The porousGasificationFoam
USER MANUAL

Tadeusz Rymarz, Bartosz Tużnik, Paweł Jan Żuk

February 11, 2022

Contents

1 Introductory remarks **3**

2 Description of the solver **3**
2.1 The porousGasificationFoam features . . . 3
2.2 Code structure . . . 3

3 Practical Issues **4**
3.1 OpenFOAM installation . . . 4
3.2 porousGasificationFoam installation . . . 4
3.3 Running cases . . . 5
3.3.1 Preprocessing . . . 5
3.3.2 Run a simulation . . . 6
3.3.3 Postprocessing . . . 6
3.3.4 Essential steps in preparing new simulations . . . 6
3.4 Following sections - test cases . . . 7

4 Wooden ball pyrolysis (macro TGA) **8**
4.1 Physical model . . . 8
4.2 Pre-processing . . . 8
4.2.1 Mesh . . . 8
4.2.2 Dictionaries . . . 9
4.2.3 Running simulation . . . 13

5 Wooden particle pyrolysis (micro TGA) **13**
5.1 Physical model . . . 13
5.2 Pre-processing . . . 13
5.2.1 Mesh . . . 13
5.2.2 Dictionaries . . . 14
5.2.3 Running simulation . . . 15

6 Pressure drop **15**
6.1 Physical model . . . 15
6.2 Running the simulation . . . 15

7 Flat plate **16**
7.1 Physical model . . . 16
7.2 Running the simulation . . . 16

8 Gasifier **17**
8.1 Physical model 17
8.2 Running the simulation 18

1 Introductory remarks

This is a manual to the `porousGasificationFoam` solver and `porousGasificationMedia` libraries. It is developed under GNU General Public License v3 by Paweł Jan Żuk (foam-extend-4.1 version) and Bartosz Tużnik (OpenFOAM 8 version).
The manual is intended to explain how to use the code without explaining ideas and modeling details which can be found in the accompanying scientific paper [1].

The code can be downloaded from:

- `https://github.com/pjzuk/porousGasificationFoam` github repository for the foam-extend-4.1 fork.
- `https://github.com/btuznik/porousGasificationFoam` github repository for the OpenFOAM 8 fork.

Both repositories contain equivalent code at the time of initial release. The specific question and issues should be addressed directly to the responsible authors.

When you use our code or any of its derivatives, please cite the following contribution[1]: *OpenFOAM solver for thermal and chemical conversion in porous media*,
Paweł Jan Żuk, Bartosz Tużnik, Tadeusz Rymarz, Kamil Kwiatkowski, Marek Dudyński, Flavio C. C. Galeazzo, Guenther C. Krieger Filho;
will be uppon acceptance.

2 Description of the solver

2.1 The porousGasificationFoam features

The `porousGasificationFoam` package (i.e. the solver and related `porousGasificationMedia` libraries) is an open-source C++ code written on top of the OpenFOAM package. We developed the `porousGasificationFoam` as a comprehensive CFD simulation tool for the reactive flow through porous media including homogeneous and heterogeneous reactions. The flow within the porous media is modeled by adding Darcy's (or Darcy's and Forchheimer's) resistance term to the Navier-Stokes equations. The mass and enthalpy transfer are calculated for both solid and gas phases, which allows for out of thermal equilibrium calculation. Heat transfer between solid an gas phases includes convection and radiation. We apply the immersed boundary method for the enthalpy conservation equation in the solid phase.

Detailed description of the modeling approach can be found in the main article [1].

2.2 Code structure

The code is available in two versions dedicated for two distinct OpenFOAM distributions:

- foam-extend-4.1,
- OpenFOAM 8.

There are differences in the two implementations resulting from the differences in the main distributions. However, we designed them and implemented them as equivalent. Both are divided into two main parts:

1. `porousGasificationFoam` solver — main code that uses porousGasificationMedia library. All calculations are scheduled here.
2. `porousGasificationMedia` library — consists of four main parts:

(a) `pyrolysisModels` — classes that evaluate state and properties of the porous medium

(b) `thermophysicalModels` — implementation of thermophysical and chemical properties of the porous medium

(c) `porosityModels` — implementation of mechanical properties of the porous medium.

(d) `radiationModels` — heterogeneous radiation model

3 Practical Issues

3.1 OpenFOAM installation

The `porousGasificationFoam` (`PGF` for short) is available in two versions, and it depends on the individual user's preference for which version to choose. The details of the OpenFOAM installation depend on its version and the user's operating system. We recommend [2] for OpenFOAM 8 and [3, 4] for foam-extend-4.1. OpenFOAM 8 is compiled with gcc 9 and foam-extended 4.1 with gcc 7. We recommend the analogous choice of compilers for installing `porousGasificationFoam` with foam version selected by the user.

3.2 porousGasificationFoam installation

The installation guide is prepared under the assumption that OpenFOAM is installed in the standard location: `/opt/OpenFOAM/openfoam8/` (OF8) and `/opt/foam/foam-extend-4.1/` (FE41).

1. Set the OpenFOAM environmental paths by typing (modify the command if your OpenFOAM is installed elsewhere).

   ```
   $ source /opt/OpenFOAM/openfoam8/etc/bashrc (OF8)
   ```

   ```
   $ source /opt/foam/foam-extend-4.1/etc/bashrc (FE41)
   ```

2. Check the environmental settings:
   run any OpenFOAM solver, e.g.

   ```
   $ icoFoam -help
   ```

   type:

   ```
   $ echo $WM_PROJECT_USER_DIR
   ```

3. Optionally change the destination path. The system variable `WM_PROJECT_USER_DIR` stores the default path. To change destination path edit `porousGasificationMediaDirectories` file located in the library installation folder.

4. Set the package environment by typing:

   ```
   $ source porousGasificationMediaDirectories
   ```

5. Run the install script:

   ```
   $ ./Allwmake
   ```

6. Test the installation by running the solver:

   ```
   $ porousGasificationFoam
   ```

3.3 Running cases

The source code is accompanied by tutorial cases. They serve as a reference to show how the solver works and what files are required. A typical procedure would be to copy the tutorial case files and make appropriate changes (in OF8 one can use `foamCloneCase` utility).

3.3.1 Preprocessing

In order to run a case a list of standard actions must be performed i.e., generating mesh (`blockMesh`, `snappyHexMesh`, Salome[5]), setting the initial and boundary conditions for `U`, `p`, `T`, ... fields, specifying control dictionaries in `system` and `constant` directories. It is assumed that the user has basic knowledge about the OpenFoam software. The case structure of the `PGF` is the same as in OpenFoam simulations:

- `0` directory contains all necessary initial and boundary conditions for the fields (`p`, `U`, `T`, `Ts`, `Yi`, `Ys` ...). Fields specific to `PGF` are:
  - `Ys` [-] : scalar field of solid species fraction where s stands for specific specie e.g. `Ychar`, `Ywood`; fields not explicitly written in `0` will be created from required `YsDefault` file
  - `Ts` [K] : solid temperature
  - `Df` [1/m2] : Darcy resistance term
  - `anisotropyK` [-] : (when not specified it is a diagonal unit tensor) anisotropy factors in heat conduction for solid
- `const` directory includes files specifying physical properties of the model. Files specific to `PGF` are:
  - `pyrolysisProperties` - selection of `heterogeneousPyrolysisModel` (only `volPyrolysis` available) and heat transfer subintegration switch `subintegrateHeatTransfer`.
  - `solidThermophysicalProperties` - contains list of solid species and their thermal properties
  - `radiationProperties` - specify choice of ratiaditve heat transfer model and its parameters.
  - `chemistryProperties` - contains gas phase chemistry model specification and heterogeneous chemistry model specification. The heterogeneous reactions are defined inside subdictionary `solidReactions`
  - `heatTransferProperties` - specifies choice of convective heat transfer and its parameters
  - `porosityProperties` - if present, specifies Forchheimer's coefficient value; if not present, the value is set to zero - no quadratic resistance term
  - `specieTransferProperties` - choice of mass transfer and its parameters; necessary, if diffusion limited reactions are used (`diffusionLimitedReactions true;` inside `chemistryProperties`).
- `system` directory contains simulation control parameters `controlDict`, discretization schemes `fvSchemes`, solution control `fvSolution` and others e.g., multiprocessor domain decomposition dictionary `decomposeParDict`.

The above description is best read during inspection of tutorial cases, which include additional comments.

3.3.2 Run a simulation

To run the case simply type (inside case folder):

```
$ porousGasificationFoam
```

To run in parallel with N cores first decompose the domain according to the instructions in `decomposeParDict` file by running:

```
$ decomposePar
```

Next, execute the parallel run by typing:

```
$ mpirun -np N porousGasificationFoam -parallel
```

3.3.3 Postprocessing

The `PGF` comes with a utility `totalMassPorousGasificationFoam` for calculating total solid mass

$$m = \int_{domain} \langle \rho \rangle^s \, (1 - \epsilon) \, \mathrm{d}\mathbf{x}, \tag{1}$$

where $\langle \rho \rangle^s$ is the density of solid matrix inside porous media, and ϵ is porosity or void fraction. The calculation is performed for each stored time step. It is installed alongside `PGF` with the install script. To use total mass one should simply type:

```
$ totalMassPorousGasificationFoam
```

inside case directory. As a result, a file `totalMass.txt` is created containing results in two columns: time step in first and m in second.

Other post-processing can be done in a standard OpenFoam manner, e.g., using paraView and post-processing utilities. OpenFoam also provides utilities that perform results processing alongside running simulation. The `runTime` processing is governed by appropriate dictionaries and entries within `controlDict`. Information on how to process the results can be found in the OpenFoam user guides.

3.3.4 Essential steps in preparing new simulations

Here are important steps related to creating a new simulation. These are not all but some issues related to `PGF` simulations.

- **Biomass distribution**: Scalar `porosityF` field determines the biomass distribution across the domain. We suggest using OpenFoam tools:

  ```
  $ setSet
  ```

  ```
  $ setFields
  ```

  in order to create a zone with cells that contain biomass and set appropriate `porosityF` value within that zone and `porosityF` equal to one everywhere else (`setFields` alone without `setSet` can be used for that purpose as well). See the OpenFoam documentation to learn how to use `setSet`. To initialize `porosityF` with `setFields` specify `setFieldsDict` within `system` directory. Other fields such as `T, Ts` and so on can also be initialized with this approach. To create more sophisticated biomass distribution, such as a detailed fixed bed gasification stack, we suggest using tools such as Salome or Blender to create STL models of biomass distribution and then use this STL file with `setSet` and `setFields`. Additionally, there is a `setPorosity` tool prepared with the installation package. It is more involved and requires compilation after each alternation of setup. The tool has a

user-editable part in file `media.H` within the tool folder. We suggest this tool only for more advanced users.

- `thermo.compressibleGas` file: often appropriate JANAF properties for particular gas species do not exist, e.g., when dealing with pyrolytic gases created during pyrolysis pseudo-reactions. In such a case, one should choose from `thermoData` inside `$FOAM_ETC` a specie mimicking as closely as possible pseudo specie.
- `solidThermophysicalProperties` file: in this file it is crucial to remember that solid properties entries are related to pure solid (true density; `porosityF` equal to zero; bulk modeling approach can be also used thou - see gasifier test case). This is true for all parameters, density, heat conductivity, and so on. In literature, bulk properties are usually described, so the user should be careful!
- `radiationPropreties` file: the radiation properties (for the proposed volumetric model) are probably the hardest ones to find in literature. The wall-solid species radiation is most crucial when dealing with the simulation of pyrolysis/gasification/combustion of small pellets or wood fragments inside a tube furnace. We suggest altering parameters from `radiationProperties` file to simulate temperature rise correctly from experiments and then using these parameters in other simulations if a radiation mechanism is needed.
- `timeStep`: typically, gas reactions are orders of magnitude faster than heterogeneous solid reactions. Thus, existing of the former implies a more restrictive time step limitation. One should take it into account when simulating slow (and thus requiring a long time to finish) processes such as gasification.

3.4 Following sections - test cases

The previous sections outlined general remarks about the solver installation and usage. The following paragraphs describe the test cases included alongside the source code. The `tutorials` directory contains the test cases and the experimental data used for validation, listed below:

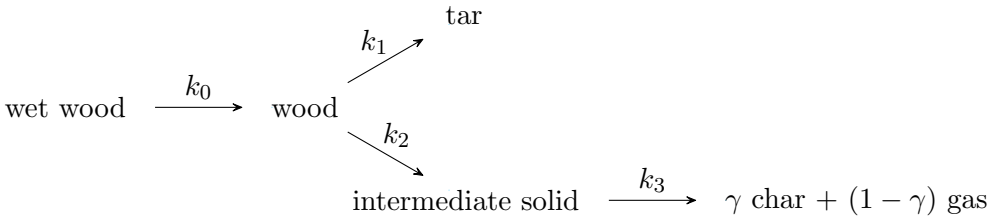
- `macroTGA_688K`
- `macroTGA_688K_fine`
- `macroTGA_879K`
- `macroTGA_879K_fine`
- `macroTGA_experimentalData`
- `microTGA`
- `biomassPressureDrop`
- `flatPlate`
- `gasifier`

The numerical results of the cases can be found in the main article [1].

4 Wooden ball pyrolysis (macro TGA)

4.1 Physical model

The macro TGA test case is based on the paper *Experimental and theoretical investigation of heat and mass transfer processes during wood pyrolysis* [6] in which an inch diameter wooden ball is pyrolyzed in a tube furnace. We present two simulations for two different furnace temperatures, i.e., 688K and 879K, and for two mesh resolutions, coarse and fine. In the paper [6], four kinetic models are described. One of the models is used in the test case:



Wood is first dried (optional). Dry wood decomposes in two parallel competing reactions. The tar path is endothermic. The intermediate solid path has zero heat of reaction. Finally, intermediate solid decomposes into char and gas in an exothermic reaction. Table 1 contains kinetic coefficients (the user should remember that `PGF solidReactions` take activation temperature $T_a = \frac{E_a}{R}$ as an input, not the activation energy E_a; R is the universal gas constant in SI units).

Reaction	0	1	2	3
A_i s^{-1}	13.6	$2 \cdot 10^{10}$	$2.51 \cdot 10^{7}$	$1.38 \cdot 10^{10}$
E_i kJ/mol	$2.41 \cdot 10^{4}$	148	117	161
Δh_i kJ/kg	2260	110	0	-210

Table 1: Arrhenius coefficients in reactions *1-3* are taken from [6]

According to [6] the wood contains no moisture at the beginning of the experiment. Therefore, the k_0 reaction is not included in the macro TGA test case, and the initial `Ywood` field is equal to 1.

4.2 Pre-processing

4.2.1 Mesh

To simplify the simulation the mesh is pure hexagonal, created with standard OpenFoam utilities `blockMesh`, `setSet` and `refineHexMesh`. Initial porosity distribution is specified with `setFields`. The `macroTGA` cases are provided with the `buildCase???.sh` bash script (??? stands for furnace temperature) which performs all the necessary preprocessing steps. The script contains the following instructions:

```
#!/bin/bash

foamClearPolyMesh
rm -r 0
cp -r save688 0
blockMesh

#first refinement level
```

```
setSet -batch setSet.c0
refineHexMesh -overwrite c0

#second refinement level
setSet -batch setSet.c1
refineHexMesh -overwrite c1

setFields
renumberMesh -overwrite
checkMesh
```

To build the case, simply run the script within the case directory (688K case example):

```
$ ./buildCase688.sh
```

4.2.2 Dictionaries

As said in section 3 in addition to standard OpenFOAM dictionaries the case must be provided with dictionaries specific to the `PGF` within the `constant` directory. The test case dictionaries contain C style comments to explain certain entries (three dotes "..." means that for space-saving and clarity only the part of the file is shown).

Due to the differences in implementation in `(FE41)` and `(OF8)`, some dictionaries are not exactly the same.

1. `chemistryProperites`

   (FE41)

```
// gas phase chemistry model
psiChemistryModel   ODEChemistryModel<gasThermoPhysics>;

// solid phase chemistry model
solidChemistryModel  ODESolidHeterogeneousChemistryModel<
    constSolidThermoPhysics,gasThermoPhysics>;

gasThermoModel      gasThermoPhysics;

chemistry           on;

// gas chemistry solver selection
chemistrySolver     ode;

// gas chemistry solver selection
// only solidODE available
solidChemistrySolver    solidOde;

initialChemicalTimeStep 1e-07;

turbulentReaction   off;

// heterogeneous reactions switches
// details can be found in the main article
solidReactionEnergyFromEnthalpy false;
stoichiometricReactions false;
showRelativeReactionRates false;
solidChemistryTimeStepControl true;

// gas chemistry solver settings
odeCoeffs
{
    ODESolver       KRR4;
```

```
    eps             0.1;
    scale           1;
}

// solid chemistry solver settings
solidOdeCoeffs
{
    ODESolver   KRR4;
    eps         0.01;
    scale       1;
}

species
(
    targas gas N2
);

solidReactions
(
    // 1
    irreversibleSolidArrheniusHeterogeneousReaction
    wood = targas
    (2.0e10 1.78e4 300 1.10e5 1)

    // 2
    irreversibleSolidArrheniusHeterogeneousReaction
    wood = char1
    (2.51e7 1.4e4 300 0e5 1)

    // 3
    irreversibleSolidArrheniusHeterogeneousReaction
    char1 = 0.65 char + 0.35 gas
    (1.38e10 1.9e4 300 -2.1e5 1)
);
```

(OF8) similar, but slightly different

```
chemistry           on;

chemistryType
{
    solver            ode;
}

solidChemistryType
{
    solver              solidOde;
    method              ODESolidHeterogeneousChemistryModel;
    solidThermoType     const<constRad<constThermo<constRho>>>;
}

chemistrySolver     ode;

solidChemistrySolver  solidOde;

initialChemicalTimeStep 1e-7;

turbulentReaction   off;

solidReactionEnergyFromEnthalpy false;
stoichiometricReactions false;
showRelativeReactionRates false;
solidChemistryTimeStepControl true;
```

```
odeCoeffs
{
    solver      seulex;
}

solidOdeCoeffs
{
    solver      seulex;
}

species
(
    targas gas N2
);

solidReactions
(
    // 1
    irreversibleSolidArrheniusHeterogeneousReaction
    wood = targas
    (2.0e10 1.78e4 300 1.10e5 1)

    // 2
    irreversibleSolidArrheniusHeterogeneousReaction
    wood = char1
    (2.51e7 1.4e4 300 0e5 1)

    // 3
    irreversibleSolidArrheniusHeterogeneousReaction
    char1 = 0.65 char + 0.35 gas
    (1.38e10 1.9e4 300 -2.1e5 1)
);
```

2. solidThermophysicalProperties

```
/* solid thermodynamic model selection; */
thermoType solidMixtureThermo<constHeterogeneous>;

/* all solid species must be specified here*/
solidComponents
(
    wood char1 char
);

/* for each solid specie defined in solidComponents
the subdictionary named with specie name + Coeffs
sufix must be declared*/
woodCoeffs
{
    transport
    {
        // thermal conductivity W/m/K
        K       0.341;
    }
    thermodynamics
    {
        // heat capacity J/kg/K
        Cp      1800;
        // enthalpy of formation
        Hf      -1.04e6;
    }
    density
```

```
    {
        // true density kg/m3
        rho      1050;
    }
};

...
```

3. radiationProperties

```
//- on/off radiation switch
radiation       on;

/* heterogeneous radiation model selection:
heterogeneousP1 or meanTemp*/
heterogeneousRadiationModel  heterogeneousP1;

//- Number of flow iterations per radiation iteration
solverFreq 1;

heterogeneousAbsorptionEmissionModel
    heterogeneousConstantAbsorptionEmission;

// These values are chosen to support
// heating rate comparable with experimental borderL*borderAs
// mesh independence borderL

heterogeneousConstantAbsorptionEmissionCoeffs
{
    // gas absorptivity
    a               a        [ 0 -1  0 0 0 0 0 ] 0;
    // solid absorptivity
    as              as       [ 0 -1  0 0 0 0 0 ] 0;
    // border solid absorptivity
    borderAs        borderAs [ 0 -1  0 0 0 0 0 ] 180;
    E               E        [ 1 -1 -3 0 0 0 0 ] 0;
    //penetration depth borderL
    borderL         borderL  [ 0  1  0 0 0 0 0 ] 1.5e-3;
}

scatterModel    constantScatter;

constantScatterCoeffs
{
    sigma           sigma [ 0 -1 0 0 0 0 0 ] 1;
    C               C     [ 0  0 0 0 0 0 0 ] 1;
}
```

4. pyrolysisProperties

```
active          true;

//pyrolysis model selection - only volPyrolysis available
heterogeneousPyrolysisModel  volPyrolysis;

pyrolysisCoeffs
{
    // subintegration on/off switch
    subintegrateHeatTransfer true;
}

//prints pyrolysis details in the log/stdout
infoOutput      true;
```

Standard OpenFOAM files such as: `reactions`, `thermo.compressibleGas`, and `thermophysicalProperties` also differs slightly in FE41 and OF8, just as they differ when used with standard OpenFOAM solver `reactingFoam` for the two different distributions.

4.2.3 Running simulation

To run the test case simply type:

```
$ porousGasificationFoam
```

To run in parallel, the number of cores must be specified within `decomposeParDict` located in `system` directory. For example:

```
numberOfSubdomains 4;

method             simple;

simpleCoeffs
{
    n                (2 2 1);
    delta            0.001;
}
```

The command

```
$ decomposePar
```

decomposes the domain into (in our case 4) processors and

```
$ mpirun -np 4 porousGasificationFoam -parallel
```

runs the case in parallel. After finishing the run, the user may want to reconstruct the case back to a single processor. The command

```
$ reconstructPar
```

is used for that purpose.

`ParaView` with `-builtin` flag can visualise decomposed data (`totalMassPorousGasificationFoam` can be used only with reconstructed case).

5 Wooden particle pyrolysis (micro TGA)

5.1 Physical model

The second case presented in the main contributions is drying and pyrolysis of the small wooden particle. The main difference is that the particle is small enough to be considered thermally thin, i.e., the temperature gradients within the sample are very small. Thus the pyrolysis takes place in a kinetic regime and is determined by the furnace wall heating rate. Another difference is that the sample contains 8% of moisture at the beginning of the simulation, so the drying stage is also included in kinetic model. It is the k_0 reaction according to the previously presented kinetic scheme.

5.2 Pre-processing

5.2.1 Mesh

The geometry is even simpler than in the macro TGA test. The micro furnace is modeled as a box with pure-hexagonal perfect mesh. Therefore only `blockMesh` and `setFields` utilities are needed. As previously, `buildCase.sh` script includes necessary commands:

```
#!/bin/bash

foamClearPolyMesh
rm -r 0
cp -r save  0
blockMesh
setFields
renumberMesh -overwrite
checkMesh
```

5.2.2 Dictionaries

The wall temperature is rising with constant speed 10 K/min, thus the `timeVaryingUniformFixedValue` BC is used for that purpose in `T` file:

```
    inlet
    {
        type            timeVaryingUniformFixedValue;
        fileName        "$FOAM_CASE/heatingRate.dat";
        outOfBounds     clamp;
        value           uniform 300;
    }
    wall
    {
        type            timeVaryingUniformFixedValue;
        fileName        "$FOAM_CASE/heatingRate.dat";
        outOfBounds     clamp;
        value           uniform 300;
    }
```

The case directory includes the `heatingRate.dat` with the start and the end points between which the temperature rises linearly:

```
(
 //(time temperature)
 (0     300)
 (4000  966)
)
```

The k_0 drying reaction is added to the `chemistryProperites` in the `constant` directory

```
...

solidReactions
(
    // 0
    irreversibleSolidArrheniusHeterogeneousReaction
    hum = H2O
    (1.36e1 2.9e3 301  2.256e6 1)

    // 1
    irreversibleSolidArrheniusHeterogeneousReaction
    wood = targas
    (2.0e10 1.78e4 300 1.10e5 1)

    // 1
    irreversibleSolidArrheniusHeterogeneousReaction
    wood = char1
    (2.51e7 1.4e4 300 0e5 1)

    // 2
```

```
    irreversibleSolidArrheniusHeterogeneousReaction
    char1 = 0.65 char + 0.35 gas
    (1.38e10 1.9e4 300 -2.1e5 1)
);
```

All others settings within `constant` directory are the same as in the macro TGA case.

5.2.3 Running simulation

To run the case type:

```
$ porousGasificationFoam
```

and to run the case in parallel, first decompose the domain:

```
$ decomposePar
```

and than run the case in parallel:

```
$ mpirun -np 4 porousGasificationFoam -parallel
```

6 Pressure drop

6.1 Physical model

This test case examines the pressure drop resulting from a flow through a porous material. The simulation follows the experimental setup described in [7]. The porous cubic shape material is placed in the apparatus, and the constant external pressure difference is applied. The case does not include any reactions or thermal fluid-solid interactions. The main role is to test the porous material flow resistance implementation. The results and the comparison with the experimental data from [7] can be found in the main article.

6.2 Running the simulation

The `biomassPressureDrop` directory contains five sub-cases `p04, p08, p10, p12, p16` which relate to five reference pressures from [7] i.e. 0.04, 0.08, 0.10, 0.12, 0.16 MPa. Simple hexahedral mesh is already present in `constant/polyMesh` directory for each of the cases. In the experiment, the pressure is controlled at the inlet, and both the pressure drop and the airflow are measured downstream. Because the inlet pressure rises as the simulation converges to a steady-state, the simple `fixedValue` for velocity BC at the inlet is not correct. Rising pressure, and thus, rising density of the air with a constant velocity at the inlet, results in increasing mass stream entering the domain. The `flowRateInletVelocity` BC was used instead, which specifies the mass inflow directly (for compressible solvers; for incompressible ones it specifies volumetric flux instead in the `foam-extend-4.1` implementation; `openfoam-8` uses explicit notation for volumetric and mass fluxes, but the idea is the same). When the pressure at the inlet rises, the velocity drops down to keep the mass influx constant. We assume all other pressure losses are present on the downstream part. The pressure BC at the outlet is set to $p_{atm} + (p_{inlet\_exp} - \Delta p_{exp})$ for each case. The `biomassPressureDrop` directory includes also `README` and several scripts to facilitate execution of the simulation:

- `Allclean.sh`
- `Alllog_pmin_pmax.sh`
- `Allrun_Df2.56e+10_f3.21e+6.sh`
- `Allrun_Df2.5e+11_f0.s`

README from the case directory explains what each of the scripts do:

```
Allclean.sh script cleans the directories of all five p?? subcases
(removes time folders, postProcessing and log file)

Allrun_Df???_f???.sh copies Df_??? and porosityProperties_??? to 5 subcases
and runs all of them in the backround (porousGasificationFoam > log&)
with those Df_??? and f_??? set for Darcy and Forchheimer coeffs.

Alllog.sh tails all 5 log files and the user can easily see the p_min and p_max.
Pressure drop equals almost exactly to p_max-p_min; the error can be neglected.

Example exectuion:
  ./Allclean.sh
  ./Allrun_Df2.56e+10_f3.21e+6.sh
  ./Alllog_pmin
```

The new feature present in this simulation is the Forchheimer quadratic term F. The RHS of the momentum equation (full equation can be found in the main article) reads:

$$-\mu_{eff}\mathbb{D}\cdot\langle u\rangle - \mathbb{F}\cdot\langle u\rangle \tag{2}$$

where:

$$\mathbb{F} = F_c\rho^f|\langle\mathbf{u}\rangle|\frac{\sqrt{3}}{|\mathbb{D}|}\mathbb{D} \tag{3}$$

The `porosityProperties` dictionary in `constant` directory allows for setting the value of F_c using the `forchheimerCoeff` entry:

```
forchheimerCoeff 3.21e6;
```

The F_c is the same for all cells in contrast to D_f, which is a field and can have a different value in each cell. The main article shows that the Forchheimer coefficient has a significant impact on the correctness of the results for fast flows. The [7] paper reports on pressures that induce flow velocities of the order 1 m/s through the porous media. Therefore, the quadratic part of the resistance is also used in this simulation. In the previous simulation, the velocities within porous material were relatively low, and only the D_f Darcy term was needed.

7 Flat plate

7.1 Physical model

This test case examines momentum and heat transfer between freestream fluid flow and flat plates made from a porous material. It does not include chemical reactions. Here we examine the skin friction coefficient C_{fx} and the Nusselt number Nu, and how the Immersed Boundary like modeling utilized in `PGF` deals with external transport phenomena.

7.2 Running the simulation

The case does not introduce any new dictionaries that need to be explained explicitly here. The validation results can be found in the main article. The main directory contains several scripts to facilitate the process of running the simulation. `Allclean.sh` cleans the case to the initial state. There are three sub-cases to be run: one for cold plate (skin friction) and two for hot plate (Nusselt number). As with the pressure drop, each version of the case is run with appropriate script:

- `Allrun_cold.sh`

- `Allrun_hot_immediate.sh`
- `Allrun_hot_stat.sh`

Scripts first copy the necessary files, which are present in the `initialData` folder, and next, they execute the computations. For example `Allrun_cold.sh` does the following steps:

```
cp -r 0.orig 0
cp initialData/pyrolysisProperties_cold constant/pyrolysisProperties
cp initialData/controlDict_cold system/controlDict
cp initialData/setFieldsDict_cold system/setFieldsDict

blockMesh -dict system/blockMeshDict
setSet -batch makeFaceSet.setSet
createPatch -overwrite
setFields

decomposePar

# Run
# porousGasificationFoam
mpirun -np 4 porousGasificationFoam -parallel > log&
```

The `Allrun_hot_immediate.sh` script is used to calculate fields necessary for the Nusselt number calculation at the beginning of the process when the porous media has an approximately uniform temperature. The `Allrun_hot_immediate.sh` script is used to calculate fields necessary for the Nusselt number calculation when the porous media is close to stationary steady state. It is initiated from the precalculated temperature fields that were mapped from the coarse mesh as it takes approximately 10000 s calculated real-time to reach the stationary state.

8 Gasifier

8.1 Physical model

The last case we present is an axisymmetric, wedge model of a 3d gasifier adapted from [8]. We model the porous bed, which is made out of packed wooden spheres, at a megascopic scale, i.e., the whole bed is treated as one uniform porous material obeying Darcy's law similarly to sand or rock deposits. Here we model a different scale of porosity than in other `PGF` cases, where the true, not bulk properties of the porous materials have to be specified. For example the density is set to bulk density of wood:

```
density
{
      rho   663;
}
```

In the modelling of an exothermic gasification process an important feature is the diffusive limitation of the reaction rate. It is introduced to the `PGF` solver by first, including a new (optional) entry in the `chemistryProperties` dictionary:

```
density
{
      diffusionLimitedReactions true;
}
```

If absent, or set to `false`, it will be ignored. When present and set to `true`, the solver will require additional dictionary `specieTransferProperties` inside the `constant` folder. This dictionary

is similar to `heatTransferProperties` dictionary and contains model name with its properties

```
specieTransferModel constST;

Parameters
{
    h         0.01;
    SAV      100;
}
```

8.2 Running the simulation

The scripts `Allclean.sh` and `Allrun.sh` contain and execute all necessary steps to run the simulation. Calculating the complete gasification may take several hours depending on available computational resources.

References

[1] P. J. Żuk, B. Tużnik, T. Rymarz, K. Kwiatkowski, M. Dudyński, F. C. C. Galeazzo, G. K. Krieger Filho, Openfoam solver for thermal and chemical conversion in porous media, Computer Physics Communications (2021).

[2] Openfoam 8 official web page, `https://openfoam.org/version/8/`.

[3] foam-extend-4.1 release web page, `https://sourceforge.net/projects/foam-extend/`.

[4] foam-extend-4.1 installation guide, `https://openfoamwiki.net/index.php/Installation/Linux/foam-e`

[5] foam-extend-4.1 installation guide, `https://www.salome-platform.org/`.

[6] W. C. Park, A. Atreya, H. R. Baum, Experimental and theoretical investigation of heat and mass transfer processes, Combustion and Flame 157 (2010) 481. doi:10.1016/j.combustflame.2009.10.006.

[7] G. Wałowski, Assessment of gas permeability coefficient of porous materials, Journal of Sustainable Mining 16 (2) (2017) 55–65. doi:https://doi.org/10.1016/j.jsm.2017.08.001.
URL `https://www.sciencedirect.com/science/article/pii/S2300396017300538`

[8] M. Fatehi, M. Kaviany, Adiabatic reverse combustion in a packed bed, Combustion and Flame 99 (1994) 1–17.