Journal of Computational Physics 227 (2008) 8855–8874

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Implicit adaptive mesh refinement for 2D reduced resistive magnetohydrodynamics

Bobby Philip [a,*], Luis Chacón [a], Michael Pernice [b]

[a] Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, United States
[b] Center for Advanced Modeling and Simulation, Idaho National Laboratory, Idaho Falls, ID 83415-2211, United States

ARTICLE INFO

Article history:
Received 18 December 2007
Received in revised form 16 June 2008
Accepted 26 June 2008
Available online 8 July 2008

Keywords:
Adaptive mesh refinement
Newton–Krylov
Implicit methods
Magnetohydrodynamics
Multilevel solvers

ABSTRACT

An implicit structured adaptive mesh refinement (SAMR) solver for 2D reduced magnetohydrodynamics (MHD) is described. The time-implicit discretization is able to step over fast normal modes, while the spatial adaptivity resolves thin, dynamically evolving features. A Jacobian-free Newton–Krylov method is used for the nonlinear solver engine. For preconditioning, we have extended the optimal "physics-based" approach developed in [L. Chacón, D.A. Knoll, J.M. Finn, An implicit, nonlinear reduced resistive MHD solver, J. Comput. Phys. 178 (2002) 15–36] (which employed multigrid solver technology in the preconditioner for scalability) to SAMR grids using the well-known Fast Adaptive Composite grid (FAC) method [S. McCormick, Multilevel Adaptive Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1989]. A grid convergence study demonstrates that the solver performance is independent of the number of grid levels and only depends on the finest resolution considered, and that it scales well with grid refinement. The study of error generation and propagation in our SAMR implementation demonstrates that high-order (cubic) interpolation during regridding, combined with a robustly damping second-order temporal scheme such as BDF2, is required to minimize impact of grid errors at coarse–fine interfaces on the overall error of the computation for this MHD application. We also demonstrate that our implementation features the desired property that the overall numerical error is dependent only on the finest resolution level considered, and not on the base-grid resolution or on the number of refinement levels present during the simulation. We demonstrate the effectiveness of the tool on several challenging problems.

1. Introduction

The magnetohydrodynamics (MHD) model is useful for studying the macroscopic behavior of fully ionized gases (plasmas). Plasmas exhibit a wide range of complex behavior, and are intrinsically multiscale both temporally and spatially. While MHD provides a tractable model for the macroscopic description of plasmas, it still presents formidable challenges for the numerical modeler. In particular, MHD (even in its simplest form) supports multiple time scales (which manifest in the form of waves) and multiple length scales (which manifest in the form of microscopic layer formation, often with macroscopic relevance).

Algorithmically, the multiscale nature of MHD needs to be addressed separately in time and space. Spatially, the dynamic formation of thin layers requires grid adaptation that can respond dynamically. While there are many options available for dynamic grid adaptation depending on the spatial representation of choice (e.g., *r*-refinement, *h*-refinement,

* Corresponding author.
E-mail addresses: bphilip@lanl.gov (B. Philip), chacon@lanl.gov (L. Chacón), michael.pernice@inl.gov (M. Pernice).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.06.029

p-refinement), our focus here is on h-refinement in the finite-volume context via structured adaptive mesh refinement (SAMR) [8,7]. A SAMR grid is organized as a hierarchy of nested refinement levels, with each level comprised of a union of rectangular patches. As the locally refined grid evolves to follow important features in the solution, these levels are created and destroyed as needed, and the solution is transferred from the old grid to the new grid to continue the simulation. While all approaches to adaptive mesh refinement have different strengths and weaknesses, we choose SAMR for a specific set of advantages it provides us. Firstly, SAMR employs uniform-grid stencils in large parts of the domain, thus providing higher accuracy than stencils with the same structure on non-uniform grids of comparable resolution [5]. Secondly, uniform-grid patches on all levels allow us to reuse single-grid smoothers and solvers developed for uniform-grid applications, such as geometric multigrid. This is important, since it is well known that exploiting geometric information when available results in better algorithmic performance [11,62]. Finally, upon regridding, SAMR only requires solver-setup operations in newly created patches, thus minimizing setup overhead. In this application, we employ the SAMR package SAMRAI [32] to handle grid and data management operations. SAMRAI is an obvious choice, since it provides interfaces [46] to state-of-the-art nonlinear solver packages such as PETSc [3].

Temporally, our interest is on applications where fast time scales are parasitic to a slower dynamical time scale of interest (such applications arise, for instance, in fusion [55,53] and space plasmas [28,49]). Accordingly, it is of interest to step over such fast time scales in order to resolve those of dynamical interest, while preserving the temporal accuracy of the approach. Explicit time-integration methods are subject to stability constraints that arise from the fastest time scales, and are inappropriate for this purpose because they force the modeler to follow the fastest time scale supported. Fully-implicit time integration methods allow stepping over fast time scales, since time steps are generally constrained only by accuracy, not stability [37]. However, they require the solution of large-scale systems of nonlinear equations at each time step, and fast, robust solution methods are necessary for implicit methods to be practical. Fortunately, Newton–Krylov methods [12] have provided such robust solvers in a variety of contexts [38], including MHD [16,15,14], provided effective preconditioning is used. In Refs. [16,15,14], the key for algorithmic performance was the use of multigrid methods in the preconditioner stage.

Patch-based refinement in the context of MHD has been explored by many previous studies in the literature, both in the context of finite-volumes (see e.g. [4,35,55,54,61,27]) and finite (and spectral) elements (e.g., [58,40,51]). In the finite-volume context, these studies have focused on various aspects of both the temporal and spatial discretization of the MHD equations on AMR grids. Spatially, authors have explored both staggered [4] and cell-centered [35,55,54] representations, with special emphasis on the preservation of conserved quantities and the solenoidal property of the magnetic field. An interesting study comparing the accuracy of finite-volumes/differences vs. spectral elements in an MHD-AMR context can be found in Ref. [45]. Temporally, most AMR implementations have relied on explicit methods, albeit with some flavor of time step subcycling for better performance (see e.g. [4,35]). However, a number of authors have explored more advanced time stepping algorithms, such as partially implicit [54] (where hyperbolic terms are treated explicitly, and diffusive terms implicitly), implicit/explicit [61] (where some blocks are treated explicitly, while others are treated with a linearly implicit method), and fully implicit [27] (albeit using unscalable direct solvers).

The focus of this study is to merge the SAMR dynamic adaptive-grid approach with efficient, scalable, fully-implicit time integration, in the context of MHD. For simplicity, we focus our attention on the 2D reduced resistive MHD model [59,20,30], which is rigorously valid in the presence of a large guide magnetic field. The reduced resistive MHD model has the advantage of simplicity while maintaining a truly multiscale character, both temporally (because it supports the fast Alfvén wave) and spatially (because it develops thin layers). Furthermore, mature fully-implicit technology is available [16], which will be reused for this study.

The advantages of fully-implicit SAMR are obvious, as it enables dynamic refinement while decoupling the time integrator from the small explicit time step stability limits (which scale with the mesh size) that would arise in the patches of finest resolution. Key to the effectiveness and scalability of the proposed approach is to generalize the multigrid treatment proposed in Refs. [16,15] to SAMR grids. This can be achieved with fast multilevel methods that exploit the structure of the mesh, such as the Fast Adaptive Composite grid (FAC) method [42], as has been already demonstrated in the context of 2D radiation diffusion [47].

Preliminary results on combining implicit time integration with SAMR for resistive MHD were first reported in [48]. Here we expand the study to include considerations of accuracy and provide details of our treatment of discretization at coarse–fine interfaces. Section 2 describes the mathematical model and its numerical discretization. Section 3 introduces the nonlinear solver of choice, Jacobian-free Newton–Krylov methods, and the preconditioning approach to make it efficient. The specifics of the coarse–fine interface treatment for this application are provided in Section 4. Finally, numerical results focusing on performance and accuracy aspects of the solver are presented in Section 5, and we conclude in Section 6.

2. Numerical model: Current–vorticity formulation of reduced MHD

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 in-plane magnetic field $\vec{B}_p$. As a result, $B_z \approx$ constant and the velocity $\vec{v}$ is nearly incompressible ($\nabla \cdot \vec{v} \approx 0$), and the general MHD formalism reduces to [59,20,30]:

$$\left(\partial_t + \vec{v}\cdot\nabla - \frac{\eta}{\mu_0}\Delta\right)\Psi + E_0 = 0, \tag{1}$$

$$\rho(\partial_t + \vec{v}\cdot\nabla - \nu\Delta)\mathscr{U} = \frac{1}{\mu_0}\vec{B}\cdot\nabla J, \tag{2}$$

$$\Delta\Phi = \mathscr{U}, \tag{3}$$

where Φ is the velocity stream function ($\vec{v} = \hat{z}\times\nabla\Phi$), $\mathscr{U}$ is the z-component of the fluid vorticity ($\mathscr{U} = \hat{z}\cdot\nabla\times\vec{v}$), Ψ is the flux function (which gives $\vec{B}_p = \hat{z}\times\nabla\Psi$), $\vec{B} = \vec{B}_p + B_z\hat{z}$ is the total magnetic field, $J = \Delta\Psi$ is the current, and ρ is the density (which is taken as constant). Note that, as defined, $\vec{B}$ is always solenoidal. The source E_0 (the applied electric field in the z-direction) has been included to balance the resistive decay of the equilibrium. The transport parameters (the kinematic viscosity ν and the resistivity η) are assumed constant. We note that $\vec{B}\cdot\nabla = \vec{B}_p\cdot\nabla$ since $\partial_z = 0$, but we keep $\vec{B}$ for the sake of generality.

Eqs. (1)–(3) are normalized as follows: $\vec{B}$ is normalized to the characteristic in-plane magnetic field B_0, ρ to the constant density ρ_0, lengths to an arbitrary length L, and the time to the Alfvén time $\tau_A = L/v_A$, where $v_A = B_0/\sqrt{\rho_0\mu_0}$ is the Alfvén speed. The normalized set of RMHD equations reads:

$$\begin{aligned}
&\partial_t\Psi + \vec{v}\cdot\nabla\Psi - \eta\Delta\Psi = -E_0,\\
&\partial_t\mathscr{U} + \vec{v}\cdot\nabla\mathscr{U} - \nu\Delta\mathscr{U} = \vec{B}\cdot\nabla J,\\
&\Delta\Phi = \mathscr{U},
\end{aligned} \tag{4}$$

where η is the normalized resistivity (the inverse of the Lundquist number) and ν is the normalized viscosity (the inverse of the Reynolds number). This set of equations is both elliptically and hyperbolically stiff. Elliptic stiffness originates from the streamfunction–vorticity coupling and the diffusion terms. Hyperbolic stiffness arises from the linear shear Alfvén wave, which for a homogeneous plasma is characterized by the dispersion relation $\omega = k_\parallel$, with ω the frequency, $k_\parallel = \vec{B}_0\cdot\vec{k}$, $\vec{k}$ the wavevector, and $|\vec{B}_0| = 1$.

While the form in (4) was successfully treated in [16], we have found that this formulation is not well suited for SAMR, due to difficulties discretizing the high-order term $\vec{B}\cdot\nabla J = \vec{B}\cdot\nabla(\Delta\Psi)$ at coarse–fine interfaces. Several discretization approaches were tried but all led to very high error accumulation along the coarse–fine interface that polluted the solution and eventually caused the simulations to fail. This is similar to difficulties reported in [58] for an ideal reduced MHD formulation. Instead, following Refs. [58,40], we use the current–vorticity formulation obtained by applying a Δ to the flux equation above, to obtain:

$$\begin{aligned}
&\partial_t J + \vec{v}\cdot\nabla J - \vec{B}\cdot\nabla\mathscr{U} - \{\Phi,\Psi\} - \eta\Delta J = -\Delta E_0,\\
&\partial_t\mathscr{U} + \vec{v}\cdot\nabla\mathscr{U} - \nu\Delta\mathscr{U} = \vec{B}\cdot\nabla J,\\
&\Delta\Psi = J,\\
&\Delta\Phi = \mathscr{U},
\end{aligned} \tag{5}$$

where $\{\Phi,\Psi\} = 2[\Phi_{xy}(\Psi_{xx} - \Psi_{yy}) - \Psi_{xy}(\Phi_{xx} - \Phi_{yy})]$. This formulation is advantageous because it avoids derivatives that are higher than second-order, and all dependent variables are determined from integration rather than differentiation. However, it features two elliptic constraints instead of one in (4). Furthermore, its implementation requires modifications to the semi-implicit preconditioner developed in [16] for (4), which will be discussed later in this paper (Section 3.2). Equation set (5) needs to be complemented with boundary conditions, which are to be imposed on the dependent variables J, $\mathscr{U}$, Φ, and Ψ. For the examples considered in this study (Section 5), we employ periodic boundary conditions in one direction. In the other direction, we follow [16] and employ Dirichlet boundary conditions for all variables. For flow quantities, we enforce impenetrable-wall $\Phi = 0$ (i.e., the normal component of the velocity vanishes) and no-stress $\mathscr{U} = 0$ (i.e., normal derivative of tangential velocity components vanishes) boundary conditions. For magnetic variables, we enforce perfect-conductor boundary conditions by setting $J = 0$ and Ψ to a constant (determined by the equilibrium). It is easy to show that, except for small diffusive terms, these boundary conditions are consistent with the $\mathscr{U}$ equation and the Ψ equation in (4) evaluated at the boundary, and therefore are also appropriate for (5).

For the temporal discretization of (5), we will explore two approaches: a θ-scheme ($\theta = 0.5$ is second-order accurate, and corresponds to the Crank–Nicolson scheme [17]), and a second-order backward differentiation formula (BDF2) [18,26]. The θ-scheme reads:

$$\begin{aligned}
&\frac{(J^{n+1} - J^n)}{\Delta t} + [\vec{v}\cdot\nabla J]^{n+\theta} - \eta\Delta J^{n+\theta} = [\vec{B}\cdot\nabla\mathscr{U}]^{n+\theta} + \{\Phi,\Psi\}^{n+\theta},\\
&\frac{(\mathscr{U}^{n+1} - \mathscr{U}^n)}{\Delta t} + [\vec{v}\cdot\nabla\mathscr{U}]^{n+\theta} - \nu\Delta\mathscr{U}^{n+\theta} = [\vec{B}\cdot\nabla J]^{n+\theta},\\
&\Delta\Psi^{n+\theta} = J^{n+\theta},\\
&\Delta\Phi^{n+\theta} = \mathscr{U}^{n+\theta},
\end{aligned} \tag{6}$$

where $n+\theta$ quantities are calculated as $\xi^{n+\theta} = (1-\theta)\xi^n + \theta\xi^{n+1}$. The BDF2 scheme discretizes the temporal derivative terms by fitting a quadratic polynomial using the $n+1$, n, and $n-1$ time levels, and approximating the time derivative at time

level $n+1$ by the derivative of this polynomial. For a constant time step, this results in the following approximation of $\partial_t J$ at $n+1$:

$$\partial_t J|_{n+1} \approx \frac{\frac{3}{2}J^{n+1} - 2J^n + \frac{1}{2}J^{n-1}}{\Delta t}$$

with a similar expression for $\partial_t \mathscr{U}$. With BDF2, the remaining terms of (6) are evaluated at the new time level $n+1$ (i.e., with $\theta = 1$). Unlike Crank–Nicolson, BDF2 features robust damping of dissipative terms [26]. The purpose of considering these two approaches is to compare their error propagation properties in the presence of coarse–fine interfaces (Section 5).

Within AMR patches, spatial operators in (6) are discretized using second-order centered finite differences. Following [16], advective terms are discretized in non-conservative form, using centered differences. Boundary conditions (either at physical boundaries or at coarse–fine interfaces) are imposed using ghost cells. The spatial treatment of coarse–fine interfaces employed in this application is described in detail later in this paper (Section 4).

3. Nonlinear solution algorithm

Our general approach to the solution of (6) is via preconditioned Jacobian-free Newton–Krylov methods (JFNK). These methods have demonstrated their effectiveness in many similar applications [38], including 2D reduced resistive MHD [16] and Hall MHD [15], and 3D resistive MHD [14]. Our approach generalizes that of [16] in two fundamental ways: firstly, we have adapted the preconditioning strategy to deal with the $J - \mathscr{U}$ formulation instead of the $\Psi - \mathscr{U}$ formulation; and secondly, we have generalized the single-mesh multigrid treatment advocated in that reference to AMR meshes using the FAC method [42]. In what follows, we summarize the JFNK philosophy and our approach to preconditioning. The next section will deal with the AMR aspects of this application.

3.1. Jacobian-free Newton–Krylov methods

Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a nonlinear function and consider calculating the solution $x^\star \in \mathbb{R}^n$ of the system of nonlinear equations

$$F(x^\star) = 0. \tag{7}$$

Classical Newton's method for solving (7) generates a sequence of approximations x_k to $x^\star$, where $x_{k+1} = x_k + s_k$ and the Newton step s_k is the solution to the system of linear equations

$$F'(x_k)s_k = -F(x_k), \tag{8}$$

where F' is the Jacobian of F evaluated at x_k. Newton's method is attractive because of its fast local convergence properties, but for large-scale problems, it is impractical to solve (8) with a direct method. Furthermore, it is often unnecessary to solve (8) using a tight convergence tolerance when x_k is far from $x^\star$, since the linearization that leads to (8) may be a poor approximation to $F(x)$. Generally, it is much more efficient to employ so-called inexact Newton methods [19], in which the linear tolerance for (8) is selected adaptively by requiring that s_k only satisfy:

$$\|F(x_k) + F'(x_k)s_k\| \leqslant \eta_k \|F(x_k)\| \tag{9}$$

for some $\eta_k \in (0,1)$ [19]. When the forcing term η_k is chosen appropriately, superlinear and even quadratic convergence of the iteration can be achieved [21].

While any iterative method can be used to find an s_k that satisfies (9), Krylov subspace methods are distinguished by the fact that they require only matrix–vector products to proceed. These matrix–vector products can be approximated by a finite-difference version of the directional (Gâteaux) derivative as:

$$F'(x_k)v \approx \frac{F(x_k + \varepsilon v) - F(x_k)}{\varepsilon}, \tag{10}$$

which is especially advantageous when F' is difficult to compute or expensive to store (as is the case in this application due to the presence of multiple grid patches). While the selection of a suitable differencing parameter ε may be non-trivial for some applications, it is generally well understood [34]. For this application, we choose:

$$\varepsilon = \sqrt{\epsilon_{\text{mach}}} \frac{\sqrt{1 + \|x_k\|}}{\|v\|},$$

where ϵ_{mach} is machine precision and $\|\cdot\|$ refers to the l_2-norm. In our applications, which are performed in double precision, ε is typically on the order of 10^{-10}.

Among the various Krylov methods available, GMRES is selected because it guarantees convergence with non-symmetric, non-positive definite systems [52] (the case here because of flow and wave propagation), and because it provides normalized Krylov vectors $\|v\| = 1$, thus bounding the error introduced in the difference approximation of (10) (whose leading error term is proportional to $\varepsilon\|v\|^2$) [44]. 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 [39]. Here, we focus on minimizing the number of GMRES iterations per Jacobian solve for efficiency and robustness by: (1) using inexact Newton techniques (as described above) and (2) improving the condition number of the Jacobian matrix by preconditioning the problem. The next section describes our approach to preconditioning.

3.2. Preconditioning

Implicit time differencing eliminates time step stability constraints stemming from Alfvén waves, advective terms, and diffusion operators, allowing us to select time steps independent of the level of mesh refinement. 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 in handling.

As is explained in [16], there are two sources of ill-conditioning in the system of reduced MHD equations: elliptic operators and hyperbolic couplings. The former can be dealt with effectively with multilevel techniques. The latter, however, cannot be unless the hyperbolic couplings are reformulated in a multilevel-friendly fashion. Refs. [16,15] provide a systematic way of doing this, which we follow here.

3.2.1. Approximate formulation of the reduced MHD system

Krylov techniques are employed here to approximately solve (8) to the dynamically selected tolerance (9) in each Newton step. Hence, the construction of the physics-based preconditioner necessarily starts from the linearized system of equations. For the system in (5), the linearized equations read (in block form):

$$\begin{pmatrix} \mathscr{L}_\eta & -\theta\vec{B}_0\cdot\nabla & U_{J,\Psi} & U_{J,\Phi} \\ -\theta\vec{B}_0\cdot\nabla & \mathscr{L}_\nu & U_{\mathscr{U},\Psi} & U_{\mathscr{U},\Phi} \\ \mathbb{I} & 0 & -\Delta & 0 \\ 0 & \mathbb{I} & 0 & -\Delta \end{pmatrix} \begin{pmatrix} \delta J \\ \delta\mathscr{U} \\ \delta\Psi \\ \delta\Phi \end{pmatrix} = \begin{pmatrix} r_J \\ r_{\mathscr{U}} \\ r_\Psi \\ r_\Phi \end{pmatrix}, \tag{11}$$

where the diagonal blocks $\mathscr{L}_\eta$ and $\mathscr{L}_\nu$ read

$$\mathscr{L}_\eta = \frac{\mathbb{I}}{\Delta t} + \theta(\vec{u}_0\cdot\nabla - \eta\Delta), \quad \mathscr{L}_\nu = \frac{\mathbb{I}}{\Delta t} + \theta(\vec{\mathbf{u}}_0\cdot\nabla - \nu\Delta)$$

and the off-diagonal entries $U_{J,\Psi}, U_{J,\Phi}, U_{\mathscr{U},\Psi}$ and $U_{\mathscr{U},\Phi}$ are given by:

$$U_{J,\Psi} = -\theta(\nabla\mathscr{U}_0\cdot\hat{z}\times\nabla + \{\Phi_0,\cdot\}), \quad U_{J,\Phi} = \theta(\nabla J_0\cdot\hat{z}\times\nabla - \{\cdot,\Psi_0\}),$$

and

$$U_{\mathscr{U},\Psi} = -\theta\nabla J_0\cdot\hat{z}\times\nabla, \quad U_{\mathscr{U},\Phi} = \theta\nabla\mathscr{U}_0\cdot\hat{z}\times\nabla.$$

In order to formulate an approximate, multilevel-friendly form of the linearized set, we follow [16] to reduce the order of the $\delta J, \delta\mathscr{U}$ equations by factoring out a Laplacian operator (thus rendering equations for $\delta\Psi, \delta\Phi$, respectively). This is done by first eliminating δJ and $\delta\mathscr{U}$ from the corresponding equations in favor of $\delta\Psi$ and $\delta\Phi$ (using the linearized elliptic constraints). In the δJ equation, one can factor out the Laplacian operator trivially (since the Laplacian is a linear operator, and the J equation was obtained by applying a Laplacian operator onto the Ψ equation in the first place). In the $\delta\mathscr{U}$ equation, the Laplacian operator can be factored out approximately in the same fashion as was shown in [16]. After these transformations, we end up with the following approximate system:

$$\mathscr{P}\begin{pmatrix} \delta\Psi \\ \delta\Phi \end{pmatrix} \approx \mathbf{\Delta}^{-1}\left[\begin{pmatrix} r_J \\ r_{\mathscr{U}} \end{pmatrix} - \mathscr{P}\begin{pmatrix} r_\Psi \\ r_\Phi \end{pmatrix}\right],$$

where

$$\mathscr{P} \equiv \begin{pmatrix} \mathscr{L}_\eta & -\theta\vec{B}_0\cdot\nabla \\ -\theta\vec{B}_0\cdot\nabla & \mathscr{L}_\nu \end{pmatrix}$$

is the same hyperbolic operator found in [16]. We note that the factorization of the Laplacian operators enables us to solve for $\delta\Psi$ and $\delta\Phi$ directly in the preconditioner, because no high-order differential operators of the original flux–vorticity formulation remain. After solving for $\delta\Psi$ and $\delta\Phi$, one can recover δJ and $\delta\mathscr{U}$ by solving:

$$\mathscr{P}\begin{pmatrix} \delta J \\ \delta\mathscr{U} \end{pmatrix} = \begin{pmatrix} r_J - \theta(\delta\vec{u}\cdot\nabla J_0 - \delta\vec{B}\cdot\nabla\mathscr{U}_0 - \{\delta\Phi,\Psi_0\} - \{\Phi_0,\delta\Psi\}) \\ r_{\mathscr{U}} - \theta(\delta\vec{u}\cdot\nabla\mathscr{U}_0 - \delta\vec{B}\cdot\nabla J_0) \end{pmatrix},$$

which again requires inverting $\mathscr{P}$.

Following [16], systems of equations $\mathscr{P}\vec{\mathbf{v}} = \vec{\mathbf{b}}$ are solved with a few iterations (2 in this paper) of the stationary method obtained from the splitting

$$\mathscr{P} = \underbrace{\begin{pmatrix} \mathscr{L}_\eta & -\theta\vec{B}_{\mathbf{0}}\cdot\nabla \\ -\theta\vec{B}_0\cdot\nabla & \mathscr{D}_v \end{pmatrix}}_{\mathscr{M}} - \begin{pmatrix} 0 & 0 \\ 0 & \mathscr{D}_v - \mathscr{L}_v \end{pmatrix}$$

with $\mathscr{D}_v = \mathrm{diag}(\mathscr{L}_v)$ the diagonal of the advection diffusion operator $\mathscr{L}_v$. This splitting results in the iteration:

$$\vec{\mathbf{v}}^{k+1} = \vec{\mathbf{v}}^k + \mathscr{M}^{-1}(\vec{b} - \mathscr{P}\vec{\mathbf{v}}^k).$$

The inversion of $\mathscr{M}$ involves first a block factorization:

$$\mathscr{M} = \begin{pmatrix} \mathbb{I} & -\theta(\vec{B}_0\cdot\nabla)\mathscr{D}_v^{-1} \\ 0 & \mathbb{I} \end{pmatrix}\begin{pmatrix} P_{\mathrm{SI}} & 0 \\ 0 & \mathscr{D}_v \end{pmatrix}\begin{pmatrix} \mathbb{I} & 0 \\ \theta\mathscr{D}_v^{-1}(\vec{B}_{\mathbf{0}}\cdot\nabla) & \mathbb{I} \end{pmatrix},$$

with $P_{\mathrm{SI}} = \mathscr{L}_\eta - \theta^2\nabla\cdot(\vec{B}_0\mathscr{D}_v^{-1}\vec{B}_0^{\mathrm{T}}\nabla)$, and then the inversion of the resulting matrices, yielding:

$$\mathscr{M}^{-1} = \begin{pmatrix} \mathbb{I} & 0 \\ -\theta\mathscr{D}_v^{-1}(\vec{B}_0\cdot\nabla) & \mathbb{I} \end{pmatrix}\begin{pmatrix} P_{\mathrm{SI}}^{-1} & 0 \\ 0 & \mathscr{D}_v^{-1} \end{pmatrix}\begin{pmatrix} \mathbb{I} & \theta\vec{B}_{\mathbf{0}}\cdot\nabla\mathscr{D}_v^{-1} \\ 0 & \mathbb{I} \end{pmatrix}.$$

The implementation of $\mathscr{M}^{-1}$ only requires the (trivial) inversion of $\mathscr{D}_v$ (which is a diagonal matrix), and the inversion of the semi-implicit operator P_{SI}. The latter is a parabolic operator, amenable to multilevel techniques, as described in [16].

4. Adaptive mesh refinement

The previous discussion has considered the generalization of the physics-based preconditioner proposed in Ref. [16] to the application at hand, without regard to the specifics of the spatial discretization employed. In what follows, we describe the AMR-specific details of our treatment of the MHD equations, with particular emphasis on (1) the spatial discretization at coarse–fine interfaces, (2) the generalization of multilevel solvers for SAMR grids, and (3) regridding and its impact on time integration.

4.1. Structured AMR grids

Let $\Omega = [x_{lo}, x_{hi}] \times [y_{lo}, y_{hi}]$ be a rectangular computational domain. We create a discrete computational domain by subdividing $[x_{lo}, x_{hi}]$ into n_x subintervals with centers $x_i = x_{lo} + (i+\frac{1}{2})h_x$ with $h_x = (x_{hi} - x_{lo})/n_x$ for $i = 0, \ldots, n_x - 1$. Each subinterval has faces located at $x_{i-\frac{1}{2}} = x_i - h_x/2$ and $x_{i+\frac{1}{2}} = x_i + h_x/2$. Likewise $[y_{lo}, y_{hi}]$ is partitioned into n_y subintervals with centers $y_j = y_{lo} + (j+\frac{1}{2})h_y$ with $h_y = (y_{hi} - y_{lo})/n_y$ for $j = 0, \ldots, n_y - 1$ and faces $y_{j-\frac{1}{2}} = y_j - h_y/2$ and $y_{j+\frac{1}{2}} = y_j + h_y/2$. The tensor product of these subintervals partitions Ω into a collection of computational cells $\Omega^h = \{\Omega_{i,j}\}$ each with size $h_x \times h_y$ centered at coordinates (x_i, y_j). These ideas are readily extended to the case where Ω is a union of non-overlapping rectangular regions, and we continue to use the same notation Ω^h to denote such a collection of computational cells. Such *regular grids* are in widespread use in computational science and engineering, and a great deal of high quality software that is tuned to regular grids, such as geometric multigrid methods, is available.

Let $K \geqslant 1$ and $\Omega_1 \equiv \Omega \subset \Omega_2 \subset \cdots \Omega_K$ be a nested set of subdomains of the computational domain Ω. For simplicity, assume that each $\Omega_\ell, 2 \leqslant \ell \leqslant K$ is a union of non-overlapping rectangular regions; these are the subregions of Ω where additional resolution is desired. A composite structured AMR (SAMR) grid Ω^c on Ω is a nested hierarchy of grids $\Omega_1^{h_1} \subset \Omega_2^{h_2} \subset \cdots \subset \Omega_K^{h_K}$ consisting of K levels, with mesh spacing $h_1 > h_2 > \cdots > h_K$, with the coarsest grid $\Omega_1^{h_1}$ covering Ω. Each level $\Omega_\ell^{h_\ell}$ consists of a union of non-overlapping rectangular regions, or *patches*, at the same resolution h_ℓ. When there is no risk of confusion we will drop the h_ℓ superscript and simply refer to Ω^ℓ. This hierarchical representation allows operations on Ω^c to be implemented as operations on individual levels Ω^ℓ, which in turn are decomposed into operations on individual rectangular patches. This property facilitates reuse of software written for regular grids. Fig. 1 shows a SAMR grid with $K = 3$ and two patches on each of the two refinement levels. Note that while each level is nested in the next coarser level, there is no requirement that a patch at one refinement level is nested fully in a patch at another refinement level, i.e., a fine patch at refinement level l may lie over one or more coarser patches at refinement level $(l-1)$. Fig. 2 shows the decomposition of a fairly complex SAMR grid with six refinement levels into its constituent refinement levels and patches, as is encountered in our simulations.

4.2. Function evaluation on SAMR grids

For a viable JFNK solver, a given application only needs to provide methods to evaluate F, set up a preconditioner, and apply the preconditioner. By generalizing these steps to a SAMR grid hierarchy, JFNK can be readily adapted to SAMR applications. In this work, we use the PETSc parallel implementation of JFNK [3], which is made SAMR-aware via the PETSc-SAMRAI interfaces described in [46]. Considerations for evaluating F are described next.

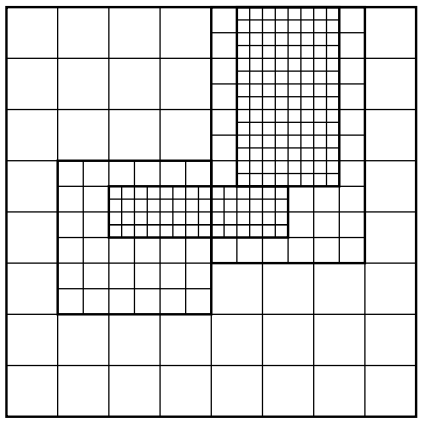

Fig. 1. Example of a multilevel SAMR grid with three levels.

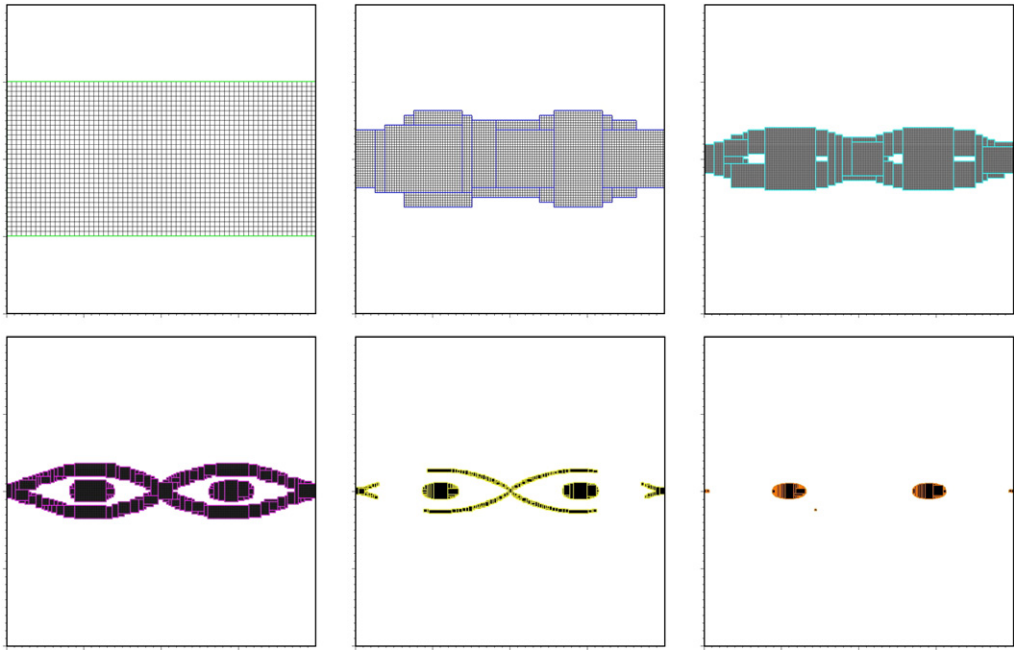

Fig. 2. The decomposition of a SAMR grid into its constituent refinement levels.

4.2.1. Single-grid discretization

In order to discretize F [given by (6)] in space, a cell-centered collocated finite-volume scheme is used for $\mathscr{U}, J, \Phi$, and Ψ. The magnetic field, $\vec{B} = (B^1, B^2)^\top$, and the velocity, $\vec{v} = (u^1, u^2)^\top$, are also stored at cell centers, and are computed using centered differences from Ψ and Φ, respectively, using the discrete curl operations:

$$\begin{aligned} B^1_{i,j} &= -\frac{\Psi_{i,j+1} - \Psi_{i,j-1}}{2h_y}, \\ B^2_{i,j} &= \frac{\Psi_{i+1,j} - \Psi_{i-1,j}}{2h_x}, \end{aligned} \tag{12}$$

and

$$\begin{aligned} u^1_{i,j} &= -\frac{\Phi_{i,j+1} - \Phi_{i,j-1}}{2h_y}, \\ u^2_{i,j} &= \frac{\Phi_{i+1,j} - \Phi_{i-1,j}}{2h_x}. \end{aligned} \tag{13}$$

We note that this discretization ensures that the divergence-free conditions on $\vec{B}$ and $\vec{v}$ are satisfied locally to numerical round-off.

Diffusive operators are discretized in each cell (i,j) by first computing approximate face-centered diffusive fluxes and then summing over the faces of each cell resulting in the standard five-point finite-volume discretization:

$$\frac{1}{h_x h_y}\int_{\Omega_{ij}} \nabla\cdot\nabla\Psi \,\mathrm{d}A \approx \frac{\Psi_{i+1,j}-\Psi_{ij}}{h_x^2} - \frac{\Psi_{ij}-\Psi_{i-1,j}}{h_x^2} + \frac{\Psi_{i,j+1}-\Psi_{ij}}{h_y^2} - \frac{\Psi_{ij}-\Psi_{i,j-1}}{h_y^2}. \tag{14}$$

Advective-like quantities (such as $\vec{v}\cdot\nabla$ and $\vec{B}\cdot\nabla$) are discretized using non-conservative centered differences. Such a discretization is appropriate in the reduced MHD context because the velocity is solenoidal, and hence no shock phenomena are supported (thus justifying the use of centered differences). Furthermore, for solenoidal fields, the nonlinearly robust, finite-volume conservative discretization for advective terms in cell-centered representations, ZIP [31,13], trivially results in the non-conservative form. So, in single grids, there is no inconsistency between using the non-conservative advective form and our generic finite-volume treatment. The situation is different, however, at coarse–fine interfaces, and this is discussed in the next section.

For non-conservative advective quantities, we employ the standard cell-centered difference discretization for the gradient operator, which can be derived by using a variant of the Gauss theorem for gradients on each cell:

$$\int_{\Omega_{ij}} \nabla\Psi \,\mathrm{d}A = \int_{\partial\Omega_{ij}} \Psi\vec{n}\,\mathrm{d}s$$

where $\vec{n}$ is the unit outward-facing normal on $\partial\Omega_{ij}$, and approximating the face centered values by averaging from cell centers. In a single grid, this results in the well-known formula:

$$\frac{1}{h_x h_y}\int_{\Omega_{ij}} \nabla\Psi \,\mathrm{d}A = \frac{\Psi_{i+1,j}-\Psi_{i-1,j}}{2h_x}\vec{i} + \frac{\Psi_{i,j+1}-\Psi_{i,j-1}}{2h_y}\vec{j}. \tag{15}$$

4.2.2. Extension to SAMR grids

The discretizations described in the previous subsection are valid in the interiors of individual patches as well as at the boundaries between two patches in the same refinement level. However, in order to maintain accuracy, changes are required at the boundaries between coarse and fine patches. Fig. 3 (left) shows the interface between a coarse and fine patch. We use ghost cells (both coarse and fine) for communication at coarse–fine interfaces, as well as between patches in the same refinement level. A fine ghost cell (Fig. 3, center) overlaps one coarse cell. A coarse ghost cell (Fig. 3, right), however, lies underneath four fine cells when a refinement ratio of 2 is used.

For computations of fine ghost cell values at coarse–fine interfaces, data is quadratically interpolated from a combination of coarse and fine grid cell data. Fig. 4 (left) shows the coarse grid cells that would be involved in performing quadratic tangential interpolation of coarse-grid data to align it with fine-grid data. Fig. 4 (right) shows the piecewise quadratic normal interpolation involving fine cells to calculate the fine ghost cell value. Once data has been interpolated to fine ghost cells, fine cells at coarse–fine interfaces can be treated identically to cells that lie in the interior of the fine patch. Fig. 4 only shows the simplest case where a sufficient number of coarse cells (in this case, three), are available to do standard quadratic interpolation tangential to the interface. In general, for block-structured AMR, many special cases need to be accounted for, where two or more fine patches may be adjacent to each other, resulting in very irregular coarse–fine interfaces (see for example Fig. 2). We do not detail the adjustments needed in each of these cases to interpolate data quadratically, due to space limitations. However, [2] provides a glimpse of the types of adjustments required.

Regarding coarse ghost cells at coarse–fine interfaces, their treatment is done as follows:

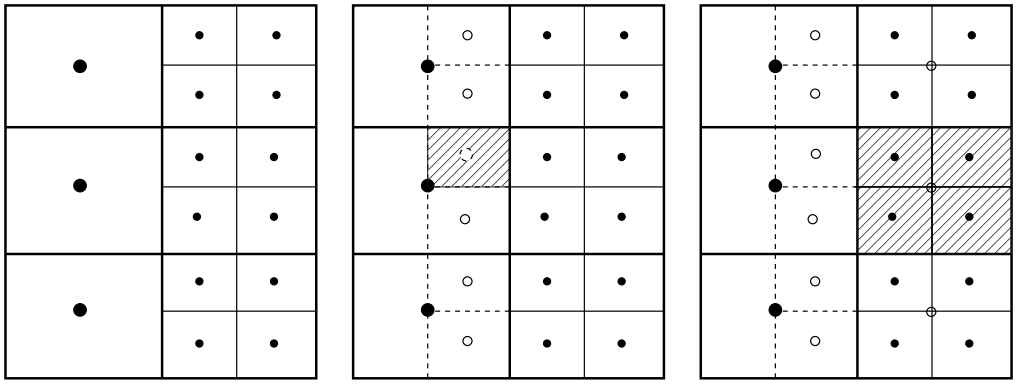

Fig. 3. (Left) A coarse–fine interface. (Center) Fine ghost cell. (Right) Coarse ghost cell.

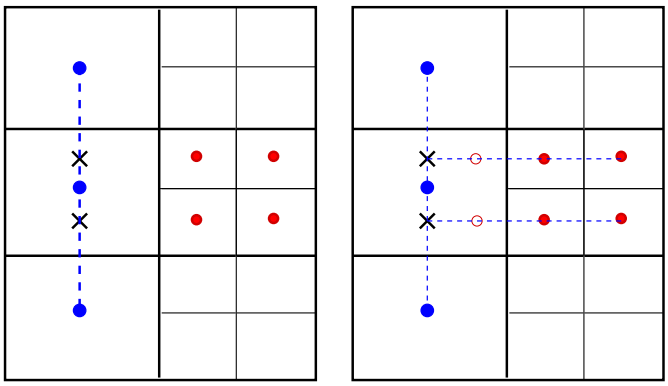

Fig. 4. Schematic of interpolation. (Left) Crosses show coarse grid data aligned with fine grid data by interpolation. (Right) Open circles denote fine ghost-cell data obtained by interpolation from aligned coarse data and fine grid data.

4.2.2.1. Magnetic and velocity fields. $\vec{B}$ and $\vec{v}$ are computed from Ψ and Φ using centered differences. While this maintains the vector fields divergence-free everywhere, it requires values of Ψ and Φ to be available at the coarse ghost cells. A simple arithmetic averaging of fine Ψ and Φ data to coarse ghost cells produces second-order-accurate coarse values, and therefore the resulting $\vec{B}$ and $\vec{v}$ are only first-order accurate. Local first-order accuracy at coarse–fine interfaces is unlikely to affect the global second-order accuracy of the calculation when the number of coarse–fine interface cells is small relative to the total number of cells. However, Ref. [1] gives examples of cases (not unlike the situations present in our simulations) where the number of coarse–fine interface cells can be up to 30% of the total number of cells. In such cases, global accuracy will be affected by the local first-order accuracy at coarse–fine interfaces. To avoid this situation, we use piecewise tensor-product cubic interpolation of fine-grid data for interpolating Ψ, Φ to coarse ghost cells. This, in turn, results in second-order discretizations for the vectors at coarse cells adjacent to the coarse–fine interface.

4.2.2.2. Diffusion operators. To compute the numerical diffusive fluxes at coarse cells adjacent to the coarse–fine interface, it is possible to use coarse ghost cell values underlying the fine grid. However, for flux conservation, we compute the diffusive fluxes at fine cell faces (as described in Section 4.2.1) and average them down to provide the coarse flux at the coarse face.

4.2.2.3. Advection operators. Advection operators at coarse–fine interfaces also employ the single-grid non-conservative form used within patches (Section 4.2.1). This form is advantageous because it avoids finding vector quantities at coarse–fine interfaces, and results in a more robust SAMR implementation. However, within a finite-volume context, it entails an implicit assumption that the solenoidal vector fields are constant on coarse–fine boundary cells. We have not found this assumption to affect the accuracy of our simulations appreciably. For the discretization of the non-conservative advection operators at coarse-patch boundary cells, we proceed as indicated in Section 4.2.1 and find coarse face-averaged values of the advected quantity. Here, this is done by averaging fine face-averaged data [29]. (The alternative is to compute a coarse face-averaged value using coarse ghost cell data, but this does not work as well.)

4.3. Regridding

For fully-dynamic AMR simulations, the grid hierarchy will change during the simulation as the solution evolves in time and space. This involves the addition of fine patches in regions where additional resolution is required and the removal of patches in regions where coarser resolution is sufficient. The regridding process involves using some refinement indicator to determine required grid resolution, and then constructing a new grid hierarchy based on this information. The ideal refinement indicator is a sharp estimate of the spatial error that is inexpensive to compute. When such an estimate is not available, refinement indicators that detect features in the solution, such as regions of large gradients or curvature, are used. We detail the refinement indicators we employ in this study in Section 5.

Once regridding is done, data needs to be transferred from the old grid hierarchy to the new one. Typically, the solution obtained on the old grid hierarchy is no longer a solution on the new grid hierarchy. This has been observed in our simulations as well as reported by others [10]. Two approaches are common in the literature. One approach, referred to as the "warm restart" [9], continues the time integration on the new grid hierarchy (with a time step comparable to that before regridding) using the interpolated data on the new grid hierarchy. The second approach, called a "cold restart" [9], alters the time step and possibly the time integration scheme to account for regridding and introduction of spatial errors.

In this study, we use a third approach. Cubic interpolation is used to transfer data from the old grid hierarchy to the new hierarchy. This is done for both newly refined and de-refined regions. As we mention in the previous paragraph, the interpolation of the data from the old to the new grid hierarchy typically introduces interpolation errors. These

are reflected in a nonlinear function residual that no longer satisfies the user-specified nonlinear solver tolerances. To correct for this, after interpolation of the required vectors (including previous time-step solutions, required for the time integration scheme), we re-solve for the current time step solution on the new grid hierarchy, using the solution interpolated from the old grid hierarchy as an initial guess. This synchronizes the current time step solution with the new grid hierarchy before advancing in time. Numerically, this procedure is robust, and avoids propagation of interpolation errors during regridding steps. We have confirmed the benefits of this approach by comparing time-evolving SAMR solutions against uniform-grid ones. A detailed comparative evaluation of all three regridding approaches is left for future work.

4.4. Preconditioning and the fast adaptive composite grid method

Physics-based preconditioning, as described in Section 3.2, requires the inversion of the parabolic operator $P_{\mathrm{SI}} = \mathscr{L}_\eta - \theta^2 \nabla \cdot (\vec{B}_0 \mathscr{D}_v^{-1} \vec{B}_0^{\mathrm{T}} \nabla)$. The operator $\nabla \cdot (\vec{B}_0 \mathscr{D}_v^{-1} \vec{B}_0^{\mathrm{T}} \nabla)$ is negative definite and self-adjoint. The discretization of this operator follows [57] so that it is compact (i.e., on a nine-point stencil), and the resulting matrix is symmetric negative definite. We note that, at coarse–fine interfaces, symmetry is lost. The convective operator in P_{SI} is discretized using a first-order upwind scheme (instead of the centered differences employed to evaluate the nonlinear residual) for robustness of the FAC smoothing step (see below). This leads to a better conditioned operator, which increases the robustness of the preconditioner but does not affect the accuracy of the solution [which is determined by (7)].

On a SAMR grid, the inversion of P_{SI} is performed efficiently by the FAC method [42,43]. FAC extends techniques from multigrid on uniform grids to AMR grids. FAC solves problems on AMR grids by combining smoothing on refinement levels with a coarse-grid solve using an approximate solver, such as a V-cycle of multigrid. First we introduce some notation to describe the FAC algorithm:

- $I_c^\ell : \Omega^c \to \Omega_\ell$ and $I_\ell^c : \Omega_\ell \to \Omega^c$, respectively denote restriction and interpolation operators between the composite SAMR grid, Ω^c, and an individual refinement level. Here, we use bilinear interpolation for I_ℓ^c and simple averaging for I_c^ℓ.
- $I_{\ell+1}^\ell : \Omega_\ell \to \Omega_{\ell+1}$ and $I_\ell^{\ell+1} : \Omega_{\ell+1} \to \Omega_\ell$, respectively denote restriction and interpolation operators between consecutive refinement levels.
- $\mathscr{L}^c$ is the composite fine grid discrete operator obtained by discretizing the PDE on Ω^c, and $\mathscr{L}^\ell$ approximates $\mathscr{L}^c$ on level ℓ.

With this notation, we can specify the FAC method as in Algorithm 1. After an initial residual is computed, smoothing is done on each level to determine a correction to the solution on that level. The levels are treated sequentially, from finest to coarsest, followed by a solve on the coarsest grid and then smoothing and correction from the coarsest to the finest levels. Algorithm 1 depicts an FAC V-cycle; as with multigrid methods, it is possible to specify alternative schedules for visiting levels, such as slash cycles or W-cycles.

Algorithm 1. FAC

Initialize: $r^c = f^c - \mathscr{L}^c u^c$; $f^\ell = I_c^\ell r^c$
foreach Ω_ℓ, $\ell = J, \ldots, 2$
  Smooth: $\mathscr{L}^\ell e^\ell = f^\ell$
  Correct: $u^c = u^c + I_\ell^c e^\ell$
  Update: $r^c = f^c - \mathscr{L}^c u^c$
  Set: $f^{\ell-1} = I_c^{\ell-1} r^c$
Solve: $\mathscr{L}^1 e^1 = f^1$
Correct: $u^c = u^c + I_1^c e^1$
foreach $\Omega_\ell, \ell = 2, \ldots, J$
  Update: $r^c = f^c - \mathscr{L}^c u^c$
  Set: $f^\ell = I_c^\ell r^c$
  Smooth: $\mathscr{L}^\ell e^\ell = f^\ell$
  Correct: $u^c = u^c + I_\ell^c e^\ell$

Algorithm 1 makes clear the *multiplicative* nature of FAC: the residual is updated with the latest correction information before each smoothing pass can proceed. To be fully effective, each smoothing pass must properly account for the data dependencies among different patches within a refinement level. In our calculations, we use red-black Gauss–Seidel smoothing on each refinement level. We also have the capability to use weighted-point-Jacobi or zebra-line Gauss–Seidel smoothing. The correction steps require synchronization of the composite grid solution to make it consistent on all refinement levels. Note that the residual update can, in principle, be computed only on the most recently corrected refinement level plus a small border on the next coarser level, but we have found that residual evaluation is not expensive enough to justify this optimization. On the coarsest level, we use one V-cycle of `hypre`'s [22] implementation of semi-coarsening multigrid (SMG) [56]. Note that an isotropic coarsening multigrid algorithm would work just as well in this context, but SMG was chosen based on `hypre` availability considerations. The small problem sizes on the coarsest level ensure that the costs for using SMG are not significant.

5. Numerical results

This section introduces several challenging test cases with the goal of demonstrating two main aspects of our implicit AMR implementation: algorithmic performance, and its accuracy properties. In regards to performance, we demonstrate that the convergence properties of the iterative approach are essentially independent of the number of grid levels present in the simulation (for equivalent fine-level resolution), and that it has good scaling properties with respect to the total number of unknowns. Both these aspects are central in a scalable implicit AMR algorithm.

Accuracy and error propagation in the SAMR context is also a major subject of this section. Clearly, the main motivation for using an AMR-based strategy is to minimize the number of unknowns required to achieve a given error level, by using fine resolution only where it is needed. However, time integration on SAMR grids can pose additional difficulties and introduce sources of error not encountered in uniform-grid calculations, especially at the interface between coarse and fine refinement levels. For nonlinear problems, these errors can exhibit themselves in unexpected ways that are often hard to identify and offset. In fact, the potential exists that errors generated at coarse–fine interfaces, combined with the wrong temporal integrator, may overwhelm the simulation and result in errors that scale with the coarsest (instead of the finest) resolution in the computational mesh.

A fundamental aspect for optimal error control in SAMR is the adequate placement of patches within the domain. Careless placement of patches can in fact offset any gains that may be obtained with the additional resolution provided. Optimal placement of patches, in turn, requires suitable error estimators. We will demonstrate the importance of this issue numerically later in this section. For our test problems, we employ refinement indicators based on the magnitude of the current, J, and the curvature in the vorticity, $\mathscr{U}$, as a guide for patch placement. In particular, we compute the cell quantities

$$\tau^1_{i,j} = \frac{|(J)_{i,j}|}{\max_{(i,j)}|(J)_{i,j}|}, \quad \tau^2_{i,j} = \frac{|h_x^2(\mathscr{U}_{xx})_{i,j}| + |h_y^2(\mathscr{U}_{yy})_{i,j}|}{0.2\max_{i,j}|\mathscr{U}_{i,j}|}. \tag{16}$$

Here, $\mathscr{U}_{xx}$ and $\mathscr{U}_{yy}$ denote second-order partial derivatives of the vorticity in x and y, respectively. The maximum values for J and $\mathscr{U}$ in (16) are calculated over each refinement level. Cells where $\tau^1_{i,j} > \epsilon_1$ or $\tau^2_{i,j} > \epsilon_2$ where ϵ_1 and ϵ_2 are user chosen thresholds, are tagged for refinement. In this application $\epsilon_1 = 0.65$ and $\epsilon_2 = 0.3$. Similar refinement indicators are used in Refs. [10,1].

For the accuracy tests, since analytic solutions are not available, the numerical error is computed by comparing a given SAMR simulation against a uniform fine-grid solution (1024×1024 unless otherwise specified), obtained with an extremely small time step.

In what follows, we discuss these issues for three test problems: a tearing mode problem, the island coalescence problem, and the tilt instability problem. All these problems feature the dynamic development of thin current layers, and benefit from an AMR treatment.

5.1. Tearing mode problem

The first problem we consider is the tearing mode problem of [16]. Tearing modes are resistive instabilities, with behavior strongly dependent on the details of the resistive layer at the rational surface (defined as the surface where $\vec{B}\cdot\vec{k} = 0$, with $\vec{k}$ the wavevector of the magnetic perturbation). As the resistive layer thickness scales as $\sqrt{\eta}$, the resolution issues become increasingly challenging with smaller η.

Following [16], we pose the problem in a square domain $[0,4]\times[0,1]$, with periodic boundary conditions in x and homogeneous Dirichlet boundary conditions in y (as specified in Section 2) for all variables. The initial conditions for this problem are $\mathscr{U}_0(x,y) = \Phi_0(x,y) = 0$, and Ψ_0 given by the Harris sheet equilibrium:

$$\Psi_0(x,y) = \frac{1}{\lambda}\ln\left[\cosh\left(\lambda\left(y-\frac{1}{2}\right)\right)\right].$$

The initial current J_0 is found as $J_0 = \Delta\Psi_0$. For the runs below, we have used $\lambda = 5$ and $\eta = \nu = 10^{-3}$.

Fig. 5 shows the evolution of the system at different times on a dynamic SAMR grid with four refinement levels. The coarsest level is a uniform 32×32 grid, with the finest level providing the same resolution as a 256×256 uniform grid. The refinement levels track the evolution of the tearing mode, providing resolution only in localized regions.

5.1.1. Performance

In general, algorithmic scalability of a PDE solver numerical algorithm requires that work per iteration be proportional to the problem size *and* that the number of iterations be independent of the problem size. In the context of SAMR, however, it is difficult to test for scalability directly by performing grid convergence studies, because the problem size is difficult to predict *a priori* when one changes base grids and levels of refinement for a given simulation. To overcome this limitation, in this work we follow two figures of merit to characterize algorithmic performance in a SAMR context. The first one is the same as in single-grid approaches, and measures algorithmic performance under uniform-grid refinement. The second one is SAMR-specific, and measures the effects on performance of changing coarse-grid resolutions and SAMR levels of refinement *for a fixed maximum resolution*. We *define* a SAMR solver to be equivalent-to-uniform-mesh if the SAMR solver performance

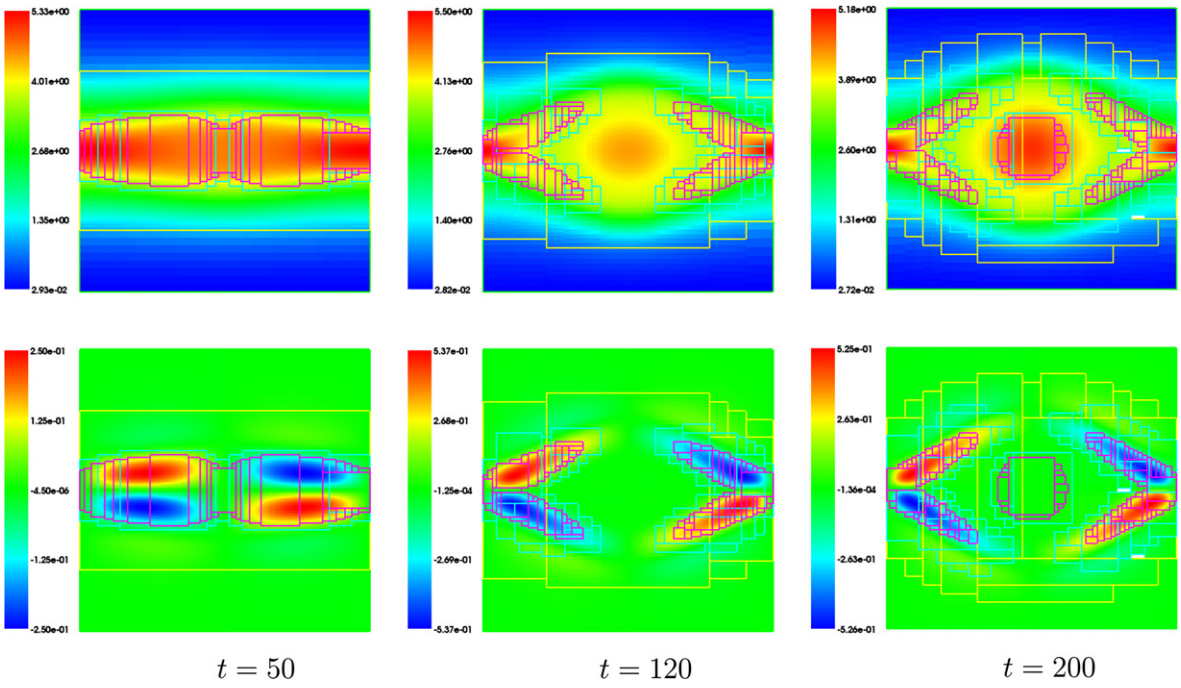

Fig. 5. Time snapshots of current (top) and vorticity (bottom) for the tearing mode problem.

depends *only* on the maximum effective resolution (i.e., solver performance is independent of base grid refinement and the number of refinement levels). Furthermore, we *define* a SAMR algorithm to be scalable if it scales under uniform-grid refinement, and if it features equivalent-to-uniform-mesh performance.

Performance data for the tearing mode simulation are presented in Table 1. These results have been obtained using $\eta_k = 0.1$, $\epsilon_{\text{rel}} = 0.0$ and $\epsilon_{\text{abs}} = 10^{-4}$, and two iterations of the SI preconditioner. Within the preconditioner, two V(3,3) cycles of FAC were used to invert the semi-implicit operator outlined in Section 3.2. The implicit time step has been fixed to a multiple of the explicit CFL, $\Delta t = 140\Delta t_{CFL}$ (which corresponds to a time step of $\Delta t = 5$ for the coarsest 64×32 grid), and we have averaged the performance results over the course of the simulation up to $T_{max} = 250$.

The first column of Table 1 specifies the resolution of the coarse grid, while levels refers to the number of refinement levels active on the SAMR grid. In Table 1, NNI is the number of nonlinear iterations and NLI is the number of linear iterations per time step. A standard uniform-grid convergence study is found in the Levels = 1 column, and shows that the solver features a constant NNI and a very mild increase in NLI (which only increases by 50% for a 64-fold increase in number of degrees of freedom). However, such mild increase was also observed in uniform-grid results in Ref. [16] for the same test problem and same values of η, v, and disappeared for smaller values of these parameters. SAMR-solver-equivalence can also be established from this table, since the maximum fine resolution is kept constant when moving diagonally from bottom left to top right, for example from the cell corresponding to the 512×256 grid with one refinement level to the cell with a 64×32 base grid and five levels of refinement. From Table1, we can readily check that both the number of nonlinear iterations, NNI, and the number of linear iterations, NLI, grow very weakly along diagonals, which establishes that our SAMR solver is effectively equivalent to the uniform-grid one.

This equivalent-to-uniform SAMR solver performance is obtained for significantly less computational effort and memory storage than uniform grids would require. For example, the 512×256 run requires a total wall-clock time of 29,919 s, while the equivalent 64×32 run with 4 levels of refinement only requires 17% of that, 4998 s (timings obtained on a MacBook Pro

Table 1
Summary of performance for tearing mode

	NNI					NLI				
Resol.\Levels	1	2	3	4	5	1	2	3	4	5
64×32	2.4	2.7	2.8	2.8	2.7	8.2	10.3	14.1	15.8	16.4
128×64	2.3	2.6	2.7	2.6	–	8.0	12.9	15.1	16.1	–
256×128	2.0	2.1	2.4	–	–	9.4	13.9	12.8	–	–
512×256	2.0	2.1	–	–	–	12.4	13.3	–	–	–

NNI: average number of nonlinear iterations; NLI: average number of linear iterations per time step.

with a 2.33 GHz Intel Core 2 Duo processor, 2 GB of RAM, running Mac OS X 10.4.10.). Furthermore, on average, the 64×32 run with 4 levels of refinement requires only 14% of the number of degrees of freedom of a uniform 512×256 run. Note that the strong correlation between the reduction in degrees of freedom and the reduction in CPU time is also a consequence of the equivalent-to-uniform property of our SAMR implementation.

Similar results both in solver performance as well as time and memory savings were obtained in all other simulations reported in this paper.

5.1.2. Time-integration errors

We seek to characterize the importance of adequate interpolation order during regridding to avoid discretization errors, and of robust damping in the implicit temporal integration scheme to avoid error propagation throughout the domain.

A common approach for time integration on AMR grids is to use linear interpolation to transfer data from an old grid hierarchy to a new grid hierarchy during regridding. For problems with discontinuous coefficients and steep solution gradients [47], this works better than higher-order interpolation schemes. However, linear interpolation is not suitable for all problems. Fig. 6 depicts the error in J on a SAMR grid after one regrid operation at $t = 2.5$ and subsequent time-stepping using Crank–Nicolson (CN) and quadratic interpolation at coarse–fine interfaces up to $t = 7.5$. Linear interpolation was used to transfer data from the old to the new grid hierarchy during regrid. The concentration of discretization errors at coarse–fine interfaces introduced during regridding is evident, despite the fact that quadratic interpolation was used at coarse–fine interfaces during the time evolution. Furthermore, the poor damping properties of CN preserves the memory of generated errors, even if the location of the coarse–fine interfaces changes. To show this, we transfer the previous solution at $t = 7.5$ to a uniform mesh with resolution equivalent to the finest SAMR patch, and run the simulation further in time to $t = 42.5$. The result is depicted in Fig. 7, and shows that the error in J generated during regridding at coarse–fine interfaces remains throughout the calculation. We note that the magnitude of the error does not appear to be changing significantly.

A related test with a different initial grid configuration is shown in Fig. 8. In this case, the error in J is largest on the left and right coarse–fine interfaces at time $t = 7.5$. After several dynamical regridding operations, the solution has memory of errors generated at all coarse–fine interfaces during regridding, and at time $t = 42.5$ the error (Fig. 9) traces all coarse–fine interfaces that have been present during the simulation. In this example, it is clear that, at some locations, the error is actually being amplified further, as for example at the left and right coarse–fine interfaces present in the initial grid hierarchy at $t = 7.5$.

The previous examples illustrate potential sources of error that can accumulate due to a combination of a poor temporal integrator (CN) and low-order interpolation during regridding, even if higher-order interpolation is used at coarse–fine interfaces. Low-order interpolation generates spatial errors that are not damped in time by the temporal scheme. Coarse-fine errors mostly disappear when sufficiently high-order interpolation is used (both at regrid operations and subsequently during time evolution) in combination with a robustly damping temporal scheme. Fig. 10 shows plots of the error in J at times $t = 10$ and $t = 50$, using cubic interpolation at regrid, quadratic interpolation in between regrid operations, and BDF2 as the time integrator. In this case, we see that there is no accumulation of error at coarse–fine interfaces, even after several regrid operations have taken place.

5.2. Island coalescence

In the island coalescence problem, two magnetic islands (current channels) attract and reconnect. In resistive MHD, and during the reconnection process, a thin, elongated current sheet forms at the reconnection site, which governs the

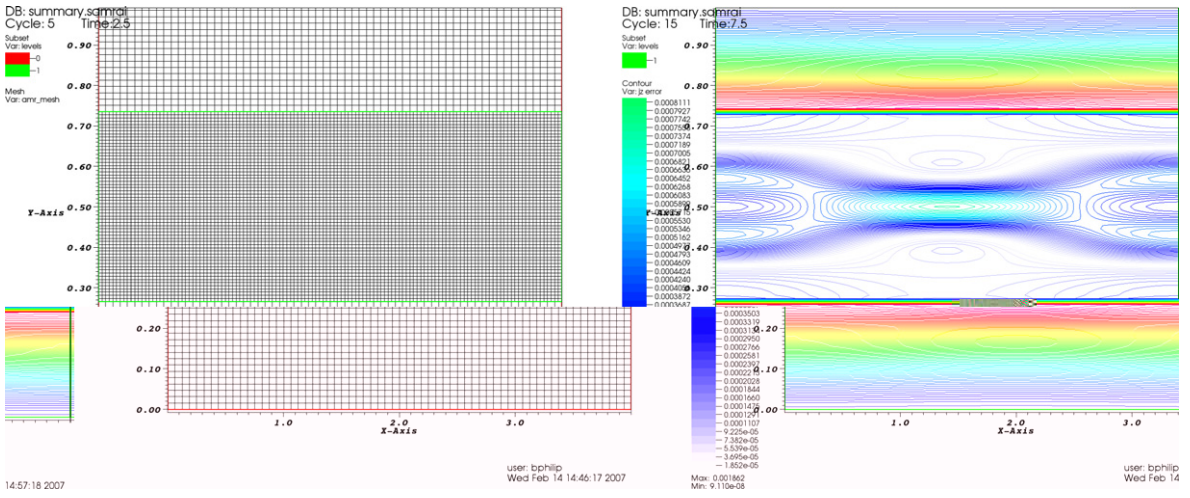

Fig. 6. Grid hierarchy and error in J after one regrid operation at $t = 2.5$ and several time steps up to $t = 7.5$.

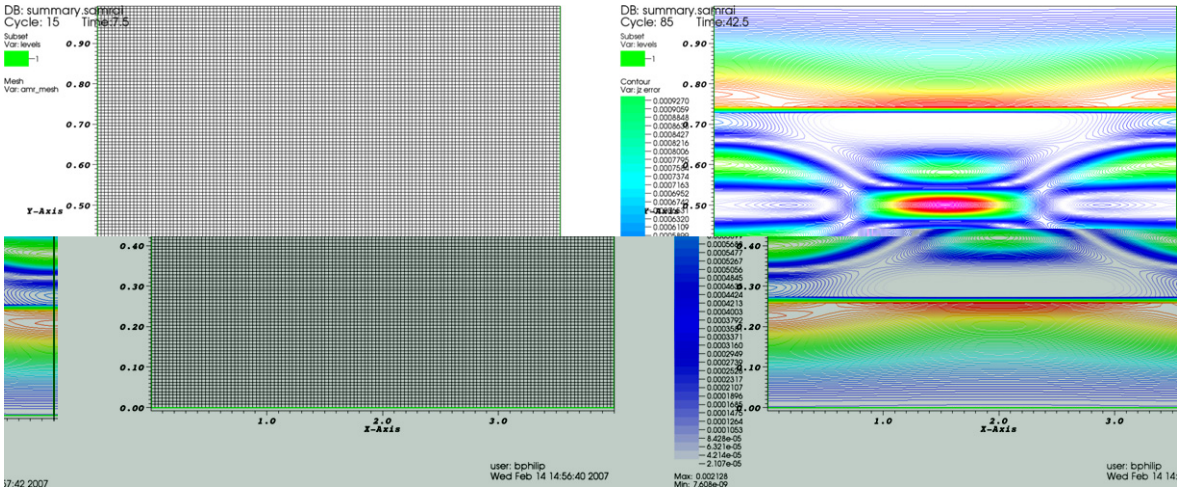

Fig. 7. Grid hierarchy and error in J after a second regrid operation to a uniform mesh at $t = 7.5$ and subsequent time-stepping until $t = 42.5$.

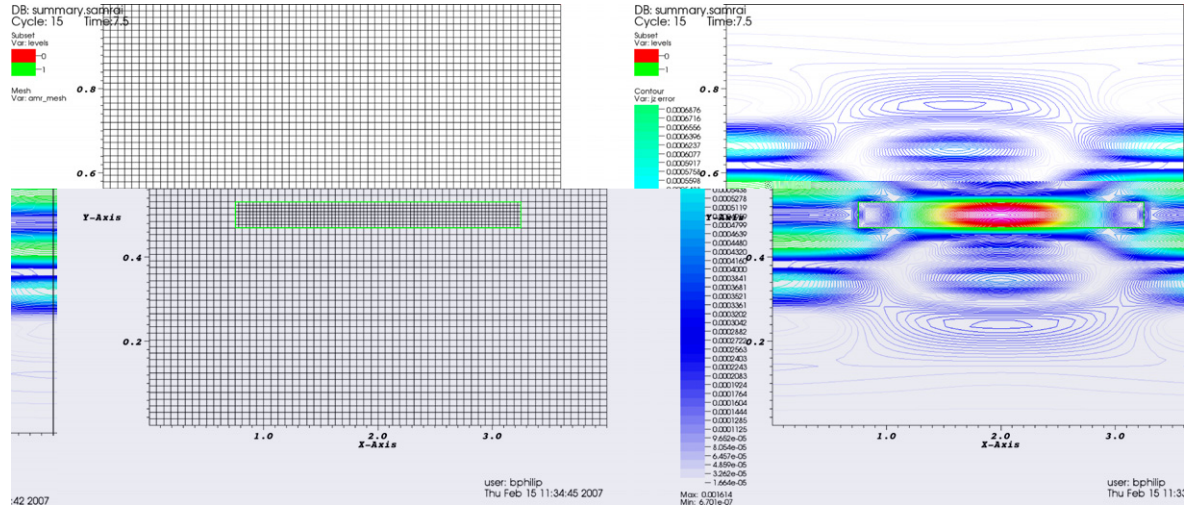

Fig. 8. Grid hierarchy and error in J at $t = 7.5$ after one regrid operation at $t = 2.5$.

reconnection rate, and therefore the global dynamics [36]. As in the tearing mode problem, the current thickness scales as $\sqrt{\eta}$, and therefore the problem becomes computationally more challenging for smaller resistivities.

The island coalescence problem equilibrium is given by $\mathscr{U}_0(x,y) = \Phi_0(x,y) = 0$, and Ψ_0 defined as [25]:

$$\Psi_0(x,y) = -\lambda_\Psi \ln\left[\cosh\left(\frac{y}{\lambda_\Psi}\right) + \epsilon\cos\left(\frac{x}{\lambda_\Psi}\right)\right], \tag{17}$$

where $\lambda_\Psi = \frac{1}{2\pi}$ is the current sheet equilibrium scale length, and $\epsilon = 0.2$ is the island width. The computational domain is $[-1,1]\times[-1,1]$. Boundary conditions are Dirichlet in y (as specified in Section 2), and periodic in x. Both η and ν are set to 10^{-4} for the calculations in Fig. 11 while they are both set to 10^{-3} for the numerical studies presented in the next subsection. The calculation is started with a perturbation in Ψ. Full dynamical SAMR regridding is employed to adapt to developing features. The island configuration at the early stages of the coalescence process is shown in Fig. 11 (left). The configuration at the peak of the reconnection rate, well in the nonlinear stage, is shown in Fig. 11 (right), and shows the formation of the current sheet in the symmetry plane between the islands. The multiscale nature of this problem is evident.

5.2.1. Numerical error generation and propagation

As mentioned earlier, the purpose of grid adaptation is to minimize the number of degrees of freedom required for a given simulation, while being compatible with a given numerical error level. In practice, the expectation is that the overall numer-

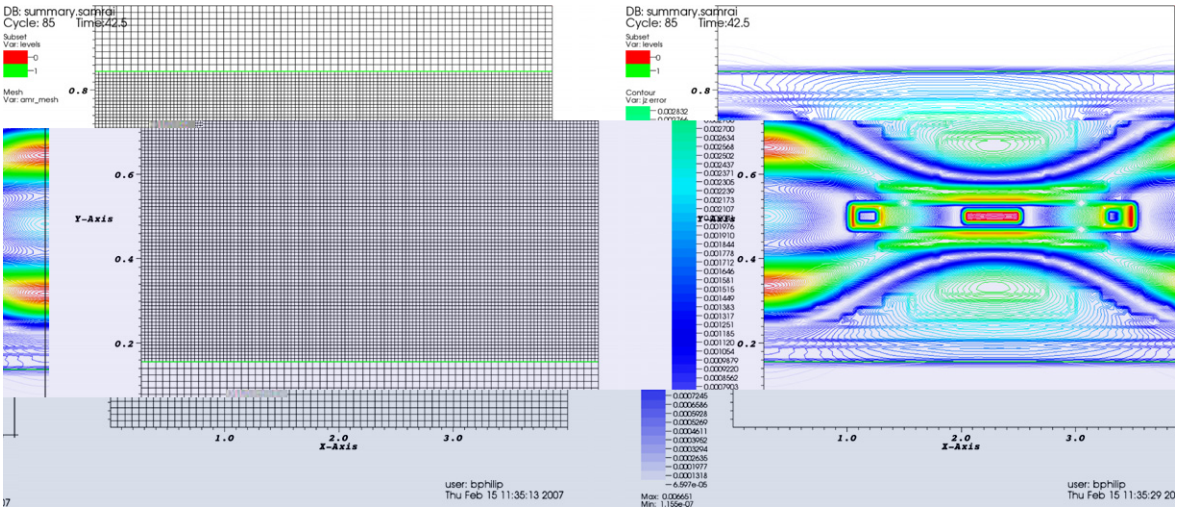

Fig. 9. Grid hierarchy and error in J at $t = 42.5$ after several regridding operations.

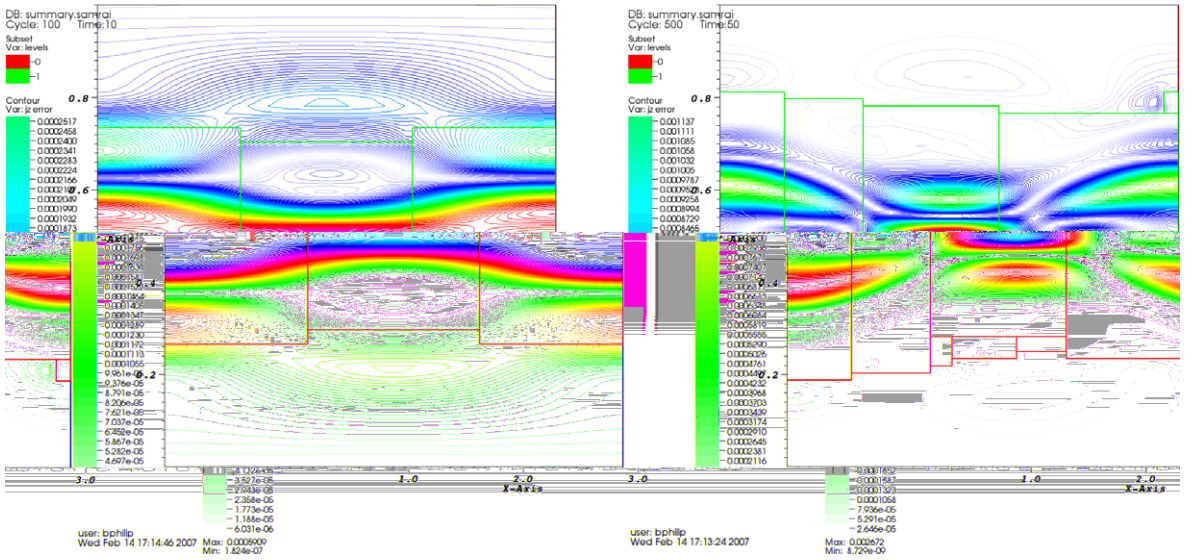

Fig. 10. Plots of error in J for $t = 10$ and $t = 50$ using BDF2 for time integration and cubic interpolation for regridding operations.

ical error of the simulation does not increase due to the presence of patches, and the related coarse–fine boundary treatment. Otherwise, it would defeat the purpose of using a patch-based adaptive scheme in the first place.

We use the island coalescence problem to characterize the generation and temporal propagation of errors in our SAMR implementation. We focus on two main aspects of error generation in SAMR: (1) the treatment of coarse–fine interfaces at patch boundaries and (2) the placement of patches themselves. The former is fundamentally a spatial discretization issue, whereas the latter is more related to error estimation. We note that, once the error estimation process tags cells on the AMR grid hierarchy as requiring refinement, we use the standard Berger–Rigoutsos algorithm [6] implemented in SAMRAI to generate the patches themselves. The parameters used were 0.85 for the efficiency tolerance and 0.85 for the combine efficiency. The efficiency tolerance determines how many cells have to be tagged in a box for the formation of a patch and the combine efficiency determines when two patches are combined into one. Further details are provided in the SAMRAI documentation.

Fig. 12 depicts several time histories of error propagation in SAMR simulations featuring different base grids and levels of refinement. The error is obtained with an *exact* error detector, which compares the SAMR solution with the 1024×1024 reference solution. Fig. 12 shows the volume weighted l_2-norm of the error (which approximates the L_2-norm). We have used a fixed time step in the calculation to isolate spatial errors. As a consequence, and as can be observed in Fig. 12, the time

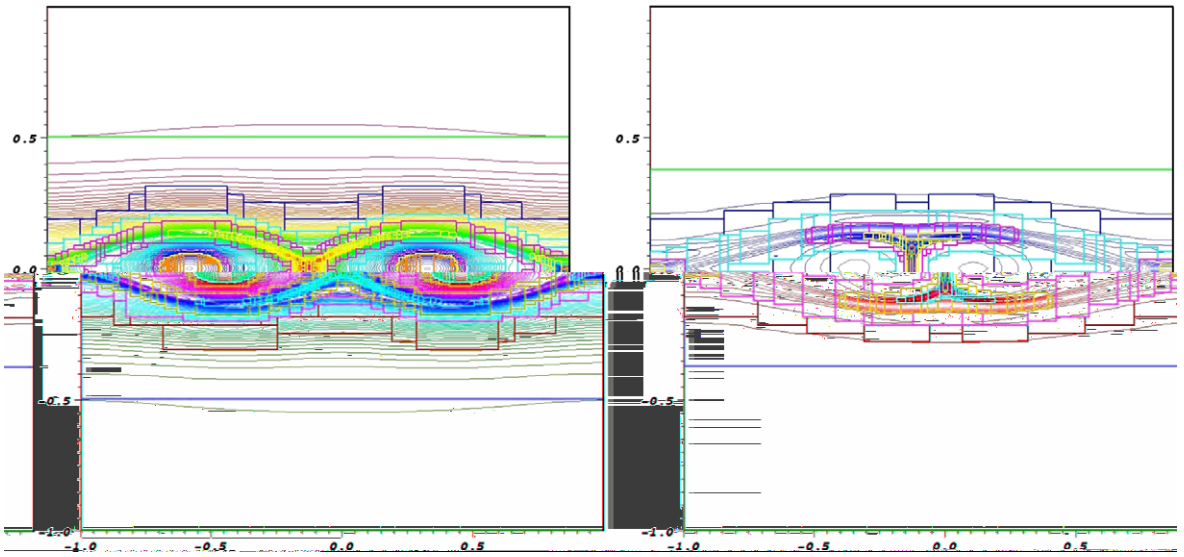

Fig. 11. Snapshots of the current at early stages of the coalescence process ($t = 4$, left) and at the peak of the reconnection ($t = 8$, right) during the coalescence process.

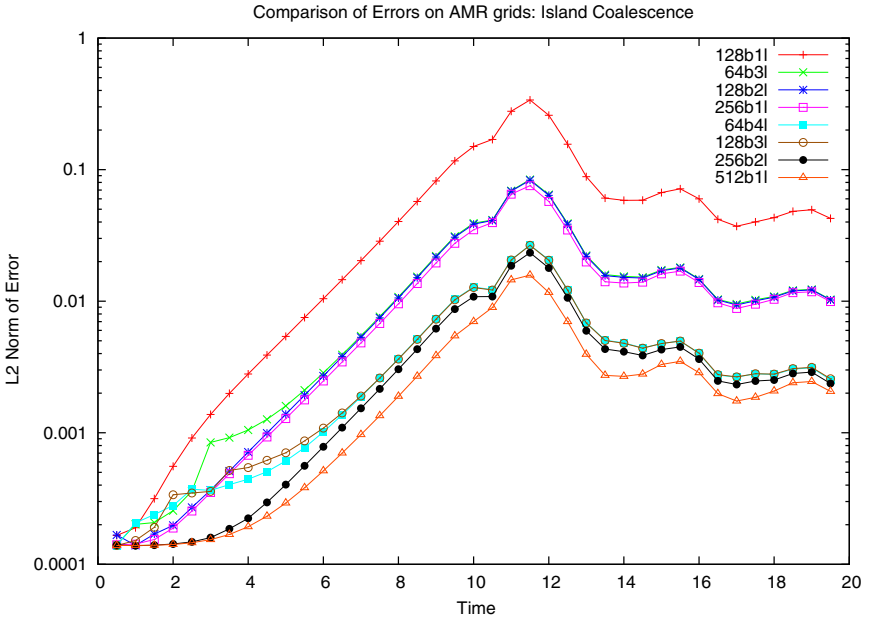

Fig. 12. Time histories of numerical errors for the island coalescence problem with various base grids and levels of refinement. The convention in the legend is *n*b*m*l, which denotes an $n \times n$ base grid with m levels of refinement.

history of the numerical error mimics the evolution of the physical instability (i.e., exponential growth followed by nonlinear saturation). The point of this plot is to establish that the error is fundamentally a function of the finest resolution employed, and not a function of the base grid size or the number of levels of refinement. For instance, the 64b3l, 128b2l, and 256b1l simulations feature the *same* error (time histories are actually superimposed), while using different base grid refinements and number of grid levels. The same is true for 64b4l, 128b3l, 256b2l, and 512b1l (although error differences are more noticeable due to the log scale of the plot). Another side point from this figure is the impact that adding a level of refinement has on the overall error of the computation: the error decreases by a factor of four on average, similar to the error reduction that would be obtained by uniform refinement.

The effect of choice of refinement indicator is depicted in Fig. 13, where time histories of the numerical error resulting from two different error estimators are provided. The *exact* error estimator defined above is compared against the *ad-hoc*

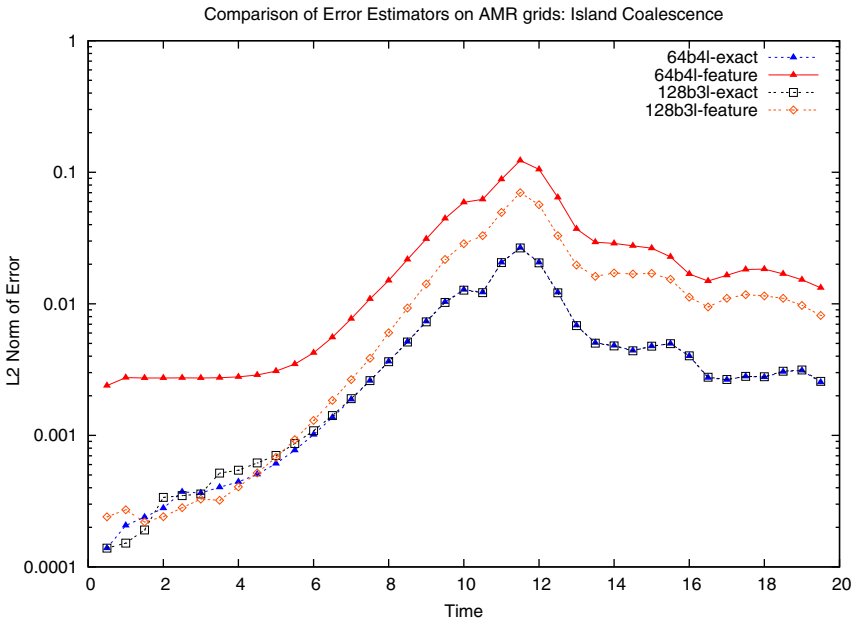

Fig. 13. Time histories of numerical errors for the island coalescence problem with different error estimators for patch placement.

indicator defined in (16). The error plots labeled '64b4l-exact' and '128b3l-exact' (which both have 512×512-equivalent resolution) depict the magnitude of the L_2 error on an AMR grid when the placement of patches is determined by the *exact* error detector. Similarly, the error plots labeled '64b4l-feature' and '128b3l-feature' depict the error on an AMR grid determined using the *ad-hoc* error estimator in (16). As can be seen from Fig. 13, the 64b4l-exact and 128b3l-exact plots lie on top of each other, demonstrating no dependence of the numerical error on the number of refinement levels. On the other hand, appreciable differences between the 64b4l-feature and 128b3l-feature error plots demonstrate a dependence of the numerical error on the number of refinement levels. Furthermore, comparison of the 64b4l-exact and 64b4l-feature plots (or the 128b3l-exact and 128b3l-feature) shows clearly that large errors (up to a factor of 5 in the nonlinear phase) result from employing the *ad-hoc* error estimator instead of the *exact* one. These error differences seem to originate from differences in both the placement of patches as well as the number of degrees of freedom made available per patch. This result underscores the importance of further research on reliable error estimators for adaptive-grid applications. This is left for future work.

5.3. Tilt mode

The third model problem we consider is the tilt-mode instability [50,60,33]. The initial conditions are $\mathscr{W}_0(x,y) = \Phi_0(x,y) = 0$, and Ψ_0 given by:

$$\Psi_0(x,y) = \begin{cases} \frac{2}{kJ_0(k)} J_1(kr)\cos(\theta) & \text{if } r \leqslant 1 \\ (r - \frac{1}{r})\cos(\theta) & \text{if } r > 1 \end{cases}$$

with $J_0 = \Delta\Psi_0$. The boundary conditions are periodic in x and Dirichlet in y for all variables (as specified in Section 2). The system is perturbed from its initial equilibrium with a rotational perturbation in Φ of the form: $\delta\Phi = 10^{-3}e^{-r^2}$. The domain for this problem is $[-2\pi, 2\pi] \times [-5, 5]$. The parameter k is the zero of the Bessel function of the first kind, i.e., $J_1(k) = 0$, and both η and ν are set to 10^{-3}. The refinement criteria for this problem is also given by (16). Fig. 14 shows snapshots of the current and vorticity during the evolution of the tilt instability at times $t = 4.0$ (early in the linear phase) and $t = 7.0$ (well in the nonlinear regime). This calculation required seven levels of refinement starting from a coarse initial 64×64 mesh.

As evolving features in this problem are extremely small compared to the domain size, this AMR calculation on average only required 0.36% of the degrees of freedom of a uniform-grid calculation (a uniform-grid calculation would have required 67,108,864 degrees of freedom, which corresponds to a 4096×4096 uniform grid with four unknowns per grid cell). This example serves to illustrate the significant savings that AMR can provide over uniform-grid calculations. We note that the relatively small number of degrees of freedom enabled us to perform this calculation on a workstation, while a parallel machine would have been required to perform the uniform-grid calculation.

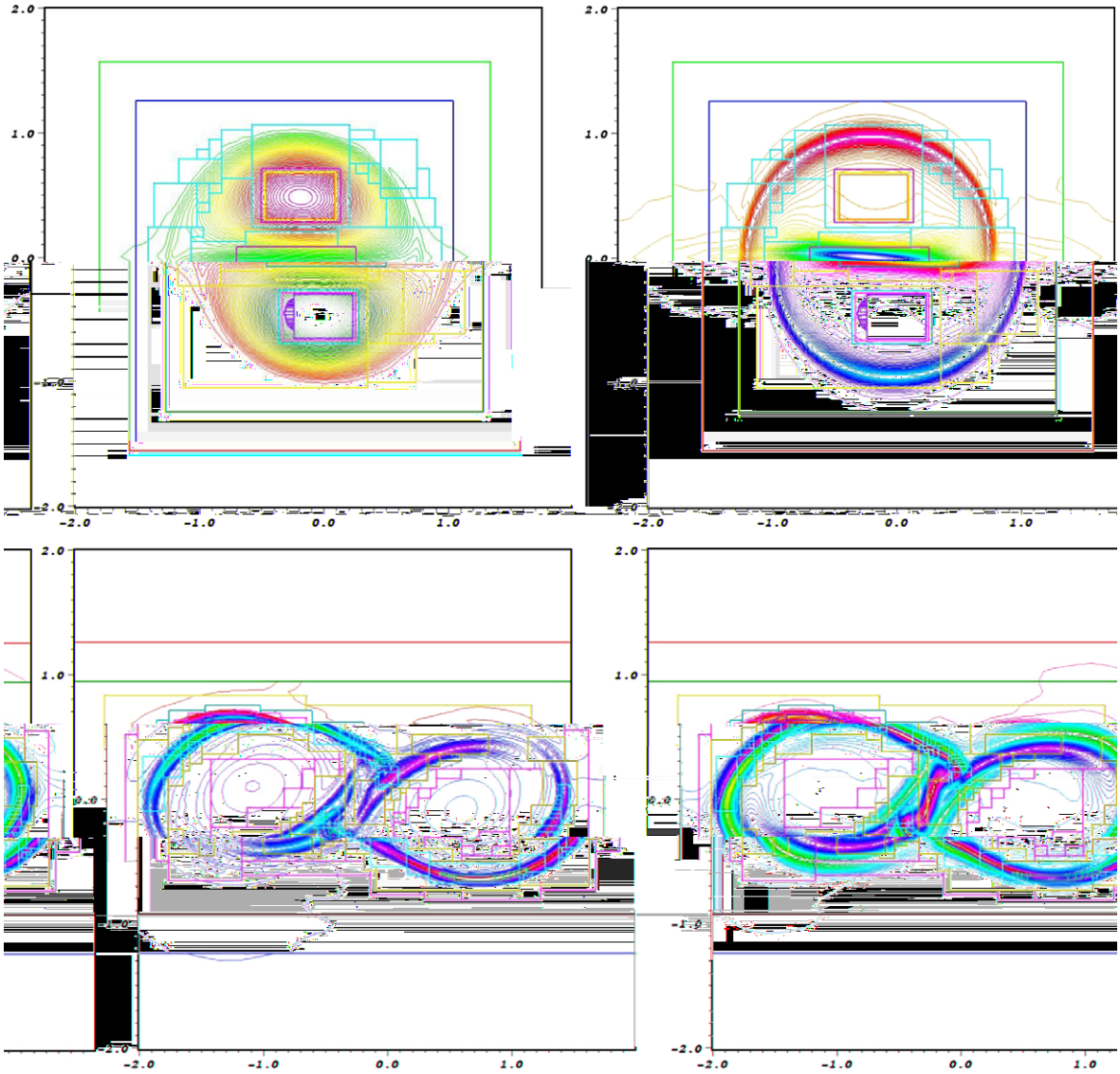

Fig. 14. Current (left) and vorticity (right) during the evolution of the tilt instability at times $t = 4.0$ (top) and $t = 7.0$ (bottom).

6. Conclusions

We have described the implementation of an algorithmically scalable, fully implicit, SAMR simulation tool for 2D reduced resistive MHD. The tool employs Jacobian-free Newton–Krylov methods as the solver engine. We use the reduced MHD solver developed in [16] as a starting point, albeit with several modifications. Following [58], we have reformulated the original problem (in terms of flux, vorticity, and streamfunction) using the current as a dynamic variable instead of the flux. This avoids issues with SAMR and the high-order derivatives in terms like $\vec{B} \cdot \nabla(\Delta\Psi)$ present in the flux formulation.

We have also extended the optimal "physics-based" approach developed in [16] (which employed multigrid solver technology in the preconditioner for scalability) for this application in two ways. Firstly, we have adapted the preconditioner formulation to deal with J instead of Ψ, as required. Secondly, we have extended the multilevel treatment in [16] to SAMR grids using the well-known Fast Adaptive Composite grid (FAC) method [42]. As a result, our approach inherits the algorithmic benefits of a multilevel treatment *and* the accuracy benefits of dynamic grid adaptation.

We have demonstrated such benefits with several challenging tests. A grid convergence study has shown that the SAMR solver performance is independent of the number of grid levels and only depends on the finest resolution considered, and that it scales well with grid refinement. The study of error generation and propagation in our SAMR implementation demonstrates that piecewise cubic interpolation at coarse–fine interfaces, combined with a robustly damping second-order temporal scheme such as BDF2, is required to minimize impact of such interfaces on the overall error of the computation. We

also demonstrate that our implementation features the desired property that, when a reliable error estimator is employed, the overall numerical error is dependent only on the finest resolution level considered, and not on the base-grid refinement or on the number of refinement levels present during the simulation.

An open aspect in our implementation is the use of more grounded, rigorous error estimators to select refinement patches, instead of our *ad-hoc* approach. In fact, we have demonstrated numerically that our *ad-hoc* error estimator, while effective, is far from optimal when compared with an *exact* one. In future work, we will explore more rigorous error estimators, such as the τ (or two-grid) error estimator, which employs differences between two grid resolutions to estimate the truncation error. The approach, which is rigorous in the context of linear, conservative operators, can be used effectively for non-linear ones (see e.g. Refs. [11,24,41,23] for theory and practical implementation details and effectiveness of this error estimator in various contexts; in particular, Ref. [23] employs it for an anisotropic AMR implementation).

Acknowledgements

The authors acknowledge useful discussions with D.A. Knoll. The work was funded by the Los Alamos Directed Research and Development program at Los Alamos National Laboratory, operated for DOE under contract No. DE-AC52-06NA25396, and by the DOE Office of ASCR program in Applied Mathematical Sciences.

References

[1] M.J. Aftosmis, M. Berger, Multilevel error estimation and adaptive *h*-refinement for cartesian meshes with embedded boundaries, in: 40th AIAA Aerospace Sciences Meeting and Exhibit, No. 2002-0863 in AIAA Paper, January 2002.
[2] A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome, A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations, J. Comput. Phys. 142 (1998) 1–46.
[3] S. Balay, W.D. Gropp, L. Curfman-McInnes, B.F. Smith, PETSc Users Manual, Tech. Report ANL-95/11 – Revision 2.1.6, Argonne National Laboratory, 2004.
[4] D.S. Balsara, Divergence-free adaptive mesh refinement for magnetohydrodynamics, J. Comput. Phys. 174 (2001) 614–648.
[5] M. Berger, M. Aftosmis, J. Melton, Accuracy, adaptive methods and complex geometry, 1996.
[6] M. Berger, I. Rigoutsos, An algorithm for point clustering and grid generation, IEEE Trans. Syst. Man Cybernet. 21 (Sep/Oct) (1991) 1278–1286.
[7] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64–84.
[8] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[9] M. Berzins, P.J. Capon, P.K. Jimack, On spatial adaptivity and interpolation when using the method of lines, Appl. Numer. Math. 26 (1997) 117–133.
[10] J.G. Blom, J.G. Verwer, Algorithm 759: Vlugr3: a vectorizable adaptive-grid solver for PDEs in 3d; part ii. code description, ACM Trans. Math. Software 22 (1996) 329–347.
[11] A. Brandt, Multigrid techniques: 1984 guide, with applications to fluid dynamics, gmd studien nr. 85, GMD, GMD-AIW, Postfach 1240, D-5205, St. Augustin 1, Germany, 1984, 191 p.
[12] P.N. Brown, Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, SIAM J. Sci. Statist. Comput. 11 (1990) 450–481.
[13] L. Chacón, A non-staggered, conservative, $\nabla \cdot \mathbf{B} = \mathbf{0}$, finite-volume scheme for 3D implicit extended magnetohydrodynamics in curvilinear geometries, Comput. Phys. Comm. 163 (2004) 143–171.
[14] L. Chacón, An optimal, parallel, fully implicit Newton–Krylov solver for three-dimensional visco-resistive magnetohydrodynamics, Phys. Plasmas 15 (2008) 056103.
[15] L. Chacón, D.A. Knoll, A 2D high-β Hall MHD implicit nonlinear solver, J. Comput. Phys. 188 (2003) 573–592.
[16] L. Chacón, D.A. Knoll, J.M. Finn, An implicit, nonlinear reduced resistive MHD solver, J. Comput. Phys. 178 (2002) 15–36.
[17] J. Crank, P. Nicolson, A practical method for numerical evaluation of solutions of partial differential equations of the heat conduction type, Proc. Cambridge Phil. Soc. 43 (1947) 50–67.
[18] C.F. Curtiss, J.O. Hirschfelder, Integration of stiff equations, Proc. Nat. Acad. Sci. 38 (1952) 235–243.
[19] R.S. Dembo, S.C. Eisenstat, T. Steihaug, Inexact Newton methods, SIAM J. Numer. Anal. 19 (1982) 400–408.
[20] J.F. Drake, T.M. Antonsen, Nonlinear reduced fluid equations for toroidal plasmas, Phys. Fluids 27 (1984) 898–908.
[21] S.C. Eisenstat, H.F. Walker, Globally convergent inexact Newton methods, SIAM J. Optim. 4 (1994) 393–422.
[22] R.D. Falgout, U.M. Yang, HYPRE: a library of high performance preconditioners, in: P.M.A. Sloot, C.J.K. Tan, J.J. Dongarra, A.G. Hoekstra (Eds.), Computational Science – CARS 2002 Part III, Lecture Notes in Computer Science, vol. 2331, Springer-Verlag, New York, 2002, pp. 632–641.
[23] L. Ferm, P. Lotstedt, Anisotropic grid adaptation for Navier–Stokes equations, J. Comput. Phys. 190 (2003) 22–41.
[24] L. Ferm, P.L. Lotstedt, Adaptive error control for steady state solutions of inviscid flow, SIAM J. Sci. Comput. 23 (2002) 1777–1798.
[25] J.M. Finn, P. Kaw, Coalescence instability of magnetic islands, Phys. Fluids 20 (1977) 72–78.
[26] C.W. Gear, Numerical initial value problems in ordinary differential equations, Prentice-Hall, 1971.
[27] K. Germaschewski, A. Bhattacharjee, C.-S. Ng, The Magnetic Reconnection code: an AMR-based fully implicit simulation suite, in: N.B. Pogorelov, G.P. Zank, (Eds.), Numerical Modeling of Space Plasma Flows, ASP Conference Series, vol. 359, 2006.
[28] T.I. Gombosi, Solution-adaptive magnetohydrodynamics for space plasmas: sun-to-earth simulations, Comput. Sci. Eng. 6 (2004) 14–35.
[29] D.M. Greaves, Simulation of viscous water column collapse using adapting hierarchical grids, Int. J. Numer. Methods Fluids 50 (2006) 693–711.
[30] R.D. Hazeltine, M. Kotschenreuther, P.J. Morrison, A four-field model for tokamak plasma dynamics, Phys. Fluids 28 (1985) 2466–2477.
[31] C.W. Hirt, Heuristic stability theory for finite-difference equations, J. Comput. Phys 2 (1968) 339–355.
[32] R.D. Hornung, S. Kohn, Managing application complexity in the SAMRAI object-oriented framework, Concurrency Comput.: Pract. Exp. 14 (2002) 347–368.
[33] S.C. Jardin, J.A. Breslau, Implicit solution of the four-field extended-magnetohydrodynamic equations using high-order high-continuity finite elements, Phys. Plasmas 12 (2005) 056101.
[34] C.T. Kelley, Iterative Methods for Linear and Nonlinear Equations, SIAM, Philadelphia, 1995.
[35] R. Keppens, Adaptive mesh refinement for conservative systems: multi-dimensional efficiency evaluation, Comput. Phys. Comm. 153 (2003) 317–339.
[36] D.A. Knoll, L. Chacón, Coalescence of magnetic islands, sloshing, and the pressure problem, Phys. Plasmas 13 (2006) 32307–32311.
[37] D.A. Knoll, L. Chacón, L. Margolin, V.A. Mousseau, On balanced approximations for the time integration of multiple time scale systems, J. Comput. Phys. 185 (2003) 583–611.
[38] D.A. Knoll, D.E. Keyes, Jacobian-free Newton-Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[39] D.A. Knoll, P.R. McHugh, Enhanced nonlinear iterative techniques applied to a nonequilibrium plasma flow, SIAM J. Sci. Comput. 19 (1998) 291–301.
[40] S. Lankalapalli, J.E. Flaherty, M.S. Shephard, H. Strauss, An adaptive finite element method for magnetohydrodynamics, J. Comput. Phys. 225 (2007) 363–381.

[41] P. Lotstedt, S. Soderberg, A. Ramage, L. Hemmingsson-Franden, Implicit solution of hyperbolic equations with space–time adaptivity, BIT 42 (2002) 134–158.
[42] S. McCormick, Multilevel Adaptive Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1989.
[43] S.F. McCormick, J.W. Thomas, The fast adaptive composite grid (FAC) method for elliptic equations, Math. Comput. 46 (1986) 439–456.
[44] P.R. McHugh, 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 (1994) 2394–2400.
[45] C.S. Ng, D. Rosenberg, K. Germaschewski, A. Pouquet, A. Bhatthacharjee, A comparison of spectral element and finite difference simulations with adaptive mesh refinement for the MHD island coalescence instability problem, Astrophys. J. Suppl., accepted for publication.
[46] M. Pernice, R.D. Hornung, Newton–Krylov-FAC methods for problems discretized on locally refined grids, Comput. Vis. Sci. 8 (2005) 107–118.
[47] M. Pernice, B. Philip, Solution of equilibrium radiation diffusion problems using implicit adaptive mesh refinement, SIAM J. Sci. Comput. 27 (2006) 1709–1726.
[48] B. Philip, M. Pernice, L. Chacón, Solution of reduced resistive magnetohydrodynamics using implicit adaptive mesh refinement, in: Olof B. Widlund, David E. Keyes (Eds.), Domain Decomposition Methods in Science and Engineering XVI, Lecture Notes in Computational Science and Engineering, vol. 55, Springer-Verlag, 2007, pp. 725–731.
[49] K.G. Powell, Parallel, AMR MHD for global space weather simulations, Lecture notes in computational science and engineering 41 (2005) 473–490.
[50] R.L. Richard, R.D. Sydora, M. Ashour-Abdalla, Magnetic reconnection driven by current repulsion, Phys. Fluids B: Plasma Phys. 2 (1990) 488–494.
[51] D. Rosenberg, A. Pouquet, P.D. Mininni, Adaptive mesh refinement with spectral accuracy for magnetohydrodynamics in two space dimensions, New J. Phys. 9 (2007) 304.
[52] Y. Saad, M. Schultz, GMRES: A generalized minimal residual algorithm for solving non-symetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[53] R. Samtaney, Adaptive mesh refinement for MHD fusion applications, Lecture Notes Comput. Sci. Eng. 41 (2005) 491–503.
[54] R. Samtaney, P. Colella, T.J. Ligocki, D.F. Martin, S.C. Jardin, An adaptive mesh semi-implicit conservative unsplit method for resistive MHD, in: A. Mezzacappa et al., (Ed.), SciDAC 2005, San Francisco, June 26–30. Journal of Physics: Conference Series, 2005.
[55] R. Samtaney, S.C. Jardin, P. Colella, D.F. Martin, 3D adaptive mesh refinement simulations of pellet injection in tokamaks, Comput. Phys. Comm. 164 (2004) 220–228.
[56] S. Schaffer, A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients, SIAM J. Sci. Comput. 20 (1999) 228–242.
[57] M. Shashkov, S. Steinberg, Support-operator finite-difference algorithms for general elliptic problems, J. Comput. Phys. 118 (1995) 131–151.
[58] H. Strauss, D. Longcope, An adaptive finite element method for magnetohydrodynamics, J. Comput. Phys. 147 (1998) 318–336.
[59] H.R. Strauss, Nonlinear, 3-dimensional magnetohydrodynamics of noncircular tokamaks, Phys. Fluids 19 (1976) 134–140.
[60] H.R. Strauss, D.W. Longcope, An adaptive finite element method for magnetohydrodynamics, J. Comput. Phys. 147 (1998) 318–336.
[61] G. Tóth, D.L.D. Zeeuw, T.I. Gombosi, K.G. Powell, A parallel explicit/implicit time stepping scheme on block-adaptive grids, J. Comput. Phys. 217 (2006) 722–758.
[62] U. Trottenberg, C.W. Oosterlee, A. Schuller, Multigrid, Academic Press Inc., San Diego, CA, 2001. With contributions by A. Brandt, P. Oswald, K. Stuben.