Journal of Computational Physics 228 (2009) 7565–7595

Contents lists available at ScienceDirect

Journal of Computational Physics

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

An accurate and efficient method for the incompressible Navier–Stokes equations using the projection method as a preconditioner

Boyce E. Griffith *

Leon H. Charney Division of Cardiology, Department of Medicine, New York University School of Medicine, 550 First Avenue, New York, NY 10016, United States

ARTICLE INFO

Article history:
Received 4 November 2008
Received in revised form 22 May 2009
Accepted 3 July 2009
Available online 10 July 2009

MSC:
65M06
65M12
65M55
76D05
76M20

Keywords:
Incompressible flow
Navier–Stokes equations
Preconditioner
Projection method
Block factorization
Approximate Schur complement
Physical boundary conditions
Multigrid

ABSTRACT

The projection method is a widely used fractional-step algorithm for solving the incompressible Navier–Stokes equations. Despite numerous improvements to the methodology, however, imposing physical boundary conditions with projection-based fluid solvers remains difficult, and obtaining high-order accuracy may not be possible for some choices of boundary conditions. In this work, we present an unsplit, linearly-implicit discretization of the incompressible Navier–Stokes equations on a staggered grid along with an efficient solution method for the resulting system of linear equations. Since our scheme is not a fractional-step algorithm, it is straightforward to specify general physical boundary conditions accurately; however, this capability comes at the price of having to solve the time-dependent incompressible Stokes equations at each timestep. To solve this linear system efficiently, we employ a Krylov subspace method preconditioned by the projection method. In our implementation, the subdomain solvers required by the projection preconditioner employ the conjugate gradient method with geometric multigrid preconditioning. The accuracy of the scheme is demonstrated for several problems, including forced and unforced analytic test cases and lid-driven cavity flows. These tests consider a variety of physical boundary conditions with Reynolds numbers ranging from 1 to 30000. The effectiveness of the projection preconditioner is compared to an alternative preconditioning strategy based on an approximation to the Schur complement for the time-dependent incompressible Stokes operator. The projection method is found to be a more efficient preconditioner in most cases considered in the present work.

1. Introduction

Since its introduction by Chorin [1,2], the projection method has been widely used as a solver for the incompressible Euler [3–8] and Navier–Stokes [9–29] equations. Generally speaking, the projection method is a fractional-step algorithm for incompressible flow problems which obtains updated values for the fluid velocity $\mathbf{u}$ and pressure p in two steps. First, an approximation to the momentum equation is solved over a time interval Δt *without* imposing the constraint of incompressibility, yielding an "intermediate" fluid velocity field $\mathbf{u}^*$, and generally $\nabla \cdot \mathbf{u}^* \neq 0$. To obtain an approximation to the updated fluid velocity which *does* satisfy the constraint of incompressibility, the projection method uses the Hodge decomposition to compute efficiently the projection of $\mathbf{u}^*$ onto the space of divergence-free vector fields. Doing so requires the solution of a

* Address: Smilow Research Building, New York University School of Medicine, 550 First Avenue, New York, United States. Tel.: +1 212 263 4131.
E-mail address: boyce.griffith@nyumc.org

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.07.001

Poisson equation. The solution φ of this linear equation is used both to project $\mathbf{u}^*$ to obtain the updated fluid velocity $\mathbf{u}$ and also to compute the updated pressure p.

The enduring popularity of the projection method may be attributed to its decomposition of a difficult problem into sub-problems for which efficient solvers are readily available. For instance, the highly influential second-order projection method of Bell et al. [10] requires only a solver for the diffusion equation arising from their implicit treatment of the viscous terms along with a Poisson solver for the discrete projection. Fast solvers employing FFT or multigrid algorithms may be used in both cases, thereby yielding incompressible Navier–Stokes solvers which are essentially algorithmically optimal.

Although the projection method framework yields timestepping schemes for the incompressible Navier–Stokes equations which require relatively simple linear solvers, this approach greatly complicates the specification of physical boundary conditions. Appropriate "artificial" boundary conditions must be prescribed for $\mathbf{u}^*$ and φ [18,23], and obtaining high-order accuracy for $\mathbf{u}$ and p may not be possible in some cases, such as at outflow boundaries [30]. Moreover, there is relatively little theory guiding the choice of approximations used to impose these artificial boundary conditions, and different standard approximations to the *same* artificial boundary conditions may result in schemes which are unstable or stable but low-order accurate, whereas non-standard approximations can result in schemes which are stable and high-order accurate [21].

In the present work, we view the projection method as an approximate solver for the time-dependent incompressible *Stokes* equations, and we use this approximate solver as a *preconditioner* for the iterative solution of those equations. This iterative solver is used with a linearly-implicit staggered-grid discretization of the incompressible *Navier–Stokes* equations which employs an implicit treatment of the viscous terms and an explicit second-order Godunov (upwind) method for the nonlinear advection terms. The Godunov scheme used in the present work is based on xsPPM7 [31], a recent version of the piecewise parabolic method (PPM) [32].

By using an unsplit scheme instead of a fractional-step algorithm, we are able to prescribe general physical boundary conditions accurately and, in most cases, easily. (There appears to be some ambiguity in the specification of normal traction boundary conditions; we describe herein the approach which we have found to be the most accurate and robust.) Numerical results for forced and unforced analytic test cases presented in Section 5 demonstrate that our scheme attains fully second-order convergence rates over a broad range of Reynolds numbers, from $Re = 1$ to $Re \approx 3000$, for a variety of non-trivial physical boundary conditions. The accuracy of the scheme is also demonstrated at $Re \approx 30000$ for a variety of boundary conditions, although these results appear under-resolved on all but the finest computational grids employed in the present work. When we employ inexact multigrid-preconditioned subdomain solvers, our numerical results demonstrate that the scheme is scalable in the sense that the number of linear solver and subdomain solver iterations are largely insensitive to the grid spacing. In many cases, the number of solver/sub-solver iterations is also demonstrated to be largely independent of Re.

The scheme is also demonstrated to converge to benchmark steady solutions [33–35] to the lid-driven cavity flow problem for $Re = 1000, 5000$, and 7500. Essentially first-order global convergence rates are obtained for the standard lid-driven cavity flow, which possesses well-known corner singularities [34], and fully second-order pointwise convergence rates are obtained for a regularized version of this problem [36]. We also demonstrate that improved accuracy (although not improved order of accuracy) may be obtained for the regularized lid-driven cavity problem by employing an alternative third-order boundary treatment. Because we do not use a fractional-step scheme, it is straightforward to use alternative, higher-order boundary condition implementations.

The projection method-based preconditioner described in the present work is similar to but distinct from a preconditioner based on an approximation to the Schur complement for the time-dependent incompressible Stokes operator. Such approximate Schur complement methods were originally introduced in the context of the Oseen equations [37,38], and these methods have been thoroughly studied by Elman and co-workers [39–41]. (Although it seems likely that the projection preconditioner could be extended to treat the Oseen equations, this has not yet been done.) In the results presented in Section 5, we compare the efficiency of the projection preconditioner to a preconditioner based on an approximate Schur complement. In nearly all cases, the projection method is found to be a significantly more efficient preconditioner than the approximate Schur complement method. Although other preconditioning strategies for the incompressible Navier–Stokes equations have been described (see, e.g., [42,43]), most alternative approaches do not appear to use subdomain solvers.

Although we describe a particular linearly-implicit timestepping scheme for the incompressible Navier–Stokes equations, we believe that our preconditioning approach is widely applicable. In particular, we expect that the basic approach described in the present work could be used to convert an existing fractional-step staggered-grid incompressible flow solver into an unsplit solver. Doing so would require only "wrapping" the existing projection code with a Krylov solver. The additional software required is modest and includes: (1) the implementation of code to apply the time-dependent incompressible Stokes operator to a given vector; (2) the implementation of a (possibly matrix-free) preconditioned Krylov solver for the incompressible Stokes system; and (3) the conversion of the existing staggered-grid projection solver into a preconditioner. Most of the work required by this conversion could be eliminated by using a high-quality numerical software library such as PETSc [44–46].

Finally, before proceeding, we note that although the present work treats the case of two spatial dimensions, the modifications required to extend the methodology to three spatial dimensions are straightforward. Results from an initial three-dimensional implementation of the scheme are encouraging and will be reported in future work.

2. The continuous equations of motion

Consider a fixed region $\Omega \subset \mathbb{R}^2$ which is filled with a viscous incompressible fluid. (To simplify the presentation, in this work the physical domain Ω is taken to be the unit square.) The equations of motion for the fluid are the incompressible Navier–Stokes equations

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{f}, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2}$$

where $\mathbf{x} = (x,y) \in \Omega$ are fixed physical coordinates, $\mathbf{u}(\mathbf{x},t) = (u(\mathbf{x},t), v(\mathbf{x},t))$ is the fluid velocity, $p(\mathbf{x},t)$ is the pressure, $\mathbf{f}(\mathbf{x},t) = (f_1(\mathbf{x},t), f_2(\mathbf{x},t))$ is an applied body force, ρ is the uniform fluid density, and μ is the uniform dynamic viscosity of the fluid. We assume that the body force is a given function and is not a function of $\mathbf{u}$ or p. Completing the specification of the problem requires initial conditions for the velocity (but not the pressure) along with boundary conditions.

In the present work, three types of boundary conditions are considered: periodic, prescribed velocity, and prescribed traction. Letting $\mathbf{n} = \mathbf{n}(\mathbf{x}_b)$ denote the outward unit normal at a position $\mathbf{x}_b$ along the domain boundary $\partial\Omega$, and letting $\boldsymbol{\tau} = \boldsymbol{\tau}(\mathbf{x}_b)$ denote the unit tangent vector at $\mathbf{x}_b \in \partial\Omega$, prescribing normal or tangential velocity boundary conditions at a position $\mathbf{x}_b \in \partial\Omega$ is equivalent to providing values for $\mathbf{u}\cdot\mathbf{n}$ or $\mathbf{u}\cdot\boldsymbol{\tau}$ at $\mathbf{x}_b$. Note that if a normal velocity boundary condition is prescribed at a position $\mathbf{x}_b \in \partial\Omega$, then no boundary condition may be imposed for the pressure at $\mathbf{x}_b$. Prescribing normal or tangential traction boundary conditions at a position $\mathbf{x}_b \in \partial\Omega$ is equivalent to providing values for components of $\boldsymbol{\sigma}\cdot\mathbf{n}$, the stress normal to the domain boundary, at $\mathbf{x}_b$. Since the stress tensor for a viscous incompressible fluid is

$$\boldsymbol{\sigma} = -p\mathbb{I} + \mu[\nabla\mathbf{u} + (\nabla\mathbf{u})^T], \tag{3}$$

prescribing the normal traction is equivalent to prescribing the value of the normal component of the normal stress,

$$\mathbf{n}\cdot\boldsymbol{\sigma}\cdot\mathbf{n} = -p + 2\mu\frac{\partial}{\partial n}(\mathbf{u}\cdot\mathbf{n}), \tag{4}$$

at the boundary, and prescribing the tangential traction is equivalent to prescribing the value of the tangential component of the normal stress,

$$\boldsymbol{\tau}\cdot\boldsymbol{\sigma}\cdot\mathbf{n} = \mu\left(\frac{\partial}{\partial n}(\mathbf{u}\cdot\boldsymbol{\tau}) + \frac{\partial}{\partial\tau}(\mathbf{u}\cdot\mathbf{n})\right), \tag{5}$$

at the boundary. It is important to note that the specification of the normal traction at a position $\mathbf{x}_b \in \partial\Omega$ is equivalent to specifying a linear combination of $p(\mathbf{x}_b)$ and $(\frac{\partial}{\partial n}(\mathbf{u}\cdot\mathbf{n}))(\mathbf{x}_b)$. In practice, we supplement the normal traction boundary condition with the divergence-free condition at the boundary to obtain two equations for these two boundary values.

It is possible to specify boundary values for both the normal and tangential components of the velocity, to specify boundary values for both the normal and tangential tractions, or to specify boundary values for one component of the velocity and the other component of the traction. In particular, it is possible to prescribe the normal velocity and the tangential traction, or to prescribe the normal traction and the tangential velocity. We consider all four cases in the present work. For more details on boundary conditions for the incompressible Navier–Stokes equations, including discussions of alternative boundary treatments, see, e.g., Chapter 3, Section 8 of Gresho and Sani [47].

3. The discretized equations of motion

3.1. Spatial discretization and finite difference approximations

In the present work, we employ a staggered-grid spatial discretization of the incompressible Navier–Stokes equations. (The layout of degrees of freedom used in the present work is the same as the MAC scheme [48], although note that we use spatial and temporal discretizations which are different from those of the MAC scheme.) Briefly, the physical domain Ω is described using a fixed $N \times N$ Cartesian grid with uniform meshwidths $\Delta x = \Delta y = h = \frac{1}{N}$. The discretized velocity field is defined in terms of those vector components that are normal to the *edges* of the grid cells (or, in three spatial dimensions, the *faces* of the grid cells), and the pressure is defined at the *centers* of the grid cells. See Fig. 1.

Before describing the finite difference approximations to the spatial differential operators appearing in Eqs. (1) and (2), we introduce notation to describe the quantities defined on the grid. The centers of the Cartesian grid cells are the points $\mathbf{x}_{i,j} = ((i+\frac{1}{2})h, (j+\frac{1}{2})h)$, where $i,j = 0,\ldots,N-1$. The pressure $p(\mathbf{x},t)$ is defined at the centers of the Cartesian grid cells, and the values of p on the grid are denoted $p^n_{i,j} = p(\mathbf{x}_{i,j}, t^n)$, where t^n is the time of the n^{th} timestep, and $\Delta t^n = \Delta t = t^{n+1} - t^n$. The centers of the x-edges of the grid cells (i.e., the centers of the cell edges $x = \text{constant}$) are the points $\mathbf{x}_{i-\frac{1}{2},j} = (ih, (j+\frac{1}{2})h)$, where $i = 0,\ldots,N$ and $j = 0,\ldots,N-1$, and the centers of the y-edges of the grid cells (i.e., the centers of the cell edges $y = \text{constant}$) are the points $\mathbf{x}_{i,j-\frac{1}{2}} = ((i+\frac{1}{2})h, jh)$, where $i = 0,\ldots,N-1$ and $j = 0,\ldots,N$. The u-component of the fluid velocity is defined at the centers of the x-edges of the grid cells, and the v-component of the fluid velocity is defined at the centers of the y-edges of the grid cells. In particular, the values of u on the grid are denoted $u^n_{i-\frac{1}{2},j} = u(\mathbf{x}_{i-\frac{1}{2},j}, t^n)$, and

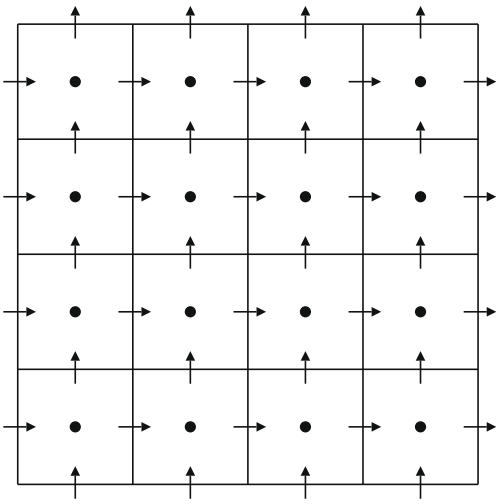

Fig. 1. A staggered-grid spatial discretization. The velocity field $\mathbf{u}$ is defined in terms of those vector components that are normal to the edges of the grid cells, and the pressure p is defined at the centers of the grid cells.

the values of v on the grid are denoted $v^n_{i,j-\frac{1}{2}} = v(\mathbf{x}_{i,j-\frac{1}{2}}, t^n)$. The components of the given body force $\mathbf{f} = (f_1, f_2)$ are likewise defined at the centers of the x- and y-edges of the Cartesian grid cells.

We next introduce notation for the finite difference approximations to the spatial differential operators appearing in the equations of motion. The divergence of $\mathbf{u} = (u, v)$ is approximated at cell centers by

$$\mathbf{D} \cdot \mathbf{u} = D^x u + D^y v, \tag{6}$$

$$(D^x u)_{i,j} = \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{h}, \tag{7}$$

$$(D^y v)_{i,j} = \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{h}, \tag{8}$$

and the gradient of p is approximated at the x- and y-edges of the grid cells by

$$\mathbf{G}p = (G^x p, G^y p), \tag{9}$$

$$(G^x p)_{i-\frac{1}{2},j} = \frac{p_{i,j} - p_{i-1,j}}{h}, \tag{10}$$

$$(G^y p)_{i,j-\frac{1}{2}} = \frac{p_{i,j} - p_{i,j-1}}{h}. \tag{11}$$

There are three different approximations to the Laplace operator required in the present scheme, one defined at the cell centers, one defined at the x-edges of the grid, and one defined at the y-edges of the grid. All employ the same standard 5-point finite difference stencil. The Laplacian of p is approximated at cell centers by

$$(L^c p)_{i,j} = \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{h^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{h^2}, \tag{12}$$

whereas the approximations to the Laplacian of u (evaluated at x-edges) and v (evaluated at y-edges) are

$$(L^x u)_{i-\frac{1}{2},j} = \frac{u_{i+\frac{1}{2},j} - 2u_{i-\frac{1}{2},j} + u_{i-\frac{3}{2},j}}{h^2} + \frac{u_{i-\frac{1}{2},j+1} - 2u_{i-\frac{1}{2},j} + u_{i-\frac{1}{2},j-1}}{h^2}, \tag{13}$$

$$(L^y v)_{i,j-\frac{1}{2}} = \frac{v_{i+1,j-\frac{1}{2}} - 2v_{i,j-\frac{1}{2}} + v_{i-1,j-\frac{1}{2}}}{h^2} + \frac{v_{i,j+\frac{1}{2}} - 2v_{i,j-\frac{1}{2}} + v_{i,j-\frac{3}{2}}}{h^2}. \tag{14}$$

The finite difference approximation to the vector Laplacian of $\mathbf{u} = (u, v)$ is denoted $\mathbf{L}\mathbf{u} = (L^x u, L^y v)$. It is important to keep in mind that although L^c, L^x, and L^y are essentially the same discretization of the Laplacian, each is applied to a different variable.

Evaluating the foregoing finite difference approximations near the boundaries of the computational domain requires the specification of boundary values along $\partial\Omega$ and "ghost" values located outside of Ω. At periodic boundaries, the ghost values are copies of the corresponding interior values, and at physical boundaries, the boundary and ghost values are determined from the physical boundary conditions as described in Section 3.3.

The nonlinear term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ is evaluated using a version of the piecewise parabolic method (PPM) of Colella and Woodward [32]. The particular approach used in the present work is based on the xsPPM7 scheme of Rider et al. [31]. PPM was

originally developed to simulate compressible fluid dynamics, and it continues to be widely used in various areas of computational fluid dynamics. PPM is typically employed to extrapolate cell-centered values defined at time t^n to edge-centered values (in two spatial dimensions) or face-centered values (in three spatial dimensions) defined at time $t^{n+\frac{1}{2}} = t^n + \frac{1}{2}\Delta t$. In the present staggered-grid context, however, we employ PPM *only* to extrapolate values in space and do *not* use PPM to extrapolate values forward in time. An overview of our edge-centered Godunov scheme is provided in Appendix A.

3.2. Time discretization

We now turn our attention to the temporal discretization of the equations of motion. One possible time discretization is obtained by the straightforward application of the implicit midpoint rule to the incompressible Navier–Stokes equations. Doing so yields

$$\rho\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}+\mathbf{N}\left(\mathbf{u}^{n+\frac{1}{2}}\right)\right) = -\mathbf{G}p^{n+\frac{1}{2}}+\mu\mathbf{L}\mathbf{u}^{n+\frac{1}{2}}+\mathbf{f}^{n+\frac{1}{2}}, \tag{15}$$

$$\mathbf{D}\cdot\mathbf{u}^{n+1} = 0, \tag{16}$$

where $\mathbf{u}^{n+\frac{1}{2}} = \frac{1}{2}(\mathbf{u}^{n+1}+\mathbf{u}^n)$ and $\mathbf{N}(\mathbf{u}^{n+\frac{1}{2}}) \approx \left[\left(\mathbf{u}^{n+\frac{1}{2}}\cdot\nabla\right)\mathbf{u}^{n+\frac{1}{2}}\right]$ is the PPM approximation to the nonlinear advection term. Since $\mathbf{N}(\mathbf{u})$ is a nonlinear function of $\mathbf{u}$, using this fully implicit time discretization would require solving a *nonlinear* system of algebraic equations at each timestep.

Rather than solving Eqs. (15) and (16) exactly, we instead employ a fixed number n_{cycles} of steps of fixed-point iteration to obtain an *approximate* solution to the fully-coupled nonlinear problem. In our approach, we treat the linear terms implicitly and the nonlinear terms explicitly, always using the most recently computed approximation to $\mathbf{u}^{n+\frac{1}{2}}$ when evaluating the nonlinear advection term. Let $\mathbf{u}^{n+1,k}$ and $p^{n+\frac{1}{2},k}$ denote the approximations to $\mathbf{u}^{n+1}$ and $p^{n+\frac{1}{2}}$ obtained after k steps of fixed-point iteration. We obtain $\mathbf{u}^{n+1,k+1}$ and $p^{n+\frac{1}{2},k+1}$ from $\mathbf{u}^{n+1,k}$ and $p^{n+\frac{1}{2},k}$ by solving the *linear* system of equations

$$\rho\left(\frac{\mathbf{u}^{n+1,k+1}-\mathbf{u}^n}{\Delta t}+\mathbf{N}\left(\mathbf{u}^{n+\frac{1}{2},k}\right)\right) = -\mathbf{G}p^{n+\frac{1}{2},k+1}+\mu\mathbf{L}\mathbf{u}^{n+\frac{1}{2},k+1}+\mathbf{f}^{n+\frac{1}{2}}, \tag{17}$$

$$\mathbf{D}\cdot\mathbf{u}^{n+1,k+1} = 0, \tag{18}$$

where $\mathbf{u}^{n+\frac{1}{2},k+1} = \frac{1}{2}(\mathbf{u}^{n+1,k+1}+\mathbf{u}^n)$ and $\mathbf{N}(\mathbf{u}^{n+\frac{1}{2},k}) \approx \left[\left(\mathbf{u}^{n+\frac{1}{2},k}\cdot\nabla\right)\mathbf{u}^{n+\frac{1}{2},k}\right]$ is an explicit PPM approximation to the nonlinear advection term. Except for the explicit approximation to $(\mathbf{u}\cdot\nabla)\mathbf{u}$, Eqs. (15) and (16) and Eqs. (17) and (18) are identical. The final updated values of the velocity and pressure are $\mathbf{u}^{n+1} = \mathbf{u}^{n+1,n_{\text{cycles}}}$ and $p^{n+\frac{1}{2}} = p^{n+\frac{1}{2},n_{\text{cycles}}}$, i.e., each timestep requires the computation of n_{cycles} solutions of the time-dependent incompressible Stokes equations.

For each timestep $n > 0$, initial approximations to the velocity and pressure are obtained from the timestep-lagged velocity and pressure, i.e., $\mathbf{u}^{n+1,0} = \mathbf{u}^n$ and $p^{n+\frac{1}{2},0} = p^{n-\frac{1}{2}}$. For the initial timestep $n = 0$, we use the initial conditions to determine $\mathbf{u}^{1,0} = \mathbf{u}^0$. Unless we solve an auxiliary system of equations, however, an initial value for the pressure is not generally available. For simplicity, we set $p^{\frac{1}{2},0} = 0$. Note that the value of $p^{n+\frac{1}{2},k}$ does not affect the value of $p^{n+\frac{1}{2},k+1}$. Instead, $p^{n+\frac{1}{2},k}$ only serves as an initial guess for the iterative solution of Eqs. (17) and (18) as described in Section 4. Consequently, although setting $p^{\frac{1}{2},0} = 0$ may increase the number of iterations required for the initial linear solve to converge when compared to subsequent solves, setting $p^{\frac{1}{2},0} = 0$ does *not* affect the accuracy of $p^{\frac{1}{2}}$.

Note that if the fixed-point iterations are truncated after only a single step ($n_{\text{cycles}} = 1$; one Stokes solve per timestep), this scheme corresponds to the application of forward Euler to the advection terms and Crank–Nicolson to the viscous terms, yielding a first-order accurate temporal discretization. Similarly, if the iterations are truncated after two steps ($n_{\text{cycles}} = 2$; two Stokes solves per timestep), this scheme is similar to the combination of the explicit midpoint rule for the advection terms and Crank–Nicolson for the viscous terms, yielding a second-order accurate temporal discretization. In practice, we typically truncate the fixed-point iterations after three steps. Although the resulting method is still only second-order accurate in time, we have found that truncating the fixed-point iterations after three steps ($n_{\text{cycles}} = 3$) appears to yield a significant improvement in stability compared to using only two steps. In particular, with $n_{\text{cycles}} = 2$, numerical experiments suggest that the largest stable CFL number for the scheme is 0.5, whereas with $n_{\text{cycles}} = 3$, we have found that the scheme remains stable up to a CFL number of 1. Moreover, the additional solve is often relatively inexpensive in practice because $\mathbf{u}^{n+1,2}$ and $p^{n+\frac{1}{2},2}$ are typically fairly accurate initial approximations to $\mathbf{u}^{n+1,3}$ and $p^{n+\frac{1}{2},3}$. In cases in which the scheme remains stable for $n_{\text{cycles}} = 2$, however, note that there appears to be essentially no improvement in accuracy for $n_{\text{cycles}} > 2$. In the tests reported in the present work, we always employ $n_{\text{cycles}} = 3$.

3.3. Physical boundary conditions

We now turn our attention the determination of the boundary and ghost values required at physical boundaries by the foregoing discretization during the time interval $[t^n, t^{n+1}]$. To simplify the presentation, we restrict our attention to the domain boundary in the vicinity of a single grid cell $(N-1, j), 0 \leqslant j < N$, which is located along the right side of the physical domain; see Fig. 2. Along this portion of the domain boundary, the outward unit normal vector is $\mathbf{n} = (1, 0)$ and the unit tangent vector is $\boldsymbol{\tau} = (0, 1)$. With $\mathbf{u} = (u, v)$, the normal velocity at the boundary is $\mathbf{u}\cdot\mathbf{n} = u$, the tangential velocity is $\mathbf{u}\cdot\boldsymbol{\tau} = v$,

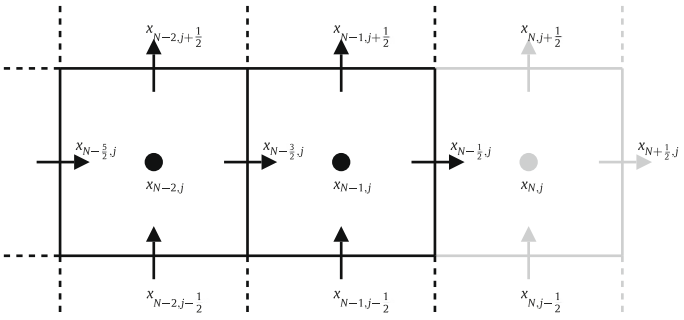

Fig. 2. Positions of the velocity components and pressure in the vicinity of cell $(N-1,j)$, which is located along the boundary along the right side of the physical domain. Interior and boundary values are indicated in black, and ghost values located outside of the physical domain are indicated in grey.

and the stress normal to the domain boundary is $\boldsymbol{\sigma}\cdot\mathbf{n} = \left(-p+2\mu\frac{\partial u}{\partial x}, \mu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right)$. The boundary treatment along the other sides of the physical domain is analogous.

For the types of boundary conditions considered in the present work, there are four cases to consider.

(1) The normal velocity is prescribed at position $\mathbf{x}_{N-\frac{1}{2}j}$:
Suppose that the normal velocity is provided at position $\mathbf{x}_{N-\frac{1}{2}j}$ as an explicit function of time, $u^{\text{norm}}_{N-\frac{1}{2}j}(t)$. We directly impose this boundary condition at time $t=t^n$ by setting

$$u^n_{N-\frac{1}{2}j} = u^{\text{norm}}_{N-\frac{1}{2}j}(t^n). \tag{19}$$

The value of $u^{n+1}_{N-\frac{1}{2}j}$ is specified analogously. In this case, no expressions are required for the ghost values $p_{N,j}$ or $u_{N+\frac{1}{2}j}$. In particular, in this case, no boundary condition for the pressure is needed, nor may a boundary condition for the pressure be prescribed.

(2) The normal traction is prescribed at position $\mathbf{x}_{N-\frac{1}{2}j}$:
Suppose that the normal traction is provided at position $\mathbf{x}_{N-\frac{1}{2}j}$ as an explicit function of time, $F^{\text{norm}}_{N-\frac{1}{2}j}(t)$. In this case, the boundary condition is

$$(\mathbf{n}\cdot\boldsymbol{\sigma}\cdot\mathbf{n})\left(\mathbf{x}_{N-\frac{1}{2}j},t\right) = \left(-p+2\mu\frac{\partial u}{\partial x}\right)\left(\mathbf{x}_{N-\frac{1}{2}j},t\right) = F^{\text{norm}}_{N-\frac{1}{2}j}(t) \tag{20}$$

and we must obtain ghost values for both u and p. To obtain two equations for the two unknown boundary values $\frac{\partial u}{\partial x}$ and p, we supplement the traction boundary condition (20) with the divergence-free condition at the boundary,

$$(\nabla\cdot\mathbf{u})(\mathbf{x}_{N-\frac{1}{2}j},t) = \left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\left(\mathbf{x}_{N-\frac{1}{2}j},t\right) = 0, \tag{21}$$

i.e.,

$$\frac{\partial u}{\partial x}\left(\mathbf{x}_{N-\frac{1}{2}j},t\right) = -\frac{\partial v}{\partial y}\left(\mathbf{x}_{N-\frac{1}{2}j},t\right). \tag{22}$$

Our approach is to obtain ghost values for u at times $t=t^n$ and t^{n+1} via a finite difference approximation to Eq. (22). We then obtain a ghost value for p at time $t=t^{n+\frac{1}{2}}$ via a finite difference approximation to Eq. (20) along with the previously-determined ghost values for u^n and u^{n+1}.

First, we use a finite difference approximation to Eq. (22) to obtain a ghost values for $u^n_{N+\frac{1}{2}j}$. To do so, we compute linearly-extrapolated values of the tangential velocity at the boundary via

$$v^n_{N-\frac{1}{2}j\pm\frac{1}{2}} = \frac{3}{2}v^n_{N-1j\pm\frac{1}{2}} - \frac{1}{2}v^n_{N-2j\pm\frac{1}{2}}. \tag{23}$$

Using these extrapolated boundary values, we then compute a finite difference approximation to the divergence-free condition via

$$\frac{u^n_{N+\frac{1}{2}j}-u^n_{N-\frac{3}{2}j}}{2h} + \frac{v^n_{N-\frac{1}{2}j+\frac{1}{2}}-v^n_{N-\frac{1}{2}j-\frac{1}{2}}}{h} = 0, \tag{24}$$

thereby obtaining a formula for the normal component of the velocity at the boundary at time $t=t^n$ in terms of values defined on the staggered computational grid, namely

$$u^n_{N+\frac{1}{2}j} = u^n_{N-\frac{3}{2}j} - 3v^n_{N-1j+\frac{1}{2}} + v^n_{N-2j+\frac{1}{2}} + 3v^n_{N-1j-\frac{1}{2}} - v^n_{N-2j-\frac{1}{2}}. \tag{25}$$

The normal component of the velocity at the boundary at time $t=t^{n+1}$ is similarly

$$u^{n+1}_{N+\frac{1}{2}j} = u^{n+1}_{N-\frac{3}{2}j} - 3v^{n+1}_{N-1j+\frac{1}{2}} + v^{n+1}_{N-2j+\frac{1}{2}} + 3v^{n+1}_{N-1j-\frac{1}{2}} - v^{n+1}_{N-2j-\frac{1}{2}}. \tag{26}$$

Having determined ghost values for the normal velocity, we next employ a timestep-centered finite difference approximation to the normal traction boundary condition,

$$-\frac{p^{n+\frac{1}{2}}_{Nj} + p^{n+\frac{1}{2}}_{N-1j}}{2} + \mu\left(\frac{u^{n}_{N+\frac{1}{2}j} - u^{n}_{N-\frac{3}{2}j}}{2h} + \frac{u^{n+1}_{N+\frac{1}{2}j} - u^{n+1}_{N-\frac{3}{2}j}}{2h}\right) = F^{\text{norm}}_{N-\frac{1}{2}j}\left(t^{n+\frac{1}{2}}\right). \tag{27}$$

Plugging Eqs. (25) and (26) into Eq. (27) along with minor rearrangement yields an expression for $p^{n+\frac{1}{2}}_{Nj}$ which involves only interior values defined on the staggered grid along with the prescribed boundary value, $F^{\text{norm}}_{N-\frac{1}{2}j}\left(t^{n+\frac{1}{2}}\right)$.

(3) The tangential velocity is prescribed at position $\mathbf{x}_{N-\frac{1}{2}j-\frac{1}{2}}$:
Suppose that the tangential velocity is provided at position $\mathbf{x}_{N-\frac{1}{2}j-\frac{1}{2}}$ as an explicit function of time, $v^{\tan}_{N-\frac{1}{2}j-\frac{1}{2}}(t)$. We impose this boundary condition at time $t = t^n$ via a linear fit to the nearest internal value along with the prescribed boundary value, so that

$$v^{n}_{Nj-\frac{1}{2}} = 2v^{\tan}_{N-\frac{1}{2}j-\frac{1}{2}}(t^n) - v^{n}_{N-1j-\frac{1}{2}}. \tag{28}$$

The value of $v^{n+1}_{Nj-\frac{1}{2}}$ is determined analogously.

(4) The tangential traction is prescribed at position $\mathbf{x}_{N-\frac{1}{2}j-\frac{1}{2}}$:
Suppose that the tangential traction is provided at position $\mathbf{x}_{N-\frac{1}{2}j-\frac{1}{2}}$ as an explicit function of time, $F^{\tan}_{N-\frac{1}{2}j-\frac{1}{2}}(t)$. We impose the boundary condition

$$(\boldsymbol{\tau}\cdot\boldsymbol{\sigma}\cdot\mathbf{n})\left(\mathbf{x}_{N-\frac{1}{2}j-\frac{1}{2}}, t\right) = \mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)\left(\mathbf{x}_{N-\frac{1}{2}j-\frac{1}{2}}, t\right) = F^{\tan}_{N-\frac{1}{2}j-\frac{1}{2}}(t) \tag{29}$$

at time t^n via the finite difference approximation

$$\mu\left(\frac{v^{n}_{Nj-\frac{1}{2}} - v^{n}_{N-1j-\frac{1}{2}}}{h} + \frac{u^{n}_{N-\frac{1}{2}j} - u^{n}_{N-\frac{1}{2}j-1}}{h}\right) = F^{\tan}_{N-\frac{1}{2}j-\frac{1}{2}}(t^n), \tag{30}$$

which, after minor rearrangement, yields an explicit formula for $v^{n}_{Nj-\frac{1}{2}}$ involving only interior values defined on the staggered grid along with the prescribed boundary value, $F^{\tan}_{N-\frac{1}{2}j-\frac{1}{2}}(t^n)$. An analogous formula is obtained for $v^{n+1}_{Nj-\frac{1}{2}}$.

4. Linear solvers

Solving for $\mathbf{u}^{n+1,k+1}$ and $p^{n+\frac{1}{2},k+1}$ in Eqs. (17) and (18) requires the solution of the block linear system

$$\begin{pmatrix} \frac{\rho}{\Delta t}I - \frac{\mu}{2}L^x & 0 & G^x \\ 0 & \frac{\rho}{\Delta t}I - \frac{\mu}{2}L^y & G^y \\ -D^x & -D^y & 0 \end{pmatrix}\begin{pmatrix} u^{n+1,k+1} \\ v^{n+1,k+1} \\ p^{n+\frac{1}{2},k+1} \end{pmatrix} = \begin{pmatrix} \left(\frac{\rho}{\Delta t}I + \frac{\mu}{2}L^x\right)u^n - \rho N_1^{n+\frac{1}{2},k} + f_1^{n+\frac{1}{2}} \\ \left(\frac{\rho}{\Delta t}I + \frac{\mu}{2}L^y\right)v^n - \rho N_2^{n+\frac{1}{2},k} + f_2^{n+\frac{1}{2}} \\ 0 \end{pmatrix}, \tag{31}$$

where $\mathbf{N}\left(\mathbf{u}^{n+\frac{1}{2},k}\right) = \left(N_1^{n+\frac{1}{2},k}, N_2^{n+\frac{1}{2},k}\right)$ and $\mathbf{f}^{n+\frac{1}{2}} = \left(f_1^{n+\frac{1}{2}}, f_2^{n+\frac{1}{2}}\right)$. Note that the choice of sign for D^x and D^y in Eq. (31) makes the matrix symmetric for certain choices of boundary conditions, e.g., in the case of periodic boundary conditions.

We solve Eq. (31) via the flexible GMRES (FGMRES) algorithm [49] using $\mathbf{u}^{n+1,k}$ and $p^{n+\frac{1}{2},k}$ as initial approximations to $\mathbf{u}^{n+1,k+1}$ and $p^{n+\frac{1}{2},k+1}$, and with one of the two preconditioners described in Sections 4.1 and 4.2. Both preconditioners are constructed using preconditioned conjugate gradient (CG) solvers for $A^x = \left(\frac{\rho}{\Delta t}I - \frac{\mu}{2}L^x\right), A^y = \left(\frac{\rho}{\Delta t}I - \frac{\mu}{2}L^y\right)$, and $-L^c = -\mathbf{D}\cdot\mathbf{G}$. In the following discussion, we shall also make use of the operator $A^c = \left(\frac{\rho}{\Delta t}I - \frac{\mu}{2}L^c\right)$. Recall that although L^x, L^y, and L^c are all essentially the same finite difference approximation to $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, each is applied to a different variable. Similarly, A^x, A^y, and A^c are all essentially the same discretized operator, but each acts on a different variable.

Note that not all Krylov methods allow for the use of other Krylov methods as preconditioners. For instance, preconditioners for standard Krylov methods such as CG, GMRES, and Bi-CGSTAB are required to be fixed linear operators. In other words, for such methods, the application of the preconditioner to a vector must correspond to matrix–vector multiplication by a matrix which does not vary between Krylov iterations. This requirement precludes the use of another Krylov method in the preconditioner because Krylov methods are non-stationary iterative methods, i.e., the application of one or more steps of a Krylov method does not correspond to multiplication by a fixed matrix. FGMRES and other so-called *flexible* Krylov methods are designed to allow for the preconditioner to vary between iterations. In particular, such algorithms allow for the use of other Krylov methods as preconditioners; see [50,51] and the references therein.

4.1. A preconditioner based on the projection method

To use the projection method as a preconditioner for Eq. (31) (see Appendix B for a brief description of the particular second-order projection method used to construct the projection preconditioner), it must be recast as an approximate solver for the linear system

$$\begin{pmatrix} A^x & 0 & G^x \\ 0 & A^y & G^y \\ -D^x & -D^y & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{G} \\ -\mathbf{D}\cdot & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ f_p \end{pmatrix} \tag{32}$$

with homogeneous boundary conditions, where $\mathbf{u} = (u, v), \mathbf{f} = (f_u, f_v)$, and

$$\mathbf{A} = \begin{pmatrix} A^x & 0 \\ 0 & A^y \end{pmatrix} = \begin{pmatrix} \frac{\rho}{\Delta t} I - \frac{\mu}{2} L^x & 0 \\ 0 & \frac{\rho}{\Delta t} I - \frac{\mu}{2} L^y \end{pmatrix}. \tag{33}$$

It is important to note that in a Krylov method, the preconditioner is applied to the residual vector, and in general, the residual does not satisfy the discrete incompressibility condition. Therefore, the projection preconditioner must allow for the case that $-\mathbf{D}\cdot\mathbf{u} = f_p \neq 0$.

We convert the projection method into a preconditioner as follows:

First, we compute $\mathbf{u}^* = (u^*, v^*)$, an "intermediate" approximation to $\mathbf{u} = (u, v)$, by solving

$$A^x u^* = \left(\frac{\rho}{\Delta t} I - \frac{\mu}{2} L^x\right) u^* = f_u, \tag{34}$$

$$A^y v^* = \left(\frac{\rho}{\Delta t} I - \frac{\mu}{2} L^y\right) v^* = f_v. \tag{35}$$

Generally, $-\mathbf{D}\cdot\mathbf{u}^* \neq f_p$. To construct a vector field $\mathbf{u}$ which does satisfy the specified divergence condition, we next project $\mathbf{u}^*$ onto the space of vector fields satisfying $-\mathbf{D}\cdot\mathbf{u} = f_p$ by solving

$$\rho \frac{\mathbf{u} - \mathbf{u}^*}{\Delta t} = -\mathbf{G}\varphi, \tag{36}$$

$$-\mathbf{D}\cdot\mathbf{u} = f_p \tag{37}$$

for $\mathbf{u}$ and φ. Taking the discrete divergence of Eq. (36) and using Eq. (37), we have that φ is the solution of the discrete Poisson problem

$$-\mathbf{D}\cdot\mathbf{G}\varphi = -L^c\varphi = -\frac{\rho}{\Delta t}(f_p + \mathbf{D}\cdot\mathbf{u}^*). \tag{38}$$

Note that the key difference between the projection method as a preconditioner and the projection method as a solver is that, in a typical projection solver, we wish to impose $-\mathbf{D}\cdot\mathbf{u} = 0$, whereas in the projection preconditioner, we generally wish to impose $-\mathbf{D}\cdot\mathbf{u} = f_p \neq 0$.

Finally, we obtain $\mathbf{u} = (u, v)$ and p by evaluating

$$u = u^* - \frac{\Delta t}{\rho} G^x \varphi, \tag{39}$$

$$v = v^* - \frac{\Delta t}{\rho} G^y \varphi, \tag{40}$$

$$p = \left(I - \frac{\Delta t}{\rho}\frac{\mu}{2} L^c\right)\varphi. \tag{41}$$

The projection preconditioner may be expressed in matrix form:

$$P_{\text{proj}} = \begin{pmatrix} I & -\frac{\Delta t}{\rho}\mathbf{G} \\ 0 & I - \frac{\Delta t}{\rho}\frac{\mu}{2} L^c \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & (-L^c)^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -\frac{\rho}{\Delta t}\mathbf{D}\cdot & -\frac{\rho}{\Delta t} I \end{pmatrix} \begin{pmatrix} \mathbf{A}^{-1} & 0 \\ 0 & I \end{pmatrix}. \tag{42}$$

In certain special cases, note that P_{proj} is actually an *exact* solver for Eq. (32) when exact solvers are used for the subdomain problems A^x, A^y, and $-L^c$. (By exact subdomain solvers, we mean either direct solvers, which solve the linear systems to within roundoff error, or iterative solvers which employ stringent convergence tolerances.) For instance, in the case of periodic boundary conditions, the P_{proj}-preconditioned FGMRES solver for Eq. (31) is guaranteed to converge in a single step when exact subdomain solvers are employed. In practice, we do not typically employ exact subdomain solvers. Instead, we typically employ inexact subdomain solvers which terminate after only a few iterations; see Section 4.3.

4.2. An approximate Schur complement preconditioner

Following [39], an alternative approach to preconditioning Eq. (31) may be obtained by considering the block LU-factorization

$$\begin{pmatrix} A^x & 0 & G^x \\ 0 & A^y & G^y \\ -D^x & -D^y & 0 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{G} \\ -\mathbf{D}\cdot & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ -\mathbf{D}\cdot\mathbf{A}^{-1} & I \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{G} \\ 0 & -S \end{pmatrix}, \tag{43}$$

where

$$\mathbf{A} = \begin{pmatrix} A^x & 0 \\ 0 & A^y \end{pmatrix} = \begin{pmatrix} \frac{\rho}{\Delta t} I - \frac{\mu}{2} L^x & 0 \\ 0 & \frac{\rho}{\Delta t} I - \frac{\mu}{2} L^y \end{pmatrix} \tag{44}$$

and $S = -\mathbf{D} \cdot \mathbf{A}^{-1} \mathbf{G}$ is the Schur complement.

The *exact* Schur complement preconditioner for Eq. (31) is the matrix

$$P_{\text{Schur}} = \begin{pmatrix} \mathbf{A} & \mathbf{G} \\ 0 & -S \end{pmatrix}^{-1}. \tag{45}$$

Note that

$$\begin{pmatrix} \mathbf{A} & \mathbf{G} \\ -\mathbf{D} \cdot & 0 \end{pmatrix} P_{\text{Schur}} = \begin{pmatrix} \mathbf{A} & \mathbf{G} \\ -\mathbf{D} \cdot & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{G} \\ 0 & -S \end{pmatrix}^{-1} = \begin{pmatrix} I & 0 \\ -\mathbf{D} \cdot \mathbf{A}^{-1} & I \end{pmatrix}. \tag{46}$$

Using this property, it can be shown [52,39] that if P_{Schur} were used as a right preconditioner for FGMRES applied to Eq. (31), then the preconditioned solver would always converge within two steps.

We wish to construct a preconditioner based on P_{Schur} which can be constructed out of subdomain solvers for A^x, A^y, and $-L^c$. We begin by computing the block factorization of P_{Schur},

$$P_{\text{Schur}} = \begin{pmatrix} \mathbf{A} & \mathbf{G} \\ 0 & -S \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -\mathbf{G} \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -S^{-1} \end{pmatrix}. \tag{47}$$

Thus, the implementation of the exact Schur complement preconditioner requires the application of $-S^{-1} = (\mathbf{D} \cdot \mathbf{A}^{-1} \mathbf{G})^{-1}$, which is generally too expensive an operation to result in an efficient preconditioner.

To obtain an efficient preconditioner, we replace S with a matrix $\widehat{S} \approx S$,

$$\widehat{S} = -\mathbf{D} \cdot \mathbf{G} \left(\left(\frac{\rho}{\Delta t} I - \frac{\mu}{2} L^c \right)^{-1} \right). \tag{48}$$

Letting $A^c = (\frac{\rho}{\Delta t} I - \frac{\mu}{2} L^c)$, we have that

$$\widehat{S} = -L^c (A^c)^{-1}. \tag{49}$$

Note that $\widehat{S}$ is obtained by assuming that $\mathbf{G} A^c \approx \mathbf{A}\mathbf{G}$, so that $\mathbf{G} \approx \mathbf{A}\mathbf{G}(A^c)^{-1}$, and thus

$$S = -\mathbf{D} \cdot \mathbf{A}^{-1} \mathbf{G} \approx -\mathbf{D} \cdot \mathbf{A}^{-1} (\mathbf{A}\mathbf{G}(A^c)^{-1}) = -\mathbf{D} \cdot \mathbf{G}(A^c)^{-1} = \widehat{S}. \tag{50}$$

Note that $S = \widehat{S}$ only in special situations in which $\mathbf{G} A^c = \mathbf{A}\mathbf{G}$, such as in the case of periodic boundary conditions. In general, $S \neq \widehat{S}$.

Using $\widehat{S}$ in place of S in P_{Schur} yields the *approximate* Schur complement preconditioner $\widehat{P}_{\text{Schur}}$:

$$\widehat{P}_{\text{Schur}} = \begin{pmatrix} \mathbf{A} & \mathbf{G} \\ 0 & -\widehat{S} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -\mathbf{G} \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -\widehat{S}^{-1} \end{pmatrix} \tag{51}$$

$$= \begin{pmatrix} \mathbf{A}^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -\mathbf{G} \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & A^c (L^c)^{-1} \end{pmatrix}. \tag{52}$$

Note that the implementation of the approximate Schur complement preconditioner requires the same subdomain solvers as the projection preconditioner P_{proj}. In cases in which $S = \widehat{S}$, such as when periodic boundary conditions are employed, note that the $\widehat{P}_{\text{Schur}}$-preconditioned FGMRES solver for Eq. (31) is guaranteed to converge in at most two steps when exact subdomain solvers are employed for A^x, A^y, and $-L^c$. As with the projection method-based preconditioner P_{proj}, we typically do not employ exact subdomain solvers, but instead employ approximate subdomain solvers; see Section 4.3.

4.3. Subdomain solvers

The preconditioners described in Sections 4.1 and 4.2 require linear solvers for the velocity subdomain systems $A^x = (\frac{\rho}{\Delta t} I - \frac{\mu}{2} L^x)$ and $A^y = (\frac{\rho}{\Delta t} I - \frac{\mu}{2} L^y)$ and for the pressure subdomain system $-L^c = -\mathbf{D} \cdot \mathbf{G}$. For the velocity subdomain systems A^x and A^y, we employ the conjugate gradient (CG) method preconditioned either by the Jacobi algorithm or by geometric multigrid, and for the pressure subdomain system $-L^c$, we employ CG preconditioned by geometric multigrid. The particular multigrid schemes we use are the SMG [53–55] and PFMG [56,55] algorithms implemented in the *hypre* software package [57,58]. In the present work, we always use a loose relative residual tolerance $\epsilon_{\text{rel}} = 0.01$ for the subdomain solvers, i.e., the subdomain solvers are said to have "converged" when the initial residual is reduced by a factor of 100. We have found that it is generally more efficient to use loose convergence thresholds such as $\epsilon_{\text{rel}} = 0.01$ (or even $\epsilon_{\text{rel}} = 0.1$) for these "inner" solvers than it is to use tight convergence thresholds. The reason for this appears to be that reaching the rather stringent convergence threshold required of the outer FGMRES solver (a relative threshold of 1.0e−10 in the present work)

usually requires multiple FGMRES iterations, even when very tight sub-solver convergence tolerances are employed, and so "over-solving" the subdomain systems reduces the efficiency of the overall algorithm. The determination of "optimal" convergence thresholds for the subdomain solvers is left as future work.

Homogeneous physical boundary conditions are required for the subdomain solvers; however, note that the choices for these "artificial" boundary conditions do not affect the accuracy of the scheme. Instead, they impact only the rate of convergence of the linear solver. Artificial boundary conditions which are "incompatible" with the true physical boundary conditions can yield a non-convergent solver, however. At boundaries where normal velocities are prescribed, we employ homogeneous Dirichlet boundary conditions for the normal components in the velocity subdomain solvers and homogeneous Neumann boundary conditions in the pressure subdomain solver. At boundaries where normal tractions are prescribed, we employ homogeneous Neumann boundary conditions for the normal components in the velocity subdomain solvers and homogeneous Dirichlet boundary conditions for the pressure subdomain solver. At boundaries where tangential velocities are prescribed, we employ homogeneous Dirichlet boundary conditions for the tangential components in the velocity subdomain solvers, and at boundaries where tangential tractions are prescribed, we employ homogeneous Neumann boundary conditions for the tangential components. Standard second-order implementations of these boundary conditions are employed.

5. Numerical results

We now present numerical examples which demonstrate the accuracy of the discretization, and which compare the efficiency of the projection method-based preconditioner to the approximate Schur complement preconditioner. Errors and

Table 1
Error in $\mathbf{u}$ at time $t = 0.5$ for the forced flow problem of Section 5.1 with $\mu = 1.0$ and with various choices of boundary conditions.

N	vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^1 error in $\mathbf{u}$ at time 0.5								
32	3.06e−03	2.01	3.82e−03	2.00	3.19e−03	2.01	3.95e−03	2.00
64	7.63e−04	2.00	9.54e−04	2.00	7.94e−04	2.00	9.88e−04	2.00
128	1.91e−04	2.00	2.39e−04	2.00	1.98e−04	2.00	2.47e−04	2.00
256	4.77e−05	2.00	5.96e−05	2.00	4.96e−05	2.00	6.17e−05	2.00
512	1.19e−05	–	1.49e−05	–	1.24e−05	–	1.54e−05	–
L^∞ error in $\mathbf{u}$ at time 0.5								
32	5.28e−03	2.01	6.37e−03	2.00	5.47e−03	1.99	6.51e−03	1.99
64	1.31e−03	2.00	1.60e−03	2.00	1.38e−03	1.99	1.64e−03	1.99
128	3.28e−04	2.00	3.99e−04	2.00	3.47e−04	2.00	4.12e−04	2.00
256	8.20e−05	2.00	9.98e−05	2.00	8.69e−05	2.00	1.03e−04	2.00
512	2.05e−05	–	2.49e−05	–	2.18e−05	–	2.58e−05	–

Table 2
Error in $\mathbf{u}$ at time $t = 0.5$ for the forced flow problem of Section 5.1 with $\mu = 0.1$ and with various choices of boundary conditions.

N	vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in $\mathbf{u}$ at time 0.5								
32	2.41e−03	1.98	2.86e−03	1.99	2.57e−03	1.95	3.78e−03	1.99
64	6.11e−04	2.00	7.18e−04	2.00	6.64e−04	1.98	9.55e−04	1.99
128	1.53e−04	2.00	1.80e−04	2.00	1.68e−04	1.99	2.40e−04	2.00
256	3.83e−05	2.00	4.49e−05	2.00	4.23e−05	2.00	6.00e−05	2.00
512	9.57e−06	–	1.12e−05	–	1.06e−05	–	1.50e−05	–

Table 3
Error in $\mathbf{u}$ at time $t = 0.5$ for the forced flow problem of Section 5.1 with $\mu = 0.01$ and with various choices of boundary conditions.

N	vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in $\mathbf{u}$ at time 0.5								
32	3.06e−03	1.92	3.17e−03	1.98	3.22e−03	1.97	3.37e−03	1.96
64	8.08e−04	2.00	8.04e−04	2.00	8.25e−04	1.99	8.67e−04	1.99
128	2.02e−04	2.00	2.01e−04	2.00	2.08e−04	1.99	2.18e−04	2.00
256	5.07e−05	2.00	5.02e−05	2.00	5.23e−05	2.00	5.45e−05	2.00
512	1.27e−05	–	1.25e−05	–	1.31e−05	–	1.36e−05	–

convergence rates are reported in discretized L^1, L^2, and L^∞ norms. When analytic solutions are available, errors and convergence rates are reported with respect to those exact solutions. When analytic solutions are not available, errors are estimated for the purpose of computing convergence rates by computing the norm of the difference of solutions obtained on an $N \times N$ grid to those obtained on a $2N \times 2N$ grid. The discrete L^1, L^2, and L^∞ norms of the cell-centered pressure p are computed using standard formulae [22]. Similar expressions are used when computing the discrete norms for the staggered-grid velocity $\mathbf{u} = (u, v)$, except that simple modifications to the definitions of the discrete L^1 and L^2 norms are required at domain boundaries to ensure that $\|(1,0)\| = \|(0,1)\| = 1$. For instance, $\|\mathbf{u}\|_1 = \|(u,v)\|_1 = \|u\|_1 + \|v\|_1$, with

Table 4
Error in $\mathbf{u}$ at time $t = 0.5$ for the forced flow problem of Section 5.1 with $\mu = 0.001$ and with various choices of boundary conditions.

N	vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in $\mathbf{u}$ at time 0.5								
32	3.19e−03	1.88	3.49e−03	1.97	3.45e−03	1.97	3.90e−03	1.95
64	8.68e−04	1.88	8.93e−04	1.99	8.81e−04	1.99	1.01e−03	1.99
128	2.36e−04	1.98	2.24e−04	2.00	2.22e−04	1.99	2.55e−04	2.00
256	5.99e−05	2.00	5.59e−05	2.01	5.58e−05	2.00	6.35e−05	2.00
512	1.50e−05	–	1.39e−05	–	1.40e−05	–	1.58e−05	–

Table 5
Error in p at time $t = 0.5$ for the forced flow problem of Section 5.1 with $\mu = 1.0$ and with various choices of boundary conditions.

N	vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^1 error in p at time 0.5								
32	1.14e−02	1.98	5.18e−03	1.98	1.09e−02	1.95	5.73e−03	2.00
64	2.90e−03	1.99	1.31e−03	1.99	2.81e−03	1.98	1.43e−03	2.00
128	7.29e−04	1.99	3.30e−04	2.00	7.12e−04	1.99	3.58e−04	2.00
256	1.83e−04	2.00	8.27e−05	2.00	1.79e−04	2.00	8.94e−05	2.00
512	4.58e−05	–	2.07e−05	–	4.49e−05	–	2.23e−05	–
L^∞ error in p at time 0.5								
32	9.47e−02	1.90	2.60e−02	1.90	8.70e−02	1.87	3.36e−02	1.90
64	2.54e−02	1.95	6.99e−03	1.95	2.38e−02	1.94	8.98e−03	1.95
128	6.58e−03	1.98	1.81e−03	1.97	6.18e−03	1.97	2.32e−03	1.98
256	1.67e−03	1.99	4.61e−04	1.99	1.58e−03	1.99	5.88e−04	1.99
512	4.22e−04	–	1.16e−04	–	3.98e−04	–	1.48e−04	–

Table 6
Error in p at time $t = 0.5$ for the forced flow problem of Section 5.1 with $\mu = 0.1$ and with various choices of boundary conditions.

N	vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in p at time 0.5								
32	1.05e−02	1.91	5.06e−03	1.94	9.05e−03	1.91	6.13e−03	1.93
64	2.79e−03	1.96	1.32e−03	1.97	2.42e−03	1.96	1.60e−03	1.97
128	7.18e−04	1.98	3.37e−04	1.98	6.23e−04	1.98	4.09e−04	1.98
256	1.82e−04	1.99	8.51e−05	1.99	1.58e−04	1.99	1.03e−04	1.99
512	4.59e−05	–	2.14e−05	–	3.98e−05	–	2.60e−05	–

Table 7
Error in p at time $t = 0.5$ for the forced flow problem of Section 5.1 with $\mu = 0.01$ and with various choices of boundary conditions.

N	vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in p at time 0.5								
32	3.93e−03	1.96	2.85e−03	1.92	2.40e−03	1.99	2.53e−03	1.98
64	1.01e−03	1.97	7.56e−04	1.96	6.07e−04	1.99	6.41e−04	1.99
128	2.59e−04	1.98	1.94e−04	1.98	1.53e−04	2.00	1.61e−04	1.99
256	6.55e−05	1.99	4.94e−05	1.99	3.83e−05	2.00	4.05e−05	2.00
512	1.65e−05	–	1.24e−05	–	9.60e−06	–	1.01e−05	–

$$\|u\|_1 = \frac{1}{2}\sum_{j=0}^{N-1}|u_{-\frac{1}{2},j}|h^2 + \sum_{i=1}^{N-1}\sum_{j=0}^{N-1}|u_{i-\frac{1}{2},j}|h^2 + \frac{1}{2}\sum_{j=0}^{N-1}|u_{N-\frac{1}{2},j}|h^2, \tag{53}$$

$$\|v\|_1 = \frac{1}{2}\sum_{i=0}^{N-1}|v_{i,-\frac{1}{2}}|h^2 + \sum_{i=0}^{N-1}\sum_{j=1}^{N-1}|v_{i,j-\frac{1}{2}}|h^2 + \frac{1}{2}\sum_{i=0}^{N-1}|v_{i,N-\frac{1}{2}}|h^2. \tag{54}$$

Convergence rates are estimated using standard methods.

For the analytic flows of Sections 5.1 and 5.2, the exact solutions for the velocity and pressure are used to compute explicit formulae for the various choices of boundary conditions. Although it is straightforward to increase the Reynolds numbers used in these test cases by modifying the physical viscosity, these analytic solutions are largely (but not completely) independent of μ. In particular, even for $\mu \ll 1$, the analytic solutions employed in Sections 5.1 and 5.2 do not exhibit characteristics of high Re flows such as thin viscous boundary layers. Nonetheless, at least for the unforced problem of Section

Table 8
Error in p at time $t = 0.5$ for the forced flow problem of Section 5.1 with $\mu = 0.001$ and with various choices of boundary conditions.

N	vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in p at time 0.5								
32	2.84e−03	1.87	2.64e−03	1.91	2.51e−03	1.99	2.52e−03	1.99
64	7.76e−04	1.97	7.02e−04	1.95	6.31e−04	1.99	6.35e−04	1.99
128	1.99e−04	1.99	1.82e−04	1.98	1.59e−04	1.99	1.60e−04	1.99
256	5.00e−05	1.99	4.61e−05	1.99	3.99e−05	2.00	4.01e−05	2.00
512	1.26e−05	–	1.16e−05	–	1.00e−05	–	1.01e−05	–

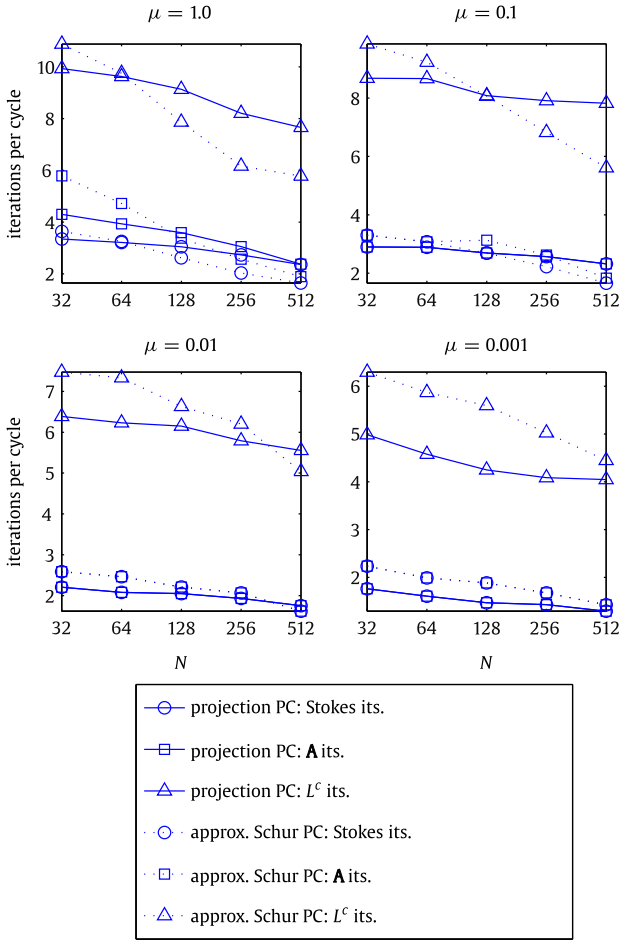

Fig. 3. Average number of linear solver iterations per incompressible Stokes solve for the projection-preconditioned and approximate Schur-preconditioned FGMRES solver for specified normal and tangential velocity boundary conditions. Note that the values reported for the numbers of iterations for the **A** and L^c solvers are the average numbers of sub-solver iterations per Stokes solve, not per FGMRES iteration. See further discussion in Section 5.1.

5.2, imposing physical boundary conditions accurately and stably at high Reynolds numbers is difficult, especially for the case of traction boundary conditions. To demonstrate the capability of the scheme for problems with characteristics of high Re flows (e.g., thin boundary layers), in Section 5.3 we also present steady solutions obtained by the method for the well-known lid-driven cavity flow, along with a regularized version [36] of this problem.

5.1. Forced flow

Our first example is taken from Brown et al. [18]. This problem employs the method of manufactured solutions, using a forcing term $\mathbf{f}(\mathbf{x},t)$ constructed so that the velocity and pressure satisfy

$$u(\mathbf{x},t) = \cos(2\pi(x-\omega(t)))(3y^2-2y), \tag{55}$$

$$v(\mathbf{x},t) = 2\pi\sin(2\pi(x-\omega(t)))y^2(y-1), \tag{56}$$

$$p(\mathbf{x},t) = -\frac{\omega'(t)}{2\pi}\sin(2\pi(x-\omega(t)))(\sin(2\pi y)-2\pi y+\pi) \tag{57}$$

$$-\mu\cos(2\pi(x-\omega(t)))(-2\sin(2\pi y)+2\pi y-\pi), \tag{58}$$

for $\rho = 1$ with

$$\omega(t) = 1+\sin(2\pi t^2). \tag{59}$$

In our tests, we impose periodic boundary conditions in the x-direction and various choices of physical boundary conditions in the y-direction. In particular, we consider: (1) specified normal and tangential velocities in the y-direction, abbreviated as "vel–vel"; (2) specified normal velocity with specified tangential traction in the y-direction, abbreviated as "vel–tra"; (3) specified normal traction with specified tangential velocity in the y-direction, abbreviated as "tra–vel"; and (4) specified normal and tangential tractions in the y-direction, abbreviated as "tra–tra". Errors in the velocity and pressure are computed at

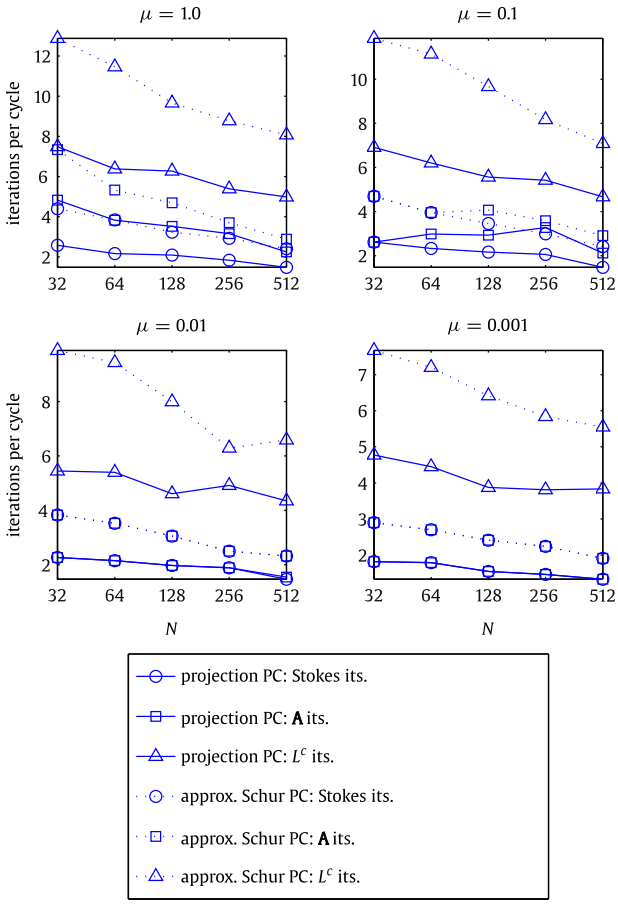

Fig. 4. Average number of linear solver iterations per incompressible Stokes solve for the projection-preconditioned and approximate Schur-preconditioned FGMRES solver for specified normal velocity and specified tangential traction boundary conditions.

time $t = 0.5$. Following Brown et al. [18], we use a uniform timestep $\Delta t = \frac{1}{2N}$, yielding a CFL number of 0.5. For these tests, we use a relative convergence tolerance of 1.0e−10 for the FGMRES algorithm and a relative convergence tolerance of 1.0e−2 for the subdomain solvers. The pressure subdomain solver uses CG preconditioned by PFMG, and the velocity subdomain solver uses CG preconditioned by SMG.

For the present example, we consider $\mu = 1.0, 0.1, 0.01$, and 0.001, corresponding to $Re = 1$, 10, 100, and 1000, respectively, for grid spacings $N = 32, 64, 128, 256$, and 512. Accuracy results are summarized in Tables 1–8, with both L^1 and L^∞ results presented for $Re = 1$, and only L^∞ results presented for all other cases. Clear second-order convergence rates are observed in all cases once the grid spacing is sufficiently fine. Moreover, the accuracy of the solver is approximately the same (within about a factor of two) for all choices of physical boundary conditions. With larger viscosities (i.e., at lower Reynolds numbers), imposing tangential traction boundary conditions ("vel–tra" and "tra–tra") results in *lower* accuracy for the velocity and *higher* accuracy for the pressure when compared to the results obtained by imposing tangential velocity boundary conditions ("vel–vel" and "tra–vel"). With smaller viscosities (i.e., at higher Reynolds numbers), these discrepancies are diminished and ultimately eliminated. These differences are less marked in the discrete L^1 norm, especially for the velocity (data not shown). With smaller viscosities (i.e., at higher Reynolds numbers), the pointwise convergence rates at coarser resolutions are generally somewhat less than two, although full second-order convergence rates are ultimately observed once the computational grid is fine enough. For the range of grid spacings considered, fully second-order convergence rates are observed in the L^1 norm for both the velocity and the pressure (data not shown).

Linear solver iteration results are summarized in Figs. 3–6. Note that these figures report the average numbers of solver iterations per incompressible Stokes solve. (Recall that with $n_{\text{cycles}} = 3$, three Stokes solves are performed per timestep.) In particular, for the **A** and L^c subdomain solvers, we report the average *total* number of solver iterations per Stokes solve, *not* the average number of iterations per FGMRES iteration. The performance results indicate that both preconditioners yield a scalable algorithm over a broad range of Reynolds numbers for all choices of physical boundary conditions. In particular, these results indicate that the outer FGMRES and inner multigrid-preconditioned CG iteration counts are largely independent of N and of Re. In most cases, the projection preconditioner outperforms the approximate Schur complement preconditioner. The only cases in which the Schur complement preconditioner outperforms the projection preconditioner are at low

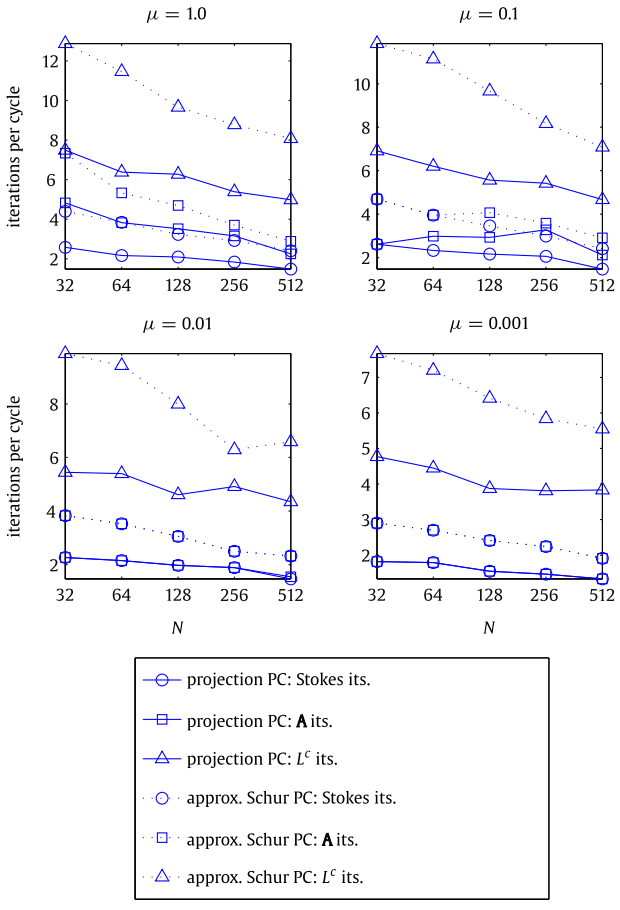

Fig. 5. Average number of linear solver iterations per incompressible Stokes solve for the projection-preconditioned and approximate Schur-preconditioned FGMRES solver for specified normal traction and specified tangential velocity boundary conditions.

Reynolds numbers for specified normal and tangential velocity ("vel–vel") boundary conditions. Note that the velocity subdomain solver generally requires only one or two iterations to reach the specified convergence threshold when SMG is used as a preconditioner.

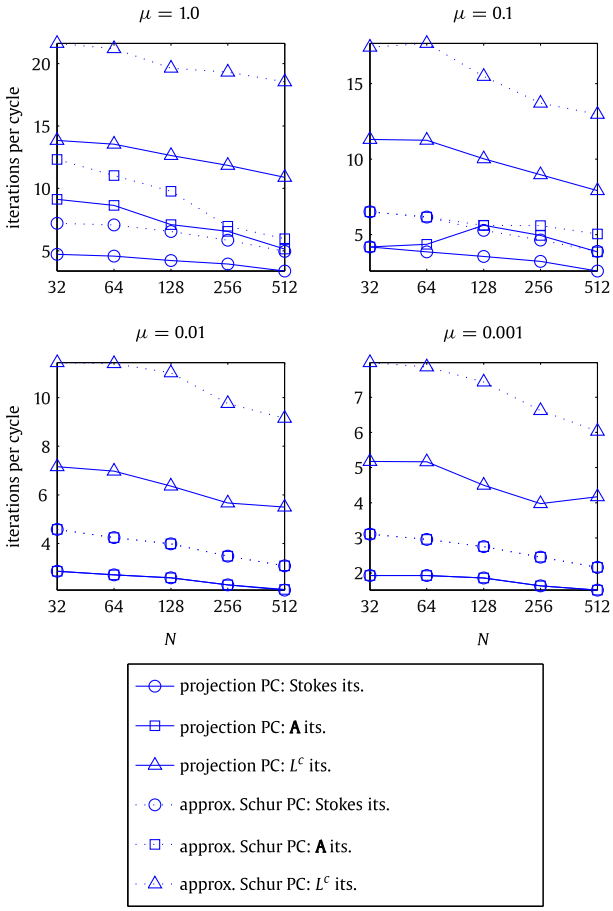

Fig. 6. Average number of linear solver iterations per incompressible Stokes solve for the projection-preconditioned and approximate Schur-preconditioned FGMRES solver for specified normal and tangential traction boundary conditions.

Table 9
Error in **u** at time $t = 0.5$ for the Taylor vortices problem of Section 5.2 with $\mu = 0.1$ and with various choices of boundary conditions.

N	Periodic		vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^1 error in **u** *at time 0.5*										
32	5.42e−04	1.99	4.01e−04	2.07	9.42e−04	1.96	1.08e−03	1.93	1.28e−03	2.01
64	1.37e−04	1.99	9.54e−05	2.03	2.41e−04	1.99	2.84e−04	1.97	3.19e−04	2.00
128	3.44e−05	1.99	2.34e−05	2.01	6.08e−05	2.00	7.25e−05	1.99	7.97e−05	2.00
256	8.63e−06	2.00	5.80e−06	2.01	1.52e−05	2.00	1.83e−05	1.99	1.99e−05	2.00
512	2.16e−06	2.00	1.44e−06	2.03	3.82e−06	2.00	4.59e−06	2.00	4.97e−06	2.00
1024	5.41e−07	–	3.55e−07	–	9.54e−07	–	1.15e−06	–	1.24e−06	–
L^∞ error in **u** *at time 0.5*										
32	6.13e−04	1.99	7.44e−04	2.07	1.64e−03	1.96	1.98e−03	1.90	2.40e−03	2.01
64	1.55e−04	1.99	1.77e−04	2.03	4.21e−04	1.99	5.30e−04	1.96	5.97e−04	2.00
128	3.89e−05	2.00	4.33e−05	2.01	1.06e−04	2.00	1.37e−04	1.98	1.49e−04	2.00
256	9.74e−06	2.00	1.07e−05	2.01	2.66e−05	2.00	3.45e−05	1.99	3.73e−05	2.00
512	2.44e−06	2.00	2.67e−06	2.02	6.65e−06	2.00	8.68e−06	2.00	9.30e−06	2.00
1024	6.10e−07	–	6.58e−07	–	1.66e−06	–	2.17e−06	–	2.32e−06	–

5.2. Unforced flow

For our second example, we consider the well-known Taylor vortices, an unforced analytic solution to the two-dimensional incompressible Navier–Stokes equations on the unit square given by

$$u(\mathbf{x},t) = 1 - 2\exp(-8\pi^2\mu t)\cos(2\pi(x-t))\sin(2\pi(y-t)), \tag{60}$$

$$v(\mathbf{x},t) = 1 + 2\exp(-8\pi^2\mu t)\sin(2\pi(x-t))\cos(2\pi(y-t)), \tag{61}$$

$$p(\mathbf{x},t) = -\exp(-16\pi^2\mu t)(\cos(4\pi(x-t)) + \cos(4\pi(y-t))), \tag{62}$$

for $\rho = 1$.

In our tests, we impose periodic boundary conditions in the x-direction and various choices of physical boundary conditions in the y-direction. In particular, we consider periodic boundary conditions in all directions along with the four combinations of physical boundary conditions considered in the previous subsection ("vel–vel," "vel–tra," "tra–vel," and "tra–tra"). Errors in the velocity and pressure are computed at time $t = 0.5$. We use a uniform timestep $\Delta t = \frac{1}{4N}$, yielding a CFL number of approximately 0.75. For these tests, we use a relative convergence tolerance of 1.0e−10 for the FGMRES algorithm and a relative convergence tolerance of 1.0e−2 for the subdomain solvers. The pressure subdomain solver uses CG preconditioned by PFMG, and the velocity solver uses CG preconditioned by point Jacobi.

Accuracy results are summarized in Tables 9–16, which present results for $\mu = 0.1, 0.01, 0.001$, and 0.0001, corresponding to $Re \approx 30, 300, 3000$, and 30000, respectively, for grid spacings $N = 32, 64, 128, 256, 512$, and 1024. Both L^1 and L^∞ results are presented for $Re \approx 30$, and only L^∞ results are presented for all other cases. Second-order convergence rates are observed for most cases once the spatial grid is sufficiently fine. At higher Reynolds numbers, fine grids are required to obtain accurate

Table 10
Error in **u** at time $t = 0.5$ for the Taylor vortices problem of Section 5.2 with $\mu = 0.01$ and with various choices of boundary conditions.

N	Periodic		vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in **u** *at time 0.5*										
32	4.36e−03	1.97	3.94e−02	2.79	2.49e−01	1.91	8.74e−02	3.05	4.68e−01	1.42
64	1.11e−03	1.99	5.71e−03	2.83	6.62e−02	2.24	1.06e−02	3.03	1.75e−01	2.13
128	2.80e−04	1.99	8.04e−04	2.46	1.40e−02	2.22	1.29e−03	2.74	3.99e−02	2.17
256	7.04e−05	2.00	1.46e−04	2.20	3.01e−03	2.10	1.93e−04	2.41	8.84e−03	2.11
512	1.77e−05	2.00	3.18e−05	2.09	7.01e−04	2.06	3.62e−05	2.01	2.05e−03	2.06
1024	4.42e−06	–	7.47e−06	–	1.68e−04	–	8.98e−06	–	4.92e−04	–

Table 11
Error in **u** at time $t = 0.5$ for the Taylor vortices problem of Section 5.2 with $\mu = 0.001$ and with various choices of boundary conditions.

N	Periodic		vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in **u** *at time 0.5*										
32	4.96e−03	2.00	5.64e−01	2.33	9.06e−01	0.38	8.80e−01	1.85	1.56e+00	0.25
64	1.24e−03	2.00	1.12e−01	2.88	6.94e−01	1.24	2.45e−01	2.70	1.31e+00	0.82
128	3.11e−04	2.00	1.52e−02	3.11	2.93e−01	1.95	3.76e−02	3.15	7.43e−01	2.52
256	7.77e−05	2.00	1.77e−03	3.17	7.56e−02	1.97	4.23e−03	3.16	1.30e−01	1.66
512	1.94e−05	2.00	1.97e−04	3.13	1.93e−02	2.15	4.73e−04	3.08	4.11e−02	2.15
1024	4.86e−06	–	2.24e−05	–	4.35e−03	–	5.58e−05	–	9.27e−03	–

Table 12
Error in **u** at time $t = 0.5$ for the Taylor vortices problem of Section 5.2 with $\mu = 0.0001$ and with various choices of boundary conditions.

N	Periodic		vel–vel		vel–tra		tra–vel	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in **u** *at time 0.5*								
32	5.04e−03	2.00	9.93e−01	0.49	1.04e+00	−0.02	1.72e+00	−0.06
64	1.26e−03	2.00	7.09e−01	2.14	1.06e+00	0.15	1.79e+00	2.67
128	3.15e−04	2.00	1.61e−01	2.89	9.52e−01	0.64	2.81e−01	2.25
256	7.87e−05	2.00	2.17e−02	3.03	6.13e−01	1.58	5.91e−02	3.07
512	1.97e−05	2.00	2.67e−03	3.05	2.06e−01	1.78	7.05e−03	3.07
1024	4.92e−06	–	3.21e−04	–	5.98e−02	–	8.40e−04	–

Table 13
Error in p at time $t = 0.5$ for the Taylor vortices problem of Section 5.2 with $\mu = 0.1$ and with various choices of boundary conditions.

N	Periodic		vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^1 error in p at time 0.5										
32	1.12e−05	2.04	6.52e−05	2.02	1.62e−04	2.13	5.50e−05	2.05	2.03e−04	2.08
64	2.72e−06	2.02	1.61e−05	2.00	3.69e−05	2.06	1.33e−05	2.03	4.80e−05	2.04
128	6.72e−07	2.01	4.02e−06	2.00	8.83e−06	2.03	3.26e−06	2.02	1.17e−05	2.02
256	1.67e−07	2.00	1.01e−06	2.00	2.16e−06	2.02	8.05e−07	2.01	2.88e−06	2.01
512	4.17e−08	2.00	2.52e−07	2.05	5.34e−07	2.01	2.00e−07	2.01	7.14e−07	2.01
1024	1.04e−08	–	6.07e−08	–	1.33e−07	–	4.98e−08	–	1.78e−07	–
L^∞ error in p at time 0.5										
32	2.74e−05	2.03	5.58e−04	1.89	1.08e−03	1.97	2.75e−04	2.12	1.20e−03	1.99
64	6.70e−06	2.01	1.51e−04	1.94	2.75e−04	1.99	6.35e−05	1.90	3.03e−04	1.97
128	1.66e−06	2.01	3.91e−05	1.97	6.93e−05	2.00	1.70e−05	1.96	7.74e−05	1.99
256	4.13e−07	2.00	9.96e−06	1.99	1.74e−05	2.00	4.39e−06	1.98	1.95e−05	1.99
512	1.03e−07	2.00	2.52e−06	2.05	4.35e−06	2.00	1.11e−06	1.99	4.90e−06	2.00
1024	2.57e−08	–	6.09e−07	–	1.09e−06	–	2.80e−07	–	1.23e−06	–

Table 14
Error in p at time $t = 0.5$ for the Taylor vortices problem of Section 5.2 with $\mu = 0.01$ and with various choices of boundary conditions.

N	Periodic		vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in p at time 0.5										
32	6.25e−03	1.93	4.42e−02	2.64	2.65e−01	1.84	7.61e−02	3.16	5.08e−01	1.86
64	1.64e−03	1.97	7.10e−03	2.51	7.39e−02	2.14	8.53e−03	2.99	1.40e−01	2.24
128	4.18e−04	1.98	1.24e−03	2.09	1.68e−02	2.17	1.08e−03	1.79	2.96e−02	2.21
256	1.06e−04	1.99	2.92e−04	2.02	3.74e−03	2.12	3.11e−04	1.90	6.41e−03	2.13
512	2.66e−05	2.00	7.20e−05	2.01	8.62e−04	2.07	8.35e−05	1.96	1.47e−03	2.07
1024	6.66e−06	–	1.79e−05	–	2.06e−04	–	2.14e−05	–	3.49e−04	–

Table 15
Error in p at time $t = 0.5$ for the Taylor vortices problem of Section 5.2 with $\mu = 0.001$ and with various choices of boundary conditions.

N	Periodic		vel–vel		vel–tra		tra–vel		tra–tra	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in p at time 0.5										
32	9.47e−03	1.94	6.47e−01	2.24	1.00e+00	0.28	1.29e+00	2.45	2.07e+00	0.54
64	2.47e−03	1.97	1.37e−01	2.84	8.25e−01	1.21	2.36e−01	2.90	1.43e+00	1.22
128	6.31e−04	1.98	1.91e−02	3.05	3.56e−01	1.86	3.16e−02	3.10	6.13e−01	2.07
256	1.60e−04	1.99	2.31e−03	3.09	9.83e−02	2.11	3.69e−03	3.16	1.46e−01	2.15
512	4.01e−05	2.00	2.71e−04	2.83	2.28e−02	2.19	4.13e−04	3.17	3.28e−02	2.20
1024	1.01e−05	–	3.82e−05	–	5.00e−03	–	4.58e−05	–	7.12e−03	–

Table 16
Error in p at time $t = 0.5$ for the Taylor vortices problem of Section 5.2 with $\mu = 0.0001$ and with various choices of boundary conditions.

N	Periodic		vel–vel		vel–tra		tra–vel	
	Error	Rate	Error	Rate	Error	Rate	Error	Rate
L^∞ error in p at time 0.5								
32	9.97e−03	1.94	1.12e+00	0.44	1.17e+00	−0.07	2.33e+00	0.58
64	2.60e−03	1.97	8.24e−01	2.08	1.23e+00	0.09	1.56e+00	2.35
128	6.62e−04	1.99	1.95e−01	2.74	1.16e+00	0.59	3.05e−01	2.73
256	1.67e−04	1.99	2.92e−02	3.00	7.69e−01	1.50	4.60e−02	3.03
512	4.20e−05	2.00	3.65e−03	3.04	2.73e−01	1.91	5.64e−03	3.05
1024	1.05e−05	–	4.43e−04	–	7.25e−02	–	6.82e−04	–

solutions, especially when tangential traction boundary conditions are prescribed. Additionally, we have found that the scheme becomes unstable for $Re \approx 30000$ when traction boundary conditions are used in both the normal and tangential directions. We experimented with alternative boundary discretizations in an attempt to overcome these instabilities without success. The instability was not alleviated either by decreasing the timestep size or by increasing the spatial resolution. In some cases, large estimated convergence rates (i.e., rates significantly larger than 2.0) are observed. Such rates are obtained when the solution is under-resolved and are not indicative of "super-convergence" of the algorithm. In most cases, convergence rates which are approximately 2.0 are obtained once the flow is well resolved by the scheme. The results which we obtain at the smallest viscosities demonstrate the difficulty of simulating high Reynolds number flow — even when the flow is smooth!

Since this flow is periodic in both the x- and y-directions, we are able to compare the accuracy obtained by the scheme in the absence of boundaries (i.e., using purely periodic boundary conditions) to that obtained with various choices of physical boundary conditions. In all cases, we find that imposing physical boundary conditions lowers the accuracy of the computed solution compared with the solution obtained by imposing periodic boundary conditions in all directions. The largest errors are generally incurred when tangential traction boundary conditions are specified. In particular, as was seen in the forced test case of Section 5.1, imposing tangential traction boundary conditions ("vel–tra" and "tra–tra") usually results in lower accuracy for the velocity when compared to the solutions obtained by imposing tangential velocity boundary conditions ("vel–vel" and "vel–tra"). Unlike the results obtained for the forced-flow problem, however, these discrepancies are not diminished at higher Reynolds numbers. Moreover, unlike the forced flow test case of Section 5.1, the approximation to the pressure is also less accurate for the "vel–tra" and "tra–tra" boundary conditions.

Linear solver iteration results are summarized in Figs. 7–12 for $N = 32, 64, 128, 256$, and 512. Note that these figures report the average number of solver iterations per incompressible Stokes solve. (Recall that with $n_{\text{cycles}} = 3$, three Stokes solves are performed per timestep.) In particular, for the **A** and L^c subdomain solvers, we report the average total number of

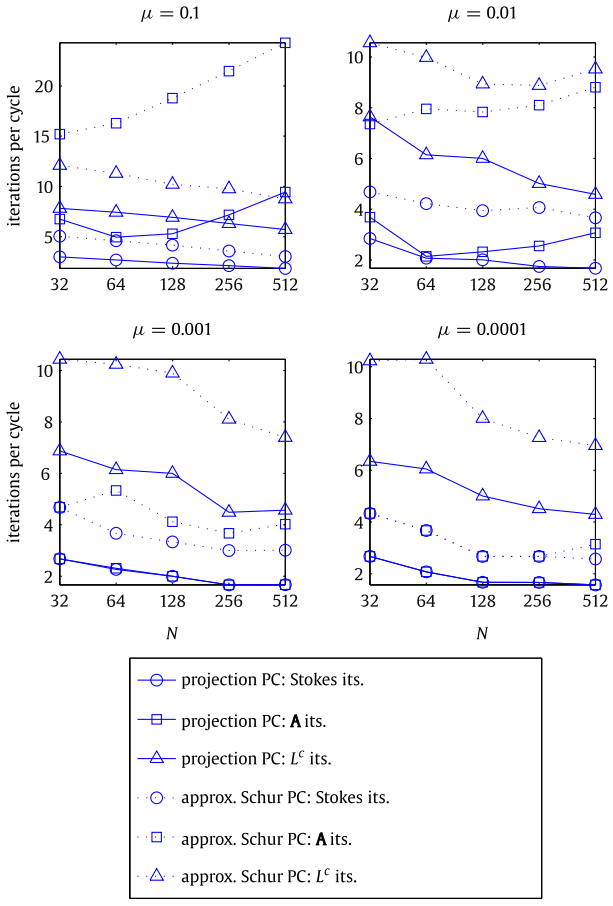

Fig. 7. Average number of linear solver iterations per incompressible Stokes solve for the projection-preconditioned and approximate Schur-preconditioned FGMRES solver for periodic conditions. Note that as in Figs. 3–6, the values reported for the numbers of iterations for the **A** and L^c solvers are the average numbers of sub-solver iterations per Stokes solve, not per FGMRES iteration. See further discussion in Section 5.2.

iterations per Stokes solve, *not* the average number of iterations per FGMRES iteration. The performance results indicate that both preconditioners result in a generally scalable algorithm, especially at higher Reynolds numbers. At lower Reynolds numbers, the velocity subdomain solver loses scalability because it employs the Jacobi algorithm as a preconditioner; see further discussion below. The projection preconditioner outperforms the approximate Schur complement preconditioner in all cases.

When Jacobi is used as a preconditioner for the velocity subdomain solver, that subdomain solver is in principle not scalable as $N \to \infty$; however, in our tests, this lack of scalability is manifest *only* at low Reynolds numbers. As demonstrated in Section 5.1, this lack of scalability can be alleviated by employing a multigrid preconditioner for the velocity subproblem. At higher Reynolds numbers, and for the range of grid spacings considered, only one to two Jacobi-preconditioned CG iterations are typically required to reach the specified relative residual tolerance $\epsilon_{\text{rel}} = 0.01$. In particular, in these cases, the algorithm is scalable in practice even without full multigrid preconditioning. Presumably, for any fixed Reynolds number, the performance of the Jacobi-preconditioned subdomain solver will eventually degrade once N is sufficiently large. The present results indicate, however, that for practical grid spacings, Jacobi may suffice as a preconditioner for moderate-to-high *Re* applications.

5.3. Lid-driven cavity flow

Our final example is the well-known lid-driven cavity flow. For this problem, we impose normal and tangential velocity boundary conditions at each boundary of the physical domain. At each domain boundary except for the upper boundary, we set $(u, v) = (0, 0)$, so that there is no penetration and no slip. At the upper boundary, where $y = 1$, we set $(u, v) = (1, 0)$, so that there is prescribed non-zero slip but no penetration. The initial velocity is set to zero, i.e., we employ a so-called impulsive start. To compute steady solutions, we run the unsteady solver at a CFL number of 0.95 until the flow reaches steady state.

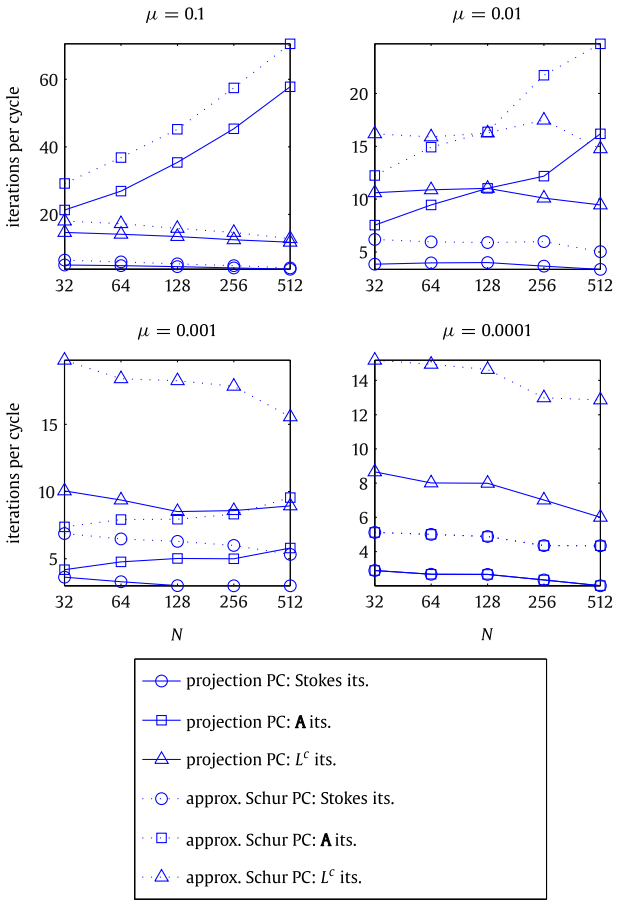

Fig. 8. Average number of linear solver iterations per incompressible Stokes solve for the projection-preconditioned and approximate Schur-preconditioned FGMRES solver for specified normal and tangential velocity boundary conditions.

To test the convergence properties of the scheme for a similar problem which does not possess corner singularities, we also consider a regularized version [36] of the lid driven cavity problem in which the lid velocity smoothly tapers to zero at the upper corners of the computational domain. In particular, for the regularized lid-driven cavity flow, we set the velocity along the upper boundary, where $y = 1$, to be $(u, v) = (\frac{1}{2}(1 + \sin(2\pi x - \frac{1}{2}\pi)), 0)$. The velocity along all other boundaries is set to be $(u, v) = (0, 0)$, as is done in the standard lid-driven cavity problem. Table 19 demonstrates that we obtain fully second-order convergence rates for this regularized problem at $Re = 1000$.

Because the error in the computed velocity is concentrated in the thin boundary layers near the boundary of the physical domain, it is tempting to try to improve the accuracy of the computed solutions by employing higher-order accurate implementations of the boundary conditions. Recall that although normal velocity boundary conditions are imposed exactly, our scheme uses linear extrapolation to prescribe tangential velocity boundary conditions. As a final test of the scheme, we replace the standard second-order accurate treatment of the tangential velocity boundary conditions described in Section 3.3 with a third-order accurate treatment, in which we determine ghost values for the tangential velocity using a quadratic fit to the nearest two interior values along with the prescribed boundary value. As shown in Table 20, the modified scheme retains second-order convergence rates for the regularized lid-driven cavity problem at $Re = 1000$. Moreover, comparing Tables 19 and 20, we see that the use of third-order accurate tangential velocity boundary conditions results in a modest improvement in the estimated accuracy of the pressure, and results in a dramatic improvement in the estimated accuracy of the velocity, especially as measured in the discrete L^∞ norm. The use and implementation of alternative boundary condition treatments is greatly facilitated by our use of an unsplit time discretization.

5.3.3. Computational performance

To compare the computational performance of the projection preconditioner to that of the approximate Schur complement preconditioner, we run our unsteady solver for the standard lid-driven cavity flow problem to time $t = 0.25$, using $\mathbf{u}(\mathbf{x}, t = 0) \equiv (0, 0)$ as the initial condition. With this initial condition, the flow is far from steady state at $t = 0.25$. For these tests, we use a relative convergence tolerance of 1.0e−10 for the FGMRES algorithm and a relative convergence tolerance of

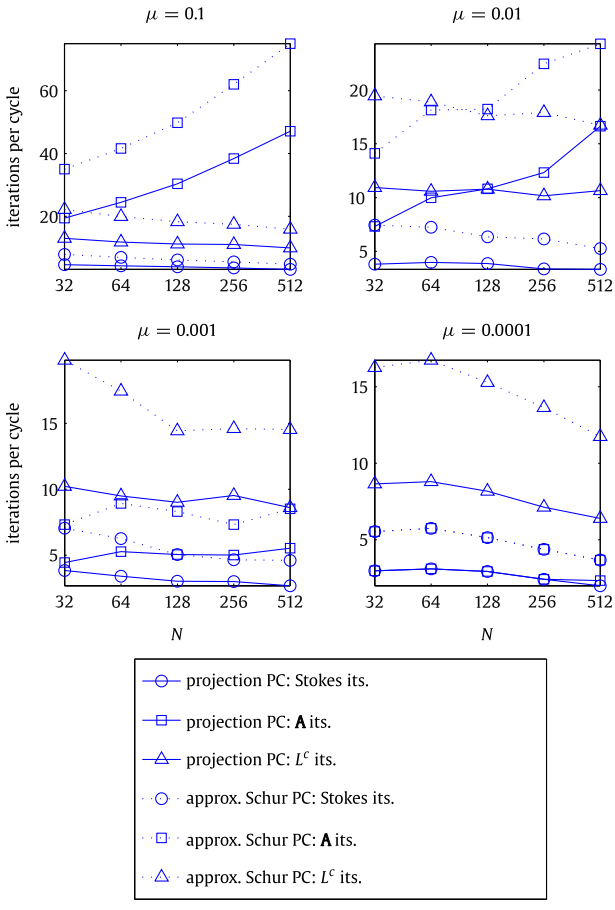

Fig. 10. Average number of linear solver iterations per incompressible Stokes solve for the projection-preconditioned and approximate Schur-preconditioned FGMRES solver for specified normal traction and tangential velocity boundary conditions.

1.0e−2 for the subdomain solvers. The pressure subdomain solver uses CG preconditioned by PFMG, and the velocity solver uses CG preconditioned by point Jacobi. Table 21 displays the total runtime normalized by the number of timesteps and by the size of the computational grid (i.e., N^2 for an $N \times N$ grid). These tests were performed on a Linux server with four quad-core AMD Opteron processors, although only a single core was used for the timings. For this problem, the projection preconditioner yields higher performance than the block factorization-based preconditioner. Both algorithms appear to be scalable in the sense that the computational work per gridpoint is nearly constant as N increases. Additionally, both preconditioners appear to be relatively insensitive to the value of Re.

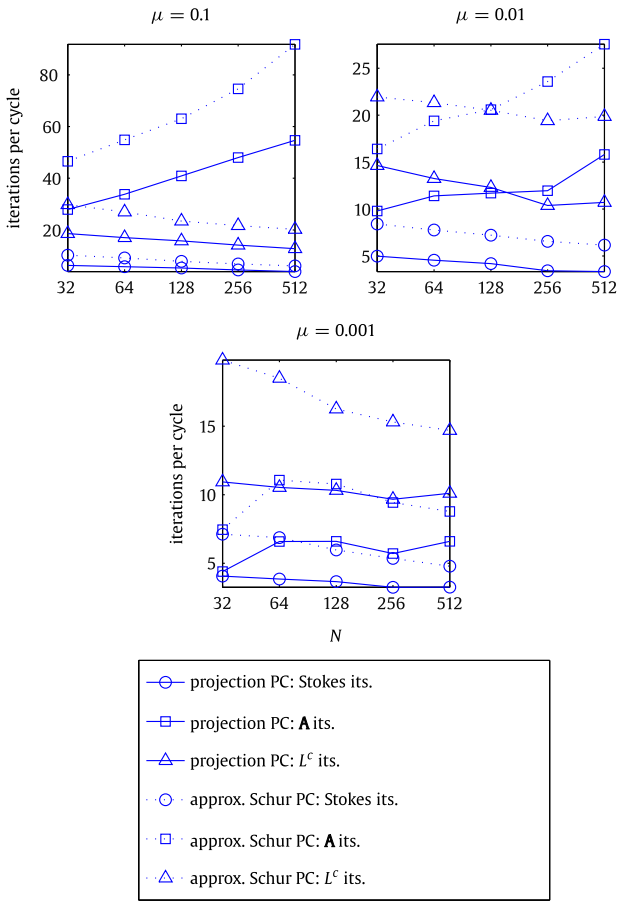

Fig. 11. Average number of linear solver iterations per incompressible Stokes solve for the projection-preconditioned and approximate Schur-preconditioned FGMRES solver for specified normal and tangential traction boundary conditions.

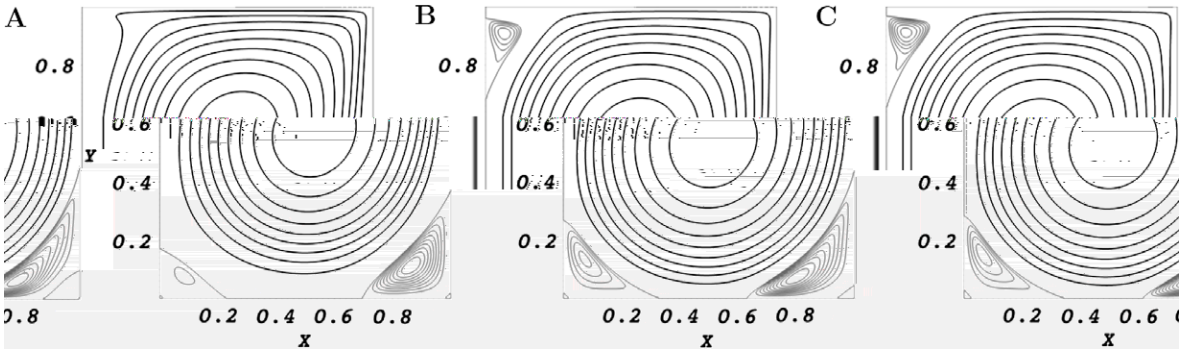

Fig. 12. Selected streamlines of the computed steady solutions to the lid-driven cavity flow problem at **A**. $Re = 1000$, **B**. $Re = 5000$, and **C**. $Re = 7500$ for $N = 256$. Regions of clockwise rotation are indicated by black lines. Counterclockwise vortices near the corners of the domain appear in grey.

6. Discussion

In this work, we have presented an unsplit, linearly-implicit discretization of the incompressible Navier–Stokes equations on a staggered grid along with an effective solution methodology for the resulting system of linear equations. Because we do not use a fractional-step algorithm, it is straightforward to specify physical boundary conditions accurately, and our convergence results demonstrate that the discretization is second-order accurate for a variety of choices of physical boundary conditions over a broad range of Reynolds numbers. Possible directions of future research include extending the methodology to treat variable density and viscosity flows, and exploring alternative time discretizations and different treatments for boundary conditions, especially traction boundary conditions. Work is presently underway to extend the solver to support adaptive mesh refinement.

Our linearly-implicit time discretization requires the solution of the time-dependent incompressible Stokes equations one or more times per timestep. To do so efficiently, we employ the projection method as a preconditioner for a FGMRES solver applied to the Stokes operator, and our results indicate that the projection method-based preconditioner is generally more effective than a similar preconditioner based on an approximation to the Schur complement of the incompressible Stokes system. When either preconditioner is equipped with multigrid-preconditioned subdomain solvers, our results demonstrate that the preconditioned FGMRES solver is algorithmically scalable for $Re \approx 1$ and for $Re \gg 1$. At higher Reynolds

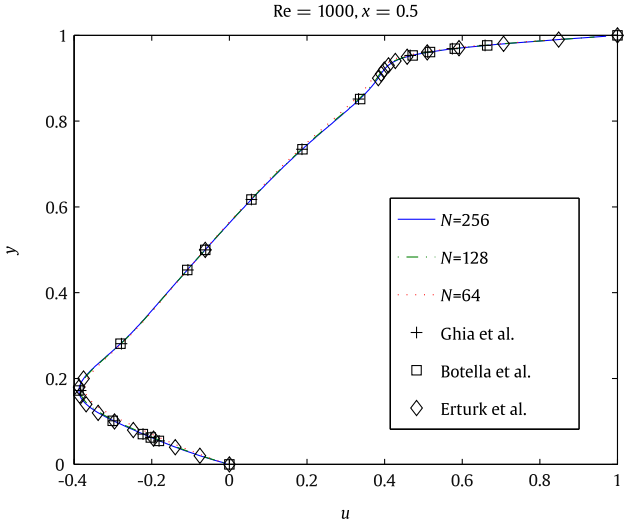

Fig. 13. The u-component of the velocity as a function of y at $x = 0.5$ for $Re = 1000$ for the lid-driven cavity flow problem.

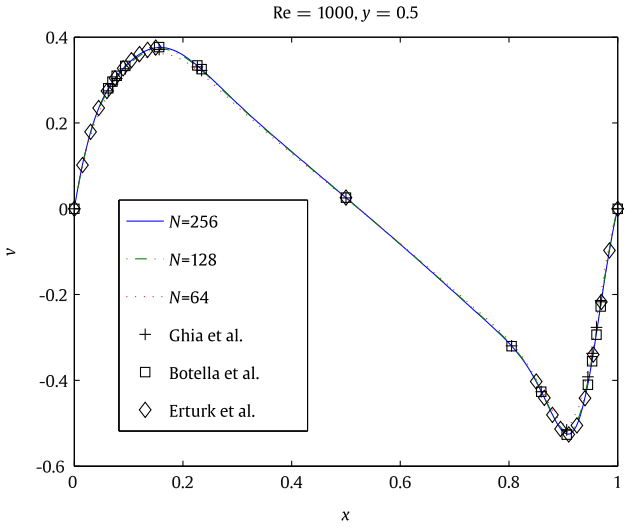

Fig. 14. The v-component of the velocity as a function of x at $y = 0.5$ for $Re = 1000$ for the lid-driven cavity flow problem.

There are at least three possible routes to extending the present solver framework to handle complex geometries. For relatively simple domains, it should be possible to extend the present finite difference methodology to mapped, logically-Cartesian multiblock grids, although in this case the boundary treatment will become substantially more difficult to implement. Another approach would be to employ a finite element spatial discretization in place of the present finite difference scheme. In this case, the construction of the corresponding projection preconditioner could possibly be accomplished in a manner similar to that described in [40,41]. It is also possible to treat complex geometry using Cartesian grid approaches such as the immersed boundary method [60,22,26,29]. In fact, we are presently developing a new immersed boundary code which employs the present projection-preconditioned incompressible flow solver.

Finally, we emphasize that we expect that the projection method will serve as an effective preconditioner not only for the present time discretization but for *any* linearly-implicit discretization of the incompressible Navier–Stokes equations which requires the solution of the time-dependent incompressible Stokes equations. Converting an existing fractional-step incompressible flow solver which uses a staggered-grid projection method to an unsplit solver is straightforward, essentially requiring only that the projection solver be "wrapped" by an outer flexible Krylov solver such as FGMRES. Thus, using the projection method as a preconditioner rather than as a solver may be an attractive approach to improving the support for general physical boundary conditions in established incompressible flow codes.

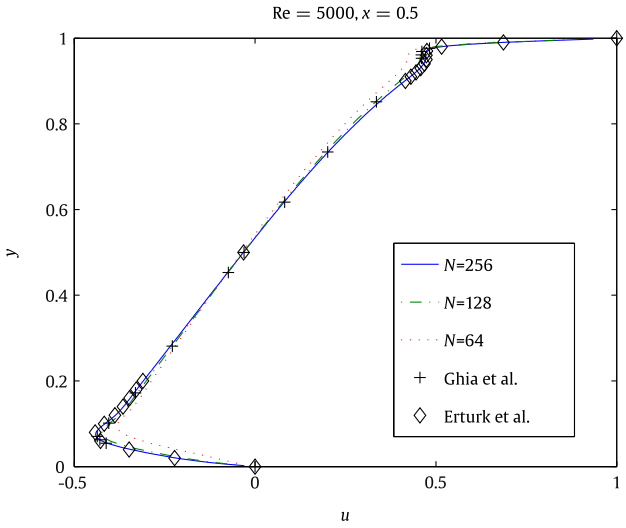

Fig. 17. The u-component of the velocity as a function of y at $x = 0.5$ for $Re = 5000$ for the lid-driven cavity flow problem.

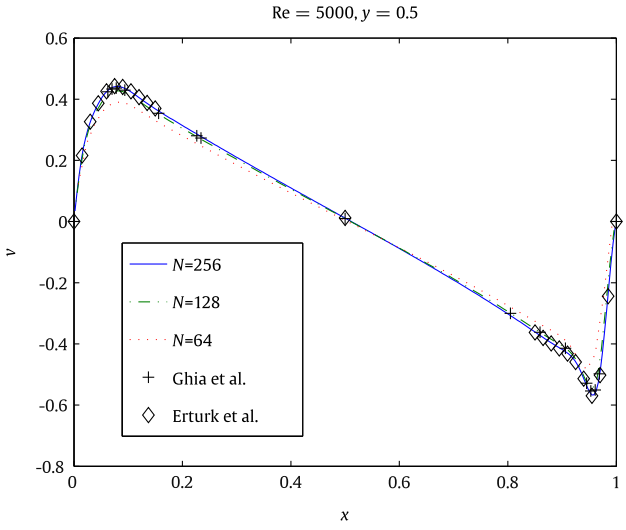

Fig. 18. The v-component of the velocity as a function of x at $y = 0.5$ for $Re = 5000$ for the lid-driven cavity flow problem.

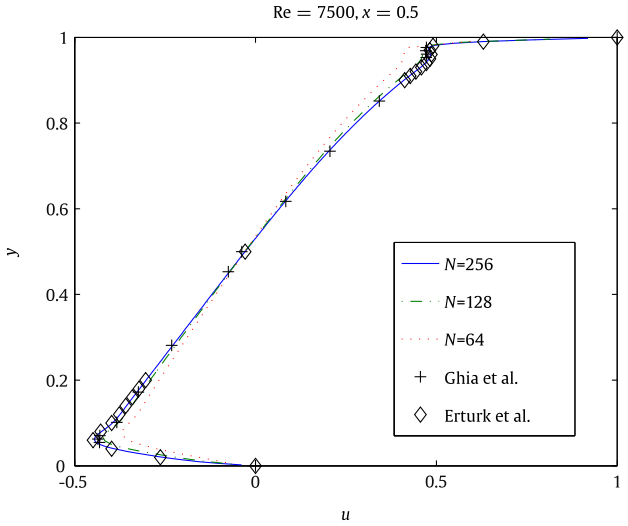

Fig. 19. The u-component of the velocity as a function of y at $x = 0.5$ for $Re = 7500$ for the lid-driven cavity flow problem.

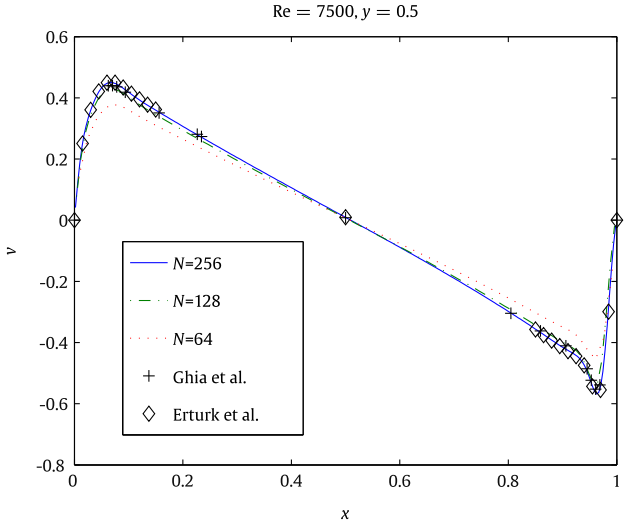

Fig. 20. The v-component of the velocity as a function of x at $y = 0.5$ for $Re = 7500$ for the lid-driven cavity flow problem.

Table 17
Estimated errors and convergence rates for u and p for the standard lid-driven cavity flow problem at $Re = 100$. Note that the exact solution possesses singularities at the upper corners of the domain, which result in a discontinuity in u and blowup in p.

	64×64	Rate	128×128	Rate	256×256
Error in u at Re = 100					
L^1	5.87e−04	1.91	1.57e−04	1.89	4.22e−05
L^2	2.28e−03	1.20	9.90e−04	1.09	4.66e−04
L^∞	5.96e−02	0.32	4.77e−02	0.18	4.20e−02
Error in p at Re = 100					
L^1	5.45e−04	1.21	2.36e−04	1.11	1.09e−04
L^2	1.01e−02	0.01	1.00e−02	0.00	1.00e−02
L^∞	4.85e−01	−0.93	9.26e−01	−0.97	1.81e+00

Table 18
Estimated errors and convergence rates for u and p for the standard lid-driven cavity flow problem at $Re = 1000$. Because the effects of the corner singularities are somewhat diminished at higher Reynolds numbers, the apparent order of accuracy of the scheme is higher at $Re = 1000$ than it is at $Re = 100$. Compare to Table 17.

	64×64	Rate	128×128	Rate	256×256
Error in u at Re = 1000					
L^1	6.04e−03	2.19	1.32e−03	1.95	3.43e−04
L^2	9.32e−03	1.81	2.65e−03	1.55	9.05e−04
L^∞	1.25e−01	0.13	1.14e−01	0.31	9.19e−02
Error in p at Re = 1000					
L^1	1.84e−03	1.84	5.11e−04	1.92	1.35e−04
L^2	2.90e−03	1.00	1.45e−03	0.42	1.09e−03
L^∞	1.17e−01	−0.34	1.48e−01	−0.61	2.26e−01

Table 19
Estimated errors and convergence rates for u and p for the *regularized* lid-driven cavity flow problem at $Re = 1000$. Note that the scheme obtains fully second-order convergence rates in all norms for this problem.

	64×64	Rate	128×128	Rate	256×256
Error in u at Re = 1000					
L^1	2.58e−03	1.91	6.85e−04	1.95	1.77e−04
L^2	3.95e−03	1.85	1.10e−03	1.92	2.89e−04
L^∞	3.72e−02	1.74	1.12e−02	1.83	3.14e−03
Error in p at Re = 1000					
L^1	6.32e−04	1.74	1.90e−04	1.96	4.88e−05
L^2	7.89e−04	1.74	2.36e−04	1.97	6.00e−05
L^∞	3.52e−03	1.80	1.01e−03	1.97	2.59e−04

Table 20
Estimated errors and convergence rates for u and p for the regularized lid-driven cavity flow problem at $Re = 1000$ with a third-order accurate tangential velocity boundary condition implementation. Note that the estimated errors are similar to or lower than those obtained with the second-order accurate tangential velocity boundary condition implementation, and that the estimated L^∞ errors in u are approximately one half to one third of those obtained with the second-order implementation. Compare to Table 19.

	64×64	Rate	128×128	Rate	256×256
Error in u at Re = 1000					
L^1	1.48e−03	1.64	4.75e−04	1.81	1.35e−04
L^2	2.24e−03	1.69	6.96e−04	1.86	1.92e−04
L^∞	1.89e−02	2.08	4.48e−03	2.03	1.10e−03
Error in p at $Re = 1000$					
L^1	3.23e−04	1.21	1.40e−04	1.78	4.07e−05
L^2	4.57e−04	1.34	1.81e−04	1.83	5.08e−05
L^∞	3.45e−03	1.70	1.06e−03	2.10	2.47e−04

Table 21
Computational performance of the solver using either the projection preconditioner P_{proj} or the approximate Schur complement preconditioner $\widehat{P}_{\text{Schur}}$ for the standard (un-regularized) lid-driven cavity flow problem. Runtimes are reported in seconds and are normalized by the number of timesteps and by the size of the computational grid (i.e., N^2). The initial velocity is set to be uniformly zero, and the unsteady solver is run to time $t = 0.25$. Note that the projection preconditioner outperforms the approximate Schur complement preconditioner for this problem for these choices of parameters. Both preconditioners yield an essentially scalable algorithm.

Re	64×64	128×128	256×256
Normalized runtime (s) using P_{proj}			
100	3.47e−5	3.54e−5	5.21e−5
1000	2.57e−5	2.36e−5	3.12e−5
5000	2.24e−5	2.07e−5	2.50e−5
Normalized runtime (s) using $\widehat{P}_{\text{Schur}}$			
100	4.35e−05	4.21e−05	5.91e−05
1000	3.72e−05	3.18e−05	4.21e−05
5000	3.22e−05	2.93e−05	3.74e−05

Acknowledgments

The author thanks D.B. Kothe of Oak Ridge National Laboratory for the initial suggestion to investigate the combination of Godunov methods and staggered-grid spatial discretizations, and acknowledges correspondence with W.J. Rider of Sandia National Laboratories on the use of Godunov methods with staggered-grid discretizations during the initial phase of this work. The author also thanks C.S. Peskin and D.M. McQueen of the Courant Institute-New York University for helpful comments on an early draft of the present manuscript, and thanks the anonymous reviewers, whose comments greatly improved this paper. This work was sponsored in part by an American Heart Association Postdoctoral Fellowship (Grant Number 0626001T). Computations were performed at New York University using computer facilities funded in large part by a generous donation by St. Jude Medical, Inc.

Appendix A. Using xsPPM7 with a staggered-grid velocity field

When applied to the numerical solution of the incompressible Navier–Stokes equations, PPM [32] and other higher order Godunov methods are typically employed with cell-centered discretizations. A notable exception is the work of Tau [12], who employed a MAC discretization with a piecewise linear Godunov scheme. In the present work, we employ a staggered-grid Godunov scheme to compute an upwinded approximation to $(\mathbf{u}\cdot\nabla)\mathbf{u}$. Note that unlike typical cell-centered Godunov schemes which extrapolate cell-centered values at time t^n to edge-centered values at time $t^{n+\frac{1}{2}} = t^n + \frac{1}{2}\Delta t$, we *only* use the Godunov scheme to extrapolate values in space and *not* in time.

Our staggered-grid Godunov scheme proceeds as follows: We consider the u- and v-components of the staggered-grid velocity field separately, and for each component, we construct a system of control volumes which are centered about that component. For instance, the control volumes centered about the u-component of the velocity field are obtained by "shifting" the computational grid by $\frac{1}{2}\Delta x = \frac{1}{2}h$ in the x-direction; see Fig. A1. Similarly, the control volumes centered about the v-component of the velocity field are obtained by shifting the computational grid by $\frac{1}{2}\Delta y = \frac{1}{2}h$ in the y-direction. Advection velocities are computed at the centers of the edges of the control volumes by linearly interpolating the staggered-grid velocity field. We next extrapolate values from the centers of the control volumes to the edges of the control volumes via the xsPPM7 scheme described in [31]. We finally employ second-order non-conservative differencing to compute approximations to $(\mathbf{u}\cdot\nabla)u$ and $(\mathbf{u}\cdot\nabla)v$ on the staggered grid, using the advection velocity and the upwinded, xsPPM7-extrapolated values. Note that once appropriate systems of control volumes have been constructed, and once the advection velocities have been determined on the edges of the control volumes, the remainder of the scheme is the same as a cell-centered Godunov scheme. In fact, our staggered-grid implementation simply re-uses an existing cell-centered advection code [22,29] with Δt set to zero, so that values are extrapolated in space but not in time.

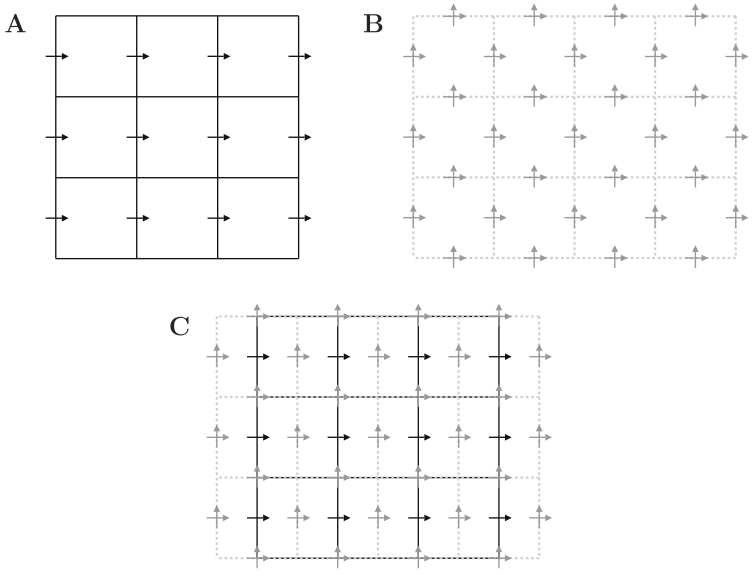

Fig. A1. Grid systems and velocity fields employed by the staggered-grid Godunov scheme to compute $(\mathbf{u}\cdot\nabla)u$. The data structures employed to compute $(\mathbf{u}\cdot\nabla)v$ are analogous, except that the control volumes are centered about the v-components of the staggered-grid velocity field instead of the u-components. A. The computational grid along with the u-components of the staggered-grid velocity field. B. The control volumes centered about the velocity components depicted in panel A along with the corresponding advection velocities. These control volumes are obtained by shifting the computational grid by $\frac{1}{2}\Delta x = \frac{1}{2}h$ in the x direction. The advection velocities at the centers of the edges of the control volumes are computed by linearly interpolating the staggered-grid velocity field $\mathbf{u} = (u, v)$. **C.** The superposition of the grids shown in panels A and B.

All ghost values required by the advection scheme in the vicinity of physical boundaries are obtained via linear extrap-

Finally, by plugging $\mathbf{u}^* = \mathbf{u}^{n+1} + \frac{\Delta t}{\rho}\mathbf{G}\varphi$ into Eq. (B.3) and comparing the result to Eq. (B.1), we have that

$$\mathbf{G}p^{n+\frac{1}{2}} = \mathbf{G}\varphi - \frac{\Delta t}{\rho}\frac{\mu}{2}\mathbf{L}\mathbf{G}\varphi. \tag{B.8}$$

We obtain a formula for $p^{n+\frac{1}{2}}$ by positing $\mathbf{L}\mathbf{G} = \mathbf{G}L^c$, so that

$$p^{n+\frac{1}{2}} = \varphi - \frac{\Delta t}{\rho}\frac{\mu}{2}L^c\varphi = \left(I - \frac{\Delta t}{\rho}\frac{\mu}{2}L^c\right)\varphi. \tag{B.9}$$

Generally, $\mathbf{L}$ and $\mathbf{G}$ do not commute, and $\mathbf{L}\mathbf{G} = \mathbf{G}L^c$ holds only in special cases, such as when periodic boundary conditions are imposed on the computational domain. In fact, for problems with periodic boundaries, the projection method is an *exact* solver for Eqs. (B.1) and (B.2). In the presence of physical boundary conditions, however, typically $\mathbf{L}\mathbf{G} \neq \mathbf{G}L^c$. Moreover, it is generally possible only to impose approximations to the true physical boundary conditions with projection methods. Enforcing normal traction, or tangential velocity or traction, boundary conditions is especially difficult; see [18,23].

References

[1] A.J. Chorin, Numerical solution of the Navier–Stokes equations, Math. Comput. 22 (104) (1968) 745–762.
[2] A.J. Chorin, On the convergence of discrete approximations to the Navier–Stokes equations, Math. Comput. 23 (106) (1969) 341–353.
[3] M.L. Minion, A projection method for locally refined grids, J. Comput. Phys. 127 (1) (1996) 158–178.
[4] A.S. Almgren, J.B. Bell, P. Colella, T. Marthaler, A Cartesian grid projection method for the incompressible Euler equations in complex geometries, SIAM J. Sci. Comput. 18 (5) (1997) 1289–1309.
[5] D.F. Martin, P. Colella, A cell-centered adaptive projection method for the incompressible Euler equations, J. Comput. Phys. 163 (2) (2000) 271–312.
[6] A.S. Almgren, J.B. Bell, W.Y. Crutchfield, Approximate projection methods: Part I. Inviscid analysis, SIAM J. Sci. Comput. 22 (4) (2000) 1139–1159.
[7] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, J. Comput. Phys. 190 (2) (2003) 572–600.
[8] S.Y. Kadioglu, R. Klein, M.L. Minion, A fourth-order auxiliary variable projection method for zero-Mach number gas dynamics, J. Comput. Phys. 227 (3) (2008) 2012–2043.
[9] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (2) (1985) 308–323.
[10] J.B. Bell, P. Colella, H.M. Glaz, A second-order projection method for the incompressible Navier–Stokes equations, J. Comput. Phys. 85 (2) (1989) 257–283.
[11] R.P. Beyer, A computational model of the cochlea using the immersed boundary method, J. Comput. Phys. 98 (1) (1992) 145–162.
[12] E.Y. Tau, A 2nd-order projection method for the incompressible Navier–Stokes equations in arbitrary domains, J. Comput. Phys. 115 (1) (1994) 147–152.
[13] A.S. Almgren, J.B. Bell, W.G. Szymczak, A numerical method for the incompressible Navier–Stokes equations based on an approximate projection, SIAM J. Sci. Comput. 17 (2) (1996) 358–369.
[14] L.H. Howell, J.B. Bell, An adaptive mesh projection method for viscous incompressible flow, SIAM J. Sci. Comput. 18 (4) (1997) 996–1013.
[15] A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome, A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations, J. Comput. Phys. 142 (1) (1998) 1–46.
[16] W.J. Rider, Filtering non-solenoidal modes in numerical solutions of incompressible flows, Int. J. Numer. Methods Fluid 28 (5) (1998) 789–814.
[17] A.M. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys. 153 (2) (1999) 509–534.
[18] D.L. Brown, R. Cortez, M.L. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 168 (2) (2001) 464–499.
[19] Z.-L. Li, M.-C. Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comput. Phys. 171 (2) (2001) 822–842.
[20] L. Lee, R.J. LeVeque, An immersed interface method for incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 25 (3) (2003) 832–856.
[21] R.D. Guy, A.L. Fogelson, Stability of approximate projection methods on cell-centered grids, J. Comput. Phys. 203 (2) (2005) 517–538.
[22] B.E. Griffith, C.S. Peskin, On the order of accuracy of the immersed boundary method: Higher order convergence rates for sufficiently smooth problems, J. Comput. Phys. 208 (1) (2005) 75–105.
[23] B. Yang, A. Prosperetti, A second-order boundary-fitted projection method for free-surface flow computations, J. Comput. Phys. 213 (2) (2006) 574–590.
[24] C. Min, F. Gibou, A second order accurate projection method for the incompressible Navier–Stokes equations on non-graded adaptive grids, J. Comput. Phys. 219 (2) (2006) 912–929.
[25] Z. Zheng, L. Petzold, Runge–Kutta–Chebyshev projection method, J. Comput. Phys. 219 (2) (2006) 976–991.
[26] B.E. Griffith, R.D. Hornung, D.M. McQueen, C.S. Peskin, An adaptive, formally second order accurate version of the immersed boundary method, J. Comput. Phys. 223 (1) (2007) 10–49.
[27] D.F. Martin, P. Colella, D. Graves, A cell-centered adaptive projection method for the incompressible Navier–Stokes equations in three dimensions, J. Comput. Phys. 227 (3) (2008) 1863–1886.
[28] D.V. Le, B.C. Khoo, K.M. Lim, An implicit-forcing immersed boundary method for simulating viscous flows in irregular domains, Comput. Methods Appl. Mech. Eng. 197 (2008) 2119–2130.
[29] B.E. Griffith, X. Luo, D.M. McQueen, C.S. Peskin, Simulating the fluid dynamics of natural and prosthetic heart valves using the immersed boundary method, Int. J. Appl. Mech. 1 (1) (2009) 137–177.
[30] J.L. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Comput. Methods Appl. Mech. Eng. 195 (44–47) (2006) 6011–6045.
[31] W.J. Rider, J.A. Greenough, J.R. Kamm, Accurate monotonicity- and extrema-preserving methods through adaptive nonlinear hybridizations, J. Comput. Phys. 225 (2) (2007) 1827–1848.
[32] P. Colella, P.R. Woodward, The piecewise parabolic method (PPM) for gas-dynamical simulations, J. Comput. Phys. 54 (1) (1984) 174–201.
[33] U. Ghia, K.N. Ghia, C.T. Shin, High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method, J. Comput. Phys. 48 (3) (1982) 387–411.
[34] O. Botella, R. Peyret, Benchmark spectral results on the lid-driven cavity flow, Comput. Fluid 27 (4) (1998) 421–433.
[35] E. Erturk, T.C. Corke, C. Gökçöl, Numerical solutions of 2-D steady incompressible driven cavity flow at high Reynolds numbers, Int. J. Numer. Methods Fluid 48 (7) (2005) 747–774.
[36] C.J. Roy, A.J. Sinclair, On the generation of exact solutions for evaluating numerical schemes and estimating discretization error, J. Comput. Phys. 228 (5) (2009) 1790–1802.
[37] D. Kay, D. Loghin, A. Wathen, A preconditioner for the steady-state Navier–Stokes equations, SIAM J. Sci. Comput. 24 (2002) 237–256.
[38] D. Silvester, H. Elman, D. Kay, A. Wathen, Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow, J. Comput. Appl. Math. 128 (1–2) (2001) 261–279.

[39] H.C. Elman, V.E. Howle, J.N. Shadid, R.S. Tuminaro, A parallel block multi-level preconditioner for the 3D incompressible Navier–Stokes equations, J. Comput. Phys. 187 (2) (2003) 504–523.
[40] H. Elman, V.E. Howle, J. Shadid, R. Shuttleworth, R. Tuminaro, Block preconditioners based on approximate commutators, SIAM J. Sci. Comput. 27 (5) (2006) 1651–1668.
[41] H. Elman, V.E. Howle, J. Shadid, R. Shuttleworth, R. Tuminaro, A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier–Stokes equations, J. Comput. Phys. 227 (3) (2008) 1790–1808.
[42] D.A. Knoll, V.A. Mousseau, On Newton–Krylov multigrid methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 163 (1) (2000) 262–267.
[43] M. Pernice, M.D. Tocci, A multigrid-preconditioned Newton–Krylov method for the incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 23 (2) (2001) 398–418.
[44] S. Balay, V. Eijkhout, W.D. Gropp, L.C. McInnes, B.F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in: E. Arge, A.M. Bruaset, H.P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhäuser Press, 1997, pp. 163–202.
[45] S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, PETSc Web page, 2009. <http://www.mcs.anl.gov/petsc>.
[46] S. Balay, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 – Revision 3.0.0, Argonne National Laboratory, 2008.
[47] P.M. Gresho, R.L. Sani, Incompressible Flow and the Finite Element Method: Advection–Diffusion and Isothermal Laminar Flow, John Wiley & Sons, 1998.
[48] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluid 8 (12) (1965) 2182–2189.
[49] Y. Saad, A flexible inner–outer preconditioned GMRES algorithm, SIAM J. Sci. Comput. 14 (2) (1993) 461–469.
[50] V. Simoncini, D.B. Szyld, Flexible inner–outer Krylov subspace methods, SIAM J. Numer. Anal. 40 (6) (2003) 2219–2239.
[51] V. Simoncini, D.B. Szyld, Recent computational developments in Krylov subspace methods for linear systems, Numer. Linear Algebra Appl. 14 (1) (2007) 1–59.
[52] M.F. Murphy, G. Golub, A.J. Wathen, A note on preconditioning for indefinite linear systems, SIAM J. Sci. Comput. 21 (6) (2000) 1969–1972.
[53] S. Schaffer, A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients, SIAM J. Sci. Comput. 20 (1) (1998) 228–242.
[54] P.N. Brown, R.D. Falgout, J.E. Jones, Semicoarsening multigrid on distributed memory machines, SIAM J. Sci. Comput. 21 (5) (2000) 1823–1834. also available as LLNL technical report UCRL-JC-130720.
[55] R.D. Falgout, J.E. Jones, Multigrid on massively parallel architectures, in: E. Dick, K. Riemslagh, J. Vierendeels (Eds.), Multigred Methods VI, Lecture Notes in Computational Science and Engineering, vol. 14, Springer-Verlag, 2000, pp. 101–107. also available as LLNL Technical Report UCRL-JC-133948..
[56] S.F. Ashby, R.D. Falgout, A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations, Nucl. Sci. Eng. 124 (1) (1996) 145–159. also available as LLNL Technical Report UCRL-JC-122359.
[57] *hypre*: high performance preconditioners. <http://www.llnl.gov/CASC/hypre>.
[58] R.D. Falgout, U.M. Yang, *hypre*: a library of high performance preconditioners, in: P.M.A. Sloot, C.J.K. Tan, J.J. Dongarra, A.G. Hoekstra (Eds.), Computational Science – ICCS 2002 Part III, Lecture Notes in Computer Science, vol. 2331, Springer-Verlag, 2002, pp. 632–641, also available as LLNL Technical Report UCRL-JC-146175.
[59] Y.-F. Peng, Y.H. Shiau, R.R. Hwang, Transition in a 2-D lid-driven cavity flow, Comput. Fluid 32 (3) (2003) 337–352.
[60] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[61] D.S. Balsara, Divergence-free adaptive mesh refinement for magnetohydrodynamics, J. Comput. Phys. 174 (2) (2001) 614–648.
[62] G. Toth, P.L. Roe, Divergence- and curl-preserving prolongation and restriction formulas, J. Comput. Phys. 180 (2) (2002) 736–750.