Explicit-Solute Implicit-Solvent Molecular Simulation with Binary Level-Set, Adaptive-Mobility, and GPU

Shuang Liu*[†] Zirui Zhang*[‡] Hsiao-Bing Cheng*[§] Li-Tien Cheng*[¶] Bo Li*[‖]

October 11, 2022

Abstract

Coarse-grained modeling and efficient computer simulations are critical to the study of complex molecular processes with many degrees of freedom and multiple spatiotemporal scales. Variational implicit-solvent model (VISM) for biomolecular solvation is such a modeling framework, and its initial success has been demonstrated consistently. In VISM, an effective free-energy functional of solute-solvent interfaces is minimized, and the surface energy is a key component of the free energy. In this work, we extend VISM to include the solute mechanical interactions, and develop fast algorithms and GPU implementation for the extended variational explicit-solute implicit-solvent (VESIS) molecular simulations to determine the underlying molecular equilibrium conformations. We employ a fast binary level-set method for minimizing the solvation free energy of solute-solvent interfaces and construct an adaptive–mobility gradient descent method for solute atomic optimization. We also implement our methods on the integrated GPU. Numerical tests and applications to several molecular systems verify the accuracy, stability, and efficiency of our methods and algorithms. It is found that our new methods and GPU implementation improve the efficiency of the molecular simulation significantly over the CPU implementation. Our fast computational techniques may enable us to simulate very large systems such as protein-protein interactions and membrane dynamics for which explicit-solvent all-atom molecular dynamics simulations can be very expensive.

Keywords: Binary level-set method, GPU implementation, variational implicit-solvent explicit-solute model, Coulomb-field approximation, molecular mechanical interactions.

*All authors have contributed equally.

[†]Department of Mathematics, University of California, San Diego, 9500 Gilman Drive, La Jolla, California 92093-0112, United States. Email:shl083@ucsd.edu.

[‡]Department of Mathematics, University of California, San Diego, 9500 Gilman Drive, La Jolla, California 92093-0112, United States. Email:zzirui@ucsd.edu.

[§]Department of Mathematics, University of California, San Diego, 9500 Gilman Drive, La Jolla, California 92093-0112, United States. Email:hscheng@ucsd.edu.

[¶]Department of Mathematics, University of California, San Diego, 9500 Gilman Drive, La Jolla, California 92093-0112, United States. Corresponding author. Email:l3cheng@ucsd.edu.

[‖]Department of Mathematics, University of California, San Diego, 9500 Gilman Drive, La Jolla, California 92093-0112, United States. Email:bli@ucsd.edu.

1 Introduction

Computer simulations are basic tools in the study of complex biomolecular processes with multiple temporal and spatial scales and many-body interactions. Efficiency and computational costs, however, are bottlenecks in such simulations for large systems with long time scales of biological interest. Examples of such systems include protein-protein interactions, membrane dynamics, and aggregation of biopolymer networks. The development of coarse-grained biophysical and mathematical modeling, together with fast numerical algorithms and computer implementation, is therefore critical to the success of computational studies of complex biomolecular systems.

Implicit-solvent models are a class of coarse-grained models in which solvent is efficiently treated in comparison with explicit-solvent all-atom molecular dynamics simulations. In recent years, variational implicit-solvent model (VISM) has shown its initial success in efficient modeling of biomolecular conformations and recognition. VISM is a mesoscale description of the solvation of charged molecules, particularly biomolecules such as proteins, in an aqueous environment [6, 7]. The central quantity of such a model is a macroscopic free-energy functional of all possible solute-solvent interfaces each of which separates the solute molecules from the aqueous solvent (i.e., water or salted water). Minimizing such a functional leads to an equilibrium molecular conformation that is often metastable, and the corresponding minimum free energy. The free energy consists mainly of the solute-solvent interfacial energy, solute-solvent van der Waals (vdW) interaction energy, and the electrostatic interaction energy that can be described by a continuum electrostatics model. Implemented by the level-set method, a numerical method for interface motion, VISM is capable of capturing qualitatively or semi-quantitatively many key features of charged molecular processes, such as the dry and wet solvation states and the effect of electrostatic interactions, and providing reasonably good estimates of the solvation free energy [3, 11, 12, 25, 30, 33, 34]. We note that several other solvation models have been developed [1, 20, 24].

In this work, we extend VISM to include the flexibility of solute atoms, and develop fast algorithms and GPU implementation for the extended variational explicit-solute implicit solvent (VESIS) simulations of molecular conformational change and binding process. This study is motivated by our recent work that couples VISM with Monte Carlos (MC) method to simulate the binding of proteins p53 and MDM2 that are treated as rigid bodies, where each MC move is followed by a solvation free-energy calculation [32]. A new and fast binary level-set algorithm that we have developed enables us to carry out such intensive MC-VISM simulations with hundreds of thousands MC moves. Clearly, the rigid-body approximation can hardly make our MC-VISM simulations reach the final p53-MDM2 bound complex. However, explicit-solvent all-atom molecular dynamics (MD) simulations starting from our MC-VISM conformations reach quickly to the final complex. It is therefore naturally for us to further develop our mesoscale molecular simulation approach to allow the solute atoms to move around as in the real system. This is what we do in our current study.

Our main results include the following:

(1) We extend VISM to include the solute-solute atomic interactions with a usual force field to construct our VESIS model. Such interactions include the mechanical bonding,

bending, and torsion, vdW interactions modeled by Lennard-Jones (LJ) potentials, and the electrostatic interactions by Coulomb's law. The coupling between these solute interactions and the implicit solvent is through the solute-solvent interactions described by a sum of integrals over the solvent region, summing over all the solute atoms.

(2) We design an adaptive-mobility gradient descent optimization method to relax all the solute atoms, and couple it with our fast binary level-set method to minimize the VESIS free-energy functional.

(3) We implement our methods and algorithms on the integrated GPU, and test our code to verify its accuracy, stability, and efficiency.

(4) We apply our VESIS model and GPU implementation to simulate several molecular systems, including the protein BphC and the protein complex p53-MDM2, to demonstrate the significant improvement of efficiency of our new algorithms and implementation over the CPU implementation.

First introduced in [32], the binary level-set method is based on the approximation of surface area of an interface separating two regions by the convolution of the characteristic functions of these regions with a compactly supported kernel. This combines two steps, diffusion and threshold, in the method of threshold dynamics [23] (cf. also [8, 26–29]) into one step. An energy functional of the interface that includes the surface area and other related quantities can then be expressed as the sum over finite-difference gird cells. Cells in the two regions separated by the interface are marked by -1 and and $+1$. Equivalently, the interface is determined by a binary level-set function taking the value -1 or $+1$ on all the grid cells. The approximated total free-energy value can then be expressed as the sum of those values over all the grid cells. When a given interface is spatially perturbed, the energy change only occurs from those cells around the interface. The method then proceeds with flipping the cells (i.e., changing the sign of the binary level-set function on the cells) near the interface and only accept the change of sign when the energy is decreased. The algorithm is seemingly simple yet is significantly more efficient than the classical continuous level-set method [32]. A key factor contributing to such efficiency is that the flipping is done only locally around the interface instead of globally in the computational box [10, 18, 19].

Our new, adaptive-mobility gradient descent optimization method is designed to efficiently optimize a multi-variable objective function that may have many local minima and saddle points and that the gradient may vary significantly. The method is of the type of the gradient descent. But the descent is not uniform for all the iteration steps. Instead, mobility constants are adaptively changed during the iteration steps. This way, one may speed up the convergence.

In section 2, we describe our VESIS modeling framework. In section 3, we present our fast binary level-set method for interface motion and adaptive-mobility optimization method for relaxing atomic positions, as well as the simulation algorithm. Section 4 is devoted to the description of our GPU implementation. In section 5, we present the numerical tests and applications to several molecular systems, and demonstrate the efficiency of our methods and implementation. Finally, in section 6, we draw conclusions and discuss our future work. Appendix collects some calculations and formulas that are used in our modeling and numerical methods.

2 A Variational Explicit-Solute Implicit-Solvent Model

We consider a few molecules immersed in an aqueous solvent (i.e., water or salted water). This system of molecular solvation is confined spatially in a bounded region $\Omega \subset \mathbb{R}^3$; cf. Figure 1. We assume that there are N atoms of these solute molecules, located at $\mathbf{r}_1, \dots, \mathbf{r}_N \in \Omega$ and carrying partial charges $Q_1, \dots, Q_N$, respectively. A closed surface Γ inside Ω and enclosing all the solute atoms $\mathbf{r}_i$ $(1 \le i \le N)$ is called a solute-solvent interface or dielectric boundary. Such an interface, which may have several disjoint connected components, divides the entire solvation region Ω into two parts. One is the solute region, denoted $\Omega_{\rm m}$ (m stands for molecule), which is the interior of the surface Γ, and the other is the solvent region, denoted $\Omega_{\rm w}$ (w stands for water) and defined by $\Omega_{\rm w} = \Omega \setminus \overline{\Omega}_{\rm m}$ (a bar denotes the closure of a set).

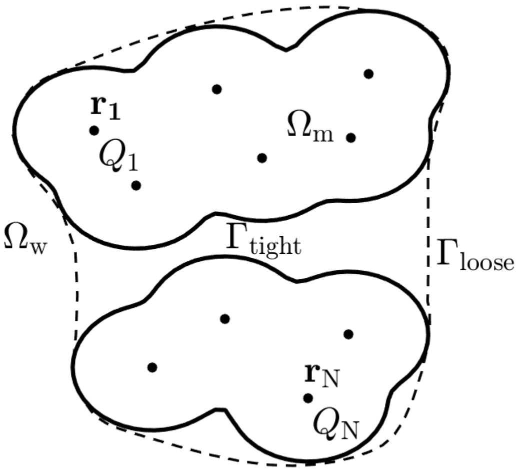

Figure 1: A schematic view of a solvation system with explicit solute and implicit solvent. The entire system region Ω is divided by a solute-solvent interface into the solvent region $\Omega_{\rm w}$ and the solute region $\Omega_{\rm m}$ containing all the solute atoms at $\mathbf{r}_i$ $(1 \le i \le N)$. Two different solute-solvent interfaces are shown. One is a tight interface $\Gamma_{\rm tight}$ (solid line) and the other a loose interface $\Gamma_{\rm loose}$ (dashed line).

Our basic assumption is that an experimentally observed equilibrium solvation system is determined by its solute-solvent interface and solute atomic positions that together minimize an effective free-energy functional [4]

$$G[\Gamma, \mathbf{R}] = G_{\rm VISM}[\Gamma, \mathbf{R}] + G_{\rm ss}[\mathbf{R}], \tag{2.1}$$

over all possible solute-solvent interfaces Γ and solute atomic positions $\mathbf{R} = (\mathbf{r}_1, \dots, \mathbf{r}_N)$. Here, the first part is the solvation free energy approximated by the VISM free energy and the second part is the solute-solute interaction potential or force field.

The VISM free energy is given by [6, 7, 30, 33]

$$\begin{aligned} G_{\rm VISM}[\Gamma, \mathbf{R}] = {} & \gamma_0 \text{Area}\,(\Gamma) + \rho_{\rm w} \sum_{i=1}^{N} \int_{\mathbb{R}^3 \setminus \Omega_{\rm m}} U_{\rm LJ}^{(i)}(|\mathbf{r} - \mathbf{r}_i|)\, dV_{\mathbf{r}} \\ & + \frac{1}{32\pi^2 \varepsilon_0} \left(\frac{1}{\varepsilon_{\rm w}} - \frac{1}{\varepsilon_{\rm m}} \right) \int_{\mathbb{R}^3 \setminus \Omega_{\rm m}} \left| \sum_{i=1}^{N} \frac{Q_i (\mathbf{r} - \mathbf{r}_i)}{|\mathbf{r} - \mathbf{r}_i|^3} \right|^2 dV_{\mathbf{r}}. \end{aligned} \tag{2.2}$$

The first term here is the solute-solvent interfacial energy, where γ_0 is the surface tension constant. The second term describes the van der Waals (vdW) type interactions between the solute atoms located at $\mathbf{r}_i$ ($1 \le i \le N$) and solvent molecules that are treated as a continuum, where $\rho_{\rm w}$ is the bulk solvent density and each $U_{\rm LJ}^{(i)}$ is a Lennard-Jones (LJ) potential of the form

$$U_{LJ}(r) = 4\varepsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right], \tag{2.3}$$

where the length parameter σ and energy parameter ε can depend on individual solute atoms. The last term in (2.2) is the Coulomb-field approximation (CFA) of the electrostatic interaction energy, where ε_0 is the vacuum permittivity, and $\varepsilon_{\rm w}$ and $\varepsilon_{\rm m}$ are the relative permittivities of the solvent and solute, respectively. Note that the integrals in (2.2) are over the region $\mathbb{R}^3 \setminus \Omega_{\rm m}$, instead of $\Omega_{\rm w}$ which is bounded. This is to account for the long-range effect of the vdW and Coulomb interactions.

We remark that one can include more terms in the VISM free energy. However, to keep our numerical implementation robust, we shall focus on this version of VISM free-energy functional.

The solute-solute interaction potential in (2.1) is given by

$$\begin{aligned} G_{\rm ss}[\mathbf{R}] = & \sum_{(i,j)} \frac{1}{2} A_{ij} (r_{ij} - r_{0ij})^2 + \sum_{(i,j,k)} \frac{1}{2} B_{ijk} (\theta_{ijk} - \theta_{0ijk})^2 \\ & + \sum_{(i,j,k,l)} \sum_{n=0}^{6} C_n [1 + \cos(n\tau_{i,j,k,l} - \psi_n)] + \sum_{(i,j)'} U_{\rm LJ}^{(i,j)}(r_{ij}) + \sum_{(i,j)'} \frac{Q_i Q_j}{4\pi\varepsilon_0 \varepsilon_{\rm w} r_{ij}}. \end{aligned} \tag{2.4}$$

Here, the first three terms account for the mechanical interaction energy from bonded solute atoms. The first term is the bonding energy of solute atoms, where the sum is taken over all pairs (i, j) of bounded solute atoms, $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$, and r_{0ij} and A_{ij} are the corresponding equilibrium distance and spring constant, respectively. The second term in (2.4) is the bending energy of solute atoms, where the sum is taken over all triplets (i, j, k) such that both pairs of solute atoms $(\mathbf{r}_i, \mathbf{r}_j)$ and $(\mathbf{r}_j, \mathbf{r}_k)$ are bonded. For such a triplet, θ_{ijk} is the angle between the vectors $\mathbf{r}_i - \mathbf{r}_j$ and $\mathbf{r}_k - \mathbf{r}_j$, $\theta_{0ijk} \in [0, \pi]$ is the corresponding equilibrium angle, and B_{ijk} is a constant parameter. The third term in (2.4) accounts for the torsion energy of solute atoms [15]. The sum is taken over all quadruples (i, j, k, l) such that $(\mathbf{r}_i, \mathbf{r}_j)$, $(\mathbf{r}_j, \mathbf{r}_k)$, and $(\mathbf{r}_k, \mathbf{r}_l)$ are all bonded. For such a quadruple (i, j, k, l), τ_{ijkl} is the torsion angle that is the angle between the plane determined by $(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k)$ and that determined by $(\mathbf{r}_j, \mathbf{r}_k, \mathbf{r}_l)$, n is the multiplicity, ψ_n is the phase factor, and all C_n are constants.

The last two terms in (2.4) account for the interaction energies from non-bonded solute atoms indicated by $(i, j)'$ in the summation. The fourth term is the solute-solute vdW interaction energy, where each $U_{\rm LJ}^{(i,j)}$ is an LJ potential of the form (2.3). The last term is the solute-solute Coulomb interaction energy.

3 Numerical Methods

We minimize the free-energy functional $G[\Gamma, \mathbf{R}]$ defined in (2.1) numerically by an iteration scheme. Each iteration step consists of two parts. In the first part, we fix the solute atomic positions and minimize numerically the VISM solvation free-energy functional (2.2) by a binary level-set method to obtain an optimal solute-solvent interface. In the second part, we fix the interface obtained in the first part, and minimize the energy functional $G[\Gamma, \mathbf{R}]$ that is a multi-variable function of the solute atomic positions $\mathbf{R} = (\mathbf{r}_1, \dots, \mathbf{r}_N)$ by an adaptive-mobility gradient descent method. The binary level-set method was introduced and used in our rigid-body MC-VISM simulations of protein binding [32]. Here, we briefly recall the method, referring to [32] for more details. We also describe in details our new, adaptive-mobility gradient descent optimization method for minimizing the function $G[\Gamma, \mathbf{R}]$ with Γ fixed. We present our step-by-step algorithm at the end of this section.

3.1 A Binary Level-Set Method

We set the solvation system region to be $\Omega = (-L, L)^3$ for some $L > 0$. The side length L is chosen to be large enough so that the region Ω includes all the solute atoms $\mathbf{r}_i$ $(1 \le i \le N)$ whose geometrical center can be shifted to the origin, if necessary; cf. Figure 1. This region Ω is also our computational box. We cover it by a uniform finite-difference grid of size h. A solute-solvent interface $\Gamma = \partial\Omega_{\rm m}$ is approximated by a binary level-set function ϕ that is defined on all the grid cells with $\phi = -1$ and $\phi = +1$ on cells interior and exterior to Γ, respectively.

We discretize the VISM solvation free-energy functional (2.2) with all the solute atoms fixed at $\mathbf{r}_i$ $(1 \le i \le N)$. Let us first rewrite this functional as

$$G_{\rm VISM}[\Gamma, \mathbf{R}] = \gamma_0 \text{Area}\,(\Gamma) + \int_{\Omega\setminus\Omega_{\rm m}} U(\mathbf{r})\, dV_{\mathbf{r}} + \int_{\mathbb{R}^3\setminus\Omega} U(\mathbf{r})\, dV_{\mathbf{r}}, \tag{3.1}$$

where

$$U(\mathbf{r}) = \rho_{\rm w} \sum_{i=1}^{N} U_{\rm LJ}^{(i)}(|\mathbf{r} - \mathbf{r}_i|) + \frac{1}{32\pi^2\varepsilon_0}\left(\frac{1}{\varepsilon_{\rm w}} - \frac{1}{\varepsilon_{\rm m}}\right)\left|\sum_{i=1}^{N} \frac{Q_i(\mathbf{r} - \mathbf{r}_i)}{|\mathbf{r} - \mathbf{r}_i|^3}\right|^2. \tag{3.2}$$

Approximation of the surface energy. The surface area of the solute-solvent interface Γ can be expressed as [32]

$$\text{Area}(\Gamma) = \frac{C_0}{\delta^4}\int_{\mathbf{x}\in\Omega_{\rm m}}\int_{\mathbf{y}\in\Omega_{\rm w}} K\left(\frac{\mathbf{x} - \mathbf{y}}{\delta}\right) d\mathbf{y} d\mathbf{x} + O(\delta^2) \quad for \quad 0 < \delta \ll 1, \tag{3.3}$$

where

$$C_0 = \left(\int_0^1 \int_{B(\mathbf{0},1)\cap\{y_3 > s\}} K(\mathbf{y})\, d\mathbf{y}\, ds\right)^{-1}$$

is a constant, $B(\mathbf{0}, A)$ for any $A > 0$ is the ball centered at the origin $\mathbf{0}$ with radius A, and y_3 is the third component of the position vector $\mathbf{y}$. The kernel function $K = K(\mathbf{x})$ $(\mathbf{x} \in \mathbb{R}^3)$

is chosen to be non-negative, compactly supported in the closure of the unit ball $B_1(\mathbf{0})$ of $\mathbb{R}^3$, and spherically symmetric (i.e., it is a function of $|\mathbf{x}|$). In our implementation, we set $K(\mathbf{x}) = \sin^2(\pi|\mathbf{x}|)$ if $|\mathbf{x}| \le 1$ and 0 elsewhere. The small parameter $\delta > 0$ is the rescaled kernel radius, defined as the radius of the ball of the support of $K(|\mathbf{x}|/\delta)$.

Discretization of the VISM free energy in the solvation region. By employing the center-point numerical integration rule, one can discretize the double-integral in (3.3) with an optimal choice $\delta = \lambda\sqrt{h}$, where $\lambda > 0$ is a constant. Consequently, we obtain the following expression of an approximation of the surface energy, the first term in (3.1) [32]:

$$\gamma_0 \text{Area}\,(\Gamma) = \frac{\gamma_0 C_0 h^4}{\lambda^4} \sum_{\mathbf{x}_j \in \Omega_\text{m}} \sum_{\substack{\mathbf{x}_k \in \Omega_\text{w} \\ |\mathbf{x}_k - \mathbf{x}_j| \le \lambda\sqrt{h}}} K(\mathbf{x}_j - \mathbf{x}_k) + O(h),$$

where $\mathbf{x}_j \in \Omega_\text{m}$ and $\mathbf{x}_k \in \Omega_\text{w}$ are the centers of grid cells in Ω_m and Ω_w, respectively. The second term in (3.1) can be approximated by the center-point integration rule:

$$\int_{\Omega\setminus\Omega_\text{m}} U(\mathbf{r})\, dV_\mathbf{r} = h^3 \sum_{\mathbf{x}_j \in \Omega_\text{w}} U(\mathbf{x}_j) + O(h).$$

Therefore, these approximations and (3.1) lead to the approximation

$$\begin{aligned} G_\text{VISM}[\Gamma, \mathbf{R}] \approx{}& \frac{\gamma_0 C_0 h^4}{\lambda^4} \sum_{\mathbf{x}_j \in \Omega_\text{m}} \sum_{\substack{\mathbf{x}_k \in \Omega_\text{w} \\ |\mathbf{x}_k - \mathbf{x}_j| \le \lambda\sqrt{h}}} K(\mathbf{x}_j - \mathbf{x}_k) \\ &+ h^3 \sum_{\mathbf{x}_j \in \Omega_\text{w}} U(\mathbf{x}_j) + \int_{\mathbb{R}^3\setminus\Omega} U(\mathbf{r})\, dV_\mathbf{r}. \end{aligned} \tag{3.4}$$

The last integral can be written analytically as iterated integrals using the spherical coordinates and evaluated by one-dimensional numerical quadrature; cf. [5].

Flipping grid cells to decrease the energy. Given a solute-solvent interface defined by a binary level-set function, we relax its VISM free energy (3.4) by flipping the grid cells, i.e., by changing the sign of the binary level-set function on the grid cells. The flipping is only done for grid cells that around the interface. This is because that any grid cells centered at $\mathbf{x}_j \in \Omega_\text{m}$ and $\mathbf{x}_k \in \Omega_\text{w}$ with $|\mathbf{x}_j - \mathbf{x}_k| > \lambda\sqrt{h}$ do not contribute to the first term in (3.4).

We pick up a grid cell that is immediate next to be the interface, flip its sign, and calculate the change of the approximate energy based on (3.4). Note that the last term in (3.4) does not change if we flip a grid cell. If the cell is centered at $\mathbf{x}_j \in \Omega_\text{m}$, so the sign of the cell is -1, then the flip of the cell leads to the change of energy

$$\Delta(G_\text{solv})_j = \frac{\gamma_0 C_0 h^4}{\lambda^4} \sum_{\substack{\mathbf{x}_k \in \Omega_\text{m} \\ |\mathbf{x}_k - \mathbf{x}_j| \le \lambda\sqrt{h}}} K(\mathbf{x}_j - \mathbf{x}_k) - \frac{\gamma_0 C_0 h^4}{\lambda^4} \sum_{\substack{\mathbf{x}_k \in \Omega_\text{w} \\ |\mathbf{x}_k - \mathbf{x}_j| \le \lambda\sqrt{h}}} K(\mathbf{x}_j - \mathbf{x}_k) + U(\mathbf{x_j}). \tag{3.5}$$

Otherwise, if the cell is centered at $\mathbf{x}_j \in \Omega_{\mathrm{w}}$, so the sign of the cell is $+1$, then the flip of the cell leads to the change of energy

$$\Delta(G_{\mathrm{solv}})_j = \frac{\gamma_0 C_0 h^4}{\lambda^4} \sum_{\substack{\mathbf{x}_k \in \Omega_{\mathrm{w}} \\ |\mathbf{x}_k - \mathbf{x}_j| \le \lambda\sqrt{h}}} K(\mathbf{x}_j - \mathbf{x}_k) - \frac{\gamma_0 C_0 h^4}{\lambda^4} \sum_{\substack{\mathbf{x}_k \in \Omega_{\mathrm{m}} \\ |\mathbf{x}_k - \mathbf{x}_j| \le \lambda\sqrt{h}}} K(\mathbf{x}_j - \mathbf{x}_k) - U(\mathbf{x_j}). \quad (3.6)$$

After calculating $\Delta(G_{\mathrm{solv}})_j$ by flipping grid cells near the interface, we put $\Delta(G_{\mathrm{solv}})_j$ in a Min-Heap. We flip the grid cell with the smallest $\Delta(G_{\mathrm{solv}})_j$ in the heap if $\Delta(G_{\mathrm{solv}})_j < 0$. With the new interface, we add energy changes for new grid cells near the new interface to the heap, delete old grid cells in the heap which are not near the new interface, update $\Delta(G_{\mathrm{solv}})_j$ for old grid cells near the new interface in the heap, then we sort the Min-Heap. This flipping process stops until all $\Delta(G_{\mathrm{solv}})_j \ge 0$, indicating the energy reaches a minimum.

Initial surfaces. Our method for minimizing the VISM free-energy functional is of steepest descent type. It starts with an initial surface and iteratively moves it with the free energy decreased in each step of the iteration. The final free-energy minimizing surface is a local minimizer of the functional and depends on the initial surface. Different initial surfaces can lead to different (meta)stable equilibrium conformations that are of interest; see Figure 1. In order to capture multiple local minimizers, we often use two types of initial surfaces. One is a loose initial surface that can be a large sphere enclosing all the solute atoms. The other is a tight initial surface that wraps up all the solute atoms tightly with vdW radii. Such a surface is the zero level-set of the continuous function

$$\varphi(\mathbf{r}) = \min_{1 \le i \le N} (|\mathbf{r} - \mathbf{r}_i| - d_i),$$

where $d_i > 0$ is the vdW radius of the ith solute atom located at $\mathbf{r}_i$ $(i = 1, \cdots, N)$. The binary level-set function for the surface can then be obtained by setting its value at the center of a grid cell to be the sign of φ-value at that center. Figure 2 shows a tight surface constructed by both a continuous and the corresponding binary level-set function.

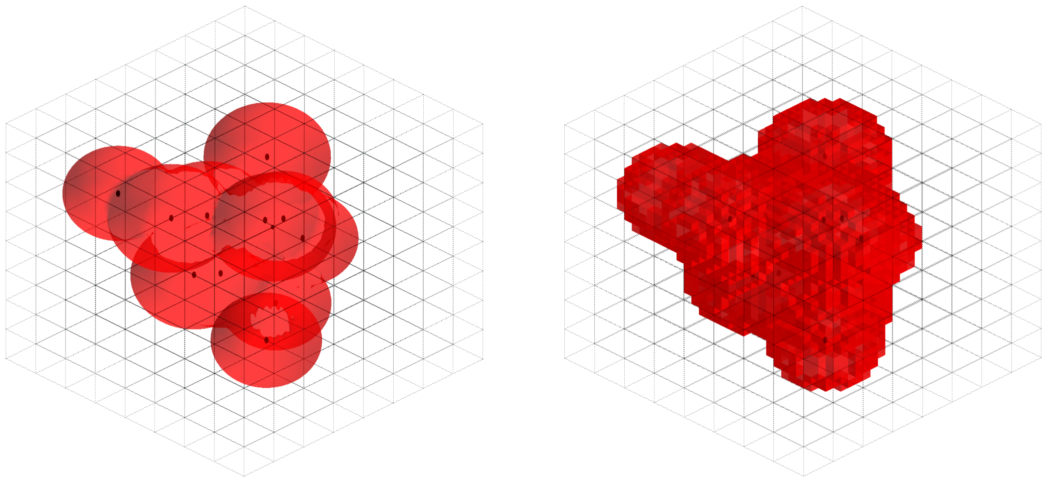
Figure 2: A tight initial solute-solvent interface constructed as the union of vdW spheres centered at solute atoms (black dots) by a continuous level-set function (Left) and a binary level-set function (Right), respectively.

3.2 Adaptive-Mobility Gradient Descent Method for the Relaxation of Solute Atoms

With a fixed solute-solvent interface Γ, we minimize the free energy $G[\Gamma, \mathbf{R}]$ defined in (2.1), (2.2), and (2.4) as a function of $\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_N)$ by solving for a steady-state solution to the system of the gradient descent equations

$$\frac{d(\mathbf{r}_n(t))_l}{dt} = -M_{nl}(\nabla_{\mathbf{r}_n} G[\Gamma, \mathbf{R}])_l, \quad n = 1, \cdots, N, \quad l = 1, 2, 3, \tag{3.7}$$

where $(\mathbf{a})_l$ denotes the lth component of a vector $\mathbf{a} \in \mathbb{R}^{\mathbf{3}}$ $(1 \leq l \leq 3)$ and all $M_{nl} > 0$ are constants (called mobility constants). The formula of the gradient $\nabla_{\mathbf{r}_n} G[\Gamma, \mathbf{R}]$ is given in (A.1) in Appendix. We use the forward Euler method to solve these equations iteratively with a fixed time step.

Due to the complex molecular interactions of an underlying system, the gradient of $G[\Gamma, \mathbf{R}]$ can vary significantly with solute atoms and with different components. If the mobility constant M_{nl} is too large, then the motion of the particle $\mathbf{r}_n$ in its gradient descent direction may possibly overshoot and increase the free energy. If M_{nl} is too small, the free energy may decrease very slowly. To improve the stability and efficiency, we therefore adaptively change the mobility constants M_{nl} in each step of iteration based on the magnitude of gradient and the total free-energy value.

In our implementation, the mobility constants are chosen to be ωM, where the "base" mobility ω is updated in each interaction step of the forward Euler method and M is also adjustable. The adjustment of M is based on the decrease or increase of the energy with a high energy threshold. It is controlled by a relative energy value $\delta G^* > 0$ depending on iteration steps, and a shrinking parameter $\alpha \in (0, 1)$ that shrinks M if the energy increases too much and too often tracked by a counting number N_{cnt} which has a threshold value N_{cnt}^*. Initially, we set $M = 1$ and $N_{\text{cnt}} = 0$.

Given all the atomic positions after a forward Euler iteration step, with some $M > 0$ and $0 \leq N_{\text{cnt}} < N_{\text{cnt}}^*$, and the corresponding energy $G[\Gamma, \mathbf{R}]$ calculated and denoted G_{old}. We calculate all the gradient of $G[\Gamma, \mathbf{R}]$ at those positions and set

$$\omega = \frac{1}{\max_{1 \leq m \leq N, 1 \leq k \leq 3} |(\nabla_{\mathbf{r}_m}(G[\Gamma, \mathbf{R}])_k|}. \tag{3.8}$$

We then move the atomic positions in one time step by the forward Euler iteration for solving (3.7) with all $M_{nl} = \omega M$. We also calculate the energy for the new atomic positions and denote it by G_{new}. If $G_{\text{new}} < G_{\text{old}}$, then we accept the moved solute atomic positions, do not change the value of M, increase N_{cnt} by 1, and continue the iteration. If $G_{\text{new}} > G_{\text{old}}$, then we choose the threshold δG^* to be some fraction of G_{old} and consider two cases. If $G_{\text{new}} \geq G_{\text{old}} + \delta G^*$, then we do not update the atomic positions but shrink M to $M := \alpha M$ reset $N_{\text{cnt}} = 0$ and start over with the next iteration step. If $G_{\text{new}} < G_{\text{old}} + \delta G^*$, then we check the counting number N_{cnt} and consider two cases:

(1) If $N_{\text{cnt}} \geq N_{\text{cnt}}^*$, which means that the same M has been used for N_{cnt}^* times, then we do not accept the new positions, shrink M to $M := \alpha M$, and reset $N_{\text{cnt}} = 0$.

(2) If $N_{\text{cnt}} < N^*_{\text{cnt}}$, we accept the new atomic positions, keep the same M, and increase N_{cnt} by 1.

Note that N_{cnt} is the number of steps where the same M is used consecutively. In our implementation, we choose $\alpha = 1/\sqrt{2}$, $N^*_{\text{cnt}} = 5$, and δG^* to be 5% of G_{old}.

3.3 Numerical Algorithm

```
Step 1. Initialization. Input all the model parameters from (2.2)–(2.4). In particular, position
    $N$ solute atoms with the center of geometry at the origin by a coordinate translation
    if necessary. Set the computational box $\Omega = (-L, L)^3$ and discretize the box with a
    uniform finite-difference grid. Set an initial binary level-set function $\phi^{(0)}$ to define the
    initial solute-solvent interface $\Gamma^0$, solute region $\Omega^0_{\text{m}}$, and solvent region $\Omega^0_{\text{w}}$.
    Set counter $N_{\text{cnt}} = 0$, count tolerance $N^*_{\text{cnt}} = 5$, an initial uniform mobility constant
    $M = 1$, and the time step $dt = 1$. Set the initial iteration number $k = 0$ and $k_{\max}=$
    3000. Set the shrinking parameter $\alpha = 1/\sqrt{2}$. Set the error tolerance $\text{Tol}_1 = 1e\text{-}4$ for
    the gradient and $\text{Tol}_2 = 1e\text{-}5$ for atomic positions update.
Step 2. Get the optimal solute-solvent interface $\Gamma^{k+1}$ with atomic positions $\mathbf{R}^k$ by the binary
    level-set method.
    Step 2.1. Calculate by (3.5) and (3.6) the change of solvation free energy $\Delta(G_{\text{solv}})_j$ on
        all grid cells next to the interface $\Gamma^k$. Sort solvation energy change $\Delta(G_{\text{solv}})_j$ in a
        Min-Heap.
    Step 2.2. Flipping Process:
        while (Smallest $\Delta(G_{\text{solv}})_j < 0$) do
            Flip : flip the corresponding grid cell with smallest $\Delta(G_{\text{solv}})_j$.
            Update : check for new grid cells next to the new interface, calculate $\Delta(G_{\text{solv}})_j$
                for new grid cells, and update $\Delta(G_{\text{solv}})_j$ for old grid cells in the heap.
            Sort : sort solvation energy change $\Delta(G_{\text{solv}})_j$ in a Min-Heap.
        endwhile
    Step 2.3. Define $\Gamma^{k+1}$ to be the optimal solute-solvent interface from the flipping process.
Step 3. Update the solute atomic positions by solving the system of equations (3.7).
    Step 3.1. Calculate by (A.1) the gradient $\nabla_{\mathbf{r}_n} G[\Gamma^{k+1}, \mathbf{R}^k]$ for all $n = 1, \cdots, N$.
    Step 3.2. Test the convergence. If absolute values of $\nabla_{\mathbf{r}_n} G[\Gamma^{k+1}, \mathbf{R}^k]$ for each solute
        atom in each coordinate $< \text{Tol}_1$, then stop the algorithm.
    Step 3.3. Update positions of all moving solute atoms according to equation (3.7) and
        (3.8). Calculate the total free energy change $\delta G[\Gamma^{k+1}, \mathbf{R}^{k+1}] = G[\Gamma^{k+1}, \mathbf{R}^{k+1}] -$
        $G[\Gamma^{k+1}, \mathbf{R}^k]$. Set $\delta G^* = 5\% G[\Gamma^{k+1}, \mathbf{R}^k]$.
    Step 3.4. Check the total free energy change:
        if $(\delta G[\Gamma^{k+1}, \mathbf{R}^{k+1}] > \delta G^*)$ or ( $N_{\text{cnt}} \geq N^*_{\text{cnt}}$ and $0 < \delta G[\Gamma^{k+1}, \mathbf{R}^{k+1}] \leq \delta G^*$)
        Put all moving solute atoms back to $\mathbf{R}^k$.
        $M = \alpha M$, $N_{\text{cnt}} = 0$, go to Step 3.3.
        else
        $N_{\text{cnt}} = N_{\text{cnt}} + 1$;
        endif
```

Step 4. Calculate the absolute error $\Delta G[\Gamma^{k+1}, \mathbf{R}^{k+1}]_{\text{abs}}$ and relative error $\Delta G[\Gamma^{k+1}, \mathbf{R}^{k+1}]_{\text{rel}}$.

Step 5. Test the convergence. If either $\Delta G[\Gamma^{k+1}, \mathbf{R}^{k+1}]_{\text{rel}} < \text{Tol}_2$ or $\Delta G[\Gamma^{k+1}, \mathbf{R}^{k+1}]_{\text{abs}} < \text{Tol}_2$ stays continuously for 100 steps, or if the number of iterations reaches to $k_{\max}$, stop the algorithm. Otherwise, go to Step 2.

We remark that there are two error tolerances and stopping criteria: One is a small tolerance 1e-4 for the gradient descent for updating the solute atomic positions. The other stop criterion is a small tolerance of relative difference or absolute difference of the total free energy. In our experiments, we set the relative difference stop criterion to be 1e-5, and absolute difference stop criteria to be 1e-5. To avoid the situation that the free energy functional decreases slowly because of small mobility factor M_{nl}, we determine that the system stops only when the relative difference stop criterion or the absolute difference stop criterion is satisfied continuously for 100 steps.

4 GPU Implementation

In this section, we discuss the parallel implementation of aspects of our free-energy functional minimization algorithm for the fast execution of our programs.

Parallel computing concerns strategies for performing simultaneous computations, usually through the use of multiple processors. This approach has become more and more important as the abilities of individual processors reach their limits under Moore's law. Integrated GPUs are nearly ubiquitous in today's client devices such as laptops and desktops. Much like dedicated GPUs, the integrated GPUs are also capable of general–purpose computation in addition to the traditional graphics role. Furthermore, modern integrated GPUs work the same way as those dedicated cards with the exception that they use system memory that is shared with the CPU [9]. The recent advent of the use of the graphics processing unit (GPU) for general purpose parallel computing, instead of traditionally multiple central processing units (CPU's), has allowed for algorithms that can take advantage of its high throughput and hundreds or thousands of cores to achieve new heights in speed. This has, for example, revolutionized the subject of deep learning in artificial intelligence.

We introduce parallel programming, using OpenCL, employing the integrated GPU for operations in our free-energy functional minimization algorithm. The operations that are particularly amenable to this kind of parallelization are usually made up of a large number of smaller, simpler ones, to take advantage of the large number of cores in the GPU that, alternatively, must work in lockstep. We find such operations in our computations of the LJ and CFA of the electrostatics in the VISM free energy (2.2), the solute-solute interaction energy (2.4), and all of their derivatives, combined in (A.1). We separate the parallelization into two cases that are treated differently, one handling VISM LJ and electrostatic terms, and their derivatives, and the other handling solute-solute interaction terms and their derivatives.

For the first case, we begin by describing the procedure introduced in [32] for the LJ and CFA contributions, though in more general terms. These contributions notably both contain integrals of the form

$$\int_{\mathbb{R}^3 \setminus \Omega_{\text{m}}} f(\mathbf{r})\, dV_{\mathbf{r}},$$

where f has some complexity in the computation of its values; in the case of LJ, $f(\mathbf{r}) = \sum_{i=1}^{N} U_{LJ}(|\mathbf{r} - \mathbf{r}_i|)$, which requires some computation when the number of solute atoms is large. With far-field approximations handling the integral outside the computational box Ω, numerical quadrature for the rest takes the general form

$$\sum_{k \in I} \alpha_k f(\mathbf{r}_k) \Delta \mathbf{x}_k,$$

where $\mathbf{x}_k$ are grid points of a grid in Ω, and for some $\alpha_k \in \mathbb{R}$. This summation is computationally intensive as f needs to be evaluated over the grid; in fact, in our problem this needs to be performed each time step, when the atoms move. A GPU parallel implementation is introduced in [32] to handle the evaluation of f over the grid which parallelizes over the grid points, passing out the computation of f at each to the cores. This works especially well because there of the large number of grid points, typically hundreds of thousands or millions, the GPU cores can work on. Note, in the parlance of OpenCL, the grid points form the work-items, which are instead known as threads under CUDA.

We apply this idea here not only to LJ and electrostatic terms, but also to their derivative terms found in the gradient of the VISM free-energy (A.1). These terms also have integrands that grow in complexity with the number of solute atoms, thus slowing down computations in the case of large numbers of moving atoms. Thus, the same parallelization techniques can be adopted to improve runtimes.

For the solute-solute interaction terms and their derivatives, no integration is present and no grid points are involved. Instead, all terms involve a summation of some interaction between solute atoms. Consider, for example, a system's solute-solute LJ interactions:

$$\sum_{1 \le i \le N} \sum_{j \neq i} U_{LJ}(|\mathbf{r}_i - \mathbf{r}_j|).$$

For our parallel implementation, we consider separately the terms

$$\sum_{j \neq i} U_{LJ}(|\mathbf{r}_i - \mathbf{r}_j|),$$

and parallelize by passing out these computations for each $1 \le i \le N$ to the cores. Note, however, there are far fewer solute atoms, typically in the hundreds or thousands, compared to the hundreds of thousands or millions of grid points. Thus, the parallelization may not be as efficient in comparison with that of our first case.

One additional note is that while double-precision machine numbers and their arithmetic are commonly available in CPU architectures, they are not universally supported on GPU's, where, for traditional graphical purposes, single-precision has been adequate. And though more and more GPU architectures now do support double-precision, due to the expansion of GPU's for general purpose computing, single-precision and even half-precision arithmetic operations are still used for faster calculations. The drawback in the use of single-precision instead of double-precision is in increased round-off error. This especially is of concern when performing a large number of operations, where round-off errors can accumulate to intolerable levels. In our application, we find such large numbers of operations in our sums, with sums over solute atoms, which can be in the thousands, and over grid points, which can be in the millions. For our sums, we adopt the strategy of summing-by-pairs [31], a binary tree-based approach to order the operations in such a way as to reduce round-off error. In our applications, we find this to result in a nearly negligible amount of error compared to double-precision results, allowing us to take advantage of the speed afforded by single-precision computations. In future work, we may consider computing with mixed-precision machine numbers to better balance round-off error and speed.

As we shall show below (cf. Tables 2, 3, and 7), the combined results of our choices in parallelization and operation orders for sums for single-precision arithmetic significantly improve in runtimes even in comparison to a ported CPU parallelization, where the program for parallelization using the GPU instead uses available CPU cores. In addition, the table reveals that there are few negative effects in our use of single-precision machine numbers rather than double-precision. Our resulting parallel GPU implementation serves as the linchpin of our computations, as without it, we would not be able to obtain results in any reasonable amount of time due to the requirements of the moving atoms.

All calculations are performed on a 2017 iMac, with *3.50 GHz Intel (R) Core (TM) i5-7600* CPU, and the integrated GPU is *AMD Radeon Pro 575*, where the maximum number of compute units is 32. In our work, all sequential computations are executed on 1 CPU core, and all parallel computations are executed on the integrated GPU with OpenCL 1.2 using all 32 compute units.

5 Numerical Experiments and Applications

We first apply the binary level-set method and its GPU implementation to two molecular systems with fixed solute particles, a system of two parallel charged plates, and the protein BphC, and show that the binary level-set method is accurate in qualitatively reproducing the known results of those two systems. We then consider the full application of our model and numerical methods to two small molecular systems, a two-particle system and the ethane molecule, to show the convergence of our algorithm. Finally, we study the p53-MDM2 binding process with solute mechanical interactions to demonstrate the efficiency of our methods and GPU implementation. Table 1 summarizes the continuum model parameters used in all these numerical computations.

Table 1: Model parameters.

Parameter	Symbol	Value	Unit
temperature	T	298	K
solvent number density	ρ_{w}	0.0333	Å^{-3}
surface tension	γ_0	0.174	$\mathrm{k_B T}/\text{Å}^2$
solute dielectric constant	ε_{m}	1	
solvent dielectric constant	ε_{w}	80	

5.1 Free-energy minimization with fixed solute atoms

We consider two molecular systems each with fixed solute atomic positions, and apply the binary level-set method with GPU implementation to minimize the solvation free-energy functional (2.2) of solute-solvent interfaces Γ with all atomic positions $\mathbf{r}_i$ $(1 \leq i \leq N)$ fixed. Both systems have been studied extensively with continuous level-set method and CPU computations [30, 33, 35]. Here we show the qualitative accuracy, and efficiency, of our new algorithm and implementation.

Two parallel charged plates. Each of these two plates consists of 6×6 CH_2 atoms with a square length of about 3 nm. The two plates are placed at a center-to-center distance d. In the following, we investigate how (a) the capillary evaporation, (b) the hydrophobic attraction, and (c) a possible hysteresis in the free energy are affected by charging up the plates. To this end, we assign central charges q_1 and q_2 to the first and second plates, respectively, with $|q_1| = |q_2|$. The total charges of these two plates are $36q_1$ and $36q_2$, respectively. We study like-charged and oppositely charged plates by choosing the values of (q_1, q_2) to $(+0.2e, +0.2e)$, and $(+0.2e, -0.2e)$. The atom-water LJ parameters are $\varepsilon = 0.262\, k_B T$, $\sigma = 3.15365\, \mathring{A}$, and the atom-atom LJ are $\varepsilon = 0.265\, k_B T$ and $\sigma = 3.54\, \mathring{A}$.

We first investigate the VISM surfaces of the two plates at different distances with the like-charge (+0.2e, +0.2e) with different initial configurations. Figure 3 shows a few snapshots of stable 3D equilibrium solute-solvent surfaces of the two parallel charged plates system obtained by the binary level set VISM calculations with loose or tight initial interface at $d = 9\, \mathring{A}$, $d = 13\, \mathring{A}$, and $d = 16\, \mathring{A}$. In the top row of Figure 3, with the loose initial interface, a stable capillary bubble remains between the two charged parallel plates at $d = 9\, \mathring{A}$ and $d = 13\, \mathring{A}$, and the bubble becomes tighter along the enlarging distance. At $d = 16\, \mathring{A}$, the bubble disappears. Comparatively, with the tight initial interface, the equilibrium state is wet at $d = 9\, \mathring{A}$, $d = 13\, \mathring{A}$, and $d = 16\, \mathring{A}$.

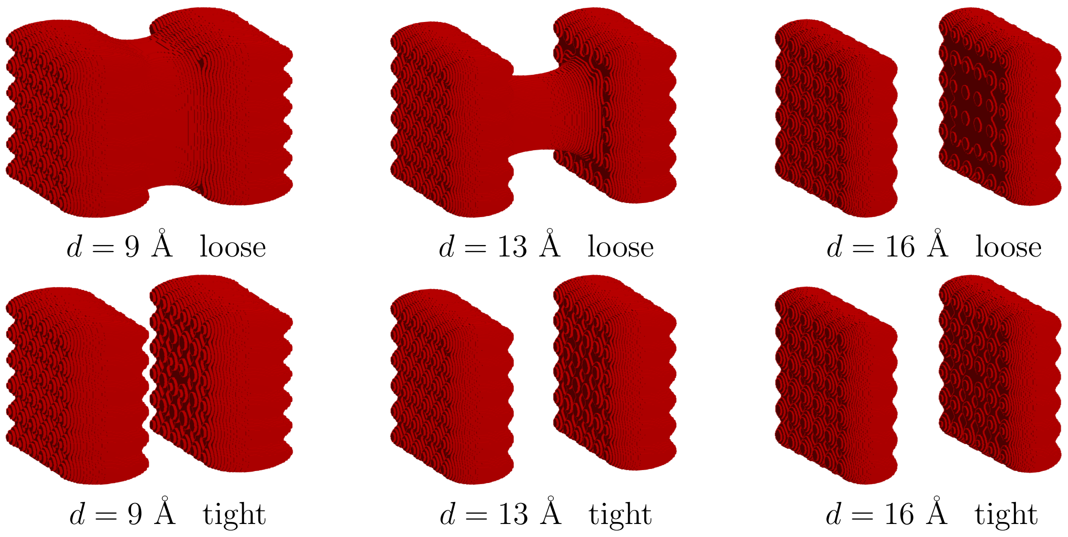

Figure 3: Stable 3D equilibrium solute-solvent surfaces of the two parallel charged plates obtained by the binary level set VISM calculations with loose (top row) or tight (bottom row) initial interface at $d = 9\,\mathring{A}$, $d = 13\,\mathring{A}$, and $d = 16\,\mathring{A}$. Atomic charges are (+0.2e,+0,2e).

We now examine the potential of mean force (PMF) of the two-plate system with respect to the plate-plate separation distance d. This is the VISM free-energy value as a function of d, with an additive constant such that the free energy is 0 at the infinite plate-plate separation. For a given d, we may have two VISM free-energy minimizing solute-solvent interfaces corresponding to a tight and a loose initial surface, respectively. We denote by $G^{\rm pmf}_{\rm VISM}(d)$ the corresponding minimum free energy of one of the two branches, and denote by $G^{\rm pmf}_{\rm geom}(d)$, $G^{\rm pmf}_{\rm vdW}(d)$, and $G^{\rm pmf}_{\rm elec}(d)$ the components of the PMF, corresponding to the first, second, and third terms in (2.2), respectively. Precise definition is given in Appendix.

Figure 4 displays the bimodal behavior and hysteresis of the two different PMF branches stemming from the equilibria of wet and dry states, i.e., the VISM free-energy minimizing surfaces corresponding to initial tight and loose surfaces. Atomic charges considered here are (+0.2e,-0,2e) and (+0,2e, +0,2e), respectively. We can see that like-charged and oppositely charged plates give different free-energy branches and hysteresis. For the like-charged cases in Figure 4, a strong hysteresis is presented for $8 \lesssim d \lesssim 15\mathring{A}$. For the oppositely charged plates, strong hysteresis is presented for $6 \lesssim d \lesssim 8\mathring{A}$.

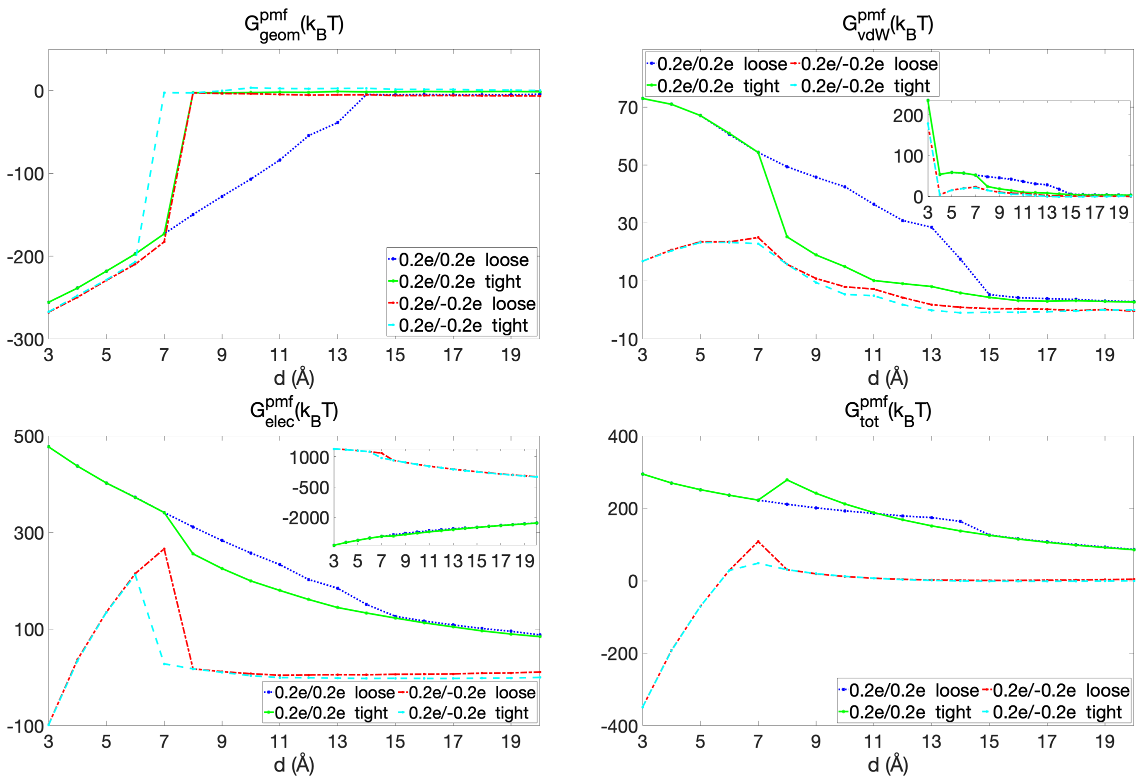

Figure 4: Different parts of the PMF of the two parallel charged plates with respect to the separation distance d, with loose and tight initial surfaces. (a) The geometrical part $G^{\mathrm{pmf}}_{\mathrm{geom}}$. (b) The vdW part $G^{\mathrm{pmf}}_{\mathrm{vdW}}$. The solute-solute vdW interactions are excluded in the curves in the main frame but included in those in the inset. (c) The electrostatic part $G^{\mathrm{pmf}}_{\mathrm{elec}}$. The solute charge-charge interactions are excluded in the curves in the main frame but included in those in the inset. (d) The total PMF $G^{\mathrm{pmf}}_{\mathrm{tot}}$. The values of $(+0.2e, -0, 2e)$ with tight initial interface are used as reference values to show the difference.

In Table 2, we show a comparison of the calculation speed and different parts of the free energy using the binary level-set method between GPU single precision code and CPU double precision code of the two charged parallel plates system. Three different grid sizes are shown. We can observe that the results for free-energy estimates from CPU double precision code and GPU single precision code are nearly the same. The improvement of speed using GPU can be obtained by comparing the time. The cost time of CPU code is around 5 times of the GPU code with three different grid sizes.

The protein BphC. In this example, we consider biphenyl-2, 3-diol-1, 2-dioxygenase (BphC), an enzyme protein (PDB code: 1dhy).

The functional unit of this protein is a homo-octamer, and each subunit consists of two domains. We set up a series of configurations where the two domains are increasingly separated from $d = 0$ to $d = 20\,\mathring{A}$ apart, perpendicular to their interface. The domain separation d is chosen here to be the reaction coordinate. Note that the zero domain separation corresponds to the native configuration in the crystal structure.

Table 2: Comparison of GPU (single precision) and CPU (double precision) for free energy (k_BT) and its components of two parallel charged plates with different charges ($+0.2e, -0,2e$) at distance $d = 10\mathring{A}$. The unit of time is second.

Grid Points	Total Energy		Surface Energy		vdW Energy		CFA		Total Time	
	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU
72^3	-2099.5	-2099.5	631.8	631.8	-98.3	-98.3	-2632.9	-2633.0	0.6	3.0
144^3	-2090.8	-2090.8	640.3	640.3	-97.9	-97.9	-2633.1	-2633.2	4.0	21.2
288^3	-2082.0	-2082.0	648.3	648.3	-110.4	-110.4	-2619.9	-2619.9	29.9	163.9

Three pairs of stable equilibrium solute-solvent interfaces of BphC at $d = 8\,\mathring{A}$, $d = 12\,\mathring{A}$, and $d = 16\,\mathring{A}$ with tight or loose initial interfaces are presented in Figure 5. The top row is with the loose initial interfaces, and the bottom row is with the tight initial interfaces. We observe that the equilibria of the loose initial interface wrap the two domains of BphC at $d = 8\,\mathring{A}$ and $d = 12\,\mathring{A}$, and the equilibria surface becomes tighter along the increasing distance. At $d = 16\,\mathring{A}$, the interfaces of two domains separate, changing to the wet state. In contract, with the tight initial interface, all equilibria states are wet.

In Figure 6, different parts of the PMF of BphC with respect to the separation of two domains, from $d = 0$ to $d = 20\mathring{A}$ obtained by our binary level set calculations using tight and loose initial surfaces are displayed. These PMFs exclude the solute-solute vdW interactions. We observe the bimodal hydration behavior: the branches of different parts of the PMF of BphC between $4\,\mathring{A}$ and $14\,\mathring{A}$, indicating that initial interfaces can strongly affect the PMF of BphC.

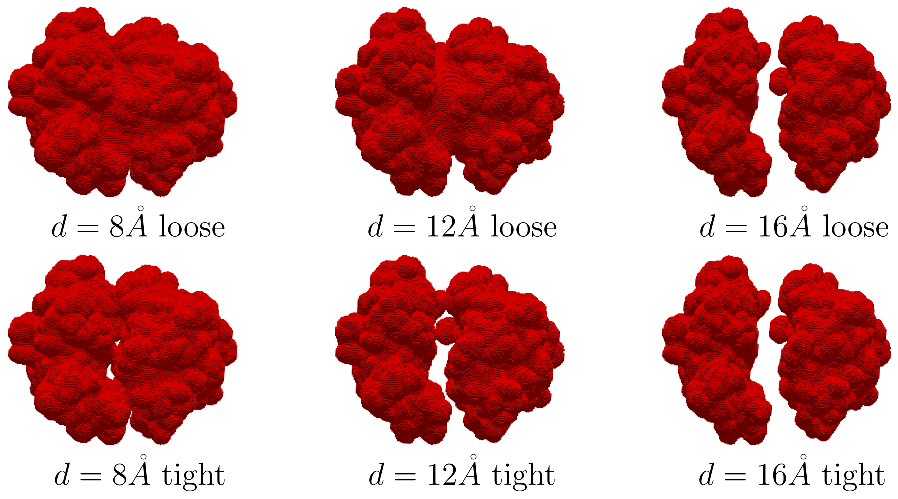
$d = 8\mathring{A}$ loose
$d = 12\mathring{A}$ loose
$d = 16\mathring{A}$ loose
$d = 8\mathring{A}$ tight
$d = 12\mathring{A}$ tight
$d = 16\mathring{A}$ tight

Figure 5: Stable 3D equilibrium solute-solvent surfaces of the BphC system obtained by the binary level set VISM calculations with loose (top) or tight (bottome) initial interface at $d = 8\,\mathring{A}$, $d = 12\,\mathring{A}$, and $d = 16\,\mathring{A}$.

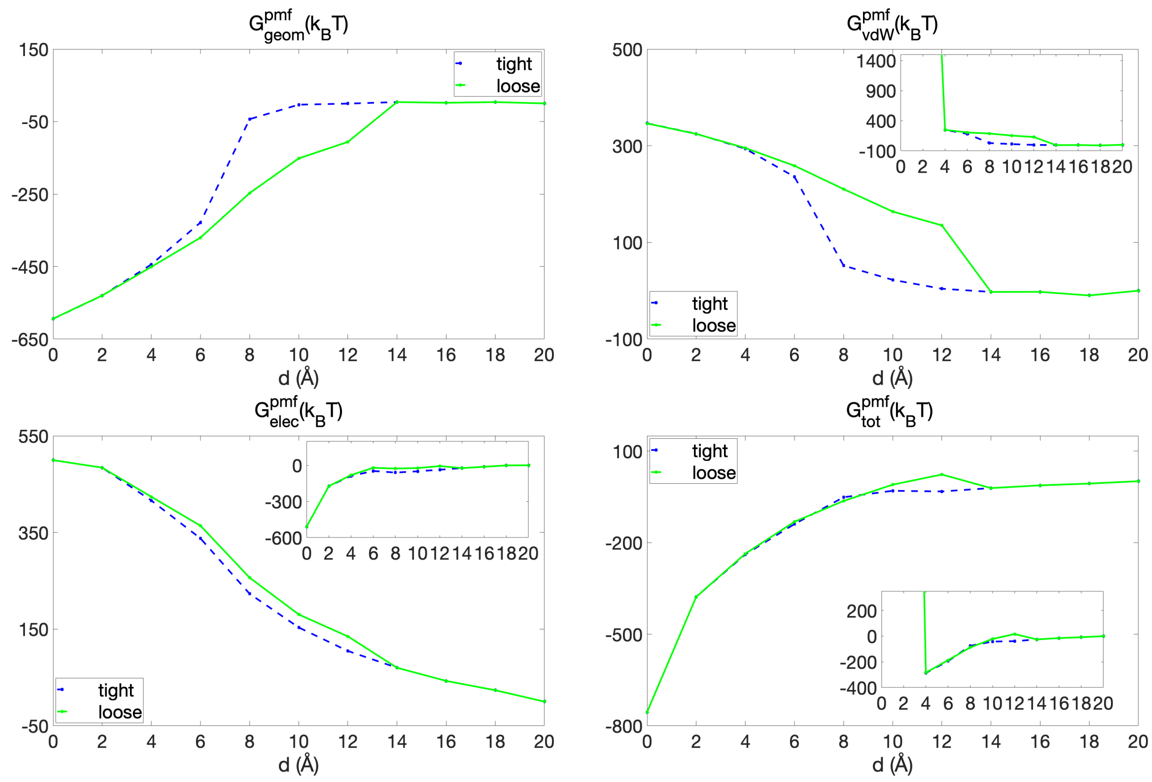

Figure 6: Different parts of the PMF of BphC with respect to the domain separations, with loose and tight initial surfaces. (a) The geometrical part $G_{\rm geom}^{\rm pmf}$. (b) The vdW part $G_{\rm vdW}^{\rm pmf}$. The solute-solute vdW interactions are excluded in the curves in the main frame but included in those in the inset. (c) The electrostatic part $G_{\rm elec}^{\rm pmf}$. The solute charge-charge interactions are excluded in the curves in the main frame but included in those in the inset. (d) The total PMF $G_{\rm tot}^{\rm pmf}$. The solute-solute vdW interactions are excluded in the curves in the main frame but included in those in the inset.

In Table 3, we show a comparison of the calculation speed and different parts of the free energy with the binary level-set method between GPU single precision code and CPU double precision code of the BphC. Three different grid sizes are shown. We observe that the results from CPU double precision code and GPU single precision code are nicely consistent. With the grid size of 50^3, 100^3, and 200^3, the time cost of CPU code is around 15 times, 78 times, and 216 times correspondingly to the time cost of the GPU code.

5.2 Explicit-solute implicit-sovent free-energy minimization

In this section, we conduct numerical experiments on two molecuales that are treated as nonpolar systems (i.e., no charges) to demonstrate the efficiency of our free-energy minimization algorithm.

A two-atom molecule. We consider an artificial molecular system of two atoms. The solute-water LJ parameters are $\sigma = 3.57\,\mathring{A}$ and $\varepsilon = 0.431\,k_BT$. We additionally assume

Table 3: Comparison of GPU (single precision) and CPU (double precision) for free energy ($k_B T$) and its components of BphC with the native configuration in the crystal structure. The unit of time is second.

Grid	Total Energy		Surface Energy		vdW Energy		CFA		Total Time	
Points	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU
50^3	52133.7	52134.2	1588.2	1588.2	53116.8	53117.4	-2571.2	-2571.3	4.1	60.6
100^3	52251.8	52252.3	1658.3	1658.3	53130.4	53131.0	-2537.0	-2537.0	4.9	381.1
200^3	52303.7	52304.2	1654.3	1654.3	53146.7	53147.3	-2497.3	-2497.3	13.5	2911.0

that the two atoms are bonded, with the spring constant in the bond stretching energy $A = 800\, k_B T/\mathring{A}^2$. We set the computational box to be $(-8, 8)^3\, \mathring{A}^3$.

We design two sets of experiments on the optimization process and equilibria with different initial configurations. In the first set of experiments, Experiment 1.1.a and Experiment 1.1.b, we set the equilibrium bond length $r_0 = 3\,\mathring{A}$. In the second set of experiments, Experiment 1.2.a and Experiment 1.2.b, we set the equilibrium bond length $r_0 = 8\,\mathring{A}$. In each set of experiments, we test two types of initial configurations. In Experiment 1.1.a and Experiment 1.2.a, we place initially the two solute atoms far away from each other so that their distance is much larger than the equilibrium bond length. We place the two solute atoms at positions $(7, 0, 0)$ and $(-7, 0, 0)$, respectively. In Experiment 1.1.b and Experiment 1.2.b, we place initially the two solute atoms very close to each other so that their distance is smaller than the equilibrium distance. Specifically, we place the two solute atoms initially at positions $(0.5, 0, 0)$ and $(-0.5, 0, 0)$, respectively.

In Figure 7, the minimization processes of Experiment 1.1.a and Experiment 1.1.b are displayed. The red dots represent two atoms, and the blue segment represents the bond. In the top row of Figure 7, we can see that initially surface consists of two disconnected spheres, then two atoms get closer, and spheres merge, until the system reaches an equilibrium state. In the bottom row of Figure 7, initially, the two atoms are very close to each other, then the atoms are pushed apart due to the force from strong bonding energy, the interface is moved accordingly, and then the system reaches to an equilibrium state.

We observe from Table 4 that that atoms have the exact same positions and free energy in the equilibrium from the two experiments 1.1.a and 1.1.b, indicating that the molecular system in the two simulations reached the same equilibrium.

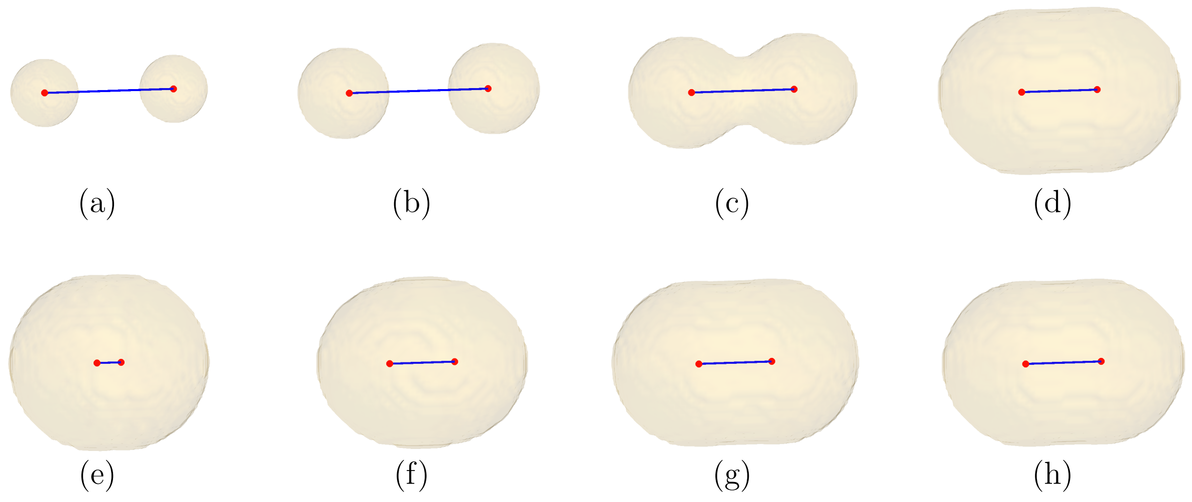

Figure 7: The free-energy minimization for a two-atom system. Top: Experiment 1.1.a. The snapshots are taken at (a) initial stage step 0, (b) step 5, (c) step 10, and (d) step 63, reaching nearly the steay state. Bottom: Experiment 1.1.b. The snapshots are taken at (e) initial stage step 0, (f) step 2, (g) step 4, and (h) step 53, reaching nearly the steay state.

Table 4: Comparison of experiments 1.1.a and 1.1.b of a two-atom system.

Experiment	Initial position		Final position		Bond length	Free Energy
1.1.a	(7,0,0)	(-7,0,0)	(1.50,0,0)	(-1.498,0,0)	3.00	21.85
1.1.b	(0.5,0,0)	(-0.5,0,0)	(1.50,0,0)	(-1.498,0,0)	3.00	21.85

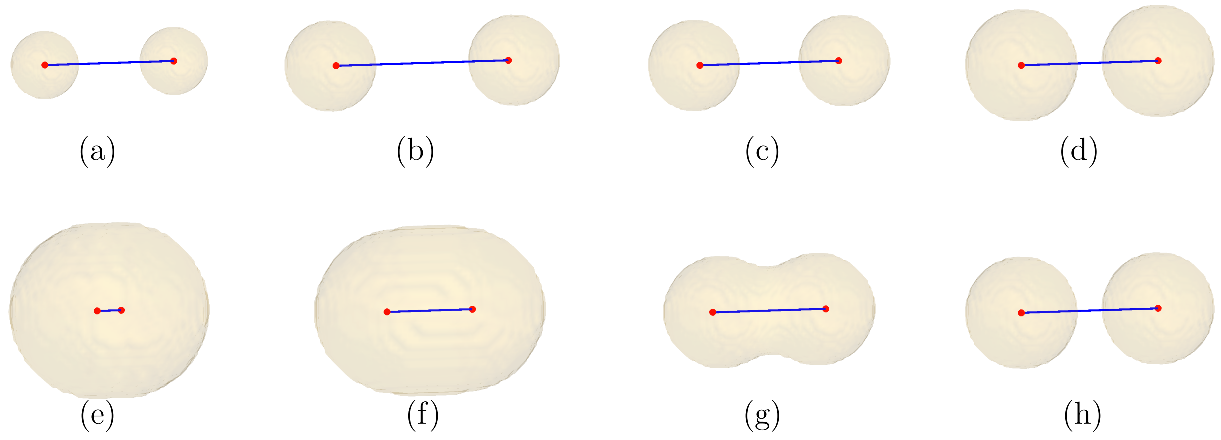

Figure 8: The free-energy functional minimization algorithm for a two-atom system. Top: Experiment 1.2.a. The screenshots are taken at (a) initial stage step 0, (b) step 2, (c) step5, and (d) the steay state step 62. Bottom: Experiment 1.2.b. The screenshots are taken at (e) initial stage step 0, (f) step 3, (g) step 7, and (h) the steay state step 64.

Figure 8 shows the minimization processes of Experiment 1.2.a and Experiment 1.2.b. In the top row of Figure 8, we can see that initially surface consists of two disconnected spheres, then two atoms get closer, until the system reaches an equilibrium state. Comparing with the equilibrium of Experiment 1.1.a, the spheres are not merged due to a larger bond length $8\mathring{A}$. In the bottom row of Figure 8, initially, the two atoms are very close to each other, then the atoms are pushed apart due to the force from strong bonding energy, the interface moves and then splits apart, then the system reaches to an equilibrium state with which the interface consists of two separate spheres. Table 5 shows that the two experiments 1.2.a and 1.2.b reach the same equilibrium.

Table 5: Comparison of Experiments 1.2.a and 1.2.b of a two-atom system.

Experiment	Initial position		Final position		Bond length	Free Energy
1.2.a	(7,0,0)	(-7,0,0)	(4.00,0,0)	(-4.00,0,0)	8.00	33.93
1.2.b	(0.5,0,0)	(-0.5,0,0)	(4.00,0,0)	(-4.00,0,0)	8.00	33.93

An ethane molecule. We consider an ethane molecule C_2H_6 in water and take from [13, 14, 17] the solute atomic positions and force field parameters. Other parameters are as follows: the carbon-water LJ parameters $\sigma = 3.4767\,\mathring{A}$ and $\varepsilon = 0.2311\,k_BT$, the hydrogen-water LJ parameters $\sigma = 3.1017\,\mathring{A}$ and $\varepsilon = 0.0989\,k_BT$, the carbon-carbon LJ parameters $\sigma = 3.4\,\mathring{A}$ and $\varepsilon = 0.344\,k_BT$, the carbon-hydrogen LJ parameters $\sigma = 3.025\,\mathring{A}$ and $\varepsilon = 0.147k_BT$, and the hydrogen-hydrogen LJ parameters $\sigma = 2.650\,\mathring{A}$ and $\varepsilon = 0.063\,k_BT$.

In the ethane molecule, each atom is connected to other atoms through bonding, bending, and torsion structure. The effect of vdW interaction energy among unbonded pairs of solute atoms is relatively less important when compared with molecular mechanical interactions, so in our simulation experiment of the ethane molecule, we neglect the vdW interaction energy among unbonded pairs of solute atoms.

We design three different initial configurations of the ethane molecule from its equilibrium:

- In Experiment 2.1.a, we stretch all hydrogen-carbon bonds to be $2\mathring{A}$.
- In Experiment 2.1.b, we stretch or shrink all hydrogen-carbon bonds such that hydrogen-carbon bonds have the length of $1\,\mathring{A}$, $1.25\,\mathring{A}$, $1.5\,\mathring{A}$, $1.75\,\mathring{A}$, $2\,\mathring{A}$, and $2.25\,\mathring{A}$.
- In Experiment 2.2, we introduce a small fluctuation, then rotate one set of three hydrogen-carbon bonds 50 degrees with respect to the carbon-carbon bond.

In Table 6, the bond lengths of hydrogen-carbon bonds and the carbon-carbon bond in their equilibrium of Experiment 2.1.a, Experiment 2.1.b, and Experiment 2.2 are compared. It is clear that in the equilibrium, all hydrogen-carbon bonds in three experiments have the same length, which is consistent with the reference length of hydrogen-carbon bond $1.093\,\mathring{A}$. The carbon-carbon bond in each of the three experiments is the same as the reference length $1.508\,\mathring{A}$. This verifies the accuracy of our free-energy minimization algorithm.

Table 6: Comparison of bond lengths ($\mathring{A}$) of ethane in equilibrium from different initial configurations. Remark: H3, H4, and H5 are hydrogen atoms bonded with the carbon atom C1, H6, H7, and H8 are hydrogen atoms bonded with the carbon atom C2.

Experiments	Experiment 2.1.a		Experiment 2.1.b		Experiment 2.2	
Bond list	Initial Length	Final Length	Initial Length	Final Length	Initial Length	Final Length
C1-H3	2	1.093	1	1.093	1.090	1.093
C1-H4	2	1.093	1.25	1.093	1.090	1.093
C1-H5	2	1.093	1.5	1.093	1.090	1.093
C2-H6	2	1.093	1.75	1.093	1.090	1.093
C2-H7	2	1.093	2	1.093	1.090	1.093
C2-H8	2	1.093	2.25	1.093	1.090	1.093
C1-C2	1.77	1.508	1.77	1.508	1.540	1.508

Figure 9 displays the snapshots of minimization process of Experiment 2.2. The red dots represent carbon atoms, the blue dots represent the hydrogen atoms, the light blue segments and green segments represent the hydrogen-carbon bonds, and the black segment represents the carbon-carbon bond. It is captured that during the relaxation, the set of three hydrogen-carbon bonds rotated back to their equilibrium, and all hydrogen-carbon bonds have the same length. Free energy of steady state in Experiment 2.1.a is 3.692 k_BT, the free energy of steady state in Experiment 2.1.b is 3.685 k_BT, and the free energy of steady state in Experiment 2.2 is 3.687 k_BT. Thus, the three experiments get to the same equilibria. We remark that the free energy here does not include the vdW interaction energy among unbonded pairs of solute atoms.

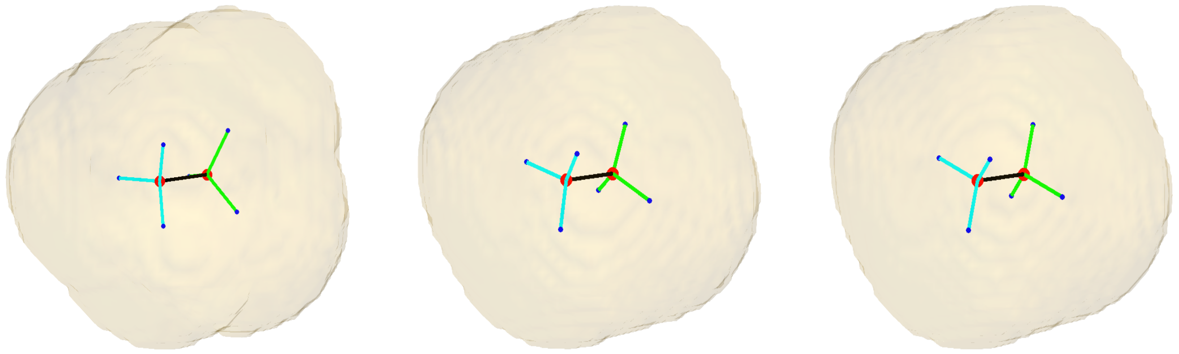

Figure 9: The free-energy minimization for ethane from Experiment 2.2. Snapshot taken at step 0 (Left), step 200 (Middle), and step 1500, reaching to a steady state (Right).

In Figure 10, we plot the free energy vs. iteration steps in numerical computations for Experiment 2.1.a and Experiment 2.2. In Experiment 2.1.a, initially, the free energy is very large, that is because the stretch or shrink of the initial bond length causes a large value of the bonding energy. We can see that the free energy decays very fast in the first few steps, which is caused by the dominant force from the bonding energy. It takes around 1000 steps to adjust positions of solute atoms in ethane molecule to reach the equilibrium. In contract,

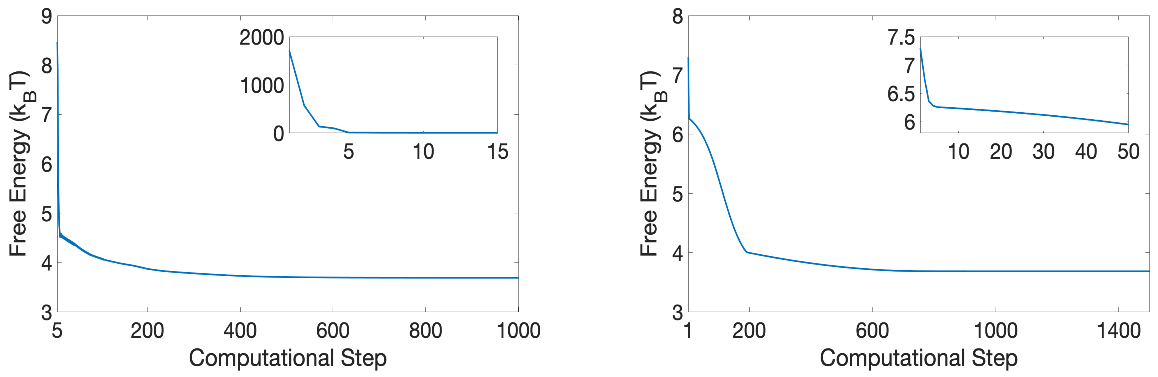

Figure 10: The free energy (k_BT) vs. the computational step in the free-energy minimization for an ethane molecule in Experiment 2.1.a and Experiment 2.2. First 15 (50) computational steps are specified in the inset of each subfigure.

the initial free energy of Experiment 2.2 in Figure 10 is a relatively small value, as we only introduced a small fluctuation to the initial atomic positions and the rotation of a set of three hydrogen-carbon bonds did not cause large free energy change. We observe that the rate of free-energy change at around the 3rd step and 200th computational step becomes slower and slower, which indicates that our free-energy minimization algorithm was adjusting the suitable mobility factor M during the minimization process. Although the initial free energy is small, it takes more than 1400 steps to rotate the hydrogen-carbon bonds back to the right position and reach the equilibrium.

5.3 Simulation of protein-protein interactions

In this section, we choose a biologically important and realistic system, the p53/MDM2 protein complex, and investigate the binding behavior using our free-energy minimization model and algorithm. To make the calculations of molecular movement easier, the receptor protein MDM2 here is fixed.

We use the CHARMM36 force field [2,16,21,22] for our VISM simulations for the binding of p53 and MDM2.

Table 7 shows the solvation free energy and its components obtained by our VISM simulations for the bound complex p53/MDM2, and the computational times of the simulations with the GPU single precision and the CPU double precision, respectively.

During those simulations, we relax the relative difference stop criterion and absolute difference stop criteria to be 1e-3, just for the efficiency comparison of the GPU code and the CPU code. We set the initial configuration of p53/MDM2 to be a tight initial interface for atomic position with small fluctuations of p53/MDM2 in the bound complex, where the bound complex is taken from the Protein Data Bank (PDB code: 1ycr.pdb). It can be observed in Table 7 that the difference between GPU code with single precision data type and CPU with double precision data type is less than 1%, but the cost time of CPU is more than 78 times, and 307 times slower than the cost time of GPU with number of grid points

50^3, and 100^3, correspondingly.

Table 7: Comparison of GPU (single precision) and CPU (double precision) for free energy (k_BT) and its components of p53/MDM2 in the steady state. The unit of time is minute.

Grid Points	Total Energy		Surface Energy		vdW Energy		CFA		Total Time	
	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU
50^3	-214.0	-212.0	800.7	802.7	-453.1	-453.0	-561.6	-561.7	10.0	784.7
100^3	-157.3	-159.1	833.8	834.0	-441.9	-440.6	-549.2	-552.4	16.7	5128.6
200^3	-140.9	-	836.5	-	-434.4	-	-543.0	-	62.6	-

We further investigate the binding behavior of p53 and MDM2 with our free-energy minimization method and algorithm. Note that, here we use a grid size $h = 0.5\,\mathring{A}$, the relative difference stop criterion and absolute difference stop criterion are 1e-5. We construct the initial configuration with a tight initial interface by pulling p53 away from the MDM2 pocket in the bound complex along the line passing through the geometrical centers.

In Figure 11, a few snapshots of numerical results of p53/MDM2 are displayed, showing the minimization process. The position of the protein MDM2 is fixed, positions of p53 atoms are adjusted by the free-energy minimization process. We color each piece of surface according to whether its closet solute atom comes from MDM2 (red) or p53 (blue) to show the relative positions of MDM2 and p53. In the initial configuration, the protein p53 and the receptor protein MDM2 are separated as we can observe a hole between them; that is a small region filled with water. In the process, the relative positions of p53 and MDM2 are adjusted, p53 and MDM2 become closer and closer. In the equilibrium, the hole disappears, and the two proteins are combined together.

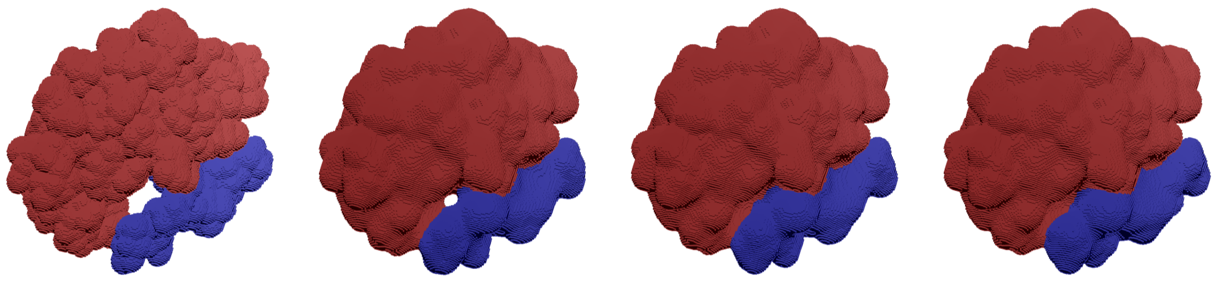

Figure 11: The free-energy minimization for p53/MDM2. The snapshots are taken at (a) initial stage, i.e., step 0, (b) step 100, (c) step 200, and (d) step 377, reaching nearly the steady state. Note that, in order to show the relative positions of MDM2 and p53, we color each piece of surface according to whether its closest solute atom comes from MDM2 (red) or p53 (blue).

6 Conclusions

This work presents the development of a GPU parallel free-energy minimization method and algorithm with the fast binary level-set method and an adaptive-mobility gradient descent method for the variational explicit-solute implicit-solvent (VESIS) molecular simulations. Minimization of the free-energy functional determines an equilibrium interface and an equilibrium molecular structure.

Our free-energy minimization is an iterative process with two stages. In the first stage, we fix solute atoms, then minimize the solvation free energy to obtain an optimal solute-solvent interface. In the second stage, with a fixed interface, we relax the solute atoms using a gradient descent type method. The proposed minimization algorithm is implemented in parallel on GPUs with single precision.

We have presented a series of numerical experiments and have demonstrated the accuracy and efficiency of our numerical methods and algorithm for the free-energy minimization. In particular, our numerical experiments of potentials of mean force for two charged systems, two charged parallel plates, and the protein BphC, have shown that VISM with the binary level-set method can capture well the sensitive response of capillary evaporation to the charge in hydrophobic confinement and the polymodal hydration behavior. Moreover, our numerical experiments for small molecular systems, the two-atom system and an ethane molecule, have demonstrated that our algorithm can capture topological changes of the solute-solvent interfaces as well as describe the equilibrium molecular structure. A key application of our algorithm is for large biomolecular simulations. We have applied our free-energy minimization method and algorithm to a realistic system, the p53/MDM2 protein complex. Our model and method describes the relaxation process of the binding of these two molecules.

To verify the performance of our algorithm with the parallel implementation on the integrated GPU, we have compared for different molecular systems our computational results and computational times with both of the CPU with double precision and the GPU with single precision. We observe that the GPU implementation is much more efficient than the CPU implementation. The GPU with single precision combining the pairwise summation can efficiently limit the grow of round-off errors. For small molecular systems such as the two charged parallel plates the computational time with the CPU is around 5 times of that with the GPU. For a relatively large molecular system, such as p53/MDM2, and fine finite-difference grids, our GPU implementation works especially well, reaching a speed about 100 times faster than that of the CPU implementation. In the meantime, both implementations lead to the same minimum free energies and even their individual components.

To speed up the computations further, our immediate next step is to construct a hybrid CPU-GPU architecture to combine CPU parallel computing and the integrated GPU parallel computing together. For the CPU parallel computing, we can use a standard domain decomposition approach. The communication between sub-domains is based on the message-passing interface (MPI). Additionally, we would like to explore the high performance efficiency and good scaling of parallel computing on dedicated GPUs.

With our fast algorithm and GPU code, we can now carry out flexible VESIS-Monte Carlo simulations for the binding of two proteins in which both the solute-solvent interface and the set of solute atomic positions change in each step of the Monte Carlo move.

598 # Appendix

Gradient of $G[\Gamma, \mathbf{R}]$. Fix n with $1 \le n \le N$. We have

$$\begin{aligned}
\nabla_{\mathbf{r}_n} G[\Gamma, \mathbf{R}] = & \frac{1}{8\pi^2\varepsilon_0}\left(\frac{1}{\varepsilon_\mathrm{w}} - \frac{1}{\varepsilon_\mathrm{m}}\right)\int_{\Omega_\mathrm{w}}\left(\sum_{i=1}^{N}\frac{Q_i(\mathbf{r}-\mathbf{r}_i)}{|\mathbf{r}-\mathbf{r}_i|^3}\right)\left(\frac{Q_n}{|\mathbf{r}-\mathbf{r}_n|^3}\right) dV_\mathbf{r} \\
& + \rho_0 \int_{\Omega_\mathrm{w}} U'_{sw}(|\mathbf{r}_n - \mathbf{r}|)\frac{\mathbf{r}_n - \mathbf{r}}{|\mathbf{r}_n - \mathbf{r}|} dV_\mathbf{r} + \sum_{(i,j)'} \delta_{ni} G'^{\mathrm{ss}}_{\mathrm{elec}}(|\mathbf{r}_i - \mathbf{r}_j|)\frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|} \\
& + \sum_{(i,j)'} \delta_{ni} U'_{ss}(|\mathbf{r}_i - \mathbf{r}_j|)\frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|} + \sum_{(i,j)} \delta_{ni} A_{ij}(|r_{ij} - r_{0ij}|)\frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|} \\
& + \sum_{(i,j,k)} \nabla_{\mathbf{r}_n} W_\mathrm{bend}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) + \sum_{(i,j,k,l)} \nabla_{\mathbf{r}_n} W_\mathrm{torsion}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k, \mathbf{r}_l), \qquad \text{(A.1)}
\end{aligned}$$

599 where $\delta_{ni} = 1$ if $n = i$ and 0 otherwise.

Force calculations of molecular mechanical interactions. For fixed $\mathbf{r}_i, \mathbf{r}_j$, and $\mathbf{r}_k$, denote the vector from $\mathbf{r}_j$ to $\mathbf{r}_i$ by $\mathbf{q}_{ji} = \mathbf{r}_i - \mathbf{r}_j$ for any i and j and the length of $\mathbf{q}_{ji}$ by $q_{ji} = |\mathbf{q}_{ji}|$. We have

$$\nabla_{\mathbf{r}_n} W_\mathrm{bend}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) = B_{ijk}(\theta_{ijk} - \theta_{0ijk})\nabla_{\mathbf{r}_n}\theta_{ijk}, \quad n = i, j, k,$$

where

$$\begin{aligned}
\nabla_{\mathbf{r}_i}\theta_{ijk} &= \frac{1}{\sin\theta_{ijk}}\left(\frac{\mathbf{q}_{ji}\cdot\mathbf{q}_{jk}}{q_{ji}^3 q_{jk}}\mathbf{q}_{ji} - \frac{1}{q_{ji}q_{jk}}\mathbf{q}_{jk}\right), \\
\nabla_{\mathbf{r}_k}\theta_{ijk} &= \frac{1}{\sin\theta_{ijk}}\left(\frac{\mathbf{q}_{ji}\cdot\mathbf{q}_{jk}}{q_{jk}^3 q_{ji}}\mathbf{q}_{ji} - \frac{1}{q_{ji}q_{jk}}\mathbf{q}_{ji}\right), \\
\nabla_{\mathbf{r}_j}\theta_{ijk} &= \frac{1}{\sin\theta_{ijk}}\left[\left(\frac{1}{q_{ji}q_{jk}} - \frac{\cos\theta_{ijk}}{q_{ji}^2}\right)\mathbf{r}_{ji} + \left(\frac{1}{q_{ji}q_{jk}} - \frac{\cos\theta_{ijk}}{q_{jk}^2}\right)\mathbf{r}_{jk}\right].
\end{aligned}$$

Recall for for fixed $\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k$, and $\mathbf{r}_l$ that

$$W_\mathrm{torsion}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k, \mathbf{r}_l) = \sum_{n=0}^{6} C_n[1 + \cos(n\tau - \psi_n)],$$

where ψ_n is the phase factor, which is introduced to shift the zero of the torsion potential. The phase angles ψ_n are usually chosen so that terms with positive C_n has minima at 180° (i.e., for odd n, $\psi_n = 0°$ and for even n, $\psi_n = 180°$). We denote

$$\mathbf{q}_1 = \mathbf{q}_{ij}, \quad \mathbf{q}_2 = \mathbf{q}_{jk}, \quad \mathbf{q}_3 = \mathbf{q}_{kl},$$

$$
\begin{aligned}
&\mathbf{u} = \mathbf{q}_1 \times \mathbf{q}_2, \quad \mathbf{v} = \mathbf{q}_2 \times \mathbf{q}_3, \\
&\tau = \tau_{ijkl}, \quad \Lambda = \Lambda_{ijkl} = \cos\tau = \frac{\mathbf{u}\cdot\mathbf{v}}{|\mathbf{u}||\mathbf{v}|}, \\
&C_n = C_{n,ijkl}, \quad n = 1,2,3,4,5,6.
\end{aligned}
$$

Due to the fact that $\psi_n = 0^\circ$ or 180°, we derive

$$
\begin{aligned}
&\nabla_{\mathbf{r}_n} W_{\text{torsion}}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k, \mathbf{r}_l) \\
&\quad = \nabla_{\mathbf{r}_n}\Lambda[(C_1\cos(\psi_1) - 3C_3\cos(\psi_3) + 5C_5\cos(\psi_5)) \\
&\qquad + \Lambda(4C_2\cos(\psi_2) - 16C_4\cos(\psi_4) + 36C_6\cos(\psi_6)) + \Lambda^2(12C_3\cos(\psi_3) - 60C_5\cos(\psi_5)) \\
&\qquad + \Lambda^3(32C_4\cos(\psi_4) - 192C_6\cos(\psi_6)) + \Lambda^4(80C_5\cos(\psi_5)) + \Lambda^5(192C_6\cos(\psi_6))],
\end{aligned}
$$

where

$$
\begin{aligned}
\nabla_{\mathbf{r}_i}\Lambda &= -\nabla_{\mathbf{q}_1}\Lambda, \\
\nabla_{\mathbf{r}_j}\Lambda &= \nabla_{\mathbf{q}_1}\Lambda - \nabla_{\mathbf{q}_2}\Lambda, \\
\nabla_{\mathbf{r}_k}\Lambda &= \nabla_{\mathbf{q}_2}\Lambda - \nabla_{\mathbf{q}_3}\Lambda, \\
\nabla_{\mathbf{r}_l}\Lambda &= \nabla_{\mathbf{q}_3}\Lambda,
\end{aligned}
$$

and

$$
\begin{aligned}
\nabla_{\mathbf{q}_1}\Lambda &= \frac{(\mathbf{q}_2\times\mathbf{v})|\mathbf{u}|^2 - (\mathbf{u}\cdot\mathbf{v})(\mathbf{q}_2\times\mathbf{u})}{|\mathbf{u}|^3|\mathbf{v}|}, \\
\nabla_{\mathbf{q}_2}\Lambda &= \frac{(-\mathbf{q}_1\times\mathbf{v})|\mathbf{u}|^2 + (\mathbf{u}\cdot\mathbf{v})(\mathbf{q}_1\times\mathbf{u})}{|\mathbf{u}|^3|\mathbf{v}|} + \frac{(\mathbf{q}_3\times\mathbf{u})|\mathbf{v}|^2 - (\mathbf{u}\cdot\mathbf{v})(\mathbf{q}_3\times\mathbf{v})}{|\mathbf{u}||\mathbf{v}|^3}, \\
\nabla_{\mathbf{q}_3}\Lambda &= \frac{(-\mathbf{q}_2\times\mathbf{u})|\mathbf{v}|^2 + (\mathbf{u}\cdot\mathbf{v})(\mathbf{q}_2\times\mathbf{v})}{|\mathbf{u}||\mathbf{v}|^3}.
\end{aligned}
$$

Potentials of Mean Force. The potential of mean force (PMF) is a general term for the effective interaction between solutes that stems from direct solute-solute interactions and that is mediated by the solvent. It is usually defined with respect to a reaction coordinate as the difference between the free energy of solvated state at a given coordinate d and that at a fixed, reference coordinate d_{ref}. Here, we recall the definition of PMF for our VISM [30].

For a solute-solvent interface Γ, we denote by $G_{\text{geom}}[\Gamma]$, $G_{\text{vdW}}[\Gamma]$, and $G_{\text{elec}}[\Gamma]$ the first, second, and last term in $G_{\text{VISM}}[\Gamma]$ (2.2), respectively. Fix now a finite coordinate d. Denote by Γ_d a corresponding VISM optimal surface, i.e., a stable equilibrium solute-solvent interface minimizing locally the VISM solvation free-energy functional. We define the (total) PMF to be the sum of its separate contributions

$$
G^{\text{pmf}}_{tot}(d) = G^{\text{pmf}}_{\text{geom}}(d) + G^{\text{pmf}}_{\text{vdW}}(d) + G^{\text{pmf}}_{\text{elec}}(d),
$$

where

$$
G^{\text{pmf}}_{\text{geom}}(d) = G_{\text{geom}}[\Gamma_d] - G_{\text{geom}}[\Gamma_\infty],
$$

$$G^{\text{pmf}}_{\text{vdW}}(d) = G_{\text{vdW}}[\Gamma_d] - G_{\text{vdW}}[\Gamma_\infty] + \sum_{i=1}^{M} \sum_{j=M+1}^{N} U_{i,j}(|\mathbf{x}_i - \mathbf{x}_j|),$$

$$G^{\text{pmf}}_{\text{elec}}(d) = G_{\text{elec}}[\Gamma_d] - G_{\text{elec}}[\Gamma_\infty] + \frac{1}{4\pi\varepsilon_m\varepsilon_0} \sum_{i=1}^{M} \sum_{j=M+1}^{N} \frac{Q_i Q_j}{|\mathbf{x_i} - \mathbf{x_j}|}.$$

Here a quantity at ∞ is understand as the limit of that quantity at a coordinate d' as $d' \to \infty$. The double-sum terms above are the solute-solute vdW and charge-charge interactions.

As d becomes large, the VISM optimal solute solvent interface Γ_d becomes the union of two separate VISM optimal solute-solvent interface Γ_I and Γ_{II}, both independent of d. They are obtained by minimizing the VISM free energy functional for the corresponding groups of fixed, solute atoms. If we denote by G_{Γ_I} and $G_{\Gamma_{II}}$ the corresponding minimum VISM free energies for these individual groups of atoms, then $G_{\Gamma_\infty} = G_{\Gamma_I} + G_{\Gamma_{II}}$ Similarly, each component of the VISM free energy is the sum of that for the two groups of solute atoms, i.e., G in the above equation can be replaced by G_{geom}, or G_{vdW} or G_{elec}.

Acknowledgment.

This work was supported in part by an AMS Simons Travel Grant (SL), the US National Science Foundation through grant DMS-1913144 and DMS-2208465 (LTC & BL), and the US National Institutes of Health through grant R01GM132106 (BL). The authors thank Dr. Clarisse G. Ricci (through the support of NIH grant GM31749) and Professor Shenggao Zhou for helpful discussions.

References

[1] P. W. Bates, Z. Chen, Y. H. Sun, G. W. Wei, and S. Zhao. Geometric and potential driving formation and evolution of biomolecular surfaces. *J. Math. Biol.*, 59:193–231, 2009.

[2] R. B. Best, X. Zhu, J. Shim, P. E. M. Lopes, M. Jeetain, M. Feig, , and A. D. MacKerell Jr. Optimization of the additive CHARMM all-atom protein force field targeting improved sampling of the backbone ϕ, ψ and side-chain $\chi 1$ and $\chi 2$ dihedral angles. *J. Chem. Theory Comput.*, 8(9):3257–3273, 2012.

[3] L.-T. Cheng, J. Dzubiella, J. A. McCammon, and B. Li. Application of the level-set method to the implicit solvation of nonpolar molecules. *J. Chem. Phys.*, 127:084503, 2007.

[4] L.-T. Cheng, Y. Xie, J. Dzubiella, J. A. McCammon, J. Che, and B. Li. Coupling the level-set method with molecular mechanics for variational implicit solvation of nonpolar molecules. *J. Chem. Theory Comput.*, 5:257–266, 2009.

[5] Li-Tien Cheng, Bo Li, and Zhongming Wang. Level-set minimization of potential controlled hadwiger valuations for molecular solvation. *Journal of computational physics*, 229(22):8497–8510, 2010.

[6] J. Dzubiella, J. M. J. Swanson, and J. A. McCammon. Coupling hydrophobicity, dispersion, and electrostatics in continuum solvent models. *Phys. Rev. Lett.*, 96:087802, 2006.

[7] J. Dzubiella, J. M. J. Swanson, and J. A. McCammon. Coupling nonpolar and polar solvation free energies in implicit solvent models. *J. Chem. Phys.*, 124:084905, 2006.

[8] S. Esdoḡlu, M. Jacobs, and P. Zhang. Kernels with prescribed surface tension and mobility for threshold dynamics schemes. *J. Comput. Phys.*, 337:62–83, 2017.

[9] P. Gera, H. Kim, H. Kim, S. Hong, V. George, and C. Luk. Performance characterisation and simulation of intel's integrated gpu architecture. In *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 139–148. IEEE, 2018.

[10] F. Gibou and R. Fedkiw. A fast hybrid k-means level set algorithm for segmentation. In *4th Annual Hawaii International Conference on Statistics and Mathematics*, pages 281–291, 2005.

[11] Z. Guo, B. Li, J. Dzubiella, L.-T. Cheng, J. A. McCammon, and J. Che. Evaluation of hydration free energy by the level-set variational implicit-solvent model with the coulomb-field approximation. *J. Chem. Theory Comput.*, 9:1778–1787, 2013.

[12] Z. Guo, B. Li, J. Dzubiella, L.-T. Cheng, J. A. McCammon, and J. Che. Heterogeneous hydration of p53/mdm2 complex. *J. Chem. Theory Comput.*, 10:1302–1313, 2014.

[13] T. A. Halgren. Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94. *J. Comput. Chem.*, 17:490–519, 1996.

[14] T. A. Halgren. Merck molecular force field. II. MMFF94 van der Waals and electrostatic parameters for intermolecular interactions. *J. Comput. Chem.*, 17:520–552, 1996.

[15] AJ Hopfinger. Computer-assisted drug design. *Journal of medicinal chemistry*, 28(9):1133–1139, 1985.

[16] J. Huang, S. Rauscher, G. Nawrocki, T. Ran, M. Feig, B. L. de Groot, H. Grubmüller, and A. D. MacKerell. CHARMM36: An improved force field for folded and intrinsically disordered proteins. In *61st Annual Meeting of the Biophysical Society*, pages 175a–176a, 2017.

[17] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives. Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *J. Amer. Chem. Soc.*, 118(45):11225–11236, 1996.

[18] J. Lie, M. Lysaker, and X.-C. Tai. A binary level set model and some applications to Mumford–Shah image segmentation. *IEEE Trans. Image Proc.*, 15:1171–1181, 2006.

[19] J. Lie, M. Lysaker, and X.-C. Tai. A variant of the level set method and applications to image segmentation. *Math. Comput*, 75(255):1155–1174, 2006.

[20] K. Lum, D. Chandler, and J. D. Weeks. Hydrophobicity at small and large length scales. *J. Phys. Chem. B*, 103:4570–4577, 1999.

[21] A. D. MacKerell Jr., D. Bashford, M. L. D. R. Bellott, R. L. Dunbrack Jr, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, et al. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *The J. Phys. Chem. B*, 102(18):3586–3616, 1998.

[22] A. D. MacKerell Jr., M. Feig, and Charles L. Brooks. Improved treatment of the protein backbone in empirical force fields. *J. Amer. Chem. Soc.*, 126(3):698–699, 2004.

[23] B. Merriman, J. Bence, and S. Osher. Diffsion generated motion by mean curvature. In J. Taylor, editor, *Computational Crystal Growers Workshop*, pages 73–83. Amer. Math. Soc., 1992.

[24] R. Ramirez and D. Borgis. Density functional theory of solvation and its relation to implicit solvent models. *J. Phys. Chem. B*, 109:6754–6763, 2005.

[25] C. G. Ricci, B. Li, L.-T. Cheng, J. Dzubiella, and J. A. McCammon. Tailoring the variational implicit solvent method for new challenges: Biomolecular recognition and assembly. *Front. Mol. Biosci.*, 5(13), 2018.

[26] S. J. Ruuth. Efficient algorithms for diffusion-generated motion by mean curvature. *J. Comput. Phys.*, 144:603–625, 1998.

[27] S. J. Ruuth and B. Merriman. Convolution-thresholding methods for interface motion. *J. Comput. Phys.*, 169:678–707, 2001.

[28] S. J. Ruuth, B. Merriman, and S. Osher. Convolution generated motion as a link between cellular automata and continuum pattern dynamics. *J. Comput. Phys.*, 151:836–861, 1999.

[29] D. Wang, H. Li, X. Wei, and X.-P. Wang. An efficient iterative thresholding method for image segmentation. *J. Comput. Phys.*, 350:657–667, 2017.

[30] Z. Wang, J. Che, L.-T. Cheng, J. Dzubiella, B. Li, and J. A. McCammon. Level-set variational implicit solvation with the Coulomb-field approximation. *J. Chem. Theory Comput.*, 8:386–397, 2012.

[31] D. S. Watkins. *Fundamentals of matrix computations*, volume 64. John Wiley & Sons, 2004.

[32] Z. Zhang, C. G. Ricci, C. Fan, L.-T. Cheng, B. Li, and J. A. McCammon. Coupling Monte Carlo, variational implicit solvation, and binary level-set for simulations of biomolecular binding. *J. Chem. Theory Comput.*, 17:2465–2478, 2021.

[33] S. Zhou, L.-T. Cheng, J. Dzubiella, B. Li, and J. A. McCammon. Variational implicit solvation with Poisson–Boltzmann theory. *J. Chem. Theory Comput.*, 10(4):1454–1467, 2014.

[34] S. Zhou, L.-T. Cheng, H. Sun, J. Che, J. Dzubiella, B. Li, and J. A. McCammon. LS-VISM: A software package for analysis of biomolecular solvation. *J. Comput. Chem.*, 36:1047–1059, 2015.

[35] S. Zhou, H. Sun, L.-T. Cheng, J. Dzubiella, B. Li, , and J. A. McCammon. Stochastic level-set variational implicit-solvent approach to solute-solvent interfacial fluctuations. *J. Chem. Phys.*, 145:054114, 2016.