IAEA
International Atomic Energy Agency

Joint ICTP-IAEA Workshop on Open-Source Nuclear Codes for Reactor Analysis
August 7-11 2023

Introduction to GeN-Foam – Using the code

Carlo Fiorina

About this lecture

What to expect

- An introduction to using GeN-Foam

What not to expect

- A full course on the use of GeN-Foam
- A hands-on exercise on the use of GeN-Foam

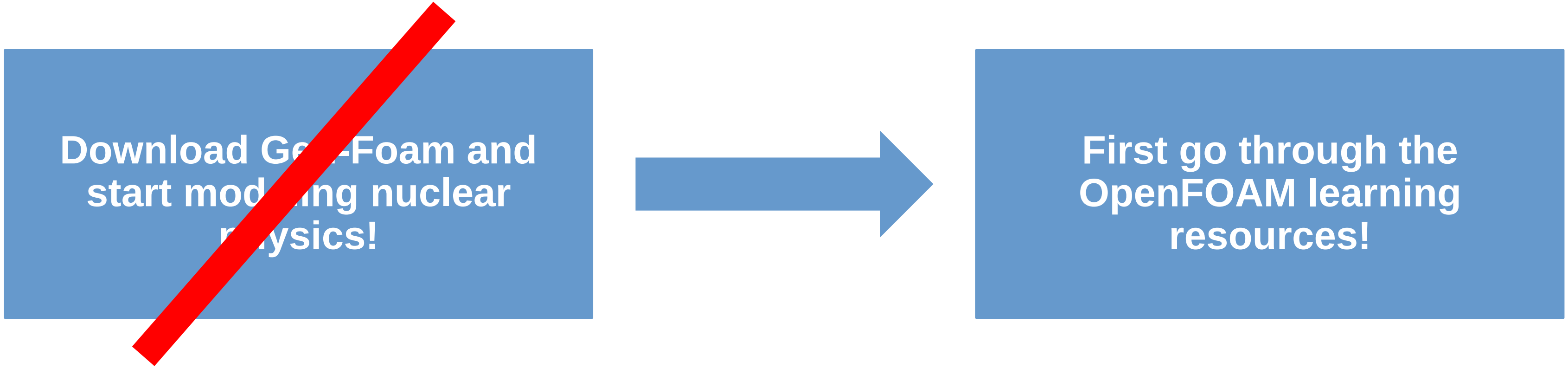
Objectives

- How to approach GeN-Foam
- References, keywords, best practices that can simplify an autonomous learning of GeN-Foam

About this lecture

- Recap of learning resources for OpenFOAM
- Additional background
  - Multi-mesh
  - Multi-material
- How to get and install GeN-Foam
- What's inside
  - Tools
  - Documentation
    - How to use it
  - Source code
    - How to use it
  - Tutorials
    - How to use them
    - List
- Summary of suggested approach and resources

How to approach GeN-Foam: prerequisites

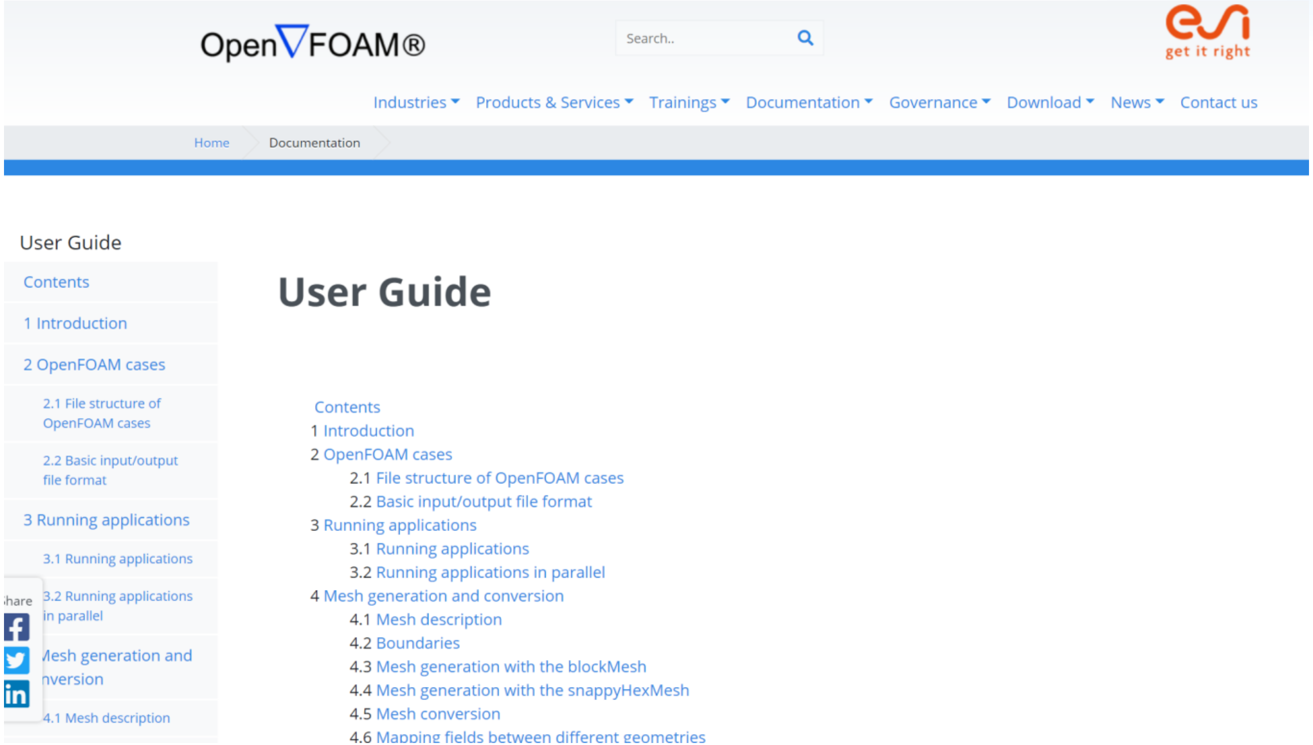


~~Download GeN-Foam and start modelling nuclear physics!~~

First go through the OpenFOAM learning resources!

Learn OpenFOAM - Official documentation

- https://www.openfoam.com/documentation/user-guide

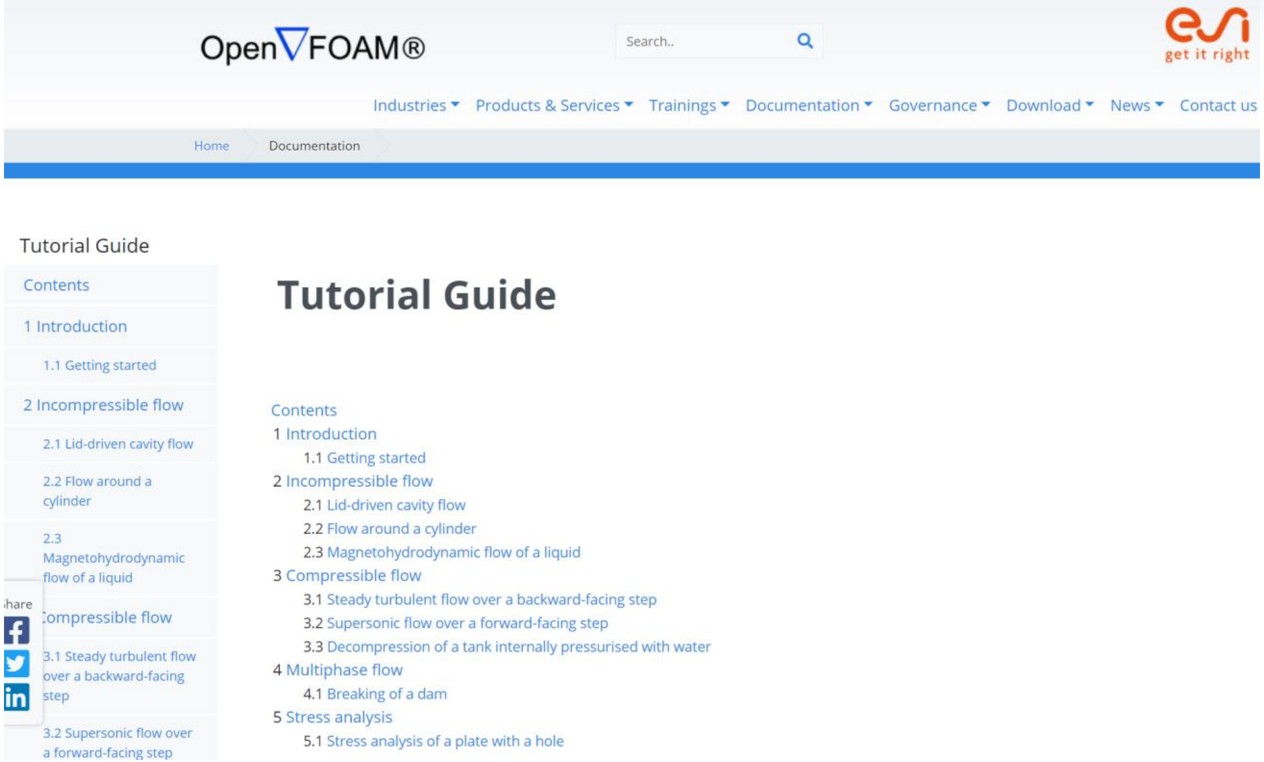


User Guide

User Guide

Learn OpenFOAM - Official documentation

- https://www.openfoam.com/documentation/tutorial-guide

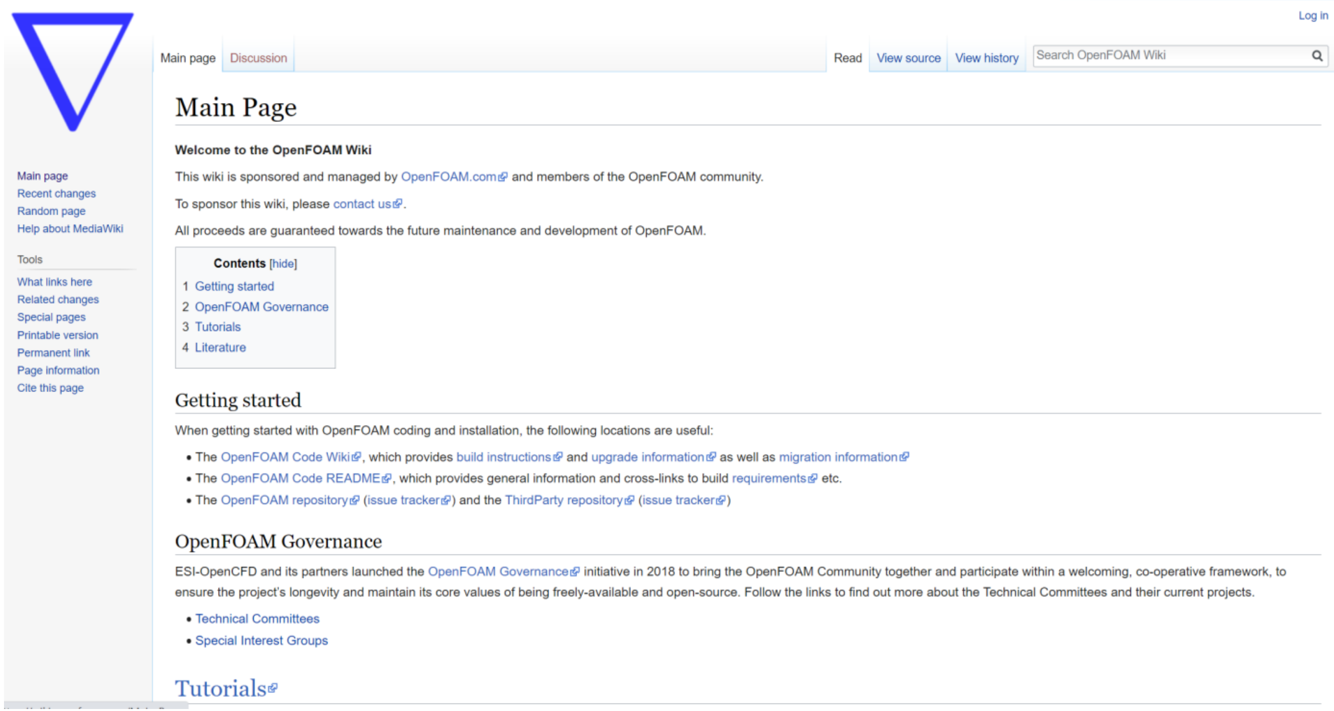


Tutorial Guide

Tutorial Guide

Learn OpenFOAM - Official documentation

- https://wiki.openfoam.com/Main_Page



Main Page

Welcome to the OpenFOAM Wiki

This wiki is sponsored and managed by OpenFOAM.com and members of the OpenFOAM community.

To sponsor this wiki, please contact us.

All proceeds are guaranteed towards the future maintenance and development of OpenFOAM.

Contents [hide]

Getting started

When getting started with OpenFOAM coding and installation, the following locations are useful:

- The OpenFOAM Code Wiki, which provides build instructions and upgrade information as well as migration information
- The OpenFOAM Code README, which provides general information and cross-links to build requirements etc.
- The OpenFOAM repository (issue tracker) and the ThirdParty repository (issue tracker)

OpenFOAM Governance

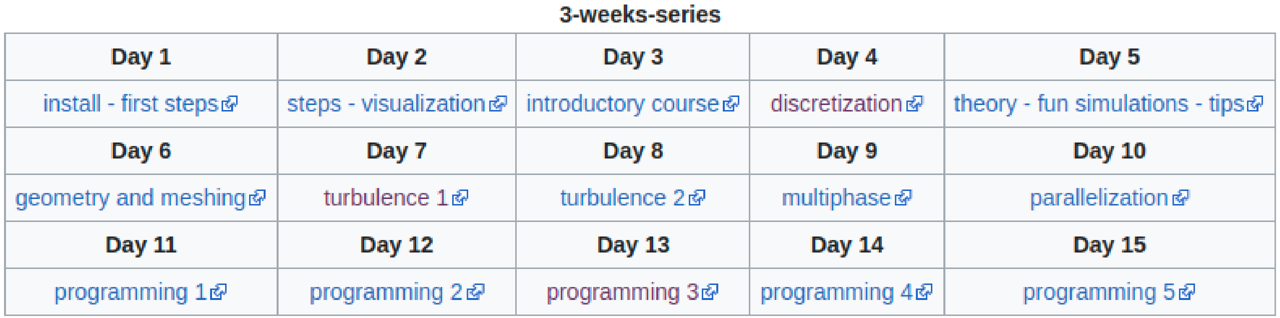
ESI-OpenCFD and its partners launched the OpenFOAM Governance initiative in 2018 to bring the OpenFOAM Community together and participate within a welcoming, co-operative framework, to ensure the project's longevity and maintain its core values of being freely-available and open-source. Follow the links to find out more about the Technical Committees and their current projects.

- Technical Committees
- Special Interest Groups

Tutorials

Learn OpenFOAM - Official documentation

https://wiki.openfoam.com/index.php?title=%223_weeks%22_series

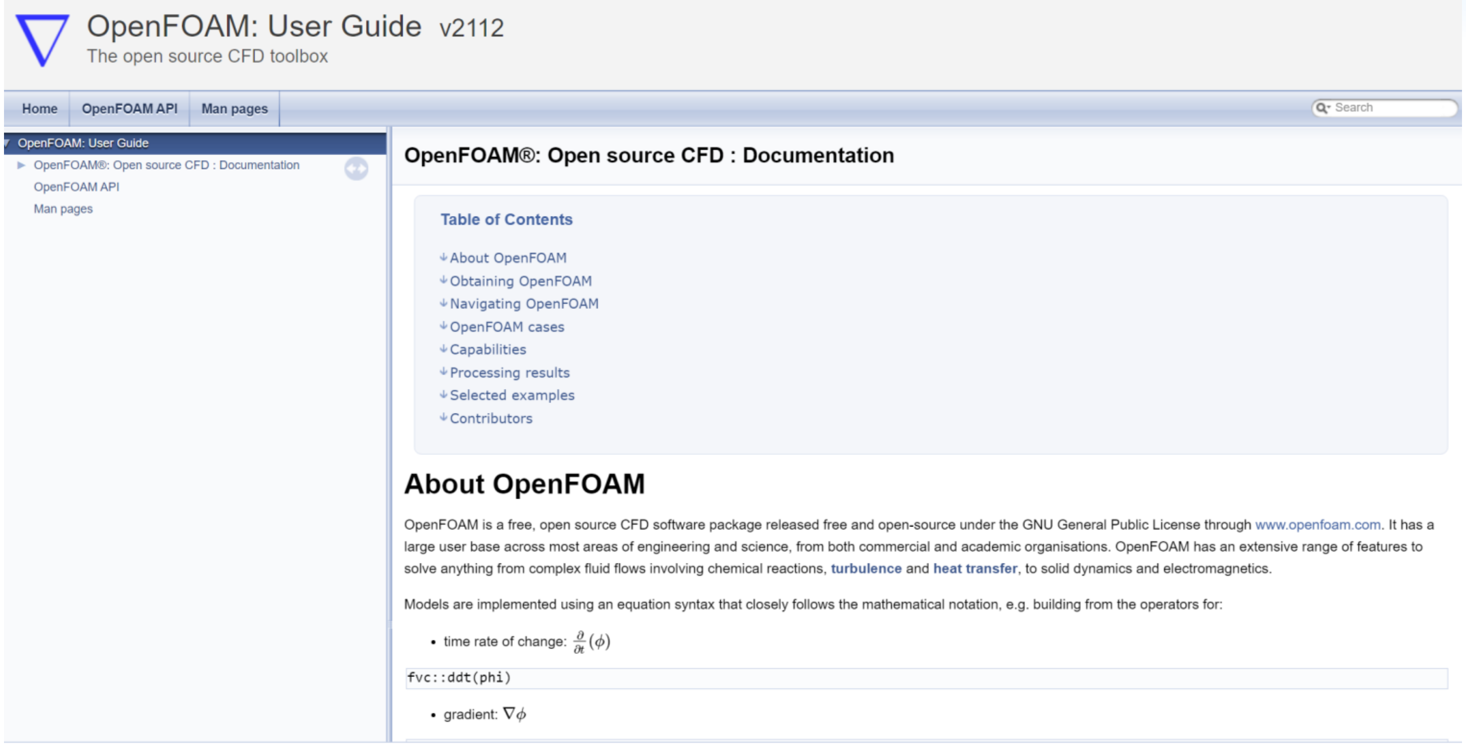


3-weeks-series

Day 1	Day 2	Day 3	Day 4	Day 5
install - first steps	steps - visualization	introductory course	discretization	theory - fun simulations - tips
Day 6	**Day 7**	**Day 8**	**Day 9**	**Day 10**
geometry and meshing	turbulence 1	turbulence 2	multiphase	parallelization
Day 11	**Day 12**	**Day 13**	**Day 14**	**Day 15**
programming 1	programming 2	programming 3	programming 4	programming 5

Learn OpenFOAM - Official documentation

- https://www.openfoam.com/documentation/guides/v2112/doc/



OpenFOAM: User Guide v2112
The open source CFD toolbox
Home
OpenFOAM API
Man pages
Search
OpenFOAM: User Guide
OpenFOAM®: Open source CFD : Documentation
OpenFOAM API
Man pages

OpenFOAM®: Open source CFD : Documentation

Table of Contents

- About OpenFOAM
- Obtaining OpenFOAM
- Navigating OpenFOAM
- OpenFOAM cases
- Capabilities
- Processing results
- Selected examples
- Contributors

About OpenFOAM

OpenFOAM is a free, open source CFD software package released free and open-source under the GNU General Public License through www.openfoam.com. It has a large user base across most areas of engineering and science, from both commercial and academic organisations. OpenFOAM has an extensive range of features to solve anything from complex fluid flows involving chemical reactions, **turbulence** and **heat transfer**, to solid dynamics and electromagnetics.

Models are implemented using an equation syntax that closely follows the mathematical notation, e.g. building from the operators for:

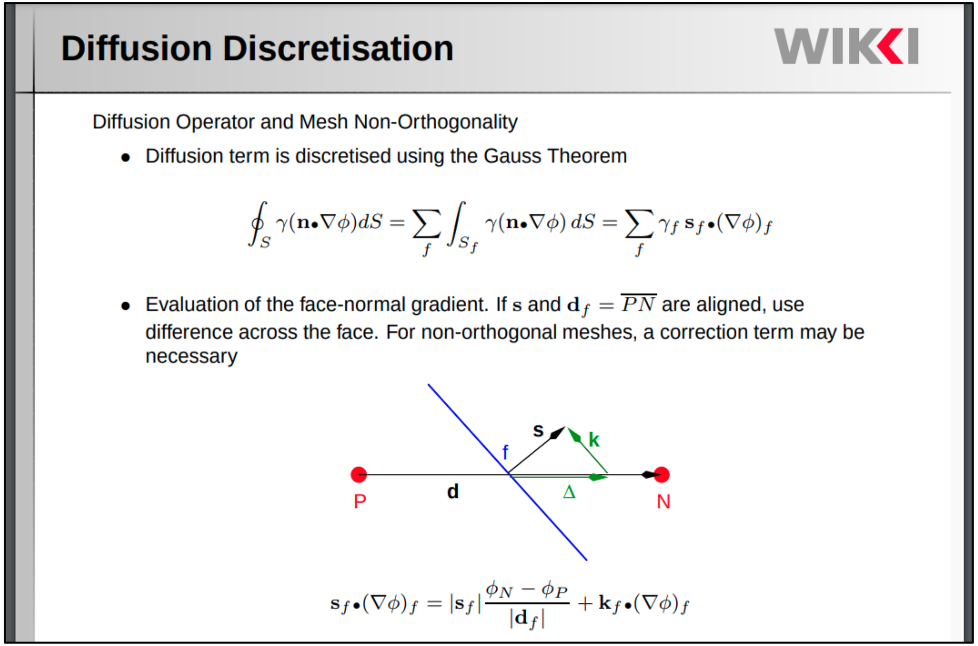
- time rate of change: $\frac{\partial}{\partial t}(\phi)$

```
fvc::ddt(phi)
```

- gradient: $\nabla\phi$

Learn OpenFOAM - Overview of Finite Volume Method from H. Jasack

https://www.youtube.com/watch?v=a4B_oXR5Kzs&ab_channel=KennethHoste



Diffusion Discretisation

Diffusion Operator and Mesh Non-Orthogonality

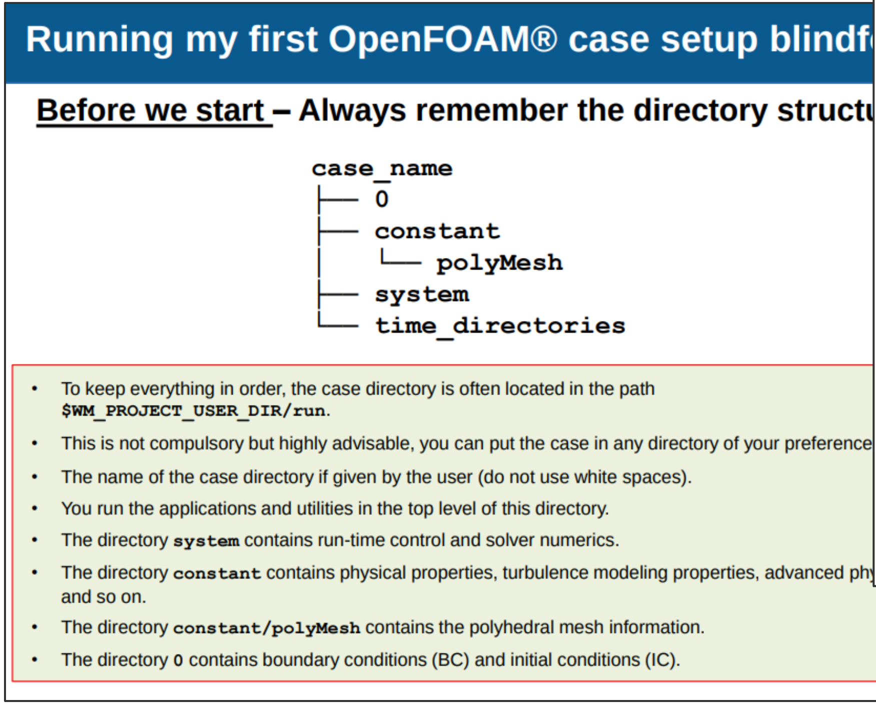
- Diffusion term is discretised using the Gauss Theorem

$$\oint_S \gamma(\mathbf{n}\bullet\nabla\phi)dS = \sum_f \int_{S_f} \gamma(\mathbf{n}\bullet\nabla\phi)\,dS = \sum_f \gamma_f\,\mathbf{s}_f\bullet(\nabla\phi)_f$$

- Evaluation of the face-normal gradient. If $\mathbf{s}$ and $\mathbf{d}_f = \overline{PN}$ are aligned, use difference across the face. For non-orthogonal meshes, a correction term may be necessary

$$\mathbf{s}_f\bullet(\nabla\phi)_f = |\mathbf{s}_f|\frac{\phi_N - \phi_P}{|\mathbf{d}_f|} + \mathbf{k}_f\bullet(\nabla\phi)_f$$

Learn OpenFOAM - Presentations from Wolf Dynamics

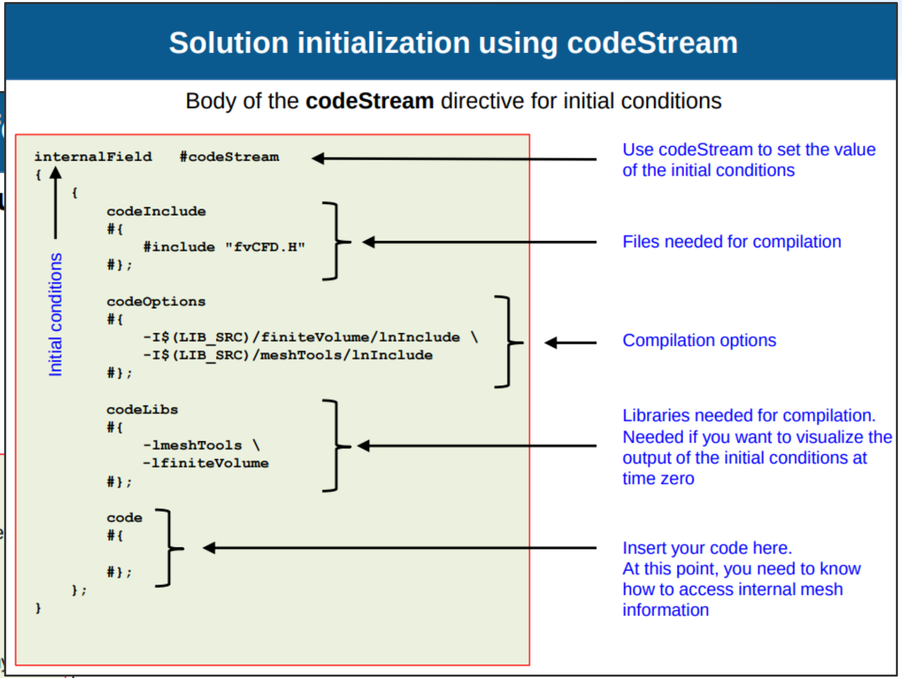


Running my first OpenFOAM® case setup blindf

Before we start – Always remember the directory structu

```
case_name
├── 0
├── constant
│   └── polyMesh
├── system
└── time_directories
```

- To keep everything in order, the case directory is often located in the path **`$WM_PROJECT_USER_DIR/run`**.
- This is not compulsory but highly advisable, you can put the case in any directory of your preference
- The name of the case directory if given by the user (do not use white spaces).
- You run the applications and utilities in the top level of this directory.
- The directory **`system`** contains run-time control and solver numerics.
- The directory **`constant`** contains physical properties, turbulence modeling properties, advanced phy and so on.
- The directory **`constant/polyMesh`** contains the polyhedral mesh information.
- The directory **`0`** contains boundary conditions (BC) and initial conditions (IC).



Solution initialization using codeStream

Body of the **codeStream** directive for initial conditions

```
internalField   #codeStream
{
    {
        codeInclude
        #{
            #include "fvCFD.H"
        #};

        codeOptions
        #{
            -I$(LIB_SRC)/finiteVolume/lnInclude \
            -I$(LIB_SRC)/meshTools/lnInclude
        #};

        codeLibs
        #{
            -lmeshTools \
            -lfiniteVolume
        #};

        code
        #{

        #};
    };
}
```

Use codeStream to set the value of the initial conditions

Files needed for compilation

Compilation options

Libraries needed for compilation. Needed if you want to visualize the output of the initial conditions at time zero

Insert your code here. At this point, you need to know how to access internal mesh information

Learn OpenFOAM - Plenty of additional resources

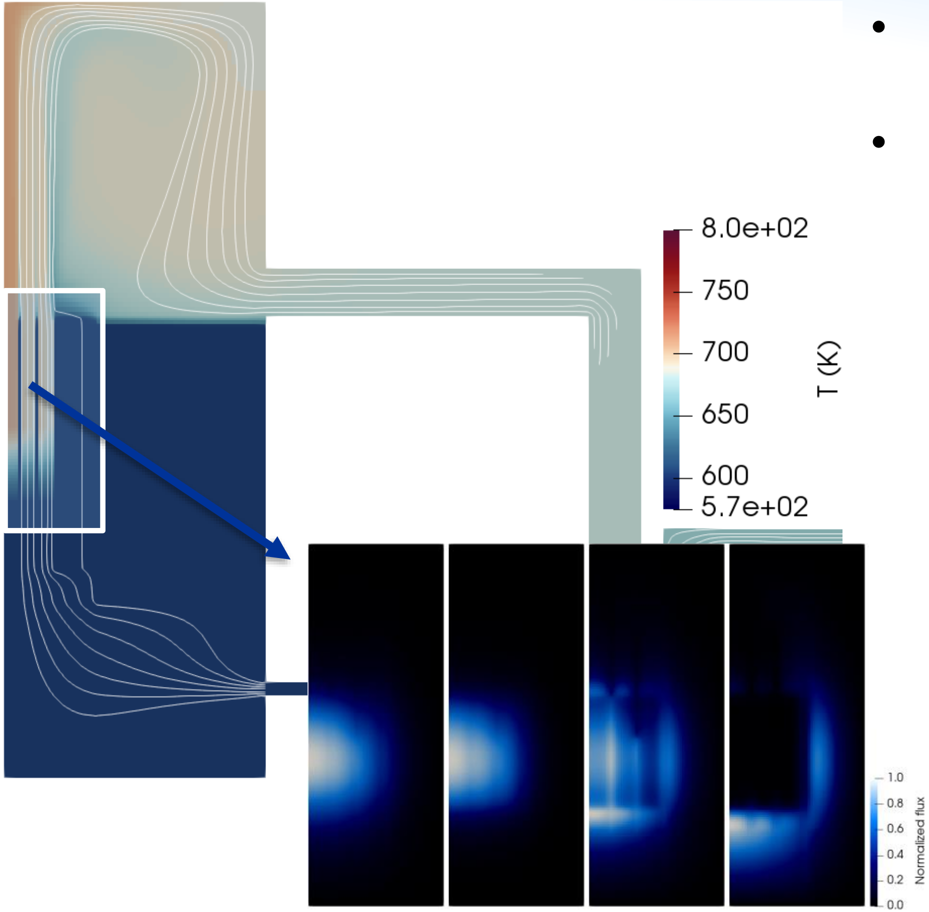
- Tutorials/lectures (have a look on Google or YouTube)
- Master/PhD thesis etc.
- Forums (including ours: https://foam-for-nuclear.org/phpBB/)
- (Often) direct communication with solver developers

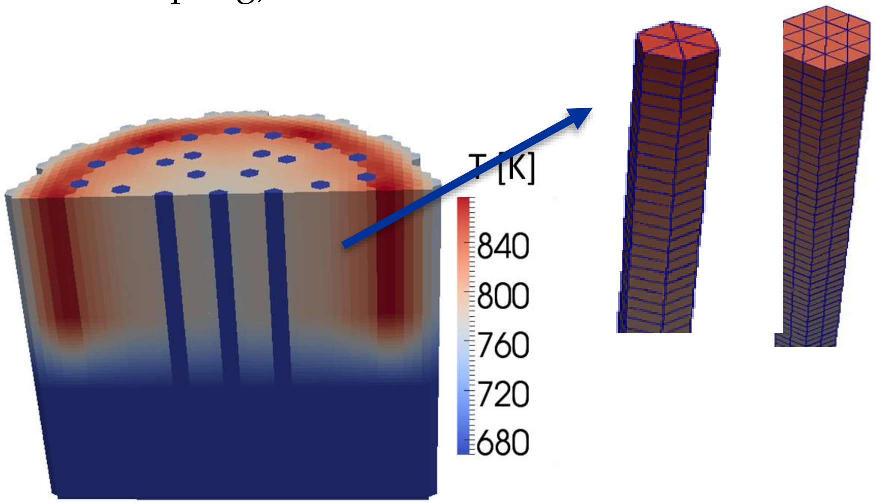
And remember:

- **Don’t get frustrated: there is always a way out with OpenFOAM and, most likely, someone who had your same problem and will be happy to help**
- **Don’t get discouraged: the entry barrier may seem steep, but skills you’ll learn will allow you to tackle any kind of problems**
- **If possible, do not do it alone!**

Additional background: multi-mesh

- Problem: need for different meshes for different "physics"
- Solution: multi-mesh (called multi-region in OpenFOAM)
  - One mesh for each "physics"
  - (Projection of fields from one mesh to the other for coupling)





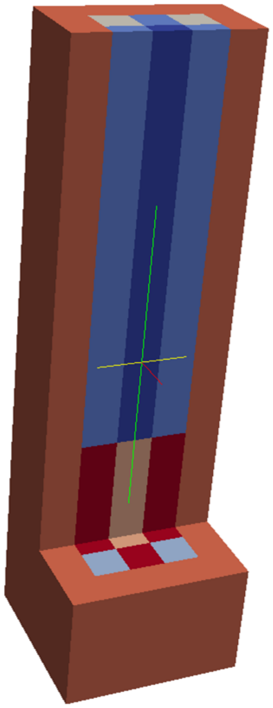
Additional background: multi-mesh in practice

```
• Case
  └ 0
     └ U
     └ T
     └ …
  └ constant
     └ turbulenceProperties
     └ …
  └ system
     └ fvSolution
     └ fvSchemes
     └ controDict
```

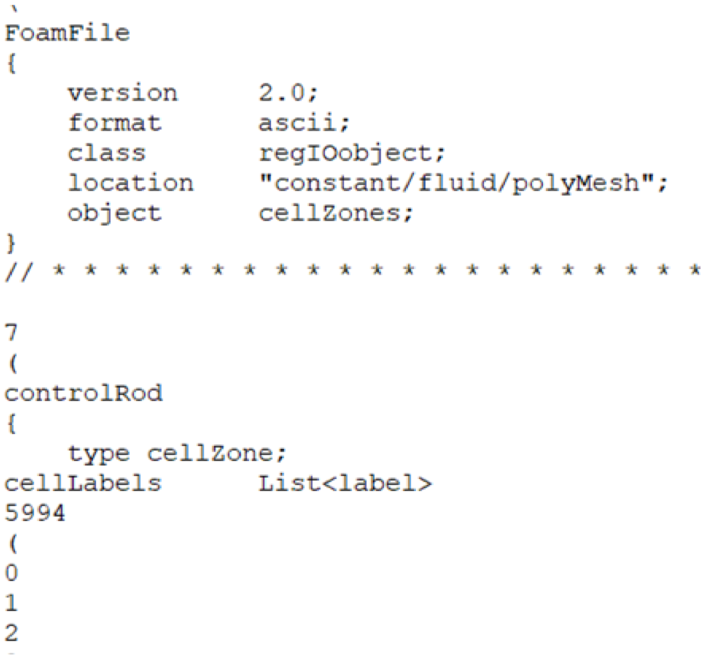
→

```
• Case
  └ 0
     └ neutroRegion
            Flux
            …
     └ fluidRegion
            U
            …
     └ thermoMechanicalRegion
            …
  └ constant
     └ neutroRegion
     └ fluidRegion
     └ thermoMechanicalRegion
     └ …
  └ system
     └ neutroRegion
     └ fluidRegion
     └ thermoMechanicalRegion
     └ …
```

Additional background: multi-material

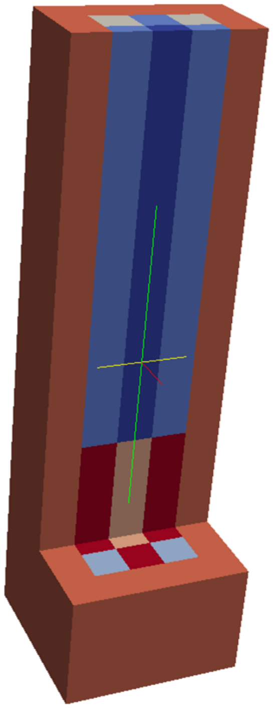


- Problem: one mesh, multiple material
- Solutions: cellZones
  - associate a label to each cell in polymesh/cellZones

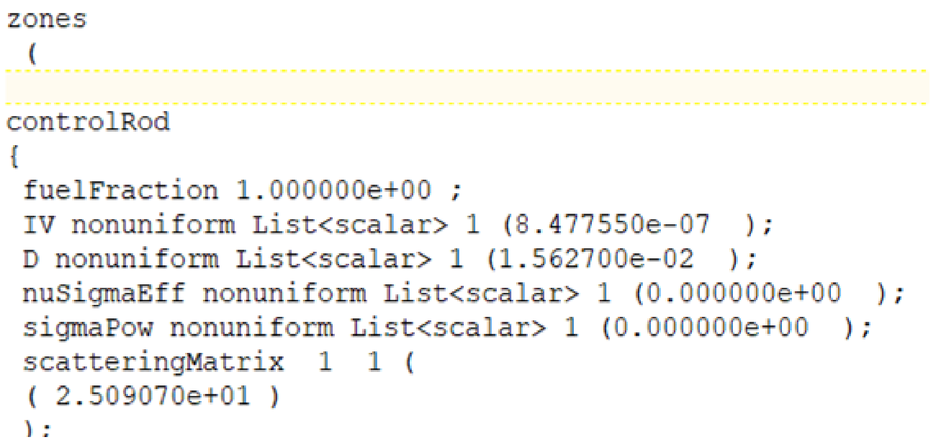


```
FoamFile
{
    version     2.0;
    format      ascii;
    class       regIOobject;
    location    "constant/fluid/polyMesh";
    object      cellZones;
}
// * * * * * * * * * * * * * * * * * * * * *

7
(
controlRod
{
    type cellZone;
cellLabels      List<label>
5994
(
0
1
2
```

Additional background: multi-material



- Then, for each physics, an input file (dictionary) is used that associates each of these labels with a set of properties. For instance in /constant/neutroRegion/nuclearData



```
zones
 (

controlRod
{
 fuelFraction 1.000000e+00 ;
 IV nonuniform List<scalar> 1 (8.477550e-07  );
 D nonuniform List<scalar> 1 (1.562700e-02  );
 nuSigmaEff nonuniform List<scalar> 1 (0.000000e+00  );
 sigmaPow nonuniform List<scalar> 1 (0.000000e+00  );
 scatteringMatrix  1  1 (
 ( 2.509070e+01 )
 );
```

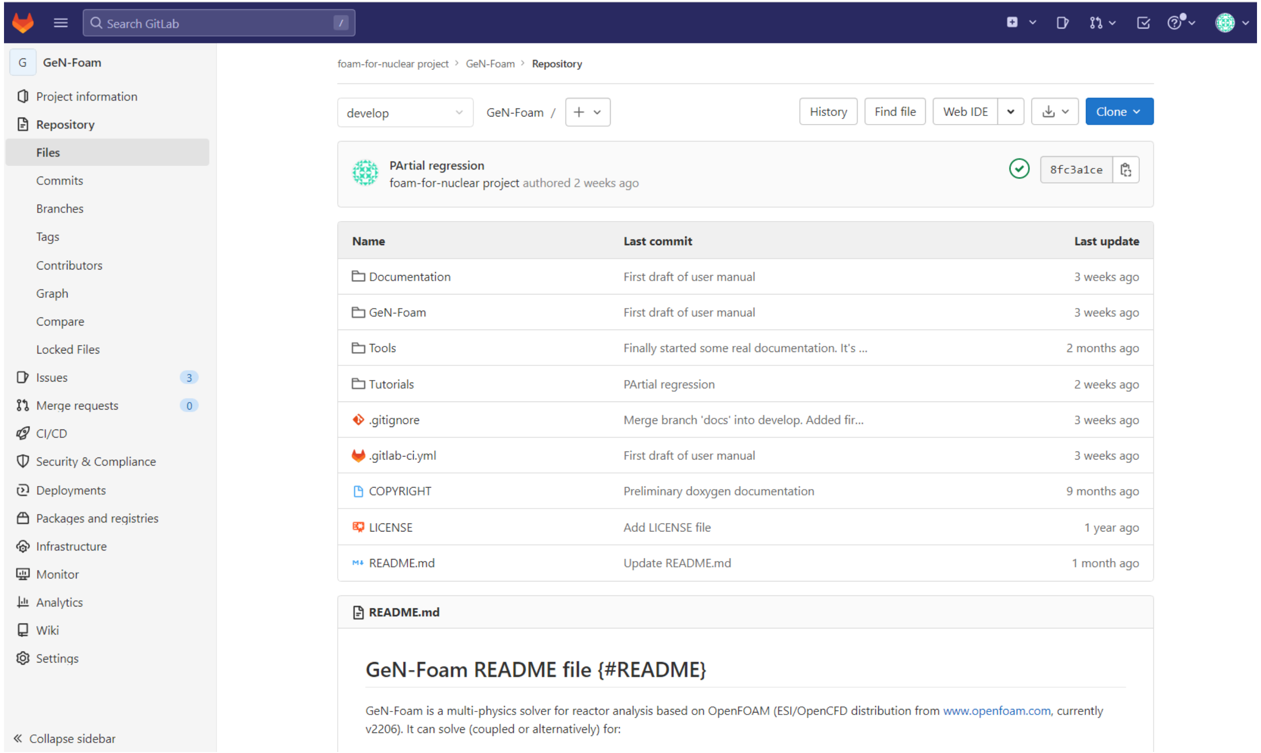
Additional background: multi-material - in practice

- How to create a multi-zone mesh:
  - All mesh generators allows for the option to generate “cellZones”
  - NB: cellZones are called in different ways (physical volumes in gmsh, groups in Salome, etc.)
  - The mesh conversion tool (e.g., gmshToFoam) takes care of converting the format

- Case folder:
  - polymesh folder will include a cellZones file
  - Dictionaries will be used to associate a cellZone to some value of a field or property

GeN-Foam: how to get it

- Free, online at https://gitlab.com/foam-for-nuclear/GeN-Foam/-/tree/develop



PArtial regression

foam-for-nuclear project authored 2 weeks ago

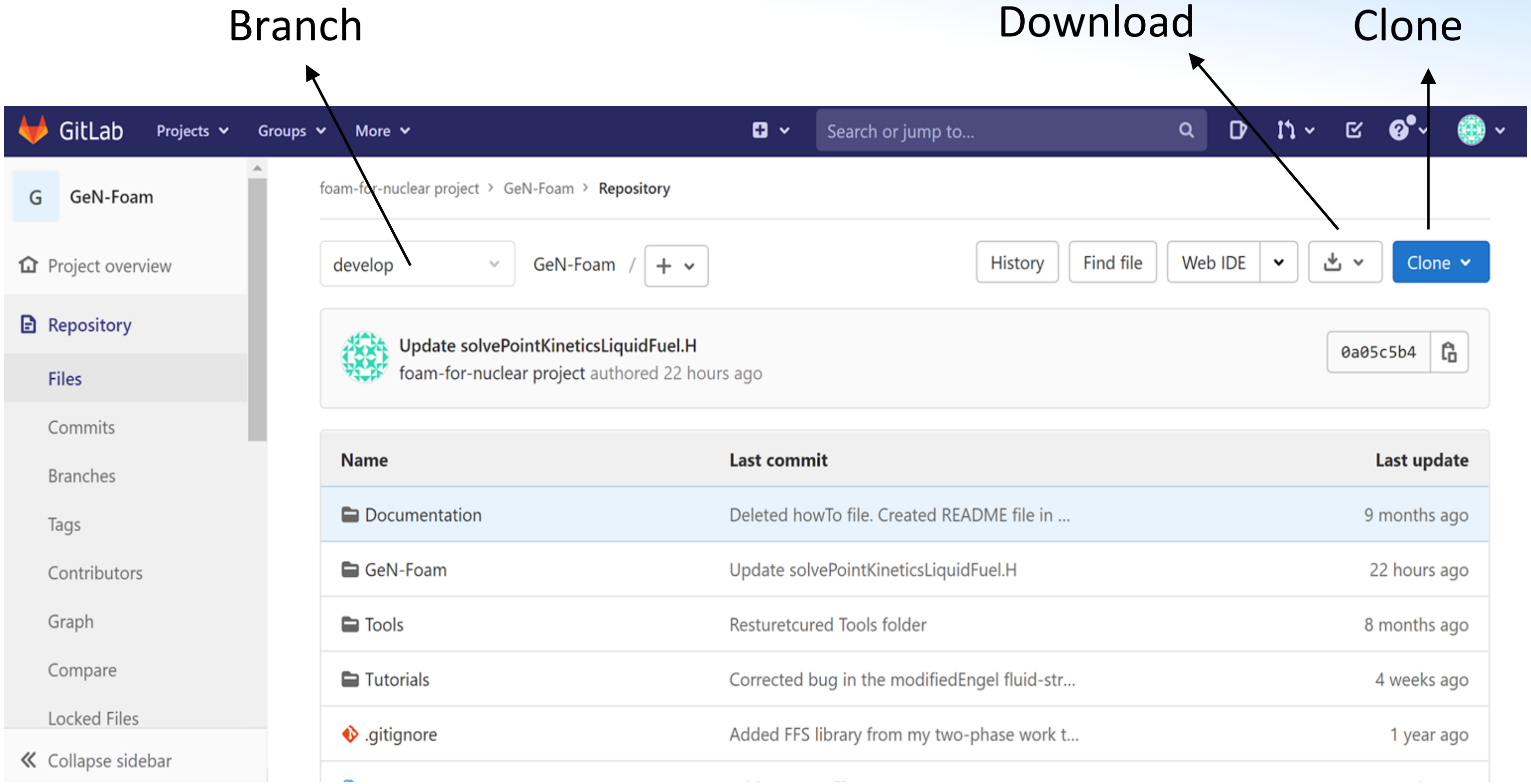
Name	Last commit	Last update
Documentation	First draft of user manual	3 weeks ago
GeN-Foam	First draft of user manual	3 weeks ago
Tools	Finally started some real documentation. It's ...	2 months ago
Tutorials	PArtial regression	2 weeks ago
.gitignore	Merge branch 'docs' into develop. Added fir...	3 weeks ago
.gitlab-ci.yml	First draft of user manual	3 weeks ago
COPYRIGHT	Preliminary doxygen documentation	9 months ago
LICENSE	Add LICENSE file	1 year ago
README.md	Update README.md	1 month ago

README.md

GeN-Foam README file {#README}

GeN-Foam is a multi-physics solver for reactor analysis based on OpenFOAM (ESI/OpenCFD distribution from www.openfoam.com, currently v2206). It can solve (coupled or alternatively) for:

GeN-Foam: how to get it



Branch

Download

Clone

foam-for-nuclear project › GeN-Foam › Repository

develop · GeN-Foam / + · History · Find file · Web IDE · Clone

Update solvePointKineticsLiquidFuel.H

foam-for-nuclear project authored 22 hours ago

0a05c5b4

Name	Last commit	Last update
Documentation	Deleted howTo file. Created README file in ...	9 months ago
GeN-Foam	Update solvePointKineticsLiquidFuel.H	22 hours ago
Tools	Resturetcured Tools folder	8 months ago
Tutorials	Corrected bug in the modifiedEngel fluid-str...	4 weeks ago
.gitignore	Added FFS library from my two-phase work t...	1 year ago

GeN-Foam: branches

- Several “working branches”
- Two main branches for distribution:
  - Develop: contain all recent tested developments. Normally stable. Full regression test before committing to the branch.
  - Master: most stable version. Merge from develop at every new OpenFOAM release (6 months)

GeN-Foam: how to install it

- Download OpenFOAM at
  - https://www.openfoam.com/download/
  - Typically the latest release, but it may take us some few weeks to update to a new release each time. The correct version to use is in the README file

- Install OpenFOAM and prepare the environment
  - https://www.openfoam.com/download/installation.php

- Download or git clone GeN-Foam

- Enter the GeN-Foam/GeN-Foam folder and run:
  - Allwclean
  - Allwmake (or Allwmake -j, to compile in parallel)

- Testing - enter any tutorial and run:
  - Allrun

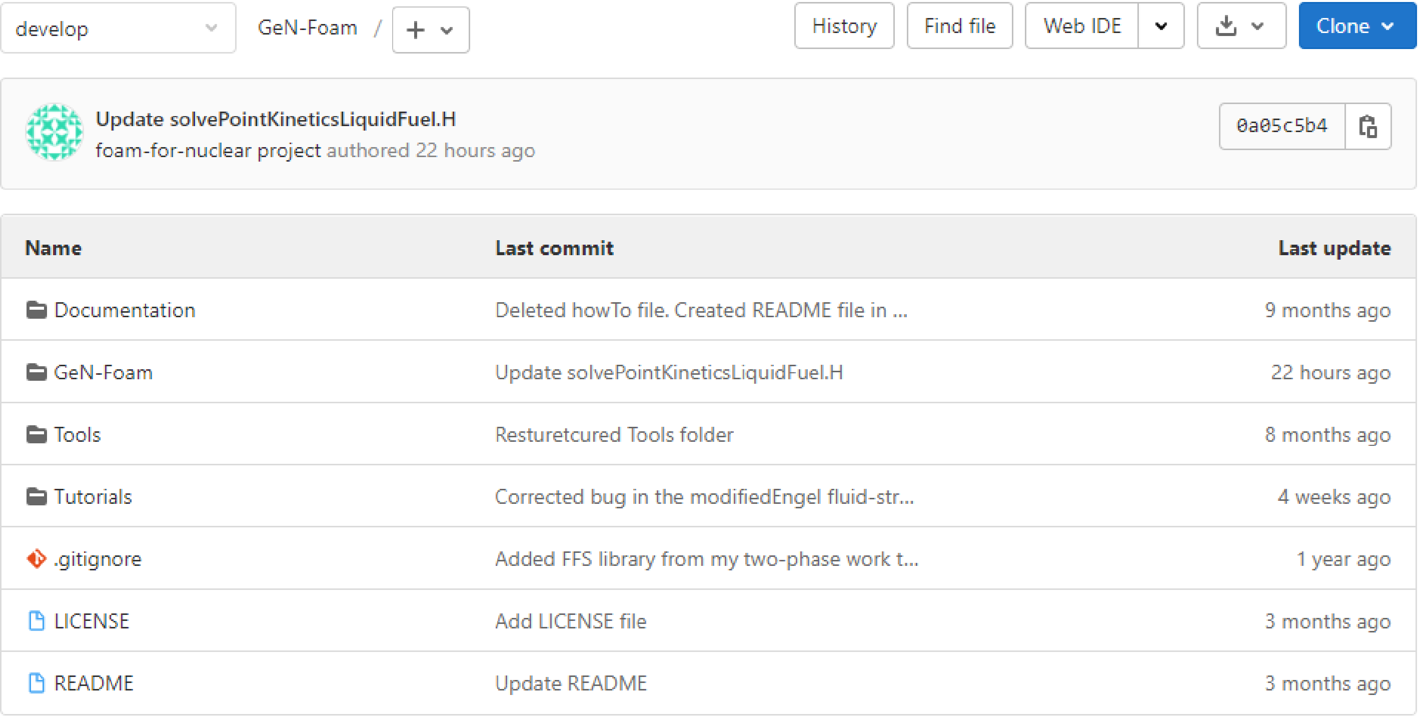
GeN-Foam: paraview

- Requires separate installation in the openfoam.com version of OpenFOAM
- Just install the latest version from paraview.org

Why isn't ParaView included in the precompiled packages? This would be much more convenient than having to compile it myself!

We would prefer to focus on extending and improving the OpenFOAM support in ParaView/VTK directly since this provides the best long-term and most universal solution

GeN-Foam: what’s inside



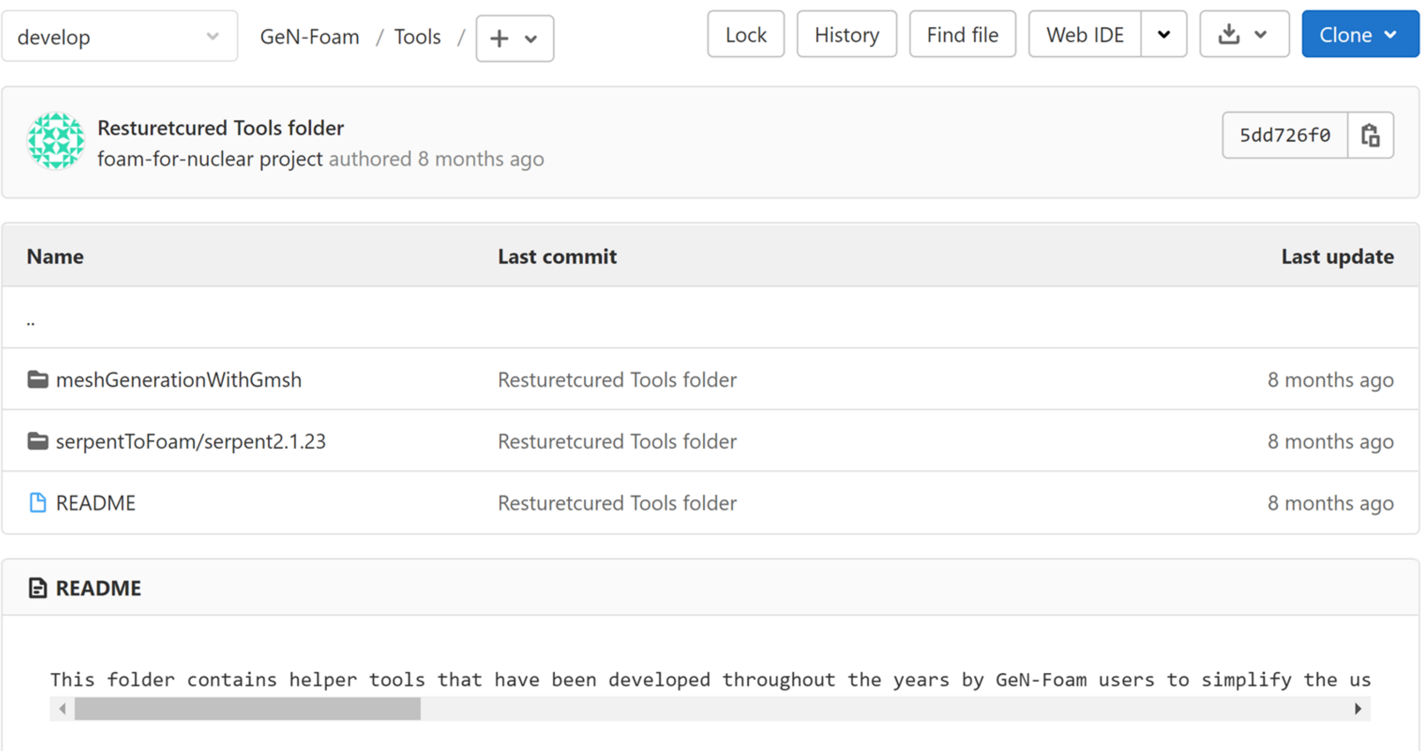
Update solvePointKineticsLiquidFuel.H
foam-for-nuclear project authored 22 hours ago

0a05c5b4

Name	Last commit	Last update
Documentation	Deleted howTo file. Created README file in ...	9 months ago
GeN-Foam	Update solvePointKineticsLiquidFuel.H	22 hours ago
Tools	Resturetcured Tools folder	8 months ago
Tutorials	Corrected bug in the modifiedEngel fluid-str...	4 weeks ago
.gitignore	Added FFS library from my two-phase work t...	1 year ago
LICENSE	Add LICENSE file	3 months ago
README	Update README	3 months ago

- README file often present to describe what’s in a subfolder

What's inside: Tools



Resturetcured Tools folder
foam-for-nuclear project authored 8 months ago

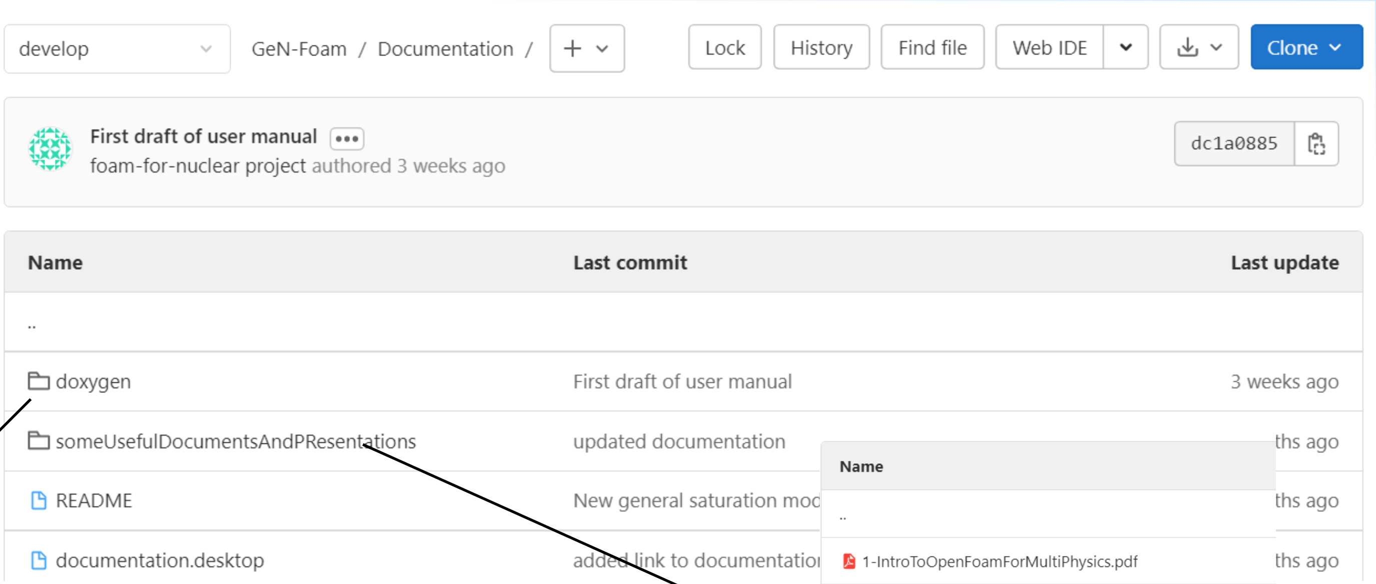
Name	Last commit	Last update
..		
meshGenerationWithGmsh	Resturetcured Tools folder	8 months ago
serpentToFoam/serpent2.1.23	Resturetcured Tools folder	8 months ago
README	Resturetcured Tools folder	8 months ago

README

```
This folder contains helper tools that have been developed throughout the years by GeN-Foam users to simplify the us
```

- Helper tools to (try to) make life of users easier
  - Example of a mesh creation with gmsh
  - Script to convert an output of Serpent into an input for GeN-Foam

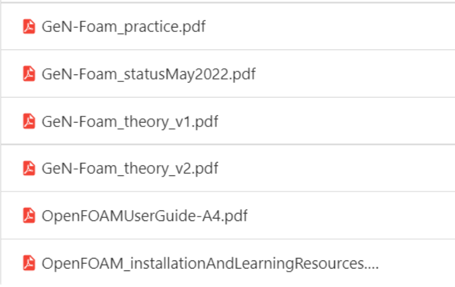
What’s inside: Documentation



First draft of user manual •••
foam-for-nuclear project authored 3 weeks ago

dc1a0885

Name	Last commit	Last update
..		
doxygen	First draft of user manual	3 weeks ago
someUsefulDocumentsAndPResentations	updated documentation	ths ago
README	New general saturation moc	ths ago
documentation.desktop	added link to documentatio	ths ago



Name
..
1-IntroToOpenFoamForMultiPhysics.pdf
GeN-Foam_practice.pdf
GeN-Foam_statusMay2022.pdf
GeN-Foam_theory_v1.pdf
GeN-Foam_theory_v2.pdf
OpenFOAMUserGuide-A4.pdf
OpenFOAM_installationAndLearningResources....

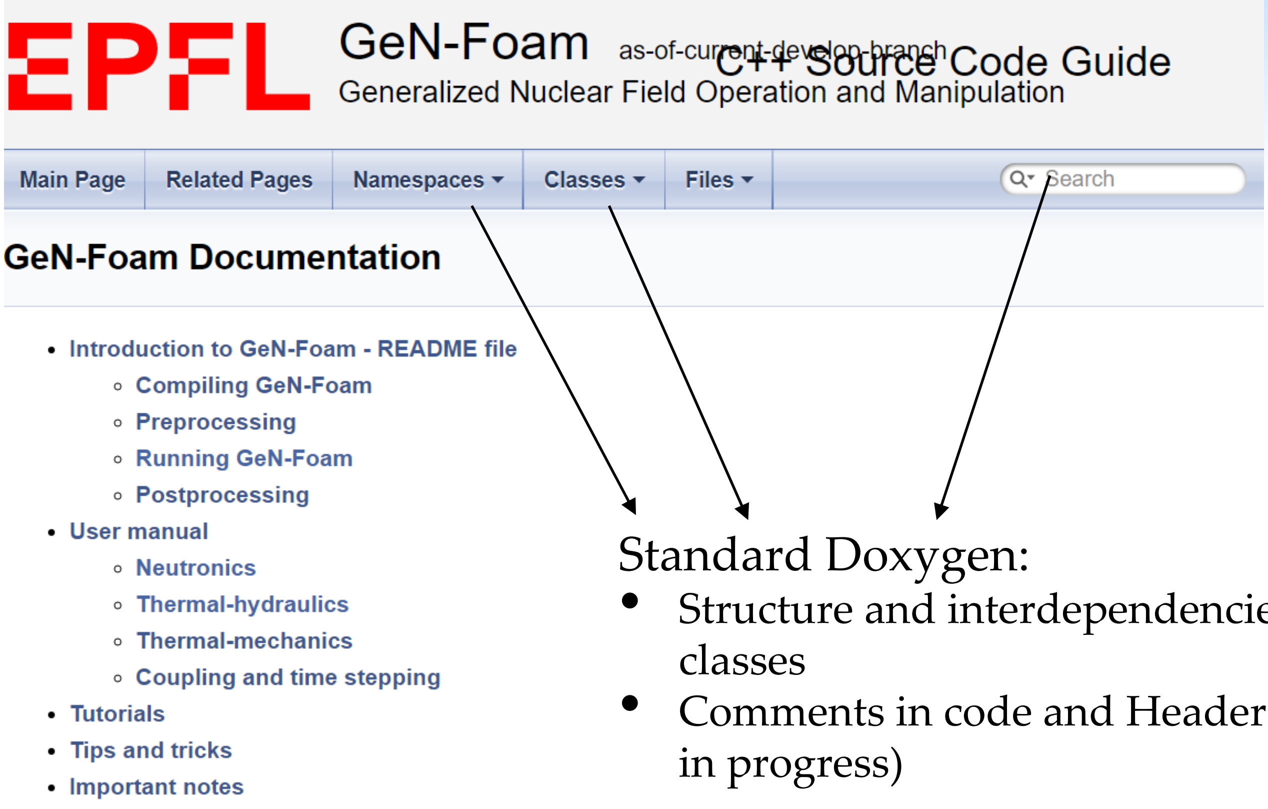
Doxygen-based documentation:

- Can be compiled locally on your machine
- Pre-compiled version available at:

https://foam-for-nuclear.gitlab.io/GeN-Foam/index.html

- Link available in the main README file

What’s inside: Documentation - doxygen



GeN-Foam as-of-current-develop-branch C++ Source Code Guide

Generalized Nuclear Field Operation and Manipulation

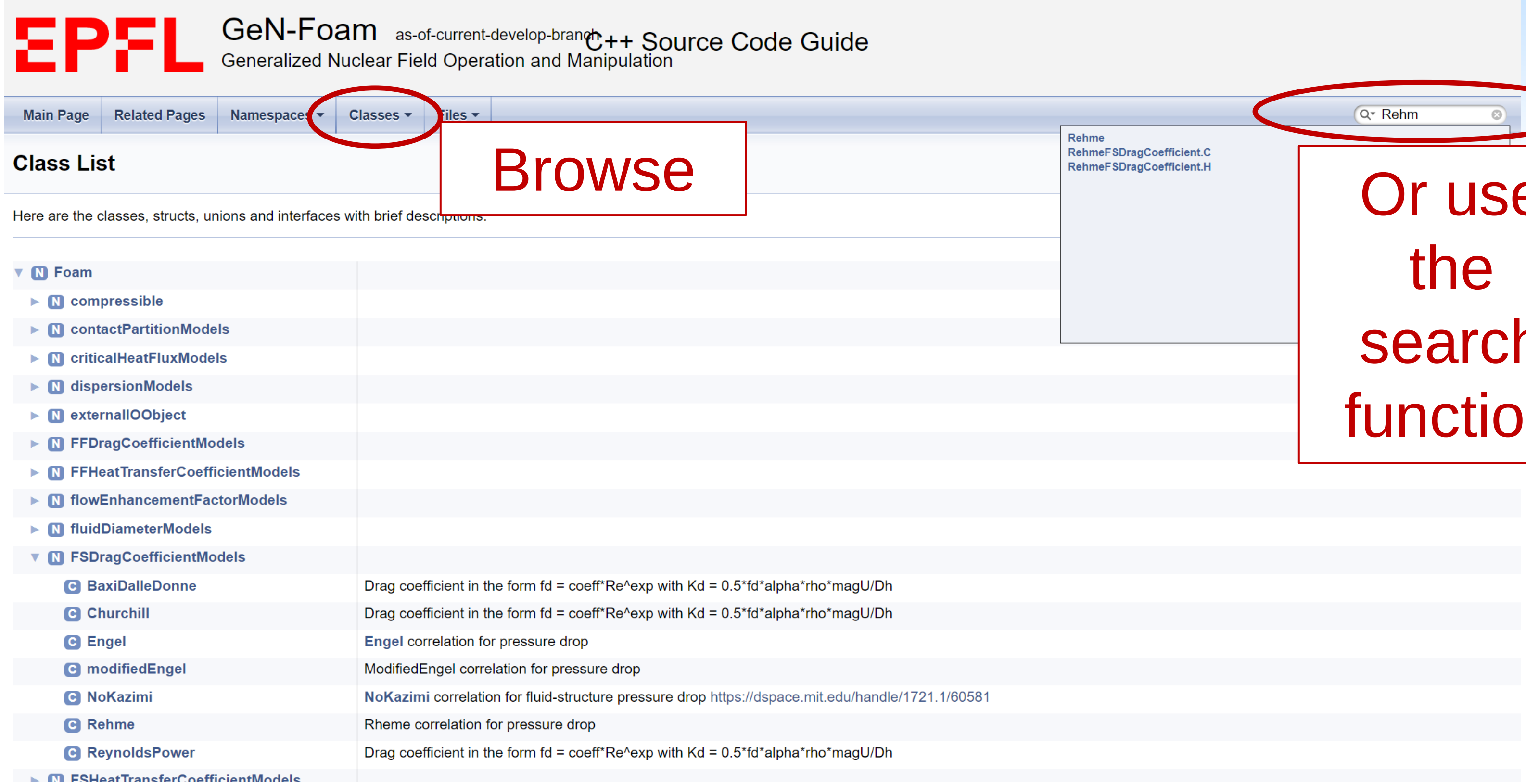
GeN-Foam Documentation

- Introduction to GeN-Foam - README file
  - Compiling GeN-Foam
  - Preprocessing
  - Running GeN-Foam
  - Postprocessing
- User manual
  - Neutronics
  - Thermal-hydraulics
  - Thermal-mechanics
  - Coupling and time stepping
- Tutorials
- Tips and tricks
- Important notes

Standard Doxygen:

- Structure and interdependencies of classes
- Comments in code and Headers (work in progress)

What’s inside: Documentation - doxygen


EPFL
GeN-Foam as-of-current-develop-branch C++ Source Code Guide
Generalized Nuclear Field Operation and Manipulation
Main Page
Related Pages
Namespaces
Classes
Files
Rehm
Rehme
RehmeFSDragCoefficient.C
RehmeFSDragCoefficient.H

Class List

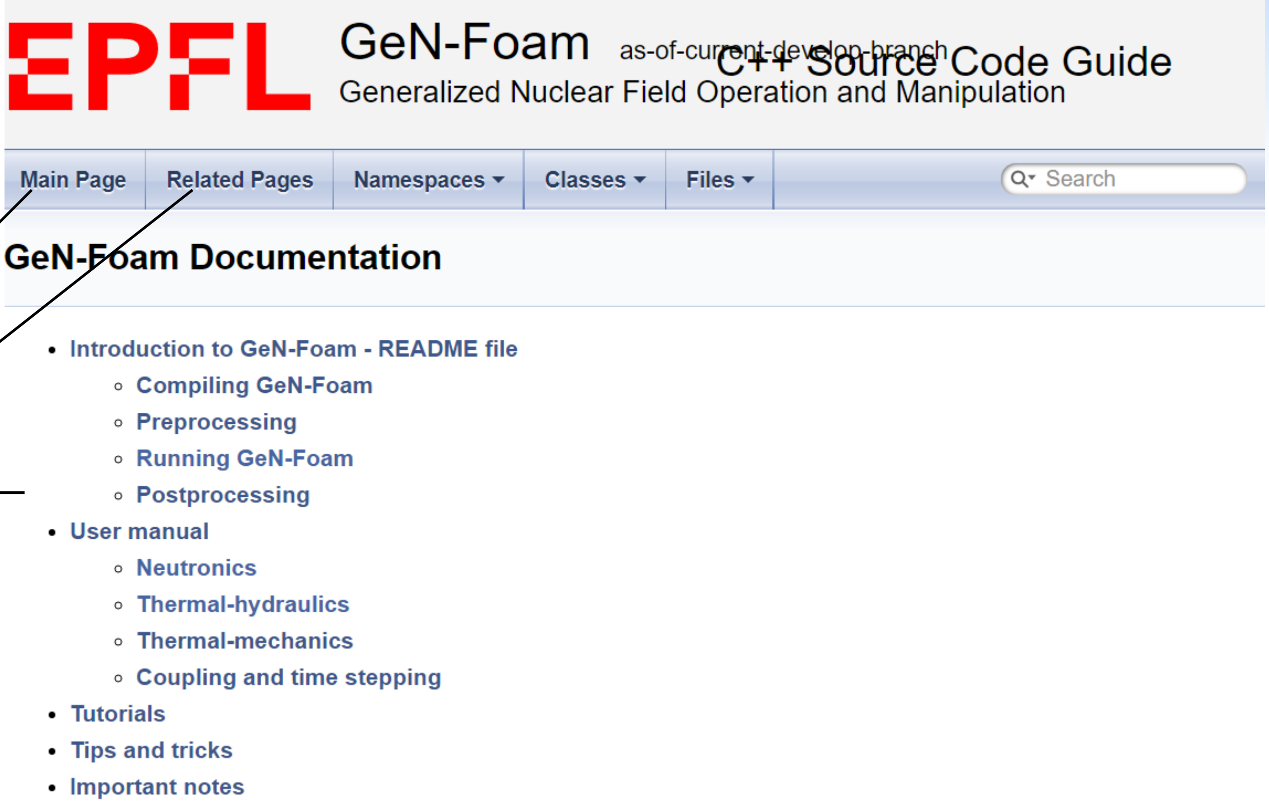
Here are the classes, structs, unions and interfaces with brief descriptions:

▼ N Foam	
▶ N compressible	
▶ N contactPartitionModels	
▶ N criticalHeatFluxModels	
▶ N dispersionModels	
▶ N externalOObject	
▶ N FFDragCoefficientModels	
▶ N FFHeatTransferCoefficientModels	
▶ N flowEnhancementFactorModels	
▶ N fluidDiameterModels	
▼ N FSDragCoefficientModels	
C BaxiDalleDonne	Drag coefficient in the form fd = coeff*Re^exp with Kd = 0.5*fd*alpha*rho*magU/Dh
C Churchill	Drag coefficient in the form fd = coeff*Re^exp with Kd = 0.5*fd*alpha*rho*magU/Dh
C Engel	Engel correlation for pressure drop
C modifiedEngel	ModifiedEngel correlation for pressure drop
C NoKazimi	NoKazimi correlation for fluid-structure pressure drop https://dspace.mit.edu/handle/1721.1/60581
C Rehme	Rheme correlation for pressure drop
C ReynoldsPower	Drag coefficient in the form fd = coeff*Re^exp with Kd = 0.5*fd*alpha*rho*magU/Dh
▶ N FSHeatTransferCoefficientModels	

Browse

Or use the search function

What's inside: Documentation - doxygen



GeN-Foam as-of-current-develop-branch C++ Source Code Guide

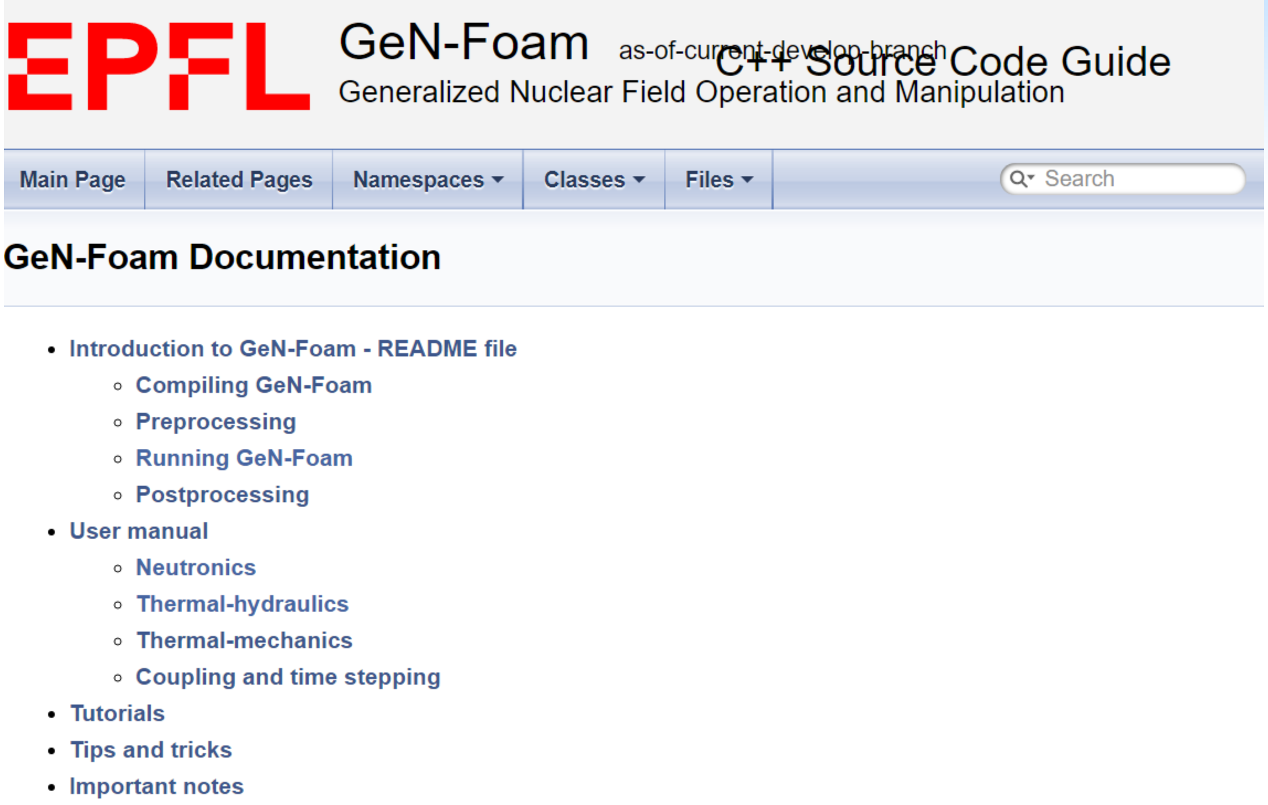
Generalized Nuclear Field Operation and Manipulation

GeN-Foam Documentation

- Introduction to GeN-Foam - README file
  - Compiling GeN-Foam
  - Preprocessing
  - Running GeN-Foam
  - Postprocessing
- User manual
  - Neutronics
  - Thermal-hydraulics
  - Thermal-mechanics
  - Coupling and time stepping
- Tutorials
- Tips and tricks
- Important notes

Additional pages

What’s inside: Documentation – how to us it

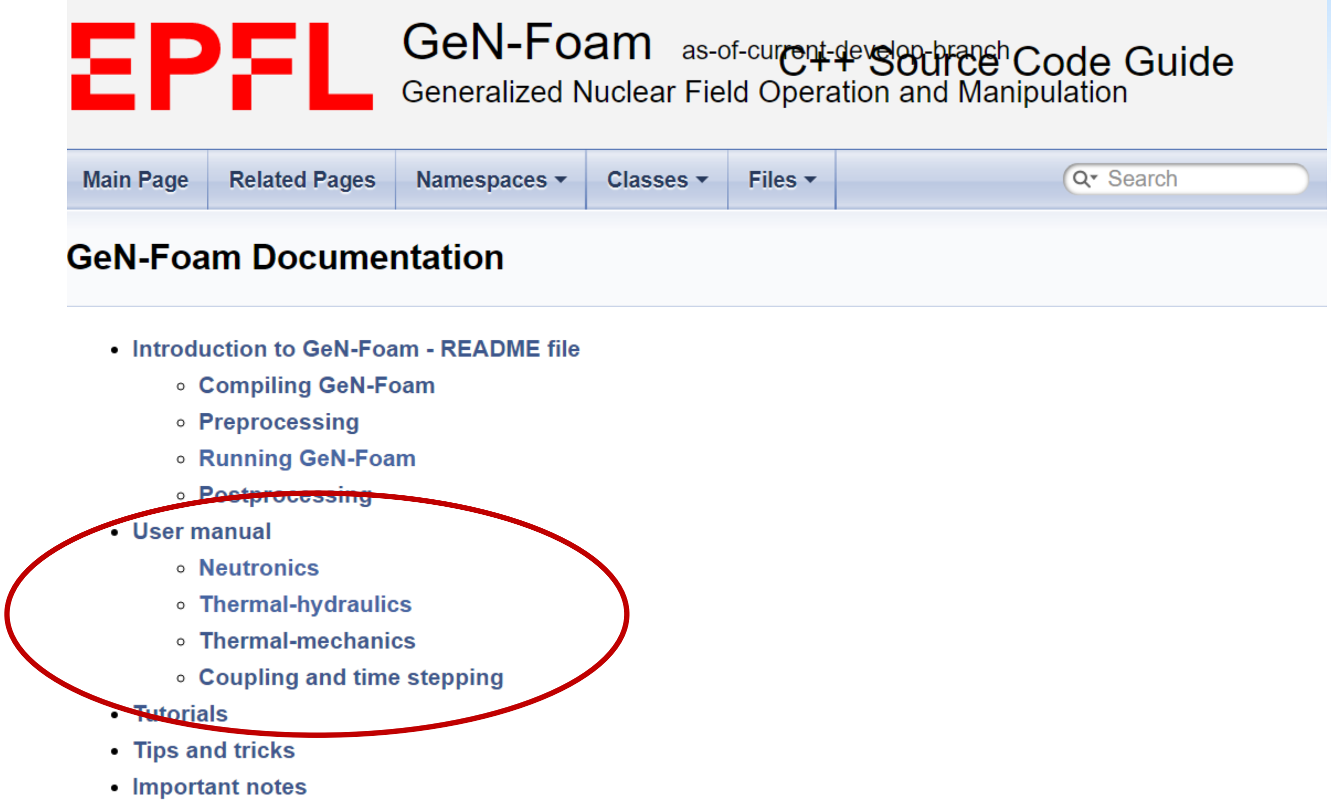


GeN-Foam as-of-current-develop-branch C++ Source Code Guide
Generalized Nuclear Field Operation and Manipulation

GeN-Foam Documentation

- Introduction to GeN-Foam - README file
  - Compiling GeN-Foam
  - Preprocessing
  - Running GeN-Foam
  - Postprocessing
- User manual
  - Neutronics
  - Thermal-hydraulics
  - Thermal-mechanics
  - Coupling and time stepping
- Tutorials
- Tips and tricks
- Important notes

What’s inside: Documentation – how to us it



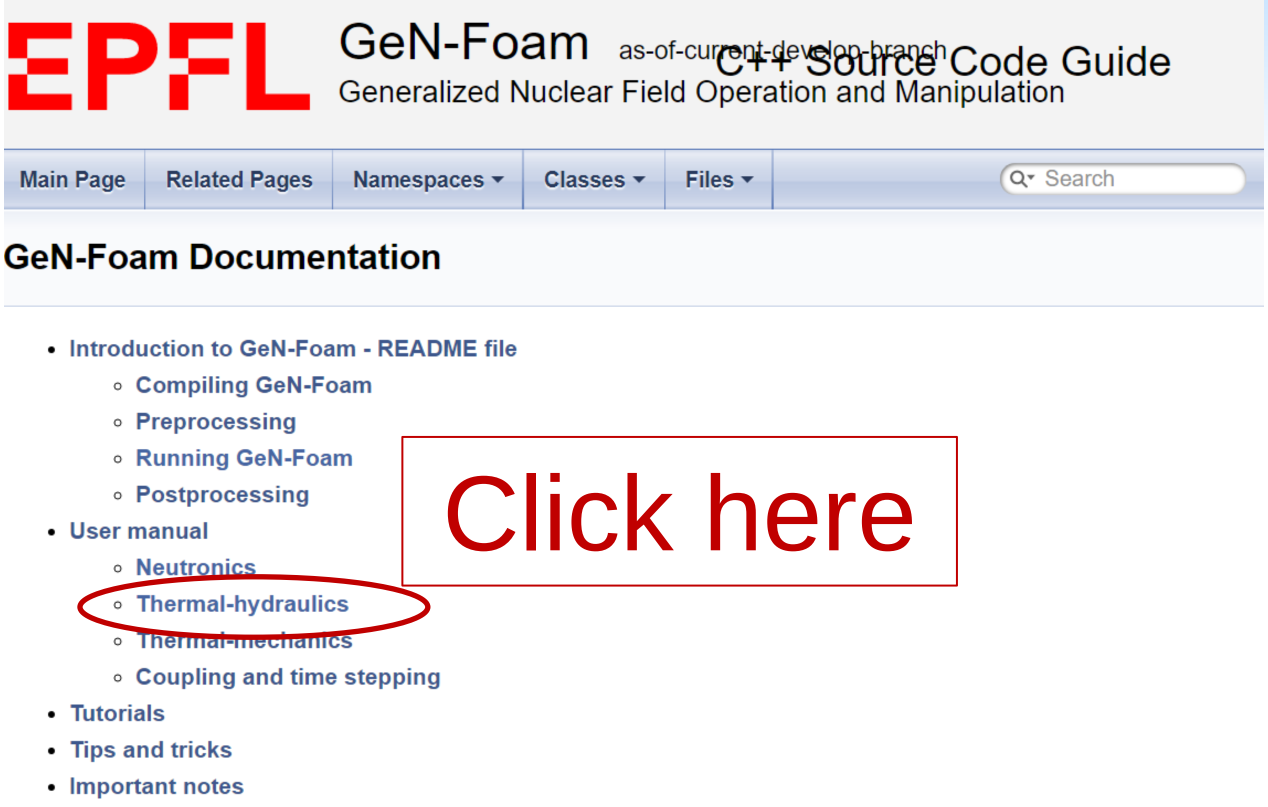
GeN-Foam as-of-current-develop-branch C++ Source Code Guide

Generalized Nuclear Field Operation and Manipulation

GeN-Foam Documentation

- Introduction to GeN-Foam - README file
  - Compiling GeN-Foam
  - Preprocessing
  - Running GeN-Foam
  - Postprocessing
- User manual
  - Neutronics
  - Thermal-hydraulics
  - Thermal-mechanics
  - Coupling and time stepping
- Tutorials
- Tips and tricks
- Important notes

What’s inside: Documentation – turbulence



GeN-Foam as-of-current-develop-branch C++ Source Code Guide

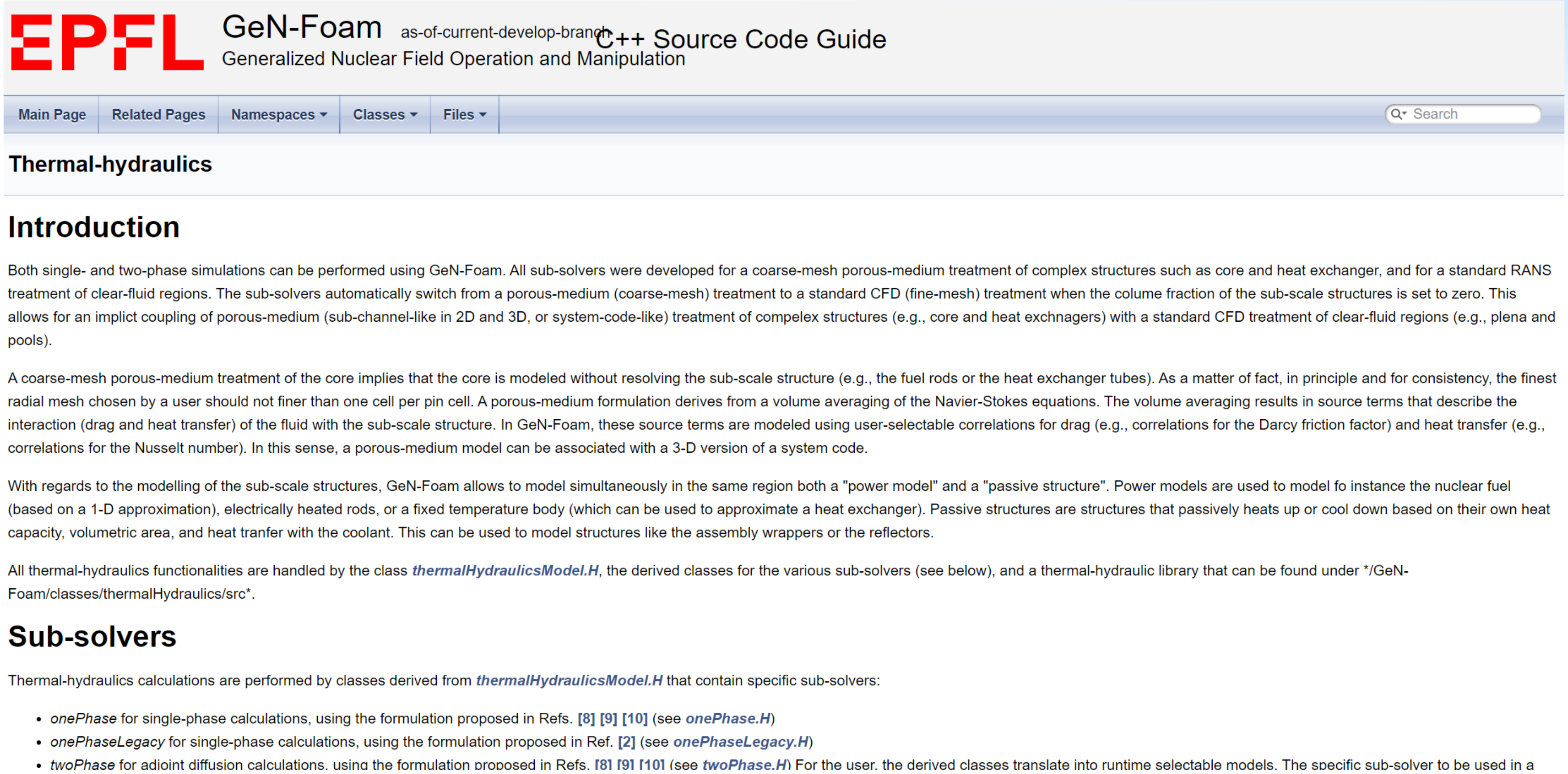
Generalized Nuclear Field Operation and Manipulation

GeN-Foam Documentation

- Introduction to GeN-Foam - README file
  - Compiling GeN-Foam
  - Preprocessing
  - Running GeN-Foam
  - Postprocessing
- User manual
  - Neutronics
  - Thermal-hydraulics
  - Thermal-mechanics
  - Coupling and time stepping
- Tutorials
- Tips and tricks
- Important notes

Click here

What's inside: Documentation – turbulence



Thermal-hydraulics

Introduction

Both single- and two-phase simulations can be performed using GeN-Foam. All sub-solvers were developed for a coarse-mesh porous-medium treatment of complex structures such as core and heat exchanger, and for a standard RANS treatment of clear-fluid regions. The sub-solvers automatically switch from a porous-medium (coarse-mesh) treatment to a standard CFD (fine-mesh) treatment when the colume fraction of the sub-scale structures is set to zero. This allows for an implict coupling of porous-medium (sub-channel-like in 2D and 3D, or system-code-like) treatment of compelex structures (e.g., core and heat exchnagers) with a standard CFD treatment of clear-fluid regions (e.g., plena and pools).

A coarse-mesh porous-medium treatment of the core implies that the core is modeled without resolving the sub-scale structure (e.g., the fuel rods or the heat exchanger tubes). As a matter of fact, in principle and for consistency, the finest radial mesh chosen by a user should not finer than one cell per pin cell. A porous-medium formulation derives from a volume averaging of the Navier-Stokes equations. The volume averaging results in source terms that describe the interaction (drag and heat transfer) of the fluid with the sub-scale structure. In GeN-Foam, these source terms are modeled using user-selectable correlations for drag (e.g., correlations for the Darcy friction factor) and heat transfer (e.g., correlations for the Nusselt number). In this sense, a porous-medium model can be associated with a 3-D version of a system code.

With regards to the modelling of the sub-scale structures, GeN-Foam allows to model simultaneously in the same region both a "power model" and a "passive structure". Power models are used to model fo instance the nuclear fuel (based on a 1-D approximation), electrically heated rods, or a fixed temperature body (which can be used to approximate a heat exchanger). Passive structures are structures that passively heats up or cool down based on their own heat capacity, volumetric area, and heat tranfer with the coolant. This can be used to model structures like the assembly wrappers or the reflectors.

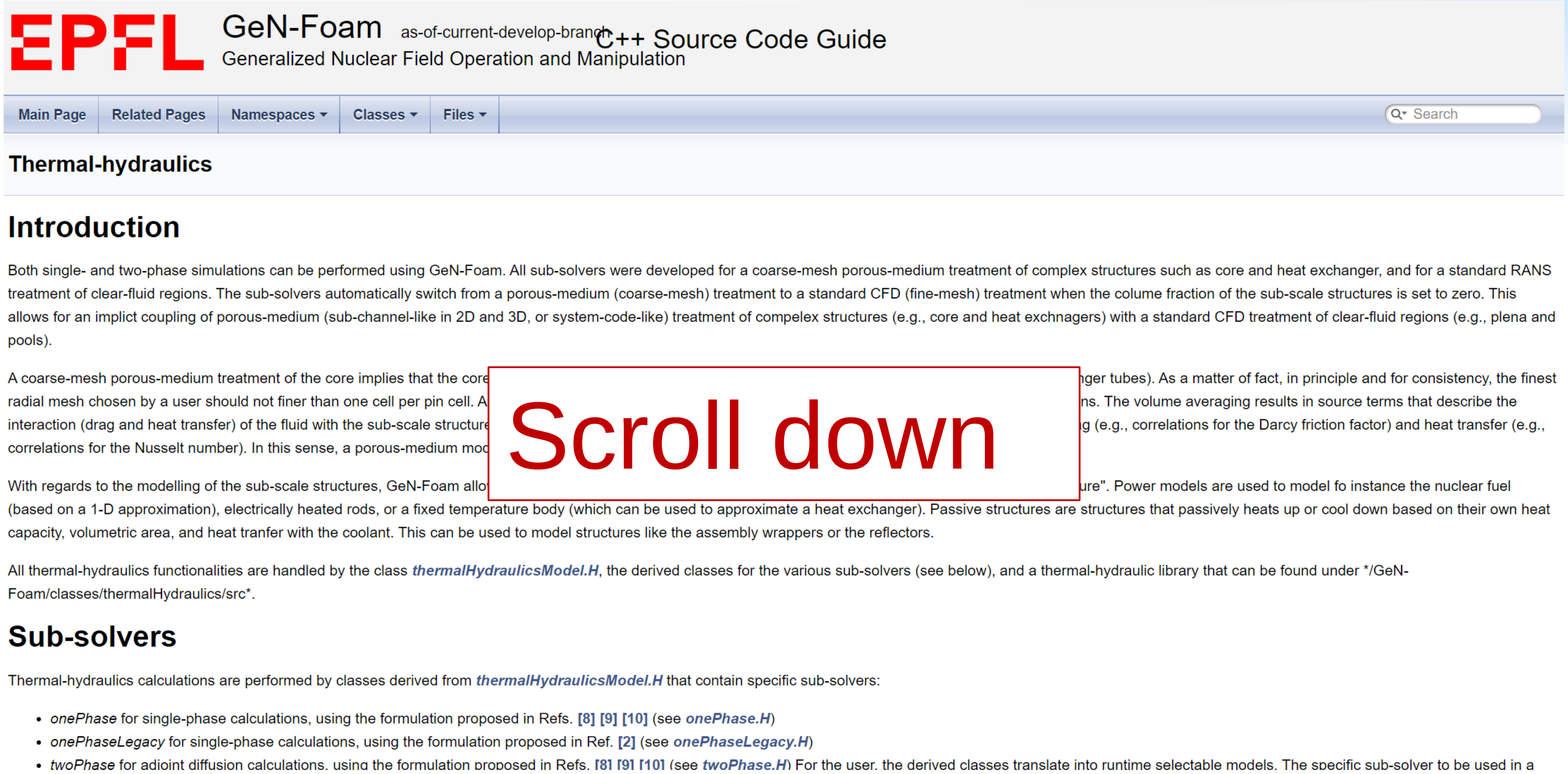
All thermal-hydraulics functionalities are handled by the class *thermalHydraulicsModel.H*, the derived classes for the various sub-solvers (see below), and a thermal-hydraulic library that can be found under */GeN-Foam/classes/thermalHydraulics/src*.

Sub-solvers

Thermal-hydraulics calculations are performed by classes derived from *thermalHydraulicsModel.H* that contain specific sub-solvers:

- *onePhase* for single-phase calculations, using the formulation proposed in Refs. [8] [9] [10] (see *onePhase.H*)
- *onePhaseLegacy* for single-phase calculations, using the formulation proposed in Ref. [2] (see *onePhaseLegacy.H*)
- *twoPhase* for adjoint diffusion calculations, using the formulation proposed in Refs. [8] [9] [10] (see *twoPhase.H*) For the user, the derived classes translate into runtime selectable models. The specific sub-solver to be used in a

What's inside: Documentation – turbulence


EPFL
GeN-Foam as-of-current-develop-branch C++ Source Code Guide
Generalized Nuclear Field Operation and Manipulation
Main Page
Related Pages
Namespaces ▾
Classes ▾
Files ▾
Search

Thermal-hydraulics

Introduction

Both single- and two-phase simulations can be performed using GeN-Foam. All sub-solvers were developed for a coarse-mesh porous-medium treatment of complex structures such as core and heat exchanger, and for a standard RANS treatment of clear-fluid regions. The sub-solvers automatically switch from a porous-medium (coarse-mesh) treatment to a standard CFD (fine-mesh) treatment when the colume fraction of the sub-scale structures is set to zero. This allows for an implict coupling of porous-medium (sub-channel-like in 2D and 3D, or system-code-like) treatment of compelex structures (e.g., core and heat exchnagers) with a standard CFD treatment of clear-fluid regions (e.g., plena and pools).

A coarse-mesh porous-medium treatment of the core implies that the core … nger tubes). As a matter of fact, in principle and for consistency, the finest radial mesh chosen by a user should not finer than one cell per pin cell. A … ns. The volume averaging results in source terms that describe the interaction (drag and heat transfer) of the fluid with the sub-scale structure … g (e.g., correlations for the Darcy friction factor) and heat transfer (e.g., correlations for the Nusselt number). In this sense, a porous-medium mod…

Scroll down

With regards to the modelling of the sub-scale structures, GeN-Foam allo… ure". Power models are used to model fo instance the nuclear fuel (based on a 1-D approximation), electrically heated rods, or a fixed temperature body (which can be used to approximate a heat exchanger). Passive structures are structures that passively heats up or cool down based on their own heat capacity, volumetric area, and heat tranfer with the coolant. This can be used to model structures like the assembly wrappers or the reflectors.

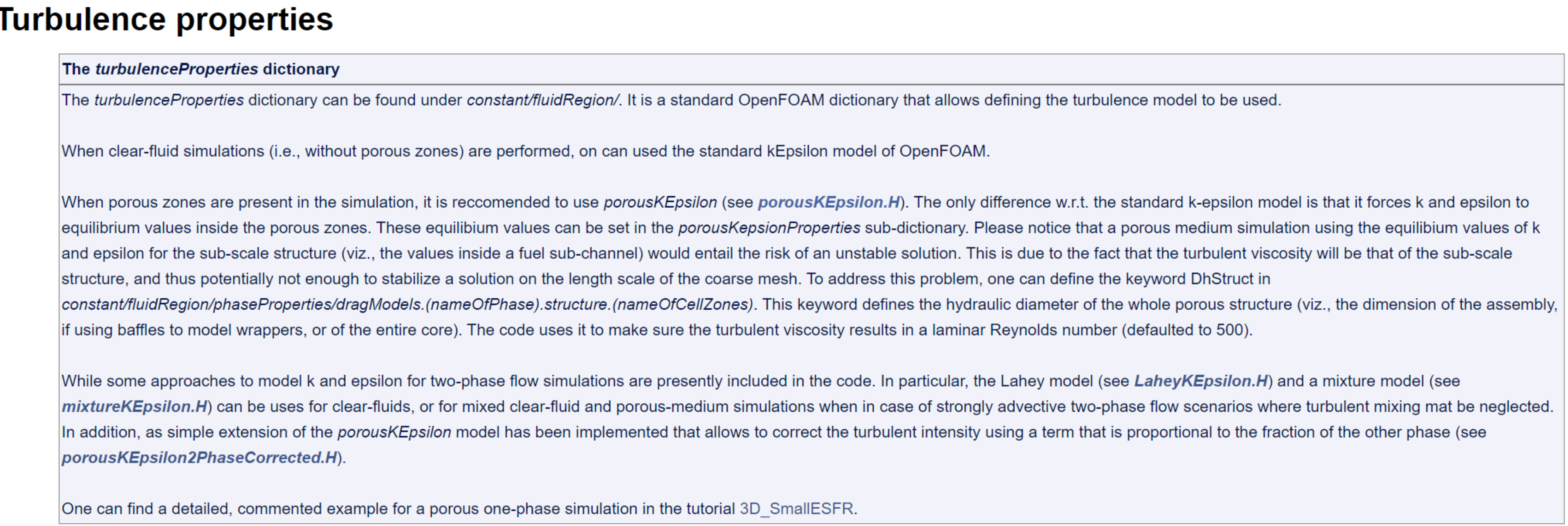
All thermal-hydraulics functionalities are handled by the class *thermalHydraulicsModel.H*, the derived classes for the various sub-solvers (see below), and a thermal-hydraulic library that can be found under */GeN-Foam/classes/thermalHydraulics/src*.

Sub-solvers

Thermal-hydraulics calculations are performed by classes derived from *thermalHydraulicsModel.H* that contain specific sub-solvers:

- *onePhase* for single-phase calculations, using the formulation proposed in Refs. [8] [9] [10] (see *onePhase.H*)
- *onePhaseLegacy* for single-phase calculations, using the formulation proposed in Ref. [2] (see *onePhaseLegacy.H*)
- *twoPhase* for adjoint diffusion calculations, using the formulation proposed in Refs. [8] [9] [10] (see *twoPhase.H*) For the user, the derived classes translate into runtime selectable models. The specific sub-solver to be used in a

What’s inside: Documentation – turbulence



Turbulence properties

The *turbulenceProperties* dictionary

The *turbulenceProperties* dictionary can be found under *constant/fluidRegion/*. It is a standard OpenFOAM dictionary that allows defining the turbulence model to be used.

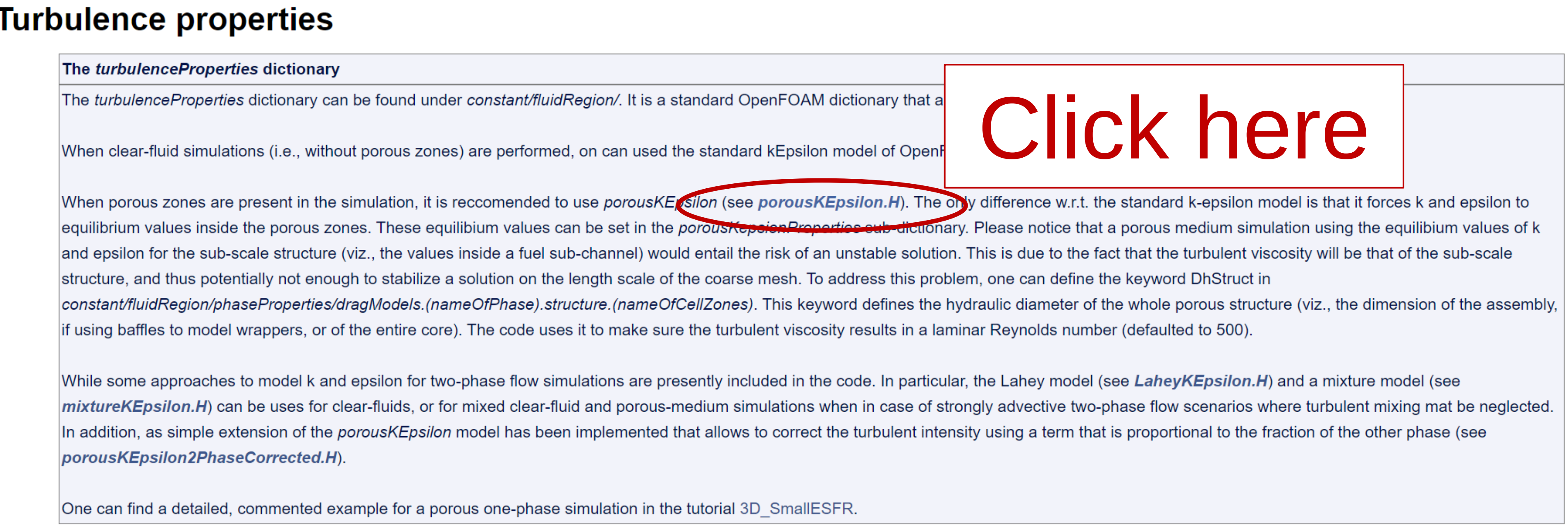
When clear-fluid simulations (i.e., without porous zones) are performed, on can used the standard kEpsilon model of OpenFOAM.

When porous zones are present in the simulation, it is reccomended to use *porousKEpsilon* (see ***porousKEpsilon.H***). The only difference w.r.t. the standard k-epsilon model is that it forces k and epsilon to equilibrium values inside the porous zones. These equilibium values can be set in the *porousKepsionProperties* sub-dictionary. Please notice that a porous medium simulation using the equilibium values of k and epsilon for the sub-scale structure (viz., the values inside a fuel sub-channel) would entail the risk of an unstable solution. This is due to the fact that the turbulent viscosity will be that of the sub-scale structure, and thus potentially not enough to stabilize a solution on the length scale of the coarse mesh. To address this problem, one can define the keyword DhStruct in *constant/fluidRegion/phaseProperties/dragModels.(nameOfPhase).structure.(nameOfCellZones)*. This keyword defines the hydraulic diameter of the whole porous structure (viz., the dimension of the assembly, if using baffles to model wrappers, or of the entire core). The code uses it to make sure the turbulent viscosity results in a laminar Reynolds number (defaulted to 500).

While some approaches to model k and epsilon for two-phase flow simulations are presently included in the code. In particular, the Lahey model (see ***LaheyKEpsilon.H***) and a mixture model (see ***mixtureKEpsilon.H***) can be uses for clear-fluids, or for mixed clear-fluid and porous-medium simulations when in case of strongly advective two-phase flow scenarios where turbulent mixing mat be neglected. In addition, as simple extension of the *porousKEpsilon* model has been implemented that allows to correct the turbulent intensity using a term that is proportional to the fraction of the other phase (see ***porousKEpsilon2PhaseCorrected.H***).

One can find a detailed, commented example for a porous one-phase simulation in the tutorial 3D_SmallESFR.

What’s inside: Documentation – turbulence



Turbulence properties

The *turbulenceProperties* dictionary

The *turbulenceProperties* dictionary can be found under *constant/fluidRegion/*. It is a standard OpenFOAM dictionary that a

When clear-fluid simulations (i.e., without porous zones) are performed, on can used the standard kEpsilon model of OpenF

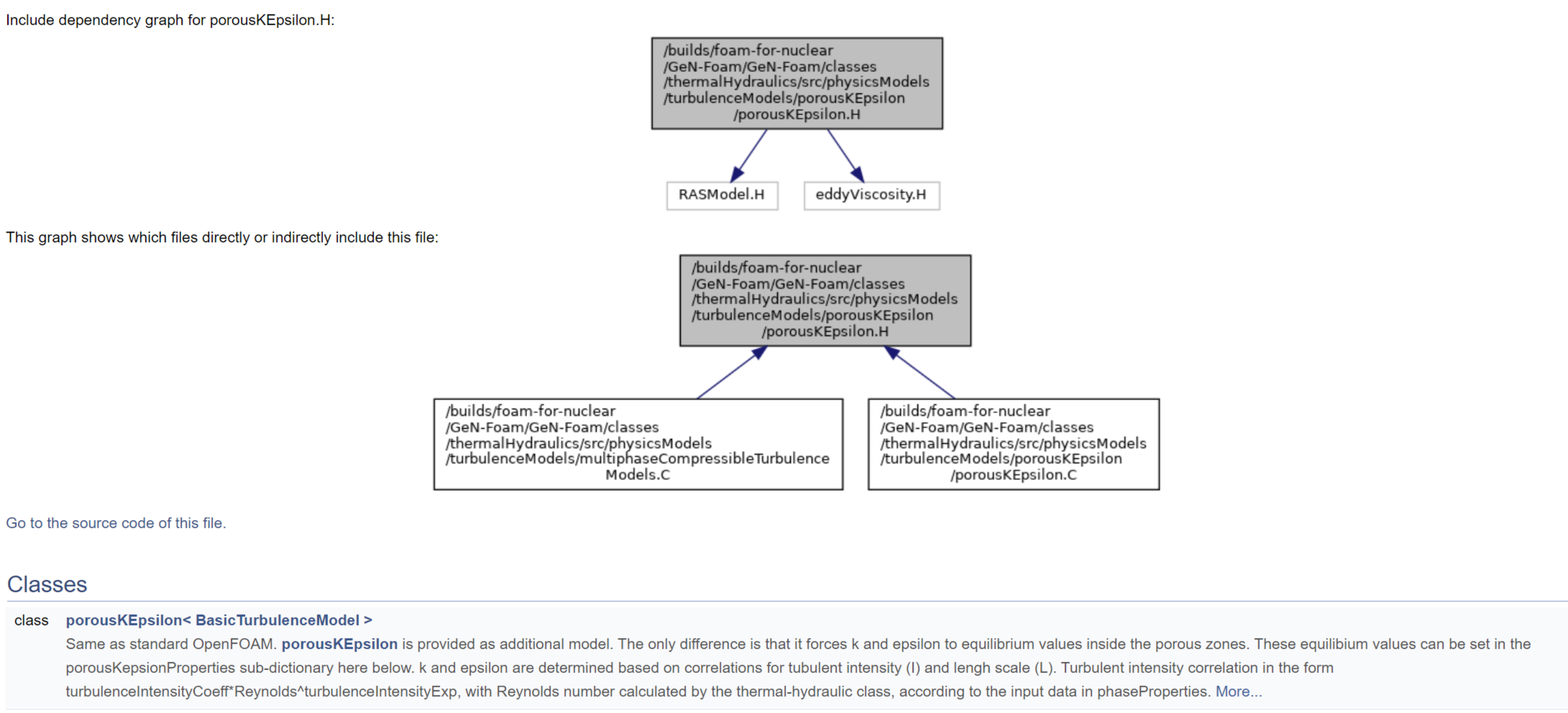
When porous zones are present in the simulation, it is reccomended to use *porousKEpsilon* (see ***porousKEpsilon.H***). The only difference w.r.t. the standard k-epsilon model is that it forces k and epsilon to equilibrium values inside the porous zones. These equilibium values can be set in the *porousKepsionProperties* sub-dictionary. Please notice that a porous medium simulation using the equilibium values of k and epsilon for the sub-scale structure (viz., the values inside a fuel sub-channel) would entail the risk of an unstable solution. This is due to the fact that the turbulent viscosity will be that of the sub-scale structure, and thus potentially not enough to stabilize a solution on the length scale of the coarse mesh. To address this problem, one can define the keyword DhStruct in *constant/fluidRegion/phaseProperties/dragModels.(nameOfPhase).structure.(nameOfCellZones)*. This keyword defines the hydraulic diameter of the whole porous structure (viz., the dimension of the assembly, if using baffles to model wrappers, or of the entire core). The code uses it to make sure the turbulent viscosity results in a laminar Reynolds number (defaulted to 500).

While some approaches to model k and epsilon for two-phase flow simulations are presently included in the code. In particular, the Lahey model (see ***LaheyKEpsilon.H***) and a mixture model (see ***mixtureKEpsilon.H***) can be uses for clear-fluids, or for mixed clear-fluid and porous-medium simulations when in case of strongly advective two-phase flow scenarios where turbulent mixing mat be neglected. In addition, as simple extension of the *porousKEpsilon* model has been implemented that allows to correct the turbulent intensity using a term that is proportional to the fraction of the other phase (see ***porousKEpsilon2PhaseCorrected.H***).

One can find a detailed, commented example for a porous one-phase simulation in the tutorial 3D_SmallESFR.

Click here

What’s inside: Documentation – turbulence



Include dependency graph for porousKEpsilon.H:

/builds/foam-for-nuclear /GeN-Foam/GeN-Foam/classes /thermalHydraulics/src/physicsModels /turbulenceModels/porousKEpsilon /porousKEpsilon.H

RASModel.H

eddyViscosity.H

This graph shows which files directly or indirectly include this file:

/builds/foam-for-nuclear /GeN-Foam/GeN-Foam/classes /thermalHydraulics/src/physicsModels /turbulenceModels/porousKEpsilon /porousKEpsilon.H

/builds/foam-for-nuclear /GeN-Foam/GeN-Foam/classes /thermalHydraulics/src/physicsModels /turbulenceModels/multiphaseCompressibleTurbulenceModels.C

/builds/foam-for-nuclear /GeN-Foam/GeN-Foam/classes /thermalHydraulics/src/physicsModels /turbulenceModels/porousKEpsilon /porousKEpsilon.C

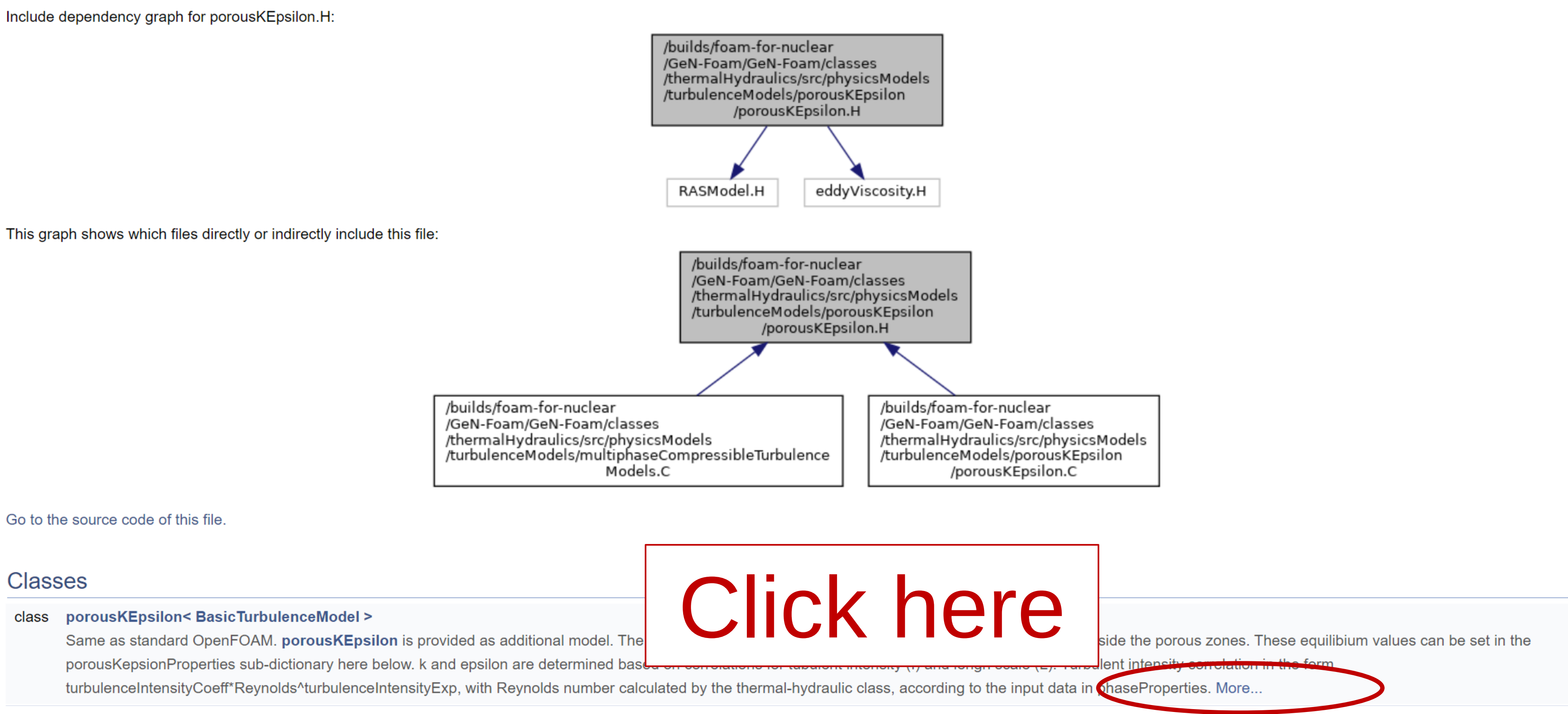
Go to the source code of this file.

Classes

class **porousKEpsilon< BasicTurbulenceModel >**

Same as standard OpenFOAM. **porousKEpsilon** is provided as additional model. The only difference is that it forces k and epsilon to equilibrium values inside the porous zones. These equilibium values can be set in the porousKepsionProperties sub-dictionary here below. k and epsilon are determined based on correlations for tubulent intensity (I) and lengh scale (L). Turbulent intensity correlation in the form turbulenceIntensityCoeff*Reynolds^turbulenceIntensityExp, with Reynolds number calculated by the thermal-hydraulic class, according to the input data in phaseProperties. More...

What's inside: Documentation – turbulence



Include dependency graph for porousKEpsilon.H:

/builds/foam-for-nuclear /GeN-Foam/GeN-Foam/classes /thermalHydraulics/src/physicsModels /turbulenceModels/porousKEpsilon /porousKEpsilon.H
RASModel.H
eddyViscosity.H

This graph shows which files directly or indirectly include this file:

/builds/foam-for-nuclear /GeN-Foam/GeN-Foam/classes /thermalHydraulics/src/physicsModels /turbulenceModels/porousKEpsilon /porousKEpsilon.H
/builds/foam-for-nuclear /GeN-Foam/GeN-Foam/classes /thermalHydraulics/src/physicsModels /turbulenceModels/multiphaseCompressibleTurbulence Models.C
/builds/foam-for-nuclear /GeN-Foam/GeN-Foam/classes /thermalHydraulics/src/physicsModels /turbulenceModels/porousKEpsilon /porousKEpsilon.C

Go to the source code of this file.

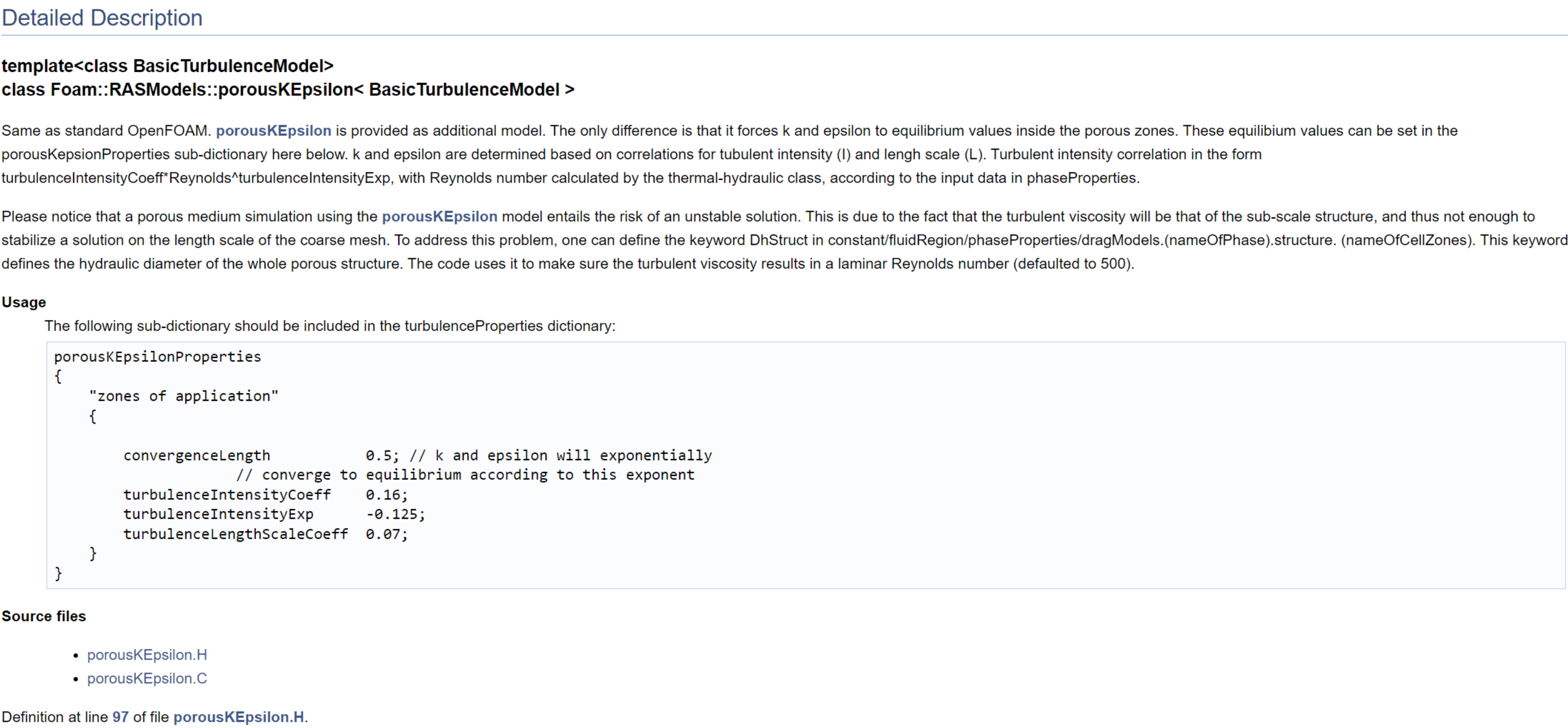
Classes

class **porousKEpsilon< BasicTurbulenceModel >**

Same as standard OpenFOAM. **porousKEpsilon** is provided as additional model. The side the porous zones. These equilibium values can be set in the porousKepsionProperties sub-dictionary here below. k and epsilon are determined based on correlations for turbulent intensity (I) and length scale (L). Turbulent intensity correlation in the form turbulenceIntensityCoeff*Reynolds^turbulenceIntensityExp, with Reynolds number calculated by the thermal-hydraulic class, according to the input data in phaseProperties. More...

Click here

What’s inside: Documentation – turbulence



Detailed Description

template<class BasicTurbulenceModel>
class Foam::RASModels::porousKEpsilon< BasicTurbulenceModel >

Same as standard OpenFOAM. porousKEpsilon is provided as additional model. The only difference is that it forces k and epsilon to equilibrium values inside the porous zones. These equilibium values can be set in the porousKepsionProperties sub-dictionary here below. k and epsilon are determined based on correlations for tubulent intensity (I) and lengh scale (L). Turbulent intensity correlation in the form turbulenceIntensityCoeff*Reynolds^turbulenceIntensityExp, with Reynolds number calculated by the thermal-hydraulic class, according to the input data in phaseProperties.

Please notice that a porous medium simulation using the porousKEpsilon model entails the risk of an unstable solution. This is due to the fact that the turbulent viscosity will be that of the sub-scale structure, and thus not enough to stabilize a solution on the length scale of the coarse mesh. To address this problem, one can define the keyword DhStruct in constant/fluidRegion/phaseProperties/dragModels.(nameOfPhase).structure. (nameOfCellZones). This keyword defines the hydraulic diameter of the whole porous structure. The code uses it to make sure the turbulent viscosity results in a laminar Reynolds number (defaulted to 500).

Usage

The following sub-dictionary should be included in the turbulenceProperties dictionary:

```
porousKEpsilonProperties
{
    "zones of application"
    {

        convergenceLength           0.5; // k and epsilon will exponentially
                     // converge to equilibrium according to this exponent
        turbulenceIntensityCoeff    0.16;
        turbulenceIntensityExp      -0.125;
        turbulenceLengthScaleCoeff  0.07;
    }
}
```

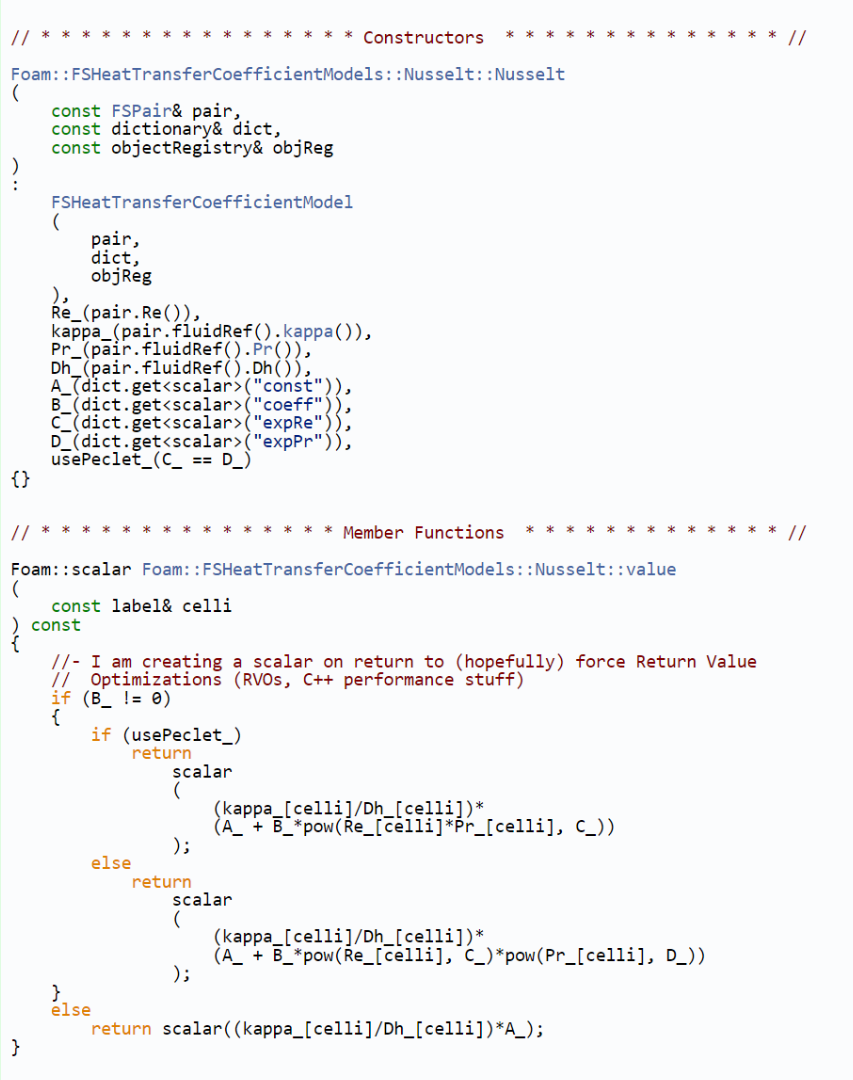
Source files

- porousKEpsilon.H
- porousKEpsilon.C

Definition at line 97 of file porousKEpsilon.H.

How to use the documentatio

- Embedded documentation still under construction
- But often the code itself is enough to understand



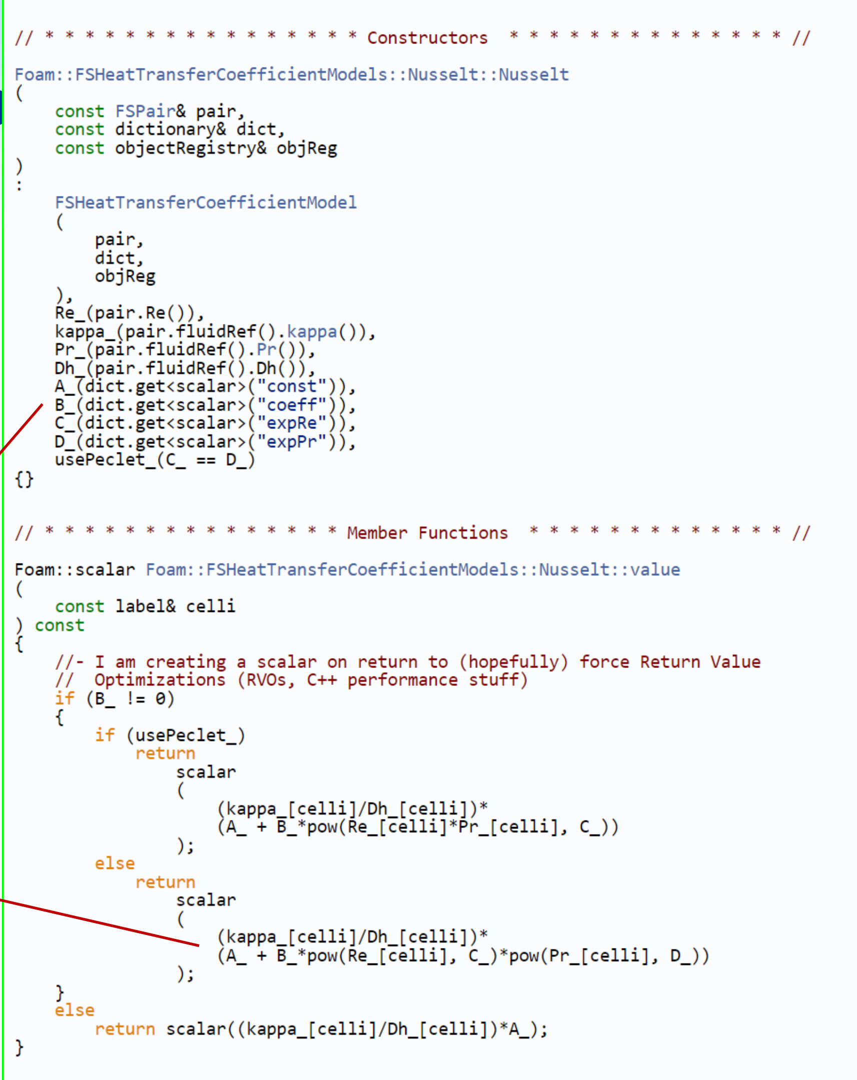
```
// * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * //

Foam::FSHeatTransferCoefficientModels::Nusselt::Nusselt
(
    const FSPair& pair,
    const dictionary& dict,
    const objectRegistry& objReg
)
:
    FSHeatTransferCoefficientModel
    (
        pair,
        dict,
        objReg
    ),
    Re_(pair.Re()),
    kappa_(pair.fluidRef().kappa()),
    Pr_(pair.fluidRef().Pr()),
    Dh_(pair.fluidRef().Dh()),
    A_(dict.get<scalar>("const")),
    B_(dict.get<scalar>("coeff")),
    C_(dict.get<scalar>("expRe")),
    D_(dict.get<scalar>("expPr")),
    usePeclet_(C_ == D_)
{}


// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * //

Foam::scalar Foam::FSHeatTransferCoefficientModels::Nusselt::value
(
    const label& celli
) const
{
    //- I am creating a scalar on return to (hopefully) force Return Value
    //  Optimizations (RVOs, C++ performance stuff)
    if (B_ != 0)
    {
        if (usePeclet_)
            return
                scalar
                (
                    (kappa_[celli]/Dh_[celli])*
                    (A_ + B_*pow(Re_[celli]*Pr_[celli], C_))
                );
        else
            return
                scalar
                (
                    (kappa_[celli]/Dh_[celli])*
                    (A_ + B_*pow(Re_[celli], C_)*pow(Pr_[celli], D_))
                );
    }
    else
        return scalar((kappa_[celli]/Dh_[celli])*A_);
}
```

How to use the documentatio

- Embedded documentation still under construction
- But often the code itself is enough to understand

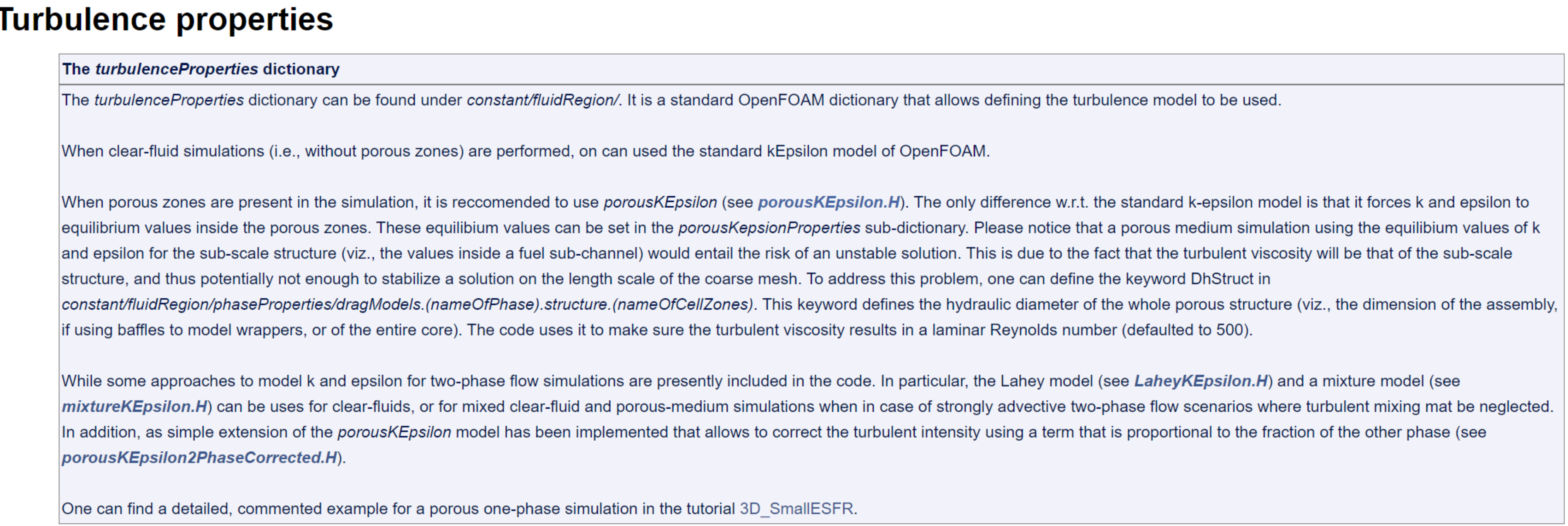
Get some constants from a dictionary

Correlation in the form:
Nu = A_+B_*(Re^C_)*(Pr^D_)



```
// * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * //

Foam::FSHeatTransferCoefficientModels::Nusselt::Nusselt
(
    const FSPair& pair,
    const dictionary& dict,
    const objectRegistry& objReg
)
:
    FSHeatTransferCoefficientModel
    (
        pair,
        dict,
        objReg
    ),
    Re_(pair.Re()),
    kappa_(pair.fluidRef().kappa()),
    Pr_(pair.fluidRef().Pr()),
    Dh_(pair.fluidRef().Dh()),
    A_(dict.get<scalar>("const")),
    B_(dict.get<scalar>("coeff")),
    C_(dict.get<scalar>("expRe")),
    D_(dict.get<scalar>("expPr")),
    usePeclet_(C_ == D_)
{}


// * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * //

Foam::scalar Foam::FSHeatTransferCoefficientModels::Nusselt::value
(
    const label& celli
) const
{
    //- I am creating a scalar on return to (hopefully) force Return Value
    //  Optimizations (RVOs, C++ performance stuff)
    if (B_ != 0)
    {
        if (usePeclet_)
            return
                scalar
                (
                    (kappa_[celli]/Dh_[celli])*
                    (A_ + B_*pow(Re_[celli]*Pr_[celli], C_))
                );
        else
            return
                scalar
                (
                    (kappa_[celli]/Dh_[celli])*
                    (A_ + B_*pow(Re_[celli], C_)*pow(Pr_[celli], D_))
                );
    }
    else
        return scalar((kappa_[celli]/Dh_[celli])*A_);
}
```

What’s inside: Documentation – turbulence



Turbulence properties

The *turbulenceProperties* dictionary

The *turbulenceProperties* dictionary can be found under *constant/fluidRegion/*. It is a standard OpenFOAM dictionary that allows defining the turbulence model to be used.

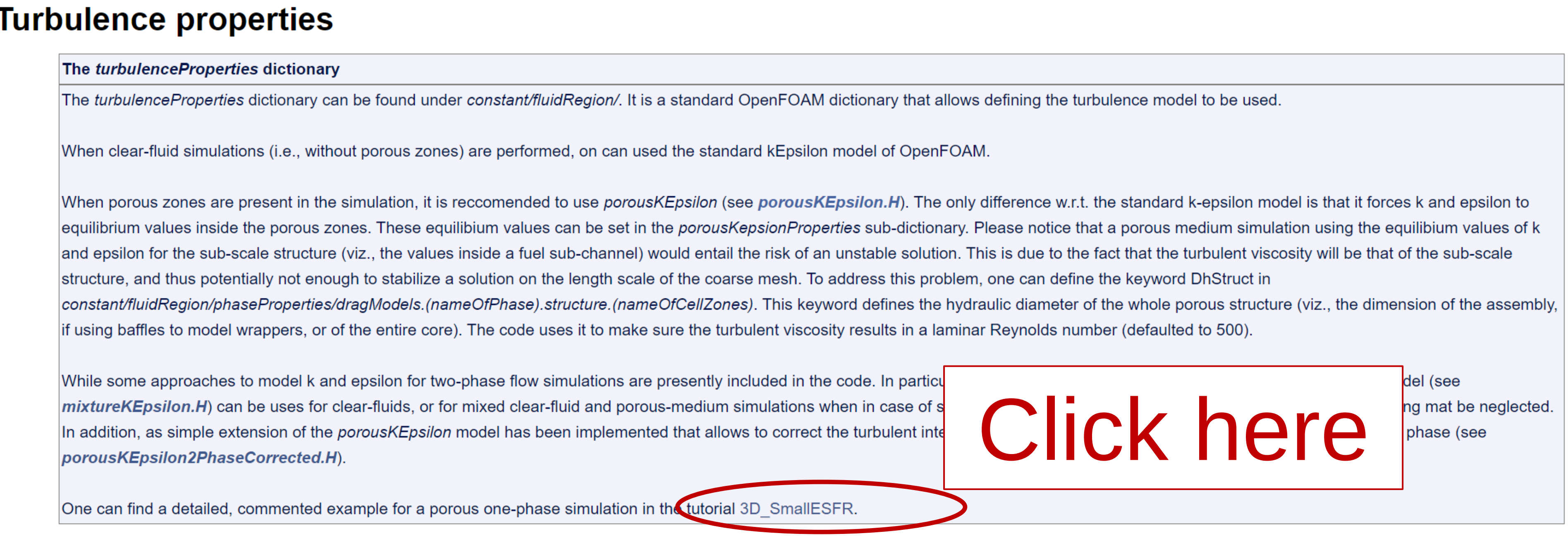
When clear-fluid simulations (i.e., without porous zones) are performed, on can used the standard kEpsilon model of OpenFOAM.

When porous zones are present in the simulation, it is reccomended to use *porousKEpsilon* (see ***porousKEpsilon.H***). The only difference w.r.t. the standard k-epsilon model is that it forces k and epsilon to equilibrium values inside the porous zones. These equilibium values can be set in the *porousKepsionProperties* sub-dictionary. Please notice that a porous medium simulation using the equilibium values of k and epsilon for the sub-scale structure (viz., the values inside a fuel sub-channel) would entail the risk of an unstable solution. This is due to the fact that the turbulent viscosity will be that of the sub-scale structure, and thus potentially not enough to stabilize a solution on the length scale of the coarse mesh. To address this problem, one can define the keyword DhStruct in *constant/fluidRegion/phaseProperties/dragModels.(nameOfPhase).structure.(nameOfCellZones)*. This keyword defines the hydraulic diameter of the whole porous structure (viz., the dimension of the assembly, if using baffles to model wrappers, or of the entire core). The code uses it to make sure the turbulent viscosity results in a laminar Reynolds number (defaulted to 500).

While some approaches to model k and epsilon for two-phase flow simulations are presently included in the code. In particular, the Lahey model (see ***LaheyKEpsilon.H***) and a mixture model (see ***mixtureKEpsilon.H***) can be uses for clear-fluids, or for mixed clear-fluid and porous-medium simulations when in case of strongly advective two-phase flow scenarios where turbulent mixing mat be neglected. In addition, as simple extension of the *porousKEpsilon* model has been implemented that allows to correct the turbulent intensity using a term that is proportional to the fraction of the other phase (see ***porousKEpsilon2PhaseCorrected.H***).

One can find a detailed, commented example for a porous one-phase simulation in the tutorial 3D_SmallESFR.

What's inside: Documentation – turbulence



Turbulence properties

The *turbulenceProperties* dictionary

The *turbulenceProperties* dictionary can be found under *constant/fluidRegion/*. It is a standard OpenFOAM dictionary that allows defining the turbulence model to be used.

When clear-fluid simulations (i.e., without porous zones) are performed, on can used the standard kEpsilon model of OpenFOAM.

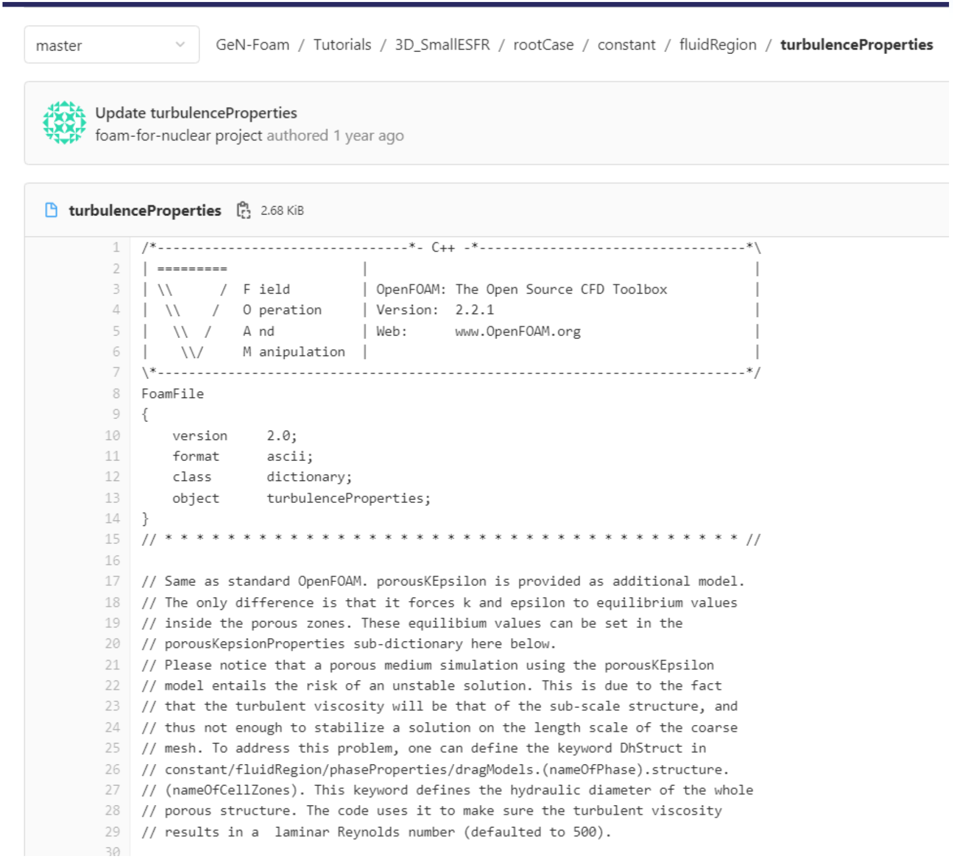
When porous zones are present in the simulation, it is reccomended to use *porousKEpsilon* (see ***porousKEpsilon.H***). The only difference w.r.t. the standard k-epsilon model is that it forces k and epsilon to equilibrium values inside the porous zones. These equilibium values can be set in the *porousKepsionProperties* sub-dictionary. Please notice that a porous medium simulation using the equilibium values of k and epsilon for the sub-scale structure (viz., the values inside a fuel sub-channel) would entail the risk of an unstable solution. This is due to the fact that the turbulent viscosity will be that of the sub-scale structure, and thus potentially not enough to stabilize a solution on the length scale of the coarse mesh. To address this problem, one can define the keyword DhStruct in *constant/fluidRegion/phaseProperties/dragModels.(nameOfPhase).structure.(nameOfCellZones)*. This keyword defines the hydraulic diameter of the whole porous structure (viz., the dimension of the assembly, if using baffles to model wrappers, or of the entire core). The code uses it to make sure the turbulent viscosity results in a laminar Reynolds number (defaulted to 500).

While some approaches to model k and epsilon for two-phase flow simulations are presently included in the code. In particu del (see ***mixtureKEpsilon.H***) can be uses for clear-fluids, or for mixed clear-fluid and porous-medium simulations when in case of s ng mat be neglected. In addition, as simple extension of the *porousKEpsilon* model has been implemented that allows to correct the turbulent inte phase (see ***porousKEpsilon2PhaseCorrected.H***).

One can find a detailed, commented example for a porous one-phase simulation in the tutorial 3D_SmallESFR.

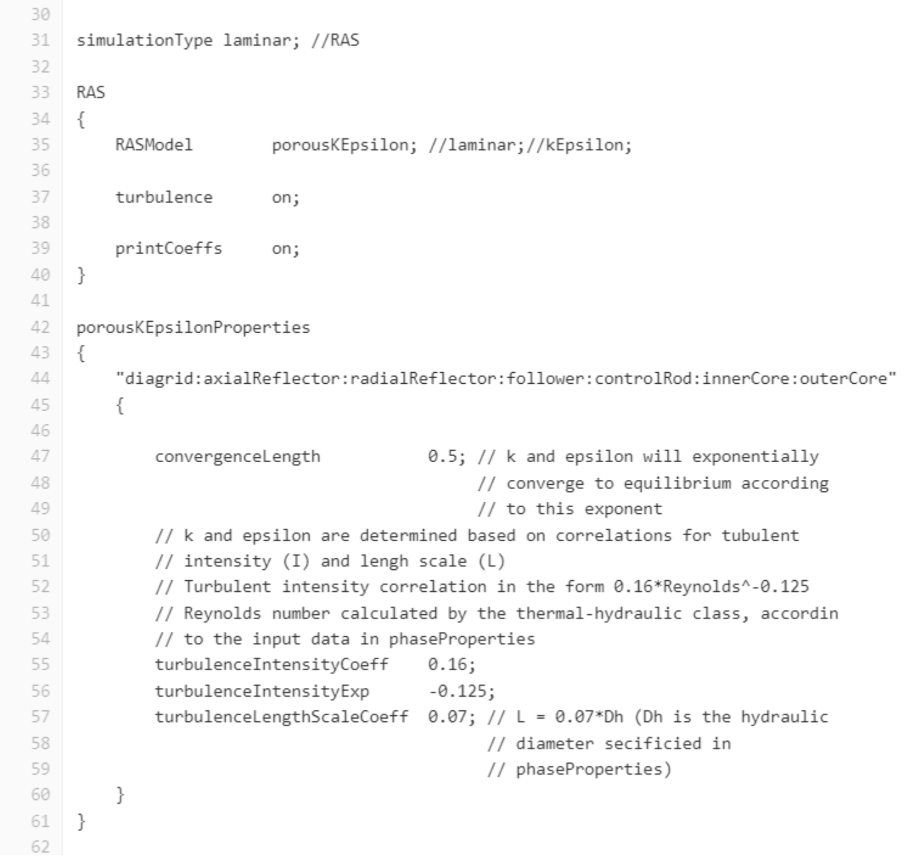
Click here

What’s inside: Documentation – turbulence



master
GeN-Foam / Tutorials / 3D_SmallESFR / rootCase / constant / fluidRegion / turbulenceProperties
Update turbulenceProperties
foam-for-nuclear project authored 1 year ago
turbulenceProperties
2.68 KiB

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Same as standard OpenFOAM. porousKEpsilon is provided as additional model.
// The only difference is that it forces k and epsilon to equilibrium values
// inside the porous zones. These equilibium values can be set in the
// porousKepsionProperties sub-dictionary here below.
// Please notice that a porous medium simulation using the porousKEpsilon
// model entails the risk of an unstable solution. This is due to the fact
// that the turbulent viscosity will be that of the sub-scale structure, and
// thus not enough to stabilize a solution on the length scale of the coarse
// mesh. To address this problem, one can define the keyword DhStruct in
// constant/fluidRegion/phaseProperties/dragModels.(nameOfPhase).structure.
// (nameOfCellZones). This keyword defines the hydraulic diameter of the whole
// porous structure. The code uses it to make sure the turbulent viscosity
// results in a  laminar Reynolds number (defaulted to 500).
```



```

simulationType laminar; //RAS

RAS
{
    RASModel        porousKEpsilon; //laminar;//kEpsilon;

    turbulence      on;

    printCoeffs     on;
}

porousKEpsilonProperties
{
    "diagrid:axialReflector:radialReflector:follower:controlRod:innerCore:outerCore"
    {

        convergenceLength           0.5; // k and epsilon will exponentially
                                         // converge to equilibrium according
                                         // to this exponent
        // k and epsilon are determined based on correlations for tubulent
        // intensity (I) and lengh scale (L)
        // Turbulent intensity correlation in the form 0.16*Reynolds^-0.125
        // Reynolds number calculated by the thermal-hydraulic class, accordin
        // to the input data in phaseProperties
        turbulenceIntensityCoeff    0.16;
        turbulenceIntensityExp      -0.125;
        turbulenceLengthScaleCoeff  0.07; // L = 0.07*Dh (Dh is the hydraulic
                                          // diameter secificied in
                                          // phaseProperties)
    }
}

```

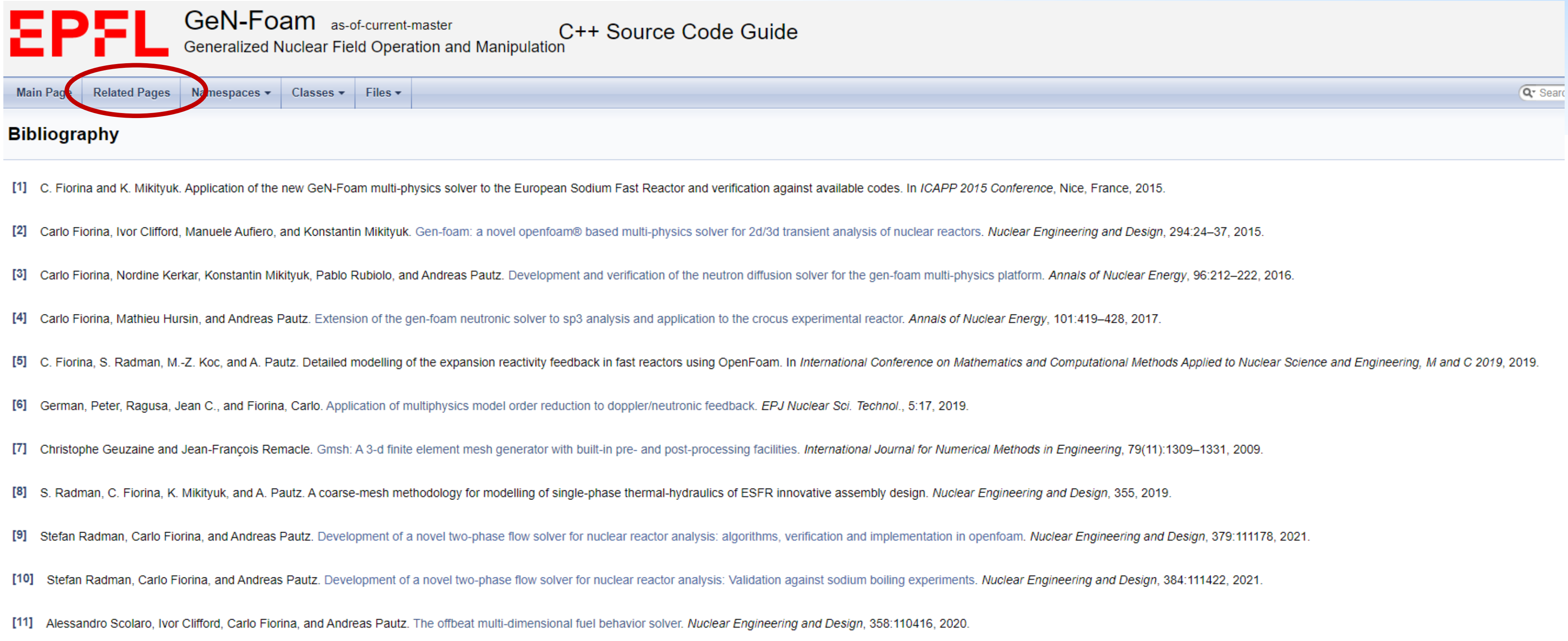
What's inside: Documentation - rationale

- Documentation designed to promote:
  - Understanding of the source code
  - Integration of code use and development

- Necessary conditions for a proficient use of GeN-Foam

- Second objective:
  - Limit inconsistencies between code and documentation

What’s inside: Documentation – references


EPFL
GeN-Foam as-of-current-master
Generalized Nuclear Field Operation and Manipulation
C++ Source Code Guide
Main Page
Related Pages
Namespaces
Classes
Files
Searc

Bibliography

[1] C. Fiorina and K. Mikityuk. Application of the new GeN-Foam multi-physics solver to the European Sodium Fast Reactor and verification against available codes. In *ICAPP 2015 Conference*, Nice, France, 2015.

[2] Carlo Fiorina, Ivor Clifford, Manuele Aufiero, and Konstantin Mikityuk. Gen-foam: a novel openfoam® based multi-physics solver for 2d/3d transient analysis of nuclear reactors. *Nuclear Engineering and Design*, 294:24–37, 2015.

[3] Carlo Fiorina, Nordine Kerkar, Konstantin Mikityuk, Pablo Rubiolo, and Andreas Pautz. Development and verification of the neutron diffusion solver for the gen-foam multi-physics platform. *Annals of Nuclear Energy*, 96:212–222, 2016.

[4] Carlo Fiorina, Mathieu Hursin, and Andreas Pautz. Extension of the gen-foam neutronic solver to sp3 analysis and application to the crocus experimental reactor. *Annals of Nuclear Energy*, 101:419–428, 2017.

[5] C. Fiorina, S. Radman, M.-Z. Koc, and A. Pautz. Detailed modelling of the expansion reactivity feedback in fast reactors using OpenFoam. In *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, M and C 2019*, 2019.

[6] German, Peter, Ragusa, Jean C., and Fiorina, Carlo. Application of multiphysics model order reduction to doppler/neutronic feedback. *EPJ Nuclear Sci. Technol.*, 5:17, 2019.

[7] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[8] S. Radman, C. Fiorina, K. Mikityuk, and A. Pautz. A coarse-mesh methodology for modelling of single-phase thermal-hydraulics of ESFR innovative assembly design. *Nuclear Engineering and Design*, 355, 2019.

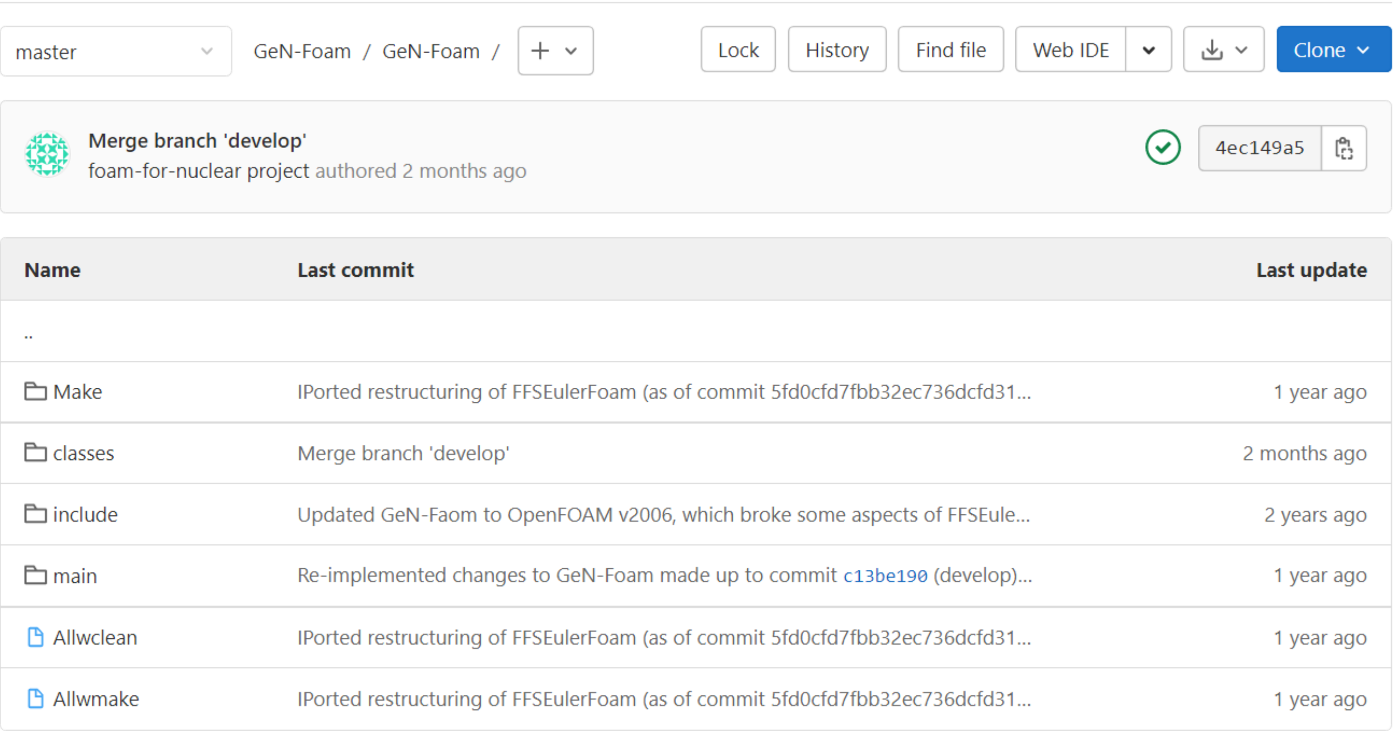
[9] Stefan Radman, Carlo Fiorina, and Andreas Pautz. Development of a novel two-phase flow solver for nuclear reactor analysis: algorithms, verification and implementation in openfoam. *Nuclear Engineering and Design*, 379:111178, 2021.

[10] Stefan Radman, Carlo Fiorina, and Andreas Pautz. Development of a novel two-phase flow solver for nuclear reactor analysis: Validation against sodium boiling experiments. *Nuclear Engineering and Design*, 384:111422, 2021.

[11] Alessandro Scolaro, Ivor Clifford, Carlo Fiorina, and Andreas Pautz. The offbeat multi-dimensional fuel behavior solver. *Nuclear Engineering and Design*, 358:110416, 2020.

https://foam-for-nuclear.gitlab.io/GeN-Foam/citelist.html

What’s inside: Source code

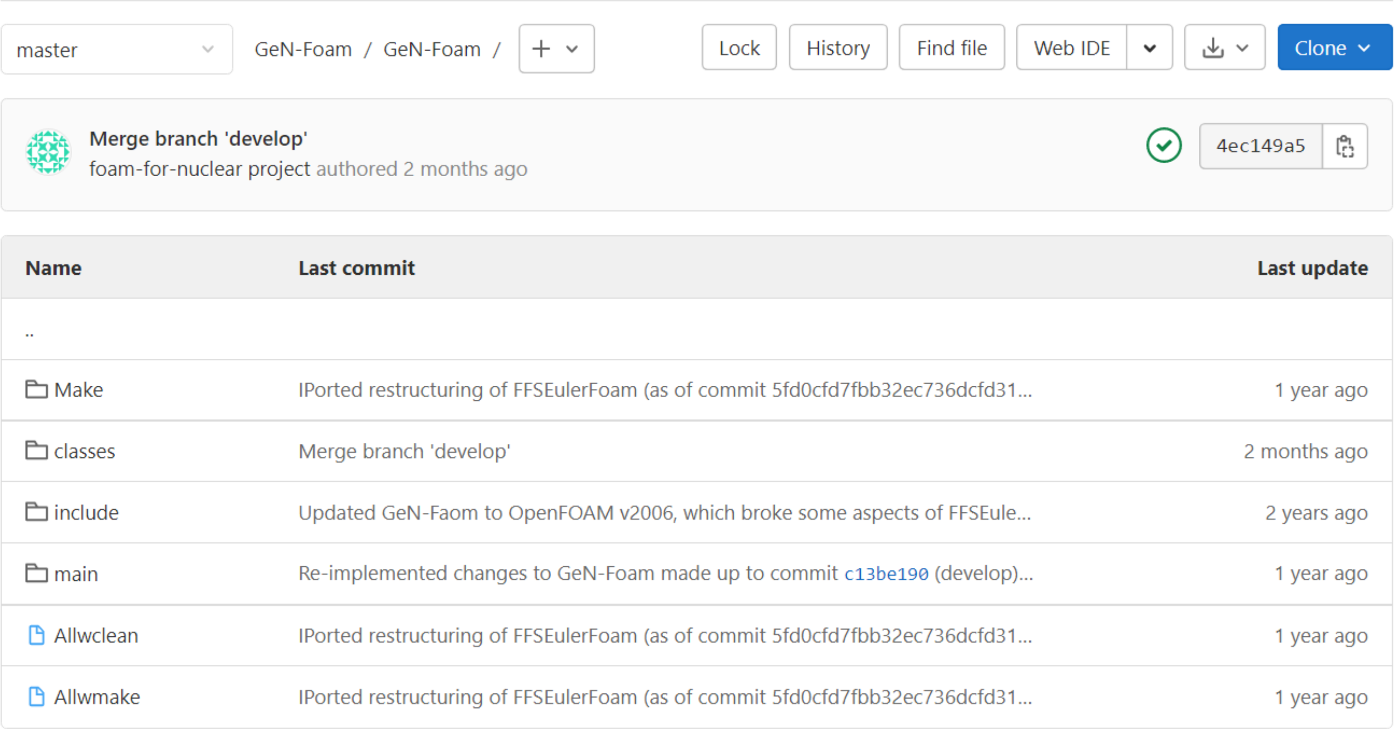


Merge branch 'develop'
foam-for-nuclear project authored 2 months ago

4ec149a5

Name	Last commit	Last update
..		
Make	IPorted restructuring of FFSEulerFoam (as of commit 5fd0cfd7fbb32ec736dcfd31...	1 year ago
classes	Merge branch 'develop'	2 months ago
include	Updated GeN-Faom to OpenFOAM v2006, which broke some aspects of FFSEule...	2 years ago
main	Re-implemented changes to GeN-Foam made up to commit c13be190 (develop)...	1 year ago
Allwclean	IPorted restructuring of FFSEulerFoam (as of commit 5fd0cfd7fbb32ec736dcfd31...	1 year ago
Allwmake	IPorted restructuring of FFSEulerFoam (as of commit 5fd0cfd7fbb32ec736dcfd31...	1 year ago

What’s inside: Source code – how to use it



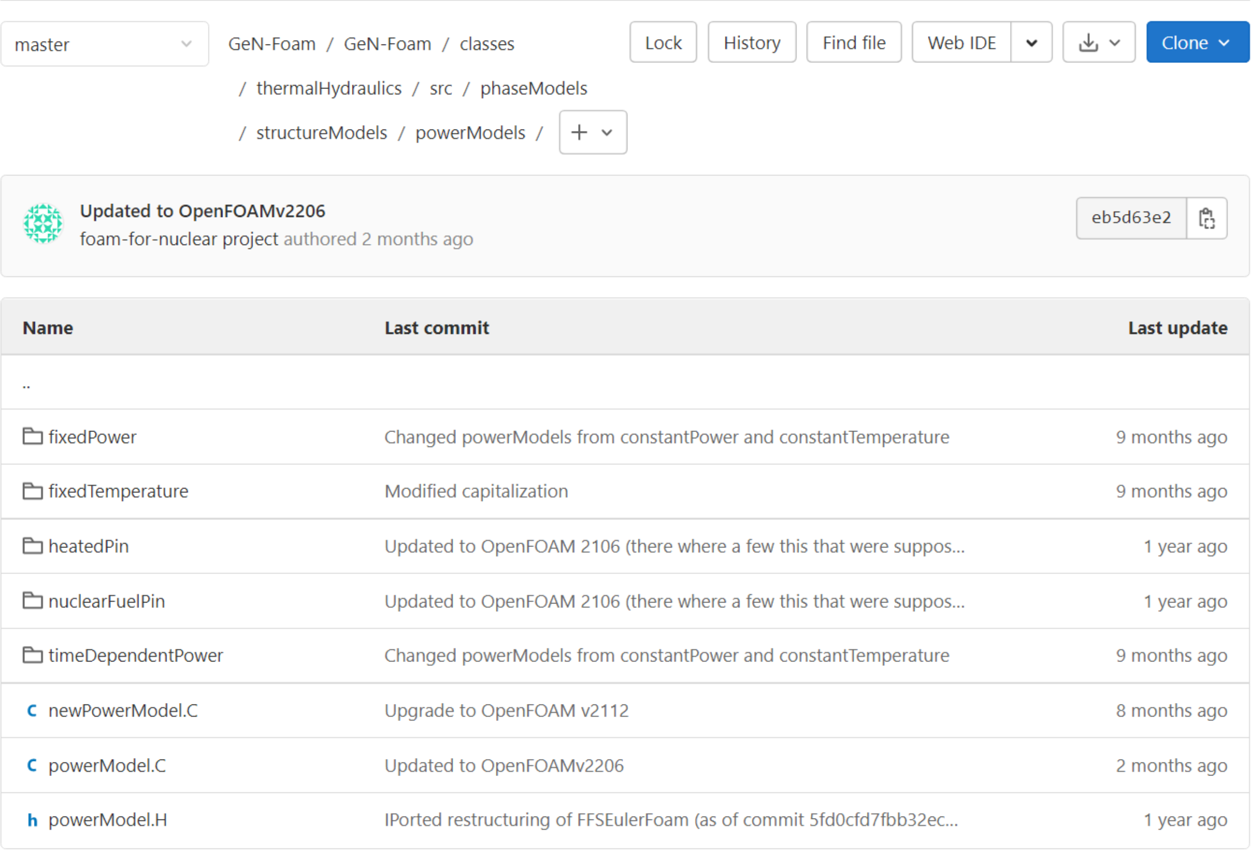
Merge branch 'develop'
foam-for-nuclear project authored 2 months ago
4ec149a5

Name	Last commit	Last update
..		
Make	IPorted restructuring of FFSEulerFoam (as of commit 5fd0cfd7fbb32ec736dcfd31...	1 year ago
classes	Merge branch 'develop'	2 months ago
include	Updated GeN-Faom to OpenFOAM v2006, which broke some aspects of FFSEule...	2 years ago
main	Re-implemented changes to GeN-Foam made up to commit c13be190 (develop)...	1 year ago
Allwclean	IPorted restructuring of FFSEulerFoam (as of commit 5fd0cfd7fbb32ec736dcfd31...	1 year ago
Allwmake	IPorted restructuring of FFSEulerFoam (as of commit 5fd0cfd7fbb32ec736dcfd31...	1 year ago

- “Classes” contains all the physics and multi-physics controls
- “main” contains what glues them together
- “include” are folders that mainly contain chunks of code that are included (#include) in the code (this is done only to avoid very long .C files)

What’s inside: Source code – power models

- Typical run-time selectable class
  - A parent class (powerModel)
  - A selector (newPowerModel)
  - Various derived classes that represent the run-time selectable models in OpenFOAM



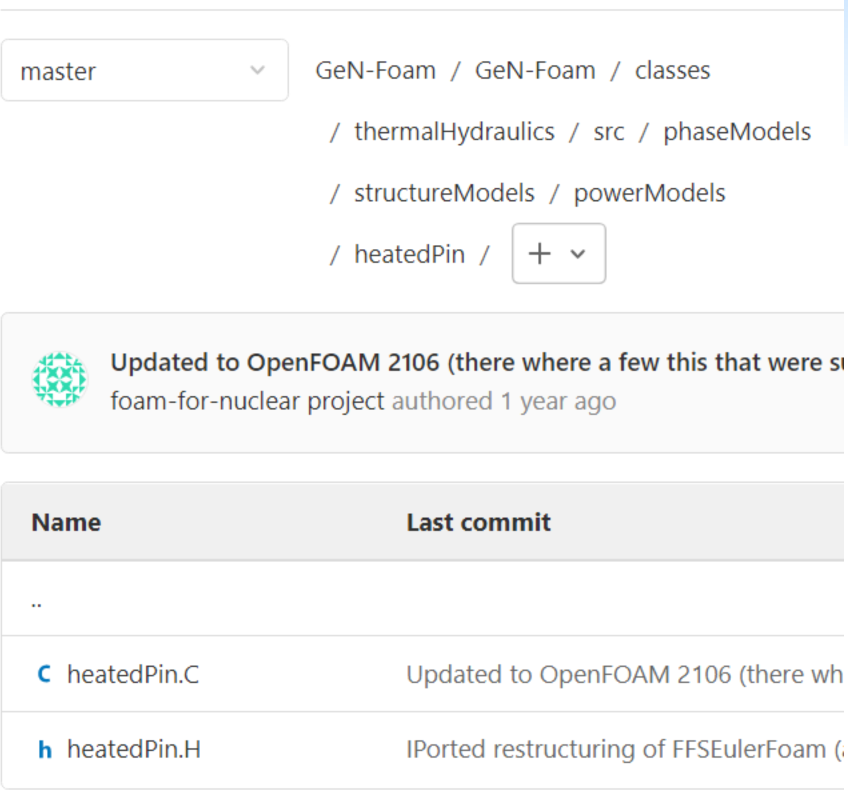
GeN-Foam / GeN-Foam / classes / thermalHydraulics / src / phaseModels / structureModels / powerModels /

Updated to OpenFOAMv2206
foam-for-nuclear project authored 2 months ago
eb5d63e2

Name	Last commit	Last update
..		
fixedPower	Changed powerModels from constantPower and constantTemperature	9 months ago
fixedTemperature	Modified capitalization	9 months ago
heatedPin	Updated to OpenFOAM 2106 (there where a few this that were suppos...	1 year ago
nuclearFuelPin	Updated to OpenFOAM 2106 (there where a few this that were suppos...	1 year ago
timeDependentPower	Changed powerModels from constantPower and constantTemperature	9 months ago
newPowerModel.C	Upgrade to OpenFOAM v2112	8 months ago
powerModel.C	Updated to OpenFOAMv2206	2 months ago
powerModel.H	IPorted restructuring of FFSEulerFoam (as of commit 5fd0cfd7fbb32ec...	1 year ago

What’s inside: Source code – power models

- Typical C++
  - Header (.H) file with declaration of members, and description of class
  - .C file that contain what the class really does



master

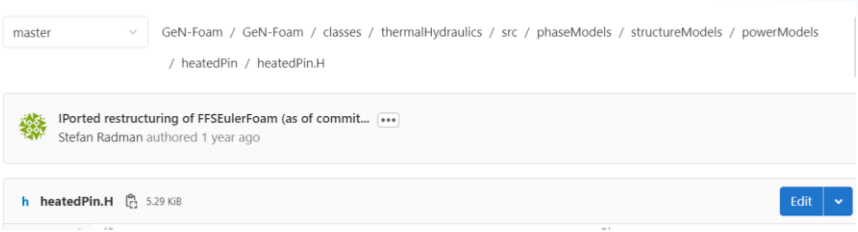
GeN-Foam / GeN-Foam / classes / thermalHydraulics / src / phaseModels / structureModels / powerModels / heatedPin / +

Updated to OpenFOAM 2106 (there where a few this that were s

foam-for-nuclear project authored 1 year ago

Name	Last commit
..	
C heatedPin.C	Updated to OpenFOAM 2106 (there wh
h heatedPin.H	IPorted restructuring of FFSEulerFoam (

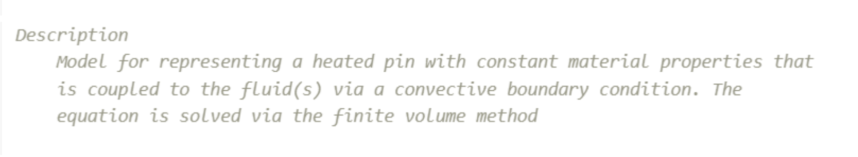
What’s inside: Source code – power models



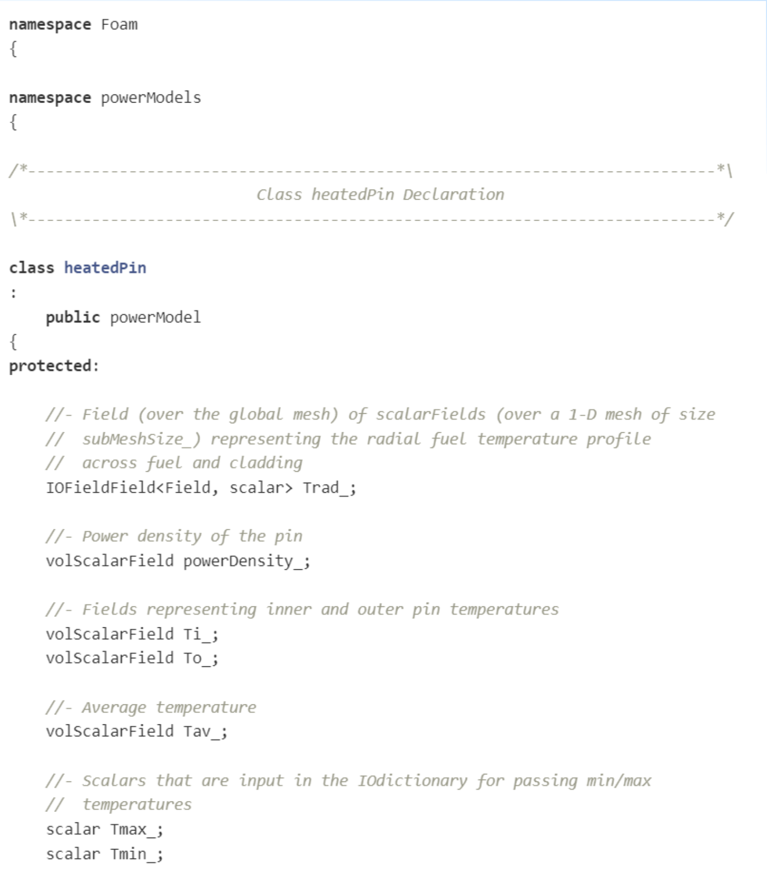
GeN-Foam / GeN-Foam / classes / thermalHydraulics / src / phaseModels / structureModels / powerModels / heatedPin / heatedPin.H

IPorted restructuring of FFSEulerFoam (as of commit... •••
Stefan Radman authored 1 year ago

heatedPin.H 5.29 KiB

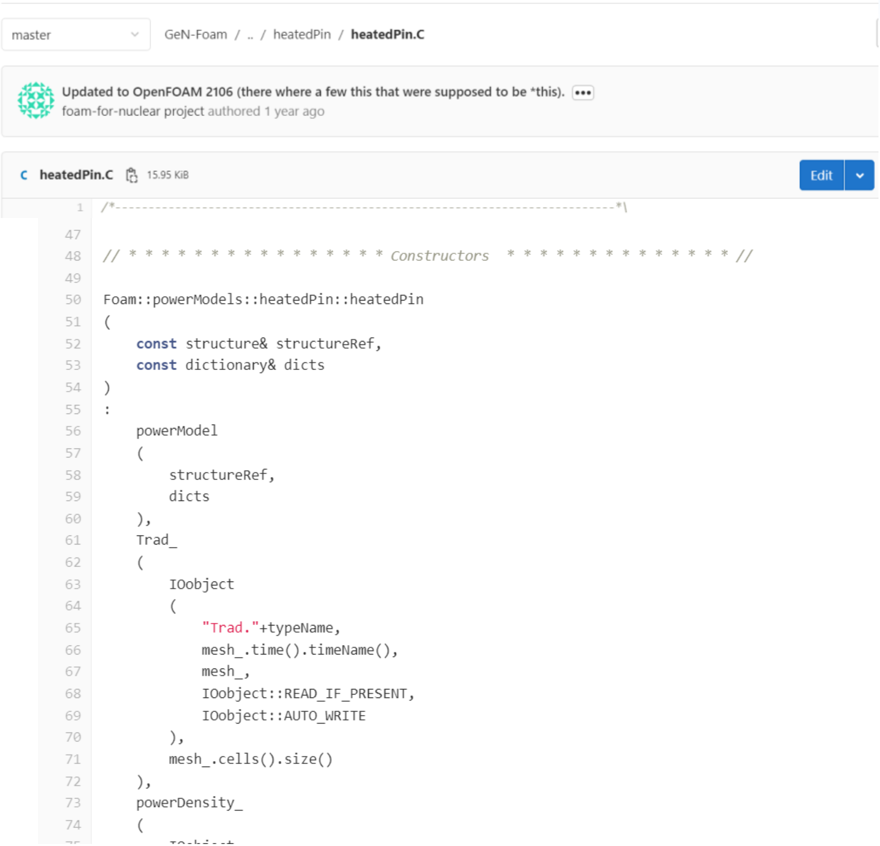


```
Description
    Model for representing a heated pin with constant material properties that
    is coupled to the fluid(s) via a convective boundary condition. The
    equation is solved via the finite volume method
```



```
namespace Foam
{

namespace powerModels
{

/*---------------------------------------------------------------------------*\
                        Class heatedPin Declaration
\*---------------------------------------------------------------------------*/

class heatedPin
:
    public powerModel
{
protected:

    //- Field (over the global mesh) of scalarFields (over a 1-D mesh of size
    //  subMeshSize_) representing the radial fuel temperature profile
    //  across fuel and cladding
    IOFieldField<Field, scalar> Trad_;

    //- Power density of the pin
    volScalarField powerDensity_;

    //- Fields representing inner and outer pin temperatures
    volScalarField Ti_;
    volScalarField To_;

    //- Average temperature
    volScalarField Tav_;

    //- Scalars that are input in the IOdictionary for passing min/max
    //  temperatures
    scalar Tmax_;
    scalar Tmin_;
```

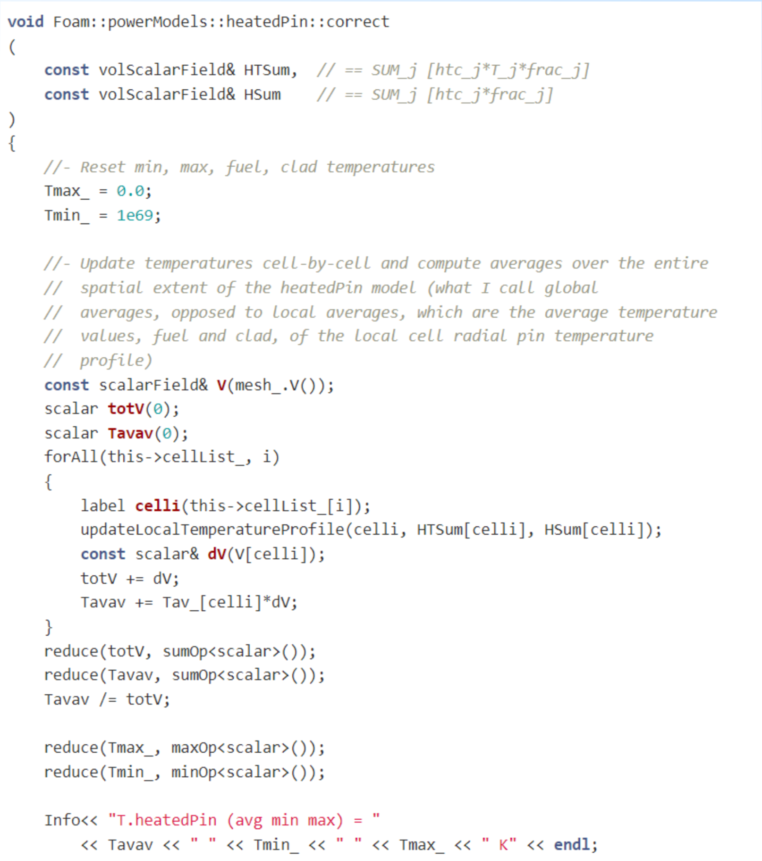
What’s inside: Source code – power models



Updated to OpenFOAM 2106 (there where a few this that were supposed to be *this).
foam-for-nuclear project authored 1 year ago

heatedPin.C 15.95 KiB

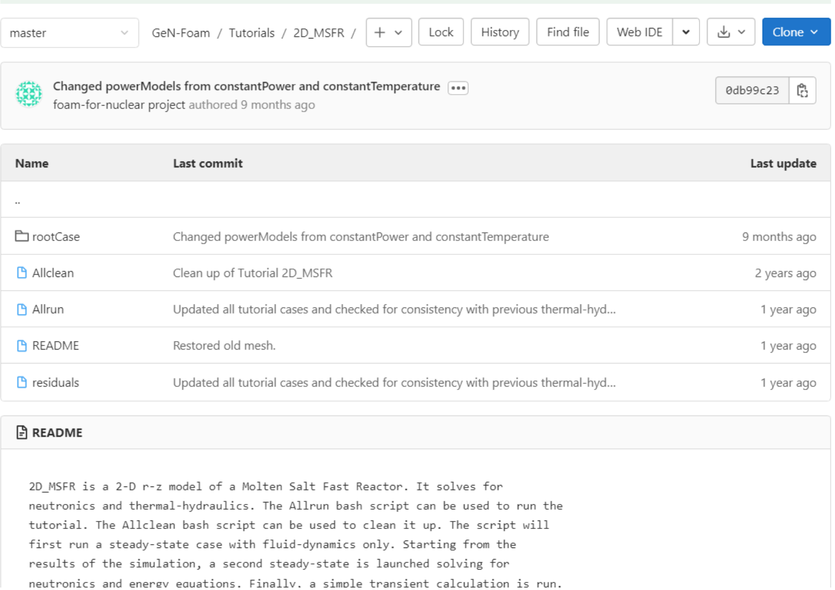
```
/*---------------------------------------------------------------------------*\

// * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * //

Foam::powerModels::heatedPin::heatedPin
(
    const structure& structureRef,
    const dictionary& dicts
)
:
    powerModel
    (
        structureRef,
        dicts
    ),
    Trad_
    (
        IOobject
        (
            "Trad."+typeName,
            mesh_.time().timeName(),
            mesh_,
            IOobject::READ_IF_PRESENT,
            IOobject::AUTO_WRITE
        ),
        mesh_.cells().size()
    ),
    powerDensity_
    (
        IOobject
```



```
void Foam::powerModels::heatedPin::correct
(
    const volScalarField& HTSum,  // == SUM_j [htc_j*T_j*frac_j]
    const volScalarField& HSum    // == SUM_j [htc_j*frac_j]
)
{
    //- Reset min, max, fuel, clad temperatures
    Tmax_ = 0.0;
    Tmin_ = 1e69;

    //- Update temperatures cell-by-cell and compute averages over the entire
    //  spatial extent of the heatedPin model (what I call global
    //  averages, opposed to local averages, which are the average temperature
    //  values, fuel and clad, of the local cell radial pin temperature
    //  profile)
    const scalarField& V(mesh_.V());
    scalar totV(0);
    scalar Tavav(0);
    forAll(this->cellList_, i)
    {
        label celli(this->cellList_[i]);
        updateLocalTemperatureProfile(celli, HTSum[celli], HSum[celli]);
        const scalar& dV(V[celli]);
        totV += dV;
        Tavav += Tav_[celli]*dV;
    }
    reduce(totV, sumOp<scalar>());
    reduce(Tavav, sumOp<scalar>());
    Tavav /= totV;

    reduce(Tmax_, maxOp<scalar>());
    reduce(Tmin_, minOp<scalar>());

    Info<< "T.heatedPin (avg min max) = "
        << Tavav << " " << Tmin_ << " " << Tmax_ << " K" << endl;
```

What’s inside: Tutorials

- Cover essentially all functionalities of GeN-Foam
- They include a README file, an Allrun file (sometimes Allrun_parallel), an Allclean file, and some extensively commented inputs

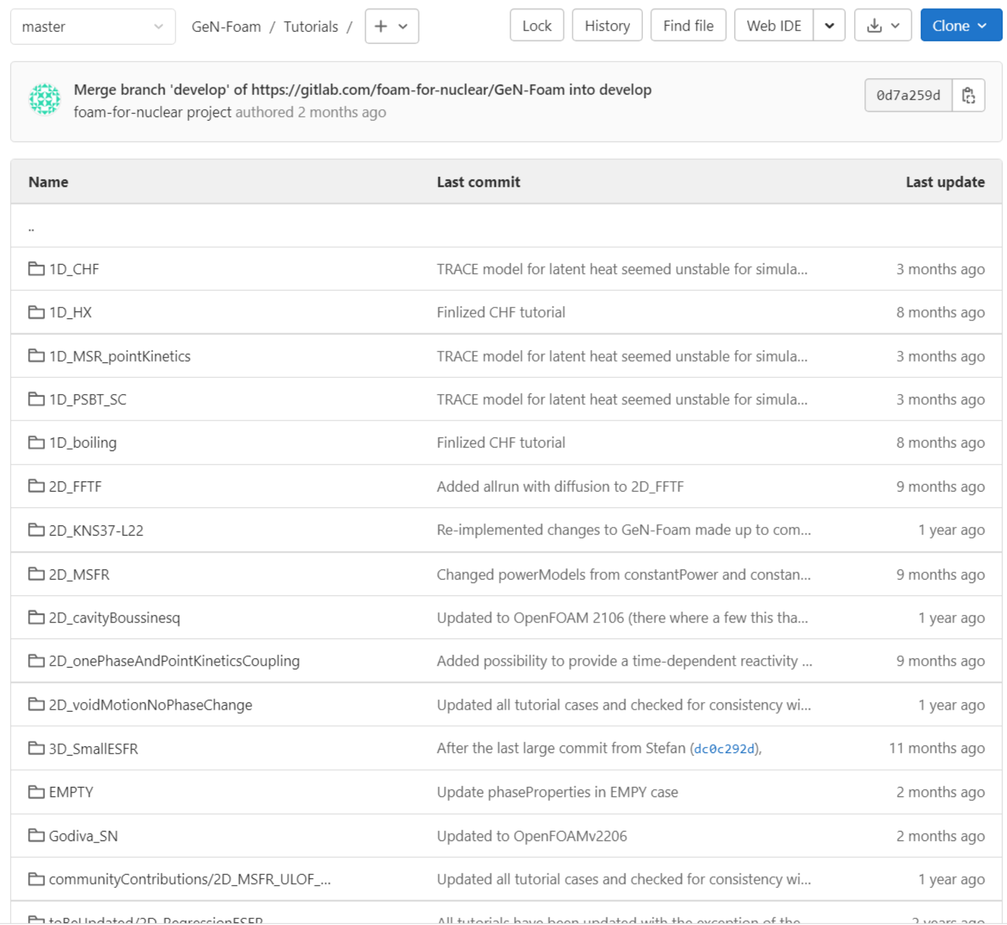


Changed powerModels from constantPower and constantTemperature •••
foam-for-nuclear project authored 9 months ago
0db99c23

Name	Last commit	Last update
..		
rootCase	Changed powerModels from constantPower and constantTemperature	9 months ago
Allclean	Clean up of Tutorial 2D_MSFR	2 years ago
Allrun	Updated all tutorial cases and checked for consistency with previous thermal-hyd...	1 year ago
README	Restored old mesh.	1 year ago
residuals	Updated all tutorial cases and checked for consistency with previous thermal-hyd...	1 year ago

README

```
2D_MSFR is a 2-D r-z model of a Molten Salt Fast Reactor. It solves for
neutronics and thermal-hydraulics. The Allrun bash script can be used to run the
tutorial. The Allclean bash script can be used to clean it up. The script will
first run a steady-state case with fluid-dynamics only. Starting from the
results of the simulation, a second steady-state is launched solving for
neutronics and energy equations. Finally, a simple transient calculation is run.
```

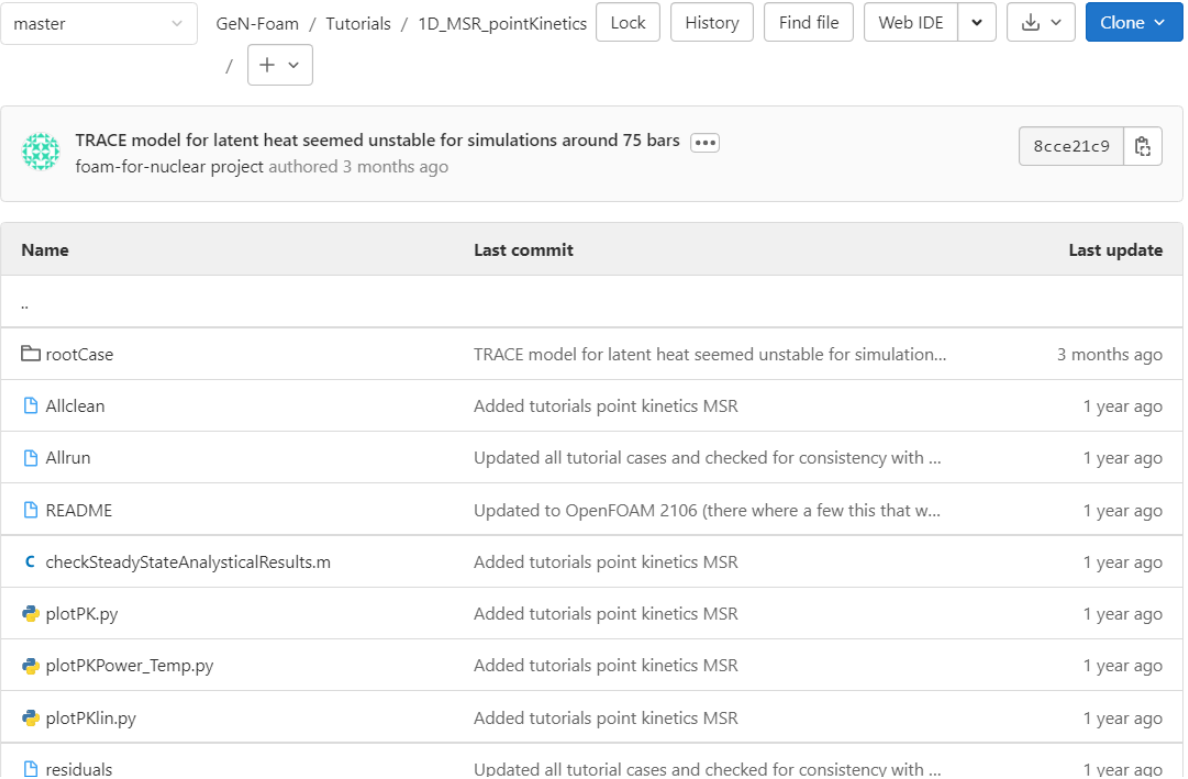


Merge branch 'develop' of https://gitlab.com/foam-for-nuclear/GeN-Foam into develop
foam-for-nuclear project authored 2 months ago
0d7a259d

Name	Last commit	Last update
..		
1D_CHF	TRACE model for latent heat seemed unstable for simula...	3 months ago
1D_HX	Finlized CHF tutorial	8 months ago
1D_MSR_pointKinetics	TRACE model for latent heat seemed unstable for simula...	3 months ago
1D_PSBT_SC	TRACE model for latent heat seemed unstable for simula...	3 months ago
1D_boiling	Finlized CHF tutorial	8 months ago
2D_FFTF	Added allrun with diffusion to 2D_FFTF	9 months ago
2D_KNS37-L22	Re-implemented changes to GeN-Foam made up to com...	1 year ago
2D_MSFR	Changed powerModels from constantPower and constan...	9 months ago
2D_cavityBoussinesq	Updated to OpenFOAM 2106 (there where a few this tha...	1 year ago
2D_onePhaseAndPointKineticsCoupling	Added possibility to provide a time-dependent reactivity ...	9 months ago
2D_voidMotionNoPhaseChange	Updated all tutorial cases and checked for consistency wi...	1 year ago
3D_SmallESFR	After the last large commit from Stefan (dc0c292d),	11 months ago
EMPTY	Update phaseProperties in EMPY case	2 months ago
Godiva_SN	Updated to OpenFOAMv2206	2 months ago
communityContributions/2D_MSFR_ULOF_...	Updated all tutorial cases and checked for consistency wi...	1 year ago
toBeUpdated/2D_RegressionESFR	All tutorials have been updated with the exception of the...	2 years ago

What’s inside: Tutorials

- Understanding the tutorial:
  1. README file
  2. Case folder
  3. Allrun file
  4. Run it and use paraview to see what happens
- N.B.
  - Very often we launch multiple simulations in the same tutorial
  - When that is the case, the case folder will contain a rootCase folder that will be duplicated multiple times



TRACE model for latent heat seemed unstable for simulations around 75 bars

foam-for-nuclear project authored 3 months ago

8cce21c9

Name	Last commit	Last update
..		
rootCase	TRACE model for latent heat seemed unstable for simulation...	3 months ago
Allclean	Added tutorials point kinetics MSR	1 year ago
Allrun	Updated all tutorial cases and checked for consistency with ...	1 year ago
README	Updated to OpenFOAM 2106 (there where a few this that w...	1 year ago
checkSteadyStateAnalyticalResults.m	Added tutorials point kinetics MSR	1 year ago
plotPK.py	Added tutorials point kinetics MSR	1 year ago
plotPKPower_Temp.py	Added tutorials point kinetics MSR	1 year ago
plotPKlin.py	Added tutorials point kinetics MSR	1 year ago
residuals	Updated all tutorial cases and checked for consistency with ...	1 year ago

What’s inside: Tutorials - 1D_MSR_pointKinetics

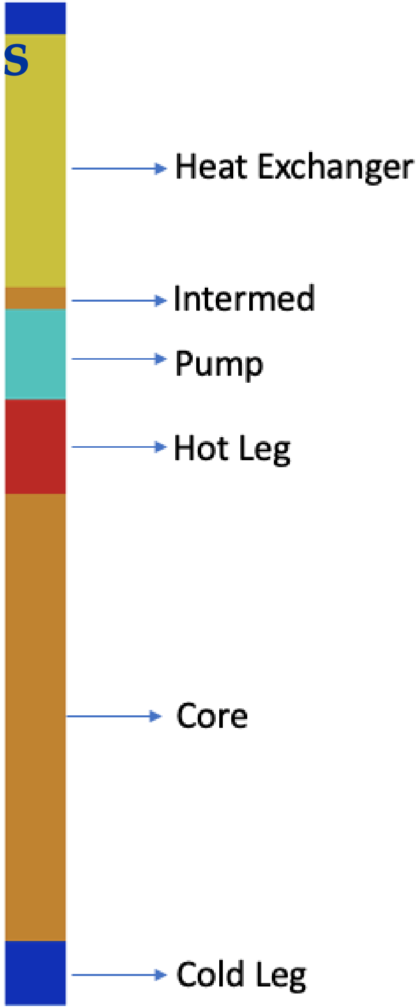
- Start from the README file

DESCRIPTION

This tutorial displays how to use the point kinetics module of GeN-Foam for MSRs. It is a simple 1-D case with core, hot leg, pump, heat exchanger and cold leg. The geometry is one dimensional and salt recirculation is simulated by making use of a cyclic boundary condition between top and bottom boundaries.

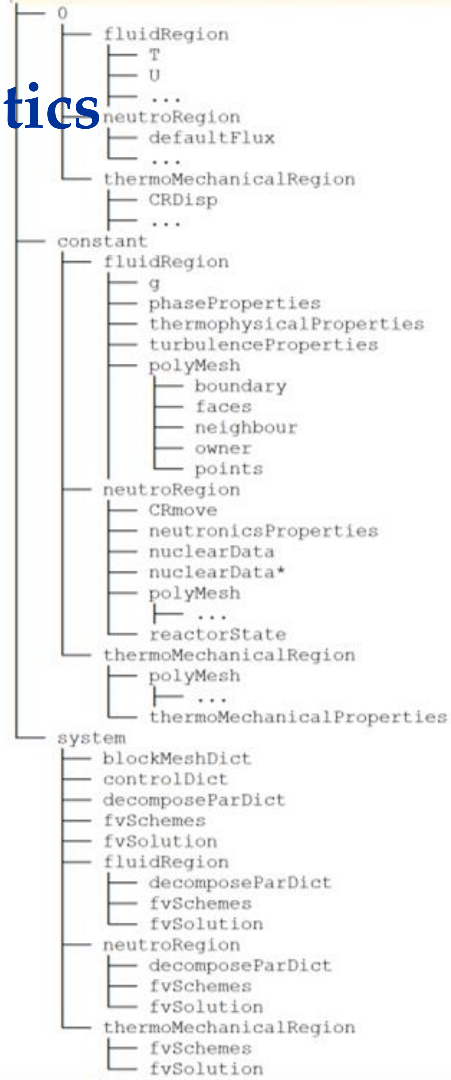
Three simulations are performed:

- energy and fluid dynamics to obtain a steady state
- energy, fluid dynamics and point kinetics to simulate a loss-of-flow
- recalculate the reactivity loss due to recirculation of the delayed neutron precursors.



What’s inside: Tutorials - 1D_MSR_pointKinetics

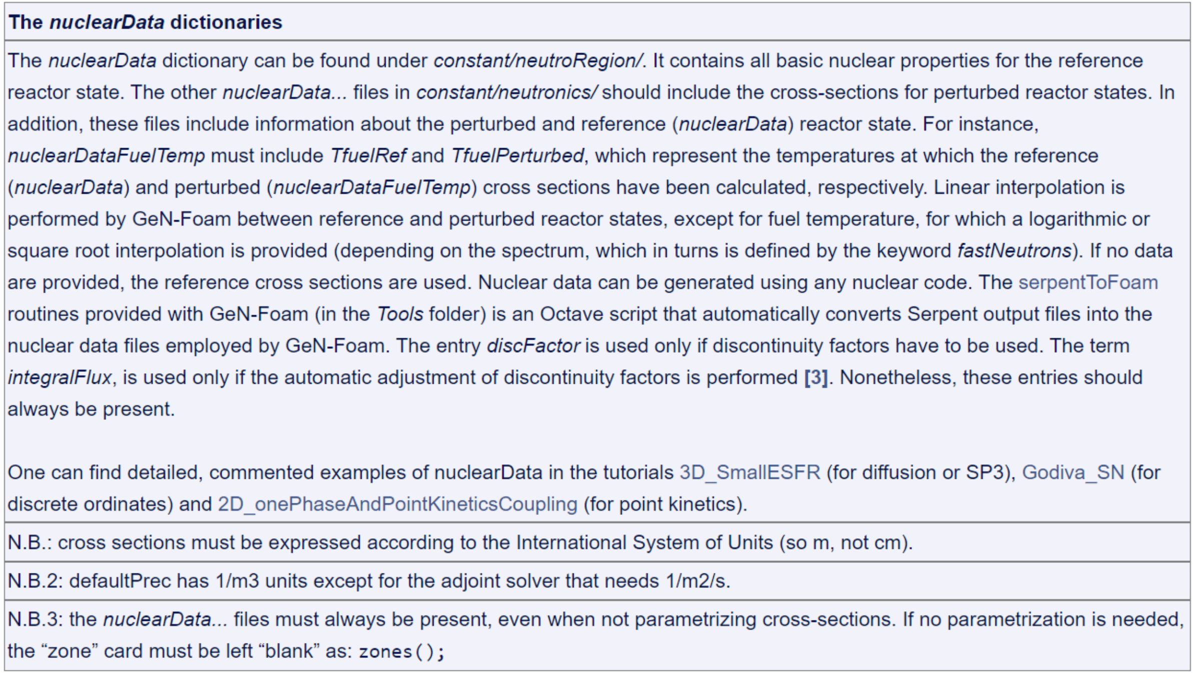
- Look at the case folder
  - 0 folder with three sub-folders containing the fields for each physics
  - constant folder with 3 sub-folders
    - 3 meshes (polyMesh folders)
    - 3 sets of dictionaries
  - system folder with:
    - 3 sub-folders with dedicated fvScheme and fvSolution for each physics
    - 1 controlDict
    - 1 common fvSolution with some multi-physics controls



```
├── 0
│   ├── fluidRegion
│   │   ├── T
│   │   ├── U
│   │   └── ...
│   ├── neutroRegion
│   │   ├── defaultFlux
│   │   └── ...
│   └── thermoMechanicalRegion
│       ├── CRDisp
│       └── ...
├── constant
│   ├── fluidRegion
│   │   ├── g
│   │   ├── phaseProperties
│   │   ├── thermophysicalProperties
│   │   ├── turbulenceProperties
│   │   └── polyMesh
│   │       ├── boundary
│   │       ├── faces
│   │       ├── neighbour
│   │       ├── owner
│   │       └── points
│   ├── neutroRegion
│   │   ├── CRmove
│   │   ├── neutronicsProperties
│   │   ├── nuclearData
│   │   ├── nuclearData*
│   │   ├── polyMesh
│   │   │   ├── ...
│   │   └── reactorState
│   └── thermoMechanicalRegion
│       ├── polyMesh
│       │   ├── ...
│       └── thermoMechanicalProperties
└── system
    ├── blockMeshDict
    ├── controlDict
    ├── decomposeParDict
    ├── fvSchemes
    ├── fvSolution
    ├── fluidRegion
    │   ├── decomposeParDict
    │   ├── fvSchemes
    │   └── fvSolution
    ├── neutroRegion
    │   ├── decomposeParDict
    │   ├── fvSchemes
    │   └── fvSolution
    └── thermoMechanicalRegion
        ├── fvSchemes
        └── fvSolution
```

What’s inside: Tutorials - 1D_MSR_pointKinetics

- Look at the dictionaries
  - All the dictionaries are explained in the user manual



The *nuclearData* dictionaries

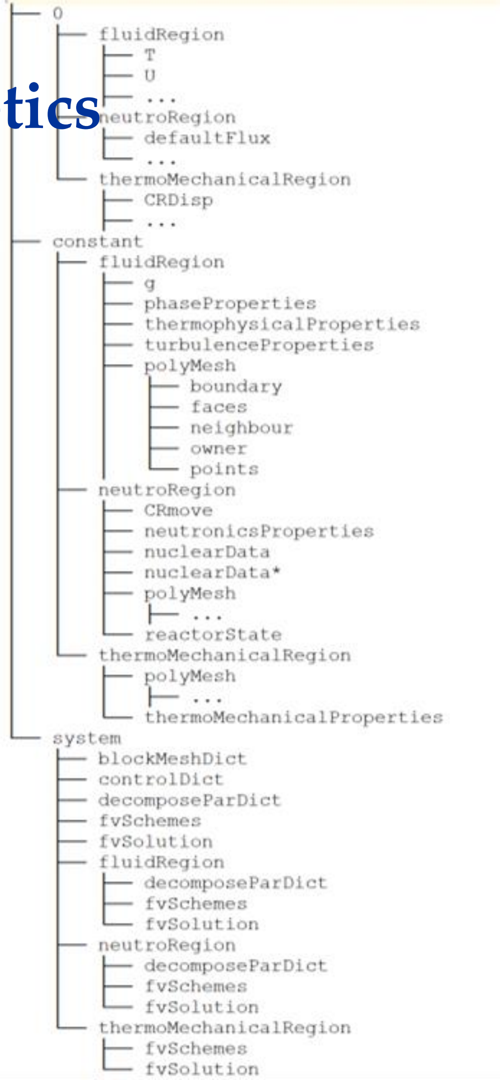
The *nuclearData* dictionary can be found under *constant/neutroRegion/*. It contains all basic nuclear properties for the reference reactor state. The other *nuclearData...* files in *constant/neutronics/* should include the cross-sections for perturbed reactor states. In addition, these files include information about the perturbed and reference (*nuclearData*) reactor state. For instance, *nuclearDataFuelTemp* must include *TfuelRef* and *TfuelPerturbed*, which represent the temperatures at which the reference (*nuclearData*) and perturbed (*nuclearDataFuelTemp*) cross sections have been calculated, respectively. Linear interpolation is performed by GeN-Foam between reference and perturbed reactor states, except for fuel temperature, for which a logarithmic or square root interpolation is provided (depending on the spectrum, which in turns is defined by the keyword *fastNeutrons*). If no data are provided, the reference cross sections are used. Nuclear data can be generated using any nuclear code. The serpentToFoam routines provided with GeN-Foam (in the *Tools* folder) is an Octave script that automatically converts Serpent output files into the nuclear data files employed by GeN-Foam. The entry *discFactor* is used only if discontinuity factors have to be used. The term *integralFlux*, is used only if the automatic adjustment of discontinuity factors is performed [3]. Nonetheless, these entries should always be present.

One can find detailed, commented examples of nuclearData in the tutorials 3D_SmallESFR (for diffusion or SP3), Godiva_SN (for discrete ordinates) and 2D_onePhaseAndPointKineticsCoupling (for point kinetics).

N.B.: cross sections must be expressed according to the International System of Units (so m, not cm).

N.B.2: defaultPrec has 1/m3 units except for the adjoint solver that needs 1/m2/s.

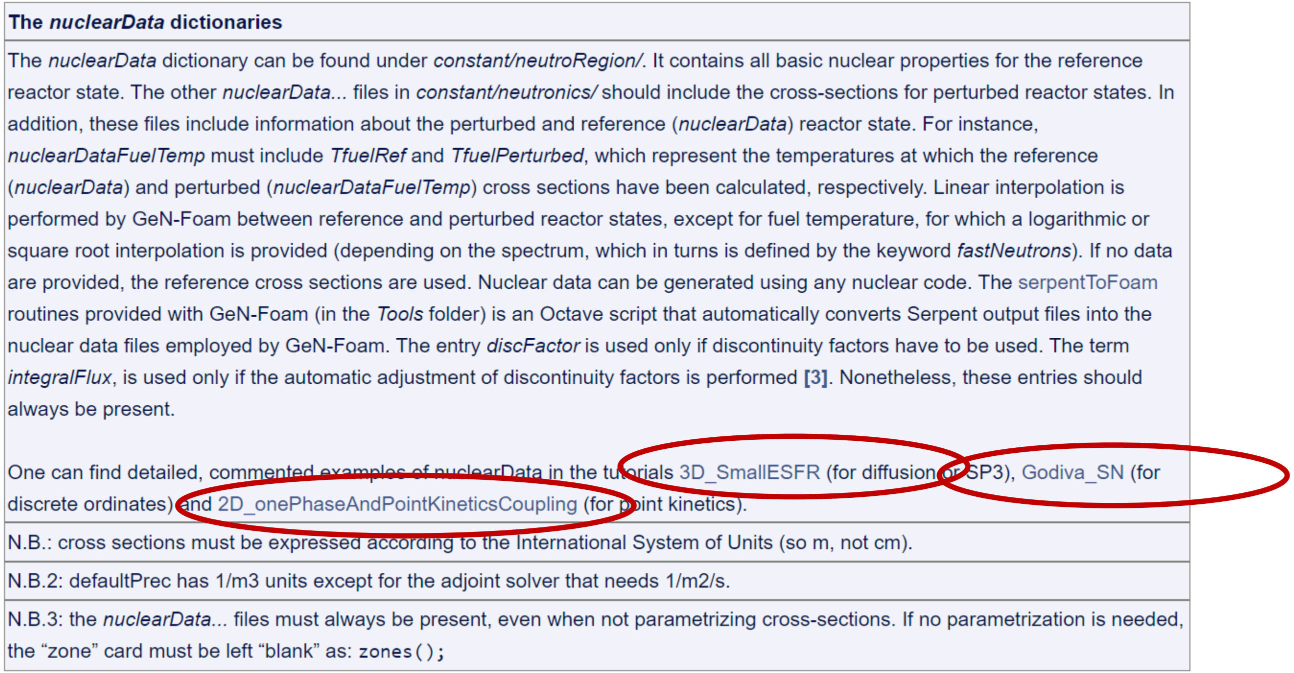
N.B.3: the *nuclearData...* files must always be present, even when not parametrizing cross-sections. If no parametrization is needed, the “zone” card must be left “blank” as: `zones();`



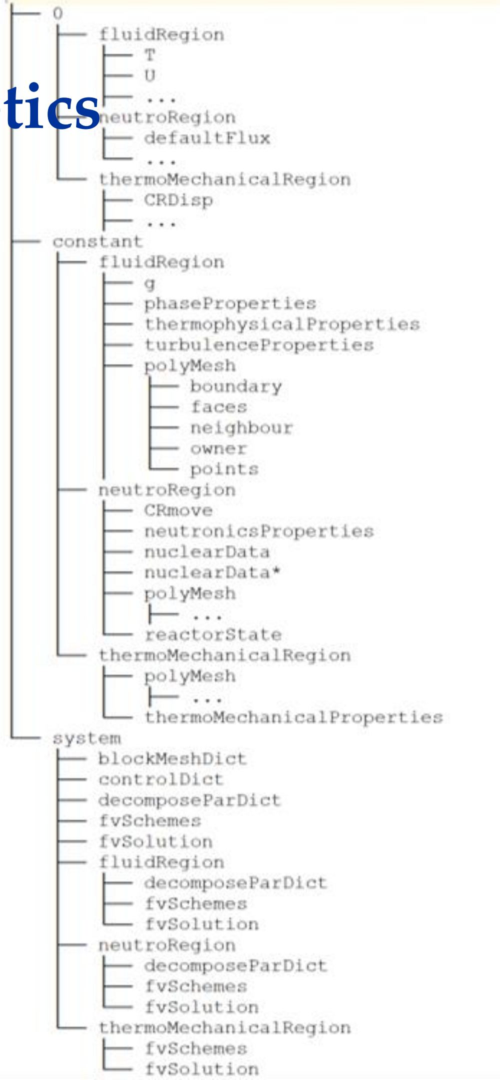
```
├── 0
│   ├── fluidRegion
│   │   ├── T
│   │   ├── U
│   │   └── ...
│   ├── neutroRegion
│   │   ├── defaultFlux
│   │   └── ...
│   └── thermoMechanicalRegion
│       ├── CRDisp
│       └── ...
├── constant
│   ├── fluidRegion
│   │   ├── g
│   │   ├── phaseProperties
│   │   ├── thermophysicalProperties
│   │   ├── turbulenceProperties
│   │   └── polyMesh
│   │       ├── boundary
│   │       ├── faces
│   │       ├── neighbour
│   │       ├── owner
│   │       └── points
│   ├── neutroRegion
│   │   ├── CRmove
│   │   ├── neutronicsProperties
│   │   ├── nuclearData
│   │   ├── nuclearData*
│   │   ├── polyMesh
│   │   │   ├── ...
│   │   └── reactorState
│   └── thermoMechanicalRegion
│       ├── polyMesh
│       │   ├── ...
│       └── thermoMechanicalProperties
└── system
    ├── blockMeshDict
    ├── controlDict
    ├── decomposeParDict
    ├── fvSchemes
    ├── fvSolution
    ├── fluidRegion
    │   ├── decomposeParDict
    │   ├── fvSchemes
    │   └── fvSolution
    ├── neutroRegion
    │   ├── decomposeParDict
    │   ├── fvSchemes
    │   └── fvSolution
    └── thermoMechanicalRegion
        ├── fvSchemes
        └── fvSolution
```

What’s inside: Tutorials - 1D_MSR_pointKinetics

- Look at the dictionaries
  - All the dictionaries are explained in the user manual, which also contain links to tutorials where the dictionary is used and extensively commented

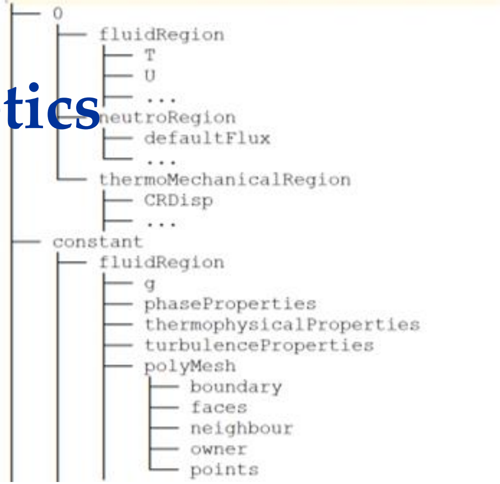


The *nuclearData* dictionaries
The *nuclearData* dictionary can be found under *constant/neutroRegion/*. It contains all basic nuclear properties for the reference reactor state. The other *nuclearData*... files in *constant/neutronics/* should include the cross-sections for perturbed reactor states. In addition, these files include information about the perturbed and reference (*nuclearData*) reactor state. For instance, *nuclearDataFuelTemp* must include *TfuelRef* and *TfuelPerturbed*, which represent the temperatures at which the reference (*nuclearData*) and perturbed (*nuclearDataFuelTemp*) cross sections have been calculated, respectively. Linear interpolation is performed by GeN-Foam between reference and perturbed reactor states, except for fuel temperature, for which a logarithmic or square root interpolation is provided (depending on the spectrum, which in turns is defined by the keyword *fastNeutrons*). If no data are provided, the reference cross sections are used. Nuclear data can be generated using any nuclear code. The serpentToFoam routines provided with GeN-Foam (in the *Tools* folder) is an Octave script that automatically converts Serpent output files into the nuclear data files employed by GeN-Foam. The entry *discFactor* is used only if discontinuity factors have to be used. The term *integralFlux*, is used only if the automatic adjustment of discontinuity factors is performed [3]. Nonetheless, these entries should always be present. One can find detailed, commented examples of nuclearData in the tutorials 3D_SmallESFR (for diffusion or SP3), Godiva_SN (for discrete ordinates) and 2D_onePhaseAndPointKineticsCoupling (for point kinetics).
N.B.: cross sections must be expressed according to the International System of Units (so m, not cm).
N.B.2: defaultPrec has 1/m3 units except for the adjoint solver that needs 1/m2/s.
N.B.3: the *nuclearData*... files must always be present, even when not parametrizing cross-sections. If no parametrization is needed, the “zone” card must be left “blank” as: `zones();`



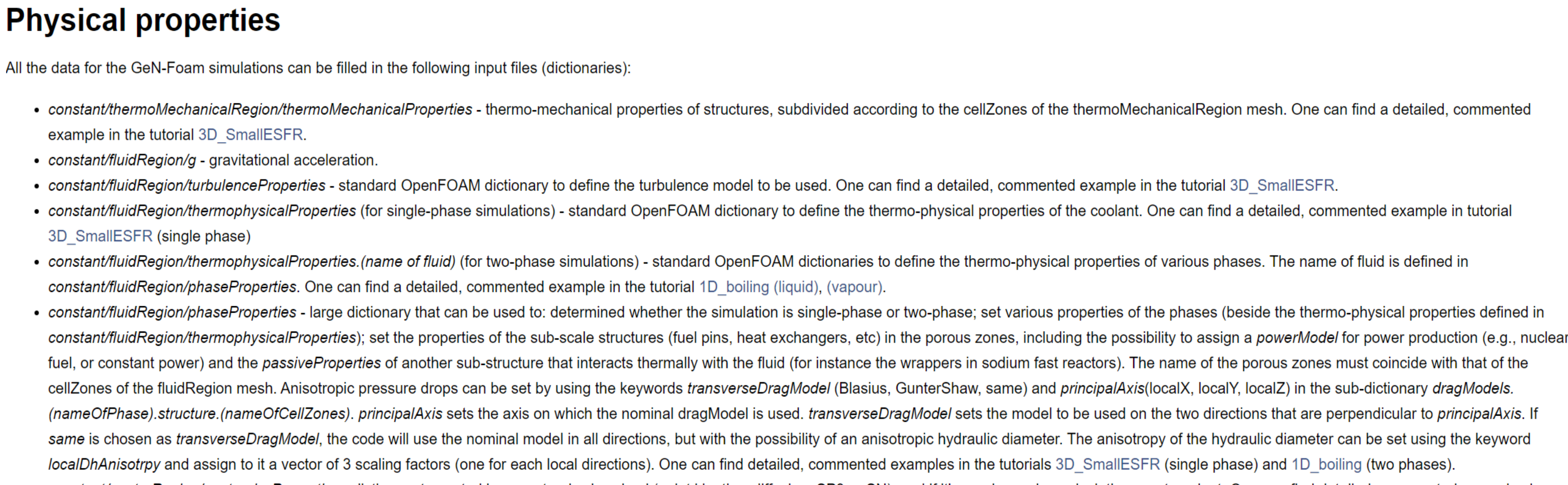
```
├── 0
│   ├── fluidRegion
│   │   ├── T
│   │   ├── U
│   │   └── ...
│   ├── neutroRegion
│   │   ├── defaultFlux
│   │   └── ...
│   └── thermoMechanicalRegion
│       ├── CRDisp
│       └── ...
├── constant
│   ├── fluidRegion
│   │   ├── g
│   │   ├── phaseProperties
│   │   ├── thermophysicalProperties
│   │   ├── turbulenceProperties
│   │   └── polyMesh
│   │       ├── boundary
│   │       ├── faces
│   │       ├── neighbour
│   │       ├── owner
│   │       └── points
│   ├── neutroRegion
│   │   ├── CRmove
│   │   ├── neutronicsProperties
│   │   ├── nuclearData
│   │   ├── nuclearData*
│   │   ├── polyMesh
│   │   │   ├── ...
│   │   └── reactorState
│   └── thermoMechanicalRegion
│       ├── polyMesh
│       │   ├── ...
│       └── thermoMechanicalProperties
└── system
    ├── blockMeshDict
    ├── controlDict
    ├── decomposeParDict
    ├── fvSchemes
    ├── fvSolution
    ├── fluidRegion
    │   ├── decomposeParDict
    │   ├── fvSchemes
    │   └── fvSolution
    ├── neutroRegion
    │   ├── decomposeParDict
    │   ├── fvSchemes
    │   └── fvSolution
    └── thermoMechanicalRegion
        ├── fvSchemes
        └── fvSolution
```

What's inside: Tutorials - 1D_MSR_pointKinetics



```
├── 0
│   ├── fluidRegion
│   │   ├── T
│   │   ├── U
│   │   └── ...
│   ├── neutroRegion
│   │   ├── defaultFlux
│   │   └── ...
│   └── thermoMechanicalRegion
│       ├── CRDisp
│       └── ...
├── constant
│   ├── fluidRegion
│   │   ├── g
│   │   ├── phaseProperties
│   │   ├── thermophysicalProperties
│   │   ├── turbulenceProperties
│   │   ├── polyMesh
│   │   │   ├── boundary
│   │   │   ├── faces
│   │   │   ├── neighbour
│   │   │   ├── owner
│   │   │   └── points
```

- Look at the dictionaries
  - All the dictionaries are explained in the user manual, which also contain links to tutorials where the dictionary is used and extensively commented
  - Also the Preprocessing section of the documentation contains the same links, all in the same documentation page



Physical properties

All the data for the GeN-Foam simulations can be filled in the following input files (dictionaries):

- *constant/thermoMechanicalRegion/thermoMechanicalProperties* - thermo-mechanical properties of structures, subdivided according to the cellZones of the thermoMechanicalRegion mesh. One can find a detailed, commented example in the tutorial 3D_SmallESFR.
- *constant/fluidRegion/g* - gravitational acceleration.
- *constant/fluidRegion/turbulenceProperties* - standard OpenFOAM dictionary to define the turbulence model to be used. One can find a detailed, commented example in the tutorial 3D_SmallESFR.
- *constant/fluidRegion/thermophysicalProperties* (for single-phase simulations) - standard OpenFOAM dictionary to define the thermo-physical properties of the coolant. One can find a detailed, commented example in tutorial 3D_SmallESFR (single phase)
- *constant/fluidRegion/thermophysicalProperties.(name of fluid)* (for two-phase simulations) - standard OpenFOAM dictionaries to define the thermo-physical properties of various phases. The name of fluid is defined in *constant/fluidRegion/phaseProperties*. One can find a detailed, commented example in the tutorial 1D_boiling (liquid), (vapour).
- *constant/fluidRegion/phaseProperties* - large dictionary that can be used to: determined whether the simulation is single-phase or two-phase; set various properties of the phases (beside the thermo-physical properties defined in *constant/fluidRegion/thermophysicalProperties*); set the properties of the sub-scale structures (fuel pins, heat exchangers, etc) in the porous zones, including the possibility to assign a *powerModel* for power production (e.g., nuclear fuel, or constant power) and the *passiveProperties* of another sub-structure that interacts thermally with the fluid (for instance the wrappers in sodium fast reactors). The name of the porous zones must coincide with that of the cellZones of the fluidRegion mesh. Anisotropic pressure drops can be set by using the keywords *transverseDragModel* (Blasius, GunterShaw, same) and *principalAxis*(localX, localY, localZ) in the sub-dictionary *dragModels.(nameOfPhase).structure.(nameOfCellZones)*. *principalAxis* sets the axis on which the nominal dragModel is used. *transverseDragModel* sets the model to be used on the two directions that are perpendicular to *principalAxis*. If *same* is chosen as *transverseDragModel*, the code will use the nominal model in all directions, but with the possibility of an anisotropic hydraulic diameter. The anisotropy of the hydraulic diameter can be set using the keyword *localDhAnisotrpy* and assign to it a vector of 3 scaling factors (one for each local directions). One can find detailed, commented examples in the tutorials 3D_SmallESFR (single phase) and 1D_boiling (two phases).

What’s inside: Tutorials - 1D_MSR_pointKinetics

- Look at the Allrun file

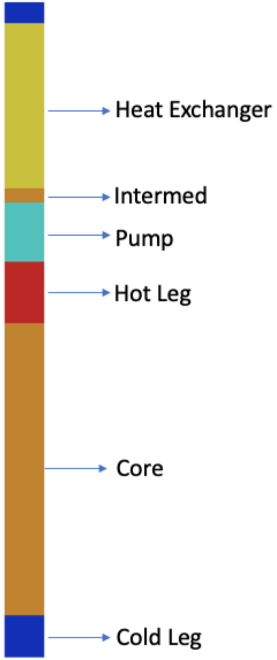
```
cases="steadyState  transient  transientEnd ‟
...
setSteadyState()
{
           runCloneCase $1 $2
           foamDictionary steadyState/system/fvSolution -entry tightlyCoupled -set false
           foamDictionary steadyState/system/controlDict -entry startTime -set 0
           foamDictionary steadyState/system/controlDict -entry endTime -set 100
           foamDictionary steadyState/system/controlDict -entry adjustTimeStep -set true
           foamDictionary steadyState/system/controlDict -entry solveFluidMechanics -set true
           foamDictionary steadyState/system/controlDict -entry solveEnergy -set true
           foamDictionary steadyState/system/controlDict -entry solveNeutronics -set false
           foamDictionary steadyState/system/controlDict -entry solveThermalMechanics -set false
=
...
setTransient()
{
           foamDictionary transient/system/controlDict -entry startTime -set 100
           foamDictionary transient/system/controlDict -entry endTime -set 400
           foamDictionary transient/system/controlDict -entry solveNeutronics -set true

...
```

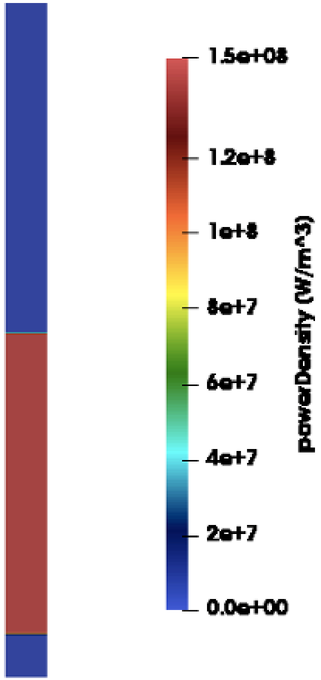
What’s inside: Tutorials - 1D_MSR_pointKinetics

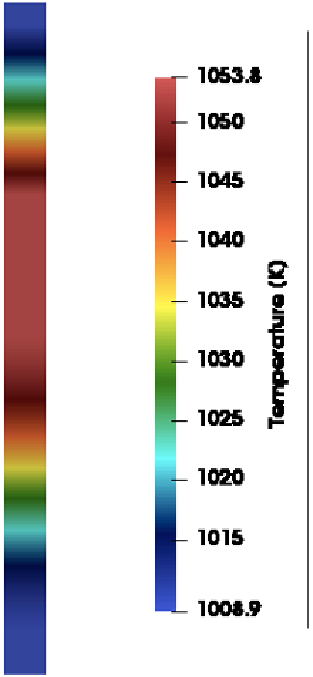
- Run the tutorial -> ./Allrun
- Check the results:
  - Choose a folder: steadyState, transient, transientEnd
  - Use:
    - paraFoam
    - log.GeN-Foam: standard OpenFOAM log
    - GeN-Foam.dat: quick overview of time behavior of main quantities (power, keff, min/max/average fuel and clad temp.)
    - constant/uniform/reactorState for keff
    - in some tutorials, a python script to extract info from log file

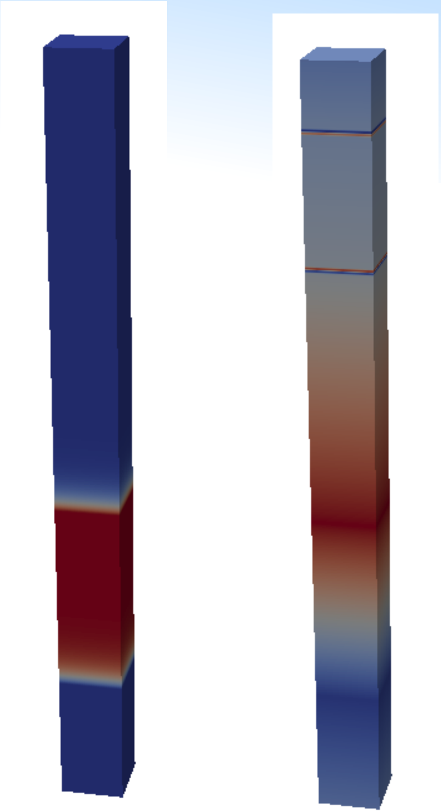
What’s inside: Tutorials - 1D_MSR_pointKinetics

- paraFoam









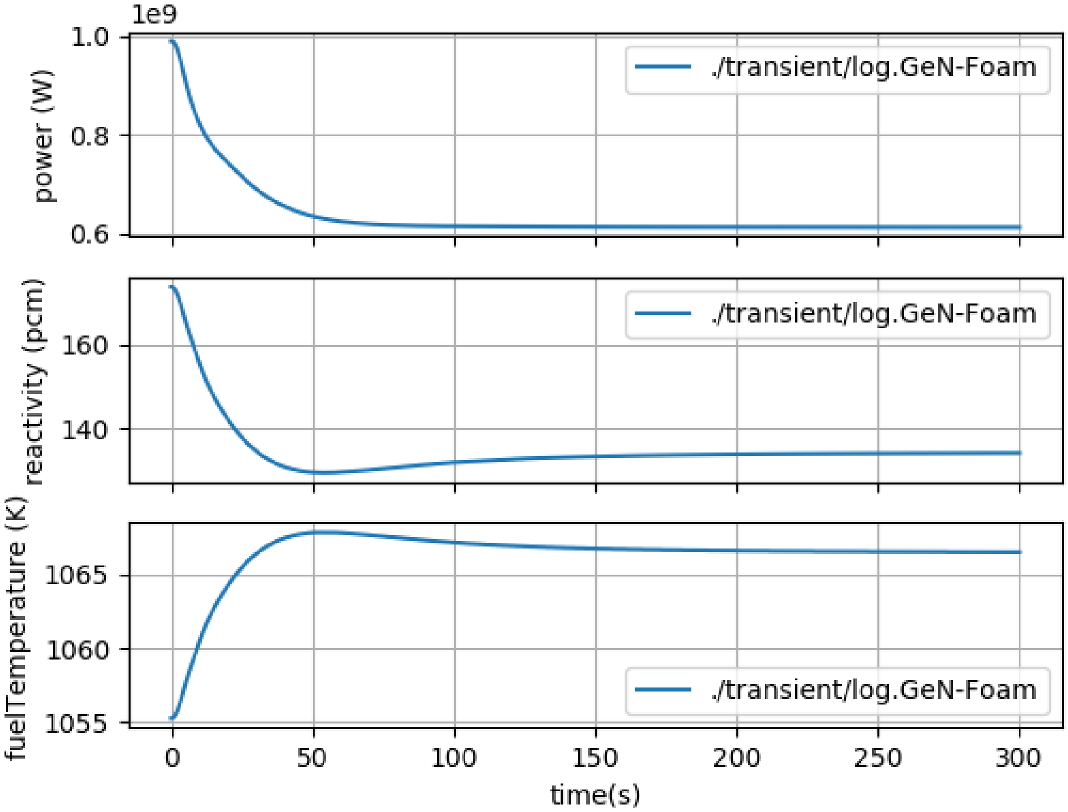
Precursors, group 0 and 7

What's inside: Tutorials - 1D_MSR_pointKinetics

- python script (extract data from log). Type in terminal:

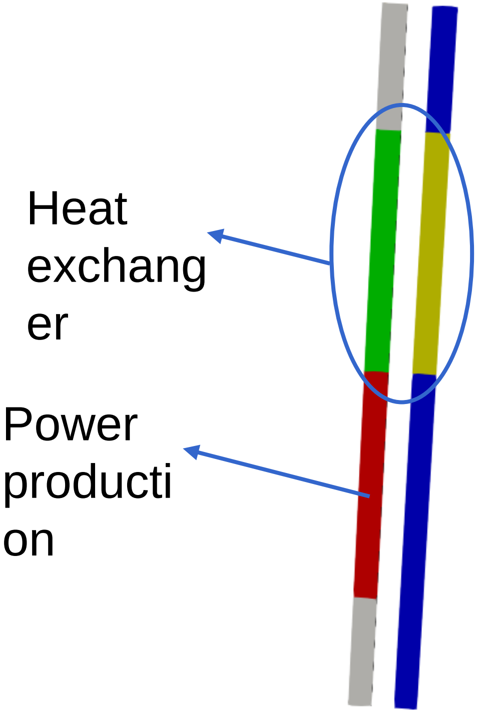
```
Python3 plotPKlin.py
./transient/log.GeN-
Foam
```

- GeN-Foam.dat (contains evolution of main fields over time)



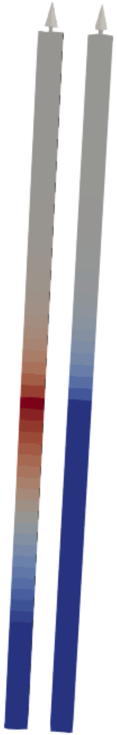
What’s inside: other tutorials - 1D_HX

- Example on how to set up a heat exchanger



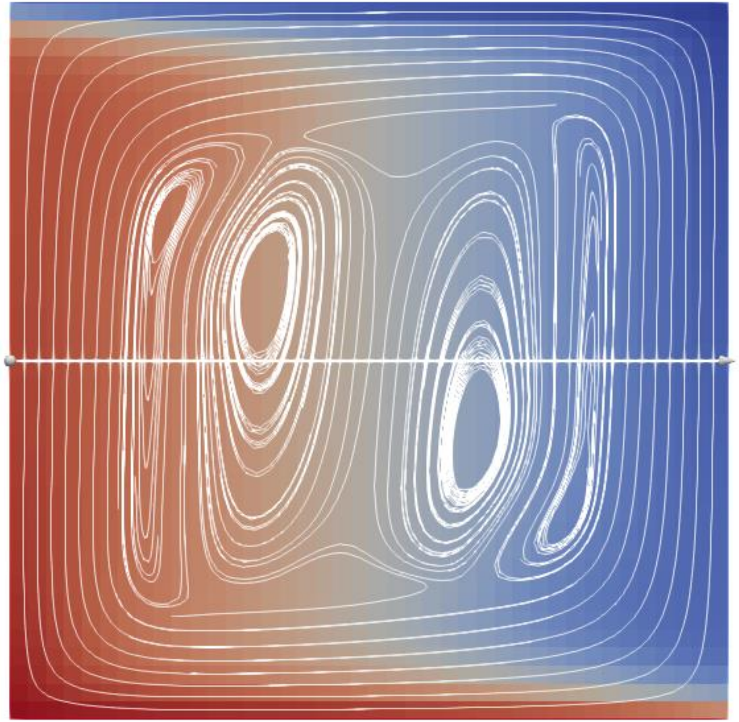
Heat exchanger

Power production



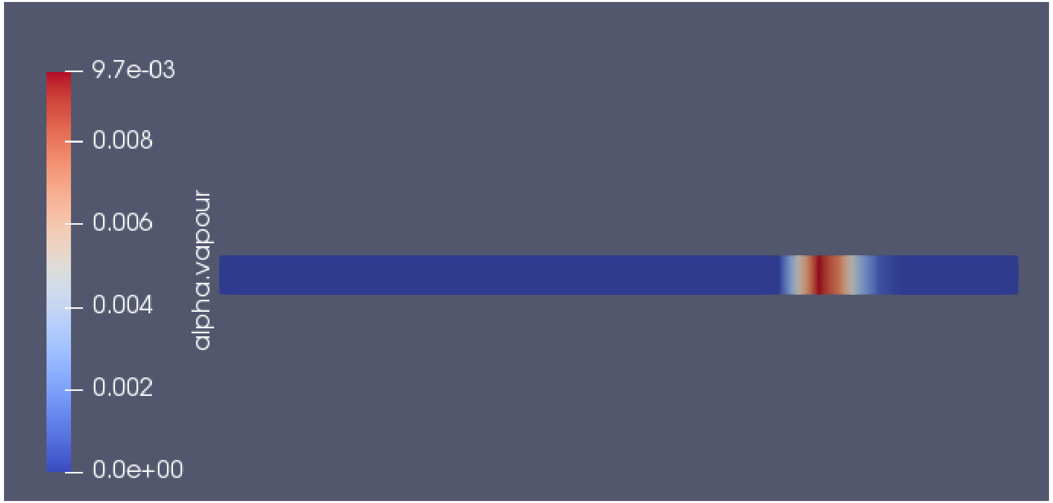
What’s inside: other tutorials - 2D_cavityBoussinesq

- Example of how to use of the Boussinesq approximation for buoyancy based on the standard buoyancy-driven cavity



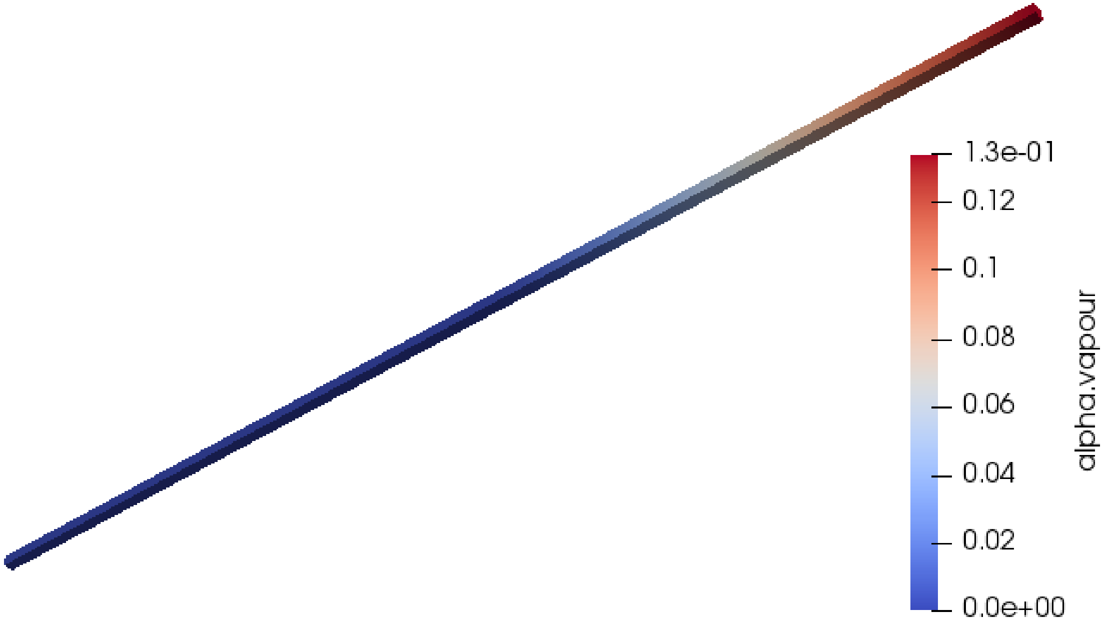
What’s inside: other tutorials - 1D_boiling

- Example of two-phase simulation. 1D channel with a pressure-driven flow of liquid sodium, with power source turned on at time 0, eventually leading to boiling. After a certain time the power is turned off



What’s inside: other tutorials - 1D_PSBT_SC and 1D_CHF

- Example of use for water boiling, based on the NEA PSBT benchmark
- Example of use for water boiling, including boiling crisis (not yet validated!)

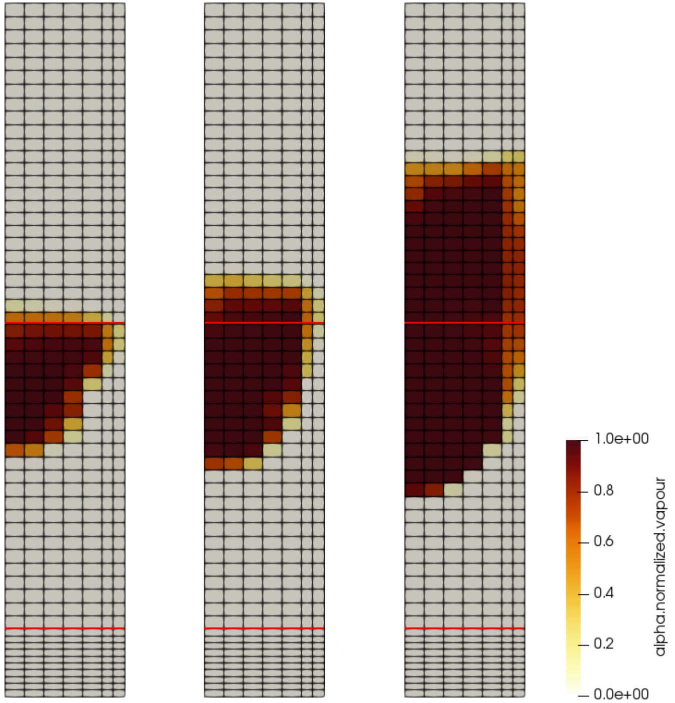


What’s inside: other tutorials - 2D_voidMotionNoPhaseChange

- Simple two-phase case without mass transfer between phases

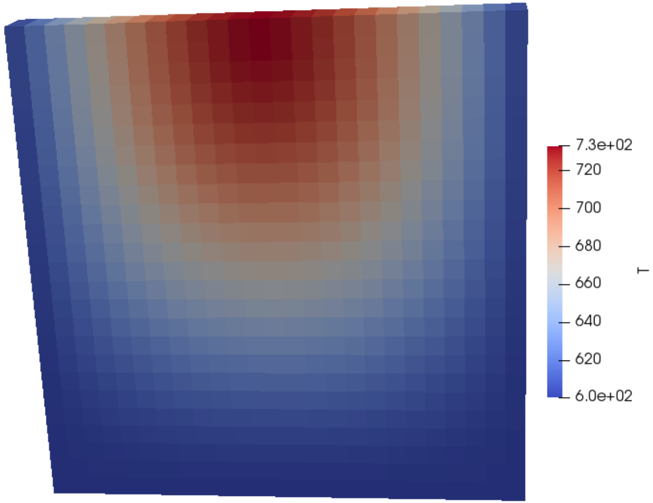
What's inside: other tutorials - 2D_KNS37-L22

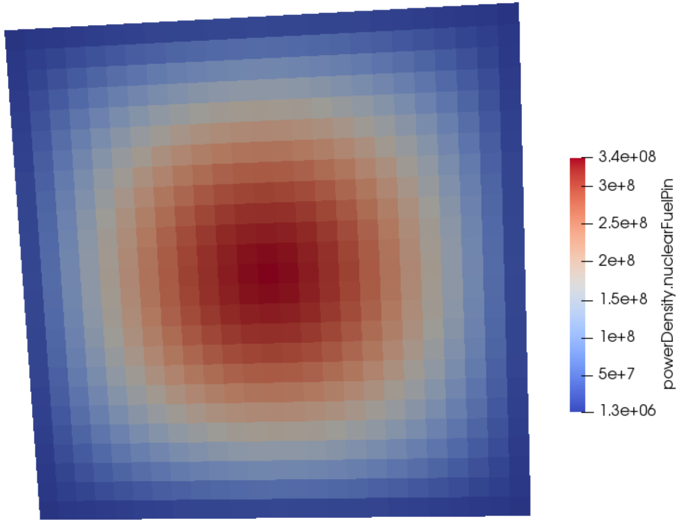
- Example of use for sodium boiling, based on the KNS experiment



What's inside: other tutorials - 2D_onePhaseAndPointKineticsCoupling

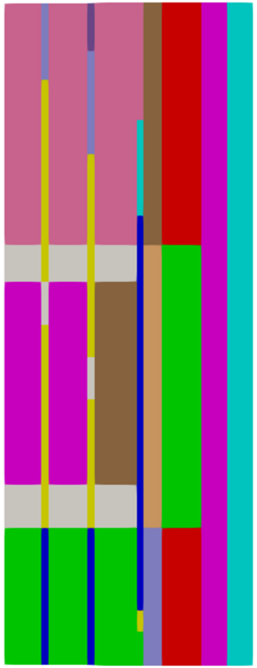
- Simple case with coupling of thermal-hydraulics and point-kinetics

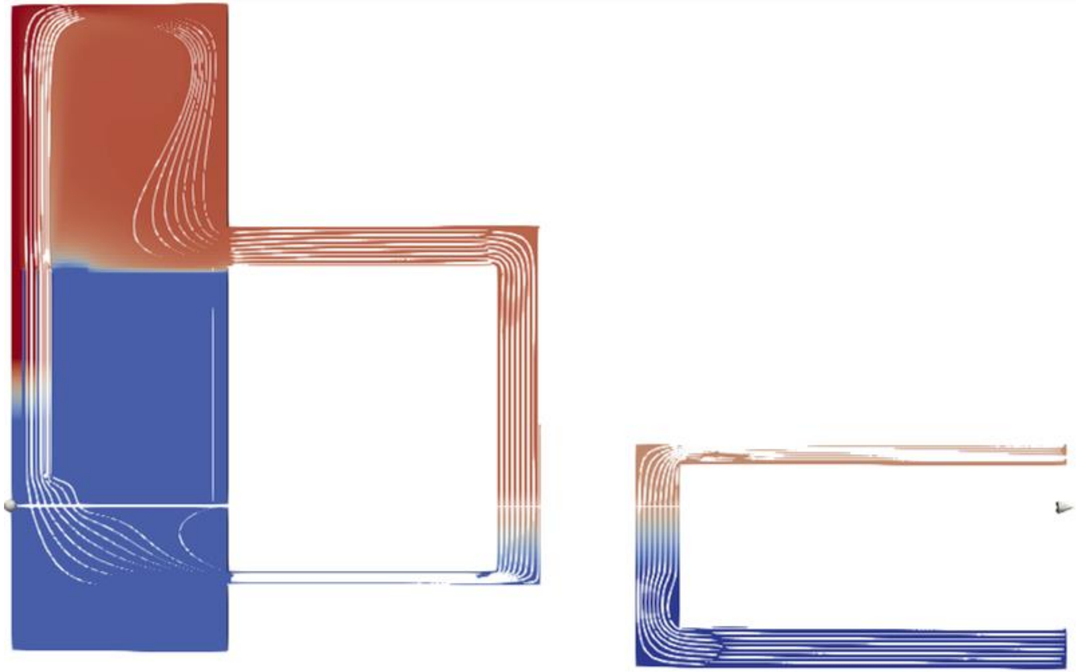




What’s inside: other tutorials - 2D_FFTF

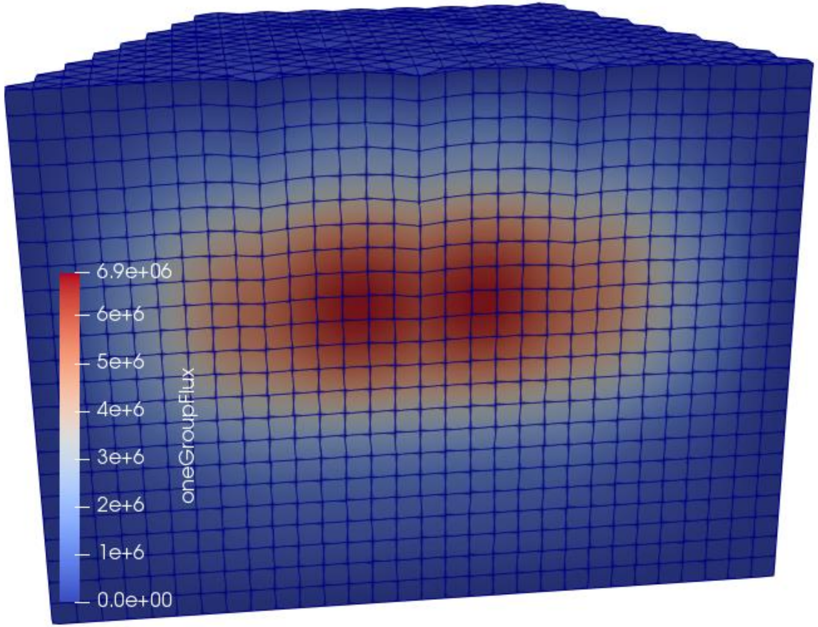
- 2-D model of the FFTF. Simulation of a ULOF. Thermal-hydraulics plus neutronics





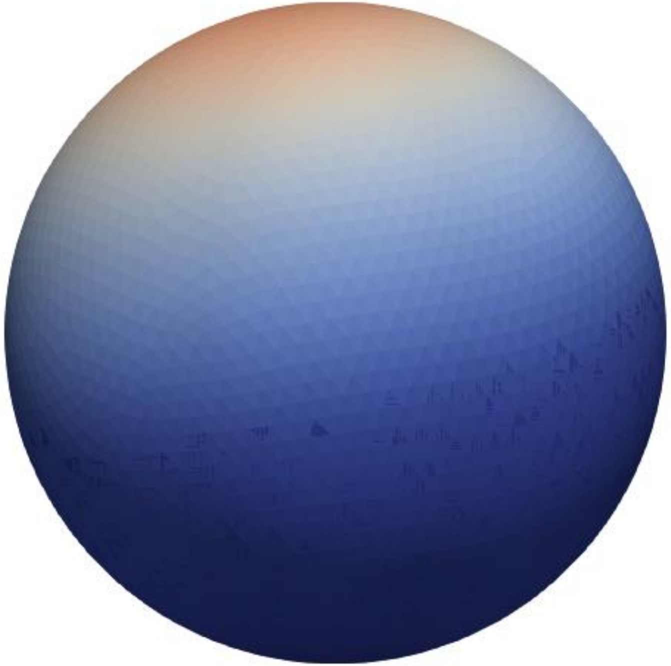
What’s inside: other tutorials - 3D_SmallESFR

- Slightly smaller version of the European Sodium Fast Reactor
- Example of a 3D full multi-physics simulation, including core deformation



What's inside: other tutorials - Godiva_SN

- Example of a discrete ordinate calculation of Godiva



What’s inside: other tutorials - EMPTY

- GeN-Foam requires a minimal set of inputs (incl. mesh) for all physics, even when not solving for them

- EMPTY contains a minimal set of dummy inputs
  - Warning: dummy meshes are small. If you need to run in parallel on many cores, you’ll have to refine the meshes so that they have at least the same number of cells as the number of parallel processes

- EMPTY can be uses as starting point for new cases

Correct way of approaching GeN-Foam

1. Learn OpenFOAM
2. Read the doxygen documentation (it won’t take more than a few hours and it will spare one weeks of possible frustration)
3. If unfamiliar with some aspects (e.g., porous-medium thermal-hydraulics), one can start by looking at the papers referenced in the online documentation
4. If unfamiliar with OpenFOAM-related aspects (meshing, schemes, linear solvers, etc.), one can refer to the various OpenFOAM resources
5. Try and understand a few tutorials that are close to your application
6. Start familiarizing with the source-code (doxygen documentation helps a lot…)
7. Pick the tutorial that is closer to your one’s own and start from there. If no tutorial close enough, start from the EMPTY case.
8. Accompany use with development/understanding of source code. This is the key to a proficient use

Summary of available resources

- General OpenFOAM resources (GeN-Foam is just a high-level application of a much larger library)
- Theory papers
  - https://foam-for-nuclear.gitlab.io/GeN-Foam/citelist.html
  - End of last lecture
- Doxygen documentation: standad doxygen, introduction, user manual, tutorials, tips and tricks, important notes, **recent changes in the case folder**
  - Online (https://foam-for-nuclear.gitlab.io/GeN-Foam/index.html)
  - Local (can be built with doxygen)
- Tutorials
  - Cover essentially all functionalities
  - All dictionaries commented in at least one tutorial (link in the documentation)
- Source code
  - Normally well written and commented
  - Important to improve understanding
- Forum
  - https://foam-for-nuclear.org/phpBB/

IAEA
International Atomic Energy Agency

Joint ICTP-IAEA Workshop on Open-Source Nuclear Codes for Reactor Analysis
August 7-11 2023

Thank you!

Contact: ONCORE@iaea.org

Course Enrolment : Multi-physics modelling and simulation of nuclear reactors using OpenFOAM
ONCORE: Open-source Nuclear Codes for Reactor Analysis