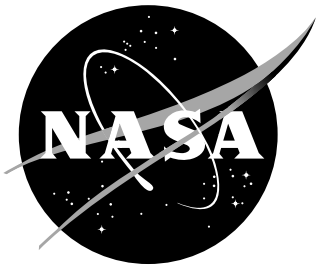

USER GUIDE

PATO

Porous material Analysis Toolbox based on OpenFOAM

Version 3.0

Jeremie B. E. Meurisse *
Jean Lachaud §
Nagi N. Mansour ¶

* *Analytical Mechanics Associates, Inc.*
at NASA Ames Research Center, Moffett Field, CA 94035, USA

§ *University of Bordeaux, Institute of Mechanical Engineering (I2M)*
33400 Talence, France

¶ *Computational Physics, LLC.*
38 S. B Street, San Mateo, CA 94401, USA

June 2022

Contents

Nomenclature

$\bar{h}$	Average solid enthalpy, $\mathrm{J} \cdot \mathrm{kg}^{-1}$
$\boldsymbol{g}$	Gravity vector, $\mathrm{m} \cdot \mathrm{s}^{-2}$
$\boldsymbol{n}$	Normal of the face
$\boldsymbol{v_g}$	Gaseous velocity, $\mathrm{m} \cdot \mathrm{s}^{-1}$
$\mathcal{A}_{ij}$	Pre-exponential factor of pyrolysis reaction j within phase i, SI
$\mathcal{E}_{ij}$	Activation energy factor of pyrolysis reaction j within phase i, $\mathrm{J} \cdot \mathrm{kg}^{-1}$
$\mathcal{F}_{ij}$	Mass fraction of pyrolysis reaction j within phase i
$\dot{E}_\Omega$	Energy flux due to the heterogeneous production rate, $\mathrm{W} \cdot \mathrm{m}^{-3}$
$\dot{E}_p$	Pyrolysis energy flux, $\mathrm{W} \cdot \mathrm{m}^{-3}$
$\dot{E}_D$	Energy flux transported by diffusion of the species, $\mathrm{W} \cdot \mathrm{m}^{-3}$
$\dot{m}$	Mass flux, $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$
$\dot{m}_{ca}$	Char ablation rate, $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$
$\dot{m}_{pg}$	Pyrolysis gas production rate, $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$
St_H	Stanton number for heat transfer
St_M	Stanton number for mass transfer
$\underline{\underline{\mathbf{C_e}}}$	Fourth order elastic constitutive tensor, Pa
$\underline{\underline{\mathbf{K_s}}}$	Solid permeability, m^2
$\underline{\underline{\mathbf{k_s}}}$	Solid thermal conductivity, $\mathrm{W} \cdot \mathrm{m}^{-1} \cdot \mathrm{K}^{-1}$
$\underline{\underline{\mathbf{X_s}}}$	Inverse of permeabilities, $\mathrm{s} \cdot \mathrm{kg}^{-1} \cdot \mathrm{m}^{-1}$
$\mathbf{v}_{ca}$	Char ablation velocity, $\mathrm{m} \cdot \mathrm{s}^{-1}$
A_i	Pre-exponential factor of pyrolysis reaction i, SI
A_k	Element k
B'	Dimensionless mass blowing rate
C'_H	Corrected heat transfer coefficient, $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$
$C(gr)$	Solid carbon graphite
C_H	Heat transfer coefficient $= \rho_e u_e St_H$, $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$
C_M	Mass transfer coefficient $= \rho_e u_e St_M$, $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$
$c_{p,s}$	Solid heat capacity, $\mathrm{J} \cdot \mathrm{kg}^{-1} \cdot \mathrm{K}^{-1}$
D	Displacement field, m
$D_{m,i}$	Diffusion coefficient of species i, $\mathrm{m}^2 \cdot \mathrm{s}^{-1}$
e	Internal energy, $\mathrm{J} \cdot \mathrm{kg}^{-1}$
E_g	Total gaseous energy, $\mathrm{J} \cdot \mathrm{m}^{-3}$
E_i	Activation temperature of pyrolysis reaction i, K
F_p	Applied pressure, $\mathrm{N} \cdot \mathrm{m}^{-2}$
F_t	Traction force, N
g_f	Projection of the gravity vector on the face, $\mathrm{m} \cdot \mathrm{s}^{-2}$
H	Volumic fluid/solid exchange coefficient, $\mathrm{W} \cdot \mathrm{K}^{-1} \cdot \mathrm{m}^{-3}$
h	Enthalpy, $\mathrm{J} \cdot \mathrm{kg}^{-1}$
h_c	Char solid enthalpy, $\mathrm{J} \cdot \mathrm{kg}^{-1}$
h_g	Gaseous enthalpy, $\mathrm{J} \cdot \mathrm{kg}^{-1}$
h_v	Virgin solid enthalpy, $\mathrm{J} \cdot \mathrm{kg}^{-1}$
h_{conv}	Convective heat transfer coefficient, $\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{K}^{-1}$

h_i	Gaseous enthalpy of species i, J $\cdot$ kg^{-1}
$h_{p,ij}$	Reference enthalpy of pyrolysis reaction j within phase i, J $\cdot$ kg^{-1}
$h_{S,s}$	Sensible solid enthalpy, J $\cdot$ kg^{-1}
h_{th}	Heat transfer coefficient, W $\cdot$ m$^{-2}\cdot$ K^{-1}
k	Bulk modulus, Pa
K_i	Equilibrium constant of species i
L	Sum of the least square difference
M_g	Gaseous molar mass, kg $\cdot$ kmole^{-1}
m_{ij}	Arrhenius law factor of pyrolysis reaction j within phase i, J $\cdot$ kg^{-1}
N_c	Number of Fourier coefficients
N_e	Number of elements
N_p	Number of phases
N_s	Number of gaseous species
N_{sp}	Number of pyrolysis reactions
p_g	Gaseous pressure, Pa
Q	Thermal power, W
q	Heat flux, W $\cdot$ m^{-2}
q_r	Radiative heat flux, W $\cdot$ m^{-3}
R	Universal gas constant, J $\cdot$ K^{-1} $\cdot$ kmole^{-1}
r	Solid density ratio
r_T	Fiber radius, m
$r_{T,0}$	Initial fiber radius, m
S_D	Courant number factor, s^{-1}
S_d	Site density, kmole $\cdot$ m^{-2}
S_i	Species i
S_s	Specific surface, m^{-1}
T	Temperature, K
T_∞	Background temperature, K
T_a	Porous material temperature, K
T_g	Gaseous temperature, K
y_i	Mass fraction of species i
z_{ijk}	Mass fraction of element k of pyrolysis reaction j within phase i
z_k	Mass fraction of element k
x_i	Mole fraction of species i
x_k	Mole fraction of element k

Greek

$\widetilde{\pi}_{tot}$	Normalized total pyrolysis production rate, s^{-1}
$\underline{\underline{\gamma_{\mathbf{g}}}}$	Convective part of the gaseous mass flux, s
$\underline{\underline{\gamma_{\mathbf{hg}}}}$	Convective part of the heat transfer, J $\cdot$ s
$\overline{\alpha}$	Absorptivity
α	Thermal diffusivity, kg $\cdot$ m$^{-1}\cdot$ s^{-1}
α_{th}	Thermal expansion coefficient, K^{-1}
β_s	Forchheimer coefficient, m^{-1}
$\boldsymbol{\partial_x}$	Space partial derivative, m^{-1}
χ_{ij}	Advancement of pyrolysis reaction j within phase i
Δt	Time step, s
δ	cell-to-cell distance, m
$\dot{\omega}_C$	Carbon production rate, kg $\cdot$ m^{-3} $\cdot$ s^{-1}
$\dot{\omega}_i$	Finite-rate of species i
ϵ_g	Gasous volume fraction
ϵ_s	Solid volume fraction
$\epsilon_{g,c}$	Char gaseous volume fraction
$\epsilon_{g,v}$	Virgin gaseous volume fraction

$\epsilon_{s,0}$	Initial solid volume fraction
η	Tortuosity
Γ	Resin volume fraction
γ	Heat capacity ratio
λ	Scaling factor for C'_H
λ_s	Second Lamé coefficient
μ	Dynamic viscosity, Pa $\cdot$ s
μ_g	Gaseous viscosity, kg $\cdot$ m^{-1} $\cdot$ s^{-1}
μ_s	First Lamé coefficient
$\nu_{i,k}$	Number of atoms of element k in molecule of species i
Ω_h	Heterogeneous production rate, kg $\cdot$ m^{-3} $\cdot$ s^{-1}
∂_t	Temporal partial derivative, s^{-1}
ϕ	Porosity
π_k	Elemental pyrolysis production rate, kg $\cdot$ m^{-3} $\cdot$ s^{-1}
π_{tot}	Total pyrolysis production rate, kg $\cdot$ m^{-3} $\cdot$ s^{-1}
ψ	Compressibility, s$^2\cdot$ m^{-2}
ψ_i	Exponent factor of pyrolysis reaction i, SI
ρ_c	Char solid density, kg $\cdot$ m^{-3}
ρ_g	Gaseous density, kg $\cdot$ m^{-3}
ρ_s	Solid density, kg $\cdot$ m^{-3}
ρ_v	Virgin solid density, kg $\cdot$ m^{-3}
ρ_{ext}	External solid density, kg $\cdot$ m^{-3}
$\rho_{sf,0}$	Initial carbon fiber solid density, kg $\cdot$ m^{-3}
ρ_{sv}	Virgin solid density, kg $\cdot$ m^{-3}
σ	Electrical conductivity, S $\cdot$ m^{-1}
σ	Stefan-Boltzmann constant, 5.670367×10^{-8} W $\cdot$ m^{-2} $\cdot$ K^{-4}
τ	Total advancement of pyrolysis
$\underline{\underline{\sigma_{\mathbf{s}}}}$	Engineering stress tensor, Pa
ε	Emissivity
ξ	Pyrolysis expansion coefficient

Subscript

0	Initial
∞	Infinity
adv	Advection
c	Char
$cond$	Conduction
$conv$	Convection
$diff$	Diffusion
eff	Effective
exp	Explicit
f	Fiber
$flux$	Corrected convection
max	Maximum
min	Minimum
pla	Plasma
rad	Radiation
S	Sensible
s	Solid
v	Virgin
w	Wall
ca	Char ablator
e	Boundary layer edge
pg	Pyrolysis gas

w	Wall

Superscript

in	Inside the material
out	Outside the material

Chapter 1

Introduction

Licence and disclaimers

The Porous material Analysis Toolbox based on OpenFOAM (PATO)[1] is distributed under NASA open source agreement version 1.3.

DISCLAIMERS

NO WARRANTY: THE SUBJECT SOFTWARE IS PROVIDED "AS IS" WITHOUT ANY WARRANTY OF ANY KIND, EITHER EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY THAT THE SUBJECT SOFTWARE WILL CONFORM TO SPECIFICATIONS, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR FREEDOM FROM INFRINGEMENT, ANY WARRANTY THAT THE SUBJECT SOFTWARE WILL BE ERROR FREE, OR ANY WARRANTY THAT DOCUMENTATION, IF PROVIDED, WILL CONFORM TO THE SUBJECT SOFTWARE. THIS AGREEMENT DOES NOT, IN ANY MANNER, CONSTITUTE AN ENDORSEMENT BY GOVERNMENT AGENCY OR ANY PRIOR RECIPIENT OF ANY RESULTS, RESULTING DESIGNS, HARDWARE, SOFTWARE PRODUCTS OR ANY OTHER APPLICATIONS RESULTING FROM USE OF THE SUBJECT SOFTWARE. FURTHER, GOVERNMENT AGENCY DISCLAIMS ALL WARRANTIES AND LIABILITIES REGARDING THIRD-PARTY SOFTWARE, IF PRESENT IN THE ORIGINAL SOFTWARE, AND DISTRIBUTES IT "AS IS".

WAIVER AND INDEMNITY: RECIPIENT AGREES TO WAIVE ANY AND ALL CLAIMS AGAINST THE UNITED STATES GOVERNMENT, ITS CONTRACTORS AND SUBCONTRACTORS, AS WELL AS ANY PRIOR RECIPIENT. IF RECIPIENT'S USE OF THE SUBJECT SOFTWARE RESULTS IN ANY LIABILITIES, DEMANDS, DAMAGES, EXPENSES OR LOSSES ARISING FROM SUCH USE, INCLUDING ANY DAMAGES FROM PRODUCTS BASED ON, OR RESULTING FROM, RECIPIENT'S USE OF THE SUBJECT SOFTWARE, RECIPIENT SHALL INDEMNIFY AND HOLD HARMLESS THE UNITED STATES GOVERNMENT, ITS CONTRACTORS AND SUBCONTRACTORS, AS WELL AS ANY PRIOR RECIPIENT, TO THE EXTENT PERMITTED BY LAW. RECIPIENT'S SOLE REMEDY FOR ANY SUCH MATTER SHALL BE THE IMMEDIATE, UNILATERAL TERMINATION OF THIS AGREEMENT.

[1] https://www.pato.ac

1.1 Overview

The Porous material Analysis Toolbox based on OpenFOAM (PATO) software is a modular thermal analysis platform for multiphase porous reactive materials. It can be run as a simple Fourier heat transfer code or include more advanced features as internal decomposition (pyrolysis, vaporization), gas-gas and gas-solid chemical interactions (combustion, cracking, cooking), gas species transport (convection, diffusion), and solid morphology evolutions (internal density changes, surface ablation). PATO is implemented as a C++ top level module of the open source (GNU GPL) computational fluid dynamics software program OpenFOAM. PATO also uses the open source (GNU LGPL) thermodynamics, transport, and chemistry library Mutation++ produced by the von Karman Institute for Fluid Dynamics. Tutorials are provided for different types of geometries (1D, 2D, 2D-axi, 3D), boundary conditions (imposed temperature and pressure, convective boundary layer, coupled to external flow) and materials (metals, composites, wood, granular beds). The computational model is a generic heat and mass transfer model for porous reactive materials containing several solid phases and a single gas phase. The detailed chemical interactions occurring between the solid phases and the gas phase are modeled at the pore scale assuming local thermal equilibrium: solid pyrolysis, pyrolysis species injection in the gas phase, heterogeneous reactions between the solid phases and the gas phase, and homogeneous reactions in the gas phase. The chemistry models are integrated in a macroscopic model derived by volume-averaging the governing equations for the conservation of solid mass, gas mass, species (finite-rate chemistry) or elements (equilibrium chemistry), momentum, and energy. This generic model is implemented in the PATO architecture. Figure 1.1 shows the overview of the PATO software.

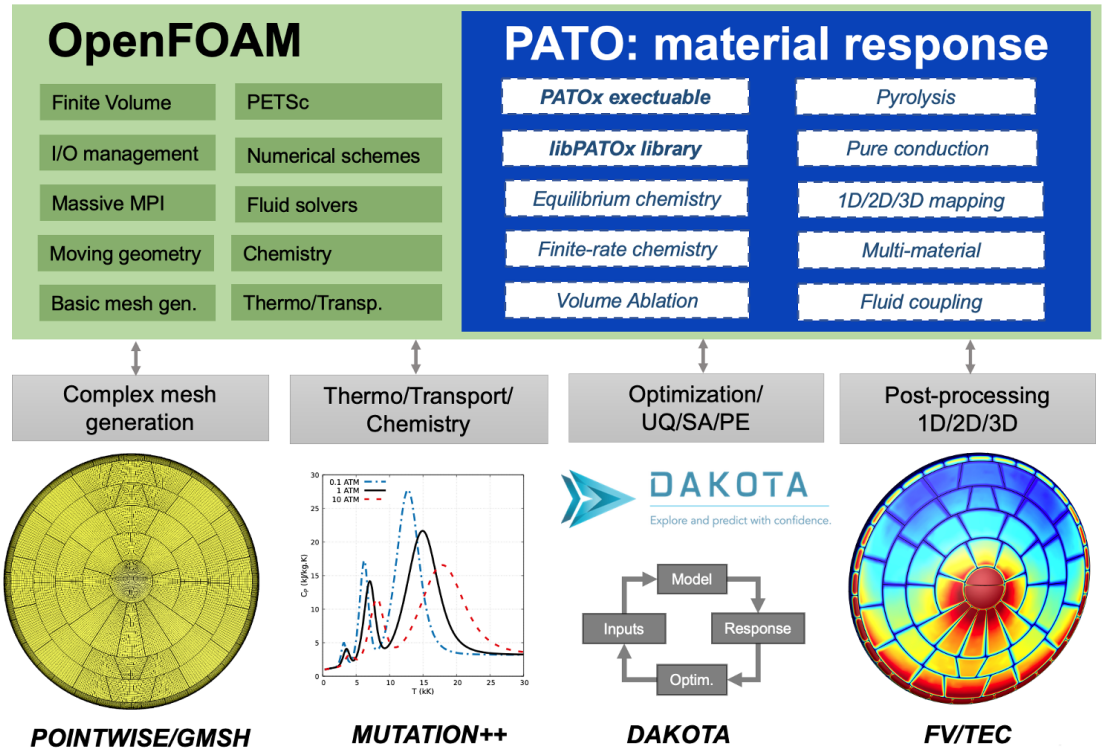

Figure 1.1. Overview of the PATO software which was implemented as a top level module of OpenFOAM enabling the coupling to mesh generation, thermochemistry, optimization, and post-processing libraries.

1.2 Architecture

The PATO software architecture is described in Fig. 1.2. The *AllwcleanAllclean* and *Allwmake* files are used to compile the PATO source code and third party. The *bashrc* file has all the needed environment variables and needs to be added to the user's bashrc file before to compile. The data folder includes the environment, fluid, material and thermodynamic/transport/chemistry properties. The documentation folder contains the main PATO publications, the fields description, the Doxygen source code and the legal Open Source Agreement. The *README.md* explains the PATO installation steps. The tutorials folder provides test cases for different types of geometry in 0D, 1D, 2D and 3D. The src folder includes the main library (**libPATOx**), the main solver (**PATOx**), various basic solver for simple cases (**basic**), the utilities including the unit testing framework.

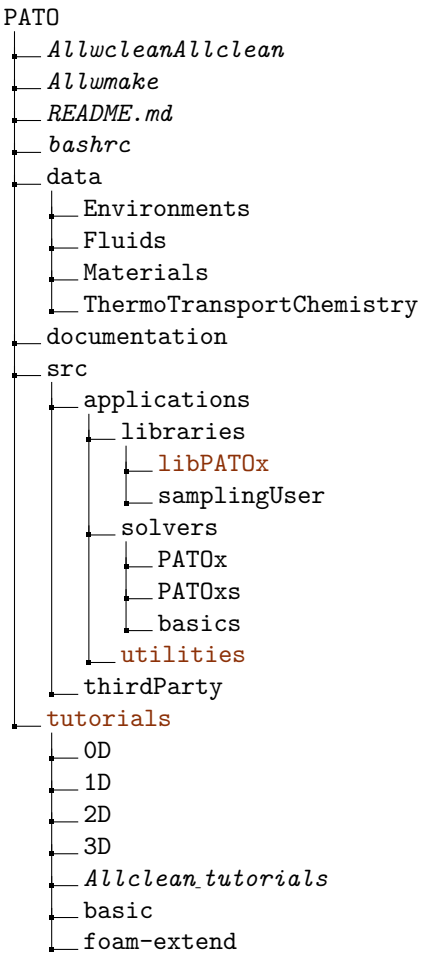

Figure 1.2. PATO architecture

Figure 1.3 shows the main PATO library, **libPATOx**, which includes the **Fluid** and **Material** models. *makeFluidModel.C* file constructs the different **Fluid** types based on the common **basicFluidModel** class. The **fvPatches** folder contains the **Fluid** boundary conditions patches. The **thermos** folder includes the 2 thermophysical models. The **Material** model is composed by sub-models and each sub-model has its own types. The **BoundaryConditions** folder includes the different classes and patches of the material boundary conditions. The **simpleMaterials** class creates the different material regions. The **simpleMaterial** class creates the different material sub-models, as described in Chapter 2.

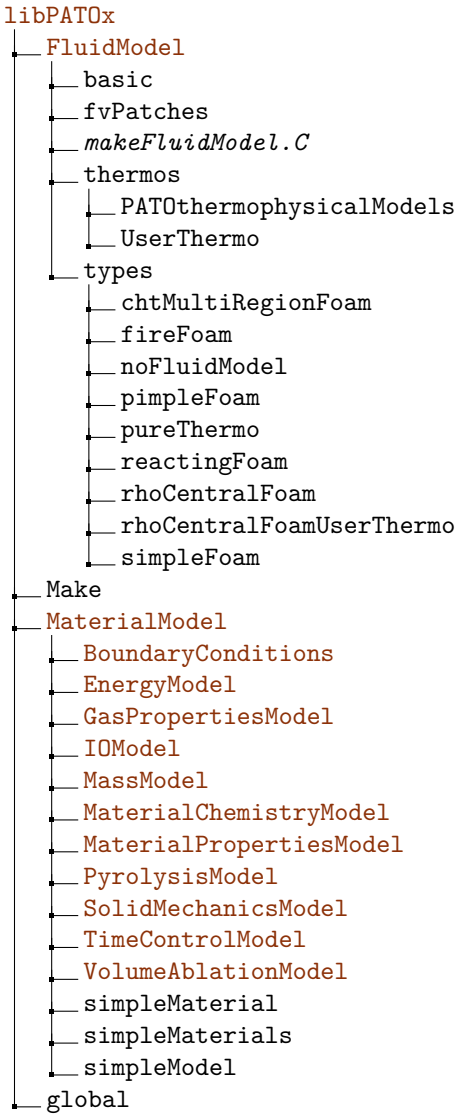

Figure 1.3. libPATOx architecture

Chapter 2

Material Model

The **simpleMaterials** class creates the static and dynamic meshes of the different material regions by reading the *regionProperties* file. The **simpleMaterial** class creates the following material sub-models, based on the **simpleModel** class, for each region and updates them over time:

- **EnergyModel**
- **GasPropertiesModel**
- **MassModel**
- **MaterialChemistryModel**
- **MaterialPropertiesModel**
- **PyrolysisModel**
- **IOModel**
- **SolidMechanicsModel**
- **TimeControlModel**
- **VolumeAblationModel**

The creation of models, fields, and properties are described in the **PATOx** output during the initialization of the **MaterialModel**. An example is given in Fig. 2.1.

```
=== BEGINNING MATERIAL: porousMat ===
Create solid mesh for region porousMat for time = 0
| Create "MaterialChemistryModel" type "no"
| Create "PyrolysisModel" type "virgin"
.  |-- Reading "nSolidPhases" scalar property from
       "$PATO_DIR/data/Materials/Composites/TACOT/constantProperties":
       nSolidPhases=2
.  |-- Create "tau" volField (READ_IF_PRESENT)
| Create "MaterialPropertiesModel" type "Porous"
.  |--| Create "MassModel" type "no"
.  .  |-- Create "p" volField (MUST_READ)
.  |-- Reference to "p" volField from Mass
.  |--| Create "EnergyModel" type "PureConduction"
.  .  |-- Create "Ta" volField (MUST_READ)
.  .  |-- Create "rho_s" volField (READ_IF_PRESENT)
.  .  |-- Create "k" volField (READ_IF_PRESENT)
.  |-- Reference to "Ta" volField from Energy
.  |-- Reference to "rho_s" volField from Energy
.  |-- Reference to "tau" volField from Pyrolysis
| Create "GasPropertiesModel" type "no"
| Reference to "MassModel" type "no"
| Reference to "EnergyModel" type "PureConduction"
| Create "SolidMechanicsModel" type "no"
| Create "VolumeAblationModel" type "no"
| Create "IOModel" type "no"
.  |-- Reference to "MaterialPropertiesModel" type "Porous"
.  |--| Initialize "MaterialProperties" type "Porous"
.  .  |-- "K" not found in mesh. It will not be updated.
.  .  |-- Reading virgin material properties
.  .  |-- The virgin material table has 28 lines.
.  .  |-- Reading char material properties
.  .  |-- The char material table has 28 lines.
.  .  |-- "emissivity" not found in mesh. It will not be updated.
.  .  |-- "absorptivity" not found in mesh. It will not be updated.
.  .  |-- Reference to "rho_s[1]" volField from Pyrolysis
.  .  |-- Reference to "rho_s[2]" volField from Pyrolysis
.  .  |-- Reference to "eps_s[1]" volField from Pyrolysis
.  .  |-- Reference to "eps_s[2]" volField from Pyrolysis
| Create "TimeControlModel" type "no"
=== END MATERIAL: porousMat ===
```

Figure 2.1. Example: initialization of the porousMat **MaterialModel**.

2.1 Mass Model

Figure 2.2 shows the architecture of the Mass model. The *makeMassModel.C* creates the different Mass types based on the **simpleMassModel** class. The different types of the Mass model are described below.

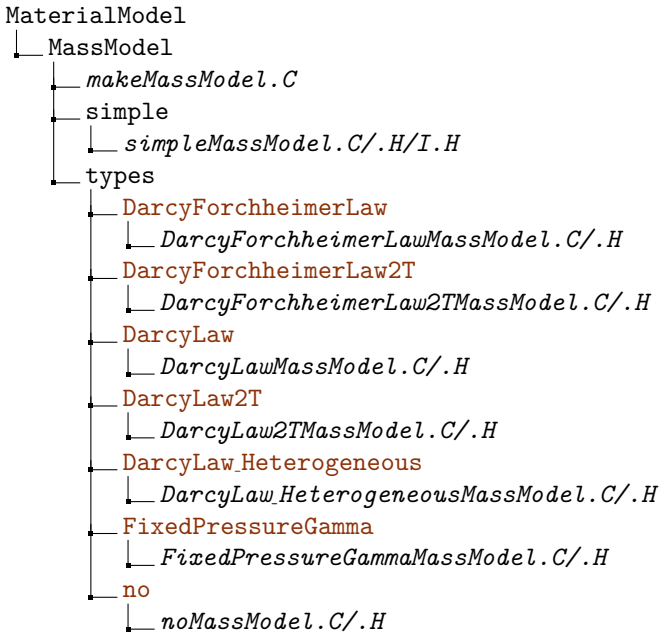

Figure 2.2. Mass model architecture

2.1.1 no Type

The **no** type is empty and serves as a place holder when the **MassModel** is not used.

2.1.2 DarcyLaw Type

The **DarcyLaw** type solves the semi-implicit pressure equation (Eq. 2.4). This equation comes from the perfect gas law (Eq. 2.1), the mass conservation (Eq. 2.2) and the Darcy's law (Eq. 2.3). The right-hand side is the total pyrolysis gas production term and is computed in the **PyrolysisModel**. The gaseous thermodynamic and transport properties are computed in the **GasPropertiesModel**. The solid material properties are computed in the **MaterialPropertiesModel**.

$$\rho_g = \frac{M_g\ p_g}{R\ T_a} \tag{2.1}$$

$$\partial_t\left(\rho_g\right) + \boldsymbol{\partial_x} \cdot \left(\epsilon_g \rho_g \boldsymbol{v_g}\right) = \pi_{tot} \tag{2.2}$$

$$\boldsymbol{v_g} = -\frac{1}{\epsilon_g}\left(\frac{1}{\mu_g}\overline{\overline{\mathbf{K_s}}}\right) \cdot \boldsymbol{\partial_x}\ p_g \tag{2.3}$$

$$\partial_t\left(\frac{\epsilon_g\ M_g\ p_g}{R\ T_a}\right) - \boldsymbol{\partial_x} \cdot \left(\frac{M_g\ p_g}{\mu_g\ R\ T_a}\overline{\overline{\mathbf{K_s}}}\ \boldsymbol{\partial_x}\ p_g\right) = \pi_{tot} \tag{2.4}$$

2.1.3 DarcyLaw_Heterogeneous Type

The **DarcyLaw_Heterogeneous** type solves the semi-implicit pressure equation (Eq. 2.5), based on the **DarcyLaw** type. The heterogeneous production rate (Eq. 2.6), updated in the **MaterialChemistryModel**, is added as source term in the right-hand side.

$$\partial_t \left(\frac{\epsilon_g \; M_g \; p_g}{R \; T_a} \right) - \boldsymbol{\partial_x} \cdot \left(\frac{M_g \; p_g \; \underline{\underline{\mathbf{K_s}}}}{\mu_g \; R \; T_a} \; \boldsymbol{\partial_x} \; p_g \right) = \pi_{tot} + \Omega_h \tag{2.5}$$

$$\Omega_h = -\dot{\omega}_C \tag{2.6}$$

2.1.4 DarcyLaw2T Type

The **DarcyLaw2T** type solves the semi-implicit pressure equation (Eq. 2.7), based on the **DarcyLaw** type. Porous material is considered in thermal non-equilibrium ($T_g \neq T_s \neq T_a$) and gaseous temperature is used instead of porous material equilibrium temperature.

$$\partial_t \left(\frac{\epsilon_g \; M_g \; p_g}{R \; T_g} \right) - \boldsymbol{\partial_x} \cdot \left(\frac{M_g \; p_g \; \underline{\underline{\mathbf{K_s}}}}{\mu_g \; R \; T_g} \; \boldsymbol{\partial_x} \; p_g \right) = \pi_{tot} \tag{2.7}$$

2.1.5 DarcyForchheimerLaw Type

The **DarcyForchheimerLaw** type solves the semi-implicit pressure equation (Eq. 2.10). This equation comes from the perfect gas law (Eq. 2.1), the mass conservation (Eq. 2.2) and the Darcy-Forchheimer's law (Eq. 2.8). The tensor $\underline{\underline{\mathbf{X_s}}}$ (Eq. 2.9) is introduced for convenience.

$$\boldsymbol{v_g} = -\frac{1}{\epsilon_g} \left(\underline{\underline{\mathbf{K_s}}} \, \underline{\underline{\mathbf{X_s}}} \right) \cdot \boldsymbol{\partial_x} \; p_g \tag{2.8}$$

$$X_{ij} = \frac{1}{\mu \, K_{ij} + \beta_{ij} \, \rho_g \, |\boldsymbol{U}|} \tag{2.9}$$

$$\partial_t \left(\frac{\epsilon_g \; M_g \; p_g}{R \; T_a} \right) - \boldsymbol{\partial_x} \cdot \left(\frac{M_g \; p_g \; \underline{\underline{\mathbf{K_s}}} \; \underline{\underline{\mathbf{X_s}}}}{R \; T_a} \; \boldsymbol{\partial_x} \; p_g \right) = \pi_{tot} \tag{2.10}$$

2.1.6 DarcyForchheimerLaw2T Type

The **DarcyForchheimerLaw2T** type solves the semi-implicit pressure equation (Eq. 2.11), based on the **DarcyForchheimerLaw** type. Porous material is considered in thermal non-equilibrium ($T_g \neq T_s \neq T_a$) and gaseous temperature is used instead of porous material equilibrium temperature.

$$\partial_t \left(\frac{\epsilon_g \; M_g \; p_g}{R \; T_g} \right) - \boldsymbol{\partial_x} \cdot \left(\frac{M_g \; p_g \; \underline{\underline{\mathbf{K_s}}} \; \underline{\underline{\mathbf{X_s}}}}{R \; T_g} \; \boldsymbol{\partial_x} \; p_g \right) = \pi_{tot} \tag{2.11}$$

2.1.7 FixedPressureGamma Type

The **FixedPressureGamma** type fixes the value of the pressure field, the gamma (Eq. 2.12) field and the boundary conditions. The user can modify the following inputs in the model properties file:

- **internalField_P** = internal cell center value of the scalar pressure.
- **boundaryField_P** = boundary face center value of the scalar pressure.
- **internalField_Gamma** = internal cell center value of the tensor gamma.

- **boundaryField_Gamma** = boundary face center value of the tensor gamma.

$$\underline{\underline{\gamma_{\mathbf{g}}}} = \frac{p_g M_g}{\mu_g R T_g} \underline{\underline{\mathbf{K_s}}} \tag{2.12}$$

2.2 Energy Model

Figure 2.3 shows the architecture of the Energy model. The *makeEnergyModel.C* creates the different Energy types based on the **simpleEnergyModel** class. The Energy types are described below.

Figure 2.3. Energy model architecture

2.2.1 no Type

The **no** type is empty and serves as a place holder when the **EnergyModel** is not used.

2.2.2 PureConduction Type

The **PureConduction** type updates the temperature by solving the energy equation (Eq. 2.13). The solid material properties are computed in the **MaterialPropertiesModel**.

$$\rho_s c_{p,s}\ \partial_t T - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\mathbf{k_s}}}\ \boldsymbol{\partial_x} T \right) = 0 \tag{2.13}$$

2.2.3 Pyrolysis Type

The **Pyrolysis** type updates the temperature by solving the energy equation (Eq. 2.14). The left-hand side is composed by the solid and gaseous storage, the solid thermal conduction, the gaseous thermal convection and the pyrolysis energy flux. The gaseous thermodynamic and transport properties are computed in the **GasPropertiesModel**. The solid material properties and the pyrolysis energy flux are computed in the **MaterialPropertiesModel**.

$$\rho_s c_{p,s}\ \partial_t T_a + \partial_t \left(\epsilon_g \rho_g E_g \right) - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\mathbf{k_s}}}\ \boldsymbol{\partial_x} T_a \right) - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\gamma_\mathbf{g}}}\ \boldsymbol{\partial_x} p_g \right) + \dot{E}_p = 0 \tag{2.14}$$

$$\underline{\underline{\gamma_{\mathbf{hg}}}} = \frac{h_g p_g M_g}{\mu_g R T_a}\ \underline{\underline{\mathbf{K_s}}} \tag{2.15}$$

2.2.4 Pyrolysis_Heterogeneous_SpeciesDiffusion Type

The **Pyrolysis_Heterogeneous_SpeciesDiffusion** type updates the temperature by solving the energy equation (Eq. 2.16) based on the **Pyrolysis** type. The heterogeneous energy flux (Eq. 2.17) and the energy flux transported by diffusion of the gaseous species (Eq. 2.18) are added as source term to the right-hand side.

$$\rho_s c_{p,s}\ \partial_t T_g + \partial_t \left(\epsilon_g \rho_g E_g \right) - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\mathbf{k_s}}}\ \boldsymbol{\partial_x} T_g \right) - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\gamma_\mathbf{g}}}\ \boldsymbol{\partial_x} p_g \right) + \dot{E}_\Omega = \dot{E}_D + \bar{h} \pi_{tot} \tag{2.16}$$

$$\dot{E}_\Omega = -h_c\ \Omega_h = h_c\ \dot{\omega}_{C(gr)} \tag{2.17}$$

$$\dot{E}_D = \sum_i^{N_s}\ \boldsymbol{\partial_x} \cdot \left(\frac{D_{m,i}\ \epsilon_g\ \rho_g\ h_i}{\eta}\ \boldsymbol{\partial_x} y_i \right) \tag{2.18}$$

2.2.5 Pyrolysis2T Type

The **Pyrolysis2T** type updates the solid temperature and the gas temperature by solving two energy equation (Eq. 2.19). The left-hand side of Eq. 2.19a is composed by the solid storage, the solid thermal conduction, the pyrolysis energy flux and the exchange between solid and fluid. The left-hand side of Eq. 2.19b is composed by the gaseous storage, the pressure energy, the gaseous convection, the gas thermal conduction and the exchange between solid and fluid. The gaseous thermodynamic and transport properties are computed in the **GasPropertiesModel**. The solid material properties and the pyrolysis energy flux are computed in the **MaterialPropertiesModel**.

$$\rho_s c_{p,s}\ \partial_t T_s - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\mathbf{k_s}}}\ \boldsymbol{\partial_x} T_s\right) + \dot{E}_p + H\left(T_s - T_g\right) = 0 \tag{2.19a}$$

$$\partial_t\left(\epsilon_g \rho_g E_g\right) - \partial_t\left(\epsilon_g p_g\right) - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\gamma_\mathbf{g}}}\ \boldsymbol{\partial_x} p_g\right) + H\left(T_g - T_s\right) = 0 \tag{2.19b}$$

$$\underline{\underline{\gamma_\mathbf{hg}}} = \frac{h_g p_g M_g}{\mu_g R T_g}\ \underline{\underline{\mathbf{K_s}}} \tag{2.20}$$

2.2.6 ForchheimerPyrolysis Type

The **ForchheimerPyrolysis** type updates the temperature by solving the energy equation (Eq. 2.21) based on the **Pyrolysis** type with gaseous thermal convection term calculated by Eq. 2.22.

$$\rho_s c_{p,s}\ \partial_t T_a + \partial_t\left(\epsilon_g \rho_g E_g\right) - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\mathbf{k_s}}}\ \boldsymbol{\partial_x} T_a\right) - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\gamma_\mathbf{g}}}\ \boldsymbol{\partial_x} p_g\right) + \dot{E}_p = 0 \tag{2.21}$$

$$\underline{\underline{\gamma_\mathbf{hg}}} = \frac{h_g\ M_g\ p_g\ \underline{\underline{\mathbf{K_s}}}\ \underline{\underline{\mathbf{X_s}}}}{R\ T_a} \tag{2.22}$$

2.2.7 ForchheimerPyrolysis2T Type

The **ForchheimerPyrolysis2T** type updates the solid temperature and the gas temperature by solving two energy equation (Eq. 2.23) based on the **Pyrolysis2T** type with gaseous thermal convection term calculated by Eq. 2.24.

$$\rho_s c_{p,s}\ \partial_t T_s - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\mathbf{k_s}}}\ \boldsymbol{\partial_x} T_s\right) + \dot{E}_p + H\left(T_s - T_g\right) = 0 \tag{2.23a}$$

$$\partial_t\left(\epsilon_g \rho_g E_g\right) - \partial_t\left(\epsilon_g p_g\right) - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\gamma_\mathbf{g}}}\ \boldsymbol{\partial_x} p_g\right) + H\left(T_g - T_s\right) = 0 \tag{2.23b}$$

$$\underline{\underline{\gamma_\mathbf{hg}}} = \frac{h_g\ M_g\ p_g\ \underline{\underline{\mathbf{K_s}}}\ \underline{\underline{\mathbf{X_s}}}}{R\ T_g} \tag{2.24}$$

2.2.8 Darcy2T Type

The **Darcy2T** type updates the temperature by solving energy equations (Eq. 2.25) based on the **Pyrolysis2T** type with constant solid heat capacity for materials without pyrolysis.

$$\rho_s c_{p,s}\ \partial_t T_s - \boldsymbol{\partial_x} \cdot \left(\underline{\underline{\mathbf{k_s}}}\ \boldsymbol{\partial_x} T_s\right) + \dot{E}_p + H\left(T_s - T_g\right) = 0 \tag{2.25a}$$

$$\partial_t\left(\epsilon_g \rho_g E_g\right) - \partial_t\left(\epsilon_g p_g\right) - c_p\ \boldsymbol{\partial_x} \cdot \left(\epsilon_g \rho_g \boldsymbol{v_g} T_g\right) + H\left(T_g - T_s\right) = 0 \tag{2.25b}$$

2.2.9 Forchheimer2T Type

The **Forchheimer2T** type is identical to the **Darcy2T** type and will be deleted in a future release.

2.2.10 CorrectBC Type

The **CorrectBC** type updates all the boundary conditions of the temperature.

2.2.11 BoundaryTable Type

The **BoundaryTable** type updates the internal cell center value of the temperature using the value of the **uniformFixedValue** boundary face.

2.2.12 FixedTemperature Type

The **FixedTemperature** type fixes the value of the temperature fields and boundary conditions. The user can modify the following inputs in the model properties file:

- **internalField_T** = internal cell center value of the temperature.
- **boundaryField_T** = boundary face center value of the temperature.

2.2.13 TabulatedTemperature Type

The **TabulatedTemperature** type fixes the value of the temperature fields from an external file and boundary conditions from the inputs. Internal fields are mapped from a tabulated value and a distance from a given patch. The user can modify the following inputs in the model properties file:

- **internalFieldTable_fileName** = file containing table of internal cell center value of the temperature.
- **originPatchName** = name of the patch from which the temperature values will be mapped.
- **boundaryField_T** = boundary face center value of the temperature.

2.2.14 ControlTemperature Type

The **ControlTemperature** type fixes a linear time dependent value of temperature inside the domain (Eq. 2.26). The user can modify the following inputs in the model properties file:

- **a** = temperature at $t = 0$.
- **b** = slope factor.

$$T_a\left(t\right) = a + b\ t \tag{2.26}$$

2.3 Input/Output (IO) Model

Figure 2.4 shows the architecture of the Input/Output (IO) model. The **SampleFunction** class creates the PATO sampling files and provides the PATO sampling methods based on the probing functions in OpenFOAM. The *makeIOModel.C* file creates the different IO types based on the **simpleIOModel** class. The IO types are described below.

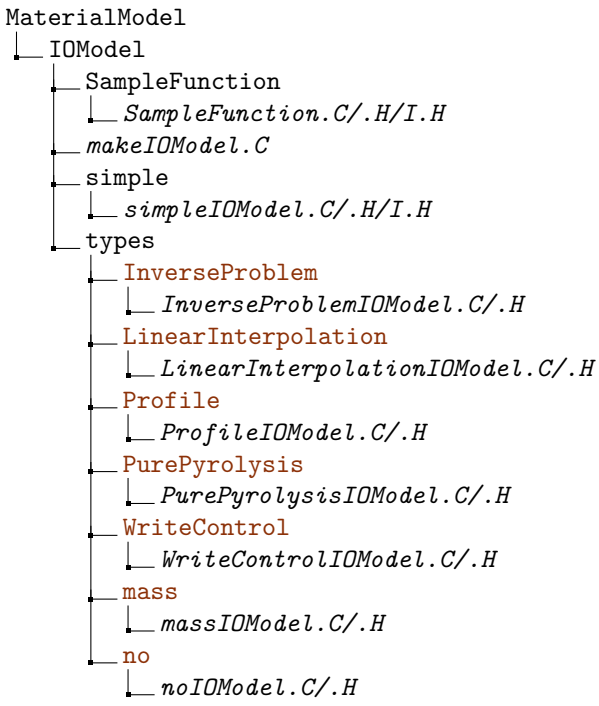

Figure 2.4. IO model architecture

2.3.1 no Type

The **no** type writes the fields from the **writeFields** input following the *controlDict* file and writes the probing fields from the **probingFunctions** input to the output folder.

2.3.2 PurePyrolysis Type

The **PurePyrolysis** type computes the solid density ratio of virgin and char (Eq. 2.27) and the normalized total pyrolysis production rate (Eq. 2.28) in function of time. This type compares also $r(t)$ and $\widetilde{\pi}_{tot}(t)$ to reference files and gives the least square difference (Eq. 2.29).

$$r(t) = \frac{\rho_s(t)}{\rho_{sv}} \tag{2.27}$$

$$\widetilde{\pi}_{tot}(t) = -\partial_t r(t) = \frac{\pi_i(t)}{\rho_{sv}} \tag{2.28}$$

$$L = \sum_i^{N_t} \left(f(t_i) - f_{ref}(t_i)\right)^2 \qquad \text{with } f(t_i) = [r(t_i), \widetilde{\pi}_{tot}(t_i)] \tag{2.29}$$

2.3.3 InverseProblem Type

The **InverseProblem** type takes a user list of fields and reference files to compare. Then, this type computes the least square difference (Eq. 2.30).

$$L = \sum_{i}^{N_t} \left(f(t_i) - f_{ref}(t_i)\right)^2 \tag{2.30}$$

2.3.4 Profile Type

The **Profile** type takes a user list of fields. Then, this types computes and writes the 1D in-depth profile of these fields.

2.3.5 LinearInterpolation Type

The **LinearInterpolation** type initializes the fields from a user list with a linear interpolation (Eq. 2.31).

$$f = \begin{cases} f_I + (f_T - f_I)\frac{y_C - y_I}{y_T - y_I} & \text{if } y_C > y_I \\ f_B + (f_I - f_B)\frac{y_C - y_B}{y_I - y_B} & \text{else} \end{cases} \tag{2.31}$$

2.3.6 mass Type

The **Mass** type computes the gas mass flow rate at the surface, the char mass flow rate, the total recession of the material and the position when the sample reached 2% and 98% of the charred state.

2.3.7 WriteControl Type

The **WriteControl** type writes the fields from **IO:writeFields** at the additional times provided in **additionalWriteTimes**. This type also writes the probing fields from **IO:probingFunctions** at the additional times provided in **additionalProbingTimes**. The user can modify the following inputs in the model properties file:

- **additionalWriteTimes** = list of additional times to write the fields.
- **additionalProbingTimes** = list of additional times to write the probing fields.

2.4 Volume Ablation Model

Figure 2.5 shows the architecture of the Volume Ablation model. The *makeVolumeAblationModel.C* file creates the different Volume Ablation types based on the **simpleVolumeAblationModel** class. The Volume Ablation types are described below.

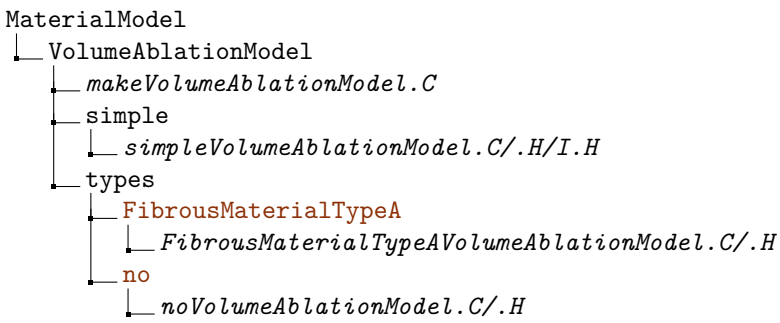

Figure 2.5. VolumeAblation model architecture

2.4.1 no Type

The **no** type is empty and serves as a place holder when the **VolumeAblationModel** is not used.

2.4.2 FibrousMaterialTypeA Type

The **FibrousMaterialTypeA** type updates the solid volume fraction by solving the Equation 2.32. This equation includes the external solid density (Eq. 2.34) and the heterogeneous production rate (Eq. 2.33), updated in the **MaterialChemistryModel** model. The fiber radius (Eq. 2.35), the specific surface (Eq. 2.36), the site density (Eq. 2.37) and the solid carbon mass fraction (Eq. 2.38) are then computed in function of the solid volume fraction.

$$\partial_t \epsilon_s + \frac{\Omega_h}{\rho_{ext}} = 0 \tag{2.32}$$

$$\Omega_h = -\dot{\omega}_C \tag{2.33}$$

$$\rho_{ext} = \begin{cases} \rho_{sf,0} & \text{if } r_T < r_{f,0} \\ \rho_s & \text{else} \end{cases} \tag{2.34}$$

$$r_T = r_{T,0} \sqrt{\frac{\epsilon_s}{\epsilon_{s,0}}} \tag{2.35}$$

$$S_s = \frac{2\ r_T\ \epsilon_{s,0}}{(r_{T,0})^2} \tag{2.36}$$

$$S_d = S_{d,0} \left[1 - exp\left(\frac{-r_T}{r_{ff}} \right) \right] \tag{2.37}$$

$$y_{C(gr)} = S_s S_d \tag{2.38}$$

2.5 Gas Properties Model

Figure 2.6 shows the architecture of the Gas Properties model. The *makeGasPropertiesModel.C* file creates the different Gas Properties types based on the **simpleGasPropertiesModel** class. The Gas Properties types are described below.

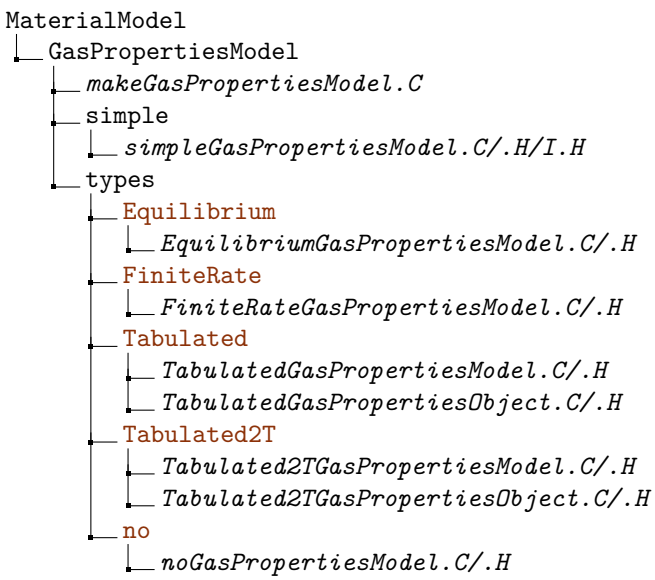

Figure 2.6. GasProperties model architecture

2.5.1 no Type

The **no** type is empty and serves as a place holder when the **GasPropertiesModel** is not used.

2.5.2 Equilibrium Type

The **Equilibrium** type updates the following gaseous transport and thermodynamic properties in function of pressure, temperature and elemental mole composition using the Mutation^{++}library. The total advancement of pyrolysis (τ) is computed in the **PyrolysisModel**. The virgin and charred properties are given by the user in the constant material properties file, read in the **MaterialPropertiesModel**.

- The gaseous molar mass, M_g
- The gaseous enthalpy, h_g
- The gaseous viscosity, μ_g
- The gaseous density, ρ_g
- The gaseous volume fraction, $\epsilon_g = \epsilon_{g,c} + (\epsilon_{g,v} - \epsilon_{g,c})\ \tau$

2.5.3 FiniteRate Type

The **FiniteRate** type updates the following gaseous transport and thermodynamic properties in function of pressure, temperature and species mass composition using the Mutation^{++}library. The total advancement of pyrolysis (τ) is computed in the **PyrolysisModel**. The virgin and charred properties are given by the user in the constant material properties file, read in the **MaterialPropertiesModel**.

- The gaseous molar mass, M_g
- The gaseous enthalpy, h_g
- The gaseous viscosity, μ_g
- The gaseous density, ρ_g
- The gaseous volume fraction, $\epsilon_g = \epsilon_{g,c} + (\epsilon_{g,v} - \epsilon_{g,c})\ \tau$
- Energy flux transported by diffusion of the species, $\dot{E}_D$

2.5.4 Tabulated Type

The **Tabulated** type updates the following gaseous transport and thermodynamic properties in function of pressure and temperature using a linear interpolation method and a table given by the user. The total advancement of pyrolysis (τ) is computed in the **PyrolysisModel**. The virgin and charred properties are given by the user in the constant material properties file, read in the **MaterialPropertiesModel**.

- The gaseous molar mass, M_g
- The gaseous enthalpy, h_g
- The gaseous viscosity, μ_g
- The gaseous density, ρ_g
- The gaseous volume fraction, $\epsilon_g = \epsilon_{gc,} + (\epsilon_{g,v} - \epsilon_{g,c})\ \tau$

2.5.5 Tabulated2T Type

The **Tabulated2T** type updates the following gaseous transport and thermodynamic properties in function of pressure and gas temperature using a linear interpolation method and a table given by the user. The total advancement of pyrolysis (τ) is computed in the **PyrolysisModel**. The virgin and charred properties are given by the user in the constant material properties file, read in the **MaterialPropertiesModel**.

- The gaseous molar mass, M_g
- The gaseous enthalpy, h_g
- The gaseous viscosity, μ_g
- The gaseous thermal conductivity k_g
- The gaseous density, ρ_g
- The gaseous volume fraction, $\epsilon_g = \epsilon_{gc,} + (\epsilon_{g,v} - \epsilon_{g,c})\ \tau$

2.6 Material Properties Model

Figure 2.7 shows the architecture of the Material Properties model. The *makeMaterialPropertiesModel.C* file creates the different Material Properties types based on the **simpleMaterialPropertiesModel** and **CommonMaterialPropertiesModel** classes. Only the fields that already exist in the mesh are updated in this model. The classes in the **Object** folder are used by different Material Properties types to interpolate values. The Material Properties types are described below.

Figure 2.7. MaterialProperties model architecture

2.6.1 no Type

The **no** type is empty and serves as a place holder when the **MaterialPropertiesModel** is not used.

2.6.2 Fourier Type

The **Fourier** type updates the following solid properties (if they already exist in the mesh) using the Fourier's law.

- The solid thermal conductivity, $\underline{\underline{\mathbf{k_s}}} = \sum_{i=0}^{N_c} k_i \, T^i$
- The solid heat capacity, $c_{p,s} = \sum_{i=0}^{N_c} c_{ps,i} \, T^i$

- The solid density, $\rho_s = \sum_{i=0}^{N_c} \ \rho_{s,i} \ T^i$
- The solid Young's modulus $E_s = \sum_{i=0}^{N_c} \ E_{s,i} \ T^i$
- The solid Poisson's ration $\nu_s = \sum_{i=0}^{N_c} \ \nu_{s,i} \ T^i$
- The solid thermal expansion coefficient $\alpha_{th} = \sum_{i=0}^{N_c} \ \alpha_{th,i} \ T^i$
- The solid Jomaa function $\xi_s = \sum_{i=0}^{N_c} \ \xi_{s,i} \ T^i$

2.6.3 Fourier_Radiation Type

The **Fourier_Radiation** type updates the following solid properties (if they already exist in the mesh) using the Fourier's law.

- The solid thermal conductivity, $\underline{\underline{\mathbf{k_s}}} = \sum_{i=0}^{N_c} \ k_i \ T^i$
- The solid heat capacity, $c_{p,s} = \sum_{i=0}^{N_c} \ c_{ps,i} \ T^i$
- The solid density, $\rho_s = \sum_{i=0}^{N_c} \ \rho_{s,i} \ T^i$
- The solid emissivity, $\varepsilon_s = \sum_{i=0}^{N_c} \ \varepsilon_{s,i} \ T^i$
- The radiative heat flux, $q_r = -\varepsilon_s \sigma_{SB} \left(T^4 - T_\infty^4\right)$
- The solid Young's modulus $E_s = \sum_{i=0}^{N_c} \ E_{s,i} \ T^i$
- The solid Poisson's ration $\nu_s = \sum_{i=0}^{N_c} \ \nu_{s,i} \ T^i$
- The solid thermal expansion coefficient $\alpha_{th} = \sum_{i=0}^{N_c} \ \alpha_{th,i} \ T^i$
- The solid Jomaa function $\xi_s = \sum_{i=0}^{N_c} \ \xi_{s,i} \ T^i$

2.6.4 Porous_const Type

The **Porous_const** type updates the solid properties (if they already exist in the mesh) using constant values from the *constantProperties* file located in the **MaterialPropertiesDirectory** folder.

- The solid thermal conductivity, $k_{s,xx} = k_{s,yy} = k_{s,zz} =$ **k_const**
- The solid heat capacity, $c_{p,s} =$ **cp_const**
- The solid density, $\rho_s =$ **rho_s_const**
- The solid permeability, $K_{s,xx} = K_{s,yy} = K_{s,zz} =$ **K_const**
- The solid Forchheimer coefficient, $\beta_{s,xx} = \beta_{s,yy} = \beta_{s,zz} =$ **Beta_const**
- The pyrolysis energy flux, $\dot{E}_p =$ **pyrolysisFlux_const**
- The solid emissivity, $\varepsilon_s =$ **emissivity_const**
- The solid absorptivity, $\alpha_s =$ **absorptivity_const**
- The solid Young's modulus $E_s =$ **E_const**
- The solid Poisson's ration $\nu_s =$ **nu_const**
- The solid thermal expansion coefficient $\alpha_{th} =$ **alpha_const**
- The solid charred enthalpy, $h_c =$ **h_c_const**
- The solid Jomaa function, $\xi_s =$ **xi_const**

2.6.5 Porous Type

The **Porous** type updates the following solid properties (if they already exist in the mesh) using the solid density and the total advancement of pyrolysis, updated in the **PyrolysisModel**.

- The solid thermal conductivity, $\underline{\underline{\mathbf{k_s}}} = \underline{\underline{\mathbf{k_{sv}}}}\ \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})} + \underline{\underline{\mathbf{k_{sc}}}}\left(1-\tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})}\right)$
- The solid heat capacity, $c_{p,s} = c_{p,sv}\ \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})} + c_{p,sc}\left(1-\tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})}\right)$
- The solid density, $\rho_s = \sum_i^{N_p} \rho_{s,i}\ \epsilon_{s,i}$
- The solid phase density, $\rho_{s,i} = \frac{\epsilon_{sv,i}}{\epsilon_{s,i}}\rho_{sv,i}\left(1-\sum_j^{N_{sp}} \mathcal{F}_{ij}\chi_{ij}\right)$
- The solid permeability, $\underline{\underline{\mathbf{K_s}}} = \underline{\underline{\mathbf{K_{sc}}}} + (\underline{\underline{\mathbf{K_{sv}}}} - \underline{\underline{\mathbf{K_{sc}}}})\ \tau$
- The solid Forchheimer coefficient, $\underline{\underline{\beta_{\mathbf{s}}}} = \underline{\underline{\beta_{\mathbf{sc}}}} + (\underline{\underline{\beta_{\mathbf{sv}}}} - \underline{\underline{\beta_{\mathbf{sc}}}})\ \tau$
- The solid average enthalpy, $\bar{h} = \frac{\rho_{sv}h_{sv} - \rho_{sc}h_{sc}}{\rho_{sv}-\rho_{sc}}$
- The pyrolysis energy flux, $\dot{E}_p = -\bar{h}\pi_{tot}$
- The solid emissivity, $\varepsilon_s = \varepsilon_{sv}\ \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})} + \varepsilon_{sc}\left(1-\tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})}\right)$
- The solid absorptivity, $\alpha_s = \alpha_{sv}\ \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})} + \alpha_{sc}\left(1-\tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})}\right)$
- The solid Young's modulus $E_s = E_{sv}\ \tau + E_{sc}\,(1-\tau)$
- The solid Poisson's ration $\nu_s = \nu_{sv}\ \tau + \nu_{sc}\,(1-\tau)$
- The solid thermal expansion coefficient $\alpha_{th} = \alpha_{th,v}\ \tau + \alpha_{th,c}\,(1-\tau)$
- The solid charred enthalpy, h_c
- The solid Jomaa function, ξ_s

The *constantProperties* file from the **MaterialPropertiesDirectory** folder includes the number of solid phases N_p, the initial solid phase densities ($\rho_{s,i}$), the initial solid phase volume fractions ($\epsilon_{s,i}$), the virgin solid permeability ($\underline{\underline{\mathbf{K_{sv}}}}$), the charred solid permeability ($\underline{\underline{\mathbf{K_{sc}}}}$), the virgin Forchheimer coefficient ($\underline{\underline{\beta_{\mathbf{sv}}}}$), the charred Forchheimer coefficient ($\underline{\underline{\beta_{\mathbf{sc}}}}$), the virgin solid Young's modulus (E_{sv}), the charred solid Young's modulus (E_{sc}), the virgin solid Poisson's ratio (ν_{sv}), the charred solid Poisson's ratio (ν_{sc}), the virgin thermal expansion coefficient ($\alpha_{th,v}$), the charred thermal expansion coefficient ($\alpha_{th,c}$), the solid Jomaa function and the reference enthalpies of pyrolysis reaction j within phase i ($h_{p,ij}$) . The solid charred and virgin material properties in function of pressure and temperature are given by the *char* and *virgin* material files from the **MaterialPropertiesDirectory** folder. These files must have the 9 following columns: pressure p, temperature T, solid heat capacity $c_{p,s}$, solid enthalpy h_s, solid thermal conductivity in i,j,k directions $\underline{\underline{\mathbf{k}}}$, solid emissivity ε_s and solid absorptivity α_s.

When the *detailedSolidEnthalpy* option is activated, the pyrolysis energy flux is computed as follows

$$h_{S,s} = \frac{\rho_{sv}\left[h_{S,sv}(p,T) - h_{Sv}(p,T_0)\right] - \rho_{sc}\left[h_{S,sc}(p,T) - h_{S,sc}(p,T_0)\right]}{\rho_{sv}-\rho_{sc}} \tag{2.39}$$

$$h_{S,sv} = \int_{T_0}^{T} c_{p,sv}\ dT \qquad\qquad h_{S,sc} = \int_{T_0}^{T} c_{p,sc}\ dT \tag{2.40}$$

$$\dot{E}_p = -\sum_i^{N_p}\sum_j^{N_{sp}} \rho_{sv,i}\ \epsilon_{sv,i}\ \mathcal{F}_{ij}\ \partial_t\chi_{ij}\left(h_{p,ij}+h_{S,s}\right) \tag{2.41}$$

2.6.6 GradedPorous Type

The **GradedPorous** type updates the following solid properties (if they already exist in the mesh) based on the **Porous** type. The total advancement of pyrolysis (τ) is updated in the **PyrolysisModel**. Porosity ϕ is calculated from values from a given file at distance from a given patch.

- The solid thermal conductivity, $\underline{\underline{\mathbf{k_s}}} = \underline{\underline{\mathbf{k_{sv}}}}\ \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})} + \underline{\underline{\mathbf{k_{sc}}}} \left(1 - \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})}\right)$
- The solid heat capacity, $c_{p,s} = c_{p,sv}\ \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})} + c_{p,sc} \left(1 - \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})}\right)$
- The solid density, $\rho_s = \sum_i^{N_p} \rho_{s,i}\ \epsilon_{s,i}$
- The solid phase density, $\rho_{s,i} = \frac{\epsilon_{sv,i}}{\epsilon_{s,i}} \rho_{sv,i} \left(1 - \sum_j^{N_{sp}} \mathcal{F}_{ij}\chi_{ij}\right)$
- The solid permeability, $\underline{\underline{\mathbf{K_s}}} = \underline{\underline{\mathbf{K_{sc}}}} + (\underline{\underline{\mathbf{K_{sv}}}} - \underline{\underline{\mathbf{K_{sc}}}})\ \tau$
- The solid Forchheimer coefficient, $\underline{\underline{\beta_{\mathbf{s}}}} = \underline{\underline{\beta_{\mathbf{sc}}}} + (\underline{\underline{\beta_{\mathbf{sv}}}} - \underline{\underline{\beta_{\mathbf{sc}}}})\ \tau$
- The solid average enthalpy, $\bar{h} = \frac{\rho_{sv} h_{sv} - \rho_{sc} h_{sc}}{\rho_{sv} - \rho_{sc}}$
- The pyrolysis energy flux, $\dot{E}_p = -\bar{h}\pi_{tot}$
- The solid emissivity, $\varepsilon_s = \varepsilon_{sv}\ \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})} + \varepsilon_{sc} \left(1 - \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})}\right)$
- The solid absorptivity, $\alpha_s = \alpha_{sv}\ \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})} + \alpha_{sc} \left(1 - \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})}\right)$
- The solid Young's modulus $E_s = E_{sv}\ \tau + E_{sc}\,(1-\tau)$
- The solid Poisson's ration $\nu_s = \nu_{sv}\ \tau + \nu_{sc}\,(1-\tau)$
- The solid thermal expansion coefficient $\alpha_{th} = \alpha_{th,v}\ \tau + \alpha_{th,c}\,(1-\tau)$
- The solid charred enthalpy, h_c
- The solid Jomaa function, ξ_s

In addition to the data needed for the **Porous** type, the *constantProperties* file from the **MaterialPropertiesDirectory** folder includes the **gradingFilesEpsI** and the **gradingPatchNameEpsI** from which the grading distance is calculated.

2.6.7 Porous_const_k_UQ Type

The **Porous_const_k_UQ** type updates the solid properties (if they already exist in the mesh) by the **Porous** type except for the solid thermal conductivity ($\underline{\underline{\mathbf{k_s}}}$) that uses constant values from the *constantProperties* file located in the **MaterialPropertiesDirectory** folder.

- The solid thermal conductivity,
  $k_{s,xx} = k_{s,yy} = k_{s,zz} =$ **k_const_v** $\ \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})} +$ **k_const_c** $\left(1 - \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})}\right)$

2.6.8 Porous_factor Type

The **Porous_factor** type updates the solid properties (if they already exist in the mesh) based on the **Porous** type. The total advancement of pyrolysis (τ) is updated in the **PyrolysisModel**. The following properties are multiplied by a field factor provided in the *constantProperties* file from the **MaterialPropertiesDirectory** folder.

- The virgin solid heat capacity, $c_{p,sv}$
- The charred solid heat capacity $c_{p,sc}$
- The virgin solid enthalpy, h_{sv}
- The charred solid enthalpy, h_{sc}
- The virgin solid thermal conductivity in i, j, k directions $\underline{\underline{\mathbf{k_v}}}$
- The charred solid thermal conductivity in i, j, k directions $\underline{\underline{\mathbf{k_c}}}$
- The virgin solid emissivity ε_{sv}
- The charred solid emissivity ε_{sc}
- The virgin solid absorptivity α_{sv}
- The charred solid absorptivity α_{sc}

2.6.9 Porous_polynomial_k_UQ Type

The **Porous_polynomial_k_UQ** type updates the solid properties (if they already exist in the mesh) by the **Porous** type except for the solid thermal conductivity ($\underline{\underline{\mathbf{k_s}}}$) that uses constant values from the *constantProperties* file located in the **MaterialPropertiesDirectory** folder.

- Constant values,
  $k_{s,v0}$ = **kCondVirgin**
  $k_{s,v3}$ = **kRadVirgin**
  $k_{s,c0}$ = **kCondChar**
  $k_{s,c3}$ = **kRadChar**
- The solid thermal conductivity,
  $k_{s,xx} = k_{s,yy} = k_{s,zz} = \left(k_{s,v0} + k_{s,v3}\ T^3\right)\ \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})} + \left(k_{s,c0} + k_{s,c3}\ T^3\right)\left(1 - \tau\ \frac{\rho_{sv}}{max(\rho_s,\rho_{sc})}\right)$

2.7 Pyrolysis Model

Figure 2.8 shows the architecture of the Pyrolysis model. The *makePyrolysisModel.C* file creates the different Pyrolysis types based on the **simplePyrolysisModel** class. The Pyrolysis types are described below.

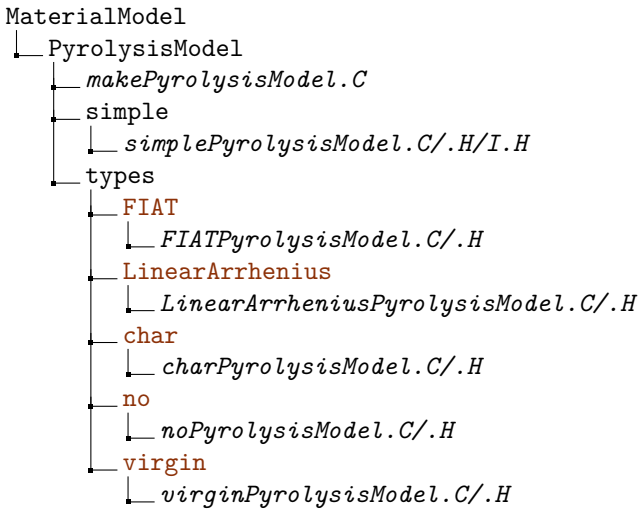

Figure 2.8. Pyrolysis model architecture

2.7.1 no Type

The **no** type is empty and serves as a place holder when the **PyrolysisModel** is not used.

2.7.2 virgin Type

The **virgin** type initializes the following fields of a virgin material using the *constantProperties* file in the **materialPropertiesDirectory** folder:

$$\pi_{tot} = 0 \qquad \tau = 1 \tag{2.42}$$

$$\rho_{sv,i} = \textbf{rhoI\_s[i]} \qquad \rho_{s,i} = \textbf{rho\_s[i]} \tag{2.43}$$

$$\epsilon_{sv,i} = \textbf{epsI\_s[i]} \qquad \epsilon_{s,i} = \textbf{eps\_s[i]} \tag{2.44}$$

$$\rho_s = \rho_{sv} = \sum_{i}^{N_p} \rho_{sv,i}\epsilon_{sv,i} \tag{2.45}$$

$$\rho_{sc} = \rho_{sv} - \sum_{i}^{N_p}\sum_{j}^{N_{sp}} \mathcal{F}_{ij}\rho_{sv,i}\epsilon_{sv,i} \tag{2.46}$$

2.7.3 char Type

The **char** type is based on the virgin material. The only difference for a charred material is the initialization of the advancement of pyrolysis ($\tau = 0$) and the solid density ($\rho_s = \rho_{s,c}$).

2.7.4 LinearArrhenius Type

The **LinearArrhenius** type updates the pyrolysis production rate by solving the pyrolysis equations (Eq. 2.47). The pyrolysis reactions are divided by phase i and sub-phase j. The total advancement of pyrolysis (Eq. 2.48), the total pyrolysis production rate (Eq. 2.49), the elemental/species pyrolysis production rates and the decomposed solid densities (Eq. 2.51) are then computed.

$$\partial_t \chi_{ij} = \mathcal{A}_{ij}\ T^{n_{ij}}\ exp\left(\frac{-\mathcal{E}_{ij}}{RT}\right)(1-\chi_{ij})^{m_{ij}} \tag{2.47}$$

$$\tau = \sum_{i}^{N_p}\sum_{j}^{N_{sp}} \frac{\epsilon_{sv,i}\ \rho_{sv,i}\ \mathcal{F}_{ij}}{\sum_i^{N_p}\sum_j^{N_{sp}} \epsilon_{sv,i}\ \rho_{sv,i}\ \mathcal{F}_{ij}}\ (1-\chi_{ij}) \tag{2.48}$$

$$\pi_{tot} = \sum_{i}^{N_p}\sum_{j}^{N_{sp}}\sum_{k}^{N_e} z_{ijk}\ \rho_{sv,i}\ \epsilon_{sv,i}\ \mathcal{F}_{ij}\ \partial_t \chi_{ij} \tag{2.49}$$

$$\pi_k = \sum_{i}^{N_p}\sum_{j}^{N_{sp}} z_{ijk}\ \rho_{sv,i}\ \epsilon_{sv,i}\ \mathcal{F}_{ij}\ \partial_t \chi_{ij} \tag{2.50}$$

$$\rho_{s,i} = \frac{\epsilon_{sv,i}}{\epsilon_{s,i}}\rho_{sv,i}\left(1-\sum_{j}^{N_{sp}}\mathcal{F}_{ij}\chi_{ij}\right) \tag{2.51}$$

Eq. 2.52 shows the update of the solid density in the **MaterialPropertiesModel** (**Porous** type).

$$\rho_s = \sum_{i}^{N_p}\rho_{s,i}\ \epsilon_{s,i} \tag{2.52}$$

2.7.5 FIAT Type

The **FIAT** type updates the pyrolysis production rate by solving the 3 pyrolysis equations (Eq. 2.53). The total pyrolysis production rate (Eq. 2.54) and the solid density (Eq. 2.55) are then computed.

$$\partial_t \rho_i = -A_i\ \rho_{v,i}\left(\frac{\rho_i-\rho_{c,i}}{\rho_{v,i}}\right)^{\psi_i} exp\left(\frac{-E_i}{T}\right) \qquad \text{i=0,1,2} \tag{2.53}$$

$$\pi_{tot} = -(1-\phi)\ \Gamma\ \partial_t\left(\rho_0-\rho_1\right) \tag{2.54}$$

$$\rho_s = (1-\phi)\ \Gamma\ (\rho_0+\rho_1) + (1-\Gamma)\ \rho_2 \tag{2.55}$$

2.8 MaterialChemistry Model

Figure 2.9 shows the architecture of the Material Chemistry model. The *makeMaterialChemistryModel.C* file creates the different Material Chemistry types based on the **simpleMaterialChemistryModel** class. The **FiniteRateChemistryModel** folder contains all files and directories for creating the different finite rate models based on *BasicFiniteRateChemistryModel.C*. The Material Chemistry types are described below.

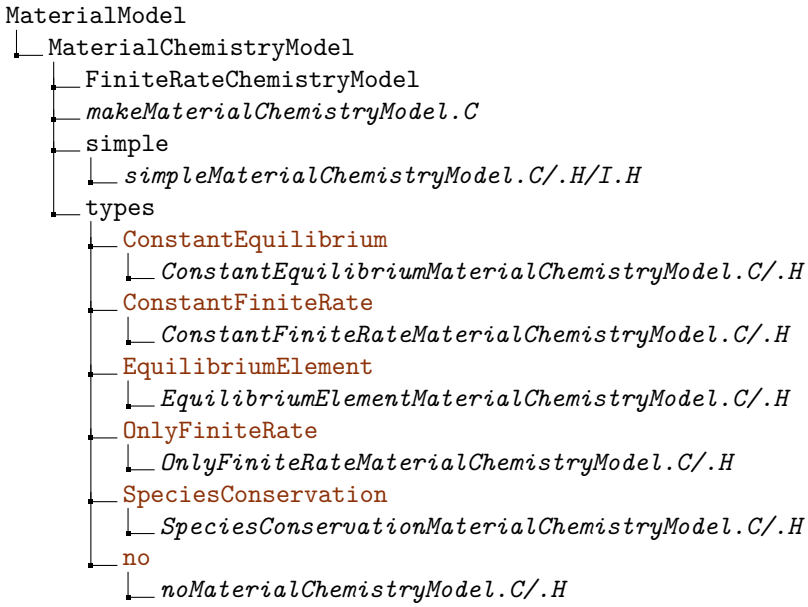

Figure 2.9. MaterialChemistry model architecture

2.8.1 no Type

The **no** type is empty and serves as a place holder when the **MaterialChemistryModel** is not used.

2.8.2 ConstantEquilibrium Type

The **ConstantEquilibrium** type creates the Mutation^{++}mixture using the **Equil** option with a constant elemental composition from the user.

2.8.3 ConstantFiniteRate Type

The **ConstantFiniteRate** type creates the Mutation^{++}mixture using the **ChemNonEq1T** option with a constant species composition from the user.

2.8.4 EquilibriumElement Type

The **EquilibriumElement** type updates the elemental mass fraction by solving the elemental mass conservation equation (Eq. 2.56). The transport and thermodynamic properties are updated in the **GasPropertiesModel**.

$$\epsilon_g \rho_g \, \partial_t z_k + \boldsymbol{\partial_x} \cdot (\epsilon_g \rho_g \boldsymbol{v_g} z_k) - \boldsymbol{\partial_x} \cdot \left(\frac{D_k \epsilon_g \rho_g}{\eta} \, \boldsymbol{\partial_x} z_k \right) = \pi_k \tag{2.56}$$

2.8.5 OnlyFiniteRate Type

The **OnlyFiniteRate** type creates the Mutation^{++}mixture using the **ChemNonEq1T** option with a constant species composition and updates the finite-rate chemistry in function of the species mass fraction, pressure and temperature: $\dot{\omega}_i(p, T, y_i)$.

2.8.6 SpeciesConservation Type

The **SpeciesConservation** type updates the species mass fraction by solving the species mass conservation equation (Eq. 2.57). The transport and thermodynamic properties are updated in the **GasProperties-Model**. This type also updates the heterogeneous reaction rate (Eq. 2.58) and the finite-rate chemistry in function of the species mass fraction, pressure and temperature: $\dot{\omega}_i(p, T, y_i)$

$$\epsilon_g \rho_g \ \partial_t y_i + \boldsymbol{\partial_x} \cdot (\epsilon_g \rho_g \boldsymbol{v_g} y_i) - \boldsymbol{\partial_x} \cdot \left(\frac{D_i \epsilon_g \rho_g}{\eta} \ \boldsymbol{\partial_x} y_i \right) = \epsilon_g \ \dot{\omega}_i + \pi_i \tag{2.57}$$

$$\Omega_h = -\dot{\omega}_{C(gr)} \tag{2.58}$$

2.9 Time Control Model

Figure 2.10 shows the architecture of the Time Control model. The *makeTimeControlModel.C* file creates the different Time Control types based on the **simpleTimeControlModel** class. The Time Control types are described below.

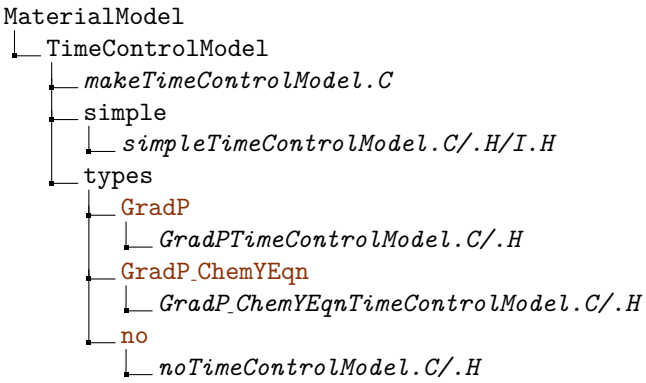

Figure 2.10. TimeControl model architecture

2.9.1 no Type

The **no** type is empty and serves as a place holder when the **TimeControlModel** is not used.

2.9.2 GradP Type

The "GradP" type updates the time step value Δt by computing the Courant number C_N (Eq. 2.60).

$$S_D = \frac{\text{mag}\left(\text{linearInterpolate}(\boldsymbol{v_g})\right)}{\Delta x \ REV_{length}} \tag{2.59}$$

$$C_N = \max(S_D) \ \Delta t \tag{2.60}$$

$$\Delta t_{fact} = \min\left[\min\left(\frac{C_{N,max}}{C_N + SMALL},\ 1 + 0.2\ \Delta t_{max}\right),\ 1.05\right] \tag{2.61}$$

$$\Delta t = \max\left[\min\left(\Delta t_{fact} \cdot \Delta t,\ \Delta t_{max}\right),\ \Delta t_{min}\right] \tag{2.62}$$

2.9.3 GradP_ChemYEqn Type

The "GradP" type updates the time step value Δt by computing the Courant number C_N (Eq. 2.64) and the chemical time step (Eq. 2.68).

$$S_D = \frac{\text{mag}\left(\text{linearInterpolate}(\boldsymbol{v_g})\right)}{\Delta x\ REV_{length}} \tag{2.63}$$

$$C_N = \max(S_D)\ \Delta t \tag{2.64}$$

$$\Delta t_{fact} = \min\left[\min\left(\frac{C_{N,max}}{C_N + SMALL},\ 1 + 0.2\ \Delta t_{max}\right),\ 1.05\right] \tag{2.65}$$

$$\Delta t_{pres} = \max\left[\min\left(\Delta t_{fact} \cdot \Delta t,\ \Delta t_{max}\right),\ \Delta t_{min}\right] \tag{2.66}$$

$$\boldsymbol{\partial_x}\ y_{max} = \max\left(\frac{y_i^{n-1} - y_i^n}{y_i^{n-1}}\right) \tag{2.67}$$

$$\Delta t_{chemY} = \begin{cases} 1/1.2\ \Delta t & \text{if } \boldsymbol{\partial_x}\ y_{max} > \text{dYtolMax} \\ 1.2\ \Delta t & \text{if } \boldsymbol{\partial_x}\ y_{max} < \text{dYtolMin} \end{cases} \tag{2.68}$$

$$\Delta t = \max\left[\min\left(\min\left[\Delta t_{chemY},\ \Delta t_{chem}\right],\ \Delta t_{pres}\right),\ \Delta t_{chem,min}\right] \tag{2.69}$$

2.10 Solid Mechanics Model

Figure 2.11 shows the architecture of the state-of-the-art material solid mechanics models implementation in PATO. The *makeSolidMechanicsModel.C* file creates the different Solid Mechanic types based on the **simpleSolidMechanicsModel** class. The Solid Mechanics types are described below. The **elasticSolidFoam** and **elasticOrthoSolidFoam** types are based on the work of Philip Cardiff from foam-extend 4.1.

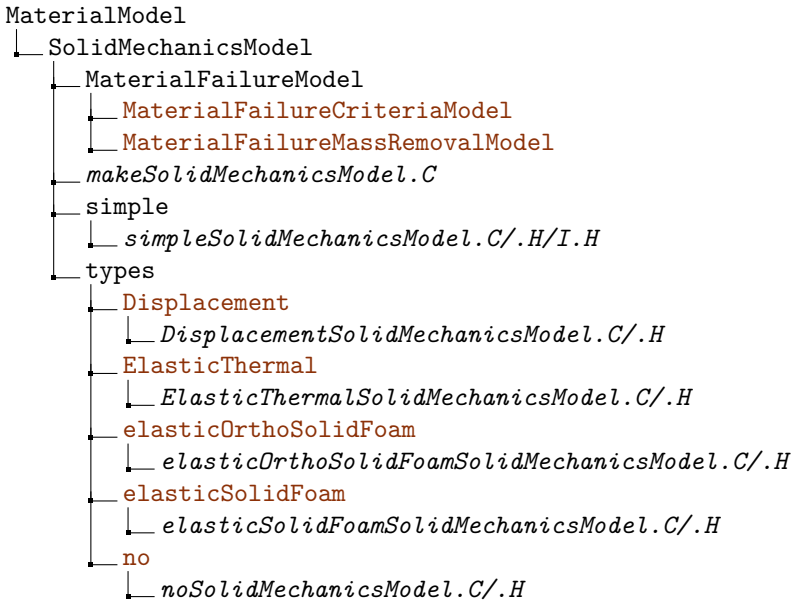

Figure 2.11. SolidMechanics model architecture

2.10.1 no Type

The **no** type is empty and serves as a place holder when the **SolidMechanicsModel** is not used.

2.10.2 ElasticThermal Type

The **ElasticThermal** type updates the solid displacement by solving a transient/steady-state segregated finite-volume formulation for small strain elastic isotropic thermal pyrolysable solid bodies. The displacement field D is solved for using a total Lagrangian approach, taking into account the material's expansion/shrinkage caused by the temperature field and the advancement of the pyrolysis reactions. The solver also provides the strain tensor field epsilon and stress tensor field sigma. The governing equation solved by this Solid Mechanics type is as follows:

$$\frac{\partial \rho \partial \boldsymbol{D}}{\partial t^2} = \partial_x \cdot \left((2\mu_s + \lambda_s) \underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}} \right) + \boldsymbol{div(\sigma)_{exp}} - \partial_x \left(3K\alpha_{th}(T - T_0) \right) + \partial_x \left(3K\xi \left(\tau_0 - \tau \right) \right) \tag{2.70}$$

where,

$$\boldsymbol{div(\sigma)_{exp}} = \partial_x \cdot \left(\underline{\underline{\sigma_{\mathbf{s}}}} - (2\mu_s + \lambda_s) \ \underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}} \right) \tag{2.71}$$

2.10.3 Displacement Type

The **Displacement** type updates the solid displacement similarly to the **ElasticThermal** model. Moreover, the mesh is moved at each time step according to the calculated displacement.

2.10.4 elasticSolidFoam Type

The **elasticSolidFoam** type updates the solid displacement by solving a transient/steady-state segregated finite-volume formulation for small strain elastic isotropic solid bodies allowing for general principal material orientations. The displacement field D is solved for using a total Lagrangian approach. The solver also provides the strain tensor field epsilon and stress tensor field sigma. The governing equation solved by this Solid Mechanics type is as follows:

$$\frac{\partial \rho \partial \boldsymbol{D}}{\partial t^2} = \partial_x \cdot \left((2\mu_s + \lambda_s)\, \underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}} \right) + \partial_{x_{exp}} \cdot \left(\underline{\underline{\sigma_{\mathbf{s}}}} - (2\mu_s + \lambda_s)\ \underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}} \right) \tag{2.72}$$

where,

$$\underline{\underline{\sigma}}_s = \mu_s\ \left(\underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}} + \underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}}^T \right) + \lambda_s\ \underline{\underline{\mathbf{I}}}\ tr(\underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}}) \tag{2.73}$$

2.10.5 elasticOrthoSolidFoam Type

The **elasticOrthoSolidFoam** type updates the solid displacement by solving a transient/steady-state segregated finite-volume formulation for small strain elastic orthotropic solid bodies [1]. The displacement field D is solved for using a total Lagrangian approach. The solver also provides the strain tensor field epsilon and stress tensor field sigma. The governing equation solved by this Solid Mechanics type is as follows:

$$\frac{\partial \rho \partial \boldsymbol{D}}{\partial t^2} = \partial_x \cdot \left(\boldsymbol{K}\ \underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}} \right) + \partial_{x_{exp}} \cdot \left(\underline{\underline{\sigma_{\mathbf{s}}}} - \boldsymbol{K}\ \underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}} \right) \tag{2.74}$$

where,

$$\underline{\underline{\sigma}}_s = \underline{\underline{\mathbf{C}}}_e : \frac{1}{2} \left(\underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}} + \underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}}^T \right) \tag{2.75}$$

2.10.6 MaterialFailureCriteria Model

Two types of stress-based failure theories are available in literature [2]: the maximum stress and the quadratic stress failure theories. In the maximum stress failure theory, the predicted by the material solver stresses of each mode are compared to the relevant material strength. Failure is predicted when the following criterion is satisfied:

$$\max \left(\frac{\sigma_{11}}{\sigma^{\mathrm{f}}_{\mathrm{t}1}}, \left| \frac{\sigma_{11}}{\sigma^{\mathrm{f}}_{\mathrm{c}1}} \right|, \frac{\sigma_{22}}{\sigma^{\mathrm{f}}_{\mathrm{t}2}}, \left| \frac{\sigma_{22}}{\sigma^{\mathrm{f}}_{\mathrm{c}2}} \right|, \frac{\sigma_{33}}{\sigma^{\mathrm{f}}_{\mathrm{t}3}}, \left| \frac{\sigma_{33}}{\sigma^{\mathrm{f}}_{\mathrm{t}3}} \right|, \left| \frac{\sigma_{12}}{\sigma^{\mathrm{f}}_{12}} \right|, \left| \frac{\sigma_{23}}{\sigma^{\mathrm{f}}_{23}} \right|, \left| \frac{\sigma_{31}}{\sigma^{\mathrm{f}}_{31}} \right| \right) > 1, \tag{2.76}$$

where $\sigma^{\mathrm{f}}_{\mathrm{t}i}$, $\sigma^{\mathrm{f}}_{\mathrm{c}i}$ are the maximum material strengths in tension (t) and compression (c) in the three directions (i = 1, 2, 3). Quadratic failure theories take into account the effective multi-axial loads on the material. Popular phenomenological failure theories for non-isotropic material are the Tsai-Hill and, the more general, Tsai-Wu theories [3, 4].

The maximum stress model was chosen as a first approach for determining failing regions. Mechanical erosion is caused by brittle failure of the fibers, for which maximum stress theories work well. Additionally, it allows for an easier identification of the responsible modes, particularly important for the study of anisotropic materials. The uncertainties related to the maximum stress model still need to be evaluated. An extensive review and comparison of different failure prediction models can be found in [5].

2.10.7 MaterialFailureMassRemoval Model

The failure criteria model, described above, gives for each computational cell the information of whether a single cell is failing under the local stress field. The MaterialFailureMassRemoval model connects this information to the actual surface mass removal (e.g. spallation), in terms of the recession velocity:

$$u_{\mathrm{r}} = \frac{l_{\mathrm{f}}}{\Delta t}, \tag{2.77}$$

where l_f is the depth of the failing region and the face of the local, failing boundary. The "failure distance" is determined at each time step, with an algorithm combining a geometrically increasing inward search to find the first non-failing cell, followed by a bisection algorithm to determine its exact distance from the surface. The mesh motion algorithm of PATO, is afterwards, responsible for applying the displacement. In-depth, disconnected patches of failing material are not accounted for here, since they involve mechanisms and phenomena, such as cracking, beyond the scope of this model.

The mass, m_f, removed from the material due to mechanical erosion during Δt is calculated for each face of area A as:

$$m_\mathrm{f} = \rho_s \; u_\mathrm{r} \; l_\mathrm{f} \; A \; \Delta t, \tag{2.78}$$

where ρ_s is the solid density.

2.11 Boundary Conditions

Figure 2.12 shows the architecture of the state-of-the-art material Boundary Conditions (BC) implemented in PATO. The material BC are explained below.

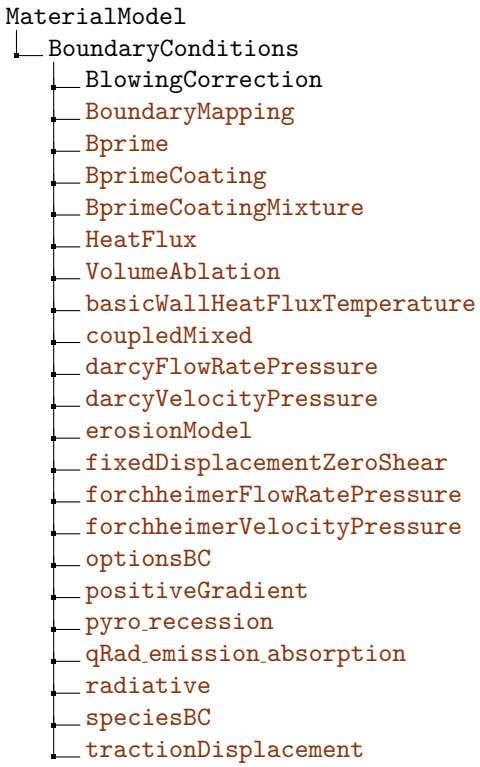

Figure 2.12. Boundary Conditions architecture

2.11.1 BoundaryMapping BC

Figure 2.13 shows the architecture of the Boundary Mapping boundary conditions.

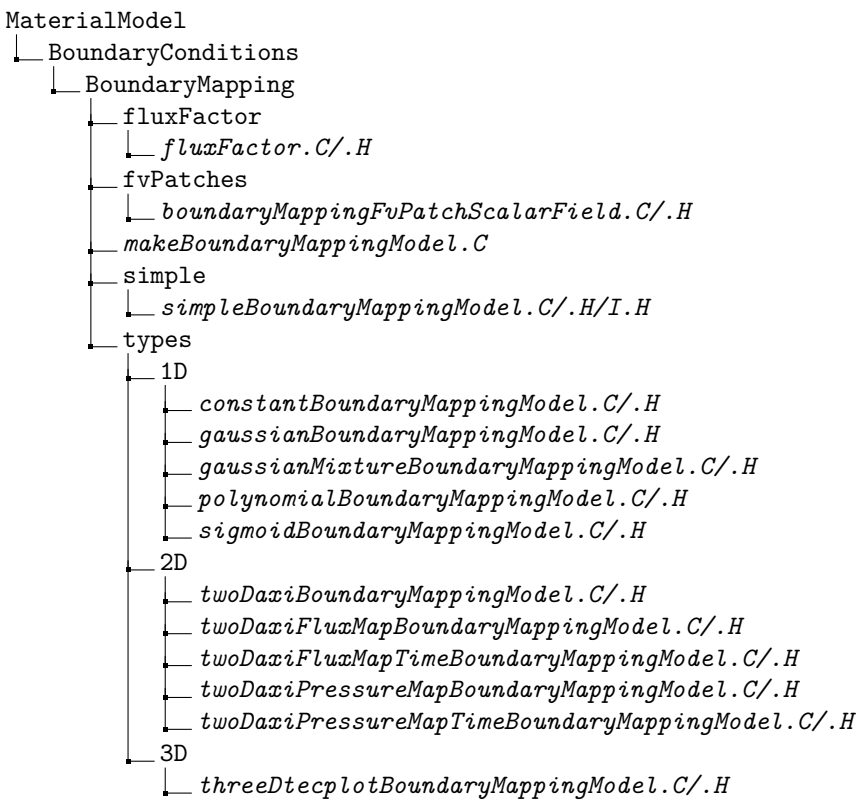

Figure 2.13. Boundary Mapping BC architecture

The **boundaryMappingFvPatchScalarField** class creates the Boundary Mapping patches. The *makeBoundaryMappingModel.C* creates the Boundary Mapping boundary conditions based on the **simpleBoundaryMappingModel** class. The **constantBoundaryMappingModel** class updates the boundary conditions of a fields list using user tables. The **gaussianBoundaryMappingModel** class updates the boundary conditions of a field's list using Gaussian equations (Eq. 2.79).

$$f(t) = A\ exp\left(\frac{-(t-B)^2}{C}\right) \tag{2.79}$$

The **polynomialBoundaryMappingModel** class updates the boundary conditions of a field's list using polynomial equations (Eq. 2.80).

$$f(t) = \sum_{i}^{N_p} A_i\ t^i \tag{2.80}$$

The **FluxFactor** class updates the 2D axi-symmetric flux mapping using the distance from the user center point to the face centers along the user normal. The **twoDaxiBoundaryMappingModel** class updates the boundary conditions of a fields list using the radius and the axial distance. The **twoDaxiFluxMapBoundaryMappingModel** class updates the boundary conditions of the flux mapping field using the **FluxFactor** class. The **twoDaxiPressureMapBoundaryMappingModel** class updates the boundary conditions of the pressure field using the **FluxFactor** class. The **threeDtecplotBoundaryMappingModel** class reads a user Tecplot file and updates the boundary conditions of a fields list with the Arbitrary Mesh Interface (AMI) of OpenFOAM which enables interfacing adjacent, disconnected mesh domains using Galerkin projection.

2.11.2 Bprime BC

Surface mass balance

The char ablation rate $\dot{m}_{ca}$ and the wall enthalpy h_w are computed with a thermochemical model in equilibrium at the wall. Figure 2.14 provides a schematic for the surface mass balance model based on the steady state element conservation in a control volume close to the wall. The equilibrium chemistry in the control volume is assumed to be quasi-steady in order to decouple the material response and the boundary layer. The time variation of p_w, T_w, $\dot{m}_{pg}$ and $\dot{m}_{ca}$ is neglected. Mechanical erosion is not considered here.

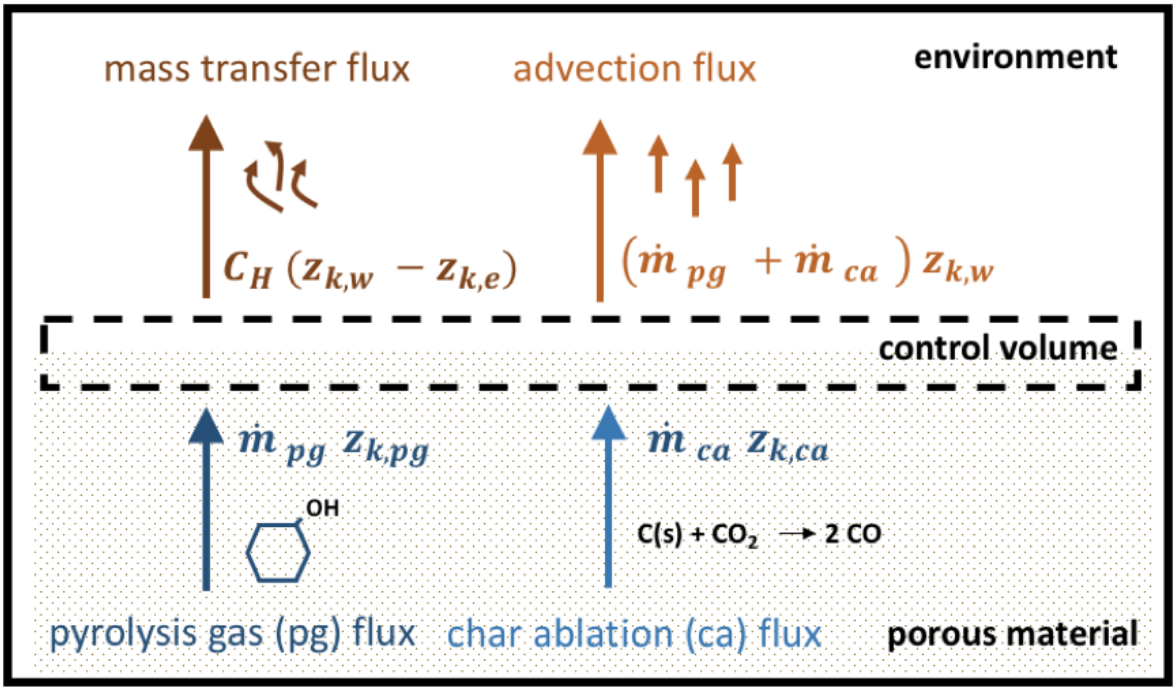

Figure 2.14. Surface mass balance at the heatshield front surface.

Under the assumption that Prandtl and Lewis numbers are equal to unity and the diffusion coefficients are identical between elements, the conservation of mass fraction of an element k in the control volume may be written as

$$C'_H\,(z_{k,w} - z_{k,e}) + (\dot{m}_{pg} + \dot{m}_{ca})\,z_{k,w} = \dot{m}_{pg}\;z_{k,pg} + \dot{m}_{ca}\;z_{k,ca} \tag{2.81}$$

$$(z_{k,w} - z_{k,e}) + \left(\dot{B}'_{pg} + \dot{B}'_{ca}\right) z_{k,w} = \dot{B}'_{pg}\;z_{k,pg} + \dot{B}'_{ca}\;z_{k,ca} \tag{2.82}$$

The formation of the species S_i from the elements A_k is formulated as follows

$$S_i \rightleftharpoons \sum_{k\in[1,N_e]} \nu_{i,k} A_k \tag{2.83}$$

Table 2.1 shows an example of the formation of the CO_2 species from the C and O elements.

Table 2.1. Example of CO_2 formation in Eq. 2.83.

S_1	$\nu_{1,1}$	$\nu_{1,2}$	E_1	E_2
CO_2	1	2	C	O

If the species are assumed perfect gas, then the chemical equilibrium is given by

$$\frac{x_i}{\prod_{k\in[1,N_e]}(x_k)^{\nu_{i,k}}} = K_i(T) \;\Leftrightarrow\; ln\,(x_i) - \sum_{k\in[1,N_e]} \nu_{i,k} ln\,(x_k) - ln\,[K_i(T)] = 0 \tag{2.84}$$

$$\sum_{i\in[1,N_s]} x_i = 1 \qquad \sum_{k\in[1,N_e]} x_k = 1 \tag{2.85}$$

The surface mass balance model computes $\dot{m}_{ca}$ and h_w from Eq. 2.81, 2.84 and 2.85.

The pyrolysis gas production rate at the heatshield front surface $\dot{m}_{pg}$ by integrating the pyrolysis, mass and transport equations. p_w and C_H are given by the aerothermal environment. T_w and C'_H are computed in the surface energy balance described below.

The material mass loss rate leads to a surface ablation velocity given by

$$\mathbf{v}_{ca} = \frac{\dot{m}_{ca}}{\rho_{sw}}\,\mathbf{n} \tag{2.86}$$

and is applied as a mesh motion in PATO.

Surface energy balance

The wall temperature T_w is computed with a surface energy balance model, as illustrated in Fig. 2.15. Heating and cooling energy fluxes from the environment and the porous material are shown. The state-of-the-art surface energy balance at the wall is given by

$$q^{out}_{cond} = C'_H\,(h_e - h_w) + \dot{m}_{pg}h_{pg} + \dot{m}_{ca}h_{ca} - (\dot{m}_{pg} + \dot{m}_{ca})\,h_w + q^{in}_{rad} - \varepsilon_w\sigma\left(T_w^4 - T_\infty^4\right) \tag{2.87}$$

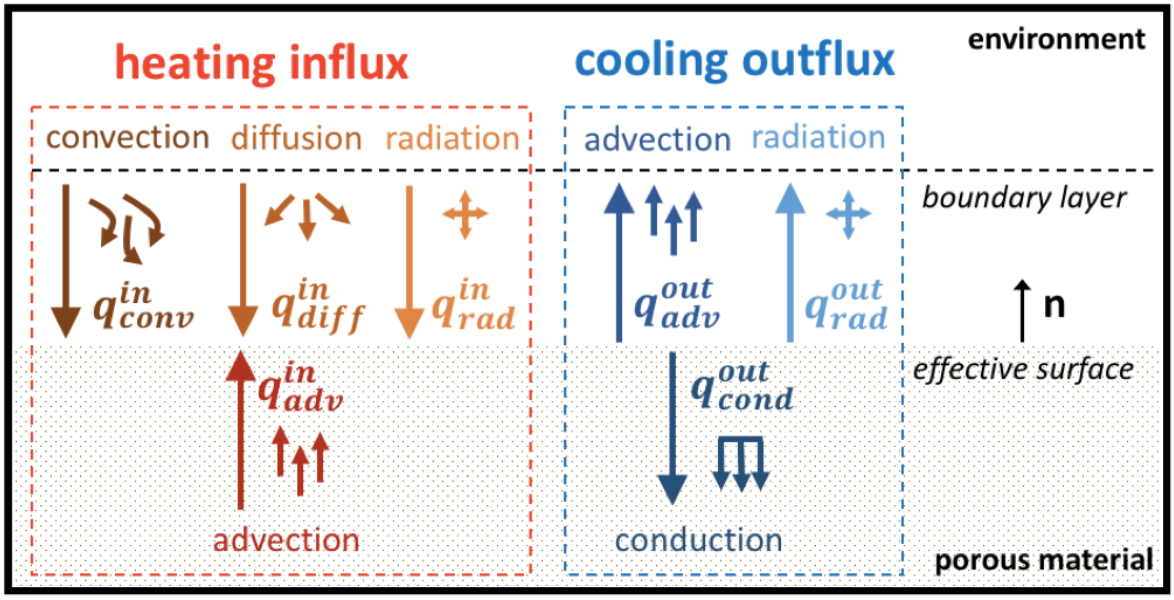

Figure 2.15. Surface energy balance at the heatshield front surface.

The equality between inward and outward fluxes yields

$$q^{in}_{conv} + q^{in}_{diff} + q^{in}_{rad} + q^{in}_{adv} = q^{out}_{cond} + q^{out}_{rad} + q^{out}_{adv} \tag{2.88}$$

The different terms of Eq. 2.88 are formulated here. The convective heat flux, under the assumption of a frozen boundary layer and a non-catalytic wall is

$$q^{in}_{conv} = C_H \left(h_e - h_{ew}\right) \tag{2.89}$$

h_{ew} is the enthalpy computed at the wall temperature, with the boundary layer edge gaseous species composition.

$$h_{ew} = \sum_{i\in[1,N_s]} y_{i,e} h_i(T_w) \tag{2.90}$$

The energy carried by diffusion of the gaseous species is given by

$$q^{in}_{diff} = C_M \left(h_{ew} - h_w\right) \tag{2.91}$$

h_w is the enthalpy at the wall temperature, with the porous material gaseous species composition.

$$h_w = \sum_{i\in[1,N_s]} y_{i,w} h_i(T_w) \tag{2.92}$$

The advective energy transport produced by the pyrolysis and the char ablation read respectively

$$q^{in}_{adv} = \dot{m}_{pg} h_{pg} + \dot{m}_{ca} h_{ca} \tag{2.93}$$

$$q^{out}_{adv} = \left(\dot{m}_{pg} + \dot{m}_{ca}\right) h_w \tag{2.94}$$

The radiative heating from the plasma is given by,

$$q^{in}_{rad} = \alpha_w \; q_{pla} + \varepsilon_w \; \sigma \; T^4_\infty \tag{2.95}$$

while the re-radiative cooling by surface emission reads

$$q^{out}_{rad} = \varepsilon_w \; \sigma \; T^4_w \tag{2.96}$$

under the assumption that the surface behaves as a gray body.

The effective heat conduction in the porous material is given by

$$q^{out}_{cond} = -\left(\overline{\overline{\mathbf{k}}}_w \cdot \frac{\partial T_w}{\partial \mathbf{n}}\right) \cdot \mathbf{n} \tag{2.97}$$

Eq. 2.88, using the different energy contributions explained above, gives

$$\begin{aligned} &-\left(\overline{\overline{\mathbf{k}}}_w \cdot \frac{\partial T_w}{\partial \mathbf{n}}\right) \cdot \mathbf{n} = C_H \left(h_e - h_{ew}\right) + C_M \left(h_{ew} - h_w\right) \\ &+ \dot{m}_{pg} \left(h_{pg} - h_w\right) + \dot{m}_{ca} \left(h_{ca} - h_w\right) - \varepsilon_w \sigma \left(T^4_w - T^4_\infty\right) + \alpha_w q_{pla} \end{aligned} \tag{2.98}$$

Assuming equal Prandtl and Lewis number and equal diffusion coefficients for all elements, Eq. 2.98 becomes

$$\begin{aligned} &-\left(\overline{\overline{\mathbf{k}}}_w \cdot \frac{\partial T_w}{\partial \mathbf{n}}\right) \cdot \mathbf{n} = C'_H \left(h_e - h_w\right) \\ &+ \dot{m}_{pg} \left(h_{pg} - h_w\right) + \dot{m}_{ca} \left(h_{ca} - h_w\right) - \varepsilon_w \sigma \left(T^4_w - T^4_\infty\right) + \alpha_w q_{pla} \end{aligned} \tag{2.99}$$

C_H is corrected to account only for the blockage induced by the pyrolysis and ablation gas blowing. The blowing correction (Eq. 2.100) is updated in the **BlowingCorrection** class where other film coefficient corrections, such as roughness and hot wall effects, are not considered.

$$C'_H = C_H \frac{ln\left[1 + 2\lambda\left(B'_{pg} + B'_{ca}\right)\right]}{2\lambda\left(B'_{pg} + B'_{ca}\right)} \tag{2.100a}$$

$$B'_{pg} = \frac{\dot{m}_{pg}}{C_M} \tag{2.100b}$$

$$B'_{ca} = \frac{\dot{m}_{ca}}{C_M} \tag{2.100c}$$

When the **chemistryOn** flag is turned on, the surface energy balance computes T_w from Eq. 2.99 using the following inputs: C'_H, h_e, h_w, $\dot{m}_{pg}$, h_{pg}, $\dot{m}_{ca}$, h_{ca}, ε_w, T_∞, α_w and q_{pla}. C'_H is computed with Eq. 2.100. C_H and h_e are given by the aerothermal environment. h_w and $\dot{m}_{ca}$ come from the surface mass balance. $\dot{m}_{pg}$ is computed by integrating the pyrolysis, mass and transport equations. h_{pg} and h_{ca} are computed using the **GasPropertiesModel** model.

When the **chemistryOn** flag is turned off, the wall temperature is updated using the following equation:

$$-\left(\bar{\bar{\mathbf{k}}}_w \cdot \frac{\partial T_w}{\partial \mathbf{n}}\right) \cdot \mathbf{n} = h_{conv}\left(T_e - T_w\right) - \varepsilon_w \sigma \left(T_w^4 - T_\infty^4\right) + \alpha_w q_{pla} \tag{2.101}$$

2.11.3 optionsBC

The following **optionsBC** types are activated with a Switch in the **Bprime** BC input file:

- **factorOptionModel** multiplies the fields from the first tuple elements in **factorList** by a factor equals to the second tuple elements.
- **timeFactorOptionModel** updates the fields from the first tuple elements in **timeFactorList** using a linear interpolation of the data from **timeFactorFile**. The second tuple elements give the column indexes of the corresponding fields in **timeFactorFile**.

Listing 2.1 shows an example of the **optionBC** types. In this case, **factorOptionModel** multiplies by 2 the pressure boundary field of the top BC. **timeFactorOptionModel** updates the char blowing rate at the top BC using a linear interpolation in time of the first column values from *constant/timeFactor*.

Listing 2.1. "origin.0/porousMat/Ta" file

```
boundaryField {
    top {
        type Bprime;
        factorOption yes;
        factorList (("p" 2));
        timeFactorOption yes;
        timeFactorFile "constant/timeFactor";
        timeFactorList (("BprimeCw" 1));
        BprimeFile "data/Materials/TACOT/BprimeTable";
        mappingType "constant";
        mappingFields ((p "1") (rhoeUeCH "2") (h_r "3"));
        Tbackground 187;
        qRad 0;
        lambda 0.5;
        chemistryOn 1;
        value uniform 300;
    }
}
```

2.11.4 BprimeCoating BC

The **BprimeCoating** BC extends the Bprime boundary condition for coated surfaces. While the underlying physics are identical with the ones of the previous section, the surface mass and energy balances are initially closed with the thermochemical parameters of the single component coating. Ablation of the material reduces the thickness of the coating till its complete removal, without deforming the mesh, an assumption justified by the fact that coatings are usually less than a millimeter in thickness. After the coating's removal the substrate material is exposed and the boundary condition behaves like in the uncoated case.

2.11.5 BprimeCoatingMixture BC

The **BprimeCoatingMixture** BC is an extension of the **BprimeCoating** BC including multiple species coatings, such as Si-O-C (silicon oxycarbide) surfaces. The coating's elemental composition needs to be specified, with the B' tables calculated at run-time.

2.11.6 HeatFlux BC

The **HeatFluxBoundaryConditions** class applies a heat flux (q_{conv}^{CFD}) directly to the boundary. This boundary condition does not currently allow for recession. The surface energy balance is given by

$$q_{cond}^{out} = q_{conv}^{CFD} + \alpha_w q_{pla} - \varepsilon_w \sigma \left(T_w^4 - T_\infty^4\right) \tag{2.102}$$

2.11.7 erosionModel BC

The **erosionModelBoundaryConditions** class updates the cellMotionU field, which directly impacts the mesh motion. The **physicsBasedErosionModel** flag, chosen by the user, selects the type of erosion model.

- type=1: the ablation rate is based on the minimum fiber radius failure and the gradient of fiber radius.

$$\boldsymbol{v}_{mesh} = \frac{r_{f,fail} - r_{f,w}}{\boldsymbol{\partial_x} r_{f,int} \cdot \boldsymbol{n}\ \Delta t}\ \boldsymbol{n} \tag{2.103}$$

- type=2: the ablation rate is based on the solid density gradient.

$$\boldsymbol{v}_{mesh} = \frac{50 - \rho_{s,w}}{\boldsymbol{\partial_x} \rho_{s,int} \cdot \boldsymbol{n}\ \Delta t}\ \boldsymbol{n} \tag{2.104}$$

- type=3: the ablation rate is based on the cell-by-cell removing.

$$\boldsymbol{v}_{mesh} = 0.98\ \boldsymbol{v}_{mesh} + 0.02\ \frac{\Delta x}{\Delta t}\ \boldsymbol{n} \tag{2.105}$$

- type=4: the ablation rate is based on the solid density.

$$\boldsymbol{v}_{mesh} = \max\left(1.05\ \boldsymbol{v}_{mesh} \cdot \boldsymbol{n},\ \frac{10^{-7}}{\rho_{s,w}}\right)\ \boldsymbol{n} \tag{2.106}$$

- type=5: the ablation rate is based on the heterogeneous rate.

$$\boldsymbol{v}_{mesh} = 0.95\ \boldsymbol{v}_{mesh} + 0.05\ \frac{\Omega_{H,int}}{\rho_{s,int}}\ \boldsymbol{n} \tag{2.107}$$

2.11.8 coupledMixed BC

The **coupledMixedFvPatchScalarField** class creates the boundary conditions patch and updates the field f_1 boundary conditions using the harmonic averaging of the neighbor field f_2 (Eq. 2.108).

$$f_{1/2} = v_f\ f_1 + (1 - v_f)\ f_2 \tag{2.108}$$

$$v_f = \frac{\frac{k_2}{\Delta x_2}}{\frac{k_1}{\Delta x_1} + \frac{k_2}{\Delta x_2}} \tag{2.109}$$

2.11.9 positiveGradient BC

The **positiveGradientFvPatchScalarField** class creates the boundary conditions patch and updates the field f_w boundary conditions using the cell center value f_1 and the positive gradient (Eq. 2.110).

$$\boldsymbol{\partial_x} f_w = \begin{cases} \frac{f_w - f_1}{\Delta x_1} & \text{if } f_w > f_1 \\ 0 & \text{if } f_w \leq f_1 \end{cases} \tag{2.110}$$

2.11.10 pyro_recession BC

pyro_recession BC updates the mesh motion velocity $\boldsymbol{v}_{mesh}$ using the recession rate r_r.

$$\boldsymbol{v}_{mesh} = r_r \ \boldsymbol{n} \tag{2.111}$$

2.11.11 radiative BC

The **radiativeBoundaryConditions** class updates the temperature BC. The wall temperature T_w is computed with a surface energy balance model (Eq. 2.112).

$$q^{in}_{conv} + q^{in}_{diff} + q^{in}_{rad} = q^{out}_{cond} + q^{out}_{rad} \tag{2.112}$$

The convective heat flux, under the assumption of a frozen boundary layer and a non-catalytic wall is

$$q^{in}_{conv} = C_H \left(h_e - h_{ew}\right) \tag{2.113}$$

h_{ew} is the enthalpy computed at the wall temperature, with the boundary layer edge gaseous species composition.

$$h_{ew} = \sum_{i \in [1, N_s]} y_{i,e} h_i(T_w) \tag{2.114}$$

The energy carried by diffusion of the gaseous species is given by

$$q^{in}_{diff} = C_M \left(h_{ew} - h_w\right) \tag{2.115}$$

h_w is the enthalpy at the wall temperature, with the porous material gaseous species composition.

$$h_w = \sum_{i \in [1, N_s]} y_{i,w} h_i(T_w) \tag{2.116}$$

The radiative heating from the plasma is given by,

$$q^{in}_{rad} = \alpha_w \ q_{pla} + \varepsilon_w \ \sigma \ T^4_\infty \tag{2.117}$$

while the re-radiative cooling by surface emission reads

$$q^{out}_{rad} = \varepsilon_w \ \sigma \ T^4_w \tag{2.118}$$

under the assumption that the surface behaves as a gray body. The effective heat conduction in the porous material is given by

$$q^{out}_{cond} = -\left(\overline{\overline{\mathbf{k}}}_w \cdot \frac{\partial T_w}{\partial \mathbf{n}}\right) \cdot \mathbf{n} \tag{2.119}$$

Eq. 2.112, using the different energy contributions explained above, gives

$$-\left(\overline{\overline{\mathbf{k}}}_w \cdot \frac{\partial T_w}{\partial \mathbf{n}}\right) \cdot \mathbf{n} = C_H \left(h_e - h_{ew}\right) + C_M \left(h_{ew} - h_w\right) - \varepsilon_w \sigma \left(T^4_w - T^4_\infty\right) + \alpha_w q_{pla} \tag{2.120}$$

Assuming equal Prandtl and Lewis number and equal diffusion coefficients for all elements, Eq. 2.120 becomes

$$-\left(\overline{\overline{\mathbf{k}}}_w \cdot \frac{\partial T_w}{\partial \mathbf{n}}\right) \cdot \mathbf{n} = C'_H \left(h_e - h_w\right) - \varepsilon_w \sigma \left(T^4_w - T^4_\infty\right) + \alpha_w q_{pla} \tag{2.121}$$

C_H is normally corrected to account for the blockage induced by the pyrolysis and ablation gas blowing. This *radiativeBoundaryConditions* class assumes there is no pyrolysis and ablation gas blowing.

$$C'_H = C_H \tag{2.122}$$

The surface energy balance computes T_w from Eq. 2.121 using the following user inputs: C_H, h_e, h_w, ε_w, T_∞, α_w and q_{pla}.

2.11.12 speciesBC BC

The **speciesBCBoundaryConditions** class updates the C'_H boundary conditions and reads the boundary layer edge species composition.

2.11.13 VolumeAblation BC

The **VolumeAblationBoundaryConditions** class updates the temperature boundary conditions and the mesh motion. The wall temperature T_w is computed with a surface energy balance model (Eq. 2.112).

$$q^{in}_{conv} + q^{in}_{rad} = q^{out}_{cond} + q^{out}_{rad} \tag{2.123}$$

The convective heat flux, under the assumption of a frozen boundary layer and a non-catalytic wall is

$$q^{in}_{conv} = C_H \left(h_e - h_{ew}\right) \tag{2.124}$$

h_{ew} is the enthalpy computed at the wall temperature, with the boundary layer edge gaseous species composition.

$$h_{ew} = \sum_{i \in [1, N_s]} y_{i,e} h_i(T_w) \tag{2.125}$$

The radiative heating from the plasma is given by,

$$q^{in}_{rad} = \alpha_w \; q_{pla} + \varepsilon_w \; \sigma \; T^4_\infty \tag{2.126}$$

while the re-radiative cooling by surface emission reads

$$q^{out}_{rad} = \varepsilon_w \; \sigma \; T^4_w \tag{2.127}$$

under the assumption that the surface behaves as a gray body. The effective heat conduction in the porous material is given by

$$q^{out}_{cond} = -\left(\overline{\overline{\mathbf{k}}}_w \cdot \frac{\partial T_w}{\partial \mathbf{n}} \right) \cdot \mathbf{n} \tag{2.128}$$

Eq. 2.123, using the different energy contributions explained above, gives

$$-\left(\overline{\overline{\mathbf{k}}}_w \cdot \frac{\partial T_w}{\partial \mathbf{n}} \right) \cdot \mathbf{n} = C_H \left(h_e - h_{ew}\right) - \varepsilon_w \sigma \left(T^4_w - T^4_\infty\right) + \alpha_w q_{pla} \tag{2.129}$$

The surface energy balance computes T_w from Eq. 2.129 using the following user inputs: C_H, h_e, $y_{i,e}$, ε_w, T_∞, α_w and q_{pla}.

Besides, the **VolumeAblationBoundaryConditions** class updates the cellMotionU boundary field, which directly impacts the mesh motion. The physicsBasedErosionModel user flag selects the type of motion model.

- type 1: the ablation rate is based on the minimum fiber radius failure and the gradient of fiber radius.

$$\boldsymbol{v}_{mesh} = \frac{r_{f,fail} - r_{f,w}}{\boldsymbol{\partial_x} r_{f,int} \cdot \boldsymbol{n} \; \Delta t} \; \boldsymbol{n} \tag{2.130}$$

- type 2: the ablation rate is based on the solid density gradient.

$$\boldsymbol{v}_{mesh} = \frac{50 - \rho_{s,w}}{\boldsymbol{\partial_x} \rho_{s,int} \cdot \boldsymbol{n} \; \Delta t} \; \boldsymbol{n} \tag{2.131}$$

- type 3: the ablation rate is based on the cell-by-cell removing.

$$\boldsymbol{v}_{mesh} = 0.98 \; \boldsymbol{v}_{mesh} + 0.02 \; \frac{\Delta x}{\Delta t} \; \boldsymbol{n} \tag{2.132}$$

- type 4: the ablation rate is based on the solid density.

$$\boldsymbol{v}_{mesh} = \max\left(1.05\ \boldsymbol{v}_{mesh}\cdot\boldsymbol{n},\ \frac{10^{-7}}{\rho_{s,w}}\right)\ \boldsymbol{n} \tag{2.133}$$

- type 5: the ablation rate is based on the heterogeneous rate.

$$\boldsymbol{v}_{mesh} = 0.95\ \boldsymbol{v}_{mesh} + 0.05\ \frac{\Omega_{H,int}}{\rho_{s,int}}\ \boldsymbol{n} \tag{2.134}$$

2.11.14 basicWallHeatFluxTemperature BC

The **basicWallHeatFluxTemperature** BC applies a heat flux condition to temperature on a wall. It is based on the OpenFOAM **externalWallHeatFlux** BC. It operates in one of three modes:

- Fixed power: supply Power Q in W.

$$T_w = T_{cell} \tag{2.135a}$$

$$\partial_w T = \frac{\frac{Q}{S_f} + q_r}{k_s} \tag{2.135b}$$

- Heat flux: supply heat flux q in $\mathrm{W}\cdot\mathrm{m}^{-2}$.

$$T_w = T_{cell} \tag{2.136a}$$

$$\partial_w T = \frac{q + q_r}{k_s} \tag{2.136b}$$

- Fixed heat transfer coefficient: supply heat transfer coefficient h_{th} in $\mathrm{W}\cdot\mathrm{m}^{-2}\cdot\mathrm{K}^{-1}$ and ambient temperature T_∞ in K.

$$T_w = \frac{h - \frac{q_r}{T_{cell}}}{h - \frac{q_r}{T_{cell}} + k_s\ \frac{1}{\delta}}\ \frac{\epsilon\ \sigma\ T^4}{h - \frac{q_r}{T_{cell}}} + \left(1 - \frac{h - \frac{q_r}{T_{cell}}}{h - \frac{q_r}{T_{cell}} + k_s\ \frac{1}{\delta}}\right)\ T_{cell} \qquad \text{if } q_r < 0 \tag{2.137a}$$

$$T_w = \frac{h}{h + k_s\ \frac{1}{\delta}}\ \frac{\epsilon\ \sigma\ T^4 + q_r}{h} + \left(1 - \frac{h}{h + k_s\ \frac{1}{\delta}}\right)\ T_{cell} \qquad \text{if } q_r \geq 0 \tag{2.137b}$$

2.11.15 darcyVelocityPressure BC

The **darcyVelocityPressure** BC provides a Darcy's velocity condition for pressure. It is derived from a **fixedGradient** condition, whereas pressure is calculated from the projection of the velocity vector and permeability tensor on the direction of the flow.

$$\partial_w p = -\mu_g * \frac{\epsilon \boldsymbol{v_g}\cdot\boldsymbol{n}}{\underline{\underline{\mathbf{K_s}}}\cdot\boldsymbol{n}\cdot\boldsymbol{n}} \tag{2.138}$$

2.11.16 darcyFlowRatePressure BC

The **darcyFlowRatePressure** BC provides a Darcy flow calculated for a uniform velocity field normal to the patch (Eq. 2.138) adjusted to match the specified flow rate. For a mass-based flux, the flow rate should be provided in $\mathrm{kg}\cdot\mathrm{s}^{-1}$. For a volumetric-based flux, the flow rate is in $\mathrm{m}^3\cdot s^{-1}$.

2.11.17 forchheimerVelocityPressure BC

The **forchheimerVelocityPressure** BC provides a compressible Darcy-Forchheimer's velocity condition for pressure. It is derived from a **fixedGradient** condition, whereas pressure is calculated from the projection of the velocity vector and permeability tensor on the direction of the flow.

$$\partial_w p = -\left(\frac{\mu_g}{\underline{\underline{\mathbf{K}}} \cdot \boldsymbol{n} \cdot \boldsymbol{n}} + \rho_g \underline{\underline{\beta}} \cdot \boldsymbol{n} \cdot \boldsymbol{n} \; |\epsilon \boldsymbol{v_g}|\right) (\epsilon \boldsymbol{v_g} \cdot \boldsymbol{n}) + \rho_g \; g_f \tag{2.139}$$

2.11.18 forchheimerFlowRatePressure BC

The **forchheimerFlowRatePressure** BC provides a compressible Darcy-Forchheimer's flow calculated for a uniform velocity field normal to the patch (Eq. 2.139) adjusted to match the specified flow rate. For a mass-based flux, the flow rate should be provided in $\mathrm{kg} \cdot \mathrm{s}^{-1}$. For a volumetric-based flux, the flow rate is in $\mathrm{m}^3 \cdot s^{-1}$.

2.11.19 tractionDisplacement BC

The **tractionDisplacement** BC provides a fixed traction BC for the standard linear elastic, fixed coefficient displacement equation with thermal and pyrolysis contribution.

$$\begin{aligned}\partial_w \boldsymbol{D} = \frac{1}{2\mu_s + \lambda_s} \Big(\boldsymbol{F_t} + F_p \; \boldsymbol{n} - \Big(\boldsymbol{n} \cdot \Big(\mu_s \underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}}^T - (\mu_s + \lambda_s) \; \underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}}\Big)\Big) \\ -\boldsymbol{n} \; \lambda_s \; tr(\underline{\underline{\partial_{\mathbf{x}} \mathbf{D}}}) + \boldsymbol{n} \; 3K \; \alpha_{th} \; (T_a - T_0) - \boldsymbol{n} \; 3K \; \xi \, (\tau_0 - \tau)\Big)\end{aligned} \tag{2.140}$$

2.11.20 fixedDisplacementZeroShear BC

The "fixedDisplacementZeroShear" BC sets the patch-normal component of displacement while allowing the two tangential directions to be free canceling any shear stress. The BC solves equation Eq. 2.140 and project its normal to the patch, setting the other components to zero.

2.11.21 qRad_emission_absorption BC

The **qRad_emission_absorption** BC updates the temperature boundary field at a patch using the radiative equilibrium condition. Other fields are mapped through **BoundaryMapping** BC.

$$-\left(\overline{\overline{\mathbf{k}}}_w \cdot \frac{\partial T_w}{\partial \mathbf{n}}\right) \cdot \mathbf{n} = -\epsilon \; \sigma \left(T_w^4 - T_{\mathrm{inf}}^4\right) + \alpha_w \; q_{rad,w} \tag{2.141}$$

Chapter 3

Fluid Model

The **basicFluidModel** class creates the control of the fluid region by reading the *regionProperties* file. The *makeFluidModel.C* creates the fluid models based on the **basicFluidModel** class. The fluid models are described below. The **fvPatches** directory contains the boundary conditions for the fluid side. The **thermos** directory contains the fluid thermodynamics models.

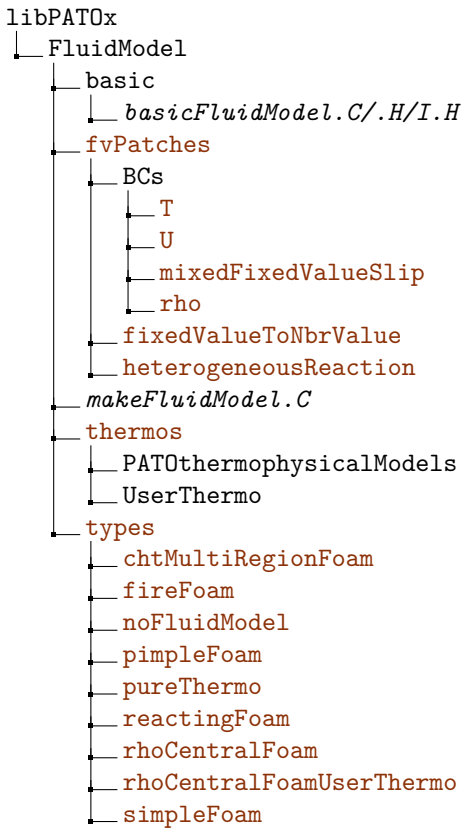

Figure 3.1. FluidModel architecture

3.1 types

3.1.1 noFluidModel Type

The **no** type is empty and serves as a place holder when the **FluidModel** is not used.

3.1.2 pureThermo Type

The **pureThermo** type updates the following thermodynamics properties using the model given by the user and write them in output directory.

- Temperature, T
- Pressure, p
- Density, ρ
- Compressibility, ψ
- Internal energy, e
- Thermal Diffusivity, α
- Effective thermal diffusivity, α_{eff}
- Electrical conductivity, σ
- Heat capacity ration, γ
- Dynamic viscosity, μ
- Effective dynamic viscosity, μ_{eff}
- Species and elements mass fraction, y_i
- Heterogeneous production rate, Ω_h

3.1.3 simpleFoam Type

The **simpleFoam** type is a steady state fluid solver for incompressible, turbulent flow, using the SIMPLE algorithm. It is based on the OpenFOAM **simpleFoam** solver extended for multiple regions.

3.1.4 pimpleFoam Type

The **pimpleFoam** type is a transient solver for incompressible, turbulent flow of Newtonian fluid, with optional mesh motion and mesh topology changes. It is based on the OpenFOAM **pimpleFoam** solver extended for multiple regions.

3.1.5 reactingFoam Type

The **reactingFoam** type is a transient solver for combustion with chemical reactions. It is based on the OpenFOAM **reactingFoam** solver extended for multiple regions.

3.1.6 fireFoam Type

The **fireFoam** type is a transient solver for fires and turbulent diffusion flames with reacting particle clouds, surface film and pyrolysis modelling. It is based on the OpenFOAM **fireFoam** solver extended for multiple regions.

3.1.7 chtMultiRegionFoam Type

The **chtMultiRegionFoam** type is a solver for steady or transient fluid flow and solid heat conduction, with conjugate heat transfer between regions, buoyancy effects, turbulence, reactions and radiation modelling.

3.1.8 rhoCentralFoam Type

The **rhoCentralFoam** type is a density-based compressible flow solver based on central-upwind schemes of Kurganov and Tadmor with support for mesh-motion topology changes. It is based on the OpenFOAM **rhoCentralFoam** solver extended for multiple regions and time stitching. Time stitching is a method to update the fluid region only if the variation of the coupled wall temperature is greater than a user defined tolerance (**tol**). The stitching is only available with **PATOxs** (or PATOx stitching) a solver under development.

$$max\left(\Delta T_w\right) = max\left(\|\frac{T_w^n - T_w^{n-1}}{\Delta t}\|\right) > tol \tag{3.1}$$

3.1.9 rhoCentralFoamUserThermo Type

The **rhoCentralFoamUserThermo** type is based on the **rhoCentralFoam** type using thermodynamics models defined in **UserThermo** directory, where the user can put its own thermodynamics types.

3.2 fvPatches

Figure 3.2 shows the architecture of the **fvPatches** in **FluidModel**. The **fvPatches** directory contains the boundary conditions. The **BCs** are the boundary conditions specific to **rhoCentralFoam** model.

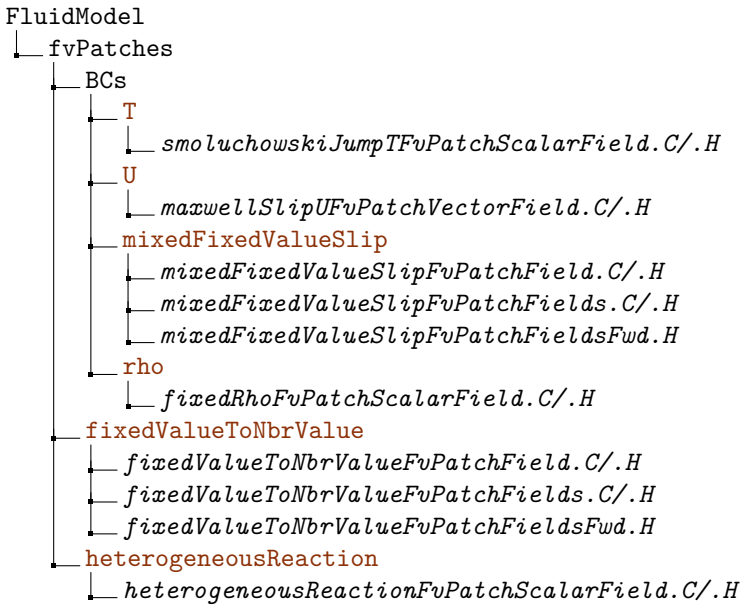

Figure 3.2. fvPatches architecture

3.2.1 mixedFixedValueSlip BC

The **mixedFixedValueSlip** is a BC specialized for supersonic fluid. It is a mixed boundary type that blends between **fixedValue** and **slip**, as opposed to the standard mixed condition that blends between **fixedValue** and **fixedGradient**; required to implement **maxwellSlipU** BC. It is based on the OpenFOAM **mixedFixedValueSlip** BC.

3.2.2 fixedRho BC

The **fixedRho** is a BC specialized for supersonic fluid. This BC provides a fixed density inlet condition for compressible solvers. It is based on the OpenFOAM **fixedRho** BC.

3.2.3 smoluchowskiJumpT BC

The **smoluchowskiJumpT** is a BC specialized for supersonic fluid. This BC provides a Smoluchowski temperature jump BC for compressible solvers. It is based on the OpenFOAM **smoluchowskiJumpT** BC.

3.2.4 maxwellSlipU BC

The **maxwellSlipU** is a BC specialized for supersonic fluid. This BC provides a Maxwell slip BC including thermal creep and surface curvature terms that can be optionally switched off. It is based on the OpenFOAM **maxwellSlipU** BC.

3.2.5 fixedValueToNbrValue BC

The **fixedValueToNbrValue** BC fixed the value of the field on the patch Φ_{own} equal to the value of the neighboring patch Φ_{nei}.

$$\Phi_{own} = \Phi_{nei} \tag{3.2}$$

3.2.6 heterogeneousReaction BC

The **heterogeneousReaction** BC calculates the heterogeneous production rate at the boundary with a fixed first order reaction parameter k_y.

$$\partial_{\boldsymbol{x}} y_i = k_y \; y_i \tag{3.3}$$

3.3 thermos

Figure 3.3 shows the architecture of the **thermos** in **FluidModel**. The **thermos** directory contains the fluid thermodynamic models specific to PATO. Modified Thermodynamics models from OpenFOAM are available in **PATOthermophysicalModels**. The *makeUserThermo.C* file creates the **UserThermo** types based on the **basicUserThermo** class. It allows PATO users to create their own thermophysical model. The user thermodynamics types are described below. Turbulence models from OpenFOAM are handled in the **turbulence** directory.

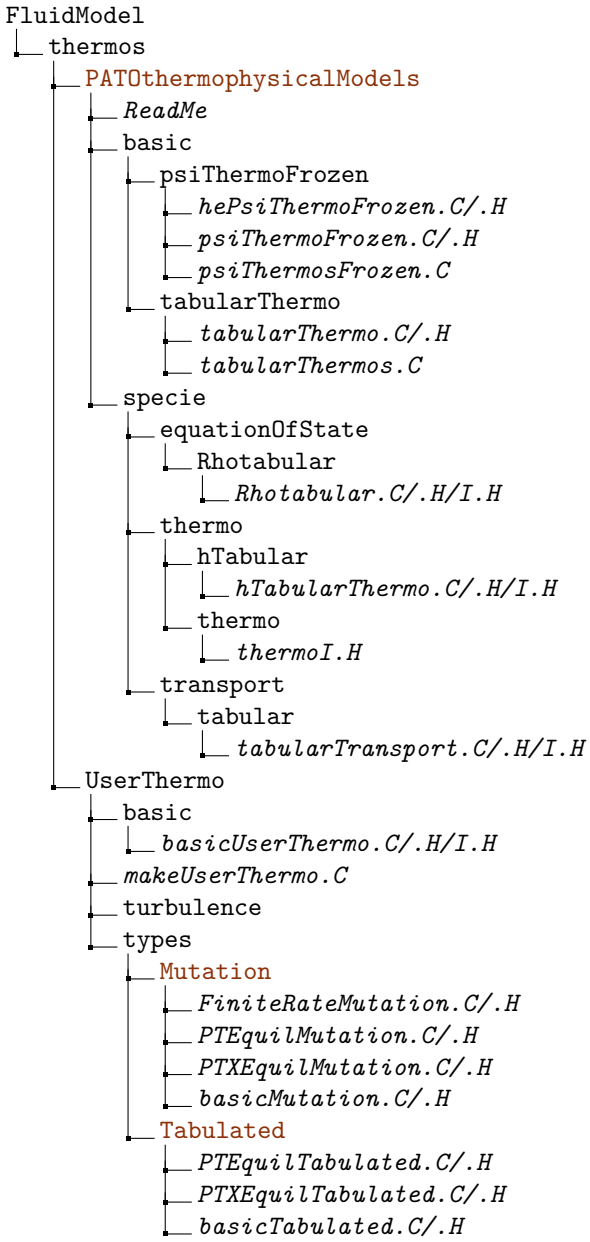

Figure 3.3. thermos architecture

3.3.1 PATOthermophysicalModels

A single model is available in the current version of PATO. This model insures the implementation of the equilibrium chemistry thermophysical property. It reads (p, T) tables of thermophysical properties of the mixture and interpolate values between points. The **hePsiThermoFrozen** is used for chemically frozen flow, and the temperature is calculated through a Newtonian method. The **tabularThermo** basic model updates the density by reading a table.

3.3.2 UserThermo: Mutation

This module utilizes MUTATION^{++} library [6, 7] to calculate mixture composition and thermophysical properties. The library has been built to centralize data and algorithms and provide accurate physico-chemical properties to CFD solvers. In this subsection, we describe the physico-chemical models implemented in the MUTATION^{++} library that has been used in PATO.

Physico-chemical models for gas phase

The energy per unit volume is written as

$$\rho e = \sum_{i \in \mathcal{H}} \rho_i e_i(T) + \rho_e e_e^{\mathrm{T}}(T),$$

with ρ being the mixture mass density, ρ_e and e_e^{T} are respectively the electron density and the electron internal energy. The internal energy for the heavy species $\mathcal{H}$ is defined as,

$$\begin{cases} e_i^{\mathrm{T}}(T) + e_i^{\mathrm{E}}(T) + e_i^{\mathrm{F}}, & \forall i \in \mathcal{A}, \\ e_i^{\mathrm{T}}(T) + e_i^{\mathrm{R}}(T) + e_i^{\mathrm{V}}(T) + e_i^{\mathrm{E}}(T) + e_i^{\mathrm{F}}, & \forall i \in \mathcal{M}, \end{cases} \tag{3.4}$$

where the superscripts T, V, R, and E represent the translational, vibrational, rotational, and electronic modes, $\mathcal{A}$ represents the set of atoms, and $\mathcal{M}$ is the set of molecules. The term e_i^{F} represents the formation energy of the species i at 298 K. The pressure is retrieved with the perfect gas law as $p = \sum_{i \in \mathcal{H}} \rho_i R_i T + \rho_e R_e T$, where R_i is the perfect gas constant of the species i.

The viscous stress tensor read as,

$$\bar{\boldsymbol{\tau}} = \mu \left[\partial_{\mathbf{x}} u + (\partial_{\mathbf{x}} u)^T - \frac{2}{3} \partial_{\mathbf{x}} \cdot u \bar{\mathbf{I}} \right]. \tag{3.5}$$

where μ is the multicomponent shear viscosity. The heat flux yields,

$$\boldsymbol{q} = \sum_{j \in \mathcal{S}} \boldsymbol{J}_i h_j + \left(\lambda^{\mathrm{T_h}} + \lambda^{\mathrm{R}} + \lambda^{\mathrm{V}} + \lambda^{\mathrm{E}} + \lambda^{\mathrm{T_e}} \right) \partial_{\mathbf{x}} T \tag{3.6}$$

where the diffusion flux of species i is,

$$\boldsymbol{J}_i = \rho_i \boldsymbol{V}_i$$

and $\boldsymbol{V}$ is the diffusion velocity.

Thermodynamic properties

The thermodynamic properties are computed with the Rigid-Rotor Harmonic-Oscillator (RRHO) model or NASA-9 database.

The RRHO model allows to express translational, rotational, vibrational, electronic and formation thermodynamic properties for each species where specific data for each species can be found in [8]. The NASA-9 database of [9] comprises the thermodynamic data of several species in the form of 9-coefficient polynomial. This data is accurate up to 20 000 K for the majority of the species and includes vibration-rotation coupling effect as well as anharmonicity corrections.

Transport properties

The transport properties such as shear viscosity μ, diffusion velocity $\boldsymbol{V}$ and thermal conductiviy λ are derived through a multiscale Chapman-Enskog perturbative solution of the Boltzmann equation [10, 11]. The interested reader is directed to [12] and [13] for a more thorough discussion.

The multicomponent shear viscosity μ is a solution of the following linear system,

$$\sum_{j\in\mathcal{H}} G^{\mu}_{ij}\alpha^{\mu}_{j} = x_i, \qquad \forall i \in \mathcal{H},$$
$$\mu = \sum_{j\in\mathcal{H}} \alpha^{\mu}_{j} x_j,$$

where x is the species mole fraction.

Similarly to the shear viscosity, the thermal conductivity of the heavy particle translational mode is a solution of the following system,

$$\sum_{j\in\mathcal{H}} G^{\lambda}_{ij}\alpha^{\lambda}_{j} = x_i, \qquad \forall i \in \mathcal{H},$$
$$\lambda^{\mathrm{T_h}} = \sum_{j\in\mathcal{H}} \alpha^{\lambda}_{j} x_i.$$

The thermal conductivity for each internal mode is given by the Eucken correction,

$$\lambda^{\mathrm{R}} = \sum_{j\in\mathcal{M}} \frac{\rho_i c^R_i}{\sum_{j\in\mathcal{H}} x_j/\mathscr{D}_{ij}},$$
$$\lambda^{\mathrm{V}} = \sum_{j\in\mathcal{M}} \frac{\rho_i c^V_i}{\sum_{j\in\mathcal{H}} x_j/\mathscr{D}_{ij},}$$
$$\lambda^{\mathrm{E}} = \sum_{j\in\mathcal{H}} \frac{\rho_i c^E_i}{\sum_{j\in\mathcal{H}} x_j/\mathscr{D}_{ij}},$$

where c^R_i, c^V_i and c^E_i are the rotational, vibrational, and electronic species specific heats per particle, respectively.

The diffusion velocity vector $\boldsymbol{V}$ is given by the generalized Stefan-Maxwell system that can be written as,

$$\sum_{j\in\mathcal{G}} G^{V}_{ij}\boldsymbol{V}_j = \boldsymbol{d}^{\Theta'}_i + k^{\Theta}_i \boldsymbol{E} \tag{3.7}$$

where $\boldsymbol{d}^{\Theta'}_i = \boldsymbol{d}'_i\Theta_i$, $k^{\Theta}_i = k_i\Theta_i$, $\Theta_i = T_h/T_i$ $i \in \mathcal{G}$. The grouping parameter k is,

$$k_j = \frac{x_j q_j}{k_B T_h} - \frac{y_i q}{k_B T_h}, \qquad \forall j \in \mathcal{G}, \quad \text{and}$$
$$q = \sum_{i\in\mathcal{G}} x_i q_i,$$

where the last equation refers to the mixture charge. Neglecting thermo-and barodiffusion, the modified driving force follows,

$$\boldsymbol{d}'_i = \frac{p}{nk_B T_h}\partial_{\mathbf{x}} x_i, \qquad \forall i \in \mathcal{G}. \tag{3.8}$$

Details on transport algorithms to solve the linear systems are provided in [14], and more details on the transport properties expressions and models are provided by [12] and [15].

Chemical kinetic

A reversible chemical reaction $r \in \mathcal{R}$ can be expressed as,

$$\sum_{i\in\mathcal{G}} \nu'_{ir} X_i \rightleftharpoons \sum_{i\in\mathcal{G}} \nu''_{ir} X_i, \quad \forall r \in \mathcal{R}$$

where X_i is the chemical symbol for species $i \in \mathcal{G}$, and $\nu_{ir}^{'}$ and $\nu_{ir}^{''}$ are the forward and backward stoichiometric coefficients for species i in reaction r.

From the Law of Mass Action, the chemical production rate is given by,

$$\dot{\omega}_i^{\text{chem}} = M_i \sum_{r \in \mathcal{R}} \nu_{ir} \mathfrak{R}_r, \quad \forall i \in \mathcal{G} \tag{3.9}$$

where $\nu_{ir} = \nu_{ir}^{''} - \nu_{ir}^{'}$ and the symbol M_i stands for the species molar mass.

The rate of progress for reaction r is given by

$$\mathfrak{R}_r = k_r^f \prod_{i \in \mathcal{G}} \left(\frac{\rho_i}{M_i} \right)^{\nu_{ir}^{'}} - k_r^b \prod_{i \in \mathcal{G}} \left(\frac{\rho_i}{M_i} \right)^{\nu_{ir}^{''}}, \quad \forall r \in \mathcal{R}, \tag{3.10}$$

where k_r^f is the forward rate that follows an Arrhenius-type and k_r^b is the backward rate. A zero rate of progress means a chemical equilibrium condition which leads to $\dot{\omega}_i^{\text{chem}} = 0$.

Chemical equilibrium

At equilibrium, the species mole fractions are obtained with a Gibbs minimization method [16], and are functions of the mixture temperature, pressure, and elemental mole fraction x_e

$$x_i = x_i \left(T, p, x_e \right) \qquad \forall \; i \in \mathcal{G} \tag{3.11}$$

leading to the species mole fraction gradient as

$$\partial_x x_i = \frac{\partial x_i}{\partial p} \partial_x p + \frac{\partial x_i}{\partial T} \partial_x T + \sum_{e \in \mathcal{E}} \frac{\partial x_i}{\partial w} \partial_x x_e, \tag{3.12}$$

where $\mathcal{E}$ is the set of elements.

The elemental diffusion flux yields,

$$\boldsymbol{J}_e^* = \mathcal{F}_e^p \partial_x p + \mathcal{F}_e^T \partial_x T + \sum_{e \in \mathcal{E}} \mathcal{F}_e^{X_e} \partial_x x_e \tag{3.13}$$

and the diffusive energy fluxes are obtained by multiplying the species mass fluxes by the mass enthalpy such that,

$$\sum_{i \in \mathcal{G}} \left(\boldsymbol{J}_i h_i \right) = \mathcal{F}_h^p \partial_x p + \mathcal{F}_h^T \partial_x T + \sum_{e \in \mathcal{E}} \mathcal{F}_h^{X_e} \partial_x x_e$$

where $\mathcal{F}_e^p$, $\mathcal{F}_e^T$, $\mathcal{F}_e^{X_l}$, $\mathcal{F}_h^p$, $\mathcal{F}_h^T$, and $\mathcal{F}_h^{x_l}$ are driving forces provided by MUTATION^{++} and their expressions can be found in [17].

basicMutation

This class is the mother class of all Mutation sub-models. Three sub-models are available:

- FiniteRateMutation
- PTEquilMutation
- PTXEquilMutation

FiniteRateMutation

This class updates the cells' and boundaries' thermodynamic and transport properties for finite-rate chemistry flows based on the different thermochemical state-vector $(\rho_i, \rho e)$, (ρ_i, T), or (y_i, T, p).

PTEquilMutation

This class updates the cells' and boundaries' thermodynamic and transport properties for chemically equilibrium flows based on the different thermochemical state-vector $(\rho, \rho e)$, or (p, T) where the elemental composition is taken by default from the mixture file.

PTXEquilMutation
This class updates the cells' and boundaries' thermodynamic and transport properties for chemically equilibrium flows based on the different thermochemical state-vector $(x_e, \rho, \rho e)$, or (y_e, p, T) where the elemental composition is a solution of the elemental mass continuity.

3.3.3 UserThermo: Tabulated

The **basicTabulated** class is the mother class of all Tabulated sub-models. Two sub-models are available:

- PTEquilTabulated
- PTXEquilTabulated

PTEquilTabulated:
This class updates the cells' and boundaries' thermodynamic and transport properties for chemically equilibrium flows. It pre-generates and stores tables for a range pressures and temperatures utilizing MUTATION^{++}.

PTXEquilTabulated:
This class updates the cells' and boundaries' thermodynamic and transport properties for chemically equilibrium flows. It pre-generates and stores tables for a range pressures, temperatures and elemental composition utilizing MUTATION^{++}.

Chapter 4

Tutorials

Figure 4.1 shows the architecture of **tutorials**. The **1D**, **2D**, and **3D** tutorials are presented here.

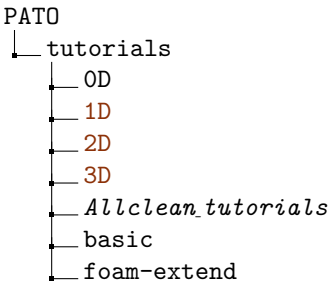

Figure 4.1. Tutorials architecture

4.1 1D tutorials

Figure 4.2 shows the architecture of the 1D tutorials. The inputs and expected outputs of the 1D tutorials are described below.

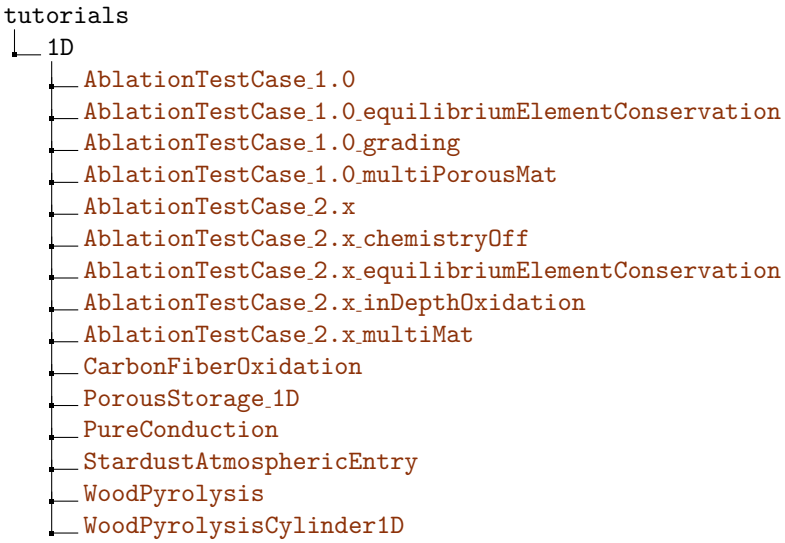

Figure 4.2. 1D tutorials architecture

4.1.1 AblationTestCase_1.0

The **AblationTestCase_1.0** tutorial computes the 1D thermal response of the TACOT material. Only the architecture of this tutorial (Fig. 4.3) is presented because the architecture of all the tutorials are similar.

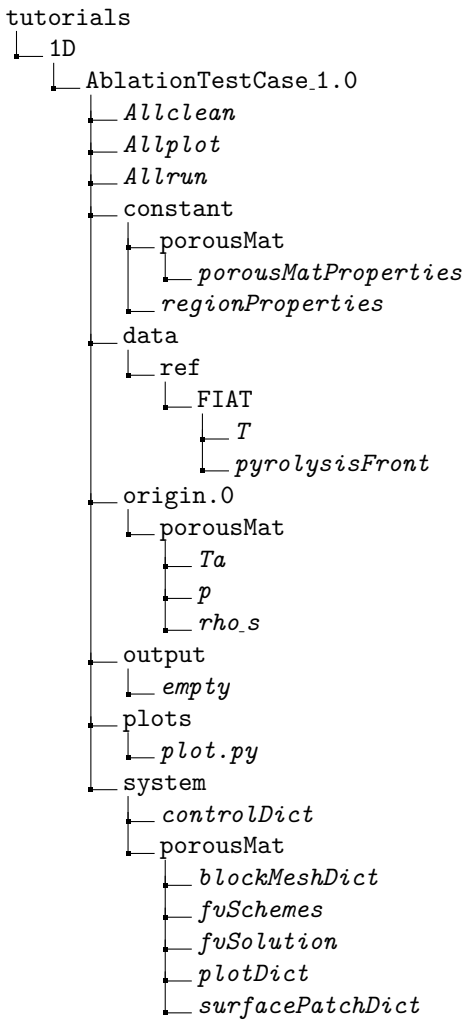

Figure 4.3. AblationTestCase_1.0 tutorial architecture

The *Allclean* file cleans the time folders, the figures, the mesh and the PATO outputs. The *Allplot* file produces the 1D plot (Fig. 4.4). The *Allrun* file runs the tutorial case. The *regionProperties* file selects the material region names. The *p*, *rho_s* and *Ta* files are respectively the initialization of the pressure, solid density and temperature fields. *controlDict* file sets the input parameters essential for the creation of the database. The *blockMeshDict* creates the 1D mesh and the boundaries. The *fvSchemes* file sets the numerical schemes for terms, such as derivatives in equations, that are calculated during a simulation. The *fvSolution* controls the equation solvers, tolerances and algorithms.
Table 4.1 shows the summary of the material sub-models used in this case. Listing 4.1 shows the *porousMatProperties* file and the different user material sub-models.

Table 4.1. Summary of the material sub-models

Model/Tutorial	AblationTestCase_1.0
Mass	**DarcyLaw**
Energy	**Pyrolysis**
IO	**no**
Gas Properties	**Tabulated**
Material Properties	**Porous**
Pyrolysis	**LinearArrhenius**
TimeControl	**GradP**

Listing 4.1. "constant/porousMat/porousMatProperties" file

```
IO {
  writeFields (Ta);
  probingFunctions (plotDict surfacePatchDict);
}
Pyrolysis {
  PyrolysisType LinearArrhenius;
}
Mass {
  MassType DarcyLaw;
}
Energy {
  EnergyType Pyrolysis;
}
GasProperties {
  GasPropertiesType Tabulated;
  GasPropertiesFile
  "$PATO_DIR/data/Materials/Composites/TACOT/gasProperties";
}
MaterialProperties {
  MaterialPropertiesType Porous;
  MaterialPropertiesDirectory
  "$PATO_DIR/data/Materials/Composites/TACOT";
}
TimeControl {
  TimeControlType GradP;
  chemTransEulerStepLimiter no;
}
```

The TACOT material is made of a carbon fiber phase and a phenolic matrix phase that evaporates at high temperature. Listing 4.2 shows the initial solid density and the volume fraction of the 2 phases.

Listing 4.2. Part of "TACOT/constantProperties" file

```
nSolidPhases     2;
rhoI[1] rhoI[1] [1 -3 0 0 0 0 0] 1600;
epsI[1] epsI[1] [0 0 0 0 0 0 0] 0.1;
rhoI[2] rhoI[2] [1 -3 0 0 0 0 0] 1200;
epsI[2] epsI[2] [0 0 0 0 0 0 0] 0.1;
x[C] 0.2282;
x[ht] 0.6623;
x[O] 0.1095;
x[N] 0.0;
```

The gaseous elemental mole fractions ($x[i]$) of the TACOT material is considered constant during the simulation. In 1 kmole of phenolic resin (C_6H_6O), we have

$$m_C = 6 \text{ kmole } \times M_{w,C} = 72.0669 \text{ kg} \tag{4.1}$$

$$m_H = 6 \text{ kmole } \times M_{w,H} = 6.0474 \text{ kg} \tag{4.2}$$

$$m_O = 1 \text{ kmole } \times M_{w,O} = 15.9994 \text{ kg} \tag{4.3}$$

$$m_{tot} = m_C + m_H + m_O = 94.1137 \text{ kg} \tag{4.4}$$

50% of the total mass evaporates into gas. All the hydrogen and oxygen are transformed into gas. Therefore, only carbon is left in the charred material.

$$m_C = 0.5\ m_{tot} - m_H - m_O = 25.01005 \text{ kg} \tag{4.5}$$

$$z_C = \frac{m_C}{m_{tot}} = 0.53149 \tag{4.6}$$

$$z_H = \frac{m_H}{m_{tot}} = 0.12851 \tag{4.7}$$

$$z_O = \frac{m_O}{m_{tot}} = 0.34 \tag{4.8}$$

$$x_C = \frac{z_C}{z_C + z_O + z_H} = 0.2282 \tag{4.9}$$

$$x_H = \frac{z_H}{z_C + z_O + z_H} = 0.6623 \tag{4.10}$$

$$x_O = \frac{z_O}{z_C + z_O + z_H} = 0.1095 \tag{4.11}$$

The writeFields input writes the *Ta* field every 0.1 s as defined in the *controlDict* file. The *plotDict* and *surfacePatchDict* files set the internal and the surface patch sampling configuration, coordinates and fields. Listing 4.3 shows an example of *Ta* file that initializes the temperature. The dimension is Kelvin. All the internal cell centers have a uniform value of 300. The top boundary field varies linearly from 300 K to 1644 K during the first 0.1 s. The sides and bottom boundary fields have a **zeroGradient** boundary conditions (Eq. 4.12).

$$\boldsymbol{\partial_x} f \cdot \boldsymbol{n} = 0 \tag{4.12}$$

Listing 4.3. Example of "origin.0/porousMat/Ta" file

```
dimensions      [0 0 0 1 0 0 0];
internalField   uniform 300;
boundaryField{
    top {
        type uniformFixedValue;
        uniformValue table ((0   300) (0.1 1644));
    }
    sides {
        type            zeroGradient;
    }
    bottom {
        type            zeroGradient;
    }
}
```

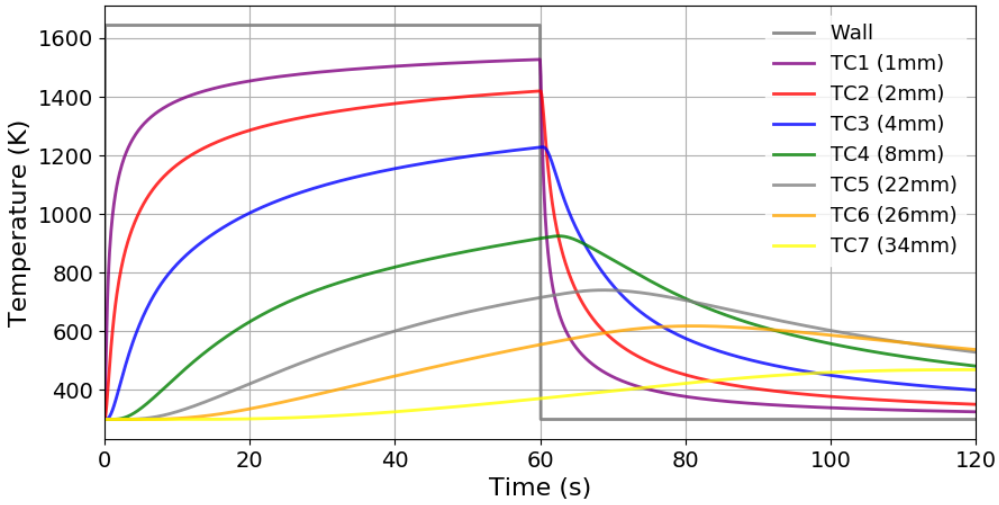

Figure 4.4. Thermal response of AblationTestCase_1.0 case

4.1.2 AblationTestCase_1.0_equilibriumElementConservation

The **AblationTestCase_1.0_equilibriumElementConservation** tutorial computes the 1D thermal response of the TACOT material using the elemental mass conservation (Fig. 4.5). Table 4.2 shows the summary of the material sub-models used in this case. Listing 4.4 shows the *porousMatProperties* file and the different user material sub-models.

Table 4.2. Summary of the material sub-models

Model/Tutorial	AblationTestCase_1.0_equiElemCons
Mass	**DarcyLaw**
Energy	**Pyrolysis**
IO	**no**
Gas Properties	**Equilibrium**
Material Properties	**Porous**
Pyrolysis	**LinearArrhenius**
MaterialChemistry	**EquilibriumElement**
TimeControl	**GradP**

Listing 4.4. "constant/porousMat/porousMatProperties" file

```
IO {
    probingFunctions (plotDict surfacePatchDict);
}
Mass {
    MassType DarcyLaw;
}
Energy {
    EnergyType Pyrolysis;
}
GasProperties {
    GasPropertiesType Equilibrium;
}
```

```
13 MaterialChemistry {
14   MaterialChemistryType EquilibriumElement;
15   mixture tacotair;
16 }
17 MaterialProperties {
18   MaterialPropertiesType Porous;
19   MaterialPropertiesDirectory
20   "$PATO_DIR/data/Materials/Composites/TACOT";
21 }
22 Pyrolysis {
23   PyrolysisType LinearArrhenius;
24 }
25 TimeControl {
26   TimeControlType GradP;
27   chemTransEulerStepLimiter no;
28 }
```

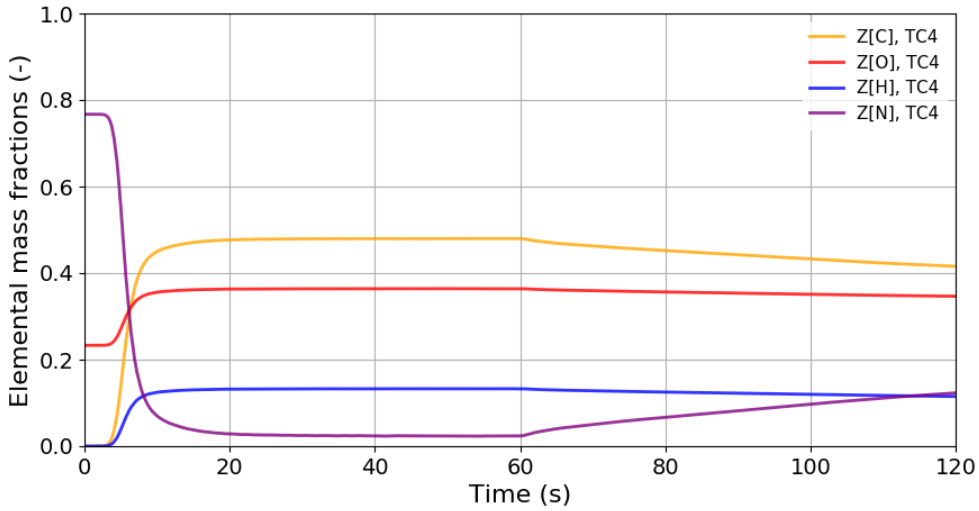

Figure 4.5. Temporal evolution of the elemental mass fractions

4.1.3 AblationTestCase_1.0_grading

The **AblationTestCase_1.0_grading** tutorial computes the 1D thermal response of a porous material with a graded phase (Fig. 4.6). Table 4.3 shows the summary of the material sub-models used in this case. Listing 4.5 shows the *porousMatProperties* file and the different user material sub-models. Listing 4.6 and 4.7 show the grading properties file and the related gradedTACOT constant properties. The phase 3 has a graded volume fraction from 0.1 to 0.02 for the first in-depth millimeter in the y-axis.

Table 4.3. Summary of the material sub-models

Model/Tutorial	AblationTestCase_1.0_grading
Mass	**DarcyLaw**
Energy	**Pyrolysis**
IO	**Profile**
Gas Properties	**Tabulated**
Material Properties	**GradedPorous**
Pyrolysis	**LinearArrhenius**
TimeControl	**GradP**

Listing 4.5. "constant/porousMat/porousMatProperties" file

```
IO {
  IOType Profile;
  topPatch top;
  bottomPatch bottom;
  plot1DProfileList (eps_s_phase1 eps_s_phase2 eps_s_phase3);
  plot1DMassLoss yes;
  probingFunctions (plotDict surfacePatchDict);
}
Mass {
  MassType DarcyLaw;
}
Energy {
  EnergyType Pyrolysis;
}
GasProperties {
  GasPropertiesType Tabulated;
  GasPropertiesFile
  "$PATO_DIR/data/Materials/Composites/TACOT/gasProperties";
}
MaterialProperties {
  MaterialPropertiesType GradedPorous;
  MaterialPropertiesDirectory
  "$PATO_DIR/data/Materials/Composites/TACOT";
}
Pyrolysis {
  PyrolysisType LinearArrhenius;
}
TimeControl {
  TimeControlType GradP;
  chemTransEulerStepLimiter no;
}
```

Listing 4.6. Example of the "constant/grading" file

```
1  //   d(m)      epsI(-)
2       0.05       0.1
3       0.049      0.02
```

Listing 4.7. Part of the "gradedTACOT/constantProperties" file

```
1  // grading prop. file name (distance and volume fraction)
2  gradingFileEpsI[3] "$FOAM_CASE/constant/grading";
3  // axis of the grading
4  gradingAxisEpsI[3] (0 1 0);
```

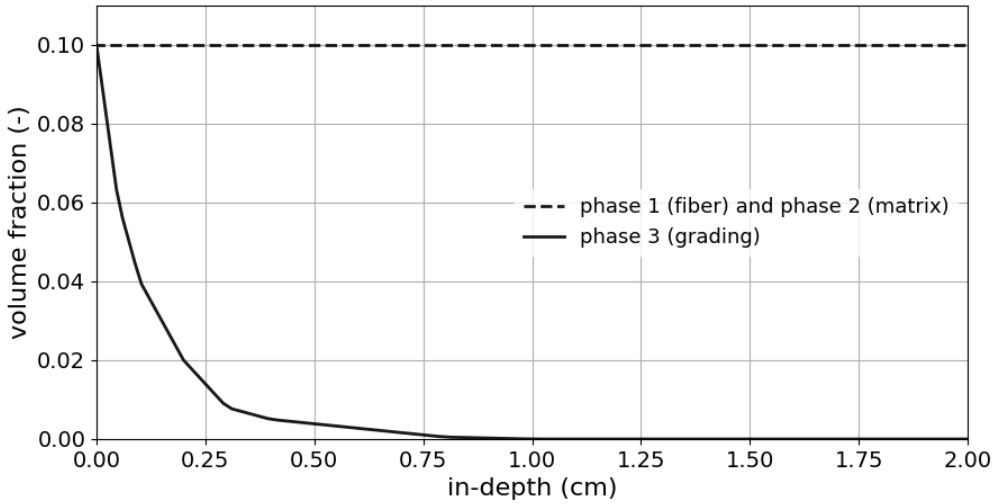

Figure 4.6. 1D profile of the phases' volume fraction

4.1.4 AblationTestCase_1.0_multiPorousMat

The **AblationTestCase_1.0_multiPorousMat** tutorial computes the 1D thermal response and the boundary interface between 2 porous materials: TACOT and Cork (Fig. 4.7). Listing 4.8 shows the mappedWall type in the *boundary* file of the porousMat1 polyMesh.

Listing 4.8. "constant/porousMat1/polyMesh/boundary" file

```
1  porousMat1_to_porousMat2
2  {
3      type            mappedWall;
4      inGroups        1(wall);
5      nFaces          1;
6      startFace       100;
7      sampleMode      nearestPatchFace;
8      sampleRegion    porousMat2;
9      samplePatch     porousMat2_to_porousMat1;
10 }
```

Listing 4.9 shows the **coupledMixed** boundary type of the pressure boundary field (Eq. 4.14).

$$\underline{\underline{\gamma}}_i^S = \frac{1}{2}\left(\underline{\underline{\gamma}}_i + \underline{\underline{\gamma}}_i^T\right) \tag{4.13}$$

$$p_{1/2} = \frac{\frac{\underline{\underline{\gamma}}_2^S}{\Delta x_2}}{\frac{\underline{\underline{\gamma}}_1^S}{\Delta x_1} + \frac{\underline{\underline{\gamma}}_2^S}{\Delta x_2}}\, p_1 + \left(1 - \frac{\frac{\underline{\underline{\gamma}}_2^S}{\Delta x_2}}{\frac{\underline{\underline{\gamma}}_1^S}{\Delta x_1} + \frac{\underline{\underline{\gamma}}_2^S}{\Delta x_2}}\right) p_2 \tag{4.14}$$

Listing 4.9. "origin.0/porousMat1/p" file

```
boundaryField {
    porousMat1_to_porousMat2   {
      type              coupledMixed;
      value             uniform 300;
      Tnbr              p;
      kappaMethod       lookup;
      kappa             Gamma_symm;
    }
}
```

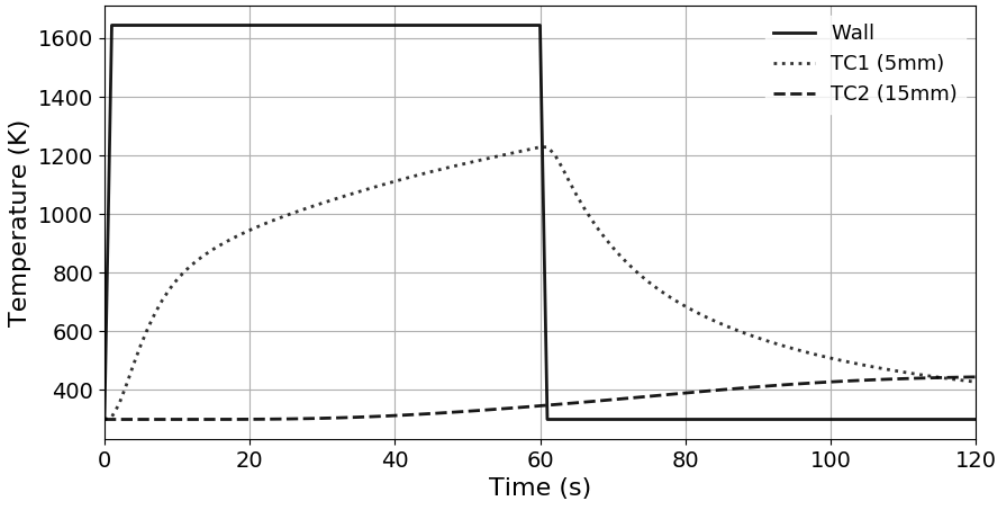

Figure 4.7. Thermal response of two adjacent porous materials

4.1.5 AblationTestCase_2.x

The **AblationTestCase_2.x** tutorial computes the 1D thermal response of the TACOT material using the surface mass and energy balance boundary conditions (Fig. 4.8). Listing 4.10 shows the **Bprime** boundary type of the temperature boundary field (Eq. 4.15). Listing 4.11 shows a part of the *BoundaryConditions* file. During the first 0.1 s, the recovery enthalpy (h_r) varies from 0 to 1.5 MJ $\cdot$ kg^{-1} and the heat transfer coefficient (C_H) from 3 to 300 kg $\cdot$ m^{-2} $\cdot$ s^{-1}. The pressure ($p = 101325$ Pa), the radiative heat flux ($q_{rad} = 0$ W $\cdot$ m^{-2}), the blowing correction coefficient ($\lambda = 0.5$) and the background temperature ($T_\infty = 300$ K) stay constant during the simulation. Listing 4.12 shows the dynamic mesh flag that allows the boundary to move over time following the **Bprime** boundary conditions type (Eq. 4.16).

$$T_w = T_{int} + \frac{\Delta x_1}{\boldsymbol{n} \cdot \underline{\underline{\mathbf{k_w}}} \cdot \boldsymbol{n}} (C'_H \left(h_e - h_w\right) + \dot{m}_{pg} \left(h_{pg} - h_w\right) + \dot{m}_{ca} \left(h_{ca} - h_w\right) - \varepsilon_w \sigma \left(T_w^4 - T_\infty^4\right) + \alpha_w q_{pla}) \tag{4.15}$$

$$\boldsymbol{v}_{mesh} = \frac{C'_H \; B'_{ca}}{\rho_{sw}} \; \mathbf{n} \tag{4.16}$$

Listing 4.10. "origin.0/porousMat/Ta" file

```
boundaryField {
    top {
        type Bprime;
        mixtureMutationBprime tacot26;
        environmentDirectory
        "$PATO_DIR/data/Environments/RawData/Earth";
        mappingType constant;
        mappingFileName
        "constant/porousMat/BoundaryConditions";
        mappingFields
        ((rhoeUeCH "1") (h_r "2") (heatOn "3"));
        p 101325;
        qRad 0;
        lambda 0.5;
        Tbackground 300;
        value uniform 300;
    }
}
```

Listing 4.11. Part of the "constant/porousMat/BoundaryConditions" file

```
// t(s)     CH(kg/m2/s)     h_r(J/kg)   heatOn(-)
   0          0.3e-2          0             1
   0.1        0.3             1.5e6         1
```

Listing 4.12. Part of the "constant/porousMat/porousMatProperties" file

```
movingMesh yes;
```

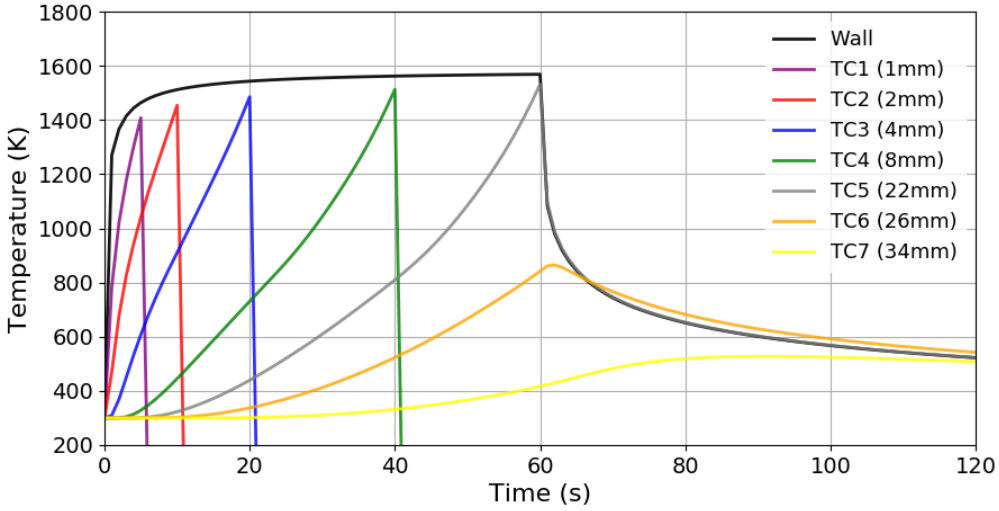

Figure 4.8. Thermal response of AblationTestCase_2.x case

4.1.6 AblationTestCase_2.x_chemistryOff

The **AblationTestCase_2.x_chemistryOff** tutorial computes the 1D thermal response of the TACOT material using the surface mass and energy balance and energy balance boundary condition without surface chemistry (Fig. 4.9). Listing 4.13 shows the **Bprime** boundary type of the temperature boundary field (Eq. 4.17).

$$T_w = T_{int} + \frac{\Delta x_1}{\boldsymbol{n} \cdot \underline{\underline{\mathbf{k_w}}} \cdot \boldsymbol{n}} - \varepsilon_w \sigma \left(T_w^4 - T_\infty^4\right) + \alpha_w q_{pla} \tag{4.17}$$

Listing 4.13. "origin.0/porousMat/Ta" file

```
boundaryField {
    top {
        type Bprime;
        mixtureMutationBprime tacot26;
        environmentDirectory
        "$PATO_DIR/data/Environments/RawData/Earth";
        movingMesh yes;
        mappingType constant;
        mappingFileName
        "constant/porousMat/BoundaryConditions";
        mappingFields
        (
            (p "1")
            (rhoeUeCH "2")
            (h_r "3")
            (hconv "4")
            (Tedge "5")
            (chemistryOn "6")
            );
        qRad 0;
```

```
21          lambda 0.5;
22          Tbackground 300;
23          value uniform 300;
24      }
25  }
```

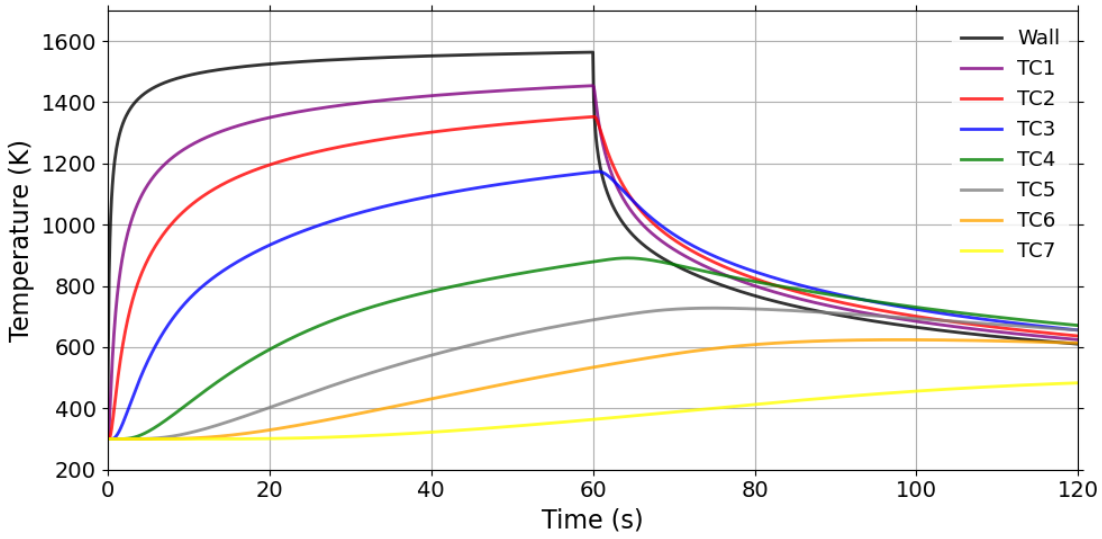

Figure 4.9. Thermal response of AblationTestCase_2.x_chemistryOff case

4.1.7 AblationTestCase_2.x_equilibriumElementConservation

The **AblationTestCase_2.x_equilibriumElementConservation** tutorial computes the 1D thermal response of the TACOT material using the **Bprime** boundary type and the material sub-models from Table 4.2.

4.1.8 AblationTestCase_2.x_inDepthOxidation

The **AblationTestCase_2.x_inDepthOxidation** tutorial computes the 1D thermal response of the TACOT material, including in-depth oxidation (Fig. 4.10). Table 4.4 shows the summary of the material sub-models used in this case. Listing 4.14 shows the *porousMatProperties* file and the different user material sub-models.

Table 4.4. Summary of the material sub-models

Model/Tutorial	AblationTestCase_2.x_inDepthOxidation
IO	**Profile**
Material Properties	**Porous**
Pyrolysis	**LinearArrhenius**
Mass	**DarcyLaw_Heterogeneous**
Energy	**Pyrolysis_Heterogeneous_SpeciesDiffusion**
Gas Properties	**FiniteRate**
Material Chemistry	**SpeciesConservation**
Volume Ablation	**FibrousMaterialTypeA**
TimeControl	**GradP**

Listing 4.14. "constant/porousMat/porousMatProperties" file

```
IO {
  IOType Profile;
  topPatch "top";
  bottomPatch "bottom";
  plot1DProfileList (Y[O2]);
  plot1DMassLoss yes;
  probingFunctions (plotDict surfacePatchDict);
}
MaterialProperties {
  MaterialPropertiesType Porous;
  MaterialPropertiesDirectory
  "$PATO_DIR/data/Materials/Composites/TACOT_newChemistry";
  detailedSolidEnthalpies yes;
}
Pyrolysis {
  PyrolysisType LinearArrhenius;
}
Mass {
  MassType DarcyLaw_Heterogeneous;
}
Energy {
  EnergyType Pyrolysis_Heterogeneous_SpeciesDiffusion;
}
GasProperties {
  GasPropertiesType FiniteRate;
}
MaterialChemistry {
  MaterialChemistryType SpeciesConservation;
  mixture TACOT_oxidation_species;
}
VolumeAblation {
  VolumeAblationType FibrousMaterialTypeA;
}
TimeControl {
  TimeControlType GradP_ChemYEqn;
  chemTransEulerStepLimiter no;
}
```

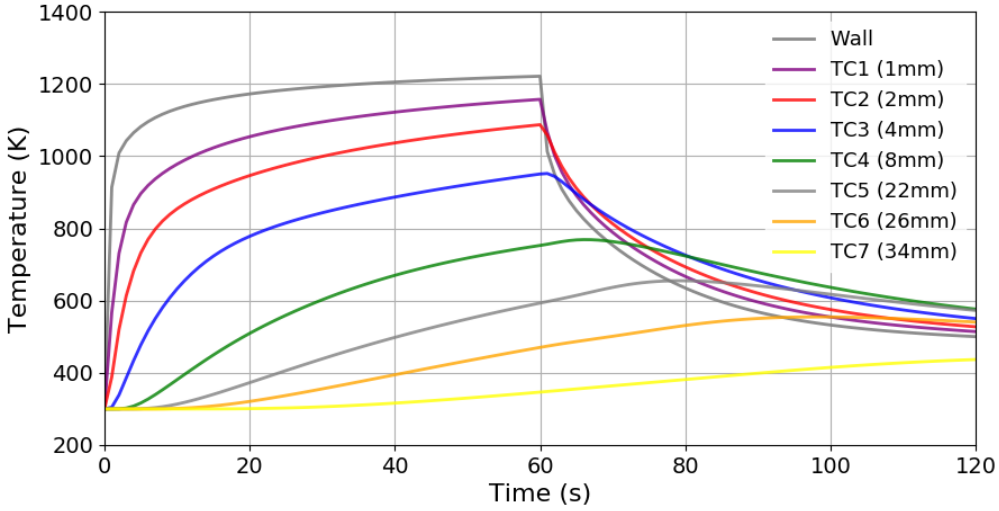

Figure 4.10. Thermal response including in-depth oxidation

4.1.9 AblationTestCase_2.x_multiMat

The **AblationTestCase_2.x_multiMat** tutorial computes the 1D thermal response of 1 porous and 2 non-porous materials. Listing 4.15 shows the *regionProperties* file that selects the 3 different materials: porousMat, subMat1 and subMat2. Table 4.5 shows the summary of the subMat1 material sub-models used in this case. Listing 4.16 shows the *subMat1Properties* file and the different user material sub-models.

Listing 4.15. "constant/regionProperties" file

```
regions (solid (porousMat subMat1 subMat2));
```

Table 4.5. Summary of the subMat1 material sub-models

Model/Tutorial	AblationTestCase_2.x_multiMat
Energy	PureConduction
Material Properties	Fourier

Listing 4.16. "constant/subMat1/subMat1Properties" file

```
Energy {
  EnergyType PureConduction;
}
MaterialProperties {
  MaterialPropertiesType Fourier;
  MaterialPropertiesDirectory
  "$PATO_DIR/data/Materials/Fourier/FourierTemplate";
}
```

4.1.10 CarbonFiberOxidation

The **CarbonFiberOxidation** tutorial computes the 1D in-depth oxidation of the CarbonFiberPreform material (Fig. 4.11 and 4.12). Table 4.6 shows the summary of the material sub-models used in this case. Listing 4.17 shows the *porousMatProperties* file and the different user material sub-models.

Table 4.6. Summary of the material sub-models

Model/Tutorial	AblationTestCase_2.x_inDepthOxidation
IO	**Profile**
Material Properties	**Porous**
Pyrolysis	**virgin**
Mass	**DarcyLaw_Heterogeneous**
Gas Properties	**FiniteRate**
Material Chemistry	**SpeciesConservation**
Volume Ablation	**FibrousMaterialTypeA**
TimeControl	**GradP**

Listing 4.17. "constant/porousMat/porousMatProperties" file

```
IO {
  IOType Profile;
  topPatch "top";
  bottomPatch "bottom";
  plot1DProfileList (Y[O2] Y[CO2] rho_s);
  plot1DMassLoss yes;
  probingFunctions (plotDict surfacePatchDict);
}
MaterialProperties {
  MaterialPropertiesType Porous;
  MaterialPropertiesDirectory
  "$PATO_DIR/data/Materials/Composites/CarbonFiberPreform";
  detailedSolidEnthalpies yes;
}
Pyrolysis {
  PyrolysisType LinearArrhenius;
}
Mass {
  MassType DarcyLaw_Heterogeneous;
}
GasProperties {
  GasPropertiesType FiniteRate;
}
MaterialChemistry {
  MaterialChemistryType SpeciesConservation;
  mixture Carbon_oxidation_species;
}
VolumeAblation {
  VolumeAblationType FibrousMaterialTypeA;
  energyConservation isothermal;
}
TimeControl {
  TimeControlType GradP_ChemYEqn;
  chemTransEulerStepLimiter no;
}
```

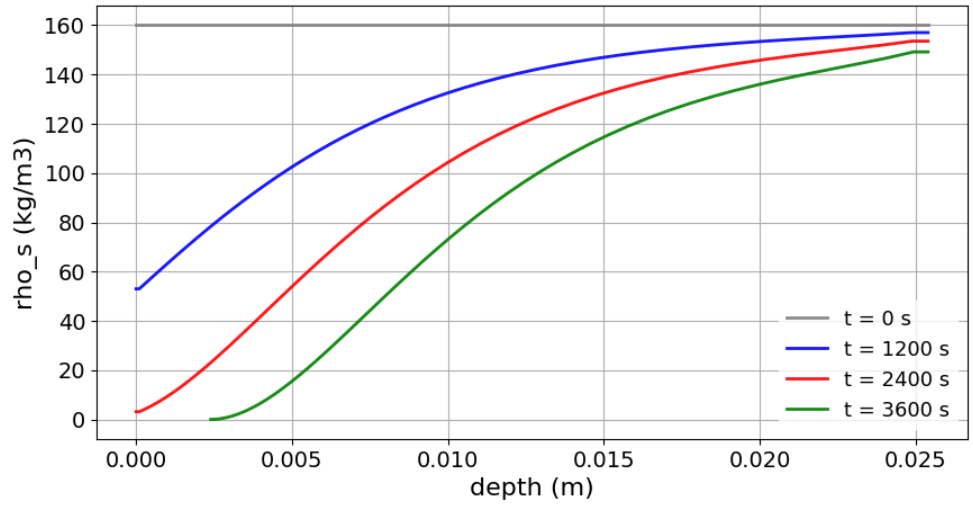

Figure 4.11. Temporal and in-depth profile of the solid density

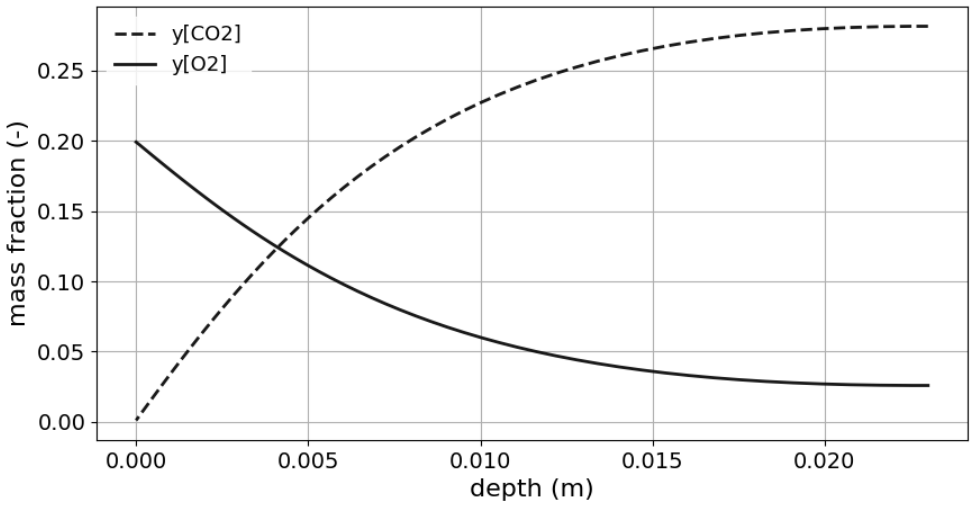

Figure 4.12. In-depth profile of the gaseous mass fractions

Listing 4.18 shows the CHEMKIN input file for the material chemistry model. There are 3 elements, 5 species and 1 finite-rate reaction (Eq. 4.18).

$$\mathrm{C(gr) + O_2 \xrightarrow{k_1(T)} CO_2}$$

$$k_1(T) = 5 \cdot 10^7 \; T^0 \; \exp\left(\frac{T}{28600}\right) \tag{4.18}$$

Listing 4.18. "Carbon_oxidation.inp" CHEMKIN file

```
ELEMENTS
  C   N   O
END
SPECIES
  N2   O2   CO2   CO   C(gr)
END
REACTIONS
   C(gr) + O2 => CO2        5e7     0     28680
END
```

4.1.11 PorousStorage_1D

The **PorousStorage_1D** tutorial computes the 1D thermal response of a quartzite bed. The *singleGraph* file produces the 1D plot (Fig. 4.13). The *Ta* and *Tg* files respectively the initialization of the solid temperature field and the gazeous temperature field. Table 4.7 show the summary of the material sub-models used in this case. Listing 4.19 shows the *porousMatProperties* file ans the different user material sub-models.

Table 4.7. Summary of the material sub-models

Model/Tutorial	PorousStorage_1D
Mass	**DarcyLaw2T**
Energy	**Pyrolysis2T**
IO	**no**
Gas Properties	**Tabulated2T**
Material Properties	**Porous**
Pyrolysis	**virgin**
TimeControl	**GradP**

Listing 4.19. "constant/porousMat/porousMatProperties" file

```
IO {
   writeFields (U vG);
   probingFunctions (plotDict);
}
Pyrolysis {
   PyrolysisType virgin;
}
Mass {
   MassType DarcyLaw2T;
}
Energy {
   EnergyType Pyrolysis2T;
}
```

```
14 GasProperties {
15   GasPropertiesType Tabulated2T;
16   GasPropertiesFile
17   "$PATO_DIR/data/Materials/Granular/Quartzite_bed/gasProperties_with_k";
18 }
19 MaterialProperties {
20   MaterialPropertiesType Porous;
21   MaterialPropertiesDirectory
22   "$PATO_DIR/data/Materials/Granular/Quartzite_bed";
23 }
24 TimeControl {
25   TimeControlType GradP;
26   chemTransEulerStepLimiter no;
27 }
```

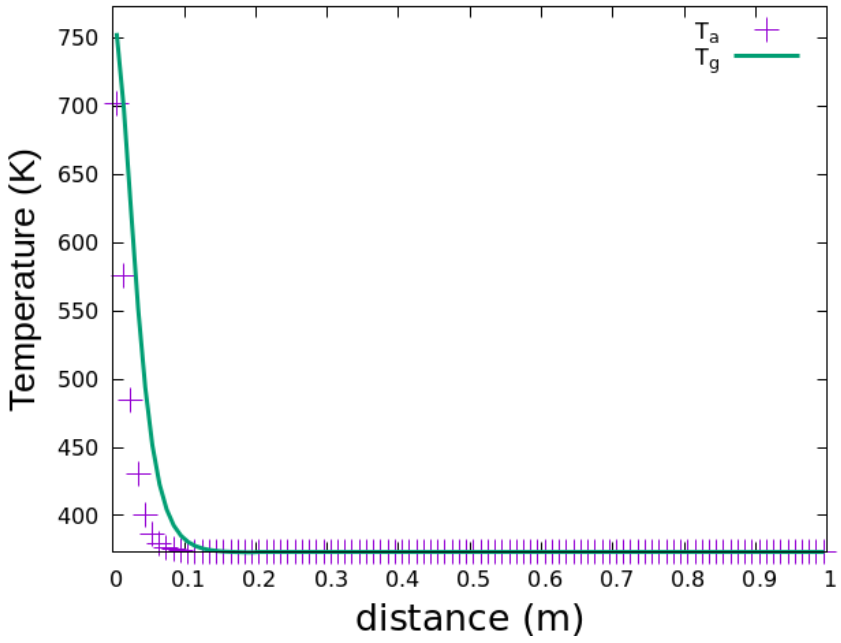

Figure 4.13. Solid and gaz temperatures at $t = 60$ s

4.1.12 PureConduction

The **PureConduction** tutorial computes the 1D thermal response of the TACOT material using the pure conduction model. Table 4.8 shows the summary of the material sub-models used in this case. Listing 4.20 shows the *porousMatProperties* file and the different user material sub-models.

Table 4.8. Summary of the material sub-models

Model/Tutorial	AblationTestCase_2.x_inDepthOxidation
IO	**no**
Energy	**PureConduction**
Material Properties	**Porous**
Pyrolysis	**virgin**

Listing 4.20. "constant/porousMat/porousMatProperties" file

```
IO {
  probingFunctions (plotDict surfacePatchDict);
}
Energy {
  EnergyType PureConduction;
}
MaterialProperties {
  MaterialPropertiesType PureConduction;
  MaterialPropertiesDirectory
  "$PATO_DIR/data/Materials/Composites/TACOT";
}
Pyrolysis {
  PyrolysisType virgin;
}
```

4.1.13 StartdustAtmosphericEntry

The **StartdustAtmosphericEntry** tutorial computes the 1D thermal response of the TACOT material using the surface mass and energy balance boundary conditions during the Stardust atmospheric entry (Fig. 4.14).

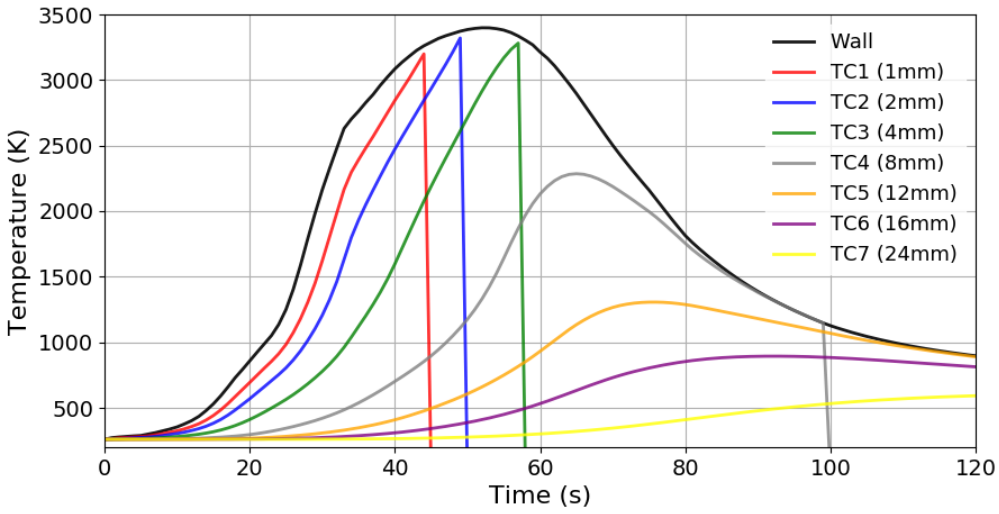

Figure 4.14. Thermal response of the Stardust atmospheric entry

4.1.14 WoodPyrolysis

The **WoodPyrolysis** tutorial computes the 1D thermal response of the Wood material (Fig. 4.15). Table 4.9 shows the summary of the material sub-models used in this case. Listing 4.21 shows the *porousMatProperties* file and the different user material sub-models.

Table 4.9. Summary of the material sub-models

Model/Tutorial	WoodPyrolysis
IO	**no**
Pyrolysis	**LinearArrhenius**
Mass	**DarcyLaw**
Energy	**Pyrolysis**
Gas Properties	**Tabulated**
Material Properties	**Porous**
TimeControl	**GradP**

Listing 4.21. "constant/porousMat/porousMatProperties" file

```
IO {
  probingFunctions (plotDict surfacePatchDict);
}
Pyrolysis {
  PyrolysisType LinearArrhenius;
}
Mass {
  MassType DarcyLaw;
}
Energy {
  EnergyType Pyrolysis;
}
GasProperties {
  GasPropertiesType Tabulated;
  GasPropertiesFile
  "$PATO_DIR/data/Materials/Wood/Generic/gasProperties";
}
MaterialProperties {
  MaterialPropertiesType Porous;
  MaterialPropertiesDirectory
  "$PATO_DIR/data/Materials/Wood/Generic"
  detailedSolidEnthalpies yes;
}
TimeControl {
  TimeControlType GradP;
}
```

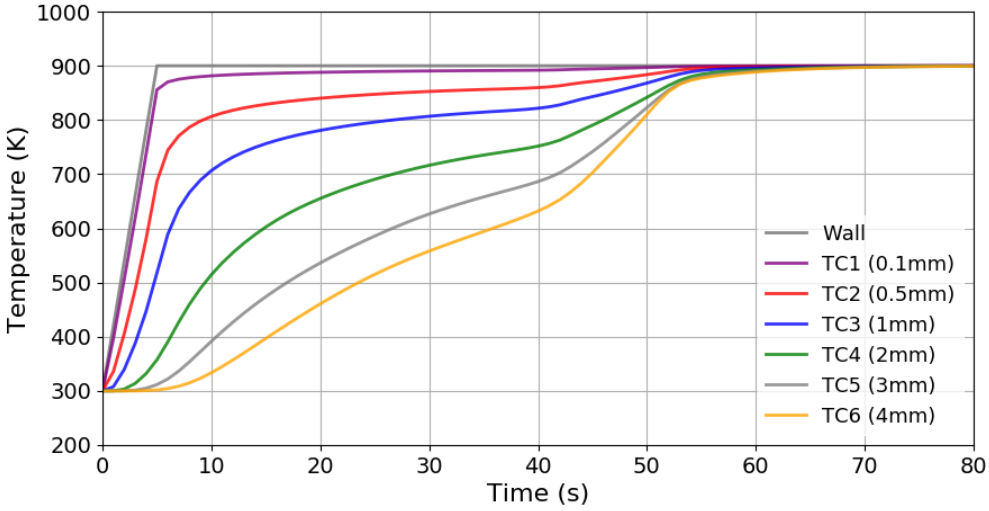

Figure 4.15. Temporal evolution of the temperature in the Wood

4.1.15 WoodPyrolysisCylinder1D

The **WoodPyrolysisCylinder1D** computes the 1D-axisymetric thermo-mechanical response of the Wood material (Fig. 4.16). Table 4.10 shows the summary of the material sub-models used in this case. Listing 4.22 shows the *porousMatProperties* file and the different user material sub-models.

Table 4.10. Summary of the material sub-models

Model/Tutorial	WoodPyrolysisCylinder1D
IO	**no**
Pyrolysis	**LinearArrhenius**
Mass	**DarcyLaw**
Energy	**Pyrolysis**
Gas Properties	**Tabulated**
Material Properties	**Porous_polynomial_k_UQ**
TimeControl	**GradP**

Listing 4.22. "constant/porousMat/porousMatProperties" file

```
1  IO {
2    probingFunctions (plotDict surfacePatchDict);
3     writeFields (tau D E alpha sigma epsilon);
4  }
5  Pyrolysis {
6    PyrolysisType LinearArrhenius;
7  }
8  Mass {
9    MassType DarcyLaw;
10 }
11 Energy {
12   EnergyType Pyrolysis;
13 }
14 SolidMechanics
```

```
15 {
16     SolidMechanicsType   Displacement;
17     planeStress          true;
18 }
19 GasProperties {
20   GasPropertiesType Tabulated;
21   GasPropertiesFile
22   "$PATO_DIR/data/Materials/Wood/GenericUNC/gasProperties";
23 }
24 MaterialProperties {
25   MaterialPropertiesType Porous;
26   MaterialPropertiesDirectory
27   "$PATO_DIR/data/Materials/Wood/GenericUNC"
28   detailedSolidEnthalpies yes;
29 }
30 TimeControl {
31   TimeControlType GradP;
32 }
```

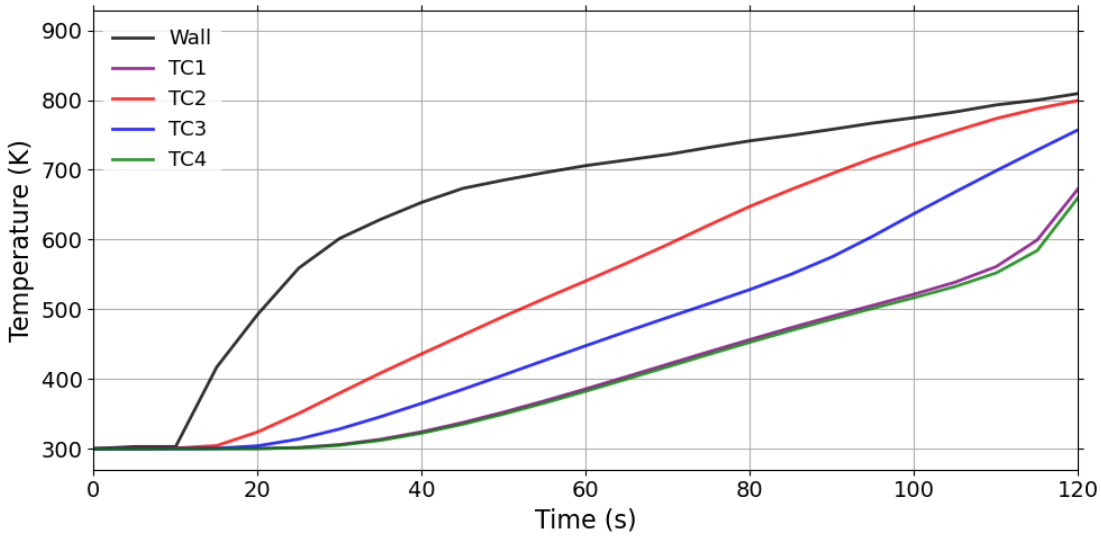

Figure 4.16. Temporal evolution of the temperature in the Wood

4.2 2D tutorials

Figure 4.17 shows the architecture of the 2D tutorials. The inputs and expected outputs of the 2D tutorials are described below.

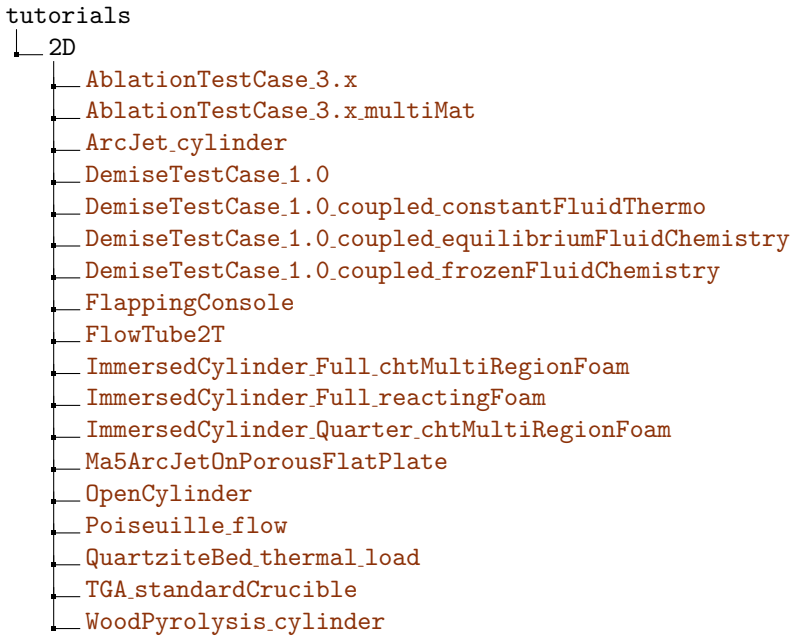

Figure 4.17. 2D tutorials architecture

4.2.1 AblationTestCase_3.x

The **AblationTestCase_3.x** tutorial computes the 2D thermal response of the TACOT material (Fig. 4.18 and 4.19). Listing 4.23 shows the pressure boundary field of **2D-axi_pressureMap BoundaryMapping** type. Listing 4.24 and 4.25 show examples of the *fluxFactorMap and BoundaryConditions* files. The pressure boundary field values follow the Equation 4.19 during the first 0.1 s at a distance up to 70 μm from **fluxFactorCenter** in the **fluxFactorNormal** direction.

$$p(x,t) = \left(\frac{10132.5 - 405.3}{0.1}\, t + 405.3\right)\left[1 - \frac{1 - 0.998}{0.00007}\, x\right] \tag{4.19}$$

$$\mathrm{t} = [0, 0.1], \quad \mathrm{x} = [0, 7 \cdot 10^{-5}] \tag{4.20}$$

Listing 4.23. "origin.0/porousMat/p" file

```
boundaryField{
    top {
        type boundaryMapping;
        mappingType "2D-axi_pressureMap";
        mappingFileName
        "constant/porousMat/BoundaryConditions";
        mappingFields ((p "1"));
        fluxFactorNormal (0 -1 0);
        fluxFactorCenter (0.0 0.1 0.0);
        fluxFactorProjection no;
```

```
        fluxFactorMapFileName
        "constant/porousMat/fluxFactorMap";
        dynamicPressureFieldName p_dyn;
        pointMotionDirection (0 -1 0);
        fluxFactorThreshold
        fluxFactorThreshold [ 0 0 0 0 0 0 0 ] 0.97;
        value uniform 405.3;
    }
}
```

Listing 4.24. Example of the "constant/porousMat/BoundaryConditions" file

```
// t(s)    p_total_w(Pa)
    0         405.3
    0.1       10132.5
```

Listing 4.25. Example of the "constant/porousMat/fluxFactorMap" file

```
//   d(m)          fluxFactor        pressureFactor
     0                  1                 1
     0.00007            1                 0.998
```

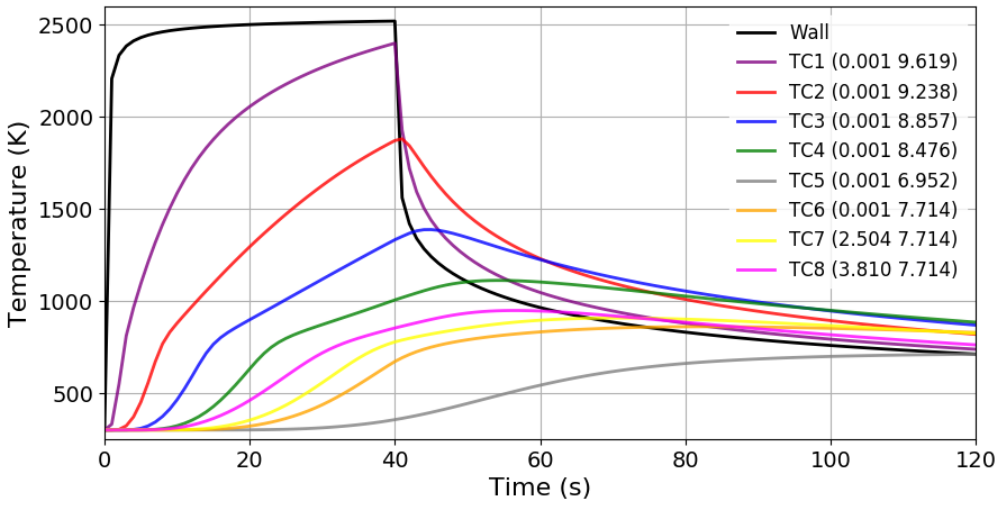

Figure 4.18. Thermal response of the **AblationTestCase_3.x**

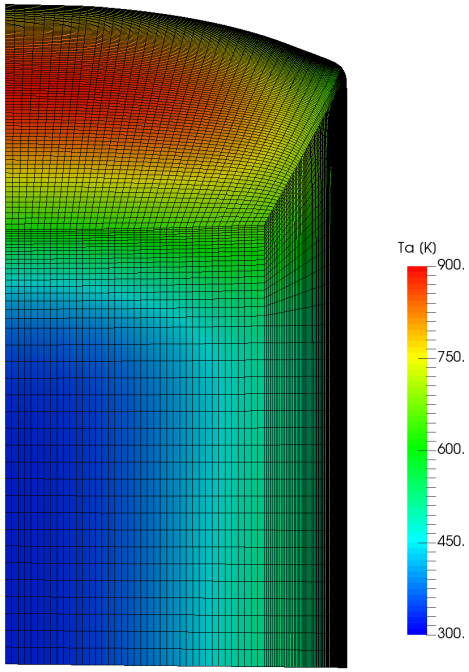

Figure 4.19. 2D Ta profile of the **AblationTestCase_3.x** (120 s)

4.2.2 AblationTestCase_3.x_multiMat

The **AblationTestCase_3.x_multiMat** tutorial computes the 2D thermal response of 1 porous and 1 non-porous materials. Listing 4.26 shows the *regionProperties* file that selects the 2 different materials: porousMat and subMat1 (Fig. 4.20). Table 4.11 shows the summary of the subMat1 material sub-models used in this case. Listing 4.27 shows the *subMat1Properties* file and the different user material sub-models. Listing 4.28 shows the temperature boundary field of "radiative" type (subsection 2.11.11). The radiative heat flux (q_{rad} = 0 W · m^{-2}) and the background temperature (T_∞ = 300 K) are constant during the simulation. The heat transfer coefficient and the recovery enthalpy varies linearly over time following the *BoundaryConditions* file.

Listing 4.26. "constant/regionProperties" file

```
1 regions (solid (porousMat subMat1));
```

Table 4.11. Summary of the subMat1 material sub-models

Model/Tutorial	AblationTestCase_3.x_multiMat
Energy	**PureConduction**
Material Properties	**Fourier**

Listing 4.27. "constant/subMat1/subMat1Properties" file

```
1 Energy {
2   EnergyType PureConduction;
3 }
4 MaterialProperties {
5   MaterialPropertiesType Fourier;
6   MaterialPropertiesDirectory
```

```
7     "$PATO_DIR/data/Materials/Fourier/FourierTemplate";
8 }
```

Listing 4.28. "origin.0/subMat1/Ta" file

```
1  boundaryField {
2      top {
3          type radiative;
4          mappingType constant;
5          mappingFileName
6          "constant/subMat1/BoundaryConditions";
7          mappingFields ((rhoeUeCH "1") (h_r "2")
8          );
9          qRad 0;
10         Tbackground 300;
11         value uniform 300;
12     }
13 }
```

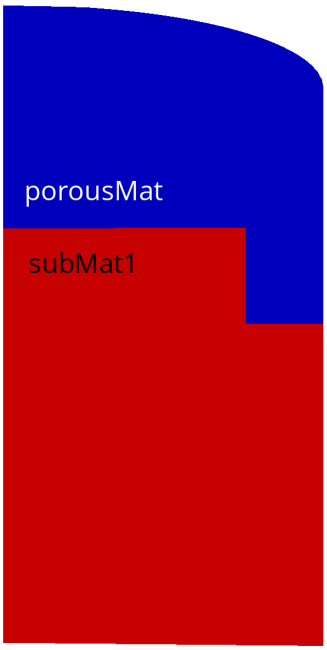

Figure 4.20. Regions of the **AblationTestCase_3.x_multiMat**

4.2.3 ArcJet_cylinder

The **ArcJet_cylinder** tutorial computes the 2D thermal response of the TACOT material (Fig. 4.21). This case uses the same inputs than **AblationTestCase_2.x**.

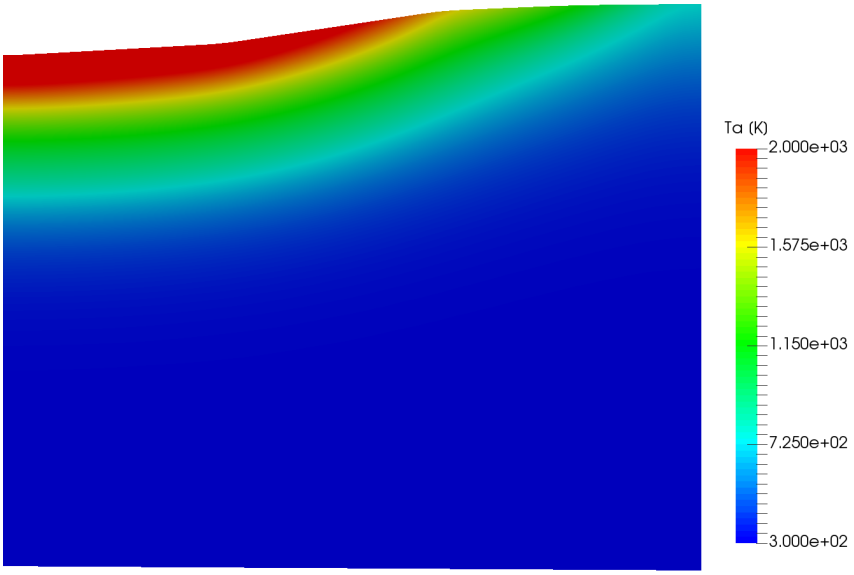

Figure 4.21. 2D Ta profile of the **Arcjet_cylinder** (120 s)

4.2.4 DemiseTestCase_1.0

The **DemiseTestCase_1.0** tutorial computes the 2D axisysmetric thermal response of AISI 316L stainless steel mounted on a cork material (Fig. 4.22 and 4.23). Listing 4.29 shows the temperature coupling boundary condition inside the **demiseMat**. Table 4.12 shows the summary of the **demiseMat** material sub-models used in this case. Table 4.13 presents the summary of the **porousMat** material submodels.

Listing 4.29. "origin.0/demiseMat/Ta" file

```
boundaryField{
  demiseMat_to_porousMat
  {
    type            compressible::turbulentTemperatureCoupledBaffleMixed;
    value           uniform 310;
    Tnbr        Ta;
    kappaMethod     lookup;
    kappa           k_abl_sym;
  }
}
```

Table 4.12. Summary of the demiseMat sub-models

Model/Tutorial	DemiseTestCase_1.0: demiseMat
IO	**LinearInterpolation**
Energy	**PureConduction**
Material Properties	**Fourier**

Table 4.13. Summary of the porousMat sub-models

Model/Tutorial	DemiseTestCase_1.0: porousMat
IO	**LinearInterpolation**
Pyrolysis	**LinearArrhenius**
Mass	**DarcyLaw**
Energy	**Pyrolysis**
Gas Properties	**Tabulated**
Material Properties	**Porous**
TimeControl	**GradP**

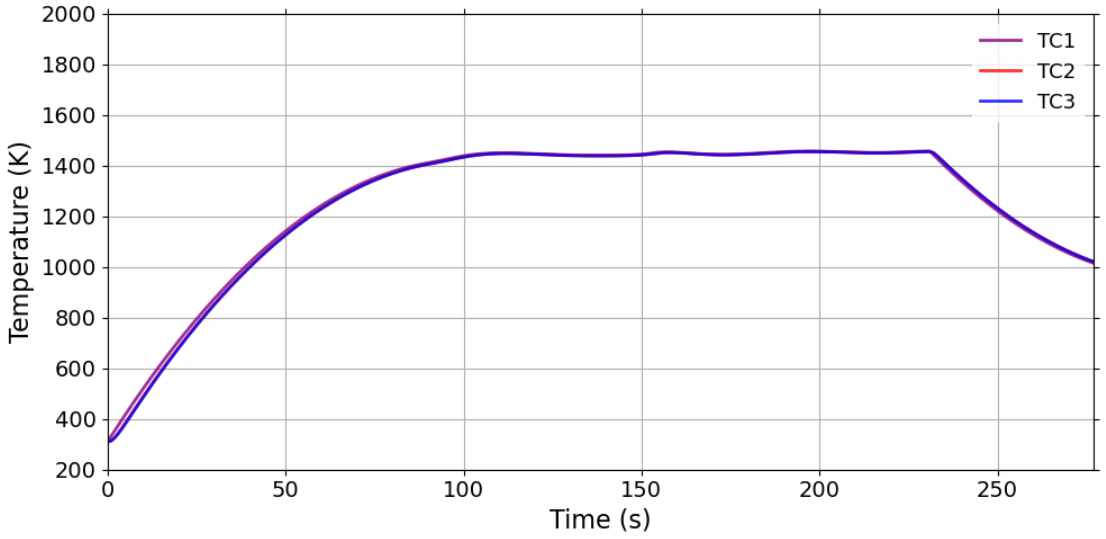

Figure 4.22. Thermal response of the **DemiseTestCase_1.0**

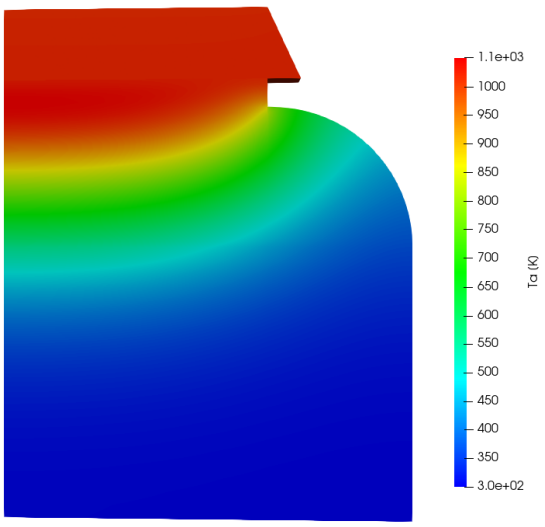

Figure 4.23. 2D Ta profile of the **DemiseTestCase_1.0** (276 s)

4.2.5 DemiseTestCase_1.0_coupled_constantFluidThermo

The **DemiseTestCase_1.0_coupled_constantFluidThermo** tutorial computes the 2D axisymetric thermal response of AISI 316L stainless steel protected by a silicon carbide cap and a porous cork holder inside the hot gas flow (Fig. 4.24 and 4.25). Listing 4.30 shows the temperature coupling boundary condition between the demise material and the hot gas flow. Listing 4.31 presents the *setCase* file. Tables 4.14, 4.15, and 4.16 shows the summary of the **ceramicMat**, **demiseMat**, and **porousMat** respectively.

Listing 4.30. "origin.0/demiseMat/Ta" file

```
boundaryField{
  demiseMat_to_hotFlow
  {
    type              compressible::turbulentTemperatureRadCoupledMixed;
    Tnbr              T;
    kappaMethod       lookup;
    kappa         k_abl_sym;
    QrNbr             none;
    Qr                Qr;
    value             uniform 310;
  }
}
```

Listing 4.31. Example of the "constant/setCase" file

```
fluidType chtMultiRegionFoam;
```

Table 4.14. Summary of the ceramicMat sub-models

Model/Tutorial	DemiseTestCase_1.0_coupled_constantFluidThermo: ceramicMat
IO	**LinearInterpolation**
Energy	**PureConduction**
Material Properties	**FourierRadiation**

Table 4.15. Summary of the demiseMat sub-models

Model/Tutorial	DemiseTestCase_1.0_coupled_constantFluidThermo: demiseMat
IO	LinearInterpolation
Energy	PureConduction

Table 4.16. Summary of the porousMat sub-models

Model/Tutorial	DemiseTestCase_1.0_coupled_constantFluidThermo: porousMat
IO	LinearInterpolation
Pyrolysis	LinearArrhenius
Mass	DarcyLaw
Energy	Pyrolysis
Gas Properties	Tabulated
Material Properties	Porous
TimeControl	GradP

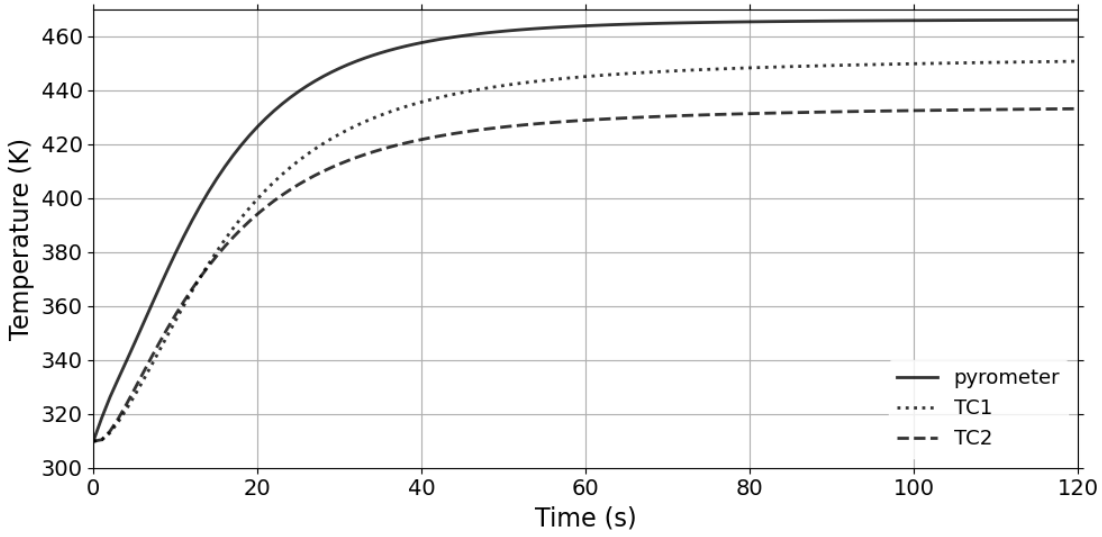

Figure 4.24. Thermal response of the **DemiseTestCase_1.0_coupled_constantFluidThermo**

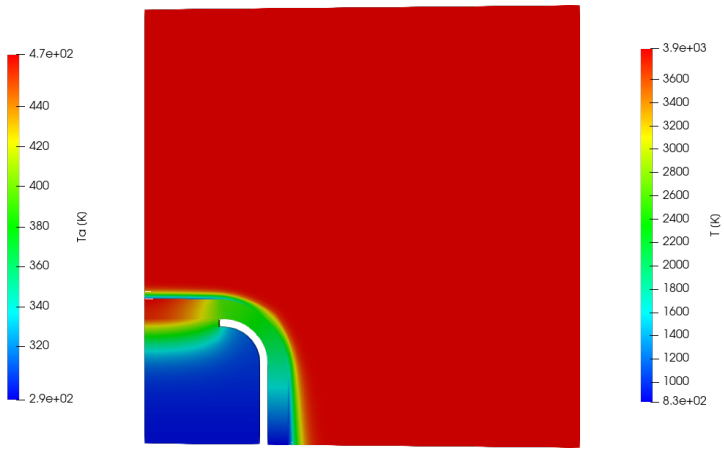

Figure 4.25. 2D Ta profile of the **DemiseTestCase_1.0_coupled_constantFluidThermo** (120 s)

4.2.6 DemiseTestCase_1.0_coupled_equilibriumFluidChemistry

The **DemiseTestCase_1.0_coupled_equilibriumFluidChemistry** tutorial computes the 2D axisymetric thermal response of AISI 316L stainless steel protected by a silicon carbide cap and a porous cork holder inside the hot gas flow in chemical equilibrium (Fig. 4.26 and 4.29). Listing 4.32 shows the *thermophysicalProperties* file for the **hotFlow**.

Listing 4.32. "constant/hotFlow/thermophysicalProperties" file

```
thermoType {
    type              heRhoThermo;
    mixture           pureMixture;
    transport         tabular;
    thermo            hTabular;
    equationOfState   Rhotabular;
    specie            specie;
    energy            sensibleEnthalpy;
}

mixture {
    specie
    {
        nMoles          1;
        molWeight       28.96;
    }

    equationOfState
    {
        file "$PATO_DIR/data/Fluids/fluidPropertyTable/Air5/Equilibrium/densityTable";
        outOfBounds clamp;
    }

    thermodynamics
    {
```

```
26      Hf 0;
27      Cp
28      {
29        file "$PATO_DIR/data/Fluids/fluidPropertyTable/Air5/Equilibrium/cpTable";
30        outOfBounds clamp;
31      }
32      h
33      {
34        file "$PATO_DIR/data/Fluids/fluidPropertyTable/Air5/Equilibrium/hTable";
35        outOfBounds clamp;
36      }
37    }
38
39    transport
40    {
41      mu
42      {
43        file "$PATO_DIR/data/Fluids/fluidPropertyTable/Air5/Equilibrium/muTable";
44        outOfBounds clamp;
45      }
46
47      kappa
48      {
49        file "$PATO_DIR/data/Fluids/fluidPropertyTable/Air5/Equilibrium/kappaTable";
50        outOfBounds clamp;
51      }
52    }
53  }
```

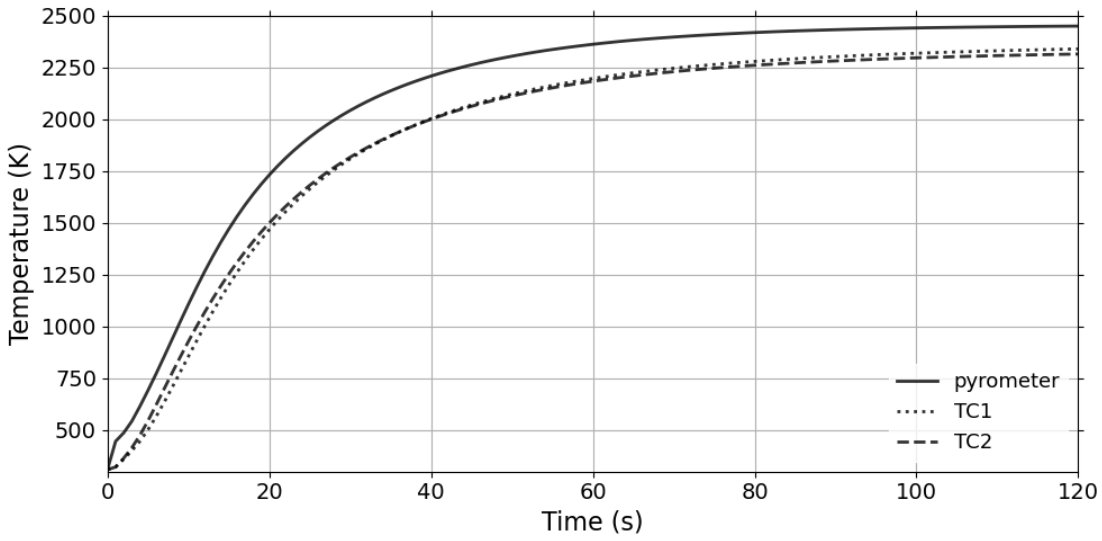

Figure 4.26. Thermal response of the **DemiseTestCase_1.0_coupled_equilibriumFluidChemistry**

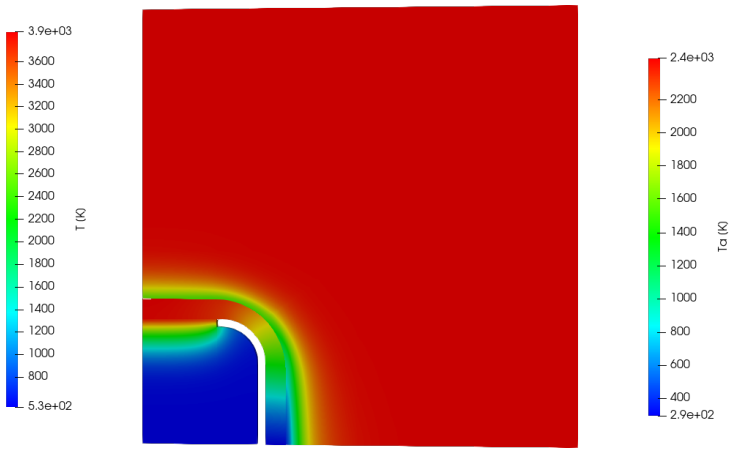

Figure 4.27. 2D Ta profile of the **DemiseTestCase_1.0_coupled_equilibriumFluidChemistry** (120 s)

4.2.7 DemiseTestCase_1.0_coupled_frozenFluidChemistry

The **DemiseTestCase_1.0_coupled_frozenFluidChemistry** tutorial computes the 2D axisymetric thermal response of AISI 316L stainless steel protected by a silicon carbide cap and a porous cork holder inside the chemically frozen hot gas flow (Fig. 4.28 and 4.29).

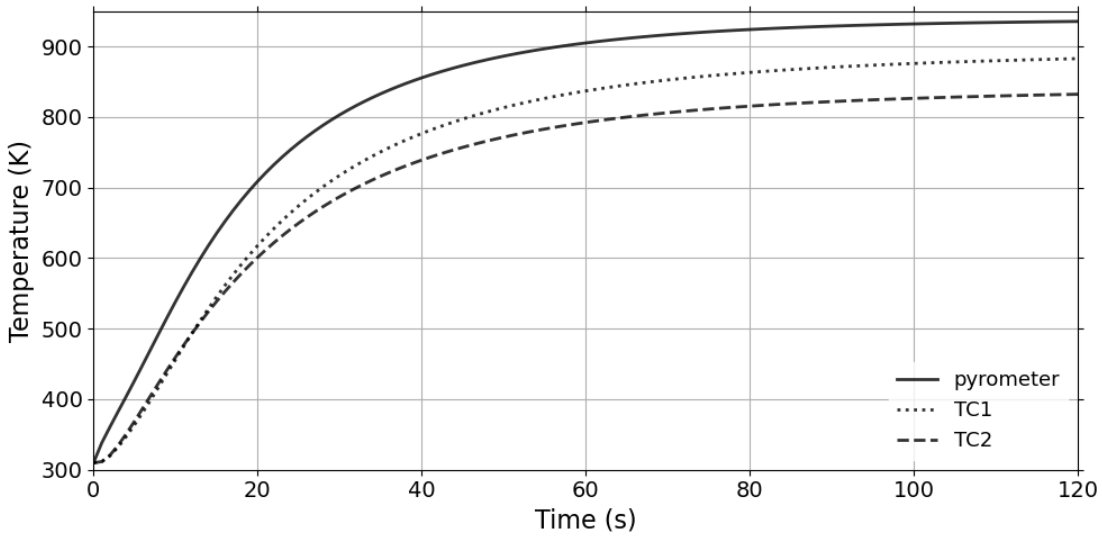

Figure 4.28. Thermal response of the **DemiseTestCase_1.0_coupled_frozenFluidChemistry**

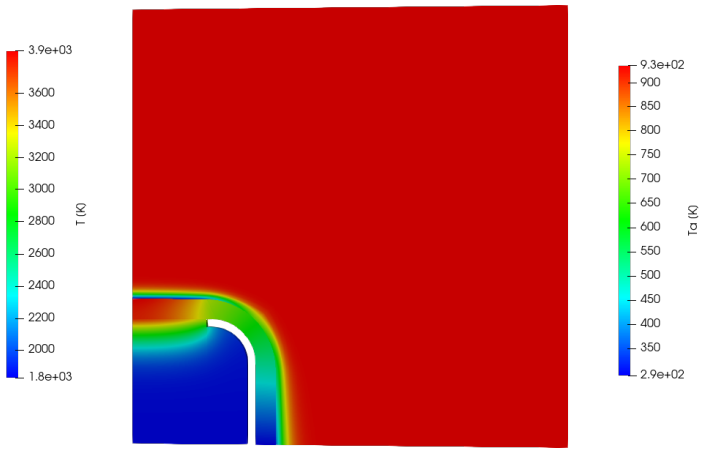

Figure 4.29. 2D Ta profile of the **DemiseTestCase_1.0_coupled_frozenFluidChemistry** (120 s)

4.2.8 FlappingConsole

The **FlappingConsole** tutorial computes the fluid solid interaction response of a solid console immersed into a flow (Fig. 4.30). Listing 4.33 shows the fluid_to_solid **cellMotionU** boundary condition, and listing 4.34 presents the **solid_to_fluid D** boundary condition.

Listing 4.33. "origin.0/fluid/cellMotionU" file

```
boundaryField{
    fluid_to_solid
    {
        type            codedFixedValue;
        value           uniform (0 0 0);
        name            solidfollowing;
        code
        #{
            vectorField n(this->patch().nf());
            scalar deltaT = this->db().time().deltaTValue();
            const mappedPatchBase& mpp =
                refCast<const mappedPatchBase>(this->patch().patch());
            const polyMesh& nbrMesh = mpp.sampleMesh();

            const label samplePatchI = mpp.samplePolyPatch().index();
            const fvPatch& nbrPatch =
                refCast<const fvMesh>(nbrMesh).boundary()[samplePatchI];

            const fvPatchField<vector>& nbrField =
                nbrPatch.lookupPatchField
                <
                    GeometricField<vector, fvPatchField, volMesh>,
                    vector
                >
                ("deltaD");

```

```
27              tmp<Field<vector>> nbrIntFld
28              (
29                  new Field<vector>(nbrField.size(), pTraits<vector>::zero)
30              );
31              nbrIntFld.ref() = nbrField.patchInternalField();
32              vectorField deltaD = nbrIntFld.ref();
33              operator==(deltaD/deltaT);
34          #};
35
36          codeInclude
37          #{
38              #include "mappedPatchBase.H"
39          #};
40
41          codeOptions
42          #{
43              -I$(LIB_SRC)/meshTools/lnInclude
44          #};
45      }
46  }
```

Listing 4.34. "origin.0/solid/D" file

```
1   boundaryField{
2       fluid_to_solid
3       {
4           type            codedFixedValue;
5           value           uniform (0 0 0);
6           name            solidfollowing;
7           code
8           #{
9               vectorField n(this->patch().nf());
10              scalar deltaT = this->db().time().deltaTValue();
11              const mappedPatchBase& mpp =
12                  refCast<const mappedPatchBase>(this->patch().patch());
13              const polyMesh& nbrMesh = mpp.sampleMesh();
14
15              const label samplePatchI = mpp.samplePolyPatch().index();
16              const fvPatch& nbrPatch =
17                  refCast<const fvMesh>(nbrMesh).boundary()[samplePatchI];
18
19              const fvPatchField<vector>& nbrField =
20                  nbrPatch.lookupPatchField
21                  <
22                      GeometricField<vector, fvPatchField, volMesh>,
23                      vector
24                  >
25                  ("deltaD");
26
27              tmp<Field<vector>> nbrIntFld
28              (
29                  new Field<vector>(nbrField.size(), pTraits<vector>::zero)
30              );
31              nbrIntFld.ref() = nbrField.patchInternalField();
32              vectorField deltaD = nbrIntFld.ref();
```

```
33              operator==(deltaD/deltaT);
34          #};
35
36          codeInclude
37          #{
38              #include "mappedPatchBase.H"
39          #};
40
41          codeOptions
42          #{
43              -I$(LIB_SRC)/meshTools/lnInclude
44          #};
45      }
46  }
```

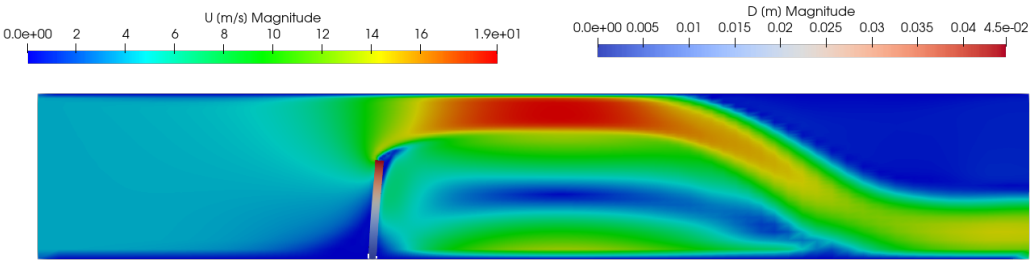

Figure 4.30. Velocity of the fluid and displacement of the solid for the **FlappingConsole** case.

4.2.9 FlowTube2T

the **FlowTube2T** tutorial computes the 2D thermal response of a porous medium heated by a gas flow (Fig. 4.31). Table 4.17 shows the summary of the **porousMat** material used in this case.

Table 4.17. Summary of the porousMat sub-models

Model/Tutorial	FlowTube2D: porousMat
IO	no
Pyrolysis	virgin
Mass	DarcyLaw2T
Energy	Pyrolysis2T
Gas Properties	Tabulated2T
Material Properties	Porous
TimeControl	GradP

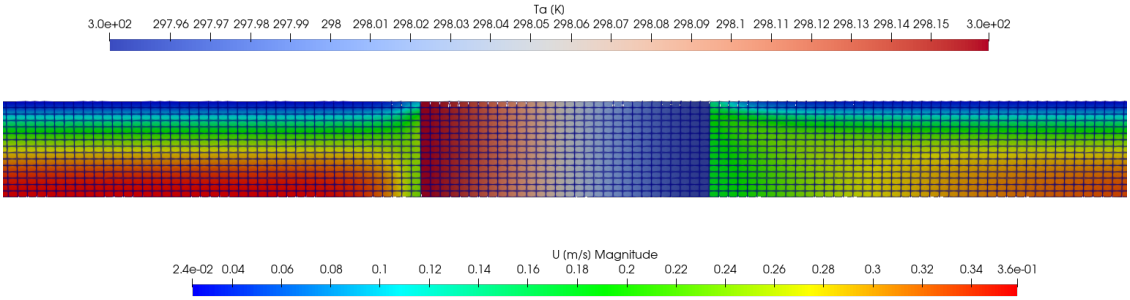

Figure 4.31. Velocity of the fluid and thermal response of the porous material for the **FlowTube2T** case.

4.2.10 ImmersedCylinder_Full_chtMultiRegionFoam

The **ImmersedCylinder_Full_chtMultiRegionFoam** tutorial computes the 2D thermal response of TACOT material inside a hot gas flow (Fig 4.32 and 4.33). Listing 4.35 shows the pressure coupling boundary condition between the porous material and the hot gas flow.

Listing 4.35. "origin.0/porousMat/p" file

```
1 boundaryField{
2 porousMat_to_flow
3   {
4     type          fixedValueToNbrValue;
5     nbr           p;
6     value         $internalField;
7   }
8 }
```

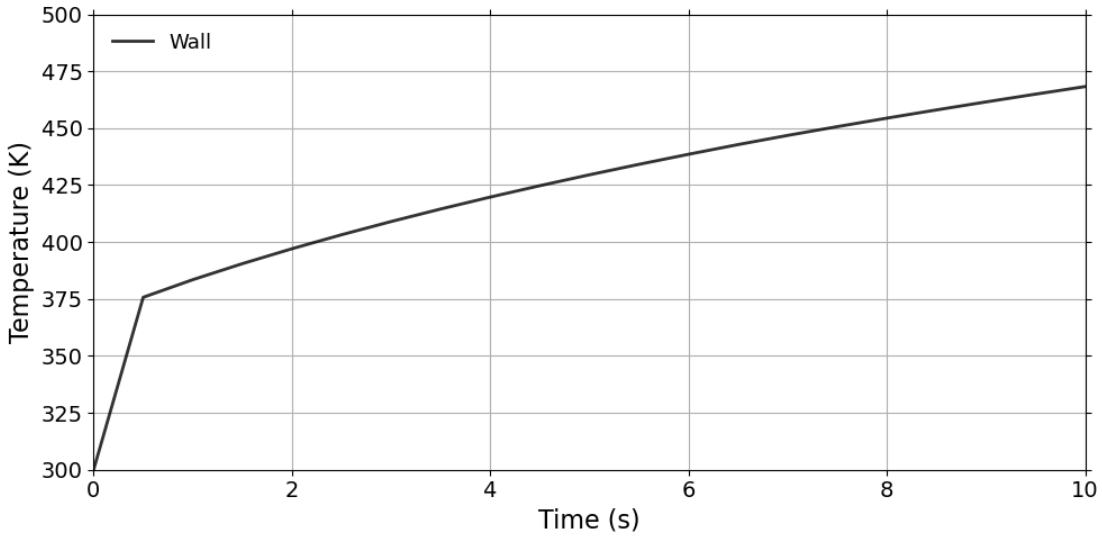

Figure 4.32. Thermal response of the **ImmersedCylinder_Full_chtMultiRegionFoam**

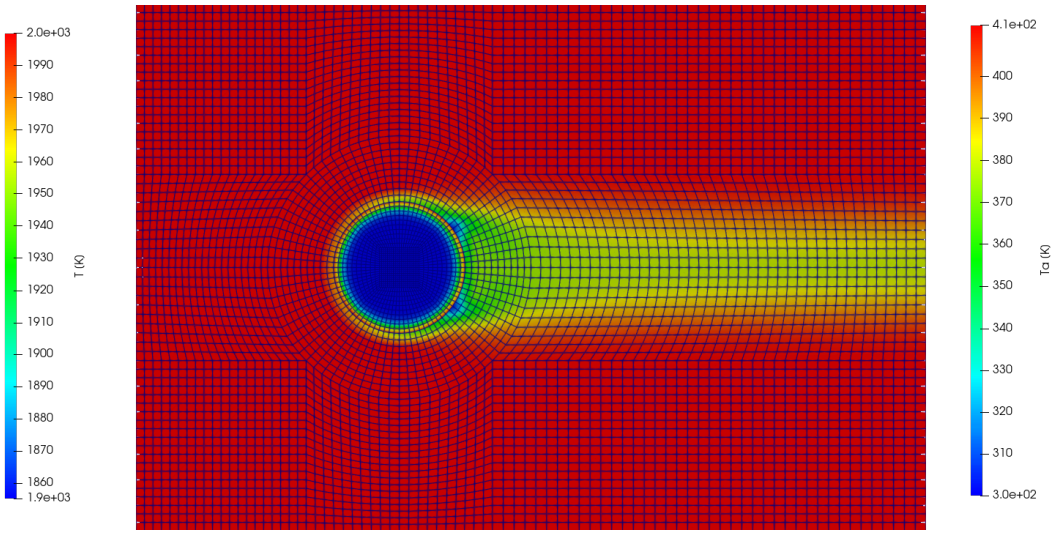

Figure 4.33. 2D temperature profile of the **ImmersedCylinder_Full_chtMultiRegionFoam** (10 s)

4.2.11 ImmersedCylinder_Full_reactingFoam

The **ImmersedCylinder_Full_reactingFoam** tutorial computes the 2D thermal response of TACOT material inside a reactive hot gas flow (Fig. 4.34 and 4.35). Listing 4.36 shows the *thermophyscialProperties* file for the **flow**.

Listing 4.36. "constant/flow/thermophysicalProperties" file

```
thermoType {
    type              heRhoThermo;
    mixture           reactingMixture;
    transport         sutherland;
    thermo            janaf;
    energy            sensibleEnthalpy;
    equationOfState   perfectGas;
    specie            specie;
}

inertSpecie CO2;

chemistryReader foamChemistryReader;

foamChemistryFile "$FOAM_CASE/constant/flow/reactions";

foamChemistryThermoFile "$FOAM_CASE/constant/flow/thermo.compressibleGas";
```

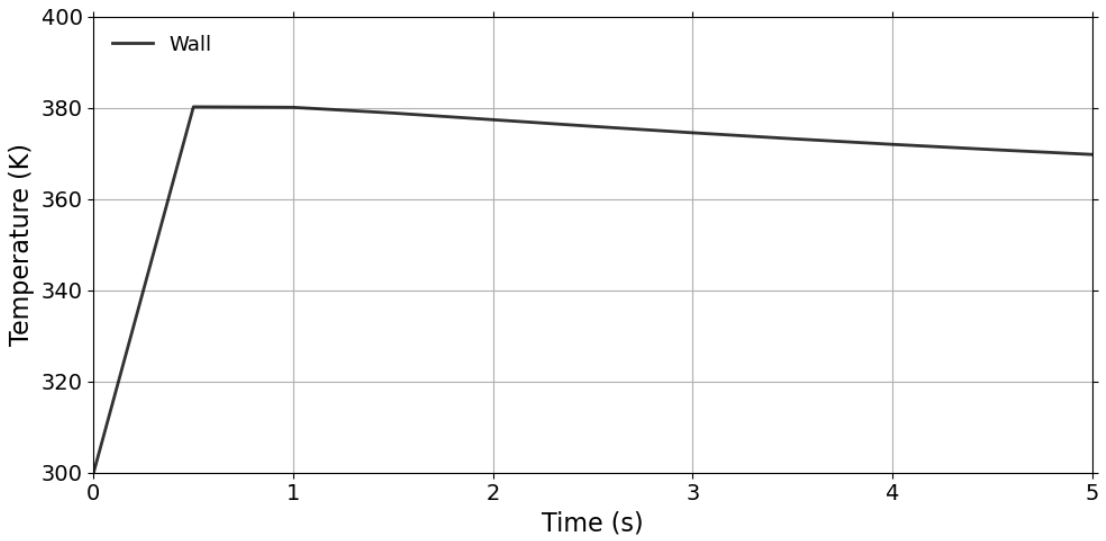

Figure 4.34. Thermal response of the **ImmersedCylinder_Full_reactingFoam**

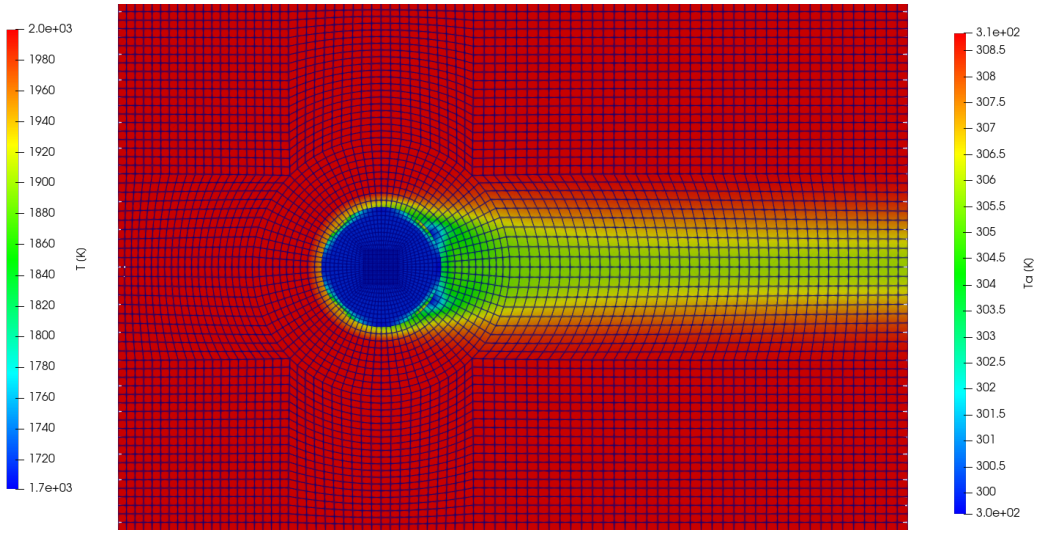

Figure 4.35. 2D temperature profile of the **ImmersedCylinder_Full_reactingFoam** (5 s)

4.2.12 ImmersedCylinder_Quarter_chtMultiRegionFoam

The **ImmersedCylinder_Quarter_chtMultiRegionFoam** tutorial computes the 2D symmetric thermal response of TACOT material inside a hot gas flow (Fig 4.36 and 4.37).

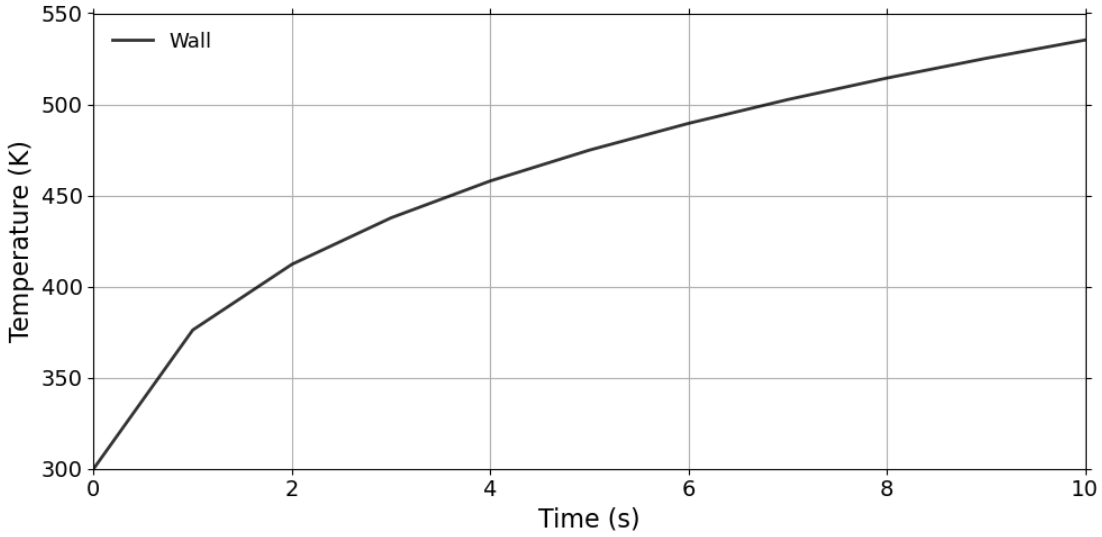

Figure 4.36. Thermal response of the **ImmersedCylinder_Quarter_chtMultiRegionFoam**

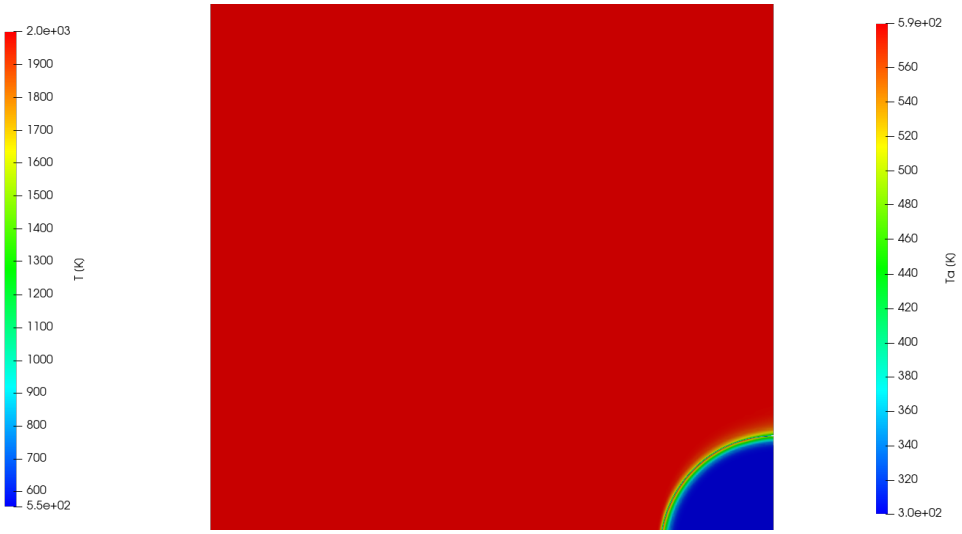

Figure 4.37. 2D temperature profile of the **ImmersedCylinder_Quarter_chtMultiRegionFoam** (10 s)

4.2.13 Ma5ArcJetOnPorousFlatPlate

The **Ma5ArcjetOnPorousFlatPlate** computes the 2D thermal response of a plate of TACOT material inside a supersonic ($Ma = 5$) arc jet (Figure 4.38 and 4.39). It is possible to run simulations on each zone (**fluid** or **porousMaterial**) separately for uncoupled simulations or on both mesh simultaneously for coupled simulations. 4 *Allrun* files are available:

- *Allrun_aero*: run the Mach-5 flow simulation alone on a single processor with **rhoCentralFoam** solver from OpenFOAM under frozen chemistry assumption.
- *Allrun_parallel_aero*: run the Mach-5 flow simulation in parallel with **rhoCentralFoam** solver from OpenFOAM. The simulation takes 8 hours on 192 processors to reach the final time of 0.004 s.
- *Allrun_coupled_standard*: run the coupled simulation using a 2 temperatures model for the porous material. The flow is initialized with a converged solution.
- *Allrun_coupled_stitching*: run the coupled simulation with a time scale splitting strategy to speed-up convergence. This method use the experimental solver **PATOxs**.

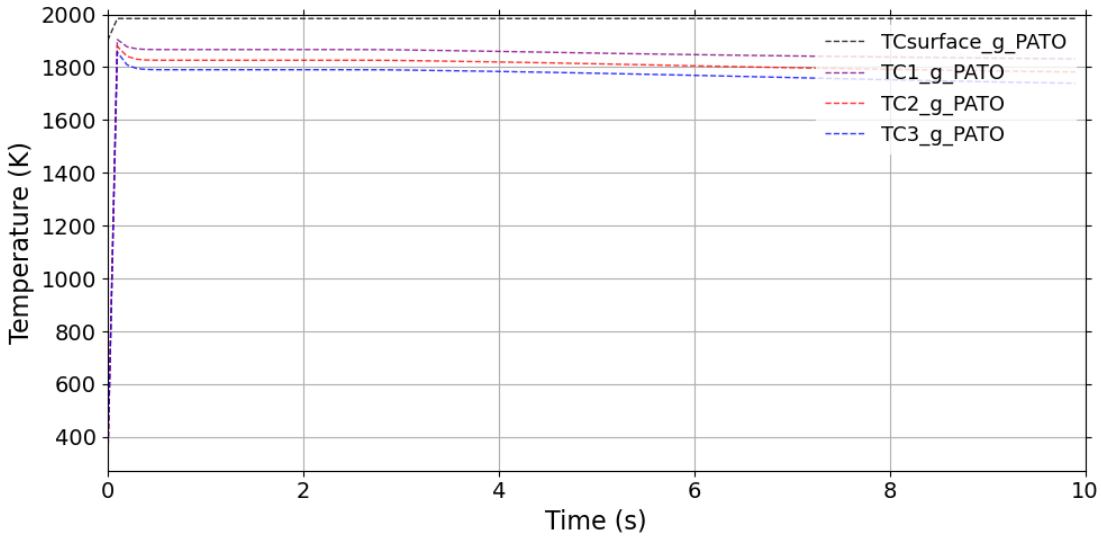

Figure 4.38. Gas thermal response of the **Ma5ArcJetOnPorousFlatPlate**

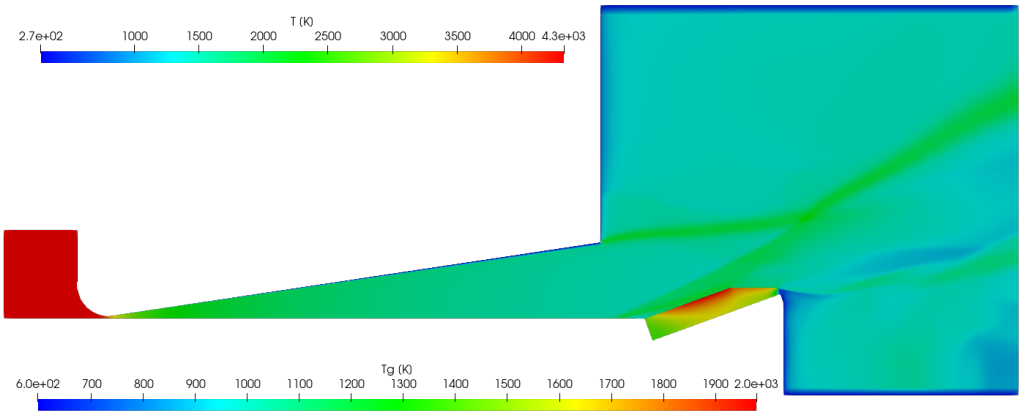

Figure 4.39. 2D gas temperature profile of the **Ma5ArcJetOnPorousFlatPlate**

4.2.14 OpenCylinder

The **OpenCylinder** tutorial compute the 2D thermo-mechanical response of a solid cylinder (Fig 4.40 and 4.41). A computation of the analytical solution is included in the tutorial (Eq. 4.21 to Eq. 4.24).

$$T = \frac{(T_{in} - T_{out})}{ln\left(\frac{0.7}{0.5}\right)}\ ln\left(\frac{0.7}{r}\right) \tag{4.21}$$

$$\sigma_R = \frac{\alpha\ E\ (Ti_{in} - T_{out})}{2(1-\nu)\ ln\left(\frac{0.7}{0.5}\right)}\ \left(-ln\left(\frac{0.7}{r}\right) - \frac{0.5^2}{0.7^2 - 0.5^2}\ \left(1 - \frac{0.7}{r}\right) ln\left(\frac{0.7}{0.5}\right)\right) \tag{4.22}$$

$$\sigma_\Theta = \frac{\alpha\ E\ (Ti_{in} - T_{out})}{2(1-\nu)\ ln\left(\frac{0.7}{0.5}\right)}\ \left(1 - ln\left(\frac{0.7}{r}\right) - \frac{0.5^2}{0.7^2 - 0.5^2}\ \left(1 + \frac{0.7}{r}\right) ln\left(\frac{0.7}{0.5}\right)\right) \tag{4.23}$$

$$\sigma_z = 0.3\ \ (\sigma_R + \sigma_\Theta) - E\ \alpha\ T \tag{4.24}$$

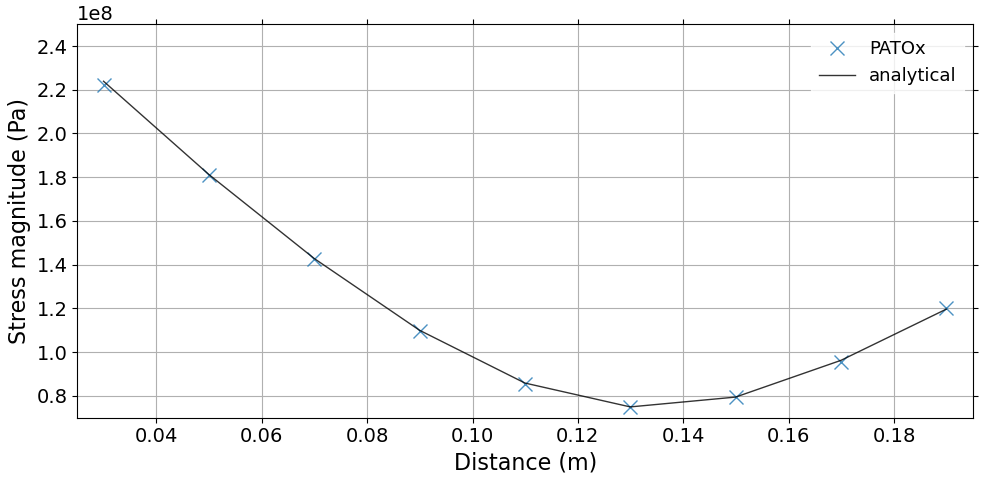

Figure 4.40. Stress response of the **OpenCylinder**

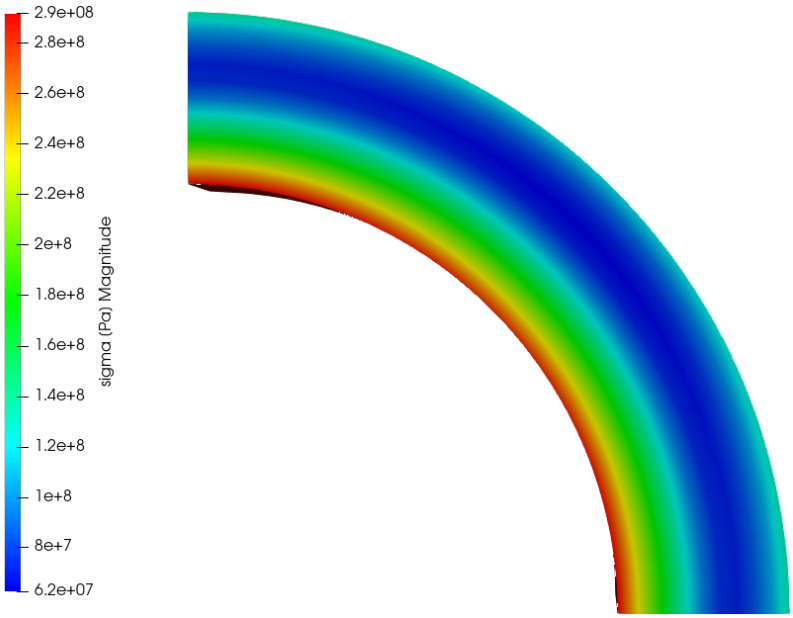

Figure 4.41. 2D stress magnitude profile of the **OpenCylinder**

4.2.15 Poiseuille_flow

The **Poiseuille_flow** tutorial computes the 2D axisymetric, incompressible, laminar Poiseuille flow and compares it with analytical results (Fig 4.42).

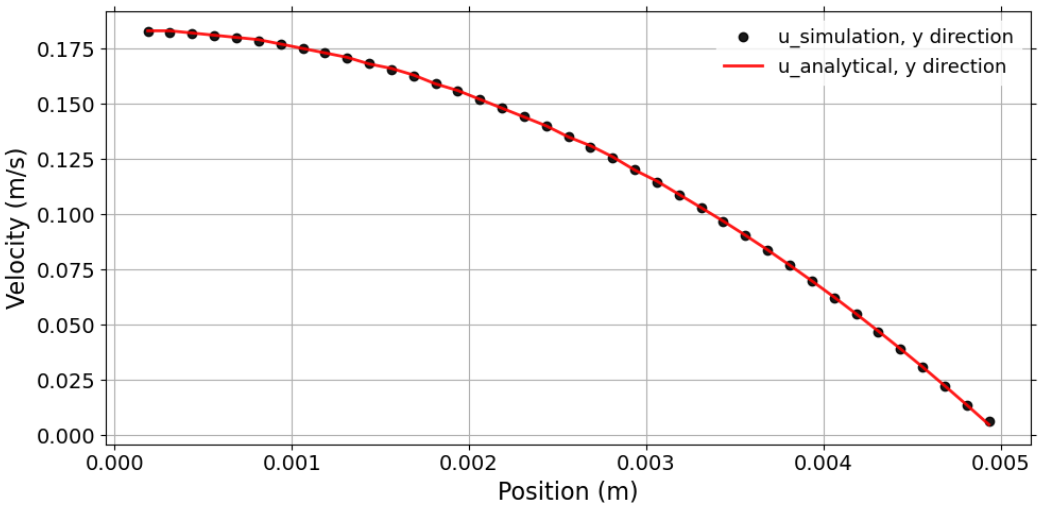

Figure 4.42. Velocity of the flow for the **Poiseuille_flow** tutorial

4.2.16 QuartziteBed_thermal_load

The **QuartziteBed_thermal_load** tutorial computes the 2D axisymetric thermo-mechanical response of the Quartzite bed material enclosed in a 16Mo3 steel (Fig 4.43). The 2 temperature thermal load is pre-calculated in the **thermalLoad** directory. Table 4.18 shows the summary of the **porousMat** material used in this case and Table 4.19 presents the summary of the **tank** material used in this case. Mechanical contact between the two material is not taken into account. There are only separated by difference in thermo-mechanical properties

Table 4.18. Summary of the porousMat sub-models

Model/Tutorial	QuartziteBed_thermal_load: porousMat
IO	**no**
Pyrolysis	**virgin**
Material Properties	**Porous**
SolidMechanics	**ElasticThermal**

Table 4.19. Summary of the tank sub-models

Model/Tutorial	QuartziteBed_thermal_load: tank
IO	**no**
Pyrolysis	**virgin**
Material Properties	**Porous_const**
SolidMechanics	**ElasticThermal**

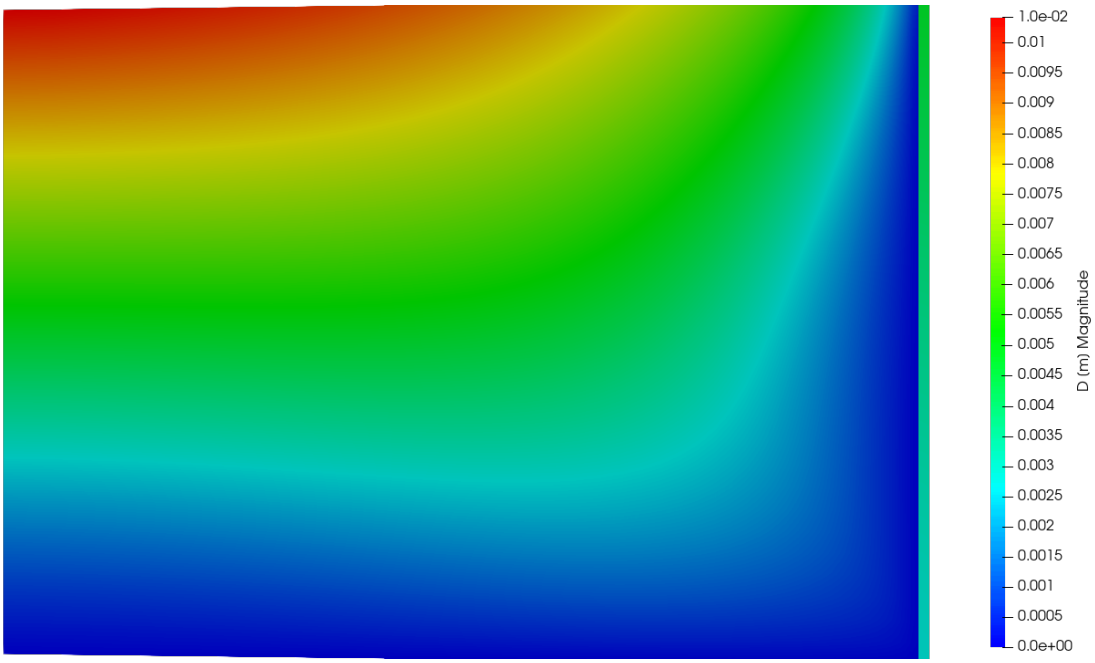

Figure 4.43. 2D displacement profile of the **QuartziteBed_thermal_load**

4.2.17 TGA_standardCrucible

The **TGA_standardCrucible** tutorial computes the 2D thermal response of the TACOT material (Fig. 4.44). This case uses the same inputs as **AblationTestCase_2.x**.

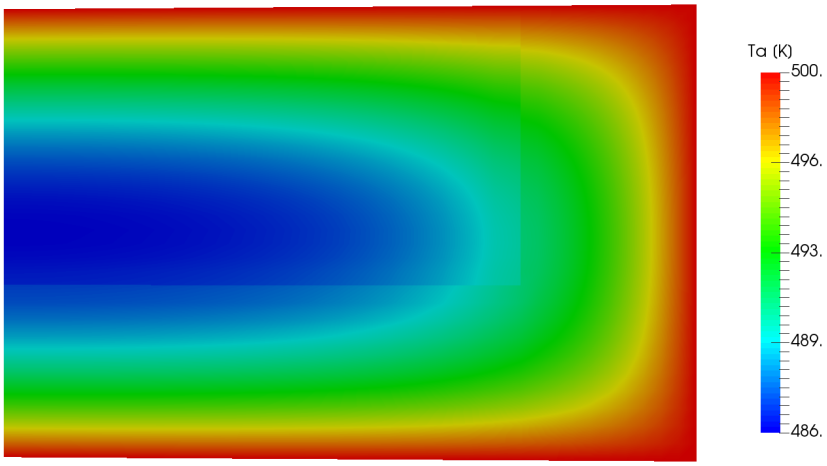

Figure 4.44. 2D Ta profile of the **TGA_standardCrucible** (10 s)

4.2.18 WoodPyrolysis_cylinder

The **WoodPyrolysis_cylinder** tutorial computes the 2D thermal response of the TACOT material (Fig. 4.45). This case uses the same inputs as **WoodPyrolysis**.

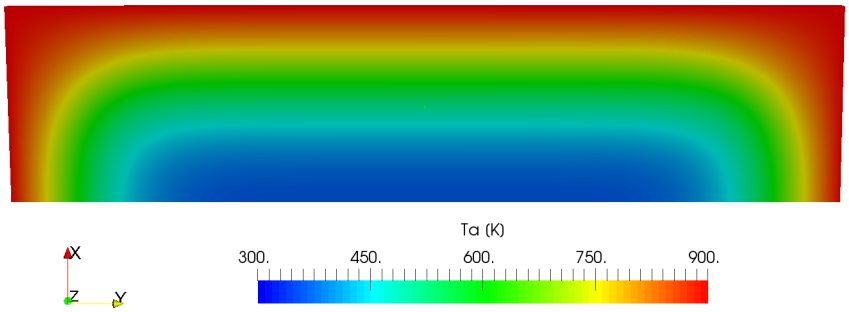

Figure 4.45. 2D Ta profile of the **WoodPyrolysis_cylinder** (30 s)

4.3 3D tutorials

Figure 4.46 shows the architecture of the 3D tutorials. The inputs and expected outputs of the 3D tutorials are described below.

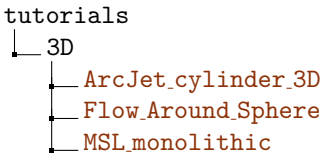

Figure 4.46. 3D tutorials architecture

4.3.1 ArcJet_cylinder_3D

The **Arcjet_cylinder_3D** tutorial computes the 3D thermal response of the TACOT material (Fig. 4.47). This case uses the same inputs as **AblationTestCase_2.x**.

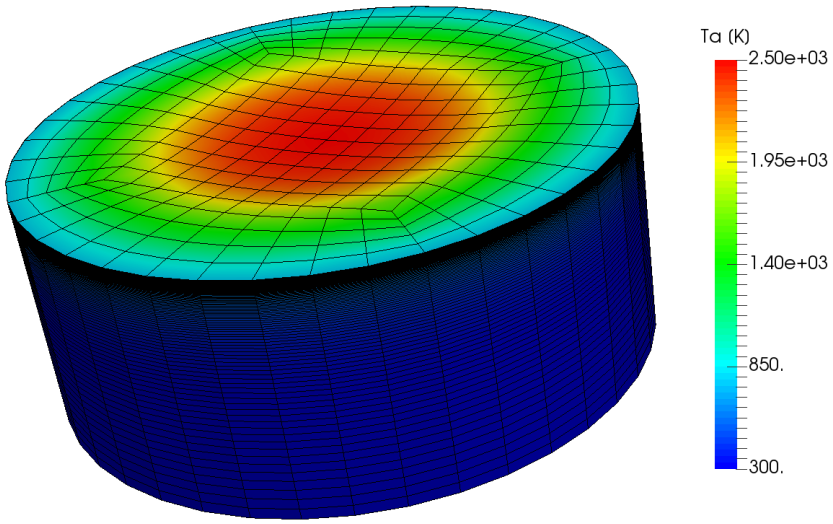

Figure 4.47. 3D material response of the **Arcjet_cylinder** (120 s)

4.3.2 MSL_monolithic

The **MSL_monolithic** tutorial computes the 3D thermal response of the monolithic heat shield of the Mars Science Laboratory (MSL) made of TACOT material during Mars atmosphere entry (Fig. 4.48). Listing 4.37 shows the temperature boundary field of **Bprime** BC type using the **3D-tecplot BoundaryMapping** type. The pressure, the heat transfer coefficient and the recovery enthalpy are linearly interpolated from the *DPLR-MSL_** Tecplot files.

Listing 4.37. "origin.0/porousMat/Ta" file

```
boundaryField {
    top {
        type Bprime;
        BprimeFile
        "$FOAM_CASE/data/Materials/TACOT-Mars/BprimeTable";
        mappingType "3D-tecplot";
        mappingFileName
        "$FOAM_CASE/data/Environment/DPLR-MSL";
        mappingSymmetry y;
        mappingFields ((p "1") (rhoeUeCH "2") (h_r "3"));
        Tbackground 187;
        qRad 0;
        lambda 0.5;
        heatOn 1;
        value uniform 300;
    }
}
```

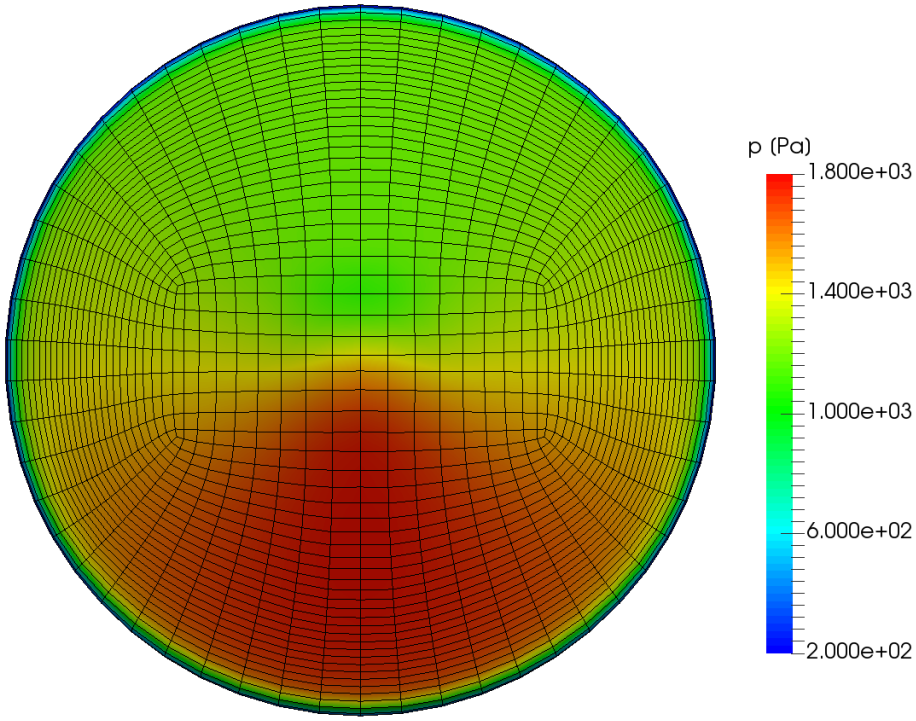

Figure 4.48. 3D material response of the **MSL_monolithic** (100 s)

4.3.3 Flow_Around_Sphere

The **flow_Around_Sphere** tutorial computes the 3D incompressible, laminar flow around a sphere and computes drag and lift coefficients (Fig 4.49).

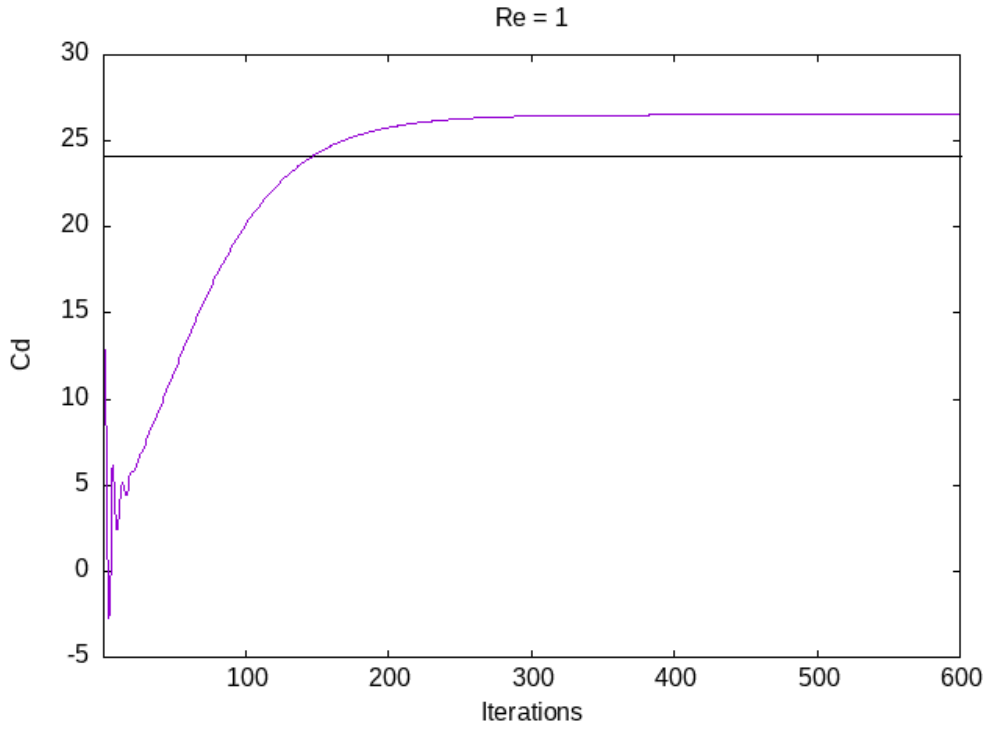

Figure 4.49. Drag coefficient of a sphere from **Flow_Around_Sphere**

Chapter 5

Utilities

Figure 5.1 shows the architecture of **utilities**.

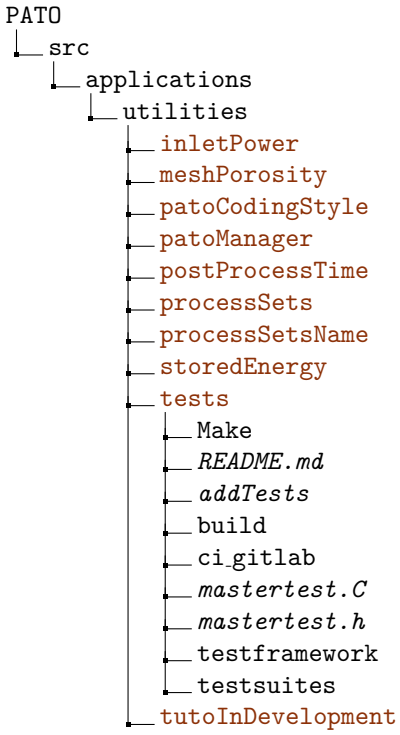

Figure 5.1. utilities architecture

5.1 meshPorosity

meshPorosity is a post-processor that computes and print the total volume, fibers volume, voids volume and porosity. It works considering that the total domain is composed by a porous domain surrounded by two buffer zones. To identify the porous domain, the script search for a patch named **fibres**. This utility also works in the case of a porous domain only (no buffer zones). In this configuration, the **fibres** patch is useless. Listing 5.1 shows **meshPorosity** usage.

Listing 5.1. Mesh porosity usage

```
1   $ meshPorosity
```

5.2 patoManager

Listing 5.2 shows how to print the **patoManager** help.

Listing 5.2. patoManager help

```
$ patoManager -h
Usage: patoManager [OPTIONS]
options:
  -addAllTests       Add all the new test files (*.C) from
                     "$LIB_PATO/src/applications/tests/testsuites"
                     to the unit testing framework
  -addBC <oldBCName/newBCName>
                     Add new BC (copy of old BC)
  -addType <modelName/oldTypeName/newTypeName>
                     Add new type (copy of old type)
  -cleanTuto <tutoDir>
                     Clean the tutorials in <tutoDir>
                     (N.B. use "all" to clean
                     all the tutorials)
  -listModels        List the available models
  -listTypes         List the available models and related types
  -removeBC <BCName>
                     Remove BC
  -removeType <modelName/typeName>
                     Remove type
  -runTest <args>    Run unit test suites
  -runTuto <tutoDir>
                     Run the tutorials in <tutoDir>
                     (N.B. use "all" to run all
                     the tutorials)
  -saveTest <tutoDir>
                     Save the reference tutorials from <tutoDir> to
                     $PATO_DIR/src/applications/tests/
                     testsuites/tutorials/ref
  -showDoc <browser>
                     Show PATO documentation generated by Doxygen
```

5.2.1 Cleaning/running tutorials

Listing 5.3 shows how to run all the tutorials in the tutorials/1D folder. Listing 5.4 shows how to clean all the tutorials in the tutorials/1D folder.

Listing 5.3. Run tutorials

```
$ patoManager -runTuto "1D"
```

Listing 5.4. Clean tutorials

```
$ patoManager -cleanTuto "1D"
```

5.2.2 Adding/removing types

Listing 5.5 shows how to list the different material sub-models. Listing 5.6 shows how to list the different types per material sub-model. Listing 5.7 shows how to add a new type based on a old type of a given material sub-model. Listing 5.7 shows how to remove a type based of a given material sub-model.

Listing 5.5. List of the material sub-models

```
$ patoManager -listModels
  Available models are:
      9
      (
          Energy
          GasProperties
          IO
          Mass
          MaterialChemistry
          MaterialProperties
          Pyrolysis
          TimeControl
          VolumeAblation
      )
```

Listing 5.6. List of the types for each material sub-model

```
$ patoManager -listTypes
  Available types for EnergyModel:
      7
      (
          BoundaryTable
          CorrectBC
          FixedTemperature
          PureConduction
          Pyrolysis
          Pyrolysis_Heterogeneous_SpeciesDiffusion
          no
      )
  ...
```

Listing 5.7. Add a new type of a given material sub-model

```
$ patoManager -addType "Energy/Pyrolysis/newModelName"
```

Listing 5.8. Remove a type of a given material sub-model

```
$ patoManager -removeType "Energy/modelName"
```

5.2.3 Adding boundary conditions

Listing 5.9 shows how to add new boundary conditions based on old ones.

Listing 5.9. Add new boundary conditions

```
$ patoManager -addBC "speciesBC/newBCname"
```

5.2.4 Showing Doxygen documentation

Listing 5.10 shows how to display the PATO Doxygen documentation.

Listing 5.10. Show Doxygen documentation

```
1  $ patoManager -showDoc "firefox"
```

5.3 inletPower

inletPower is a post-processor that computes the power on a specified field. Reference temperature, patch and temperature field name should be provided. You should provide the region name. You can select which time step you want to post process. Listing 5.11 shows **inletPower** usage.

Listing 5.11. inletPower usage

```
1  $ inletPower -Tfield "Ta" -Tref 300 -field "U" "patch"
```

5.4 storedEnergy

storedEnergy is a post-processor that computes the stored energy on solid phase. Reference temperature, temperature field name and region name should be provided. You can select which time step you want to post process. Listing 5.12 shows **storedEnergy** usage.

Listing 5.12. storedEnergy usage

```
1  $ storedEnergy -Tfield "Ta"
```

5.5 postProcessTime

postProcessTime creates a file that contains field values as a function of time from the files in **postProcessing** directory. Listing 5.13 shows **postProcessTime** usage.

Listing 5.13. postProcessTime usage

```
1  $ postProcessTime "dictName" "regionName" "inputFileName" "outputFileName"
```

5.6 processSets

processSets is run after OpenFOAM post-processor **sample** to grab **cloud** data in the **sets** directory and rewrite them in a single file. Currently, it can only extract data from *list_Ta_rho_s* file. Listing 5.14 shows **processSets** usage.

Listing 5.14. processSets usage

```
1  $ processSets
```

5.7 processSetsName

processSetsName is an extended version of **processSets** where you can provide name of the input and output files, number of probe fields, switch to allow moving probes and region name. Listing 5.15 shows **processSetsName** usage.

Listing 5.15. processSetsName usage

```
1   $ processSetsName -npf 2 -movingProbe 0 -region "porousMat" "input" "output"
```

5.8 tutoInDevelopment

tutoInDevelopment is used to send an error message and then stop the run of a tutorial in development. The tutorial will run normally if the **debug** flag is **yes** allowing the continuation of the development. Listing 5.16 shows **tutoInDevelopment** usage.

Listing 5.16. tutoInDevelopment usage

```
1   $ tutoInDevelopment -debug "no" -errorMsg "This tutorial is in development."
```

The *tutoInDevel.sh* script was added to **PATH** to simplify the usage of **tutoInDevelopment** by allowing default arguments. Listing 5.17 shows how to call the *tutoInDevel.sh* script in the *Allrun* tutorial file and Listing 5.18 shows its usage.

Listing 5.17. "Allrun" file

```
1 cd ${0%/*} || exit 1    # Run from this directory
2 eval `tutoInDevel.sh`   # Tutorial in development
```

Listing 5.18. "tutoInDevel.sh" usage

```
1   $ ./Allrun "no" "This tutorial is in development."
```

5.9 patoCodingStyle

Listing 5.19 shows how to format the C++ code in PATO. This utility needs to be run before a commit.

Listing 5.19. patoCodingStyle usage

```
1   $ patoCodingStyle -dir "$PATO_DIR" # -format yes -verbose
```

5.10 tests

tests directory contains the testing framework and testing suite. The unit tests can be run through the **runtests** executable. During the tests, the outputs from the models and tutorials are compared to reference files located in **testsuites** directory. Listing 5.20 shows **runtests** usage.

Listing 5.20. runtests usage

```
1   $ runtests # -h for help
```

Listing 5.21 shows an output example of successful unit testing (**TestIOFunctions**).

Listing 5.21. Output example of a successful unit test

```
1 [-- TestIOFunctions -- Test 1 / 4: Success -- ClockTime = 00h00m03s --]
2 [-- TestIOFunctions -- Test 2 / 4: Success -- ClockTime = 00h00m03s --]
3 [-- TestIOFunctions -- Test 3 / 4: Success -- ClockTime = 00h00m03s --]
4 [-- TestIOFunctions -- Test 4 / 4: Success -- ClockTime = 00h00m03s --]
```

Chapter 6

Conclusion

PATO is a research and development platform for multiphase porous reactive materials submitted to high-temperature environments or other unusual conditions. It is implemented as a C++ top-level module of OpenFOAM enabling the coupling to mesh generation (e.g. Pointwise), optimization (e.g. Dakota), post-processing (e.g. Tecplot), and thermochemical (MUTATION^{++}) tools. The PATO modularity allows to use physical models from simple Fourier heat transfer to more advanced ones such as internal decomposition (pyrolysis, vaporization), gas-gas and gas-solid chemical interactions (combustion, cracking, coking), gas species transport (convection, diffusion), and solid morphology evolutions (internal density changes, surface ablation).

Acknowledgment

This work was supported by the NASA Entry Systems Modeling project (Michael Barnhardt project manager and Aaron Brandis principal investigator) as part of the NASA Game Changing Development program. Jeremie B. E. Meurisse and Nagi N. Mansour were funded by NASA contract NNA15BB15C to Analytical Mechanics Associates (AMA), Inc. The authors would like to thank Sergio Fraile Izquierdo, John Thornton, Georgios Bellas Chatzigeorgis, Cyril Levet, and Bruno Dias for their insightful discussions and reviews on this work.

Bibliography

[1] P. Cardiff, A. Karač, A. Ivanković, A large strain finite volume method for orthotropic bodies with general material orientations, Computer Methods in Applied Mechanics and Engineering 268 (2014) 318–335.

[2] A. A. Baker, Composite materials for aircraft structures, AIAA, 2004.

[3] S. W. Tsai, E. M. Wu, A general theory of strength for anisotropic materials, Journal of composite materials 5 (1) (1971) 58–80.

[4] I. M. Daniel, Failure of composite materials, Strain 43 (1) (2007) 4–12.

[5] P. Soden, M. Hinton, A. Kaddour, A comparison of the predictive capabilities of current failure theories for composite laminates, Composites science and technology 58 (7) (1998) 1225–1254.

[6] J. B. Scoggins, V. Leroy, G. Bellas-Chatzigeorgis, B. Dias, T. E. Magin, Mutation++: MUlticomponent Thermodynamic And Transport properties for IONized gases in C++, SoftwareX 12 (2020) 100575. doi:10.1016/j.softx.2020.100575.

[7] J. B. Scoggins, T. E. Magin, Development of Mutation++: Multicomponent Thermodynamic and Transport Properties for Ionized Plasmas written in C++, in: 11th AIAA/ASME Joint Thermophysics and Heat Transfer Conference, Atlanta,GA, 2014, p. 2966, AIAA 2014-2966. doi:10.2514/6.2014-2966.

[8] L. V. Gurvich, I. V. Veits, C. B. Alcock, Thermodynamics properties of individual substances. Volume 1 - Elements O, H/D, T/, F, Cl, Br, I, He, Ne, Ar, Kr, Xe, Rn, S, N, P, and their compounds. Part 1 - Methods and computation. Part 2 - Tables (4th revised and enlarged edition), New York, NY (US); Hemisphere Publishing Corp., 1989.

[9] B. J. McBride, S. Gordon, M. A. Reno, Thermodynamic Data for Fifty Reference Elements, NASA TP-3287-REV1, Glenn Research Center, Cleveland, Ohio (February 2001).

[10] W. G. Vincenti, C. H. Kruger, Introduction to Physical Gas Dynamics. , Vol. 70, Cambridge University Press (CUP), 1965. doi:10.1017/s0001924000057304.

[11] J. H. Ferziger, H. G. Kaper., Mathematical Theory of Transport Processes in Gases., North-Holland Publishing Company, 1972.

[12] T. E. Magin, A Model for Inductive Plasma Wind Tunnels, Ph.D. thesis, von Karman Institute for Fluid Dynamics & Universitè Libre de Bruxelles (2004).

[13] T. E. Magin, G. Degrez, Transport properties of partially ionized and unmagnetized plasmas, Physical Review E 70 (4). doi:10.1103/physreve.70.046412.

[14] T. E. Magin, G. Degrez, Transport algorithms for partially ionized and unmagnetized plasmas, Journal of Computational Physics 198 (2004) 424–449. doi:10.1016/j.jcp.2004.01.012.

[15] J. B. Scoggins, Development of numerical methods and study of coupled flow, radiation, and ablation phenomena for atmospheric entry, Ph.D. thesis, von Karman Institute for Fluid Dynamics & Centrale-Supèlec (2017).

[16] J. B. Scoggins, T. E. Magin, The Gibbs Function Continuation Method for Linearly Constrained Multiphase Equilibria, Combustion and Flame 162 (12) (2015) 4514–4522. doi:10.1016/j.combustflame.2015.08.027.

[17] J. Lachaud, J. Scoggins, T. Magin, M. Meyer, N. Mansour, A generic local thermal equilibrium model for porous reactive materials submitted to high temperatures, International Journal of Heat and Mass Transfer 108 (2017) 1406–1417. doi:10.1016/j.ijheatmasstransfer.2016.11.067.