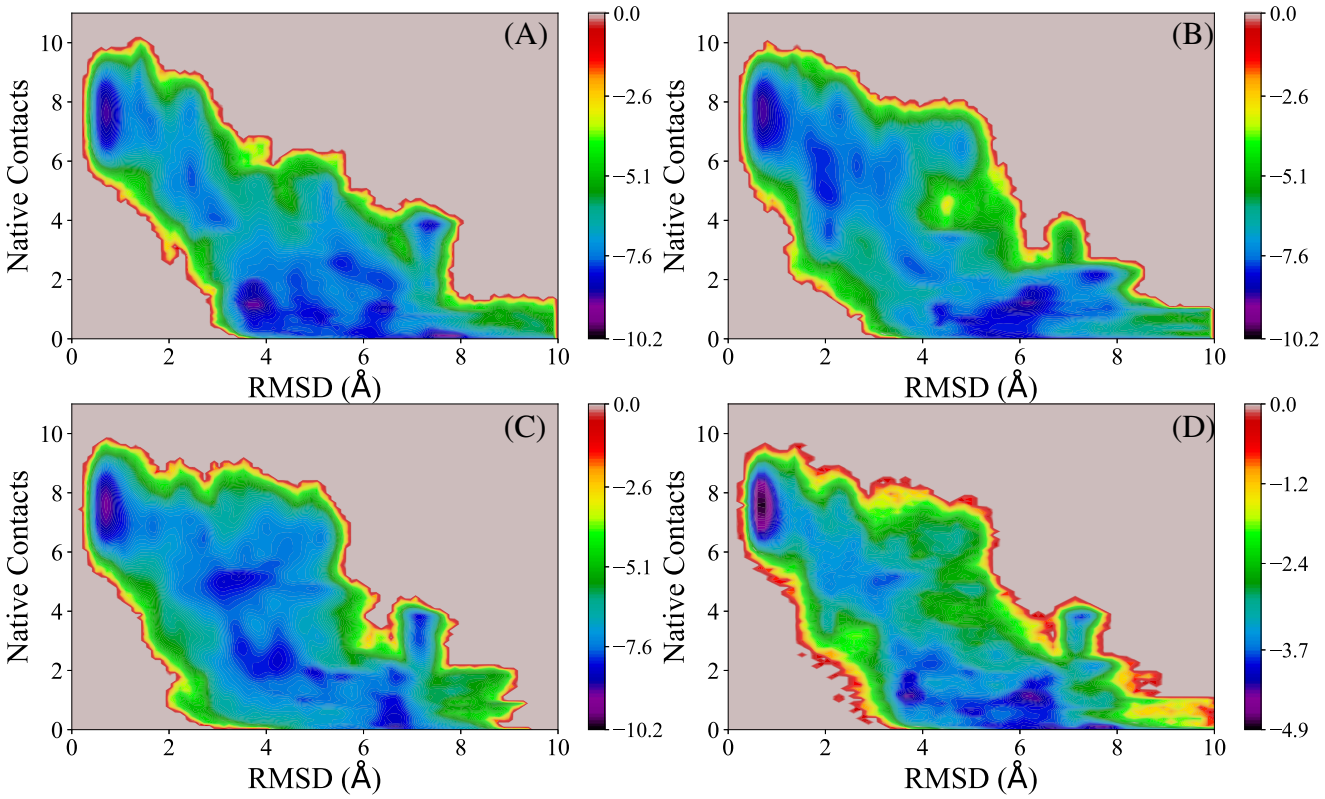

FIGURE 2 Free energy landscape of trpcage on its two collective variables: C_α-RMSD and number of native contacts. (A–C) are generated from the sampling data of three different unbiased replicas in a mixing REMD simulation at 300 K. (D) obtained from the combination of the sampling data from all the unbiased replicas.

conformational space from the folded state (at upper-left corner) to the unfolded state (at bottom). Mixing REMD helps all the unbiased replicas across the free energy barrier efficiently in the simulation.

Sometimes, predefined collective variables are not reliable enough to recover the transition process between important states. It might be necessary to do a dimension reduction on the sampling data in a high dimensional space. For this purpose, we implement four different dimension reduction methods in FSATOOL, including PCA,[23] TICA,[24,25] SRV,[28] and TAE.[29] The first two are traditional methods and the last two are deep learning methods developed in recent years. In this work, the initial high-dimensional space is spanned by 136 C_α-C_α atom distances in trpcage with an interval of at least four residues on the amino acid sequence. They are recorded by the unbiased replicas every 10 ps in the previous mixing REMD simulation. Totally there are 600,000 records in the trajectory file. By the four methods, all the sampling data are reduced from the 136-dimensional space to a two-dimensional reduced component space. TICA, SRV and TAE have the same lag time 10 ns. In the SRV training, the architecture of the neural network is [136, 50, 20]. That is, 136 input distances are first reduced to 20 outputs by the neural network and then they are

1096987x, 2023, 22, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/jcc.27132 by Huazhong University Of Sci & Tech, Wiley Online Library on [17/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

used for the search of the two main components.[28] In the TAE training, the network has a symmetric architecture [136, 50, 2, 50, 136]. The network inputs are 136 distances at time t and the outputs are the same distances at time $t + 10$ ns. TAE reduces the dimensionality of the input data by its two neurons in the bottleneck layer. Both SRV and TAE use a batch size of 100. The number of epochs is 100 and the activation function is Sigmoid. The learning rate is 0.0001 for SRV and 0.00001 for TAE. PCA, TICA, SRV, and TAE produce four different sets of components. The free energy landscapes on these components are presented in Figure 3.

As we discussed before, the generated components are the representative collective variables in a complicated conformational transition process. Accelerating the movement on these components may overcome the severe rare event sampling problem. Now, we perform four 2000 ns ABMD[16] enhanced sampling simulations for trpcage at 300 K. The initial conformation is shown in Figure 1. Each simulation has four identical biased replicas. And the bias potential is constructed on the two main components determined by the PCA, TICA, SRV, and TAE method individually. The flooding time is 50 ps. All the other simulation parameters are the same as before. After the simulation, all the sampling data are projected onto the same two collective variables as in Figure 2 (C_α-RMSD, number of native contacts). The distribution scope of the sampling data in the four accelerated simulations are presented in Figure 4. As shown in the figure, all methods except PCA are good enough to help trpcage approaches its native state at the upper-left corner (RMSD <1.0 Å).

Now, we compare the sampling efficiency between the enhanced sampling simulation and conventional molecular dynamics simulation. We first do a clustering analysis on all the sampling data of the three unbiased replicas of trpcage in a mixing REMD simulation at 300 K. As shown in Figure 2, the data distribute widely from the unfolded to the native state. We cluster the sampling data into 100 states by k-means method[18] in a two-dimensional space (C_α-RMSD vs. number of native contact). These 100 states are used as a reference dataset for the evaluation of the sampling efficiency of a simulation method.

In Figure 5, we present the change of the number of sampled states in the enhanced ABMD simulation (red dashed line), which was accelerated on two SRV components in Figure 3C. The same data of a conventional MD simulation at the same temperature are also provided (blue solid line). It shows that the enhanced simulation has a higher sampling speed than the conventional method. For example, it finds 80% of the states in the reference dataset within 400 ns and conventional MD simulation requires 1600 ns to reach the same percentage.

Finally, we present the speed test data of FSATOOL in the explicit solvent simulation on GPU (GeForce RTX 2080). We do a short 1 ns conventional molecular dynamics simulation for 10 typical

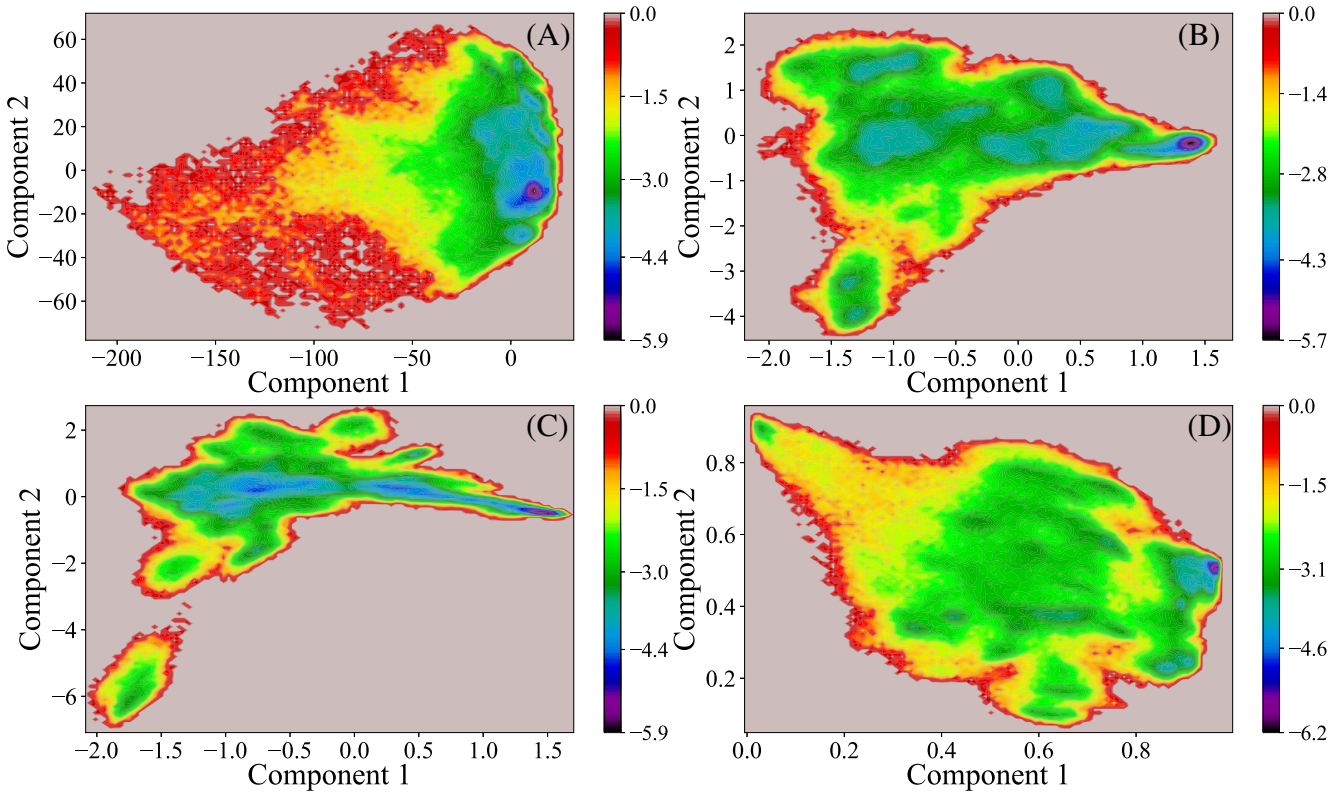

FIGURE 3 Free energy landscape of trpcage on the two components generated by four different dimension reduction methods. (A) PCA.[23] (B) TICA.[24,25] (C) SRV.[28] (D) TAE.[29] The data come from three unbiased replicas in a mixing REMD simulation at 300 K.

1096987x, 2023, 22, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/jcc.27132 by Huazhong University Of Sci & Tech, Wiley Online Library on [17/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

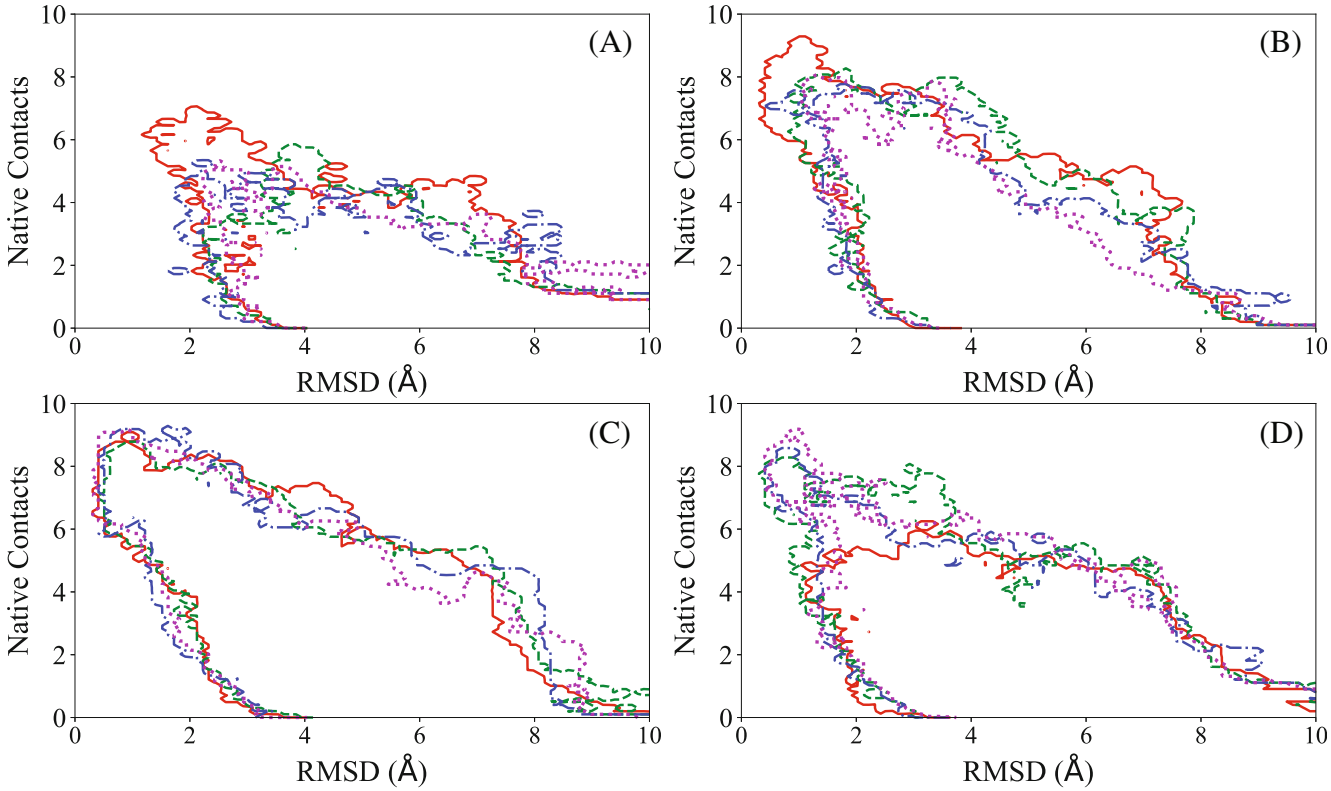

FIGURE 4 Sampling scope of trpcage in the four ABMD[16] enhanced sampling simulations at 300 K. The two collective variables in the figure are C_α-RMSD and number of native contacts. The borders of the sampling scope are defined by a very small histogram value ($1.0 \times 10^{-10} \times \exp(0.01/RT)$, R: gas constant, T: temperature) on a 100 × 100 grid system. The simulations are accelerated by a bias potential on the two main components that are determined by PCA[23] (A), TICA[24,25] (B), SRV[28] (C), and TAE[29] (D), respectively (presented in Figure 3).

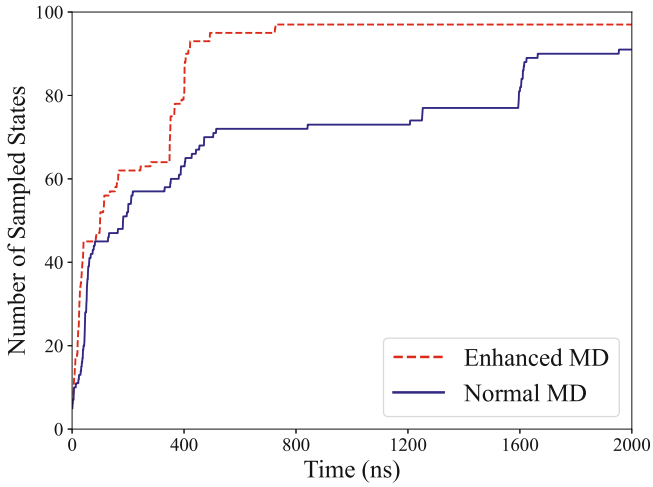

FIGURE 5 The change of the number of sampled states in the conventional MD simulation (blue solid line) and enhanced ABMD simulation (red dashed line). Both simulations are performed at 300 K with four replicas.

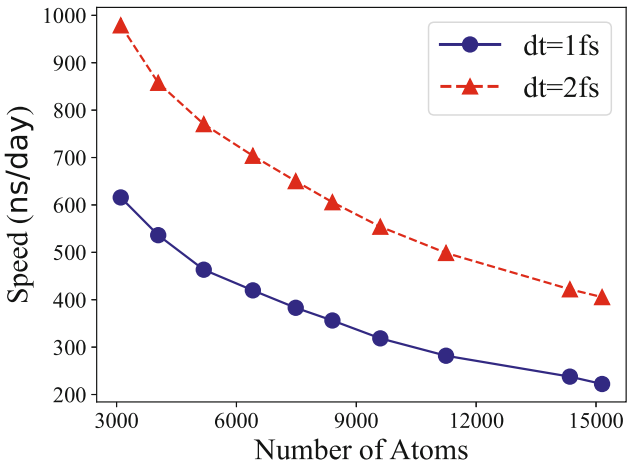

FIGURE 6 Speed test data for FSATOOL on GPU (GeForce RTX 2080). Simulation time step is 1.0 fs (blue solid line with circles) and 2.0 fs (red dashed line with triangles). The 10 test proteins are solvated into a periodic box of waters with different size.

proteins. All the protein structures are downloaded from the protein data bank[63] and embedded into TIP3P waters[59] in a periodic box by AmberTools.[31] The force field is FF14SBonlysc.[51] The PDB ID and atom number of the proteins (including waters) are given below: 1l2y (3104 atoms), 1vii (4042 atoms), 1fme (5182 atoms), 2wxc (6411 atoms), 1mjc (7484 atoms), 1prb (8402 atoms), 2a3d (9601 atoms), 2hba (11,246 atoms), 2f21 (14,346 atoms), and 1mi0 (15,152 atoms). Simulation time step size is 1.0 fs (no constraints) and 2.0 fs

1096987x, 2023, 22, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/jcc.27132 by Huazhong University Of Sci & Tech, Wiley Online Library on [17/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

(constraints on bonds involving hydrogens). Other parameters are the same as that in the previous simulation of trpcage. The final speed test data are shown in Figure 6. As to the performance comparisons between FSATOOL and other programs like OpenMM[32] and AMBER,[31] they are presented in the Supplementary Material (Section S3. Performance comparison).

4 | CONCLUSIONS

Explicit solvent simulation is widely used in the study of biomolecules. Due to lots of degrees of freedom in the system, the calculation quantity is expected to be very large. As a powerful simulation method, SPME[13] greatly accelerates the calculation by integrating the Ewald summation method[9,10] with the B-spline interpolation method.[36] In this paper, we introduce the basic idea of SPME and its implementation details in our published program FSATOOL.[14] Our codes on CPU and GPU are both written by Fortran language and compiled by the same compiler. Program installation is easy and fast. Moreover, the simulation data in explicit solvent can be directly processed by many existed modules in FSATOOL, including free energy calculation, structure clustering, transition path construction, dimension reduction and collective variable calculation.

To expand the application of FSATOOL, we also implement an important deep learning module in the program. The module is able to construct a fully connected neural network and optimize the weights by the standard forward and backward propagation. Based on the deep learning module, we further implement two trajectory analysis methods, SRV[28] and TAE.[29] Both of them are for the finding of the representative collective variables in the conformational transition process of biomolecules.

We know that a lot of molecular dynamics simulation packages have been published in the past. Among them, OpenMM[32] is a high-performance and high-extensible one. But many advanced analyses work like enhanced sampling, free energy calculation, Markov model analysis, has to depend on external plugins. Due to the additional work for data communication and file format conversion, developing or using these plugins are not very convenient to users. For example, to the best of our knowledge, there is no special tool in OpenMM for the enhanced sampling simulation on both SRV and TAE components. In comparison, FSATOOL integrates a lot of modules to do the related analyses work. It does not need to install external plugins. Users can complete the calculations by a few commands with required parameters. But we have to note that FSATOOL is not that extensible and efficient as OpenMM. In the future, we will continuously improve the performance of FSATOOL.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 32271290. Undergraduate student Yongzhuo Chen participates into the writing of the initial deep learning codes partially.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Changjun Chen https://orcid.org/0000-0002-6188-5223

REFERENCES

[1] J. K. Lai, J. Ambia, Y. Wang, P. Barth, *Structure* **2017**, *25*(11), 1758.
[2] A. Cumberworth, J. M. Bui, J. Gsponer, *J. Comput. Chem.* **2016**, *37*(7), 629.
[3] K. A. Dill, J. L. MacCallum, *Science* **2012**, *338*(6110), 1042.
[4] A. Jost Lopez, P. K. Quoika, M. Linke, G. Hummer, J. Köfinger, *J. Phys. Chem. B* **2020**, *124*(23), 4673.
[5] B. Roux, T. Simonson, *Biophys. Chem.* **1999**, *78*(1), 1.
[6] J. Mongan, C. Simmerling, J. A. McCammon, D. A. Case, A. Onufriev, *J. Chem. Theory Comput.* **2007**, *3*(1), 156.
[7] H. Tjong, H.-X. Zhou, *J. Phys. Chem. B* **2007**, *111*(11), 3055.
[8] S. L. Michael, R. S. J. Freddie, L. B. I. Charles, *J. Chem. Phys.* **2002**, *116*(24), 10606.
[9] P. P. Ewald, *Ann. Phys.* **1921**, *369*(3), 253.
[10] D. Williams, *Acta Crystallogr. A* **1971**, *27*(5), 452.
[11] R. W. Hockney, J. W. Eastwood, *Computer simulation using particles*, CRC, **1988**.
[12] T. Darden, D. York, L. Pedersen, *J. Chem. Phys.* **1993**, *98*(12), 10089.
[13] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, L. G. Pedersen, *J. Chem. Phys.* **1995**, *103*(19), 8577.
[14] Z. Shu, M. Wu, J. Liao, C. Chen, *J. Comput. Chem.* **2022**, *43*(3), 215.
[15] Y. Sugita, Y. Okamoto, *Chem. Phys. Lett.* **1999**, *314*(1), 141.
[16] V. Babin, C. Roland, C. Sagui, *J. Chem. Phys.* **2008**, *128*(13), 134101.
[17] H. Zhang, Q. Gong, H. Zhang, C. Chen, *J. Comput. Chem.* **2019**, *40*(20), 1806.
[18] J. A. Hartigan, M. A. Wong, *J Royal Stat Soc Series C (Appl Stat)* **1979**, *28*(1), 100.
[19] H.-S. Park, C.-H. Jun, *Expert Syst. Appl.* **2009**, *36*(2, Part 2), 3336.
[20] D. Shukla, C. X. Hernández, J. K. Weber, V. S. Pande, *Acc. Chem. Res.* **2015**, *48*(2), 414.
[21] F. Noé, C. Schütte, E. Vanden-Eijnden, L. Reich, T. R. Weikl, *Proc. Natl. Acad. Sci.* **2009**, *106*(45), 19011.
[22] J. D. Chodera, F. Noé, *Curr. Opin. Struct. Biol.* **2014**, *25*, 135.
[23] S. Wold, K. Esbensen, P. Geladi, *Chemom. Intell. Lab. Syst.* **1987**, *2*(1), 37.
[24] Y. Naritomi, S. Fuchigami, *J. Chem. Phys.* **2013**, *139*(21), 215102.
[25] G. Pérez-Hernández, F. Paul, T. Giorgino, G. De Fabritiis, F. Noé, *J. Chem. Phys.* **2013**, *139*(1), 015102.
[26] S. Kumar, J. M. Rosenberg, D. Bouzida, R. H. Swendsen, P. A. Kollman, *J. Comput. Chem.* **1992**, *13*(8), 1011.
[27] I.-C. Yeh, M. S. Lee, M. A. Olson, *J. Phys. Chem. B* **2008**, *112*(47), 15064.
[28] W. Chen, H. Sidky, A. L. Ferguson, *J. Chem. Phys.* **2019**, *150*(21), 214114.
[29] C. Wehmeyer, F. Noé, *J. Chem. Phys.* **2018**, *148*(24), 241703.
[30] B. Isralewitz, M. Gao, K. Schulten, *Curr. Opin. Struct. Biol.* **2001**, *11*(2), 224.
[31] D. A. Case, I. Y. Ben-Shalom, S. R. Brozell, D. S. Cerutti, T. E. Cheatham III, V. W. D. Cruzeiro, T. A. Darden, R. E. Duke, D. Ghoreishi, M. K. Gilson, H. Gohlke, A. W. Goetz, D. Greene, R. Harris, N. Homeyer, Y. Huang, S. Izadi, A. Kovalenko, T. Kurtzman, T. S. Lee, S. LeGrand, P. Li, C. Lin, J. Liu, T. Luchko, R. Luo, D. J. Mermelstein, K. M. Merz, Y. Miao, G. Monard, C. Nguyen, H. Nguyen, I. Omelyan, A. Onufriev, F. Pan, R. Qi, D. R. Roe, A. Roitberg, C. Sagui, S. Schott-Verdugo, J. Shen, C. L. Simmerling, J. Smith, R. SalomonFerrer, J. Swails, R. C. Walker, J. Wang, H. Wei, R. M. Wolf, X. Wu, L. Xiao,

1096987x, 2023, 22, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/jcc.27132 by Huazhong University Of Sci & Tech, Wiley Online Library on [17/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

D. M. York, P. A. Kollman, *AMBER*, University of California, San Francisco **2018**.
[32] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, V. S. Pande, *PLoS Comp Biol* **2017**, *13*(7), e1005659.
[33] M. P. Howard, J. A. Anderson, A. Nikoubashman, S. C. Glotzer, A. Z. Panagiotopoulos, *Comput. Phys. Commun.* **2016**, *203*, 45.
[34] Z. Yao, J.-S. Wang, G.-R. Liu, M. Cheng, *Comput. Phys. Commun.* **2004**, *161*(1), 27.
[35] C. Hastings, J. T. Wayward, J. P. Wong, *Approximations for Digital Computers*, Princeton University Press, **1955**.
[36] I. J. Schoenberg, *Cardinal Spline Interpolation*, Society for Industrial and Applied Mathematics, **1973**.
[37] H. J. C. Berendsen, J. P. M. Postma, W. F. V. Gunsteren, A. DiNola, J. R. Haak, *J. Chem. Phys.* **1984**, *81*(8), 3684.
[38] D. McKie, C. McKie, *Essentials of Crystallography*, Blackwell Science Inc, **1986**.
[39] R. Salomon-Ferrer, A. W. Götz, D. Poole, S. Le Grand, R. C. Walker, *J. Chem. Theory Comput.* **2013**, *9*(9), 3878.
[40] S. G. D. Gonzalo, S. Huang, J. Gómez-Luna, S. Hammond, O. Mutlu, W. Hwu. IEEE/ACM International Symposium on Code Generation and Optimization (CGO), February 16–20, 2019, pp. 73–84. **2019**.
[41] C. L. Wennberg, T. Murtola, B. Hess, E. Lindahl, *J. Chem. Theory Comput.* **2013**, *9*(8), 3527.
[42] Y. Yu, A. Kramer, R. M. Venable, B. R. Brooks, J. B. Klauda, R. W. Pastor, *J. Chem. Theory Comput.* **2021**, *17*(3), 1581.
[43] C. X. Hernandez, H. K. Wayment-Steele, M. M. Sultan, B. E. Husic, V. S. Pande, *Phys. Rev. E* **2018**, *97*(6–1), 062412.
[44] D. Bhowmik, S. Gao, M. T. Young, A. Ramanathan, *BMC Bioinform* **2018**, *19*(Suppl 18), 484.
[45] A. Brace, M. Salim, V. Subbiah, H. Ma, M. Emani, A. Trifa, A. R. Clyde, C. Adams, T. Uram, H. Yoo, A. Hock, J. Liu, V. Vishwanath, A. Ramanathan. Proc. Platform Adv. Sci. Comput. Conf. 2021. pp. 1–13.
[46] A. Mardt, L. Pasquali, H. Wu, F. Noé, *Nat. Commun.* **2018**, *9*(1), 5.
[47] J.-H. Prinz, H. Wu, M. Sarich, B. Keller, M. Senne, M. Held, J. D. Chodera, C. Schütte, F. Noé, *J. Chem. Phys.* **2011**, *134*(17), 174105.
[48] L. Bonati, G. Piccini, M. Parrinello, *Proc. Natl. Acad. Sci.* **2021**, *118*(44), e2113533118.
[49] M. Li, T. Zhang, Y. Chen, A. J. Smola, *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining; Association for Computing Machinery: New York*, USA, New York **2014**, p. 661.
[50] J. W. Neidigh, R. M. Fesinmeyer, N. H. Andersen, *Nat. Struct. Biol.* **2002**, *9*(6), 425.
[51] H. Nguyen, J. Maier, H. Huang, V. Perrone, C. Simmerling, *J. Am. Chem. Soc.* **2014**, *136*(40), 13959.
[52] H. Nguyen, D. R. Roe, C. Simmerling, *J. Chem. Theory Comput.* **2013**, *9*(4), 2020.
[53] P. Deift, X. Zhou, *Ann. Math.* **1993**, *137*(2), 295.
[54] M. R. Hestenes, E. Stiefel, *J. Res. Natl. Bur. Stand.* **1952**, *49*(6), 409.
[55] R. J. Loncharich, B. R. Brooks, R. W. Pastor, *Biopolymers* **1992**, *32*(5), 523.
[56] R. W. Pastor, B. R. Brooks, A. Szabo, *Mol. Phys.* **1988**, *65*(6), 1409.
[57] M. A. Cuendet, W. F. van Gunsteren, *J. Chem. Phys.* **2007**, *127*(18), 184102.
[58] J.-P. Ryckaert, G. Ciccotti, H. J. C. Berendsen, *J. Comput. Phys.* **1977**, *23*(3), 327.
[59] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, M. L. Klein, *J. Chem. Phys.* **1983**, *79*(2), 926.
[60] W. Humphrey, A. Dalke, K. Schulten, *J. Mol. Graph.* **1996**, *14*(1), 33.
[61] M. R. Shirts, C. Klein, J. M. Swails, J. Yin, M. K. Gilson, D. L. Mobley, D. A. Case, E. D. Zhong, *J. Comput. Aided Mol. Des.* **2017**, *31*(1), 147.
[62] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, *J. Chem. Phys.* **1953**, *21*(6), 1087.
[63] S. K. Burley, C. Bhikadiya, C. Bi, S. Bittrich, L. Chen, G. V. Crichlow, C. H. Christie, K. Dalenberg, L. Di Costanzo, J. M. Duarte, S. Dutta, Z. Feng, S. Ganesan, D. S. Goodsell, S. Ghosh, R. K. Green, V. Guranović, D. Guzenko, B. P. Hudson, C. L. Lawson, Y. Liang, R. Lowe, H. Namkoong, E. Peisach, I. Persikova, C. Randle, A. Rose, Y. Rose, A. Sali, J. Segura, M. Sekharan, C. Shao, Y.-P. Tao, M. Voigt, J. D. Westbrook, J. Y. Young, C. Zardecki, M. Zhuravleva, *Nucleic Acids Res.* **2021**, *49*(D1), D437.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: M. Wu, J. Liao, Z. Shu, C. Chen, *J. Comput. Chem.* **2023**, *44*(22), 1845. https://doi.org/10.1002/jcc.27132

1096987x, 2023, 22, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/jcc.27132 by Huazhong University Of Sci & Tech, Wiley Online Library on [17/07/2023]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License