Journal of Computational Physics 229 (2010) 2479–2496

Contents lists available at ScienceDirect

Journal of Computational Physics

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

On the accuracy of direct forcing immersed boundary methods with projection methods

Robert D. Guy [a,*], David A. Hartenstine [b]

[a] Department of Mathematics, University of California Davis, CA 95616, United States
[b] Department of Mathematics, Western Washington University, United States

ARTICLE INFO

Article history:
Received 8 March 2009
Received in revised form 9 October 2009
Accepted 13 October 2009
Available online 21 October 2009

Keywords:
Direct forcing method
Immersed boundary method
Projection methods
Incompressible flow
Navier–Stokes equations

ABSTRACT

Direct forcing methods are a class of methods for solving the Navier–Stokes equations on nonrectangular domains. The physical domain is embedded into a larger, rectangular domain, and the equations of motion are solved on this extended domain. The boundary conditions are enforced by applying forces near the embedded boundaries. This raises the question of how the flow outside the physical domain influences the flow inside the physical domain. This question is particularly relevant when using a projection method for incompressible flow. In this paper, analysis and computational tests are presented that explore the performance of projection methods when used with direct forcing methods. Sufficient conditions for the success of projection methods on extended domains are derived, and it is shown how forcing methods meet these conditions. Bounds on the error due to projecting on the extended domain are derived, and it is shown that direct forcing methods are, in general, first-order accurate in the max-norm. Numerical tests of the projection alone confirm the analysis and show that this error is concentrated near the embedded boundaries, leading to higher-order accuracy in integral norms. Generically, forcing methods generate a solution that is not smooth across the embedded boundaries, and it is this lack of smoothness which limits the accuracy of the methods. Additional computational tests of the Navier–Stokes equations involving a direct forcing method and a projection method are presented, and the results are compared with the predictions of the analysis. These results confirm that the lack of smoothness in the solution produces a lower-order error. The rate of convergence attained in practice depends on the type of forcing method used.

1. Introduction

The immersed boundary (IB) method was developed by Peskin for simulating the coupled motion of an elastic boundary immersed in a viscous fluid [1]. Typically the fluid velocity is discretized on a Cartesian grid and the boundary is discretized using a Lagrangian grid. One reason for the popularity of the IB method is that no internal boundary conditions are required on the immersed boundary. The boundary moves with the local fluid velocity and these deformations generate forces which affect the motion of the fluid.

Goldstein et al. [2] introduced a variation of the IB method for flows around solid objects. The solid objects are embedded in a larger computational domain, and the velocity field is extended throughout the objects. The no-slip condition is enforced

* Corresponding author.
E-mail addresses: guy@math.ucdavis.edu (R.D. Guy), david.hartenstine@wwu.edu (D.A. Hartenstine).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.10.027

on the surface of the objects by applying a body force near the surfaces to bring the velocity to zero. As with the immersed boundary method, complicated domains can be discretized using regular Cartesian grids. A disadvantage of this method is that it requires choosing numerical forcing parameters which can make the equations stiff.

Another type of forcing method was introduced by Mohd-Yusof [3] that does not require numerical forcing parameters. The forcing is computed from the algebraic equations in the discretized problem. This approach avoids the stiffness encountered from the penalty forces used in previous forcing methods. As this method has gained popularity, many variations on how the forcing is applied have appeared in the literature [4–8]. These methods are often referred to as *direct forcing methods*.

Projection methods are often used to enforce the divergence-free constraint in incompressible flow. The momentum equation is advanced in time to give an intermediate velocity that is not required to be divergence-free. This intermediate velocity is then projected onto the subspace of divergence-free fields to find the velocity at the next time step. Performing the projection involves solving a Poisson equation, which when used with a forcing method, is often solved over the extended computational domain. Thus the flow outside the physical domain necessarily influences the flow on the physical domain, even if these two subdomains were effectively decoupled in the solution of the momentum equation.

We note that the projection need not be performed over the extended domain. There are several methods for discretizing the Poisson equation only on the fluid domain using a Cartesian grid, such as the embedded boundary method [9] or the ghost-fluid method [10]. These methods have been used in conjunction with projection methods and immersed boundary methods; see, for example, [11–13]. However, in many, if not most, forcing methods the projection is performed over the entire domain. Further, the projection can be the most computationally expensive step of a fluid solver, and solving on a regular domain allows the straightforward use of fast solvers based on FFTs or geometric multigrid. In this paper we focus on methods in which the projection is performed over the entire domain.

How the flow outside the domain influences the flow in the physical domain is not well understood. In spite of the fact that direct forcing and projection methods are commonly used together successfully, it is not clear that the solution from this combination of methods should converge to the correct solution. Several authors have already commented on this issue. Saiki and Biringen [14] comment that they found it necessary to apply forces on the extension to obtain convergence to the correct solution. Fadlun et al. [4] experimented with different treatments of flow on the extension, and they concluded that the flow on the extension has little effect on the physical flow. Similar observations were reported by Zhang and Zheng [8]. Recently, Domenichini [15] performed a series of numerical tests aimed at understanding the effect of using projection methods and direct forcing methods. The problem has not yet been analyzed to understand why forcing methods do seem to work with projection methods. A deeper understanding is necessary in order to drive the development of more accurate methods.

In this paper we explore the performance of projection methods when the projection is performed over an extended domain. We show that if the intermediate velocity is continuous across the embedded boundaries and close, in some sense, to a divergence-free field, then the error produced by performing the projection on the extended domain is small. We show that the intermediate velocity meets these conditions, and the error introduced by solving on the extended domain is of the same size as the error of the projection method itself.

The intermediate velocity generated using a forcing method generally has a jump in the first derivative near the embedded boundaries. We analyze the spatially discrete projection and show that this lack of smoothness introduces a first-order error during the projection step. Numerical tests of the projection alone show that this error is localized near the embedded boundary, and so higher rates of convergence are achieved in integral norms. These results show that forcing methods are generally first-order accurate in the max-norm, but the accuracy in integral norms depends on how the forcing is performed in the momentum equation.

The remainder of the paper is organized as follows. We briefly describe the main ideas behind direct forcing and projection methods in Sections 2 and 3, respectively. Our analysis of projections on extended domains, including the spatially discrete problem, appears in Section 4. In Section 5, numerical tests validating our analysis and further exploring the accuracy of direct forcing methods are presented. The effect of the smoothness of the extended field on the accuracy of the projection is demonstrated in Section 5.1, and in Section 5.2, the accuracy of projection methods for solving the Navier–Stokes equations is considered. Finally, in Section 6 we describe how our results explain some observations in previous papers, and we discuss some ideas for developing more accurate methods.

2. Direct forcing methods

Direct forcing methods are a class of numerical methods for solving the incompressible Navier–Stokes equations

$$\rho(\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u}) = -\nabla p + \mu\Delta\boldsymbol{u} + \boldsymbol{f}, \tag{1}$$

$$\nabla\cdot\boldsymbol{u} = 0 \tag{2}$$

on a nonrectangular domain using a Cartesian grid. As usual, $\boldsymbol{u}$ is the fluid velocity, p is the pressure, ρ is the density, μ is the dynamic viscosity, and $\boldsymbol{f}$ is a body force density acting on the fluid. The physical domain is embedded in a larger rectangular domain, and the boundary conditions on the physical boundaries are enforced by applying a localized force on the Cartesian grid. The basic idea of choosing the force is very simple. Consider the explicit, discrete-time update for the momentum equation:

$$\frac{\boldsymbol{u}^{n+1}-\boldsymbol{u}^{n}}{\Delta t}=\boldsymbol{G}+\boldsymbol{f}, \tag{3}$$

where the term $\boldsymbol{G}$ represents the pressure gradient, advection terms, and viscous terms. The force is selected so that on the boundary $\boldsymbol{u}^{n+1}=\boldsymbol{U}_b^{n+1}$, where $\boldsymbol{U}_b^{n+1}$ is the given velocity on the boundary at time level $n+1$. The appropriate force is computed from simply rearranging Eq. (3):

$$\boldsymbol{f}=\frac{\boldsymbol{U}_b^{n+1}-\boldsymbol{u}^{n}}{\Delta t}-\boldsymbol{G}. \tag{4}$$

This illustrates the idea behind choosing the force, but to apply the method one must specify where this forcing term is applied and decide how to handle implicit discretizations.

As written, the equation for the force (4) is slightly ambiguous because the velocity $\boldsymbol{u}^n$ is computed on the Cartesian grid and the boundary velocity $\boldsymbol{U}_b^{n+1}$ is given on the embedded boundary which does not align with the Cartesian grid. There are two variations on how this force is computed in practice: on the embedded boundary [6–8,16] and on the Cartesian grid [3–5].

In order to compute the forcing term on the embedded boundary, the velocity from the Cartesian grid, $\boldsymbol{u}^n$, and the explicit terms, $\boldsymbol{G}$, must be interpolated to the embedded boundary. Similarly, the force must be transmitted back to the Cartesian grid where the fluid equations are solved. The methods of interpolating the velocity and distributing the force are typically based on discrete delta functions, as in the original immersed boundary method. Computing the forces on the embedded boundary and spreading it to the grid using discrete delta functions is in some ways easier than computing the force on the grid. However, the use of delta functions limits the order of accuracy near the boundary to first-order in space [17–19]. We note that a recent method has been proposed which spreads the force to grid, but the spreading is not based on discrete delta functions [16]. This new method may yield higher-order accuracy, because the idea behind computing the forces is more similar to methods that compute the forces on the grid, which do not limit the accuracy.

Methods that compute the forcing term directly on the Cartesian grid avoid the transfer of data back and forth between the boundary and the grid. The physical boundary condition is given at the embedded boundary, and so one needs a way to represent the boundary conditions on the Cartesian grid. This is typically accomplished using interpolation/extrapolation. For example, the boundary condition can be enforced by requiring that at points on the Cartesian grid adjacent to the embedded boundary the velocity be interpolated from the boundary points and nearby points.

Once the representation for the boundary condition on the Cartesian grid has been chosen, Eq. (4) can be used to compute the necessary forces to satisfy this boundary condition. However, the momentum equation is often discretized implicitly in time, which means that the term $\boldsymbol{G}$ in (4) depends on $\boldsymbol{U}_b^{n+1}$. There are two typical methods for handling implicit discretizations. Treating the force as an additional unknown, Eqs. (3) and (4) could be solved simultaneously for the velocity and force. Alternatively a predictor–corrector method could be used.

3. Projection methods

In the previous section we described the ideas of direct forcing methods for solving only the momentum equation (1). Projection methods are a popular class of methods for solving the incompressible Navier–Stokes equations that avoid solving the momentum equation and continuity equation simultaneously [20,21]. In a projection method, the momentum equation (1) is advanced in time without enforcing the incompressibility constraint to give an intermediate velocity. This intermediate velocity is then projected onto the space of divergence-free fields to obtain the incompressible velocity field.

The projection is accomplished by computing the Hodge decomposition of the intermediate velocity field. That is, any sufficiently regular vector field $\boldsymbol{u}^*$ can be decomposed into the sum of a divergence-free field and a gradient field:

$$\boldsymbol{u}^*=\boldsymbol{u}+\nabla\phi, \tag{5}$$

where $\nabla\cdot\boldsymbol{u}=0$. Taking the divergence of this equation gives the Poisson equation,

$$\Delta\phi=\nabla\cdot\boldsymbol{u}^*. \tag{6}$$

With suitable boundary conditions on ϕ, this equation can be solved for ϕ, and $\boldsymbol{u}$ can then be computed using (5).

The use of projection methods with direct forcing methods raises some interesting questions. The projection is performed over a computational domain that includes the physical domain and regions that are outside the physical domain. Since the projection is performed by solving a Poisson equation over this extended domain, clearly the velocity outside the physical domain influences the flow in the physical domain. In the following section we investigate under what circumstances the projection can be performed on the extended domain without affecting results on the physical domain.

4. Projections on extended domains

Suppose we are solving the incompressible Navier–Stokes equations (1) and (2) on a domain Ω_1 with the velocity given on $\partial\Omega_1$. Throughout this section, we assume that all domains considered are sufficiently regular for the purposes of our arguments. Let $\boldsymbol{u}^*$ be the intermediate velocity field that arises from advancing the momentum equation in time while ignoring

the divergence-free constraint. We assume that $\boldsymbol{u}^*$ satisfies the boundary conditions for $\boldsymbol{u}$. The intermediate velocity is decomposed into the sum of a gradient field and a divergence-free field as in (5). The decomposition is performed by solving (6) in Ω_1 with homogeneous Neumann boundary conditions, and then the divergence-free velocity is found from (5). Note that a solution to this Poisson problem exists by the divergence theorem and the assumption that $\boldsymbol{u}$ and $\boldsymbol{u}^*$ are equal on the boundary of Ω_1. We denote the projection operator which projects fields onto the space of divergence-free fields on Ω_1 by P_1, so that

$$P_1(\boldsymbol{u}^*) = \boldsymbol{u}, \tag{7}$$

where $\boldsymbol{u}^*$ and $\boldsymbol{u}$ are as in (5).

Suppose that Ω_1 is extended to the larger domain $\Omega = \Omega_1 \cup \Omega_2$. Let $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$ denote the interface between the original domain, Ω_1, and the extension, Ω_2. One such example is pictured in Fig. 1. Let $\boldsymbol{u}_e^*$ be an L^2 extension of $\boldsymbol{u}^*$ to all of Ω such that $\boldsymbol{u}_e^* = \boldsymbol{u}^*$ on Ω_1. As in (5), assuming that $\boldsymbol{u}_e^*$ is sufficiently regular ($\nabla \cdot \boldsymbol{u}_e^* \in L^2(\Omega)$ is enough) and that $\int_\Omega \nabla \cdot \boldsymbol{u}_e^* = 0$ (a necessary and sufficient condition for the solvability of the following problem (9) and (10)), this extended velocity can be decomposed

$$\boldsymbol{u}_e^* = \boldsymbol{u}_e + \nabla\phi_e, \tag{8}$$

where $\boldsymbol{u}_e$ and $\nabla\phi_e$ are L^2 vector fields with $\nabla \cdot \boldsymbol{u}_e = 0$ (in the sense of distributions), by solving

$$\Delta\phi_e = \nabla \cdot \boldsymbol{u}_e^* \quad \text{on } \Omega, \tag{9}$$

$$\frac{\partial\phi_e}{\partial n} = 0 \quad \text{on } \partial\Omega. \tag{10}$$

Analogously to P_1, we denote the operator which projects fields onto the space of divergence-free fields on the extended domain Ω by P_e, so that

$$P_e(\boldsymbol{u}_e^*) = \boldsymbol{u}_e, \tag{11}$$

where $\boldsymbol{u}_e^*$ and $\boldsymbol{u}_e$ are described above.

Ideally, the restriction of $P_e(\boldsymbol{u}_e^*)$ to Ω_1 would equal $P_1(\boldsymbol{u}^*)$. In other words, the divergence-free velocities that result from projecting on the extended domain and projecting on the original domain would be identical. Clearly, this will not always be the case, because this would mean that the projection is independent of the choice of the extension.

In this section, we show that while it is always possible to choose an extension that does not affect the projection, this is impractical. However, we demonstrate that if the intermediate velocity is in some sense close to a divergence-free field, then the projected velocities are also close. As we argue in Section 4.3, this is exactly the case in projection methods, and the error of projecting on the extended domain is the same size as the error of the projection method on the original domain. In Section 4.4 we argue that the velocity fields projected in direct forcing methods are generally only continuous across the embedded boundary. This lack of smoothness introduces additional error in the spatially discrete problem, which is analyzed in Section 4.5.

4.1. Existence of an extension

We first show that it is possible to extend $\boldsymbol{u}^*$ to all of Ω in such a way that $P_1(\boldsymbol{u}^*) = P_e(\boldsymbol{u}_e^*)$ on Ω_1. By performing the projection only on Ω_1 we obtain the divergence-free field $P_1(\boldsymbol{u}^*) = \boldsymbol{u}$ and the gradient field $\nabla\phi$, as in Eq. (5). We can extend $\boldsymbol{u}$ and

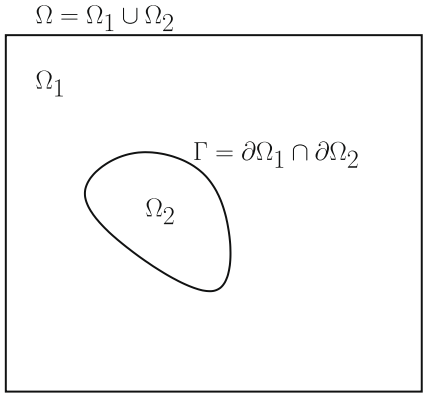

Fig. 1. The physical domain, Ω_1, is embedded in the larger domain $\Omega = \Omega_1 \cup \Omega_2$, and $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$ denotes the interface between the physical domain and extended domain.

ϕ to all of Ω so that the extension of $\boldsymbol{u}$ is divergence-free. Suppose both extensions are smooth, and denote them by $\boldsymbol{u}_e$ and ϕ_e. Define the extended velocity field by

$$\boldsymbol{u}_e^* = \boldsymbol{u}_e + \nabla\phi_e. \tag{12}$$

The projection of this velocity field gives $P_e(\boldsymbol{u}_e^*) = \boldsymbol{u}_e$, which by construction equals the velocity $P_1(\boldsymbol{u}^*) = \boldsymbol{u}$ on Ω_1. Therefore such an extension exists.

Although we have shown that the intermediate velocity can be extended in such a way as to give the "correct" projected velocity, the method used in the proof above requires that we first project on the original domain, which defeats the purpose of extending in the first place.

4.2. Comparison of projected fields

In general it is difficult to determine conditions on the extension so that $P_1(\boldsymbol{u}^*) = P_e(\boldsymbol{u}_e^*)$ without knowledge of the projection of $\boldsymbol{u}^*$ on Ω_1. However, in special cases, such as when the gradient portion vanishes on Γ, any sufficiently smooth divergence-free extension will preserve the projection. This is the case when the intermediate velocity is divergence-free to begin with. Considering divergence-free extensions of divergence-free fields does not seem to be interesting or useful, but this idea leads us to consider the case when the intermediate velocity and its extension are close to divergence-free fields.

We compare $P_1(\boldsymbol{u}^*)$ to $P_e(\boldsymbol{u}_e^*)$ in the physical domain Ω_1 in the L^2 norm. By the decompositions (5) and (8) and the fact that $\boldsymbol{u}^* = \boldsymbol{u}_e^*$ in Ω_1,

$$\|P_e(\boldsymbol{u}_e^*) - P_1(\boldsymbol{u}^*)\| = \|P_e(\boldsymbol{u}_e^*) - \boldsymbol{u}_e^* + \boldsymbol{u}^* - P_1(\boldsymbol{u}^*)\| \leqslant \|\nabla\phi_e\| + \|\nabla\phi\|. \tag{13}$$

A consequence of inequality (13) is that if $\boldsymbol{u}^*$ and $\boldsymbol{u}_e^*$ are close to being divergence-free in the sense that the difference between each of these fields and their projections is small, then so is the difference between the projections of the two fields. As a result, if the intermediate field is close to the desired divergence-free field, then the projection (on the extended domain) is just as close to the desired field. As we discuss below, this result is useful in the context of projection methods because the intermediate velocity is not arbitrary. It is close to the desired divergence-free velocity field, and this is one key reason why the projection can be performed on the extended domain and still converge to the appropriate solution.

In the case when $\boldsymbol{u}^*$ and $\boldsymbol{u}_e^*$ are both divergence-free, inequality (13) shows that $P_e(\boldsymbol{u}_e^*)$ equals $P_1(\boldsymbol{u}^*)$ in Ω_1, as claimed above. This suggests that the velocity on the extension be chosen to be as close to a divergence-free field as possible. Thus, a velocity that is divergence-free on the extension which is easy to generate is a reasonable choice.

4.3. Intermediate velocity in projection methods

In this section we show that the intermediate velocity field in a projection method is indeed close to a divergence-free field. For simplicity we consider the time-dependent Stokes equations. Suppose we want to solve the system

$$\boldsymbol{u}_t = \Delta\boldsymbol{u} - \nabla p, \tag{14}$$

$$\nabla\cdot\boldsymbol{u} = 0 \tag{15}$$

in Ω_1 with Dirichlet boundary conditions, and we are given the initial velocity $u(x,t_0)$ and pressure $p_0(x) = p(x,t_0)$. We hold the pressure constant in time, and advance the momentum equation in time to $t = t_0 + \Delta t$ by solving

$$\boldsymbol{u}_t^* = \Delta\boldsymbol{u}^* - \nabla p_0(x). \tag{16}$$

The approximate solution to (14) and (15) is then generated by performing the projection

$$\boldsymbol{u}(x,t_0+\Delta t) \approx P_1\boldsymbol{u}^*(x,t_0+\Delta t). \tag{17}$$

To show that $\boldsymbol{u}^*(x,t_0+\Delta t)$ is close to a divergence-free field, we expand $\boldsymbol{u}^*(x,t_0+\Delta t)$ and $\boldsymbol{u}(x,t_0+\Delta t)$ as $\Delta t\to 0$ and show that $\boldsymbol{u}(x,t_0+\Delta t) = \boldsymbol{u}^*(x,t_0+\Delta t) + \mathcal{O}(\Delta t^2)$. First, the expansion of $\boldsymbol{u}(x,t_0+\Delta t)$ is

$$\boldsymbol{u}(x,t_0+\Delta t) = \boldsymbol{u}(x,t_0) + \Delta t\boldsymbol{u}_t(x,t_0) + \frac{\Delta t^2}{2}\boldsymbol{u}_{tt}(x,t_0) + \mathcal{O}(\Delta t^3). \tag{18}$$

Using Eq. (14) to eliminate one time derivative gives

$$\boldsymbol{u}(x,t_0+\Delta t) = \boldsymbol{u}(x,t_0) + \Delta t(\Delta\boldsymbol{u}(x,t_0) - \nabla p(x,t_0)) + \frac{\Delta t^2}{2}(\Delta\boldsymbol{u}_t(x,t_0) - \nabla p_t(x,t_0)) + \mathcal{O}(\Delta t^3). \tag{19}$$

Similarly,

$$\boldsymbol{u}^*(x,t_0+\Delta t) = \boldsymbol{u}^*(x,t_0) + \Delta t(\Delta\boldsymbol{u}^*(x,t_0) - \nabla p_0(x)) + \frac{\Delta t^2}{2}\frac{\partial}{\partial t}(\Delta\boldsymbol{u}^*(x,t) - \nabla p_0(x))|_{t=t_0} + \mathcal{O}(\Delta t^3). \tag{20}$$

Because $\boldsymbol{u}^*(x,t_0)=\boldsymbol{u}(x,t_0)$ all occurrences of $\boldsymbol{u}^*$ can be eliminated on the right side of this equation to get

$$\boldsymbol{u}^*(x,t_0+\Delta t)=\boldsymbol{u}(x,t_0)+\Delta t(\Delta\boldsymbol{u}(x,t_0)-\nabla p_0(x))+\frac{\Delta t^2}{2}(\Delta\boldsymbol{u}_t(x,t_0))+\mathcal{O}(\Delta t^3). \tag{21}$$

Comparing the expansions (19) and (21) and using that $p_0(x)=p(x,t_0)$, we get that

$$\boldsymbol{u}^*(x,t_0+\Delta t)=\boldsymbol{u}(x,t_0+\Delta t)+\frac{\Delta t^2}{2}\nabla p_t(x,t_0)+\mathcal{O}(\Delta t^3). \tag{22}$$

Therefore, the intermediate velocity that is projected is within $\mathcal{O}(\Delta t^2)$ of a divergence-free field. Furthermore, $P_1\boldsymbol{u}^*(x,t_0+\Delta t)=\boldsymbol{u}(x,t_0+\Delta t)+\mathcal{O}(\Delta t^3)$, since the second-order term is a gradient field.

4.4. Continuity of extensions from direct forcing

The intermediate velocity that is generated by a forcing method will be continuous across the embedded boundaries because the forcing enforces the Dirichlet boundary condition. However, the solution will not generally be smooth across the embedded boundaries. We argue via a simple example why the extended intermediate velocity is only continuous.

Consider the solution to the forced diffusion equation in one dimension on the semi-infinite domain $x>0$ with homogeneous Dirichlet boundary condition $u(0,t)=0$. Discretize the entire real line using the points $x_j=(j+1/2)h$, and solve the discrete equations for all j. The boundary condition is enforced by applying a force at the first grid point of the extension, x_{-1}. The force applied is chosen to enforce the algebraic condition $u^n_{-1}=-u^n_0$ for all n, where $u^n_j=u(x_j,n\Delta t)$.

The discrete solution at the grid points in the extension $j<-1$ are coupled to the physical domain only through u^n_{-1}. We may view the solution in the extension as the solution to the unforced heat equation on the semi-infinite domain $x<-h/2$ with the time-dependent Dirichlet boundary condition $u(-h/2,n\Delta t)=u^n_{-1}$. Because $u^n_{-1}=-u^n_0=\mathcal{O}(h)$, the solution on the extension is $\mathcal{O}(h)$. Therefore as $h\to 0$, the solution in the extension converges to zero, while the solution in the physical domain converges to the solution of the PDE, which is zero at the boundary, but need not have zero derivative. Therefore the discrete solution on the extended domain converges to a function that is continuous but not necessarily differentiable across the interface between the physical domain and the extension. This is illustrated in Fig. 2.

When the intermediate velocity is only continuous across the embedded boundaries, it is not obvious that projection and decomposition (8) can be performed, since this involves derivatives of functions that are not differentiable. As mentioned previously, we only need that $\nabla\cdot\boldsymbol{u}^*_e\in L^2(\Omega)$ in order to perform the projection. In general, the intermediate velocity is differentiable on the closure of Ω_1, differentiable on the closure of Ω_2, and continuous in Ω. In particular, $\boldsymbol{u}^*_e$ is continuous across Γ. In other words, there could be a jump in the one-sided derivatives of the components of $\boldsymbol{u}^*_e$ on Γ. As a result, the components of $\boldsymbol{u}^*_e$ are weakly differentiable in Ω, with derivatives that are bounded, and thus $\nabla\cdot\boldsymbol{u}^*_e\in L^2(\Omega)$. Therefore, the projection can be applied to the intermediate velocity in direct forcing methods despite the lack of smoothness.

4.5. Spatially discrete problem

In this section we consider the spatially discrete problem, and we show that when the extended velocity is not smooth across Γ, the error in the discrete problem is larger near this interface than in the rest of the domain. Although the scheme converges as the mesh spacing goes to zero, the order of accuracy is reduced by the lack of smoothness.

Suppose that the extended domain, Ω, is discretized into regular cells of width h with centers $(x_i,y_j)=((i+1/2)h,(j+1/2)h)$. Let $\boldsymbol{u}^h=(u^h,v^h)$ represent a discrete velocity field and ϕ^h represent a discrete scalar field. We consider the standard staggered discretization, so that the horizontal component of the velocity, u^h, is stored at the vertical cell edges, the vertical component of the velocity, v^h, is stored at the horizontal cell edges, and scalars (e.g. ϕ^h) are stored at the cell center. See Fig. 3.

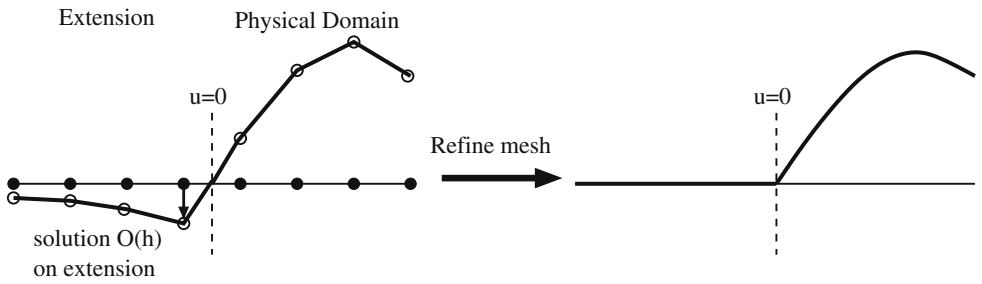

Fig. 2. Illustration of solution behavior from a forcing method for the one-dimensional forced diffusion equation with homogeneous boundary condition. The discrete solution typically converges to a function that is continuous but not differentiable across the embedded boundary.

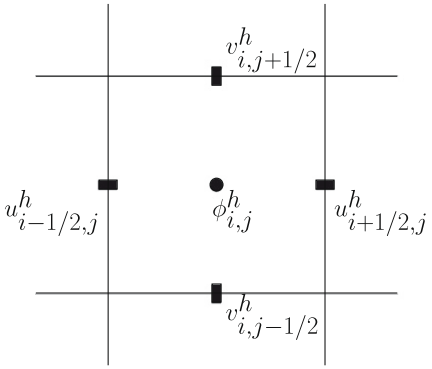

Fig. 3. Example of a staggered grid discretization. The velocity field $\boldsymbol{u}^h = (u^h, v^h)$ is stored at the cell edges as shown, and scalar fields are stored at the cell centers.

Let D denote the discrete divergence operator, defined as

$$(D\boldsymbol{u}^h)_{ij} = \frac{u^h_{i+1/2,j} - u^h_{i-1/2,j}}{h} + \frac{v^h_{ij+1/2} - v^h_{ij-1/2}}{h}. \tag{23}$$

Let G be the discrete gradient operator, $G\phi = (G_1\phi^h, G_2\phi^h)$, where the first component is

$$\left(G_1\phi^h\right)_{i-1/2j} = \frac{\phi^h_{ij} - \phi^h_{i-1,j}}{h}, \tag{24}$$

and similarly the second component is

$$\left(G_2\phi^h\right)_{ij-1/2} = \frac{\phi^h_{ij} - \phi^h_{ij-1}}{h}. \tag{25}$$

As with the continuous problem, a discrete vector field $\boldsymbol{u}^{h*}$ can be decomposed into the sum of a discretely divergence-free field and a discrete gradient by solving the discrete Poisson equation

$$L\phi^h = DG\phi^h = D\boldsymbol{u}^{h*}, \tag{26}$$

where L is the standard five-point, second-order accurate, discrete Laplacian. The discretely divergence-free field is then

$$\boldsymbol{u}^h = \boldsymbol{u}^{h*} - G\phi^h. \tag{27}$$

In the absence of an internal interface, the discrete projection is a second-order accurate approximation to the continuous projection. We next address how the internal interface affects the accuracy of the discrete approximation. We define *regular* grid cells as those whose centers and four edge velocities lie entirely in Ω_1 or entirely in Ω_2, i.e. the cell center and the four centers of the edges are all on the same side of the interface Γ. Other grid points are called *irregular* points. See Fig. 4.

We assume that the extended velocity field, $\boldsymbol{u}_e$, is at least C^3 in Ω_1 and in Ω_2. For the regular grid points

$$D\boldsymbol{u}^h_e = \nabla\cdot\boldsymbol{u}_e + \mathcal{O}(h^2), \tag{28}$$

where the truncation error term is proportional to the third derivatives of $\boldsymbol{u}_e$. At the irregular grid points, the truncation error of the derivatives depends on the smoothness of the velocity across Γ.

To illustrate the effect of smoothness on the spatially discrete problem, consider a scalar function $g(x)$ defined by

$$g(x) = \begin{cases} g^-(x) & \text{if } x < x_0, \\ g^+(x) & \text{if } x > x_0, \end{cases} \tag{29}$$

where we assume that both g^- and g^+ are smooth. The centered difference of g at x_0 is

$$\begin{aligned} \frac{g(x_0+h/2) - g(x_0-h/2)}{h} = {} & \frac{1}{h}\left(g^+(x_0) - g^-(x_0)\right) + \frac{1}{2}\left(g^+_x(x_0) + g^-_x(x_0)\right) + \frac{h}{8}\left(g^+_{xx}(x_0) - g^-_{xx}(x_0)\right) \\ & + \frac{h^2}{48}\left(g^+_{xxx}(x_0) + g^-_{xxx}(x_0)\right) + \mathcal{O}(h^3). \end{aligned} \tag{30}$$

Thus if g is at least C^2 at x_0, then the centered difference is second-order accurate at x_0. If g is only C^1, then this centered difference is first-order accurate, and if g is only C^0, then the error in the approximation is order one. Although we considered the centered difference at x_0 for simplicity, the orders of accuracy hold for the centered difference about any point in the interval $(x_0 - h/2, x_0 + h/2)$.

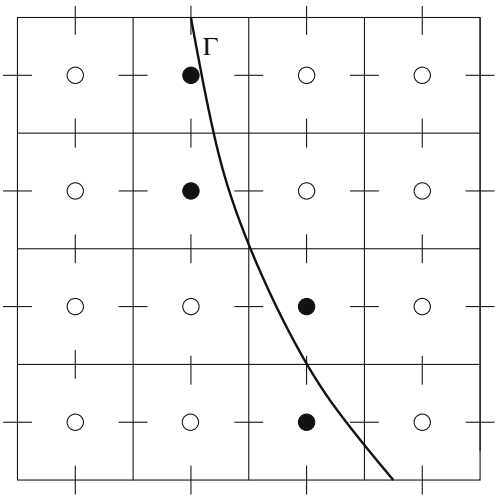

Fig. 4. The open circles denote the centers of regular grid cells; the cell center and all four centers of the edges are on the same side of the interface. The solid circles represent the centers of the irregular cells.

At the irregular grid cells, the discrete divergence involves velocities from both sides of the interface. If the intermediate velocity is C^2 across the interface Γ, then the truncation error at the irregular points is $\mathcal{O}(h^2)$, which is the same as at the regular points, and the discrete projection is second-order accurate. If the intermediate velocity is only C^1 across the interface, then the truncation error in the discrete divergence is $\mathcal{O}(h)$ at the irregular points. The error term is proportional to the jump in the second derivative. It has long been observed in practice that if the truncation error is $\mathcal{O}(h)$ on a lower dimensional set of points (codimension one in the limit) and $\mathcal{O}(h^2)$ everywhere else, then the error in the solution of the discrete Poisson equation is $\mathcal{O}(h^2)$ in the max-norm. However, it was not until recently that this result was rigorously proved by Beale and Layton [22].

Next consider the most interesting case in which the velocity is only C^0 across the interface, which is the most relevant case for understanding direct forcing methods. In this case, the truncation error of the discrete divergence is $\mathcal{O}(1)$ at the irregular points, i.e. this is an inconsistent discretization. Although Theorem 2.1 from [22] does not apply in this case, the lemmas used to prove the theorem can be adapted to give bounds on the error.

We summarize two lemmas from [22] that we use to estimate the error in the discrete projection. Let f^h_{irr} be a discrete scalar function that is nonzero only on irregular cells. By Lemma 2.2 of [22], there exists a vector grid function $\boldsymbol{F}^h$ such that

$$f^h_{\text{irr}} = D\boldsymbol{F}^h, \tag{31}$$

and

$$\|F^h_k\|_\infty \leqslant Ch\|f^h_{\text{irr}}\|_\infty, \tag{32}$$

where F^h_k is the kth component of $\boldsymbol{F}^h$, for some constant C independent of the grid spacing h. Lemma 2.3 of [22] gives a bound on the solution to the discrete Poisson equation in terms of the size of the forcing function at the regular and irregular grid points. Specifically, suppose that ϕ^h is the solution to

$$L\phi^h = f^h_{\text{reg}} + f^h_{\text{irr}} = f^h_{\text{reg}} + D\boldsymbol{F}^h, \tag{33}$$

with homogeneous Dirichlet boundary conditions. Then

$$\|\phi^h\|_\infty \leqslant C_0\left(\|f^h_{\text{reg}}\|_2 + \sum_k \|F^h_k\|_\infty\right), \tag{34}$$

and

$$\|G\phi^h\|_\infty \leqslant C_1 \log(h^{-1})\left(\|f^h_{\text{reg}}\|_2 + \sum_k \|F^h_k\|_\infty\right), \tag{35}$$

where the constants C_0 and C_1 are independent of the grid spacing. Although this second lemma applies to the Dirichlet problem, it can be adapted to the Neumann problem as discussed in [22].

To apply these lemmas to the discrete projection, consider the case when $\boldsymbol{u}$ is a divergence-free field on Ω_1, and $\boldsymbol{u}_e$ is an extension of $\boldsymbol{u}$ to all of Ω that is divergence-free and continuous (but not differentiable) across Γ. (Note that the error bounds derived below also apply to the case when the velocity projected is not divergence-free.) Let $\boldsymbol{u}^h_e$ be the representation of $\boldsymbol{u}_e$ on the grid with spacing h. Then

$$D\boldsymbol{u}_e^h = \begin{cases} \mathcal{O}(h^2) & \text{at regular points,} \\ \mathcal{O}(1) & \text{at irregular points.} \end{cases} \tag{36}$$

To perform the discrete projection, we solve

$$L\phi^h = D\boldsymbol{u}_e^h. \tag{37}$$

Comparing this equation with (33), we use (31), (32) and (35) to arrive at the bound on the gradient

$$\|G\phi^h\|_\infty \leqslant C_1 \log(h^{-1})(\mathcal{O}(h^2) + \mathcal{O}(h)) = \mathcal{O}(h\log(h)). \tag{38}$$

The projected discrete velocity is then

$$P\boldsymbol{u}_e^h = \boldsymbol{u}_e^h - G\phi^h = \boldsymbol{u}_e^h + \mathcal{O}(h\log(h)). \tag{39}$$

Therefore as h goes to zero the discrete projection operator converges to the continuous projection operator in the max-norm. Although this bound on the error is less than first-order, as we show in our numerical tests, we observe a first-order error in practice in the max-norm, which is consistent with the observations from [22]. In practice it is difficult to distinguish between $\mathcal{O}(h\log(h))$ and $\mathcal{O}(h)$.

The bound above applies to the max-norm of the error. Because the larger truncation errors are concentrated near the interface Γ one would hope that the errors are also concentrated near the interface. Our numerical tests confirm this, and we obtain higher rates of convergence in the 1-norm and 2-norm as discussed in Section 5.1.1.

5. Computational tests

5.1. Projection

We begin by demonstrating how the smoothness of the extension affects the accuracy of the discrete projection. Since we are interested in the projection of nearly divergence-free fields, we apply the discrete projection to divergence-free fields. The first test problem is only continuous across an embedded boundary. We verify our bound on the max-norm of the error and demonstrate that the errors converge at a faster rate in the 1-norm and 2-norm, indicating that the additional error introduced by the lack of smoothness is indeed concentrated near the interface.

The domain used in these problems is a circle of radius R. This circular domain is embedded into a larger square domain. In the notation of the previous section,

$$\Omega_1 = \{(x,y) : x^2 + y^2 < R^2\}, \tag{40}$$
$$\Omega = [-L/2, L/2] \times [-L/2, L/2], \tag{41}$$
$$\Omega_2 = \Omega \setminus \Omega_1. \tag{42}$$

In the computational tests, $L = 1$ and $R = 0.4$. We use a staggered grid and a MAC projection as described in Section 4.5.

5.1.1. Projection of a C^0 divergence-free velocity

For a C^0 divergence-free test problem, we begin with a rotational flow defined by the angular velocity

$$u_\theta = Ar^2(R - r)H(R - r), \tag{43}$$

where

$$A = \frac{27}{4R^3}, \tag{44}$$

so that the maximum velocity is 1, and H is the Heaviside function. In the basis of rectangular coordinates, the velocity field is

$$\boldsymbol{u} = Ar(R - r)H(R - r)(y\boldsymbol{i} - x\boldsymbol{j}), \tag{45}$$

where $\boldsymbol{i}$ and $\boldsymbol{j}$ are the unit direction vectors. This velocity field is divergence-free almost everywhere, but it is not differentiable across the interface $r = R$. The error introduced by the discrete projection is

$$\boldsymbol{e}^h = \boldsymbol{u}^h - P\boldsymbol{u}^h, \tag{46}$$

which is equal to the discrete gradient field $G\phi^h$.

Table 1 shows the 1-norm, 2-norm, and max-norm of the error in the first component of the velocity from the discrete projection as the grid spacing decreases. By symmetry, the error is the same in both components of the velocity. The max-norm of the error shows first-order convergence (or slightly less than first-order), as predicted by our analysis. The convergence in the 1-norm and 2-norm is more rapid. The error shows second-order convergence in the 1-norm, and order 1.5 in the 2-norm as demonstrated by the log plots of the errors in Fig. 5. These orders of convergence in the integral norms indicate that the errors are first-order near the embedded boundary and second-order away from the boundary. In Fig. 6 we plot

Table 1
Errors from a refinement study of the discrete projection of the velocity field (45) which is only C^0 across the internal boundary for different grid spacings h. The error in the first component is shown. By symmetry, the norms of the errors are identical in the two components. The convergence is second-order in the 1-norm, order 1.5 in the 2-norm, and first-order in the max-norm.

h	$\|e\|_1$	$\|e\|_2$	$\|e\|_\infty$
2^{-5}	$8.62 \cdot 10^{-4}$	$1.86 \cdot 10^{-3}$	$1.67 \cdot 10^{-2}$
2^{-6}	$3.28 \cdot 10^{-4}$	$8.42 \cdot 10^{-4}$	$8.88 \cdot 10^{-3}$
2^{-7}	$6.83 \cdot 10^{-5}$	$2.81 \cdot 10^{-4}$	$5.57 \cdot 10^{-3}$
2^{-8}	$1.94 \cdot 10^{-5}$	$1.07 \cdot 10^{-4}$	$3.08 \cdot 10^{-3}$
2^{-9}	$5.02 \cdot 10^{-6}$	$3.73 \cdot 10^{-5}$	$1.52 \cdot 10^{-3}$

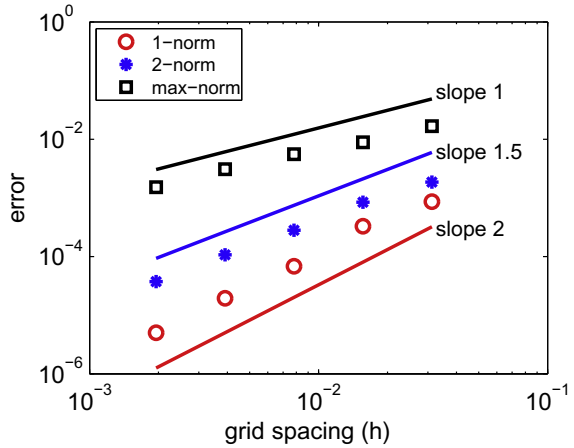

Fig. 5. Plots of the errors from the refinement study data in Table 1 for the C^0 velocity field. For reference, lines of slope 1, 1.5, and 2 are displayed, which indicate first-order convergence in the max-norm, order 1.5 convergence in the 2-norm, and second-order in the 1-norm, respectively.

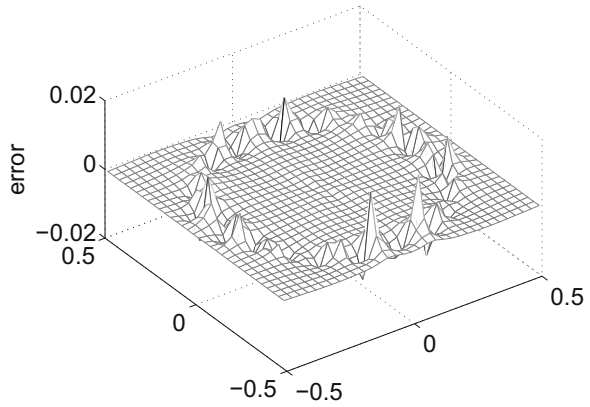

Fig. 6. Plot of the error from the discrete projection of the velocity field (45) which is only C^0 across the boundary $R = 0.4$. The error in the first component is plotted. The errors are much larger near the boundary than in the remainder of the domain.

the errors on the 32×32 grid, and we see that indeed the errors are much larger near the embedded boundary than in the rest of the domain.

5.1.2. Projection of a C^1 divergence-free velocity

For comparison, we consider a velocity field which is divergence-free and C^1 across the internal boundary. We again use a rotational flow with angular velocity

$$u_\theta = Ar^2(R - r)^2 H(R - r), \tag{47}$$

and the normalization is

Table 2
Errors from a refinement study of the discrete projection of the velocity field (47) which is C^1 across the internal boundary for different grid spacings h. The error in the first component is shown. By symmetry, the norms of the errors are identical in the two components. The convergence is second-order in all norms.

h	$\|e\|_1$	$\|e\|_2$	$\|e\|_\infty$
2^{-5}	$3.14 \cdot 10^{-4}$	$4.23 \cdot 10^{-4}$	$1.41 \cdot 10^{-3}$
2^{-6}	$7.31 \cdot 10^{-5}$	$9.57 \cdot 10^{-5}$	$3.46 \cdot 10^{-4}$
2^{-7}	$1.81 \cdot 10^{-5}$	$2.36 \cdot 10^{-5}$	$9.18 \cdot 10^{-5}$
2^{-8}	$4.50 \cdot 10^{-6}$	$5.83 \cdot 10^{-6}$	$2.29 \cdot 10^{-5}$
2^{-9}	$1.13 \cdot 10^{-6}$	$1.46 \cdot 10^{-6}$	$5.83 \cdot 10^{-6}$

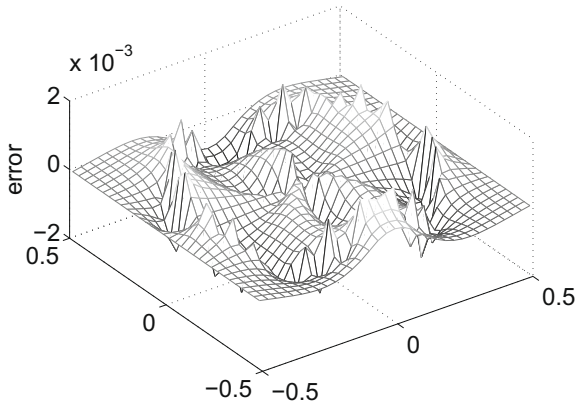

Fig. 7. Plot of the error of the discrete projection of the velocity field (47) which is C^1 across the boundary $R = 0.4$. The error in the first component is plotted. The errors near the boundary are comparable in size to those throughout the domain.

$$A = \frac{16}{R^4}, \tag{48}$$

so that, as before, the maximum velocity is 1, and H is the Heaviside function. Again the error introduced by the discrete projection is (46).

Table 2 shows the 1-norm, 2-norm, and max-norm of the error in the projection as the grid spacing decreases. As expected, the convergence in all norms is second-order, even though the local truncation error is first-order near the embedded boundary. In Fig. 7 we plot the errors for the 32×32 grid, and we see that the error near the boundary is comparable in size to the error in the rest of the domain.

5.2. Navier–Stokes tests

In the previous section we verified the results of our analysis on the accuracy of the projection alone. In this section we consider the accuracy of projection methods for solving the Navier–Stokes equations. We solve the incompressible Navier–Stokes equations in two spatial dimensions for the flow through a channel with a circular obstacle inside the channel. The computational domain is $[0,3] \times [0,1]$ (dimensionless) with periodic boundary conditions in the horizontal direction and no-slip boundary conditions on the top and bottom of the channel. There is a stationary circular obstacle with radius 0.2 with center $(1,0.4)$ on which the flow satisfies the no-slip condition. The Reynolds number is set to 50. The fluid is initially at rest; we drive the flow with a constant background force in the x-direction, $\boldsymbol{f} = (8.0, 0)^{\mathrm{T}}$, and we solve for the velocity field at time $t = 0.1$. The maximum magnitude of the velocity field is approximately 1.51. The maximum horizontal velocity is also approximately 1.51, and the maximum vertical velocity is about 0.55. All reported errors are absolute errors. The velocity field is displayed in Fig. 8.

5.2.1. Zero velocity on the extension

The computational domain is discretized using a staggered grid. The momentum equation is solved only on the physical domain (outside the obstacle). Before projecting, the velocity inside the obstacle is set to zero, as motivated by the discussion in Section 4.2. To advance the momentum equation for the intermediate velocity, the advection terms are discretized explicitly using a second-order Adams–Bashforth method, and the pressure gradient is lagged from the previous time step. A Crank–Nicolson discretization is used for the viscous terms. At grid points adjacent to the obstacle, the stencil of the discrete Laplacian is modified to include the point(s) on the boundary of the obstacle. For example, in one dimension, suppose that

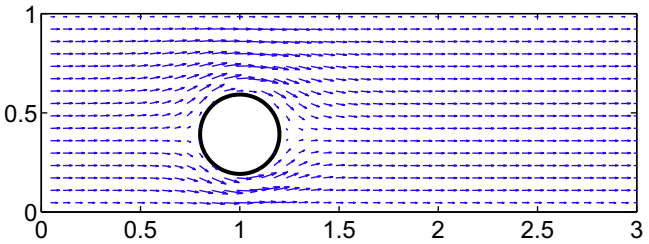

Fig. 8. The velocity field through a two dimensional channel with a circular obstruction which is used to test the accuracy of the Navier–Stokes solver.

$x_j = jh$ is in the physical domain, $x_{j+1} = (j+1)h$ is outside the physical domain, and the obstacle boundary is located at $x_b = (j+\alpha)h$ for some $0 < \alpha < 1$. The discrete second derivative at point j is

$$(u_{xx})_j = \frac{2\alpha u_{j-1} - 2(1+\alpha)u_j + 2u_b}{\alpha(1+\alpha)h^2} + \mathcal{O}(h). \tag{49}$$

Although the local truncation error is first-order in space, the error that results in the solution is second-order.

The solver used for the momentum equation alone (no projection) is second-order accurate in space and time. This has been verified with computational tests (results not shown). The projection is performed on the extended domain, $[0,3] \times [0,1]$ (i.e. inside and outside of the obstacle). The intermediate velocity on the extension is zero (inside the obstacle). Thus the velocity is divergence-free on the extension, and because the velocity satisfies the no-slip condition on the obstacle, the velocity is continuous across the obstacle boundary. For more details on the discretization, see Appendix A.1.

We perform a refinement study to estimate the rate of convergence. On the coarsest mesh, the space step is 2^{-5} and time step is 10^{-2}. Time and space are refined simultaneously by factors of two. Since an analytic solution for the velocity is not available, we estimate the error on a given mesh by subtracting the numerical solution on the next finest mesh. The velocity on the finer mesh must be interpolated to the coarser mesh, and this interpolation is appropriately modified near the obstacle boundary.

The results of the refinement study are displayed in Table 3. We report the norms of the estimated errors for the 1-norm, 2-norm, and max-norm for both the horizontal and vertical components of the velocity field, u and v, respectively. The order of convergence is computed from fitting a line through the log of the norm of the error versus the log of the grid spacing.

Our analysis predicts that the rate of convergence should be first-order in the max-norm, and our previous numerical tests of the projection suggest that the convergence will be second-order in the 1-norm and order 1.5 in the 2-norm. These are roughly the orders that we observe for the vertical velocity, v. For the horizontal velocity, the rate of convergence is slightly better than predicted in the 2-norm and in the max-norm. The convergence appears to be second-order in the 2-norm.

Recall that the source of the lower-order errors is related to the jumps in the derivatives of the velocity across the embedded boundary. In the tests involving only the projection presented in Section 5.1.1, we constructed the velocity so that the jump in the derivative was uniform around the embedded boundary. In this test involving the Navier–Stokes equations, the jump in the velocity is not uniform around the embedded boundary, and as a result we observe higher rates of convergence.

5.2.2. Forcing method

For comparison, we repeat this test using a forcing method to solve the momentum equation. We use the method of Kim et al. [5]. The forces are chosen so that at the first grid point on the extended domain (inside the obstacle in this case) the intermediate velocity is a linear extrapolation of the velocity from the physical domain.

This method is in the form of a predictor–corrector. A predicted velocity is used to estimate the force necessary to ensure that the velocity at the first cell in the extension is the extrapolation from the physical domain. This force is then applied at

Table 3
Refinement study for Navier–Stokes equation in which the momentum equation is solved only on the physical domain, and then the solution is extended continuously with a divergence-free field (identically zero).

h	u			v		
	$\|e\|_1$	$\|e\|_2$	$\|e\|_\infty$	$\|e\|_1$	$\|e\|_2$	$\|e\|_\infty$
2^{-5}	$1.46 \cdot 10^{-2}$	$1.90 \cdot 10^{-2}$	$7.93 \cdot 10^{-2}$	$2.96 \cdot 10^{-3}$	$7.85 \cdot 10^{-3}$	$1.28 \cdot 10^{-1}$
2^{-6}	$3.89 \cdot 10^{-3}$	$5.02 \cdot 10^{-3}$	$3.23 \cdot 10^{-2}$	$7.52 \cdot 10^{-4}$	$1.95 \cdot 10^{-3}$	$4.31 \cdot 10^{-2}$
2^{-7}	$9.43 \cdot 10^{-4}$	$1.30 \cdot 10^{-3}$	$1.27 \cdot 10^{-2}$	$2.33 \cdot 10^{-4}$	$7.79 \cdot 10^{-4}$	$2.39 \cdot 10^{-2}$
2^{-8}	$2.32 \cdot 10^{-4}$	$3.46 \cdot 10^{-4}$	$5.41 \cdot 10^{-3}$	$6.91 \cdot 10^{-5}$	$2.67 \cdot 10^{-4}$	$1.12 \cdot 10^{-2}$
Order	2.00	1.93	1.30	1.80	1.60	1.14

these points, and the momentum equation is advanced. We modify this procedure in our tests to iterate until the velocity is within some tolerance of the extrapolated value. Further details on the discretization are in Appendix A.2. Although setting the grid values by this extrapolation gives an even larger truncation error (order one) for the discrete Laplacian, the solution to the momentum equation is still second-order accurate. We have verified that in the absence of the projection, the solution to the momentum equation is second-order accurate in space and time (results not shown).

The results of the refinement study are presented in Table 4. In all norms and for both components of the velocity, the convergence appears to be between first- and second-order. Comparing with the results of the previous test, the estimated errors in this test are generally larger, except in the max-norm of the vertical velocity on the finest mesh.

There are two significant differences between this forcing method, and the method presented previously. First, the way in which the boundary conditions on the embedded boundary are enforced is different, and second, the velocity on the extension is nonzero in the forcing method. Since this method would give second-order accurate results without the projection, it appears that it is the nonzero velocity on the extension which is giving larger than expected errors.

As argued in Section 4.4, we expect the velocity on the extension to go to zero as the mesh is refined. In Table 5 we show the 1-norm and max-norm of the velocity on the extension, and indeed it is going to zero as the mesh is refined. The large errors are arising from the lack of smoothness in the velocity across the extension. It is the size of the jump in the velocity

Table 4
Refinement study for Navier–Stokes equations using a forcing method.

h	u			v		
	$\|e\|_1$	$\|e\|_2$	$\|e\|_\infty$	$\|e\|_1$	$\|e\|_2$	$\|e\|_\infty$
2^{-5}	$2.89 \cdot 10^{-2}$	$2.61 \cdot 10^{-2}$	$1.75 \cdot 10^{-1}$	$6.97 \cdot 10^{-3}$	$1.10 \cdot 10^{-2}$	$1.31 \cdot 10^{-1}$
2^{-6}	$8.25 \cdot 10^{-3}$	$1.11 \cdot 10^{-2}$	$1.21 \cdot 10^{-1}$	$4.01 \cdot 10^{-3}$	$6.76 \cdot 10^{-3}$	$6.61 \cdot 10^{-2}$
2^{-7}	$2.40 \cdot 10^{-3}$	$2.50 \cdot 10^{-3}$	$3.58 \cdot 10^{-2}$	$1.04 \cdot 10^{-3}$	$1.70 \cdot 10^{-3}$	$2.22 \cdot 10^{-2}$
2^{-8}	$2.38 \cdot 10^{-3}$	$1.77 \cdot 10^{-3}$	$1.14 \cdot 10^{-2}$	$8.36 \cdot 10^{-4}$	$1.08 \cdot 10^{-3}$	$6.01 \cdot 10^{-3}$
Order	1.26	1.38	1.36	1.11	1.21	1.49

Table 5
Norms of the velocity on the extension as the grid is refined.

h	$\|u_{ext}\|_1$	$\|u_{ext}\|_\infty$	$\|v_{ext}\|_1$	$\|v_{ext}\|_\infty$
2^{-5}	$2.12 \cdot 10^{-1}$	$8.23 \cdot 10^{-1}$	$1.05 \cdot 10^{-1}$	$5.74 \cdot 10^{-1}$
2^{-6}	$9.84 \cdot 10^{-2}$	$6.26 \cdot 10^{-1}$	$5.75 \cdot 10^{-2}$	$3.04 \cdot 10^{-1}$
2^{-7}	$5.13 \cdot 10^{-2}$	$3.69 \cdot 10^{-1}$	$2.82 \cdot 10^{-2}$	$1.45 \cdot 10^{-1}$
2^{-8}	$2.69 \cdot 10^{-2}$	$1.97 \cdot 10^{-1}$	$1.35 \cdot 10^{-2}$	$8.61 \cdot 10^{-2}$

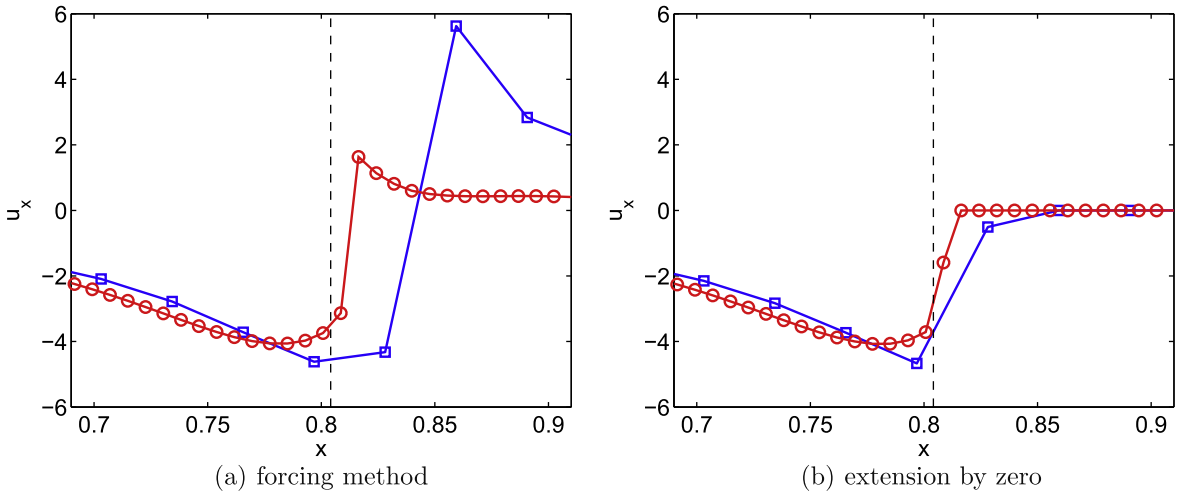

Fig. 9. Plot of u_x computed using a finite difference along the line $y = 0.328125$ computed from the (a) forcing method and (b) method in which the intermediate velocity is extended by zero. The dashed line denotes the physical location of the embedded boundary. The results are shown for two different mesh spacings. Squares are used for $h = 2^{-5}$, and circles are used for $h = 2^{-7}$.

gradient which controls this error. In Fig. 9 we show u_x computed with a finite difference along a line of fixed height for both the forcing method and the method in which the velocity on the extension is set to zero. In the forcing method the velocity is smooth across the interface, but there is a large jump in the derivative at the second point inside the extension. The size of the discontinuity in the derivative is much larger in the forcing method than in the other method. This explains the much larger errors observed in the forcing method.

These plots also suggest why the error in the max-norm decreased at a faster rate than our analysis predicts. It is not because the method is more accurate, but that the size of the jump in the derivative is decreasing as the grid is refined. Thus the size of the first-order error introduced from the lack of smoothness is decreasing, giving the illusion of faster convergence. Also, the jump in the derivative occurs one grid cell inside the extension, rather than at the boundary between the physical domain and the extension.

6. Discussion

In this paper we presented analysis and numerical tests aimed at understanding how well, if at all, projection methods work when the projection is performed over an extended domain that includes regions outside the physical domain. This work was motivated by direct forcing methods, in which the equations of motion are solved on these extended domains. In general, one expects the flow on the extended domain to affect the flow on the physical domain through the projection. The intermediate velocity field that arises in a projection method is not arbitrary; it is close to a divergence-free field. This is the key reason why projection methods on extended domains work. For the spatially continuous problem, we showed that the error in performing the projection on the extended domain is of the same size as the error introduced from the projection method on the original domain.

Generally, the intermediate velocity that arises from direct forcing methods is not smooth across the embedded boundaries. When performing the projection discretely, this lack of smoothness introduces a large error near the embedded boundaries. We analyzed the accuracy of the discrete projection by extending the results of Beale and Layton [22]. We showed that the projection is first-order accurate in the max-norm when the intermediate velocity is only continuous across the embedded boundaries. Numerical tests showed that this error is highly localized near the embedded boundaries, and as a result, we observed second-order convergence in the 1-norm, although when the projection was combined with a direct forcing method to solve the incompressible Navier–Stokes equations, the convergence rate depended on the method used to solve the momentum equation.

Several papers have commented on the treatment of the velocity on the extension. Fadlun et al. [4] and Zhang and Zheng [8] compared letting the velocity evolve on the extension and setting it to the value of the solid body after each time step. Both papers concluded that there is little difference in the results, although they did not quantify the difference. We argue that as discrete time and space are refined, the velocity on the extension will converge to the velocity of the solid body, and so it is not surprising that these approaches gave similar results.

From our analysis, we conclude that the dominant error in projecting on the extended domain is caused by the lack of smoothness of the intermediate velocity across the internal boundaries, not the flow on the extension. This explains why others have observed that the treatment of the extension does not strongly influence the flow on the physical domain. In our numerical tests, we observed smaller errors when the velocity was reset on the extension. This is partly because the jumps in the derivatives of the velocity were smaller when the velocity was reset.

Fadlun et al. [4] speculated that the projection worked in their forcing method because the gradient portion of the intermediate velocity was effectively zero on the embedded boundaries, although they did not provide any numerical evidence to support this conjecture. This argument can be formalized to show that if the gradient portion of the intermediate velocity is indeed zero on the embedded boundary, then the projection is independent of the extension provided the extension is divergence-free. A zero gradient may result in some specific problems, but it is not generally true that this gradient field would or should be zero [15]. In our numerical tests we observed a nonzero pressure gradient near the embedded boundaries. This paper provides a more complete picture of why projection methods can be used with direct forcing methods.

According to our analysis, the solution will generally be first-order accurate in the max-norm, regardless of the accuracy of the method used to advance the momentum equation. In practice, one may observe higher rates of convergence. The first-order error is proportional to the size of the jump in the velocity gradient across the embedded boundaries. If the jumps in the derivatives are small, the error from the lack of smoothness may be smaller than other errors in the problem.

Some papers on forcing methods have reported second-order convergence in max-norm [5,6,16], which appears contrary to the analysis presented in this paper. Our analysis predicts that the limitation on accuracy arises from the lack of smoothness of the intermediate velocity across the embedded boundary. In the refinement studies of [5,6], the velocity field tested was chosen to be smooth across the embedded boundary, and so their results are consistent with the predictions of this paper. In more general tests, one will not have a smooth extension across the embedded boundary. The analytic results from this paper and the recent numerical results from [16] indicate that further investigation on higher-order accuracy is needed.

Our numerical tests involving the Navier–Stokes equations showed that the convergence rate depends on the type of forcing method used to solve the momentum equation. We obtained approximately first-order convergence in the max-norm and second-order convergence in the 1-norm and 2-norm when the momentum equation was solved only on the physical domain and the velocity on the extended domain was set to zero. The convergence rates in the max-norm and 1-norm agreed

with those from the test of the projection alone. The better convergence in the 2-norm resulted because the first-order errors were concentrated at only a few points near the boundary, while in the test of the projection alone the larger errors were more uniform along the boundary.

The convergence rate of the forcing method in which the extended domain and the physical domain were coupled in the discrete momentum equation was between first- and second-order in both the max-norm and in the integral norms. The leading-order error is controlled by the jump in the derivative of the intermediate velocity across the embedded boundary. For this method, the effective jump decreased as the grid was refined, which gave the illusion of better than first-order convergence. The slower convergence in the integral norms indicates that the error did not remain concentrated near the embedded boundary. This may result from the stronger coupling of the physical domain and the extended domain in the discrete momentum equation.

In this paper we did not address the many different ways of using forcing to enforce the boundary conditions in the momentum equation. We only analyzed how performing the projection over the extended domain affects the accuracy of the method. It would be interesting to explore how the different treatments of the momentum equation affect the accuracy. Such a comparison is beyond the scope of the current paper. However, we comment on methods which use discretized delta functions to transmit the force from the immersed boundary to the Cartesian grid [6,7]. These methods effectively smear the location of the boundary over several grid cells, and so the intermediate velocity is thus seemingly smoother near the boundary. This smearing does not lead to higher-order accuracy because the intermediate velocity that results is generally only first-order accurate near the boundary [17–19] and because as the mesh is refined, the region over which the smearing occurs goes to zero. Despite the local smoothing, the velocity is again approaching a function that is not differentiable across the boundary.

Finally, we comment on how one might design a forcing method to obtain a second-order accurate velocity. To achieve this accuracy, the intermediate velocity must be differentiable across the embedded boundary. Since there is some freedom in the treatment of the velocity on the extension, it may be possible to force the velocity to be smooth across the embedded boundary. Since the larger errors remained highly localized in some of our numerical tests, it may be sufficient to ensure that the jump in the derivative occurs further inside the extended domain, away from the embedded boundary. It is not clear how to design efficient methods to achieve this smoothness. Additionally, one must be careful to avoid introducing a large gradient field on the extension, in which case our analysis breaks down, and convergence to the correct solution is not guaranteed.

Acknowledgments

The authors would like to thank Aaron Fogelson for helpful discussions related to this problem, and the referees for pointing out some useful additional references. The work of RG was supported in part by NSF-DMS Grant 0540779 and by UC Lab Fees Grant 09-LR-03-116724-GUYR.

Appendix A. Discretization details of Navier–Stokes tests

A.1. Zero velocity on extension

Given the velocity field, pressure, and pressure gradient at time level n, the solution is advanced in time in five steps, which are described below.

(1) **Advection terms:** The advection terms are differenced in conservative form $\nabla\cdot(\boldsymbol{uu})$. They are approximated at the half-time level by

$$\nabla\cdot(\boldsymbol{uu})^{n+1/2}=\frac{3}{2}\nabla\cdot(\boldsymbol{uu})^{n}-\frac{1}{2}\nabla\cdot(\boldsymbol{uu})^{n-1}. \tag{A.1}$$

The spatial differencing is performed with a centered difference. For example, the component of $\nabla\cdot(\boldsymbol{uu})$ in the x-direction is computed by

$$((uu)_x+(uv)_y)_{i+1/2j}=\frac{(uu)_{i+3/2j}-(uu)_{i-1/2j}}{2h}+\frac{(uv)_{i+3/2j}-(uv)_{i-1/2j}}{2h}. \tag{A.2}$$

Because a staggered grid is used, u and v are stored at different spatial locations. One of the velocity components must be spatially averaged to compute the products uv. In the difference formula above this product is computed by

$$(uv)_{i+1/2j}=u_{i+1/2j}\left(\frac{v_{ij-1/2}+v_{ij+1/2}+v_{i+1j-1/2}+v_{i+1j+1/2}}{4}\right). \tag{A.3}$$

The component in the y-direction is computed similarly. No modifications to differences are made near the embedded boundary. This introduces an $\mathcal{O}(1)$ local error, but contributes only an $\mathcal{O}(h^2)$ error to the solution of the momentum equation.

(2) **Viscous terms:** The viscous terms are discretized implicitly, using Crank–Nicolson discretization in time:

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -\nabla p^n + \frac{1}{2}(\Delta \boldsymbol{u}^* + \Delta \boldsymbol{u}^n) + \boldsymbol{g}^{n+1/2}, \tag{A.4}$$

where the term $\boldsymbol{g}^{n+1/2}$ includes the advection terms and background forces. The Laplacian is discretized using the standard, five-point, second-order difference away from the boundary. Adjacent to the embedded boundary the stencil is modified as described in Section 5.2.1. The physical domain and the extended domain decouple in this discretization. The discrete equations are solved using SOR.

(3) **Reset intermediate velocity on extension:** Before projecting the intermediate velocity is set to zero in the extension.

(4) **Projection:** The projection is performed over the extended domain, ignoring the embedded boundary. The projection is performed as described in Section 4.5 using Eqs. (26) and (27). The discrete equations are solved using multigrid. The pressure and the pressure gradient are updated by

$$p^{n+1} = p^n + \frac{\phi}{\Delta t}, \tag{A.5}$$

$$\nabla p^{n+1} = \nabla p^n + \frac{\nabla \phi}{\Delta t}. \tag{A.6}$$

This pressure update is only first-order for the pressure, but this error does not affect the error in the velocity field [23].

(5) **Reset velocity on the extension:** After performing the projection, the velocity field is reset to zero on the extended domain.

A.2. Forcing method

For the forcing method the advection terms and projection are handled exactly as described above. The forcing method differs from the method described in Section A.1 in two ways. The velocity is never reset on the extension, and the boundary conditions on the embedded boundary are enforced using direct forcing rather than modifying the stencil.

The momentum equation is again discretized in time using Crank–Nicolson:

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -\nabla p^n + \frac{1}{2}(\Delta \boldsymbol{u}^* + \Delta \boldsymbol{u}^n) + \boldsymbol{g}^{n+1/2} + \boldsymbol{f}, \tag{A.7}$$

where, as before, $\boldsymbol{g}^{n+1/2}$ includes the advection terms and any background forces, and $\boldsymbol{f}$ represents the forces needed to enforce the boundary conditions on the embedded boundary. The forcing terms are applied only at the points in the extended domain adjacent to the embedded boundary. We call these the forcing points. The forces are chosen so that the velocity field at these forcing points agrees with a velocity that is interpolated using points only from the physical domain and on the embedded boundary. The interpolation scheme is that used by Kim et al. [5], which is described below.

Algebraically, the interpolation can be written as

$$B_1 \boldsymbol{u}^* + B_2 \boldsymbol{u}_\mathrm{b} = I_\mathrm{F} \boldsymbol{u}^*, \tag{A.8}$$

where B_1 and B_2 are matrices defined by the interpolation scheme, $\boldsymbol{u}_\mathrm{b}$ represents the velocity evaluated at a set of discrete points on the embedded boundary, and I_F is the diagonal matrix corresponding to the discrete characteristic function of the forcing points. The left side of this equation is the velocity interpolated to the forcing points from the physical domain and the boundary points, and the right side is the value of the velocity at the forcing points.

Because the scheme is implicit, applying the forcing is not as straightforward as described by Eq. (4). Eqs. (A.7) and (A.8) together determine the velocity and force. To approximately solve this coupled system of equations, we use the iteration

$$\frac{\boldsymbol{u}^{*,k+1} - \boldsymbol{u}^n}{\Delta t} = -\nabla p^n + \frac{1}{2}(\Delta \boldsymbol{u}^{*,k+1} + \Delta \boldsymbol{u}^n) + \boldsymbol{g}^{n+1/2} + \boldsymbol{f}^k, \tag{A.9}$$

$$\boldsymbol{f}^{k+1} = \boldsymbol{f}^k + \frac{B_1 \boldsymbol{u}^{*,k+1} + B_2 \boldsymbol{u}_\mathrm{b} - I_\mathrm{F} \boldsymbol{u}^{*,k+1}}{\Delta t}. \tag{A.10}$$

This iteration is performed until $\|\boldsymbol{f}^{k+1} - \boldsymbol{f}^k\|_\infty < 1$, which is sufficient to obtain second-order accuracy in space and time. The accuracy of this scheme was tested, and in the absence of the projection, the solution is second-order accurate in space and time in the max-norm. We experimented with a tolerance of $\mathcal{O}(\Delta t)$, and the results did not change significantly.

To complete the description of the scheme, we give the details of the interpolation scheme, which is the scheme used by Kim et al. [5]. There are two types of forcing points: those with one neighbor inside the physical domain and those with two neighbors inside the physical domain. We ensure that there are no forcing points with three neighbors in the physical domain. First consider the case of a forcing point which has only one neighbor point in the physical domain. See Fig. A.1(a and b). Note that the figure is drawn as a node-centered grid for simplicity, but in the computation we use a staggered grid. There are five labeled collinear points in Fig. A.1(a and b). The point F is where the forcing is applied. Points A and C are mesh points in the physical domain. Point B is on the boundary in between points F and C at a distance ah away from point F, and point I is distance $2ah$ away, where $0 < a < 1$.

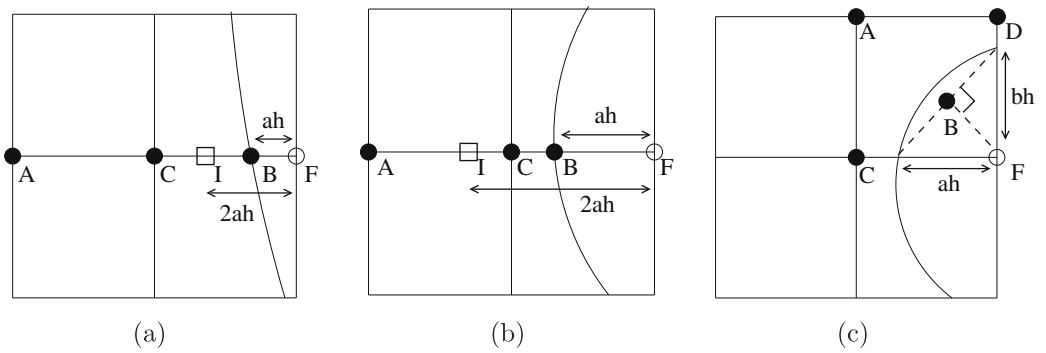

Fig. A.1. Solid circles represent the points used in the interpolation from the physical domain to the forcing point, which is represented by the open circle. (a and b) The interpolation is performed by two successive linear interpolations. First to the point I, represented by the square, followed by linear extrapolation to point F using the values at I and B. (c) The value at point B is set so that the bilinear interpolant through points A, D, C, F satisfies the boundary condition at point B.

The interpolation formula is derived in two steps. First the velocity is interpolated to point I using either points C and B, as in Fig. A.1(a), or using points C and A, as in Fig. A.1(b). The interpolated value at point F, U_F, is then linearly extrapolated using the values at points I and B. Combining these two interpolations gives the formula

$$U_F = \begin{cases} \frac{1}{1-a}U_B - \frac{a}{1-a}U_C & \text{if } a \leqslant 1/2, \\ 2U_B - 2(1-a)U_C + (1-2a)U_A & \text{if } a > 1/2. \end{cases} \tag{A.11}$$

Next consider the case when the forcing point has two neighbors in the physical domain, as depicted in Fig. A.1(c). As before, point F is where the forcing is applied. Points C and D represent the neighbor points in the physical domain and point A is the common neighbor point to points C and D. The boundary is approximated as a linear function through the two points where the boundary intersects the line segments CF and DF. The value at F results from requiring that the bilinear interpolant on ADFC satisfy the boundary condition at point B. This gives the interpolation formula at point F as

$$U_F = \frac{1}{(1-\alpha)(1-\beta)}U_B - \frac{\alpha}{1-\alpha}U_C - \frac{\beta}{1-\beta}U_D - \frac{\alpha\beta}{(1-\alpha)(1-\beta)}U_A, \tag{A.12}$$

where

$$\alpha = \frac{ab^2}{a^2+b^2} \quad \text{and} \quad \beta = \frac{a^2 b}{a^2+b^2}. \tag{A.13}$$

References

[1] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[2] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, J. Comput. Phys. 366 (1993) 354–366.
[3] J. Mohd-Yusof, Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries, Annual Research Briefs, Center for Turbulence Research, 1997.
[4] E.A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, J. Comput. Phys. 161 (2000) 35–60.
[5] J. Kim, D. Kim, H. Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, J. Comput. Phys. 171 (2001) 132–150.
[6] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, J. Comput. Phys. 209 (2005) 448–476.
[7] S.-W. Su, M.-C. Lai, C.-A. Lin, An immersed boundary technique for simulating complex flows with rigid boundary, Comput. Fluids 36 (2007) 313–324.
[8] N. Zhang, Z. Zheng, An improved direct-forcing immersed-boundary method for finite difference applications, J. Comput. Phys. 221 (2007) 250–268.
[9] H. Johansen, P. Colella, A Cartesian grid embedded boundary method for Poisson's equation on irregular domains, J. Comput. Phys. 147 (1) (1998) 60–85.
[10] F. Gibou, R.P. Fedkiw, L.-T. Cheng, M. Kang, A second-order-accurate symmetric discretization of the Poisson equation on irregular domains, J. Comput. Phys. 176 (1) (2002) 205–227.
[11] R. Mittal, H. Dong, M. Bozkurttas, F.M. Najjar, A. Vargas, A. von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, J. Comput. Phys. 227 (10) (2008) 4825–4852.
[12] Y.-H. Tseng, J.H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, J. Comput. Phys. 192 (2) (2003) 593–623.
[13] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, J. Comput. Phys. 156 (2) (1999) 209–240.
[14] E.M. Saiki, S. Biringen, Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method, J. Comput. Phys. 123 (1996) 450–465.
[15] F. Domenichini, On the consistency of the direct forcing method in the fractional step solution of the Navier–Stokes equations, J. Comput. Phys. 227 (2008) 6372–6384.
[16] M. Vanella, E. Balaras, A moving-least-squares reconstruction for embedded-boundary formulations, J. Comput. Phys. 228 (18) (2009) 6617–6628.
[17] R.P. Beyer, R.J. LeVeque, Analysis of a one-dimensional model for the immersed boundary method, SIAM J. Numer. Anal. 29 (1992) 332–364.
[18] Y. Mori, Convergence proof of the velocity field for a Stokes flow immersed boundary method, Commun. Pure Appl. Math. 61 (2007) 1213–1263.

[19] A.-K. Tornberg, B. Engquist, Numerical approximations of singular source terms in differential equations, J. Comput. Phys. 200 (2004) 462–488.
[20] A.J. Chorin, Numerical solutions of the Navier–Stokes equations, Math. Comput. 22 (1968) 745–762.
[21] A.J. Chorin, On the convergence of discrete approximations to the Navier–Stokes equations, Math. Comput. 23 (1969) 341–353.
[22] J.T. Beale, A.T. Layton, On the accuracy of finite difference methods for elliptic problems with interfaces, Commun. Appl. Math. Comput. Sci. 1 (2006) 91–119.
[23] D.L. Brown, R. Cortez, M.L. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 168 (2001) 464–499.