Multiphysics simulations of nuclear reactors and more
Gothenburg Region OpenFOAM® User Group Meeting

Klas Jareteg

klas.jareteg@chalmers.se
Division of Nuclear Engineering
Department of Applied Physics
Chalmers University of Technology

November 12, 2014
Outline

- Multiphysics and multiscale in Light Water Reactors (LWRs)
- Neutronics and thermal-hydraulics in OpenFOAM®:
  - Multiple mesh methodology; mapping and parallelization
  - Handling large sets of input parameters
  - Example of a simplified Pressurized Water Reactor assembly

and more ...
  - Coupled pressure and velocity solver in foam-extend-3.1
Multiscale systems
Multiphysics systems

Light Water Reactor (LWR) Multiphysics

Fuel reaction probabilities:
- Temperature and density coupling

Power density:
- Energy source in fuel temperature

Temperatures:
- Dependence between water and fuel properties

Water cross-sections:
- Water density coupling
FIRE - Fine mesh deterministic REactor modelling

Development of a fine mesh computational tool for nuclear fuel bundles:

- integrated approach for solving neutronics and thermal-hydraulics
- single and two-phase flow models based on first principles
- high-resolution coupling on fine meshes using HPC
- fuel bundle size calculations, ultimately coupled to coarse mesh solvers

Figure: Power density distribution at fuel pins.

Financing:
- SKC - Swedish Center for Nuclear Technology

Supervisors:
- Christophe Demazière, Paolo Vinai, Srdjan Sasic
Neutronics solver

**Steady-state eigenvalue problem**
- Eigenvalue problem solved by power iteration method
- Diffusion solver based on multiple groups:

\[ \nabla \cdot (D_g \nabla \Phi_g) + \Sigma_{T,g} \Phi_g = S_g + \frac{1}{k} F_g \]  \hspace{1cm} (1)

  - Fast but limited accuracy for fine-mesh with heterogeneous parameters
  - Implemented using a block matrix formulation

- Discrete ordinates method, multiple groups and multiple directions:

\[ \Omega_m \cdot \nabla \Psi_{m,g} + \Sigma_{T,g} \Psi_{m,g} = S_{m,g} + \frac{\chi_g}{k} F_g \]  \hspace{1cm} (2)

  - More accurate, but computationally heavy
  - Transport method required for fine mesh problems

**More information:**
Block matrix systems in OFW9 training, Zagreb, 2014: "Block coupled matrix solvers in foam-extend-3", Klas Jareteg and Ivor Clifford
Neutronics solver - Discrete ordinates method

Implementation of discrete ordinates:

- Standard matrix solvers potentially inefficient
- Following ordinate $\Omega_m$, a direct sweep through the domain allows $\Psi_{m,g}$ to be computed in one sweep
- Sweeping order of the cells calculated at initialization
- Parallelization handled as in GaussSeidel solvers (corresponding to full boundary exchange after finishihed sweep)

Code: steadyDOMSweeNeutronics.C

```
for (register label cellI=0; cellI<nCells; cellI++)
{
    // Modification for ordered sweep
    register label cellJ = sweepOrder[cellI];
```

**Figure:** Example of a sweep ordering for an unstructured mesh
Thermal-hydraulics single-phase solver

**Steady state single phase solver:**

- Approach equivalent to `simpleFoam`
- Buoyancy taken into account for density variation from heating
- Generally slow convergence for pressure and velocity

**Figure:** Temperature distribution in coolant around a single fuel pin.

**More information:**
Presentation on coupled pressure and velocity solver from OFW9, Zagreb, 2014: "puCoupledFoam - an open source coupled incompressible pressure-velocity solver based on foam-extend" Klas Jareteg, Vuko Vukcevic, Hrvoje Jasak
Thermal-hydraulics: conjugate heat transfer

- Multiregion approach, separate meshes for all solid and liquid regions
- Implicit heat transfer by using regionCouple
  - Explicit iteration over all regions not an alternative, too many regions
- Temperature dependent thermophysical properties
  - Based on table interpolation, inheriting basicThermo
Thermal-hydraulics two-phase solver

- Two-phase flow model under development to extend solver to Boiling Water Reactors (BWRs)
- Large variety of complex behavior to be covered
- Solver based on a population balance methods

**Figure:** Void fraction, bubble mean diameter and axial velocity
Steady state coupling algorithm

- Explicit scheme, coupled convergence iteration
- Neutronics and thermal-hydraulic modules implemented as classes. Easy testing of different implementations using inheritance and run-time choosable mechanism

**Figure:** Iterative coupled scheme. [1]
Class structure

- Solvers for each field written as objects
- Direct inheritance allows for different models to be tested in the coupled scheme

Figure: Simplified class structure of coupled solver
Transient multiphysics calculations
Master thesis project by Rasmus Andersson

- Currently developing a transient multiphysics solver
- Thermal-hydraulic solvers based on PISO algorithm, with variable density
- Neutronics solver based on diffusion solver
Multiple meshes - Generation and setup

- Repeating structure of fuel pins allows for repeated mesh
- Automatic setup of region coupled interfaces for the thermal-hydraulics

**Figure:** Mesh example for a quarter of a fuel pin
Multiple meshes - Mapping I

Different meshes in different regions:

- Finer mesh in the liquid coolant than in neutronics in general
- Mapping algorithm needed for data exchange:

Figure: Example of mapping of two overlapping meshes. [1]

- Fully conservative transfer of extensive quantities
- Geometric overlaps computed equivalent to GGI face overlaps, extended to 3D
Multiple meshes - Mapping II

- Mapping handled locally on each CPU

**Code: neuthFoam-2.0.C**

```cpp
// Compute cell intersection between all fuel meshes and neutronics
PtrList<intersectionChecker> iSolid(I);

forAll(iSolid, i)
{
    iSolid.set(i, new intersectionChecker(solidMeshes[i], neutronicsMesh));
}
```

**Code: neuthFoam-2.0.C**

```cpp
// Transfer power distribution from neutronics to fuel meshes
const volScalarField& Pnk = neutronics->P();
PtrList<volScalarField>& PS = thermalhydraulics->P();

forAll(solidMeshes, i)
{
    // Transfer the power field and immediately normalize
    solidMeshes[i].backwardFieldVolumetric(PS[i], Pnk);
}
```
Multiple meshes - Mapping III

- Parallelization: all meshes are split in same planes
- `meshPartitionerRegion` (not yet released) used to generate cell sets.

**Code: meshPartitionerDict**

```plaintext
Directions (1 1 4);
Mode "manual";
X (0.0 1.0);
Y (0.0 1.0);
Z (-100.0 0.4 0.7 1.1 100.0);
```
Neutronics mesh - cross-section handling

- Different cross-sections in different materials
- Handled using cell sets

Figure: Horizontal view of test system. [1]

Code: Cell sets for cross-sections

```plaintext
// Automatically generated cell set for fuel region, input to setSet
cellSet fue\_I0\_J0\_U100\_out new cylinderToCell
   (0.0e+00 0.0e+00 0.0e+00) (0.0e+00 0.0e+00 1.0e−02) 1.340407e−03
...

cellSet fue_I0_J0_U100 new rotatedBoxToCell
   (0.0e+00 0.0e+00 0.0e+00) (1.366667e−03 0.0e+00 0.0e+00)
   (8.368420e−20 1.366667e−03 0.0e+00) (0.0e+00 0.0e+00 1.0e−02)
cellSet fue_I0_J0_U100 delete cellToCell fue_I0_J0_U100_box
```

2 \% UOX
2 \% UOX with 2\% Gd$_2$O$_3$
4 \% UOX
Water, 1000 ppm boron
Gap, helium
Reflective boundary
Example case

- Quarter of $15 \times 15$ system
- PWR conditions
- System height 100 cm ($+2 \times 20$ cm reflector)
- 6,088,000 cells in coolant, 798,000 in neutronics
- Decomposed in 64 domains
- 8 neutron energy groups
- $S_8$ (80 directions)

**Figure:** Horizontal view of test system. [1]
Example case - Results 1

- Problem converged in 14 hours using 64 CPUs (Nehalem Intel® Xeon® E5520, 2.27GHz)
- Multiphysics coupling converged in 8 iterations

Figure: Moderator temperature. [1]
Example case - Results I

Figure: Scalar flux at mid-elevation for the fast group. [1]
Slow convergence in incompressible flow solvers

**Background**
- Generally slow convergence of segregated steady-state incompressible solvers
- A large number of iterations needed to resolve coupling between pressure and velocity
- Coupled solvers potentially give an increased convergence rate

**Applications**
- Single-phase flow around PWR between fuel pins
- Low mach simulations around external surfaces
- ...
Implicit formulation

- Navier-Stokes, incompressible, steady-state equations:
  \[ \nabla \cdot (\mathbf{U}) = 0 \]  
  \[ \nabla \cdot (\mathbf{UU}) - \nabla (\nu \nabla \mathbf{U}) = -\frac{1}{\rho} \nabla p \]  

- with the semi-discretized form:
  \[ \sum_{\text{faces}} \mathbf{U}_f \cdot \mathbf{S}_f = 0 \]  
  \[ \sum_{\text{faces}} [(\mathbf{UU} - \nu \nabla \mathbf{U})_f \cdot \mathbf{S}_f = - \sum_{\text{faces}} P_f \mathbf{S}_f \]  

- Rhie-Chow interpolation in the continuity equation:
  \[ \sum_{\text{faces}} \left[ \mathbf{U}_f - \overline{\mathbf{D}}_f \left( \nabla P_f - \overline{\nabla P_f} \right) \right] \cdot \mathbf{S}_f = 0 \]  

a coupled system of four equations is formulated.
pUCoupledFoam - Coupled incompressible solver

**Coupled solver**
- Coupled solver implemented in foam-extend-3

**Example case**
- Structured mesh, 4800 cells, laminar case
- Comparison of SIMPLE algorithm and coupled algorithm

*Figure:* Geometry and velocity solution for back facing step case
pUCoupledFoam - Coupled incompressible solver II

Figure: Performance of simpleFoam compared to pUCoupledFoam.
Questions?