On Accuracy of Alternating Direction Implicit Methods for Parabolic Equations

Seongjai Kim* and Jim Douglas, Jr.†

Abstract

We study accuracy of alternating direction implicit (ADI) methods for parabolic equations. The original ADI method applied to parabolic equations is a perturbation of the Crank-Nicolson difference equation and has second-order accuracy both in space and time. The perturbation error is on the same order as the discretization error, in terms of mathematical description. However, we often observe in simulation that the truncation error is one order larger than the discretization error when the solution is moderately or highly oscillatory. This paper presents the observation and strategies recovering the accuracy of the Crank-Nicolson difference equation.

1 Introduction

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, $T > 0$. The problem to be considered in this article is

$$\begin{array}{llll} \text{(a)} & u_t = \nabla \cdot (a \nabla u) + c(t,x,y,u) + f, & t \in J, & (x,y) \in \Omega, \\ \text{(b)} & a u_\nu + \beta u = g, & t \in J, & (x,y) \in \Gamma, \\ \text{(c)} & u = u_0, & t = 0, & (x,y) \in \Omega, \end{array} \tag{1.1}$$

where the coefficients a, c, and β are given functions defined on the domain Ω, the subscript ν denotes the outer unit normal from the domain, u_0 is the prescribed solution for $t = 0$, and f and g are the sources.

Alternating direction implicit/iterative (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 [6, 17, 27], when applied to the heat equation of the form (1.1), is a perturbation of the Crank-Nicolson difference equation

*Department of Mathematics, University of Kentucky, Lexington, KY 40506-0027 USA Email: skim@ms.uky.edu

†Department of Mathematics, Purdue University, W. Lafayette, IN 47097 USA Email: douglas@math.purdue.edu

and is second-order correct both in space and time, in theory. Unfortunately, in simulation, we often observe that the error introduced from the perturbation is much larger than the truncation error brought forward from the numerical discretizations.

The object of this paper is to present a *numerical* strategy for ADI methods which recovers the original accuracy of the Crank-Nicolson difference equation. When c is independent of u or a predictor-corrector generalization is employed, the discretization error becomes $\mathcal{O}(h^2 + k^2)$, where h and k are respectively the spatial and temporal grid sizes. The new ADI algorithm in each time step incorporates *improved initial* values of the from

$$Eu^n = 2u^n - u^{n-1};$$

the algorithm is called the *ADI of improved initial* (ADI-II). The perturbation error for ADI-II is $\mathcal{O}(k^3)$, while the original ADI method introduces the perturbation error $\mathcal{O}(k^2)$. It has been observed that ADI-II increases the computational cost only about 5-7% over the original ADI method and is accurate enough for moderately small k. For large temporal grid sizes, i.e. $k > h$, we may adapt the algorithm to keep up the original accuracy: ADI-II is followed by m iterations of Gauss-Seidel relaxation, which is called ADI-II(m). For all cases we have tested, ADI-II(4) recovers the original accuracy satisfactorily. Its L^2-error hardly differs from that of a direct solver, mostly less than 1%; its computation cost becomes only one fourth higher than the original ADI method.

The ADI method was first suggested by Douglas and Peaceman [17], Douglas [6], and Peaceman and Rachford [27] for solving the heat equation in 2D. The method was extended for mildly nonlinear problems, [7], three space variables [10], and non-symmetric problems [24, 29]. A general formulation for ADI methods for parabolic and hyperbolic problems can be found in [16]. Pearcy [28] 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 can be found in [33]. A collection of interesting results were presented applying ADI to finite element methods [12], p-version finite element methods [11], mixed finite element methods [15], and collocation methods [25, 31]. The ADI methods have been applied to various physical problems [1, 3, 4, 5, 19, 20, 22, 23]. Parallelization of ADI methods has been tried in [21, 26, 29]. See Varga [32, Ch.7] and Strikwerda [30, §7.3] for variants and systematic descriptions for the ADI method.

An outline of the paper is as follows. In §2, we briefly review the classical ADI method, a perturbation of the Crank-Nicolson difference equation. In §3, the perturbation error is numerically tested. It has been observed that the perturbation error often becomes one order larger than the discretization error. In §4, we suggest a strategy for getting rid of the perturbation error. Its efficiency is numerically verified in §5. The resulting algorithm reduces the perturbation error less than 1% of the total error; its computation cost increases only one fourth of the original ADI method.

2 Classical ADI method

In this section we review the classical ADI method applied to the heat equation of the form (1.1). Let's start with linear reactions, i.e. $c(t, x, y, u) = c(t, x, y) \cdot u$. Define

$$A_1 u = -(a u_x)_x - \frac{1}{2} c u, \quad A_2 u = -(a u_y)_y - \frac{1}{2} c u.$$

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

$$u_t + A_1 u + A_2 u = f. \tag{2.1}$$

Let the time interval $J = (0, T]$ be partitioned into $\{0 = t^0 < t^1 < \cdots < t^{n_t} = T\}$, where

$$t^n = n \cdot k, \quad n = 0, 1, \cdots, n_t, \quad k = T/n_t. \tag{2.2}$$

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:

$$\begin{aligned} x_i &= a_x + i \cdot h_x, \quad i = 0, 1, \cdots, n_x, \quad h_x = (b_x - a_x)/n_x, \\ y_j &= a_y + j \cdot h_y, \quad j = 0, 1, \cdots, n_y, \quad h_y = (b_y - a_y)/n_y. \end{aligned} \tag{2.3}$$

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

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

or

$$\Big(I + \frac{k}{2} A_1 + \frac{k}{2} A_2\Big) u^{n+1} = \Big(I - \frac{k}{2} A_1 - \frac{k}{2} A_2\Big) u^n + k f^{n+1/2} + \mathcal{O}(k^3), \tag{2.4}$$

where

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

Now, we replace the operators A_1 and A_2 in (2.4) respectively by their spatial approximations A_{1h} and A_{2h}, central finite differences on the partition (2.3). Then, (2.4) becomes

$$(I + B_1 + B_2) u^{n+1} = (I - B_1 - B_2) u^n + k f^{n+1/2}. \tag{2.5}$$

where

$$B_1 = \frac{k}{2} A_{1h} \text{ and } B_2 = \frac{k}{2} A_{2h}.$$

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

The ADI method starts with adding $B_1 B_2 u^{n+1}$ to the both sides of (2.5). Then it reads

$$\begin{aligned} (I + B_1 + B_2 + B_1 B_2) u^{n+1} = {} & (I - B_1 - B_2 + B_1 B_2) u^n \\ & + B_1 B_2 (u^{n+1} - u^n) + k f^{n+1/2}, \end{aligned} \tag{2.6}$$

which can be factored as

$$\begin{aligned}(I+B_1)(I+B_2)u^{n+1} = (I-B_1)(I-B_2)u^n \\ +B_1B_2(u^{n+1}-u^n)+kf^{n+1/2}.\end{aligned} \tag{2.7}$$

Consider the second term on the right side. We have

$$u^{n+1} = u^n + \mathcal{O}(k),$$

so with the k^2 factor this second term is $\mathcal{O}(k^3)$, which is the same order as the truncation error already introduced. Ignoring the term

$$G_h(u^{n+1}, u^n) := B_1B_2(u^{n+1}-u^n), \tag{2.8}$$

we can rewrite (2.7) as

$$(I+B_1)(I+B_2)u^{n+1} = (I-B_1)(I-B_2)u^n + kf^{n+1/2}. \tag{2.9}$$

To solve (2.9), Douglas, Peaceman, and Rachford [6, 17, 27] proposed

$$\begin{array}{llcll} \text{(a)} & (I+B_1)\widetilde{u}^{n+1/2} & = & (I-B_2)u^n + \dfrac{k}{2}f^{n+1/2}, & (x\text{-sweep}) \\ \text{(b)} & (I+B_2)u^{n+1} & = & (I-B_1)\widetilde{u}^{n+1/2} + \dfrac{k}{2}f^{n+1/2}. & (y\text{-sweep}) \end{array} \tag{2.10}$$

(The algorithm was presented in a slightly different form in the beginning. It is easy to see that the original algorithm is equivalent to (2.10); see [8].) Theoretical aspects of the method were treated in detail in [6], while practical aspects of the calculation are considered in the companion paper [27] in considerable detail. Through this paper, algorithm (2.10) will be called the *Douglas-Peaceman-Rachford* (DPR) method, if we need to specify it from other ADI methods later developed. Note that for each sweep the matrix to be inverted is tridiagonal. The algorithm requires $\mathcal{O}(N)$ flops ($N := n_t n_x n_y$, the number of grid points) for the whole computation. As in Wachspress and Habetler [34], we can economize the algorithm by writing (2.10b) in a form in which the matrix B_1 does not appear explicitly. Substitute for $B_1\widetilde{u}^{n+1/2}$ from (2.10a). Then, the algorithm can be rewritten as

$$\begin{array}{llcll} \text{(a)} & (I+B_1)\widetilde{u}^{n+1/2} & = & (I-B_2)u^n + \dfrac{k}{2}f^{n+1/2}, & (x\text{-sweep}) \\ \text{(b)} & (I+B_2)u^{n+1} & = & 2\widetilde{u}^{n+1/2} - (I-B_2)u^n. & (y\text{-sweep}) \end{array} \tag{2.11}$$

Other ADI schemes can be derived starting with other basic schemes. Starting with the backward-time central-space scheme for the problem (2.1), one can have

$$(I + kA_{1h} + kA_{2h})u^{n+1} = u^n + kf^{n+1} + \mathcal{O}(k^2 + kh^2).$$

Adding $k^2A_{1h}A_{2h}u^{n+1} (:= 4B_1B_2u^{n+1})$ to the both sides of the above equation, and factoring its left side, we have

$$\begin{aligned}(I+2B_1)(I+2B_2)u^{n+1} = u^n + 4B_1B_2u^n + kf^{n+1} \\ +4B_1B_2(u^{n+1}-u^n) + \mathcal{O}(k^2+kh^2).\end{aligned}$$

Ignore the term

$$4B_1B_2(u^{n+1} - u^n). \tag{2.12}$$

Then, we can derive the Douglas-Rachford method [18]:

$$\begin{array}{llll} \text{(a)} & (I + 2B_1)\tilde{u}^{n+1/2} & = & (I - 2B_2)u^n + kf^{n+1}, \quad (x\text{-sweep}) \\ \text{(b)} & (I + 2B_2)u^{n+1} & = & \tilde{u}^{n+1/2} + 2B_2u^n. \qquad\quad (y\text{-sweep}) \end{array} \tag{2.13}$$

It is easy to see that algorithms (2.11) and (2.13) cost *almost* the same for the same size of problems. It has been observed that algorithm (2.11) performs 2 to 7% faster than (2.10) for most cases of various sources f and grid sizes. In this paper, we are more interested in accuracy of ADI methods than in a slight improvement in numerical performance; experiments will be focused on algorithm (2.10).

3 Numerical accuracy of ADI

ADI methods ignoring terms in (2.8) or (2.12) are often showing unsatisfactory performances in terms of accuracy. More specifically speaking, the *perturbation error* arising from the ignored terms can be much larger than the error originated from the discretizations. In this section, we will numerically verify the above claim. Let's start with choosing two different solutions:

$$\begin{array}{lll} u_+(t,x,y) & = & \sin(2\pi\nu_t t) + \sin(2\pi\nu_x x) + \sin(2\pi\nu_y y), \\ u_\times(t,x,y) & = & \sin(2\pi\nu_t t) \cdot \sin(2\pi\nu_x x) \cdot \sin(2\pi\nu_y y), \end{array} \tag{3.1}$$

where ν_t, ν_x, and ν_y are respectively the frequencies of the solution in t, x, and y directions. Set $J \times \Omega = (0,1) \times (0,1)^2$, $a \equiv 1$, and $c = \beta \equiv 0$. For the solution frequencies, choose $\nu_t = \nu_x = \nu_y = 1$. The sources f and g are set to satisfy (1.1). For discretizations, uniform meshes are selected; $n := n_t = n_x = n_y$. To compare the computation cost and accuracy, we implemented two algorithms: a PCG-ILU for (2.5) and the ADI algorithm (2.10). The zero-level ILU is considered, that is, no fill-in is allowed for ILU. For the PCG-ILU, the initial point is set to be the extrapolation

$$u^{n+1,0} = 2u^n - u^{n-1}.$$

The main/driver routines are written in C++ and C, and the core routines in F77. Every computation is carried out on a Gateway Solo, a 266 MHz laptop having 128M 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 two different algorithms show the same 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-ILU to converge up to the relative L^2 tolerance 10^{-5}. The error in PCG-ILU seems to come from discretizations; the same error level has been observed when the problem in each time step is solved by LU-factorization.

	$n = 40$		$n = 80$		$n = 160$	
	CPU	L^2-error	CPU	L^2-error	CPU	L^2-error
PCG-ILU	0.9	4.11e-3	9.0	1.00e-3	91.7	2.48e-4
ADI	0.5	4.10e-3	4.0	1.00e-3	33.3	2.47e-4

Table 1: The performances of PCG-ILU and ADI for $u = u_+$. For the computation, $a \equiv 1$, $c = \beta \equiv 0$, and $\nu_t = \nu_x = \nu_y = 1$. Meshes are set to be uniform: $n := n_t = n_x = n_y$.

	$n = 40$		$n = 80$		$n = 160$	
	CPU	L^2-error	CPU	L^2-error	CPU	L^2-error
PCG-ILU	1.5	2.46e-4	18.7	5.97e-5	207.5	1.42e-5
ADI	0.8	8.44e-3	6.7	2.02e-3	54.4	4.90e-4

Table 2: The performances of PCG-ILU and ADI for $u = u_\times$. The problem coefficients and algorithm parameters are chosen the same as in Table 1.

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-ILU shows a slight *super-linearity* in its computation cost, as one can expect. For marching a time step, average 12 to 22 iterations were required for PCG-ILU to converge, with the iteration counts becoming larger for more grid points. (Note that the solutions change non-trivially at every grid point in each time level.) In the table, both algorithms show the second-order accuracy. However, ADI produces approximately 34 times larger error than PCG-ILU for the same grid size. This implies that ADI requires approximately 5.75 times more grid points in each direction to keep up the same level of accuracy as PCG-ILU, which would result in an algebraic system about 190 times larger. Take a look at the errors for $n = 40$ in PCG-ILU and $n = 160$ in ADI. The number of grid points for ADI is 64 times larger than that for PCG-ILU, but the error is still twice larger. In summary, ADI can produce a huge amount of perturbation error that eventually makes the algorithm much more expensive.

One can easily see from the above tables that the larger errors result from the ignorance of the term $G_h(u^{n+1}, u^n)$ in (2.8). It is obvious that G_h applied to u_+ is near zero, which explains the same level of accuracy for ADI and PCG-ILU in Table 1. For $u_\times$ in Table 2, the dominant error source for the ADI method is the ignorance of G_h. In the next section, we will consider a recipe for ADI methods to get rid of the perturbation error (without introducing extra grid points).

4 ADI as predictor and corrector

This section presents strategies to get rid of the perturbation error in ADI methods and recover the accuracy of the Crank-Nicolson equation, for a small fraction of the computation cost. As most iterative algorithms can be explained as a matrix splitting [32], we first consider the following splitting: $I \pm B_1 \pm B_2 = P_\pm - Q$, where

$$P_\pm = I \pm B_1 \pm B_2 + B_1 B_2 = (I \pm B_1)(I \pm B_2), \quad Q = B_1 B_2.$$

Then, it is clear that the solution of (2.10) at the n-th time step, u^{n+1}, is the first iterate of the following algorithm: Find $u^{n+1,m+1}$, $m = 0, 1, \cdots$, by recursively solving

$$\begin{aligned} P_+ u^{n+1,m+1} &= Q u^{n+1,m} + (I - B_1 - B_2) u^n + k f^{n+1/2} \\ &= Q(u^{n+1,m} - u^n) + P_- u^n + k f^{n+1/2}, \end{aligned} \tag{4.1}$$

starting with the initial value

$$u^{n+1,0} = u^n. \tag{4.2}$$

(Here we have omitted the description of the intermediate steps.) Now, let us utilize the initial guess of the form

$$u^{n+1,0} = 2u^n - u^{n-1}, \quad n \geq 1. \tag{4.3}$$

and perform just one iteration. Then, the resulting algorithm can be formulated as follows, called an *ADI of improved initial* (ADI-II):

$$P_+ u^{n+1} = P_- u^n + Q(u^n - u^{n-1}) + k f^{n+1/2}, \quad n \geq 1. \tag{4.4}$$

Compared with (2.7), algorithm (4.4) in each time step would introduce the perturbation error

$$Q(u^{n+1} - 2u^n + u^{n-1}). \tag{4.5}$$

It is easy to see that the above term is $\mathcal{O}(k^4)$ and therefore it is on one order higher in k than the discretization error!

We see from (4.3) that information is required at two preceding time levels for (4.4) to advance in time. Thus a staring procedure is needed to define u^1 which will retain the overall accuracy of the method.

Let v^1 be the solution of (2.10) for $n = 0$:

$$P_+ v^1 = P_- u^0 + k f^{1/2}. \tag{4.6}$$

Then, it is easy to see that the error correction can be achieved by solving

$$P_+ \delta v = Q(\delta v + v^1 - u^0) \tag{4.7}$$

or

$$(I + B_1 + B_2)\delta v = Q(v^1 - u^0), \tag{4.8}$$

and updating

$$u^1 = v^1 + \delta v.$$

To solve (4.7), we may employ (with the initial guess $e^0 = 0$)

$$P_+ e^{\ell+1} = Q(e^\ell + v^1 - u^0), \tag{4.9}$$

or (with the initial guess $p^0 = v^1 - u^0$)

$$P_+ p^{\ell+1} = Q p^\ell. \tag{4.10}$$

Theorem: *Algorithms* (4.9) *and* (4.10) *are convergent and*

$$e^\ell = \sum_{j=1}^{\ell} p^j \to \delta v, \quad \textit{as } \ell \to \infty. \tag{4.11}$$

Proof. Recall that B_1 and B_2 are symmetric and positive semi-definite. (Symmetry of B_1 and B_2 can be obtained from a discretization technique that mimics the bilinear finite element method incorporating the Trapezoid quadrature rule.) Define

$$R := (I + B_2)^{-1}(I + B_1)^{-1}\, B_1 B_2 \tag{4.12}$$

and $\rho(R)$ be the the spectral radius of matrix R. Then,

$$\begin{aligned} \rho(R) &= \rho\Big((I + B_1)^{-1} B_1 B_2 (I + B_2)^{-1}\Big) \\ &\le \Big\|(I + B_1)^{-1} B_1\Big\| \cdot \Big\|B_2 (I + B_2)^{-1}\Big\| \\ &= \max_i \frac{\lambda_i}{1 + \lambda_i} \cdot \max_j \frac{\mu_j}{1 + \mu_j} < 1, \end{aligned} \tag{4.13}$$

where $\|\cdot\|$ is the induced matrix L^2 norm and λ_i and μ_j are respectively the nonnegative eigenvalues of B_1 and B_2. We have proved the convergence. The equality in (4.11) holds clearly. □

We summarize the starting procedure: For a prescribed accuracy level ε and reduction level γ, solve the following

$$\begin{array}{ll} \text{(a)} & P_+ v^1 = P_- u^0 + k f^{1/2}; \\ \text{(b)} & \text{Set } u^1 = v^1; \quad p^0 = v^1 - u^0; \\ & \text{For } \ell = 0, 1, 2, \cdots \\ & \quad P_+ p^{\ell+1} = Q p^\ell; \\ & \quad u^1 \leftarrow u^1 + p^{\ell+1}; \\ & \quad \text{If } \|p^{\ell+1}\| < \max(\varepsilon, \gamma \|p^0\|), \text{ go to the next time step;} \\ & \text{End} \end{array} \tag{4.14}$$

Remark. One can solve (4.8) for δv using e.g. PCG-ILU with the *zero* initial guess, which is equivalent to solve (2.5) by PCG-ILU with the initial guess v^1, where v^1 is the solution of (2.10) for $n = 0$. It has been observed that the resulting algorithm performs about 10-20% faster in computation time than PCG-ILU applied to (2.5).

Remark. We can employ the algorithm (4.14) for all time levels. It is easy to see from (4.12) and (4.13) that algorithm (4.10) reduces low-frequency components of the iterates p^ℓ (the error) more rapidly, while most standard iterative algorithms decrease high-frequency components of the error more efficiently. This observation suggests that (4.14b) can be combined with a few smoothing iterations of e.g. the Gauss-Seidel (GS) method, to improve the performance of the algorithm. For numerical comparisons, it is implemented and called ADG(ε, γ; m), where m is the number of GS iterations.

Remark. The truncation error in (4.5) is one order higher in k than the discretization error, as mentioned earlier. However, the operator Q includes approximations of second-order spatial derivatives. So, the truncation error can be large for the solutions of large curvatures, i.e. for oscillatory solutions. As the same reason as in the last remark, ADI-II can be followed by m GS iteration. We call the resulting algorithm ADI-II(m).

5 Numerical experiments

Choose the domain $\Omega = (0,1)^2$ and the time interval $J = (0,1]$. The diffusion coefficients are selected as

$$\begin{aligned} a_1(x,y) &= 1, \\ a_2(x,y) &= 1/(2+\cos(3\pi x)\cdot\cos(2\pi y)), \\ a_3(x,y) &= \begin{cases} 1+0.5\cdot\sin(5\pi x)+y^3, & \text{if } x \le 0.5, \\ 1.5/(1+(x-0.5)^2)+y^3, & \text{else,} \end{cases} \\ a_4(x,y) &= \begin{bmatrix} a_2(x,y) & 0 \\ 0 & a_3(x,y) \end{bmatrix}. \end{aligned} \tag{5.1}$$

For the correction step of (4.14), we choose $\varepsilon = 10^{-5}$, $\gamma = 0.1$, and four GS iterations follow ($m = 4$). The first time step of (4.4) is performed by utilizing (4.14). Also we try to solve all time steps by using (4.14); the resulting algorithm is called ADG(1e-5,0.1;4). For comparisons, we implement six different algorithms to solve the problem in each time level: LU, PCG-ILU, ADI(DPR), ADG($\varepsilon, \gamma; m$), ADI-II, and ADI-II(m).

Table 3 presents the performances of the algorithms for various diffusion coefficients. The ADI method deteriorates the solution, whose error is often about a digit larger than the original truncation error. ADG(1e-5,0.1;4) performs average 3-4.5 ADI correction iterations for a time step; it seems eliminating more than 95% of the perturbation error, for the extra cost of about 60-80% of the standard ADI. ADI-II requires only about 5-7% extra cost over the ADI method; it shows the accuracy less than 1% different from that of the direct solver. ADI-II is superior to ADG(1e-5,0.1;4) in accuracy and computation time.

	$a = a_1$		$a = a_2$		$a = a_3$	
	CPU	L^2-error	CPU	L^2-error	CPU	L^2-error
LU	44.6	1.10e-3	52.7	3.53e-3	45.6	5.35e-3
PCG-ILU	35.1	1.10e-3	40.2	3.52e-3	40.6	5.36e-3
ADI	13.4	1.70e-2	20.0	1.02e-2	14.5	2.67e-2
ADG(1e-5,0.1;4)	25.3	1.13e-3	29.7	3.55e-3	26.6	5.37e-3
ADI-II	14.2	1.10e-3	20.9	3.54e-3	15.5	5.35e-3

Table 3: The performances of LU, PCG-ILU, ADI(DPR), ADG(1e-5,0.1;4), and ADI-II. For the computation, $c = \beta \equiv 0$, $\nu_t = 1$, $\nu_x = 4$, $\nu_y = 3$, $n = 100$, and $u = u_\times$.

	$k = 2h$		$k = h$		$k = h/2$	
	CPU	L^2-error	CPU	L^2-error	CPU	L^2-error
PCG-ILU	44.8	2.14e-3	67.1	2.15e-3	109.6	2.14e-3
ADI	15.1	2.01e-1	29.8	6.76e-2	58.5	1.75e-2
ADI-II	16.0	1.10e-2	31.1	2.16e-3	61.5	2.13e-3
ADI-II(4)	19.8	2.13e-3	38.5	2.12e-3	75.9	2.13e-3

Table 4: The performances of PCG-ILU, ADI(DPR), ADI-II, and ADI-II(4). Set $a = a_4$, $c = \beta \equiv 0$, $\nu_t = 2.0$, $\nu_x = 6.25$, $\nu_y = 7$, $h = h_x = h_y = 1/120$, and $u = u_\times$.

Table 4 shows numerical results for various time steps k. Choose $a = a_4$ (an anisotropic diffusivity), $c = \beta \equiv 0$, $\nu_t = 2.0$, $\nu_x = 6.25$, and $\nu_y = 7$. The discretization selects $h = h_x = h_y = 1/120$. It is clear from the table that ADI-II is suffering accuracy for large time steps k. It is expected! To improve its performance, we introduce ADI-II(m) that is ADI-II followed by m GS iterations. ADI-II(4) performs quite satisfactorily in accuracy for various time steps; it takes only one fourth extra computation time over the original ADI method, DPR.

For Table 5, the solution is selected as

$$u(t,x,y) = e^{-t} \cdot \sin(2\pi\nu_x x) \cdot \sin(2\pi\nu_y y), \quad \nu_x = \nu_y = 3.$$

The nonlinear reaction is given as

$$c(t,u) = -u^2 e^u / \sqrt{0.1 + t^2 + 2u^2}$$

and treated with either the fourth-order Runge-Kutta scheme (RK4),

$$C(U^n_{RK4}) = \frac{1}{6}\Big(\mathcal{C}^n_1 + 2\mathcal{C}^n_2 + 2\mathcal{C}^n_3 + \mathcal{C}^n_4\Big),$$

where

$$\begin{aligned}
\mathcal{C}^n_1 &= c(t^n, u^n),\\
\mathcal{C}^n_2 &= c(t^{n+1/2}, u^n + k\mathcal{C}^n_1/2),\\
\mathcal{C}^n_3 &= c(t^{n+1/2}, u^n + k\mathcal{C}^n_2/2),\\
\mathcal{C}^n_4 &= c(t^{n+1}, u^n + k\mathcal{C}^n_3),
\end{aligned}$$

	Linear ($c \equiv 0$)		$C(U^n_{RK4})$		$C(EU^n)$	
	CPU	L^2-error	CPU	L^2-error	CPU	L^2-error
LU	20.9	4.06e-3	25.2	1.94e-2	22.6	3.96e-3
PCG-ILU	16.1	4.07e-3	19.6	1.96e-2	16.9	3.97e-3
ADI-II(4)	12.3	4.06e-3	15.9	1.94e-2	13.6	3.96e-3

Table 5: The performances of LU, PCG-ILU, and ADI-II(4) for nonlinear problem. Set $a = a_4$, $\beta \equiv 0$, and $n_t = n_x = n_y = 120$. The nonlinear reaction is given as $c(t, u) = -u^2 e^u/(0.1 + t^2 + 2u^2)^{1/2}$.

or the extrapolation

$$C(EU^n) = \begin{cases} C(U^n_{RK4}), & n = 0, \\ c\Big(t^{n+1/2}, \dfrac{3}{2}u^n - \dfrac{1}{2}u^{n-1}\Big), & n \geq 1. \end{cases}$$

The linear case ($c \equiv 0$) is reported to see the performance of algorithms for exponentially decreasing solutions and to compare the effects of *incomplete iterations* for the nonlinear term. In the table the nonlinear solver with the extrapolation shows a slightly less error than the linear case, which is not common. We have found from various numerical experiments that the extrapolation for nonlinear *reaction* is accurate enough for most cases, in particular, when $k \leq h$. See e.g. [2, 9, 13, 14] for analyses of incomplete iterations. ADI-II(4) performs satisfactorily for all cases. It should be noticed that ADI-II(4) shows the same accuracy as the direct solver LU.

References

[1] S. Abarbanel, D. Dwoyer, and D. Gottlieb, *Improving the convergence rate to steady state of parabolic ADI methods*, J. Comput. Phys., 67 (1986), pp. 236–239.

[2] J. Bramble, J. Pasciak, P. Sammon, and V. Thomée, *Multistep backward difference methods for parabolic problems with smooth and nonsmooth data*, Math. Comp., 52 (1989), pp. 339–367.

[3] R. Chin, T. Manteuffel, and J. De Pillis, *ADI as preconditioning for solving the convection-diffusion equation*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 291–299.

[4] C. Chiu and N. Walkington, *An ADI method for hysteretic reaction-diffusion systems*, SIAM J. Numer. Anal., 34 (1997), pp. 1185–1206.

[5] H. De Vries, *A comparative study of ADI splitting methods for parabolic equations in two space dimensions*, J. Comput. Appl. Math., 10 (1984), pp. 179–193.

[6] J. Douglas, Jr., *On the numerical itegration of* $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t}$ *by implicit methods*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 42–65.

[7] ——, *Alternating direction iteration for mildly nonlinear elliptic differential equations*, Numer. Math., 3 (1961), pp. 92–98.

[8] ——, *Alternating direction methods for three space variables*, Numer. Math., 4 (1961), pp. 41–63.

[9] ——, *On incomplete iteration for implicit parabolic difference equations*, J. Soc. Indust. Appl. Math., 9 (1961), pp. 433–439.

[10] ——, *Alternating direction methods for three space variables*, Numer. Math., 4 (1962), pp. 41–63.

[11] ——, *Alternating-direction iteration for the p-version of the finite element method*, in Partial Differential Equations and Applications, vol. 177 of Lecture Notes in Pure and Appl. Math., Dekker, New York, 1996, pp. 121–135.

[12] J. Douglas Jr. and T. Dupont, *Alternating direction Galerkin methods on rectangles*, in Numerical Solution of Partial Differential Equations-II, B. Hubbard, ed., Academic Press, New York, 1971, pp. 133–214.

[13] J. Douglas Jr., T. Dupont, and R. Ewing, *Incomplete iteration for time-stepping a Galerkin method for a quasilinear parabolic problem*, SIAM J. Numer. Anal., 16 (1979), pp. 503–522.

[14] J. Douglas, Jr., T. Dupont, and P. Percell, *A time-stepping method for Galerkin approximations for nonlinear parabolic equations*, in Numerical Analysis, Dundee 1977, vol. 630 of Lecture Notes in Mathematics, Springer-Verlag, Berlin, 1978.

[15] J. Douglas, Jr., R. Durán, and P. Pietra, *Alternating-direction iteration for mixed finite element methods*, in Numerical Approximation of Partial Differential Equations, E. Ortiz, ed., North-Holland, Ansterdam, 1987, pp. 21–30.

[16] J. Douglas, Jr. and J. Gunn, *A general formulation of alternating direction methods Part I. Parabolic and hyperbolic problems*, Numer. Math., 6 (1964), pp. 428–453.

[17] J. Douglas, Jr. and D. Peaceman, *Numerical solution of two-dimensional heat flow problems*, American Institute of Chemical Engineering Journal, 1 (1955), pp. 505–512.

[18] J. Douglas, Jr. and H. Rachford, *On the numerical solution of heat conduction problems in two and three space variables*, Transaction of the American Mathematical Society, 82 (1960), pp. 421–439.

[19] D. Evans and G. Avelas, *The solution of elliptic partial differential equations in R–θ geometry by extrapolated A.D.I. methods*, Math. Comput. Simulation, 23 (1981), pp. 367–372.

[20] A. Jarzebsky and J. Thullie, *A stable highly accurate ADI method for hyperbolic heat conduction equation*, J. Comput. Phys., 63 (1986), pp. 236–239.

[21] S. Johnson, Y. Saad, and M. Schultz, *Alternating direction methods on multiprocessors*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 686–700.

[22] S. Kim, *A parallelizable iterative procedure for the Helmholtz problem*, Appl. Numer. Math., 14 (1994), pp. 435–449.

[23] ——, *Domain decomposition iterative procedures for solving scalar waves in the frequency domain*, Numer. Math., 79 (1998), pp. 231–259.

[24] D. Kwak, *A norm estimate for the ADI method for nonsymmetric problems*, Linear Algebra and Its Applications, 266 (1997), pp. 127–141.

[25] B. Li, G. Fairweather, and B. Bialecki, *Discrete-time orthogonal spline collocation methods for Schrödinger equations in two space variables*, SIAM J. Numer. Anal., 35 (1998), pp. 453–477.

[26] S. Malhotra, C. Douglas, and M. Schultz, *Parameter choices for ADI-like methods on parallel computers*, Comp. Appl. Math., 17 (1998), pp. 221–236.

[27] D. Peaceman and H. Rachford, *The numedrical solution of parabolic and elliptic equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28–41.

[28] C. Pearcy, *On convergence of alternating direction procedures*, Numer. Math., 4 (1962), pp. 172–176.

[29] G. Starke, *Alternating direction preconditioning for nonsymmetric systems of linear equations*, SIAM J. Sci. Comput., 15 (1994), pp. 369–384.

[30] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Wadsworth & Brooks/Cole, Pacific Grove, California, 1989.

[31] P. Tsompanopoulou and E. Vavalis, *ADI methods for cubic spline collocation discretizations of elliptic PDEs*, SIAM J. Sci. Comput., 19 (1998), pp. 341–363.

[32] R. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, 1962.

[33] E. Wachspress, *Optimum alternating-direction-implicit iteration parameters for a model problem*, J. Soc. Indust. Appl. Math., 10 (1962), pp. 339–350.

[34] E. Wachspress and G. Habetler, *An alternating-direction-implicit iteration technique*, J. Soc. Indust. Appl. Math., 8 (1960), pp. 403–424.