Implement a simple electromagnetic solver

Implement a simple electromagnetic solver

Contents

- You will implement a simple two-equation solver from scratch, and validate it with a test case.

Prerequisites

- You are familiar with the directory structure of OpenFOAM applications.
- You are familiar with user compilation procedures of applications.
- You are familiar with the fundamental high-level components of application codes, and how new classes can be introduced to an application.

Learning outcomes

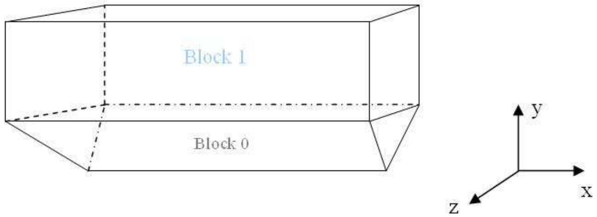
- You will practice high-level coding and modification of solvers.
- You will adapt case set-ups according to the new solver.
- You will improve your understanding of classes and object orientation, from a high-level perspective.

Note that you will be asked to pack up your final cleaned-up directories and submit them for assessment of completion.

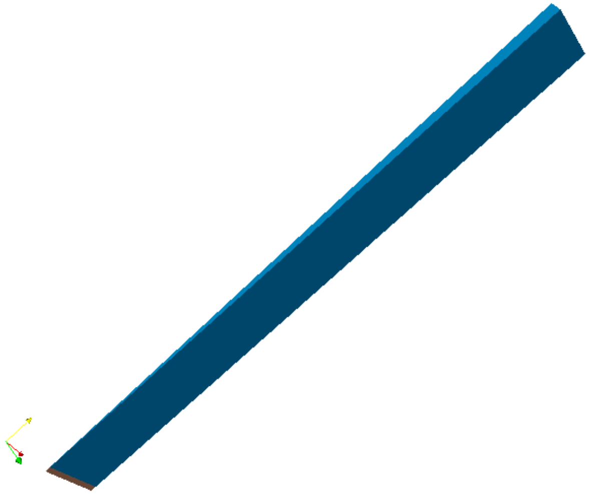
Problem: Electromagnetics of a rod surrounded by air

Geometry, computational domain, and rod/air regions.

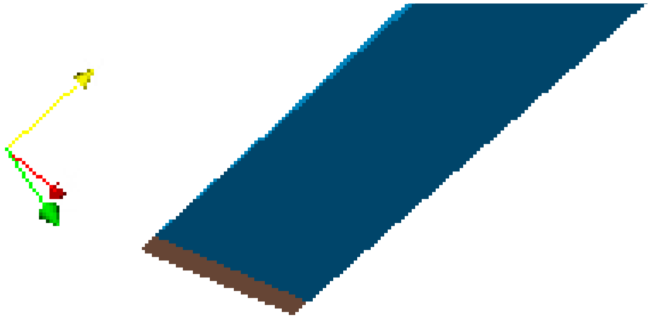
Electric rod.



Computational domain



In paraFoam
A 2D axi-symmetric case, with a wedge mesh



Zoom-up of rod.
For 2D wedge, the symmetry axis must be aligned with the x-axis, the wedge angle should be 5 degrees, with half on each side of the $z = 0$ plane.

Governing equations

Maxwell's equation:

$$\nabla \times E = 0 \tag{1}$$

where E is the electric field strength.

$$\nabla \cdot B = 0 \tag{2}$$

where B is the magnetic flux density.

$$\nabla \times H = J \tag{3}$$

where H is the magnetic field strength and J is current density.

Charge continuity:

$$\nabla \cdot J = 0 \tag{4}$$

Ohm's law:

$$J = \sigma E \tag{5}$$

where σ is the electric conductivity.
Constitutive law:

$$B = \mu_0 H \tag{6}$$

where μ_0 is the magnetic permeability of vacuum.

Combining Equations (1)-(6) and assuming Coulomb gauge condition ($\nabla \cdot A = 0$) leads to a Poisson equation for the magnetic potential and a Laplace equation for the electric potential...

Governing equations in OpenFoam

Magnetic potential:

$$\nabla^2 A = \mu_0 \sigma (\nabla \phi) \tag{7}$$

OpenFOAM representation:

```
solve
    (
    fvm::laplacian(A) ==
    sigma*muMag*(fvc::grad(ElPot))
    );
```

Electric potential:

$$\nabla \cdot [\sigma (\nabla \phi)] = 0 \tag{8}$$

OpenFOAM representation:

```
solve
    (
    fvm::laplacian(sigma,ElPot)
    );
```

We see that A depends on ϕ, but not vice-versa.

Implementing the rodFoam solver

Create the basic files in your user directory:

```
cd $WM_PROJECT_USER_DIR
mkdir -p applications/solvers/electromagnetics/rodFoam
cd applications/solvers/electromagnetics/rodFoam
foamNewSource App rodFoam
tree
```

We see:

```
.
|-- Make
|    |-- files
|    `-- options
`-- rodFoam.C
```

Make sure that the binary file ends up in your user directory:

```
sed -i s/FOAM_APPBIN/FOAM_USER_APPBIN/g Make/files
```

Try to compile. If it fails (for old versions), have a look at `Make/options` of e.g. `$FOAM_SOLVERS/basic/laplacianFoam/Make/options` to see that you should also add `meshTools`. This was a bug (or missing feature) in `foamNewSource`

Add a few lines to rodFoam.C

We need a mesh to discretize our equations on, and we need to initialize properties and fields.
After `#include "createTime.H"`, add:

```
#include "createMesh.H"    //In the OpenFOAM installation
#include "createFields.H"  //Must be implemented - see next slides
```

Continue adding (after the above), our equations:

```
solve ( fvm::laplacian(sigma, ElPot) );
solve ( fvm::laplacian(A)==sigma*muMag*(fvc::grad(ElPot)) );
```

Add some additional things that can be computed when we know `A` and `ElPot`:

```
B = fvc::curl(A);
Je = -sigma*(fvc::grad(ElPot));
```

We also want to write out the results to a new time directory.
Continue adding:

```
runTime++;
sigma.write();
ElPot.write();
A.write();
B.write();
Je.write();
```

The createFields.H file (1/6)

We need to construct and initialize `muMag`, `sigma`, `Elpot`, `A`, `B`, and `Je`.
Edit the `createFields.H` file.

Read `muMag` from a dictionary:

```
Info<< "Reading physicalProperties\n" << endl;
IOdictionary physicalProperties
(
    IOobject
    (
        "physicalProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ,
        IOobject::NO_WRITE
    )
);
dimensionedScalar muMag
(
    "muMag",
    dimensionSet(1, 1, -2, 0, -2, 0, 0),
    physicalProperties
);
```

The createFields.H file (2/6)

Construct `volScalarField sigma`:

```
Info<< "Reading field sigma\n" << endl;
volScalarField sigma
(
    IOobject
    (
        "sigma",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

The createFields.H file (3/6)

Construct `volScalarField Elpot`:

```
volScalarField ElPot
(
    IOobject
    (
        "ElPot",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

The createFields.H file (4/6)

Construct `volVectorField A`:

```
Info<< "Reading field A\n" << endl;
volVectorField A
(
    IOobject
    (
        "A",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

The createFields.H file (5/6)

Construct and initialize `volVectorField B`:

```
Info << "Calculating magnetic field B \n" << endl;
volVectorField B
(
    IOobject
    (
        "B",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    fvc::curl(A)
);
```

The createFields.H file (6/6)

Construct and initialize `volVectorField Je`:

```
volVectorField  Je
(
    IOobject
    (
        "Je",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    -sigma*(fvc::grad(ElPot))
);
```

Compile the solver

We have implemented a solver, which is compiled by:

```
wmake
```

If successful, the output should end something like:

```
-o /chalmers/users/hani/OpenFOAM/oscfd-plus/platforms/linux64GccDPInt32Opt/bin/rodFoam
```

We now need a case to use the solver on. It is provided to you (`rodFoamCase.tgz`), since it is too much to describe in slides. Unpack and run using:
(NOTE, you may have to do `sudo apt install gv` first, for showing the plots at the end)

```
tar xzf rodFoamCase.tgz; cd rodFoamCase; ./Allrun 2>&1 | tee log_Allrun
```

Boundary and initial conditions

- We solve for the magnetic potential `A` (A) and the electric potential `ElPot` (ϕ), so we need boundary conditions:

	block 0, sides	block 1, sides	block1, top
A	$\nabla A = 0$	$\nabla A = 0$	$A = 0$
ϕ	$\phi_{left} = 707, \phi_{right} = 0$	$\nabla \phi = 0$	$\nabla \phi = 0$

  and we initialize the fields to zero.

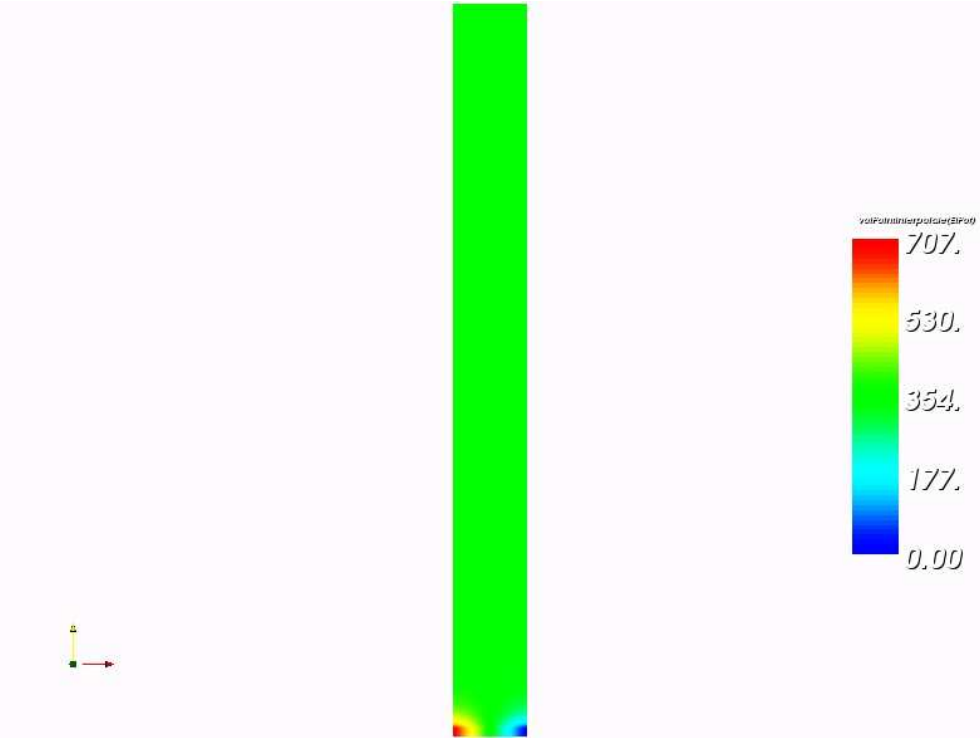
- The internal field of the electric conductivity sigma (σ) is nonuniform:

$$\sigma = \begin{cases} 2700 & \text{if } x < R \text{ where R -radius of the block 1} \\ 1e-5 & \text{otherwise} \end{cases}$$

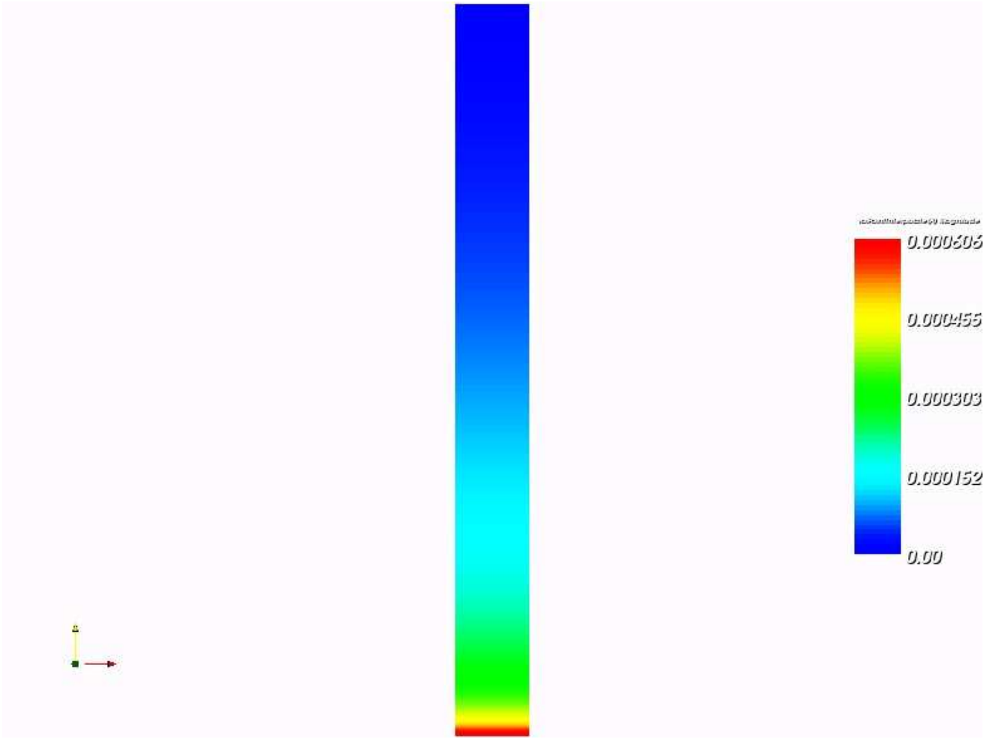
  so we use a `volScalarField` and `setFields` to set the internal field.

- The magnetic permeability of vacuum (μ_0) is read from the `constant/physicalProperties` dictionary.

View the results in paraFoam



Electric potential (ϕ)



Magnitude of magnetic potential vector (A)

Validation of components of A and B using Gnuplot

- Our numerical results should be validated with analytical results
- For this we need to extract the components and extract the values along a line:

```
postProcess -func 'components(A)' -time 1
postProcess -func 'components(B)' -time 1
postProcess -func singleGraph -time 1
```

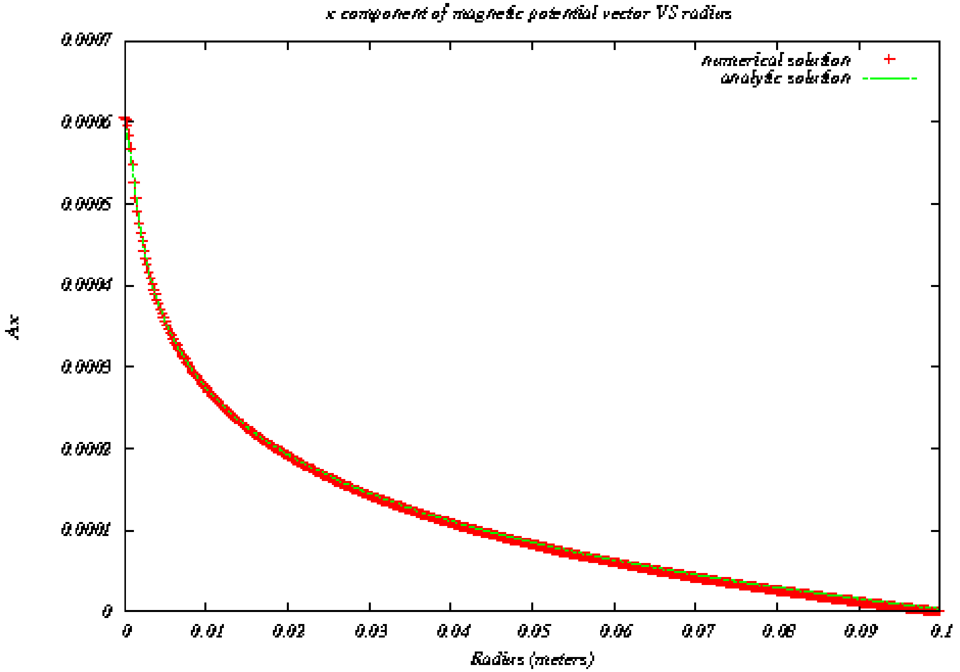
- The results are validated with the analytical solution using Gnuplot:

```
gnuplot rodComparisonAxBz.plt
```

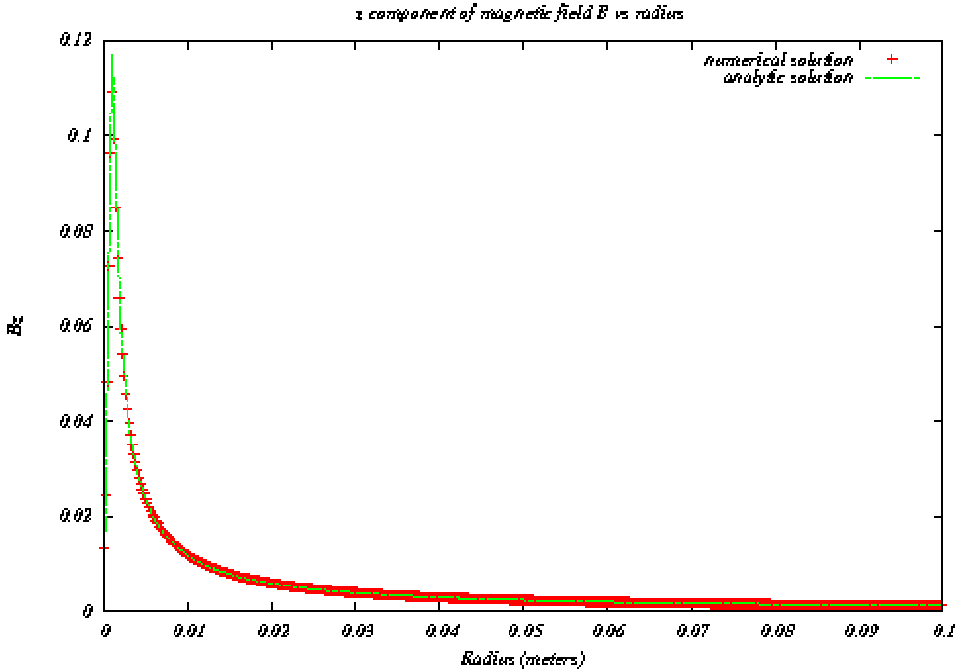
- Visualize using:

```
gv rodAxVSy.ps
gv rodBzVSy.ps
```

Validation of components of A and B using Gnuplot



x-component of magnetic potential vector A vs radius of the domain.



z-component of the magnetic field B vs radius of the domain

Analytic solution

- Analytic solution for x component of magnetic potential vector A

$$A_x = \begin{cases} A_x(0) - \frac{\mu_0 J x^2}{4} & \text{if } r < R, \\ A_x(0) - \frac{\mu_0 J R^2}{2}[0.5 + ln(r/R)] & \text{otherwise} \end{cases}$$

  where $A_x(0) = 0.000606129$, $J = 19.086e + 7$ is the current density and R is the radius of the electric rod.

- Analytic solution for z component of magnetic field B

$$B_z = \begin{cases} \frac{\mu_0 J x}{2} & \text{if } r < R, \\ \frac{\mu_0 J R^2}{2r} & \text{otherwise} \end{cases}$$

  where $J = 19.086e + 7$ is the current density and R is the radius of the electric rod.

- Have a look in `rodComparisonAxBz.plt` to see how to plot a function in Gnuplot.