OpenFOAM

Tulio Rodarte Ricciardi

January 7, 2025

Contents

1 TODO

- transport models

library up to openfoam 6

OpenFOAM-8 “working” version

- sponge (?)

2 Building

2.1 Pre-reqs

Setup environment

- DANE:
  ```
  module unload intel-classic/2021.6.0-magic mvapich2/2.3.7
  module load gcc/11.2.1 openmpi/4.1.2
  ```
  check that cmake exists

- Campus Cluster:
  ```
  module load cmake/3.26.3
  module unload gcc/11.2.0
  module load openmpi/.4.1.4-gcc-12.2.0
  ```
  `src/OpenFOAM/db/IOstreams/hashes/`: see **link**

- Local installation:
  ```
  sudo apt install flex
  ```

openMPI

Install openMPI

Cantera

```
python3 -m pip install cantera
```

gmsh

```
cd /g/g92/<your-username>
 wget https://gmsh.info/bin/Linux/gmsh-4.13.1-Linux64.tgz
 tar zxv gmsh-4.13.1-Linux64.tgz
 export PATH="$PATH:/g/g92/<your-username>/gmsh-4.13.1-Linux64/bin"
```

2.2 openFOAM

1. Download openFOAM:
   OFversion=...
   wget -O - http://dl.openfoam.org/third-party/${OFversion} | tar zxv
   mv ThirdParty-${OFversion}-version-${OFversion} ThirdParty-${OFversion}
   wget -O - http://dl.openfoam.org/source/${OFversion} | tar zxv
   mv OpenFOAM-${OFversion}-version-${OFversion} OpenFOAM-${OFversion}

2. Build openFOAM:
   source <path>/OpenFOAM-${OFversion}/etc/bashrc
   ./Allwmake <-j np>

3. Add to .bashrc for future use (or as an alias):
   source <path>/OpenFOAM-${OFversion}/etc/bashrc

2.3 Adding a new solver

2.3.1 reactingDNS

1. Needs OpenFOAM 8, so follow the previous step with `OFversion=8` but do not build it. Just download everything.

2. Copy the folder `reactingDNS` from `<github>/applications/solvers/combustion` to `<path>/OpenFOAM-8/applications/solvers/combustion`

3. Sometimes, this file gives issue. I noticed that, independent on the version, it always works fine with the updated file:
   ```
   cd <path>/OpenFOAM-8/src/OpenFOAM/db/IOstreams/hashes/
   mv OSHA1stream.H _OSHA1stream.H
   mv <github>/OpenFOAM-8/src/OpenFOAM/db/IOstreams/hashes/ .
   ```

4. Source and build OpenFOAM 8

5. Use the files in `<github-folder>/reactingDNS` to run the simulation

2.4 PATO

- Build PATO:
  wget https://github.com/nasa/pato/archive/refs/tags/3.1.tar.gz
  tar -zxvf 3.1.tar.gz
  export PATO_DIR=$(pwd)/PATO/pato-3.1
  source <path>/OpenFOAM-7/etc/bashrc
  source $PATO_DIR/bashrc
  cd $PATO_DIR/src/thirdParty/mutation++
  See instruction in 2.5 for `mutation++`
  ln -s $(pwd)/thirdparty/eigen/Eigen/ ./install/include
  cd $PATO_DIR
  ./Allwmake

- Alias in .bashrc

2.5 Mutation++

If outside PATO:

```
git clone https://github.com/mutationpp/Mutationpp.git
```

Build Mutation:

```
cmake -DCMAKE_INSTALL_PREFIX:PATH=<path>/install
make -j 4 install
```

If installing inside PATO, it is done.

Else, if stand-alone, add to .bashrc:

```
export MPP_DIRECTORY=<path>
export MPP_DATA_DIRECTORY=$MPP_DIRECTORY/data
export PATH=$MPP_DIRECTORY/install/bin:$PATH
export LD_LIBRARY_PATH=$MPP_DIRECTORY/install/lib64:$LD_LIBRARY_PATH
```

Can check just in case:

```
checkmix air_11
```

2.6 TODO: (ignore this)

swak4Foam

swak4Foam

3 Environment configuration and running

3.1 Dane:

Before running OpenFOAM:

```
module unload intel-classic/2021.6.0-magic mvapich2/2.3.7
module load gcc/11.2.1 openmpi/4.1.2
source /g/g92/<your-username>/openFOAM/OpenFOAM-X/etc/bashrc
cd <path/to/the/case>
```

A whole bunch of configuration (i.e., case-dependent)

```
sh setFields.sh
```

Run with this (add to the queue system file):

```
mpirun -np 3 buoyantReactingFoam -parallel
```

4 Setup

4.1 Basic steps

Steps for running openFOAM simulations:

```
gmshToFoam mesh.msh
> update constant/boundary with wedge if axisymmetric
> update constant/boundary with wall may be necessary
create 0 folder with initial and boundary conditions
> setFields -region flow for non-uniform IC
splitMeshRegions -cellZones -overwrite
> decomposePar -allRegions if running in parallel
```

4.2 Mesh generation

`gmshToFoam mesh.msh`

Quads that touch the boundaries with a single node gives trouble and make a new "boundary" (defaultFaces). This may lead to issues on the simulation.

So far, the best I could do was to run mixed meshes, with quads/hexes on well behaved regions (transfinite) and triangles everywhere else.

4.2.1 blockMesh

`paraFoam -block`

4.2.2 axisymmetry

Using `writeFormat binary;` in `controlDict` helps depending on openFOAM version
When meshing, add two separate boundaries: front and back
Add `wedge` as BC for all variables in front and back
in `polyMesh/boundary`, modify from `patch` to `wedge` where necessary
link

PS: I see non-zero Z-velocity on the wedge faces.

4.3 Initial conditions

Units: [kg m s K mol A cd]

In an incompressible solver, N-S equation is divided by a uniform density rho. This causes (1) the dimension of pressure of [0 2 -2 0 0 0 0] and (2) the kinematic viscosity ν in Laplacian term. In an incompressible solver, pressure is assumed to be relative. The atmosphere will be 0 usually.

In a compressible solver, N-S equation is not divided by density. So, the dimension of pressure is [1 -1 -2 0 0 0 0] as usual. The dynamic viscosity μ appears in Laplacian term. In a compressible solver, the absolute pressure must be provided in p file because the value of pressure will be used to calculate other physical properties. The atmosphere will be O(1e5) usually.

4.4 Boundary conditions

4.4.1 Variable BCs

custom boundary conditions:
link
link

4.4.2 Pressure BC + gravity

Walls :: use `fixedFluxPressure` or `fixedFluxExtrapolatedPressure` (instead of `zeroGradient`) or it will give issues.

Outlet :: To satisfy the characteristics and avoid under/over-specification, gotta prescribe U and T (potentially Y) when there is reversed flow in the outlet; pressure is extrapolated by using a zero-gradient BC. For outflow, the internal properties are extrapolated but pressure is prescribed.

See inletOutlet

See fixedMeanOutletInlet

Far-field :: Potentially, using the non-reflective `waveTransmissive` may help suppress some weird(-but-small) behavior (which may be waves reflection). This would be useful in the sides/bottom, but **may not** be robust in the outlet. Additionally, the extra source terms (damping and isentropic) can be used in the bottom and right to damp velocity fluctuations and the initial vortex for increased robustness.

```
farfield
{
    type            waveTransmissive;
    gamma           1.4;
    fieldInf        101325.0;
    lInf            0.1;
    value           uniform 101325.0;
}
```

4.4.3 time-dependent BCs

Linear interpolation from t1 to t2

```
inlet
{
    type uniformFixedValue;
    uniformValue table
    (
    (0.0 (0 0.02 0))  // t1 (Ux_1 Uy_1 Uz_1)
    (2.5 (0 0.10 0))  // t2 (Ux_2 Uy_2 Uz_2)
    );
}
```

4.4.4 Robin BC

https://www.cfd-online.com/Forums/openfoam/219113-how-access-boundary-conditions-species-react

— —

https://foamingtime2.wordpress.com/wp-content/uploads/2017/07/convection-bc_1.pdf

https://www.cfd-online.com/Forums/openfoam-pre-processing/225340-mixed-boundary-condition.htm

https://www.cfd-online.com/Forums/openfoam/238186-implementation-robin-boundary-condition-ope

https://www.cfd-online.com/Forums/openfoam-solving/69464-robin-b-c-mixed-b-c.html

https://www.cfd-online.com/Forums/openfoam-solving/83194-mixed-boundary-condition.html

https://www.cfd-online.com/Forums/openfoam-pre-processing/74593-mixed-bc-heat-transfer-laplacia

4.4.5 change of BCs

change BC
foamGetDict

4.5 Chemistry

4.5.1 mechanisms

need `yaml2ck` from Cantera

```
$>> chemkinToFoam uiuc_20sp.dat thermo_uiuc_20sp.dat ...
transportProperties ../constant/reactions ...
../constant/thermo.compressibleGas
```

This may still present issues with chemkin format (e.g., spacing) and it may be necessary to fix by hand

4.5.2 possible mechanism simplification

see Sandia flame example in reactingFoam > RAS

4.6 Radiation

`system/flow/fvOptions`

```
radiation
{
    type            radiation;
    libs            ("libradiationModels.so");
}
```

4.6.1 viewFactor

`faceAgglomerate -region flow`

`viewFactorsGen -region flow`

```
flow_to_solidLeft
{
    type           greyDiffusiveRadiationViewFactor;
    qro            uniform 0;
    emissivityMode solidRadiation;
    value          uniform 0;
    emissivity     uniform 1.0; // Emissivity of the wall
}
```

4.6.2 P1 model

`constant file`

`fvSolution`

`G file`

Species coefficients

https://boyaowang.github.io/boyaowang_OpenFOAM.github.io/2020/09/16/P1_model/

https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/node112.htm

https://www.cfd-online.com/Forums/openfoam-solving/219601-adding-radiations-chtmultiregionsimpl

https://www.cfd-online.com/Forums/openfoam/78052-boundary-condition-p1-marshakradiation.html

https://www.cfd-online.com/Forums/openfoam-programming-development/135502-understanding-ma

https://www.cfd-online.com/Forums/openfoam/183686-radiation-boundary-conditions-flow-through-b

https://www.cfd-online.com/Forums/openfoam-solving/240250-p1-model-no-participating-media-heat

https://www.cfd-online.com/Forums/openfoam-solving/216879-radiation-models-general-p1-implemen

4.7 System

4.7.1 fvSchemes

fvSchemes

4.7.2 fvSolution

PISO - PIMPLE - SIMPLE

convergence:

In some case, if tolerances are too big, the solution may look a bit off, with pressure oscillations and spurious velocity fields

Also, it may happen that the flow won't evolve if it is "converged" due to a high tolerance, with 0 iterations in the system solvers

In both cases, just modify the `fvSolution` files to have smaller tolerances.

PIMPLE:

Looks like 2 (or more) `nCorrectors` loops are key to not screw up pressure-velocity coupling. Additionally, `nNonOrthogonalCorrectors` may help in regions where the mesh is not properly orthogonal. Issues in this regard were observed when solving a finer mesh. Previously, on a coarser mesh, `nCorrectors = 1` and `nNonOrthogonalCorrectors = 0` were used just fine.

4.8 fvOptions

```
limitT
{
    type            limitTemperature;
    active          yes;

    selectionMode   all;
    min             200;
    max             2400;
}
```

verticalDamping.C

```
verticalDamping
{
    type            verticalDamping;

    selectionMode   all;
```

```
        origin          (0 0.35 0);  // x y z
        direction       (0 1 0);

        scale  // still unclear what it exactly means
        {
            type        halfCosineRamp;
            start       0;
            duration    .05;
        }

        lambda          [0 0 -1 0 0 0 0] 1000; // Damping coefficient

        timeStart       0;
        duration        5; // time

        writeForceFields true;
    }
```

4.9 controlDict

modify controlDict on the fly

```
/*application       rhoPimpleFoam;*/
startFrom         latestTime;
/*startFrom         startTime;*/
startTime         0;
stopAt            endTime;
endTime           10;
deltaT            5e-5;
writeControl      adjustableRunTime;
writeInterval     0.2;
/*writeControl      timeStep;*/
/*writeInterval     50;*/
purgeWrite        0;
writeFormat       ascii;
/*writeFormat       binary;*/
writePrecision    9;
writeCompression off;
timeFormat        fixed;
timePrecision     6;
runTimeModifiable true;
adjustTimeStep    yes;
maxCo             5.0;
```

4.9.1 functions

```
functions
{
    ...
}

   gradient
   {
       type            grad;
       libs            ("libfieldFunctionObjects.so");
       field           T;

       // Optional (inherited) entries
       writePrecision  8;
       writeToFile     true;
       region          flow;
       writeControl    writeTime;
   }

   wallHeatFlux
   {
       type            wallHeatFlux;
       libs            ("libfieldFunctionObjects.so");
       patches         ("solid_to_flow");
       qr              qr;
       writePrecision  8;
       writeToFile     true;
       region          flow;
       writeControl    writeTime;
   }
```

PS: may have to modify the `constant/boundary` with wall patch.
If coupled domains, using `mappedWall`, it seems to work.

```
   probes
   {
      type                probes;
      functionObjectLibs ("libsampling.so");
      writeControl        timeStep;
      writeInterval       5;
      region              flow;
      fields              (T);
      probeLocations (
          ( 0.0001 0.105 0.0)
          ...
      );
   }
```

```
patchProbes
{
   type                 patchProbes;
   functionObjectLibs ("libsampling.so");

   // Patches to sample (wildcards allowed)
   patchName            wall;

   // Name of the directory for probe data
   //name                 patchProbes;

   writeControl         writeTime;
   //writeControl         timeStep;
   //writeInterval        5;

   //region               flow;
   fields               (wallHeatFlux);

    // Locations to probe. These get snapped onto the nearest point
    // on the selected patches
   probeLocations (
       ( 0.0 0.11 0.0)
   );
}
```

```
line
{
    type                  sets;
    functionObjectLibs   ("libsampling.so");
    enabled               true;
    writeControl          timeStep;
    writeInterval         10;
    region                flow;
    interpolationScheme cellPoint;
    setFormat             raw;
    sets
    (
        line1
        {
            type lineUniform;
            axis distance;
            start ( 1e-5 0.105 0 );
            end   ( 0.05 0.105 0 );
            nPoints 10;
        }
    );
    fields ( p T );
}

volAverage
{
    libs                ("libfieldFunctionObjects.so");
    type                volFieldValue;
    operation           volAverage;
    region              porousMat;
    fields              (Ta tau rho_s[1] rho_s[2]);
    writeFields         false;
    writeControl        timeStep;
    writeInterval       5;
}

volIntegrate
{
    libs                ("libfieldFunctionObjects.so");
    type                volFieldValue;
    operation           volIntegrate;
    region              porousMat;
    fields              (rho_s[1] rho_s[2]);
    writeFields         false;
    writeControl        timeStep;
    writeInterval       5;
}
```

4.10 setFields

Non-uniform IC, specified in `setFieldDict` inside `system/<region>`

```
setFields
setFields -region fluid
```

4.11 parallel

Define system/decomposeParDict

```
numberOfSubdomains  <n>;
method              scotch;
```

to partition the mesh and IC, run:

```
decomposePar <-allRegions>
```

run openFOAM with mpi:

```
mpirun -np <n> <foamSolver> -parallel
```

Combine results with

```
reconstructPar <-allRegions>
```

where additional options can be used, such as:
`-latestTime` :: select the latest time
`-newTimes` :: only reconstruct new times (i.e. that do not exist already)

Depending on the version (7 vs 10), each sub-domain (fluid, wall, porous material etc) will need its own `decomposeParDict` in `system`

To modify parameters in a parallel simulation, may be easier to recombine the latest solution file; modify then re-decompose

5 Solvers

5.1 chtMultiRegionFoam

Put all BCs in "0/" (including the wall as fluid_to_solid and solid_to_fluid)

`splitMeshRegions -overwrite` plus:

`-cellZones` :: the usual, where the domains will be split

`-cellZonesOnly` :: in case one of the domains is disjoint, i.e., no shared nodes. Otherwise, it will be split.

May need to copy a `regionProperties` to `constant`

6 postProcess

postProcessing

`postProcess -func "grad(T)"`

`postProcess -func writeCellVolumes -region porousMat`

`postProcess -list`

6.1 foamToVTK

Useful in case paraFoam misbehaves (aka Segmentation Fault)

If parallel simulation, first run `reconstructPar`.

`foamToVTK`

`-region flow`

Older versions :: `-region <fluid>` then `-region <solid>`

Newer versions :: `-allRegions`

`-nearCellValue`

`-fields '(p p_rgh T U CO2 H2O CO N2)'`

Specify initial time for conversion :: `-time 5.0:`

6.2 solver-dependent

7 Formulation

7.1 Pressure

For buoyant solvers, pressure is computed with

$$\begin{aligned} \rho g h + P' &= P && (7.1) \\ -\nabla(\rho g h) - \nabla P' &= -\nabla P && (7.2) \\ -\cancel{\rho h \nabla g} - \rho g \nabla h - g h \nabla \rho - \nabla P' &= -\nabla P && (7.3) \\ -\rho g - g h \nabla \rho - \nabla P' &= -\nabla P && (7.4) \end{aligned}$$

where $\nabla h = \mathbf{I}$.

link

link

link

7.2 Energy equation

$$C_P(T) = \sum_i c_i T^i = \frac{\partial e}{\partial T} \tag{7.5}$$

$$\kappa \nabla T = \kappa \frac{C_P}{C_P} \nabla T \tag{7.6}$$

$$\kappa \nabla T = \frac{\kappa}{C_P} \frac{\partial e}{\partial T} \nabla T \tag{7.7}$$

$$\kappa \nabla T = \frac{\kappa}{C_P} \nabla e \tag{7.8}$$

$$\kappa \nabla T = \alpha_{\text{eff}} \nabla e \quad \text{where} \quad \alpha_{\text{eff}} = {}^{\kappa}/_{C_p} \tag{7.9}$$

7.3 Species equation

See openFOAM-10 `unityLewisFourier.H`

$$Le = \frac{\alpha}{D} \tag{7.10}$$

$$Le = \frac{\kappa}{\rho C_p} \frac{1}{D} = \frac{\alpha_{\text{eff}}}{\rho D} \tag{7.11}$$

```
virtual tmp<scalarField> DEff
(
    const volScalarField& Yi,
    const label patchi
) const
{
    return
        this->thermo().kappa().boundaryField()[patchi]
       /this->thermo().Cp().boundaryField()[patchi];
}
```

7.4 Transport model

The individual species viscosity is given by

$$\mu_i = A_{s,i} \frac{\sqrt{T}}{1 + \frac{T_{s,i}}{T}} \tag{7.12}$$

with mixture value given by weighted sum of individual viscosities by the respective mass fractions:

$$\mu = \sum_i Y_i \mu_i \tag{7.13}$$

The thermal conductivity is given by the Eucken correlation as

$$\kappa = \mu c_v \left(1.32 + 1.77 \frac{R}{c_v} \right) \tag{7.14}$$

Species diffusivity based on unitary **Lewis** or **Schmidt** number (may depend on openFOAM version)

transport model