OpenFOAM Manual:
Transient dynamics in the outflow of energy from a system in a nonequilibrium stationary state

Data S1

Paweł Jan Żuk

The Data S1 Content

The Data set S1 contains:

- `eRhoFullFOAM` - the `OpenFOAM` solver for solving equations (1);
- `eRhoReducedFOAM` - the `OpenFOAM` solver for solving equations (2);
- `statInitLine` - the `OpenFOAM` utility to set stationary state initial conditions;
- `fullSolution` - the 1D case including travelling front;
- `reducedSolution` - the 1D case including diffusive heat transport only;
- `shortManual.pdf` - the short manual explaining the usage of the code

The content was written, used and tested under `foam-extend 4.1` distribution.

Installation guide

To install use attached codes it is advised to have the `foam-ext-4.1` version of the `OpenFOAM`. This code has not been tested under other `OpenFOAM` distributions. Next, to install all components it is sufficient to enter each `<folder>` (`eRhoFullFOAM`, `eRhoReducedFOAM`, `statInitLine`)

```
cd <folder>
```

and execute

```
wmake
```

command as for every standard `OpenFOAM` utility. The place of the installation can be changed by manipulating the content of

```
<foldr>/Make/files
```

file.

eRhoFullFOAM

The solver `eRhoFullFOAM` is written to solve mass, momentum and

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\mathbf{v}\rho) = 0, \tag{1a}$$

$$\rho \left[\frac{\partial \mathbf{v}}{\partial t} + \nabla \cdot (\mathbf{v}\mathbf{v}) \right] = -\nabla p - \nabla \cdot \mathbf{\Pi}, \tag{1b}$$

$$\frac{3R}{2M} \frac{\partial \rho T}{\partial t} = -\nabla \cdot \left(\frac{3}{2} \frac{R}{M} \rho T \mathbf{v} \right) - p \nabla \cdot \mathbf{v} - \mathbf{\Pi} : \nabla \mathbf{v} + \kappa \nabla \cdot \nabla T, \tag{1c}$$

$$\mathbf{\Pi} = \frac{2}{3} \mu (\nabla \cdot \mathbf{v}) \mathbf{I} - \mu \left(\nabla \mathbf{v} + (\nabla \mathbf{v})^{\mathrm{T}} \right), \tag{1d}$$

$$p = \rho \frac{RT}{M}, \tag{1e}$$

$$u = \frac{3}{2} \frac{RT}{M}. \tag{1f}$$

In the above ρ is density, t is time, $\mathbf{v}$ is velocity, p is pressure, $\mathbf{\Pi}$ is dynamic part of stress tensor, R is gas constant, M is molar mass, $\mathbf{\Pi}$ is dynamic part of stress tensor, κ is thermal conductivity, μ is viscosity, $\mathbf{I}$ denotes unit tensor and u is the internal energy per unit mass.

The equations are solved iteratively in a time loop with to deal with the pressure-velocity coupling problem in Navier-Stokes equation

solve continuity equation (1a);
for i=1 to `nOuterCorrectors` **do**
  solve momentum conservation equation predictor step (1b);
  solve energy conservation equation (1c);
  for j=1 to `nCorrectors` **do**
    solve momentum conservation correctors (1b), which may include number of nonorthogonality corrections `nNonOrthogonalCorrectors`;
    correct pressure for conserved mass (if `closedVolume` is true)
  end for
end for

Two control values `nOuterCorrectors` and `nCorrectors` are specified as

```
PIMPLE
{
    nOuterCorrectors         <integer>;
    nCorrectors              <integer>;
    nNonOrthogonalCorrectors <integer>;
}
```

subdictionary in the `fvSolution` dictionary. For `nOuterCorrIons`= 1 the solver uses the PISO algorithm for iterative solution of pressure-velocity coupling. In case `nOuterCorrIons`> 1 the solver uses PIMPLE algorithm.

The solver needs dictionaries `thermophysicalProperties` and `turbulenceProperties` specified in the `constant` folder. These dictionaries contain physical constants necessary for the solution defined in standard `OpenFOAM` manner. We added only explicit information if the computational domain has closed volume `closedVolume` inside `thermophysicalProperties` dictionary, which can be set either to `true` or `false`.

eRhoReducedFOAM

The diffusive solution follows reduced set of equations

$$\frac{\partial \left(\int \rho \mathrm{d}\mathbf{r}\right)}{\partial t} = 0, \tag{2a}$$

$$\frac{3R}{2M}\frac{\partial \rho T}{\partial t} = \kappa \nabla \cdot \nabla T, \tag{2b}$$

$$p = \rho \frac{RT}{M} \tag{2c}$$

We implemented this solver to be compatible with `eRhoPimpleFOAM` thus the algorithm looks similar

for i=1 to `nOuterCorrectors` **do**
  solve energy conservation equation (2b);
  for j=1 to `nCorrectors` **do**
    correct pressure for conserved mass (if `closedVolume` is true) (2a)
    calculate density profile (2c)
  end for
end for

The settings in case folder `syste/fvSolution` dictionary should be set to `nOuterCorrectors`= 1 and `nCorrectors`= 1.

statInitLine

To introduce initial state given by

$$T_{\mathrm{st}} = T_{\mathrm{eq}} + \nabla T x \tag{3a}$$

$$p_{\mathrm{st}} = p_{\mathrm{eq}} \frac{\frac{\nabla T L}{T_{\mathrm{eq}}}}{\log(1 + \frac{\nabla T L}{T_{\mathrm{eq}}})} \tag{3b}$$

$$v = 0 \tag{3c}$$

$$\rho_{\mathrm{st}} = \rho_{\mathrm{eq}} \frac{\frac{\nabla T L}{T_{\mathrm{eq}}}}{\log(1 + \frac{\nabla T L}{T_{\mathrm{eq}}})} \frac{1}{(1 + \frac{\nabla T L}{T_{\mathrm{eq}}} \frac{x}{L})}. \tag{3d}$$

First we specify equilibrium state by p_{eq} (`p` file), T_{eq} (`T` file) and $\mathbf{v}_{\mathrm{eq}} = 0$ (`U` file) inside `0` folder. The program will calculate ρ_{eq} using values from `constant/thermophysicalProperties` file automatically. Next, we need to specify the positions and temperatures on the boundaries using the `constat/stationaryParameters` file

```
x1  <x position of wall 1>;
x2  <x position of wall 2>;
t1  <T at wall 1>;
t2  <T at wall 2>;
```

The program will read equilibrium properties, calculate initial values corresponding to the prescribed stationary state having the same total mass as equilibrium state and write them out with `_stat` suffix in `0` folder.

fullSolution

The folder contains all necessary files to run the full simulation presented in the main paper. It is sufficient to run the

```
./Allrun
```

inside the folder. It will mesh geometry, create initial conditions, run the simulation for 2×10^{-6} s, which is sufficient for a front to travel twice the geometry length and finally create easy to visualise files in `postProcessing` folder.

reducedSolution

The folder contains all necessary files to run the diffusion only simulation presented in the main paper. It is sufficient to run the

```
./Allrun
```

inside the folder. It will mesh geometry, create initial conditions, run the simulation for 2×10^{-6} s, which is sufficient for a front to travel twice the geometry length and finally create easy to visualise files in `postProcessing` folder.