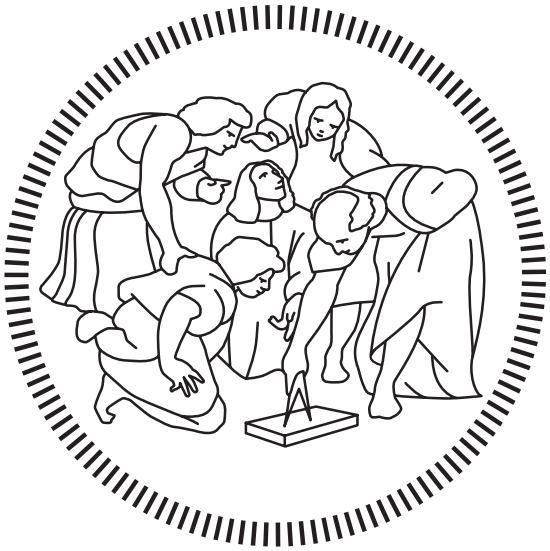
**Politecnico di Milano**

SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING
Master of Science – Energy Engineering



# Porous Media Approach in CFD Thermohydraulic Simulation of Nuclear Generation-iv Lead-cooled Fast Reactor ALFRED

Supervisor
**Marco Enrico Ricotti**

Co-Supervisor
**Stefano Lorenzi**

Candidate
**Martín Gómez – 935881**

**Academic Year 2020 – 2021**

# Acknowledgements

The only proper way I can find to start this thesis is by thanking everyone who, despite the rough times, has made these last two years better.

First of all, I should thank my supervisor, prof. Marco E. Ricotti, for letting me be part of such an interesting project. A very special thank you also goes to my co-supervisor, prof. Stefano Lorenzi, who has been my constant and unconditional academic support in this thesis, helping me whenever I was lost (more times than I am willing to recognize). Thank you also to the Politecnico di Milano and all the people composing it, you are part of an incredible institution which has giving its best even in the darkest moments. It is only natural to mention also the Universidad Politécnica de Madrid, responsible of my previous education as engineer and which gave me the chance of studying a double degree program in a foreign university. I will never be able to properly express how much I have learned thanks to you all.

Devo ringraziare anche a la città di Milano, dove ho trascorso una delle stagioni più felici della mia vita quasi senza rendermene conto. E alla sua gente, che mi ha accolto così calorosamente dal primo momento. Sei stato un esempio di resistenza e, ne sono certo, alla fine *tutto andrà bene*.

Tampoco me puedo olvidar de aquellos quienes, a pesar de la distancia y mis involuntarios ataques asociales, han sabido estar siempre ahí, esperando. A mi padre, a mi mere, al resto de mi familia en todas sus formas, a mis amigos y conocidos, a todos: perdón y gracias. Vosotros me habéis recordado que la vida es más que meros hitos académicos.

Thank you. Grazie. Gracias.

# Sommario

Da diversi anni, molti studi hanno evidenziato i vantaggi dell'utilizzo delle energie rinnovabili in sinergia all'energia nucleare, proponendo una strategia a lungo termine per ridurre le emissioni di $CO_2$ che coinvolga un mix di fonte nucleare e rinnovabile. Proprio per l'interesse per le sue basse emissioni di carbonio, la produzione di energia nucleare è stata in continuo sviluppo sin dal suo inizio nei primi anni '50, avanzando verso progetti sempre più sicuri ed efficienti. Attualmente, la cosiddetta Generazione IV di reattori nucleari è in fase di sviluppo nel tentativo di soddisfare obiettivi che coprono quattro grandi aree: sostenibilità, economicità, sicurezza e affidabilità, resistenza alla proliferazione e protezione fisica. In base a questi obiettivi, sono stati valutati diversi concetti, selezionando infine sei tecnologie nucleari, incluso il reattore veloce raffreddato a piombo (LFR). Nell'ambito dei finanziamenti europei, il reattore ALFRED è stato proposto come dimostratore di questa tecnologia.

Questo lavoro di tesi mira alla simulazione CFD del reattore ALFRED, sia in condizioni operative normali sia durante possibili ostruzioni di un elemento di combustibile. A tal fine, poiché una simulazione geometrica dettagliata per l'intero core richiederebbe un calcolo molto impegnativo, in questa tesi viene proposto un approccio di mezzo poroso, utilizzando il software open source OpenFOAM. La griglia di calcolo è specificamente progettata per generare i diversi elementi del reattore, dividendo ciascuno di essi nelle varie regioni assiali. Un coefficiente di porosità viene quindi assegnato a ciascuna di queste regioni in base alla perdita di carico prevista, i cui valori sono calcolati dalle correlazioni e da precedenti valutazioni effettuate dai progettisti del nocciolo. Un trattamento esplicito della porosità viene adottato per la facilità di implementazione in qualsiasi risolutore. Inoltre, la generazione di potenza all'interno del nocciolo è simulata da sorgenti di calore volumetriche.

Come risultato di questo lavoro di tesi, tre casi principali sono simulati e analizzati: uno stato stazionario adiabatico, uno stato stazionario con generazione di potenza e un caso con ostruzione di un elemento di combustibile. I risultati vengono criticamente analizzati e alcune osservazioni conclusive vengono tratte. Tra queste, si possono evidenziare i promettenti risultati ottenuti simulando le perdite di carico all'interno del nocciolo, in particolare durante le simulazioni stazionarie del funzionamento operazionale. D'altra parte, si riscontrano alcune incongruenze nel campo di velocità rispetto ai risultati attesi, specialmente nelle simulazioni transitorie per casi di ostruzione, che derivano da profili di temperatura imprecisi. Questo problema sembra indicare l'incapacità dell'approccio esplicito di mezzo poroso di calcolare correttamente i campi di velocità impostando solo le perdite di carico nel dominio. Su questo aspetto, alcune possibili soluzioni e ulteriori linee di ricerca vengono delineate e raccomandate.

# Abstract

From several years ago, different studies have pointed out the benefits of exploiting renewable energies together with nuclear, concluding that a good long-term strategy for reducing $CO_2$ emissions could be considering a mix of nuclear and renewable energies. Thus, due to the interest on its low carbon emissions, nuclear energy production has been developing since its inception in the early 50s, advancing towards safer and more efficient designs. Now, the so-called Generation-IV of nuclear reactors is being developed in an attempt to accomplish certain goals covering four broad areas: sustainability, economics, safety and reliability, as well as proliferation resistance and physical protection. Considering these goals, several design concepts were evaluated and six nuclear technologies were finally selected together with the Lead-cooled Fast Reactor (LFR) concept. Inside the European framework, the ALFRED reactor has been proposed as a demonstrator of this kind of technology.

This thesis work focuses on the CFD simulation of this ALFRED nuclear core, both during normal operating conditions and during possible flow blockage, i.e. when possible obstructions occur. For this purpose, as a complete scope geometry simulation for the whole core would be very computing demanding, a porous media approach using the open source software OpenFOAM is proposed. Therefore, the core mesh is specifically designed to generate the different core assemblies and to divide each of them into the different axial regions composing them. Then, a porosity coefficient is assigned to each of those regions according to the pressure drop expected, whose values are calculated by correlations and previous assessments performed by the core designers. An explicit treatment of porosity is chosen in order to easily implement it when using any solver. Additionally, power generation inside the core is simulated by volumetric heat sources.

As a result of this thesis work, three main cases have been included: an adiabatic steady-state, a steady-state with heat sources calculated from the fission power, and a case with obstructions. An extensive analysis on these results has been performed and some concluding remarks have been reached. Among them, it can be highlighted the promising results obtained when simulating the pressure drops inside the core, specially during normal operation steady-state simulations. On the other hand, some inconsistencies have been found in the velocity field with respect to the expected results, specially in transient simulations for obstruction cases, which derives in inaccurate temperature profiles. This issue seems to indicate the inability of explicit porous media approach to correctly induce velocity fields only by setting pressure drops in the domain. In that line of reasoning, some possible solutions and further lines of research are also suggested.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

This first chapter will introduce some of the main concepts used in this thesis work as well as some background. In the first section, tje thesis work will be introduced through its scope and objectives. Then, a brief reflection on nuclear generation evolution will be made, presenting the different technologies included in Generation-IV reactors delving into features of LFR and the European project ALFRED.

## 1.1 Objectives and scope of the thesis

For some years now, Computational Fluid Dynamics (CFD) has been playing a key-role in the development and feasibility of new technologies. Its advantages against experimental approaches are numerous: simplicity of application, cheaper, flexibility of options, delocation... Thus, CFD analysis has been widely implemented in most of engineering branches in which fluid dynamics is involved, and the thermo-hydraulic study of a nuclear core is not an exception.

Additionally, and as will be further explained later, nuclear generation plays an important role in nowadays energy and electricity mix, and offers some advantages when facing $CO_2$ emissions and climate change. Therefore, the development of nuclear energy technologies is currently an issue that seems to be further developed in the near future.

Following these ideas, the aim of this thesis is to study the behaviour of the Generation-IV Lead-cooled Fast Reactor ALFRED by CFD analysis, simulating both operating conditions and flow blockage scenarios. For this purpose, as a complete scope geometry simulation for the whole core would be very computing demanding, a porous media approach using the open source software OpenFOAM is proposed.

In order to organize the work and facilitate the accomplishment of such a general and ambitious aim, a division in more specific objectives can be done:

- Generate a suitable mesh geometry so that the most important information can be extracted from the different analyses.
- Study the implementation of a porous media approach to this particular kind of nuclear reactor.
- Study the performance of the approach under normal operation conditions.

- Study the performance of the approach when obstructions are simulated and the possible detection of hotspots in those cases.

These points will be helpful when articulating the final conclusions, which should answer about the thesis successfulness.

While achieving these objectives, some basic principles will be present all throughout this thesis:

- Flexibility. All the elements introduced in the following chapters (mesh, different scripts...) are intended to be as flexible as possible so that different parameter values can be easily changed in order to reproduce other conditions.
- Repeatability. All the necessary information to repeat the simulations is included in this document so that a process of external verification could be carried out.
- Simplicity. Once the previous principles are satisfied, the general decision criteria throughout this thesis is to keep things as simple as possible. This is applied when choosing physical models for example.

## 1.2 GEN-IV reactors: the role of Lead-cooled Fast Reactors and ALFRED

One of the biggest environmental challenges nowadays is the reduction of carbon dioxide and pollutant emissions, which are mainly related to energy generation, industrial production and transportation. Multiple studies certify that there is not an unique path to achieve this reduction goal, but the only way to definitively reduce those emissions is by using all the available technologies in a smart and cooperative way. It is precisely in this framework where nuclear generation can be useful, as it provides a near zero-$CO_2$ impact energy (a whole Life Cycle Analysis Life Cycle Analysis (LCA)

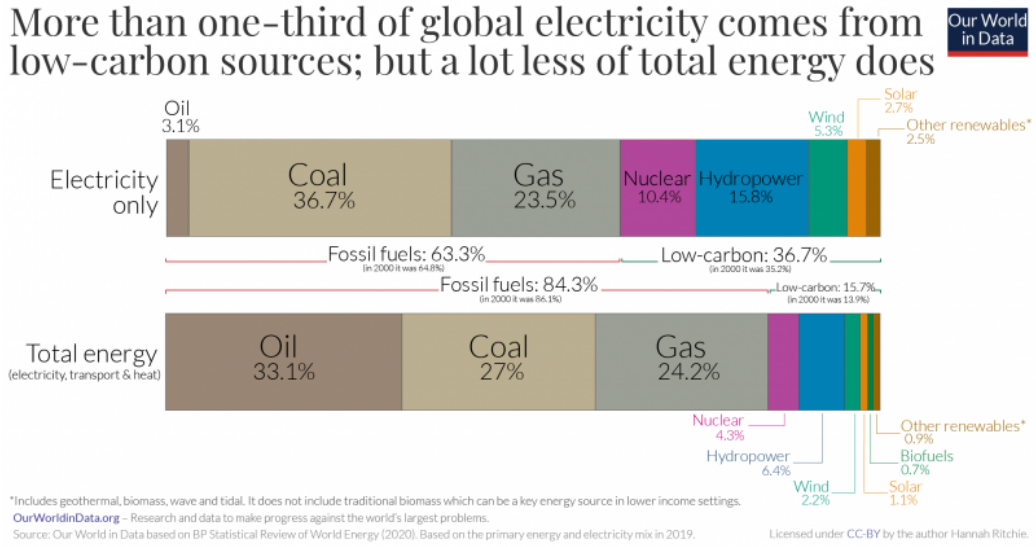



Figure 1.1. World electricity mix and total primary energy mix. Source: Our World in Data.[31]

would show some emissions during construction, fuel transport, dismantling...). This feature together with the almost constant power output under any ambient condition (concept of base power) and improved security measures have turned nuclear generation into one of the oldest low-carbon energy technologies, having nowadays a great impact in the World electricity mix.[31] This can be observed in Figure 1.1.

Different studies state the benefits of exploiting renewable energies together with nuclear, concluding that the best option to reduce $CO_2$ emissions in the long-term seems to be considering a mix of nuclear and renewable energy at least in some OECD countries.[42,45] However, the slower growth in carbon emissions in 2019 from the sharp increase seen in the previous year is not due to an increase in nuclear generation, but to the fact that primary energy consumption decelerated and renewables and natural gas displaced coal from the energy mix. In fact, serious safety concerns after the Fukushima accident in 2011 and the production of highly radioactive materials that must be correctly treated have slowed down nuclear global generation growth for the last years. But, in 2019, nuclear generation has experienced its fastest growth since 2004 and well above the 10-year average, mainly motivated by the construction of new reactors in China and the recover of Japan after the impact of Fukushima accident.[4] This can be observed in Figure 1.2.

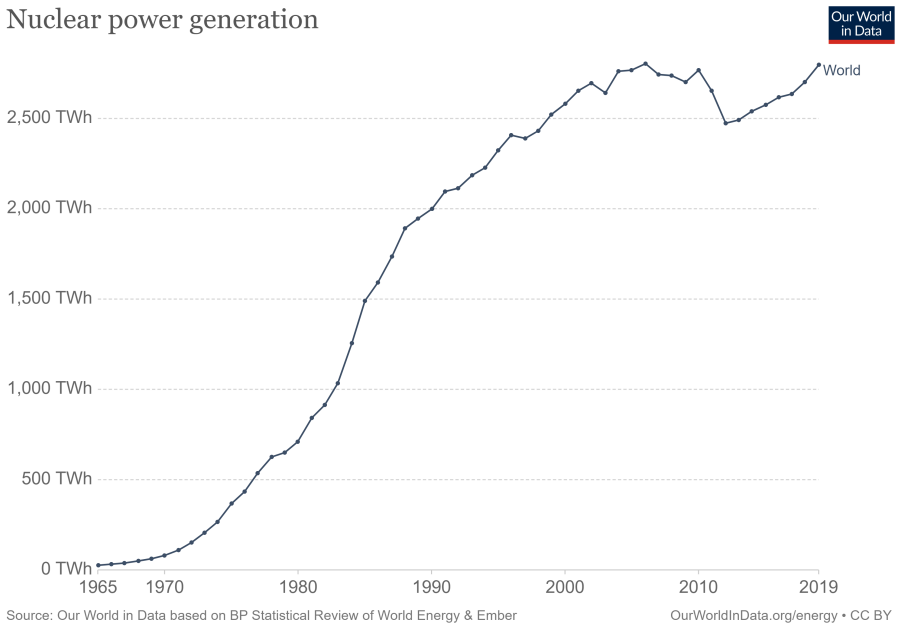



Figure 1.2. Worldwide nuclear power generation. Source: Our World in Data.[32]

Precisely in an attempt of solving these concerns (safety and production of highly radioactive materials), there has been a continuous development in nuclear civil energy production since its inception in the 1950s. Its history has been commonly subdivided in so called generations representing a group of technologies and design, construction, economical and managerial philosophies which have characterized each historical period. It is observed from Figure 1.3 that past and near future nuclear reactors are commonly divided in four generations. The first of them, developed between the 50s and 60s corresponds to the early prototype reactors. They were followed by the second generation in the early 70s, from which main part of nowadays

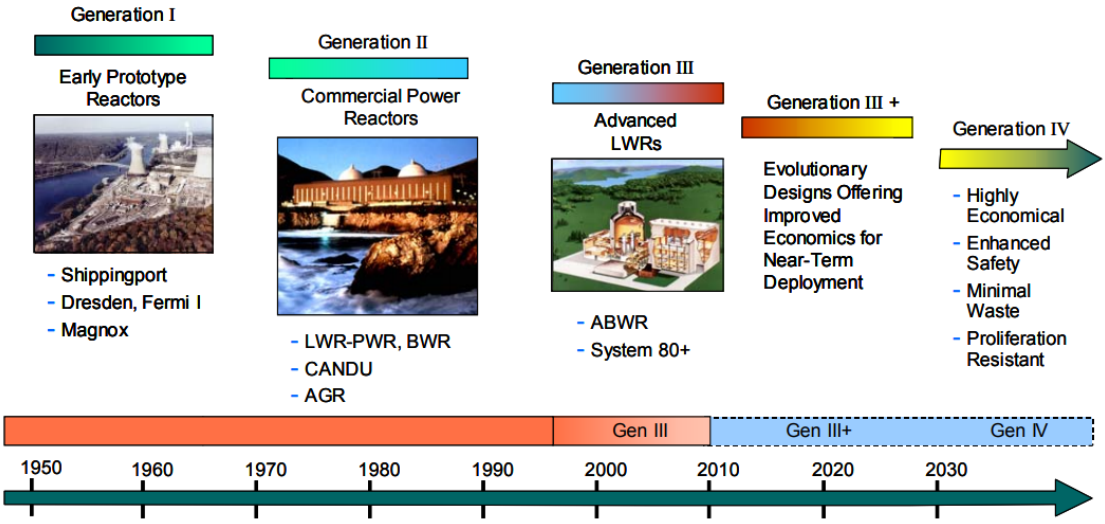



Figure 1.3. Evolution of nuclear technologies divided in different generations. Source: GIF.[13]

commercial power reactors in operation belong. Designs development impacting upon safety and economics derived in Generation III during the 90s, which has been under study for further advances resulting in the so called Generation III+. Beyond 2030, some goals have been established for nuclear reactors covering four broad areas: sustainability, economics, safety and reliability, as well as proliferation resistance and physical protection. Actually, Generation IV is being developed with the objective of achieving this goals with reactors available by the year 2030, when many of the world's currently operating nuclear power plants will be at or near the end of their operating licenses.[48]

According to Generation IV goals, an evaluation on several reactor concepts derived in GIF selecting six nuclear reactor technologies for further research and development. They include thermal and fast neutron spectrum cores, closed and open cycles, and sizes ranging from very small to very large. These six selected technologies are:

- Gas-cooled Fast Reactor (GFR)
- Lead-cooled Fast Reactor (LFR)
- Molten Salt Reactor (MSR)
- Supercritical Water Reactor (SCWR)
- Sodium-cooled Fast Reactor (SFR)
- Very High Temperature Reactor (VHTR)

Particularly, LFR is considered as one of the most promising technologies to meet the requirements introduced for Generation IV nuclear plants and it is being studied worldwide.[25] This kind of reactor operates in fast-neutron spectrum and uses a closed fuel cycle for efficient conversion of fertile uranium, although actinides from spent Light Water Reactor (LWR) fuel can be also consumed (working as a burner). Hence, excellent material management capabilities are shown. Additionally, the choice of

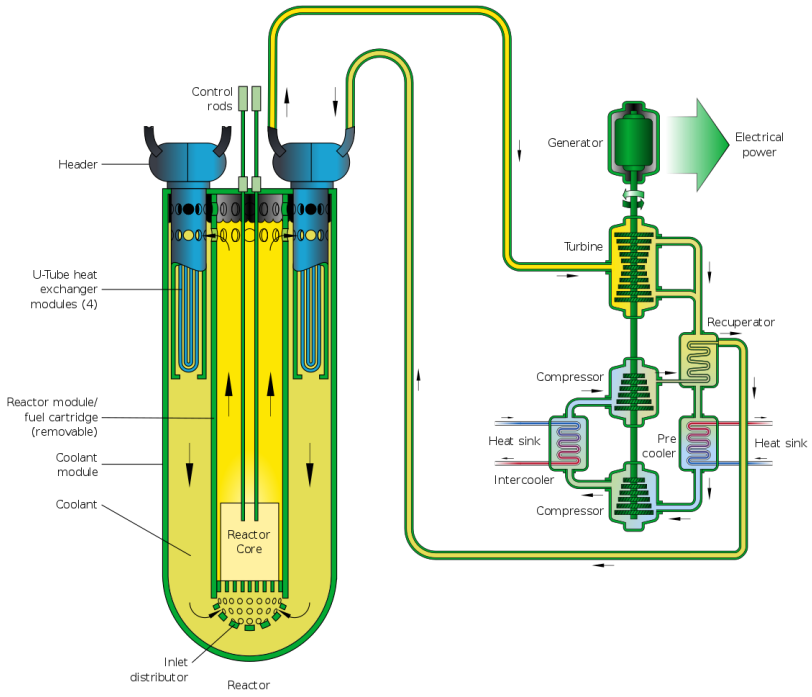



Figure 1.4. General schematic view of a typical LFR configuration. Source: GIF.[13]

molten lead as coolant also results in enhanced safety, more flexible operation and design, proliferation resistance and improved economic performance as will be further explained later. A general schematic view of a typical LFR configuration is shown in Figure 1.4.

As anticipated, this kind of reactor is cooled by molten lead (or lead-based alloys), whose peculiarities among other available coolants introduce some advantages. Lead very high boiling point (around 1743ºC) and low vapour pressure allow to operate the reactor at high temperature and near atmospheric pressure. Thus, the problem of coolant boiling and risk of a Loss Of Coolant Accident (LOCA) can be both virtually eliminated. This constitutes a great safety and operational advantage.[13] In general, good thermophysical properties from lead allow to high pitch/diameter core designs with low pressure drops and, hence, lower auxiliary power for pumping coolant. Passive safety is also possible by removing the decay power in natural circulation regime, which constitutes a redundant system when operated together with active safety systems. Furthermore, molten lead high density avoids the risk of fuel compaction reaching critical conditions in case of core melting and reduces the risk of steam inlet inside the core in case of breakage of the steam generator tubes. On the other hand, fast-neutron spectrum is favoured by the scattering properties of lead which allow the sustainment of high neutron energy and relatively low parasitic absorption of neutrons.[36] Additionally, the high shielding capability against gamma radiation offers a great protection for surrounding people, receiving very low doses.

Mixed Oxide Fuel (MOX) used in this kind of reactor can contain actinides from spent LWR fuel, making these systems unattractive for the extraction of weapon-usable materials. Nuclear properties of the coolant allow the realization of long life cores which are not useful for the production of weapon-grade plutonium. Lead also

constitutes a physical protection to the public and environment as it is characterized by its relative chemical inertness (compared to other coolants like water or sodium). This reduces the need for strong protection against rapid chemical interactions that can lead to energy release in the event of accident conditions derived from natural causes or acts of sabotage. Another important advantage is the tendency of lead to retain fission products in the case of a catastrophic event. In addition, The absence of inflammable substances reduces the risk of fire propagation.

In terms of sustainability, lead is an abundant material and hence available, even in case of deployment of a large number of reactors. Information about lead world reserves can be found in USGS Mineral Commodity Summaries (2021).[49]

To sum up what it has been stated so far, all the characteristics exposed before enable an improved utilization of resources with important features in achieving sustainability, longer core life, effective burning of minor actinides impacting resistance to the proliferation, enhanced passive safety and core reliability, as well as fuel cycle economics.

But, of course, there are also some research challenges related to the development of this technology. For example, the high melting point of lead (327 $^{o}$C) requires to maintain primary cooling loop at temperatures to prevent solidification that may cause obstructions in the flow. For this reason, understanding how the core temperature and coolant flow are affected in the presence of obstructions is a key aspect and will be investigated in this thesis. Furthermore, lead opacity hinders inspection and monitoring activities. But maybe one of the biggest challenges is lead tendency to be corrosive specially at high temperatures and in contact with oxygen and structural steels.[1]

In order to tackle these research challenges, different investigation and development initiatives in many countries have been proposed including Russia, USA, South Korea, Japan, China and Sweden. Particularly, in Europe, activities related to LFR technology development are carried out by two main systems:

1. Multi-purpose hYbrid Reaserch Reactor for High-tech Applications (MYRRHA) which is an Acceleration Driven System (ADS) demonstrator using Lead Bismuth Eutectic mixture (LBE) as coolant and a neutron source of spallation activated by a proton beam. It is built by SCK-CEN in Belgium. Further information about this project can be found explained by De Bruyn *et al.* (2015).[10]

2. ALFRED which is a 300 MWth demonstrator LFR that is under consideration for construction in Romania. This reactor is the one studied in this thesis and hence further information about it will be given in the following sections and chapters.

The final objective is the study of an industrial-sized reactor under the name European Lead Fast Reactor (ELFR). These research activities are driven by the LEADER project, funded by the European Commission (EC/Euratom). Major technological issues concern the following main topics:[25]

- Material studies and physical-chemistry coolant characterization
- Irradiation studies

- Thermal-hydraulic properties
- Instrumentation

This thesis, as already mentioned, will verse about the thermohydraulic performance of the ALFRED core under normal operating conditions and under the possible presence of obstructions in the flow.

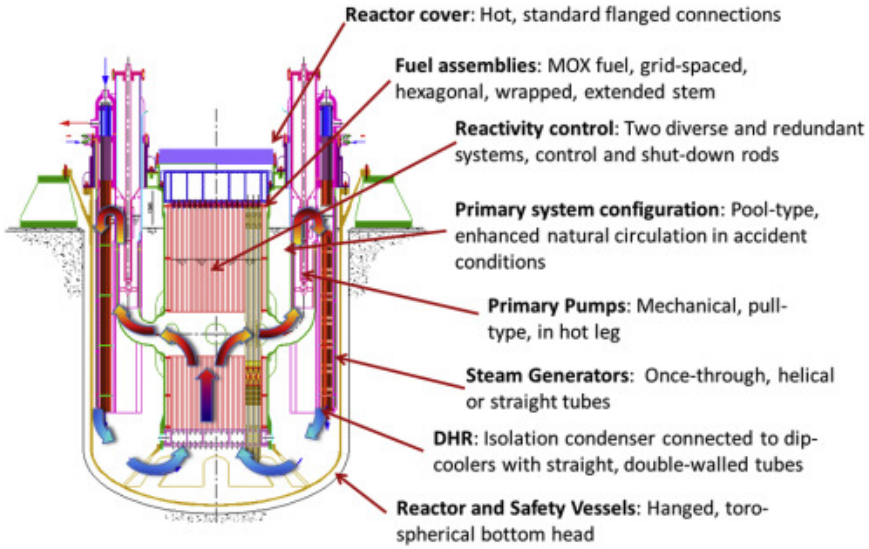



Figure 1.5. ALFRED primary system configuration. Source: Lorusso *et al.* (2018).[25]

### 1.2.1 ALFRED overview

ALFRED is a demonstrator of the lead fast reactor technology, with a foreseen thermal power of 300 MWth. It is conceived as a pool-type LFR whose primary system current configuration is depicted in Figure 1.5. The nuclear core, primary pumps and steam generators are all contained inside the main reactor vessel, which is located in a large pool within the reactor tank. ALFRED core organisation will be explained later in detail, but it can be mentioned that it is formed by wrapped hexagonal elements organised in a matrix. There are different kind of assemblies contained in this configuration: Fuel Assemblies FAs, Dummy Elements DEs, Control Rods CRs and Safety Rods SRs. Their characteristics will be further explained in future sections.

ALFRED is provided with eight steam generators placed inside the pool around the nuclear core as shown in Figure 1.6. Hot lead is conducted through a large pipe to each of the vertical pumps which circulates lead through the tube bundles forming the steam generators. Those bundles are composed by vertical bayonet tubes with an external safety tube and an internal insulating layer, delimited by a slave tube. This configuration is aimed at ensuring superheated dry steam production and can be observed in Figure 1.7, where red and blue arrows represent lead and water/steam flow respectively. As indicated, the gap at the most external wall is filled with pressurized helium and high thermal conductivity particles to enhance heat exchange between hot lead and cooling water/steam.

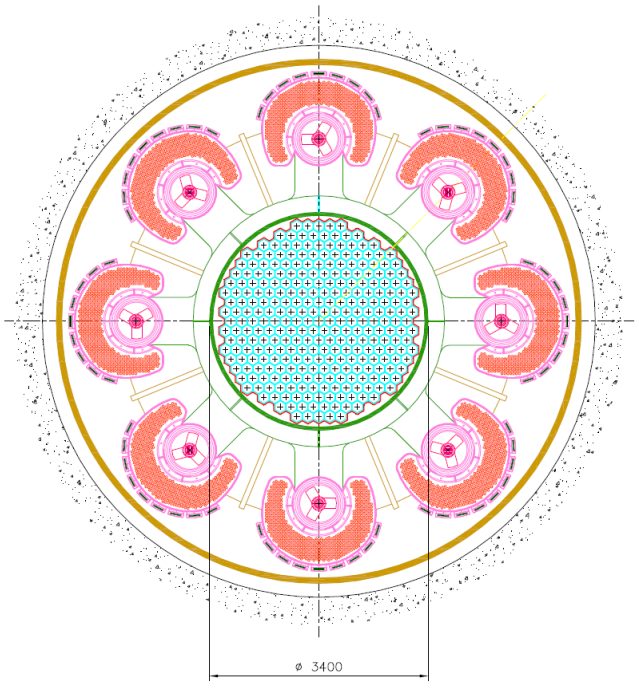



Figure 1.6. ALFRED reactor block configuration in which the eight steam generators can be observed. Source: Internal LEADER documentation.

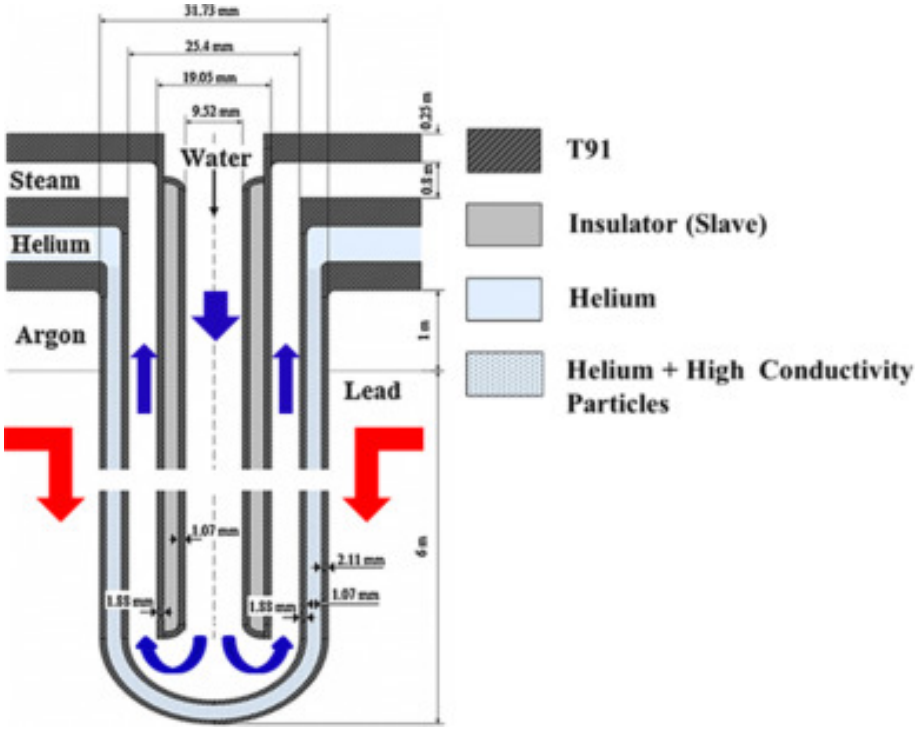



Figure 1.7. ALFRED bayonet tube configuration. Source: Ponciroli *et al.* (2015).[37]

# Chapter 2

# Methodology

The aim of this chapter is to give a detailed review of the methodology used in this thesis work. In order to do so, a first section about general CFD theory and some hints of OpenFOAM will be included, as well as a brief explanation about the motivation in choosing this particular program. In the two following sections, all the characteristics from ALFRED reactor that might be useful in future simulations will be defined and used in the generation of a suitable geometry able to capture all the requested physical processes. In the following two sections, the theory and implementation of the two main physical processes of interest will be introduced, which are: porous media approach for pressure drop simulation and power generation inside the core.

## 2.1 OpenFOAM and CFD basic theory

OpenFOAM is a free, open source CFD software widely used in various fields of engineering, which counts with an extensive range of features to solve anything from complex fluid flows involving chemical reactions, turbulence and heat transfer, to acoustics, solid mechanics and electromagnetics.[30] In a more technical way, it could be said that OpenFOAM is an object-oriented C++ framework that can be used to build a variety of computational solvers for problems in continuum mechanics with a focus on finite volume discretization.[27]

Despite its many advantages like a friendly syntax for partial differential equations, no license costs and wide range of applications and models ready to use; OpenFOAM also presents several drawbacks, like the absence of an integrated graphical user interface and the lack of a detailed guide for programmers, making learning progress slow.

Anyway, the aim of this section is not to offer a complete overview about OpenFOAM characteristics and functioning, but to give some hints about the way of facing a general simulation case so that future explanations might be better understood.

Apart from the information explained in this section, and some notions about porosity approach and heat transfer in future sections, further information about many other important and interesting features can be found in the OpenFOAM User Guide.[29]

### 2.1.1 File structure of OpenFOAM cases

As already mentioned, OpenFOAM is an open source and very flexible software and, although this may offer many advantages, it is not precisely user-friendly. Therefore, some issues might arise, such as the kind of information required to solve a particular case and how it could be implemented. So, in order to clarify future explanations about the modification of certain files, a brief overview of the file structure in OpenFOAM cases seems necessary.

There is a basic directory structure which contains the minimum set of files required to run an application. This structure is depicted in Figure 2.1 and organized in the following directories:

- **Constant directory**, which contains information describing the mesh in a subdirectory called *polymesh* and files containing physical properties and models used for the specific application (*transportProperties*, *turbulenceProperties*, *thermophysicalProperties*...).
- **System directory**, where parameters related to solution procedure are set. The required minimum files are three: *controlDict* in which run control parameters are set including the solver used, start/end time, time step and parameters for data output; *fvSchemes* where discretization schemes used are specified; and *fvSolution* in which equation solvers, tolerances and under-relaxation factors among other controls are set. Additional files depending on the case might be found in this directory too (e.g. *blockMeshDict* for the mesh generation using the application blockMesh). In this thesis work two important files will be found here: *fvOptions* to specify the porosity coefficients for each cell region and *setFieldsDict* to set the volumetric heat sources in the corresponding active zone regions.

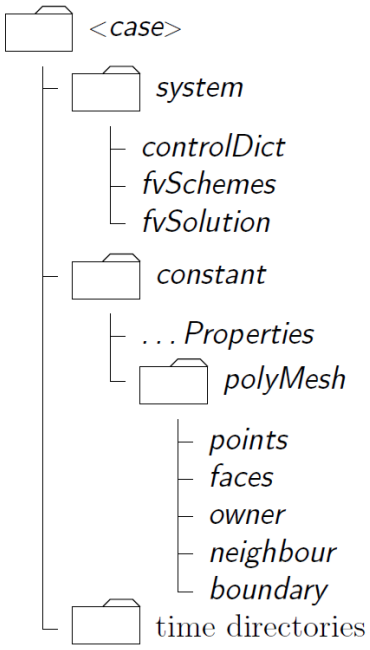



Figure 2.1. Case directory structure. Source: OpenFOAM User Guide.[29]

- **"Time" directories**, which contain files of data for the particular fields studied in the specific case: velocity and pressure in general, temperature when energy equations are used, turbulence parameters for the different turbulence models when used... Initial values for boundary and internal field conditions must be given in a directory called *0*. Subsequent time directories are simulation results written to file by OpenFOAM.

Each of the files is also structured in a specific way so that information can be correctly extracted and used by the application. The only files that will be completely generated specifically for the cases in this thesis are *fvOptions* for the definition of porosity coefficients (Section 2.4) and *setFieldsDict* for the set of volumetric heat sources (Section 2.5). The rest of files have a basic and common structure.

The file *blockMeshDict* will not be found in any of the cases contained in this thesis, as mesh generation is made using GMSH instead of blockMesh. This procedure will be correctly explained in Section 2.3.

### 2.1.2 Governing equations: Navier-Stokes and turbulence models

Governing equations, as for any fluid dynamic problem, are Navier-Stokes ones which are, basically, conservation equations of mass (continuity), momentum and energy. They are usually expressed by their derivative form as follows:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 \\ \frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) &= +\rho \mathbf{g} - \nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{S_u} \\ \frac{\partial (\rho e_t)}{\partial t} + \nabla \cdot (\rho e_t \mathbf{u}) &= -\nabla \cdot (p\mathbf{u}) + \nabla \cdot (k \nabla T) + \boldsymbol{\Phi} + \boldsymbol{S_e} \end{aligned} \tag{2.1}$$

Where $\mathbf{u}$, $p$, $e_t$ and $T$ represent the flow-field variable velocity, pressure, internal energy and temperature respectively. Density is given by $\rho$ and thermal conductivity by $k$. Additionally, $\mathbf{g}$, $\boldsymbol{\tau}$ and $\Phi$ respectively denote gravity, viscous stress tensor and dissipation function. Source terms are given by $\boldsymbol{S_u}$ and $\boldsymbol{S_e}$, representing additional body forces (like the resistance of porous media from Section 2.4) and heat sources (as volumetric ones from Section 2.5).

But Navier-Stokes equations require to be complemented in order to solve a fluid dynamic case by additional thermodynamic variables equations to close the problem, relationships to relate transport properties and closure equations for the turbulence models (when needed). In fact, one of the biggest issues when facing these equations is how to deal with turbulence. A detailed explanation on the definition of turbulence, its characteristics and the different numerical methods developed to capture its effects can be found in Versteeg and Malalasekera (2008).[50] But, in this section, only small hints about turbulence models will be introduced.

As Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS) are very computing demanding, the most used method for engineering flow calculations over the last decades has been Reynolds-Averaged Navier-Stokes (RANS), and simulations

of a reactor core of these characteristics is not an exception. When conducting DNS simulation for an incompressible flow, the starting point would be:

$$\begin{aligned} \nabla \cdot (\mathbf{u}) &= 0 \\ \frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) &= -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} \end{aligned} \tag{2.2}$$

Where the original equations have been simplified to facilitate the explanation (no energy, no gravity and no sources) as well as newtonian flow has been assumed (affecting how viscous stress tensor is computed). This corresponds to a deterministic approach to turbulence in which all the eddies and effects are directly calculated. Using this method requires a very fine mesh and very small time steps, having a great impact on computing resources.

On the other hand, for RANS method, attention is focused on the mean flow and the effects of turbulence on the mean flow properties, which means that all turbulence is modelled and no eddies are solved (not even large ones like in LES). So, a Reynolds decomposition is made, which consists in splitting the instantaneous value of primitive variables (i.e. velocity **u**) into a mean component and a fluctuating one (respectively **U** and **u'** following the example). An averaging of the different quantities with a few rules to simplify equations is also performed. The result is that governing equations are now expressed as:

$$\begin{aligned} \nabla \cdot (\mathbf{U}) &= 0 \\ \frac{\partial(\rho\mathbf{U})}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) &= -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{U} - \nabla \cdot (\overline{\mathbf{u'}\mathbf{u'}}) \end{aligned} \tag{2.3}$$

Where, if time derivative is present, the method is usually called Unsteady Reynolds-Averaged Navier-Stokes (URANS). For the sake of simplicity, in following steps time derivative will be neglected. The last term from the previous equation, which contains all velocity fluctuations, is known as Reynolds stress tensor and is commonly defined as:

$$\boldsymbol{\tau}^R = -\rho(\overline{\mathbf{u'}\mathbf{u'}}) = -\begin{pmatrix} \rho u'u' & \rho u'v' & \rho u'w' \\ \rho v'u' & \rho v'v' & \rho v'w' \\ \rho w'u' & \rho w'v' & \rho w'w' \end{pmatrix} \tag{2.4}$$

Whereas the rest of the terms from Equations 2.3 can be computed from the mean flow, solving velocity fluctuations requires very fine meshes (and small time-steps in the case of unsteady simulation), so Reynolds stresses from Equation 2.4 are considered as unknowns and, hence, must be predicted through models. Although it is possible to derive its own governing equations for the Reynolds stress tensor (for example obtaining Reynolds Stress Models (RSM)), one of the most common approaches is the use of the Boussinesq hypothesis which states that: Reynolds stresses might be proportional to mean rates of deformation multiplied by a constant (which is defined as the turbulent eddy viscosity $\mu_t$). Thus, Reynolds stress tensor is predicted as:

$$\boldsymbol{\tau}^R = -\rho(\overline{\mathbf{u'}\mathbf{u'}}) = 2\mu_t\overline{\mathbf{D}}^R - \frac{2}{3}\rho k\mathbf{I} = \mu_t\left[\nabla\mathbf{U} + (\nabla\mathbf{U})^{\mathrm{T}}\right] - \frac{2}{3}\rho k\mathbf{I} \tag{2.5}$$

Where $\overline{\mathbf{D}}^R$, $\mathbf{I}$, $k$ and $\mu_t$ respectively denote the Reynolds averaged strain-rate tensor, the identity matrix, the turbulent kinetic energy, and the turbulent eddy viscosity.

The last term represents the normal stresses arising from Boussinesq approximation and, therefore, is analogous to the pressure term that arises in the viscous stress tensor.

Hence, combining Equations 2.3 and 2.5, the following expression for the momentum equation can be obtained:

$$\nabla \cdot (\mathbf{UU}) = -\frac{1}{\rho}\left(\nabla p + \frac{2}{3}\rho\nabla k\right) + \nabla \cdot \left[\frac{1}{\rho}(\mu + \mu_t)\nabla \mathbf{U}\right] \tag{2.6}$$

At the end, two new variables ($k$ and $\mu_t$) have appeared in the equations when applying the Reynolds averaging and the Boussinesq hypothesis. These extra variables ought to be modelled with a turbulence model which adds several additional conservation equations to the previous ones (the number of added equations depends on the model used). RANS method has been widely validated in different engineering fields, including this kind of nuclear technology as showed by Roelofs *et al.* (2019).[41]

Two of the most widely used turbulence models are $k-\epsilon$ and $k-\omega$ with their multiple variations. In each of them, turbulence is described by two parameters and their corresponding two conservation equations. These descriptors are: $k$ representing the turbulent kinetic energy (as already mentioned before) and $\epsilon$ denoting the turbulence dissipation rate (rate at which $k$ is transformed into thermal internal energy) or $\omega$ as the turbulent eddy dissipation frequency (rate of conversion of $k$ into thermal internal energy).

How these models are obtained and what their specific characteristics are will not be dealt with here, but their original description can be found in Launder *et al.* (1974)[23] and Wilcox (1998)[52] respectively. However, it seems important to point out that the main difference between these two models is that in case of $k-\omega$ mesh refinement in the walls must be much higher as boundary conditions are not modelled by wall functions and $y^+$ needs to be close to 1, although it has a superior performance for wall-bounded boundary layers, free shear and low Reynolds number flows. That implies a finer mesh and, hence, higher computing times and resources.

Many other models can be found in literature: some still based on Boussinesq hypothesis as Spalart-Allmaras model[47] and hybrid models between $k-\epsilon$ and $k-\omega$; and some others based on stress transport model as the previously mentioned RSM.[22]

In the end, any turbulent flow transport equation for a general variable $\varphi$ follows this structure:

| RATE OF CHANGE TERM ($\varphi$ rate of increase in the fluid element) | + | CONVECTIVE TERM ($\varphi$ net rate of flow out of the fluid element) | = | DIFFUSIVE TERM ($\varphi$ rate of increase due to diffusion) | + | SOURCE TERM ($\varphi$ rate of increase due to sources) |
|---|---|---|---|---|---|---|

This general structure can be mathematically expressed by the following general formulation:

$$\frac{\partial(\rho\varphi)}{\partial t} + \nabla \cdot (\rho \mathbf{u}\varphi) = \nabla \cdot (\Gamma \nabla \varphi) + s \tag{2.7}$$

where $\varphi$ can be substituted by 1 (continuity equation), velocity components (momentum equations), i (energy equation), or turbulent parameters depending on the model

($k$, $\epsilon$, $\omega$...). The diffusion coefficient $\Gamma$ has to be correctly defined for each case. The term $s$ denotes source expressions, in which the gradient of pressure is included for the case of momentum conservation equations.

In conclusion, these conservation equations are more complex than it may appear, as they are non-linear, coupled and difficult to solve. Although experience shows that the Navier-Stokes equations describe the flow of a Newtonian fluid accurately, analytical solutions by the existing mathematical tools can only be achieved in a few and very simple cases, and simplification assumptions are also usually required. For this reason CFD, based in the finite volume method, is extensively used in many engineering fields. In the two following sections, the basis of this methodology will be explained.

### 2.1.3 Discretization schemes

In the finite volume method, solution domain is divided into a finite number of arbitrary control volumes which will be called *cells*. Although they can be of any shape, elements need to be convex and its faces planar. Solution is sought in each of these control volumes (denoted as $V_c$), thus Navier-Stokes equations need to be integrated over each of them. When integrating the general formulation of a conservation equation 2.7, and applying the Gauss theorem, the following general expression can be obtained:

$$\int_{V_c} \frac{\partial(\rho\varphi)}{\partial t} dV + \int_{S_c} \rho\varphi \mathbf{u} \cdot \mathbf{n} dS - \int_{S_c} \Gamma \nabla\varphi \cdot \mathbf{n} dS - \int_{V_c} s dV = 0 \tag{2.8}$$

Where the second and third terms can be respectively understood as convective and diffusive fluxes ($\mathbf{J}_{\text{conv}} = \rho\varphi\mathbf{u}$ and $\mathbf{J}_{\text{diff}} = \Gamma\nabla\varphi$). Continuity equation has a different structure and needs to be solved differently:

$$\int_{V_c} \frac{\partial\rho}{\partial t} dV + \int_{S_c} \rho \mathbf{u} \cdot \mathbf{n} dS = 0 \tag{2.9}$$

When working on unsteady simulations, a discretization of time derivatives is performed. A first order approximation would be like:

$$\int_{V_c} \frac{\partial(\rho\varphi)}{\partial t} dV \cong \frac{(\rho\varphi)_P^{t+1} - (\rho\varphi)_P^t}{\Delta t} V_P \tag{2.10}$$

However, different order approximations can be used.

Additionally, spatial discretization is aimed to compute the face values of fields whose volumes are stored at the cell centre. For example, surface integrals from Equation 2.8 require the evaluation of face fluxes from information in the node (cell centre). These face fluxes are due to convective and diffusive transport. The first one requires face values $\varphi_e$:

$$\int_{S_e} \mathbf{J}_{\text{conv},e} \cdot \mathbf{n} dS \cong \rho\varphi_e u_e S_e \tag{2.11}$$

On the other hand, diffusive transport requires face gradient values $\{\nabla\varphi\}_e$:

$$\int_{S_e} \mathbf{J}_{\text{diff},e} \cdot \mathbf{n} dS \cong \Gamma_e \{\nabla\varphi\}_e S_e \tag{2.12}$$

Both required values can be obtained by diverse approximation techniques of different order, which means different discretization schemes. Their suitability in each specific case ought to be evaluated according to boundedness (convergence), conservativeness (consistency) and transportiveness (stability).

In OpenFOAM, information related to discretization of the domain (the mesh) is contained in the directory *constant/polyMesh*. Additionally, this software also includes a vast number of time and spatial discretization schemes, from which only a few are typically recommended for real-world, engineering applications.[29] They are set in *system/fvSchemes* dictionary and are subdivided into the following categories:

- *timeScheme*: first and second time derivatives terms, $\partial/\partial t$ and $\partial^2/\partial^2 t$
- *gradSchemes*: gradient terms, $\nabla$
- *divSchemes*: divergence terms, $\nabla\cdot$
- *laplacianSchemes*: laplacian terms, $\nabla^2$
- *interpolationSchemes*: cell to face interpolations of values
- *snGradSchemes*: component of gradient normal to a cell face
- *wallDist*: distance to wall calculation, where required

In general, recommended discretization schemes will be used in the different cases simulated in this thesis.

### 2.1.4 Solution algorithms and numerical methods

Solution of the momentum equations presents two main problems:

- Convective terms contain non-linear quantities (derivative of squared velocity), while linearized-discretized equations are proposed from previous section. This affects the solution and, hence, an iterative approach is required.
- All momentum equations are strongly coupled because every velocity component appears in each momentum equation and, also, in continuity equation. Nevertheless, the most complex role is played by pressure, as there is no apparent equation for it.

The strategy facing these problems is to guess pressure following the idea that, when setting the correct pressure field, continuity equation will be satisfied by the resulting velocity distribution. For this purpose, iterative algorithms based on a prediction and correction process are used, solving the pressure field so that velocity components satisfy the continuity equation. This process is performed by three main available methods:

- Pressure Implicit with Splitting of Operators (PISO) is specially useful for unsteady flow problems or for meshes containing cells with higher than average skew.

- Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) is a robust default scheme, used for steady-state problems.
- PIMPLE is a combined algorithm of the previous two ones mainly used for transient problems.

These methods correspond to a pressure-based approach which was initially developed for low-speed incompressible flows. However, it has been extended and reformulated to solve and operate for a wide range of flow conditions beyond its original intent. Pressure-based methods are the default option in most of CFD solvers, including OpenFOAM.

Another widely known approach is the density-based one, where the continuity equation is used to obtain the density field while the pressure field is determined from the equation of state. It was developed for high-speed compressible flows, but it has also been improved and extended.

In OpenFOAM, the equation solvers, tolerances and algorithms are controlled from the *system/fvSolution* dictionary. It is organised in a set of subdictionaries:

- *solvers* specifies each linear solver used for solving each equation. They will iterate until reaching any of the tolerance values set by the user or reaching a maximum number of iterations. Selection of this tolerance value is of vital importance, as bad tolerances might lead to inaccuracies and wrong physics.
- *relaxation factors* is a feature typical of steady solvers using the SIMPLE method. The idea behind these factors is finding a balance between the old and the new values.
- *PISO*, *SIMPLE* or *PIMPLE*, where the pressure-velocity coupling method and its options are set.

How to set all these features highly depends on the specific case to be solved, and will be correctly defined in Chapter 3 for each of the simulations performed.

## 2.2 ALFRED characteristics

The objective of this section is to introduce all the important parameters from the reactor that might be of interest in the simulations and all the previous calculations and processes: definition of the geometry, implementation of models, adjustment of options for solvers...

As a thermohydraulic analysis is performed, there is not an initial interest on the neutronic part of the reactor, so no explanations or features about the actual neutronic processes inside the core will be given. However, it should be highlighted that thermohydraulic and neutronic processes are highly coupled because the fuel and coolant temperatures affect the fission reaction cross section, i.e. neutronic parameters. Although nowadays there are more and more codes that couple neutronics and thermal-hydraulics, e.g. using OpenFOAM for solving both, this process might involve the external coupling of different codes: one tackling the neutron density field (neutronics part) and the other tackling the flow and temperature fields (thermal-hydraulics).[24,35]

### 2.2.1 The core design

In this section, a quick overview of ALFRED reactor core most important geometric characteristics will be given. The objective is to define all the elements that might be useful for the mesh generation and the calculation of the different parameters used in the simulation. All this information has been extracted from Grasso *et al.* (2014)[15] and from LEADER project internal documentation.

While one of the proposed typical LFR reactor cores is made up of 427 FAs and 24 absorber elements (control and safety rods), ALFRED reactor core is formed by an hexagonal lattice bundle with a total of 297 positions. From them, 171 correspond to FAs forming a cylindrical core and are subdivided into two radial zones with different plutonium enrichment guaranteeing an effective power flattening. Two surrounding rings formed by 110 DEs (which are geometrically identical to FAs) serve as neutronic reflector. Additionally, two independent control rod systems are found, 12 positions correspond to the CR system which is used for power regulation and reactivity swing compensation, and the other 4 positions correspond to the SR system which is simultaneously used together with the former for SCRAM purposes (reliable and safe shutdown).[16] This configuration is shown in Figure 2.2.

In the following paragraphs, a description of these different elements will be introduced, but, first, it seems important to mention that CR and SR systems design has been adapted from the CDT-MYRRHA project.[21]

**Fuel Assemblies and Dummy Elements** They both have exactly the same geometric configuration and their only difference is that DE pin bundle is not charged with fuel and, hence, power production inside of them is almost negligible and their

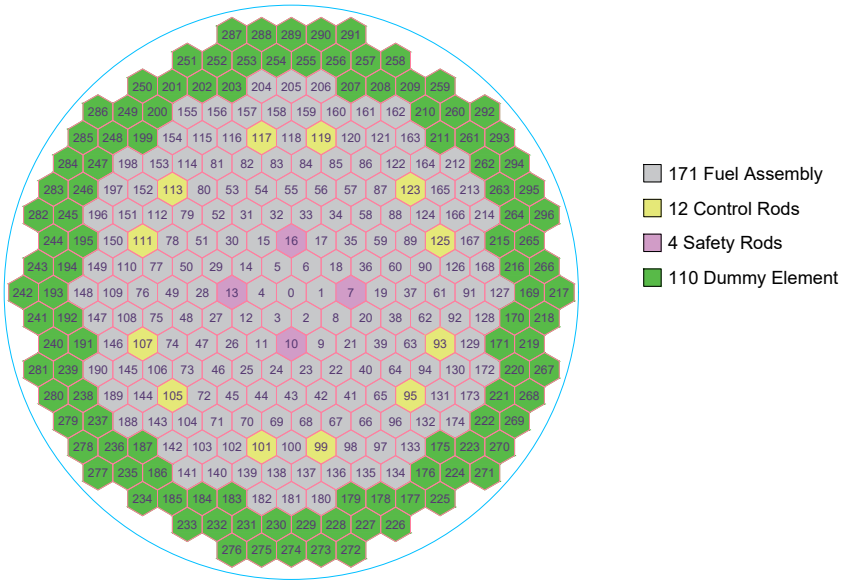



Figure 2.2. ALFRED reactor core layout.

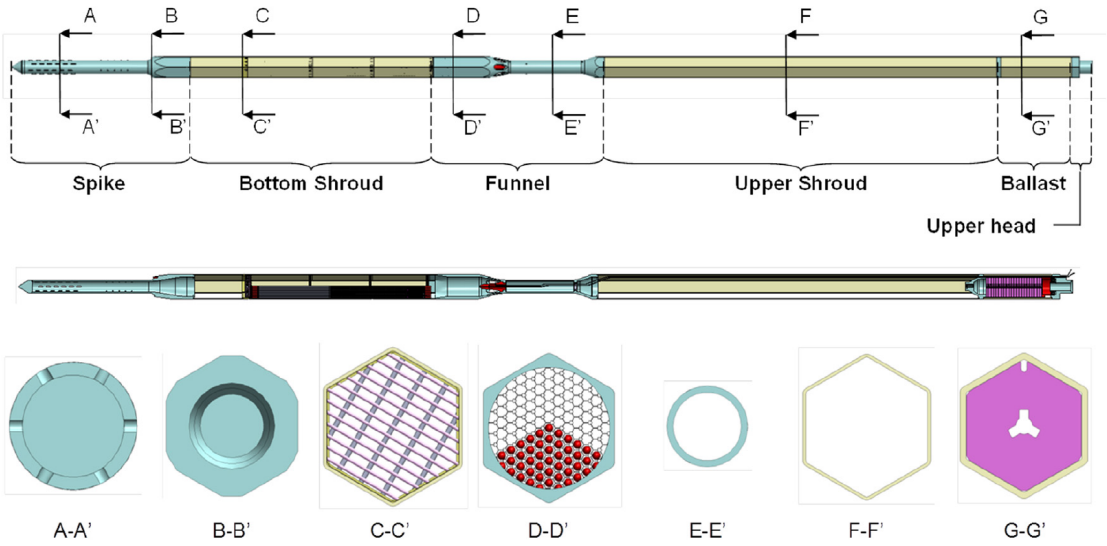



Figure 2.3. Axial sketch of a general FA from ALFRED nuclear reactor. Source: Grasso *et al.* (2014).[15]

function is to act as a neutron reflector. Geometry for FAs and DEs consists of an hexagonal wrapper containing the pin bundle, support grid and grid spacers among others. The different sections composing them can be observed in Figure 2.3. This structure is made up of the following six parts:

- The Spike, provided with multiple orifices, which fits into the lower diagrid to ensure the correct positioning of the FA and serves as inlet for the coolant liquid lead inside the core. There, a so-called *gagging device* is placed right after the inlet orifices section, whose aim is regulating the mass flow rate through the FA. This configuration is depicted in Figure 2.4, where it can be observed that a first group of orifices provides the inlet of coolant into the FA, while a second group

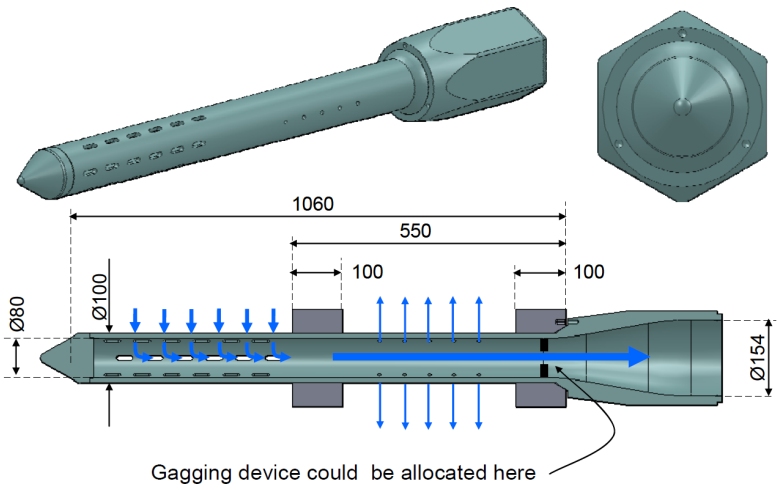



Figure 2.4. Detailed view of the spike, indicating the gagging device placed just after the orifices region. Source: Internal LEADER documentation.

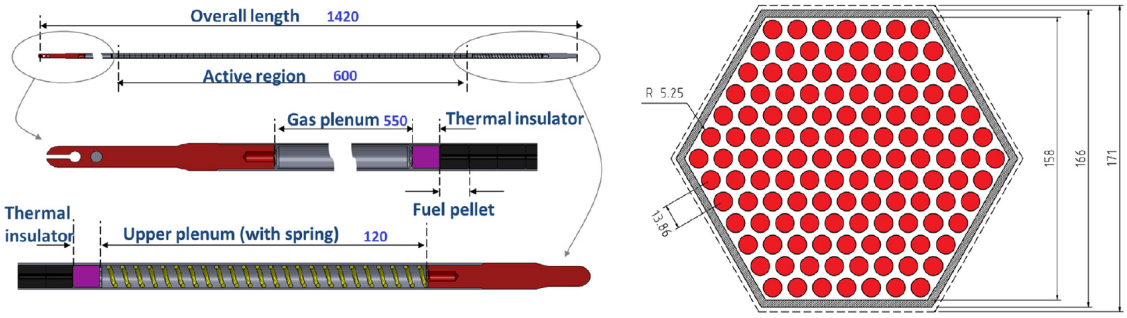



Figure 2.5. ALFRED fuel pin (left) and pin bundle cross-section (right). Source: Grasso *et al.* (2014).[15]

allows the outlet of spare coolant due precisely to the gagging device regulating the mass flow rate (hence, regulating the pressure drop at the inlet). Finally, the spike also guarantees the lead flow into the by-pass region between adjacent FAs.

- The Bottom Shroud where, after an empty space, the pin bundle is sustained by the corresponding support grid and grid spacers. Each of the pins has a length of 1.42 m from which 60 cm correspond to the so-called *active zone*, where fuel pellets are contained. Investigations suggested the viability of this configuration where the bundle groups a total of 127 pins arranged in a triangular lattice with 13.86 mm pitch length. All these main geometric characteristics and some other are depicted in the Figure 2.5.

- The Funnel geometry is specially designed to facilitate the hot lead outlet from the core towards steam generators, from where cooled lead will be recirculated again into the core. The design of this part counts with a *nose* (red device in Figure 2.3) able to host neutron and temperature instrumentation.

- The Upper Shroud is the structural element connecting the Funnel under the lead free surface and the Ballast element well above the lead free surface.

- The Ballast, which is placed in the upper part of the FA, counterbalances buoyancy forces while refueling and, hence, maintains its position when the upper grid is not present.

- The Upper Head is used to connect the FA to the upper grid and guarantees its position during normal operation condition.

The three last parts described before do not generate thermal power and are not cooled by lead as they emerge from coolant free surface. Thus, thermohydraulic analysis is not applicable around these elements and will not be considered in this thesis. In conclusion, the only modelled regions will be the spike, all the elements composing the bottom shroud and the outlet funnel.

Some special considerations about the grid spacers seem to be necessary, as they represent critical points for the flow obstruction, causing hotspots that may derive in clad material failure. For that reason, analysis about pressure losses and temperature

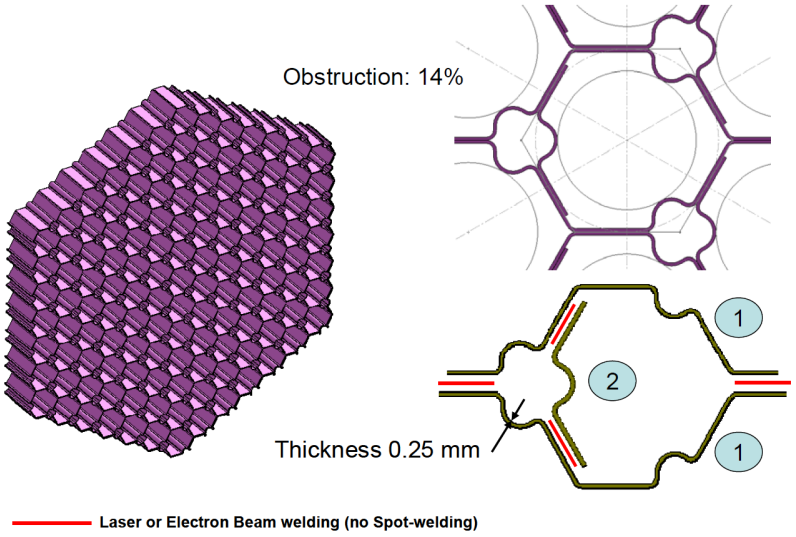
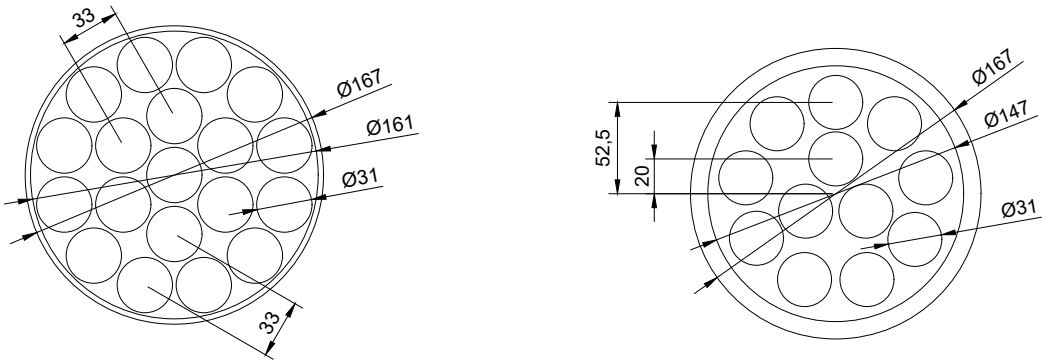



Figure 2.6. ALFRED FA grid spacer schematic drawings. Source: Internal LEADER documentation.

distributions due to different grid spacer designs were conducted as part of the LEADER project.[26] A final candidate design was chosen and is depicted in Figure 2.6. According to this, geometrical obstruction to the flow caused by the grid spacer is 14%. This value will be later used when calculating pressure drop through the grid spacer.

**Control rods** Each CR is made of a cylindrical bundle of 19 absorber pins (see layout in Figure 2.7a) cooled by primary lead. They are positioned in a guiding tube occupying a position in the outer zone of the core map (the one with higher fuel enrichment) as depicted in Figure 2.2 (colour yellow). They are inserted upwards by the actuation of motors, but an electromagnetic connection allows a rapid insertion into the core exploiting buoyancy effects in case of SCRAM. As observed in Figure 2.8, withdrawn position is below the active zone, so no matter whether inserted or




(a) CR cross-section design.

(b) SR cross-section design.

Figure 2.7. Cross-section designs for CR and SR systems.

not, the total pressure drop across the CR will be almost the same.

**Safety rods** The SR system is similar to the previous one, it is made of a bundle of 12 absorbing pins, cooled by primary lead and positioned in a guiding tube occupying a position in the core map (colour pink in Figure 2.2). Cross-section configuration can be observed in Figure 2.7b. During normal operation, SR system is withdrawn, staying still atop the active zone. Their actuation is only due to safety reasons (SCRAM), unlocking an electromagnet and activating a pneumatic acceleration system, which rapidly pushes the rods into the core. The addition of a ballast atop SR system ensures the insertion of the rods counteracting buoyancy even at a reduced speed.

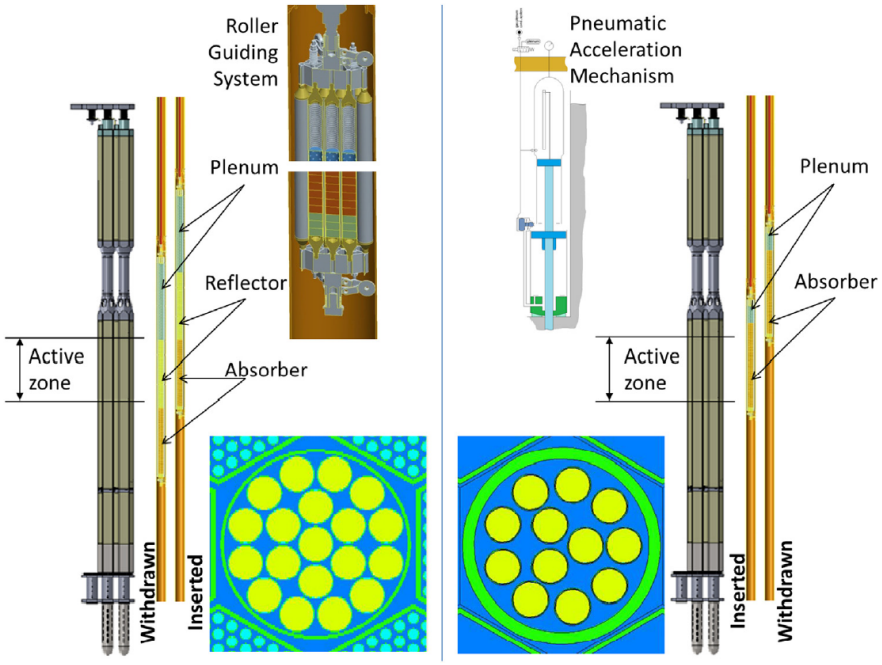



Figure 2.8. Schematics and cross-sections of ALFRED CRs (left) and SRs (right). Source: Grasso *et al.* (2014).[15]

### 2.2.2 Operating conditions and coolant fluid properties

Defining fluid properties is a critical point for thermohydraulic simulation. This information will be implemented in OpenFOAM as *thermophysicalProperties* or *transportProperties*, depending whether heat transfer process is simulated or not respectively. For the first case, thermophysical models concerning with energy, heat and physical properties are required. The choice of the corresponding models will condition the variables which value is needed. On the other hand, for the second case only transport variables must be defined and usually viscosity value is enough. The objective of this section is to present all these fluid variables that must be used in calculations and simulations.

Coolant process in LFR has been proposed using liquid pure lead and molten Pb-Bi eutectic alloy. The second one has been used mainly for military applications,

but the scarcity of bismuth and some radiative capture issues have focused attention on liquid pure lead for civil nuclear energy production. This represents the case of ALFRED, where the advantages of using this kind of coolant have already been discussed. Therefore, due to their importance, pure lead and Pb-Bi eutectic alloy fluid properties have been widely studied for the last years and a very complete summary of them was made by the NEA (2015).[28] These physical properties, as usually, have a significant dependence on temperature and pressure, so it is important to define the main operating conditions of the coolant fluid, which are presented in Table 2.1.

Table 2.1. Normal operating conditions for coolant fluid in ALFRED nuclear core.

| Property name | Units | Value |
|---|---|---|
| Inlet temperature, $T_{\text{in}}$ | [ºC] | 400 |
| Outlet temperature, $T_{\text{out}}$ | [ºC] | 480 |
| Free surface pressure, $p$ | [Pa] | 101325 |

Temperature range inside the core is not very high (around 80ºC according to Table 2.1), so that lead properties can be chosen for an average temperature of 440ºC. Those properties used in this thesis are found in Table 2.2.

Table 2.2. Liquid pure lead properties at 440ºC. Source: NEA (2015).[28]

| Property name | Units | Value |
|---|---|---|
| Density, $\rho$ | [kg/m$^3$] | 10,503 |
| Kinematic viscosity, $\nu$ | [m$^2$/s] | $1.90 \times 10^{-7}$ |
| Dynamic viscosity, $\mu$ | [Pa·s] | 0.002 |
| Thermal conductivity, $k$ | [W/(mK)] | 17.15 |
| Specific heat at constant pressure, $c_p$ | [J/(kgK)] | 145.9 |
| Prandtl number, Pr | [-] | 0.01698 |
| Turbulent Prandtl number, Pr$_t$ | [-] | 0.9 |

Additionally, density dependence on temperature may have an impact on simulations, specially when working with natural recirculation. For that reason, Boussinesq approximation for buoyancy effect will be used for expressing the density in the momentum equation as:

$$\rho = \rho_0[1 - \beta(T - T_0)] \tag{2.13}$$

Where temperature is in kelvins and density in [kg/m$^3$]. The values for parameters $T_0$, $\rho_0$ and $\beta$ (volumetric expansion coefficient) will be extracted from the linear temperature dependence proposed by Sobolev (2008):[46]

$$\rho = 11,441 - 1.2795 \cdot T \tag{2.14}$$

Setting $T_0 = 400$ [ºC] $= 673$ [K], parameter $\rho_0 = 10,580$ [kg/m$^3$] can be calculated. Knowing the value for this two parameters, it is trivial obtaining $\beta = 1.209 \times 10^{-4}$ from Equation 2.14.

The implementation of these values in the different physical models will be explained for each specific case in Chapter 3.

## 2.3 Geometry and mesh generation: *layers approach* and GMSH

As already explained, equations are spatially discretized and, hence, the domain is required to be divided into discrete cells where these equations can be solved, creating a mesh. Good quality of this mesh is wanted because solution accuracy and stability deteriorate when mesh cells deviate from ideal shape. In fact, bad mesh quality can cause convergence difficulties, bad physics description and diffuse solution. Therefore, mesh quality must be correctly assessed before running any simulation. Additionally, it is important to mention the trade off between efficiency and accuracy: although finer meshes capture physical processes and geometric details better, computing times exponentially grow with refinement. The usual approach is to refine for high solution gradients (e.g. close to walls) and fine geometric details while coarse mesh is used elsewhere.

All these considerations confirm that mesh generation is a critical point where many factors influence the geometry design. First and most important, mesh must capture all the real geometric details of the domain. In this sense, the core consists on a cylindrical vessel with an arrange of numerous hexagonal prisms (FA, DE, CR and SR), each of them containing its corresponding elements (pin bundle, grid spacers, support grid...). This results in a really complex geometry, where computing resources and even stability and convergence of the case can be compromised. This is the main motivation in choosing a porous media approach, where all the components inside the core elements are not modelled by the mesh and their impact on flow is simulated by a porosity coefficient.

But, apart from the geometric details, it is also important to think about the different physical processes that want to be simulated as well as models and solver settings. For example, the choice of turbulence model will condition the refinement of the mesh near the walls, depending on whether wall functions are used or not. Additionally, a porous media approach will be used, so specific regions for each porosity zone had to be defined in the geometry. Power generation in the active zone of each FA forced also the creation of a specific region where a heat source could be set.

All the reasons expressed before derived in the so called *layers approach*, in which each FA, represented by an hexagonal prism of constant cross-section area $A = 253.23$ [$\text{cm}^2$], is divided into zones (layers) corresponding to its different regions (inlet orifices, support grid, pin bundle...). Each of these layers will have a characteristic porosity value (translated into a pressure drop) and power generation that can be later assigned in OpenFOAM. This particular structure is depicted in the top Figure 2.9, where *orf* corresponds to inlet orifices, *empty* to the empty tube region, *sg* to the support grid, *z1 z2 z3* to the three different pin bundle regions, *gs1 gs2 gs3* to the three grid spacers, *act1 act2* to the active region, and *funnel* to the outlet funnel. These layers are designed according to the real dimensions of a FA, with additional domain inlet and outlet regions where initial boundary conditions can be set and resulting distribution of flow and temperature fields will be observed. But, in order to reduce the number of cells, and consequently reduce computing time and resources, inlet orifices (*orf*), empty tube (*empty*) and outlet funnel (*funnel*) layers lengths are shortened. This decision was motivated by three main facts: pressure drop is set by a porosity

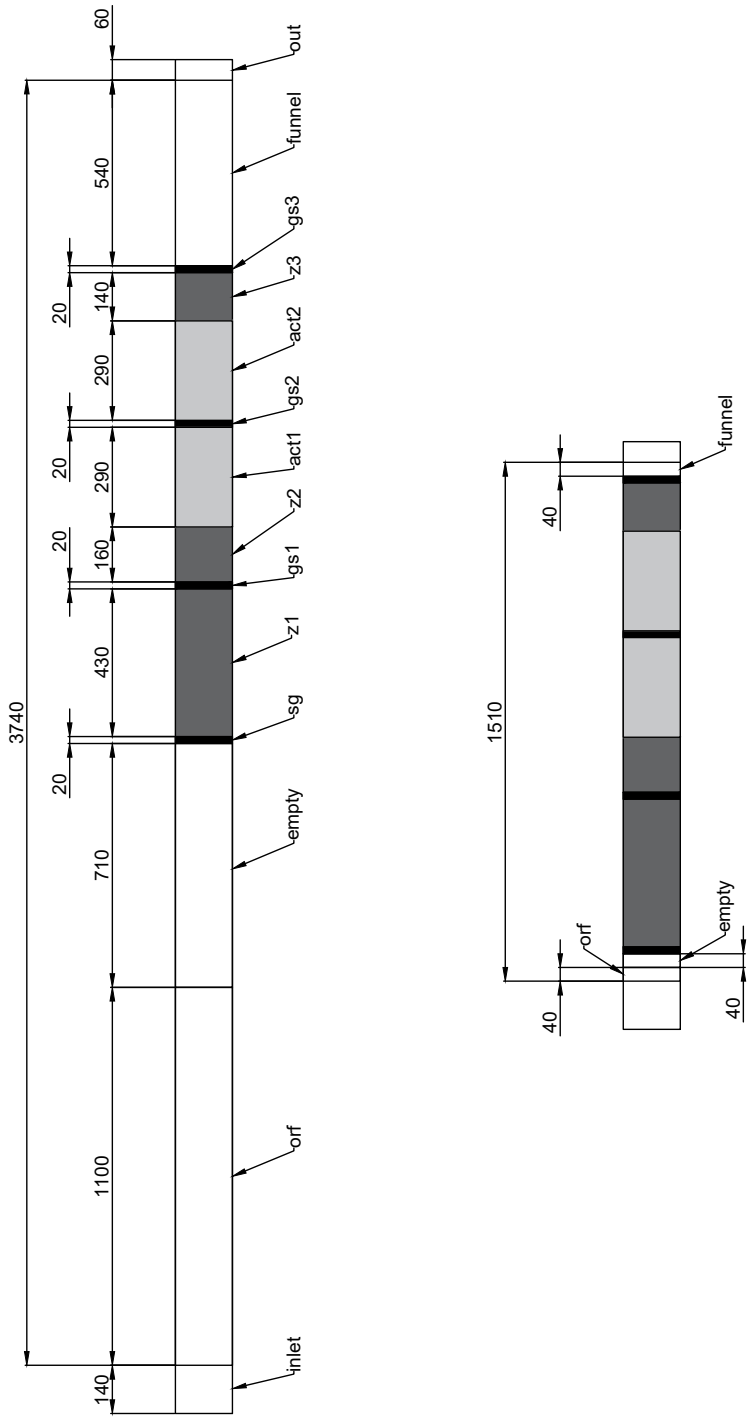
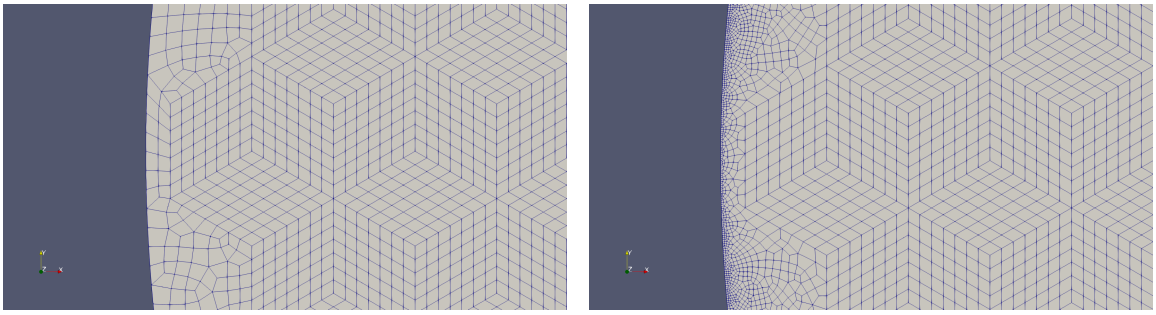



Figure 2.9. Layers approach of a single general FA. Top figure represents the division according to the real dimensions and the bottom one shows the reduction of the first layers.



(a) Without refinement, same grid size of 2 cm. (b) High refinement, grid size of 0.2 cm.

Figure 2.10. Different refinement levels of the space left between FAs and inner vessel wall. General FA grid size is kept constant as 2 cm.

coefficient that can be adjusted considering real/simulated length, there is no interest in studying the flow development inside these regions, and heat transfer inside these regions is not important. The result can be observed in the bottom Figure 2.9.

Each of those *layers* will be completely defined by some geometric parameters: real cross-section area $A$, real model length $L$ and simulated mesh length $L_{\text{mesh}}$. Values for those parameters in each layer are gathered in Table 2.3.

Table 2.3. Geometric parameters for each of the layers in a general FA.

| | $A$ [cm$^2$] | $L$ [cm] | $L_{\text{mesh}}$ [cm] |
|---|---|---|---|
| **Inlet orifices** | 253.23 | 110 | 4 |
| **Outlet funnel** | 216.19 | 54 | 4 |
| **Empty tube** | 216.19 | 71 | 4 |
| **Pin bundle** | 106.23 | 137 | 137 |
| **Grid spacer** | 91.36 | 2 | 2 |
| **Support grid** | 87.11 | 2 | 2 |

Cell type is also a critical point. The geometric mesh simplicity described before allows to use hexahedral elements, which guarantees, comparing to tetrahedral cells, fewer elements and a naturally anisotropic domain. In addition, the mesh can be coarse in general (grid size = 2 cm), as no geometric details are modelled.

Although a structured (or regular) grid can be used for the hexagonal prisms composing the core elements, an unstructured grid is needed for the space left between them and the inner vessel wall due to its complex geometry. This results in a worse mesh quality and a more computer time demanding solution. But this narrow space introduces another problem: mesh near the vessel must be fine enough so that the suitable wall functions are able to capture correctly the physics. Refining this zone to obtain a valid $y^+$ value results in complex unstructured grids that may affect mesh quality and computing times even more. This refining process is shown in Figure 2.10. Two solutions can be considered to fix this problem: first, refining the whole mesh, even the hexagonal core assemblies, but the computing advantage of using a coarse mesh on porous media approach would be partially wasted; or, secondly, neglecting that narrow space and setting a slip wall boundary condition in the outer assemblies of the core. The latter option was chosen in this thesis work because

flow passing through that zone (by-pass flow) is almost negligible and no important thermohydraulic process takes place there. So, the vessel wall was removed and only the hexagonal core assemblies were modelled as can be observed in Figure 2.11. Three basic boundaries are considered in this geometry: inlet, outlet and wall (formed by some of the walls from outer core assemblies). Additionally, 4158 cell zones (14 layers by 297 core elements) are defined as described before in Figure 2.9.

Geometry and mesh were generated using the software Gmsh, which is an open source 3D finite element mesh generator with a built-in CAD engine.[14] This tool was preferred over the basic *blockMesh* as some programming functions can be used to help in building repetitive geometries, which is very useful when creating the hexagonal matrix of the core. Gmsh is provided with its own scripting language (*.geo* files), but C++, C, Python or Julia Application Programming Interface (API) can be also used. In Appendix A the script corresponding to the *.geo* file for the mesh generation defined in Figure 2.11 is included. Once the geometry script is loaded and the 3D mesh generated, it can be exported as a *.msh* file which is later converted to OpenFOAM's format by the following utility:

```
1 > gmshToFoam "mesh_name".msh -case "case_name"
```

When the mesh is imported into the case, it is vital to check the different boundaries of the geometry, specially if the names match with those set in initial boundary conditions and if wall patch type is correctly assigned. This information is located in the file *constant/polymesh/boundary* and can be consulted by any text editor (*nano*, *vim*, *gedit*...).

Dimensions of the mesh must also be checked. In this case, as indicated in the script, dimensions used for the mesh generation are set in centimetres, while default length unit in OpenFOAM are metres. In order to change the units of the mesh from

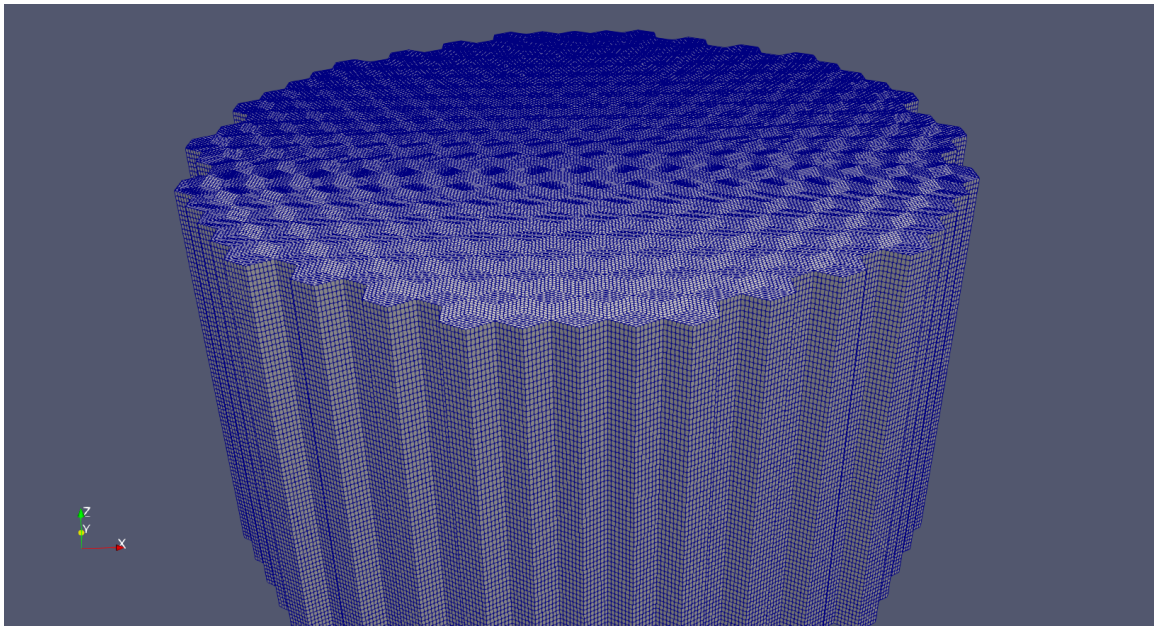


Figure 2.11. Final core mesh configuration with a general cell size of 2 cm as viewed from Paraview.

centimetres to metres, the following command is used:

```
> transformPoints -scale '(0.01 0.01 0.01)'
```

Finally, the general properties of the mesh can be checked using the utility *checkMesh*. The output produced in this specific case is the following:

```
Mesh stats
points:           3748585
faces:            11082771
internal faces:   10921365
cells:            3667356
faces per cell:   6
boundary patches: 3
point zones:      0
face zones:       0
cell zones:       4158

Overall number of cells of each type:
hexahedra:     3667356
prisms:        0
wedges:        0
pyramids:      0
tet wedges:    0
tetrahedra:    0
polyhedra:     0

Checking topology...
Boundary definition OK.
Cell to face addressing OK.
Point usage OK.
Upper triangular ordering OK.
Face vertices OK.
Number of regions: 1 (OK).

Checking patch topology for multiply connected surfaces...
Patch    Faces    Points   Surface topology
inlet    43659    44101    ok (non-closed singly connected)
outlet   43659    44101    ok (non-closed singly connected)
wall     74088    74970    ok (non-closed singly connected)

Checking geometry...
Overall domain bounding box (-1.6245 -1.57963 0) (1.6245
   1.57963 1.71)
Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
Mesh has 3 solution (non-empty) directions (1 1 1)
Boundary openness (9.77293e-17 5.14911e-16 1.20966e-13) OK.
Max cell openness = 2.21918e-16 OK.
Max aspect ratio = 2.93922 OK.
Minimum face area = 0.000172149. Maximum face area =
```

```
    0.000292418.  Face area magnitudes OK.
Min volume = 3.44298e-06. Max volume = 3.57263e-06.  Total
    volume = 12.861.  Cell volumes OK.
Mesh non-orthogonality Max: 30.0313 average: 22.6495
Non-orthogonality check OK.
Face pyramids OK.
Max skewness = 0.500472 OK.
Coupled point location match (average 0) OK.

Mesh OK.

End
```

From this output it is observed that the number of boundary patches (3), the number of cell zones (4158), the type of cells (all of them hexahedra) and main domain dimensions (from domain bounding box) seem all correct. Additionally, main mesh quality parameters are kept within default value ranges, hence overall mesh quality seems to be correct.

Before running the corresponding solver, it is advisable to use the utility *renumberMesh* which renumbers the cell list in order to reduce the bandwidth, reading and renumbering all fields from all the time. Its effect is to make the linear system more diagonal dominant, speeding-up the linear solvers and, therefore, reducing computing time.

```
> renumberMesh -overwrite
```

After having generated the mesh and correctly prepared it following the previous steps, geometry of the case is ready for simulation by running the corresponding solver.

## 2.4 Porous Media approach in OpenFOAM

As already mentioned before, the task of simulating the whole geometry of the reactor core, including all the pin bundles, inlet orifices, outlet funnels and the rest of elements, can become almost impossible. A very detailed mesh would have to be used and a value of $y^+$ near 1 would be searched in order to simulate the entire boundary layer, which means paying special attention to refinements near the walls and close to possible obstructions. Also, powerful turbulence models and discretization schemes would be needed. The result is that computing times and resources would be immensely high and even the problem convergence itself would be compromised.

To avoid these problems, a porous media approach based on a source term in the Navier-Stokes equations can be used instead of a full scope one. This has been implemented in a wide range of engineering applications when geometry was too complicated to simulate, as for cooling systems by Hörmann *et al.* (2005)[19] and for heat exchangers by Hayes *et al.* (2008).[18] It has also been implemented for thermohydraulic simulations in advanced nuclear reactors, as for a pebble bed one by Dahl and Su (2017).[9] As stated by Domaingo *et al.* (2016),[11] these source terms

not only allow to model the physics of interest but also have a strong impact on the reliability, stability, and convergence of the numerics involved.

In order to implement the porous media approach, a source term $S$ has to be added to the Navier-Stokes equations 2.1, specifically to the momentum conservation ones. Although there are different models for the formulation of this source term, the one derived from Darcy-Forchheimer law is widely used in OpenFOAM and can be expressed in the case of simple homogeneous porous media as:[17]

$$S_i = -\left(\mu D + \frac{1}{2}\rho|u_{jj}|F\right)u_i \tag{2.15}$$

Where Darcy constant $D$ accounts for viscous losses and the Forchheimer constant $F$ for losses due to turbulent effects in the porous media. This will be the formulation used in this thesis, although the source term can also be modelled as a power law of the velocity magnitude:

$$S_i = -\rho C_0|u_i|^{(C_1-1)/2} \tag{2.16}$$

Where $C_0$ and $C_1$ are user defined empirical coefficients.

When assuming 1D flow, the previous expression for the Darcy-Forchheimer law can be rewritten in terms of pressure drop over a porous region of thickness $L$ as follows:

$$\Delta p = L\left(\mu D + \frac{1}{2}\rho F u\right)u \tag{2.17}$$

It is important to note here that $L$ refers to the porous region thickness in the generated mesh, which may or may not match with the actual real porous region thickness. The very same thing happens with $u$, which is referred to the simulated flow velocity and can match or not the actual real velocity inside the corresponding region. Hence, in order to avoid mistakes, they will be denoted as $L_{\text{mesh}}$ and $u_{\text{sim}}$ respectively from now on.

The present case will be considered 1D flow, as coolant flow is forced to pass upwards through the core, in z-direction. Additionally, viscous losses will be neglected and hence $D$ will be set as 0 in any direction of the flow. Taking these considerations into account, a very easy expression for the calculation of Forchheimer coefficients can be derived:

$$F = \frac{2\Delta p}{\rho u_{\text{sim}}^2 L_{\text{mesh}}} \tag{2.18}$$

Observing this expression, it is obvious that a good estimation of pressure drop through the different elements is needed to calculate porosity coefficients. The general process is shown in Figure 2.12.

According to this diagram, the process starts with mass flow rate data from the different sections and elements of interest. These will be used, together with the corresponding geometry parameters from the real model (mainly the cross-section area of each zone). Once velocities are known, and using again some geometry parameters, pressure drops can be estimated using the corresponding correlations for the different layers inside a general FA and for the SR and CR systems. With these pressure drop values, the calculation of porosity coefficients is immediate.

All the steps are extensively explained in the following sections.

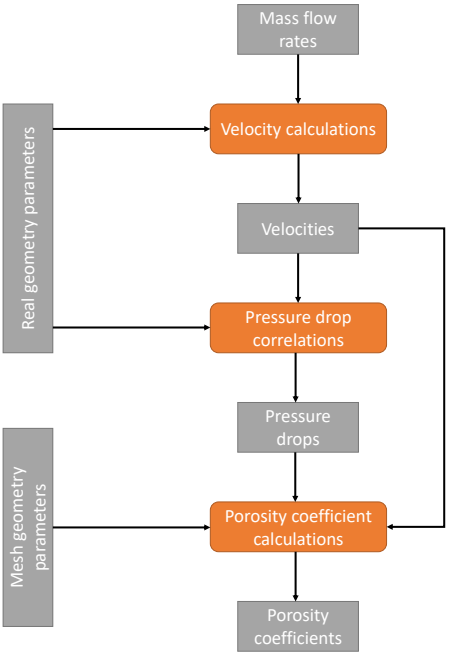



Figure 2.12. Porosity coefficients calculation diagram.

### 2.4.1 Velocity values calculations

The first step to calculate porosity coefficients is the estimation of suitable values for flow velocity in the zones of interest.

As part of the LEADER project, and based on the neutronic results of the spatial power distribution in the ALFRED core, a 1D thermohydraulic analysis was performed in order to assess the entire FA design and provide feedbacks.[26] One of the results is the division of the FA in different so-called *cooling groups*, which are depicted in Figure 2.13, and a first approximation of the mass flow rate across those groups, as reflected in Table 2.4. This division is the result of an attempt to develop a gagging scheme (obtained through the regulation of the gagging device from Figure 2.4) for the core in order to render the coolant temperature radial distribution at the core

Table 2.4. Parameters of the cooling groups for the ALFRED core. Source: LEADER project.[26]

| Cooling group | Power [MW] | $\dot{m}_{\text{average}}$ per element [kg/s] | $\dot{m}_{\text{total}}$ per cooling group [kg/s] |
|---|---|---|---|
| Group I | 294 | 172.3 | 14990 |
| Group II | | 145.2 | 3484 |
| Group III | | 117.5 | 4231 |
| Group IV | | 93.4 | 2241 |
| Control and safety rods | 1.7 | 16.3 | 261 |
| Dummy elements | 3.1 | 1.3 | 143 |
| Bypass | 1.2 | 110 | 110 |
| Sum | 300 | | 25460 |

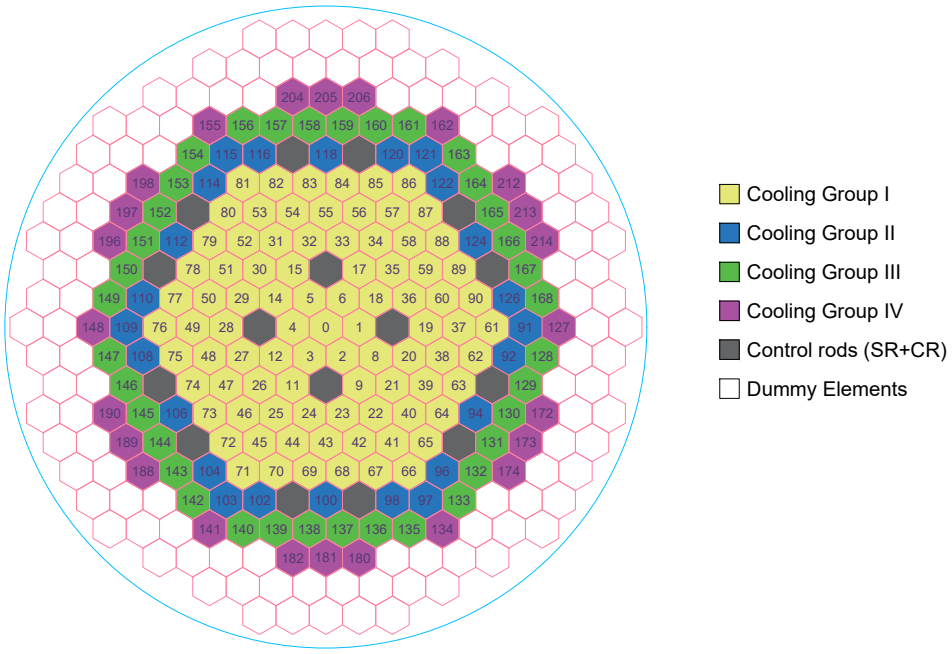



Figure 2.13. Cooling groups for the ALFRED core. Source: LEADER project.[26]

outlet as flat as possible.[15]

Using the information contained in Table 2.4, a first approximation of the coolant fluid velocities can be calculated for the different layers considered inside a general FA in each of the cooling groups, as well as in SR and CR systems, bypass and global inlet to the domain. These velocity values will be required when calculating pressure drop and, hence, porosity coefficients, as explained before. Calculations are performed knowing that $\dot{m} = \rho A v$, where $A$ values are taken from Table 2.3. Results for the different cooling groups are available in Table 2.5, while results for CR and SR systems and global inlet to the domain are collected in Table 2.6.

Table 2.5. Characteristic average velocities for the different cooling groups and DEs in each of the regions composing a general FA.

| Layer | $v$ [m/s] | | | | |
|---|---|---|---|---|---|
| | CG-I | CG-II | CG-III | CG-IV | DE |
| Inlet orifices | 0.648 | 0.546 | 0.442 | 0.351 | 0.005 |
| Outlet funnel | 0.759 | 0.639 | 0.517 | 0.411 | 0.006 |
| Empty tube | 0.759 | 0.639 | 0.517 | 0.411 | 0.006 |
| Pin bundle | 1.544 | 1.301 | 1.053 | 0.837 | 0.012 |
| Grid spacer | 1.796 | 1.513 | 1.225 | 0.973 | 0.014 |
| Support grid | 1.796 | 1.513 | 1.225 | 0.973 | 0.014 |

Table 2.6. Characteristic velocities for CR and SR systems and global inlet to the domain.

| Element | $v$ [m/s] |
|---|---|
| CR system | 0.258 |
| SR system | 0.196 |
| Global inlet to the domain | 0.321 |

### 2.4.2 Pressure drop estimations

One of the key variables when studying the thermohydraulic performance, not just in the specific case of a nuclear reactor, but in any fluid system in general, is the pressure drop produced through it. For most nuclear reactors, knowing the axial coolant pressure drop through the core is vital for reactor design and performance calculations.

As already known from Section 2.2.1, in ALFRED reactor core different assemblies can be found: fuel assemblies, safety rods, control rods and dummy elements; each of them having different geometric characteristics and, hence, different pressure drop values along their vertical axis. In the following subsections, an analysis of the pressure drop correlations for the different zones in the core will be presented.

#### Pressure drop correlations in a fuel assembly

The methodology used to calculate the pressure drop in a general fuel assembly is based on the one described by Schikorr *et al.* (2010).[44] Following their reasoning, the total pressure drop through a fuel assembly can be decomposed as follows:

$$\Delta p_{\mathrm{FA}} = \Delta p_{\mathrm{orf}} + \Delta p_{\mathrm{empty}} + \Delta p_{\mathrm{sg}} + \Delta p_{\mathrm{tube}} + \Delta p_{\mathrm{gss}} + \Delta p_{\mathrm{fun}} \tag{2.19}$$

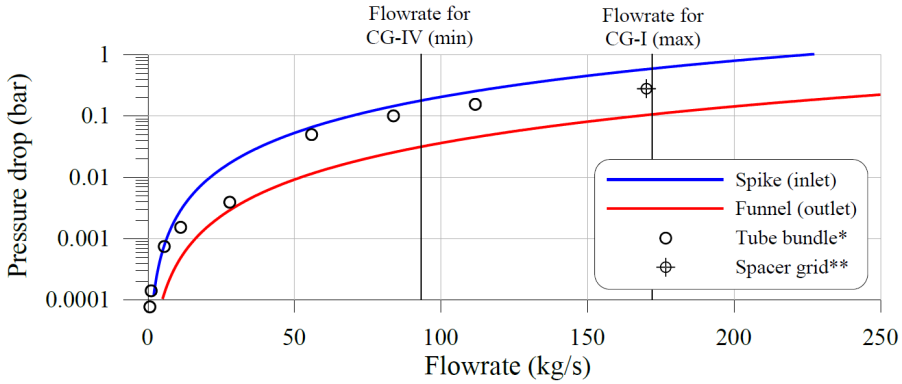
Where, according to the geometry described in Section 2.2.1, *orf*, *empty*, *sg*, *tube*, *gss* and *fun* refer to the inlet orifices, empty region, support grid, tube bundle, grid spacers and funnel (fuel assembly outlet) pressure losses respectively. Each of these components can be evaluated separately in order to obtain a proper pressure loss coefficient or friction factor that could be used later in the porous media approach. In order to simplify their analysis, these components will be divided into local pressure drops, distributed pressure drops, and pressure drop in grid spacers.

**Local pressure drops** Pressure losses across partial flow obstructions as inlet orifices and funnel can be evaluated as:

$$\Delta p_{\mathrm{i}} = K_{\mathrm{i}} \frac{1}{2} \rho v^2 \tag{2.20}$$

Where $i$ represents the different regions mentioned before and $K_{\mathrm{i}}$ is the pressure loss coefficient in the corresponding region. As it seems obvious, $\rho$ is the density of the coolant [kg/m$^3$] and $v$ is its velocity [m/s]. Pressure loss coefficients are usually determined experimentally and there are different tables of values that can be consulted, as those from Idel'Chik (2008).[20] Additionally, a specific CFD analysis was performed on behalf of the LEADER project[26] to evaluate the pressure losses at the inlet and outlet of the fuel assembly at different flow rates. As a result, dependencies

of the pressure losses at the inlet and outlet on the flow rate were derived and are depicted in Figure 2.14.




* pressure drop is recalculated for the whole length of the tube bundle (1330 mm)
** recalculated for 4 spacer grids

Figure 2.14. Pressure drops at the spacer grids, FA inlet and outlet as well as rod bundle as a function of mass flow rate as calculated by the CFD codes. Source: LEADER project.[26]

Using the expression from Equation (2.20) and velocity values from Table 2.5, a first approximation for the pressure loss coefficients can be reached. These primary results are resumed in Table 2.7. Values for geometry parameters and fluid properties are described in Sections 2.2.1 and 2.2.2 respectively.

Table 2.7. Pressure loss coefficients for inlet orifices and funnel for a general FA.

| | $\Delta p$ [bar] | $K_i$ |
|---|---|---|
| **Inlet orifices** | 0.17-0.58 | 26.251-26.318 |
| **Outlet funnel** | 0.03-0.10 | 3.376-3.307 |

Although grid spacers and the support grid could also be considered as a local pressure drop caused by a partial obstruction of the grid, special considerations about their correlations must be made and will be discussed later.

**Distributed pressure drops** They constitute friction losses in the empty or tube bundle regions, and can be calculated as:

$$\Delta p_{\mathrm{i}} = f_{\mathrm{fric,i}} \left( \frac{L_{\mathrm{i}}}{D_{h,\mathrm{i}}} \right) \frac{1}{2} \rho v^2 \tag{2.21}$$

Where $i$ refers to *empty* or *tube*, $L$ is the region length [m], $D_h$ is the equivalent hydraulic diameter of the flow section [m] and $f_{\mathrm{fric}}$ is the flow friction factor.

Different correlations for friction factor can be used. The most simple one was proposed by Blasius (1913)[3] for a smooth circular tube and reads as:

$$f_{\mathrm{fric}} = \frac{0.316}{\mathrm{Re}^{0.25}} \tag{2.22}$$

Where Re is the Reynolds number based on the hydraulic diameter of the flow channel, which means:

$$\mathrm{Re} = \frac{\rho v D_h}{\mu} \tag{2.23}$$

Where $D_h$, the hydraulic diameter, is commonly calculated as $D_h = 4A/P$, four times the cross-sectional area of the flow divided by the wetted perimeter of the cross-section. That general expression will be used for the hexagonal empty tube obtaining $D_h = 15.80$ cm. On the other hand, for the rod bundle hydraulic diameter will be considered as the one of a triangular lattice:

$$D_h = d\left[\frac{2\sqrt{3}}{\pi}\left(\frac{p}{d}\right)^2 - 1\right] \tag{2.24}$$

Where $d$ refers to the rod diameter and $p$ corresponds to the pitch (distance between two rods). In this case $D_h = 0.97$ cm.

Considering this method, the results from calculations are expressed in Table 2.8. As before, values for geometry parameters and fluid properties are described in Sections 2.2.1 and 2.2.2 respectively. Velocity range values used to calculate Re number are the ones from Table 2.5.

Table 2.8. Distributed pressure losses in the empty tube and rod bundle regions for a general FA.

| | | Re | $f_{\mathrm{fric}}$ | $\Delta p$ [bar] |
|---|---|---|---|---|
| **Empty tube** | CG-I | 631,001 | 0.0112 | 0.0015 |
| | CG-II | 531,755 | 0.0117 | 0.0011 |
| | CG-III | 430,311 | 0.0123 | 0.0008 |
| | CG-IV | 342,051 | 0.0131 | 0.0005 |
| **Pin bundle** | CG-I | 78,623 | 0.0189 | 0.3494 |
| | CG-II | 66,257 | 0.0197 | 0.2590 |
| | CG-III | 53,617 | 0.0208 | 0.1788 |
| | CG-IV | 42,620 | 0.0220 | 0.1196 |

An improved methodology was developed by Rehme (1973a)[39] which allows the prediction of friction factors in non-circular channels, as the ones present in this case: hexagonal empty channels and rod bundles organized in hexagonal arrays. Anyway, for the sake of simplicity and the wide range of flow conditions applicability, Blasius formula from Equation (2.22) will be used for both cases, empty and rod bundle, where Re value will be the one changing due to the different hydraulic diameter.

**Pressure drop in grid spacers and support grid** When working with fast breeder reactors as ALFRED, two basically different configurations for the fuel assembly can be found: (i) wire-wrapped pin bundles, and (ii) fuel bundle with grid spacers. An exhaustive review of the existing correlations for the prediction of pressure drop for the first configuration has been done by Chen *et. al* (2014).[6] But ALFRED uses the second configuration, where one of the most accepted correlations is the one

proposed by Rehme (1973b).[38] According to this, the pressure loss due to a grid spacer is calculated as:

$$\Delta p_{\text{gs}} = C_v \epsilon^2 \frac{1}{2} \rho v^2 \tag{2.25}$$

Where $C_v$ is a modified drag coefficient and $\epsilon$ is the blockage factor of the grid spacer (expressed as the ratio of areas: $A_{\text{grid spacer}}/A_{\text{flow}}$). It is important to note here that this correlation refers to the pressure drop due to just one grid spacer, and so its index is *gs* instead of *gss*. The support grid will be considered for pressure drop calculations as an additional grid spacer.

From experimental data, Cigarini and Dalle Donne (1988)[8] derived a correlation for the modified drag coefficient as function of Re:

$$C_{v,\text{norm}} = 3.5 + \frac{73.14}{\text{Re}^{0.264}} + \frac{2.79 \times 10^{10}}{\text{Re}^{2.79}} \tag{2.26}$$

With a maximum value to be written as:

$$C_{v,\text{max}} = \frac{2}{\epsilon^2} \tag{2.27}$$

So, at the end, the modified drag coefficient is written as:

$$C_v = \min\left[C_{v,\text{norm}}, C_{v,\text{max}}\right] \tag{2.28}$$

So, finally, using the method explained above and values for geometry parameters and fluid properties described in Sections 2.2.1 and 2.2.2 respectively, the results for pressure drop in grid spacers and support grid are displayed in Table 2.9. Here, $\Delta p_{\text{gs}}$ represents the pressure drop across one single grid spacer. Thus, $\Delta p_{\text{gss}} = N_{\text{spacers}} \times \Delta p_{\text{gs}}$ represents the total pressure drop across all the grid spacers inside a general FA, where $N_{\text{spacers}}$ is the number of grid spacers (3 plus the support grid, 4 in total). The result for the grid spacer hydraulic diameter calculation is $D_h = 1.386$ cm. Although geometrical obstruction is 0.14 (see Section 2.2.1), some examples of $\epsilon$ ranging from 0.15 to 0.44 depending on the specific design can be found in Rehme (1973b).[38] For this specific case, blockage factor seems to be higher than just the geometric obstruction[44] and, subsequently, is set to $\epsilon = 0.32$. This value offers pressure drop results consistent with primary simulations from Mikityuk *et al.* (2013).[26]

Table 2.9. Pressure drop in grid spacers and support grid.

| | | Re | $C_v$ | $\Delta p$ [bar] |
|---|---|---|---|---|
| **Grid spacer** | CG-I | 112,651 | 6.893 | 0.0884 |
| | CG-II | 94,933 | 7.049 | 0.0642 |
| | CG-III | 76,822 | 7.254 | 0.0433 |
| | CG-IV | 61,066 | 7.489 | 0.0282 |

It is important to mention that other correlations for grid spacers drag coefficients can be found in the literature, as the ones proposed by Vog *et al.* (1971),[51] Savatteri *et al.* (1986),[43] Cevolani (1995),[5] Epiney *et al.* (2010)[12] or Pacio *et al.* (2014).[33] A discussion among the first pressure drop correlations across grid spacers can be

found in Chenu *et al.* (2011),[7] where the study in sodium flow has shown that the Savatteri, Vog and Epiney correlations acceptably reproduced the test data. Another comparison between the two last correlations is carried out by Batta and Class (2017),[2] where CFD analysis is used to study the impact of different grid spacer geometries on pressure drop and then compared to values obtained from the different correlations. However, after all this analysis and due to its simplicity and extensive use, Cigarini and Dalle Donne correlation will be used in this thesis work.

### Comparison to previous results and conclusions

Results for maximum (corresponding to CG-I) and minimum (corresponding to CG-IV) pressure drops in the different sections from the correlations above can be compared to the results obtained in an extensive CFD analysis performed by Mikityuk *et al.* (2013)[26] using several codes as part of the LEADER project. This comparison is shown in Table 2.10 together with the corresponding absolute and relative error.

Table 2.10. Comparison of pressure drops [bar] at the different zones of a FA as calculated by the CFD codes of Mikityuk *et al.* (2013)[26] and as calculated by the correlations used in this thesis.

| Layer | CFD simulation | | Correlations | | Error | | % Error | |
|---|---|---|---|---|---|---|---|---|
| | CG-I | CG-IV | CG-I | CG-IV | CG-I | CG-IV | CG-I | CG-IV |
| Grid sp. (4) | 0.090 | 0.026 | 0.088 | 0.028 | -0.002 | 0.002 | -1.79% | 8.54% |
| In. orifices | 0.390 | 0.114 | 0.579 | 0.170 | 0.189 | 0.056 | 48.53% | 49.31% |
| Out. funnel | 0.104 | 0.031 | 0.101 | 0.030 | -0.003 | -0.001 | -2.84% | -4.22% |
| Pin bundle | 0.450 | 0.149 | 0.351 | 0.120 | -0.099 | -0.029 | -22.03% | -19.36% |
| Total | 1.304 | 0.398 | 1.385 | 0.433 | 0.081 | 0.035 | 6.19% | 8.78% |

From here it can be noticed that total relative error between the results from Mikityuk CFD simulations and correlations used in this thesis is below 10%, which is considered more than an acceptable deviation. When focusing on each layer relative error, it is observed that the greatest difference is for inlet orifices, where pressure drop values were taken from a specific CFD simulation. On the other hand, pin bundle shows a pressure drop underestimation, which partially compensates error from inlet orifices. Anyway, although these results show that the application of these correlations offer consistent pressure drop estimations, it can be considered a rather trivial matter. Eventually, pressure drop correlations can be improved and even substituted by better experimental data. So, at the end, values from correlations will be considered valid and used for porosity coefficient calculations.

Pressure drop values in the different layers for all the cooling groups calculated by the correlations above are summarized in Table 2.11. These will be part of the values used for the porosity coefficient calculations.

This table shows what could be anticipated from the beginning, depending on the cooling group, and conditioned by the different mass flow rates passing through them, the pressure drop across a FA varies according to the gagging scheme already mentioned before. But, in order to guarantee the corresponding mass flow rate for each FA, the total pressure drop in the whole core must be more or less the same and equal to the maximum one 1.385 [bar].

Table 2.11. Summary on pressure drop values in the different layers for all the cooling groups calculated by correlations.

| Layer | $\Delta p_i$ [bar] | | | |
|---|---|---|---|---|
| | CG-I | CG-II | CG-III | CG-IV |
| Grid spacer (4) | 0.088 | 0.064 | 0.043 | 0.028 |
| Inlet orifices | 0.579 | 0.411 | 0.269 | 0.170 |
| Outlet funnel | 0.101 | 0.072 | 0.047 | 0.030 |
| Pin bundle | 0.349 | 0.259 | 0.179 | 0.120 |
| Empty tube | 0.001 | 0.001 | 0.001 | 0.001 |
| Total | 1.385 | 1.000 | 0.669 | 0.433 |

Knowing that the total pressure drop across any cooling group must be adjusted to 1.385 [bar] (pressure drop in CG-I), it is trivial finding that the additional inlet pressure drops produced by the gagging device for CG-II, CG-III and CG-IV must be 0.385 [bar], 0.716 [bar] and 0.952 [bar] respectively. These values are added to inlet orifices pressure drops and thus the following results are obtained $\Delta p_{\text{orf,CG-I}} = 0.579$ [bar], $\Delta p_{\text{orf,CG-II}} = 0.796$ [bar], $\Delta p_{\text{orf,CG-III}} = 0.985$ [bar] and $\Delta p_{\text{orf,CG-IV}} = 1.122$ [bar]. Porosity coefficients for inlet orifices layer will be hence calculated with these values.

**Pressure drop in dummy elements, control rods and safety rods**

The previous idea in which each FA must be subjected to the same total pressure drop is also applicable to DEs and CR and SR systems.

**Dummy elements** According to Section 2.2.1, construction characteristics of dummy elements are exactly the same as those from a general FA. However, power generated in them is negligible due to the lack of fuel inside their pin bundles and they mainly work as a neutron reflector. Additionally, mass flow rate in these elements is very low and main pressure drop is only produced in the inlet when adjusting it. For those reasons, the interest in studying the pressure drop distribution inside DEs is low and hence a constant pressure drop along its length will be considered. This reasoning will be used when calculating porosity coefficients in the following section, where the only important information is that $\Delta p_{\text{DE}} = 1.385$ [bar] and its corresponding mass flow rate from Table 2.4.

**Control and safety rods** A similar reasoning is followed for CR and SR systems. Although part of their design is available from Figures 2.7a, 2.7b and 2.8; CRs and SRs geometric full description is unknown and, therefore, it is difficult to deduce a pressure drop distribution in different layers inside of them. Additionally, the interest in studying flow through them is completely minor. Thus, same procedure as before will be taken: assuming a constant pressure drop equal to maximum total one and the mass flow rate specified from Table 2.4. Hence, a uniform porosity coefficient will be assigned for the whole length of these elements.

### 2.4.3 Porosity coefficients and implementation in OpenFOAM

Once all pressure drops for the different layers in each cooling group and for the rest of the core elements, porosity coefficients calculations can be performed. For that exact purpose, Equation 2.18 will be used, where values for $\Delta p$ and $L_{\text{mesh}}$ will be respectively extracted from Tables 2.11 (considering modifications mentioned for inlet orifices and total pressure drop for DEs, CRs and SRs) and 2.3. On the other hand, simulated flow velocity $u_{\text{sim}}$ needs to be calculated. Porous media does not account for flow area variations from layer to layer, hence velocity is constant along the whole element length. It can be calculated thanks to the mass flow rate knowing that cross-section area is constant $A = 253.23$ [cm$^2$]. Simulated flow velocity values are collected in Table 2.12.

Table 2.12. Simulated flow velocity values in the different groups composing the nuclear core.

| Group | $u_{\text{sim}}$ [m/s] |
|---|---|
| CG-I | 0.648 |
| CG-II | 0.546 |
| CG-III | 0.442 |
| CG-IV | 0.351 |
| CRs and SRs | 0.064 |
| DE | 0.005 |

With all the required data, porosity coefficients calculations can be performed. All cooling groups will have the same porosity coefficients for each layer except for the inlet orifices, where, as already explained, pressure drop must be adjusted in order to obtain the required mass flow rate. General results for porosity coefficients in the different layers (except for the inlet orifices) and elements in the core are gathered in Table 2.13. On the other hand, inlet orifices porosity coefficients can be found in Table 2.14.

Table 2.13. Porous coefficients for each of the layers and elements forming the core mesh.

| Layer or element | $F_z$ [1/m] |
|---|---|
| Empty tube | 1.809 |
| Pin bundle | 12.467 |
| Grid spacer | 208.645 |
| Outlet funnel | 114.625 |
| CR and SR system | 4,208.343 |
| Dummy Elements | 704,618.952 |

Once porosity coefficients have been correctly calculated, the next step is implementing this porous media approach in OpenFOAM. The addition of the source term $S$ mentioned at the beginning of this section can be done in two ways: using specific porous solvers provided in the latest versions of OpenFOAM, or by addition of explicit porous zones by the *fvOptions* file. The first way allows an implicit treatment for

Table 2.14. Porous coefficients for the inlet orifices layer from the different cooling groups.

| | $F_z$ [1/m] | | | |
|---|---|---|---|---|
| | CG-I | CG-II | CG-III | CG-IV |
| Inlet orifices | 657.107 | 1,271.649 | 2,402.995 | 4,331.425 |

porosity, which is supposed to be more robust if the resistances are large, heavily anisotropic or not aligned with the global coordinates.[17] However, only two specific solvers are provided: incompressible solver *porousSimpleFoam* and compressible one *rhoPorousSimpleFoam*. In both of them, the source term $S$ is added to the momentum equation by calling a function *addResistance*. Both solvers are used for steady-state scenarios where energy is not simulated, so that further modifications in those solvers should be made if transient simulations accounting for energy were to be performed. On the other hand, using the second way by the *fvOptions* file, the function *addResistance* is explicitly added to the momentum equation no matter the solver used, so that, in theory, porosity can be simulated in any case. Nevertheless, only explicit treatment can be performed this way, which can affect stability and convergence of the solution, hence lower under-relaxation factors are usually needed and simulations take longer.

Further information about the specific solvers can be found in *$FOAM_SOLVERS* and even some tutorials may be found in *$FOAM_TUTORIALS* depending on the OpenFOAM version. Additionally, information about the porous media model might be found in *$FOAM_SRC/finiteVolume/cfdTools/general/porosityModel*.

Anyway, the second way to implement porous media in OpenFOAM will be used in this thesis work, where porous regions with explicit porosity sources have to be defined in the *fvOptions* file as follows:

```
porosity
{
    type            explicitPorositySource;
    active          yes;

    explicitPorositySourceCoeffs
    {
        type            DarcyForchheimer;
        selectionMode   cellZone;
        cellZone        porous;

        DarcyForchheimerCoeffs
        {
            d   d [0 -2 0 0 0 0 0] ($dx $dy $dz);
            f   f [0 -1 0 0 0 0 0] ($fx $fy $fz);

            coordinateSystem
            {
                type    cartesian;
                origin  (0 0 0);
```

```
21                  coordinateRotation
22                  {
23                      type    axesRotation;
24                      e1   (1 0 0);
25                      e2   (0 1 0);
26                  }
27              }
28          }
29      }
30  }
```

Here, it can be observed that, in this case, Darcy-Forchheimer law is applied in the cell zone called *porous* where the coordinate system has been kept as the mesh general one. As already anticipated, Darcy coefficients will be set to 0, while z-direction Forchheimer coefficient *fz* has been previously calculated for the different regions. On the other hand, *fx* and *fy* values will be set to a relatively high number, so that one dimension flow through those regions is imposed.

Creating a *fvOptions* file for the actual mesh, where there are 4158 cell zones, which means writing 4158 different entries as the previous one explained, is not a trivial matter. For that reason, a python script was used for the writing of such file and can be found in Appendix B.

## 2.5 Heat transfer in the core

Apart from the porous media approach for pressure drop simulation, the other important physical process to model is the heat exchange produced between the pin bundle filled with fuel and the coolant liquid lead. In that way, the coolant temperature difference produced by its way through the core can be later used for the generation of steam. Following that idea, a good prediction of the average coolant temperature can be useful for the study of steam generation and, hence, power output. Additionally, an uniform temperature distribution is sought at the core outlet in order to improve efficiency and avoid mechanical stresses. Finally, the analysis of temperature distribution inside the core is important from a safety point of view, analysing the appearance of possible hotspots if obstructions are produced. In that sense, coolant flow temperature must be kept low enough to avoid liquid lead degradation and clad failure.

The core power distribution is obtained from the neutronic characterization of the core. Although it is strongly influenced by thermohydraulics (coolant temperature and neutronics are coupled), neutronic characterization of the core has been already studied as part of the LEADER project by means of two independent codes: MCNPX and ERANOS. Further information about these codes can be respectively found in Pelowitz *et al.* (2011)[34] and Rimpault (2002).[40] The complete procedure for such a neutronic characterization will not be explained in this thesis, but for a discussion on different approaches and results see Grasso *et al.* (2014).[15]

The only useful result from this neutronic characterization for this thesis work will be the power generation in the FAs that is shown in Figure 2.15, calculated at BOC (in green) and EOC (in red). The average power generation per FA is 1.75 [$MW_{th}$].

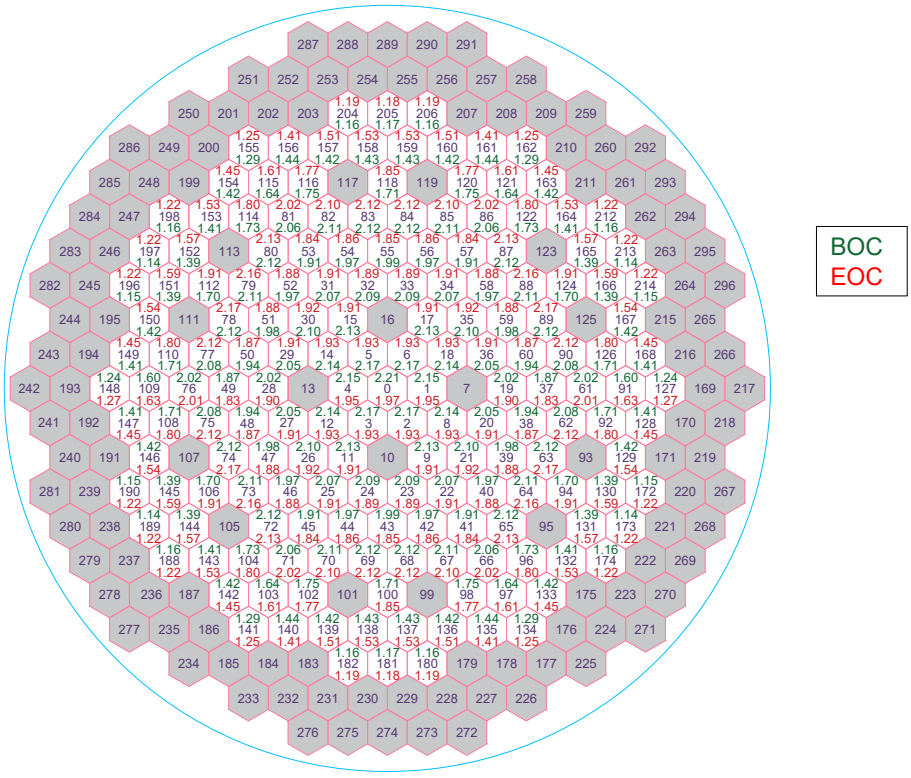



Figure 2.15. Power generation [$MW_{th}$] in the different FAs at BOC and EOC.

It is observed that the maximum value corresponds to the centre FA (number 0) at BOC (2.21 [$MW_{th}$]) and to outer elements (number 63, 74, 78 and 89) at EOC (2.17 [$MW_{th}$]). Whether this FAs correspond also to the hottest ones or not is something that should be analysed by simulations.

Additionally, the power is deposited not only in the fuel, but also in other materials due to the gamma neutron emission in the fission process. This has also been studied by the neutronic simulations and the results are summarized in Table 2.15.

Table 2.15. Power distribution according to the different zones. Source: LEADER project.

| Zone | % | $MW_{th}$ |
|---|---|---|
| MOX fuel | 93.06% | 279.17 |
| Dummy elements | 0.95% | 2.84 |
| Control Rods | 0.57% | 1.71 |
| Safety Rods | 0.17% | 0.50 |
| Pb in bypass zone | 0.09% | 0.26 |

In OpenFOAM, thermal power generation will be implemented as volumetric heat sources for the active region, which in this case is composed by layers *act1*, *act2* and *gs2* (revisit Figure 2.9). This can be done by creating a new field for the heat source and adding it as a new term in the energy equation. For this purpose, solver files have to be modified and recompiled.

In this case, an example for the modification of *buoyantSimpleFoam* will be included.

This solver can be found in *$FOAM_SOLVERS/heatTransfer/buoyantSimpleFoam* and can be copied to a personal folder as *my_bouyantSimpleFoam* in order to change those copied files instead the original solver. Once there, a new field, *volPow*, representing the volumetric thermal power generation, must be included to the beginning of *createFields.H* file as follows:

```
Info<< "Reading field volPow\n" << endl;
volScalarField volPow
(
     IOobject
     (
          "volPow",
          runTime.timeName(),
          mesh,
          IOobject::NO_READ,
          IOobject::NO_WRITE
     ),
     mesh
);
```

Then, *EEqn.H* file can be modified to add the term *volPow* to the second member of the energy equation:

```
{
    volScalarField& he = thermo.he();

    fvScalarMatrix EEqn
    (
        fvm::div(phi, he)
      + (
            he.name() == "e"
          ? fvc::div(phi, volScalarField("Ekp", 0.5*magSqr(U)
             + p/rho))
          : fvc::div(phi, volScalarField("K", 0.5*magSqr(U)))
        )
      - fvm::laplacian(turbulence->alphaEff(), he)
     ==
        rho*(U&g)
      + radiation->Sh(thermo, he)
      + fvOptions(rho, he)
      + volPow
    );

    EEqn.relax();

    fvOptions.constrain(EEqn);

    EEqn.solve();

```

```
26      fvOptions.correct(he);
27
28      thermo.correct();
29      radiation->correct();
30  }
```

Now, the solver file name has to be changed from *buoyantSimpleFoam.C* to *my_buoyant SimpleFoam.C* and *files* file inside *Make* folder must be modified as follows:

```
1  my_buoyantSimpleFoam.C
2
3  EXE = $(FOAM_USER_APPBIN)/my_buoyantSimpleFoam
```

Finally, recompilation can be made when placed inside the solver folder by using the following utility:

```
1  wmake
```

This way, the new solver can be called when using OpenFOAM by its name, *my_buoyantSimpleFoam*.

It is important to highlight that, as a new field has been defined, initial and boundary conditions have to be set for this new field. This can be made by using *setFields* utility just before running the simulation. For this purpose, a *setFieldsDict* is needed where a value for *volPow* can be assigned to the different cell regions defined in the mesh geometry (see Section 2.3). The general aspect of such a dictionary would be something like:

```
1  regions
2  (
3      zoneToCell
4      {
5          name "act1_0";
6          fieldValues
7          (
8              volScalarFieldValue volPow 145451482
9          );
10     }
11 );
```

Where *volPow* values are expressed in [W/m$^3$] and can be calculated thanks to data from Figure 2.15. As for the porous media approach from previous section, writing a script containing the structure before for each of the active regions (*act1*, *act2*, and *gs2* in each FA) is not a trivial matter and a python program, which can be observed in Appendix C, has been used to ease this task.

Additionally, a *volPow* file inside *0* folder of each particular case where power generation wants to be used is required for setting initial and boundary conditions for that field. As field values are set thanks to the *setFields* utility, here *volPow* has to be set to 0 everywhere as follows:

```
dimensions      [1 -1 -3 0 0 0 0];

internalField   uniform 0;

boundaryField
{
        inlet
        {
                type    fixedValue;
                value   uniform 0;
        }
        outlet
        {
                type    fixedValue;
                value   uniform 0;
        }
        wall
        {
                type    fixedValue;
                value   uniform 0;
        }
}
```

When using *setFields* utility, the internal field will be automatically changed from uniform 0 to the desired distribution.

Once all these steps have been accomplished, heat sources can be simulated and energy exchange can be studied inside the nuclear core as seen in Chapter 3.

# Chapter 3

# Results

In this chapter, main results from simulations will be presented. They are divided in different so-called cases, representing the different physical considerations facing the problem. The two first cases are considered as steady simulations of the core normal operation and have two main purposes: verifying the implementation of porous media and heat sources in OpenFOAM, as well as obtaining resulting variable fields that can be used as initial conditions for further transient simulations (i.e. flow obstructions).

Each of the cases will be divided in pre-processing, processing and post-processing. Proper results are contained in the latter one, but it seems important to explain the procedure followed and the inputs used for each of the simulations. Pre-processing and processing sections will be extensively explained for the first case and barely commented for the rest of the cases to avoid unnecessary repetitions.

## 3.1 Case 1: Adiabatic steady-state

As already mentioned, the first case considered will be the steady-state simulation of a porous media for the different regions inside the core without generation of thermal power. The main objective is to test some of the assumptions and methods related to porous media that were introduced in the previous chapter, as well as obtaining variable fields that can be used as initial conditions for further simulations. Refining and studying the scope of this approach is of great importance for future cases where thermal power generation will also be simulated.

### 3.1.1 Pre-processing

The first step when facing a simulation is finding some expected results which can be later compared to the post-process output and that can even help in setting some parameters for the simulation. All of these expected results were already addressed in Chapter 2, which, in this case, are velocity and pressure drop distributions inside the different elements composing the geometry. All those values are mainly collected in Tables 2.11 and 2.12.

Matters about geometry discretization, mesh generation, its conversion to OpenFOAM's format, scaling and quality check were all already commented in Section 2.3.

The solver used in this case is *simpleFoam*, which is described by OpenFOAM User

Guide (2019)[29] as a steady-state solver for incompressible, turbulent flow, using the SIMPLE algorithm. Coolant fluid is liquid lead and, hence, is considered incompressible. However, after some iterations, *pimpleFoam* transient solver will be also used to simulate for some seconds the core steady simulation and try to reach a solution as converged as possible.

Information about physical models is contained in *constant* directory and, for this specific solver, only two of them are required: a transport model in the file *transportProperties* and a turbulence model in the file *turbulenceProperties*. The first one is set to Newtonian model which assumes constant kinematic viscosity and the script is written as:

```
transportModel  Newtonian;

nu              [0 2 -1 0 0 0 0] 1.9e-07;
```

On the other hand, in *turbulenceProperties* file, the simulation type is set to Reynolds-Averaged Simulation (RAS) and $k-\epsilon$ model is chosen. The general aspect of the file is:

```
simulationType RAS;

RAS
{
        RASModel        kEpsilon;

        turbulence      on;

        printCoeffs     on;
}
```

The choice of this turbulence model over $k-\omega$ is mainly motivated by the advantages described in Section 2.1.2.

As already explained, explicit porosity treatment will be implemented thanks to the *fvOptions* framework, which allows to simulate various sources into an existing solver without modifying its code. Porosity coefficients are contained in the file *system/fvOptions* which is configured just as indicated in Section 2.4.3.

The only material property required is kinematic viscosity for the transport model and which value is set according to information from Section 2.2.2, specifically from Table 2.2. However, it is important to mention that, as an incompressible solver is being used, some quantities and fields will be divided by density, whose value does not need to be specified. That is the reason why kinematic viscosity is used instead of dynamic.

Initial and boundary conditions are required for the variable fields used in the simulation, which, in this case, are: velocity $U$, pressure $p$, turbulent kinetic energy $k$, turbulence dissipation rate $\epsilon$ and turbulent kinematic viscosity $\nu_t$. Each of them has its corresponding file in directory *0*.

Velocity field is initially set to 0 in any direction (it is a vectorial variable) for the internal field. Inlet is set according to Table 2.6 while outlet is set to zero

gradient (quantity is extrapolated to the patch form the nearest cell value). As already anticipated, vessel inner wall has not be modelled in the geometry, thus slipping wall condition has been adopted for the wall formed by the outer core elements. Its corresponding file is showed in the following script:

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
        inlet
        {
                type            fixedValue;
                value           uniform (0 0 0.321);
        }
        outlet
        {
                type            zeroGradient;
        }
        wall
        {
                type            slip;
        }
}
```

As an incompressible simulation is considered, pressure magnitude is divided by density, which can be checked by the dimensions [$m^2/s^2$]. This means that any pressure result has to be multiplied by density in order to know the actual pressure value (in [Pa]). Initial conditions are set to 0 anywhere inside the internal field, while the boundary conditions are a fixed value of 0 at the outlet (this will be taken as reference atmospheric pressure) and zero gradient condition both for inlet and wall. This script represents the file containing pressure field boundary and initial conditions.

```
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
        inlet
        {
                type            zeroGradient;
        }
        outlet
        {
                type            fixedValue;
                value           uniform 0;
        }
```

```
        wall
        {
                type            zeroGradient;
        }
}
```

Some calculations are required for $k$ and $\epsilon$ initial and boundary conditions. According to literature,[50] when no information is available, rough approximations for the inlet distributions can be obtained by the following simple formulas:

$$k = \frac{3}{2}(U_{ref}T_i)^2 \qquad \epsilon = C_\mu^{3/4}\frac{k^{3/2}}{0.07L} \tag{3.1}$$

Where $U_{ref}$ will be taken as the inlet velocity, $T_i$ corresponds to turbulence intensity for which a typical value of 5% will be assumed, $C_\mu = 0.09$ is a fixed constant from the model, and $L$ denotes a characteristic length which will be taken as the inlet core diameter. Results from these calculations will be used for initial internal field value and for inlet boundary condition. On the other hand, zero gradient condition will be set both for outlet and wall boundaries. No wall function is required as slipping wall condition has been chosen.

Files for $k$ and $\epsilon$ initial and boundary conditions can be found in the following two scripts.

```
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.0003862;

boundaryField
{
        inlet
        {
                type            fixedValue;
                value           uniform 0.0003862;
        }
        outlet
        {
                type            zeroGradient;
        }
        wall
        {
                type            zeroGradient;
        }
}
```

```
dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 5.3985e-06;

boundaryField
```

```
{
        inlet
        {
                type            fixedValue;
                value           uniform 5.3985e-06;
        }
        outlet
        {
                type            zeroGradient;
        }
        wall
        {
                type            zeroGradient;
        }
}
```

At last, turbulent kinetic viscosity $\nu_t$ boundary conditions are calculated by OpenFOAM according to the corresponding turbulence model.

```
dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
        inlet
        {
                type            calculated;
                value           uniform 0;
        }
        outlet
        {
                type            calculated;
                value           uniform 0;
        }
        wall
        {
                type            calculated;
                value           uniform 0;
        }
}
```

As an incompressible simulation is performed, according to Section 2.1.4, SIMPLE algorithm is needed. In order to implement an explicit solver required by the explicit treatment of porosity, the number of correctors needs to be set as *nUCorrectors = 0*. This configuration may induce some convergence and stability problems, so relatively low relaxation factors are recommended specially for the first iterations. Additionally, most of the times using the multigrid solver *GAMG* is the best choice for pressure and should converge fast. A basic *smoothSolver* is adopted for the velocity field. This

settings are contained in the file *system/fvSolution*, which script for this case is the following:

```
solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-08;
        reltol          0.05;
        smoother        GaussSeidel;
        nCellsInCoarsestLevel 20;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-06;
        relTol          0.1;
    }

    "(k|epsilon)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-07;
        relTol          0.1;
    }
}

SIMPLE
{
        nUCorrectors    0;
        nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
        fields
        {
                p               0.3;
        }
        equations
        {
                U               0.5;
                k               0.7;
```

```
47                      epsilon         0.7;
48              }
49      }
```

However, tighter tolerances can be set after the first iterations to get more accurate results. This is specially important for pressure equation, as it governs mass conservation and it is usually the expensive part of the whole iterative process.

When using *pimpleFoam* solver, some conditions for *Final* variables need to be adjusted too.

Discretization schemes are set in *fvSchemes* file inside the *system* folder which has the following structure:

```
1   ddtSchemes
2   {
3           default         steadyState;
4   }
5
6   gradSchemes
7   {
8           default         Gauss linear;
9   }
10
11  divSchemes
12  {
13          default         none;
14
15          div(phi,U)      bounded Gauss upwind;
16          div((nuEff*dev2(T(grad(U))))) Gauss linear;
17          div(phi,epsilon) bounded Gauss upwind;
18          div(phi,k)      bounded Gauss upwind;
19  }
20
21  laplacianSchemes
22  {
23          default         Gauss linear corrected;
24  }
25
26  interpolationSchemes
27  {
28          default         linear;
29  }
30
31  snGradSchemes
32  {
33          default         corrected;
34  }
```

In time discretization schemes, *steadyState* is used for steady state simulations like when using *simpleFoam*, but can be changed to *CrankNicolson* when using *pimpleFoam*.

For the rest, *Gauss linear* corresponds to a central difference scheme and is second order accurate unbounded (unless formulated otherwise), while *bounded Gauss upwind* is a first order bounded method.

### 3.1.2 Processing

The cluster from energy department is used to run the simulation in parallel on distributed processors so that computing times are significantly lower, permitting solving larger and more complex problems. This method is based on domain decomposition, in which the geometry and associated fields are broken into parts and allocated to separate processors for solution.

Different nodes are available for calculations inside this cluster, but the main characteristics of a general one are the following:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                16
On-line CPU(s) list:   0-15
Thread(s) per core:    1
Core(s) per socket:    8
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 45
Stepping:              7
CPU MHz:               2700.003
BogoMIPS:              5399.28
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              20480K
NUMA node0 CPU(s):     0,2,4,6,8,10,12,14
NUMA node1 CPU(s):     1,3,5,7,9,11,13,15
```

The most important value from this script is the number of CPUs (cores) available, which in this case is 16. This value limits the maximum number of pieces in which geometry can be decomposed and distributed among the different processors. However, it is not recommended to use all the available cores for the parallel running because some of them might be used for other tasks.

Three steps have to be performed in order to run OpenFOAM in parallel:

1. Decomposition of the domain using the *decomposePar* utility. This needs the *system/decomposeParDict* dictionary which structure is the following:

```
numberOfSubdomains 12;

```

```
3 method          scotch;
```

The method *scotch* seeks minimizing the number of processor boundaries and it is highly recommended because the only input required from the user is the number of subdomains/cores (12 in this case). Some other domain decomposition methods can be found depending on the OpenFOAM version: multiLevel, simple, manual, structured...

2. Distribution of the jobs among the available cores using the standard Message Passing Interface (MPI). This allows to run a copy of the solver in each of the domain parts decomposed. The command for such an action is the following:

```
1 > mpirun -np "NPROCS" "SOLVER" -parallel
```

Where *NPROCS* refers to the number of processors and must be equal to the number of subdomains defined before.

3. After running the solver, the decomposed domain needs to be reconstructed again, which is done by the *reconstruct* utility. Unlike before, this utility does not need a dictionary.

A critical point when running the simulation is deciding when the solution has converged and the process can be stopped. There are no universal metrics for judging convergence and residual definitions that are useful for a specific case might be misleading for another one. Therefore, it is advisable to judge convergence by monitoring residuals, some relevant quantities as well as mass and energy balances. In this case:

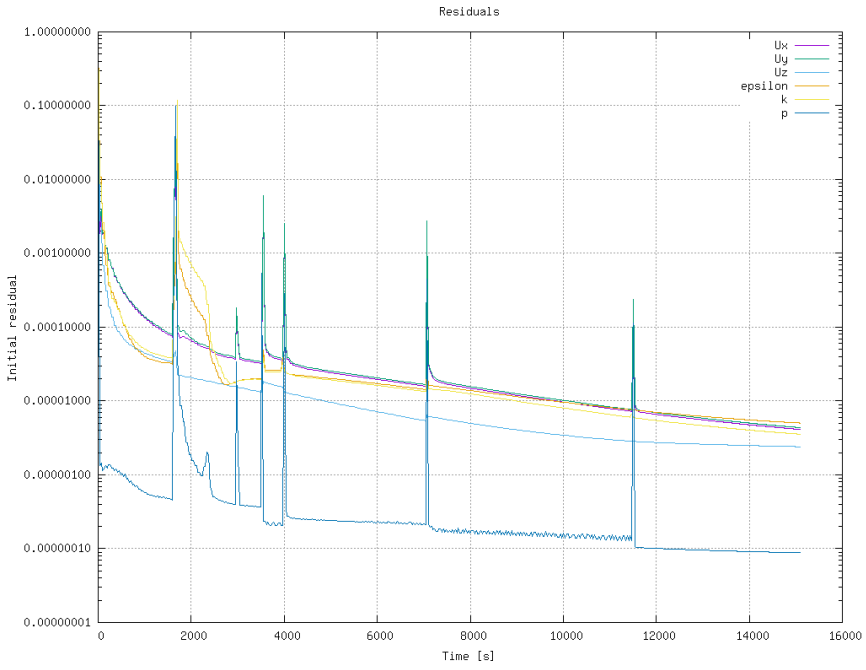



Figure 3.1. Example of residuals evolution in *simpleFoam* simulation.

1. Residuals are checked by saving the terminal output to a log file and then executing *pyFoamPlotWatcher.py* utility on a different terminal tab. An example of these residuals for the *simpleFoam* simulation is shown in Figure 3.1, where the peaks represent different adjustments for tolerances and under-relaxation factors from *fvSolution* file.

2. For this kind of cases, it is very common to monitor inlet pressure and outlet velocity area averaged values, because outlet pressure and inlet velocity are imposed by boundary conditions. The objective is checking the moment in which these values do not change any longer with an additional iteration. But, as a supposed incompressible fluid, outlet velocity convergence can be indirectly checked by overall mass balances as will be explained in the following point. On the other hand, for this specific case, it is very important to check the velocity convergence in each of the cooling groups defined before, as each of them is characterized by a particular flow velocity conditioned by pressure drop and which has a strong impact on heat exchange. This monitoring action will be carried out by defining the subdictionary *functionObjects* at the end of *controlDict* file as follows:

```
functions
{
    inPressure
    {
        type            surfaceFieldValue;
        libs            ("libfieldFunctionObjects.so");
        enabled         true;

        log             true;
        writeControl    writeTime;
        writeFields     true;

        surfaceFormat   none;
        regionType      patch;
        name            inlet;

        operation       areaAverage;

        fields
        (
        p
        );
    }

    volVelCGI
    {
        type            volFieldValue;
        libs            ("libfieldFunctionObjects.so");

        log             true;
```

```
31          writeControl    writeTime;
32          writeFields     true;
33
34          regionType      cellZone;
35          name            act1_0;
36          operation       volAverage;
37
38          fields
39          (
40          U
41          );
42      }
43
44      volVelCGII
45      {
46          type            volFieldValue;
47          libs            ("libfieldFunctionObjects.so");
48
49          log             true;
50          writeControl    writeTime;
51          writeFields     true;
52
53          regionType      cellZone;
54          name            act1_91;
55          operation       volAverage;
56
57          fields
58          (
59          U
60          );
61      }
62
63
64      volVelCGIII
65      {
66          type            volFieldValue;
67          libs            ("libfieldFunctionObjects.so");
68
69          log             true;
70          writeControl    writeTime;
71          writeFields     true;
72
73          regionType      cellZone;
74          name            act1_168;
75          operation       volAverage;
76
77          fields
78          (
79          U
80          );
```

```
81      }
82
83      volVelCGIV
84      {
85          type            volFieldValue;
86          libs            ("libfieldFunctionObjects.so");
87
88          log             true;
89          writeControl    writeTime;
90          writeFields     true;
91
92          regionType      cellZone;
93          name            act1_127;
94          operation       volAverage;
95
96          fields
97          (
98          U
99          );
100     }
101 };
```

Note that velocity values are active zone volumetric averages for elements 0, 91, 168 and 127, corresponding to cooling groups I, II, III and IV respectively.

Following this procedure, average values of pressure at the inlet and velocity at the different cooling groups are obtained for each of the iterations saved as results. These values are stored in the directory *postProcessing* inside the case and can be plotted during the simulation by *foamMonitor* utility for example or further analysed by any other software. Plots for the *pimpleFoam* simulation

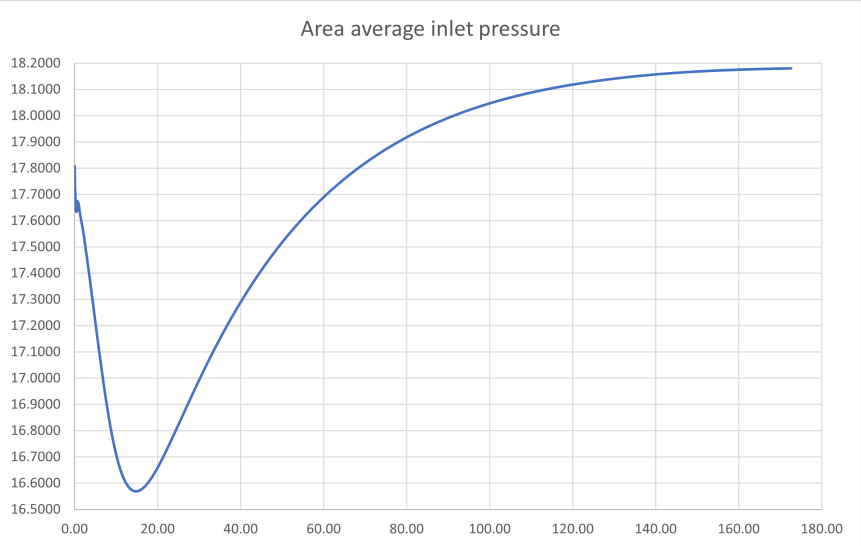



Figure 3.2. Example of area average inlet pressure [$m^2/s^2$] evolution during *pimpleFoam* simulation.

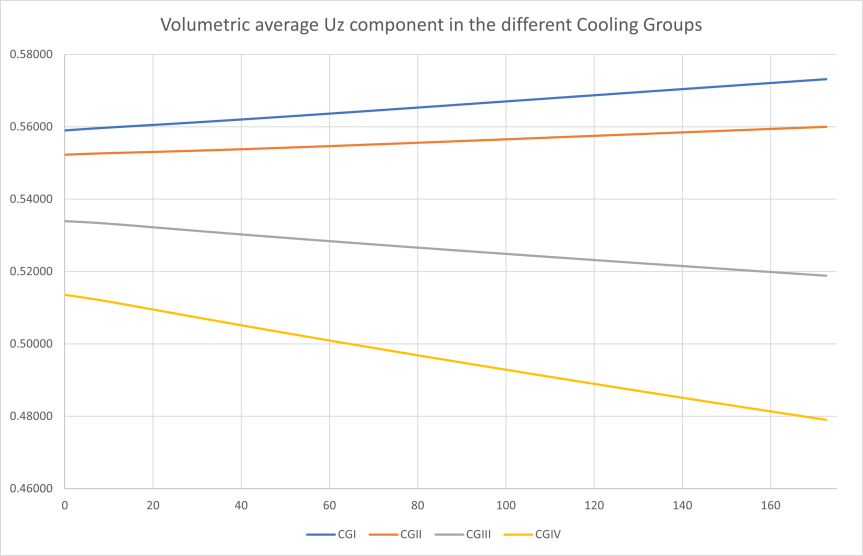



Figure 3.3. Example of volumetric average z-velocity component [m/s] evolution during *pimpleFoam* simulation in the different cooling groups.

are depicted in Figures 3.2 and 3.3.

3. Overall mass balances are performed by the solver in each iteration and are showed as time step continuity errors. An example plot obtained thanks to *pyFoam* utility is shown in Figure 3.4.

Anyway, it is difficult to judge when to stop the simulation, specially when variable changes are really slow, like in this case as observed from figures before. In this

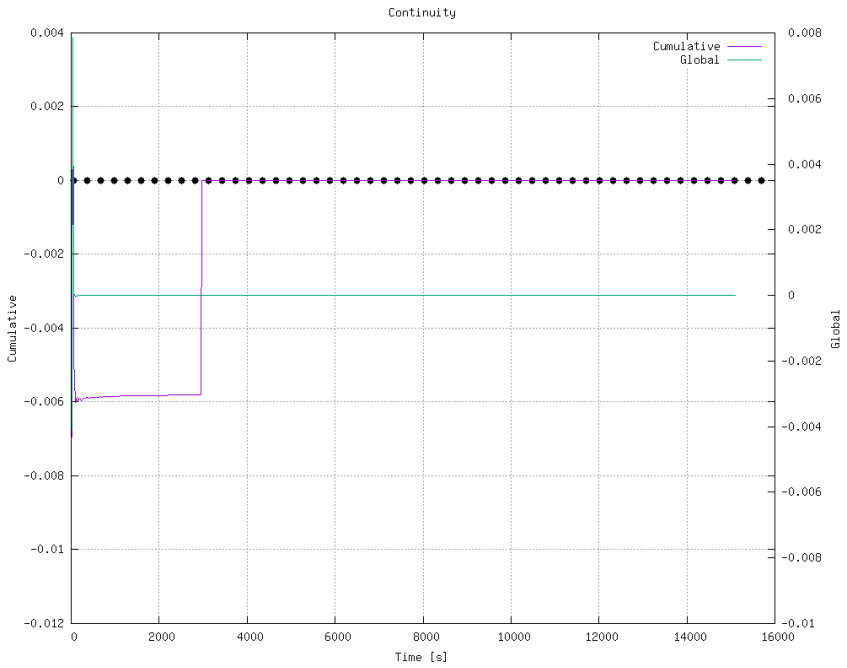



Figure 3.4. Example of continuity errors evolution during *simpleFoam* simulation.

case, simulation is decided to be stopped when the inlet pressure variation is below 0.02% and average z-component velocity variation is below 0.2% both for the last 10 simulated seconds. This represents a ridiculously small variation compared to transient scenarios which should take the order of seconds. Additionally, all the residuals are monitored to be well below $10^{-4}$ and continuity error (mass balance in the domain) to be below $10^{-11}$. All those values are considered more than acceptable in order to assume that the solution is converged.

### 3.1.3 Post-processing

Once it has been decided that the simulation solution is converged, results can be analysed.

Thanks to the software *paraview*, graphical visualizations of the different variable fields can be obtained. For example, reduced pressure (pressure divided by density) distribution from a vertical section of the core is depicted in Figure 3.6 and the different visible regions in a vertical core are shown in Figure 3.5. Here, it can be observed that pressure field at the inlet seems a bit higher than expected, specially in the outer region where the pressure drop for the DEs is set to a greater value. In fact, using *postProcess* utility, an average inlet reduced pressure value of 18.1796 [$m^2/s^2$] is obtained, which corresponds to 1.909 [bar]. This value is significantly above the expected result (1.385 [bar]), specifically it is around 37.86% higher. This could be the result of a not completely converged solution or, more probably, it could be caused by the flow redistribution derived from an uniform inlet feeding the different cooling groups and elements.

Anyway, pressure gradients in the different regions composing the mesh seem consistent with the expected results: near uniform pressure drop profiles for CRs, SRs and DEs; while the FAs are subjected to a particular profile defined by the different layers which form them. Deviations from ideal theoretical behaviour are mainly found in the first centimetres of contact between DEs and outer elements from CGIV, where pressure gradient in y-direction is considerably high (in theory it should be a discontinuity produced by the bottom shroud of the different elements). This

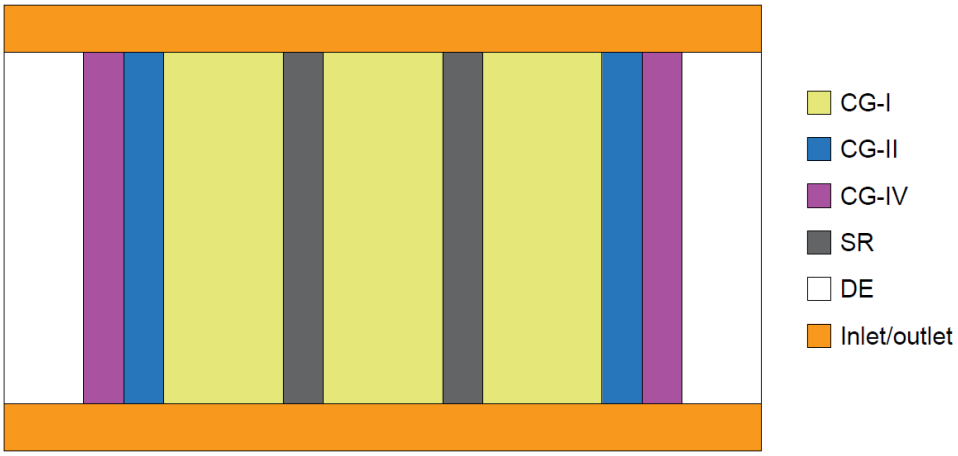



Figure 3.5. Different visible regions and cooling groups in a vertical section of the core.

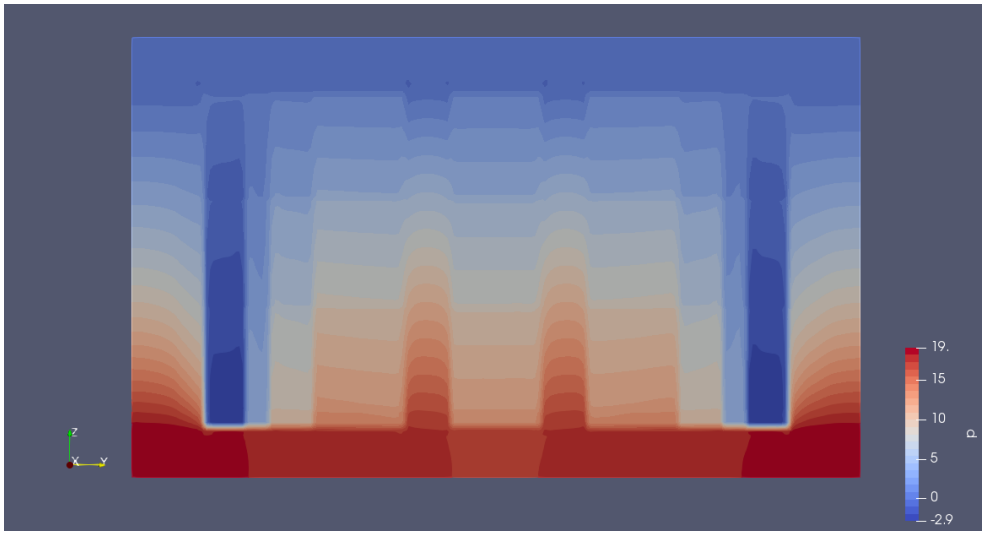



Figure 3.6. Reduced pressure [$m^2/s^2$] distribution from a vertical section of the nuclear core.

lack of uniformity is present in all the regions close to boundaries between different cooling groups and can be found in Figure 3.8, where some horizontal sections at different heights representing the pressure field are depicted.

It can be observed how the lack of pressure field uniformity inside each cooling group is higher near boundaries between groups and in lower horizontal sections (closer to the inlet). This latter observation is certainly related to the fact that the greatest pressure drop is found in a very reduced space defined by the first layers of each assembly, producing a pressure gradient difficult to simulate.

An analysis of the pressure drop in the different elements composing a FA can also be performed and is depicted in Figure 3.7. For this graph, FA number 0 has been chosen as a representative and the results from the simulation have been compared

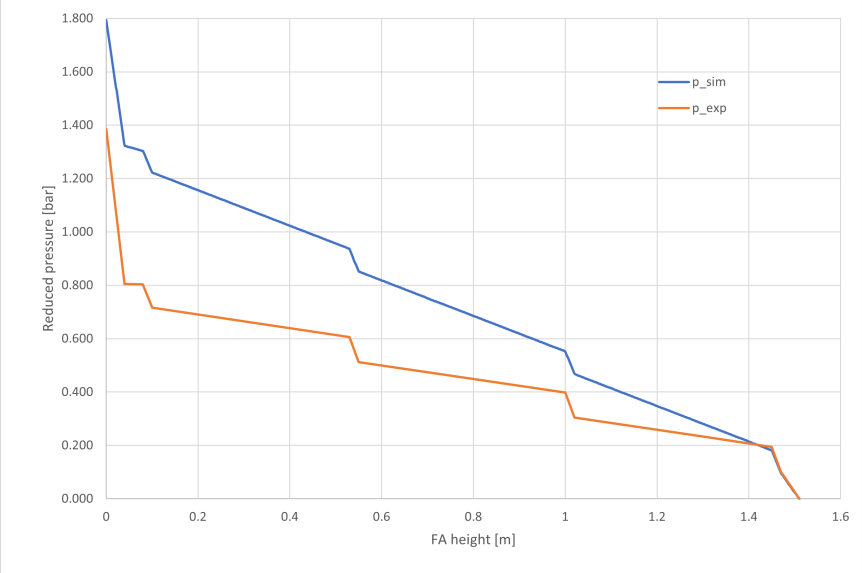
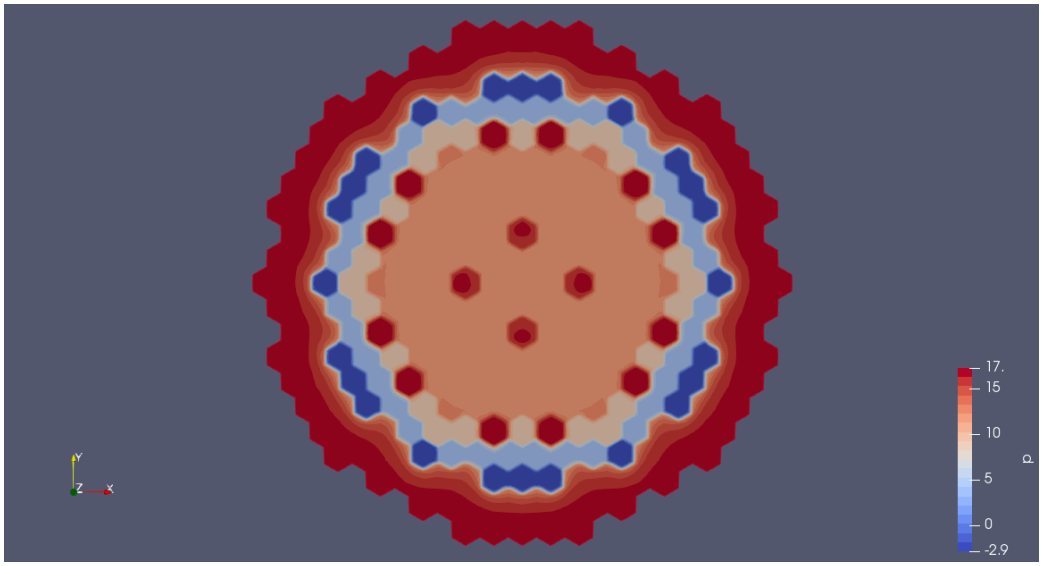
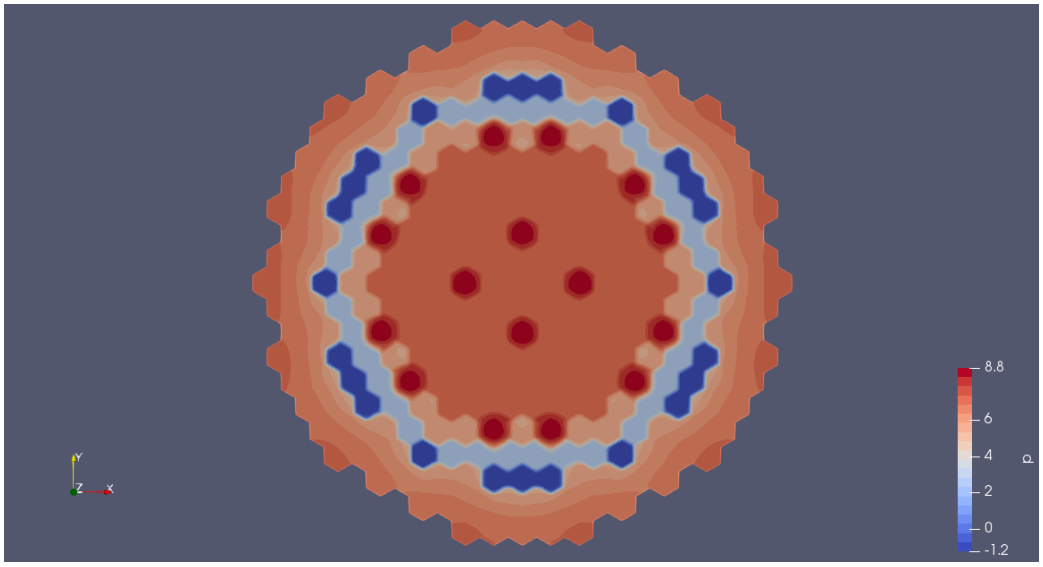


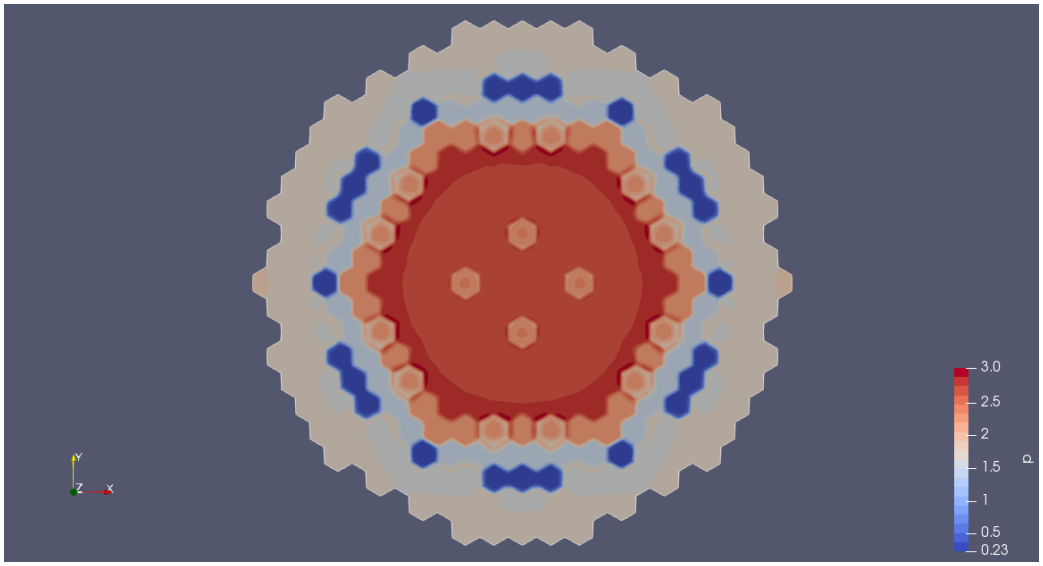

Figure 3.7. Pressure drop [bar] along FA number 0 according to simulation results (blue) and theoretically expected results (orange).




(a) Pressure field at z = 0.3 m.




(b) Pressure field at z = 0.955 m (middle).




(c) Pressure field at z = 1.5 m.

Figure 3.8. Pressure field [$m^2/s^2$] in horizontal sections at different heights.

to the expected results obtained from the pressure drop correlations in Chapter 2. It can be observed that, as already checked before, total pressure drop is higher for the simulated case than expected. Although pressure drop in pin bundle regions (z1, z2, act1, act2 and z3) are quite steeper for the simulated case than expected, pressure drop results for the grid spacers, inlet orifices and outlet funnel seem all quite consistent with respect to expected. This indicates that total pressure drop difference with respect to expected results is mainly concentrated in what happens for pin bundle regions, which are highly dependent on velocity profile. Hence, Darcy-Forchheimer coefficients in those regions should probably be calibrated, while the ones for the rest of the regions seem to work fine as they are set.

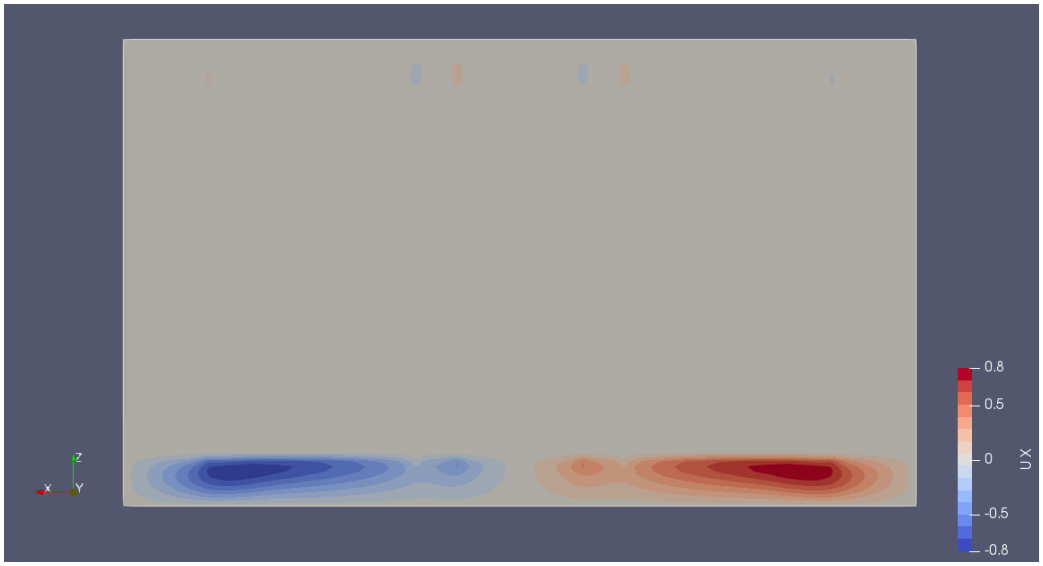
Apart from pressure, velocity components observed from a vertical section of the core are represented in the different subfigures from Figure 3.9. Thanks to this representation, it is confirmed the existence of a non trivial x- and y-direction flow in the inlet region corresponding to a flow redistribution in the different assemblies, causing an increase in turbulence and also pressure drop. On the other hand, the z-component seems to behave consistently with what is theoretically expected, although the obtained values may not completely coincide with those from Table 2.12. In fact, average velocity values have been studied for the active region of the different groups and compared to expected values. The results from this comparison are found in Table 3.1.

Table 3.1. Comparison between velocities values obtained from the simulation and expected ones.

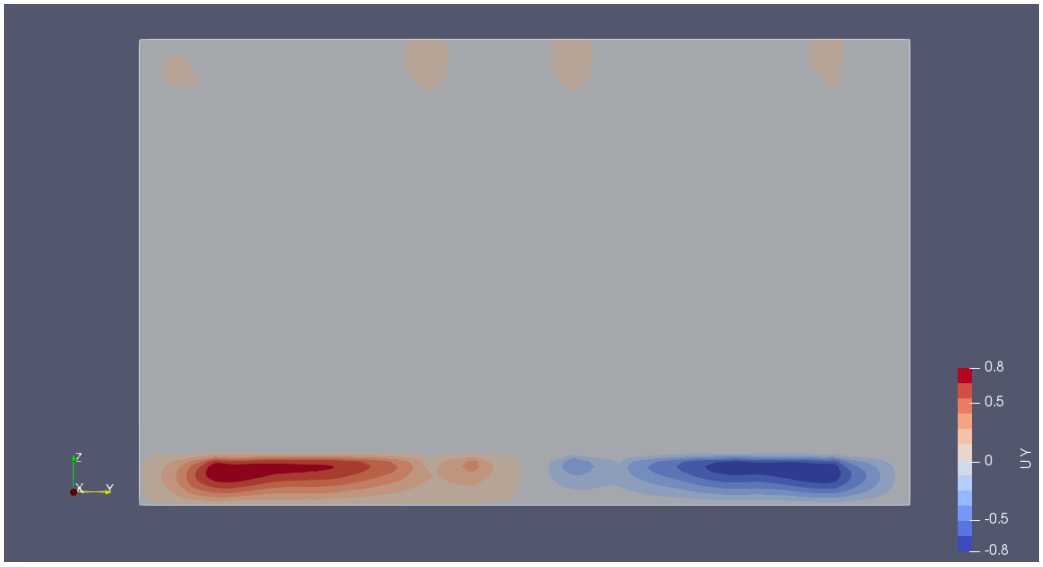
| Group | Representative element | Simulated velocity [m/s] | Expected velocity [m/s] | Relative error [%] |
|---|---|---|---|---|
| CGI | 0 | 0.5730 | 0.6478 | -11.55 |
| CGII | 91 | 0.5599 | 0.5459 | 2.56 |
| CGIII | 168 | 0.5190 | 0.4418 | 17.47 |
| CGIV | 127 | 0.4793 | 0.3512 | 36.49 |
| CR | 93 | 0.0964 | 0.0644 | 49.71 |
| SR | 7 | 0.0864 | 0.0644 | 34.07 |
| Dummy | 219 | 0.0052 | 0.0050 | 5.39 |

This table shows big differences between the obtained and the expected results specially in CGIV and in CR and SR systems. A possible explanation is that the flow redistribution at the inlet causes turbulent eddies (as seen in Figures 3.9a and 3.9b) which physics can not be completely captured by such a coarse mesh prepared for porous media approach. This may cause some convergence issues and affect general velocity distribution in the domain. A solution for this issue would be enlarging inlet region and refining the mesh there, but that would increase computing time and resources for calculations in a region of lower interest. In order to study solution accuracy dependence on mesh refinement, a coarser mesh case was simulated, obtaining velocity values very similar to the ones from the finer mesh case. This comparison is contained in Table 3.2 and shows that solution accuracy is quite independent from mesh refinement, although further analysis could be made.

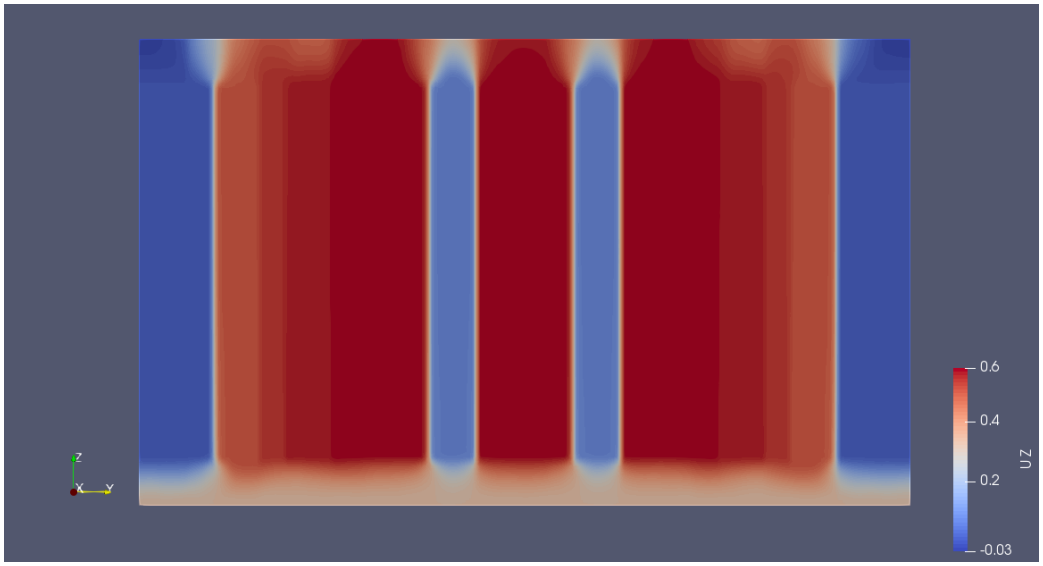
Another possible solution would be setting a non uniform inlet condition which




(a) Velocity x-component looking from positive y-axis.




(b) Velocity y-component looking from positive x-axis.




(c) Velocity z-component looking from positive x-axis.

Figure 3.9. Different velocity components [m/s] in a vertical section of the core.

Table 3.2. Comparison between velocities values obtained from finer and coarser mesh cases.

| Group | Representative element | Finer case velocity [m/s] | Coarser mesh velocity [m/s] | Difference [%] |
|---|---|---|---|---|
| CGI | 0 | 0.5730 | 0.5622 | -1.88 |
| CGII | 91 | 0.5599 | 0.5507 | -1.65 |
| CGIII | 168 | 0.5190 | 0.5269 | 1.51 |
| CGIV | 127 | 0.4793 | 0.5003 | 4.39 |
| CR | 93 | 0.0964 | 0.0744 | -22.84 |
| SR | 7 | 0.0864 | 0.0727 | -15.89 |
| Dummy | 219 | 0.0052 | 0.0047 | -9.92 |

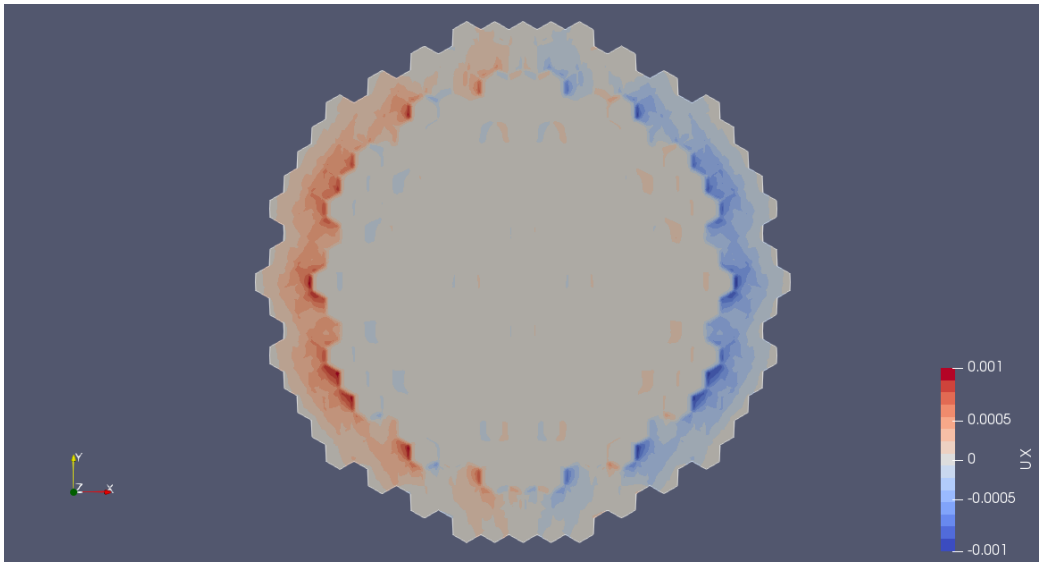
could help minimizing the redistribution flow. Darcy-Forchheimer coefficients could be also slightly modified to correct the pressure and velocity fields. In fact, a recursive procedure should be adopted, by changing the D-F coefficients, in order to reach the correct value.

As a consequence for that velocity profile deviation from theory and although qualitative behaviour of pressure seems consistent as discussed before, pressure drop values for the different regions and groups can not be consistent with theoretical expected results as velocity is a dominant variable in pressure calculations.

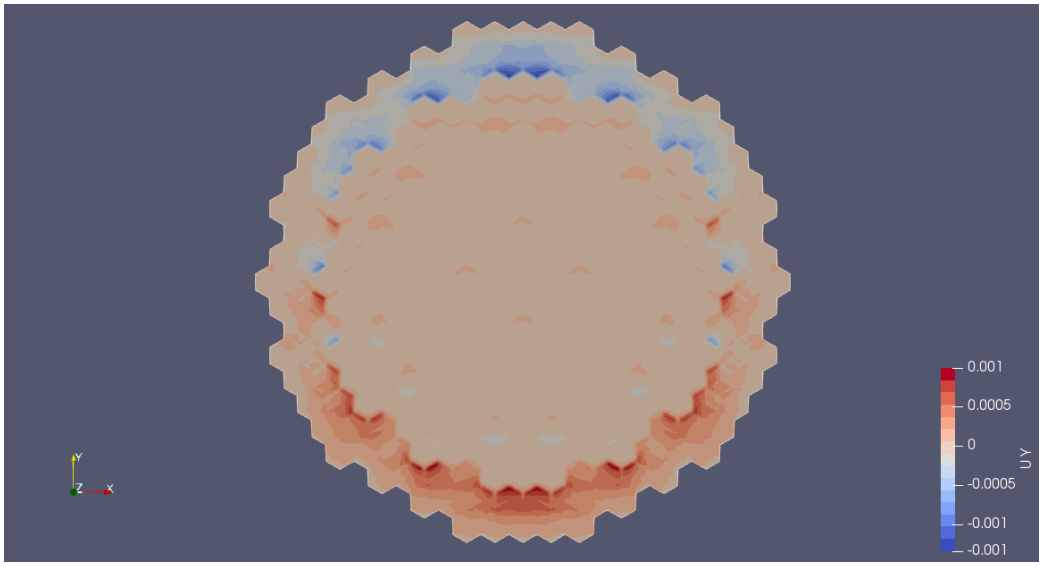
Anyway, the results presented prove the feasibility of porous media implementation for pressure drop simulation in this kind of reactor. Its ability to restrict the flow in x- and y-direction is reflected in Figure 3.10, and it can be easily observed how these components are almost negligible once inside the FAs. The biggest values for x and y velocity components are found in the DEs and close to their boundary with CGIV, where the difference of porosity coefficient values between adjacent elements is bigger producing an attempt of flow migration. Anyway the simulated value for x- and y-components is usually kept under 0.001 absolute value which is good and could be further reduced by increasing x- and y-direction Darcy-Forchheimer coefficient values.

Some problems may arise in the boundaries between different groups where a discontinuity in the velocity field is theoretically expected. As this condition is difficult to be simulated, some lack of uniformity might be found in those regions. This is closely related to what happened with pressure, as velocity and pressure fields are strongly coupled.

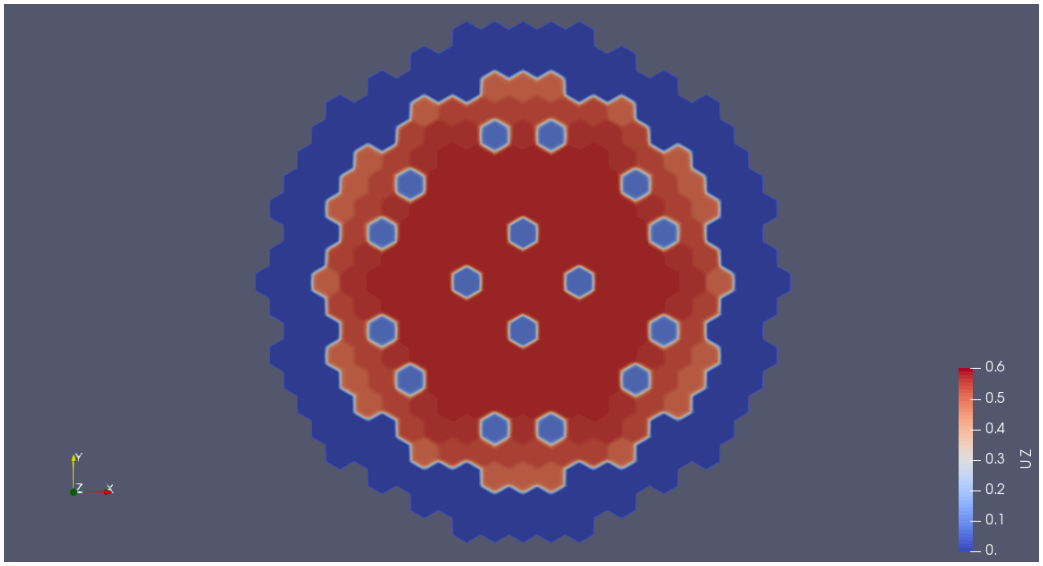
In conclusion, porous media approach has been successfully implemented obtaining consistent qualitative results, although numerical values for the different variable fields may vary from what was theoretically expected. Additionally, some possible causes and solutions have been purposed for those issues.




(a) Velocity x-component looking from positive z-axis.




(b) Velocity y-component looking from positive z-axis.




(c) Velocity z-component looking from positive z-axis.

Figure 3.10. Different velocity components [m/s] in a horizontal section of the core.

## 3.2 Case 2: Steady-state with heat sources

Once a converged steady-state solution just for the porous media has been reached, it can be used as the initial condition for another steady-state porous media simulation, but now setting the corresponding heat sources for the active region. Objectives now are: analysing the feasibility of the method explained in Section 2.5 to set the different internal heat sources, verifying the results obtained with respect to theoretically expected results, and obtaining stable variable fields that can be used as initial conditions for later transient simulations.

### 3.2.1 Pre-processing

Main part of the pre-processing information is exactly the same as the one from the case before, so only specific differences for this case will be highlighted in this section.

Following the explanation from Section 2.5, the solver used in this case is *my_buoyant SimpleFoam* so that volumetric power sources can be simulated.

As now heat transfer needs to be simulated, physical models required for this simulation are a bit more complex. Instead of having a transport model as in the previous case, some thermophysical models need to be set and are contained in the file *thermophysicalProperties*, which is written as follows:

```
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState Boussinesq;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        molWeight       207.2; //[g/mol]
    }
    equationOfState
    {
        rho0            10580;  //reference density
        T0              673;    //reference temperature, [K]
        beta            1.209e-04;  //volumetric expansion
    }
    thermodynamics
    {
        Cp              145; //[J/kg.K]
        Hf              0;
    }
```

```
    transport
    {
        mu              0.002;   //dynamic viscosity, [Pa.s]
        Pr              0.01698; //Prandtl number, cp*mu/k
    }
}
```

All the data for those models have been extracted from its corresponding sections in Chapter 2.

As already mentioned, the converged solution from the previous case is used as initial condition, so that *U*, *k*, *epsilon* and *nut* files are directly copied from the last time step of Case 1. On the other hand, pressure field requires to be adapted because previous results were given as reduced pressure and now total pressure values are needed. In order to do so, the Python program from Appendix D can be used. That processed pressure field is then set as initial condition for *p* and *p_rgh*. In addition, three other new fields are required: *T* (representing the temperature), *alphat* (representing the turbulent energy diffusion) and *volPow* (which was already explained in Section 2.5 and represents the volumetric power source).

A summary of all boundary conditions can be obtained thanks to *pyFoamCaseReport* utility and its output is shown below:

```
The boundary conditions for t = 0
================================


Boundary:  inlet
----------------

:Type:   patch
:Physical:       patch
:Faces:  43659


======= ============ ==================
Field   type         value
======= ============ ==================
T       fixedValue   uniform 673
U       fixedValue   uniform (0 0 0.321)
alphat  calculated   uniform 0
epsilon fixedValue   uniform 5.3985e-06
k       fixedValue   uniform 0.0003862
nut     calculated   uniform 0
p       zeroGradient
p_rgh   zeroGradient
volPow  fixedValue   uniform 0
======= ============ ==================


Boundary:  outlet
```

```
29  -----------------
30
31  :Type:   patch
32  :Physical:       patch
33  :Faces:  43659
34
35
36  ======= ============ =========
37  Field   type         value
38  ======= ============ =========
39  T       zeroGradient
40  U       zeroGradient
41  alphat  calculated   uniform 0
42  epsilon zeroGradient
43  k       zeroGradient
44  nut     calculated   uniform 0
45  p       fixedValue   uniform 0
46  p_rgh   fixedValue   uniform 0
47  volPow  fixedValue   uniform 0
48  ======= ============ =========
49
50
51  Boundary:  wall
52  ---------------
53
54  :Type:   wall
55  :Physical:       wall
56  :Faces:  82908
57
58
59  ======= ============ =========
60  Field   type         value
61  ======= ============ =========
62  T       zeroGradient
63  U       slip
64  alphat  calculated   uniform 0
65  epsilon zeroGradient
66  k       zeroGradient
67  nut     calculated   uniform 0
68  p       zeroGradient
69  p_rgh   zeroGradient
70  volPow  fixedValue   uniform 0
71  ======= ============ =========
```

Setting solvers, tolerances and relaxation factors in *fvSolution* file for this case is very similar to the previous one, although some slight changes must be included: $p$ has to be substituted by $p\_rgh$ and a solver for $e$ (energy term) has to be defined too. For this case, $e$ is included in the same group as $k$ and *epsilon* before. SIMPLE algorithm is still required and the same considerations as in the previous case about

tolerances, correctors and relaxation factors are used.

About discretization schemes, the only difference with respect to the previous case is changing and adding some divergence schemes as follows:

```
divSchemes
{
        default           none;

        div(phi,U)        bounded Gauss upwind;
        div(phi,e)        bounded Gauss upwind;

        div(phi,k)        bounded Gauss upwind;
        div(phi,epsilon) bounded Gauss upwind;

        div(phi,Ekp)      bounded Gauss linear;
        div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}
```

The rest of the *fvSchemes* file may remain unchanged.

### 3.2.2 Processing

Same considerations are followed as in the previous case. Residuals, some important field variables (inlet pressure and velocity in the different cooling groups) and mass balance in the domain are monitored in order to check for convergence. In addition, a new function is monitored in this case: area average outlet temperature. As inlet temperature is fixed by boundary conditions to 673 [K], the objective now is checking for the time step at which average outlet temperature is established around 753 [K].

For this case, the solution is supposed to be converged when the inlet pressure variation is under 0.35%, the average z-component velocity variation is below 0.2%, and the average outlet temperature variation is around 0.01% for the last one thousand iterations. Those values are considered to represent a very small variation when compared to transient simulations. Additionally, all the residuals are monitored to be well below $10^{-5}$, except pressure residual, which value is oscillating around $10^{-4}$. Furthermore, continuity error (mass balance in the domain) is below $10^{-11}$. All those values are considered more than acceptable in order to assume that the solution is converged.

### 3.2.3 Post-processing

The most important results to be checked in this case are the ones related to energy transfer: power generated by the volumetric heat sources and temperature distribution inside the core.

It can be checked that both pressure and velocity fields remain constant from the previous case, so there is no reason to make any new comment. Respective vertical sections of those fields are depicted in Figures 3.11 and 3.12. As in the previous case, average inlet pressure shows around a 40% deviation with respect to theoretically

expected results, while main velocity field deviations are found in CR, SR and CGIV reaching values between 30 and 50%.

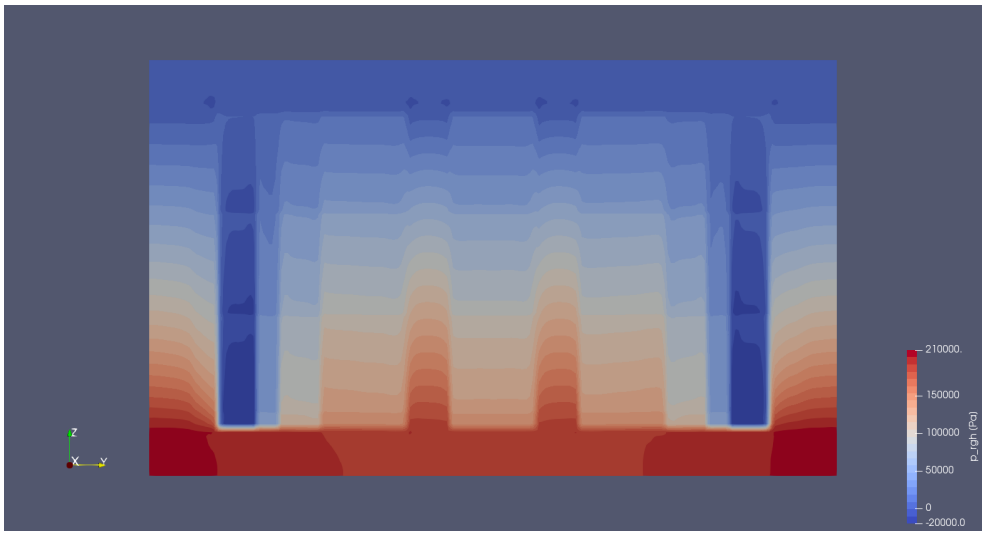



Figure 3.11. Pressure [Pa] distribution from a vertical section of the nuclear core.

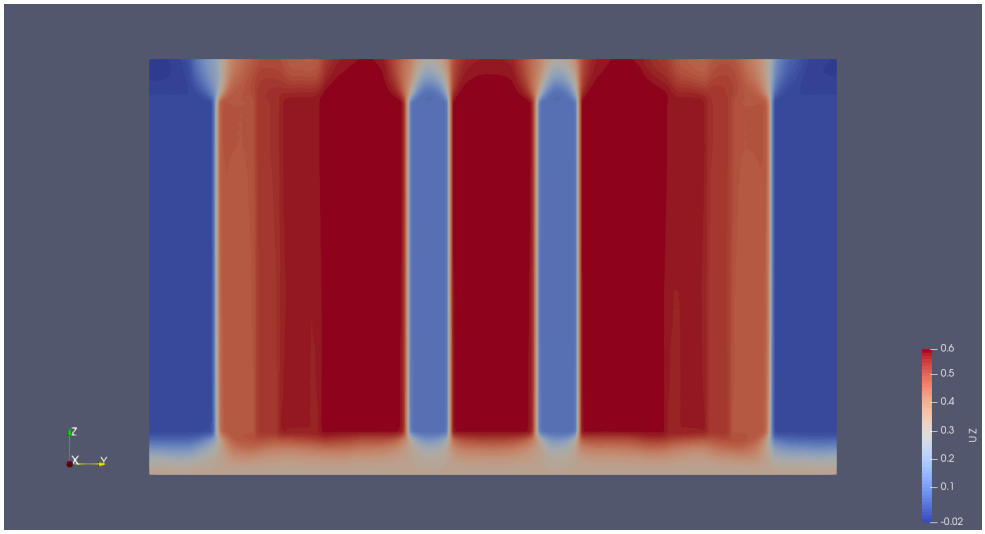



Figure 3.12. Velocity z-component [m/s] from positive x-axis in a vertical section of the nuclear core.

On the other hand, one of the first tasks when analysing the results of this new case is checking if volumetric power sources have been correctly applied. In order to do so, *volPow* field at time 0 can be represented in *paraview* and is shown in Figure 3.13. In that way, it can be checked that the corresponding values for the internal heat sources are correctly applied. All the values shown here seem consistent with the fields applied by *setFields* utility.

Once that the model implementation has been checked, temperature results depicted in Figure 3.14 can be analysed. Heat transfer seems consistent with what was theoretically expected: coolant temperature increases with height thanks to the heat transfer mainly produced in the active zone of the FA, and a maximum is observed in

the outlet central part, corresponding to the elements transferring higher heat values. In fact, the average outlet temperature is 750.047 [K] while the theoretically expected result is 753 [K]. This means that the temperature difference between the inlet and the outlet is 77.047 [K] instead of 80 [K], which represents only a 3.69% deviation.

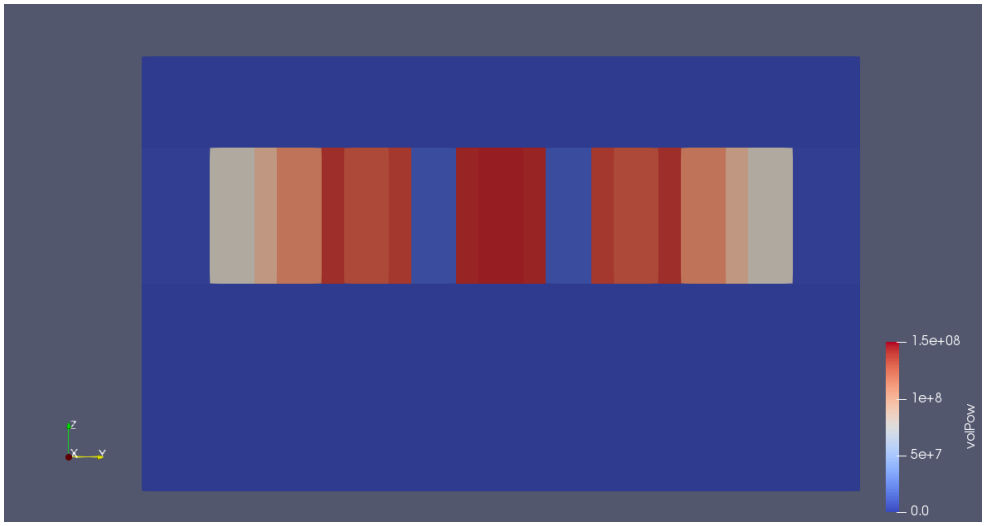



Figure 3.13. Representation of internal volumetric power sources (*volPow* field) [W/m$^3$] in the inside of the nuclear core.

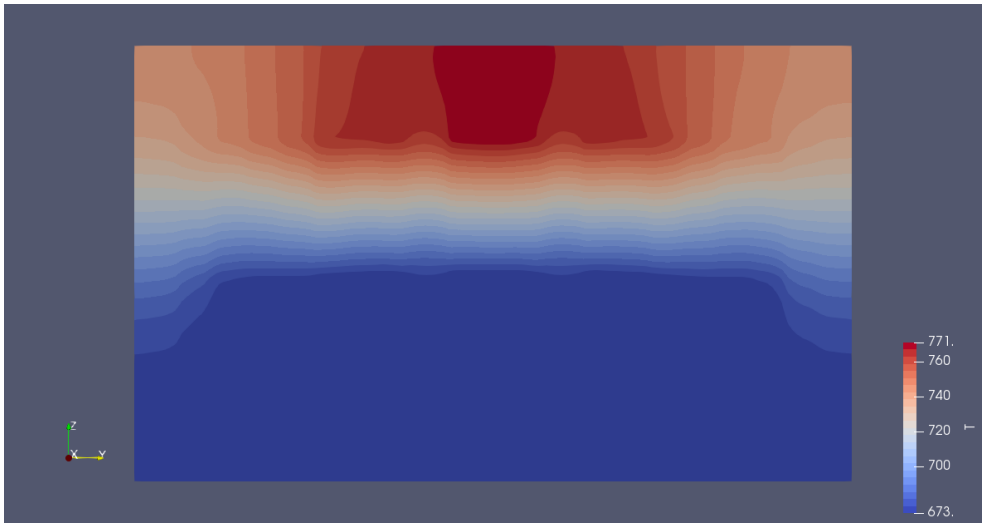



Figure 3.14. Temperature distribution [K] in a vertical section of the nuclear core.

Heat transfer through a general FA is better understood thanks to Figure 3.15, which represents the temperature profile across the height of FA number 0. Here it is observed how temperature increase, and hence heat transfer, is produced in the active zone. This result can be related to velocity field (mass flow) and heat sources using some of the thermophysical properties explained in previous sections. This particular linear shape is related to the fact that a constant volumetric heat source is assumed in this case. For the real case in which generated power has an axial distribution close to a sinusoidal curve, the shape of this graph would be similar to a S-curve.

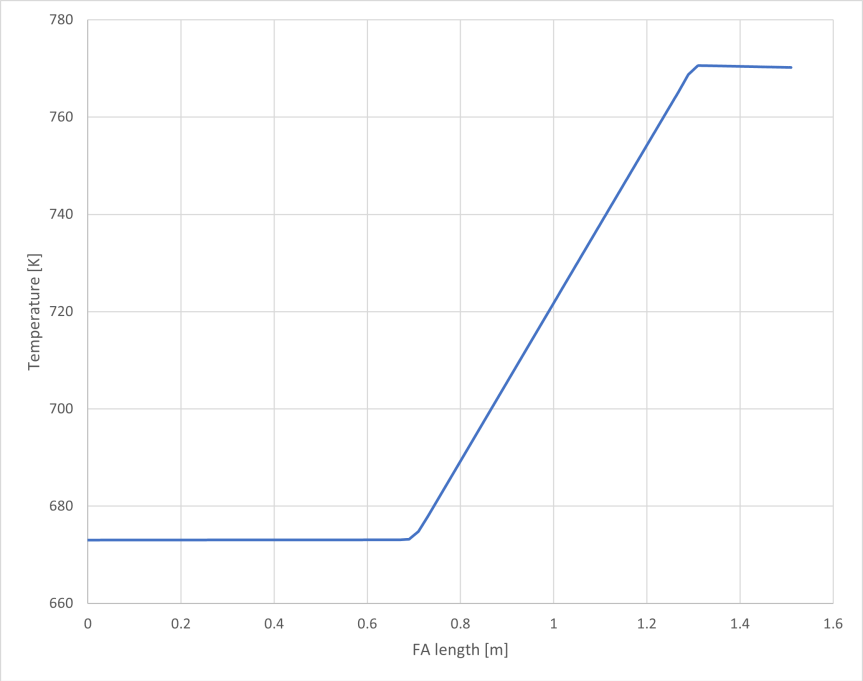



Figure 3.15. Temperature profile [K] through FA number 0 height.

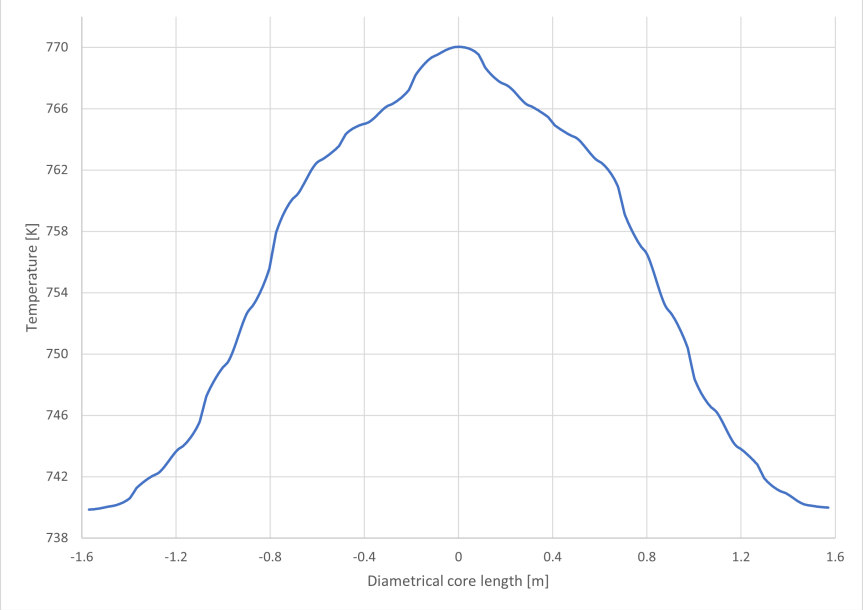



Figure 3.16. Temperature profile [K] in the outlet region of the core through y-axis.

But, although the average outlet temperature value seems consistent with theory, its profile might not be so related to the expected results. This can be better observed in Figure 3.16, which represents the temperature profile in the outlet region through diametrical length (y-axis). The shape of this profile denotes a maximum temperature peak in the central part of the core, which corresponds to a bigger heat transfer produced in central elements. Some fluctuations are observed probably due to velocity field (turbulence at the outlet) and numerical issues.

When comparing this temperature profile to expected results from previous simulations,[15] a higher difference between maximum and minimum coolant temperatures at the core outlet is found: $\sim$30[ºC] versus $\sim$17[ºC]. Main reason for this deviation from expected results could be the fact that, as already concluded from previous case, velocity field is not accurately simulated with respect to expected results, which certainly has an impact on heat transfer. Different velocity values means different mass flows evacuating heat and, hence, a different outlet temperature distribution. This issue should be fixed once velocity field values are closer to what is expected as nothing seems to indicate that there is an implicit problem in energy transfer.

In conclusion, heat sources have been successfully implemented obtaining consistent average temperature values with respect to theoretically expected results. It is true that some differences are found when observing the outlet temperature profile, but deviations from ideal behaviour seem to be only derived from velocity field inaccuracies.

## 3.3 Case 3: Obstructions

The previous two cases studied the implementation of the different models for the simulation of steady operating condition, which is supposed to be the main working regime in nuclear core's life. Those cases also constitute a preparation for unsteady simulations, as they can be used as departure point.

Transient phenomena may arise in different situations: thermal power changes through fuel life cycle, control rods insertion, safety rods insertion (SCRAM), flow obstructions, etc. The aim of this case is precisely this latter one, the simulation of flow obstructions, studying its effect on pressure, velocity and temperature fields. The importance of this study remains in the fact that flow obstructions may become an issue in LFR due to high coolant density and may be mainly produced in the inlet orifices and in the different grid spacers (which correspond to the narrowest regions). As a consequence of a flow obstruction in a particular FA, mass flow can be strongly restricted, reducing heat transfer capacity and, hence, increasing coolant temperature upto a point in which it could affect core safety.

### 3.3.1 Pre-processing

Settings required for this case are quite similar to the previous case ones. The only difference is that a transient simulation is wanted in order to capture the coolant flow evolution with time when an obstruction is produced. For that reason, and following explanations from Section 2.5, the solver used for this case is *my_buoyantPimpleFoam*, which accounts for a transient simulation in presence of internal heat sources. This

introduces some slight and unimportant changes in *fvSolution* and *fvSchemes* files, but all the rest is pretended to remain unchanged.

No change is introduced in physical models files from *constant* directory.

Initial conditions are taken from the previous case converged solution and boundary conditions are defined exactly in the same way as before.

The *fvOptions* file remains mainly unchanged, except the porosity coefficient of the region where the obstruction wants to be simulated, which has to be increased. For example, in this case the obstruction is simulated in the inlet orifices of FA number 0. How big this increase should be is not a trivial matter as this would affect pressure and velocity fields which are both strongly coupled, so an estimation of the required Darcy-Forchheimer coefficient is quite complicated. For this first approach, it has been decided to use a value for the inlet orifices pressure drop four times higher than the original one (from Table 2.7). Following the same reasoning from Section 2.4 and using the same reference velocity, a new value for the porosity coefficient in this obstructed region (inlet orifices from FA number 0) has been found: 2,628.428 [1/m]. Note that this value is nothing but the result of multiplying the original porosity coefficient of this region by four.

### 3.3.2 Processing

Same procedure as in previous cases is followed: monitoring the same variables, residuals and mass balance. Simulation is stopped when the variation for the last 10 simulated seconds of average inlet pressure is under 0.05%, of velocity fields is below 0.15% in general, and around 0.005% for average outlet temperature. Residuals are kept below or around $10^{-4}$ for the whole simulation (except for the very first time steps) and continuity error is below $10^{-9}$. These values could be considered small enough to assume that the transient response has been successfully simulated.

### 3.3.3 Post-processing

As in the previous cases, pressure, velocity and temperature fields are analysed. A bigger pressure drop at the inlet orifices of FA number 0 is expected as a consequence of flow obstruction simulated by setting a higher porosity coefficient in that region, as well as a reduction in the velocity field inside that whole FA. Additionally, the spare mass flow is expected to be redistributed among the neighbouring FAs, which leads to an increase in the velocity field inside of them. As a consequence of this new velocity field, temperature profile is expected to change accordingly, even showing some hotspots in obstructed regions where coolant flow might not be enough to evacuate all the generated heat. This transient response is expected to last around some few seconds, as reference inlet velocity is set to 0.321 [m/s] and domain dimensions are 1.91 [m] high.

But, before entering in any further analysis, one of the first observations that can be made is that time scale is somehow lost, dynamics effects are not well captured. This is reflected in the fact that flow changes are produced much slower than expected, and the results after 164 seconds seem far to what was previously expected. For this reason, the following results cannot be studied as the solution from a transient simulation, but will be taken as the final converged solution of a steady one.

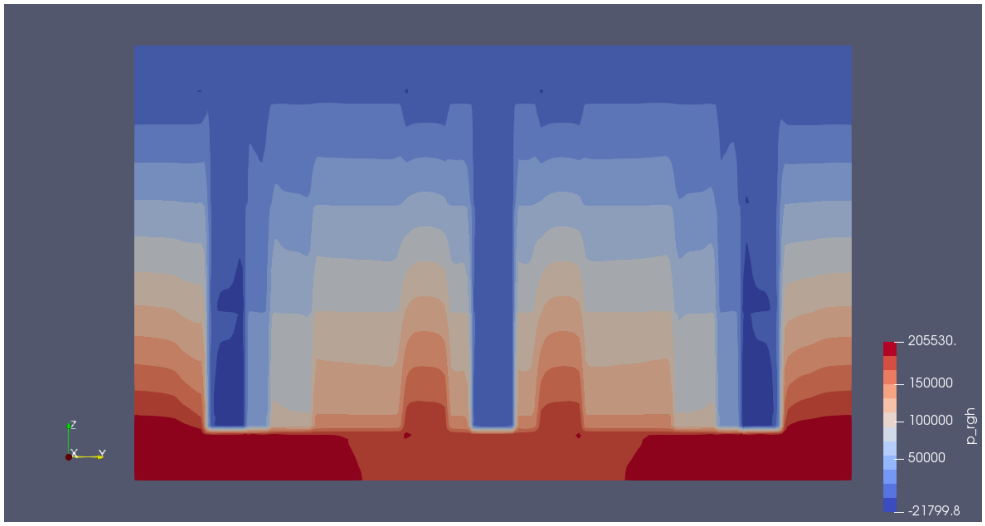



Figure 3.17. Pressure [Pa] distribution from a vertical section of the nuclear core when an obstruction at the inlet orifices of FA number 0 is produced.

Some comments on pressure field, which vertical section is depicted in Figure 3.17, can be made. First, area average inlet pressure at the end of the simulation (second 164) is 1.940 [bar] which is only 0.031 [bar] higher than previous cases. This slight difference can be considered almost negligible as represents around a 1.5% variation that could be due to numerical reasons. Anyway, in Figure 3.17, the increase of pressure drop at inlet orifices from FA number 0 is observed and, hence, the higher porosity coefficient implementation in that region is confirmed. This effect is also observed in Figure 3.18, where a field representing the pressure difference before and after the obstruction is shown. There, it can be noticed how pressure drop for FA number 0 is concentrated at the inlet orifices region, while for the rest of axial regions pressure drops are lower than the ones from the previous case. Some slight differences that can be observed at the core outer region are only due to numerical variations in

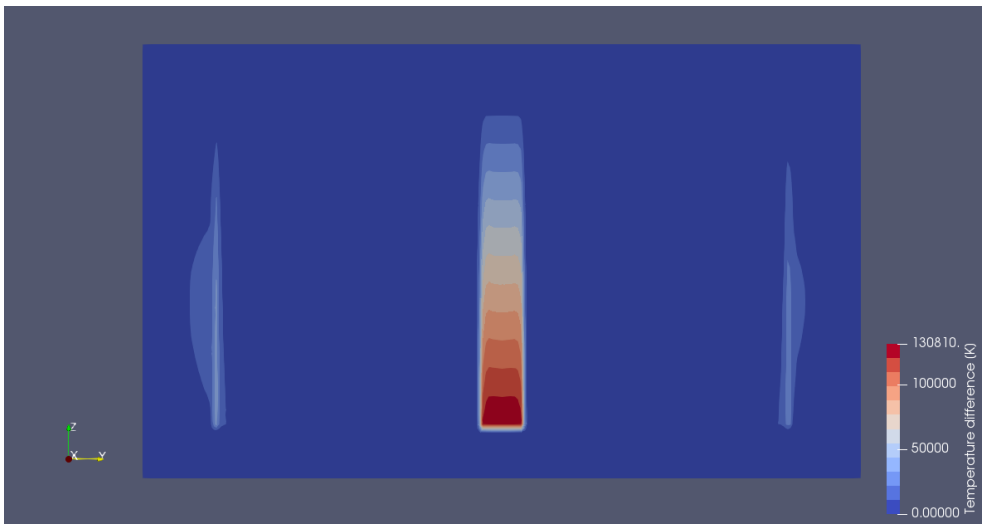



Figure 3.18. Pressure difference [Pa] distribution before and after the obstruction in a vertical section of the core.

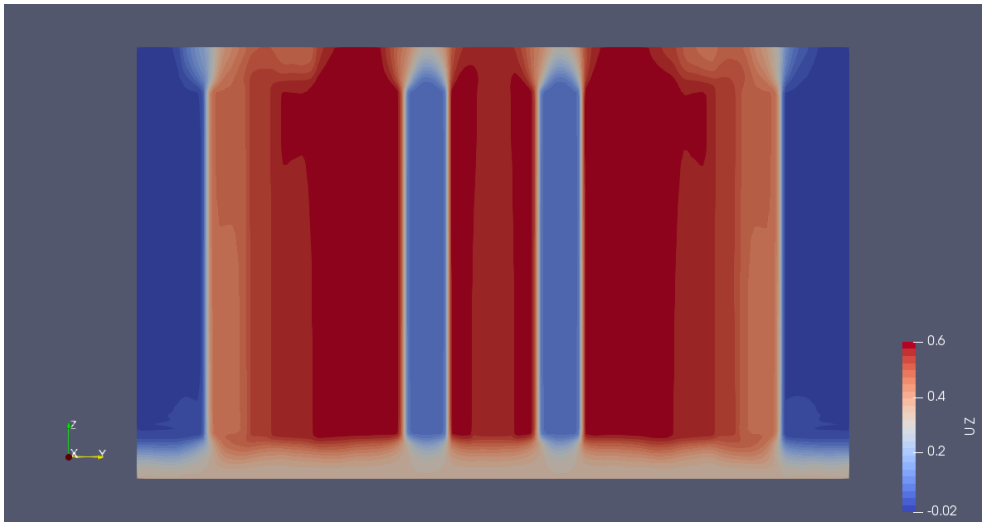



Figure 3.19. Velocity z-component [m/s] from positive x-axis in a vertical section of the nuclear core.

boundary regions between different cooling groups.

But, although some differences can be noticed in pressure field, velocity does not seem much affected by the simulated obstruction. In Figure 3.19, the velocity z-component in a vertical section of the core is depicted, where it can be observed how velocity field inside FA number 0 seems slightly lower than before.

The decrease of velocity z-component in FA number 0 due to the obstruction can be better observed in Figure 3.20, where the velocity difference before and after the obstruction is plotted. A small decrease in velocity field inside FA number 0 caused by the obstruction is confirmed. According to this Figure, the magnitude of the decrease seems to be between 0.01 and 0.02 [m/s], which is a really low difference with respect to what was expected. A similar result can be obtained when comparing the volumetric average velocity in FA number 0 before the obstruction (0.5803 [m/s])

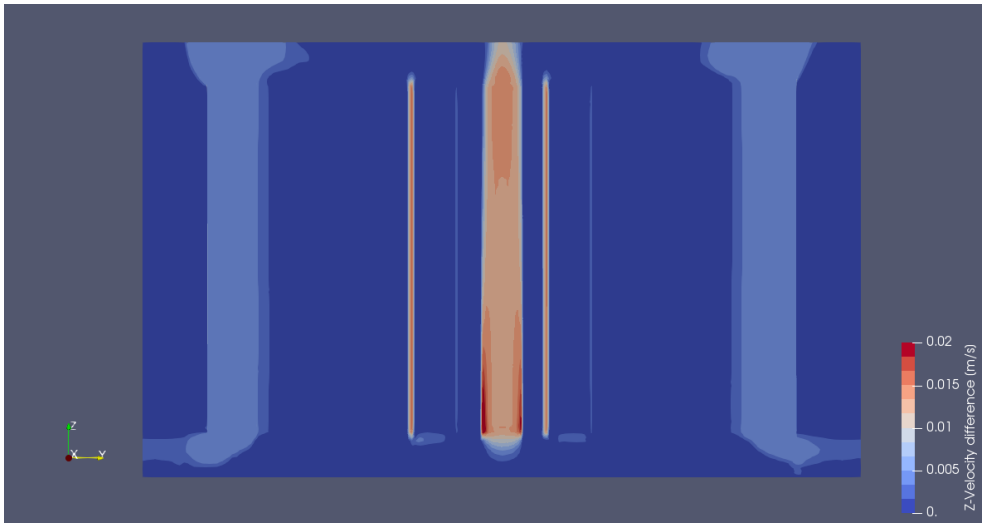



Figure 3.20. Velocity z-component difference [m/s] distribution before and after the obstruction in a vertical section of the core.

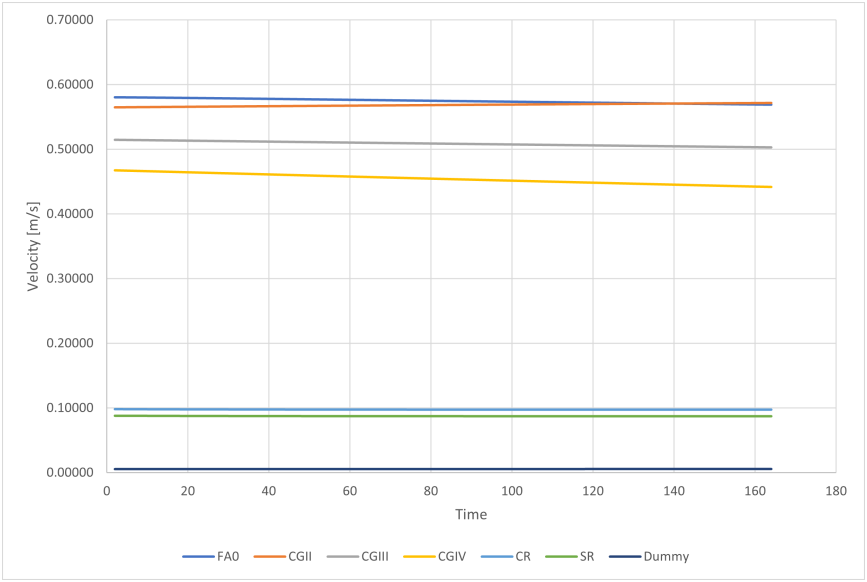



Figure 3.21. Comparison of volumetric average velocities [m/s] time evolution in the different cooling groups.

and after (0.5689 [m/s]), obtaining an volumetric average velocity difference of 0.0114 [m/s]. This minor variation observed does not seem coherent with the additional pressure drop produced by the obstruction.

In fact, the variation of average velocities in the different cooling groups and FA number 0, which is depicted in Figure 3.21, shows a very slow time evolution which does not correspond with the physics intended to be simulated in this case.

As a consequence of this minor effect on velocity distribution inside the nuclear core, temperature field remains quite unchanged from previous case and no hotspots are detected as observed in Figure 3.22.

As already made for pressure and velocity, temperature field change before and after

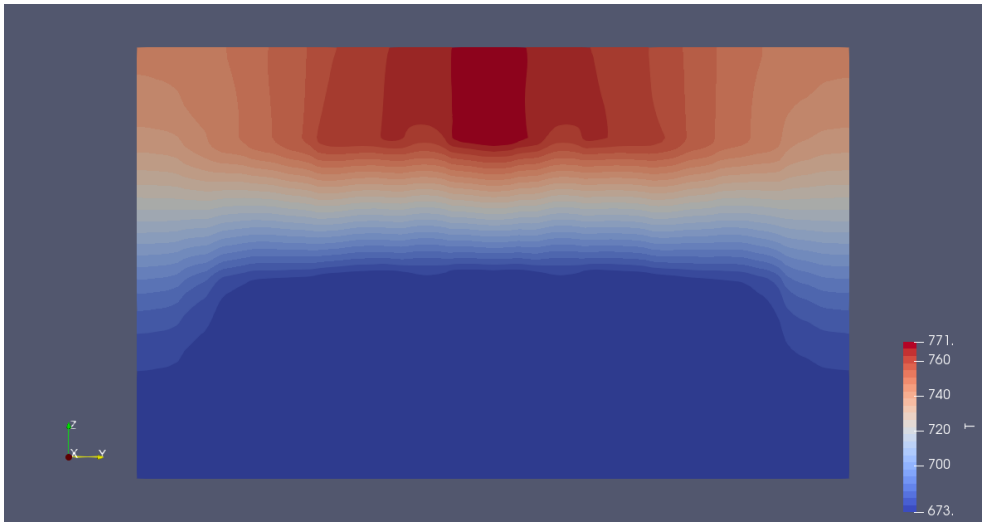



Figure 3.22. Temperature distribution [K] in a vertical section of the nuclear core when an obstruction at the inlet orifices of FA number 0 is produced.

the obstruction is better observed in Figure 3.23, where the temperature difference is plotted. It is shown how temperature differences are very low (4 [K] as maximum) and they are mainly placed far from the region of interest in general. This probably means that the temperature variation is caused by a better convergence of the flow in outer regions rather than by the flow obstruction, although some slight changes can be observed in FA number 0 active region too. Anyway, these changes are very small and almost negligible.

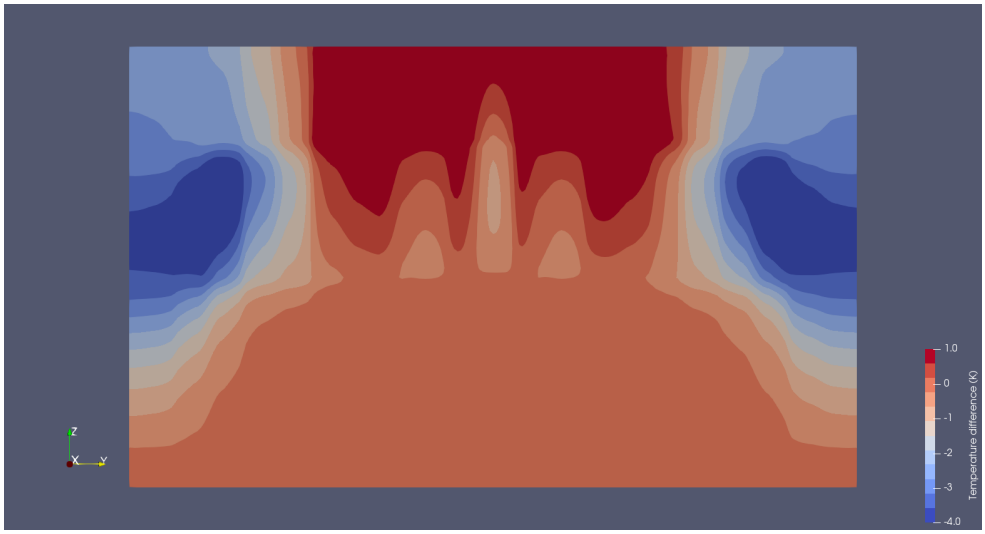



Figure 3.23. Temperature difference [K] distribution before and after the obstruction in a vertical section of the core.

All the results exposed from this case, and its comparison with respect to the previous one, seem to indicate that this explicit porous media approach is not behaving well under transient simulations, specially for the velocity field, which has a great impact on energy transfer and, hence, in the resulting temperature profile.

## 3.4 Concluding remarks

Before entering the final conclusions and future lines for development, it seems a good exercise discussing some concluding remarks from all the previous cases.

First of all, it has been shown how an explicit porous media approach has been successfully implemented, which was possible partially thanks to a correct geometry definition which can generate different cell zones for the different core assemblies and axial regions, the so-called *layers approach*. This configuration offers a lot of flexibility when tuning porosity coefficients and simulating different operating conditions, e.g. obstructions in any of the regions.

As a result of this porous media implementation, a good prediction of pressure field in general has been obtained, although some over pressure has been found at the inlet and in longer regions of the FAs. This could be due to the flow redistribution produced in the inlet region as a consequence of an uniform boundary condition when, on the other hand, each of the cooling groups demand a different mass flow defined by

the specific gagging scheme. The solution for this issue could be enlarging the inlet region or even changing the boundary condition to a non-uniform one.

In opposition, velocity field has shown worse results in general: values quite deviated from expected results and longer times to converge. This was specially noticeable during the transient simulations for obstructions, where velocity field did not change as fast as expected. This may indicate that explicit porous treatment is not a good tool for inducing velocity just from a pressure distribution specially during transients.

As a natural consequence, energy transfer is strongly influenced by velocity field behaviour and, hence, the resulting temperature field is quite deviated from expected results. However, the implementation of the internal heat sources calculated from the fission power and the rest of physical models seem to work fine.

# Chapter 4

# Final conclusions and future developments

As already mentioned at the beginning of this document, the aim of this thesis is to study the general operation of the Generation-IV Lead-cooled Fast Reactor ALFRED by CFD analysis, simulating the different operating conditions that may appear. As this is quite a general and broad objective, it seems useful establishing a division in some specific objectives, which were already included in Section 1.1. Extending this reasoning, the following conclusions for each of those specific objectives can be reached:

1. A suitable mesh geometry which captures the most relevant information from the simulation and allows to work with the maximum flexibility has been generated. For that purpose, a division of the geometry into different cell zones for the different core assemblies and axial regions composing them was performed thanks to the software GMSH. This method has proven to be very flexible for the simulation of many different operating conditions of the core, like obstructions in any of the regions defined before.

   Additionally, mesh quality has been satisfactory checked and even a coarser case was simulated in order to try to prove solution independence on grid size.

2. An explicit porous media approach has been successfully implemented. For that purpose an extensive analysis on the different pressure drops inside a general FA and the rest of core assemblies has been carried out. Pressure drop values, obtained by different correlations and information from previous assessments performed by core designers, have been compared to those expected by core designers previous CFD simulations and only small deviations have been found, which supports the calculations used in this thesis work. From those values, a methodology for porosity coefficients calculation and implementation has been proposed, reaching positive results.

3. In general, this approach has shown good performance at normal operating conditions simulated as a steady-state case, specially calculating the pressure field. On the other hand, velocity field has shown worse results: greater deviations from expected values and longer times to converge. This may indicate that inducing the velocity field only by setting a pressure scheme (thanks to porous

media approach) is difficult, which might be caused by two main reasons: porous approach does not set any actual flow area restriction which really conditions the mass flow, and the fact of using an explicit treatment for porosity instead of implicit.

It seems important highlighting here that energy transfer is strongly influenced by the velocity field, so an inaccurate velocity field leads to an inaccurate temperature field with respect to expected results. Apart from that issue, volumetric heat sources and thermophysical models seem to work fine.

4. When simulating a transient case for the obstruction simulation, the effects described in the previous point have been even more visible: pressure field seems quite consistent with respect to expected results while the velocity field shows deviated values. Additionally, time scale seems to be somehow lost and dynamics effects are not well captured, taking longer simulation times than expected. All these only confirm the ideas from the former point: velocity field is not well induced by the explicit porous media approach, specially in transient simulations.

At this point, it is important to note that, although fuel temperature is one of the most critical points in a nuclear application like this one, temperature field in this thesis work is always referred to the flow temperature. Anyway, it would be very easy to calculate, by means of a thermal program using an energy balance, the fuel or even the clad temperature starting from the simulated flow value.

Finally, it seems necessary to highlight the fact that, in achieving previous results and conclusions, all the three basic principles of flexibility, repeatability and simplicity, which were mentioned at the beginning of this thesis work, have been followed as best as possible.

The author understands that this work only represents an intermediate step that might support the lines for future further developments, which could be basically organized in three key points in which attention ought to be focused:

1. An extensive analysis on grid sensitivity could be useful for improving the results. Additionally, the mesh could be optimized in order to improve accuracy and convergence, for example by refining or enlarging the inlet region.

2. Another wide area of improvement would be the tuning of the Darcy-Forchheimer coefficients used for the porous media approach. Further experiments in different research facilities are required to understand what pressure drop values can be expected under different circumstances and, hence, what values are acceptable to be used as porosity coefficients.

3. The major area for future work could be the implementation of an implicit treatment for porosity in different solvers. This means diving into the code of those solvers and even discussing about how the velocity should be affected. In this sense, maybe a factor accounting for velocity restriction due to area reduction in some porous zones should be considered.

# Appendix A

# Mesh script

```
//Inputs (in cm)!!
x = 8.55; //x semi-distance of the cylinder
y = 9.87269; //y semi-distance

//Heights of the different sections
Dinlet = 14; //inlet region (setting the inlet flow)
Dorf = 4; //orifices region
Dempty = 4; //empty region
Dsg = 2; //support grid region
Dz1 = 43; //1st region of tube bundle
Dz2 = 16; //1st region of tube bundle
Dz3 = 14; //1st region of tube bundle
Dgs = 2; //grid spacer region
Dact = 29; //active region (x2)
Dfunnel = 4; //element outlet funnel region
Dout = 6; //outlet region (setting the outlet flow)

gs = 2; //gridsize

// x center coordinate of each fuel element
cx={};
// y center coordinate of each fuel element
cy={};

dim = #cx[]-1;

d = -1;

//Arrays to store the surfaces of the inlet, outlet and walls
   boundaries
inlet1[] = {};
inlet2[] = {};
inlet3[] = {};
outlet1[] = {};
outlet2[] = {};
```

```
outlet3[] = {};
walls[] = {};

Macro HexagonalFA

//vertices of bottom hexagon
p1 = newp; Point(p1) = {cx[i], cy[i], 0, gs};
p2 = newp; Point(p2) = {cx[i], cy[i]+y, 0, gs};
p3 = newp; Point(p3) = {cx[i]+x, cy[i]+y/2, 0, gs};
p4 = newp; Point(p4) = {cx[i]+x, cy[i]-y/2, 0, gs};
p5 = newp; Point(p5) = {cx[i], cy[i]-y, 0, gs};
p6 = newp; Point(p6) = {cx[i]-x, cy[i]-y/2, 0, gs};
p7 = newp; Point(p7) = {cx[i]-x, cy[i]+y/2, 0, gs};

//lines of the bottom hexagon
l1 = newl; Line(l1) = {p1, p2};
l2 = newl; Line(l2) = {p1, p4};
l3 = newl; Line(l3) = {p1, p6};
l4 = newl; Line(l4) = {p2, p3};
l5 = newl; Line(l5) = {p3, p4};
l6 = newl; Line(l6) = {p4, p5};
l7 = newl; Line(l7) = {p5, p6};
l8 = newl; Line(l8) = {p6, p7};
l9 = newl; Line(l9) = {p7, p2};

//bottom surfaces
ll1 = newll; Line Loop(ll1) = {l1, l4, l5, -l2};
s1 = news; Plane Surface(s1) = {ll1};
ll2 = newll; Line Loop(ll2) = {l2, l6, l7, -l3};
s2 = news; Plane Surface(s2) = {ll2};
ll3 = newll; Line Loop(ll3) = {l3, l8, l9, -l1};
s3 = news; Plane Surface(s3) = {ll3};

//Bottom mesh
Transfinite Line{l1, l4, l5, -l2} = 2*x/gs;
Transfinite Surface{s1};
Recombine Surface{s1};

Transfinite Line{l2, l6, l7, -l3} = 2*x/gs;
Transfinite Surface{s2};
Recombine Surface{s2};

Transfinite Line{l3, l8, l9, -l1} = 2*x/gs;
Transfinite Surface{s3};
Recombine Surface{s3};

//Inlet extrusion
outinlet[]=
Extrude {0, 0, Dinlet} {
        Surface{s1};
```

```
85          Surface{s2};
86          Surface{s3};
87          Layers{Dinlet/gs};
88          Recombine;
89  };
90
91  //Volume of the inlet region
92  Physical Volume(Sprintf("inlet_%g", i), i+1) = {outinlet[1],
       outinlet[7], outinlet[13]};
93
94  //Orf extrusion
95  outorf[]=
96  Extrude {0, 0, Dorf} {
97          Surface{outinlet[0]};
98          Surface{outinlet[6]};
99          Surface{outinlet[12]};
100         Layers{Dorf/gs};
101         Recombine;
102 };
103
104 p = 1000 + i;
105 //Volume of the orf region
106 Physical Volume(Sprintf("orf_%g", i), p) = {outorf[1], outorf
       [7], outorf[13]};
107
108 //Empty extrusion
109 outempty[]=
110 Extrude {0, 0, Dempty} {
111         Surface{outorf[0]};
112         Surface{outorf[6]};
113         Surface{outorf[12]};
114         Layers{Dempty/gs};
115         Recombine;
116 };
117
118 p = 2000 + i;
119 //Volume of the empty region
120 Physical Volume(Sprintf("empty_%g", i), p) = {outempty[1],
       outempty[7], outempty[13]};
121
122 //Support grid extrusion
123 outsg[]=
124 Extrude {0, 0, Dsg} {
125         Surface{outempty[0]};
126         Surface{outempty[6]};
127         Surface{outempty[12]};
128         Layers{Dsg/gs};
129         Recombine;
130 };
131
```

```
132 p = 3000 + i;
133 //Volume of the support grid region
134 Physical Volume(Sprintf("sg_%g", i), p) = {outsg[1], outsg[7],
        outsg[13]};
135
136 //1st tube bundle extrusion
137 outz1[]=
138 Extrude {0, 0, Dz1} {
139         Surface{outsg[0]};
140         Surface{outsg[6]};
141         Surface{outsg[12]};
142         Layers{Dz1/gs};
143         Recombine;
144 };
145
146 p = 4000 + i;
147 //Volume of the 1st tube bundle region
148 Physical Volume(Sprintf("z1_%g", i), p) = {outz1[1], outz1[7],
        outz1[13]};
149
150 //1st grid spacer extrusion
151 outgs1[]=
152 Extrude {0, 0, Dgs} {
153         Surface{outz1[0]};
154         Surface{outz1[6]};
155         Surface{outz1[12]};
156         Layers{Dgs/gs};
157         Recombine;
158 };
159
160 p = 5000 + i;
161 //Volume of the 1st grid spacer region
162 Physical Volume(Sprintf("gs1_%g", i), p) = {outgs1[1], outgs1
       [7], outgs1[13]};
163
164 //2nd tube bundle extrusion
165 outz2[]=
166 Extrude {0, 0, Dz2} {
167         Surface{outgs1[0]};
168         Surface{outgs1[6]};
169         Surface{outgs1[12]};
170         Layers{Dz2/gs};
171         Recombine;
172 };
173
174 p = 6000 + i;
175 //Volume of the 2nd tube bundle region
176 Physical Volume(Sprintf("z2_%g", i), p) = {outz2[1], outz2[7],
        outz2[13]};
177
```

```
178 //1st active zone extrusion
179 outact1[]=
180 Extrude {0, 0, Dact} {
181         Surface{outz2[0]};
182         Surface{outz2[6]};
183         Surface{outz2[12]};
184         Layers{Dact/gs};
185         Recombine;
186 };
187
188 p = 7000 + i;
189 //Volume of the 1st active zone region
190 Physical Volume(Sprintf("act1_%g", i), p) = {outact1[1],
       outact1[7], outact1[13]};
191
192 //2nd grid spacer extrusion
193 outgs2[]=
194 Extrude {0, 0, Dgs} {
195         Surface{outact1[0]};
196         Surface{outact1[6]};
197         Surface{outact1[12]};
198         Layers{Dgs/gs};
199         Recombine;
200 };
201
202 p = 8000 + i;
203 //Volume of the 2nd grid spacer region
204 Physical Volume(Sprintf("gs2_%g", i), p) = {outgs2[1], outgs2
       [7], outgs2[13]};
205
206 //2nd active zone extrusion
207 outact2[]=
208 Extrude {0, 0, Dact} {
209         Surface{outgs2[0]};
210         Surface{outgs2[6]};
211         Surface{outgs2[12]};
212         Layers{Dact/gs};
213         Recombine;
214 };
215
216 p = 9000 + i;
217 //Volume of the 2nd active zone region
218 Physical Volume(Sprintf("act2_%g", i), p) = {outact2[1],
       outact2[7], outact2[13]};
219
220 //3rd tube bundle extrusion
221 outz3[]=
222 Extrude {0, 0, Dz3} {
223         Surface{outact2[0]};
224         Surface{outact2[6]};
```

```
225         Surface{outact2[12]};
226         Layers{Dz3/gs};
227         Recombine;
228 };
229
230 p = 10000 + i;
231 //Volume of the 3rd tube bundle region
232 Physical Volume(Sprintf("z3_%g", i), p) = {outz3[1], outz3[7],
        outz3[13]};
233
234 //3rd grid spacer extrusion
235 outgs3[]=
236 Extrude {0, 0, Dgs} {
237         Surface{outz3[0]};
238         Surface{outz3[6]};
239         Surface{outz3[12]};
240         Layers{Dgs/gs};
241         Recombine;
242 };
243
244 p = 11000 + i;
245 //Volume of the 3rd grid spacer region
246 Physical Volume(Sprintf("gs3_%g", i), p) = {outgs3[1], outgs3
       [7], outgs3[13]};
247
248 //Element outlet extrusion
249 outfunnel[]=
250 Extrude {0, 0, Dfunnel} {
251         Surface{outgs3[0]};
252         Surface{outgs3[6]};
253         Surface{outgs3[12]};
254         Layers{Dfunnel/gs};
255         Recombine;
256 };
257
258 p = 12000 + i;
259 //Volume of the element outlet funnel region
260 Physical Volume(Sprintf("funnel_%g", i), p) = {outfunnel[1],
       outfunnel[7], outfunnel[13]};
261
262 //Outlet extrusion
263 outout[]=
264 Extrude {0, 0, Dout} {
265         Surface{outfunnel[0]};
266         Surface{outfunnel[6]};
267         Surface{outfunnel[12]};
268         Layers{Dout/gs};
269         Recombine;
270 };
271
```

```
272 p = 13000 + i;
273 //Volume of the outlet region
274 Physical Volume(Sprintf("out_%g", i), p) = {outout[1], outout
       [7], outout[13]};
275
276 Return
277
278 For i In {0 : dim}
279
280 outinlet[] = {0};
281 outorf[] = {0};
282 outempty[] = {0};
283 outsg[] = {0};
284 outz1[] = {0};
285 outgs1[] = {0};
286 outz2[] = {0};
287 outact1[] = {0};
288 outgs2[] = {0};
289 outact2[] = {0};
290 outz3[] = {0};
291 outgs3[] = {0};
292 outfunnel[] = {0};
293 outout[] = {0};
294
295 Call HexagonalFA;
296
297 inlet1[i] = s1;
298 inlet2[i] = s2;
299 inlet3[i] = s3;
300
301 outlet1[i] = outout[0];
302 outlet2[i] = outout[6];
303 outlet3[i] = outout[12];
304
305 If(i==217||i==267||i==296)
306
307         d = d + 1;
308         walls[d] = outinlet[3];
309         d = d + 1;
310         walls[d] = outorf[3];
311         d = d + 1;
312         walls[d] = outempty[3];
313         d = d + 1;
314         walls[d] = outsg[3];
315         d = d + 1;
316         walls[d] = outz1[3];
317         d = d + 1;
318         walls[d] = outgs1[3];
319         d = d + 1;
320         walls[d] = outz2[3];
```

```
321             d = d + 1;
322             walls[d] = outact1[3];
323             d = d + 1;
324             walls[d] = outgs2[3];
325             d = d + 1;
326             walls[d] = outact2[3];
327             d = d + 1;
328             walls[d] = outz3[3];
329             d = d + 1;
330             walls[d] = outgs3[3];
331             d = d + 1;
332             walls[d] = outfunnel[3];
333             d = d + 1;
334             walls[d] = outout[3];
335
336             d = d + 1;
337             walls[d] = outinlet[4];
338             d = d + 1;
339             walls[d] = outorf[4];
340             d = d + 1;
341             walls[d] = outempty[4];
342             d = d + 1;
343             walls[d] = outsg[4];
344             d = d + 1;
345             walls[d] = outz1[4];
346             d = d + 1;
347             walls[d] = outgs1[4];
348             d = d + 1;
349             walls[d] = outz2[4];
350             d = d + 1;
351             walls[d] = outact1[4];
352             d = d + 1;
353             walls[d] = outgs2[4];
354             d = d + 1;
355             walls[d] = outact2[4];
356             d = d + 1;
357             walls[d] = outz3[4];
358             d = d + 1;
359             walls[d] = outgs3[4];
360             d = d + 1;
361             walls[d] = outfunnel[4];
362             d = d + 1;
363             walls[d] = outout[4];
364
365             d = d + 1;
366             walls[d] = outinlet[9];
367             d = d + 1;
368             walls[d] = outorf[9];
369             d = d + 1;
370             walls[d] = outempty[9];
```

```
371           d = d + 1;
372           walls[d] = outsg[9];
373           d = d + 1;
374           walls[d] = outz1[9];
375           d = d + 1;
376           walls[d] = outgs1[9];
377           d = d + 1;
378           walls[d] = outz2[9];
379           d = d + 1;
380           walls[d] = outact1[9];
381           d = d + 1;
382           walls[d] = outgs2[9];
383           d = d + 1;
384           walls[d] = outact2[9];
385           d = d + 1;
386           walls[d] = outz3[9];
387           d = d + 1;
388           walls[d] = outgs3[9];
389           d = d + 1;
390           walls[d] = outfunnel[9];
391           d = d + 1;
392           walls[d] = outout[9];
393
394   EndIf
395
396   //This If structure needs to be repeated for the different
         boundary elements
397
398   EndFor
399
400   dimwalls = #walls[]-1;
401
402   Physical Surface("inlet") = {inlet1[{0:dim}], inlet2[{0:dim}],
          inlet3[{0:dim}]};
403   Physical Surface("outlet") = {outlet1[{0:dim}], outlet2[{0:dim
         }], outlet3[{0:dim}]};
404   Physical Surface("wall") = {walls[{0:dimwalls}]};
```

# Appendix B

# Python program writing *fvOptions* script

```
code01 = """FoamFile
{
        version     2.0;
        format      ascii;
        class       dictionary;
        location    "system";
        object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

//S: safety rods, C: control rods, D: dummy elements, the rest are fuel assemblies
//for inlet layer
inlet_fx 0;
inlet_fy 0;
inlet_fz 0;

inlet_fxS 0;
inlet_fyS 0;
inlet_fzS 0;

inlet_fxC 0;
inlet_fyC 0;
inlet_fzC 0;

inlet_fxD 0;
inlet_fyD 0;
inlet_fzD 0;

//for orf layer
orf_fzI 657.107; orf_fzII 1271.649; orf_fzIII 2402.995; orf_fzIV 4331.425;
```

```
orf_fxI 5e07;    orf_fxII 5e07;     orf_fxIII 5e07;
   orf_fxIV 5e07;
orf_fyI 5e07;    orf_fyII 5e07;     orf_fyIII 5e07;
   orf_fyIV 5e07;

orf_fzS 4208.343;
orf_fxS 5e07;
orf_fyS 5e07;

orf_fzC 4208.343;
orf_fxC 5e07;
orf_fyC 5e07;

orf_fzD 704618.952;
orf_fxD 5e07;
orf_fyD 5e07;

//for empty layer
empty_fz 1.809;
empty_fx 5e07;
empty_fy 5e07;

empty_fzS 4208.343;
empty_fxS 5e07;
empty_fyS 5e07;

empty_fzC 4208.343;
empty_fxC 5e07;
empty_fyC 5e07;

empty_fzD 704618.952;
empty_fxD 5e07;
empty_fyD 5e07;

//for sg layer
sg_fz 208.645;
sg_fx 5e07;
sg_fy 5e07;

sg_fzS 4208.343;
sg_fxS 5e07;
sg_fyS 5e07;

sg_fzC 4208.343;
sg_fxC 5e07;
sg_fyC 5e07;

sg_fzD 704618.952;
sg_fxD 5e07;
sg_fyD 5e07;
```

```
79
80  //for z1 layer
81  z1_fz 12.467;
82  z1_fx 5e07;
83  z1_fy 5e07;
84
85  z1_fzS 4208.343;
86  z1_fxS 5e07;
87  z1_fyS 5e07;
88
89  z1_fzC 4208.343;
90  z1_fxC 5e07;
91  z1_fyC 5e07;
92
93  z1_fzD 704618.952;
94  z1_fxD 5e07;
95  z1_fyD 5e07;
96
97  //for gs1 layer
98  gs1_fz 221.112;
99  gs1_fx 5e07;
100 gs1_fy 5e07;
101
102 gs1_fzS 4208.343;
103 gs1_fxS 5e07;
104 gs1_fyS 5e07;
105
106 gs1_fzC 4208.343;
107 gs1_fxC 5e07;
108 gs1_fyC 5e07;
109
110 gs1_fzD 704618.952;
111 gs1_fxD 5e07;
112 gs1_fyD 5e07;
113
114 //for z2 layer
115 z2_fz 12.467;
116 z2_fx 5e07;
117 z2_fy 5e07;
118
119 z2_fzS 4208.343;
120 z2_fxS 5e07;
121 z2_fyS 5e07;
122
123 z2_fzC 4208.343;
124 z2_fxC 5e07;
125 z2_fyC 5e07;
126
127 z2_fzD 704618.952;
128 z2_fxD 5e07;
```

```
129  z2_fyD 5e07;
130
131  //for act1 layer
132  act1_fz 12.467;
133  act1_fx 5e07;
134  act1_fy 5e07;
135
136  act1_fzS 4208.343;
137  act1_fxS 5e07;
138  act1_fyS 5e07;
139
140  act1_fzC 4208.343;
141  act1_fxC 5e07;
142  act1_fyC 5e07;
143
144  act1_fzD 704618.952;
145  act1_fxD 5e07;
146  act1_fyD 5e07;
147
148  //for gs2 layer
149  gs2_fz 221.112;
150  gs2_fx 5e07;
151  gs2_fy 5e07;
152
153  gs2_fzS 4208.343;
154  gs2_fxS 5e07;
155  gs2_fyS 5e07;
156
157  gs2_fzC 4208.343;
158  gs2_fxC 5e07;
159  gs2_fyC 5e07;
160
161  gs2_fzD 704618.952;
162  gs2_fxD 5e07;
163  gs2_fyD 5e07;
164
165  //for act2 layer
166  act2_fz 12.467;
167  act2_fx 5e07;
168  act2_fy 5e07;
169
170  act2_fzS 4208.343;
171  act2_fxS 5e07;
172  act2_fyS 5e07;
173
174  act2_fzC 4208.343;
175  act2_fxC 5e07;
176  act2_fyC 5e07;
177
178  act2_fzD 704618.952;
```

```
179  act2_fxD 5e07;
180  act2_fyD 5e07;
181
182  //for z3 layer
183  z3_fz 12.467;
184  z3_fx 5e07;
185  z3_fy 5e07;
186
187  z3_fzS 4208.343;
188  z3_fxS 5e07;
189  z3_fyS 5e07;
190
191  z3_fzC 4208.343;
192  z3_fxC 5e07;
193  z3_fyC 5e07;
194
195  z3_fzD 704618.952;
196  z3_fxD 5e07;
197  z3_fyD 5e07;
198
199  //for gs3 layer
200  gs3_fz 221.112;
201  gs3_fx 5e07;
202  gs3_fy 5e07;
203
204  gs3_fzS 4208.343;
205  gs3_fxS 5e07;
206  gs3_fyS 5e07;
207
208  gs3_fzC 4208.343;
209  gs3_fxC 5e07;
210  gs3_fyC 5e07;
211
212  gs3_fzD 704618.952;
213  gs3_fxD 5e07;
214  gs3_fyD 5e07;
215
216  //for funnel layer
217  funnel_fz 114.625;
218  funnel_fx 5e07;
219  funnel_fy 5e07;
220
221  funnel_fzS 4208.343;
222  funnel_fxS 5e07;
223  funnel_fyS 5e07;
224
225  funnel_fzC 4208.343;
226  funnel_fxC 5e07;
227  funnel_fyC 5e07;
228
```

```
229 funnel_fzD 704618.952;
230 funnel_fxD 5e07;
231 funnel_fyD 5e07;
232
233 //for out layer
234 out_fx 0;
235 out_fy 0;
236 out_fz 0;
237
238 out_fxS 0;
239 out_fyS 0;
240 out_fzS 0;
241
242 out_fxC 0;
243 out_fyC 0;
244 out_fzC 0;
245
246 out_fxD 0;
247 out_fyD 0;
248 out_fzD 0;
249 """
250
251 with open('fvOptions', 'w') as f:
252 f.write(code01)
253
254 code02 = """
255 porosity_{name}_{i}
256 {{
257     type      explicitPorositySource;
258     active    yes;
259
260     explicitPorositySourceCoeffs
261     {{
262         type            DarcyForchheimer;
263         selectionMode   cellZone;
264         cellZone        {name}_{i};
265
266         DarcyForchheimerCoeffs
267         {{
268             d   d [0 -2 0 0 0 0 0] (0 0 0);
269             f   f [0 -1 0 0 0 0 0] (${var}_fx{x}{d} ${var}_fy{
                    x}{d} ${var}_fz{x}{d});
270
271             coordinateSystem
272             {{
273                 type    cartesian;
274                 origin  (0 0 0);
275                 coordinateRotation
276                 {{
277                     type    axesRotation;
```

```
                    e1   (1 0 0);
                    e2   (0 1 0);
                }}
            }}
        }}
    }}
}}
"""

n = 0

# fuel elements corresponding to CGI
cg1 = [0, 1, 2, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 17, 18, 19,
   20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
    35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
   49, 50, 51, 52, 53, 54, 55, 56, 56, 57, 58, 59, 60, 61, 62,
    63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76,
   77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90]
# fuel elements corresponding to CGII
cg2 = [91, 92, 94, 96, 97, 98, 100, 102, 103, 104, 106, 108,
   109, 110, 112, 114, 115, 116, 118, 120, 121, 122, 124, 126]
# fuel elements corresponding to CGIII
cg3 = [128, 129, 130, 131, 132, 133, 135, 136, 137, 138, 139,
   140, 142, 143, 144, 145, 146, 147, 149, 150, 151, 152, 153,
    154, 156, 157, 158, 159, 160, 161, 163, 164, 165, 166,
   167, 168]
# fuel elements corresponding to CGIV
cg4 = [127, 134, 141, 148, 155, 162, 172, 173, 174, 180, 181,
   182, 188, 189, 190, 196, 197, 198, 204, 205, 206, 212, 213,
    214]

for layer in range(14):

        if layer==0:
                zone = 'inlet'
                var = zone
        elif layer==1:
                zone = 'orf'
                var = zone
        elif layer==2:
                zone = 'empty'
                var = zone
        elif layer==3:
                zone = 'sg'
                var = zone
        elif layer==4:
                zone = 'z1'
                var = zone
        elif layer==5:
                zone = 'gs1'
```

```
317                 var = zone
318         elif layer==6:
319                 zone = 'z2'
320                 var = zone
321         elif layer==7:
322                 zone = 'act1'
323                 var = zone
324         elif layer==8:
325                 zone = 'gs2'
326                 var = zone
327         elif layer==9:
328                 zone = 'act2'
329                 var = zone
330         elif layer==10:
331                 zone = 'z3'
332                 var = zone
333         elif layer==11:
334                 zone = 'gs3'
335                 var = zone
336         elif layer==12:
337                 zone = 'kout'
338                 var = zone
339         elif layer==13:
340                 zone = 'out'
341                 var = zone
342
343         # safety rod elements
344         safe = [7, 10, 13, 16]
345         # control rod elements
346         control = [93, 95, 99, 101, 105, 107, 111, 113, 117,
               119, 123, 125]
347         ex = [172, 173, 174, 180, 181, 182, 188, 189, 190,
               196, 197, 198, 204, 205, 206, 212, 213, 214]
348         # dummy elements
349         dummy = [x for x in range(169, 297) if x not in ex]
350
351         with open('fvOptions', 'a') as f:
352                 for i in range(0, 297):
353                         if i in safe:
354                                 f.write(code02.format(i=i,d='S
                                        ',x='',name=zone,var=var))
355                         elif i in control:
356                                 f.write(code02.format(i=i,d='C
                                        ',x='',name=zone,var=var))
357                         elif i in dummy:
358                                 f.write(code02.format(i=i,d='D
                                        ',x='',name=zone,var=var))
359                         else:
360                                 if i in cg1 and layer==1:
361                                         x = 'I'
```

```
362                                     elif i in cg2 and layer==1:
363                                             x = 'II'
364                                     elif i in cg3 and layer==1:
365                                             x = 'III'
366                                     elif i in cg4 and layer==1:
367                                             x = 'IV'
368                                     else:
369                                             x = ''
370                                     f.write(code02.format(i=i,d=''
                                           ,x=x,name=zone,var=var))
```

# Appendix C

# Python program writing *setFieldsDict* script

```
code01 = """FoamFile
{
        version     2.0;
        format      ascii;
        class       dictionary;
        location    "system";
        object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
volScalarFieldValue volPow 0
);

//volumetric power in each region [W/m3]

regions
(
"""

with open('setFieldsDict', 'w') as f:
f.write(code01)

code02 = """
zoneToCell
{{
                name "{name}_{i}";
                fieldValues
                (
                volScalarFieldValue volPow {volPow}
                );
}}
```

```
"""

# safety rod elements
safe = [7, 10, 13, 16]
# control rod elements
control = [93, 95, 99, 101, 105, 107, 111, 113, 117, 119, 123, 125]
ex = [172, 173, 174, 180, 181, 182, 188, 189, 190, 196, 197, 198, 204, 205, 206, 212, 213, 214]
# dummy elements
dummy = [x for x in range(169, 297) if x not in ex]

BOC = [] # Array specifying thermal power generation in each element at BOC.
EOC = [] # Array specifying thermal power generation in each element at EOC.

with open('setFieldsDict', 'a') as f:
for i in range(0, 297):
if i in safe:
f.write(code02.format(i=i,volPow=0,name='act1'))
elif i in control:
f.write(code02.format(i=i,volPow=0,name='act1'))
elif i in dummy:
f.write(code02.format(i=i,volPow=0,name='act1'))
else:
f.write(code02.format(i=i,volPow=BOC[i],name='act1'))

with open('setFieldsDict', 'a') as f:
for i in range(0, 297):
if i in safe:
f.write(code02.format(i=i,volPow=0,name='gs2'))
elif i in control:
f.write(code02.format(i=i,volPow=0,name='gs2'))
elif i in dummy:
f.write(code02.format(i=i,volPow=0,name='gs2'))
else:
f.write(code02.format(i=i,volPow=BOC[i],name='gs2'))

with open('setFieldsDict', 'a') as f:
for i in range(0, 297):
if i in safe:
f.write(code02.format(i=i,volPow=0,name='act2'))
elif i in control:
f.write(code02.format(i=i,volPow=0,name='act2'))
elif i in dummy:
f.write(code02.format(i=i,volPow=0,name='act2'))
else:
f.write(code02.format(i=i,volPow=BOC[i],name='act2'))
```

# Appendix D

# Python program to convert reduced pressure file into a normal pressure file

```
code01 = """FoamFile
{
        version     2.0;
        format      ascii;
        class       volScalarField;
        location    "0";
        object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   nonuniform List<scalar>
"""
with open('p_rgh', 'w') as f:
f.write(code01)

mylines = []                        # Declare an empty list named
    mylines.
pred = []                           # Declare an array named pred
    (reduced pressure).
pvalues = []                        # Declare an array named
   pvalues (pressure values once multiplied by density).

rho = 10503                         # Lead density.

with open ('p', 'rt') as myfile: # Open p file for reading
   text data.
for myline in myfile:               # For each line, stored as
   myline,
mylines.append(myline)              # add its contents to mylines
```

```
27
28 init = 22                          # Corresponds to the line
      number -1 where p array starts.
29 dim = int(mylines[20])             # Dimension of the p array
      extracted from line 21 (21-1) and converted to int type.
30
31 for i in range(0,dim):                # For each component
      inside p array,
32 pred.append(float(mylines[i+init]))  # adds coverted to float
      value to pred array
33 pvalues.append(rho*pred[i])          # and stores those values
       multiplied by density in p values.
34
35 code02="""{dim}
36 (
37 """
38
39 with open('p_rgh', 'a') as f:
40 f.write(code02.format(dim=dim))
41
42 with open('p_rgh', 'a') as f:
43 for i in range(0,dim):
44 f.write("%f\n" % pvalues[i])
45
46 code03=""")
47 ;
48
49 boundaryField
50 {
51         inlet
52         {
53                 type            zeroGradient;
54         }
55         outlet
56         {
57                 type            fixedValue;
58                 value           uniform 0;
59         }
60         wall
61         {
62                 type            zeroGradient;
63         }
64 }
65
66 """
67
68 with open('p_rgh', 'a') as f:
69 f.write(code03)
```

# Acronyms

| | |
|---|---|
| **ADS** | Acceleration Driven System |
| **ALFRED** | Advanced Lead-cooled Fast Reactor European Demonstrator |
| **API** | Application Programming Interface |
| **BOC** | Beginning Of Cycle |
| **CFD** | Computational Fluid Dynamics |
| **CR** | Control Rod |
| **DE** | Dummy Element |
| **DNS** | Direct Numerical Simulation |
| **ELFR** | European Lead Fast Reactor |
| **EOC** | End Of Cycle |
| **FA** | Fuel Assembly |
| **GFR** | Gas-cooled Fast Reactor |
| **GIF** | Generation IV International Forum |
| **LBE** | Lead Bismuth Eutectic mixture |
| **LCA** | Life Cycle Analysis |
| **LEADER** | Lead-cooled European Advanced Demonstrator Reactor |
| **LES** | Large Eddy Simulation |
| **LFR** | Lead-cooled Fast Reactor |
| **LOCA** | Loss Of Coolant Accident |
| **LWR** | Light Water Reactor |
| **MOX** | Mixed Oxide Fuel |
| **MPI** | Message Passing Interface |
| **MSR** | Molten Salt Reactor |
| **MYRRHA** | Multi-purpose hYbrid Reaserch Reactor for High-tech Applications |
| **NEA** | Nuclear Energy Agency |
| **OpenFOAM** | Open source Field Operation And Manipulation |
| **PISO** | Pressure Implicit with Splitting of Operators |
| **RANS** | Reynolds-Averaged Navier-Stokes |

| | |
|---|---|
| **RAS** | Reynolds-Averaged Simulation |
| **RSM** | Reynolds Stress Models |
| **SCWR** | Supercritical Water Reactor |
| **SFR** | Sodium-cooled Fast Reactor |
| **SIMPLE** | Semi-Implicit Method for Pressure-Linked Equations |
| **SR** | Safety Rod |
| **URANS** | Unsteady Reynolds-Averaged Navier-Stokes |
| **VHTR** | Very High Temperature Reactor |

# Bibliography


[1] Angiolini, M., Agostini, P., Bassini, S., Fabbri, F., Di Fonzo, F. and Tarantino, M., 2017. Material issues in heavy liquid metal cooled systems. In: 1st IAEA workshop on "Challenges for coolants in fast spectrum system: chemistry and materials", Vienna, Austria.

[2] Batta, A. and Class, A., 2017. CFD analysis of pressure drop across grid spacers in rod bundles compared to correlations and heavy liquid metal experimental data. *Nuclear Engineering and Design*, 312, pp.121-127.

[3] Blasius, P. R. H., 1913. Das aehnlichkeitsgesetz bei reibungsvorgangen in flussigk eiten. *Forschungsheft*, 131, pp.1-41.

[4] BP, 2020. *Statistical Review of World Energy, 69th Edition.* Available at: https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html

[5] Cevolani, S., 1995. ENEA thermohydraulic data base for the advanced water cooled reactor analysis, Tech. Rep., 1st Research Co-ordination Meeting of IAEA CRP on Termohydraulic Relationships for Advanced Water Cooled Reactors.

[6] Chen, S., Todreas, N. E. and Nguyen, N. T., 2014. Evaluation of existing correlations for the prediction of pressure drop in wire-wrapped hexagonal array pin bundles. *Nuclear Engineering and Design*, 267, pp.109-131.

[7] Chenu, A., Mikityuk, K. and Chawla, R., 2011. Pressure drop modeling and comparisons with experiments for single- and two-phase sodium flow. *Nuclear Engineering and Design*, 241(9), pp.3898-3909.

[8] Cigarini, M., and Dalle Donne M., 1988. Thermohydraulic optimization of homogeneous and heterogeneous advanced pressurized water reactors. *Nuclear Technology*, 80(1), pp.107-132.

[9] Dahl, P. M. and Su, J., 2017. Simulation in CFD of a pebble bed - Advanced high temperature reactor core using OpenFOAM. In: International Nuclear Atlantic Conference - INAC 2017, Belo Horizonte, Brazil.

[10] De Bruyn, D., Abderrahim, H., Baeten, P. and Leysen, P., 2015. The MYRRHA ADS Project in Belgium Enters the Front End Engineering Phase. *Physics Procedia*, 66, pp.75-84.

[11] Domaingo, A., Langmayr, D., Somogyi, B. and Almbauer, R., 2016. A Semi-implicit Treatment of Porous Media in Steady-State CFD. *Transport in Porous Media*, 112(2), pp.451-466.

[12] Epiney, A., Mikityuk, K. and Chawla, R., 2010. TRACE qualification via analysis of the EIR gas-loop experiments with smooth rods. *Annals of Nuclear Energy*, 37(6), pp.875-887.

[13] Generation IV systems, 2021. *Generation IV International Forum*. [online] Available at: https://www.gen-4.org/gif/jcms/c_9260/public [Accessed 26 January 2021].

[14] Gmsh.info, 2021. *Gmsh: a three-dimensional finite element mesh generator withbuilt-in pre- and post-processing facilities*. [online] Available at: https://gmsh.info/ [Accessed 12 February 2021].

[15] Grasso, G., Petrovich, C., Mattioli, D., Artioli, C., Sciora, P., Gugiu, D., Bandini, G., Bubelis, E. and Mikityuk, K., 2014. The core design of ALFRED, a demonstrator for the European lead-cooled reactors. *Nuclear Engineering and Design*, 278, pp.287-301.

[16] Grasso, G., Petrovich, C., Mikityuk, K., Mattioli, D., Manni, F. and Gugiu, D., 2013. Demonstrating the effectiveness of the European LFR concept: the ALFRED core design. In: Proceedings of the International Conference on Fast Reactors and Related Fuel Cycles: Safe Technologies and Sustainable Scenarios (FR 13), Paris, France.

[17] Hafsteinsson, H. E., 2009. Porous Media in OpenFOAM. Chalmers University of Technology, Gothemburg.

[18] Hayes, A., Khan, J., Shaaban, A. and Spearing, I., 2008. The thermal modeling of a matrix heat exchanger using a porous medium and the thermal non-equilibrium model. *International Journal of Thermal Sciences*, 47(10), pp.1306-1315.

[19] Hörmann, T., Lechner, B., Puntigam, W., Moshammer, T. and Almbauer, R. A., 2005. Numerical and experimental investigation of flow and temperature fields around automotive cooling systems. *SAE Technical Paper*, 2005-01-2006.

[20] Idel'Chik, I., 2008. *Handbook of hydraulic resistance*. Redding, CT: Begell House.

[21] Lamberts, D., 2011. Control rods and safety rods. In: SCK · CEN/PSD, LEADER Project Meeting, Bologna, Italy.

[22] Launder, B., Reece, G. and Rodi, W., 1975. Progress in the development of a Reynolds-stress turbulence closure. *Journal of Fluid Mechanics*, 68(3), pp.537-566.

[23] Launder, B. and Spalding, D., 1974. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3(2), pp.269-289.

[24] Leppänen, J., Hovi, V., Ikonen, T., Kurki, J., Pusa, M., Valtavirta, V. and Viitanen, T., 2015. The Numerical Multi-Physics project (NUMPS) at VTT Technical Research Centre of Finland. *Annals of Nuclear Energy*, 84, pp.55-62.

[25] Lorusso, P., Bassini, S., Del Nevo, A., Di Piazza, I., Giannetti, F., Tarantino, M. and Utili, M., 2018. GEN-IV LFR development: Status & perspectives. *Progress in Nuclear Energy*, 105, pp.318-331.

[26] Mikityuk, K. *et al.*, March 2013. Thermal-hydraulic assessment of the EDTR core. LEADER DOC056-2012, rev. 0, LEADER project.

[27] Moukalled, F., Mangani, L. and Darwish, M., 2016. *The Finite Volume Method In Computational Fluid Dynamics*. Cham: Springer.

[28] NEA, 2015. *Handbook on Lead-bismuth Eutectic Alloy and Lead properties, Materials Compatibility, Thermal-hydraulics and Technologies*. OECD/NEA, No. 7268.

[29] OpenFOAM, The OpenFOAM Foundation, 2019. *User Guide, version 7*. Available at: https://openfoam.org

[30] Openfoam.com, 2021. *Openfoam - Official Home Of The Open Source Computational Fluid Dynamics (CFD) Toolbox*. [online] Available at: https://www.openfoam.com/ [Accessed 25 January 2021].

[31] Our World in Data, 2021. *Electricity Mix*. [online] Available at: https://ourworldindata.org/electricity-mix [Accessed 25 January 2021].

[32] Our World in Data. 2021. *Nuclear Energy*. [online] Available at: https://ourworldindata.org/nuclear-energy [Accessed 25 January 2021].

[33] Pacio, J., Daubner, M., Fellmoser, F., Litfin, K., Marocco, L., Stieglitz, R., Taufall, S. and Wetzel, T., 2014. Heavy-liquid metal heat transfer experiment in a 19-rod bundle with grid spacers. *Nuclear Engineering and Design*, 273, pp.33-46.

[34] Pelowitz, D. B., Durkee, J. W., Elson, J. S., Fensin, M. L., Hendricks, J. S., James, M. R., Johns, R. C., McKinney, G. W., Mashnik, S. G., Verbeke, J. M., Waters, L. S. and Wilcox, T. A., 2011. MCNPX 2.7.E Extensions. Technical Report LA-UR-11-01502, Los Alamos.

[35] Pérez Mañes, J., Sánchez Espinoza, V., Chiva, S. and Stieglitz, R., 2014. A New Coupled CFD/Neutron Kinetics System for High Fidelity Simulations of LWR Core Phenomena: Proof of Concept. *Science and Technology of Nuclear Installations*, 2014, pp.1-13.

[36] Pioro, I., 2016. *Handbook of Generation IV Nuclear Reactors*. Woodhead Publishing.

[37] Ponciroli, R., Cammi, A., Della Bona, A., Lorenzi, S. and Luzzi, L., 2015. Development of the ALFRED reactor full power mode control system. *Progress in Nuclear Energy*, 85, pp.428-440.

[38] Rehme, K., 1973. Pressure Drop Correlations for Fuel Element Spacers. *Nuclear Technology*, 17(1), pp.15-23.

[39] Rehme, K., 1973. Simple method of predicting friction factors of turbulent flow in non-circular channels. *International Journal of Heat and Mass Transfer*, 16, pp.933-950.

[40] Rimpault, G., Plisson, D., Tommasi, J., Jacqmin, R., Rieunier, J. M., Verrier, D. and Biron, D., 2002. The ERANOS code and data system for fast reactor neutronic analyses. In: Proceedings of PHYSOR 2002 - International Conference on the New Frontiers of Nuclear Technology: Reactor Physics, Safety and High-Performance Computing, Seoul, Korea.

[41] Roelofs, F., Uitslag-Doolaard, H., Dovizio, D., Mikuz, B., Shams, A., Bertocchi, F., Rohde, M., Pacio, J., Di Piazza, I., Kennedy, G., Van Tichelen, K., Obabko, A. and Merzari, E., 2019. Towards validated prediction with RANS CFD of flow and heat transport in a wire-wrap fuel assembly. Nuclear Engineering and Design, 353, p.110273.

[42] Saidi, K. and Omri, A., 2020. Reducing $C0_2$ emissions in OECD countries: Do renewable and nuclear energy matter? *Progress in Nuclear Energy*, 126, p.103425.

[43] Savatteri, C., Warnsing, R., Loens, J. and Kottowski, H., 1986. Results and comparison of dry-out in grid and wire spaced bundles at single- and two-phase flow. In: *Proc. 12th Liquid Metal Boiling Working Group (LMBWG)*, pp.164-190.

[44] Schikorr, M., Bubelis, E., Mansani, L. and Litfin, K., 2010. Proposal for pressure drop prediction for a fuel bundle with grid spacers using Rehme pressure drop correlations. *Nuclear Engineering and Design*, 240(7), pp.1830-1842.

[45] Shirizadeh, B. and Quirion, P., 2020. Low-carbon options for the French power sector: What role for renewables, nuclear energy and carbon capture and storage? *Energy Economics*, p.105004.

[46] Sobolev, V., Schuurmans, P. and Benamati, G., 2008. Thermodynamic properties and equation of state of liquid lead and lead–bismuth eutectic. *Journal of Nuclear Materials*, 376(3), pp.358-362.

[47] Spalart, P. and Allmaras, S., 1994. A one-equation turbulence model for aerodynamic flows. *La Recherche Aerospatiale*, 1, 5-21.

[48] U.S. DOE Nuclear Energy Research Advisory Committee, Generation IV International Forum, 2002. *A technology roadmap for Generation IV nuclear energy systems*. USA: U.S. DOE Nuclear Energy Research Advisory Committee, Generation IV International Forum. Report No.: GIF-002-00.

[49] U.S. Geological Survey, 2021. *Mineral Commodity Summaries 2021*. [online] Available at: https://pubs.usgs.gov/periodicals/mcs2021/mcs2021.pdf [Accessed 1 February 2021].

[50] Versteeg, H. and Malalasekera, W., 2008. *An introduction to computational fluid dynamics*. Harlow [u.a.]: Pearson/Prentice Hall.

[51] Vog, P., Markfort, D. and Ruppert, E., 1971. A thermal-hydraulic analysis for fuel elements with liquid metal cooling. In: *International Conference of Heat and Mass Transfer*.

[52] Wilcox, D., 1998. *Turbulence modeling for CFD*. La Cañada, Calif.: DCW Industries.