JOURNAL OF COMPUTATIONAL PHYSICS **101**, 11–24 (1992)

Solution of Simultaneous Partial Differential Equations Using Dynamic ADI: Solution of the Streamlined Darwin Field Equations

D. W. HEWETT, D. J. LARSON, AND S. DOSS

University of California, Lawrence Livermore National Laboratory, Livermore, California 94550

Received October 1, 1990; revised July 15, 1991

We apply a particular version of ADI called Dynamic ADI (DADI) to the strongly coupled 2nd-order partial differential equations that arise from the streamlined Darwin field (SDF) equations. The DADI method is applied in a form that we show is guaranteed to converge to the desired solution of the finite difference equation. We give overviews of our test case, the SDF problem, and the DADI method, with some justification for our choice of operator splitting. Finally, we apply DADI to the strongly coupled SDF equations and present the results from our test case. Our implementation requires a factor of 7 less storage and has proven to be a factor of 4 (in the worst case) to several orders of magnitude faster than competing methods. © 1992 Academic Press, Inc.

I. INTRODUCTION

In computational physics one frequently encounters the need to solve a strongly coupled set of partial differential equations. Examples can readily be found [1–5] in solutions of Maxwell's equations and their various approximations needed in the simulation of plasma physics phenomena. Our desire to model laboratory plasma behavior is the source of our difficulties: if we were content to study small time and spatial scales, the time honored leapfrog integration of Maxwell's equations provides an excellent solution—though some issues regarding outflow boundary conditions on oblique boundaries are troublesome. Inevitably, time or spatial scales are enlarged and the bounds of affordable plasma simulation are encountered. Fortunately many important macroscopic plasma studies do not require the richness of the full electromagnetic model. Implicit PIC methods would seem to provide relief in these situations because large time steps allow unneeded high-frequency physics to damp away. However, implicit codes encounter similar bounds of affordability; though an irrelevant timescale need not be resolved, inevitably some computational machinery must be exercised just in case the user chooses a small time step.

Much of computational plasma physics is addressed to finding appropriate "reduced" models that trade increased computational complexity for the complete neglect of selected space and time scales. The increased complexity usually shows up in model equations that include more and stronger coupling to each other. As time steps increase it becomes more difficult to freeze one variable while we advance another. Implicit formulations of the model are the answer, in principle, but the numerical solution must be efficient enough to provide a reward for the increased computational effort required. In our considerations of these issues we have exploited an old method in a new way that is proving useful for these strongly coupled equations.

Our particular motivation in the work reported here has been to find an effective yet affordable method to apply in simulation models employing the streamlined Darwin field (SDF) equations [5]. The Darwin approximation of Maxwell's equations has been discussed by several authors; the essence of the model is that the part of the displacement current that gives rise to purely electromagnetic (EM) oscillations is ignored. Dropping this term is equivalent to ignoring the retardation that generates these modes. The beneficial effects are that the restrictive Courant condition on EM propagation need not be satisfied and that the coupling of plasma source fluctuations in noisy PIC representations does not lead to excessive energy loss to purely EM modes.

A problem with the traditional Darwin approach is that boundary conditions are required that are beyond the usual limits of physical intuition. There are two levels of difficulties. The most straightforward level comes from our need to break up the **E** into its component solenoidal (electromagnetic) and irrotational (electrostatic) parts—introducing some ambiguity into the boundary conditions needed for each part so that the sum represents the correct physics. The other level is less obvious. At the heart of the calculation for the solenoidal **E** field is an equation that requires the solenoidal part of a plasma source term, the time derivative of the total plasma current, that is itself derived from other plasma source terms and field quantities. To decompose this source term properly requires boundary conditions that are interrelated to several others in both time and space. The complexity that results has severely restricted the utility of the Darwin model—leading to the usual trick of moving the plasma "far away" from all boundaries. Unfortunately, for plasma "near" physical boundaries such issues cannot be ignored.

The purpose of the SDF formulation is to provide an alternative solution path for the Darwin model that obviates the most troubling boundary conditions. The cost for boundary condition relief is that the equations for the inductive electric field become even more strongly coupled. The purpose of this paper is to present an uncommon use of the alternating direction implicit (ADI) technique that provides a clean, straightforward solution, without iteration between the several equations. The method works so well that we expect there to be many other applications, though no others will be presented here.

The plan of this paper is as follows: in Section II we present an abbreviated derivation of the Darwin inductive electric field equations and show how the SDF formulation eliminates the least intuitive of the boundary conditions. The SDF formulation is then followed to derive the coupled equations for our test problem. Section III gives an intuitive view of how the dynamic ADI algorithm works and how we have applied it to the coupled equations of Section II. Finally in Section IV, we give our results and a brief comparison with the BCG method applied to this same problem.

II. THE INDUCTIVE E EQUATIONS IN THE DARWIN APPROXIMATION

II.A. Motivation

Boundary conditions pose significant difficulties with the traditional Darwin model for all but the trivial case where all boundary conditions are assumed far from the plasma so that boundary condition choices are insignificant. These difficulties arise at several points in the model but are nowhere more acute than in the algorithm used to compute the inductive electric field.

Before starting a discussion of the equations, a few definitions are given. From Helmholtz's theorem any vector may be represented as the sum of solenoidal and irrotational parts. The solenoidal part of a vector has no divergence and could be obtained from the curl of a vector potential, and the irrotational part has no curl and might come from the gradient of a scalar potential. We can always find such potentials though we may choose not to for convenience. We use the subscript *sol* to refer to the solenoidal part and the subscript *irr* to refer to the irrotational part. Vector decomposition of a typical vector $\mathbf{J}$ is accomplished by defining $\mathbf{J}_{\text{irr}}$ and $\mathbf{J}_{\text{sol}}$ with

$$\mathbf{J}_{\text{sol}} = \mathbf{J} - \mathbf{J}_{\text{irr}} = \mathbf{J} + \nabla\psi, \tag{1a}$$

where

$$\nabla^2\psi = -\nabla\cdot\mathbf{J}. \tag{1b}$$

Generally the boundaries are not far from the physics of interest and therefore the choice of boundary conditions is important. Unfortunately, boundary conditions that properly represent the physics can easily become the hardest part of the problem. Efforts to properly represent a plasma–wall interaction generally use a Neumann condition on one or more of the simulation boundaries. This condition has the form

$$\nabla_{\text{normal}}\psi = -\mathbf{J}_{\text{irr}}\cdot\hat{\mathbf{n}} \tag{1c}$$

which corresponds physically to the normal component of the irrotational part of $\mathbf{J}$.

The traditional Darwin model has been motivated and discussed by several authors [4–8]. The reader is directed to the work of Nielson and Lewis [6] and the streamlined version of Hewett and Boyd [5] as prerequisites for the brief details and summary we give here. We start with the Darwin limit of Maxwell's equations in which the solenoidal part of the displacement current has been discarded:

$$\begin{aligned}\nabla\times\mathbf{E} &= -\frac{1}{c}\frac{\partial\mathbf{B}}{\partial t}\\ \nabla\times\mathbf{B} &= \frac{4\pi}{c}\mathbf{J} + \frac{1}{c}\frac{\partial\mathbf{E}_{\text{irr}}}{\partial t} = \frac{4\pi}{c}\mathbf{J}_{\text{sol}}.\end{aligned} \tag{2}$$

Introducing the solenoidal vector potential $\mathbf{A}$, such that $\mathbf{B} = \nabla\times\mathbf{A}$ and $\nabla\cdot\mathbf{A} = 0$, we have

$$\nabla^2\mathbf{A} = -\frac{4\pi}{c}\mathbf{J}_{\text{sol}}, \tag{3}$$

and the equation for the electrostatic potential Φ is the usual

$$\nabla^2\mathbf{\Phi} = -4\pi\rho. \tag{4}$$

Using $\mathbf{E} = -\nabla\mathbf{\Phi} - (1/c)\dot{\mathbf{A}}$, we obtain the irrotational part of the electric field from the gradient of $\mathbf{\Phi}$. The equation for $\mathbf{E}_{\text{sol}}$ comes from the time derivative of Eq. (3) with the above definition of $\mathbf{E}$, resulting in

$$\nabla^2\mathbf{E}_{\text{sol}} = \frac{4\pi}{c^2}\dot{\mathbf{J}}_{\text{sol}}. \tag{5}$$

As in Nielson and Lewis,

$$\frac{4\pi}{c^2}\dot{\mathbf{J}} = \mathbf{K} + \zeta \times \mathbf{B} + \mu(\mathbf{E}_{\text{irr}} + \mathbf{E}_{\text{sol}}). \tag{6a}$$

We define Q such that

$$\frac{4\pi}{c^2}\dot{\mathbf{J}} = \mathbf{Q} + \mu \mathbf{E}_{\text{sol}}, \tag{6b}$$

so that Eq. (5) becomes

$$\nabla^2 \mathbf{E}_{\text{sol}} = [\mathbf{Q} + \mu \mathbf{E}_{\text{sol}}]_{\text{sol}}. \tag{7}$$

These equations comprise the Darwin limit of Maxwell's equations. They require boundary conditions that are, with the notable exception of Eq. (7), physically motivated but often strongly interrelated [5].

Equation (7) has proven difficult to solve—especially for large μ. The limit of large μ corresponds to an important parameter regime for the Darwin model. The parameter μ is the sum over species of the square of the plasma frequencies divided by the square of the speed of light—the inverse of the square of the collisionless electromagnetic skin depth. Comparing the scaling of the terms in Eq. (7), we see that the ratio of the μ term to the ∇^2 term is a measure of spatial resolution of the skin depth, with the small μ case corresponding to good resolution. If forced to work with the radiationless Darwin model only in the small μ limit by numerical solution difficulties, we will have lost a large part of the advantage afforded by the Darwin model. Thus, we examine the difficulties more carefully.

Difficulties arise in solving Eq. (7) in at least two ways. First, Eq. (7) must be solved by iteration because the unknown appears on the right-hand side. One takes the best guess for $\mathbf{E}_{\text{sol}}$, plugs it into the right-hand side, forms the quantity in the brackets, vector decomposes it, and finally solves for a new $\mathbf{E}_{\text{sol}}$. Adding to the confusion is the fact that $\mathbf{E}_{\text{sol}}$ may itself develop an irrotational part that must be removed before the next iteration. This procedure is, at best, messy and often converges very slowly.

Second, and the major stumbling block for many applications, the decomposition of $\dot{\mathbf{J}}$ (the quantity in the bracket) requires the specification of physically obscure boundary conditions. As in the discussion leading to Eqs. (1),

$$\dot{\mathbf{J}}_{\text{sol}} = \dot{\mathbf{J}} - \dot{\mathbf{J}}_{\text{irr}} = [\mathbf{Q} + \mu \mathbf{E}_{\text{sol}}] + \nabla\psi \tag{8a}$$

$$\nabla^2\psi = -\nabla \cdot \dot{\mathbf{J}} = -\nabla \cdot [\mathbf{Q} + \mu \mathbf{E}_{\text{sol}}]. \tag{8b}$$

To solve this Poisson equation, one needs to specify a boundary condition on all surfaces. The Neumann condition physically corresponds to specifying the *normal component of the irrotational part of* $\dot{\mathbf{J}}$—which is certainly beyond the usual limits of our physical intuition. Fortunately symmetry conditions often allow boundary conditions on some boundaries to be Dirichlet—implying that the normal component or slope of the function is determined by the requirements of the governing equations. However, it is very easy to imagine situations, such as the injection of a particle beam of finite extent or an absorption process, for which only Neumann conditions give adequate control of the physics. The SDF procedure discussed in the next section eliminates the need for this decomposition of $\dot{\mathbf{J}}$ and thus any further concern over these issues.

II.B. The SDF Formulation

As discussed in the previous section, the SDF formulation eliminates the least intuitive of the required boundary conditions. The procedures needed to accomplish this were described by Hewett and Boyd [5] and we give here a brief summary. We begin with the following manipulations of Eq. (5):

$$\nabla^2 \mathbf{E}_{\text{sol}} = \frac{4\pi}{c^2}(\dot{\mathbf{J}} - \dot{\mathbf{J}}_{\text{irr}}) \tag{9}$$

$$\nabla^2 \mathbf{E}_{\text{sol}} + \frac{4\pi}{c^2}\dot{\mathbf{J}}_{\text{irr}} = \frac{4\pi}{c^2}\dot{\mathbf{J}}. \tag{10}$$

Defining

$$-\nabla^2(\nabla\psi) \equiv \frac{4\pi}{c^2}\dot{\mathbf{J}}_{\text{irr}}, \tag{11}$$

we have

$$\nabla^2(\mathbf{E}_{\text{sol}} - \nabla\psi) = \frac{4\pi}{c^2}\dot{\mathbf{J}}. \tag{12}$$

Comparing with Eq. (5), Eq. (12) contains a new term on the left-hand side that just compensates for the irrotational term that we have retained on the right. Defining

$$\xi \equiv \mathbf{E}_{\text{sol}} - \nabla\psi, \tag{13}$$

we have

$$\nabla^2\xi = \frac{4\pi}{c^2}\dot{\mathbf{J}}. \tag{14}$$

Using Eq. (6b) we may further modify Eq. (14) as

$$\nabla^2\xi = \mathbf{Q} + \mu \mathbf{E}_{\text{sol}}$$

$$\nabla^2\xi - \mu\xi = \mathbf{Q} + \mu\nabla\psi.$$

From the definition equation (13) we note that

$$\xi_{\text{irr}} = -\nabla\psi,$$

so that

$$\nabla^2\psi = -\nabla \cdot \xi.$$

The equation for $\mathbf{E}_{\text{sol}}$, Eq. (5), then becomes the two coupled equations

$$\nabla^2\xi - \mu\xi = \mathbf{Q} + \mu\,\nabla\psi \tag{15}$$

$$\nabla^2\psi = -\nabla \cdot \xi. \tag{16}$$

What is gained by this change? Most importantly, the required boundary conditions are simple. Since we only care about $\nabla\psi$ on the boundaries rather than ψ itself, the boundary conditions for Eq. (16) can be Dirichlet zero. For Eq. (15) we use

$$\xi = \mathbf{E}_{\text{sol}}(\text{specified}) - \nabla\psi \tag{17}$$

or the normal derivative of this equation—no more difficult than that required for Eq. (5). Now we do have coupled equations over which we must iterate—but this is much easier than all the decomposition that is done to solve Eq. (7) the traditional way. Hewett and Boyd proved that this system works, and Boyd [8] has made this work on real problems in cylindrical geometry using a Picard iteration over the set of equations. What was missing was a quick, reliable procedure for solving these coupled equations. We present in Section III a new technique based on ADI that satisfies these requirements.

A recent paper by Weitzner and Lawson [9] addresses a related problem. They are concerned with how one selects $\mathbf{E}_{\text{sol}}$ (specified). Their solution determines the boundary condition by performing a global minimization of $\mathbf{E}_{\text{sol}}$ over the entire simulation region. The rationale is that the Darwin limit minimizes the amount of energy in purely inductive fields. We agree with this motivation though our intuition, based on the concept that $\mathbf{E}_{\text{irr}}$ field lines end on charge while $\mathbf{E}_{\text{sol}}$ must be closed loops, has not yet failed.

Finally, we note that although we are working in the Coulomb gauge of the magnetic vector potential $\mathbf{A}$, we have not needed to specify the gauge of the new vector ξ. The final set of equations, Eqs. (15) and (16), do specify the gauge of ξ; Eq. (16) is the gauge condition equation. As with any gauge condition, the physics that determines $\mathbf{E}_{\text{sol}}$ is independent of gauge. Here the gauge of the equations is related in a complicated way to the irrotational part of $\mathbf{Q}$ and the gradient of μ.

III. SDF SOLUTION TECHNIQUES

If the coupling between the two vector components of ξ and scaler ψ is relatively weak, $\mu \ll \nabla^2$, the obvious choice is Picard iteration—that is, using a guess for ψ and solving for each component of ξ then using these latest values for ξ and solving (16) for ψ. This iteration is repeated in turn until both equations are satisfied. When μ is not small compared to other terms in the equations or for cases in which μ has strong gradients, this method converges slowly at best. We are interested in the physics for cases in which μ is not small relative to other terms. When μ is significant we are compelled, by slow convergence, to find some method which will determine both components of ξ and ψ simultaneously. Even using present day computers with large memory it is unreasonable to attempt to solve the full matrix that would result from straightforward finite differencing. All methods considered here make use of the banded nature of the matrix.

In the course of this investigation, our objectives changed. At first we were interested in finding any banded-matrix method, regardless of speed or memory requirements, that would find a solution to the set of strongly coupled SDF equations. We considered the biconjugate gradient (BCG) method made popular by Z. Mikic and E. Morse [10] and again by Anderson *et al.* [3]. Using the preconditioned biconjugate gradient solver CPDES2, developed by Anderson *et al.* [3], we readily obtained reliable, robust solutions. With solutions in hand, we next considered methods that might provide these solutions with much smaller CPU and memory requirements. Splitting or alternating direction implicit (ADI) methods offer reductions in memory requirements, though our first implementations were not very fast and certainly not robust. Later in the investigation, as we began to understand a version of ADI with dynamically determined acceleration factors called DADI, our method of choice shifted. As we will show, the DADI method, for similar accuracy, is significantly faster and always uses less storage in this application. We now give, in two subsections, (A) an intuitive discussion of DADI and (B) the details of the application of DADI as applied to our model problem.

III.A. An Intuitive View of the Dynamic ADI Algorithm

ADI or splitting algorithms are most attractive because they turn large sparse matrix problems into simple banded matrix problems. Thus ADI offers a considerable advantage over most other methods in reduced storage requirements. Other methods have claimed to have the advantage over ADI in CPU speed. Dynamic selection of the acceleration parameters, known here as DADI, in our recent experience, reduces and can reverse that advantage in situations with nonsymmetric operators. Our method does not depend on a "carefully" chosen preconditioner. We have not used any preconditioning in this work—though we may find increases in speed and such a techniques easily fit into the DADI scheme. An unattractive feature of ADI has been the care that must be taken in implementation to achieve a splitting scheme that converges, and converges to the desired solution. We now outline some of the techniques and guiding "rules-of-thumb" that have proven useful in implementing DADI.

III.A.1. *DADI Applied to the Poisson Equation*

Consider a simple Poisson equation such as Eq. (1b). Many types of iterative methods can be considered, conceptually at least, as the addition of a fictitious time derivative operator

$$\frac{-\partial\psi}{\partial t} + \nabla^2\psi = \sigma \tag{18}$$

that is then finite-differenced and stepped forward to the time-asymptotic state. We have replaced the right-hand side with an arbitrary source term σ for this discussion. If our iteration can deliver us to this asymptotic state, then the fictitious time derivative will vanish, the "iteration" is said to have converged, and we have the desired solution. ADI is simply a mechanism for time integration that attempts to achieve the asymptotic limit with an operator splitting procedure that exploits the banded matrix properties of the elliptic operator. The finite-differenced representation is

$$(-\omega + H)\,\psi^{n+1/2} = (-\omega - V)\,\psi^n + \sigma \tag{19a}$$

$$(-\omega + V)\,\psi^{n+1} = (-\omega - H)\,\psi^{n+1/2} + \sigma, \tag{19b}$$

where the superscript n is the iteration counter, $\omega = 2/\Delta t$, where Δt is the "time" step, and H and V represent the horizontal and vertical parts of the finite-differenced ∇^2 operator,

$$H\psi \equiv (\psi(i+1,j) - 2\psi(i,j) + \psi(i-1,j))/\Delta x^2 \tag{20a}$$

$$V\psi \equiv (\psi(i,j+1) - 2\psi(i,j) + \psi(i,j-1))/\Delta y^2, \tag{20b}$$

in this simple case. Conceptually, these ideas apply to more complex operators and in higher dimensions as well, though some of the strong convergence properties of ADI may be reduced. ADI gives some implicitness so that "large" time steps may be taken with some degree of robustness. A crucial part of this intuitive picture is the selection of the fictitious time step.

If we define

$$\delta\psi^{n+\alpha} \equiv \psi^{n+\alpha} - \psi^n \tag{21}$$

then Eqs. (19) become

$$(-\omega + H)\,\delta\psi^{n+1/2} = (-H - V)\,\psi^n + \sigma \tag{22a}$$

$$(-\omega + V)\,\delta\psi^{n+1} = -2\omega\delta\psi^{n+1/2}. \tag{22b}$$

With this formulation it is trivial to see that, if the iteration converges so that $\delta\psi^{n+1} = \delta\psi^{n+1/2} = 0$, we have a solution of the finite difference version of our equation. The right-hand side of Eq. (22a) is in fact the definition of the residual at level n.

If we further define the error after the nth iteration as

$$e^n \equiv \psi^n - \psi_{\text{exact}}$$

and if H and V commute then we have

$$e_j^{n+1} = \frac{(\omega + h_j)(\omega + v_j)}{(-\omega + h_j)(-\omega + v_j)}\, e_j^n, \tag{23}$$

where h_j and v_j are jth eigenvalues of the H and V operator matrices and e_j is the projection of the error on the jth eigenfunction. Since we know that $\omega > 0$ and both H and V are negative definite, both h_j and v_j are negative numbers and thus the coefficient of e_j^n will be less than 1 for all j and convergence is assured.

The final concern is the rate of convergence. The power of ADI is that, in principle, one may choose ω so that the eigenmode projection of the next error e^{n+1} may be made arbitrarily small. What remains is to define a scheme to optimally select ω as we progress towards the asymptotic state (convergence). The traditional ADI of Peaceman and Rachford [11] required estimates of the largest and smallest eigenvalues of the operator matrix. Their approach is to move the "acceleration" parameter ω as a function of the iteration parameter n in a cyclic fashion between the eigenvalue bounds. A more successful approach is that of Doss and Miller [12]. They describe a dynamic selection method that uses a comparison between a "single" H and V sequence (denoted by HV) followed by a double sequence consisting of HV, HV passes with half the "time" step. The L2 norm of the *difference* between the result of the HV and HVHV sequences is compared to the L2 norm of the *change* between the result of the HVHV sequence and the solution at the start of the iteration step. The ratio between these two L2 norms, RATIO = *difference*/*change*, then determines a multiplicative factor f_ω used to change ω for the next iteration.

We give in Table I the scheme for modifying ω as we use it. The idea is that the L2 norm of the difference gives a measure of the truncation error and the L2 norm of the change measures how much progress we are making towards the asymptotic state. The requirements conflict somewhat: if the difference norm in the numerator is large, the indication is that we are not resolving some time-

TABLE I

RATIO	f_ω	Comment
<0.02	0.125	Increase Δt by 8
0.05	0.250	Increase Δt by 4
0.10	0.500	Increase Δt by 2
0.30	1.250	Decrease Δt by 4/5
0.40	2.000	Decrease Δt by 2
>0.60	256.0	Dicard, restore previous level and decrease Δt by 256

dependent behavior and we need to reduce the "time step" (increase ω); however, if the change norm in the denominator is small, the concern is that the effective time step is too small for anything to change. A final observation is that near convergence, both numerator and denominator become small—requiring that the norms be computed with high precision. The analysis of Doss and Miller [12] determines an optimal range for RATIO to be the interval between 0.1 and 0.3. Finally, a most important feature is the provision for discarding an iterate completely should the ratio become larger than 0.6.

Some of the behavior of DADI can be discerned in Table II, where some results for this simple symmetric Poisson's equation are summarized. The first of the DADI entries is the result of a solution with the initial $\omega = 100.0$. At completion of the solution $\omega = 0.0477$. The second DADI entry displays the result for DADI started with $\omega = 0.0477$. Evidently some effort is wasted finding the optimal range for ω or Δt. The method does not require the operator matrix to be symmetric. For comparison we have given the results for a highly optimized ICCG algorithm that does take advantage of the symmetry—and requires 15 scratch arrays compared to DADI's four. Clearly, unless storage is an issue, one should use ICCG or perhaps cyclic reduction for this simple symmetric Poisson's equation just as one should use FFTs for periodic problems.

For more general problems ADI provides advantages. We have used this same DADI algorithm successfully in several other configurations. Nonlinear situations with $\sigma(\psi)$ converge more slowly than the simple equation above but are now certainly competitive with ICCG which here must use Picard iteration. DADI enjoys the advantage that $\sigma(\psi)$ may be updated as the iteration progresses. For reasons that will be discussed, ADI works well in these cases provided σ is recomputed only before the start of the H, V sequence and not updated in the middle. For examples of this type, we have found solutions for a strongly nonlinear exponential source function σ in our studies of the elongated field-reversed configuration [13]. This method also works well for situations that have strong cross derivatives in the equation for the charge correction step in the electromagnetic direct implicit PIC code AVANTI [2].

TABLE II

Method	Iterations	Residual	CPU (seconds)
DADI	11	2.60×10^{-11}	0.370
DADI	8	2.82×10^{-11}	0.263
ICCG	21	7.41×10^{-9}	0.039

III.A.2. *Applying DADI to More General Scalar Eliptic Equations*

We next consider a slightly more complicated equation. Adding a linear term with a multiplicative coefficient μ leads to the scalar Helmholtz equation. Adding this term to Eq. (18) gives

$$\frac{-\partial\psi}{\partial t} + \nabla^2\psi - \mu\psi = \sigma. \tag{24}$$

Again the method will be to finite-difference and integrate forward to the time-asymptotic state. The finite-differenced representation is

$$\left(-\omega + H - \frac{\mu}{2}\right)\psi^{n+1/2} = \left(-\omega - V + \frac{\mu}{2}\right)\psi^n + \sigma \tag{25a}$$

$$\left(-\omega + V - \frac{\mu}{2}\right)\psi^{n+1} = \left(-\omega - H + \frac{\mu}{2}\right)\psi^{n+1/2} + \sigma. \tag{25b}$$

The new term is called the Helmholtz term and the coefficient μ may have arbitrary spatial dependence.

As before, Eq. (25) can be written

$$\left(-\omega + H - \frac{\mu}{2}\right)\delta\psi^{n+1/2} = (-H - V + \mu)\,\psi^n + \sigma \tag{26a}$$

$$\left(-\omega + V - \frac{\mu}{2}\right)\delta\psi^{n+1} = -2\omega\delta\psi^{n+1/2} \tag{26b}$$

with the same property that, if the iteration converges, we have the desired solution of the finite difference version of our equation. The expression for e^{n+1} also has the additional terms

$$e_j^{n+1} = \frac{(\omega + h_j - \mu/2)(\omega + v_j - \mu/2)}{(-\omega + h_j - \mu/2)(-\omega + v_j - \mu/2)}\, e_j^n \tag{27}$$

that allow the same possibilities for rapid convergence as before.

One question that is frequently raised concerns the splitting of the μ term in equal parts on the left and right sides of Eqs. (25). If all we are attempting to accomplish is to find the zero residual (or the time-asymptotic limit) state, why not put all of the μ contribution on the left to make the set of equations "more" implicit? The answer is discovered by performing the same analysis as before with this new system—leading to this new form for the $\delta\psi$ formulae,

$$(-\omega + H - \mu)\,\delta\psi^{n+1/2} = (-H - V + \mu)\,\psi^n + \sigma \tag{28a}$$

$$(-\omega + V - \mu)\,\delta\psi^{n+1} = -(2\omega + \mu)\,\delta\psi^{n+1/2}. \tag{28b}$$

Note that, as with Eqs. (22), if the iteration converges, it will produce the desired solution. The problem with this form can be seen with the further reduction to the $n+1$ error formula that no longer enjoys the simple cancellation of binomial coefficients unless H and μ commute. The new form is

$$e_j^{n+1} = \frac{(\omega + h_j)(\omega + v_j)\,e_j^n - (H\mu - \mu H)\,e_j^{n+1/2}}{(-\omega + h_j - \mu)(-\omega + v_j - \mu)}. \tag{29}$$

It is easy to cast this problem into a more symmetric form that requires V and μ to commute but the point is the same: in the course of iteration, ψ may evolve into a spatial configuration such that an error introduced by the first pass is the only error fixed in the second pass.

This interplay is apparent from the $n + \frac{1}{2}$ level error term that does not cancel. The result is, especially for nonlinear equations, convergence to a residual value that cannot be made smaller by further iteration. The iteration has converged to a fixed point oscillation wherein the H and V passes just undo the changes made by each other. Often the value of the residual at which this oscillation begins is smaller than the error criterion and the user is never aware of the difficulty—adding to the frustration when it is noticeable. Further confusion can result from the fact that, until the oscillation is noticeable, the scheme with μ entirely on the implicit side often converges in fewer iterations. The conservative cure is to split the Helmholtz term as in the preceding discussion.

This analysis can be generalized to understand similar results that sometimes appear when first-order derivatives

$$\frac{-\partial\psi}{\partial t} + \nabla^2\psi + \frac{\partial\psi}{\partial x} + \frac{\partial\psi}{\partial y} = \sigma \tag{30}$$

are included. Using the additional notation for these first-order terms,

$$D_x\psi(i,j) \equiv (\psi(i+1,j) - \psi(i-1,j))/(2\Delta x) \tag{31a}$$

$$D_y\psi(i,j) \equiv (\psi(i,j+1) - \psi(i,j-1))/(2\Delta y), \tag{31b}$$

one might think that a direct extension of Eq. (23), the error expression for the scheme in Eqs. (19a) and (19b), would be appropriate. Denoting with d_x and d_y the "eigenvalues" of D_x and D_y, respectively, we obtain

$$e_j^{n+1} = \frac{(\omega + h_j - d_x)(\omega + v_j - d_y)}{(-\omega + h_j - d_x)(-\omega + v_j - d_y)}\,e_j^n. \tag{32}$$

Since d_x and d_y are purely imaginary, it would follow from Eq. (32) that $\|e_j^{n+1}/e_j^n\| < 1$. However, the catch here is that Eq. (32) cannot hold since $(H + D_x)$ does not commute with $(V + D_y)$ and, therefore, the analysis is not valid. Indeed we saw empirically that convergence was unpredictable.

A better option is to evaluate these first-order derivatives [illegible] expression is

$$e_j^{n+1} = \frac{(\omega + h_j)(\omega + v_j) - 2\omega(d_x + d_y)}{(-\omega + h_j)(-\omega + v_j)}\,e_j^n. \tag{33}$$

In general one cannot show that the magnitude of the coefficient in Eq. (33) is less than one. However, one observes that for both large and small ω this condition is satisfied. For very small ω (large Δt), DADI reduces the low frequency error rapidly and then shifts to the large ω limit and converges more slowly to the specified tolerance. In practice, DADI manages the choice of ω well with this option; this is, in fact, the splitting choice we use in the next section.

It is worth noting that this analysis gives some clues to the proper handling of nonlinear terms. It is frequently possible to subtract the linear part of nonlinear terms from both sides of the equation. The new contribution on the left is evaluated at the advanced time and its contribution is frequently split as in the analysis leading to Eq. (27) rather than that leading to Eq. (29). The remaining part of the nonlinear term stays on the right-hand side and is updated only at the end of a full double sweep. Our choices in this matter are influenced by analysis similar to that leading to Eq. (33) rather than Eq. (32). As is certainly obvious by now, there are no hard rules to handle nonlinearities with ADI and certain amount of testing is warranted.

III.B. Application of DADI to the Coupled Equations of Section II

We now discuss the application of this DADI algorithm to the coupled set of equations, Eqs. (15) and (16), for the inductive electric field in the SDF formulation. As we have discussed, when the coupling between these equations is strong, we need to solve these equations simultaneously.

Our notation describing the application to the two-dimensional SDF equations requires the definitions in

Eqs. (20) and (31). Using these definitions we split the system of three coupled scalar equations as

H-PASS

$$\begin{aligned}
&(-\omega + H - \mu/2)\,\xi_x^{n+1/2} \\
&\qquad = (-\omega - V + \mu/2)\,\xi_x^n + \mu D_x \psi^n + Q_x \\
&(-\omega + H - \mu/2)\,\xi_y^{n+1/2} \\
&\qquad = (-\omega - V + \mu/2)\,\xi_y^n + \mu D_y \psi^n + Q_y \\
&(-\omega + H)\,\psi^{n+1/2} = (-\omega - V)\,\psi^n - D_x \xi_x^n - D_y \xi_y^n
\end{aligned} \tag{34a}$$

V-PASS

$$\begin{aligned}
&(-\omega + V - \mu/2)\,\xi_x^{n+1} \\
&\qquad = (-\omega - H + \mu/2)\,\xi_x^{n+1/2} + \mu D_x \psi^n + Q_x \\
&(-\omega + V - \mu/2)\,\xi_y^{n+1} \\
&\qquad = (-\omega - H + \mu/2)\,\xi_y^{n+1/2} + \mu D_y \psi^n + Q_y \\
&(-\omega + V)\,\psi^{n+1} \\
&\qquad = (-\omega - H)\,\psi^{n+1/2} - D_x \xi_x^n - D_y \xi_y^n.
\end{aligned} \tag{34b}$$

As in simple ADI an iterated solution is obtained by solving the equations in the horizontal (H) pass, using the latest values for the second-order unknowns on the right side, followed by the vertical (V) pass, that uses the results from the previous H pass and so on, until convergence. Defining a vector Ξ of ordered unknowns,

$$\Xi = (\xi_x(1), \xi_y(1), \psi(1), \xi_x(2), \xi_y(2), \psi(2), \xi_x(3), \ldots),$$

the three equations under each H or V pass are solved simultaneously by a banded linear matrix solver

$$\begin{aligned}
\mathscr{H}\Xi^{n+1/2} &= \mathscr{V}\Xi^n + D\Xi^n + Q \\
\mathscr{V}\Xi^{n+1} &= \mathscr{H}\Xi^{n+1/2} + D\Xi^n + Q,
\end{aligned}$$

where $\mathscr{H}$ and $\mathscr{V}$ are banded matrices obtained from the left-hand side of Eqs. (34a) and (34b), respectively, using an ordering consistent with definition of Ξ. Convergence, for tests presented here, is defined to occur when both the L2 norm of the normalized change, $\|\Xi_{n+1} - \Xi_n\|/\|\Xi_n\|$, and the L2 norm of the residual $\|res\|/\|Q\|$, are less then a given error criterion.

Only one acceleration parameter was used for all three equations though each ω could be "preconditioned" by normalizing it to the diagonal coefficient in each equation at each point. All first-order derivative terms were placed on the right-hand side and updated only at the start of each iteration. An analysis similar to that used to obtain Eqs. (27) and (33) reveals when terms can be updated and still converge to the desired solution. The expression for the $n+1$ errors are

$$\begin{aligned}
e_{x,j}^{n+1} &= \frac{(\omega + h_{x,j} - \mu/2)(\omega + v_{x,j} - \mu/2)\,e_{x,j}^n - 2\omega\mu d_x e_{\psi,j}^n}{(-\omega + h_{x,j} - \mu/2)(-\omega + v_{x,j} - \mu/2)} \\
e_{y,j}^{n+1} &= \frac{(\omega + h_{y,j} - \mu/2)(\omega + v_{y,j} - \mu/2)\,e_{y,j}^n - 2\omega\mu d_y e_{\psi,j}^n}{(-\omega + h_{x,j} - \mu/2)(-\omega + v_{x,j} - \mu/2)} \\
e_{\psi,j}^{n+1} &= \frac{(\omega + h_{\psi,j})(\omega + v_{\psi,j})\,e_{\psi,j}^n + 2\omega\mu d_x e_{x,j}^n + 2\omega\mu d_y e_{y,j}^n}{(-\omega + h_{\psi,j})(-\omega + v_{\psi,j})}.
\end{aligned} \tag{35}$$

Note that we have split the $\mu\xi$ term just as we did for $\mu\psi$ in the scalar Helmholtz equation, Eq. (24). The first-order term $\mu D\psi$ is treated as we did for Eq. (30).

A rule of thumb is that terms should be computed implicitly on the left before that term is updated on the right. For example, convergence will degrade and perhaps simply oscillate at a residual level above our convergence criteria if one uses a $n + \frac{1}{2}$ value computed in the H pass in the first-order terms on the right-hand side of the V pass equations.

Another variant that provides better symmetry between H and V passes than Eqs. (34) is

H-PASS

$$\begin{aligned}
&(-\omega + H - \mu/2)\,\xi_x^{n+1/2} \\
&\qquad = (-\omega - V + \mu/2)\,\xi_x^n + Q_x + \mu D_x \psi^{n-1/2} \\
&(-\omega + H - \mu/2)\,\xi_y^{n+1/2} \\
&\qquad = (-\omega - V + \mu/2)\,\xi_y^n + Q_y + \mu D_y \psi^n \\
&(-\omega + H)\,\psi^{n+1/2} \\
&\qquad = (-\omega - V)\,\psi^n - D_x \xi_x^{n-1/2} - D_y \xi_y^n
\end{aligned} \tag{36a}$$

V-PASS

$$\begin{aligned}
&(-\omega + V - \mu/2)\,\xi_x^{n+1} \\
&\qquad = (-\omega - H + \mu/2)\,\xi_x^{n+1/2} + Q_x + \mu D_x \psi^{n+1/2} \\
&(-\omega + V - \mu/2)\,\xi_y^{n+1} \\
&\qquad = (-\omega - H + \mu/2)\,\xi_y^{n+1/2} + Q_y + \mu D_y \psi^n \\
&(-\omega + V)\,\psi^{n+1} \\
&\qquad = (-\omega - H)\,\psi^{n+1/2} - D_x \xi_x^{n+1/2} - D_y \xi_y^n
\end{aligned} \tag{36b}$$

that does provide increased performance for small μ. It does complicate the algorithm somewhat and does require the storage of ξ_x and ψ at the half iteration level. This variant is compared in the results section.

This symmetric version offers another possibility for the first-order terms. We explored the effect of making the first-order terms implicit on the appropriate pass; for example, in the H pass on the equation for $\xi_x^{n+1/2}$ we could easily change the term $\mu D_x \psi^{n-1/2}$ to $\mu D_x \psi^{n+1/2}$, thus making the iteration more implicit. This possibility gives a pentadiagonal rather than a tridiagonal system. Though this arrangement may ultimately prove to give the smallest CPU time to convergence, there are several obstacles in its implementation. A minor consideration is the additional storage; we need to store five diagonals rather than three, each of which is $3 \times N_x \times N_y$. A clue to the most crucial issue is the fact that we need to store the entire diagonal. It is necessary to store these elements because banded matrix solutions are inherently recursive. The whole matrix solution can be vectorized only by making the innermost DO loops over the other direction—and vectorization is imperative for unitasking speed. Though a scalar pentadiagonal method can easily be modified for this type of vectorization, we have no a priori evidence that it will converge faster; it will take longer per iteration and it will take more storage. As the reader may guess, we feel this method already has enough degrees of freedom and, as we show in the next section, the results using the tridiagonal DADI method are already quite attractive.

IV. TESTS AND COMPARISONS

IV.A. The Test Problem

To test the DADI method on a realistic problem relevant to our proposed Darwin application, we have contrived a test by specifying the answer $\mathbf{E}_{0\,\text{sol}}$ and then "deriving" the source function $\mathbf{Q}$ to be used in Eq. (15)—as in Ref. [5]. We then evaluate our solution procedures as they regenerate this solution. In this test case, we work on a Cartesian mesh with 39×31 mesh points in the x and y directions. The mesh spacing is uniform with $\Delta x = 0.526$ and $\Delta y = 0.667$. To

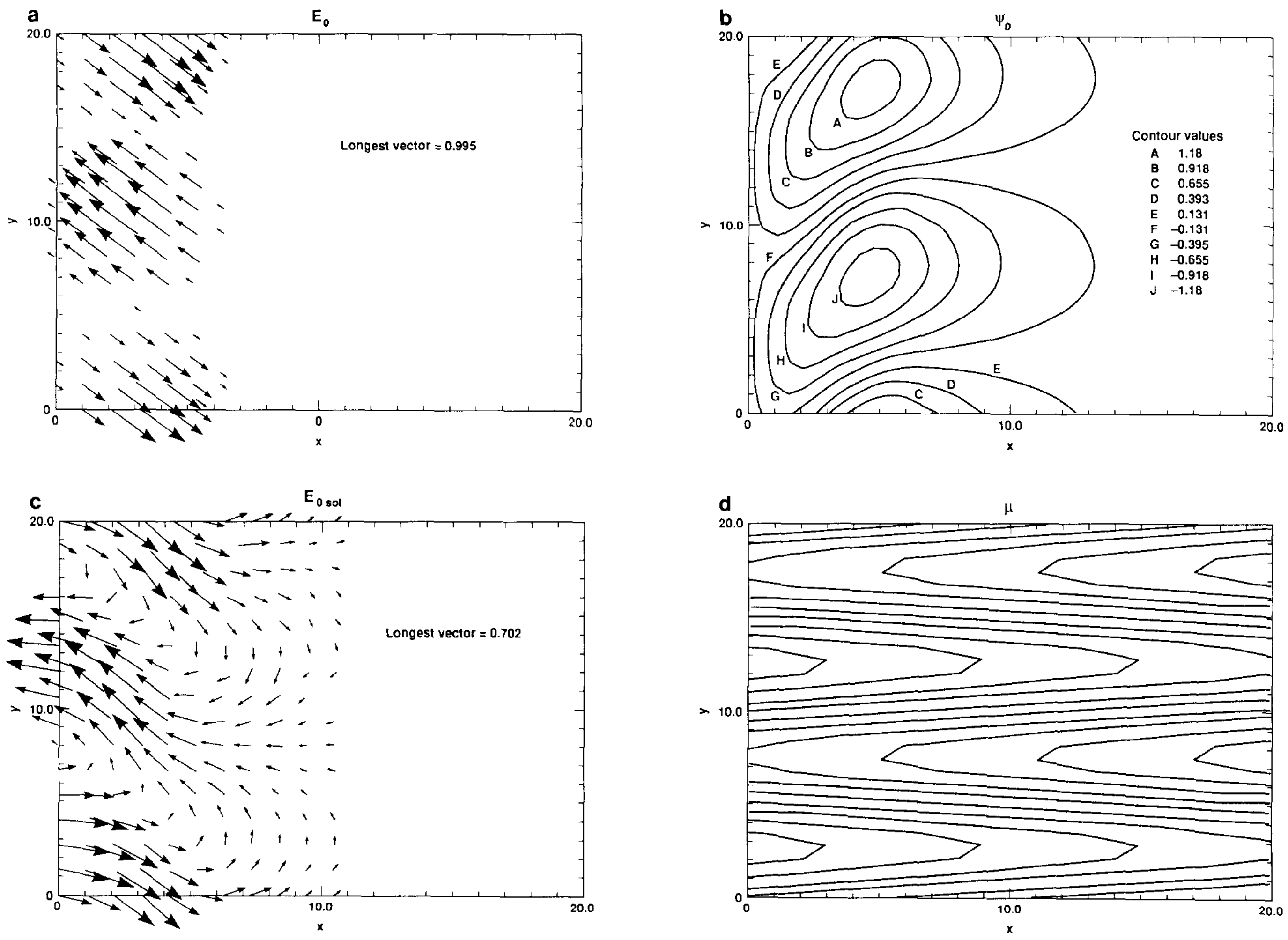


FIG. 1. These plots show graphically the functional forms used for the test case. (a) shows the initial choice of the E field; (b) shows the potential ψ_0 from which we derive the irrotational part of the initial E field in (a). (c) is a vector representation of the solenoidal part $\mathbf{E}_{\text{sol}}$ of the original $\mathbf{E}_0$—obtained by subtracting the irrotational part from the original. This vector field $\mathbf{E}_{0\,\text{sol}}$ serves as the desired solution. The coupling function μ with $\mu_0 = 1.0$ used in this test case is shown in (d).

minimize boundary condition difficulties we use periodic boundary conditions in y.

For the tests considered here, we have chosen the desired

$$\mathbf{E} = \exp(-(x-a)^2/5)(\hat{\mathbf{e}}_x - \hat{\mathbf{e}}_y)\cos(\pi y/10).$$

The solenoidal field itself is obtained by decomposition using the procedure given in Eqs. (1). Shown in Fig. 1 are graphical representations of this process. In Fig. 1a we have a vector plot of the total **E** field followed, in Fig. 1b, by a contour plot of the potential function that is consistent with the irrotational part of the total field. In Fig. 1c is a vector plot of the solenoidal field $\mathbf{E}_{\text{sol}}$ that is the "answer"—here denoted as $\mathbf{E}_{0\,\text{sol}}$. Using this field we can construct, using finite differencing, $\nabla^2\mathbf{E}_{\text{sol}}$ that, using Eq. (5), determines $\dot{\mathbf{J}}_{\text{sol}}$.

The next step is to determine a value for **Q**. If we take the form of $\dot{\mathbf{J}}_{\text{sol}}$ that we have just constructed as the source term $\dot{\mathbf{J}}$, then we can construct the **Q** array using Eq. (6b) as soon as we specify μ. The physics of the model tells us that μ will be related to the plasma density and thus positive definite: we choose the form

$$\mu(i,j) = \mu_0(x(i)/20 + 1 - \sin(\pi y(j)/5)).$$

A contour of this form is given in Fig. 1d. We now have all the source-related arrays needed for the solution of Eqs. (15) and (16) initialized. The test is to see how well the solution of Eqs. (15) and (16) followed by (17) agrees with the original $\mathbf{E}_{0\,\text{sol}}$.

IV.B. The Preconditioned Biconjugate Gradient BCG Method

The biconjugate gradient method [3], a generalization of the conjugate gradient method, is an iterative method that solves the nonsymmetric matrix equation $\mathbf{Ax} = \mathbf{b}$. The convergence rate depends upon the condition number of **A** and the extent to which the eigenvalues of **A** are "clustered." A large condition number will yield poor convergence. If the eigenvalues are closely grouped the condition number will be small and rapid convergence will result. The BCG method does not possess a minimization property as does CG; thus, a monotonic decrease of error is not guaranteed and the possibility of breakdown exists. However, for most cases BCG yields faster convergence than applying CG to the normal (squared matrix) equations.

The preconditioning used in the CPDES2 package is incomplete LU factorization, a less elaborate procedure than that required for the CG method. Preconditioning transforms the equation $\mathbf{Ax} = \mathbf{b}$ into $\mathbf{My} = \mathbf{c}$, such that the eigenvalues of **M** are more clustered than those of **A**. For ILU preconditioning, $\mathbf{M} = \mathbf{BA}$, where **B** is an approximate inverse of **A**. Thus **M** is approximately a unit matrix with condition number $\simeq 1$. By specifying a small number of parameters, the routine will automatically generate the

task of setting up the coefficient matrix **A** straightforward.

It is necessary to dimension the number of non-trivial diagonals to the maximum number possible for the most general coupling allowed by the 5- or 9-point operator stencil. In our case there were 19 non-zero diagonals; however, we were using a 5-point operator stencil and it turned out that the required number of non-trivial diagonals was 31. Because each diagonal represents three full 2D arrays, this extra storage is of concern when using machines with small memory or unfavorable memory charging algorithms. Our application of CPDES2 required the storage of 72×3 full 2D arrays—though at least 24 of these arrays remain unused by this application.

IV.C. Results

In Table III we summarize the results of our studies carried out on the **E** machine at NERSC, a Cray XMP, using the standard CFT compiler. We give, for BCG and two variants of DADI applied to our test problem, the number of iterations **iter**, the largest residual **res** of any equation across the entire mesh, the largest difference **dif**

TABLE III

Method	μ_0	Iter	Res	Dif	CPU
DADI	0	13	1.21×10^{-4}	5.43×10^{-2}	0.486
DADI′	0	13	1.21×10^{-4}	5.43×10^{-2}	0.485
BCG	0	42	7.63×10^{-4}	5.43×10^{-2}	5.20
DADI	1	55	5.06×10^{-4}	3.23×10^{-2}	2.14
DADI′	1	40	8.83×10^{-4}	3.23×10^{-2}	1.64
BCG	1	162	5.79×10^{-4}	3.22×10^{-2}	18.2
DADI	10	266	6.27×10^{-4}	1.26×10^{-2}	10.4
DADI′	10	55	5.89×10^{-4}	1.38×10^{-2}	2.28
BCG	10	420	1.98×10^{-4}	1.15×10^{-2}	46.0
DADI	100	20	3.76×10^{-5}	8.22×10^{-3}	0.756
DADI′	100	38	2.90×10^{-4}	3.56×10^{-3}	1.46
BCG	100	665	5.38×10^{-4}	3.58×10^{-3}	72.3
DADI	1,000	7	1.07×10^{-4}	2.10×10^{-3}	0.245
DADI′	1,000	18	3.26×10^{-4}	9.52×10^{-4}	0.679
BCG	1,000	754	9.94×10^{-4}	2.37×10^{-3}	81.9
DADI	2,000	7	6.39×10^{-5}	9.17×10^{-4}	0.246
DADI′	2,000	16	3.00×10^{-4}	5.40×10^{-4}	0.601
BCG	2,000	758	8.60×10^{-4}	1.86×10^{-3}	82.2
DADI	10,000	8	1.86×10^{-5}	1.99×10^{-4}	0.288
DADI′	10,000	12	3.44×10^{-4}	3.56×10^{-4}	0.438
BCG	10,000	875	8.44×10^{-4}	2.04×10^{-3}	94.8

between the "correct" answer and the rederived answer, **dif** $= |\mathbf{E}_{0\,\mathrm{sol}} - \mathbf{E}_{\mathrm{sol}}|$, and the CPU time in seconds required by each solution. We have used a modest residual criterion $E_{tst} = 1 \times 10^{-3}$, consistent with the expected Darwin model application and our finite-difference truncation error.

For the current test case, we find DADI, as embodied in Eqs. (34), to be significantly faster than BCG. In addition to the speed, the storage required by DADI is only 7×3 (one for each unknown) full 2D arrays, counting the unknown itself. The algorithm we used for BCG requires at least 72×3 full 2D arrays—though, as previously noted, at least 24 of these arrays appear not to be needed. Storage requirements aside, however, it is clear that even the conservative DADI provides a significant CPU advantage over BCG in this application.

The results obtained with DADI′, a variant of DADI with all the μ terms on the left as in the three-equation analog of Eqs. (28), are also given. Despite the caveats concerning the possible oscillation of the residual using DADI′ discussed in the previous section for this variant, DADI′ works faster for small μ_0 by roughly a factor of two. Consistent with our previous discussions, we still use this variant with caution and note that the original method DADI, that cannot exhibit this residual oscillation, is much superior for large μ_0—an advantage especially apparent for smaller error criterion. To restate, DADI *always works.* DADI is far superior for large μ_0 and only a factor of two slower for small μ_0, see Table IV. The factor of two advantage that DADI′ enjoys for small μ_0 suggests that the expected residual oscillation occurs at residual values smaller than our E_{tst} criterion. Further, we may be able to take advantage of this situation with an adaptive implicit–explicit mix for the μ term that depends on the local μ value.

Finally, we remark that for $\mu_0 = 10.0$ and 100.0, a regime in which μ and ∇^2 are roughly comparable for our mesh, the choice of DADI schemes may vary with E_{tst}, see Table V. Our truncation error is also evident in that the maximum

TABLE IV

Method	μ_0	Iter	Res	Dif	CPU
$E_{tst} = 10^{-5}$					
DADI	1	151	8.42×10^{-6}	2.77×10^{-2}	5.93
DADI′	1	90	8.85×10^{-6}	2.77×10^{-2}	3.78
DADI	10	1135	6.67×10^{-6}	1.65×10^{-2}	44.6
DADI′	10	383	6.51×10^{-6}	1.65×10^{-2}	17.6
$E_{tst} = 10^{-4}$					
DADI	1000	97	3.90×10^{-5}	3.84×10^{-3}	3.75
DADI′	1000	550	8.57×10^{-5}	7.91×10^{-4}	21.6
DADI	2000	10	3.19×10^{-5}	2.34×10^{-3}	0.37
DADI′	2000	357	4.29×10^{-5}	3.94×10^{-4}	14.4

TABLE V

Method	E_{tst}	Iter	Res	Dif	CPU
$\mu_0 = 10$					
DADI	10^{-3}	266	6.27×10^{-4}	1.26×10^{-2}	10.4
DADI′	10^{-3}	55	5.89×10^{-4}	1.38×10^{-2}	2.28
BCG	10^{-3}	420	1.98×10^{-4}	1.15×10^{-2}	46.0
DADI	10^{-4}	621	7.24×10^{-5}	1.17×10^{-2}	24.3
DADI′	10^{-4}	237	5.05×10^{-5}	1.18×10^{-2}	10.6
BCG	10^{-4}	427	7.87×10^{-5}	1.15×10^{-2}	46.5
$\mu_0 = 100$					
DADI	10^{-3}	20	3.76×10^{-5}	8.22×10^{-3}	0.756
DADI′	10^{-3}	38	2.90×10^{-4}	3.56×10^{-3}	1.46
BCG	10^{-3}	665	5.38×10^{-4}	3.58×10^{-3}	72.3
DADI	10^{-4}	1044	4.54×10^{-5}	2.87×10^{-3}	41.1
DADI′	10^{-4}	307	7.35×10^{-5}	3.12×10^{-3}	13.3
BCG	10^{-4}	685	9.47×10^{-5}	3.58×10^{-3}	74.5

discrepancy $|\mathbf{E}_{0\,\mathrm{sol}} - \mathbf{E}_{\mathrm{sol}}|$ is nearly the same for $E_{tst} = 10^{-3}$ and 10^{-4}. This behavior, consistently found for $\mu_0 \leqslant 10.$, suggests that more stringent error criteria provide only better solutions to the finite-difference equations; the physical solution does not change.

Further, we note that the CPU time required to reach the smaller E_{tst} increases only for the two DADI versions—suggesting that the major portion of the residual error is corrected early in the iteration. On the other hand, the CPU time for BCG remains almost the same—suggesting that the part of the error vector responsible for the large residual is not corrected by BCG until near the end of the iteration. If for some reason the iteration is stopped before the convergence criteria is met, it appears that the DADI answer will be more nearly correct.

We stress again that we have made no effort to find an optimal preconditioner for either DADI or BCG. It is entirely possible, perhaps probable, that BCG can be made to converge more rapidly with a preconditioner motivated by physics intuition. It is also likely that DADI would respond favorable to such attention.

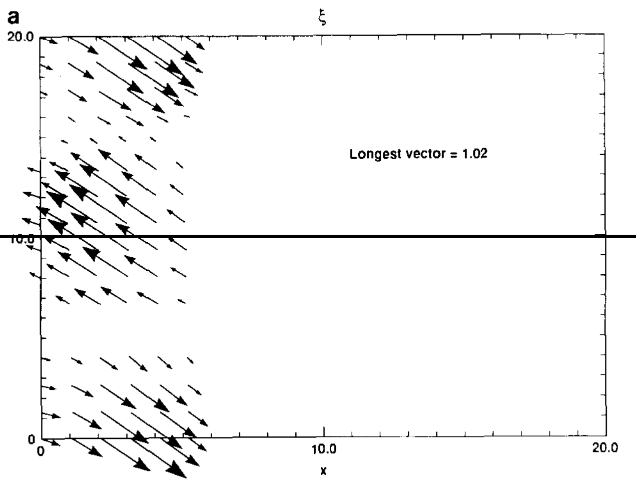
We also find a hint that DADI′ can be improved by roughly another factor of two for the smaller values of μ by the more symmetric form of splitting the first-order terms as given by Eqs. (36). For example, the symmetrized version of DADI′ converges in 181 iterations taking 8.27 s for $E_{tst} = 10^{-4}$ and $\mu_0 = 100$. The symmetrized version of DADI is also faster—taking 583 iterations in 26.6 s for the same case—though both symmetric and nonsymmetric forms of DADI are significantly slower than either form of DADI′ for these parameters. These symmetrized forms require another three full 2D storage arrays for the additional "time" levels and are slightly more complex to code.

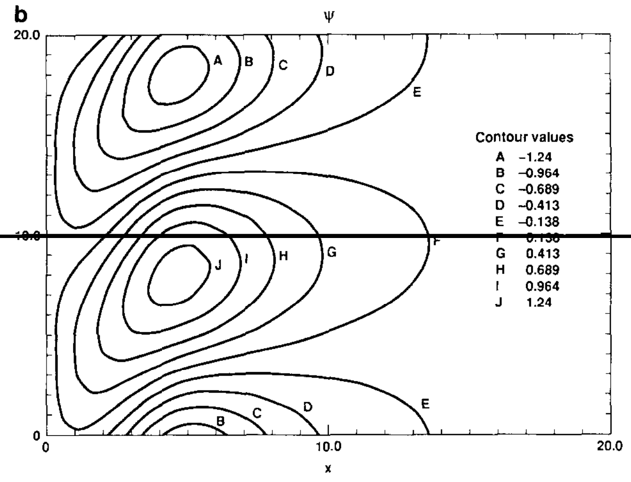
The symmetrized versions do not work better in *all* cases, however, and previous experience suggests that the symmetric form may be less tolerant in strongly nonlinear problems. Now that we have demonstrated that a low cost solution of the SDF equations can be obtained with this method, we choose to leave the final optimization choices for specific applications to the user, where other issues such as geometric dependence and robustness may influence the choices.

IV.D. Boundary Conditions

Boundary conditions were chosen such that the original $\mathbf{E}_{0\,\mathrm{sol}}$ could be recovered. The boundary conditions needed to accomplish this are not unique. For the typical $\mu_0 = 1.0$ case shown in Fig. 2 we have taken $\psi = 0$ at $x_{\min}$ and $x_{\max}$ and have used conditions given by Eq. (17) for ξ so that the desired $\mathbf{E}_{0\,\mathrm{sol}}$ is regenerated at the boundary; y-boundaries are periodic. Shown in Figs. 2a and b are a vector plot of ξ and a contour plot of ψ. Shown in Fig. 2c is a vector plot of the difference between the original $\mathbf{E}_{0\,\mathrm{sol}}$ and the new one that has just been computed by DADI. The number given on the figure is the magnitude of the largest difference. This value is consistent with the truncation error of our 5-point ∇^2 operators.

The choice of $\psi = 0$ guarantees that $\nabla\psi$ will be zero for the y component of ξ on the boundary. Another equally acceptable choice that generates the same physical solution is the Neumann condition $\nabla\psi = 0$ in the x-direction. In this case $\nabla\psi \neq 0$ in the y-direction and results in significantly different representations for both ξ and ψ. Shown in Fig. 3 are





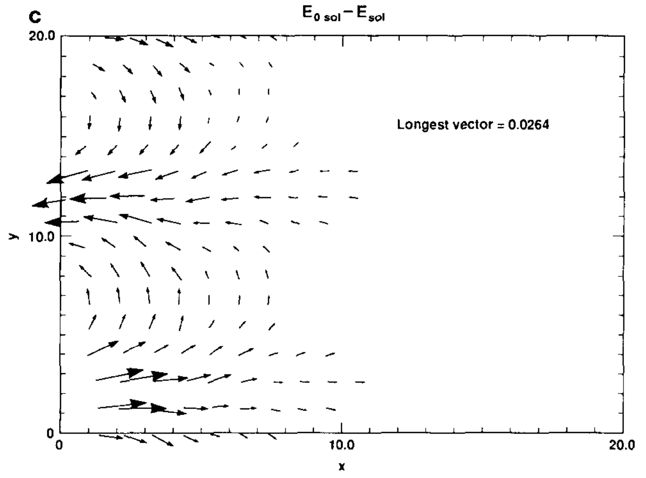


FIG. 2. In (a) we show a vector plot of ξ and (b) the corresponding scalar ψ that results from our solution with $\mu_0 = 1.0$. This is the solution for the case with Dirichlet boundary conditions at $x_{\min}$ and $x_{\max}$ and periodic boundary conditions in the y direction. In (c) we show the vector difference between the derived $\mathbf{E}_{\mathrm{sol}}$, using (a) and (b) from this figure, and the original $\mathbf{E}_{0\,\mathrm{sol}}$ from Fig. 1c. Note that the magnitude of the largest difference, shown in the inset, is two orders of magnitude smaller than the magnitude of the solution and is consistent with the truncation error of the finite difference approximation.

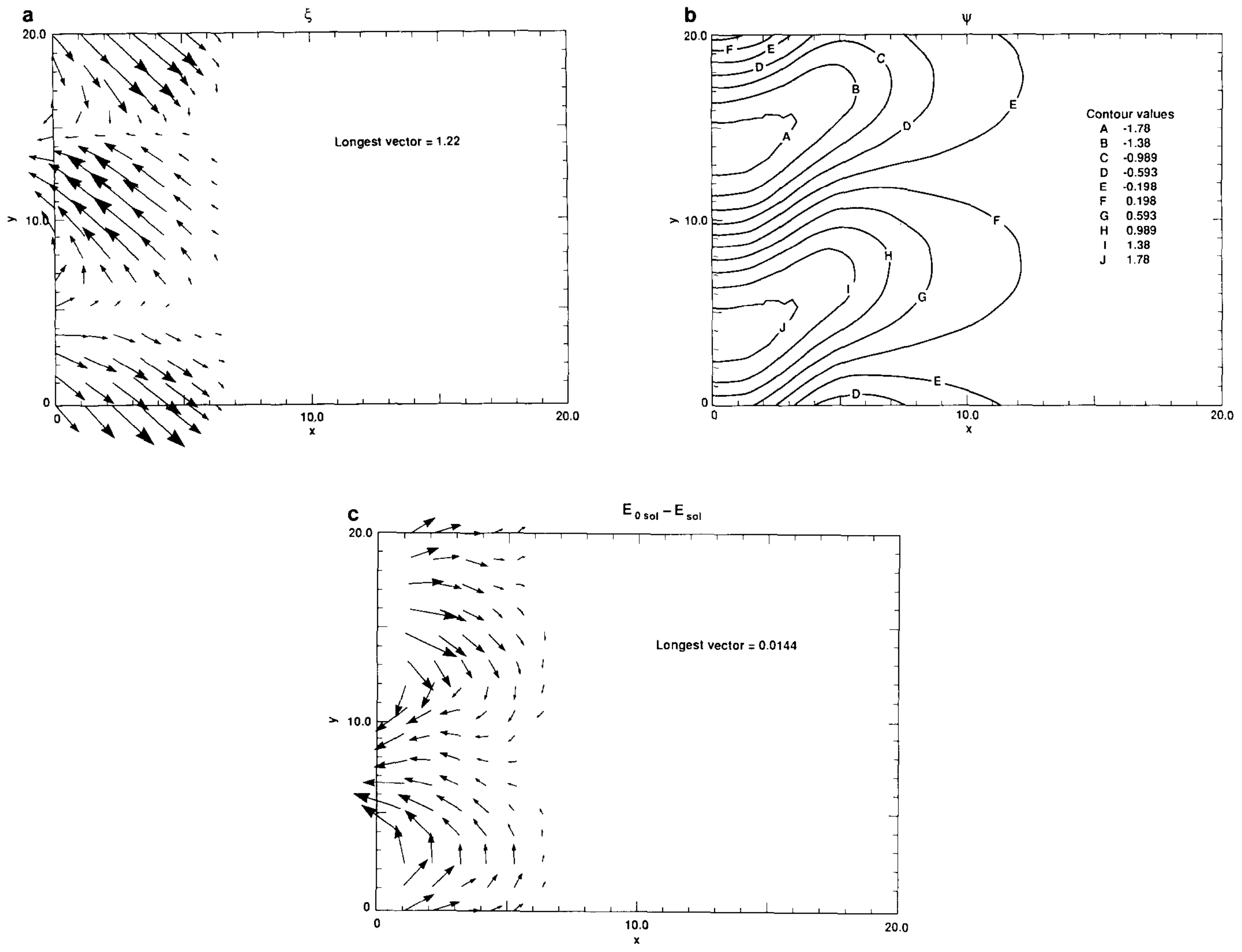


FIG. 3. Again we show a vector plot of (a) ξ and the corresponding scalar (b) ψ that result with $\mu_0 = 1.0$. In this case the solution $\mathbf{E}_{sol}$ is the same as in Fig. 2, to within truncation errors, but ξ and ψ look much different because of the selection of Neumann zero boundary conditions at x_{min}. We have maintained Dirichlet conditions at x_{max} and periodic boundary conditions in the y directions. (c) shows the vector difference between this solution $\mathbf{E}_{sol}$, using (a) and (b) from this figure, and the original $\mathbf{E}_{0\,sol}$ in Fig. 1c.

the plots corresponding to those in Fig. 2 that have this Neumann condition applied only to the x_{min} boundary. Comparing the solutions for ξ and ψ with the Dirichlet conditions in Fig. 2 with these solutions reveals quite different spatial dependence but, as evident by the small maximum vector difference, the original solution is recovered.

V. SUMMARY

While more work remains to be done in the area of finding optimal splitting choices for these and other equations as functions of their parameters, we feel that our results to date display exciting possibilities for dynamic ADI solution of strongly coupled equations. We believe that we may now achieve the advantages inherent in the SDF model without squandering the CPU gains on solution methods not well suited to the SDF model.

ACKNOWLEDGMENTS

We thank Dr. D. V. Anderson for numerous discussions and help with the ILUBCG algorithm. We acknowledge very useful discussions with Dr. R. Galinas in the development of this generalization of DADI. We also thank Dr. Jack Byers for a critical reading of the manuscript. Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

REFERENCES

1. See, for example, D. W. Hewett, *J. Comput. Phys.* **38**, 378 (1980); E. J. Horowitz, D. E. Shumaker, and D. V. Anderson, *J. Comput. Phys.* **84**, 279 (1989); D. S. Harned, *J. Comput. Phys.* **47**, 452 (1982).

2. D. W. Hewett and A. B. Langdon, *J. Comput. Phys.* **72**, 121 (1987).
3. D. V. Anderson, A. E. Koniges, and D. E. Shumaker, *Comput. Phys. Commun.* **51**, 391 (1988).
4. D. W. Hewett and C. W. Nielson, *J. Comput. Phys.* **29**, 219 (1978).
5. D. W. Hewett and J. K. Boyd, *J. Comput. Phys.* **70**, 166 (1987).
6. C. W. Nielson and H. R. Lewis, in *Methods Comput. Phys.*, Vol. 16, p. 367 edited by B. Alder, S. Fernbach, M. Rotenberg, and J. Killeen (Academic Press, New York, 1976).
7. J. Busnardo-Neto, P. L. Pritchett, A. T. Lin, and J. M. Dawson, *J. Comput. Phys.* **23**, 300 (1977).
8. J. K. Boyd, G. J. Caporaso, and A. G. Cole, *IEEE Trans. Nucl. Sci.* **Ns-32** (5), 2602 (1985).
9. H. Weitzner and W. Lawson, *Phys. Fluids B1* **10**, 1953 (1989).
10. Z. Mikic and E. C. Morse, *J. Comput. Phys.* **61**, 154 (1985).
11. D. W. Peaceman and H. H. Rachford, *J. Soc. Indus. Appl. Math.* **3**, 28 (1955).
12. S. Doss and K. Miller, *SIAM J. Numer. Anal.* **16**, 837 (1979).
13. D. W. Hewett and R. L. Spencer, *Phys. Fluids* **26**, 1299 (1983).