IAEA
International Atomic Energy Agency

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

# An overview on the use of OpenFOAM as a multi-physics library for nuclear reactor analysis

Carlo Fiorina, Stephan Kelm, (Ivor Clifford)

# Content of this webinar

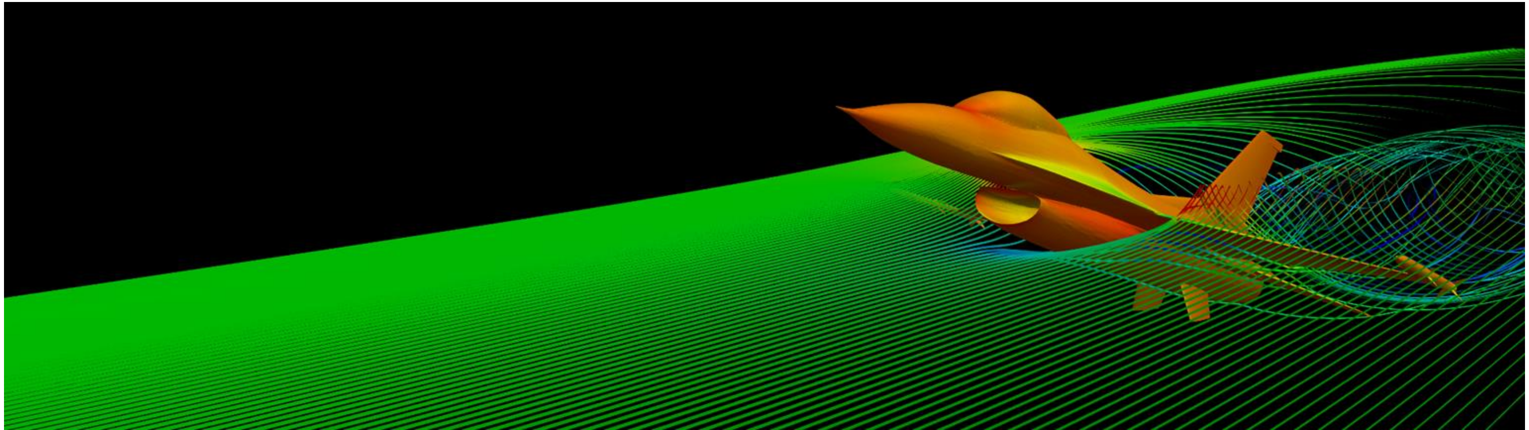
- Introduction to OpenFOAM
- Examples of use of OpenFOAM for multi-physics modelling in nuclear
- How to approach a new problem with OpenFOAM
- Lessons learnt

# OpenFOAM

Open∇FOAM

*The Open Source CFD Toolbox*

- Officially described as an open-source CFD toolbox
  - Capabilities mirror those of commercial CFD
  - Free-to-use software without paying for licensing
- ~10k to 20k estimated users worldwide



# What is OpenFOAM really?

Open∇FOAM

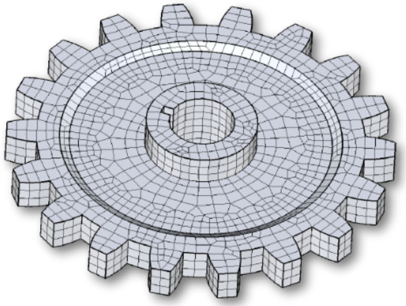
IAEA

*The Open Source CFD Toolbox*

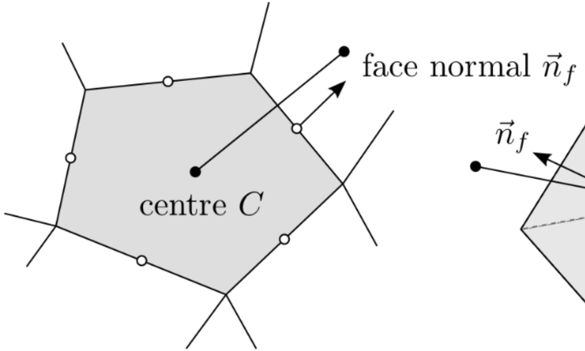
OpenFOAM stands for Open Field Operation and Manipulation

- 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

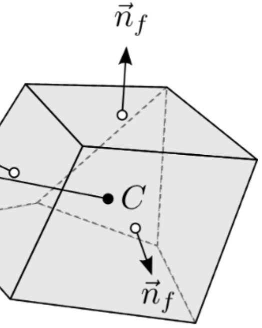
- Object-oriented, with a high-level “fail-safe” API



Discretized Domain



2D



3D

# Equation Mimicking

- Natural language of continuum mechanics: **partial differential equations**
- Example: turbulence kinetic energy equation

$$\frac{dk}{dt} + \nabla \cdot (\vec{u}k) - \nabla \cdot [(\nu + \nu_t)\nabla k] = \nu_t \left[\frac{1}{2}(\nabla\vec{u} + \nabla\vec{u}^T)\right]^2 - \frac{\epsilon_0}{k_0} k$$

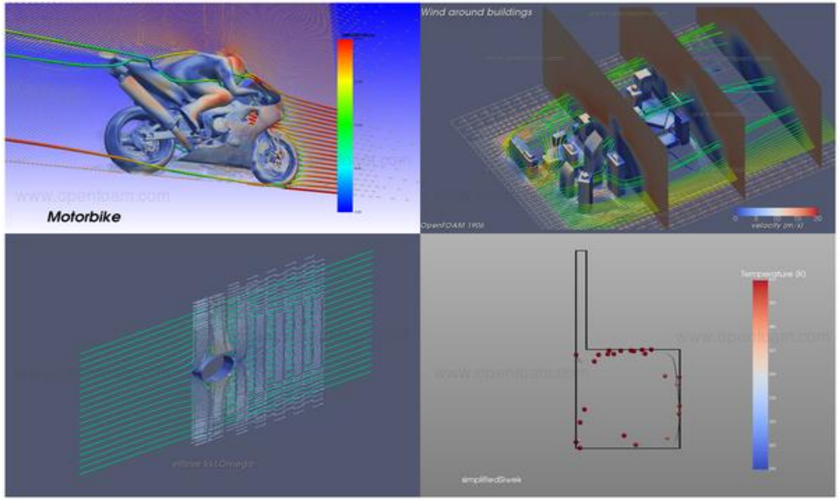
- Objective: represent PDEs in their natural language

```
solve
(
    fvm::ddt(k)
  + fvm::div(phi, k)
  - fvm::laplacian(nu() + nut, k)
==
    nut*magSqr(symm(fvc::grad(U)))
  - fvm::Sp(epsilon/k, k)
);
```

- Correspondence between implementation and equation is clear

# OpenFOAM: Solvers

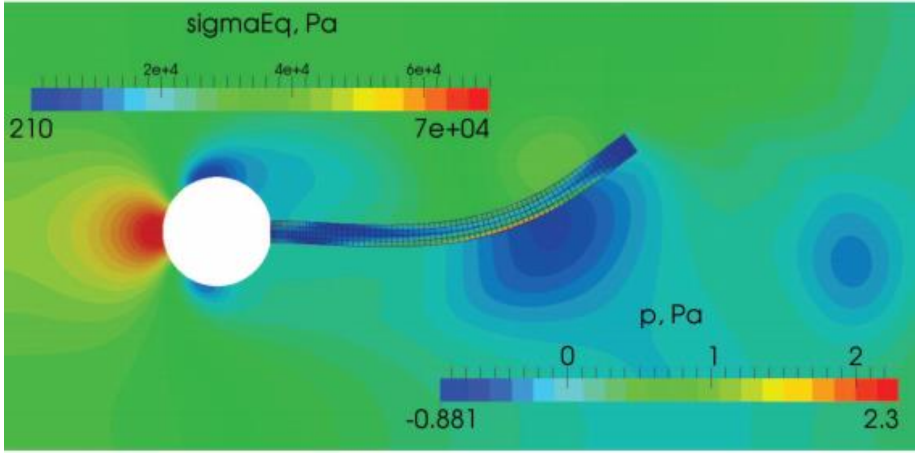
- Several solvers already available in the standard distribution:
  - 5 for basic CFD
  - 14 for incompressible flow (incl. adjoint, rotating frame, non-Newtonian, ...)
  - 11 for compressible flow (incl. trans-sonic and super-sonic)
  - 25 for multi-phase flow (incl., Euler-Euler, VOF, cavitation, free-surface,  and options for mesh topology changes and adaptive re-meshing)
  - 1 for DNS
  - 10 for combustion
  - 9 for heat transfer (incl. multi-region solid-fluid)
  - 17 for particle tracking
  - 2 for molecular dynamics
  - 1 for Monte Carlo simulations
  - 3 for electromagnetics (incl. MHD)
  - 2 for stress analysis
  - 1 for finance



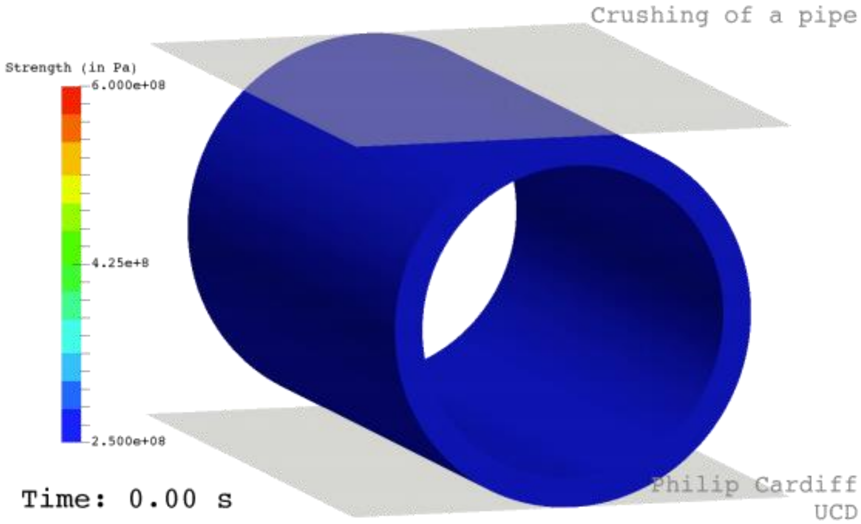
https://www.openfoam.com/news/main-news/openfoam-v1906/post-processing

# OpenFOAM: Solvers

- Several solvers (and solver collections) developed by the community:
  - e.g., solids4foam: large collection of solvers for solid mechanics from UC Dublin



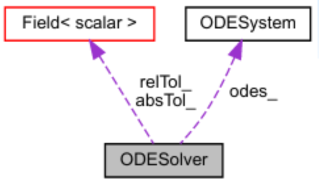
*Z. Tukovic et al. "OpenFOAM Finite Volume Solver for Fluid-Structure Interaction", 2018*

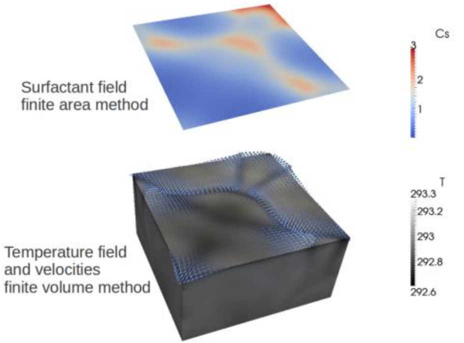


*P. Cardif et al. "A Lagrangian cell-centred finite volume method for metal forming simulation", 2016*

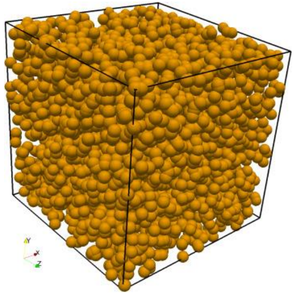
# OpenFOAM: Functionalities

- Large library with lots of available functionalities (in addition to finite-volume discretization and solution):
  - Mesh to mesh projections
  - Dynamic meshes, including adaptive meshes with topological changes
  - ODE solvers
  - Finite area method
  - Monte Carlo (Direct simulation Monte Carlo for multi-species flows)
  - Lagrangian particle tracking (two-phase flows, aerosols, DPM, etc.)

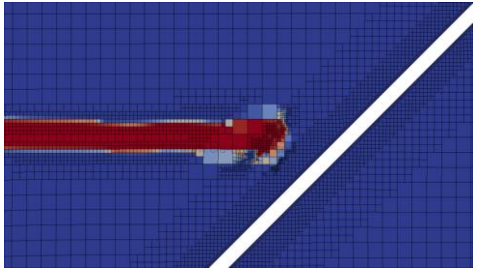




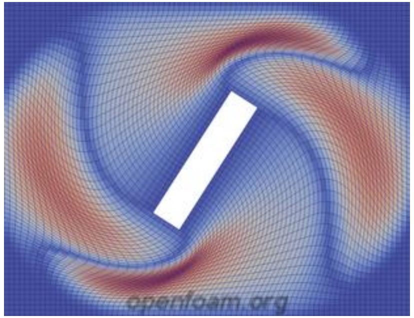
http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2011/SamFredriksson/Tutorial_buoyantBoussinesqPisoSurfactantFoam.pdf



https://www.sciencedirect.com/science/article/pii/S0010465517303375



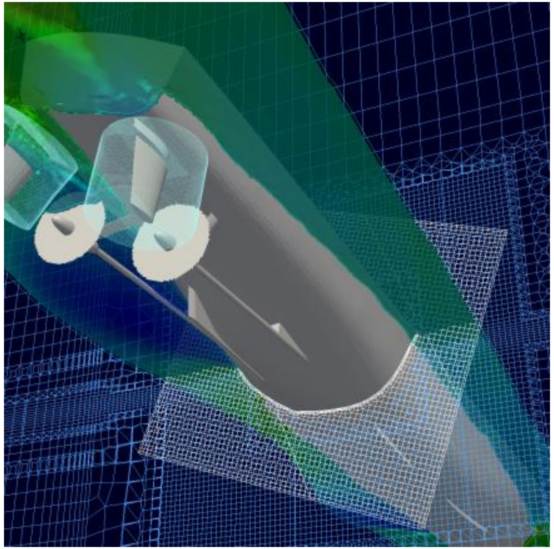
https://cfd-training.com/2018/01/06/how-to-use-dynamicrefinefvmesh-library/



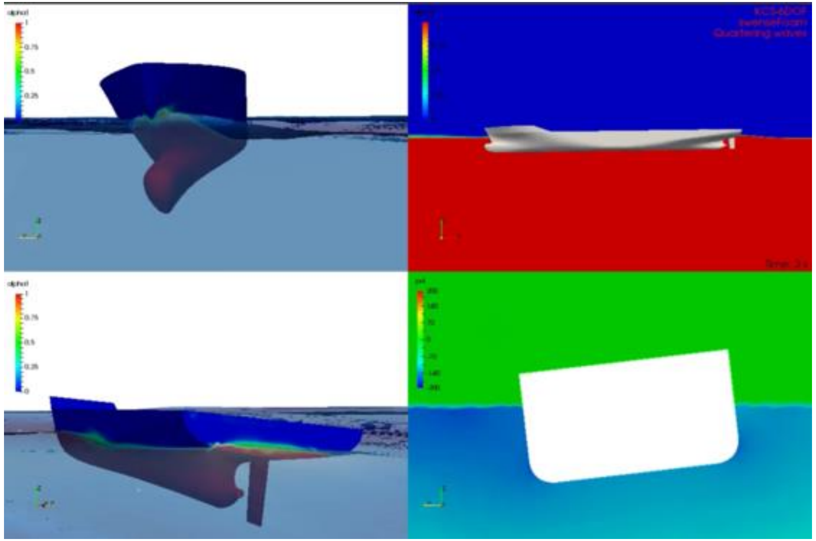
https://openfoam.org/release/2-3-0/mesh-motion/

# OpenFOAM: Functionalities

- Several additional functionalities (and libraries) developed by the community:
  - e.g., foam-extend project (https://sourceforge.net/projects/foam-extend/)

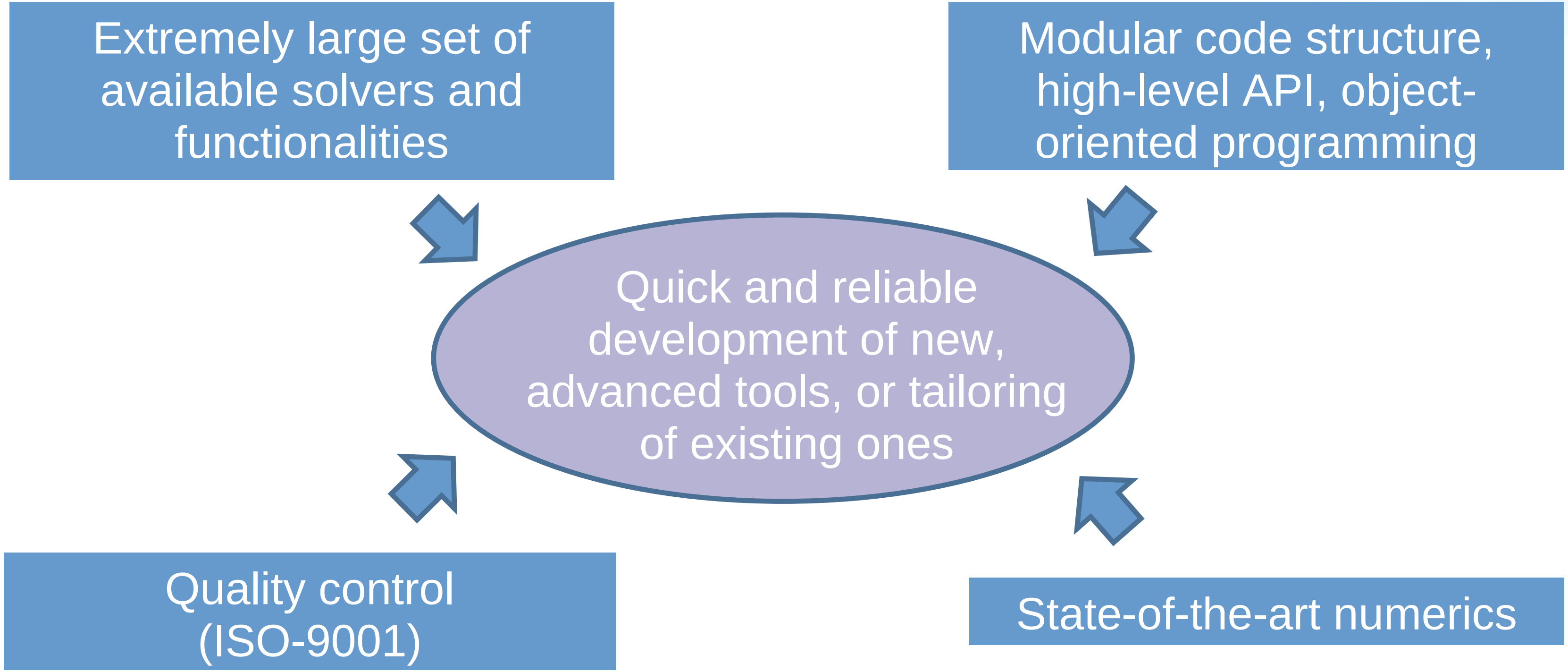


https://foam-extend.fsb.hr



http://openfoam-extend.sourceforge.net/OpenFOAM_Workshops/OFW11_2016_Guimaraes/special.html

# OpenFOAM: Standing on the shoulders of giants



Extremely large set of available solvers and functionalities

Modular code structure, high-level API, object-oriented programming

Quick and reliable development of new, advanced tools, or tailoring of existing ones

Quality control (ISO-9001)

State-of-the-art numerics

# Disclaimer

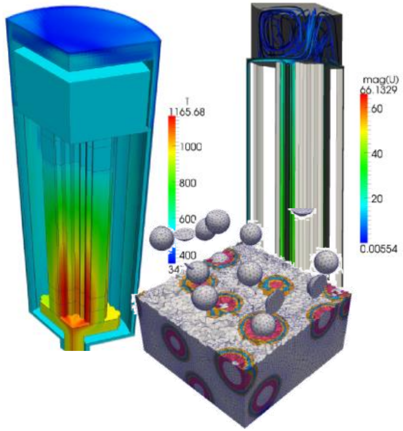
- Most of the following content is taken from
  - Carlo Fiorina, Ivor Clifford, Stephan Kelm, Stefano Lorenzi, 2022. “On the development of multi-physics tools for nuclear reactor analysis based on OpenFOAM ®: state of the art, lessons learned and perspectives”. Nuclear Engineering and Design 387, 111604.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

# Use of OpenFOAM for nuclear multi-physics

2000-2010
First activities

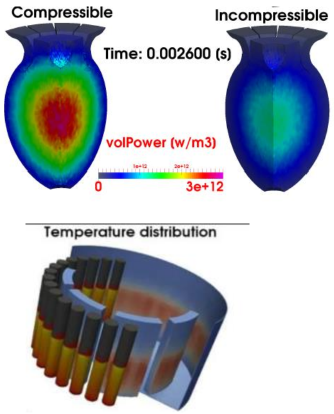
2010-2015
First widespread use

2015-2023
First coordinated and persistent developments

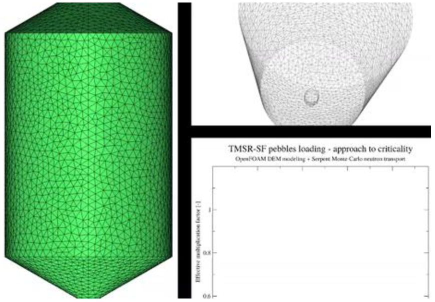


*Pebble bed and prismatic HTGRs*

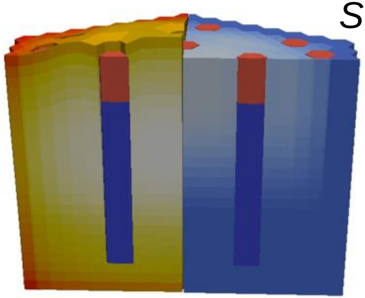
*MSRs*

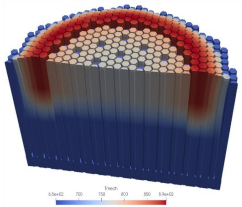


*FHRs*

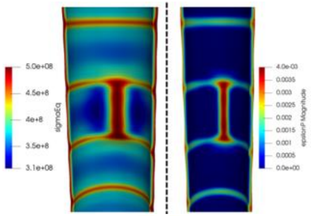

TMSR-SF pebbles loading - approach to criticality
OpenFOAM DEM modeling + Serpent Monte Carlo neutron transport
Effective multiplication factor [-]
1
0.8
0.6

*SFRs*

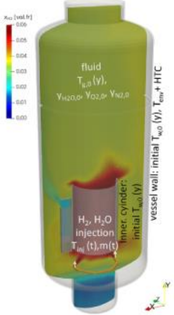




*GeN-Foam*



*Fuel Behaviour (OFFBEAT)*

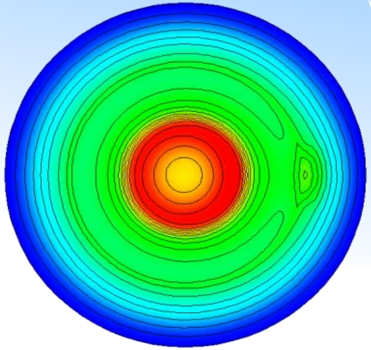

$x_{H2}$ [vol.fr]
0.06
0.05
0.04
0.03
0.02
0.01
0.00
fluid
$T_{g,0}(y)$,
$Y_{H2O,0}$, $Y_{O2,0}$, $Y_{N2,0}$
$H_2$, $H_2O$ injection
$T_{inj}(t)$, m(t)
inner cylinder: initial $T_{w0}(y)$
vessel wall: initial $T_{w0}(y)$, $T_{env}$ + HTC

*Containment Flows containmentFoam*

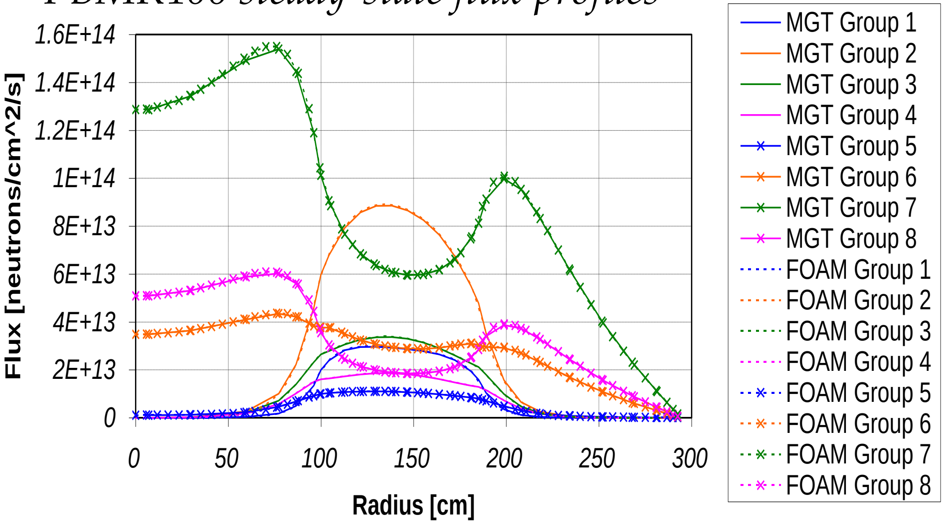
# Pebble Bed HTGR Modelling (PBMR)

- First known attempt to model reactor multi-physics using OpenFOAM
- Goal to develop next generation pebble bed HTGR solver
  - Fully 3D, unstructured mesh, parallelised, extensible
  - 3D multi-group diffusion
  - Delayed neutrons
  - Xenon/Samarium
  - CFD-like modelling of fluid
- Key question whether OpenFOAM could handle time-dependent multi-group neutron diffusion in HTGRs…

*Flux shift following control rod ejection*

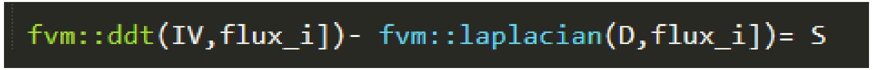


*PBMR400 steady-state flux profiles*



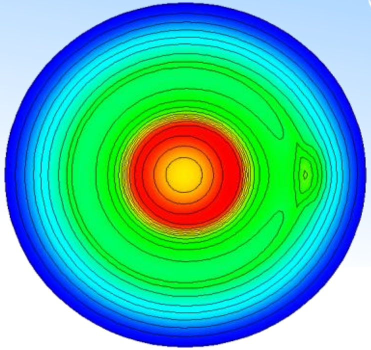
# Pebble Bed HTGR Modelling (PBMR)

- First known attempt to model reactor multi-physics using OpenFOAM
- Goal to develop next generation pebble bed HTGR solver
  - Fully 3D, unstructured mesh, parallelised, extensible
  - 3D multi-group diffusion
  - Delayed neutrons
  - Xenon/Samarium
  - CFD-like modelling of fluid
- Key question whether OpenFOAM could handle time-dependent multi-group neutron diffusion in HTGRs…
- … with a positive answer:
  - Seamless implementation of equations
  - Stable solution (segregated approach, or possibility of matrix-coupled approach thanks to foam-extend)

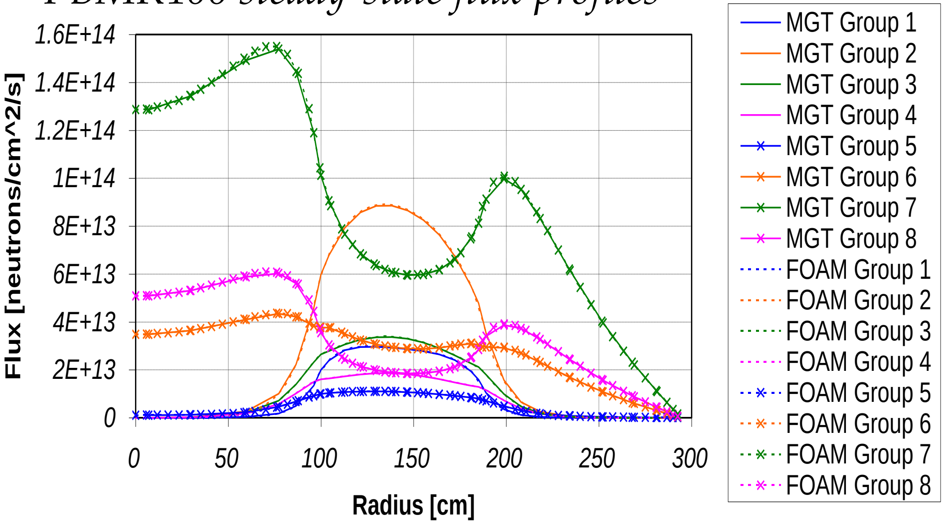


```
fvm::ddt(IV,flux_i])- fvm::laplacian(D,flux_i])= S
```

*Flux shift following control rod ejection*



*PBMR400 steady-state flux profiles*



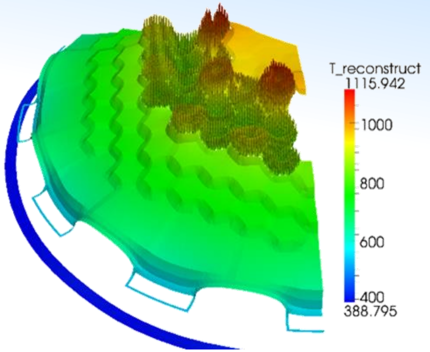
# Prismatic HTGR (Penn State Univ.)

Multi-scale thermal conduction

- Homogenization of subscale models with capability of reconstructing temperature down to TRISO particle level
- Subscale response using reduced order models (ROMs)

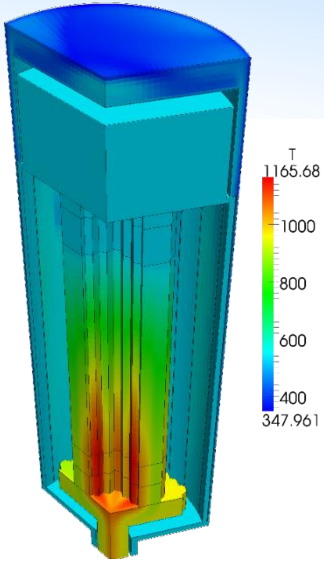
CFD-like approaches applied to heat transfer and fluid flow in prismatic HTGRs

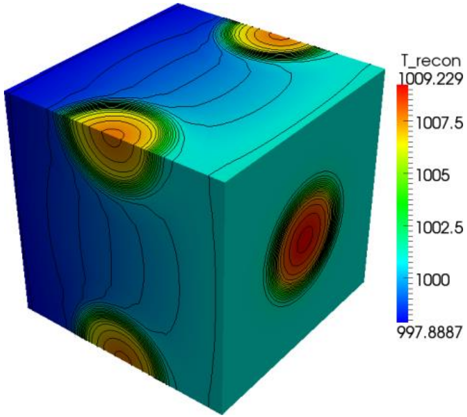
- Porous medium flow: RANS with porosity terms; modified discretization to treat domain discontinuities; turbulence modelling in porous media



*ROM reconstructed temperatures in core*

*Full-core coarse-mesh thermal-hydraulics*





*ROM reconstructed temperatures in TRISO coated particles*

# Prismatic HTGR (Penn State Univ.)

Multi-scale thermal conduction

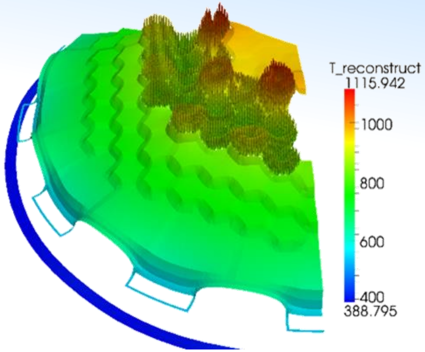
- Homogenization of subscale models with capability of reconstructing temperature down to TRISO particle level
- Subscale response using reduced order models (ROMs)

CFD-like approaches applied to heat transfer and fluid flow in prismatic HTGRs

- Porous medium flow: RANS with porosity terms; modified discretization to treat domain discontinuities; turbulence modelling in porous media

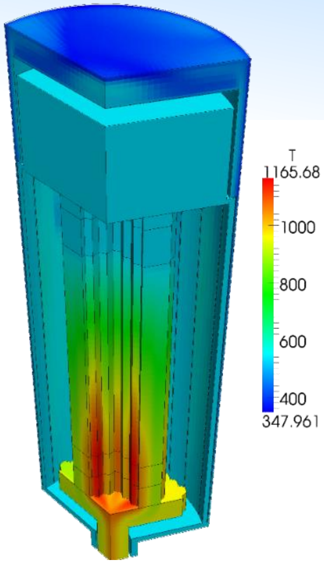
Benefits of OpenFOAM

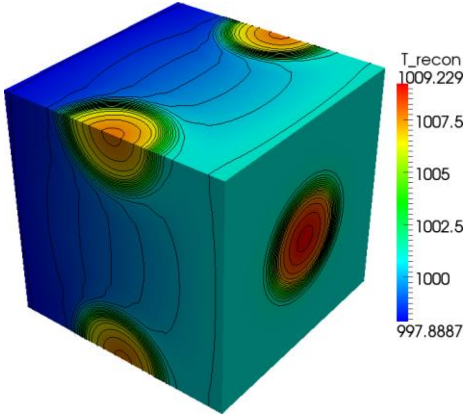
- Existing CFD solvers (incl. turbulence)
- Easy tailoring of equations
- Available functionalities (multi-mesh, multi-zone, ODE, POD, …)
- Streamlined modification of discretization schemes



*ROM reconstructed temperatures in core*

*Full-core coarse-mesh thermal-hydraulics*





*ROM reconstructed temperatures in TRISO coated particles*

# Porous-medium thermal-hydraulics: governing equations

- The coarse-mesh governing equations for a region with uniform porosity:

$$\frac{\partial \gamma\rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{u}_{\mathbf{D}}) = 0$$

$$\frac{\partial \rho\boldsymbol{u}_{D}}{\partial t} + \frac{1}{\gamma}\nabla\cdot(\rho\boldsymbol{u}_{\mathbf{D}}\otimes\boldsymbol{u}_{\mathbf{D}})$$

$$= \nabla\cdot(\mu_T\nabla\boldsymbol{u}) - \gamma\nabla p + \gamma\boldsymbol{F}_{\mathbf{g}} + \gamma\boldsymbol{F}_{\mathbf{ss}} - (\rho\boldsymbol{u}_{\mathbf{D}}\otimes\boldsymbol{u}_{\mathbf{D}})\nabla\frac{1}{\gamma}$$

$$\frac{\partial \gamma\rho e}{\partial t} + \nabla\cdot(\boldsymbol{u}_{\mathbf{D}}(\rho e + p))$$

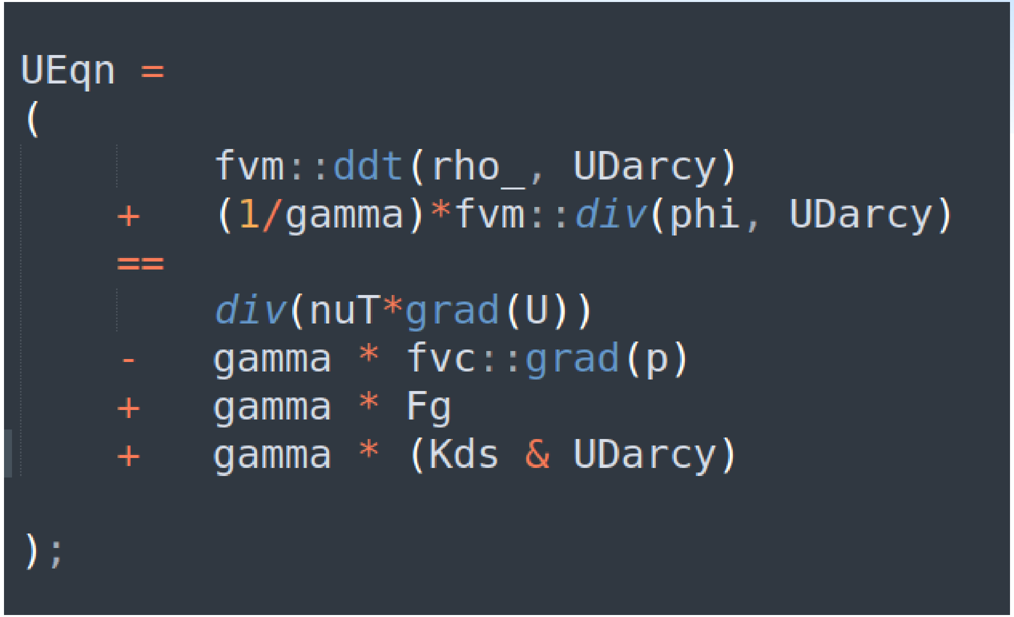
$$= \gamma\nabla\cdot(k_T\nabla T) + \boldsymbol{F}_{\mathbf{ss}}\cdot\boldsymbol{u}_{\mathbf{D}} + \gamma\dot{Q}_{ss} + (k_T\nabla T)\cdot\nabla\gamma$$

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

# Porous-medium thermal-hydraulics: governing equations

Ideal situation…

$$\frac{\partial \rho \boldsymbol{u}_D}{\partial t} + \frac{1}{\gamma}\nabla\cdot(\rho \boldsymbol{u}_{\mathbf{D}} \otimes \boldsymbol{u}_{\mathbf{D}}) = \nabla\cdot(\mu_T \nabla \boldsymbol{u}) - \gamma\nabla p + \gamma \boldsymbol{F}_{\mathbf{g}} + \gamma \boldsymbol{F}_{\mathbf{ss}}$$

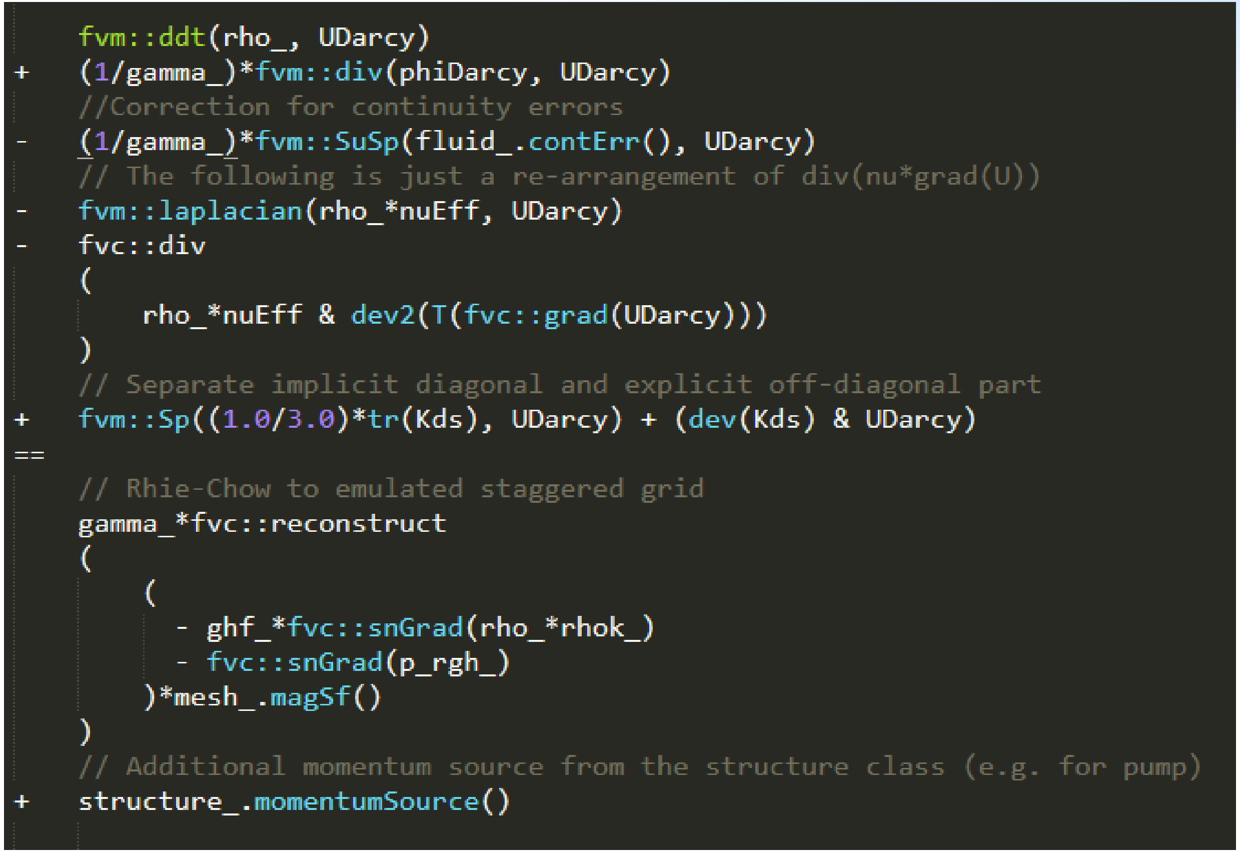


```
UEqn =
(
        fvm::ddt(rho_, UDarcy)
    +   (1/gamma)*fvm::div(phi, UDarcy)
    ==
        div(nuT*grad(U))
    -   gamma * fvc::grad(p)
    +   gamma * Fg
    +   gamma * (Kds & UDarcy)

);
```

# Porous-medium thermal-hydraulics: governing equations

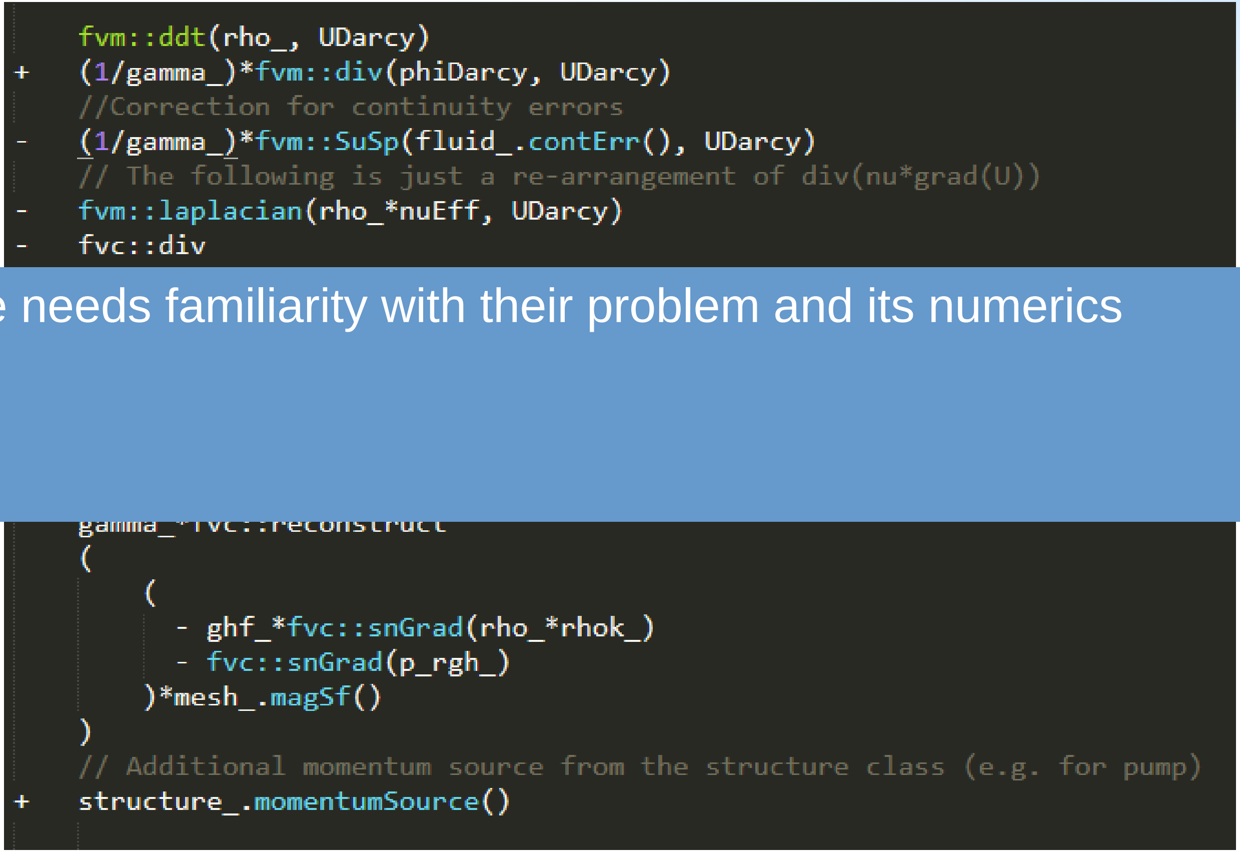
In practice…



```
    fvm::ddt(rho_, UDarcy)
+   (1/gamma_)*fvm::div(phiDarcy, UDarcy)
    //Correction for continuity errors
-   (1/gamma_)*fvm::SuSp(fluid_.contErr(), UDarcy)
    // The following is just a re-arrangement of div(nu*grad(U))
-   fvm::laplacian(rho_*nuEff, UDarcy)
-   fvc::div
    (
        rho_*nuEff & dev2(T(fvc::grad(UDarcy)))
    )
    // Separate implicit diagonal and explicit off-diagonal part
+   fvm::Sp((1.0/3.0)*tr(Kds), UDarcy) + (dev(Kds) & UDarcy)
==
    // Rhie-Chow to emulated staggered grid
    gamma_*fvc::reconstruct
    (
        (
          - ghf_*fvc::snGrad(rho_*rhok_)
          - fvc::snGrad(p_rgh_)
        )*mesh_.magSf()
    )
    // Additional momentum source from the structure class (e.g. for pump)
+   structure_.momentumSource()
```

# Porous-medium thermal-hydraulics: governing equations

In practice…



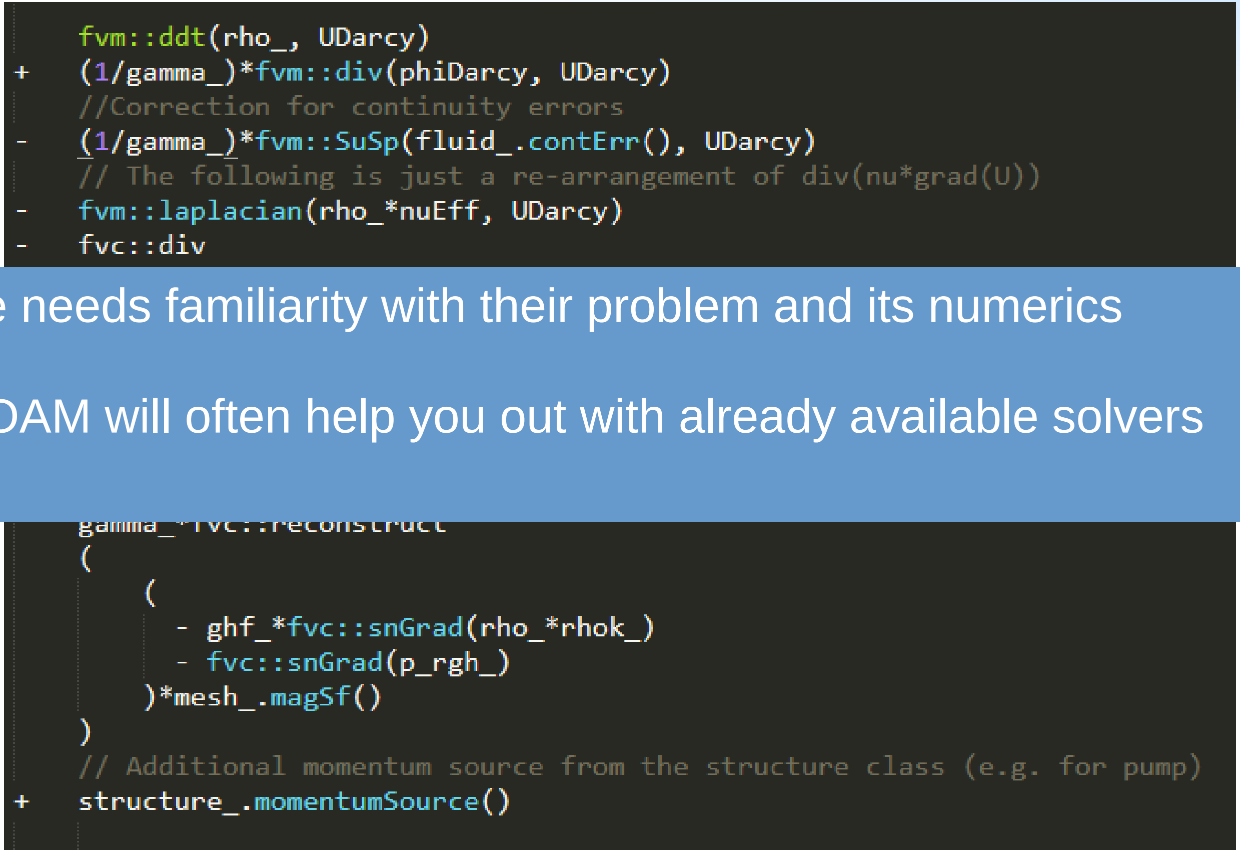
```
    fvm::ddt(rho_, UDarcy)
+   (1/gamma_)*fvm::div(phiDarcy, UDarcy)
    //Correction for continuity errors
-   (1/gamma_)*fvm::SuSp(fluid_.contErr(), UDarcy)
    // The following is just a re-arrangement of div(nu*grad(U))
-   fvm::laplacian(rho_*nuEff, UDarcy)
-   fvc::div
```

One needs familiarity with their problem and its numerics

```
    gamma_*fvc::reconstruct
    (
        (
          - ghf_*fvc::snGrad(rho_*rhok_)
          - fvc::snGrad(p_rgh_)
        )*mesh_.magSf()
    )
    // Additional momentum source from the structure class (e.g. for pump)
+   structure_.momentumSource()
```

# Porous-medium thermal-hydraulics: governing equations

In practice…



```
    fvm::ddt(rho_, UDarcy)
+   (1/gamma_)*fvm::div(phiDarcy, UDarcy)
    //Correction for continuity errors
-   (1/gamma_)*fvm::SuSp(fluid_.contErr(), UDarcy)
    // The following is just a re-arrangement of div(nu*grad(U))
-   fvm::laplacian(rho_*nuEff, UDarcy)
-   fvc::div

    gamma_*fvc::reconstruct
    (
        (
          - ghf_*fvc::snGrad(rho_*rhok_)
          - fvc::snGrad(p_rgh_)
        )*mesh_.magSf()
    )
    // Additional momentum source from the structure class (e.g. for pump)
+   structure_.momentumSource()
```

One needs familiarity with their problem and its numerics

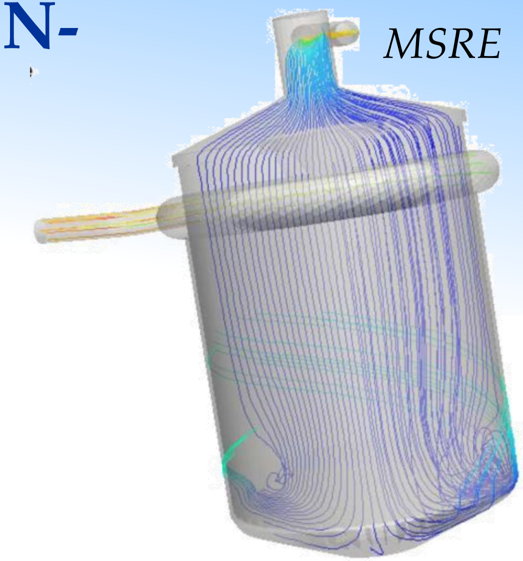
OpenFOAM will often help you out with already available solvers

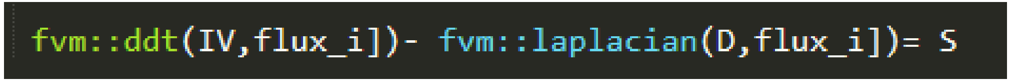
# MSR modelling (PoliMi -> CNRS / GeN-Foam)

- Among the first fully-fledged multi-physics solvers for MSRs
- A reference today for the MSR community

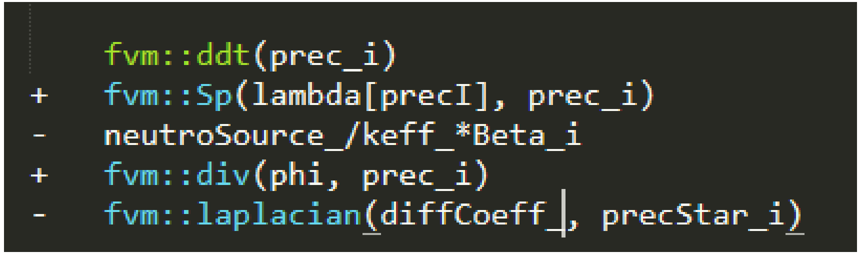
- Benefits of OpenFOAM
  - Available CFD solvers
  - Arbitrary geometries
  - Streamlined implementation of diffusion and DNP equations

*MSRE*



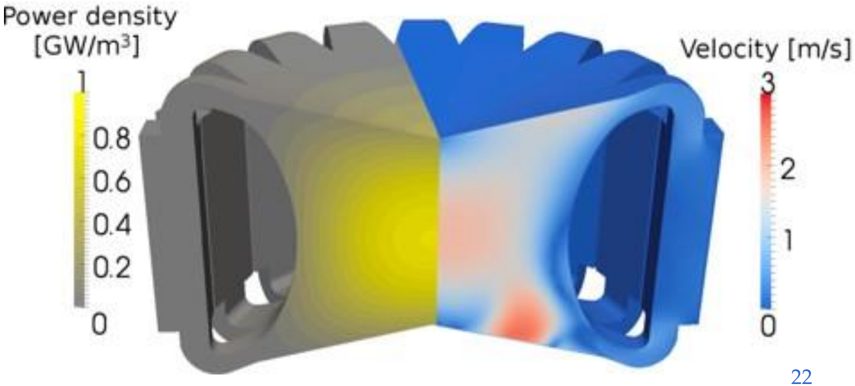


```
fvm::ddt(IV,flux_i])- fvm::laplacian(D,flux_i])= S
```



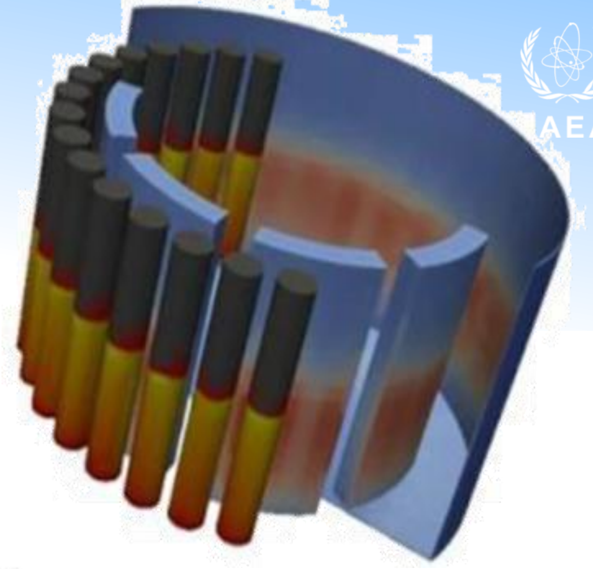
```
    fvm::ddt(prec_i)
+   fvm::Sp(lambda[precI], prec_i)
-   neutroSource_/keff_*Beta_i
+   fvm::div(phi, prec_i)
-   fvm::laplacian(diffCoeff_, precStar_i)
```

*MSFR*

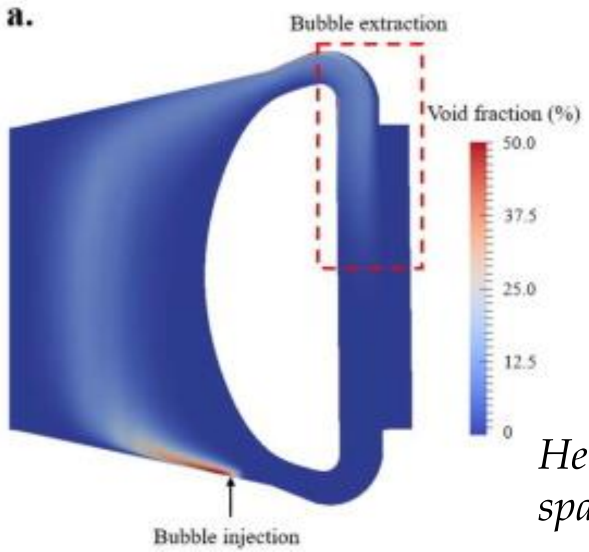


# MSR modelling: advanced

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



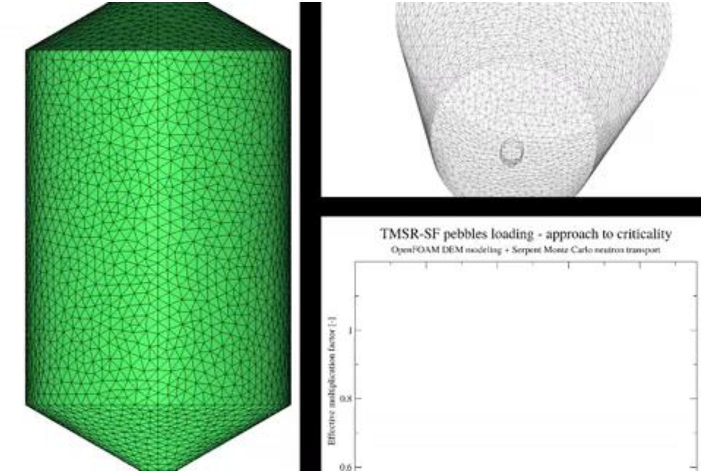
*Dump tanks*



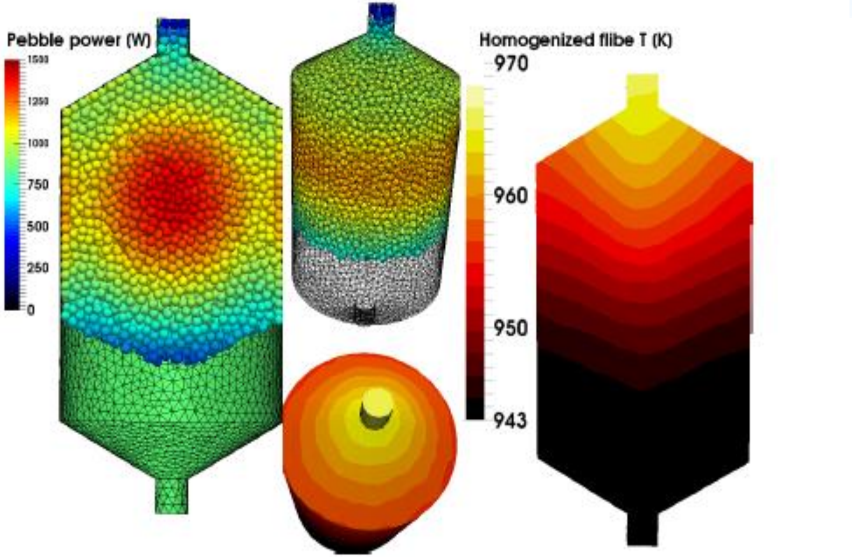
*Helium sparging*

# Fluoride Salt-Cooled High-Temperature Reactor (FHR, 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

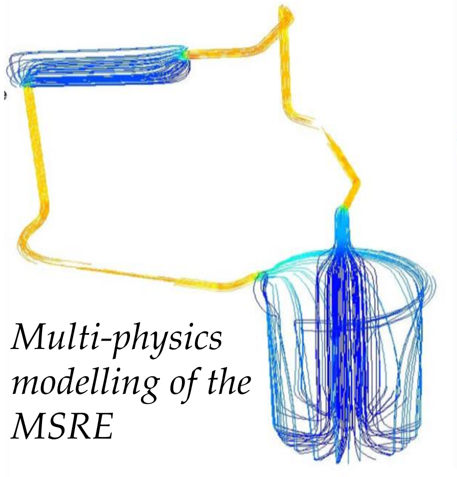
*Approach to criticality*



*Coupled DEM and porous-medium solution for thermal-hydaulics*

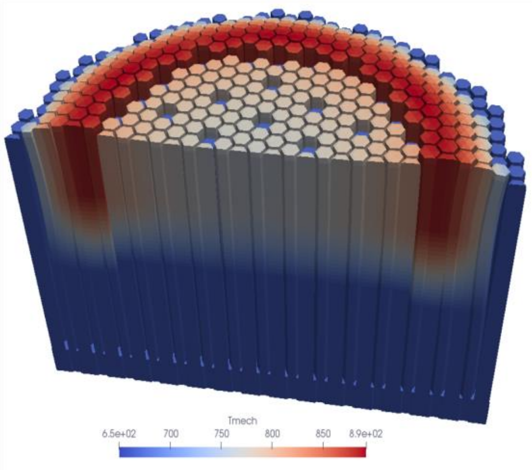
# GeN-Foam: Generalized Nuclear Field operation and manipulation

- First general solver for reactor safety based on OpenFOAM

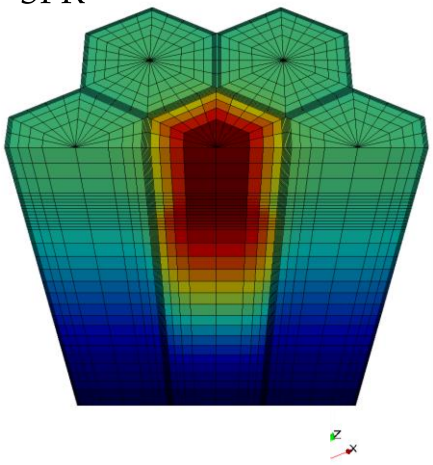


*Multi-physics modelling of the MSRE*

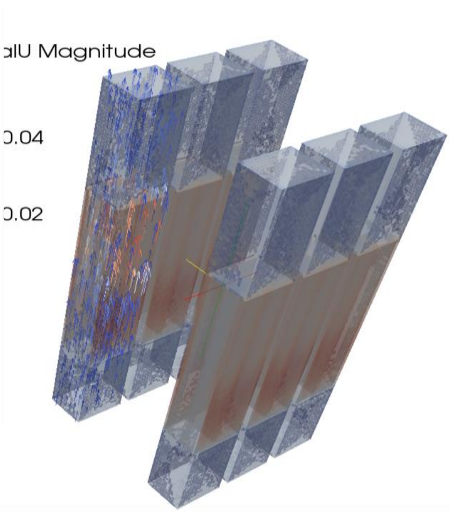
*Core flowering in a SFR*



*Assembly windows in a SFR*



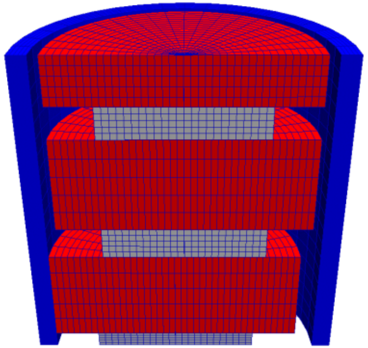
*The Argonaut reactor*

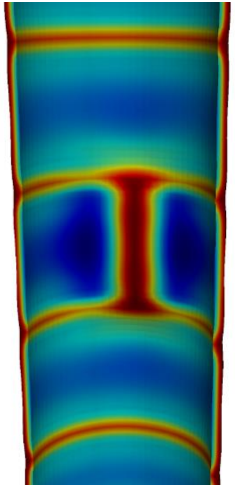


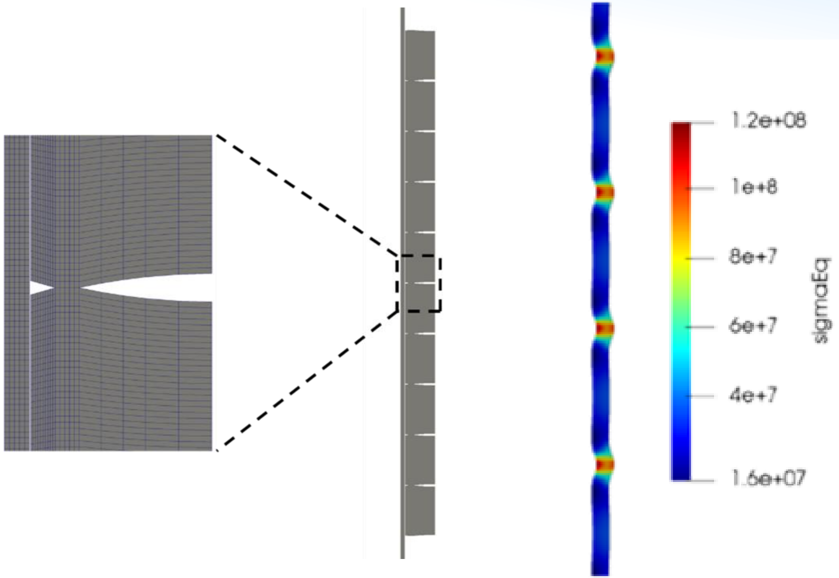
- Benefits of OpenFOAM
  - Open-source + object-oriented -> use of previous work
  - Available CFD solvers
  - Available thermo-mechanics solver
  - Multi-mesh with projection algorithms
  - Multi-material
  - Mesh deformations
  - ….

# OFFBEAT: OpenFoam Fuel BEhavior Tool

- Fuel thermo-mechanics with finite volumes: from  a wild idea to a multi-dimensional solver for fuel behavior included in several Euratom project (in 5 years!)



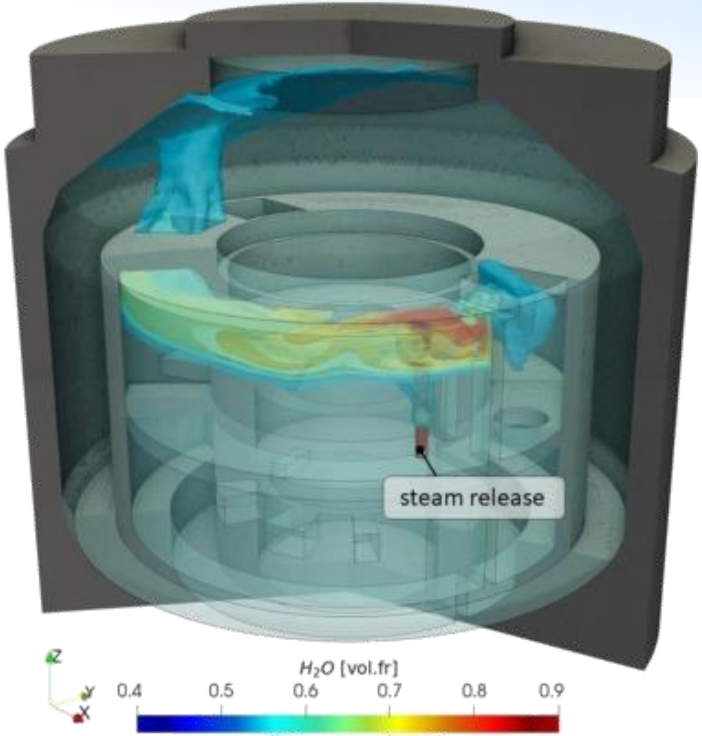




- Benefits of OpenFOAM
  - Use of community contributions (solid4foam)
  - Region-coupled boundaries and AMI
  - Multi-material (cellZones)
  - Object-oriented programming to streamline inclusion of correlations
  - …

# HPC-oriented containment analysis - containmentFoam

- From a general CFD tool to a next-generation tool for containment analysis
- Benefits of OpenFOAM
  - Available solvers (incl. Monte Carlo radiative heat transfer!)
  - Turbulence models
  - Conservative formulation
  - Parallel scalability
  - …



*ISP-37 VANAM-M3 experiment with containmentFOAM*

# Lessons Learned

With a bit of ingenuity and imagination,

one can model pretty much everything…

# Lessons Learned

What's the effort?

What competences do I need?

How do I approach the problem?

What about the license?

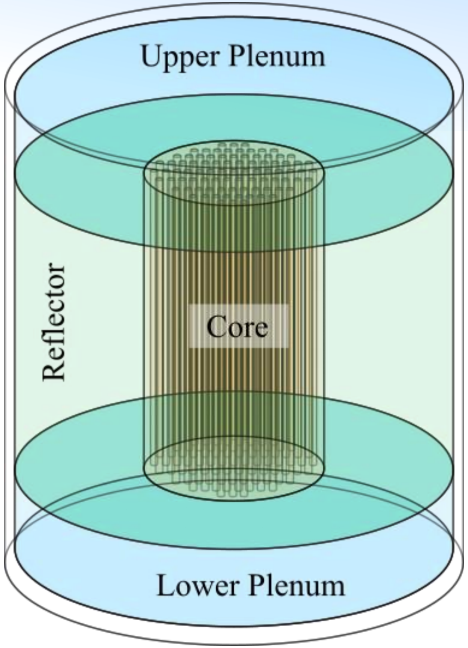
What is the quality of the result?

# How to Approach the Problem

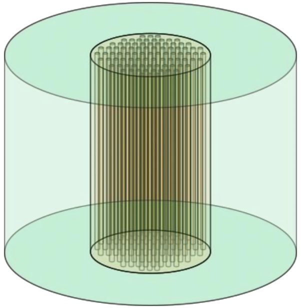
Let's consider some hypothetical reactor

- Monolithic block core with coolant channels
- Lower and upper plena
- RPV

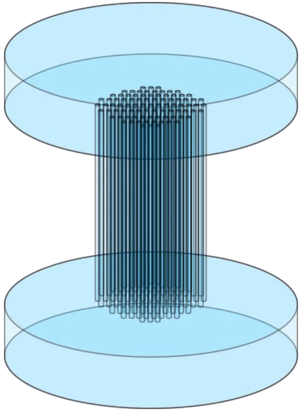
We want to model thermal-hydraulics coupled to 3D kinetics



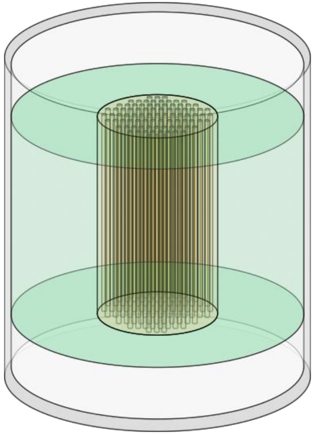
# How to Approach the Problem



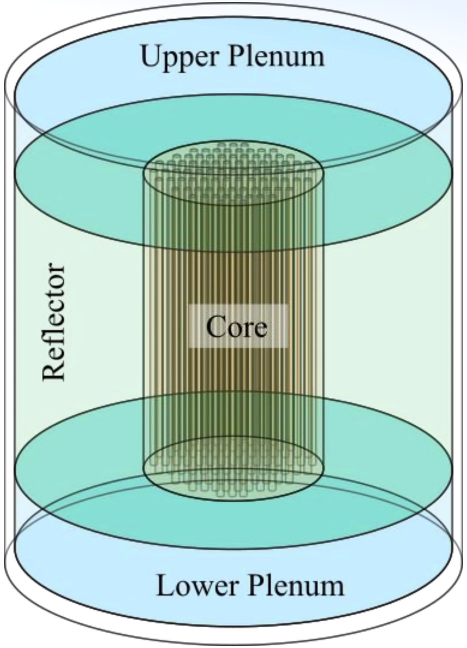
Neutronic Domain



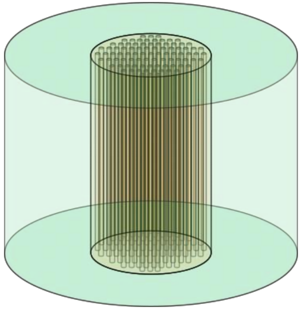
Coolant



Solid Structures



# How to Approach the Problem

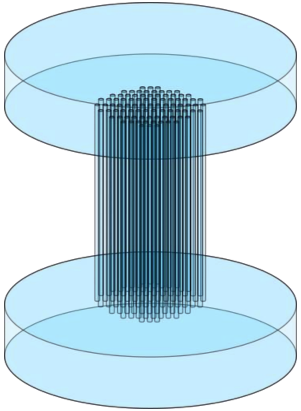


**Neutronics Domain**

Neutronics **mesh**

**Fields**:
Cross-sections, fluxes, DN precursors, power

**Equations**:
neutron diffusion, delayed neutron production/decay

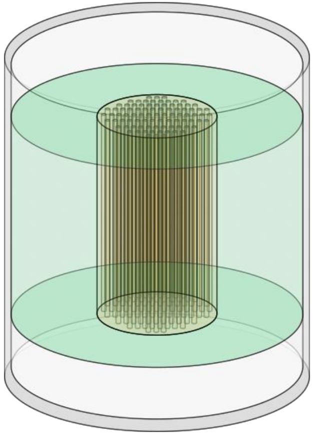


**Coolant**

Coolant **mesh** (porous?)

**Fields**:
Velocity, Pressure, Temperature, thermophysical properties

**Equations**: RANS (porous?)



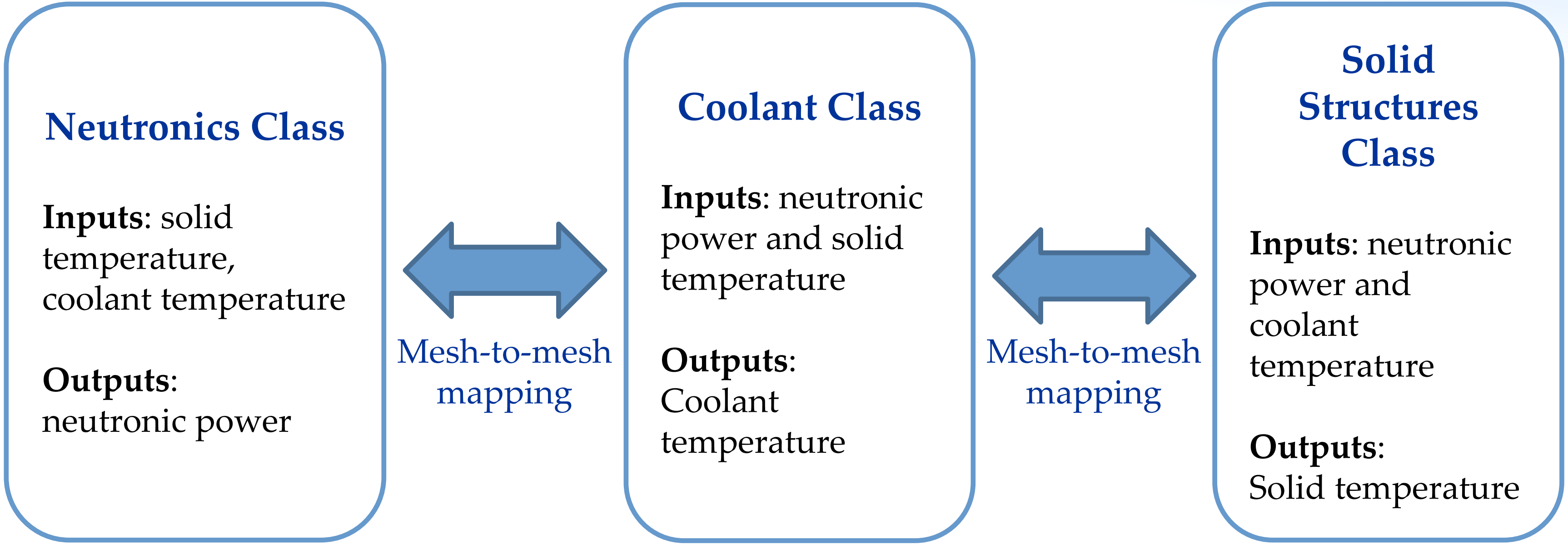
**Solid Structures**

Solid **mesh** (porous?)

Fields: Temperature, thermophysical properties

**Equations**:
Heat conduction (porous?)

# How to Approach the Problem



### Neutronics Class

**Inputs**: solid temperature, coolant temperature

**Outputs**: neutronic power

Mesh-to-mesh mapping

### Coolant Class

**Inputs**: neutronic power and solid temperature

**Outputs**: Coolant temperature

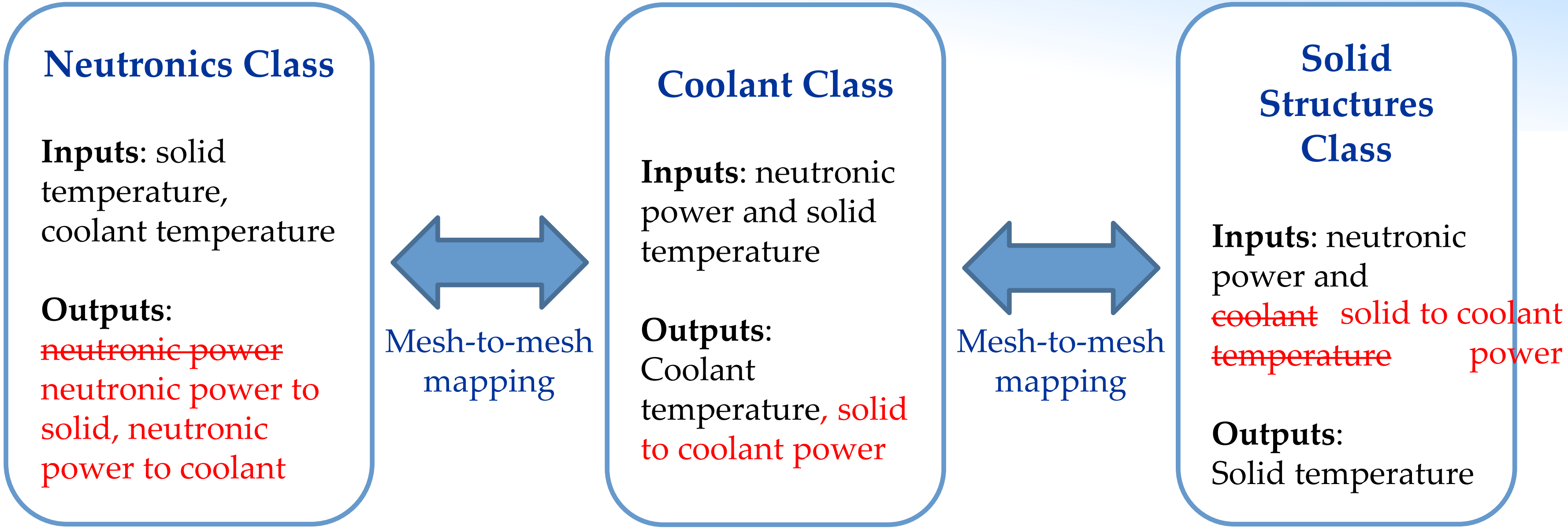
Mesh-to-mesh mapping

### Solid Structures Class

**Inputs**: neutronic power and coolant temperature

**Outputs**: Solid temperature

# How to Approach the Problem



**Neutronics Class**

**Inputs**: solid temperature, coolant temperature

**Outputs**: ~~neutronic power~~ neutronic power to solid, neutronic power to coolant

Mesh-to-mesh mapping

**Coolant Class**

**Inputs**: neutronic power and solid temperature

**Outputs**: Coolant temperature, solid to coolant power

Mesh-to-mesh mapping

**Solid Structures Class**

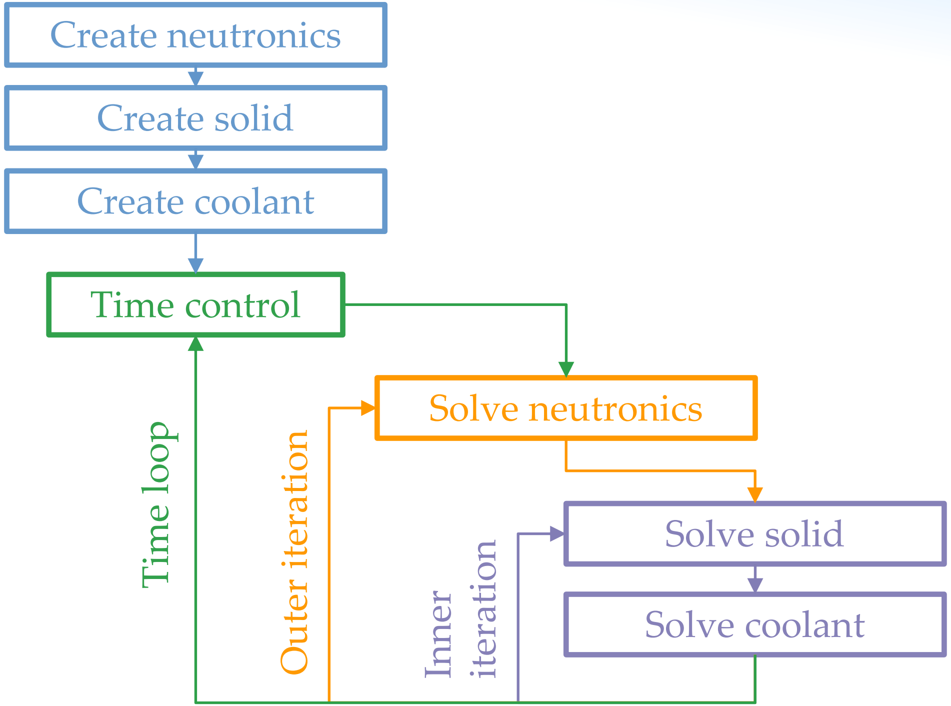
**Inputs**: neutronic power and ~~coolant temperature~~ solid to coolant power

**Outputs**: Solid temperature

**In reality it's a bit more complicated than this…**

- The class API needs to match the physical and numerical requirements
- Each class may need to contain nested classes (e.g. cross-sections, thermophysical properties, heat transfer correlations)

# How to Approach the Problem



Create neutronics

Create solid

Create coolant

Time control

Solve neutronics

Solve solid

Solve coolant

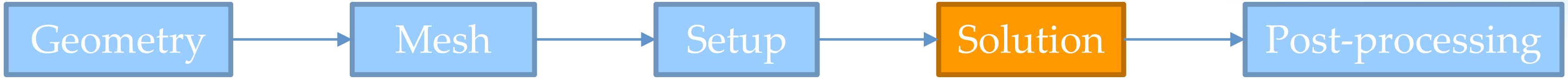
# License

**GNU GPL v3 license**

- Copyleft type license: automatically affects derivative works
  - If you develop a code based on OpenFOAM, you cannot distribute it without including the source code
- Favors a collaborative development with minimal work duplication
- Can limit investments from commercial players

# OpenFOAM Workflow

Workflow mirrors that of traditional CFD workflow



**Downsides**

- No official graphical user interface
- Meshing, pre-processing and post-processing are performed with separate tools
- Geometry preparation and meshing often require proprietary tools
- Requires familiarity with Linux
- Documentation often scattered
- Steep learning curve (please don’t use as a black-box)

**Advantages**

- Transparent
- Access to source code

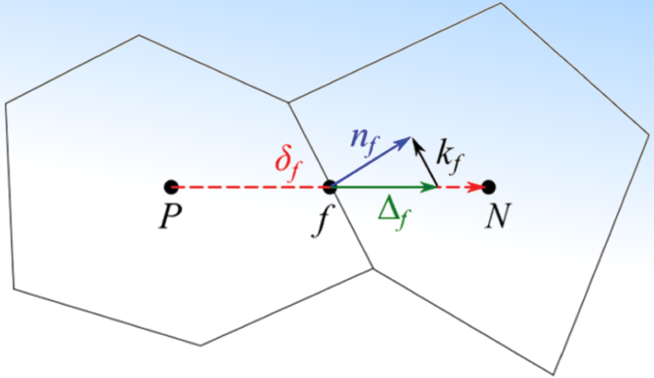
Better integration of application and development

# Structure of the base library

- Very complete
  - Discretization and linear system solution
  - Mesh-to-mesh projections
  - Mesh deformation
  - Mesh manipulation
  - Dense matrix algebra
  - Ordinary differential equations
  - Monte Carlo methods (Direct simulation Monte Carlo solver for transient, multi-species flows + molecular dynamics solver for fluid dynamics)
  - Octree-based mesh search
  - Proper orthogonal decomposition (foam-extend)
  - Built-in (e.g., multi-application coupling) and third-party (e.g., PRECICE) code coupling functionalities
  - …

- Object oriented
  - Data encapsulation
  - Multi-level API

# Finite volumes



**Pros**:

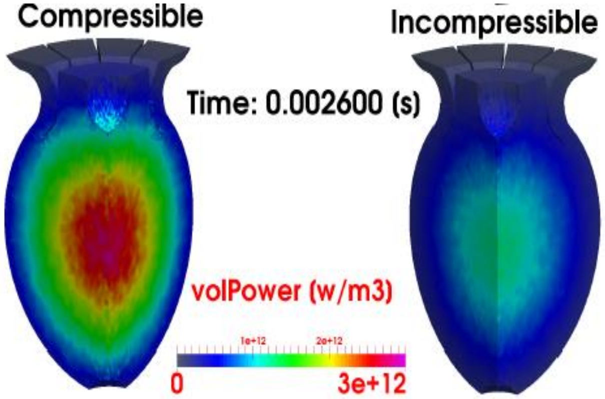
- Flexible
- Scalable
- Intuitive
- Mathematically conservative formulation
- Ideal for convection-driven problems; CFD-friendly
- Ok for diffusion problems; thermo-mechanics and neutron diffusion
- Generally yield sparse diagonally dominant matrices; fast efficient matrix solution

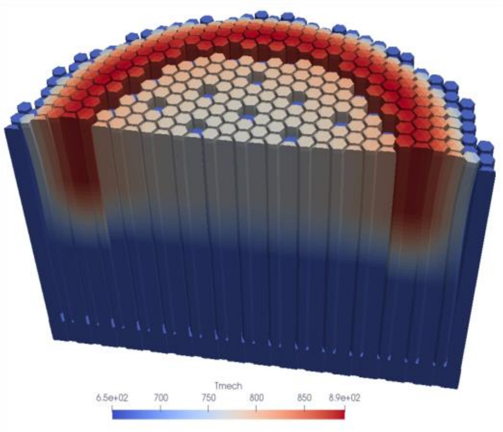
**Cons**:

- Require good quality meshes (non-orthogonality, skewness, aspect ratio, etc.)
- Max second order accuracy in space
- First order elements, with flat faces → high mesh resolution needed for curved surfaces
- Users require familiarity with concepts associated with PDEs (well-posed problems, initial and boundary conditions), geometry creation, meshing, discretization, linear solution, etc.

# Unstructured meshes

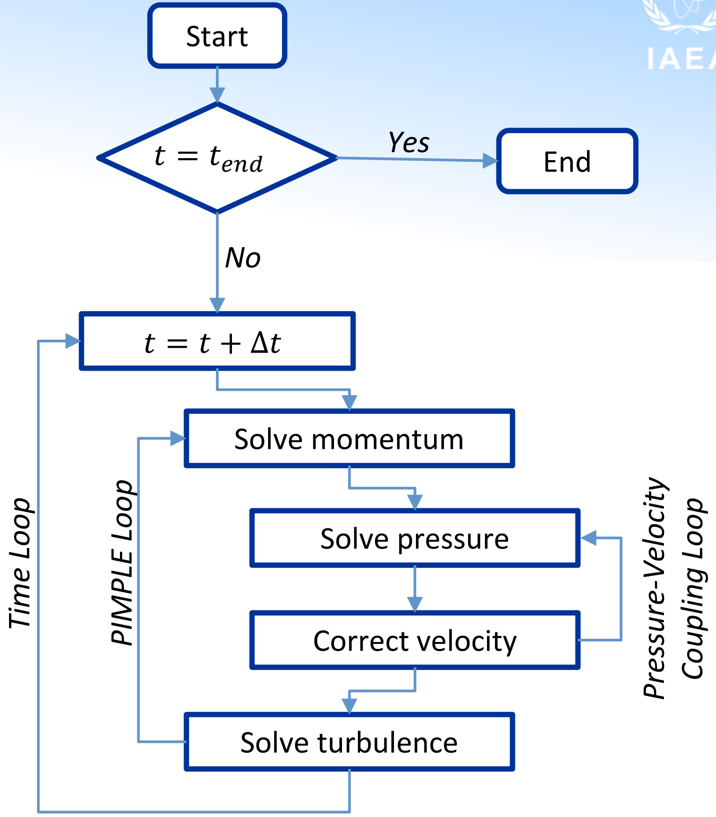
- Complete flexibility in terms of geometry
  - Appropriate for non-traditional reactor designs and complex components
- All cells are 3D
  - 1D and 2D meshes can be mimicked, but…
  - Requires one to think out of the box in some cases, e.g. 1D pipes, thin gaps.
- Higher computational footprint than, for example, fixed rectangular grids





# Operator-splitting

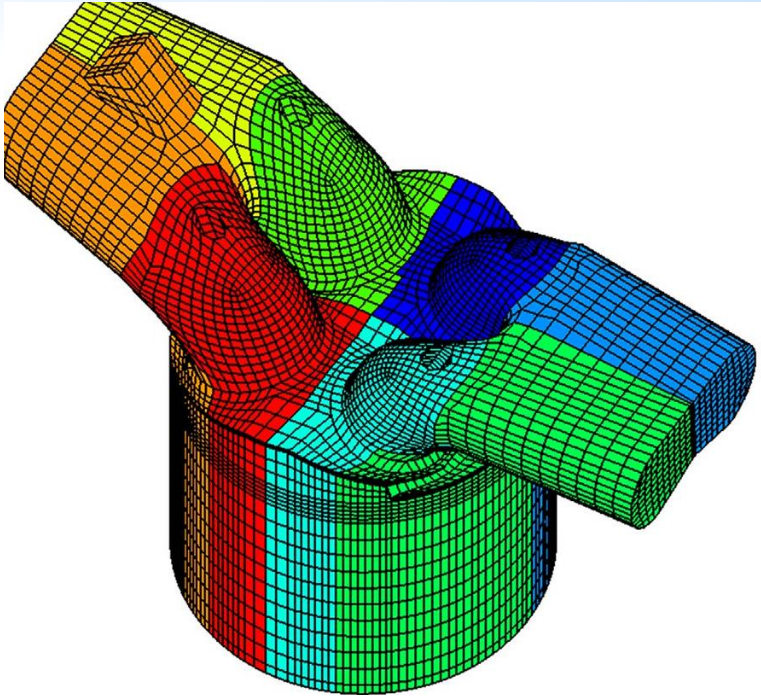
- One matrix for each equation + fixed point iteration
  - Equation coupling terms treated explicitly
- Pros
  - Easier preconditioning and optimal choice of solution method
  - No need to solve all physics at once
  - Simpler development and easier to debug; focus on one equation at a time.
- Cons
  - Can be slow to converge for weakly-coupled / strongly non-linear equations
  - Can be unstable for stiff problems, requiring numerical tricks to get a stable solution



**PIMPLE Algorithm in OpenFOAM**

# Parallelization

- Domain decomposition using MPI
- Optimally scales up to thousands of CPU cores
- Some bottlenecks (common to most FEM and FVM solvers)
  - the sub-optimal sparse matrices storage format (LDU) that does not enable any cache-blocking mechanism (SIMD, vectorization)
  - I/O can be limiting for very large problems
- The OpenFOAM HPC Technical Committee is currently working on the limitations
  - interface to external linear algebra libraries
  - recent work from NVIDIA
  - ongoing Horizon2020 exaFoam project



# 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 neutron transport -> several thousand solution fields -> 200 GB of memory per million cells

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