STABLE AND SPECTRALLY ACCURATE SCHEMES FOR THE NAVIER–STOKES EQUATIONS

JUN JIA† AND JIE LIU‡

Abstract. In this paper, we present an accurate, efficient and stable numerical method for the incompressible Navier–Stokes equations (NSEs). The method is based on (1) an equivalent pressure Poisson equation formulation of the NSE with proper pressure boundary conditions, which facilitates the design of high-order and stable numerical methods, and (2) the Krylov deferred correction (KDC) accelerated method of lines transpose (MoL$^T$), which is very stable, efficient, and of arbitrary order in time. Numerical tests with known exact solutions in three dimensions show that the new method is spectrally accurate in time, and a numerical order of convergence 9 was observed. Two-dimensional computational results of flow past a cylinder and flow in a bifurcated tube are also reported.

Key words. Navier–Stokes equations, Krylov deferred correction, pressure Poisson equation, open and traction boundary conditions

AMS subject classifications. 65M60, 76D05, 65B05

DOI. 10.1137/090754340

1. Introduction. Since the spectral deferred correction (SDC) method was introduced [7], deferred correction-type methods have regained the attention of people interested in designing temporally high-order numerical methods. The SDC method uses a simple low-order method to solve a series of correction equations, which improves the numerical solution iteratively. The Krylov deferred correction (KDC) method [22, 24] was developed as an accelerated version of SDC [21] and coupled with the method of lines transpose (MoL$^T$) to solve partial differential equations (PDEs) [23]. The KDC provides faster convergence than the SDC and is not affected by the order reduction posed by the stiffness of an ordinary differential equation (ODE).

The main advantage of a deferred correction method is that a simple low-order method can be used on an equivalent error equation formulation to compute high-order accurate solutions. It also inherits the computational complexity and stability properties of the underlying low-order method and, in some cases, can improve these properties. To solve PDEs, one can use deferred correction with the traditional method of lines; the PDE system becomes an ODE system, and then deferred correction is applied. With MoL$^T$, fast elliptic solvers such as fast multipole methods can be used directly to solve PDEs with deferred correction. The deferred correction technique is a robust tool for constructing stable and accurate numerical time-stepping methods.

*Submitted to the journal’s Computational Methods in Science and Engineering section March 29, 2009; accepted for publication (in revised form) June 27, 2011; published electronically September 29, 2011. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/33-5/75434.html

†Computational Mathematics Group, Oak Ridge National Laboratory, Oak Ridge, TN 37831 (jiaj@ornl.gov). The work of this author was performed under contract number DE-AC0500OR22750 between the U.S. Department of Energy and Oak Ridge Associated Universities.

‡Department of Mathematics, National University of Singapore, Singapore 119076 (matlj@nus. edu.sg). This author was supported by a start-up grant from the National University of Singapore, R-146-000-129-133.
In this paper, we design high-order KDC accelerated schemes to efficiently solve the Navier–Stokes equations (NSEs) for incompressible fluid flow. The model equations are as follows:

\[ \begin{align*}
\partial_t \vec{u} + \vec{u} \cdot \nabla \vec{u} + \nabla p &= \nu \Delta \vec{u} + \vec{f} \quad \text{in } \Omega, \\
\nabla \cdot \vec{u} &= 0 \quad \text{in } \Omega,
\end{align*} \]

where \( \Omega \) is a domain in \( \mathbb{R}^m \) \((m \geq 2)\), \( \vec{u} \) is the fluid velocity, \( p \) is the pressure, \( \vec{f} \) is the body force, and \( \nu = 1/Re \) is the kinematic viscosity, where \( Re \) is the Reynolds number.

The ability to handle boundary conditions without compromising the high-order accuracy is important when designing high-order methods. In our framework, different boundary conditions can be treated, for example, the periodic boundary condition, or the no-slip boundary condition on \( \Gamma_1 \) and the open boundary condition on \( \Gamma_2 \) (\( \partial \Omega = \Gamma_1 \cup \Gamma_2 \)):

\[ \begin{align*}
\vec{u} &= \vec{g}_1 \quad \text{on } \Gamma_1, \\
\nu \partial_n \vec{u} - p \vec{n} &= \vec{g}_2 \quad \text{on } \Gamma_2.
\end{align*} \]

The proposed method is demonstrated with boundary conditions (1.3) and (1.4). The extension to other boundary conditions is straightforward. For example, the traction boundary condition,

\[ \nu(\nabla \vec{u} + (\nabla \vec{u})^\top)\vec{n} - p \vec{n} = \vec{g}_2 \quad \text{on } \Gamma_2, \]

can be enforced on \( \Gamma_2 \). In this case, adding \( \nu \nabla \cdot \vec{u} \) to the right-hand side of (1.1) allows us to treat this similarly to (1.4) \([20, 15, 34]\). These boundary conditions (1.4) and (1.5) are interesting because one often needs to truncate a large physical domain to make the problem tractable. Moreover, the traction boundary condition (1.5) is also related to free boundary and fluid-structure interaction problems.

**NSE formulations for temporally high-order methods.** Different NSE formulations can be used with high-order deferred correction methods. We prefer the pressure Poisson equation (PPE) formulation in our approach. In the literature, researchers have been actively working on the construction of temporally high-order methods for the NSE. In such a context, the projection method introduced by Chorin \([6]\) and Temam \([40]\), in which the updates of velocities and pressure are decoupled, is a very popular subject. In the original projection method, the velocity field is obtained by first solving a Burgers’ equation and then projecting the intermediate velocity field onto the divergence-free space. Before the deferred correction-type methods, it was generally assumed that it was nontrivial to develop and analyze higher-order projection methods \([15]\). Many second-order and third-order temporally accurate projection methods are reviewed in \([15]\). However, the pressure correction schemes \([15]\) which are a generalization of the popular second-order scheme proposed by Timmermans, Minev, and Van de Vosse \([41]\), have an instability for both small and large \( \Delta t \) when the temporal order reaches four \([33]\). Therefore it is nontrivial to design higher-order projection methods using only the pressure correction idea. Besides the pressure correction, velocity correction, consistent splitting, and inexact factorization schemes reviewed in \([15]\), another active area of research is in developing high-order projection methods that are designed to accurately approximate the boundary conditions for the auxiliary velocity field \([30, 10, 3, 33]\). Deferred correction-type methods are
applicable in this context, and in [35] one of the authors of [3] has already combined an auxiliary variable projection method with SDC and achieved high-order temporal accuracy. Periodic boundary conditions were used in [35], so that the dominant error term came from the temporal integration and hence was easily demonstrated. Yet, to the best of our knowledge, there is no discussion about how to design accurate boundary conditions for the auxiliary velocity field when the boundary condition is of Neumann type (1.4). So, instead of accurate auxiliary velocity boundary conditions, we turn to accurate pressure boundary conditions with PPE.

Historically, closely related to projection methods, people have developed methods based on the PPE formulation [36, 27, 13], whose computational costs are similar to those of projection methods. As demonstrated in [15], many projection methods are equivalent to PPE-based methods when the spatial variables are continuous, particularly, the consistent splitting scheme [16]. The pressure boundary condition introduced in [36] is particularly important for many later developments of PPE formulation. An important breakthrough was made by Johnston and Liu in [25], where they showed a deeper understanding of the PPE and presented how to use $C^0$ finite elements when the pressure boundary condition contains second-order derivatives of velocities. Later developments include sharper and more general pressure estimates [31], error estimates of a first-order scheme [32], a unified approach to high-order schemes that has (both) accurate boundary conditions for auxiliary velocity field or/and accurate boundary conditions for the pressure [33], extension to Neumann-type boundary conditions [34], and the spectrally accurate schemes that we will discuss in this paper.

There are also other possible high-order formulations of the NSE, for example, the Runge–Kutta methods by E and Liu [9] or the popular mixed finite element/inf-sup stable finite elements methods [11], where velocity and pressure are coupled together in each time step. Interested readers are welcome to investigate along these directions and report back.

In summary, we show in this paper that KDC, with PPE and proper pressure boundary conditions, can achieve efficient high-order accurate solutions in time with Neumann-type boundary conditions. One notable difference between the traditional SDC and our KDC/SDC is that the temporal derivative of the targeted function is used as the unknown variable here, whereas the traditional SDC method uses the function itself (e.g., the temporal derivative of the velocity verses the velocity in the case of NSE).

This paper is organized as follows: In section 2, we briefly review the PPE formulation and a stable semi-implicit low-order scheme for the NSE with no-slip and open boundaries, which later becomes the building block of KDC accelerated MoL$^T$. In section 3, we discuss the main ideas of the KDC accelerated MoL$^T$ and its applications to the NSE. In the last section, we present numerical results showing the spectral accuracy and stability of the resulting schemes. Numerical results for flow past a cylinder and flow in a bifurcated tube are also reported.

### 2. Well-posedness with Neumann-type boundary conditions, pressure Poisson formulation, and the building block for the KDC accelerated MoL$^T$

With $\bar{g}_2 = 0$, (1.4) or (1.5) means that the pseudotraction or the traction is set to be zero on $\Gamma_2$, which certainly is physically meaningful as it mimics the case in elasticity where the free boundary of an elastic body is set to be traction-free. But in the fluid mechanics literature, there are many other boundary conditions that have been proposed on the outflow boundary when the outflux is not given [39]. We choose (1.4) or (1.5) because of the well-posedness: with (1.4) or (1.5), the resulting system...
is proven well-posed by Grubb and Solonnikov [14] and by Heywood, Rannacher, and Turek [20]. To our knowledge, there is no literature about the well-posedness of other open boundary conditions cited in [39]. The well-posedness is a very important issue, as a limited number of numerical tests will never exhaust all the cases one can meet in real applications, not to mention that the proof of the well-posedness can lead to good numerical schemes. Indeed, as we will discuss later, with (1.4), the pressure boundary conditions proposed in [34] lead to a PPE, from which one can conclude that the pressure in the momentum equation can be controlled by the viscosity term. This means for the Stokes equations that, if the viscosity term in the momentum equation is treated implicitly, the resulting scheme can still be unconditionally stable even when the pressure is treated explicitly.

In PPE formulation, the $\nabla \cdot \vec{u} = 0$ constraint is replaced by the pressure equation. On the no-slip boundary, the boundary condition for the pressure is first proposed by [36]. Recently, motivated by the work of [25], [34] generalizes the PPE formulation to the case when part of or the whole boundary is of open or traction type. A Dirichlet-type boundary condition for the pressure is prescribed on the open boundary $\Gamma_2$:

\begin{align}
\Delta p &= \nabla \cdot (\vec{f} - \vec{u} \cdot \nabla \vec{u}) \quad \text{in } \Omega, \\
\partial_n p &= \vec{n} \cdot (\vec{f} - \vec{u} \cdot \nabla \vec{u} - \partial_t \vec{g}_1) - \nu \vec{n} \cdot \nabla \times \nabla \times \vec{u} \quad \text{on } \Gamma_1, \\
p &= \nu \vec{n} \cdot (\partial_n \vec{u}) - \nu \nabla \cdot \vec{u} - \vec{n} \cdot \vec{g}_2 \quad \text{on } \Gamma_2,
\end{align}

with $\vec{n}$ being the outward normal vector on $\Gamma_1$ and $\Gamma_2$. One should use $p = \nu \vec{n}^T (\nabla \vec{u} + \nabla \vec{u}^T) - \nu \nabla \cdot \vec{u} - \vec{n} \cdot \vec{g}_2$ on $\Gamma_2$ if the velocity boundary condition on $\Gamma_2$ is (1.5) instead of (1.4). The resulting system $\{(1.1), (1.3), (1.4), (2.1), (2.2), (2.3)\}$ is the so-called PPE formulation of the NSE, which contains a momentum equation and a pressure equation. Note that even though the divergence-free restriction disappears in the PPE formulation, it will be automatically satisfied if the initial velocity is divergence-free. The reason for this is because if $\vec{u}$ satisfies (1.1)--(1.4) and $p$ satisfies (2.1)--(2.3), then one can show that $w = \nabla \cdot \vec{u}$ satisfies $\partial_t w = \nu \Delta w$, $\partial_n w|_{\Gamma_1} = 0$, and $w|_{\Gamma_2} = 0$. So $w = 0$ as long as it is zero initially. Please note that the pressure boundary condition in (2.2) contains higher-order derivatives of computed solutions, and hence one needs to be more careful in computing these quantities than the boundary conditions found in projection methods (see (2.14)).

Equations (2.1)--(2.3) imply that the pressure at time $t$ is uniquely determined by the velocity at the same instant. Note that there is no time lag in (2.1)--(2.3) that determines $p(t)$ by $\vec{u}(t)$. If we denote the solution operator by $P$, then (2.1)--(2.3) can be written as

\begin{equation}
p(t) = P(\vec{u}(t), \vec{f}, \vec{g}_1, \vec{g}_2).
\end{equation}

Plugging (2.4) into (1.1), the PPE formulation can be written as

\begin{align}
\partial_t \vec{u} + \vec{u} \cdot \nabla \vec{u} + \nabla P(\vec{u}(t), \vec{f}, \vec{g}_1, \vec{g}_2) + \nu \Delta \vec{u} + \vec{f} &= 0 \quad \text{in } \Omega, \\
\vec{u} &= \vec{g}_1 \quad \text{on } \Gamma_1, \\
\nu \partial_n \vec{u} &= P(\vec{u}(t), \vec{f}, \vec{g}_1, \vec{g}_2)\vec{n} + \vec{g}_2 \quad \text{on } \Gamma_2.
\end{align}

The building block of KDC accelerated MoL$^T$ discretization is the following low-order stable scheme, which will be applied in each substep in the final deferred cor-
rection scheme: Given \( \bar{u}^n \), we first solve for \( p^n \) from the PPE

\[ \frac{\Delta p^n}{\Delta t} = \nabla \cdot (\tilde{f}^n - \bar{u}^n \cdot \nabla \bar{u}^n) \quad \text{in } \Omega, \]

\[ n \cdot \nabla p^n = n \cdot (\tilde{f}^n - \bar{u}^n \cdot \nabla \bar{u}^n - \partial_t \bar{g}_1(t_n)) - \nu \bar{u}^n \cdot \nabla \times \nabla \times \bar{u}^n \quad \text{on } \Gamma_1, \]

\[ p^n = \nu \bar{u} \cdot (\partial_t \bar{u}^n) - \nu \nabla \cdot \bar{u}^n - \bar{u} \cdot \bar{g}_2 \quad \text{on } \Gamma_2. \]

Then we calculate \( \bar{u}^{n+1} \) by the momentum equation

\[ \frac{1}{\Delta t} \left( \bar{u}^{n+1} - \bar{u}^n \right) + \nabla p^n = \nu \Delta \bar{u}^{n+1} + \tilde{f}^{n+1} - \bar{u}^n \cdot \nabla \bar{u}^n \quad \text{in } \Omega, \]

\[ \bar{u}^{n+1} = \bar{g}_1^{n+1} \quad \text{on } \Gamma_1, \]

\[ \nu \partial_n \bar{u}^{n+1} = p^n \bar{n} + \bar{g}_2^{n+1} \quad \text{on } \Gamma_2. \]

This first-order stable semi-implicit scheme was proposed and studied in [34]. We stress once again that both pressure and nonlinear terms are treated explicitly in the momentum equation. In particular, the updates of pressure and velocity are decoupled. Following [25], it is proved in [34] that the above scheme is unconditionally stable for Stokes equations in strip \( \Omega = [-1,1] \times (0,2\pi) \), where we have periodic boundary conditions in the \( y \) direction, no-slip or open boundary conditions on \( \{ x = -1 \} \), and no-slip or open boundary conditions on \( \{ x = 1 \} \). Moreover, using the ideas in [25], [34] was able to use \( C^0 \) finite elements to do the spatial discretization even though there are second-order derivatives in the boundary condition (2.9): Define

\[ Z_0 = \{ \bar{v} \in H^1(\Omega; \mathbb{R}^m); \bar{v}|_{\Gamma_1} = 0 \}, \quad Y_0 = \{ \phi \in H^1(\Omega); \phi|_{\Gamma_2} = 0 \}, \]

let \( Z_h \) and \( Y_h \) be the \( C^0 \) finite element spaces for velocity and pressure, and let \( Z_{0,h} = Z_h \cap Z_0, Y_{0,h} = Y_h \cap Y_0 \). Given \( \bar{u}^n_h \in Z_h \), we first calculate \( p^n_h \in Y_h \) by solving

\[ \langle \nabla p^n_h, \nabla \phi_h \rangle = \left( \langle \tilde{f}^n - \bar{u}^n_h \cdot \nabla \bar{u}^n_h, \nabla \phi_h \rangle - \langle \bar{n} \cdot \partial_t \bar{g}_1(t_n), \phi_h \rangle \right)_{\Gamma_1} \]

\[ + \nu \langle \nabla \times \bar{u}^n_h, \bar{n} \times \nabla \phi_h \rangle_{\Gamma_1}, \]

for any \( \phi_h \in Y_{0,h} \). To solve for \( p^n_h \), we also need the boundary condition \( p^n_h = \nu r^n_h - \bar{n} \cdot \bar{g}_2 \) on \( \Gamma_2 \), where \( r^n_h \) is the projection of \( \bar{n} \cdot (\partial_n \bar{u}^n_h) - \nabla \cdot \bar{u}^n_h \) \( |_{\Gamma_2} \) onto \( Y_h \) \( |_{\Gamma_2} \). Once we get \( p^n_h \), we calculate a preliminary \( \bar{u}^{n+1,*} \in Z_h \) by solving

\[ \frac{1}{\Delta t} \left( \langle \bar{u}^{n+1,*}_h, \bar{v}_h \rangle - \langle \bar{u}^n_h, \bar{v}_h \rangle \right) - \langle p^n_h, \nabla \cdot \bar{v}_h \rangle + \nu \langle \nabla \bar{u}^{n+1,*}_h, \nabla \bar{v}_h \rangle \]

\[ = \langle \tilde{f}^{n+1} - \bar{u}^n_h \cdot \nabla \bar{u}^n_h, \bar{v}_h \rangle + \langle \bar{g}_2^{n+1}, \bar{v}_h \rangle \] \( \Gamma_2 \)

for any \( \bar{v}_h \in Z_{0,h} \) with boundary condition \( \bar{u}^{n+1,*}_h = \bar{g}_1^{n+1} \) on \( \Gamma_1 \). Finally, we calculate \( q^{n+1}_h \in Y_{0,h} \) by solving

\[ \langle \nabla q^{n+1}_h, \nabla \phi_h \rangle = - \langle \nabla \cdot \bar{u}^{n+1,*}_h, \phi_h \rangle \]

for any \( \phi_h \in Y_{0,h} \) and obtain \( \bar{u}^{n+1}_h \in Z_h \) as the \( L^2 \) projection of \( \bar{u}^{n+1,*}_h - \nabla q^{n+1}_h \) onto the finite element space \( Z_h \), namely

\[ \langle \bar{u}^{n+1}_h, \bar{v}_h \rangle = \langle \bar{u}^{n+1,*}_h - \nabla q^{n+1}_h, \bar{v}_h \rangle \]
for any \( \bar{v}_h \in Z_h \). Please note that (2.16)–(2.17) means that we are doing discrete Leray projection to further suppress the divergence error [32]. This step is necessary in the benchmark computations when the viscosity is small and hence the solution is not very regular, but can be neglected when the solution is smooth. We like to point out that \( q_h^{n+1} \in Y_{0,h} = \{ q_h \in Y_h \subset H^1(\Omega), q_h|_{\Gamma_2} = 0 \} \) means that, on the open boundary, the tangential velocity instead of the normal velocity is preserved in the Leray projection of \( \bar{u}^{n+1,*} \). The boundary condition for \( q_h^{n+1} \) is perfectly physical since for the exact solution \( q \equiv 0 \). Using Dirichlet boundary conditions for \( q_h^{n+1} \) on \( \Gamma_2 \) is necessary for two reasons: (I) If not, the resulting Poisson equation for \( q_h^{n+1} \), (2.16), might be not consistent since numerically we might not have exactly \( \langle \nabla \cdot \bar{u}_h^{n+1,*}, 1 \rangle = \int_{\Gamma_2} \bar{n} \cdot \bar{u}_h^{n+1,*} = 0 \) as the velocity \( \bar{u}_h^{n+1,*} \) is no longer prescribed on the whole boundary. (II) Because now \( p_h^{n+1} \) and \( q_h^{n+1} \) have the same type of boundary conditions on \( \Gamma_1 \) and \( \Gamma_2 \), we can combine the two Poisson equations for them into one equation and solve once instead of twice per time step. The computational cost is then similar to the original projection method of Chorin and Temam. For that, we introduce

\[
\bar{p}_h^n = p_h^n - \frac{1}{\Delta t} q_h^n,
\]

which is obtained from solving

\[
\langle \nabla \bar{p}_h^n, \nabla \phi_h \rangle = \left\langle \bar{f}^n - \bar{u}_h^n \cdot \nabla \bar{u}_h^n, \nabla \phi_h \right\rangle + \frac{1}{\Delta t} \left\langle \nabla \cdot \bar{u}_h^n, \phi_h \right\rangle_{\Gamma_1} + \nu \left\langle \nabla \times \bar{u}_h^n, \bar{n} \times \nabla \phi_h \right\rangle_{\Gamma_1}
\]

for any \( \phi_h \in Y_{0,h} \). Dirichlet boundary condition \( p_h^n = \nu r_h^n - \bar{n} \cdot \bar{g}_{2} \) is enforced on \( \Gamma_2 \). Then we solve \( \bar{u}_h^{n+1} \) from the momentum equation

\[
\frac{1}{\Delta t} \left( \left\langle \bar{u}_h^{n+1}, \bar{v}_h \right\rangle - \left\langle \bar{u}_h^n, \bar{v}_h \right\rangle - \left\langle \bar{p}_h^n, \nabla \cdot \bar{v}_h \right\rangle + \nu \left\langle \nabla \bar{u}_h^{n+1}, \nabla \bar{v}_h \right\rangle \right) = \left\langle \bar{f}_h^{n+1} - \bar{u}_h^n \cdot \nabla \bar{u}_h^n, \bar{v}_h \right\rangle + \left\langle \bar{g}_{2}^{n+1}, \bar{v}_h \right\rangle_{\Gamma_2}
\]

for any \( \bar{v}_h \in Z_{0,h} \). Dirichlet boundary condition \( \bar{u}_h^{n+1} = \bar{g}_1^{n+1} \) is enforced on \( \Gamma_1 \). Please note that the \( \bar{u}_h^{n+1} \) in (2.20) is the \( \bar{u}_h^{n+1,*} \) in (2.15) and we have dropped the * as there are no more intermediate velocities. Besides the usage of \( \bar{p}_h^n \) or \( p_h^n \), the only difference left between (2.19)–(2.20) and (2.14)–(2.17) is whether the velocity before or after the Leray projection is used in \( \bar{u} \cdot \nabla \bar{u} \) and \( r := \bar{n} \cdot (\partial_u \bar{u}) - \nabla \cdot \bar{u} \big|_{\Gamma_2} \). But the velocity accuracy remains the same. If one wants to recover the pressure with full accuracy, one can use (2.14). See [33] for more related discussions.

We would like to point out that it is confirmed by various benchmark tests [32, 33, 34] that the above \( C^0 \) finite element scheme does not require the inf-sup compatibility condition between the finite element spaces for velocity and pressure. When \( \Gamma_2 = \emptyset \), see [32] for the proof of this claim of related schemes that use \( C^1 \) finite elements for velocity. In our finite element computations, we simply use \( P_h/P_h \) elements.

3. KDC accelerated MoL\(^7\). The basic idea of deferred correction-type methods is to first divide a time step into several smaller substeps and then to compute a provisional solutions on these sub-time steps. An equation for error/defect is then formed and solved using a lower-cost method, which is often a lower-order method, and the numerical results are used to improve the provisional solutions iteratively.
The frame work of KDC accelerated MoL\(^T\) was introduced in [23]. The MoL\(^T\) is not new, but is much less commonly known when compared to the traditional method of lines (MoL). It has been previously studied by different groups and sometimes is referred to as the transversal line method or Rothe’s method. MoL\(^T\) works by first discretizing the temporal direction and leads to a system of elliptic equations, in contrast to a differential algebraic equation (DAE)/ODE system for the MoL. This approach is also inspired by the development of highly efficient elliptic solvers in the past twenty years, such as the fast multipole method (FMM) [12], multiwavelet method [2], and so on.

In the numerical examples section, we demonstrate the KDC accelerated MoL\(^T\) for NSE with collocative spectral and finite element methods in space. The grid is fixed, and in this case MoL and MoL\(^T\) do not make much difference in the terms of the resulting algebraic system, except the order of discretization. However, they will make a difference as soon as the grid becomes adaptive in time or one wants to solve the underlying elliptic equations using a sophisticated solver like FMM. One goal of this paper is to demonstrate this idea and motivate investigation of the possibilities of integrating KDC with other fast efficient elliptic solvers. For detail of the MoL\(^T\), please refer to [23] and the references there.

In the following, we briefly describe the methodology as well as its application to the NSE.

### 3.1. Temporal discretization and spectral accuracy in time.

Suppose we are given a general parabolic PDE of the form

\[
\mathcal{L}(\partial_t u, u, Du, D^2u) = 0.
\]

For example, NSE (2.5)–(2.7) can be written as \(\partial_t \vec{u} - \mathcal{F}(\vec{u}, D\vec{u}, D^2\vec{u}) = 0\). To match the evolutionary PDE (3.1) from \(t = t_0\) to \(t = t_0 + \Delta t\), we first discretize the temporal direction using \(q\) Gaussian-type nodes, \([t_1, t_2, \ldots, t_q]\). If we know the values of \(\partial_t u = a \) at \([t_1, t_2, \ldots, t_q]\), which are denoted by \(a_1, \ldots, a_q\), and if we know \(u(t_0) = u_0\), we can approximate \(u(t)\) by \(u_0 + \int_{t_0}^{t} a(s)ds\), where \(a(s)\) is the Lagrange interpolation \(a(s) = \sum_{i=1}^{q} a_i \ell_i(s)\) with \(\ell_i(s) = \prod_{j \neq i} (s - t_j)/(t_i - t_j)\). Plugging \(a(s) = \sum_{i=1}^{q} a_i \ell_i(s)\) into \(\int_{t_0}^{t} a(s)ds\) and letting \(t \in \{t_1, \ldots, t_q\}\), we find that we can approximate \(u(t_k)\) for \(k = 1, \ldots, q\) by

\[
u_0 + \int_{t_0}^{t_k} a(s)ds = u_0 + \int_{t_0}^{t_k} \sum_{i=1}^{q} a_i \ell_i(s)ds = u_0 + \sum_{i=1}^{q} S_{ki} a_i,
\]

where \(S_{ki} = \int_{t_0}^{t_k} \ell_i(s)ds\). We mention in passing that, given \(q\), we can first calculate \(S_{ki}^{[0,1]}\) with \([t_1, \ldots, t_q]\) being the Gaussian-type nodes on \([0,1]\); then the \(S_{ki}\) we used in (3.2) is simply \(\Delta t S_{ki}^{[0,1]}\). Going back to (3.1), if we define the spectral integration matrix \(S = (S_{ki}) \in \mathbb{R}^{q \times q}, a = [a_1, \ldots, a_q]^\top\), and \(u_0 = [u_0, \ldots, u_0]^\top\), very naturally we would like to require the \(q\) unknowns \(\{a_1, \ldots, a_q\}\) to satisfy the following \(q\) equations:

\[
\mathcal{L}(a_k, u_0 + [Sa]_k, D(u_0 + [Sa]_k), D^2(u_0 + [Sa]_k)) = 0 \quad \text{for } k = 1, \ldots, q,
\]

where \([Sa]_k\) is the \(k\)th entry of the \(q \times 1\) vector \(Sa\).

Note that the solution to (3.3) gives spectral accuracy in time since the polynomial interpolation \(u(t) = u_0 + \int_{t_0}^{t} a(s)ds\) satisfies (3.1) at \(t = t_1, \ldots, t_q\). Please refer to
Consider the system of ODEs \( \dot{y} = f(t, y) \) on interval \([t_0, t_0 + \Delta t]\) with initial condition \( y(t_0) = y_0 \). Let \( t_0 \leq t_1 < \cdots < t_q \leq t_0 + \Delta t \). The collocation polynomial \( v(t) \) is a polynomial of degree \( q \) satisfying

\[
\begin{align*}
    v(t_0) &= y_0, \\
    \dot{v}(t_k) &= f(t_k, v(t_k)), \quad k = 1, \ldots, q.
\end{align*}
\]

The collocation method for ODE \( \dot{y} = f(t, y) \), \( y(t_0) = y_0 \) is defined by

\[
y_1 = v(t_0 + \Delta t).
\]

**Theorem 3.2.** If the \( t_1, \ldots, t_q \) in Definition 3.1 are the quadrature nodes of a quadrature on \([t_0, t_0 + \Delta t]\) and if the degree of precision of the quadrature is \( m - 1 \), then the order of the collocation method is \( m \); namely,

\[
y_1 - y(t_0 + \Delta t) = O(\Delta t^m + 1).
\]

**3.2. Application of KDC accelerated MoL to NSE.** Let \( \tilde{a} = [\tilde{a}_1, \ldots, \tilde{a}_q]^{\top} \in \mathbb{R}^{q \times m} \) be the approximation of \( \partial_t \tilde{u} \) at time \( t_1, \ldots, t_q \), where \( \tilde{a}_i \) is the acceleration vector. Let \( [S\tilde{a}]_k \) denote the \( k \)th row of the \( q \times m \) matrix \( S\tilde{a} \); \( \tilde{u} \) at time \( t_k \) is represented by \( \tilde{u}_0 + [S\tilde{a}]_k \). Equation (3.3) in the NSE case is written out more precisely as

\[
\begin{align*}
    \tilde{a}_k + (\tilde{u}_0 + [S\tilde{a}]_k) \cdot \nabla(\tilde{u}_0 + [S\tilde{a}]_k) + \nabla P(\tilde{u}_0 + [S\tilde{a}]_k, \tilde{f}, \tilde{g}_1, \tilde{g}_2) \\
    = \nu \Delta(\tilde{u}_0 + [S\tilde{a}]_k) + \tilde{f}(t_k) \quad \text{in } \Omega,
\end{align*}
\]

\[
\begin{align*}
    \tilde{u}_0 + [S\tilde{a}]_k & = \tilde{g}_1(t_k) \quad \text{on } \Gamma_1, \\
    \nu \partial_n(\tilde{u}_0 + [S\tilde{a}]_k) & = P(\tilde{u}_0 + [S\tilde{a}]_k, \tilde{f}, \tilde{g}_1, \tilde{g}_2) \tilde{n} + \tilde{g}_2(t_k) \quad \text{on } \Gamma_2
\end{align*}
\]

for \( k = 1, \ldots, q \), where the operator \( P \), defined by (2.4), is the solution of the Poisson equation (2.1)–(2.3).

Due to the fact that all the components of \( \tilde{a} \) are coupled together by the dense matrix \( S \), the numerical solution to such a huge nonlinear system could be very expensive. For efficient solutions, we apply the KDC method.

Suppose we are given a provisional solution \( \tilde{\alpha} = [\tilde{a}_1, \ldots, \tilde{a}_q]^{\top} \in \mathbb{R}^{q \times m} \), and define \( \tilde{v}_k = \tilde{u}_0 + [S\tilde{a}]_k \) and \( \tilde{\delta} = [\tilde{g}_1, \ldots, \tilde{g}_2]^{\top} = \tilde{a} - \tilde{\alpha} \); a substitution yields the error equation

\[
\begin{align*}
    \tilde{a}_k + (\tilde{v}_k + [S\tilde{\delta}]_k) \cdot \nabla(\tilde{v}_k + [S\tilde{\delta}]_k) + \nabla P(\tilde{v}_k + [S\tilde{\delta}]_k, \tilde{f}, \tilde{g}_1, \tilde{g}_2) \\
    = \nu \Delta(\tilde{v}_k + [S\tilde{\delta}]_k) + \tilde{f}(t_k) \quad \text{in } \Omega,
\end{align*}
\]

\[
\begin{align*}
    \nu \partial_n(\tilde{v}_k + [S\tilde{\delta}]_k) & = P(\tilde{v}_k + [S\tilde{\delta}]_k, \tilde{f}, \tilde{g}_1, \tilde{g}_2) \tilde{n} + \tilde{g}_2(t_k) \quad \text{on } \Gamma_2
\end{align*}
\]

for \( k = 1, \ldots, q \).

We introduce the following two types of approximation of \( S \): (I) we can approximate \( \int_{t_0}^{t_k} \delta(s) \, ds \) by a rectangular rule using the left end point, which is equivalent to the explicit Euler’s method:

\[
[S\delta]_k = \int_{t_0}^{t_k} \delta(s) \, ds \approx \sum_{j=1}^{k} \Delta t_j \delta_{j-1} = \sum_{j=2}^{k} \Delta t_j \delta_{j-1} = [S_{ex}\delta]_k.
\]
Note that we have $\delta_0 = 0$. In (3.13), we have used $\Delta t_j = t_j - t_{j-1}$ and introduced the $q \times q$ matrix

$$
S_{ex} = \begin{bmatrix}
0 & 0 & \cdots & 0 & 0 \\
\Delta t_2 & 0 & \cdots & 0 & 0 \\
\Delta t_2 & \Delta t_3 & \cdots & 0 & 0 \\
\Delta t_2 & \Delta t_3 & \cdots & \Delta t_q & 0
\end{bmatrix}.
$$

We can also approximate $\int_{t_0}^{t_k} \delta(s)ds$ by rectangular rule using the right end point, which is equivalent to the implicit Euler method:

$$
|S\delta|_k = \int_{t_0}^{t_k} \delta(s)ds \approx \sum_{j=1}^{k} \Delta t_j \delta_j = |S_{im}\delta|_k,
$$

where

$$
S_{im} = \begin{bmatrix}
\Delta t_1 & 0 & \cdots & 0 & 0 \\
\Delta t_1 & \Delta t_2 & \cdots & 0 & 0 \\
\Delta t_1 & \Delta t_2 & \cdots & \Delta t_{q-1} & 0 \\
\Delta t_1 & \Delta t_2 & \cdots & \Delta t_{q-1} & \Delta t_q
\end{bmatrix}.
$$

Motivated by the stable first-order semi-implicit scheme discussed in section 2, we rewrite the error equation as follows:

$$
\tilde{a}_k + \delta_k = (\tilde{v}_k + [S_{ex}\delta]_k) \cdot \nabla(\tilde{v}_k + [S_{ex}\delta]_k) + \nabla P(\tilde{v}_k + [S_{ex}\delta]_k, \tilde{f}, \tilde{g}_1, \tilde{g}_2) + \nu \Delta(\tilde{v}_k + [S_{im}\delta]_k) + \tilde{f}(t_k)
$$

in $\Omega$,

$$
\tilde{v}_k + [S_{im}\delta]_k = \tilde{g}_1(t_k) \quad \text{on } \Gamma_1,
$$

$$
\nu \partial_n(\tilde{v}_k + [S_{im}\delta]_k) = P(\tilde{v}_k + [S_{ex}\delta]_k, \tilde{f}, \tilde{g}_1, \tilde{g}_2)\tilde{n} + \tilde{g}_2(t_k) \quad \text{on } \Gamma_2,
$$

for $k = 1, \ldots, q$. Recalling that $S_{ex}$ and $S_{im}$ in (3.14) and (3.16) are strictly lower and lower triangular, the system is decoupled, and we solve for $\delta_1, \ldots, \delta_q$ one by one.

The procedure of solving (3.17)–(3.19) for $\delta$ also implicitly defines the mapping $\tilde{a} \mapsto \tilde{\delta}$:

$$
\tilde{\delta} = H(\tilde{a}).
$$

Note that when $\tilde{\delta} = H(\tilde{a}) = 0$, $\tilde{a}$ solves (3.7)–(3.9), which gives us a temporally spectrally accurate solution to the NSE.

This treatment not only decouples the equations but also facilitates the efficient applications of Krylov methods. Recall that if variable $\delta$ depends on variable $a$ implicitly by satisfying the equation $\ell(a + \delta) = 0$, then $\frac{d\delta}{da} = -\frac{\ell}{\tau} = -1$. So the Jacobian of the mapping $\tilde{a} \mapsto \tilde{\delta}$ (see (3.20)), which is denoted by $J_H$, is close to $-I$. It is not exactly $-I$ because (3.17)–(3.19) is not exactly of the form $\ell(a + \delta) = 0$. System (3.17)–(3.19) is an approximation of (3.10)–(3.12), and the latter is exactly of the type $\ell(a + \delta) = 0$. Nevertheless, $J_H \approx -I$ is enough to guarantee that we can efficiently apply the Newton–Krylov method to find a zero of the function $H(\tilde{a})$. For detail on the Newton–Krylov method, please refer to [28, 29]. The Newton–Krylov method
can also approximate the Jacobian by a forward difference, which means we need only to know how to evaluate $H(\tilde{a})$, which is equivalent to solving $\delta$ from (3.17)–(3.19) given $\tilde{a}$. This procedure is described in section 2.

Remark: The first-order scheme in section 2 uses a lagged pressure to compute the velocity; however, in the deferred correction methods, the first-order scheme is used to compute the correction; i.e., only the corrections are lagged. Therefore, when the solution converges, the correction/error goes to zero, and we obtain (3.7)–(3.9) which contains no lagging at all.

3.3. Divergence suppression. As we mentioned in section 2, we need some extra help from the discrete Leray projection (2.16)–(2.17) to further suppress the divergence error in our benchmark problem computations. We follow the idea in (2.14)–(2.17). For any $k \in \{1, 2, \ldots, q\}$, we first use (3.17)–(3.19) to calculate a provisional $\tilde{v}_k + [S_{im}\tilde{\delta}]_k$, then we do the discrete Leray project of this $\tilde{v}_k + [S_{im}\tilde{\delta}]_k$ just like (2.16)–(2.17), and from that we obtain the final $\tilde{v}_k + [S_{im}\tilde{\delta}]_k$ and hence $\tilde{\delta}_k$. This $\tilde{\delta}_k$ is then stored and used in the successive steps. All of our finite element computations use this technique. Alternatively, to eliminate the Poisson solver for the Leray projection step, we can modify the finite element equation for $P(\tilde{v}_k + [S_{ex}\tilde{\delta}]_k, \tilde{f}, \tilde{g}_1, \tilde{g}_2)$ like in (2.19) by adding the additional divergence damping: \[ \Delta t \langle \nabla \cdot (\tilde{v}_k + [S_{ex}\tilde{\delta}]_k), \phi_h \rangle. \]

4. Numerical examples. In this section, we present numerical tests of our scheme (3.17)–(3.19) and its variant that includes the discrete Leray projection mentioned at the end of last section. The first test is a temporal accuracy check of the original (3.17)–(3.19) with known exact solution in three dimensions where collocative spectral methods in space are used [42, 38]. The result indicates that the final scheme is fully spectrally accurate in the space-time domain. Then we did stability tests using both collocative spectral and finite element discretization in space. In the end, the results of flow past a cylinder and flow in a bifurcated tube are presented. In all the finite element calculations, we used equal-order isoparametric Lagrange finite elements for both velocity and pressure which are known to not satisfy the inf-sup compatibility condition.

The current codes were implemented in MATLAB. We chose Gaussian quadratures for KDC accelerated MoL T temporal discretization and utilized the Newton–BiCGStab solver by Kelley [28, 29], which provides a forward difference approximation for the Jacobian and adaptive linear tolerance control based on the modified Eisenstat–Walker formula. For detail please refer to the references above. The only input that needs to be prescribed by users is the tolerance. In all the numerical tests, we required a 0 absolute tolerance (equivalent to turning it off) with a desired relative tolerance per test.

The finite element package we used is in some sense an upgraded version of iFEM due to Chen [4]. Based on MATLAB, iFEM is an adaptive piecewise linear finite element package and provides efficient MATLAB subroutines to manipulate the mesh (e.g., see [5, 1]). In particular, local refinement and coarsening can be done fairly easily. For our purposes, we have extended it to isoparametric Lagrange elements up to P4. The finite element mesh is generated using DistMesh of Persson and Strang [37]. The contour plots on unstructured meshes are generated by the MATLAB routine tricontour.m due to Engwirda [8].

4.1. Exact solution example with pseudospectral method. In this example, the computational domain is $[0, \pi]^3$, and the force $f$ is selected so that the exact
solution is \( \vec{u} = (u_{ex}, v_{ex}, w_{ex}) \) with zero Dirichlet boundary conditions:

\[
\begin{align*}
(4.1) \quad u_{ex} &= \cos(t) \sin^2(x)(\sin(2y) \sin^2(z) - \sin^2(y) \sin(2z)), \\
(4.2) \quad v_{ex} &= \cos(t) \sin^2(y)(\sin(2z) \sin^2(x) - \sin^2(z) \sin(2x)), \\
(4.3) \quad w_{ex} &= \cos(t) \sin^2(z)(\sin(2x) \sin^2(y) - \sin^2(x) \sin(2y)), \\
(4.4) \quad p_{ex} &= \cos(t) \cos(x) \sin(y) \cos(z).
\end{align*}
\]

With 20 Chebyshev quadrature points in each spatial direction, the spatial errors are about \(10^{-10}\). Therefore, the errors in the figures are contributed mainly from the temporal direction. In the temporal accuracy check test, we set \(\nu = 1\) and the relative tolerance to \(10^{-12}\). As shown in Figure 4.1, using six Gaussian points per time step, we observed a convergence order about 9.3. The tests were run from \(t = 0\) to \(t = 6\) with time step sizes varying from 1 to 2. Note that we did not achieve the full order, which should be 12, and this phenomena is called order reduction. Order reductions of collocation methods for stiff ODE systems and DAEs have been observed as described in [17, 19], and the order of the convergence for KDC accelerated MoL\(^T\) is still the subject of ongoing research. For more details, please refer to [23].

As shown in Figure 4.2, for a fixed time step size 1, the error decays exponentially as the number of Gaussian points used per step increases, which demonstrates the spectral accuracy in time.

To test the stability of the current method, we chose \(\nu = 1/20\) and used a fixed 20 × 20 × 20 mesh. The relative tolerance was set to \(10^{-10}\). The scheme could integrate...
Fig. 4.2. Results with 3–7 Gaussian points used per time step.

to $T = 1000$ with $\Delta t = 1$, while maintaining an $L^\infty$ error of the velocity fields to be around $10^{-5}$. In this test, we also used six Gaussian points per time step.

Figures 4.3 and 4.4 show the comparisons of elapsed CPU time in MATLAB by different time stepping methods when we integrate from $t = 0$ to $t = 3$. As we can see, the KDC accelerated MoL provides a fairly decent speedup for the first-order semi-implicit method and the SDC methods, especially when higher accuracy is required. The data were obtained by varying time step sizes. The step sizes used for the first-order method are [0.02, 0.002, 0.0005, 0.0001], from left to right in the figures; [3, 1, 0.5, 0.3, 0.25] for 3-point SDC/KDC; [3, 1.5, 1, 0.75] for 4-point SDC/KDC. Please also note that in our SDC implementation, for every time step, the correction iteration is repeated until the desired tolerance is satisfied. This way, we make sure that the SDC converges to the underlying collocation formulation so that the accuracy argument for KDC in section 3.1 still applies. This is somewhat different from the original SDC where a fixed number of correction steps are performed.

In Figure 4.3, the relative tolerance was set to $10^{-8}$ for KDC and SDC, in spite of different step sizes. However, we note that for larger step sizes where the final solutions are not very accurate, setting tolerance to $10^{-8}$ is overkill. Figure 4.4 shows the comparison with a tuned relative tolerance setting for SDC and KDC. In these tests, the tolerance was set to one more digit than the corresponding accuracy of the final solution as in Figure 4.3. This figure is provided to show the potential performance boost of deferred correction-type methods through better tolerance setting. It is a part of the open research on optimal time-stepping of SDC/KDC.

In the following examples, we implemented finite element methods in space for two-dimensional problems.
Fig. 4.3. CPU time comparison. These test results were generated in single thread mode on a 2.2GHz AMD64 processor with 3G memory.

Fig. 4.4. CPU time comparison. Potential speedup with optimal tolerance.
4.2. Exact solution example with finite element method. This example was set up with the following exact solution:

$$u_{ex} = \cos(t) \cos^2(\pi x/2) \sin(\pi y),$$  \hspace{1cm} (4.5)

$$v_{ex} = -\cos(t) \sin(\pi x) \cos^2(\pi y/2),$$  \hspace{1cm} (4.6)

$$p_{ex} = \cos(t) \cos(\pi x/2) \sin(\pi y/2).$$  \hspace{1cm} (4.7)

The computational domain is $[-0.5, 0.5] \times [-0.5, 0.5] \setminus \{(x^2 + y^2 \leq 0.2^2\}$, and the finite element mesh is shown in Figure 4.5. The boundary condition, with a value of $\nu \partial_n \vec{u} - p \vec{n}$, is enforced at boundary $\{x = 0.5\}$. The rest of the boundary is a Dirichlet boundary. Table 4.1 shows that our scheme allows very large time steps stably taken with moderate $\nu$. We also tested smaller $\nu$ in the next example.

4.3. Flow past a cylinder. This benchmark test is from [26]. We followed the setup, and the domain is $[0, 2.2] \times [0, 0.41] \setminus \{(x - 0.2)^2 + (y - 0.2)^2 \leq 0.05^2\}$ (see Figure 4.6). $\nu$ was set to 1/1000. The relative tolerance was set to $10^{-7}$. The time dependent inflow profile

$$\vec{u}(t, 0, y) = 0.41^{-2} \sin(\pi t/8)(6y(0.41 - y), 0)$$  \hspace{1cm} (4.8)

is prescribed. The outflow boundary condition is $\nu \partial_n \vec{u} - p \vec{n} = 0$, which is enforced at outflow boundary $\{x = 2.2\}$. The computational mesh is shown in Figure 4.6, and the streamline plot at $t = [2, 4, 5, 6, 7, 8]$ is shown in Figure 4.7.

**Table 4.1**

The $-\log_{10}$ of the $L^\infty$ error of $\vec{u}$, using 4 Gaussian quadrature points per time step. $P^4$ isoparametric elements for both velocity and pressure. Mesh is fixed to be the one in Figure 4.5. $\nu = 1$. Integrate to $t = 1000$. The CFL number is estimated as $\max(|\vec{u}| \Delta t / h)$, and the von Neumann number is estimated as $\max(\nu \Delta t h^2)$, where $\max$ is taken over all the triangles and $h$ is the side length of the triangle.

<table>
<thead>
<tr>
<th>$\Delta t$</th>
<th>0.25</th>
<th>0.5</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-\log_{10} |\vec{u} - \vec{u}<em>h|</em>{L^\infty}$</td>
<td>3.9534</td>
<td>3.8814</td>
<td>2.8960</td>
<td>2.4227</td>
<td>1.2138</td>
<td>-0.0169</td>
</tr>
<tr>
<td>CFL #</td>
<td>8.860</td>
<td>17.720</td>
<td>35.439</td>
<td>70.878</td>
<td>141.756</td>
<td>283.513</td>
</tr>
<tr>
<td>von Neumann #</td>
<td>556</td>
<td>1111</td>
<td>2222</td>
<td>4444</td>
<td>8889</td>
<td>17778</td>
</tr>
</tbody>
</table>

**Fig. 4.5.** Mesh used in stability and temporal accuracy check.
Fig. 4.6. Mesh used in flow past a cylinder calculation when $\nu = 1/1000$.

Fig. 4.7. Flow past a cylinder with open boundary condition. $\nu = 1/1000$. 763 isoparametric $P^4$ elements (degrees of freedom (dof) $= 6322$) for each variable. $h_{\min} = 0.0082$. $h_{\max} = 0.117$. The velocity at $t = [2, 4, 5, 6, 7, 8]$. $\Delta t = 0.01$ with 4 Gaussian quadrature points.
The $x$ and $y$ components of the following quantity are calculated, which are denoted as $c_d(t)$ and $c_l(t)$, respectively the drag and lift coefficients:

\[
2 \int_S \nu \partial_n \vec{u} - p \vec{n},
\]

where $S$ is the surface of the cylinder. Since our goal was to test the scheme, we faithfully calculated the above quantity by surface integration, instead of transforming it into volume integration [26, equations (4), (5)] which is known to be more accurate. We calculated when the maxima of $c_d$ and $c_l$ occurred. These values are shown in Figure 4.8 together with the pressure difference between the front and the back of the cylinder,

\[
\Delta p(t) = p(t, 0.15, 0.2) - p(t, 0.25, 0.2).
\]

The maxima of $c_d$ and $c_l$ and when they occurred agree rather well with the reference values given in [26], even though [26] uses prescribed outflow profile at the outflow boundary \{x = 2.2\} while we use $\nu \partial_n \vec{u} - p \vec{n} = 0$. One notable difference is that at $t = 8$ the last eddy from the lower wall remains to the left of \{x = 2.2\} in [26], while it goes across \{x = 2.2\} in our current computation (see the last plot in Figure 4.7). But it seems that this difference in the downstream will not affect the flow around the cylinder since our results in Figure 4.8 agree well with results in [26].

In this benchmark test, the viscosity $\nu = 1/1000$ is fairly small, and a step size of 0.01 was taken stably while giving satisfactory solutions. This provides a notable improvement over the results in [33], where for the same problem with the same spatial discretization but a third-order multistep time-stepping, the solution bellows up if $\Delta t = 5 \times 10^{-4}$.

4.4. Flow past a bifurcated tube. For the test, the computational domain is

\[
\Omega = [0, 8] \times [-0.5, 0.5] \setminus \{(0, 0.5) \times [-0.5, 0] \cup [1.5, 8] \times [-0.1, 0.2]\}
\]

and $\nu = 1/600$. The computational mesh is shown in Figure 4.9.
At the inflow boundary \( \{ x = 0 \} \), we started from rest and gradually increased the boundary velocity \((u, v)\) to \((12y(1 - 2y), 0)\). The time dependent function we used for gradually increasing velocity is \((1 - \cos(\pi t))/2\) on \([0, 1]\). \(\nu \partial_n \vec{u} - p \vec{n} = 0\) was enforced at the outflow boundary \( \{ x = 8 \} \) which contains two parts: \( \{ x = 8, 0.2 \leq y \leq 0.5 \} \) (top branch) and \( \{ x = 8, -0.5 \leq y \leq -0.1 \} \) (bottom branch). Since the tube bifurcates, we have no way to determine a priori what the outflow profile for each branch is, and hence it is impossible to use schemes that require Dirichlet boundary conditions for velocity on the whole boundary. Nevertheless, we can prescribe open boundary conditions \(\nu \partial_n \vec{u} - p \vec{n} = 0\) on the outflow boundaries. Figure 4.10 shows the streamline of the flow at \( t = 20 \). We also calculated the outflux \( \int_\Gamma \vec{n} \cdot \vec{u}'s\) at \( \{ x = 8, 0.2 \leq y \leq 0.5 \} \) (top) and at \( \{ x = 8, -0.5 \leq y \leq -0.1 \} \) (bottom) and confirmed that their sum equals the influx at the \( \{ x = 0 \} \) boundary.

Acknowledgments. We thank Prof. Long Chen for the finite element package iFEM based on which the finite element calculations in this paper were done. We thank the referees for valuable comments on a previous version of this paper which greatly improved the presentation of this paper. All opinions expressed in this report are the authors’ and do not necessarily reflect policies and views of the U.S. Department of Energy or the Oak Ridge National Laboratory.

REFERENCES

References


[8] D. Engwirda, Contours for triangular grids, software package, MATLAB Central
do?objectId=10408.


[12] M. L. Minion, Open and traction boundary conditions for the incompressible Navier-Stokes


conditions for the incompressible Navier-Stokes equations, Internat. J. Numer. Methods

[21] J. Huang, J. Jia, and M. Minion, Accelerating the convergence of spectral deferred correction

[22] J. Huang, J. Jia, and M. Minion, Arbitrary order Krylov deferred correction methods for

[23] J. Jia and J. Huang, Krylov deferred correction accelerated method of lines transpose for

Constraints, Ph.D. thesis, Department of Mathematics, University of North Carolina at Chapel
Hill, Chapel Hill, NC, 2006.


Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.


