Outline

- Introduction to problem
- compressibleInterFoam solver
- Equations for electric field
- Modify solver to include E-equation
  - Exercise
- Create and implement new BC
- Test case
  - Results
- Future work

Learning outcomes

With this tutorial the reader should be able to modify an existing solver and create/implement a new boundary condition.

- Use compressibleInterFoam solver
- Modify a solver for specific need
- Implement a new BC

Introduction

Electric field representing laser beam

Two different ways of representing the energy deposition of a laser:

- Ray Tracing
  - Beam is discretized into a finite number of rays
  - Each ray carries energy, which is absorbed or reflected upon irradiation
  - Energy absorption is dependent on wave length
- Electromagnetic wave equations (Maxwell's equations)
  - Electromagnetic theory of optics
  - Represent beam with electric field

Introduction

Features of solver

In order to model a laser beam heat source as an electric field a model for heat transfer, fluid flow and electric field is needed. The electric field has to be coupled to the energy equation to provide input from heat source. The solver should also include interface capturing to distinguish between liquid/solid and gas. `compressibleInterFoam` solver should serve as a good starting point when developing this model.

OpenFOAM structure

All sovlers are located under `applicaitons/solvers` directory. Using the command `tree -d -L 1 $WM_PROJECT_DIR` can be used to visualize the structure:

```
$WM_PROJECT_DIR
|– applications
|– bin
|– doc
|– etc
|– platforms
|– src
|– tutorials
|– wmake
```

```
|-- applications
     |-- solvers
     |-- test
     |-- utilities
```

The compressibleInterFoam solver is located in: OpenFOAM/OpenFOAM-2.4.x/applications/solvers/multiphase/compressibleInterFoam

Solver for electric field

compressibleInterFoam

The compressibleInterFoam solver is a solver for two compressible non-isothermal fluids using VOF method for interface capturing. Momentum and fluid properties as density and velocity are of mixture type.

- non-isothermal
- 2 fluids
- compressible
- mixture model
- VOF

Lets have a look at the `compressibleInterFoam` solver.

- Start by initializing the OpenFOAM-2.4.x environment:
  `OF24x`
- The solver can be reached using the command:
  `cd $FOAM_APP/applications/solvers/multiphase/compressibleInterFoam/`

The compressibleInterFoam directory consists of following files, type ls

```
compressibleInterFoam.C createFields.H
TEqn.H    pEqn.H    UEqn.H
alphaEqns.H alphaEqnsSubCycle.H
```

And a Make-directory which contains instructions for compilation

```
files    options
```

And a directory for the mixture model:

twoPhaseMixtureThermo

compressibleInterFoam 3/3

- Variables used in `compressibleInterFoam.C` are constructed and initialized in `createFields.H` Open `createFields.H` and have a look.
- Tutorials or test cases are provided in OpenFOAM. Use environment variable `tut` and look for a tutorial for compressibleInterFoam.
  - There are two cases: `depthCharge2D` and `depthCharge3D`
  - Copy `depthCharge2D` to User-directory and run it.
  - View the result using paraFoam

Time to implement a new PDE

The purpose with a solver for electric field is to model laser beam welding with a heat source represented by an electric field.

Now it is time to understand what equations to add and how to do so in order to calculate an electric field E and appropriate energy input.

Next couple of slides is a detailed description of how to modify `compressibleInterFoam` solver.

Gaussian beam 1/2

Laser beam propagation can be approxiamted by an ideal Gaussian beam described by TEM_{00}mode.

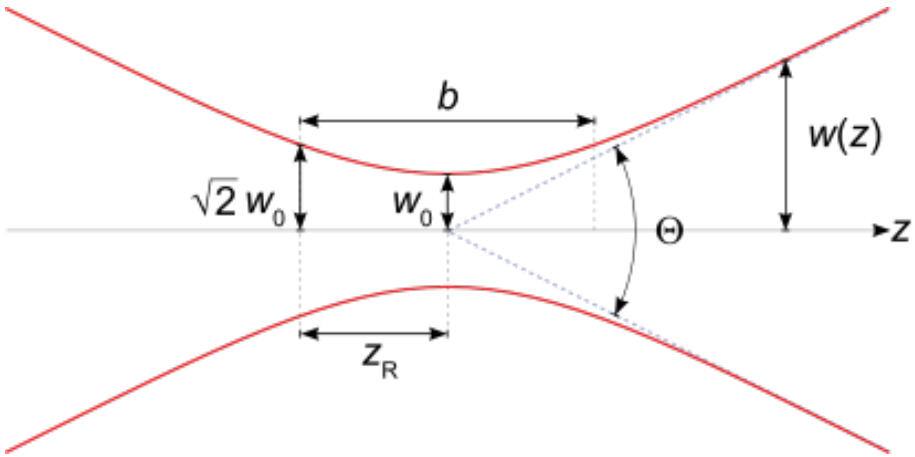

Figure: Schematic image of Gaussian beam with propagation along z-axis, minimum waist, w_0, Rayleigh range, z_R, and width, w(z)

Gaussian beam 2/2

For a given wavelength the shape of the Gaussian beam is determined from w_0.

To use a Gaussian beam model $w_0 < 2\lambda$ must be fulfilled.

General wave equation

The general wave equation is a 2nd order partial differential equation used to describe different type of waves, for example light. For electromagnetic waves it is written:

$$\frac{\delta^2 E}{\delta t^2} = c^2 \nabla^2 E \tag{1}$$

It describes propagation of electromagnetic waves through a medium or in vacuum.

It derives from Maxwell's equations.

Background electromagnetic wave equation

Maxwell's equations

Maxwell's equations is a basis for electromagnetic theory of optics and is a set of PDE's.

$$\nabla \cdot \bar{D} = \rho \qquad (2)$$

$$\nabla \cdot \bar{B} = 0 \qquad (3)$$

$$\nabla \times \bar{E} = -\frac{\delta \bar{B}}{\delta t} \qquad (4)$$

$$\nabla \times \bar{H} = \bar{J} + \frac{\delta \bar{D}}{\delta t} \qquad (5)$$

$$\bar{B} = \mu \bar{H} \qquad (6)$$

$$\bar{D} = \epsilon \bar{E} \qquad (7)$$

D - electric displacement, ρ - volume charge density, B - magnetic field, E - electric field, H - magnetic field intensity and J - current density.

continue Maxwell's equations

Equation for electric field given by combining equation (4-6) with harmonic wave equation:

$$\nabla^2 \bar{E}(\bar{r}) + w^2 \mu(\epsilon - \frac{i\sigma}{w} \bar{E}(\bar{r}) = 0 \tag{8}$$

The equation is further simplified by assuming a non-conductive media so that σ -term disappears, giving:

$$\nabla^2 \bar{E} + k_0^2 \cdot \epsilon_r \cdot \mu_r \cdot \bar{E} = 0 \tag{9}$$

with k_0 wave number of free space, ϵ_r is relative permittivity and μ_r relative permeability.

Energy deposition source term

From Maxwells equations –> energy deposition generates a source term which is explicitly coupled with the energy equation.
Energy source term to be implemented is written like:

$W = 2 \cdot \bar{E}^2 \cdot \epsilon$

And the total energy input is the sum of the source term over all cells.

Equations to be implemented

To summarize the equations to be implemented in order to calculate the electric field E are:

$$\nabla^2 \bar{E} + k_0^2 \cdot \epsilon_r \cdot \mu_r \cdot \bar{E} = 0 \tag{10}$$

$$W = 2 \cdot \bar{E}^2 \cdot \epsilon \tag{11}$$

And for initial electric field representing incoming beam:

$$E_\tau = \tau * E_0 e^{-(\frac{r_b}{w(z)})^2} \times cos(kw * z_f + kw * (\frac{r_b}{2R_{zf}}) - \psi) \tag{12}$$

Copy and modify existing solver

How to implement a new solver 1/8

In order to add a new equation to an existing solver start by :

- Copy an existing solver to user directory:

```
cd $WM_PROJECT_USER_DIR
cp -r $FOAM_SOLVERS/multiphase/compressibleInterFoam/ .
```

- Set the name of the source file to the name of the new solver:

```
mv compressibleInterFoam.C compressibleInterFoamEikonal.C
```

- Change `compressibleInterFoam` to `compressibleInterFoamEikonal` everywhere in the .C file:

```
sed -i
s/"compressibleInterFoam"/"compressibleInterFoamEikonal"\
/g compressibleInterFoamEikonal.C
```

Clean and compile

How to implement a new solver 2/8

- In Make/files- file change the path for the executable:

  `compressibleInterFoamEikonal.C`

  `EXE = ($FOAM_USER_APPBIN)/compressibleInterFoamEikonal`

- In Make/options-file make sure all necessary libraries are included
- Clean to remove dependency lists and compile the solver:

  `wclean`

  `wmake`

Construct and declare new fields and variables

How to implement a new solver 3/8

To make the code easier to read and interpret the new fields are declared and constructed in a seperate file: `createEmgFields.H`

- Construct new field for E:

```
volVectorField E
(
    IOobject
   (
    "E",
    runTime.constant(),
    mesh,
    IOobject::MUST_READ,
    IOobject::AUTO_WRITE,
   ),
   mesh
);
```

Construct and declare new fields and variables

- Construct and declare new fields for heat source and source term:

```
volScalarField Slaser
(
    IOobject
    (
     "Slaser",
     runTime.constant(),
     mesh,
     IOobject::NO_READ,
     IOobject::AUTO_WRITE,
    ),
    mesh,
    dimensionSet(1, -1, -2, 0, 0, 0,
0)
);
```

```
volScalarField QSlaser
(
    IOobject
    (
     "QSlaser",
     runTime.constant(),
     mesh,
     IOobject::NO_READ,
     IOobject::AUTO_WRITE,
    ),
    mesh,
    dimensionSet (1 -1 -3 0 0 0 0)
);
```

Construct and declare new fields and variables

How to implement a new solver 5/9

```
dimensionedScalar sumSlaserVol
(
    sumSlaserVol
    (
     "sumSlaserVol",
     dimensionSet(1, 2, -2, 0, 0, 0, 0),
     0
);
```

Create EEqn.H

When all fields and scalars are constructed it is time to add the new equation to be solved.

- Create a file, EEqn.H for E-equation

  ```
  vi EEqn.H
  ```

- In the new file write the equation :

```
(
    solve
    (
     fvm::laplacian(E)+sqr(k)*epsR*muR*E(1.0-alpha1)
);
```

Exercise: create Slaser in emgSourceTerm.H

A source term, Slaser, is calculated from the electric energy density. A file `emgSourceTerm.H` has to be created and a loop to calculate total energy input.

- Open `emgSourceTerm.H` :
  `vi emgSourceTerm.H`
- In `emgSourceTerm.H` add:

```
dimensionedScalar sumSlaser=0.0;

Slaser=0.5*(E&E)*eps.value()*Foam::(neg-alpha1);
```

- `Now calculate sumSlaserVol over all cells (a product of Slaser(celli)*Volume(celli).`
- `Add Qlaser:`
  `{Qlaser = (Slaser/sumSlaserVol)*EffLaserPower);`

Exercise: compile emgSourceTerm.H

Now compile the solver using wmake

Problem 1

For some reason the dimension of Slaser is "lost" when calculating the sum in the following loop:

```
scalar adimSumSlaser = 0.0;
forAll(mesh.C(), celli)
(
adimSumSlaser+=Slaser[celli];
)
```

Problem 1 - Solution

The solution is to write

```
sumSlaser.value() = adimSumSlaser;
```

And then calculate the total energy according to eq.(20) by Courtois et al.(2)

```
Qlaser = (Slaser/sumSlaserVol)*EffLaserPower;
```

Couple electric field and energy equation

How to implement a new solver 7/8

$\mathrm{TEqn} = \mathrm{S}_{laser} \cdot (\alpha_1 / C_{p1} + \alpha_2 / C_{p2})$

In OpenFOAM language:

- In `TEqn.H` after `TEqn.solve()` add the following lines:

```
(
    solve
    (
    TEqn == QSlaser
     *(
     alpha1/mixture.thermo1().Cv()
    +alpha2/mixture.thermo2().Cv()
     )
);
mixture.correct();
```

How to implement a new solver 8/8

include files in .C file

The final step is to include all new files and equations in the source code. Open `compressibleInterFoamEikonal.C` and add:

- just under `#include createFields.H` add:

```
#include createEmgFields.H
```

- in the loop just before `#include TEqn.H` add:

```
#include EEqn.H
#include emgSourceTerm.H
```

Equations of new BC

An incoming electric field with Gaussian distribution, representing the light beam, should be applied at top patch, [2].

$$E_\tau = \tau * E_0 e^{-(\frac{r_b}{w(z)})^2} \times cos(kw * z_f + kw * (\frac{r_b}{2R_{zf}}) - \psi) \quad (13)$$

With w(z), ψ and R_{zf}:

$$w(z) = w_0 \sqrt{1 + (\frac{z_R}{z_{Ra}})^2} \quad (14)$$

$$\psi = arctan(\frac{z_R}{z_{Ra}}) \quad (15)$$

$$R_{zf} = z_f * (1 + \frac{{z_{Ra}}^2}{z_f}) \quad (16)$$

w(x): width of beam,
ψ: Gouy phase,
R_{zf}: spherical front radius.
E_0: amplitude of electric field,
w_0: minimum waist of beam,
z: position along propagation axis,
z_f: distance from ebam center to focal point and kw: wave number.

Start by copying an existing BC, in this case `parabolicVelocity` which is from the OpenFOAM-extend project.
`parabolicVelocity` will be compiled and used as a dynamic library.

```
mkdir -p
$WM_PROJECT_USER_DIR/src/finiteVolume/fields/fvPatchFields/derived

cd $WM_PROJECT_USER_DIR/src/finiteVolume/fields/fvPatchFields/derived

svn checkout svn://svn.code.sf.net/p/openfoam-extend/svn/trunk/\
Core/OpenFOAM-1.5-dev/src/finiteVolume/fields/fvPatchFields/\
derived/parabolicVelocity

cd $WM_PROJECT_USER_DIR/src/finiteVolume/fields/\
fvPatchFields/derived/parabolicVelocity
```

How to implement new BC 3/5

Change the name of the .C-file and .H-file and create Make/files

- Change name

  ```
  mv parabolicVelocityFvPatchField.C gaussianElectricFvPatchField.C
  ```

- change the name parabolicVelocityFvPatchField to gaussianElectricFvPatchField everywhere in the code.
- create Make/files file by copying from(`$FOAM_SRC/finiteVolume/Make`) :

  ```
  fvPatchFields = fields/fvPatchFields
  derivedFvPatchFields = $(fvPatchFields)/derived
  $(derivedFvPatchFields)/gaussianElectric/gaussianElectricFvPatchvectorField.C

  LIB = $(FOAM_USER_LIBBIN)/libmyFiniteVolume
  ```

- In options-file change:

  ```
  EXE_INC = \
  -I$(LIB_SRC)/finiteVolume/lnInclude
  EXE_LIBS =
  ```

- Compile to see that it works

  ```
  wmake libso
  ```

How to implement new BC 4/5

initiate parameters and variables

- In .H-file new parameters are declared (for example):

```
//unit vector normal to patch
vector n_;

//unit vector along 1D patch
vector tau_;
```

- In .C-file initiate the parameters needed
  kw, z_{Ra}, r_b, z_f, Etau, R_{zf}, psi, omega0
- add new member functions to calculate new variables

```
//distance from beam centre to cell centre
scalarField rb = mag((Ccf-focalP_) & tau_);

//Guoy phase equation
scalar psi = Foam::atan(zf_zRa);
```

Have a look in `gaussianElectricFvPatchVectorField.C` for all new member functions

The equation to be solved, eq (7), is written in OpenFOAM language:

```
Etau = tau_E0_*Foam::exp(-(Foam::sqr(rb/omegaZ)))
Foam::cos(kw*sz+kw*Foam::sqr(rb/2.0*Rzf))-psi);

vectorField::operator=(Etau);
```

compile the new library

```
wmake libso
```

Input parameters for new BC

The implemented boundary condition for an initial Gaussian distribution of the electric field requires some input parameters:

- **n** - direction of field E on boundary (0 1 0)
- **tau** - direction along 1D boundary line (1 0 0)
- **lambda** - wave length
- **omega0** - minimum waist of beam
- **E0** - amplitude of electric field
- **focalP** - coordinates for location of focal point
- **value** - this is just a uniform value (0.0 0.0 0.0)

Test case general

General steps of how to set up a new test case

- Copy an existing case for the compressibleInterFoam solver and modify, or set up your own test case to verify the new solver.
- Change in blockMeshDict in constant/polymesh according to wanted geometry
- Update setFieldsDict in system/ -directory
- Create inital conditions for new variables in 0/-directory
- Update controlDict in system/-directory to new solver name

Test case eikonal2D

Specific test case Eikonal solver

A new test case is set up for validation of the new solver for electric field.
The test case is called `eikonal2D` and is a domain consisting of metal and gas (air). It is built from 6 blocks and with the dimensions seen in the image.

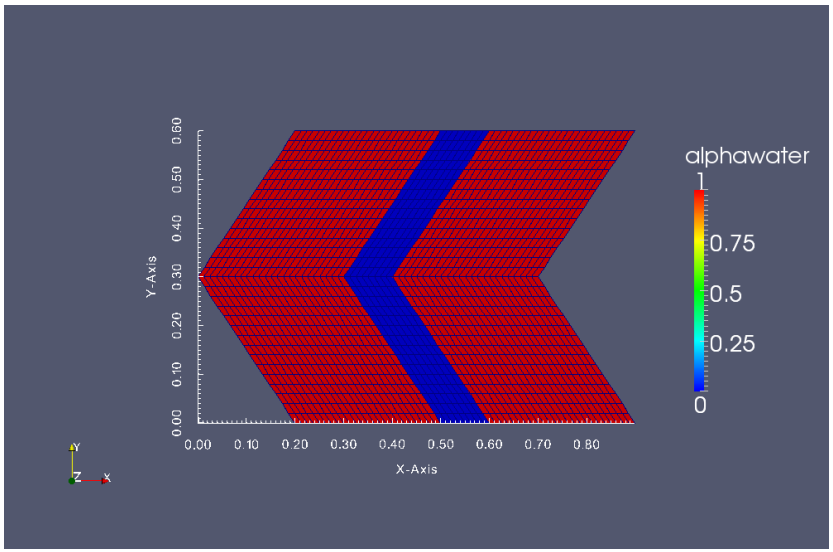

Figure: Geometry of test case

New initial- and boundary conditions has to be assigned.
In 0-directory set boundary conditions according to table:

Patch	**E**	**T**	**U**	**p**
top	gaussianElectric	zeroGradient	inletOutlet	zeroGradient
bottom	zeroGradient	zeroGradient	zeroGradient	zeroGradient
left	fixedValue	fixedValue	fixedValue	zeroGradient
right	fixedValue	fixedValue	fixedValue	zeroGradient
frontAndBack	empty	empty	empty	empty

Table: Boundary conditions

Initial field for gas/liquid is set using setFields option rotatedBoxToCell in system/setFieldsDict

With the following inputs the blocks are rotated 30 degrees around z-axis.

```
rotatedBoxToCell
origin (0.5e-3 0 0);
i (0.1e-3 0 0);
j (-0.2e-3 0.3e-3 0);
k (0 0 0.01e-3);
```

```
rotatedBoxToCell
origin (0.3e-3 0.3e-3 0);
i (0.1e-3 0 0);
j (0.2e-3 0.3e-3 0);
k (0 0 0.01e-3);
```

Results

When simulation finished successfully result data can be analyzed using ParaView application.

In case directory write `paraFoam` to initialize the tool.

Results Test case eikonal2D

When input parameters need to be optimized...

A first result is shown in image:

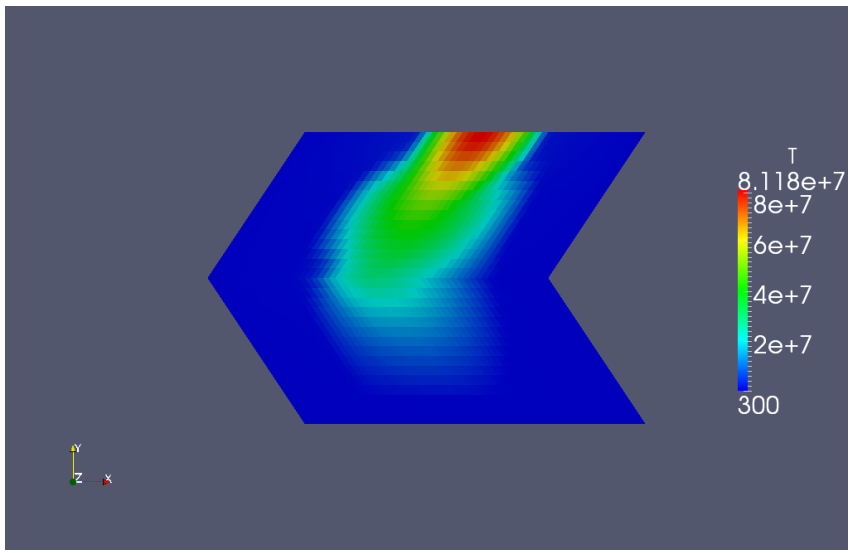

Figure: Temperture distribution

Future work

Optimize input parameters to get an accurate solution that converges.

Since energy conservation equation and electromagnetic field equations have different requirements for grid size a multi-region method could reduce computational cost.

- multi-region approach
  - chtMultiRegionSolver
- couple energy equation and electromagnetic field solved on different grid size
- iteratively updated

Questions?

Thank you for your attention!

Do you have any questions?

References

1. T. Maric, J. Hpken, K. Mooney (2014) The OpenFOAM Technology Primer
   Publisher: sourceflux UG
2. M. Courtois, M. Carin, P. Le Masson, S. Gaied, M. Balabane (2013) A new approach to compute multi-reflections of laser beam in a keyhole for heat transfer and fluid flow modelling in laser welding *Journal of Physics D* **46**, pp.505305-505319
3. M. Courtois, M. Carin, P. Le Masson, S. Gaied, M. Balabane (2014) A complete model of keyhole and melt pool dynamics to analyze instabilities and collapse during laser welding *Journal of Laser Applications* **26**, pp.042001
4. A. Satya Narayanan, S. K. Saha (2015) Waves and oscillations in nature - An Introduction
   CRC Press, Taylor and Francis Group
5. O. Svelto (2010) Principles of laser
   Springer, 5th edition
6. *www.openfoam.org*
7. *https://en.wikipedia.org/wiki/Gaussian_beam*