Journal of Computational Physics **178,** 15–36 (2002)
doi:10.1006/jcph.2002.7015, available online at http://www.idealibrary.com on IDEAL®

An Implicit, Nonlinear Reduced Resistive MHD Solver

L. Chacón,*,[1] D. A. Knoll,† and J. M. Finn*

*MS K717 and †MS B216, PO Box 1663, Los Alamos National Laboratory, Los Alamos, New Mexico 87545
E-mail: chacon@lanl.gov, nol@lanl.gov, and finn@lanl.gov

Received August 29, 2001; revised January 25, 2002

Implicit time differencing of the resistive magnetohydrodynamic (MHD) equations can step over the limiting time scales—such as Alfvén time scales—to resolve the dynamic time scales of interest. However, nonlinearities present in these equations make an implicit implementation cumbersome. Here, viable paths for an implicit, nonlinear time integration of the MHD equations are explored using a 2D reduced viscoresistive MHD model. The implicit time integration is performed using the Newton–Raphson iterative algorithm, employing Krylov iterative techniques for the required algebraic matrix inversions, implemented Jacobian-free (i.e., without ever forming and storing the Jacobian matrix). Convergence in Krylov techniques is accelerated by preconditioning the initial problem. A "physics-based" preconditioner, based on a semi-implicit approximation to the original set of partial differential equations, is employed. The preconditioner employs low-complexity multigrid techniques to invert approximately the resulting elliptic algebraic systems. The resulting 2D reduced resistive MHD implicit algorithm is shown to be successful in dealing with large time steps (on the order of the dynamical time scale of the problem) and fine grids. The algorithm is second-order accurate in time and scalable under grid refinement. Comparison of the implicit CPU time with an explicit integration method demonstrates CPU savings even for moderate (64×64) grids, and close to an order of magnitude in fine grids (256×256). © 2002 Elsevier Science (USA)

Key Words: Jacobian-free; Newton–Krylov; multigrid; physics-based preconditioning; implicit differencing; resistive MHD.

1. INTRODUCTION

The resistive magnetohydrodynamic (MHD) equations present a formidable challenge for efficient implicit differencing due to the disparity of time scales, the nonlinear couplings

[1] Corresponding author. Fax: (505) 665-7150.

0021-9991/02 $35.00

present in the equations, and the fine grids needed to resolve current sheets. Partially implicit time-differencing schemes have been employed in multidimensional MHD calculations, in which the *discretized* set of partial differential equations (PDEs) is modified by adding implicitly and subtracting explicitly some suitable operator (called semi-implicit operator), and only part of the equations is integrated implicitly at a given time step. This allows time steps larger than the explicit Courant–Friedrichs–Lewy (CFL) stability limit Δt_{CFL} while requiring less work than a direct implicit integration for a given time step. Such partially implicit techniques include alternating direction implicit (ADI) methods [1, 2] and so-called Harned–Kerner (H–K) semi-implicit techniques [3–6]. In the former, the semi-implicit operator decouples the integration in the different directions of the problem [7]. Consequently, it is possible to restrict the implicit integration to one direction while the others remain explicit, alternating directions every time step. However, the semi-implicit operator in ADI techniques is of purely algebraic convenience and has no physical significance. In addition, the implementation of ADI methods to solve the MHD equations is cumbersome due to the nonlinear couplings and the presence of mixed derivatives, requiring further simplifications to render the problem tractable. As a result, ADI methods in MHD problems only allow time steps $\sim$10–20 times larger than the explicit numerical stability limit for stability [1, 2].

In H–K semi-implicit methods, the semi-implicit operator slows down the propagation of the fastest waves in the problem, thus achieving unconditional numerical stability for a sufficiently large semi-implicit coefficient. However, stability does not guarantee accuracy. The accuracy of the numerical solution obtained with MHD semi-implicit techniques is strongly dependent on the choice of the form of the semi-implicit operator [5, 6]. Although rigorous methods to derive accurate semi-implicit operators are available [8], again simplifications required to make the system tractable, as well as constraints on the magnitude of the semi-implicit coefficient to guarantee stability, may result in inaccurate semi-implicit operators. As a consequence, the time step in H–K semi-implicit methods is typically limited by *accuracy* considerations. In Ref. [5], a time-step limit of $\gamma \Delta t \leq 0.05$ was reported for the the $m = 1$, $n = -2$ tearing mode, with γ the growth rate. In general, time-consuming convergence studies are required to find the time-step accuracy limit.

There has been a recent attempt to implement a nonlinear, implicit difference scheme for the MHD equations [9]. This approach employs block Gauss–Seidel techniques to invert an approximate Jacobian matrix (and LU-decomposition techniques to invert the blocks) to obtain convergence, including the nonlinear terms in the equations. However, the simplifications in the Jacobian (required to render it manageable) limit the applicability of the method, because they introduce CFL restrictions for large and moderate viscosities and resistivities. Furthermore, the nonlinear residual is never used to check nonlinear convergence (the size of the nonlinear update δx is used for convergence), and hence nonlinear errors stemming from the approximate Jacobian remain unmonitored.

Modern matrix inversion methods such as multigrid-preconditioned Krylov subspace iterative methods are employed in Refs. [10, 11]. In Ref. [10], an implicit, nonlinear MHD solver is proposed based on a implicit operator split (IOS) method developed earlier for hydrodynamic applications [12]. The split algorithm is iterated upon in a Gauss–Seidel manner, so that some degree of nonlinear consistency is achieved (although no measure of nonlinear convergence is explicitly provided). As is recognized in [10], large numerical errors in transients are possible with this algorithm for large implicit time steps unless enough nonlinear iterations are taken; still, large implicit time steps can be employed in

the search of stationary solutions. This notwithstanding, it is reported [10] that the implicit code takes ∼40% more CPU time than explicit counterparts for certain test calculations.

The present document explores efficient paths for fully implicit, fully nonlinear, unsplit differencing of the resistive MHD equations. The objective is to avoid the sources of numerical error present in semi-implicit and IOS methods for large implicit time steps, and to demonstrate a computational tool that can accurately describe transients in the dynamic time scale of interest with implicit time steps comparable to such dynamical time scales [i.e., such that $\gamma \Delta t \sim O(1)$], even when such implicit time steps are much longer than the time scale of normal modes (waves) in the system. For this purpose, we focus on a 2D reduced resistive MHD model, supporting the shear Alfvén wave. Convergence in the nonlinear system is achieved using the Newton–Raphson iterative algorithm. Krylov iterative techniques [13], implemented Jacobian-free [14, 15] (i.e., without ever forming and storing the Jacobian matrix), are employed for the required algebraic matrix inversions, because of their efficiency and the possibility of accelerating convergence via preconditioning. Here, GMRES (generalized minimal residuals [16]) is employed due to the lack of symmetry in the algebraic system.

The efficiency of Krylov methods depends heavily on adequate preconditioning [13]. Here, a "physics-based" preconditioner [17, 18] is developed based on a semi-implicit formulation of the reduced MHD equations in vorticity stream-function form. Elliptic systems in the preconditioner are inverted using low-complexity multigrid methods (MG) [19–21].

The rest of the paper is organized as follows. Section 2 introduces the base model equations. Section 3 introduces the Krylov methods and the specifics of the Jacobian-free implementation. In Section 4, the "physics-based" preconditioner for this particular application is developed. Validation and efficiency results of the implicit algorithm are presented in Section 5, where the following is numerically demonstrated:

1. The algorithm performs nearly optimally under grid refinement (the number of Krylov iterations scales very weakly with N, the total number of mesh points).
2. The algorithm obtains a sublinear scaling of the number of Krylov iterations with the implicit time step, which results in a favorable CPU time scaling.
3. The algorithm features second-order-accurate time step convergence.
4. Implicit time steps on the order of the dynamical time scale are possible without sacrificing accuracy.

Finally, we conclude in Section 6.

2. 2D REDUCED MHD MODEL

In the 2D reduced MHD (RMHD) formalism, the magnetic field component in the ignorable direction B_z is much larger than the magnitude of the poloidal magnetic field $\mathbf{B}_p$. As a result, $B_z \approx$ constant, the poloidal velocity $\mathbf{v}$ is incompressible ($\nabla \cdot \mathbf{v} = 0$), and the general MHD formalism reduces to [22–24]

$$\nabla^2 \Phi = \omega, \tag{1}$$

$$\left(\partial_t + \mathbf{v} \cdot \nabla - \frac{\eta}{\mu_0} \nabla^2\right) \Psi + E_0 = 0, \tag{2}$$

$$\rho(\partial_t + \mathbf{v} \cdot \nabla - \nu \nabla^2)\omega + \dot{S}_\omega = \frac{1}{\mu_0} \mathbf{B} \cdot \nabla(\nabla^2 \Psi), \tag{3}$$

where Φ is the poloidal velocity stream function ($\mathbf{v} = \mathbf{z} \times \nabla\Phi$), ω is the vorticity in the poloidal plane ($\omega = \mathbf{z} \cdot \nabla \times \mathbf{v}$), Ψ is the poloidal flux function (which gives $\mathbf{B}_p = \mathbf{z} \times \nabla\Psi$), $\mathbf{B} = \mathbf{B}_p + B_z\mathbf{z}$ is the total magnetic field, and ρ is the density (which is taken as constant). Sources E_0 (the applied electric field in the z direction) and $\dot{S}_\omega$ have been included to balance the decay of the equilibrium due to transport terms. The transport parameters (the kinematic viscosity ν and the resistivity η) are assumed constant. We note that $\mathbf{B} \cdot \nabla = \mathbf{B}_p \cdot \nabla$ since $\partial_z = 0$, but we keep $\mathbf{B}$ for the sake of generality.

Equations (1)–(3) are normalized as follows: $\mathbf{B}$ is normalized to the poloidal magnetic field at the wall in equilibrium B_{pw}, ρ to ρ_0, lengths to the characteristic length in the y-direction L_y, and the time to the poloidal Alfvén time $\tau_A = L_y/v_A$, where $v_A = B_{pw}/\sqrt{\rho_0\mu_0}$ is the Alfvén speed. The normalized set of RMHD equations reads

$$\nabla^2\Phi = \omega, \tag{4}$$

$$\left(\partial_t + \mathbf{v} \cdot \nabla - \frac{1}{S_L}\nabla^2\right)\Psi + E_0 = 0, \tag{5}$$

$$\left(\partial_t + \mathbf{v} \cdot \nabla - \frac{1}{\mathrm{Re}}\nabla^2\right)\omega + \dot{S}_\omega = \mathbf{B} \cdot \nabla(\nabla^2\Psi), \tag{6}$$

where $S_L = \frac{\mu_0 L_y v_A}{\eta}$ is the Lundquist number and $\mathrm{Re} = \frac{L_y v_A}{\nu}$ is the Reynolds number. All magnitudes in the equations are dimensionless at this point.

Equations (4)–(6) are discretized using a theta difference scheme, as follows.

$$\nabla^2\Phi^{n+1} = \omega^{n+1}, \tag{7}$$

$$\frac{\Psi^{n+1} - \Psi^n}{\Delta t} + [\mathbf{v} \cdot \nabla\Psi]^{n+\theta} - \frac{\nabla^2\Psi^{n+\theta}}{S_L} = -E_0, \tag{8}$$

$$\frac{\omega^{n+1} - \omega^n}{\Delta t} + [\mathbf{v} \cdot \nabla\omega]^{n+\theta} - \frac{\nabla^2\omega^{n+\theta}}{\mathrm{Re}} = [\mathbf{B} \cdot \nabla(\nabla^2\Psi)]^{n+\theta} - \dot{S}_\omega, \tag{9}$$

where quantities at the $n + \theta$ time level are calculated as $\xi^{n+\theta} = \theta\xi^{n+1} + (1 - \theta)\xi^n$, with θ a shifting parameter [$0.5 \leq \theta < 1$ for the method to remain implicit; $\theta = 1$ is first-order backward Euler, $\theta = 0.5$ is second-order Crank–Nicolson (CN)]. Second-order accuracy in time is crucial for the ability to use large implicit time steps that follow the dynamical time scale accurately (see Section 5.2.1 for numerical evidence). While CN is second-order accurate in time, for sufficiently large implicit time steps it is known to "ring" (i.e., to propagate weakly damped short-wavelength harmonics) in stiff problems (such as diffusion operators in this context) [25, 26]. CN ringing can be avoided while preserving second-order accuracy by employing the so-called "Rannacher time stepping" (in which the second-order time evolution is preceded by at least two first-order backward Euler steps to damp short-wavelength harmonics on the grid), and/or by redefining the shifting parameter as $\theta \approx \frac{1}{2} + c\Delta t$, where c is related to the dynamical time scale of the problem [25, 26]. In this paper, Rannacher's time stepping is employed unless otherwise specified, and no CN ringing was observed.

Spatial operators are discretized using second-order centered finite differences, except for the advection terms, which are discretized using quadratic upstream interpolation (QUICK [27]). Notice that the first equation in the previous set of difference equations does not involve a time step and represents a purely elliptic constraint. This constraint must be

satisfied to the prescribed tolerance independently of the time step chosen, thus imposing additional strain on the solver.

3. JACOBIAN-FREE NEWTON–KRYLOV SOLVER

Once the RMHD equations are discretized in time and space, the next step is to find the new-time solution $\mathbf{x}^{n+1} = \{\mathbf{\Phi}^{n+1}, \mathbf{\Psi}^{n+1}, \omega^{n+1}\}^T$ from the current-time solution $\mathbf{x}^n$ by solving the nonlinear, coupled system of equations in Eqs. (7)–(9), symbolized by $\mathbf{G}(\mathbf{x}^{n+1}) = \mathbf{0}$ (where $\mathbf{G} = \{\mathbf{G}_\Phi, \mathbf{G}_\Psi, \mathbf{G}_\omega\}^T$). This is accomplished iteratively with the Newton–Raphson algorithm, which requires the solution of a series of algebraic systems of the form

$$J_k \delta \mathbf{x}_k = -\mathbf{G}(\mathbf{x}_k). \tag{10}$$

Here, $J_k = (\frac{\partial \mathbf{G}}{\partial \mathbf{x}})_k$ is the Jacobian matrix, $\mathbf{x}_k$ is the kth state vector, $\delta \mathbf{x}_k$ is the kth Newton update (from which the $(k+1)$th Newton state vector is obtained, $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta \mathbf{x}_k$), $\mathbf{G}(\mathbf{x}_k)$ is the vector of residuals, and k is the nonlinear iteration level. Nonlinear convergence is achieved when

$$\|\mathbf{G}(\mathbf{x}_k)\|_2 < \epsilon_{newton} \|\mathbf{G}(\mathbf{x}_0)\|_2, \tag{11}$$

where $\|\cdot\|_2$ is the ℓ_2-norm, ϵ_{newton} is the Newton convergence tolerance (set to 10^{-4} in this work), and $\mathbf{G}(\mathbf{x}_0)$ is the initial residual. Upon convergence, the solution at the new time step is found as $\mathbf{x}^{n+1} = \mathbf{x}_{k+1}$.

Each of these iterative steps requires inverting the Jacobian system in Eq. (10). This is performed here using Krylov methods [13], because of the following:

1. As stand-alone solvers, they are already competitive against direct or standard iterative techniques and have the capability of increased efficiency via adequate preconditioning.

2. All that these methods require to proceed is the product of the system matrix times a Krylov vector $\mathbf{v}$, which is provided by the iterative algorithm. In Newton's method, the Jacobian-vector product can be calculated using the directional (Gateaux) derivative, approximated here as

$$J_k \mathbf{v} \approx \frac{\mathbf{G}(\mathbf{x}_k + \epsilon \mathbf{v}) - \mathbf{G}(\mathbf{x}_k)}{\epsilon}, \tag{12}$$

where ϵ is small but finite (discussed later in this section). Thus, the evaluation of the Jacobian-vector product only requires the function evaluation $\mathbf{G}(\mathbf{x}_k + \epsilon \mathbf{v})$. There is no need to form or store the Jacobian matrix, and hence the name Jacobian-free.

3. An additional advantage of not forming the Jacobian matrix is that cumbersome difference schemes (such as conservative and/or high-order difference schemes) are of straightforward implementation and maintenance.

Among the various Krylov methods available, GMRES is selected because it guarantees convergence with nonsymmetric, nonpositive definite systems [16] (the case here because of flow and wave propagation), and because it provides normalized Krylov vectors $|\mathbf{v}| = 1$, thus bounding the error introduced in the difference approximation of Eq. (12) (whose leading error term is proportional to $\epsilon |\mathbf{v}|^2$) [28]. However, GMRES can be memory intensive (storage increases linearly with the number of GMRES iterations per Jacobian solve) and

expensive (computational complexity of GMRES increases with the square of the number of GMRES iterations per Jacobian solve). Restarted GMRES can in principle deal with these limitations; however, it lacks a theory of convergence, and stalling is frequently observed in real applications [29]. Here, we focus on minimizing the number of GMRES iterations per Jacobian solve for efficiency, by (i) using inexact Newton techniques [30], and (ii) improving the condition number of the Jacobian matrix by preconditioning the problem.

The inexact Newton method adjusts the GMRES convergence tolerance at every Newton iteration according to the size of the Newton residual, as follows.

$$\|J_k\delta\mathbf{x}_k + \mathbf{G}(\mathbf{x}_k)\|_2 < \zeta_k\|\mathbf{G}(\mathbf{x}_k)\|_2, \tag{13}$$

where ζ_k is the inexact Newton parameter or forcing term. Thus, the convergence tolerance of GMRES is loose when the state vector $\mathbf{x}_k$ is far from the nonlinear solution, but becomes increasingly tighter as $\mathbf{x}_k$ approaches the exact solution, and GMRES works hard only when the state vector is close to the exact solution. Although the superlinear convergence rate of the inexact Newton method is possible if the sequence of ζ_k is chosen properly [31], the choice $\zeta_k = \zeta$ constant, while providing only a linear Newton convergence rate, is found to be satisfactory efficiency-wise in various Newton–GMRES applications [31]. Although the optimal value of ζ is application dependent, $\zeta \sim 0.1$–0.01 is common in the literature [17, 31, 32], and values of $\zeta \sim 5 \times 10^{-2}$ work well for this application.

Preconditioning consists of operating on the system matrix J_k with an operator P_k (preconditioner) such that $J_k P_k$ is well-conditioned. The Jacobian-free implementation of the right preconditioner operator is straightforward when considering the equivalent system:

$$(J_k P_k)\left(P_k^{-1}\delta\mathbf{x}_k\right) = -\mathbf{G}(\mathbf{x}_k). \tag{14}$$

Thus, GMRES will solve

$$(J_k P_k)\mathbf{z} = -\mathbf{G}(\mathbf{x}_k), \tag{15}$$

and the Newton update $\delta\mathbf{x}_k$ is found upon obtaining $\mathbf{z}$ by finding $\delta\mathbf{x}_k = P_k\mathbf{z}$. Notice that the system in Eq. (14) is equivalent to the original system for any nonsingular operator P_k. Thus, the choice of P_k does not affect the accuracy of the final solution but crucially determines the efficiency of the algorithm.

To solve Eq. (15) using GMRES, it is required to compute the Jacobian-vector product $(J_k P_k)\mathbf{v}_j$ (where $\mathbf{v}_j$ is the Krylov vector of the jth iteration) to proceed. This is implemented in two steps:

1. Compute $\mathbf{y} = P_k\mathbf{v}_j$. This is the so-called preconditioning step. Often, P_k is not an exact inverse of any particular matrix but an approximate inverse—obtained, for instance, using operator splitting and/or low-complexity MG methods—of the exact Jacobian, or even an approximate inverse of an approximation of the Jacobian. The specifics of the formation of the preconditioner operator P_k for this application are discussed in Section 4.

2. Compute $J_k\mathbf{y}$ using the Jacobian-free approximation $J_k\mathbf{y} \approx \frac{\mathbf{G}(\mathbf{x}_k + \epsilon\mathbf{y}) - \mathbf{G}(\mathbf{x}_k)}{\epsilon}$, where ϵ is small but finite. Newton convergence is insensitive to ϵ within a window two to three orders in magnitude; ϵ is calculated here as

$$\epsilon = 10^{-5}\left(1 + \frac{\|\mathbf{x}_k\|_2}{\|\mathbf{y}\|_2}\right).$$

The first step determines the efficiency of the algorithm (and leaves room for exploration,

since P_k is in principle an arbitrary nonsingular operator), while the second step determines the accuracy of the solution [according to the discretization of the nonlinear system $\mathbf{G}(\mathbf{x}^{n+1}) = \mathbf{0}$].

To maximize efficiency, the preconditioning operator P_k should approximate the inverse of the Jacobian J_k while being relatively inexpensive. There are generally two choices as to how to approach the preconditioning problem:

1. *Algebraic methods*: These approximately invert a close representation of the Jacobian J_k, obtained analytically or numerically, using inexpensive algebraic techniques (such as stationary iterative techniques, incomplete LU decomposition, multigrid techniques, etc.). These techniques are "problem independent" (can be employed in a variety of different problems), but by the same token they cannot exploit specific physics knowledge of the problem at hand. In addition, algebraic preconditioners typically require forming and storing the Jacobian matrix.

2. *PDE-based or physics-based methods* [17, 18]: These integrate an approximate set of PDEs, obtained, for instance, by simple Picard linearization, by semi-implicit techniques, and/or by implicit operator splitting (IOS). They do not require forming and storing the complete Jacobian and, hence, take better advantage of the Jacobian-free implementation than do algebraic methods. In addition, they can be optimized for the problem at hand. Although the general concept of physics-based preconditioning can be applied to a large variety of problems, the details of the implementation are typically "problem specific."

The latter is explored here, with semi-implicit methods forming the basis of the preconditioner strategy. The next section describes in more detail the nature of the approximations employed here to construct the physics-based preconditioner. Recall that these approximations have no bearing on the accuracy of the converged solution, only on the convergence rate of GMRES.

4. PHYSICS-BASED PRECONDITIONER

Implicit differencing ensures absolutely stable numerical descriptions, for any time step and level of mesh refinement, by introducing dispersion in waves and by treating elliptic operators (such as diffusion) nonlocally. However, some of the mechanisms that are sources of numerical instabilities in explicit methods continue to manifest themselves in implicit schemes in the form of ill-conditioned algebraic systems, which iterative techniques have difficulty handling.

There are two sources of ill conditioning in the system of MHD equations: elliptic operators and hyperbolic couplings. The former manifests itself in a power scaling N^α (with $\alpha > 1$) of the computational complexity of iterative solver techniques. (Direct solvers are in principle suitable for dealing with poorly conditioned matrices; however, they do not scale adequately for sparse banded systems in 2D and 3D [7, 21].) Elliptic stiffness is dealt with here with multigrid preconditioning (MG), which employs low-complexity multilevel solvers [20] to invert the elliptic operators approximately. The multilevel aspect of MG (which employs a "divide-and-conquer" approach by which the different scales of the global solution are decoupled in multiple grids of varying mesh refinement) results, as a solver, in an optimal $O(N)$ scaling of the computational complexity [33], and, as a preconditioner, in a number of Krylov iterations virtually independent of the problem size (see [19, 20] and results in Section 5.2.2).

Ill conditioning from hyperbolic couplings manifests itself in a loss of diagonal dominance due to short-wavelength harmonics when the implicit time step is larger than the explicit wave CFL limit (short-wavelength harmonics are responsible for numerical instability in explicit methods). This can clearly be seen in this application by an order-of-magnitude analysis of the blocks in the Jacobian matrix, which formally reads (see the next section for the derivation of these terms)

$$J = \begin{pmatrix} -\nabla^2 & 0 & I \\ -\mathbf{B}_0 \cdot \nabla & \frac{1}{\Delta t} + \mathbf{v}_0 \cdot \nabla - \frac{\nabla^2}{S_L} & 0 \\ -\nabla^2(\mathbf{v}_0) \cdot \nabla & [-\mathbf{B}_0 \cdot \nabla(\nabla^2) + \nabla^2(\mathbf{B}_0) \cdot \nabla] & \left[\frac{1}{\Delta t} + \mathbf{v}_0 \cdot \nabla - \frac{\nabla^2}{\mathrm{Re}}\right] \end{pmatrix}. \tag{16}$$

In normalized units, $\mathbf{B}_0 \sim \mathbf{v}_A$, the Alfvén velocity. In the regime in which the implicit time step is much larger than the wave CFL limit ($\frac{1}{\Delta t} \ll \mathbf{k} \cdot \mathbf{v}_A$), in which flows are typically much slower than the characteristic Alfvén speed ($v_0 \ll v_A$), and in which transport coefficients are small (S_L, Re $\gg 1$), the order of magnitude of the Jacobian blocks is

$$\begin{pmatrix} k^2 & 0 & 1 \\ k v_A & \frac{1}{\Delta t} + k v_0 & 0 \\ k\omega_0' & k^3 v_A & \frac{1}{\Delta t} + k v_0 \end{pmatrix},$$

which is clearly non-diagonally dominant when the wavenumber $k \gg 1$, and hence handled poorly by iterative techniques.

It is possible, however, to reformulate the physical equations so that the resulting algebraic systems are better conditioned. The basic idea is to produce a well-conditioned (diagonally dominant) second-order parabolic operator from an ill-conditioned, first-order hyperbolic system of equations by implicitly discretizing the hyperbolic equations and suitably combining them by direct substitution. This is, for instance, the basis of the ICE (implicit continuous-fluid Eulerian) method for the compressible Navier–Stokes equations [34]. The procedure can be understood easily with a first-order hyperbolic linear system:

$$\partial_t u = \partial_x v,$$
$$\partial_t v = \partial_x u.$$

Differencing implicitly in time (with backward Euler for simplicity), we have

$$u^{n+1} = u^n + \Delta t \partial_x v^{n+1},$$
$$v^{n+1} = v^n + \Delta t \partial_x u^{n+1}.$$

It is now possible to substitute the second equation into the first to obtain the parabolic equation

$$(I - \Delta t^2 \partial_{xx}) u^{n+1} = u^n + \Delta t \partial_x v^n,$$

which is equivalent to the set of two discretized equations, but much better conditioned because the parabolic operator is diagonally dominant.

The goal here is to find a semi-implicit formulation for the RMHD equations that removes the stiffness associated with the Alfvén wave in the preconditioning stage. Although the

general guiding principle is the same as in the simple example above, specifics of the RMHD model (such as the presence of advection and diffusion) make this task difficult. Furthermore, due to the ω–Φ coupling, the vorticity equation contains $\partial_t \nabla^2 \Phi$, which in principle precludes the direct substitution step. The next section describes how to get around these issues.

4.1. Approximate Formulation of the RMHD System

Krylov techniques are employed here to invert the Jacobian system in each Newton step. Hence, the construction of the physics-based preconditioner necessarily starts from the linearized system of equations. For the system in Eqs. (7)–(9), the linearized equations read

$$L_{S_L} \delta\Psi = -\theta \delta\mathbf{v} \cdot \nabla\Psi_0 - G_\Psi, \tag{17}$$

$$L_{\mathrm{Re}} \delta\omega = \theta[-\delta\mathbf{v} \cdot \nabla\omega_0 + \mathbf{B}_0 \cdot \nabla(\nabla^2 \delta\Psi) + \delta\mathbf{B} \cdot \nabla(\nabla^2 \Psi_0)] - G_\omega, \tag{18}$$

$$\delta\omega = \nabla^2 \delta\Phi - G_\Phi. \tag{19}$$

Here, δ quantities represent perturbations, the subscript "0" represents the solution at the previous Newton step, $\{G_\Phi, G_\Psi, G_\omega\}$ are the nonlinear residuals, and

$$L_\chi = \frac{1}{\Delta t} + \theta[\mathbf{v}_0 \cdot \nabla - \chi^{-1} \nabla^2].$$

Again, the goal is to reformulate the hyperbolic couplings in these equations as a single parabolic operator. The procedure is as follows:

1. The first step is to eliminate $\delta\omega$.
2. The resulting system is fourth order in $\delta\Phi$ due to the vorticity-stream-function elliptic coupling. The next step is to approximate this fourth-order operator by the composition of two second-order operators, of easier inversion.
3. The final step is to approximate the inversion of the coupled, second-order two-equation system in $\delta\Phi - \delta\Psi$ by the inversion of a single parabolic operator embedded in a Jacobi iteration scheme.

This section deals with the first two steps. The next section expands on the third step.

We start by noting that $\delta\mathbf{v} \cdot \nabla\Psi_0 = -\mathbf{B}_0 \cdot \nabla\delta\Phi$, $\delta\mathbf{v} \cdot \nabla\omega_0 = -(\mathbf{z} \times \nabla\omega_0) \cdot \nabla\delta\Phi$, and $\delta\mathbf{B} \cdot \nabla(\nabla^2\Psi_0) = -[\mathbf{z} \times \nabla(\nabla^2\Psi_0)] \cdot \nabla\delta\Psi$. Substituting Eq. (19) into Eq. (18) and regrouping yields

$$L_{S_L} \delta\Psi = \theta \mathbf{B}_0 \cdot \nabla\delta\Phi - G_\Psi, \tag{20}$$

$$[L_{\mathrm{Re}} \nabla^2 - \theta(\mathbf{z} \times \nabla\omega_0) \cdot \nabla]\delta\Phi = \theta[(\mathbf{B}_0 \cdot \nabla)\nabla^2 - \mathbf{z} \times \nabla(\nabla^2\Psi_0)] \cdot \nabla]\delta\Psi - G_\omega + L_{\mathrm{Re}}(G_\Phi). \tag{21}$$

At this point, we note that $\mathbf{z} \times \nabla(\nabla^2\Psi_0) = \nabla^2 \mathbf{B}_0$ and $\mathbf{z} \times \nabla\omega_0 = \mathbf{z} \times \nabla(\nabla^2\Phi_0) = \nabla^2 \mathbf{v}_0$, because $\mathbf{z} \times \nabla$ commutes with ∇^2. Hence,

$$L_{\mathrm{Re}} \nabla^2 - \theta(\mathbf{z} \times \nabla\omega_0) \cdot \nabla = \frac{\nabla^2}{\Delta t} - \theta \frac{\nabla^4}{\mathrm{Re}} + \theta[(\mathbf{v}_0 \cdot \nabla)\nabla^2 - (\nabla^2 \mathbf{v}_0) \cdot \nabla].$$

The operator in square brackets acting on $\delta\Phi$ can be expressed in Einstein notation as

$$[(\mathbf{v}_0\cdot\nabla)\nabla^2-(\nabla^2\mathbf{v}_0)\cdot\nabla]\delta\Phi=\partial_i\partial_j[v_{0j}\partial_i\delta\Phi-\delta\Phi\partial_i v_{0j}]\sim\nabla^2(\mathbf{v}_0\cdot\nabla)\delta\Phi, \tag{22}$$

where we have used $\nabla\cdot\mathbf{v}_0=0$. In the last step, we have neglected terms with $i\neq j$. Although this approximation cannot be justified rigorously in general, it is exact for uniform $\mathbf{v}_0$. The approximation will not affect the accuracy of the converged solution (because it is done only in the preconditioner), and the effectiveness of the resulting algorithm will be its ultimate justification. Hence, we conclude that $L_{\mathrm{Re}}\nabla^2-\theta(\mathbf{z}\times\nabla\omega_0)\cdot\nabla\sim\nabla^2 L_{\mathrm{Re}}$. Identically, we find $(\mathbf{B}_0\cdot\nabla)\nabla^2-\mathbf{z}\times\nabla(\nabla^2\Psi_0)\sim\nabla^2(\mathbf{B}_0\cdot\nabla)$, and hence the linearized system in Eqs. (20) and (21) can be approximated, for preconditioning purposes, by

$$L_{S_L}\delta\Psi=\theta\mathbf{B}_0\cdot\nabla\delta\Phi-G_\Psi, \tag{23}$$

$$L_{\mathrm{Re}}\delta\Phi=\theta\mathbf{B}_0\cdot\nabla\delta\Psi+\nabla^{-2}[-G_\omega-L_{\mathrm{Re}}(-G_\Phi)]. \tag{24}$$

The importance of the approximation in Eq. (22) is now obvious, because we have effectively reduced the order of the $\delta\Phi$-equation from fourth order to two second-order inversions, hence allowing the direct substitution step to form the semi-implicit operator. The vorticity perturbation $\delta\omega$ is trivially obtained from $\delta\Phi$ by using Eq. (19).

4.2. Formulation of the Semiimplicit Preconditioner

Although Eqs. (23) and (24) represent a great simplification of the original reduced MHD system, it is still cumbersome as a preconditioner because it requires inverting three operators (L_{S_L}, L_{Re}, ∇^2), and it is coupled. The last step toward a single parabolic operator is to implement a Jacobi iterative scheme in the $\delta\Phi$-equation (which lends the semi-implicit character to the preconditioner). For this, we split L_{Re} into its diagonal D_{Re} and off-diagonal M_{Re} parts and define the iterative procedure as

$$D_{\mathrm{Re}}\delta\Phi^{m+1}=-M_{\mathrm{Re}}\delta\Phi^m+\theta\mathbf{B}_0\cdot\nabla\delta\Psi^{m+1}+\nabla^{-2}[-G_\omega-L_{\mathrm{Re}}(-G_\Phi)],$$

where m is the Jacobi iteration count. Using $M_{\mathrm{Re}}=L_{\mathrm{Re}}-D_{\mathrm{Re}}$ and rearranging, we find

$$\delta\Phi^{m+1}=\delta\Phi^m+D_{\mathrm{Re}}^{-1}\{\theta\mathbf{B}_0\cdot\nabla\delta\Psi^{m+1}+\nabla^{-2}[-G_\omega-L_{\mathrm{Re}}(-G_\Phi)]-L_{\mathrm{Re}}\delta\Phi^m\}. \tag{25}$$

Combining this result and Eq. (23) (with $\delta\Psi=\delta\Psi^{m+1}$ and $\delta\Phi=\delta\Phi^{m+1}$), the final form of the semi-implicit preconditioner is the iterative procedure defined by Eq. (25) and

$$P_{SI}\delta\Psi^{m+1}=-G_\Psi+\theta(\mathbf{B}_0\cdot\nabla)\left\{D_{\mathrm{Re}}^{-1}\nabla^{-2}[-G_\omega-L_{\mathrm{Re}}(-G_\Phi)]+\left[I-D_{\mathrm{Re}}^{-1}L_{\mathrm{Re}}\right]\delta\Phi^m\right\}.$$

The parabolic operator $P_{SI}=L_{S_L}-\theta^2(\mathbf{B}_0\cdot\nabla)D_{\mathrm{Re}}^{-1}(\mathbf{B}_0\cdot\nabla)$ contains the Alfvén wave propagator $(\mathbf{B}_0\cdot\nabla)^2$ and is much better conditioned than the original hyperbolic system. As formulated, the preconditioner only requires the inversion of P_{SI} per Jacobi iteration (the inversion of ∇^2 is only required initially to form the right hand sides, and the diagonal matrix D_{Re} is trivially invertible).

Upon termination of the iteration, the vorticity is trivially found from

$$\delta\omega=\nabla^2\delta\Phi^{m+1}-G_\Phi.$$

The preconditioner step outlined above approximates the inversion of the block Jacobian system $J_k \delta \mathbf{x}_k = -\mathbf{G}(\mathbf{x}_k)$ [with J_k having the block structure given in Eq. (16)] as $\delta \mathbf{x}_k \approx P_k[-\mathbf{G}(\mathbf{x}_k)]$, where P_k is the preconditioner operator [Eq. (15)]. The generalization of the algorithm to apply P_k on arbitrary vectors (as required by the GMRES algorithm) is straightforward.

The next section dwells on some very specific details of the preconditioner implementation. These are not required to understand the discussion of the numerical results presented in Section 5.

4.3. Some Comments on the Implementation of the Preconditioner

In determining the optimal number of Jacobi iterations m_{max}, there is a trade-off between the efficiency and the effectiveness of the preconditioner algorithm. Thus, large values of m_{max} will result in an effective, but expensive, preconditioner, while small values of m_{max} will result in inexpensive preconditioner steps but may lose effectiveness, particularly in viscoresistive regimes (Stokes' limit) and/or regimes with large flows (although we have used the algorithm in Kelvin–Helmholtz configurations with super-Alfvénic flows [35] with little or no performance degradation). Here, a compromise is reached by using a moderate number of Jacobi iterations ($m_{max} = 4$), and making each iteration as inexpensive as possible by inverting P_{SI} approximately using a low-complexity MG solver. The latter consists of a single V-cycle of n_{grid} grid levels, determined as

$$n_{grid} = \text{int}\{\min[\log_2(N_x), \log_2(N_y)]\} - 2,$$

so that the coarsest grid in either axis (given by $\min[\frac{N_x}{2^{n_{grid}}}, \frac{N_y}{2^{n_{grid}}}]$) has at least four points to support the discretization stencil. The residual at each restriction and prolongation step is smoothed with three passes of point Jacobi (although line smoothing is preferable for MG as a stand-alone solver when $\Delta x \neq \Delta y$, its performance as a preconditioner is less sensitive to these types of issues [19, 20], and point smoothing presents other advantages for adaptive mesh refinement and parallelization). The Jacobi-smoothed MG solver for P_{SI} is implemented matrix-free, and only the diagonal elements of P_{SI} need be known to proceed, as discussed in [20]. This avoids forming and storing P_{SI}, but the matrix–vector product is nonoptimal. A crucial detail for the effectiveness of the preconditioner is to use the solution of the previous Jacobi iteration as the initial guess for the MG solve at the current iteration.

In general, the discretization in the preconditioner need not be of the same order of accuracy as in the original system of difference equations $\mathbf{G}(\mathbf{x}) = \mathbf{0}$. This is particularly important in the treatment of the advective term, since higher order discretizations of the advective term are not diagonally dominant for time steps larger than the flow CFL. Here, the preconditioner features in L_{S_L} and L_{Re} a first-order, upwinded discretization of advective terms, which increases diagonal dominance and produces a robust preconditioner [36].

The discretization of the Alfvén wave propagator in P_{SI} deserves some comments. Since the operator $(\mathbf{B}_0 \cdot \nabla) D_{\text{Re}}^{-1} (\mathbf{B}_0 \cdot \nabla)$ is self-adjoint and negative definite, its spatial discretization has been designed so that the corresponding matrix is symmetric negative definite while preserving a compact stencil support (in this case, a nine-point stencil).

5. RESULTS

In what follows, the computational domain is a uniformly discretized Cartesian rectangle of size $L_x \times 1$. Boundary conditions are periodic in x. In y, we impose no stress ($\omega = 0$), perfect conductor ($\Psi = 0$), and impenetrable wall ($\Phi = 0$). In all simulations, the Newton nonlinear convergence tolerance [Eq. (11)] is set to $\epsilon_{Newton} = 10^{-4}$, and the inexact Newton parameter is set to $\zeta = 0.05$.

5.1. Validation Tests

The code has been benchmarked successfully against explicit fluid and MHD codes in Kelvin–Helmholtz and tearing problems, respectively. In what follows, the code is tested by propagating the shear Alfvén wave, and by modeling classical resistive instabilities (tearing modes).

5.1.1. *Wave Propagation*

The theta differencing scheme allows dissipation-free wave propagation for $\theta = 1/2$. This has been tested by propagating shear Alfvén waves in a uniform magnetic field, with zero physical dissipation ($\nu = \eta = 0$) and $\theta = 1/2$ (zero numerical and physical dissipation is possible in this special case because the magnetic field is uniform, but it is not possible in general). The initial conditions are $\Psi_0 = -y$ (i.e., $B_{x,0} = 1$ and $B_{y,0} = 0$) and $\omega_0 = \Phi_0 = 0$. A standing Alfvén wave is excited with $\delta\Psi = 10^{-3} \sin(\pi y) \cos(\frac{2\pi}{L_x} x)$ and $\delta\Phi = 0$. Simulation parameters are $L_x = 3$ and $v_A = 1$ (due to normalization). The exact dispersion relation is $\omega = \pm k_x = \pm\frac{2\pi}{L_x}$, and the period is $T = \frac{2\pi}{|\omega|} = L_x = 3\,\tau_A$. The wave kinetic energy is followed in time with $\Delta t = \Delta t_{CFL}$ (where Δt_{CFL} is the shear Alfvén wave explicit CFL limit, given by $\Delta t_{CFL} = \frac{L_x}{N_x} = 0.047\tau_A$) on a 64×64 grid. The result is depicted in Fig. 1 (solid line) and shows that there is no amplitude decay, evidence of the absence of numerical dissipation, and that there are no secular errors in the average of the wave kinetic energy (the energy is purely oscillatory, neither increasing nor decreasing in average).

When larger time steps are employed ($\Delta t = 5\Delta t_{CFL}$, dashed line in Fig. 1), phase errors (dispersion) are starting to be apparent (the wave period increases slightly). This is a natural consequence of the implicit differencing. The apparent change in amplitude for $\Delta t = 5\Delta t_{CFL}$ is in fact periodic and due to sampling errors of the wave peak due to the large time step employed.

While taking even longer time steps with respect to the wave, CFL will inevitably result in an inaccurate description of the wave propagation speed, for long-time-scale phenomena the waves will remain balanced in spite of these phase errors because no operator splitting is introduced. Thus, larger implicit time steps will not result in numerical contamination of the solution on longer dynamical time scales due to errors propagating on the wave time scale. This is demonstrated numerically in Section 5.2.1.

5.1.2. *Classical Resistive Instabilities (Tearing Mode)*

The previous test checks linear wave propagation, in which only certain terms of the equations enter. Modeling resistive instabilities (tearing modes), however, brings in more physics and allows us to test both linear and nonlinear physics. The classical tearing mode problem is initialized with a Harris current sheet $\Psi_0(x, y) = \frac{1}{\lambda} \ln[\cosh \lambda(y - \frac{1}{2})]$ and $\Phi_0 = \omega_0 = 0$.

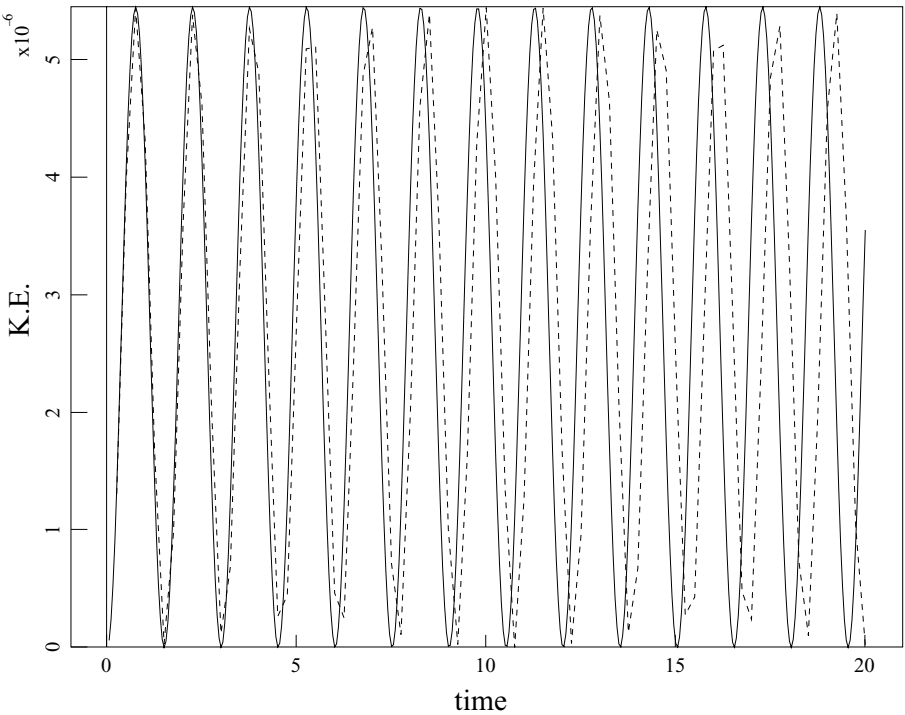

FIG. 1. Plots of the Alfvén wave kinetic energy (K.E.) as a function of time (in τ_A units), propagated with $\Delta t = \Delta t_{CFL} = 0.047\tau_A$ (solid) and $\Delta t = 5\Delta t_{CFL}$ (dashed).

The parameter λ is the inverse of the characteristic width of the current sheet and determines the tearing mode growth rate (the larger λ is, the narrower the current sheet and the larger the tearing growth rate are).

The mode is excited with a perturbation in the poloidal flux $\delta\Psi = 10^{-3}\sin(\pi y)\cos(\frac{2\pi}{L_x}x)$. The simulation parameters are $L_x = 3$, $\lambda = 5$, and $\mathrm{Re} = S_L = 10^3$. The simulation is performed in a 64×64 grid with $\Delta t = 5\tau_A = 100\Delta t_{CFL}$ until $T_f = 250\tau_A$, close to saturation. Plots of Ψ, Φ, ω, and the parallel current $j_\parallel = \nabla_\perp^2\Psi$ at T_f are depicted in Fig. 2, where the following features of tearing modes are visible:

1. The magnetic island ("cat's eye") is visible in the contours of the poloidal flux Ψ.
2. The flow organizes itself into four vortices of alternate sign of vorticity on the separatrix, as shown in the stream-function (Φ) plot.
3. The vorticity ω is strongly concentrated on the separatrix.
4. The parallel current $j_\parallel$ has a large perpendicular gradient at the separatrix, and $j_\parallel$ is nearly equal at the X- and O-points. The latter is expected to hold exactly at saturation, because $\partial_t\Psi = 0$ and $\mathbf{v}\cdot\nabla\Psi = \mathbf{B}\cdot\nabla\Phi = 0$ at the X- and O-points, and hence, from Ohm's law [Eq. (2)], $\eta j_\parallel = E_0$.

The tearing mode exponential growth rate for this simulation is $\gamma = 0.0435$. This value has been validated with results from other codes (specifically, an explicit version of this code—see Section 5.2.2—and a semi-Lagrangian, spectral explicit RMHD code). The theoretical scaling of γ with the Lundquist number S_L is $\gamma \sim S_L^{-3/5}$ for an inertial tearing mode (i.e., with negligible viscosity) and $\gamma \sim S_L^{-5/6}$ for a viscous tearing mode [37]. For a fixed viscosity ($\mathrm{Re} = 10^3$), the tearing mode will behave as viscous when S_L is large, and as inertial when

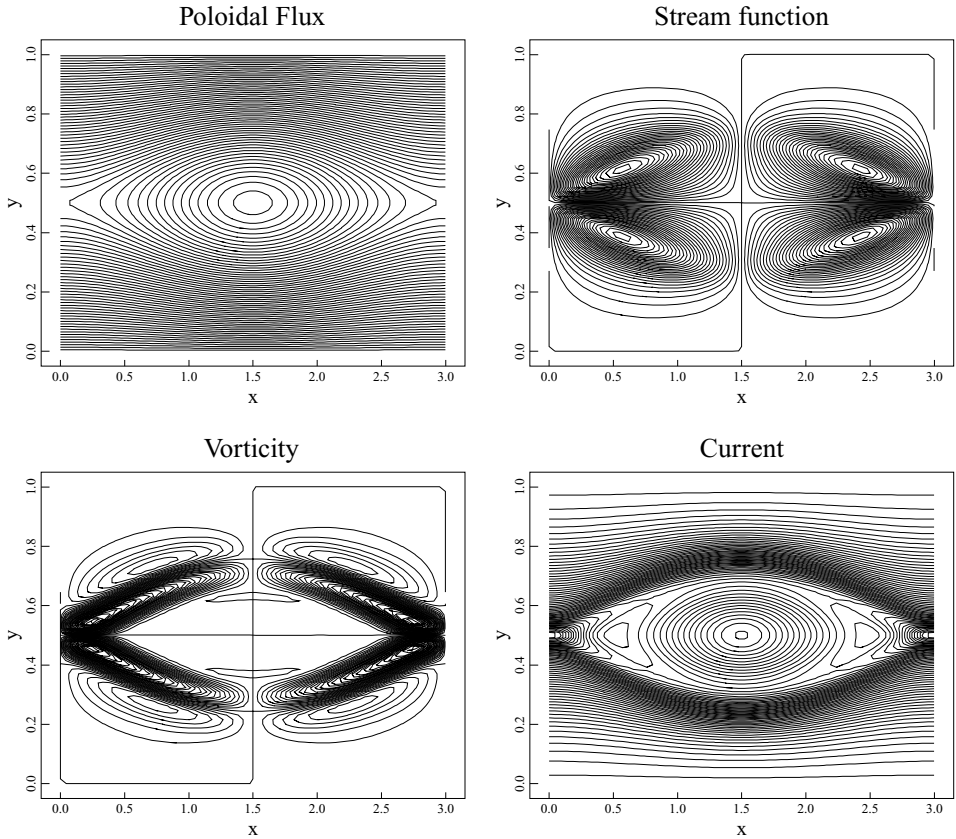

FIG. 2. Plots of poloidal flux function Ψ, stream function Φ, vorticity ω, and parallel current $j_\parallel = \nabla^2\Psi$, corresponding to a saturated tearing mode ($T_f = 250\tau_A$) with $L_x = 3$, $\lambda = 5$, $\mathrm{Re} = S_L = 10^3$ in a 64×64 grid.

S_L is small [37]. These results are reproduced by the code, as shown in Fig. 3. The scalings break down for small S_L, as expected.

5.2. Performance of the Algorithm

Maximizing the efficiency of the solver requires minimizing the CPU time required for a given computation. The CPU time can be functionally expressed as

$$CPU \propto N \times \frac{T_f}{\Delta t} \times n_{Nt} \times n_{GM} \times (a + b \times n_{GM}), \tag{26}$$

where $N = N_x N_y$ is the total number of mesh points, T_f is the final time (in τ_A units), Δt is the time step (in τ_A units), n_{GM} is the number of preconditioned GMRES iterations per Newton step, n_{Nt} is the number of Newton iterations per time step, and a, b are work factors. In Eq. (26), the term $n_{GM} \times (a + b \times n_{GM})$ represents the computational work of the GMRES algorithm, which is composed of two elements: (i) a linear term $a\, n_{GM}$ representing the work of routines associated with the GMRES algorithm (such as preconditioning calls), and (ii) a quadratic term $b\, (n_{GM})^2$ representing the work of the GMRES algorithm itself. Although typically $b \ll a$ (because the preconditioner is initially more expensive than the GMRES algorithm; see Table II), the quadratic term will dominate if n_{GM}

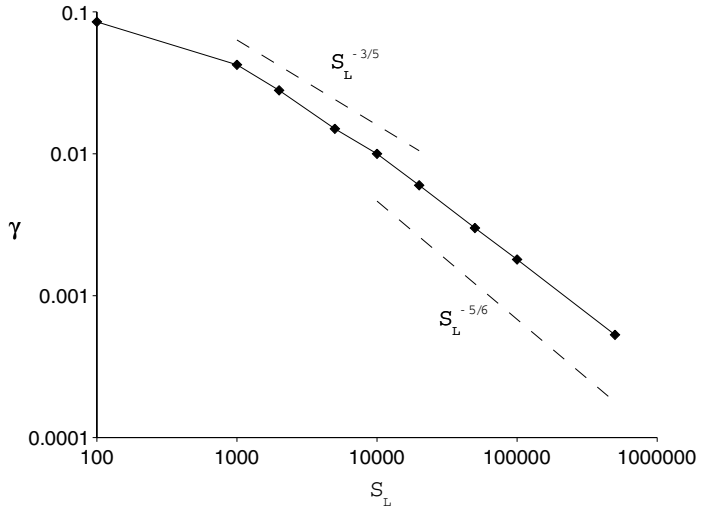

FIG. 3. Variation of the tearing mode growth rate γ with the Lundquist number S_L for a fixed Reynolds number $\mathrm{Re} = 10^3$. Different theoretical scalings are shown for comparison ($\gamma \sim S_L^{-5/6}$ for a viscous tearing mode, and $\gamma \sim S_L^{-3/5}$ for an inertial tearing mode).

is large enough. Hence, minimizing the CPU-time for given N, Δt, and T_f requires minimizing n_{GM}.

The tearing instability described in the previous section (with the same simulation parameters, unless otherwise specified) is chosen for the subsequent numerical experiments. Numerical data is averaged over five time steps, unless otherwise stated. CPU times are obtained in a single 600-MHz Pentium III processor.

5.2.1. *Performance in Accuracy*

The algorithm has been constructed to be second-order accurate in time. This is demonstrated numerically in Fig. 4, where the numerical error—computed at $T_f = 60\,\tau_A$—is plotted against $\gamma\Delta t$ (where $\gamma = 0.0435$) using the tearing problem in a 64×64 grid. The numerical error is measured as $\|\mathbf{\Psi} - \mathbf{\Psi}_g\|_2$, where $\mathbf{\Psi}$ is the poloidal flux solution vector obtained with arbitrary Δt, and $\mathbf{\Psi}_g$ is a "gauge" solution obtained with $\Delta t = \Delta t_{CFL} = 0.047\tau_A$. In Fig. 4, the error is computed using $\theta = 0.5$. These results demonstrate that the error scales as Δt^2.

The capability of the algorithm to take very large implicit time steps without sacrificing accuracy is evidenced in Fig. 5, where time histories of the ℓ_2-norm of the magnetic perturbation $\log \|\delta\Psi\|_2$ (global measure) and the current at the grid midpoint $j_\parallel(\frac{L_x}{2}, \frac{L_y}{2})$ (local measure) are compared for the second-order implicit code (CN) with a very small time step ($\Delta t = 0.047\tau_A$; $\gamma\Delta t = 0.002$; 1 Alfvén CFL in a 64×64 grid) and a very large time step ($\Delta t = 5\tau_A$; $\gamma\Delta t = 0.218$; 106 Alfvén CFLs in a 64×64 grid). Time histories obtained with a first-order predictor–corrector explicit solver (EX; see the next section) and first-order implicit solver [backward Euler (BE)] for a 64×64 grid have also been included for comparison. CN results show excellent agreement under time-step refinement (time histories are virtually superimposed, despite the fact that the large implicit time step is much larger than the explicit CFL limit and on the order of the dynamical time scale

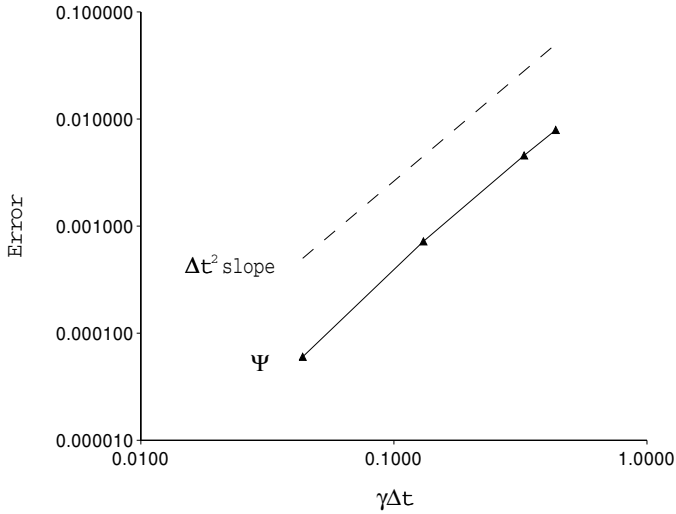

FIG. 4. Scaling of numerical error with $\gamma\,\Delta t$ (with $\gamma = 0.0436$) for the tearing instability in a 64×64 grid. The error is measured at $T_f = 60\tau_A$ by comparing the solution vector with a gauge solution, obtained with $\Delta t = \Delta t_{CFL}$.

of the problem). EX shows excellent agreement with CN. However, BE shows substantial differences with both CN and EX, indicating that second-order accuracy in time is crucial for accuracy when taking large implicit time steps, as expected.

The effect of different time step sizes on the tearing mode growth rate γ is indicated in Table I. Clearly, the algorithm preserves the accuracy of the solution even for $\gamma\,\Delta t \sim 0.44$ [i.e., of $O(1)$], an order of magnitude larger than the $\gamma\,\Delta t < 0.05$ limit characteristic of linearized semi-implicit implementations [5]. Again, the robustness of the answer is due to the second-order-accurate time stepping with no operator splitting. Incidentally, we note

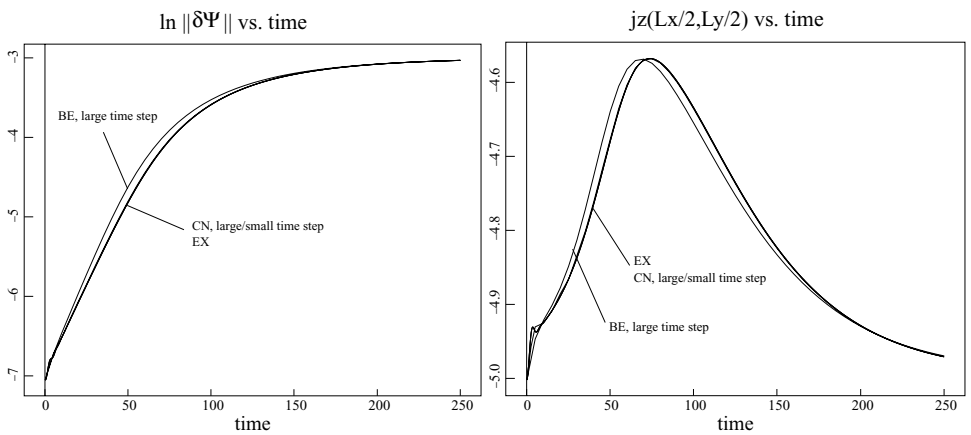

FIG. 5. Time histories of the ℓ_2-norm of the magnetic perturbation (global measure) and the current at the grid center point (local measure) for the tearing instability in a 64×64 grid using different Crank–Nicolson (CN) implicit time steps ($\Delta t = 0.047\tau_A$, $\Delta t = 5\tau_A$). Runs obtained with a first-order explicit (EX) and first-order implicit [backward Euler (BE)] time stepping are included for comparison.

TABLE I
Tearing Mode Growth Rate γ in a 64×64 Grid Using Different Time Steps

Δt	γ
$0.5\ \tau_A$	0.0436
$2.5\ \tau_A$	0.0432
$5\ \tau_A$	0.0435
$10\ \tau_A$	0.0447

that the theoretical CN error in approximating the exponential growth rate γ for $\gamma \Delta t = 1/2$ is $\sim 2\%$, consistent with the results in Table I.

5.2.2. *Performance in Efficiency*

Profiling results obtained for the tearing problem with $\Delta t = 5\tau_A$ in a 64×64 grid are presented in Table II. The most striking feature is that 77% of the CPU time is spent in the preconditioner. This indicates that the preconditioner—based on an optimal SI/MG algorithm—is responsible for the bulk of the CPU time. This is consistent with the alternate viewpoint of considering the Newton/GMRES algorithm as a nonlinear convergence accelerator of an otherwise inefficient nonlinear or linearized solver [38]. Such an accelerator, while ensuring overall nonlinear consistency, should have a small CPU overhead, as shown in Table II [as a side note, the small CPU overhead of GMRES justifies the previous assumption of $b \ll a$ in Eq. (26)]. It should also be noted that the Newton/GMRES accelerator, by converging on an unsplit nonlinear formulation of the equations, allows much larger time steps than would otherwise be possible with the linearized semi-implicit preconditioner as a stand-alone solver.

A grid convergence study is performed using the tearing problem. The number of Newton iterations per time step n_{Nt}, the number of GMRES iterations per Newton step n_{GM}, and the CPU time are monitored for different mesh refinements and time steps [measured in Alfvén CFL units, $\Delta t\,(CFL) = \Delta t / \Delta t_{CFL}$] for a run extending to $T_f = 30\,\tau_A$. The results

TABLE II
Breakdown of Total Wall-Clock Time Percentage Spent in Different Tasks of the Implicit Algorithm for the Tearing Problem with $\Delta t = 5\tau_A$ in a 64×64 Grid

Task	% total CPU time
Newton	99
GMRES	94
Preconditioner	77
MG solver	64

Note. CPU time breakdown is inclusive downward, i.e., every task includes all tasks below it.

TABLE III
Grid Convergence Study with $\Delta t = 20, 40, 160\Delta t_{CFL}$ for the Tearing Instability with $S_L = \mathrm{Re} = 10^3$

Δt	Grid	$\Delta t(\tau_A)$	n_{Nt}	n_{GM}	$\frac{\text{GMRES}}{\text{time step}}$	CPU time	$\widehat{CPU}$
$20\Delta t_{CFL}$	32×32	1.875	3	2.7	8	12.7	1.6
	64×64	0.9375	3	2.5	7.5	120	16
	128×128	0.46875	3	2.5	7.5	1128	150
	256×256	0.234375	3	3.5	10.4	13563	1304
$40\Delta t_{CFL}$	32×32	3.75	3	3.4	10.1	7.5	0.74
	64×64	1.875	3	3.3	10	72	7.2
	128×128	0.9375	3	3.5	10.6	723	68
	256×256	0.46875	3	5	15	9385	625
$160\Delta t_{CFL}$	32×32	15	3.5	6.3	22	3.5	0.16
	64×64	7.5	3.25	6	19	31	1.6
	128×128	3.75	3	6	18	277	15
	256×256	1.875	3	10.3	31	4367	141

Note. Results have been obtained for a run of $T_f = 30\tau_A$. $\widehat{CPU}$ is the CPU time normalized to $\frac{\text{GMRES}}{\text{time step}}$.

are presented in Table III and show the following:

1. The number of Newton iterations per time step remains virtually constant around 3, only increasing slightly for very large time steps.
2. For fixed $\Delta t(CFL)$, n_{GM} remains small (below 10 iterations even for extremely long time steps) and virtually constant with mesh refinement (by virtue of the MG preconditioning).
3. For fixed $\Delta t(CFL)$, the CPU time normalized to $\frac{\text{GMRES}}{\text{time step}} = n_{Nt} \times n_{GM}$, $\widehat{CPU}$, increases by ~9–10 when $N = N_x N_y$ increases by 4. This slightly exceeds the $O(N^{3/2})$ scaling expected from Eq. (26) (because $\Delta t \sim \frac{\Delta t(CFL)}{\sqrt{N}}$, and hence $\widehat{CPU} \sim \frac{N}{\Delta t} \sim \frac{N^{3/2}}{\Delta t(CFL)}$). This result is consistent with observations in previous work [17] and is due to cache memory effects as the problem size is increased.
4. For fixed N, $\frac{\text{GMRES}}{\text{time step}}$ scales vary sublinearly with $\Delta t(CFL)$.
5. For fixed N, $\widehat{CPU} \sim [\Delta t(CFL)]^{-1}$, as expected from Eq. 26.

The results in Table III suggest that employing the largest time step compatible with accuracy in a given calculation is the most efficient route. Note that for $S_L = \mathrm{Re} = 10^3$, the performance of the preconditioner degrades slightly for very fine meshes. This is due to the approximate inversion of the semi-implicit operator P_{SI} described in Section 4.2. The performance for fine meshes recovers when smaller diffusion coefficients are employed, as shown in Table IV, obtained for $S_L = \mathrm{Re} = 10^4$.

To gauge the performance of the implicit code against explicit methods, the code has been adapted to integrate the equations explicitly using Brailovskaya's first-order accurate, predictor–corrector explicit time discretization [39]. Van Leer's total-variation-diminishing method [40] is employed for monotonic, second-order advection. The elliptic coupling between vorticity and stream function is inverted at each time step using MG-preconditioned conjugate gradient (CG), with the previous time step solution as initial guess. This results in an average of <2 CG iterations per time step (Table V) for a residual convergence tolerance of

TABLE IV
Grid Convergence Study with S_L = Re = 10^4

Δt	Grid	$\Delta t\,(\tau_A)$	n_{Nt}	n_{GM}	$\frac{\text{GMRES}}{\text{time step}}$	CPU time	$\widehat{CPU}$
$20\Delta t_{CFL}$	32×32	1.875	3	2.6	7.8	12.8	1.6
	64×64	0.9375	3	2	5.9	102	17.3
	128×128	0.46875	2.8	1.4	3.8	793	209
	256×256	0.234375	3	1	3	6537	2179
$40\Delta t_{CFL}$	32×32	3.75	3	3.8	11.5	8.2	0.71
	64×64	1.875	3	3.3	10	73.6	7.4
	128×128	0.9375	3	2	6	517	86
	256×256	0.46875	3	1.6	5	4248	850
$160\Delta t_{CFL}$	32×32	15	3	9.3	28	4.2	0.15
	64×64	7.5	3	6.3	19	29	1.5
	128×128	3.75	3.1	4.6	14.2	234	16
	256×256	1.875	3.6	5.9	21.5	3220	150

10^{-4} (relative to the ℓ_2-norm of the independent term). The explicit time step is taken as [39]

$$\Delta t_{exp} = 0.9 \min\left[\left(\frac{v_{x,max}}{\Delta x} + \frac{v_{y,max}}{\Delta y} + \frac{B_{x,max}}{\Delta x} + \frac{B_{y,max}}{\Delta y}\right)^{-1}, \frac{\min(\text{Re}, S_L)}{2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)}\right], \quad (27)$$

where (v_x, v_y) are the velocity components and (B_x, B_y) are the magnetic field components (in typical simulations, $|\mathbf{v}| \ll |\mathbf{B}|$).

For the explicit/implicit comparison, the implicit time step is fixed to $\Delta t = 5\tau_A$. CPU times for different grid refinements are given in Table V. The implicit-to-explicit time step ratio is also presented, showing that $\Delta t/\Delta t_{exp} \propto \sqrt{N}$, as expected from the explicit wave CFL restriction (the Courant condition for the diffusion parameters chosen starts being relevant for $\sqrt{N} > 500$). Note that for the finest grid considered (256×256), the implicit time step is ~600 times larger than the explicit time step, and $CPU_{exp}/CPU \sim 8$. This demonstrates the efficiency advantage of the implicit method even for moderate (64×64) grid refinements, and more so since there is still room for optimization in the implicit solver by avoiding the matrix-free implementation of the operator P_{SI} (Section 4.3).

TABLE V
CPU Time Comparison between Implicit and Explicit Time Algorithms for a $T_f = 10\tau_A$ Run Using the Tearing Problem

Grid	Explicit	Implicit ($\Delta t = 5\tau_A$)	$\Delta t/\langle\Delta t_{exp}\rangle$
64×64	47 s (1.51)	11 s	142
128×128	437 s (1.61)	73 s	294
256×256	4604 s (1.90)	588 s	578

Note. In parentheses is shown, for the explicit runs, the average number of CG iterations per time step. The ratio of the implicit time step ($\Delta t = 5\tau_A$) to the average explicit time step [Eq. (27)] is also presented.

6. CONCLUSIONS

An efficient, fully implicit, fully nonlinear, second-order-accurate 2D reduced visco-resistive MHD solver has been implemented using Jacobian-free Newton–Krylov techniques (GMRES). Convergence is accelerated with a "physics-based" preconditioner, in which an approximate semi-implicit solution of the original difference equations is employed. The preconditioner deals with a well-conditioned parabolic system (instead of hyperbolic, thereby avoiding the wave stiffness) and employs multigrid methods to invert the resulting operators (thereby removing grid stiffness).

The algorithm has been benchmarked by propagating the supported shear Alfvén waves, and by modeling resistive instabilities (tearing modes). A grid convergence study of the implicit solver shows that the performance of the algorithm is very competitive with respect to the problem size [because $n_{GM} \sim O(N^0)$, with $N = N_x N_y$] and the time-step size (because n_{GM} varies very sublinearly with the time step). As a consequence, the algorithm shows a substantial reduction of the computational expense with increasing time steps, and a favorable scaling with N.

The implicit algorithm has demonstrated the capability of accurately taking time steps on the order of the time scale of interest (with $\gamma \Delta t \approx 0.5$). CPU time gains of the implicit approach over explicit methods of an order of magnitude have also been demonstrated.

Despite the simplicity of the MHD model used here (reduced, 2D), it has provided valuable insight into the development of implicit solvers for multiple-time-scale systems supporting stiff waves. The approach presented here should carry over to three-dimensional MHD models, since no specifics about the dimensionality of the system are involved in the procedure, and successful three-dimensional applications of MG-preconditioned Newton–Krylov algorithms exist [41]. Also, the conceptual framework presented here provides an excellent starting point to develop preconditioners for more complete MHD models (including, for instance, compressibility, which brings in the fast magnetosonic wave), since the direct substitution step that is the basis of the preconditioning strategy presented here can be done with great generality (in fact, it can be argued that the RMHD model presents a perverse example of a hyperbolic system due to the presence of $\partial_t \nabla^2 \Phi$ and, hence, is a particularly challenging problem).

Another relevant issue is the use of realistic plasma transport parameters (particularly in three dimensions), which are typically orders of magnitude smaller than those employed in the numerical experiments presented here. Credible modeling of such conditions requires adaptive mesh refinement capabilities. Recently, Hornung and Pernice [42] have demonstrated the use of Jacobian-free Newton–Krylov methods on structured adaptive mesh refinement (SAMR) grids through the development of interfaces between SAMRAI (the structured adaptive mesh refinement applications infrastructure [43]) and the PETSc (portable, extensible toolkit for scientific computing [44, 45]) package. The future emphasis of this project is extending the current implementation to use SAMR grids.

ACKNOWLEDGMENTS

L.C. acknowledges useful discussions with Dr. Giovanni Lapenta. L.C. has been funded by a Director Postdoctoral Fellowship at Los Alamos National Laboratory.

REFERENCES

1. I. Lindemuth and J. Killeen, Alternating direction implicit techniques for two-dimensional magnetohydrodynamic calculations, *J. Comput. Phys.* **13**, 181 (1973).
2. D. Schnack and J. Killeen, Nonlinear, two-dimensional magnetohydrodynamic calculations, *J. Comput. Phys.* **35**, 110 (1980).
3. D. S. Harned and W. Kerner, Semi-implicit method for three-dimensional compressible magnetohydrodynamic simulation, *J. Comput. Phys.* **60**, 62 (1985).
4. D. S. Harned and D. D. Schnack, Semi-implicit method for long time scale magnetohydrodynamic computations in three dimensions, *J. Comput. Phys.* **65**, 57 (1986).
5. D. D. Schnack, D. C. Barnes, D. S. Harned, and E. J. Caramana, Semi-implicit magnetohydrodynamic calculations, *J. Comput. Phys.* **70**, 330 (1987).
6. D. S. Harned and Z. Mikic, Accurate semi-implicit treatment of the Hall effect in magnetohydrodynamic computations, *J. Comput. Phys.* **83**, 1 (1989).
7. J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations* (Chapman & Hall, London/New York, 1989).
8. E. J. Caramana, Derivation of implicit difference schemes by the method of differential approximation, *J. Comput. Phys.* **96**, 484 (1991).
9. O. S. Jones, U. Shumlak, and D. S. Eberhardt, An implicit scheme for nonideal magnetohydrodynamics, *J. Comput. Phys.* **130**, 231 (1997).
10. A. Hujeirat, IRMHD: an implicit radiative and magnetohydrodynamical solver for self-gravitating systems, *Mon. Not. R. Astron. Soc.* **298**, 310 (1998).
11. A. Hujeirat and R. Rannacher, On the efficiency and robustness of implicit methods in computational astrophysics, *New Astron. Rev.* **45**, 425 (2001).
12. A. Hujeirat and R. Rannacher, A method for computing compressible, highly stratified flows in astrophysics based on operator splitting, *Int. J. Numer. Methods Fluids* **28**, 1 (1998).
13. Y. Saad, *Iterative Methods for Sparse Linear Systems* (PWS Publishing, Boston, 1996).
14. T. F. Chan and K. R. Jackson, Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms, *SIAM J. Sci. Stat. Comput.* **5**, 533 (1984).
15. P. N. Brown and Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, *SIAM J. Sci. Stat. Comput.* **11**, 450 (1990).
16. Y. Saad and M. Schultz, GMRES: A generalized minimal residual algorithm for solving non-symetric linear systems, *SIAM J. Sci. Stat. Comput.* **7**, 856 (1986).
17. V. A. Mousseau, D. A. Knoll, and W. J. Rider, Physics-based preconditioning and the Newton–Krylov method for non-equilibrium radiation diffusion, *J. Comput. Phys.* **160**, 743 (2000).
18. D. A. Knoll, W. B. Vanderheyden, V. A. Mousseau, and D. B. Kothe, On preconditioning Newton–Krylov methods in solidifying flow applications, *SIAM J. Sci. Comput.* **23**(2), 381 (2001).
19. D. A. Knoll and W. J. Rider, A multigrid preconditioned Newton–Krylov method, *SIAM J. Sci. Comput.* **21**(2), 691 (1999).
20. D. A. Knoll, G. Lapenta, and J. U. Brackbill, A multilevel iterative field solver for implicit, kinetic, plasma simulation, *J. Comput. Phys.* **149**, 377 (1999).
21. L. Chacón, D. C. Barnes, D. A. Knoll, and G. H. Miley, An implicit energy-conservative 2D Fokker-Planck algorithm. II. Jacobian-free Newton-Krylov solver, *J. Comput. Phys.* **157**, 654 (2000).
22. H. R. Strauss, Nonlinear, 3-dimensional magnetohydrodynamics of noncircular tokamaks, *Phys. Fluids* **19**(1), 134 (1976).
23. J. F. Drake and T. M. Antonsen, Nonlinear reduced fluid equations for toroidal plasmas, *Phys. Fluids* **27**(4), 898 (1984).
24. R. D. Hazeltine, M. Kotschenreuther, and P. J. Morrison, A four-field model for tokamak plasma dynamics, *Phys. Fluids* **28**(8), 2466 (1985).
25. M. Luskin and R. Rannacher, On the smoothing property of the Crank-Nicolson scheme, *Appl. Anal.* **14**, 117 (1982).

26. R. Rannacher, Finite element solution of diffusion problems with irregular data, *Numer. Math*. **43**, 309 (1984).
27. B. P. Leonard, A stable and accurate convective modelling procedure based on quadratic upstream interpolation, *Comput. Methods Appl. Mech. Eng.* **19**, 59 (1979).
28. P. R. McHugh and D. A. Knoll, Inexact Newton's method solution to the incompressible Navier-Stokes and energy equations using standard and matrix-free implementations, *AIAA J.* **32**(12), 2394 (1994).
29. D. A. Knoll and P. R. McHugh, Enhanced nonlinear iterative techniques applied to a nonequilibrium plasma flow, *SIAM J. Sci. Comput.* **19**, 291 (1998).
30. R. Dembo, S. Eisenstat, and R. Steihaug, Inexact Newton methods, *J. Numer. Anal.* **19**, 400 (1982).
31. C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations* (Soc. for Industr. & Appl. Math., Philadelphia, 1995).
32. D. A. Knoll and V. Mousseau, On Newton-Krylov multigrid methods for the incompressible Navier-Stokes equations, *J. Comput. Phys.* **163**, 262 (2000).
33. W. L. Briggs, *A Multigrid Tutorial* (Soc. for Industr. & Appl. Math., Philadelphia, 1987).
34. F. H. Harlow and A. A. Amsden, A numerical fluid dynamical calculation method for all flow speeds, *J. Comput. Phys.* **8**, 197 (1971).
35. D. A. Knoll and L. Chacón, Magnetic reconnection in the two-dimensional Kelvin–Helmholtz instability, *Phys. Rev. Lett.*, to appear.
36. D. A. Knoll, An improved convection scheme applied to recombining divertor plasma flows, *J. Comput. Phys.* **142**, 473 (1998).
37. H. P. Furth, J. Killeen, and M. N. Rosenbluth, Finite-resistivity instabilities of a sheet pinch, *Phys. Fluids* **6**(4), 459 (1963).
38. T. Kerkhoven and Y. Saad, On acceleration methods for coupled nonlinear elliptic systems, *Numer. Math.* **60**, 525 (1992).
39. J. C. Tannehill, D. A. Anderson, and R. H. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, 2nd ed. (Taylor & Francis, London, 1997).
40. B. Van Leer, Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection, *J. Comput. Phys.* **23**, 276 (1977).
41. M. Pernice and M. D. Tocci, A multigrid-preconditioned Newton-Krylov method for the incompressible Navier-Stokes equations, *SIAM J. Sci. Comput.* **23**(2), 398 (2001).
42. M. Pernice and R. Hornung, A nonlinear solvers package for SAMRAI (2002). Available at: http://www.c3.lanl.gov/~pernice/samrai/docs/nlsolvers/html/index.html.
43. R. Hornung, SAMRAI: Structured adaptive mesh refinement application infrastructure (2001). Available at: http://www.llnl.gov/CASC/SAMRAI/.
44. S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in *Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen (Birkhäuser, Basel, 1997), p. 163.
45. S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *PETSc Users Manual,* Technical Report ANL-95/11—Revision 2.1.0, Argonne National Laboratory (2001).