Finite Volume thermal-hydraulics and neutronics coupled calculations

Araújo Silva, Vitor V.; Campagnole dos Santos, A. A. and Mesquita, Amir Z.
Centro de Desenvolvimento da Tecnologia Nuclear - CDTN/CNEN
Av. Antônio Carlos 6627 – Campus Pampulha
Tel.: (+55) 31 3069 3466, Fax (+55) 31 3069 3411, vitors@cdtn.br

Bernal, Álvaro; Miró, Rafael and Verdú, Gumersindo
ISIRYM, Universitat Politècnica de València - UPV
Camino de Vera, s/n - 46022 Valencia
Tel.: (+34) 96 387 70 00 , Fax: (+34) 96 387 90 09

Pereira, Cláubia
Departamento de Engenharia Nuclear - DEN/UFMG
Av. Antônio Carlos 6627 – Escola de Engenharia
Tel.: (+55) 31 3409 6666 , Fax (+55) 31 3409 6660

Abstract – *The computer power processing available nowadays brings to a practical level computations that only few years ago would be unfeasible. In this context, this ongoing work aims to couple two different problems which, despite being naturally coupled, are usually treated separately. Using the finite volume technique, a 3D diffusion nodal code was implemented. This code can handle non-structured meshes, what allows for complicated geometries calculations and therefore more flexibility. In order to fully exploit this neutronics calculation flexibility, a computational fluid dynamics (CFD) code was used in order to obtain the same level of details for the thermal-hydraulics calculations. The chosen code is OpenFOAM, an open-source CFD tool. Changes in OpenFOAM allow simple coupled calculations of a PWR fuel rod with the neutronics code. OpenFOAM sends coolant density information and fuel temperature to the neutronics code which sends back power information. In this version meshes do not need to be the same, but only rod geometry. A mapping function is used to average values when one node in one side corresponds to many nodes in the other side. Data is exchanged between codes by library calls. As the results of a fuel rod calculations progresses, more complicated and processing demanding geometries will be simulated, aiming to a real scale PWR fuel assembly simulation.*

I. INTRODUCTION

Coupling neutronics and thermal-hydraulics is not something new. The coupled nature of these two physical phenomena – which in nature can be seen as one unique phenomenon – is well known by scientists and engineers. The computational power available nowadays allows perform these coupled calculations.

The present methodology forces at least one constraint to the separated codes in order to perform coupled calculations: both codes must use the same geometry. However, meshes can be different for each code as long as the internal surfaces are the same – this last constraint can have impact on the mesh definition since mesh elements must respect the internal surfaces.

The coupling approach used in this work can be classified as *internal* [1]. The boundary conditions are defined by each code individually as if each one would perform their calculations in stand-alone mode. The coupled execution works as the thermal-hydraulics code starting the process and controlling the execution of neutronics inside its main loop. Data is shared using a *mapping* function aimed to generalize the use of different neutronics and thermal-hydraulics meshes. The mapping function is crucial to data sharing and will be discussed in details in the next sections.

II. NEUTRONICS

The neutronics code used in this work is developed in the ISIRYM group in the Universitat Politècnica de València [2]. This code uses de *Finite Volume Method (FVM)* algorithm implemented in the Arb solver [3] to discretize the neutron diffusion equation. The matrices obtained after discretization make a generalized eigenvalue problem and are therefore solved using the SLEPc library [4].

The strength of this code is its capability of dealing with non-structured meshes, allowing complex geometry modeling and thus, the neutronics simulation of a wide set of fuel rods, assemblies or full reactor cores.

The code uses 2-energy group neutron diffusion approximation and the cross-sections are calculated previously considering volume and materials weight at each volume element.

The current version does not work for transient neutronic calculations. In order to extend code's functionality to transient calculations, terms for samarium and xenon production must be added to the system of equations. Also, the corresponding cross-section data of these two elements must be added and, eventually, interpolated to the operation temperatures.

III. THERMAL-HYDRAULICS

The thermal-hydraulics calculations are performed using the OpenFOAM software package [5]. OpenFOAM is much more than a code, but a full set of tools aimed to solve a wide range of engineering, mathematical and physical problems using FVM methods. OpenFOAM offers a set of different *solvers*, which can be seen as modules to solve one type of problem. The solver chosen to simulate the fluid dynamics and the heat transfer in this problem is called *chtMultiRegionSimpleFoam*. This solver is in fact a combination of two other solvers and is capable of solving the conjugate heat transfer problem among many solid and fluid regions in a stationary state. This solver considers the fluids being simulated as compressible fluids. Figure 1 shows a simplified version of the algorithm used to solve the coupled problem. It is worth noting that this is the modified version of OpenFOAM's *chtMultiRegionSimpleFoam* and includes the neutronics calculations. Changes made to the original algorithm will be addressed in the next section.

In order to be able to solve the problem generaly described, it is necessary to generate problem's geometry and mesh, define constraints, constants, physical properties and many other variables. This step is commonly called *pre-processing* [6]. The mesh used in this problem represents a single "square" fuel rod and is depicted in Figure 2. The reason to use this unconventional geometry is to simplify the calculations. In terms of OpenFOAM, this means not using any kind of non-orthogonal corrections. Moreover, it simplifies the validation of the *mapping*-function due to the simple shape of each individual mesh element. The four materials are fuel, gap, cladding and cooler. The physical properties of each material are described individually.

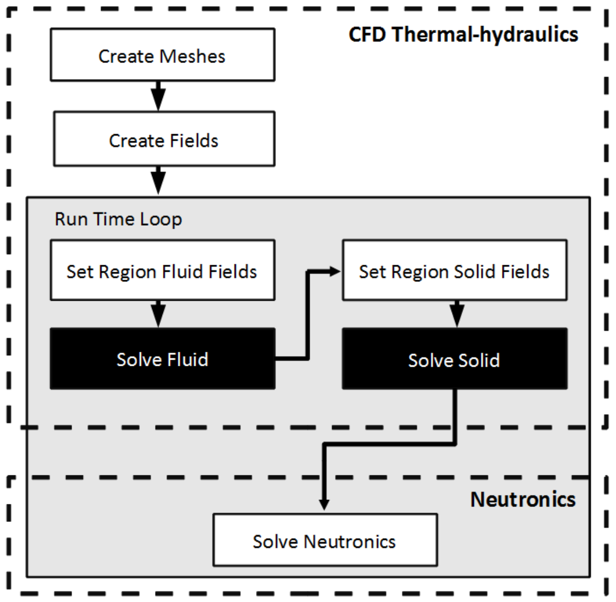

Figure 1: Coupled algorithm scheme

The main information for the CFD point of view, however, is the physical state of the material. Depending if it is fluid or solid, a different equation will be used to the calculations (shown in Figure 1). The fuel, cladding and gap are modelled as solids and cooler as fluid. Despite being a thin layer filled of helium gas, the gap is considered solid to simplify calculations since any convection in this layer brings no valuable contribution to the calculations.

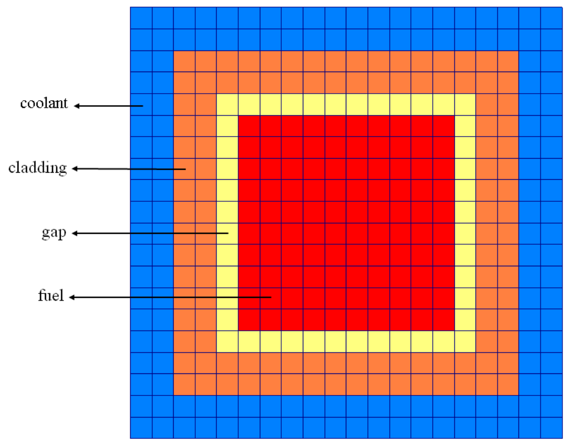

Figure 2: Fuel "square" rod mesh top-view – materials shown.

The mesh is formed of 36,000 hexahedra based on an extruded two dimensions mesh. This makes a structured mesh. It is worth noting that this is not a constraint and both CFD and neutronics codes are able to work with non-structured meshes.

The equations solved for the fluid regions are three: *momentum equation* for fluid velocity, energy equation for heat in the fluid – also using the turbulence model – and the equation for pressure. After solving all fluid regions – only one in the simulation presented – the solid regions loop starts. For this region, only a single equation is solved for *heat*. This equation was changed with the addition of a source-term (Q) in order to be able to get power information from neutronics code or from other external source.

IV. COUPLING

A set of changes had to be made to the original OpenFOAM solver to couple its CFD calculations to the neutronics code. The first one consisted of changing the original heat equation in the solid to use a *source-term*. OpenFOAM offers its own way of adding source terms and other features to its equation in running time. This new feature was added in version 2.2, the same used for the coupling. Unfortunately, the way it is implemented in OpenFOAM – by means of *functors* [7] – was not practical for coupling purposes. So, the new code was removed and a simple field representing power as Q was added to the equation (1).

$$- \alpha\nabla^2 u - Q = 0 \qquad (1)$$

In order to add the source-term to OpenFOAM's equation, a new volume field had to be created to get Q values. Since this is a matter of internal implementation, these changes are not shown for sake of clarity.

The neutronics module shown in Figure 1 was added to the CFD solver. It is the stand-alone code [2] adapted to run trough function calls and compiled as a library. Once compiled as a static or dynamic library, it is called from CFD C++ code. Since the neutronics library is written in Fortran, it is called from the CFD as a C function. This is done transparentily in Linux environments and fews adjustments in compilation and linking parameters allows the neutronics library to communicate to the CFD by means of a single function call presented below:

```
extern "C" void
voldiff_(int*,double*,double*,double*);
```

The code tells to C++ compiler to consider this function a C function and present four parameters: the first integer is a flag to tell the neutronics code that it is being called from CFD. The others represent a vector sending densities, temperatures and a reference to a vector where the power will be returned to the CFD.

However, the communication process is only part of the coupling scheme. More than only change data, the codes must agree on what the data represents. To clarify this point, the simpler coupling scheme would be both codes sharing the same mesh for all regions. With both codes seeing the same domain, the communication would be straight forward. Any volume node in neutronics would have its counterpart in the CFD side. This means all data being shared directly. However, in order to achieve a more general coupling scheme, a *mapping function* was implemented.

IV. A. Mapping-function

The *mapping-function* depicted in Figure 3 works averaging values from one mesh to another. This functions reads the node information provided by neutronics as a text-file when starting the *solver*. This file contains the Cartesian vertex of each node. Based on this information, the CFD creates a mapping relating its internal mesh volumes to the node.

In order to deal with the common situation in which there is no perfect matching of CFD mesh elements and node surfaces, the mapping algorithm considers the center of the mesh element. So, if the geometrical center of mesh element belongs to one node, it is considered in the node. This approach can be improved considering, for example, the volume of the mesh element inside de node and weighting its contribution for each node it is partially contained. However, this will cost much more in terms of memory and running time and this approach will be considered carefully in the future.

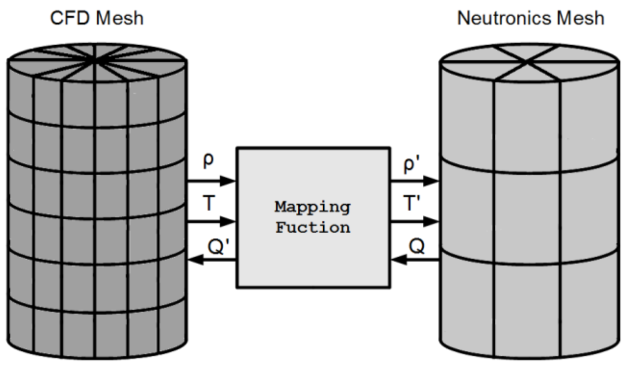

Figure 3: Data flow between CFD and neutronics.

The drawback of this solution for matching is the increasing error in averaging when the CFD mesh elements are big and the nodes are big and both have unmatching surfaces. Both considerations are valid for unstructured meshes, both in CFD side and neutronics side. Another issue with the mapping-function is that it can only represent structured regular meshes. A better mapping-function will be considered in future implementations.

The mapping function boundaries are defined by the `Nodes.Info` file, which is necessary for the coupled calculations. If the CFD solver does not find this file, it works as a non-coupled solver. The file has a simple format: -x x -y y -z z for each line. The number of lines read is the number of nodes. This simple format was defined to be read by both Fortran and C++ codes in the simplest way.

Once the mapping-function is defined, there are two fluxes of information. Fluid densities and solid temperatures calculated by the CFD are averaged and these averaged values are set to the respective node. The power data received from the neutronics represents the bigger node and this value is set to the CFD elements. This power is the source-term used to solve the heat equation for each node.

It is worth noting that this simple approach to mapping two different meshes is aimed and only applicable to structured meshes. In order to extend the mapping functionality to generic meshes, more complex and robust algorithms must be used. One possible approach is to extend surface matching/coupling algorithms – like the *Intersection method* [8] - to volume matching.

IV. A. Coupling control

The coupling scheme is controlled by the CFD. As depicted in Figure 1 (lower dashed box) the neutronics calculations are performed inside the *run time* loop of the CFD solver. This means after the CFD performs all its calculations using a pre-defined uniform power value for the first iteration, the neutronics is called with the first densities and temperatures values. The neutronics library performs its calculations using these values and the gives back to the CFD the power for each node. The neutronics convergence is up to the neutronics code which uses pre-defined iterations and residuals values. These values are set by text files used by the neutronics code in its setup. Before the power values are mapped and assigned to CFD mesh elements, they are normalized to the nominal power in the simulation

The first CFD iteration ends and the next one will start using power values calculated by the neutronics. The convergence of the simulation follows the parameters defined in the pre-processing phase of the CFD simulation.

There is no unique convergence criterion for the coupled calculations in this implementation. In this simple approach, inner power calculations are performed by the neutronics code inside the power loop of CFD code. Improvements linking the convergence criteria for both codes are envisaged but not yet implemented.

IV. SIMULATION

The “square” rod mesh used for this coupled simulation is 1cm side and 7.2cm long. This unusual size aims to have a “rod” shot enough to represent the physical phenomena but also provide a manageable geometry to results verification. The full “square” rod is depicted in Figure 4.

The simulation used a time-step of 0.001 and a total time of 100. Data was written to disk at every complete 100 iterations.

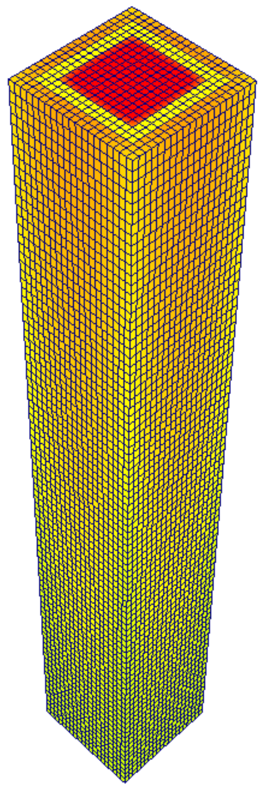

Figure 4: Mesh longitudinal view

The fuel rod is in vertical position with water flowing inside from the top to the bottom at 1 mm per second and the internal water field is also moving at the same velocity. This slow value for velocity was used in order to test simulation's convergence in a simple situation. The water enters the domain at a temperature of 300K (27°C). This is not the real temperature of the water influx in a PWR reactor, but the rationale behind this decision was to verify the water heating, its density variation and provide a strong cooling domain to impact the system heating trough the power provided by the neutronics.

Initial temperatures in each material are presented in Table I.

The turbulence model used was kappa-epsilon with κ 0.001 and ε 0.03. OpenFOAM uses a different thermo physical file for every defined material. The properties of each solid were defined by polynomials representing the variation of each properties for the temperatures in a PWR reactor.

TABLE I

Materials initial temperatures

Material	Initial temperature (K)	Initial temperature (°C)
Fuel	900	627
Gap	600	327
cladding	600	327
cooler	562	289

The domain external walls were considered symmetric in OpenFOAM's terminology, which means there is the same surface at the other side of the patch.

IV .A. Results

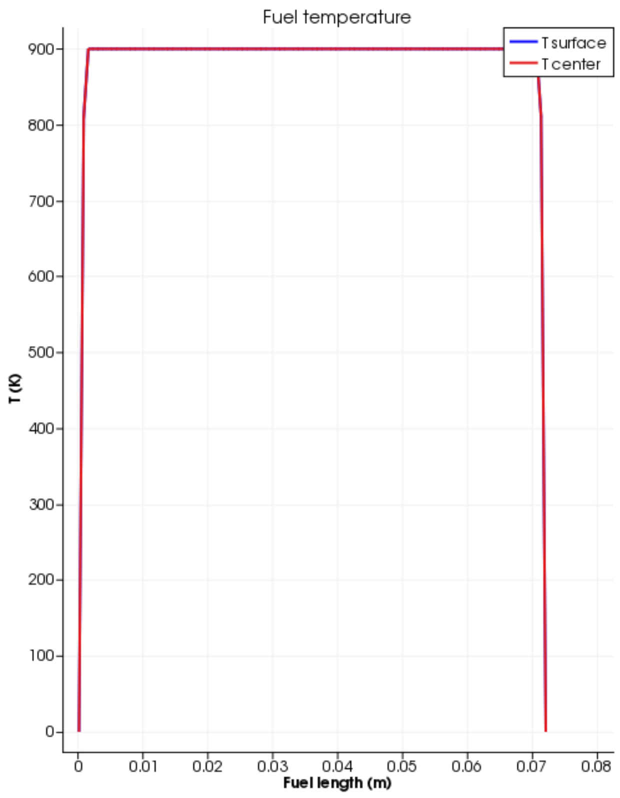

Figure 5: Initial fuel temperature

Simulation time was roughly 701,385 secs what is almost 8 days and 3 hours. The reason for such long time using such a simple "square" rod is that the solver version used was implemented to run only in parallel only in stand-alone mode. Therefore, the case presented was run using a single core in a multi-core processor. Neutronics calculations are performed with a fixed number of iterations of five and its overhead is negligible compared to the running time of the CFD calculations and agrees with the results in the literature [8].

Neutronics results had no major variation as expected, since the simulation was running in steady-state mode. However, due to a small variation in water densities due to the thermo physical mode available in OpenFOAM, the neutronics variation was negligible. The averaged five first eigenvectors calculated by the neutronics code during the coupled simulation are presented in Table II.

TABLE II

Averaged first five eigenvalues pairs from neutronics.

k	||Ax-kx||/||kx||
0.563750	2.43539e-09
0.499966	2.26323e-09
0.416348	1.45105e-09
0.414043	1.28779e-09
0.379014	2.44053e-10

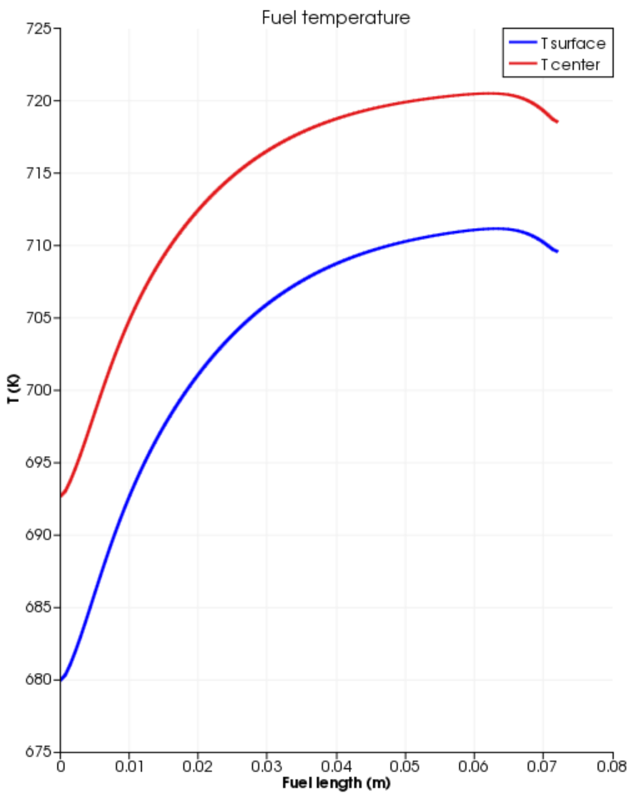

Figure 6: Fuel temperatures after one iteration

The values presented in Table II show that the simulated system is sub-critical. This is not an issue from the computational perspective, since the sub-critical system will yield values for power which will not increase fuel temperatures. The CFD simulation will calculate the heat transfer anyway and the initial system will lose heat due to the influx of colder water.

Temperatures in the center of the fuel and at its surface at different times are presented in Figures 5, 6 and 7 respectively. In Figure 5 the graphics of temperatures in the center and at surface overlaps.

As time passes, the temperature in the top of the fuel gets colder than in the bottom since the power generated by the neutronics is not enough to avoid the cooling of the fuel. Despite using a steady-state algorithm, the cooling phenomena is well represented in time.

The expected system behavior can be confirmed checking the averaged final temperatures for each material are presented in Table III.

TABLE III

Materials averaged final temperatures

Material	Averaged final temperature (K)	Averaged final temperature (°C)
Fuel	550.55	277.55
Gap	549.02	276.02
cladding	547.84	274.84
cooler	547.65	274.65

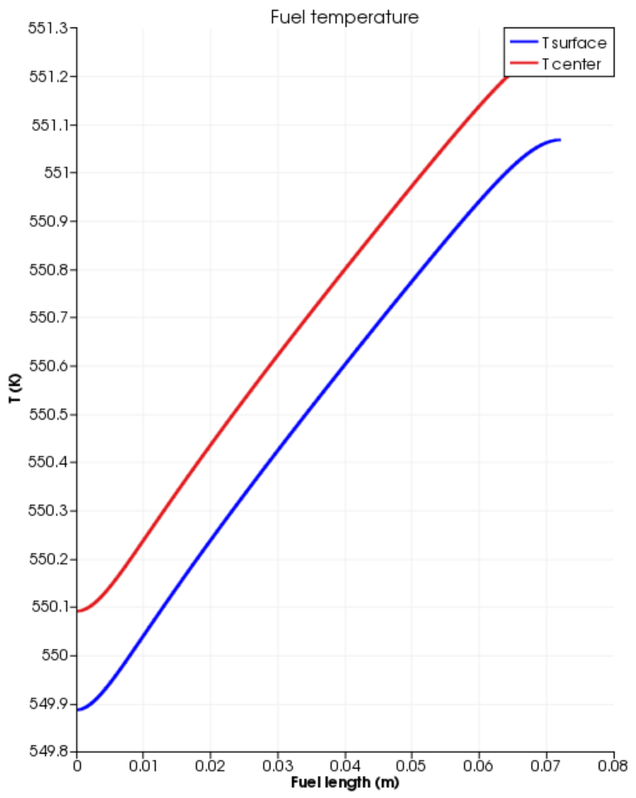

Figure 7: Fuel final temperatures

A graphical visualization of the fuel after the cooling process is shown in Figure 8.

Temperatures trough all materials are shown in Figure 9. At the end of the simulation there are still differences among all material layers. It is possible to check the intersection between the neighbor materials.

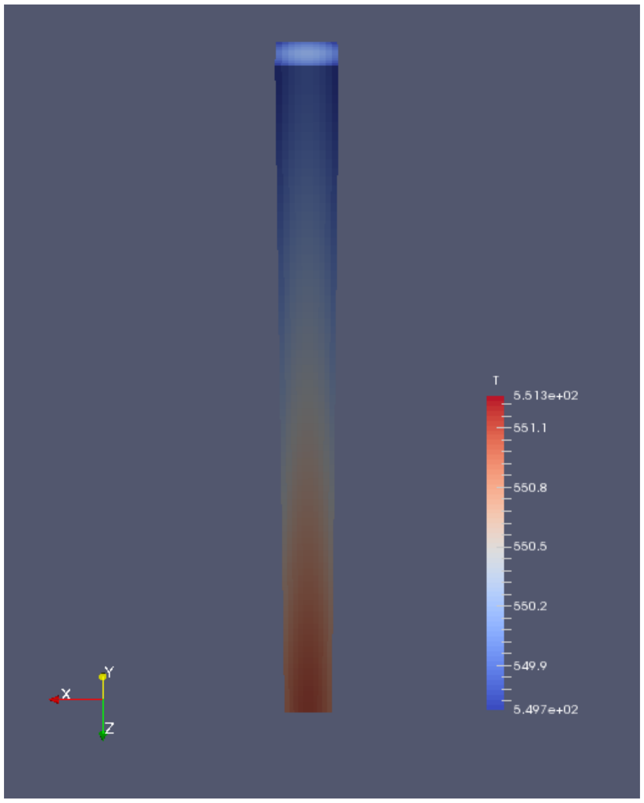

Figure 8: Fuel surface temperature

The results shown a well expected behavior of heat transfer in the different materials.

V. CONCLUSIONS

This work can be seen from two very different perspectives. The first one is to analyze it as a software project. In this case, it can be considered successful, although there is room to improvements. The two main improvements are finishing the parallel implementation. Based on the simulation time, it is clear that the time spent to solve the problem of a “square” rod is not practical if run sequentially. The solver is already implemented in parallel for stand-alone cases and the parallel implementation for the coupled case is straightforward. Simulations in parallel of the stand-alone solver shown a speedup of 16 times the sequential running time. A similar speedup is expected for the coupled case. The second main improvement is to change the *mapping-function* to better deal with differences in geometries. One possible approach is to avoid using C++ STL functions and change OpenFOAM’s data structures to store the node information used by the *mapping-function.* Changes in the convergence criteria for the neutronics calculations can also lead to a better performance of the coupled calculations.

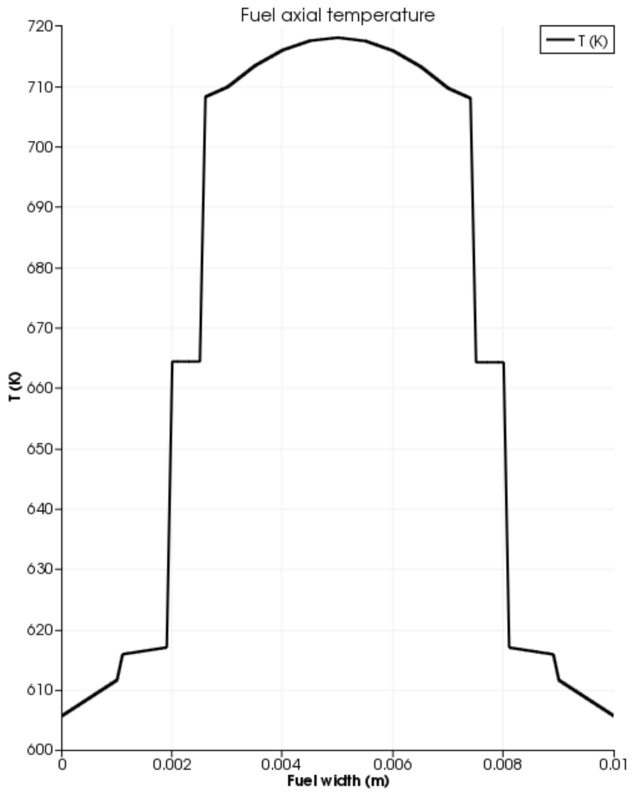

Figure 9: Axial cut of fuel at its center

Another perspective is from the engineering point of view. In this case, the work needs a lot more improvements. Specially if considering the results of a non-critical “square” fuel rod presented. However, there is a lot of information provided by the presented simulation to be used in the next ones. First of all is to decrease the nodes size. With smaller nodes, the averaging process will impact less in the neutronics and there will be less non fissionable material in nodes containing fuel. This will give a better power volume to be passed to the CFD and, correspondently, better spatial precision to data feeding the neutronics calculations. Another important question arose from this simulation is the modeling of a gap. Its contribution to the calculations is null, since it is only one element tick. In more realistic geometry, generate a mesh for a full length rod can be a daunting task with a gap due to its very small dimension compared to a four meter rod. One possible approach is to replace the gap for a contact resistance.

Concluding, the results in this work show that the coupling scheme works fine and data is shared between codes without issues. The coupling implementation is internal by function calls avoiding the expense of I/O operations to write files on disk.

VI. FUTURE WORK

As presented before, this is a work in progress. There is still a lot to improve before having a practical system to PWR simulation. Some points to be addressed in the future in order of importance:

- Parallel implementation for coupled calculations (in progress, but not ready at the time of this paper preparation);
- Simulation with a mesh representative of a real cylindrical rod and then the simulation of a 5x5 fuel rod bundle;
- Investigation of thermo physical properties for fluids in OpenFOAM, specially in relation to density variation;
- Investigation of fluid velocities near the heating wall.

ACKNOWLEDGMENTS

The author thanks CAPES for the grant process number 11.312/13-0 for conducting part of this research in the ISIRYM/UPV in the terms of the PDSE program.

NOMENCLATURE

ISIRYM: Instituto de Seguridad Industrial, Radiofísica y Medioambiental.
CFD: Computational Fluid Dynamics.
PWR: Pressurized Water Reactor.
FVM: Finite Volume Method.
STL: Standard Template Library.
I/O: Input/Output.

REFERENCES

1. K. IVANOV and M. AVRAMOVA, “Challenges in Coupled Thermal–hydraulics and Neutronics Simulations for LWR Safety Analysis”, *Annals of Nuclear Energy, 34*, 501-513 (2007).

2. A. BERNAL, R. MIRÓ, D. GINESTAR and G. VERDÚ, “Resolution of the Generalized Eigenvalue Problem in the Neutron Diffusion Equation Discretized by the Finite Volume Method”, *Abstract and Applied Analisys*, **2014,** 15 pages (2014). Article ID 913043. http://dx.doi.org/10.1155/2014/913043.

3. F. JOHANSSON, “Arb: a C library for ball arithmetic”, *ACM Communications in Computer Algebra*, **4,** *47*,166-169 (2013).

4. V. HERNANDEZ, J. E. ROMAN, and V. VIDAL, “SLEPc: A Scalable and Flexible Toolkit for the Solution of Eigenvalue Problems*”. ACM Transactions on Mathematical Software*, **3**, *31*, 351-362 (2005).

5. OPENFOAM, *The Open Source CFD Toolbox: User guide*, OpenFOAM Foundation. Version 2.2.0 (2013).

6. H. K. VERSTEEG and W. MALALASEKERA, *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, Pearson Education Limited (2007).

7. A. ALAIN, *Jumping into C++*, Cprogramming.com publishing (2013).

8. A. BOER, A. VAN ZUIJLEN and H. BIJL, “Review of Coupling Methods for Non-Matching Meshes”, *Computer Methods in Applied Mechanics and Engineering*, *196*, 1515-1525 (2007).

9. J. YAN, B. KOCHUNAS, M. HURSIN, T. DOWNAR, Z. KAROUTAS and E. BAGLIETTO, “Coupled Computational Fluid Dynamics and MOC Neutronic Simulations of Westinghouse PWR Fuel Assemblies with Grid Spacer”, *Proceedings of NURETH-14*, Toronto, Canada, September 25-30 (2011).