Electromagnetic solvers in 2D and 3D

Ben Dudson (UoY),
John Omotani (CCFE), Luke Easy (CCFE/UoY), Brendan Shanahan (UoY)

York Plasma Institute, Department of Physics,
University of York, Heslington, York YO10 5DD, UK

BOUT++ Workshop

16th September 2014
Potential fields in reduced MHD models
2D direct solvers used in BOUT and BOUT++
2D iterative solvers in BOUT++ (using PETSc)
Issues with 2D solvers

3D solvers and preliminary results
Some different approaches

Discussion
Drift-reduced fluid models usually formulated in terms of a vorticity

- Fluid velocity assumed to have form $v = v_\parallel b + \text{drifts}$
- Rather than evolving $v$, solve for $v_\parallel$ and a scalar vorticity $\omega = b \cdot \nabla \times (m_i n_i v)$
- From either momentum or charge conservation $\nabla \cdot J = 0$:

$$\frac{\partial}{\partial t} \nabla \cdot \left( \frac{m_i n_i}{B^2} \nabla_\perp \phi \right) = \nabla \cdot (J_\parallel b) + \nabla \cdot J_{\text{dia}} + \text{Higher order terms}$$

with $\nabla_\perp = \nabla - bb \cdot \nabla$

- Inverting the operator $\nabla \cdot \left( \frac{m_i n_i}{B^2} \nabla_\perp \phi \right)$ to obtain the potential $\phi$ is a major part of the complexity and computational expense in a drift-reduced fluid simulation.
In curvilinear coordinates the operator to be inverted is

\[ \nabla \cdot \left( \frac{m_i n}{B^2} \nabla_{\perp} \phi \right) = \frac{1}{J} \frac{\partial}{\partial u^i} \left( J \frac{m_i n}{B^2} g^{ij} (\nabla_{\perp} \phi)_j \right) \]

- The Clebsch coordinate system \( \mathbf{B} = \nabla \psi \times \nabla \alpha \) used in BOUT++ is non-orthogonal, since \( \alpha = \) toroidal angle.
- This enables FFTs to be used, but \( g^{ij} (\nabla_{\perp} \phi)_j \neq 0 \) along \( \mathbf{B} \) direction.
- Drift ordering \( k_{||} \ll k_{\perp} \) is usually used to drop derivatives along \( \mathbf{B} \).
- This reduces the number of dimensions to 2, reducing the computational difficulty.
The standard BOUT++ coordinate system (inherited from BOUT)\(^1\) uses toroidal angle $\zeta$ as one of its coordinates:

\[
\begin{aligned}
x &= \psi - \psi_0 \\
y &= \theta \\
z &= \zeta - \int_{\theta_0}^{\theta} \nu (\psi, \theta) \, d\theta
\end{aligned}
\]

with $\nu (\psi, \theta) = \frac{\mathbf{B} \cdot \nabla \zeta}{\mathbf{B} \cdot \nabla \theta}$ is the local field-line pitch.

- In these coordinates equilibrium quantities and metric components are constant, so Fourier transforms can be used
- **But:** Only if we assume that the coefficient is constant in $\zeta$

\[
\frac{1}{J} \frac{\partial}{\partial u^i} \left( J m_i n \frac{g_{ij}}{B^2} (\nabla \perp \phi)_j \right)
\]

Commonly called the **Boussinesq approximation**

\(^1\)See coordinates manual for details
In the Laplacian class implementations, the operator is expanded in a non-conservative form:

\[ \nabla \cdot (\alpha \nabla_\perp \phi) = \omega \rightarrow \nabla_\perp^2 \phi + \frac{1}{\alpha} \nabla_\perp \alpha \cdot \nabla_\perp \phi = \omega / \alpha \]

which is solved by setting coefficients:

\[ D \nabla_\perp^2 x + \frac{1}{C} \nabla_\perp C \cdot \nabla_\perp x = b \]

**Note:** If any coefficients depend on \( z (\zeta) \) then they are averaged
The Laplacian operator can be written in terms of $\psi$ derivatives as:

$$
\nabla^2_{\perp} = (RB_\theta)^2 \left[ \frac{\partial^2}{\partial \psi^2} + \frac{B^2}{(RB_\theta)^4} \frac{\partial^2}{\partial z^2} \right] + \frac{1}{J} \frac{\partial}{\partial \psi} \left[ J (RB_\theta)^2 \right] \frac{\partial}{\partial \psi} - \frac{1}{J} \frac{\partial}{\partial y} \left( \frac{B_\zeta}{B^2_\theta R} \right) \frac{\partial}{\partial z}
$$

Taking Fourier transforms in $z$,

$$
\frac{\partial}{\partial z} \rightarrow -ik_z
$$

- For each toroidal mode $k_z$, these equations reduce to a second order equation in $\psi$ (or $x$).
- These can be solved independently using efficient algorithms.
The 1D equations in $x$ are discretised using a 3-point stencil
- Tridiagonal system of equations

Boundary conditions need to be set on inner and outer $x$
- Zero value (the default)
- Zero gradient
- Decaying Laplacian approximation
- Cylindrical boundary condition
- ...

See include/invert_laplace.hxx
Set using a system of flags

phiSolver->setInnerBoundaryFlags(INVERT_DC_GRAD);
phiSolver->setOuterBoundaryFlags(INVERT_AC_GRAD);

Note the distinction between DC ($k_z = 0$) and AC ($k_z \neq 0$) components: $k_z = 0$ is a special case due to gauge invariance
The Boussinesq approximation can lead to non-conserved energy

- Solving equations for the shear Alfvén wave: Vorticity $\omega$ and electromagnetic potential $A_\parallel$, with auxiliary equations for the electrostatic potential $\phi$ and parallel current $j_\parallel = b_0 \cdot j$:

\[
\frac{\partial \omega}{\partial t} = \nabla \cdot (b_0 j_\parallel) \quad \frac{\partial A_\parallel}{\partial t} = -b_0 \cdot \nabla \phi
\]

\[
\omega = \nabla \cdot \left( \frac{m_i n}{B^2} \nabla_\perp \phi \right) \quad \nabla^2 A_\parallel = -\mu_0 j_\parallel
\]

- This has a conserved energy

\[
E = \frac{1}{2} \int dV \left[ \frac{m_i n}{B^2} |\nabla_\perp \phi|^2 + \frac{1}{\mu_0} |\nabla_\perp A_\parallel|^2 \right]
\]
Making the approximation

\[ \nabla \cdot \left( \frac{m_i n}{B^2} \nabla_{\perp} \phi \right) \simeq \nabla \cdot \left( \frac{m_i n_0}{B^2} \nabla_{\perp} \phi \right) \]

Axisymmetric, constant

modifies the conserved energy, but the approximation

\[ \nabla \cdot \left( \frac{m_i n}{B^2} \nabla_{\perp} \phi \right) \simeq n \nabla \cdot \left( \frac{m_i}{B^2} \nabla_{\perp} \phi \right) \]

introduces an energy source

\[ \frac{dE}{dt} = \int dV \left[ \phi \frac{m_i}{B^2} \frac{\partial \nabla_{\perp} \phi}{\partial t} \cdot \nabla n \right] \]

→ Would like to remove the Boussinesq approximation
BOUT++ can use the PETSc library to solve these equations.

Contains a number of iterative schemes e.g. CG, GMRES, ...

To find a solution efficiently, a preconditioner is needed.

See talk at 2013 workshop: bout2013.llnl.gov

Examples: examples/blob2d

Need to compile and configure with PETSc

```
./configure --with-petsc
```
Boundary conditions: A separate boundary condition is imposed on each $x – z$ plane ($\psi – \zeta$ in most simulations). This may over-constrain the problem.
Issues with 2D solvers

1. **Boundary conditions**: A separate boundary condition is imposed on each $x - z$ plane ($\psi - \zeta$ in most simulations). This may over-constrain the problem.

2. **Difficulty with $m = 0$ modes**: If $z$ is toroidal angle, then the $y$ (parallel; poloidal) derivatives cannot be neglected for the $n = 0, m = 1$ mode.
   - Becomes particularly problematic in X-point geometry
   - Switching to using planes in $\psi - \theta$ helps, but loses the ability to take Fourier transforms and reduce to 1-D problem.
The standard method previously used has problems solving for global-scale electric fields (e.g. $n = 0$) in X-point geometry.
The standard method previously used has problems solving for global-scale electric fields (e.g. $n = 0$) in X-point geometry.

Neglect of “small” quantities can lead to unphysical solutions.
The standard method previously used has problems solving for global-scale electric fields (e.g. \( n = 0 \)) in X-point geometry.

Neglect of “small” quantities can lead to unphysical solutions.

Efficient methods implemented to solve full 3D problem.

Should enable more realistic simulation of global X-point geometry.
Discussion

A big problem for all 3D simulations is **Fast timescales**: Parallel electron dynamics tends to make timesteps very small (≪ ion cyclotron time).

- How to include parallel Ohm’s law, but remove these timescales?
- Preconditioning complicated by mixing of $k_\perp$ and $k_\parallel$.
  - → Multigrid methods?
- P. Tamain: Combining Ohm’s law and Vorticity equation into 3D equation for $\phi$
- Is there a better way?