**Noname manuscript No.**
(will be inserted by the editor)

# A Compact Coupling Interface Method with Second-Order Gradient Approximation for Elliptic Interface Problems

**Ray Zirui Zhang · Li-Tien Cheng**

the date of receipt and acceptance should be inserted later

**Abstract** We propose the Compact Coupling Interface Method (CCIM), a finite difference method capable of obtaining second-order accurate approximations of not only solution values but their gradients, for elliptic complex interface problems with interfacial jump conditions. Such elliptic interface boundary value problems with interfacial jump conditions are a critical part of numerous applications in fields such as heat conduction, fluid flow, materials science, and protein docking, to name a few. A typical example involves the construction of biomolecular shapes, where such elliptic interface problems are in the form of linearized Poisson-Boltzmann equations, involving discontinuous dielectric constants across the interface, that govern electrostatic contributions. Additionally, when interface dynamics are involved, the normal velocity of the interface might be comprised of the normal derivatives of solution, which can be approximated to second-order by our method, resulting in accurate interface dynamics. Our method, which can be formulated in arbitrary spatial dimensions, combines elements of the highly-regarded Coupling Interface Method, for such elliptic interface problems, and Smereka's second-order accurate discrete delta function. The result is a variation and hybrid with a more compact stencil than that found in the Coupling Interface Method, and with advantages, borne out in numerical experiments involving both geometric model problems and complex biomolecular surfaces, in more robust error profiles.

**Keywords** Elliptic interface problems, Compact Coupling Interface Method, complex interfaces, Second-order method for gradient

Ray Zirui Zhang
E-mail: zzirui@ucsd.edu
Li-Tien Cheng
E-mail: lcheng@math.ucsd.edu

Department of Mathematics, University of California, San Diego
9500 Gilman Drive, La Jolla, CA, 92093-0112, USA

## 1 Introduction

### 1.1 Applications

Elliptic interface problems with interfacial jump conditions can be found at the heart of a variety of physical and biological problems involving interfaces. These interfaces may be material interfaces or phase boundaries, static or dynamic, and in subjects relating to heat conduction [1,2], fluid dynamics [3], materials science [4,5], electromagnetics [6,7], or electrostatics [8,9,10], tumor growth [11,12]. The interfacial jump conditions are due, frequently, to material properties and sources that are discontinuous, or have discontinuous derivatives, across the interface. This leads to solutions that also have discontinuities in values or derivatives at the interface. These discontinuities, especially when they are large, are what presents the main difficulties in this problem.

Our motivating application that fits into this framework involves biomolecular shapes. Consider a set of solute atoms making up a biomolecule, or several biomolecules, such as proteins involved in a docking process. One interest in this situation is how the atoms affect the solvent that it resides in, usually a solution resembling salt-water. The implicit solvation approach introduces an interface to separate a continuously modeled solvent from the solute atoms and vacuum [13,14] It additionally pairs with this a free energy involving contributions such as nonpolar van der Waals forces, surface effects, and electrostatics, with the minimizer serving as the desired interface [15,9,16,17]. The electrostatics portion here provides the elliptic interface problem with interfacial jump conditions we are interested in, arising from linearization approximations of the governing Poisson-Boltzmann equation [9,18,10,19,20].

### 1.2 Setup

Let $\Omega \subset \mathbb{R}^d$ be a rectangular box and consider an orientable $\mathcal{C}^1$ hypersurface $\Gamma$ that separates it into an inside region $\Omega^-$ and an outside region $\Omega^+$. Also let $\mathbf{n}$ denote outward unit normal vectors on the interface (see Fig. 1). Then, for given functions $\epsilon$, $f$, $a$: $\Omega \to \mathbb{R}$, possibly discontinuous across the interface, and given functions $\tau$, $\sigma$: $\Gamma \to \mathbb{R}$, our specific elliptic interface problem of interest, with interfacial jump conditions, takes the form:

$$\begin{cases} -\nabla \cdot (\epsilon \nabla u) + au = f & \text{in } \Omega \setminus \Gamma, \\ [u] = \tau, \quad [\epsilon \nabla u \cdot \mathbf{n}] = \sigma & \text{on } \Gamma, \\ u = g & \text{on } \partial\Omega. \end{cases} \tag{1}$$

Here, for any $v : \Omega \to \mathbb{R}$ a function and $x \in \Gamma$, we employ the commonly used notation of $[v]$ to denote the jump of $v$ across the interface at $x$:

$$[v] = v^+ - v^-. \tag{2}$$

The superscript $+$ or $-$ denotes the limiting value of a function from $\Omega^+$ or $\Omega^-$, respectively. Additionally, we will refer to $\epsilon \nabla u \cdot \mathbf{n}$ as the flux; $\tau$, $\sigma$: $\Gamma \to \mathbb{R}$

as the value of the jump conditions; $\Omega$ as the computational domain; and $g$ as the value of the Dirichlet boundary conditions on $\partial\Omega$. Note, we allow the dimension $d$ to be general for the formulation of our method, but restrict to the case $d = 3$ for computations, which we find to have sufficient complexity for many real-world problems.

In our motivating application, this form is achieved in a linearized Poisson-Boltzmann equation for electric potential $u$, charge density $f$, and dielectric coefficient $\epsilon$, which can take on values of around 1 or 2 in the solute region and 80 in the solvent region [8,10,9].

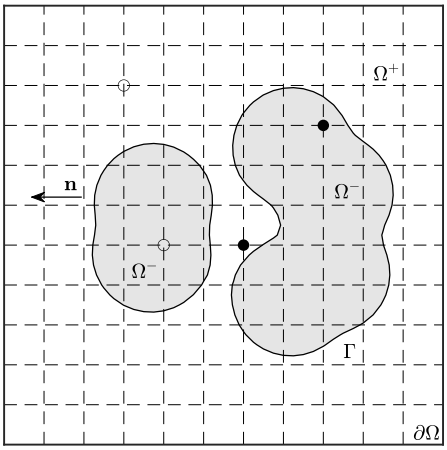



Fig. 1: Schematic for the elliptic interface problem. $\Gamma$ is an interface that separates a cubical domain $\Omega$ with boundary $\partial\Omega$ into $\Omega^+$ and $\Omega^-$. The normal to the interface is denoted as $\mathbf{n}$. The dashed lines are the grid lines of the uniform mesh. The circles are interiors points, where standard central difference can be used. The disks are examples of on-front points, where special stencils are required.

### 1.3 Interface Dynamics and Gradients

In applications that concern both such elliptic interface problems and moving interfaces, the velocity of the interface may depend on the gradient of the solution. This makes it imperative to accurately calculate not only the solution's values, but its gradient as well [11,21,9]. In fact, it is preferrable for errors to be measured under the infinity norm, to obtain accurate pointwise velocities that can be used in both front tracking methods and level-set methods for interface dynamics.

In our motivating biomolecular application, gradient descent on the free-energy functional can be used to capture the solute-solvent interface of interest

[14]. This introduces a time variable and interface dynamics into the originally static problem. Additionally, gradient descent in this case translates to a normal velocity that depends on the effective dielectric boundary force, which in turn depends on the jump of the normal derivative of the electrostatic potential at the interface [21,9].

In another and perhaps better known example, the Stefan problem [2,1] is used to model an interface separating ice from water, where the interface moves when ice melts or water freezes. In simple terms, for a given configuration of the ice and water, an elliptic interface problem, with $\epsilon$ as thermal conductivity, can be solved for the temperature $u$. The interface between the ice and water then evolves due to this temperature, with its normal velocity depending on the jump in the normal derivative of the temperature.

From these considerations, we can finally formulate our goal: to accurately and efficiently solving elliptic interface problems with jump conditions for both solution and gradient.

### 1.4 Finite Difference Methods

There are many existing approaches that solve elliptic interface problems with jump conditions, however, they can roughly be categorized by the framework they employ. An example is the Coupling Interface Method (CIM) [22], which is a finite difference method, employing a fixed grid that provides ease in resolution and in the construction of accurate schemes such as central differencing. In fact, the main classes of methods found include finite difference, finite element, boundary integral, finite volume, and deep learning methods, each with their own inherent advantages and disadvantages and suitability for chosen applications.

An example of such suitability involves elliptic interface problems with their jump boundary conditions set on moving boundaries. In this setting, one has the added concern of how well a chosen framework interacts with a chosen numerical representation of the boundary interface. If the level-set method [23,24] – a finite difference method that has proven itself in its handling of complicated dynamics, especially topological changes – were used to represent the interface, then a finite difference method for the elliptic interface problem would be the natural fit; in fact, both methods could share the same underlying grid.

This example, though, is exactly the reason that we decide to concentrate on finite difference methods in this paper. In our biomolecule application of interest, under implicit solvation, where the solvent is represented as a continuous medium, the goal is to capture the interface that separates a biomolecule from the surrounding solvent. The Variational Implicit Solvation Model (VISM) constructs the free energy of the system and seeks a minimizing interface [14,13,25]. One approach to obtaining the minimizer is gradient descent on the energy, resulting in the flow of an initial interface to, in steady state, a minimizer. This interface flow was handled, in [25,15,9], exactly by

the level-set method. Thus, to build on the work and results there, in adding electrostatic effects, which can be described by elliptic interface problems with jump boundary conditions on the moving interfaces, we choose the finite difference method as our framework.

Again, this is not to say other frameworks are not, in other circumstances, advantageous. We simply believe a finite difference method would best fit our problem of interest in our application of interest and likely others as well. Under other frameworks, we note especially the boundary integral methods of [26,10,27,18]; finite volume methods of [3,28]; finite element methods of [29, 30,31,32], particularly those involving unfitted meshes that are suitable for moving interfaces [33,34,35,36,37,38,39,40,41,42,43]; and, recently, the deep learning methods of [44,45]. This list, furthermore, is by no means an exhaustive account of methods under other frameworks. The topic of elliptic interface problems with jump boundary conditions is a very active area of research with a large body of literature on the algorithms, analysis, and applications.

As discussed in [22], within finite difference methods, there are essentially three types of approaches: regularization, dimension unsplitting, and dimension splitting approaches. A regularizatation approach applies smoothing techniques to discontinuous coefficients, or regularization techniques to singular sources [46,47]. An major example of this that is related to our problem of interest is the Immersed Boundary Method (IBM) [48,49]. In dimension unsplitting approaches, finite difference methods are derived from local Taylor expansions in multi-dimensions. One popular method in this category is the Immersed Interface Method [50] and its various extensions, including the Maximum Principle Preserving Immersed Interface Method (MIIM)[51], the Fast Immersed Interface Method (FIIM) [52], and the Augmented Immersed Interface Method (AIIM) [53]. For dimension splitting approaches, the finite difference methods are derived from Taylor expansions in each dimension. This category includes the Ghost Fluid Method [54,55,56], the Explicit-jump Immersed Interface Method (EJIIM) [57], the Decomposed Immersed Interface Method (DIIM) [58], the Matched Interface and Boundary Method (MIB) [59,60,61], and CIM and its variation, Improved Coupling Interface Method (ICIM) [22,8,62]. For a more detailed discussion on these different types, we refer readers to [22].

In fact, towards our goal, we will be combining two pieces of work set in the finite difference framework, namely CIM [22] and Smereka's work on discrete delta functions [63]. The former is able to produce second-order accurate solutions and first order accurate derivatives of general elliptic interface problems with jump boundary conditions, while the latter can perform second-order accurate operations with a discrete delta function derived within a specific class of elliptic interface problems with jump boundary conditions. To see the connection between delta functions and elliptic interface problems, consider the Green's function for Laplace's equation: $\Delta u = \delta_\Gamma$ with $u = 0$ on $\partial\Omega$, where $\delta_\Gamma$ is the delta function supported on the interface $\Gamma$. This is equivalent to the elliptic interface problem: $\Delta u = 0$ with $[u] = 0$ and $[\nabla u \cdot \mathbf{n}] = 1$. This con-

nection is used to construct a second-order accurate discrete approximation of the delta function in [63].

### 1.5 The Coupling Interface Method

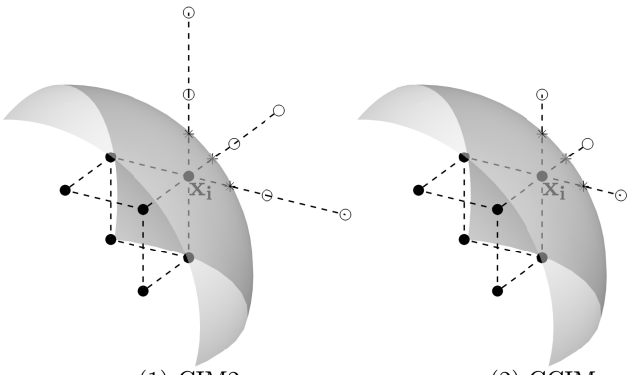
Among these finite difference methods, CIM is one of the top ones in terms of accuracy, in both solutions values and gradients; ease of use; and detail of study (see [22,8,62]). In CIM, the standard central differencing stencil is used when the grid point is away from the interface. When the grid point is next to the interface that forbids the use of the standard central differencing stencil, CIM uses polynomial approximations on either side of the interface, in each dimension, and connects them with jump conditions at the interface. This leads to a coupled linear system of equations to be solved for the principal second-order derivatives in terms of values at gridpoints.

One version of this approach, called CIM1, chooses linear polynomials and lower-order approximations of mixed derivatives for a lower-order but widely applicable approach; another, called CIM2, chooses quadratic polynomials and higher-order approximations of mixed derivatives for a higher-order approach that, however, requires certain larger stencils. CIM is a hybrid of these that uses CIM2 approximations at points where the stencils allow, and CIM1 approximations at all other gridpoints, called exceptional points. Note, these exceptional points do commonly exist but, as noted in [22], not in great numbers, allowing CIM to be second-order accurate in solution values under the infinity norm. For gradients, however, this approach is only first-order accurate, especially at exceptional points.

The Improved Coupling Interface Method (ICIM) [8] fixes this issue and achieves uniformly second-order accurate gradient by incorporating two recipes that handle exceptional points. One attempts to "shift": at gridpoints where it is difficult to achieve a valid first-order approximation of the principal or mixed second-order derivative, the finite difference approximations at adjacent gridpoints, on the same side, are instead shifted over. The other attempts to "flip": at some gridpoints, the signature of the domain (inside or outside) can be flipped, so that the usual second-order CIM2 discretization may apply, allowing for second-order accurate solutions and gradients for the flipped interface. Extrapolation from neighboring nonflipped gridpoints originally on the same side can then be used to recover the solutions and gradients of the original and desired interface. Note, the decisions on when to use shifts and when to use flips are listed in [8].

### 1.6 Our Proposed Method and Contributions

We propose what can be considered a hybrid that we call the Compact Coupling Interface Method (CCIM). Our method combines elements of CIM [22] and Smereka's work on second-order accurate discrete delta functions by setting up an elliptic interface problem with interfacial jump conditions [63]. The




(1) CIM2 (2) CCIM

Fig. 2: Examples of a (1) CIM2 stencil and a (2) CCIM stencil at $\mathbf{x_i}$. The circles and disks are grid points on different sides of the interface. The asterisks are the intersections of the surface and the grid lines. In this case, CIM2 and ICIM have the same stencil, which requires 2 points on the same side of the interface in each dimension [22]. Points that does not satisfy this requirement are handled in ICIM [8]. The CCIM stencil requires fewer points so it's more compact. Both CIM2 and CCIM need some extra grid points compared to the standard central difference stencil.

use of Smereka's setup, itself based on Mayo's work in [64], allows us to remove the quadratic polynomial approximations of CIM2 and its need for two points on either side of the interface in a direction that crosses the interface, thus compacting the stencil and allowing more applicability in generating accurate principal second-order derivatives. Additional schemes are introduced to accurately handle mixed second-order derivatives by using more compact stencils on the same side or stencils from the opposite side, with the help of jump conditions, allowing for the removal of exceptional points. The result is that our constructed CCIM has a more compact stencil (shown in Fig 2) and it can approximate values and gradients of the solutions of elliptic interface problems with jump conditions with second-order accuracy in infinity norm for a variety interfaces, with observed advantages in robust convergence behavior in complex situations.

Note, while ICIM improves on CIM2 through shifting or flipping, the coupling equations remain largely the same, which only includes the principal second-order derivatives, and similar finite difference stencils [8]. Our CCIM expands the coupling equations to include first-order derivatives as well, and utilizes more compact finite difference stencils, for the removal of additional exceptional points.

### 1.7 Outline

This paper is organized as follows. Section 2 outlines the derivation and algorithm of CCIM. In Section 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 of the gradient at the interface. Section 4 is the conclusion.

## 2 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 side of the region $\Omega$. 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 the $k$-th coordinate $x_k = -1 + i_k h$. 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 $\Delta u$ for the Laplacian of $u$ and $\nabla^2 u$ for the Hessian matrix of $u$. We use $\overline{\mathbf{x_i}\mathbf{x_{i+e}}_k}$ to denote the grid segment between $\mathbf{x_i}$ and $\mathbf{x}_{\mathbf{i}+\mathbf{e}_k}$, and assume that the interface intersects 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 (1) at $\mathbf{x_i}$ as

$$-\sum_{k=1}^{d} \frac{\partial \epsilon}{\partial x_k}\frac{\partial u}{\partial x_k} - \epsilon \sum_{j=1}^{d} \frac{\partial^2 u}{\partial x_k^2} + au = f \tag{3}$$

We classify the grid points into two categories: if $\mathbf{x}_{\mathbf{i}-\mathbf{e}_k}$, $\mathbf{x_i}$ and $\mathbf{x}_{\mathbf{i}+\mathbf{e}_k}$ are in the same region in each coordinate direction, then we call $\mathbf{x_i}$ an **interior point**, otherwise $\mathbf{x_i}$ is called an **on-front 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 on-front points. The overall accuracy will still be second-order since the on-front points belong to a lower dimensional set [50]. In this section, we derive a first-order approximation for the term $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$ in terms of u-values on neighboring grid points. We denote the set of neighboring grid points of $\mathbf{x_i}$ as $B_r = \{\mathbf{x_j} \mid \|\mathbf{j} - \mathbf{i}\|_\infty \le r\}$ and call $r$ the radius of our finite difference stencil. As an example in 2, the CIM2/ICIM stencil has $r = 2$ and the CCIM stencil has $r = 1$.

### 2.1 Dimension-by-dimension discretization

This section follows the derivation found in Smereka's work [63]. Along the coordinate direction $\mathbf{e}_k$, if the interface does not intersect the grid segment

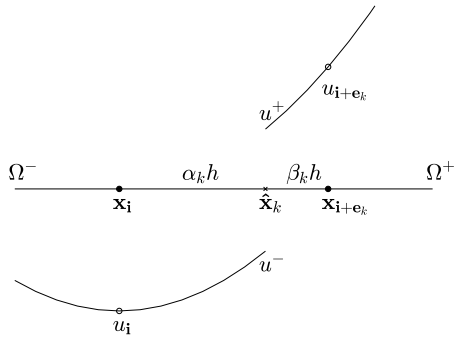



Fig. 3: The interface intersects the grid segment $\overline{\mathbf{x_i x_{i+e_k}}}$ at $\hat{\mathbf{x}}_k$. $u^-$ and $u^+$ are the limits of u at $\hat{\mathbf{x}}_k$ from $\Omega^-$ and $\Omega^+$. $u_{\mathbf{i}}$ and $u_{\mathbf{i}+\mathbf{e}_k}$ are approximated by Taylor's expansion at the interface.

$\overline{\mathbf{x_i x_{i+e_k}}}$, then by Taylor's theorem,

$$u_{\mathbf{i}+\mathbf{e}_k} - u_{\mathbf{i}} = h\frac{\partial u}{\partial x_k} + \frac{h^2}{2}\frac{\partial^2 u}{\partial x_k^2} + \mathcal{O}\big(h^3\big). \tag{4}$$

Suppose the interface intersects the grid segment $\overline{\mathbf{x_i x_{i+e_k}}}$ at the interface point $\hat{\mathbf{x}}_k$. Let $\alpha_k = \|\hat{\mathbf{x}}_k - \mathbf{x_i}\|/h$ and $\beta_k = 1 - \alpha_k$. Suppose $\mathbf{x_i}$ is located in $\Omega^-$. Denote the limit of $u(\mathbf{x})$ as $\mathbf{x}$ approaches $\hat{\mathbf{x}}_k$ from $\Omega^-$ by $u^-$ , and the limit from the other side by $u^+$ (see Fig 3). By Taylor's theorem,

$$\begin{aligned} u_{\mathbf{i}} &= u^- - \alpha_k h\frac{\partial u^-}{\partial x_k} + \frac{(\alpha_k h)^2}{2}\frac{\partial^2 u^-}{\partial x_k^2} + \mathcal{O}\big(h^3\big), \\ u_{\mathbf{i}+\mathbf{e}_k} &= u^+ + \beta_k h\frac{\partial u^+}{\partial x_k} + \frac{(\beta_k h)^2}{2}\frac{\partial^2 u^+}{\partial x_k^2} + \mathcal{O}\big(h^3\big). \end{aligned} \tag{5}$$

Subtract the above two equations and write the right-hand side in terms of jumps and quantities from $\Omega^-$:

$$\begin{aligned} u_{\mathbf{i}+\mathbf{e}_k} - u_{\mathbf{i}} = [u] + \beta_k h\left[\frac{\partial u}{\partial x_k}\right] + h\frac{\partial u^-}{\partial x_k} + \\ \frac{h^2}{2}\beta_k^2\left[\frac{\partial^2 u}{\partial x_k^2}\right] + \frac{h^2}{2}(\beta_k^2 - \alpha_k^2)\frac{\partial^2 u^-}{\partial x_k^2} + \mathcal{O}\big(h^3\big). \end{aligned} \tag{6}$$

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

$$\frac{\partial u^-}{\partial x_i} = \frac{\partial u}{\partial x_i} + \alpha_k h\frac{\partial^2 u}{\partial x_i \partial x_k} + \mathcal{O}\big(h^2\big), \tag{7}$$

$$\frac{\partial^2 u^-}{\partial x_i \partial x_k} = \frac{\partial^2 u}{\partial x_i \partial x_k} + \mathcal{O}(h). \tag{8}$$

with $1 \le i \le k \le d$. Together with the given jump condition, $[u] = \tau$, (6) can be written as

$$\begin{aligned} u_{\mathbf{i}+\mathbf{e}_k} - u_{\mathbf{i}} = \tau + \beta_k h \left[\frac{\partial u}{\partial x_k}\right] + h\left(\frac{\partial u}{\partial x_k} + \alpha_k h \frac{\partial^2 u}{\partial x_k^2}\right) \\ + \frac{h^2}{2}\beta_k^2 \left[\frac{\partial^2 u}{\partial x_k^2}\right] + \frac{h^2}{2}(\beta_k^2 - \alpha_k^2)\frac{\partial^2 u}{\partial x_k^2} + \mathcal{O}(h^3). \end{aligned} \tag{9}$$

We emphasize that all the first and second-order derivatives in (9) are evaluated at the on-front point $\mathbf{x_i}$, and all the jump term $[\partial u/\partial x_k]$ and $[\partial^2 u/\partial x_k^2]$ are defined and evaluated at $\hat{\mathbf{x}}_k$, which are intermediate quantities that will be eliminated in the coupling equation.

## 2.2 Coupling Equation

This section then sets up coupling equations following the work of [22]. In (9), 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 $1 \le k \le d$ 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, (9) or (4), by considering the two grid segments $\overline{\mathbf{x_i}\mathbf{x}_{\mathbf{i}+s\mathbf{e}_k}}$ for $s = \pm 1$. In $d$ dimensions we have $2d$ equations and $2d$ unknowns: the first-order derivatives $\partial u/\partial x_k$ and the principal 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:

$$M \begin{pmatrix} \frac{\partial u}{\partial x_k} \\ \frac{\partial^2 u}{\partial x_k^2} \end{pmatrix}_{1\le k\le d} = \frac{1}{h^2}\left(L_{k,s}(u_{\mathbf{j}\in B_r})\right)_{1\le k\le d, s=\pm 1} + \mathcal{O}(h) \tag{10}$$

where $L_{k,s}(u_{\mathbf{j}\in B_r})$ is some affine function of $u$-values in the neighborhood $B_r$. We call (10) the coupling equation and $M$ the coupling matrix. By inverting $M$, we can approximate $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$ in terms of $u$-values and obtain the finite difference approximation of the PDE (3) at the on-front point $\mathbf{x_i}$.

In the next few sections, we describe the ingredients to construct the coupling equation (10). We will use "y = L(x)" to denote "write the quantities y in terms of affine function of quantities x", where L represent a generic affine function and it depends on the geometric quantities of the interface. In Section 2.3, we derive expressions to approximate the jump of the first-order derivatives $[\partial u/\partial x_k]$ in (9). In Section 2.4, we approximate the jump of the principal second-order derivatives $[\partial^2 u/\partial x_k^2]$ in (9). In Section 2.5, we discuss how to approximate the mixed derivatives, which is used to approximate $[\partial u/\partial x_k]$ and $[\partial^2 u/\partial x_k^2]$. In Section 2.6, we combine all the ingredients and describe our algorithm to obtain the coupling equation.

### 2.3 Approximation of $[\partial u/\partial x_k]$

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

$$[\nabla u] = [\nabla u \cdot \mathbf{n}]\mathbf{n} + \sum_{j=1}^{d-1} [\nabla u \cdot \mathbf{s}_j]\mathbf{s}_j = [\nabla u \cdot \mathbf{n}]\mathbf{n} + \sum_{j=1}^{d-1} (\nabla \tau \cdot \mathbf{s}_j)\mathbf{s}_j, \tag{11}$$

because $[\nabla u \cdot \mathbf{s}_j] = \nabla \tau \cdot \mathbf{s}_j$ for $1 \le j \le d-1$.

We use the following trick frequently to decouple the jump in subsequent derivations:

$$[\epsilon v] = \epsilon^+ [v] + [\epsilon] v^-. \tag{12}$$

The jump condition $[\epsilon \nabla u \cdot \mathbf{n}] = \sigma$ can be rewritten as

$$[\nabla u \cdot \mathbf{n}] = \frac{1}{\epsilon^+}(\sigma - [\epsilon]\nabla u^- \cdot \mathbf{n}). \tag{13}$$

Substitute (13) into (11), in direction $\mathbf{e}_k$, we have

$$\left[\frac{\partial u}{\partial x_k}\right] = \frac{1}{\epsilon^+}(\sigma - [\epsilon]\nabla u^- \cdot \mathbf{n})(\mathbf{n} \cdot \mathbf{e}_k) + \sum_{j=1}^{d-1} (\nabla \tau \cdot \mathbf{s}_j)(\mathbf{s}_j \cdot \mathbf{e}_k). \tag{14}$$

Approximate $\nabla u^-$ by Taylor's theorem (7), we get

$$\begin{aligned} \left[\frac{\partial u}{\partial x_k}\right] = \frac{1}{\epsilon^+} \left( \sigma - [\epsilon] \sum_{j=1}^{d} \left( \frac{\partial u}{\partial x_j} + \alpha_k h \frac{\partial^2 u}{\partial x_j \partial x_k} \right) (\mathbf{n} \cdot \mathbf{e}_j) \right) (\mathbf{n} \cdot \mathbf{e}_k) \\ + \sum_{j=1}^{d-1} (\nabla \tau \cdot \mathbf{s}_j)(\mathbf{s}_j \cdot \mathbf{e}_k). \end{aligned} \tag{15}$$

Notice that the jump in the first-order derivative $[\partial u/\partial x_k]$ can be written as linear combinations of the first-order derivatives $\partial u/\partial x_l$, $1 \le l \le d$ and the second-order derivatives $\partial^2 u/\partial x_l \partial x_m$, $1 \le l \le m$:

$$\left[\frac{\partial u}{\partial x_k}\right] = L(\nabla u, \nabla^2 u). \tag{16}$$

where $L$ represents the affine function in (15). Our goal is to approximate the mixed derivatives $\partial^2 u/\partial x_k \partial x_l$, $k \ne l$ in terms of the neighboring $u$-values $u_{\mathbf{j}}$, $\mathbf{j} \in B_r$, the first-order derivatives $\partial u/\partial x_k$, and the principal second-order derivatives $\partial^2 u/\partial x_k^2$, $1 \le k \le d$ (see Section 2.4 and 2.5), which are the terms used in the coupling equation (10).

### 2.4 Approximation of $[\partial^2 u/\partial x_k^2]$

To remove the jump of the principal second-order derivatives $[\partial^2 u/\partial x_k^2]$, $k = 1, 2, \ldots, d$ in (9), we need to solve a system of linear equations, whose unknowns are all the jump of the principal and the mixed second-order derivatives. This idea is used in Smereka's work [63] to arrive at the discrete approximation of the delta function. The detailed derivation is given in the Appendix A. However, we would like to note here our definition of the gradient of a vector field as there is no standard one. For a vector field $\mathbf{v}$, we define the matrix $\nabla\mathbf{v}$ to have entries in row $i$ and column $j$:

$$(\nabla\mathbf{v})_{ij} = \frac{\partial v_j}{\partial x_i}.$$

This particular form is used in our equations on the jumps of second derivatives.

**Tangential derivative of jump of tangential derivative:** The first set of equations are obtained by differentiating the interface boundary condition in the tangential directions. For $m = 1, \cdots, d-1$ and $n = m, \cdots, d-1$, we get $d(d-1)/2$ equations

$$\nabla[\nabla u \cdot \mathbf{s}_m] \cdot \mathbf{s}_n = \nabla(\nabla\tau \cdot \mathbf{s}_m) \cdot \mathbf{s}_n. \tag{17}$$

Expand each term, and with the help of (12) and (14), we get

$$\mathbf{s}_n^T[\nabla^2 u]\mathbf{s}_m = \mathbf{s}_n^T\nabla^2\tau\mathbf{s}_m - \frac{1}{\epsilon^+}(\sigma - [\epsilon]\nabla u^- \cdot \mathbf{n})\mathbf{s}_n^T\nabla\mathbf{n}\mathbf{s}_m - (\nabla\tau\cdot\mathbf{n})\mathbf{s}_n^T\nabla\mathbf{n}\mathbf{s}_m. \tag{18}$$

**Tangential derivative of flux jump:** By differentiating the jump of flux in the tangential directions, we get another $d-1$ equations for $m = 1, \cdots, d-1$,

$$\nabla[\epsilon\nabla u \cdot \mathbf{n}] \cdot \mathbf{s}_m = \nabla\sigma \cdot \mathbf{s}_m. \tag{19}$$

After expansion,

$$\begin{aligned}\mathbf{s}_m^T[\nabla^2 u]\mathbf{n} = {} & \frac{1}{\epsilon^+}\nabla\sigma\cdot\mathbf{s}_m - \frac{[\epsilon]}{\epsilon^+}\mathbf{s}_m^T\nabla^2 u^-\mathbf{n} - \frac{[\epsilon]}{\epsilon^+}\mathbf{s}_m^T\nabla\mathbf{n}\nabla u^- \\ & - \sum_{k=1}^{d-1}(\nabla\tau\cdot\mathbf{s}_k)\mathbf{s}_m^T\nabla\mathbf{n}\mathbf{s}_k - \frac{1}{(\epsilon^+)^2}(\nabla\epsilon^+\cdot\mathbf{s}_m)(\sigma - [\epsilon]\nabla u^-\cdot\mathbf{n}) \\ & - \frac{1}{\epsilon^+}[\nabla\epsilon\cdot\mathbf{s}_m](\nabla u^-\cdot\mathbf{n}).\end{aligned} \tag{20}$$

**Jump of PDE:** The final equation comes from the jump of the PDE:

$$[-\nabla\cdot(\epsilon\nabla u) + au] = [f]. \tag{21}$$

After expansion, we have

$$\begin{aligned}[\Delta u] = {} & -\left[\frac{f}{\epsilon}\right] + \frac{a^+}{\epsilon^+}\tau + \left[\frac{a}{\epsilon}\right]u^- \\ & - \frac{1}{\epsilon^+}\sum_{k=1}^{d-1}(\nabla\tau\cdot\mathbf{s}_k)(\nabla\epsilon^+\cdot\mathbf{s}_k) - \left[\frac{\nabla\epsilon}{\epsilon}\right]\cdot\nabla u^-.\end{aligned} \tag{22}$$

Combining (20), (18), and (22), we arrive at a system of linear equations whose unknowns are the jump of the second-order derivatives:

$$G\left(\left[\frac{\partial^2 u}{\partial x_k \partial x_l}\right]\right)_{1\le k\le l\le d} = L\left(u^-, \nabla u^-, \nabla^2 u^-\right). \tag{23}$$

where $G$ is a matrix that only depends on the normal and the tangent vectors, $L$ stands for the affine function in (18), (20) and (22). In two and three dimensions, it can be shown that the absolute value of the determinant of $G$ is 1 (See Appendix A).

As an example of (23) 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 (23) is given by

$$\begin{pmatrix} s_1^2 & s_2^2 & 2s_1s_2 \\ s_1n_1 & s_2n_2 & s_1n_2 + s_2n_1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} [u_{xx}] \\ [u_{yy}] \\ [u_{xy}] \end{pmatrix} = \\ \begin{pmatrix} \mathbf{s}^T\nabla^2\tau\mathbf{s} - \frac{1}{\epsilon^+}\left(\sigma - [\epsilon]\nabla u^- \cdot \mathbf{n}\right)\mathbf{s}^T\nabla\mathbf{n}\mathbf{s} - (\nabla\tau\cdot\mathbf{n})\mathbf{s}^T\nabla\mathbf{n}\mathbf{s} \\ \frac{1}{\epsilon^+}\nabla\sigma\cdot\mathbf{s} - \frac{[\epsilon]}{\epsilon^+}(\mathbf{s}^T\nabla^2u^-\mathbf{n} + \mathbf{s}^T\nabla\mathbf{n}\nabla u^-) - (\nabla\tau\cdot\mathbf{s})\mathbf{s}^T\nabla\mathbf{n}\mathbf{s} \\ -[\frac{f}{\epsilon}] + \frac{a^+}{\epsilon^+}\tau - [\frac{a}{\epsilon}]u^- \end{pmatrix}. \tag{24}$$

By Taylor's theorem as in (4), (7) and (8), $u^-$, $\nabla u^-$ and $\nabla^2 u^-$ can all be approximated by $u_{\mathbf{j}}$, $\mathbf{j} \in B_r$ and components of $\nabla u$ and $\nabla^2 u$ at the grid point. Therefore, after substitution, (23) has the form

$$G\left(\left[\frac{\partial^2 u}{\partial x_k \partial x_l}\right]\right)_{1\le k\le l\le d} = L\left(u_{\mathbf{j}\in B_r}, \nabla u, \nabla^2 u\right) \tag{25}$$

where $\mathbf{j} \in B_r$ and $L$ represent an updated affine function based on (23) after substitution.

Recall that our goal is to write every quantities in terms of $u_{\mathbf{j}}$, $\mathbf{j} \in B_r$, $\partial u/\partial x_k$, $\partial^2 u/\partial x_k^2$, $1 \le k \le d$, called "allowable terms", which are the terms in the coupling equation (10). Here we lay out the steps to achieve this goal. Firstly, we approximated the mixed derivatives $\partial^2 u/\partial x_k \partial x_l$, $k \ne l$, using the allowable terms and the jump of the mixed derivatives (see Section 2.5):

$$\frac{\partial^2 u}{\partial x_k \partial x_l} = L\left(u_{\mathbf{j}\in B_r}, \frac{\partial u}{\partial x_k}, \frac{\partial^2 u}{\partial x_k^2}, \left[\frac{\partial^2 u}{\partial x_k \partial x_l}\right]\right) \tag{26}$$

where $L$ represent the finite difference scheme to approximate the mixed derivatives. Secondly, we substitute the above expressions (26) into (25) to eliminate the mixed derivatives:

$$G\left(\left[\frac{\partial^2 u}{\partial x_k \partial x_l}\right]\right)_{1\le k\le l\le d} = L\left(u_{\mathbf{j}\in B_r}, \nabla u, \left(\frac{\partial^2 u}{\partial x_k^2}\right)_{1\le k\le d}\right) \tag{27}$$

with an updated affine function $G$ and $L$ matrix based on (25). Notice that we have eliminated the mixed derivatives in $\nabla^2 u$ in (25). Now the right hand side

of (27) only contains the allowable terms. By solving the linear system of equations, we can write the jump of the second-order derivatives $[\partial^2 u/\partial x_k \partial x_l]$, $1 \le k \le l \le d$ using the allowable terms.

If jump of the second-order derivatives is used in (26), then we can substitute the above expressions (27) into (26) to eliminate the jump of the second-order derivatives, $[\partial^2 u/\partial x_k \partial x_l]$, $1 \le k < l \le d$. This leads to

$$\frac{\partial^2 u}{\partial x_k \partial x_l} = L\left(u_{\mathbf{j}\in B_r}, \frac{\partial u}{\partial x_k}, \frac{\partial^2 u}{\partial x_k^2}\right) \tag{28}$$

with some affine function $L$ obtained by eliminating the jump of the second-order derivatives in (26). Now the right hand side of (28), and hence (15), also only contains the allowable terms.

Finally, we substitute the expressions of $[\partial u/\partial x_k]$, and $[\partial^2 u/\partial x_k^2]$, both only involve the allowable terms at this stage, into (9). After rearrangement, we obtain one row of the coupling equation (10). The full algorithm will be summarized in Section 2.6.

## 2.5 Approximation of the mixed derivative

Depending how the interface intersects the grid, different schemes are needed to approximate the mixed derivative $\partial^2 u/\partial x_k \partial x_l$, $k \neq l$ at $\mathbf{x_i}$. Notice that we are allowed to make use of the first-order and the second-order derivatives, as they are the variables in the coupling equation (10).

In Section 2.5.1, we first describe the available finite difference schemes for the mixed derivatives in different scenarios. In Section 2.5.2, we describe the decision to choose different schemes when multiple schemes are available.

### *2.5.1 Schemes for the mixed derivatives*

Though any $\mathcal{O}(h)$ approximation suffices, we prefer schemes with smaller local truncation error. Therefore, in all the following formula, we also compute the $\mathcal{O}(h)$ term explicitly. In Fig. 4, we demonstrate examples of different scenarios, and we describe the schemes in the following:

**Case 1 (central difference)**

$$\frac{\partial^2 u}{\partial x_k \partial x_l} = \frac{1}{4h^2}\left(u_{\mathbf{i}+\mathbf{e}_k+\mathbf{e}_l} - u_{\mathbf{i}-\mathbf{e}_k+\mathbf{e}_l} - u_{\mathbf{i}+\mathbf{e}_k-\mathbf{e}_l} + u_{\mathbf{i}-\mathbf{e}_k-\mathbf{e}_l}\right) + \mathcal{O}\big(h^2\big). \tag{29}$$

**Case 2 (biased differencing with rectangular stencil)**

$$\begin{aligned}\frac{\partial^2 u}{\partial x_k \partial x_l} &= \frac{1}{2h^2}\left(u_{\mathbf{i}+\mathbf{e}_k} - u_{\mathbf{i}+\mathbf{e}_k-\mathbf{e}_l} - u_{\mathbf{i}-\mathbf{e}_k} + u_{\mathbf{i}-\mathbf{e}_k-\mathbf{e}_l}\right) \\ &\quad + \frac{1}{2}h\frac{\partial^3 u}{\partial x_k \partial x_l^2} + \mathcal{O}\big(h^2\big)\end{aligned} \tag{30}$$

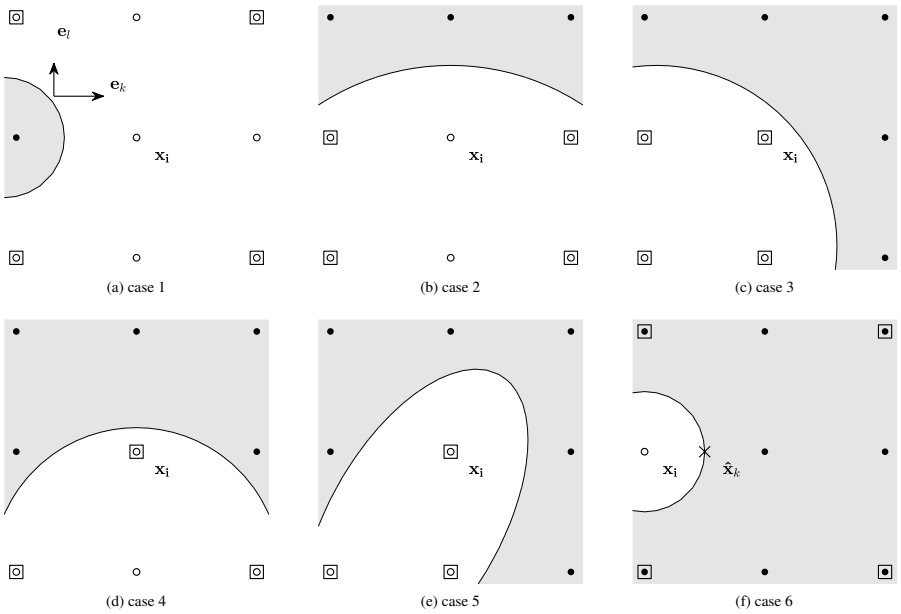



Fig. 4: Approximation of the mixed derivative $\partial^2 u/\partial x_k \partial x_l$ at $\mathbf{x_i}$. The circles and disks are grid points in $\Omega^-$ and $\Omega^+$. The $u$-values at the squares are used to approximate the mixed derivative. Case 1 is the usual central difference. Case 2 and 3 are biased difference. Case 4 uses the first-order derivatives at at $\mathbf{x_i}$. Case 5 uses the first-order and the second-order derivatives at at $\mathbf{x_i}$. Case 6 uses the jump $[\partial^2 u/\partial x_k \partial x_l]$ at $\hat{\mathbf{x}}_k$ and the mixed second-order derivative on the other side at $\mathbf{x}_{\mathbf{i}+\mathbf{e}_k}$, which is approximated by central difference.

**Case 3 (biased differencing with square stencil)**

$$\begin{aligned}\frac{\partial^2 u}{\partial x_k \partial x_l} &= \frac{1}{h^2}(u_{\mathbf{i}} - u_{\mathbf{i}-\mathbf{e}_k} - u_{\mathbf{i}-\mathbf{e}_l} + u_{\mathbf{i}-\mathbf{e}_k-\mathbf{e}_l}) \\ &\quad + \frac{1}{2}h\left(\frac{\partial^3 u}{\partial x_k \partial x_l^2} + \frac{\partial^3 u}{\partial x_k^2 \partial x_l}\right) + \mathcal{O}(h^2)\end{aligned} \tag{31}$$

**Case 4 (triangular stencil with first-order derivatives)** In case 4, we can make use of the first-order derivatives

$$\begin{aligned}\frac{\partial^2 u}{\partial x_k \partial x_l} &= \frac{1}{2h^2}\left(2h\frac{\partial u}{\partial x_k} + u_{\mathbf{i}+\mathbf{e}_k-\mathbf{e}_l} - u_{\mathbf{i}-\mathbf{e}_k-\mathbf{e}_l}\right) \\ &\quad + \frac{1}{2}h\left(\frac{\partial^3 u}{\partial x_k \partial x_l^2} + \frac{1}{3}\frac{\partial^3 u}{\partial x_k^3}\right) + \mathcal{O}(h^2)\end{aligned} \tag{32}$$

**Case 5 (triangular stencil with first and second-order derivatives)** In case 5, we can make use of the first-order and the principal second-order

derivatives

$$\begin{aligned}\frac{\partial^2 u}{\partial x_k \partial x_l} = \frac{1}{h^2}\left(h\frac{\partial u}{\partial x_k} - \frac{h^2}{2}\frac{\partial^2 u}{\partial x_k^2} - u_{\mathbf{i}-\mathbf{e}_l} + u_{\mathbf{i}-\mathbf{e}_k-\mathbf{e}_l}\right) \\ + \frac{1}{2}h\left(\frac{\partial^3 u}{\partial x_k \partial x_l^2} + \frac{\partial^3 u}{\partial x_k^2 \partial x_l} + \frac{1}{3}\frac{\partial^3 u}{\partial x_k^3}\right) + \mathcal{O}(h^2)\end{aligned} \tag{33}$$

**Case 6 (shifting to the other side)** When there are not enough grid points on the same side, we can make use of the mixed derivative on the other side of the interface and the jump of the mixed derivative

$$\frac{\partial^2 u}{\partial x_k \partial x_l} = \frac{\partial^2 u}{\partial x_k \partial x_l}(\mathbf{x}_{\mathbf{i}+\mathbf{e}_k}) - \left[\frac{\partial^2 u}{\partial x_k \partial x_l}\right]_{\hat{\mathbf{x}}_k} + \mathcal{O}(h) \tag{34}$$

where $\partial^2 u/\partial x_k \partial x_l\ (\mathbf{x_i} + h\mathbf{e}_k)$ can be approximated by $u$-values on the other side of the interface using central difference as in case 1 to 3. In case 6, the finite difference stencil will have a radius $r = 2$, as $u$-values of more than one grid point away are used. For example, as shown in Fig. 4 (case 6), $u(\mathbf{x}_{\mathbf{i}+2\mathbf{e}_k+\mathbf{e}_l})$ is used to approximate the mixed derivative at $\mathbf{x_i}$.

**Case 7 (shifting to the same side)** Though not illustrated in Fig. 4, it's also possible to approximate the mixed derivative at $\mathbf{x_i}$ by the mixed derivative of $\mathbf{x_i}$'s direct neighbor that is on the same side of the region:

$$\frac{\partial^2 u}{\partial x_k \partial x_l} = \frac{\partial^2 u}{\partial x_k \partial x_l}(\mathbf{x}_{\mathbf{i}+s\mathbf{e}_m}) + \mathcal{O}(h), \tag{35}$$

where $s = \pm 1$ and $1 \le m \le d$, and $\mathbf{x}_{\mathbf{i}+s\mathbf{e}_m}$ is on the same side of the interface as $\mathbf{x_i}$. Then the $\partial^2 u/\partial x_k \partial x_l\ (\mathbf{x}_{\mathbf{i}+s\mathbf{e}_m})$ can be approximated using case 1, 2 or 3, which only involves $u$-values on the same side of the interface. This is the same as the "shifting" strategy used in ICIM [8]. If $m = k$ or $l$, that is, we are shifting in the $kl$-plane, then the stencil have a radius $r = 2$. Otherwise, we are shifting out of the $kl$-plane, and the stencil would have a radius $r = 1$. Therefore, shifting out of plane is preferred for a more compact stencil.

We note that Case 1, 2 and 3, which approximate the mixed derivatives by neighboring u-values, are the default choices in CIM and hence ICIM. When such approximations cannot be found, Case 7 (shifting to the same side) is considered in ICIM. Case 4 and 5 are unique to our CCIM, as they make use the first-order derivatives. These two techniques are not allowed in the CIM and ICIM, because their coupling equations do not involve the first-order derivatives. Case 6 (shifting to the other side) is also unique to our CCIM, as it makes use of the jump of the mixed derivatives. These techniques (case 4, 5 and 6) help to make our stencil more compact, and allow us to handle very complicated interface even when the grid is relatively coarse.

### *2.5.2 Ordering of Different Schemes*

Multiple schemes to approximate the mixed derivatives might be available at the same grid point, and a natural question is which one to choose. Here we

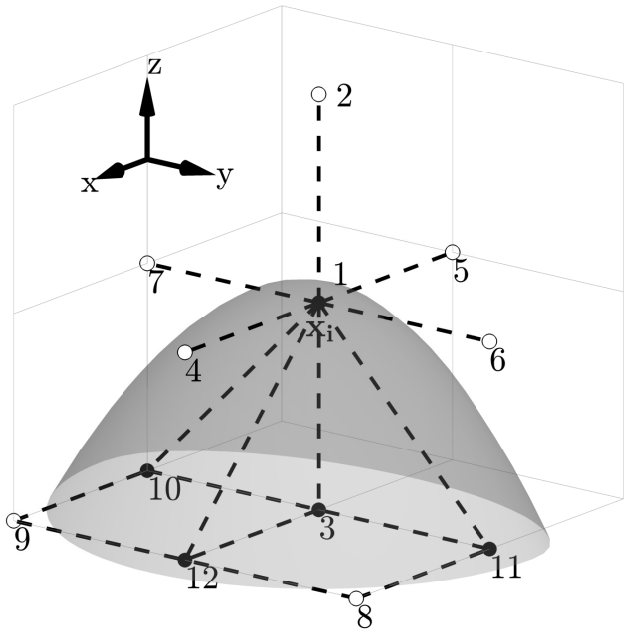



Fig. 5: An example of CCIM stencil at $\mathbf{x_i}$. The grid points in the stencil are labelled from 1 to 12. The disks and the circles are the grid points outside and inside the surface. (1,10,11) are used to approximate $u_{yz}(\mathbf{x_i})$ (see Fig. 4 case 4). (1, 3, 12) are used to approximate $u_{zx}(\mathbf{x_i})$ (Fig. 4 case 5). (8, 9, 10, 11) can be used to approximate $u_{xy}(\mathbf{x_i})$. Alternatively, $u_{xy}(\mathbf{x_i})$ can be approximated by using $u_{xy}$ at the other side (any one of 2, 4, 5, 6 or 7) and the jump condition (see Fig. 4 case 6).

discuss the criteria for choosing the scheme. Overall, we would like the scheme to be simple, compact and accurate, both in terms of the local truncation error and the condition number of the coupling matrix.

**Simplicity** For simplicity, we prefer schemes that only use $u$-values as they are easy to implement. Therefore case 1, 2 and 3, which using central differencing or biased differencing are preferred. When the derivatives have to be involved, case 4 only use first derivatives, and is preferred over case 5, which also uses second derivatives.

**Compactness** For compactness, we want to have a smaller radius $r$ for our finite difference stencil. Therefore case 6a (shifting to the other side) is the least preferable. When case 7 (shifting to the same side) has to be used, we prefer shifting out-of-plane than shifting in-plane.

**Local truncation error** For accuracy, we look at the $\mathcal{O}(h)$ of the local truncation error. Central differencing (case 1) is preferred over biased differencing (and case 2 over case 3). Case 3 or case 4 has similar local truncation

error. In general, case 6 (shifting) will leads to larger local truncation error compared with case 1 to 5.

**Condition number of the coupling equation** 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 and case 4 are available, we choose the scheme that leads to the coupling matrix with a smaller estimated condition number computed by [65]. The approximate condition number provide a good estimate of the actual condition number, which is computationally expensive to compute. The effect is shown in Section 3.

**Ordering** As a summary of the above discussion on the criteria of choosing the differencing schemes for approximating the mixed derivatives, here is how we rank the schemes from the most preferable to the least preferable: case 1 (central differencing); case 2 (biased differencing with rectangular stencil); case 3 (biased differencing with square stencil); or case 4 (triangular stencil with first-order derivatives), whichever leads to coupling matrix with smaller estimated condition number; case 5 (triangular stencil with first and second-order derivatives); case 7 (shifting to the same side); case 6 (shifting to the other side); When multiple schemes are available to approximate the mixed derivatives, we use the most preferable scheme available. Though we can construct surfaces for a 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. We note that case 5 and case 6 can be removed by refining the grid, while case 4 cannot, which is proved in [8].

### 2.6 Algorithm

We describe our method to obtain the coupling equation at an on-front point in algorithmic order in Algorithm 1. Once we have the coupling equations (10), by inverting the coupling matrix, $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$, $1 \le k \le d$ can be approximated by linear functions of $u_{\mathbf{j}}$, $\mathbf{j} \in B_r$

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 algorithm to obtain two systems of coupling equations, and choose the system with a smaller estimated condition number of the coupling matrix. The effect of this criterion is demonstrated in Section 3.1.

Let $A\mathbf{u} = f$ be the system of linear equations obtained from the finite difference approximation of the PDE (3), where $\mathbf{u}$ is the vector of unknown $u$-values at the grid points. The algorithm to assemble the system of linear equations $A\mathbf{u} = f$ is summarized in Algorithm 2.

## 3 Numerical results

We test our method in three dimensions with different surfaces. The first set of tests contains six geometric surfaces that are used in [8]. And the second set

**Algorithm 1** Coupling equation at an on-front point $\mathbf{x_i}$

**Notation:** we use $y = L(x)$ to denote "write the quantities y in terms of affine function of quantities x", where $L$ represent a generic affine function.

**for** $1 \le k \le d$ **do**
  **for** $s = \pm 1$ **do**
    **if** the interface intersects $\overline{\mathbf{x_i x_{i+se_k}}}$ at $\hat{\mathbf{x}}_k$ **then**
      **for** $1 \le j \le d$, $j \ne k$ **do**
        As described in Section 2.5

$$\frac{\partial^2 u}{\partial x_k \partial x_j} = L\left(u_{\mathbf{j}\in B_r}, \frac{\partial u}{\partial x_k}, \frac{\partial^2 u}{\partial x_k^2}, \left[\frac{\partial^2 u}{\partial x_k \partial x_j}\right]\right)$$

      **end for**
      As described in Section 2.4,

$$\left[\frac{\partial^2 u}{\partial x_k \partial x_j}\right]_{1\le k\le j\le d} = L\left(u_{\mathbf{j}\in B_r}, \left(\frac{\partial u}{\partial x_j}\right)_{1\le j\le d}, \left(\frac{\partial^2 u}{\partial x_j^2}\right)_{1\le j\le d}\right)$$

      By back substitution

$$\left(\frac{\partial^2 u}{\partial x_k \partial x_j}\right)_{1\le j\le d} = L\left(u_{\mathbf{j}\in B_r}, \left(\frac{\partial u}{\partial x_j}\right)_{1\le j\le d}, \left(\frac{\partial^2 u}{\partial x_j^2}\right)_{1\le j\le d}\right)$$

      Substitute the expression for the mixed derivatives into (15)

$$\left[\frac{\partial u}{\partial x_k}\right] = L\left(u_{\mathbf{j}\in B_r}, \left(\frac{\partial u}{\partial x_j}\right)_{1\le j\le d}, \left(\frac{\partial^2 u}{\partial x_j^2}\right)_{1\le j\le d}\right)$$

      Substituting $[\partial u/\partial x_k]$ and $[\partial^2 u/\partial x_k^2]$ into (9), and after rearrangement, this gives one row of the coupling equations (10).
    **else**
      Direct application of Taylor's theorem (4) gives one row of the coupling equations (10).
    **end if**
  **end for**
**end for**

**Algorithm 2** Linear systems of equations for the elliptic interface problem

**for** Every grid point $\mathbf{x_i}$ **do**
  **if** $\mathbf{x_i}$ is an on-front point **then**
    Use Algorithm 1 to obtain the coupling equation at $\mathbf{x_i}$. Solve the coupling equation to obtain the finite difference approximations of $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$, $1 \le k \le d$.
  **else**
    Use standard central difference to approximate $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$, $1 \le k \le d$.
  **end if**
  Substitute the approximations of $\partial u/\partial x_k$ and $\partial^2 u/\partial x_k^2$, $1 \le k \le d$, into the PDE (3), and obtain one row of the linear system of equations.
**end for**

of tests uses two complex biomolecular surfaces. These two sets are compared with our implementation of ICIM [8] with the same setup. As tests in [8] do not include the $a(\mathbf{x})$ term, the third set of tests are the same six geometric surfaces with the $a(\mathbf{x})$ 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 (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 the intersections of the interface and the grid lines, denoted as $\|\nabla u_e - \nabla u\|_{\infty,\Gamma}$. For the expanding sphere, we look at the maximum error and the Root Mean Square Error (RMSE) of the radius at all the intersections of the interface and the grid lines. All the tests are performed on a 2017 iMac with 3.5 GHz Intel Core i5 and 16GB memory. We use the AMG method implemented in the HYPRE library [66] to solve the sparse linear systems to a tolerance of $10^{-9}$.

In our work, the interface is represented by a level set function. To obtain the location of the interface on the grid segment, we use a degree-6 interpolating polynomial of the level set function and find the root using regula-falsi method. For our numerical tests, and in many applications [8,9,10], we have the expression of $\tau(x,y,z)$ and $\sigma(x,y,z)$. The quantities $\nabla\tau$, $\nabla^2\tau$, $\nabla\sigma$, the normal vectors and the tangent vectors are all approximated numerically by central differencing at the grid point, and then interpolated to the interface location. We believe that our method only requires these geometric quantities to be approximated to second-order accuracy for computational purposes. And this is verified empirically in our numerical tests. We note that since we assume interface $\Gamma$ and the interfacial jump condition $\tau$ and $\sigma$ are all smooth, we can consider any smooth extension of $\tau$ and $\sigma$ off the interface. Even if such smooth extensions are not available in an analytical form, the extensions can be computed to second order accuracy by algorithm such as the fast marching method [67].

### 3.1 Example 1

We test several geometric interfaces as in [8]. The surfaces are shown in Fig.6. 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 $(x_k,y_k,z_k) = ((-1)^{\lfloor \bar{k}/4\rfloor}\times 0.5, (-1)^{\lfloor k/2\rfloor}\times 0.5, (-1)^k\times 0.5)$
- 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

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

The computational domain is $\Omega = [-1,1]^3$. Let $\Omega^+ = \{\phi < 0\}$ and $\Omega^- = \{\phi > 0\}$, which are the interior and exterior of the surface respectively. The boundary condition is $u = 0$ on the boundary of the computational domain. The setup in this example is the same as that in [22,8]: we take $a = 0$ and use use the same exact solution (36) and the coefficient (37):

$$u_e(x,y,z) = \begin{cases} xy + x^4 + y^4 + xz^2 + \cos\big(2x + y^2 + z^3\big) & \text{if } (x,y,z) \in \Omega^+ \\ x^3 + xy^2 + y^3 + z^4 + \sin\big(3(x^2+y^2)\big) & \text{if } (x,y,z) \in \Omega^- \end{cases} \tag{36}$$

and

$$\epsilon(x,y,z) = \begin{cases} \epsilon^+ & \text{if } (x,y,z) \in \Omega^+ \\ \epsilon^- & \text{if } (x,y,z) \in \Omega^- \end{cases}. \tag{37}$$

with $\epsilon^+ = 80$ and $\epsilon^- = 2$. The source term and the jump conditions are calculated accordingly.

Fig 7 shows the convergence result of the six interfaces. The convergence of the solution at grid points is second-order, and the convergence of the gradient at the interface is close to second-order.

### *3.1.1 Effect of different schemes for the mixed derivatives*

Next we demonstrate the effect of choosing the approximation schemes for the mixed derivatives based on the estimated condition numbers of the coupling matrices. As mentioned in Section 2.5, when both case 3 and case 4 are available to approximate the mixed derivatives, we choose the scheme that gives a smaller estimated condition number of the coupling matrix. We denote this scheme as "CCIM". Alternatively, we can fix the order of preference for different methods. In "scheme 1", we always prefer case 4 (triangular stencil with first-order derivatives) to case 3 (biased differencing with square stencil).

Fig. 8 demonstrates the effect of this decision using the banana shape surface as an example. For different $N$ and for both schemes, Fig. 8a plots the maximum condition numbers (not the estimation from [65]) of all the coupling

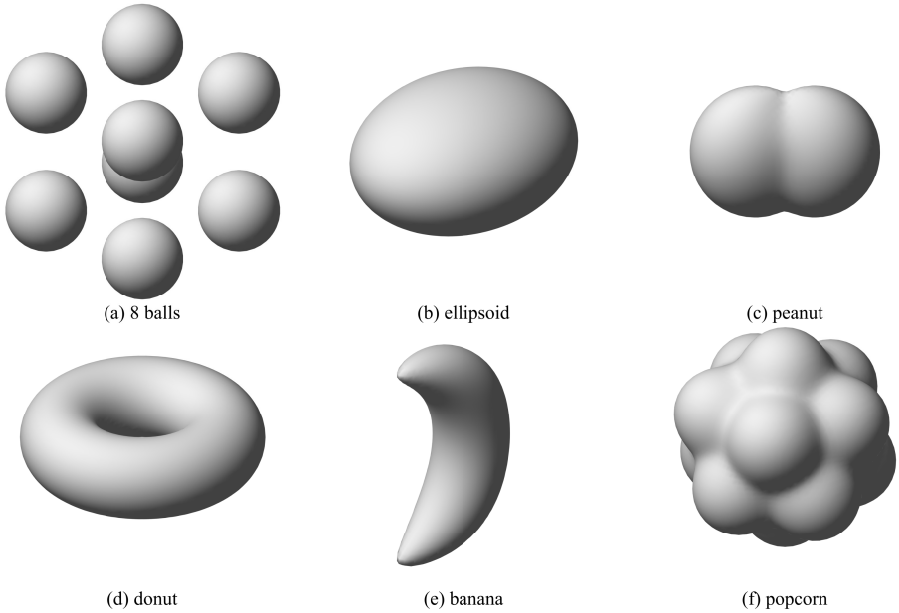
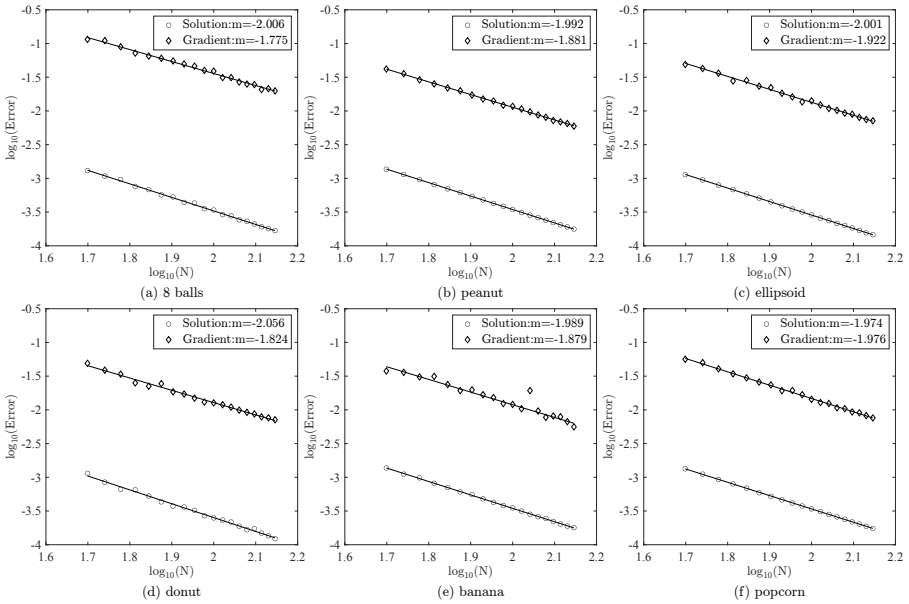



Fig. 6: The six interfaces (a) eight balls; (b) ellipsoid; (c) peanut; (d) donut; (e) banana; (f) popcorn

matrices, and Fig. 8b plots the convergence results of these two schemes. From Fig. 8a, we can see that the maximum condition numbers in CCIM are almost always smaller than those in scheme 1 (except at $N = 75$, due to the estimation error [65]). As shown in Fig. 8b, for most of the tests, CCIM and scheme 1 have roughly the same maximum error. We noticed that for $N = 110$, with scheme 1, at the interface point with 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 can get smaller error and obtain more stable convergence result in the gradient. If we prefer case 3 to case 4, then the results are similar to scheme 1: at some grid points large condition number is correlated to large error, and CCIM has more stable convergence behavior.

### *3.1.2 Investigation of the outlier error*

Though we can get a more stable convergence behavior by considering the condition number of the coupling matrices, there is an outlier of the error of the gradient for the banana interface at $N = 110$ in Fig. 8b. A detailed analysis of the error reveals that it is caused by relatively large local truncation error when approximating $u_{xz}$. Fig. 9 shows the contour line of the mixed derivative $u_{xz}$. However, due to the alignment of the surface with the grid, at $\mathbf{x}_{i,k}$, our algorithm uses the 4-point stencil $\mathbf{x}_{i,k}$, $\mathbf{x}_{i-1,k}$, $\mathbf{x}_{i,k-1}$ and $\mathbf{x}_{i-1,k-1}$



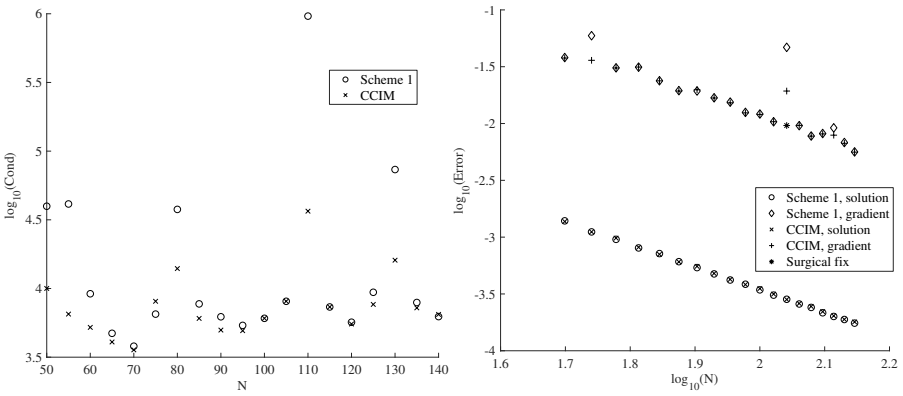

(a) 8 balls (b) peanut (c) ellipsoid

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

Fig. 7: 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 $\|\nabla u_e - \nabla u\|_{\infty,\Gamma}$. $m$ is the slope of the fitting line.

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 8b 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 the gradient.

In summary, though the overall order of convergence is second-order no matter which scheme is used to approximate the mixed second-order derivatives, a relatively large error can be caused by a large condition number of the coupling matrix, or a large local truncation error when approximating the mixed second-order derivative. When different schemes to approximate the mixed second-order derivatives are available, ideally we prefer the scheme that produces smaller local truncation error and smaller condition number of the coupling matrix. However, these two goals might be incompatible sometimes. It's time consuming to search through all the available schemes and find the one that leads the smallest condition number of the coupling matrix. It's also difficult to tell a priori which scheme gives smaller local truncation error. Therefore we try to find a middle ground by only considering the condition




(a) maximum condition number (b) Convergence

Fig. 8: Comparison of maximum condition numbers and convergence results with banana surface between scheme 1 and CCIM. In scheme 1, case 4 is preferred to case 3. In CCIM, case 3 and case 4 are chosen based on estimated condition numbers of the coupling matrices. (a) The maximum condition number of coupling matrices with different methods. (b) Log-log plot of the maximum errors in solution and gradient at the interface. The relatively large error at $N = 110$ for CCIM can be reduced by a surgical fix that choose the stencil with a smaller local truncation error.

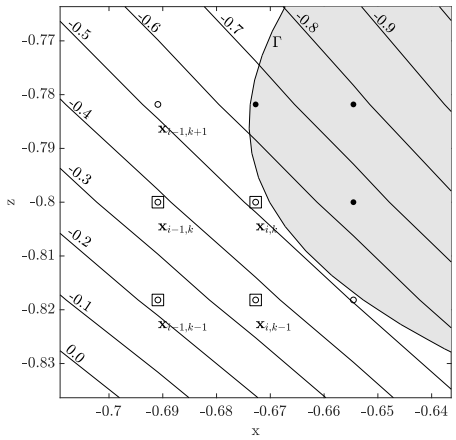



Fig. 9: Contour lines of $u_{xz}$ at the grid point where the maximum error in the gradient occurs for the banana interface at $N = 110$ The disks and the circles are grid points outside and inside the surface. Grid points marked with squares (Fig. 4 case 3) are used to approximate $u_{xz}(\mathbf{x}_{i,k})$ due to its simplicity, but has a local truncation error 0.160723, The three points stencil (Fig. 4 case 4) with $\mathbf{x}_{i,k}$, $\mathbf{x}_{i-1,k}$, $\mathbf{x}_{i-1,k+1}$ has a smaller local truncation error of 0.041853. Use this stencil to approximate $u_{xz}(\mathbf{x}_{i,k})$ would reduce the error in the gradient at this grid point.

number when both case 1 and 2 are not available but case 3 and case 4 are available.

The resulting linear system for the PDE is sparse and asymmetric, and can be solved with any "black-box" linear solvers. Fig. 10 shows the log-log plot for the number of iterations versus $N$. We used Biconjugate Gradient Stabilized Method with ILU preconditioner (abbreviated as BICG) and Algebraic Multigrid Method (AMG), both are implemented in the HYPRE library [66]. The number of iterations grows linearly with $N$ for BiCGSTAB and sub-linearly for AMG. Though AMG has better scaling property, for the range of $N$ in Fig 10, both solvers take approximately the same CPU time. Fig. 11 shows the log-log plot of the CPU time versus $N^3 \log_{10}(N)$ for the six surfaces using AMG. The slopes of the regression lines are close to 1. This scaling behavior is on par with that in ICIM (see Fig.14 in [8]). All the tests are performed on an Apple M2 Max CPU.

Empirically, our method does not encounter issue when the interface points are very close to the grid points: the minimum $\alpha$ among all the test cases in Fig. 7 is 2.3282e-18. In addition, in Appendix B, we show that our method can handle high contrast problems with a large difference in the coefficients $\epsilon^+$ and $\epsilon^-$.

### 3.2 Example 2

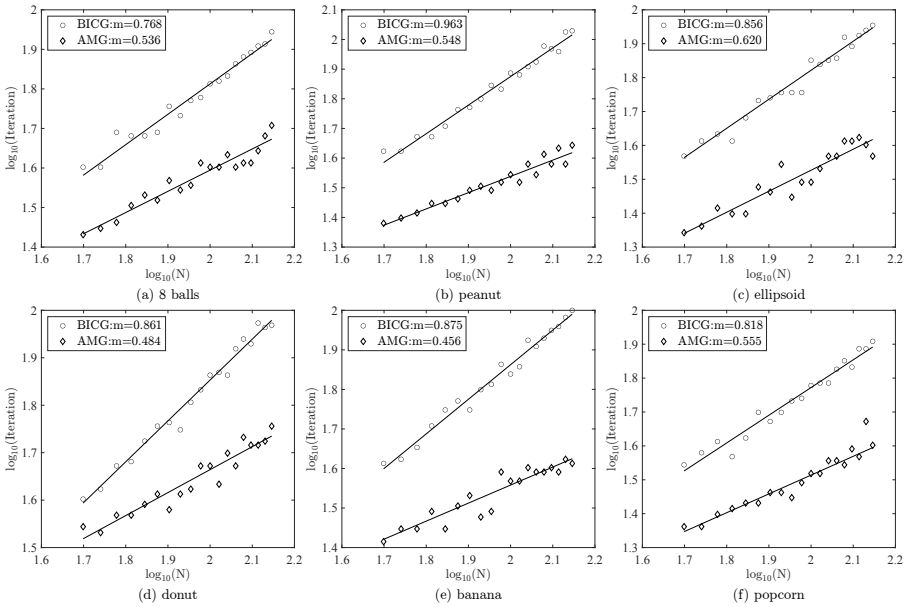
Next we test our method on two complex molecular surfaces and compare CCIM with our implementation of ICIM [8]. The solvent accessible surface describes the interface between solute and solvent. Such interfaces are complex and important in applications. We construct the surfaces as in [8]: from the PDB file of 1D63 [68] and MDM2 [69], we use the PDB2PQR [70] software to assign charges and radii using the AMBER force field. The PQR files contain information of the positions $\mathbf{p_i}$ and radii $r_i$ of the atoms. We scale the positions and radii such that the protein fit into our computation box. Then we construct the level set function of the interface as the union of smoothed bumps:

$$\phi(\mathbf{x}) = c - \sum_i \chi_\eta(r_i - \|\mathbf{x} - \mathbf{p_i}\|), \tag{38}$$

where $\chi_\eta$ is a smoothed characteristic function

$$\chi_\eta = \frac{1}{2}\left(1 + \tanh\left(\frac{x}{\eta}\right)\right). \tag{39}$$

The molecule 1D63 has 486 atoms and has a double-helix shape, as shown in Fig. 12(a). MDM2 has 1448 atoms, and the surface has a deep pocket to which other proteins can bind, as shown in Fig 13(a). We also implement ICIM [8] and compare the convergence results between CCIM and ICIM in Fig. 12 and Fig. 13.




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

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

Fig. 10: The log-log plot of the number of iterations versus $N$ for the six surfaces. In each figure, N ranges from 50 to 140 with the increment $\Delta N = 5$. Circles are the numbers of iterations using Biconjugate Gradient Stabilized Method with ILU preconditioner (abbreviated as BICG). Diamonds are the numbers of iterations using AMG. $m$ is the slope of the fitting line. Though AMG has better scaling (smaller $m$), we report that both solvers take approximately the same CPU time in this range of $N$.

As shown in Fig. 12 and Fig. 13, compared with our implementation of ICIM, the convergence results of CCIM is very robust even for complex interfaces. There is little fluctuation in the convergence results. In our ICIM implementation, the order of convergence exceeds second-order because large errors at coarse grid points skew 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.

### 3.3 Example 3

We also test our problem with the same exact solution $u_e$ (36) and coefficients $\epsilon$ (37), but with a non-zero $a(x, y, z)$ term:

$$a(x,y,z) = \begin{cases} 2\left(\sin(x) + e^{y^2}\right) & \text{if } (x,y,z) \in \Omega^- \\ 80\left(\cos(z) + e^{-y^2}\right) & \text{if } (x,y,z) \in \Omega^+. \end{cases} \tag{40}$$

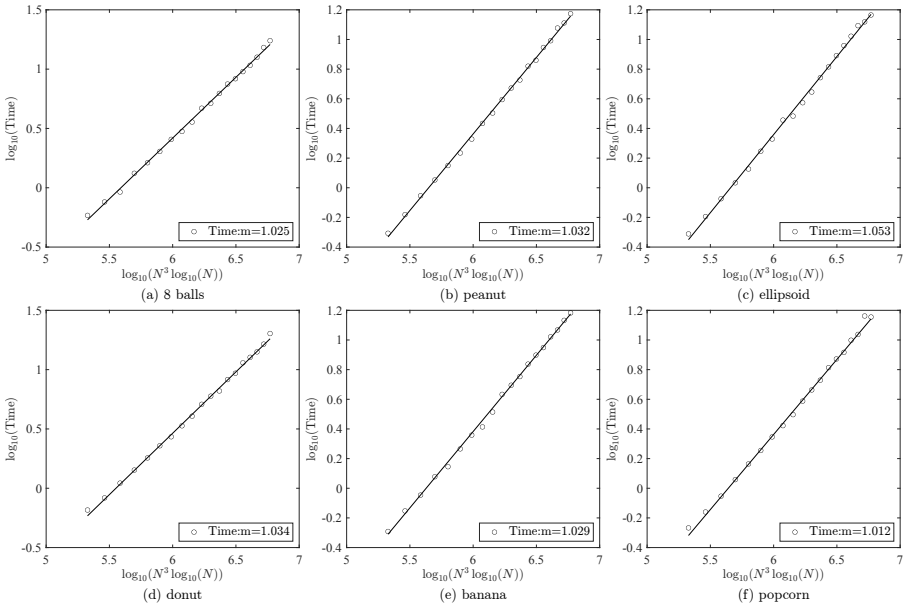



Fig. 11: The log-log plot of the CPU time versus $N^3 \log_{10}(N)$ for the six surfaces using AMG. $m$ is the slope of the fitting line.

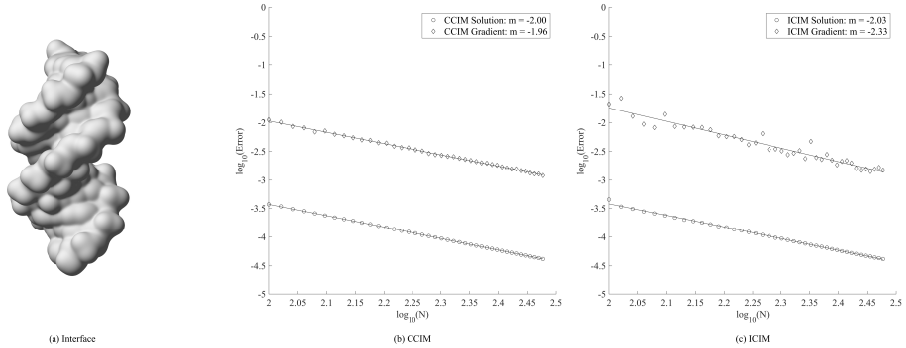



Fig. 12: Convergence result for 1D63 interface with $c = 0.25$ and $\eta = 1/40$. (a) The smooth surface of 1D63. (b) log-log plot of error by CCIM. (b) log-log plot of error by ICIM. $N$ ranges from 100 to 340 with the increment $\Delta N = 5$.

This term is not tested in CIM [22] and ICIM[8]. In Fig. 14, we show the convergence result of the six surfaces. The grids are perturbed by some uniformly distributed random numbers between 0 and mesh size $h$ in each coordinate. This changes the alignment of the surface with the grid lines compared with Fig. 7. We can see the convergence of the solution at grid points and the convergence of the gradient at the interface are second-order.

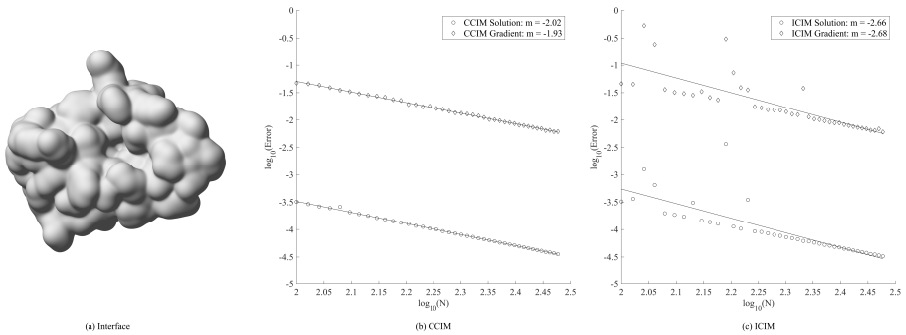



Fig. 13: Convergence result for MDM2 interface with $c = 0.25$ and $\eta = 1/30$. (a) The smooth surface of MDM2. (b) log-log plot of error by CCIM. (b) log-log plot of error by ICIM. $N$ ranges from 100 to 340 with the increment $\Delta N = 5$.

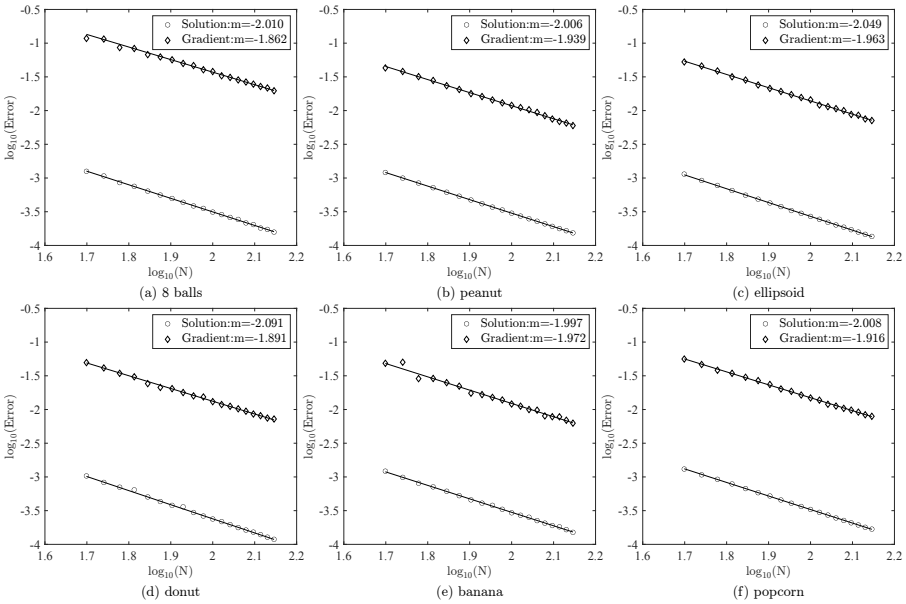



Fig. 14: The log-log plot of error with $a(\mathbf{x})$ term versus $N$ for the six surfaces. In each figure, N ranges from 50 to 140 with the increment $\Delta N = 5$. The grids are perturbed by some uniformly distributed random numbers between 0 and mesh size $h$ in each coordinate. Circles are the maximum errors of the solution $\|u_e - u\|_\infty$. Diamonds are the maximum errors of the gradient at interface $\|\nabla u_e - \nabla u\|_{\infty,\Gamma}$. $m$ is the slope of the fitting line.

### 3.4 Example 4

In this example we look at the evolution of an interface driven by the jump of the normal derivative of the solution using the level set method [23]. Suppose the surface $\Gamma$ is evolved with normal velocity $v_n = [\nabla u \cdot n]$. $\phi = \phi(\mathbf{x}, t)$ is a level set function representing the evolving surface $\Gamma = \Gamma(t)$, i.e., $\Gamma(t) = \{\mathbf{x} \mid \phi(\mathbf{x}, t) = 0\}$. The dynamics of the interface is given by the level set equation,

$$\phi_t + v_n |\nabla \phi| = 0. \tag{41}$$

We use the forward Euler method for first-order accurate time discretization, Godunov scheme for the Hamiltonian, and the Fast Marching Method [67] to extend $v_n$ to the whole computational domain.

We start with the radially symmetric exact solution

$$u_e(\mathbf{x}) = \begin{cases} \frac{1}{1+\|\mathbf{x}\|^2} & \mathbf{x} \in \Omega^- \\ -\frac{1}{1+\|\mathbf{x}\|^2} & \mathbf{x} \in \Omega^+ \end{cases} \tag{42}$$

and

$$a(\mathbf{x}) = \begin{cases} 2\sin(\|\mathbf{x}\|) & \mathbf{x} \in \Omega^- \\ 80\cos(\|\mathbf{x}\|) & \mathbf{x} \in \Omega^+ \end{cases}. \tag{43}$$

The coefficient $\epsilon$ is the same as (37). 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

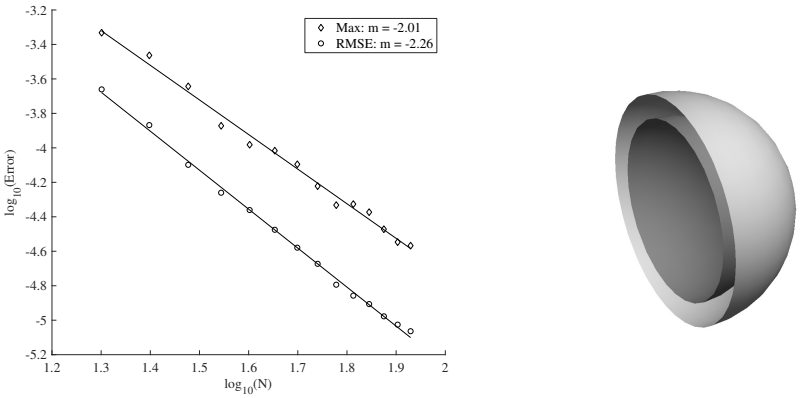
$$v_n(r) = [\nabla u \cdot n] = \frac{4r}{(1+r^2)^2}. \tag{44}$$

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

$$\frac{dr}{dt} = v_n(r), \quad r(0) = 0.5 \tag{45}$$

which can be computed to high accuracy. The result is a sphere expanding at varying time-dependent speeding.

In Fig. 15a, we look at the maximum error and the Root Mean Squared Error (RMSE) of all the radii obtained from the intersections of the surface and the grid lines at the final time $t = 0.1$ for different grid size $N$. The results are second-order accurate. In Fig. 15b, 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.




(a) Error for radii at final time. (b) Initial (inner) and final (outer) surface

Fig. 15: (a) Maximum error and Root Mean Squared error (RMSE) of the radii measured at all the intersections of the surface and the grid lines. $N$ ranges from 20 to 85 with the increment $\Delta N = 5$. (b) Initial surface (inner) and final surface (outer) for $N = 20$.

## 4 Conclusions

In this paper, we proposed the Compact Coupling Interface Method (CCIM) to solve elliptic interface boundary value problems in any dimension. Our method combines elements from the Coupling Interface Method (CIM) and Mayo's approach to Poisson's equation on irregular regions. Standard central difference schemes are used at interior points. At on-front points, coupling equations of the first-order derivatives and principal second-order derivatives are derived in a dimension-splitting approach by differentiating the jump conditions. 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 of the solution gradient at the interface. Our method has more compact finite difference stencils compared with those in CIM2 and is suitable for complex interfaces. We tested our method in three dimensions with complex interfaces, including two protein surfaces, and demonstrated that the solution and the gradient at the interface are uniformly second-order accurate, and the convergence results are very robust. We also tested our method with a moving surface whose normal velocity is given by the jump in the gradient at the interface and achieved second-order accurate interface at the final time.

## Acknowledgement

This work was funded by NSF Awards 1913144 and 2208465. The authors would like to thank Professor Bo Li for helpful discussions, guidance and sup-

port in numerical aspects of the paper. The second author would like to thank Professor Yu-Chen Shu for helpful discussions on CIM.

## References

1. F. Gibou, R. Fedkiw, Journal of Computational Physics **202**(2), 577 (2005). DOI 10.1016/j.jcp.2004.07.018
2. F. Gibou, R.P. Fedkiw, L.T. Cheng, M. Kang, Journal of Computational Physics **176**(1), 205 (2002). DOI 10.1006/jcph.2001.6977
3. D. Bochkov, F. Gibou, Journal of Computational Physics **407**, 109269 (2020). DOI 10.1016/j.jcp.2020.109269
4. T.Y. Hou, Z. Li, S. Osher, H. Zhao, Journal of Computational Physics **134**(2), 236 (1997). DOI 10.1006/jcph.1997.5689
5. R. Kafafy, T. Lin, Y. Lin, J. Wang, International Journal for Numerical Methods in Engineering **64**(7), 940 (2005). DOI 10.1002/nme.1401
6. S. Zhao, Journal of Computational Physics **229**(9), 3155 (2010). DOI 10.1016/j.jcp.2009.12.034
7. G. Hadley, Journal of Lightwave Technology **20**(7), 1210 (2002). DOI 10.1109/JLT.2002.800361
8. Y.C. Shu, I.L. Chern, C.C. Chang, Journal of Computational Physics **275**, 642 (2014). DOI 10.1016/j.jcp.2014.07.017
9. S. Zhou, L.T. Cheng, J. Dzubiella, B. Li, J.A. McCammon, Journal of Chemical Theory and Computation **10**(4), 1454 (2014). DOI 10.1021/ct401058w
10. Y. Zhong, K. Ren, R. Tsai, Journal of Computational Physics **359**, 199 (2018). DOI 10.1016/j.jcp.2018.01.021
11. P. Macklin, J. Lowengrub, Journal of Computational Physics **203**(1), 191 (2005). DOI 10.1016/j.jcp.2004.08.010
12. P. Macklin, J.S. Lowengrub, Journal of Scientific Computing **35**(2), 266 (2008). DOI 10.1007/s10915-008-9190-z
13. J. Dzubiella, J.M.J. Swanson, J.A. McCammon, Physical Review Letters **96**(8), 087802 (2006). DOI 10.1103/PhysRevLett.96.087802
14. J. Dzubiella, J.M.J. Swanson, J.A. McCammon, The Journal of Chemical Physics **124**(8), 084905 (2006). DOI 10.1063/1.2171192
15. Z. Wang, J. Che, L.T. Cheng, J. Dzubiella, B. Li, J.A. McCammon, Journal of Chemical Theory and Computation **8**(2), 386 (2012). DOI 10.1021/ct200647j
16. Z. Zhang, C.G. Ricci, C. Fan, L.T. Cheng, B. Li, J.A. McCammon, Journal of Chemical Theory and Computation (2021). DOI 10.1021/acs.jctc.0c01109
17. R.Z. Zhang, L.T. Cheng, SIAM Journal on Scientific Computing pp. B618–B645 (2023). DOI 10.1137/22M1508339
18. F. Izzo, Y. Zhong, O. Runborg, R. Tsai. Corrected Trapezoidal Rule-IBIM for linearized Poisson-Boltzmann equation (2022). DOI 10.48550/arXiv.2210.03699
19. M. Holst, F. Saied, Journal of Computational Chemistry **14**(1), 105 (1993). DOI 10.1002/jcc.540140114
20. M. Holst, R.E. Kozack, F. Saied, S. Subramaniam, Proteins **18**(3), 231 (1994). DOI 10.1002/prot.340180304
21. B. Li, SIAM Journal on Mathematical Analysis **40**(6), 2536 (2009). DOI 10.1137/080712350
22. I.L. Chern, Y.C. Shu, Journal of Computational Physics **225**(2), 2138 (2007). DOI 10.1016/j.jcp.2007.03.012
23. S. Osher, R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences (Springer-Verlag, New York, 2003). DOI 10.1007/b98879
24. S. Osher, J.A. Sethian, Journal of Computational Physics **79**(1), 12 (1988). DOI 10.1016/0021-9991(88)90002-2
25. L.T. Cheng, Z. Wang, P. Setny, J. Dzubiella, B. Li, J.A. McCammon, The Journal of Chemical Physics **131**(14), 144102 (2009). DOI 10.1063/1.3242274

26. JT. Beale, SIAM JOURNAL ON NUMERICAL ANALYSIS **42**(2), 599 (2004). DOI 10.1137/S0036142903420959
27. R. Guo, X. Zhang, Journal of Computational Physics **441**, 110445 (2021). DOI 10.1016/j.jcp.2021.110445
28. A. Guittet, M. Lepilliez, S. Tanguy, F. Gibou, Journal of Computational Physics **298**, 747 (2015). DOI 10.1016/j.jcp.2015.06.026
29. ZM. Chen, J. Zou, NUMERISCHE MATHEMATIK **79**(2), 175 (1998). DOI 10.1007/s002110050336
30. JG. Huang, J. Zou, IMA JOURNAL OF NUMERICAL ANALYSIS **22**(4), 549 (2002). DOI 10.1093/imanum/22.4.549
31. ZL. Li, WC. Wang, IL. Chern, MC. Lai, SIAM JOURNAL ON SCIENTIFIC COMPUTING **25**(1), 224 (2003). DOI 10.1137/S106482750139618X
32. H. Guo, X. Yang, Journal of Computational Physics **356**, 46 (2018). DOI 10.1016/j.jcp.2017.11.031
33. R. Guo, T. Lin, Y. Lin, Journal of Scientific Computing **79**(1), 148 (2019). DOI 10.1007/s10915-018-0847-y
34. R. Guo, SIAM Journal on Numerical Analysis **59**(2), 797 (2021). DOI 10.1137/20M133508X
35. Y. Gong, B. Li, Z. Li, SIAM Journal on Numerical Analysis **46**(1), 472 (2008). DOI 10.1137/060666482
36. R. Becker, E. Burman, P. Hansbo, Computer Methods in Applied Mechanics and Engineering **198**(41), 3352 (2009). DOI 10.1016/j.cma.2009.06.017
37. E. Burman, Comptes Rendus Mathematique **348**(21), 1217 (2010). DOI 10.1016/j.crma.2010.10.006
38. C.C. Chu, I.G. Graham, T.Y. Hou, Mathematics of Computation **79**(272), 1915 (2010)
39. P. Zunino, L. Cattaneo, C.M. Colciago, Applied Numerical Mathematics **61**(10), 1059 (2011). DOI 10.1016/j.apnum.2011.06.005
40. N. Barrau, R. Becker, E. Dubach, R. Luce, Comptes Rendus Mathematique **350**(15), 789 (2012). DOI 10.1016/j.crma.2012.09.018
41. S. Adjerid, I. Babuška, R. Guo, T. Lin, Computer Methods in Applied Mechanics and Engineering **404**, 115770 (2023). DOI 10.1016/j.cma.2022.115770
42. A. Massing, M.G. Larson, A. Logg, M.E. Rognes, Journal of Scientific Computing **61**(3), 604 (2014). DOI 10.1007/s10915-014-9838-9
43. J. Guzmán, M.A. Sánchez, M. Sarkis, Journal of Scientific Computing **73**(1), 330 (2017). DOI 10.1007/s10915-017-0415-x
44. W.F. Hu, T.S. Lin, M.C. Lai. A Discontinuity Capturing Shallow Neural Network for Elliptic Interface Problems (2021). DOI 10.48550/arXiv.2106.05587
45. H. Guo, X. Yang, Communications in Computational Physics **31**(4), 1162 (2022). DOI 10.4208/cicp.OA-2021-0201
46. A.K. Tornberg, B. Engquist, Journal of Computational Physics **200**(2), 462 (2004). DOI 10.1016/j.jcp.2004.04.011
47. A.K. Tornberg, B. Engquist, Journal of Scientific Computing **19**(1), 527 (2003). DOI 10.1023/A:1025332815267
48. CS. Peskin, in *ACTA NUMERICA 2002, VOL 11*, *Acta Numerica*, vol. 11, ed. by A. Iserles (CAMBRIDGE UNIV PRESS, THE PITT BUILDING, TRUMPINGTON ST, CAMBRIDGE CB2 1RP, CAMBS, ENGLAND, 2002), pp. 479–517. DOI 10.1017/S0962492902000077
49. CS. PESKIN, JOURNAL OF COMPUTATIONAL PHYSICS **25**(3), 220 (1977). DOI 10.1016/0021-9991(77)90100-0
50. R.J. LeVeque, Z. Li, SIAM Journal on Numerical Analysis **31**(4), 1019 (1994). DOI 10.1137/0731054
51. Z. Li, K. Ito, SIAM Journal on Scientific Computing **23**(1), 339 (2001). DOI 10.1137/S1064827500370160
52. Z. Li, SIAM Journal on Numerical Analysis **35**(1), 230 (1998). DOI 10.1137/S0036142995291329
53. Z. Li, H. Ji, X. Chen, SIAM Journal on Numerical Analysis **55**(2), 570 (2017). DOI 10.1137/15M1040244
54. X.D. Liu, T. Sideris, Mathematics of Computation **72**(244), 1731 (2003). DOI 10.1090/S0025-5718-03-01525-4

55. X.D. Liu, R.P. Fedkiw, M. Kang, Journal of Computational Physics **160**(1), 151 (2000). DOI 10.1006/jcph.2000.6444
56. R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, Journal of Computational Physics **152**(2), 457 (1999). DOI 10.1006/jcph.1999.6236
57. A. Wiegmann, KP. Bube, SIAM JOURNAL ON NUMERICAL ANALYSIS **37**(3), 827 (2000). DOI 10.1137/S0036142997328664
58. PA. Berthelsen, JOURNAL OF COMPUTATIONAL PHYSICS **197**(1), 364 (2004). DOI 10.1016/j.jcp.2003.12.003
59. Y.C. Zhou, S. Zhao, M. Feig, G.W. Wei, Journal of Computational Physics **213**(1), 1 (2006). DOI 10.1016/j.jcp.2005.07.022
60. S. Yu, Y. Zhou, G.W. Wei, Journal of Computational Physics **224**(2), 729 (2007). DOI 10.1016/j.jcp.2006.10.030
61. S. Yu, G.W. Wei, Journal of Computational Physics **227**(1), 602 (2007). DOI 10.1016/j.jcp.2007.08.003
62. Y.C. Shu, C.Y. Kao, I.L. Chern, C.C. Chang, JOURNAL OF COMPUTATIONAL PHYSICS **229**(24), 9246 (2010). DOI 10.1016/j.jcp.2010.09.001
63. P. Smereka, Journal of Computational Physics **211**(1), 77 (2006). DOI 10.1016/j.jcp.2005.05.005
64. A. Mayo, SIAM Journal on Numerical Analysis **21**(2), 285 (1984). DOI 10.1137/0721021
65. W.W. Hager, SIAM Journal on Scientific and Statistical Computing **5**(2), 311 (1984). DOI 10.1137/0905023
66. R.D. Falgout, U.M. Yang, in *Computational Science — ICCS 2002*, ed. by P.M.A. Sloot, A.G. Hoekstra, C.J.K. Tan, J.J. Dongarra (Springer, Berlin, Heidelberg, 2002), Lecture Notes in Computer Science, pp. 632–641. DOI 10.1007/3-540-47789-6_66
67. J.A. Sethian, Proceedings of the National Academy of Sciences **93**(4), 1591 (1996). DOI 10.1073/pnas.93.4.1591
68. D.G. Brown, M.R. Sanderson, E. Garman, S. Neidle, Journal of Molecular Biology **226**(2), 481 (1992). DOI 10.1016/0022-2836(92)90962-J
69. P.H. Kussie, S. Gorina, V. Marechal, B. Elenbaas, J. Moreau, A.J. Levine, N.P. Pavletich, Science (New York, N.Y.) **274**(5289), 948 (1996). DOI 10.1126/science.274.5289.948
70. T.J. Dolinsky, J.E. Nielsen, J.A. McCammon, N.A. Baker, Nucleic Acids Research **32**(suppl_2), W665 (2004). DOI 10.1093/nar/gkh381
71. E. Burman, J. Guzmán, M.A. Sánchez, M. Sarkis, IMA Journal of Numerical Analysis **38**(2), 646 (2018). DOI 10.1093/imanum/drx017
72. E. Burman, P. Zunino, in *Frontiers in Numerical Analysis - Durham 2010*, ed. by J. Blowey, M. Jensen, Lecture Notes in Computational Science and Engineering (Springer, Berlin, Heidelberg, 2012), pp. 227–282. DOI 10.1007/978-3-642-23914-4_4

## Statements and Declarations

Funding

This work was funded by NSF Awards 1913144 and 2208465.

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Data Availability

The code is available at `https://github.com/Rayzhangzirui/ccim`.

## A Differentiation of the jump condition

In this appendix, we detail the calculation of the formula for approximation of $[\,\partial^2 u/\partial x_k^2\,]$, through equations involving terms of $[\nabla^2 u]$, as found in Section 2.4. In the following derivation, quantities related to $f$, $a$, $\epsilon$ and $\tau$ are all known. Our final goal is to write the jump of the second derivatives $[\nabla^2 u]$ in terms of the known quantities and the one-sided derivatives $\nabla u^-$, $\nabla^2 u^-$. Since our interface is smooth, we can consider any smooth extension of $\tau$ and $\sigma$ off the interface, therefore quantities such as $\nabla^2\tau$ and $\nabla\tau\cdot\mathbf{n}$ are well-defined.

We first consider jumps of the first derivatives of $u$ at the interface, especially in terms of jumps of the normal and tangential derivatives of $u$. Taking tangential derivatives on both sides of $u^+ - u^- = [u] = \tau$, we get

$$[\nabla u\cdot\mathbf{s}_j] = \nabla u^+\cdot\mathbf{s}_j - \nabla u^-\cdot\mathbf{s}_j = \nabla\tau\cdot\mathbf{s}_j,$$

for $j = 1,\ldots,d-1$. On the other hand, with

$$[\epsilon v] = \epsilon^+[v] + [\epsilon]v^-, \tag{12 revisited}$$

we can get, when $v = \nabla u\cdot n$,

$$[\nabla u\cdot\mathbf{n}] = \frac{1}{\epsilon^+}(\sigma - [\epsilon](\nabla u^-\cdot\mathbf{n})). \tag{13 revisited}$$

These equations for the jumps of the normal and tangential derivatives of $u$ can then be used to get $[\nabla u]$, from

$$[\nabla u] = [\nabla u\cdot\mathbf{n}]\mathbf{n} + \sum_{j=1}^{d-1}(\nabla\tau\cdot\mathbf{s}_j)\mathbf{s}_j. \tag{11 revisited}$$

Thus having handled jumps of first derivatives of $u$, we now turn our attention jumps of second derivatives and derive equations for the terms of $[\nabla^2 u]$ in three ways.

**Tangential derivative of jump of tangential derivative:** We can get equations on terms of $[\nabla^2 u]$ by starting with the jump of the tangential derivative of $u$ along $\mathbf{s}_m$, namely $[\nabla u\cdot s_m]$, and taking its tangential derivative along $\mathbf{s}_n$, giving $\nabla[\nabla u\cdot s_m]\cdot s_n$. This quantity can be written as

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

However, we also have, using (11), that

$$\begin{aligned}\nabla\left[\nabla u\cdot\mathbf{s}_m\right]\cdot\mathbf{s}_n &= \mathbf{s}_n^T\left[\nabla^2 u\right]\mathbf{s}_m + \mathbf{s}_n^T\nabla\mathbf{s}_m\left[\nabla u\right]\\ &= \mathbf{s}_n^T\left[\nabla^2 u\right]\mathbf{s}_m + \frac{1}{\epsilon^+}\left(\sigma - [\epsilon]\nabla u^-\cdot\mathbf{n}\right)\mathbf{s}_n^T\nabla\mathbf{s}_m\mathbf{n} + \sum_{j=1}^{d-1}(\nabla\tau\cdot\mathbf{s}_j)\mathbf{s}_n^T\nabla\mathbf{s}_m\mathbf{s}_j\end{aligned}$$

and, additionally, that

$$\begin{aligned}\nabla(\nabla\tau\cdot\mathbf{s}_m)\cdot\mathbf{s}_n &= \mathbf{s}_n^T\nabla^2\tau\mathbf{s}_m + s_n^T\nabla\mathbf{s}_m\nabla\tau\\ &= \mathbf{s}_n^T\nabla^2\tau\mathbf{s}_m + (\nabla\tau\cdot\mathbf{n})\mathbf{s}_n^T\nabla\mathbf{s}_m\mathbf{n} + \sum_{j=1}^{d-1}(\nabla\tau\cdot\mathbf{s}_j)\mathbf{s}_n^T\nabla\mathbf{s}_m\mathbf{s}_j.\end{aligned}$$

Thus, equating these and solving for the term with jumps in second derivatives of $u$, especially simplifying using the fact that

$$\nabla\mathbf{n}\ \mathbf{s}_j = -\nabla\mathbf{s}_j\mathbf{n}, \quad j = 1,\ldots,d-1,$$

from taking the gradient on both sides of $\mathbf{n}\cdot\mathbf{s}_j = 0$ for $j = 1,\ldots,d-1$, we get (18):

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

**Tangential derivative of flux jump:** For more equations on terms of $[\nabla^2 u]$, we consider the tangential derivative of the jump $[\epsilon \nabla u \cdot \mathbf{n}]$ along $\mathbf{s}_m$, which satisfies

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

where $\sigma$ is the jump of the flux. Expanding, we can get

$$\begin{aligned}
\nabla \left[\epsilon \nabla u \cdot \mathbf{n}\right] \cdot \mathbf{s}_m &= \mathbf{s}_m^T \nabla \left(\epsilon^+ \left[\nabla u\right] + \left[\epsilon\right] \nabla u^-\right) \mathbf{n} + \mathbf{s}_m^T \nabla \mathbf{n} (\epsilon^+ \left[\nabla u\right] + \left[\epsilon\right] \nabla u^-) \\
&= \epsilon^+ \mathbf{s}_m^T \left[\nabla^2 u\right] \mathbf{n} + \mathbf{s}_m^T \nabla \epsilon^+ [\nabla u] \cdot \mathbf{n} + \left[\epsilon\right] \mathbf{s}_m^T \nabla^2 u^- \mathbf{n} + \mathbf{s}_m^T \left[\nabla \epsilon\right] \nabla u^- \cdot \mathbf{n} \\
&\quad + \epsilon^+ \mathbf{s}_m^T \nabla \mathbf{n} \left[\nabla u\right] + \left[\epsilon\right] \mathbf{s}_m^T \nabla \mathbf{n} \nabla u^-.
\end{aligned}$$

Now, substituting $[\nabla u]$ by equation (11) and rearranging, we get (20):

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

**Jump of PDE:** For our final equations on terms of $[\nabla^2 u]$, we consider the original PDE. In $\Omega^+$, we have

$$-\epsilon^+ \Delta u^+ - \nabla \epsilon^+ \cdot \nabla u^+ + a^+ u^+ = f^+,$$

while in $\Omega^-$, we have

$$-\epsilon^- \Delta u^- - \nabla \epsilon^- \cdot \nabla u^- + a^- u^- = f^-.$$

Dividing these equations by $\epsilon^+$ and $\epsilon^-$, respectively, and finding the jump from their difference, we get

$$\begin{aligned}
[\Delta u] &= -\left[\frac{f}{\epsilon}\right] - \left[\frac{\nabla \epsilon}{\epsilon} \cdot \nabla u\right] + \left[\frac{a}{\epsilon} u\right] \\
&= -\left[\frac{f}{\epsilon}\right] - \frac{\nabla \epsilon^+}{\epsilon^+} \cdot [\nabla u] - \left[\frac{\nabla \epsilon}{\epsilon}\right] \cdot \nabla u^- + \frac{a^+}{\epsilon^+} [u] + \left[\frac{a}{\epsilon}\right] u^- \\
&= -\left[\frac{f}{\epsilon}\right] - \left[\frac{\nabla \epsilon}{\epsilon}\right] \cdot \nabla u^- + \frac{a^+}{\epsilon^+} [u] + \left[\frac{a}{\epsilon}\right] u^- \\
&\quad - \frac{1}{\epsilon^+} \left( [\nabla u \cdot \mathbf{n}] (\nabla \epsilon^+ \cdot \mathbf{n}) + \sum_{j=1}^{d-1} (\nabla \tau \cdot \mathbf{s}_j)(\nabla \epsilon^+ \cdot \mathbf{s}_j) \right),
\end{aligned}$$

which is (22).

The left hand side of equations (18) (20) and (22):

$$\mathbf{s}_n^T [\nabla^2 u] \mathbf{s}_m, \quad \mathbf{s}_m^T [\nabla^2 u] \mathbf{n}, \quad [\Delta u] \quad \text{for } m = 1, \ldots, d-1 \text{ and } n = m, \ldots, d-1$$

where $[\nabla^2 u]$ are the unknown quantities, can be written as in the form of a matrix-vector product

$$G \left( \left[ \frac{\partial^2 u}{\partial x_k \partial x_l} \right] \right)_{1 \le k \le l \le d}.$$

where $G$ is a matrix that only depends on the normal and the tangent vectors, and the vector is a half-vectorization of the jump of the symmetric Hessian matrix $[\nabla^2 u]$.

We can show that the absolute value of the determinant of $G$ is 1 in two and three dimensions. Since the equations are obtained at some interface point $\hat{\mathbf{x}}$, we can use a local coordinate system such that $\mathbf{s}_i = \mathbf{e}_i$ for $i = 1, \ldots, d-1$ and $\mathbf{n} = \mathbf{e}_d$. By choosing a specific

ordering for the equations and the half-vectorization, we can write the matrix-vector product in 2D as

$$\begin{pmatrix} \mathbf{s}_1[\nabla^2 u]\mathbf{s}_1 \\ \mathbf{s}_1[\nabla^2 u]\mathbf{n} \\ [\Delta u] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} [u_{xx}] \\ [u_{xy}] \\ [u_{yy}] \end{pmatrix}$$

And in 3D

$$\begin{pmatrix} \mathbf{s}_1[\nabla^2 u]\mathbf{s}_1 \\ \mathbf{s}_2[\nabla^2 u]\mathbf{s}_2 \\ [\Delta u] \\ \mathbf{s}_1[\nabla^2 u]\mathbf{s}_2 \\ \mathbf{s}_1[\nabla^2 u]\mathbf{n} \\ \mathbf{s}_2[\nabla^2 u]\mathbf{n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} [u_{xx}] \\ [u_{xy}] \\ [u_{yy}] \\ [u_{xy}] \\ [u_{xz}] \\ [u_{yz}] \end{pmatrix}$$

Therefore, the determinant of $G$ is $\pm 1$, depending on the ordering of the equations and the half-vectorization.

## B High Contrast Problem

One difficulty of interface problems is the so-called large contrast problem, where the ratio of the coefficients $\epsilon^+/\epsilon^- \gg 1$. There are several works that analyze the high-contrast problems in the context of unfitted Nitsche finite element method [71] and unfitted finite element methods [38, 43, 72, 71]. For example, it can be shown that the flux error estimate is independent of the contrast for a class of unfitted Nitsche finite element methods [71].

Here we numerically demonstrate that our method is robust for high contrast problems. We consider the same exact solution (36) and coefficients (37) as in Example 1, but with $\epsilon^- = 1$ and $\epsilon^+ = 1$ or $\epsilon^+ = 1e6$. Fig 16 shows the convergence result of the six interfaces. We see that both the solution and the gradient at the interface are uniformly second-order accurate for both cases. We also see that the error between the two cases is similar, demonstrating the robustness of our method for high contrast problems. The theoretical analysis of the high contrast problem is beyond the scope of this paper and will be studied in the future.

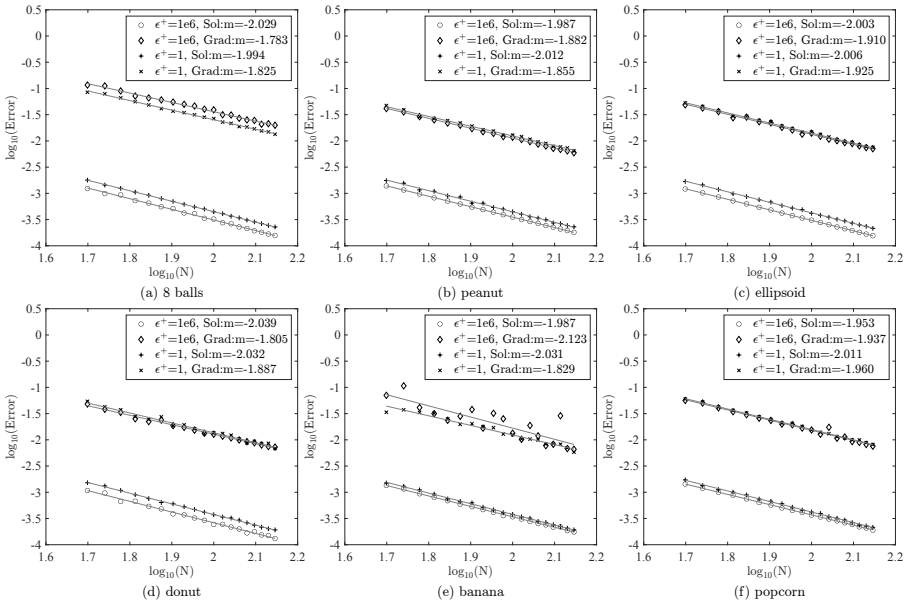



Fig. 16: The log-log plot of the error versus $N$ for the six surfaces. $\epsilon^- = 1$ and $\epsilon^+ = 1$ or $\epsilon^+ = 1e6$. In each figure, N ranges from 50 to 140 with the increment $\Delta N = 5$. "Sol" denotes maximum errors of the solution $\|u_e - u\|_\infty$. "Grad" denotes the maximum errors of the gradient at interface $\|\nabla u_e - \nabla u\|_{\infty,\Gamma}$. $m$ is the slope of the fitting line.