Magnetic induction and electric potential solvers for incompressible MHD flows

Alessandro Tassone

DIAEE - Nuclear Section,
Sapienza Università di Roma,
Rome, Italy

05/12/2016
Fusion reactors

Liquid metal blankets are the leading candidate for tritium production in MCF reactors. Interaction between LM and magnetic field cause transition to magnetohydrodynamic flow.

**MHD-related issues**
- High pressure drops
- Enhanced corrosion rates
- Turbulence suppression
- etc.

Figure: ITER experiment
To support the blanket design a CFD software able to model MHD flows is needed.

**Required parameters**

- $M = 10^4$
- $Re = 10^4$
- $Gr = 10^{12}$

No mature CMHD code is currently available.

**Figure:** MHD calculations progress (Smolentsev, 2015)
MHD governing equations

A laminar, isotherm and incompressible flow flow is assumed

\[ \nabla \cdot \mathbf{u} = 0 \]  \hspace{1cm} (1)

\[ \frac{D\mathbf{u}}{Dt} = -\nabla \left( \frac{p}{\rho} \right) + \nu \nabla^2 \mathbf{u} + \frac{\mathbf{J} \times \mathbf{B}}{\rho} \]  \hspace{1cm} (2)

\[ \frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{u} \times \mathbf{B}) + \frac{1}{\mu \sigma} \nabla^2 \mathbf{B} \]  \hspace{1cm} (3)

\[ \mathbf{J} = \frac{1}{\mu} \nabla \times \mathbf{B} \]  \hspace{1cm} (4)

This set is called the \textit{B-formulation} of the MHD governing equations
Inductionless approximation

The equation (3) can be simplified, reducing the u-B coupling non-linearity, if the self-induced magnetic field is negligible. This corresponds to the \textit{inductionless} condition

\[ R_m \ll 1 \]

The parameter \( R_m = u_0 L / \mu \sigma \) is called the magnetic Reynolds number. For the typical values encountered in LM flows the condition is valid and the magnetic field can be uncoupled from the fluid velocity, i.e. it depends just from the boundary conditions.
Electric potential formulation

A Poisson equation for the electric potential and the Ohm’s law substitute (3) and (4)

\[ \nabla \cdot \mathbf{u} = 0 \quad (1) \]

\[ \frac{D\mathbf{u}}{Dt} = -\nabla \left( \frac{p}{\rho} \right) + \nu \nabla^2 \mathbf{u} + \left( \mathbf{J} \times \mathbf{B} \right)/\rho \quad (2) \]

\[ \nabla^2 \phi = \nabla \cdot \left( \mathbf{u} \times \mathbf{B} \right) \quad (5) \]

\[ \mathbf{J} = \sigma \left( -\nabla \phi + \mathbf{u} \times \mathbf{B} \right) \quad (6) \]

The new set is called the \( \phi \)-formulation of the MHD equations.
Parameters for incompressible LM MHD flow

**Hartmann number:** adimensional measure of the magnetic field intensity

\[ M = BL \sqrt{\frac{\sigma}{\rho \nu}} \]  

(7)

**Wall conductance ratio:** measures relative electrical conductivity of the wall compared to the fluid

\[ c = \frac{\sigma_w}{\sigma} \frac{t}{L} \]  

(8)

Others: Reynolds number (Re), Rm, Grashof number (Gr, buoyant flows), interaction parameter (N)
The mhdFoam solver

A magnetic induction solver

Numerical constrains to ensure

\[ \nabla \cdot (B_0 + b)^{t+1} = 0 \]

Employ arbitrary scheme to "project" calculated field to very close divergence-free one

Analogy

\[ \nabla \cdot u = 0 \quad \nabla \cdot B = 0 \]

Application

PISO(u, p) \rightarrow BPISO(B, pB)

Figure: Solenoid magnetic field
The mhdFoam solver

How to find mhdFoam source code

A solver based on the magnetic induction formulation is available in the OpenFOAM-4.0 distribution. To access the source code

```bash
cd $FOAM_SOLVERS/electromagnetics/mhdFoam
ls
gedit mhdFoam.C
```
The mhdFoam solver

Momentum equation - Line 78 to 90

```cpp
fvVectorMatrix UEqn
(
    fvm::ddt(U)
    + fvm::div(phi, U)
    - fvc::div(phiB, 2.0*DBU*B)
    - fvm::laplacian(nu, U)
    + fvc::grad(DBU*magSqr(B))
);

if (piso.momentumPredictor())
{
    solve(UEqn == -fvc::grad(p));
}
```

Lorentz force term is defined implicitly

UEqn solved to obtain velocity field prediction for following PISO
The mhdFoam solver

Predictor/corrector - from Line 133 to 155

Remember from (3)?

\[
\text{phiB} = \text{fvc::flux}(B) + \text{rABf*phiB}\]

while (bpiso.correct())
{
fvVectorMatrix BEqn
(
 fvm::ddt(B)
 + fvm::div(phi, B)
 - fvm::div(phiB, U)
 - fvm::laplacian(DB, B)
);

BEqn.solve();

Flux coefficients are calculated

fvScalarMatrix pBEqn
(
 fvm::laplacian(rABf, pB)
 == fvc::div(phiB)
);

Poisson equation for pB
The mhdFoam solver

Projection scheme - from Lines 159 to 165

Magnetic flux coefficients are updated

```
if (bpiso.finalNonOrthogonalIter())
{
    phiB -= pBEqn.flux();
}
```

Evaluate magnetic flux divergence error (from `magneticFieldErr.H`)

```
mag(fvc::div(phiB))().weightedAverage(mesh.V()).value()
```
The mhdFoam solver

The hartmann tutorial

Case parameters

- $M = 20$
- $Re = 2$
- $c = 0$
- $B_{|_{wall}} = B_0$

Available in

$FOAM_TUTORIALS$
/electromagnetics
/mhdFoam/hartmann

Figure: Hartmann problem
The mhdFoam solver

Comparison of numerical results against theory for M=20

-1 0 1
0 0.2 0.4 0.6 0.8 1.0 1.2
OF Analytical
The mhdFoam solver

M influence on the flow

![Graph showing the influence of M on the flow]

- M=0
- M=10
- M=20
- M=50
An electric potential solver

Why employ a solver based on the \( \phi \)-formulation?

**Advantages**
- Faster
- More stable
- Simpler boundary conditions
- More accurate for coarse mesh

**Drawbacks**
- Requirement on \( \mathbf{J} \) interpolation
- Nonconservative treatment of Lorentz force
- Worse convergence behavior
- Constrain on charge conservation \( \nabla \cdot \mathbf{J} = 0 \)
A solver based on the electric potential formulation was developed based on previous work by Francesco Ferroni of Imperial College. It is constructed starting from the default icoFoam. To have a look at the source code download epotFoam.tar.gz

```
tar xzf epotFoam.tar.gz
rm epotFoam.tar.gz
cd epotFoam
gedit epotFoam.C
```
The epotFoam solver

Pre-PISO - Line 57 and from Line 67 to 78

Lorentz force initialization

```
volVectorField
lorentz = sigma *
(-fvc::grad(PotE) ^ B0)
+ sigma * ((U ^ B0) ^ B0);
```

Lorentz force is defined explicitly in momentum predictor for PISO loop

```
fvVectorMatrix UEqn
(
  fvm::ddt(U)
  + fvm::div(phi, U)
  - fvm::laplacian(nu, U)
  - (1.0/rho) * lorentz
);

if (piso.momentumPredictor())
{
  solve(UEqn == -fvc::grad(p));
}
```
After the PISO loop corrected \( u \) field is employed to estimate \( J \):

\[
surfaceScalarField \ psi_{ib} = fvc::interpolate(U ^ \ ^B0) \\
& mesh.Sf();
\]

Evaluation of cross product \( u \times B \):

\[
fvScalarMatrix \ PotEEqn \\
( \\
  fvm::laplacian(PotE) \\
  == \\
  fvc::div(psi_{ib}) \\
); \\
PotEEqn.setReference( \\
PotERefCell, PotERefValue); \\
PotEEqn.solve();
\]

Poisson equation for electric potential:

\[
fvScalarMatrix \ PotEEqn \\
( \\
  fvm::laplacian(PotE) \\
  == \\
  fvc::div(psi_{ib}) \\
); \\
PotEEqn.setReference( \\
PotERefCell, PotERefValue); \\
PotEEqn.solve();
\]
Interpolation scheme for current density

```plaintext
surfaceScalarField jn = -(fvc::snGrad(PotE) * mesh.magSf()) + psiub;

surfaceVectorField jnv = jn * mesh.Cf();

volVectorField jfinal = fvc::surfaceIntegrate(jnv) -(fvc::surfaceIntegrate(jn) * mesh.C());
```
Current density distribution is updated and employed to calculate Lorentz force

\[
\text{jfinal.correctBoundaryConditions();}
\]

\[
\text{lorentz} = \sigma \times (\text{jfinal} \wedge \text{B0});
\]
The epotFoam solver

Compile epotFoam

Move the epotFoam folder in the user directory and compile it

```
mkdir -p $WM_PROJECT_USER_DIR/applications/solvers/MHD
mv epotFoam $WM_PROJECT_USER_DIR/applications/solvers/MHD
cd $WM_PROJECT_USER_DIR/applications/solvers/MHD/epotFoam
wmake
```

Download the case directory epotHunt.tar.gz and move it to the run directory

```
tar xzf epotHunt.tar.gz
rm epotHunt.tar.gz
mv epotHunt $WM_PROJECT_USER_DIR/run
cd $WM_PROJECT_USER_DIR/run/epotHunt
```
The huntFlow tutorial

Case parameters

- $M = 20$
- $Re = 2$
- $c_s = 0$
- $c_h = \infty$

Boundary layers scaling laws

\[
\delta_H = \frac{1}{M} \quad (9a)
\]
\[
\delta_S = \frac{1}{M^{1/2}} \quad (9b)
\]

Figure: Hunt problem
Running the case

To run the case...

blockMesh
epotFoam >& log &

...since it will take a while, have a look at the case files
A uniform 40x40x40 mesh is employed to simulate the half-duct over the centerline for $y > 0$

vertices
(0 0 -1) //0
(20 0 -1) //1
(20 1 -1) //2
(0 1 -1) //3
(0 0 1) //4
(20 0 1) //5
(20 1 1) //6
(0 1 1) //7
);
Boundary Conditions

Electrical potential boundary equations are specified in 0/PotE

Conductive wall \( \equiv \phi(\Gamma) = 0 \)

- type fixedValue;
  - value uniform 0;

Insulating wall \( \equiv \frac{\partial \phi}{\partial n} = 0 \)

- type zeroGradient;

For other patches the setting is zeroGradient, except for the symmetry plane. Velocity and pressure fields employ usual BCs (i.e. noSlip, inlet-velocity/outlet-pressure)
Magnetic induction intensity is defined in `electromagneticProperties`. All `transportProperties` are assumed to be constant and equal to 1.

Temporal discretization scheme: Crank-Nicholson (explicit, second order).
Spacial discretization scheme: Gauss liner (central differencing).
Residual tolerance: 1e-06 (p), 1e-05 (u, PotE).
To validate the results, the analytical velocity profiles available in hunt.dat would be employed. To extract the numerical data and compare it

postProcess -func sample
./plot.sh

The sampling data utility produces data for every written time step. The plot.sh is a script for the generation of the comparison picture employing gnuplot
Results

![Graph showing comparison between Side Profile, Hartmann Profile, and Hunt, 1965 results.](image-url)
Results

**Figure**: Velocity contour Hunt flow

**Figure**: Potential contour Hunt flow
Follow-up activities

Tutorials for elemental 2D MHD flows (Hunt and Shercliff cases) were developed for both mhdFoam and epotFoam.

- Extension of epotFoam capabilities to cases with wall of finite conductivity, i.e. coupling solid and fluid domain to calculate potential.

- Implementation of wall function treatment to ease computational time at high $M$ (Widlund, 2003).

- Code validation and development of tutorials for non-uniform magnetic field (Smolentsev, 2015).

- Q2D MHD turbulence modeling.
Thank you for your attention

Questions?