Use of OpenFOAM for multi-physics in nuclear

Carlo Fiorina

École polytechnique fédérale de Lausanne

This workshop

Carlo Fiorina

What to expect

- Overview of the multiphysics modelling capabilities of OpenFOAM
- Lessons learnt
- A crash introduction and learning best practices for OpenFOAM
- (A crash introduction and learning best practices for existing nuclear solvers - GeN-Foam)

What not to expect

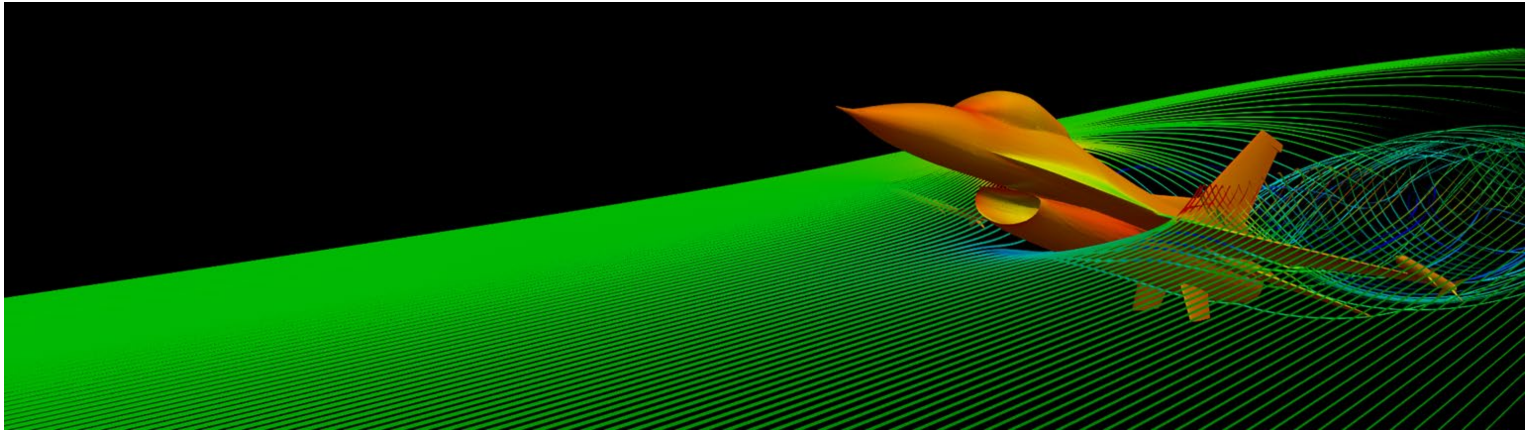
- A full course on the use of OpenFOAM or GeN-Foam

- Possibly more slides than I can present
- Objective to have consistent and readable material

Open∇FOAM

The Open Source CFD Toolbox

- ❑ What is OpenFOAM?
  - ✓ Distributed as CFD toolbox
  - ✓ ~10k to 20k estimated users worldwide



Open∇FOAM

The Open Source CFD Toolbox

- ❑ What is OpenFOAM?
  - ✓ Distributed as CFD toolbox
  - ✓ ~10k to 20k estimated users worldwide
  - ✓ OpenFOAM = Open Field Operation And Manipulation
  - ✓ Essentially a large, well organized, HPC-scalable, C++ library for the finite-volume discretization and solution of PDEs, and including several functionalities like ODE solvers, projection algorithms, and mesh search algorithms
  - ✓ Object-oriented, with a high-level “fail-safe” API

$$\frac{1}{v_i}\frac{\partial \varphi_i}{\partial t} - \Delta(D_i\varphi_i) = S$$

```
fvm::ddt(IV,flux_i])- fvm::laplacian(D,flux_i])= S
```

Open∇FOAM

The Open Source CFD Toolbox

- ❑ What is OpenFOAM?
  - ✓ Distributed as **CFD toolbox**
  - ✓ **~10k to 20k estimated users** worldwide
  - ✓ OpenFOAM = Open Field Operation And Manipulation
  - ✓ Essentially a large, well organized, HPC-scalable, C++ library for the **finite-volume** discretization and solution of PDEs, and **including several functionalities** like ODE solvers, projection algorithms, and mesh search algorithms
  - ✓ **Object-oriented**, with a **high-level “fail-safe” API**

$$\frac{1}{v_i}\frac{\partial \varphi_i}{\partial t} - \Delta(D_i \varphi_i) = S$$

```
fvm::ddt(IV,flux_i])- fvm::laplacian(D,flux_i])= S
```

Disclaimer

Part of the following is content taken from

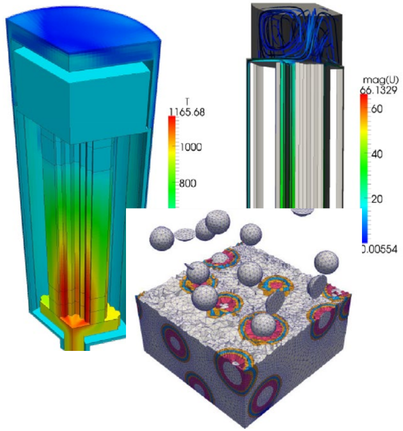
- ❑ Carlo Fiorina, Ivor Clifford, Stephan Kelm, Stefano Lorenzi, 2022. “On the development of multi-physics tools for nuclear reactor analysis based on OpenFOAM ®: state of the art, lessons learned and perspectives”. Nuclear Engineering and Design 387, 111604. https://www.sciencedirect.com/science/article/pii/S0029549321005562

Use of OpenFOAM for multi-physics

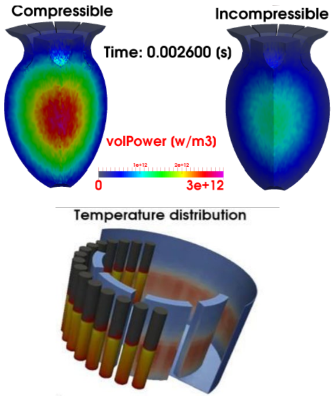
2000-2010
First activities

2010-2015
First widespread use

2015-2021
First coordinated and persistent developments

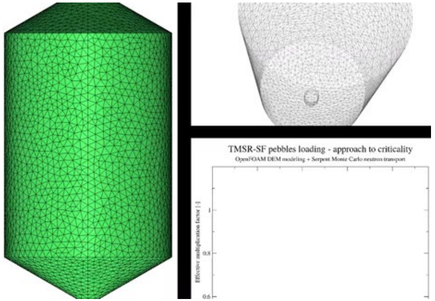


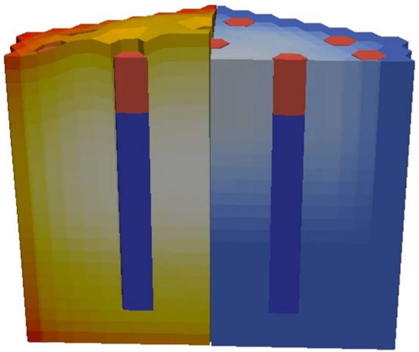
PBMRs and HTRs



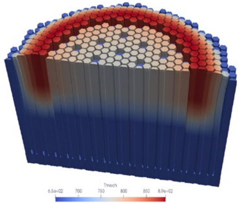
MSRs

FHRs

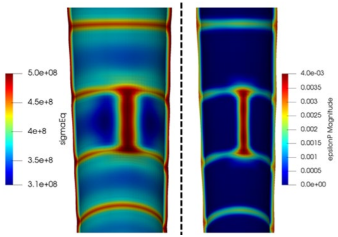

TMSR-SF pebbles loading - approach to criticality
OpenFOAM DEM modeling + Serpent Monte Carlo neutron transport
Effective multiplication factor [-]
1
0.8
0.6



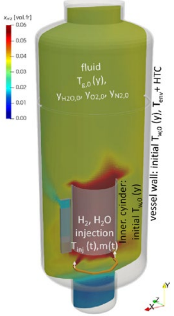
SFRs



GeN-Foam

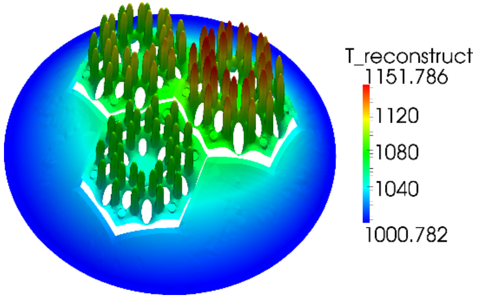


OFFBEAT


x_{H2} [vol.fr]
0.06
0.05
0.04
0.03
0.02
0.01
0.00
fluid
$T_{g,0}$ (y),
$y_{H2O,0}$, $y_{O2,0}$, $y_{N2,0}$
vessel wall: initial $T_{w,0}$ (y), T_{env} + HTC
Inner. cyinder: initial $T_{w,0}$ (y)
H_2, H_2O injection
T_{inj} (t), m(t)
Y
Z
X

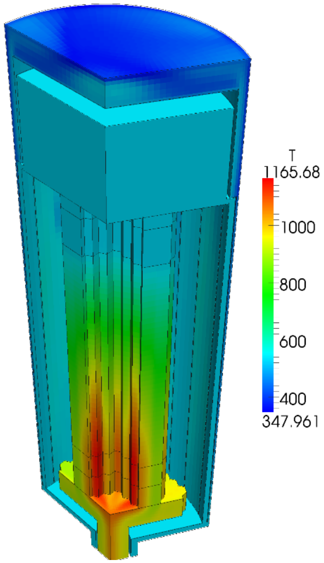
containmentFoam

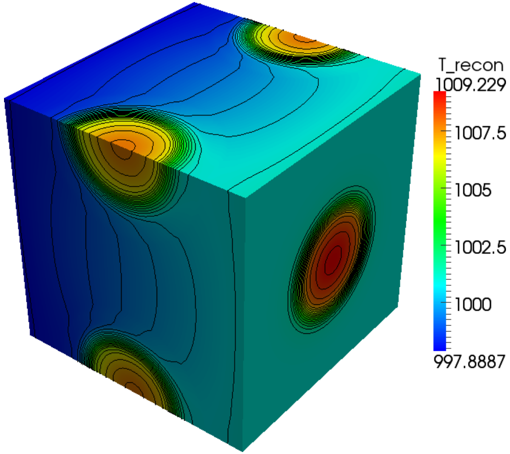
HTR modelling (PBMR -> I. Clifford)



ROM reconstructed solution for a fuel element

Full-core coarse-mesh thermal-hydraulics



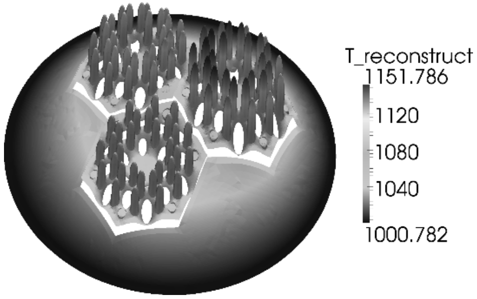


ROM reconstructed solution for TRISO coated particles

HTR modelling (PBMR -> I. Clifford)

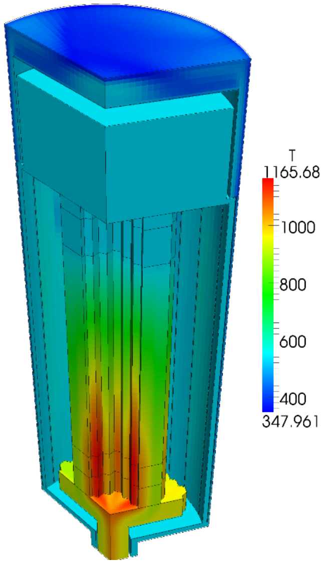
- Porous-medium thermal-hydraulics

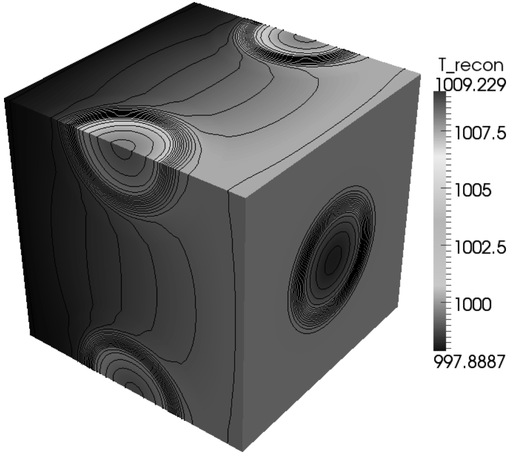
Carlo Fiorina



ROM reconstructed solution for a fuel element

Full-core coarse-mesh thermal-hydraulics

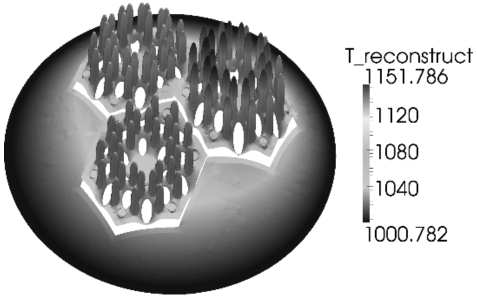




ROM reconstructed solution for TRISO coated particles

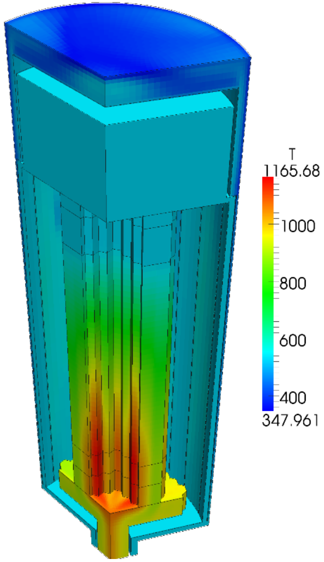
HTR modelling (PBMR -> I. Clifford)

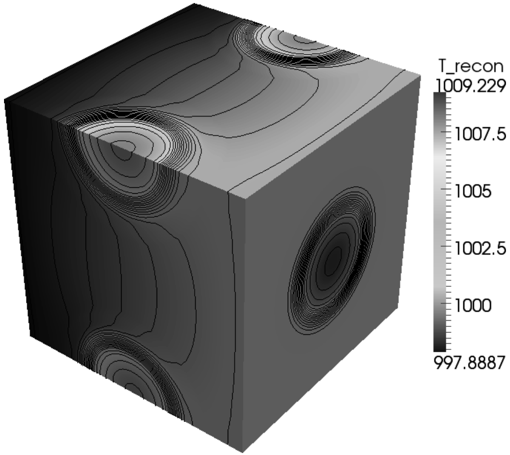
- ❑ Porous-medium thermal-hydraulics
  - ✓ Available CFD RANS



ROM reconstructed solution for a fuel element

Full-core coarse-mesh thermal-hydraulics



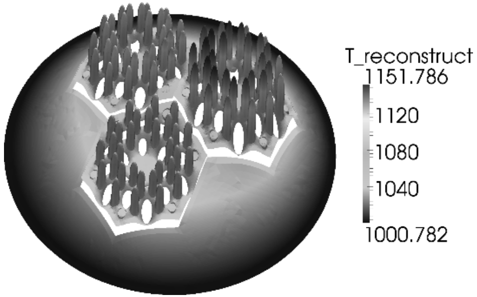


ROM reconstructed solution for TRISO coated particles

Carlo Fiorina

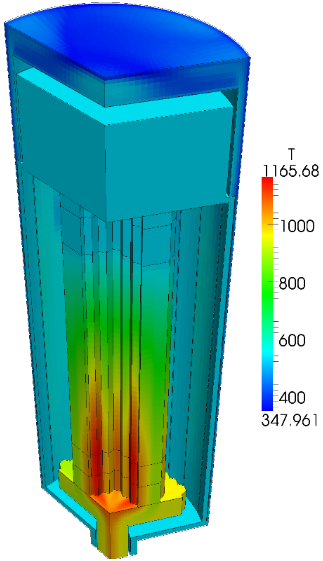
HTR modelling (PBMR -> I. Clifford)

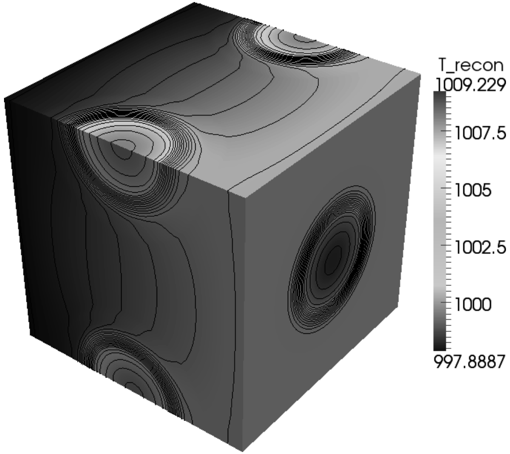
- ❑ Porous-medium thermal-hydraulics
  - ✓ Available CFD RANS plus source terms



ROM reconstructed solution for a fuel element

Full-core coarse-mesh thermal-hydraulics





ROM reconstructed solution for TRISO coated particles

Carlo Fiorina

Porous-medium thermal-hydraulics: governing equations

The coarse-mesh governing equations (Navier-Stokes and enthalpy) are:

$$\frac{\partial}{\partial t}(\alpha_i \rho_i) + \nabla \cdot (\alpha_i \mathbf{u}_i \rho_i) = -\Gamma_{i\to j}$$

$$\frac{\partial}{\partial t}(\alpha_i \rho_i \mathbf{u}_i) + \nabla \cdot (\alpha_i \rho_i \mathbf{u}_i \otimes \mathbf{u}_i) =$$

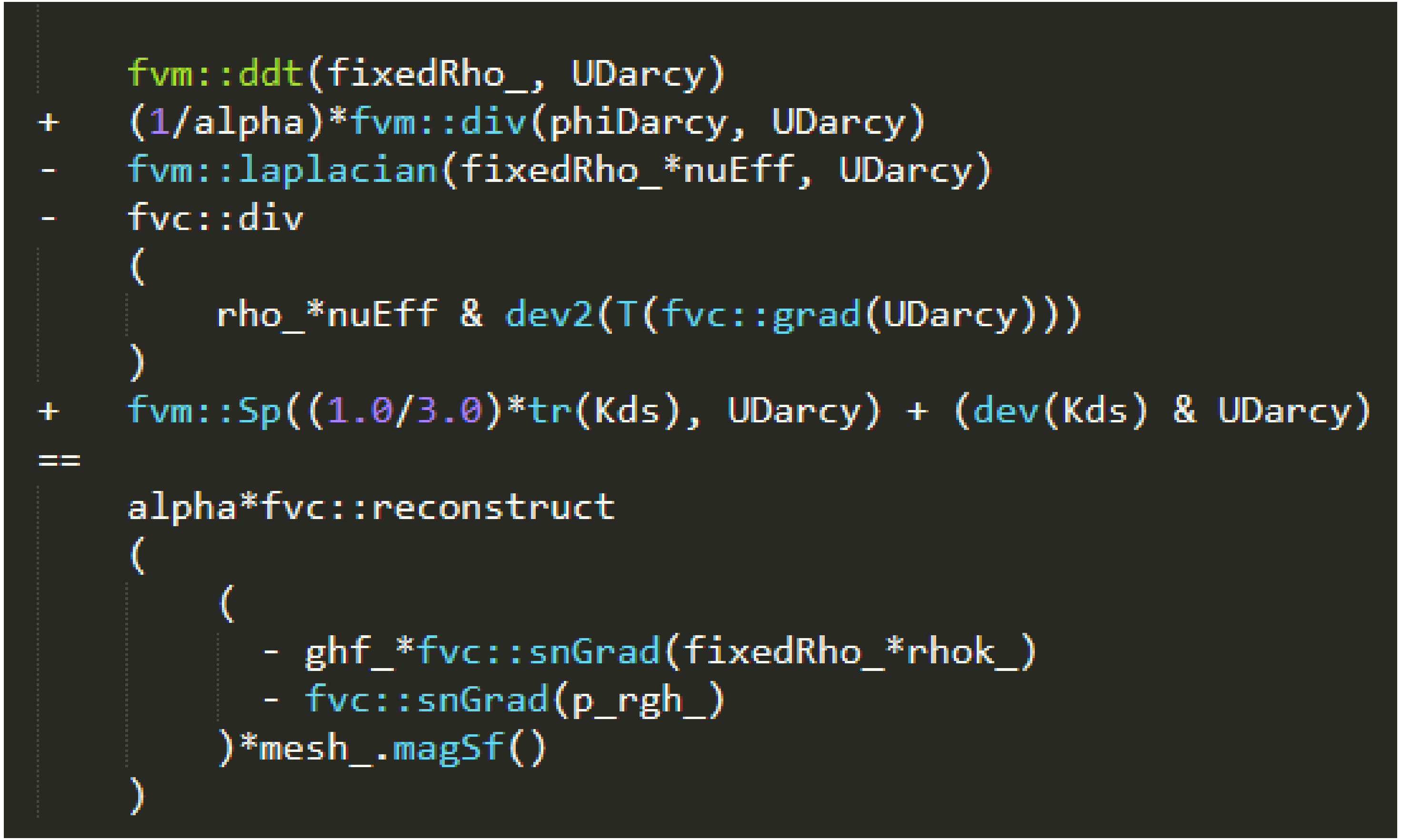
$$-\alpha_i \nabla p + \nabla \cdot (\alpha_i \boldsymbol{\sigma}_{d,i}) + \alpha_i \rho_i \mathbf{g} - \mathbf{S}_{\mathbf{u},i\to j}$$

$$\frac{\partial}{\partial t}(\alpha_i \rho_i h_i) + \nabla \cdot (\alpha_i \mathbf{u}_i \rho_i h_i) =$$

$$\nabla \cdot (\alpha_i \kappa_i \boldsymbol{T}_i \cdot \nabla T_i) + \alpha_i \frac{\partial}{\partial t} p + \alpha_i \rho_i \mathbf{u}_i \cdot \mathbf{g} + \alpha_i q_{int,i} - S_{h,i\to j}$$

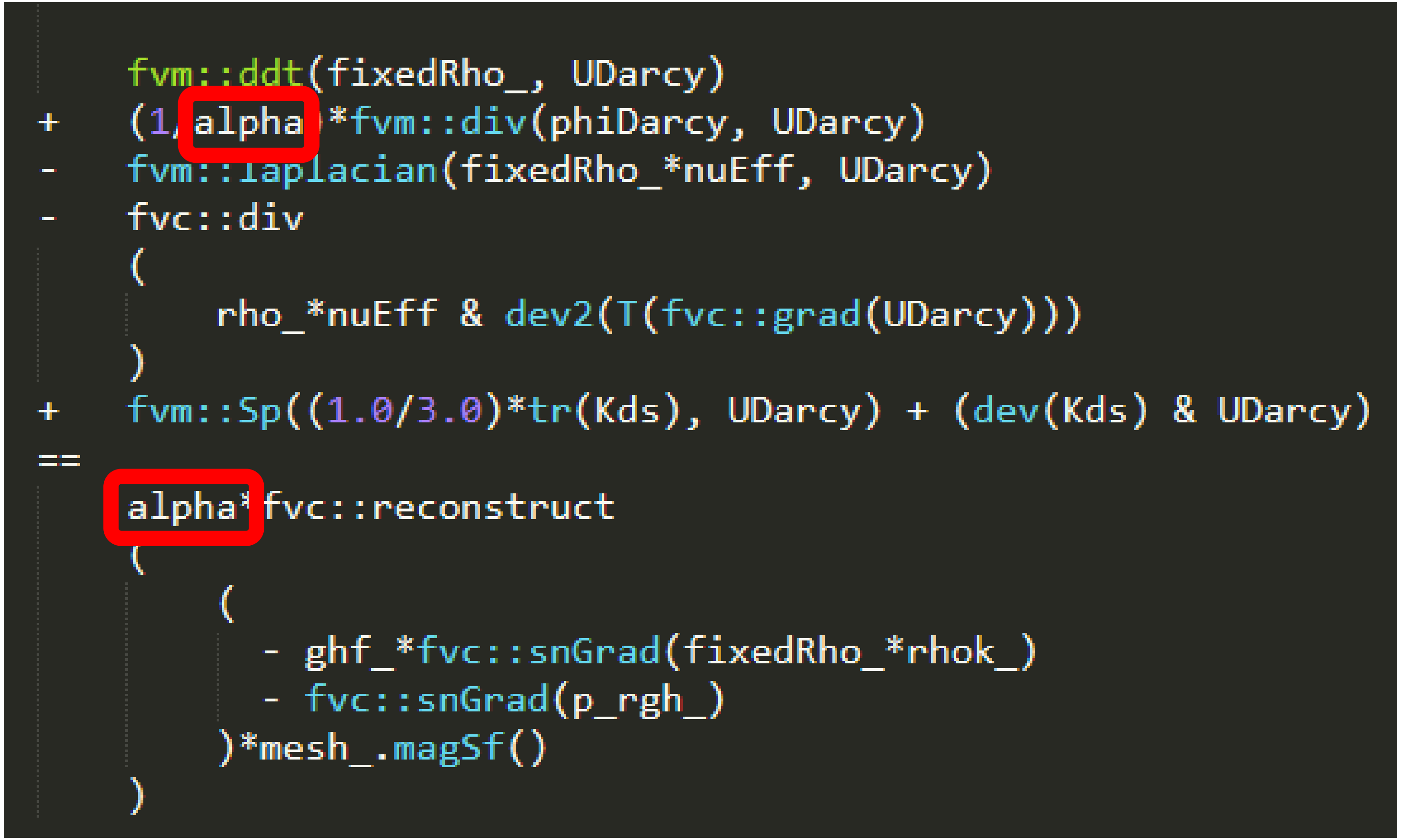
These reduce to traditional CFD approaches in clear fluid regions and a system-code-like approach in 1-D regions (multiple scales).

Porous-medium thermal-hydraulics: governing equations



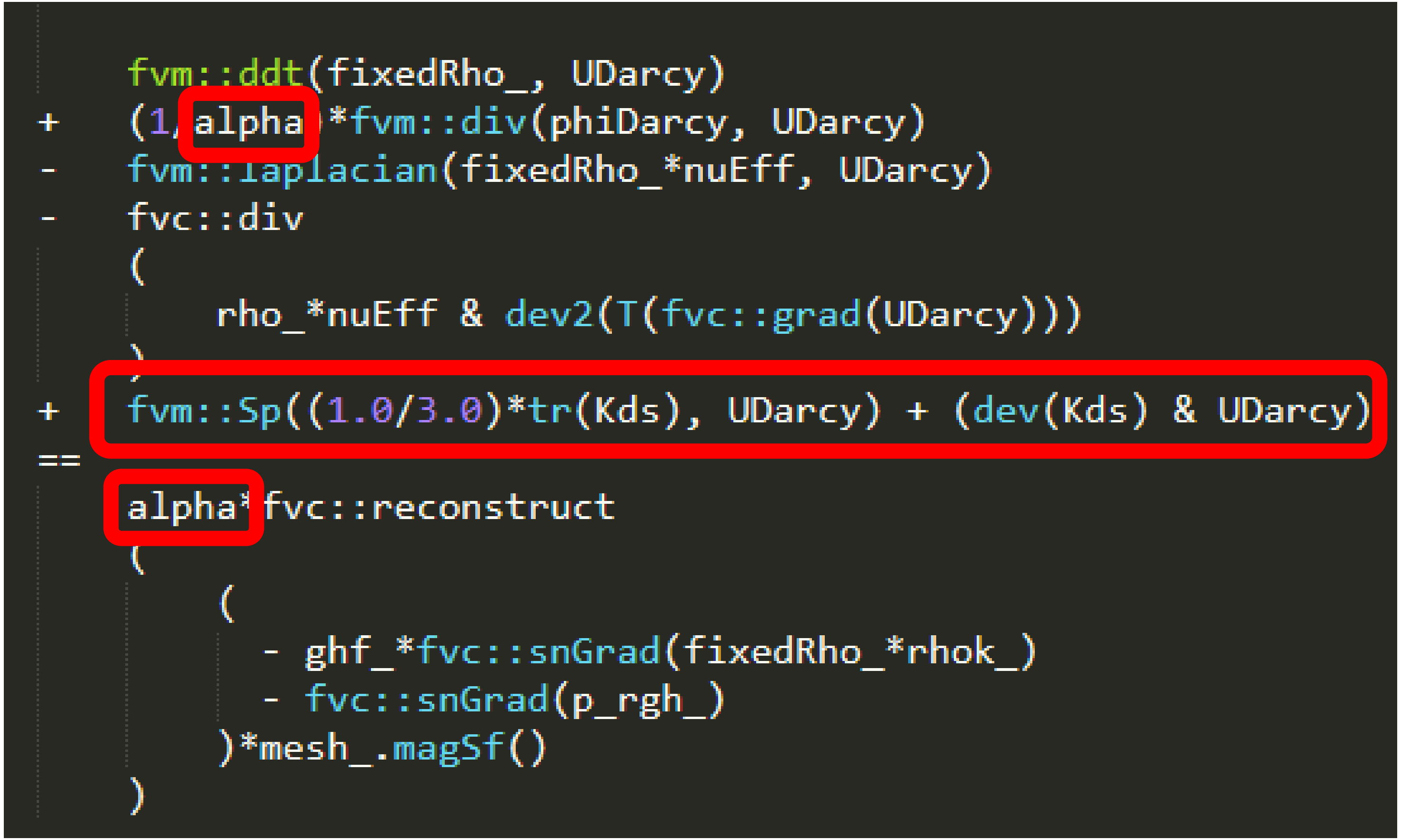
```
    fvm::ddt(fixedRho_, UDarcy)
+   (1/alpha)*fvm::div(phiDarcy, UDarcy)
-   fvm::laplacian(fixedRho_*nuEff, UDarcy)
-   fvc::div
    (
        rho_*nuEff & dev2(T(fvc::grad(UDarcy)))
    )
+   fvm::Sp((1.0/3.0)*tr(Kds), UDarcy) + (dev(Kds) & UDarcy)
==
    alpha*fvc::reconstruct
    (
        (
          - ghf_*fvc::snGrad(fixedRho_*rhok_)
          - fvc::snGrad(p_rgh_)
        )*mesh_.magSf()
    )
```

Porous-medium thermal-hydraulics: governing equations



```
    fvm::ddt(fixedRho_, UDarcy)
+   (1/alpha)*fvm::div(phiDarcy, UDarcy)
-   fvm::laplacian(fixedRho_*nuEff, UDarcy)
-   fvc::div
    (
        rho_*nuEff & dev2(T(fvc::grad(UDarcy)))
    )
+   fvm::Sp((1.0/3.0)*tr(Kds), UDarcy) + (dev(Kds) & UDarcy)
==
    alpha*fvc::reconstruct
    (
        (
          - ghf_*fvc::snGrad(fixedRho_*rhok_)
          - fvc::snGrad(p_rgh_)
        )*mesh_.magSf()
    )
```

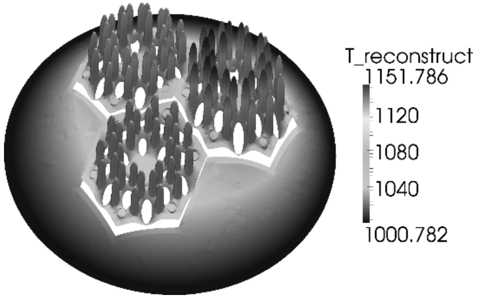
Porous-medium thermal-hydraulics: governing equations



```
    fvm::ddt(fixedRho_, UDarcy)
+   (1/alpha)*fvm::div(phiDarcy, UDarcy)
-   fvm::laplacian(fixedRho_*nuEff, UDarcy)
-   fvc::div
    (
        rho_*nuEff & dev2(T(fvc::grad(UDarcy)))
    )
+   fvm::Sp((1.0/3.0)*tr(Kds), UDarcy) + (dev(Kds) & UDarcy)
==
    alpha*fvc::reconstruct
    (
        (
          - ghf_*fvc::snGrad(fixedRho_*rhok_)
          - fvc::snGrad(p_rgh_)
        )*mesh_.magSf()
    )
```

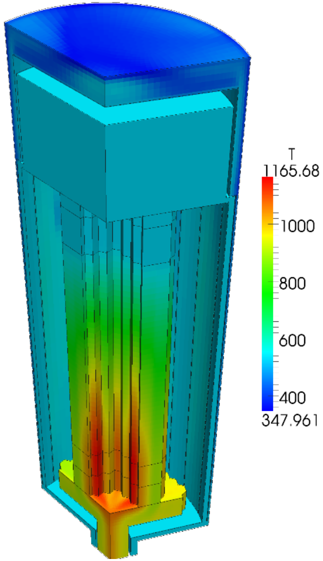
HTR modelling (PBMR -> I. Clifford)

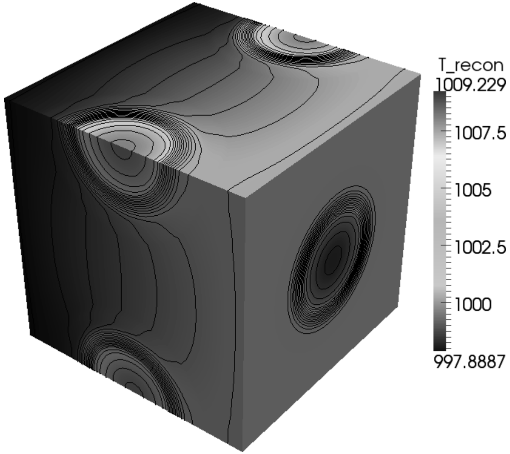
- Porous-medium thermal-hydraulics
  - ✓ Available CFD RANS plus source terms



ROM reconstructed solution for a fuel element

Full-core coarse-mesh thermal-hydraulics



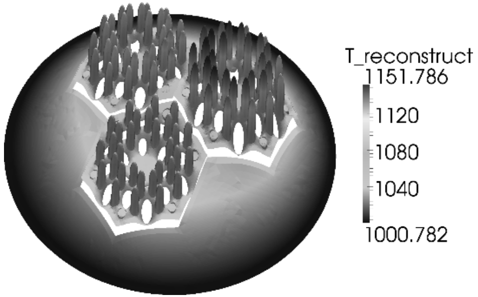


ROM reconstructed solution for TRISO coated particles

Carlo Fiorina

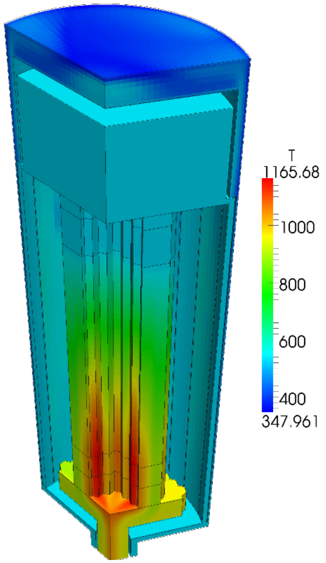
HTR modelling (PBMR -> I. Clifford)

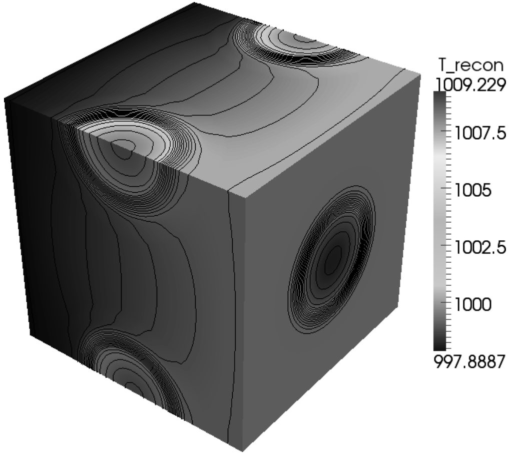
- ❑ Porous-medium thermal-hydraulics
  - ✓ Available CFD RANS plus source terms
  - ✓ Modified discretization to account for discontinuous pressure



ROM reconstructed solution for a fuel element

Full-core coarse-mesh thermal-hydraulics



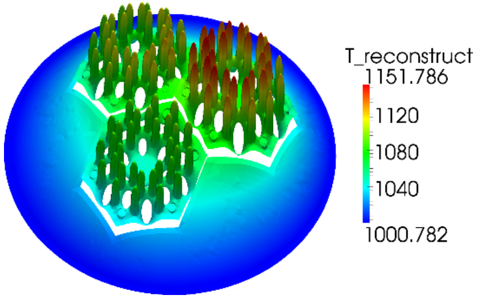


ROM reconstructed solution for TRISO coated particles

Carlo Fiorina

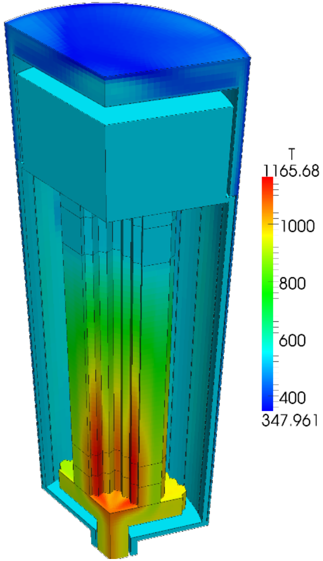
HTR modelling (PBMR -> I. Clifford)

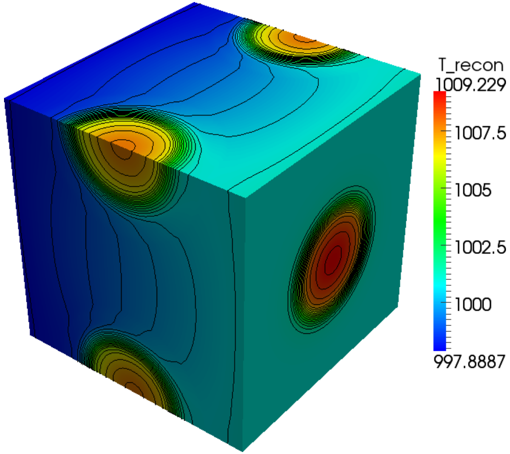
- ❑ Porous-medium thermal-hydraulics
  - ✓ Available CFD RANS plus source terms
  - ✓ Modified discretization to account for discontinuous pressure
- ❑ ROM reconstructed multi-scale temperature
  - ✓ Multi-mesh
  - ✓ Mesh-to-mesh projections
  - ✓ Available ROM library
  - ✓ Built-in ODE solvers



ROM reconstructed solution for a fuel element

Full-core coarse-mesh thermal-hydraulics



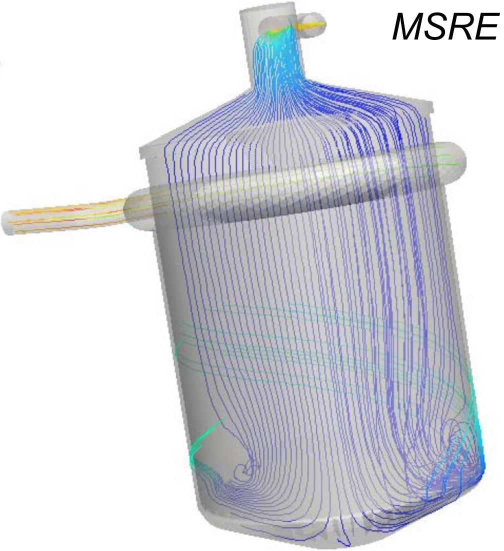


ROM reconstructed solution for TRISO coated particles

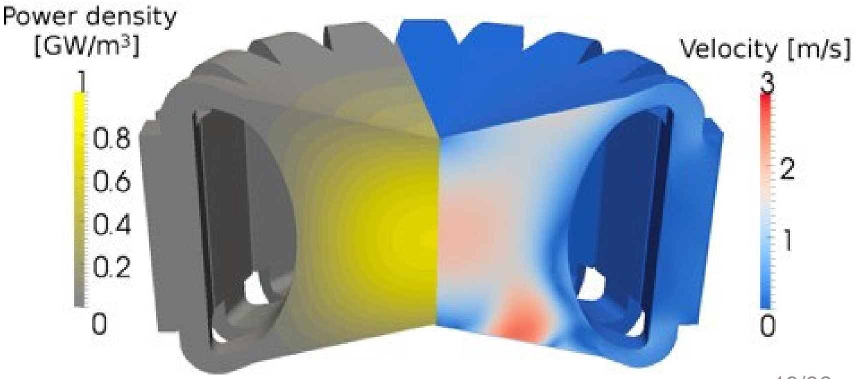
Carlo Fiorina

MSR modelling (M. Aufiero -> PoliMi + CNRS / GeN-Foam)

MSRE

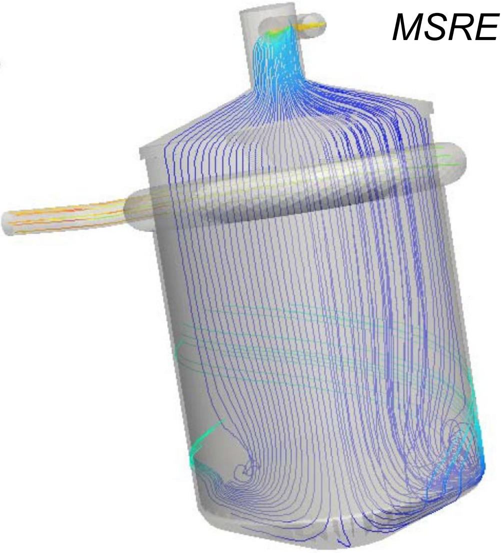


MSFR



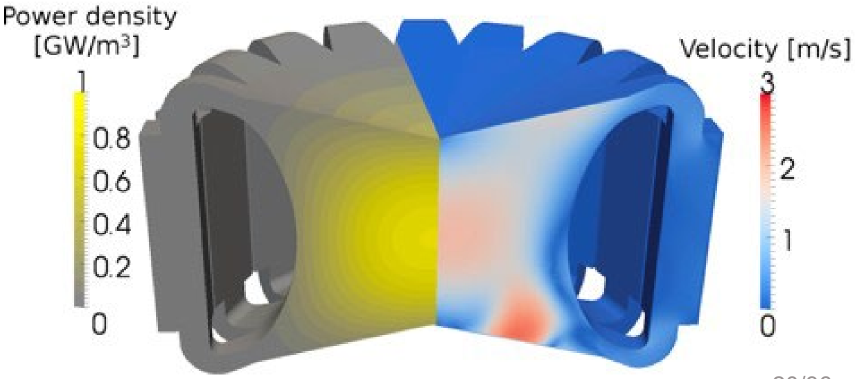
MSR modelling (M. Aufiero -> PoliMi + CNRS / GeN-Foam)

- Available CFD solvers
- Arbitrary geometries



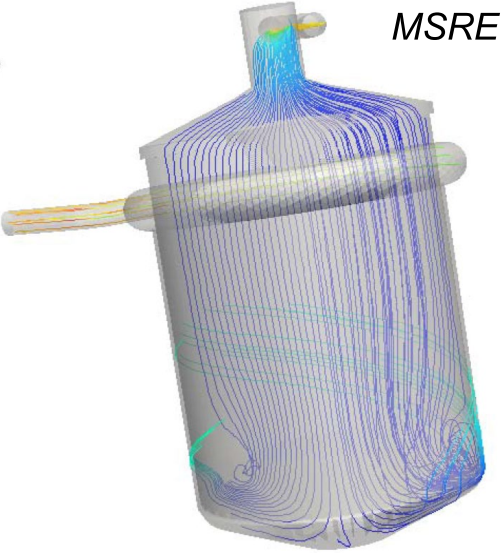
MSRE

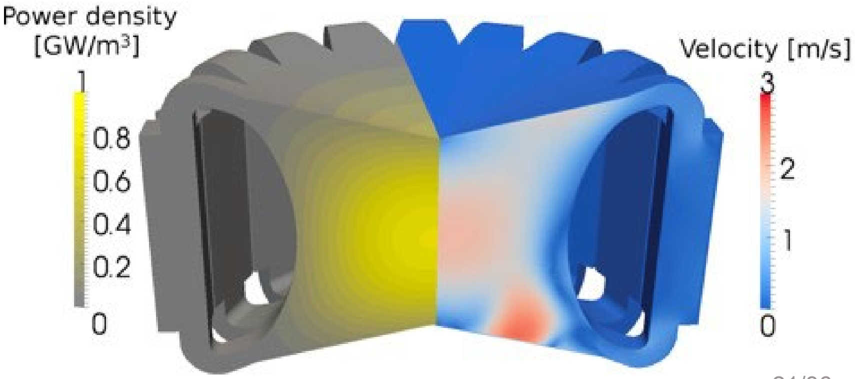
MSFR



MSR modelling (M. Aufiero -> PoliMi + CNRS / GeN-Foam)

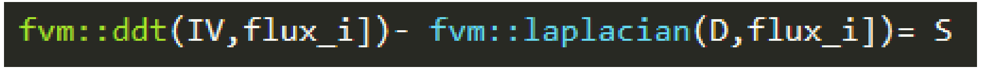
- ❑ Available CFD solvers
- ❑ Arbitrary geometries
- ❑ Streamlined implementation of diffusion and DNP equations





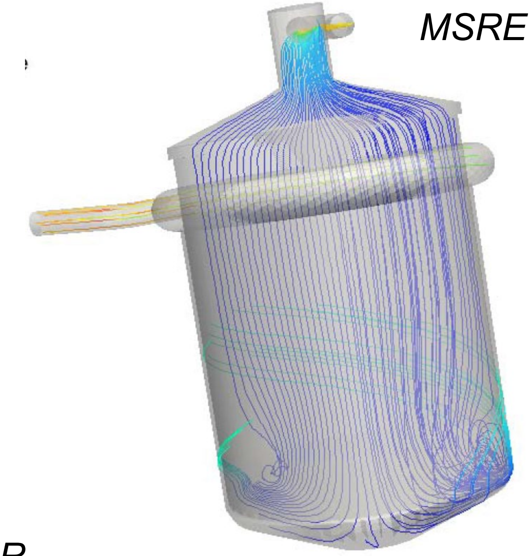
MSR modelling (M. Aufiero -> PoliMi + CNRS / GeN-Foam)

- Available CFD solvers
- Arbitrary geometries
- Streamlined implementation of diffusion and DNP equations

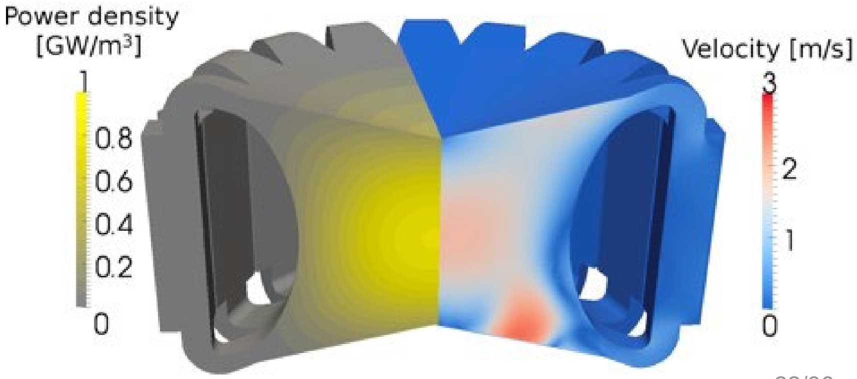


```
fvm::ddt(IV,flux_i])- fvm::laplacian(D,flux_i])= S
```

MSRE



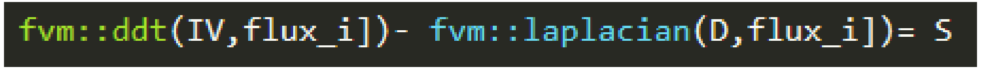
MSFR



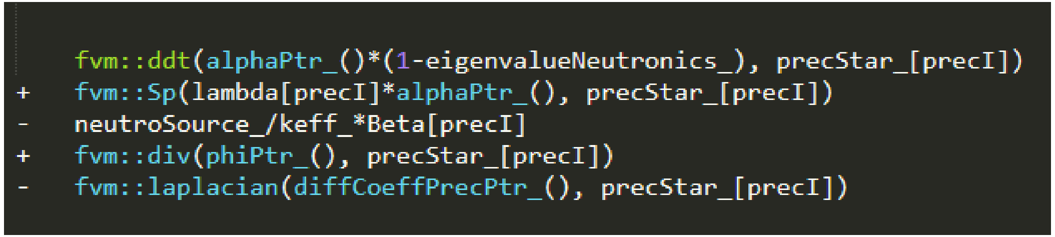
Carlo Fiorina

MSR modelling (M. Aufiero -> PoliMi + CNRS / GeN-Foam)

- Available CFD solvers
- Arbitrary geometries
- Streamlined implementation of diffusion and DNP equations

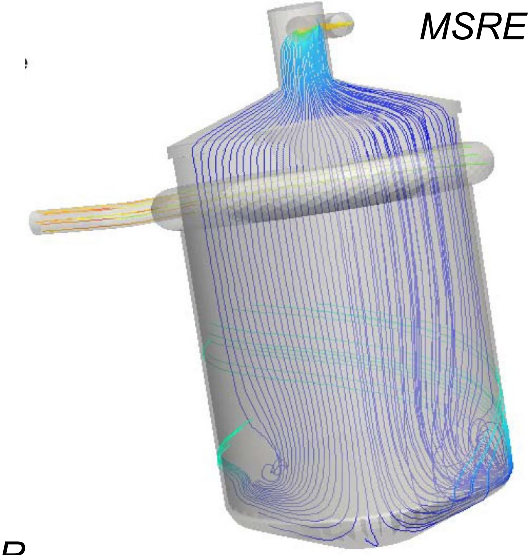


```
fvm::ddt(IV,flux_i])- fvm::laplacian(D,flux_i])= S
```

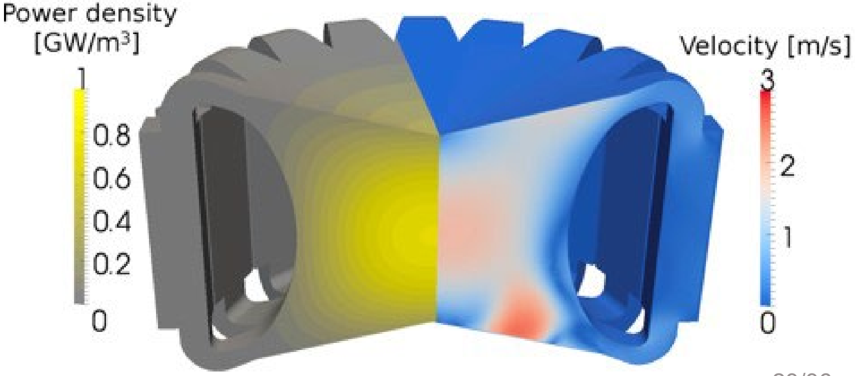


```
    fvm::ddt(alphaPtr_()*(1-eigenvalueNeutronics_), precStar_[precI])
+   fvm::Sp(lambda[precI]*alphaPtr_(), precStar_[precI])
-   neutroSource_/keff_*Beta[precI]
+   fvm::div(phiPtr_(), precStar_[precI])
-   fvm::laplacian(diffCoeffPrecPtr_(), precStar_[precI])
```

MSRE



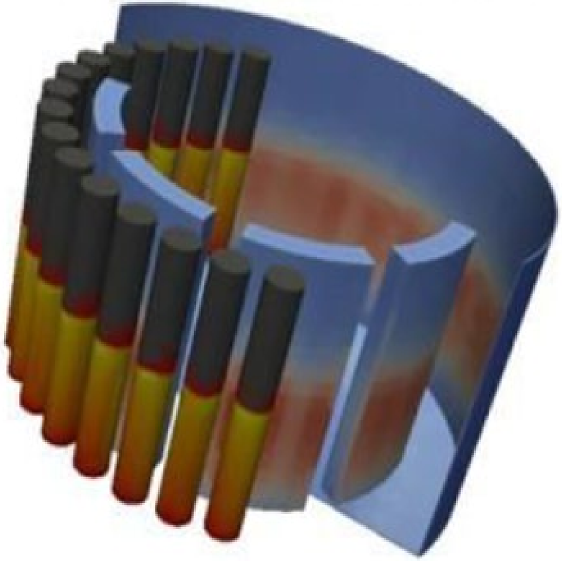
MSFR

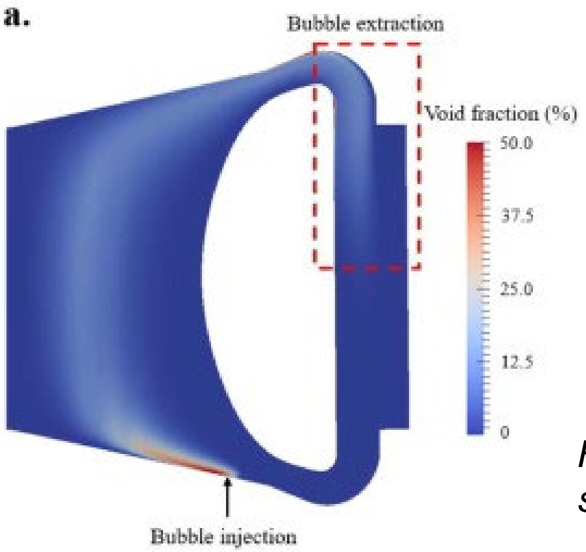


MSR modelling: advanced

- ❑ Available two-phase CFD solvers
- ❑ Radiative heat transfer
- ❑ Thermal-mechanics and moving mesh
- ❑ …

Dump tanks



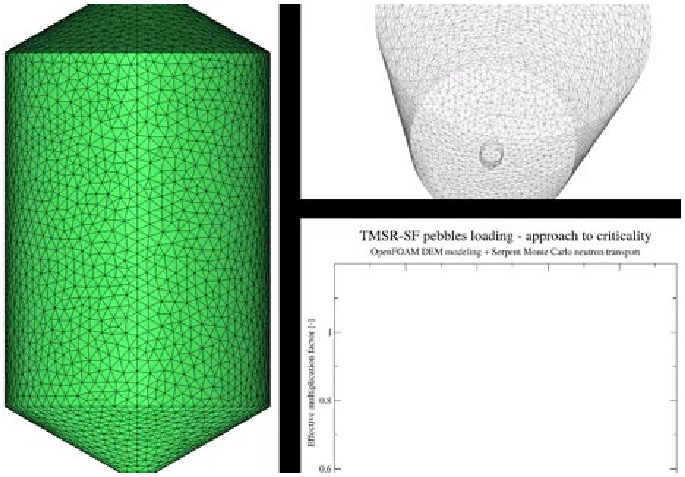


Helium sparging

Carlo Fiorina

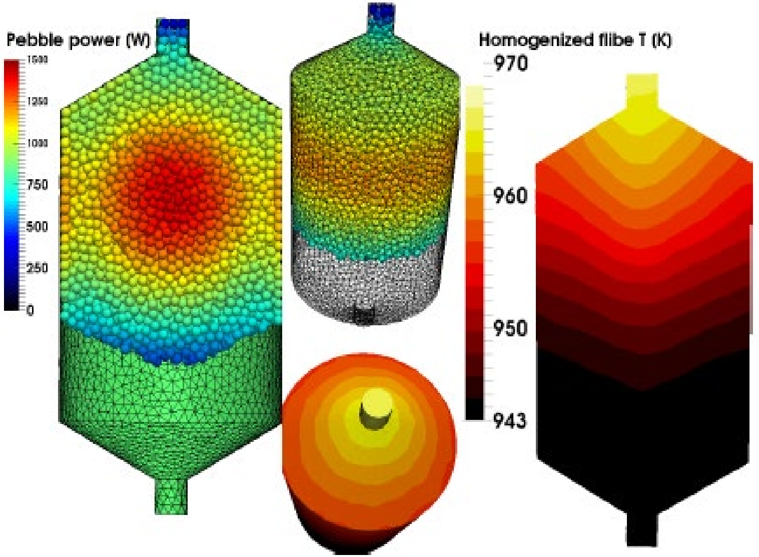
FHR modelling (UCB)

- Discrete Element Method + coarse-mesh thermal-hydraulics + Serpent Multi-physics interface



TMSR-SF pebbles loading - approach to criticality
OpenFOAM DEM modeling + Serpent Monte Carlo neutron transport
Effective multiplication factor [-]
1
0.8
0.6

Helium sparging

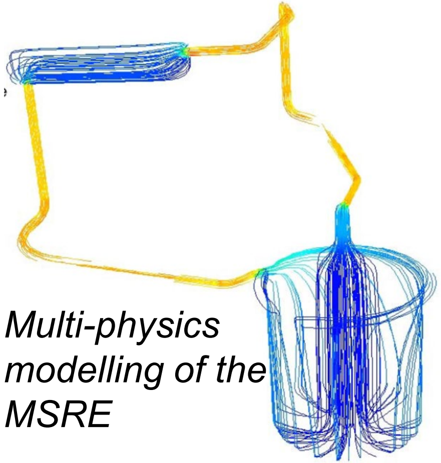


Dump tanks

Carlo Fiorina

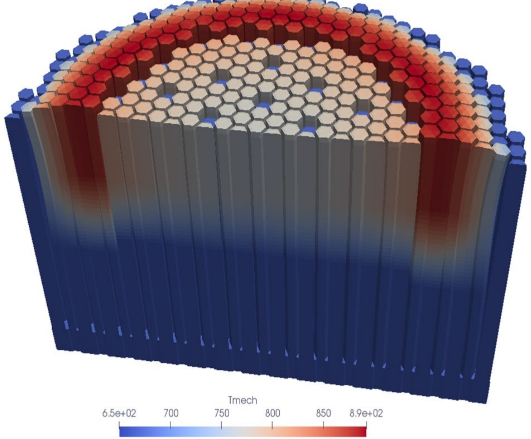
GeN-Foam: Generalized Nuclear Field operation and manipulation

- ❑ **First general solver for reactor safety based on OpenFOAM**

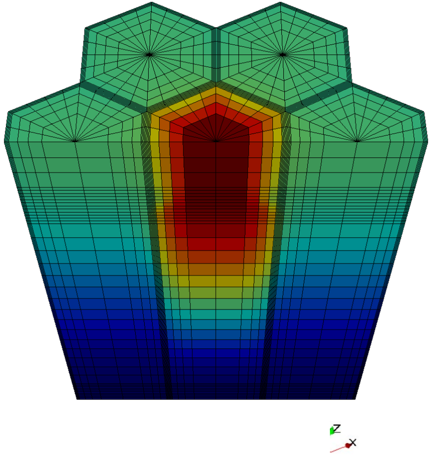


Multi-physics modelling of the MSRE

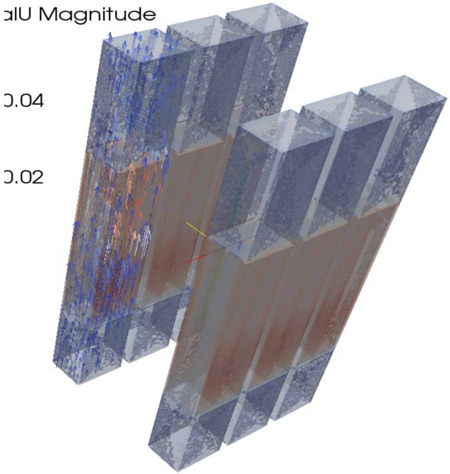
Core flowering in a SFR



Assembly windows in a SFR



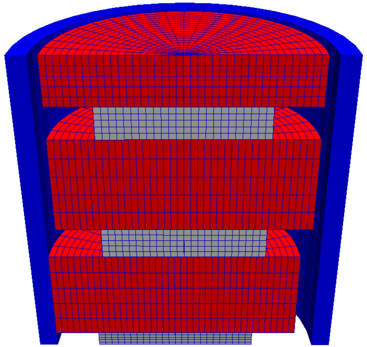
The Argonaut reactor

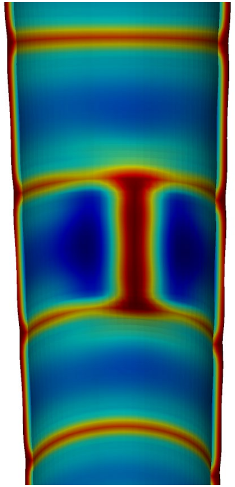


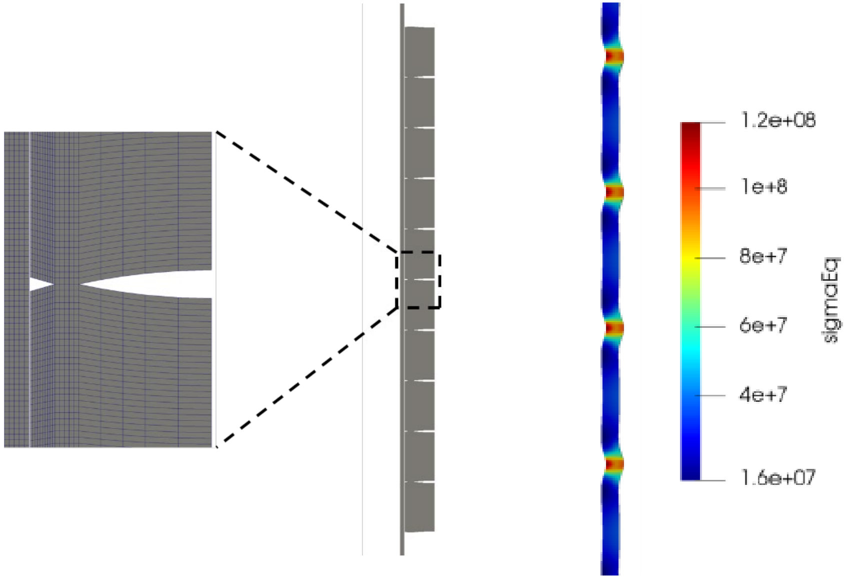
- ❑ Open-source + object -> use of previous work
- ❑ CFD solvers
- ❑ Thermal-mechanics solvers
- ❑ Multi-mesh with projection algorithms
- ❑ Multi-material
- ❑ Mesh deformations
- ❑ ….

OFFBEAT: OpenFoam Fuel BEhavior Tool

- ❑ **Thermal-mechanics with finite volumes….**





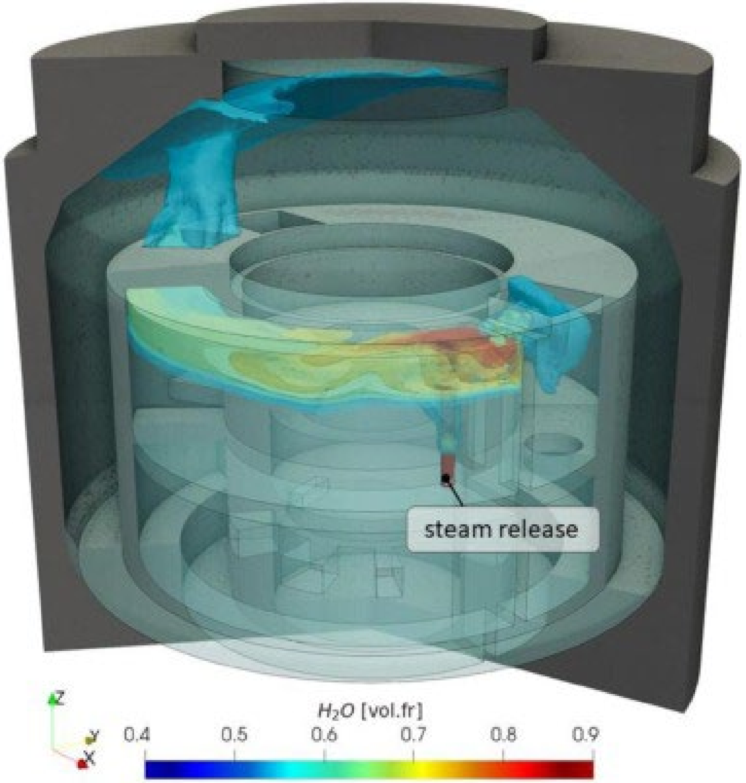


- ❑ Community contributions
- ❑ Region-coupled boundaries
- ❑ Multi-material
- ❑ …

HPC-oriented containment analysis - containmentFoam

From a general CFD tool to a nuclear-dedicated solver

- ❑ Available solvers (incl. Monte Carlo!)
- ❑ Turbulent models
- ❑ Conservative formulation
- ❑ Parallel scalability
- ❑ …



ISP-37 VANAM-M3 experiment
with containmentFOAM

Lessons learnt

One can model pretty much everything…

Lessons learnt

What's the effort?

What competences do I need?

What about the license?

What is the quality of the result?

Workflow

Carlo Fiorina

Downsides

- No graphical user interface (distributed with the code)
- Meshing and post-processing are performed with separate tools
- Meshing often requires proprietary tools
- Requires familiarity with Linux
- Limited documentation

Advantages

- Transparent
- Access to source code

Better integrations of application and development

Structure of the base library

Very complete

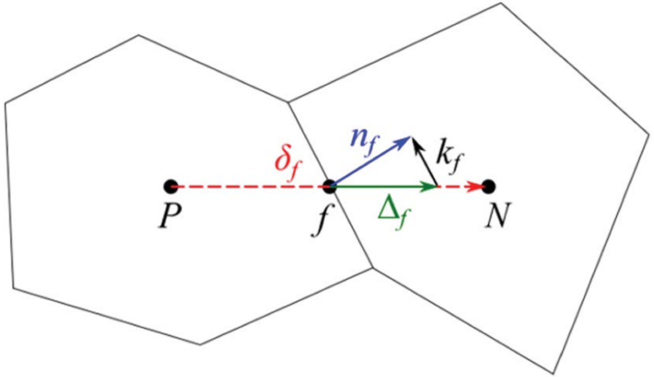
- discretization and linear system solution
- mesh-to-mesh projections
- mesh deformation
- mesh manipulation
- ordinary differential equations
- Monte Carlo methods
- octree-based mesh search
- methods for reduced-order modelling
- built-in and third-party code coupling schemes
- …

Object oriented

- encapsulation
- multi-level API

Finite volumes

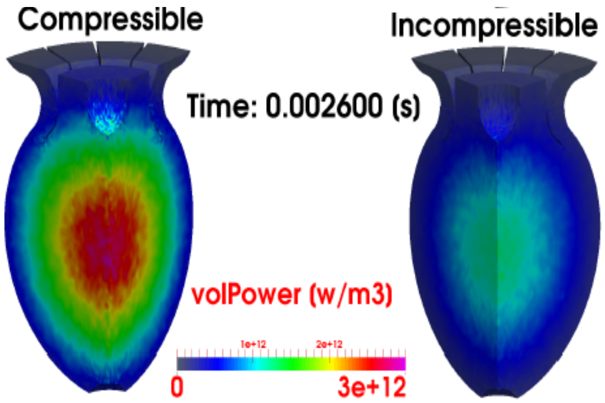
- ❑ Pros:
  - ✓ Flexible
  - ✓ Scalable
  - ✓ Conservative
  - ✓ Intuitive
  - ✓ CFD-friendly
  - ✓ Good for thermal-mechanics
  - ✓ Ok for neutronics

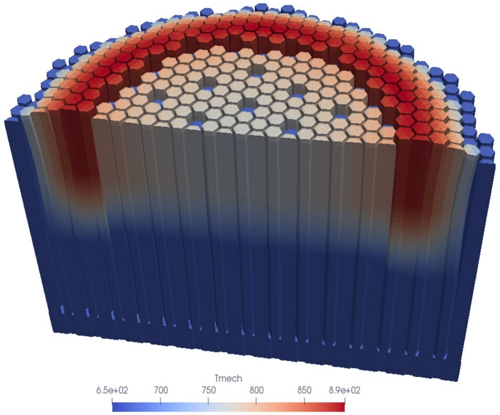


- ❑ Cons:
  - ✓ Still require familiarity with concepts associated with PDEs (well-posed problems, initial and boundary conditions), geometry creation, meshing, discretization, linear solution, etc.
  - ✓ Require good quality meshes
  - ✓ Max second order in space

Unstructured meshes

- Complete flexibility in terms of geometry -> non-traditional reactor designs and complex component
- Significant computational footprint
- First order, with all cell faces that are flat -> a high mesh resolution for curved surfaces





Operator-splitting

One matrix for each equation + iteration

Pros

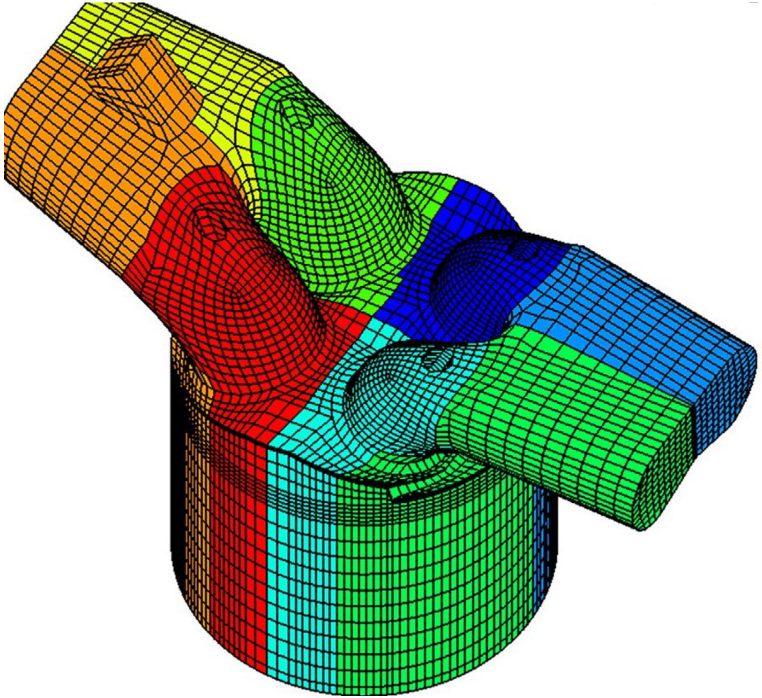
- Easier preconditioning and optimal choice of solution method
- No need to solve all physics at each coupling/time step

Cons

- Can be hard to converge for “weakly-coupled” / strongly non-linear equations

Parallelization

- Domain decomposition and the MPI
- Optimally scale up to few thousands of CPU cores
- Some bottlenecks
  - ✓ the sub-optimal sparse matrices storage format (LDU) that does not enable any cache-blocking mechanism (SIMD, vectorization)
  - ✓ the I/O data storage system
- The OpenFOAM HPC Technical Committee is currently working on the limitations
  - ✓ interface to external linear algebra libraries
  - ✓ recent work from NVIDIA



Computational requirements

CPU cores

- rule of thumb: 30’000 mesh cells per CPU core
- CFD
  - 2D RANS-> several hundred thousand cells -> 10 CPU cores
  - 3D RANS -> several hundred millions cells -> 5000 CPU cores
- coarse-mesh thermal-hydraulics and neutron diffusion
  - full-core models -> few hundred thousand to few million cells -> workstations or laptops

Runtime

- Steady-state simulations on the optimal number of CPU cores: several minutes to several hours
- Long-running time-dependent problems: up to a week
- In some specific applications, such as detailed containment simulations: up to a month

Memory requirements

- Single-phase RANS CFD simulation -> order of 10 fields -> 1 GB of memory per million cells
- 3D discrete ordinates -> several thousand solution fields -> 200 GB of memory per million cells

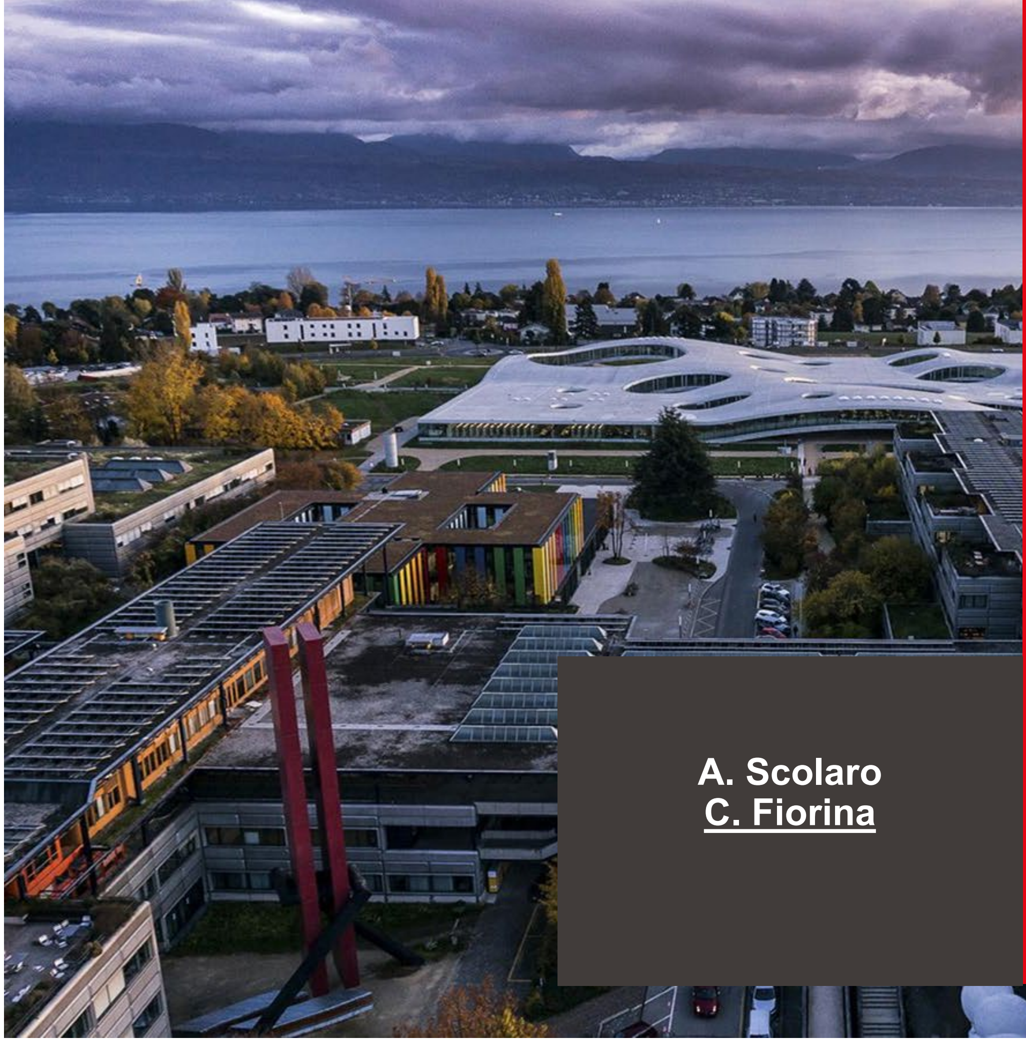
License

- GNU GPLv3 license
  - copyleft type license: automatically affect derivative work
  - favors a collaborative development with minimal work duplication
  - limits investments from commercial players

EPFL

Thank you

Carlo Fiorina



Introduction to OpenFOAM

A. Scolaro
C. Fiorina

École
polytechnique
fédérale
de Lausanne

I am curious about OpenFOAM … but which version?

OpenFOAM®

openfoam.***com***

OpenFOAM

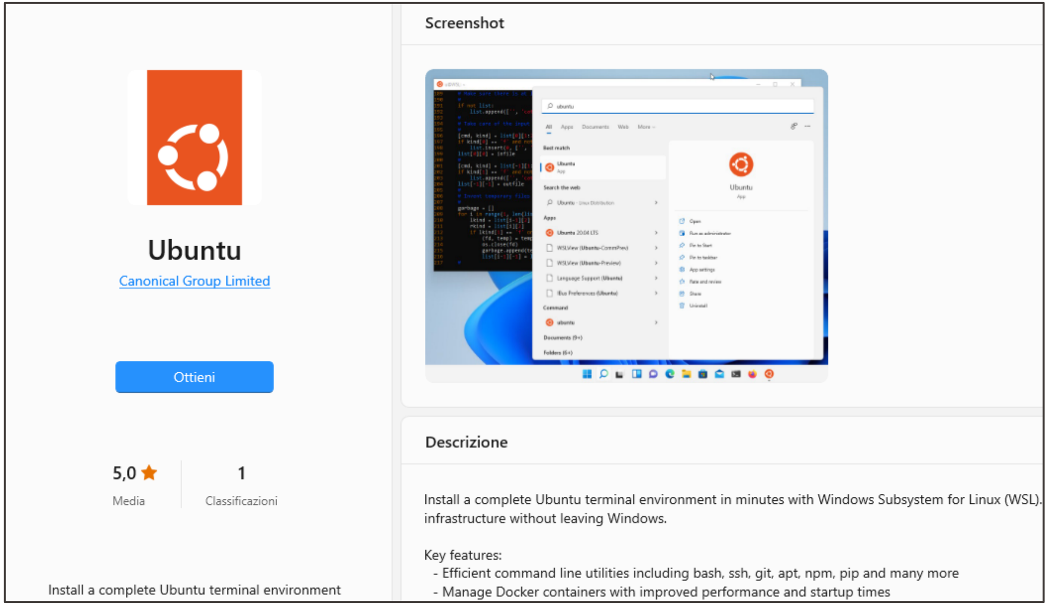
openfoam.***org***

IMPORTANT!

If you want to use an available solver, or take features from available solvers for your own solver, be very careful and select the right OF version!

Can I use it on my computer?

OpenFOAM runs natively on Linux systems…



Ubuntu

Canonical Group Limited

Ottieni

5,0 ★ Media

1 Classificazioni

Install a complete Ubuntu terminal environment

Screenshot

Descrizione

Install a complete Ubuntu terminal environment in minutes with Windows Subsystem for Linux (WSL).
infrastructure without leaving Windows.

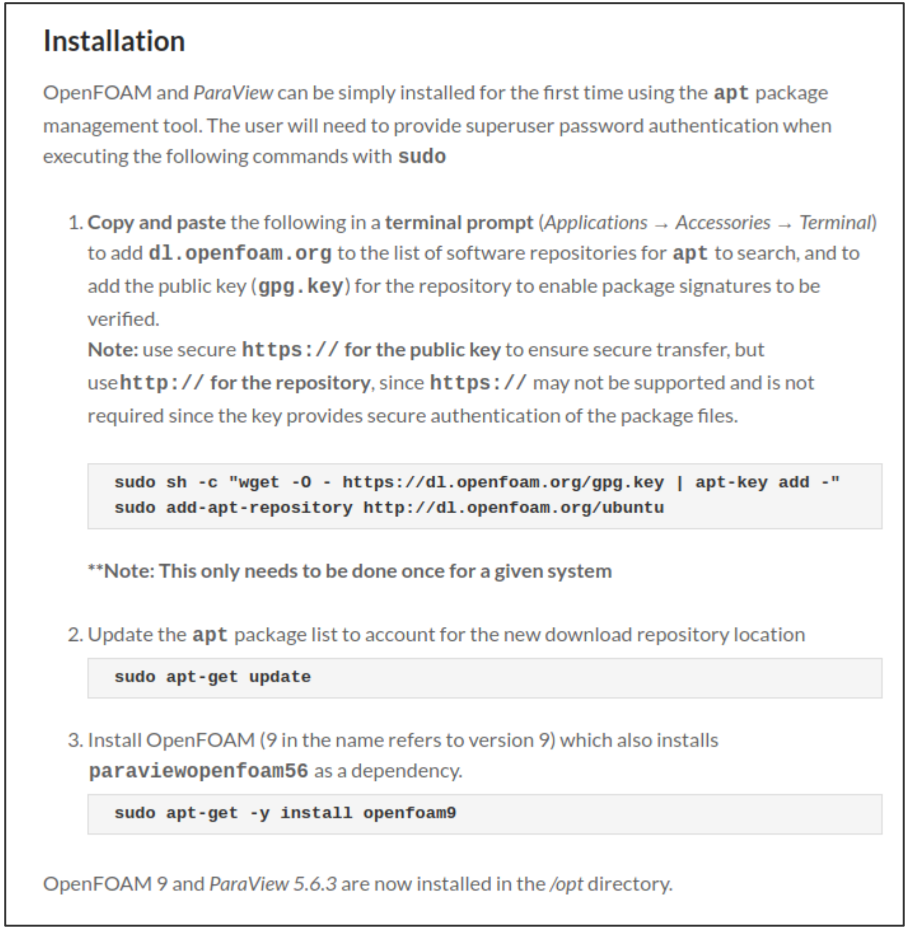
Key features:
- Efficient command line utilities including bash, ssh, git, apt, npm, pip and many more
- Manage Docker containers with improved performance and startup times

MAC, or the Linux subsystem for Windows can be used, but **not recommended by the presenter**

How to get OpenFOAM?

Follow the simple steps on the download page

(example for OF-9 from the .org version)



Installation

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

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

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

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

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

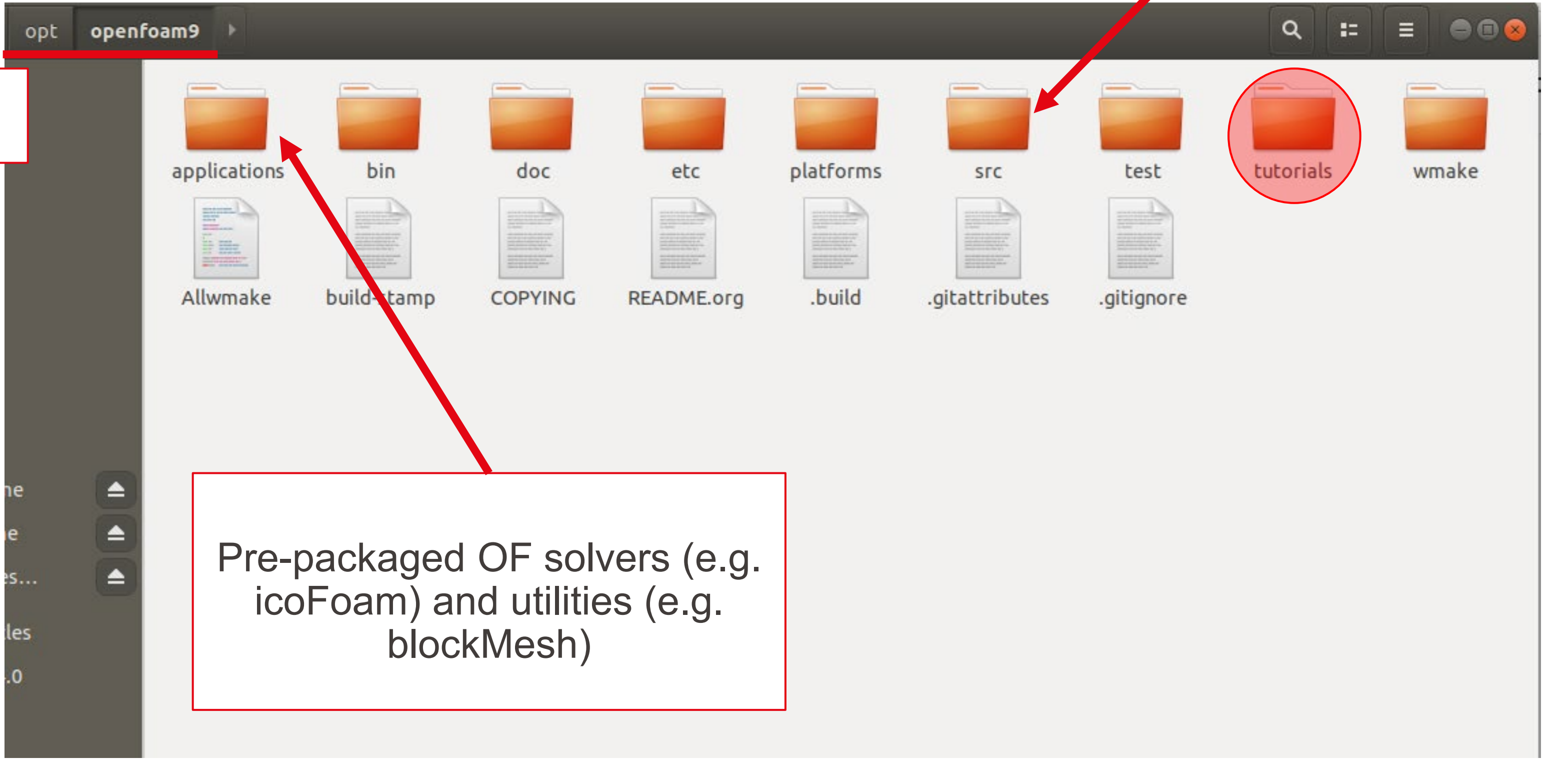
   ```
   sudo apt-get update
   ```

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

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

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

What comes with OpenFOAM?



Main OF library

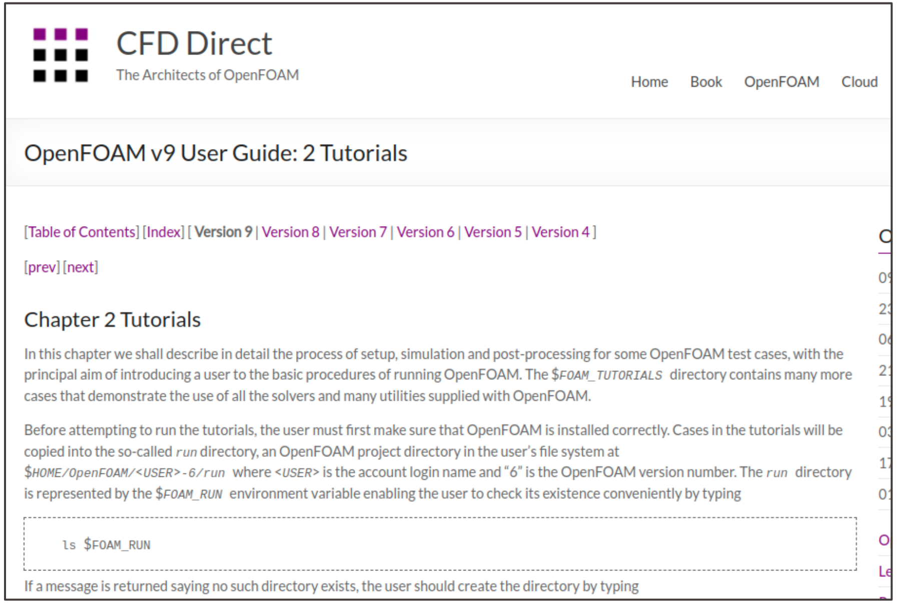
Typical location

Pre-packaged OF solvers (e.g. icoFoam) and utilities (e.g. blockMesh)

Learn OpenFOAM - Official documentation

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

It includes some post-processing examples



OpenFOAM v9 User Guide: 2 Tutorials

[Table of Contents] [Index] [Version 9 | Version 8 | Version 7 | Version 6 | Version 5 | Version 4]

[prev] [next]

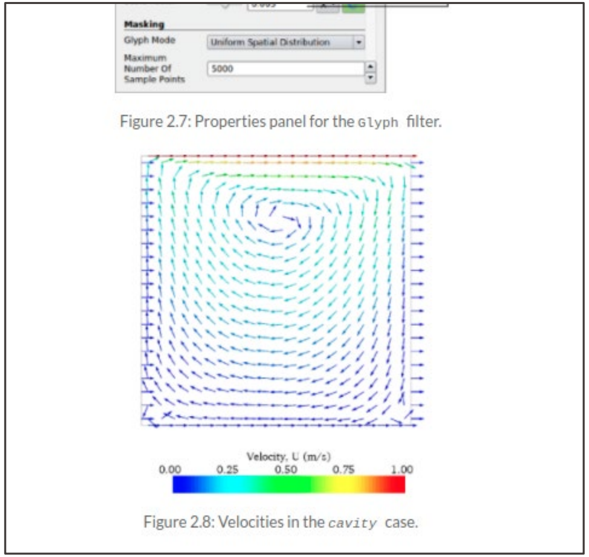
Chapter 2 Tutorials

In this chapter we shall describe in detail the process of setup, simulation and post-processing for some OpenFOAM test cases, with the principal aim of introducing a user to the basic procedures of running OpenFOAM. The $FOAM_TUTORIALS directory contains many more cases that demonstrate the use of all the solvers and many utilities supplied with OpenFOAM.

Before attempting to run the tutorials, the user must first make sure that OpenFOAM is installed correctly. Cases in the tutorials will be copied into the so-called *run* directory, an OpenFOAM project directory in the user's file system at $HOME/OpenFOAM/<USER>-6/run where <USER> is the account login name and "6" is the OpenFOAM version number. The *run* directory is represented by the $FOAM_RUN environment variable enabling the user to check its existence conveniently by typing

```
ls $FOAM_RUN
```

If a message is returned saying no such directory exists, the user should create the directory by typing



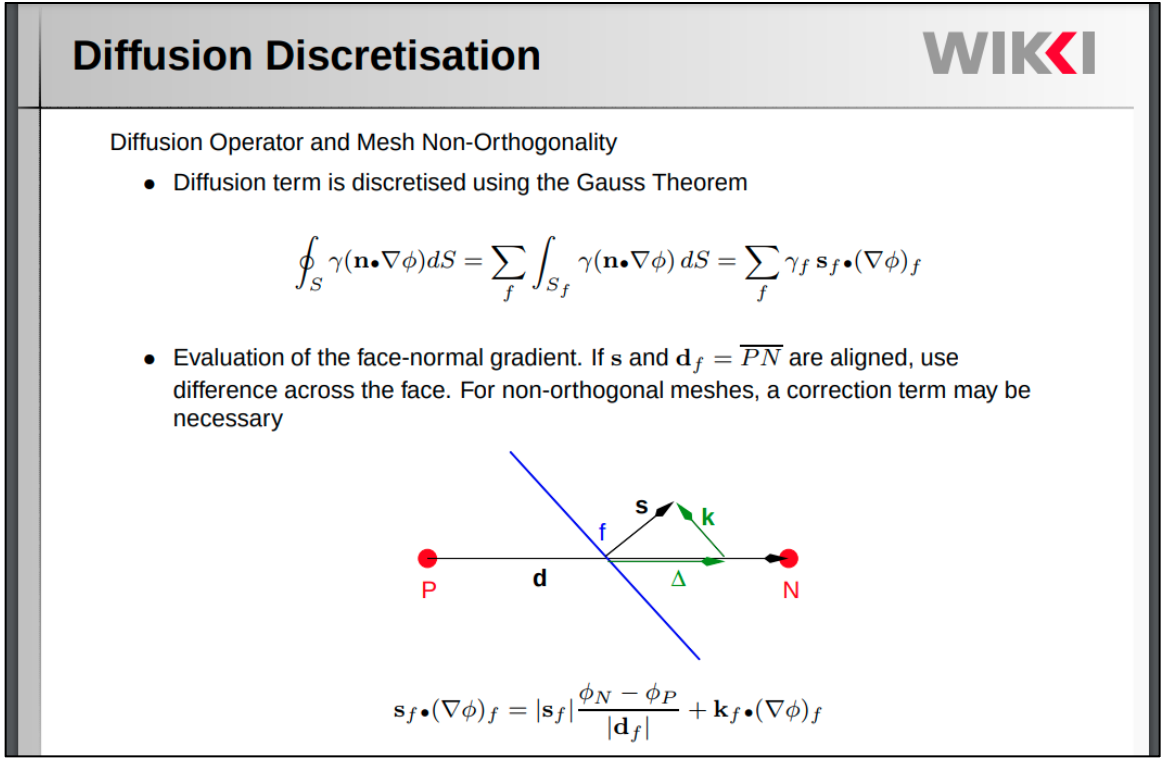
Masking
Glyph Mode: Uniform Spatial Distribution
Maximum Number Of Sample Points: 5000

Figure 2.7: Properties panel for the Glyph filter.

Figure 2.8: Velocities in the cavity case.

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

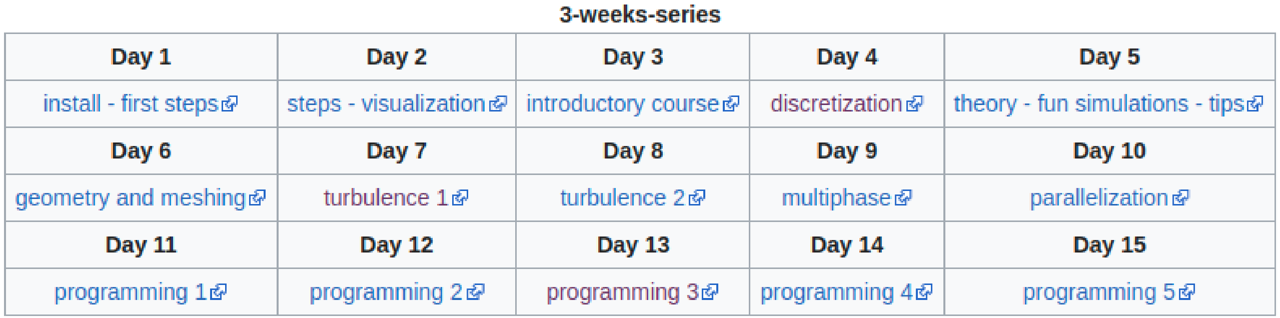
$$\oint_S \gamma(\mathbf{n}\bullet\nabla\phi)dS = \sum_f \int_{S_f} \gamma(\mathbf{n}\bullet\nabla\phi)\, dS = \sum_f \gamma_f\, \mathbf{s}_f\bullet(\nabla\phi)_f$$

- 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 - Take your time, follow the “3 weeks” series

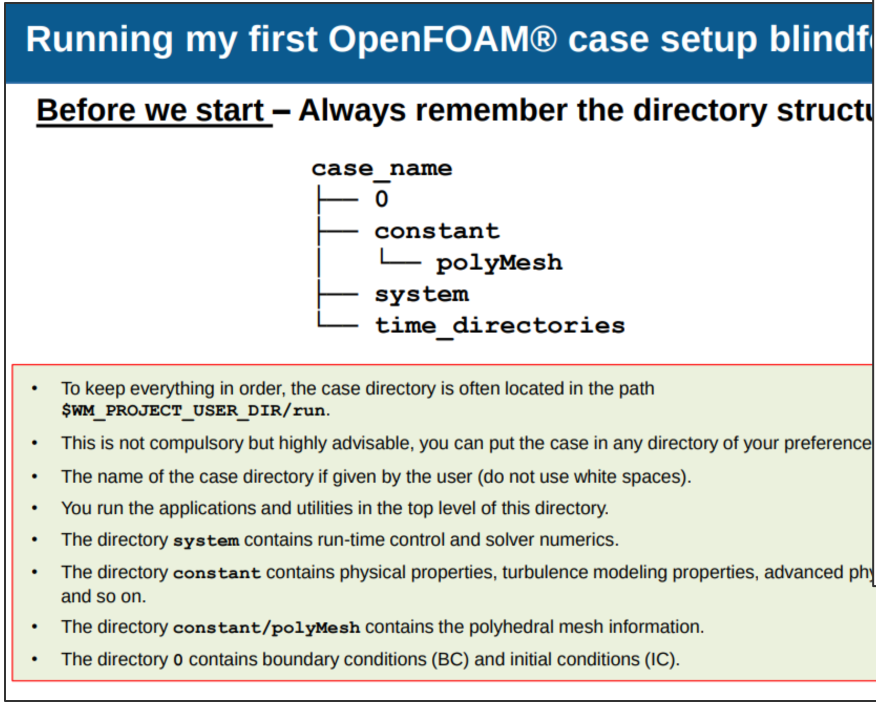
https://wiki.openfoam.com/index.php?title=%223_weeks%22_series



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 - Presentations from Wolf Dynamics

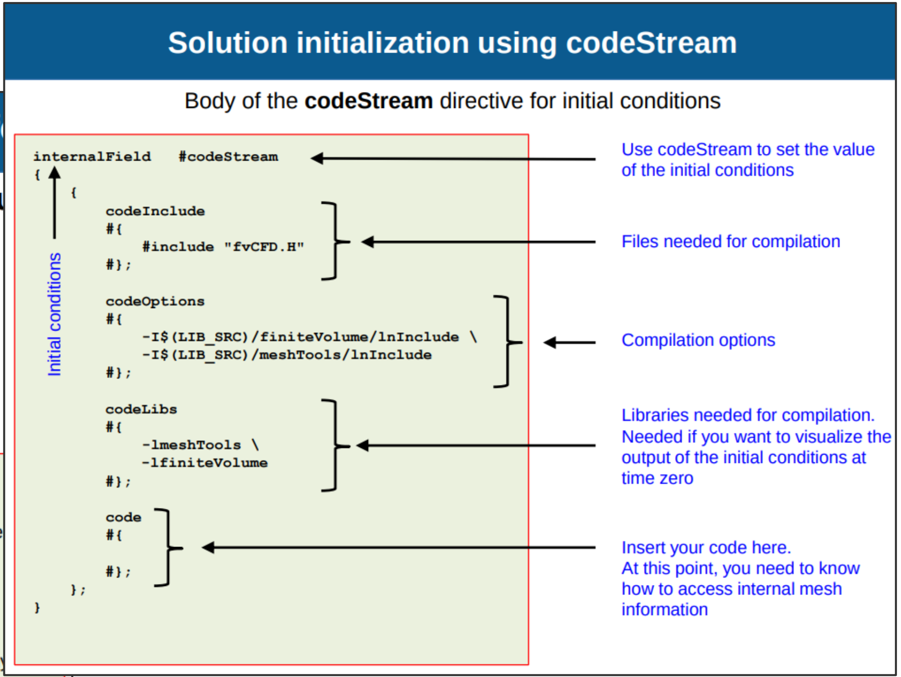


Running my first OpenFOAM® case setup blindf

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
        #{

        #};
    };
}
```

Initial conditions

Use codeStream to set the value of the initial conditions

Files needed for compilation

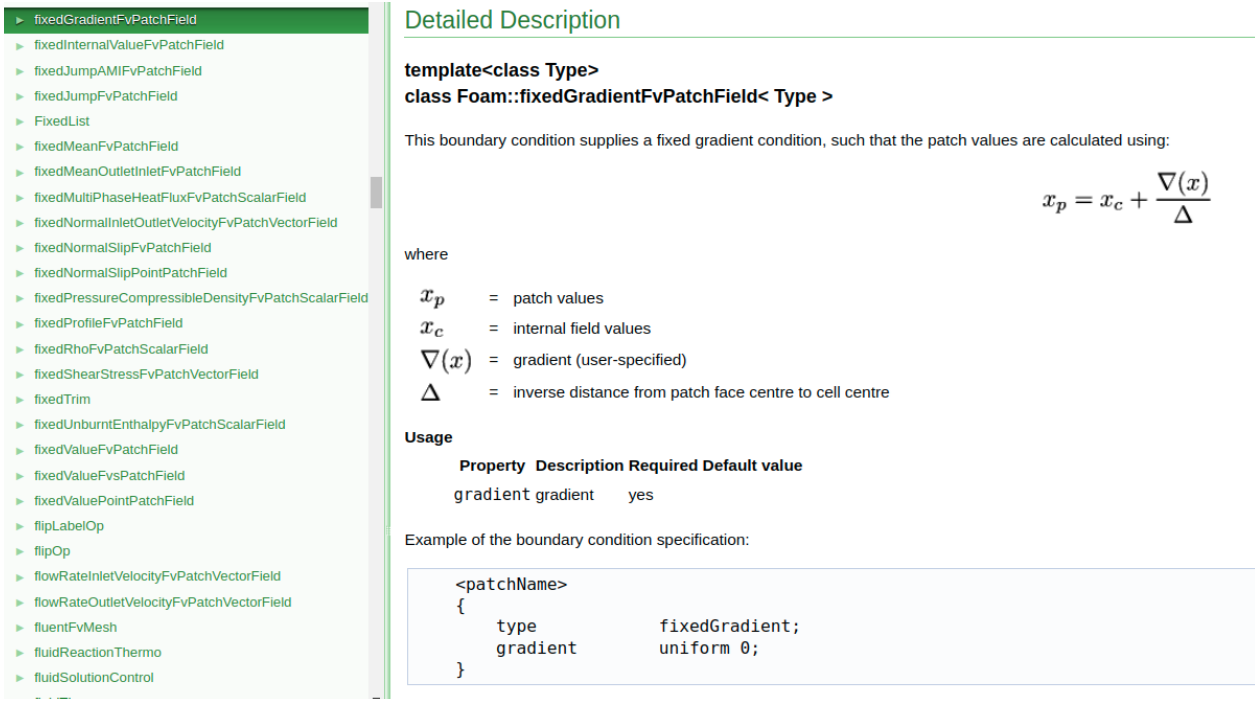
Compilation options

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 - Browse the C++ source guide official documentation

- https://www.openfoam.com/documentation/guides/v2112/doc/
- https://cpp.openfoam.org/v9/



- ▸ fixedGradientFvPatchField
- ▸ fixedInternalValueFvPatchField
- ▸ fixedJumpAMIFvPatchField
- ▸ fixedJumpFvPatchField
- ▸ FixedList
- ▸ fixedMeanFvPatchField
- ▸ fixedMeanOutletInletFvPatchField
- ▸ fixedMultiPhaseHeatFluxFvPatchScalarField
- ▸ fixedNormalInletOutletVelocityFvPatchVectorField
- ▸ fixedNormalSlipFvPatchField
- ▸ fixedNormalSlipPointPatchField
- ▸ fixedPressureCompressibleDensityFvPatchScalarField
- ▸ fixedProfileFvPatchField
- ▸ fixedRhoFvPatchScalarField
- ▸ fixedShearStressFvPatchVectorField
- ▸ fixedTrim
- ▸ fixedUnburntEnthalpyFvPatchScalarField
- ▸ fixedValueFvPatchField
- ▸ fixedValueFvsPatchField
- ▸ fixedValuePointPatchField
- ▸ flipLabelOp
- ▸ flipOp
- ▸ flowRateInletVelocityFvPatchVectorField
- ▸ flowRateOutletVelocityFvPatchVectorField
- ▸ fluentFvMesh
- ▸ fluidReactionThermo
- ▸ fluidSolutionControl

Detailed Description

template<class Type>
class Foam::fixedGradientFvPatchField< Type >

This boundary condition supplies a fixed gradient condition, such that the patch values are calculated using:

$$x_p = x_c + \frac{\nabla(x)}{\Delta}$$

where

x_p	=	patch values
x_c	=	internal field values
$\nabla(x)$	=	gradient (user-specified)
Δ	=	inverse distance from patch face centre to cell centre

Usage

Property	Description	Required	Default value
`gradient`	gradient	yes	

Example of the boundary condition specification:

```
<patchName>
{
    type            fixedGradient;
    gradient        uniform 0;
}
```

Learn OpenFOAM - Plenty of additional resources

- Tutorials/lectures (have a look on Google or YouTube)
- Master/PhD thesis etc.
- Forums
- (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!**

Thank you

A. Scolaro
C. Fiorina

Approaching the "nuclear" solvers

Carlo Fiorina

École polytechnique fédérale de Lausanne

Background: Some essential features of OpenFOAM

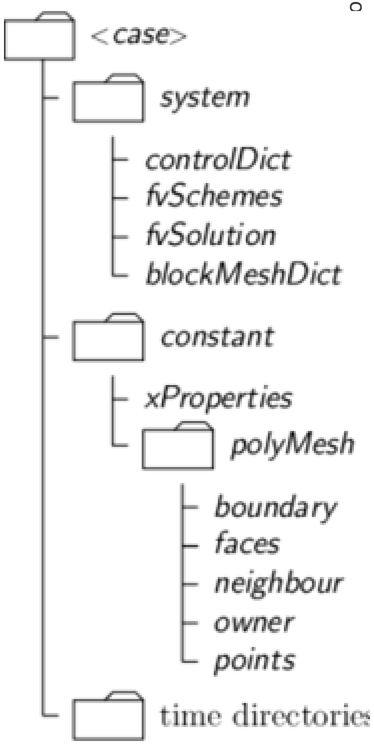
- Workflow divided in 4 distinct steps
  - Mesh creation
  - Input data and mesh are gathered inside a Case Folder
  - Running
  - Post-processing

Background: Some essential features of OpenFOAM - Mesh creation

- Don’t take it lightly:
  - one of the most time consuming steps
  - requires good understanding of methods to decide the type of mesh and its refinement
  - a bad mesh will give a bad solution (especially for CFD)
  - in some unlucky cases, a bad mesh will give a non-convergent solution
- Several available free tools: blockMesh (embedded in OpenFOAM), Salome, gmsh, cfMesh, snappyHexMesh…
- Complex geometries and situations where high-quality mesh are needed may require the use of commercial software
- Make sure that the tool you chose allows you to separate your mesh into zones (called cellZones in OpenFOAM). They are necessary to assign different physical properties to different materials!

Background: Some essential features of OpenFOAM - Input data

- ❑ All data (inlc. mesh) that OpenFOAM needs are collected into a Case Folder
- ❑ Inside a case folder you'll find at least 3 sub-folders
  - ○ The folder "0", and possible other time directories, containing, for each field (viz., velocity, pressure, density):
    - Initial conditions
    - Boundary condition
  - ○ The folder "constant" containing:
    - the mesh
    - all physical properties, gathered into "dictionaries"
    - the types of models (for instance k-epsilon or k-omega for turbulence), alse gathered into "dictionaries"
  - ○ The folder "system" containing at least:
    - "controlDict" gathers main simulation parameters like initial time, time steps, final time, etc.
    - "fvSchemes" to set the type of discretization for various equations
    - "fvSolution" to set the parameters of the linear solvers

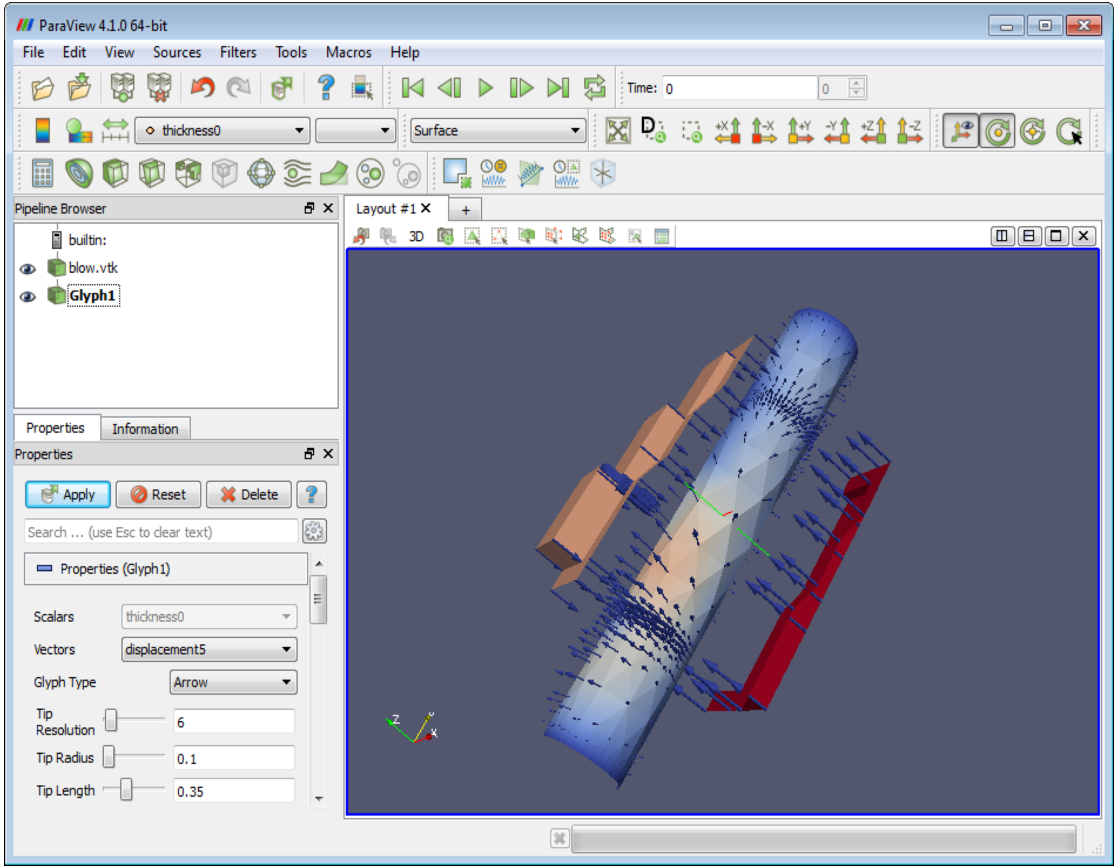


Background: Some essential features of OpenFOAM - Running

- Via command line:
  - “name of the solver”, such as: icoFoam, pimpleFoam or… GeN-Foam
- If parallel
  - decomposePar
  - mpirun -np “number of mpi processes” “name of the solver” -parallel
  - reconstructPar

Background: Some essential features of OpenFOAM - Post-processing

- Typically with paraview
- OpenFOAM also has some mechanisms to directly output, during or after simulation, specific quantities of interest



Time: 0

thickness0

Pipeline Browser:
- builtin:
- blow.vtk
- Glyph1

Scalars: thickness0

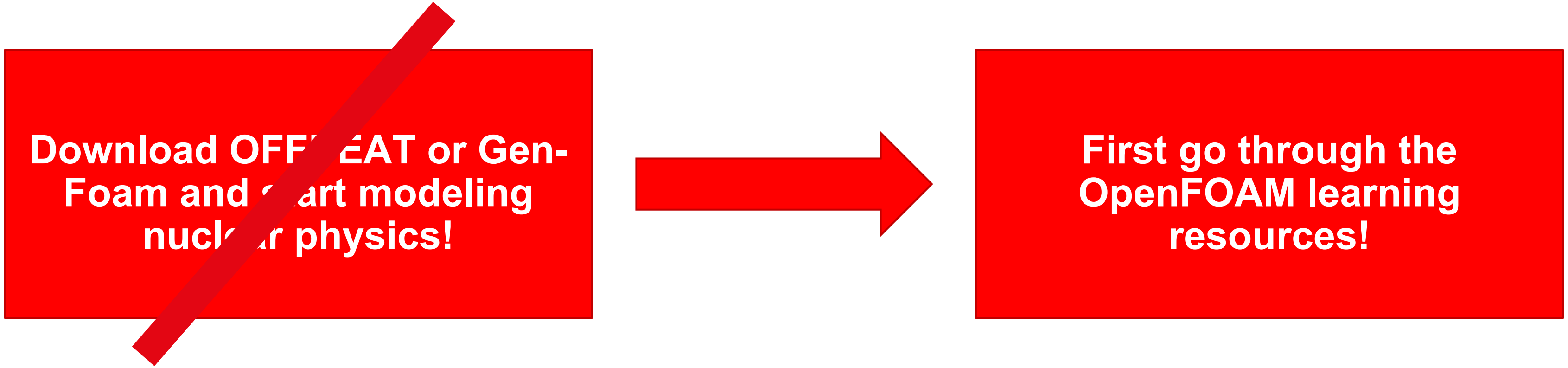
Vectors: displacement5

Glyph Type: Arrow

Tip Resolution: 6

Tip Radius: 0.1

Tip Length: 0.35



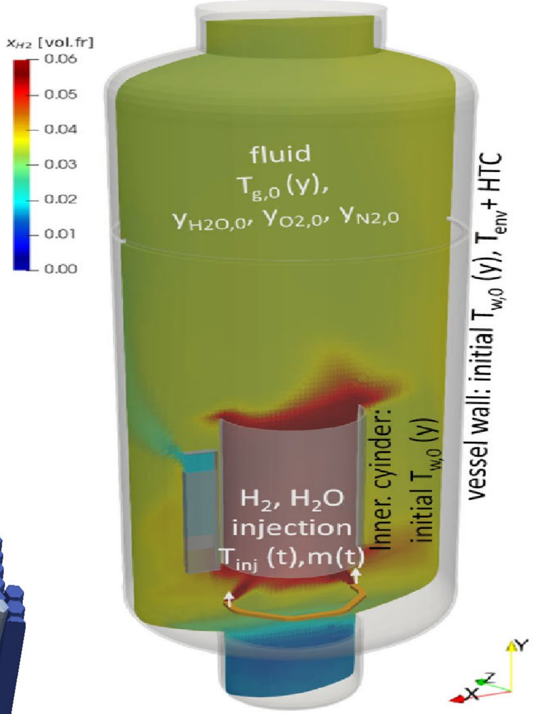
~~Download OFFBEAT or GeN-Foam and start modeling nuclear physics!~~

First go through the OpenFOAM learning resources!

Use of OpenFOAM for nuclear multi-physics

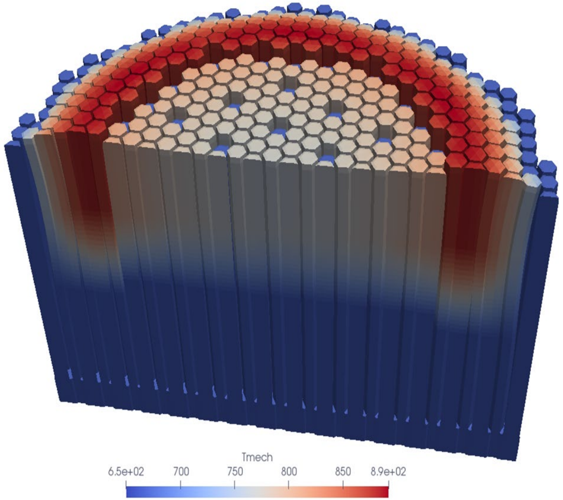
Carlo Fiorina

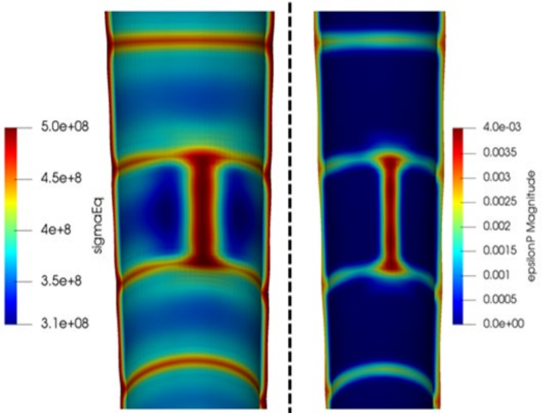
- Similar logic as other OpenFOAM solvers but
  - More complex
  - Typically multi-physics
  - Often multi-material
  - Sometimes multi-mesh



containmentFoam

GeN-Foam

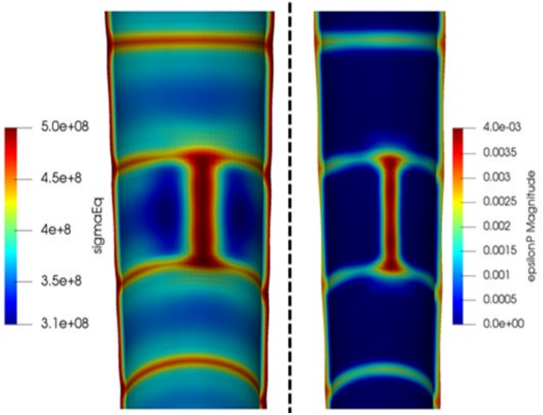




OFFBEAT

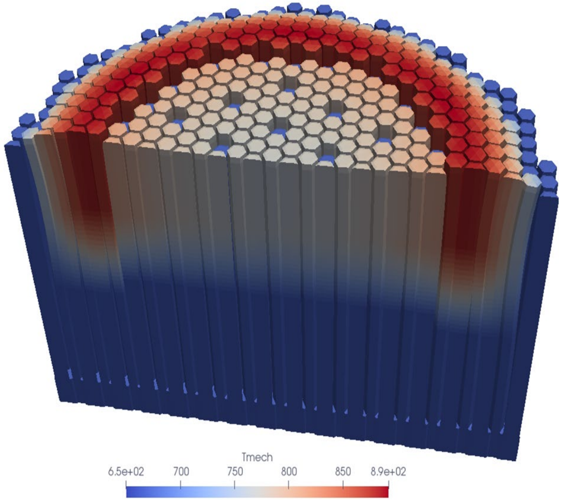
Use of OpenFOAM for nuclear multi-physics

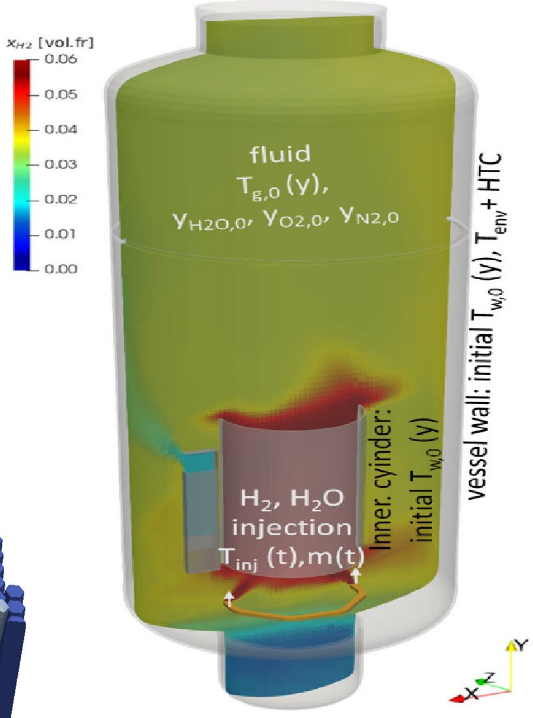
- Similar logic as other OpenFOAM solvers but
  - More complex
  - Typically multi-physics
  - **Often multi-material**
  - **Sometimes multi-mesh**



OFFBEAT

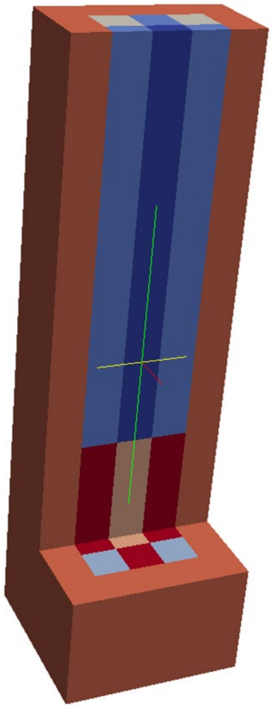
GeN-Foam



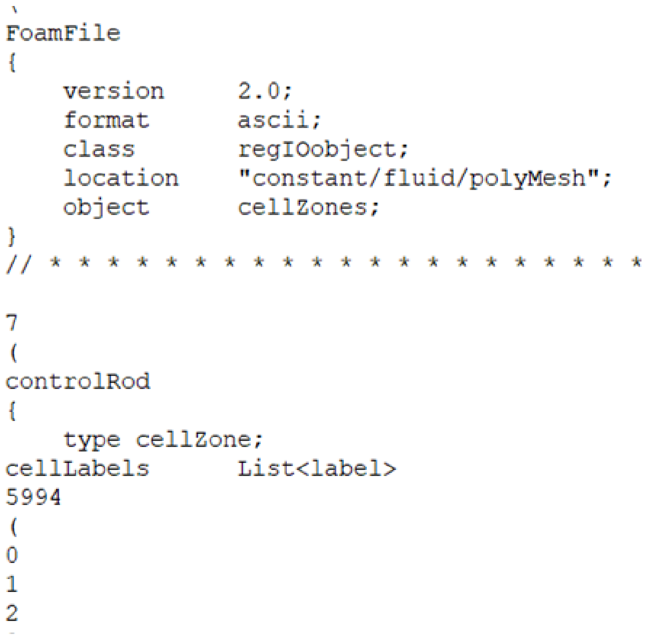


containmentFoam

Multi-material in OpenFOAM



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

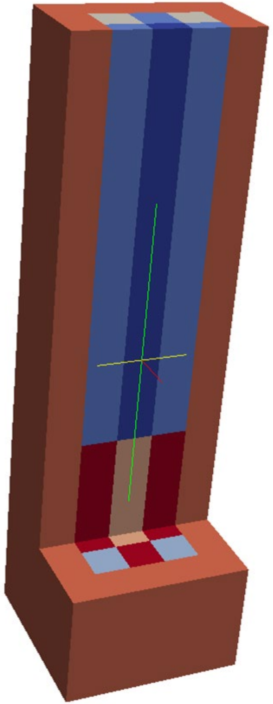


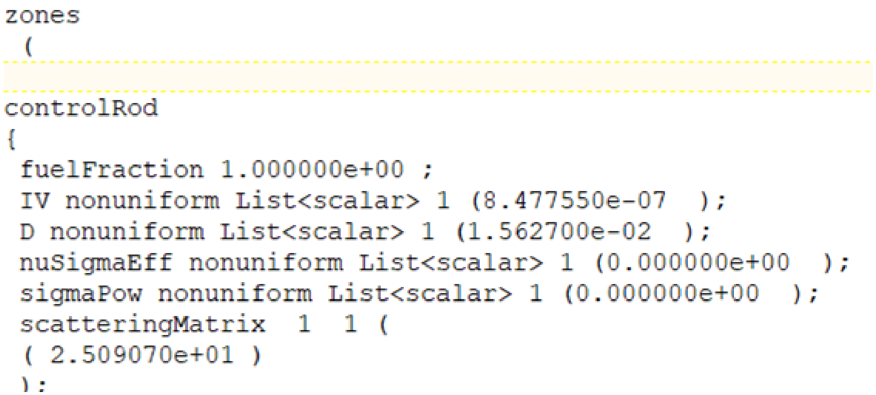
```
FoamFile
{
    version     2.0;
    format      ascii;
    class       regIOobject;
    location    "constant/fluid/polyMesh";
    object      cellZones;
}
// * * * * * * * * * * * * * * * * * * * * *

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

Multi-material in OpenFOAM

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





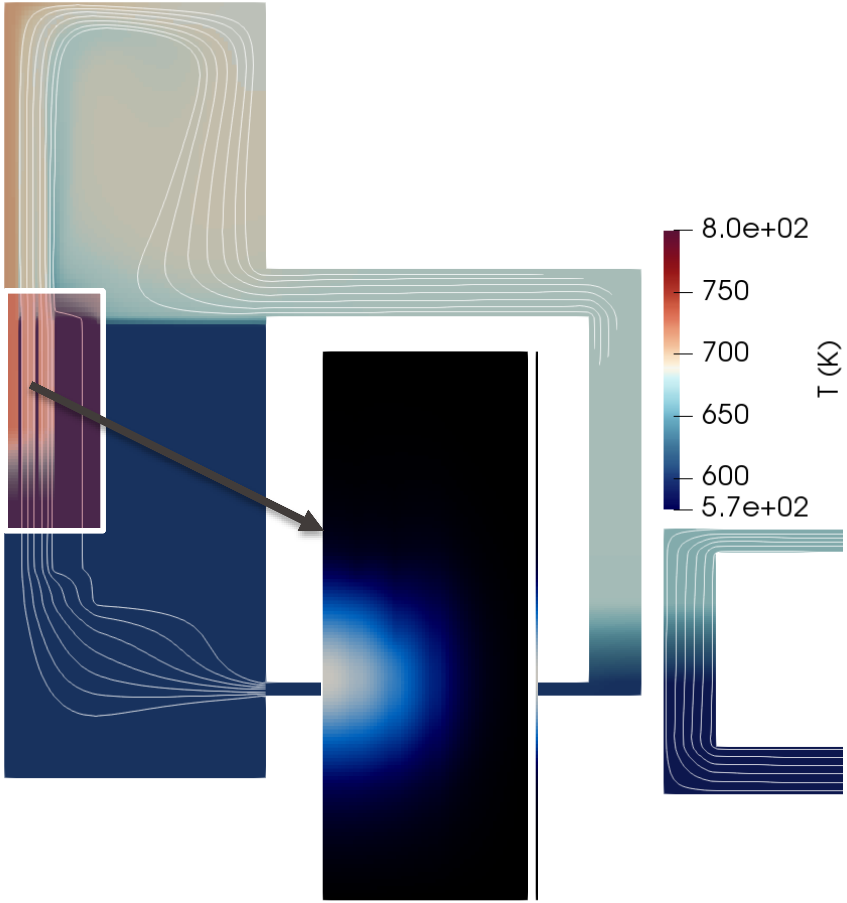
```
zones
 (

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 )
 );
```

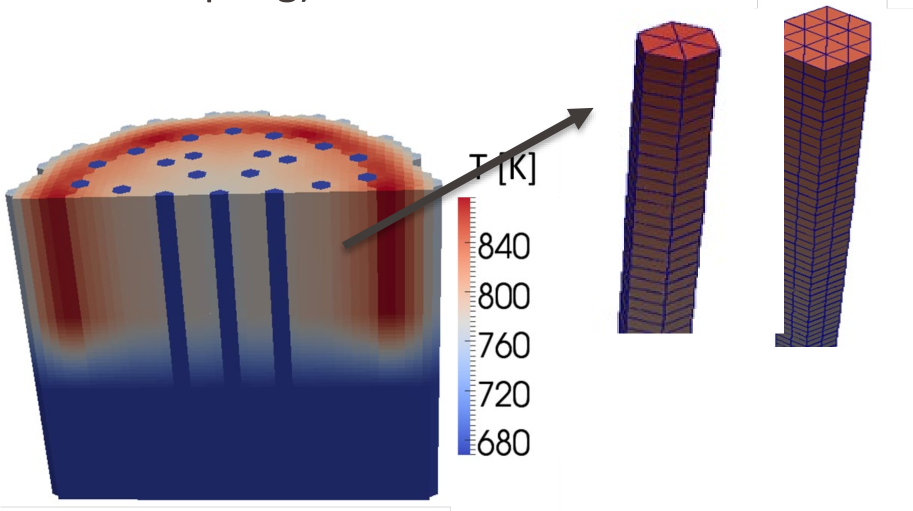
Multi-material in OpenFOAM: in practice

- **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 including cellZones (normally created automatically during mesh conversion)
  - Dictionaries that associates a cellZone to some value of a field or property

Multi-mesh in OpenFOAM



- Problem: 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)



Multi-mesh in OpenFOAM

```
• 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
        └ …
```

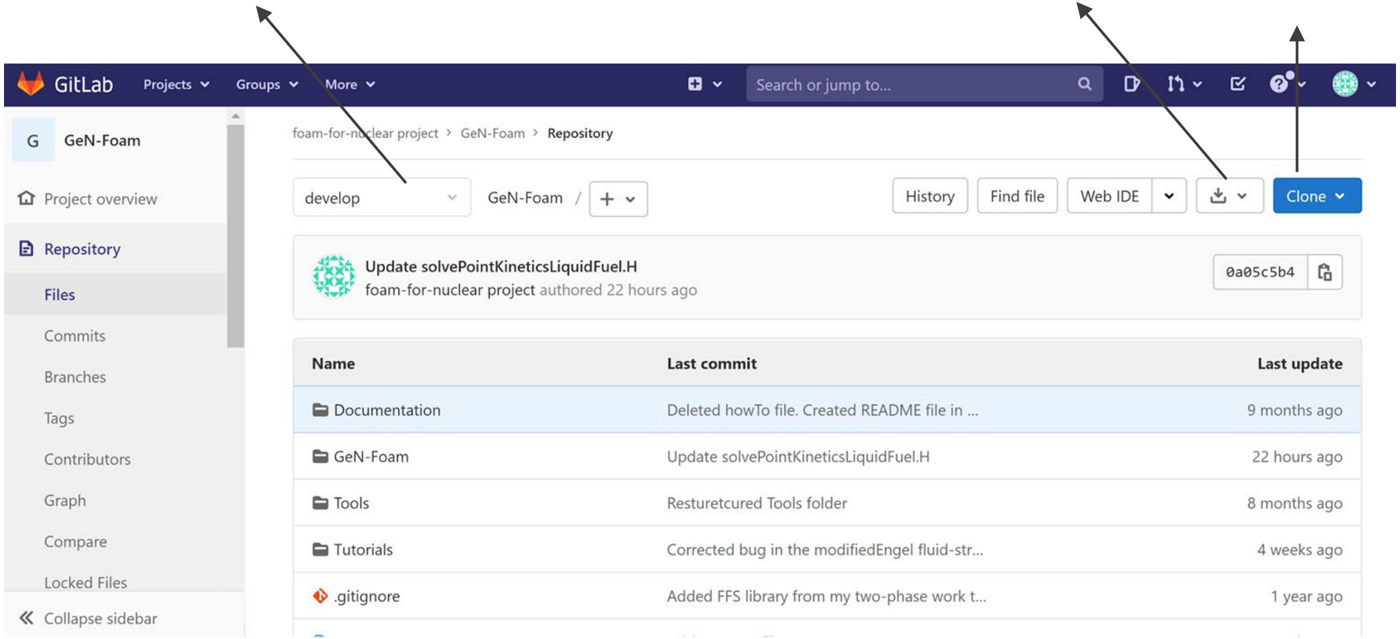
- Mesh-to-mesh projection to project from one mesh to the other

Carlo Fiorina

GeN-Foam: how to get it

- **Free, online at** https://gitlab.com/foam-for-nuclear/GeN-Foam/-/tree/develop
  - “Develop” branch or “Master” branch
  - Either
    - git clone https://gitlab.com/foam-for-nuclear/GeN-Foam.git”
    - or, simply download

GeN-Foam: how to get it



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

0a05c5b4

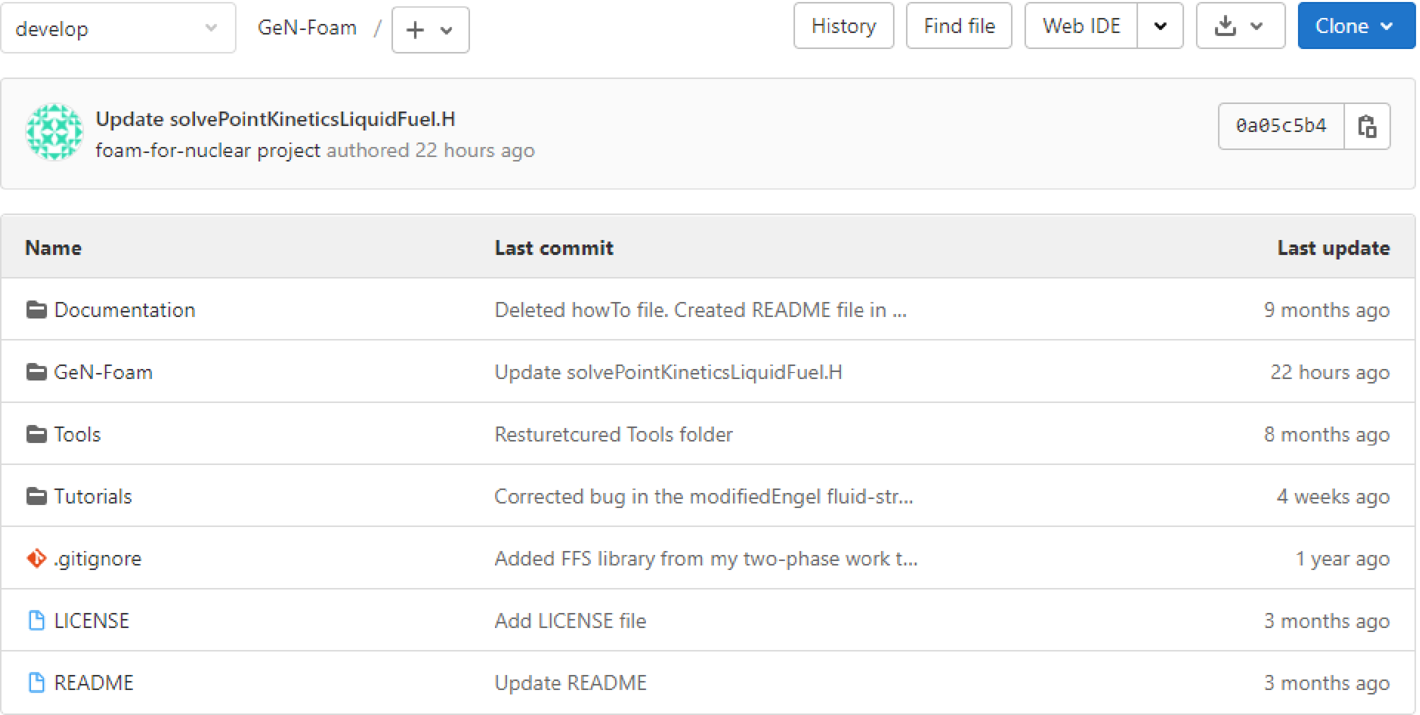
Name	Last commit	Last update
Documentation	Deleted howTo file. Created README file in ...	9 months ago
GeN-Foam	Update solvePointKineticsLiquidFuel.H	22 hours ago
Tools	Resturetcured Tools folder	8 months ago
Tutorials	Corrected bug in the modifiedEngel fluid-str...	4 weeks ago
.gitignore	Added FFS library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

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

What’s inside

Carlo Fiorina



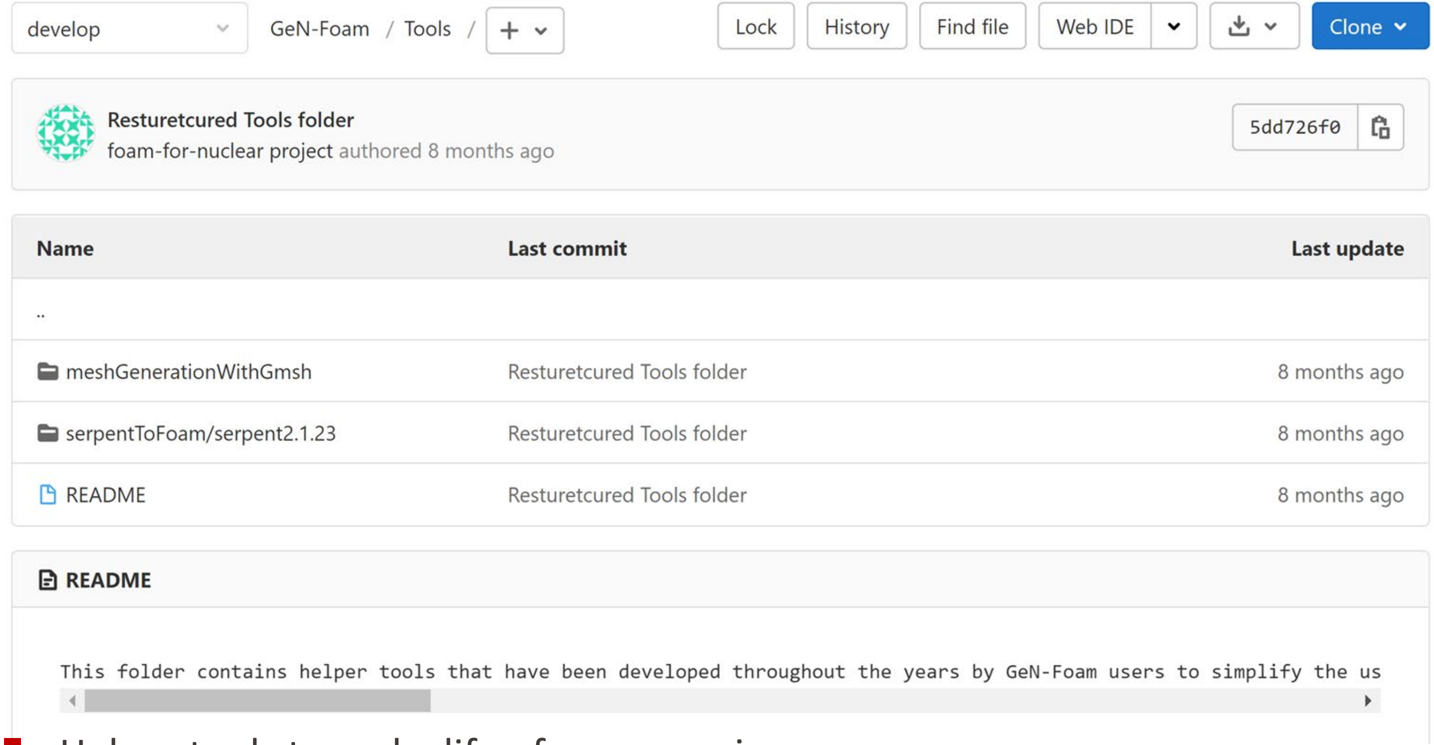
Update solvePointKineticsLiquidFuel.H
foam-for-nuclear project authored 22 hours ago

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

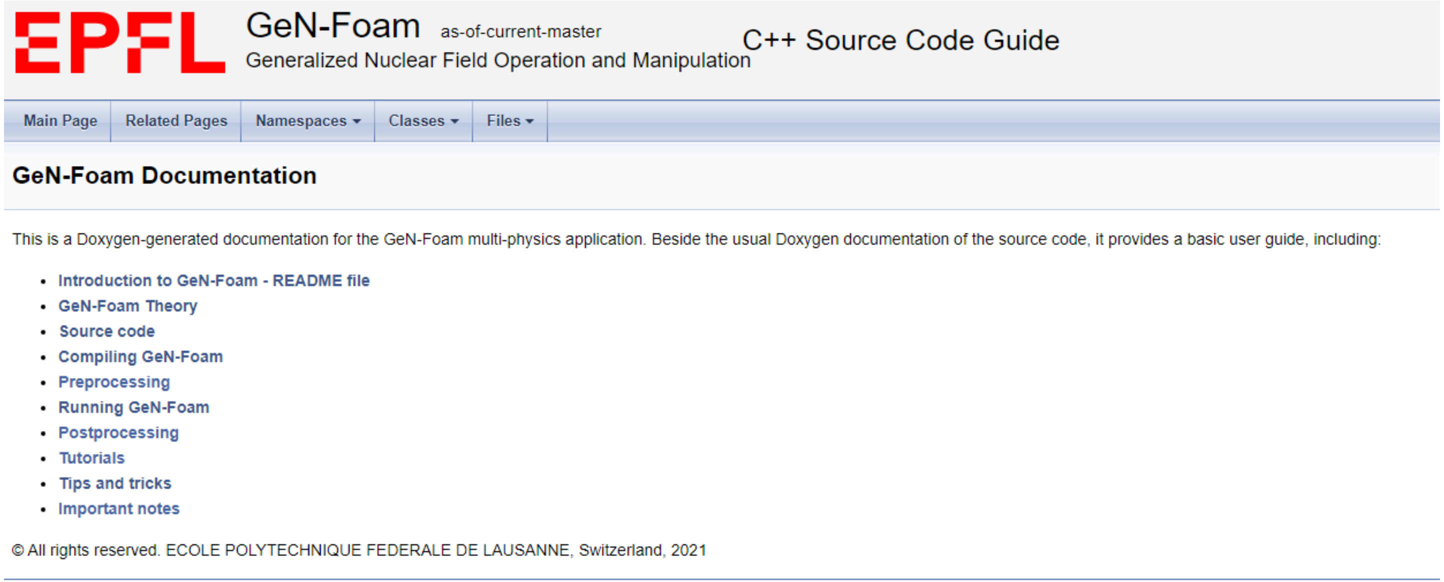
```
This folder contains helper tools that have been developed throughout the years by GeN-Foam users to simplify the us
```

- Helper tools to make life of a user 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

GeN-Foam is an unusually complex OpenFOAM solver. For this reason, some documentation (in the form of an online Doxygen-generated documentation has been prepared to facilitate its use. In addition, several commented tutorials have been prepared to showcase use and capabilities of the solver. An EMPTY case is also provided that can be used for step-by-step building one's own case. It is recommended to start from the EMPTY case to build each new case, as it already includes a consistent minimum set of (dummy) files that have to be present independent of the physics that are solved for. Beside this documentation, users ar encouraged to make use of the typical OpenFOAM ways:

- the high-level C++-based object-oriented language of OpenFOAM, which normally allows to easily understand the logic of a solver;
- the comments that are typically available in the source code and, in particular, in the header files of each class;
- the support of the community.


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

GeN-Foam Documentation

This is a Doxygen-generated documentation for the GeN-Foam multi-physics application. Beside the usual Doxygen documentation of the source code, it provides a basic user guide, including:

- Introduction to GeN-Foam - README file
- GeN-Foam Theory
- Source code
- Compiling GeN-Foam
- Preprocessing
- Running GeN-Foam
- Postprocessing
- Tutorials
- Tips and tricks
- Important notes

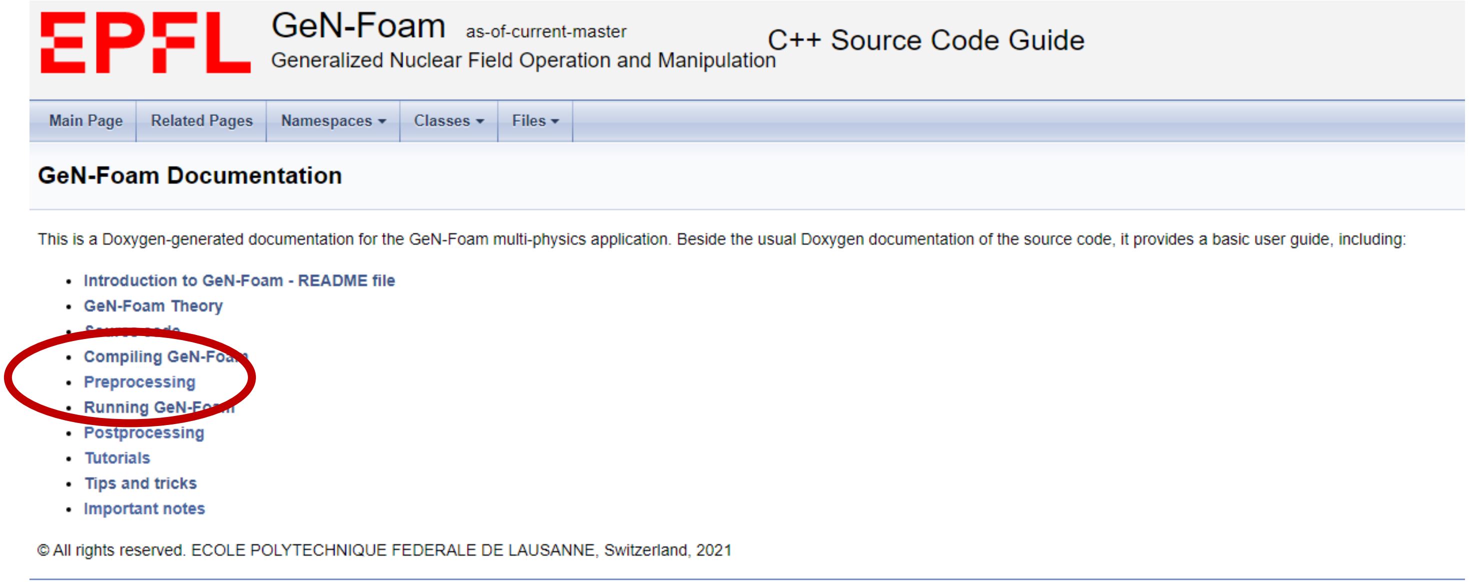
https://foam-for-nuclear.gitlab.io/GeN-Foam/index.html

Carlo Fiorina

What’s inside: Documentation

GeN-Foam is an unusually complex OpenFOAM solver. For this reason, some documentation (in the form of an online Doxygen-generated documentation has been prepared to facilitate its use. In addition, several commented tutorials have been prepared to showcase use and capabilities of the solver. An EMPTY case is also provided that can be used for step-by-step building one's own case. It is recommended to start from the EMPTY case to build each new case, as it already includes a consistent minimum set of (dummy) files that have to be present independent of the physics that are solved for. Beside this documentation, users ar encouraged to make use of the typical OpenFOAM ways:

- the high-level C++-based object-oriented language of OpenFOAM, which normally allows to easily understand the logic of a solver;
- the comments that are typically available in the source code and, in particular, in the header files of each class;
- the support of the community.



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

GeN-Foam Documentation

This is a Doxygen-generated documentation for the GeN-Foam multi-physics application. Beside the usual Doxygen documentation of the source code, it provides a basic user guide, including:

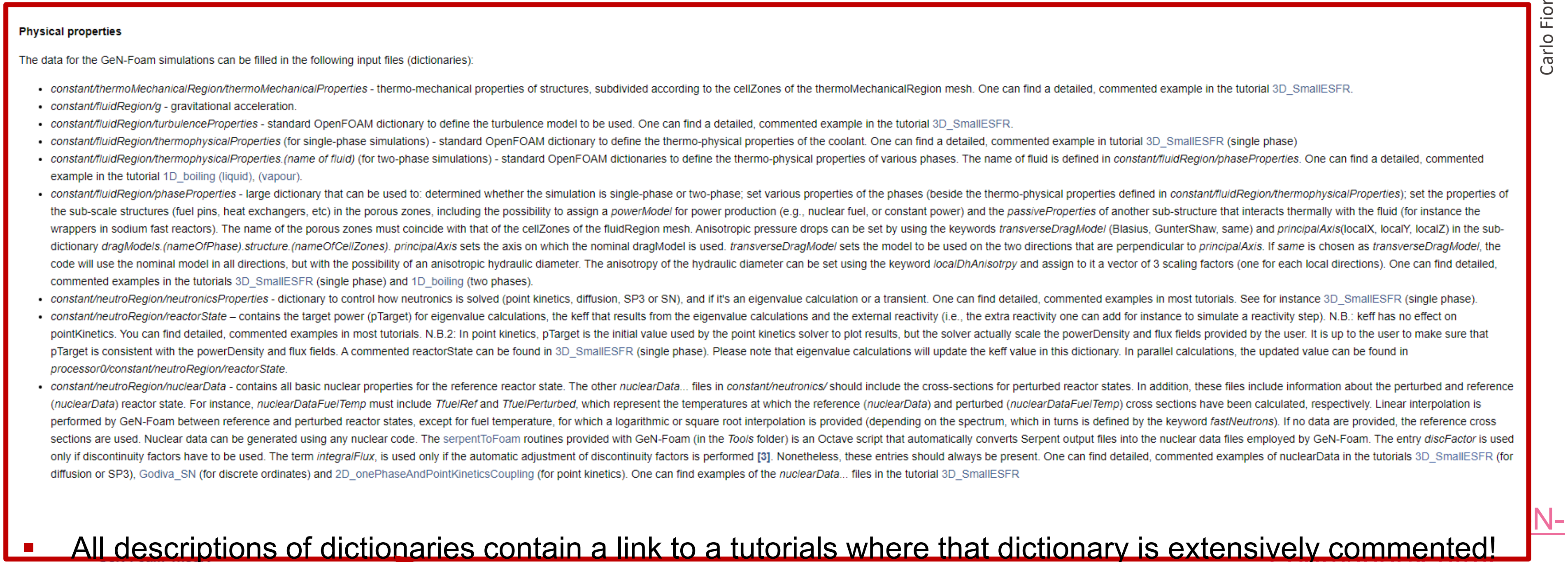
- Introduction to GeN-Foam - README file
- GeN-Foam Theory
- Source code
- Compiling GeN-Foam
- Preprocessing
- Running GeN-Foam
- Postprocessing
- Tutorials
- Tips and tricks
- Important notes

© All rights reserved. ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE, Switzerland, 2021

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

Carlo Fiorina

What’s inside: Documentation



Physical properties

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).
- *constant/neutroRegion/neutronicsProperties* - dictionary to control how neutronics is solved (point kinetics, diffusion, SP3 or SN), and if it's an eigenvalue calculation or a transient. One can find detailed, commented examples in most tutorials. See for instance 3D_SmallESFR (single phase).
- *constant/neutroRegion/reactorState* – contains the target power (pTarget) for eigenvalue calculations, the keff that results from the eigenvalue calculations and the external reactivity (i.e., the extra reactivity one can add for instance to simulate a reactivity step). N.B.: keff has no effect on pointKinetics. You can find detailed, commented examples in most tutorials. N.B.2: In point kinetics, pTarget is the initial value used by the point kinetics solver to plot results, but the solver actually scale the powerDensity and flux fields provided by the user. It is up to the user to make sure that pTarget is consistent with the powerDensity and flux fields. A commented reactorState can be found in 3D_SmallESFR (single phase). Please note that eigenvalue calculations will update the keff value in this dictionary. In parallel calculations, the updated value can be found in *processor0/constant/neutroRegion/reactorState*.
- *constant/neutroRegion/nuclearData* - 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). One can find examples of the *nuclearData*... files in the tutorial 3D_SmallESFR

- All descriptions of dictionaries contain a link to a tutorials where that dictionary is extensively commented!

N-

Foam/index.html

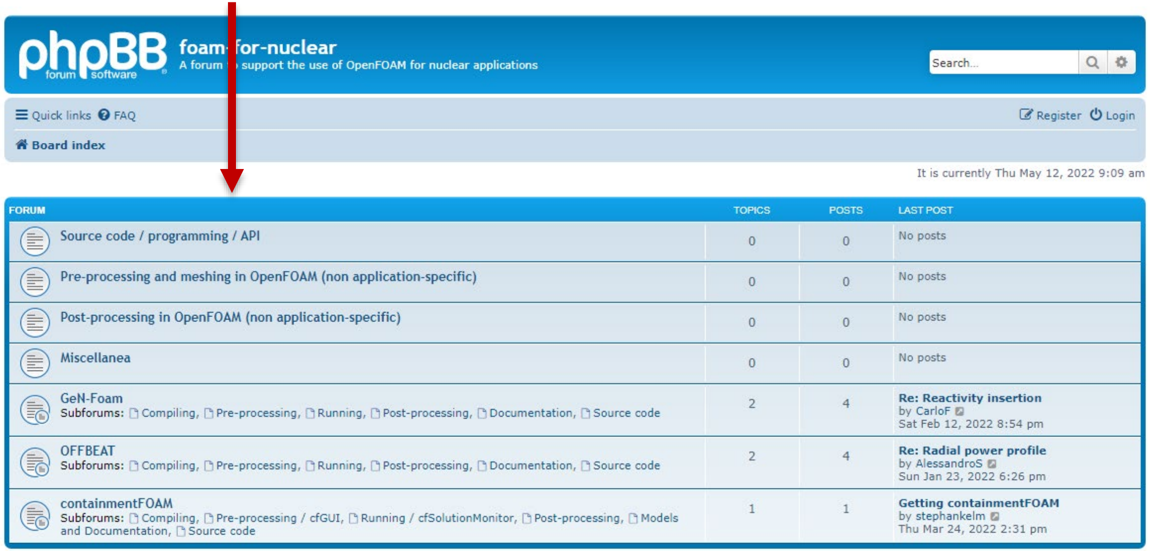
- Source code
- Compiling GeN-Foam
- Preprocessing
- Running GeN-Foam
- Postprocessing
- Tutorials
- Tips and tricks
- Important notes

What’s inside: Documentation

Carlo Fiorina

GeN-Foam is an unusually complex OpenFOAM solver. For this reason, some documentation (in the form of an online Doxygen-generated documentation has been prepared to facilitate its use. In addition, several commented tutorials have been prepared to showcase use and capabilities of the solver. An EMPTY case is also provided that can be used for step-by-step building one's own case. It is recommended to start from the EMPTY case to build each new case, as it already includes a consistent minimum set of (dummy) files that have to be present independent of the physics that are solved for. Beside this documentation, users ar encouraged to make use of the typical OpenFOAM ways:

- the high-level C++-based object-oriented language of OpenFOAM, which normally allows to easily understand the logic of a solver;
- the comments that are typically available in the source code and, in particular, in the header files of each class;
- the support of the community.



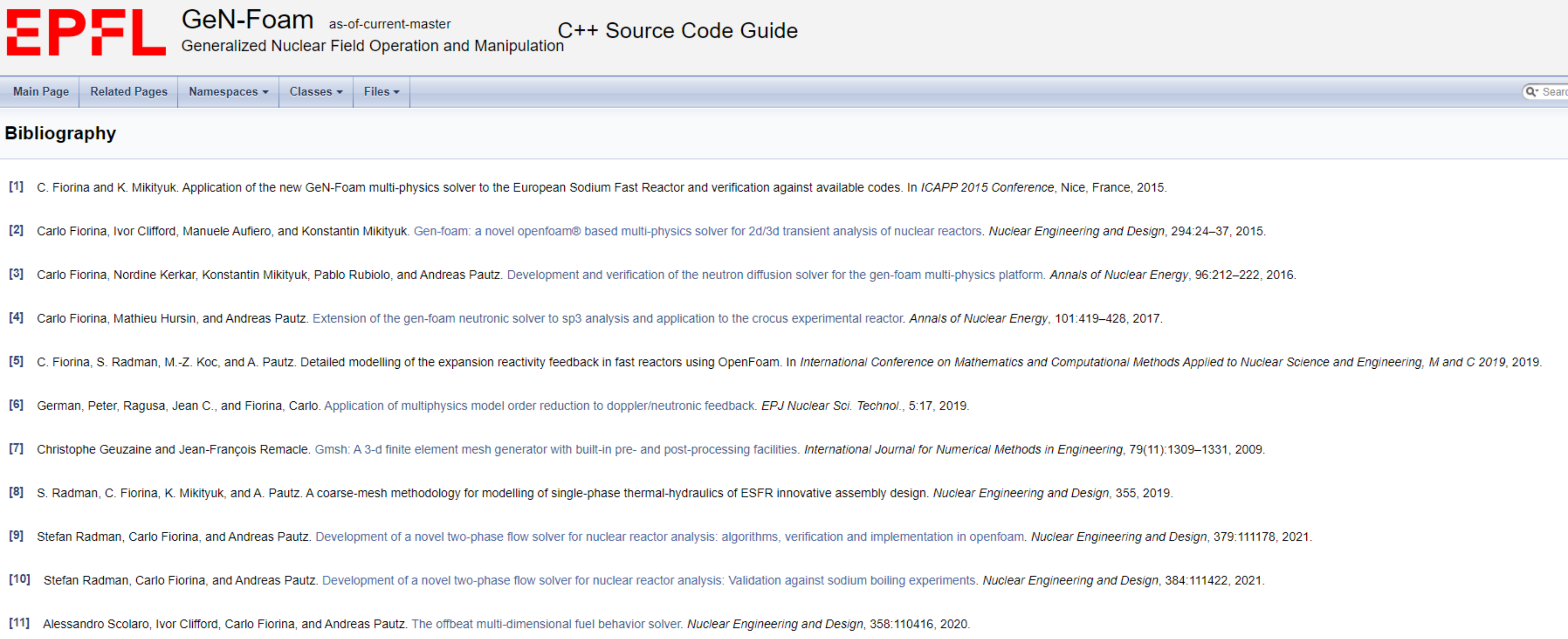
foam-for-nuclear
A forum to support the use of OpenFOAM for nuclear applications

It is currently Thu May 12, 2022 9:09 am

FORUM	TOPICS	POSTS	LAST POST
Source code / programming / API	0	0	No posts
Pre-processing and meshing in OpenFOAM (non application-specific)	0	0	No posts
Post-processing in OpenFOAM (non application-specific)	0	0	No posts
Miscellanea	0	0	No posts
GeN-Foam Subforums: Compiling, Pre-processing, Running, Post-processing, Documentation, Source code	2	4	**Re: Reactivity insertion** by CarloF Sat Feb 12, 2022 8:54 pm
OFFBEAT Subforums: Compiling, Pre-processing, Running, Post-processing, Documentation, Source code	2	4	**Re: Radial power profile** by AlessandroS Sun Jan 23, 2022 6:26 pm
containmentFOAM Subforums: Compiling, Pre-processing / cfGUI, Running / cfSolutionMonitor, Post-processing, Models and Documentation, Source code	1	1	**Getting containmentFOAM** by stephankelm Thu Mar 24, 2022 2:31 pm

https://foam-for-nuclear.org/phpBB/

What’s inside: Documentation



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.

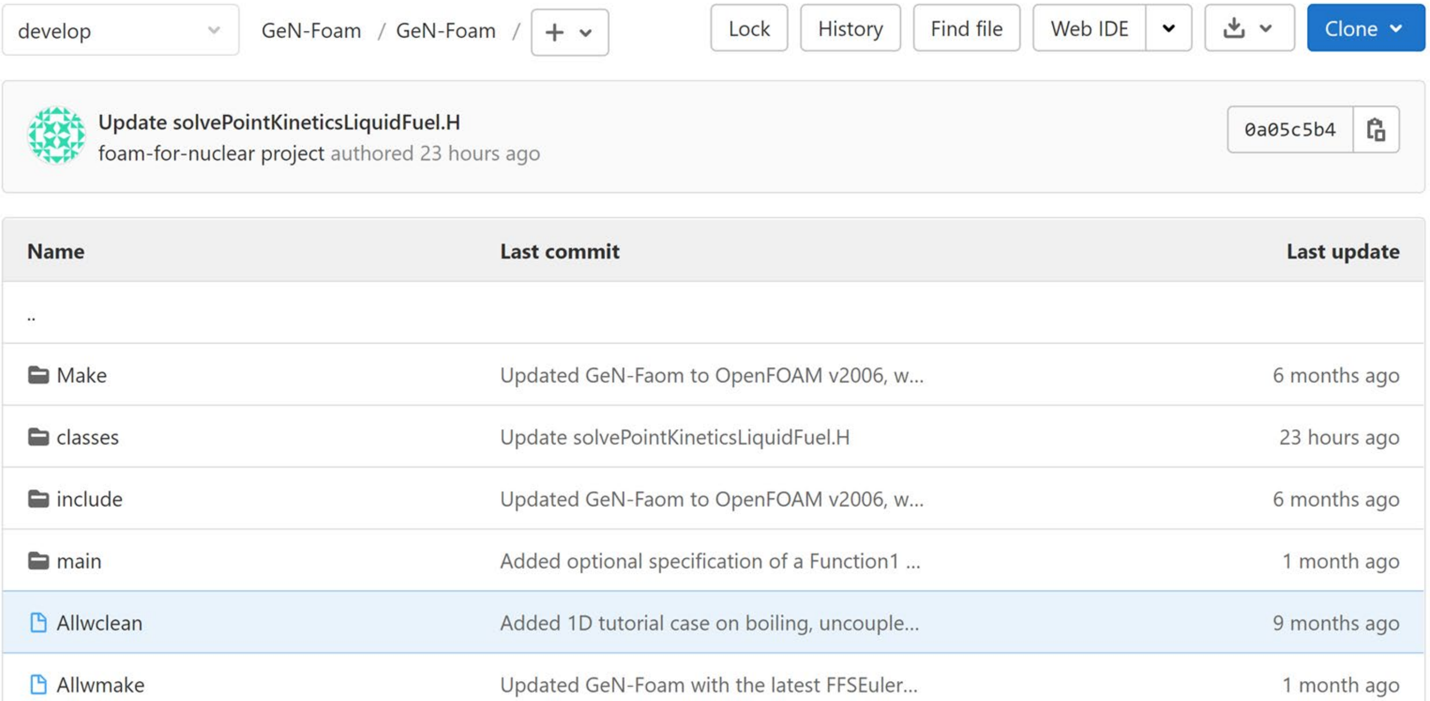
[9] Stefan Radman, Carlo Fiorina, and Andreas Pautz. Development of a novel two-phase flow solver for nuclear reactor analysis: algorithms, verification and implementation in openfoam. *Nuclear Engineering and Design*, 379:111178, 2021.

[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



Update solvePointKineticsLiquidFuel.H
foam-for-nuclear project authored 23 hours ago

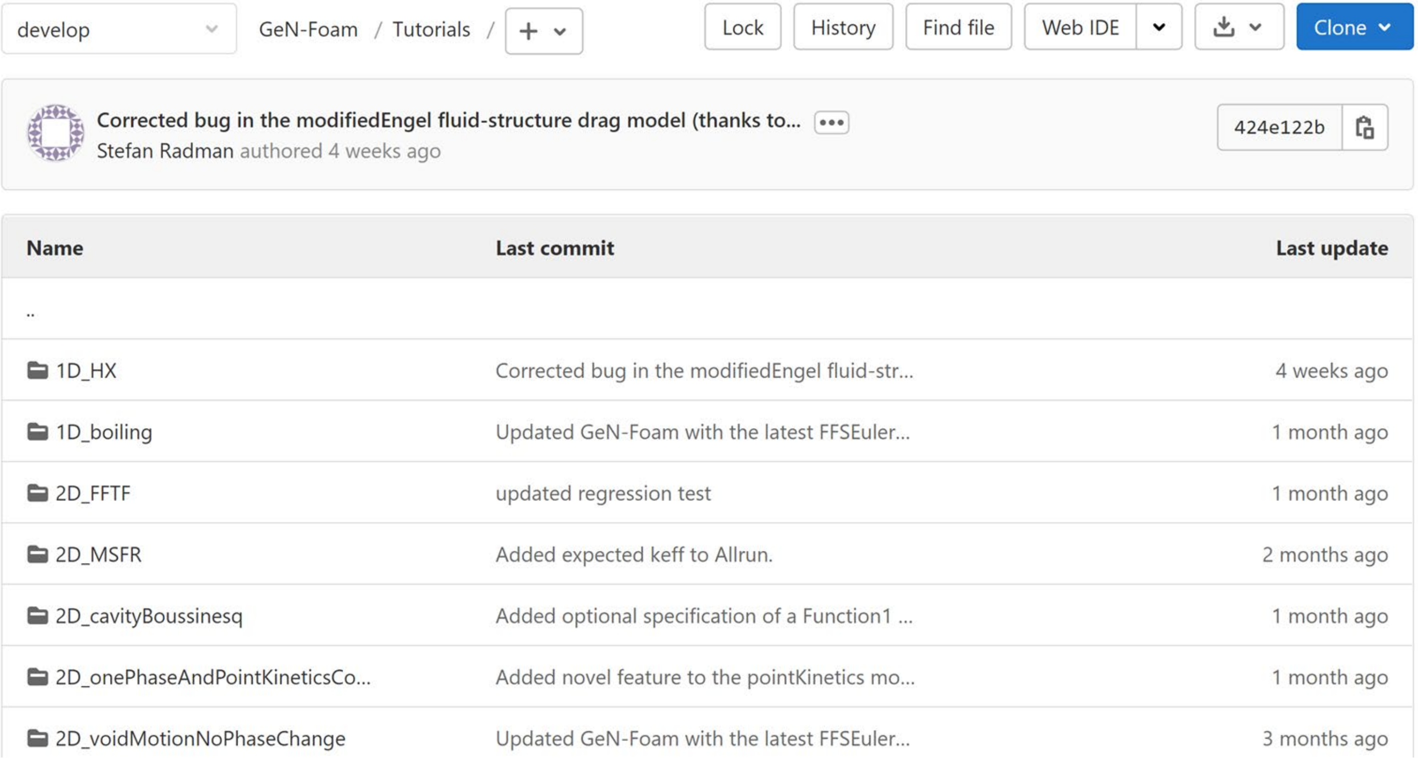
0a05c5b4

Name	Last commit	Last update
..		
Make	Updated GeN-Faom to OpenFOAM v2006, w...	6 months ago
classes	Update solvePointKineticsLiquidFuel.H	23 hours ago
include	Updated GeN-Faom to OpenFOAM v2006, w...	6 months ago
main	Added optional specification of a Function1 ...	1 month ago
Allwclean	Added 1D tutorial case on boiling, uncouple...	9 months ago
Allwmake	Updated GeN-Foam with the latest FFSEuler...	1 month ago

- “Classes” contains all the physics
- “main” contains what glues them together
- “include” are folders that mainly contain chunks of code that perform specific tasks and that are included (#include) in the code

https://gitlab.com/foam-for-nuclear/GeN-Foam/-/tree/develop/GeN-Foam

What’s inside: Tutorials



Corrected bug in the modifiedEngel fluid-structure drag model (thanks to... •••
Stefan Radman authored 4 weeks ago

Name	Last commit	Last update
..		
1D_HX	Corrected bug in the modifiedEngel fluid-str...	4 weeks ago
1D_boiling	Updated GeN-Foam with the latest FFSEuler...	1 month ago
2D_FFTF	updated regression test	1 month ago
2D_MSFR	Added expected keff to Allrun.	2 months ago
2D_cavityBoussinesq	Added optional specification of a Function1 ...	1 month ago
2D_onePhaseAndPointKineticsCo...	Added novel feature to the pointKinetics mo...	1 month ago
2D_voidMotionNoPhaseChange	Updated GeN-Foam with the latest FFSEuler...	3 months ago

- 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

Carlo Fiorina

An example: 1D_MSR_pointKinetics

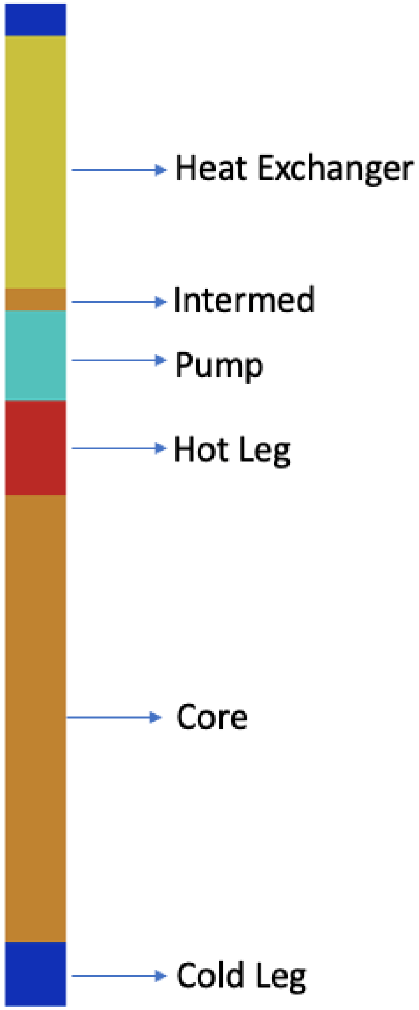
- https://gitlab.com/foam-for-nuclear/GeN-Foam/-/tree/develop/Tutorials/1D_MSR_pointKinetics
- Understanding the tutorial:
  - README file
  - Case folder
  - Allrun file
  - Run it and use paraview to see what happens

An example: 1D_MSR_pointKinetics

- Start from the README file (https://gitlab.com/foam-for-nuclear/GeN-Foam/-/tree/develop/Tutorials/1D_MSR_pointKinetics/README)

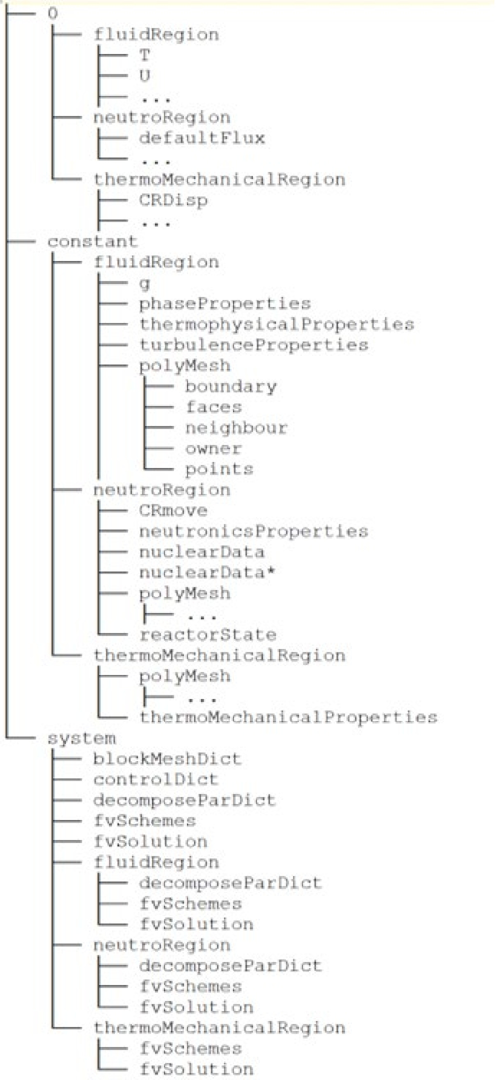
```
DESCRIPTION
This tutorial displays how to use the point kinetics module of
GeN-Foam for MSRs. It is a simple 1-D case with core, hot leg,
pump, heat exchanger and cold leg. The geometry is one
dimensional and salt recirculation is simulated by making use
of a cyclic boundary condition between top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



An example: 1D_MSR_pointKinetics

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

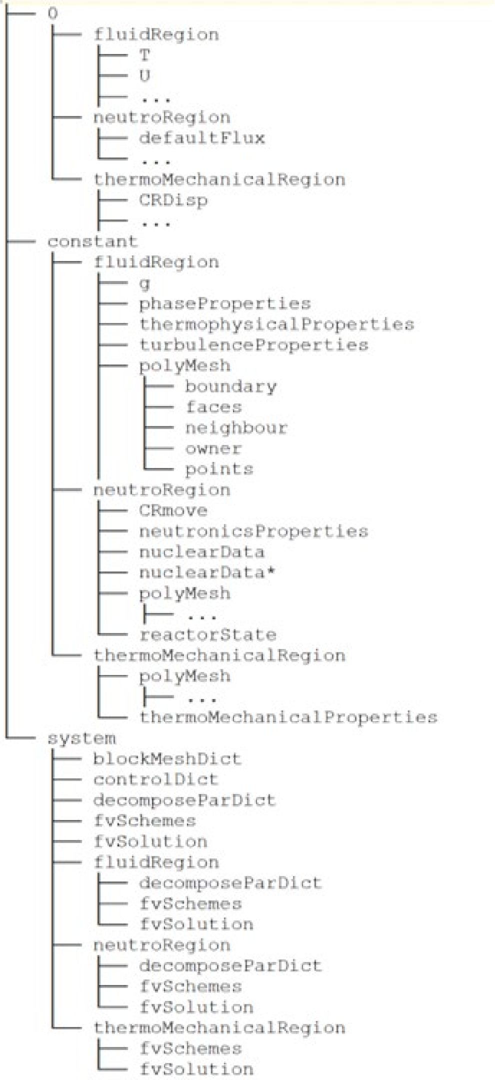


```
├── 0
│   ├── fluidRegion
│   │   ├── T
│   │   ├── U
│   │   ├── ...
│   ├── neutroRegion
│   │   ├── defaultFlux
│   │   ├── ...
│   └── thermoMechanicalRegion
│       ├── CRDisp
│       ├── ...
├── constant
│   ├── fluidRegion
│   │   ├── g
│   │   ├── phaseProperties
│   │   ├── thermophysicalProperties
│   │   ├── 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
```

Carlo Fiorina

An example: 1D_MSR_pointKinetics

- Look at the dictionaries
  - All the dictionaries are extensively commented in at least one of the tutorials
  - Which tutorial to look at for every dictionary? Look in the Prepreprocessing section of the documentation https://foam-for-nuclear.gitlab.io/GeN-Foam/PREPROCESSING.html
  - In our case, the tutorial is mainly dedicated to the point kinetics model. Look at constant/neutroRegion/nuclearData https://gitlab.com/foam-for-nuclear/GeN-Foam/-/blob/master/Tutorials/2D_onePhaseAndPointKineticsCoupling/rootCase/constant/neutroRegion/nuclearData



```
├── 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
```

An example: 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

...
```

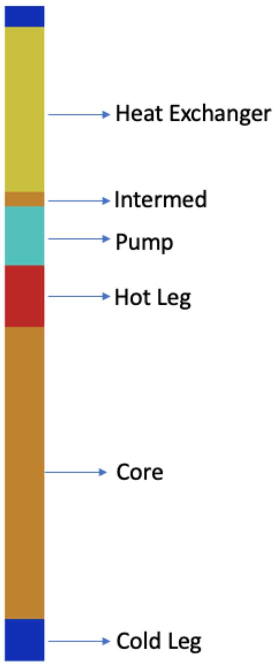
An example: 1D_MSR_pointKinetics

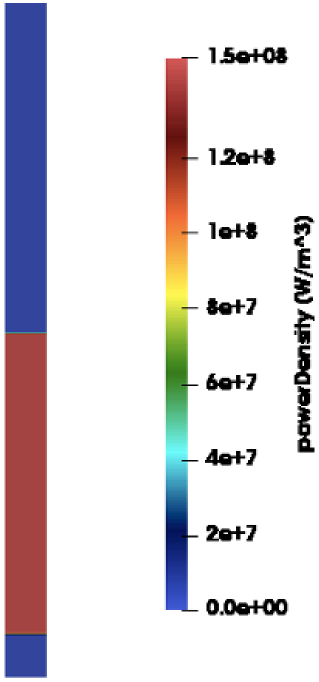
Carlo Fiorina

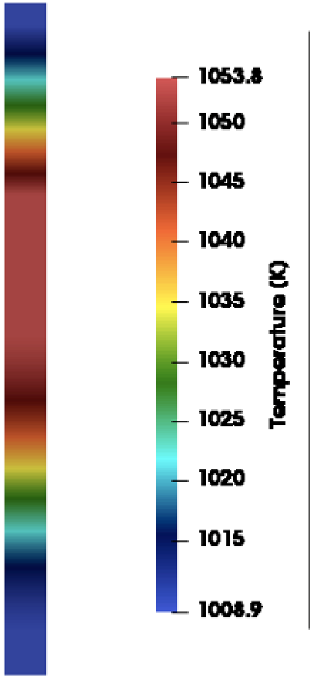
- Run the tutorial -> ./Allrun
- Check the results:
  - Choose a folder: steadyState, transient, transientEnd
  - Use:
    - paraFoam
    - ./log.GeN-Foam: standard OpenFOAM log
    - ./GeN-Foam.dat: quick overview of time behavior of main quantities (power, keff, min/max/average fuel and clad temp.)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

An example: 1D_MSR_pointKinetics

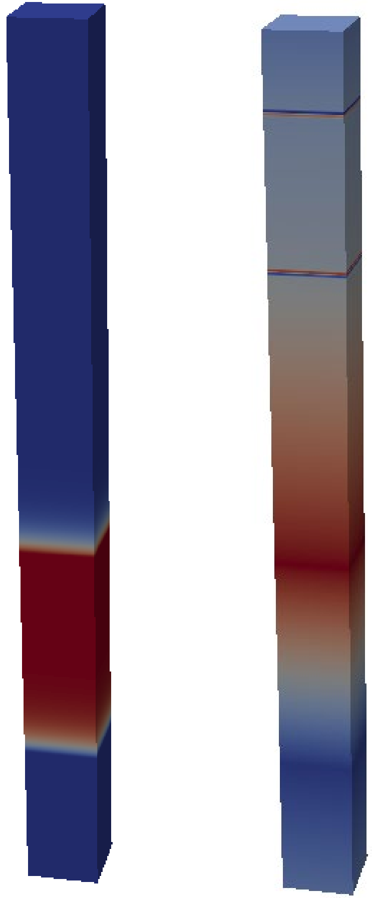
- paraFoam







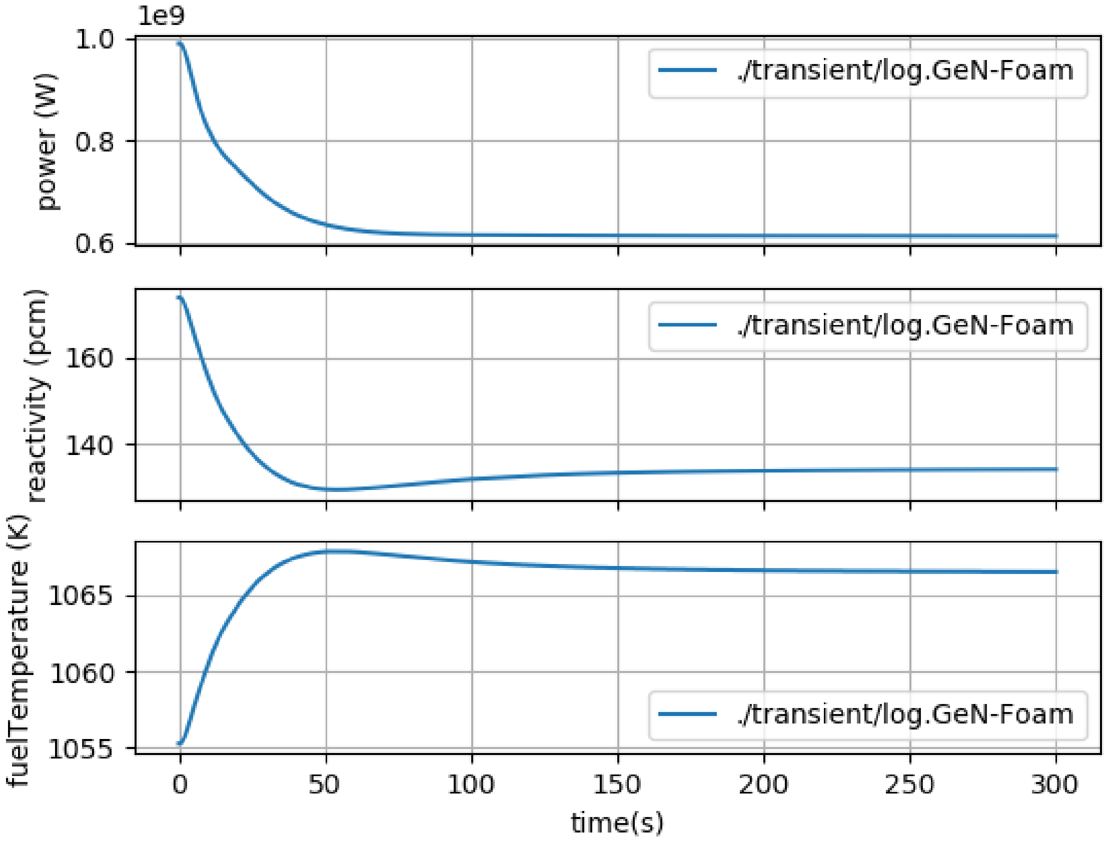
Precursors, group 0 and 7



An example: 1D_MSR_pointKinetics

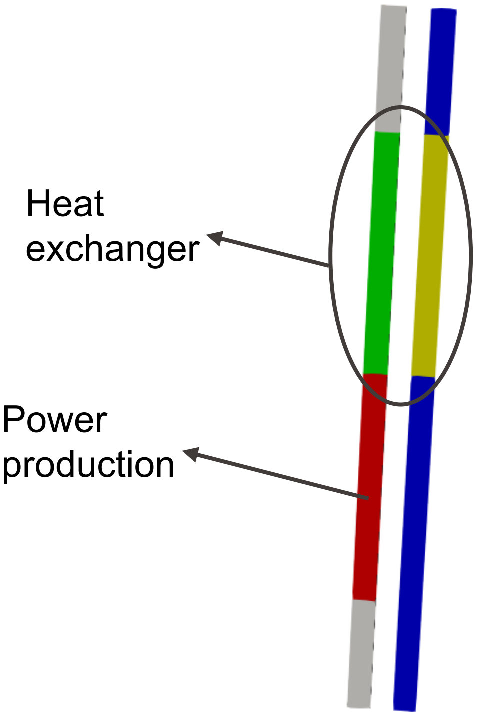
- python script (extract data from log)
- Type in terminal:

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



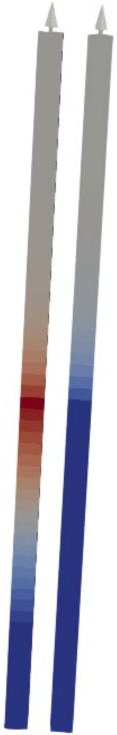
Other tutorials: 1D_HX

Example on how to set up a heat exchanger



Heat exchanger

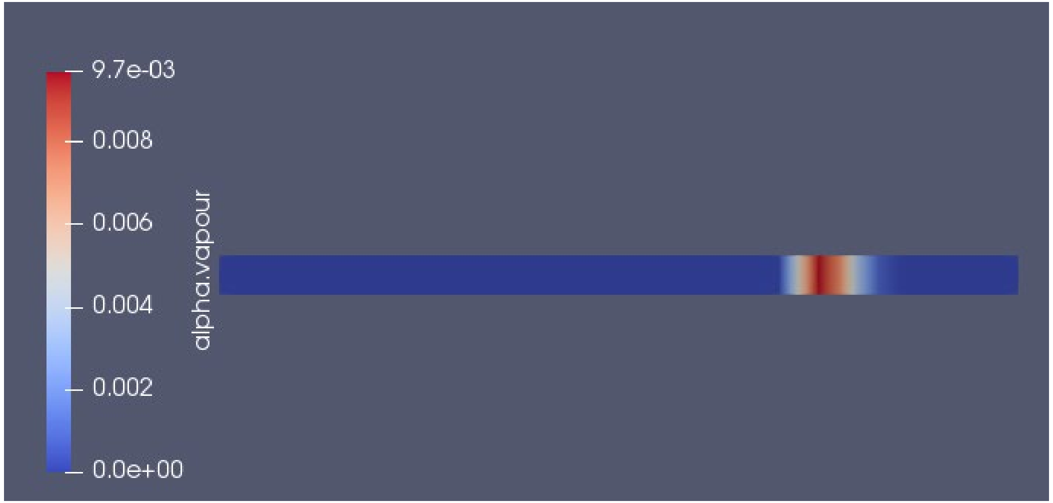
Power production



Other tutorials: 1D_boiling

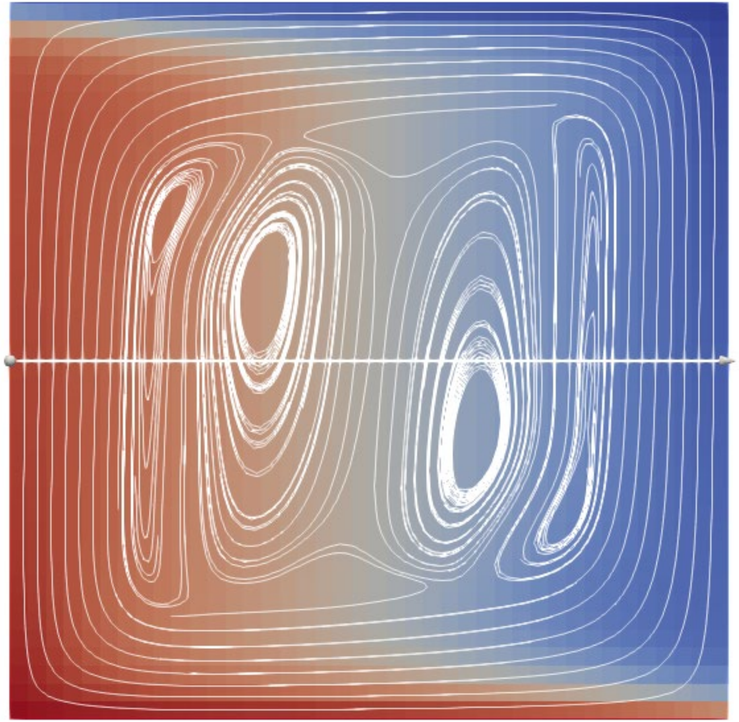
Carlo Fiorina

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



Other tutorials: 2D_cavityBoussinesq

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

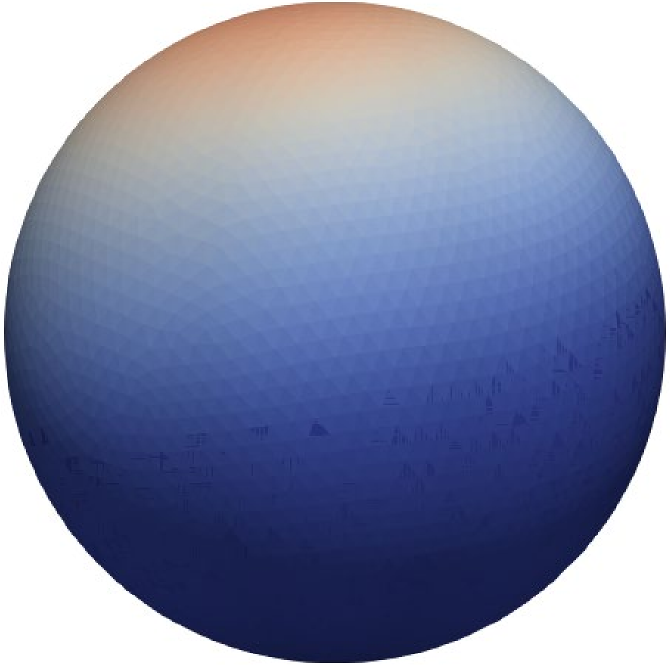


Other tutorials: 2D_voidMotionNoPhaseChange

Simple two-phase case without mass transfer between phases

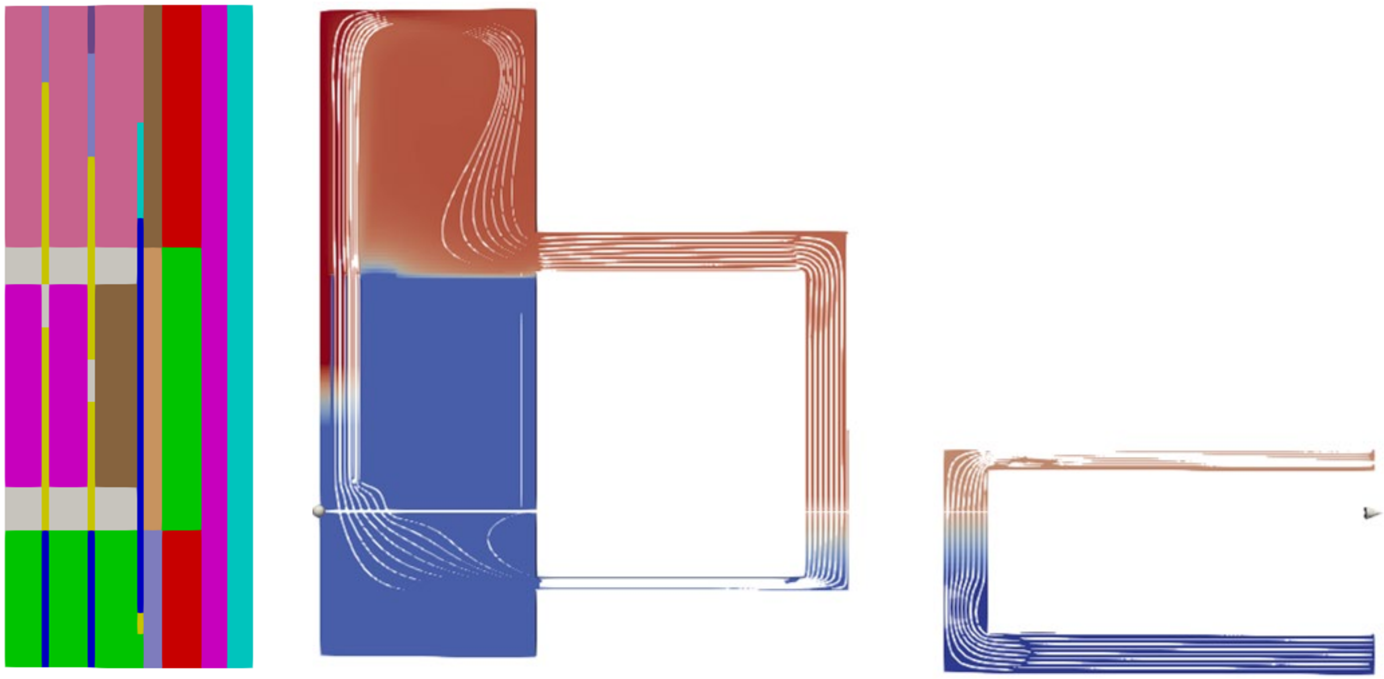
Other tutorials: Godiva_SN

Example of a discrete ordinate calculation of Godiva



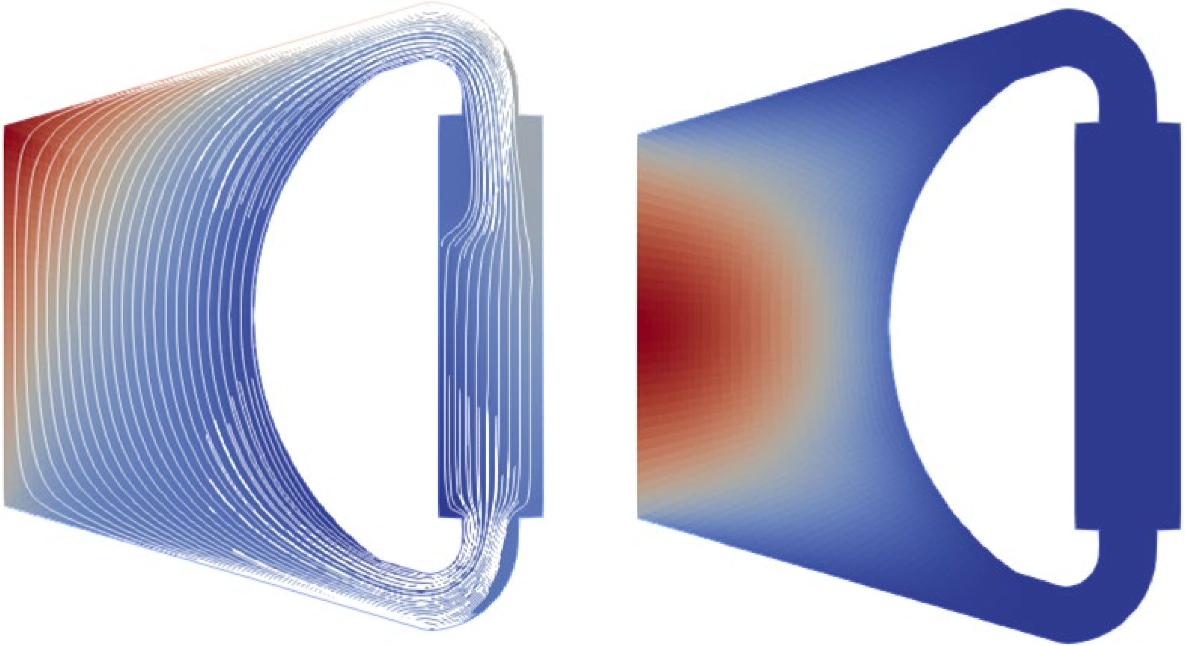
Other tutorials: 2D_FFTF

2-D model of the FFTF. Simulation of a ULOF



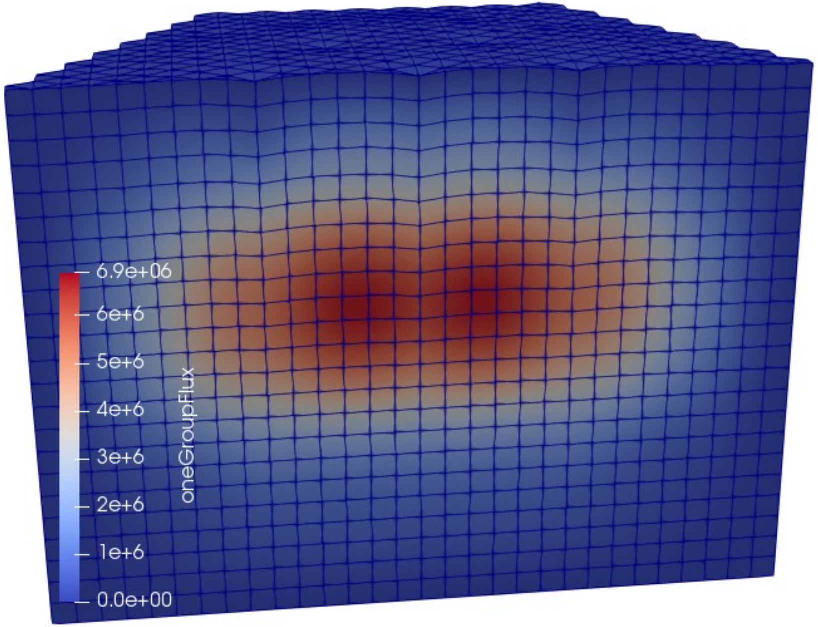
Other tutorials: 2D_MSFR

2-D model of the MSFR



Other tutorials: 3D_SmallESFR

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



Thank you

Carlo Fiorina

How to install it - 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!
Some more details are given in modules/visualization, but essentially the paraview version distributed with the operating system or a newer binary package is likelyfully adequate for your needs. 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

The source code of the GeN-Foam multi-physics solver

- All sub-solvers are organized into C++ classes
    - ○ Easier to understand its coding
    - ○ Possible to easily extract sub-solvers for use in other solvers
        - ✓ You have complete freedom to freely use and modify
        - ✓ (Does not mean that copyright does not exists: acknowledgment of previous the work of other authors is always good practice and consistent with ethics in open-source development)

What is a C++ class

- C++ is object oriented
- Object-oriented roughly means that you can organize you code into classes
- Classes are a set of data, and functions that operate on those data
- For instance, in GeN-Foam, classes for:
  - neutronics
  - cross-sections
  - thermal-hydraulics
  - thermal-mechanics
  - other “functional classes” e.g. for handling multi-physics simulations
- For instance, the neutronics class contains:
  - neutronics quantities, such as keff, power field, etc.
  - functions that manipulate these quantities

What is a C++ class

- Classes can have *derived classes*, i.e., classes that can “see” everything in the original class, but that contains additional data and functions
- In GeN-Foam, this is used to “specialize” solver classes into sub-solvers
- For instance, from the neutronic class, we derive classes for:
  - diffusion
  - Sp3
  - SN
  - point-kinetics
- For instance, the “diffusion” derived class contains:
  - all data and functions from the neutronics class
  - additional data (e.g., multi-group fluxes)
  - additional functions, the most important being the function that solves for the fluxes at every time step