UNIVERSITY OF CALIFORNIA SAN DIEGO

Fast and Efficient Numerical Methods in Level-Set Variational Implicit Solvent Model

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Mathematics

by

Zirui Zhang

Committee in charge:

Professor Li-Tien Cheng, Chair
Professor Chung Kuan Cheng
Professor Alexander Cloninger
Professor Bo Li
Professor Rayan Saab

2022

The Dissertation of Zirui Zhang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

University of California San Diego

2022

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor and mentor Li-Tien Cheng. His support, patience, insight and enthusiasm are inspiring. I'm also grateful to my co-advisor Bo Li for his invaluable supervision and support. Thanks goes to Rayan Saab, Alexander Cloninger, Chung Kuan Cheng, Randolph Bank, Sébastien Michelin for their guidance and support in research. I would like to thank Clarisse Gravina Ricci, James Andrew McCammon, Chao Fan, Shuang Liu for their help and contribution to the research projects. I would like to thank NSF DMS 1913144 for partial support.

Chapter 3, in part, is a reprint of the material Zirui Zhang and Li-Tien Cheng. "Binary Level Set Method for Variational Implicit Solvation Model" (2021). The dissertation author was the primary investigator and author of the material.

Chapter 4, in part, is a reprint of the material Zirui Zhang, Clarisse G. Ricci, Chao Fan, Li-Tien Cheng, Bo Li, and J. Andrew McCammon. "Coupling Monte Carlo, Variational Implicit Solvation, and Binary Level-Set for Simulations of Biomolecular Binding". *Journal of Chemical Theory and Computation* (2021). The dissertation author was the primary investigator and author of the material

Chapter 5, in part, is a reprint of the material Zirui Zhang and Li-Tien Cheng. "A Compact Coupling Interface Method with Accurate Gradient Approximation for Elliptic Interface Problems" (2021). The dissertation author was the primary investigator and author of the material.

VITA

2016	B.Eng, Civil Engineering, The University of Hong Kong
2019	M.Sc., Computational Science and Engineering, University of California San Diego
2022	Ph.D., Mathematics, University of California San Diego

ABSTRACT OF THE DISSERTATION

Fast and Efficient Numerical Methods in Level-Set Variational Implicit Solvent Model

by

Zirui Zhang

Doctor of Philosophy in Mathematics

University of California San Diego, 2022

Professor Li-Tien Cheng, Chair

The level-set method (LS) is a widely-used and powerful tool for capturing moving interfaces under complex dynamics in fields ranging from two-phase flow to image segmentation. It has recently been successfully applied, in the Variational Implicit Solvent Model (VISM), to find the "shape" of a biomolecule, the interface separating the solute atoms of a biomolecule from the surrounding solvent. In the introduces fast and efficient numerical methods for the application of the level-set method to VISM (LS-VISM), and can be boiled down to two major contributions. The first of these involves the implementation and analysis of a more discrete binary level-set method in LS-VISM that replaces traditional continuous level-set functions with binary ones, and traditional partial differential equation

solvers with discrete "flips" that minimize the free energy of the system. This results in vast improvements in speed, with runtime decreasing from hours to seconds, which ultimately allowed for its application in Monte Carlo simulations of the protein binding process. The second contribution in my dissertation involves the construction and analysis of the Compact Coupling Interface Method (CCIM), a finite difference method for elliptic interface problems with interfacial jump conditions. This method is able to robustly and accurately calculate values of not only the solution but its derivative as well, which ultimately allows for the accurate handling of electrostatic contributions of the solute and solvent in LS-VISM, which take this form as linearized Poisson-Boltzmann equations with discontinuous dielectric constants across the interface.

Chapter 1

Introduction

We present two fast and efficient numerical methods related to the Level Set - Variational Implicit Solvent Model (LS-VISM). The first is a binary level set method for VISM. The second is the Compact Coupling Interface Method (CCIM) for elliptic interface boundary value problems.

In Chapter 2, we introduce the biophysical background of the Variational Implicit Solvent Model, which is a theoretical and computational tool to study biomolecular systems with complex topology. Central in VISM is an effective free energy of all possible interfaces separating solutes (e.g., proteins) from solvent (e.g., water). We briefly review the level set method (LS), which can be used to minimize the VISM energy functional numerically to determine the stable equilibrium interface and solvation free energies. We point out the challenges of LS-VISM and the motivation for the two new numerical methods in this dissertation.

In Chapter 3, we introduce the binary level set method and apply it to LS-VISM. Instead of representing the interface as a zero level set of a continuous function, we approximate the interface by a binary level set function that only takes values ± 1 on the solute or solvent region. An important ingredient is approximating the surface area by convolution of an indicator function with a compactly supported kernel. We prove the formula and analyze the discretization error. By discretizing the energy functional

using midpoint rule, we obtain a discrete formulation of VISM energy. The energy can be minimized by iteratively "flipping" the binary level set function in a steepest descent fashion. Numerical results are given to show that binary level set methods can be more than 100 times faster in estimating the solvation energy.

In Chapter 4, we combine the fast binary level set - VISM and Monte Carlo (MC) to study binding of biomolecule. In MC simulation of protein binding, we need to intensively sample the energy landscape of the system, and the solvation energy needs to be evaluated millions of time. This is only possible with our fast binary level set method, which can obtain the energy in seconds instead of minutes compared with the classical continuous level set method. We apply our method to the protein pair p53-MDM2. Our method successfully capture some configurations before the final bound state of the system.

In Chapter 5, we describe and derive the Compact Coupling Interface Method (CCIM), which is a finite different method for the elliptic interface problem with interfacial jump conditions. Our method can calculate solution values and their derivatives up to second-order accuracy in arbitrary ambient space dimensions. Numerical results on various geometric shapes and on complex protein shapes in three dimensions demonstrate the efficacy, accuracy and robustness of our method.

Chapter 2

Level Set - Variational Implicit Solvent Model

2.1 Variational Implicit Solvent Model

Water is the ubiquitous solvent of life and an important player in may biomolecular processes, including protein folding and protein binding [24, 1]. One common method to study and capture the effect of water is the explicit solvent molecular dynamics (MD) simulations. In these simulations, the trajectories of each individual water molecules are computed explicitly. MD simulations are accurate but computationally expensive, as the number of water molecules can be orders of magnitude larger than the protein.

In contrast to explicit models of water, in an implicit solvent model, water is treated as an continuum medium, which is separated from the solute by the solute-solvent interface. Common implicit solvent model rely on a predefined interface that only make use of properties and coordinates of the solute atoms. For example, the solvent accessible interface (SAS) is defined by rolling a ball over the surface of the molecule. Although these pre-established interface can be useful in estimating the solvation energy, they fail to capture some important behaviors that are observed in explicit simulations, such as polymodal hydration - in which the interfaces fluctuate between different states, and dewetting - in which water are completely excluded in the "pocket" of a biomolecule.

Instead of "guessing" where is the interface, in the Variational Implicit-Solvent

Model (VISM) [12, 11], an energy functional is defined over all the interfaces that enclose the molecule. The VISM energy consists of surface energy, van der Walls interaction energy, and electrostatic energy. The solute-solvent interface is the minimizer of the energy functional. Therefore, the effect of different component of the energy are coupled and the interface is the output of the theory.

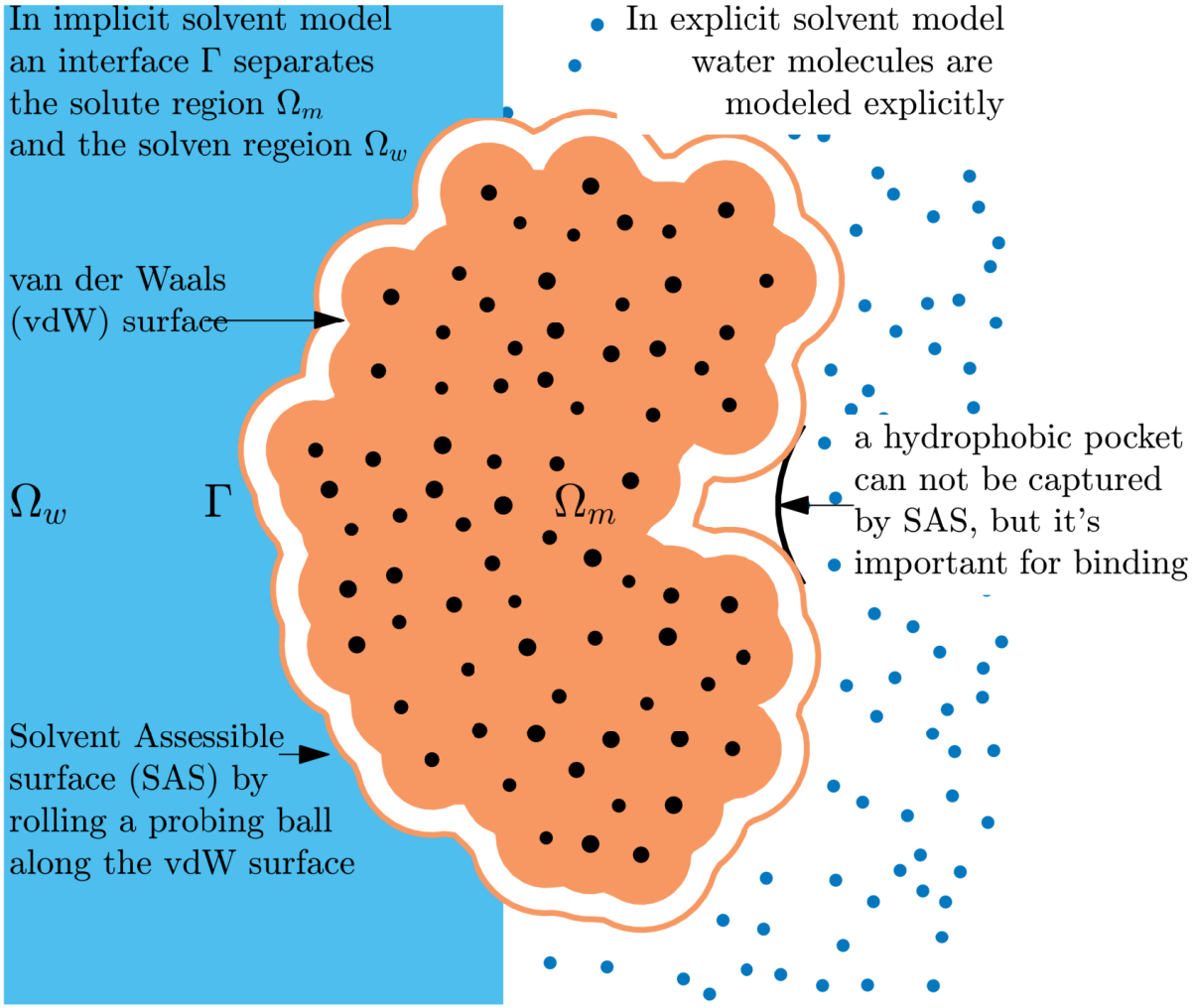


Figure 2.1. Explicit and implicit solvent model. The solute region, solvent region, and solute-solvent interface are denoted by Ω_m, Ω_w, and Γ

The VISM solvation free-energy functional of all possible solute-solvent interfaces Γ is defined as follows

$$G[\Gamma] = G_{\text{surf}}[\Gamma] + G_{\text{vdW}}[\Gamma] + G_{\text{elec}}[\Gamma] \tag{2.1}$$

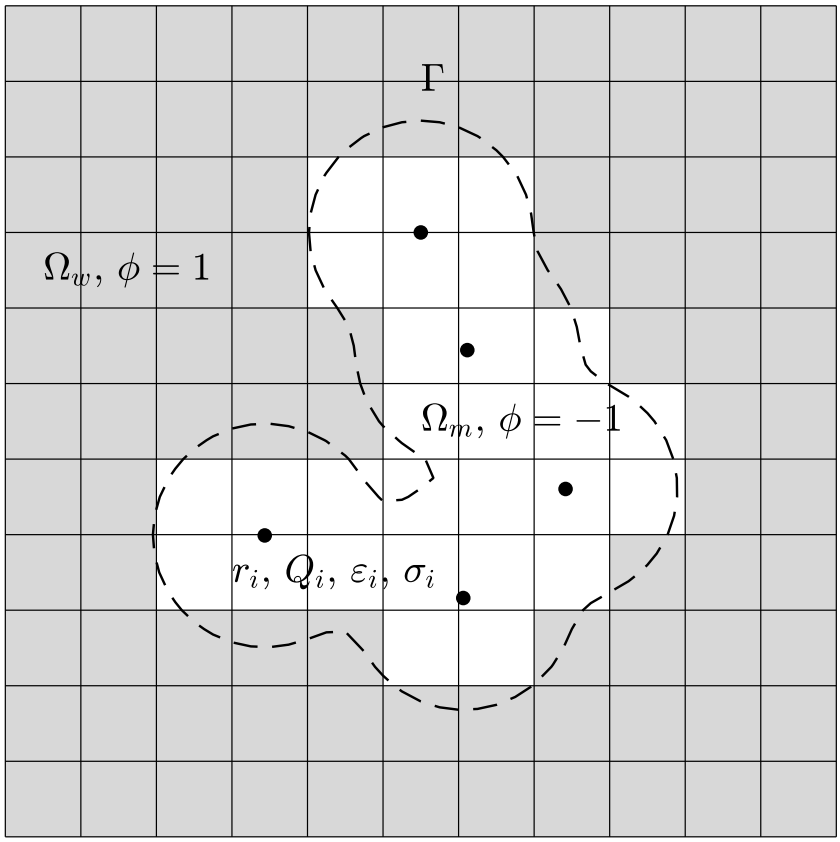


Figure 2.2. Schematic view of a molecular system with implicit solvent. The atoms are located at $\mathbf{r}_i$ with charge Q_i and LJ parameter σ_i and ε_i. An interface Γ (dashed line) separates the solvent region Ω_w from the solute region Ω_m. In continuous level set method, Γ is represented by the zero level set of a function. In the binary level set method, the computational domain is discretized in to grid cell. $\phi = -1$ for grid cells inside the solute region (white) and $\phi = 1$ for grid cells in the solvent region (grey)

Here, the first term is the surface surface energy

$$G_{\text{surf}}[\Gamma] = \int_{\Gamma} \gamma \, dS \tag{2.2}$$

where γ is the interfacial surface tension. In general, it's assumed that $\gamma(\mathbf{x}) = \gamma_0(1-\tau H(\mathbf{x}))$, where γ_0 is the constant surface tension of a planar liquid-vapor interface, τ is the curvature correction coefficient called the Tolman length [12] and H is the mean curvature of the interface.

The second term is the van der Waals (vdW) type interaction energy between the solute and the solvent

$$G_{\text{vdW}}[\Gamma] = \rho_w \sum_{i=1}^{N} \int_{\Omega_{\text{w}}} U_i(|\mathbf{x} - \mathbf{r}_i|) dV. \tag{2.3}$$

Here ρ_w is the water density. U_i is the Lennard-Jones (LJ) potential for atom i

$$U_i(r) = 4\epsilon_i \left[\left(\frac{\sigma_i}{r}\right)^{12} - \left(\frac{\sigma_i}{r}\right)^{6} \right], \tag{2.4}$$

where ϵ_i and σ_i are the energy and length parameter of atom i.

We consider two different formulations of the electrostatic energy $G_{\text{elec}}[\Gamma]$: the Coulomb Field Approximation (CFA) and the Poisson-Boltzmann theory. The CFA allows for a simple analytical formula for the effective boundary force, which is used as the normal velocity in the level set numerical optimization. However, it does not describe the effect of ionic charges in the solvent. The PB theory is a well-established continuum description of electrostatic interactions of biomolecules in an aqueous solvent. However, we need to solve the PB equation with complex dielectric boundary, and the effective boundary force depends on the jump in the gradient of the solution.

In the Coulomb Field Approximation (CFA) [35], the electrostatic energy is given

by

$$G_{\text{elec}}^{\text{CFA}}[\Gamma] = \frac{1}{32\pi^2\epsilon_0}\left(\frac{1}{\epsilon_w} - \frac{1}{\epsilon_m}\right)\int_{\Omega_{\text{w}}}\left|\sum_{i=1}^{N}\frac{Q_i(\mathbf{x}-\mathbf{r}_i)}{|\mathbf{x}-\mathbf{r}_i|^3}\right|^2 dV. \tag{2.5}$$

Here the solute atoms carry partial charges Q_i's. ε_0 is the vacuum permittivity. The relative dielectric permittivities of the solute and solvent regions are denoted by ε_m and ε_w.

Alternatively, in the Poisson-Boltzmann (PM) theory, the electrostatic energy is given by

$$G_{\text{elec}}^{\text{PB}}[\Gamma] = \frac{1}{2}\sum_{i=1}^{N} Q_i\Psi_{\text{reac}}(\mathbf{r}_i) - \frac{1}{2}\int_{\Omega_{\text{w}}}\sum_{j=1}^{M} q_j c_j^{\infty}\Psi e^{-\beta q_j\psi}dV - \beta^{-1}\int_{\Omega_{\text{w}}}\sum_{j=1}^{M} c_j^{\infty}(e^{-\beta q_j\Psi}-1)dV. \tag{2.6}$$

Here, we assume that there are M ionic species in the solvent. For the j-th ionic species, c_j^{∞} is the bulk concentration, and q_j is the charge. $\beta^{-1} = k_BT$ with k_B the Boltzmann constant and T is the absolute temperature. The first term in (2.6) is the electrostatic potential energy corresponding to the fixed solute charges, and the other terms corresponds to the ions in the solvent. $\Psi = \Psi(\mathbf{x})$ is the electrostatic potential, $\Psi_{\text{reac}} = \Psi - \Psi_{\text{ref}}$ is the reaction field, and

$$\Psi_{\text{ref}}(\mathbf{x}) = \sum_{i=1}^{N}\frac{Q_i}{4\pi\varepsilon_0\varepsilon_m|\mathbf{x}-\mathbf{r}_i|}, \tag{2.7}$$

which is the potential for the reference state, with ε_0 the vacuum permittivity and ε_m the dielectric coefficient of the solute region.

The electrostatic potential Ψ solves the boundary-value problem of the PB equation

$$\begin{cases} -\varepsilon_0\varepsilon_m\nabla\cdot(\nabla\Psi) = \sum_{i=1}^{N} Q_i\delta_{\mathbf{r}_i} & \text{in } \Omega_{\mathrm{m}} \\ -\varepsilon_0\varepsilon_w\nabla\cdot(\nabla\Psi) = \sum_{j=1}^{M} q_j c_j^{\infty} e^{-\beta q_j \Psi} & \text{in } \Omega_{\mathrm{w}} \\ [\Psi] = \tau & \text{on } \Gamma \\ \left[\varepsilon\frac{\partial\Psi}{\partial n}\right] = 0 & \text{on } \Gamma \\ \Psi = \Psi_0 & \text{on } \partial\Omega \end{cases} \tag{2.8}$$

Here, $\delta_{\mathbf{r}_i}$ is the dirac delta function. $\varepsilon = \varepsilon(\mathbf{x})$ is a piecewise constant function, which takes the value ε_m in Ω_{m} and ε_w in Ω_{w}. We use the notation $[f]$ for the jump of a function f across the interface

$$[f](\mathbf{x}) = \lim_{h\to 0} f(\mathbf{x}+h\mathbf{n}) - f(\mathbf{x}-h\mathbf{n}) \tag{2.9}$$

And Ψ_0 is the Dirichlet boundary condition often used in practice.

$$\Psi_0(\mathbf{x}) = \sum_{i=0}^{N} \frac{Q_i e^{-\kappa|\mathbf{x}-\mathbf{r}_i|}}{4\pi\varepsilon_0\varepsilon_w|\mathbf{x}-\mathbf{r}_i|},\ \kappa = \sqrt{\frac{\varepsilon_0\varepsilon_w}{\sum_{j=1}^{M} c_j^{\infty} q_j^2}} \tag{2.10}$$

where κ is called the inverse Debye length.

Instead of solving for the potential directly, we solve for Ψ_{reac}

$$\begin{cases} -\varepsilon_0\varepsilon_m\nabla\cdot(\nabla\Psi_{\mathrm{reac}}) = 0 & \text{in } \Omega_{\mathrm{m}} \\ -\varepsilon_0\varepsilon_w\nabla\cdot(\nabla\Psi_{\mathrm{reac}}) = \sum_{j=1}^{M} q_j c_j^{\infty} e^{-\beta q_j(\Psi_{\mathrm{reac}}+\Psi_{\mathrm{ref}}} & \text{in } \Omega_{\mathrm{w}} \\ [\Psi_{\mathrm{reac}}] = 0 & \text{on } \Gamma \\ \left[\varepsilon\frac{\partial\Psi_{\mathrm{reac}}}{\partial n}\right] = [\varepsilon]\frac{\partial\Psi_{\mathrm{ref}}}{\partial n} & \text{on } \Gamma \\ \Psi_{\mathrm{reac}} = \Psi_0 - \Psi_{\mathrm{ref}} & \text{on } \partial\Omega \end{cases} \tag{2.11}$$

Notice that this is still a nonlinear elliptic PDE with interface jump condition. To solve it,

we use Newton's iteration, for $k = 1, \ldots, p$

$$\begin{cases} -\varepsilon_0 \varepsilon_m \nabla \cdot \left(\nabla \Psi_{\text{reac}}^{(k+1)} \right) = 0 & \text{in } \Omega_{\text{m}} \\ -\varepsilon_0 \varepsilon_w \nabla \cdot \left(\nabla \Psi_{\text{reac}}^{(k+1)} \right) = \sum_{j=1}^{M} q_j c_j^{\infty} (1 + \beta q_j \Psi_{\text{reac}}^{(k)}) e^{-\beta q_j (\Psi_{\text{reac}}^{(k)} + \Psi_{\text{ref}})} & \text{in } \Omega_{\text{w}} \\ \left[\Psi_{\text{reac}}^{(k+1)} \right] = 0 & \text{on } \Gamma \\ \left[\varepsilon \frac{\partial \Psi_{\text{reac}}^{(k+1)}}{\partial n} \right] = [\varepsilon] \frac{\partial \Psi_{\text{ref}}}{\partial n} & \text{on } \Gamma \\ \Psi_{\text{reac}}^{(k+1)} = \Psi_0 - \Psi_{\text{ref}} & \text{on } \partial\Omega \end{cases} \tag{2.12}$$

which is a linear elliptic PDE with interface jump condition. PDEs such as (2.12) are called elliptic interface problems. As will be explained section 2.3 later , in order to find the minimizer, we need accurate solution of Ψ and the gradient $\partial\Psi/\partial n$ at the interface. We proposed a finite different method for elliptic interface problem in 5 that is second order accurate in both the solution and the gradient at the interface.

In Table 2.1, we list the physical constants used in VISM

Table 2.1. The VISM parameters.

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

2.2 Level Set Method

The Level Set Method (LSM) [29, 28] is a widely used numerical method to capture the dynamics of an moving interface. One advantage of the level set method is that it can easily handle topological changes, such as merging and breaking. Some applications includes two-phase fluid flow, crystal growth, shape optimization, image processing, etc.

As shown in Figure 2.3, in LSM, the evolving interface $\Gamma = \Gamma(t)$ is represented as the zero level set of a level set function $\phi = \phi(\mathbf{x}, t)$. The motion of the interface is described by the partial differential equation (PDE) called the level-set equation

$$\phi_t + v_n|\nabla\phi| = 0, \tag{2.13}$$

where $v_n = v_n(\mathbf{x})$ is the normal velocity of the interface. Various geometric quantities can be obtained from the level set function, such as the unit normal vector $\mathbf{n}$, the mean curvature H, and the Gaussian Curvature K:

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|}, \quad H = \frac{1}{2}\nabla\cdot\mathbf{n}, \quad K = \mathbf{n}\cdot\mathrm{adj}(\mathrm{He}(\phi))\mathbf{n}, \tag{2.14}$$

where $\mathrm{He}(\phi)$ is the Hessian matrix of the function ϕ, and $\mathrm{adj}(\mathrm{He}(\phi))$ is the adjoint of the Hessian. The level set function is usually assumed to be Lipschitz continuous, and sometimes it's chosen to be the signed distance function to the interface.

One major advantage of the level set method is that it can easily handle topological changes, such as merging and breaking of two shapes. For example, in Figure 2.3, imagining the whole level set function is lifted up, than we can see the zero level set will break into two circles, and the two circles will shrink and disappear.

2.3 LS-VISM

The VISM energy functional can be minimized using LSM by evolving the the interface in the steepest descent direction of the the VISM energy functional.

$$v_n(\mathbf{x}) = -\delta_\Gamma G[\Gamma] = v_n^{\mathrm{surf}}(\mathbf{x}) + v_n^{\mathrm{vdW}}(\mathbf{x}) + v_n^{\mathrm{elec}}(\mathbf{x}) \tag{2.15}$$

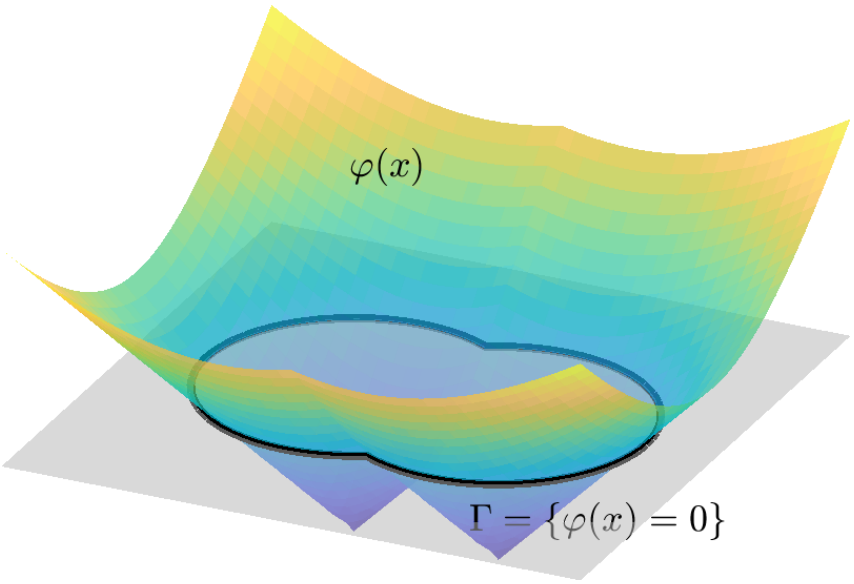


Figure 2.3. Schematics of the level set method

The force due to surface energy is given by

$$v_n^{\text{surf}}(\mathbf{x}) = 2\gamma_0(H(\mathbf{x}) - \tau K(\mathbf{x})) \tag{2.16}$$

where H is the mean curvature and K is the Gaussian curvature.

The force due to vdW energy is given by

$$v_n^{\text{vdW}}(\mathbf{x}) = \rho_w \sum_{i=1}^{N} U_i(|\mathbf{x} - \mathbf{r}_i|) \tag{2.17}$$

The force due to electrostatic energy using CFA is given by

$$v_n^{\text{elec,CFA}}(\mathbf{x}) = \frac{1}{32\pi^2\varepsilon_0}\left(\frac{1}{\varepsilon_w} - \frac{1}{\varepsilon_m}\right)\left|\sum_{i=1}^{N}\frac{Q_i(\mathbf{x} - \mathbf{r}_i)}{|\mathbf{x} - \mathbf{r}_i|^3}\right| \tag{2.18}$$

The force due to electrostatic energy using PB theory is given by

$$v_n^{\text{elec,PB}}(\mathbf{x}) = \frac{\varepsilon_0}{2}\left(\frac{1}{\varepsilon_w} - \frac{1}{\varepsilon_m}\right)\left(\varepsilon\frac{\partial\Psi}{\partial n}\right)^2 + \frac{\varepsilon_0}{2}\left(\varepsilon_m - \varepsilon_w\right)|(I - \mathbf{n}\otimes\mathbf{n})\nabla\Psi|^2 - \beta^{-1}\sum_{j=1}^{M} c_j^{\infty}(e^{-\beta q_j\Psi} - 1) \tag{2.19}$$

where I is the identity matrix. $(I - \mathbf{n}\otimes\mathbf{n})$ is the projection matrix to the tangent plane. Note that $\varepsilon\,\partial\Psi/\partial n$ and $|(I - \mathbf{n}\otimes\mathbf{n})\nabla\Psi|$ are continuous on the interface. In order to have accurate dielectric boundary force, we not only need accurate values of Ψ but also accurate gradient at the interface. This motivates our work in 5, the Compact Coupling Interface Method (CCIM).

2.4 Motivation

We point out the challenges of LS-VISM, and how the numerical methods in the dissertation address those issues. Firstly, LS-VISM is computationally expensive, as we need to solve a PDE to steady state to obtain the solvation energy. Hence it's impractical to apply VISM in some intensive sampling method, such as Monte Carlo method, where the energy needs to be evaluated millions of times. That motivates our binary level set method in chapter 3, which can be hundreds times faster. And the speedup allow us to apply VISM in Monte Carlo simulation of protein binding in chapter 4.

Secondly, when we used the Poisson-Boltzmann theory to describe the electrostatic energy, the interfacial velocity in LS-VISM depends on the jump in the derivative of the solution to an elliptic boundary value problem with interfacial jump conditions. Thus, we need accurate solution and gradient so that the evolving interface is accurate. In addition, in the context of our application, the interface will have complex shapes following the biomolecule, and the grid might be resolve all the detail of the interface. Therefore, the finite difference scheme needs to be compact and robust. Our Compact Coupling Interface

Method (CCIM) can approximate the gradient accurately and the convergence result is robust even for complex shapes.

Chapter 3

Binary Level Set Method for Variational Implicit Solvent Model

3.1 Introduction

In this chapter, we consider the a simpler model of surface energy, where the surface tension is a constant $\gamma = \gamma_0$, so the surface energy is

$$G_{\text{surf}}[\Gamma] = \gamma_0 \,\text{Area}(\Gamma) \tag{3.1}$$

We also use the CFA formulation of electrostatics. Our energy has this form

$$G[\Gamma] = \gamma_0 \,\text{Area}(\Gamma) + \int_{\Omega_{\text{w}}} U(\mathbf{x}) dV \tag{3.2}$$

The second term is a volume integral in the water region.

3.1.1 Binary Level Set Method

In the Binary Level Set Method (BLS), we approximate the interface by a binary level set function, which is -1 in the solute region Ω_{m} and 1 in the solvent region Ω_{w}. Compared with a continuous level set function, we no longer a smooth representation of the interface. And it's difficult to obtain accurate and convergent geometric quantities such as the normal vector and the curvature.

The idea of BLS is introduced to minimize the Mumford Shah functional in image segmentation problems. Consider a two-dimensional image u_0 defined on the image domain Ω, and suppose we want to approximate u_0 by a two-phase image, which is a piecewise constant function that takes value c_m in Ω_{m} and c_w in Ω_{w}. Here we use Ω_{w} and Ω_{m} as some partition of the image $\Omega = \Omega_{\mathrm{w}} \cup \Omega_{\mathrm{m}}$, without the physical meaning of solvent and solute region. The segmenting curve Γ can be represented as the zero level set of a continuous level set function ϕ. $\Omega_{\mathrm{w}} = \{\mathbf{x} : \mathcal{H}(\phi(\mathbf{x})) = 1\}$, and similarly for Ω_{m}. The following "energy" is minimized

$$F(\phi, c_m, c_w) = \mu \int_\Omega |\nabla \mathcal{H}(\phi)| d\Omega + \lambda_w \int_\Omega (u_0 - c_w)^2 \mathcal{H}(\phi) d\Omega + \lambda_m \int_\Omega (u_0 - c_m)^2 (1 - \mathcal{H}(\phi)) d\Omega. \tag{3.3}$$

In the level set formulation, $\mathrm{length}(\Gamma) = \int_\Omega |\nabla \mathcal{H}(\phi)| d\Omega$. μ, λ_w, and λ_m are parameters provided by the user.

Notice that energy (3.3) is similar to our VISM functional (3.2). They both includes the length/area of the curve/surface, and integral in the inside or outside region. In image segmentation problem, the length is an hyper-parameter that serves are a regularizing term [14]. For an image without noise, we can even set $\mu = 0$, as the length scale is not important. However, in our energy functional (3.2), we need to approximate the surface area accurately, so that the surface energy component is accurate.

To minimize the piecewise constant Mumford-Shah functional (3.3), one way is to formulate it as a constrained optimization problem, with the constrain $\phi^2 = 1$ that forces ϕ to be a binary level set function, then apply the projected Lagrangian method or the Augmented Lagrangian method [25]. In [33], the energy is minimized directly by "flipping" the value of the binary level set function in a steepest descent direction. This approach is similar to our method of minimizing the VISM energy functional. However, the length of

the curve is approximated in an ad-hoc way:

$$\int_{\Omega} |\nabla \mathcal{H}(\phi)| d\Omega \approx \sum_{i,j} \sqrt{\left(H(\phi_{i+1,j}) - H(\phi_{i,j})\right)^2 + \left(H(\phi_{i,j+1}) - H(\phi_{i,j})\right)^2} \tag{3.4}$$

Essentially the formula approximate the curve by the edges or diagonal of the grid cells of length 1 or $\sqrt{2}$. And the formula do not converge to the length of the curve. Hence, it remains to devise a formula to approximate the surface area of an interface defined by a binary level set function.

In [34, 26], the following expression is used to approximate the interfacial area between two region:

$$P_{\delta t}(\Omega) = \sqrt{\frac{\pi}{\delta t}} \int_{\Omega^c} G_{\delta t} * \mathbf{1}_{\Omega} dx, \tag{3.5}$$

where

$$G_{\delta t}(\mathbf{x}) = \frac{1}{4\pi\delta t} \exp\left(-\frac{|\mathbf{x}|^2}{4\delta t}\right) \tag{3.6}$$

is the heat kernel. Formula (3.5) is first convolving the heat kernel $G_{\delta t}$ with the indicator function of Ω, then integrating in Ω^c. The integral measures the amount of heat that escapes out of Ω in a short period of time, and it's know that as δt converge to 0, the expression converge to the perimeter of a regular set [20].

In this work, we use the following proposition. We detailed the derivation in section 3.5.1.

Proposition 1 (Integral formula of surface area)**.** *Let $\Gamma = \partial\Omega$ be a compact smooth hypersurface in $\mathbb{R}^d$, K be a continuous, radially symmetric, compact kernel with unit radius. Then for $0 < r \ll 1$*

$$\text{Area}(\Gamma) = C_{K,r,d} \int_{\mathbf{x}\in\Omega} \int_{\mathbf{y}\in\Omega^c} K\left(\frac{|\mathbf{x}-\mathbf{y}|}{r}\right) d\mathbf{y} d\mathbf{x} + \mathcal{O}(r^2), \tag{3.7}$$

where

$$C_{K,r,d} = \left(r^{d+1} C_d \int_0^1 K(r) r^d dr \right)^{-1}, \quad C_d = \frac{2\pi^{\frac{d-1}{2}}}{(d-1)\Gamma(\frac{d-1}{2})} \tag{3.8}$$

Here, $\Gamma(\cdot)$ *in* C_d *is the Gamma function.*

3.2 Numerical Method

3.2.1 Discretization

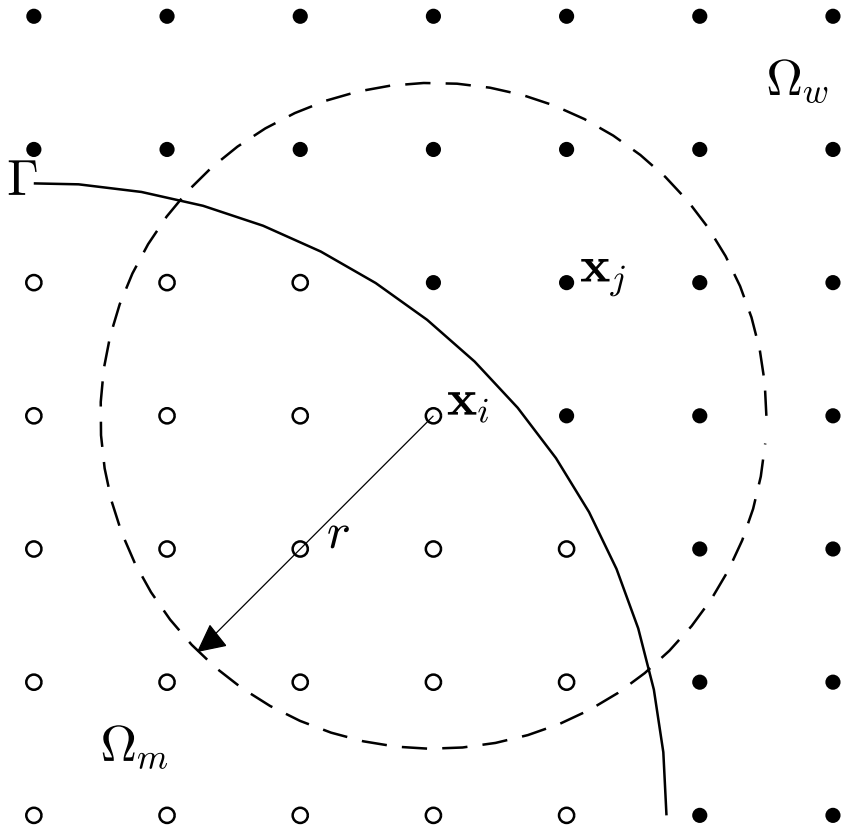


Figure 3.1. Illustration of a scaled kernel centered at the center $\mathbf{x}_i$ of a grid cell and vanishing outside a sphere (indicated by the broken lines). Black dots represent centers of grid cells in the solvent region Ω_w and circles represent the centers of grid cells in the solute region Ω_m.

Using our integral formulation of the surface area (3.7), every term in the VISM functional (2.1) is a volume integral and can be approximated by the midpoint rule. For the surface area (3.7), we need to scale the kernel radius r correctly. For instance, if we

choose $r \sim h$, then as h goes to 0, we are "zooming in" to the interface and in each kernel, the interface will become essentially flat. The correct scaling is that $r \sim \sqrt{h}$ and we prove the following proposition in section 3.5.2.

Proposition 2 (Numerical approximation of surface area)**.** *If K is twice continuously differentiable in* (3.7)*, and $r \sim \sqrt{h}$, then*

$$\text{Area}[\Gamma] = C_{K,r,d}\, h^6 \sum_{\mathbf{x}_i \in \Omega} \sum_{\substack{\mathbf{x}_j \in \Omega^c \\ |\mathbf{x}_j - \mathbf{x}_i| \le r}} K\left(\frac{|\mathbf{x}_i - \mathbf{x}_j|}{r}\right) + \mathcal{O}(h) \tag{3.9}$$

Therefore, the surface energy has the following approximation

$$G_{surf}[\Gamma] = \sum_{\mathbf{x}_i \in \Omega_{\mathrm{m}}} \sum_{\substack{\mathbf{x}_j \in \Omega_{\mathrm{w}} \\ |\mathbf{x}_j - \mathbf{x}_i| \le r}} K_{ij} + \mathcal{O}(h) \tag{3.10}$$

where

$$K_{ij} = \gamma_0 C_{K,r,d} h^6 K\left(\frac{|\mathbf{x}_i - \mathbf{x}_j|}{r}\right). \tag{3.11}$$

In words, we go through the grid points $\mathbf{x}_i$ in Ω_{m}, put a kernel centered at $\mathbf{x}_i$, and sum up the part of the kernel in Ω_{w}, as illustrated in Figure 3.1.

The volume integrals $G_{\mathrm{vdW}}[\Gamma]$ and $G_{\mathrm{elec}}[\Gamma]$ can also be approximated by the midpoint rule:

$$G_{\mathrm{vdW}}[\Gamma] = \sum_{x_i \in \Omega_{\mathrm{w}}} (G_{\mathrm{vdW}})_i + \mathcal{O}(h) \tag{3.12}$$

where

$$(G_{\mathrm{vdW}})_i = \rho_w \sum_{j=1}^{N} U_j(|\mathbf{x}_i - \mathbf{r}_j|) h^3 \tag{3.13}$$

Similarly, for the electrostatic energy

$$G_{\mathrm{elec}}[\Gamma] = \sum_{\mathbf{x}_i \in \Omega_{\mathrm{w}}} (G_{\mathrm{elec}})_i + \mathcal{O}(h), \tag{3.14}$$

where

$$(G_{\text{elec}})_i = \frac{1}{32\pi^2\epsilon_0}\left(\frac{1}{\epsilon_w}-\frac{1}{\epsilon_m}\right)\left|\sum_{j=1}^{N}\frac{Q_j(\mathbf{x}_i-r_j)}{|\mathbf{x}_i-\mathbf{r}_j|^3}\right|^2 h^3 \tag{3.15}$$

Here, we can think of $(G_{\text{vdW}})_i$ and $(G_{\text{elec}})_i$ as the contribution to the vdW and electrostatic energy from the grid cell centered at $\mathbf{x}_i$ when it's filled with water.

The discrete VISM total energy is given by

$$G^{\text{disc}}_{\text{VISM}}[\Omega_{\text{m}}] = \sum_{x_i\in\Omega_{\text{m}}}\sum_{\substack{x_j\in\Omega_{\text{w}}\\ |x_j-x_i|<r}} K_{ij} + \sum_{x_i\in\Omega_{\text{w}}}(G_{\text{vdW}})_i + (G_{\text{elec}})_i, \tag{3.16}$$

which is determined by the configuration of the grid cells classified to be in the solute region Ω_{m}.

A useful quantity in our minimization algorithm is the energy change ΔG_i if the value of ϕ is flipped: if $\mathbf{x}_i \in \Omega_{\text{m}}$,

$$\Delta G_i = \sum_{\substack{\mathbf{x}_j\in\Omega_{\text{w}}\\ |x_j-x_i|<r}} K_{ij} - \sum_{\substack{\mathbf{x}_j\in\Omega_{\text{m}}\\ |\mathbf{x}_j-\mathbf{x}_i|<r}} K_{ij} + (G_{\text{vdW}})_i + (G_{\text{elec}})_i. \tag{3.17}$$

Here, the first two terms compute the difference of the kernel in Ω_{w} and Ω_{m}. The third and the forth terms are the contribution to the vdW energy and electrostatic energy if we fill the cell with water. If $\mathbf{x}_i \in \Omega_{\text{w}}$, the change will be $-\Delta G_i$. Note that if $\mathbf{x}_i$ is flipped, then the ΔG_j of the neighboring grids ($\mathbf{x}_j$ with $|\mathbf{x}_j-\mathbf{x}_i|<r$) needs to be updated because of the compactness of the kernel. More specifically, suppose we flip one grid cell et each step of our algorithm, and let $\Delta G_i^{(k)}$ be the energy change if ϕ_i is flipped at the step k, then

$$\Delta G_j^{(k+1)} = \begin{cases} -\Delta G_j^{(k)} & j=i \\ \Delta G_j^{(k)} - \phi_i\phi_j 2K_{ij} & j\neq i, |\mathbf{x}_j-\mathbf{x}_i|<r \\ \Delta G_j^{(k)} & \text{otherwise} \end{cases} \tag{3.18}$$

Notice that the work to perform the update is proportional to the number of grid cell covered by the kernel.

Integration outside of computation box

For details on how to compute the integral $G_{\text{vdW}}[\Gamma]$ and $G_{\text{elec}}[\Gamma]$ outside of the computational box, see [8]. The idea is to partition the region outside of the computational box Ω^c, write the integral in different partition in cylindrical or spherical coordinate, integrate analytically in two of the dimensions, and finally compute the one-dimensional integral with composite Simpson's rule.

Here we discuss another method that make use of parallel computing. Let Ω be a cube of equal side in $\mathbb{R}^3$, centered at the origin, and S be the sphere inscribed in Ω. Then we can write the integral in spherical coordinate

$$\begin{aligned}\int_{\Omega_c} f(x)dx &= \int_{S^c} \mathbf{1}_\Omega(x,y,z)f(x,y,z)dxdydz \\ &= \int_0^\pi \int_0^{2\pi} \int_R^\infty \hat{\mathbf{1}}_\Omega(\theta,\varphi,r)\hat{f}(\theta,\varphi,r)r^2\sin\theta dr d\theta d\varphi \\ &= \int_0^\pi \int_0^{2\pi} \int_0^{1/R} \hat{\mathbf{1}}_\Omega(\theta,\varphi,\frac{1}{\rho})\hat{f}(\theta,\varphi,\frac{1}{\rho})\rho^{-4}\sin\theta d\rho d\theta d\varphi\end{aligned} \tag{3.19}$$

In the second equality, we change from Cartesian coordinate to Polar coordinate. In the last equality, we perform a change of variable $\rho = r^{-1}$.

We apply the mid point rule

$$\int_{\Omega_c} f(x)dx \approx \sum_{i,j,k} \hat{\mathbf{1}}_\Omega(\theta_j,\varphi_k,\frac{1}{\rho_i})\hat{f}(\theta_j,\varphi_k,\frac{1}{\rho_i})\rho^{-4}h^3 + \mathcal{O}(h) \tag{3.20}$$

Note that the summand can be computed in parallel. Through experiment we found that our implementation of this method (using OpenCL and OpenMP) achieve desirable accuracy in shorter time compared with the method in [8].

3.2.2 Algorithm and Implementation Detail

In this section, we first outline our algorithm, and then we discussion some computation details of the algorithm.

We minimize the discrete VISM energy (3.16) in a steepest descent fashion by iteratively "flipping" the ϕ value. Since we need to repeatedly find the grid cell with the smallest negative ΔG_i, we use the min-heap data structure, which takes logarithmic time to retrieve the smallest element. The element of the heap is the pair $(\Delta G_i,\, i)$. The following is the outline of the algorithm.

Algorithm 3.2.1: Binary Level Set - VISM algorithm

input : γ_0, ρ_w, ε_0, ε_m, ε_w, and atomic parameters $\mathbf{r}_i$, Q_i, ε_i, and σ_i, for all $i = 1, \ldots, N$. Choose a computational box $[-a, a]^3$ according to the atomic coordinates and discretize the box uniformly with the prescribed computational grid size h.

1 Initialize the kernel function and the binary level set function. Construct the collection of interface points $\mathcal{I}$

2 Compute and store $(G_{\text{vdW}})_i$ (3.13) and $(G_{\text{elec}})_i$ (3.15) at grid cells i.

3 Compute ΔG_i (3.17) for all $i \in \mathcal{I}$. Insert the pair $(i,\, \Delta G_i)$ to the heap data structure $\mathcal{H}$ if $\Delta G_i < 0$

4 **while** $\min_{i \in \mathcal{H}} \Delta G_i < 0$ **do**

5 | Extract and remove the minimum value $(i,\Delta G_i)$, flip ϕ_i

6 | Update ΔG_j (3.18) at the neighboring center point $\mathbf{x}_j$ with $|\mathbf{x}_j - \mathbf{x}_i| \le r$.

7 | Update $\mathcal{I}$ and $\mathcal{H}$

8 **end**

As the flipping proceeds, the total energy of the system decreases monotonically. In the end, we reach a local minimum where there is no single flipping that can decrease the energy. However, there might be simultaneous flipping that can further reduce the energy. In fact, the global minimum can be found by min-cut max-flow algorithm, as explained in Appendix 3.2.3, though it's much more time consuming.

There are different ways to carry out the flipping procedure. In Jacobi iteration, one goes through all the grid cell and flips all the grid cells with negative ΔG together, and then compute their updated ΔG. In a Gauss-Seidel iteration, one flips the grid cell

one by one [33] and update ΔG immediately. Here we are taking a flow-based approach, aiming to find the local minimum that is close to the initial guess. From our experience, for simple shapes, these different approaches lead to the same minimum.

Initial Guess

Because the VISM energy functional is non-convex, we may obtain local minimums depending on the initial guess. These local minimizer corresponds to polymodal hydration states. In MD simulations, the interface would fluctuate between those different states, and such fluctuation affects the binding process [30] Therefore, it's desirable to capture different local minimizers. We usually use two types of initial surfaces called "tight" and "loose" initial. The tight initials is the unions of spheres center at the solute atoms. In our computation, we choose the radius of the sphere to be σ_i, so that the tight initial is also the vdW surface. The loose initial is a large surface that loosely encloses all the atoms. By marking all the grid point as solute region, we get a loose initial that is the boundary of the computation box

Efficient Update

We implement our binary heap with some extra information to make our flipping algorithm fast, at the expanse of extra memory use. During the flipping procedure, we only keep elements with negative energy change and indices next to the interface, The aims is to keep the number of elements in the heap as small as possible. As a result, points may be inserted or deleted several times during the whole flipping procedure. Also notice that ΔG_i can be computed in time proportional to the number of points in the kernel, but update in constant time as in (3.18). Therefore, as the interface evolves, we will compute and store ΔG_i whenever the i-th grid cell is first encountered, and update its value afterwards, even though the element may not be in the heap at the current step.

At the beginning, we insert all the interface points with negative energy change into

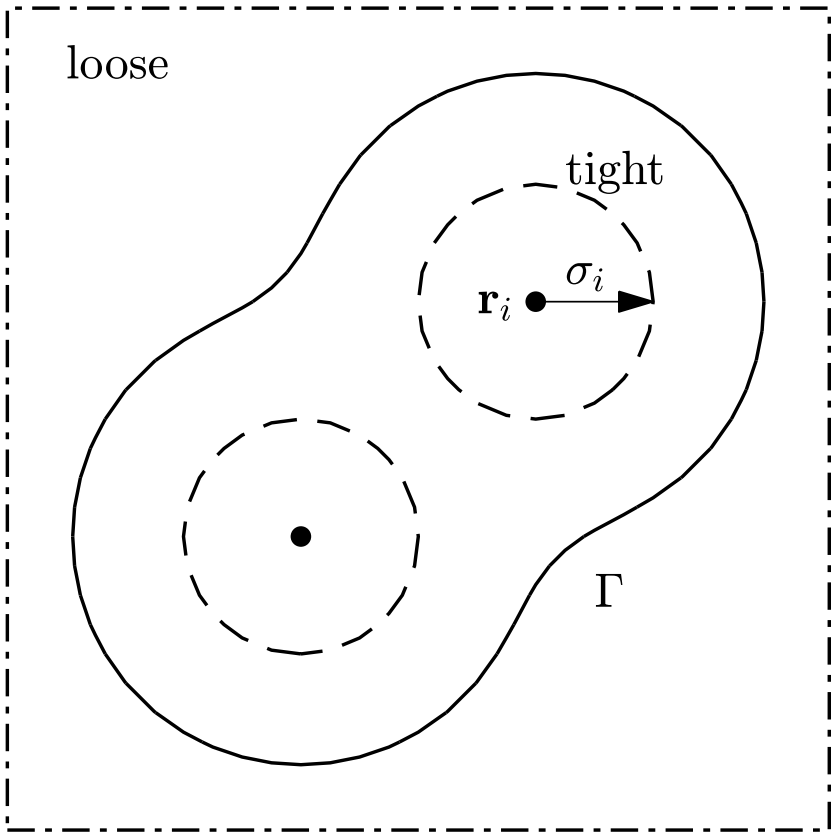


Figure 3.2. Illustration of a tight initial surface (dashed line), a loose initial surface (dash-dotted line), and a VISM relaxed surface (solid line) Γ surrounding the atoms (dots). For loose initial, we set $\phi = -1$ for all grid cells. The tight initial is the union of sphere of radius σ_i centered at $\mathbf{r}_i$.

the heap. Suppose we first remove $(\Delta G_i,\, i)$ and flip the sign of ϕ_i. Then we go through all the neighboring grid points x_j within the kernel, and there are a few different situations:

1. If the $\mathbf{x}_j$ is not an interface point before the flip, but an new interface point afterward, then we compute and store ΔG_j.
2. If the $\mathbf{x}_j$ is an interface point before and after the flip, we update ΔG_j using (3.18).
3. If the $\mathbf{x}_j$ is an interface point before the flip, but no longer an interface afterward, we delete the element from the heap.
4. If the $\mathbf{x}_j$ is not an interface point before and after the flip, nothing is performed.

Here, ΔG_j is stored whenever it is computed or updated, but only inserted into the heap if it is negative.

Parallel Computing

We discuss the implementation of aspects of our free energy functional minimization algorithm using parallel computing. By performing simultaneous computations, we can significantly speed up the calculation of VISM free energy. In our work, we leverage modern graphical processing unit (GPU) and supercomputer clusters.

Both G_{elec} and G_{vdW} are volume integral in Ω_{w} of the following form, which can be approximated using midpoint rule,

$$\int_{\Omega_{\text{w}}} U(\mathbf{R}, \mathbf{x}) d\mathbf{x} = \sum_{\mathbf{x}_i \in \Omega} \mathbf{1}_{\Omega_{\text{w}}}(\mathbf{x}_i) U(\mathbf{R}, \mathbf{x}_i) h^3 + \mathcal{O}(h)$$

Here U depends on the position of all the atoms $\mathbf{R}$, and the summation is over all n^3 grid points in the computational box. This summation is computational intensive as the n is typically between 100 and 200. However, each summand can be evaluated independently and in parallel. In GPU implementation, each grid point form a work-item (OpenCL) or thread (CUDA). In an OpenMP implementation, we can use a parallel for–loop. Similar idea is also used to handle the integration outside the computational box Ω in section

3.2.1.

3.2.3 Max-flow Formulation

We also note that a flow network can be constructed such that the capacity of a graph cut corresponds to the discrete VISM energy of a binary level set function. Therefore, min-cut max-flow algorithms can be used to find the global minimum energy.

Let M be a constant such that $M + G_i^{vdw} + G_i^{elec} \geq 0$ for all x_i. Such a constant exist because $(G_{\text{vdW}})_i + (G_{\text{elec}})_i$ is always bounded below. The purpose is to make sure that the capacity of all the edges in the flow net is non negative. Let every grid point be a node in the graph and denoted by $\mathbf{x}_i \in \Omega$. Connect the source s to $\mathbf{x}_i$ with capacity $G_i^{vdW} + M + G_i^{elec}$, and connect the $\mathbf{x}_i$ to the sink with capacity M. Then for every pair of distinct nodes $\mathbf{x}_i$ and $\mathbf{x}_j$, if $|\mathbf{x}_j - \mathbf{x}_i| < r$, connect them to each other with two directed edges of capacity K_{ij}. An illustration of the flow net is shown in Fig. 3.3. Then any graph cut $(\{s, \Omega_{\text{m}}\}, \{t, \Omega_{\text{w}}\})$ has the capacity

$$\sum_{\mathbf{x}_i \in \Omega_m} \sum_{\substack{\mathbf{x}_j \in \Omega_w \\ |\mathbf{x}_j - \mathbf{x}_i| < r}} K_{ij} + \sum_{\mathbf{x}_i \in \Omega_w} ((G_{\text{vdW}})_i + (G_{\text{elec}})_i + M) + \sum_{\mathbf{x}_i \in \Omega_m} M = G[\Omega_m] + \sum_{\mathbf{x} \in \Omega} M. \tag{3.21}$$

which has the same minimizer as the discrete VISM energy (3.16).

We briefly experimented with the Boykov's min-cut max-flow algorithm [3] on our discrete VISM energy. For simple test cases with one or two atoms, the results are the same as those from our flipping algorithm. For the p53-MDM2 experiment, the global minimum is only slightly below the minimum obtained from flipping with a tight initial. However, min-cut max-flow algorithm is much more time consuming. Hence we didn't make further investigation along this line.

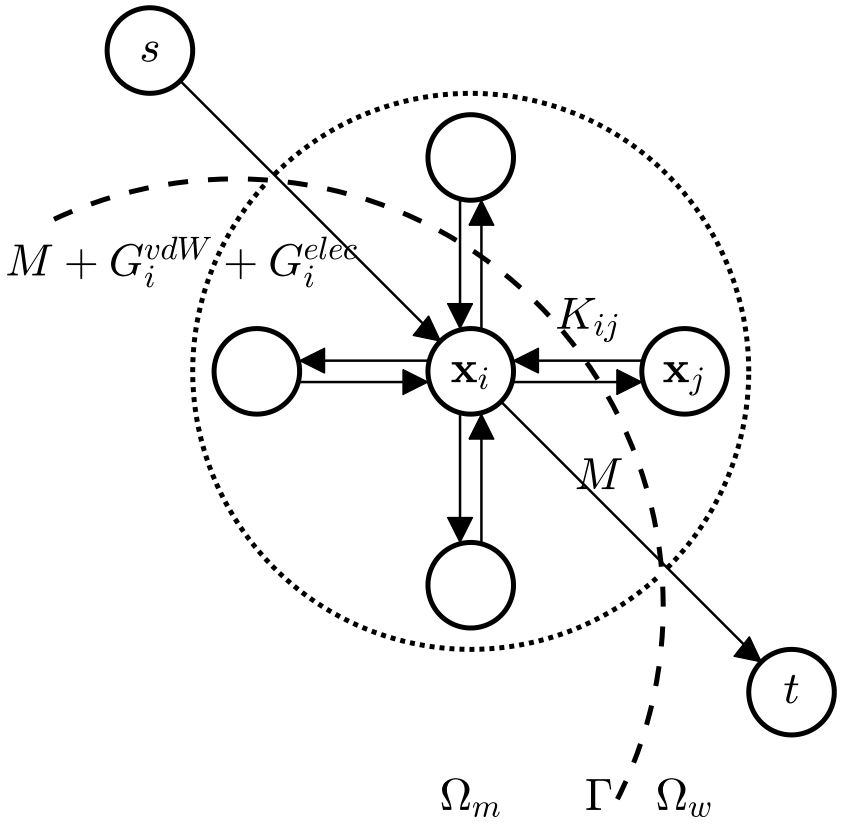


Figure 3.3. Illustration of a flow net whose min-cut corresponds to the minimizing interface.

3.3 Numerical Results

In this section, we present several numerical tests, all in three dimensions. We first test our formula of approximating surface area (3.4) using a sphere. We experiment with different kernel functions and kernel radii. Then we test our method with different systems: one atom, two atoms, and a pair of protein. We mainly compare the new BLS-VISM with the previous CLS-VISM, and demonstrate that the BLS-VISM can achieve similar accuracy as CLS-VISM but in much shorter time. For CLS-VISM calculations, the forward Euler method is used to discretize the time derivative in the level-set equation, and a fifth-order WENO (weighted essential-non-oscillation) scheme is used to discretize the spacial variables. For details, the reader can refer to [35].

For comparison of speed, we only compare the time for flipping in BLS-VISM and solving the time dependent PDE in CLS-VISM. We exclude the time for initialization, which comprises step 1 and 2 in the algorithm, and is common in both BLS-VISM and CLS-VISM. All the tests are performed on a 2017 iMac with 3.5 GHz Intel Core i5 and 16GB memory. We denote n the number of sub-intervals in each dimension of the cubical computation box. So the total number of grid points is $(n+1)^3$.

3.3.1 Surface Area of a Sphere

Here we apply our method to approximate the area of a sphere. Consider a sphere of radius 0.5 in a $[-1,1]^3$ box. Let $r = C\sqrt{h}$. Here we consider 3 different kernel: $K_1(r) = \mathbf{1}_{B_1(0)}$, $K_2(r) = \sin(\pi r)^2$ and $K_3(r) = \cos(\pi r) + 1$ for $0 \le r \le 1$. K_1 is simply the indicator function, which is discontinuous at $r = 1$. Both K_2 and K_3 are C^∞ functions. The parameter C control the size of the kernel. In Fig. 3.4, we briefly look at the performance of different kernel and the effect of C. In both figure, We plot the relative error versus the number of interval n in one edge of the computational box. n ranges from 20 to 200 with increment 5. Each data point is an average of 6 spheres with random centers close to the

origin.

In Fig. 3.4 (L), we fix $C = 3$ and use different kernel function. The approximations are all first order accuracy as the slopes of the least-square lines are close to 1. Even though our analysis require the kernel function to go to zero smoothly at $r = 1$, the indicator function still demonstrate first order accuracy, albeit with larger variance compared with the other two kernel function.

In Fig. 3.4 (R), we use the sin-squared kernel function $K_2(r)$ and look at the effect of different values of C. The parameter C controls the number of grid point in the kernel, which is proportionate to the amount of work for updating the energy change (3.17) or computing the surface area. In three dimensions, the number of grid point in the kernel is of order $\mathcal{O}((C/\sqrt{h})^3)$. Therefore, a smaller value is C is preferable in terms of computation speed. However, as shown in Fig. 3.4 (R), for $C = 5, 3, 1, 0.5, 0.3$ in order, the overall error first decreases and then increases, while variance of the convergence line keep increasing. For $C = 5$ or 3, the data is fitted nicely by a straight line with slope close to 1. However, for $C = 0.5$ or 0.3, the data oscillate wildly as n increases. However, the overall accuracy is still at least first order. Therefore, for demonstration of clear and stable convergence behavior, a relatively larger value of C is preferred.

In practice, we rarely send n to infinity. Instead, we usually focus on a relatively small range of resolution, which is limited by our computation resources and the requirement on overall accuracy. Also the length scale is dictated by the physics of the problem at hand. For example, in our application of BLS-VISM to protein binding simulation in Section 3.3.4, the average radius of the atoms in a protein is around 3 Å and we only consider the grid size around 1 Å for reasonable accuracy and computation time. Then we would choose the parameter C based on numerical experiments with similar setting.

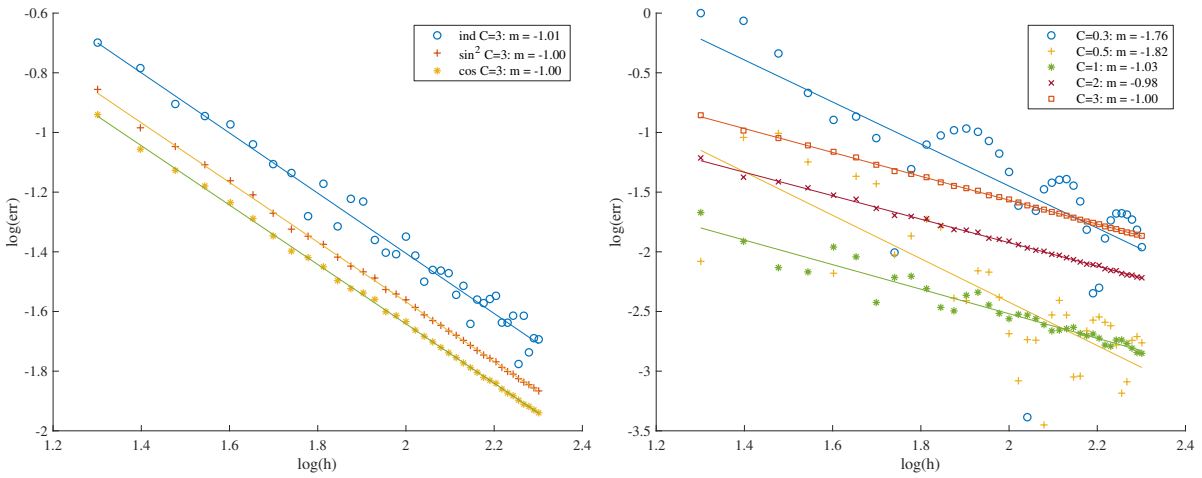


Figure 3.4. log-log plot of the relative error versus the number of interval n in one edge of the computational box. n ranges from 20 to 200 with increment 5. Each data point is an average of 6 spheres with random centers. m is the slope of the line fitted by least-square.

3.3.2 One atom

We consider a single charged atom carrying a partial charge Q. The VISM free-energy functional (2.1) is then a function of the radius R of the spherical solute region:

$$G(R) = 4\pi R^2\gamma_0 + 16\pi\rho_w\varepsilon\left(\frac{\sigma^{12}}{9R^9} - \frac{\sigma^6}{3R^3}\right) + \frac{Q^2}{8\pi\varepsilon_0 R}\left(\frac{1}{\varepsilon_w} - \frac{1}{\varepsilon_m}\right), \tag{3.22}$$

where σ and ε are the LJ parameters between the atom and a water molecule. The function $G(R)$ can be minimized accurately. We use $Q = 1$ and the VISM parameters in Table 2.1 that are close to real systems [7].

In Table 3.1, we show a comparison of the calculation speed between the CLS-VISM and the BLS-VISM. The computation box is $[-5, 5]^3$(unit Å) and the atom is fixed at the origin. The numbers in the parenthesis are the relative error with respect to the exact solution (3.22). Both method are very accurate in terms of total energy, the total error is less than 1% with n = 50. The total energy exhibit first order convergence for BLS and second order for CLS. The individual component might have larger relative error. One reason is that the vdW and electrostatic energy are sensitive to the boundary, as

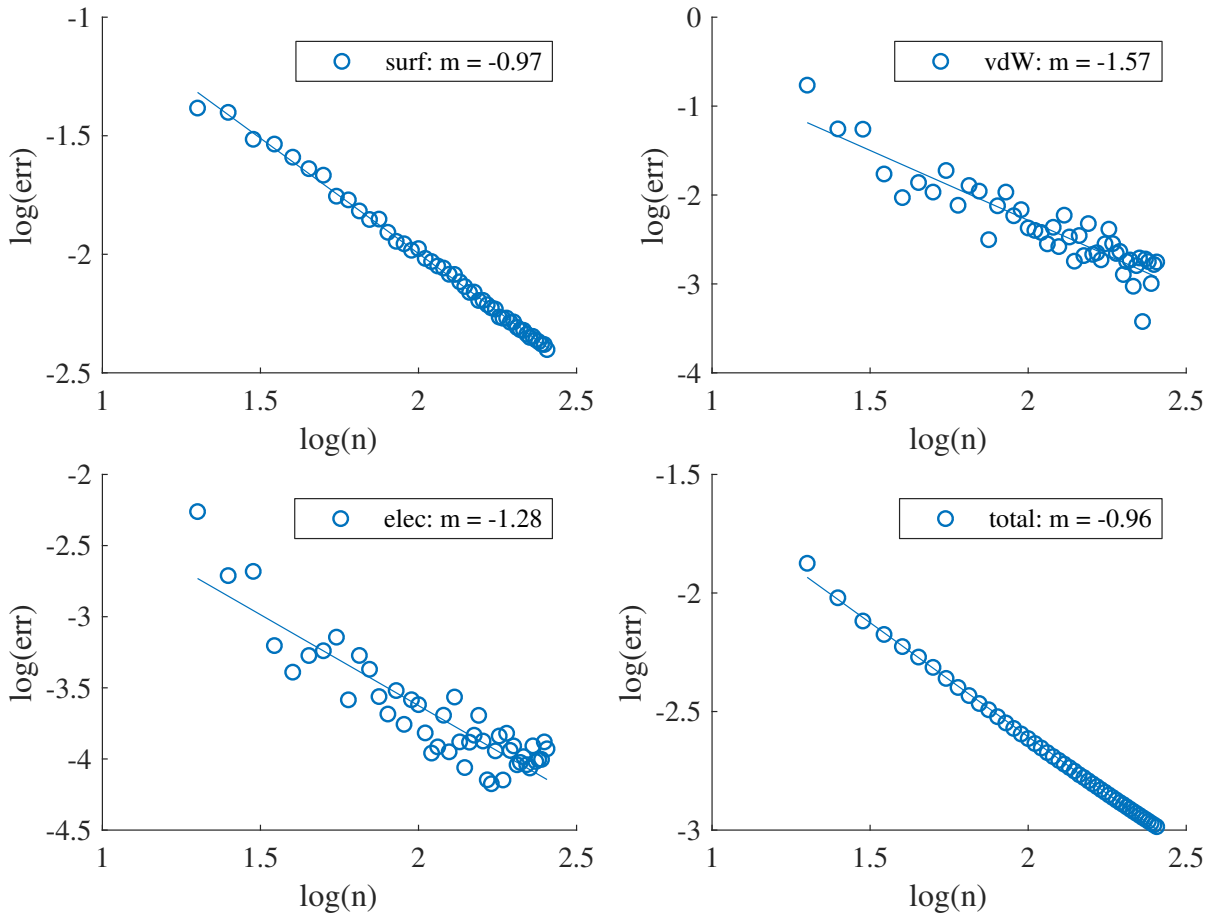


Figure 3.5. log-log plot of the relative error of each component versus the number of interval n in one edge of the computational box. n ranges from 20 to 200 with increment 5. m is the slope of the line fitted by least-square

mentioned before. Another reason is that the energy components have different magnitudes. In this particular example, the electrostatic is much smaller in magnitude. Hence even though the relative error may seems large, it has insignificant impact on the error of total energy. What's remarkable is that the BLS-VISM is about 100 times faster while still fairly accurate for our application.

Table 3.1. Solvation free energy (k_BT) and computation time (s) for different grid numbers. Here, cont. stands for the continuous LSM and binary for the binary LSM. The number in the parenthesis is the relative error with respect to the exact solution (3.22)

n	Surf		vdW		Elec		total		Time	
	CLS	BLS	CLS	BLS	CLS	BLS	CLS	BLS	CLS	BLS
25^3	17.0(0.9%)	17.7(5.4%)	-97.1(1.9%)	-96.2(-2.9%)	2.6(48.0%)	3.9(-24.3%)	-77.5(0.6%)	-74.6(-3.3%)	0.006	3.7
50^3	16.8(0.1%)	17.1(1.8%)	-98.3(0.7%)	-98.2(-0.8%)	4.2(17.1%)	4.6(-9.4%)	-77.4(0.3%)	-76.5(-0.8%)	0.043	7.1
100^3	16.7(0.4%)	16.9(0.7%)	-98.9(0.1%)	-98.6(-0.4%)	4.9(4.0%)	4.8(-6.3%)	-77.2(0.2%)	-77.0(-0.2%)	0.977	103.9
200^3	16.7(0.4%)	16.9(0.5%)	-99.0(0.0%)	-98.8(-0.2%)	5.1(0.8%)	4.8(-5.9%)	-77.2(0.1%)	-77.1(-0.0%)	24.055	2952.1

3.3.3 Two atoms

In this experiment, two atoms are placed at $(-d/2, 0, 0)$ and $(d/2, 0, 0)$ for $d = 4$, 6, 8 in the usual xyz coordinate system. The physical parameters are the same as those in the previous experiment.

The computation box is $[-10, 10]^3$ (unit Å). We choose $n = 100$ for BLS and $n = 50$ for CLS. Because BLS is only first order accurate, a finer grid is needed to achieve similar accuracy as the CLS. In the binary LS-VISM, we choose the kernel radius $r = 3\sqrt{h}$. For both continuous and binary LS-VISM, the tight and loose initial relax to the same final interface and energy. Notice that topological changes are handled easily as the center-to-center distance increases.

In Fig. 3.6, we show the final interfaces from BLS and CLS for different configurations. We also show the cross section of both interfaces overlaid. For the interfaces from BLS, we render the faces of the voxels, which are the little cubes of side length h and centered at the grid points with $\phi = -1$. We can see that the interfaces from BLS closely approximate those from CLS.

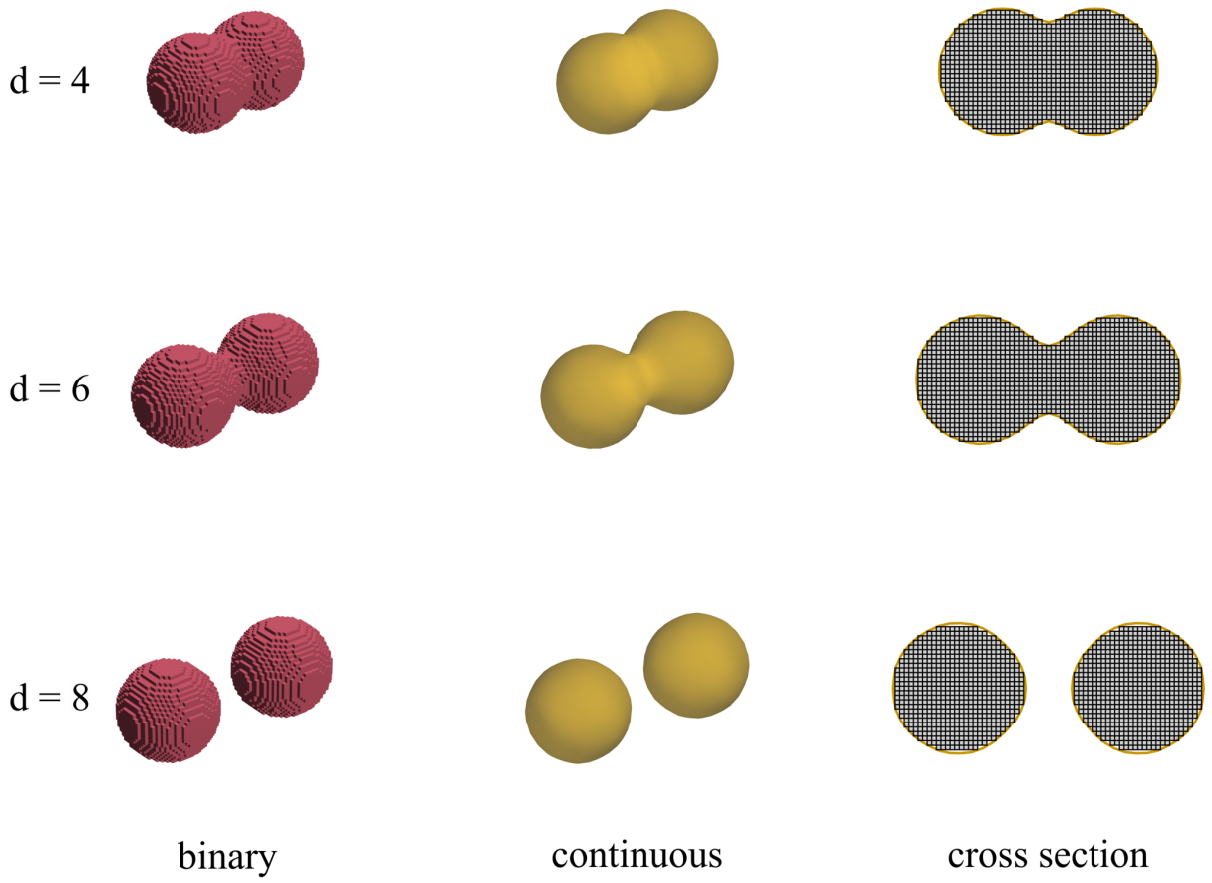


Figure 3.6. Final interfaces of d = 4, 6, 8 Å. Left: BLS-VISM. Middle: CLS-VISM. Right: cross section of both interface at $z = 0$. $n = 100$ for BLS and $n = 50$ for CLS.

In Table 3.2, we show a comparison of the energy and speed between the BLS and CLS, both using tight initial. Again, BLS-VISM can obtain accurate energy and it's hundreds times faster than than CLS-VISM.

Table 3.2. Solvation free energy (k_BT) and computation time (s) for the two atoms system. $n = 100$ for BLS and $n = 50$ for CLS.

d	Surf		vdW		Elec		total		Time	
	BLS	CLS	BLS	CLS	BLS	CLS	BLS	CLS	BLS	CLS
4	27.0	27.8	14.1	12.2	-317.9	-314.3	-276.8	-274.2	0.7	11.4
6	32.2	33.2	10.5	9.8	-285.9	-283.1	-243.3	-240.1	0.7	20.7
8	33.3	34.6	10.1	8.8	-265.6	-262.4	-222.2	-219.0	0.8	5.7

3.3.4 Biomolecules

We apply our method to a complex biomolecular system from the Protein Data Bank (PDB) [2]: p53-MDM2 (PDB ID 1YCR) and BphC (PDB ID 1DHY)[31]. The p53-MDM2 system consists of more than a thousand atoms. It is a relevant pharmacological target for anticancer therapeutics [6]. During their binding process, the binding pocket fluctuates between dry and wet states [16]. The parameters for the atoms comes from the force field in CHARMM36[19]. For both experiment, to generate the positions of the atom, we start with the bounded structure from PDB, and manually pull away the protein pairs along their center-to-center axis for a distance d. Here $d = 0$ corresponds to the bounded state.

In Fig. 3.7 and 3.8, we compare the surfaces obtained from binary LSM and continuous LSM from tight or loose initial surfaces. In both BLS-VISM and CLS-VISM, the tight initial leads to a final state of two disjoint surfaces, which is called wetting as there is water between the two proteins. In contrast, the loose initial results in a final state of one connected interface, which is called dewetting, as there is no water in between. The ability of CLS-VISM to capture the dewetting effects of complex molecules is well-established [35, 41]. Here we demonstrate that the BLS-VISM preserve this characteristic feature of

CLS-VISM.

In Table 3.3, we compare the energy and speed between BLS-VISM and CLS-VISM of the protein systems. For p53-MDM2 with either tight or loose fit, the relative error with respect to CLS-VISM is within 1% for the surface energy, and within 5% for the vdW and electrostatic energies. The relative error is 15% for the total energy. For BphC with either tight or loose fit, the relative error with respect to CLS-VISM is within 3% for the surface energy, and within 13% for the vdW and electrostatic energies. The relative error is 50% for the total energy. As mentioned before, the vdW and electrostatic energies are singular at the position of the atoms, and they are sensitive to the exact location of the interface. And the more atoms there are, the faster these two components go to positive or negative infinity as the surface get closer to the atoms. The larger relative error in the total energy might be due to the cancellation of the large positive and negative energy components when computing the total energy, and this effect is more severe for larger system with more atoms. Nevertheless, due to the complex topology of the solvation interface and the large number of atoms, such discrepancies are expected and remain reasonably accurate when estimating solvation energy of protein systems [38].

What's impressive is the speed at which BLS-VISM compute the solvation energy, which is hundreds or thousands times faster then CLS-VISM. And the speedup is more significant as the number of atoms increases. This is because the work required in each flip of BLS does not depends on the number of atoms. But in CLS-VISM, when evolving the interface, the work required to evaluate the normal velocity (2.15) of the interface depends on the number of atoms. From the table, it seems that the speedup with loose initial is less impressive than that with tight fit. That's because in our original implementation of CLS-VISM with the loose initial, a coarse grid is used to speed up the evolution of the interface at the first stage. Similar idea can also be applied to BLS-VISM through a multi-resolution or adaptive grid.

The significant speedup in computing VISM energy allow us to couple BLS-VISM

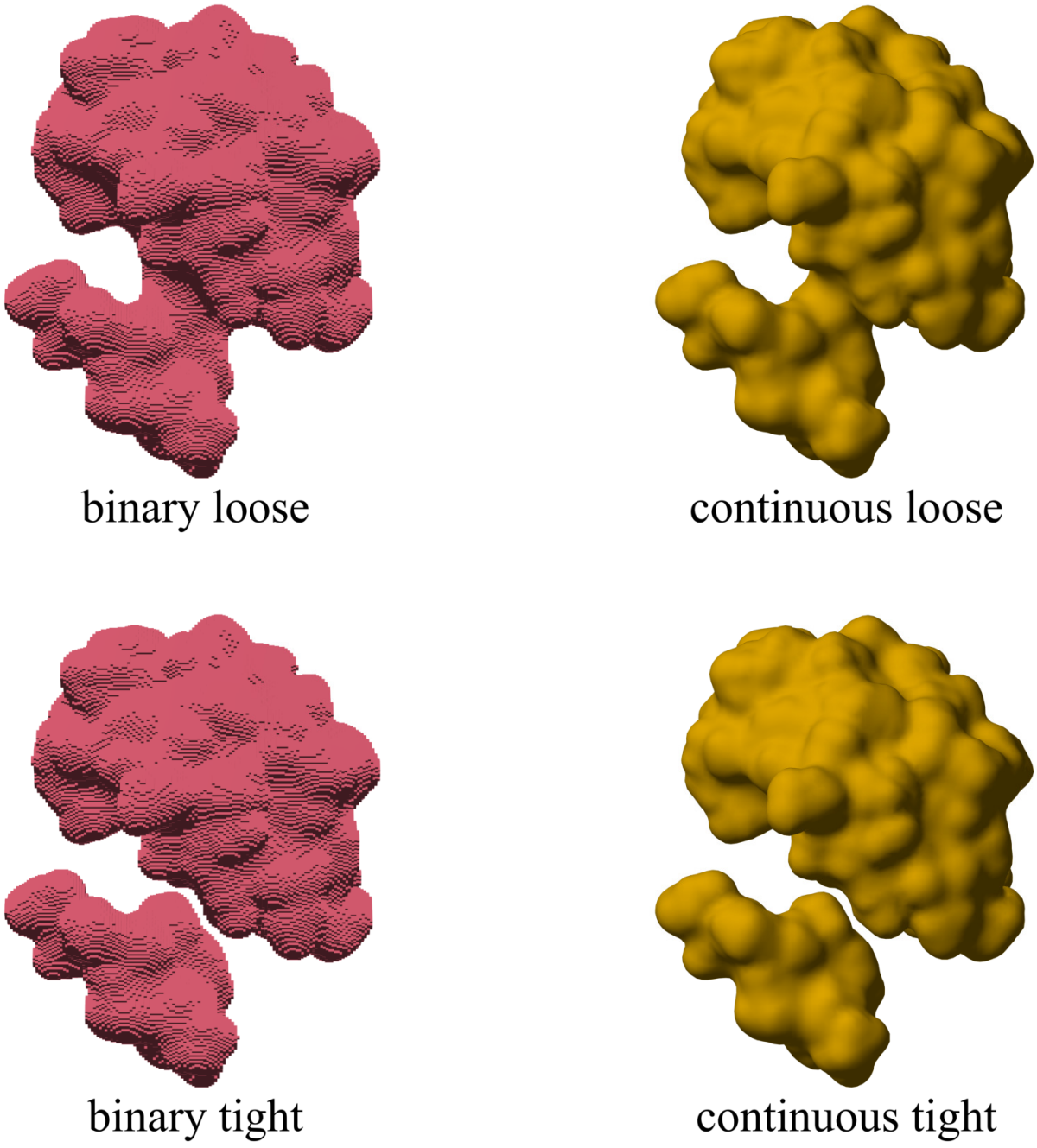


Figure 3.7. Stable equilibrium solute–solvent interfaces of p53-MDM2 obtained at d = 14 Å using tight and loose initials. $n = 200$ for BLS-VISM, $n = 100$ for CLS-VISM

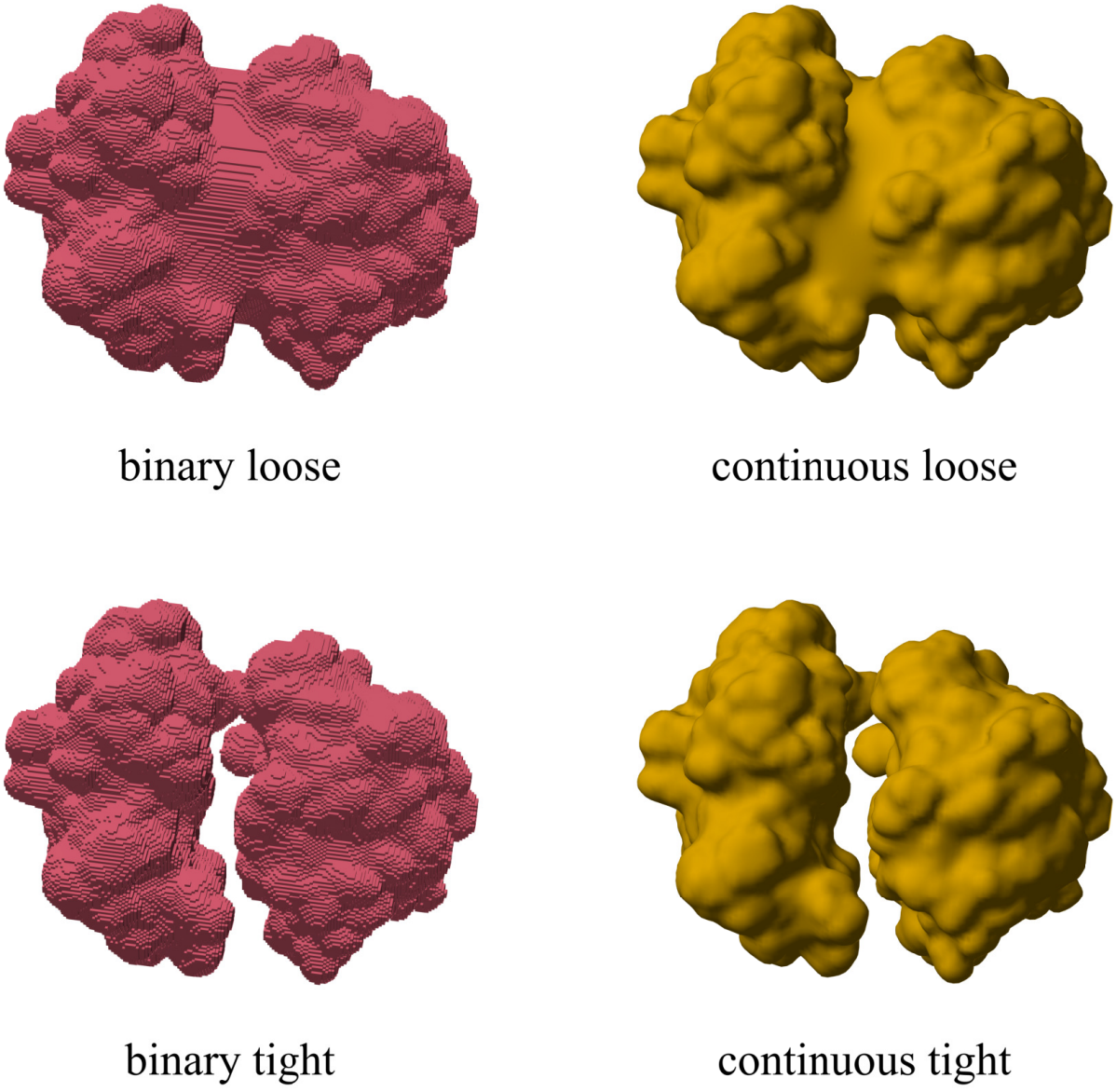


Figure 3.8. Stable equilibrium solute–solvent interfaces of BphC obtained at d = 12 Å.

with Monte Carlo (MC) Method to simulate protein binding. In Chapter 4, the BLS-VISM is coupled with rigid-body MC to simulate binding of the p53-MDM2 system [38]. During the MC simulation, the VISM energy needs to be evaluated millions of times, which is impossible with CLS-VISM. We note that in BLS-VISM with tight initial, the speed is no longer bottlenecked by the optimization process. Instead, the initialization process will take more time than flipping.

Table 3.3. Solvation free energy (k_BT) and computation time (s) for the protein systems

system	Surf		vdW		Elec		total		Time	
	BLS	CLS	BLS	CLS	BLS	CLS	BLS	CLS	BLS	CLS
p53MDM2 loose	992.2	995.8	-437.2	-440.2	-932.3	-895.7	-377.4	-340.2	52.1	1510.1
p53MDM2 tight	1019.4	1017.9	-473.4	-461.1	-944.9	-903.5	-398.8	-346.8	1.1	813.4
BphC loose	2089.8	2032.0	-1218.5	-1083.0	-1624.7	-1456.6	-753.3	-507.6	52.2	25589.2
BphC tight	2259.7	2215.4	-1401.4	-1240.6	-1679.8	-1528.2	-821.4	-553.3	1.2	14858.0

3.4 Conclusion

In this chapter, we introduce a fast binary level set method to minimize the VISM free-energy functional of solute-solvent interfaces. In the binary level set method, the interface is represented by a binary level set function that takes value ± 1 on the solute or solvent region. The key component in our formulation is the approximation of surface area by convolution of indicator function with compactly supported kernel. The resulting discrete VISM energy is minimized by iteratively flipping the value of the binary level set function in a steepest descent fashion. As demonstrated by our numerical experiments, compared with the PDE-based level set method, the binary level set approach is hundreds times faster, and still provide fairly accurate solvation energy. It also has the ability to capture different equilibrium solute-solvent interfaces, which is an characteristic feature of VISM and is important in capturing polymodal hydration.

Future work include estimating the curvature of an interface and incorporating the Poisson-Boltzmann (PB) theory of electrostatics into the binary level set framework.

Further performance gain might be achieved through an adaptive Cartesian grids. We are also interested in applying our fast algorithm to simulate and study the binding and folding of biomolecules.

Chapter 3, in part, is a reprint of the material Zirui Zhang and Li-Tien Cheng. "Binary Level Set Method for Variational Implicit Solvation Model" (2021). The dissertation author was the primary investigator and author of the material.

3.5 Proof of Propositions

3.5.1 Integral formula of Surface Area

Let ϕ be a signed distance function representing the interface Γ with $\phi < 0$ being the region Ω enclosed by Γ.

$$\phi(\mathbf{x}) = \begin{cases} -\inf_{\mathbf{y}\in\Omega} |\mathbf{x}-\mathbf{y}| & \mathbf{x} \in \Omega \\ \inf_{\mathbf{y}\in\Omega^c} |\mathbf{x}-\mathbf{y}| & \mathbf{x} \in \Omega^c \end{cases} \tag{3.23}$$

Let $\mathcal{H}$ denotes the one-dimensional Heaviside function, and δ is the Dirac-delta function. We denote the θ level set of ϕ as $\Gamma_\theta = \{\mathbf{x}|\phi(\mathbf{x}) = \theta\}$ and $\Gamma_0 = \Gamma$. Denote $\mathbf{n} = \mathbf{n}(\mathbf{x})$ the unit normal vector at $\mathbf{x} \in \Gamma$. As a property of the sign distance function, $\nabla\phi(\mathbf{x}) = \mathbf{n}$.

Then consider the following expression

$$\begin{aligned} V(r) &= \int_{\mathbb{R}^d} \mathcal{H}(-\phi(\mathbf{x})) \int_{\mathbb{R}^d} \mathcal{H}(\phi(\mathbf{y}))K\left(\frac{|\mathbf{x}-\mathbf{y}|}{r}\right) d\mathbf{y}d\mathbf{x} \\ &= \int_{\mathbb{R}^d} \mathcal{H}(-\phi(\mathbf{x}))W(\mathbf{x},r)d\mathbf{x} \\ &= \int_{\{-r\le\phi(\mathbf{x})\le 0\}} W(\mathbf{x},r)d\mathbf{x} \end{aligned} \tag{3.24}$$

where

$$\begin{aligned} W(\mathbf{x}, r) &= \int_{\mathbb{R}^d} \mathcal{H}(\phi(\mathbf{y})) K\left(\frac{|\mathbf{x}-\mathbf{y}|}{r}\right) d\mathbf{y} \\ &= r^d \int_{B_1(0)} \mathcal{H}(\phi(r\hat{\mathbf{y}}+\mathbf{x})) K(|\hat{\mathbf{y}}|) d\hat{\mathbf{y}}. \end{aligned} \tag{3.25}$$

The integrand in $V(r)$ is supported on $\{\mathbf{x} \mid -r \le \phi(\mathbf{x}) \le 0\}$ by the compactness of K. Let $\theta \in [0, r]$. Given $\mathbf{z} \in \Gamma_{-\theta}$, there is a unique $\mathbf{x} \in \Gamma$ such that $\mathbf{z} = \mathbf{x} - \theta\mathbf{n}$, as long as r is smaller than the minimum radius of curvature. Therefore we can reparametrize $\Gamma_{-\theta}$ by Γ, By the coarea formula, we can write the volume (3.24) as integral over level sets of ϕ.

$$\begin{aligned} V(r) &= \int_0^r \int_{\Gamma_{-\theta}} W(\mathbf{x}, r) d\mathbf{x} d\theta \\ &= \int_0^r \int_{\Gamma_{-\theta}} W(\mathbf{x}, r) d\mathbf{x} d\theta \\ &= \int_0^r \int_{\Gamma} W(\mathbf{z}-\theta\mathbf{n}, r) Y(\mathbf{z}, \theta) d\mathbf{x} d\theta \\ &= r \int_0^1 \int_{\Gamma} W(\mathbf{z}-r\theta\mathbf{n}, r) Y(\mathbf{z}, r\theta) d\mathbf{x} d\theta \end{aligned} \tag{3.26}$$

Here, $Y(\mathbf{z}, s)$ is the Jacobian that accounts for the change of variables made in the reparametrization of $\Gamma_{-\theta}$ by Γ In two dimension, $Y(\mathbf{z}, s) = 1 - H(\mathbf{z})s$. In three dimensions, $Y(\mathbf{z}, s) = 1 - H(\mathbf{z})s + G(\mathbf{z})s^2$. $H(\mathbf{z})$ is the non-averaged mean curvature. $G(\mathbf{z})$is the Gaussian curvature. In d dimensions, $Y(\mathbf{z}, s) = \prod_{i=1}^{d-1}(1 - s\kappa_i(\mathbf{z}))$, where κ_i is the i-th principal curvature of the hypersurface Γ at $\mathbf{z}$ [21].

We find the expansion of $W(\mathbf{z} - r\theta\mathbf{n}, r)$ with respect to r, cf Fig. 3.9

$$\begin{aligned} W(\mathbf{z}-r\theta\mathbf{n}, r) &= r^d \int_{B_1(0)} \mathcal{H}(\phi(\mathbf{z} + r(\hat{\mathbf{y}} - \theta\mathbf{n}))) K(|\hat{\mathbf{y}}|) d\hat{\mathbf{y}} \\ &= r^d (a_0(\mathbf{z}, \theta) + r a_1(\mathbf{z}, \theta) + \mathcal{O}(r^2)). \end{aligned} \tag{3.27}$$

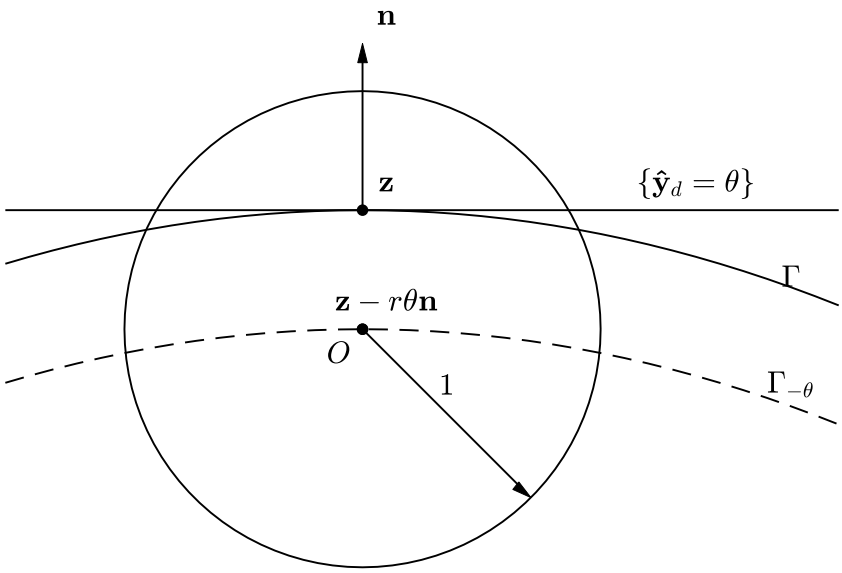


Figure 3.9. Illustration of the term $W(\mathbf{z} - r\theta\mathbf{n}, r)$.

That leads to the expansion of $V(r)$.

$$\begin{aligned} V(\epsilon) &= \int_0^1 \int_{\Gamma_0} W(\mathbf{z} - \epsilon\theta\mathbf{n}, \epsilon) Y(\mathbf{z}, \epsilon\theta) \epsilon dA d\theta \\ &= r^{d+1} \int_0^1 \int_{\Gamma_0} a_0(\mathbf{z}, \theta) dA d\theta + r[a_1(\mathbf{z}, \theta) - H(\mathbf{z})\theta a_0(\mathbf{z}, \theta)] dA d\theta + \mathcal{O}(r^2) \\ &= r^{d+1} \left(v_0 + r v_1 + \mathcal{O}(\epsilon^2) \right) \end{aligned} \tag{3.28}$$

Next, we simplifies the zeroth order term to find the constant $C_{K,r,d}$. By Taylor's expansion,

$$\begin{aligned} \phi(\mathbf{z} + r(\mathbf{y} - \theta\mathbf{n})) &= \phi(\mathbf{z}) + \nabla\phi(\mathbf{z}) \cdot r(\mathbf{y} - \theta\mathbf{n}) + \frac{1}{2} r(\mathbf{y} - \theta\mathbf{n}) \cdot \nabla^2\phi(\mathbf{z}) r(\mathbf{y} - \theta\mathbf{n}) + \mathcal{O}(r^3) \\ &= r\mathbf{n} \cdot (\mathbf{y} - \theta\mathbf{n}) + \frac{1}{2} r^2 (\mathbf{y} - \theta\mathbf{n}) \cdot \nabla^2\phi(\mathbf{z})(\mathbf{y} - \theta\mathbf{n}) + \mathcal{O}(r^3) \end{aligned} \tag{3.29}$$

Together with the fact that $\mathcal{H}$ is homogeneous of degree 0 (i.e. $\mathcal{H}(a\mathbf{x}) = \mathcal{H}(\mathbf{x})$).

$$\mathcal{H}(\phi(\mathbf{z} + r(\mathbf{y} - \theta\mathbf{n}))) = \mathcal{H}(\mathbf{n} \cdot (\mathbf{y} - \theta\mathbf{n}) + \frac{1}{2} r(\mathbf{y} - \theta\mathbf{n}) \cdot \nabla^2\phi(\mathbf{z})(\mathbf{y} - \theta\mathbf{n}) + \mathcal{O}(r^2)) \tag{3.30}$$

We have

$$\lim_{r\to 0}\mathcal{H}(\phi(\mathbf{z}+r(\mathbf{y}-\theta\mathbf{n})))=\mathcal{H}(\mathbf{n}\cdot(\mathbf{y}-\theta\mathbf{n})), \tag{3.31}$$

where $\{\mathbf{y}\mid \mathbf{n}\cdot(\mathbf{y}-\theta\mathbf{n})=0\}$ is the hyperplane passing through $\theta\mathbf{n}$ with normal vector $\mathbf{n}$. Therefore $\{\mathbf{y}\mid \mathbf{n}\cdot(\mathbf{y}-\theta\mathbf{n})\geq 0\}$ is the side of the hyperplane in the direction of $\mathbf{n}$

$$\begin{aligned}
a_0(\mathbf{z},\theta)&=\lim_{r\to 0}W(\mathbf{z}-r\theta\mathbf{n},r)\\
&=\int_{B_1(0)}\mathcal{H}(\mathbf{n}\cdot(\hat{\mathbf{y}}-\theta\mathbf{n}))K(|\hat{\mathbf{y}}|)d\hat{\mathbf{y}}\\
&=\int_{B_1(0)\cap\{\mathbf{n}\cdot(\hat{\mathbf{y}}-\theta\mathbf{n})\geq 0\}}K(|\hat{\mathbf{y}}|)d\hat{\mathbf{y}}\\
&=\int_{B_1(0)\cap\{\hat{\mathbf{y}}_d\geq\theta\}}K(|\hat{\mathbf{y}}|)d\hat{\mathbf{y}}
\end{aligned} \tag{3.32}$$

$\hat{\mathbf{y}}_d$ is the d-th coordinate of $\hat{\mathbf{y}}$. Note $a_0(\mathbf{z},\theta)=a_0(\theta)$ does not depend on $\mathbf{z}$.

$$\begin{aligned}
v_0&=\int_0^1\int_{\Gamma_0}a_0(\theta)dAd\theta\\
&=\left(\int_0^1 a_0(\theta)d\theta\right)\text{Area}(\Gamma)
\end{aligned} \tag{3.33}$$

The constant can be further simplified by writing the integral in polar coordinates $r=|\mathbf{y}|\in(0,1]$ and $\mathbf{x}=\mathbf{y}/|\mathbf{y}|\in S^{d-1}$

$$\begin{aligned}
\int_0^1 a_0(\theta)d\theta&=\int_0^1\int_{B_1(0)}\mathbf{1}_{\{\mathbf{y}_d\geq\theta\}}K(|\mathbf{y}|)d\mathbf{y}d\theta\\
&=\int_0^1\int_0^1\int_{S^{d-1}}\mathbf{1}_{\{r\mathbf{x}_d\geq\theta\}}K(r)r^{d-1}dAdrd\theta\\
&=\int_0^1\int_{S^{d-1}}\left(\int_0^1\mathbf{1}_{\{r\mathbf{x}_d\geq\theta\}}d\theta\right)K(r)r^{d-1}dAdr\\
&=\int_0^1\int_{S^{d-1}}\mathbf{1}_{\{\mathbf{x}_d>0\}}r\mathbf{x}_dK(r)r^{d-1}dAdr\\
&=\left(\int_{S^{d-1}\cap\{\mathbf{x}_d>0\}}\mathbf{x}_d dA\right)\int_0^1 K(r)r^d dr\\
&=C_d\int_0^1 K(r)r^d dr
\end{aligned} \tag{3.34}$$

where

$$C_d = \frac{2\pi^{\frac{d-1}{2}}}{(d-1)\Gamma(\frac{d-1}{2})} \tag{3.35}$$

For $d = 3$, $C_d = \pi$. For $d = 2$, $C_d = 2$.

For the first order term, we use the identity that $0 = \nabla(\mathbf{n} \cdot \mathbf{n}) = 2\nabla^2\phi(\mathbf{z})\mathbf{n}$. And without loss of generality, we can assume $\mathbf{n} = e_d$, and hence $[\nabla^2\phi(\mathbf{z})]_{dd} = 0$

$$\begin{aligned}
a_1(\mathbf{z}, \theta) &= \frac{d}{dr}\Big|_{r=0} \int_{B_1(0)} \mathcal{H}(\phi(\mathbf{z} + r(\mathbf{y} - \theta\mathbf{n})))K(|\mathbf{y}|)d\mathbf{y} \\
&= \int_{B_1(0)} \frac{d}{dr}\Big|_{r=0} \mathcal{H}(\phi(\mathbf{z} + r(\mathbf{y} - \theta\mathbf{n})))K(|\mathbf{y}|)d\mathbf{y} \\
&= \int_{B_1(0)} \delta(\mathbf{n} \cdot (\mathbf{y} - \theta\mathbf{n}))\frac{1}{2}(\mathbf{y} - \theta\mathbf{n}) \cdot \nabla^2\phi(\mathbf{z})(\mathbf{y} - \theta\mathbf{n})K(|\mathbf{y}|)d\mathbf{y} \\
&= \frac{1}{2}\int_{B_1(0)\cap\{\mathbf{y}_d=\theta\}} (\mathbf{y} - \theta\mathbf{n}) \cdot \nabla^2\phi(\mathbf{z})(\mathbf{y} - \theta\mathbf{n})K(|\mathbf{y}|)d\mathbf{y} \\
&= \frac{1}{2}\int_{B_1(0)\cap\{\mathbf{y}_d=\theta\}} \mathbf{y} \cdot \nabla^2\phi(\mathbf{z})\mathbf{y}K(|\mathbf{y}|)d\mathbf{y} \\
&= \frac{1}{2}\int_{B_1(0)\cap\{\mathbf{y}_d=\theta\}} \mathbf{y} \cdot \nabla^2\phi(\mathbf{z})\mathbf{y}K(|\mathbf{y}|)d\mathbf{y} \\
&= \frac{1}{2}\sum_{i,j=1}^{d} \int_{B_1(0)\cap\{\mathbf{y}_d=\theta\}} \mathbf{y}_i\mathbf{y}_j[\nabla^2\phi(\mathbf{z})]_{ij}K(|\mathbf{y}|)d\mathbf{y}
\end{aligned} \tag{3.36}$$

The region of integration is a spherical section of the unit ball. Suppose $i \neq j$, and without loss of generality, suppose $j \neq d$, then the integrand is an odd function with respect to $\mathbf{y}_j$.

By symmetry of the region, the integral is 0. Therefore

$$\begin{aligned} a_1(\mathbf{z},\theta) &= \frac{1}{2}\sum_{i,j=1}^{d}\int_{B_1(0)\cap\{\mathbf{y}_d=\theta\}} \mathbf{y}_i\mathbf{y}_j[\nabla^2\phi(\mathbf{z})]_{ij}K(|\mathbf{y}|)d\mathbf{y} \\ &= \frac{1}{2}\left(\sum_{i=1}^{d}[\nabla^2\phi(\mathbf{z})]_{ii}\right)\int_{B_1(0)\cap\{\mathbf{y}_d=\theta\}}\mathbf{y}_1^2K(|\mathbf{y}|)d\mathbf{y} \\ &= \frac{1}{2}H(z)\int_{B_1(0)\cap\{\mathbf{y}_d=\theta\}}\mathbf{y}_1^2K(|\mathbf{y}|)d\mathbf{y} \\ &= H(z)\hat{a}_1(\theta) \end{aligned} \tag{3.37}$$

Note that

$$\begin{aligned} \int_0^1\theta a_0(\theta)d\theta &= \int_0^1\int_{B_1(0)\cap\{\mathbf{y}_d\geq\theta\}}\theta K(|\mathbf{y}|)d\mathbf{y}d\theta \\ &= \int_{B_1(0)\cap\{\mathbf{y}_d\geq 0\}}\int_0^1\mathcal{H}(\mathbf{y}_d-\theta)\theta K(|\mathbf{y}|)d\theta d\mathbf{y} \\ &= \int_{B_1(0)\cap\{\mathbf{y}_d\geq 0\}}\left(\int_0^{\mathbf{y}_d}\theta d\theta\right)K(|\mathbf{y}|)d\mathbf{y} \\ &= \frac{1}{2}\int_{B_1(0)\cap\{\mathbf{y}_d\geq 0\}}{\mathbf{y}_d}^2K(|\mathbf{y}|)d\mathbf{y} \\ &= \int_0^1\hat{a}_1(\theta)d\theta \end{aligned} \tag{3.38}$$

Therefore

$$v_1 = \int_{\Gamma_0}\int_0^1 a_1(\mathbf{z},\theta)-\theta H(\mathbf{z})a_0(\mathbf{z},\theta)d\theta dA = 0 \tag{3.39}$$

and

$$\text{Area}(\Gamma) = C_{K,r,d}V(r)+\mathcal{O}(r^2) \tag{3.40}$$

where

$$C_{K,r,d} = \left(r^{d+1}C_d\int_0^1 K(r)r^d dr\right)^{-1} \tag{3.41}$$

3.5.2 Discretization error of Integral formula

Let $\Gamma=\partial\Omega$ be a smooth compact closed hypersurface contained in a cube $\mathcal{C}\subset\mathbb{R}^d$. $\mathcal{C}=\Omega\cup\Gamma\cup\Omega^c$.

Suppose $\mathcal{C}$ is covered with a uniform Cartesian grid of size h, with n grid cells in each dimension. Let c_i be the grid cell centered at $\mathbf{x}_i$ with side length h, where i is a multi-index $(i_1, \ldots, i_d)$, with $i_k \in \{1, \ldots, n\}$. With a slight abuse of notation, we also denote Ω the index set containing all the index, so $i \in \Omega$ if $\mathbf{x}_i \in \Omega$. Similarly for $\mathcal{C}$ and Ω^c. We also define the index set $I = \{i \mid c_i \cap \Gamma \neq \emptyset\}$. I contains all the "interface points", whose grid cells touch the interface. I^c contains all the "interior points", whose grid cells are completely in Ω or Ω^c.

Recall the midpoint rule for numerical integration. Suppose f is twice continuously differentiable. Let $\mathbf{x}^*$ be the center of a d dimensional cube of side length h. Then the approximation on the cube has error $\mathcal{O}(h^{d+2})$.

$$\begin{aligned}\int_{[0,h]^d} f(\mathbf{x})d\mathbf{x} &= \int_{[0,h]^d} f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*) \cdot \nabla f(\mathbf{x}^*) + \mathcal{O}(h^2)d\mathbf{x} \\ &= h^d f(\mathbf{x}^*) + \mathcal{O}(h^{d+2})\end{aligned} \tag{3.42}$$

If the region is not a cube but some region $\omega \subset [0, h]^d$, and the volume of the region is known, then our error is $\mathcal{O}(h^{d+1})$

$$\begin{aligned}\int_{\omega} f(\mathbf{x})d\mathbf{x} &= \int_{\omega} f(\mathbf{x}^*) + \mathcal{O}(h)d\mathbf{x} \\ &= \text{vol}(\omega) f(\mathbf{x}^*) + \mathcal{O}(h^{d+1})\end{aligned} \tag{3.43}$$

However, if the exact volume of the region ω is unknown, then the following approximation has error $\mathcal{O}(h^d)$

$$\int_{\omega} f(\mathbf{x})d\mathbf{x} = h^d \mathbf{1}_{\omega}(\mathbf{x}^*) f(\mathbf{x}^*) + \mathcal{O}(h^d) \tag{3.44}$$

Recall the formula of $V(r)$

$$\begin{aligned} W(\mathbf{x}, r) &= \int_{\Omega^c} K\left(\frac{|\mathbf{x}-\mathbf{y}|}{r}\right) d\mathbf{y} \\ V(r) &= \int_{\Omega} W(\mathbf{x}, r) d\mathbf{x} \end{aligned} \tag{3.45}$$

Let $\hat{\alpha}_i = \text{vol}(c_i \cap \Omega)$, $\hat{\beta}_i = \text{vol}(c_i \cap \Omega^c)$. These are the exact volume fractions of the grid cells. We define the intermediate quantity $\hat{V}_h(r)$, which is composite midpoint rule approximation of $V(r)$ using exact volume fraction.

$$\hat{V}_h(r) = \sum_i \sum_j h^{2d} \hat{\alpha}_i \hat{\beta}_i K_{r,ij}, \tag{3.46}$$

where the summation is taken over all ordered tuple of indices $(i, j) \in \mathcal{C} \times \mathcal{C}$, and

$$K_{r,ij} = K\left(\frac{|\mathbf{x}_i - \mathbf{x}_j|}{r}\right), \tag{3.47}$$

To look at the error of $\hat{V}_h(r)$, first, we approximate $W(\mathbf{x}_i, r)$ by composite midpoint rule with exact volume fraction. For each interface point, the error is $\mathcal{O}(h^{d+1})$. The number of interface points in a ball of radius r is $\mathcal{O}(r^{d-1}h^{1-d})$, Therefore, the total error is given by

$$W(\mathbf{x}_i, r) = \sum_j h^d \hat{\beta}_j K_{r,ij} + \mathcal{O}(r^{d-1}h^2) \tag{3.48}$$

Next, we approximate $V(r)$ by composite midpoint rule with exact volume fraction. The number of points near the interface is given by $|I| = \mathcal{O}(h^{1-d})$. Since $W(\mathbf{x}_i, r) = \mathcal{O}(r^d)$, the error on individual interface point is $\mathcal{O}(r^d h^{d+1})$. The total error is given by

$$V(r) = \sum_i h^d \hat{\alpha}_i W(\mathbf{x}_i, r) + \mathcal{O}(h^2 r^d) \tag{3.49}$$

Plug in the estimation of $W(\mathbf{x}_i, r)$, and recall that $W(\mathbf{x}_i, r)$ is nonzero for points in

a tubular neighborhood of the interface. Let $T = \{i \mid -r \leq \phi(\mathbf{x}_i) \leq 0\}$. $|T| = \mathcal{O}(r/h^d)$. Hence

$$\begin{aligned} V(r) &= \sum_i \sum_j h^{2d} \hat{\alpha}_i \hat{\beta}_i K_{r,ij} + \mathcal{O}(h^d (r/h^d)(h^2 r^{d-1}) + h^2 r^d) \\ &= \hat{V}_h(r) + \mathcal{O}(h^2 r^d) \end{aligned} \tag{3.50}$$

We also define $V_h(r)$, which is composite midpoint rule approximation of $V(r)$ without knowing the exact volume fraction.

$$V_h(r) = \sum_{i,j} h^{2d} \alpha_i \beta_j K_{r,ij} \tag{3.51}$$

where $\alpha_i = 1$ if $i \in \Omega$, and $\alpha_i = 0$ if $i \in \Omega^c$. $b_i = 1 - a_i$

We analyze the difference between $V_h(r)$ and $\hat{V}_h(r)$

$$V_h(r) - \hat{V}_h(r) = \sum_{i,j} h^{2d} (\alpha_i \beta_j - \hat{\alpha}_i \hat{\beta}_j) K_{r,ij} \tag{3.52}$$

by partitioning $\mathcal{C} \times \mathcal{C}$ into three disjoint subsets Z_1,Z_2 and Z_3, cf Fig. 3.10.

Case 1: $Z_1 = I^c \times I^c$. Both i and j are interior points. Then $\hat{\alpha}_i = 1$ if $i \in \Omega$, and 0 otherwise. So $\hat{\alpha}_i = \alpha_i$ and $\hat{\beta}_i = \beta_i$. On this set, the difference between the two methods is 0.

Case 2: We define $R_i(j) = 2i - j$, which is the point reflection of j with respect to i. Let $\hat{\phi}_i = \text{sgn}(\phi(\mathbf{x}_i))$ be the sign of the level set function at $\mathbf{x}_i$. So the condition $\hat{\phi}_i \hat{\phi}_j = -1$ means that $\mathbf{x}_i$ and $\mathbf{x}_j$ are on opposite side of the interface. And define the following two sets that are disjoint

$$\begin{aligned} S_1 &= \{(i,j) \mid i \in I, j \in I^c, R_i(j) \in I^c, \hat{\phi}_{R_i(j)} \hat{\phi}_j = -1\} \\ S_2 &= \{(i,j) \mid i \in I^c, j \in I, R_j(i) \in I^c, \hat{\phi}_{R_j(i)} \hat{\phi}_i = -1\} \end{aligned} \tag{3.53}$$

In words, for tuples (i,j) in S_1, i is an interface point, j is an interior point, the reflection

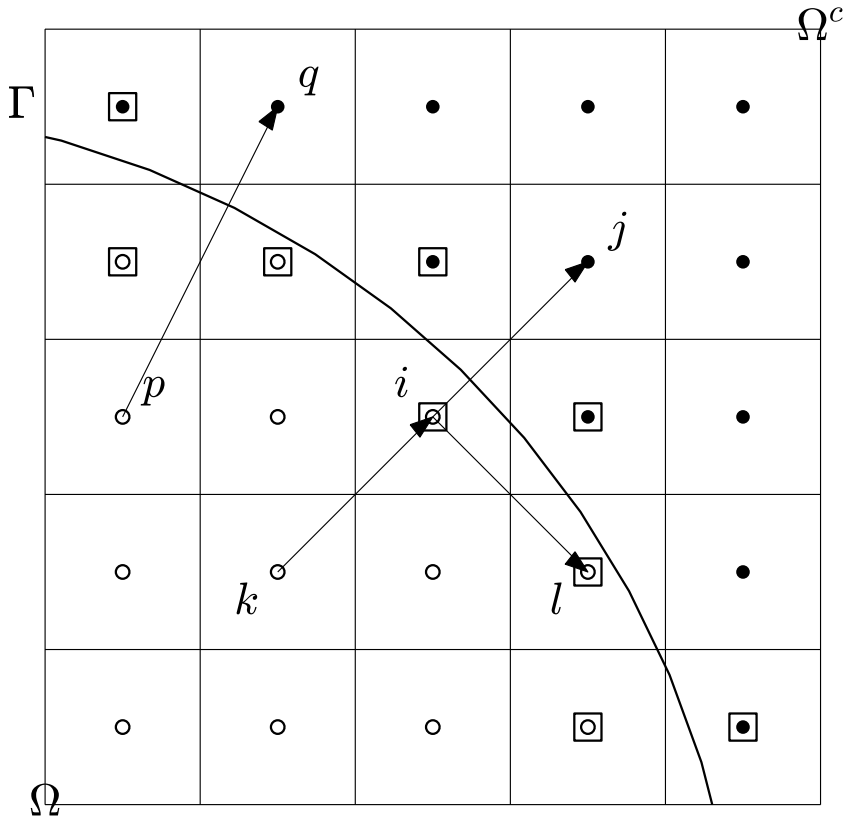


Figure 3.10. Illustration of different cases in the error. Squared grid points are in the set I, their grid cell intersects with the interface Γ. $(p, q) \in Z_1$. (i, j) and (k, i) are paired and belong to Z_2. $(i, l) \in Z_3$

of j is also an interior point on the opposition side of the interface. By definition, the map $(i,j) \mapsto (R_i(j), i)$ is a bijection between $S_1$and S_2. For case 2, $Z_2 = S_1 \cup S_2$.

Given that $(i,j) \in Z_2$, we can pair up the error,

$$\begin{aligned} &\sum_{(i,j)\in S_1 \cup S_2} h^{2d}(\alpha_i\beta_j - \hat{\alpha}_i\hat{\beta}_j)K_{r,ij} \\ &= \sum_{(i,j)\in S_1} h^{2d}\left[(\alpha_i\beta_j - \hat{\alpha}_i\hat{\beta}_j) + (\alpha_k\beta_i - \hat{\alpha}_k\hat{\beta}_i)\right]K_{r,ij} \end{aligned} \tag{3.54}$$

where $k = R_i(j)$. Because both j and k are interior points, $\hat{\beta}_j = \beta_j$ and $\hat{\alpha}_k = \alpha_k$. Because j and k are in different side, $\alpha_k = 1 - \alpha_j = \beta_j$. Together with the fact that $\alpha_i + \beta_i = 1$ and $\hat{\alpha}_j + \hat{\beta}_j = 1$, we have

$$(\alpha_i\beta_j - \hat{\alpha}_i\hat{\beta}_j) + (\alpha_k\beta_i - \hat{\alpha}_k\hat{\beta}_i) = \alpha_i\beta_j - \hat{\alpha}_i\beta_j + \beta_j\beta_i - \beta_j\hat{\beta}_i = 0. \tag{3.55}$$

Hence, on case 2, the difference between the two methods also is 0.

Case 3: Z_3 contains all the remaining tuples, where there is no cancellation that can be exploited, so $\alpha_i\beta_j - \hat{\alpha}_i\hat{\beta}_j = \mathcal{O}(1)$. Our goal is to bound the size of Z_3. Given $i \in I$, we define the following index set:

$$Q_{r,i} = \{|\mathbf{x}_i - \mathbf{x}_j| \le r\} \cap \Big(\{j \in I\} \cup \{j \in I^c, R_i(j) \in I \text{ or } \hat{\phi}_{R_i(j)}\hat{\phi}_j = 1\}\Big) \tag{3.56}$$

In words, given that i is an interface point, among all the points $\mathbf{x}_j$ that are within kernel radius of $\mathbf{x}_i$, either j is also an interface point, or j is an interior point whose reflection is an interface point or an interior point on the same side as i. Then Z_3 can be written as

$$Z_3 = \{(i,j) \mid i \in I, j \in Q_{r,i}\} \cup \{(i,j) \mid j \in I, i \in Q_{r,j}\} \tag{3.57}$$

We use the fact that an embedded hypersurface can be locally approximated by

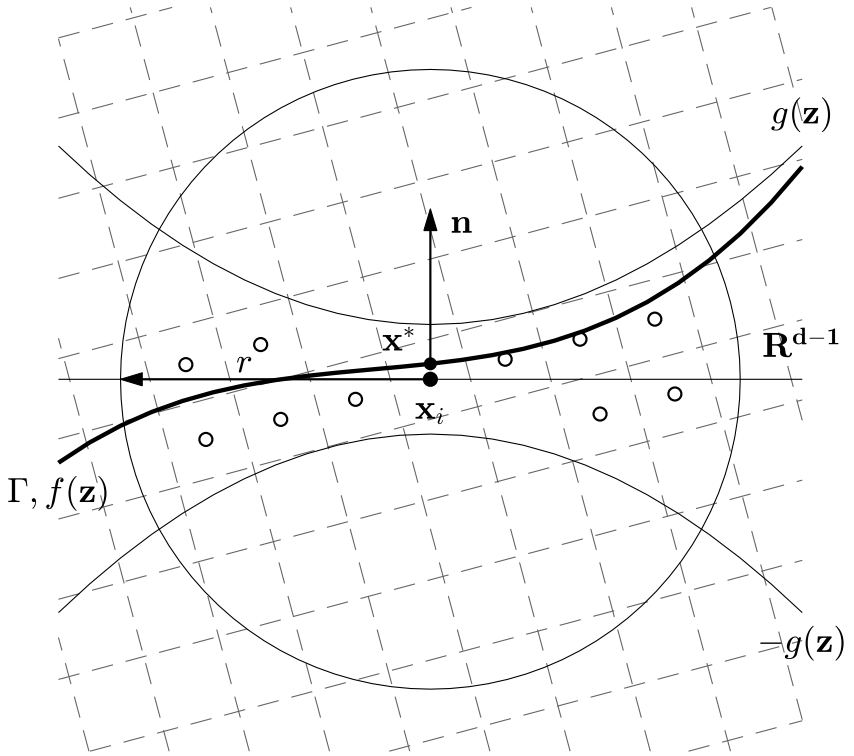


Figure 3.11. Estimation of $|Q_{r,i}|$. $\mathbf{x}_i$ is an interface point. $\mathbf{x}^*$ is the closest point to $\mathbf{x}_i$ on Γ. In local coordinate, Γ is the graph of the function $f(\mathbf{z})$. Points in $Q_{r,i}$ are circled and bounded between the quadratic functions $g(z)$ and $-g(z)$

the graph of a quadratic function. Given $\mathbf{x}_i \in I$, we can find the closest point $\mathbf{x}^*$ on the interface Γ, so that $\mathbf{x}_i = \mathbf{x}^* - s\mathbf{n}(\mathbf{x}^*)$ for some s, and $|s| \leq d^{1/2}h/2$, which is half the length of the diagonal of a grid cell. We can create a local coordinate where $\mathbf{x}_i$ is the origin, and the first $d-1$ coordinate is parallel to the tangent plane of Γ at $\mathbf{x}^*$. Then locally, Γ can be parametrized as a graph $(\mathbf{z}, f(\mathbf{z}))$, $\mathbf{z} \in \mathbb{R}^{d-1}$,

$$f(\mathbf{z}) = s + \frac{1}{2}\left(\kappa_1(z_1) + \cdots + \kappa_{d-1}(z_{d-1})\right) + \mathcal{O}(|\mathbf{z}|^3), \tag{3.58}$$

where $\kappa_1, \ldots, \kappa_{d-1}$ are the principal curvatures of Γ at $\mathbf{x}^*$. So $f(\mathbf{z})$ will be bounded above by some quadratic function $g(\mathbf{z})$.

$$g(\mathbf{z}) = |s| + \mathcal{O}(|\mathbf{z}|^2), \tag{3.59}$$

And $-g(\mathbf{z}) < f(\mathbf{z}) < g(\mathbf{z})$. Hence, if $j \in Q_{r,i}$, $\mathbf{x}_i$ and $\mathbf{x}_j$ will both be in the region between $g(\mathbf{z})$ and $-g(\mathbf{z})$. Otherwise $(i,j) \in Z_2$.

By the compactness of the kernel, we only need to consider $\mathbf{z} \in B(r)$, a $d-1$ dimensional ball of radius r. The volume of the region under the graph is bounded by

$$\int_{B(r)} |s| + \mathcal{O}(\mathbf{z}^2) d\mathbf{z} = \mathcal{O}(hr^{d-1} + r^{d+1}). \tag{3.60}$$

Hence

$$|Q_{r,i}| = \mathcal{O}\left(h^{1-d}r^{d-1} + h^{-d}r^{d+1}\right) \tag{3.61}$$

and the size of Z_3 can be bounded

$$|Z_3| \leq 2|I||Q_{r,i}| = \mathcal{O}\left(r^{d-1}h^{2(1-d)} + r^{d+1}h^{1-2d}\right). \tag{3.62}$$

And we obtained the error between $\hat{V}_h(r)$ and $V_h(r)$:

$$\begin{aligned} \hat{V}_h(r) &= V_h(r) + \mathcal{O}(|Z_3|)h^{2d} \\ &= V_h(r) + \mathcal{O}(h^2 r^{d-1} + h r^{d+1}) \end{aligned} \tag{3.63}$$

Hence the error between $\hat{V}_h(r)$ and $V(r)$ is given by:

$$\begin{aligned} V(r) &= \hat{V}_h(r) + \mathcal{O}(h^2 r^d) \\ &= V_h(r) + \mathcal{O}(h^2 r^{d-1} + h r^{d+1} + h^2 r^d) \end{aligned} \tag{3.64}$$

Recall that $C_{K,r,d} = \mathcal{O}(r^{-(d+1)})$, we have

$$\begin{aligned} \text{Area}(\Gamma) &= C_{K,r,d} V(r) + \mathcal{O}(r^2) \\ &= C_{K,r,d} V_h(r) + \mathcal{O}(h^2/r^2 + h + h^2/r + r^2) \end{aligned} \tag{3.65}$$

Therefore, for $r \sim \sqrt{h}$, we obtained a first order approximation of the surface area.

$$\text{Area}(\Gamma) = C_{K,r,d} V_h(r) + \mathcal{O}(h) \tag{3.66}$$

Chapter 4

Coupling Monte Carlo, Variational Implicit Solvation, and Binary Level-Set for Simulations of Biomolecular Binding

4.1 Introduction

Biomolecular binding in aqueous solvent is fundamental to biological functions yet extremely complex due to the many-body interactions spanning across multiple temporal and spatial scales. There have been growing interest in understanding the mechanisms of such biomolecular processes, due to particularly the rapid development in rational drug design [1]. Water is recognized as an important player in many biomolecular activities, including protein conformational changes and protein binding. However, and explicit solvent MD simulation remains computationally expensive.

In this chapter, we describe our work [38] that combine the Monte Carlo (MC) method and the fast binary level set - VISM in the previous chapter 3. The MC method is used to simulate the diffusion of individual proteins and the formation of biomolecular complex. The binary level set - VISM is used to estimate the solvation free energy of the system. The simulation consists of a sequence of MC moves, which are accepted or rejected by the Metropolis criterion. During the MC simulation, the VISM energy needs

to be evaluated millions of times, which is impossible with the original LS-VISM.

4.2 Theory and Algorithm

We consider two molecules A (with N_A atoms) and B (with N_B atoms) in an aqueous solvent, as shown in Figure 4.1. We denote by $\mathbf{r}_i^A$, Q_i^A $(i = 1, \ldots, M)$ the solute atomic positions and partial charges of A, and similarly for B. Let $\mathbf{R}$ be the positions of all the atoms from both molecules.

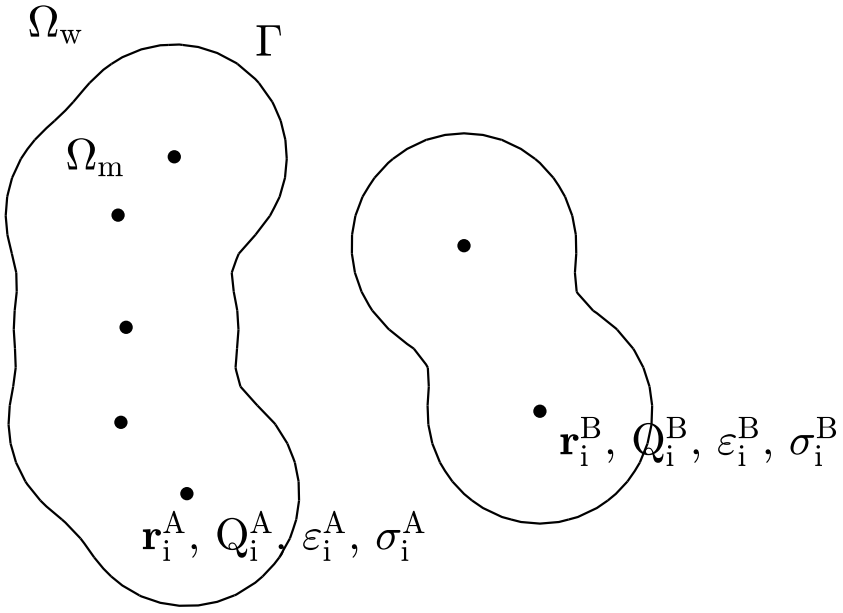


Figure 4.1. schematic of MC-VISM. The parameters for the two biomolecules are indicated by superscript A and B.

The total interaction free energy of this system is

$$G_{\text{total}}[\mathbf{R}] = G_{\text{solvation}}[\mathbf{R}] + G_{\text{vdW,ss}}[\mathbf{R}] + G_{\text{elec,ss}}[\mathbf{R}] \tag{4.1}$$

The first term is the VISM solvation energy described in the previous chapter 3, but considering the position of the atoms as variables.

$$G_{\text{solvation}}[\mathbf{R}] = \min_{\Gamma} G_{\text{VISM}}[\Gamma, \mathbf{R}] \tag{4.2}$$

The second term is the solute-solute vdW interaction energy using Lennard-Johns(LJ) potential.

$$G_{\mathrm{vdW,ss}}[\mathbf{R}] = \sum_{i=1}^{N_A}\sum_{j=1}^{N_B} 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{|\mathbf{r}_i^A - \mathbf{r}_j^B|}\right)^{12} - \left(\frac{\sigma_{ij}}{|\mathbf{r}_i^A - \mathbf{r}_j^B|}\right)^{6}\right], \tag{4.3}$$

where ε_{ij} and σ_{ij} are the energy and length parameters of the LJ potential atom at $\mathbf{r}_i^A$ and $\mathbf{r}_j^B$. The third term is the solute-solute electrostatic interaction energy using Coulomb potential.

$$G_{\mathrm{elec,ss}}[\mathbf{R}] = \frac{1}{4\pi\varepsilon_0\varepsilon_{\mathrm{w}}}\sum_{i=1}^{N_A}\sum_{j=1}^{N_B}\frac{Q_i^A Q_j^B}{|\mathbf{r}_i^A - \mathbf{r}_j^B|}, \tag{4.4}$$

where ε_0 is the vacuum permittivity, and ε_w is the relative permittivity of the solvent.

We sample the configuration space using Monte Carlo Method with the Metropolis criterion [18]. The two molecules are treated as rigid bodies, that is, the relative positions within each molecule does not change. Therefore, the degrees of freedom is drastically reduced, as the position of all the atoms only depends on the centers of mass and the orientations of the two molecules. However, this might

In the simulation, we fix the molecule with a larger number of atoms (assumed to be molecule A). Now the configuration of the system only depends on the center of mass and orientation of molecule B. Notice that the contribution to the vdW energy $(G_{\mathrm{vdW}})_i$ (3.13) and electrostatic energy $(G_{\mathrm{elec}})_i$ (3.15) in each grid cell depends on the position of all the atoms, so we can pre-compute the contributions from atoms in molecule A, and these can also noticeably reduce the time to compute the solvation energy.

4.3 p53-MDM2: Simulation results and analysis

4.3.1 Solvation Free Energy of p53-MDM2

We study the solvation behavior of the p53 using Binary Level Set - VISM. We generate an artificial dissociation pathway along the axis formed by the geometrical centers

Algorithm 4.2.1: MC-VISM Algorithm

input : Initial position $\mathbf{R}$, physical constants, M = maximum number of iteration

Compute $G_{\text{total}}[\mathbf{R}]$ **for** *k=1:M* **do**
  Randomly rotate and translate molecule B to get new position $\mathbf{R}'$.
  Compute $G_{\text{solvation}}[\mathbf{R}']$ (4.2) by minimizing the VISM free-energy functional.
  Compute $G_{\text{total}}[\mathbf{R}']$ (4.1) and $\Delta G = G_{\text{total}}[\mathbf{R}'] - G_{\text{total}}[\mathbf{R}]$.
  Generate a random number $\alpha \in [0, 1]$.
  if $\exp(-\Delta G/k_B T) < \alpha$ **then**
    accept the Monte Carlo move, $R \leftarrow R'$.
  end
end

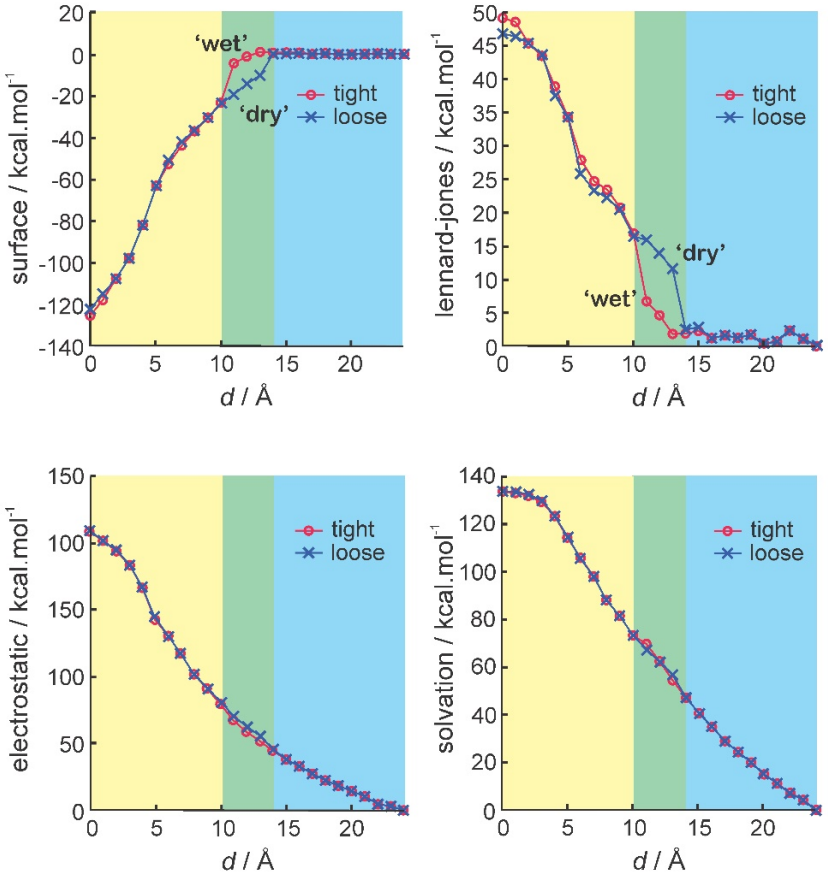


Figure 4.2. Solvation free energy (and relative components) of MDM2 and p53 along the reaction coordinate, d, obtained from tight and loose initial conditions. Highlighted in yellow and blue are the regions for which loose and tight calculations converge producing either desolvated or solvated states, respectively, and highlighted in green is the region where tight and loose calculations diverge producing different solvation boundaries depending on the initial conditions ("branching").

of the two protein in the bound complex. The distance along this coordinate varied from $d = 0$ (X-ray structure) to $d = 24$ Å, with the energy calculated every 1 Åstarting from both the loose and tight initial surfaces. The results are shown in figure 4.3 For small ($d < 10$) or large ($d > 14$) distance, the tight and loose initials capture similar solvation states, and thus results in the same energy. For intermediate distance ($10 < d < 14$), "branching" of the solvation free energies reveals the existence of heterogeneous solvation states. Loose initial leads to the "wet" state, where there is water between p53 and MDM2, and the interface Γ has two disconnected component. Tight initial leads to the "dry" state, where there is no water between p53 and MDM2, and the interface Γ has only one connected component that includes both p53 and MDM2. These result show that binary level-set VISM preserves a significant feature of the original continuous level-set VISM, that is the ability to capture different stable minima in the solvation landscape.

Our binary level-set VISM qualitatively capture the solvation behavior: the dry and wet states in the p53-MDM2 complex have been observed in MD simulations and the continuous level-set VISM calculations [16]. In particular, from explicit-solvent MD simulations, [30] strong dewetting is observed in the MDM2 binding pocket when the two proteins are apart by < 7.6 Å and dewetting fluctuations in the inter-protein region when the proteins are apart as far as 15 Å. In addition, our binary level-set VISM calculations provided quantitatively reasonable estimation of the solvation free energy and its components, compared with known MD simulations results [39]. What's impressive with the fast binary level-set method is the speed: each data point in Figure 4.2 within seconds of computational time.

4.3.2 Rigid-Body MC-VISM Simulations of the Binding of p53-MDM2

We apply our algorithm to capture binding events between p53 peptide and MDM2. The p53-MDM2 interaction is a relevant pharmacological target for anticancer therapeutics

[6], and an important model for studying protein-protein binding [40].

We start with different initial unbound configurations, which are generated by pulling p53 by 15 Å away from MDM2 along the axis connected their geometric centers in the bound complex (PDB ID 1YCR). Then we randomly rotate p53 by less than 90°. Initial positions with steric clashes are rejected. For each initial conformation, we perform 5 trails, each consisting of 100,000 MC moves. For each random perturbation, the direction of translation and the axis of rotation are uniformly distributed on the unit sphere. The magnitude of the translation is uniformly distributed in [0,1]. The magnitude of rotation is uniformly distributed between 0°and 3.72°. This maximum degree is chosen such that with maximum rotation, the out-most atom move 1 Å. As a metric for binding, we used the average of pairwise distances between C_α atoms near the binding site as proposed by [40]. A large binding distance could correspond to an unbound state or to an incorrectly bound state.

Figure 4.3 show the distribution of the many MC-VISM trajectories. The horizontal axis is the binding distance, and the vertical axis is the total energy. The initial poses are marked by asterisks, the final poses are marked by circles, and the intermediate poses are colored from blue to yellow. Many simulations resulted in large binding distances with some decrease in the binding energy, suggesting that p53 engaged in some kind of nonspecific interactions with MDM2, as consistent with the rough energy landscapes of binding that is expected. Some simulations, however, produced binding distances < 12 Å that were accompanied by a sharp and favorable decrease in the binding energy, indicating the formation of specific interactions between p53 and MDM2 (highlighted area in Figure 4.3A). These were considered productive simulations, as they resulted in productive (specific) interactions between p53 and MDM2.

A visual inspection of productive MC simulations reveals that they produced essentially the same binding mode, with the N-terminal portion of the p53 peptide well positioned for binding, while the central Y23 and the C-terminal portion are not yet buried

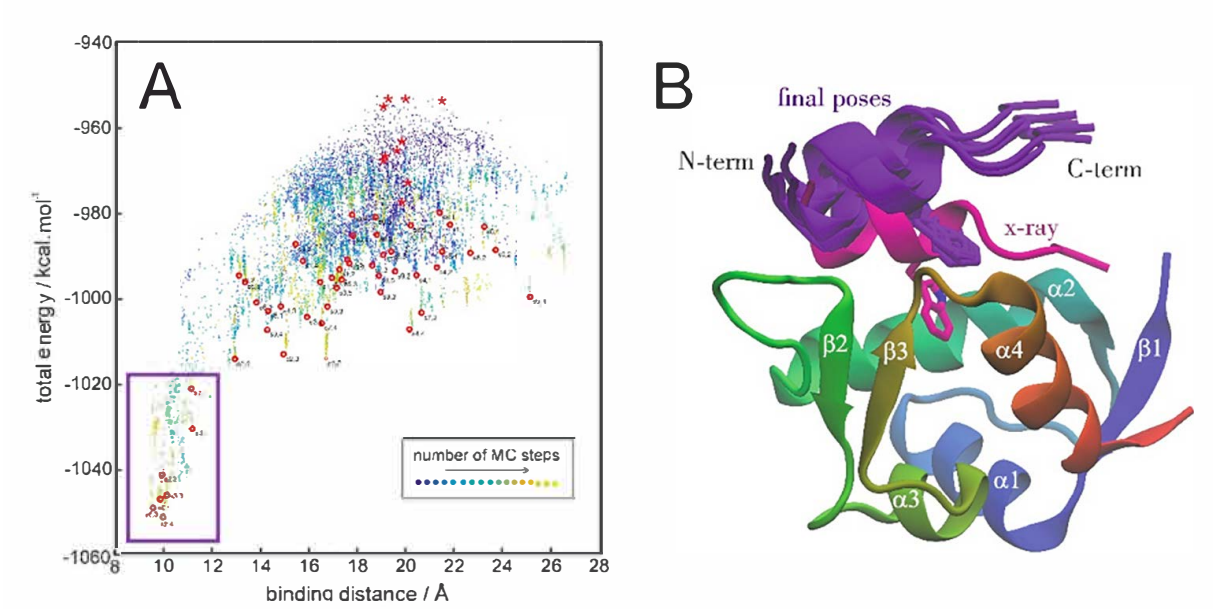


Figure 4.3. (A) binding distance and total energy of samples from the trajectories of simulations. Initial configurations are marked by red asterisks. Final configurations are marked by red circles. Configurations are colored from blue to yellow corresponding to the number of MC steps. Productive binding encounters with small binding distance and small total energy are highlighted. (B) Superimposition of the final binding poses from productive MC-VISM simulations (purple) and the X-ray complex (magenta). For reference, the central W23 residue is displayed. The MDM2 secondary structure is colored from the N- to the C-terminal

within the MDM2 binding cleft, as shown in Figure 4.3(B). We consider these poses to be a prebound state because the energy is significantly lower than other final poses, and the display part of the interactions observed in the native bound state (X-ray structure). Thus, our simulation suggests that p53 is initially anchored to MDM2 by its N-terminal end. This observation agrees with recent experimental and MD simulations [40].

The main obstacle of reaching the final binding pose in our MC-VISM simulation is the lack of conformational flexibility. To further investigate this aspect, we used the prebound states produced by MC-VISM as starting points for explicit solvent MD simulations [5]. In those MD simulations, p53 quickly tucked the C-terminal tail within the MDM2 binding pocket and reach fully bound states. Hence, we showed that the binding poses predicted by rigid MC-VISM display interactions characteristic of a prebound state and easily lead to the crystallographic binding pose once the proteins are allowed some degree of conformational flexibility. This suggest that our rigid-body MC-VISM can be used to pre-sampling the binding phase space, and the results can be refined by MD simulation.

While our binary level-set method is fast, the solvation free energy is $\mathcal{O}(h)$ accurate, where h is the grid size in Å. Smaller h results in more accurate estimation of the solvation energy, but require longer computation time. For our MC-VISM simulation of p53-MDM2, we find that there is no visible differences between $h = 0.5$ Å and $h = 0.67$ Å. However, we find that there are less binding events observed if we used $h = 1$ Å. so we choose $h = 0.67$ Å in our simulation.

4.4 Conclusion

We show that our binary level-set VISM can efficiently capture the heterogeneous hydration states of the p53-MDM2 complex, and it's fast enough to be coupled with the rigid-body MC simulation of protein-protein interactions. Our extensive simulation

successfully captured some prebound states of the complex, and MD simulations starting from those configurations quickly reach the final bound state obtained from X-ray structure. Therefore, our method can be used as an efficient and reasonably accurate approach to pre-sample the binding poses, and the results can be refined by MD simulations.

There are a few direction where we can improve our method. Firstly, we need to include the Tolman correction of the surface energy and the Poisson-Boltzmann theory of the electrostatic energy into our binary level-set VISM. Secondly, we can incorporate water fluctuation into our model. Currently we are using the tight initial guess at every step. While in reality, the surface fluctuation between different local minima. Thirdly, we need relax our restriction of rigid-body MC move to allow for conformation change of the protein, which is important in protein binding.

Chapter 4, in part, is a reprint of the material Zirui Zhang, Clarisse G. Ricci, Chao Fan, Li-Tien Cheng, Bo Li, and J. Andrew McCammon. "Coupling Monte Carlo, Variational Implicit Solvation, and Binary Level-Set for Simulations of Biomolecular Binding". *Journal of Chemical Theory and Computation* (2021). The dissertation author was the primary investigator and author of the material

Chapter 5

A Compact Coupling Interface Method for Elliptic Interface Problems

5.1 Introduction

In this chapter, we introduce our Compact Coupling Interface Method (CCIM) for the following elliptic interface problem:

$$\begin{cases} -\nabla \cdot (\epsilon \nabla u) + au = f & \text{in } \Omega \setminus \Gamma, \\ [u] = \tau, \quad [\epsilon \nabla u \cdot \mathbf{n}] = \sigma & \text{on } \Gamma, \\ u = g & \text{on } \partial\Omega. \end{cases} \tag{5.1}$$

Here Γ is an interface that separates a regular computational domain $\Omega \subset \mathbb{R}^d$ into an inside region Ω^- and an outside region Ω^+. See Fig.5.1. $\mathbf{n}$ is the outward unit normal vector of the interface. g is the Dirichlet boundary condition. ϵ, f, a: $\Omega \to \mathbb{R}$ are given functions that might be discontinuous across Γ. τ, σ: $\Gamma \to \mathbb{R}$ are given interface jump conditions. For some function $v : \Omega \to \mathbb{R}$ and $x \in \Gamma$, denote the limiting value approaching from different side of the interface as

$$v^+(\mathbf{x}) = \lim_{\mathbf{x} \to \Gamma^+} v(\mathbf{x}), \quad v^-(\mathbf{x}) = \lim_{\mathbf{x} \to \Gamma^-} v(\mathbf{x}), \tag{5.2}$$

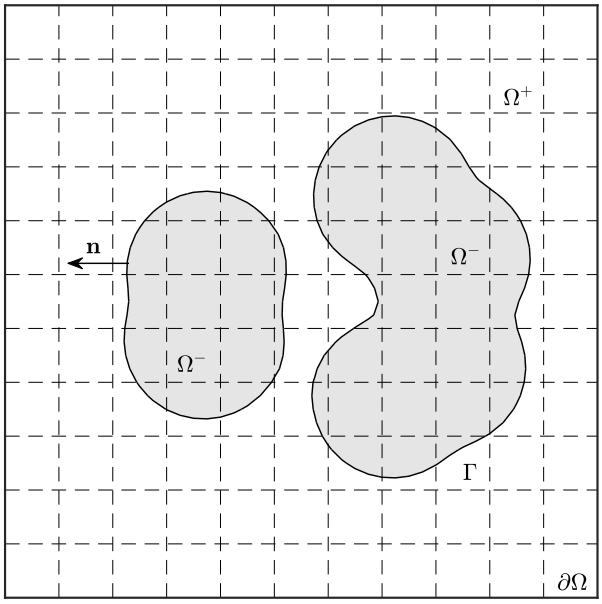


Figure 5.1. schematic for elliptic interface problem

and use the notation $[v]$ for the jump of v across the interface

$$[v] = v^+ - v^-. \tag{5.3}$$

The term $\epsilon \nabla v \cdot \mathbf{n}$ is sometime called the flux.

The elliptic interface problem arises from various physical and biological problems such as fluid dynamics, heat conduction, electrostatics, where material interfaces or phase boundaries are involved. The interface can be static or dynamic. In these problems, material properties and sources may be discontinuous across the interface, which lead to discontinuous solution or flux across the interface.

In some applications, the dynamics of the interface depends on gradient of the solution, therefore it is important to have accurate solutions and gradient. As explained in section 2.3, the velocity of the interface due to electrostatics depends on the jump of the normal derivative of the electrostatic potential at the interface. The Stefan problem [15] models the dynamic interface where one material is converted into the other. In this context, the interface velocity depends on the jump in normal derivative of the temperature.

We proposed a Compact Coupling Interface Method (CCIM). Our method combines elements of the CIM and Mayo's approach for elliptic interface boundary value problem[27]. The jump in each dimension is obtained by differentiating the jump condition in tangential direction. And the coupling equation involve the first-order derivative and principal second-order derivative. Our method is second order accurate in the solution and the gradient at the interface. The stencil is more compact, requiring a minimum of 10 grid points in three dimension. The compactness make the method applicable to more general situations and leads to more stable convergence results.

This chapter is organized as follows. Section 5.2 outlines the derivation and algorithm of our CCIM method. In section 5.3, we show the convergence tests in three dimensions on geometric surfaces and two complex protein surfaces. We also test our method on a moving surface driven by the jump in gradient at the interface. Section 5.4 is the conclusion.

5.2 Numerical Method

In d dimensions, let $\Omega = [-1, 1]^d$ and discretize the domain uniformly with mesh size $h = 2/N$, where N is the number of subintervals on one edge $[-1, 1]$ of the region Ω. Let $\mathbf{i} = (i_1, \ldots, i_d)$ be the multi-index with $i_k = 0, 1, \ldots, N$ for $k = 1, 2, \ldots, d$. The grid points are denoted as $\mathbf{x_i}$ with $\mathbf{x}_{i_k} = -1 + i_k h$. Let $\mathbf{e}_k$, $k = 1, 2, \ldots, d$ be the unit coordinate vectors. We also write $u(\mathbf{x_i}) = u_{\mathbf{i}}$. Here we use Δu for the Laplacian of u and $\nabla^2 u$ for the Hessian matrix of u. We assume that $\nabla^2 u$ is symmetric. We use $\overline{\mathbf{x_i x_{i+e_k}}}$ to denote the grid segment between $\mathbf{x_i}$ and $\mathbf{x_{i+e_k}}$, and assume that the interface intersect with any grid segment at most once.

Let $\mathbf{x_i}$ be a grid point at which we try to discretize the PDE. For notational simplicity, we drop the argument $\mathbf{x_i}$ and the dependency on $\mathbf{i}$ is implicit. We rewrite the

PDE (5.1) at $\mathbf{x_i}$ as

$$-\sum_{k=1}^{d}\frac{\partial\epsilon}{\partial x_k}\frac{\partial u}{\partial x_k}-\epsilon\sum_{j=1}^{d}\frac{\partial^2 u}{\partial x_k^2}+au=f \tag{5.4}$$

If $\mathbf{x_{i-e_k}}$, $\mathbf{x_i}$ and $\mathbf{x_{i+e_k}}$ are in the same region in each coordinate direction, then we call $\mathbf{x_i}$ a interior point, otherwise $\mathbf{x_i}$ is called a interface point. At interior points, standard central differencing gives a local truncation error of $\mathcal{O}(h^2)$ in $\mathbf{e}_k$ direction. Our goal is to construct finite difference schemes with $\mathcal{O}(h)$ local truncation error at interface points. The over all accuracy will still be second order since the interface points belongs to a lower dimensional set [23, 9].Next, we derive a first-order approximation for the term $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$.

5.2.1 Dimension-by-dimension discretization

Along coordinate direction $\mathbf{e}_k$, if the interface does not intersect the grid segment $\overline{\mathbf{x_i x_{i+e_k}}}$, then by Taylor expansion

$$u_{\mathbf{i+e}_k}-u_{\mathbf{i}}=h\frac{\partial u}{\partial x_k}+\frac{h^2}{2}\frac{\partial^2 u}{\partial x_k^2}+\mathcal{O}(h^3). \tag{5.5}$$

Suppose the interface intersects the grid segment $\overline{\mathbf{x_i x_{i+e_k}}}$ at $\hat{\mathbf{x}}_\mathbf{k}$. Let $\alpha_k=\|\hat{\mathbf{x}}_\mathbf{k}-\mathbf{x_i}\|/h$ and $\beta_k=1-\alpha_k$. Suppose $\mathbf{x_i}$ is located in Ω^-. Denote u^- the limit of $u(\mathbf{x})$ as $\mathbf{x}$ approaches $\hat{\mathbf{x}}_\mathbf{k}$ from Ω^-, u^+ as the limit from the other side. See Fig 5.2.

By Taylor expansion at the interface $\hat{\mathbf{x}}_\mathbf{k}$, we can express $u_\mathbf{i}$ and $u_{\mathbf{i+e}_k}$ as

$$\begin{aligned}u_{\mathbf{i}}&=u^--\alpha h\frac{\partial u^-}{\partial x_k}+\frac{(\alpha h)^2}{2}\frac{\partial^2 u^-}{\partial x_k^2}+\mathcal{O}(h^3)\\ u_{\mathbf{i+e}_k}&=u^++\beta h\frac{\partial u^+}{\partial x_k}+\frac{(\beta h)^2}{2}\frac{\partial^2 u^+}{\partial x_k^2}+\mathcal{O}(h^3)\end{aligned} \tag{5.6}$$

Subtract and write the RHS in terms of jumps and quantities from Ω^-:

$$u_{\mathbf{i+e}_k}-u_{\mathbf{i}}=[u]+\beta h\left[\frac{\partial u}{\partial x_k}\right]+h\frac{\partial u^-}{\partial x_k}+\frac{h^2}{2}\beta^2\left[\frac{\partial^2 u}{\partial x_k^2}\right]+\frac{h^2}{2}(\beta^2-\alpha^2)\frac{\partial^2 u^-}{\partial x_k^2}+\mathcal{O}(h^3). \tag{5.7}$$

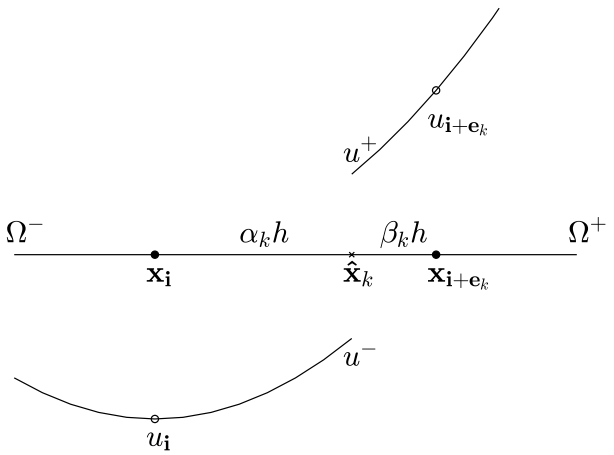


Figure 5.2. $u_\mathbf{i}$ and $u_{\mathbf{i}+\mathbf{e}_k}$ is approximated by Taylor expansion at interface

We can approximate components of ∇u^- and $\nabla^2 u^-$ by

$$\frac{\partial u^-}{\partial x_i} = \frac{\partial u}{\partial x_i} + \alpha_k h \frac{\partial^2 u}{\partial x_i \partial x_k} + \mathcal{O}(h^2), \tag{5.8}$$

$$\frac{\partial^2 u^-}{\partial x_i \partial x_k} = \frac{\partial^2 u}{\partial x_i \partial x_k} + \mathcal{O}(h). \tag{5.9}$$

Thus, (5.7) can be written as

$$u_{\mathbf{i}+\mathbf{e}_k} - u_\mathbf{i} = [u] + \beta h \left[\frac{\partial u}{\partial x_k}\right] + h\left(\frac{\partial u}{\partial x_k} + \alpha h \frac{\partial^2 u}{\partial x_k^2}\right) + \frac{h^2}{2}\beta^2 \left[\frac{\partial^2 u}{\partial x_k^2}\right] + \frac{h^2}{2}(\beta^2 - \alpha^2)\frac{\partial^2 u}{\partial x_k^2} + \mathcal{O}(h^3). \tag{5.10}$$

We denote the set of neighboring grid point of $\mathbf{i}$ as

$$B_r = \{\mathbf{j} \mid \|\mathbf{j} - \mathbf{i}\|_\infty \leq r\} \tag{5.11}$$

and call r the radius of our finite difference stencil. In three dimensions, B_1 contains 27 grid points forming a cube.

By the given jump condition, $[u] = \tau$. Suppose we can approximate the jump $[\partial u/\partial x_k]$ and $[\partial^2 u/\partial x_k^2]$ in terms of $u_\mathbf{j}$, $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$, with $1 \leq k \leq d$ and $\mathbf{j} \in B_r$

for some stencil radius r. Then in each coordinate direction, for $1 \le k \le d$, we can write down two equations, (5.10) or (5.5), by considering the two grid segments $\overline{\mathbf{x_i x_{i+se_k}}}$ for $s = \pm 1$. In d dimensions we have $2d$ equations and $2d$ unknowns: the first-order derivatives $\partial u/\partial x_k$ and principle second-order derivatives $\partial^2 u/\partial x_k^2$ for $1 \le k \le d$. This leads to a system of linear equations of the following form:

$$M \begin{bmatrix} \frac{\partial u}{\partial x_k} \\ \frac{\partial^2 u}{\partial x_k^2} \end{bmatrix}_{1 \le k \le d} = \frac{1}{h^2} \left[D_{k,s} u + b_{k,s} \right]_{1 \le k \le d, s = \pm 1} + \mathcal{O}(h). \tag{5.12}$$

where $D_{k,s} u = D_{k,s}(u_{\mathbf{j}})$, $\mathbf{j} \in B_r$, is some linear function of neighboring u-values and $b_{k,s}$ is some constant. We call (5.12) coupling equation and M coupling matrix. By inverting M, we attain the finite difference formula to approximate $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$ in terms of u-values.

In the next section, we describe how to remove the jumps in (5.10).

5.2.2 Approximation of the jump condition

Let $\mathbf{n}$ be the unit normal vector at the interface, and $\mathbf{s}_1, \ldots, \mathbf{s}_{d-1}$ be unit tangent vectors. The tangent vectors can be obtained by projecting the unit coordinate vector onto the tangent plane. We can write

$$[\nabla u] = [\nabla u \cdot \mathbf{n}]\mathbf{n} + \sum_{j=1}^{d-1} [\nabla u \cdot \mathbf{s}_j]\mathbf{s}_j. \tag{5.13}$$

In the $\mathbf{e}_k$ coordinate direction, this gives

$$\left[\frac{\partial u}{\partial x_k} \right] = [\nabla u \cdot \mathbf{n}](\mathbf{n} \cdot \mathbf{e}_k) + \sum_{j=1}^{d-1} [\nabla u \cdot \mathbf{s}_j](\mathbf{s}_j \cdot \mathbf{e}_k). \tag{5.14}$$

We use the trick repeatedly in our derivation to decouple the jump

$$[\epsilon v] = \epsilon^+[v] + [\epsilon]v^-. \tag{5.15}$$

Taking $v = \nabla u \cdot \mathbf{n}$, together with with $[\nabla u \cdot \mathbf{s}_k] = \nabla \tau \cdot \mathbf{s}_k$, the jump condition $[\epsilon \nabla u \cdot \mathbf{n}] = \sigma$ can be rewritten as

$$\left[\frac{\partial u}{\partial x_k}\right] = \frac{1}{\epsilon^+}(\sigma - [\epsilon]\nabla u^- \cdot \mathbf{n})(\mathbf{n} \cdot \mathbf{e}_k) + \sum_{j=1}^{d-1}(\nabla \tau \cdot \mathbf{s}_n)(\mathbf{s}_n \cdot \mathbf{e}_k). \tag{5.16}$$

By (5.8), we get

$$\left[\frac{\partial u}{\partial x_k}\right] = \frac{1}{\epsilon^+}\left(\sigma - [\epsilon]\sum_{j=1}^{d}\left(\frac{\partial u}{\partial x_j} + \alpha_k \frac{\partial^2 u}{\partial x_j \partial x_k}\right)(\mathbf{n} \cdot \mathbf{e}_j)\right)(\mathbf{n} \cdot \mathbf{e}_k) + \sum_{j=1}^{d-1}(\nabla \tau \cdot \mathbf{s}_n)(\mathbf{s}_n \cdot \mathbf{e}_k). \tag{5.17}$$

Notice that now we approximate the jump in the first-order derivative at the interface in terms of the first-order and second-order derivatives at grid point.

To remove the jump in the principal second-order derivative $[\partial^2 u/\partial x_k^2]$, we solve a system of linear equations. The equations are obtained by differentiating the interface boundary condition in tangential directions. For $m = 1, \cdots, d-1$ and $n = m, \cdots, d-1$, we get $d(d-1)/2$ equations

$$\nabla[\nabla u \cdot \mathbf{s}_m] \cdot \mathbf{s}_n = \nabla(\nabla \tau \cdot \mathbf{s}_m) \cdot \mathbf{s}_n. \tag{5.18}$$

After expansion,

$$\mathbf{s}_n^T[\nabla^2 u]\mathbf{s}_m = \mathbf{s}_n^T \nabla^2 \tau \mathbf{s}_m + \frac{1}{\epsilon^+}(\sigma - [\epsilon]\nabla u^- \cdot \mathbf{n})\mathbf{s}_n^T \nabla \mathbf{n} \mathbf{s}_m - (\nabla \tau \cdot \mathbf{n})\mathbf{s}_n^T \nabla \mathbf{n} \mathbf{s}_m. \tag{5.19}$$

By differentiating the jump in flux in tangential direction, we get another $d-1$

equations for $m = 1, \cdots, d-1$,

$$\nabla[\epsilon \nabla u \cdot \mathbf{n}] \cdot \mathbf{s}_m = \nabla \sigma \cdot \mathbf{s}_m \tag{5.20}$$

After expansion,

$$\begin{aligned} \mathbf{s}_m^T [\nabla^2 u] \mathbf{n} = & \frac{1}{\epsilon^+} \nabla \sigma \cdot \mathbf{s}_m - \frac{[\epsilon]}{\epsilon^+} \mathbf{s}_m^T \nabla^2 u^- \mathbf{n} - \frac{[\epsilon]}{\epsilon^+} \mathbf{s}_m^T \nabla \mathbf{n} \nabla u^- - \sum_{k=1}^{d-1} (\nabla \tau \cdot \mathbf{s}_k) \mathbf{s}_m^T \nabla \mathbf{n} \mathbf{s}_k \\ & - \frac{1}{(\epsilon^+)^2} (\nabla \epsilon^+ \cdot \mathbf{s}_m)(\sigma - [\epsilon] \nabla u^- \cdot \mathbf{n}) - \frac{1}{\epsilon^+} [\nabla \epsilon \cdot \mathbf{s}_m](\nabla u^- \cdot \mathbf{n}). \end{aligned} \tag{5.21}$$

Together with the PDE

$$[-\nabla \cdot (\epsilon \nabla u) + au] = [f]. \tag{5.22}$$

After expansion,

$$\begin{aligned} & = - \left[\frac{f}{\epsilon} \right] + \frac{a^+}{\epsilon^+} \tau + \left[\frac{a}{\epsilon} \right] u^- \\ & - \frac{1}{(\epsilon^+)^2} (\sigma - [\epsilon] \nabla u^- \cdot \mathbf{n})(\nabla \epsilon^+ \cdot \mathbf{n}) - \frac{1}{\epsilon^+} \sum_{k=1}^{d-1} (\nabla \tau \cdot \mathbf{s}_k)(\nabla \epsilon^+ \cdot \mathbf{s}_k) - \left[\frac{\nabla \epsilon}{\epsilon} \right] \cdot \nabla u^-. \end{aligned} \tag{5.23}$$

We arrive at a system of linear equations of where the variables are the jump in second-order derivatives

$$G \left[\left[\frac{\partial^2 u}{\partial x_k \partial x_l} \right] \right]_{1 \le k \le l \le d} = L \left(u^-, \left(\frac{\partial u^-}{\partial x_k} \right)_{1 \le k \le d}, \left(\frac{\partial^2 u^-}{\partial x_k \partial x_l} \right)_{1 \le k \le l \le d} \right) + b. \tag{5.24}$$

where G is a matrix that only depends on the normal and tangent vectors, L stands for some general linear function and b stands for some constant. In two and three dimensions, through a direct calculation, the absolute value of the determinant of G is 1.

As an example, in two dimensions, let $\mathbf{s} = [s_1, s_2]^T$ and $\mathbf{n} = [n_1, n_2]^T$, and assume

that $\epsilon(\mathbf{x})$ is a piecewise constant function, then the system of linear equations is given by

$$\begin{bmatrix} s_1^2 & s_2^2 & 2s_1s_2 \\ s_1n_1 & s_2n_2 & s_1n_2+s_2n_1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} [u_{xx}] \\ [u_{yy}] \\ [u_{xy}] \end{bmatrix} = \\ \begin{bmatrix} \mathbf{s}^T\nabla^2\tau\mathbf{s} + \frac{1}{\epsilon^+}\left(\sigma - [\epsilon]\nabla u^- \cdot \mathbf{n}\right)\mathbf{s}^T\nabla\mathbf{n}\mathbf{s} - (\nabla\tau\cdot\mathbf{n})\mathbf{s}^T\nabla\mathbf{n}\mathbf{s} \\ \frac{1}{\epsilon^+}\nabla\sigma\cdot\mathbf{s} - \frac{[\epsilon]}{\epsilon^+}(\mathbf{s}^T\nabla^2u^-\mathbf{n} + \mathbf{s}^T\nabla\mathbf{n}\nabla u^-) - (\nabla\tau\cdot\mathbf{s})\mathbf{s}^T\nabla\mathbf{n}\mathbf{s} \\ -[\frac{f}{\epsilon}] + \frac{a^+}{\epsilon^+}\tau - [\frac{a}{\epsilon}]u^- \end{bmatrix}. \tag{5.25}$$

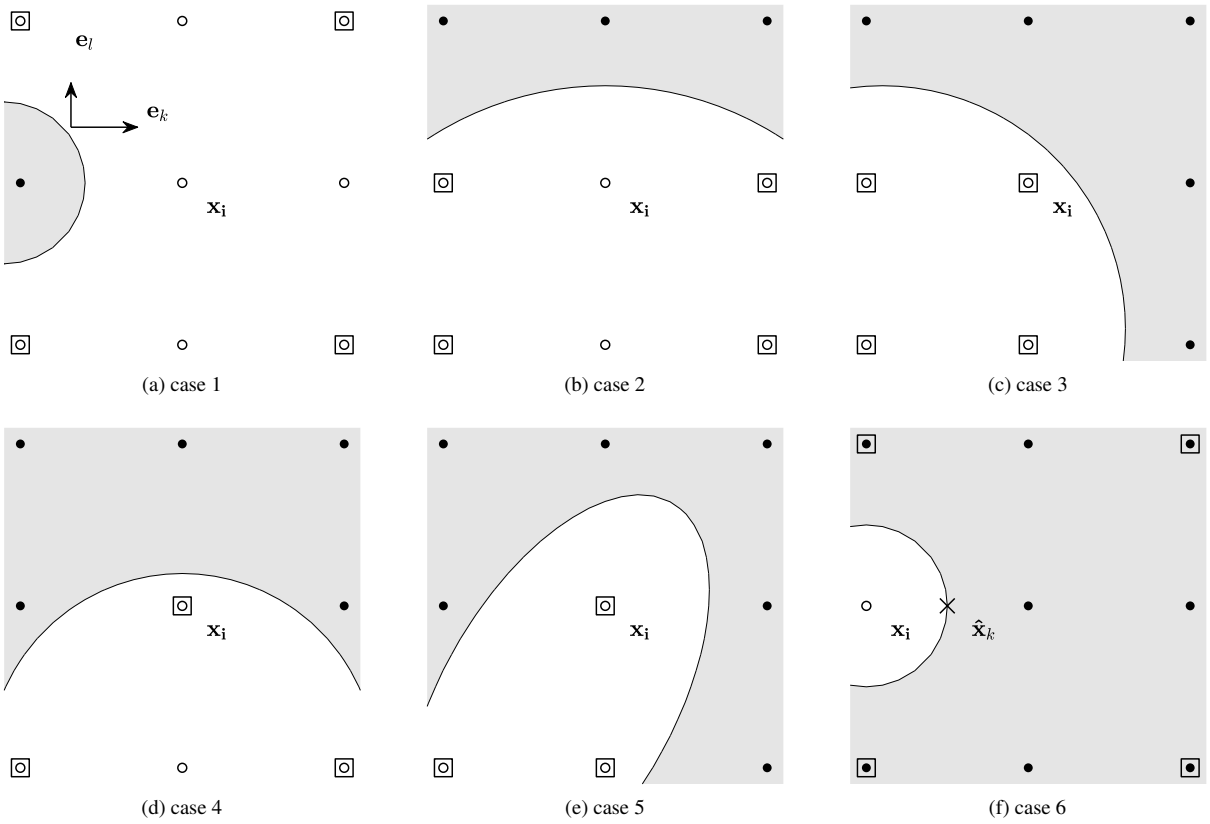
By Taylor's expansion as in (5.5) (5.8), and (5.9), u^- and components of ∇u^- and $\nabla^2 u^-$ can all be approximated by u and components of ∇u and $\nabla^2 u$ at the grid point. Therefore, after substitution, with different L and b, (5.24) becomes

$$G\left[\left[\frac{\partial^2 u}{\partial x_k \partial x_l}\right]\right]_{1\le k\le l\le d} = L\left(u, \left(\frac{\partial u}{\partial x_k}\right)_{1\le k\le d}, \left(\frac{\partial^2 u}{\partial x_k\partial x_l}\right)_{1\le k\le l\le d}\right) + b + \mathcal{O}(h). \tag{5.26}$$

If we could approximate the mixed derivatives $\partial^2 u/\partial x_k\partial x_l$, $k \neq l$ in terms of u-values and the jump in second-order derivatives, then by inverting G, the jump in second-order derivatives can be approximated by u-values, first-order derivatives, and principal second-order derivatives, which are the terms used in the coupling equation. By back substitution, the mixed derivatives and thus the jump in the first-order derivative can also be approximated by these terms. Next, we describe how to approximate the mixed derivatives.

5.2.3 Approximation of the mixed derivative

Depending on the position of the interface with respect to the grid points, different schemes are needed in order to approximate $\frac{\partial^2 u}{\partial x_k\partial x_l}$, $k \neq l$. In Fig. 5.3 case 1, we use the



(a) case 1 (b) case 2 (c) case 3

(d) case 4 (e) case 5 (f) case 6

Figure 5.3. Approximation of mixed derivative at x_i. The circles and disks are grid points in Ω_- and Ω_+. The squares are grid points that are used to approximate the mixed derivative.

usual central difference formula,

$$\frac{\partial^2 u}{\partial x_k \partial x_l} = \frac{1}{4h^2}\left(u_{\mathbf{i}+\mathbf{e}_k+\mathbf{e}_l} - u_{\mathbf{i}-\mathbf{e}_k+\mathbf{e}_l} - u_{\mathbf{i}+\mathbf{e}_k-\mathbf{e}_l} + u_{\mathbf{i}-\mathbf{e}_k-\mathbf{e}_l}\right) + \mathcal{O}(h^2). \tag{5.27}$$

We can also use biased differencing as in Fig. 5.3 case 2 and case 3

$$\begin{aligned}\frac{\partial^2 u}{\partial x_k \partial x_l} &= \frac{1}{2h^2}\left(u_{\mathbf{i}+\mathbf{e}_k} - u_{\mathbf{i}+\mathbf{e}_k-\mathbf{e}_l} - u_{\mathbf{i}-\mathbf{e}_k} + u_{\mathbf{i}-\mathbf{e}_k-\mathbf{e}_l}\right) + \mathcal{O}(h) \\ &= \frac{\partial^2 u}{\partial x_k \partial x_l}(\mathbf{x}_\mathbf{i} - \frac{1}{2}\mathbf{e}_l) + \mathcal{O}(h^2),\end{aligned} \tag{5.28}$$

$$\begin{aligned}\frac{\partial^2 u}{\partial x_k \partial x_l} &= \frac{1}{h^2}(u_\mathbf{i} - u_{\mathbf{i}-\mathbf{e}_k} - u_{\mathbf{i}-\mathbf{e}_l} + u_{\mathbf{i}-\mathbf{e}_k-\mathbf{e}_l}) + \mathcal{O}(h) \\ &= \frac{\partial^2 u}{\partial x_k \partial x_l}(\mathbf{x}_\mathbf{i} - \frac{1}{2}h\mathbf{e}_k - \frac{1}{2}h\mathbf{e}_l) + \mathcal{O}(h^2).\end{aligned} \tag{5.29}$$

In case 4, we can make use of the first-order derivatives

$$\begin{aligned}\frac{\partial^2 u}{\partial x_k \partial x_l} &= \frac{1}{2h^2}\left(2h\frac{\partial u}{\partial x_k} - u_{\mathbf{i}+\mathbf{e}_k-\mathbf{e}_l} + u_{\mathbf{i}-\mathbf{e}_k-\mathbf{e}_l}\right) + \mathcal{O}(h) \\ &= \frac{\partial^2 u}{\partial x_k \partial x_l}(\mathbf{x}_\mathbf{i} - \frac{1}{2}h\mathbf{e}_l) - \frac{1}{6}h\frac{\partial^4 u}{\partial x_k^4} + \mathcal{O}(h^2).\end{aligned} \tag{5.30}$$

In case 5, we can make use of the first-order and second-order derivatives

$$\begin{aligned}\frac{\partial^2 u}{\partial x_k \partial x_l} &= \frac{1}{h^2}\left(h\frac{\partial u}{\partial x_k} - \frac{h^2}{2}\frac{\partial^2 u}{\partial x_k^2} - u_{\mathbf{i}-\mathbf{e}_l} - u_{\mathbf{i}-\mathbf{e}_k-\mathbf{e}_l}\right) + \mathcal{O}(h) \\ &= \frac{\partial^2 u}{\partial x_k \partial x_l}(\mathbf{x}_\mathbf{i} - \frac{1}{2}h\mathbf{e}_l) - \frac{1}{6}h\frac{\partial^3 u}{\partial x_k^3} + \mathcal{O}(h^2).\end{aligned} \tag{5.31}$$

In Fig. 5.3 case 6, when there are not enough grid points on the same side, we can make use of information from the other side of the interface and the jump in mixed derivative

$$\frac{\partial^2 u}{\partial x_k \partial x_l} = \frac{\partial^2 u}{\partial x_k \partial x_l}(\mathbf{x}_\mathbf{i} + h\mathbf{e}_k) - \left[\frac{\partial^2 u}{\partial x_k \partial x_l}\right] + \mathcal{O}(h). \tag{5.32}$$

By approximating the mixed derivatives $\partial^2 u/\partial x_k \partial x_l$, $1 \le k < l \le d$ in terms of

u_N, $\partial u/\partial x_k$, $\partial^2 u/\partial x_k^2$ and $[\partial^2 u/\partial x_k \partial x_l]$, where $u_{\mathbf{j}}$, $\mathbf{j} \in B_r$ for some kernel radius r, we can write the right hand side of (5.24) in terms of $u_{\mathbf{j}}$, $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$, $1 \leq k \leq d$, thus eliminating the mixed derivatives. Then by solving the linear system (5.24), the jump in second derivative $[\partial^2 u/\partial x_k^2]$ can be approximated in terms of $u_{\mathbf{j}}$, $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$, $1 \leq k \leq d$.

Finally, substituting $[\partial u/\partial x_k]$, $[\partial^2 u/\partial x_k^2]$ into (5.7), we get an equation involving only $u_{\mathbf{j}}$, $\partial u/\partial x_k$, and $\partial^2 u/\partial x_k^2$ for $1 \leq k \leq d$. In each dimension, we have 2 equations (by looking forward and backward). Therefore, with $2d$ equations and $2d$ unknowns (the first and second order derivatives), we can solve for $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$ in terms of u-values.

Multiple methods will be available at the same grid point. We would like the method to be simple and accurate. For simplicity, we prefer method that only make use of u-value, as in case 1, 2 and 3. For accuracy, we provide a heuristic criteria: for homogeneous polynomial of degree three, the approximation on the right hand side is exact for some nearby points $\mathbf{x_i} - \frac{1}{2}h\mathbf{e}_k - \frac{1}{2}h\mathbf{e}_l$ or $\mathbf{x_i} - \frac{1}{2}h\mathbf{e}_l$, We would like this point to be as close to $\mathbf{x_i}$ as possible. In this respect, we prefer case 4 to case 3. Another consideration for accuracy is the condition number of the coupling equation. Solving a linear system with large condition number is prone to large numerical errors. Therefore, in cases where both case 3 (5.29) and case 4 (5.30) are available, we choose the method with a smaller estimated condition number [17].

Though we can construct surfaces for specific grid size such that none of the above schemes works, for smooth surfaces we can refine the grid such that the above schemes suffice. In addition, we note that case 5 and case 6 can be removed by refining the grid, while case 4 can not [32].

5.2.4 Algorithm

We describe our method to attain the coupling equations at an interface point in algorithmic order. By inverting the coupling matrix, the $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$, $1 \leq k \leq d$

can be approximated by a linear combination of $u_{\mathbf{j}}$, $\mathbf{j} \in B_r$ and some constant. These expressions gives one row of the final linear system.

Algorithm 5.2.1: Coupling equation at interface point $\mathbf{x_i}$

```
for 1 ≤ k ≤ d do
    for s = ±1 do
        if the interface intersects x_i x_{i+s e_k} at x̂_k then
            for 1 ≤ j ≤ d, j ≠ k do
                Approximate the mixed derivative ∂²u/∂x_k∂x_j in terms of u_j,
                 j ∈ B_r and [∂²u/∂x_k∂x_j]. If Fig.5.3 case 6 (using information
                 from the other side) is used, then r = 2 and the coefficient of
                 [∂²u/∂x_k∂x_j] is non-zero. Otherwise r = 1 and the term
                 [∂²u/∂x_k∂x_j] vanishes ;
            end
            Differentiate the jump condition and obtain a system of equations of
             the jump in second derivatives (5.24). Substitute the
             approximation of the mixed derivatives. Invert G, then the jumps
             in second derivatives [∂²u/∂x_k∂x_j], 1 ≤ j ≤ d, are approximated
             in terms of u_j, j ∈ B_r, ∂u/∂x_j, ∂²u/∂x_j², 1 ≤ j ≤ d.;
            By back substitution, the mixed derivative ∂²u/∂x_k∂x_j and thus
             [∂u/∂x_k] (5.17) can be expressed in terms of u_j, j ∈ B_r, ∂u/∂x_j,
             ∂²u/∂x_j², 1 ≤ j ≤ d.;
            Substitute the expression for [∂u/∂x_k] and [∂²u/∂x_k²] into (5.10).
             After rearrangement, this gives one row of the coupling equations
             (5.12).;
        else
            Taylor's expansion (5.5) gives one row of the coupling equations
             (5.12).;
        end
    end
end
```

To get more stable convergence results, at grid points where case 1 and 2 not available, but case 3 and 4 are available, we use the above algorithm to obtain two coupling equations, and choose the coupling equation with a smaller estimated condition number of the coupling matrix. The effect of this criterion is demonstrated in section 5.3.1.

5.3 Numerical Results

We test our method in three dimensions with different surfaces. The first sets of test contains six geometric surfaces. And the second set of tests consists of two complex biomolecular surface. These two sets are compared with ICIM [32] with the same setup. As tests in [32] does not include $a(\mathbf{x})$ term, the third set of tests are the same six geometric surfaces with $a(\mathbf{x})$ term. The last test is a sphere expanding under a normal velocity given by the derivative of the solution in normal direction. Let u_e be the exact solution of (5.1), and u be the numerical solution. For tests with a static interface, we look at the maximum error of the solution at all grid points, denoted as $\|u_e - u\|_\infty$, and the maximum error of the gradient at all intersection of interface and grid lines, denoted as $\|\nabla u_e - \nabla u\|_{\infty,\Gamma}$. For the expanding sphere, we look at the maximum error and Root Mean Square Error of the radius at all intersections of the interface and grid line. All the tests are performed on a 2017 iMac with 3.5 GHz Intel Core i5 and 16GB memory. We use AMG implemented by the HYPRE library [13] to solve the sparse linear system to a tolerance of 10^{-9}.

5.3.1 Example 1

We test several geometric interfaces as in [32]. The surfaces are shown in Fig.5.4. Their level set functions are given below.

- Eight balls: $\phi(x,y,z) = \min_{0\le k\le 7} \sqrt{(x-x_k)^2+(y-y_k)^2+(z-z_k)^2} - 0.3$, where $(x_k, y_k, z_k) = ((-1)^{\lfloor k/4 \rfloor} \times 0.5, (-1)^{\lfloor k/2 \rfloor} \times 0.5, (-1)^k \times 0.5)$
- Ellipsoid: $\phi(x,y,z) = 2x^2 + 3y^2 + 6z^2 - 1.3$
- Peanut: $\phi(x,y,z) = \phi(r,\theta,\psi) = r - 0.5 - 0.2\sin(2\theta)\sin(\psi)$
- Donut: $\phi(x,y,z) = (\sqrt{x^2+y^2} - 0.6)^2 + z^2 - 0.4^2$
- Banana: $\phi(x,y,z) = (7x+6)^4 + 2401y^4 + 3601.5z^4 + 98(7x+6)^2(y^2+z^2) + 4802y^2z^2 - 94(7x+6)^2 + 3822y^2 - 4606z^2 + 1521$

- Popcorn:

$$\phi(x,y,z) = \sqrt{x^2+y^2+z^2} - r_0 - \sum_{k=0}^{11} \exp\big(25((x-x_k)^2+(y-y_k)^2+(z-z_k)^2)\big)$$

where

$$\begin{aligned}(x_k,y_k,z_k) &= \frac{r_0}{\sqrt{5}}\left(2\cos\left(\frac{2k\pi}{5}-\left\lfloor\frac{k}{5}\right\rfloor\right), 2\sin\left(\frac{2k\pi}{5}-\left\lfloor\frac{k}{5}\right\rfloor\right), (-1)^{\left\lfloor\frac{k}{5}\right\rfloor}\right), 0\le k\le 9\\ &= r_0(0,0,(-1)^{k-10}), 10\le k\le 11.\end{aligned}$$

The exact solution and the coefficients are given by

$$u_e(x,y,z) = \begin{cases} xy + x^4 + y^4 + xz^2 + \cos(2x+y^2+z^3) & \text{if } (x,y,z)\in\Omega^+ \\ x^3 + xy^2 + y^3 + z^4 + \sin(3(x^2+y^2)) & \text{if } (x,y,z)\in\Omega^- \end{cases} \tag{5.33}$$

and

$$\epsilon(x,y,z) = \begin{cases} 80 & \text{if } (x,y,z)\in\Omega^+ \\ 2 & \text{if } (x,y,z)\in\Omega^- \end{cases}. \tag{5.34}$$

The source term and the jump conditions are calculated accordingly.

Fig 5.5 shows the convergence result of the six interfaces. The convergence of the solutions at grid points is second order, and the convergence of the gradient at the interface is slightly below second order.

Next we demonstrate the effect of choosing the stencil for mixed derivative based on estimated condition number. As mentioned in section 5.2.3, when both case 3 and case 4 are available to approximate the mixed derivative, we choose the method that gives a smaller estimated condition number of the coupling matrix. We denote this scheme CCIM. Alternatively, we can fix the order of preference. We call it scheme 1 the method that always prefer case 4 to case 3. Fig. 5.6 and 5.7 demonstrate the effect of this decision

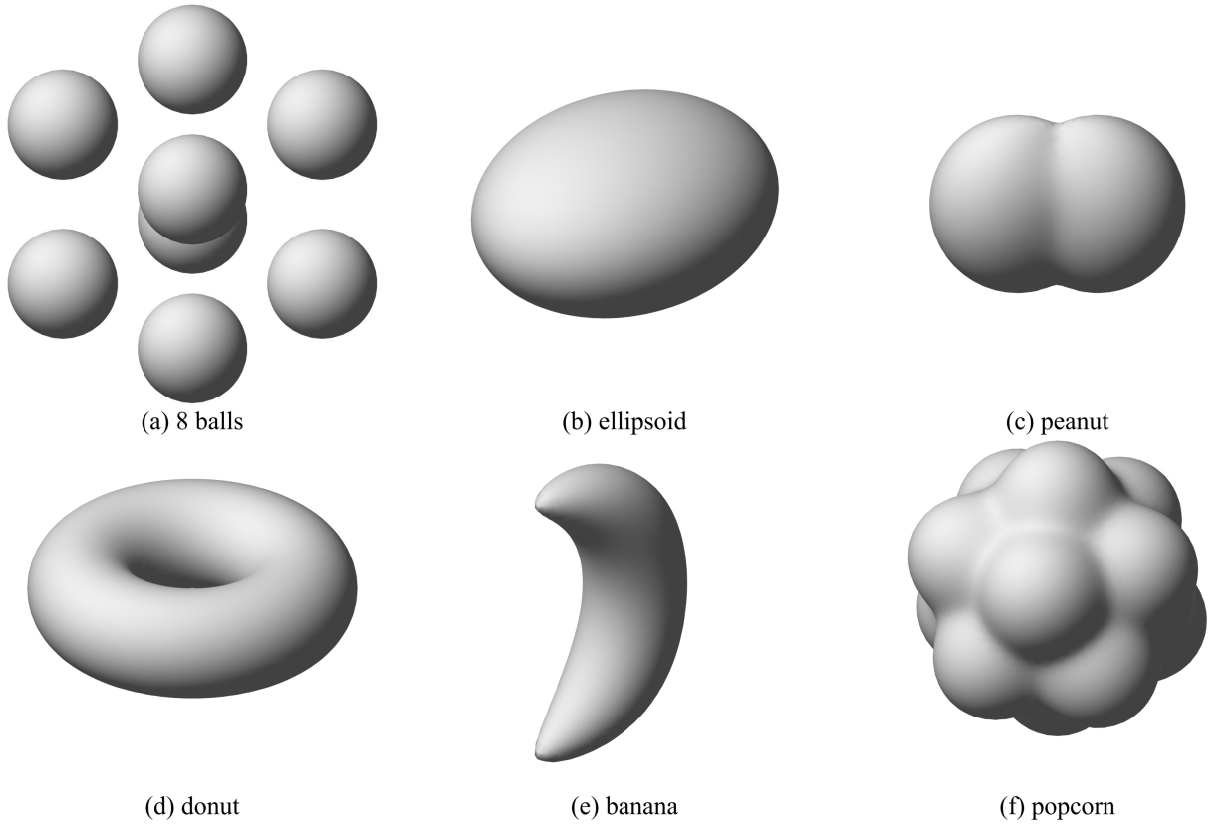
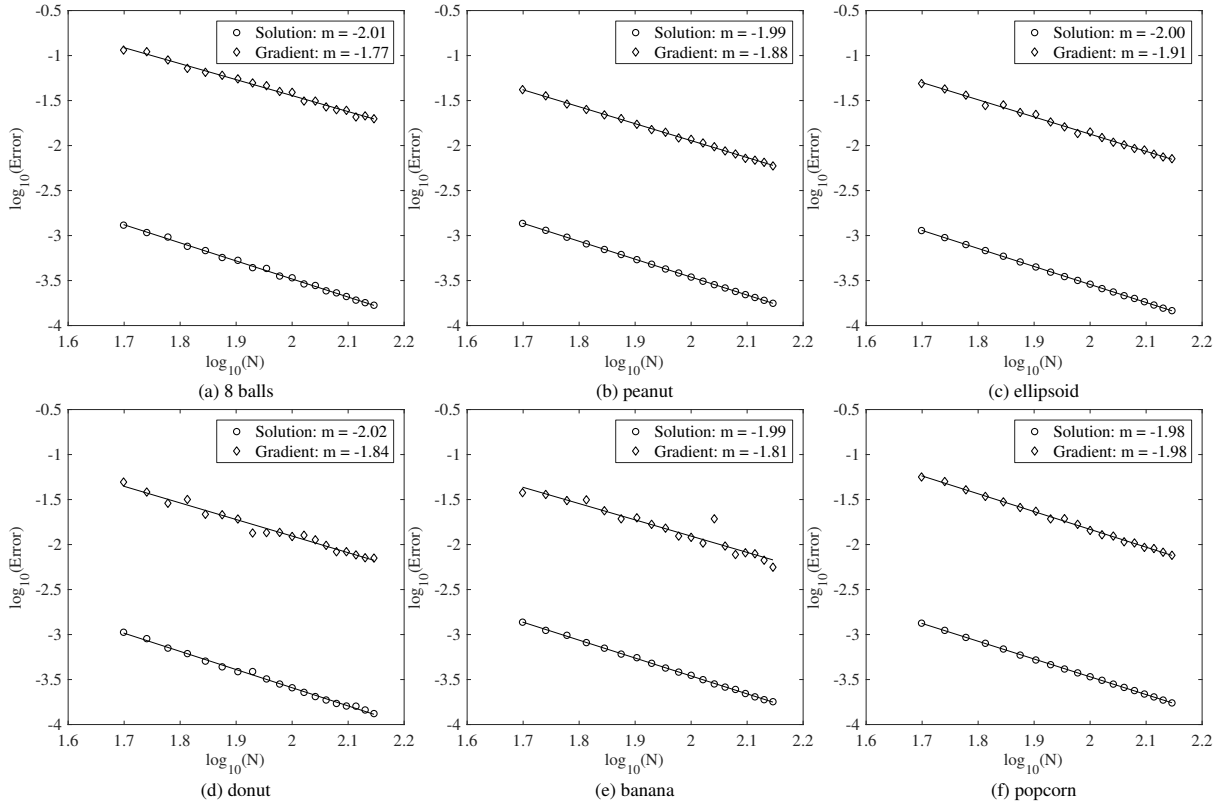


Figure 5.4. The six interfaces (a) eight balls; (b) ellipsoid; (c) peanut; (d) donut; (e) banana; (f) popcorn



(a) 8 balls (b) peanut (c) ellipsoid

(d) donut (e) banana (f) popcorn

Figure 5.5. The log-log plot of the error versus N for the six surfaces. In each figure, N ranges from 50 to 140 with the increment $\Delta N = 5$. Circles are the maximum errors of the solution $\|u_e - u\|_\infty$. Diamonds are the maximum errors of the gradient at interface $\|\nabla u_e - \nabla u\|_{\infty,\Gamma}$.

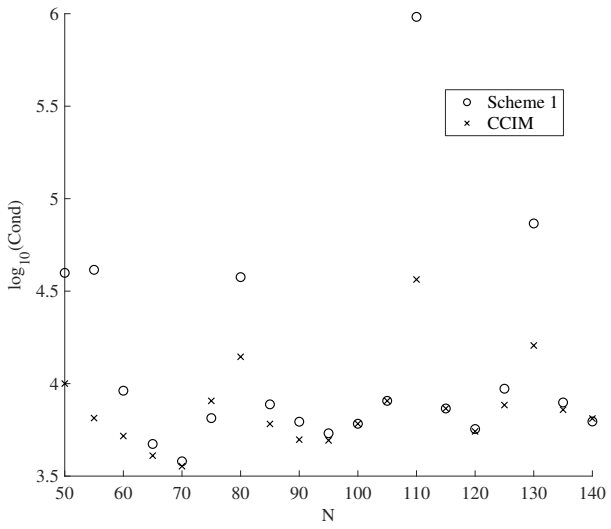


Figure 5.6. The maximum condition number of the coupling matrices with banana surface using scheme 1 and CCIM

using the banana shape surface as an example. For different N and for both schemes, Fig. 5.6 plots the maximum condition number of all the coupling matrices, and Fig. 5.6 plots the convergence results of these two scheme. For most of the N-values, CCIM and scheme 1 have roughly the same maximum error. We noticed that for $N = 110$, with scheme 1, at the interface point that produces the maximum error in the gradient, the coupling matrix has an exceptionally large condition number. By choosing the method with a smaller estimated condition number, we get a more stable convergence result in the gradient. If we prefer case 3 to case 4, then results are similar: at some grid points large condition number leads to large error, and CCIM gives more stable convergence results.

Though we are able to get a more stable convergence result by considering the condition number of the coupling matrices, there is a small jump for the banana interface at $N = 110$. A detailed analysis of the error reveals that it was caused by relatively large local truncation error when approximating u_{xz}. Fig. 5.8 shows that contour plot of the mixed derivative u_{xz}, the grid, the surface, and the stencil for approximating the mixed derivative. Notice that u_{xz} change rapidly along the northeast direction. However, due to the alignment of the surface with the grid, at $x_{i,k}$, our algorithm use the 4-point stencil

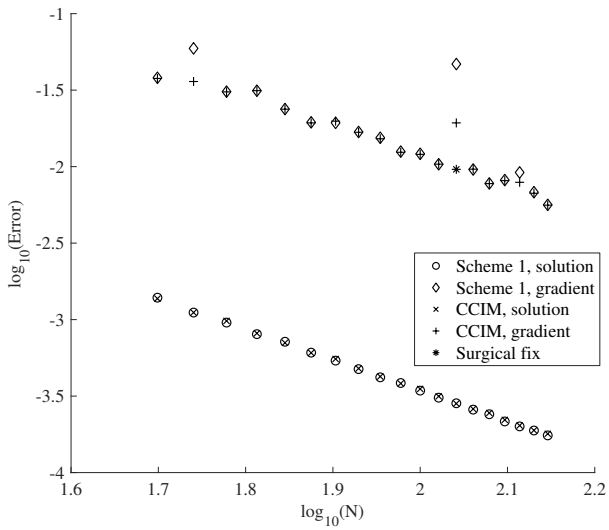


Figure 5.7. log-log plot of the maximum error in solution and gradient with banana surface

$\mathbf{x}_{i,k}$, $\mathbf{x}_{i-1,k}$, $\mathbf{x}_{i,k-1}$ and $\mathbf{x}_{i-1,k-1}$ to approximate $u_{xz}(\mathbf{x}_{i,k})$ and has a local truncation error 0.160723. If we use the three point stencil $\mathbf{x}_{i,k}$, $\mathbf{x}_{i-1,k}$, $\mathbf{x}_{i-1,k+1}$, the local truncation error would be 0.041853, and the coupling matrix does not have large condition number. With this surgical fix, the final error would be in line with the rest of the data points as shown in Fig 5.7 at N = 110, marked as "Surgical fix". This type of outliers happens rarely and does not affect the overall order of convergence. We apply this surgical fix only at this specific grid point to demonstrate a possible source of large error.

In summary, though the overall order of convergence is second order no mater which method is used to approximate the mixed derivatives, a relatively large error can be caused by large condition number of the coupling matrix, or a large local truncation error when approximating the mixed derivative. When different method to approximate the mixed derivative are available, ideally we prefer the method that produces smaller local truncation error and smaller condition number of the coupling matrix. However these two goal might be incompatible sometimes. It's time consuming to search through all available methods to approximate the mixed derivative and find the one that leads a small condition number of the coupling matrix. It's also difficult to tell a priori which stencil

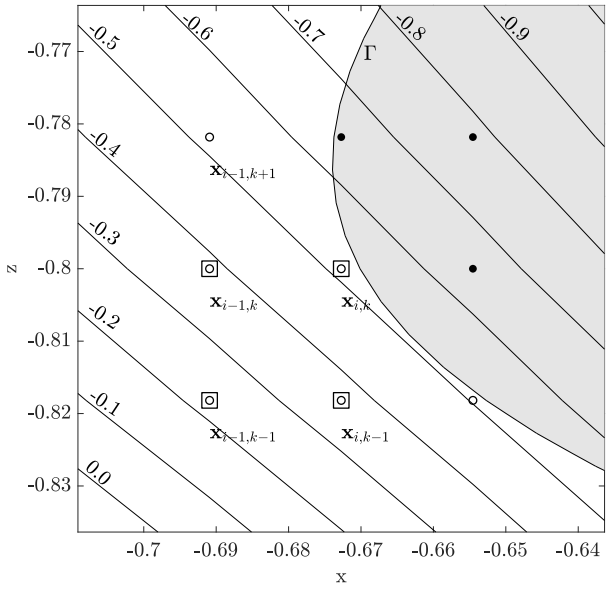


Figure 5.8. Contour of u_{xz} at the grid point with maximum error in gradient.

gives smaller local truncation error. Therefore we try to find a middle ground by only considering the condition number when both case 3 and case 4 are available.

The resulting linear system is sparse and asymmetric, and can be solved with any "black-box" linear solvers. Fig. 5.9 shows the log-log plot for the number of iteration versus N. We used BiCGSTAB with ILU preconditioner and Algebraic Multigrid Method (AMG). The number of iteration grows linearly with N for BICGSTAB and sub-linearly for AMG. Though AMG has better scaling properties, for the range of N in Fig 5.9, both solvers take approximately the same CPU time.

5.3.2 Example 2

Next we test our method on two complex molecular surfaces and compare our CCIM with our implementation of ICIM[32]. The solvent accessible surface describe the interface between solute and solvent. Such interfaces are complex and important in applications. We construct the surfaces as in [32]: from the PDB file of 1D63 [4] and MDM2 [22], we use the PDB2PQR[10] software to assign charges and radii with AMBER force field. The PQR file contains information of the positions $\mathbf{p_i}$ and radii r_i of the atoms. We scale the

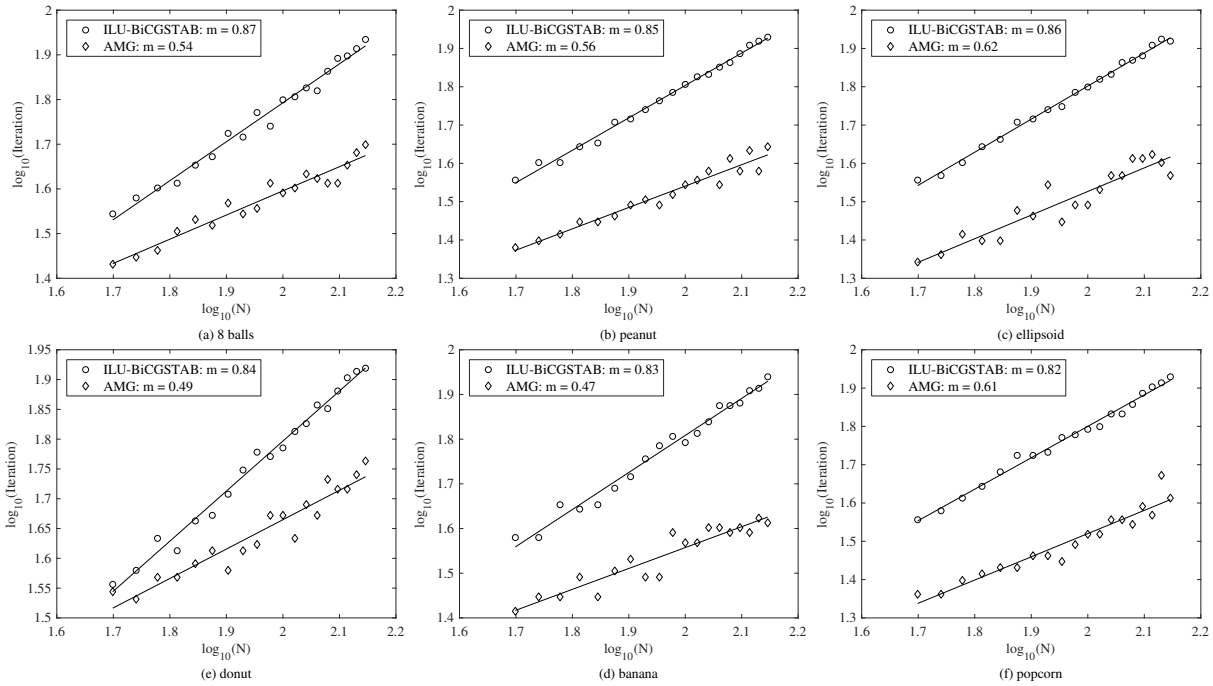


Figure 5.9. The log-log plot of the number of iterations versus N for the six surfaces

positions and radii such that the protein fit into the unit box. Then we construct the level set function as union of smoothed bumps:

$$\phi(\mathbf{x}) = c - \sum_i \chi_\eta(r_i - \|\mathbf{x} - \mathbf{p_i}\|) \tag{5.35}$$

where χ_η is a smoothed characteristic function

$$\chi_\eta = \frac{1}{2}\left(1 + \tanh\left(\frac{x}{\eta}\right)\right) \tag{5.36}$$

The molecular surfaces 1D63 has 486 atoms, and the surface has a double-helix shape, as shown in Fig. 5.10a. MDM2 has 1448 atoms, and the surface has a deep pocket to which other proteins can bind, as shown in Fig 5.11a. We also implement ICIM [32] and compare the convergence results between CCIM and ICIM in Fig. 5.10 and Fig. 5.11.

As shown in Fig 5.10 and 5.11, compared with our implementation of ICIM, the convergence results of CCIM is very robust even for complex interfaces. There is

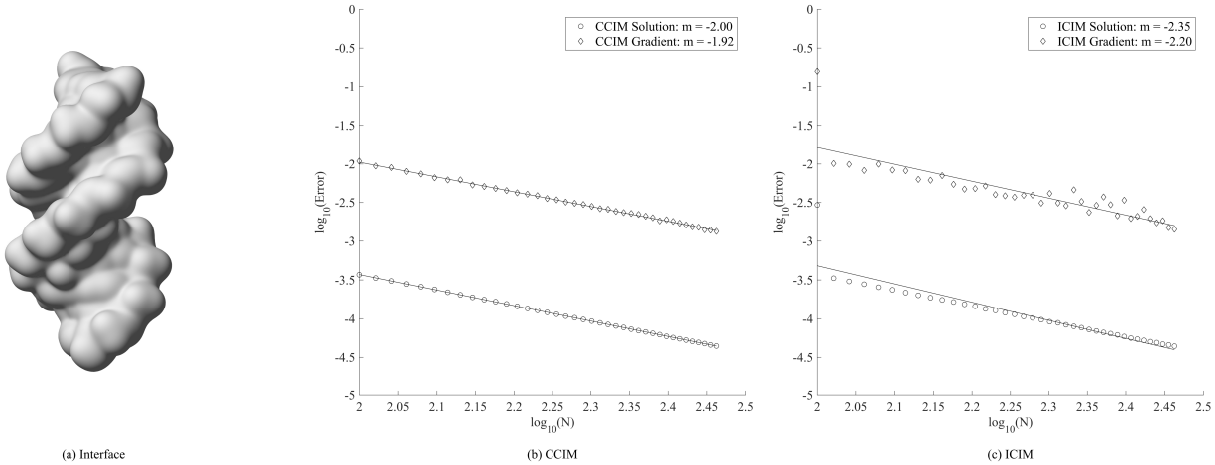


Figure 5.10. Convergence result for 1D63 interface with $c = 0.25$ and $\eta = 1/30$. (a) The smooth surface of 1D63. (b) log-log plot of error by CCIM. (b) log-log plot of error by ICIM. N ranges from 100 to 340 with the increment $\Delta N = 5$

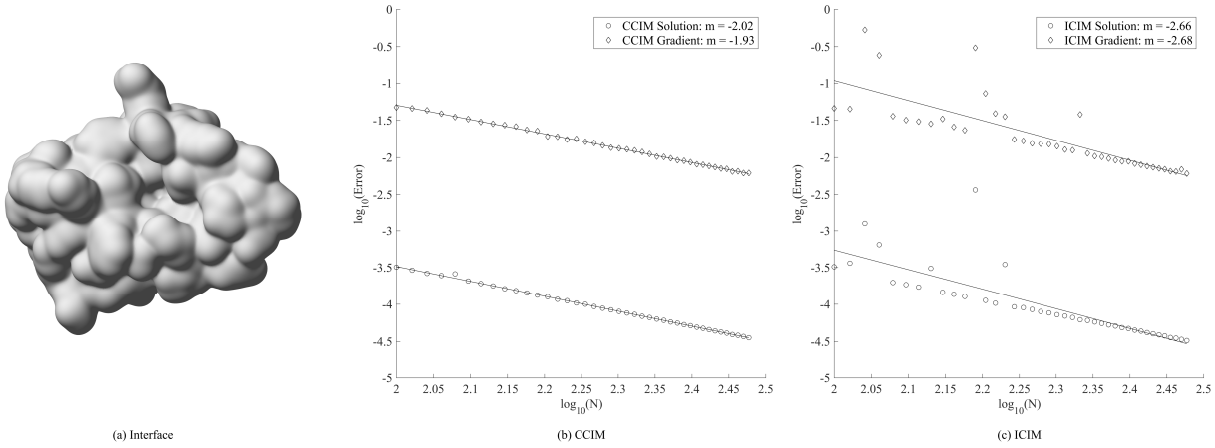


Figure 5.11. Convergence result for MDM2 interface with $c = 0.25$ and $\eta = 1/30$. (a) The smooth surface of MDM2. (b) log-log plot of error by CCIM. (b) log-log plot of error by ICIM. N ranges from 100 to 340 with the increment $\Delta N = 5$

little fluctuation in the convergence results. In our ICIM implementation, the order of convergence exceed second order because large error at coarse grid points skews the fitting line to have a more negative slope. The results demonstrate the advantage of the compactness in our CCIM formulation when dealing with complex surfaces.

5.3.3 Example 3

We also test our problem with the same exact solution (5.33) and coefficients (5.34), but with an $a(x, y, z)$ term, which is not handled in the ICIM formulation.

$$a(x,y,z) = \begin{cases} 2\sin(x) & \text{if } (x,y,z) \in \Omega^- \\ 80\cos(z) & \text{if } (x,y,z) \in \Omega^+ \end{cases} \tag{5.37}$$

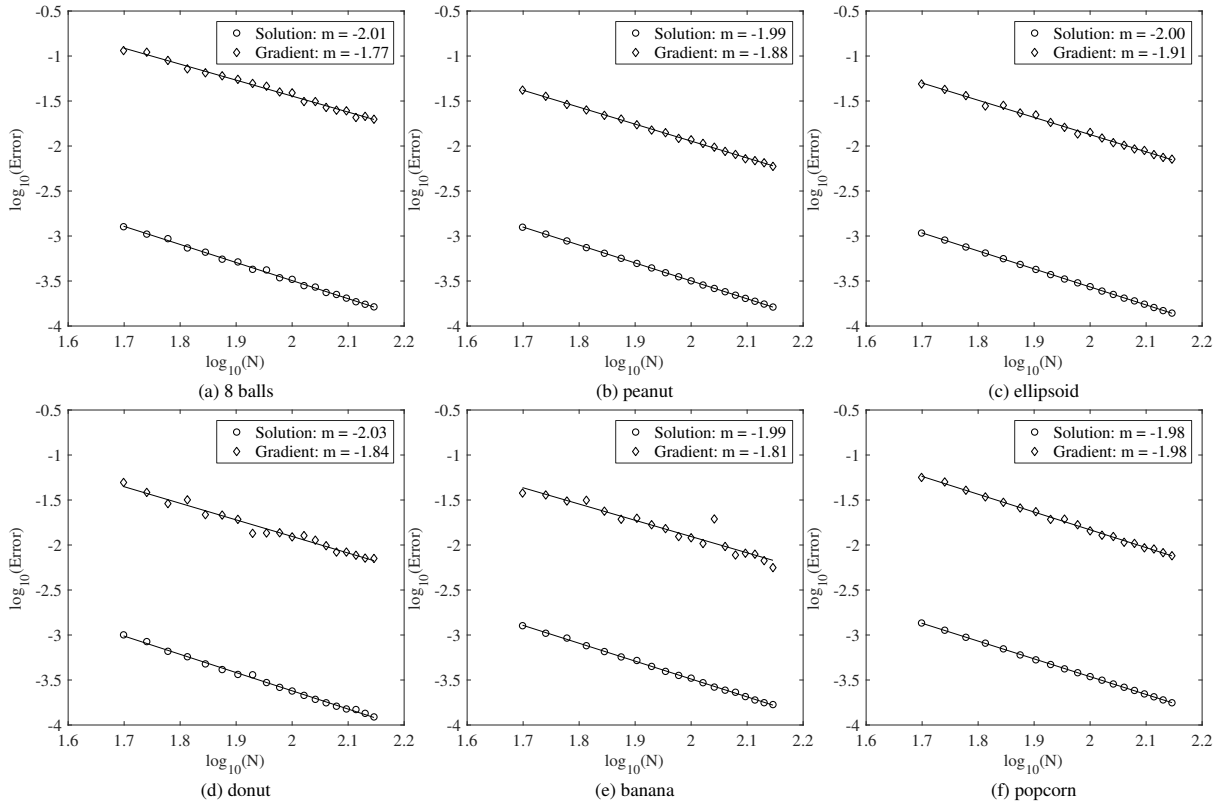
As shown in Fig.5.12, the convergence result is almost identical to those without the $a(x, y, z)$ term. Therefore the convergence result mainly depends on the alignment of the surface with the grid line.

5.3.4 Example 4

In this example, we look at the evolution of an interface driven by the jump in the normal derivative of the solution. Suppose the surface Γ is evolved with normal velocity $v_n = [\nabla u \cdot n]$. We use the forward Euler method for first-order accurate time discretization, Godunov scheme for the Hamiltonian, and the Fast Marching Method [28] to extend v_n to the whole computational domain.

Consider the radially symmetric functions

$$u_e(\mathbf{x}) = \begin{cases} \frac{1}{1+\|\mathbf{x}\|^2} & \mathbf{x} \in \Omega^- \\ -\frac{1}{1+\|\mathbf{x}\|^2} & \mathbf{x} \in \Omega^+ \end{cases} \tag{5.38}$$



(a) 8 balls (b) peanut (c) ellipsoid

(d) donut (e) banana (f) popcorn

Figure 5.12. The log-log plot error with $a(\mathbf{x})$ term versus N for the six surfaces. In each figure, N ranges from 50 to 140 with the increment $\Delta N = 5$.

and

$$a(\mathbf{x}) = \begin{cases} 2\sin(\|\mathbf{x}\|) & \mathbf{x} \in \Omega^- \\ 80\cos(\|\mathbf{x}\|) & \mathbf{x} \in \Omega^+ \end{cases}. \tag{5.39}$$

The coefficient ϵ is the same as (5.34). The source term and the jump conditions are calculated accordingly. If the surface is a sphere of radius r, by symmetry, the normal velocity is uniform over the sphere and is given by

$$v_n(r) = [\nabla u \cdot n] = \frac{4r}{(1+r^2)^2}. \tag{5.40}$$

Let the initial surface be a sphere of radius 0.5, then the motion of the surface is described by the ODE

$$\frac{dr}{dt} = v_n(r), \quad r(0) = 0.5 \tag{5.41}$$

which can be computed with high accuracy. The result is a sphere expanding at varying speeding.

In Fig. 5.13, we look at the maximum error and the Root Mean Squared Error (RMSE) of radius calculated at final time $t = 0.1$ for different N. The exact radius at final time should be 0.629428. The results are second order accurate. In Fig. 5.14, we plot the initial and final surface for $N = 20$. The shape is well-preserved. Without accurate gradient approximation, the surface might become distorted or oscillatory.

5.4 Conclusion

In this chapter, we introduced a Compact Coupling Interface Method to solve an elliptic interface problem in any dimension. Our method combines elements from Coupling Interface Method (CIM) and Mayo's approach to Poisson's equation on irregular regions. Standard central difference schemes are used at interior grid points. At interface grid point, coupling equations of the first-order derivatives and principal second-order

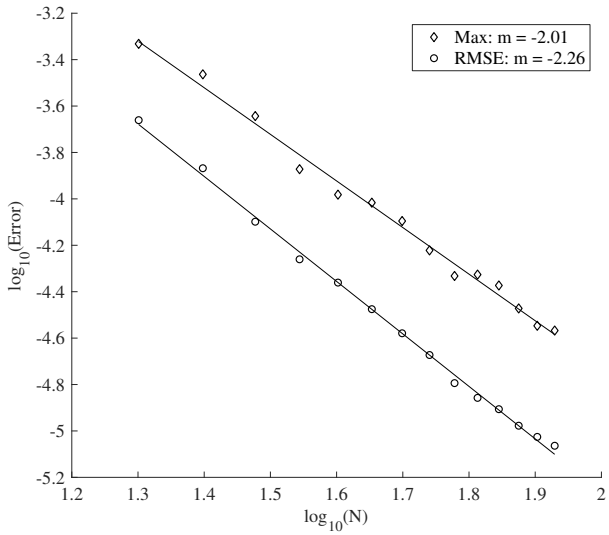


Figure 5.13. Error for radii at final time

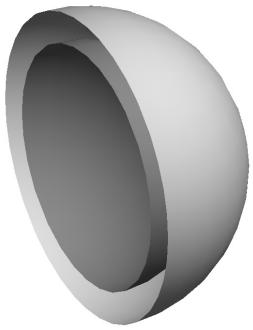


Figure 5.14. Initial (inner) and final (outer) surface

derivatives are derived in an dimension-splitting approach and by differentiating the jump condition. Our method obtains second order accurate solution at the grid points and second order accurate gradient at the interface. The accurate approximation for the gradient is important in applications where the dynamics of the surface is driven by the jump in the solution gradient at the interface. Our method has a more compact finite difference stencil compared with CIM and is applicable to complex interfaces. We test our method in three dimensions with complex interfaces, including two protein surfaces, and demonstrate that the solution and the gradient at the interface are uniformly second-order accurate, and the convergence results are very stable. We also test our method with a dynamic surface whose normal velocity is given by the jump in gradient at the interface and obtained second order accurate interface at the final time.

Chapter 5, in part, is a reprint of the material Zirui Zhang and Li-Tien Cheng. "A Compact Coupling Interface Method with Accurate Gradient Approximation for Elliptic Interface Problems" (2021).The dissertation author was the primary investigator and author of the material.

Bibliography

[1] Riccardo Baron and J. Andrew McCammon. "Molecular Recognition and Ligand Association". *Annual Review of Physical Chemistry* 64.1 (2013), pp. 151–175.

[2] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. "The Protein Data Bank". *Nucleic Acids Research* 28.1 (2000), pp. 235–242.

[3] Y. Boykov and V. Kolmogorov. "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.9 (2004), pp. 1124–1137.

[4] David G. Brown, Mark R. Sanderson, Elspeth Garman, and Stephen Neidle. "Crystal structure of a berenil-d(CGCAAATTTGCG) complex: An example of drug-DNA recognition based on sequence-dependent structural features". *Journal of Molecular Biology* 226.2 (1992), pp. 481–490.

[5] Hai-Feng; Luo Chen Ray. "Binding Induced Folding in p53-MDM2 Complex". *Journal of the American Chemical Society* 129.10 (2007), pp. 2930–2937.

[6] Patrick Chène. "Inhibiting the p53-MDM2 interaction: an important target for cancer therapy". *Nature Reviews. Cancer* 3.2 (2003), pp. 102–109.

[7] Li-Tien Cheng, Joachim Dzubiella, J. Andrew McCammon, and Bo Li. "Application of the level-set method to the implicit solvation of nonpolar molecules". *The Journal of Chemical Physics* 127.8 (2007), p. 084503.

[8] 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 (2010), pp. 8497–8510.

[9] I-Liang Chern and Yu-Chen Shu. "A coupling interface method for elliptic interface problems". *Journal of Computational Physics* 225.2 (2007), pp. 2138–2174.

[10] Todd J. Dolinsky, Jens E. Nielsen, J. Andrew McCammon, and Nathan A. Baker. "PDB2PQR: an automated pipeline for the setup of Poisson–Boltzmann electrostatics calculations". *Nucleic Acids Research* 32 (suppl_2 2004), W665–W667.

[11] J. Dzubiella, J. M. J. Swanson, and J. A. McCammon. "Coupling Hydrophobicity, Dispersion, and Electrostatics in Continuum Solvent Models". *Physical Review Letters* 96.8 (2006), p. 087802.

[12] J. Dzubiella, J. M. J. Swanson, and J. A. McCammon. "Coupling nonpolar and polar solvation free energies in implicit solvent models". *The Journal of Chemical Physics* 124.8 (2006), p. 084905.

[13] Robert D. Falgout and Ulrike Meier Yang. "hypre: A Library of High Performance Preconditioners". *Computational Science — ICCS 2002*. Ed. by Peter M. A. Sloot, Alfons G. Hoekstra, C. J. Kenneth Tan, and Jack J. Dongarra. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2002, pp. 632–641.

[14] Frederic Gibou and Ronald Fedkiw. "A Fast Hybrid k-Means Level Set Algorithm For Segmentation". *4th Annual Hawaii International Conference on Statistics and Mathematics* (2005), p. 11.

[15] Frederic Gibou, Ronald P. Fedkiw, Li-Tien Cheng, and Myungjoo Kang. "A Second-Order-Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains". *Journal of Computational Physics* 176.1 (2002), pp. 205–227.

[16] Zuojun Guo, Bo Li, Joachim Dzubiella, Li-Tien Cheng, J. Andrew McCammon, and Jianwei Che. "Heterogeneous Hydration of p53/MDM2 Complex". *Journal of Chemical Theory and Computation* 10.3 (2014), pp. 1302–1313.

[17] William W. Hager. "Condition Estimates". *SIAM Journal on Scientific and Statistical Computing* 5.2 (1984), pp. 311–316.

[18] W. K. Hastings. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". *Biometrika* 57.1 (1970), pp. 97–109.

[19] Jing Huang, Sarah Rauscher, Grzegorz Nawrocki, Ting Ran, Michael Feig, Bert L. de Groot, Helmut Grubmüller, and Alexander D. MacKerell. "CHARMM36m: an improved force field for folded and intrinsically disordered proteins". *Nature Methods* 14.1 (2017), pp. 71–73.

[20] Michele Miranda Jr, Diego Pallara, Fabio Paronetto, and Marc Preunkert. "Short-time heat flow and functions of bounded variation in RN" (), p. 22.

[21] Catherine Kublik, Nicolay M. Tanushev, and Richard Tsai. "An implicit interface boundary integral method for Poisson's equation on arbitrary domains". *Journal of Computational Physics* 247 (2013), pp. 279–311.

[22] P. H. Kussie, S. Gorina, V. Marechal, B. Elenbaas, J. Moreau, A. J. Levine, and N. P. Pavletich. "Structure of the MDM2 oncoprotein bound to the p53 tumor suppressor transactivation domain". *Science (New York, N.Y.)* 274.5289 (1996), pp. 948–953.

[23] Randall J. LeVeque and Zhilin Li. "The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources". *SIAM Journal on Numerical Analysis* 31.4 (1994), pp. 1019–1044.

[24] Yaakov Levy and José N. Onuchic. "Water Mediation in Protein Folding and Molecular Recognition". *Annual Review of Biophysics and Biomolecular Structure* 35.1 (2006), pp. 389–415.

[25] J. Lie, M. Lysaker, and Xue-Cheng Tai. "A binary level set model and some applications to Mumford-Shah image segmentation". *IEEE Transactions on Image Processing* 15.5 (2006), pp. 1171–1181.

[26] Selim Esedoglu Lu and Felix Otto. "Threshold Dynamics for Networks with Arbitrary Surface Tensions". *Communications on Pure and Applied Mathematics* 68.5 (2015), pp. 808–864.

[27] A. Mayo. "The Fast Solution of Poisson's and the Biharmonic Equations on Irregular Regions". *SIAM Journal on Numerical Analysis* 21.2 (1984), pp. 285–299.

[28] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences. New York: Springer-Verlag, 2003.

[29] Stanley Osher and James A Sethian. "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations". *Journal of Computational Physics* 79.1 (1988), pp. 12–49.

[30] Clarisse G. Ricci and J. Andrew McCammon. "Heterogeneous Solvation in Distinctive Protein–Protein Interfaces Revealed by Molecular Dynamics Simulations". *The Journal of Physical Chemistry B* 122.49 (2018), pp. 11695–11701.

[31] Toshiya Senda, Kazuyuki Sugiyama, Hiroki Narita, Takeshi Yamamoto, Kazuhide Kimbara, Masao Fukuda, Mitsuo Sato, Keiji Yano, and Yukio Mitsui. "Three-dimensional Structures of Free Form and Two Substrate Complexes of an Extradiol Ring-cleavage Type Dioxygenase, the BphC Enzyme fromPseudomonassp. Strain KKS102". *Journal of Molecular Biology* 255.5 (1996), pp. 735–752.

[32] Yu-Chen Shu, I-Liang Chern, and Chien C. Chang. "Accurate gradient approximation for complex interface problems in 3D by an improved coupling interface method". *Journal of Computational Physics* 275 (2014), pp. 642–661.

[33] Bing Song and Tony Chan. "A Fast Algorithm for Level Set Based Optimization". *CAM-UCLA* 68 (2002), pp. 02–68.

[34] Dong Wang, Haohan Li, Xiaoyu Wei, and Xiao-Ping Wang. "An efficient iterative thresholding method for image segmentation". *Journal of Computational Physics* 350 (2017), pp. 657–667.

[35] Zhongming Wang, Jianwei Che, Li-Tien Cheng, Joachim Dzubiella, Bo Li, and J. Andrew McCammon. "Level-Set Variational Implicit-Solvent Modeling of Biomolecules with the Coulomb-Field Approximation". *Journal of Chemical Theory and Computation* 8.2 (2012), pp. 386–397.

[36] Zirui Zhang and Li-Tien Cheng. "A Compact Coupling Interface Method with Accurate Gradient Approximation for Elliptic Interface Problems" (2021).

[37] Zirui Zhang and Li-Tien Cheng. "Binary Level Set Method for Variational Implicit Solvation Model" (2021).

[38] Zirui Zhang, Clarisse G. Ricci, Chao Fan, Li-Tien Cheng, Bo Li, and J. Andrew McCammon. "Coupling Monte Carlo, Variational Implicit Solvation, and Binary Level-Set for Simulations of Biomolecular Binding". *Journal of Chemical Theory and Computation* (2021).

[39] Haizhen Zhong and Heather A. Carlson. "Computational studies and peptidomimetic design for the human p53-MDM2 complex". *Proteins: Structure, Function, and Bioinformatics* 58.1 (2004), pp. 222–234.

[40] Guangfeng Zhou, George A. Pantelopulos, Sudipto Mukherjee, and Vincent A. Voelz. "Bridging Microscopic and Macroscopic Mechanisms of p53-MDM2 Binding with Kinetic Network Models". *Biophysical Journal* 113.4 (2017), pp. 785–793.

[41] Shenggao Zhou, Li-Tien Cheng, Joachim Dzubiella, Bo Li, and J. Andrew McCammon. "Variational Implicit Solvation with Poisson–Boltzmann Theory". *Journal of Chemical Theory and Computation* 10.4 (2014), pp. 1454–1467.