IAEA
International Atomic Energy Agency

Joint ICTP-IAEA Workshop on
Open-Source Nuclear Codes for Reactor Analysis
07 - 11 Aug 2023
Contact: ONCORE@iaea.org

Introduction to OFFBEAT

10 August 2023

Alessandro Scolaro - EPFL

Objectives

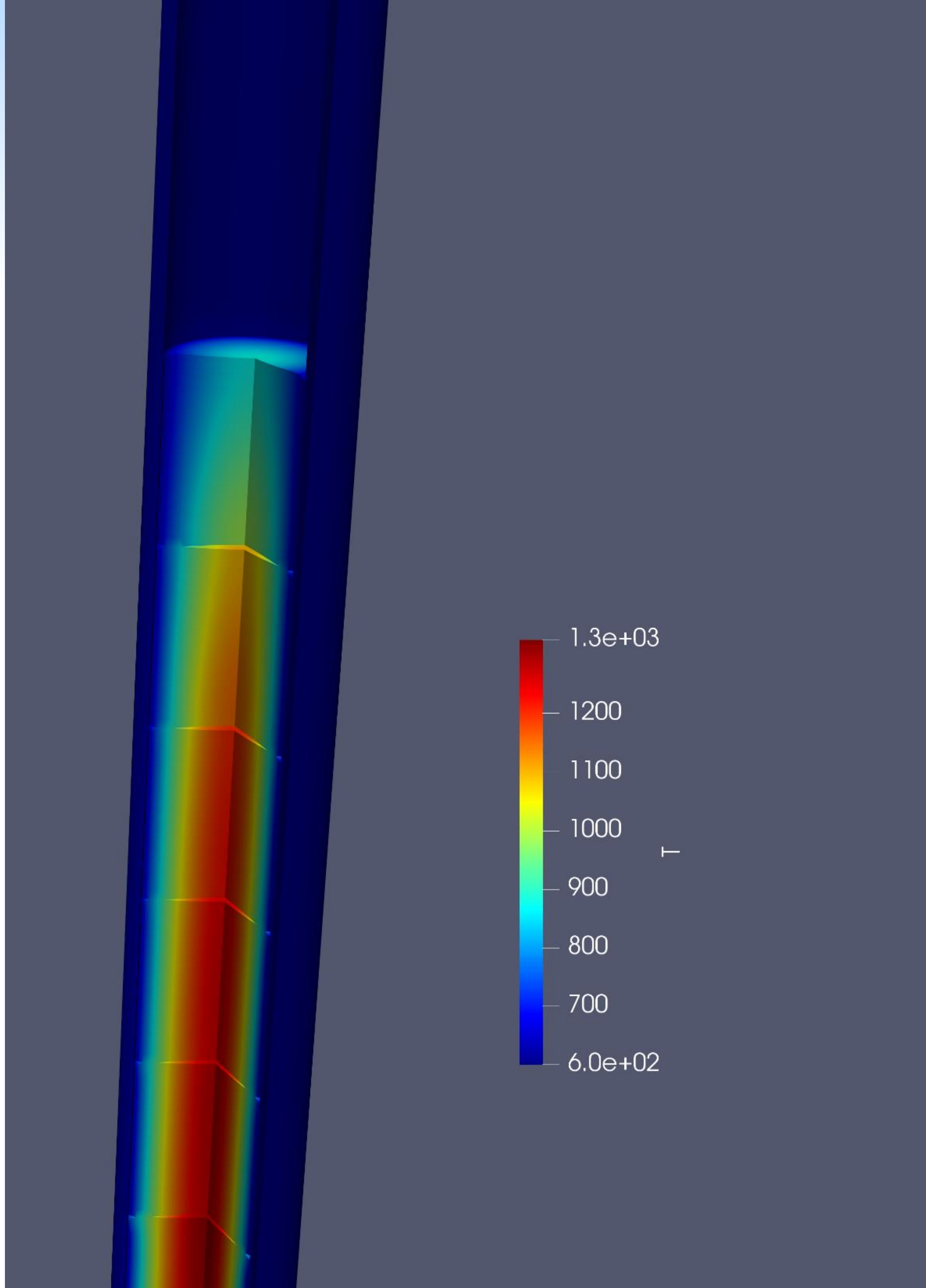
1. Introduce OFFBEAT main capabilities and features

2. Provide example of applications

3. Basic knowledge on how to approach OFFBEAT

OFFBEAT – OpenFOAM Fuel Behavior Analysis Tool

- Multi-dimensional fuel performance code:
  - 1D, 2D, 3D with arbitrary geometry & mesh
- Based on OpenFOAM® C++ library
- In development since 2017 at EPFL and PSI (Switzerland)
- Open-source at: https://gitlab.com/foam-for-nuclear/offbeat



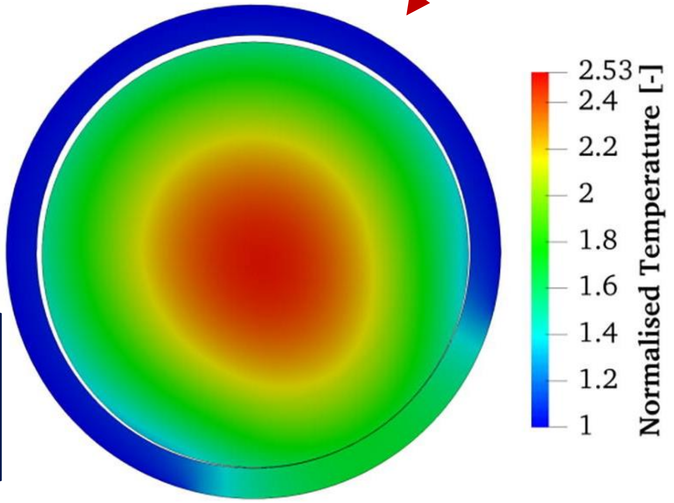
Project prompted by a fuel failure in a Swiss NPP

2017 - studies at PSI by I. Clifford et al., Ann. Nucl. En., 2019

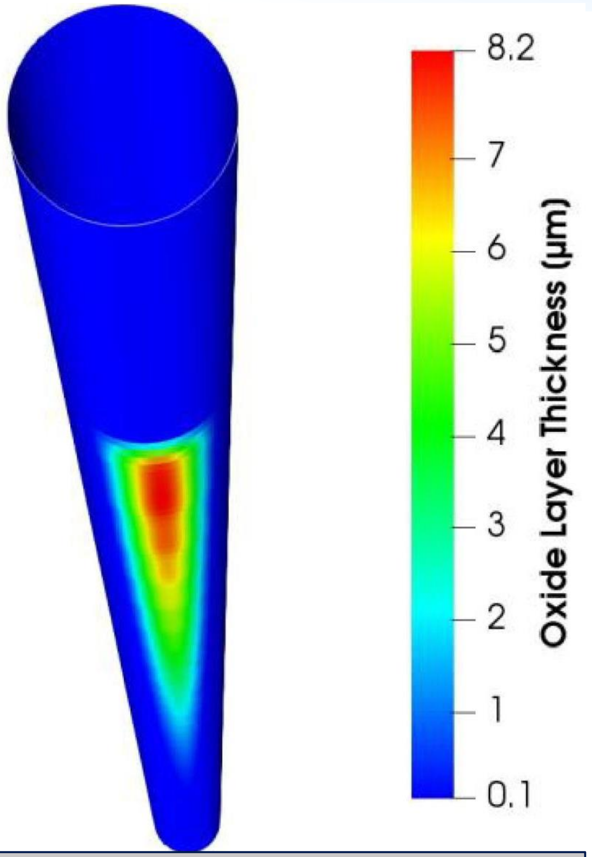
Community would benefit from a readily available tool that:

- Enables ***multi-dimensional*** analysis.
- Clears the way to ***multi-physics*** and coupling.
- Allows ***straightforward tailoring***.

Not feasible with traditional tools



Temperature distribution for eccentric fuel under power peaking and partial dryout



Cladding oxide layer thickness under partial dryout

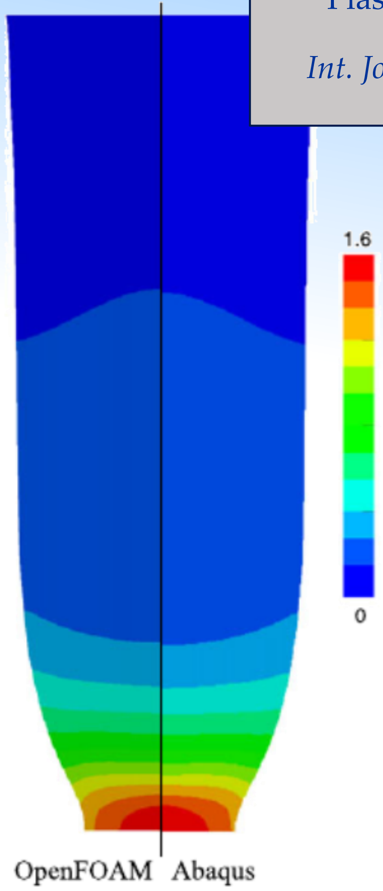
Solid mechanics & Finite Volume?

Extension of the FVM to **solid mechanics** in the last 30 years:

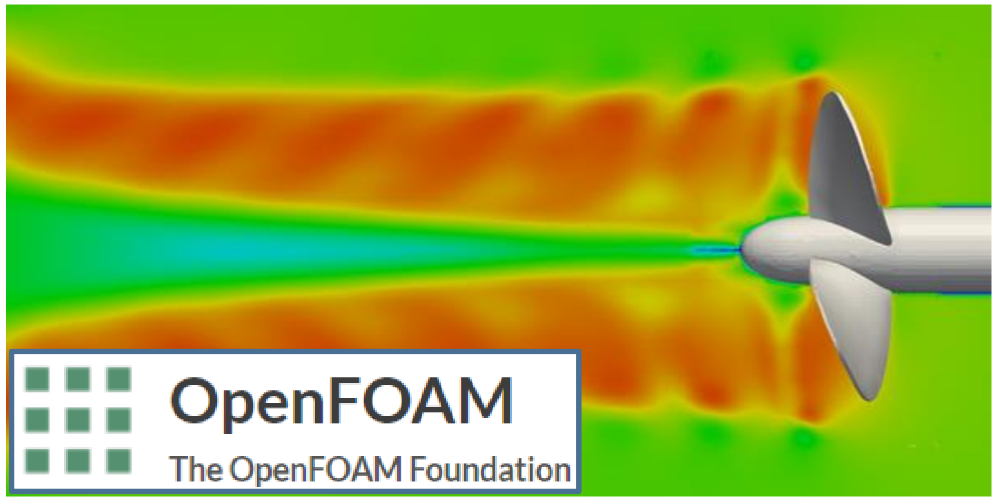
- Only simple linear elastic solver in standard OpenFOAM..
- … but lots of great work done by the community!

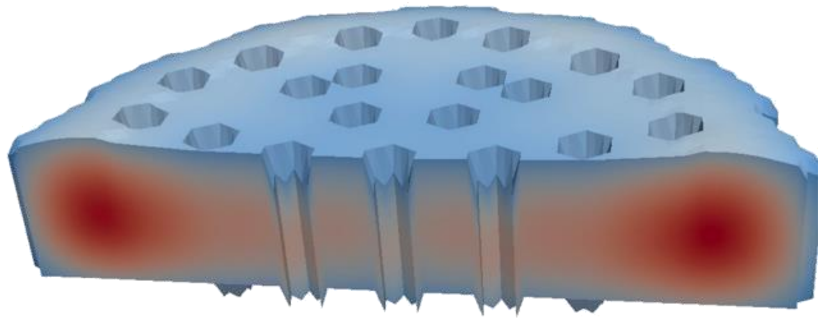
  (e.g. foam-extend, ***solids4Foam***)

Choice of OpenFOAM also motivated by EPFL-PSI experience for reactor analysis (e.g. ***Gen-Foam*** platform)



Plastic strain - Bar necking
P. Cardiff et al.,
Int. Jor. Num. Meth. Eng., 2017





Flux in the ESFR core
C. Fiorina et al.,
Nuc. Eng. Des., 2015.

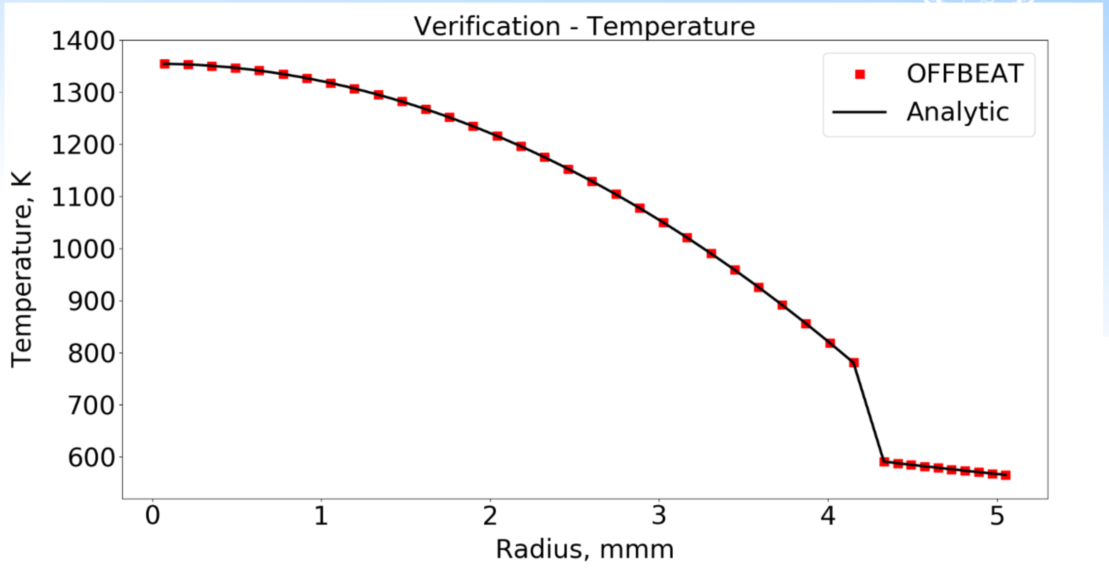
V&V status and past applications

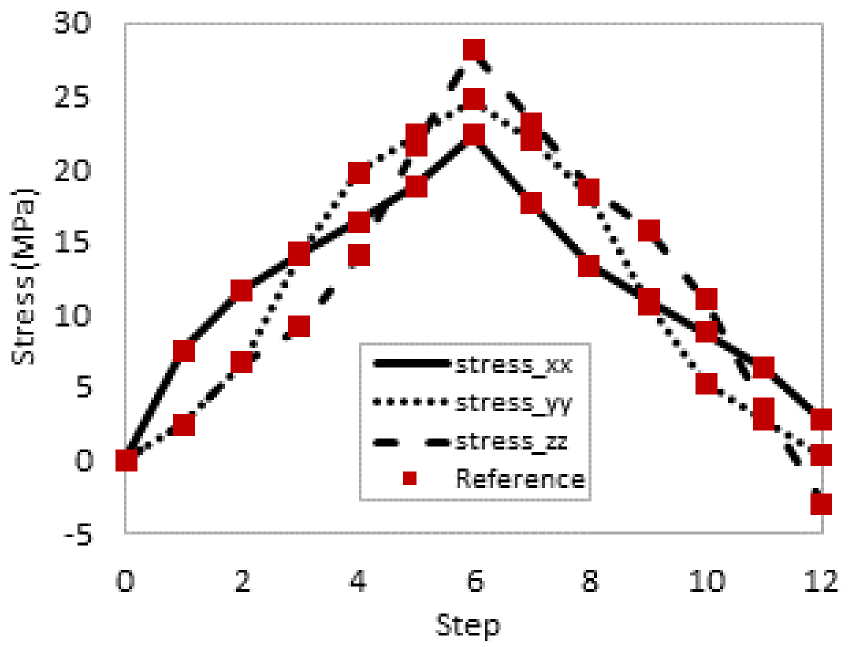
Verification

1. Simple cases with focus on ***single aspects.***
2. Useful for finding bugs.

Main tests:

- Rod free volume
- Heat transfer across the gap
- ***Steady-state*** and transient ***temperature***/stress profile
- Rate-independent plasticity model
- Corrosion layer thickness
- Limbäck creep model and ***plasticity***
- Contact model(s) benchmark
- Corrosion
- Neutronics
- Porosity
- TRISO properties and IAEA benchmarks





Validation database

Main focus on thermal analysis:

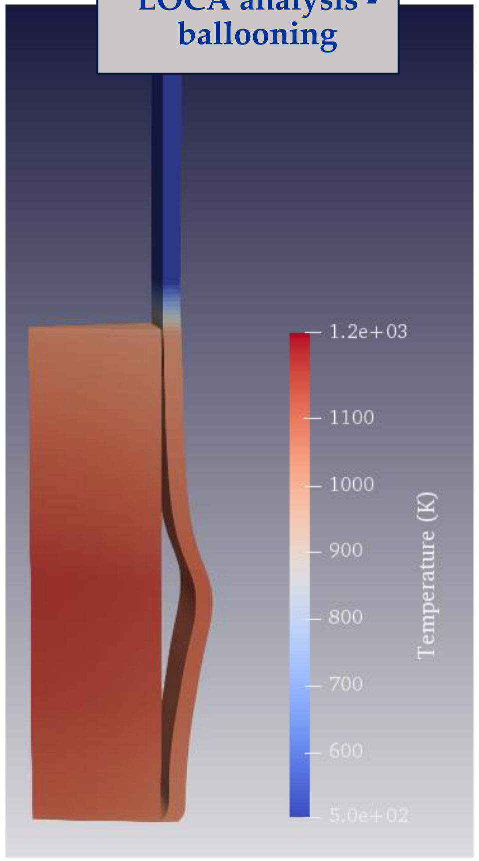
- Fuel Centerline Temperature (FCT)
- Fission Gas Release (FGR)

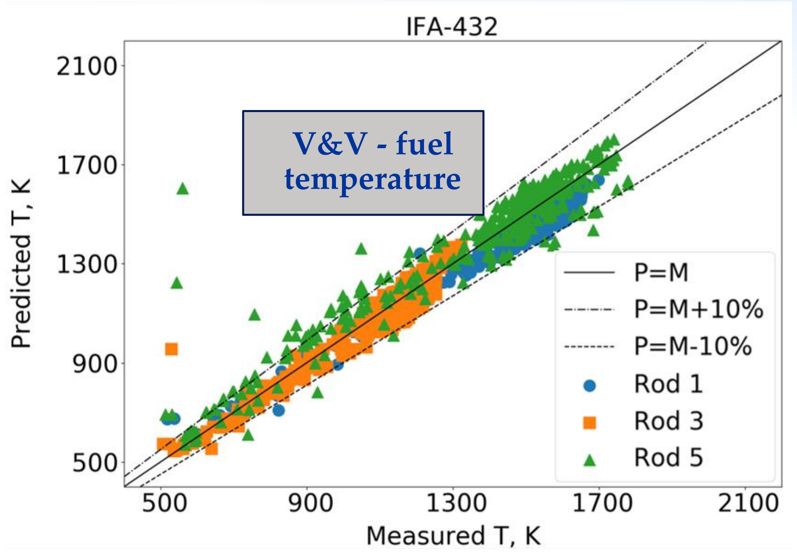
NOTE:

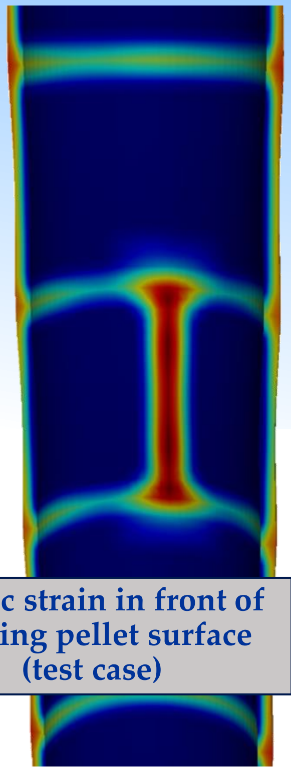
- **Mechanics is indirectly embedded in thermal analysis (e.g. gap size)**
- **But validation for quantity such as elongation or cladding expansion is essential for future efforts.**

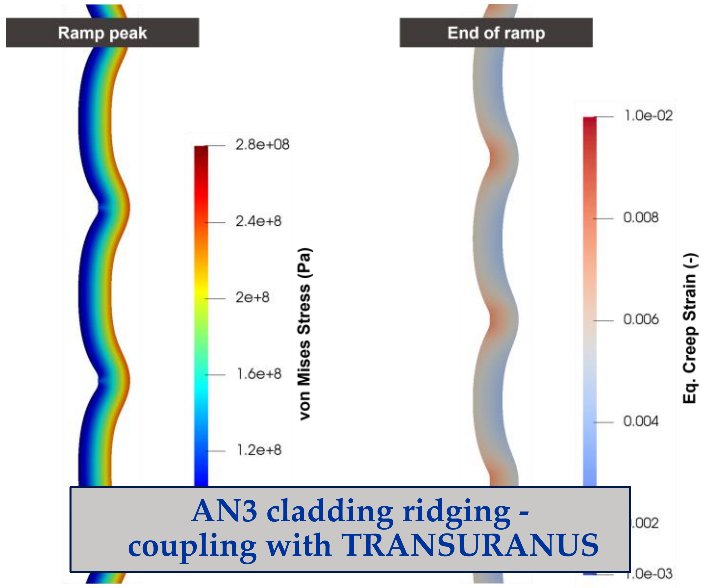
Experiment	Rod number/name		Burnup, MWd/kg	Quantity of interest
IFA-562.1	Lower Cluster	1 2 3 4 5 6	~10	FCT
	Upper Cluster	7 8 9 10 11 12		
IFA-432	1 3 5		~30/40	FCT
Super Ramp	PW3	2 3	~40	FGR
	PK1	1 2 3 4		
	PK2	1 2 3 4 S		
	PK4	1 2 3 S		
	PK6	2 3 S		
	BK7	3 4 5 6		
Risø-3	AN	1 2 3 4 8 10 11	~30/40	FGR (FCT for refabricated rods)
	II	3 5		
	GE	2 4 6 7		

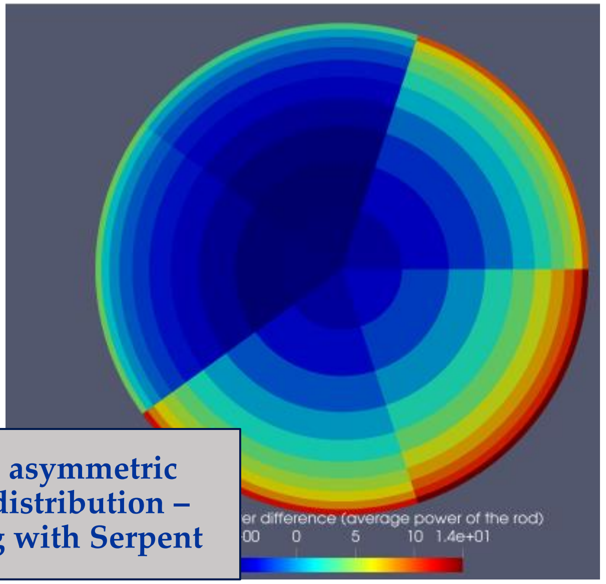
Main applications for LWR fuel (examples)


LOCA analysis - ballooning

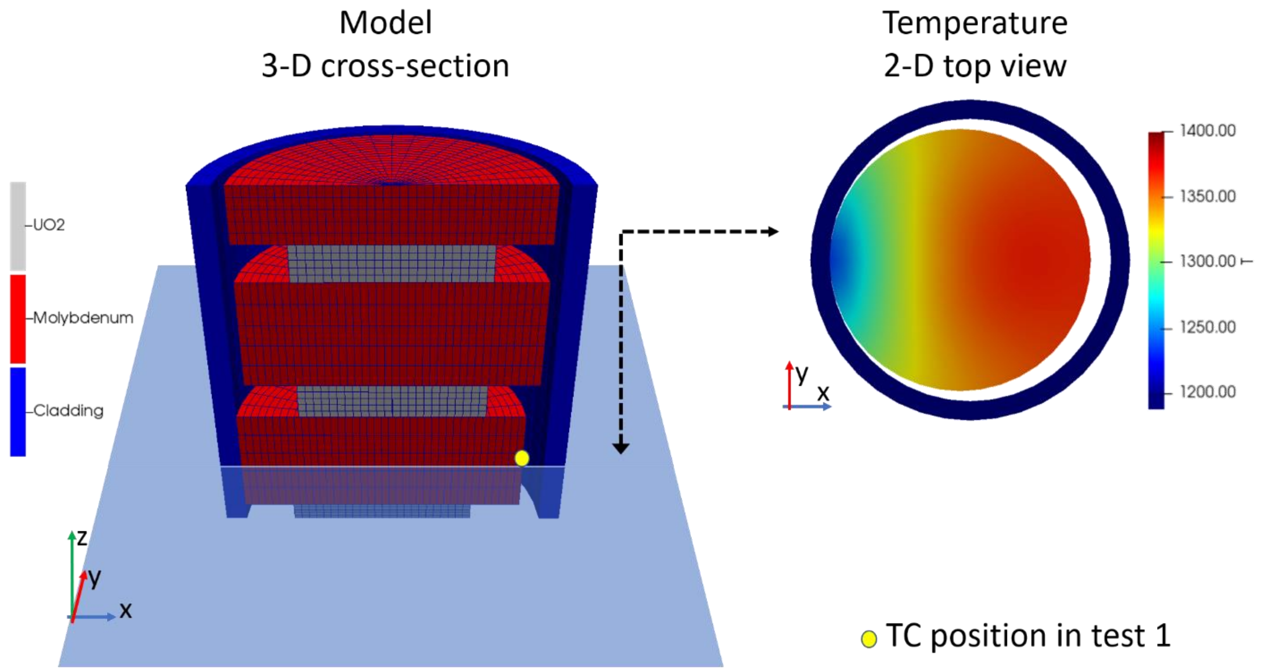

V&V - fuel temperature


Plastic strain in front of missing pellet surface (test case)


AN3 cladding ridging - coupling with TRANSURANUS

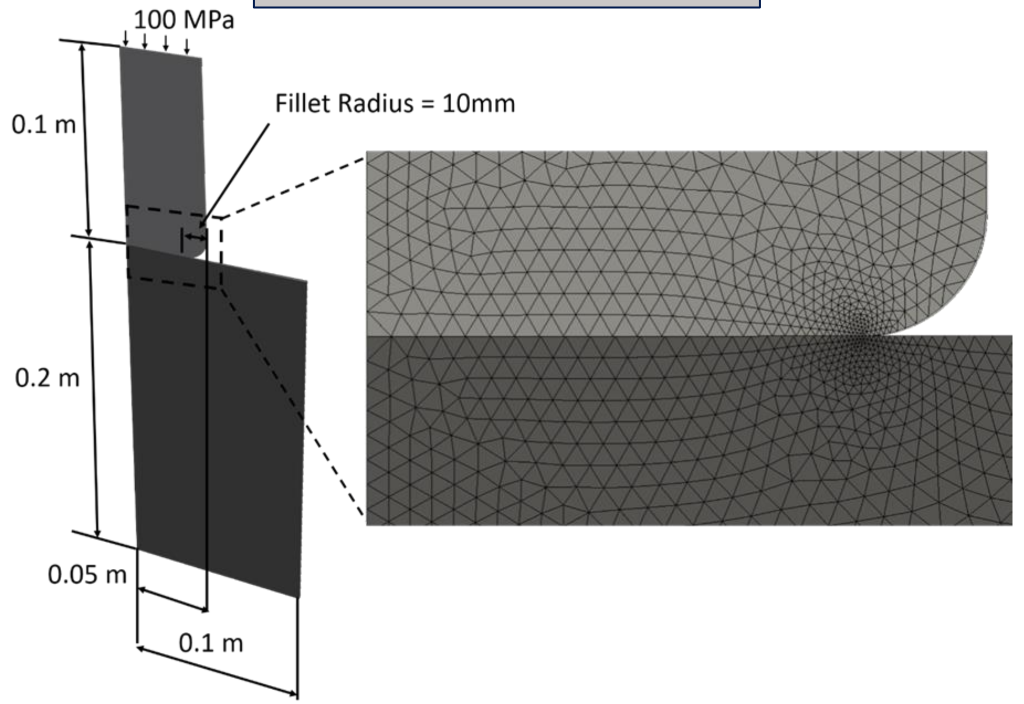

Coarse asymmetric power distribution – coupling with Serpent

Not *just* standard configurations and materials!

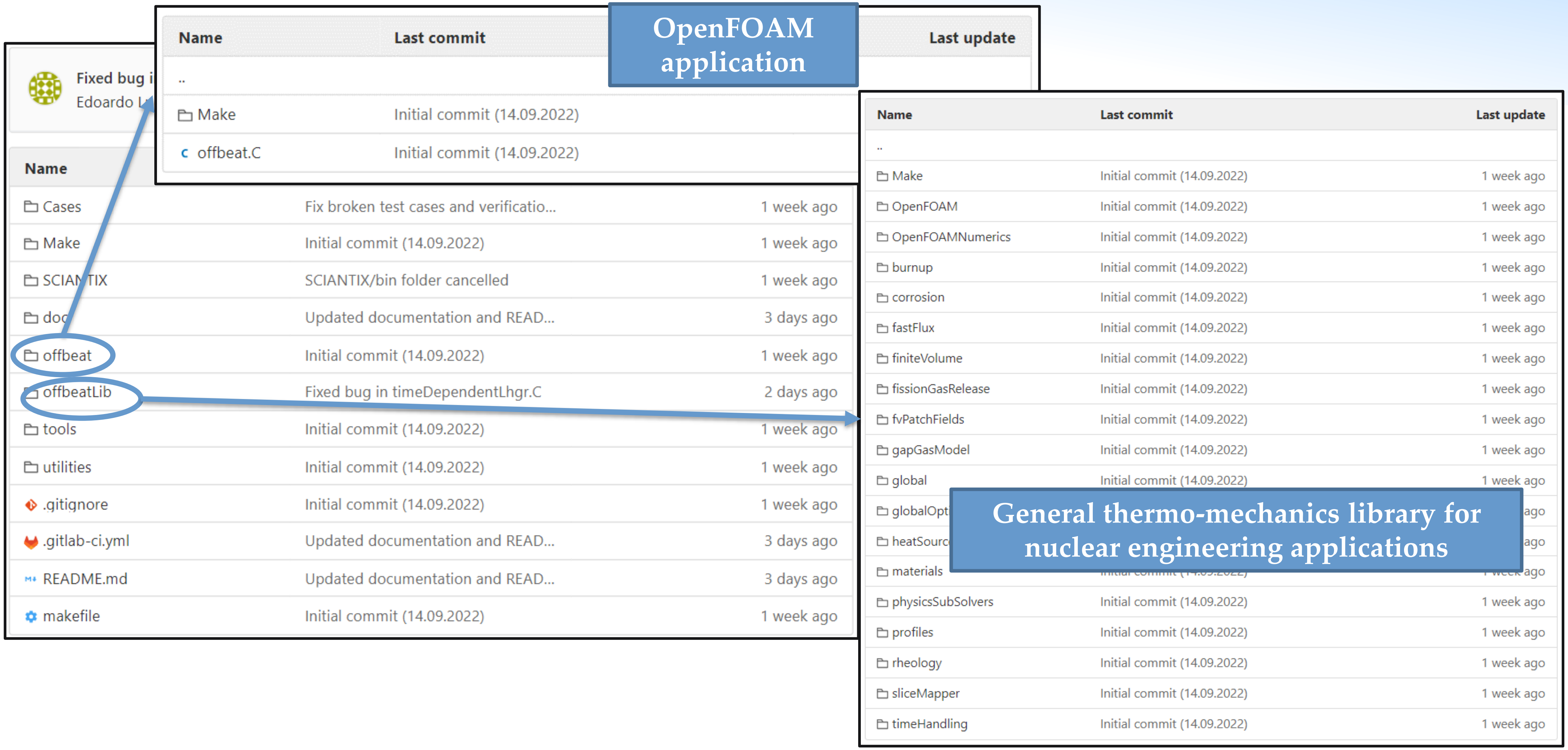


Temperature in eccentric fuel disc irradiation.
A. Scolaro et al.,
Nucl. Eng. Techn., 2021

Punch test – part of contact verification cases.
A. Scolaro et al.,
Num. Meth. Eng., 2021



Not *just* standard configurations and materials!



Fixed bug i
Edoardo L

Name		
Cases	Fix broken test cases and verificatio...	1 week ago
Make	Initial commit (14.09.2022)	1 week ago
SCIANTIX	SCIANTIX/bin folder cancelled	1 week ago
doc	Updated documentation and READ...	3 days ago
offbeat	Initial commit (14.09.2022)	1 week ago
offbeatLib	Fixed bug in timeDependentLhgr.C	2 days ago
tools	Initial commit (14.09.2022)	1 week ago
utilities	Initial commit (14.09.2022)	1 week ago
.gitignore	Initial commit (14.09.2022)	1 week ago
.gitlab-ci.yml	Updated documentation and READ...	3 days ago
README.md	Updated documentation and READ...	3 days ago
makefile	Initial commit (14.09.2022)	1 week ago

OpenFOAM application

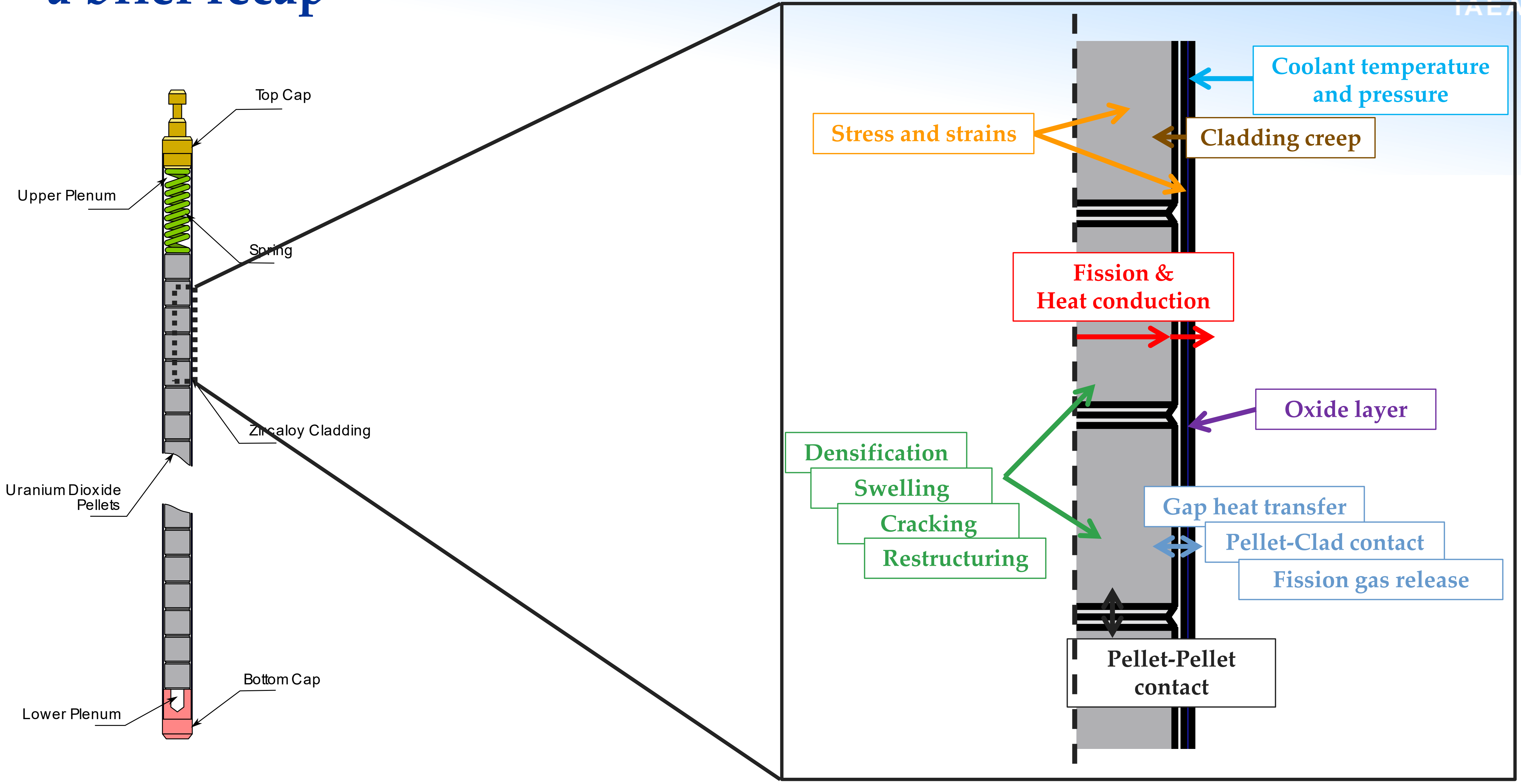
Name	Last commit	Last update
..		
Make	Initial commit (14.09.2022)	
offbeat.C	Initial commit (14.09.2022)	

Name	Last commit	Last update
..		
Make	Initial commit (14.09.2022)	1 week ago
OpenFOAM	Initial commit (14.09.2022)	1 week ago
OpenFOAMNumerics	Initial commit (14.09.2022)	1 week ago
burnup	Initial commit (14.09.2022)	1 week ago
corrosion	Initial commit (14.09.2022)	1 week ago
fastFlux	Initial commit (14.09.2022)	1 week ago
finiteVolume	Initial commit (14.09.2022)	1 week ago
fissionGasRelease	Initial commit (14.09.2022)	1 week ago
fvPatchFields	Initial commit (14.09.2022)	1 week ago
gapGasModel	Initial commit (14.09.2022)	1 week ago
global	Initial commit (14.09.2022)	1 week ago
globalOpt		ago
heatSourc		ago
materials		ago
physicsSubSolvers	Initial commit (14.09.2022)	1 week ago
profiles	Initial commit (14.09.2022)	1 week ago
rheology	Initial commit (14.09.2022)	1 week ago
sliceMapper	Initial commit (14.09.2022)	1 week ago
timeHandling	Initial commit (14.09.2022)	1 week ago

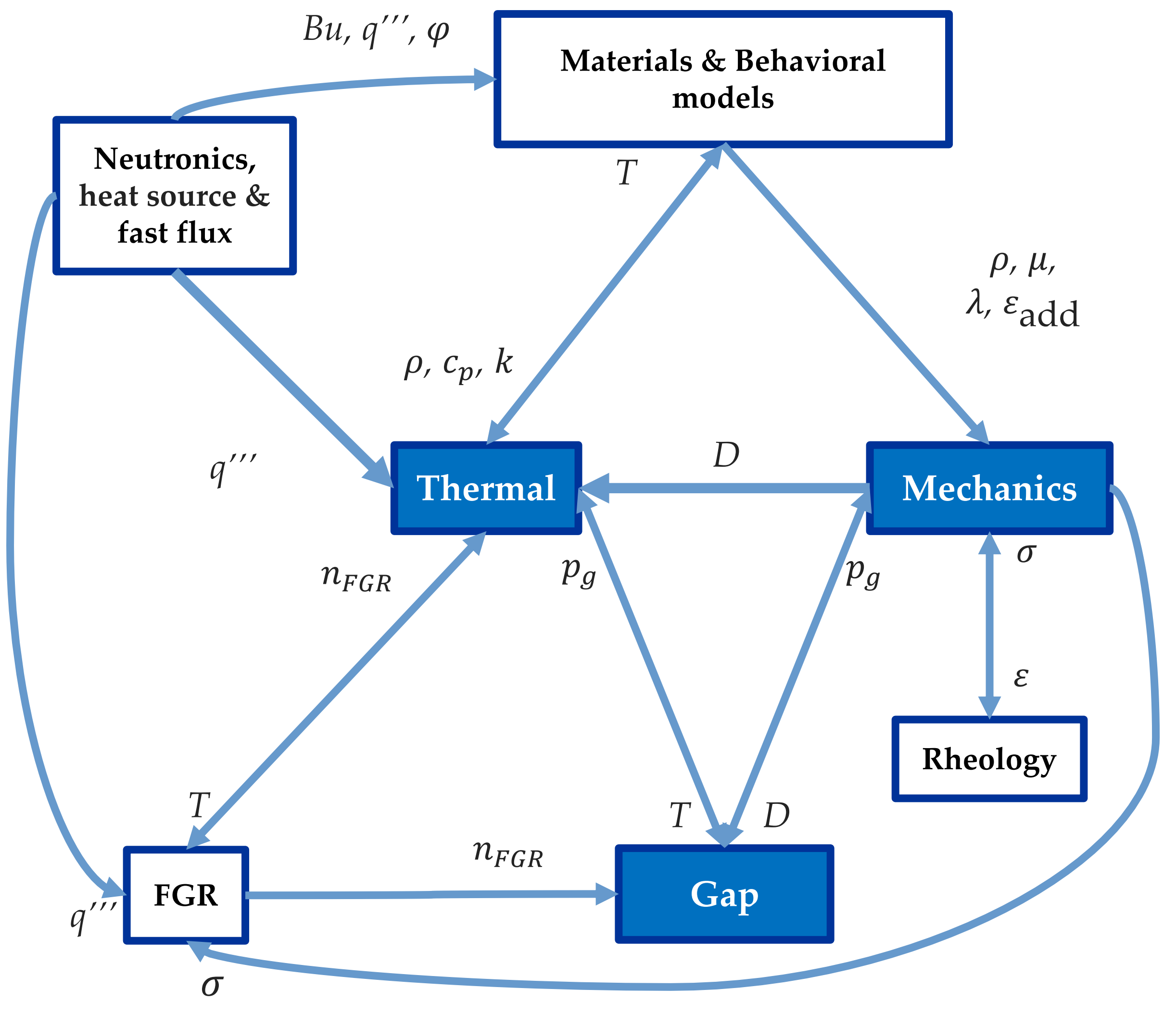
General thermo-mechanics library for nuclear engineering applications

OFFBEAT
Code structure

The complexity of fuel behavior: a brief recap



Code structure (only main dependencies are shown)

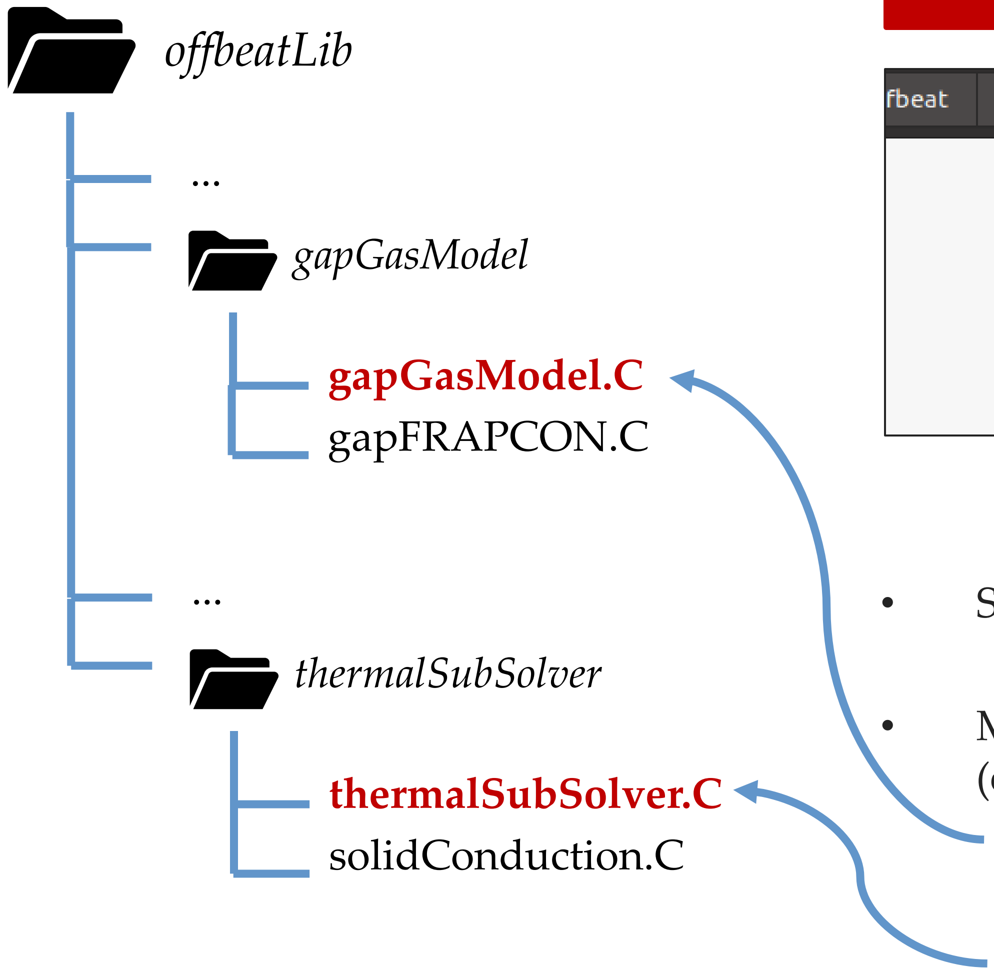


- Separate aspects of fuel behavior encapsulated in separate classes.

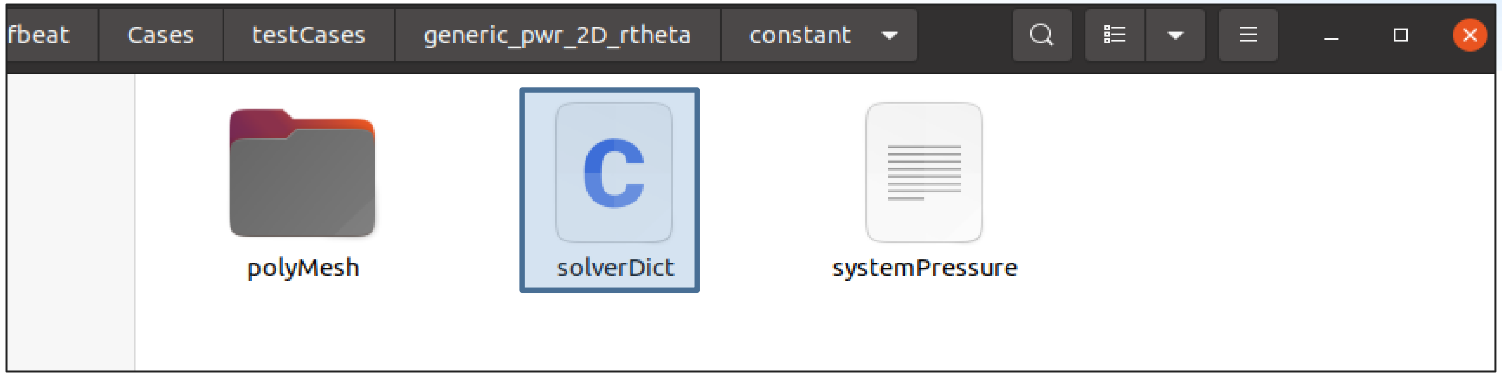
- Some physics are reproduced with models and correlations…
- …others by solving governing equations.

- Fields and variables exchanged through functions or registry-lookup.

Switching on/off physics



NOTE: most OFFBEAT options are in a single file, the *solverDict*.



- Solvers/models are run-time selectable in the solverDict.
- Mother class selected to switch off the physics/model (e.g. for pure thermal analysis):
  - *none:* no additional field is created (e.g. for empty gap gas or fission gas release model)
  - *fromLatestTime:* the main field (e.g. temperature) is created but left unchanged to default/initial value.

OFFBEAT
Physics and solution strategy

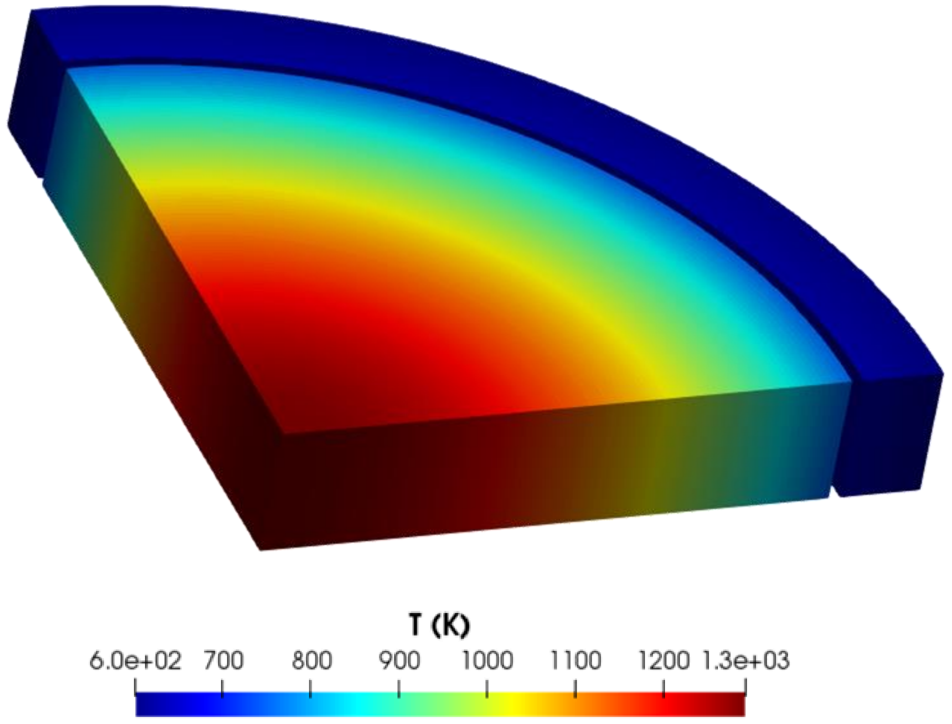
Thermal solver - Heat diffusion equation

$$\int_V \frac{\partial \rho c_p T}{\partial t} dV = \oint_S \mathbf{n} \cdot k \nabla T \, dS + \boxed{\int_V q''' \, dV}$$

Volumetric heat density q''' handled by `heatSource` class

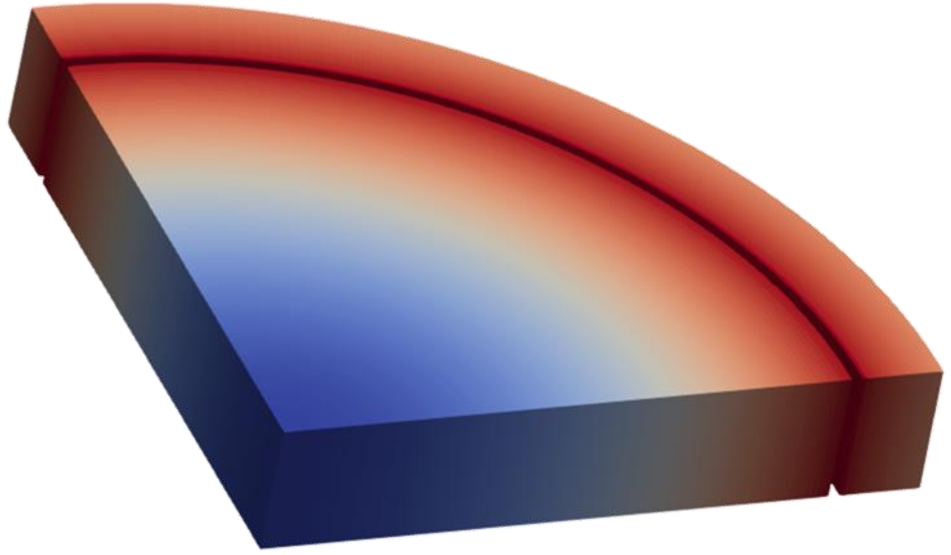
1. Through q''' file (e.g. mapped from separate neutronics code or calculated via `fvOptions` or `codeStream`).

2. Providing (time-dependent) **linear heat generation rate** (*lhgr* or $\boldsymbol{q'}$) or **volumetric heat generation rate** (*vhgr* or $\boldsymbol{q'''}$)



Mechanics Solver – Momentum balance equation(s)

$$\int_V \frac{\partial^2 \rho \mathbf{D}}{\partial t^2} dV = \oint_S \mathbf{n} \cdot \boldsymbol{\sigma} \, dS + \int_V \rho \mathbf{b} \, dV \quad \begin{matrix} \xrightarrow{\mathbf{D}} \\ \xleftarrow[\boldsymbol{\sigma},\, \boldsymbol{\varepsilon}]{} \end{matrix} \quad \boxed{\text{Rheology}}$$



What needs to be decided:

1. Approach to momentum balance equations
   - Mesh updated or not
   - Total or incremental field
2. Definition of strainSmall-strain or finite strain
3. Constitutive law for mechanical behavior

Mechanics - selection of the solver

smallStrain is a good approximation for most scenarios:

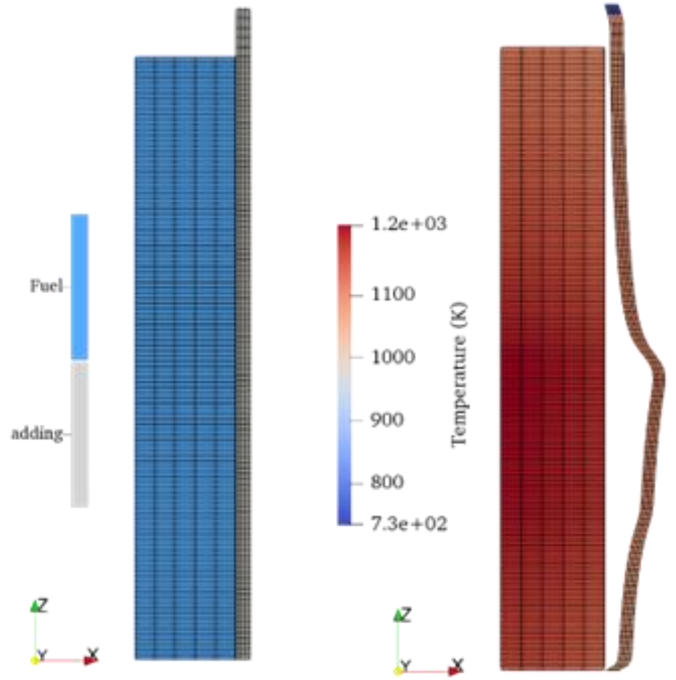
$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla\mathbf{D} + (\nabla\mathbf{D})^{\mathrm{T}}\right)$$

For finite strain scenarios (or when the change of the domain is relevant) the best options typically are updated formulations:

- ***smallStrainIncrementalUpdated*** → small strain but the mesh is updated (takes mostly care of geometric non-linearity).
- ***lageStrainUpdatedLagrangian*** → non-linear treatment of strains and compatible with non-linear constitutive laws.

offbeatLib
- ...
- *mechanicsSubSolver*
  - **smallStrain.C**
  - smallStrainIncrementalUpdated.C
  - largeStrainTotalLagrangian.C
  - largeStrainUpdatedLagrangian.C

NOTE:
when choosing updated formulations the main variable is the increment of D or DD



Dedicated BCs

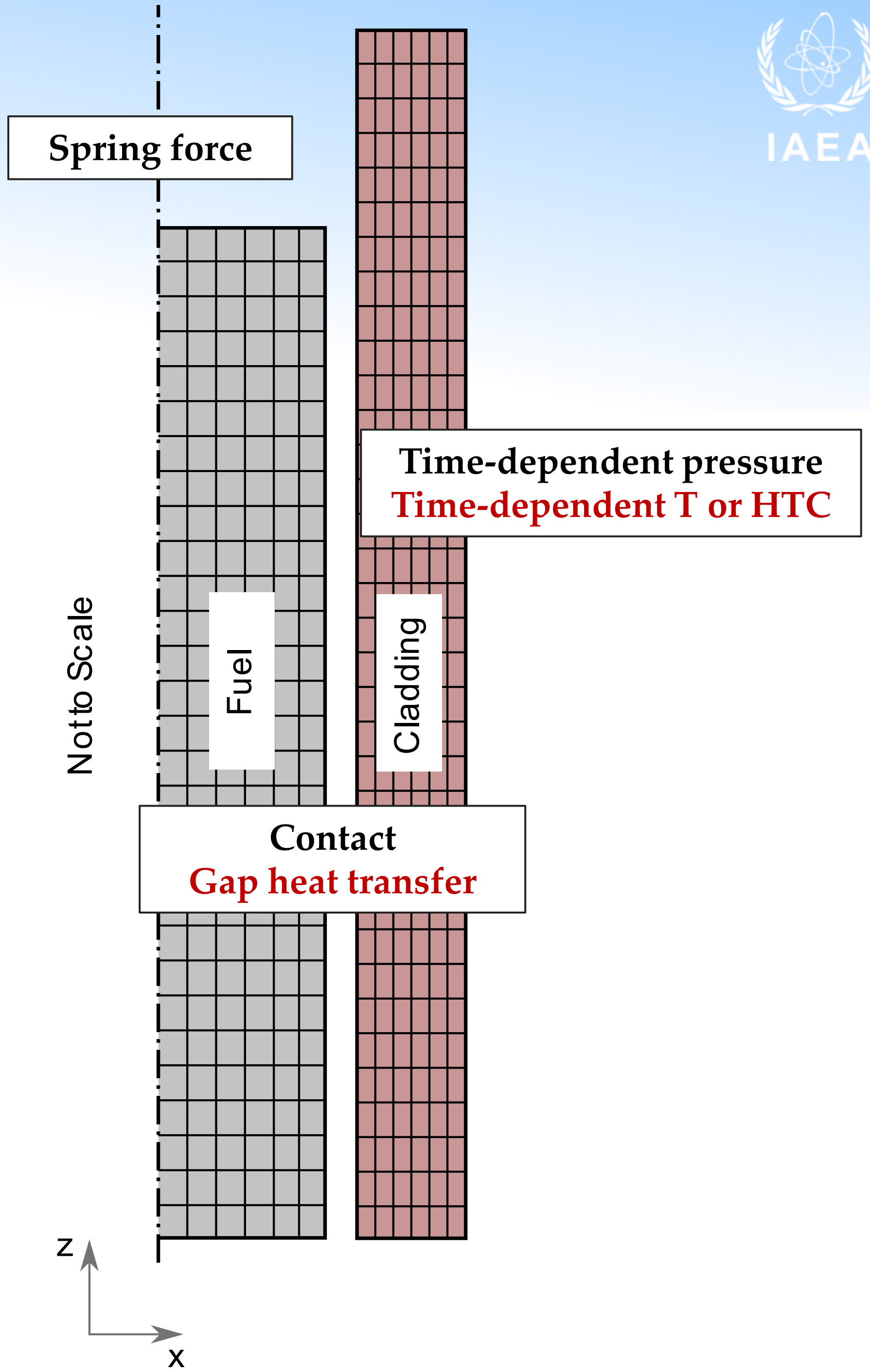
Thermo-mechanics solution obtained with a mix of standard (e.g. fixed-value) and dedicated BCs

Examples of dedicated BCs for mechanics:

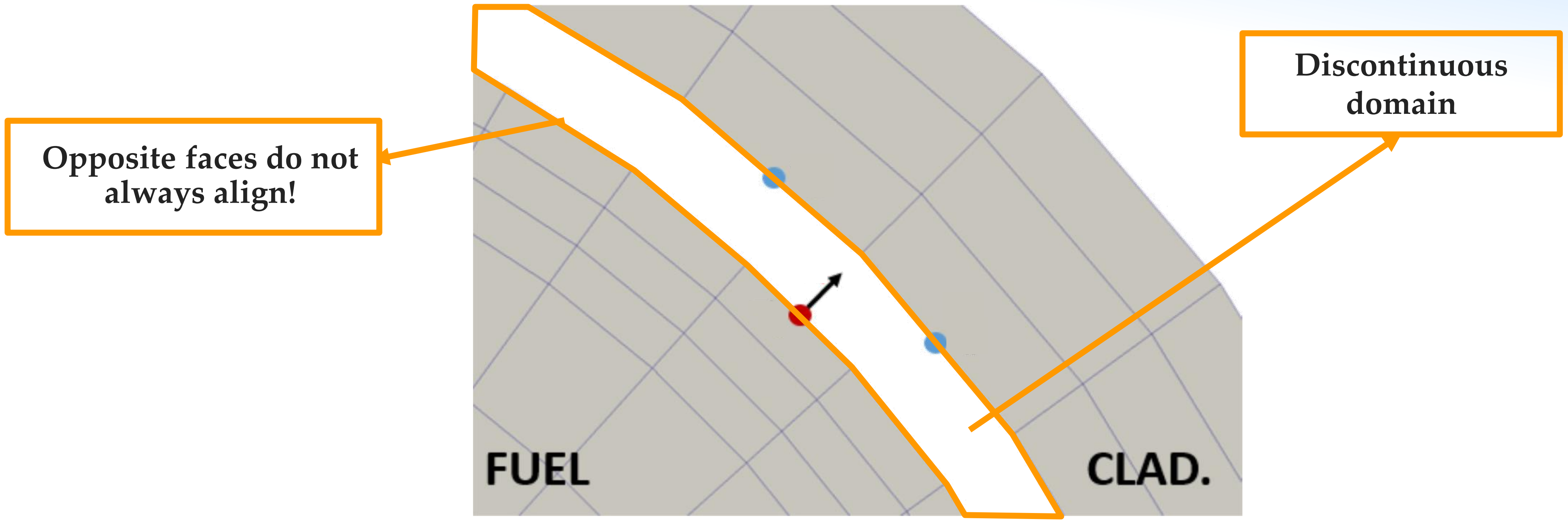
1. **Fixed or time-dependent pressure**
2. **Contact**
3. **Plenum spring pressure**

Examples of dedicated BCs for thermal analysis:

1. **Fixed or time-dependent temperature with axial profile**
2. **Fixed or time-dependent HTC with axial profile**
3. **Gap conductance heat transfer**



Mapping methodology for non-conforming patches



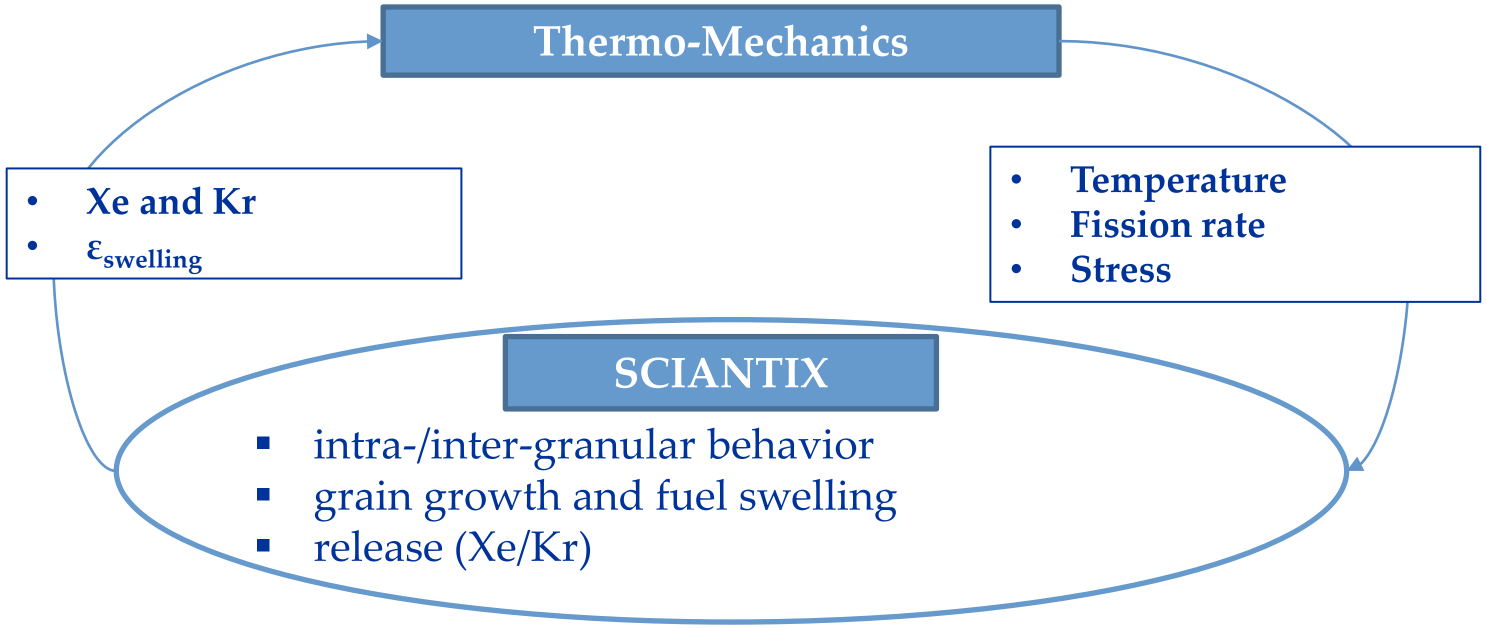
- Based on **Arbitrary Mesh Interface (AMI)** readapted to the presence of small gap.
- Heat transfer BC solved implicitly (i.e. using current value of the temperature field)

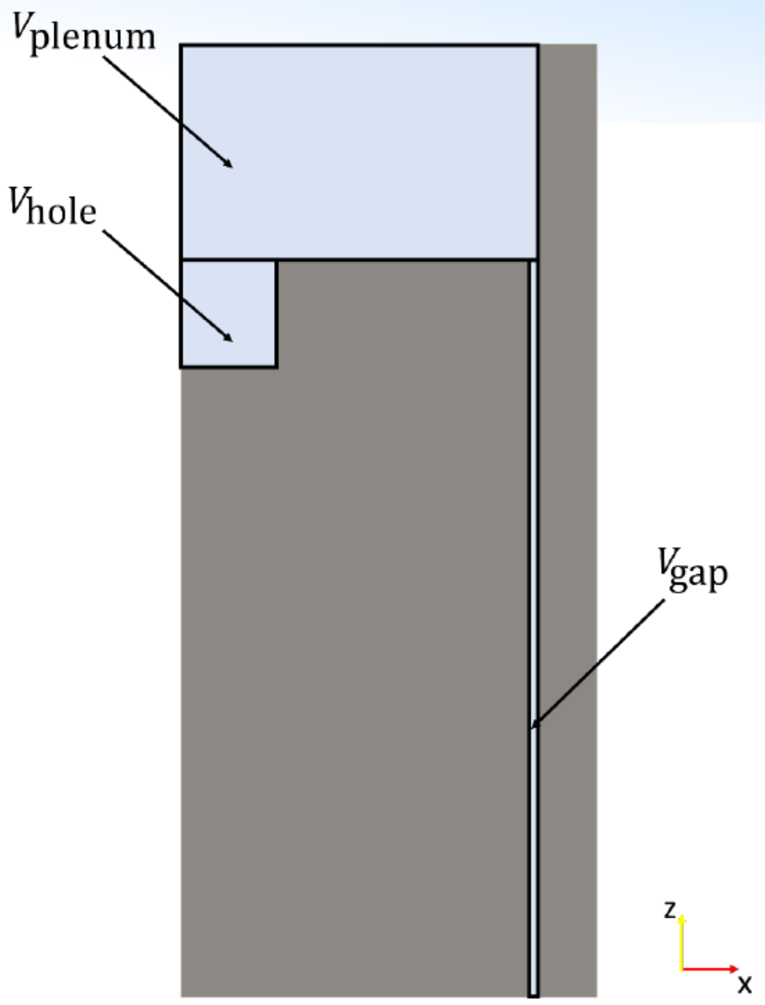
NOTE: the user must take care of defining master/slave patches as coupled!

Additional physics and models

- **Gap gas model**: keeps track of gas free volume, temperature, pressure and composition

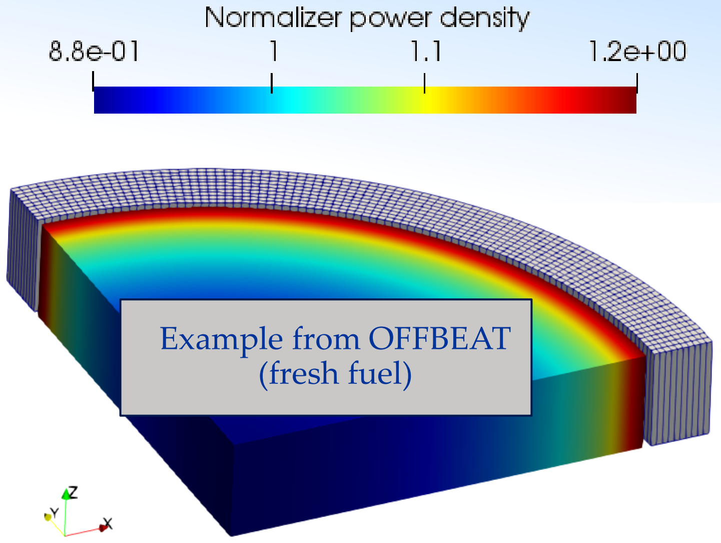
- **Fission gas release** with the SCIANTIX 0-D code: feedback on gap conductance, gas pressure and fuel swelling



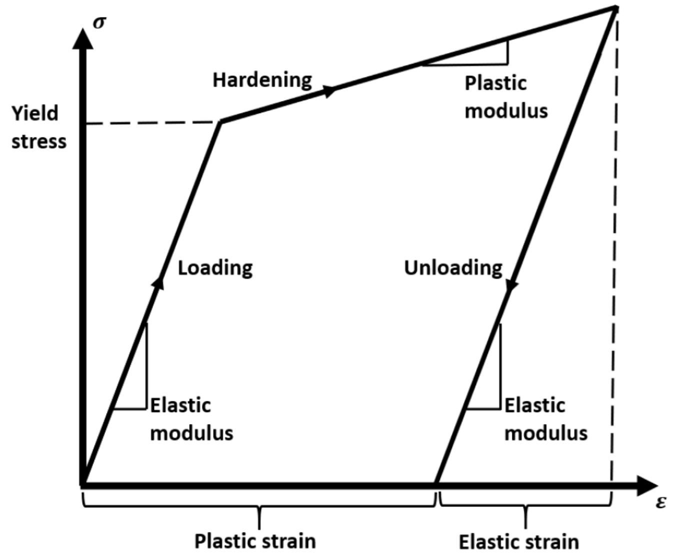


Additional physics and models

- **Burnup & neutronics** as capturing peaked power profile in fuel pellet is relevant for:
  - Rod's thermal response (lower maximum temperature).
  - Nuclide distribution: the source for advance models e.g. chemistry, plutonium redistribution etc



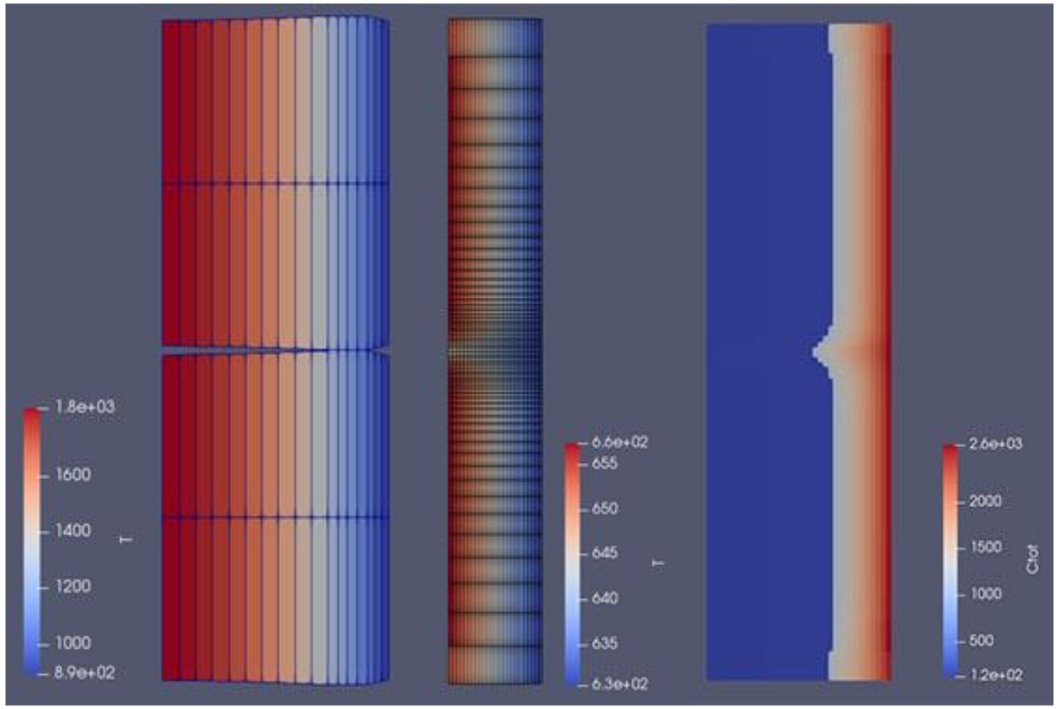
- **Heat-source & fast-flux/fast-fluence** classes

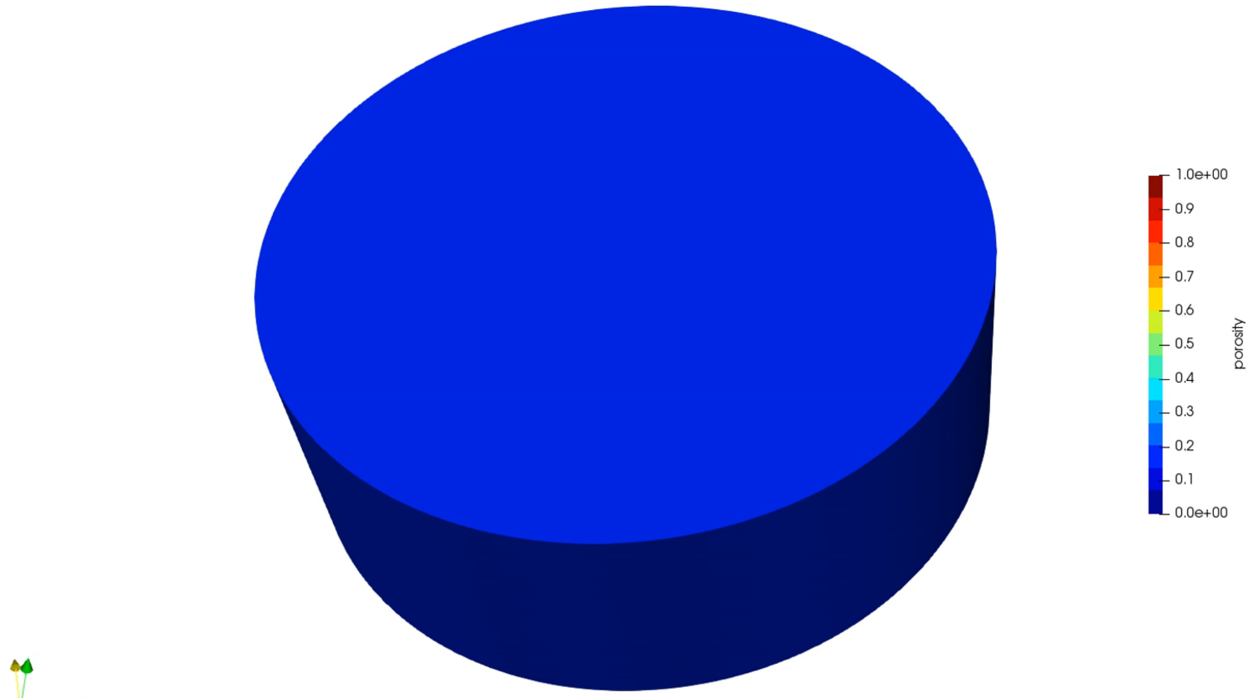


- **Material properties and rheology** class for capturing:
  - Evolution of material properties (e.g. conductivity, Young's modulus)
  - Constitutive mechanical behavior such as elasticity, plasticity or creep
  - Additional strain components due to behavioral models such as swelling or densification

Additional physics and models

- Element transport:
  - Porosity redistribution
  - Minor actinides redistribution
  - Hydrogen pickup
- Corrosion $Zr + H_2O \longrightarrow ZrO_2 + 2H_2$





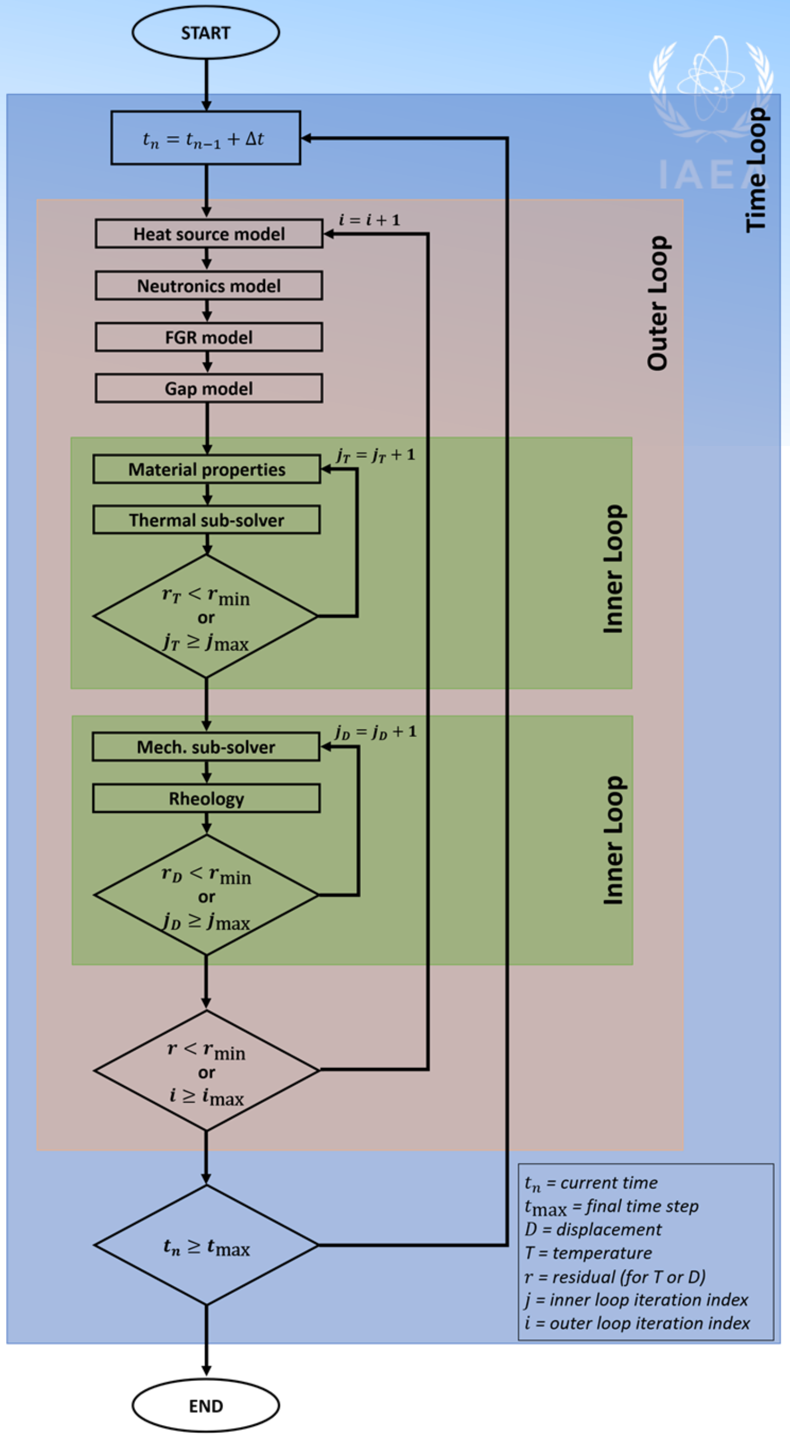
Segregated solution strategy

- Coupling through outer iterations within time step:
  - Each physics and each component solved sequentially.
  - Inner iteration for single main physics is possible.

- Convergence: user-defined residuals threshold.

$$r = \frac{1}{n}\sum |[\mathbf{b_u}] - [A_\mathbf{u}][\mathbf{u}]|$$

- Block-coupled approaches exist in the community (see solids4Foam or foamExtend)…
- … potential developments for mechanics solver (i.e. solving x-, y- and z- direction in one matrix).



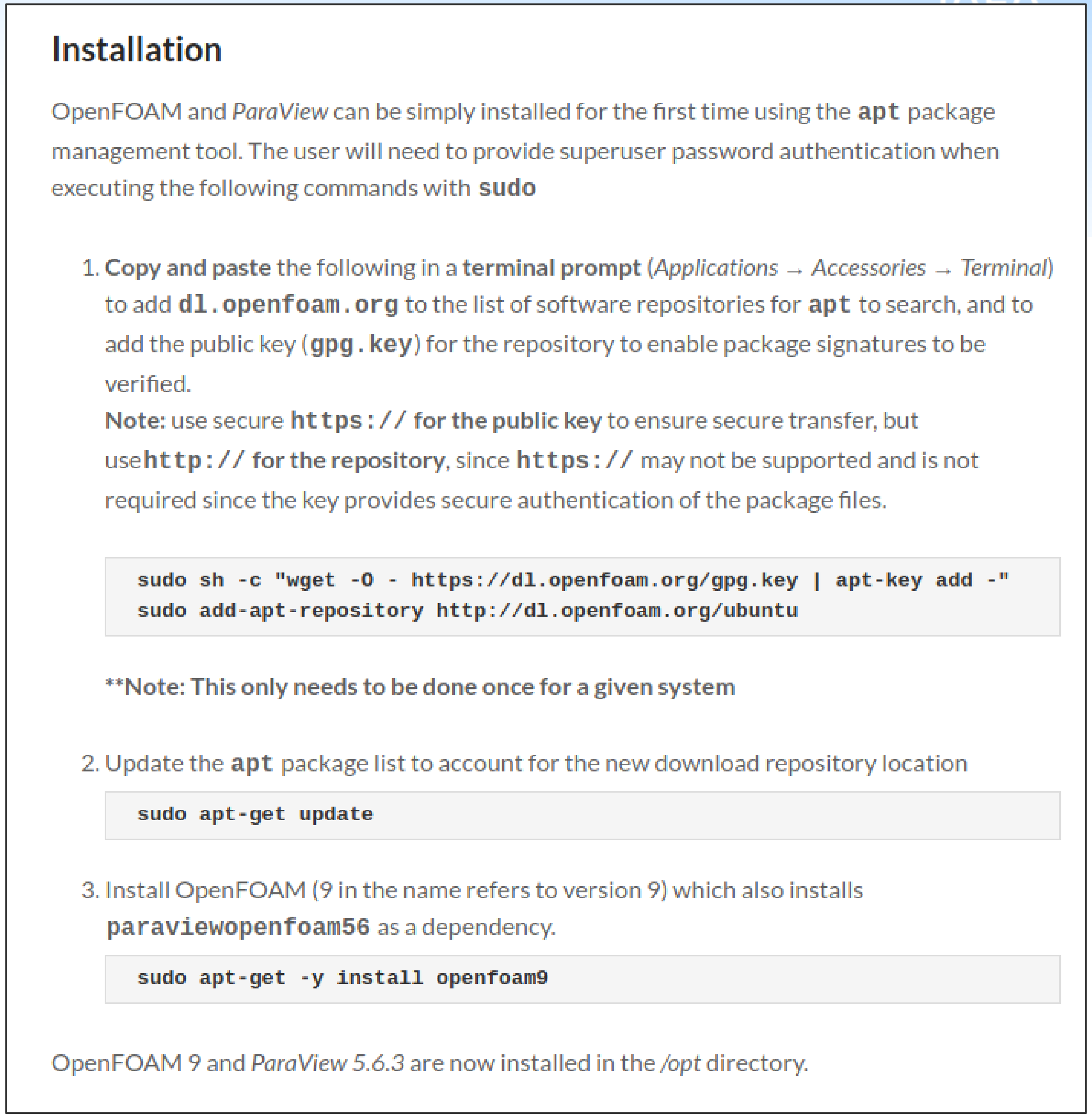
How to approach OFFBEAT Installation

Which OpenFOAM version?

Main developments are compatible with **openfoam.org version 9** (simple installation following download page)

version 10 and 11 have abandoned AMI for mapping (relevant for gap/contact treatment)… porting to new version is in progress!

A recent branch (`develop_OF2212`) compiles both on **openfoam.org version 9** and **openfoam.com v2212**



Installation

OpenFOAM and *ParaView* can be simply installed for the first time using the **`apt`** package management tool. The user will need to provide superuser password authentication when executing the following commands with **`sudo`**

1. **Copy and paste** the following in a **terminal prompt** (*Applications* → *Accessories* → *Terminal*) to add **`dl.openfoam.org`** to the list of software repositories for **`apt`** to search, and to add the public key (**`gpg.key`**) for the repository to enable package signatures to be verified.
   Note: use secure **`https://` for the public key** to ensure secure transfer, but use**`http://` for the repository**, since **`https://`** may not be supported and is not required since the key provides secure authentication of the package files.

   ```
   sudo sh -c "wget -O - https://dl.openfoam.org/gpg.key | apt-key add -"
   sudo add-apt-repository http://dl.openfoam.org/ubuntu
   ```

   ****Note: This only needs to be done once for a given system**

2. Update the **`apt`** package list to account for the new download repository location

   ```
   sudo apt-get update
   ```

3. Install OpenFOAM (9 in the name refers to version 9) which also installs **`paraviewopenfoam56`** as a dependency.

   ```
   sudo apt-get -y install openfoam9
   ```

OpenFOAM 9 and *ParaView 5.6.3* are now installed in the */opt* directory.

How to get OFFBEAT

Open-source online at
https://gitlab.com/foam-for-nuclear/offbeat

Open a terminal (typically inside the path OpenFOAM/<userName>-9/applications/solvers) and clone the repository:

```
git clone https://gitlab.com/foam-for-nuclear/offbeat.git offbeat
```

Move to the new folder and check the stable *master* (or the more advanced *develop*) branch

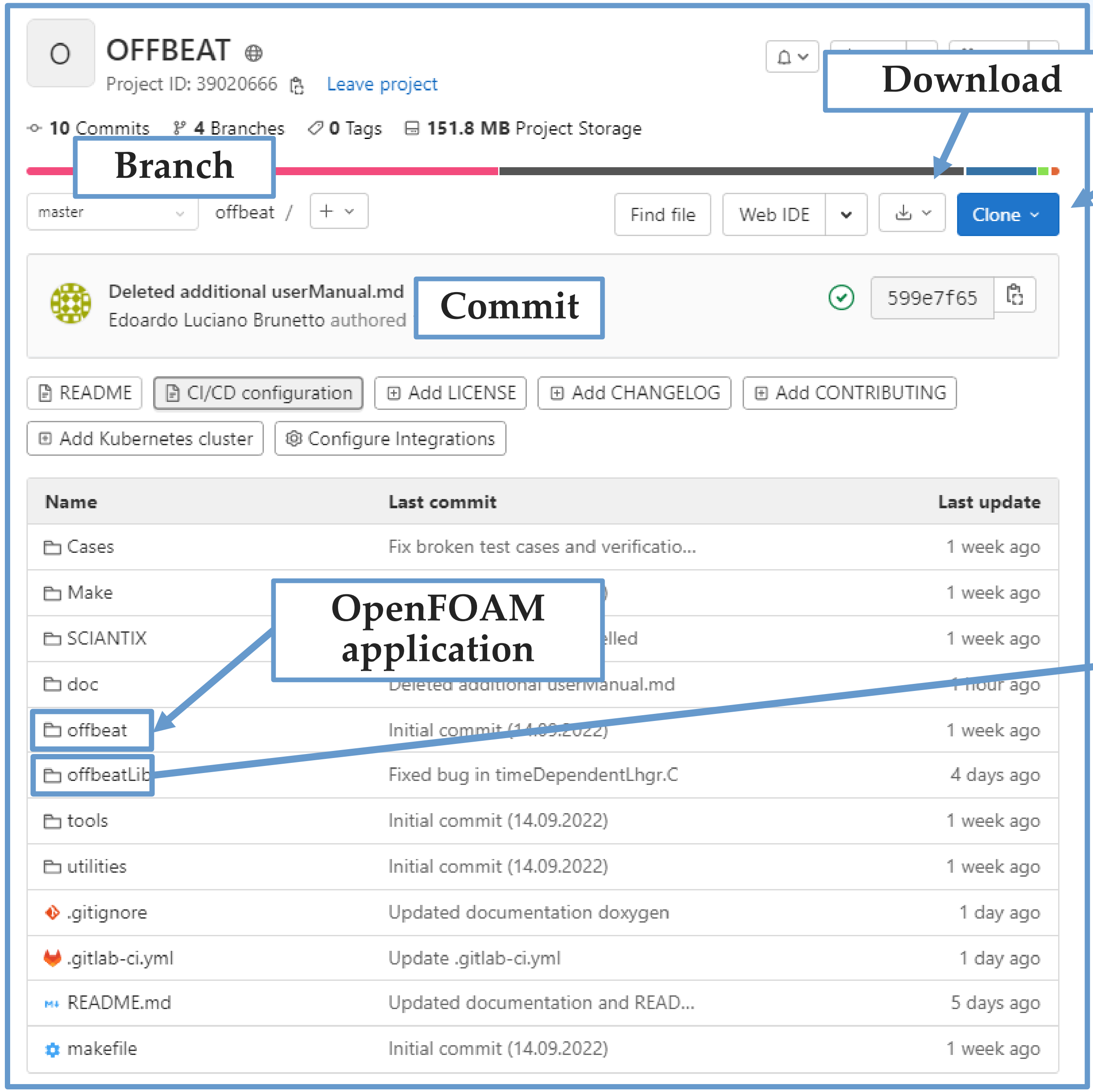
```
cd offbeat
git checkout master
```

Clean (not always necessary) and install

```
make clean
make
```

How to approach OFFBEAT Documentation

A quick look at the repository



OFFBEAT
Project ID: 39020666 Leave project

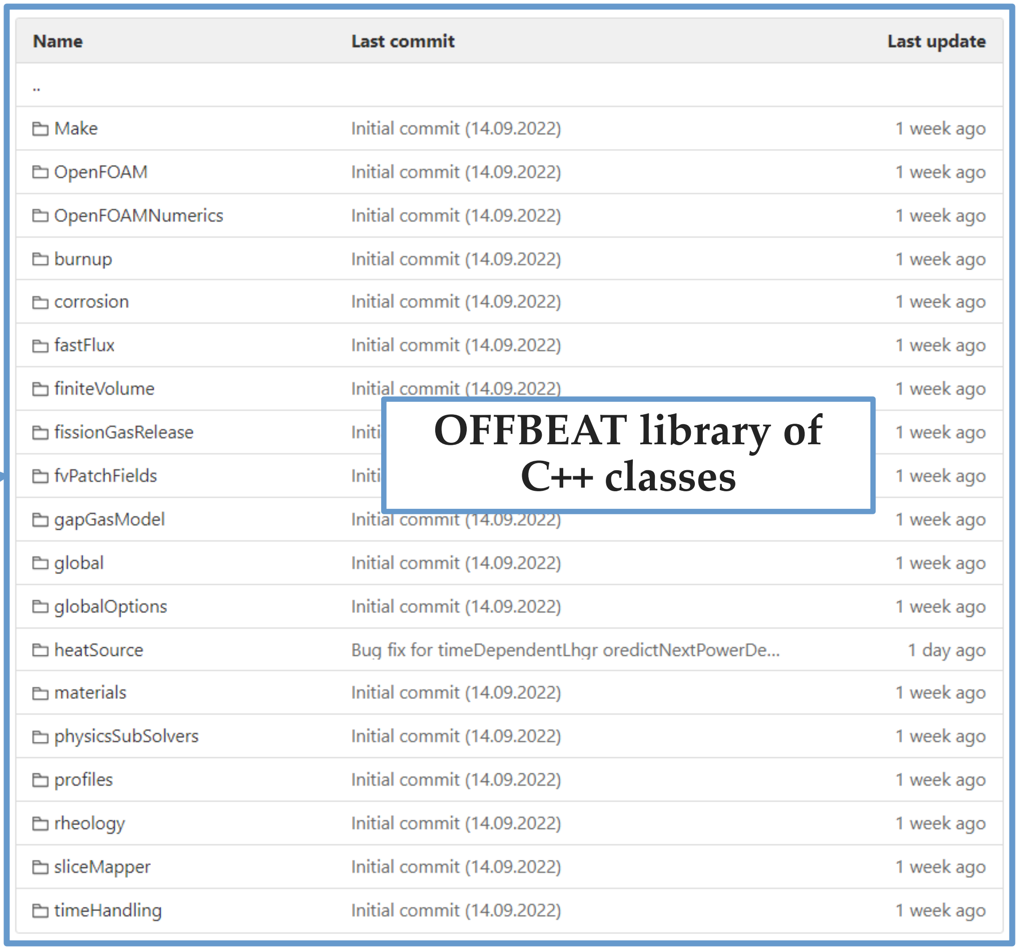
10 Commits 4 Branches 0 Tags 151.8 MB Project Storage

master offbeat / + Find file Web IDE Clone

Deleted additional userManual.md
Edoardo Luciano Brunetto authored
599e7f65

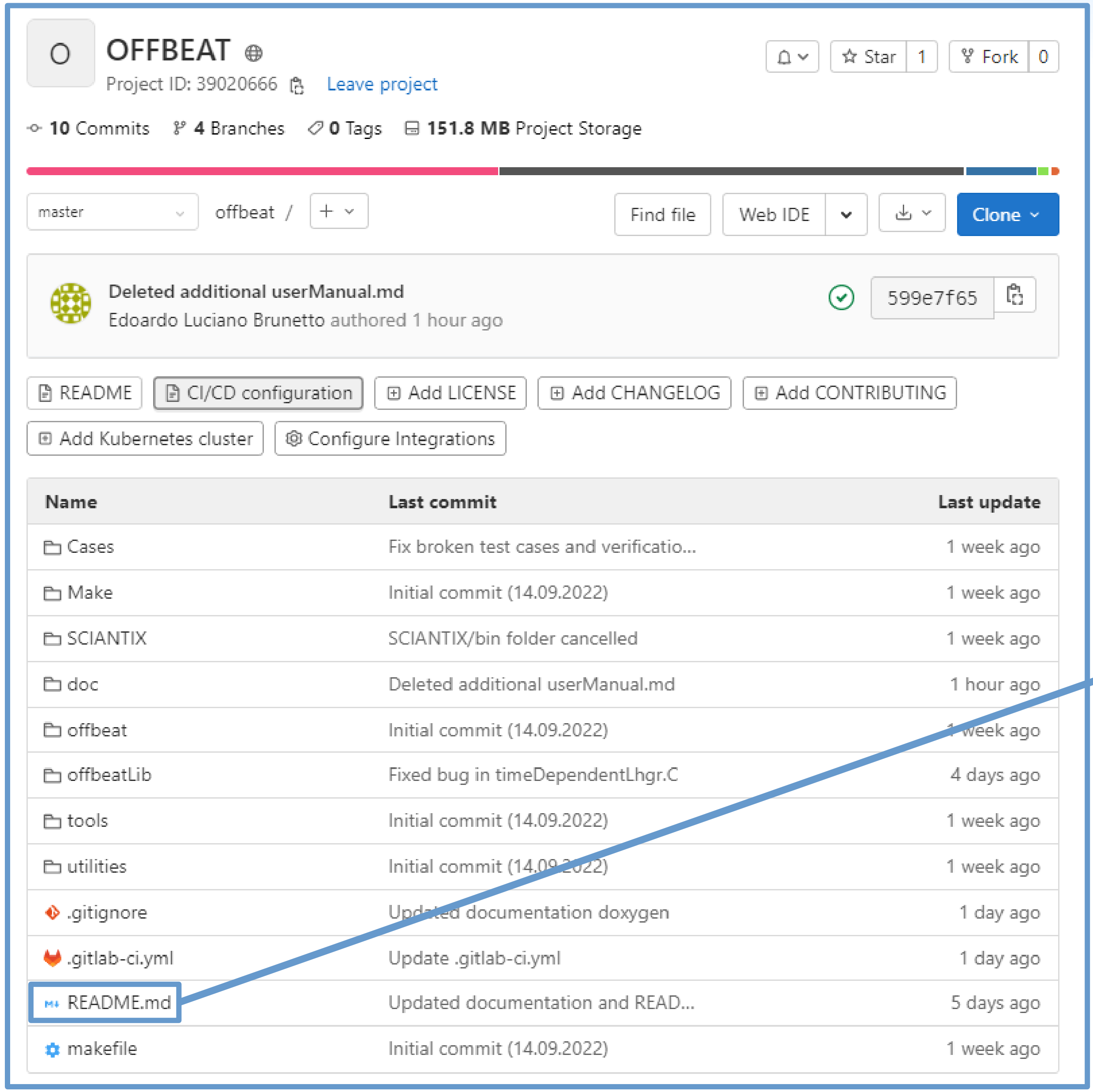
README CI/CD configuration Add LICENSE Add CHANGELOG Add CONTRIBUTING
Add Kubernetes cluster Configure Integrations

Name	Last commit	Last update
Cases	Fix broken test cases and verificatio...	1 week ago
Make	)	1 week ago
SCIANTIX	elled	1 week ago
doc	Deleted additional userManual.md	1 hour ago
offbeat	Initial commit (14.09.2022)	1 week ago
offbeatLib	Fixed bug in timeDependentLhgr.C	4 days ago
tools	Initial commit (14.09.2022)	1 week ago
utilities	Initial commit (14.09.2022)	1 week ago
.gitignore	Updated documentation doxygen	1 day ago
.gitlab-ci.yml	Update .gitlab-ci.yml	1 day ago
README.md	Updated documentation and READ...	5 days ago
makefile	Initial commit (14.09.2022)	1 week ago



Name	Last commit	Last update
..		
Make	Initial commit (14.09.2022)	1 week ago
OpenFOAM	Initial commit (14.09.2022)	1 week ago
OpenFOAMNumerics	Initial commit (14.09.2022)	1 week ago
burnup	Initial commit (14.09.2022)	1 week ago
corrosion	Initial commit (14.09.2022)	1 week ago
fastFlux	Initial commit (14.09.2022)	1 week ago
finiteVolume	Initial commit (14.09.2022)	1 week ago
fissionGasRelease	Initi	1 week ago
fvPatchFields	Initi	1 week ago
gapGasModel	Initial commit (14.09.2022)	1 week ago
global	Initial commit (14.09.2022)	1 week ago
globalOptions	Initial commit (14.09.2022)	1 week ago
heatSource	Bug fix for timeDependentLhgr oredictNextPowerDe...	1 day ago
materials	Initial commit (14.09.2022)	1 week ago
physicsSubSolvers	Initial commit (14.09.2022)	1 week ago
profiles	Initial commit (14.09.2022)	1 week ago
rheology	Initial commit (14.09.2022)	1 week ago
sliceMapper	Initial commit (14.09.2022)	1 week ago
timeHandling	Initial commit (14.09.2022)	1 week ago

Access the documentation from main page



OFFBEAT
Project ID: 39020666 Leave project

Star 1 Fork 0

10 Commits 4 Branches 0 Tags 151.8 MB Project Storage

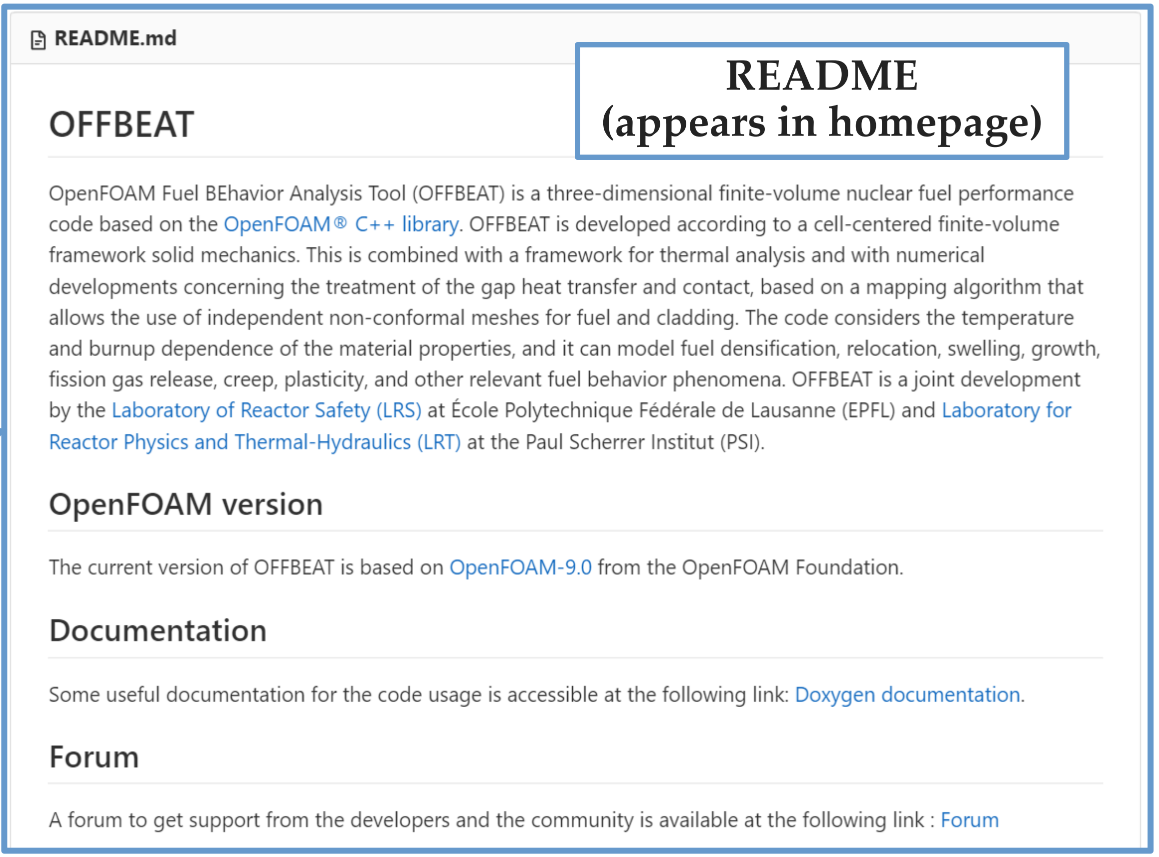
master offbeat / +

Find file Web IDE Clone

Deleted additional userManual.md
Edoardo Luciano Brunetto authored 1 hour ago
599e7f65

README CI/CD configuration Add LICENSE Add CHANGELOG Add CONTRIBUTING
Add Kubernetes cluster Configure Integrations

Name	Last commit	Last update
Cases	Fix broken test cases and verificatio...	1 week ago
Make	Initial commit (14.09.2022)	1 week ago
SCIANTIX	SCIANTIX/bin folder cancelled	1 week ago
doc	Deleted additional userManual.md	1 hour ago
offbeat	Initial commit (14.09.2022)	1 week ago
offbeatLib	Fixed bug in timeDependentLhgr.C	4 days ago
tools	Initial commit (14.09.2022)	1 week ago
utilities	Initial commit (14.09.2022)	1 week ago
.gitignore	Updated documentation doxygen	1 day ago
.gitlab-ci.yml	Update .gitlab-ci.yml	1 day ago
README.md	Updated documentation and READ...	5 days ago
makefile	Initial commit (14.09.2022)	1 week ago



README.md

README
(appears in homepage)

OFFBEAT

OpenFOAM Fuel BEhavior Analysis Tool (OFFBEAT) is a three-dimensional finite-volume nuclear fuel performance code based on the OpenFOAM® C++ library. OFFBEAT is developed according to a cell-centered finite-volume framework solid mechanics. This is combined with a framework for thermal analysis and with numerical developments concerning the treatment of the gap heat transfer and contact, based on a mapping algorithm that allows the use of independent non-conformal meshes for fuel and cladding. The code considers the temperature and burnup dependence of the material properties, and it can model fuel densification, relocation, swelling, growth, fission gas release, creep, plasticity, and other relevant fuel behavior phenomena. OFFBEAT is a joint development by the Laboratory of Reactor Safety (LRS) at École Polytechnique Fédérale de Lausanne (EPFL) and Laboratory for Reactor Physics and Thermal-Hydraulics (LRT) at the Paul Scherrer Institut (PSI).

OpenFOAM version

The current version of OFFBEAT is based on OpenFOAM-9.0 from the OpenFOAM Foundation.

Documentation

Some useful documentation for the code usage is accessible at the following link: Doxygen documentation.

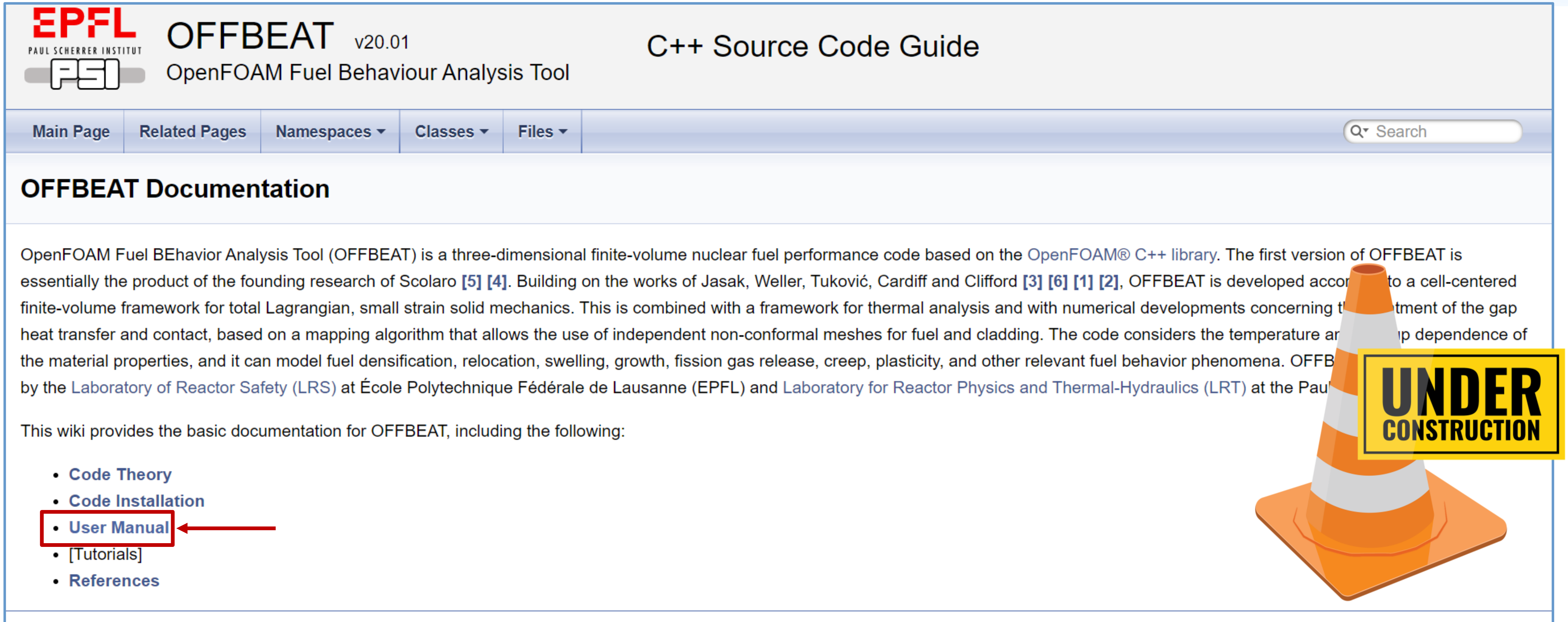
Forum

A forum to get support from the developers and the community is available at the following link : Forum

Documentation

Doxygen generated documentation at https://foam-for-nuclear.gitlab.io/offbeat/index.html

The documentation is constantly improving! Please report issues, typos or suggest corrections/new content!



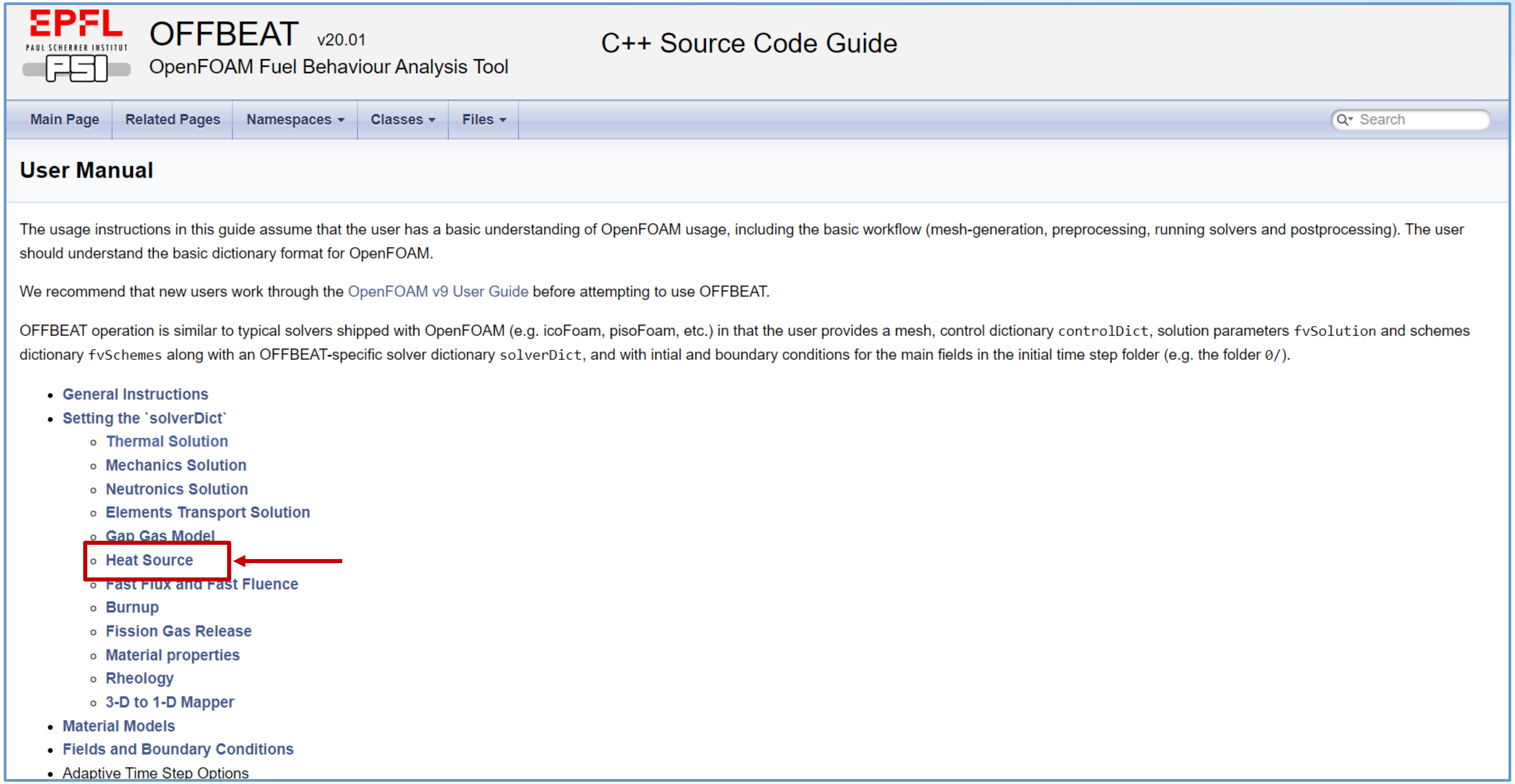
OFFBEAT Documentation

OpenFOAM Fuel BEhavior Analysis Tool (OFFBEAT) is a three-dimensional finite-volume nuclear fuel performance code based on the OpenFOAM® C++ library. The first version of OFFBEAT is essentially the product of the founding research of Scolaro [5] [4]. Building on the works of Jasak, Weller, Tuković, Cardiff and Clifford [3] [6] [1] [2], OFFBEAT is developed accor to a cell-centered finite-volume framework for total Lagrangian, small strain solid mechanics. This is combined with a framework for thermal analysis and with numerical developments concerning t tment of the gap heat transfer and contact, based on a mapping algorithm that allows the use of independent non-conformal meshes for fuel and cladding. The code considers the temperature ar p dependence of the material properties, and it can model fuel densification, relocation, swelling, growth, fission gas release, creep, plasticity, and other relevant fuel behavior phenomena. OFFB by the Laboratory of Reactor Safety (LRS) at École Polytechnique Fédérale de Lausanne (EPFL) and Laboratory for Reactor Physics and Thermal-Hydraulics (LRT) at the Pau

This wiki provides the basic documentation for OFFBEAT, including the following:

- Code Theory
- Code Installation
- User Manual
- [Tutorials]
- References

Most physics and models have a page in the User Manual



OFFBEAT v20.01

OpenFOAM Fuel Behaviour Analysis Tool

C++ Source Code Guide

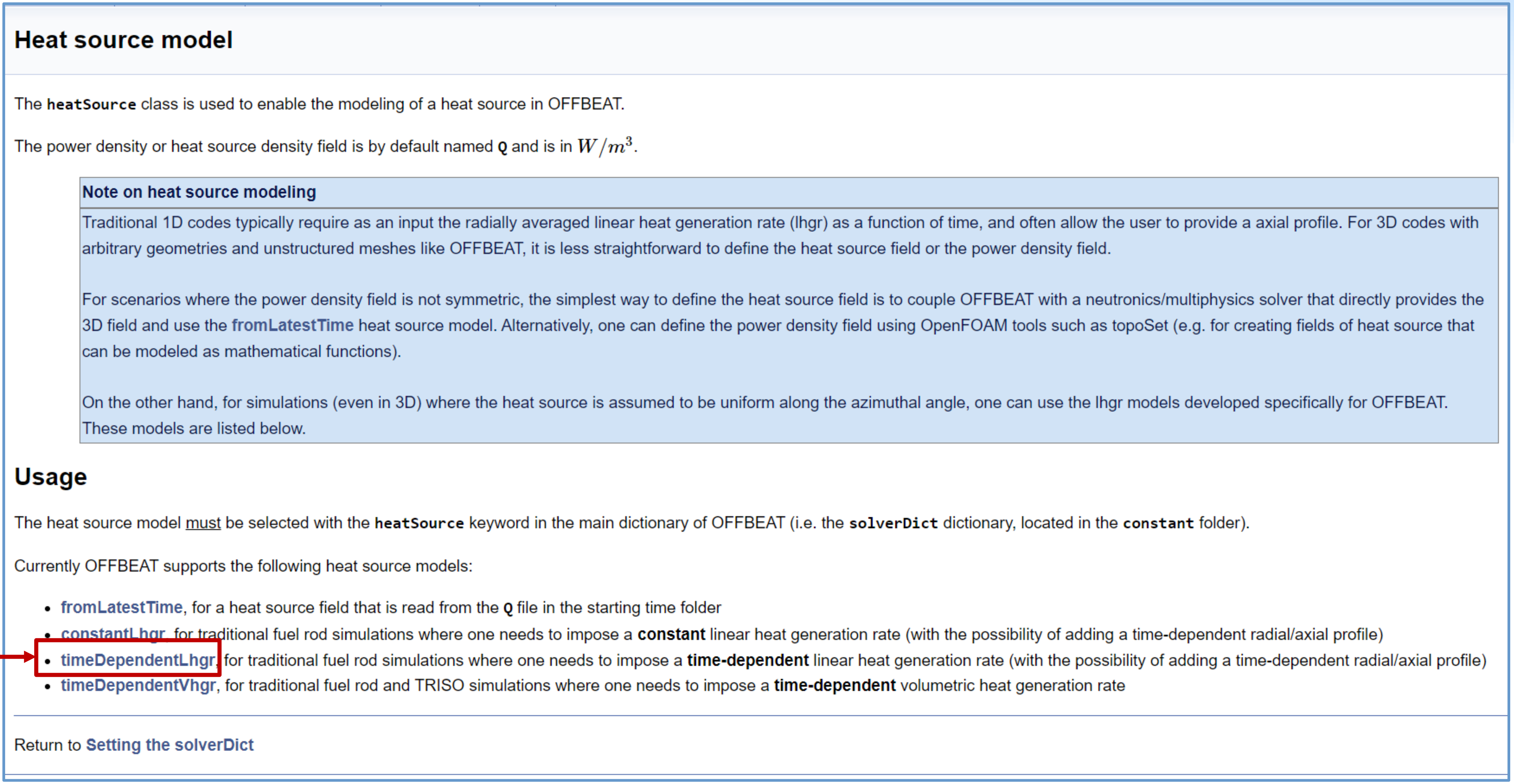
User Manual

The usage instructions in this guide assume that the user has a basic understanding of OpenFOAM usage, including the basic workflow (mesh-generation, preprocessing, running solvers and postprocessing). The user should understand the basic dictionary format for OpenFOAM.

We recommend that new users work through the OpenFOAM v9 User Guide before attempting to use OFFBEAT.

OFFBEAT operation is similar to typical solvers shipped with OpenFOAM (e.g. icoFoam, pisoFoam, etc.) in that the user provides a mesh, control dictionary `controlDict`, solution parameters `fvSolution` and schemes dictionary `fvSchemes` along with an OFFBEAT-specific solver dictionary `solverDict`, and with intial and boundary conditions for the main fields in the initial time step folder (e.g. the folder `0/`).

Most physics and models have a page in the User Manual



Heat source model

The **heatSource** class is used to enable the modeling of a heat source in OFFBEAT.

The power density or heat source density field is by default named **Q** and is in W/m^3.

Note on heat source modeling

Traditional 1D codes typically require as an input the radially averaged linear heat generation rate (lhgr) as a function of time, and often allow the user to provide a axial profile. For 3D codes with arbitrary geometries and unstructured meshes like OFFBEAT, it is less straightforward to define the heat source field or the power density field.

For scenarios where the power density field is not symmetric, the simplest way to define the heat source field is to couple OFFBEAT with a neutronics/multiphysics solver that directly provides the 3D field and use the fromLatestTime heat source model. Alternatively, one can define the power density field using OpenFOAM tools such as topoSet (e.g. for creating fields of heat source that can be modeled as mathematical functions).

On the other hand, for simulations (even in 3D) where the heat source is assumed to be uniform along the azimuthal angle, one can use the lhgr models developed specifically for OFFBEAT. These models are listed below.

Usage

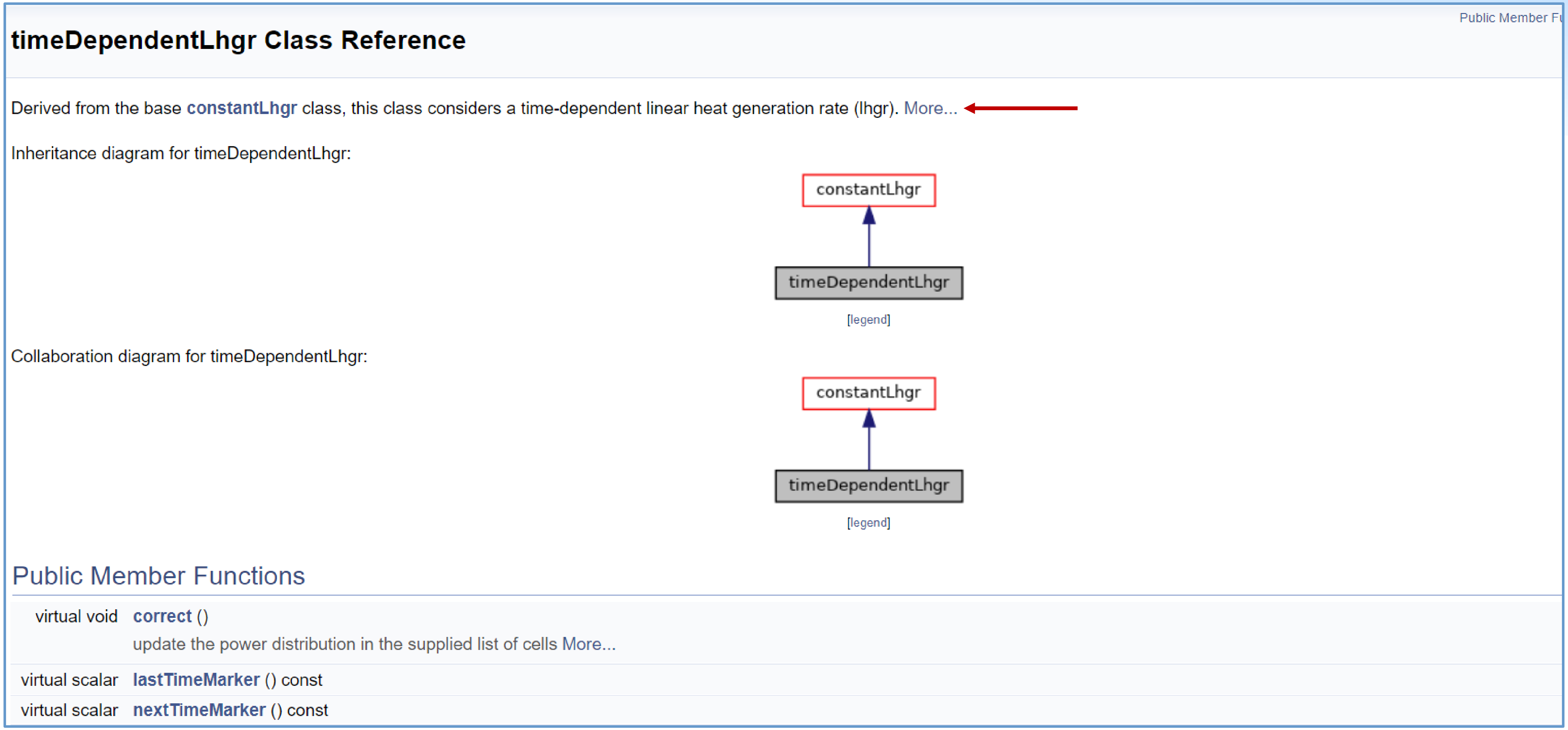
The heat source model must be selected with the **heatSource** keyword in the main dictionary of OFFBEAT (i.e. the **solverDict** dictionary, located in the **constant** folder).

Currently OFFBEAT supports the following heat source models:

- fromLatestTime, for a heat source field that is read from the **Q** file in the starting time folder
- constantLhgr, for traditional fuel rod simulations where one needs to impose a **constant** linear heat generation rate (with the possibility of adding a time-dependent radial/axial profile)
- timeDependentLhgr, for traditional fuel rod simulations where one needs to impose a **time-dependent** linear heat generation rate (with the possibility of adding a time-dependent radial/axial profile)
- timeDependentVhgr, for traditional fuel rod and TRISO simulations where one needs to impose a **time-dependent** volumetric heat generation rate

Return to Setting the solverDict

Most physics and models have a page in the User Manual



Public Member Fu

timeDependentLhgr Class Reference

Derived from the base **constantLhgr** class, this class considers a time-dependent linear heat generation rate (lhgr). More...

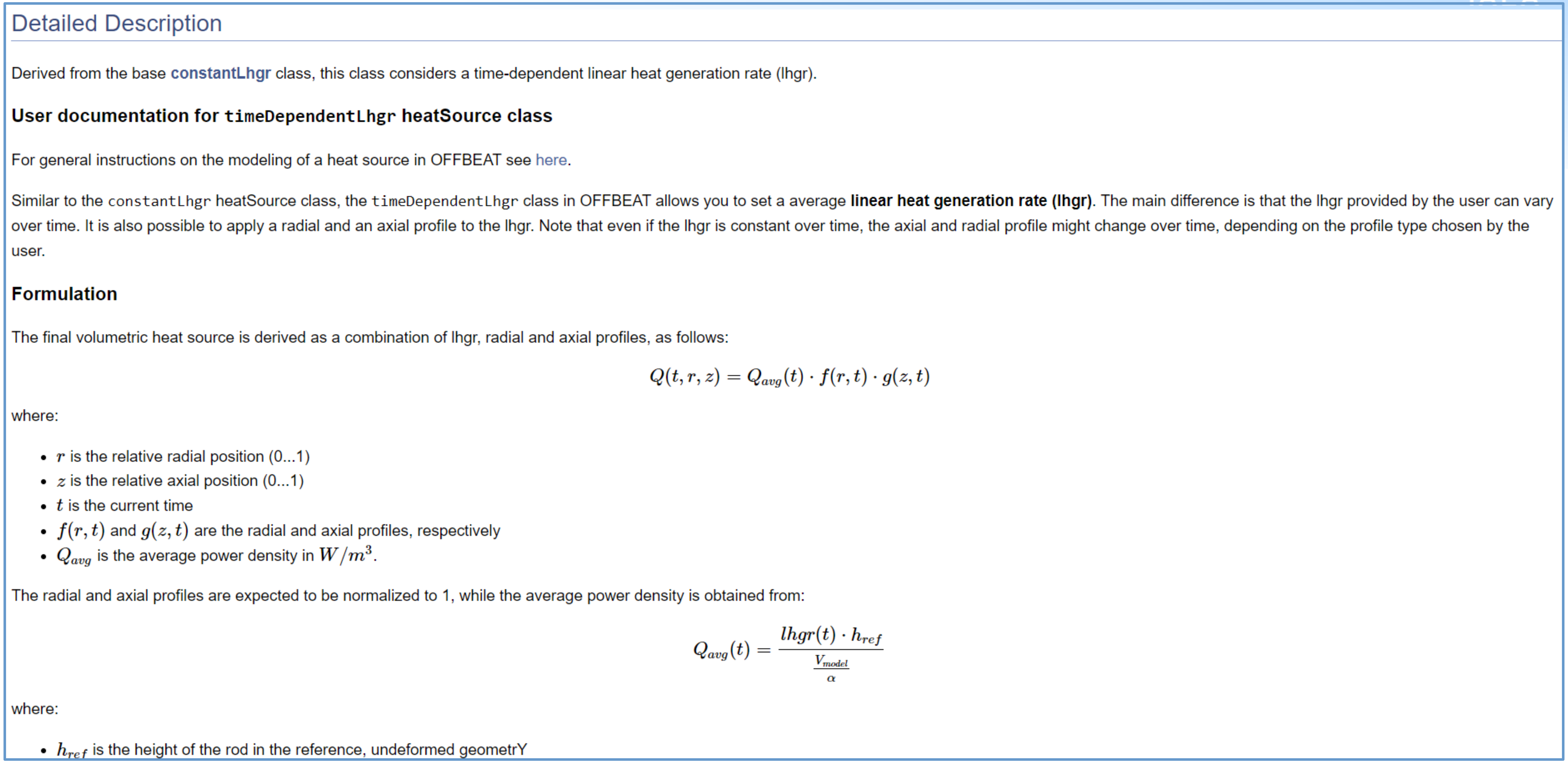
Inheritance diagram for timeDependentLhgr:

Collaboration diagram for timeDependentLhgr:

Public Member Functions

virtual void	**correct** () update the power distribution in the supplied list of cells More...
virtual scalar	**lastTimeMarker** () const
virtual scalar	**nextTimeMarker** () const

Most physics and models have a page in the User Manual



Detailed Description

Derived from the base **constantLhgr** class, this class considers a time-dependent linear heat generation rate (lhgr).

User documentation for `timeDependentLhgr` heatSource class

For general instructions on the modeling of a heat source in OFFBEAT see here.

Similar to the `constantLhgr` heatSource class, the `timeDependentLhgr` class in OFFBEAT allows you to set a average **linear heat generation rate (lhgr)**. The main difference is that the lhgr provided by the user can vary over time. It is also possible to apply a radial and an axial profile to the lhgr. Note that even if the lhgr is constant over time, the axial and radial profile might change over time, depending on the profile type chosen by the user.

Formulation

The final volumetric heat source is derived as a combination of lhgr, radial and axial profiles, as follows:

$$Q(t,r,z) = Q_{avg}(t) \cdot f(r,t) \cdot g(z,t)$$

where:

- r is the relative radial position (0...1)
- z is the relative axial position (0...1)
- t is the current time
- $f(r,t)$ and $g(z,t)$ are the radial and axial profiles, respectively
- Q_{avg} is the average power density in W/m^3.

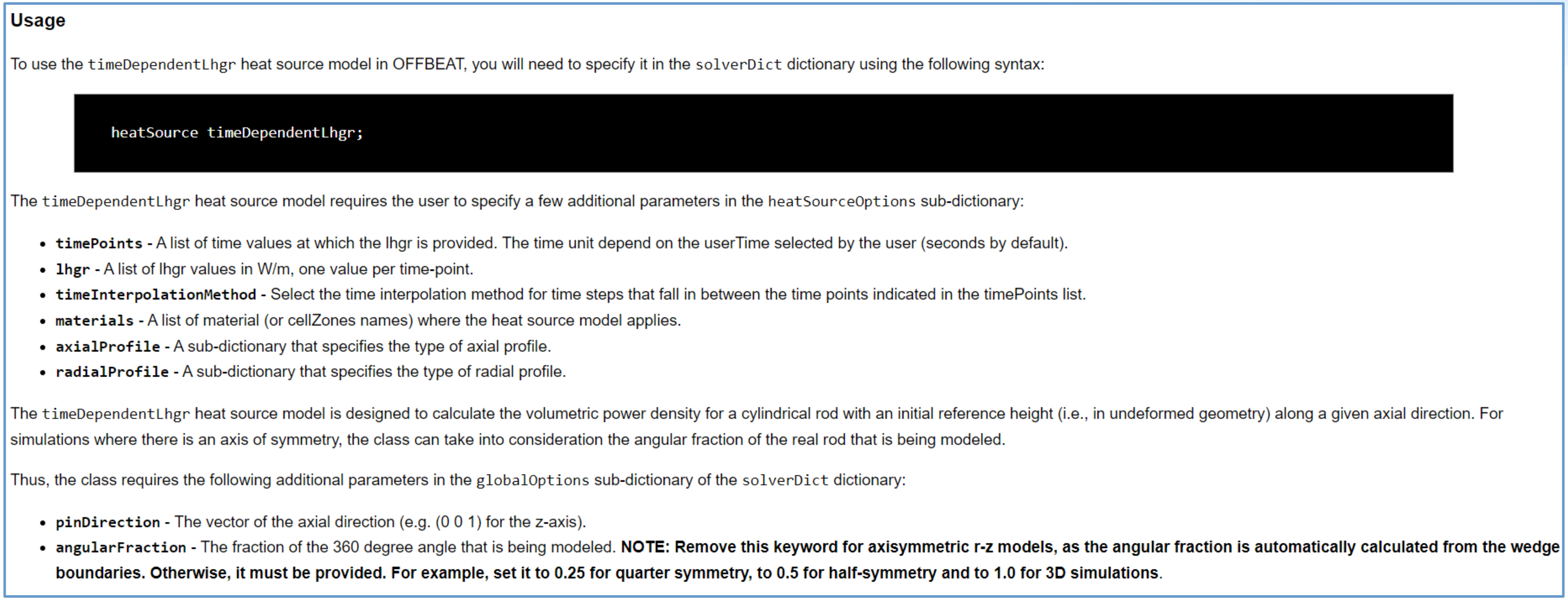
The radial and axial profiles are expected to be normalized to 1, while the average power density is obtained from:

$$Q_{avg}(t) = \frac{lhgr(t) \cdot h_{ref}}{\frac{V_{model}}{\alpha}}$$

where:

- h_{ref} is the height of the rod in the reference, undeformed geometrY

Most physics and models have a page in the User Manual



Usage

To use the `timeDependentLhgr` heat source model in OFFBEAT, you will need to specify it in the `solverDict` dictionary using the following syntax:

```
heatSource timeDependentLhgr;
```

The `timeDependentLhgr` heat source model requires the user to specify a few additional parameters in the `heatSourceOptions` sub-dictionary:

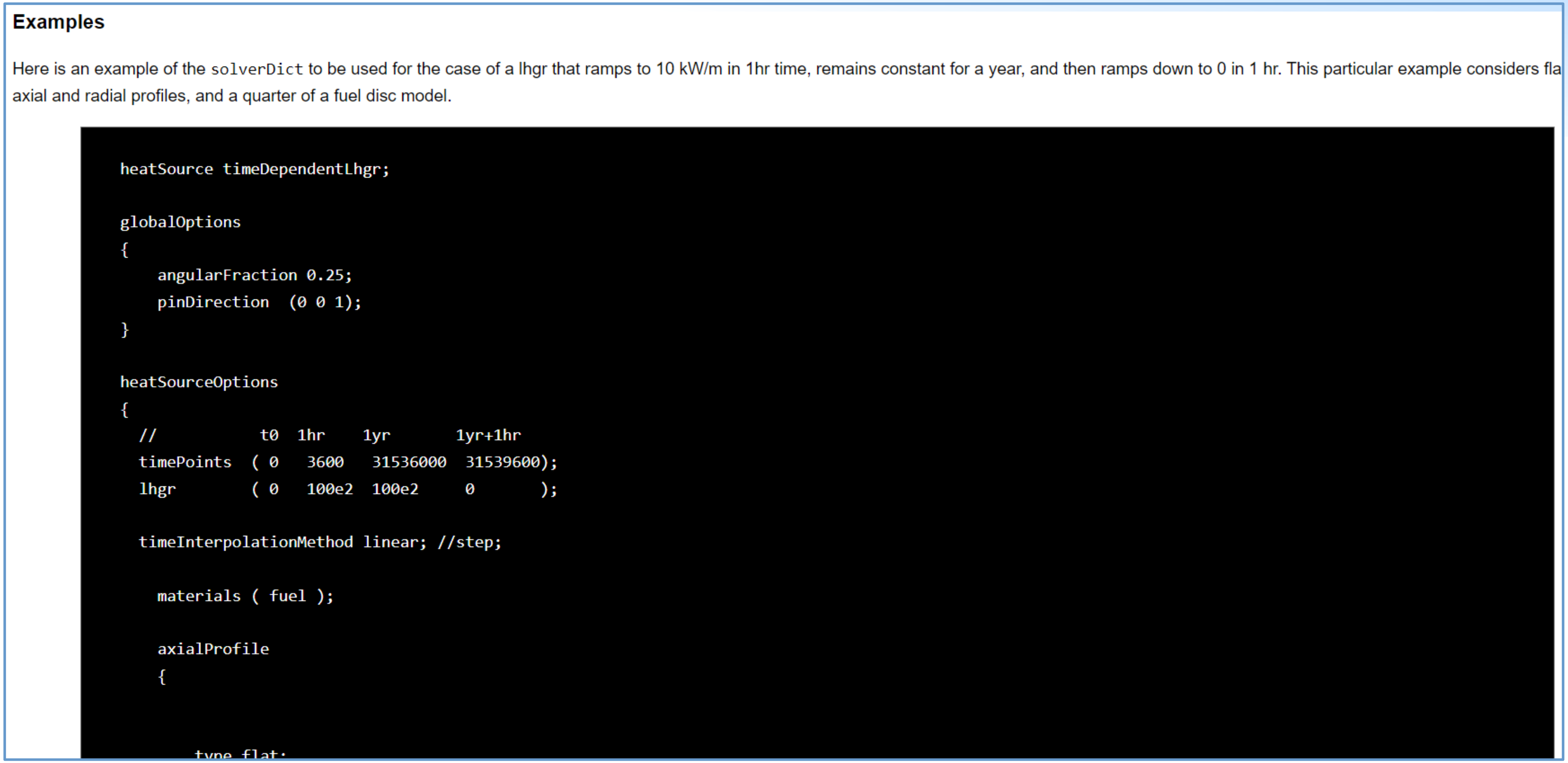
- **`timePoints`** - A list of time values at which the lhgr is provided. The time unit depend on the userTime selected by the user (seconds by default).
- **`lhgr`** - A list of lhgr values in W/m, one value per time-point.
- **`timeInterpolationMethod`** - Select the time interpolation method for time steps that fall in between the time points indicated in the timePoints list.
- **`materials`** - A list of material (or cellZones names) where the heat source model applies.
- **`axialProfile`** - A sub-dictionary that specifies the type of axial profile.
- **`radialProfile`** - A sub-dictionary that specifies the type of radial profile.

The `timeDependentLhgr` heat source model is designed to calculate the volumetric power density for a cylindrical rod with an initial reference height (i.e., in undeformed geometry) along a given axial direction. For simulations where there is an axis of symmetry, the class can take into consideration the angular fraction of the real rod that is being modeled.

Thus, the class requires the following additional parameters in the `globalOptions` sub-dictionary of the `solverDict` dictionary:

- **`pinDirection`** - The vector of the axial direction (e.g. (0 0 1) for the z-axis).
- **`angularFraction`** - The fraction of the 360 degree angle that is being modeled. **NOTE: Remove this keyword for axisymmetric r-z models, as the angular fraction is automatically calculated from the wedge boundaries. Otherwise, it must be provided. For example, set it to 0.25 for quarter symmetry, to 0.5 for half-symmetry and to 1.0 for 3D simulations**.

Most physics and models have a page in the User Manual



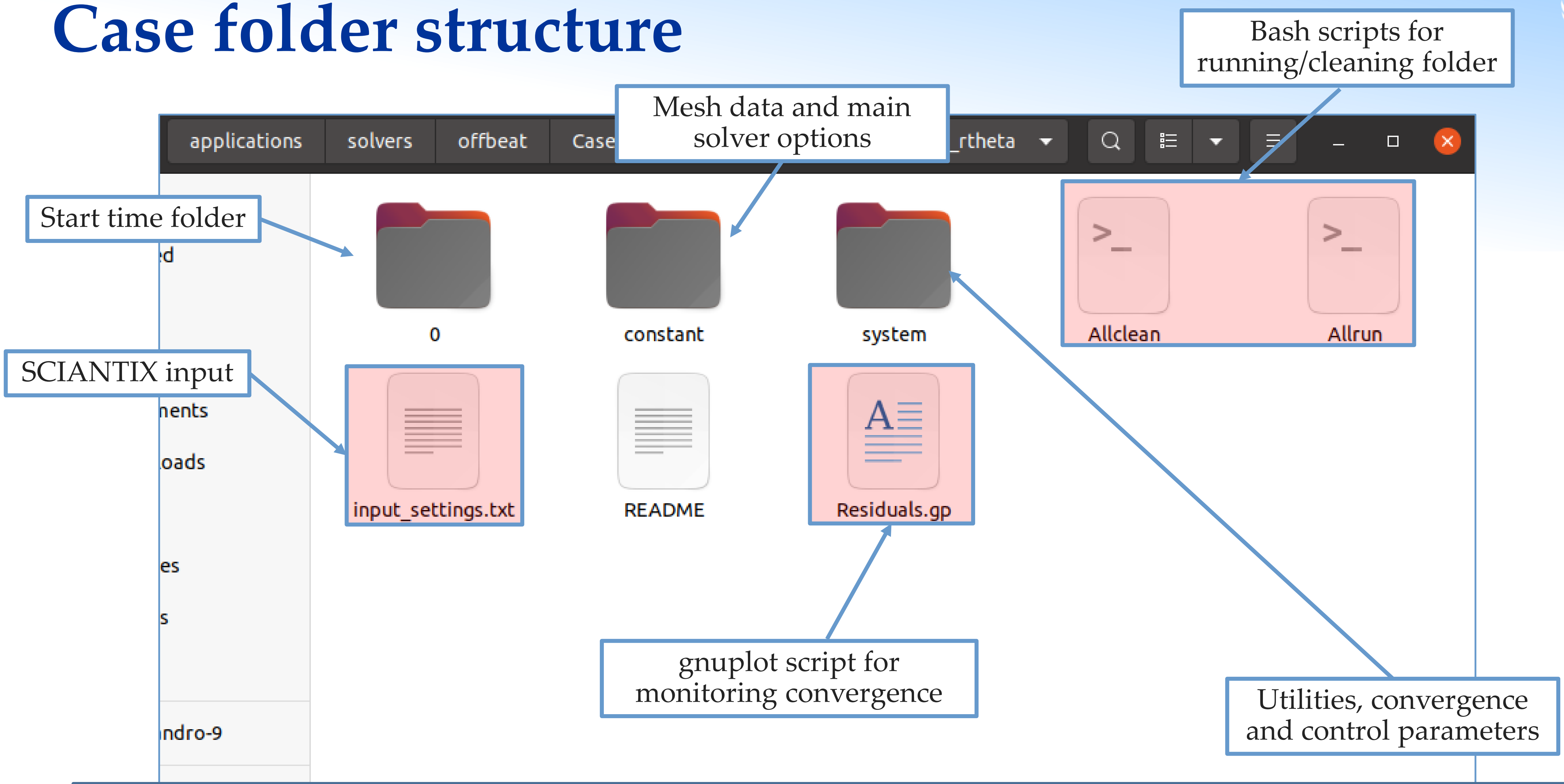
Examples

Here is an example of the `solverDict` to be used for the case of a lhgr that ramps to 10 kW/m in 1hr time, remains constant for a year, and then ramps down to 0 in 1 hr. This particular example considers fla axial and radial profiles, and a quarter of a fuel disc model.

```
heatSource timeDependentLhgr;

globalOptions
{
    angularFraction 0.25;
    pinDirection  (0 0 1);
}

heatSourceOptions
{
  //           t0  1hr    1yr       1yr+1hr
  timePoints  ( 0   3600   31536000  31539600);
  lhgr        ( 0   100e2  100e2     0       );

  timeInterpolationMethod linear; //step;

    materials ( fuel );

    axialProfile
    {


        type flat;
```

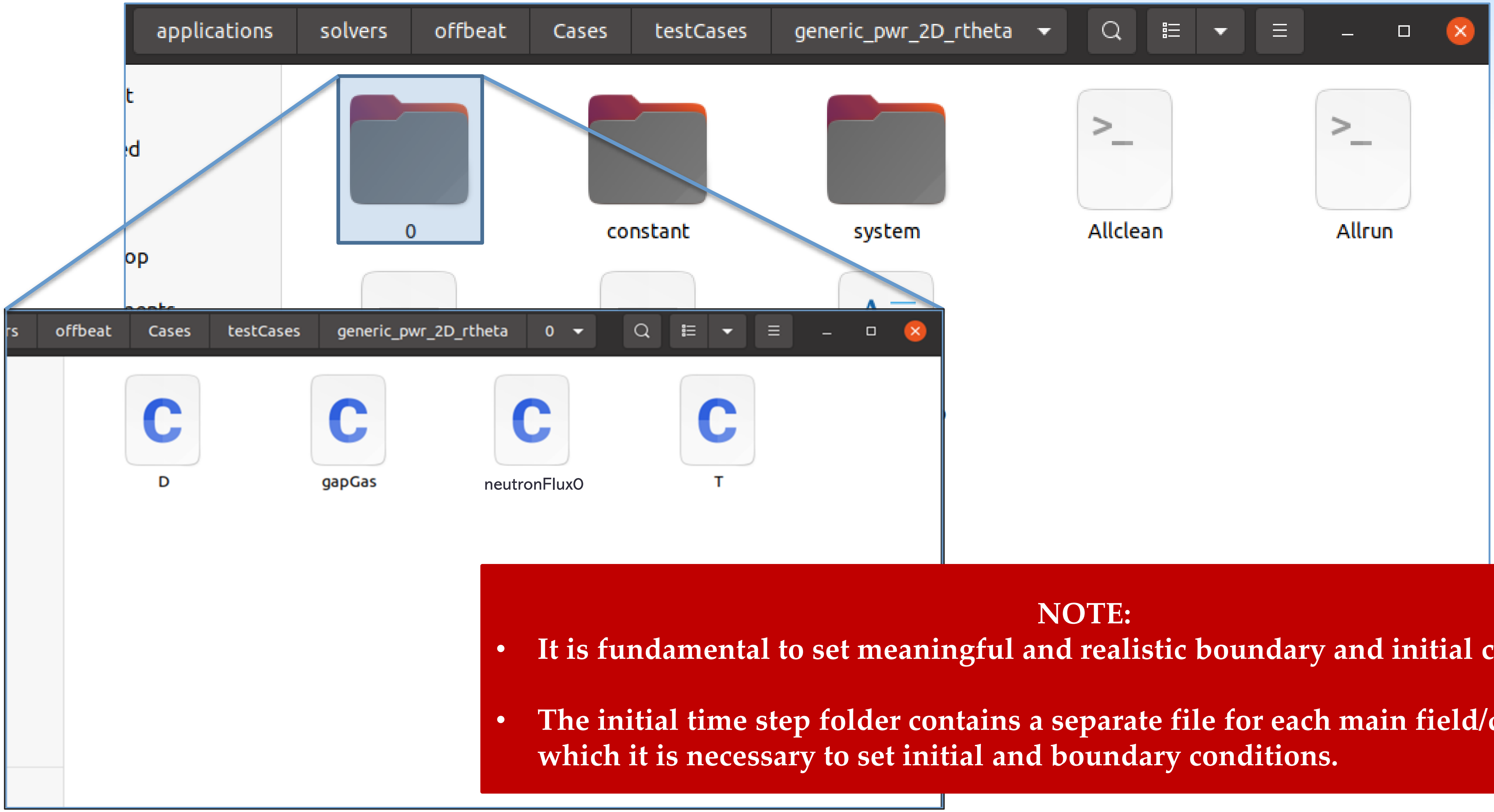
Case folder

Case folder structure


Bash scripts for running/cleaning folder
Mesh data and main solver options
applications
solvers
offbeat
Case
_rtheta
Start time folder
0
constant
system
Allclean
Allrun
SCIANTIX input
input_settings.txt
README
Residuals.gp
gnuplot script for monitoring convergence
Utilities, convergence and control parameters

The `Allrun` and `Allclean` bash scripts can be used to run the case or clean the folder…
… but in the next slides we will unpack each command from mesh creation to data post-processing!

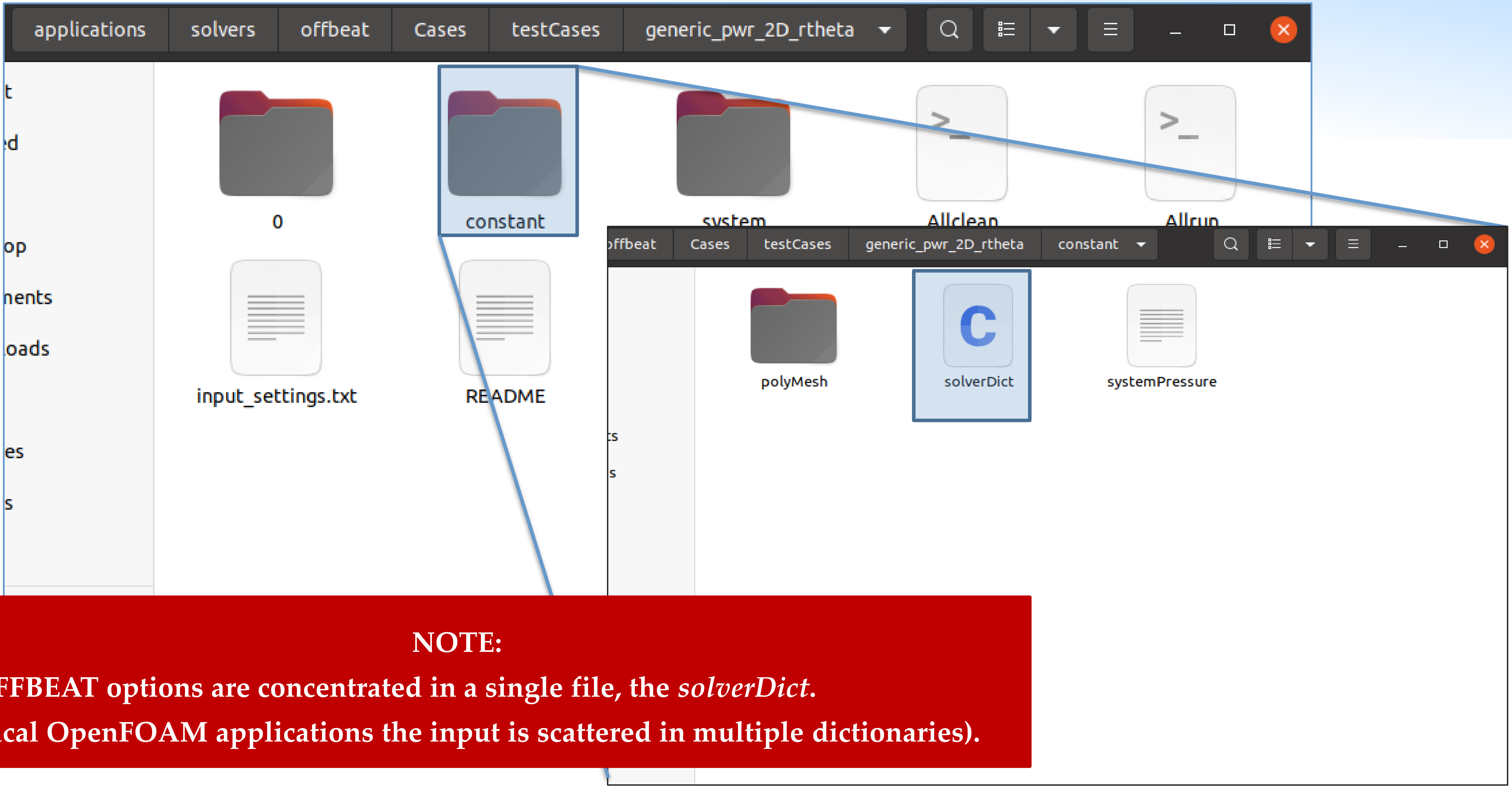
Case folder structure: the initial time folder



NOTE:

- It is fundamental to set meaningful and realistic boundary and initial conditions.
- The initial time step folder contains a separate file for each main field/quantity for which it is necessary to set initial and boundary conditions.

Case folder structure: the solverDict



NOTE:

Most OFFBEAT options are concentrated in a single file, the *solverDict*.

(for typical OpenFOAM applications the input is scattered in multiple dictionaries).

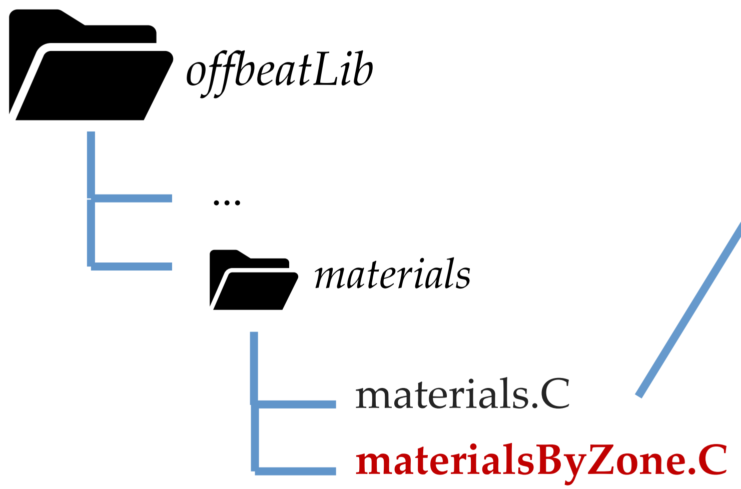
Case folder structure: the solverDict

3 main sections:

- Global options for selecting physics and models;
- Separate dictionaries for most physics and models;
- A materials dictionary for material properties, rheology etc.

```
// Thermal and Mechanical solver selection:
thermalSolver               solidConduction;
mechanicsSolver             smallStrain;
neutronicsSolver            diffusion

// Material and rhelogy treatment:
materialProperties          byZone;

…

mechanicsSolverOptions
{
   forceSummary on;
   …
}

…

materials
{
   fuel
   {
      …
   }
}
```

How to model multiple materials in a single mesh?



One of the most fundamental OFFBEAT classes:

- Handles material properties and behavioral models.
- Stores material addressing (accessed by most other OFFBEAT classes).

$$\begin{tikzcd} \textbf{Materials} \arrow[r, shift left] & \textbf{cellZones} \arrow[l, shift left] \end{tikzcd}$$

collection of cell IDs with the same name (e.g. *fuel* or *cladding*)

IMPORTANT!

- **Make sure that you can separate mesh into cellZones (might be called in different ways e.g. *groups* in Salome).**
- **The mesh creation utility (e.g. *gmshToFoam*) takes care of converting the format**

How to select material properties?

Example (pseudo-input)

```
materials
{
    'fuel.*'
    {
        material UO2;
        …
    }
    cladding
    {
        material zircaloy;
        …
    }
    …
}
```

Subdictionary inside *$CASE/constant/solverDict:*

- One (and only one) subdictionary per zone.

cellZone naming:

- No specific rules (case sensitive).
- No specific order in the *materials* dictionary
- Collective **wild cards** are possible
  (e.g. several cellzones with identical portion of the name such as *fuel1*, *fuel2*, *fuel3* etc.).

Define a *constant* material

```
// List of materials, one per cellZone.
materials
{
    ...

    cladding
    {
        material constant;

        rho         rho      [1 -3 0 0 0]    6560;
        Cp          Cp       [0 2 -2 -1 0]   285;
        k           k        [1 1 3 -1 0]    21.5;
        emissivity emissivity [0 0 0 0 0]    0.808642;
        E           E        [1 -1 -2 0 0]   9.93e10;
        nu          nu       [0 0 0 0 0]     0.37;
        G           G        [1 -1 -2 0 0]   3.6241e+10;
        alpha       alpha    [0 0 0 0 0]     6e-6;
        Tref        Tref     [0 0 0 1 0]     300;

        ...
    }

    ....
}
```

- All thermo-mechanical properties must be provided.
- Material properties do not evolve over time.
- Behavioral models are not taken into account.

Select a preset material

Example (pseudo-input)

```
materials
{
      fuel
      {
              material UO2;

              densityModel              UO2Constant;
              heatCapacityModel         UO2MATPRO;
              conductivityModel         UO2MATPRO;

              densificationModel        UO2FRAPCON;
              resinteringDensityChange 0.1;

              swellingModel             none;
              …
      }
      …
```

- Available materials:
  - UO_2
  - MOX
  - Zircaloy
  - A few others (molybdenum, Inconel600)
- Specific correlations and models selected by default.
- Different correlations and models can be selected with the keyword **'$property_name$Model'**
- Often behavioral models (even if the default ones) might require **additional input (relocation and densification in particular, check the documentation)**
- **Behavioral models can be switched off.**

Test cases
(check the `develop` branch)

Graphite stringer - overview

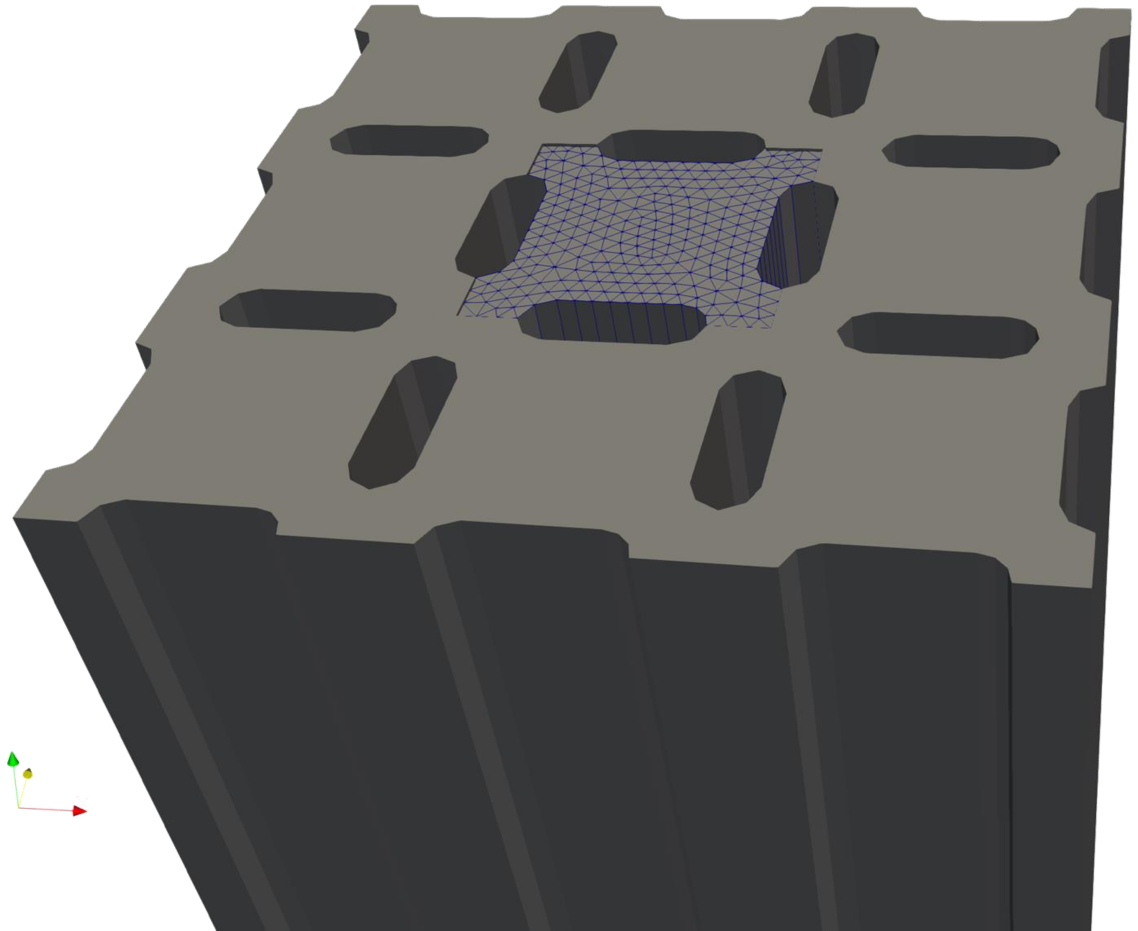
- Complex shape derived from MSRE:
  - Fraction of power released in graphite
  - Interest to know temperature distribution

- Triangular prisms mesh form Salome (NETGEN algorithm)

- Transient simulation (from 0 to 100 s)

```
ddtSchemes
{
    default          Euler;
}
```

- NOTE: power density increased 100x w.r.t MSRE



Graphite stringer - overview

- Purely thermal case (in **`constant/solverDict`**)

```
thermalSolver       solidConduction;
mechanicsSolver     fromLatestTime;
neutronicsSolver    fromLatestTime;
elementTransport    fromLatestTime;

materialProperties  byZone;
rheology            byMaterial;

// With fromLatestTime the power Q will be read
// directly from the 0/ folder
heatSource          fromLatestTime;
```

- Constant graphite properties (in **`constant/solverDict`**)

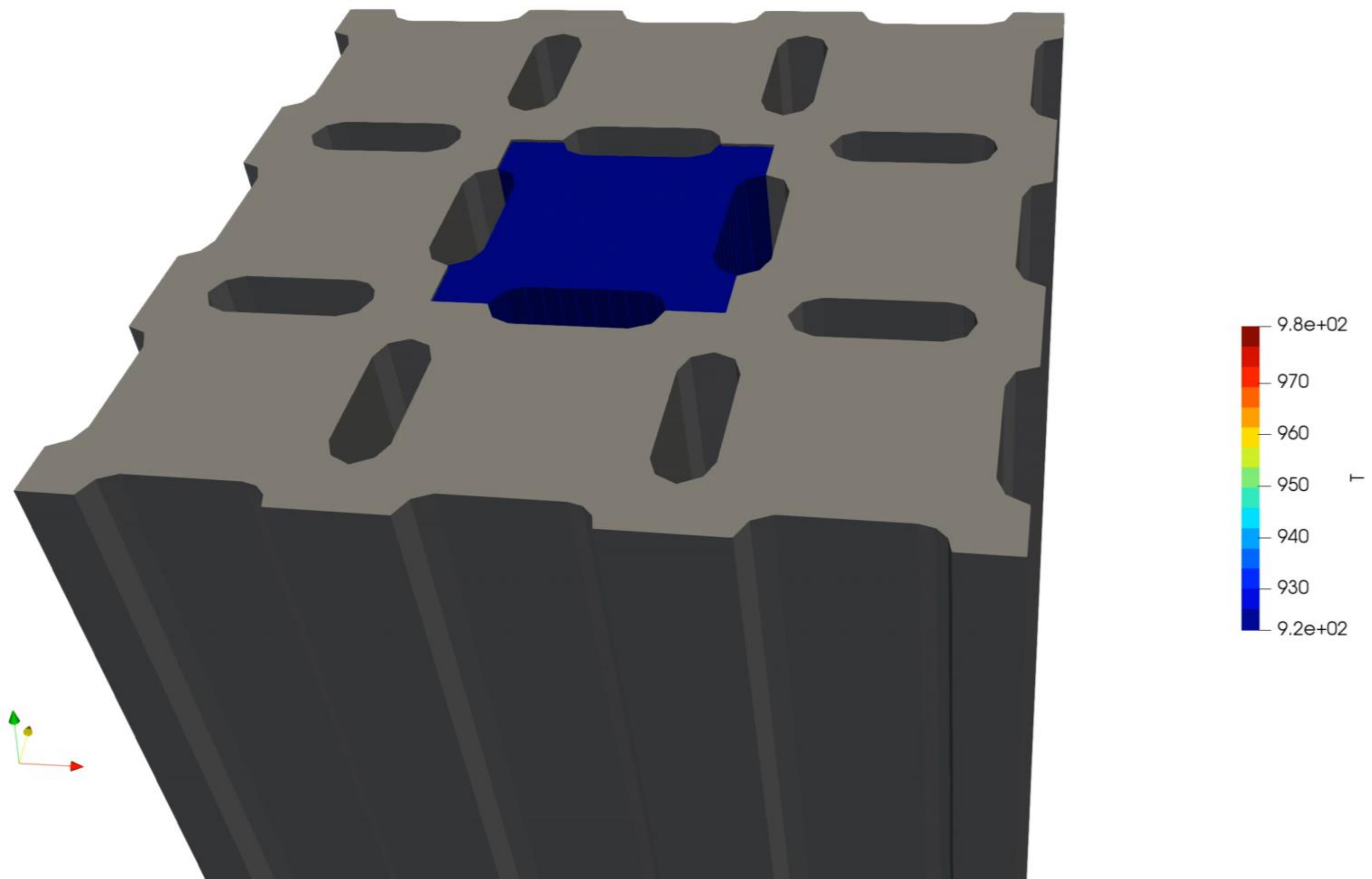
```
materials
{
    stringer
    {
        material                constant;

        // Values from Zanetti et al (2015)
        // At https://doi.org/10.1016/j.pnucene.2015.02.014
        rho         rho     [1 -3 0 0 0]    1874;
        Cp          Cp      [0 2 -2 -1 0]   1772;
        k           k       [1 1 3 -1 0]    53;
```

- Non-uniform **`Q`** field (in **`0/`** folder) with **`codeStream`**

```
internalField  #codeStream
{
    code
    #{
        const IOdictionary& d = static_cast<const IOdictionary&>(dict);
        const fvMesh& mesh = refCast<const fvMesh>(d.db());
        scalarField fld(mesh.nCells(), 2.5e7);
        const vectorField& C(mesh.C());
        scalar pi(Foam::constant::mathematical::pi);

        forAll(fld, i)
        {
            scalar Cz(C[i].z());

            fld[i] *= cos((Cz-1)*pi/2 );
        }

        writeEntry(os, "", fld);
    #};
```

- Fixed-value BC could be changed for more realistic BCs or coupling with TH

Graphite stringer - result



Copy folder from
`Cases/testCases/graphite_stringer`

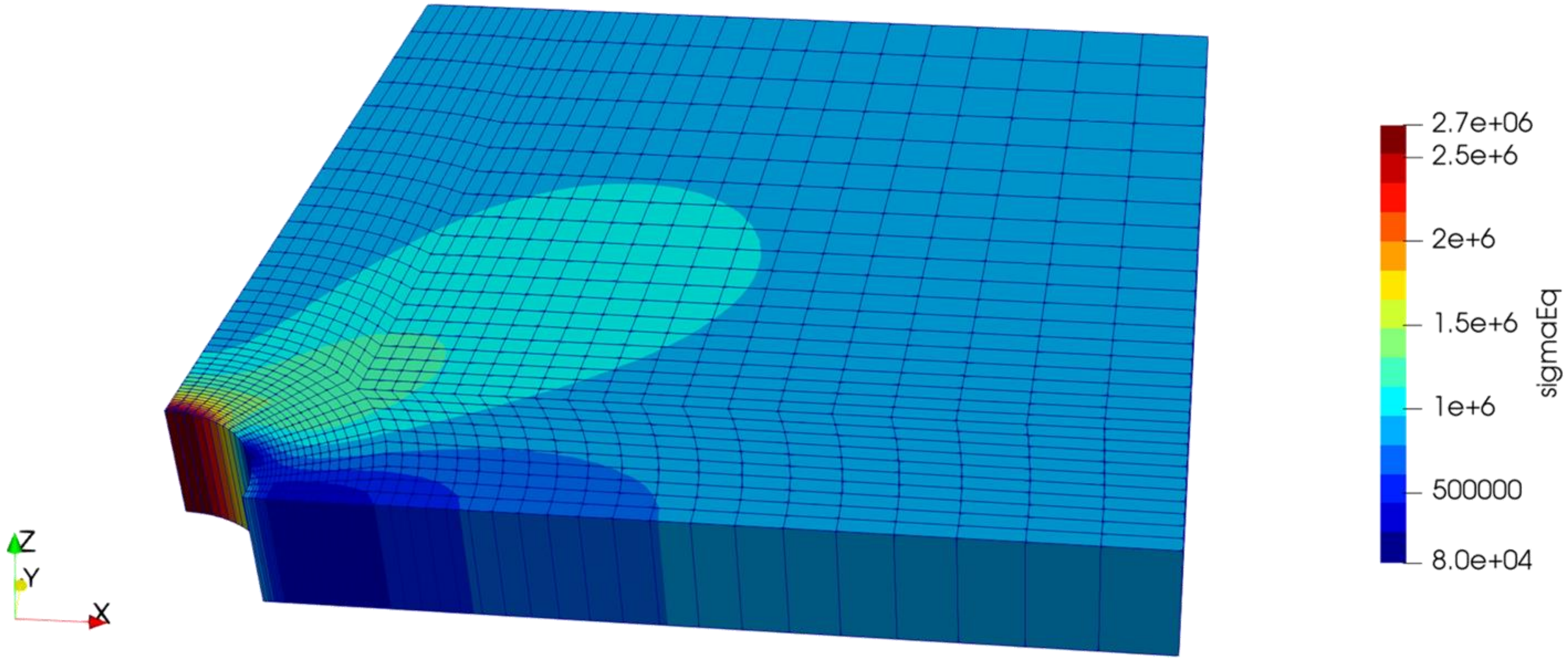
Use **`Allrun`** script or

- Create mesh with:
  `ideasUnvToFoam stringer.unv`
- Run the case with:
  `offbeat`

Check results with:
`paraFoam`

Mechanics-only cases

- For mechanics-only test cases check the folder **`Cases/Verification:`**
- **`multiMaterial/multiMaterialBlock`** and **`multiMaterial/multiMaterialCylinder`**
- **`thickCylinderExpansion`**
- **`plateHole`**



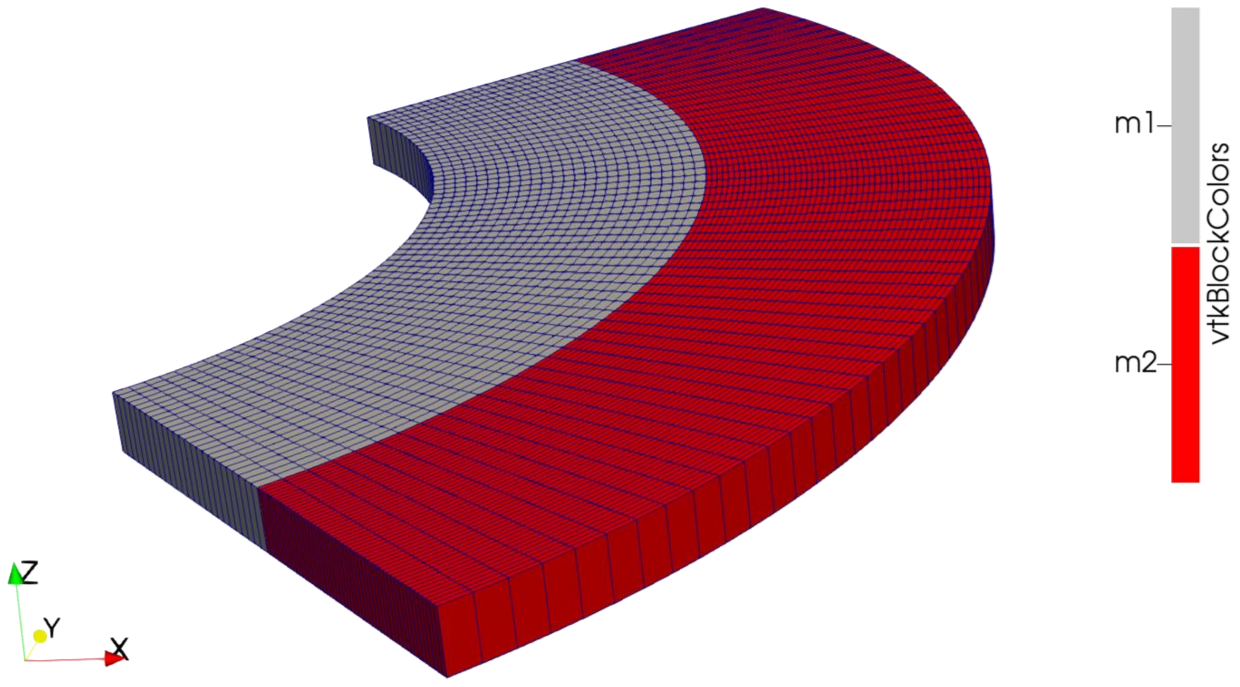
Hot bi-material cylinder - overview

- Bi-material cylinder subjected to temperature and pressure gradient from inner (800 K, 1e7 Pa) to outer surface (300 K, 1e5 Pa).

- Thermo-mechanical case (in **`constant/solverDict`**)

```
//- Select the physics to solve for
thermalSolver              solidConduction;
mechanicsSolver            smallStrain;
neutronicsSolver           fromLatestTime;
elementTransport           fromLatestTime;
```

- Two cellzones with different material properties defined in **`constant/solverDict.materials`**



Hot bi-material cylinder - overview

- Bi-material cylinder subjected to temperature and pressure gradient from inner (800 K, 1e7 Pa) to outer surface (300 K, 1e5 Pa).

- Thermo-mechanical case (in **`constant/solverDict`**)

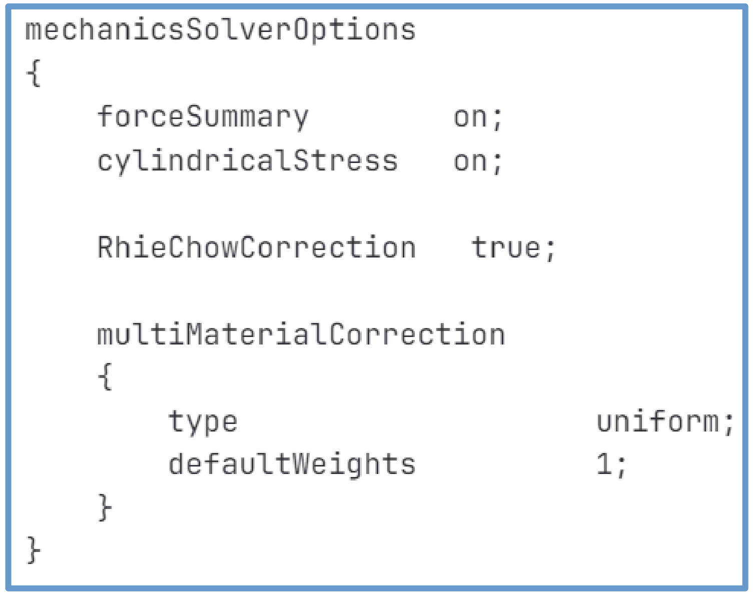
```
//- Select the physics to solve for
thermalSolver             solidConduction;
mechanicsSolver           smallStrain;
neutronicsSolver          fromLatestTime;
elementTransport          fromLatestTime;
```

- Two cellzones with different material properties defined in **`constant/solverDict.materials`**

```
materials
{
    inner
    {
        material    constant;
        rho         "rho" [1 -3 0 0 0] 8000.0;
        Cp          "Cp" [0 0 0 0 0] 1000.0;
        k           "k" [0 0 0 0 0] 50;
        alpha       "alpha" [0 0 0 -1 0] 5e-6;
        emissivity  "emissivity" [0 0 0 0 0] 0.7;
        E           "E" [0 0 0 0 0] 200e9;
        nu          "nu" [0 0 0 0 0] 0.3;
        Tref        "Tref" [0 0 0 1 0] 300;

        rheologyModel               elasticity;
    }

    outer
    {
        material    constant;
        rho         "rho" [1 -3 0 0 0] 8000.0;
        Cp          "Cp" [0 0 0 0 0] 1000.0;
        k           "k" [0 0 0 0 0] 20;
        alpha       "alpha" [0 0 0 -1 0] 5e-6;
        emissivity  "emissivity" [0 0 0 0 0] 0.7;
        E           "E" [0 0 0 0 0] 50e9;
        nu          "nu" [0 0 0 0 0] 0.35;
        Tref        "Tref" [0 0 0 1 0] 300;

        rheologyModel               elasticity;
    }
}
```

Hot bi-material cylinder – multiMaterialCorrection

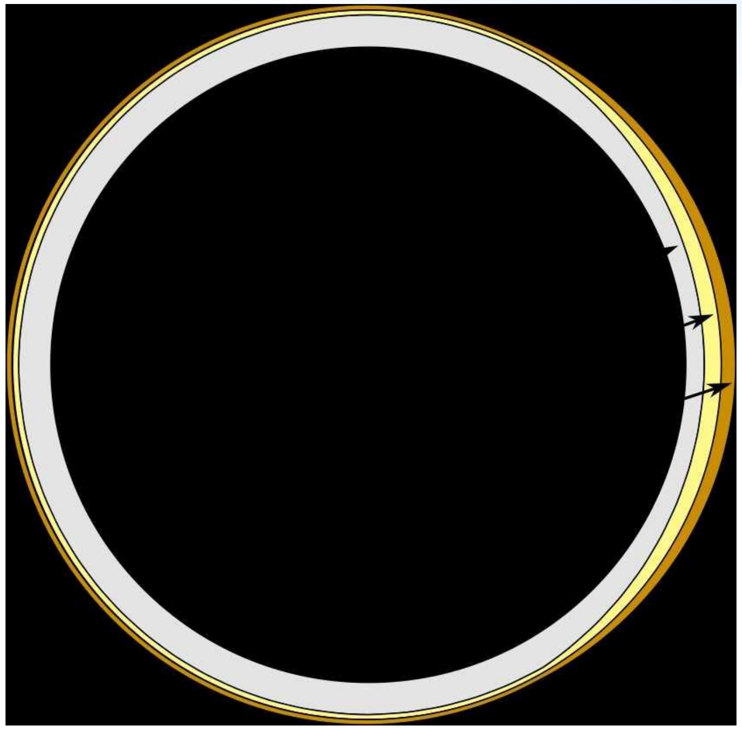
- **`multiMaterialCorrection`** must be activated (in **`constant/solverDict`**)



```
mechanicsSolverOptions
{
    forceSummary        on;
    cylindricalStress   on;

    RhieChowCorrection   true;

    multiMaterialCorrection
    {
        type                uniform;
        defaultWeights      1;
    }
}
```

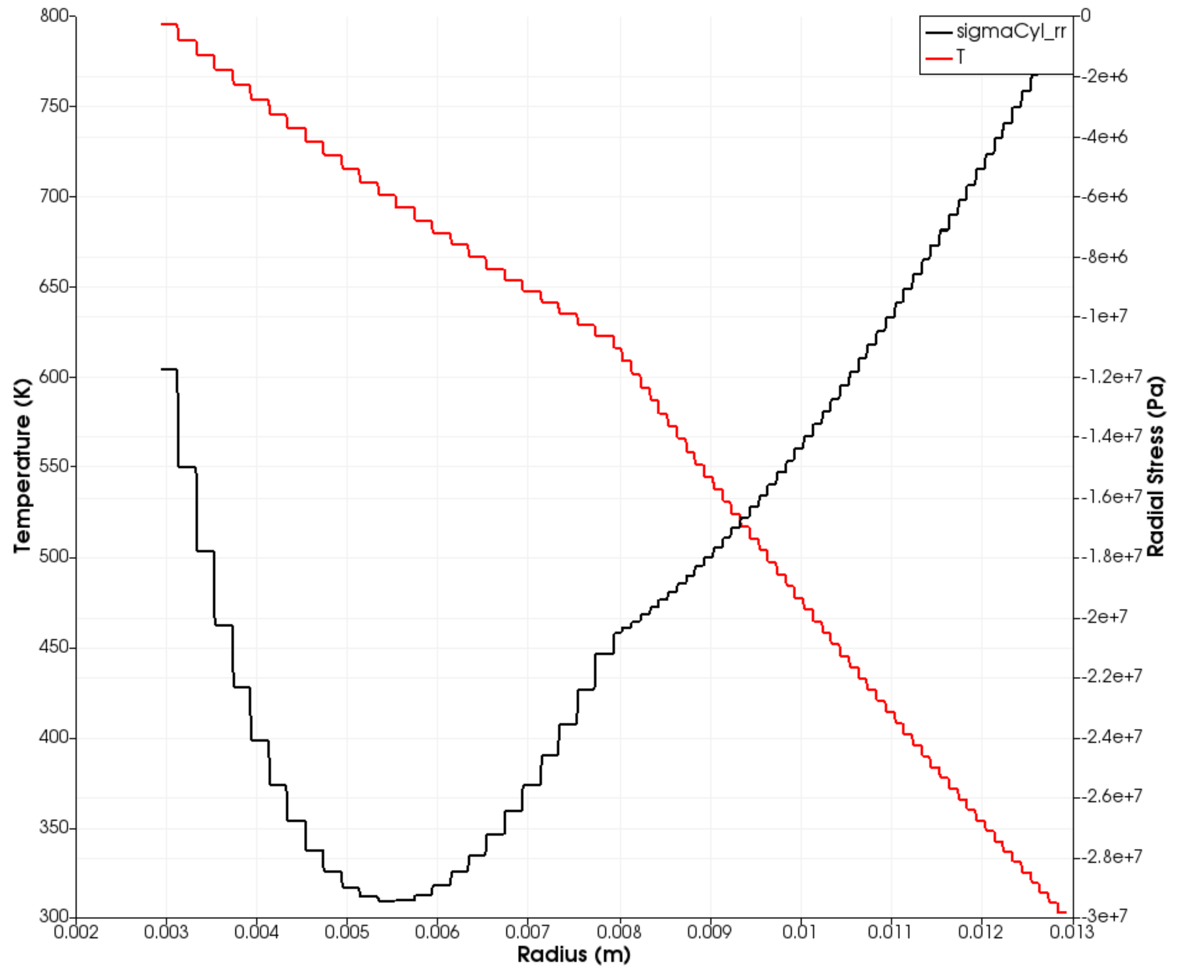
- Why? Only displacement and normal stress must be *continuous* at the interface…
  … but material properties (E, μ, ν) and gradient of displacement *not necessarily!*

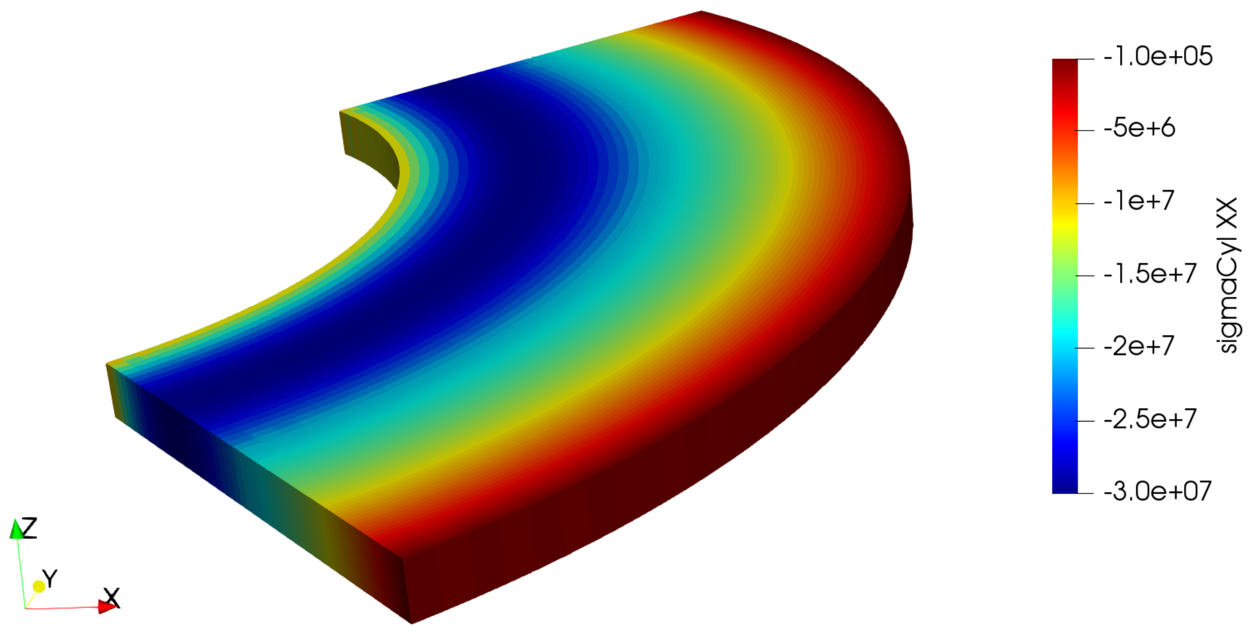
- Interest for coated fuels, clad with liners, oxide layers



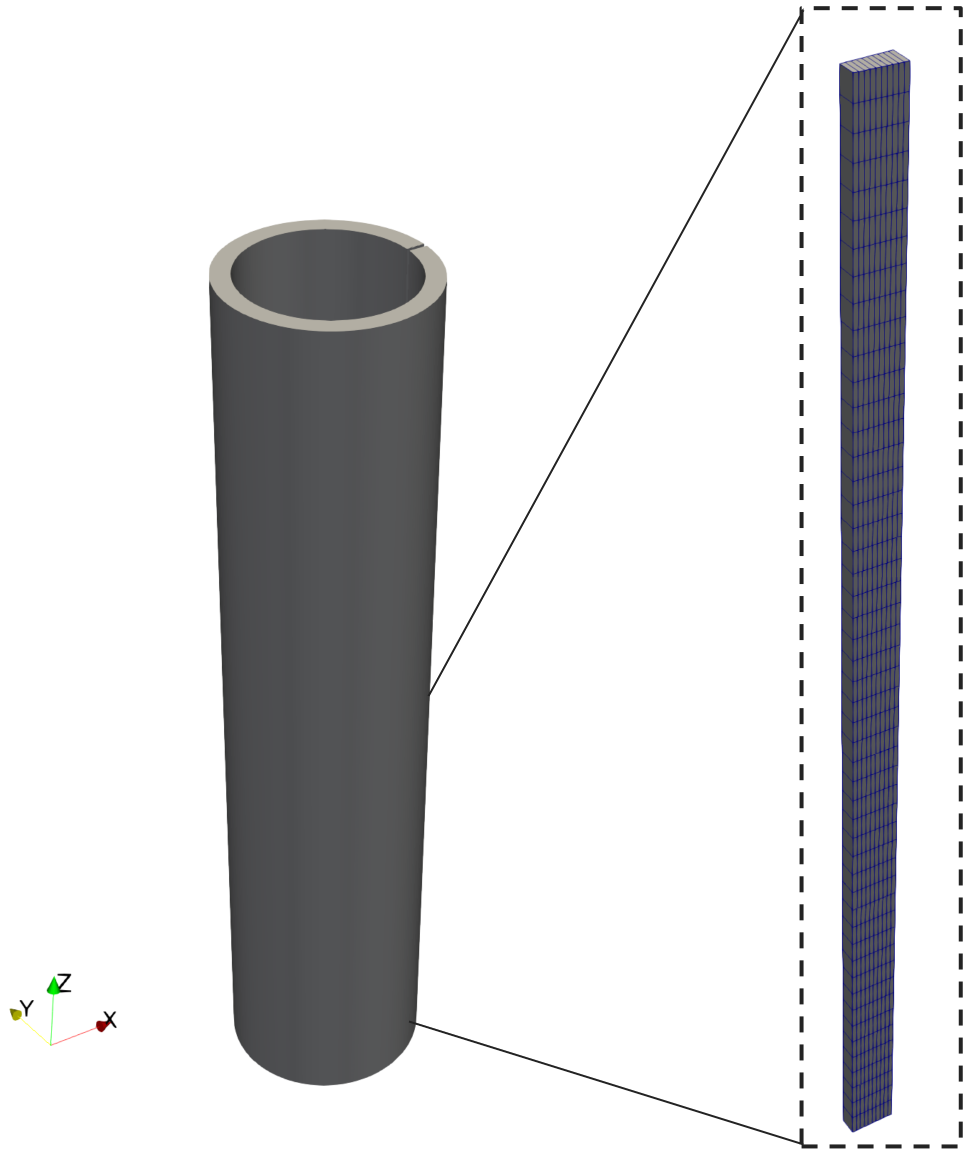
NOTE:
- **Similar situation when modeling coarse meshes with high gradients in temperature or strains.**
- **We find benefit from applying this correction even in the abscence of actual multi-material bodies.**

Hot bi-material cylinder - results





Generic PUZRY - overview



- Experiments performed to study clad burst during LOCA
- Short Zircaloy segments (5cm) internally pressurized until failure:
  - Different pressure ramp rates
  - Different temperature levels
- Half-symmetry model created with blockMesh
- Thermo-mechanics case (**`constant/solverDict`**) with incremental mesh-updated solver

```
thermalSolver         solidConduction;
mechanicsSolver       smallStrainIncrementalUpdated;
neutronicsSolver      fromLatestTime;
elementTransport      fromLatestTime;
```

Generic PUZRY – Time-dependent BCs

Time-dependent inner pressure (**`0/DD.boundaryField`**)

```
inner
{
    type            tractionDisplacement;
    pressureList
    {
        type            table;
        file            "$FOAM_CASE/constant/data/p_PUZRY";
        format          foam;
        outOfBounds     clamp;
        interpolationScheme linear;
    }
    traction        uniform ( 0 0 0 );
    relax           1;
    value           $internalField;
}
```

Time-dependent outer temperature (**`0/T.boundaryField`**)

```
outer
{
    type            timeDependentAxialProfileT;
    axialProfileDict
    {
        axialLocations  ( 0 0.005 0.01 0.015 0.02 0.025 );
        #include        "$FOAM_CASE/constant/data/Tprofile_PUZRY";
    }
    value           $internalField;
}
```

Generic PUZRY – LOCA creep model

Creep (LOCA) rheological model for cladding in (**constant/solverDict.materials**)

```
materials
{
    cladding
    {
        material                zircaloy;
        …
        rheologyModel   misesPlasticCreep;
        rheologyModelOptions
        {
            plasticStrainVsYieldStress table
            ( (0     600e6) );

            creepModel LimbackCreepModelLOCA;
            relax 0.1;

            primaryCreep     off;
            irradiationCreep off;

            NewtonRaphsonMethod on;
        }
    }
}
```

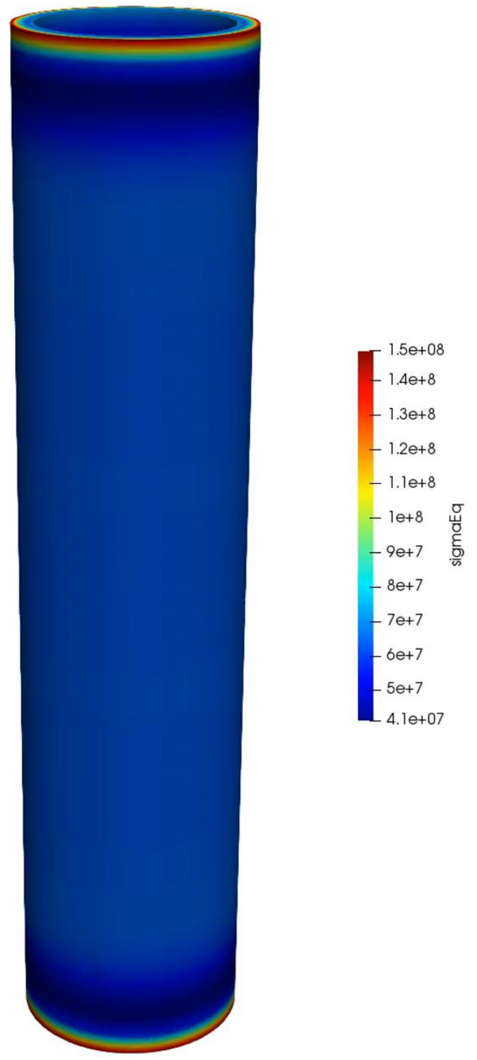
Creep increment dictate time-step size (**system/controlDict**)

```
…

adjustableTimeStep true;
maxDeltaT        10;
minDeltaT        1e-06;
maxRelativeDeltaTIncrease 1.5;
minRelativeDeltaTDecrease 1.5;
maxRelativePowerIncrease 1e+09;
maxRelativePowerDecrease 1e+09;
maxBurnupIncrease 1e+09;
maxAverageCreep  0.0001;
maxMaximumCreep  0.0001;
maxFGR           1e+09;

runTimeModifiable true;
```

NOTE:

- **Creep might need relaxation (sometimes as low as 0.1).**
- **For most simulations relax can be set to 1 (and for some other simulations not under-relaxing creep even improves stability…)**

Generic PUZRY - result



- Copy folder from **`Cases/testCases/generic_PUZRY`**
- Use **`Allrun`** script or
  - Create mesh with: **`blockMesh`**
  - Run the case with: **`offbeat`**
- Check results with: **`paraFoam`**
- Failure takes place at 1118.16 seconds …
  lots of time-step are needed due to high creep-rate!

Generic TRISO - overview

- TRISO fuel case derived from IAEA benchmark

- Double wedge geometry for spherical symmetry (1D case)

- Multiple layers: kernel, **buffer**, IPyc, SiC and OPyC
  - Material models for each layer defined in **`solverDict`**
  - Multi-material correction necessary
  - Buffer and IPyC start already detached (automatic debonding is on its way…)

Generic TRISO – time-dependent heat source

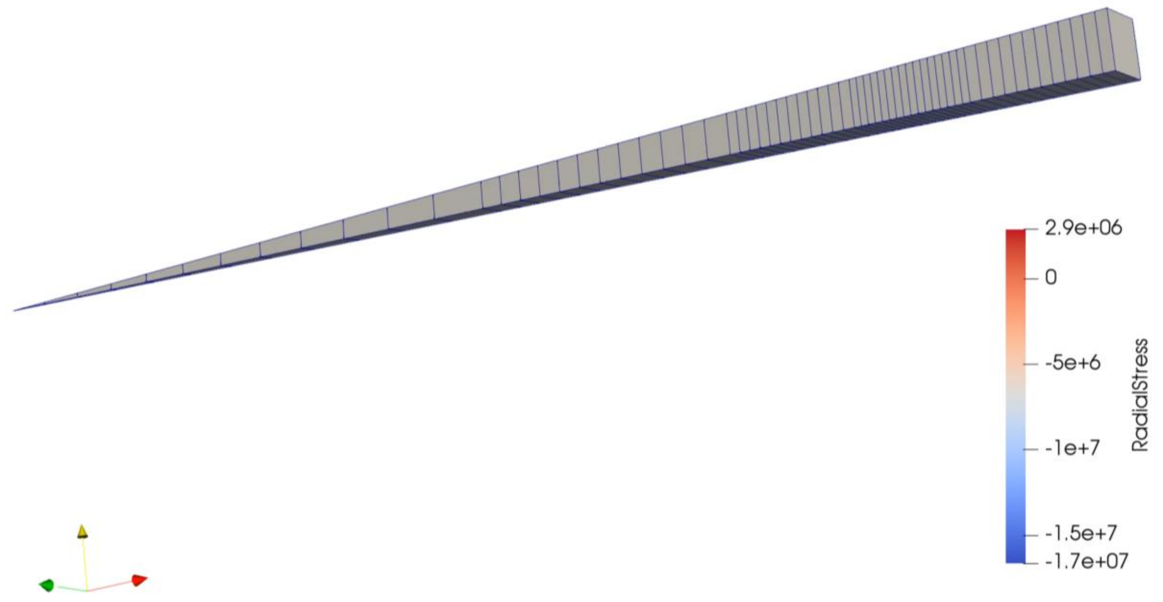
A time dependent heat source can be set in **`constant/solverDict/heatSourceOptions`**
(Similar syntax for setting a linear heat rate or lhgr for rod cases)

```
heatSource timeDependentVhgr;

…

heatSourceOptions
{
    timePoints   (0.0 3600 49996400 5e7);
    vhgr         (0.0 3e9  3e9      0.0);

    timeInterpolationMethod linear;

    materials ( Kernel );
}

…
```

- `timePoints` depends on selected time unit.
- `vhgr` given in W/m^3.
- `vhgr` and `timePoints` must have same length.

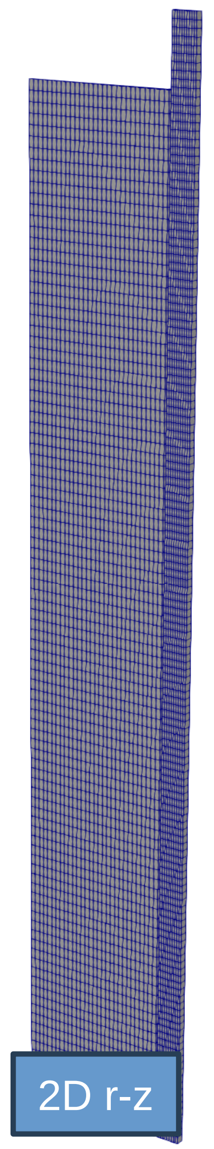
`timeInterpolationMethod` can be *linear* or *step*.

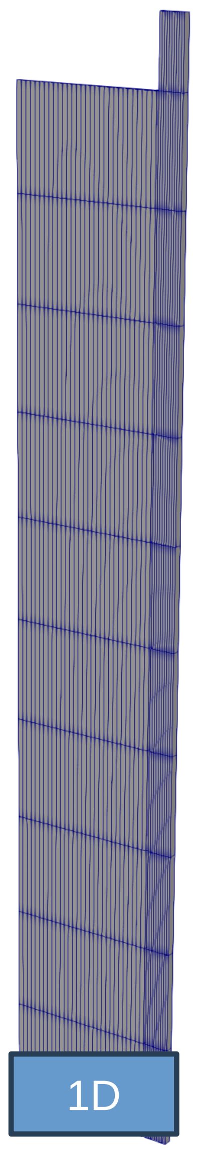
Generic TRISO - result

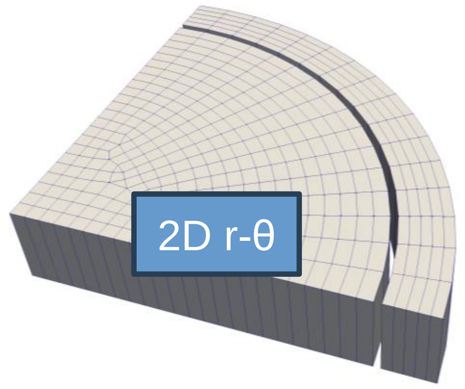


- Copy folder from **`Cases/testCases/generic_TRISO`**
- Use **`Allrun`** script or
  - Create mesh with **`blockMesh`**
  - Run the case with **`offbeat`**
- Check results with **`paraFoam`**

Generic PWR - overview





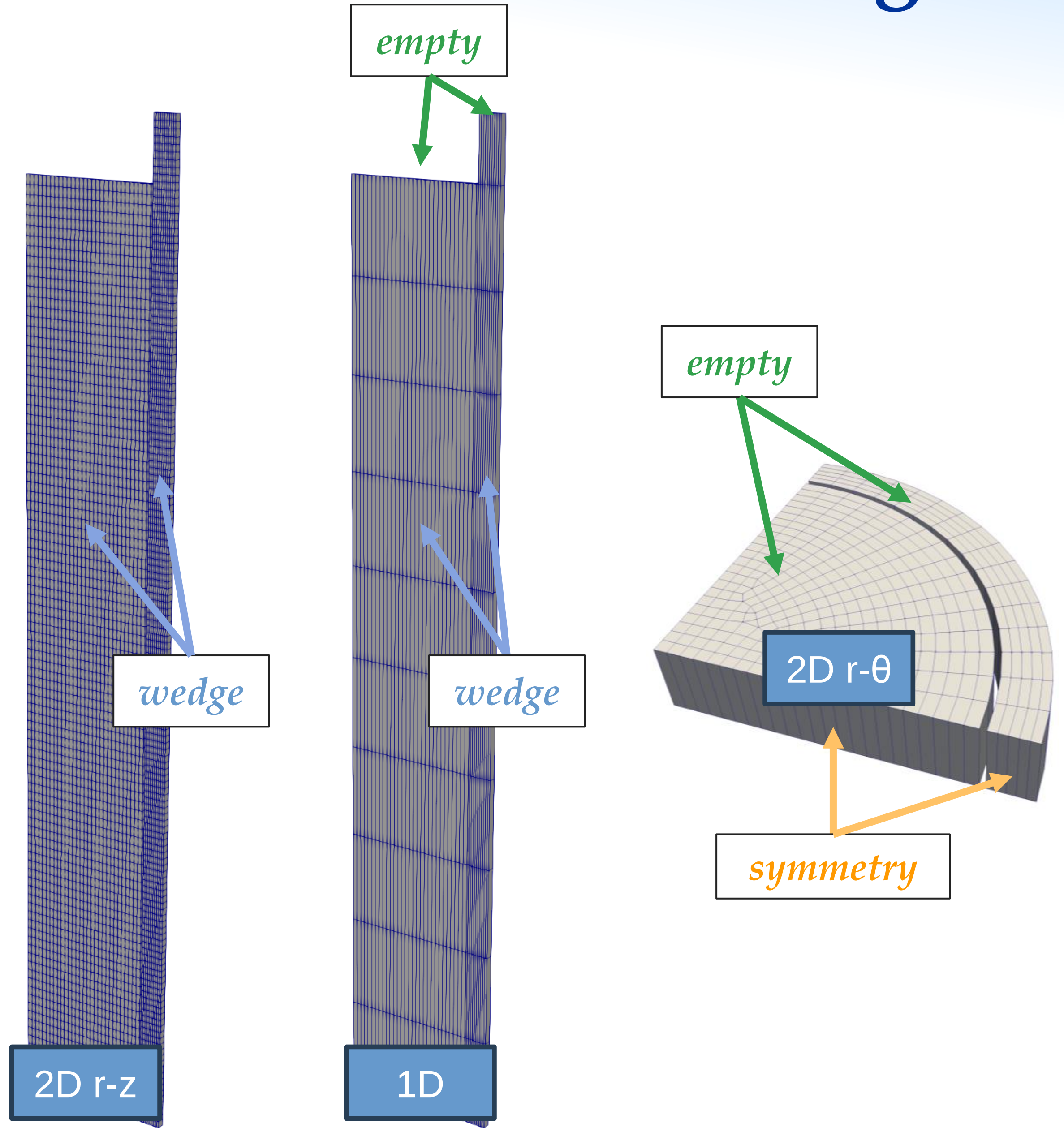


- Generic PWR rod irradiated for 1 year

- ***More complex case***! Most models are activated:
  - Gap gas model
  - Burnup
  - Axial profile linear heat source
  - Contact, gap heat transfer etc…

- 3 versions available:
  - 1D coarse mesh
  - 2D r-z fine mesh (smeared column)
  - 2D r-θ (disc)

Generic PWR - setting the geometry



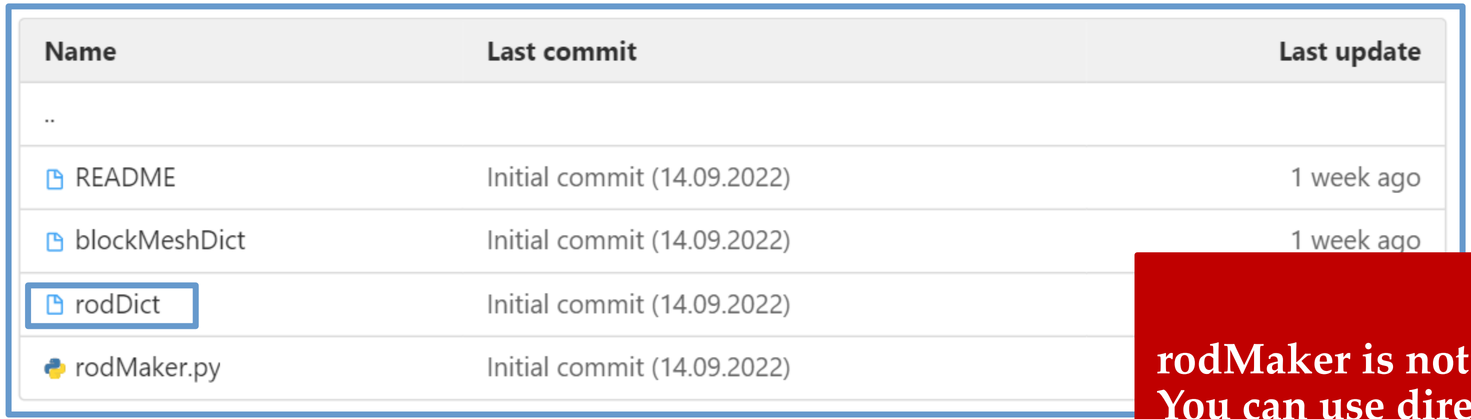
- *wedge* type patches makes the case axisymmetric 2D: OFFBEAT neglects solution along y (i.e. azimuthal direction).

- The addition of *empty* type patches on top and bottom makes the case 1D (we need to activate the switch ***modifiedPlaneStrain***)

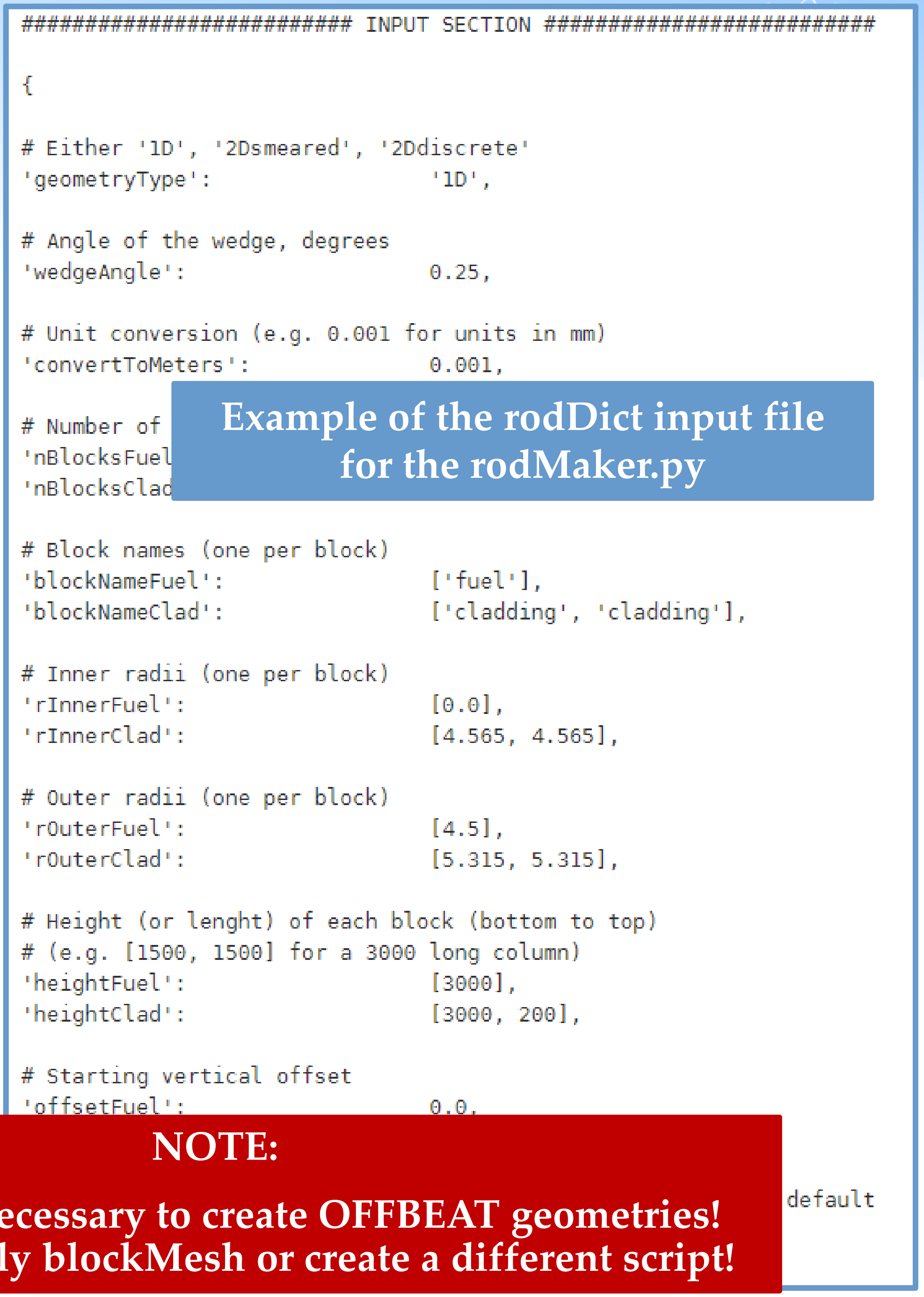
- Alternatively, 1-cell-thick disc with *empty* patches makes the case 2D in the r-theta plane

Tools for mesh generation

- *blockMesh* good tool for axisymmetric geometries (1D and 2D r-z).
- *rodMaker.py* available in the *tools/* folder.
- A similar script can be found in the TRISO tutorial and verification cases for TRISO 1D geometries



Name	Last commit	Last update
..		
README	Initial commit (14.09.2022)	1 week ago
blockMeshDict	Initial commit (14.09.2022)	1 week ago
rodDict	Initial commit (14.09.2022)	
rodMaker.py	Initial commit (14.09.2022)	



```
######################### INPUT SECTION #########################

{

# Either '1D', '2Dsmeared', '2Ddiscrete'
'geometryType':                 '1D',

# Angle of the wedge, degrees
'wedgeAngle':                   0.25,

# Unit conversion (e.g. 0.001 for units in mm)
'convertToMeters':              0.001,

# Number of
'nBlocksFuel
'nBlocksClad

# Block names (one per block)
'blockNameFuel':                ['fuel'],
'blockNameClad':                ['cladding', 'cladding'],

# Inner radii (one per block)
'rInnerFuel':                   [0.0],
'rInnerClad':                   [4.565, 4.565],

# Outer radii (one per block)
'rOuterFuel':                   [4.5],
'rOuterClad':                   [5.315, 5.315],

# Height (or lenght) of each block (bottom to top)
# (e.g. [1500, 1500] for a 3000 long column)
'heightFuel':                   [3000],
'heightClad':                   [3000, 200],

# Starting vertical offset
'offsetFuel':                   0.0,

                                                         default
```

Example of the rodDict input file for the rodMaker.py

NOTE:

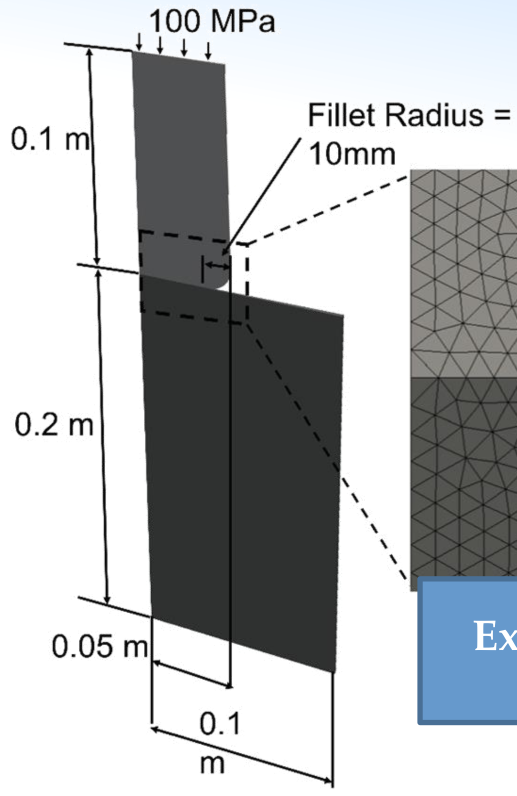
rodMaker is not necessary to create OFFBEAT geometries! You can use directly blockMesh or create a different script!

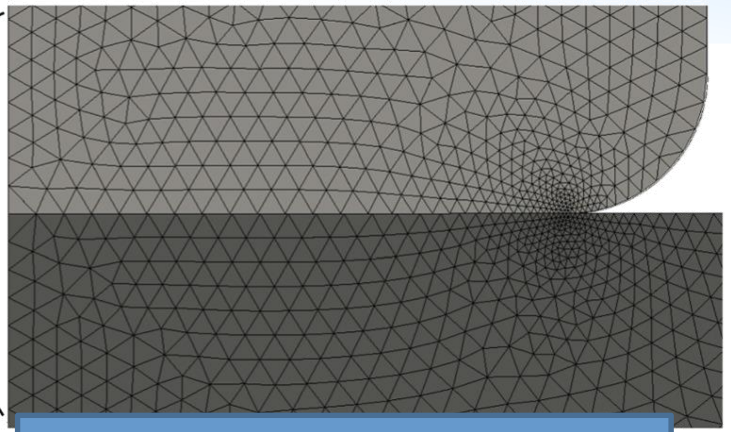
Tools for mesh generation

- ***blockMesh*** good tool for axisymmetric geometries (1D and 2D r-z).
- ***rodMaker.py*** available in the *tools/* folder.
- A similar script can be found in the TRISO tutorial and verification cases for TRISO 1D geometries

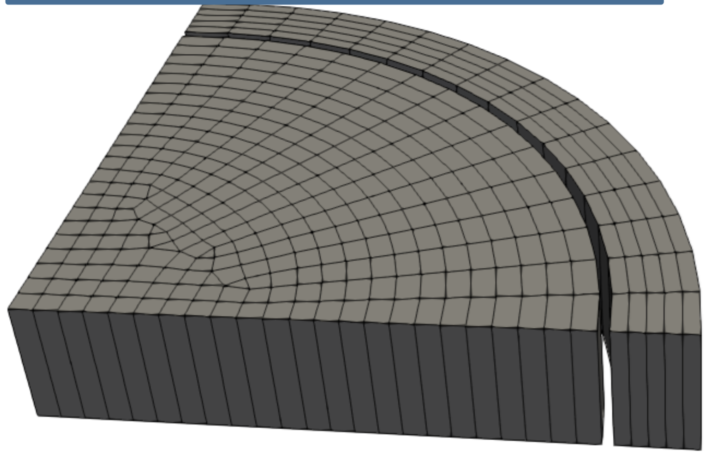
Limited to structured meshes!

Difficult to extend to complex 3D cases!

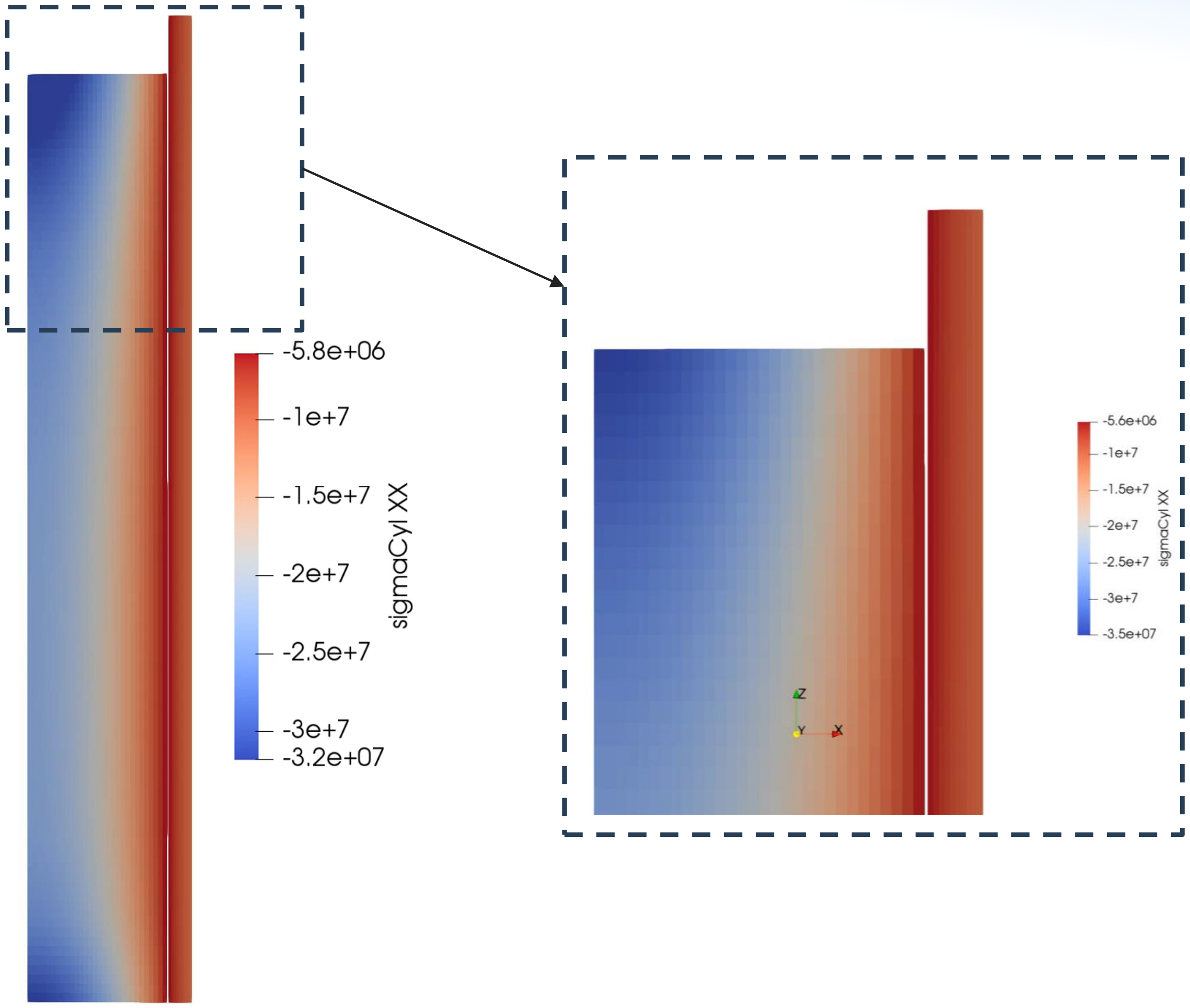




Example of complex meshes obtained with Salome



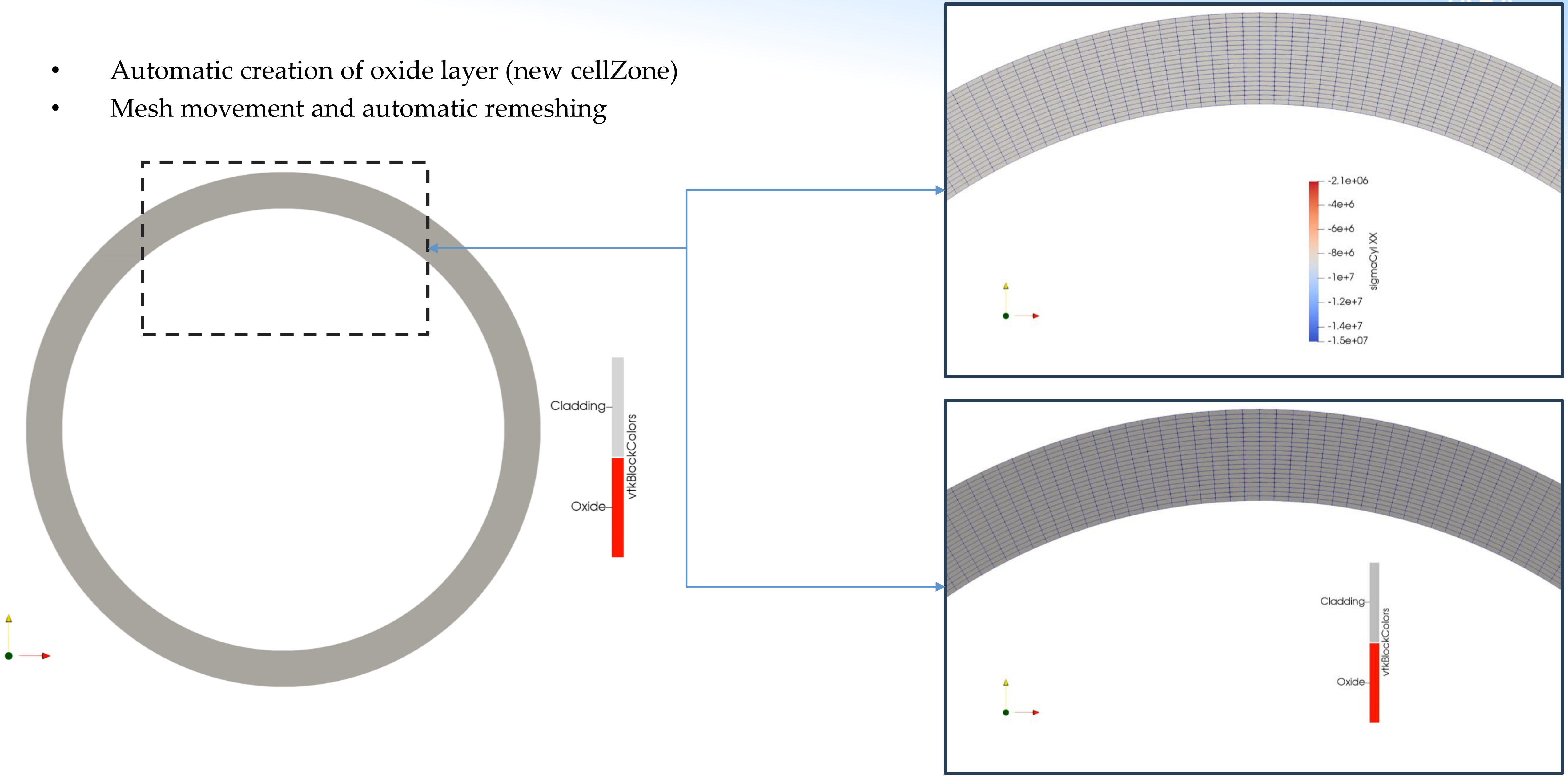
Generic PWR - results



- Copy folder from **`Cases/testCases/generic_pwr_2D_rz`**
- Use **`Allrun`** script or
  - Create mesh with **`blockMesh`**
  - Run the case with **`offbeat`**
- Check results with **`paraFoam`**

Cladding corrosion – coming soon!

- Automatic creation of oxide layer (new cellZone)
- Mesh movement and automatic remeshing



IAEA

Tips & tricks

Time stepping

A dynamic time-step size is often fundamental to:

- Manage the computing time (e.g. avoid too long simulations).
- Improve convergence (e.g. when creep is activated and time steps need to adapt to creep rate).

Dynamic time step size can be activated in the *controlDict* file in the *system/* folder:

```
…
//- Adjustable time step options:
adjustableTimeStep true;

maxDeltaT                  6.048e5;
minDeltaT                  0.001;

maxRelativeDeltaTIncrease 1e9;
minRelativeDeltaTDecrease 1e9;

maxRelativePowerIncrease   1e9;
maxRelativePowerDecrease   1e9;

maxBurnupIncrease          0.1;

maxAverageCreep            1e-4;
maxMaximumCreep            1e-4;

maxFGR                      1e-8;
```

Max and min time step size allowed during simulation.

Limit relative increase/decrease of power and or time-step size during simulation.

Maximum burnup increase in MWd/kg.

Maximum creep increment (suggested 1e-4 for most cases).

Time unit

- OpenFOAM time unit is seconds.
- Not always convenient for fuel performance simulations.
- In OFFBEAT the time step unit can be changed in the *controlDict* file.

```
…
userTime
{
    // type is seconds by default
    type seconds;//hours;//days;
}
…
```

NOTE:
When changing time unit all time-dependent input must be changed accordingly!

```
coolantPressureList
{
    type          table;

    values
    (
        (0.0    1e5)
        (3600.0 1e6)
    );
}
```

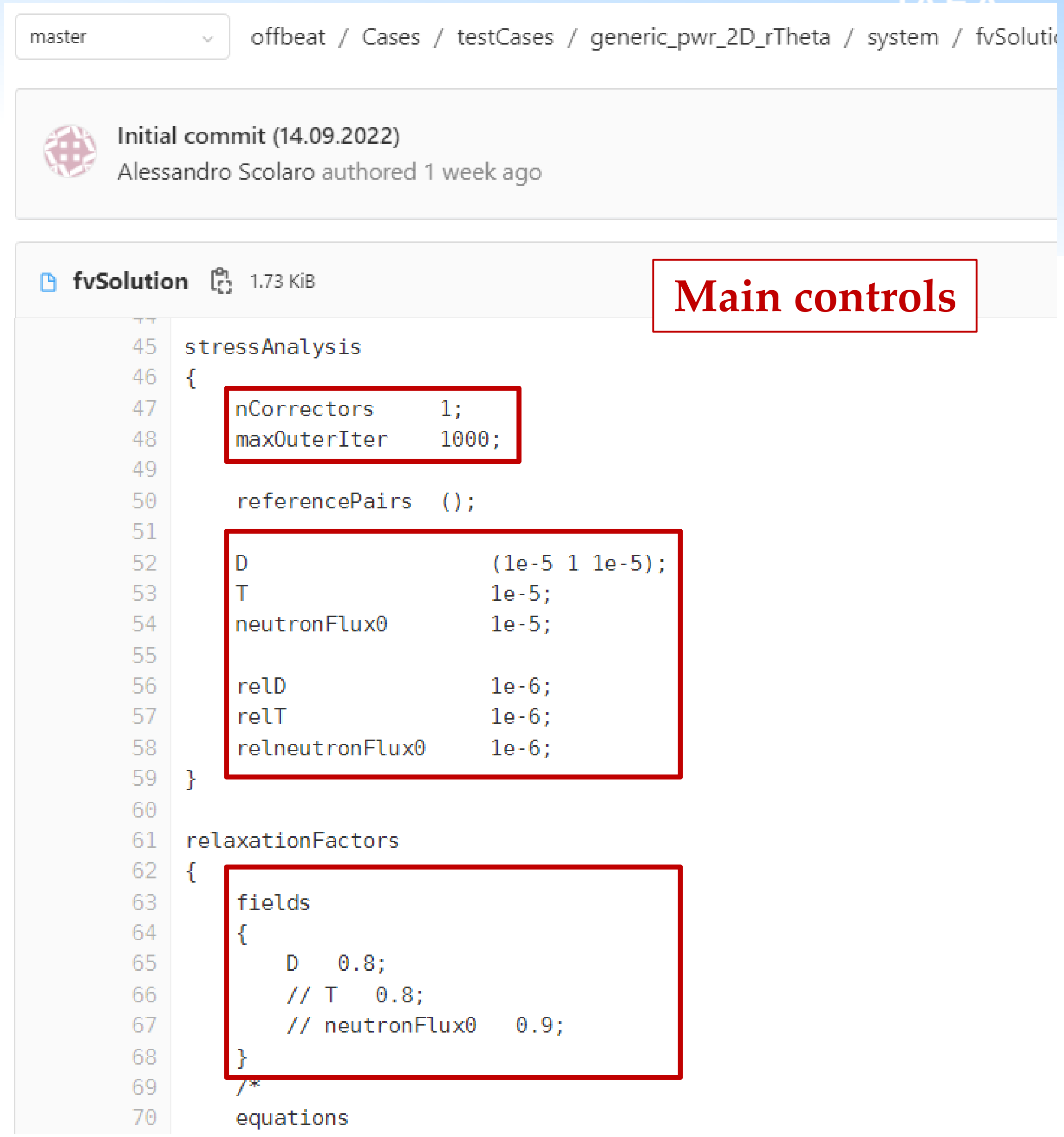
From seconds to hours

```
coolantPressureList
{
    type          table;

    values
    (
        (0.0    1e5)
        (1.0    1e6)
    );
}
```

Simulation control

Main controls are in *fvSolution* file inside *system/* folder

- `nCorrectors:` number of inner iterations for main physics (neutronics, thermal, mechanics).
- `maxOuterIter:` maximum number of outer iterations before end of time step (if not converged yet).
- `D, T, neutronFlux0:` are the residual thresholds. Values between 1e-6 and 1e-5 work best.
- `relaxationFactors:` relax main fields such as D and T. Values between 0.8 and 1 work best.
- It is also possible to relax the `equations` and not the `fields`, but typically it has worse convergence properties.



master | offbeat / Cases / testCases / generic_pwr_2D_rTheta / system / fvSoluti

Initial commit (14.09.2022)
Alessandro Scolaro authored 1 week ago

fvSolution 1.73 KiB

Main controls

```
stressAnalysis
{
    nCorrectors      1;
    maxOuterIter     1000;

    referencePairs  ();

    D                     (1e-5 1 1e-5);
    T                     1e-5;
    neutronFlux0          1e-5;

    relD                  1e-6;
    relT                  1e-6;
    relneutronFlux0       1e-6;
}

relaxationFactors
{
    fields
    {
        D   0.8;
        // T   0.8;
        // neutronFlux0   0.9;
    }
    /*
    equations
```

Simulation control

Other three important relaxation parameters:

- Relax the heat transfer coefficient in fuelRodGap in the thermal BCs.
- Relax interface pressure in gapContact in the mechanics BCs.
- Relax creep in the rheologcal model dictionary for a specific material.

In *$CASE/0/D:*

```
"fuelOuter|cladInner"
{
    type                gapContact;
    patchType           regionCoupledOFFBEAT;
    penaltyFactor       0.1;
    frictionCoefficient         0;
    relax               1.0;
    relaxInterfacePressure          0.1;
    traction            uniform (0 0 0);
    pressure            uniform 15.5e5;
    value               $internalField;
}
```

In *$CASE/constant/solverDict:*

```
cladding
{
    material                    zircaloy;
    Tref                        Tref [ 0 0 0 1 0 ] 293;

    PoissonRatioModel ZyConstant;

    rheologyModel    misesPlasticCreep;
    rheologyModelOptions
    {
        plasticStrainVsYieldStress table
        (
            (0    250e6)
        );

        creepModel LimbackCreepModel;
        relax 1.0;
    }
}
```

In *$CASE/0/T:*

```
fuelOuter
{
    type                fuelRodGap;
    patchType           regionCoupledOFFBEAT;
    kappa               k;
    coupled             true;
    alpha               uniform 5000;
    roughness           uniform 2.2e-6;
    value               $internalField;
    relax               1;
}
```

Troubleshooting

Sometimes the simulation might fail or might have problems converging.

1. Always checks the residuals: the simulation might go on but the residual might not properly converge.

2. Check the mesh: is it valid? Are the patch name assigned correctly?

3. Check the BC conditions: is the body properly fixed? Are the BC for the top cladding and fuel surfaces correct?

4. Check the solverDict: start from the main solvers and then focus on the finer options.

5. Check the relaxation factors:
   - fuelRodGap relaxation and gapContact tend to be activated most of the time (down to 0.1).
   - Fields relaxation often helps too (0.9 is a typical value).
   - Creep as a last resource might help the simulation to converge.

6. Check the time step size: is the initial time step too large for the creep to converge? Does the creep time-stepping criterion need to be tighter?

Conclusions

- OFFBEAT is open-source!
  - Download it from: https://gitlab.com/foam-for-nuclear/offbeat
  - It consists of a thermo-mechanics library for application in nuclear engineering.
  - Documentation has recently improved significantly and will continue to grow!

- Extensive verification and ongoing validation for several different conditions
- Large scope of possible applications (even beyond the fuel rod)!

- Interesting features are coming soon, such as:
  - Mesh movement/modification capabilities for corrosion, porosity, layer detachment.
  - More tutorials e.g. for RIA, LOCA.
  - Examples of coupling with TH (or integration with GeN-Foam in a unified library).

- Contributions, comments, suggestions, bug reports etc. are more than welcomed!
- Get in touch with us if you want to participate to the project

References

- A. Scolaro, Development of a Novel Finite Volume Methodology for Multi-Dimensional Fuel Performance Applications, 2021.
- A. Scolaro, I. Clifford, C. Fiorina, A. Pautz, The OFFBEAT multi-dimensional fuel behavior solver. Nuclear Engineering and Design Volume 358, March 2020.
- I. Clifford, M. Pecchia, R. Mukin, C. Cozzo, H. Ferroukhi, A. Gorzel, Studies on the effects of local power peaking on heat transfer under dryout conditions in BWRs, Ann. Nucl. Energy, 2019.
- A. Scolaro, Y. Robert, C. Fiorina, I. Clifford, A. Pautz, Coupling methodology for the multi-dimensional fuel performance code OFFBEAT and the Monte Carlo neutron transport code serpent. In: Proceedings of Global and Top Fuel 2019 Conference, September 22–26, 2019, Seattle, Washington USA.
- A. Scolaro, C. Fiorina, I. Clifford, A. Pautz, Cladding plasticity modeling with the multi-dimensional fuel performance code OFFBEAT. In: Proceedings of Global and Top Fuel 2019 Conference, September 22–26, 2019, Seattle, Washington USA.
- A. Scolaro, I. Clifford, C. Fiorina, A. Pautz,Multi-dimensional creep analysis using the novel OFFBEAT fuel performance code. IAEA Technical Meeting on PCI/SCC, 8–11 Oct. 2019, Aix-en-Provence, France.
- A. Scolaro, P. Van Uffelen, C. Fiorina, A. Schubert, I. Clifford, A. Pautz, Investigation on the effect of eccentricity for fuel disc irradiation tests, Nucl. Eng. Technol., 2020.
- A. Scolaro, C. Fiorina, I. Clifford, A. Pautz, Development of a semi-implicit contact methodology for finite volume stress solvers, Int. J. Numer. Methods Eng., 2021.
- A. Scolaro, P. Van Uffelen, A. Schubert ,C. Fiorina, E. Brunetto, I. Clifford, A. Pautz, Towards coupling conventional with high-fidelity fuel behavior analysis tools, Progress in Nuclear Energy, Volume 152, 2022.

IAEA
International Atomic Energy Agency

Joint ICTP-IAEA Workshop on Open-Source Nuclear Codes for Reactor Analysis

07 - 11 Aug 2023

Contact: ONCORE@iaea.org

Thank you!

Contact: ONCORE@iaea.org

ONCORE: Open-source Nuclear Codes for Reactor Analysis