Fractional Time-Stepping Methods
for Unsteady Flow Problems

Seongjai Kim* and Jim Douglas, Jr.†

Abstract

We study accuracy of fractional time-stepping (FTS) methods such as the alternating direction implicit method for the computation of heat flows and the velocity-pressure splitting methods for Navier-Stokes flows. The FTS techniques can be viewed as a perturbation of conventional temporal-spatial discretization schemes; they in theory invoke a perturbation error that is on the same order as the discretization error. However, it has been observed in simulation that the perturbation error can be much (one-order) larger than the discretization error unless the timestep size is sufficiently small, which often degrades benefits of the FTS method. The article suggests variants of FTS methods in which the splitting error is on one-order higher than the discretization error and therefore negligible for moderate timestep sizes. The new FTS methods require a small fraction of additional computation cost over the original ones. Numerical verifications and applications for the numerical solution of the heat equation and unsteady Navier-Stokes equations are presented.

*Department of Mathematics, University of Kentucky, Lexington, KY 40506-0027 USA   Email: skim@ms.uky.edu
†Department of Mathematics, Purdue University, W. Lafayette, IN 47907 USA   Email: douglas@math.purdue.edu
Contents

1. Introduction 3

2. ADI for the heat equation 4
   2.1. Classical ADI method ........................................ 5
   2.2. Numerical accuracy of ADI ................................ 7

3. Accuracy improvement for ADI 9
   3.1. Correction term for ADI ..................................... 9
   3.2. Accuracy and efficiency of ADI-II ........................ 11
   3.3. Applications to elliptic problems .......................... 12

4. FTS techniques for Navier-Stokes equations 13
   4.1. Preliminaries .................................................. 13
   4.2. New FTS method ............................................... 15
   4.3. Finite volume method and iterative solvers ............... 16

5. Numerical experiments for NS equations 20

6. Conclusions 21
1. Introduction

The fractional time-stepping (FTS) method is rather popular in modern unsteady flow simulation, in particular, in higher dimensions. It introduces a certain degree of additional error called the splitting error, which in theory is the same order as (or, not larger than) the discretization error. However, one can easily observe in practical simulation that the splitting error is often much larger than expected. The article is concerned with strategies for FTS techniques with which the resulting algorithms can enjoy a better accuracy for the numerical solution of unsteady flow problems. We begin with a simple model problem.

Let $\Omega = (a_x, b_x) \times (a_y, b_y)$ be a domain in $\mathbb{R}^2$ with its boundary $\Gamma = \partial \Omega$ and $J = (0, T]$ be the time interval, $T > 0$. Consider the heat equation

\begin{align}
\text{(a)} & \quad u_t - \nabla \cdot (a \nabla u) + cu = f, \quad (x, t) \in \Omega \times J, \\
\text{(b)} & \quad au_{\nu} + bu = g, \quad (x, t) \in \Gamma \times J, \\
\text{(c)} & \quad u = u_0, \quad x \in \Omega, \ t = 0,
\end{align}

where the coefficients $a$, $c$, and $b$ are given functions, the subscript $\nu$ denotes the outer unit normal from $\Gamma$, $u_0$ is the prescribed solution for $t = 0$, and $f$ and $g$ are the sources/sinks.

Alternating direction implicit (ADI) methods have proved valuable in the approximation of the solutions of parabolic and elliptic differential equations in two and three variables. The original method [7, 14, 25], when applied to the heat equation of the form (1.1), is a perturbation of the Crank-Nicolson difference equation that incorporates a splitting error of $O(k^2)$, where $k$ is the temporal grid size. The original ADI method thus is second-order correct in both space and time. However, one can observe in simulation that the total error of ADI is often one-order larger than those of other non-perturbing solvers.

The first object of the paper is to suggest a numerical strategy for ADI so that it can recover the original numerical accuracy of the Crank-Nicolson difference equation. The basic idea is to add an appropriate correction term which turns the resulting perturbation error into $O(k^3)$, one-order higher than the discretization error. The new algorithm increases the computational cost only about 5-7\% over the original ADI method and recovers accuracy of the Crank-Nicolson difference equation for moderate timestep sizes, as shown in later sections.

Another object of the article is to apply the idea of adding-a-correction-term to velocity-pressure splitting algorithms for the numerical solution of unsteady, incompressible Navier-Stokes (NS) equations:

\begin{align}
\text{(a)} & \quad \nabla \cdot (\rho \mathbf{v}) = 0, \quad (x, t) \in \Omega \times J, \\
\text{(b)} & \quad \frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v} \mathbf{v}) = \nabla \cdot \mathbf{T} + \rho \mathbf{g}, \quad (x, t) \in \Omega \times J,
\end{align}
where $\rho$ is the density (mass per unit volume) which is independent of time but may vary in space, $v$ denotes the velocity (momentum per unit mass), and $g$ is the gravity vector. Here the stress tensor $T$, which is the molecular transport rate of momentum, reads

$$T = -pI + \tau, \quad \tau = 2\mu D + \left[\left(\kappa - \frac{2}{3}\mu\right)\nabla \cdot v\right]I,$$

where $p$ is the static pressure, $\tau$ is the viscous part of the stress tensor, $\mu$ and $\kappa$ are respectively the shear coefficient of viscosity and the bulk coefficient of viscosity, $I$ is the unit (identity) tensor, and $D$ is the rate of strain (deformation) tensor defined by

$$D = \frac{1}{2}\left(\nabla v + (\nabla v)^T\right).$$

We have assumed that the equations (1.2) are given appropriate boundary and initial conditions.

In simulating NS equations of the form (1.2) by employing a FTS technique, the velocity components and the pressure are decoupled in fractional steps. An iterative method (solving for velocity and pressure in turn) can be applied to get an accurate solution; however, it can be expensive. Moin and his colleagues [4, 5] suggested a direct FTS method having three fractional steps. Their algorithm can be viewed as a perturbation of a second-order method that incorporates the Crank-Nicolson method in time and the central finite difference (FD) discretization in space. The algorithm of Choi and Moin [4] introduces a second-order splitting error. In this article, we modify the algorithm to make the splitting error third-order correct and implement the resulting algorithm with the second-order finite volume (FV) discretization in space.

An outline of the paper is as follows. In §2, the classical ADI method is briefly reviewed and its perturbation error is investigated numerically. It has been found that the perturbation error can be one-order larger than the discretization error. §3 suggests an adding-a-correction-term variant of ADI, called the ADI with improved initial (ADI-II) of which the perturbation error is on a third-order. Its accuracy and efficiency are numerically verified, in the same section, and some efficiency issues are discussed for applications of ADI and ADI-II to elliptic problems. In §4, we apply the adding-a-correction-term strategy to velocity-pressure splitting algorithms for the NS equations. Algorithmic details are presented for the nonlinear iteration, FV discretizations, and elliptic solvers. §5 presents numerical results for NS flows. The last section includes concluding remarks.

2. ADI for the heat equation

ADI was first suggested by Douglas, Peaceman, and Rachford [7, 14, 25] for solving the heat equation in two spatial variables. The method was extended for mildly
nonlinear problems [8], three space variables [9], and nonsymmetric problems [22, 27]. A general formulation for ADI for parabolic and hyperbolic problems can be found in [13]. Pearcy [26] showed convergence of ADI without requiring commutativity of operators, for the first time. Optimum ADI parameters for the cycle length of the form $2^m$, $m \geq 0$, can be found in [32]. A collection of interesting results were presented applying ADI to finite element methods [11], p-version finite element methods [10], mixed finite element methods [12], and collocation methods [23, 30]. ADI has been applied to various physical problems [1, 2, 3, 6, 16, 18, 20, 21]. Parallelization of ADI has been tried in [19, 24, 27]. For variants and systematic descriptions for ADI, see Strikwerda [29, §7.3], Varga [31, Ch.7], and Wachspress [33].

2.1. Classical ADI method

In this subsection we review the classical ADI method applied to the heat equation of the form (1.1). Define

$$A_1 u = -(au_x)_x + \frac{1}{2} cu, \quad A_2 u = -(au_y)_y + \frac{1}{2} cu.$$ 

Then, the equation (1.1a) can be rewritten as

$$u_t + A_1 u + A_2 u = f.$$ (2.1)

Let the time interval $J = (0, T]$ be partitioned into $\{ 0 = t^0 < t^1 < \cdots < t^n = T \}$, where $t^n = n \cdot k$, $n = 0, 1, \cdots, n_t$, $k = T/n_t$. For partitions in the spatial directions, we choose $n_x + 1$ and $n_y + 1$ grid points equally distributed in $x$ and $y$ directions, respectively, as follows:

$$x_i = a_x + i \cdot h_x, \quad i = 0, 1, \cdots, n_x, \quad h_x = (b_x - a_x)/n_x,$$

$$y_j = a_y + j \cdot h_y, \quad j = 0, 1, \cdots, n_y, \quad h_y = (b_y - a_y)/n_y.$$ (2.2)

The temporal discretization employs the Crank-Nicolson scheme that is centering the difference scheme about $t = (n - \frac{1}{2})k$. By the Taylor series, (2.1) becomes

$$\frac{u^n - u^{n-1}}{k} + \frac{1}{2}(A_1 u^n + A_1 u^{n-1}) + \frac{1}{2}(A_2 u^n + A_2 u^{n-1}) = f^{n-1/2} + O(k^2)$$

or

$$\left( I + \frac{k}{2} A_1 + \frac{k}{2} A_2 \right) u^n = \left( I - \frac{k}{2} A_1 - \frac{k}{2} A_2 \right) u^{n-1} + kf^{n-1/2} + O(k^3),$$ (2.3)

where

$$f^{n-1/2} = \frac{1}{2}(f^n + f^{n-1}).$$

Now, we replace the operators $A_1$ and $A_2$ in (2.3) respectively by their spatial approximations $A_{1h}$ and $A_{2h}$, central FDs on the partition (2.2). Define

$$B_1 = \frac{k}{2} A_{1h} \quad \text{and} \quad B_2 = \frac{k}{2} A_{2h}.$$
Then, (2.3) becomes

\[(I + B_1 + B_2)u^n = (I - B_1 - B_2)u^{n-1} + kf^{n-1/2}.\]  \hspace{1cm} (2.4)

Here the local truncation error involved is \(\mathcal{O}(kh^2 + k^3)\); after a time integration over \(J\), the global error becomes \(\mathcal{O}(h^2 + k^3)\).

ADI begins with adding \(B_1 B_2 u^n\) to the both sides of (2.4). Then it reads

\[(I + B_1 + B_2 + B_1 B_2)u^n = (I - B_1 - B_2 + B_1 B_2)u^{n-1} + B_1 B_2(u^n - u^{n-1}) + kf^{n-1/2},\]  \hspace{1cm} (2.5)

which can be factored as

\[(I + B_1)(I + B_2)u^n = (I - B_1)(I - B_2)u^{n-1} + B_1 B_2(u^n - u^{n-1}) + kf^{n-1/2}.\]  \hspace{1cm} (2.6)

Note that

\[B_1 B_2(u^n - u^{n-1}) = \mathcal{O}(k^3),\]  \hspace{1cm} (2.7)

which is the same order as the local truncation error already introduced. Dropping the term in (2.7) from (2.6), we have

\[(I + B_1)(I + B_2)u^n = (I - B_1)(I - B_2)u^{n-1} + kf^{n-1/2}.\]  \hspace{1cm} (2.8)

The above perturbed problem can be easily solved as follows:

(a) \((I + B_1)u^* = (I - B_1)(I - B_2)u^{n-1} + kf^{n-1/2}, \) \hspace{1cm} (x-sweep)

(b) \((I + B_2)u^n = u^*, \) \hspace{1cm} (y-sweep)  \hspace{1cm} (2.9)

or equivalently [7, 14, 25]

(a) \((I + B_1)u^* = (I - B_2)u^{n-1} + \frac{k}{2}f^{n-1/2}, \) \hspace{1cm} (x-sweep)

(b) \((I + B_2)u^n = (I - B_1)u^* + \frac{k}{2}f^{n-1/2}. \) \hspace{1cm} (y-sweep)  \hspace{1cm} (2.10)

Theoretical aspects of the method were treated in detail in [7], while practical aspects of the calculation are considered in the companion paper [25] in considerable detail. Note that for each sweep the matrix to be inverted is tridiagonal. The algorithm requires \(\mathcal{O}(N)\) flops, where \(N = mn_x n_y\), the number of grid points.

Remark. Other ADI schemes can be derived starting with other basic schemes. For example, adopting the backward-time central-space scheme for (2.1), one has

\[(I + k A_{1h} + k A_{2h})u^n = u^{n-1} + kf^n + \mathcal{O}(k^2 + kh^2).\]  \hspace{1cm} (2.11)
Define
\[ C_1 = kA_{1h} \quad \text{and} \quad C_2 = kA_{2h}. \]
Adding \( C_1C_2u^n \) to the both sides of (2.11), and factoring its left side, we have
\[
(I + C_1)(I + C_2)u^n = (I + C_1C_2)u^{n-1} + kf^n + C_1C_2(u^n - u^{n-1}) + \mathcal{O}(k^2 + kh^2).
\]
Then, the corresponding ADI method reads [15]
\[
\begin{align*}
(I + C_1)u^* &= (I - C_2)u^{n-1} + kf^n, \\
(I + C_2)u^n &= u^* + C_2u^{n-1}.
\end{align*}
\tag{2.12}
\]
Here the local perturbation error is \( C_1C_2(u^n - u^{n-1}) = \mathcal{O}(k^3) \), which is one-order higher than the local discretization error in time.

Through this paper, the term “ADI” refers to (2.10) unless otherwise specified.

2.2. Numerical accuracy of ADI
ADI often shows unsatisfactory performances in terms of accuracy. More specifically speaking, the perturbation error arising from the ignored term (2.7) can be much larger than the discretization error.

In this section, we will numerically verify the above claim. Let’s start with choosing two different solutions:
\[
\begin{align*}
u(t, x, y) &= \sin(2\pi \nu_t t) + 2\pi \nu_x x + 2\pi \nu_y y, \\
\nu(t, x, y) &= \sin(2\pi \nu_t t) \cdot \sin(2\pi \nu_x x) \cdot \sin(2\pi \nu_y y),
\end{align*}
\tag{2.13}
\]
where \( \nu_t, \nu_x, \) and \( \nu_y \) are respectively the frequencies of the solution in \( t, x, \) and \( y \) directions. Set \( \Omega \times J = (0, 1)^2 \times (0, 1), a \equiv 1, \) and \( c = b \equiv 0 \) in (1.1). For the solution frequencies, choose \( \nu_t = \nu_x = \nu_y = 1. \) The sources \( f \) and \( g \) are set to satisfy (1.1). Uniform meshes are selected; \( n := n_t = n_x = n_y. \) To compare the computation cost and accuracy, we implemented three algorithms: an LU-based algorithm for (2.4), PCG-ILU0 for (2.4), and ADI (2.10). Here PCG-ILU0 denotes the conjugate gradient method preconditioned by the zero-level (not allowing fill-in) incomplete LU-factorization. For PCG-ILU0 in each time level, the initial point is set to be the extrapolation
\[ u^{n,0} = 2u^{n-1} - u^{n-2}, \quad n \geq 2, \]
and the iteration stops when the residual is reduced by \( 10^{-5} \). The main/driver routines are written in C++ and C, and the core routines in Fortran. Every computation is carried out on a 400 MHz laptop having 196M memory and a Linux operating system.

Table 1 presents the elapsed times and numerical errors for \( u_+ \) for various grid sizes. One can see from the table that the three different algorithms show the same
Table 1: The performances of the LU-based, PCG-ILU0, and ADI for \( u = u_+ \). For the computation, \( a \equiv 1 \), \( c = b \equiv 0 \), and \( \nu_t = \nu_x = \nu_y = 1 \). Meshes are set to be uniform: \( n := n_t = n_x = n_y \).

<table>
<thead>
<tr>
<th></th>
<th>( n = 40 )</th>
<th>( n = 80 )</th>
<th>( n = 160 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU ( L^2 )-error</td>
<td>CPU ( L^2 )-error</td>
<td>CPU ( L^2 )-error</td>
</tr>
<tr>
<td>LU-based</td>
<td>0.74 4.10e-3</td>
<td>9.07 1.00e-3</td>
<td>126 2.47e-4</td>
</tr>
<tr>
<td>PCG-ILU0</td>
<td>0.46 4.11e-3</td>
<td>5.67 1.00e-3</td>
<td>53.4 2.47e-4</td>
</tr>
<tr>
<td>ADI</td>
<td>0.26 4.10e-3</td>
<td>2.16 1.00e-3</td>
<td>17.9 2.47e-4</td>
</tr>
</tbody>
</table>

Table 2: The performances of the LU-based, PCG-ILU0, and ADI for \( u = u_\times \). The problem coefficients and algorithm parameters are chosen the same as in Table 1.

errors and their second-order convergence. The elapsed time (CPU) is measured in second and the error in \( L^2 \)-norm is evaluated at \( t = 1 \). Average 7-9 iterations were performed for PCG-ILU0 to converge in a timestep.

In Table 2, we present the results for \( u = u_\times \). The problem coefficients and algorithm parameters are chosen the same as in Table 1. The computation cost for ADI increases linearly as the number of grid points grows, while PCG-ILU0 shows a slight super-linearity in its computation cost, as one can expect. For marching a time level, average 12 to 22 iterations were required for PCG-ILU0 to converge, with the iteration counts becoming larger for more grid points. (The solution \( u_\times \) changes more than \( u_+ \) does.) In the table, the algorithms show the second-order accuracy. However, ADI produces approximately 34 times larger error than the LU-based or PCG-ILU0 for the same grid size. Take a look at the errors for \( n = 40 \) in LU-based/PCG-ILU0 and \( n = 160 \) in ADI. The number of grid points for ADI is 64 times larger than that for PCG-ILU0, but the error is still twice larger. In summary, ADI can produce a huge amount of perturbation error that degrades efficiency of the algorithm.

Note that the truncation error for the Crank-Nicolson difference equation is of the form

\[
O \left( h_x^2 \frac{\partial^4 u}{\partial x^4} \right) + O \left( h_y^2 \frac{\partial^4 u}{\partial y^4} \right) + O \left( k^2 \frac{\partial^3 u}{\partial t^3} \right),
\]
while the (global) perturbation error of ADI is

$$\mathcal{O}\left(k^2 \frac{\partial^2}{\partial x^2} \frac{\partial^2}{\partial y^2} \frac{\partial}{\partial t} u\right).$$

That is, roughly speaking, why ADI incorporates no perturbation error for $u_+$, and a large perturbation error for $u_\times$.

3. Accuracy improvement for ADI

This section presents a strategy to reduce the perturbation error of ADI and recover accuracy of the Crank-Nicolson difference equation, for a small fraction of the computation cost.

3.1. Correction term for ADI

We first recall that the perturbation arising in (2.8), out of (2.6), is

$$-B_1 B_2 (u^n - u^{n-1}).$$

It is reasonable to choose the following correction term

$$+B_1 B_2 (u^{n-1} - u^{n-2}), \quad n \geq 2.$$ (3.1)

Thus the total perturbation error becomes

$$-B_1 B_2 (u^n - u^{n-1}) + B_1 B_2 (u^{n-1} - u^{n-2})$$

$$= -B_1 B_2 (u^n - 2u^{n-1} + u^{n-2}) = \mathcal{O}(k^4),$$ (3.2)

which is one-order higher than the discretization error.

The corresponding ADI algorithm incorporating the correction term (3.1) reads

(a) \[ (I + B_1)u^* = (I - B_1 - B_2)u^{n-1} + k f^{n-1/2} \]

\[ + B_1 B_2 (2u^{n-1} - u^{n-2}), \]

(b) \[ (I + B_2)u^n = u^*, \]

which is obtained by adding the correction term in the right side of (2.9.a).

The following remarks are worth being considered.

Remark. Algorithm (3.3) requires information from two preceding time levels to advance in time. Thus a starting procedure is needed to define $u^1$ which may retain the overall accuracy of the method. One may set $u^{-1} = u^0$. In this case, (3.3) becomes (2.10) for $n = 1$. To get a better accuracy, one can apply a relaxation method such as SOR over the solution $u^1$ obtained from (3.3).
**Remark.** Algorithms (2.10) and (3.3) can be compared as follows. As most iterative algorithms can be explained in terms of *matrix splitting* [31], we consider the following splitting:

\[
I + B_1 + B_2 = (I + B_1 + B_2 + B_1B_2) - (B_1B_2).
\]

Then, it is clear from a view point of (2.8) that the solution of (2.10) at the \(n\)-th time step, \(u^n\), is the first iterate of the following iterative algorithm: Find \(u^{n,m}, m \geq 1\), by recursively solving

\[
(I + B_1 + B_2 + B_1B_2)u^{n,m} = (I - B_1 - B_2)u^{n-1} + B_1B_2u^{n,m-1} + kf^{n-1/2},
\]

starting with the initial value

\[
u^{n,0} = u^{n-1}.
\]

On the other hand, the solution of (3.3) is the first iterate of (3.4), with the initial value

\[
u^{n,0} = 2u^{n-1} - u^{n-2}.
\]

Algorithm (3.3) will thus be called the ADI with improved initial (ADI-II) in the remainder of the article.

**Remark.** One may consider higher-order corrections, choosing one of the following:

\[
\begin{aligned}
u^{n,0} &= 2u^{n-1} - 2u^{n-3} + u^{n-4}, & n &\geq 4, \\
u^{n,0} &= 4u^{n-1} - 6u^{n-2} + 4u^{n-3} - u^{n-4}, & n &\geq 4, \\
u^{n,0} &= \frac{1}{5}(18u^{n-1} - 24u^{n-2} + 14u^{n-3} - 3u^{n-4}), & n &\geq 4.
\end{aligned}
\]

But they have shown instability (over-correction!) unless \(k\) is sufficiently small. So using a high-order correction in (3.6) is not beneficial in practice. At this moment, one may ask the question: Is it possible for (3.5) to over-correct? If an over-correction is expected, (3.5) can be modified as

\[
u^{n,0} = u^{n-1} + \eta(u^{n-1} - u^{n-2}), \quad 0 \leq \eta \leq 1.
\]

Clearly, ADI and ADI-II respectively correspond to \(\eta = 0\) and \(\eta = 1\). It has been observed that (3.5) can over-correct for large timestep sizes. However, it has been also verified that when the timestep size is reasonably chosen (e.g., \(k \leq 2h\)), ADI-II works better than ADI and it performs best for \(\eta = 1\), for various heterogeneous media including anisotropy; see Tables 3 and 4 in §3.2. So the blending does not seem necessary for solving the heat equation in practice. Such blending is worth being considered for applications of ADI to elliptic problems (see §3.3), where the acceleration parameter corresponds to a large timestep size.
### 3.2. Accuracy and efficiency of ADI-II

To check accuracy and efficiency of ADI-II (3.3), we choose the domain $\Omega = (0, 1)^2$ and the time interval $J = (0, 1]$. The diffusion coefficients are selected as

\[
\begin{aligned}
a_1(x, y) &= 1, \\
a_2(x, y) &= 1/(2 + \cos(3\pi x) \cdot \cos(2\pi y)), \\
a_3(x, y) &= \begin{cases} 
1 + 0.5 \cdot \sin(5\pi x) \cdot y^3, & \text{if } x \leq 0.5, \\
1.5/(1 + (x - 0.5)^2) \cdot y^3, & \text{else},
\end{cases} \\
a_4(x, y) &= \begin{bmatrix}
a_2(x, y) & 0 \\
0 & a_3(x, y)
\end{bmatrix}.
\end{aligned}
\]

The first timestep of ADI-II is carried out by using ADI followed by SOR iterations. We compare four different algorithms to solve the problem in each time level: the LU-based, PCG-ILU0, ADI, and ADI-II.

Table 3 presents the performances of the four algorithms for the first three diffusion coefficients in (3.8), when $u = u_x$ with $\nu_t = 1$, $\nu_x = 4$, and $\nu_y = 3$. ADI deteriorates accuracy of the solution; its error is 16, 3, and 5 times larger than the original discretization error for $a = a_1$, $a = a_2$, and $a = a_3$, respectively. ADI-II requires only about 5-7% extra cost over the ADI method; its accuracy hardly differs from that of the direct solver.
Table 4 shows numerical results for various timestep sizes. Choose \( a = a_4 \) (an anisotropic diffusivity), \( c = b \equiv 0 \), \( \nu_t = 2.0 \), \( \nu_x = 6.25 \), and \( \nu_y = 7 \). The discretization selects \( h = h_x = h_y = 1/120 \). ADI shows a large splitting error even for small timesteps. Here again the improved initial (3.5) improves accuracy of ADI a lot, for a few percent of extra cost. However, as one can see from the table, ADI-II suffers inaccuracy as the timestep size increases. It was expected! To improve its performance, one may introduce a few SOR iterations following the ADI-II sweeps. For \( k = 2h \), the combination of ADI-II and four symmetric SOR iterations recovers the original accuracy; it takes 11.0 seconds, which is one quarter extra cost over ADI-II.

The Douglas-Rachford ADI method (2.12) incorporating the backward-time discretization has not been implemented to verify the adding-a-correction-term strategy. However, one can easily see that any correction term hardly improves accuracy of the algorithm in practical simulation, because the splitting error is already one-order higher than the temporal discretization error and the timestep size should be sufficiently small to get a reasonably accurate numerical solution.

3.3. Applications to elliptic problems

In this subsection, we discuss some efficiency issues for ADI and ADI-II applied to the elliptic problem

\[
\begin{align*}
(a) & \quad -\nabla \cdot (a \nabla u) + cu = f, \quad x \in \Omega, \\
(b) & \quad u = g, \quad x \in \Gamma.
\end{align*}
\]  

(3.9) When the central FD scheme is applied to (3.9), the algebraic system reads

\[
Au = (H + V)u = b,
\]

(3.10) where \( H \) and \( V \) correspond to \( -(au_x)_x + \frac{e}{2}u \) and \( -(au_y)_y + \frac{e}{2}u \), respectively. ADI for (3.10) can be formulated as follows: For given \( u^0 \), find \( u^m \), \( m \geq 1 \), by solving

\[
\begin{align*}
(rI + H)u^* &= (rI - V)u^{m-1} + b, \\
(rI + V)u^m &= (rI - H)u^* + b,
\end{align*}
\]

(3.11) where \( r > 0 \) is an acceleration parameter. Defining \( \rho = 1/r \), the above algorithm can be equivalently written as

\[
\begin{align*}
(I + \rho H)u^* &= (I - \rho H)(I - \rho V)u^{m-1} + 2\rho b, \\
(I + \rho V)u^m &= u^*.
\end{align*}
\]

(3.12) Notice the similarity between the above algorithm and (2.9). A variant of (3.12) corresponding to ADI-II (3.3) and the blending (3.7) reads

\[
\begin{align*}
(I + \rho H)u^* &= (I - \rho H - \rho V)u^{m-1} + 2\rho b + \rho^2 HV \left( u^{m-1} + \eta (u^{m-1} - u^{m-2}) \right), \\
(I + \rho V)u^m &= u^*.
\end{align*}
\]

(3.13)
Elliptic iterative solvers such as (3.12) can be considered as a *time-marching* method that stops when the solution becomes stationary. Thus the timestep can be set large to reach at the stationary solution as quickly as possible. However, the error incorporated due to a large timestep may degrade the convergence.

When \( \Omega = (0,1)^2 \), \( a = 1 \), \( c = 0 \), and the uniform grid spacing \( h = 1/N \), for some integer \( N > 1 \), the eigenvalues of \( H \) and \( V \) are known as

\[
\lambda_k = 2 - 2\cos \left( \frac{k\pi}{N} \right), \quad k = 1, 2, \ldots, N - 1.
\]

Thus the optimal parameter \( \hat{\rho} \) of (3.11) can be found as (see e.g. [31, §7.1] and [32])

\[
\hat{\rho} = \sqrt{\lambda_1 \lambda_{N-1}}. \tag{3.14}
\]

When \( N = 100 \) (\( \hat{\rho} \approx 0.0437 \)) and the source terms \( f \) and \( g \) correspond to \( u(x,y) = \sin(17\pi x) \cdot \sin(20\pi y) \), ADI (3.12) with \( \rho = \hat{\rho} := 1/\hat{\rho} \approx 22.9 \) converges in 51 iterations (taking 1.1 seconds) for the stopping criterion \( \| u^m - u^{m-1} \|_{\infty} / \| u^m \|_{\infty} < 10^{-5} \). The performance of ADI-II (3.13) depends on the blending parameter as follows:

<table>
<thead>
<tr>
<th>( \eta )</th>
<th>0.0</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>51</td>
<td>35</td>
<td>28</td>
<td>39</td>
<td>51</td>
<td>331</td>
</tr>
<tr>
<td>CPU</td>
<td>1.14</td>
<td>0.8</td>
<td>0.64</td>
<td>0.9</td>
<td>1.14</td>
<td>6.97</td>
</tr>
</tbody>
</table>

Over-correction appears as \( \eta \rightarrow 1 \). It has been observed from various numerical experiences that for elliptic problems, ADI-II performs about 1/3 faster than ADI when the blending parameter is set about 1/3. We do not know of any mathematical analysis addressing such a phenomenon for ADI.

4. FTS techniques for Navier-Stokes equations

4.1. Preliminaries

The NS equations (1.2) can be equivalently rewritten in index notation:

\[
\begin{align*}
(a) \quad & \frac{\partial (\rho v_j)}{\partial x_j} = 0, \quad (x, t) \in \Omega \times J, \\
(b) \quad & \frac{\partial (\rho v_i)}{\partial t} + \frac{\partial (\rho v_j v_i)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_i} + \rho g_i, \quad (x, t) \in \Omega \times J,
\end{align*}
\]

(4.1)

where the Einstein convention has been utilized. For a simpler presentation, we define an operator \( H(v_i) \) representing convective and diffusive terms

\[
H(v_i) := \frac{\partial (\rho v_j v_i)}{\partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j}.
\]
The Crank-Nicolson scheme for (4.1.b) can be formulated as follows:

\[
\frac{(\rho v_i)^n - (\rho v_i)^{n-1}}{\Delta t^n} + \frac{1}{2} \left[ H_h(v_i^n) + H_h(v_i^{n-1}) \right] = -\frac{1}{2} \left( \frac{\delta p^n}{\delta x_i} + \frac{\delta p^{n-1}}{\delta x_i} \right) + \rho g_i, \tag{4.2}
\]

where \( \Delta t^n = t^n - t^{n-1} \) and \( H_h \) is an approximation of \( H \) by the second-order FV discretization. Then, the scheme (4.2) shows the truncation error of

\[
\mathcal{O}\left( (\Delta x)^2 + (\Delta y)^2 + (\Delta t^n)^2 \right).
\]

A FTS method for (4.2) has been studied by Choi and Moin [4]:

(a) \( \frac{(\rho v_i)^* - (\rho v_i)^{n-1}}{\Delta t^n} + \frac{1}{2} \left[ H_h(v_i^*) + H_h(v_i^{n-1}) \right] = -\frac{\delta p^{n-1}}{\delta x_i} + \rho g_i, \)

(b) \( \frac{(\rho v_i)^{**} - (\rho v_i)^*}{\Delta t^n} = \frac{1}{2} \frac{\delta p^{n-1}}{\delta x_i}, \)

(c) \( \frac{(\rho v_i)^n - (\rho v_i)^{**}}{\Delta t^n} = -\frac{1}{2} \frac{\delta p^n}{\delta x_i}. \tag{4.3} \)

In the first step of the above algorithm, the velocity is advanced with the pressure explicitly treated; half the old pressure gradient is removed in the second step; and the final step computes the velocity in the new time level after finding the new pressure by solving

\[ \frac{\delta}{\delta x_i} \left( \frac{\delta p^n}{\delta x_i} \right) = \frac{2}{\Delta t^n} \frac{\delta (\rho v_i)^{**}}{\delta x_i}, \tag{4.4} \]

which is obtained by applying divergence to (4.3.c) to fill the requirement that the new velocity should satisfy the continuity equation (4.1.a).

Thus the three-step FTS method (4.3) satisfies the continuity equation and the momentum equations as follows:

\[
\frac{(\rho v_i)^n - (\rho v_i)^{n-1}}{\Delta t^n} + \frac{1}{2} \left[ H_h(v_i^*) + H_h(v_i^{n-1}) \right] = -\frac{1}{2} \left( \frac{\delta p^n}{\delta x_i} + \frac{\delta p^{n-1}}{\delta x_i} \right) + \rho g_i. \tag{4.5}
\]

The splitting error incorporated in the method is

\[ H_h(v_i^n) - H_h(v_i^*) = H'_h(\tilde{v}_i)(v_i^n - v_i^*), \]

for some \( \tilde{v}_i \) between \( v_i^n \) and \( v_i^* \). Adding (4.3.b) and (4.3.c) leads to

\[ v_i^n - v_i^* = \frac{\Delta t^n}{2\rho} \frac{\delta}{\delta x_i} (p^n - p^{n-1}). \]

Thus the splitting error of (4.3) satisfies

\[ H_h(v_i^n) - H_h(v_i^*) = \mathcal{O}(\Delta t^n)^2), \tag{4.6} \]
which is the same order as the truncation error of the Crank-Nicolson method already introduced. However, the splitting error can be much larger than expected as shown for the ADI method for the heat equation.

Note that (4.3.a) is nonlinear. It should be solved iteratively up to the satisfaction of a narrow tolerance. One can employ Picard iteration or Newton method, or their combination (a few Picard iterations followed by Newton iterations). In this article, we utilize Picard iteration only; see §4.3.

4.2. New FTS method

As for ADI applied to the numerical solution of the heat equation, we modify (4.3) so as to have a splitting error one-order higher than the truncation error. Consider the following modification: for $0 \leq \eta \leq 1$,

$$
\frac{(\rho u_i^*) - (\rho u_i)^{n-1}}{\Delta t^n} + \frac{1}{2} \left[ H_h(u_i^n) + H_h(u_i^{n-1}) \right] = -\frac{\delta}{\delta x_i} \left( \left( 1 + \frac{\eta}{2} \frac{\Delta t^n}{\Delta x_{i-1}} \right) p^{n-1} - \frac{\eta}{2} \frac{\Delta t^n}{\Delta x_{i-1}} p^{n-2} \right) + \rho g_i,
$$

(a)

$$
\frac{(\rho u_i^{**}) - (\rho u_i^*)}{\Delta t^n} = \frac{1}{2} \frac{\delta}{\delta x_i} \left( \left( 1 + \frac{\eta}{2} \frac{\Delta t^n}{\Delta x_{i-1}} \right) p^{n-1} - \frac{\eta}{2} \frac{\Delta t^n}{\Delta x_{i-1}} p^{n-2} \right),
$$

(b)

$$
\frac{(\rho u_i)^n - (\rho u_i)^{**}}{\Delta t^n} = -\frac{1}{2} \frac{\delta}{\delta x_i} p^n.
$$

(c)

Equation (4.7) thus solves the equation of the form (4.5), in addition to satisfying the continuity equation (4.1.a). But, here, the splitting error is one-order higher than (4.6) when $\eta = 1$. Indeed, adding (4.7.b) and (4.7.c) reads

$$
v_i^n - v_i^* = -\frac{\Delta t^n}{2\rho^2} \frac{\delta}{\delta x_i} \left( p^n - \left( \frac{\eta}{2} \frac{\Delta t^n}{\Delta x_{i-1}} \right) p^{n-1} - \frac{\eta}{2} \frac{\Delta t^n}{\Delta x_{i-1}} p^{n-2} \right),
$$

which is $\mathcal{O}(\Delta t^n)$ when $\eta = 1$.

Remark. Algorithm (4.7) can be applied for $n \geq 2$; a special care would better be taken for $u_i^1$. We apply two iterations of (4.3) in which for the second iteration, $p^0$ in the right side of (4.3.a) is replaced by $(p^{1,1} + p^0)/2$, where $p^{1,1}$ is the first iterate of the pressure, and (4.3.b) is skipped. (One may apply more iterations, up to convergence with a narrow tolerance. However, in practice it hardly improves accuracy.)
Figure 1: The mesh and staggered grid. The nodal points for scalar quantities are cell-centered (marked by ●) and those of velocity components are located at centroids of cell boundaries.

4.3. Finite volume method and iterative solvers

We implement algorithms (4.3) and (4.7) employing a second-order finite volume (FV) method on a rectangular domain: \( \Omega = (a_x, b_x) \times (a_y, b_y) \). Let \( \{ K_{l,m} \} \subset \Omega \) be the control volumes which differ for the pressure and velocity components. (See Figure 2.) We rewrite the algorithm (4.7) in its integral form:

\[
\begin{align*}
(a) & \quad \frac{1}{\Delta t^n} \int_{K_{l,m}} \rho v_i^* \, d\Omega + \frac{1}{2} \int_{\partial K_{l,m}} \rho v_i^* v_j^* \cdot \mathbf{n} \, dS - \frac{1}{2} \int_{\partial K_{l,m}} \tau_{ij}^* I_j \cdot \mathbf{n} \, dS = Q_{l,m}^{n-1}, \\
(b) & \quad \frac{1}{\Delta t^n} \int_{K_{l,m}} \rho v_i^{**} \, d\Omega = \frac{1}{\Delta t^n} \int_{K_{l,m}} \rho v_i^* \, d\Omega \\
& \quad + \frac{1}{2} \int_{\partial K_{l,m}} \left( (1 + \eta \frac{\Delta t^n}{\Delta t^{n-1}}) p^{n-1} - \eta \frac{\Delta t^n}{\Delta t^{n-1}} p^{n-2} \right) I_i \cdot \mathbf{n} \, dS, \\
(c) & \quad \frac{1}{\Delta t^n} \int_{K_{l,m}} \rho v_i^q \, d\Omega = \frac{1}{\Delta t^n} \int_{K_{l,m}} \rho v_i^{**} \, d\Omega - \frac{1}{2} \int_{\partial K_{l,m}} p^n I_i \cdot \mathbf{n} \, dS,
\end{align*}
\]

where

\[
Q_{l,m}^{n-1} = \frac{1}{\Delta t^n} \int_{K_{l,m}} \rho v_i^{n-1} \, d\Omega + \int_{K_{l,m}} \rho g_i \, d\Omega \\
- \frac{1}{2} \int_{\partial K_{l,m}} \rho v_i^{n-1} \mathbf{v}^{n-1} \cdot \mathbf{n} \, dS + \frac{1}{2} \int_{\partial K_{l,m}} \tau_{ij}^{n-1} I_j \cdot \mathbf{n} \, dS \\
- \int_{\partial K_{l,m}} \left( (1 + \frac{\eta}{2} \frac{\Delta t^n}{\Delta t^{n-1}}) p^{n-1} - \frac{\eta}{2} \frac{\Delta t^n}{\Delta t^{n-1}} p^{n-2} \right) I_i \cdot \mathbf{n} \, dS.
\]

In the following, we consider a staggered mesh and corresponding control volumes, an implicit method of velocity updates for (4.10.a), and the Stone’s method for the linearized algebraic systems for velocity components.

**Mesh generation and staggered control volumes:** The mesh is generated as follows. For the partitioning of \((a_x, b_x)\) into \(n_x(= 2k)\) subintervals, let \(c_x\) be the
center of the interval \((a_x, b_x)\), i.e., \(c_x = (a_x + b_x)/2\). The interval \((a_x, c_x)\) is divided into \(k\) subintervals: \(a_x = x_0 < x_1 < \cdots < x_k = c_x\), satisfying

\[
\Delta x_{\ell+1} = \sigma \Delta x_{\ell}, \quad \sigma \geq 1, \quad \ell = 1, \cdots, k-1,
\]

where \(\Delta x_{\ell} = x_{\ell} - x_{\ell-1}\). The other half, \((c_x, b_x)\), is partitioned by the points that are the reflection image of \(\{x_0, x_1, \cdots, x_k\}\) with respect to \(c_x = x_k\). So the grid points in the \(x\)-direction are \(a_x = x_0 < x_1 < \cdots < x_{n_x} = b_x\), where \(n_x = 2k\). See Figure 1. The same can be applied to the generation of \(y\)-directional grid points: \(a_y = y_0 < y_1 < \cdots < y_{n_y} = b_y\). Thus the cells composed by the grid lines are

\[
(x_{\ell-1}, x_{\ell}) \times (y_{m-1}, y_m), \quad \ell = 1, \cdots, n_x, \quad m = 1, \cdots, n_y.
\]

Now we build the control volumes \(K_{\ell,m}\) which differ for the pressure and the velocity components. Let

\[
x_{c,\ell} = (x_{\ell-1} + x_{\ell})/2, \quad \ell = 1, \cdots, n_x, \quad y_{c,m} = (y_{m-1} + y_m)/2, \quad m = 1, \cdots, n_y.
\]

The nodal points and the corresponding control volumes for the pressure and the velocity components are depicted in Figure 2. The control volumes for the pressure coincide with the cells of the mesh. Note that nodal points for the velocity components may not be located at the center of the corresponding control volumes.

**Implicit velocity update:** Equation (4.10.a) includes nonlinear terms and its linearized problem results in an algebraic system of equations which requires an efficient solver. As a part of iterative procedure for (4.10.a), we utilize the Picard iteration and a deferred correction (with blending) for the convection term as follows:
Set $v_i^{s,0} = v_i^{n-1}$ and find $v_i^{s,k}$, $k \geq 1$, satisfying
\[
\frac{1}{\Delta t} \int_{K_{i,m}} \rho v_i^{s,k} \, d\Omega + \frac{1}{2} \int_{\partial K_{i,m}} \rho v_i^{s,k} v_{i,j,UDS} \cdot \mathbf{n} \, dS - \frac{1}{2} \int_{\partial K_{i,m}} \tau_{ij}^{s,k} I_j \cdot \mathbf{n} \, dS \nonumber \\
= \beta \frac{1}{2} \int_{\partial K_{i,m}} \rho (v_i^{s,k-1,UDS} - v_i^{s,k-1,CDS}) v_i^{s,k-1} \cdot \mathbf{n} \, dS + Q_{i,m}^{n-1},
\] (4.12)
where $0 \leq \beta \leq 1$ and $v_i^{s,k,UDS}$ and $v_i^{s,k,CDS}$ denote respectively the upwind and central approximations when evaluated on the interfaces of the control volumes. Other terms are approximated using central schemes. Note that the limit of the iterates (if any) solves (4.10.a) with the convection term approximated by the upwind scheme ($\beta = 0$), the central scheme ($\beta = 1$), or their blending ($0 < \beta < 1$). For implementation details, see e.g. [17, §7.5].

**The solver for the linear system:** When every term in (4.12) is approximated by the FV method, each of the equations ($i = 1, 2$) results in an algebraic system of the form
\[
Au = b.
\] (4.13)
To solve it, we adopt Stone’s *strongly implicit procedure* (SIP) [28]. We briefly describe the basic idea of SIP.

As for other iterative methods, SIP is based on a regular splitting, $A = M - N$, with $M$ being an incomplete LU (ILU) factorization:
\[
M := L_I U_I = A + N,
\] (4.14)
where $L_I$ and $U_I$ are respectively the lower and upper triangular components of the ILU factorization of $A$, with the entries of the main diagonal of $U_I$ being all one. Note that the iteration corresponding to the splitting (4.14) is formulated as
\[
L_I U_I u^k = Nu^{k-1} + b,
\] (4.15)
or, equivalently,
\[
\begin{align*}
(a) & \quad r^{k-1} = b - Au^{k-1}, \\
b) & \quad L_I U_I \delta^{k-1} = r^{k-1}, \\
c) & \quad u^k = u^{k-1} + \delta^{k-1}.
\end{align*}
\] (4.16)

To make the iteration (4.15) converge fast, we may have to choose elements of $L_I$ and $U_I$ in a way that $N$ is as small as possible. For a 2D problem in a rectangular mesh where the grid points are ordered in the row-wise manner, the Stone’s SIP
determines the entries of $L_I$ and $U_I$ as follows:

\[
\begin{align*}
L^\ell_S &= A^\ell_S / (1 + \alpha U^{\ell, m-1}_S), \\
L^\ell_W &= A^\ell_W / (1 + \alpha U^{\ell-1, m}_W), \\
L^\ell_C &= A^\ell_C + \alpha (L^\ell_S U^{\ell, m-1}_E + L^\ell_W U^{\ell-1, m}_N) \\
&\quad - L^\ell_S U^{\ell, m-1}_S - L^\ell_W U^{\ell-1, m}_W, \\
U^{\ell,m}_E &= (A^\ell_E - \alpha L^\ell_S U^{\ell, m-1}_E) / I^{\ell,m}_C, \\
U^{\ell,m}_N &= (A^\ell_N - \alpha L^\ell_W U^{\ell-1, m}_N) / I^{\ell,m}_C,
\end{align*}
\]  

(4.17)

where the pair of superscripts $(\ell, m)$ corresponds to the $(\ell, m)$-th control volume (so it determines a row in the algebraic system). The subscripts $S$, $W$, $E$, and $N$ indicate the matrix entries related to the adjacent nodal points of $C$ in the south, west, east, and north directions, respectively. (See Figure 2.) Here $\alpha \in (0, 1]$ is a parameter. One may set $\alpha = 1$ for a high-order approximation of $L_I U_I$ to $A$, but Stone found that stability requires $\alpha < 1$; it is often chosen between 0.92 and 0.96 [17, §5.3.4]. Entries of $L_I$ and $U_I$ used in (4.17) whose indices are outside the index boundaries should be set zero.

**The pressure solver:** The Poisson equation (4.4) reads, in its integral form,

\[
\int_{\partial K_{\ell,m}} \nabla p \cdot \mathbf{n} \, ds = \frac{2}{\Delta p} \int_{\partial K_{\ell,m}} \rho \mathbf{v}^* \cdot \mathbf{n} \, ds.
\]  

(4.18)

No flux condition is often imposed on the boundary. When using the second-order FV method, we can obtain the five-point scheme as follows:

\[
\int_{\partial K_{\ell,m}} \nabla p \cdot \mathbf{n} \, ds \approx \frac{1}{2} \left( y_m - y_{m-1} \right) \left( \frac{p_E - p_C}{x_{c,m+1} - x_{c,m}} - \frac{p_C}{x_{c,m} - x_{c,m-1}} \right) \\
+ \frac{1}{2} \left( x_\ell - x_{\ell-1} \right) \left( \frac{p_N - p_C}{y_{c,\ell+1} - y_{c,\ell}} - \frac{p_C}{y_{c,\ell} - y_{c,\ell-1}} \right).
\]

The right side of (4.18) can be easily integrated without averaging the velocity components.

The resulting algebraic system can be solved by employing SIP, as for linearized velocity components. Since the matrix is symmetric and positive definite (when the pressure is fixed at a nodal point on the boundary), we may explore a more efficient algorithm which possibly takes advantage of such nice matrix properties. As an alternative to SIP, we would adopt a PCG-ILU0 with the preconditioner being either the Stone’s decomposition (4.17) or the conventional ILU0 utilized in solving the heat equation.
Table 5: The error at $t = 20$ and the elapsed times. Choose $50 \times 50$ uniform cells and set $\alpha = 0.92$, $\beta = 1$, and $\eta = 1$.

### 5. Numerical experiments for NS equations

Set $\Omega = (0, 1)^2$, $\Gamma = \partial \Omega$, and consider two lid-driven cavity problems of which the boundary conditions are specified as follows: for $t \in J = (0, T]$, $T > 0$,

$$
\text{CP1:} \quad \begin{cases} 
  \mathbf{v}(\mathbf{x}, t) = 1, & \mathbf{x} \in \{y = 1\}, \\
  \mathbf{v}(\mathbf{x}, t) = 0, & \text{else on } \Gamma,
\end{cases} \\
\text{CP2:} \quad \begin{cases} 
  \mathbf{v}(\mathbf{x}, t) = 1, & \mathbf{x} \in \{y = 1\}, \\
  \mathbf{v}(\mathbf{x}, t) = \min(1, t/10), & \mathbf{x} \in \{y = 0\}, \\
  \mathbf{v}(\mathbf{x}) = 0, & \text{else on } \Gamma.
\end{cases}
$$

The solutions of the cavity problems become static as $t$ grows. To check accuracy and efficiency of (4.3) and (4.7) for transient phenomena of NS flows, we choose $T = 20$.

The density $\rho \equiv 1$ and the shear coefficient of viscosity $\mu = 0.001$. (So the Reynolds number $\text{Re} = 1000$.) The bulk coefficient of viscosity is set as $\kappa = \frac{3}{2}\mu$, for simplicity.

The domain is partitioned into $n_x \times n_y$ cells, either uniformly or non-uniformly, by setting either $\sigma = 1$ or $\sigma > 1$ in (4.11). The initial values are set all zero:

$$
\mathbf{v}(\mathbf{x}, 0) = p(\mathbf{x}, 0) = 0, \quad \mathbf{x} \in \Omega.
$$

In each time level, the Picard iteration and iterative solvers for linear algebraic systems stop when satisfying the following stopping criteria:

$$
\| \mathbf{v}^{*,k} - \mathbf{v}^{*,k-1} \|_\infty \leq 10^{-6}, \quad \text{(Picard iteration (4.12))}
$$

$$
\| \mathbf{r}^k - \mathbf{r}^{k-1} \|_\infty / \| \mathbf{r}^0 \|_\infty \leq 10^{-3}, \quad \text{(Linearized velocity solve (4.16))}
$$

$$
\| \mathbf{r}^k - \mathbf{r}^{k-1} \|_\infty / \| \mathbf{r}^0 \|_\infty \leq 10^{-6}, \quad \text{(Poisson solve (PCG-ILU0))}
$$

We use the same computer as in the heat flow simulation: a 400 MHz laptop having 196M memory and a Linux operating system.

Table 5 presents numerical results for the FTS algorithms (4.3) and (4.7) applied to the cavity problems (5.1). The domain is partitioned into $50 \times 50$ uniform cells
Figure 3: The computed velocity fields at $t = 20$ for CP1 (left) and CP2 (right).

($\sigma = 1$) and set $\alpha = 0.92$ in (4.17), $\beta = 1$ in (4.12), and $\eta = 1$ in (4.11). To check the temporal error, we first compute the solution $\{\tilde{\mathbf{v}}, \tilde{\mathbf{w}}\}$ using the Choi-Moin algorithm (4.3) with $\Delta t = \Delta t^0 = 0.02$ (1000 equal timesteps for $J = (0,20]$). Then, the error is obtained as

$$EV_T = \| \tilde{\mathbf{v}}(\cdot, T) - \mathbf{v}_{\Delta t}(\cdot, T) \|_\infty,$$

where $\mathbf{v}_{\Delta t}(\cdot, T)$ is the computed velocity at $t = T$ with the timestep size $\Delta t$. (Here the spatial partitioning is set the same for all cases.) The elapsed time CPU is measured in second. For CP1, the new algorithm reduces the temporal error by a factor of three, and it performs slightly faster than Choi-Moin’s algorithm (due to a small gain in Poisson solve). On the other hand, the new algorithm for CP2 is hardly advantageous over the old algorithm. Notice that the error convergence rates of both (4.3) and (4.7) are far below two; the (second-order) splitting error included in the solution of (4.3) seems negligible for this case. The new algorithm chooses the correction parameter $\eta = 1$, for all cases in the table; no over-correction is observed.

Figure 3 shows the computed velocity fields at $t = 20$ by using (4.7) for CP1 (left) and CP2 (right). The timestep size $\Delta t = 20/400 = 0.05$ and the domain is partitioned into $50 \times 50$ cells; the grid spacing factors are chosen as $\sigma = 1.05$ and $\sigma = 1$ respectively for CP1 and CP2. For the nonuniform mesh, the maximum and the minimum spatial step sizes respectively become 0.035 and 0.01.

6. Conclusions

Fractional time-stepping (FTS) methods such as ADI [7, 14, 25] and the velocity-pressure splitting technique [4, 5] have been considered for the numerical solution
of the heat equation and the unsteady incompressible Navier-Stokes equations in the two-dimensional space. Such FTS algorithms have proved efficient in the approximation of the solutions of evolutionary differential equations in two and three variables; however, they may introduce a large splitting error unless the timestep size is sufficiently small. We have introduced a correction term to turn the splitting error to be one-order higher than the discretization in time. To overcome a possible over-correction of higher-order correction schemes, the strategy of blending has been suggested. It has been numerically verified that the FTS algorithms incorporating the correction term and blending improve the overall accuracy by a little to one-order over the original algorithms; the new FTS algorithms require about 5-7% extra cost in the heat flow simulation and no extra cost for the numerical solution of the Navier-Stokes equations.

Acknowledgement

The codes utilized in this article have been implemented for the model-code library: Graduate Research and Applications for Differential Equations (GRADE), and are available through internet access to http://www.ms.uky.edu/~skim/GRADE/ or by asking S. Kim (skim@ms.uky.edu).

References


