The porousGasificationFoam
USER MANUAL

Tadeusz Rymarz, Bartosz Tużnik, Paweł Jan Żuk

March 7, 2021

Contents

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 it's 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 resistance term to the Navier-Stokes equations. The mass and enthalpy transfer are calculated for both solid and gas phases, which allows for non-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, which result from the differences in the main distribution, however we designed them and implemented 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 which version to choose. The details of the OpenFOAM installation depend on it's version and on 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 analogous choice of compilers for installing `porousGasificationFoam` with foam version selected by the user.

3.2 porousGasificationFoam installation

The installation guide is prepared under assumption that OpenFOAM is installed in 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 with 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 the 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 it's 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 it's parameters
- `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. The detailed tutorial cases description is provided in section 4.

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 standard OpenFoam manner e.g., with the use of paraView and post processing utilities. OpenFoam also provides utilities that perform results processing alongside running simulation. The `runTime` processing is governed but appropropriate dictionaries and entries within `controlDict`. Information on how to process the results can be found in the OpenFoam userguides.

3.3.4 Essential steps in preparing new simulations

Here are important steps related to creating new simulation. These are of course 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 are related to pure solid (true density; `porosityF` equal to zero). This is true for all parameters, density, heat conductivity and so on. In literature usually bulk properties are described, so the user should be careful!
- `radiationPropreties` file: the radiation properties (for proposed volumetric model) are probably the most hard one to find in literature. The wall-solid species radiation is most crucial when dealing with simulation of pyrolysis / gasification / combustion of small pellets or wood fragments inside tube furnace. We suggest altering parameters from `radiationProperties` file to simulate correctly temperature rise from experiments and then using these parameters in other simulations if radiation mechanism is needed.
- `timeStep`: typically gas reactions are orders of magnitude faster than heterogeneous solid reactions thus existing of the former implies 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.

4 Test case - wooden ball pyrolysis (macro TGA)

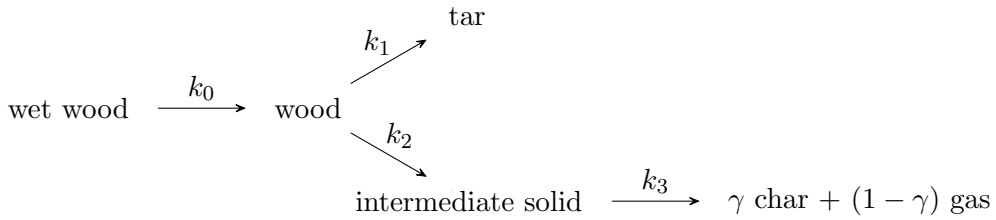
The previous sections outlined general remarks about the solver instillation and usage. The following paragraphs provide description of the test cases included alongside the source code. The `tutorials` directory contains 5 separate directory cases and the experimental data folder used for validation:

- `macroTGA_688K`
- `macroTGA_688K_fine`
- `macroTGA_879K`
- `macroTGA_879K_fine`
- `macroTGA_experimentalData`
- `microTGA`

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

4.1 Physical and mathematical 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 pyrolysied in a tube furnace. We present two simulations for two different furnace temperatures i.e. 688K and 879K and for two mash 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. Tar path is endothermic. Intermediate solid path has zero heat of reaction. Finally intermediate solid decomposes into char and gas in 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 coefficeints. 0 this work, 1-3 [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 case is provided with the `buildCase???.sh` bash script (??? stands for furnace temperature) which performs all the necessary preprocesing 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 8;

method            simple;

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

The command

```
$ decomposePar
```

decomposes the domain into (in our case 8) processors and

```
$ mpirun -np 8 porousGasificationFoam -parallel
```

runs the case in parallel. After finishing the run user may want to reconstruct the case back to 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 Test case - wooden particle pyrolysis (micro TGA)

5.1 Physical and mathematical 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 as 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` scrpit 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 8 porousGasificationFoam -parallel
```

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.