Modeling free surface thermal flow with relative motion of heat souce and drop injector with respect to a liquid pool

Pradip Aryal

Department of Welding Technology,
University West,
Trollhattan, Sweden

November 28, 2019

Contents

Introduction

Various applications of droplet impact

Engineering applications of droplet impact on a solid surface or a liquid pool

- Inkjet printing
- Spray cooling
- Anti- icing
- Liquid atmoization
- Welding

Introduction

Relative motion of heat source/droplet and a base metal

In particular case of welding for instance, falling droplets travels with the heat source along the base metal. This results in a relative motion between heat source/droplets and a base metal.

Two different ways of modeling motion of heat source

- Moving heat source in a fixed coordinate system [1]

- Fixed heat source in a moving coordinate system [2]

Introduction

Solver for modeling free surface thermal flow

Solver prerequisite

- Two-phase fluid flow
- Interface capturing to distinguish between liquid and gas phase.
- Energy conservation equation formulated with temperature for heat transfer
- Heat source term coupled with energy equation

interFoam solver can be used as a starting solver

Introduction

OpenFoam structure

Initialize OpenFOAM in the terminal window,

Use the command `tree -d -L 1 $WM_PROJECT_DIR` to visualize all the OpenFOAM structure.

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

All OpenFOAM solvers are located in applications/solvers

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

The interFoam solver is located in: OpenFOAM/OpenFOAM-3.0.1/applications/solvers/multiphase/interFoam

The interFoam solver is for two incompressible, isothermal immiscible fluids using VOF method for interface capturing.

Introduction

interFoam

Acess interFoam solver by typing:

```
cd $WM_PROJECT_DIR/applications/solvers/multiphase/interFoam
```

Type `ls` to see all the files in the folder.

Open the main solver file `interFoam.C`

- In the beginning, it contains several headers necessary to initialize diffirent fields and to obtain libraries needed for finite volume solution.
- It is followed by the main function. It contains several classes for time and mesh control.
- Then runTime is initiated using while loop.
- Inside the runTime loop, pimple loop is initiated for pressure-velocity corrector.
- When looping is finished, it writes out the data file at every write interval

Solver for free surface thermal flow

Creating a new solver: interThermalFoam

Creating solver I: interThermalFoam

- Copy interFoam solver to user directory

```
cd $WM_PROJECT_USER_DIR
cp -r $WM_PROJECT_DIR/applications/solvers/multiphase/interFoam $WM_PROJECT_USER_DIR/applications/solvers/multiphase/interFoam
```

- Rename the copied solver directory using command

```
cd $WM_PROJECT_USER_DIR/applications/solvers/multiphase
mv interFoam interThermalFoam
```

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

```
cd interThermalFoam
mv interFoam.C interThermalFoam.C
```

- Change interFoam to interThermalFoam everywhere in the .C file with:

```
sed -i s/"interFoam"/"interThermalFoam"/g interThermalFoam.C
```

- Change the path for executable in Make/file:

```
interThermalFoam.C
EXE = ($FOAM_USER_APPBIN)/interThermalFoam
```

- Clean to remove the dependency list and compile the solver.

```
wclean
wmake
```

Solver for free surface thermal flow

Including energy conservation equation in interThermalFoam

The enery conservation equation for single fluid model formulated with temperature is given as

$$\frac{\partial(\rho C_p T)}{\partial t} + \nabla.\left(\rho U C_p T\right) = \nabla.(k\nabla T) + S_e$$

Here Se, is the energy source term. The incoming energy source term is given in terms of surface heat flux Qhs and assumed to have gaussian distribution [3] on the top metal surface given by

$$Qhs = \frac{\eta \dot{Q}}{\pi {r_L}^2} exp\left(-\frac{\eta r^2}{{r_L}^2}\right)$$

η = Efficiency of heat source

$\dot{Q}$ = Power of heat source

r = distance between a point on the radial coordinate and the centre of heat source in x and y coordinates.

r_L = effective radius of heat source beam

Solver for free surface thermal flow

Implementing TEqn in the solver

- Go to the new solver interThermalFoam

```
cd $WM_PROJECT_USER_DIR
cd applications/solvers/multiphase/interThermalFoam
```

- Create a file named Teqn.H

```
vi TEqn.H
```

- And write the following lines of code to implement PDE of energy equation in OF language

```
surfaceScalarField alpha1f = min(max(fvc::interpolate(alpha1), scalar(0)), scalar(1));
kappaf = alpha1f*kappa1 + (1.0 - alpha1f)*kappa2;

fvScalarMatrix TEqn
    (
       fvm::ddt(rhoCp, T)
      +fvm::div(rhoPhiCpf, T)
      -fvm::laplacian(kappaf, T)
    );
TEqn.relax();
   solve ( TEqn == Qhs*mag(fvc::grad(alpha1))*factor);
```

- Save and close the file.

Solver for free surface thermal flow

Implementing TEqn in the solver

```
surfaceScalarField alpha1f = min(max(fvc::interpolate(alpha1), scalar(0)), scalar(1));
kappaf = alpha1f*kappa1 + (1.0 - alpha1f)*kappa2;           //Thermal conductivity of mixture

fvScalarMatrix Teqn
    (
        fvm::ddt(rhoCp, T)

      +fvm::div(rhoPhiCpf, T)

      -fvm::laplacian(kappaf, T)
    );
TEqn.relax();
   solve ( TEqn == Qhs*mag(fvc::grad(alpha1))*factor);
```

rhoCp depends on alpha. Open alphaEqnSubCycle.H and write the following line of code at the end.

```
rhoCp == alpha1*rho1*cp1 + alpha2*rho2*cp2;
```

`rhoPhiCpf` is updated with alpha equation. Open alphaEqn.H and write the following line of code below the equation of rhoPhi at two instances.

```
rhoPhiCpf = alphaPhi*(rho1*cp1 - rho2*cp2) + phiCN*rho2*cp2;
```

- Save and close the file.

$$\frac{2\rho}{\rho 1 + \rho 2}$$

`factor` is updated with the update of `rho1 and rho2 in every subcycle of` alpha equation. Open alphaEqnSubCycle.H and write the code between rho and rhoCp.

```
factor = scalar(2)*rho/(rho1+rho2);
```

Solver for free surface thermal flow

Construct and declare new fields and variables

- Inclusion of energy equation require to construct and initialize additional fields in createFields.H
- They are:

Three `VolScalarField (i.e. T, rhoCp, and factor)`
Three `surfaceScalarField (i.e. alpha1f, kappaf, rhoPhiCpf)`
Four `dimensionedScalar(i.e. cp1, cp2 , kappa1 and kappa2)`

Open createFields.H and add the following line of code code after the end of `volScalarField rho (i.e. rho.oldTime();)` and before `surfaceScalarField rhoPhi`

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

Solver for free surface thermal flow

Construct and declare new fields and variables

```
Info<< "Reading thermophysicalProperties of phase 1\n" << endl;
IOdictionary thermophysicalPropertiesPhase1
(
  Ioobject
  (
    "thermophysicalPropertiesPhase1",
    runTime.constant(),
    mesh,
    IOobject::MUST_READ,
    IOobject::NO_WRITE
  ),
);

// specific heat capacity of phase 1
const dimensionedScalar& cp1 =
thermophysicalPropertiesPhase1.lookup("cp1");



// thermal conductivity of phase 1
const dimensionedScalar& kappa1 =
thermophysicalPropertiesPhase1.lookup("kappa1");
```

Solver for free surface thermal flow

Construct and declare new fields and variables

```
Info<< "Reading thermophysicalProperties of phase 2\n" << endl;
IOdictionary thermophysicalPropertiesPhase2
(
  Ioobject
  (
    "thermophysicalPropertiesPhase2",
    runTime.constant(),
    mesh,
    IOobject::MUST_READ,
    IOobject::NO_WRITE
  ),
);


// specific heat capacity of phase 2
const dimensionedScalar& cp2 =
thermophysicalPropertiesPhase2.lookup("cp2");



// thermal conductivity of phase 2
const dimensionedScalar& kappa2 =
thermophysicalPropertiesPhase2.lookup("kappa2");
```

Solver for free surface thermal flow

Construct and declare new fields and variables

```
Info<< "Reading / initializing kappaf\n" <<endl;
surfaceScalarField kappaf
(
    IOobject
  (
      "kappaf",
       runTime.timeName(),
       mesh,
      IOobject::NO_READ,
      IOobject::NO_WRITE
),
  mesh,
  dimensionedScalar("kappaf",dimensionSet(1,1,-3,-1,0,0,0),
0.0)
);
```

Solver for free surface thermal flow

Construct and declare new fields and variables

```
Info<< "Reading / calculating rho*cp\n" <<endl;
volScalarField rhoCp
(
  Ioobject
  (
    "rho*cp",
    runTime.timeName(),
    mesh,
    IOobject::NO_READ,
    IOobject::NO_WRITE
  ),
  alpha1*rho1*cp1+alpha2*rho2*cp2,
  alpha1.boundaryField().types()
);
rhoCp.oldTime();
```

Solver for free surface thermal flow

Construct and declare new fields and variables

```
Info<< "Reading / calculating rhoPhi*cp\n" <<endl;
surfaceScalarField rhoPhiCpf
(
  Ioobject
  (
    "rhoPhicpf",
    runTime.timeName(),
    mesh,
    IOobject::NO_READ,
    IOobject::NO_WRITE
  ),
  fvc::interpolate(rhoCp)*phi
);
```

Solver for free surface thermal flow

Construct and declare new fields and variables

```
Info<< "Reading / calculating factor\n" <<endl;
volScalarField factor
(
  Ioobject
  (
    ”factor",
    runTime.timeName(),
    mesh,
    IOobject::NO_READ,
    IOobject::NO_WRITE
  ),
  scalar(2)*rho/(rho1+rho2)
);
```

Solver for free surface thermal flow

Construct and declare new fields and variables

In order to make code easier to implement and interpret, the fields and properties for the heat source term are declared and constructed in a separate file: createLaserFields.H

```
Info<< "Reading the laser beam heat source\n" <<endl;
 IOdictionary beamProperties
(
  Ioobject
  (
    ”beamProperties",
    runTime.timeName(),
    mesh,
    IOobject::MUST_READ,
    IOobject::NO_WRITE
  ),
);
```

Solver for free surface thermal flow

Construct and declare new fields and variables

```
//laser speed
dimensionedVector Ulaser
(
    "Ulaser",
    dimensionSet(0, 1, -1, 0, 0, 0, 0),
    beamProperties.lookup("Ulaser")
);
```

```
//Initial position of laser beam
dimensionedVector Xbeam0
(
    ”Xbeam0",
    dimensionSet(0, 1, 0, 0, 0, 0, 0),
    beamProperties.lookup(”Xbeam0")
);
```

```
//laser beam starting time
dimensionedScalar timeStartLaser
(
    "timeStartLaser",
    dimensionSet(0, 0, 1, 0, 0, 0, 0),
    beamProperties.lookup(”timeStartLaser")
);
```

```
//radius of laser beam
dimensionedScalar rBeam
(
    "rBeam",
    dimensionSet(0, 1, 0, 0, 0, 0, 0),
    beamProperties.lookup(”rBeam")
);
```

```
//laser power
dimensionedScalar Q_L
(
    "Q_L",
    dimensionSet(1, 2, -3, 0, 0, 0, 0),
beamProperties.lookup(”Q_L")
);
```

Solver for free surface thermal flow

Construct and declare new fields and variables

```
Info<< "Reading field Qhs\n" <<endl;
volScalarField Qhs
(
  Ioobject
  (
    ”Qhs",
    runTime.timeName(),
    mesh,
    IOobject::READ_IF_PRESENT,
    IOobject::AUTO_WRITE
  ),
  mesh,
  dimensionedScalar
  (
    "Qhs",
    dimensionSet(1, 0, -3, 0, 0, 0, 0),
    scalar(0.0)
   )
);
```

Solver for free surface thermal flow

Construct and declare new fields and variables

The heat source moves in the computational domain as the simulation progresses. Therefore, the location of heat source need to be updated at each deltaT.

Create a file called `updateLaserFields.H` and add,

- `// To update heat source field at each deltaT`

```
const volVectorField& cellCentre = mesh.C();

volScalarField r_local = Foam::sqrt(
                                    Foam::sqr(cellCentre.component(vector::X)-(Xbeam0.component(vector::X)
                                  +  (Ulaser.component(vector::X)*runTime.time())))
                                  +  Foam::sqr(cellCentre.component(vector::Z)-(Xbeam0.component(vector::Z)
                                  +  (Ulaser.component(vector::Z)*runTime.time())))
                                    );
```

- `Qhs = eta_L*Q_L/(pi*Foam::sqr(rBeam))*dl*Foam::exp(-dl*(Foam::sqr(r_local)/Foam::sqr(rBeam)));`

Solver for free surface thermal flow

Construct and declare new fields and variables

In the final step, include new files to main solver file `interThermalFoam.C`.
Open `interThermalFoam.C` and after `#include "correctPhi.H"`, Add

```
#include "createLaserFields.H"
```

Then at the end of while (pimple.loop()), Add

```
#include "updateLaserFields.H"

#include "TEqn.H"
```

Save and close.

Finally, clean and compile.

```
wclean
wmake
```

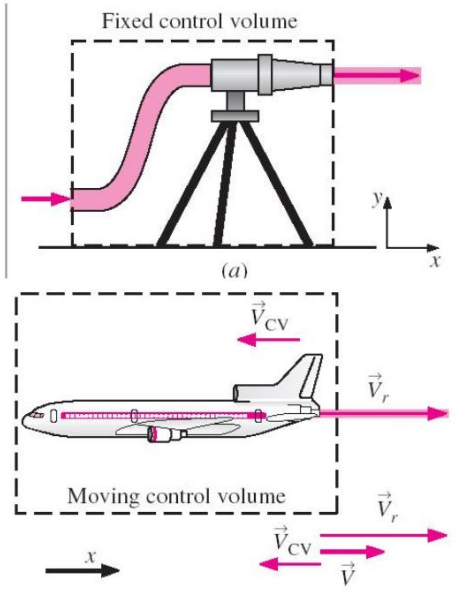
Solver I is formulated and successfully compiled

Theory of moving control volume

Reynolds transport theorem

- The most general Reynolds transport theorem for fixed control volume is given by

$$\frac{d}{dt}(B_{syst}) = \frac{d}{dt}\left(\int_{CV} \beta\rho dv\right) + \left(\int_{CS} \beta\rho(V \cdot n)dA\right)$$



(a) Fixed Control Volume (b) Moving control volume [4]

- However, when control volume moves with constant uniform velocity Vcv, the observed fixed on the control volume expereince relative velocity Vrel given *Vrel = V – Vcv* [4]. Thus RTT is given as

$$\frac{d}{dt}(B_{syst}) = \frac{d}{dt}\left(\int_{CV} \beta\rho dv\right) + \left(\int_{CS} \beta\rho(V_{rel} \cdot n)dA\right)$$

Theory of moving control volume

Motion of heat source in MRF

- In welding application If the heat source moves with speed Uweld, under the principle of relative motion, the heat source can be assumed to be fixed in space whereas the base metal can be assumed to be moving on the negative direction with equal magnitude [5]. Such a case can be modeled in a moving reference frame (MRF). In MRF the velocity of fluid in the domain is given by

$$U_{rel} = U - U_{weld}$$

where *Urel* is the velocity at any point in the computational domain in MRF. *U* is the absolute convective velocity of the fluid and *Uweld* is the uniform velocity of the moving heat source.

Solver in moving reference frame

Creating solver II: interThermalReferenceFoam

- Copy interThermalFoam solver to a new solver named interThermalReferenceFoam

```
cd $WM_PROJECT_USER_DIR/application/solvers/multiphase
cp –r interThermalFoam interThermalReferenceFoam
```

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

```
cd $WM_PROJECT_USER_DIR/application/solvers/multiphase/interThermalReferenceFoam
mv interThermalFoam.C interThermalReferenceFoam.C
```

- Change interThermalFoam to interThermalReferenceFoam everywhere in the .C file with:

```
sed –i s/”interThermalFoam"/”interThermalReferenceFoam"/g interThermalReferenceFoam.C
```

- Change the path for executable in Make/file:

  interThermalReferenceFoam.C

  EXE = ($FOAM_USER_APPBIN)/interThermalReferenceFoam

- Clean to remove the dependency list and compile the solver.

  wclean

  wmake

Solver in moving reference frame

Continuity equation in moving reference frame

- Continuity equation in moving reference frame

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho U) = 0$$

- Replace U by Urel

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho U_{rel}) = 0$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho (U - U_{weld})) = 0$$

This term reduces to 0 since both ρ and U_{weld} are constant

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho U) - \nabla \cdot (\rho U_{weld}) = 0$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho U) = 0$$

So the continuity equation is unchanged in moving reference frame

Solver in moving reference frame

Momentum equation in moving reference frame

Momentum equation

Fixed Reference Frame

$$\frac{\partial}{\partial t}\rho U + \nabla \cdot (\rho U U) = -\nabla p + \nabla \cdot (\mu_{eff}(\nabla U + (\nabla U)^T)) + \rho g + F_S$$

Additional term in moving reference frame

Moving Reference Frame

$$\frac{\partial}{\partial t}\rho U + \nabla \cdot (\rho U U) \boxed{- \rho \cdot (U_{weld} \cdot \nabla U)} = -\nabla p + \nabla \cdot (\mu_{eff}(\nabla U + (\nabla U)^T)) + \rho g + F_S$$

Solver in moving reference frame

Implementation of momentum equation in moving reference frame

Momentum equation in moving reference frame implementation in OF

$$\frac{\partial}{\partial t}\rho U + \nabla \cdot (\rho U U) - \rho \cdot (U_{weld} \cdot \nabla U) = -\nabla p + \nabla \cdot (\mu_e f f(\nabla U + (\nabla U)^T)) + \rho g + F_S$$

- Open UEqn.H and add the additional term on the left hand side of `fvVectorMatrix UEqn.`

```
 fvVectorMatrix UEqn
(

  fvm::ddt(rho, U)
+ fvm::div(rhoPhi, U)
- rho*(fvc::grad(U) & Uweld)
+ MRF.DDt(rho, U)
+ turbulence->divDevRhoReff(rho, U)

 ==
    fvOptions(rho, U)
);

   UEqn.relax();
```

- Save and close the file

Solver in moving reference frame

Energy equation in moving reference frame

Energy equation

Fixed Reference Frame

$$\rho Cp\frac{\partial T}{\partial t} + \nabla \cdot (\rho CpUT) - \nabla \cdot (K\nabla T) = 0$$

Additional terms in moving reference frame

Moving Reference Frame

$$\rho Cp\frac{\partial T}{\partial t} + \nabla \cdot (\rho CpUT) - \nabla \cdot (K\nabla T) = \rho Cp \cdot (U_{weld} \cdot \nabla T)) + T(U_{Weld} \cdot \nabla(\rho Cp))$$

Solver in moving reference frame

Implementation of energy equation in moving reference frame

Implementation of Energy equation in moving reference frame

$$\rho Cp\frac{\partial T}{\partial t} + \nabla \cdot (\rho CpUT) - \nabla \cdot (K\nabla T) = \rho Cp \cdot (U_{weld} \cdot \nabla T)) + T(U_{Weld} \cdot \nabla(\rho Cp))$$

- Open TEqn.H and add the additional term on the right hand side of `fvScalarMatrix TEqn.`

```
 fvScalarMatrix TEqn
(

  fvm::ddt(rhoCp, T)
+ fvm::div(rhoPhiCpf, T)
- fvm::laplacian(kappaf, T)

 ==
   rhoCp * (Uweld & fvc::grad(T))
+ T      * (Uweld & fvc::grad(rhoCp))
);

   TEqn.relax();
```

- Save and close the file

Solver in moving reference frame

alpha equation in moving reference frame

alpha equation

Fixed Reference Frame $$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha U) + \nabla \cdot ((1-\alpha)\alpha U_r) = 0$$

Additional term in moving reference frame

Moving Reference Frame $$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha U) - \boxed{(\nabla \alpha) \cdot U_{weld}} + \nabla \cdot ((1-\alpha)\alpha U_r) = 0$$

Solver in moving reference frame

Implementation of alpha equation in moving reference frame in OF

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha U) - (\nabla \alpha) \cdot U_{weld} + \nabla \cdot ((1-\alpha)\alpha U_r) = 0$$

- Open alphaEqn.H and add the additional term on the left hand side of `fvScalarMatrix alpha1Eqn.`

```
fvScalarMatrix alpha1Eqn
(
    (
        LTS
      ? fv::localEulerDdtScheme<scalar>(mesh).fvmDdt(alph
      : fv::EulerDdtScheme<scalar>(mesh).fvmDdt(alpha1)
    )
 + fv::gaussConvectionScheme<scalar>
   (
       mesh,
      phiCN,
      upwind<scalar>(mesh, phiCN)
   ) .fvmDiv(phiCN, alpha1)
   - (Uweld & fvc::grad(alpha1))

);
 alpha1Eqn.solve();
```

- Save and close the file

Solver in moving reference frame

Construct and declare new fields and variables

- Now we need to construct and initialize a new variable named `Uweld`.
  Open `createLaserFields` and below `dimensionedVector Ulaser`, add

```
Info<< "Reading field Uweld\n" <<endl;
volVectorField Uweld
(
  Ioobject
  (
    ”Uweld",
    runTime.timeName(),
    mesh,
    IOobject::NO_READ,
    IOobject::NO_WRITE
  ),
  mesh,
  dimensionedVector("Uweld",dimensionSet(0,1,-1,0,0,0,0), Ulaser.value())
);
```

Solver in moving reference frame

Construct and declare new fields and variables

- Modify the equation of `r_local` since the heat source remains fixed in this solver.

- Open `createLaserFields.H` and modify the equaton of `r_local` to

```
// For fixed heat source

const volVectorField& cellCentre = mesh.C();
volScalarField r_local = Foam::sqrt(
                                    Foam::sqr(cellCentre.component(vector::X)-(Xbeam0.component(vector::X)
                                  +  Foam::sqr(cellCentre.component(vector::Z)-(Xbeam0.component(vector::Z)
                                  );
```

Solver in moving reference frame

Construct and declare new fields and variables

- Now Open the main solver file `interThermalReferenceFoam.C` and comment or delete `#updateLaserFields.H` in the pimple loop.
- Save and close the file.
- Clean and compile.

```
wclean
wmake
```

Solver II is formulated and successfully compiled

Formulating and implementing new BC

- The new periodically recurring and moving BC for droplet is applied on the top atmospheric surface for drop injection
- Start by copying a similar existing BC. In this case oscillatingFixedValue BC is copied and modified for BC for drop injection

```
mkdir -p $WM_PROJECT_USER_DIR/src/myBCs/myFiniteVolume/fields/fvPatchFields/derived
cd $WM_PROJECT_USER_DIR/src/myBCs/myFiniteVolume/fields/fvPatchFields/derived
cp -r $WM_PROJECT_DIR/src/finiteVolume/fields/fvPatchFields/derived/oscillatingFixedValue .
```

- Change name of the new BC folder and all the files inside

```
mv oscillatingFixedValue localTranslatingFixedValue
cd localTranslatingFixedValue
mv oscillatingFixedValueFvPatchField.H localTranslatingFixedValueFvPatchField.H
mv oscillatingFixedValueFvPatchField.C localTranslatingFixedValueFvPatchField.C
mv oscillatingFixedValueFvPatchFields.H localTranslatingFixedValueFvPatchFields.H
mv oscillatingFixedValueFvPatchField.C localTranslatingFixedValueFvPatchFields.C
mv oscillatingFixedValueFvPatchFieldFwd.H localTranslatingFixedValueFvPatchFieldFwd.H
```

- Replace all the 'oscillating' string in all the files with localTranslating

```
Sed -i s/oscillating/localTranslating/g localTranslatingFixedValueFvPatchField*
```

- Create Make/files

```
cd $WM_PROJECT_USER_DIR/src/myBCs
cp -r $WM_PROJECT_DIR/src/finiteVolume/Make .
```

Formulating and implementing new BC

- Remove all the lines of code in Make/file and replace with

```
fvPatchFields = myFiniteVolume/fields/fvPatchFields
derivedFvPatchFields = $(fvPatchFields)/derived
$(derivedFvPatchFields)/localTranslatingFixedValue/localTranslatingFixedValueFvPatchFields.C
LIB = $(FOAM_USER_LIBBIN)/libmyBCs
```

Formulating and implementing new BC

- Declare new parameters in `localTranslatingFixedValueFvPatchField.H` file
- Replace the private data with

```
Field<Type> dropInletValue_;
Field<Type> dropOutsideValue_;
List<vector> dropCentre0_;
List<vector> dropCentre_;
List<scalar> dropRadius_;
Field<vector> faceCentres_;
label curTimeIndex_;
List<vector> dropTranslatingVelocity_;
scalar dropFrequency_;
```

- Comment or delete existing private member function

```
// Private Member Functions
//- Return current scale
//scalar currentScale() const;
```

Formulating and implementing new BC

- Replace the member function with

```
Too long to be included here. Please refer to the report.
```

- Save and close `localTranslatingFixedValueFvPatchField.H`

- Open `localTranslatingFixedValueFvPatchField.C` and
  - Modify constructor and the member function

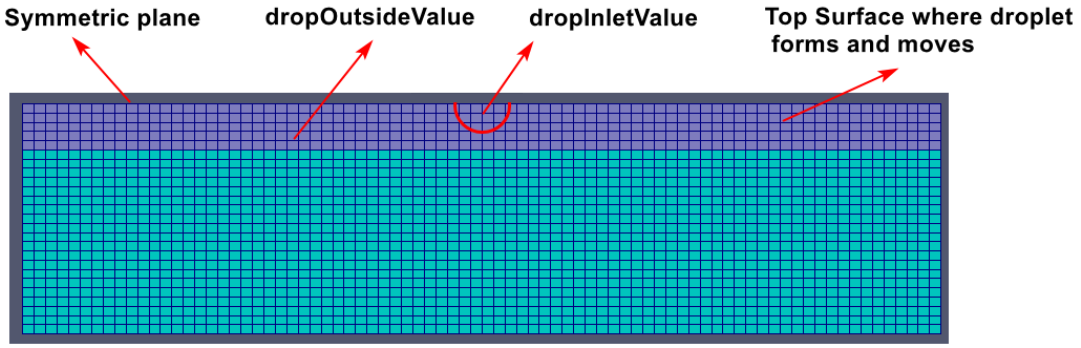
```
Too long to be included here. Please refer to the report.
```

- Clean and compile

```
cd $WM_PROJECT_USER_DIR/SRC/myBCs
wclean
wmake
```

Formulating and implementing new BC

- The equation to be solved for periodic injection of droplet is written as



Top atmosphere surface BC for drop injection

```
float tinj_ = 1/(2*dropFrequency_);
const int ninj = floor(t_/tinj_);

fvPatchField<Type>::operator==(dropInletValue_);
    {
        {
          if(mag(this->faceCentres()[i]-dropCentre_[j])<=dropRadius_[j]
          && (t_ > 0.5 && ninj % 2 == 0)){isDrop=true;}

        }
         if(!isDrop){patchfield.data()[i]= dropOutsideValue_[i];}
    }
```

- Clean and compile

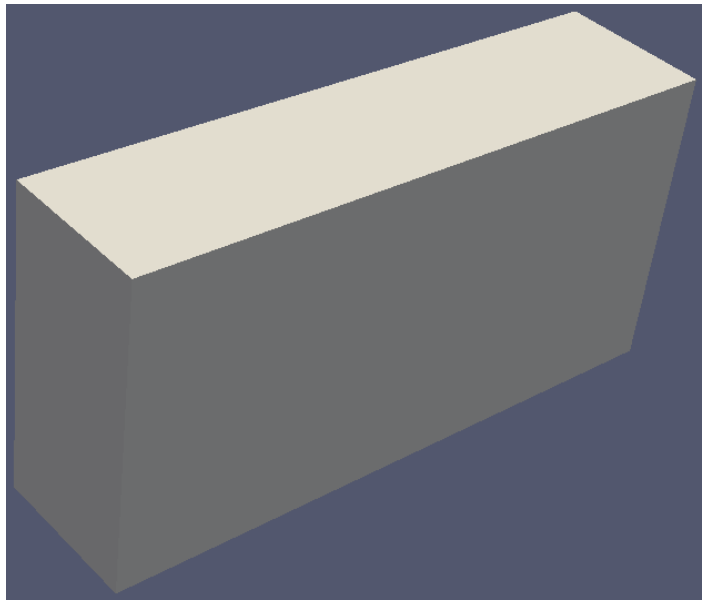
```
cd $WM_PROJECT_USER_DIR/SRC/myBCs
wclean
wmake
```

*BC code only partially shown for simplicity
*Please refer to the report for complete BC

Setting up Test case

Test Case I : Importing teste case

- Test case I is called movingLaser and it is solved using solver interThermalFoam
- Copy an existing test case tutorial of damBreak and rename it to movingLaser.
- Change the blockMeshDict file by accessing it using `vi system/blockMeshDict` . Modify the vertices to obtain the computation as shown in figure.



Computation domain

Setting up Test case

Test Case I: Adding/modifying constant properties

- In transport properties, change air and water phases to argon and steel phases respectively. Also change the value of density, viscosity and surface tension.
- Create a file named thermophysicalPropertiesPhase1 in constant/ and specify heat capacity cp1 and thermal conductivity kappa1 for steel.
- Copy thermophysicalPropertiesPhase1 and to a new file named thermophysicalPropertiesPhase2 and specify specify heat capacity cp2 and thermal conductivity kappa2 for argon.
- Create a file called beamProperties in constant/ and specify beam properties.

Setting up Test case

Test Case I: Adding/modifying files for initial and BC

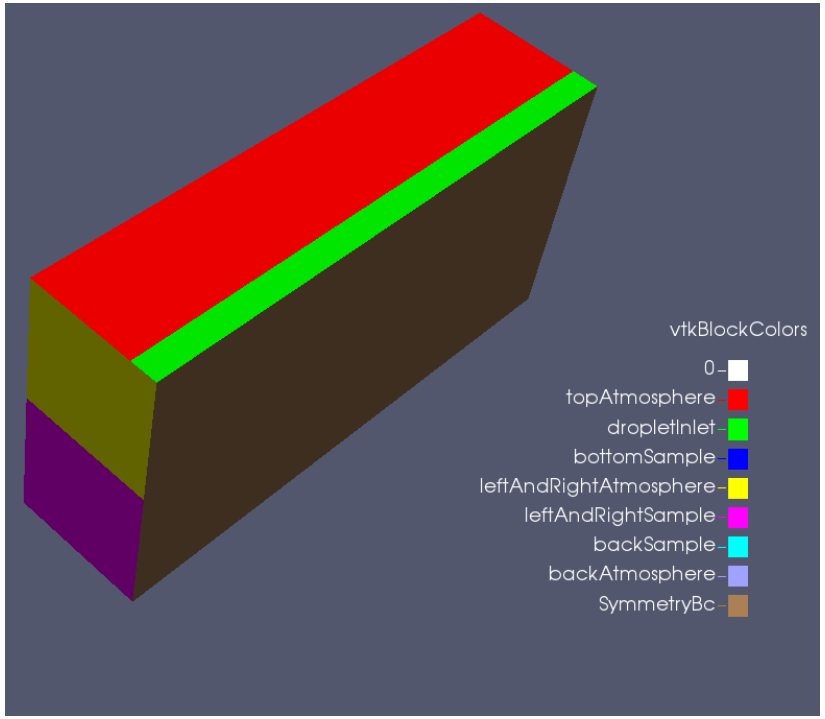
- Create T file in 0/ folder and specify initial and boundary conditions for temperature.
- The new BC is applied on the dropletInlet surface for U and for alpha:

- In 0/U for dropletInlet patch

```
dropletInlet
{
      type localTranslatingFixedValue;
      value                           uniform (0 0 0);
      dropInletValue                  uniform (0.0 -1.0 0);
      dropOutsideValue                uniform (0 0 0);
      dropCentre0                     ((0.01 0.01 0.0));
      dropCentre                      ((0.01 0.01 0.0));
      dropRadius                      (0.0006);
      dropTranslatingVelocity         ((0.01 0 0));
      dropFrequency                   167;
}
```

- In 0/alpha.steel.org for dropletInlet patch

```
dropletInlet
{
      type                            localTranslatingFixedValue;
      value                           uniform (0 0 0);
      dropInletValue                  uniform 1;
      dropOutsideValue                uniform (0 0 0);
      dropCentre0                     ((0.01 0.01 0.0));
      dropCentre                      ((0.01 0.01 0.0));
      dropRadius                      (0.0006);
      dropTranslatingVelocity         ((0.01 0 0));
      dropFrequency                   167;
}
```



BC Patch

Setting up Test case

Test Case I: Modification of control parameters, fvSchemes and fvSolution

- In system/controlDict, modify some of the control parameters.

```
deltaT 1.0e-5;
writeInterval 0.001;
maxCo 0.8;
maxAlphaCo 0.8;
maxDeltaT 1.0e-3;
```

- Add `libs ("libmyBCs.so");` at the end of the controlDict file.

- At the end of divSchemes, add divergence scheme for T in fvSchemes:

```
div(rhoPhiCpf,T) Gauss Minmod;
```

- Add solver and predonditioner for T in fvSolution.

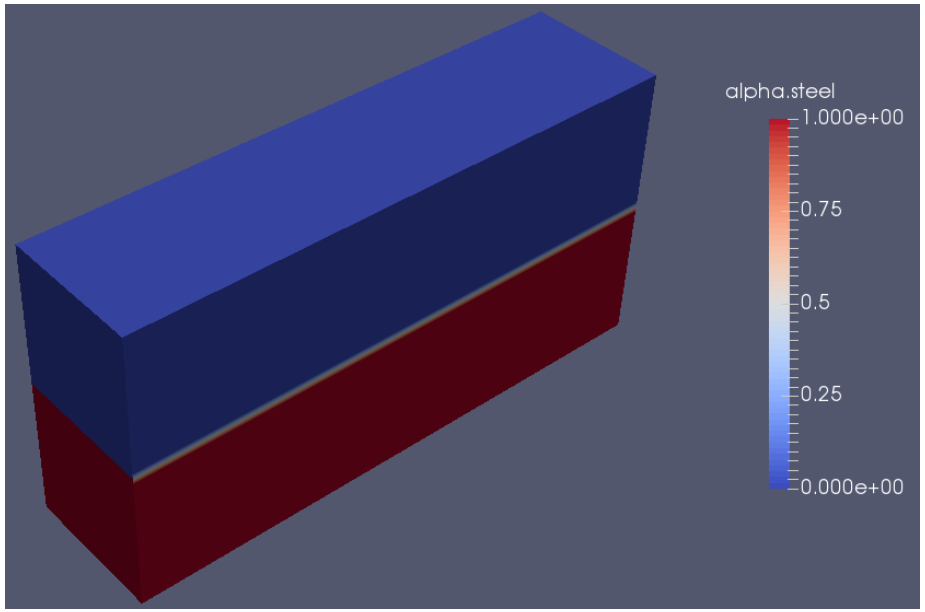
```
T
{
    solver BICCG;
    preconditioner DILU;
    tolerance 1.0e-6;
    relTol 0.0001;
}
TFinal
{
    $T;
    relTol 0.0;
}
```

Setting up Test case

Test Case I: Runnng the case

- Initialize the fieldS to obtain steel and argon phase as shown in the figure.

```
cp 0/alpha.steel.org 0/alpha.steel
setFields
```



Initial Fields

- Run the test case I and output the logfile with:

```
interThermalFoam >& log.interThermalFoam &
```

Setting up Test case

Test Case II : Importing teste case

- Test case II is called movingFrame and it is solved with interThermalReferenceFoam solver
- Copy test case movingLaser to movingFrame.

```
cp –r movingLaser movingFrame
```

- Change `dropTranslatingVelocity` value in 0/U and 0/alpha.steel.org to ((0,0,0))
- Initialize the fields

```
cp 0/alpha.steel.org 0/alpha.steel
setFields
```

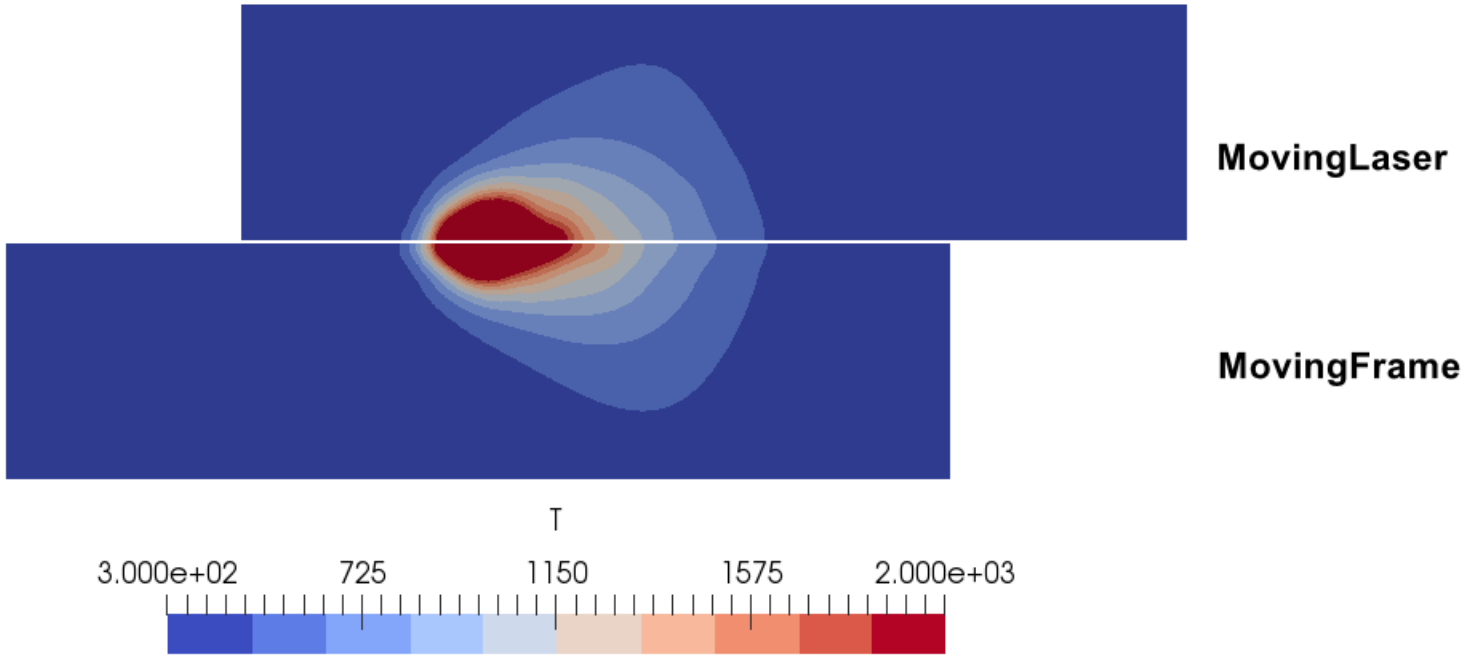
- Run the test case II and output the logfile to

```
cd movingFrame
interThermalReferenceFoam >& log.interThermalReferenceFoam &
```

- When simulation is completed. Results are visualized using paraview by typing `paraFoam`.

Results

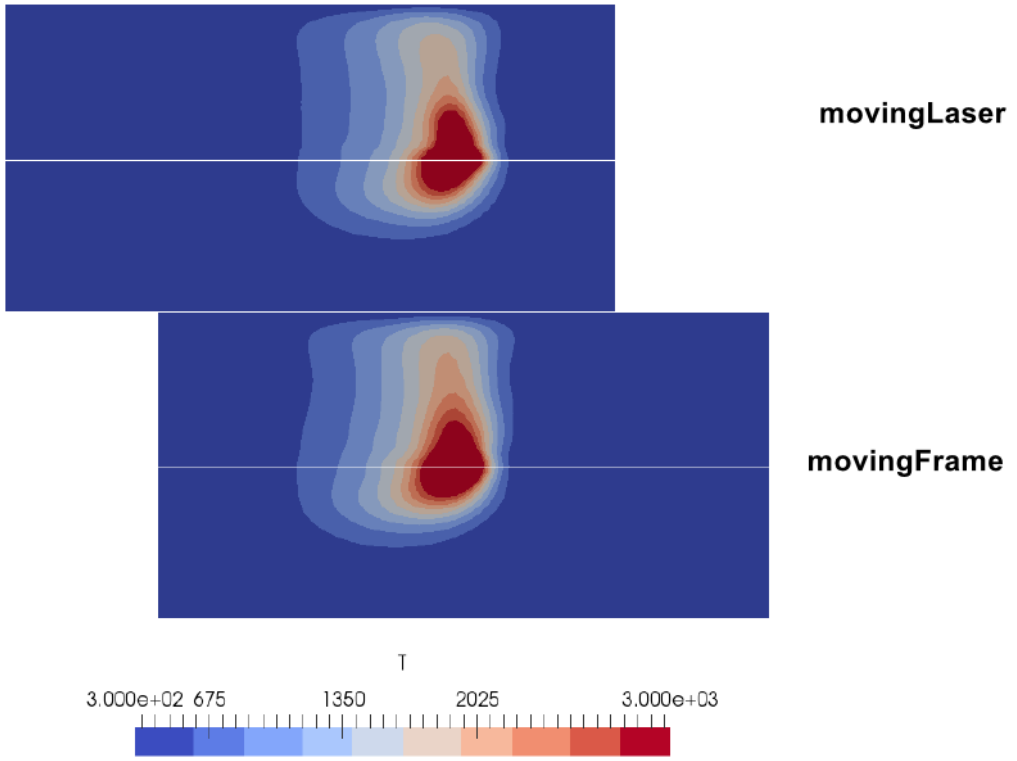
Comparison of temperature distributions



Temperature distribution on top steel surface at 0.5 s

Results

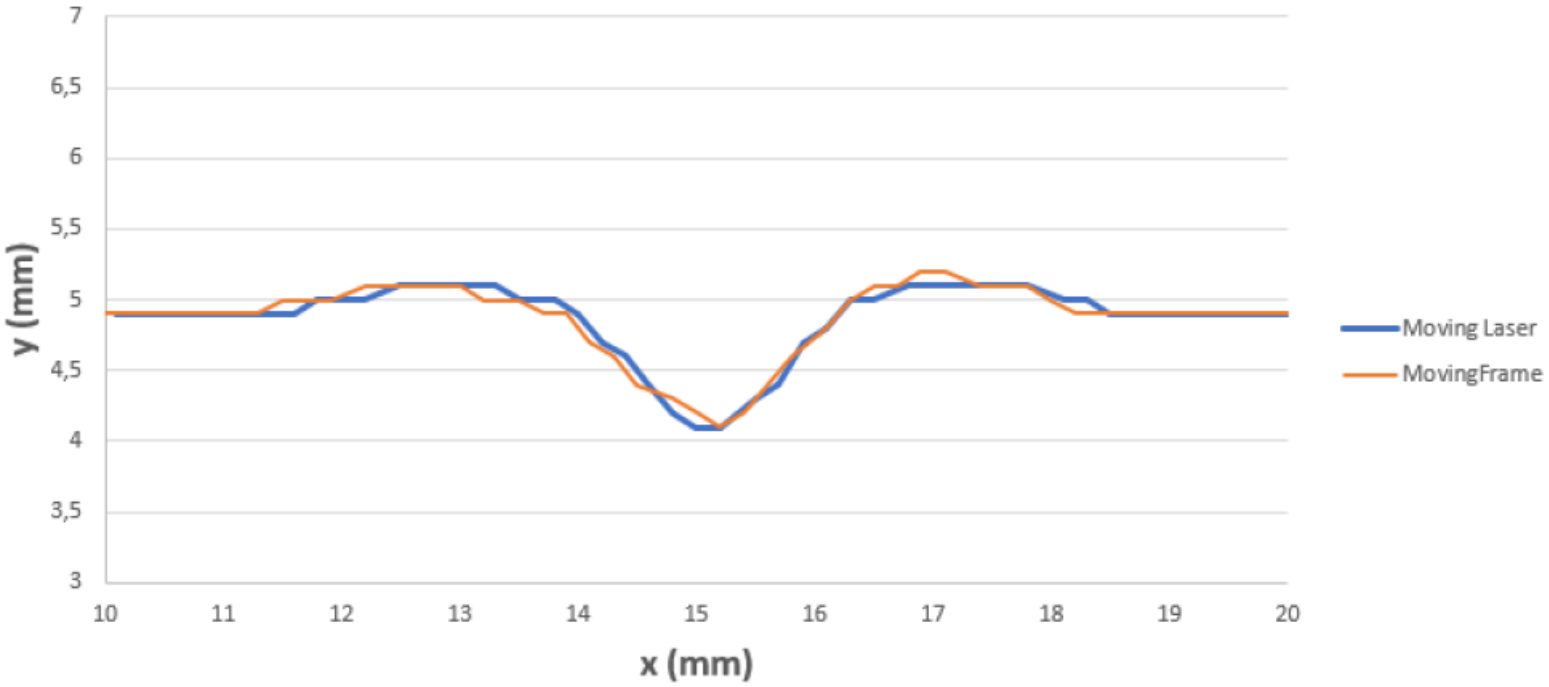
Comparison of temperature distributions



Temperature distribution along symmetrical plane at 0.5 s

Results

Comparison of surface deformations



Surface deformation due to droplet impact at 0.52 s

Conclusion

- Temperature distribution on steel surface and along the symmetry plane agrees very well between the two cases.
- The surface deformation between two test cases agree very well.
- The simulation time for a test case in moving reference frame is 11.25% lower than for a test case in fixed coordinate system.
- The derivation of governing equations in moving reference frame are accurate to model relative motion between two bodies.

Questions are welcome !

Thank You

References

[1] Akash Aggarwal and Arvind Kumar. Particle scale modelling of selective laser melting-based additive manufacturing process using open-source cfd code openfoam. Transactions of the Indian Institute of Metals, 71(11):2813{2817, 2018.

[2] Vaibhav K Arghode, et al. Computational modeling of gmaw process for joining dissimilar aluminum alloys. Numerical Heat Transfer, Part A: Applications, 53(4):432{455, 2008.

[3] Solana, Pablo, and Guillermo Negro. "A study of the effect of multiple reflections on the shape of the keyhole in the laser processing of materials." *Journal of Physics D: Applied Physics* 30.23 (1997): 3216.

[4] https://www.kau.edu.sa

[5] Hu, Renzhi, et al. Metal transfer in wire feeding-based electron beam 3D printing: Modes, dynamics, and transition criterion. *International Journal of Heat and Mass Transfer* 126 (2018): 877-887.