Implicit MHD Based on SUNDIALS Software

Future Directions for M3D Workshop
Princeton Plasma Physics Lab
20 March 2007

Daniel R. Reynolds, *drreynolds@ucsd.edu*
Ravi Samtaney, *samtaney@pppl.gov*
Carol S. Woodward, *cswoodward@llnl.gov*

Outline

I. MHD Description and Equations

II. Space-Time Discretization

III. Solution Approach

IV. Preconditioning Approach

V. Numerical Results

VI. Conclusions and Continuing Research

Resistive MHD – Conservation Form

Using standard vector identities,

$$\nabla \times (\mathbf{a} \times \mathbf{b}) = (\nabla \mathbf{a})\mathbf{b} - (\nabla \mathbf{b})\mathbf{a} + (\nabla \cdot \mathbf{b})\mathbf{a} - (\nabla \cdot \mathbf{a})\mathbf{b},$$
$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \cdot \mathbf{c})\mathbf{b} - (\mathbf{a} \cdot \mathbf{b})\mathbf{c}, \quad \ldots$$

we reformulate the restive MHD model as the advection-diffusion system,

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0,$$
$$\partial_t (\rho \mathbf{v}) + \nabla \cdot \left(\rho \mathbf{v}\mathbf{v}^T - \mathbf{B}\mathbf{B}^T + \left(p + \tfrac{1}{2}\mathbf{B} \cdot \mathbf{B}\right) \bar{\mathrm{I}} \right) + \mathbf{B}(\nabla \cdot \mathbf{B}) = \nabla \cdot \bar{\tau},$$
$$\partial_t e + \nabla \cdot \left((e + p + \tfrac{1}{2}\mathbf{B} \cdot \mathbf{B})\mathbf{v} - \mathbf{B}(\mathbf{B} \cdot \mathbf{v}) \right) = \nabla \cdot (\bar{\tau}\mathbf{v} + \kappa \nabla T)$$
$$+ \nabla \cdot \left(\eta \left(\tfrac{1}{2}\nabla(\mathbf{B} \cdot \mathbf{B}) - \mathbf{B}(\nabla \mathbf{B})^T \right) \right) + \nabla \cdot (\mathbf{B}(\nabla \cdot \mathbf{B})),$$
$$\partial_t \mathbf{B} + \nabla \cdot \left(\mathbf{v}\mathbf{B}^T - \mathbf{B}\mathbf{v}^T \right) = \nabla \cdot \left(\eta \nabla \mathbf{B} - \eta \left(\nabla \mathbf{B} \right)^T \right).$$

- Standard models drop terms proportional to $\nabla \cdot \mathbf{B}$.
- Allows numerical methods that preserve conservation to machine precision.

Condensed Form

Condensing notation, we solve for $\mathbf{U} = (\rho,\ \rho\mathbf{v},\ \mathbf{B},\ e)^T$:

$$\partial_t \mathbf{U} = -\nabla \cdot \mathbf{F}_h(\mathbf{U}) + \nabla \cdot \mathbf{F}_v(\mathbf{U}) = \nabla \cdot \mathbf{F}(\mathbf{U}),$$

where

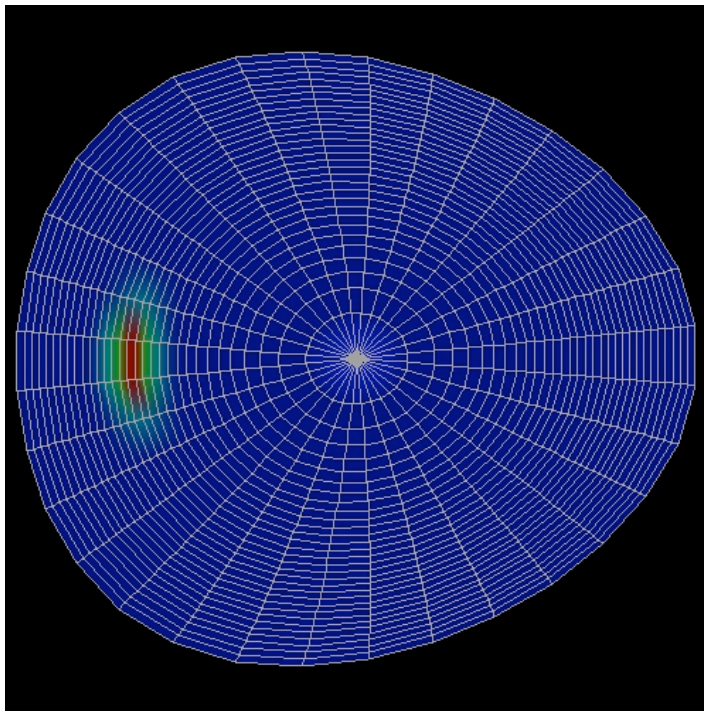
$$\mathbf{F}_h(\mathbf{U}) = \begin{pmatrix} \rho\mathbf{v} \\ \rho\mathbf{v}\mathbf{v}^T - \mathbf{B}\mathbf{B}^T + (p + \frac{1}{2}\mathbf{B}\cdot\mathbf{B})\bar{\mathrm{I}} \\ \mathbf{v}\mathbf{B}^T - \mathbf{B}\mathbf{v}^T \\ (e + p + \frac{1}{2}\mathbf{B}\cdot\mathbf{B})\mathbf{v} - \mathbf{B}(\mathbf{B}\cdot\mathbf{v}) \\ 0 \end{pmatrix}, \longleftarrow \text{Hyperbolic flux (\textit{Ideal MHD})}$$

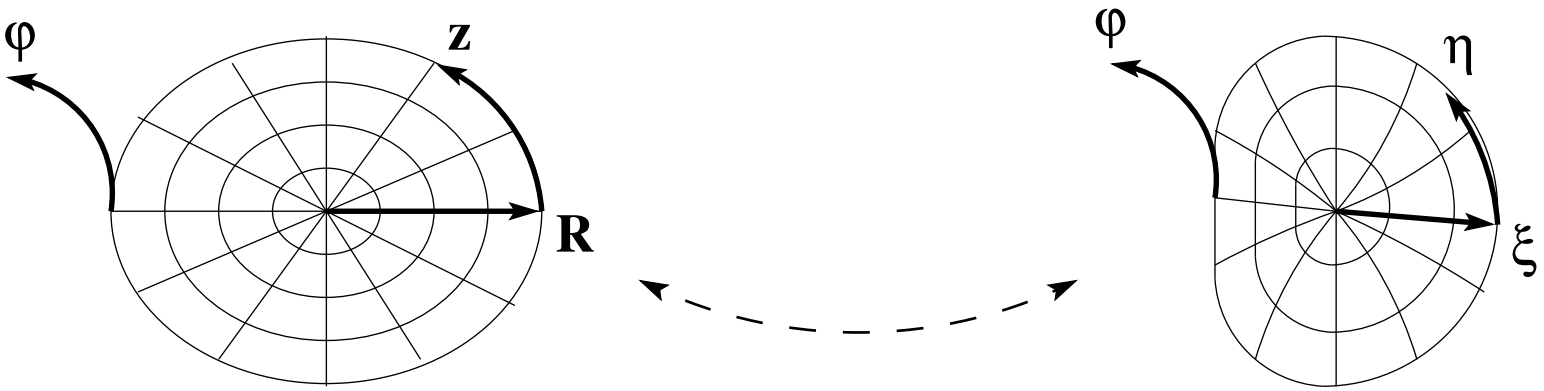
$$\mathbf{F}_v(\mathbf{U}) = \begin{pmatrix} 0 \\ \bar{\tau} \\ \eta\left(\nabla\mathbf{B} - (\nabla\mathbf{B})^T\right) \\ \bar{\tau}\mathbf{v} + \kappa\nabla T + \eta\left(\nabla(\frac{1}{2}\mathbf{B}\cdot\mathbf{B}) - \mathbf{B}(\nabla\mathbf{B})^T\right) \\ 0 \end{pmatrix}, \longleftarrow \text{Parabolic flux}$$

Shaped Plasma Geometry

We adopt a flux-tube coordinate system for modeling the shaped plasma:

- Flux surfaces ψ are determined from a separate equilibrium calculation
- Cylindrical coordinates (R, φ, z) are mapped to the new system (η, φ, ξ): $\xi = \xi(R, z),\ \eta = \eta(R, z)$
- Flux surfaces given by $\psi = \psi_0 \xi$





Shaped Grid Equations

In cylindrical coordinates, (R, φ, z), the system becomes

$$\partial_t \mathbf{U} + \frac{1}{R}\frac{\partial}{\partial R}(R\mathbf{F}(\mathbf{U})) + \frac{\partial}{\partial z}\mathbf{H}(\mathbf{U}) + \frac{1}{R}\frac{\partial}{\partial \varphi}\mathbf{G}(\mathbf{U}) = \mathbf{S}(\mathbf{U}) + \nabla \cdot \mathbf{F}_v(\mathbf{U}),$$

where $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ are the previous hyperbolic fluxes, and $\mathbf{S}$ is a local source term in the $\rho\mathbf{v}_R$, $\rho\mathbf{v}_\varphi$ and $\mathbf{B}_\varphi$ equations due to the transformation.

Transforming to the mapped system, $(R, \varphi, z) \rightarrow (\eta, \varphi, \xi)$, we have

$$\partial_t(\mathbf{U}\mathcal{J}) + \frac{1}{R}\frac{\partial}{\partial \xi}(R\tilde{\mathbf{F}}(\mathbf{U})) + \frac{1}{R}\frac{\partial}{\partial \eta}(R\tilde{\mathbf{H}}(\mathbf{U})) + \frac{1}{R}\frac{\partial}{\partial \varphi}(\mathcal{J}\mathbf{G}(\mathbf{U})) = \tilde{\mathbf{S}}(\mathbf{U}) + \nabla \cdot \mathbf{F}_v(\mathbf{U}),$$

where $\tilde{\mathbf{F}} = \frac{\partial z}{\partial \eta}\mathbf{F} - \frac{\partial R}{\partial \eta}\mathbf{H}$ and $\tilde{\mathbf{H}} = -\frac{\partial R}{\partial \xi}\mathbf{F} + \frac{\partial z}{\partial \xi}\mathbf{H}$, and $\mathcal{J} = \frac{\partial R}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial R}{\partial \eta}\frac{\partial z}{\partial \xi}$.

Note: in both of these formulations, the transformation adds only a local 'reaction' term to the general structure of the advection-diffusion system.

Outline

I. MHD Description and Equations

II. Space-Time Discretization

III. Solution Approach

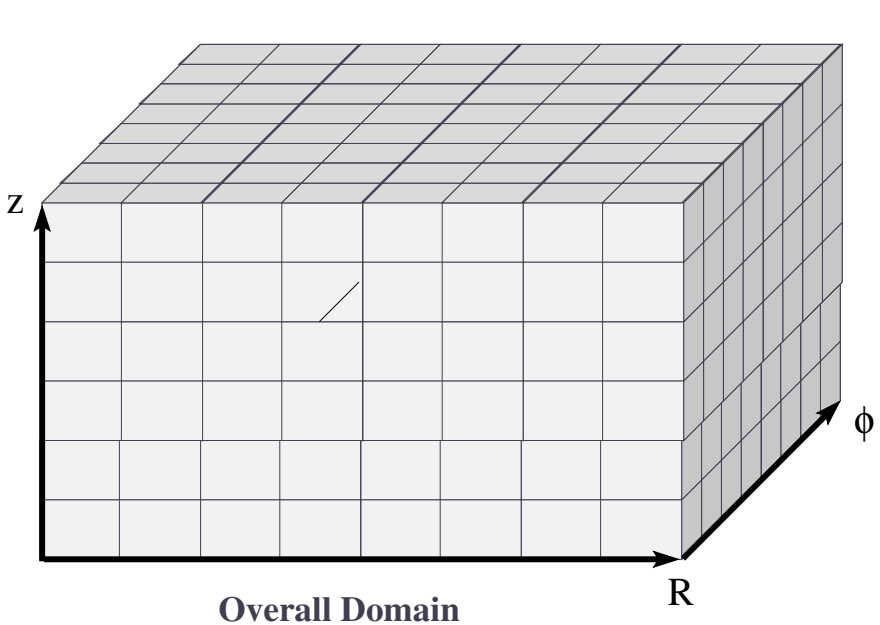
IV. Preconditioning Approach

V. Numerical Results

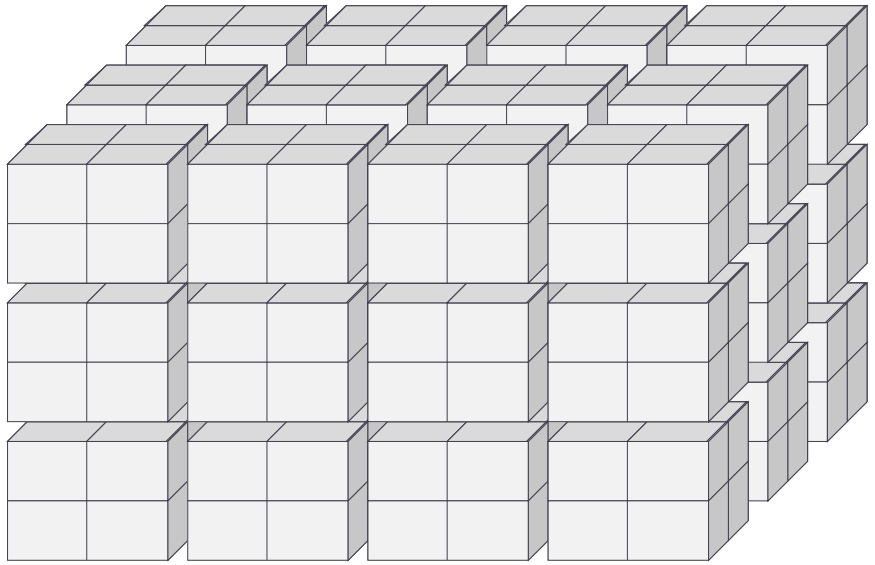
VI. Conclusions and Continuing Research

Spatial Discretization

We treat the cell-centered simulation unknowns as a 3D grid, and decompose our parallel simulation in all directions equally.



Overall Domain



Processor Decomposition

Computational Fluid Dynamics Approach

- Foliate space-time domain into distinct spatial snapshots,

$$\Omega \times [t_0, \infty) \rightarrow \bigcup_n \Omega(t_n).$$

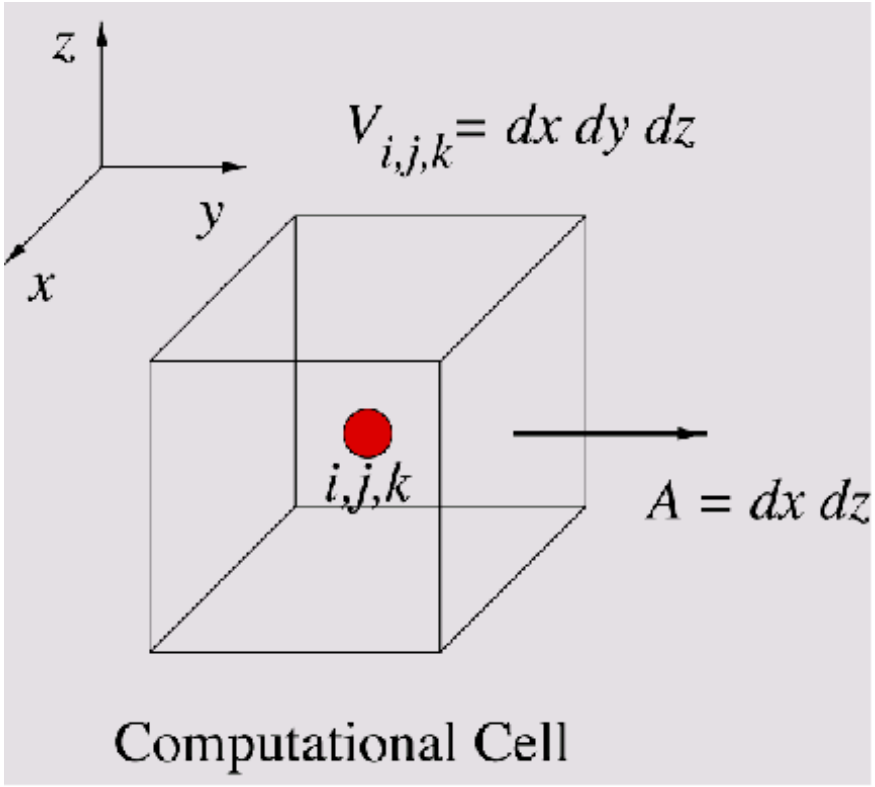
- Cut each slab into cells (boxes),

$$\Omega(t_n) = \bigcup_i \Omega_i(t_n),$$

$\mathbf{U}(x_i, t_n)$ defined at cell centers.

- Evolve $\mathbf{U}(x_i)$ in time using Gauss' Theorem (conservation form):

$$\int_{\Omega_i} \partial_t \mathbf{U}\, dx = \int_{\Omega_i} \nabla \cdot \mathbf{F}(\mathbf{U})\, dx$$
$$= \int_{\partial\Omega_i} \mathbf{F}(\mathbf{U}) \cdot \mathbf{n}\, ds$$



"Finite-Volume" method

Computational Fluid Dynamics Approach

- Foliate space-time domain into distinct spatial snapshots,

$$\Omega \times [t_0, \infty) \rightarrow \bigcup_n \Omega(t_n).$$

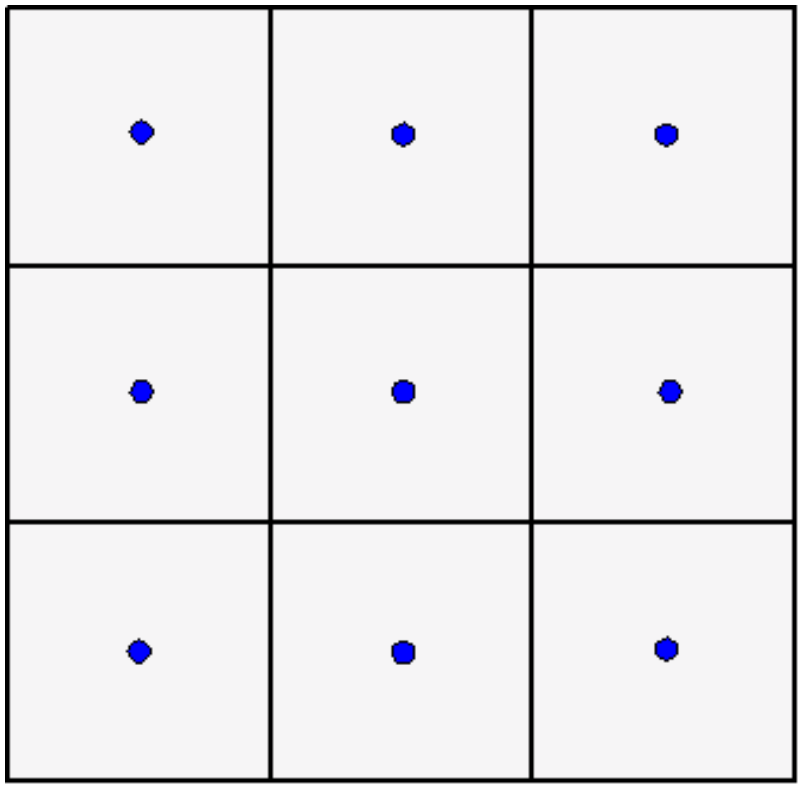
- Cut each slab into cells (boxes),

$$\Omega(t_n) = \bigcup_i \Omega_i(t_n),$$

$\mathbf{U}(x_i, t_n)$ defined at cell centers.

- Evolve $\mathbf{U}(x_i)$ in time using Gauss' Theorem (conservation form):

$$\int_{\Omega_i} \partial_t \mathbf{U}\, dx = \int_{\Omega_i} \nabla \cdot \mathbf{F}(\mathbf{U})\, dx$$
$$= \int_{\partial\Omega_i} \mathbf{F}(\mathbf{U}) \cdot \mathbf{n}\, ds$$



Spatial domain ($\mathbf{U}_{i,j,k}$)

Computational Fluid Dynamics Approach

- Foliate space-time domain into distinct spatial snapshots,

$$\Omega \times [t_0, \infty) \rightarrow \bigcup_n \Omega(t_n).$$

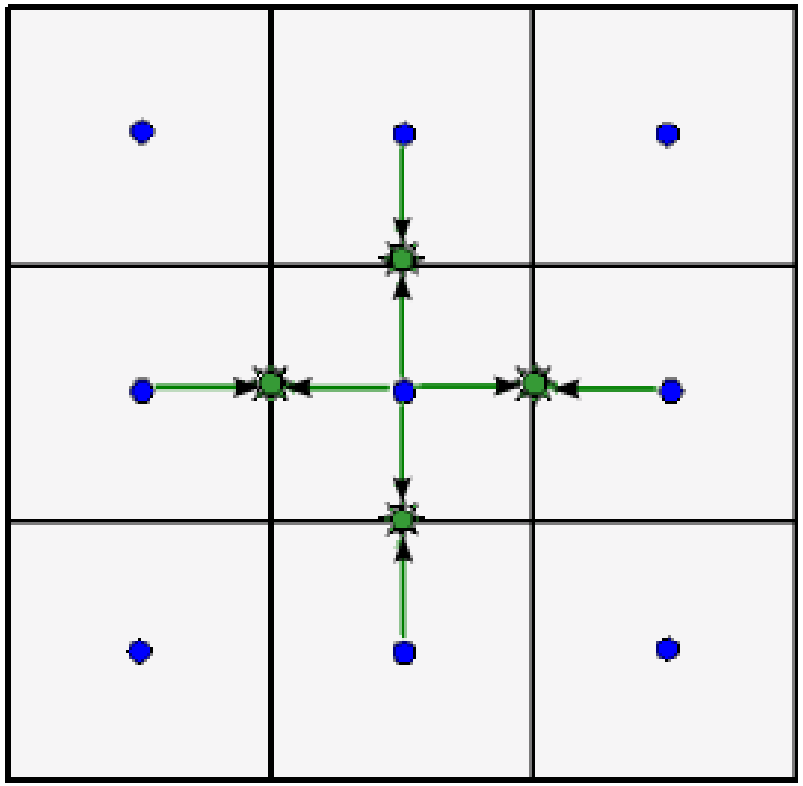
- Cut each slab into cells (boxes),

$$\Omega(t_n) = \bigcup_i \Omega_i(t_n),$$

$\mathbf{U}(x_i, t_n)$ defined at cell centers.

- Evolve $\mathbf{U}(x_i)$ in time using Gauss' Theorem (conservation form):

$$\int_{\Omega_i} \partial_t \mathbf{U}\, dx = \int_{\Omega_i} \nabla \cdot \mathbf{F}(\mathbf{U})\, dx$$
$$= \int_{\partial\Omega_i} \mathbf{F}(\mathbf{U}) \cdot \mathbf{n}\, ds$$



Interface reconstruction ($\mathbf{U}_{i+1/2,j,k}$)

Computational Fluid Dynamics Approach

- Foliate space-time domain into distinct spatial snapshots,

$$\Omega \times [t_0, \infty) \rightarrow \bigcup_n \Omega(t_n).$$

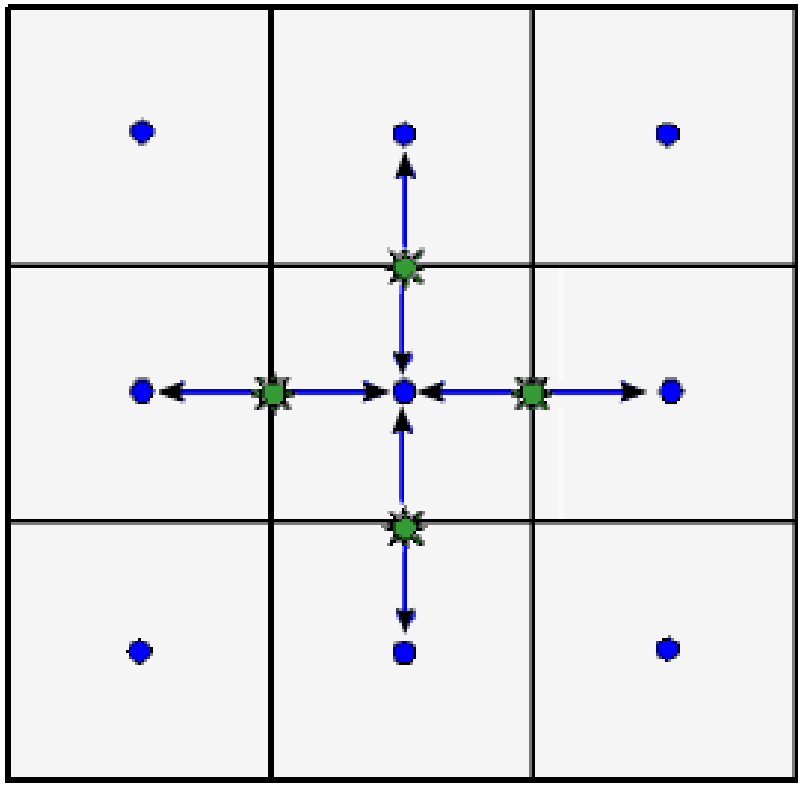
- Cut each slab into cells (boxes),

$$\Omega(t_n) = \bigcup_i \Omega_i(t_n),$$

$\mathbf{U}(x_i, t_n)$ defined at cell centers.

- Evolve $\mathbf{U}(x_i)$ in time using Gauss' Theorem (conservation form):

$$\int_{\Omega_i} \partial_t \mathbf{U}\, dx = \int_{\Omega_i} \nabla \cdot \mathbf{F}(\mathbf{U})\, dx$$
$$= \int_{\partial\Omega_i} \mathbf{F}(\mathbf{U}) \cdot \mathbf{n}\, ds$$



Conservative evolution $\mathbf{F}(\mathbf{U}_{i+1/2,j,k})$

Flux Computations

Reconstruction algorithm for values at interfaces $\partial\Omega_i$ is the heart of CFD techniques. Reconstruction method can dramatically affect

- Physicality of results: $\nabla \cdot \mathbf{B}$, shock resolution
- Numerical stability in presence of steep gradients (Gibbs phenomena)
- Solution accuracy and cost

Our simulations may use any one of:

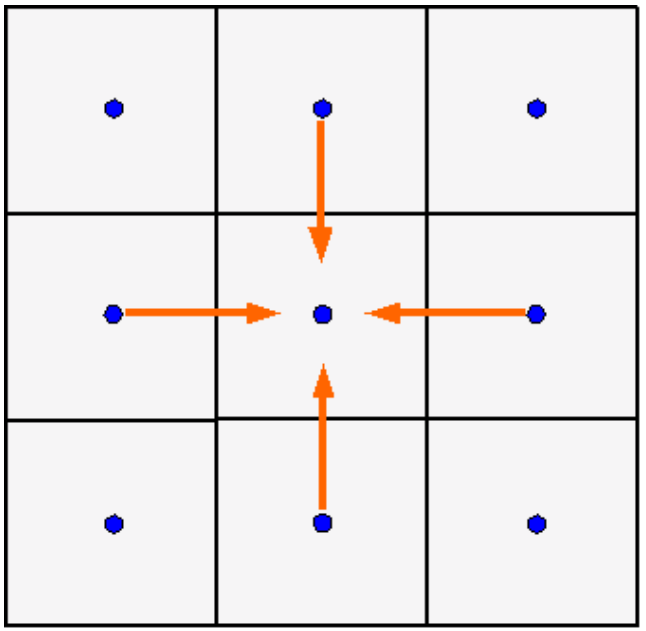
1. *Godunov*: $\mathcal{O}(h^2)$ Roe and Riemann methods; inherently dissipative; helpful for capturing shocks; non-commutative ($\pi_i\pi_j \neq \pi_j\pi_i$)
2. *WENO*: $\mathcal{O}(h^2)$ adaptive stencil; helpful for capturing shocks; $\pi_i\pi_j \neq \pi_j\pi_i$
3. *Centered*: $\mathcal{O}(h^2)$ or $\mathcal{O}(h^4)$ interpolation based on neighbor averages; non-dissipative; increased dispersion error; $\pi_i\pi_j = \pi_j\pi_i$
4. *Tuned Centered*: similar to (3), but with reduced dispersion error
5. *Zip*: similar interpolation properties to (3), possibly improved stability

Explicit vs. Implicit Integration

We may discretize $\partial_t \mathbf{U} = f(\mathbf{U})$ in time:

$$\mathbf{U}^n = \mathbf{U}^{n-1} + \Delta t f(\mathbf{U}^{n-1}), \quad \text{[explicit]}$$

$$\mathbf{U}^n = \mathbf{U}^{n-1} + \Delta t f(\mathbf{U}^n), \quad \text{[implicit]}$$



Explicit methods simpler but less stable:

- Δt *limited by smallest* Δx and fastest dynamical speed, λ, in the problem.
- Results in so-called *CFL condition*,

$$\Delta t_{CFL} \leq \Delta x / \lambda$$

Implicit methods can be stable for *all* time step sizes:

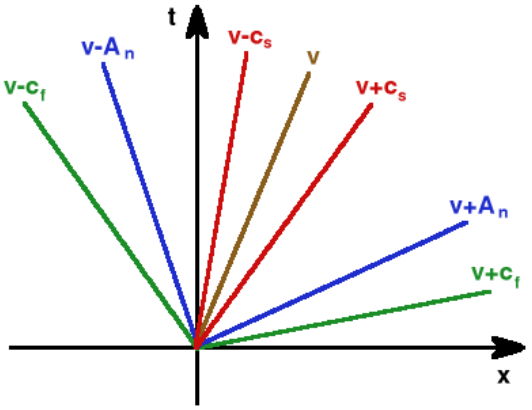
- Potential scalability in time (no Δt dependence on Δx)
- Nontrivial: require a (non)linear solver for solution of the resulting system.

Both allow high-order methods, unlike semi-implicit and linearly-implicit methods which relegate time accuracy to first-order (at best).

Stiffness in Resistive MHD

Resistive MHD stiffness results from fast hyperbolic waves and diffusive effects. The ideal MHD waves are given by

$$\lambda_e = \mathbf{v} \quad \text{(entropy wave)}$$
$$\lambda_d = \mathbf{v} \quad \text{(magnetic-flux wave)}$$
$$\lambda_a = \mathbf{v} \pm A_n \quad \text{(Alfvén waves)}$$
$$\lambda_{f,s} = \mathbf{v} \pm c_{f,s} \quad \text{(fast/slow magnetosonic)}$$



- The characteristic speeds satisfy $c_s << A_n < c_f$. Typically $c_f \approx 10^6$ m/s for fusion plasmas.
- Diffusive (resistive, viscous, XMHD) effects induce quadratic explicit time step dependence on the spatial mesh ($\Delta t \propto (\Delta x)^2$).
- For magnetic reconnection, explicit methods would thus require simulation times that scale as $O(S^{3/2})$.

Outline

I. MHD Description and Equations

II. Space-Time Discretization

III. Solution Approach

IV. Preconditioning Approach

V. Numerical Results

VI. Conclusions and Continuing Research

Solution Approach: Time Stepping (CVODE)

We consider time discretization of the system of ordinary differential equations,

$$\partial_t \mathbf{U}_i = f_i(\mathbf{U}) := \frac{1}{|\Omega_i|} \int_{\partial\Omega_i} \mathbf{F}(\mathbf{U}) \cdot \mathbf{n}\, ds,$$

and evolve the solution using a high-order implicit multistep method (BDF),

$$g(\mathbf{U}^n) := \mathbf{U}^n - \Delta t \beta_0 f(\mathbf{U}^n) - \sum_{i=1}^{q} \left[\alpha_i \mathbf{U}^{n-i} + \Delta t \beta_i f(\mathbf{U}^{n-i})\right].$$

- The time-evolved state $\mathbf{U}^n = \mathbf{U}(x, t_n)$ is found as the solution to the system of nonlinear algebraic equations $g(\mathbf{U}) = 0$.
- For a given order of accuracy q, the parameters α_i and β_i are fixed, e.g.

  BDF1: $\mathbf{U}^n = \mathbf{U}^{n-1} + \Delta t f(\mathbf{U}^n)$

  BDF2: $\mathbf{U}^n = \frac{1}{3}\left(4\mathbf{U}^{n-1} - \mathbf{U}^{n-2} + 2\Delta t f(\mathbf{U}^n)\right)$

- If $q \leq 2$, the method is *A-stable*, otherwise it is *A(α)-stable*
- http://www.llnl.gov/CASC/sundials/

[Gear & Saad, 1983; Hindmarsh et al., 2005]

Adaptive Time Step, Order Selection

CVODE adaptively chooses the time step size Δt and the BDF method order q at each time step to best balance

- Solution accuracy: compares explicit predictor with implicit corrector to estimate local temporal truncation error.
- Temporal stability: monitors solution history data $\{\mathbf{U}^{n-i}\}_{i=1}^{q}$ to detect BDF instability at $q \geq 3$.
- Solver efficiency: estimates optimal Δt to maximize time step while minimizing solution cost.

[Hindmarsh 1992, 1995; Hindmarsh et al., 2005]

Solution Approach: Nonlinear

To solve the nonlinear algebraic equations for the time-evolved solution $\mathbf{U}^n$:

- Algorithms must have (nearly) linear space and time complexity on a sequential computer to extend toward large-scale problems.
- Algorithms must also scale (nearly) linearly with the number of processors on large parallel computers for use on next-generation petascale machines.

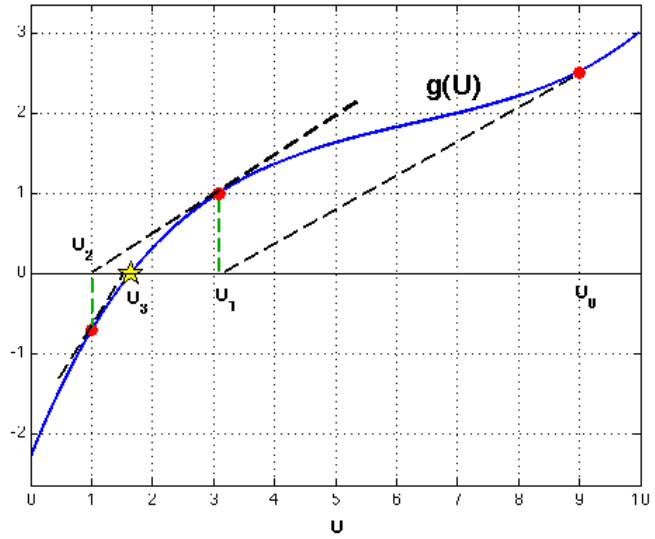
We use *globalized Inexact Newton* methods, which construct a sequence of iterates $\{\mathbf{U}_k\}$, $\mathbf{U}_k \rightarrow \mathbf{U}^n$, each solving a local linearized Newton system:

- Newton's method exhibits a convergence rate that is **independent** of spatial resolution in many PDE systems [Allgower et al., 1986; Weiser et al., 2005].
- Solution of the Newton linearization systems dominate spacetime complexity of the solution algorithm (all else has linear complexity).

Solution Approach: Inexact Newton method

Beginning with an explicitly-predicted initial guess $\mathbf{U}^n \approx \mathbf{U}_0 \in \text{span}\{\mathbf{U}^{n-i}\}$, solve $g(\mathbf{U}^n) = 0$ using a sequence of linearized solutions:

1. Given: $\mathbf{U}_k$
2. Find $\delta\mathbf{U}_k$ solving: $J(\mathbf{U}_k)\,\delta\mathbf{U}_k = -g(\mathbf{U}_k) + r$
3. Set: $\mathbf{U}_{k+1} = \mathbf{U}_k + \lambda\,\delta\mathbf{U}_k$
4. Check: $\|g(\mathbf{U}_{k+1})\| < \varepsilon$



- The Gâteaux derivative $J(\mathbf{U})\mathbf{V} := \lim_{\tau\to 0} \frac{d}{d\tau} g(\mathbf{U} + \tau\mathbf{V})$ may be approximated,

$$J(\mathbf{U})\mathbf{V} \approx \left[g(\mathbf{U} + \tau\mathbf{V}) - g(\mathbf{U})\right]/\tau, \quad 0 < \tau \ll 1.$$

- $\|\cdot\|$ is a 2-norm, weighted by relative magnitudes of solution components.

[Dembo et al., 1982; Dennis & Schnabel, 1983; Brown & Saad, 1990; . . .]

Solution Approach: Linear Subproblems

The subproblems are solved via a *Krylov subspace approximation algorithm* (GMRES). Constructs another sequence $\{\delta\mathbf{U}_{k,l}\}_l$, $\delta\mathbf{U}_{k,l} \to \delta\mathbf{U}_k$, where

$$\delta\mathbf{U}_{k,l} = \underset{\mathbf{V}\in K_l(J(\mathbf{U}_k),g(\mathbf{U}_k))}{\text{Argmin}} \|J(\mathbf{U}_k)\,\mathbf{V} + g(\mathbf{U}_k)\|_2,$$

with the enlarging approximation subspace defined by

$$K_l(J,g) = \text{span}\{g, Jg, J^2g, \ldots, J^lg\}.$$

- Iteration stops when $\|J(\mathbf{U}_k)\,\delta\mathbf{U}_{k,l} + g(\mathbf{U}_k)\|_2 = \|r\|_2 < \delta$.
- Requires only products with the linear operator $J(\mathbf{U}_k)\mathbf{V}$; we use the approximation $\tilde{J}(\mathbf{U}_k)\mathbf{V} = [g(\mathbf{U}+\tau\mathbf{V}) - g(\mathbf{U})]/\tau$.
- Convergence guaranteed for nonsingular $J(\mathbf{U})$; must store basis for $K_l(J,g)$.
- Convergence rate depends on the spectrum of $J(\mathbf{U})$: method minimizes the interpolating polynomial through approximate spectrum at each step.

Krylov: [Saad & Schultz, 1986; Greenbaum, 1997; Trefethen & Bau, 1997; . . .]

Implicit resistive MHD: [D.R., Samtaney & Woodward, 2006; Keyes, D.R. & Woodward, 2006]

General Comments on SUNDIALS Solvers

- The solver infrastructure only requires vector-space operations, hence is applicable to any spatial discretization method (AMR, FEM), etc.
- High-order BDF integration requires that previous solution vectors be defined on the same data so that linear combinations make sense. For spatially-adaptive approaches, we would be limited to BDF1 or BDF2.
- Due to directional derivative approximations, no Jacobian need be computed/stored, though the RHS functions $f(\mathbf{U})$ must be sufficiently differentiable.
- As SUNDIALS has no data-structure knowledge, preconditioning is not supplied and is the responsibility of the application.
- CVODE time integrator is usable from within PETSc.

Outline

I. MHD Description and Equations

II. Space-Time Discretization

III. Solution Approach

IV. Preconditioning Approach

V. Numerical Results

VI. Conclusions and Continuing Research

Implicit MHD Preconditioner Acceleration

Instead of solving the original Newton systems $J\,\delta\mathbf{U} = -g$, we may instead solve $(JP^{-1})(P\,\delta\mathbf{U}) = -g$ for some nonsingular operator P, effectively manipulating the spectrum of J to minimize the number of required Krylov iterations.

- P may then be any approximation to the linearized system that can be solved efficiently.
- The choice of P does not affect the accuracy of the nonlinear solution, only the convergence properties of the linear solver.
- P may be formed using problem-specific knowledge, enabling increased efficiency through physical insight.

Since MHD stiffness results from fast hyperbolic and diffusive effects, we set

$$P^{-1} = P_h^{-1} P_d^{-1} = J(\mathbf{U})^{-1} + \mathcal{O}(\Delta t^2).$$

Often termed *operator-splitting*, and widely used as a stand-alone solver, we use it to accelerate convergence of our more stable and accurate approach.

Ideal MHD Preconditioner (P_h)

P_h solves for the fastest wave effects within the implicit hyperbolic system. Denoting $(\cdot)$ as the location of action for the linear operator, this has Jacobian

$$\begin{aligned}
J_h(\mathbf{U}) &= I + \Delta t \left[\partial_x(J_x(\cdot)) + \partial_y(J_y(\cdot)) + \partial_z(J_z(\cdot))\right] \\
&= I + \Delta t \left[L_x^{-1} L_x \partial_x(J_x(\cdot)) + L_y^{-1} L_y \partial_y(J_y(\cdot)) + L_z^{-1} L_z \partial_z(J_z(\cdot))\right] \\
&= I + \Delta t \left[L_x^{-1} \partial_x(L_x J_x(\cdot)) - L_x^{-1} \partial_x(L_x) J_x \right. \\
&\qquad + L_y^{-1} \partial_y(L_y J_y(\cdot)) - L_y^{-1} \partial_y(L_y) J_y \\
&\qquad \left. + L_z^{-1} \partial_z(L_z J_z(\cdot)) - L_z^{-1} \partial_z(L_z) J_z\right]
\end{aligned}$$

We then form a directionally-split $\mathcal{O}(\Delta t^2)$ preconditioner as

$$\begin{aligned}
P_h &= P_x P_y P_z P_{\text{local}} \\
&= \left[I + \Delta t L_x^{-1} \partial_x(L_x J_x(\cdot))\right] \left[I + \Delta t L_y^{-1} \partial_y(L_y J_y(\cdot))\right] \left[I + \Delta t L_z^{-1} \partial_z(L_z J_z(\cdot))\right] \\
&\quad \left[I - \Delta t\, L_x^{-1} \partial_x(L_x) J_x - \Delta t\, L_y^{-1} \partial_y(L_y) J_y - \Delta t\, L_z^{-1} \partial_z(L_z) J_z\right].
\end{aligned}$$

Mapped Grid Adjustments

In the mapped-grid curvilinear case, we may additionally handle the local source terms with this approach:

$$
\begin{aligned}
J_h(\mathbf{U}) &= I + \Delta t \left[\partial_\xi(R J_{\tilde{\mathbf{F}}}(\cdot)) + \partial_\eta(R J_{\tilde{\mathbf{H}}}(\cdot)) + \mathcal{J}\partial_\varphi(J_{\mathbf{G}}(\cdot)) + J_{\tilde{\mathbf{S}}}\right] \\
&= I + \Delta t \left[L^{-1}_{\tilde{\mathbf{F}}} L_{\tilde{\mathbf{F}}} \partial_\xi(R J_{\tilde{\mathbf{F}}}(\cdot)) + L^{-1}_{\tilde{\mathbf{H}}} L_{\tilde{\mathbf{H}}} \partial_\eta(R J_{\tilde{\mathbf{H}}}(\cdot)) + \mathcal{J} L^{-1}_{\mathbf{G}} L_{\mathbf{G}} \partial_\varphi(J_{\mathbf{G}}(\cdot)) + J_{\tilde{\mathbf{S}}}\right] \\
&= I + \Delta t \Big[L^{-1}_{\tilde{\mathbf{F}}} \partial_\xi(R L_{\tilde{\mathbf{F}}} J_{\tilde{\mathbf{F}}}(\cdot)) - R L^{-1}_{\tilde{\mathbf{F}}} \partial_\xi(L_{\tilde{\mathbf{F}}}) J_{\tilde{\mathbf{F}}} \\
&\qquad + L^{-1}_{\tilde{\mathbf{H}}} \partial_\eta(R L_{\tilde{\mathbf{H}}} J_{\tilde{\mathbf{H}}}(\cdot)) - R L^{-1}_{\tilde{\mathbf{H}}} \partial_\eta(L_{\tilde{\mathbf{H}}}) J_{\tilde{\mathbf{H}}} \\
&\qquad + \mathcal{J} L^{-1}_{\mathbf{G}} \partial_\varphi(L_{\mathbf{G}} J_{\mathbf{G}}(\cdot)) - \mathcal{J} L^{-1}_{\mathbf{G}} \partial_\varphi(L_{\mathbf{G}}) J_{\mathbf{G}} + J_{\tilde{\mathbf{S}}}\Big]
\end{aligned}
$$

We again form our directionally-split $\mathcal{O}(\Delta t^2)$ preconditioner as

$$
\begin{aligned}
P_h &= P_\xi P_\eta P_\varphi P_{\text{local}} \\
&= \left[I + \Delta t L^{-1}_{\tilde{\mathbf{F}}} \partial_\xi(R L_{\tilde{\mathbf{F}}} J_{\tilde{\mathbf{F}}}(\cdot))\right] \left[I + \Delta t L^{-1}_{\tilde{\mathbf{H}}} \partial_\eta(R L_{\tilde{\mathbf{H}}} J_{\tilde{\mathbf{H}}}(\cdot))\right] \left[I + \Delta t \mathcal{J} L^{-1}_{\mathbf{G}} \partial_\varphi(L_{\mathbf{G}} J_{\mathbf{G}}(\cdot))\right] \\
&\quad \left[I - \Delta t\, R L^{-1}_{\tilde{\mathbf{F}}} \partial_\xi(L_{\tilde{\mathbf{F}}}) J_{\tilde{\mathbf{F}}} - \Delta t\, R L^{-1}_{\tilde{\mathbf{H}}} \partial_\eta(L_{\tilde{\mathbf{H}}}) J_{\tilde{\mathbf{H}}} - \Delta t\, \mathcal{J} L^{-1}_{\mathbf{G}} \partial_\varphi(L_{\mathbf{G}}) J_{\mathbf{G}} - \Delta t\, J_{\tilde{\mathbf{S}}}\right].
\end{aligned}
$$

Solving the Directional Systems

- L_i is the spatially-local left eigenvector matrix for J_i, i.e. at a given $x \in \Omega$,

$$L_i(x)J_i(x) = \Lambda_i(x)L_i(x), \quad \Lambda_i = \mathrm{Diag}(\lambda^1, \ldots, \lambda^8)$$

- These eigen-decompositions are already computed for upwind fluxes.
- The P_i systems may be decoupled into 1D wave equations along characteristics:

$$\left[I + \Delta t L_i^{-1} \partial_i (L_i J_i(\cdot))\right] \nu = \beta \quad \Leftrightarrow \quad L_i \left[I + \Delta t L_i^{-1} \partial_i (\Lambda_i L_i(\cdot))\right] \nu = L_i \beta$$

$$\Longleftrightarrow$$

$$\zeta + \Delta t\, \partial_i(\Lambda_i \zeta) = \eta \quad \Leftrightarrow \quad \zeta^l + \Delta t\, \partial_i(\lambda_i^l \zeta^l) = \eta^l, \;\; l = 1, \ldots, 8,$$

  where $\zeta = L_i \nu$ and $\eta = L_i \beta$.

- We need only solve for the fastest, stiffness-inducing, waves (fast magnetosonic, Alfvèn).
- These are solved to low-order ($\mathcal{O}(\Delta x^2)$) since used for preconditioning only.

Parallelism of Directional Solves

For each wave, we have a simple 1D tridiagonal system, spread over all processors in a spatial row. We parallelize the solve with a phased approach:

1. (local) Each processor solves its own portion of the system, assuming neighboring boundary values are zero.
2. (global) The processor-boundary unknowns are now decoupled from the interiors. This $(2p \times 2p)$ system is formed and solved on each processor.
3. (local) With the updated processor-boundary values, the local interior solutions are updated accordingly.

Benefits:

- Two sets of global communication: once to set up 1D MPI sub-communicators, once at each solve for global system.
- Non-iterative, i.e. 'true' solution is known at end of second phase.
- Easily generalizable to arbitrary boundary conditions.

[adapted from Arbenz & Gander, ETH Zürich, Technical Report, 1994]

Solving the P_{local} System

For spatially heterogeneous problems, or those involving local source terms, the P_{local} term is required:

$$\begin{aligned} P_{\text{local}} &= I + \Delta t \left[L_x^{-1} \partial_x \left(L_x \right) J_x + L_y^{-1} \partial_y \left(L_y \right) J_y + L_z^{-1} \partial_z \left(L_z \right) J_z + J_S \right] \\ &= I + \Delta t \left[L_x^{-1} \partial_x \left(L_x \right) L_x^{-1} \Lambda_x L_x \right. \\ &\qquad + L_y^{-1} \partial_y \left(L_y \right) L_y^{-1} \Lambda_y L_y \\ &\qquad \left. + L_z^{-1} \partial_z \left(L_z \right) L_z^{-1} \Lambda_z L_z + J_S \right] \end{aligned}$$

P_{local} may thus be constructed from available eigen-information. Since it has no spatial couplings on the unknown, the dense local matrices may be solved easily:

- Pre-compute the 8×8 block-matrices P_{local} at each spatial location
- Factorize each block $P_{\text{local}} = L_{\text{local}} U_{\text{local}}$
- Use this decomposition for fast solves at each Krylov iteration.

P_d: Diffusive MHD Preconditioner

P_d solves the remaining diffusive effects within the implicit system,

$$\partial_t \mathbf{U} - \nabla \cdot \mathbf{F}_v = 0.$$

We set P_d to be the Jacobian of this operator,

$$P_d = J_v(\mathbf{U}) = I - \Delta t \frac{\partial}{\partial \mathbf{U}} (\nabla \cdot \mathbf{F}_v)$$

$$= \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I - \Delta t D_{\rho \mathbf{v}} & 0 & 0 \\ 0 & 0 & I - \Delta t D_{\mathbf{B}} & 0 \\ -\Delta t L_\rho & -\Delta t L_{\rho \mathbf{v}} & -\Delta t L_{\mathbf{B}} & I - \Delta t D_e \end{bmatrix}$$

We may then exploit the lower-triangular structure of P_d to achieve an efficient and accurate solution.

P_d: Implementation Details

To solve $P_d\, y = b$ for $y = [y_\rho, y_{\rho\mathbf{v}}, y_\mathbf{B}, y_e]^T$:

1. Update $y_\rho = b_\rho$
2. Solve $(I - \Delta t D_{\rho\mathbf{v}})\, y_{\rho\mathbf{v}} = b_{\rho\mathbf{v}}$ for $y_{\rho\mathbf{v}}$
3. Solve $(I - \Delta t D_\mathbf{B})\, y_\mathbf{B} = b_\mathbf{B}$ for $y_\mathbf{B}$
4. Update $\tilde{b}_e = b_e + \Delta t\, (L_\rho\, y_\rho + L_{\rho\mathbf{v}}\, y_{\rho\mathbf{v}} + L_\mathbf{B}\, y_\mathbf{B})$
5. Solve $(I - \Delta t D_e)\, y_e = \tilde{b}_e$ for y_e.

- Due to their diffusive nature, steps 2, 3 and 5 are solved using a system-based geometric multigrid solver [HYPRE].
- Step 4 may be approximated through one finite-difference, instead of constructing and multiplying by the individual sub-matrices:

$$L_\rho\, y_\rho + L_{\rho\mathbf{v}}\, y_{\rho\mathbf{v}} + L_\mathbf{B}\, y_\mathbf{B} = \tfrac{1}{\sigma}\left[\nabla \cdot \mathbf{F}_v(U + \sigma W) - \nabla \cdot \mathbf{F}_v(\mathbf{U})\right]_e + O(\sigma),$$

where $W = [y_\rho, y_{\rho\mathbf{v}}, y_\mathbf{B}, 0]^T$.

Outline

I. MHD Description and Equations

II. Space-Time Discretization

III. Solution Approach

IV. Preconditioning Approach

V. Numerical Results

VI. Conclusions and Continuing Research

P_h Results – Linear Wave Advection

Ideal MHD linear wave advection test problem:

- 2.5D problem, slow wave propagation at 44.5^o to x-axis
- 50 time steps, fixed $\Delta t = C * \Delta t_{CFL}$, nonlinear tolerance $\varepsilon = 10^{-7}$
- Preconditioning: none [N], 8-wave [FW], block-split [BT]

Mesh	C	CPU[N]	CPU[BT]	CPU[FW]	Krylov[N]	Krylov[BT]	Krylov[FW]
64^2	50	14.75	52.31	17.31	620	50	50
	100	26.27	78.29	15.59	1226	111	50
	500	31.10	593.19	285.00	1531	1283	5146
128^2	50	56.17	227.00	64.12	661	50	50
	100	100.46	364.89	64.87	1254	120	50
	500	422.11	1941.88	599.23	4729	927	2482
256^2	50	307.12	873.00	278.93	618	50	56
	100	661.75	1333.38	274.23	1409	113	50
	500	2951.58	7692.26	1880.34	6209	966	1701
512^2	50	1285.05	3719.43	991.43	608	50	50
	100	2765.79	6278.70	1000.86	1265	133	50
	500	14791.33	35547.76	1009.03	6444	1055	56

P_h Results – Weak Parallel Scaling

Problem size increases with processor count, [procs in ()]; $\varepsilon = 10^{-7}$.
Preconditioning: none [N], formulation 0 [FW0], formulation 1 [FW1]

Mesh	C	Newt[N]	Newt[FW0]	Newt[FW1]	Kry[N]	Kry[FW0]	Kry[FW1]
128^2	50	50	50	50	708	50	50
(1)	100	50	50	50	1307	50	50
	500	50	65	59	3186	5103	3910
256^2	50	50	50	50	619	50	50
(4)	100	50	50	50	1386	50	50
	500	79	50	50	7682	1095	713
512^2	50	50	50	50	626	50	50
(16)	100	50	50	50	1279	50	50
	500	79	50	50	7473	1568	741
1024^2	50	50	50	50	635	50	50
(64)	100	50	50	50	1309	50	50
	500	73	52	50	6817	2153	75
2048^2	50	50	50	50	634	50	50
(256)	100	50	50	50	1275	50	50
	500	72	50	53	6742	1922	2768
4096^2	50	50	50	50	645	50	50
(1024)	100	50	50	50	1260	50	50
	500	70	50	50	6590	1442	165

P_h Results – Kelvin Helmholtz

Ideal MHD Kelvin Helmholtz test problem:

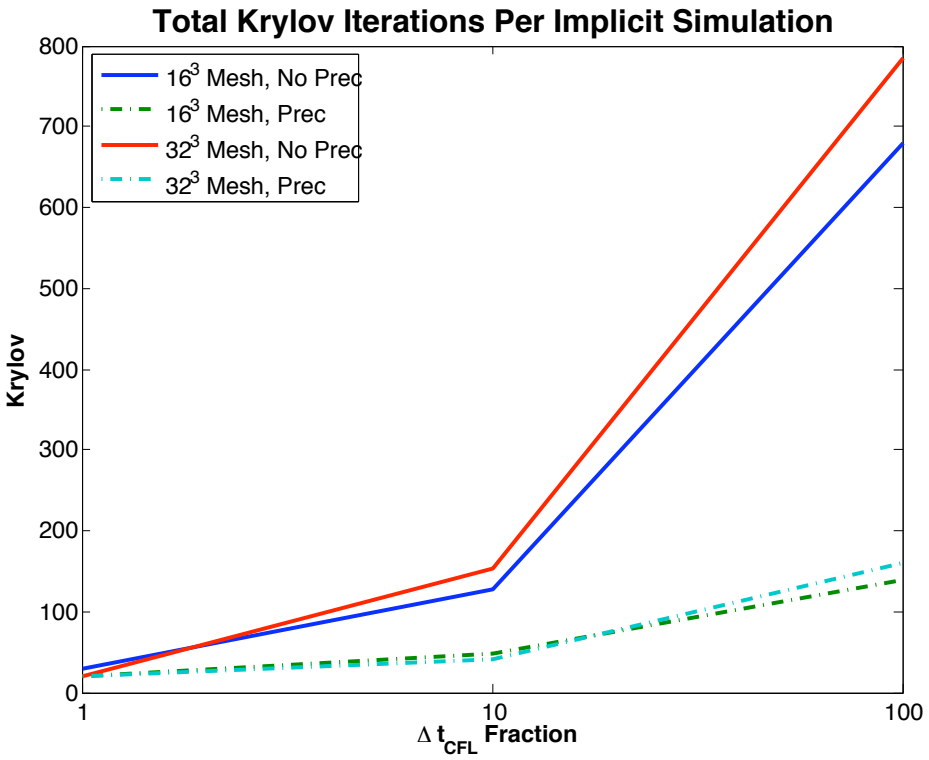
- 3D problem, initial perturbation 10^{-3}
- 50 time steps, fixed $\Delta t = C * \Delta t_{CFL}$, $\varepsilon = 10^{-7}$
- Preconditioning: none [N], 8-wave [FW]

Mesh	C	CPU[N]	CPU[FW]	Newton[N]	Newton[FW]	Krylov[N]	Krylov[FW]
$16 \times 8 \times 8$	1	5.70	8.76	150	100	300	199
	5	9.36	15.64	150	150	756	435
	10	14.29	22.64	157	156	1369	690
$32 \times 16 \times 16$	1	23.77	56.07	120	100	240	190
	5	52.19	92.30	150	150	767	425
	10	74.28	125.63	150	150	1439	633
$64 \times 32 \times 32$	1	111.20	309.24	100	100	200	174
	5	258.62	589.83	148	149	730	408
	10	427.93	796.38	150	150	1346	618

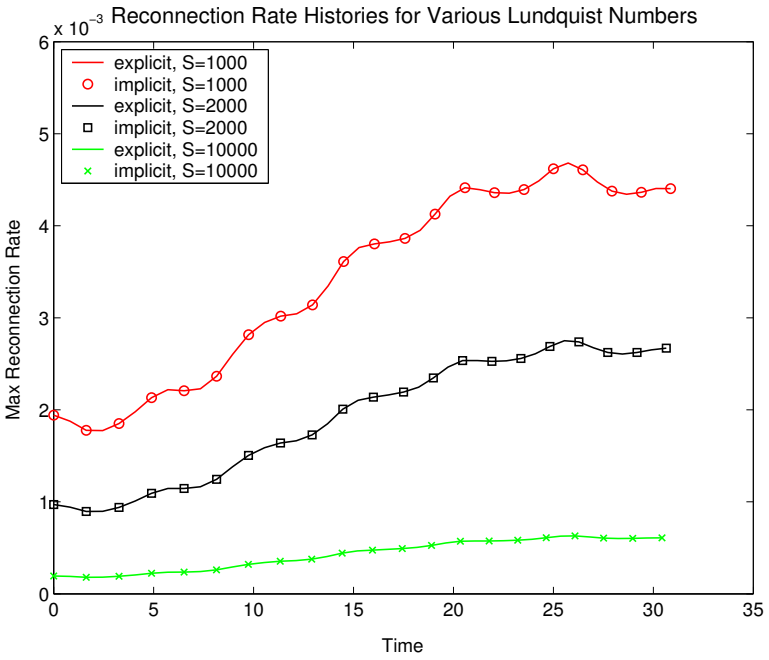
Initial P_d Results – Diffusion Dominated

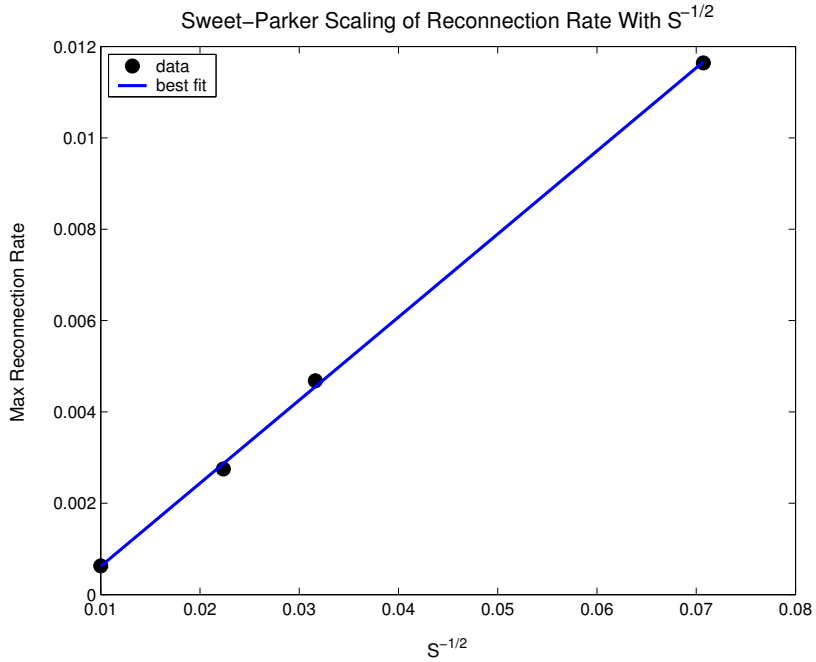
- 3D resistive MHD pellet-injection test problem ($\eta = 1$)
- 50 time steps, fixed $\Delta t = C * \Delta t_{CFL}$
- Preconditioning: none [N], Diffusive [P_d]

Mesh	C	Newton [N, P_d]	Krylov [N, P_d]
16^3	1	21, 20	30, 20
16^3	10	30, 40	128, 49
16^3	100	34, 45	680, 140
32^3	1	20, 20	20, 20
32^3	10	30, 40	153, 40
32^3	100	35, 51	785, 160



Reconnection Results ($P = I$)



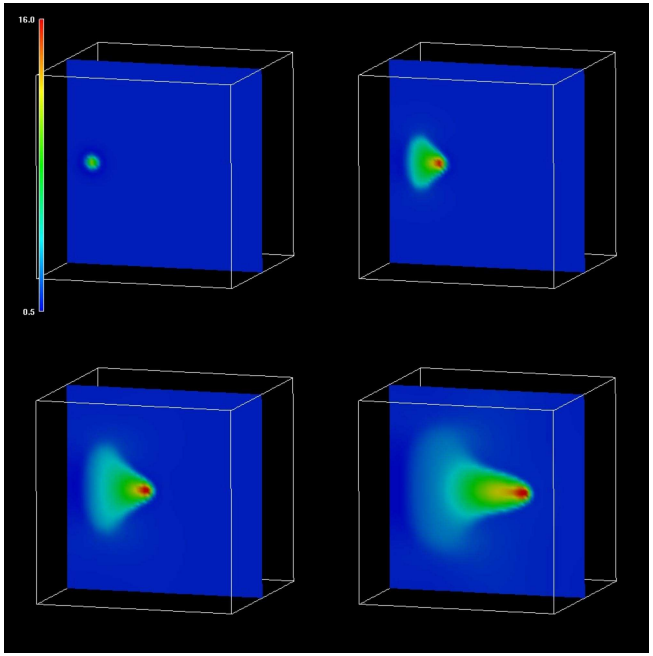


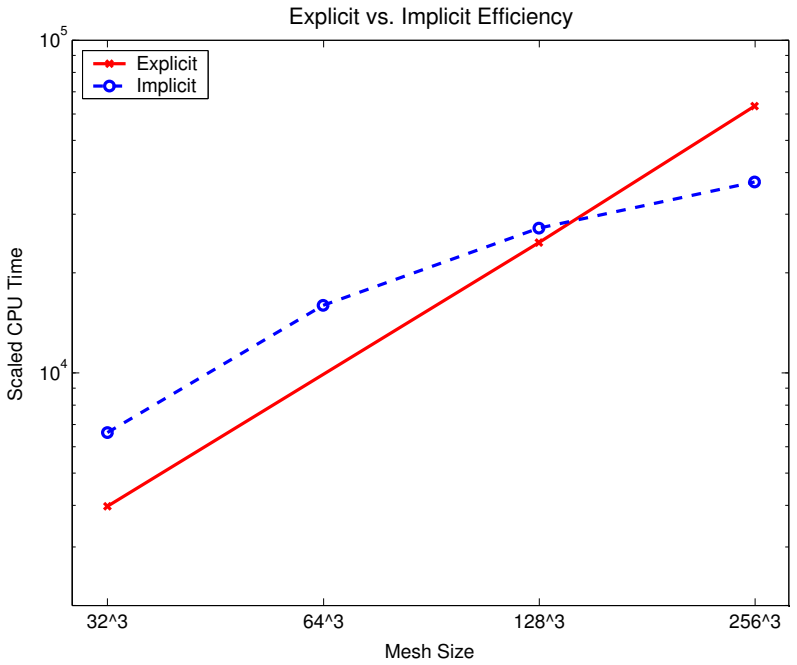
Implicit method produces identical physical results as explicit, captures theoretical scaling properties.

mesh size	Exp. CPU Time	Exp Nt	Exp $\overline{\Delta t}$	Imp. CPU Time	Imp Nt	Imp $\overline{\Delta t}$
64×32	75 s.	1636	3.06e-2	47 s.	1144	4.37e-2
128×64	768 s.	3247	1.54e-2	285 s.	1158	4.32e-2
256×128	8214 s.	6493	7.70e-3	1817 s.	1075	4.65e-2
512×256	80348 s.	12985	3.85e-3	14203 s.	1473	3.39e-2

Explicit and implicit CPU times, time steps, step sizes to reach $t = 50$. (CVODE takes conservative Δt)

Pellet Injection Results ($P = I$)





Pellet injection/ablation; Explicit, Implicit scaled CPU times.

mesh size (procs)	Exp. CPU Time	Exp Nt	Exp $\overline{\Delta t}$	Imp. CPU Time	Imp Nt	Imp $\overline{\Delta t}$
32^3 (1)	4198 s.	2844	1.05e-3	7168 s.	6221	4.82e-4
64^3 (8)	9136 s.	4886	6.14e-4	16520 s.	6467	4.64e-4
128^3 (64)	23136 s.	8995	3.34e-4	28598 s.	8979	3.34e-4
256^3 (256)	49507 s.	17619	1.70e-4	40842 s.	9725	3.08e-4

Explicit and implicit CPU times, time steps, step sizes to reach $t = 3$.

Conclusions

- For MHD systems dominated by hyperbolic stiffness from fast wave effects, we have developed a preconditioning approach that:
  - uses characteristic info. otherwise used only within upwind methods,
  - allows preconditioning of any combination of MHD waves,
  - is fully parallel, requiring minimal communication per P_h^{-1} solve.
- For MHD systems dominated by diffusive stiffness, we have a multigrid-based preconditioning approach that:
  - uses scalable solver technology for diffusive problems,
  - is easily extensible to highly-anisotropic heat conduction (AMG).
- Most problems exhibit both of these effects, so they may be combined in an operator-split fashion.

Continuing Research

- Investigate coupled operator-split preconditioning approach on resistive MHD problems of balanced type (advection & diffusion)
- Extend preconditioners to mapped curvilinear grids (through P_{local} term), allowing for implicit finite-volume simulations of toroidal fusion devices.
- Investigate approaches for constrained implicit evolution of finite-volume MHD equations (direct enforcement of $\nabla \cdot \mathbf{B} = 0$)
- Compare operator split preconditioner with other approaches for fully implicit evolution of compressible MHD (in collab. w/ Chacon).

Thanks and Acknowledgements

- U.S. Department of Energy, Scientific Discovery through Advanced Computing Program. Specifically:
  - Towards Optimal Petascale Simulations (TOPS),
  - Applied Partial Differential Equations (APDEC),
  - Center for Extended MHD Modeling (CEMM)

- UCSD Mathematics and Physics Departments

- Princeton Plasma Physics Laboratory

- Lawrence Livermore National Lab:
  - Nonlinear Solvers and Differential Equations Project
  - Scalable Linear Solvers Project