Implement boundary conditions in OpenFOAM

FOAM@Iberia 2023

R. Ribeiro, B. Ramoa, R. Costa, J.M. Nóbrega – University of Minho

contact: ricardoribeiro1616@gmail.com

This offering is not approved or endorsed by OpenCFD Limited, the producer of the OpenFOAM software and owner of the OPENFOAM@ and OpenCFD@ trademarks

Computational Rheology @IPC

University of Minho School of Engineering

INSTITUTE FOR POLYMERS AND COMPOSITES

Outline

1. **Implement a standard boundary condition in OpenFOAM**
   - Objectives
   - Procedure
   - Useful commands/functions
2. **Implement a coded boundary condition in OpenFOAM**
   - Objectives
   - Procedure
3. **Conclusions**
4. **Acknowledgments**

1

Implement a standard boundary condition in OpenFOAM

Objectives

Implement a standard boundary condition in which the temperature of the represented red *patch* varies according the mean temperature (T) of all domain cells.

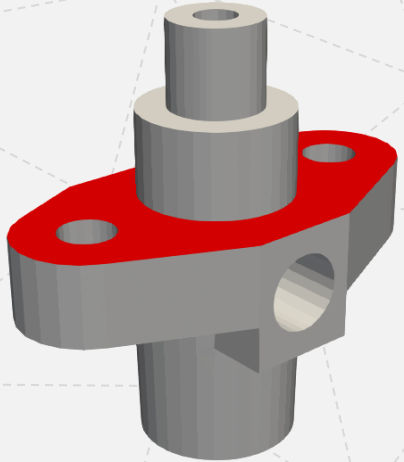
If the mean temperature of all domain cells is lower than a switch value (*TSwitch*):

THeating is applied

else

TCooling is applied

- THeating, TCooling and TSwitch must be read from the dictionary T (boundary field for temperature).



Geometry retrieved from *flange* tutorial of *laplacianFoam* solver

Based on: Basic OpenFOAM Programming Tutorial: Writing a Custom Boundary Condition – Vuko Vukcevic

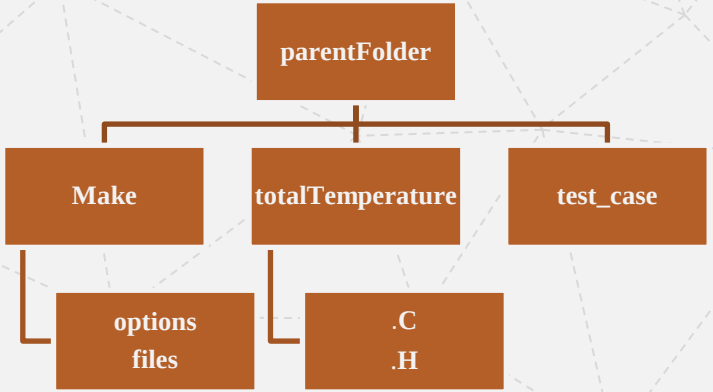
Procedure

Steps to follow

1) Copy a boundary condition from *OpenFOAM/OpenFOAM-vxxxx/src/finiteVolume/fields/fvPatchFields/…* (do not change original boundary conditions!). If possible, choose a boundary condition with similarities to what you want to do;

2) Place the BC inside a folder with a custom name;

3) Copy also the Make folder from *OpenFOAM/OpenFOAM-vxxxx/src/finiteVolume/Make;*



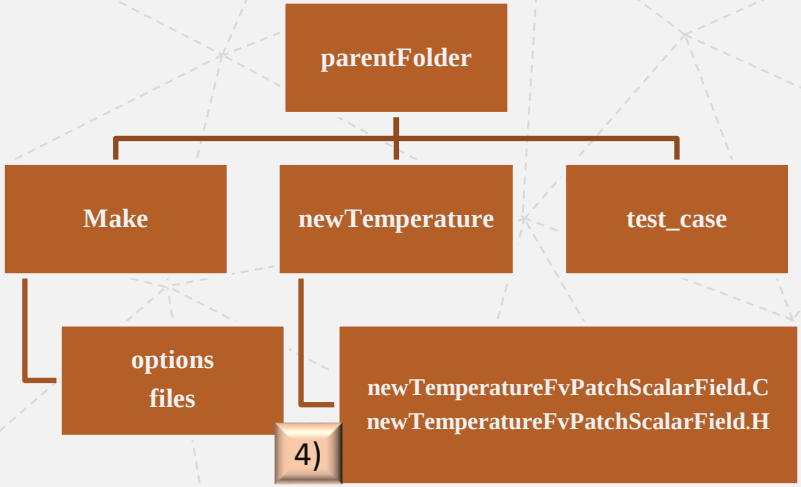
Procedure

Steps to follow

4) Adapt BC folder name and ".C" and ".H" file names in order to be suggestive;

5) Open ".C" and ".H" files and replace all instances of the old names by the new ones (using Ctrl+H may be helpful);

6) Adapt "*files*" and "*options*" inside Make folder;



5)

```
35.     // * * * * * * * * Constructors  * * * * * * * * //
36.     Foam::newTemperatureFvPatchScalarField::newTemperatureFv
37.     PatchScalarFied
38.     (
39.       const fvPatch& p,
40.       const DimensionedField<scalar, volMesh>& iF
41.     )
42.     :

...
```

6)

```
1.      newTemperature/newTemperatureFvPatchScalarField.C
2.
3.      LIB = $(FOAM_USER_LIBBIN)/libMyBoundaryConditions
```

```
1.      EXE_INC = \
2.      -I$(LIB_SRC)/fileFormats/lnInclude \
3.      -I$(LIB_SRC)/surfMesh/lnInclude \
4.      -I$(LIB_SRC)/meshTools/lnInclude \
5.      -I$(LIB_SRC)/dynamicMesh/lnInclude \
6.      -I$(LIB_SRC)/finiteVolume/lnInclude
7.      LIB_LIBS = \
8.      -lOpenFOAM \
9.      -lfiniteVolume
```

Procedure

Steps to follow

Adapt our *.H* file, including:
7) Add the private members we are going to need to our class;
8) Add member functions to access those private members;

7)

```
77.     class newTemperatureFvPatchScalarField
78.     :
79.     public fixedValueFvPatchScalarField
80.     {
81.     // Private data

83.     //Switch value of T
84.     scalar TSwitch_;

86.     //Cooling temperature
87.     scalar TCooling_;

89.     //Heating temperature
90.     scalar THeating_;

    . . .
```

8)

```
160.    // Member functions

162.    // Access

164.    //- Return switch temperature
165.    scalar TSwitch() const
166.    {
167.      return TSwitch_;
168.    }

170.    //- Return Cooling temperature
171.    scalar TCooling() const
172.    {
173.      return TCooling_;
174.    }

176.     //- Return Heating temperature
177.    scalar THeating() const
178.    {
179.      return THeating_;
180.    }
```

Procedure

Steps to follow

Adapt our .C file, including:
9) **Adapt constructors of the class;**
10) Adapt member functions;

```
1.      // * * * * * * * * * * * * * Constructors  * * * * * * * * * * * //

3.      Foam::newTemperatureFvPatchScalarField::newTemperatureFvPatchScalarField
4.      (
5.       const fvPatch& p,
6.       const DimensionedField<scalar, volMesh>& iF
7.      )
8.      :
9.      fixedValueFvPatchScalarField(p, iF),
10.     TSwitch_(450),
11.     TCooling_(300),
12.     THeating_(600)

14.     {}
```

Useful commands/functions

Steps to follow

Adapt our .C file, including:
10) **Adapt member functions**

Let's explore some useful commands/functions!

Procedure

Steps to follow

Adapt our .C file, including:
9) Adapt constructors of the class;
10) **Adapt member functions – In this case, we are dealing with a fixedValue type boundary condition, and we only need to take care about updateCoeffs() and write() functions.**

```
171.    void Foam::newTemperatureFvPatchScalarField::write(Ostream& os) const
172.    {
173.        fvPatchScalarField::write(os);
174.        os.writeKeyword("TSwitch") << TSwitch_ << token::END_STATEMENT << nl;
175.        os.writeKeyword("TCooling") << TCooling_ << token::END_STATEMENT << nl;
176.        os.writeKeyword("THeating") << THeating_ << token::END_STATEMENT << nl;
177.        writeEntry("value", os);
178.    }
```

```
140.    void Foam::newTemperatureFvPatchScalarField::updateCoeffs()
141.    {
142.        if (updated())
143.        {
144.            return;
145.        }
146.    //Acess the dimensioned internalField
147.        const DimensionedField<scalar, volMesh>& tInt =
148.                this->internalField();
149.    //Calculate cell volume weighted average of the field
150.        const dimensionedScalar tAverage =
151.                tInt.weightedAverage(patch().boundaryMesh().mesh().V());
152.    //set the fixed value boundary condition according specified switch value
153.    //and calculated average value
154.        if(tAverage.value() < TSwitch_)
155.        {
156.            operator==(THeating_);
157.        }
158.        else
159.        {
160.            operator==(TCooling_);
161.        }
162.        fixedValueFvPatchScalarField::updateCoeffs();
163.    }
```

Procedure

Steps to follow

11) Compile the boundary condition;
12) Add the new library to the test case *controlDict;*
13) Adapt temperature BC and run the case!!

11) Command:
wmake libso

12)
```
libs ("libMyBoundaryConditions.so");
```

13)
```
type            newTemperature;
value           uniform 350;
TSwitch         400;
TCooling        350;
THeating        650;
```

Let's check the results on paraview!

2

Implement a coded boundary condition in OpenFOAM

Objectives

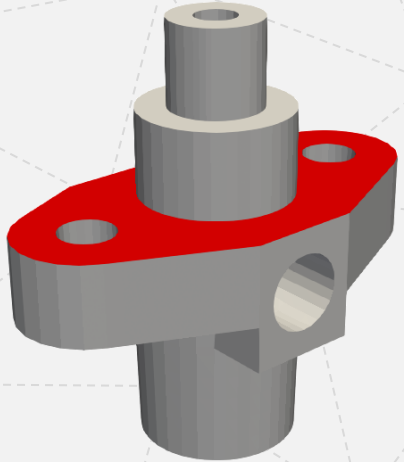
Implement a coded boundary condition for the same problem and compare results!

If the mean temperature of all domain cells is lower than a switch value (*TSwitch*):

THeating is applied

else

TCooling is applied



Geometry retrieved from *flange* tutorial of *laplacianFoam* solver

Procedure

Steps to follow

1 – All code must be generated inside “T” file, namely:

- Get mean value of temperature in all domain cells;
- Create variables: TSwitch, TCooling and THeating;
- Create a conditional statement in which we decide if we should heat or cool our domain.

```
27.     patch2
28.     {
29.        type            codedFixedValue;
30.        value uniform   350;
31.        name            newTemperature_coded;
32.        code
33.        #{
34.     //Access the dimensioned internalField
35.        const DimensionedField<scalar, volMesh>& tInt =
36.                this->internalField();
37.     //Calculate cell volume weighted average of the field
38.        const dimensionedScalar tAverage =
39.                tInt.weightedAverage(patch().boundaryMesh().mesh().V());
40.     //set the fixed value boundary condition
41.        scalar TSwitch_(400);
42.        scalar TCooling_(350);
43.        scalar THeating_(650);
44.        if (tAverage.value() < TSwitch_)
45.           {
46.              operator==(THeating_);
47.           }
48.        else
49.           {
50.              operator==(TCooling_);
51.           }
52.        #};
53.     }
```

Conclusions

Why don't we always proceed with coded boundary conditions?

- They are not available in foamExtend

Which standard BC should we copy before adapt?

- A similar BC to what we want to do

In which functions can we make our calculations?

- updateCoeffs() and evaluate()

Acknowledgments

The authors would like to acknowledge all sponsors of FOAM@Iberia 2023 and FCT - Portuguese Foundation for Science and Technology, Reference PhD grant 2022.11884.BD.

END

Thank you!!!

CRheo@IPC