Scalable parallel implicit solvers for 3D magnetohydrodynamics

L Chacón
Los Alamos National Laboratory, Los Alamos, NM 87545 USA
E-mail: chacon@lanl.gov

Abstract. The efficient integration of the extended magnetohydrodynamics (XMHD) model is critical for present and future modeling needs of fusion plasmas and is at the core of several interacting SciDAC centers. However, XMHD is a remarkably challenging system of PDEs, featuring a stiff multiscale character both in time and space. Such stiff character has hindered progress toward a truly scalable, efficient, and accurate nonlinear XMHD algorithm. In this paper, we discuss recent algorithmic developments that demonstrate that such an XMHD algorithm is within reach. The approach is based on fully implicit time integration and employs parallel Jacobian-free Newton-Krylov technology, preconditioned with multigrid methods for algorithmic scalability. The algorithm has been tested on a variety of 2D and 3D configurations that demonstrate its excellent algorithmic scalability properties, both serially and in parallel up to 4096 processors and 134 million unknowns.

1. Introduction
With the advent of petascale computing, fundamental scientific understanding will be enabled by the possibility of ever more realistic modeling of complex physical systems. This is, in fact, the ultimate goal of SciDAC. However, hardware power alone is not sufficient to enable such massive computations. Algorithms, as has been recognized in various DOE [1, 2] and NSF [3] reports, are as important, if not more. The emphasis is, or course, on algorithmic scalability, namely, that the computational complexity of a given algorithm scales linearly with the number of unknowns, and inversely proportional to the number of processors.

In this paper, we focus on the development of such an algorithmic capability for the extended magnetohydrodynamic (XMHD) model, which is relevant in modeling various plasmas of interest to DOE such as solar, magnetospheric, and laboratory (e.g., fusion) plasmas. This paper builds on recently published work [4], where a scalable, parallel, fully implicit, fully nonlinear solver for the 3D resistive compressible MHD equations was demonstrated, and augments this work in two fundamental ways. First, we generalize the formulation in the reference (which was limited to small flows) to arbitrary-size plasma flows. This is of interest to enhance the robustness of the solver in the presence of large plasma flows that may appear locally during the simulation (e.g., due to magnetic reconnection processes) and to enable direct-to-steady-state solution procedures. Second, we generalize the formulation in [4] to include electron Hall effects. As a proof of principle, we consider the cold-ion limit to demonstrate the soundness of the approach.

As in [4], we base our nonlinear solver approach on the Newton-Raphson iterative algorithm. Krylov iterative techniques [5], implemented Jacobian-free [6, 7] (i.e., without ever forming and storing the Jacobian matrix) for memory efficiency, are employed for the required algebraic
matrix inversions. Here, FGMRES (Flexible Generalized Minimal RESiduals [8, 9]) is employed as the Krylov solver of choice, because of the lack of symmetry in the algebraic system of interest. The flexible character of FGMRES relaxes some of the constraints in the preconditioner step of regular GMRES, which we have found useful in our implementation. In particular, FGMRES allows the preconditioner to change between successive GMRES iterations. For parallelization, we employ the PETSc library [10, 11, 12].

The efficiency and scalability of Krylov methods depend strongly on adequate preconditioning [5]. Here, as described above, we extend previous work on 3D resistive MHD to include arbitrary flows and electron Hall effects. As in [13, 4], a suitable multigrid-based preconditioner is developed around the parabolization concept, whereby a hyperbolic system is reformulated as a parabolic one, which is in turn amenable to a multilevel treatment. The connection between the parabolization procedure and the Schur block decomposition, outlined first in [13], and extended to 3D resistive MHD in [4], is suitably generalized here to accommodate the new physics of interest. We shall show that such generalization is simple conceptually and of straightforward implementation.

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. Section 4 discusses the physics-based preconditioner for this particular application is discussed. Section 5 presents serial and parallel efficiency results of the resulting implicit algorithm in various 2D and 3D configurations. We conclude in Section 6.

2. Three-dimensional Hall MHD model

We consider the dimensionless compressible Hall MHD model, given in Alfvénic units (i.e., Alfvén speed \( v_A = B_0/\sqrt{\rho_0\mu_0} \), Alfvén time \( \tau_A = L/v_A \), where \( B_0, \rho_0, \) and \( L \) are the reference magnetic field, density, and length, respectively) by

\[
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v} - D\nabla \rho) = 0, \tag{1}
\]

\[
\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} = 0 \tag{2}
\]

\[
\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{vv} - \mathbf{BB}) + \mathbb{I}(p + \frac{B^2}{2}) - \rho \nu \nabla \mathbf{v} = 0, \tag{3}
\]

\[
\frac{\partial T}{\partial t} + \mathbf{v} \cdot \nabla T + (\gamma - 1) \left[ T\nabla \cdot \mathbf{v} - \frac{\kappa \nabla^2 T + Q}{\alpha T \rho} \right] = 0, \tag{4}
\]

with \( \mathbb{I} \) the identity operator, \( p = (1 + \alpha_T)\rho T \) the pressure, \( \alpha_T = T_i/T_e \) the ion/electron temperature ratio, and \( \rho \) the particle density. In these equations, \( \mathbf{v} \) is the plasma velocity; \( \mathbf{B} \) is the magnetic field; \( \eta \) and \( \nu \) are the reciprocals of the Lundquist number and the Reynolds number, respectively; \( D \) is a dimensionless particle diffusivity that models cross-field particle diffusion (such particle diffusivity \( D \) is not self-consistently considered in equation 3 and hence should be regarded as ad hoc); and \( Q = \eta j^2 + \rho \nu \nabla \mathbf{v} : \nabla \mathbf{v} \) contains the Joule and viscous heating sources. Simple closures for the heat flux \( -\kappa \nabla T_e \) and the viscous stress tensor \( -\rho \nu \nabla \mathbf{v} \) are considered at this stage. More accurate closures (such as parallel electron heat transport and gyro-viscous stresses) will be considered in future work. The electric field \( \mathbf{E} \) is given by

\[
\mathbf{E} = -\mathbf{u} \times \mathbf{B} + \eta \nabla \times \mathbf{B} + \frac{d_i}{\rho} \left( j \times \mathbf{B} - \nabla p_e \right), \tag{5}
\]

where \( p_e = \rho T \) is the electron pressure and \( d_i = c/(\omega_{pi}L) \) is the ion skin depth, and is a measure of the importance of two-fluid physics for a given simulation.
The system in equations 1–4 is hyperbolic, supporting a variety of linear and dispersive normal modes, depending on the value of \( kd_i \), with \( k \) the wavenumber. Linear modes \( (kd_i \ll 1) \) include fast and slow magnetoacoustic waves, and the shear Alfvén wave. Dispersive modes \( (kd_i \gg 1) \) include Whistler and kinetic Alfvén wave. In some applications (notably, in magnetic fusion confinement), dynamical time scales of interest are much slower than those associated with these normal modes. In such contexts, an implicit approach that steps over the normal-mode time scales to resolve the time scales of interest is useful. This is the subject of this paper. However, as we shall see, the hyperbolic character of the MHD model makes the task of developing an optimal, scalable solver difficult.

A word about the discretization of equations 1–4 is in order. Spatially, the system is discretized by using finite volumes, as detailed in [14]. Such spatial discretization has proved to be conservative, solenoidal in the magnetic field to numerical round-off, and remarkably robust in the absence of physical and/or numerical dissipation. Temporally, we employ a \( \theta \)-scheme, with \( \theta = 0.5 \) (second-order Crank-Nicolson). This choice results in a set of nonlinear algebraic equations \( \mathbf{G}(\mathbf{x}) = 0 \), with \( \mathbf{x}^T = (\rho, T, \mathbf{B}, \mathbf{v}) \), that needs to be inverted every time step. For this, we employ preconditioned Newton-Krylov methods. The next section introduces these methods and some of their properties.

3. Jacobian-free Newton-Krylov methods

We give a brief introduction to Jacobian-free Newton-Krylov methods (JFNK). The motivated reader can find extensive discussions on this approach elsewhere [15]. Newton’s method solves the nonlinear system \( \mathbf{G}(\mathbf{x}) = 0 \) iteratively by inverting linear systems of the form

\[
\frac{\partial \mathbf{G}}{\partial \mathbf{x}}|_k \delta \mathbf{x}_k = -\mathbf{G}(\mathbf{x}_k),
\]

with \( \mathbf{x}_{k+1} = \mathbf{x}_k + \delta \mathbf{x}_k \). Nonlinear convergence is achieved when

\[
\| \mathbf{G}(\mathbf{x}_k) \|_2 < \epsilon_a + \epsilon_r \| \mathbf{G}(\mathbf{x}_0) \|_2 = \epsilon_t,
\]

where \( \| \cdot \|_2 \) is the \( \ell_2 \)-norm (Euclidean norm), \( \epsilon_a = \sqrt{N} \times 10^{-15} \) (with \( N \) the total number of degrees of freedom) is an absolute tolerance to avoid converging below round-off, \( \epsilon_r \) is the Newton relative convergence tolerance (set to \( 10^{-4} \) in this work), and \( \mathbf{G}(\mathbf{x}_0) \) is the initial residual.

Such linear systems are solved iteratively with Krylov methods, which require only matrix-vector products to proceed. Because the linear system matrix is a Jacobian matrix, such matrix-vector products can be implemented Jacobian-free by using the Gateaux derivative

\[
\frac{\partial \mathbf{G}}{\partial \mathbf{x}}|_k \mathbf{v} = \lim_{\epsilon \to 0} \frac{\mathbf{G}(\mathbf{x}_k + \epsilon \mathbf{v}) - \mathbf{G}(\mathbf{x}_k)}{\epsilon},
\]

where in practice a small but finite \( \epsilon \) is employed (p. 79 in [15]). Thus, the evaluation of the Jacobian-vector product requires only the function evaluation \( \mathbf{G}(\mathbf{x}_k + \epsilon \mathbf{v}) \), and there is no need to form or store the Jacobian matrix. This, in turn, allows for a memory-efficient implementation.

An inexact Newton method [16] is used to adjust the convergence tolerance of the Krylov method at every Newton iteration according to the size of the current Newton residual, as follows:

\[
\| J_k \delta \mathbf{x}_k + \mathbf{G}(\mathbf{x}_k) \|_2 < \zeta_k \| \mathbf{G}(\mathbf{x}_k) \|_2 ,
\]

where \( \zeta_k \) is the inexact Newton parameter and \( J_k = \frac{\partial \mathbf{G}}{\partial \mathbf{x}}|_k \) is the Jacobian matrix. Thus, the convergence tolerance of the Krylov method is loose when the Newton state vector \( \mathbf{x}_k \) is far from the nonlinear solution but tightens as \( \mathbf{x}_k \) approaches the solution. Hence, the linear solver
works the hardest when the Newton state vector is closest to the nonlinear root. Superlinear convergence rates of the inexact Newton method are possible if the sequence of $\zeta_k$ is chosen properly (p. 105 in [15]). Here, we employ the same prescription as in [13]:

$$\zeta_k^A = \gamma \left( \frac{\|G(x_k)\|_2}{\|G(x_k-1)\|_2} \right)^\alpha,$$

$$\zeta_k^B = \min(\zeta_{max}, \max(\zeta_k^A, \gamma \zeta_{k-1}^A)),$$

$$\zeta_k = \min(\zeta_{max}, \max(\zeta_k^B, \gamma \epsilon_t \|G(x_k)\|_2)),$$

with $\alpha = 1.5$, $\gamma = 0.9$, and $\zeta_{max} = 0.8$. The convergence tolerance $\epsilon_t$ is defined in equation 6.

In this prescription, the first step ensures superlinear convergence (for $\alpha > 1$), the second avoids volatile decreases in $\zeta_k$, and the last avoids oversolving in the last Newton iteration.

A further advantage of Krylov methods is that they can be preconditioned by considering the alternate (but equivalent) systems $J_kP_k^{-1}J_k\delta x_k = -G_k$ (right preconditioning) or $P_k^{-1}J_k\delta x_k = -P_k^{-1}G_k$ (left preconditioning). Such preconditioned systems can be straightforwardly and efficiently implemented in the Krylov algorithm as two consecutive matrix-vector products. A crucial feature of preconditioning is that, while it can substantially improve the convergence properties of the Krylov iteration if $P_k^{-1} \approx J_k^{-1}$, it does not alter the solution of the Jacobian system upon convergence (because the solution $\delta x_k$ of the preconditioned system is the same as that of the original system). Therefore, one can explore suitable approximations in the preconditioner for efficiency purposes without compromising the accuracy of the converged result.

We next discuss our approach to preconditioning.

4. Preconditioning

JFNK requires preconditioning for algorithmic scalability (i.e., with convergence rates independent of the number of unknowns and the number of processors considered). Multigrid methods (MG) have been shown in a variety of applications [17] to produce such optimal JFNK convergence rates, and are at the core of our approach. In particular, our work builds on previous developments in 2D resistive [18] and Hall [13] reduced MHD and 3D resistive MHD [4], where such optimal behavior has been demonstrated. In these references, the key for an effective multigrid implementation was the parabolization of otherwise hyperbolic PDEs. The approach, which was termed “physics-based,” aimed at reformulating the semi-discrete (temporally discrete, spatially continuous) set of PDEs into a diagonally dominant set, amenable to classical smoothing (based on stationary iterative techniques) in an MG setting. MG methods employ a divide-and-conquer approach where multiple grids of varying refinement are employed [19, 20]. The underlying idea is that oscillatory components of the error can be readily attached at a given grid level (with a so-called smoother), but smooth ones are difficult. In its simplest form (an MG V-cycle), the procedure then involves smoothing the error on a given level and then coarsening (restricting) the smooth components to the next coarse grid level. In the new level, some of the smooth components will appear oscillatory and therefore can be subjected to further smoothing. The process is performed recursively until the coarsest grid level is reached, at which point a direct solve can be performed very cheaply. The solution is then interpolated (prolongated) up the grid hierarchy, until the finest level is reached. While variations of the basic V-cycle are possible to improve the convergence rate of a given MG implementation (p. 47 in [19]), one or several V-cycles are generally enough for preconditioning purposes. As can be understood from the description, the crucial element for a working MG solver is the availability of a smoother. While smoothers can be found fairly easily for diagonally dominant systems (in a point or block sense; p. 96 in [20]), it is remarkably hard otherwise.

Hyperbolic systems (such as MHD) can be shown to be diagonally submissive when time steps larger than the explicit CFL stability constraint are employed (see Ref. [18] for an in-depth
explanation of this issue). This, in turn, hinders the task of finding a suitable smoother for MG. However, hyperbolic systems can be conveniently parabolized in an implicit time-stepping setting [18, 13, 4]. The basic idea is to produce a well-conditioned (diagonally dominant) parabolic operator from an ill-conditioned hyperbolic system of equations. Algebraically, the parabolization procedure is equivalent to a Schur complement factorization. For 3D resistive MHD (we will generalize this to Hall MHD in Sec. 4.2), one can succinctly write the Jacobian matrix for the linearized system in terms of the linear updates \( \delta \rho, \delta T, \delta B, \) and \( \delta v \) as [4]

\[
J \delta x = \begin{bmatrix}
D_\rho & 0 & 0 & U_{\rho v} \\
0 & D_T & 0 & U_{Tv} \\
0 & 0 & D_B & U_{Bv} \\
L_{v\rho} & L_{v T} & L_{v B} & D_v
\end{bmatrix}
\begin{bmatrix}
\delta \rho \\
\delta T \\
\delta B \\
\delta v
\end{bmatrix}.
\]

The diagonal blocks \( D_\rho, D_T, D_B, D_v \) are given in [4] and contain advection-diffusion terms, which can be readily inverted by using MG if upwinding is employed for the advective terms (only in the preconditioning stage [18, 13]). Off-diagonal blocks \( L \) and \( U \) (also given in the reference) contain all relevant hyperbolic couplings. The Jacobian matrix has an “arrow” structure, which suggests considering the following 2x2 block structure for analysis purposes:

\[
J \delta x = \begin{bmatrix}
M & U \\
L & D_v
\end{bmatrix}
\begin{bmatrix}
\delta y \\
\delta v
\end{bmatrix},
\]

where \( \delta y = (\delta \rho, \delta T, \delta B)^T \), and

\[
M = \begin{pmatrix}
D_\rho & 0 & 0 \\
0 & D_T & 0 \\
0 & 0 & D_B
\end{pmatrix}.
\]

The block \( M \) can be easily invertible, since \( M \) is block diagonal, and, as stated earlier, the blocks themselves are amenable to MG techniques. The Schur factorization of the inverse of the 2x2 block Jacobian matrix yields

\[
\begin{bmatrix}
M & U \\
L & D_v
\end{bmatrix}^{-1} = \begin{bmatrix}
I & -M^{-1}U \\
0 & I
\end{bmatrix} \begin{bmatrix}
M^{-1} & 0 \\
0 & P_{\text{Schur}}^{-1}
\end{bmatrix} \begin{bmatrix}
I & 0 \\
LM^{-1} & I
\end{bmatrix},
\]

where \( P_{\text{Schur}} = D_v - LM^{-1}U \) is the Schur complement, which contains all the information from the off-diagonal blocks \( L \) and \( U \). At this point, the MHD system has been effectively parabolized. The Schur factorization translates into the following three-step exact inversion algorithm (with \( G \) the nonlinear function residual):

- **Predictor**: \( \delta y^* = -M^{-1}G_y \),
- **Velocity update**: \( \delta v = P_{\text{Schur}}^{-1}[-G_v - L\delta y^*] \),
- **Corrector**: \( \delta y = \delta y^* - M^{-1}U\delta v \).

(9)

We note that, at this point, no approximations have been introduced, and the exact Jacobian inverse requires only finding \( M^{-1} \) and \( P_{\text{Schur}}^{-1} \). Suitable approximations in the preconditioning strategy will use equation 9 as a starting point.
4.1. Preconditioner for arbitrary plasma flow

As it stands, inverting \( P_{\text{Schur}} \) is impractical because of the presence of \( M^{-1} \). In [4], the small flow limit was considered, in which \( M^{-1} \approx \Delta t \mathbb{I} \) (i.e., advection and diffusion were neglected). In this section, we derive a tractable preconditioner formulation that 1) does not require \( M^{-1} \) to form \( P_{\text{Schur}} \), and 2) is valid for arbitrary flows. Our starting point is the Schur complement, 
\[
P_{\text{Schur}} = D_\nu - LM^{-1}U. \tag{10}
\]
which eliminates the need of calculating \( M^{-1} \), does not require the inverse of \( M_\nu \), and features the same operator \( LU \) as in the small-flow limit formulation. The resulting \( P_s^{-1} \) can be readily implemented in equation (9) by inverting \( (D_\nu M_\nu - LU)^{-1} \) (using matrix-light multigrid techniques, as described in [4]), followed by an application of \( D_\nu M_\nu \). By comparison with the small-flow approach, the arbitrary-flow approach only requires an application of \( M_\nu \) in the velocity update and an additional inversion of \( M^{-1} \) in the corrector step, making it slightly more expensive.

There remains to find a suitable operator \( M_\nu \). For this, we notice that all diagonal blocks in \( M \) correspond to the discretization of time-dependent advection-diffusion equations and are therefore similar. Furthermore, the upper blocks \( U \),
\[
\begin{align*}
U_{\nu \nu} \delta \nu &= \theta \nabla \cdot (\rho_0 \delta \nu), \\
U_{T\nu} \delta \nu &= \theta (\gamma - 1) T_0 \nabla \cdot \delta \nu, \\
U_{B\nu} \delta \nu &= \theta \nabla \cdot [\delta \nu (B_0 - B_\nu) - D_\nu \nabla \rho_0 - D_\nu \nabla \rho_0]\,
\end{align*}
\]
correspond to linearized advective terms and are also similar. Given these similarities, we focus on the terms of \( M_\nu \) that correspond to the density update \( \delta \rho \) (i.e., \( D_\rho U_{\rho \nu} \delta \rho \), with \( D_\rho \delta \rho = \frac{\delta \rho}{\Delta t} + \theta \nabla \cdot (\rho_0 \delta \nu) - D_\nu \nabla \rho_0 \nabla \cdot \rho_0 \delta \nu \)) to infer a suitable \( M_\nu \). After some manipulation, one can derive exactly that
\[
D_\rho U_{\rho \nu} \delta \nu = \theta \nabla \cdot \left( \rho_0 \left( \frac{\delta \nu}{\Delta t} + \frac{\delta \rho}{\rho_0} \nabla \cdot (\rho_0 \delta \nu) - \frac{\partial}{\rho_0} \nabla \cdot (\rho_0 \delta \nu) \right) \right) = U_{\rho \nu} D_\nu \delta \nu,
\]
with \( D_\nu \delta \nu = \frac{\delta \nu}{\Delta t} + \theta \frac{\rho_0}{\rho_0} \nabla \cdot (\rho_0 \delta \nu) - \frac{\partial}{\rho_0} \nabla \cdot (\rho_0 \delta \nu) \). Inspired by \( D_\rho \), and for preconditioning purposes, we define \( M_\nu \) as
\[
M_\nu \delta \nu = \frac{\delta \nu}{\Delta t} + \theta \nabla \cdot \delta \nu.
\]
Notice we have kept only the flow term (which is anisotropic, and common to all variables) and neglected the diffusion term (which is isotropic, and depends on the variable of interest). This completes the formulation of the arbitrary-flow preconditioner.

4.2. Preconditioner for Hall MHD in the cold-ion limit

The XMHD model in equations 1-5 supports dispersive waves with dispersion relation \( \omega \sim k^2 \), such as the whistler and kinetic Alfvén waves. In explicit methods, the presence of dispersive waves results in explicit CFL time step limits \( \Delta t_{\text{CFL}} \sim \Delta x^2 \), substantially slowing their numerical integration. In implicit methods, dispersive waves result in poorly conditioned matrices, difficult to treat with iterative methods. In the context of Krylov methods, adequate treatment of
dispersive waves puts a premium on preconditioning. We now demonstrate a proof of principle for a preconditioner strategy for XMHD. For this, we focus on the cold ion limit, $T_e \gg T_i$. In this limit, $p \approx p_e$, and therefore the Ohm’s law (equation 5) reads:

$$E \approx -u \times B + \eta \nabla \times B + \frac{d_i}{\rho} (j \times B - \nabla p) = -u \times B + \eta \nabla \times B + \frac{d_i}{\rho} (\partial_t u + u \cdot \nabla u - \nu \nabla^2 u).$$ (11)

With this simplification in mind, the linearized XMHD model in equations 1–4 has the same coupling structure as the resistive MHD model presented earlier,

$$J \delta x \approx \begin{bmatrix} D_\rho & 0 & 0 & U_{up} \\ 0 & D_T & 0 & U_{uT} \\ L_{\rho u} & L_{Tu} & L_{Bu} & U_{ub} + U_{H} \\ L_{\rho u} & L_{Tu} & L_{Bu} & D_u \end{bmatrix} \begin{bmatrix} \delta \rho \\ \delta T \\ \delta u \\ \delta B \end{bmatrix},$$

but with a new contribution $U_{ub}$, coming from the modified Ohm’s law in equation 11. Consequently, the modified Schur complement $P_c$ contains a new term $L_{Bu}U_{ub} \approx \theta d_iB_0 \times (\nabla \times \nabla \times \delta u)$, which is the whistler wave propagator in the momentum equation in Hall MHD. We have analytical and numerical proof that such term can be smoothed effectively with damped Jacobi or Gauss-Seidel techniques and therefore can be treated effectively with multigrid methods.

5. Numerical results
In what follows, we present results for several test problems aimed at demonstrating some of the advertised properties of the solver. In particular, we employ a 2D magnetic Kelvin-Helmholtz instability (KHI) [14] to demonstrate its performance in the presence of large flows, a 2D tearing mode problem [4] to demonstrate its effectiveness in Hall MHD, and a 3D island coalescence problem [4] to demonstrate its excellent parallel scalability properties. As in [4], the explicit solver employed as a reference to calculate implicit CPU speedup is a second-order predictor-corrector method, which requires two function evaluations per time step. The explicit time step $\Delta t_{CFL}$ is calculated here as $\pi/(2\omega_{max})$, with $\omega_{max}$ the maximum local normal-mode frequency.

We employ multigrid V-cycles to approximate $M^{-1}$ and $P_c^{-1}$ where required. Restriction and prolongation employ second-order splines (of local processor data only). As a smoother, we employ a few passes of weighed Jacobi (p. 10 in [19]; p. 118 in [20]), with weight $\omega_{JB} = 0.7$, for both the restriction and the prolongation steps. In MG terminology, such V-cycle is identified as $V(m,n)$, where the two integers indicate restriction and prolongation smoothing steps, respectively.

5.1. Finite flow: Kelvin-Helmholtz instability
The KHI equilibrium is defined [14] by uniform density and pressure ($\rho_0 = T_0 = 1$), uniform magnetic field in the ignorable direction ($B_{z0} = B_{y0} = 0$, $B_{z0} = 1$), and a sheared velocity profile $v_{y0} = V_0 \tanh((x-0.5)/\lambda)$, with $V_0 = 0.5$ and $\lambda = 0.2$. The computation is performed in a 2D rectangular domain with $x \in [0, 1]$ and $y \in [0, 2.5]$, and with $D = 0$ and $\eta = \nu = 10^{-3}$, $\gamma = 5/3$. The boundary conditions are periodic in $y$ and perfect conductor ($B_x = 0$), no stress ($\partial_x v_y = 0$), and impenetrable wall ($v_x = 0$) in $x$. Homogeneous Neumann boundary conditions are imposed in $x$ for both $\rho$ and $T$.

The KHI is an ideal instability and therefore features $\gamma \sim O(1)$. This limits the attainable implicit time step in practice because, for a second-order accurate discretization, accuracy requires $\gamma \Delta t < 1/2$ [18]. However, implicit gains can still be obtained for sufficiently fine grids. Table 1 gives results of a grid convergence study with a fixed implicit time step $\Delta t = 0.1$. For this test, we use one $V(3,3)$ MG cycle. Performance results are averaged over ten time steps. The table shows good performance under grid refinement and moderate gains for sufficiently fine grids, despite the fact that flows are comparable to the Alfvén speed.
Table 1. Serial grid convergence study for the KHI problem with $\Delta t = 0.1$. Results are averaged over ten time steps. The table shows the average number of Newton and FGMRES iterations per time step, the total CPU time, the CPU speedup vs. and explicit code ($CPU_{exp}/CPU$, with $CPU_{exp}$ the CPU of the explicit approach), and the implicit time step size relative to the explicit CFL limit ($\Delta t/\Delta t_{CFL}$).

| $N$     | Newton/\$\Delta t$ | FGMRES/\$\Delta t$ | CPU (s) | $CPU_{exp}/CPU$ | $\Delta t/\Delta t_{CFL}$ |
|---------|---------------------|---------------------|---------|-----------------|-----------------------------|
| 32x32   | 4.0                 | 4.7                 | 29      | 0.5             | 17                          |
| 64x64   | 4.0                 | 5.0                 | 102     | 1.2             | 34                          |
| 128x128 | 4.2                 | 5.9                 | 434     | 2.3             | 74                          |
| 256x256 | 4.9                 | 9.9                 | 2652    | 3.3             | 156                         |

Table 2. Serial grid convergence study for Hall MHD using the tearing mode problem with $\Delta t = 1.0$. Results are taken for a single time step. Other quantities reported are defined as in table 1.

| $N$     | Newton/\$\Delta t$ | FGMRES/\$\Delta t$ | CPU (s) | $CPU_{exp}/CPU$ | $\Delta t/\Delta t_{CFL}$ |
|---------|---------------------|---------------------|---------|-----------------|-----------------------------|
| 32x32   | 5                   | 32                  | 21.6    | 0.9             | 110                         |
| 64x64   | 5                   | 9                   | 25.2    | 9.3             | 384                         |
| 128x128 | 5                   | 8                   | 91.5    | 37.9            | 1436                        |
| 256x256 | 4                   | 10                  | 467.0   | 117.0           | 5660                        |

5.2. Cold-ion Hall MHD: Tearing mode

The tearing mode equilibrium is defined by a uniform density and temperature, no flow, and a force-free magnetic Harris-sheet configuration given by [14, 4]:

$$B_{x0} = 0, B_{y0}(x) = \tanh[(x - 0.5)/\lambda], B_{z0} = \sqrt{1 - B_{y0}^2(x)},$$

with $\lambda = 0.2$. The domain is rectangular of dimensions with $x \in [0,1]$ and $y \in [0,4]$. The computation is performed with $\eta = 10^{-3}$, $\nu = 10^{-3}$, $D = 0$, and $\gamma = 5/3$. The boundary conditions are the same as for the KHI problem. We perturb the equilibrium with a sinusoidal perturbation in $\rho$, $\delta \rho = 10^{-3} \sin(2\pi y L_y) \cos(2\pi x L_x)$.

We demonstrate the performance of the Hall MHD preconditioner (using the small-flow approximation of [4]) with the tearing mode problem, using $\delta_i = 0.05$. Grid scaling results are presented in table 2. We employ classical MG V(3,3) cycles (with under-damped Jacobi as a smoother, and a linear tolerance of $10^{-2}$) to approximately invert $M$ and $P$. We have fixed the time step to $\Delta t = 1.0$, which spans anywhere from 110 explicit CFL time steps for the coarsest grid to 5660 for the finest. The table reports data for a single implicit time step. (Taking multiple time steps requires including a higher-order dissipation term in the Ohm’s law, equation (11), which is left for future work.) Notice from the table that the CPU speedup increases sharply as the grid is refined, reaching a factor of 117 for the finest grid considered.

5.3. Parallel performance: 3D resistive island coalescence

The island coalescence equilibrium is a modification of the Harris sheet equilibrium of the previous section. It features a uniform density, no flow, and magnetic field and pressure given by [4]:

$$\Psi(x,y) = -\lambda \log \left[ \cosh \left( \frac{x}{\lambda} \right) + \epsilon \cos \left( \frac{y}{\lambda} \right) \right],$$

$$p(x,y) = p_0 + \frac{(1 - \epsilon^2)}{2} \left[ \cosh \left( \frac{x}{\lambda} \right) + \epsilon \cos \left( \frac{y}{\lambda} \right) \right]^{-2},$$
Parallel performance results of the algorithm are given in figure 1. The left figure shows traces of the wall-clock CPU and the CPU per FGMRES iteration. The right figure shows the averaged number of FGMRES per time step (averaged over 10 fixed implicit time steps $\Delta t = 0.1$). These parallel performance data have been obtained in a weak sense, using a base resolution of $16^3$ grid points per processor. Such small base resolution is unfavorable from a parallel scaling standpoint because the surface-to-volume ratio (which is a measure of the percentage of points of the local domain that need to be communicated) is fairly large ($\sim 37\%$ for $N_p = 16^3$). Nevertheless, the algorithm performs well in parallel up to thousands of processors. From figure 1, left, it is clear that the scaling of the wall-clock CPU is optimal up to 1000 processors, showing some degradation after that. However, such degradation is not due to parallel bottlenecks in the algorithm: the CPU per FGMRES iteration does not increase, and communication costs remain small for all processor numbers considered. Rather, it is a loss of algorithmic performance, manifested in the growth of FGMRES iterations past the 1000-processor level (figure 1, right), which is the root cause of the increase. Such growth is attributed to the lack of a coarse-grid solver (our parallel MG implementation employs a fixed number of levels, determined by the size of the local processor problem), and to the fact that the problem is getting harder as the number of processors increases ($N$ scales linearly with the number of processors $n_p$, $N \propto n_p$, and the explicit CFL scales as $\Delta t_{CFL} \sim N^{-1/3}$; therefore, $\Delta t/\Delta t_{CFL} \sim N^{1/3} \sim n_p^{1/3}$).

6. Conclusions
In this paper, we have described a fully implicit, scalable approach for 3D compressible Hall MHD. The approach is based on Jacobian-free Newton-Krylov methods. Key to the approach is the availability of suitable preconditioners. In [4], a suitable preconditioner was proposed for 3D
compressible resistive MHD in the small-flow limit. Here, we have extended this development in two ways: we have generalized the small-flow preconditioner strategy to deal with arbitrary plasma flows, and we have extended the formulation to deal with Hall MHD in the cold-ion limit. The arbitrary-flow generalization is important to ensure robustness of the solver (as finite flows may develop in nonlinear regimes in simulations of interest, both locally and globally) and to enable direct-to-steady-state solution procedures. The cold-ion Hall MHD development, while still in a proof-of-principle stage, demonstrates that the approach holds promise to deal with the very challenging extended MHD formulation, which is of interest to many SciDAC applications. Moreover, we have presented parallel scalability results for a 3D resistive MHD problem using up to 4096 processors and 134 million unknowns, which demonstrate the potential of the approach to exploit massively parallel environments.

Acknowledgments
We acknowledge useful discussions with D. A. Knoll, J. N. Shadid, and R. Pawlowski. We thank the PETSc team for technical support. The work was funded by the Department of Energy at Los Alamos National Laboratory under contracts No. DE-AC52-06NA25396 and the DOE Office of ASCR program in Applied Mathematical Sciences. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References
[1] Colella P, Hou T and Petzold L 2004 Report of the first multiscale mathematics workshop: First steps toward a roadmap http://www.sc.doe.gov/ascr/Research/AM/MultiscaleMathWorkshop1.pdf
[2] Dolbrow J, Khaleel M A and Mitchell J 2004 Multiscale mathematics initiative: A roadmap Tech. Rep. PNNL-14966 Pacific Northwest National Laboratory http://www.sc.doe.gov/ascr/Research/AM/MultiscaleMathWorkshop3.pdf
[3] Oden J T, Belytschko T, Fish J, Hughes T J, Johnson C, Keyes D, Laub A, Petzold L, Srolovitz D and Yip S 2006 Simulation-based engineering science: Revolutionizing engineering science through simulation Tech. rep. National Science Foundation
[4] Chacón L 2008 Phys. Plasmas 15 056103
[5] Saad Y 1996 Iterative Methods for Sparse Linear Systems (Boston: PWS Publishing Company)
[6] Chan T F and Jackson K R 1984 SIAM J. Sci. Stat. Comput. 5 533–542
[7] Brown P N and Saad Y 1990 SIAM J. Sci. Stat. Comput. 11 430–481
[8] Saad Y and Schultz M 1986 SIAM J. Sci. Stat. Comput. 7 856–869
[9] Saad Y 1992 SIAM J. Sci. Comput. 14 461–469
[10] Balay S, Buschelman K, Eijkhout V, Gropp W D, Kaushik D, Knepley M G, McInnes L C, Smith B F and Zhang H 2004 PETSc users manual Tech. Rep. ANL-95/11 - Revision 2.1.5 Argonne National Laboratory
[11] Balay S, Eijkhout V, Gropp W D, McInnes L C and Smith B F 1997 Modern Software Tools in Scientific Computing ed Arge E, Bruaset A M and Langtangen H P (Birkhäuser Press) pp 163–202
[12] Balay S, Buschelman K, Gropp W D, Kaushik D, Knepley M G, McInnes L C, Smith B F and Zhang H 2001 PETSc: Web page http://www.mcs.anl.gov/petsc
[13] Chacón L and Knoll D A 2003 J. Comput. Phys. 188 573–592
[14] Chacón L 2004 Comput. Phys. Comm. 163 143–171
[15] Kelley C T 1995 Iterative methods for linear and nonlinear equations (Philadelphia: SIAM)
[16] Dembo R, Eisenstat S and Steihaug T 1982 J. Numer. Anal. 19 400
[17] Knoll D A and Keyes D E 2004 J. Comput. Phys. 193 357–97
[18] Chacón L, Knoll D A and Finn J M 2002 J. Comput. Phys. 178 15–36
[19] Briggs W L 1987 A Multigrid Tutorial (Philadelphia, PA: SIAM)
[20] Wesseling P 1992 An Introduction to Multigrid Methods (Chichester: John Wiley & Sons)
[21] Elman H C, Howle V E, Shadid J N and Tuminaro R S 2003 J. Comput. Phys. 185 504 – 23 ISSN 0021-9991