COMMUNICATIONS IN COMPUTATIONAL PHYSICS
Vol. 1, No. 5, pp. 874-885

Commun. Comput. Phys.
October 2006

An Augmented Approach for the Pressure Boundary Condition in a Stokes Flow

Zhilin Li[1,*], Xiaohai Wan[2], Kazufumi Ito[1] and Sharon R. Lubkin[1]

[1] *Center for Research in Scientific Computation & Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205, USA.*

[2] *Biomathematics Graduate Program & Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8203, USA.*

Received 2 October 2005; Accepted (in revised version) 28 February 2006

Abstract. An augmented method is proposed for solving stationary incompressible Stokes equations with a Dirichlet boundary condition along parts of the boundary. In this approach, the normal derivative of the pressure along the parts of the boundary is introduced as an additional variable and it is solved by the GMRES iterative method. The dimension of the augmented variable in discretization is the number of grid points along the boundary which is $\mathcal{O}(N)$. Each GMRES iteration (or one matrix-vector multiplication) requires three fast Poisson solvers for the pressure and the velocity. In our numerical experiments, only a few iterations are needed. We have also combined the augmented approach for Stokes equations involving interfaces, discontinuities, and singularities.

Key words: Incompressible Stokes equations; pressure boundary condition; augmented method; interface problem; immersed interface method; fast Poisson solver; GMRES method.

1 Introduction

In this paper, we consider the following stationary incompressible Stokes equations:

$$\nabla p = \nabla \cdot \mu \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right) + \mathbf{g}(\mathbf{x}), \qquad (x,y) \in [a,\ b] \times [c,\ d], \tag{1.1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{1.2}$$

$$\mathbf{u}(x,c) = \mathbf{u}_1(x), \qquad \mathbf{u}(x,d) = \mathbf{u}_2(x), \tag{1.3}$$

$$\mathbf{u}(a,y) = \mathbf{u}(b,y), \qquad p(a,y) = p(b,y), \tag{1.4}$$

[*] Correspondence to: Zhilin Li, Center for Research in Scientific Computation & Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205, USA. Email: zhilin@math.ncsu.edu

where $\mathbf{u} = (u,v)^T$ is the velocity, p is the pressure, μ is the viscosity, $\mathbf{x} = (x,y)$ is the Cartesian coordinate variable, $\mathbf{u}_1(x)$ and $\mathbf{u}_2(x)$ are two given vector functions, and $\mathbf{g} = (g_1, g_2)$ is an external force. In other words, we have a Dirichlet boundary condition for the velocity at the two sides $y = c$ and $y = d$, and a periodic boundary condition for all variables at $x = a$ and $x = b$, see Fig. 1 for an illustration. Such Stokes equations have many applications. In this paper, we assume that the viscosity is constant or piecewise constant.

Note that if the viscosity is constant, then by applying the divergence operator to the momentum equation, we get

$$\Delta p = \nabla \cdot \mathbf{g}, \tag{1.5}$$

due to the incompressibility condition. A fast Poisson solver, *e.g.*, Fishpack [1] can be applied to solve the pressure if the boundary condition of the pressure is Dirichlet, Neumann, or periodic. After the pressure is solved, the velocity can be solved from

$$\Delta u = \frac{p_x - g_1}{\mu}, \qquad \Delta v = \frac{p_y - g_2}{\mu}, \tag{1.6}$$

by applying the same fast Poisson solver twice. This approach is called *the three Poisson equations approach.* For example, if all variables (p and $\mathbf{u}$) are periodic, then the stationary Stokes equations can be solved by calling a fast Poisson solver three times.

If a Dirichlet boundary condition is prescribed for the velocity, then it is well known that the pressure along the boundary is not a free variable and it is uniquely determined (up to a constant) from the governing equation (1.1)-(1.4). In other words, we can not specify the boundary condition for the pressure arbitrarily. Otherwise the problem is over-determined. This brings some difficulties to numerical schemes using uniform Cartesian grids. Note that from the momentum equation, we do have a boundary condition (1.7) for the pressure, but it is coupled with the velocity. For time-dependent Stokes or Navier-Stokes equations, some approaches in dealing with pressure boundary conditions are discussed in [5, 8, 15]. However, for *stationary* Stokes equations, one cannot use the three fast Poisson solver approach directly because the pressure boundary condition is coupled with the velocity as follows

$$\left.\frac{\partial p}{\partial n}\right|_{y=c,y=d} = \Big(\mu\,(\Delta \mathbf{u}) \cdot \mathbf{n} + \mathbf{g} \cdot \mathbf{n} \Big)\Big|_{y=c,y=d}. \tag{1.7}$$

In this paper, we propose a novel method by introducing an augmented variable that is only defined along the boundary where a Dirichlet boundary condition of the velocity is prescribed. The idea of the new method is to set $\partial p/\partial n$ as part of the unknowns, which we call the augmented variable, along the boundary, which should satisfy the momentum equation projected to the same boundary. Given a guess of the augmented variable, we can solve the Stokes equations easily by calling a fast Poisson solver three times. Since the augmented variable is defined along the boundary, we can get a small system of equations that can be solved efficiently using the GMRES iterative method. Each iteration requires a fast Stokes solver with a known $\partial p/\partial n$. The number of iterations is very small, $3 \sim 4$ for a problem with smooth solutions, and $7 \sim 9$ for an interface problem in which the pressure is discontinuous and the velocity is non-smooth as shown in Section 3.2.

The idea of the augmented approach is from the recent research for solving incompressible Stokes equations with discontinuous viscosity [12] where the authors introduce an augmented variable to decouple the jump condition for the pressure and the velocity. While augmented approaches have been developed for elliptic interface problems or problems defined on irregular domains in [2, 6, 7, 11, 13, 14, 17], the augmented approach proposed in this paper provides a way to get a second order finite difference discretization to solve the stationary Stokes equations with a prescribed velocity. Note that the Stokes equations can also be solved as a bi-harmonic equation by introducing the vorticity $\omega = \nabla \times \ \mathbf{u}$ to get $-\Delta\psi = \omega$, and $\mu\Delta^2\psi = -\nabla \times \mathbf{g}$, where ψ is the stream function. With this, Glowinski in [3] proposed an algorithm that may bear some similarities to the augmented method proposed in this paper. In [3], the pressure (not the normal derivative) along the boundary is introduced as a Lagrangian multiplier. The multiplier then is coupled with another equation from the formulation of the related bi-harmonic equation discussed in [4]. A mixed finite element method is often used to solve the system.

2 The augmented method for the pressure boundary condition

Assume that we are given a pressure boundary condition, either periodic or of Neumann type, along the four sides of the rectangle $R = [a,\ b] \times\ [c,\ d]$. Then an approximation of the pressure $P_{ij} \approx p(x_i, y_j)$ can be obtained from the finite difference equations

$$\begin{aligned} &\frac{P_{i-1,j} - 2P_{i,j} + P_{i+1,j}}{h_x^2} + \frac{P_{i,j-1} - 2P_{i,j} + P_{i,j+1}}{h_y^2} \\ &\quad = \frac{g_1(x_{i+1}, y_j) - g_1(x_{i-1}, y_j)}{2h_x} + \frac{g_2(x_i, y_{j+1}) - g_2(x_i, y_{j-1})}{2h_y}, \end{aligned} \tag{2.1}$$

where h_x and h_y are the mesh spacing in the x- and y- directions, respectively. The linear system of equations can be solved by, for example, the fast Poisson solver from Fishpack [1]. The fast solver allows Dirichlet, Neumann, and periodic boundary conditions along each side of a rectangular domain. Once we have solved the pressure, then we can solve the velocity from

$$\frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h_x^2} + \frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{h_y^2} = \frac{P_{i+1,j} - P_{i-1,j}}{\mu\, 2h_x} - \frac{g_1(x_i, y_j)}{\mu}, \tag{2.2}$$

$$\frac{V_{i-1,j} - 2V_{i,j} + V_{i+1,j}}{h_x^2} + \frac{V_{i,j-1} - 2V_{i,j} + V_{i,j+1}}{h_y^2} = \frac{P_{i,j+1} - P_{i,j-1}}{\mu\, 2h_y} - \frac{g_2(x_i, y_j)}{\mu}, \tag{2.3}$$

where $(U_{ij}, V_{ij}) \approx (u(x_i, y_j), v(x_i, y_j))$ is an approximation of the velocity at the grid point (x_i, y_j).

2.1 Introducing the augmented variable

Assume that the velocity is prescribed along two sides $y = c$ and $y = d$. The Neumann boundary condition of the pressure is coupled with the Laplacian of the velocity which is unknown. Therefore, we can not solve the pressure using (2.1) directly.

The idea of our new method is to set $\frac{\partial p}{\partial n}|_{y=c,y=d}$ as part of the unknowns, which we call the augmented variable

$$q = \left.\frac{\partial p}{\partial \boldsymbol{n}}\right|_{y=c,y=d}, \tag{2.4}$$

which should satisfy the following equation

$$q = \left.\frac{\partial p}{\partial n}\right|_{y=c,y=d} = \Big(\mu\,(\Delta \mathbf{u}) \cdot \mathbf{n} + \mathbf{g} \cdot \mathbf{n} \Big)\Big|_{y=c,y=d}. \tag{2.5}$$

The equations (2.4)-(2.5) above, together with the Stokes equations (1.1)-(1.4), form a closed system. Given a guess of q, we can solve the Stokes equations easily by calling a fast Poisson solver three times. The solution will depend on q, so it can be denoted as $(p(q), \mathbf{u}(q))$, which is the solution to the original problem if (1.7) is satisfied. Note that the choice of augmented variable is not unique. Different choices will lead to different algorithms. Since we wish to use three fast Poisson solvers and decouple the pressure and the velocity, it is natural to choose $\frac{\partial p}{\partial n}$ as the augmented variable.

In discretization, we set

$$\mathbf{Q} = \left[\begin{array}{ccccccc} q_1 & q_2 & \cdots & q_{M-1} & q_M & \cdots & q_{2M-2} \end{array}\right]^T \tag{2.6}$$

to define q at (x_1, c), (x_2, c), $\cdots$, (x_{M-1}, c), and (x_1, d), (x_2, d), $\cdots$, (x_{M-1}, d). The dimension of $\mathbf{Q}$ is $2M - 2$ which is much smaller than MN, where M and N are the number of grid lines in the x- and y- directions respectively.

Let us put the discrete solution $\{P_{ij}(\mathbf{Q})\}$, $\{U_{ij}(\mathbf{Q})\}$, and $\{V_{ij}(\mathbf{Q})\}$ together as a big vector $\tilde{\mathcal{U}}$ whose dimension is $\mathcal{O}(3MN)$. Then the discrete solution of (2.1)-(2.3) can be written as

$$A\tilde{\mathcal{U}} + B\mathbf{Q} = \mathbf{F}_1, \tag{2.7}$$

for some vector $\mathbf{F}_1$ and sparse matrices A and B. The operator A corresponds to solving three Poisson equations to get $\tilde{\mathcal{U}}\,(\mathbf{Q})$ given $\mathbf{Q}$.

Once we know the solution $\tilde{\mathcal{U}}$ for a given $\mathbf{Q}$, we can use $\mathbf{U}$ to get

$$R(\mathbf{Q}) = \left(\frac{\partial P(\mathbf{Q})}{\partial n} - \mu\,(\Delta \mathbf{U}(\mathbf{Q})) \cdot \mathbf{n} - \mathbf{G} \cdot \mathbf{n} \right)\Bigg|_{y=c,y=d} \tag{2.8}$$

along the boundary $y=c$ and $y=d$, or (x_i,c) and (x_i,d), where $\mathbf{G}$ is the vector formed from $\mathbf{g}$ at the grid points along the two sides $y=c$ and $y=d$. If $\|R(\mathbf{Q})\|$ is smaller than a given tolerance, then the method has already converged and P_{ij}, $\mathbf{U}_{ij}$ is a set of approximate solutions. The computation of the residual vector $R(\mathbf{Q})$, which will be explained in the next sub-section, depends on $\tilde{\boldsymbol{\mathcal{U}}}$ and $\mathbf{Q}$ *linearly.* Therefore we can write

$$R(\mathbf{Q}) = \left(\frac{\partial P(\mathbf{Q})}{\partial n} - \mu\left(\Delta \mathbf{U}(\mathbf{Q})\right)\cdot\mathbf{n} - \mathbf{G}\cdot\mathbf{n}\right)\Bigg|_{y=c,y=d} = S\tilde{\boldsymbol{\mathcal{U}}} + E\mathbf{Q} - \mathbf{F}_2, \tag{2.9}$$

where S and E are two sparse matrices, and $\mathbf{F}_2$ is a vector. The matrices depend on the interpolation scheme but do not need to be actually constructed in the algorithm. We need to choose a vector $\mathbf{Q}$ such that (2.5) is satisfied along the two sides $y=c$ and $y=d$. If we put the two matrix-vector equations (2.7) and (2.9) together, we get

$$\begin{bmatrix} A & B \\ S & E \end{bmatrix}\begin{bmatrix} \tilde{\boldsymbol{\mathcal{U}}} \\ \mathbf{Q} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix}. \tag{2.10}$$

Note that $\mathbf{Q}$ is defined only on a set of grid points along two sides $y=c$ and $y=d$, while $\tilde{\boldsymbol{\mathcal{U}}}$ is defined at all grid points. The Schur complement for $\mathbf{Q}$ is

$$(E - SA^{-1}B)\mathbf{Q} = \mathbf{F}_2 - SA^{-1}\mathbf{F}_1 = \bar{\mathbf{F}}. \tag{2.11}$$

If we can solve the system above to get $\mathbf{Q}$, then we can get $\tilde{\boldsymbol{\mathcal{U}}}$ easily. Because the dimension of $\mathbf{Q}$ is much smaller than that of $\tilde{\boldsymbol{\mathcal{U}}}$, we expect to get a fast algorithm for the Stokes equations if we can solve (2.11) efficiently.

In the implementation, the GMRES iterative method [16] is used to solve (2.11). The GMRES method only requires matrix-vector multiplication. We explain below how to evaluate the right-hand side $\bar{\mathbf{F}}$ of the Schur complement, and how to evaluate the matrix-vector multiplication needed by the GMRES iteration. We can see why we do not need to form the coefficient matrix $E - SA^{-1}B$ explicitly.

2.2 Computing the right-hand side of the Schur complement

First we set $\mathbf{Q}=\mathbf{0}$ and solve the system (2.1)-(2.3), or (2.7) in the discrete form, to get $\tilde{\boldsymbol{\mathcal{U}}}(\mathbf{0})$ which is $A^{-1}\mathbf{F}_1$ from (2.7). From the interpolation (2.9), we also have

$$R(\mathbf{0}) = \mathbf{0} - \Big(\mu\left(\Delta \mathbf{U}(\mathbf{0})\right)\cdot\mathbf{n} + \mathbf{G}\cdot\mathbf{n}\Big) = S\tilde{\boldsymbol{\mathcal{U}}}(\mathbf{0}) + E\,\mathbf{0} - \mathbf{F}_2 = S\tilde{\boldsymbol{\mathcal{U}}}(\mathbf{0}) - \mathbf{F}_2. \tag{2.12}$$

Note that the residual of the Schur complement for $\mathbf{Q}=\mathbf{0}$ is

$$\begin{aligned} R_s(\mathbf{0}) &= (E - SA^{-1}B)\,\mathbf{0} - \bar{\mathbf{F}} = -\bar{\mathbf{F}} \\ &= -\left(\mathbf{F}_2 - SA^{-1}\mathbf{F}_1\right) = -\mathbf{F}_2 + S\tilde{\boldsymbol{\mathcal{U}}}(\mathbf{0}) \\ &= \left(\frac{\partial P(\mathbf{0})}{\partial n} - \mu\left(\Delta \mathbf{U}\right)\cdot\mathbf{n} - \mathbf{G}\cdot\mathbf{n}\right)\Bigg|_{y=c,y=d}, \end{aligned} \tag{2.13}$$

which is exactly the same as $R(\mathbf{0})$ defined in (2.8). Therefore $R_s(\mathbf{0}) = R(\mathbf{0})$, and $R(\mathbf{0})$ gives the right-hand side of the Schur complement system with the opposite sign.

2.3 Computing the matrix-vector multiplication

Once the system of equations for $\mathbf{Q}$ is determined, we use the GMRES iterative method to solve $\mathbf{Q}$. Once $\mathbf{Q}$ has been found, the pressure and velocity can be obtained by solving three Poisson equations. Given an initial guess $\mathbf{Q}^0$, the GMRES method will produce a sequence $\{\mathbf{Q}^k\}$. Each iteration requires the matrix-vector multiplication $(E-SA^{-1}B)\mathbf{Q}^k$. For simplification of notations, we omit the superscript k and write $(E-SA^{-1}B)\mathbf{Q}^k$ as $(E-SA^{-1}B)\mathbf{Q}$.

The matrix-vector multiplication of the Schur complement system for a given $\mathbf{Q}$ is obtained from the following two steps:

Step 1: Solve the Stokes system (2.1)-(2.3), or (2.7) in the discrete form, to get $\tilde{\boldsymbol{\mathcal{U}}}(\mathbf{Q})$.

Step 2: Interpolate $\tilde{\boldsymbol{\mathcal{U}}}(\mathbf{Q})$ using (2.9) to get $R(\mathbf{Q})$ defined in (2.8). The procedure is explained in the next subsection. Then the matrix-vector multiplication is

$$\left(E-SA^{-1}B\right)\mathbf{Q}=R(\mathbf{Q})-R(\mathbf{0}). \tag{2.14}$$

This is because

$$\begin{aligned}\left(E-SA^{-1}B\right)\mathbf{Q}&=E\mathbf{Q}-SA^{-1}B\mathbf{Q}\\&=E\mathbf{Q}-S\left(A^{-1}\mathbf{F}_1-\tilde{\boldsymbol{\mathcal{U}}}(\mathbf{Q})\right) &&\Big(\text{from (2.7)}\Big),\\&=E\mathbf{Q}+S\tilde{\boldsymbol{\mathcal{U}}}(\mathbf{Q})-\mathbf{F}_2+\mathbf{F}_2-SA^{-1}\mathbf{F}_1\\&=R(\mathbf{Q})+\bar{\mathbf{F}} &&\Big(\text{from (2.9)}\Big),\\&=R(\mathbf{Q})-R(\mathbf{0}), &&\Big(\text{from (2.13)}\Big).\end{aligned}$$

Now we can see that a matrix-vector multiplication is equivalent to solving the system (1.1)-(1.4) with $\frac{\partial p}{\partial n}|_{y=c,y=d}=q$, or (2.7) in the discrete form, to get $\tilde{\boldsymbol{\mathcal{U}}}$, and using an interpolation scheme (2.9) to get $R(\mathbf{Q})$ at the grid points along two sides $y=c$ and $y=d$.

Since we know the right-hand side of the linear system of equations and the matrix-vector multiplication of the coefficient matrix, it is straightforward to use GMRES or other iterative methods.

In summary, the system of equations for $\mathbf{Q}$ can be written as

$$A_q\mathbf{Q}=\mathbf{b}, \tag{2.15}$$

where

$$\mathbf{b}=-R(\mathbf{0})=-\left(\frac{\partial P(\mathbf{0})}{\partial n}-\mu\left(\Delta\mathbf{U}(\mathbf{0})\right)\cdot\mathbf{n}-\mathbf{G}\cdot\mathbf{n}\right)\Bigg|_{y=c,y=d}. \tag{2.16}$$

The coefficient matrix A_q does not need to be formed explicitly. Given a guess of $\mathbf{Q}$, the matrix-vector multiplication that is needed for the GMRES iterative method is simply

$$A_q\mathbf{Q} = R(\mathbf{Q}) - R(\mathbf{0}). \tag{2.17}$$

The only question remaining is how to compute the Laplacian of the velocity along the sides $y = c$ and $y = d$, which is not trivial.

2.4 Computing the Laplacian of the velocity along a boundary for a Dirichlet boundary condition

In this subsection, we explain how to interpolate $\mathbf{u}(q)$ to get the Laplacian $\Delta\mathbf{u}(q)$ that is needed in (2.8) along the boundary $y = d$. The discussion for the boundary $y = c$ is similar. Without loss of generality, we assume $v_{xx} = 0$ along the side $y = d$. Since $\mathbf{n} = (0,\ 1)$ along $y = d$, we just need to approximate $\Delta v = v_{xx} + v_{yy} = v_{yy}$. To compute v_{yy} along $y = d$, we make use of the incompressibility condition $u_x + v_y = 0$. From the Taylor expansion we have

$$\begin{aligned} v(x, d-h_y) &= v(x,d) - v_y(x,d)h_y + \frac{1}{2}v_{yy}(x,d)h_y^2 + \mathcal{O}(h_y^3) \\ &= v(x,d) + u_x(x,d)h_y + \frac{1}{2}v_{yy}(x,d)h_y^2 + \mathcal{O}(h_y^3). \end{aligned} \tag{2.18}$$

Thus we get an approximation for v_{yy}

$$v_{yy}(x,d) \approx \frac{2\left(v(x,d-h_y) - v(x,d) - u_x(x,d)h_y\right)}{h_y^2}. \tag{2.19}$$

Note that $u_x(x,d)$ is computable since $u(x,t)$ is given along $y = d$, so is its first order derivative, $v(x, d-h_y)$ has been obtained after we have solved the three Poisson equations for $p(q)$, $u(q)$, $v(q)$ given q.

3 Numerical examples of the augmented pressure boundary condition method

We have carried out a number of experiments to validate our new method. All the computations are done on a P4 3.00GHz DELL desktop PC with 1GB memory. The computational domain is $\Omega = [-2,2] \times [-2,2]$. The tolerance of the GMRES iteration is set to be 10^{-5}.

3.1 An example of a rectangular domain

In this example, we solve the Stokes equations with a body force $\mathbf{g}$. The boundary condition is periodic in the x direction. In the y direction ($y = c$ and $y = d$), Dirichlet boundary

Table 1: A grid refinement analysis for the example in Section 3.1. Second order convergence is confirmed.

n	$\|E_n(p)\|_\infty$	$Order(p)$	$\|E_n(u)\|_\infty$	$Order(u)$	$\|E_n(v)\|_\infty$	$Order(v)$
32	$1.4068\ 10^{-1}$		$4.5500\ 10^{-2}$		$2.5701\ 10^{-2}$	
64	$3.7910\ 10^{-2}$	1.8918	$1.1499\ 10^{-2}$	1.9844	$6.3541\ 10^{-3}$	2.0161
128	$9.8619\ 10^{-3}$	1.9426	$2.8843\ 10^{-3}$	1.9952	$1.5875\ 10^{-3}$	2.0009
256	$2.5159\ 10^{-3}$	1.9708	$7.2161\ 10^{-4}$	1.9989	$3.9646\ 10^{-4}$	2.0015
512	$6.3542\ 10^{-4}$	1.9853	$1.8044\ 10^{-4}$	1.9997	$9.9062\ 10^{-5}$	2.0008
1024	$1.5967\ 10^{-4}$	1.9926	$4.5114\ 10^{-5}$	1.9999	$2.4759\ 10^{-5}$	2.0004

Table 2: The number of GMRES iterations and CPU time for the example in Section 3.1. Note that only three iterations are needed regardless of grids.

n	n_{iter}	CPU Time (s)	n	n_{iter}	CPU Time (s)
32	3	<1	256	3	1
64	3	<1	512	3	4
128	3	<1	1024	3	18

conditions for the velocity $\mathbf{u} = (u, v)$ are prescribed. The constructed exact solutions are

$$\begin{aligned} &p = y\, e^y \cos\left(\frac{\pi x}{2}\right) + y \sin(\pi x), \\ &u = 4\, y \cos\left(\frac{\pi x}{2}\right), \qquad v = y^2\, \pi \sin\left(\frac{\pi x}{2}\right). \end{aligned}$$

The body force $\mathbf{g}$ is

$$\begin{aligned} g_1 &= -\frac{\pi}{2} \sin\left(\frac{\pi x}{2}\right) e^y y + \cos(\pi x)\, \pi y + \pi^2 \cos\left(\frac{\pi x}{2}\right) y, \\ g_2 &= \cos\left(\frac{\pi x}{2}\right) e^y y + \cos\left(\frac{\pi x}{2}\right) e^y + \frac{\pi^3}{4} \sin\left(\frac{\pi x}{2}\right) y^2 - 2\pi \sin\left(\frac{\pi x}{2}\right) + \sin(\pi x), \end{aligned} \tag{3.1}$$

which is determined from the governing Stokes equations. In Table 1 we show a grid refinement analysis by doubling the the grid lines in each coordinate direction. The order of convergence is defined as

$$Order(\cdot) = \frac{\log\left(\|E_{n/2}(\cdot)\|_\infty / \|E_n(\cdot)\|_\infty\right)}{\log 2}. \tag{3.2}$$

We clearly see second-order convergence in the maximum norm. In Table 2, we show an efficiency analysis for the algorithm. The number of GMRES iterations is only 3 and it is independent of the mesh size. The CPU time is also listed which indicates that our method behaves like a fast algorithm for the Stokes equations.

3.2 An example of an interface problem

Now we present an example of an interface problem where the source term has a singularity

$$\mathbf{g}(\mathbf{x}) + \int_\Gamma \mathbf{f}(s)\,\delta\left(\mathbf{x} - \mathbf{X}(s)\right) ds, \tag{3.3}$$

where $\Gamma \subset R$ is a C^2 curve in the solution domain; δ is the Dirac delta function in two dimensions which is defined in the distribution sense, $\mathbf{X}$ is a point on the interface. For this type of problem, we have the standard Stokes equations (1.1)-(1.2) in the interior domain excluding the interface. We define $\mathbf{n}$ and $\boldsymbol{\tau}$ as the unit normal and tangential directions of the interface Γ. The solutions in each sub-domain are coupled with the following jump conditions.

$$[p] = \hat{f}_1, \qquad \left[\frac{\partial p}{\partial n}\right] = \frac{d\hat{f}_2}{ds}, \tag{3.4}$$

$$[u] = 0, \qquad \left[\frac{\partial u}{\partial n}\right] = \hat{f}_2 \sin\theta, \tag{3.5}$$

$$[v] = 0, \qquad \left[\frac{\partial v}{\partial n}\right] = -\hat{f}_2 \cos\theta, \tag{3.6}$$

where θ is the angle between the normal direction and the x- axis, $\hat{f}_1 = \mathbf{f}\cdot\mathbf{n}$, $\hat{f}_2 = \mathbf{f}\cdot\boldsymbol{\tau}$, and $d\hat{f}_2/ds$ is the tangential derivative of $\hat{f}_2$ which is defined along the interface. The immersed interface method has been applied to solve this type of interface problem with the pressure boundary condition being either a periodic or an approximated Neumann boundary condition. Given a Dirichlet, or Neumann, or periodic boundary condition for the pressure and the velocity, the pressure and the velocity can still be solved by calling the three Poisson solver three times. The only modification needed is to add correction terms to the right-hand side of the finite difference equations at irregular grid points, where the interface cuts through the standard central stencil. For example, the finite difference equation for the pressure would be

$$\begin{aligned} &\frac{P_{i-1,j} - 2P_{i,j} + P_{i+1,j}}{h_x^2} + \frac{P_{i,j-1} - 2P_{i,j} + P_{i,j+1}}{h_y^2} \\ &\quad = \frac{g_1(x_{i+1}, y_j) - g_1(x_{i-1}, y_j)}{2h_x} + \frac{g_2(x_i, y_{j+1}) - g_2(x_i, y_{j-1})}{2h_y} + C_{ij}, \end{aligned} \tag{3.7}$$

where C_{ij} is zero at a regular grid point, and a non-zero correction term at an irregular grid point. The correction term depends on the source density $\mathbf{f}$ and the geometry of the interface (tangential and normal direction and the curvature). We refer the readers to [9,10] on how to determine the correction terms. The emphasis of this paper is to show that the augmented method for the pressure boundary condition works well for interface problems as well.

Now we present such a numerical example. We assume $\mu = 1.0$ in the solution domain. The interface Γ is chosen as a periodic function along the x direction

$$y = \cos\left(\frac{\pi x}{2}\right). \tag{3.8}$$

The exact solution of $(\mathbf{u}, p)$ is constructed as

$$(u, v, p) = \begin{cases} \left(\cos(\pi x/2) + y,\ \dfrac{\pi y}{2}\sin(\pi x/2),\ y\right) & \text{if } y < \cos(\pi x/2), \\ \left(2y,\ \dfrac{\pi}{2}\sin(\pi x/2)\cos(\pi x/2),\ \dfrac{\pi}{2}\sin(\pi x/2) + y\right) & \text{if } y \geq \cos(\pi x/2). \end{cases} \tag{3.9}$$

The body force term $\mathbf{g} = (g_1, g_2)^T$ is determined from the Stokes equations

$$(g_1, g_2) = \begin{cases} \left(\dfrac{\pi^2}{4}\cos(\pi x/2),\ 1 + \dfrac{\pi^3 y}{8}\sin(\pi x/2)\right) & \text{if } y < \cos(\pi x/2), \\ \left(\dfrac{\pi^2}{4}\cos(\pi x/2),\ 1 + \dfrac{\pi^3}{2}\cos(\pi x/2)\sin(\pi x/2)\right) & \text{if } y \geq \cos(\pi x/2) \end{cases} \tag{3.10}$$

and the force density is

$$\hat{f}_1 = \frac{\pi}{2}\sin(\pi x/2), \qquad \hat{f}_2 = 1 + \frac{\pi^2}{4} - \frac{\pi^2}{4}\cos(\pi x/2)^2. \tag{3.11}$$

In Table 3 we show a grid refinement analysis by doubling the the grid lines in each coordinate direction. We clearly see second order convergence in the maximum norm. In Table 4, we show an efficiency analysis for the algorithm. The number of GMRES iterations is only about 7 and it is independent of the mesh size. The number of iterations is more than that in Table 1 reflecting the fact that we are solving a more difficult problem due to the presence of the interface. The CPU time is also listed which indicates that our method behaves like a fast algorithm for the Stokes equations. We show the solution domain and the interface in Fig. 1(a), and the error plot of the pressure computed from a 60 by 60 grid in Fig. 1(b). While the error of the pressure is large along $y = d = 2$ where the exact solution pressure is a sine function of x, the error is smooth and decreases with $h = \max\{h_x, h_y\}$ quadratically. The error on the other side $y = c = -2$ is small because pressure is a linear function.

4 Conclusions

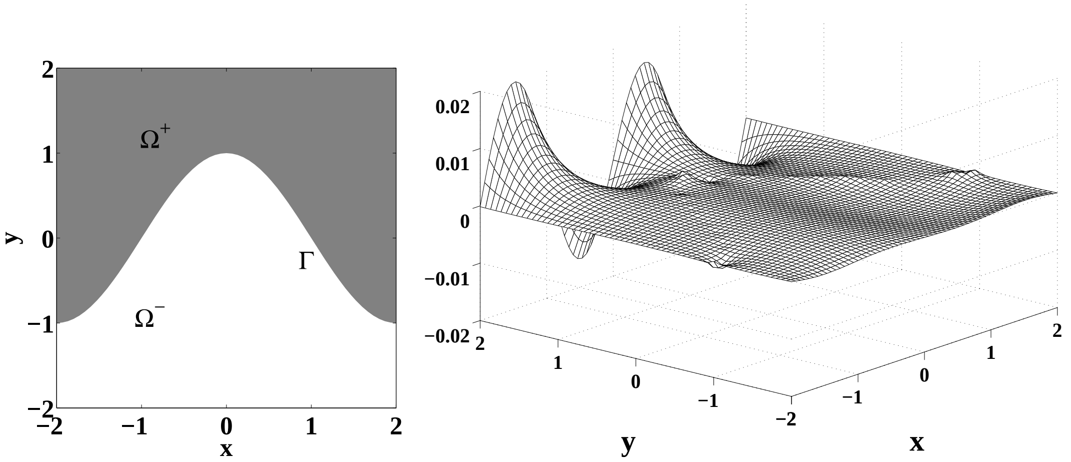
In this paper, we proposed an augmented method for solving incompressible Stokes equations with a Dirichlet boundary condition along parts of the boundary. The main idea is to introduce $\partial p/\partial n$ as an augmented variable along the parts of the boundary where the Dirichlet boundary condition is known. The augmented equation is the momentum equation projected to the part of the boundary. In the discrete case, the focus is to solve the

Table 3: A grid refinement analysis for the example in Section 3.2. Second order convergence is confirmed.

n	$\|E_n(p)\|_\infty$	$Order(p)$	$\|E_n(u)\|_\infty$	$Order(u)$	$\|E_n(v)\|_\infty$	$Order(v)$
32	$6.9641\ 10^{-2}$		$1.6940\ 10^{-2}$		$1.3313\ 10^{-2}$	
64	$1.7424\ 10^{-2}$	1.9989	$4.4485\ 10^{-3}$	1.9290	$3.1352\ 10^{-3}$	2.0862
128	$4.4292\ 10^{-3}$	1.9760	$1.1796\ 10^{-3}$	1.9150	$7.3526\ 10^{-4}$	2.0922
256	$1.1082\ 10^{-3}$	1.9988	$2.9497\ 10^{-4}$	1.9997	$1.8091\ 10^{-4}$	2.0230
512	$2.7856\ 10^{-4}$	1.9922	$7.6178\ 10^{-5}$	1.9531	$4.4888\ 10^{-5}$	2.0109
1024	$6.9494\ 10^{-5}$	2.0030	$1.8919\ 10^{-5}$	2.0095	$1.1159\ 10^{-5}$	2.0081

Table 4: The number of GMRES iterations and CPU time for the example in Section 3.2. Note that only $7 \sim 9$ iterations are needed regardless of grids.

n	n_{iter}	CPU Time (s)	n	n_{iter}	CPU Time (s)
32	9	<1	256	7	2
64	8	<1	512	7	7
128	7	<1	1024	7	28



(a)

(b)

Figure 1: (a). Computational domain and interface for the example in Section 3.2. (b). Error plot of the computed pressure.

augmented variable whose dimension is much smaller than those for the pressure and the velocity by the GMRES iterative method. Each GMRES iteration (or one matrix-vector multiplication) requires three fast Poisson solvers for the pressure and the velocity. We have also implemented the augmented approach for Stokes equations involving interfaces.

Acknowledgments

The authors were partially supported by USA NSF-NIH grant #0201094, the USA-ARO under grant number 43751-MA, and an NSF grant DMS-0412654. We would also like to thank the referees for helpful comments.

References

[1] J. Adams, P. Swarztrauber and R. Sweet, Fishpack: Efficient Fortran subprograms for the solution of separable elliptic partial differential equations, http://www.netlib.org/fishpack/.

[2] G. Chen, Z. Li and P. Lin, A fast finite difference method for biharmonic equations on irregular domains, Technical Report CRSC-TR04-09, North Carolina State University, 2004.

[3] R. Glowinski, Numerical Methods for Nonlinear Variational Problems, Springer-Verlag, New York, 1984.

[4] R. Glowinski and O. Pironneau, Numerical methods for the first biharmonic equation and for the two-dimensional Stokes problem, SIAM Rev., 21 (1979), 167–212.

[5] J. L. Guermond and J. Shen, A new class of truly consistent splitting schemes for incompressible flows, J. Comput. Phys., 192 (2003), 262–276.

[6] T. Hou, Z. Li, S. Osher and H. Zhao, A hybrid method for moving interface problems with application to the Hele-Shaw flow, J. Comput. Phys., 134 (1997), 236–252.

[7] J. Hunter, Z. Li and H. Zhao, Autophobic spreading of drops, J. Comput. Phys., 183 (2002), 335–366.

[8] H. Johnston and J. Liu, Accurate, stable and efficient Navier-Stokes solvers based on explicit treatment of the pressure term, J. Comput. Phys., 188 (2004), 221–259.

[9] R. J. LeVeque and Z. Li, Immersed interface method for Stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput., 18 (1997), 709–735.

[10] Z. Li, The Immersed Interface Method — A Numerical Approach for Partial Differential Equations with Interfaces, Ph.D Thesis, University of Washington, 1994.

[11] Z. Li, A fast iterative algorithm for elliptic interface problems, SIAM J. Numer. Anal., 35 (1998), 230–254.

[12] Z. Li, K. Ito and M-C. Lai, An augmented approach for Stokes equations with a discontinuous viscosity and singular forces, Technical Report CRSC-TR04-23, North Carolina State Univeristy, 2004.

[13] Z. Li and C. Wang, A fast finite difference method for solving Navier-Stokes equations on irregular domains, J. Commun. Math. Sci., 1 (2003), 180–196.

[14] Z. Li, H. Zhao and H. Gao, A numerical study of electro-migration voiding by evolving level set functions on a fixed cartesian grid, J. Comput. Phys., 152 (1999), 281–304.

[15] R. Peyret and T. D. Taylor, Computational Methods for Fluid Flow, Springer-Verlag, New York, 1983.

[16] Y. Saad, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., 7 (1986), 856–869.

[17] A. Wiegmann and K. Bube, The explicit jump immersed interface method: Finite difference methods for PDEs with piecewise smooth solutions, SIAM J. Numer. Anal., 37 (2000), 177–200.