Implicit Anisotropic Magnetic Resistivity in the FLASH Code

E. C. Hansen[1], A. C. Reyes[1], M. B. P. Adams[1], J. Carroll-Nellenback[1,2], J. R. Davies[2], K. Weide[3], D. Q. Lamb[3], P. Tzeferacos[1,2]

[1]Department of Physics and Astronomy, University of Rochester, USA; [2]Laboratory for Laser Energetics, University of Rochester, USA; [3]Department of Astronomy and Astrophysics, University of Chicago, USA

Abstract

Explicit methods of solving the resistive MHD equations can be subject to restrictive time step constraints to maintain numerical stability. The bottleneck is often caused by a small diffusion time from the resistive term in the magnetic induction equation. Implicit methods increase temporal stability, allowing for time steps orders of magnitude larger than those used in explicit methods while maintaining accuracy. In FLASH, we have implemented an implicit algorithm for solving the anisotropic resistive term in the magnetic induction equation. The algorithm also updates the face-centered fields on a staggered mesh in a way that preserves the divergence-free constraint. Energy is updated via calculations of Ohmic heating and the Poynting flux. We present a simple 1D test problem that illustrates the computational speed-up from an explicit method to the new implicit method. The new implementation will be most useful for modeling laser-driven plasma experiments that exhibit relatively high magnetic resistivity.

Resistive MHD

The resistive term in the induction equation is $\frac{d\vec{B}}{dt} = -\vec{\nabla}\times\vec{E} = -\vec{\nabla}\times(\overleftrightarrow{\eta}\cdot\vec{J})$, where $\overleftrightarrow{\eta}$ is the resistivity tensor. This tensor can be broken into components: $\eta_\parallel, \eta_\perp$, and $\eta_\wedge$ (for this work, we will ignore $\eta_\wedge$). In FLASH, we write the component-wise form of the resistive E-field in three different ways, and the code will use one depending on the desired physics and solver method.

Isotropic: $\vec{E} = \eta_\parallel \vec{J}$

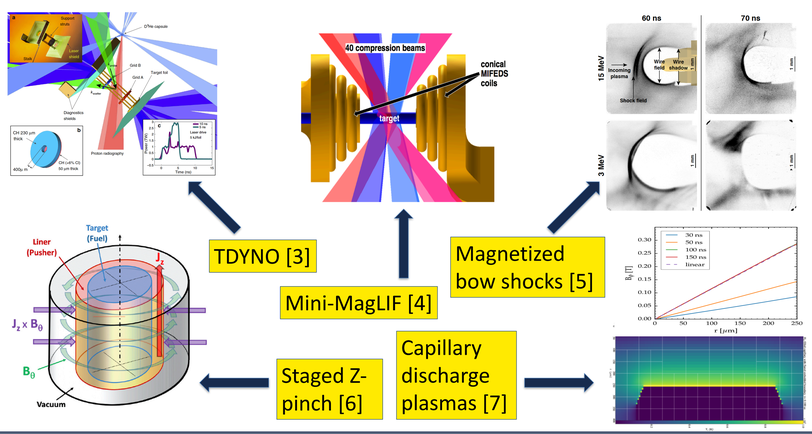
Anisotropic (diffusive): $\vec{E} = \eta_\perp \vec{J} + (\eta_\parallel - \eta_\perp)\vec{b}\,(\vec{b}\cdot\vec{J})$

(advective): $\vec{E} = \eta_\parallel \vec{J} - (\eta_\parallel - \eta_\perp)(\vec{b}\times\vec{J})\times\vec{b}$

One can use vector identities to show that the two anisotropic forms are equivalent. The isotropic form can be solved implicitly or with a flux-based explicit approach. The diffusive anisotropic form is always solved implicitly, and the advective form explicitly. The implicit solver is compatible with FLASH's unsplit staggered MHD solver [1], which preserves solenoidality at machine precision. We have also incorporated new resistivity coefficients into FLASH, which were derived from numerical calculations using the Fokker-Planck code OSHUN and give the coefficients as analytic functions of Z (effective ionization) and χ (electron Hall parameter) [2].

Motivation and Applications

The primary motivation for an implicit method is a practical computational one. Explicit methods are temporally limited to the diffusion time step $dt_{diff} = \frac{(\Delta r)^2}{\eta}$, which can be orders of magnitude smaller than the hydrodynamic time step. Hence, an implicit method can save a significant amount of computational time, and below we show a few examples of problems that would benefit from such an approach.



Procedure

Time step *n* calculations

We calculate face-centered E-fields according to the form of the resistivity (e.g., $\vec{E}^n_{i-1/2} = \eta_{i-1/2}\vec{J}^n_{i-1/2}$ for the isotropic case). We also calculate the face-centered Poynting flux:

$$\vec{S}^n_{i-1/2} = \vec{E}^n_{i-1/2} \times \vec{B}^n_{i-1/2}$$

Implicit diffusion

In 3D, HYPRE uses a 57-point stencil due to complexity from curl operators.

We set up a matrix equation of the form:

$$\left(1 - \theta\, dt\, F(\eta)\right)\vec{B}^{n+1} = \left(1 + (1-\theta)\, dt\, F(\eta)\right)\vec{B}^n \quad ,$$

where $F(\eta)$ is a matrix diffusion operator the size of the domain. θ determines the implicit scheme and is typically set to 0 (forward Euler), 1 (backward Euler), or 1/2 (Crank-Nicolson). The Crank-Nicolson scheme has the advantage of being time-centered 2nd order accurate, whereas the others are 1st order accurate in time. We use the HYPRE library to iteratively solve for $\vec{B}^{n+1}$ (the cell-centered magnetic field).

Time step *n+1* calculations

Again calculate face-centered E-fields and Poynting fluxes, but now we are using the diffused cell-centered B-fields. Like in the first step of the procedure, the cell-centered B-fields are used when calculating the face-centered currents:

$$\vec{J}^{n+1}_{i-1/2} = \vec{\nabla}\times\vec{B}^{n+1}$$

Face B-field update

Not done in 1D.

First, average the face-centered E-fields to get edge-centered values. Then these edge-centered values can be used to update the face-centered B-fields using Stokes' Theorem, which maintains $\vec{\nabla}\cdot\vec{B} = 0$. Here is an example of the equation for updating B_x in 2D:

$$B^{n+1}_{x,i-1/2} = B^n_{x,i-1/2} - \frac{dt}{A_{i-1/2}}\begin{pmatrix} \theta E^{n+1}_{z,i-1/2,j+1/2} \\ -\theta E^{n+1}_{z,i-1/2,j-1/2} \\ (1-\theta)E^n_{z,i-1/2,j+1/2} \\ -(1-\theta)E^n_{z,i-1/2,j-1/2} \end{pmatrix}$$

where $A_{i-1/2}$ is the face area, which becomes dy in 2D. This use of edge-centered E-fields to update face-centered B-fields is described in more detail in [8], although their equations use values at *n + 1/2,* whereas we have made use of θ to allow for the use of either or both time step data.

Cell-centered B-fields are re-updated by averaging the new face values.

Energy update

If face field update was done in previous step, there is an additional energy correction since $\vec{S}$ was derived from cell-centered values.

We now use the Poynting flux to update the total energy density: $u^{n+1} = u^n - dt\,(\theta\vec{\nabla}\cdot\vec{S}^{n+1} + (1-\theta)\vec{\nabla}\cdot\vec{S}^n)$.

It can be shown that $\frac{du}{dt} = -\vec{\nabla}\cdot\vec{S} = \vec{J}\cdot\vec{E} + \frac{d(\frac{B^2}{2})}{dt}$. In other words, the change in total energy is equal to the addition from Ohmic heating plus the change in magnetic energy. By knowing the old and new total energy and old and new magnetic energy, we can calculate the Ohmic heating term, which is then added to the electron internal energy. Depending on the EOS, the code can now update the temperature and pressure.

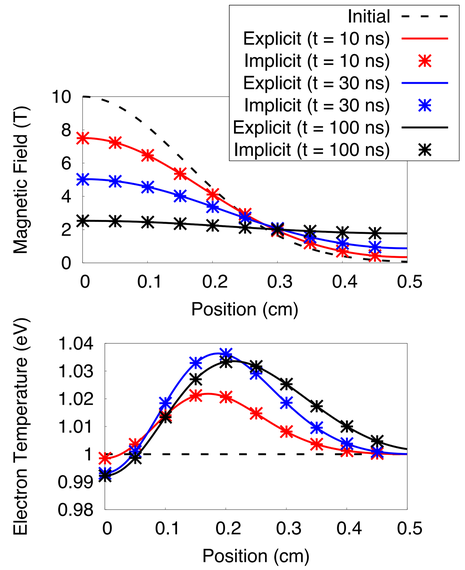
Test Problem

We run a 1D cylindrical test problem in which all initial input parameters are constant except for B_z which has a radial Gaussian profile. Initial conditions are given in the table to the right. This problem aims to verify the accuracy of the implicit solver as compared to the flux-based explicit solver. It tests isotropic resistivity, because under these initial conditions, $\eta_\parallel \approx \eta_\perp$.

Parameter	Value
ρ (mg/cm^3)	1.0
Te (eV)	1.0
Peak B_z (T)	10
Z	1.0
R_{max} (cm)	0.5
ΔR (μm)	5.0

Results

The figures below show a time series of radial profiles of magnetic field (*top*) and electron temperature (*bottom*). The solution from the flux-based explicit method is represented by solid lines, and the solution from the diffusive implicit method is represented by data points.



The results are nearly exact. In the explicit method, the magnetic flux (i.e., electric field) is saved, and a direct calculation of $\vec{J}\cdot\vec{E}$ (Ohmic heating) is used to update the internal energy. The implicit method uses the Poynting flux and conservation of energy as described in the procedure to the left. As evidenced by the electron temperature profiles, both approaches to calculating the Ohmic heating term produce equivalent results.

The tests were run on a single core, and the table below convincingly shows the speed-up from using an implicit method.

Method	Time steps	Run time (s)
Explicit	336123	2759
Implicit	487	7

Speed-up is significant when $dt_{diff} \ll dt_{hydro}$.

Future Work

- More complex setups in 2D and 3D need to be tested to observe anisotropy
- Current implementation only works for FLASH's uniform grid configuration; much more work to be done to make implicit solver compatible with AMR

References

[1] D. Lee, Journal of Computational Physics 243, 269 (2013)
[2] J. R. Davies, H. Wen, Jeong-Young Ji, and Eric D. Held, Physics of Plasmas 28, 012305 (2021)
[3] P. Tzeferacos et al., Nature Communications 9, 591 (2018)
[4[E. C. Hansen et al., Physics of Plasmas 27, 062703 (2020)
[5] J. Levesque et al., submitted to Physics of Plasmas (2021)
[6] E. Ruskov, P. Ney, and H. Rahman, Physics of Plasmas 27, 042709 (2020)
[7] N. Cook et al., Journal of Physics Conference Series 1596, 012063 (2020)
[8] D. S. Balsara and D. S. Spicer, Journal of Computational Physics 149, 270 (1999)

This work was supported in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0001272. The Flash Center for Computational Science acknowledges support from the U.S. Department of Energy National Nuclear Security Administration under Subcontract No. 536203 with Los Alamos National Laboratory, Subcontract B632670 with Lawrence Livermore National Laboratory, and Award Numbers DE-NA0003842 and DE-NA0003934. We also Acknowledge support from the National Science Foundation, Award Number PHY-2033925.