Journal of Computational Physics 230 (2011) 7567–7597

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A Correction Function Method for Poisson problems with interface jump conditions

Alexandre Noll Marques [a,*], Jean-Christophe Nave [b], Rodolfo Ruben Rosales [c]

[a] Department of Aeronautics and Astronautics, Massachusetts Institute of Technology Cambridge, MA 02139-4307, United States
[b] Department of Mathematics and Statistics, McGill University Montreal, Quebec, Canada H3A 2K6
[c] Department of Mathematics, Massachusetts Institute of Technology Cambridge, MA 02139-4307, United States

ARTICLE INFO

Article history:
Received 4 October 2010
Received in revised form 7 May 2011
Accepted 9 June 2011
Available online 3 July 2011

Keywords:
Poisson equation
Interface jump condition
Ghost Fluid Method
Gradient augmented level set method
High accuracy
Hermite cubic spline

ABSTRACT

In this paper we present a method to treat interface jump conditions for constant coefficients Poisson problems that allows the use of standard "black box" solvers, without compromising accuracy. The basic idea of the new approach is similar to the Ghost Fluid Method (GFM). The GFM relies on corrections applied on nodes located across the interface for discretization stencils that straddle the interface. If the corrections are solution-independent, they can be moved to the right-hand-side (RHS) of the equations, producing a problem with the same linear system as if there were no jumps, only with a different RHS. However, achieving high accuracy is very hard (if not impossible) with the "standard" approaches used to compute the GFM correction terms.

In this paper we generalize the GFM correction terms to a correction function, defined on a band around the interface. This function is then shown to be characterized as the solution to a PDE, with appropriate boundary conditions. This PDE can, in principle, be solved to any desired order of accuracy. As an example, we apply this new method to devise a 4th order accurate scheme for the constant coefficients Poisson equation with discontinuities in 2D. This scheme is based on (i) the standard 9-point stencil discretization of the Poisson equation, (ii) a representation of the correction function in terms of bicubics, and (iii) a solution of the correction function PDE by a least squares minimization. Several applications of the method are presented to illustrate its robustness dealing with a variety of interface geometries, its capability to capture sharp discontinuities, and its high convergence rate.

1. Introduction

1.1. Motivation and background information

In this paper we present a new and efficient method to solve the constant coefficients Poisson equation in the presence of discontinuities across an interface, with a high order of accuracy. Solutions of the Poisson equation with discontinuities are of fundamental importance in the description of fluid flows separated by interfaces (*e.g.* the contact surfaces for immiscible multiphase fluids, or fluids separated by a membrane) and other multiphase diffusion phenomena. Over the last three decades, several methods have been developed to solve problems of this type numerically [1–17]. However, obtaining a high order of accuracy still poses great challenges in terms of complexity and computational efficiency.

* Corresponding author.
E-mail address: noll@mit.edu (A.N. Marques).

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.06.014

When the solution is known to be smooth, it is easy to obtain highly accurate finite-difference discretizations of the Poisson equation on a regular grid. Furthermore, these discretizations commonly yield symmetric and banded linear systems, which can be inverted efficiently [18]. On the other hand, when singularities occur (*e.g.* discontinuities) across internal interfaces, some of the regular discretization stencils will straddle the interface, which renders the whole procedure invalid.

Several strategies have been proposed to tackle this issue. Peskin [1] introduced the Immersed Boundary Method (IBM) [1,10], in which the discontinuities are re-interpreted as additional (singular) source terms concentrated on the interface. These singular terms are then "regularized" and appropriately spread out over the regular grid—in a "thin" band enclosing the interface. The result is a first order scheme that smears discontinuities. In order to avoid this smearing of the interface information, LeVeque and Li [3] developed the Immersed Interface Method (IIM) [3,4,11,13], which is a methodology to modify the discretization stencils, taking into consideration the discontinuities at their actual locations. The IIM guarantees second order accuracy and sharp discontinuities, but at the cost of added discretization complexity and loss of symmetry.

The new method advanced in this paper builds on the ideas introduced by the Ghost Fluid Method (GFM) [6–9,12,14,19]. The GFM is based on defining both actual and "ghost" fluid variables at every node on a narrow band enclosing the interface. The ghost variables work as extensions of the actual variables across the interface—the solution on each side of the interface is assumed to have a smooth extension into the other side. This approach allows the use of standard discretizations everywhere in the domain. In most GFM versions, the ghost values are written as the actual values, plus corrections that are independent of the underlying solution to the Poisson problem. Hence, the corrections can be pre-computed, and moved into the source term for the equation. In this fashion the GFM yields the same linear system as the one produced by the problem without an interface, except for changes in the right-hand-side (sources) only, which can then be inverted just as efficiently.

The key difficulty in the GFM is the calculation of the correction terms, since the overall accuracy of the scheme depends heavily on the quality of the assigned ghost values. In [6–9,12] the authors develop first order accurate approaches to deal with discontinuities. In the present work, we show that for the constant coefficients Poisson equation we can generalize the GFM correction term (at each ghost point) concept to that of a correction function defined on a narrow band enclosing the interface. Hence we call this new approach the *Correction Function Method* (CFM). This correction function is then shown to be characterized as the solution to a PDE, with appropriate boundary conditions on the interface—see Section 4. Thus, at least in principle, one can calculate the correction function to any order of accuracy, by designing algorithms to solve the PDE that defines it. In this paper we present examples of 2nd and 4th order accurate schemes (to solve the constant coefficients Poisson equation, with discontinuities across interfaces, in 2D) developed using this general framework.

A key point (see Section 5) in the scheme developed here is the way we solve the PDE defining the correction function. This PDE is solved in a weak fashion using a least squares minimization procedure. This provides a flexible approach that allows the development of a robust scheme that can deal with the geometrical complications of the placement of the regular grid stencils relative to the interface. Furthermore, this approach is easy to generalize to 3D, or to higher orders of accuracy.

1.2. Other related work

It is relevant to note other developments to solve the Poisson equation under similar circumstances—multiple phases separated by interfaces—but with different interface conditions. The Poisson problem with Dirichlet boundary conditions, on an irregular boundary embedded in a regular grid, has been solved to second order of accuracy using fast Poisson solvers [19], finite volume [5], and finite differences [20–23] approaches. In particular, Gibou et al. [21] and Jomaa and Macaskill [22] have shown that it is possible to obtain symmetric discretizations of the embedded Dirichlet problem, up to second order of accuracy. Gibou and Fedkiw [23] have developed a fourth order accurate discretization of the problem, at the cost of giving up symmetry. More recently, the same problem has also been solved, to second order of accuracy, in non-graded adaptive Cartesian grids by Chen et al. [24]. Furthermore, the embedded Dirichlet problem is closely related to the Stefan problem modeling dendritic growth, as described in [25,26].

The finite-element community has also made significant progress in incorporating the IIM and similar techniques to solve the Poisson equation using embedded grids. In particular the works by Gong et al. [15], Dolbow and Harari [16], and Bedrossian et al. [17] describe second order accurate finite-element discretizations that result in symmetric linear systems. Moreover, in these works the interface (or boundary) conditions are imposed in a weak fashion, which bears some conceptual similarities with the CFM presented here, although the execution is rather different.

1.3. Interface representation

Another issue of primary importance to multiphase problems is the representation of the interface (and its tracking in unsteady cases). Some authors (see [1,19,20]) choose to represent the interface explicitly, by tracking interface particles. The location of the neighboring particles is then used to produce local interpolations (*e.g.* splines), which are then applied to compute geometric information—such as curvature and normal directions. Although this approach can be quite accurate, it requires special treatment when the interface undergoes either large deformations or topological changes—such as mergers or splits. Even though we are not concerned with these issues in this paper, we elected to adopt an implicit representation, to avoid complications in future applications. In an implicit representation, the interface is given as the zero level of a function that is defined everywhere in the regular grid—the level set function [27]. In particular, we adopted the Gradient-Augmented Level Set (GA-LS) method [28]. With this extension of the level set method, we can obtain highly accurate

representations of the interface, and other geometric information, with the additional advantage that this method uses only local grid information. We discuss the question of interface representation in more detail in Section 5.6.

1.4. Organization of the paper

The remainder of the paper is organized as follows. In Section 2 we introduce the Poisson problem that we seek to solve. In Section 3 the basic idea behind the solution method, and its relationship to the GFM, are explored. Next, in Section 4, we introduce the concept of the correction function and show how it is defined by a PDE problem. In Section 5 we apply this new framework to build a 4th order accurate scheme in 2D. Since the emphasis of this paper is on high-order schemes, we describe the 2nd order accurate scheme in Appendix C. Next, in Section 6 we demonstrate the robustness and accuracy of the 2D scheme by applying it to several example problems. The conclusions are in Section 7. In Appendix A we review some background material, and notation, on bicubic interpolation. Finally, in Appendix B we discuss some technical issues regarding the construction of the sets where the correction function is solved for.

2. Definition of the problem

Our objective is to solve the constant coefficients Poisson's equation in a domain Ω in which the solution is discontinuous across a co-dimension 1 interface Γ, which divides the domain into the subdomains Ω^+ and Ω^-, as illustrated in Fig. 1. We use the notation u^+ and u^- to denote the solution in each of the subdomains. Let the discontinuities across Γ be given in terms of two functions defined on the interface: $a = a(\vec{x})$ for the jump in the function values, and $b = b(\vec{x})$ for the jump in the normal derivatives. Furthermore, assume Dirichlet boundary conditions on the "outer" boundary $\partial\Omega$ (see Fig. 1). Thus the problem to be solved is

$$\nabla^2 u(\vec{x}) = f(\vec{x}) \quad \text{for } \vec{x} \in \Omega, \tag{1a}$$

$$[u]_\Gamma = a(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \tag{1b}$$

$$[u_n]_\Gamma = b(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \tag{1c}$$

$$u(\vec{x}) = g(\vec{x}) \quad \text{for } \vec{x} \in \partial\Omega, \tag{1d}$$

where

$$[u]_\Gamma = u^+(\vec{x}) - u^-(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \tag{2a}$$

$$[u_n]_\Gamma = u_n^+(\vec{x}) - u_n^-(\vec{x}) \quad \text{for } \vec{x} \in \Gamma. \tag{2b}$$

Throughout this paper, $\vec{x} = (x_1, x_2, \ldots) \in \mathbb{R}^v$ is the spatial vector (where $v = 2$, or $v = 3$), and ∇^2 is the Laplacian operator defined by

$$\nabla^2 = \sum_{i=1}^{v} \frac{\partial^2}{\partial x_i^2}. \tag{3}$$

Furthermore,

$$u_n = \hat{n} \cdot \vec{\nabla} u = \hat{n} \cdot (u_{x_1}, u_{x_2}, \ldots) \tag{4}$$

denotes the derivative of u in the direction of $\hat{n}$, the unit vector normal to the interface Γ pointing towards Ω^+ (see Fig. 1).

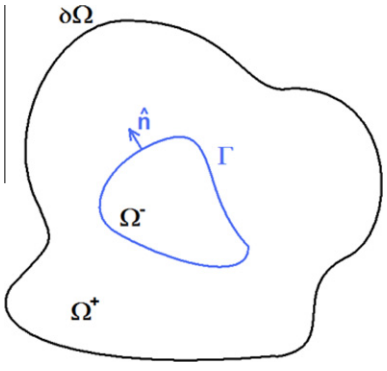


Fig. 1. Example of a domain Ω, with an interface Γ.

It is important to note that our method focuses on the discretization of the problem in the vicinity of the interface only. Thus, the method is compatible with any set of boundary conditions on $\partial\Omega$, not just Dirichlet.

3. Solution method – the basic idea

To achieve the goal of efficient high order discretizations of Poisson equation in the presence of discontinuities, we build on the idea of the Ghost Fluid Method (GFM). In essence, we use a standard discretization of the Laplace operator (on a domain without an interface Γ) and modify the right-hand-side (RHS) to incorporate the jump conditions across Γ. Thus, the resulting linear system can be inverted as efficiently as in the case of a solution without discontinuities.

Let us first illustrate the key concept in the GFM with a simple example, involving a regular grid and the standard second order discretization of the 1D analog of the problem we are interested in. Thus, consider the problem of discretizing the equation $u_{xx} = f(x)$ in some interval $\Omega = \{x : x_L < x < x_R\}$, where u is discontinuous across some point x_Γ—hence Ω^+ (respectively Ω^-) is the domain $x_L < x < x_\Gamma$ (respectively $x_\Gamma < x < x_R$). Then, see Fig. 2, when trying to approximate u_{xx} at a grid point x_i such that $x_i < x_\Gamma < x_{i+1}$, we would like to write

$$u^+_{xx_i} \approx \frac{u^+_{i-1} - 2u^+_i + u^+_{i+1}}{h^2}, \tag{5}$$

where $h = x_{j+1} - x_j$ is the grid spacing. However, we do not have information on u^+_{i+1}, but rather on u^-_{i+1}. Thus, the idea is to estimate a correction for u^-_{i+1}, to recover u^+_{i+1}, such that Eq. (5) can be applied:

$$u^+_{xx_i} \approx \frac{u^+_{i-1} - 2u^+_i + \overbrace{(u^-_{i+1} + D_{i+1})}^{u^+_{i+1}}}{h^2}, \tag{6}$$

where $D_{i+1} = u^+_{i+1} - u^-_{i+1}$ is the correction term. Now we note that, if D_{i+1} can be written as a correction that is independent on the solution u, then it can be moved to the RHS of the equation, and absorbed into f. That is

$$\frac{u^+_{i-1} - 2u^+_i + u^-_{i+1}}{h^2} = f_i - \frac{D_{i+1}}{h^2}. \tag{7}$$

This allows the solution of the problem with prescribed discontinuities using the same discretization as the one employed to solve the simpler problem without an interface—which leads to a great efficiency gain.

Remark 1. The error in estimating D is crucial in determining the accuracy of the final discretization. Liu et al. [9] introduced a dimension-by-dimension linear extrapolation of the interface jump conditions, to get a first order approximation for D. Our new method is based on generalizing the idea of a correction term to that of a *correction function*, for which we can write an equation. One can then obtain high accuracy representations for D by solving this equation, without the complications which dimension-by-dimension (with Taylor expansions) approaches run into. □

Remark 2. An additional advantage of the correction function approach is that D can be calculated at any point near the interface Γ. Hence it can be used with any finite differences discretization of the Poisson equation, without regard to the particulars of its stencil (as would be the case with any approach based on Taylor expansions). □

4. The correction function and the equation defining it

As mentioned earlier, the aim here is to generalize the correction term concept to that of a *correction function*, and then to find an equation (a PDE, with appropriate boundary conditions) that uniquely characterizes the *correction function*. Then, at least in principle, one can design algorithms to solve the PDE in order to obtain solutions to the *correction function* of any desired order of accuracy.

Let us begin by considering a small region Ω_Γ enclosing the interface Γ, defined as the set of all the points within some distance $\mathcal{R}$ of Γ, where $\mathcal{R}$ is of the order of the grid size h. As we will see below, we would like to have $\mathcal{R}$ as small as possible.

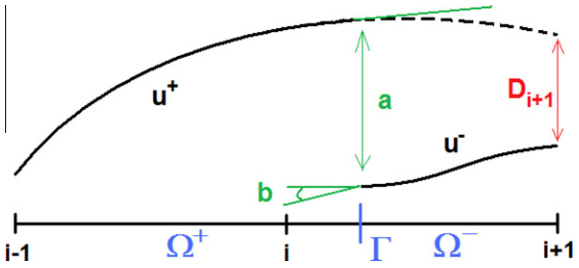


Fig. 2. Example in 1D of a solution with a jump discontinuity.

On the other hand, Ω_Γ has to include all the points where the CFM requires corrections to be computed, which means[1] that $\mathcal{R}$ cannot be smaller than $\sqrt{2}h$. In addition, algorithmic considerations (to be seen later) force $\mathcal{R}$ to be slightly larger than this last value.

Next, we assume we that can extrapolate both u^+ and u^-, so that they are valid everywhere within Ω_Γ, in such a way that they satisfy the Poisson equations

$$\nabla^2 u^+(\vec{x}) = f^+(\vec{x}) \quad \text{for } \vec{x} \in \Omega_\Gamma, \tag{8a}$$

$$\nabla^2 u^-(\vec{x}) = f^-(\vec{x}) \quad \text{for } \vec{x} \in \Omega_\Gamma, \tag{8b}$$

where f^+ and f^- are smooth enough (see Remark 3 below) extensions of the source term f to Ω_Γ. In particular, notice that the introduction of f^+ and f^- allows the possibility of a discontinuous source term across Γ. The *correction function* is then defined by $D(\vec{x}) = u^+(\vec{x}) - u^-(\vec{x})$.

Taking the difference between the equations in (8), and using the jump conditions (1b)–(1c), yields

$$\nabla^2 D(\vec{x}) = f^+(\vec{x}) - f^-(\vec{x}) = f_D(\vec{x}) \quad \text{for } \vec{x} \in \Omega_\Gamma, \tag{9a}$$

$$D(\vec{x}) = a(\vec{x}) \quad \text{for } \vec{x} \in \Gamma, \tag{9b}$$

$$D_n(\vec{x}) = b(\vec{x}) \quad \text{for } \vec{x} \in \Gamma. \tag{9c}$$

This achieves the aim of having the *correction function* defined by a set of equations, with some provisos—see Remark 4 below. Note that:

1. If $f^+(\vec{x}) = f^-(\vec{x})$, for $\vec{x} \in \Omega_\Gamma$, then $f_D(\vec{x}) = 0$, for $\vec{x} \in \Omega_\Gamma$.
2. Eq. (9c) imposes the true jump condition in the normal direction, whereas some versions of the GFM rely on a dimension-by-dimension approximation of this condition (see Ref. [9]).

Remark 3. The smoothness requirement on f^+ and f^- is tied up to how accurate an approximation to the correction term D is needed. For example, if a 4th order algorithm is used to find D, this will (generally) necessitate D to be at least C^4 for the errors to actually be 4th order. Hence, in this case, $f_D = f^+ - f^-$ must be C^2. □

Remark 4. Eq. (9) is an elliptic Cauchy problem for D in Ω_Γ. In general, such problems are ill-posed. However, we are seeking for solutions within the context of a numerical approximation where

(a) There is a frequency cut-off in both the data $a = a(\vec{x})$ and $b = b(\vec{x})$, and the description of the curve Γ.
(b) We are interested in the solution only a small distance away from the interface Γ, where this distance vanishes simultaneously with the inverse of the cut-off frequency in point (a).

What (a) and (b) mean is that the arbitrarily large growth rate for arbitrarily small perturbations, which is responsible for the ill-posedness of the Cauchy problem in Eq. (9), does not occur within the special context where we need to solve the problem. This large growth rate does not occur because, for the solutions of the Poisson equation, the growth rate for a perturbation of wave number $0 < k < \infty$ along some straight line, is given by $e^{2\pi k d}$—where d is the distance from the line. However, by construction, in the case of interest to us kd is bounded. □

Remark 5. Let us be more precise, and define a number characterizing how well posed the discretized version of Eq. (9) is, by

$$\alpha = \text{largest growth rate possible},$$

where growth is defined relative to the size of a perturbation to the solution on the interface. This number is determined by $\mathcal{R}$ (the "radius" of Ω_Γ) as the following calculation shows: First of all, there is no loss of generality in assuming that the interface is flat, provided that the numerical grid is fine enough to resolve Γ. In this case, let us introduce an orthogonal coordinate system $\vec{y}$ on Γ, and let d be the signed distance to Γ (say, $d > 0$ in Ω^-). Expanding the perturbations in Fourier modes along the interface, the typical mode has the form

$$\varphi_{\vec{k}} = e^{2\pi i \vec{k}\cdot\vec{y} \pm 2\pi k d},$$

where $\vec{k}$ is the Fourier wave vector, and $k = |\vec{k}|$. The shortest wave-length that can be represented on a grid with mesh size $0 < h \ll 1$ corresponds to $k = k_{\max} = 1/(2h)$. Hence, we obtain the estimate

$$\alpha \approx e^{\pi \mathcal{R}_c / h}$$

for the maximum growth rate. □

[1] For the particular discretization of the Laplace operator that we use in this paper.

Remark 6. Clearly, α is intimately related to the condition number for the discretized problem—see Section 5. In fact, at leading order, the two numbers should be (roughly) proportional to each other—with a proportionality constant that depends on the details of the discretization. For the discretization used in this paper (described further below), $\sqrt{2}h \leqslant \mathcal{R} \leqslant 2\sqrt{2}h$, which leads to the rough estimate $85 < \alpha < 7{,}200$. On the other hand, the observed condition numbers vary between 5,000 and 10,000. Hence, the actual condition numbers are only slightly higher than α for the ranges of grid sizes h that we used (we did not explore the asymptotic limit $h \to 0$). □

Remark 7. Eq. (9) depends on the known inputs for the problem only. Namely: f^+, f^-, a, and b. Consequently D does not depend on the solution u. Hence, after solving for D, we can use a discretization for u that does not involve the interface: Whenever u is discontinuous, we evaluate D where the correction is needed, and transfer these values to the RHS. □

Remark 8. When developing an algorithm for a linear Cauchy problem, such as the one in Eq. (9), the two key requirements are consistency and stability. In particular, when the solution depends on the "initial conditions" globally, stability (typically) imposes stringent constraints on the "time" step for any local (explicit) scheme. This would seem to suggest that, in order to solve Eq. (9), a "global" (involving the whole domain Ω_Γ) method will be needed. This, however, is not true: because we need to solve Eq. (9) for one "time" step only—*i.e.* within an $\mathcal{O}(h)$ distance from Γ, stability is not relevant. Hence, consistency is enough, and a fully local scheme is possible. In the algorithm described in Section 5 we found that, for (local) quadrangular patches, the Cauchy problem leads to a well behaved algorithm when the length of the interface contained in each patch is of the same order as the diagonal length of the patch. This result is in line with the calculation in Remark 5: we want to keep the "wavelength" (along Γ) of the perturbations introduced by the discretization as long as possible. In particular, this should then minimize the condition number for the local problems—see Remark 6. □

5. A 4th order accurate scheme in 2D

5.1. Overview

In this section we use the general ideas presented earlier to develop a specific example of a 4th order accurate scheme in 2D. Before proceeding with an in-depth description of the scheme, we highlight a few key points:

(a) We discretize Poisson's equation using a compact 9-point stencil. Compactness is important since it is directly related to the size of $\mathcal{R}$, which has a direct impact on the problem's conditioning—see Remarks 4–6.
(b) We approximate D using bicubic interpolations (bicubics), each valid in a small neighborhood $\Omega_\Gamma^{i,j}$ of the interface. This guarantees local 4th order accuracy with only 12 interpolation parameters—see [28]. Each $\Omega_\Gamma^{i,j}$ corresponds to a point in the grid at which the standard discretization of Poisson's equation involves a stencil that straddles the interface Γ.
(c) The domains $\Omega_\Gamma^{i,j}$ are rectangular regions, each enclosing a portion of Γ, and all the nodes where D is needed to complete the discretization of the Poisson equation at the (i,j)th stencil. Each is a sub-domain of Ω_Γ.
(d) Starting from (b) and (c), we design a local solver that provides an approximation to D inside each domain $\Omega_\Gamma^{i,j}$.
(e) The interface Γ is represented using the Gradient-Augmented Level Set approach—see [28]. This guarantees a local 4th order representation of the interface, as required to keep the overall accuracy of the scheme.
(f) In each $\Omega_\Gamma^{i,j}$, we solve the PDE in (9) in a least squares sense. Namely: First we define an appropriate positive quadratic integral quantity J_P—Eq. (17)—for which the solution is a minimum (actually, zero). Next we substitute the bicubic approximation for the solution into J_P, and discretize the integrals using Gaussian quadrature. Finally, we find the bicubic parameters by minimizing the discretized J_P.

Remark 9. Solving the PDE in a least squares sense is crucial, since an algorithm is needed that can deal with the myriad ways in which the interface Γ can be placed relative to the fixed rectangular grid used to discretize Poisson's equation. This approach provides a scheme that (i) is robust with respect to the details of the interface geometry, (ii) has a formulation that is (essentially) dimension independent—there are no fundamental changes from 2D to 3D, and (iii) has a clear theoretical underpinning that allows extensions to higher orders, or to other discretizations of the Poisson equation. □

5.2. Standard Stencil

We use the standard 4th order accurate 9-point discretization of Poisson's equation[2]:

$$L^5 u_{i,j} + \frac{1}{12}\left(h_x^2 + h_y^2\right)\hat{\partial}_{xx}\hat{\partial}_{yy}u_{i,j} = f_{i,j} + \frac{1}{12}\left(h_x^2 (f_{xx})_{i,j} + h_y^2 (f_{yy})_{i,j}\right), \tag{10}$$

where L^5 is the second order 5-point discretization of the Laplace operator:

$$L^5 u_{i,j} = \hat{\partial}_{xx}u_{i,j} + \hat{\partial}_{yy}u_{i,j} \tag{11}$$

[2] Notice that here we allow for the possibility of different grid spacings in each direction.

and

$$\hat{\partial}_{xx}u_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2}, \tag{12}$$

$$\hat{\partial}_{yy}u_{i,j} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2}. \tag{13}$$

The terms $(f_{xx})_{i,j}$ and $(f_{yy})_{i,j}$ may be given analytically (if known), or computed using appropriate second order discretizations.

In the absence of discontinuities, Eq. (10) provides a compact 4th order accurate representation of Poisson's equation. In the vicinity of the discontinuities at the interface Γ, we define an appropriate domain $\Omega_\Gamma^{i,j}$, and compute the correction terms necessary to Eq. (10)—as described in detail next.

To understand how the correction terms affect the discretization, let us consider the situation depicted in Fig. 3. In this case, the node (i,j) lies in Ω^+ while the nodes $(i+1,j)$, $(i+1,j+1)$, and $(i,j+1)$ are in Ω^-. Hence, to be able to use Eq. (10), we need to compute $D_{i+1,j}$, $D_{i+1,j+1}$, and $D_{i,j+1}$.

After having solved for D where necessary (see Section 5.3 and Section 5.4), we modify Eq. (10) and write

$$L^5 u_{i,j} + \frac{1}{12}\left(h_x^2 + h_y^2\right)\hat{\partial}_{xx}\hat{\partial}_{yy}u_{i,j} = f_{i,j} + \frac{1}{12}\left(h_x^2 (f_{xx})_{i,j} + h_y^2 (f_{yy})_{i,j}\right) + C_{i,j}, \tag{14}$$

which differs from Eq. (10) by the terms $C_{i,j}$ on the RHS only. Here the $C_{i,j}$ are the CFM correction terms needed to complete the stencil across the discontinuity at Γ. In the particular case illustrated in Fig. 3, we have

$$C_{i,j} = \left[\frac{1}{6}\frac{\left(h_x^2 + h_y^2\right)}{(h_x h_y)^2} - \frac{1}{h_y^2}\right] D_{i+1,j} + \left[\frac{1}{6}\frac{\left(h_x^2 + h_y^2\right)}{(h_x h_y)^2} - \frac{1}{h_x^2}\right] D_{i,j+1} - \frac{1}{12}\frac{\left(h_x^2 + h_y^2\right)}{(h_x h_y)^2} D_{i+1,j+1}. \tag{15}$$

Similar formulas apply for the other possible arrangements of the Poisson's equation stencil relative to the interface Γ.

5.3. Definition of $\Omega_\Gamma^{i,j}$

There is some freedom on how to define $\Omega_\Gamma^{i,j}$. The basic requirements are

(i) $\Omega_\Gamma^{i,j}$ should be a rectangle.
(ii) The edges of $\Omega_\Gamma^{i,j}$ should be parallel to the grid lines.
(iii) $\Omega_\Gamma^{i,j}$ should be small, since the problem's condition number increases exponentially with the distance from Γ—see Remarks 5 and 6.
(iv) $\Omega_\Gamma^{i,j}$ should contain all the nodes where D is needed. For the example, in Fig. 3 we need to know $D_{i+1,j+1}$, $D_{i+1,j}$, and $D_{i,j+1}$. Hence, in this case, $\Omega_\Gamma^{i,j}$ should include the nodes $(i+1,j+1)$, $(i+1,j)$, and $(i,j+1)$.
(v) $\Omega_\Gamma^{i,j}$ should contain a segment of Γ, with a length that is as large as possible—*i.e.* comparable to the length of the diagonal of $\Omega_\Gamma^{i,j}$. This follows from the calculation in Remark 5, which indicates that the wavelength of the perturbations (along Γ) introduced by the discretization should be as long as possible. This should then minimize the condition number for the local problem—see Remark 6.

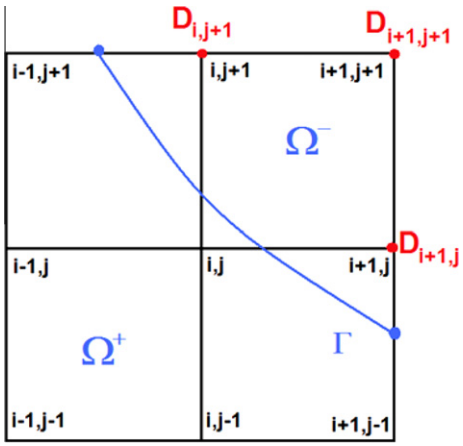


Fig. 3. The 9-point compact stencil next to the interface Γ.

Requirements (i) and (ii) are needed for algorithmic convenience only, and do not arise from any particular argument in Section 4. Thus, in principle, this convenience could be traded for improvements in other areas—for example, for better condition numbers for the local problems, or for additional flexibility in dealing with complex geometries. However, for simplicity, in this paper we enforce (i) and (ii). As explained earlier (see Remark 9), we solve Eq. (9) in a least squares sense. Hence integrations over $\Omega_\Gamma^{i,j}$ are required. It is thus useful to keep $\Omega_\Gamma^{i,j}$ as simple as possible.

A discussion of various aspects regarding the proper definition of $\Omega_\Gamma^{i,j}$ can be found in Appendix B. For instance, the requirement in item (ii) is convenient only when an implicit representation of the interface is used. Furthermore, although the definition of $\Omega_\Gamma^{i,j}$ presented here proved robust for all the applications of the 4th order accurate scheme (see Section 6), there are specific geometrical arrangements of the interface for which (ii) results in extremely elongated $\Omega_\Gamma^{i,j}$. These elongated geometries can have negative effects on the accuracy of the scheme. We noticed this effect in the 2nd order accurate version of the method described in Appendix C. These issues are addressed by the various algorithms (of increasing complexity) presented in Appendix B.

With the points above in mind, here (for simplicity) we define $\Omega_\Gamma^{i,j}$ as the smallest rectangle that satisfies the requirements in (i), (ii), (iv), and (v)—then (iii) follows automatically. Hence $\Omega_\Gamma^{i,j}$ can be constructed using the following three easy steps:

1. Find the coordinates $(x_{\min_\Gamma}, x_{\max_\Gamma})$ and $(y_{\min_\Gamma}, y_{\max_\Gamma})$ of the smallest rectangle satisfying condition (ii), which completely encloses the section of the interface Γ contained by the region covered by the 9-point stencil.
2. Find the coordinates $(x_{\min_D}, x_{\max_D})$ and $(y_{\min_D}, y_{\max_D})$ of the smallest rectangle satisfying condition (ii), which completely encloses all the nodes at which D needs to be known.
3. Then $\Omega_\Gamma^{i,j}$ is the smallest rectangle that encloses the two previous rectangles. Its edges are given by

$$x_{\min} = \min(x_{\min_\Gamma}, x_{\min_D}), \tag{16a}$$

$$x_{\max} = \max(x_{\max_\Gamma}, x_{\max_D}), \tag{16b}$$

$$y_{\min} = \min(y_{\min_\Gamma}, y_{\min_D}), \tag{16c}$$

$$y_{\max} = \max(y_{\max_\Gamma}, y_{\max_D}). \tag{16d}$$

Fig. 4 shows an example of $\Omega_\Gamma^{i,j}$ defined using these specifications.

Remark 10. Notice that for each node next to the interface we construct a domain $\Omega_\Gamma^{i,j}$. When doing so, we allow the domains to overlap. For example, the domain $\Omega_\Gamma^{i,j}$ shown in Fig. 4 is used to determine $C_{i,j}$. It should be clear that $\Omega_\Gamma^{i-1,j+1}$ (used to determine $C_{i-1,j+1}$), and $\Omega_\Gamma^{i+1,j-1}$ (used to determine $C_{i+1,j-1}$), each will overlap with $\Omega_\Gamma^{i,j}$.

The consequence of these overlaps is that different computed values for D at the same node can (in fact, will) happen—depending on which domain is used to solve the local Cauchy problem. However, because we solve for D—within each $\Omega_\Gamma^{i,j}$—to 4th order accuracy, any differences that arise from this multiple definition of D lie within the order of accuracy of the scheme. Since it is convenient to keep the computations local, the values of D resulting from the domain $\Omega_\Gamma^{i,j}$, are used to evaluate the correction term $C_{i,j}$. □

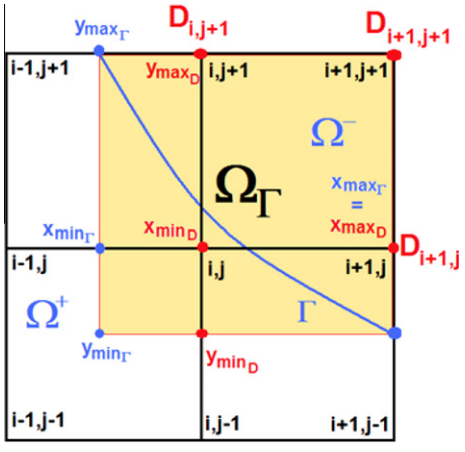


Fig. 4. The set $\Omega_\Gamma^{i,j}$ for the situation in Fig. 3.

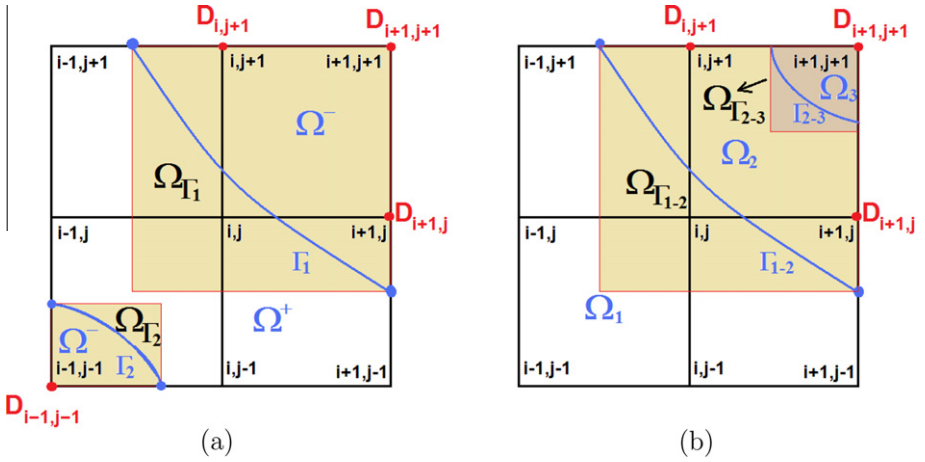


Fig. 5. Configuration where multiple $\Omega^{i,j}_{\Gamma_2}$ are needed in the same stencil. (a) Same interface crossing the stencil multiple times. (b) Distinct interfaces crossing the same stencil.

Remark 11. While rare, cases where a single interface crosses the same stencil multiple times can occur. In Section 6.3 we present such an example. A simple approach to deal with situations like this is as follows: First associate each node where the correction function is needed to a particular piece of interface crossing the stencil (say, the closest one). Then define one $\Omega^{i,j}_{\Gamma}$ for each of the individual pieces of interface crossing the stencil.

For example, Fig. 5(a) depicts a situation where the stencil is crossed by two pieces of the same interface (Γ_1 and Γ_2), with D needed at the nodes $(i+1,j+1)$, $(i+1,j)$, $(i,j+1)$, and $(i-1,j-1)$. Then, first associate: (i) $(i+1,j+1)$, $(i+1,j)$, and $(i,j+1)$ to Γ_1, and (ii) $(i-1,j-1)$ to Γ_2. Second, define

1. $\Omega^{i,j}_{\Gamma_1}$ is the smallest rectangle, parallel to the grid lines, that includes Γ_1 and the nodes $(i+1,j+1)$, $(i+1,j)$, and $(i,j+1)$.
2. $\Omega^{i,j}_{\Gamma_2}$ is the smallest rectangle, parallel to the grid lines, that includes Γ_2 and the node $(i-1,j-1)$.

After the multiple $\Omega^{i,j}_{\Gamma}$ are defined within a given stencil, the local Cauchy problem is solved for each $\Omega^{i,j}_{\Gamma}$ separately. For example, in the case shown in Fig. 5(a), the solution for D inside $\Omega^{i,j}_{\Gamma_1}$ is done completely independent of the solution for D inside $\Omega^{i,j}_{\Gamma_2}$. The decoupling between multiple crossings renders the CFM flexible and robust enough to handle complex geometries without any special algorithmic considerations. □

Remark 12. When multiple distinct interfaces are involved, a single stencil can be crossed by different interfaces – *e.g.*: see Sections 6.5 and 6.6. This situation is similar to the one described in Remark 11, but with an additional complication: there may occur distinct domain regions that are not separated by an interface, but rather by a third (or more) regions between them. An example is shown in Fig. 5(b), where Γ_{1-2} and Γ_{2-3} are not part of the same interface. Here Γ_{1-2} is the interface between Ω_1 and Ω_2, while Γ_{2-3} is the interface between Ω_2 and Ω_3. There is no interface Γ_{1-3} separating Ω_1 from Ω_3, hence no jump conditions between these regions are provided. Nonetheless, $D_{1-3} = (u_3 - u_1)$ is needed at $(i+1,j+1)$.

Situations such as these can be easily handled by noticing that we can distinguish between primary (*e.g.* D_{1-2} and D_{2-3}) and secondary correction functions, which can be written in terms of the primary functions (*e.g.* $D_{1-3} = D_{1-2} + D_{2-3}$) and need not be computed directly. Hence we can proceed exactly as in Remark 11, except that we have to make sure that the intersections of the regions where the primary correction functions are computed include the nodes where the secondary correction functions are needed. For example, in the particular case in Fig. 5(b), we define

1. $\Omega^{i,j}_{\Gamma_{1-2}}$ is the smallest rectangle, parallel to the grid lines, that includes Γ_{1-2} and the nodes $(i+1,j+1)$, $(i+1,j)$, and $(i,j+1)$.
2. $\Omega^{i,j}_{\Gamma_{2-3}}$ is the smallest rectangle, parallel to the grid lines, that includes Γ_{2-3} and the node $(i+1,j+1)$. □

5.4. Solution of the local Cauchy problem

Since we use a 4th order accurate discretization of the Poisson problem, we need to find D with 4th order errors (or better) to keep the overall accuracy of the scheme—see Section 5.7. Hence we approximate D using cubic Hermite splines (bicubic interpolants in 2D), which guarantees 4th order accuracy—see [28]. Note also that, even though the example scheme developed here is for 2D, this representation can be easily extended to any number of dimensions.

Given a choice of basis functions,[3] we solve the local Cauchy problem defined in Eq. (9) in a least squares sense, using a minimization procedure. Since we do not have boundary conditions, but interface conditions, we must resort to a minimization functional that is different from the standard one associated with the Poisson equation. Thus we impose the Cauchy interface conditions by using a penalization method. The functional to be minimized is then

$$J_P = \left(\ell_c^{i,j}\right)^3 \int_{\Omega_\Gamma^{i,j}} [\nabla^2 D(\vec{x}) - f_D(\vec{x})]^2 dV + c_P \int_{\Gamma \cap \Omega_\Gamma^{i,j}} [D(\vec{x}) - a(\vec{x})]^2 dS + c_P \left(\ell_c^{i,j}\right)^2 \int_{\Gamma \cap \Omega_\Gamma^{i,j}} [D_n(\vec{x}) - b(\vec{x})]^2 dS, \tag{17}$$

where $c_P > 0$ is the penalization coefficient used to enforce the interface conditions, and $\ell_c^{i,j} > 0$ is a characteristic length associated with $\Omega_\Gamma^{i,j}$—we used the shortest side length. Clearly J_P is a quadratic functional whose minimum (zero) occurs at the solution to Eq. (9).

In order to compute D in the domain $\Omega_\Gamma^{i,j}$, its bicubic representation is substituted into the formula above for J_P, with the integrals approximated by Gaussian quadratures—in this paper we used six quadrature points for the 1D line integrals, and 36 points for the 2D area integrals. The resulting discrete problem is then minimized. Because the bicubic representation for D involves 12 basis polynomials, the minimization problem produces a 12×12 (self-adjoint) linear system.

Remark 13. We explored the option of enforcing the interface conditions using Lagrange multipliers. While this second approach yields good results, our experience shows that the penalization method is better. □

Remark 14. The scaling using $\ell_c^{i,j}$ in Eq. (17) is so that all the three terms in the definition of J_p behave in the same fashion as the size of $\Omega_\Gamma^{i,j}$ changes with (i,j), or when the computational grid is refined.[4] This follows because we expect that

$$\nabla^2 D - f = \mathcal{O}\left(\ell_c^2\right),$$

$$D - a = \mathcal{O}\left(\ell_c^4\right),$$

$$D_n - b = \mathcal{O}\left(\ell_c^3\right).$$

Hence each of the three terms in Eq. (17) should be $\mathcal{O}\left(\ell_c^9\right)$. □

Remark 15. Once all the terms in Eq. (17) are guaranteed to scale the same way with the size of $\Omega_\Gamma^{i,j}$, the penalization coefficient c_P should be selected so that the three terms have (roughly) the same size for the numerical solution (they will, of course, not vanish). In principle, c_P could be determined from knowledge of the fourth order derivatives of the solution, which control the error in the numerical solution. This approach does not appear to be practical. A simpler method is based on the observation that c_P should not depend on the grid size (at least to leading order, and we do not need better than this). Hence it can be determined empirically from a low resolution calculation. In the examples in this paper we found that $c_P \approx 50$ produced good results. □

Remark 16. A more general version of J_P in Eq. (17) would involve different penalization coefficients for the two line integrals, as well as the possibility of these coefficients having a dependence on the position along Γ of $\Omega_\Gamma^{i,j}$. These modifications could be useful in cases where the solution to the Poisson problem has large variations—*e.g.* a very irregular interface Γ, or a complicated forcing f. □

5.5. Computational cost

We can now infer something about the cost of the present scheme. To start with, let us denote the number of nodes in the x and y directions by

$$N_x = \frac{1}{h_x} + 1, \quad N_y = \frac{1}{h_y} + 1, \tag{18}$$

assuming a 1 by 1 computational square. Hence, the total number of degrees of freedom is $M = N_x N_y$. Furthermore, the number of nodes adjacent to the interface is $\mathcal{O}(M^{1/2})$, since the interface is a 1D entity.

The discretization of Poisson's equation results in a $M \times M$ linear system. Furthermore, the present method produces changes only on the RHS of the equations. Thus, the basic cost of inverting the system is unchanged from that of inverting the system resulting from a problem without an interface Γ. Namely: it varies from $\mathcal{O}(M)$ to $\mathcal{O}(M^2)$, depending on the solution method.

Let us now consider the computational cost added by the modifications to the RHS. As presented above, for each node adjacent to the interface, we must construct $\Omega_\Gamma^{i,j}$, compute the integrals that define the local 12×12 linear system, and

[3] The basis functions that we use for the bicubic interpolation can be found in Appendix A.

[4] The scaling also follows from dimensional consistency.

invert it. The cost associated with these tasks is constant: it does not vary from node to node, and it does not change with the size of the mesh. Consequently the resulting additional cost is a constant times the number of nodes adjacent to the interface. Hence it scales as $M^{1/2}$. Because of the (relatively large) coefficient of proportionality, for small M this additional cost can be comparable to the cost of inverting the Poisson problem. Obviously, this extra cost becomes less significant as M increases.

5.6. Interface representation

As far as the CFM is concerned, the framework needed to solve the local Cauchy problems is entirely described above. However, there is an important issue that deserves attention: the representation of the interface. This question is independent of the CFM. Many approaches are possible, and the optimal choice is geometry dependent. The discussion below is meant to shed some light on this issue, and motivate the solution we have adopted.

In the present work, generally we proceed assuming that the interface is not known exactly – since this is what frequently happens. The only exceptions to this are the examples in Sections 6.5 and 6.6, which involve two distinct (circular) interfaces touching at a point. In the generic setting, in addition to a proper representation of the interfaces, one needs to be able to identify the distinct interfaces, regions in between, contact points, as well as distinguish between a single interface crossing the same stencil multiple times and multiple distinct interfaces crossing one stencil. While the CFM algorithm is capable of dealing with these situations once they have been identified (*e.g.* see Remarks 11 and 12), the development of an algorithm with the capability to detect such generic geometries is beyond the scope of this paper, and a (hard) problem in interface representation. For these reasons, in the examples in Sections 6.5 and 6.6 we use an explicit exact representation of the interface.

To guarantee the accuracy of the solution for D, the interface conditions must be applied with the appropriate accuracy—see Section 5.7. Since these conditions are imposed on the interface Γ, the location of Γ must be known with the same order of accuracy desired for D. In the particular case of the 4th order implementation of the CFM algorithm in this paper, this means 4th order accuracy. For this reason, we adopted the Gradient-Augmented Level Set (GA-LS) approach, as introduced in [28]. This method allows a simple and completely local 4th order accurate representation of the interface, using Hermite cubics defined everywhere in the domain. This approach also allows the computation of normal vectors in a straightforward and accurate fashion.

We point out that the GA-LS method is not the only option for an implicit 4th order representation of the interface. For example, a regular level set method [27], combined with a high-order interpolation scheme, could be used as well. Here we adopted the GA-LS approach because of the algorithmic coherence that results from representing both the level set, and the correction functions, using the same bicubic polynomial base.

5.7. Error analysis

A naive reading of the discretized system in Eq. (14) suggests that, in order to obtain a fourth order accurate solution u, we need to compute the CFM correction terms $C_{i,j}$ with fourth order accuracy. Thus, from Eq. (15), it would follow that we need to know the correction function D with sixth order accuracy! This is, however, *not correct*, as explained below.

Since we need to compute the correction function D only at grid-points an $\mathcal{O}(h)$ distance away from Γ, it should be clear that errors in the $D_{i,j}$ are equivalent to errors in a and b of the same order. But errors in a and b produce errors of the same order in u—see Eq. (1) and Eq. (2). Hence, if we desire a fourth order accurate solution u, we need to compute the correction terms $D_{i,j}$ with fourth order accuracy only. This argument is confirmed by the convergence plots in Figs. 7, 9, and 11.

5.8. Computation of gradients

Some applications require not only the solution to the Poisson problem, but also its gradient. Hence, in Section 6, we include plots characterizing the behavior of the errors in the gradients of the solutions. A key question is then: how are these gradients computed?

To compute the gradients near the interface, the correction function can be used to extend the solution across the interface, so that a standard stencil can be used. However, for this to work, it is important to discretize the gradient operator using the same nodes that are part of the 9-point stencil—so that the same correction functions obtained while solving the Poisson equation can be used. Hence we discretize the gradient operator with a procedure similar to the one used to obtain the 9-point stencil. Specifically, we use the following 4th order accurate discretization:

$$\partial_x u_{i,j} = \hat{\partial}_x u_{i,j} + \frac{h_x^2}{6}[\hat{\partial}_{xx}\hat{\partial}_y u_{i,j} - (f_x)_{i,j}], \tag{19}$$

$$\partial_y u_{i,j} = \hat{\partial}_y u_{i,j} + \frac{h_y^2}{6}[\hat{\partial}_{yy}\hat{\partial}_x u_{i,j} - (f_y)_{i,j}], \tag{20}$$

where

$$\hat{\partial}_x u_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2h_x}, \tag{21}$$

$$\hat{\partial}_y u_{i,j} = \frac{u_{i,j+1} - u_{i,j-1}}{2h_y} \tag{22}$$

and $\hat{\partial}_{xx}$ and $\hat{\partial}_{yy}$ are defined by (12) and (13), respectively. The terms $(f_x)_{i,j}$ and $(f_y)_{i,j}$ may be given analytically (if known), or computed using appropriate second order accurate discretizations.

This discretization is 4th order accurate. However, since the error in the correction function is (generally) not smooth, the resulting gradient will be less than 4th order accurate (worse case scenario is 3rd order accurate) next to the interface.

6. Results

6.1. General comments

In this section we present five examples of computations in 2D using the algorithm introduced in Section 5. We solve the Poisson problem in the unit square $[0,1] \times [0,1]$ for five different configurations. Each example is defined below in terms of the problem parameters (source term f, and jump conditions across Γ), the representation of the interface(s)—either exact or using a level set function, and the exact solution (needed to evaluate the errors in the convergence plots). Notice that

1. As explained in Section 5.6, in examples 1 through 3 we represent the interface(s) using the GA-LS method. Hence, the interface is defined by a level set function ϕ, with gradient $\vec{\nabla}\phi = (\phi_x, \phi_y)$—both of which are carried within the GA-LS framework [28].
2. Below the level set is described via an analytic formula. In examples 1 through 3 this formula is converted into the GA-LS representation for the level set, before it is fed into the code. Only this representation is used for the actual computations. This is done so as to test the code's performance under generic conditions – where the interface Γ would be known via a level set representation only.
3. Within the GA-LS framework we can, easily and accurately, compute the vectors normal to the interface—anywhere in the domain. Hence, it is convenient to write the jump in the normal derivative, $[u_n]_\Gamma$, in terms of the jump in the gradient of u dotted with the normal to the interface $\hat{n} = (n_x, n_y)$.

The last two examples involve touching circular interfaces and were devised to demonstrate the robustness of the CFM in the presence of interfaces that are very close together. In these last two examples, for the reasons discussed in Section 5.6, we decided to use an exact representation of the circular interfaces.

6.2. Example 1

- Problem parameters:

$$f^+(x,y) = -2\pi^2 \sin(\pi x)\sin(\pi y),$$

$$f^-(x,y) = -2\pi^2 \sin(\pi x)\sin(\pi y),$$

$$[u]_\Gamma = \sin(\pi x)\exp(\pi y),$$

$$[u_n]_\Gamma = \pi[\cos(\pi x)\exp(\pi y)n_x + \sin(\pi x)\exp(\pi y)n_y].$$

- Level set defining the interface: $\phi(x,y) = (x - x_0)^2 + (y - y_0)^2 - r_0^2$, where $x_0 = 0.5$, $y_0 = 0.5$, and $r_0 = 0.1$.
- Exact solution:

$$u^+(x,y) = \sin(\pi x)\sin(\pi y),$$

$$u^-(x,y) = \sin(\pi x)[\sin(\pi y) - \exp(\pi y)].$$

Fig. 6 shows the numerical solution with a fine grid (193×193 nodes). The discontinuity is captured very sharply, and it causes no oscillations in the solution. In addition, Fig. 7 shows the behavior of the error of the solution and its gradient in the L_2 and L_∞ norms. As expected, the solution presents 4th order convergence as the grid is refined. Moreover, the gradient converges to 3rd order in the L_∞ norm and to 4th order in the L_2 norm, which is a reflection of the fact that the error in the solution is not smooth in a narrow region close to the interface only.

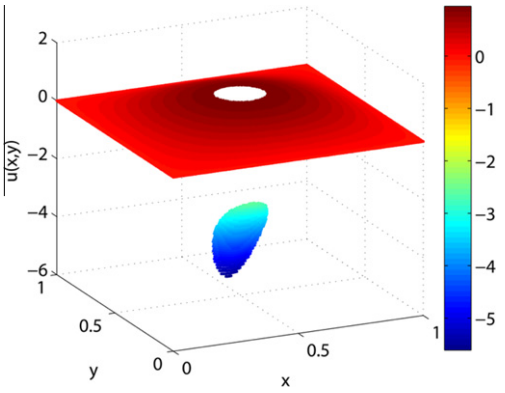
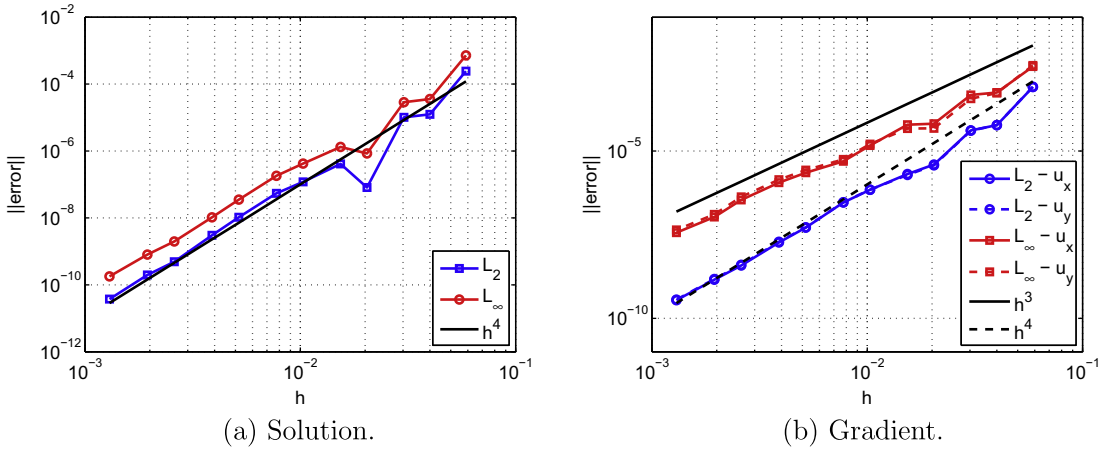


Fig. 6. Example 1 – numerical solution with 193 × 193 nodes.



(a) Solution. (b) Gradient.

Fig. 7. Example 1 – error behavior of the solution and its gradient in the L_2 and L_∞ norms.

6.3. Example 2

- Problem parameters:

$$f^+(x,y) = 0,$$

$$f^-(x,y) = 0,$$

$$[u]_\Gamma = -\exp(x)\cos(y),$$

$$[u_n]_\Gamma = -\exp(x)\cos(y)n_x + \exp(x)\sin(y)n_y.$$

- Level set defining the interface: $\phi(x,y) = (x-x_0)^2 + (y-x_0)^2 - r^2(\theta)$, where $r(\theta) = r_0 + \epsilon\sin(5\theta)$, $\theta(x,y) = \arctan(\frac{y-y_0}{x-x_0})$, $x_0 = 0.5$, $y_0 = 0.5$, $r = 0.25$, and $\epsilon = 0.05$.
- Exact solution:

$$u^+(x,y) = 0,$$

$$u^-(x,y) = \exp(x)\cos(y).$$

Fig. 8 shows the numerical solution with a fine grid (193 × 193 nodes). Once again, the overall quality of the solution is very satisfactory. Fig. 9 shows the behavior of the error of the solution and its gradient in the L_2 and L_∞ norms. Again, the solution converges to 4th order, while the gradient converges to 3rd order in the L_∞ norm and close to 4th order in the L_2 norm. However, unlike what happens in example 1, small wiggles are observed in the error plots. This behavior can be explained in terms of the construction of the sets $\Omega_\Gamma^{i,j}$—see Section 5. The approach used to construct $\Omega_\Gamma^{i,j}$ is highly dependent on the way in which the grid points are placed relative to the interface. Thus, as the grid is refined, the arrangement of

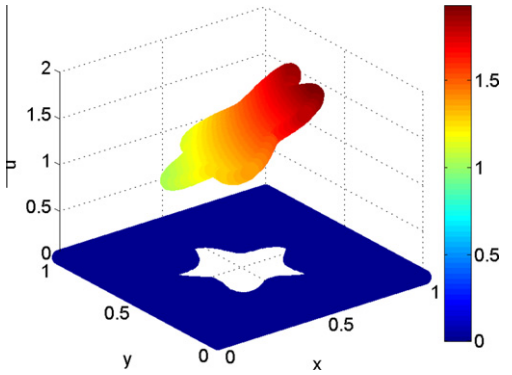


Fig. 8. Example 2 – numerical solution with 193 × 193 nodes.

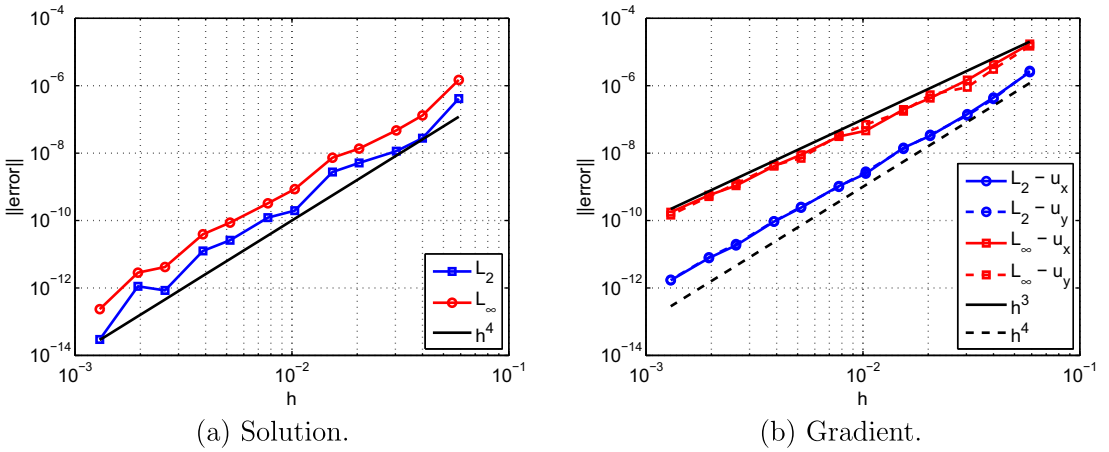


Fig. 9. Example 2 – error behavior of the solution and its gradient in the L_2 and L_∞ norms.

the Ω_Γ^{ij} can vary quite a lot—specially for a "complicated" interface such as the one in this example. What this means is that, while one can guarantee that the correction function D is obtained with 4th order precision, the proportionality coefficient is not constant—it may vary a little from grid to grid. This variation is responsible for the small oscillations observed in the convergence plot. Nevertheless, despite these oscillations, the overall convergence is clearly 4th order.

6.4. Example 3

- Problem parameters:

$$f^+(x,y) = \exp(x)[2 + y^2 + 2\sin(y) + 4x\sin(y)],$$

$$f^-(x,y) = 40,$$

$$[u]_\Gamma = \exp(x)[x^2\sin(y) + y^2] - 10(x^2 + y^2),$$

$$[u_n]_\Gamma = \{\exp(x)[(x^2 + 2x)\sin(y) + y^2] - 20x\}n_x + \{\exp(x)[x^2\cos(y) + 2y] - 20y\}n_y.$$

- Level set defining the interface:

$$\phi(x,y) = \left[(x - x_1)^2 + (y - y_1)^2 - r_1^2\right]\left[(x - x_2)^2 + (y - y_2)^2 - r_2^2\right],$$ where $x_1 = y_1 = 0.25$, $r_1 = 0.15$, $x_2 = y_2 = 0.75$, and $r_2 = 0.1$.

- Exact solution:

$$u^+(x,y) = \exp(x)[x^2\sin(y) + y^2],$$

$$u^-(x,y) = 10(x^2 + y^2).$$

Fig. 10 shows the numerical solution with a fine grid (193 × 193 nodes). In this example, there are two circular interfaces in the solution domain. The two regions inside the circles make Ω^-, while the remainder of the domain is Ω^+. This example shows that the method is general enough to deal with multiple interfaces, keeping the same quality in the solution. Fig. 11 shows that the solution converges to 4th order in both L_∞ and L_2 norms, while the gradient converges to 3rd order in the L_∞ norm and close to 4th order in the L_2 norm.

6.5. Example 4

- Problem parameters:

$$f_1(x,y) = -2\pi^2 \sin(\pi x)\sin(\pi y),$$

$$f_2(x,y) = \exp(x)[2 + y^2 + 2\sin(y) + 4x\sin(y)],$$

$$f_3(x,y) = -2\pi^2 \sin(\pi x)\sin(\pi y),$$

$$[u]_{\Gamma_{1-2}} = \exp(x)[x^2\sin(y) + y^2] - \sin(\pi x)\sin(\pi y) - 5,$$

$$[u_n]_{\Gamma_{1-2}} = \{\exp(x)[(x^2 + 2x)\sin(y) + y^2] - \pi\cos(\pi x)\sin(\pi y)\}n_x + \{\exp(x)[x^2\cos(y) + 2y] - \pi\sin(\pi x)\cos(\pi y)\}n_y,$$

$$[u]_{\Gamma_{2-3}} = \sin(\pi x)[\sin(\pi y) - \exp(\pi y)] - \exp(x)[x^2\sin(y) + y^2],$$

$$[u_n]_{\Gamma_{2-3}} = \{\pi\cos(\pi x)[\sin(\pi y) - \exp(\pi y)] - \exp(x)[(x^2 + 2x)\sin(y) + y^2]\}n_x + \{\pi\sin(\pi x)[\cos(\pi y) - \exp(\pi y)] - \exp(x) \times [x^2\cos(y) + 2y]\}n_y.$$

- Interface (exact representation):
  - Region 1: inside of the big circle.
  - Region 2: outer region.
  - Region 3: inside of the small circle.
  - Interface 1–2 (Big circle):

$$r_B = 0.3,$$

$$x_{0_B} = 0.5,$$

$$y_{0_B} = 0.5.$$

  - Interface 2–3 (Small circle):

$$r_S = 0.3,$$

$$x_{0_S} = x_{0_B} + r_B\cos(\pi/e^2) - r_S\cos(\pi(1/e^2 + 1)),$$

$$y_{0_S} = y_{0_B} + r_B\sin(\pi/e^2) - r_S\sin(\pi(1/e^2 + 1)).$$

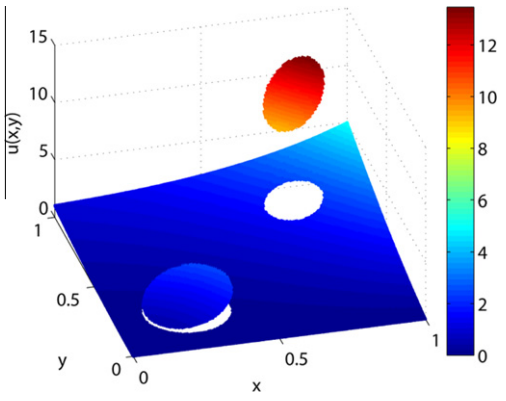
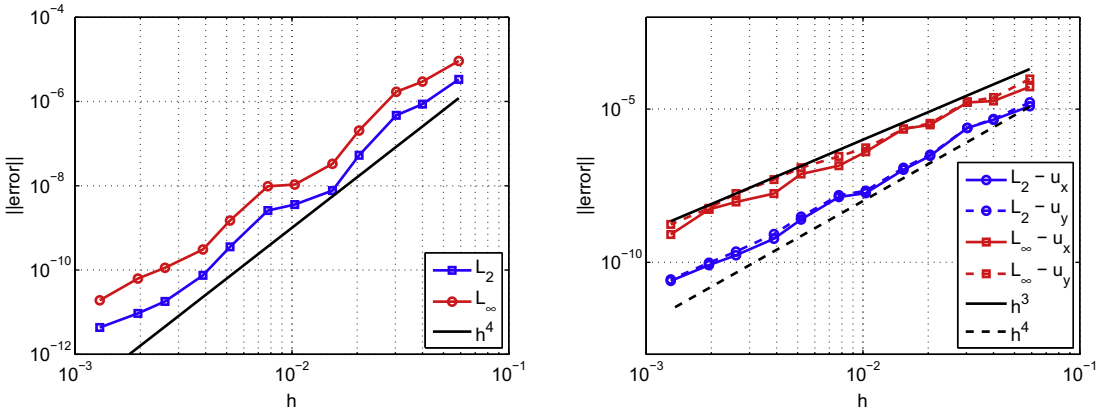


Fig. 10. Example 3 – numerical solution with 193 × 193 nodes.



(a) Solution. (b) Gradient.

Fig. 11. Example 3 – error behavior of the solution and its gradient in the L_2 and L_∞ norms.

- Exact solution:

$$u_1(x,y) = \sin(\pi x)\sin(\pi y) + 5,$$

$$u_2(x,y) = \exp(x)[x^2\sin(y) + y^2],$$

$$u_3(x,y) = \sin(\pi x)[\sin(\pi y) - \exp(\pi y)].$$

Fig. 12 shows the numerical solution with a fine grid (193 × 193 nodes). In this example the big circle is centered within the square integration domain and the small circle is external to it, with a common point of tangency. The point of contact is placed along the boundary of the big circle at the polar angle $\theta = \pi/e^2$—use the center of the big circle as the polar coordinates' origin. These choices guarantee that, as the grid is refined, a wide variety of configurations involving two distinct interfaces crossing the same stencil occurs in a neighborhood of the contact point. In particular, the selection of the angle θ is so that no special alignments of the grid with the local geometry near the contact point can happen. Fig. 13 shows the behavior of the error as $h \to 0$ in the L_2 and L_∞ norms. Once again we observe 4th order convergence (with small superimposed oscillations) for the solution. Moreover, the gradient converges to 3rd order in the L_∞ norm and close to 4th order in the L_2 norm. This example shows that the CFM is robust even in situations where distinct interfaces can get arbitrarily close (tangent at a point).

6.6. Example 5

- Problem parameters:

$$f_1(x,y) = -2\pi^2\sin(\pi x)\sin(\pi y),$$

$$f_2(x,y) = -2\pi^2\sin(\pi x)\sin(\pi y),$$

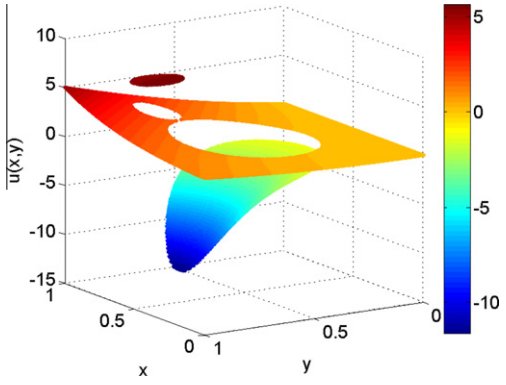
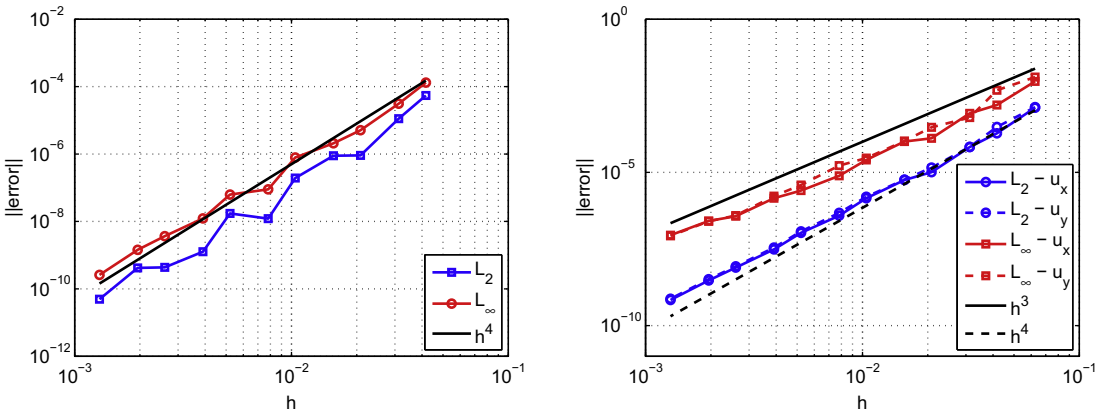


Fig. 12. Example 4 – numerical solution with 193 × 193 nodes.



(a) Solution. (b) Gradient.

Fig. 13. Example 4 – error behavior of the solution and its gradient in the L_2 and L_∞ norms.

$$f_3(x,y) = \exp(x)[2 + y^2 + 2\sin(y) + 4x\sin(y)],$$

$$[u]_{\Gamma_{1-2}} = -[\sin(\pi x)\exp(\pi y) + 5],$$

$$[u_n]_{\Gamma_{1-2}} = -\pi[\cos(\pi x)\exp(\pi y)n_x + \sin(\pi x)\exp(\pi y)n_y],$$

$$[u]_{\Gamma_{2-3}} = \exp(x)[x^2\sin(y) + y^2] - \sin(\pi x)[\sin(\pi y) - \exp(\pi y)],$$

$$[u_n]_{\Gamma_{2-3}} = \{\exp(x)[(x^2 + 2x)\sin(y) + y^2] - \pi\cos(\pi x)[\sin(\pi y) - \exp(\pi y)]\}n_x + \{\exp(x)[x^2\cos(y) + 2y] - \pi\sin(\pi x)[\cos(\pi y) - \exp(\pi y)]\}n_y.$$

- Interface (exact representation):
  - Region 1: inside small circle.
  - Region 2: region between circles.
  - Region 3: outer region.
  - Interface 2–3 (Big circle):

    $r_B = 0.3,$

    $x_{0_B} = 0.5,$

    $y_{0_B} = 0.5,$

  - Interface 1–2 (Small circle):

    $r_S = 0.3,$

    $x_{0_S} = x_{0_B} + (r_B - r_S)\cos(\pi/e^2),$

    $y_{0_S} = y_{0_B} + (r_B - r_S)\sin(\pi/e^2).$

- Exact solution:

$$u_1(x,y) = \sin(\pi x)\sin(\pi y) + 5,$$

$$u_2(x,y) = \sin(\pi x)[\sin(\pi y) - \exp(\pi y)],$$

$$u_3(x,y) = \exp(x)[x^2\sin(y) + y^2].$$

This example complements the example in Section 6.5, the sole difference being that here the small circle is inside the big circle. The results are entirely similar. Fig. 14 shows the numerical solution with a fine grid (193 × 193 nodes), while Fig. 15 shows the behavior of the error in the L_2 and L_∞ norms.

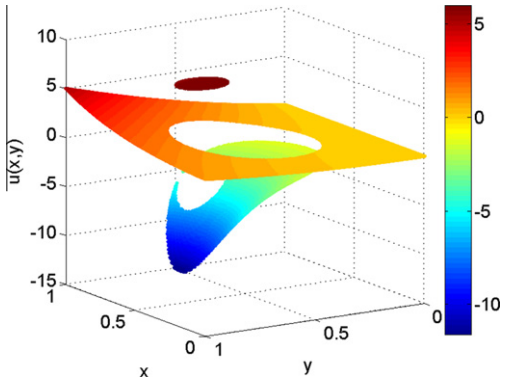


Fig. 14. Example 5 – numerical solution with 193 × 193 nodes.

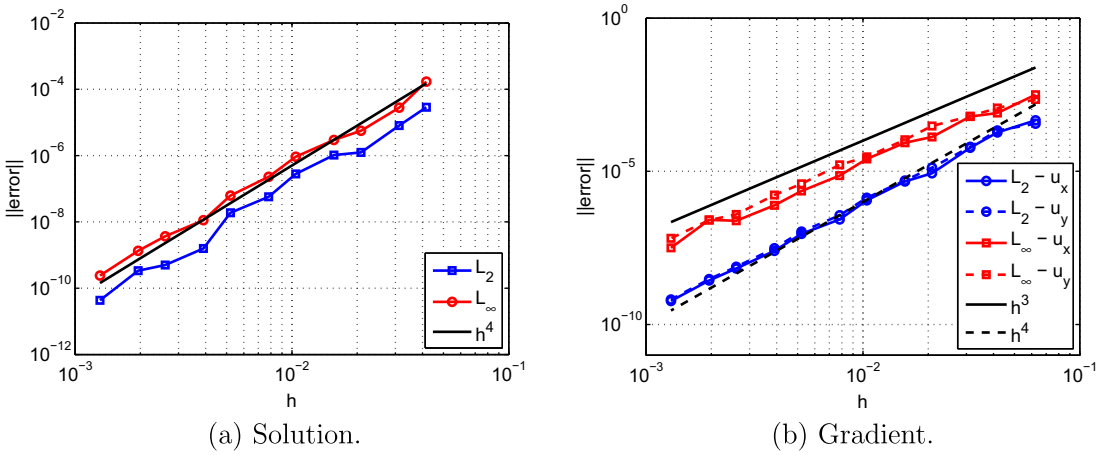


Fig. 15. Example 5 – error behavior of the solution and its gradient in the L_2 and L_∞ norms.

7. Conclusions

In this paper we have introduced the Correction Function Method (CFM), which can, in principle, be used to obtain (arbitrary) high-order accurate solutions to constant coefficients Poisson problems with interface jump conditions. This method is based on extending the correction terms idea of the Ghost Fluid Method (GFM) to that of a correction function, defined in a (narrow) band enclosing the interface. This function is the solution of a PDE problem, which can be solved (at least in principle) to any desired order of accuracy. Furthermore, like the GFM, the CFM allows the use of standard Poisson solvers. This feature follows from the fact that the interface jump conditions modify (via the correction function) only the right-hand-side of the discretized linear system of equations used in standard linear solvers for the Poisson equation.

As an example application, the new method was used to create a 4th order accurate scheme to solve the constant coefficients Poisson equation with interface jump conditions in 2D. In this scheme, the domain of definition of the correction function is split into many grid size rectangular patches. In each patch the function is represented in terms of a bicubic (with 12 free parameters), and the solution is obtained by minimizing an appropriate (discretized) quadratic functional. The correction function is thus pre-computed, and then is used to modify (in the standard way of the GFM) the right hand side of the Poisson linear system, incorporating the jump conditions into the Poisson solver. We used the standard 4th order accurate 9-point stencil discretization of the Laplace operator, to thus obtain a 4th order accurate method.

Examples were computed, showing the developed scheme to be robust, accurate, and able to capture discontinuities sharply, without creating spurious oscillations. Furthermore, the scheme is cost effective. First, because it allows the use of standard "black-box" Poisson solvers, which are normally tuned to be extremely efficient. Second, because the additional costs of solving for the correction function scale linearly with the mesh spacing, which means that they become relatively small for large systems.

Finally, we point out that the present method cannot be applied to all the interesting situations where a Poisson problem must be solved with jump discontinuities across an interface. Let us begin by displaying a very general Poisson problem with jump discontinuities at an interface. Specifically, consider

$$\vec{\nabla}\cdot(\beta^+(\vec{x})\vec{\nabla}u^+(\vec{x})) = f^+(\vec{x}) \quad \text{for } \vec{x}\in\Omega^+, \tag{23a}$$

$$\vec{\nabla}\cdot(\beta^-(\vec{x})\vec{\nabla}u^-(\vec{x})) = f^-(\vec{x}) \quad \text{for } \vec{x}\in\Omega^-, \tag{23b}$$

$$[\alpha u]_\Gamma = a(\vec{x}) \quad \text{for } \vec{x}\in\Gamma, \tag{23c}$$

$$[(\gamma u)_n]_\Gamma + [\eta u]_\Gamma = b(\vec{x}) \quad \text{for } \vec{x}\in\Gamma, \tag{23d}$$

$$u(\vec{x}) = g(\vec{x}) \quad \text{for } \vec{x}\in\partial\Omega, \tag{23e}$$

where we use the notation in Section 2 and

7.1 The brackets indicate jumps across the interface, for example:

$$[\alpha u]_\Gamma = (\alpha^+u^+)(\vec{x}) - (\alpha^-u^-)(\vec{x}) \quad \text{for } \vec{x}\in\Gamma.$$

7.2 The subscript n indicates the derivative in the direction of $\hat{n}$, the unit normal to the interface Γ pointing towards Ω^+.
7.3 $\beta^+ > 0$ and f^+ are smooth functions of $\vec{x}$, defined in the union of Ω^+ and some finite width band enclosing the interface Γ.
7.4 $\beta^- > 0$ and f^- are smooth functions of $\vec{x}$, defined in the union of Ω^- and some finite width band enclosing the interface Γ.
7.5 $\alpha^\pm > 0$, $\gamma^\pm > 0$, and $\eta^\pm$ are smooth functions, defined on some finite width band enclosing the interface Γ.
7.6 The Dirichlet boundary conditions in (23e) could be replaced any other standard set of boundary conditions on $\partial\,\Omega$.
7.7 As usual, the degree of smoothness of the various data functions involved determines how high an order an algorithm can be obtained.

Assume now that $\gamma^\pm = \alpha^\pm$, $\eta^\pm = c\alpha^\pm$—where c is a constant, and that

$$\left.\begin{aligned} \vec{A} &= \tfrac{1}{\beta^+}\vec{\nabla}\beta^+ = \tfrac{1}{\beta^-}\vec{\nabla}\beta^-, \\ B &= \tfrac{\alpha^+}{\beta^+}\vec{\nabla}\cdot\left(\beta^+\vec{\nabla}\left(\tfrac{1}{\alpha^+}\right)\right) = \tfrac{\alpha^-}{\beta^-}\vec{\nabla}\cdot\left(\beta^-\vec{\nabla}\left(\tfrac{1}{\alpha^-}\right)\right), \end{aligned}\right\} \tag{24}$$

applies in the band enclosing Γ where all the functions are defined.[5] In this case the methods introduced in this paper can be used to deal with the problem in (23) with minimal alterations. The main ideas carry through, as follows

7.a We assume that both u^+ and u^- can be extended across Γ, so that they are defined in some band enclosing the interface.
7.b We define the correction function, in the band enclosing Γ where the $\alpha^\pm$ and the $u^\pm$ exist, by $D = \alpha^+(\vec{x})u^+(\vec{x}) - \alpha^-(\vec{x})u^-(\vec{x})$.
7.c We notice that the correction function satisfies the elliptic Cauchy problem

$$\nabla^2 D + \vec{A}\cdot\vec{\nabla}D + BD = \frac{\alpha^+}{\beta^+}f^+ - \frac{\alpha^-}{\beta^-}f^-, \tag{25}$$

with $D = a$ and $D_n = b - c\,a$ at the interface Γ.
7.d We notice that the GFM correction terms can be written with knowledge of D.

Unfortunately, the conditions in (24) exclude some interesting physical phenomena. In particular, in two-phase flows the case where $\alpha^\pm$ and $\gamma^\pm$ are constants (but distinct) and $\eta^\pm = 0$ arises. We are currently investigating ways to circumvent these limitations, so as to extend our method to problems involving a wider range of physical phenomena.

Acknowledgements

The authors would like to acknowledge the National Science Foundation support—this research was partially supported by grant DMS-0813648. In addition, the first author acknowledges the support by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES – Brazil) and the Fulbright Commission through grant BEX 2784/06-8. The second author also acknowledges support from the NSERC Discovery program. Finally, the authors would also like to acknowledge many helpful

[5] Note that (24) implies that β^+ is a multiple of β^-.

conversations with Prof. B. Seibold at Temple University, and the many helpful suggestions made by the referees of this paper.

Appendix A. Bicubic interpolation

Bicubic interpolation is similar to bilinear interpolation, and can also be used to represent a function in a rectangular domain. However, whereas bilinear interpolation requires one piece of information per vertex of the rectangular domain, bicubic interpolation requires 4 pieces of information: function value, function gradient, and first mixed derivative (*i.e.* f_{xy}). For completeness, the relevant formulas for bicubic interpolation are presented below.

We use the classical multi-index notation, as in Ref. [28]. Thus, we represent the 4 vertices of the domain using the vector index $\vec{v} \in \{0,1\}^2$. Namely, the 4 vertices are $\vec{x}_{\vec{v}} = (x_1^0 + v_1 \Delta x_1, x_2^0 + v_2 \Delta x_2)$, where (x_1^0, x_2^0) are the coordinates of the left-bottom vertex and Δx_i is the length of the domain in the x_i direction. Furthermore, given a scalar function ϕ, the 4 pieces of information needed per vertex are given by

$$\phi_{\vec{\alpha}}^{\vec{v}} = \partial^{\vec{\alpha}} \phi(\vec{x}_{\vec{v}}), \tag{A.1}$$

where both $\vec{v},\ \vec{\alpha} \in \{0,1\}^2$ and

$$\partial^{\vec{\alpha}} = \partial_1^{\alpha_1} \partial_2^{\alpha_2}, \quad \partial_i^{\alpha_i} = (\Delta x_i)^{\alpha_i} \frac{\partial^{\alpha_i}}{\partial x_i^{\alpha_i}}. \tag{A.2}$$

Then the 16 polynomials that constitute the standard basis for the bicubic interpolation can be written in the compact form

$$W_{\vec{\alpha}}^{\vec{v}} = \prod_{i=1}^{2} w_{\alpha_i}^{v_i}(\bar{x}_i), \tag{A.3}$$

where $\bar{x}_i = \frac{x_i - x_i^0}{\Delta x_i}$, and w_α^v is the cubic polynomial

$$w_\alpha^v(x) = \begin{cases} f(x) & \text{for } v = 0 \quad \text{and } \alpha = 0, \\ f(1-x) & \text{for } v = 1 \quad \text{and } \alpha = 0, \\ g(x) & \text{for } v = 0 \quad \text{and } \alpha = 1, \\ -g(1-x) & \text{for } v = 1 \quad \text{and } \alpha = 1, \end{cases} \tag{A.4}$$

where $f(x) = 1 - 3x^2 + 2x^3$ and $g(x) = x(1-x)^2$.

Finally, the bicubic interpolation of a scalar function ϕ is given by the following linear combination of the basis functions:

$$\mathcal{H}(\vec{x}) = \sum_{\vec{v},\vec{\alpha} \in \{0,1\}^2} W_{\vec{\alpha}}^{\vec{v}} \phi_{\vec{\alpha}}^{\vec{v}} \tag{A.5}$$

As defined above (standard bicubic interpolation), 16 parameters are needed to determine the bicubic. However, in Ref. [28] a method (“cell-based approach”) is introduced, that reduces the number of degrees of freedom to 12, without compromising accuracy. This method uses information from the first derivatives to obtain approximate formulae for the mixed derivatives. In the present work, we adopt this cell-based approach.

Appendix B. Issues affecting the construction of Ω_Γ^{ij}

B.1. Overview

As discussed in Section 5, the CFM is based on local solutions to the PDE (9) in sub-regions of Ω_Γ – which we call Ω_Γ^{ij}. However, there is a certain degree of arbitrariness in how Ω_Γ^{ij} is defined. Here we discuss several factors that must be considered when solving Eq. (9), and how they influence the choice of Ω_Γ^{ij}. We also present four distinct approaches to construct Ω_Γ^{ij}, of increasing level of robustness (and, unfortunately, complexity).

The only requirements on Ω_Γ^{ij} that the discussion in Section 4 imposes are

- Ω_Γ^{ij} should be small, since the local problems' condition numbers increase exponentially with distance from Γ —see Remarks 5 and 6.
- Ω_Γ^{ij} should contain all the nodes where the correction function D is needed.

In addition, practical algorithmic considerations further restrict the definition of Ω_Γ^{ij}, as explained below.

First, we solve for D in a weak fashion, by locally minimizing a discrete version of the functional J_P defined in Eq. (17). This procedure involves integrations over Ω_Γ^{ij}. Thus, it is useful if Ω_Γ^{ij} has an elementary geometrical shape, so that simple quadrature rules can be applied to evaluate the integrals. Second, if Ω_Γ^{ij} is a rectangle, we can use a (high-order) bicubic (see

Appendix A) to represent D in $\Omega_\Gamma^{i,j}$. Hence we restrict $\Omega_\Gamma^{i,j}$ to be a rectangle.[6] Third, another consideration when constructing $\Omega_\Gamma^{i,j}$ is how the interface is represented. In principle the solution to the PDE in (9) depends only on the information given along the interface, and it is independent on the underlying grid. Nevertheless, consider an interface described implicitly by a level set function, known only in terms of its values (and perhaps derivatives) at the grid points. It is then convenient if the portions of the interface contained within $\Omega_\Gamma^{i,j}$ can be easily described in terms of the level set function discretization—*e.g.* in terms of a pre-determined set of grid cells, such as the cells that define the discretization stencil. The approaches in Sections B.2–B.4 are based on this premise.

Although the prior paragraph's strategy makes for an easier implementation, it also ties $\Omega_\Gamma^{i,j}$ to the underlying grid, whereas it should depend only on the interface geometry. Hence it results in definitions for $\Omega_\Gamma^{i,j}$ that cannot track the interface optimally. For this reason, we developed the approach in Section B.5, which allows $\Omega_\Gamma^{i,j}$ to freely adapt to the local interface geometry, regardless of the underlying grid. The idea is to first identify a piece of the interface based on a pre-determined set of grid cells, and then use this information to construct an optimal $\Omega_\Gamma^{i,j}$. This yields a somewhat intricate, but very robust definition for $\Omega_\Gamma^{i,j}$.

Finally, note that explicit representations of the interface are not constrained by the underlying grid. Moreover, information on the interface geometry is readily available anywhere along the interface. Hence, in this case, an optimal $\Omega_\Gamma^{i,j}$ can be constructed without the need to identify a piece of the interface in terms of a pre-determined set of grid cells. This fact makes the approach in Section B.5 straightforward with explicit representations of the interface. By contrast, the less robust approaches in Sections B.2–B.4 become more involved in this context, because they require the additional work of constraining the explicit representation to the underlying grid.

Obviously, the algorithms presented here (Sections B.2–B.5) represent only a few of the possible ways in which $\Omega_\Gamma^{i,j}$ can be defined. Nevertheless, these approaches serve as practical examples of how different factors must be balanced to design robust schemes.

B.2. Naive grid–aligned stencil–centered approach

In this approach, we fit $\Omega_\Gamma^{i,j}$ to the underlying grid by defining it as the $2h_x \times 2h_y$ box that covers the 9-point stencil. Fig. B.1 shows two examples.

This approach is very appealing because of its simplicity, but it has serious flaws and *we do not recommend it*. The reason is that the piece of the interface contained within $\Omega_\Gamma^{i,j}$ can become arbitrarily small – see Fig. B.1(b). Then the arguments that make the local Cauchy problem well posed no longer apply—see Remarks 5, 6, and 8. In essence, the biggest frequency encoded in the interface, $k_{\max} \approx 1/\text{length}(\Gamma/\Omega_\Gamma^{i,j})$, can become arbitrarily large—while the characteristic length of $\Omega_\Gamma^{i,j}$ remains $\mathcal{O}(h)$. As a consequence, the condition number for the local Cauchy problem can become arbitrarily large. We describe this approach here merely as an example of the problems that can arise from too simplistic a definition of $\Omega_\Gamma^{i,j}$.

B.3. Compact grid–aligned stencil–centered approach

This is the approach described in detail in Section 5.3. In summary: $\Omega_\Gamma^{i,j}$ is defined as the smallest rectangle that

(i) Is aligned with the grid.
(ii) Includes the piece of the interface contained within the stencil.
(iii) Includes all the nodes where D is needed.

Fig. B.2 shows three examples of this definition. As it should be clear from this figure, a key consequence of (i–iii) is that the piece of interface contained within $\Omega_\Gamma^{i,j}$ is always close to its diagonal—hence it is never too small relative to the size of $\Omega_\Gamma^{i,j}$. Consequently, this approach is considerably more robust than the one in Section B.2. In fact, we successfully employed it for all the examples using the 4th order accurate scheme—see Section 6 (Fig. B.3).

Unfortunately, the requirements (i–ii) in this approach tie $\Omega_\Gamma^{i,j}$ to the grid and the stencil. As mentioned earlier, these constraints may lead to an $\Omega_\Gamma^{i,j}$ which is not the best fit to the geometry of the interface. Fig. B.2(c) depicts a situation where this strategy yields relatively poor results. This happens when there is an almost perfect alignment of the interface with the grid, which can result in an excessively elongated $\Omega_\Gamma^{i,j}$—in the worse case scenario, this set could reduce to a line. Although the local Cauchy problem remains well conditioned, the elongated sets can interfere with the process we use to solve the equation for D in (9). Essentially, the representation of a function by a bicubic (or a modified bilinear in the case of the 2nd accurate scheme in Appendix C) becomes an ill-defined problem as the aspect ratio of the rectangle $\Omega_\Gamma^{i,j}$ vanishes (however, see the next paragraph). In the authors' experience, when this sort of alignment happens the solution remains valid and the errors are still relatively small—but the convergence rate may be affected if the bad alignment persists over several grid refinements.

We note that we observed the difficulties described in the paragraph above with the 2nd accurate scheme in Appendix C only. We attribute this to the fact that the bicubics result in a much better enforcement of the PDE (9) than the modified

[6] Clearly, other simple geometrical shapes, with other types approximations for D, should be possible—though we have not investigated them.

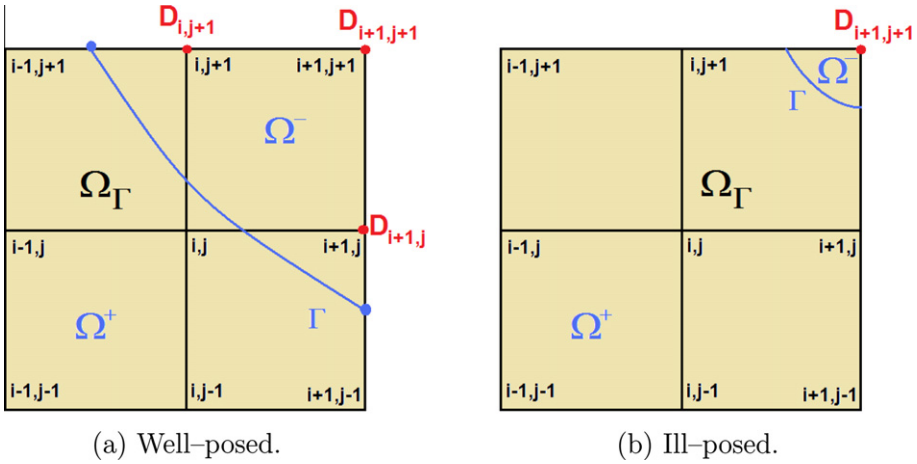


Fig. B.1. $\Omega_\Gamma^{i,j}$ as defined by the naive grid-aligned stencil-centered approach.

bilinears. The latter are mostly determined by the interface conditions—see Remark C.1. Hence, the PDE (9) provides a much stronger control over the bicubic parameters, making this interpolation more robust in elongated sets.

Note that this issue is much less severe than the problem affecting the approach in Section B.2, and could be corrected by making the Cauchy solver "smarter" when dealing with elongated sets. Instead we adopted the simpler solution of having a definition for $\Omega_\Gamma^{i,j}$ that avoids elongated sets. The most robust way to do this is to abandon the requirements in (i–ii), and allow $\Omega_\Gamma^{i,j}$ to adapt to the local geometry of the interface. This is the approach introduced in Section B.5. A simpler, compromise solution, is presented in Section B.4.

B.4. Free stencil–centered approach

Here we present a compromise solution for avoiding elongated $\Omega_\Gamma^{i,j}$, which abandons the requirement in (i), but not in (ii)—since (ii) is convenient when the interface is represented implicitly. In this approach $\Omega_\Gamma^{i,j}$ is defined as the smallest rectangle that

(i*) Is aligned with the grid rotated by an angle θ_r, where $\theta_r = \theta_\Gamma - \pi/4$ and θ_Γ characterizes the interface alignment with respect to the grid (*e.g.* the polar angle for the tangent vector to the interface section at its mid-point inside the stencil).
(ii*) Includes the piece of the interface contained within the stencil.
(iii*) Includes all the nodes where D is needed.

Fig. B.3 shows two examples of this approach.

The implementation of the present approach is very similar to that of the one in Section B.3. The only additional work is to compute θ_r and to write the interface and points where D is needed in the rotated frame of reference. In both these approaches the diagonal of $\Omega_\Gamma^{i,j}$ is very close to the piece of interface contained within the stencil, which guarantees a well conditioned local problem. However, here the addition of a rotation keeps $\Omega_\Gamma^{i,j}$ nearly square, and avoids elongated geometries. The price paid for this is that the sets $\Omega_\Gamma^{i,j}$ created using Section B.4 can be a little larger than the ones from Section B.3—with both sets including the exact same piece of interface. In such situations, the present approach results in a somewhat larger condition number.

B.5. Node-centered approach

Here we define $\Omega_\Gamma^{i,j}$ in a fashion that is completely independent from the underlying grid and stencils. In fact, instead of associating each $\Omega_\Gamma^{i,j}$ to a particular stencil, we define a different $\Omega_\Gamma^{i,j}$ for each node where the correction is needed—hence the name node-centered, rather than stencil-centered. As a consequence, whereas the prior strategies lead to multiple values of D at the same node (one value per stencil, see Remark 10), here there is a unique value of D at each node.

In this approach, $\Omega_\Gamma^{i,j}$ is defined by the following steps:

1. Identify the interface in the 4 grid cells that surround a given node. This step can be skipped if the interface is represented explicitly.
2. Find the point, P_0, along the interface that is closest to the node. This point becomes the center of $\Omega_\Gamma^{i,j}$. There is no need to obtain P_0 very accurately. Small errors in P_0 result in small shifts in $\Omega_\Gamma^{i,j}$ only.

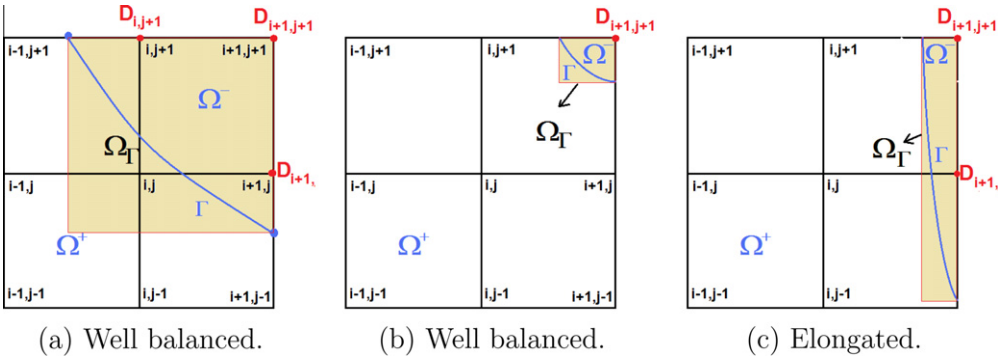


Fig. B.2. $\Omega_\Gamma^{i,j}$ as defined by the compact grid-aligned stencil-centered approach.

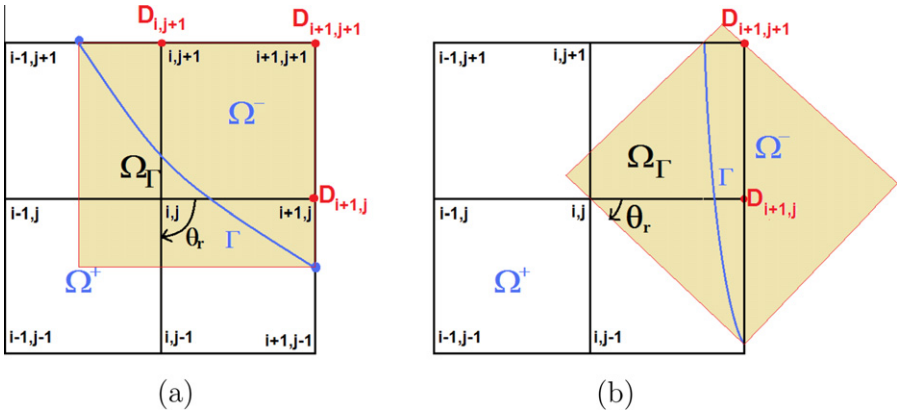


Fig. B.3. $\Omega_\Gamma^{i,j}$ as defined by the free stencil-centered approach.

3. Compute $\hat{t}_0$, the vector tangent to the interface at P_0. This vector defines one of the diagonals of $\Omega_\Gamma^{i,j}$. The normal vector $\hat{n}_0$ defines the other diagonal. Again, high accuracy is not needed.
4. Then $\Omega_\Gamma^{i,j}$ is the square with side length $2\sqrt{h_x^2 + h_y^2}$, centered at P_0, and diagonals parallel to $\hat{t}_0$ and $\hat{n}_0$—$\Omega_\Gamma^{i,j}$ need not be aligned with the grid.

Fig. B.4 shows two examples of this approach.

Note that the piece of interface contained within $\Omega_\Gamma^{i,j}$, as defined by steps 1–4 above, is not necessarily the same found in step 1. Hence, after defining $\Omega_\Gamma^{i,j}$, we still need to identify the piece of interface that lies within it. For an explicit representation of the interface, this additional step is not particularly costly, but the same is not true for an implicit representation.

This approach is very robust because it always creates a square $\Omega_\Gamma^{i,j}$, with the interface within it close to one of the diagonals—as guaranteed by steps 2 and 3. Hence, the local Cauchy problem is always well conditioned. Furthermore, making $\Omega_\Gamma^{i,j}$ square (to avoid elongation) does not result in larger condition numbers as in Section B.4 because the larger $\Omega_\Gamma^{i,j}$ contain an equally larger piece of the interface.

Finally, we point out that the (small) oscillations observed in the convergence plots shown in Section 6 occur because in these calculations we use the approach in Section B.3—which produces sets $\Omega_\Gamma^{i,j}$ that are not uniform in size, nor shape, along the interface. Tests we have done show that these oscillations do not occur with the node-centered approach here, for which all the $\Omega_\Gamma^{i,j}$ are squares of the same size. Unfortunately, as pointed out earlier, the node-centered approach is not well suited for calculations using an interface represented implicitly.

Appendix C. 2nd order accurate scheme in 2D

C.1. Overview

In Section 5 we present a 4th order accurate scheme to solve the 2D Poisson problem with interface jump conditions, based on the correction function defined in Section 4. However, there are many situations in which a 2nd order version

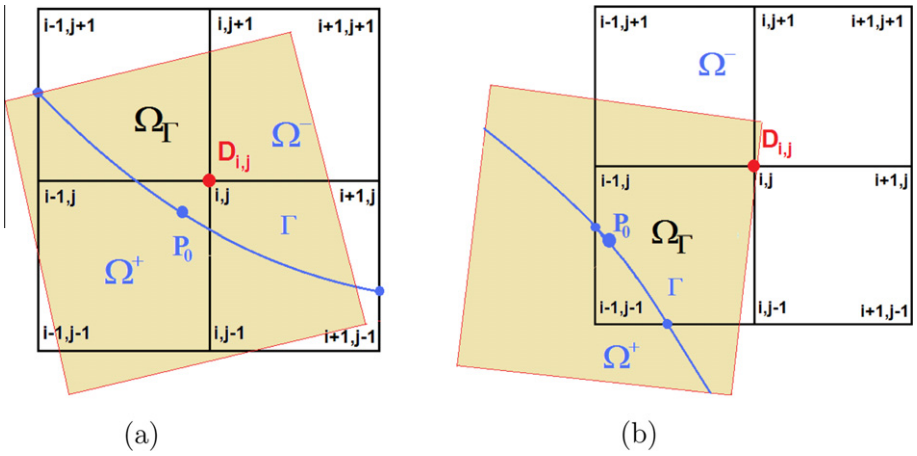


Fig. B.4. $\Omega_\Gamma^{i,j}$ as defined by the node-centered approach.

of the method could be of practical relevance. Hence, in this section we use the general framework provided by the correction function method (CFM) to develop a specific example of a 2nd order accurate scheme in 2D. The basic approach is analogous to the 4th version presented in Section 5. A few key points are

(a) We discretize Poisson's equation using the standard 5-point stencil. This stencil is compact, which is an important requirement for a well conditioned problem for the correction function D—see Remarks 4–6.
(b) We approximate D using modified bilinear interpolants (defined in Section C.2), each valid in a small neighborhood $\Omega_\Gamma^{i,j}$ of the interface. This guarantees local 2nd order accuracy with only 5 interpolation parameters. Each $\Omega_\Gamma^{i,j}$ corresponds to a grid point at which the standard discretization of Poisson's equation involves a stencil that straddles the interface Γ.
(c) The domains $\Omega_\Gamma^{i,j}$ are rectangular regions, each enclosing a portion of Γ, and all the nodes where D is needed to complete the discretization of Poisson's equation at the (i,j)th stencil. Each is a sub-domain of Ω_Γ.
(d) Starting from (b) and (c), we design a local solver that provides an approximation to D inside each domain $\Omega_\Gamma^{i,j}$.
(e) The interface Γ is represented using the standard level set approach—see [27]. This guarantees a local 2nd order representation of the interface, as required to keep the overall accuracy of the scheme.
(f) In each $\Omega_\Gamma^{i,j}$, we solve the PDE in (9) in a least squares sense—see Remark 9. Namely, we seek the minimum of the positive quadratic integral quantity J_P in (17), which vanishes at the solution: We substitute the modified bilinear approximation for D into J_P, discretize the integrals using Gaussian quadratures, and minimize the resulting discrete J_P.

Remark C.1. Using standard bilinear interpolants to approximate D in each $\Omega_\Gamma^{i,j}$ also yields 2nd order accuracy. However, the Laplacian of a standard bilinear interpolant vanishes. Thus, this basis cannot take full advantage of the fact that D is the solution to the Cauchy problem in (9). The modified bilinears, on the other hand, incorporate the average of the Laplacian into the formulation—see Section C.2. In Section C.6 we include results with both the standard and the modified bilinears, to demonstrate the advantages of the latter. □

Below, in Sections C.2 to C.5 we describe the 2nd order accurate scheme, and in Section C.6 we present applications of the scheme to three test cases.

C.2. Modified bilinear

The modified bilinear interpolant used here builds on the standard bilinear polynomials. To define them, we use the multi-index notation—see Appendix A and Ref. [28]. Thus, we label the 4 vertices of a rectangular cell using the vector index $\vec{v} \in \{0,1\}^2$. Then the vertices are $\vec{x}_{\vec{v}} = (x_1^0 + v_1\Delta x_1, x_2^0 + v_2\Delta x_2)$, where (x_1^0, x_2^0) are the coordinates of the left-bottom vertex and Δx_i is the length of the domain in the x_i direction. The standard bilinear interpolation basis is then given by the 4 polynomials

$$W^{\vec{v}} = \prod_{i=1}^{2} w^{v_i}(\bar{x}_i), \tag{C.1}$$

where $\bar{x}_i = \frac{x_i - x_i^0}{\Delta x_i}$, and w^v is the linear polynomial

$$w^v(x) = \begin{cases} 1-x & \text{for } v = 0, \\ x & \text{for } v = 1. \end{cases} \tag{C.2}$$

The standard bilinear interpolation of a scalar function ϕ is given (in each cell) by

$$\mathcal{H}_s(\vec{x}) = \sum_{\vec{v}\in\{0,1\}^2} W^{\vec{v}}\phi(\vec{x}_{\vec{v}}). \tag{C.3}$$

In the modified version, we add a quadratic term proportional to $x^2 + y^2$, so that the Laplacian of the modified bilinear is no longer identically zero. The coefficient of the quadratic term can be written in terms of the average value of the Laplacian over the domain, $\overline{\nabla^2\phi}$. This yields the following formula for the modified bilinear interpolant

$$\mathcal{H}_m(\vec{x}) = \mathcal{H}_s(\vec{x}) - \frac{1}{4}[w^0(\bar{x})w^1(\bar{x})(\Delta x)^2 + w^0(\bar{y})w^1(\bar{y})(\Delta y)^2]\overline{\nabla^2\phi}. \tag{C.4}$$

C.3. Standard Stencil

We use the standard 2nd order accurate 5-point discretization of the Poisson equation

$$L^5 u_{i,j} = f_{i,j}, \tag{C.5}$$

where L^5 is defined in (11). In the absence of discontinuities, (C.5) provides a compact 2nd order accurate representation of the Poisson equation. In the vicinity of the discontinuities at the interface Γ, we define an appropriate domain $\Omega_\Gamma^{i,j}$, and use it to compute the correction terms needed by (C.5)—as described below.

To understand how the correction terms affect the discretization, consider the situation in Fig. C.1. In this case, the node (i,j) lies in Ω^+ while the nodes $(i+1,j)$ and $(i,j+1)$ are in Ω^-. Hence, to be able to use Eq. (C.5), we need to compute $D_{i+1,j}$ and $D_{i,j+1}$.

After having solved for D where necessary (see Section C.4 and Section C.5), we modify Eq. (C.5) and write

$$L^5 u_{i,j} = f_{i,j} + C_{i,j}, \tag{C.6}$$

which differs from (C.5) by the terms $C_{i,j}$ on the RHS only. Here the $C_{i,j}$ are the CFM correction terms needed to complete the stencil across the discontinuity at Γ. In the particular case illustrated by Fig. C.1, we have

$$C_{i,j} = -\frac{1}{h_x^2}D_{i+1,j} - \frac{1}{h_y^2}D_{i,j+1}. \tag{C.7}$$

Similar formulas apply for all the other possible arrangements of the stencil for the Poisson's equation, relative to the interface Γ.

C.4. Definition of $\Omega_\Gamma^{i,j}$

As discussed in Appendix B, the construction of $\Omega_\Gamma^{i,j}$ presented in Section 5.3 may lead to elongated shapes for $\Omega_\Gamma^{i,j}$, which can cause accuracy loses. As mentioned earlier (see the end of Section B.3) this is not a problem for the 4th order scheme, but it can be one for the 2nd order one. To resolve this issue we could have implemented the robust construction of $\Omega_\Gamma^{i,j}$ given in Section B.5. However the simpler compromise version in Section B.4 proved sufficient.

The approach in Section B.4 requires $\Omega_\Gamma^{i,j}$ to include the piece of interface contained "within the stencil." For the 9-point stencil, this naturally means "within the $2h_x \times 2h_y$ box aligned with the grid that includes the nine points of the stencil." We could use the same meaning for the 5-point stencil, but this would not take full advantage of the 5-point stencil compactness. A better choice is to use the quadrilateral defined by the stencil's four extreme points—*i.e.* the dashed box in Fig. C.1.

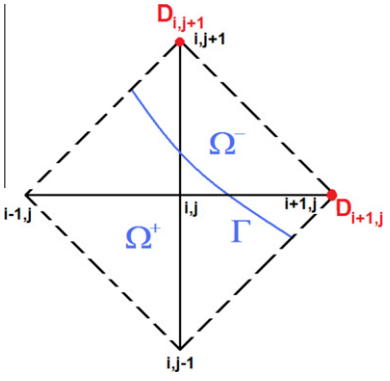


Fig. C.1. The 5-point stencil next to the interface Γ. The dashed box shows a compact quadrangular region that contains the stencil.

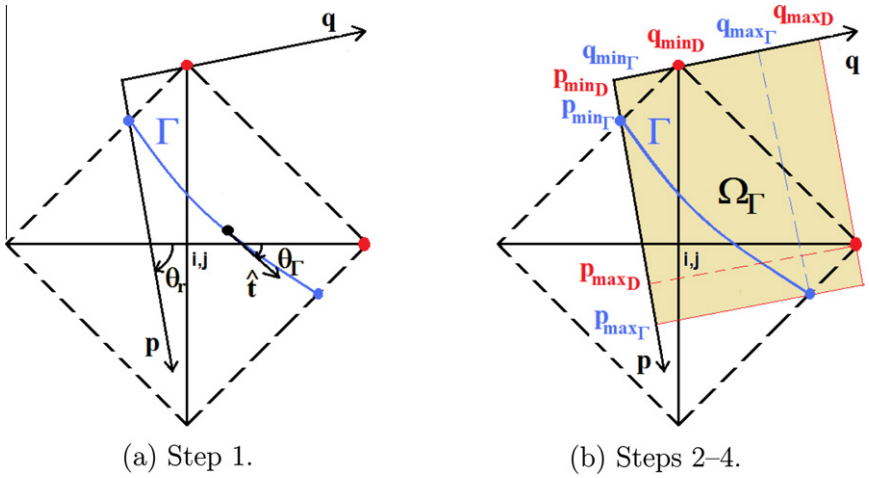


Fig. C.2. The set $\Omega_\Gamma^{i,j}$ for the situation in Fig. C.1.

This choice is used only to determine the piece of interface to be included within $\Omega_\Gamma^{i,j}$. It does not affect the definition of the interface alignment angle θ_L used by the Section B.4 approach. Hence the following four steps define $\Omega_\Gamma^{i,j}$.

1. Find the angle θ_Γ between the vector tangent to the interface at the mid-point within the stencil, and the x-axis. Introduce the coordinate system $p-q$, resulting from rotating $x-y$ by $\theta_r = \theta_\Gamma - \pi/4$—see Fig. C.2(a).
2. Find the coordinates $(p_{\min_\Gamma}, p_{\max_\Gamma})$ and $(q_{\min_\Gamma}, q_{\max_\Gamma})$ characterizing the smallest p-q coordinate rectangle enclosing the section of the interface contained within the stencil—see Fig. C.2(b).
3. Find the coordinates $(p_{\min_D}, p_{\max_D})$ and $(q_{\min_D}, q_{\max_D})$ characterizing the smallest p-q coordinate rectangle enclosing all the nodes at which D is needed—see Fig. C.2(b).
4. $\Omega_\Gamma^{i,j}$ is the smallest $p-q$ coordinate rectangle enclosing the two previous rectangles. Its edges are characterized by

$$p_{\min} = \min(p_{\min_\Gamma}, p_{\min_D}), \tag{C.8a}$$
$$p_{\max} = \max(p_{\max_\Gamma}, p_{\max_D}), \tag{C.8b}$$
$$q_{\min} = \min(q_{\min_\Gamma}, q_{\min_D}), \tag{C.8c}$$
$$q_{\max} = \max(q_{\max_\Gamma}, q_{\max_D}). \tag{C.8d}$$

Fig. C.2 shows an example of $\Omega_\Gamma^{i,j}$ defined in this way.

C.5. Solution of the local Cauchy problem

The remainder of the 2nd order accurate scheme follows the exact same procedure used for the 4th order accurate version described in detail in Section 5. In particular, as explained in item (f) of Section C.1, we solve the local Cauchy problem defined by Eq. (9) in a least squares sense (using the same minimization procedure described in Section 5.4). The only differences are that: (i) In the 2nd order accurate version of the method we use 4 Gaussian quadrature points for the 1D line integrals, and 16 points for the 2D area integrals. (ii) Because the modified bilinear representation for D involves 5 basis polynomials, the minimization problem produces a 5×5 (self-adjoint) linear system—instead of the 12×12 system that occurs for the 4th order algorithm.

C.6. Results

Here we present three examples of solutions to the 2D Poisson's equation using the algorithm described above. Each example is defined below in terms of the problem parameters (source term f and jump conditions across the interface Γ), the representation of the interface, and the exact solution (needed to evaluate the errors in the convergence plots). Note that

1. In all cases we represent the interface using a level set function ϕ.
2. Below the level set function is described via an analytic formula. This formula is converted into a discrete level set representation used by the code. Only this discrete representation is used for the actual computations.

3. The level set formulation allows us to compute the vectors normal to the interface to 2nd order accuracy with a combination of finite differences and standard bilinear interpolation. Hence, it is convenient to write the jump in the normal derivative, $[u_n]_\Gamma$, in terms of the jump in the gradient of u dotted with the normal to the interface $\hat{n} = (n_x, n_y)$.

We use[7] example 1_s to test the 2nd order scheme in a generic problem with a non-trivial interface geometry. In addition, in this example the correction function D has a non-zero Laplacian. Thus, we used this problem to compare the performance of the Standard Bilinear (SB) and Modified Bilinear (MB) interpolations as basis for the correction function. Finally, examples 2_s and 3_s here correspond to examples 1 and 3 in [3], respectively. Hence they can be used to compare the performance of the present 2nd order scheme with the Immersed Interface Method (IIM), in two distinct situations.

C.6.1. Example 1_s

- Domain: $(x,y) \in [0,1] \times [0,1]$.
- Problem parameters:

$$f^+(x,y) = 4,$$

$$f^-(x,y) = 0,$$

$$[u]_\Gamma = x^2 + y^2 - \exp(x)\cos(y),$$

$$[u_n]_\Gamma = [2x - \exp(x)\cos(y)]n_x + [2y + \exp(x)\sin(y)]n_y.$$

- Level set defining the interface: $\phi(x,y) = \sqrt{(x-x_0)+(y-x_0)} - r(\theta)$, where $r(\theta) = r_0 + \epsilon\sin(5\theta)$, $\theta(x,y) = \arctan\left(\frac{y-y_0}{x-x_0}\right)$, $x_0 = 0.5$, $y_0 = 0.5$, $r = 0.25$, and $\epsilon = 0.05$.
- Exact solution:

$$u^+(x,y) = x^2 + y^2,$$

$$u^-(x,y) = \exp(x)\cos(y).$$

Fig. C.3 shows the numerical solution with a fine grid (193×193 nodes). The non-trivial contour of the interface is accurately represented and the discontinuity is captured very sharply. For comparison we solved this problem using both the SB and MB interpolations to represent the correction function. Although Fig. C.3 shows the solution obtained with the modified bilinear version, both versions produce small errors and are visually indistinguishable.

Fig. C.4(a) shows the convergence of the solution error in the L_2 and L_∞ norms, for both the SB and MB versions. As expected, the overall behavior indicates 2nd order convergence, despite the small oscillations that are characteristic of this implementation of the method, as explained in Section 6.3. Note that the MB version produces significantly smaller errors (a factor of about 20) than the SB version. It also exhibits a more robustly 2nd order convergence rate. Moreover, Fig. C.4(b) shows the convergence of the error of the gradient of the MB solution in the L_2 and L_∞ norms. Here, the gradient was computed using standard 2nd order accurate centered differences, as in (21) and (22). As we can observe, the gradient converges to 1st order in the L_∞ norm and, apparently, as $h^{3/2}$ in the L_2 norm.

C.6.2. Example 2_s

- Domain: $(x,y) \in [-1,1] \times [-1,1]$.
- Problem parameters:

$$f^+(x,y) = 0,$$

$$f^-(x,y) = 0,$$

$$[u]_\Gamma = \log(2\sqrt{x^2+y^2}),$$

$$[u_n]_\Gamma = \frac{xn_x + yn_y}{x^2+y^2}.$$

- Level set defining the interface: $\phi(x,y) = \sqrt{(x-x_0)^2 + (y-y_0)^2} - r_0$, where $x_0 = 0$, $y_0 = 0$, and $r_0 = 0.5$.

[7] A subscript s is added to the example numbers of the 2nd order method, to avoid confusion with the 4th order method examples.

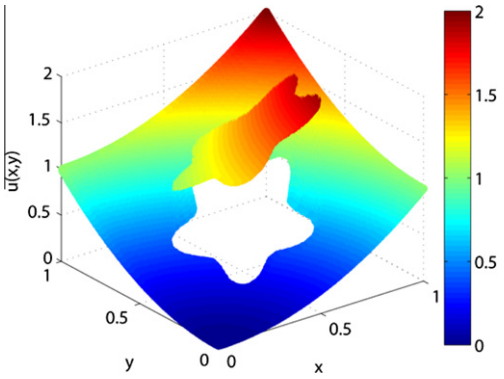
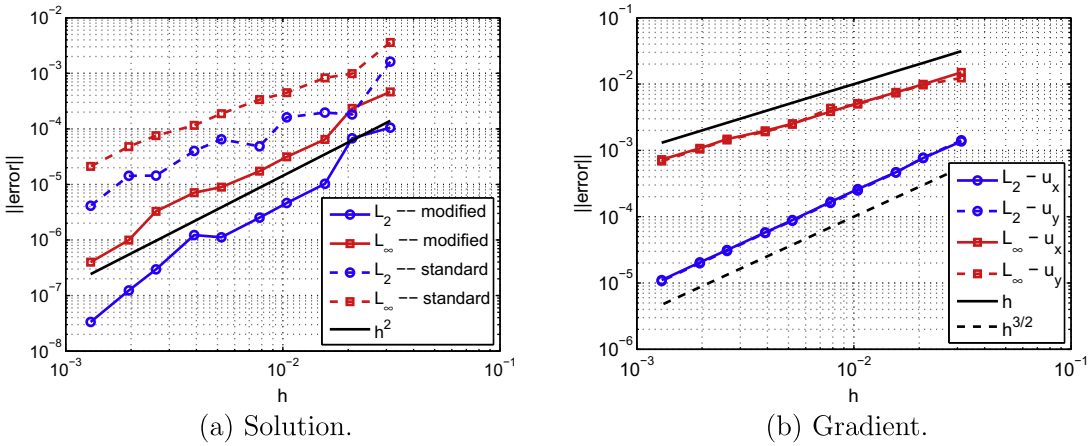


Fig. C.3. Example 1_s. Numerical solution with 193×193 nodes, 2^{nd} order scheme, modified bilinear version.



(a) Solution. (b) Gradient.

Fig. C.4. Example 1_s – error behavior of the solution and its gradient in the L_2 and L_∞ norms.

- Exact solution:

$$u^+(x,y) = 1 + \log(2\sqrt{x^2+y^2}),$$

$$u^-(x,y) = 1.$$

As mentioned earlier, this example corresponds to example 1 in [3], where the same problem is solved using the Immersed Interface Method. Hence, this provides a good opportunity to compare the CFM with the well established IIM. Fig. C.5 shows the numerical solution with a fine grid (161×161 nodes). Once again, the overall quality of the solution is

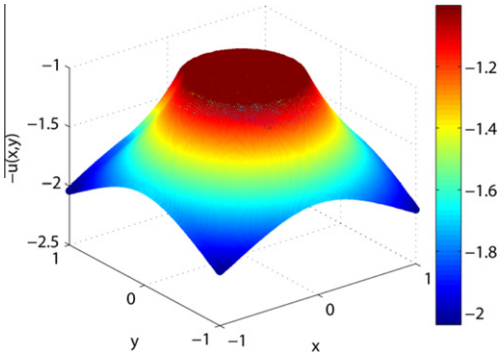
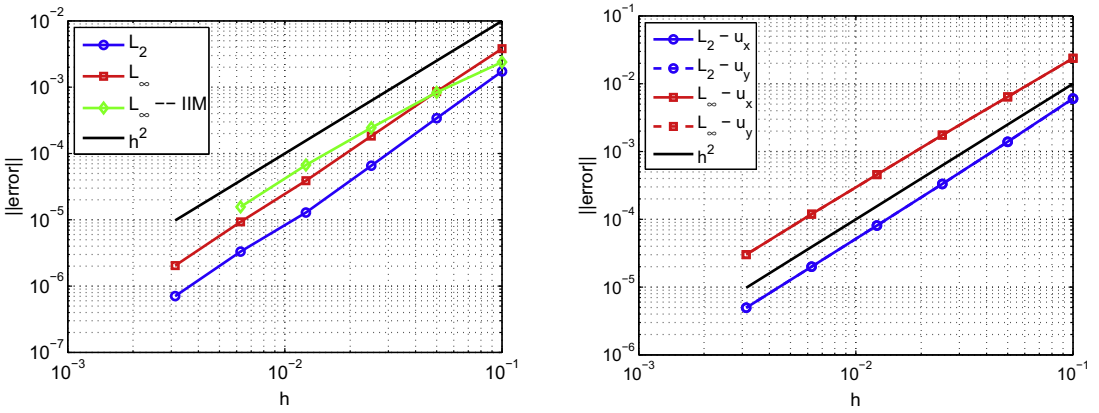


Fig. C.5. Example 2_s. Numerical solution with 161×161 nodes.



(a) Solution. (b) Gradient.

Fig. C.6. Example 2_s – error behavior of the solution and its gradient in the L_2 and L_∞ norms. IIM refers to results obtained with the Immersed Interface Method in [3] (Copyright ©1994 Society for Industrial and Applied Mathematics. Data compiled with permission. All rights reserved).

very satisfactory. Fig. C.6 shows the behavior of the error in the L_2 and L_∞ norms and the solution converges to 2^{nd} order. However, in this example the gradient also appears to converge to 2^{nd} order.

Fig. C.6(a) also includes the convergence of the solution error in the L_∞ norm obtained with the IIM—we plot the errors listed in table 1 of [3]. Both methods produce similar convergence rates. However, in this example at least, the CFM produces slightly smaller errors—by a factor of about 1.7.

C.6.3. Example 3_s

- Domain: $(x,y) \in [-1,1] \times [-1,1]$.
- Problem parameters:

$$f^+(x,y) = 0,$$

$$f^-(x,y) = 0,$$

$$[u]_\Gamma = -\exp(x)\cos(y),$$

$$[u_n]_\Gamma = \exp(x)[-\cos(y)n_x + \sin(y)n_y].$$

- Level set defining the interface: $\phi(x,y) = \sqrt{(x-x_0)^2 + (y-y_0)^2} - r_0$, where $x_0 = 0$, $y_0 = 0$, and $r_0 = 0.5$.
- Exact solution:

$$u^+(x,y) = 0,$$

$$u^-(x,y) = \exp(x)\cos(y).$$

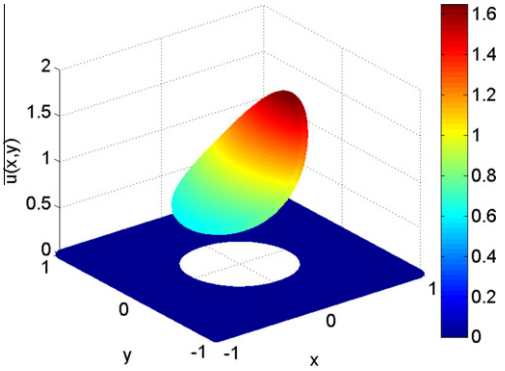
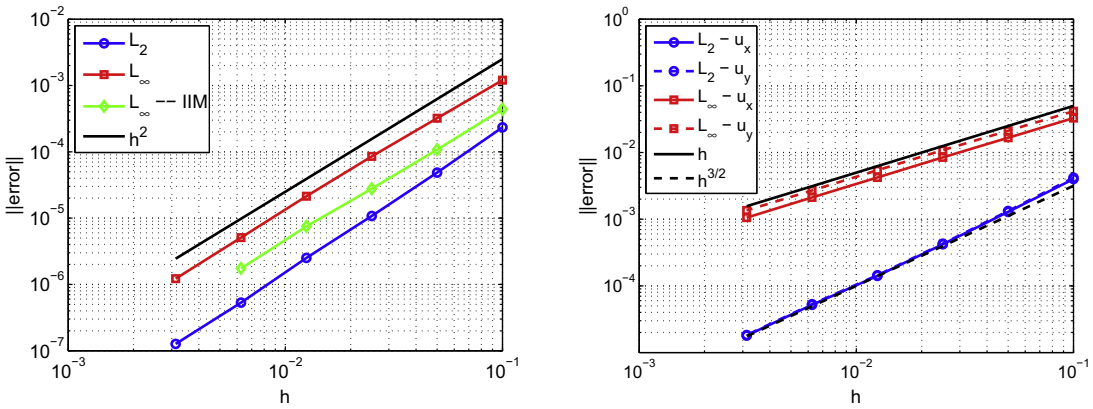


Fig. C.7. Example 3_s. Numerical solution with 161×161 nodes.



(a) Solution. (b) Gradient.

Fig. C.8. Example 3_s – error behavior of the solution and its gradient in the L_2 and L_∞ norms. IIM refers to results obtained with the Immersed Interface Method in [3] (Copyright ©1994 Society for Industrial and Applied Mathematics. Data compiled with permission. All rights reserved).

This example corresponds to example 3 in [3] for the IIM method. Fig. C.7 shows the numerical solution with a fine grid (161×161 nodes) while Fig. C.8 presents convergence results for the errors in the L_2 and L_∞ norms. In addition, Fig. C.8(a) also includes the L_∞ norm of the error obtained with the IIM—we plot the errors listed in table 3 of [3]. In this case the IIM produces slightly smaller errors than the CFM—by a factor of about 2.8. Therefore, based on the results from examples 2_s and 3_s, we may conclude that the IIM and the CFM produce results of comparable accuracy—each method generating slightly smaller errors in different cases.

References

[1] C.S. Peskin, Numerical analysis of blood flow in the heart, Journal of Computational Physics 25 (3) (1977) 220–252, doi:10.1016/0021-9991(77)90100-0.
[2] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, Journal of Computational Physics 114 (1) (1994) 146–159, doi:10.1006/jcph.1994.1155.
[3] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM Journal on Numerical Analysis 31 (4) (1994) 1019–1044, doi:10.1137/0731054.
[4] R.J. LeVeque, Z. Li, Immersed interface methods for Stokes flow with elastic boundaries or surface tension, SIAM Journal on Scientific Computing 18 (3) (1997) 709–735, doi:10.1137/S1064827595282532.
[5] H. Johansen, P. Colella, A Cartesian grid embedded boundary method for Poisson's equation on irregular domains, Journal of Computational Physics 147 (1) (1998) 60–85, doi:10.1006/jcph.1998.5965.
[6] R.P. Fedkiw, T. Aslam, S. Xu, The ghost fluid method for deflagration and detonation discontinuities, Journal of Computational Physics 154 (2) (1999) 393–427, doi:10.1006/jcph.1999.6320.
[7] R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), Journal of Computational Physics 152 (2) (1999) 457–492, doi:10.1006/jcph.1999.6236.
[8] M. Kang, R.P. Fedkiw, X.-D. Liu, A boundary condition capturing method for multiphase incompressible flow, Journal of Scientific Computing 15 (2000) 323–360, doi:10.1023/A:1011178417620.
[9] X.-D. Liu, R.P. Fedkiw, M. Kang, A boundary condition capturing method for Poisson's equation on irregular domains, Journal of Computational Physics 160 (1) (2000) 151–178, doi:10.1006/jcph.2000.6444.
[10] M.-C. Lai, C.S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, Journal of Computational Physics 160 (2) (2000) 705–719, doi:10.1006/jcph.2000.6483.
[11] Z. Li, M.-C. Lai, The immersed interface method for the Navier–Stokes equations with singular forces, Journal of Computational Physics 171 (2) (2001) 822–842, doi:10.1006/jcph.2001.6813.
[12] D.Q. Nguyen, R.P. Fedkiw, M. Kang, A boundary condition capturing method for incompressible flame discontinuities, Journal of Computational Physics 172 (1) (2001) 71–98, doi:10.1006/jcph.2001.6812.
[13] L. Lee, R.J. LeVeque, An immersed interface method for incompressible Navier–Stokes equations, SIAM Journal on Scientific Computing 25 (3) (2003) 832–856, doi:10.1137/S1064827502414060.
[14] F. Gibou, L. Chen, D. Nguyen, S. Banerjee, A level set based sharp interface method for the multiphase incompressible Navier–Stokes equations with phase change, Journal of Computational Physics 222 (2) (2007) 536–555, doi:10.1016/j.jcp.2006.07.035.
[15] Y. Gong, B. Li, Z. Li, Immersed-interface finite-element methods for elliptic interface problems with nonhomogeneous jump conditions, SIAM Journal on Numerical Analysis 46 (1) (2008) 472–495, doi:10.1137/060666482.
[16] J. Dolbow, I. Harari, An efficient finite element method for embedded interface problems, International Journal for Numerical Methods in Engineering 78 (2) (2009) 229–252, doi:10.1002/nme.2486.
[17] J. Bedrossian, J.H. von Brecht, S. Zhu, E. Sifakis, J.M. Teran, A second order virtual node method for elliptic problems with interfaces and irregular domains, Journal of Computational Physics 229 (18) (2010) 6405–6426, doi:10.1016/j.jcp.2010.05.002.
[18] L.N. Trefethen, D. Bau, Numerical linear algebra, SIAM: Society for Industrial and Applied Mathematics (1997).
[19] A. Mayo, The fast solution of Poisson's and the biharmonic equations on irregular regions, SIAM Journal on Numerical Analysis 21 (2) (1984) 285–299, doi:10.1137/0721021.
[20] H.S. Udaykumar, R. Mittal, W. Shyy, Computation of solid–liquid phase fronts in the sharp interface limit on fixed grids, Journal of Computational Physics 153 (2) (1999) 535–574, doi:10.1006/jcph.1999.6294.
[21] F. Gibou, R.P. Fedkiw, L.-T. Cheng, M. Kang, A second-order-accurate symmetric discretization of the Poisson equation on irregular domains, Journal of Computational Physics 176 (1) (2002) 205–227, doi:10.1006/jcph.2001.6977.

[22] Z. Jomaa, C. Macaskill, The embedded finite difference method for the Poisson equation in a domain with an irregular boundary and Dirichlet boundary conditions, Journal of Computational Physics 202 (2) (2005) 488–506, doi:10.1016/j.jcp.2004.07.011.
[23] F. Gibou, R. Fedkiw, A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains, with applications to the Stefan problem, Journal of Computational Physics 202 (2) (2005) 577–601, doi:10.1016/j.jcp.2004.07.018.
[24] H. Chen, C. Min, F. Gibou, A supra-convergent finite difference scheme for the Poisson and heat equations on irregular domains and non-graded adaptive Cartesian grids, Journal of Scientific Computing 31 (1) (2007) 19–60, doi:10.1007/s10915-006-9122-8.
[25] J.A. Sethian, J. Strain, Crystal growth and dendritic solidification, Journal of Computational Physics 98 (2) (1992) 231–253, doi:10.1016/0021-9991(92)90140-T.
[26] S. Chen, B. Merriman, S. Osher, P. Smereka, A simple level set method for solving Stefan problems, Journal of Computational Physics 135 (1) (1997) 8–29, doi:10.1006/jcph.1997.5721.
[27] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, Journal of Computational Physics 79 (1) (1988) 12–49, doi:10.1016/0021-9991(88)90002-2.
[28] J.-C. Nave, R.R. Rosales, B. Seibold, A gradient-augmented level set method with an optimally local, coherent advection scheme, Journal of Computational Physics 229 (10) (2010) 3802–3827, doi:10.1016/j.jcp.2010.01.029.