A direct forcing, immersed boundary method for conjugate heat transport

Kimmo Koponen[a], Amneet Pal Singh Bhalla[b], Brennan Sprinkle[c], Ning Wu[d], Nils Tilton[a,*]

[a] *Colorado School of Mines, Department of Mechanical Engineering, Golden, CO, USA*
[b] *San Diego State University, Department of Mechanical Engineering, San Diego, CA, USA*
[c] *Colorado School of Mines, Department of Applied Mathematics and Statistics, Golden, CO, USA*
[d] *Colorado School of Mines, Department of Chemical and Biological Engineering, Golden, CO, USA*

Abstract

Motivated by applications to fluid flows with conjugate heat transfer and electrokinetic effects, we propose a direct forcing immersed boundary method for simulating general, discontinuous, Dirichlet and Robin conditions at the interface between two materials. In comparison to existing methods, our approach uses smaller stencils and accommodates complex geometries with sharp corners. The method is built on the concept of a "forcing pair," defined as two grid points that are adjacent to each other, but on opposite sides of an interface. For 2D problems this approach can simultaneously enforce discontinuous Dirichlet and Robin conditions using a six-point stencil at one of the forcing points, and a 12-point stencil at the other. In comparison, prior work requires up to 14-point stencils at both points. We also propose two methods of accommodating surfaces with sharp corners. The first locally reduces stencils in sharp corners. The second uses the signed distance function to globally smooth all corners on a surface. The smoothing is defined to recover the actual corners as the grid is refined. We verify second-order spatial accuracy of our proposed methods by comparing to manufactured solutions to the Poisson equation with challenging discontinuous fields across immersed surfaces. Next, to explore the performance

*Corresponding author
Email address: ntilton@mines.edu (Nils Tilton)

of our method for simulating fluid flows with conjugate heat transport, we couple our method to the incompressible Navier-Stokes and continuity equations using a finite-volume projection method. We verify the spatial-temporal accuracy of the solver using manufactured solutions and an analytical solution for circular Couette flow with conjugate heat transfer. Finally, to demonstrate that our method can model moving surfaces, we simulate fluid flow and conjugate heat transport between a stationary cylinder and a rotating ellipse or square.

Keywords: Immersed Boundary Methods, Conjugate Heat Transport, Finite Volume Methods

---

Nomenclature

Symbols

$\alpha,\ \beta$	Coefficients in the Robin boundary condition, see Eqn. (4).
Δt	Time step (s).
$\Delta x,\ \Delta y$	Cell size in x and y-directions (m), see Fig. 1(b).
δ	Offset for the smoothing method (m).
$\dot{\theta}$	Angular velocity (s^{-1}).
Γ	Surface separating phases A and B, see Fig. 1(a).
NL	Nonlinear advection term in the Navier-Stokes equation, see Eq. 25.
$\mathbf{n}$	Surface normal unit vector, see Fig. 1(a).
$\mathbf{u}$	Velocity vector (ms^{-1}).
$\mathbf{x_s}$	Point where Γ intersects a grid line, see Fig. 4.
μ	Dynamic viscosity ($kgm^{-1}s^{-1}$).
Ω_A	Domain occupied by phase A, see Fig. 1(a).
Ω_B	Domain occupied by phase B, see Fig. 1(a).
$\phi(\mathbf{x})$	Distance function.
ρ	Density (kgm^{-3}).
$\widehat{x},\ \widehat{y}$	Local Cartesian coordinates, see Fig. 5.
$a_k,\ b_k$	Coefficients used to apply the Dirichlet condition at Γ. See Eqns. (8)-(9).
c	Specific heat ($Jkg^{-1}K^{-1}$).

c_k	Coefficients used to apply the Robin condition at Γ. See Eqn. (11).
ERR_2	L_2 norm used to measure error. See Eqn. (15).
ERR_∞	L_∞ norm used to measure error. See Eqn. (15).
k	Thermal conductivity ($\mathrm{Wm^{-1}K^{-1}}$).
L	Length and width of computational domain, see Figs. 18(b), and 22.
l	Side length of immersed square, see Fig. 22(b).
l_1, l_2	Semi-major and semi-minor axes of immersed ellipse, see Fig. 22(a).
N	Number of finite volume cells in both x and y-directions.
p	Pressure (Pa).
$r,\ \theta$	Polar coordinates, see Fig. 18.
R_Γ	Radius of Γ, see Fig. 18.
R_e	Radial distance from $(x, y) = (0, 0)$ to the local surface of immersed ellipse or square, see Eq. (30).
R_i	Inner radius of the annular solid region, see Fig. 18.
R_o	Outer radius of the annular fluid region, see Fig. 18.
Re	Reynolds number.
T	Temperature (K).
t	Time (s).
$x,\ y$	Cartesian coordinates (m).
$x_c,\ y_c$	Centroid of immersed object (m).
IB	Immersed boundary.

1. Introduction

This study is motivated by the challenge of simulating fluid flows over complex surfaces with conjugate heat transfer, as in heat exchangers [1] and porous media [2, 3]. In such flows, the temperature field must simultaneously satisfy a Dirichlet and Neumann condition at the fluid-solid interface. In cases where the surface geometry is also complex, or a parametric study requires simulating many potential geometries, the generation of body-fitted grids is often a time-consuming bottleneck [4]. Though not the focus of the current study, when the

surface also moves, as in particulate flows [5], the use of body-fitted grids can become altogether untenable. Furthermore, while the current study is motivated by conjugate heat transfer, similar interface conditions arise in flows with electrodynamic effects [6] and phase changes [7].

Immersed boundary (IB) methods are a popular alternative to body-fitted grids when simulating fluid flows with complex surface geometries. IB methods can be roughly divided into two approaches, depending on whether they use continuous or discrete forcing [8]. Continuous forcing was first proposed by Peskin [9] for simulating blood flow through a heart valve. Peskin's approach approximates an immersed boundary by introducing a smoothly varying forcing function in the momentum equations. The method was originally developed to simulate elastic boundaries, but was extended to rigid boundaries by Beyer and Leveque [10] and Goldstein *et al.* [11]. Though second-order accuracy was reported, these extensions were subject to stability constraints, and did not provide a sharp representation of the interface [8]. The subsequent rigid multi-blob method [12, 13] overcame these stability restrictions, but it is not a sharp interface method, nor is it capable of handling arbitrary boundary conditions.

The discrete forcing approach [14, 15] alleviates the stability constraints of continuous forcing by discretely deriving a forcing term that provides a sharp representation of the surface, with up to second-order accuracy. It should be noted, however, that recomputing the discrete forcing at each time step can make the simulation of moving boundaries more difficult than with continuous forcing [8]. Within the family of discrete forcing methods, prior work has simulated conjugate heat transfer using the cut-cell method [16–18], immersed interface method [19], and direct forcing method [15]. The cut-cell method cuts and reshapes finite-volume cells traversed by the immersed boundary. The approach is physically intuitive and accurately represents advective and diffusive fluxes through the interface. However, cutting and reshaping finite-volume cells around complex surfaces is complicated in 3D. In some cases, the method also generates cut cells with small volumes that reduce numerical stability [20]. As an alternative, the direct forcing method uses numerical interpolation or ex-

trapolation to force the desired interface condition at the immersed boundary. Though several interpolation methods for simulating conjugate heat transfer have been developed [3–5, 21, 22], the proposed extrapolation and interpolation schemes are complicated. In some cases, the simulation of a 2D problem requires up to 14-point interpolation stencils at every grid point adjacent to an immersed surface [22]. Such large stencils work well for smooth surfaces but are difficult to accommodate in sharp corners.

The objective of the current study is to explore a potentially simpler direct forcing method that can simulate general discontinuous Dirichlet and Robin conditions to second-order accuracy over complex surfaces with sharp corners. Though we are motivated by applications to conjugate heat transfer, we develop and verify the method for general discontinuous boundary conditions for potential future applications to surfaces with phase changes. Our extrapolation procedure uses the concept of a "forcing pair," defined as two grid points that are adjacent to each other, but on opposite sides of an interface. For 2D problems, we show that we can simultaneously enforce discontinuous Dirichlet and Robin conditions using a six point stencil at one of the forcing points and a 12-point stencil at the other. In contrast, prior work requires up to 14-point stencils at both points. The savings become more apparent in 3D problems, where our method requires a six point stencil at one point and a 20-point stencil at the other, compared to up to 30-point stencils at each point using prior methods. Furthermore, the extrapolations used in our approach are comparable in complexity to those already used in cut cell methods [17], but bypass the need to cut and reshape computational cells.

We note here that the Ghost Fluid Method (GFM) [23] is another popular method for solving Poisson problems with jump conditions. As opposed to the current method, GFM does not achieve second-order accuracy when thermophysical properties such as the thermal conductivity vary widely between domains [24]. The GFM uses a dimension-by-dimension splitting approach to apply jump conditions. Generally, only the normal component of the jump condition is sharply resolved, whereas the tangential components are smeared. The

present method is inherently multidimensional and resolves all components of the jump condition sharply. We show that our approach remains second-order accurate, regardless of the conductivity ratio between the two domains.

The remaining article is organized as follows. In section 2, we present our IB method for the case of a 2D Poisson problem. In section 3, we verify the spatial accuracy for eight different immersed surfaces, including cases with sharp corners. We compare the performance of two approaches of accommodating sharp corners. The first uses reduced stencils and cuts sharp corners along a grid line when necessary. The second approach uses the signed distance function to globally smooth all corners. The smoothing is defined to recover the actual corners as the grid is refined. In section 4, we extend our IB method to fluid flows with conjugate heat transport. For that, we couple our method to the incompressible Navier-Stokes and continuity equations using the finite-volume projection method of Bell *et al.* [25]. We verify the spatial-temporal accuracy of the method by comparing with a manufactured solution. We then apply our method to the simulation of a circular Couette flow with conjugate heat transfer through the inner cylinder, which is a test problem considered previously in [21, 22]. Finally, to demonstrate that our method can model moving surfaces, we simulate fluid flow and conjugate heat transfer in a circular Couette flow for which the inner cylinder is replaced with a rotating ellipse or square. To verify our results, simulations are repeated using both a fixed frame and a frame that rotates with the ellipse or square. Section 5 summarizes our conclusions. For completeness, we also provide a comparison of our method with prior work in the Supplementary Information.

2. Methodology

We present our approach by considering a 2D Poisson equation in the domain $x \in [0, 2\pi]$ and $y \in [0, 2\pi]$ sketched in Fig. $1(a)$. Two stationary phases A and B occupy the domains Ω_A and Ω_B, respectively, and are separated by the surface

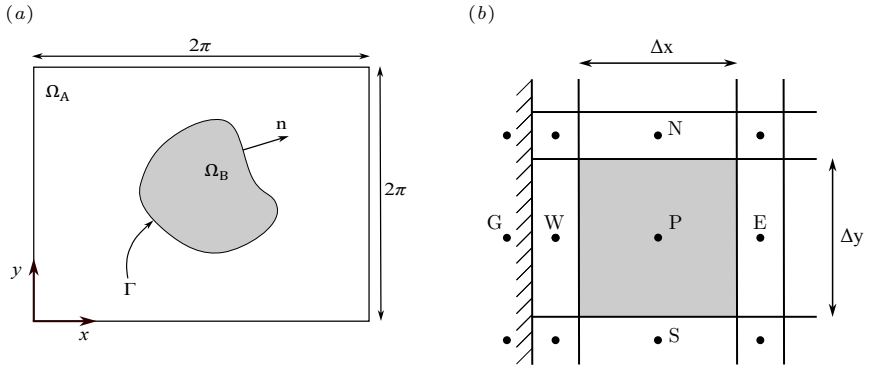

Figure 1: (a) A two-dimensional test problem where phase B is immersed in phase A. The phases are separated by surface Γ. The unit normal $\mathbf{n}$ points from phase B to phase A. (b) The Cartesian, non-uniform, finite volume grid. The temperature is stored at the cell centroids (solid dots). Cell boundaries are drawn using solid lines. Point G is a ghost node used to apply boundary conditions on the external boundary $x = 0$.

Γ. The temperature $T(x, y)$ satisfies the following Poisson equations,

$$\nabla^2 T_A = f_A(x, y) \qquad \text{for} \quad \mathbf{x} \in \Omega_A, \tag{1}$$

$$\nabla^2 T_B = f_B(x, y) \qquad \text{for} \quad \mathbf{x} \in \Omega_B, \tag{2}$$

where f_A and f_B are prescribed forcing functions, and the subscripts A and B denote quantities in the A and B phases, respectively. On the surface Γ, we consider general discontinuous Dirichlet and Robin boundary conditions,

$$T_A\Big|_\Gamma - T_B\Big|_\Gamma = g(x, y) \tag{3}$$

$$\Big[\alpha_A T_A + \beta_A(\boldsymbol{\nabla} T_A \cdot \mathbf{n})\Big]_\Gamma - \Big[\alpha_B T_B + \beta_B(\boldsymbol{\nabla} T_B \cdot \mathbf{n})\Big]_\Gamma = h(x, y), \tag{4}$$

where α_A, α_B, β_A, and β_B are constant coefficients, and g and h are prescribed forcing terms. In a conjugate heat transport problem with continuous temperature and diffusive heat flux at Γ, the terms g, h, α_A, and α_B are typically all zero. On the exterior boundary, we apply the homogeneous Dirichlet conditions

$$T_A\Big|_{x=0,2\pi} = T_A\Big|_{y=0,2\pi} = 0. \tag{5}$$

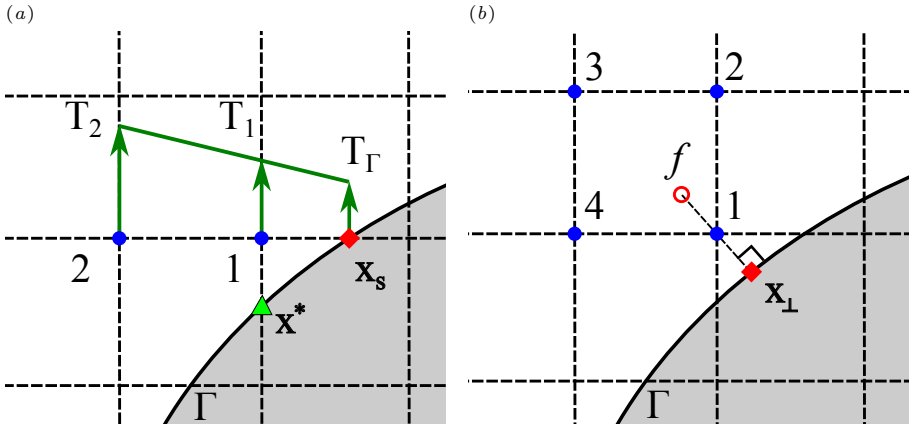

Figure 2: (a) The direct forcing method of Fadlun *et al.* [15]. The solid dots denote grid points where temperature is stored, and the dashed lines denote grid lines passing through the grid points. Temperature values at points 1 and 2 are computed by linear extrapolation from the boundary condition. (b) Method proposed by Balaras [27]. Temperature at the fictitious point f is interpolated using the neighbouring points so that the Dirichlet condition at the boundary is satisfied.

We discretize Eqns. (1) and (2) using standard, second-order, finite volume methods [26] on a non-uniform Cartesian grid, as sketched in Fig. 1(b). The temperature is stored at the cell centroids (solid dots), and cell boundaries are drawn using solid lines. For the cell shaded gray, the discretized Poisson equation (1) can be written as

$$\frac{1}{\Delta x}\left[\frac{T_E - T_P}{x_E - x_P} - \frac{T_P - T_W}{x_P - x_W}\right] + \frac{1}{\Delta y}\left[\frac{T_N - T_P}{y_N - y_P} - \frac{T_P - T_S}{y_P - y_S}\right] = f_A(x_P, y_P), \quad (6)$$

where Δx and Δy are the cell dimensions, and the subscripts denote evaluation at the grid points labeled in Fig. 1(b). The exterior boundary conditions are applied using ghost nodes. For example, the Dirichlet condition at $x = 0$ is approximated as $(T_W + T_G)/2 = 0$, where T_G is labeled in Fig. 1(b).

2.1. Immersed boundary method

Our IB method can be described as an extension of the original direct forcing method of Fadlun *et al.* [15], which was developed to apply Dirichlet conditions

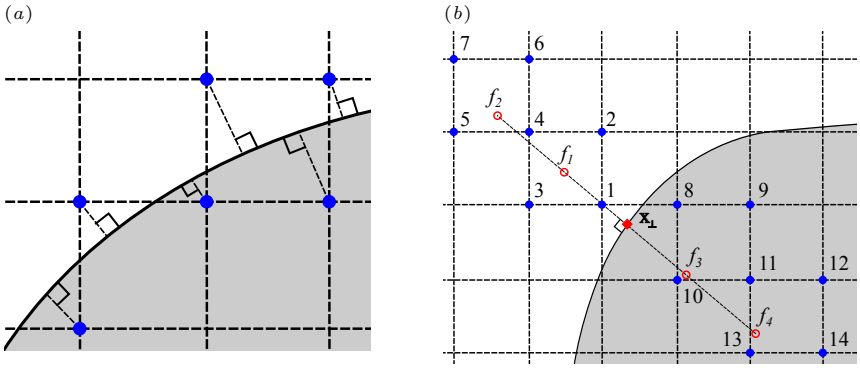

Figure 3: (a) The forcing points (solid blue dots) do not share the same surface point. This complicates applying conjugate heat transfer conditions. (b) Method by Nagendra *et al.* [22]. Four fictitious points (red circles) and their respective interpolation stencils (solid blue dots) are used to model conjugate heat transfer at Γ.

on immersed surfaces. For demonstration, consider the segment of Γ in Fig. 2(a), where the solid dots denote grid points where the temperature is stored, and the dashed lines denote grid lines passing through the grid points. Suppose we want to solve the Poisson equation (1) for T_A, subject to a desired temperature T_Γ on the surface Γ. The method of Fadlun *et al.* begins by designating the point T_1 as a "forcing point," because it has a neighbour in phase B. At the forcing point, the discretized Eqn. (6) is replaced with the condition

$$T_1(1+\sigma) - \sigma T_2 = T_\Gamma, \tag{7}$$

where $\sigma = (x_s - x_1)(x_1 - x_2)$. This condition applies T_Γ at $\mathbf{x_s}$ using linear extrapolation from points 1 and 2. The method is second-order accurate, but has some ambiguity as to whether the forcing point T_1 should be used to force the boundary condition at $\mathbf{x_s}$ or the point labeled with a solid triangle in Fig. 2(a). To remove this ambiguity, Balaras [27] used bilinear extrapolation to apply T_Γ to the point $\mathbf{x}_\perp$, which is the closest surface point to T_1, as shown in Fig. 2(b). For that, they introduce a fictitious point f whose value is approximated using linear extrapolation along the normal $\mathbf{n}$ passing through T_1 and $\mathbf{x_s}$. T_f is then approximated in terms of T_1, T_2, T_3, and T_4.

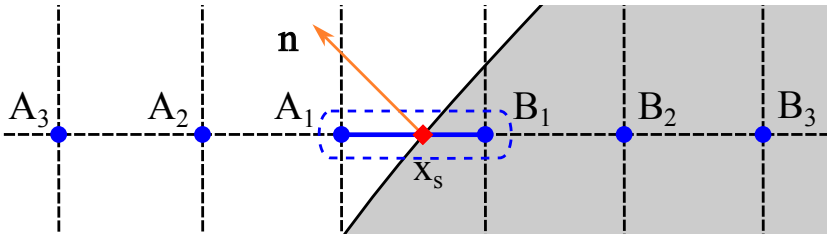

Figure 4: Forcing pair A_1-B_1 circled by a blue dashed line, and the six point extrapolation stencil used for the Dirichlet condition.

The method of Balaras has been extended to Neumann boundary conditions by several authors [28–30]. This is usually done by adding a second fictitious point along the surface normal to allow an approximation of the Neumann condition at the immersed surface. However, this approach becomes less straightforward when modeling conjugate heat transfer, because the forcing points on opposite sides of Γ do not share a common surface point, as shown in Fig. 3(a). Nagendra *et al.* [22] addressed this issue using a procedure that introduces four fictitious points, labelled f_1, f_2, f_3, and f_4 in Fig. 3(b), that lie on a line normal to the surface, passing through the forcing point, labelled T_1 in Fig. 3(b). The temperatures at these fictitious points are interpolated using neighbouring grid points, labelled 1-14 in Fig. 3(b). Consequently, each forcing point requires an interpolation stencil of up to 14 points for 2D simulations, or up to 30 points for 3D problems. For an unsteady problem, the complexity can be reduced by using the temperature field from a previous time step for the interpolation [3], though this may impact numerical stability.

2.2. Current work

The current work builds on that of Nagendra *et al.* [22] and Das *et al.* [2, 3], but leverages the concept of a "forcing pair," defined as two adjacent grid points on a common grid line, but on opposite sides of Γ. In Fig. 4, the points labelled A_1 and B_1 (circled by a blue dashed line) are a forcing pair. These two points are used to apply the discontinuous Dirichlet and Robin conditions (3)-(4) at the shared surface point $\mathbf{x_s}$, where Γ intersects the grid line. This is similar to

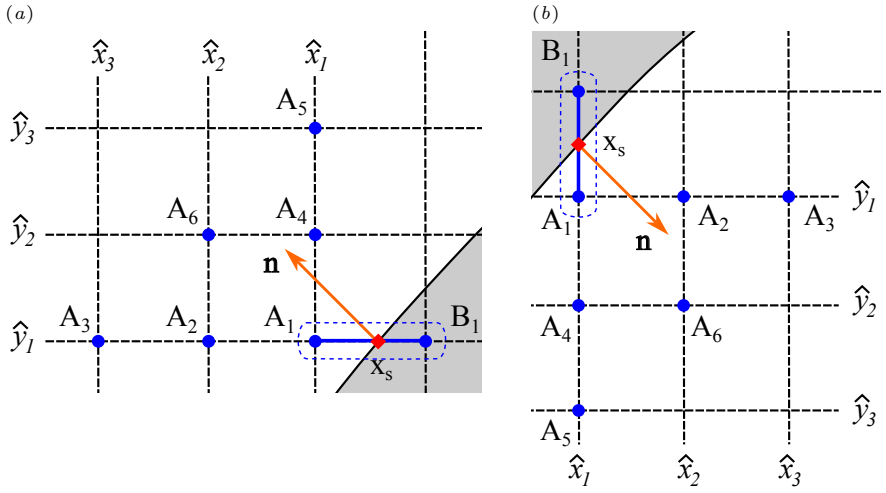

Figure 5: Extrapolation stencils to approximate $\boldsymbol{\nabla} T_A$. (a) shows a case where a forcing pair lies on a horizontal grid line, $\widehat{y}_1$, and (b) where a forcing pair lies on a vertical grid line $\widehat{x}_1$.

the original approach of Fadlun *et al.*, which uses surface points $\mathbf{x_s}$ on the grid lines. This allows the Dirichlet condition (3) to be applied by approximating T_A and T_B at $\mathbf{x_s}$ using quadratic extrapolation along the grid line,

$$T_A\Big|_{\mathbf{x_s}} \approx a_1 T_{A1} + a_2 T_{A2} + a_3 T_{A3}, \tag{8}$$

$$T_B\Big|_{\mathbf{x_s}} \approx b_1 T_{B1} + b_2 T_{B2} + b_3 T_{B3}, \tag{9}$$

where the extrapolation coefficients a_j and b_j are provided in Appendix A. In Fig. 4, $\mathbf{x_s}$ lies on a horizontal grid line, but the same method applies when $\mathbf{x_s}$ lies on a vertical grid line. Substituting these approximations into condition (3) produces a six point stencil. Note that when setting a desired temperature on an immersed surface, the method of Fadlun *et al.* [15] achieves second-order accuracy using linear extrapolation. For conjugate heat transfer, however, we found that quadratic extrapolation is required to consistently produce second-order accuracy for the problems considered in section 3.

The application of the Robin condition (4) requires estimates for $\boldsymbol{\nabla} T_A \cdot \mathbf{n}$ and $\boldsymbol{\nabla} T_B \cdot \mathbf{n}$. For that, consider the two cases in Fig. 5. Panel (a) shows a case where $\mathbf{x_s}$ lies on the horizontal grid line labelled $\widehat{y}_1$. Panel (b) shows a case

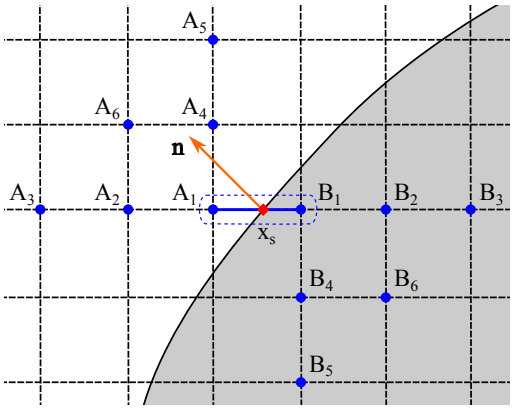

Figure 6: Forcing pair A_1-B_1 circled by a blue dashed line, and the 12-point extrapolation stencil for the Robin condition.

where $\mathbf{x_s}$ lies on a vertical grid line labelled $\widehat{x}_1$. To approximate $\boldsymbol{\nabla} T_A \cdot \mathbf{n}$, we first approximate T_A using a second-order polynomial of the form

$$T_A(\widehat{x}, \widehat{y}) = c_1 + c_2\widehat{x} + c_3\widehat{x}^2 + c_4\widehat{y} + c_5\widehat{y}^2 + c_6\widehat{x}\widehat{y}, \tag{10}$$

where the spatial coordinates $\widehat{x}$ and $\widehat{y}$ are measured relative to the node A_1. The coefficients c_j are determined by fitting Eqn. (10) to six temperature nodes. A unique solution requires the six nodes to include three points with a unique x-coordinate, three points with unique a y-coordinate, and a maximum of three points along any one grid line. Whenever possible, we choose the six points labelled A_1 to A_6 in Fig. 5. The points A_2 and A_3 lie on the horizontal grid line passing through A_1, while A_4 and A_5 lie on the vertical grid line passing through A_1. The points A_2 and A_3 are taken to the left of A_1 when $n_x < 0$, and to the right of A_1 when $n_x > 0$. The points A_4 and A_5 follow a similar rule with respect to n_y. Leveraging the fact that $\widehat{y} = 0$ for A_2 and A_3, and $\widehat{x} = 0$ for A_4 and A_5, the expansion coefficients have the solution

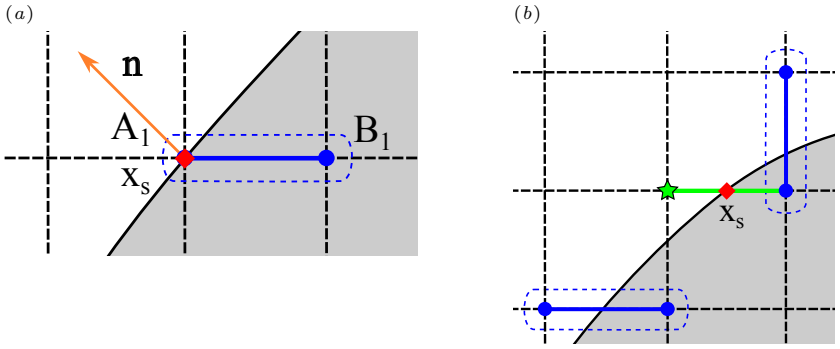

Figure 7: (a) Special case where the forcing point A_1 lies on Γ. (b) Orphan point (green star) and its corresponding surface point, (x_s, y_s).

$$c_2 = \frac{(x_2^2 - x_3^2)T_{A1} + x_3^2 T_{A2} - x_2^2 T_{A3}}{x_2 x_3 (x_3 - x_2)}, \qquad c_3 = \frac{(x_3 - x_2)T_{A1} - x_3 T_{A2} + x_2 T_{A3}}{x_2 x_3 (x_3 - x_2)},$$

$$c_4 = \frac{(y_2^2 - y_3^2)T_{A1} + y_3^2 T_{A4} - y_2^2 T_{A5}}{y_2 y_3 (y_3 - y_2)}, \qquad c_5 = \frac{(y_3 - y_2)T_{A1} - y_3 T_{A4} + y_2 T_{A5}}{y_2 y_3 (y_3 - y_2)},$$

$$c_1 = T_{A1}, \qquad c_6 = \frac{T_{A1} - T_{A2} - T_{A4} + T_{A6}}{x_2 y_2}. \tag{11}$$

The normal gradient $\boldsymbol{\nabla} T_A \cdot \mathbf{n}$ can then be approximated at $\mathbf{x_s}$ as

$$\boldsymbol{\nabla} T_A \cdot \mathbf{n} = n_x(c_2 + 2c_3\widehat{x}_s + c_6\widehat{y}_s) + n_y(c_4 + 2c_5\widehat{y}_s + c_6\widehat{x}_s). \tag{12}$$

Expression (12) avoids the need to invert a six-by-six matrix, as in some cut-cell methods [17]. On a uniform grid, the coefficients c_j can also be precomputed. In the special case where $\mathbf{n}$ points along a grid line, the polynomial $T_a(\widehat{x}, \widehat{y})$ reduces to a one-dimensional expression. For example, when $n_y = 0$, Eqn. (10) simplifies to

$$T_A(\widehat{x}) = T_{A1} + c_2\widehat{x} + c_3\widehat{x}^2, \tag{13}$$

requiring only three grid points along $\widehat{y}_1$ for a second-order extrapolation. Finally, to apply the Robin condition (4), we repeat this procedure in phase B to approximate $\boldsymbol{\nabla} T_B \cdot \mathbf{n}$ at $\mathbf{x_s}$. We also approximate T_A and T_B at $\mathbf{x_s}$ as demonstrated in Eqns. (8)-(9). This produces the 12-point stencil shown in Fig. 6.

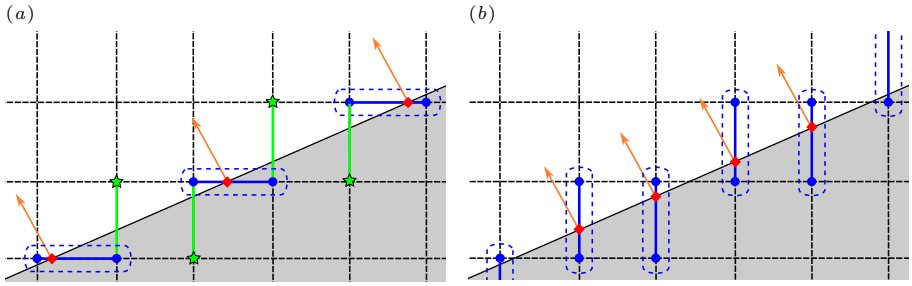

Figure 8: Prioritizing forcing pairs in the horizontal direction (a), and vertical direction (b). Prioritizing forcing pairs in the surface normal direction reduces the number of orphan points.

2.3. Special cases

We found four cases in which the procedure in section 2.2 must be modified. The cases labelled 1 and 2 below can arise on smooth surfaces, such as a circle. The cases labelled 3 and 4 tend to arise on surfaces with corners. It is worth noting that cases 1, 3, and 4 occur in other IB methods as well [3, 30, 31]. Case 2, however, is unique to our method.

2.3.1. Case 1. Forcing point on Γ

When a forcing point lies on Γ, we arbitrarily treat it as a forcing point in phase A, paired to a neighbouring point in phase B. Fig. 7(a) shows an example where node A_1 lies on Γ, and point B_1 is assigned as a partner. Dirichlet and Robin boundary conditions are then applied as before.

2.3.2. Case 2: Orphan points

Orphan points are forcing points whose only potential partner has already been paired to another point. Fig. 7(b) shows an orphan point labelled with a green star symbol. Orphan points only allow the application of one boundary condition. For simplicity, we apply the Dirichlet condition.

The number of orphan points can be minimized by optimizing the pairing algorithm. Fig. 8 shows a case where prioritizing pairs along horizontal grid lines (panel a) produces four orphan points, while prioritizing pairs along vertical grid lines (panel b) produces no orphans. More generally, we find that if a forcing

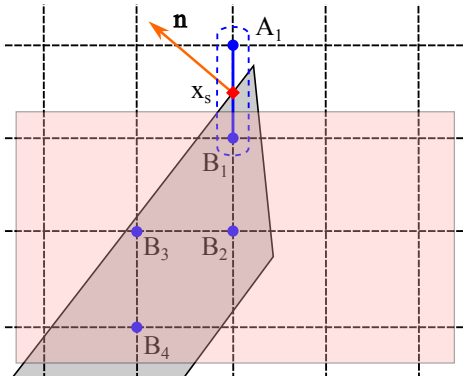

Figure 9: The extrapolation stencil search area shaded in red. There are only four eligible extrapolation points in phase B for the forcing pair circled by a blue dashed line.

point has two potential partners, the partner along the vertical grid line should be prioritized when $|n_y| > |n_x|$, and vice versa when $|n_x| > |n_y|$.

2.3.3. Case 3: Alternate stencils

In cases where the local surface geometry cannot accommodate a preferred extrapolation stencil, we seek an alternative six-point stencil within a maximum preset search window. If six suitable points are found, we determine the extrapolation coefficients using a matrix inversion method detailed in Appendix B. In cases where a six-point stencil is not found, we consider reduced stencils of four and five points. Figure 9 shows a reduced four point stencil, labelled B_1 - B_4. Note that the minimum number of points required to extrapolate the Robin condition is three, assuming that $\mathbf{n}$ does not point along a grid line. The three-point stencil must be formed so that there is a maximum of two points along one grid line. An example of a valid three-point stencil would be points B_1 - B_3 in Fig. 9. However, in our experience, the surface modifications discussed in Case 4 below prevent the need for such three-point stencils, at least for all geometries considered in this study.

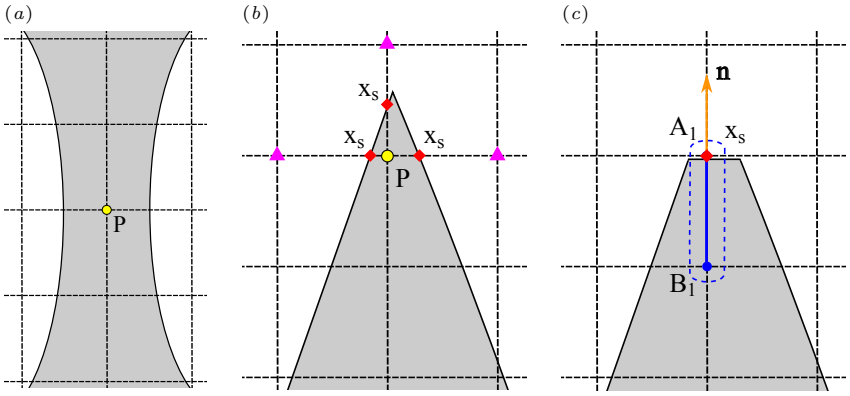

Figure 10: (a) A problem point (labelled P) that can be resolved by refining the grid. (b) A problem point (labelled P) that cannot be resolved by grid refinement. (c) Resolution of the problem point shown in panel (b) using the cutting method.

2.3.4. Case 4: Points requiring surface modification

Some surface geometries generate problem points at which no suitable stencil can be formed within a reasonable search window. In some cases, the problem point can be resolved by refining the grid, such as the problem point labelled P in Fig. 10(a). Other cases, however, require special treatment. These cases tend to occur in sharp corners. For example, the problem point labelled P in Fig. 10(b) has three potential forcing partners (labelled as purple triangles) that can only pair with P; however, there is no reasonable stencil for approximating $\partial T_B/\partial x$ at any of the surface points $\mathbf{x_s}$. Refining the grid tends to simply push the issue further into the corner.

We explore two remedies for such problem points. The first cuts the surface along a grid line passing through the problem point, as in Fig. 10(c). The problem point is then considered as lying on Γ, and treated as in Case 1 above. By cutting along the grid line, the surface normal points along the perpendicular grid line, permitting a co-linear three-point stencil.

The second approach eliminates problem points by rounding all sharp corners on the surface Γ. This is done as "preprocessing," before seeking forcing pairs, and has the added benefit of reducing the number of Case 3 points. How-

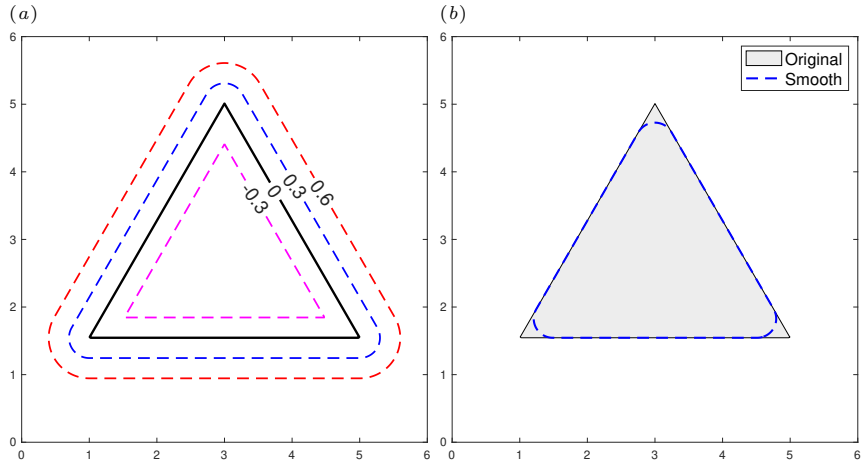

Figure 11: (a) The contour levels of a signed distance function $\phi(\mathbf{x})$. (b) Smoothing applied to a equilateral triangle. The true shape is shown as the shaded area, and the smoothed object in blue dashed line. The smoothing is exaggerated for illustrative purposes.

ever, it has the disadvantage that all corners are rounded, regardless of whether they contain problem points. We implement the smoothing using an approach detailed in Fayolle *et al.* [32]. The approach uses the signed distance function $\phi(\mathbf{x})$, for which Γ is the zero isocontour. For demonstration, Fig. 11(a) shows isocontours of $\phi(\mathbf{x})$ for an equilateral triangle. The original, unsmoothed, triangle is given by the isocontour $\phi = 0$ (solid line). The isocontours $\phi = -0.3$, $\phi = 0.3$, and $\phi = 0.6$ are shown as dashed lines. We see that positive isocontours ($\phi > 0$) smooth convex corners, but also expand the object. In contrast, negative isocontours ($\phi < 0$) preserve convex corners, but compress the object. To smooth the convex corners of an object, while otherwise preserving the original surface away from the corners, Fayolle *et al.* first compute the distance function $\phi_c(\mathbf{x})$ to the compressed surface Γ_c, where Γ_c is defined as the zero isocontour of the function $f(\mathbf{x}) = \phi(\mathbf{x}) - \delta$, where δ is a positive number. The final smoothed surface is then set to the isocontour $\phi_c(\mathbf{x}) = \delta$. Fig. 11$(b)$ demonstrates the smoothed surface (blue dashed line) when $\delta = 0.5$. The original unsmoothed triangle is shaded grey. By setting δ proportional to some measure of the cell

size (such as the maximum Δx or Δy in the domain), the smoothing recovers the original object as the grid is refined. Using a similar approach, one can also smooth concave corners [32].

It is worth noting that both methods above (cutting and smoothing) modify Γ from its original shape, which is a common issue when simulating sharp corners using immersed boundary methods [30, 31, 33]. Though beyond the scope of this study, future applications to multi-phase fluid-fluid flows may prefer the cutting method, because it has a smaller impact on the volume of the phases, particularly on coarse grids.

3. Verification of spatial accuracy for the Poisson equation

Appendix C describes an algorithm we coded in MATLAB to solve the Poisson problem (1)-(5). Here, we use the algorithm to explore the spatial accuracy of our method with respect to the following exact solution of Eqns. (1)-(2),

$$T_e = \begin{cases} \sin(x)\sin(y) & \mathbf{x} \in \Omega_A, \\ \cos(x)\cos(y) & \mathbf{x} \in \Omega_B, \end{cases} \tag{14}$$

which is forced by setting $f_A = -2\sin(x)\sin(y)$ and $f_B = -2\cos(x)\cos(y)$. Equation (14) is a challenging test solution with discontinuous values and gradients on Γ. We consider the eight surface geometries sketched in Fig. 12. Panel (a), which we refer to as a "puzzle piece," shows a case where both phases are subject to Dirichlet conditions on the external boundary. The remaining panels show cases where phase B is fully immersed in phase A. The shapes are a circle (b), square (c), cross (d), hexagram (e), equilateral triangle (f), isosceles triangle (g), and a right triangle (h) with angles of 30°, 60°, and 90°. In panels (b)-(h), we place the centroid of phase B at $(x_c, y_c) = (3.68, 3.68)$. The distance functions for the geometries in panels (b)-(h) are available in [34]. For all shapes, we repeat our numerical tests using immersed Neumann conditions ($\alpha_A = \alpha_B = 0$) and Robin conditions. For the Neumann conditions, we consider

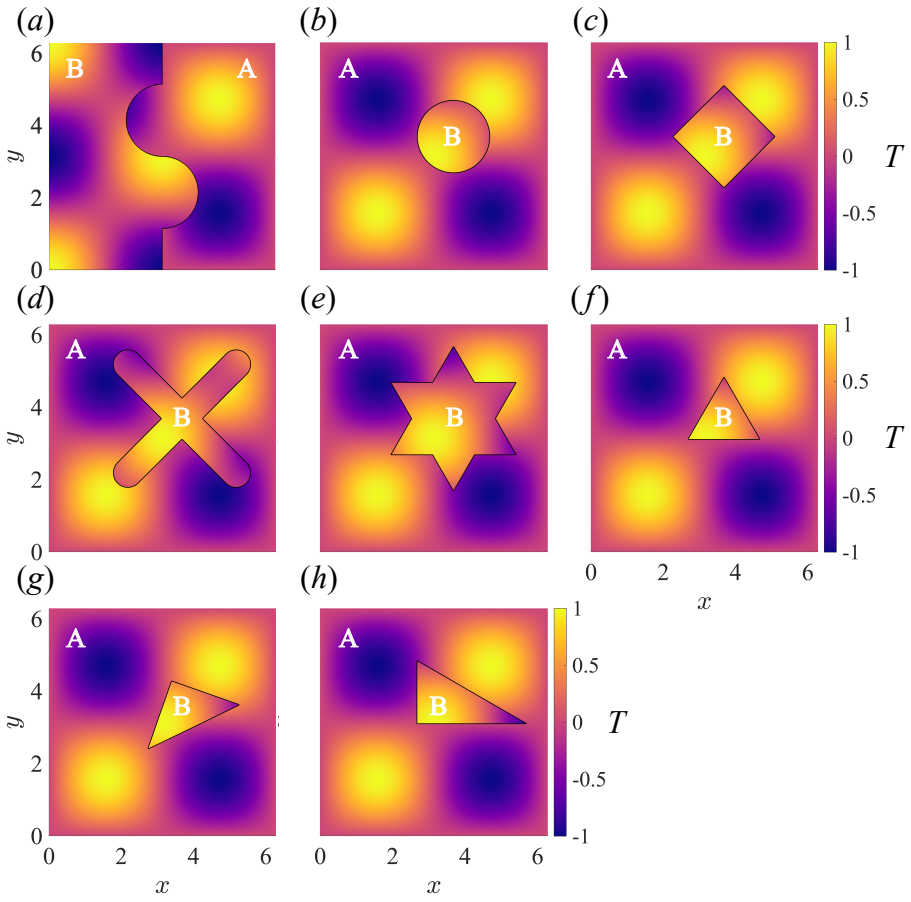

Figure 12: Eight geometries used to verify the method for the Poisson equation. Phases A and B are labelled in white font. In panel (a) phase B is shaped similar to a puzzle piece. In the remaining panels phase B is a circle (b), square (c), cross (d), hexagram (e), equilateral triangle (f), isosceles triangle (g), and right triangle (h), surrounded by phase A.

the simpler case where $\beta_A = \beta_B = 1$, as well as the more challenging case where $\beta_A = 1$ and $\beta_B = 100$. The latter models a case where the thermal conductivity of the two phases differ by two orders-of-magnitude. For the Robin conditions, we set $\alpha_A = 3,\ \alpha_B = 2$ and $\beta_A = \beta_B = 1$. In all cases, we set $h(x, y)$ and $g(x, y)$ to the forcing terms required by the analytical solution (14). For completeness, tests were repeated for uniform and non-uniform grids. The results shown here are produced using a non-uniform grid detailed in Appendix D.

Geometry	Surface modification	Slope of fit		R^2	
		ERR_∞	ERR_2	ERR_∞	ERR_2
Puzzle piece	Not applied	-1.89	-1.90	0.996	0.996
Circle	Not applied	-2.00	-2.00	0.998	0.998
Square	Cut	-2.07	-2.00	0.998	0.998
	Smooth	-2.07	-2.00	0.998	0.998
Cross	Cut	-2.03	-2.03	0.991	0.991
	Smooth	-2.00	-2.00	0.990	0.988
Hexagram	Cut	-1.97	-1.99	0.948	0.955
	Smooth	-2.05	-2.05	0.996	0.996
Equilateral triangle	Cut	-2.12	-2.11	0.987	0.988
	Smooth	-2.02	-2.00	0.974	0.975
Isosceles triangle	Cut	-1.97	-1.98	0.889	0.894
	Smooth	-2.09	-2.08	0.950	0.954
Right triangle	Cut	-1.84	-1.90	0.947	0.954
	Smooth	-2.03	-2.02	0.994	0.994

Table 1: The slopes of the linear fits and coefficients of determination (R^2) of the ERR_∞ and ERR_2 errorplots for the Neumann case with $\beta_A = 1,\ \beta_B = 100$.

We measure the spatial error using the two common norms below,

$$ERR_\infty = \left|\left|T_e - T_n\right|\right|_\infty, \qquad ERR_2 = \left[\int_0^{L_x}\int_0^{L_y}(T_e - T_n)^2 dxdy\right]^{0.5}, \quad (15)$$

where T_n is the numerical answer. For ERR_∞, we use the maximum absolute error over the internal grid points (excluding ghost nodes). For ERR_2, we approximate the integral using the mid-point method. In cases where cutting or smoothing Γ causes a grid point to change phase, we set T_e to that of the modified surface. For example, in Fig. 10, the exact solution at problem point P is set to T_A, not T_B. For the smoothing method, the offset δ is set to the smallest possible value that eliminates all problem points. This value is problem

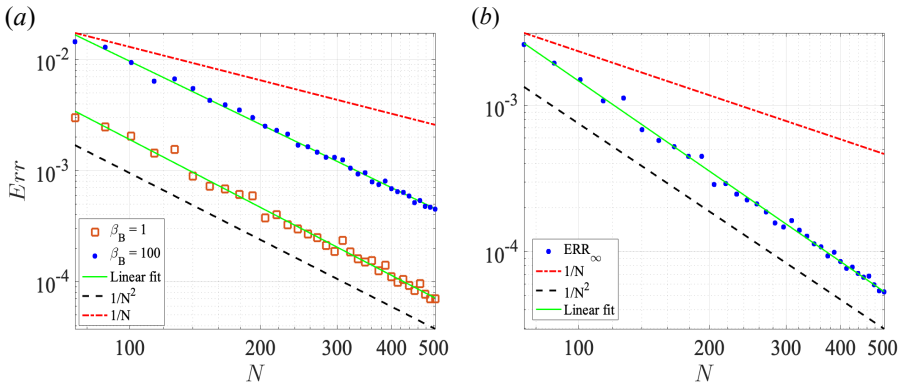

Figure 13: Results for the puzzle piece. Panel (a) shows results for the Neumann conditions ($\beta_A = \beta_B = 1$ and $\beta_A = 1,\ \beta_B = 100$). Panel ($b$) shows results for the Robin condition.

specific, and related to the grid resolution.

To explore spatial accuracy, we use N finite volume cells in each direction (a total of N^2 cells), and we investigate the variation of ERR_∞ between $N = 75$ and $N = 500$. We use linear regression to find the best fit for all error plots, and report the coefficient of determination (R^2) values for each plot. Figure 12 shows the numerically approximated temperature fields for each geometry when $N = 500$ and $\alpha_A = \alpha_B = 0$, $\beta_A = \beta_B = 1$. When comparing ERR_∞ and ERR_2, we found both methods produce similar results. Therefore, we only present error plots for the ERR_∞ norm. Table 1, however, presents the observed order-of-accuracies and R^2 values for each shape, using both norms.

3.1. Puzzle piece

The puzzle piece is the only geometry considered where phase B is not immersed in phase A. No surface modification is applied, because Γ produces no problem points. Figure 13(a) shows ERR_∞ versus N for the immersed Neumann condition with $\beta_A = \beta_B = 1$ (squares), and $\beta_A = 1,\ \beta_B = 100$ (solid dots). Figure 13(b) shows ERR_∞ versus N for the immersed Robin condition. The dashed black lines show $1/N^2$, the red dash-dotted lines show $1/N$, and solid green lines show best fit power laws. All cases show second-order spatial accuracy, though ERR_∞ in panel(a) is roughly five times larger when $\beta_B = 100$.

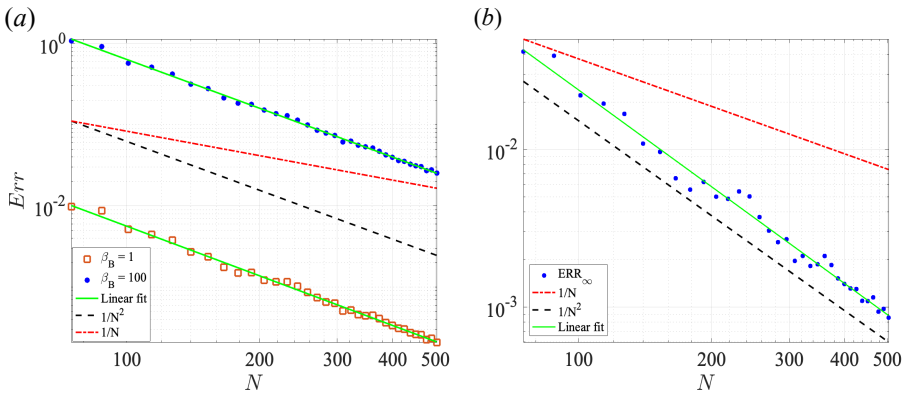

Figure 14: Results for circle. Panel (a) shows results for the Neumann conditions ($\beta_A = \beta_B = 1$ and $\beta_A = 1,\ \beta_B = 100$). Panel (b) shows results for the Robin condition.

3.2. Circle

The circle is the simplest of the fully immersed geometries, because the preferred extrapolation stencil is available for all forcing points, and no surface modification is needed. Figure 14(a) shows ERR_∞ versus N for the Neumann conditions with $\beta_A = \beta_B = 1$ (squares) and $\beta_A = 1,\ \beta_B = 100$ (solid dots). Figure 14(b) shows ERR_∞ for the Robin condition. All cases show second-order spatial accuracy. In panel (a), however, the error is roughly 100 times larger when $\beta_B = 100$. We observe this trend for all cases where phase B is fully immersed in phase A. We hypothesize that it occurs because the temperature field in a fully immersed object is no longer fixed by Dirichlet conditions on the external boundary, in contrast to the puzzle piece. It is also worth stressing that the manufactured solution is an unphysical test problem chosen to challenge our IB method. In a physical heat transfer problem, the coefficients β_A and β_B appear in the governing equations, and strongly influence T_A and T_B.

For all immersed geometries considered, the Neumann and Robin conditions produce nearly identical orders-of-accuracy. Hereinafter, we only present results for the Neumann condition with $\beta_A = 1,\ \beta_B = 100$, and we focus our discussion on the treatment of Case 4 points using cutting and smoothing.

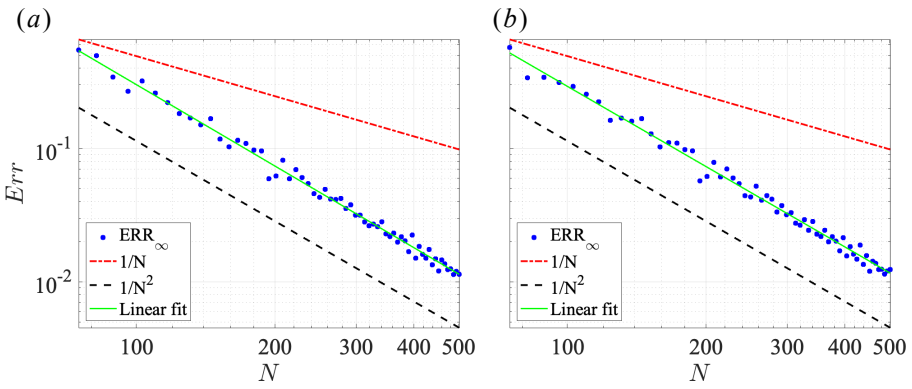

Figure 15: Spatial error plots for the cross produced by the cutting (a) and smoothing (b) methods.

3.3. Square and Cross

Figure 15 shows error plots for the cross using the cutting method (panel a) and smoothing (panel b). The error plots for the square are nearly identical to those in Fig. 15, and are not shown for brevity. Though the cross and square have 90° angles, the number of Case 3 and 4 points are few. The cutting and smoothing methods consequently produce similar second-order accuracy.

3.4. Hexagram

The hexagram is more prone to Case 4 problem points, because it has 60° angles. Figure 16 shows the error plots generated by the cutting (a) and smoothing (b) methods. We observe second-order accuracy for both methods; however, the cutting method produces a cloudier error plot. This likely arises because smoothing eliminates many Case 3 points, whereas cutting only eliminates Case 4 points.

3.5. Triangles

The error plots for the equilateral and isosceles triangles are qualitatively similar to those observed for the hexagram, and are not shown for brevity. Figure 17 shows the error plot for the right triangle. As observed for the hexagram, the cutting method produces a cloudier plot, which likely explains the

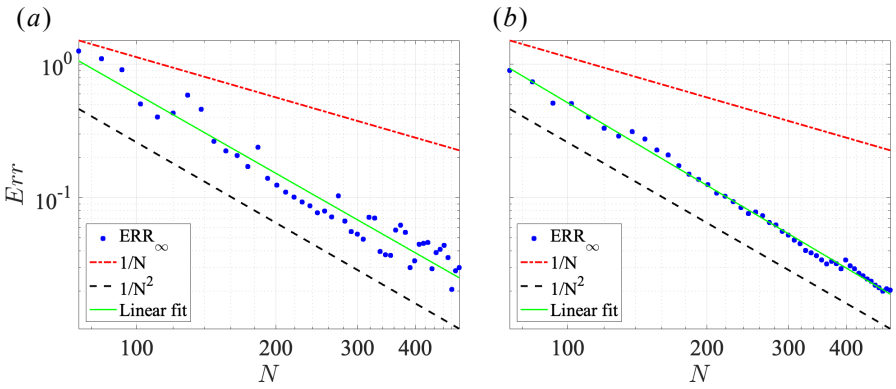

Figure 16: Spatial error plots for the hexagram produced by the cutting (a) and smoothing (b) methods.

slight reduction in the order-of-accuracy to 1.84. The smoothing has an order-of-accuracy of 2.03. It is worth noting that the right triangle is the only geometry we tested for which the cutting method decreased the order-of-accuracy below 1.97.

4. Conjugate heat transport with fluid flow

This section verifies the accuracy of our method when simulating fluid flows with conjugate heat transport. First, we use the method of manufactured solutions to verify the spatial and temporal accuracy of our method when coupled to the incompressible Navier-Stokes and continuity equations. Next, we simulate a circular Couette flow with conjugate heat transfer through the inner cylinder, which is a test problem considered previously in Refs. [21, 22]. Finally, to demonstrate that our method can model moving surfaces, we simulate fluid flow and conjugate heat transfer in a circular Couette flow for which the inner cylinder is replaced with a rotating ellipse or square. To verify our results, we repeat the simulations using both a fixed reference frame and a frame that rotates with the same angular velocity as the ellipse or square.

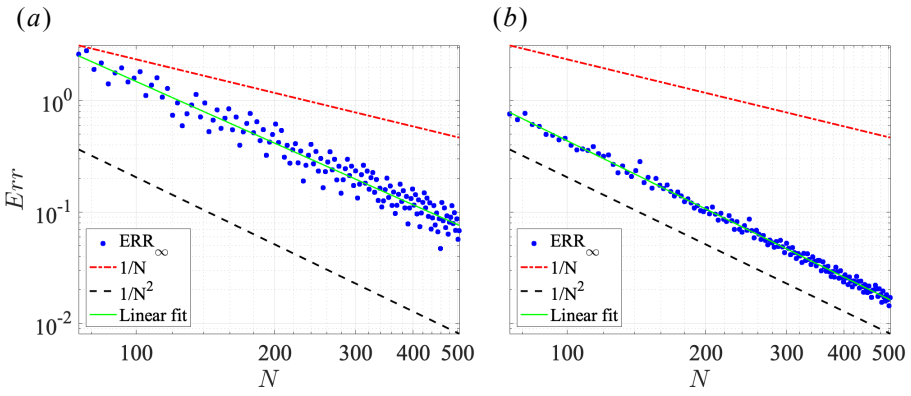

Figure 17: Spatial error plots for the right triangle produced by the cutting (a) and smoothing (b) methods.

4.1. Verification with manufactured solutions

To verify the spatial-temporal accuracy of our method when coupled to a fluid flow, we consider the 2D domain $x \in [0, 2\pi],\ y \in [0, 2\pi]$ sketched in Fig. 18(a). Phase B is a solid stationary circle of radius 1.5 centered at $(x_c, y_c) = (\pi, \pi)$. Phase A is a Newtonian fluid, in which the velocity $\mathbf{u}$, pressure p, and temperature T are governed by the incompressible Navier-Stokes, continuity, and heat equations,

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0, \tag{16}$$

$$\rho_A \left[\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} \right] = -\boldsymbol{\nabla} p + \mu_A \nabla^2 \mathbf{u} + \mathbf{F}, \tag{17}$$

$$\rho_A c_A \left[\frac{\partial T_A}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla}) T_A \right] = k_A \nabla^2 T + q_A, \tag{18}$$

where ρ_A, μ_A, c_A, and k_A are the fluid's density, dynamic viscosity, specific heat, and thermal conductivity, respectively. In phase B, T_B satisfies

$$\rho_B c_B \frac{\partial T_B}{\partial t} = k_B \nabla^2 T + q_B, \tag{19}$$

where ρ_B, c_B, and k_B are the solid's density, specific heat, and thermal conductivity, respectively. In Eqns. (17) - (19), the terms $\mathbf{F}$, q_A, and q_B are added to force the following manufactured solution,

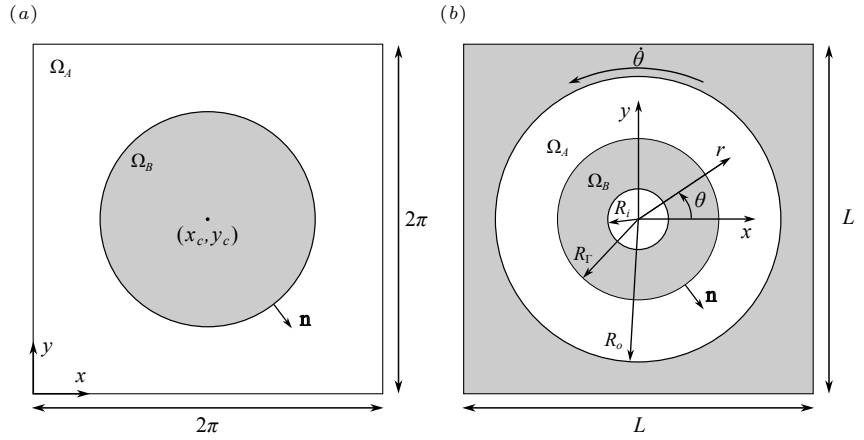

Figure 18: (a) A solid circle (phase B) immersed in a Newtonian fluid (phase A). Phase B is centered at $(x_c, y_c) = (\pi, \pi)$. (b) Circular Couette flow with conjugate heat transfer through the inner cylinder. The inner cylinder occupies $R_i \leq r < R_\Gamma$, the fluid occupies the region $R_\Gamma \leq r \leq R_o$, and the outer cylinder occupies $r > R_o$. Constant temperatures are applied at $r = R_i$ and $R = R_o$, driving conjugate heat transfer through the inner cylinder and fluid.

$$T_e = \begin{cases} \sin(x)\sin(y)\cos(\omega t), & \mathbf{x} \in \Omega_A, \\ \cos(x)\cos(y)\cos(\omega t), & \mathbf{x} \in \Omega_B, \end{cases} \tag{20}$$

$$u_e = \sin(x)\sin(y)\cos(\omega t), \qquad \mathbf{x} \in \Omega_A, \tag{21}$$

$$v_e = -\cos(x)\sin(y)\cos(\omega t), \qquad \mathbf{x} \in \Omega_A, \tag{22}$$

$$p_e = \sin(x)\sin(y)\cos(\omega t), \qquad \mathbf{x} \in \Omega_A. \tag{23}$$

The solution is steady when $\omega = 0$. We apply the following homogeneous Dirichlet conditions at the exterior boundaries,

$$T_A\Big|_{x=0,2\pi} = T_A\Big|_{y=0,2\pi} = 0, \qquad \mathbf{u}\Big|_{x=0,2\pi} = \mathbf{u}\Big|_{y=0,2\pi} = 0. \tag{24}$$

On the immersed surface, we apply the temperature conditions (3)-(4) with $\alpha_A = \alpha_B = 0$, $\beta_A = \beta_B = 1$. We also apply the velocity conditions $\mathbf{u}\big|_\Gamma = \mathbf{u}_e\big|_\Gamma$ using the linear extrapolation method of Fadlun *et al.* [15].

We approximate Eqns. (16)-(19) numerically using the methods detailed in Ref. [30], which are only briefly summarized here. Equations (16)-(19) are

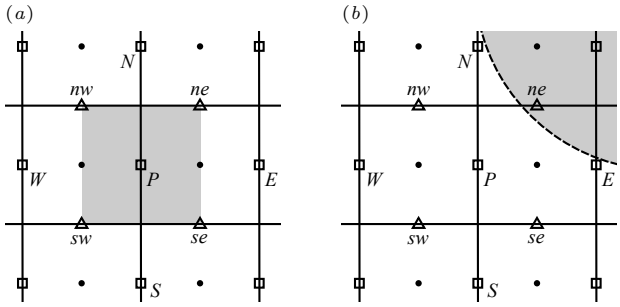

Figure 19: (a) A section of a staggered grid where temperature and pressure are stored at the cell centroids (solid dots), and velocity components u and v are stored at the cell faces (squares and triangles, respectively). The shaded area illustrates the control volume for discretizing the x-component of the Navier–Stokes equations. Nodes P, N, E, W, S, ne, nw, se, and sw form the stencil for discretizing the nonlinear advection term. (b) A special case where node P is a regular fluid node but node ne is in the solid.

discretized temporally using a second-order semi-implicit method in which the diffusion terms are approximated using the Crank-Nicolson method, and the advection terms are approximated using the Adams-Bashforth method. For example, the semi-discrete Navier-Stokes equations take the form

$$\rho\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t}+\frac{3\mathbf{NL}^{n}-\mathbf{NL}^{n-1}}{2}=-\boldsymbol{\nabla}p^{n+1/2}+\frac{\mu}{2}\nabla^{2}(\mathbf{u}^{n+1}+\mathbf{u}^{n})+\frac{\mathbf{F}^{n+1}+\mathbf{F}^{n}}{2}, \tag{25}$$

where Δt is the time step, the superscript n denotes time $t = n\Delta t$, and $\mathbf{NL}$ refers to the nonlinear advection term. Equations (16)-(19) are discretized spatially using standard, second-order, finite volume methods on a uniform staggered grid [26]. The pressure coupling is approximated using the projection method of Bell *et al.* [25].

Staggered grids can require special treatment when spatially discretizing the nonlinear advection term ($\mathbf{NL}$) in Eqn. (25) near an immersed boundary. Consider the section of staggered grid in Fig. 19(a), where the temperature and pressure are stored at cell centroids (solid dots), while the velocity components u and v are stored at the cell faces (squares and triangles, respectively). The shaded region represents the control volume used to discretize the x-component

of the Navier–Stokes equations. Following Ferziger and Perić [26], the discretization of **NL** at node P depends on the u-values at nodes P, E, W, N, and S, as well as the v-values at nodes ne, nw, se, and sw. However, a complication arises near an immersed surface, as shown in Fig. 19(b). Here, node P is a normal fluid point, but node ne lies in the solid (shaded area bounded by the dashed line) and has no physical value for v. To compute **NL** at node P, we approximate v_{ne} using linear extrapolation from neighbouring fluid nodes along a grid line. A similar technique was employed by Yang and Balaras [35], who extended the velocity and pressure fields into a layer of nodes in the solid domain.

We verify spatial accuracy by setting $\omega = 0$ and integrating from the initial condition $\mathbf{u} = p = T_A = T_B = 0$ to steady-state using N finite volume cells in each direction. We then measure the relative error of each flow field using the infinity norm (ERR_∞) defined in Eqn. (15). The pressure error is not computed in cells with forcing points. We set all thermophysical properties to unity. Figure 20(a) shows ERR_∞ versus N for u (asterisks), v (circles), p (squares), and T (solid dots). The dashed and dash-dotted lines show $1/N^2$ and $1/N$, respectively. We observe second-order spatial accuracy for u, v, and T, and first-order accuracy for p. Note that the reduced accuracy of the pressure field is a documented issue with projection methods, even in simulations without immersed boundaries [36–40], and is not related to our proposed IB method. In our experience, the accuracy of the pressure field produced by the projection method of Bell *et al.* [25] is problem dependent, and even in simulations without immersed surfaces, typically varies between roughly first to second-order. Though projection methods are used extensively in the immersed boundary literature, studies often do not report the order of accuracy of the pressure field. We consequently note here that all remaining simulations in this section produced first-order spatial accuracy for pressure, similar to that in Fig. 20(a), and are not shown for brevity.

We verify the temporal accuracy by setting $\omega = 4\pi$ and integrating from $t = 0$ to $t = 1$ using exact initial conditions. The error ERR_∞ of each flow field is then measured at time $t = 1$. Fig. 20(b) shows the resulting ERR_∞ versus Δt.

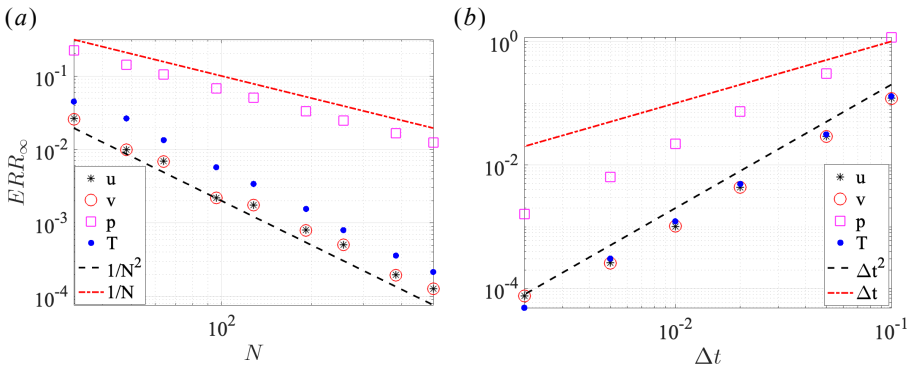

Figure 20: (a) Spatial error plot and (b) temporal error plot. Velocity components u and v, pressure p, and temperature T shown in asterisks, circles, squares, and solid dots, respectively.

The dashed and dash-dotted lines show Δt^2 and Δt, respectively. We observe second-order accuracy for u, v, and T, and nearly second-order accuracy for p.

4.2. Circular Couette flow

As further validation, we consider a 2D circular Couette flow with conjugate heat transfer through the inner cylinder, as sketched in Fig. 18(b). A stationary solid (phase B) occupies the annular region $R_i \leq r < R_\Gamma$. A fluid (phase A) occupies the annular region $R_\Gamma \leq r \leq R_o$. The temperature field is subjected to the following Dirichlet conditions,

$$T^{n+1}\Big|_{r=R_i} = T_i, \qquad T^{n+1}\Big|_{r=R_o} = T_o. \tag{26}$$

At $r = R_\Gamma$ we apply the continuity of temperature and heat flux

$$T_A^{n+1}\Big|_{r=R_\Gamma} = T_B^{n+1}\Big|_{r=R_\Gamma}, \qquad k_A\Big(\mathbf{n}\cdot\boldsymbol{\nabla}T_A\Big)_{r=R_\Gamma}^{n+1} = k_B\Big(\mathbf{n}\cdot\boldsymbol{\nabla}T_B\Big)_{r=R_\Gamma}^{n+1}. \tag{27}$$

The outer cylinder ($r = R_o$) rotates with a steady angular velocity $\dot{\theta}$, subjecting the velocity field to the Dirichlet conditions

$$u_r^{n+1}\Big|_{r=R_\Gamma,R_o} = 0, \qquad u_\theta^{n+1}\Big|_{r=R_\Gamma} = 0, \qquad u_\theta^{n+1}\Big|_{r=R_o} = \dot{\theta}R_o, \tag{28}$$

where u_r and u_θ are the velocity components in the r and θ directions, respectively. Note that the immersed conditions (26)-(28) are all discretized implicitly in time. Analytical solutions for $\mathbf{u}$, p, and T are provided in Appendix E.

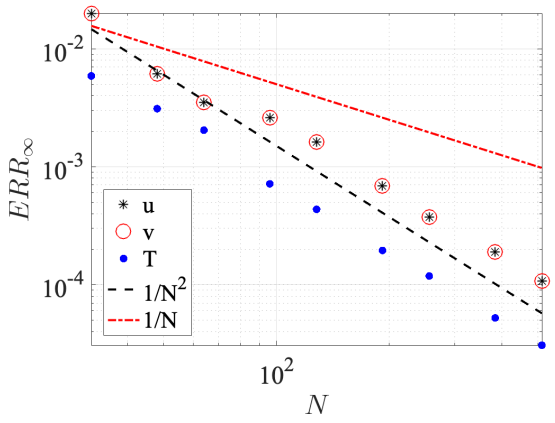

Figure 21: Error plot for the circular Couette flow. Velocity components u and v, and temperature T shown in asterisks, circles, and solid dots, respectively.

We simulate the problem using the square computational domain $x \in [-L/2, L/2]$, $y \in [-L/2, L/2]$, as shown in Fig. 18(b). The cylinders are centered at $(x, y) = (0, 0)$. On the outer boundary, we apply

$$\mathbf{u}\Big|_{x=\pm L/2} = \mathbf{u}\Big|_{y=\pm L_y/2} = 0, \qquad T\Big|_{x=\pm L/2} = T\Big|_{y=\pm L/2} = T_o. \tag{29}$$

We set all thermophysical properties to unity, except for the thermal conductivity of the solid, which is set to $k_B = 120$. The dimensions of the system are set to $L = 2\pi$, $R_i = 0.45$, $R_\Gamma = 1.2$, and $R_o = 2.4$. We also set $T_i = 200$, $T_o = 0$, and $\dot{\theta} = 1$. We initialize the simulations with $\mathbf{u} = 0$, $T = 0$, $p = 0$, and integrate in time to steady-state. The error ERR_∞ is then computed for each field in the physical domain $R_i \leq r \leq R_o$. Figure 21 shows ERR_∞ versus N. We observe second-order accuracy for u (asterisks), v (circles), and T (solid dots).

4.3. Moving solids

Though not the focus of our study, here we demonstrate the ability of our method to simulate moving solids. For that, we consider fluid flow and conjugate heat transport in the rotating Couette flows shown in Fig. 22. In panel (a), fluid occupies the region labelled Ω_A, between an outer circular cylinder of radius R_o and an inner co-axial elliptic cylinder with semi-major and semi-minor axes l_1

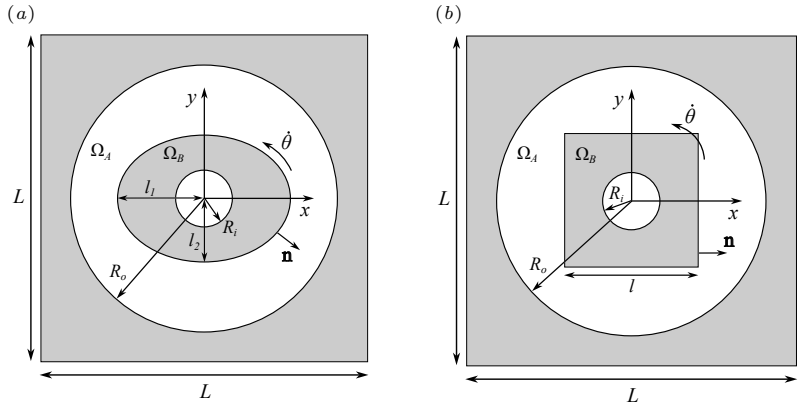

Figure 22: A rotating Couette flow where the inner cylinder is replaced with a (a) rotating ellipse of size l_1 by l_2, or (b) rotating square of size l by l. The fluid and the rotating inner cylinder occupy regions Ω_A, and Ω_B, respectively. A constant temperature difference is applied between $r = R_i$ and $R = R_o$, driving conjugate heat transfer through the inner cylinder and fluid. The outer cylinder is fixed, and immersed in a domain of L by L.

and l_2, respectively. The outer cylinder is stationary, while the elliptic cylinder rotates about its axis $(x = 0, y = 0)$ with constant angular velocity $\dot{\theta}$. On the outer cylinder we apply $\mathbf{u} = 0$. On the inner cylinder, we apply

$$u_r^{n+1}\Big|_{r=R_e} = 0, \qquad u_\theta^{n+1}\Big|_{r=R_e} = \dot{\theta} R_e, \tag{30}$$

where R_e refers to the radial distance from $(x, y) = (0, 0)$ to the local surface of the elliptic cylinder. As in section 4.2, we drive conjugate heat transport through the fluid and inner cylinder by applying the fixed temperature conditions (26) at $r = R_i$ and $r = R_0$. On the surface of the rotating ellipse, we apply the continuity of temperature and heat flux, as in Eqn. (27).

In addition to the flow in Fig. 22(a), we also consider that in Fig. 22(b), where the ellipse is replaced with a square of size l by l, rotating about its axis $(x = 0, y = 0)$ with angular velocity $\dot{\theta}$. For both cases in Fig. 22, we set $L = 2\pi$, $R_i = 0.45$, and $R_o = 2.4$. The size of the ellipse is $l_1 = 1.2$, $l_2 = 1.0$, and the size of the square is $l = 2.12$. To eliminate potential Case 4 problem points on the square, we apply a smoothing of $\delta = 2\Delta x$, where Δx is the cell size

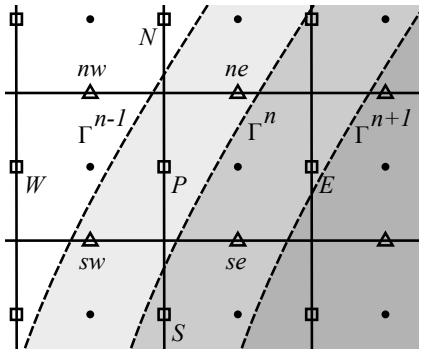

Figure 23: A staggered finite volume grid, and surface Γ (dashed line) moving to the right at three consecutive time levels. At time t^{n-1} point P is a solid point, at t^n a forcing point, and at t^{n+1} a fluid point.

of the uniform grid. All thermophysical properties are set to unity, except for $k_B = 120$. We also set $T_i = 200$, $T_o = 0$, and $\dot{\theta} = \pi/4$.

Simulating a moving solid raises a new challenge when grid points change their phase from solid to fluid. These grid points, sometimes called "freshly cleared cells" [41, 42], do not have the required time history to compute terms such as $\mathbf{NL}^{n-1}$ in the semidiscrete equation (25). This is a common issue in sharp interface IB methods, including the cut-cell [43], direct forcing [35], and ghost-cell methods [42]. For demonstration, Fig. 23 illustrates a section of a staggered finite volume grid. The dashed lines show an interface Γ moving to the right at three consecutive time levels t^{n-1}, t^n, and t^{n+1}. At time $t = t^{n-1}$, the u-velocity node labelled P in Fig. 23 lies in the solid (shaded grey), and does not have a physically meaningful velocity. At $t = t^n$, the surface Γ moves to the right, and node P becomes a forcing point in the fluid. Because we apply the immersed boundary conditions implicitly in time, node P requires no special treatment at this time step. However, at $t = t^{n+1}$, node P becomes a regular fluid point, and we must apply the discretized momentum Eqn. (25). This raises issues when computing the explicit terms $\mathbf{NL}^{n-1}$, $\nabla^2\mathbf{u}^n$, and $\boldsymbol{\nabla} p^n$.

Because node P lies in the solid at time t^{n-1}, we discretize advection terms in the momentum and energy equations using the forward Euler method on

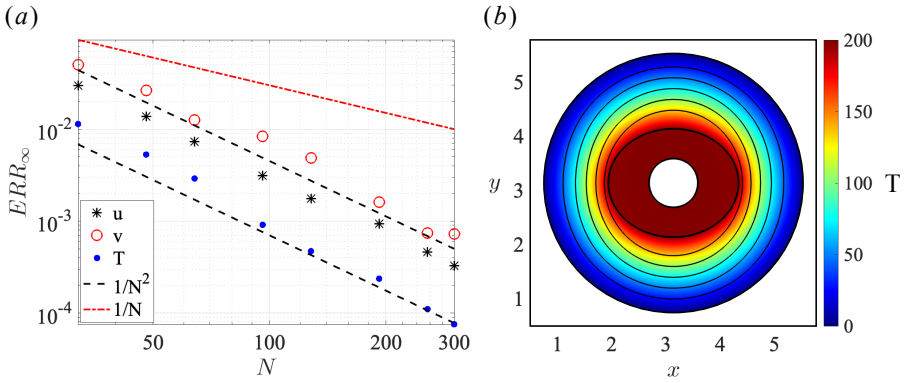

Figure 24: (a) Errorplot for the rotating ellipse. Velocity components u and v, and temperature T are shown using asterisks, circles, and solid dots, respectively. (b) Temperature contour plot with streamlines.

freshly cleared cells, as suggested by Udaykumar *et al.* [41]. This avoids the computation of $\mathbf{NL}^{n-1}$. For the case shown in Fig. 23, the approximation of $\nabla^2\mathbf{u}^n$, and $\boldsymbol{\nabla}p^n$ require u_E^n and p_E^n, which both lie in the solid. We address this issue as previously illustrated in Fig. 19, using the field extension method of Yang and Balaras [35]. Finally, we restrict the time step to ensure that no grid point transitions from a solid point to a regular fluid point in a single time step. Such a scenario would result in the absence of velocity data at t^n. This time step restriction is common in sharp IB methods [8, 35, 44].

Because there is no analytical solution for the flows in Fig. 22, we establish reference solutions by simulating the flows using a fine grid ($N = 600$) with respect to a coordinate system that rotates with constant angular velocity $\dot{\theta}$. In this rotating coordinate system, the ellipse and square in Fig. 22 are stationary, while the outer cylinder rotates with an angular velocity $-\dot{\theta}$. Note that Eqns. (16)-(19) must also be modified in the non-inertial rotating frame [45], as detailed in Appendix F. The reference simulations are initialized with $\mathbf{u} = 0$, $T = T_i$, and $p = 0$, and are integrated in time until they reach steady-state, which occurs within a full revolution.

To evaluate the spatial accuracy of our method, we run a series of simulations on grids with $N \leq 320$ in the stationary coordinate system. All simulations are

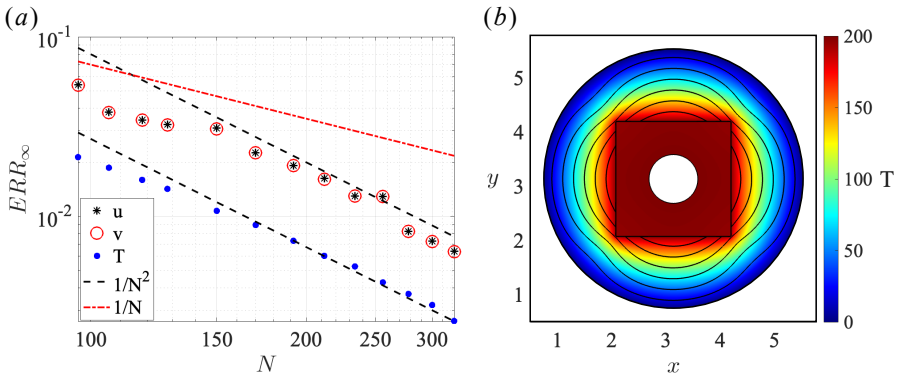

Figure 25: (a) Errorplot for the rotating square. Velocity components u and v, and temperature T are shown using asterisks, circles, and solid dots, respectively. (b) Temperature contour plot with streamlines.

initialized with $\mathbf{u} = 0$, $T = T_i$, and $p = 0$, and integrated in time until the ellipse or square completes one revolution. We then use bicubic interpolation to approximate the velocity and temperature at 40 points equally distributed along a circle of radius $r = 1.8$, centered at $(x = 0, y = 0)$. ERR_∞ is computed by comparing results obtained in the stationary coordinate system to the reference solution in the rotating coordinate system.

Figure 24(a) shows the error plots for the ellipse. The results demonstrate second-order accuracy for u (asterisks), v (circles), and T (solid dots). Figure 24(b) shows the temperature contour plot overlaid with streamlines. Figure 25(a) shows the corresponding error plots for the square. We again observe second-order accuracy for both velocity components and temperature for $N \geq 150$. The reduced accuracy for $N \leq 150$ likely arises due to the strong impact of smoothing on coarse grids. Figure 25(b) shows the temperature contour plot overlaid with streamlines. Note that from the streamlines in Figs. 24(b) and 25(b) that the rotating ellipse and square did not generate vortical structures. This is because the Reynolds number for these simulations, $Re = \rho U R_o/\mu$, where $U = R_o\dot{\theta}$ is only 4.5.

5. Summary and Conclusion

In this work, we developed a direct forcing IB method that simulates general discontinuous Dirichlet and Robin boundary conditions. Our method is built on the concept of a forcing pair, defined as two grid points that are adjacent to each other, but on opposite sides of an interface. For 2D problems, we can simultaneously enforce discontinuous Dirichlet and Robin conditions using a six-point stencil at one of the forcing points, and a 12-point stencil at the other. In comparison, prior work requires up-to 14-point stencils at both points. We identified four cases (Cases 1-4) that require special treatment. Cases 1, and 2 can arise for any surface geometry, whereas Cases 3 and 4 tend to arise only on surfaces with corners. Case 4 occurs when an extrapolation stencil cannot be formed in a sharp corner, requiring surface modification. We proposed two surface modification methods (cutting and smoothing) to eliminate Case 4 points. We verified the spatial accuracy of our method by solving the Poisson equation for eight geometries using a manufactured solution with discontinuous Dirichlet, Neumann, and Robin conditions. We observed second-order spatial accuracy for all cases except three, which nevertheless had spatial accuracies above 1.8. In some cases, we also observed that the cutting method produces cloudier error plots than smoothing. We also explored the performance of our IB method for simulating fluid flows with conjugate heat transport over fixed and moving solids. For that, we coupled our method to the incompressible Navier-Stokes and continuity equations. First, we verified the spatial-temporal accuracy of the solver using manufactured solutions and an analytical solution for circular Couette flow with conjugate heat transfer. We observed second-order spatial and temporal accuracies for velocity and temperature. For pressure, we observed first-order spatial, and nearly second-order temporal accuracy, which is consistent with the projection method [36] used in our discretization of the Navier-Stokes and continuity equations. Finally, we simulated fluid flow and conjugate heat transport between a stationary cylinder and a rotating ellipse or square. Because no analytical solution for flow around a rotating ellipse or

square exists, we used results obtained in a non-inertial rotating frame as a reference solution. Second-order spatial accuracy was observed for velocity and temperature, and first-order for pressure.

Ongoing work focuses on implementing our method in a parallelized 3D CFD algorithm, with the intent of modelling conjugate heat transport in heat exchangers and porous media. Our longer term objective is to extend our method to moving particles and explore applications to active colloids and packed bed heat exchangers.

6. Acknowledgments

This work was generously supported by the National Science Foundation under grant numbers 2306329 (N. Tilton), DMR-2314339 (N. Wu), and OAC-1931368, CBET CAREER 2234387, and CBET 2407938 (A.P.S. Bhalla). The authors also thank Dr. Elias Balaras (George Washington University), who suggested our treatment of freshly cleared cells in section 4.3.

Appendix A. Extrapolation coefficients for Dirichlet conditions

Consider the example shown in Fig. 4. The extrapolation coefficients a_1, a_2, and a_3 in Eqn. (8) can be derived by fitting a one-dimensional, second-order Lagrange polynomial to nodes A_1, A_2, and A_3. If the extrapolation nodes lie along a horizontal grid line, the coefficients become

$$a_1 = \frac{(x_s - x_2)(x_s - x_3)}{(x_1 - x_2)(x_1 - x_3)}, \qquad a_2 = \frac{(x_s - x_1)(x_s - x_3)}{(x_2 - x_1)(x_2 - x_3)}, \tag{A.1}$$

$$a_3 = \frac{(x_s - x_1)(x_s - x_2)}{(x_3 - x_1)(x_3 - x_2)}, \tag{A.2}$$

where x_1, x_2, x_3 are the x-coordinates of nodes A_1, A_2, and A_3, respectively. Coefficients b_i are similarly computed using the nodes B_1, B_2 and B_3.

Appendix B. Matrix method for Robin conditions

When the preferred extrapolation stencil is not available, we compute the coefficients for the discretized Robin condition (4) as follows. Consider the term

$$\Big[\alpha_A T_A + \beta_A(\boldsymbol{\nabla} T_A \cdot \mathbf{n})\Big]_\Gamma \tag{B.1}$$

in Eqn. (4). Assuming a six-point extrapolation stencil, we approximate (B.1) as

$$\Big[\alpha_A T_A + \beta_A(\boldsymbol{\nabla} T_A \cdot \mathbf{n})\Big]_\Gamma = \sum_{j=1}^{6} \widehat{c}_j T_{Aj} \tag{B.2}$$

where $\widehat{c}_j$ are coefficients, and T_{Aj} are the temperatures at the six nodes. To determine $\widehat{c}_j$, we expand the temperatures T_{Aj} in a 2D Taylor series about $\mathbf{x_s}$,

$$T_{Aj} = T_A\Big|_{\mathbf{x_s}} + \Delta x_j \frac{\partial T_A}{\partial x}\bigg|_{\mathbf{x_s}} + \Delta y_j \frac{\partial T_A}{\partial y}\bigg|_{\mathbf{x_s}} + \frac{\Delta x_j^2}{2}\frac{\partial^2 T_A}{\partial x^2}\bigg|_{\mathbf{x_s}} + \ldots \\ \frac{\Delta y_j^2}{2}\frac{\partial^2 T_A}{\partial y^2}\bigg|_{\mathbf{x_s}} + \frac{\Delta y_j \Delta x_j}{2}\frac{\partial^2 T_A}{\partial y \partial x}\bigg|_{\mathbf{x_s}}, \tag{B.3}$$

where $\Delta x_j = x_s - x_{Aj}$, and $\Delta y_j = y_s - y_{Aj}$. Substituting these expansions in Eqn. (B.2) and comparing coefficients, one can show that a second-order approximation requires

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ \Delta x_1 & \Delta x_2 & \Delta x_3 & \Delta x_4 & \Delta x_5 & \Delta x_6 \\ \Delta y_1 & \Delta y_2 & \Delta y_3 & \Delta y_4 & \Delta y_5 & \Delta y_6 \\ \Delta x_1^2 & \Delta x_2^2 & \Delta x_3^2 & \Delta x_4^2 & \Delta x_5^2 & \Delta x_6^2 \\ \Delta y_1^2 & \Delta y_2^2 & \Delta y_3^2 & \Delta y_4^2 & \Delta y_5^2 & \Delta y_6^2 \\ \Delta y_1 \Delta x_1 & \Delta y_2 \Delta x_2 & \Delta y_3 \Delta x_3 & \Delta y_4 \Delta x_4 & \Delta y_5 \Delta x_5 & \Delta y_6 \Delta x_6 \end{bmatrix} \begin{bmatrix} \widehat{c}_1 \\ \widehat{c}_2 \\ \widehat{c}_3 \\ \widehat{c}_4 \\ \widehat{c}_5 \\ \widehat{c}_6 \end{bmatrix} = \begin{bmatrix} \alpha_A \\ \beta_A n_x \\ \beta_A n_y \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{B.4}$$

Note that the coefficients $\widehat{c}_j$ obtained using this approach are input directly into the global matrix, and are therefore not equivalent to the coefficients c_j given in Eqn. (11). However, the approach shown here is equivalent to the procedure in Section 2.2, where we approximate T_A in the vicinity of Γ as a biquadratic polynomial (10).

Appendix C. Practical implementation in MATLAB

Here we summarize the algorithm we coded in MATLAB to solve the Poisson problem (1)-(5). The algorithm is used in Section 3 to explore the spatial accuracy of our method. For a given grid with a total of $M = (N_x + 2)(N_y + 2)$ grid points, the algorithm builds a sparse matrix problem $\mathbf{AT} = \mathbf{b}$, where $\mathbf{T}$ is a M x 1 vector containing the unknown temperatures at the grid nodes, $\mathbf{b}$ is a M x 1 vector containing the forcing terms (*e.g.* $f_A,\ f_B,\ g,\ h$), and $\mathbf{A}$ is a M x M matrix containing the spatial discretization of the differential operators in (1)-(5).

To identify forcing points and compute the normal $\mathbf{n}$ to the interface, we use the signed distance function $\phi(\mathbf{x})$, defined as positive in phase A and negative in phase B. For example, if phase B is a circle of radius R centered at $(x, y) = (x_c, y_c)$, as in Fig. 12(b), the distance function is given by

$$\phi(x, y) = \sqrt{(x - x_c)^2 + (y - y_c)^2} - R. \tag{C.1}$$

Consider the grid point labelled P in Fig. 1(b). The point is defined a non-forcing point if ϕ_P is non-zero and has the same sign as ϕ_E, ϕ_N, ϕ_W, ϕ_S, where the subscripts denote ϕ evaluated at the points labeled in Fig. 1(b). We build the corresponding row of $\mathbf{A}$ and $\mathbf{b}$ by applying the Poisson equation (1) if $\phi_P > 0$ and (2) if $\phi_P < 0$. For ghost nodes, we apply boundary condition (3).

Any grid point for which $\phi_P = 0$, or for which ϕ_P has a different sign than any of its four neighbours, is a forcing point. In that case, the computation of the corresponding rows in $\mathbf{A}$ and $\mathbf{b}$ depend on the procedure for Case 4 points.

Smoothing: In the case of smoothing, we compute $\phi(\mathbf{x})$ using the method detailed in Section 2.3.4, prior to defining forcing and non-forcing points. We set the smoothing factor to $\delta = \gamma \Delta x_{max}$, where Δx_{max} is the maximum cell width in the domain. As the algorithm loops through the grid, defining forcing and non-forcing points, it defines a forcing point as a problem point if the point has three potential partners that can only pair with the forcing point, as in

Fig. 10(b). If a problem point is found, the algorithm terminates and displays a warning to increase γ. Though this process was sufficient for our purposes, the algorithm could be automated to find the minimum required smoothing.

If no problem points are found, the algorithm loops through all forcing points searching for points with two potential partners. If found, pairs are formed based on the surface normal, as detailed in section 2.3.2. The surface normal, $\mathbf{n}$ is computed from the signed distance function using centered differencing. When a forcing pair is formed, both grid points are flagged so that they cannot be used to form another pair. After all forcing points with two potential partners have been identified and paired, the algorithm then loops through the remaining forcing points to either pair them, or label them as orphan points.

The algorithm then loops through every forcing pair, building extrapolation stencils as detailed in Section 2.2, and computing the extrapolation coefficients, and forcing terms (g, h) for the Dirichlet and Robin conditions. When seeking an extrapolation stencil for the Robin condition, the algorithm first checks if the preferred stencil is available. If not, the algorithm searches for other six-point stencils where the points A_2 and A_3 lie on the horizontal grid line passing through A_1, while A_4 and A_5 lie on the vertical grid line passing through A_1, as labelled in Fig. 5. The extrapolation coefficients for these stencils can be computed analytically. If such stencils are not found, the algorithm seeks any six-point stencil containing three points with a unique x-coordinate, three points with unique a y-coordinate, and a maximum of three points along any one grid line. If found, the extrapolation coefficients can be computed using the method detailed in Appendix B. If a six point-stencil is not found, the algorithm uses a Case 3 alternate stencil. If a problem point with no suitable extrapolation stencil is encountered, as in Fig. 10(a), the algorithm terminates and displays a warning to refine the grid. Once the extrapolation coefficients and forcing terms are computed for every forcing pair and orphan point, they are input into the corresponding rows of $\mathbf{A}$ and $\mathbf{b}$. The matrix is then solved using MATLAB's sparse direct solver.

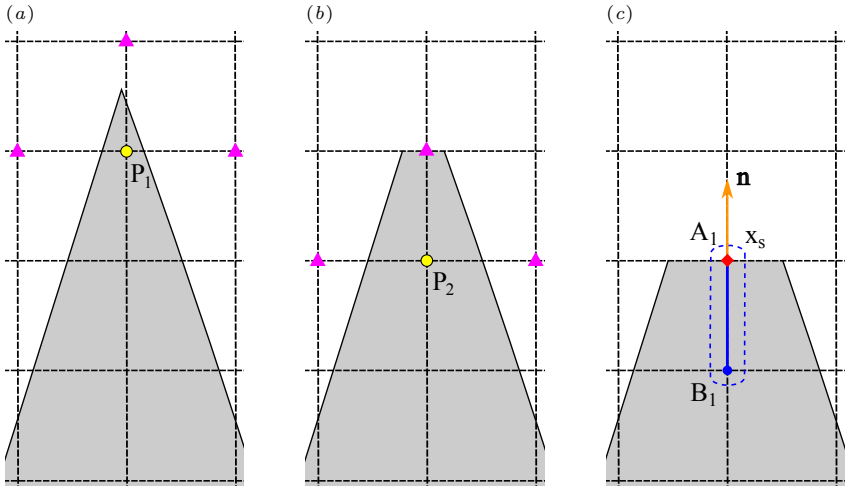

Figure C.26: (a) Problem point P_1 requiring cutting. (b) Surface after cutting along the grid line passing through P_1 and its two potential partners. Cutting created a new problem point P_2 that also requires cutting. (c) Surface after cutting along grid line passing through P_2 and its two potential partners. Point A_1 is not a problem point, and is paired with B_1.

Cutting: Using the unsmoothed distance function, the cutting method loops through all grid points and labels them as forcing or non-forcing points. When the algorithm encounters a problem point with three neighbours in the opposite phase that can only be paired with the problem point, it cuts the surface along the grid line passing through the problem point and its two potential partners. For the case shown in panels (b) and (c) of Fig. 10, the problem point P is treated as though it belongs to phase A, and paired with point B_1. Note from Fig. 10(c) that after cutting, point B_1 now has three neighbours in phase A. However, it is not a problem point, because the neighbour to the right can be paired with a different partner. The point to the left of B_1 can be treated as an orphan. In some cases, however, a problem point requires two cuts. An example is shown in Fig. C.26, where point P_1 in panel (a) is a problem point that requires cutting. Panel (b) shows the surface after cutting along the grid line passing through P_1 and two of its potential partners. In this case, cutting through P_1 created a new problem point, labelled P_2 in panel (b), that requires

cutting again. Panel (c) shows the surface after cutting through the grid line passing through P_2 and its two potential partners. As a result, point A_1 is not a problem point and is paired with B_1.

All forcing pairs formed through cutting are flagged so that the algorithm treats them as Case 4 pairs when computing the extrapolation coefficients. No changes are made to the signed distance function to account for the cut. After all problem points are paired, the algorithm follows the same steps as the smoothing method.

Appendix D. Non-uniform grid

Our non-uniform grid places the vertical cell faces at the x-locations

$$x_j = \pi\left[1 + \cos\left(\frac{\pi j}{N}\right)\right], \qquad j = 0, 1, 2, ..., N. \tag{D.1}$$

We similarly place the horizontal cell faces at the y-locations

$$y_j = \pi\left[1 + \cos\left(\frac{\pi j}{N}\right)\right], \qquad j = 0, 1, 2, ..., N. \tag{D.2}$$

Appendix E. Analytical solution for circular Couette flow with conjugate heat transfer

The circular Couette flow in Fig. 18(b) has the analytical solution,

$$u_r = 0 \qquad \text{for all } r, \tag{E.1}$$

$$u_\theta(r) = \begin{cases} C_1 r + \frac{C_2}{r} & \text{for } r \in \Omega_A, \\ 0 & \text{otherwise,} \end{cases} \tag{E.2}$$

$$p(r) = \begin{cases} C_1^2 \frac{r^2}{2} + 2C_1 C_2 \ln(r) - \frac{C_2^2}{2r^2} & \text{for } r \in \Omega_A, \\ 0 & \text{otherwise,} \end{cases} \tag{E.3}$$

$$T(r) = \begin{cases} T_i + \dfrac{T_o - T_i}{\ln\left(\frac{R_\Gamma}{R_i}\right) + \frac{k_A}{k_B}\ln\left(\frac{R_o}{R_\Gamma}\right)} \ln\left(\dfrac{r}{R_i}\right) & \text{for } r \in \Omega_B, \\ T_o - \dfrac{T_o - T_i}{\frac{k_A}{k_B}\ln\left(\frac{R_\Gamma}{R_i}\right) + \ln\left(\frac{R_o}{R_\Gamma}\right)} \ln\left(\dfrac{R_o}{r}\right) & \text{for } r \in \Omega_f, \end{cases} \tag{E.4}$$

where

$$C_1 = \frac{\dot{\theta} R_o^2}{R_o^2 - R_\Gamma^2}, \qquad C_2 = \frac{\dot{\theta} R_\Gamma^2 R_o^2}{R_m^2 - R_o^2}. \tag{E.5}$$

Appendix F. Governing equations in non-inertial rotating frame

Modelling Eqns. (16)-(19) in a non-inertial rotating coordinate frame requires the addition of centripetal and centrifugal terms to the Navier-Stokes equation [45]. The heat equation in the fluid (phase A) remains unchanged, but requires an additional term in the solid (phase B). The governing equations in the rotating coordinate system are

$$\boldsymbol{\nabla} \cdot \mathbf{u}_{rot} = 0, \tag{F.1}$$

$$\rho_A \left[\frac{\partial \mathbf{u}_{rot}}{\partial t} + (\mathbf{u}_{rot} \cdot \boldsymbol{\nabla}) \mathbf{u}_{rot} \right] = -\boldsymbol{\nabla} p + \mu_A \nabla^2 \mathbf{u}_{rot} + \mathbf{F}_{cor} + \mathbf{F}_{cf}, \tag{F.2}$$

$$\rho_A c_A \left[\frac{\partial T_A}{\partial t} + (\mathbf{u}_{rot} \cdot \boldsymbol{\nabla}) T_A \right] = k_A \nabla^2 T, \tag{F.3}$$

$$\rho_B c_B \frac{\partial T_B}{\partial t} = k_B \nabla^2 T + \mathbf{F}_B, \tag{F.4}$$

where $\mathbf{u}_{rot}$, $\mathbf{F}_{cor}$ and $\mathbf{F}_{cf}$ denote the velocity in the rotating frame, coriolis force, and centrifugal force, respectively. The extra term $\mathbf{F}_B$ in Eqn. (F.4) arises from transforming the temporal derivative of T_B from stationary to rotating frame [46]. The additional terms are defined as

$$\mathbf{F}_{cor} = 2\rho_A \mathbf{r} \times \dot{\theta}, \tag{F.5}$$

$$\mathbf{F}_{cf} = \rho_A (\dot{\theta} \times \mathbf{r}) \times \dot{\theta}, \tag{F.6}$$

$$\mathbf{F}_B = (\dot{\theta} \times \mathbf{r}) \cdot \boldsymbol{\nabla} T, \tag{F.7}$$

and are discretized explicitly in time. The boundary conditions for velocity in the rotating frame can be written as

$$u_{rot,r}^{n+1}\Big|_{r=R_s,R_o} = 0, \qquad u_{rot,\theta}^{n+1}\Big|_{r=R_s} = 0, \qquad u_{rot,\theta}^{n+1}\Big|_{r=R_o} = -\dot{\theta} R_o, \tag{F.8}$$

whereas the boundary conditions for temperature remain unchanged.

References

[1] T. Alam, M.-H. Kim, A comprehensive review on single phase heat transfer enhancement techniques in heat exchanger applications, Renewable and Sustainable Energy Reviews 81 (2018) 813–839. doi:10.1016/j.rser.2017.08.060.

[2] S. Das, N. G. Deen, J. A. M. Kuipers, Direct numerical simulation for flow and heat transfer through random open-cell solid foams: Development of an IBM based CFD model, Catalysis Today 273 (2016) 140–150. doi:10.1016/j.cattod.2016.03.048.

[3] S. Das, A. Panda, N. G. Deen, J. A. M. Kuipers, A sharp-interface immersed boundary method to simulate convective and conjugate heat transfer through highly complex periodic porous structures, Chemical Engineering Science 191 (2018) 1–18. doi:10.1016/j.ces.2018.04.061.

[4] G. Iaccarino, S. Moreau, Natural and forced conjugate heat transfer in complex geometries on Cartesian adapted grids, Journal of Fluids Engineering 128 (4) (2005) 838–846. doi:10.1115/1.2201625.

[5] Z. Yu, X. Shao, A. Wachs, A fictitious domain method for particulate flows with heat transfer, Journal of Computational Physics 217 (2) (2006) 424–452. doi:10.1016/j.jcp.2006.01.016.

[6] W. D. Ristenpart, I. A. Aksay, D. A. Saville, Electrohydrodynamic flow around a colloidal particle near an electrode with an oscillating potential, Journal of Fluid Mechanics 575 (2007) 83–109. doi:10.1017/S0022112006004368.

[7] K. Ling, W.-Q. Tao, A sharp-interface model coupling VOSET and IBM for simulations on melting and solidification, Computers & Fluids 178 (2019) 113–131. doi:10.1016/j.compfluid.2018.08.027.

[8] R. Mittal, G. Iaccarino, Immersed boundary methods, Annual Review of Fluid Mechanics 37 (1) (2005) 239–261. doi:10.1146/annurev.fluid.37.061903.175743.

[9] C. S. Peskin, Flow patterns around heart valves: A numerical method, Journal of Computational Physics 10 (2) (1972) 252–271. doi:10.1016/0021-9991(72)90065-4.

[10] R. P. Beyer, R. J. LeVeque, Analysis of a one-dimensional model for the immersed boundary method, SIAM Journal on Numerical Analysis 29 (2) (1992) 332–364. doi:10.1137/0729022.

[11] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, Journal of Computational Physics 105 (2) (1993) 354–366. doi:10.1006/jcph.1993.1081.

[12] A. P. S. Bhalla, R. Bale, B. E. Griffith, N. A. Patankar, A unified mathematical framework and an adaptive numerical method for fluid–structure interaction with rigid, deforming, and elastic bodies, Journal of Computational Physics 250 (2013) 446–476. doi:10.1016/j.jcp.2013.04.033.

[13] F. B. Usabiaga, B. Kallemov, B. Delmotte, A. P. S. Bhalla, B. E. Griffith, A. Donev, Hydrodynamics of suspensions of passive and active rigid particles: A rigid multiblob approach, Communications in Applied Mathematics and Computational Science 11 (2) (2016) 217–296. arXiv:1602.02170, doi:10.2140/camcos.2016.11.217.

[14] J. Mohd-Yusof, Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries, Annual Research Briefs 1 (1) (1997) 317–327.

[15] E. A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, Journal of Computational Physics 161 (1) (2000) 35–60. doi:10.1006/jcph.2000.6484.

[16] H. S. Udaykumar, W. Shyy, M. M. Rao, Elafint: A mixed Eulerian–Lagrangian method for fluid flows with complex and moving boundaries, International Journal for Numerical Methods in Fluids 22 (8) (1996) 691–712. doi:10.1002/(SICI)1097-0363(19960430)22:8¡691::AID-FLD371¿3.0.CO;2-U.

[17] T. Ye, R. Mittal, H. S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, Journal of Computational Physics 156 (2) (1999) 209–240. doi:10.1006/jcph.1999.6356.

[18] M.-H. Chung, An adaptive Cartesian cut-cell method for conjugate heat transfer on arbitrarily moving fluid-solid interfaces, Computers & Fluids 178 (2019) 56–72. doi:10.1016/j.compfluid.2018.09.013.

[19] 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. arXiv:2158113.

[20] M. Berger, A note on the stability of cut cells and cell merging, Applied Numerical Mathematics 96 (2015) 180–186. doi:10.1016/j.apnum.2015.05.003.

[21] S. Kang, G. Iaccarino, F. Ham, DNS of buoyancy-dominated turbulent flows on a bluff body using the immersed boundary method, Journal of Computational Physics 228 (9) (2009) 3189–3208. doi:10.1016/j.jcp.2008.12.037.

[22] K. Nagendra, D. K. Tafti, K. Viswanath, A new approach for conjugate heat transfer problems using immersed boundary method for curvilinear grid based solvers, Journal of Computational Physics 267 (2014) 225–246. doi:10.1016/j.jcp.2014.02.045.

[23] 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.

[24] S. Ianniello, GFMxP: A novel ghost fluid method to solve the variable coefficient Poisson equation, International Journal of Multiphase Flow 170 (2024) 104634. doi:10.1016/j.ijmultiphaseflow.2023.104634.

[25] J. B. Bell, P. Colella, H. M. Glaz, A second-order projection method for the incompressible Navier-Stokes equations, Journal of Computational Physics 85 (2) (1989) 257–283. doi:10.1016/0021-9991(89)90151-4.

[26] J. H. Ferziger, M. Perić, Computational Methods for Fluid Dynamics, 3rd Edition, Springer, 2002.

[27] E. Balaras, Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations, Computers & Fluids 33 (3) (2004) 375–404. doi:10.1016/S0045-7930(03)00058-6.

[28] K. Luo, Z. Zhuang, J. Fan, N. E. L. Haugen, A ghost-cell immersed boundary method for simulations of heat transfer in compressible flows under different boundary conditions, International Journal of Heat and Mass Transfer 92 (2016) 708–717. doi:10.1016/j.ijheatmasstransfer.2015.09.024.

[29] M. Yousefzadeh, I. Battiato, High order ghost-cell immersed boundary method for generalized boundary conditions, International Journal of Heat and Mass Transfer 137 (2019) 585–598. doi:10.1016/j.ijheatmasstransfer.2019.03.061.

[30] J. Lou, J. Johnston, N. Tilton, Application of projection and immersed boundary methods to simulating heat and mass transport in membrane distillation, Computers & Fluids 212 (2020) 104711. doi:10.1016/j.compfluid.2020.104711.

[31] J. Finn, S. V. Apte, Relative performance of body fitted and fictitious domain simulations of flow through fixed packed beds of spheres, International Journal of Multiphase Flow 56 (2013) 54–71. doi:10.1016/j.ijmultiphaseflow.2013.05.001.

[32] P.-A. Fayolle, O. Fryazinov, A. Pasko, Rounding, filleting and smoothing of implicit surfaces, Computer-Aided Design and Applications 15 (3) (2018) 399–408. doi:10.1080/16864360.2017.1397890.

[33] R. Thirumalaisamy, N. A. Patankar, A. P. S. Bhalla, Handling Neumann and Robin boundary conditions in a fictitious domain volume penalization framework, Journal of Computational Physics 448 (2022) 110726. doi:10.1016/j.jcp.2021.110726.

[34] I. Quilez, 2D Distance Functions, https://iquilezles.org/articles/distfunctions2d/.

[35] J. Yang, E. Balaras, An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, Journal of Computational Physics 215 (1) (2006) 12–40. doi:10.1016/j.jcp.2005.10.035.

[36] J. L. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Computer Methods in Applied Mechanics and Engineering 195 (44) (2006) 6011–6045. doi:10.1016/j.cma.2005.10.010.

[37] J. Johnston, J. Lou, N. Tilton, Application of projection methods to simulating mass transport in reverse osmosis systems, Computers & Fluids 232 (2022) 105189. doi:10.1016/j.compfluid.2021.105189.

[38] D. L. Brown, R. Cortez, M. L. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, Journal of Computational Physics 168 (2) (2001) 464–499. doi:10.1006/jcph.2001.6715.

[39] R. D. Guy, A. L. Fogelson, Stability of approximate projection methods on cell-centered grids, Journal of Computational Physics 203 (2) (2005) 517–538. doi:10.1016/j.jcp.2004.09.005.

[40] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier-Stokes equations, Journal of Computational Physics 59 (2) (1985) 308–323. doi:10.1016/0021-9991(85)90148-2.

[41] 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.

[42] R. Mittal, H. Dong, M. Bozkurttas, F. M. Najjar, A. Vargas, A. von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, Journal of Computational Physics 227 (10) (2008) 4825–4852. doi:10.1016/j.jcp.2008.01.028.

[43] H. S. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface Cartesian grid method for simulating flows with complex moving boundaries, Journal of Computational Physics 174 (1) (2001) 345–380. doi:10.1006/jcph.2001.6916.

[44] A. Gilmanov, F. Sotiropoulos, A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies, Journal of Computational Physics 207 (2) (2005) 457–492. doi:10.1016/j.jcp.2005.01.020.

[45] J. R. Taylor, Classical Mechanics, University Science Books, 2005.

[46] F. Cariglino, N. Ceresola, R. Arina, External aerodynamics simulations in a rotating frame of reference, International Journal of Aerospace Engineering 2014 (1) (2014) 654037. doi:10.1155/2014/654037.