Title

Immersed boundary smooth extension: A high-order method for solving PDE on arbitrary smooth domains using Fourier spectral methods

Authors

Stein, DB
Guy, RD
Thomases, B

Publication Date

2016

DOI

10.1016/j.jcp.2015.10.023

Peer reviewed

Immersed Boundary Smooth Extension: A high-order method for solving PDE on arbitrary smooth domains using Fourier spectral methods

David B. Stein[a,*], Robert D. Guy[a], Becca Thomases[a]

[a]*Department of Mathematics, University of California, Davis, Davis, CA 95616-5270, USA*

Abstract

The Immersed Boundary method is a simple, efficient, and robust numerical scheme for solving PDE in general domains, yet it only achieves first-order spatial accuracy near embedded boundaries. In this paper, we introduce a new high-order numerical method which we call the Immersed Boundary Smooth Extension (IBSE) method. The IBSE method achieves high-order accuracy by smoothly extending the unknown solution of the PDE from a given smooth domain to a larger computational domain, enabling the use of simple Cartesian-grid discretizations (e.g. Fourier spectral methods). The method preserves much of the flexibility and robustness of the original IB method. In particular, it requires minimal geometric information to describe the boundary and relies only on convolution with regularized delta-functions to communicate information between the computational grid and the boundary. We present a fast algorithm for solving elliptic equations, which forms the basis for simple, high-order implicit-time methods for parabolic PDE and implicit-explicit methods for related nonlinear PDE. We apply the IBSE method to solve the Poisson, heat, Burgers', and Fitzhugh-Nagumo equations, and demonstrate fourth-order pointwise convergence for Dirichlet problems and third-order pointwise convergence for Neumann problems.

Keywords: Embedded boundary, Immersed Boundary, Fourier spectral method, Complex geometry, Partial Differential Equations, High-order

1. Introduction

The Immersed Boundary (IB) method was originally developed for the study of moving, deformable structures immersed in a fluid, and it has been widely applied to such problems since its introduction [1–3]. Recently, the method has been adapted to more general fluid-structure problems, including the motion of rigid bodies immersed in a fluid [4] and fluid flow through a domain with either stationary boundaries or boundaries with prescribed motion [5, 6]. In this broadened context, we use the term *Immersed Boundary method* to refer only to methods in which (*i*) the boundary is treated as a Lagrangian structure embedded in a geometrically simple domain, (*ii*) the background PDE (e.g. the Navier-Stokes equations) are solved on a Cartesian grid everywhere in that domain, and (*iii*) all communication between the Lagrangian structure and the underlying PDE is mediated only by convolutions with regularized δ-functions. These methods have many desirable properties: they make use of robust and efficient Cartesian-grid methods for solving the underlying PDE, are flexible to a wide range of problems, and are simple to implement, requiring minimal geometric information and processing to describe the boundary.

The IB method belongs to the broad category of methods known as *embedded boundary* (EB) methods, including the Immersed Interface [7], Ghost Fluid [8], and Volume Penalty methods [9]. These methods share a common feature: they enable solutions to PDE on nontrivial domains to be computed using efficient and robust structured-grid discretizations; yet these methods differ largely in how boundary conditions are

*Corresponding author
Email address: dbstein@math.ucdavis.edu (David B. Stein)

enforced and whether or not the solution is produced in the entirety of a simple domain. Methods which compute the solution everywhere in a d-dimensional rectangle admit the simplest discretizations and enable the use of high-order discretizations such as Fourier spectral methods. Unfortunately, this simplicity comes coupled with a fundamental difficulty: the *analytic* solution to these problems is rarely globally smooth on the entire domain. Consider the one-dimensional Poisson problem $\Delta u = f$ on the periodic interval $\mathbb{T} = [0, 2\pi]$ with Dirichlet boundary conditions $u(a) = u(b) = 0$ for $a \neq b \in \mathbb{T}$. Even if $f \in C^\infty(\mathbb{T})$, the solution u will typically display jumps in its derivative at the values $x = a$ and $x = b$ (see Figure 2a). The lack of regularity in the analytic problem leads to low-order convergence in many numerical schemes, including the Immersed Boundary method.

The advantages of EB methods are substantial enough that significant effort has been expended on improving their accuracy [10–26]. Two different approaches are generally taken. The first approach involves locally altering the discretization of the PDE in the vicinity of the boundary to accommodate the lack of smoothness in the solution. One example of this approach is the Immersed Interface method [7]. Such approaches are particularly useful for interface problems where the solution is required on both sides of the embedded boundary. When the solution is only needed on one side of the interface, a second approach may be taken in which variables are redefined outside of the domain of interest to obtain higher global regularity. Improved convergence rates are achieved as a natural consequence of the properties of the discretization scheme when applied to smooth problems. Variations of this basic idea have been used by the Fourier Continuation (FC) [23–25] and the Active Penalty (AP) [26] methods to provide high order solutions to PDE on general domains.

In this paper, we introduce a new method termed the *Immersed Boundary Smooth Extension* (IBSE) method. This method is a first step in remedying two deficiencies of Immersed Boundary methods:

1. For generic problems, the IB method produces discretizations that are only *first-order accurate* in the vicinity of domain boundaries (or fluid-structure interfaces) [27].
2. The IB method is only able to specify Dirichlet boundary conditions (no-slip, for fluid problems) without specialized interpolation schemes subordinate to the geometry [28]. Although this is not of interest when solving traditional IB fluid-interface problems, it allows the IB method to be generalized for solving other PDE (i.e. reaction-diffusion equations).

In this paper we restrict our attention to PDE set on stationary domains, and consider only PDE without a global divergence constraint. Direct-forcing IB methods produce solutions to PDE that are C^0, with jumps in the normal-derivative of the solution across the boundary, and converge at first order in the grid-spacing Δx [4, 5]. Drawing on ideas from the AP and FC methods, we use Fourier spectral methods to obtain a highly accurate discretization, while adding a volumetric forcing to non-physical portions of the computational domain to force the solution to be *globally* C^k. We will use the shorthand IBSE-k to refer to our method when we need to explicitly denote the global regularity of the solution that is enforced. High-order accuracy is achieved naturally, as a simple consequence of the convergence properties of the Fourier transform.

The IBSE method retains the essential robustness and simplicity of the original IB method. All communication between the Lagrangian boundary and the underlying Cartesian grid is achieved by convolution with regularized δ-functions or normal derivatives of those δ-functions. This allows an absolute minimum of geometric information to be used. In the traditional IB method, only the position of the Lagrangian structure must be known; the IBSE method additionally requires normals to that structure and an indicator variable denoting whether Cartesian grid points lie inside or outside of the physical domain where the PDE is defined. Additionally, since normal derivatives can be accurately approximated by convolution with derivatives of regularized δ-functions, Neumann and Robin boundary conditions can be imposed in the same way that Dirichlet boundary conditions are in direct-forcing IB approaches.

In contrast to other approaches based on the smooth extension of the forcing function or the solution [19, 23–26], we do not extend the forcing function or solution from a previous timestep. Instead, we *smoothly extend the unknown solution* to the entire computational domain. This approach directly enforces smoothness of the solution, and it allows for high-order *implicit*-time discretizations for parabolic equations and *implicit-explicit* discretizations for many nonlinear PDE. Remarkably, the coupled problem for the solution to an elliptic PDE and that solution's smooth extension can be reduced to the solution of a relatively small

dense system of equations (with size a small multiple of the number of points used to discretize the boundary), along with several FFTs. The dense system depends only on the boundary and the discretization, and so it can be formed and prefactored to allow for efficient time-stepping of parabolic equations.

This paper is organized as follows. In Section 2, we introduce the method, ignoring the details of the numerical implementation. The fundamental contributions of this paper are contained in the modification to the IB discretization that enforces smoothness of the solution, and in the system of equations (given in Section 2.2) that allows for the simultaneous solution of the PDE along with the smooth extension of that unknown solution. In Section 3, we discuss the particular numerical implementation that we choose, detailing a fast algorithm for solving elliptic equations. In Section 4, we compute solutions to the Poisson problem in one and two dimensions, demonstrating high-order pointwise convergence: up to fourth-order for Dirichlet problems and third-order for Neumann problems. In Section 5, we discuss discretization of the heat equation, detailing a fast algorithm for *implicit* timestepping, and demonstrate fourth-order convergence in space and time. These numerical tests include direct comparisons to the Fourier Continuation [24] and Active Penalty [26] methods. Finally, in Section 6, we solve two nonlinear problems: a 2D Burgers' equation with homogeneous Dirichlet boundary conditions, and the Fitzhugh-Nagumo equations with homogeneous Neumann boundary conditions. We demonstrate fourth-order convergence in space and time for Burgers' equation and third-order convergence for the Fitzhugh-Nagumo equations.

2. Methods

Let $\Omega \subset C$, $\Gamma = \partial\Omega$, and $E = C \setminus \overline{\Omega}$. We will assume that Ω is smooth and does not self-intersect. Two typical domains are shown in Figure 1. We refer to Ω as the *physical domain*, E as the *extension domain*, and C as the *computational domain*. We first consider the Poisson problem with Dirichlet boundary conditions:

$$\Delta u = f \qquad \text{in } \Omega, \tag{1a}$$
$$u = g \qquad \text{on } \Gamma. \tag{1b}$$

For now, we assume that f and g are smooth (C^∞) functions defined in Ω and Γ respectively. Let f_e be a smooth extension of f to C, that is, $f_e \in C^\infty(C)$ and $f_e(x) = f(x)$ for all $x \in \Omega$. The *direct forcing* formulation of the Immersed Boundary method provides a way to solve Equation (1):

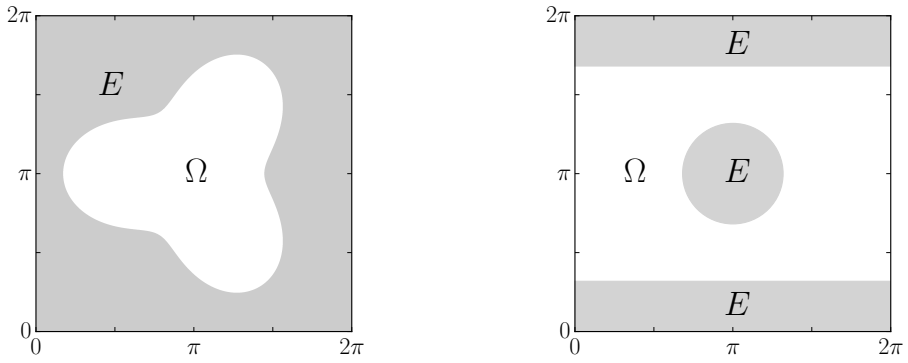
$$\Delta u(x) + \int_\Gamma G(s)\delta(x-s)\,ds = f_e(x) \qquad \text{in } C, \tag{2a}$$
$$\int_C u(x)\delta(x-s)\,dx = g(s) \qquad \text{in } \Gamma. \tag{2b}$$

Equation (1a) is solved in the entire computational domain C, while the boundary condition is enforced by the addition of a singular force G supported on the boundary that acts as a Lagrange multiplier. This singular force term leads to jumps in the normal derivative of the solution u across the boundary Γ. This lack of smoothness restricts Immersed Boundary formulations such as Equation (2) to first-order convergence in the vicinity of the boundary unless one-sided discrete δ-functions are used, which make the implementation of the method more difficult [10, 27].

We define an example one-dimensional problem to illustrate the limited regularity of solutions produced by Equation (2). Let the computational domain be $C = \mathbb{T}$, where the one-dimensional torus $\mathbb{T}$ is identified with the periodic interval $[0, 2\pi]$, and let the physical domain be given by the interval $\Omega = \mathbb{T} \setminus [3, 4]$. The extension domain for this problem is $E = [3, 4]$. On this domain, we will solve the problem:

$$\Delta u = \sin x \qquad \text{in } \Omega = \mathbb{T} \setminus [3,4], \tag{3a}$$
$$u = 0 \qquad \text{on } \Gamma = \{3, 4\}. \tag{3b}$$

As f is chosen to be $f = \sin x$ in Ω, we may choose $f_e = \sin x$ for all $x \in \mathbb{T}$. The analytic solution to this problem is shown in Figure 2a, along with the solution to the extended problem given by Equation (2). Note



(a) Irregular domain

(b) Flow around cylinder

Figure 1: Two different and typical domains to solve PDE on. The physical domain Ω is shown in white, the extension domain E is shown in gray. In both cases, we have taken the computational domain C to be the 2-torus, $\mathbb{T}^2 = [0, 2\pi] \times [0, 2\pi]$. Figure 1a, shows an irregular, connected domain Ω embedded in C. Figure 1b shows a domain that would be used to compute flow around a cylinder; here the domain Ω is itself periodic in one dimension, while the extension region E is not connected.

that the solution u is *only continuous* despite the fact that $f_e \in C^\infty(\mathbb{T})$. Choosing f_e to be smooth leads to low regularity in the solution when the solution to the associated homogeneous problem is nontrivial, as is the case here. Without modification, direct discretizations of this problem will exhibit slow convergence due to the limited regularity of u in the vicinity of the boundary.

Rather than choose a smooth extension to the forcing function f, we choose an extension of the forcing function that gives a smooth solution on the entire computational domain. Let u_e be *any* smooth extension of the unknown solution u into C. We can compute a forcing function F_e associated with u_e:

$$F_e = \Delta u_e. \tag{4}$$

The extended forcing function f_e is then defined to be

$$\tilde{f_e} = \chi_\Omega f + \chi_E F_e, \tag{5}$$

where χ_X denotes the characteristic function of the domain X. Figure 2b shows the extended forcing function $\tilde{f_e}$, along with the associated smooth solution u to Equation (3). Since the solution u is smooth, we expect that standard discretizations of Equation (2) will converge more rapidly when computed using the extension $\tilde{f_e}$ than when using the extension f_e. We emphasize that the extended forcing function $\tilde{f_e}$ *depends on the unknown solution* u.

This motivates us to define a reformulated version of Equation (2), with the extended forcing function f_e given by $\tilde{f_e}$ as defined in Equation (5):

$$\Delta u(x) - (\chi_E \Delta \mathcal{E}_k u)(x) + \int_\Gamma G(s)\delta(x-s)\,ds = \chi_\Omega f(x) \qquad \text{in } C, \tag{6a}$$

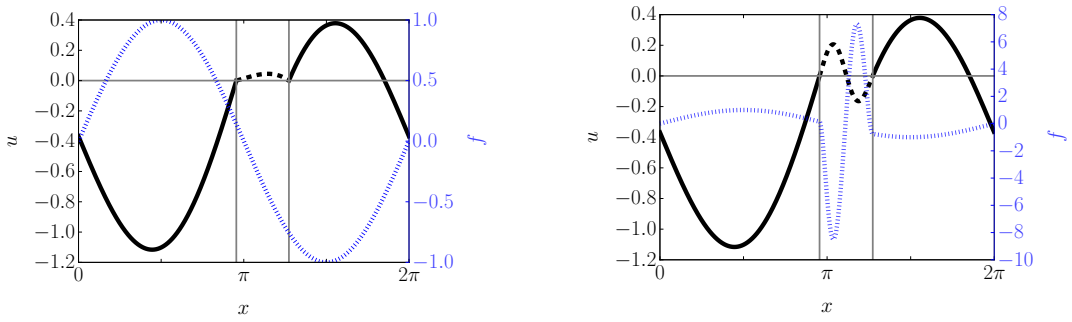
$$\int_C u(x)\delta(x-s)\,dx = g(s) \qquad \text{in } \Gamma. \tag{6b}$$

Here $\mathcal{E}_k$ is *any* smooth extension operator that satisfies

$$\mathcal{E}_k : C^0(C) \cap C^k(\Omega) \to C^k(C), \tag{7a}$$

$$(\mathcal{E}_k u)(x) = u(x) \qquad \forall x \in \Omega. \tag{7b}$$

When restricted to the physical domain Ω, Equation (6a) reduces to $\Delta u = f$, the problem of physical interest. When restricted to the extension domain E, Equation (6a) reduces to $\Delta u = \Delta \mathcal{E}_k u$. As long as



(a) with extended forcing f_e

(b) with extended forcing $\tilde{f}_e$

Figure 2: The solid lines show the analytic solution u to Equation (3) in the physical domain Ω. The dashed line gives the solution u to the extended problem in Equation (2), computed using the extended forcing function f_e in the left figure, and $\tilde{f}_e$ in the right figure. The extended forcing functions are shown as the dotted lines. Note the different scales on the forcing functions in the two figures; the forcing is equal everywhere in the physical domain Ω.

$u \in C^0(C)$, then Equation (7b) ensures that $u|_\Gamma = (\mathcal{E}_k u)|_\Gamma$, and hence $u = \mathcal{E}_k u$ in E. Again by Equation (7b), $u = \mathcal{E}_k u$ in Ω, and so $u = \mathcal{E}_k u$ in C. Since $\mathcal{E}_k u \in C^k(C)$ by Equation (7a), then $u \in C^k(C)$. This additional regularity of the solution u allows standard discretizations of Equation (6) to converge at a faster rate than discretizations of Equation (2). Using $C = \mathbb{T}^d$ and a simple Fourier-spectral discretization, numerical solutions to Equation (6) should converge at $\mathcal{O}(\Delta x^{k+1})$ [29].

In the remainder of this section, we lay out the remaining components needed to fully specify the IBSE method *that do not depend on the particular choice of numerical implementation.* In Section 2.1, we discuss how to smoothly extend a known function from Ω to C. In Section 2.2, we outline a system of equations that allows us to solve for u and its smooth extension simultaneously. A numerical implementation of this method will be discussed in Section 3.

2.1. Smooth extension of a known function from Ω to C

One way to construct a smooth extension to a function is to solve a high-order PDE. Compared to the localized and effectively one-dimensional extension strategies taken in [24, 26], the decision to extend a function by solving a fully d-dimensional PDE may appear to be needlessly complex. However, we will show in Sections 2.2 and 3 that this choice leads to a robust method that is straightforward to implement and requires relatively low computational cost.

In particular, to compute a $C^k(C)$ extension to a function $v \in C^k(\Omega)$, we solve the following equation in the extension domain E:

$$\mathcal{H}^k \xi = 0 \qquad \text{in } E, \tag{8a}$$

$$\frac{\partial^j \xi}{\partial n^j} = \frac{\partial^j v}{\partial n^j} \qquad \text{on } \Gamma, \ \forall\, 0 \le j \le k. \tag{8b}$$

Here $\partial^j \xi / \partial n^j$ denotes the j^{th} normal derivative of ξ on the boundary Γ; a computational formula for $\partial^j \xi / \partial n^j$ is given later in Equation (22). A simple choice of the operator $\mathcal{H}^k$ is the polyharmonic operator $\mathcal{H}^k = \Delta^{k+1}$. From an analytic perspective, this choice is sufficient, although we will show in Section 3.3 that other choices of $\mathcal{H}^k$ will produce better results due to numerical issues. The smooth extension $\zeta = \mathcal{E}_k v$ may then be constructed as

$$\zeta = \chi_\Omega v + \chi_E \xi. \tag{9}$$

Remark 1. Our choice of smoothed forcing $\tilde{f}_e$ *only depends on $\mathcal{E}_k u$ in the extension region E* (see Equation (5)), so there is no need to compute the actual extension ζ since the values of $\mathcal{E}_k u$ in Ω are irrelevant.

For our purposes, we can think of an 'extension' as simply a C^k function in the computational domain C that shares its first k normal derivatives with u on the boundary Γ, i.e. the function ξ.

As with Equation (1), we may solve Equation (8) in Ω by extending the problem to all of C. This can be accomplished by adding singular forces supported along the boundary:

$$\mathcal{H}^k \xi(x) + \sum_{j=0}^{k} (-1)^j \int_\Gamma F_j(s) \frac{\partial^j \delta(x-s)}{\partial n^j}\, ds = 0 \qquad \forall x \in \mathbb{T}^d, \tag{10a}$$

$$(-1)^j \int_C \xi(x) \frac{\partial^j \delta(x-s)}{\partial n^j}\, dx = \frac{\partial^j v}{\partial n^j}(s) \qquad \forall s \in \Gamma,\ 0 \le j \le k. \tag{10b}$$

The integral on the left-hand side of Equation (10b) is notation for the action of the distribution $\partial^j \delta / \partial n^j$ on the smooth function ξ. The boundary condition given in Equation (10b) forces the solution ξ to be C^k, so there is no need to alter this formulation to provide additional regularity.

For notational convenience, we define the operators T_k and T_k^* by:

$$T_k F(x) = \sum_{j=0}^{k} (-1)^j \int_\Gamma F_j(s) \frac{\partial^j \delta(x-s)}{\partial n^j}\, ds, \tag{11a}$$

$$T_k^* \xi(s) = \left(\int_C \xi(s)\delta(x-s)\, ds \quad \int_C \xi(s) \tfrac{\partial \delta(x-s)}{\partial n} \quad \cdots \quad \int_C \xi(s) \tfrac{\partial \delta^k(x-s)}{\partial n^k} \right)^{\mathsf{T}}. \tag{11b}$$

In the language of the Immersed Boundary method, T_k^* is an *interpolation* operator; T_k is a *spread* operator. The notation is suggestive of the fact that these operators obey the adjoint property

$$\langle u, T_k F \rangle_C = \langle T_k^* u, F \rangle_\Gamma \tag{12}$$

for all sufficiently smooth u and F, where the notation $\langle \cdot, \cdot \rangle_X$ denotes the L^2 inner product on X. Using the operators T_k and T_k^*, we may represent Equation (10) as

$$\begin{pmatrix} \mathcal{H}^k & T_k \\ T_k^* & \end{pmatrix} \begin{pmatrix} \xi \\ F \end{pmatrix} = \begin{pmatrix} 0 \\ T_k^* v \end{pmatrix}. \tag{13}$$

2.2. A coupled system of equations for u and its extension

Let the spread (S) and interpolation (S^*) operators be defined as

$$SG(x) = \int_\Gamma G(s)\delta(x-s)\, ds, \tag{14a}$$

$$S^*\xi(s) = \int_C \xi(x)\delta(x-s)\, dx, \quad \forall s \in \Gamma. \tag{14b}$$

Notice that $S = T_0$ and $S^* = T_0^*$. We can now represent Equation (6) simply as

$$\Delta u - \chi_E \Delta \xi + SG = \chi_\Omega f, \tag{15a}$$

$$S^* u = g, \tag{15b}$$

where ξ is defined by the extension equation

$$\mathcal{H}^k \xi = 0, \tag{16}$$

along with the constraint that

$$T_k^* \xi = T_k^* u. \tag{17}$$

Equations (15b), (16) and (17) can be combined into one system of equations:

$$\begin{pmatrix} \Delta & -\chi_E\Delta & & S \\ & \mathcal{H}^k & T_k & \\ -T_k^* & T_k^* & & \\ S^* & & & \end{pmatrix} \begin{pmatrix} u \\ \xi \\ F \\ G \end{pmatrix} = \begin{pmatrix} \chi_\Omega f \\ 0 \\ 0 \\ g \end{pmatrix}. \tag{18}$$

Equation (18) is the system of equations that allows the IBSE method to solve for u and its extension ξ simultaneously. The remainder of this paper will be concerned with the discretization and inversion of Equation (18) and numerical studies demonstrating the accuracy of solutions produced by those discretizations. We remark that Equation (18) is equally valid for operators of the Helmholtz type $(\Delta - \kappa\mathbb{I})$ that appear in the discretization of parabolic equations, which will be discussed in Section 5.

3. Numerical implementation

For concreteness, we will discuss the numerical implementation in the context of Fourier spectral methods, with the computational domain C given by $C = \mathbb{T}^d = [0, 2\pi]^d$. We will discretize the domain using a regular Cartesian mesh with n points x_n discretizing each dimension: $\Delta x = 2\pi/n$ and $x_n = n\Delta x$. Differential operators are discretized in the usual way.

Remark 2. Few of the details depend upon the choice to use Fourier spectral methods, and they are chosen because of their simple implementation, computational speed, and high order of accuracy. Inversion of the elliptic operator $\mathcal{L}$, as well as the extension operator $\mathcal{H}^k$ is also greatly simplified when using Fourier spectral methods. However, any discretization based on a regular Cartesian mesh could be used with minimal modifications to the method.

In order to fully describe a discretization and solution strategy to Equation (18), we must describe several key elements.

1. In Section 3.1, we discretize the spread (S, T_k) and interpolation (S^*, T_k^*) operators.
2. In Section 3.2 we define a new regularization of the δ-function that is accurate and smooth.
3. In Section 3.3, we define our extension operator $\mathcal{H}^k$ and show how this choice of $\mathcal{H}^k$ controls the numerical conditioning of the extension problem.
4. In Section 3.4, we describe an efficient inversion strategy for the IBSE-k system given by Equation (18).
5. Finally, in Section 3.5, we provide implementational details and briefly discuss the computational complexity of our inversion scheme.

3.1. Discretization of S, S^*, T_k, and T_k^*

Let the boundary Γ be parametrized by the function $X(s)$. In all examples in this manuscript, the boundary Γ is one-dimensional and closed; the single parameter s is defined on the periodic interval $[0, 2\pi]$. The spread operator $S : \Gamma \to C$ is defined as

$$(SF)(x) = \int_\Gamma F(s)\delta(x - X(s))\, ds. \tag{19}$$

The discrete version of this operator requires a regularized δ-function and a discretization of the integral over Γ. Construction of a regularized δ-function with the properties necessary for the IBSE-k method is non-trivial; we will delay discussion of this choice until Section 3.2 and simply denote the regularized δ-function as $\tilde{\delta}$. Multivariate δ-functions are computed as Cartesian products of the univariate δ and also denoted by $\tilde{\delta}$. Discretization of the integral over Γ is made by choosing n_{bdy} quadrature nodes $\tilde{\Gamma} = \{X_i\}_{i=1}^{n_{\text{bdy}}}$, equally spaced in the parameter interval $[0, 2\pi]$ so that $X_i = X(s_i)$ and $s_i = (i-1)2\pi/n_{\text{bdy}}$. Quadrature weights are computed at each quadrature node to be $\Delta s_i = \left\| \frac{\partial X}{\partial s}(s_i) \right\|_2$; this is a spectrally accurate quadrature rule

for the integral of smooth periodic functions on Γ. The discrete spread operator S maps functions sampled at points in $\tilde{\Gamma}$ to C by

$$(SF)(x) = \sum_{i=1}^{n_{\text{bdy}}} F(s_i)\tilde{\delta}(x - X_i)\Delta s_i. \tag{20}$$

We do not adopt explicit notation to distinguish between the analytic and discretized operators. The number of points in the quadrature is chosen so that $\Delta s \approx 2\Delta x$. This choice of node-spacing is wider than that recommended for the traditional IB method [1] but has been observed empirically to be the optimal choice in other studies of *direct-forcing* IB methods [4]. The interpolation operator S^* may be defined by the adjoint property $\langle u, SF\rangle_C = \langle S^*u, F\rangle_\Gamma$, but we note that the discrete interpolation operator S^* produces a discrete function

$$(S^*u)(s_k) = \int_C u(x)\delta(x - X(s_k))\, dx. \tag{21}$$

Discrete integrals over C are straightforward sums computed over the underlying uniform Cartesian mesh.

Normal derivatives of $\tilde{\delta}$ are computed by the formula [30]

$$\frac{\partial^j \tilde{\delta}}{\partial n^j} = n_{i_1} \cdots n_{i_2} \frac{\partial^j \tilde{\delta}}{\partial x_{i_1} \cdots \partial x_{i_j}}, \tag{22}$$

where the Einstein summation convention has been used to indicate sums over repeated indices and $\partial^j \tilde{\delta}/\partial x_{i_1} \cdots \partial x_{i_j}$ is computed as Cartesian products of the appropriate derivatives of the one-dimensional $\tilde{\delta}$. For example, $\partial\tilde{\delta}/\partial n$ in two dimensions is computed as

$$\frac{\partial \tilde{\delta}}{\partial n} = n_x \tilde{\delta}' \otimes \tilde{\delta} + n_y \tilde{\delta} \otimes \tilde{\delta}'. \tag{23}$$

Analogous to the definition of S, we define the spread operator for the j^{th} normal derivative, $T_{(j)}$, as

$$(T_{(j)}F)(x) = (-1)^j \sum_{i=1}^{n_{\text{bdy}}} F(s_i) \frac{\partial^j \tilde{\delta}}{\partial n^j}(x - X_i)\Delta s_i \tag{24}$$

and again define the interpolation operator $T^*_{(j)}$ by the adjoint property $\left\langle u, T_{(j)}F\right\rangle_C = \left\langle T^*_{(j)}u, F\right\rangle_\Gamma$. Analogous to the action of S^*, the operator $T^*_{(j)}$ produces approximations of the j^{th} normal derivative of a function defined on C at the quadrature nodes of $\tilde{\Gamma}$. Finally, the composite operator T^*_k is defined by:

$$T^*_k = \begin{pmatrix} T^*_{(0)} & T^*_{(1)} & \cdots & T^*_{(k)} \end{pmatrix}^{\mathsf{T}}, \tag{25}$$

while $T_k = \sum_{j=1}^{k} T_{(j)}$.

3.2. Construction of a smooth discretization of δ

The IBSE-k method is capable of producing solutions that converge at $\mathcal{O}(\Delta x^{k+1})$. In order to achieve this accuracy, our choice of regularized δ-function must satisfy several conditions. For a general value of k:

1. Its interpolation accuracy must be at least $\mathcal{O}(\Delta x^{k+1})$.
2. It must be at least C^k, so that its k^{th} normal derivative is continuous, allowing it to be used in the discretization of T_k and T^*_k.

In addition, we will require that the δ-function has compact (and small) support so that the spread and interpolation operators S, S^*, T_k, and T_k^* may be rapidly applied. The commonly used 'four point' δ-function is C^1, has a support of four gridpoints, and produces approximations of S, S^*, T_1 and T_1^* that are accurate to $\mathcal{O}(\Delta x^2)$ [31]. The use of this δ-function regularization is sufficient for the implementation of IBSE-1, but it is not sufficient for higher order methods, due to both its limited *accuracy* and *regularity*.

In this manuscript, we discretize the IBSE-k method for $k = 1$, 2, and 3, corresponding to second, third, and fourth order accuracy in Δx for Dirichlet problems. We therefore need a regularization of δ that is C^3 and has an interpolation accuracy of $\mathcal{O}(\Delta x^4)$ for smooth functions. We are not aware of any functions with these properties currently defined in the literature, and so we construct such a function here. For simplicity, we do not attempt to impose other conditions that are often imposed on δ-functions, such as the *even-odd* condition or *sum of squares* condition [32]. The strategy that we follow is to choose a δ-function with sufficient accuracy but limited regularity and convolve it against itself to increase its smoothness. A similar approach was used in [33] to generate a smoother (C^2) version of the traditional Peskin four point δ-function [1]. In order to provide an analytic formula, a base δ-function with a simple functional form must be used. We start with the function

$$\delta_{IB_4}(r) = \begin{cases} 1 - \frac{1}{2}|r| - |r|^2 + \frac{1}{2}|r|^3 & 0 \le |r| \le 1, \\ 1 - \frac{11}{6}|r| + |r|^2 - \frac{1}{6}|r|^3 & 1 \le |r| \le 2, \\ 0 & 2 \le |r|, \end{cases} \tag{26}$$

which has $\mathcal{O}(\Delta x^4)$ interpolation accuracy, C^0 regularity, and a support of four gridpoints [34, 35]. Define:

$$\tilde{\delta} = \delta_{IB_4} * \delta_{IB_4} * \delta_{IB_4} * \delta_{IB_4}, \tag{27}$$

where $*$ denotes convolution. It is clear that $\tilde{\delta} \in C^3$, its support is 16 gridpoints, and it is a simple exercise to show that convolution preserves interpolation accuracy. The formula for $\tilde{\delta}$ is given in Appendix A.

Remark 3. While our choice of δ-function regularization limits the spatial discretization in this paper to fourth-order accuracy, this is not a fundamental limitation of the method. One way to generalize to higher orders would be to construct smoother and more accurate regularizations of the δ-function similar to the one that we construct, which maintain finite support for computational efficiency. An alternative approach would be to use globally supported regularizations of the δ-function that are C^∞ (i.e. sinc or Gaussian functions) and make use of fast sinc transforms or fast Gaussian transforms in order to rapidly apply the spread and interpolation operators [36, 37].

Remark 4. Our construction of $\tilde{\delta}$ is likely not optimal, in that there probably exist C^3 δ-function regularizations accurate to $\mathcal{O}(\Delta x^4)$ with a support of less than 16 gridpoints (e.g. a δ-function with C^3 regularity and accuracy of $\mathcal{O}(\Delta x^2)$ with support of only six gridpoints is defined in [38]). The construction of such a δ-function would be worthwhile, allowing for coarser discretizations of problems with closely spaced boundaries and faster application of the operators S, S^*, T_k, and T_k^*.

3.3. Choice of extension operator $\mathcal{H}^k$

Due to the nullspace of the periodic polyharmonic operator Δ^{k+1}, we choose the operator $\mathcal{H}^k$ used in the extension problem given by Equation (8) to be the invertible operator

$$\mathcal{H}^k = \Delta^{k+1} + (-1)^{k+1}\Theta(k, n). \tag{28}$$

Here Θ is a positive scalar function that depends on the smoothness of the extension (k) and the number of Fourier modes (n) used in the discretization of the problem. The function Θ is chosen to mitigate the numerical condition number of the operator $\mathcal{H}^k$:

$$\kappa = 1 + \frac{(n/2)^{2(k+1)}}{\Theta}. \tag{29}$$

Minimizing the condition number κ (by taking Θ to be large) must be balanced against the need to resolve the intrinsic length scale $L = \Theta^{-1/2(k+1)}$ introduced to the problem. In practice, we find that taking Θ to be

$$\Theta^* = \max\left(1, \alpha\varepsilon\left(\frac{n}{2}\right)^{2(k+1)}\right), \tag{30}$$

where ε is machine precision (2^{-52} for double precision and 2^{-112} for quadruple precision) and $\alpha = 0.001$ produces stable and accurate solutions. Numerical results are not particularly sensitive to the choice of α (see Figure 3a). This stability is demonstrated later (in Section 4.1) with highly accurate solutions shown for the IBSE-1, IBSE-2, and IBSE-3 methods up to $n = 2^{22}$. All numerical results in this paper are produced using Θ^* to construct $\mathcal{H}^k$.

For large values of n and k, the value of Θ can substantially impact the overall accuracy of the IBSE-k algorithm. We demonstrate this effect in Figure 3 for $n = 2^{16}$, $k = 3$, and solutions to Equation (3). In Figure 3a, we plot the L^∞ error in the solution u as a function of Θ across 32 orders of magnitude. The minimum error observed is 7.63×10^{-12}, at $\Theta = 10^{17}$, which is within an order of magnitude of Θ^* (denoted by a vertical grey line). Although there is some sensitivity to Θ near the minimum, it is small: the error is bounded by 10^{-10} over 6 orders of magnitude of Θ. In Figure 3b, we show a zoom of the extension near the domain boundary at $x = 3$ for the solution produced using Θ^*, which has an error of 1.12×10^{-11}. The extension decays rapidly to 0, but is well resolved by the discretization: there are 150 gridpoints between the boundary at $x = 3$ and the peak at $x \approx 3.014$. In Figure 3c, we show a zoom of the extension produced when $\Theta = 10^{30}$, which gives a solution with an error of 1.23×10^{-7}, four orders of magnitude worse than the error produced when $\Theta = \Theta^*$. Although taking Θ this large leads to a very well-conditioned operator $\mathcal{H}^k$, the extension decays *very* rapidly to 0, and the discretization is no longer able to fully resolve the length scales: there are only *three* gridpoints between the boundary at $x = 3$ and the peak at $x \approx 3.0004$. Finally, in Figure 3d, we use $\Theta = 1$. Although there are no fine length-scales to be resolved, ill-conditioning in the operator $\mathcal{H}^k$ leads to an error of 5.2×10^{-2}.

3.4. Inversion of the IBSE-k system (18)

In this section, we consider solving Equation (18) for an *invertible* elliptic operator $\mathcal{L}$ in place of Δ. The case where $\mathcal{L}$ is the periodic Laplacian is complicated by the nullspace of Δ: even though Equation (18) is invertible, the method for inversion that we outline in this section is not directly applicable; details for the resolution of this problem are provided in Appendix B. The system of equations is

$$\left(\begin{array}{cc|c} \mathcal{L} & -\chi_E\mathcal{L} & S \\ & \mathcal{H}^k & T_k \\ \hline -T_k^* & T_k^* & \\ S^* & & \end{array}\right)\begin{pmatrix} u \\ \xi \\ F \\ G \end{pmatrix} = \begin{pmatrix} \chi_\Omega f \\ 0 \\ 0 \\ g \end{pmatrix}. \tag{31}$$

This system is large: in two spatial dimensions, the number of equations in the system is $2n^2 + (k+1)n_{\text{bdy}}$. By block Gaussian elimination, we can find a *Schur-complement* for this matrix, which we label SC:

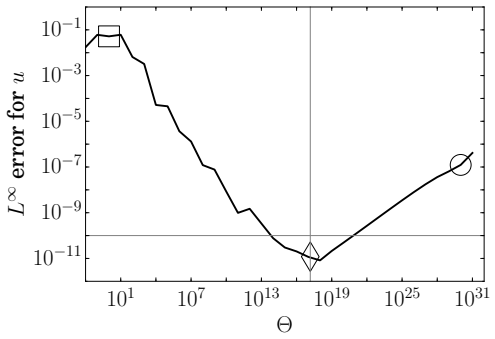
$$SC = \begin{pmatrix} -T_k{}^* & T_k{}^* \\ S^* & \end{pmatrix}\begin{pmatrix} \mathcal{L} & -\chi_E\mathcal{L} \\ & \mathcal{H}^k \end{pmatrix}^{-1}\begin{pmatrix} & S \\ T_k & \end{pmatrix}, \tag{32}$$

along with an associated system of equations for the Lagrange multipliers F and G:

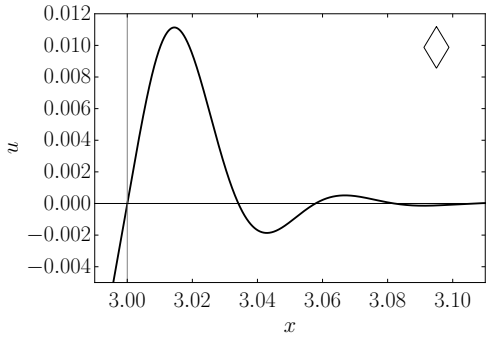
$$\begin{pmatrix} & SC & \end{pmatrix}\begin{pmatrix} F \\ G \end{pmatrix} = \begin{pmatrix} S^*\mathcal{L}^{-1}\chi_\Omega f - g \\ T_k^*\mathcal{L}^{-1}\chi_\Omega f \end{pmatrix}. \tag{33}$$

The size of SC is comparatively small, only $(k+1)n_{\text{bdy}}$ square. This equation is the key to the efficiency of the algorithm, as it allows the Lagrange multipliers F and G to be computed rapidly without first solving for u or ξ. Once F and G are determined, we may solve for ξ:

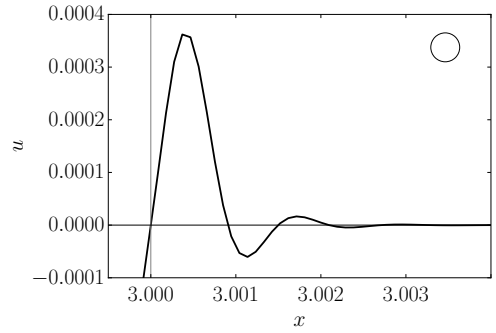
$$\xi = -(\mathcal{H}^k)^{-1}T_k F, \tag{34}$$



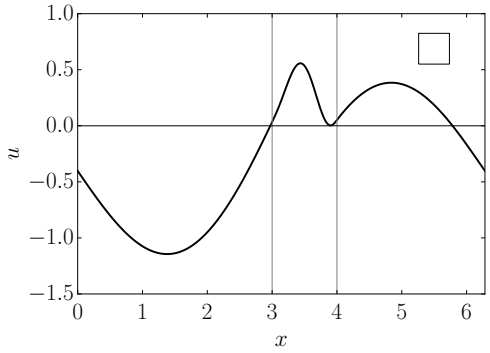
(a) L^∞ error as a function of Θ



(b) $\Theta = \Theta^*$



(c) $\Theta = 1 \times 10^{30}$



(d) $\Theta = 1$

Figure 3: The effect that the value Θ, used to adjust the conditioning of the extension operator $\mathcal{H}^k$, has on the solution u of Equation (3). Figure 3a shows a plot of the L^∞ error against Θ. Three special points are denoted in this figure: $\Theta = \Theta^*$ as defined in Equation (30), denoted by ($\Diamond$); $\Theta = 10^{30}$, denoted by ($\bigcirc$); and $\Theta = 1$, denoted by ($\square$). The solutions corresponding to these special values of Θ, along with their extensions, are shown in Figures 3b to 3d, respectively. For Figures 3b and 3c, only a zoom of the area near $x = 3$ is shown. See Section 3.3 for a detailed discussion.

and once ξ is known, u may be computed as

$$u = \mathcal{L}^{-1}\left(\chi_D f + \chi_E \mathcal{L}\xi - SG\right). \tag{35}$$

The computation of ξ and u requires only a series of fast operations: FFTs, multiplies, and applications of the spread operators.

Rapid inversion of the system of equations for F and G given in Equation (33) is not a trivial task. Because we have restricted to problems set on stationary domains and the size of SC is small, it is feasible to proceed using dense linear algebra. We form SC by repeatedly applying it to basis vectors. This operation is expensive: for two dimensions it is $\mathcal{O}(N^{3/2}\log N)$ in the total number of unknowns $N = n^2$. The Schur-complement is then factored by the LU algorithm provided by LAPACK [39]. Once this factorization is computed Equation (33) can be solved rapidly. This Schur-complement depends only on the domain and the discretization, so the LU-decomposition can be reused to solve multiple problems on the same domain or in each timestep when solving time-dependent problems. See Section 3.5 for a discussion of the computational complexity of the method and Table 2 for the actual numerical cost of the method applied to a test problem.

3.5. Summary of algorithm and numerical complexity

For a given geometry, choice of elliptic operator $\mathcal{L}$, and discretization size n, the implementation of the IBSE-k method proceeds as follows.

1. Setup
   (a) The boundary Γ is discretized as described in Section 3.1, and stencils for the evaluation of $\tilde{\delta}$ and its first k normal derivatives are computed at all nodes of the discrete boundary in $\tilde{\Gamma}$ to allow for the rapid application of the S, S^*, T_k, and T_k^* operators.
   (b) The Schur-complement matrix (Equation (32) for invertible $\mathcal{L}$, and Equation (70) for $\mathcal{L} = \Delta$ or other non-invertible $\mathcal{L}$) is formed columnwise, by repeatedly applying it to basis vectors. For invertible $\mathcal{L}$, this task may be completed as follows (for non-invertible $\mathcal{L}$, the process is nearly identical):
      i. One element of either F or G is set to 1, all other elements are set to 0.
      ii. These forces are spread to the Eulerian grid, by the application of T_k and S.
      iii. ξ, and then u, are computed by applying Equation (34) followed by Equation (35).
      iv. The quantities S^*u and $-T_k^*u + T_k^*\xi$ are computed, and placed in the appropriate rows and column of the Schur-complement.

      Note that item (ii) corresponds to application of the right-most matrix in Equation (32), item (iii) corresponds to inverting the middle matrix in Equation (32), while item (iv) corresponds to the application of the left-most matrix in Equation (32). We remark that although this operation is expensive (see Table 2), it is trivially parallelizable, as the task of applying the Schur-complement to each individual basis vector is independent.
   (c) The LU decomposition of the Schur-complement is computed.
   (d) Optionally, these items may be saved, bypassing the expensive computation and factorization of the Schur-complement in future uses of the IBSE-k method when applied to the same geometry, $\mathcal{L}$, and discretization size n.
2. Solve
   (a) The right-hand side of Equation (33) (or equivalently, Equation (70) when $\mathcal{L}$ is not invertible) is computed.
   (b) Using the pre-computed LU decomposition of the Schur complement matrix, the Lagrange multipliers F and G are found by solving Equation (33) (or equivalently, Equation (70) when $\mathcal{L}$ is not invertible).
   (c) The smooth extension ξ, and finally u, are computed via Equations (34) and (35) (or equivalently, Equation (72) when $\mathcal{L}$ is not invertible)

Dimension	Cost of FFT	Cost of Setup	Cost of Solve
2	$\mathcal{O}(N \ln N)$	$\mathcal{O}(N^{3/2} \ln N)$	$\mathcal{O}(N \ln N)$
3	$\mathcal{O}(N \ln N)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^{4/3})$

Table 1: Numerical cost of the IBSE-k algorithm as a function of $N = n^d$ for dimensions $d = 2$ and $d = 3$, along with the cost of an FFT. The 'Cost of Solve' refers to the cost of solving Equation (18) once, which is equivalent to the cost of solving one elliptic problem or taking one implicit timestep of a parabolic problem.

In items 1(b), 2(a), and 2(c), application of the FFT and IFFT is used to move variables between frequency and physical space, while differential operators are applied in frequency space and multiplications with characteristic functions are carried out in physical space.

The computational work for this method can be broken into two main tasks: (*i*) the setup items, listed above, which are dominated by the formation and factorization of the Schur-complement defined in Equation (32) and (*ii*) the production of one solution u given the Schur-complement, its factorization, the body forcing f, and the boundary conditions g. We summarize the scaling of the algorithm in Table 1; see Table 2 in Section 5.1 for the actual numerical cost of the method applied to a test problem. In two dimensions, the cost of a solve scales like the cost of an FFT; in three dimensions the cost is slightly worse: $\mathcal{O}(N^{4/3})$ in the total number of unknowns $N = n^3$. The higher cost of the algorithm in three-dimensions is due to the cost of factoring and inverting the Schur-complement, however, these matrices are highly structured. It is quite possible that the cost of inversion could be reduced with an appropriately preconditioned iterative method, by directly exploiting the structure of the matrix, e.g. with an inverse Multipole Method [40], or by indirectly exploiting that structure using an algorithm like HODLR [41] or Algebraic Multigrid [42].

4. Results: Poisson equation

4.1. One-dimensional test problem

To demonstrate both the power and the limitations of the IBSE method, we compute the solution to the one-dimensional example defined in Equation (3). Solutions are computed on grids ranging from $n = 2^4$ to 2^{22} points, in both double- and quadruple-precision arithmetic, for the IB, IBSE-1, IBSE-2, and IBSE-3 methods. Figure 4 shows a refinement study demonstrating the expected first-, second-, third-, and fourth-order accuracy in L^∞ for the IB, IBSE-1, IBSE-2, and IBSE-3 methods, respectively. The IB method achieves an error of 4.23×10^{-7} with $n = 2^{22}$ grid points. Imposing additional smoothness on the solution allows this error to be matched by the IBSE-1 method at $n = 4096$, by the IBSE-2 method at $n = 1024$, and by the IBSE-3 method at $n = 512$. For the IBSE-3 method, this is a factor of nearly 10000 less gridpoints; equivalent to a factor of 100 million less gridpoints for two-dimensional problems. In practice, obtaining solutions accurate to six digits with the traditional IB method is often impractical; with the IBSE method this kind of accuracy can be achieved on reasonably sized grids.

In double precision, all of the methods achieve the expected convergence rate up to some value of n. The double-precision solutions begin to diverge from the quadruple-precision solutions at $n = 2^{17}$ for IBSE-1, at $n = 2^{16}$ for IBSE-2, and at $n = 2^{12}$ for IBSE-3. Using $\Theta = \Theta^*$ (see Section 3.3) to control the condition number of the extension operator $\mathcal{H}^k$ provides sufficient numerical stability at all values of n to enable the computation of solutions accurate to 10-11 digits. In quadruple precision, all methods exhibit the expected order of convergence across a wide range of values of n, although the fourth order method begins to fall short of the expected path of convergence for extremely fine grids ($n \geq 2^{20}$).

Remark 5. We note that this test problem, as well as the test problems in Sections 5.1, 6.1 and 6.2, are *exterior* problems set on periodic domains. Periodicity of solutions in the physical domain Ω is enforced naturally in these problems through the use of the Fourier basis.

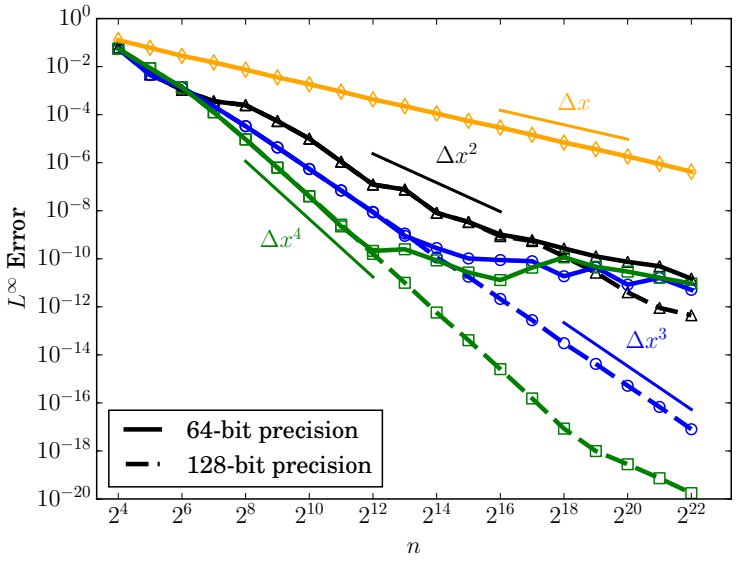


Figure 4: L^∞ error for solutions to Equation (3) produced with IB (◇), IBSE-1 (△), IBSE-2 (○), and IBSE-3 (□), demonstrating $\mathcal{O}(\Delta x)$, $\mathcal{O}(\Delta x^2)$, $\mathcal{O}(\Delta x^3)$, and $\mathcal{O}(\Delta x^4)$ convergence, respectively. Solutions computed in double precision (64-bit) are shown with solid lines; solutions computed with quadruple precision (128-bit) are shown with dashed lines.

4.2. Two-dimensional test problem: solution inside a circle

Let $\Omega = B_2((\pi, \pi))$ be the circle of radius 2 centered at (π, π), and identify $C = \mathbb{T}^2$ with the square $[0, 2\pi] \times [0, 2\pi]$. We will solve the Dirichlet problem:

$$\Delta u = -4 \qquad \text{in } \Omega, \tag{36a}$$

$$u = 0 \qquad \text{on } \Gamma = \partial\Omega, \tag{36b}$$

which has an analytic solution u_a given by

$$u_a = 4 - (x - \pi)^2 - (y - \pi)^2. \tag{37}$$

We solve this problem in double-precision arithmetic with the IBSE-1, IBSE-2, and IBSE-3 methods for $n = 2^4$ to $n = 2^{11}$. In Figure 5, we show L^∞ error as a function of n, indicating second-, third-, and fourth-order accuracy for IBSE-1, IBSE-2, and IBSE-3 respectively.

Using the solutions to this problem, we demonstrate the property that the IBSE method produces *globally smooth* solutions. In Figure 6, we show the solution u to Equation (36), along with its first, second, and third derivatives in the x-direction, produced using the IB, IBSE-1, IBSE-2, and IBSE-3 methods with $n = 2^9$. For simplicity, all functions are shown only along the slice $y = \pi$, and the intersection of this slice with the boundary Γ is shown as gray vertical lines. Derivatives are computed spectrally using the FFT. We expect solutions produced by the IBSE-k method to exhibit discontinuities in the $(k+1)^{\text{st}}$ normal derivative across interfaces, and for all derivatives of lower order to be continuous. We see this expectation validated in the solutions presented in Figure 6: functions shown in plots with shading behind them in Figure 6a are *at least continuous*, all others are discontinuous. This property of *global smoothness of the solutions* enables the Fourier-spectral discretization to obtain high-order accuracy.

4.3. Two-dimensional test problem with Neumann boundary conditions

Traditional Immersed Boundary approaches are unable to provide solutions to Neumann problems since convolution-style estimation of the normal derivative is not convergent at the boundary due to the lack

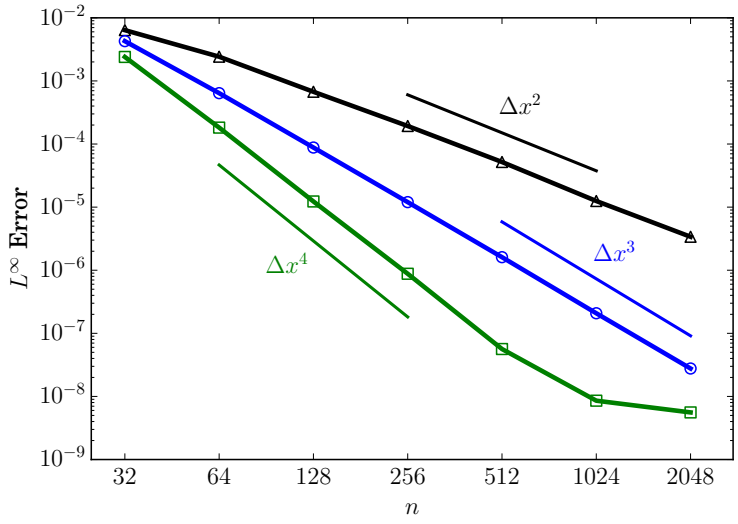
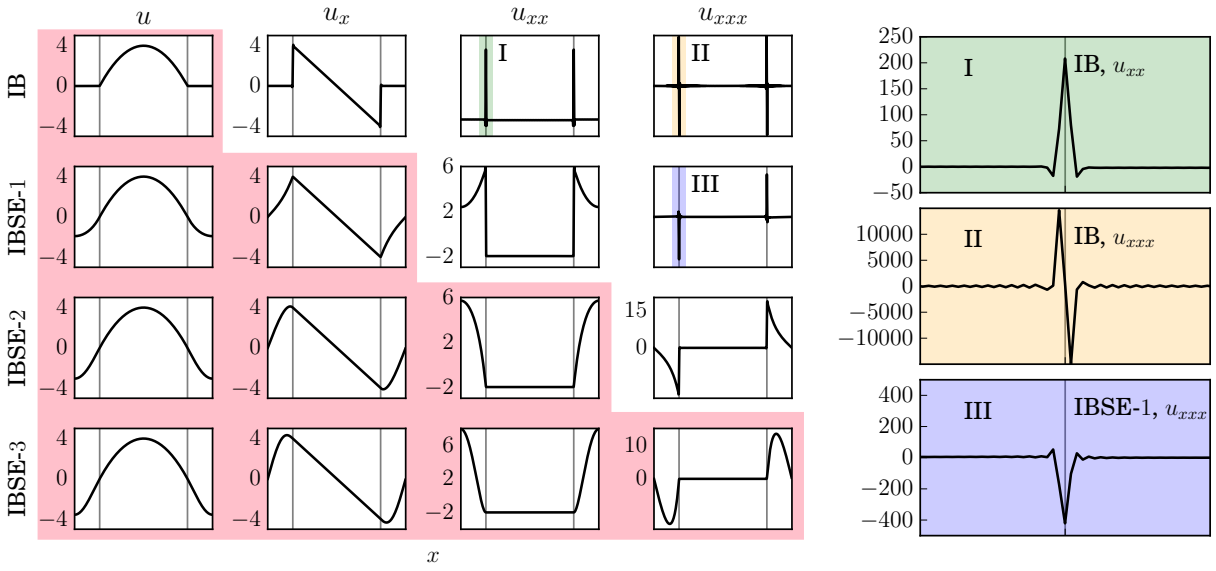


Figure 5: L^∞ error for solutions to Equation (36) produced with IBSE-1 (△), IBSE-2 (○), and IBSE-3 (□), demonstrating $\mathcal{O}(\Delta x^2)$, $\mathcal{O}(\Delta x^3)$, and $\mathcal{O}(\Delta x^4)$ convergence, respectively.



(a) (b)

Figure 6: A demonstration of the smoothness of the solution u to Equation (36) produced by the IB, IBSE-1, IBSE-2, and IBSE-3 methods. Figure 6a shows the solution u and the solution's first, second, and third x derivatives. All plots show only the slice $y = \pi$ from $x = 0$ to $x = 2\pi$ and vertical gray lines denote the intersection of that slice with the boundary Γ at $x = \pi - 2$ and $x = \pi + 2$. The functions in plots with shading behind them (to the lower left) are *at least continuous*, all other functions shown are *discontinuous*. Solutions produced using the IBSE-k method exhibit discontinuities in the $(k+1)^{\text{st}}$ normal derivative across Γ; all lower-order derivatives are continuous. Scales of the three plots in the upper right have been omitted. For these plots, zooms of the region near the boundary at $x = \pi - 2$ are shown in Figure 6b. The x-axis in these plots runs from $x = \pi - 2 - 0.3$ to $x = \pi - 2 + 0.3$.

of regularity of the solution. The IBSE method does not have this limitation because of the additional smoothness of the solution. The only modification necessary to adapt Equation (18) to solve Neumann problems is to change the S and S^* operators that enforce and specify the boundary conditions. Consider the general Neumann problem:

$$\Delta u = f \qquad \text{in } \Omega, \tag{38a}$$

$$\frac{\partial u}{\partial n} = g \qquad \text{on } \Gamma. \tag{38b}$$

Define

$$T_{(1)}F(x) = \int_\Gamma F_j(s)\frac{\partial \delta(x-s)}{\partial n}\,ds, \tag{39a}$$

$$T^*_{(1)}\xi(s) = -\int \xi(x)\frac{\partial \delta(x-s)}{\partial n}\,dx, \quad \forall s \in \Gamma. \tag{39b}$$

The analogue to Equation (18) is

$$\begin{pmatrix} \Delta & -\chi_E\Delta & T_{(1)} \\ & \mathcal{H}^k & \\ -T_k^* & T_k^* & \\ T^*_{(1)} & & \end{pmatrix} \begin{pmatrix} u \\ \xi \\ F \\ G \end{pmatrix} = \begin{pmatrix} \chi_\Omega f \\ 0 \\ 0 \\ g \end{pmatrix}. \tag{40}$$

Remark 6. It is just as simple to modify the IBSE method to allow the solution of Robin problems. Rather than replacing the upper-right and lower-left operators in Equation (18) with $T_{(1)}$ and $T^*_{(1)}$, they should instead be replaced with linear combinations of S, $T_{(1)}$, S^*, and $T^*_{(1)}$.

Let $\Omega = B_1((\pi,\pi))$. We will solve the problem

$$\Delta u = e^{\sin x}\left(\cos^2 x - \sin x\right) - \cos y \qquad \text{in } \Omega, \tag{41a}$$

$$\frac{\partial u}{\partial n} = \cos^2 x\, e^{\sin x} - \sin^2 y \qquad \text{on } \Gamma, \tag{41b}$$

which has the analytic solution

$$u_a = e^{\sin x} + \cos y. \tag{42}$$

We solve this problem with the IBSE-1, IBSE-2, and IBSE-3 methods, for which we expect first-, second-, and third-order convergence, respectively. The reason for the lower order of convergence than in Dirichlet problems is that our discretization of $T^*_{(1)}$ is one order less accurate than S^* when acting on functions of the same regularity. Despite this, the solution u produced by IBSE-k still has global C^k regularity. Solutions are computed in double precision, for $n = 2^4$ to $n = 2^{11}$. Figure 7 shows L^∞ error as a function of n. Second- and third-order convergence is observed for the IBSE-2 and IBSE-3 methods. The IBSE-1 method converges at a rate that is slightly less than first-order.

5. Results: heat equation

We now consider solving the heat equation with Dirichlet boundary conditions:

$$u_t - \Delta u = f(x,t) \qquad \text{in } \Omega, \tag{43a}$$

$$u = g(x,t) \qquad \text{on } \Gamma. \tag{43b}$$

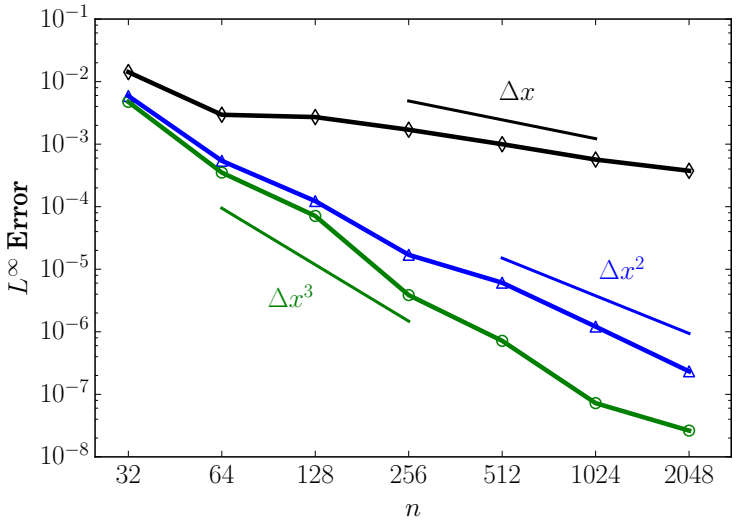


Figure 7: L^∞ error for solutions to Equation (41) produced using IBSE-1 (△), IBSE-2 (○), and IBSE-3 (□) demonstrating slightly less than first-order convergence, second-order convergence, and third-order convergence, respectively.

Since the IBSE method can directly invert elliptic equations, it can easily be adapted to provide high-order implicit-time discretizations for the heat equation. Many implicit methods could be used; we choose BDF4 [43]:

$$\left(\mathbb{I} - \frac{12}{25}\Delta t\Delta\right) u^{n+1} = \frac{1}{25}\left(12\Delta t f(t+\Delta t) + 48u^n - 36u^{n-1} + 16u^{n-2} - 3u^{n-3}\right). \tag{44}$$

Applying this scheme to Equation (43b) is equivalent to solving the problem:

$$\mathcal{L}u^{n+1} = \tilde{f} \qquad \text{in } \Omega, \tag{45a}$$

$$u^{n+1} = g(x, t+\Delta t) \qquad \text{on } \Gamma, \tag{45b}$$

where we define:

$$\mathcal{L} = \mathbb{I} - \frac{12}{25}\Delta t\Delta, \tag{46a}$$

$$\tilde{f} = \frac{1}{25}\left(12\Delta t f(t+\Delta t) + 48u^n - 36u^{n-1} + 16u^{n-2} - 3u^{n-3}\right). \tag{46b}$$

This may be solved using the algorithm described in Section 3.4. There are several comments worth making regarding this discretization:

1. The use of an implicit scheme allows for large timesteps to be taken. For all test problems presented in this section, we choose Δt to be:

$$\Delta t = \frac{t_{\text{final}}}{2}\left\lceil \frac{t_{\text{final}}}{\Delta x}\right\rceil^{-1}, \tag{47}$$

   where $\lceil\cdot\rceil$ is the *ceiling* function, so that $\Delta t \approx \Delta x/2$.
2. The fact that $\mathcal{L}$ depends on Δt implies that the Schur-complement depends on Δt as well. This means that the Schur-complement must be reformed and refactored whenever Δt is changed, complicating the use of a method that uses adaptive timestepping.
3. Modifying this formulation to solve the heat equation with Neumann and Robin boundary conditions may be done in the same way as shown for the Poisson equation in Section 4.3.

$n \times n$	IBSE-1		IBSE-2		IBSE-3		timesteps to $t = 0.1$
	prep time	timestep	prep time	timestep	prep time	timestep	
32×32	674	39	895	39	1451	43	2
64×64	957	34	1776	39	2371	39	3
128×128	1310	30	2207	31	3241	37	5
256×256	1833	21	2833	24	3917	23	9
512×512	3478	21	4493	22	5782	20	17
1024×1024	5328	19	7521	19	10187	19	33
2048×2048	9951	18	13707	18	18686	18	66

Table 2: The computational time, normalized by the time to compute an FFT, required for precomputation (labeled 'prep time'), and per timestep of Equation (48) (labeled 'timestep'). We note that even for $n = 2048$, the 'prep time', shown here as requiring 18686 FFTs, requires only 50 minutes of wall time using serial code that has not been carefully optimized. See text for further discussion. The number of timesteps taken to advance Equation (48) from $t = 0$ to $t = 0.1$ is also shown.

5.1. Solution in a periodic domain outside an obstacle

In this section we demonstrate high-order convergence to a heat-equation set on a periodic domain with a simple obstacle in it. To provide direct numerical comparison with results from the Active Penalty (AP) method, the following problem is from Section 6.2 of [26]:

$$u_t - \Delta u = \left[\cos y + e^{\sin x}\left(\sin x - \cos^2 x\right)\right]\cos t - \left(e^{\sin x} + \cos y\right)\sin t \qquad \text{in } \Omega, \tag{48a}$$

$$u(x, y, t = 0) = e^{\sin x} + \cos y \qquad \text{in } \Omega, \tag{48b}$$

$$u = (e^{\sin x} + \cos y)\cos t \qquad \text{on } \Gamma, \tag{48c}$$

where the physical domain is $\Omega = \mathbb{T}^2 \setminus B_{1/4}(\pi, \pi)$. Equation (48) has the analytic solution

$$u_a = (e^{\sin x} + \cos y)\cos t. \tag{49}$$

The initial condition is integrated from $t = 0$ to $t = 0.1$ by solving Equation (45b) at each timestep. Startup values for BDF4 are computed by evaluating the analytic solution at $t = -\Delta t$, $t = -2\Delta t$, and $t = -3\Delta t$. In Figure 8, L^∞ errors at $t = 0.1$ for solutions generated with the IBSE-1, IBSE-2, and IBSE-3 methods are shown, demonstrating second-, third-, and fourth-order convergence in space and time, respectively. On this plot, we also show errors from the second- and third-order AP methods with $n = 512$, the finest resolution reported in [26]. Our method yields errors lower by several orders of magnitude.

In contrast to the AP method [26], we are able to use an *implicit* time discretization, enabling the use of large timesteps. The ability to take large timesteps efficiently comes at the cost of a significant precomputation for the IBSE method, a cost that the AP method does not incur. This precomputation prevents the IBSE method as described in this paper from being efficiently applied to moving boundary problems, a limitation not faced by the AP method.

In Table 2, we provide the computational time required by the IBSE-1, IBSE-2, and IBSE-3 methods to setup the problem (dominated by the formation and factorization of Equation (32)), the time required to take one timestep, and the number of timesteps taken to advance the equation from $t = 0$ to $t = 0.1$. Computational times are given as a multiple of the time required to take one FFT. In our code, 8 FFTs are used per timestep regardless of k; the time required to take a timestep does not vary significantly across the methods. The precomputation time increases approximately linearly in k, but even for the finest discretization presented, the precomputation time is not prohibitive: for $n = 2048$ and the IBSE-3 method, the precomputation requires about 50 minutes using serial code that has not been carefully optimized.

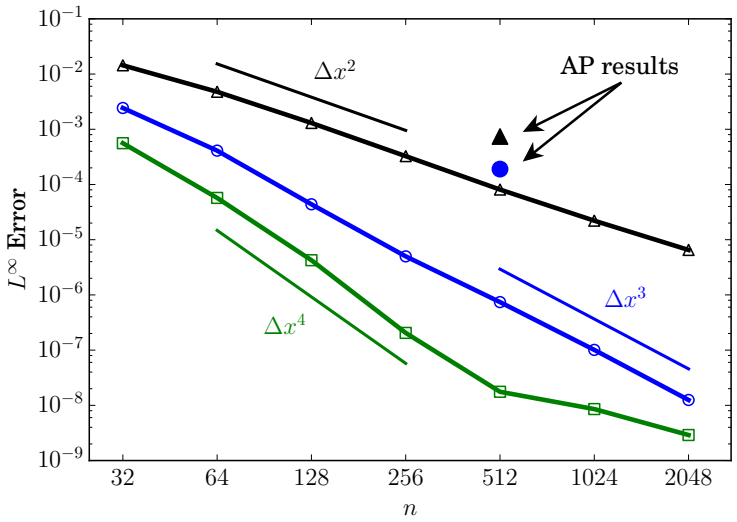


Figure 8: L^∞ error for solutions to Equation (48) produced using IBSE-1 (△), IBSE-2 (○), and IBSE-3 (□), demonstrating $\mathcal{O}(\Delta x^2)$, $\mathcal{O}(\Delta x^3)$, and $\mathcal{O}(\Delta x^4)$ convergence, respectively. Results from [26] are shown for $n = 512$, for the second- (▲) and third-order(●) Active Penalty methods [26].

5.2. Solution inside a parametrically defined region

In this section, we demonstrate high-order convergence to a heat equation set inside a parametrically defined domain. This problem is from Section 6.1 of [24], allowing direct comparison of the IBSE method with the Fourier Continuation method [24]. For convenience, we rescale the domain of the problem. We will solve

$$u_t - \Delta u = \pi\left(2 - \Delta\phi\right)\cos(\pi\phi) + \pi^2\left(\phi_x^2 + \phi_y^2\right)\sin(\pi\phi) \qquad \text{in } \Omega, \tag{50a}$$

$$u(x, y, t = 0) = \sin(\pi\phi(x, y, t = 0)) \qquad \text{in } \Omega, \tag{50b}$$

$$u = \sin(\pi\phi) \qquad \text{on } \Gamma, \tag{50c}$$

where

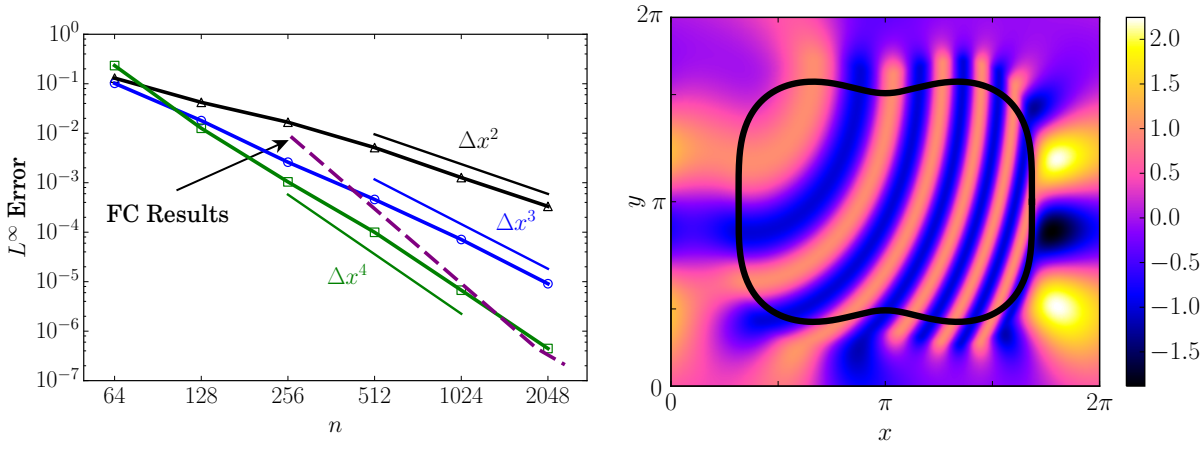
$$\phi(x, y, t) = 9\left(\frac{x - \pi}{4} + 1\right)^2 + 4\left(\frac{y - \pi}{4} + 1\right)^2 + 2t. \tag{51}$$

The domain Ω is the region inside the boundary Γ defined by the parametric equations:

$$x(\theta) = \left(10\sin^2(2\theta) + 3\cos^3(2\theta) + 40\right)\cos\theta/20 + \pi, \tag{52a}$$

$$y(\theta) = \left(10\sin^2(2\theta) + 3\cos^3(2\theta) + 40\right)\sin\theta/20 + \pi, \tag{52b}$$

for $0 \leq \theta < 2\pi$. Equation (50) has the analytic solution $u = \sin(\pi\phi)$. The initial condition is integrated from $t = 0$ to $t = 0.01$ by solving Equation (45b) at each timestep. Startup values for BDF4 are computed by evaluating the analytic solution at $t = -\Delta t$, $t = -2\Delta t$, and $t = -3\Delta t$. In Figure 9, we report the maximum error over space and time, demonstrating second-, third-, and fourth-order convergence in space and time with the IBSE-1, IBSE-2, and IBSE-3 methods, respectively. Figure 9b shows a solution to Equation (50), together with its C^1 extension, computed using IBSE-1 and $n = 2^9$ at $t = 1.0$. Also shown in Figure 9a are the results produced by the FC method from [24]. The errors for the FC method are reported using Δx in [25]; in Figure 9 we plot the results as a function of n so that the values of Δx are equivalent (accounting for the rescaling of the problem). For both the IBSE and FC methods, the timestep has been taken low



(a) refinement study

(b) solution with extension

Figure 9: Figure 9a shows L^∞ errors for Equation (50) at $t = 1.0$ produced by the IBSE-1 (△), IBSE-2 (○), and IBSE-3 (□) methods, demonstrating $\mathcal{O}(\Delta x^2)$, $\mathcal{O}(\Delta x^3)$, and $\mathcal{O}(\Delta x^4)$ convergence, respectively. The dashed line shows results from the Fourier Continuation method [24]; the value of 'n' at which we plot their results is set so that the grid-spacing Δx is the same for the FC and IBSE methods, taking into account our rescaling of the domain. Figure 9b shows the numerical solution to Equation (50) computed with the ISBE-1 method and $n = 2^9$ at $t = 1.0$. The boundary $\Gamma = \partial\Omega$ of the domain Ω is shown in black. Note that the physical domain Ω is contained inside the boundary.

enough so that the dominant error is the spatial error. Despite the fact that the FC method is fifth-order in space, the IBSE-3 method is able to achieve comparable errors at the finest grid-spacing reported[1]. For all of the FC results, the timestep is taken to be 1×10^{-5}. Although their Alternating-Direction Implicit (ADI) scheme is unconditionally stable, it is only second-order accurate, and thus requires the use of small timesteps to obtain errors small enough to not impede the rapid convergence of the spatial discretization. The ADI nature of their scheme complicates implementation of higher-order schemes, although this issue may be resolved using high-order Richardson extrapolation [23]. The structure of the IBSE method allows for the use of simple and high-order timestepping methods. In these computations, we use BDF4 and are able to take substantially larger timesteps while still obtaining comparable error.

6. Results: nonlinear problems

One significant advantage of the IBSE method is the simplicity with which the elliptic and parabolic equation solvers can be integrated into methods to solve more complicated PDE. We solve two nonlinear problems, the Burgers' equation and the Fitzhugh-Nagumo equations, as a demonstration. The IBSE method is able to obtain high-order convergence in both space and time using simple timestepping methods and a straightforward pseudo-spectral computation of nonlinearities.

[1]The method used in [25] was based on an oversampled formulation near the boundary. Newer FC methods [44] do not require this oversampling, effectively halving the number of discretization points in each dimension required to obtain a given accuracy.

Solutions	L^∞ difference	ratio	$\log_2$ ratio
2^6, 2^7	3.65e-02	-	-
2^7, 2^8	9.75e-03	3.7	1.9
2^8, 2^9	6.87e-04	14.2	3.8
2^9, 2^{10}	2.83e-05	24.3	4.6

Table 3: Unnormalized L^∞ differences between solutions to Equation (53) at successive levels of grid refinement, computed using IBSE-3. Ratios of the L^∞ differences are computed, indicating fourth-order convergence in space and time.

6.1. Burgers' equation

In this section, we apply the IBSE method to a two-dimensional homogeneous Burgers' equation, a simple model with a nonlinearity that has the same form as Navier-Stokes:

$$u_t - \nu\Delta u + u\cdot\nabla u = 0 \qquad \text{in } \Omega, \tag{53a}$$
$$u = 0 \qquad \text{on } \Gamma, \tag{53b}$$
$$u(x,y,t=0) = \psi(x,y), \tag{53c}$$

Here Ω is the region *outside* of the polar region defined by $r = 1+\cos(\theta+\pi/4)/4$, shifted by $(3\pi/2, 3\pi/2)$; this domain is shown in Figure 10a. The initial condition is given by the Gaussian pulse $\psi(x,y) = 2e^{-40(x-2.5)^2}e^{-40(y-4.3)^2}$, and $\nu = 0.01$. As with the other examples, the computational domain C is taken to be $\mathbb{T}^2 = [0,2\pi]\times[0,2\pi]$.

We integrate this equation in time using a fourth-order implicit-explicit (IMEX) Backward Differentiation formula [45]:

$$\frac{25}{12}u^{n+1} - 4u^n + 3u^{n-1} - \frac{4}{3}u^{n-2} + \frac{1}{4}u^{n-3} = \\ \Delta t\left[\mathcal{I}(u^{n+1}) + 4\mathcal{E}(u^n) - 6\mathcal{E}(u^{n-1}) + 4\mathcal{E}(u^{n-2}) - \mathcal{E}(u^{n-3})\right], \tag{54}$$

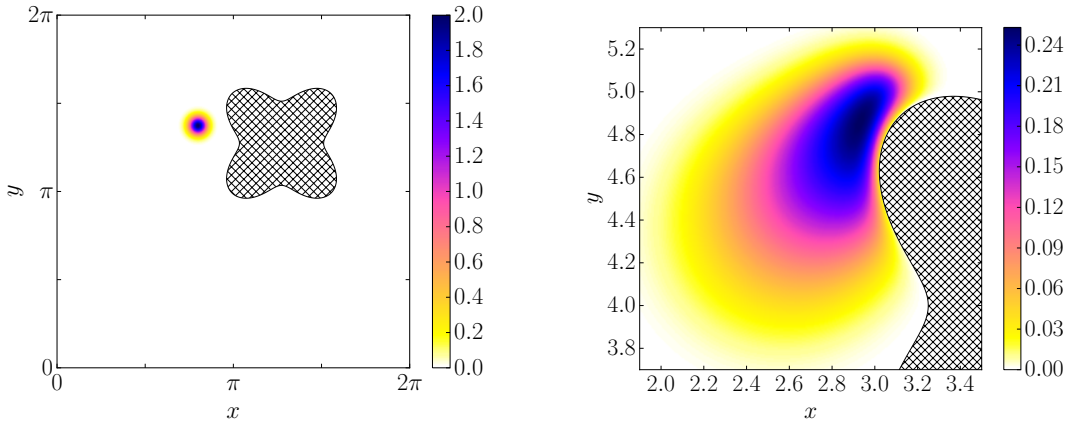
where $\mathcal{I}$ evaluates the *stiff* terms in u_t and $\mathcal{E}$ evaluates the *non-stiff* terms in u_t. For this problem, $\mathcal{I}(u) = \nu\Delta u$ and $\mathcal{E}(u) = -u\cdot\nabla u$. In practice, timestepping involves the explicit computation of a forcing function:

$$\frac{25}{12}f = 4u^n - 3u^{n-1} + \frac{4}{3}u^{n-2} - \frac{1}{4}u^{n-3} + \Delta t\left[4\mathcal{E}(u^n) - 6\mathcal{E}(u^{n-1}) + 4\mathcal{E}(u^{n-2}) - \mathcal{E}(u^{n-3})\right], \tag{55}$$

followed by solution of the equation

$$\left(\mathbb{I} - \frac{12}{25}\nu\Delta t\Delta\right)u^{n+1} = f \qquad \text{in } \Omega, \tag{56a}$$
$$u^{n+1} = 0 \qquad \text{on } \Gamma. \tag{56b}$$

Since u^n is smooth in the entire computational domain C, the explicit terms such as $\mathcal{E}(u^n)$ may be accurately computed using simple pseudo-spectral methods. Equation (56) is solved using the method described in Section 3.4. For this test, the startup values u^{-1}, u^{-2}, and u^{-3} are all taken to be ψ. We integrate Equation (53) to $t = 2$ with a timestep of $\Delta t = \Delta x/20$, for $n = 2^6$ to $n = 2^{10}$, using IBSE-3 when solving Equation (56). Figure 10a and Figure 10b show the initial condition and a zoom of the solution at $t = 2$ for $n = 2^{10}$. Convergence is assessed by comparing the ratio of the L^∞ difference between solutions at different resolutions, computed over the spatial locations in the coarsest grid. Results are shown in Table 3, demonstrating fourth-order convergence in both space and time.



(a) initial condition and geometry

(b) zoom of solution at $t = 2$

Figure 10: The initial condition, geometry, and solution at $t = 2.0$ to Equation (53). The extension domain E is indicated by the hatched region. The physical domain Ω is everything outside of E.

6.2. Fitzhugh-Nagumo equation

The Fitzhugh-Nagumo equation [46] is a system of nonlinear reaction-diffusion equations often used to model excitable media in biological systems (see e.g. [47]). We solve the Fitzhugh-Nagumo equations with homogeneous Neumann boundary conditions and an initially unperturbed recovery variable w:

$$v_t - \nu\Delta v + v(a - v)(1 - v) + w = 0 \qquad \text{in } \Omega, \tag{57a}$$

$$w_t + \epsilon(\gamma w - v) = 0 \qquad \text{in } \Omega, \tag{57b}$$

$$\frac{\partial v}{\partial n} = 0 \qquad \text{on } \Gamma \tag{57c}$$

$$v(x, y, t = 0) = \psi(x, y) \tag{57d}$$

$$w(x, y, t = 0) = 0. \tag{57e}$$

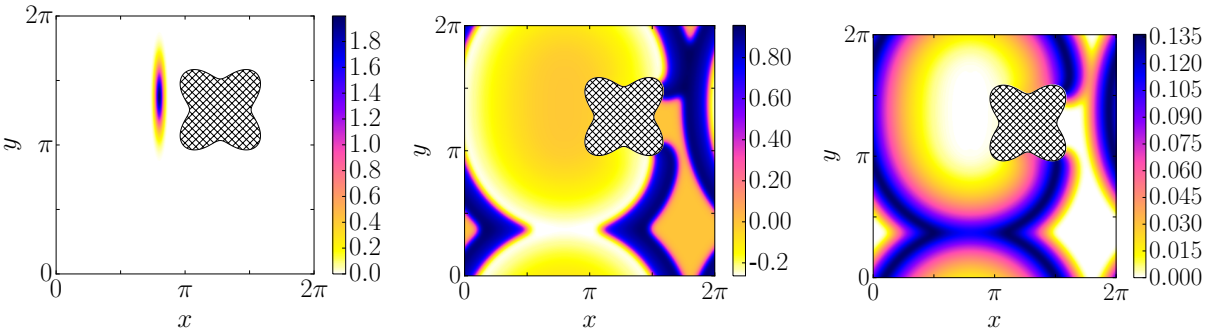
The physical domain Ω and the computational domain C are the same as those used for the solution to Burgers' equation in Section 6.1. The initial data is the pulse $\psi(x, y) = 2e^{-100(x-2.5)^2}e^{-10(y-4.3)^2}$. High-order time integration is implemented using the same IMEX-BDF scheme used in Section 6.1, with the implicit terms given by

$$\mathcal{I}\begin{pmatrix} v \\ w \end{pmatrix} = \begin{pmatrix} \nu\Delta v \\ 0 \end{pmatrix} \tag{58}$$

and the explicit terms given by

$$\mathcal{E}\begin{pmatrix} v \\ w \end{pmatrix} = \begin{pmatrix} v(a - v)(v - 1) - w \\ \epsilon(v - \gamma w). \end{pmatrix} \tag{59}$$

We set the parameters to be $a = 0.1$, $\gamma = 2$, $\epsilon = 0.005$, and $\nu = 0.001$. For this test, the startup values v^{-1}, v^{-2}, and v^{-3} are all taken to be ψ. We integrate Equation (57) to $t = 200$ with a timestep of $\Delta t = \Delta x/2$, for $n = 2^6$ to $n = 2^{10}$ with the IBSE-3 method when solving for the updated value of v. Figure 11 shows the initial condition ψ, as well as the solutions v and w at time $t = 200$. Convergence is assessed by comparing the ratio of the L^∞ difference between solutions at different resolutions, computed over the spatial locations in the coarsest grid. Results are shown in Table 4, demonstrating at least third-order convergence in both space and time, as expected when solving with Neumann boundary conditions.



(a) initial condition and geometry (b) v at $t = 200$ (c) w at $t = 200$

Figure 11: The initial condition for v, and the solutions to v and w at $t = 200$ for Equation (57) generated with the IBSE-3 method. The extension domain E is indicated by the hatched region. The physical domain Ω is everything outside of E.

Solutions	L^∞ difference	ratio	$\log_2$ ratio
2^6, 2^7	2.23e-01	-	-
2^7, 2^8	2.38e-02	9.4	3.2
2^8, 2^9	1.86e-03	12.8	3.7
2^9, 2^{10}	2.68e-04	6.9	2.8

(a) Refinement study for v

Solutions	L^∞ difference	ratio	$\log_2$ ratio
2^6, 2^7	1.63e-02	-	-
2^7, 2^8	2.25e-03	7.3	2.9
2^8, 2^9	1.91e-04	11.8	3.6
2^9, 2^{10}	8.68e-06	22.0	4.5

(b) Refinement study for w

Table 4: Unnormalized L^∞ differences between solutions to Equation (57) at successive levels of grid refinement, computed with IBSE-3. Ratios of the L^∞ differences are computed, indicating at least third-order convergence in space and time.

7. Discussion

We have developed a new numerical method, the IBSE method, for solving elliptic and parabolic PDE on general smooth domains to an arbitrarily high order of accuracy using simple Fourier spectral methods. Accuracy is obtained by forcing the solution to be globally smooth on the entire computational domain. This is accomplished by solving for the smooth extension of the unknown solution, enabling the method to directly solve elliptic equations. Remarkably, this computation can be effectively reduced to a small, dense system of equations, and the cost to invert this system can be minimized by precomputing and storing its LU-factorization. This provides a very efficient algorithm for implicit timestepping on stationary domains. The method requires minimal geometric information regarding the boundary: only its position, normals, and an indicator variable denoting whether points in the computational grid are inside of the physical domain Ω or not, enabling simple and robust code. The method is flexible enough to allow high-order discretization in space and time for a wide range of nonlinear PDE using straightforward implicit-explicit timestepping schemes.

The IBSE method shares some similarities with Fourier Continuation (FC) [23–25] and Active Penalty (AP) [26] methods. The FC method uses Fourier methods to obtain a high-order discretization, and the idea that smooth, non-periodic functions can be turned into smooth and periodic functions by extending them into a larger domain in some appropriate way. There are two key differences between the FC method and the IBSE method. The FC method (*i*) uses dimensional splitting to reduce the problem to a set of one-dimensional problems and (*ii*) relies on data from the previous timestep in order to generate the Fourier continuations that allow for their high-order spatial accuracy. This forces the FC method to use an Alternating-Direction Implicit scheme to take large stable timesteps when solving parabolic equations, complicating the implementation of high-order timestepping. In addition, the FC method requires the use of an iterative solver for computing solutions to the Poisson equation, complicating an efficient discretization of the *incompressible* Navier-Stokes equations, although the FC method has been used to solve the compressible Navier-Stokes equations to high order [23]. In contrast to the conditioning issues faced by the IBSE method, the one-dimensional nature of the Fourier-continuation problem allows the FC method to use high-precision arithmetic in certain precomputations to effectively eliminate the conditioning problem inherent in constructing smooth extensions [48]. This enables the FC method to obtain stable methods that converge to higher order than can be achieved by the IBSE method in double-precision arithmetic.

The AP method, like the FC method, relies on data from previous timesteps in the way that it imposes smoothness on the solution of the PDE. A large drag force is applied that penalizes deviations of the solution in the extension region from the smooth extension of the solution at the previous timestep. This dependence on data from previous timesteps forces the AP method to use *explicit* timestepping when solving parabolic equations. The smoothness constraint is also enforced only *approximately*; solutions are C^1 regardless of how many derivatives are matched when generating the extension functions. We believe this property explains both the relatively small difference in L^∞ error between the second-order AP method and the second-order IBSE-1 method, and the orders of magnitude difference in L^∞ error between the third-order AP method and the third-order IBSE-2 method observed in solutions to Equation (48). Despite these disadvantages, the AP method does not require the solution of a dense system of equations and should be immediately applicable to moving boundary problems.

We have only implemented two-dimensional examples in this paper. The IBSE method extends to three dimensions without any changes, although there is one minor difficulty that must be resolved: the production of an accurate enough quadrature for the discretization of the boundary integrals in the spread operators S and T_k. For two-dimensional problems, this comes nearly for free since the simple quadrature rule given in Section 3.1 is spectrally accurate for closed one-dimensional curves. High-order quadrature rules for two-dimensional surfaces are more complicated but well-developed, and high-order surface representation has been incorporated into the Immersed Boundary method [49, 50].

We have discretized the IBSE-k method for $1 \leq k \leq 3$. In this work, we are limited to $k = 3$ largely by the *accuracy and regularity of our regularization of* δ. This limitation is not fundamental. There are at least two paths forward to obtain higher-order accuracy. Smoother and more accurate δ-functions with compact support could be constructed, however the method that we use to construct $\tilde{\delta}$ (convolving δ-functions with

limited regularity against themselves) would most likely not be of use here. For example, to discretize the IBSE-5 method, we could start with the sixth-order analog to the base δ-function that we use (see Section 3.2), which has a support of six grid points [35]. To obtain a C^5 δ-function, this would need to be convolved against itself 5 times, resulting in a δ with a support of 30 gridpoints (900 for a two-dimensional δ-function). An alternate approach is to use non-compactly supported δ-functions, combined with fast transform methods for the application of the spread and interpolation operators [36, 37].

When solving the IBSE-k equations for high values of k, the conditioning of the extension operator $\mathcal{H}^k$ given in Equation (7) becomes problematic. Although the conditioning can be controlled by an appropriate choice of Θ, the length scales introduced by Θ rapidly become unresolvable as the discretization is refined. A different choice of extension operator, e.g. a polynomial of the Laplacian whose spectra is explicitly designed to minimize the condition number while limiting the introduction of fine scales, could potentially be used to mitigate this problem.

In this manuscript, we have developed two of the necessary components for the implementation of a high-order numerical scheme for the incompressible Navier-Stokes equations based on the IBSE method: a direct solver for the Poisson equation and a high-order implicit-explicit discretization of a nonlinear advection-diffusion equation. The missing component is *how to impose a global divergence constraint* without compromising the accuracy of the discretization. This is a non-trivial problem even for finite-difference discretizations on curvilinear domains [51]. We have obtained preliminary results for solving the Stokes equation using the IBSE method; this work will be presented in a forthcoming contribution.

Finally, the high-efficiency that we obtain is only currently achievable on stationary domains. This is because the inversion method that we use in this paper for the IBSE method relies on an expensive precomputation that depends on the physical domain and the discretization. However, there is no fundamental obstacle to the application of the IBSE method to problems with moving domains. Instead, the challenge is to find a robust and efficient method to invert the IBSE system in Equation (18) that does not require substantial precomputation. Recent progress has been made for preconditioning similar systems of equations for the simulation of rigid-body motion in an Immersed Boundary framework [4]. The integration of these ideas into the IBSE method to allow for simulation of moving boundary problems will be an area of active future research.

Acknowledgements

This work was supported in part by the National Science Foundation under Grant DMS-1160438. The authors would like to thank Grady Wright for useful discussions regarding numerical conditioning problems that helped lead to the form of the operator $\mathcal{H}^k$ and the definition of Θ^* in Section 3.3.

References

[1] Charles S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002. ISSN 0962-4929. doi: 10.1017/S0962492902000077.

[2] Rajat Mittal and Gianluca Iaccarino. Immersed Boundary Methods. *Annual Review of Fluid Mechanics*, 37(1):239–261, 2005. ISSN 0066-4189. doi: 10.1146/annurev.fluid.37.061903.175743.

[3] Sarah D Olson and Anita T Layton. Simulating biofluid-structure interactions with an immersed boundary framework–a review. *Biological Fluid Dynamics: Modeling, Computations, and Applications*, 628:1, 2014.

[4] Bakytzhan Kallemov, A Bhalla, Boyce E Griffith, and Aleksandar Donev. An immersed boundary method for rigid bodies. *arXiv preprint arXiv:1505.07865*, 2015.

[5] Kunihiko Taira and Tim Colonius. The immersed boundary method: A projection approach. *Journal of Computational Physics*, 225(2):2118–2137, 2007. ISSN 00219991. doi: 10.1016/j.jcp.2007.03.005.

[6] Joseph M Teran and Charles S Peskin. Tether force constraints in Stokes flow by the immersed boundary method on a periodic domain. *SIAM Journal on Scientific Computing*, 31(5):3404–3416, 2009.

[7] Zhilin Li and Kazufumi Ito. *The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains*, volume 33. Siam, 2006.

[8] Ronald P Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A Non-oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method). *Journal of Computational Physics*, 152(2):457–492, 1999. ISSN 00219991. doi: 10.1006/jcph.1999.6236. URL http://www.sciencedirect.com/science/article/pii/S0021999199962368.

[9] Philippe Angot, Charles-Henri Bruneau, and Pierre Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik*, 81(4):497–520, 1999. ISSN 0029-599X. doi: 10.1007/s002110050401.

[10] Ming-Chih Lai and Charles S. Peskin. An Immersed Boundary Method with Formal Second-Order Accuracy and Reduced Numerical Viscosity. *Journal of Computational Physics*, 160(2):705–719, 2000. ISSN 00219991. doi: 10.1006/jcph.2000.6483. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999100964830.

[11] Andreas Mark and Berend G M van Wachem. Derivation and validation of a novel implicit second-order accurate immersed boundary method. *Journal of Computational Physics*, 227(13):6660–6680, 2008. ISSN 00219991. doi: 10.1016/j.jcp.2008.03.031.

[12] Mark N. Linnick and Hermann F. Fasel. A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains. *Journal of Computational Physics*, 204(1):157–192, 2005. ISSN 00219991. doi: 10.1016/j.jcp.2004.09.017.

[13] Jian Kang Liu and Zhou Shun Zheng. Efficient high-order immersed interface methods for heat equations with interfaces. *Applied Mathematics and Mechanics*, 35(51174236):1189–1202, 2014. ISSN 02534827. doi: 10.1007/s10483-014-1851-6.

[14] Sheng Xu and Z. Jane Wang. An immersed interface method for simulating the interaction of a fluid with moving boundaries. *Journal of Computational Physics*, 216(2):454–493, 2006. ISSN 00219991. doi: 10.1016/j.jcp.2005.12.016.

[15] Xiaolin Zhong. A new high-order immersed interface method for solving elliptic equations with imbedded interface of discontinuity. *Journal of Computational Physics*, 225(1):1066–1099, 2007. ISSN 00219991. doi: 10.1016/j.jcp.2007.01.017.

[16] S Yu, Y Zhou, and G Wei. Matched interface and boundary (MIB) method for elliptic problems with sharp-edged interfaces. *Journal of Computational Physics*, 224(2):729–756, 2007. ISSN 00219991. doi: 10.1016/j.jcp.2006.10.030. URL http://dx.doi.org/10.1016/j.jcp.2006.10.030.

[17] Y. C. Zhou, Shan Zhao, Michael Feig, and G. W. Wei. High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. *Journal of Computational Physics*, 213(1):1–30, 2006. ISSN 00219991. doi: 10.1016/j.jcp.2005.07.022.

[18] Frédéric Gibou and Ronald 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):577–601, 2005. ISSN 00219991. doi: 10.1016/j.jcp.2004.07.018.

[19] John P. Boyd. Fourier embedded domain methods: extending a function defined on an irregular region to a rectangle so that the extension is spatially periodic and C^{∞}. *Applied Mathematics and Computation*, 161(2):591–597, 2005. ISSN 00963003.

[20] Alfonso Bueno-Orovio. Fourier embedded domain methods: Periodic and c^{∞} extension of a function defined on an irregular region to a rectangle via convolution with gaussian kernels. *Applied Mathematics and Computation*, 183(2):813–818, 2006. ISSN 00963003. doi: 10.1016/j.amc.2006.06.029.

[21] S. H. Lui. Spectral domain embedding for elliptic PDEs in complex domains. *Journal of Computational and Applied Mathematics*, 225(2):541–557, 2009. ISSN 03770427. doi: 10.1016/j.cam.2008.08.034. URL http://dx.doi.org/10.1016/j.cam.2008.08.034.

[22] Feriedoun Sabetghadam, Shervin Sharafatmandjoor, and Farhang Norouzi. Fourier spectral embedded boundary solution of the Poisson's and Laplace equations with Dirichlet boundary conditions. *Journal of Computational Physics*, 228(1):55–74, 2009. ISSN 00219991. doi: 10.1016/j.jcp.2008.08.018. URL http://dx.doi.org/10.1016/j.jcp.2008.08.018.

[23] Nathan Albin and Oscar P. Bruno. A spectral FC solver for the compressible NavierStokes equations in general domains I: Explicit time-stepping. *Journal of Computational Physics*, 230(16):6248–6270, July 2011. ISSN 00219991. doi: 10.1016/j.jcp.2011.04.023. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999111002695.

[24] Mark Lyon and Oscar P. Bruno. High-order unconditionally stable FC-AD solvers for general smooth domains I. Basic Elements. *Journal of Computational Physics*, 229(9):3358–3381, 2010. ISSN 00219991. doi: 10.1016/j.jcp.2010.01.006. URL http://dx.doi.org/10.1016/j.jcp.2009.11.020.

[25] Mark Lyon and Oscar P. Bruno. High-order unconditionally stable FC-AD solvers for general smooth domains II. Elliptic,

parabolic and hyperbolic PDEs; theoretical considerations. *Journal of Computational Physics*, 229(9):3358–3381, 2010. ISSN 00219991. doi: 10.1016/j.jcp.2010.01.006. URL http://dx.doi.org/10.1016/j.jcp.2010.01.006.

[26] David Shirokoff and J-C Nave. A Sharp-Interface Active Penalty Method for the Incompressible Navier-Stokes Equations. *Journal of Scientific Computing*, 62(1):53—-77, 2015. URL http://arxiv.org/abs/1303.5681.

[27] Richard P. Beyer and Randall J. LeVeque. Analysis of a one-dimensional model for the immersed boundary method, 1992. URL http://epubs.siam.org/doi/abs/10.1137/0729022.

[28] Arturo Pacheco-Vega, J. Rafael Pacheco, and Tamara Rodic. A General Scheme for the Boundary Conditions in Convective and Diffusive Heat Transfer With Immersed Boundary Methods. *Journal of Heat Transfer*, 129(11):1506, 2007. ISSN 00221481. doi: 10.1115/1.2764083.

[29] John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[30] Fritz John. Partial differential equations, volume 1 of applied mathematical sciences, 1982.

[31] Boyce E. Griffith and Charles S. Peskin. On the order of accuracy of the immersed boundary method: Higher order convergence rates for sufficiently smooth problems. *Journal of Computational Physics*, 208(1):75–105, 2005. ISSN 00219991. doi: 10.1016/j.jcp.2005.02.011.

[32] Yang Liu and Yoichiro Mori. Properties of discrete Delta Functions and local convergence of the immersed boundary method. *SIAM Journal on Numerical Analysis*, 50(6):2986–3015, 2012. ISSN 0036-1429. doi: 10.1137/110836699. URL http://epubs.siam.org/doi/abs/10.1137/110836699.

[33] Xiaolei Yang, Xing Zhang, Zhilin Li, and Guo-Wei He. A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations. *Journal of Computational Physics*, 228(20):7821–7836, November 2009. ISSN 00219991. doi: 10.1016/j.jcp.2009.07.023. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999109004136.

[34] Anna-Karin Tornberg and Björn Engquist. Numerical approximations of singular source terms in differential equations. *Journal of Computational Physics*, 200(2):462–488, November 2004. ISSN 00219991. doi: 10.1016/j.jcp.2004.04.011. URL http://linkinghub.elsevier.com/retrieve/pii/S0021999104001767.

[35] Thomas T Bringley. *Analysis of the Immersed Boundary Method for Stokes Flow*. PhD thesis, 2008.

[36] Leslie Greengard and John Strain. The Fast Gauss Transform. *Statistics and Computing*, 12(1):79–94, 1991.

[37] Leslie Greengard, June-Yub Lee, and Souheil Inati. The Fast Sinc Transform and Image Reconstruction from Nonuniform Samples in k-Space. *Communications in Applied Mathematics and Computational Science*, 1(1):121–131, 2006.

[38] Yuan-xun Bao, Jason Kaye, and Charles S Peskin. A Gaussian-Like Immersed Boundary Kernel with Improved Translational Invariance and Smoothness. *arXiv preprint arXiv:1505.07529*, pages 1–8, 2015.

[39] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999. ISBN 0-89871-447-8 (paperback).

[40] Sivaram Ambikasaran and Eric Darve. The Inverse Fast Multipole Method. *arXiv preprint arXiv:1407.1572*, pages 1–25, 2014. URL http://arxiv.org/abs/1407.1572.

[41] Sivaram Ambikasaran and Eric Darve. An $\mathcal{O}(N \log N)$ fast direct solver for partial hierarchically semi-separable matrices *. *Journal of Scientific Computing*, 57(3):477–501, 2013.

[42] JW Ruge and Klaus Stüben. Algebraic multigrid. *Multigrid methods*, 3:73–130, 1987.

[43] Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

[44] O. P. Bruno and a. Prieto. Spatially Dispersionless, Unconditionally Stable FCAD Solvers for Variable-Coefficient PDEs. *Journal of Scientific Computing*, 58(2):331–366, 2014. ISSN 0885-7474. doi: 10.1007/s10915-013-9734-8. URL http://link.springer.com/10.1007/s10915-013-9734-8.

[45] Willem Hundsdorfer and Steven J. Ruuth. IMEX extensions of linear multistep methods with general monotonicity and boundedness properties. *Journal of Computational Physics*, 225(2):2016–2042, 2007. ISSN 00219991. doi: 10.1016/j.jcp.2007.03.003.

[46] J. Nagumo, S. Arimoto, and S. Yoshizawa. An Active Pulse Transmission Line Simulating Nerve Axon. *Proceedings of the IRE*, 50(10):2061–2070, 1962. ISSN 0096-8390. doi: 10.1109/JRPROC.1962.288235.

[47] Rubin R Aliev and Alexander V Panfilov. A simple two-variable model of cardiac excitation. *Chaos, Solitons & Fractals*, 7(3):293–301, 1996.

[48] Rodrigo B. Platte, Lloyd N. Trefethen, and Arno B. J. Kuijlaars. Impossibility of Fast Stable Approximation of Analytic Functions from Equispaced Samples. *SIAM Review*, 53(2):308–318, 2011. ISSN 0036-1445. doi: 10.1137/090774707.

[49] Boyce E Griffith and Xiaoyu Luo. Hybrid finite difference/finite element version of the immersed boundary method. *Submitted in revised form*, 2012.

[50] Varun Shankar, Grady B. Wright, Aaron L. Fogelson, and R. M. Kirby. A Study of Different Modeling Choices For Simulating Platelets Within the Immersed Boundary Method. pages 1–33, October 2012. URL http://arxiv.org/abs/1210.1885v1.

[51] David Shirokoff and Rodolfo Ruben Rosales. An efficient method for the incompressible Navier-Stokes equations on irregular domains with no-slip boundary conditions, high order up to the boundary. *Journal of Computational Physics*, 230(23):50, 2010. ISSN 00219991. doi: 10.1016/j.jcp.2011.08.011. URL http://arxiv.org/abs/1011.3589.

Appendices

A. Formula for C^3 δ-function with interpolation accuracy of $\mathcal{O}(\Delta x^4)$

We give the formula here for $\tilde{\delta}$, defined in Section 3.2. For brevity, we define $\tilde{\delta}$ for $r \geq 0$, which fully defines the function as $\tilde{\delta}$ is even. The formula for $\tilde{\delta}$ is defined piecewise; each portion is a fifteenth order polynomial. Coefficients are provided in the table below.

	$0 \leq r < 1$	$1 \leq r < 2$	$2 \leq r < 3$	$3 \leq r < 4$	$4 \leq r < 5$	$5 \leq r < 6$	$6 \leq r < 7$	$7 \leq r < 8$	$r >= 8$
r^0	$\frac{12949745023}{20432412000}$	$\frac{3177441629}{5003856000}$	$\frac{21914742667}{35026992000}$	$\frac{4094824493}{17513496000}$	$\frac{-1606651889}{5837832000}$	$\frac{163201885541}{35026992000}$	$\frac{1005507698627}{245188944000}$	$\frac{-5005877248}{1915538625}$	0
r^1	0	$\frac{-171811}{22861440}$	$\frac{-2566373}{38102400}$	$\frac{3468455}{4191264}$	$\frac{301286857}{62868960}$	$\frac{-4590637187}{1089728640}$	$\frac{-14398288259}{1089728640}$	$\frac{421762048}{127702575}$	0
r^2	$\frac{-16459}{30240}$	$\frac{-9089}{17280}$	$\frac{-14339}{51840}$	$\frac{-163307}{285120}$	$\frac{-703993}{95040}$	$\frac{-2987689}{673920}$	$\frac{56979607}{3991680}$	$\frac{-23168}{45045}$	0
r^3	0	$\frac{-64649}{3265920}$	$\frac{-1946191}{5443200}$	$\frac{-3764137}{2993760}$	$\frac{32748677}{8981280}$	$\frac{101232611}{11975040}$	$\frac{-17097977}{2395008}$	$\frac{-2091088}{1403325}$	0
r^4	$\frac{81491}{453600}$	$\frac{141751}{777600}$	$\frac{288599}{777600}$	$\frac{6701891}{4276800}$	$\frac{560257}{1425600}$	$\frac{-4187303}{777600}$	$\frac{78901349}{59875200}$	$\frac{642734}{467775}$	0
r^5	0	$\frac{88517}{5443200}$	$\frac{61633}{3024000}$	$\frac{-93301}{151200}$	$\frac{-1620853}{1360800}$	$\frac{1038857}{604800}$	$\frac{255833}{604800}$	$\frac{-538927}{850500}$	0
r^6	$\frac{-11737}{340200}$	$\frac{-119603}{2332800}$	$\frac{-269467}{2332800}$	$\frac{16957}{1166400}$	$\frac{218939}{388800}$	$\frac{-488741}{2332800}$	$\frac{-5818427}{16329600}$	$\frac{773411}{4082400}$	0
r^7	$\frac{143}{145152}$	$\frac{298727}{45722880}$	$\frac{630773}{15240960}$	$\frac{1057927}{15240960}$	$\frac{-1137851}{9144576}$	$\frac{-152395}{3048192}$	$\frac{1823393}{15240960}$	$\frac{-1825543}{45722880}$	0
r^8	$\frac{3223}{793800}$	$\frac{28127}{5443200}$	$\frac{-7337}{5443200}$	$\frac{-9859}{388800}$	$\frac{7069}{907200}$	$\frac{157289}{5443200}$	$\frac{-966457}{38102400}$	$\frac{58621}{9525600}$	0
r^9	$\frac{-143}{435456}$	$\frac{-30173}{19595520}$	$\frac{-79651}{32659200}$	$\frac{4771}{1306368}$	$\frac{61009}{19595520}$	$\frac{-44227}{6531840}$	$\frac{24421}{6531840}$	$\frac{-69019}{97977600}$	0
r^{10}	$\frac{-3211}{13608000}$	$\frac{-403}{11664000}$	$\frac{7033}{11664000}$	$\frac{91}{2916000}$	$\frac{-247}{243000}$	$\frac{11519}{11664000}$	$\frac{-32227}{81648000}$	$\frac{2447}{40824000}$	0
r^{11}	$\frac{13}{483840}$	$\frac{13}{207360}$	$\frac{-13}{345600}$	$\frac{-221}{2280960}$	$\frac{13}{84480}$	$\frac{-221}{2280960}$	$\frac{53}{1774080}$	$\frac{-299}{79833600}$	0
r^{12}	$\frac{1}{181440}$	$\frac{-1}{155520}$	$\frac{-1}{155520}$	$\frac{7}{427680}$	$\frac{-1}{71280}$	$\frac{1}{155520}$	$\frac{-19}{11975040}$	$\frac{1}{5987520}$	0
r^{13}	$\frac{-1}{1451520}$	$\frac{-1}{2612736}$	$\frac{29}{21772800}$	$\frac{-13}{9580032}$	$\frac{113}{143700480}$	$\frac{-173}{622702080}$	$\frac{1}{17791488}$	$\frac{-47}{9340531200}$	0
r^{14}	$\frac{-1}{25401600}$	$\frac{1}{10886400}$	$\frac{-1}{10886400}$	$\frac{1}{17107200}$	$\frac{-1}{39916800}$	$\frac{1}{141523200}$	$\frac{-1}{838252800}$	$\frac{1}{10897286400}$	0
r^{15}	$\frac{1}{203212800}$	$\frac{-1}{261273600}$	$\frac{1}{435456000}$	$\frac{-1}{958003200}$	$\frac{1}{2874009600}$	$\frac{-1}{12454041600}$	$\frac{1}{87178291200}$	$\frac{-1}{1307674368000}$	0

B. Inversion of the IBSE-k system (18) for $\mathcal{L} = \Delta$

For the special case of a Fourier discretization of the Poisson problem, where $\mathcal{L}$ is the periodic Laplacian, the method of inversion given in Section 3.4 is complicated by the nullspace of Δ. In particular, the main block in Equation (18),

$$\begin{pmatrix} \Delta & -\chi_E \Delta \\ & \mathcal{H}^k \end{pmatrix} \tag{60}$$

is not invertible, and thus the Schur-complement SC (32) cannot be directly formed and prefactored. The resolution of this problem is conceptually simple but computationally involved, so we will give a detailed description for the simpler case of a direct-forcing Immersed Boundary discretization and state the result for

our problem. A direct-forcing IB discretization of the Poisson problem with Dirichlet boundary conditions is

$$\begin{pmatrix} \Delta & S \\ S^* & \end{pmatrix} \begin{pmatrix} u \\ G \end{pmatrix} \begin{pmatrix} f \\ b \end{pmatrix}. \tag{61}$$

Formally, G may be computed by solving

$$(S^*\Delta^{-1}S)G = S^*\Delta^{-1}f - b. \tag{62}$$

Despite the fact that Equation (61) is invertible, the periodic Laplacian has a nullspace, making direct implementation of this approach to compute G impossible. Instead, let us decompose the solution u as

$$u = u_0 - Uv, \tag{63}$$

where u_0 has mean 0, U is a scalar, and $v = \mathbb{1}(\Delta x)^d$, where $\mathbb{1}$ denotes the vector of all ones. We note that v spans the nullspace of the self-adjoint operator Δ, and we have scaled v so that $v^\intercal u$ is equal to the the discrete integral of u. We will also denote the averaging operator $\fint_C w = |C|^{-1}\int_C w$, discretely this is given for C by $\fint_C w = (2\pi)^{-d}v^\intercal w$ and for Γ by $\fint_\Gamma H = (2\pi)^{-1}\Delta s^\intercal H$, where Δs is the vector of quadrature weights on the boundary (see Section 3.1). Plugging the decomposition given in Equation (63) into Equation (61), we find that

$$\Delta u_0 + SG = f. \tag{64}$$

Define a family of pseudo-inverses to Δ, $\{\mathcal{A}_\mu\}_{\mu>0}$, by its Fourier series to be

$$\widehat{\mathcal{A}_\mu f} = \begin{cases} \mu\widehat{f_0} & |k| = 0, \\ \frac{-1}{|k|^2}\widehat{f_k} & |k| \neq 0. \end{cases} \tag{65}$$

Because u_0 has mean 0, $\mathcal{A}_\mu\Delta u_0 = u_0$ for any μ. Applying $\mathcal{A}_\mu$ to Equation (64), we find that $u_0 = \mathcal{A}_\mu f - \mathcal{A}_\mu SG$, and applying S^* to this equation, along with the constraint that $S^*u = b$, yields that

$$S^*\mathcal{A}_\mu SG + S^*(Uv) = S^*\mathcal{A}_\mu f - b, \tag{66}$$

which is a reformulation of the formal equation (62). This is now an equation for the *two* unknowns G and U, and must be supplemented with the additional equation found by projecting $\Delta u + SG = f$ onto the nullspace of Δ, giving $v^\intercal SG = v^\intercal f$. The combination of these two equations forms an augmented Schur-complement system

$$\begin{pmatrix} S^*\mathcal{A}_\mu S & S^*v \\ v^\intercal S & \end{pmatrix} \begin{pmatrix} G \\ U \end{pmatrix} = \begin{pmatrix} S^*\mathcal{A}_\mu f - b \\ v^\intercal f \end{pmatrix} \tag{67}$$

that may be directly formed and prefactored. With (G, U) known, we may compute $u_0 = \mathcal{A}_\mu f - \mathcal{A}_\mu SG$, and finally $u = u_0 - Uv$. This system may be simplified by exploiting the fact that the nullspace of Δ is only constant functions and that $\fint_C SG = \fint_\Gamma G$, giving the equivalent system

$$\begin{pmatrix} S^*\mathcal{A}_\mu S & \beta^{-1}\mathbb{1} \\ \Delta s^\intercal/2\pi & \end{pmatrix} \begin{pmatrix} G \\ U \end{pmatrix} = \begin{pmatrix} S^*\mathcal{A}_\mu f - b \\ \fint_C f \end{pmatrix}, \tag{68}$$

where β is a free parameter. We remark that when Δs is a constant vector, choosing $\beta = n_{\text{bdy}}$ symmetrizes the matrix in Equation (68). Although any choice of μ and β may be used, we have observed empirically that $\mu = 0$ and $\beta = n_{\text{bdy}}$ yields the most well-conditioned system. Once G and U are known, we may compute u to be

$$u = \mathcal{A}_\mu(f - SG) - \beta^{-1}U. \tag{69}$$

Following the same procedure for Equation (18) gives the augmented Schur-complement:

$$\begin{pmatrix} T_k^*(\mathcal{A}_\mu \chi_E \Delta - I)(\mathcal{H}^k)^{-1} T_k & T_k^* \mathcal{A}_\mu S & \beta^{-1} \mathbb{Y}_k \\ S^* \mathcal{A}_\mu \chi_E \Delta (\mathcal{H}^k)^{-1} T_k & S^* \mathcal{A}_\mu S & \beta^{-1} \mathbb{1} \\ \fint_C \chi_E \Delta (\mathcal{H}^k)^{-1} T_k & \fint_C S & \end{pmatrix} \begin{pmatrix} F \\ G \\ \lambda \end{pmatrix} = \begin{pmatrix} S^* \mathcal{A}_\mu \chi_\Omega f - g \\ T_k^* \mathcal{A}_\mu \chi_\Omega f \\ \fint_C \chi_\Omega f \end{pmatrix}. \tag{70}$$

Here $\mathbb{Y}_k$ is defined by $\mathbb{Y}_k = \begin{pmatrix} \mathbb{1}_{n_{\text{bdy}}} & \mathbb{O}_{(k-1)n_{\text{bdy}}} \end{pmatrix}^\intercal$; this form is due to the fact that the estimates of *values* produced by T_k^* are affected by changes in mean while estimates of *normal derivatives* are not. Again, we empirically observe that choosing $\mu = 0$ and $\beta = n_{\text{bdy}}$ provides the most well-conditioned system. Once F and G are known, we can compute ξ as

$$\xi = -(\mathcal{H}^k)^{-1} T_k F, \tag{71}$$

and once ξ is known, u can be computed as

$$u = \mathcal{A}_\mu(\chi_D f + \chi_E \Delta \xi - SG) - \beta^{-1} U. \tag{72}$$