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

Outline

- 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

Potential fields in reduced MHD

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

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

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

  with $\nabla_\perp = \nabla - \mathbf{bb} \cdot \nabla$

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

2D inversion

In curvilinear coordinates the operator to be inverted is

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

- 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 $g^{ij} \left(\nabla_\perp \phi \right)_j \neq 0$ along $\mathbf{B}$ direction.
- Drift ordering $k_\parallel \ll k_\perp$ is usually used to drop derivatives along $\mathbf{B}$.
- This reduces the number of dimensions to 2, reducing the computational difficulty.

2D inversion using FFTs

The standard BOUT++ coordinate system (inherited from BOUT)[1] uses toroidal angle ζ as one of its coordinates:

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

with $\nu\left(\psi, \theta\right) = \dfrac{\mathbf{B} \cdot \nabla\zeta}{\mathbf{B} \cdot \nabla\theta}$ 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 ζ

$$\frac{1}{J}\frac{\partial}{\partial u^i}\left(\underbrace{J\frac{m_i n}{B^2} g^{ij}}_{\text{Constant in } \zeta} \left(\nabla_\perp \phi\right)_j \right)$$

Commonly called the **Boussinesq approximation**

[1] See coordinates manual for details

2D inversion using FFTs

In the `Laplacian` class implementations, the operator is expanded in a non-conservative form:

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

which is solved by setting coefficients:

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

```
Laplacian* phiSolver = Laplacian::create();

phiSolver->setCoefD(1.0); // This is the default
phiSolver->setCoefC(alpha);

Field3D phi = phiSolver->solve(omega / alpha);
```

Note: If any coefficients depend on z (ζ) then they are averaged

2D inversion using FFTs

The Laplacian operator can be written in terms of ψ derivatives as:

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

Taking Foutier 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 ψ (or x).
- These can be solved independently using efficient algorithms

2D inversion using FFTs: Implementation

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

Energy conservation

The Boussinesq approximation can lead to non-conserved energy

- Solving equations for the shear Alfvén wave: Vorticity ω and electromagnetic potential $A_{||}$, with auxiliary equations for the electrostatic potential ϕ and parallel current $j_{||} = \mathbf{b}_0 \cdot \mathbf{j}$:

$$\frac{\partial \omega}{\partial t} = \nabla \cdot (\mathbf{b}_0 j_{||}) \qquad \frac{\partial A_{||}}{\partial t} = -\mathbf{b}_0 \cdot \nabla \phi$$

$$\omega = \nabla \cdot \left(\frac{m_i n}{B^2} \nabla_\perp \phi \right) \qquad \nabla_\perp^2 A_{||} = -\mu_0 j_{||}$$

- This has a conserved energy

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

Energy conservation

Making the approximation

$$\nabla \cdot \left(\frac{m_i n}{B^2} \nabla_\perp \phi \right) \simeq \nabla \cdot \Bigg(\underbrace{\frac{m_i n_0}{B^2}}_{\text{Axisymmetric, constant}} \nabla_\perp \phi \Bigg)$$

modifies the conserved energy, but the approximation

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

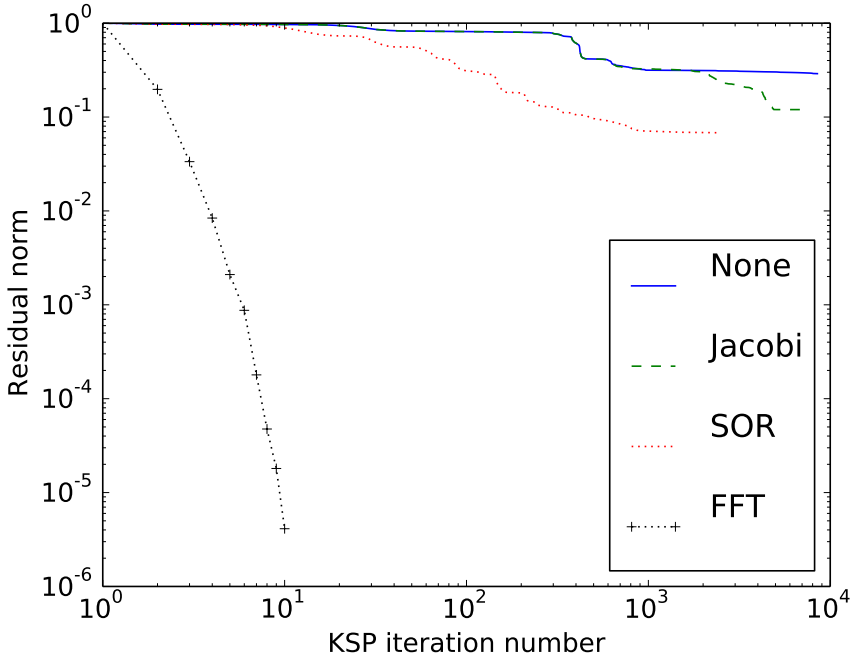
introduces an energy source

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

$\rightarrow$ Would like to remove the Boussinesq approximation

2D inversion using PETSc

- 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
```

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

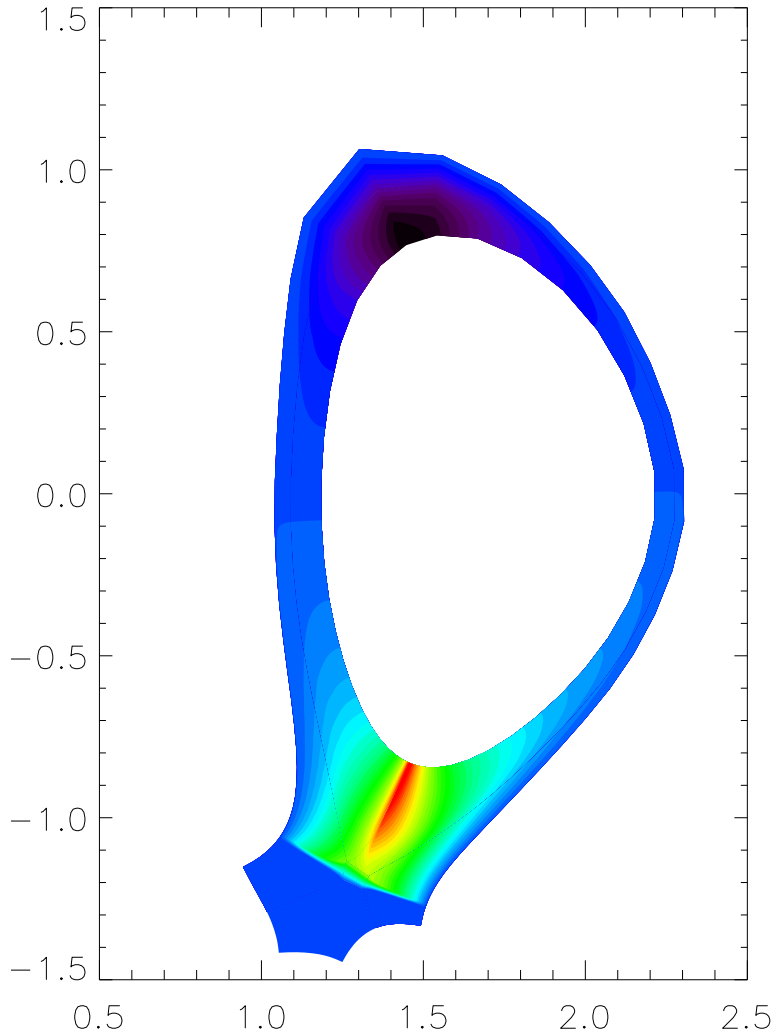
Issues with 2D solvers

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.

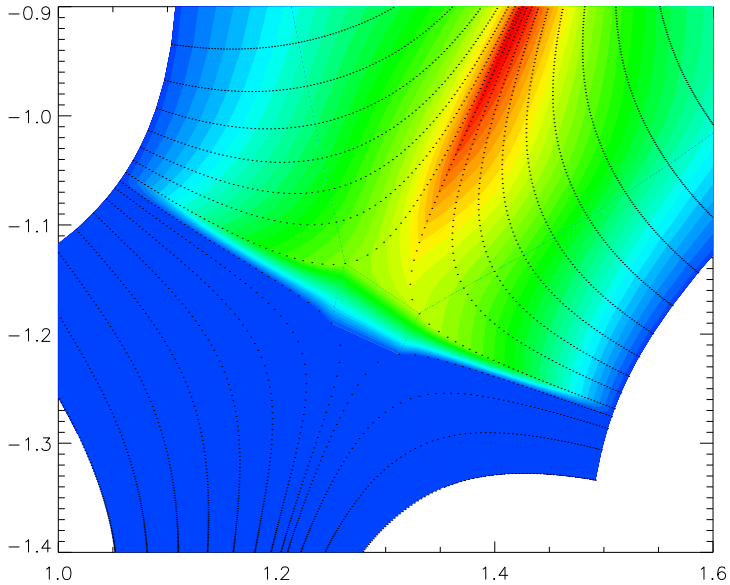
Numerical methods - Global X-point geometry

- The standard method previously used has problems solving for global-scale electric fields (e.g. $n = 0$) in X-point geometry



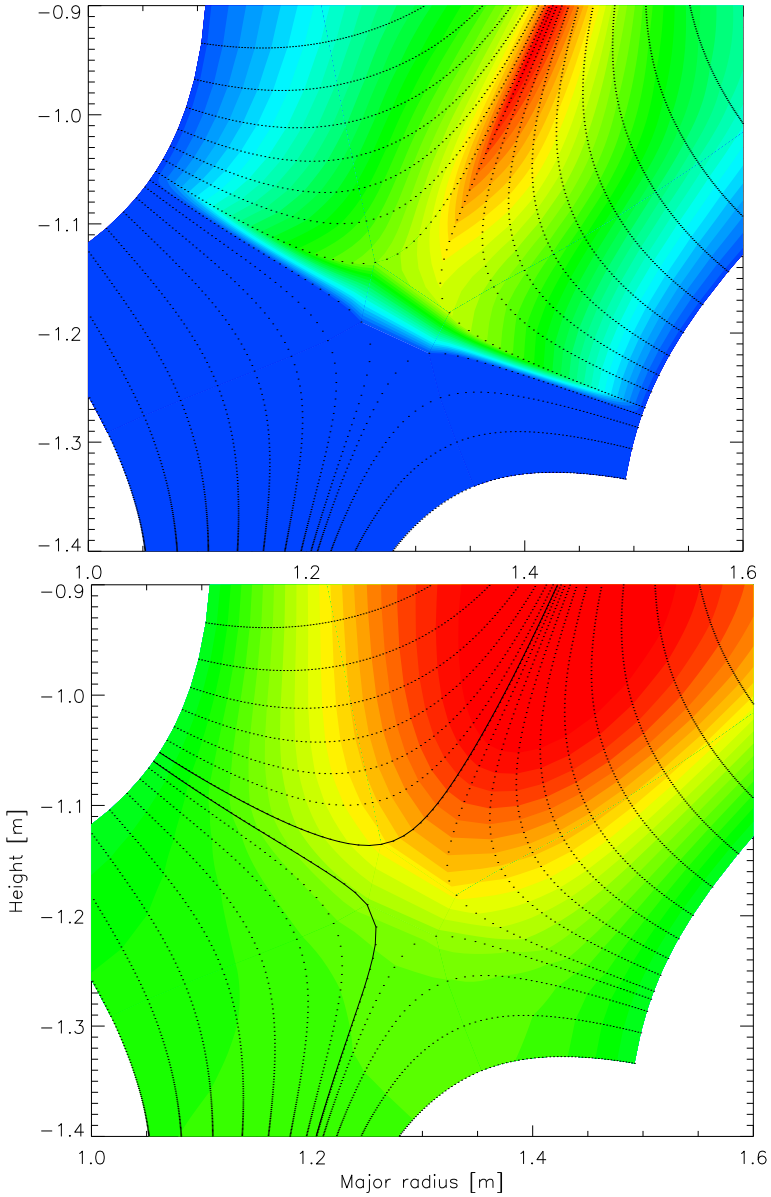
Numerical methods - Global X-point geometry

- 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



Numerical methods - Global 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
- 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 ($\ll$ ion cyclotron time).

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