



**Politecnico
di Torino**

POLITECNICO DI TORINO

Master's Degree in
Sustainable Nuclear Energy

Master's Degree Thesis

**Development of multi-scale interfaces for the
thermal-hydraulic module of the FRENETIC code**

Supervisors

Prof. Roberto Bonifetto, PhD

Ing. Nicolò Abrate, PhD

Ing. Danilo Caterino

Candidate

Giorgia Versino

s347090

Academic Year 2025-2026

Abstract

Liquid Metal Fast Reactors (LMFRs) represent one of the most promising concepts within the Generation IV framework. Fission reactor core design is an inherently complex, multi-physics challenge, making the development of reliable and computationally efficient simulation tools essential. These tools support preliminary design phases and accurately predict core behavior during operational and off-normal transients. The unique characteristics of LMFRs, specifically their fast neutron spectrum and the high thermal inertia of liquid metal coolants, yield dynamic behaviors that differ fundamentally from traditional Light Water Reactors, necessitating dedicated computational codes.

In this context, the Fast REactor Neutronics/Thermal-hydraulicS (FRENETIC) code was developed at Politecnico di Torino as an advanced multi-physics tool for the full-core simulation of liquid-metal-cooled fast reactors. FRENETIC performs coupled neutronic and thermal-hydraulic (TH) calculations for both steady-state and transient scenarios. It achieves a low computational burden by employing optimized physical models, where the TH module conceptualizes the hexagonal core assemblies as parallel channels. These are solved using a 1D axial formulation radially coupled to neighbouring assemblies to capture 3D heat transfer effects.

The primary objective of this thesis is the extensive refactoring, enhancement, and validation of the FRENETIC TH module. The original monolithic architecture was completely restructured to significantly improve computational performance, portability, and flexibility, with special emphasis on rigorous memory management and code modularity. This deeply optimized structure streamlines future developments and facilitates coupling with system codes. Specifically, a novel Python-driven explicit coupling interface was developed to link the sub-channel solver with the ATHLET system-level code. This architecture leverages event-driven callback functions and a macro-step synchronization strategy, allowing FRENETIC to align its physical time with the system master clock while ensuring numerical stability.

To validate the standalone modernized FRENETIC TH module against experimental data and benchmark it against the legacy version, the updated tool was applied to simulate the CIRCE experimental facility core. Furthermore, to evaluate the performance of the novel coupling architecture between ATHLET and FRENETIC, a simplified model of the ALFRED reactor was selected as a test case, enabling a direct comparison with the results in the framework of the IAEA benchmark. By successfully reproducing these experimental

data, this work confirms the combined capability of the refactored FRENETIC code and ATHLET to reliably capture the complex passive safety dynamics inherent to liquid-metal fast reactors.

Acknowledgements

Thanks to my family that supported me even when it was not so possible. Thanks to people that was near to me and gave me new possibilities during this last year.

Contents

List of Tables	viii
List of Figures	x
1 Introduction	1
1.1 Generation II, III, and IV Reactors	1
1.2 Liquid Metal Fast Cooled Reactors	2
1.3 The Advanced Lead-cooled Fast Reactor European Demonstrator (ALFRED)	5
1.4 Reactor Core modeling	7
1.4.1 General features of the core modeling problem	7
1.4.2 FRENETIC & ATHLET Codes	9
1.5 Aim of the thesis	10
1.6 Thesis outline	10
2 Computational Tools for Reactor Analysis	13
2.1 The FRENETIC Thermal-Hydraulic Module	13
2.1.1 Code Architecture and Modularity	14
2.1.2 Calculation Domain and Core Modeling Approach	15
2.1.3 Spatial Discretization and Numerical Method	16
2.1.4 Boundary Conditions and Peripheral Assemblies	17
2.1.5 Physical Models and Governing Equations	18
2.2 The ATHLET Code	19
2.2.1 TFD Module: physical models and governing equations	20
2.2.2 The Heat Conduction and Heat Transfer Module (HECU)	23
2.2.3 The Neutron Kinetics Model (NEUKIN)	24
2.2.4 The General Control Simulation Module (GCSM)	25
2.2.5 Coupling Strategy Between FRENETIC-TH and ATHLET	26
3 Refactoring of the Thermal-Hydraulic Module Main Code main.f90	29
3.1 New Architecture and Modularity	30
3.2 Power Variable Standardization: <code>powtot0</code> vs. <code>Pow0</code>	31
3.3 Impact of Code Refactoring on the Validation Process	32

4	New code structure to account for the thermal expansion feedback	35
4.1	Mathematical Framework and Finite Volume Suitability	35
4.2	Dynamic Geometrical Updating Strategy	37
4.2.1	Code Verification	37
4.3	Thermo-Mechanical Coupling Implementation	39
4.3.1	Data Exchange Strategy and Modularity	39
4.3.2	Execution Sequence and Transient Integration	39
5	TH Standalone Cases Analysis & Results	41
5.1	1D Standalone Transient	43
5.1.1	Results & Post-processing	43
5.2	3D Standalone Transient	44
5.2.1	Results & Post-processing	45
5.3	3D Power Transient	47
5.3.1	Results & Post-processing	47
5.4	1D Standalone CIRCE: ICE Facility Validation	50
5.4.1	Results & Post-processing	51
6	Coupling FRENETIC-TH and ATHLET	55
6.1	Python Library for ATHLET	55
6.2	Modification of the FRENETIC <code>main.py</code> Script	56
6.2.1	The Concept of Callback Functions (Hooks)	57
6.2.2	Code Architecture and Execution Flow	60
6.3	Architecture of the Thermal-Hydraulic Driver Interface (<code>driver.py</code>)	62
6.4	Architecture of the API Thermal-Hydraulic Driver Interface of FRENETIC (<code>TH_driver_API.f90</code>)	64
7	Coupling architecture FRENETIC-ATHLET Test Cases & Results	67
7.1	<code>TH_1D_standalone_transient</code>	67
7.2	ALFRED Reactor	72
7.2.1	ATHLET Model	72
7.2.2	FRENETIC Model	75
7.2.3	Coupling	77
7.2.4	FRENETIC Grid Independence Analysis	81
8	Conclusions and Future Developments	87
A	ATHLET System Nodalization Study	91
B	Python Master Driver Script	95

C	Material properties	103
C.1	Coolant properties	103
C.2	Fuel rod properties	103
C.3	Wrapper properties	104
D	Constitutive relations	105
D.1	Friction factor	105
D.2	Nusselt number	106
E	Acronyms	109

List of Tables

4.1	<i>Output obtained from the test case TH_1D_standalone_flowering.</i>	38
7.1	<i>Boundary Conditions and setup parameters in the ATHLET domain.</i>	69
7.2	<i>Boundary Conditions and setup parameters in the FRENETIC domain.</i>	70
7.3	<i>Variables dynamically exchanged between the two codes at synchronization intervals.</i>	70
7.4	<i>ATHLET Steady-state operational parameters.</i>	74
7.5	<i>Average mass flow rate per channel for the ALFRED core configuration [1].</i>	75
7.6	<i>FRENETIC Steady-state operational parameters.</i>	77
7.7	<i>Variables dynamically exchanged and monitored between the two codes at synchronization intervals.</i>	78
7.8	<i>Relative error analysis for the spatial grid independence study.</i>	83
D.1	<i>Coefficients for bare rod subchannel friction factor constants in hexagonal array.</i>	106

List of Figures

1.1	<i>ALFRED Reactor Block: vertical view [2].</i>	5
1.2	<i>ALFRED Core Layout.</i>	7
2.1	<i>Schematic of the bypass flow domain [3].</i>	16
2.2	<i>Spatial nomenclature for mesh discretization [4].</i>	17
2.3	<i>Detailed functional and folder architecture of the FRENETIC code.</i>	19
2.4	<i>Schematic representation of the ATHLET 1D computational domain.</i>	23
2.5	<i>Detailed modular architecture of the ATHLET system code.</i>	26
2.6	<i>Schematic of the multi-scale explicit coupling between ATHLET and FRENETIC via Python Master Driver and Text-File exchange.</i>	28
3.1	<i>Modular restructuring of the FRENETIC main driver for the thermal-hydraulic module.</i>	31
5.1	<i>1D Radial Temperature variation at $z = 0.80$ m.</i>	44
5.2	<i>1D Temperature Profiles.</i>	44
5.3	<i>Radial Temperatures Variation in a 3D domain.</i>	46
5.4	<i>Axial Temperature Profiles.</i>	46
5.5	<i>Radial Temperature Variation in a 3D domain with power transient.</i>	48
5.6	<i>Axial Temperature Profiles in the power transient.</i>	49
5.7	<i>Temperature Evolution during the transient.</i>	49
5.8	<i>Cross-section of the Integral Circulation Experiment (ICE) heat source bundle [5].</i>	51
5.9	<i>Radial Temperature Variation at different $z = 0.80$ m.</i>	52
5.10	<i>Temperature Profiles evolution in time.</i>	52
6.1	<i>Sketch of the Simple Loop used as the test bench.</i>	59
6.2	<i>Workflow of the explicit coupling strategy and Event-Driven synchronization between ATHLET and FRENETIC.</i>	62
7.1	<i>Sketch of the thermal-hydraulic loop for the test-case.</i>	69
7.2	<i>Time evolution of the exchanged variables during the coupled transient (the acronym at the beginning indicates the domain to which the variable belongs, while the acronym at the end indicates the domain from which it comes).</i>	71

7.3	<i>ATHLET detailed nodalization scheme for the ALFRED test case (reference in Chapter 1 in Fig. 1.1).</i>	74
7.4	<i>ALFRED core configuration per cooling groups [1].</i>	76
7.5	<i>Time evolution of the exchanged variables during the coupled transient for the ALFRED case (the acronym at the beginning indicates the destination domain, while the acronym at the end indicates the origin domain).</i>	78
7.6	<i>Time evolution of the exchanged variables during the coupled transient for the ALFRED case (the acronym at the beginning indicates the destination domain, while the acronym at the end indicates the origin domain).</i>	79
7.7	<i>Time Evolution (10 s) of the Maximum Coolant Temperature with different meshes.</i>	82
7.8	<i>Time Evolution (10 s) of the Maximum Cladding Temperature with different meshes.</i>	82
7.9	<i>Time Evolution (10 s) of the Maximum Coolant Temperature taken from FRENETIC using different meshes.</i>	84
7.10	<i>Time Evolution (10 s) of the Maximum Cladding Temperature taken from FRENETIC using different meshes.</i>	84
A.1	<i>Nodalization Study on Temperature.</i>	91
A.2	<i>Nodalization Study on Pressure.</i>	92
A.3	<i>Nodalization Study on the Computational Cost.</i>	93

Chapter 1

Introduction

Nuclear energy (currently based on Generation II and III technologies) accounts for approximately 10% of global electricity production and has the opportunity to play a very significant role thanks to the high capacity factors achievable.

These reactors often guarantee factors exceeding 90% with very long operating cycles, making them extremely suitable for meeting base-load requirements. Furthermore, these plants feature operational carbon dioxide emissions that are essentially negligible compared to alternative power generation systems [6].

Despite these successes, new concepts belonging to a new generation of fission reactors are currently under study to ensure long-term viability [6]. In fact, Generation IV reactors will have to meet very stringent standards compared to current technologies, aiming primarily at increased sustainability through the adoption of a closed fuel cycle. The main advantages of these innovative systems over current technologies include a far better use of resources and a reduced volume of final waste.

1.1 Generation II, III, and IV Reactors

Currently, the vast majority of nuclear power plants in operation worldwide belong to Generation II and Generation III. These are predominantly Light Water Reactors (LWRs), including Pressurized Water Reactors (PWRs) and Boiling Water Reactors (BWRs), which use water as both coolant and neutron moderator. While these reactors have successfully provided reliable, low-carbon base-load electricity for decades, they present certain limitations regarding the fuel cycle and waste management. Those are particularly related to the fact that because of their thermal spectrum, the uranium utilization is relatively low, and long-lived radioactive wastes, including minor actinides, are produced.

To overcome these challenges and ensure long-term sustainability, Generation IV reactors are currently under intensive research and development. To satisfy global energy expectations, Generation IV designs must meet significantly more stringent standards compared to current operational technologies. In preparing the technical roadmap toward the realization of these advanced concepts, the Generation IV International Forum (GIF) [7]

established a comprehensive set of goals distributed across four broad areas:

1. **Sustainability:** Enhanced sustainability is achieved primarily through the adoption of a closed fuel cycle. The main advantages of such systems lie in their optimized resource utilization and a substantial reduction in the volume of high-level final waste. Advanced waste management strategies include the transmutation of long-lived nuclides, such as minor actinides, alongside cost-effective decay-heat management and flexible interim storage solutions. This drastically limits final waste radiotoxicity, thereby improving the protection of public health and the environment.
2. **Economics:** In order to be competitive within the global energy market, Generation IV nuclear systems must offer a clear life-cycle cost advantage over other energy sources, while maintaining a financial risk profile comparable to other major energy infrastructure projects. Achieving a high thermal efficiency is also a key target to maximize economic returns.
3. **Safety and Reliability:** Generation IV plants are designed to guarantee highly safe and reliable operations. By extensively incorporating passive safety features, these designs significantly minimize the likelihood and degree of reactor core damage. Furthermore, they aim to inherently eliminate the technical justification for offsite emergency responses in the event of an accident.
4. **Proliferation Resistance and Physical Protection:** To mitigate proliferation risks, design architectures employing fuel recycling explicitly avoid the separation of plutonium from other actinides. Additionally, they incorporate intrinsic features to reduce the proliferation of nuclear materials at every stage of the fuel cycle. Simultaneously, these nuclear power plants must implement enhanced physical protection systems against potential acts of terrorism and sabotage.

With these targets in mind, the Generation IV International Forum (GIF) selected six specific reactor technologies for further research and development. These concepts comprise the Gas-Cooled Fast Reactor (GFR), the Lead-Cooled Fast Reactor (LFR), the Molten Salt Reactor (MSR), the Sodium-Cooled Fast Reactor (SFR), the SuperCritical-Water-Cooled Reactor (SCWR), and the Very-High-Temperature Reactor (VHTR) [7]. Notably, the first four of these innovative concepts are characterized by a fast neutron spectrum, the physics and advantages of which will be detailed in the next section.

1.2 Liquid Metal Fast Cooled Reactors

Fast Neutron Spectrum Operating in a higher neutron energy range enables the use of U-238 and Pu-239 as primary fissile materials. Because Pu-239 yields more neutrons per fission than U-235, it is easier to sustain a chain reaction despite the lower interaction cross-sections typical of a fast spectrum. To maintain this fast spectrum, a non-moderating

coolant, such as a gas or a liquid metal (LM), is essential.

Liquid metals are characterized by minimal neutron absorption and moderation, making them ideal for breeder reactor designs. Breeding reduces the overall inventory of fissile material required, as the core only needs a small reactivity surplus at the beginning of the cycle. Furthermore, fast spectrum reactors generate significantly less long-lived nuclear waste. Minor actinides can be utilized as additional fissionable material; they are preferentially transmuted at high neutron energies, whereas in Light Water Reactors (LWRs), neutron capture often transforms them into higher actinides [8].

While this closed fuel cycle enhances sustainability, the fast spectrum does negatively impact kinetic parameters. The low moderation decreases the reactivity swing and prevents massive reactivity insertions from changes in coolant state, reducing risks during accidental control rod withdrawals. Additionally, the high thermal inertia and negative reactivity feedbacks of LM systems afford operators more time for corrective action. Even during an unprotected transient (such as ULOF, Unprotected Loss of Flow; UTOP, Unprotected Transient Overpower; ULOHS, Unprotected Loss of Heat Sink), small positive reactivity feedbacks are offset by strong negative radial core expansion, which limits power excursions [9]. Conversely, the short neutron lifetime, combined with a lower delayed neutron fraction, results in faster reactor dynamics compared to thermal reactors. Thermal feedback behaviors also differ; for instance, the Doppler coefficient in a fast reactor is proportional to $1/T$, whereas in thermal reactors it is typically proportional to $1/\sqrt{T}$ [10].

The fast neutron spectrum and elevated neutron flux inherently increase the irradiation damage sustained by the core structures, a degradation typically quantified in Displacements Per Atom (DPA). Consequently, the primary coolant must be appropriately selected to provide excellent intrinsic radiation shielding, a property for which heavy liquid metals such as lead are exceptionally well-suited. The core structural materials must be rigorously qualified to ensure not only a sufficient operating lifetime under extreme irradiation but also a robust thermochemical compatibility with the highly corrosive coolant and erosive environment.

LM Characteristics and Core Design Among Generation IV concepts, Lead Fast Reactors (LFRs) offer distinct advantages. The massive atomic weight difference between neutrons and LM coolant atoms produces a harder neutron spectrum than any other design, minimizing parasitic absorption. LMs also provide exceptional thermal-hydraulic benefits.

The high thermal conductivity and heat capacity of metal coolants facilitate efficient heat removal even at very low flow velocities, mitigating the risk of local hot spots. Furthermore, the high boiling point of LMs allows reactors to operate at elevated temperatures, boosting plant efficiency without the need to pressurize the primary system. This dramatically reduces structural complexity and cost. In accident scenarios, the substantial margin to boiling virtually eliminates the risk of core voiding. The immense thermal inertia provided by lead’s high latent heat and thermal capacity guarantees a long grace period for

operator action during a loss of heat sink, reducing reliance on offsite emergency responses [10].

LFRs can be configured in either loop or pool designs. Loop designs introduce pipework that increases the risk of primary circuit breaches. Pool designs eliminate this risk by enclosing the primary system within a larger vessel, though they present different maintenance challenges: the opaque coolant requires X-ray or ultrasonic equipment for inspections, and the high coolant density creates strong positive buoyancy, requiring all submerged components to be securely anchored.

Because LMs are solid at room temperature, heating systems are required during startup, maintenance, and refueling to prevent unplanned freezing and flow blockage. Additionally, the high density and shear stress of heavy LMs impose significant mechanical loads and potential erosion on structural components.

Neutronically, low coolant moderation allows for greater spacing between fuel pins, reducing pressure drops and increasing the coolant volume fraction without severe reactivity penalties. Fuel pins are typically separated by spacer grids or helical wire wraps. While wire wraps provide robust lattice support and prevent cladding contact, they complicate thermal-hydraulic modeling by altering pressure drops, surface heat transfer, and transverse mixing between sub-channels [10]. Fuel assemblies are usually enclosed in hexagonal wrappers to provide structural support and flow separation, allowing for precise control of radial temperature distributions. However, wrappers penalize the neutron economy by introducing more steel into the core.

A paramount safety advantage of lead is its chemical inertness. Unlike sodium or water, lead does not react violently with air or water, eliminating fire and explosion risks in the nuclear island. This allows air and water to be used safely as ultimate heat sinks for Decay Heat Removal (DHR) systems, enabling passive natural circulation and plant simplification. Furthermore, lead naturally traps and retains hazardous fission products (such as iodine and cesium), acting as an inherent barrier against radioactive release.

The primary drawback of heavy liquid metals is their corrosiveness, particularly at high temperatures: above 550°C, typical cladding materials suffer severe dissolution compromising the structural integrity [7]. Because traditional LWR Zircaloy lacks both high-temperature strength and LM compatibility, designers rely on specialized steels. Corrosion mitigation strategies include applying external coatings or using silicon, chromium, and aluminum-alloyed steels to promote a protective oxide layer. This requires precise control of the dissolved oxygen within the coolant to maintain the barrier without forming flow-blocking oxide sediments.

The choice of structural steel involves a trade-off:

- Ferritic-Martensitic steels (e.g., T91): Offer lower thermal stresses, excellent resistance to irradiation creep, and high tolerance to LM corrosion.
- Austenitic steels (e.g., 15-15Ti): Exhibit less neutron embrittlement and thermal creep, though they are more prone to void swelling.

1.3 The Advanced Lead-cooled Fast Reactor European Demonstrator (ALFRED)

The Advanced Lead Fast Reactor European Demonstrator (ALFRED) (Fig. 1.1) represents a cornerstone initiative within the framework of Generation IV nuclear systems, specifically aimed at demonstrating the technical and economic viability of Lead-cooled Fast Reactors (LFRs). Developed under European research programs such as the LEADER project, ALFRED is conceptualized as a pool-type fast reactor with a nominal thermal power rating of 300 MW_{th} [2]. The primary objective of this demonstrator is to bridge the gap between small-scale experimental facilities and future commercial-scale industrial LFR units, proving the feasibility of heavy liquid metal technologies for power generation, sustainability, and enhanced safety.

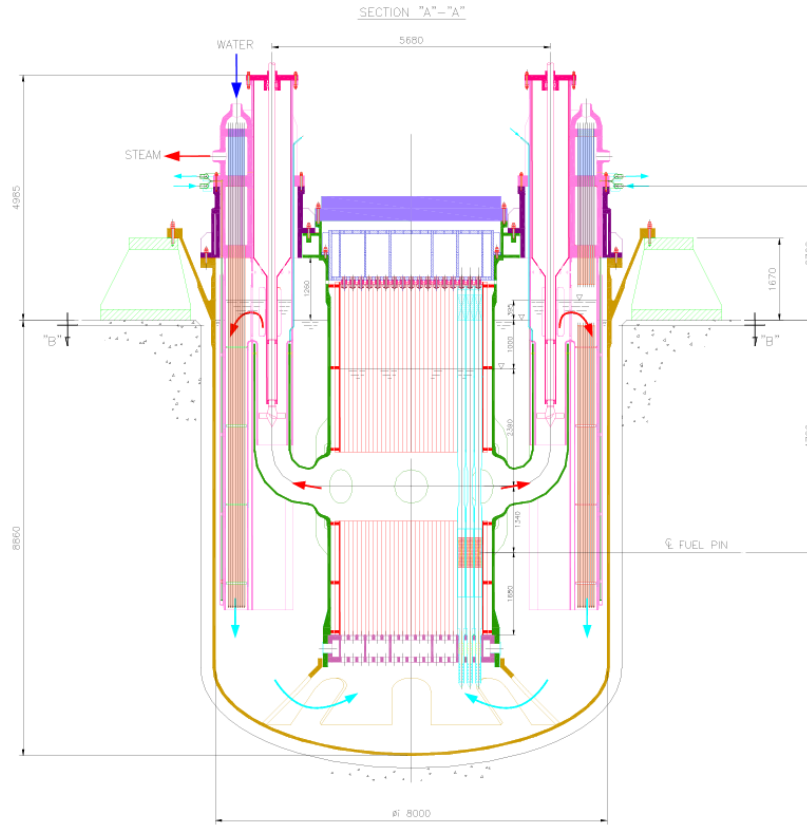


Figure 1.1: *ALFRED Reactor Block: vertical view [2].*

The selection of pure liquid lead (Pb) as the primary coolant dictates the fundamental thermal-hydraulic and structural features of the reactor. Lead exhibits unique intrinsic properties that significantly enhance the safety profile of the plant compared to traditional light water reactors or sodium-cooled fast systems:

- a) *High Boiling Point* ($> 1740\text{ }^{\circ}\text{C}$): this virtually eliminates the risk of coolant boiling or vaporization during operational occurrences or severe accidental transients. Con-

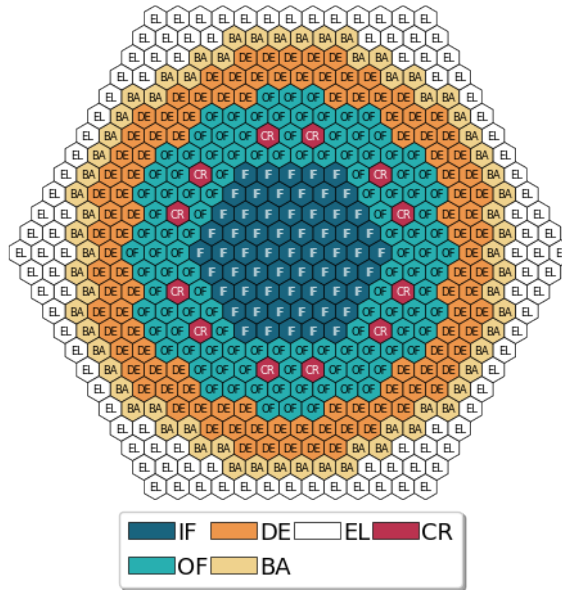
sequently, the primary system can operate at near-atmospheric pressure, drastically reducing the mechanical stress on the reactor vessel and preventing high-pressure injection scenarios.

- b) *Chemical Inertness*: Does not react violently or exothermically with water or atmospheric oxygen. This chemical stability simplifies the design of the containment and the secondary heat removal interfaces.
- c) *Low Neutronic Absorption and High Scattering*: Ensures an excellent neutron economy, which is a prerequisite for sustaining a fast spectrum capable of breeding fissile material and transmuting long-lived actinides.
- d) *High Thermal Inertia and Density*: Provide a substantial thermal buffer, slowing down temperature excursions during loss-of-flow accidents and fostering the establishment of robust natural circulation.

The core (see Fig. 1.2) is characterized by a compact, heterogeneous arrangement of wrapped hexagonal fuel assemblies (FAs), and it is conceptually partitioned into distinct radial zones to optimize the neutron flux distribution and power profile:

- **Fuel Zones**: The core comprises 171 fuel assemblies, sub-divided into an inner zone (57 assemblies) and an outer zone (114 assemblies) with varying plutonium enrichment levels to flatten the radial power distribution. The fuel matrix consists of mixed uranium-plutonium oxide (MOX), wrapped in advanced austenitic steel structures (such as 15-15Ti or AIM1) designed to withstand high fast neutron fluences and corrosive environments.
- **Control and Safety Systems**: The reactivity control and shutdown capabilities are provided by two independent, redundant systems. These consist of 12 Control Rods (CRs) for power regulation and burnup compensation, and 4 Safety Rods (SRs) dedicated exclusively to emergency scram operations.
- **Surrounding Layout**: The active core region is radially bounded by 108 dummy elements, which serve as a shielding and reflector zone to mitigate radiation damage on the inner vessel structures and optimize neutron reflection.

The active fuel height is limited to a compact value of 0.6 m, minimizing the primary pressure drop across the active core while maintaining an appropriate power density [2].

Figure 1.2: *ALFRED Core Layout.*

1.4 Reactor Core modeling

1.4.1 General features of the core modeling problem

In fission reactors, neutronic (NE) and thermal-hydraulic (TH) phenomena are extremely coupled. Accurately reproducing core behavior in numerical simulations requires a deep understanding of this mutual influence. The neutron flux distribution dictates the power map governing the heat transport equations, while also driving thermo-mechanical (TM) phenomena like swelling and thermal expansion. Conversely, the resulting coolant and fuel temperature distributions alter material cross-sections, directly impacting neutronic reactivity.

Given the characteristic high neutron flux and elevated temperatures of Lead Fast Reactors (LFRs), a precise assessment of the thermal field within the sub-assembly (SA) is a critical safety requirement. Adhering to the Generation IV philosophy, which embeds safety directly into the conceptual phase, TH conditions must be quantitatively evaluated early in the design process. Consequently, a set of numerical simulations is employed, ranging, e.g., from simplified 1D TH models to high-fidelity Computational Fluid Dynamics (CFD).

The reactor core serves as the critical nexus where TH, NE, and TM fields intersect with structural design, fuel assembly (FA) modeling, and material selection. Multiphysics models are therefore essential to support the design process. The primary objective of core design is to establish a configuration that meets performance targets while strictly adhering to TH, technological, TM, and economic constraints under both nominal and accidental

conditions. Establishing these boundaries aligns with the *defense in depth*¹ philosophy, combining inherent safety features with adequate margins to mitigate potential hazards at their source.

Similarly, the core’s TH design requires a rigorous analysis of flow distributions to prevent excessive thermal gradients. This involves ensuring adequate coolant flow external to the FA, which is governed by the core bypass flow. The diagrid’s TM design must simultaneously ensure the precise positioning of core elements and manage concentrated pressure drops to maintain this assumed bypass flow rate.

Neutronic design encompasses the core, the radiation shielding of adjacent structures, and the layout of control and safety systems. A key objective is defining the fuel enrichment zoning to optimize reactor functionality. This zoning must respect maximum burn-up limits, constrain cladding and fuel temperatures, and ensure a flat temperature profile. A uniform temperature distribution is crucial: it minimizes local hot spots (which can exacerbate corrosion), maximizes overall fuel utilization, and significantly reduces mechanical stresses among FA components, thereby simplifying their structural design.

Capturing these simultaneous phenomena requires sophisticated multiphysics simulations. High-fidelity Verification-Oriented Codes (VOCs), such as coupled CFD-Monte Carlo tools like Serpent-OpenFOAM [12], offer unparalleled accuracy. Monte Carlo neutronic codes are increasingly favoured for multiphysics applications due to their continuous-energy models and ability to handle arbitrary geometric complexities without cross-section approximations. However, simulating a full core with such tools imposes an extreme computational burden in terms of both time and memory.

Therefore, simpler Design-Oriented Codes (DOCs) are preferred during the iterative design phases. By carefully balancing complexity and accuracy, DOCs reduce computational time, often by focusing on specific regions or simplified phenomena [3]. Tools such as the system-level thermal-hydraulic code ATHLET and the core-level sub-channel code FRENETIC perfectly exemplify this category. By conceptually and numerically coupling these distinct yet complementary codes, engineers can achieve a robust and computationally efficient multiphysics framework, laying the necessary groundwork for the coupled architecture detailed in the following sections. A well-designed DOC targets three main objectives [3]:

- *Equilibrium*: The code must maintain consistent resolution across parameters, avoiding unnecessary computational expense on variables that minimally affect overall uncertainty.

¹A safety design based on the Defence-in-Depth (DiD) principle ensures plant reliability across all operational states, including normal operation, design basis accidents, and design extension conditions. The primary objectives of the DiD approach are to maintain continuous reactivity control, ensure adequate heat removal from the core, confine radioactive materials, and mitigate any accidental environmental releases [11].

- *Computational Efficiency:* Utilizing numerical methods tailored to the specific scale of interest to ensure short simulation times without requiring excessively complex architectures.
- *Focused Application Domain:* Maintaining a clear, well-defined scope of application.

Developing numerical codes for LFRs introduces distinct deviations, from Light Water Reactor (LWR) or thermal-spectrum modeling, that must be taken into account. Certain LFR design characteristics allow for simplified approaches without sacrificing accuracy. For instance, the coolant’s incompressibility and exceptionally high boiling point significantly streamline TH calculations. Furthermore, the neutron kinetics in LFRs are generally less complex than in thermal reactors. Because the neutron mean free path in a fast spectrum spans several centimeters, the flux profile is less perturbed by individual fuel rods. Consequently, the point-kinetics approximation, assuming a constant spatial flux profile, remains highly robust during transients.

However, the coupled temperature-reactivity dynamics are markedly more intricate. Models must accurately capture the radial expansion of the FA diagrid and the fuel Doppler coefficient. Transients in Heavy Liquid Metal Cooled Reactors (HLMCRs) operate on time scales, temperature gradients, and overall dynamics that differ fundamentally from other reactor types. Thus, precisely calculating fuel temperature is paramount, not only to define the TH operating margins preventing cladding failure, but also to accurately capture the critical Doppler reactivity feedback.

1.4.2 FRENETIC & ATHLET Codes

The **Fast REactor Neutronics/Thermal-hydraulicS (FRENETIC)** code [13] [14], written in Fortran, has been developed at Politecnico di Torino for fast multi-physics simulation of coupled neutronics/thermal-hydraulics steady-state and transient calculations in fast reactors featuring a closed hexagonal fuel assembly geometry. The code implements a quasi-3D approach by coupling:

- A neutronic module: solves the multigroup diffusion equations via a nodal method.
- A thermal-hydraulic module: employs a "1D+2D" model: it solves the 1D axial conservation equations (mass, momentum, energy) for the coolant and fuel pins within each independent hexagonal assembly, while explicitly modeling the 2D horizontal thermal coupling between adjacent assemblies through the wrapper walls and the inter-wrapper gap.

While FRENETIC provides a detailed full-core analysis, the study of the entire plant dynamics requires a system code. In this context, **ATHLET (Analysis of Thermal-hydraulics of LEaks and Transients)** is a best-estimate system code capable of modeling the complex hydraulic networks of the primary and secondary circuits, including pumps, heat exchangers, and control systems [15].

By coupling the detailed core modeling of FRENETIC with the comprehensive system-level simulation of ATHLET, it is possible to fully capture the dynamic interactions between the reactor core and the external plant components, which is essential for accurately simulating complex transients.

1.5 Aim of the thesis

The primary aim of this thesis is the preliminary validation of the refactoring of the thermal-hydraulic (TH) module of the FRENETIC code and its application in a coupled framework with the ATHLET system code to perform a comprehensive analysis of the ALFRED reactor. The work initially focuses on the rigorous validation of the standalone TH module against established experimental data obtained from the CIRCE experimental facility. This involves defining the geometrical model, setting up the physical properties of the coolant, establishing appropriate boundary conditions.

Following the validation of the core thermal-hydraulic model, the thesis aims to implement and apply the coupling between FRENETIC and ATHLET. In this coupled configuration, ATHLET resolves the primary circuit dynamics, providing inlet boundary conditions to the core, while FRENETIC calculates the detailed core thermal-hydraulics and power distribution, returning the total thermal power and core pressure drop to the system code. This integrated approach is finally employed to set the basis for a full-scale dynamics of the ALFRED reactor during selected transients, providing a robust comparison with historical experimental data.

1.6 Thesis outline

This thesis is organized as follows:

- Chapter 2 offers a description of the FRENETIC and ATHLET codes, outlining their architecture and respective physical models.
- Chapter 3 dives into the modification of the thermal-hydraulic main code in order to provide a better structured architecture of it.
- Chapter 4 is dedicated to the inclusion of the flowering phenomenon into the general thermohydraulic calculations.
- Chapter 5 is dedicated to the validation of the thermal-hydraulic module, with the creation of standalone cases to compare the new results with the old ones understanding if they have maintained the accuracy or not.
- Chapter 6 details the practical implementation of the coupling strategy of FRENETIC with ATHLET.

- Chapter 7 presents the results of the simulations for the coupling scheme, discussing the comparison between the computational predictions and the experimental data, followed by concluding remarks and suggestions for future work.

Chapter 2

Computational Tools for Reactor Analysis

2.1 The FRENETIC Thermal-Hydraulic Module

The thermal-hydraulic (TH) module within FRENETIC is specifically designed to compute the axial evolution of the coolant’s thermal-hydraulic parameters (velocity, pressure, and temperature) averaged over each Hexagonal Assembly (HA), alongside the average fuel temperature. By applying specific boundary conditions and utilizing the power distribution provided by the neutronic module, the TH solver evaluates the fundamental conservation equations for mass, momentum, and energy for each individual HA [4].

The temperatures computed by the TH module are then fed back into the neutronic (NE) module to update the macroscopic cross-sections, closing the feedback loop. The temperature dependence of the macroscopic cross-sections for a given material is modelled using a bi-variate linear interpolation based on the fuel and coolant temperatures characterizing the specific computational volume. The cross-section Σ is updated according to the following formulation [14]:

$$\Sigma(T_c, T_f) = \Sigma(T_{c,0}, T_{f,0}) + \frac{\partial \Sigma}{\partial T_f}(T_f - T_{f,0}) + \frac{\partial \Sigma}{\partial T_c}(T_c - T_{c,0}) \quad (2.1)$$

where T_c and T_f are the current coolant and fuel temperatures, while $T_{c,0}$ and $T_{f,0}$ represent their respective reference reference states.

Operationally, the NE and TH modules run independently, each advancing according to its own characteristic time step. To ensure rigorous numerical stability and exact alignment at coupling intervals, the TH module employs an internal adaptive sub-stepping logic. Through this mechanism, the solver autonomously executes multiple internal micro-steps, dynamically adjusting their size to seamlessly cover the transient progression and perfectly match the required synchronization time without overshooting.

Multiphysics coupling is achieved by synchronizing the modules at user-defined intervals to exchange critical physical quantities: the TH module provides the updated temperature

fields, while the NE module supplies the resulting power deposition. This coupling strategy effectively accommodates the distinct time scales associated with fast and delayed neutron populations, as well as their consequent effects on both the amplitude and the spatial shape of the power distribution. However, it is important to note that while this optimal resolution holds true within the internal neutronic time-step, the overall temporal accuracy of the inter-domain feedback effects remains inherently limited by the chosen coupling interval (Δt_{sync}). Ultimately, this staggered data-exchange approach ensures the accurate prediction of transient thermal feedbacks within the reactor core.

2.1.1 Code Architecture and Modularity

To ensure computational efficiency and maintainability, FRENETIC Thermal-Hydraulic Module is built upon a modular architecture. The core execution is driven by a main program that orchestrates the initialization, multi-physics coupling, and transient progression. The code structure (see Fig. 2.3) relies on different folders that contain different Fortran codes that serve to different functions:

- **coupling**: used to ensure that all the data necessary to perform the coupling between neutronics and thermal-hydraulics are declared consistently and in the most efficient way.
- **inout**: contains all the instructions for reading the input and producing the output files that the main code of the thermal hydraulic module should go to use.
- **modules**: contains all the information related to data and boundary conditions that will be used during the simulations.
- **prop**: contains all the list of material properties.
- **solve**: contains all the codes related to the discretization of the equations describing the thermal-hydraulic model (advection, conduction, ...), with the related boundary conditions; and contains information related to the resolution of the steady state and the transient. Furthermore, the distinction is also made (for filling the matrices) between the channel in which the fuel is contained, the channels in which no fuel elements are contained and the empty ones.
- **struct_alloc**: contains all the codes that are used to provide memory allocation and deallocation for the information present in the main code.
- **utilities**: contains additional information that are used to perform the simulations.

The standard execution flow begins with the reading of configuration files and geometry allocation, followed by the initialization of output files and multi-physics coupling parameters. Once the initial thermal state is defined, the code enters the transient phase loop, automatically handling transition regimes (such as axial-to-radial heat transfer shifts).

2.1.2 Calculation Domain and Core Modeling Approach

The computational domain (see Fig. 2.1) spans the axial length of the assemblies, specifically focusing on the region characterized by a constant cross-section, located downstream of the bypass holes and base enlargements. Radially, the Inter-Wrapper (IW) region separating adjacent assemblies is generally treated as stagnant lead. Under normal operating conditions, the mass flow rate in this region accounts for only about 5% of the total inlet flow to the Fuel Assembly (FA) [16]. Given the turbulent flow regime and the absence of blockages, this bypass flow does not significantly contribute to inter-assembly heat transfer. However, to maintain modularity, the code allows for the insertion of a pseudo-resistance in place of pure conduction, enabling the simulation of different convective behaviors in the flowing lead. While inter-wrapper flow can become highly relevant in the transverse direction during off-normal scenarios requiring full 3D thermal-hydraulic (TH) evaluations, such conditions currently fall outside the primary scope of FRENETIC.

According to these design requirements, the TH module must compute the average pressure, velocity, coolant temperature, and fuel temperature for each Hexagonal Assembly (HA) at every axial node. The core is modelled by explicitly defining the various HAs along with their specific fuel and control rod distributions.

Because the thermal coupling between neighbouring hexagons is relatively weak compared to the rapid axial transit of the coolant, the 3D core TH problem can be effectively decoupled. It is treated as a network of parallel, weakly coupled 1D channels with similar geometry and physics. The validity of this 1D (axial) + 2D (radial) splitting approach has been demonstrated in previous studies, originally developed by the Politecnico di Torino research group for the thermal-hydraulic modeling of superconducting magnets in nuclear fusion reactors, and subsequently adapted for FRENETIC [13].

Future improvements to this single-channel approach may include the implementation of a dynamic model that resolves additional axial equations for non-heated "dummy" rods. Currently, these structures only factor into the computation of flow areas and wetted perimeters, with their heat capacities neglected in the energy equations. Introducing a radially lumped transient computation for the temperature of these non-heated rods could smooth unphysical coolant temperature peaks that are occasionally observed during fast transients.

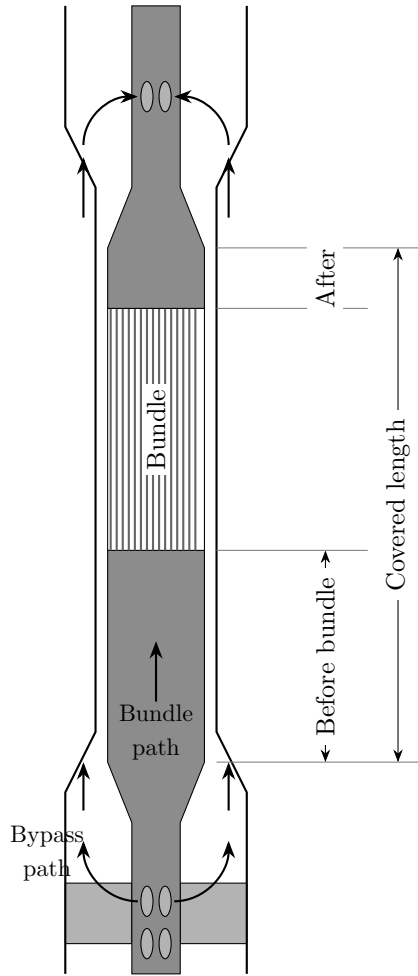


Figure 2.1: *Schematic of the bypass flow domain [3].*

2.1.3 Spatial Discretization and Numerical Method

In the new version of the FRENETIC TH module [16], spatial discretization is performed using a Finite Volume (FV) approach [17]. Each HA is axially subdivided into a series of control volumes (Fig. 2.2). This mesh can be non-uniform, allowing for increased axial resolution in critical areas, such as the active core region. The governing equations retain a space-dependent cross-sectional area term, A_i , to maintain generality and allow for the future implementation of smooth transitions between domains with varying flow areas (though currently, a stepwise variation is utilized).

A staggered mesh scheme is employed: scalar variables, like temperatures and thermophysical material properties, are evaluated at the cell centers (representing the mean values of the control volume), whereas vector quantities, like velocities, are evaluated at the cell faces. The module implicitly solves the transient 1D mass, momentum, and energy balances for each HA in the axial (z) direction. Each channel is then radially coupled to its adjacent neighbours via a resistive model to achieve the full-core 3D solution. The resulting set of ordinary differential equations is linearized using a frozen-coefficients as-

sumption, and the coupling matrix is treated implicitly. This numerical strategy preserves the banded structure of the system matrix, ensuring high computational efficiency and reducing CPU time. Furthermore, the decoupling of the various problem blocks allows the solution to be computed sequentially.

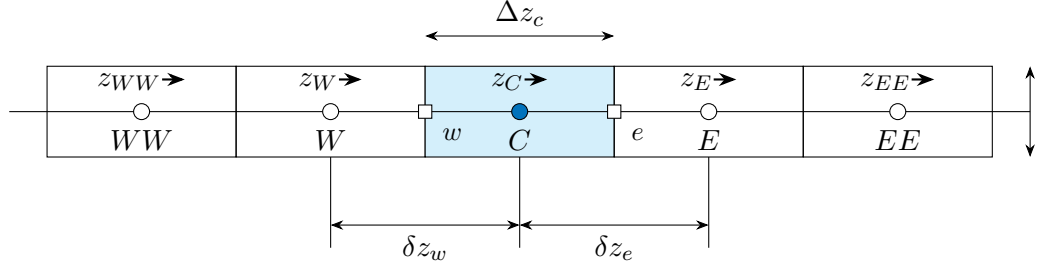


Figure 2.2: *Spatial nomenclature for mesh discretization [4].*

2.1.4 Boundary Conditions and Peripheral Assemblies

The primary domain of interest lies between the core’s lower and upper plenums. The TH module accepts user-specified distributions of the inlet mass flow rate for each HA, typically derived from external flow-split calculations. The accepted boundary conditions can vary both in space and time, encompassing inlet temperatures, inlet mass flow rates, or specified inlet/outlet pressures.

A critical aspect of full-core modeling is the treatment of boundary assemblies, which represent the core barrel and the surrounding external lead shield. While a detailed TH solution for these regions would require complex 3D CFD calculations well beyond FRENETIC’s scope, these components must be included to maintain geometrical consistency with the neutronic (NE) module, which requires them for proper flux evaluation. To bridge this gap, the barrel and external lead regions are simulated in the TH module as fictitious HAs. A practical strategy adopted for these unheated zones is to assign them a zero mass flow rate and force their temperature to equal the inlet coolant temperature. This simplification is highly acceptable since these boundary regions lie far from the active core where steep thermal gradients occur. This simplification is highly acceptable since these boundary regions lie far from the active core where steep thermal gradients occur. Nevertheless, the explicit inclusion of these peripheral assemblies remains crucial for the overall transient analysis, as they significantly contribute to accurately modelling the total thermal inertia of the system.

Finally, the TH module receives the spatially and temporally varying volumetric heat generation directly from the NE module’s flux distribution. A future development goal is to allow the user to impose an overall core pressure drop as an input boundary condition, prompting the code to iteratively compute the resulting mass flow rate distribution using under-relaxation factors to ensure numerical convergence.

2.1.5 Physical Models and Governing Equations

FRENETIC employs a coupled 2D+1D approach for neutronics and a 1D+2D approach for thermal-hydraulics. The thermal-hydraulic module solves the 1D axial conservation equations for the liquid metal coolant coupled with the 2D (radial and axial) heat conduction equation within the fuel pins [4].

For the **incompressible single-phase liquid metal coolant**, the 1D mass, momentum, and energy conservation equations are expressed as:

$$\begin{cases} \frac{\partial \rho_c}{\partial t} + \frac{\partial(\rho_c v)}{\partial z} = 0 \\ \rho_c \frac{\partial v}{\partial t} + \rho_c v \frac{\partial v}{\partial z} = -\frac{\partial p}{\partial z} - \rho_c g - f \frac{\rho_c v^2}{2D_h} \\ \rho_c c_{p,c} \left(\frac{\partial T_c}{\partial t} + v \frac{\partial T_c}{\partial z} \right) = \frac{q'' P_h}{A_c} + \frac{\partial}{\partial z} \left(k_c \frac{\partial T_c}{\partial z} \right) \end{cases} + \text{I.C. and B.C.} \quad (2.2)$$

where ρ_c is the coolant density, v the axial velocity, p the pressure, f the friction factor, D_h the hydraulic diameter, $c_{p,c}$ the specific heat capacity, T_c the coolant temperature, q'' the wall heat flux, P_h the heated perimeter, and A_c the flow area.

The **temperature distribution** within the solid fuel rod (fuel pellet, gap, and cladding) is governed by the transient heat conduction equation, subjected to appropriate initial and boundary conditions:

$$\rho_f c_{p,f} \frac{\partial T_f}{\partial t} = \nabla \cdot (k_f \nabla T_f) + q''' \quad (2.3)$$

where ρ_f , $c_{p,f}$, T_f , and k_f are the properties of the fuel, and q''' is the volumetric heat generation rate provided by the coupled neutronic module.

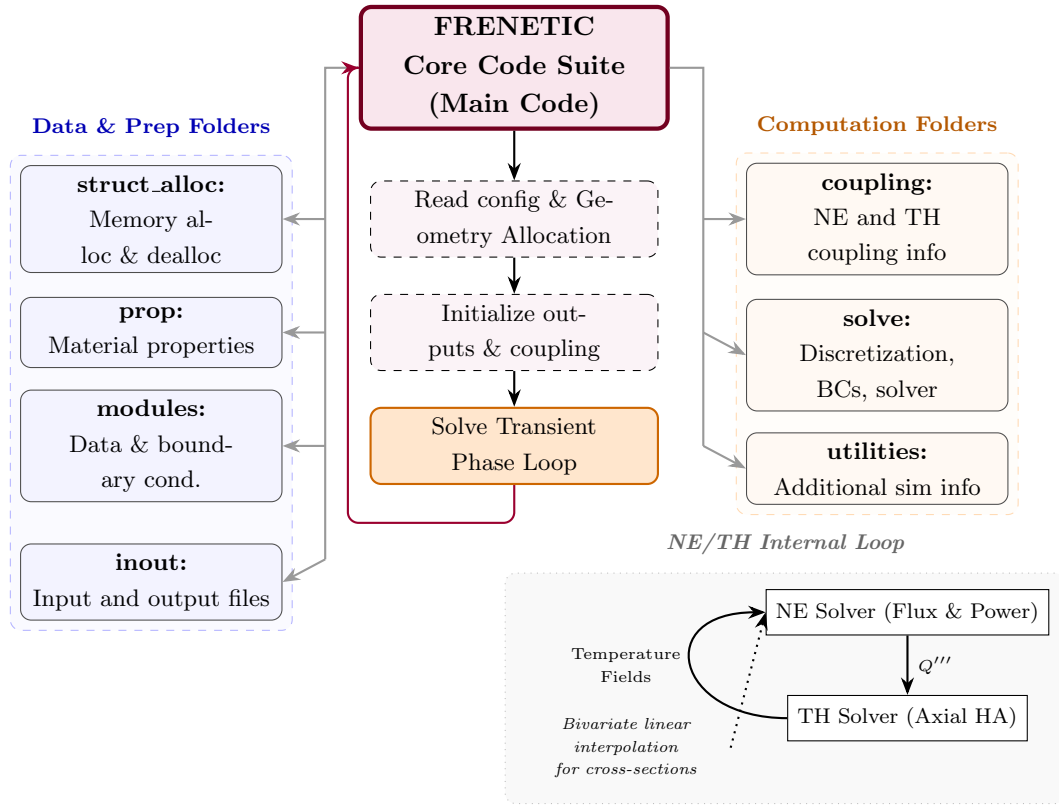


Figure 2.3: Detailed functional and folder architecture of the FRENETIC code.

2.2 The ATHLET Code

The Analysis of THERmal-hydraulics of LEaks and Transients (ATHLET) system code is developed by the Gesellschaft für Anlagen- und Reaktorsicherheit (GRS). Written primarily in Fortran 90, it is an advanced best-estimate code designed to simulate the whole spectrum of operational conditions, transients, and design-basis accidents in various nuclear reactor types, including those employing heavy liquid metals [15].

ATHLET features a highly modular architecture that enforces a strict separation between physical models and numerical solution methods. The code is primarily composed of the following fundamental modules (see Fig. 2.5):

- **Thermo-Fluid Dynamics (TFD):** The core module responsible for calculating fluid flow and thermal energy transport. It supports 1D networks and includes a 3D module for resolving complex multidimensional flows.
- **Heat Conduction and Heat Transfer (HECU):** Models the thermal response of solid structures, calculating 1D heat conduction in fuel rods, pipe walls, and other components, along with the respective boundary heat transfer coefficients.
- **Neutron Kinetics (NEUKIN):** Handles nuclear heat generation through either point-kinetics or 1D/3D neutronic models.

- **General Control Simulation Module (GCSM):** Simulates the balance of plant (BOP), control mechanisms, and protection systems.

All ordinary differential equations (ODEs) generated by these modules are integrated simultaneously in time using the dedicated mathematical solver **FEBE (Forward-Euler, Backward-Euler)**. FEBE incorporates a rigorous error-control mechanism that evaluates the local truncation error at every time step. If the calculated error exceeds a specified tolerance, the solver rejects the current step, dynamically halves the time increment (Δt), and recalculates the solution. This adaptive approach ensures numerical stability and precision, even during severe or highly dynamic transient events.

2.2.1 TFD Module: physical models and governing equations

The Thermo-Fluid Dynamics (TFD) module of ATHLET is highly flexible and offers different equation systems depending on the required accuracy, the complexity of the transient, and the fluid's state. It is capable of modeling single-phase and complex two-phase flows using two primary approaches:

1. The 5-Equation Model: This model is based on separate continuity and energy conservation equations for the liquid and vapour phases ($k = l$ for liquid and $k = v$ for vapour), combined with a single two-phase mixture momentum equation. Because it solves only one momentum equation, it assumes a strong mechanical coupling between the phases. To account for the relative velocity (slip) between liquid and vapour, a full-range drift-flux model (often based on kinematic constitutive equations) is utilized. This approach is computationally efficient, highly robust, and includes dynamic mixture-level tracking capabilities, making it suitable for most standard operational transients where severe phase separation is not the dominant phenomenon.

The conservation equations for this model are the following:

$$\underline{\text{Mass Cons.}} \quad \longrightarrow \quad \begin{cases} \frac{\partial(\alpha_l \rho_l)}{\partial t} + \nabla \cdot (\alpha_l \rho_l \vec{v}_l) = \Gamma_l \\ \frac{\partial(\alpha_v \rho_v)}{\partial t} + \nabla \cdot (\alpha_v \rho_v \vec{v}_v) = \Gamma_v \end{cases} \quad (2.4)$$

Where Γ_l and Γ_v represent the interfacial mass transfer rates for the liquid and vapour, respectively. In order to satisfy the total mixture mass conservation, the mass leaving one phase must enter the other, meaning $\Gamma_l = -\Gamma_v$.

$$\underline{\text{Energy Cons.}} \quad \longrightarrow \quad \begin{cases} \frac{\partial(\alpha_l \rho_l h_l)}{\partial t} + \nabla \cdot (\alpha_l \rho_l \vec{v}_l h_l) = \alpha_l \frac{\partial p}{\partial t} + q_{w,l} + q_{i,l} + \Gamma_l h_l^* \\ \frac{\partial(\alpha_v \rho_v h_v)}{\partial t} + \nabla \cdot (\alpha_v \rho_v \vec{v}_v h_v) = \alpha_v \frac{\partial p}{\partial t} + q_{w,v} + q_{i,v} + \Gamma_v h_v^* \end{cases} \quad (2.5)$$

Where p is the common system pressure, q_w is the wall heat transfer rate, q_i is the sensible interfacial heat transfer rate, and Γh^* represents the energy transported across

the interface due to phase change (h^* being the saturation enthalpy of the phase changing state).

$$\text{Mix. Momentum Cons.} \rightarrow \frac{\partial \vec{G}_m}{\partial t} + \nabla \cdot (\alpha_l \rho_l \vec{v}_l \vec{v}_l + \alpha_v \rho_v \vec{v}_v \vec{v}_v) = -\nabla p + \rho_m \vec{g} - \vec{F}_{w,m} \quad (2.6)$$

Where $\vec{G}_m = \alpha_l \rho_l \vec{v}_l + \alpha_v \rho_v \vec{v}_v$ is the mixture mass flux vector, and $\rho_m = \alpha_l \rho_l + \alpha_v \rho_v$ is the mixture density. The term ∇p is the spatial pressure gradient, $\rho_m \vec{g}$ represents the gravitational body force, and $\vec{F}_{w,m}$ is the two-phase wall friction force per unit volume opposing the flow. To mathematically close the system and determine the individual phase velocities ($\vec{v}_l$ and $\vec{v}_v$), ATHLET relies on empirical Drift-Flux correlations that calculate the relative slip velocity ($\vec{v}_r = \vec{v}_v - \vec{v}_l$) based on the local two-phase flow regime.

2. The 6-Equation (Two-Fluid) Model: This advanced model provides the highest level of detail by featuring completely separated balance equations for mass, energy, and momentum for both the liquid and vapour phases (indicated again by the letter k). By tracking the momentum of each phase independently, it allows for both thermal and mechanical non-equilibrium. This provides superior accuracy, with respect to the 5-equation model, in simulating complex two-phase phenomena, such as counter-current flows, stratified flows in horizontal pipes, and severe loss-of-coolant accidents (LOCA). The conservation equations for this model are the following:

$$\text{Mass Cons.} \rightarrow \frac{\partial(\alpha_k \rho_k)}{\partial t} + \nabla \cdot (\alpha_k \rho_k \vec{v}_k) = \Gamma_k \quad (2.7)$$

This equation dictates the mass balance for phase k . The variable α_k is the volumetric fraction (void fraction) of the phase, subject to the geometric constraint $\alpha_l + \alpha_v = 1$. The term ρ_k is the macroscopic density, and $\vec{v}_k$ is the phase velocity vector. The source term on the right-hand side, Γ_k , represents the interfacial mass transfer rate per unit volume (in other words, evaporation or condensation). To satisfy the overall mixture mass conservation, the mass leaving one phase must enter the other, yielding $\Gamma_l + \Gamma_v = 0$.

$$\text{Momentum Cons.} \rightarrow \frac{\partial(\alpha_k \rho_k \vec{v}_k)}{\partial t} + \nabla \cdot (\alpha_k \rho_k \vec{v}_k \vec{v}_k) = -\alpha_k \nabla p + \alpha_k \rho_k \vec{g} - \vec{F}_{w,k} \pm \vec{F}_i + \text{I.C. and B.C.} \quad (2.8)$$

The momentum balance accounts for the rate of change of momentum and its convective transport. On the right-hand side, the forces driving or resisting the flow are detailed: $-\alpha_k \nabla p$ represents the pressure gradient driving force, assuming a common local pressure p for both phases at the interface; $\alpha_k \rho_k \vec{g}$ is the volumetric body force due to gravity; $-\vec{F}_{w,k}$ denotes the wall friction force per unit volume acting on phase k , dissipating momentum due to contact with the pipe or channel boundaries; and $\pm \vec{F}_i$ is the interfacial friction force (drag), which dynamically couples the two momentum equations. It accounts for the momentum exchange resulting from the relative velocity between liquid and vapour.

By Newton’s third law, the drag acting on the vapour is equal and opposite to the drag acting on the liquid.

$$\underline{\text{Energy Cons.}} \longrightarrow \frac{\partial(\alpha_k \rho_k h_k)}{\partial t} + \nabla \cdot (\alpha_k \rho_k \vec{v}_k h_k) = \alpha_k \frac{\partial p}{\partial t} + q_{w,k} + q_{i,k} + \Gamma_k h_k^* \quad + \text{ I.C. and B.C.} \quad (2.9)$$

The energy equation is formulated in terms of the specific enthalpy of the phase, h_k . The terms on the right-hand side dictate the energy exchange mechanisms: $\alpha_k \frac{\partial p}{\partial t}$ is the transient pressure work term, representing the reversible mechanical work done by pressure variations over time; $q_{w,k}$ represents the volumetric heat transfer rate from the structural walls (e.g., fuel rods and heat exchangers) directly to phase k ; $q_{i,k}$ is the sensible interfacial heat transfer rate between the liquid and vapour phases; and $\Gamma_k h_k^*$ accounts for the energy transported across the interface due to phase change, where h_k^* represents the saturation enthalpy of the phase changing state. This term is crucial to ensure strict energy conservation during evaporation and condensation processes.

The differential equations described above contain more unknown variables than available equations. To mathematically close the system, ATHLET employs a comprehensive set of empirical and semi-empirical constitutive relations. These include thermodynamic equations of state, heat transfer coefficient maps to determine $q_{w,k}$ and $q_{i,k}$, and friction correlations to compute pressure drops ($\vec{F}_{w,k}$) and interfacial drag ($\vec{F}_i$). These correlations are continuously updated based on the dynamically evaluated local flow regime (e.g., bubbly, slug, annular, or dispersed flow).

For spatial discretization, ATHLET translates the physical plant geometry into a network of 1D thermo-fluid dynamic objects (TFOs). The code employs a finite-volume approach based on a *staggered grid* topology, which strictly separates the locations where scalar and vector variables are calculated:

- **Control Volumes (CVs):** These represent the spatial nodes where mass and energy are stored. Scalar thermodynamic variables, such as pressure (p), specific enthalpies (h_l, h_v), and void fraction (α), are evaluated at the center of the CVs. ATHLET uniquely features both *Homogeneous CVs* and *Mixture-Level CVs*, the latter being capable of tracking the exact physical elevation of the liquid-vapour interface within components like the pressurizer or the steam generator.
- **Junctions:** These define the flow paths connecting the centers of two adjacent CVs. They do not possess mass or volume. Vector variables, strictly related to momentum conservation, such as mass flow rates (W) and phase velocities ($\vec{v}_l, \vec{v}_v$), are computed exactly at the junctions.

This staggered configuration inherently prevents unphysical spatial oscillations (checkerboard effect) in the pressure and velocity fields. The resulting Ordinary Differential Equations (ODEs) from all spatial nodes are then simultaneously advanced in time by the

FEBE solver. This advanced numerical solver dynamically adjusts the time step based on the Courant-Friedrichs-Lewy (CFL) limit, internal convergence criteria, and the severity of the simulated transient, ensuring both numerical stability and optimal computational efficiency.

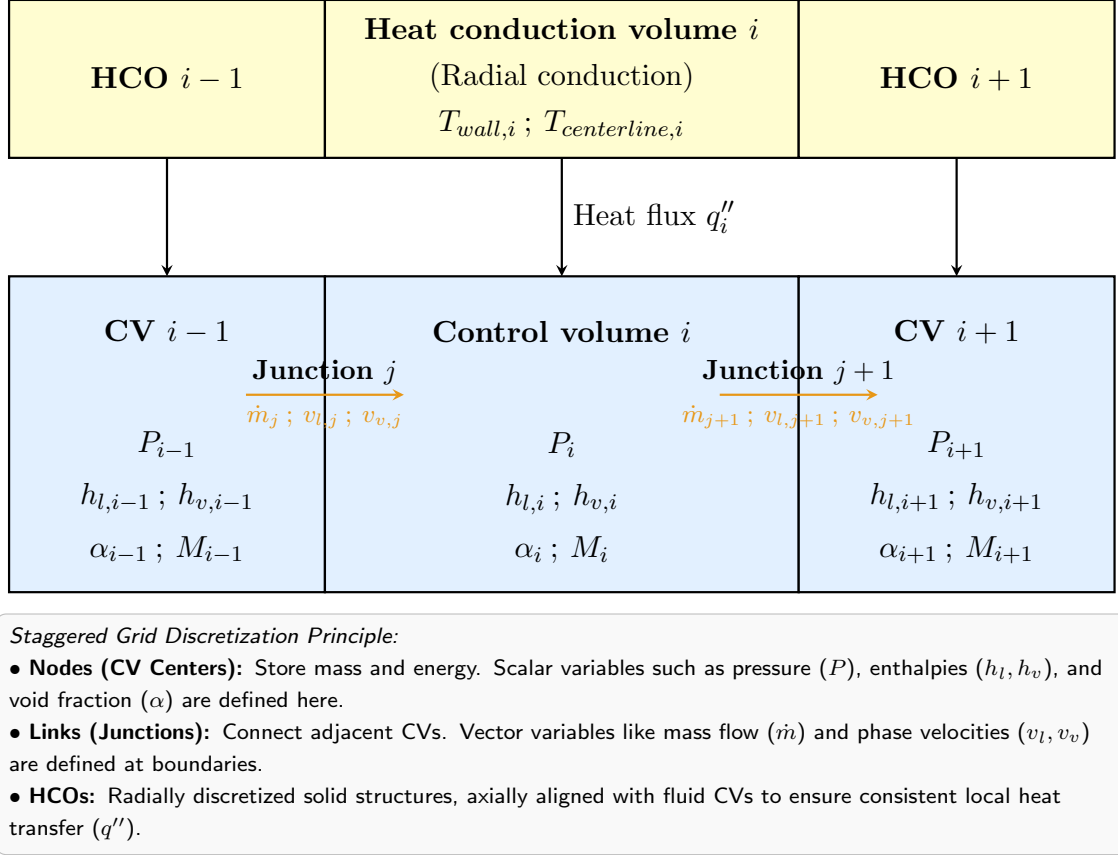


Figure 2.4: Schematic representation of the ATHLET 1D computational domain.

2.2.2 The Heat Conduction and Heat Transfer Module (HECU)

The HECU module is responsible for simulating the thermal response of solid structures and their interaction with the fluid. These structures, referred to as Heat Conduction Objects (HCOs), represent components such as fuel rods, steam generator tubes, and vessel or pipe walls.

The core of the module is the solution of the one-dimensional, transient heat conduction equation in various geometries (slabs, cylinders, or spheres). The general governing equation for the temperature field $T(r, t)$ within a solid structure is:

$$\rho c_p \frac{\partial T(r, t)}{\partial t} = \frac{1}{r^n} \frac{\partial}{\partial r} \left(r^n \lambda \frac{\partial T(r, t)}{\partial r} \right) + q''' \quad + \text{I.C. and B.C.} \quad (2.10)$$

where ρ , c_p , and λ are the density, specific heat capacity, and thermal conductivity of the material (often defined as temperature-dependent functions), while q''' represents the

volumetric heat source (e.g., nuclear fission or decay heat). The exponent n defines the geometry: $n = 0$ for a flat plate, $n = 1$ for a cylinder, and $n = 2$ for a sphere.

The HECU module employs a precise spatial discretization:

- **Radial Discretization:** HCOs are divided into radial zones or "layers" to resolve the internal temperature profile. This is crucial for calculating the stored energy and the surface temperature of the cladding.
- **TFO Coupling:** Each HCO is axially subdivided into Heat Conduction Volumes (HCVs) that strictly correspond to the Control Volumes of the adjacent TFD objects. This ensures a consistent local coupling between the solid and the fluid phases.

The thermal coupling at the solid-fluid interface is governed by the Heat Transfer Map. ATHLET dynamically selects the most appropriate heat transfer correlation based on local hydraulic conditions (e.g., Reynolds number, pressure, void fraction). The surface heat flux q'' is calculated as:

$$q'' = h \cdot (T_{wall} - T_{fluid}) \quad (2.11)$$

where h is the Heat Transfer Coefficient (HTC). Additionally, HECU can model two-sided heat transfer, which is the standard approach for heat exchangers where energy is transferred from a primary to a secondary fluid through a metallic boundary.

2.2.3 The Neutron Kinetics Model (NEUKIN)

The NEUKIN module provides the power source term for the HECU objects representing the reactor core. In its standard implementation, ATHLET utilizes a Point Kinetics Model (PKE) to describe the time-dependent behavior of the total reactor power. This model assumes that the spatial distribution of the neutron flux remains constant, while its amplitude $n(t)$ varies according to:

$$\begin{cases} \frac{dn(t)}{dt} = \frac{\rho(t) - \beta}{\Lambda} n(t) + \sum_{i=1}^6 \lambda_i C_i(t) \\ \frac{dC_i(t)}{dt} = \frac{\beta_i}{\Lambda} n(t) - \lambda_i C_i(t) \quad (i = 1, \dots, 6) \end{cases} + \text{I.C.} \quad (2.12)$$

where $\rho(t)$ is the total reactivity, β is the delayed neutron fraction, Λ is the prompt neutron generation time, and C_i represents the concentration of the i -th group of delayed neutron precursors with decay constant λ_i .

The total reactivity $\rho(t)$ is a combination of external reactivity (e.g., control rod movement) and internal feedback effects:

$$\rho(t) = \rho_{ext}(t) + \Delta\rho_{Doppler}(T_{fuel}) + \Delta\rho_{mod}(T_{mod}, \rho_{mod}) \quad (2.13)$$

These feedback mechanisms ensure the physical coupling between the neutronics and the thermal-hydraulics. Specifically, the Doppler effect responds to fuel temperature variations (HECU), while moderator feedbacks respond to density and temperature changes in the coolant (TFD). For advanced multi-dimensional analyses, ATHLET can be coupled with external 3D neutron kinetics codes (like PARCS or QUABOX/CUBBOX) to resolve spatial flux tilts during complex transients [18].

2.2.4 The General Control Simulation Module (GCSM)

The GCSM module is a block-oriented tool designed to simulate the logic and hardware of the plant's Instrumentation and Control (I&C) systems, as well as the Balance of Plant (BOP). The module processes "signals" through a sequence of functional blocks to control actuators within the TFD and HECU modules.

The GCSM architecture is divided into three functional categories:

1. **Control Signals:** Act as "sensors" that extract physical variables from the simulation (e.g., pressure in a specific CV, mass flow in a junction, or temperature in an HCO).
2. **Basic Blocks:** These are mathematical and logical operators (adders, multipliers, integrators, delays, and logical gates like AND/OR/NOT) that process the input signals according to a user-defined logic.
3. **Control Elements (Actuators):** The processed signals are fed back into the thermo-fluid dynamic network to command hardware components, such as adjusting a valve's opening area, modulating a pump's rotational speed, or triggering a reactor trip.

In the context of multi-code coupling, the GCSM is of paramount importance. It serves as the data gateway for the preCICE interface, allowing the extraction of boundary variables from ATHLET and the imposition of feedback signals received from external codes like FRENETIC.

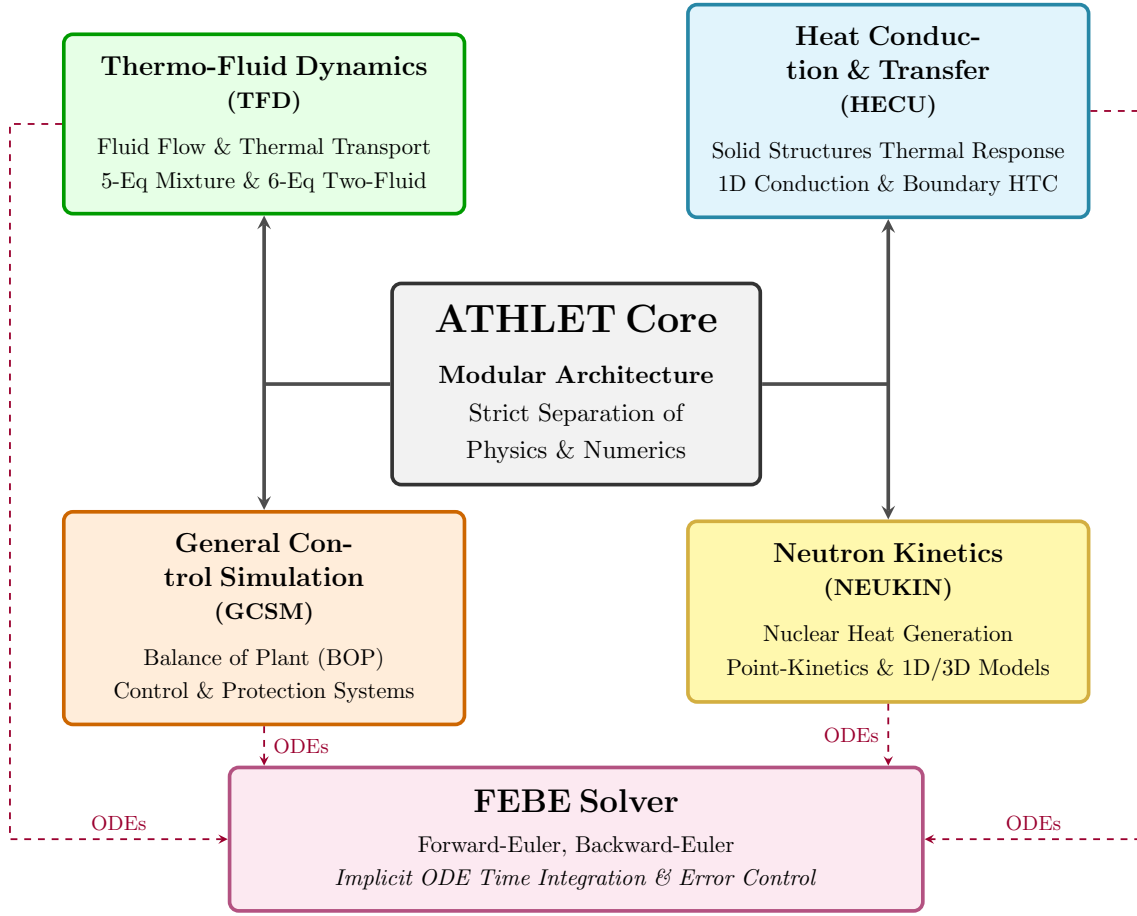


Figure 2.5: *Detailed modular architecture of the ATHLET system code.*

2.2.5 Coupling Strategy Between FRENETIC-TH and ATHLET

To achieve a comprehensive simulation of the entire nuclear plant while maintaining high-fidelity resolution within the reactor core, a multi-scale thermal-hydraulic coupling is established between the FRENETIC-TH module and the ATHLET system code.

In this architecture, ATHLET acts as the macroscopic system code, computing the overall primary circuit dynamics, the Balance of Plant (BOP), and the heat exchangers through its 1D Thermo-Fluid Dynamics (TFD) network. Conversely, FRENETIC-TH acts as the assembly-level core solver, providing a detailed 3D (1D axial + 2D radial) evaluation of the core thermal-hydraulics (see Fig. 2.6). The communication and synchronization between the two independent executables are guaranteed by a **Python-based Master Driver**, utilizing the official PyATHLET shared library interface (`libathlet.so`) and an **Event-Driven** paradigm [15].

The data exchange between the two codes occurs at the core interfaces through a specific set of fluid-fluid boundary conditions, ensuring the conservation of mass, momentum, and energy across the coupled domains. Since the coupling strategy can be adapted depending

on the specific transient and geometry (e.g., a 1D single-channel loop versus a full 3D pool reactor like ALFRED), the exchanged variables are generally defined as follows:

- **From ATHLET to FRENETIC-TH:** The system code provides the macroscopic thermal-hydraulic boundary conditions at the core interfaces. Depending on the simulation setup, these typically include the core inlet fluid temperature, the core outlet pressure, and the core inlet mass flow rate. Within ATHLET, the core domain can be virtually replaced by Time Dependent Volumes (TDVs) and fill junctions, whose values are exported to the Python Master Driver by querying ATHLET's internal variables (TFO objects) and transferring them via dedicated text files.
- **From FRENETIC-TH to ATHLET:** Using the boundary conditions provided by ATHLET and the internally evaluated power distribution, FRENETIC-TH computes the detailed core dynamics. It then returns integrated macroscopic quantities or source terms back to ATHLET. These feedback values (which may include the core integral thermal power, the average core outlet temperature, or the core pressure drops, depending on the specific coupling scheme) are received by the Python Master Driver through specific text files and dynamically injected into ATHLET's domain boundaries and thermal objects via the General Control Simulation Module (GCSM) API.

From a numerical perspective, this mapping requires a sophisticated synchronization mechanism. The coupling strategy detailed in this work relies on a **fully explicit scheme**, where the global time advancement is dictated by ATHLET's internal FEBE solver acting as the master clock. However, while this fully explicit methodology significantly reduces the computational overhead associated with iterative coupling loops, it inherently introduces potential drawbacks. The most notable disadvantage is its susceptibility to numerical instabilities: if the chosen macroscopic synchronization interval (Δt_{sync}) is too large, the delayed feedback resulting from the frozen boundary conditions can lead to non-physical oscillations and diverge the coupled calculation. Unlike implicit methodologies, this approach strictly avoids interface iterations, state saving, and trial steps. Instead, the two codes operate with independent time-steps and exchange data exclusively at predefined, discrete macroscopic synchronization intervals (Δt_{sync}), through .txt files. During the intermediate micro-steps, the two solvers run decoupled, maintaining the exchanged information frozen at their last exchanged values. The physical synchronization occurs strictly *after* the ATHLET FEBE solver has successfully achieved numerical convergence and accumulated time up to the specified macroscopic interval (Hook `ATRANS_AfterFEBE`). Only upon this successful convergence, the Python supervisor intervenes to transfer the updated information, commanding FRENETIC-TH to execute its catch-up phase and permanently commit the explicitly coupled time-step.

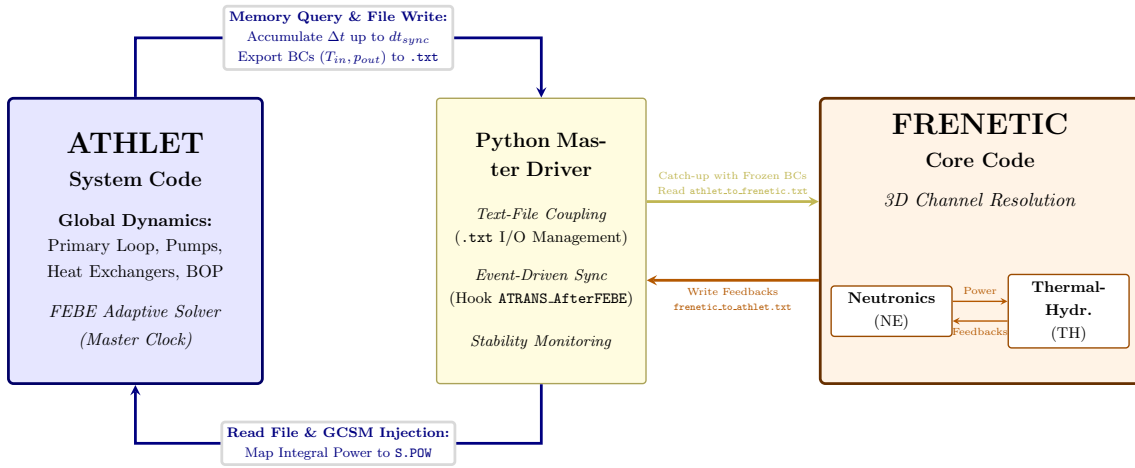


Figure 2.6: Schematic of the multi-scale explicit coupling between ATHLET and FRENETIC via Python Master Driver and Text-File exchange.

Chapter 3

Refactoring of the Thermal-Hydraulic Module Main Code `main.f90`

This chapter details the modifications applied to the original main program of the FRETIC thermal-hydraulic (TH) module. The primary motivation behind this refactoring was to improve the code’s overall architecture, making it highly modular and maintainable.

A detailed analysis of the original `main.f90` file revealed several architectural and programming criticalities:

- **Mixing of Abstraction Levels:** High-level operations (like transient phase initialization) and extremely low-level operations (like direct array manipulation) coexisted within the same file. This lack of separation made the code excessively heavy. The main program should strictly be restricted to managing the execution flow, transitioning from geometry setup, to steady-state, and finally to transient execution, which was not previously the case.
- **Scattered Coupling Management:** The multiphysics coupling operations were distributed randomly throughout the code. Logical checks on the coupling variable (defined as `isneth`) were repeated in entirely different sections. This redundancy posed a significant risk for future developments: adding new coupling features would require modifying the main code in multiple places, greatly increasing the likelihood of introducing bugs.
- **Chaotic Input/Output Management:** The handling of input and output files was highly disorganized, making the data flow difficult to trace. Consequently, ensuring proper data ingestion and generating coherent output results under all conditions became challenging. This disorganization also severely hindered the debugging process, as the origin of various variables was unclear.

- **Unsafe Operating System Interaction:** The previous version relied on external shell commands to interact with the OS. This approach is highly risky for complex software, especially when running on High-Performance Computing (HPC) clusters with strict security policies, where external calls might be blocked or fail silently. To resolve this, native and safe Fortran constructs for OS interactions were adopted.
- **Implicit Global States (Side Effects):** The original main program invoked numerous subroutines that lacked explicit input arguments, silently modifying global system variables. In software engineering, this is known as a "side effect". For rigorous validation, data must be explicitly passed between functions using derived types (like `type(State)`).

3.1 New Architecture and Modularity

To address these issues, the original code was systematically dismantled to separate the specific functionalities. Obsolete elements, both from a programming and physical modeling perspective, were completely removed. The execution logic was then encapsulated into four new, dedicated modules, strategically placed within the source directory tree (see Fig. 3.1):

1. **TH_Config_Reader_Mod** (located in the `inout` folder): Dedicated to opening files and efficiently reading system data and geometry.
2. **TH_IO_Manager_Mod** (located in the `inout` folder): Manages the output data flow, including the closure of temporary files and the execution of HDF5 writes.
3. **TH_Coupling_Manager_Mod** (located in the `coupling` folder): Centralizes all multi-physics coupling logic between the thermal-hydraulic and neutronic modules, isolating it from the main solver.
4. **TH_State_Mod** (located in the `modules` or `solve` folder): Handles the physical initialization of the thermal-hydraulic state, preparing matrices and temperatures at time $t = 0$ (the first step of the TH solver).

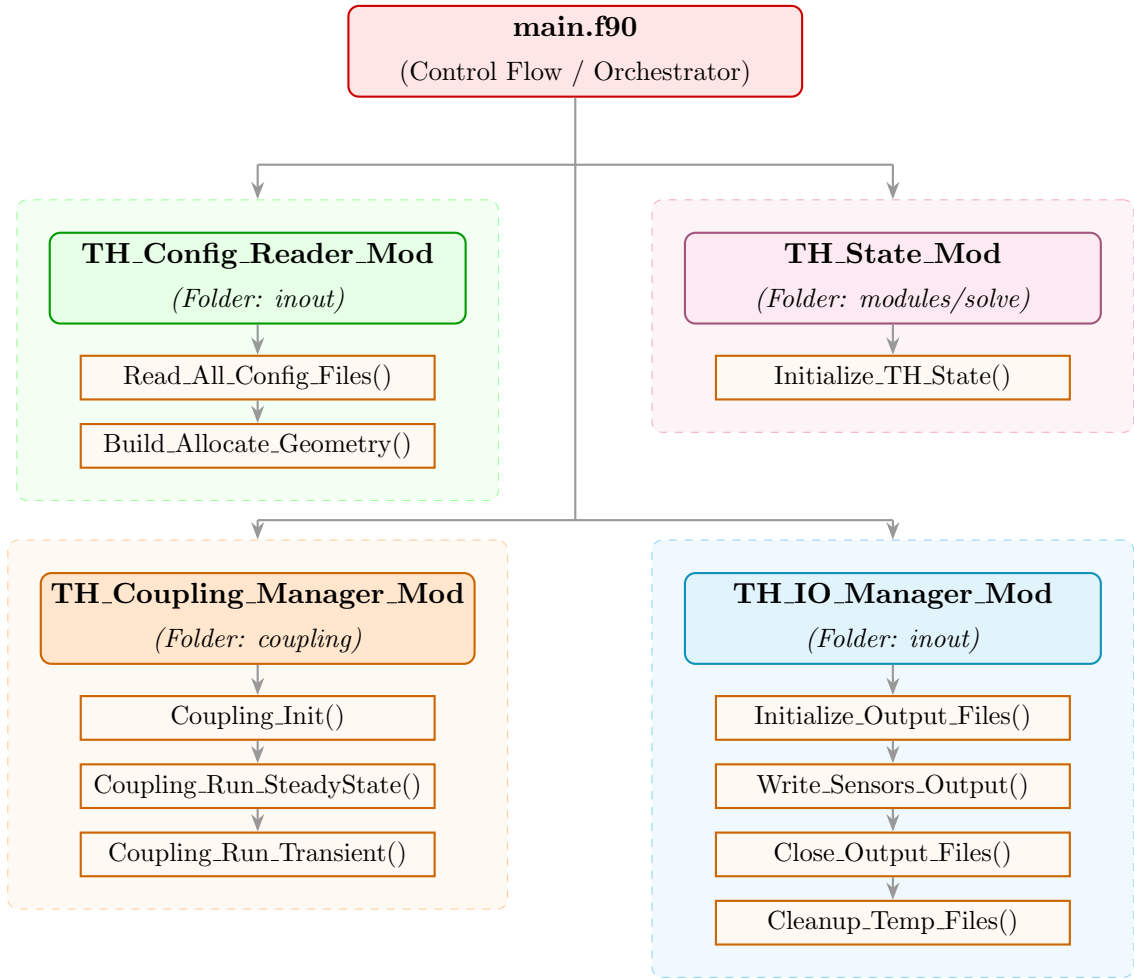


Figure 3.1: *Modular restructuring of the FRENETIC main driver for the thermal-hydraulic module.*

3.2 Power Variable Standardization: *powtot0* vs. *Pow0*

Another significant modification, particularly essential for the post-processing phase utilizing the `COREutils` library [19], involved standardizing the power variable used throughout the code. Originally, the FRENETIC thermal-hydraulic subroutines tracked two distinct power variables: *powtot0*, representing the total thermal power expressed in Watts [W], and *Pow0*, representing the linear heat generation rate expressed in Watts per meter [W/m]. To streamline the code and avoid ambiguity, it was decided to exclusively retain *Pow0* and discard *powtot0*.

This decision was driven by both physical and numerical considerations. From a physical point of view, the primary objective of these computational codes is to evaluate the fuel temperature evolution. The linear power [W/m] is the direct driver of the radial thermal gradient within the fuel pin, which determines the pellet centerline temperature and warns against potential fuel melting conditions.

From a numerical perspective, FRENETIC solves the energy balance equations by discretizing the flow channel into multiple axial nodes, each with a height of Δz . Because the governing equation for a single node requires a local heat source term, using the linear power allows the code to perform a simple elementary multiplication to obtain the nodal thermal power ($Q_{nod} = \text{Pow0} \cdot \Delta z$). Conversely, relying on the total thermal power in Watts would necessitate more convoluted calculations. Furthermore, if a non-uniform axial power profile is introduced, utilizing W/m becomes the only mathematically feasible and consistent approach to allocate power locally.

An analysis of the source code revealed that `powtot0` was previously only used within a specific formula to derive the linear power in the old `main.f90` file (a routine later moved to `TH_State_Mod_modified.f90` during the restructuring). To eliminate redundancy and ensure uniform logic across the solver, the following strategy was implemented:

- The `powtot0` variable was completely deleted from all files where it was previously initialized (`input.f90`, `numerics_mod.f90`).
- In the newly created `TH_State_Mod_modified.f90` module, the initialization of `powtot0` was removed. The formula used to calculate the internal variable `ti%qp` was updated to directly ingest `Pow0` (already expressed in W/m).
- For the post-processing routines within the `COREutils` folder, targeted adjustments were made primarily to two Python scripts:
 - In `frenetic_namelist.py`, the obsolete variable was removed, and checks were added to ensure `Pow0` (and its correlated variables) were correctly tracked.
 - In `InpGen.py`, `Pow0` was inserted in the exact data-reading locations where `powtot0` was previously expected. Additionally, an algorithm adaptation was introduced: since the upstream `Core.py` file reads power values in W, `InpGen.py` now automatically converts these values into W/m by leveraging the `zmesh` spatial discretization array.

3.3 Impact of Code Refactoring on the Validation Process

The structural refactoring of the main code is not merely a software engineering exercise; it is a fundamental prerequisite for the successful Verification and Validation (V&V) of the FRENETIC code.

In the context of nuclear reactor simulation, computational tools must be strictly deterministic, traceable, and reliable. The elimination of side effects and the enforcement of explicit data passing via derived types guarantee that thermal-hydraulic state variables are not unpredictably altered during execution.

Furthermore, breaking the monolithic structure into isolated modules lays the essential groundwork for the future implementation of a comprehensive unit-testing framework.

Although not yet fully deployed, the systematic unit testing of the I/O, coupling, and state initialization routines represents a necessary future development to automatically and continuously verify the integrity of each individual component. A clean, transparent code structure ensures that when the simulation results are compared against experimental benchmark data, any discrepancies can be confidently attributed to the underlying physical models rather than hidden software bugs or unpredictable data handling. Thus, this robust and "clean" architecture provides the necessary scientific solidity to formally validate the thermal-hydraulic module and guide its upcoming quality assurance steps.

Chapter 4

New code structure to account for the thermal expansion feedback

In Liquid Metal Fast Reactors (LMFRs), such as the Advanced Lead Fast Reactor European Demonstrator (ALFRED), fuel assemblies are tightly arranged and subjected to severe transverse thermal gradients and intense fast neutron fluxes. Over the reactor cycle, these conditions induce macroscopic deformations of the hexagonal wrappers, primarily as a result of thermal bowing and irradiation-induced swelling [20]. This complex thermo-mechanical behavior, for instance the **flowering** phenomenon, causes a dynamic variation of the inter-assembly clearance, locally altering the coolant cross-sectional flow area (A_{Cl}) and the wetted perimeter (p_{wet}).

The implementation of this phenomenon into the 1D-axial thermal-hydraulic code FRENETIC required a comprehensive assessment of the native finite-volume conservation equations and the development of a dynamic geometrical updating strategy.

4.1 Mathematical Framework and Finite Volume Suitability

A preliminary theoretical analysis was conducted to evaluate the impact of a variable cross-sectional area on the conservation equations. The 1D-axial thermal energy conservation equation for the coolant can be written in its differential form as:

$$\rho c_p A_{Cl}(z) \frac{\partial T}{\partial t} + \dot{m} c_p \frac{\partial T}{\partial z} - \frac{\partial}{\partial z} \left(k A_{Cl}(z) \frac{\partial T}{\partial z} \right) = q'_{tot}(z) + q'''_{vol}(z) A_{Cl}(z) \quad (4.1)$$

where the terms on the left-hand side represent the transient energy storage, the convective heat transfer, and the axial heat conduction, respectively, while the right-hand side accounts for the linear and volumetric heat generation sources.

In a classical finite-difference approach, applying the chain rule to the spatial derivative of the conduction term would explicitly generate a funnelling term (dA_{Cl}/dz) due to the

variable area:

$$\frac{\partial}{\partial z} \left(k A_{Cl}(z) \frac{\partial T}{\partial z} \right) = k \frac{dA_{Cl}}{dz} \frac{\partial T}{\partial z} + k A_{Cl}(z) \frac{\partial^2 T}{\partial z^2} \quad (4.2)$$

However, FRENETIC adopts a **staggered-grid Finite Volume Method (FVM)**. The FVM intrinsically bypasses the explicit computation of the dA_{Cl}/dz derivative by integrating the conservation laws over the control volume of the j -th axial node, bounded by the lower face at $z_{j-1/2}$ and the upper face at $z_{j+1/2}$. Applying the Divergence Theorem, the volumetric integral of the gradient is reduced to the exact fluxes crossing the control surfaces, and the conduction term becomes:

$$\int_{z_{j-1/2}}^{z_{j+1/2}} \frac{\partial}{\partial z} \left(k A_{Cl}(z) \frac{\partial T}{\partial z} \right) dz = \left(k A_{Cl} \frac{\partial T}{\partial z} \right)_{j+1/2} - \left(k A_{Cl} \frac{\partial T}{\partial z} \right)_{j-1/2} \quad (4.3)$$

Discretizing the temperature gradients, the algebraic balance equation for the axial conduction becomes:

$$\dots = k_{j+1/2} A_{Cl,j+1/2} \frac{T_{j+1} - T_j}{\Delta z} - k_{j-1/2} A_{Cl,j-1/2} \frac{T_j - T_{j-1}}{\Delta z} \quad (4.4)$$

This mathematical formulation demonstrates that the inlet and outlet areas ($A_{Cl,j-1/2}$ and $A_{Cl,j+1/2}$) are evaluated independently. Consequently, if the wrapper undergoes radial expansion, the code naturally multiplies the exiting flux by the widened area, ensuring strict mass and energy conservation without altering the source code of the principal solvers (e.g., `conduction.f90`, `clntenesolve.f90`).

Similarly, in the native 1D-axial momentum conservation equation (`momentumsolver.f90`), expressed in terms of the mass flow rate $\dot{m}$, can be written in its differential form as:

$$\frac{\partial \dot{m}}{\partial t} + \frac{\partial}{\partial z} \left(\frac{\dot{m}^2}{\rho A_{Cl}} \right) = -A_{Cl} \frac{\partial P}{\partial z} - \rho g A_{Cl} - F_{frict} \quad (4.5)$$

where the second term on the left-hand side represents the convective momentum flux, which is responsible for the Bernoulli effect.

Applying the Finite Volume Method (FVM), the equation is integrated over the spatial control volume of the j -th node, spanning from the inlet face $z_{j-1/2}$ to the outlet face $z_{j+1/2}$ with length Δz . The integral of the spatial derivative of the convective term yields the exact difference of the dynamic pressure (kinetic energy) terms evaluated at the nodal interfaces:

$$\int_{z_{j-1/2}}^{z_{j+1/2}} \frac{\partial}{\partial z} \left(\frac{\dot{m}^2}{\rho A_{Cl}} \right) dz = \left(\frac{\dot{m}^2}{\rho A_{Cl}} \right)_{j+1/2} - \left(\frac{\dot{m}^2}{\rho A_{Cl}} \right)_{j-1/2} \quad (4.6)$$

Consequently, the fully discretized momentum balance equation solved by the `momentumsolver.f90` module takes the following form:

$$\begin{aligned} \Delta z \frac{d\dot{m}_j}{dt} + \left[\left(\frac{\dot{m}^2}{\rho A_{Cl}} \right)_{j+1/2} - \left(\frac{\dot{m}^2}{\rho A_{Cl}} \right)_{j-1/2} \right] \\ = -\bar{A}_{Cl,j} (P_{j+1/2} - P_{j-1/2}) - \bar{\rho}_j g \bar{A}_{Cl,j} \Delta z - \bar{F}_{frict,j} \Delta z \end{aligned} \quad (4.7)$$

This formulation clearly shows that if the wrapper undergoes a physical expansion ($A_{Cl,j+1/2} > A_{Cl,j-1/2}$), the outgoing kinetic energy term decreases. To satisfy the momentum balance, the solver will intrinsically compute a pressure recovery ($P_{j+1/2} > P_{j-1/2}$, net of gravity and friction), thus natively capturing the Bernoulli effect without requiring the explicit formulation of a geometrical derivative dA/dz .

Finally, regarding the solid structures (`solvenonheated.f90`), it is known that the flowering phenomenon affects the macroscopic displacement of the hexagonal wrapper but does not alter the nominal diameter of the internal fuel pins. Therefore, the solid transverse area remains constant along the z -axis, requiring no modifications to the radial conduction solvers.

4.2 Dynamic Geometrical Updating Strategy

Since the FVM solvers are natively capable of handling variable flow areas, the implementation effort was entirely focused on the dynamic update of the macroscopic geometry prior to the iterative solver calls.

To prevent an infinite recursive accumulation error during the transient execution, whereby the expanded area would mathematically add to itself at each time step ($A^{t+\Delta t} = A^t + \Delta A$) causing numerical instability, a strict reference geometry was introduced. Nominal, undeformed variables for the flow area (`ACl_nom`) and wetted perimeter (`pwet_nom`) were declared as global pointers in the channel data structure (`data_chan.f90`) and allocated in memory. A static snapshot of the undeformed geometry is captured at $t = 0$ immediately after the input parsing in the `initia.f90` subroutine.

During the transient execution (`transient.f90`), a dedicated subroutine updates the local parameters by computing the time-dependent spatial deformation exclusively with respect to the nominal reference values, overwriting the dynamic vectors (`ACl`, `pwet`) passed to the thermal-hydraulic drivers.

4.2.1 Code Verification

Incorporating the *flowering* phenomenon into the analysis of the governing equations requires the evaluation of spatial and temporal variations in the coolant flow area.

To verify the correct implementation of the thermal-hydraulic numerical solvers and their response to a variable cross-section, a preliminary validation test was required to ensure strict **adherence to the mass conservation principle**. This test is conducted under zero thermal power conditions; consequently, the fluid density ρ remains constant, yielding the following continuity relation between the inlet and the outlet of the channel:

$$v_{\text{in}} A_{\text{in}} = v_{\text{out}} A_{\text{out}} \quad (4.8)$$

The adopted approach involves mathematically constraining the outlet area to be twice the inlet area ($A_{\text{out}} = 2A_{\text{in}}$), which must intrinsically result in an outlet velocity equal to half of the inlet velocity. The execution of this test is facilitated by introducing a temporary linear expansion profile to calculate the updated coolant flow area. Reflecting the staggered-grid Finite Volume Method (FVM) adopted by FRENETIC, the computational procedure is divided into two distinct steps:

1. Evaluation at Nodal Interfaces: A primary loop iterates over the $N + 1$ faces of the control volumes (where N is the total number of nodes). A normalized spatial coordinate, acting as a linear expansion factor $f_{\text{exp}}(z) = z/H_{\text{tot}}$, is computed. This factor smoothly interpolates between 0 at the channel inlet ($z = 0$) and 1 at the channel outlet ($z = H_{\text{tot}}$). The local flow area is thus evaluated as:

$$A_{\text{Cl}}(z) = A_{\text{Cl,nom}}(z) \cdot [1 + f_{\text{exp}}(z)] \quad (4.9)$$

2. Evaluation at Nodal Centers: A subsequent loop iterates over the N node centers. Relying on the same linear expansion factor evaluated at the mid-point of each node, the variation in the wetted perimeter P_{wet} is proportionally updated. Consequently, the local cell volume and the hydraulic diameter are strictly re-evaluated to guarantee geometric coherence.

To rigorously verify the correct implementation of the *flowering* process, a dedicated test case designated as `TH_1D_standalone_flowering` was developed. This configuration is directly derived from the `TH_1D_standalone_transient` case, with the sole modification being the applied thermal power. A negligible value of 1 W was imposed instead of absolute zero to prevent numerical instabilities, such as the generation of NaN (Not-a-Number) values, which would prematurely terminate the simulation. Furthermore, to allow for a direct assessment of mass conservation, specific output instructions were embedded within the `transient.f90` subroutine to track the simulated inlet and outlet areas alongside their corresponding fluid velocities.

The computational results, summarized in Tab. 4.1, demonstrate that the numerical code accurately resolves the continuity equation under variable cross-section conditions, successfully validating its capability to simulate the *flowering* phenomenon.

Parameter	Value	Unit
Inlet Area, A_{in}	$1.13 \cdot 10^{-2}$	m^2
Outlet Area, A_{out}	$2.26 \cdot 10^{-2}$	m^2
Inlet Velocity, v_{in}	1.44	m/s
Outlet Velocity, v_{out}	0.72	m/s

Table 4.1: Output obtained from the test case `TH_1D_standalone_flowering`.

4.3 Thermo-Mechanical Coupling Implementation

Following the successful verification of the thermal-hydraulic solvers under variable cross-section conditions, the framework was expanded to establish a functional coupling between the TH module and the external Thermo-Mechanical (TM) solver. The objective was to replace the analytical area expansion used for testing with the actual structural deformations computed dynamically by the TM code.

4.3.1 Data Exchange Strategy and Modularity

To preserve the modularity of the overall FRENETIC architecture, a file-based data exchange protocol was adopted. At each coupling iteration, the TM module computes the wrapper thermal bowing and irradiation swelling, exporting the updated geometric parameters into specifically formatted text files: `TM.New_Areas.txt` and `TM.Displacements.txt`. The former contains the deformed cross-sectional flow areas mapped over the axial nodes and radial channels, while the latter provides the updated macroscopic spatial coordinates (X and Y) of the assemblies.

To manage this bidirectional data flow, a dedicated part in the Fortran module named `TH_Coupling_Manager_Mod` was developed. Centralizing the multi-physics communication within a specialized module ensures a clean software architecture, allowing the native input/output managers to remain focused exclusively on the standard simulation outputs, such as the HDF5 output files.

Within this module, the `TM_Coupling_Read` subroutine was implemented to parse the text files and overwrite the internal geometric arrays of the TH module. To guarantee computational robustness during the simulation, the reading routines are protected by input/status checks (`iostat`). In the event of file synchronization delays or unreadable data, the subroutine issues a warning and gracefully retains the geometry of the previous time step, thereby preventing fatal execution crashes.

4.3.2 Execution Sequence and Transient Integration

The physical consistency of the coupled simulation heavily relies on the precise positioning of the data exchange calls within the temporal integration loop. To ensure that the fluid dynamics natively respond to the structural deformations, the `TM_Coupling_Read` subroutine is invoked at the very beginning of the time step Δt , inside the main `transient.f90` program. The execution follows a strict chronological sequence:

1. **Geometry Update:** The TH code reads the TM output files and updates the flow area arrays (`dt%AC1`) and the geometric coordinates (`dt%XCOORD`) for all the spatial nodes.
2. **Fluid Dynamic Solution:** The native momentum solvers are executed. Operating on the newly deformed geometry, the code computes the local velocity variations

necessary to satisfy mass conservation and natively evaluates the dynamic pressure gradients (the Bernoulli effect) induced by the variable area.

3. **Thermal Solution:** The energy solvers compute the updated temperature fields based on the newly evaluated mass flow rates.
4. **Feedback to TM:** At the end of the time step, the updated coolant and structure temperatures are extracted and formatted by a complementary subroutine (`TM_Coupling_temp`), providing the necessary boundary conditions for the subsequent TM calculation.

Finally, to maintain backward compatibility and preserve the ability to execute the analytical standalone verification tests, a specific execution flag (`isCoupledTM`) was integrated into the initialization namelists. Evaluated prior to the transient execution, this flag dynamically routes the computational flow: if the coupling is active, the geometry is driven by the TM interface files; if inactive, the solver bypasses the file reading and relies on the previous values obtained.

Chapter 5

TH Standalone Cases Analysis & Results

The thermal-hydraulic (TH) module of the FRENETIC code solves the 1D axial transient advection, diffusion, and heat conduction equations for the coolant, coupled with the heat conduction equation inside the fuel pins. To assure the robustness and physical accuracy of the solvers prior to the full multi-physics coupling, a comprehensive set of standalone TH tests was developed and integrated into the continuous integration pipeline of the code. The latter represents an automated workflow that systematically compiles the code and runs predefined tests every time a new modification is committed to the repository. This practice ensures that ongoing developments do not inadvertently break existing functionalities or compromise the numerical stability of the solvers.

Four primary standalone testing scenarios were modelled to evaluate different aspects of the numerical solvers and physical correlations:

1. `TH_3D_standalone_transient`
2. `TH_1D_standalone_transient`
3. `TH_3D_pow_transient`
4. `TH_1D_standalone_CIRCE`

Once the various standalone cases for the thermal-hydraulic (TH) module were successfully set up, the post-processing phase was conducted to definitively verify the successful validation of the code following the extensive restructuring of the `main.f90` program. To efficiently handle the large volume of multi-dimensional data generated during the simulations, the TH module exports the results using the Hierarchical Data Format version 5 (HDF5) [21]. These `.h5` files are subsequently read and processed via dedicated Python scripts, allowing for a structured extraction, analysis, and visualization of the key physical quantities.

The successful execution and data extraction for all the test cases confirmed that the

generalized restructuring of the code was implemented correctly without compromising its fundamental functionalities.

The COREutils Library and Input Generation To streamline the simulation workflow, the COREutils Python library [19] was employed. This library acts as a comprehensive pre- and post-processing framework. By leveraging `.json` configuration files, COREutils automatically generates the necessary input files and manages the data extraction for all the simulated cases. It is important to note, however, that the actual execution of the simulation is not autonomously orchestrated by COREutils itself. Instead, the run is driven by a dedicated Python binding script (such as `frenetic.py`), which links and interfaces directly with the compiled Fortran source code utilizing `f2py` (a widely used Fortran-to-Python interface generator) [22].

The `.json` files are structured as a system of nested dictionaries (key-value pairs), categorized into specific sections to initialize different sets of input data. The primary dictionaries utilized in this framework are:

1. **CI (Common Input):** This section manages the creation of the `common_input.inp` file. It contains the overarching parameters required to define the simulation, such as the transient execution flag, the initial power level, the total simulation time, and time-step sizes.
2. **GE (Geometry):** This dictionary is responsible for the core geometry initialization. It typically reads an external `.txt` file that provides the core map layout. Subsequently, all the primary geometrical and physical characteristics of the individual assemblies comprising the core are defined within this section.
3. **TH (Thermal-Hydraulics):** This section stores all the specific information, parameters, and boundary conditions concerning the thermal-hydraulic module. In fully coupled simulations, this dictionary is usually accompanied by a Neutronic (NE) dictionary; however, since the test matrix consists exclusively of TH standalone cases, the NE dictionary is inherently omitted.
4. **FRENETIC-NML (Namelists):** This dictionary handles any additional parameters required to fine-tune the simulation. It allows the user to insert specific namelist keywords that must override the code's default values for the case study under analysis (e.g., `HeatingType`, `tPowEnd`, `isNETH`, etc.).

5.1 1D Standalone Transient

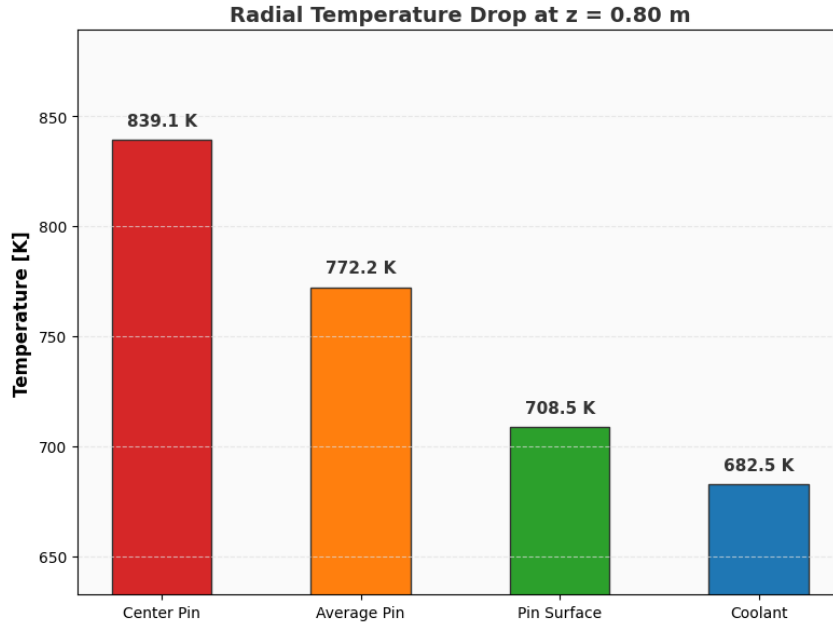
In the 1D case (single channel), the computational domain is reduced to a single representative hexagonal assembly. Hence, the variables that were modified are: `nR` = 1 and `nChan` = 1; and, again, the values of `isneth` and of the power (those were left as equal to the 3D case).

5.1.1 Results & Post-processing

The first case analyzed, `TH_1D_standalone_transient`, aimed to validate the code on a single, isolated generic fuel assembly (1D standalone) composed of a bundle of UO_2 fuel pins cooled by liquid lead. During the initial setup, a critical issue was identified within the Python-Fortran interface (`COREutils`). The Fortran namelist strictly requires the linear power to be passed using the `Pow0` keyword (case-sensitive). An initial bug in the `InpGen.py` script used a lowercase key, causing the input generator to drop the value and pass a default power of 0.0 W to the Fortran solver, effectively leaving the reactor in a "cold" state.

Once the parsing bug was fixed, the solver accurately computed the heat flux generation (`q_pin`). To verify the thermodynamic consistency of the model, a stationary simulation was executed over a 1.0 m active channel length with a constant thermal power of 300 kW and an inlet mass flow rate of 200 kg/s. By applying the steady-state calorimetric enthalpy balance for incompressible fluids ($\Delta T = \frac{Q}{\dot{m} \cdot c_p}$), a bulk coolant temperature rise of approximately 10 K was expected.

The resulting post-processed graphs (Fig. 5.1 and 5.2) perfectly capture this physical behavior. The axial profiles show a consistent, linear rise in the coolant temperature from 673 K to 683 K, with the cladding and fuel centerline temperatures maintaining the expected parallel gradients under uniform heating. Furthermore, the radial temperature profile extracted at $z = 0.8$ m accurately resolves the intra-pin heat transfer. It clearly displays the characteristic parabolic temperature distribution within the UO_2 pellet, driven by internal volumetric heat generation, followed by the conductive temperature drops across the gap and cladding, down to the bulk coolant temperature.

Figure 5.1: 1D Radial Temperature variation at $z = 0.80$ m.

Evolution of different temperatures in the central channel (FA n.1)

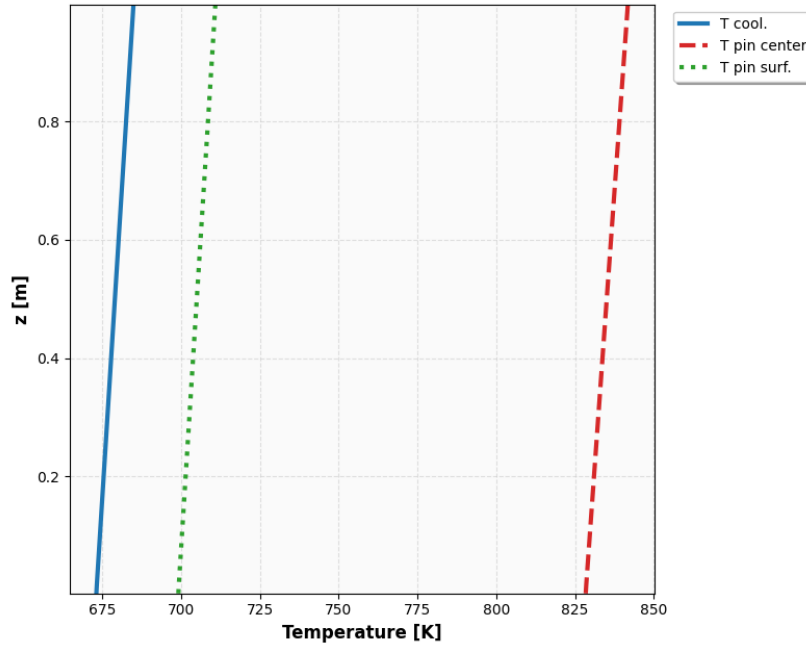


Figure 5.2: 1D Temperature Profiles.

5.2 3D Standalone Transient

To model the 3D case (multi-channel full core) the input and data values were taken from the coupled test case called NETH3D. The parameters that were changed were: the variable `isneth` triggering the coupling was set equal to 0 in order to represent the standalone case;

and, since there was not more the coupling with the neutronic module, it was hypothesized a power value of 10 kW (using `HeatingType` = 0 in order to represent the square wave source).

5.2.1 Results & Post-processing

To increase the geometrical complexity, the validation was extended to a `TH_3D_standalone` model. This case simulates an entire reactor core layout, mapping different regions such as the Inner Fuel (IF), Outer Fuel (OF), and Dummy/Reflector (DR) assemblies. A major challenge encountered during this phase was related to severe numerical instabilities. Initial runs resulted in immediate crashes, with the solver reporting contradictory errors, such as fuel melting at temperatures exceeding 3000 K and lead freezing simultaneously in different core regions. Through a detailed physical analysis, the root cause was identified as a localized divergence in the flow distribution solver. This was primarily exacerbated by the inadvertent activation of the *flowering* testing subroutine, which forced unrealistic geometric expansions and locally stalled the mass flow rate in specific channels. Furthermore, the extreme mismatch between the massive nominal flow rates prescribed in the `.json` input file and the initial mere test power of 10 kW pushed the lead properties out of their valid correlation range ($T > 1300$ K) whenever the coolant flow stagnated.

To resolve this issue and properly demonstrate the code's capability to simulate 3D domains, the artificial geometric expansion was deactivated, restoring the nominal flow areas. Concurrently, the total core power was scaled up to 1.25 MW while maintaining a stable inlet mass flow rate of 180 kg/s per channel over a 1.0 m active length. While this power-to-flow ratio is significantly lower than nominal fast reactor operations, it provided a safe heat generation level to test the 3D solver without hitting physical material limits.

Under these corrected conditions, the code successfully reached steady-state convergence. Due to the extremely high mass flow rate relative to the applied power, the overall thermal gradient across the core was physically minimal ($\Delta T < 1$ K). Remarkably, as shown by the post-processed radial and axial temperature distributions ((Fig. 5.3 and 5.4), the solver is capable of resolving even these minute intra-pin and coolant temperature variations with high numerical precision. The code accurately handles the multi-channel flow distribution and zone-specific heat transfer correlations, natively capturing the thermal micro-gradients between the highly heated Inner Fuel region and the cooler peripheral assemblies.

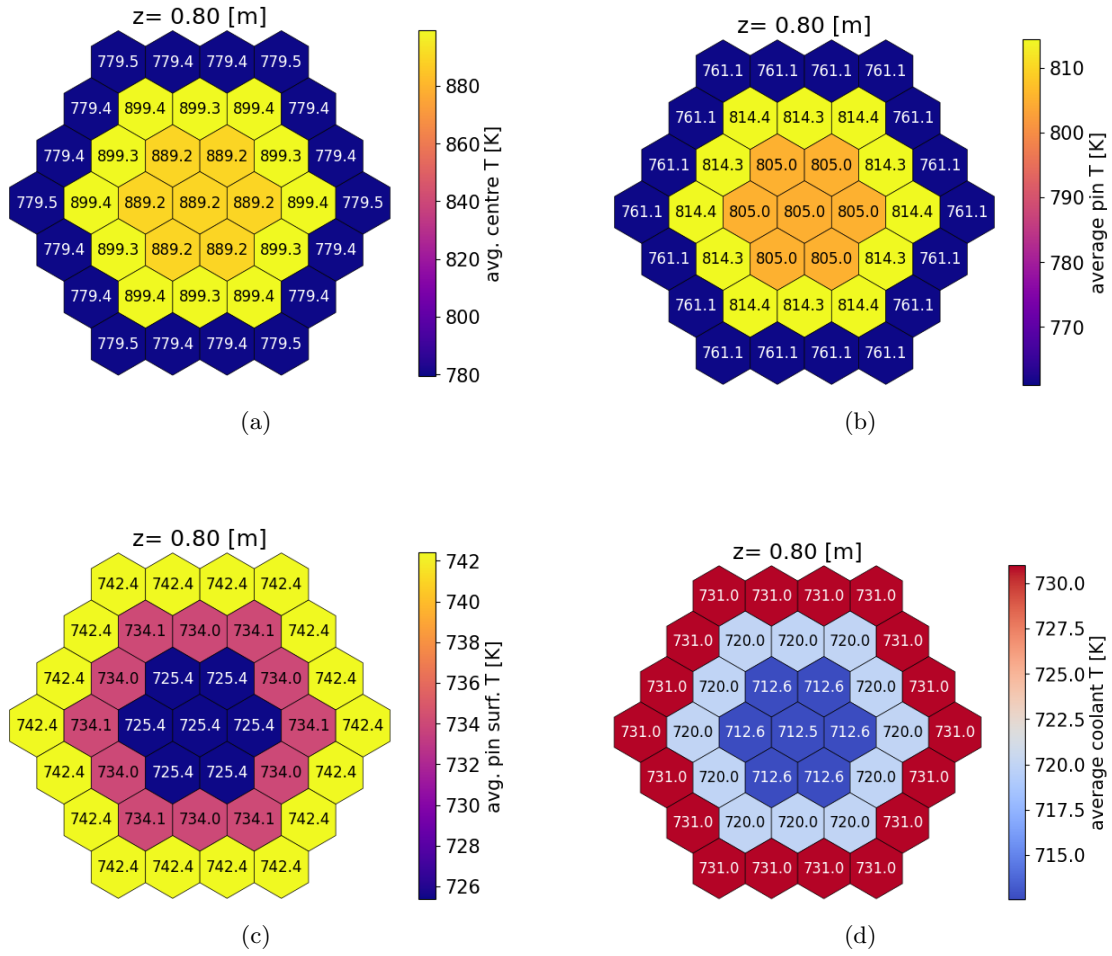


Figure 5.3: Radial Temperatures Variation in a 3D domain.

Evolution of different temperatures in the central channel (FA n.1)

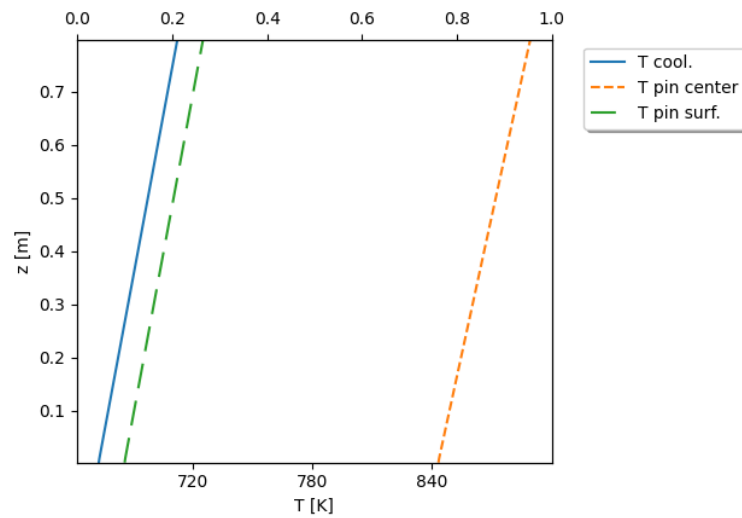


Figure 5.4: Axial Temperature Profiles.

5.3 3D Power Transient

In this scenario, the heat generation is decoupled from the neutronic module and is instead driven by an external, time-dependent forcing function (`HeatingType = -1`).

The power evolution $Q(t)$ is provided via an external input file (`power.inp`), which specifies how the power transient is defined for specific time instants. The code interpolates these discrete points to impose a continuous and dynamic heat flux $q''(z, t)$ at the cladding-coolant interface.

5.3.1 Results & Post-processing

To further test the dynamic and transient capabilities of the thermal-hydraulic module, in this test case, the `HeatingType` parameter was modified to read a time-dependent power history from an external `power.inp` file.

The simulated transient was set up with a starting baseline power of 1.0 MW/m at $t = 0.0$ s, followed by a power ramp up to a peak of 1.5 MW/m at $t = 5.0$ s, and a sudden drop to 0.0 MW/m at $t = 10.0$ s, simulating a SCRAM (emergency shutdown) event. To accurately reproduce the thermal-hydraulics of a Lead Fast Reactor (LFR), the inlet mass flow rate was adjusted to 30.0 kg/s. This ensured a physically realistic temperature distribution across the 0.8 m active channel length. To properly capture the rapid dynamic response, particularly the power peak at $t = 5.0$ s, the solver's temporal resolution and the output saving frequency were strictly synchronized to a refined time step (`StpMax` and `dtAutoSaveTH` set to 0.1 s).

The plotted results visually confirm the thermal inertia and the dynamic response of the system. During the power peak at $t = 5.0$ s, Fig. 5.5 and 5.6 show a clear widening of the temperature gradient between the pin center, the pin surface, and the coolant. As the power decreases towards zero at $t = 10.0$ s, the temperatures begin to drop (see Fig. 5.7). Notably, the profiles at $t = 10.0$ s do not immediately collapse into a vertical isotherm at the inlet temperature (673 K). Instead, they clearly demonstrate the physical thermal inertia of the system: the UO_2 fuel pins retain residual heat, and the coolant temperature profile reflects the ongoing, non-instantaneous heat transfer during the shutdown phase.

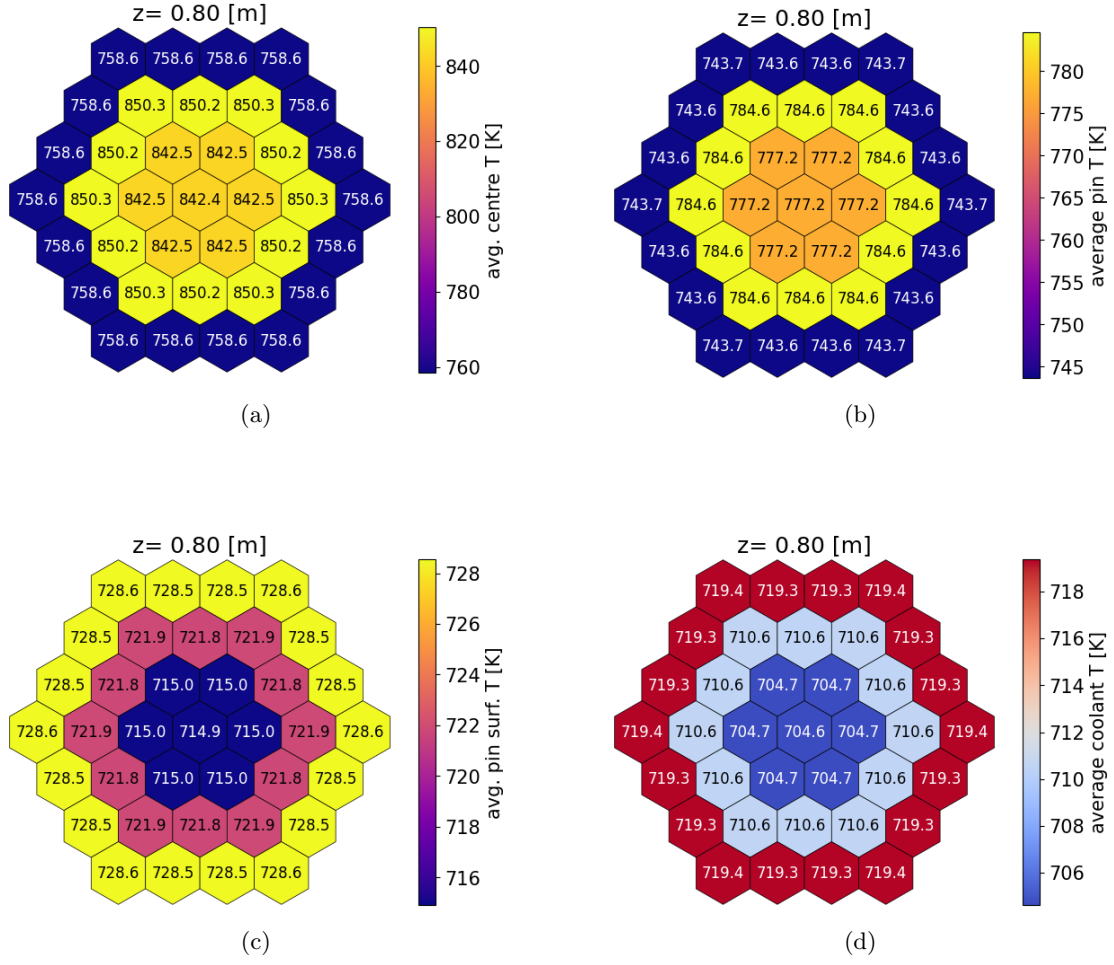


Figure 5.5: *Radial Temperature Variation in a 3D domain with power transient.*

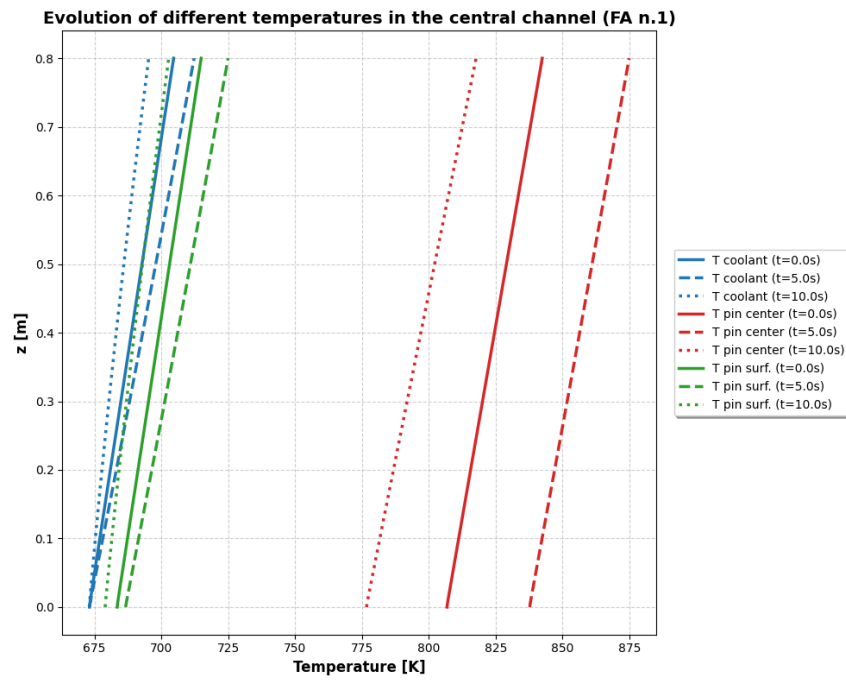


Figure 5.6: Axial Temperature Profiles in the power transient.

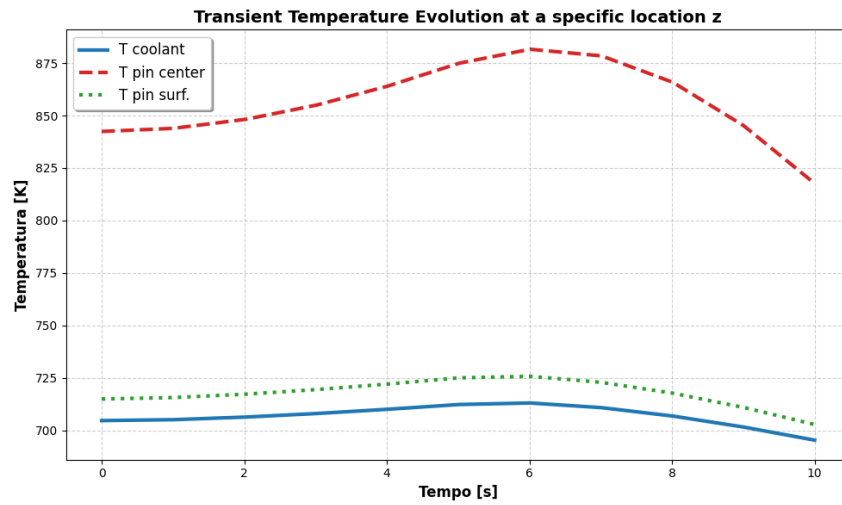


Figure 5.7: Temperature Evolution during the transient.

5.4 1D Standalone CIRCE: ICE Facility Validation

To move beyond numerical verification and provide an experimental validation of the code, a specific 1D model was created based on the Integral Circulation Experiment (ICE) performed in the CIRCE pool facility at the ENEA Brasimone Research Centre [5].

The domain of the experiment represents the Fuel Pin Simulator (FPS), which consists of an electrically heated bundle of 37 pins arranged in a hexagonal lattice (see Fig. 5.8). The primary parameters implemented in the model accurately reflect the facility design:

- Coolant: Lead-Bismuth Eutectic (LBE) with temperature-dependent thermophysical properties.
- Cladding Material: Stainless Steel 316 (SS316).
- Geometry: Pin outer diameter $D = 8.2 \text{ mm}$ and pitch-to-diameter ratio $P/D = 1.8$.
- Active Length: $L = 1000 \text{ mm}$.

Given the specific lattice geometry, the convective boundary condition between the cladding and the LBE was modelled using the *Ushakov* correlation for rod bundles that is used to evaluate the Nusselt number as follows:

$$Nu = 7.55 \left(\frac{P}{D} \right) - 20 \left(\frac{P}{D} \right)^{-13} + \frac{0.041}{(P/D)^2} Pe^{0.56+0.19(P/D)} \quad (5.1)$$

The transient reproduces the first full-power characterization test of the ICE campaign. The boundary conditions impose a constant mass flow rate of 60 kg/s and an initial inlet temperature of 570 K (obtained from the graph presented in [23]). An initial power ramp drives the FPS from 0 to 800 kW (its nominal power) over a period of 300 seconds, eventually reaching steady-state natural or assisted circulation.

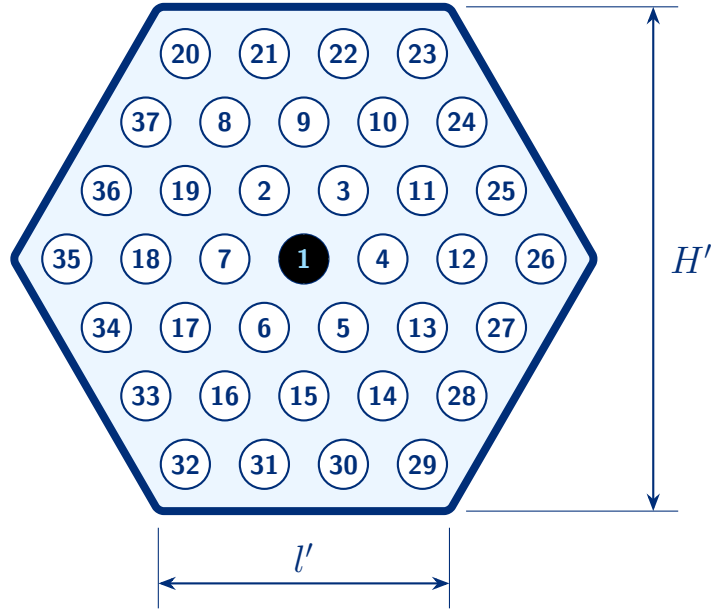


Figure 5.8: *Cross-section of the Integral Circulation Experiment (ICE) heat source bundle [5].*

5.4.1 Results & Post-processing

This final standalone case is crucial as it provides a benchmark against realistic experimental operating conditions for Heavy Liquid Metal (HLM) systems, utilizing Lead-Bismuth Eutectic (LBE) as the primary coolant. The simulation was set up with a nominal thermal power of 800 kW and an LBE mass flow rate of 60.0 kg/s, with an initial inlet temperature of 570 K. To rigorously test the dynamic capabilities of the refactored code, a time-dependent boundary condition (BC) perturbation was introduced via the `.json` interface. The system was maintained at steady-state conditions for the first 500 s, followed by a linear ramp of the inlet temperature up to 580 K (a +10 K variation) over the subsequent 500 s.

The post-processing script successfully extracted both the radial and temporal temperature profiles, validating the physical and numerical consistency of the solver. The radial temperature distribution at $z = 0.8$ m accurately models the heat transfer mechanics, showing the steep conductive drop from the UO_2 fuel centerline (peaking at approximately 950 K) to the bulk LBE coolant, while confirming an overall axial coolant temperature rise (ΔT) of about 90 K across the active channel (see Fig. 5.9). Furthermore, the transient temperature evolution graph captures the imposed boundary condition perturbation, comparing the computed results directly with the experimental reference data (see Fig. 5.10). Regarding the reliability of these reference measurements, the official ENEA report specifies the use of N-type thermocouples with an isolated hot junction, guaranteeing a high measurement accuracy of $\pm 0.1^\circ\text{C}$ for sensors installed within the Fuel Pin Simulator (FPS) and $\pm 1^\circ\text{C}$ for the remaining facility probes [24]. As depicted in the plot, the computed temperatures remain stable during the initial 500 s phase, followed by a

synchronized, linear temperature increase across all radial nodes (coolant, cladding, and fuel) driven by the inlet temperature ramp, demonstrating a highly consistent agreement with the experimental facility’s behavior.

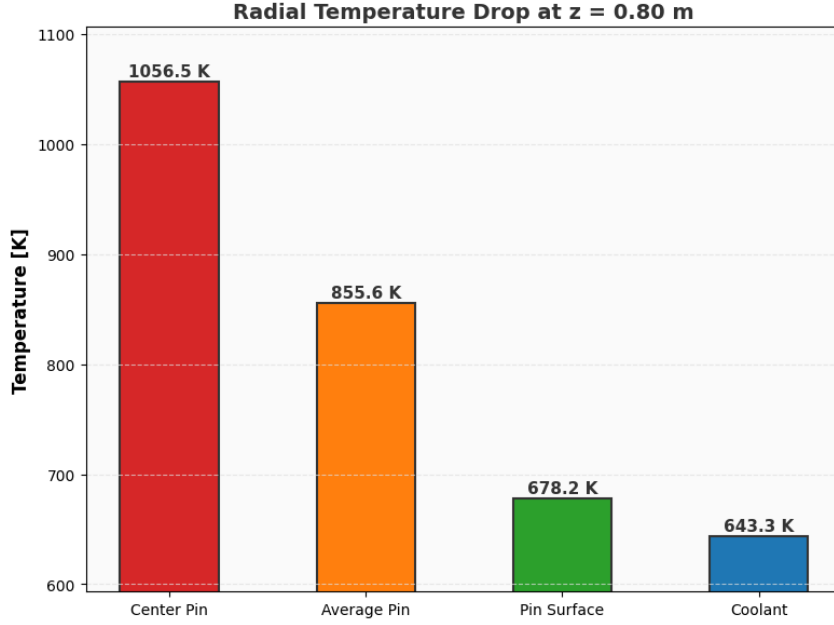


Figure 5.9: *Radial Temperature Variation at different $z = 0.80$ m.*

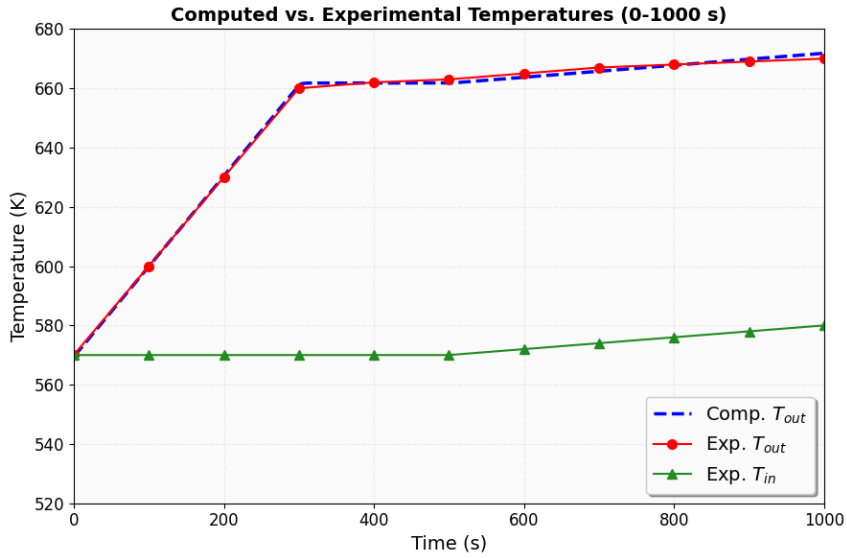


Figure 5.10: *Temperature Profiles evolution in time.*

The systematic execution and analysis of the four standalone test cases definitively demonstrate the successful validation of the FRENETIC thermal-hydraulic module. The extensive structural refactoring of the `main.f90` core program and its associated subroutines was accomplished while rigorously preserving the numerical stability and physical accu-

racy of the solver. Furthermore, the debugging of the Python `COREutils` interface ensured a robust, automated, and error-free pipeline for input generation and data extraction. By accurately simulating steady-state thermodynamics, dynamic power transients featuring distinct thermal inertia, and time-dependent boundary perturbations across both 1D simplified geometries and complex 3D core maps, the primary objectives of this Master's thesis project have been fully met. The refactored TH module is now comprehensively verified, stable, and structurally prepared for the upcoming multi-physics coupling with external system codes, specifically ATHLET.

Chapter 6

Coupling FRENETIC-TH and ATHLET

6.1 Python Library for ATHLET

To perform the coupling between the thermal-hydraulic module of FRENETIC and the ATHLET system code, as described in Chapter 2, a Python interface is employed. In this framework, the system code is treated as a *shared library*.

The ATHLET Python extension, distributed by GRS in the AC2 package, is organized into six primary scripts:

1. `__init__.py`: This file defines the directory as a proper importable Python package, allowing the use of the `import athlet` command. It handles the initial setup and imports the various modules related to the *controllers*.
2. `_controller.py`: This script is responsible for loading the shared library (`.so` file), mapping ATHLET's internal variables to Python objects, and providing the necessary commands to advance or pause the simulation. It also defines the aliases for the Hooks required in the coupling routine. Furthermore, it manages time-step control: since ATHLET can reject a time-step if convergence is not fully reached, this module forces the code to discard the latest calculation and restore a previous state from memory.
3. `_protocolController.py`: An extension of the base controller, its primary purpose is to allow the user to provide a Python-formatted text file. Rather than merely being read at the end of the steady-state calculation, this file is specifically executed by ATHLET to define the sequence of events, dynamically update the boundary conditions, and provide the necessary physical inputs to drive the subsequent transient phase of the simulation. It contains two dictionaries: one for the controller's default options and another listing the three types of objects managed by the system (`Variable`, `Trigger`, `Action`).

4. `_protocolPrinter.py`: This script functions as a diagnostic and tracking system. It monitors ATHLET's memory and prints the internal state in a human-readable format. This is particularly useful for verifying that *triggers* are activating at the correct time and that *variables* are assuming the expected values.
5. `main.py`: This file serves as the command-line entry point for ATHLET. It parses terminal commands, prepares the necessary files, initializes the controllers, and launches the simulation. The execution logic relies on the instruction `Controller(args.pop('--lib'), **args).start( args=startArgs)`, which calls the `Controller` class, provides the path to the shared library for RAM loading, reads the input file, and starts the transient.
6. `controllers.py`: This script acts as a central hub, grouping all controller-related files through `import` statements for easier access.

Once the structure of the shared library and the coupling commands are established, ensuring robust time-step management becomes a priority. Although implicit schemes theoretically offer higher numerical stability through iterations, an **explicit coupling** approach was ultimately selected for this project. This decision is primarily motivated by the inherent complexity of synchronizing two independent solvers.

Specifically, attempting to force an implicit convergence between FRENETIC and ATHLET would introduce significant computational overhead, primarily due to the challenging adaptation of the internal time-steps required by each code. By adopting an explicit scheme instead, the two solvers are permitted to exchange data strictly at predefined, discrete synchronization points. This strategy allows each module to preserve its internal, optimized numerical advancement without the burden of resolving iterative loops at the interface, a choice that directly guided the specific restructuring of the main execution script. Nevertheless, the implementation of a fully implicit coupling architecture remains a valuable objective for achieving ultimate numerical robustness, and it is therefore earmarked as a natural progression for future developments and code enhancements.

6.2 Modification of the FRENETIC *main.py* Script

Since the coupling methodology relies on a Python-based architecture, the Python version of FRENETIC, using the `f2py` interface [22], is employed. To enable a seamless integration while preserving code maintainability, the thermal-hydraulic driver (called `runTH`) within the FRENETIC `main.py` file had to be significantly refactored. A key modification was the introduction of a boolean flag named `isCoupledAthlet`. Depending on whether it is set to `True` or `False`, this flag dictates if the code must run in explicitly coupled mode or in standard standalone mode (analogous to the `isNETH` flag utilized in FRENETIC for the internal neutronics/thermal-hydraulics coupling).

To prevent the main execution script from becoming overly convoluted and to strictly adhere to the software engineering principle of *Separation of Concerns* [25], the entirety of the ATHLET coupling logic was extracted from `main.py` and encapsulated into a dedicated external module named `athlet_coupling.py`. When the coupling flag is set to `True`, the main script bypasses the standard monolithic execution and delegates the control to this external module. The module is responsible for initializing the `Controller` class from the `PyATHLET` interface, directly linking the compiled Fortran/C shared library (`libathlet.so`) into the Python runtime memory. Furthermore, within this module, a dedicated synchronization manager (`sync_manager`) and a stability monitoring function were introduced to rigorously handle a robust text-file-based explicit data exchange and to dynamically assess the numerical health of the coupled simulation.

Before fully integrating the two codes, ATHLET’s capability to operate as a shared library driven by Python was tested through a preliminary standalone script. This script executed a fictitious thermal transient, demonstrating the robustness of the FEBE solver under dynamic memory manipulation. Most importantly, this preliminary phase served to validate the *hook* mechanism, which constitutes the cornerstone of the explicit coupling architecture [26].

6.2.1 The Concept of Callback Functions (Hooks)

In software engineering, a **hook** is a mechanism that allows an external routine to intercept, monitor, or modify the execution flow of a primary software component [26]. Within the context of ATHLET, the Fortran backend is designed to execute a continuous, monolithic time-advancement loop. Without hooks, Python would only be able to start the simulation and wait for its conclusion. Hooks solve this problem by establishing predefined “interruption points” within the Fortran source code.

When the Fortran solver reaches one of these specific points (an event), it temporarily pauses its execution and “calls back” the Python interpreter, asking if there are any custom instructions to execute. Once the Python function completes its task, control is returned to Fortran, which resumes the numerical integration. This mechanism is paramount for coupled simulations, as it allows external codes to inject data, extract variables, or alter boundary conditions precisely when it is physically and numerically appropriate.

As mentioned previously, to verify the capabilities of the hook mechanisms, implemented in the Python-ATHLET library distributed by GRS, a representative test bench was configured in ATHLET creating two codes: one for the execution of ATHLET alone, one for the post-processing of the results.

Setup of the ATHLET Test Bench The test bench is modeled as a Simple Closed Loop (see Fig. 6.1). The main goal of this input is not to represent a complex, large-scale nuclear power plant, but to be used simply as an example test to understand whether

the definition of hooks for ATHLET works correctly or not, and to test the coupling implementation. The working fluid selected for the loop is liquid Lead (LEAD). The geometry is modeled using a main PIPE Thermo-Fluid Object (TFO) divided into multiple control volumes to simulate the circuit’s piping. The boundary conditions are governed by two specific components:

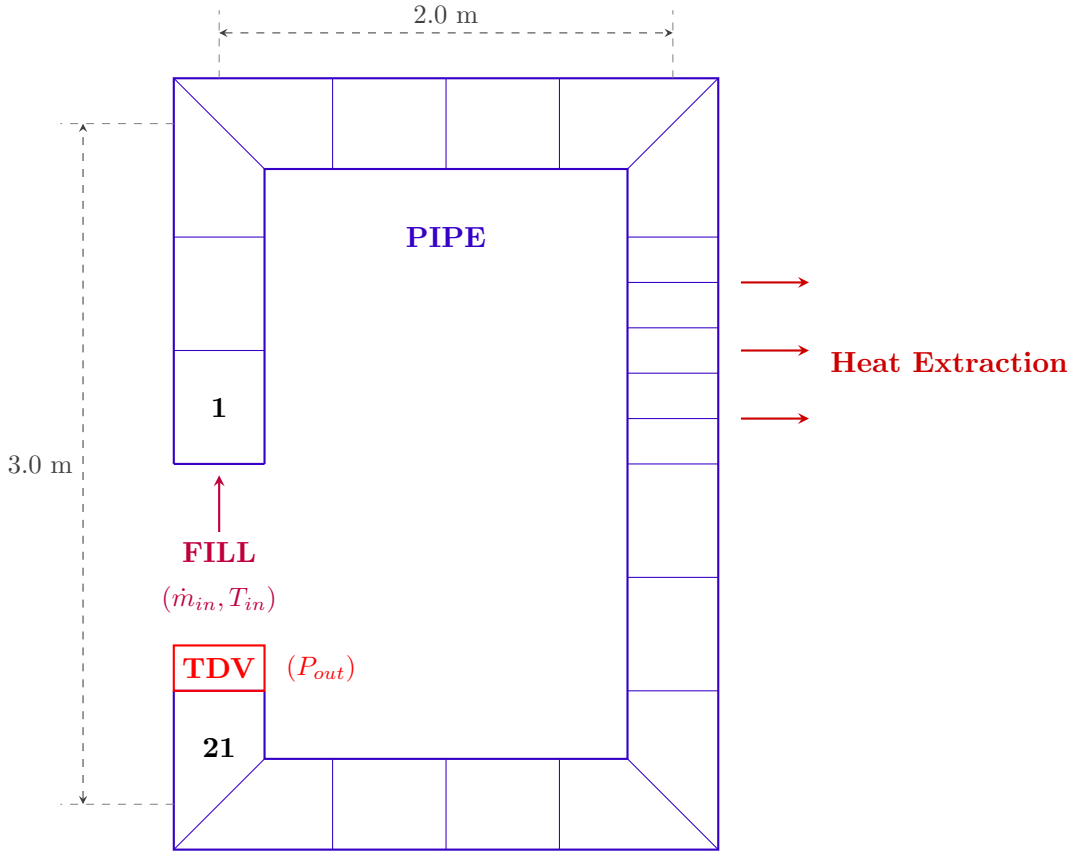
- A FILL component at the inlet, which regulates the mass flow rate and fixes the inlet fluid temperature at 600 °C.
- A Time Dependent Volume (TDV) at the outlet, which acts as a pressure boundary condition, maintaining the system pressure at $1 \cdot 10^5$ Pa.

A fundamental aspect of the test bench setup is the thermal exchange modeling. In the current simplified representation, Heat Conduction Objects (HCOs) are not implemented. Instead, a total power of 100 MW is directly extracted from the fluid, integrated across five dedicated control volumes located on the right side of the loop. Within the framework of the coupled simulation, this extracted power is designed to dynamically follow the core power distribution evaluated by the FRENETIC solver.

To achieve a rigorous steady-state solution at the onset of the simulation, the ZEROTRANS pseudo-keyword is utilized in the ATHLET input deck [15]. This command forces the solver to perform a zero-transient calculation—a simplified transient run executed with constant boundary conditions that strictly follows the initial Steady-State Calculation and precedes the actual transient simulation. By simulating an initial stabilization phase of 3000 s in the negative time domain (from $t = -3000$ s to $t = 0$ s), the system’s thermal-hydraulics, flow losses, and pressure drops naturally stabilize without the external perturbations of the active transient.

As indicated in the ATHLET User Manual [15], executing this intermediate stabilization phase in the negative time domain is not merely a precautionary measure, but a highly recommended approach for complex topologies where the standard SSC alone might struggle to reach strict convergence criteria. This is particularly relevant for pool-type, liquid-metal-cooled reactors like ALFRED. In such systems, starting the simulation from zero initial mass flows can induce heavy pressure waves that exceed the approximation range of the fluid property tables during a standard SSC.

Consequently, this simplified intermediate step ensures that when the active coupled transient commences at $t = 0$ s, the complex system has reached a true physical steady-state condition in terms of the ATHLET models, effectively avoiding initial numerical artifacts and ensuring the overall stability of the calculation.

Figure 6.1: *Sketch of the Simple Loop used as the test bench.*

Post-Processing To handle the ATHLET results, the `hdf5_writer` plugin is activated within the input deck, allowing the solver to export the transient data into a highly structured, hierarchical HDF5 (`.h5`) file.

A standalone Python script was developed to process this file, in order to extract, process, and visualize the output data. Relying on the `h5py` and `matplotlib` libraries, the script navigates the HDF5 tree structure to extract the necessary arrays. For instance, the macroscopic simulation time is retrieved from the `GENERAL` data block, while specific local variables, such as the fluid temperature (`TFLUID`), are extracted directly from the corresponding nodes of the `PIPE` object.

To effectively present the results and check the stabilization logic, the post-processing script generates two distinct visualizations:

1. *Full Transient Response*: A comprehensive plot illustrating the evolution of the target variables (e.g., fluid temperature) over the entire simulation timeframe.
2. *Stabilization Phase Zoom*: A targeted plot focused strictly on the `ZEROTRANS` phase, bounding the time axis from -3000 s to 50 s. A vertical marker at $t = 0$ s visually demonstrates the perfect equilibrium achieved by the system immediately before the transient onset.

In the fully coupled configuration, a similar post-processing approach will be extended to parse the 3D output arrays generated by FRENETIC, allowing for a comparative and coupled analysis of the macroscopic loop data against the detailed core sub-channel fields.

6.2.2 Code Architecture and Execution Flow

To accommodate the various numerical schemes of the two solvers while averting the computational overhead of iterative convergence, the explicit coupling algorithm is structured around an *Event-Driven* programming paradigm. In this master-slave configuration, ATHLET acts as the master clock.

A critical advancement in the code architecture is its modular design, developed to natively support the concurrent execution of the Neutronics (NE) and Thermal-Hydraulics (TH) modules via the Python `multiprocessing` library. The global execution is managed by the `runNETH` function, which spawns separate parallel workers for the NE and TH processes. Within the thermal-hydraulic worker, the `runTH` function acts as a dispatcher: evaluated against the `isCoupledAthlet` boolean flag, it either triggers FRENETIC-TH in standalone mode or redirects the execution by invoking the external `ATHLET_coupling()` function imported from the newly developed module.

This dedicated module receives the ATHLET input file and the initialized TH instance as arguments, encapsulating the entire ATHLET engine instantiation and the master-slave data exchange logic. To guarantee code robustness and prevent segmentation faults in memory, a **Text-File Coupling methodology** was adopted. The coupling information are securely exchanged via specifically formatted `.txt` files. The execution flow (see Fig. 6.2) is orchestrated through a specific set of callback functions (hooks) that manage the simulation across two fundamental phases:

- **Initialization and Input Processing:** This phase is monitored by the `ATHLET_AfterPreset` and `ATHLET_InputDone` hooks, which provide critical verification points when the system code begins and concludes the parsing of the `.in` file. A critical insight gained from the preliminary tests was the necessity of strategic hook registration: memory-dependent operations, such as verifying the allocation of Thermo-Fluid Objects (TFO) and Heat Conduction Objects (HCO), cannot be reliably performed until the input file is fully parsed. Consequently, the `ATHLET_InputDone` hook is essential to guarantee that physical topology is correctly mapped, and memory is fully allocated, right before the transient integration begins.
- **Explicit Time Integration and Synchronization:** This phase governs the core data exchange between the sub-channel code and the system code, driven exclusively by the `ATRANS_AfterFEBE` hook. To ensure an efficient and robust explicit coupling, the strategy unfolds in a highly controlled, sequential manner:
 1. *Time Accumulation:* Following each successful convergence of the ATHLET FEBE solver, the execution hook triggers and reads the internal macroscopic

- time-step (Δt). The Python supervisor accumulates these micro-steps into a dedicated variable (`accumulated_dt`) without interrupting FRENETIC, continuing until a predefined macroscopic synchronization target (`dt_sync`) is reached.
2. *FRENETIC Catch-up Phase:* Once ATHLET reaches the synchronization threshold, Python commands FRENETIC to advance its physical time to match ATHLET's actual time via the `advance_time_to_sync_point()` method. To adhere to the explicit operator-splitting paradigm, FRENETIC spans the entire accumulated interval utilizing the *old* frozen boundary conditions from the previous macro-step, thus averting non-physical paradoxes (i.e., using future conditions to compute present states).
 3. *Text-File Data Exchange:* With both solvers perfectly aligned in the temporal domain, the formal data exchange occurs. ATHLET evaluates the macroscopic fluid variables and writes them into a `athlet_to_frenetic.txt` file. Concurrently, FRENETIC calculates the updated heat source, integrates it to obtain the scalar total core power, extracts its own core related quantities, and writes them into a `frenetic_to_athlet.txt` file.
 4. *Stability Monitoring:* Before applying the newly generated conditions, a custom modular function named `evaluate_stability()` assesses the numerical stability of the exchanged quantities (e.g., the ATHLET outlet temperature). This function employs a dynamic rolling window to compute smoothed macro-derivatives, assessing the relative variation amplitude. An instability is mathematically detected if the product of the last two local derivatives is strictly negative, indicating a numerical oscillation.
 5. *Boundary Update and Data Injection:* FRENETIC is instructed to read the newly generated ATHLET text file, translating the macroscopic output states into its updated boundary conditions for the next integration step. Simultaneously, Python reads the FRENETIC text file, extracting the calculated core feedback variables. To ensure a robust data transfer, these values are dynamically supplied to the ATHLET General Control Simulation Module (GCSM) through the Python API. Specifically, the PyATHLET interface allows the user to directly connect to the internal signals representing the specific boundary conditions or thermal source terms of the case study. This permits the seamless overwriting of their values at each synchronization step according to the transient requirements, without being strictly tied to a single physical variable. Finally, the synchronization accumulator is reset to zero, the boolean flag is lowered, and the cycle awaits the next macro-step target.

Ultimately, the `engine_Athlet.start()` command transfers the primary loop control from Python to the ATHLET native solver. From this point forward, the Python script transitions into a background supervisor role, temporarily suspending the Fortran execution

solely to synchronize the physical time domains and exchange interface variables whenever the registered system hooks are triggered. To guarantee computational safety, the solver execution is wrapped in a `try-finally` block: regardless of whether the simulation completes successfully or encounters numerical exceptions, the `engine_Athlet.terminate()` method is used, ensuring proper memory cleanup and safe termination of the parallel processes.

The `athlet_coupling.py` script, featuring the explicit coupling scheme and stability monitoring logic, is presented in the Appendix part B.

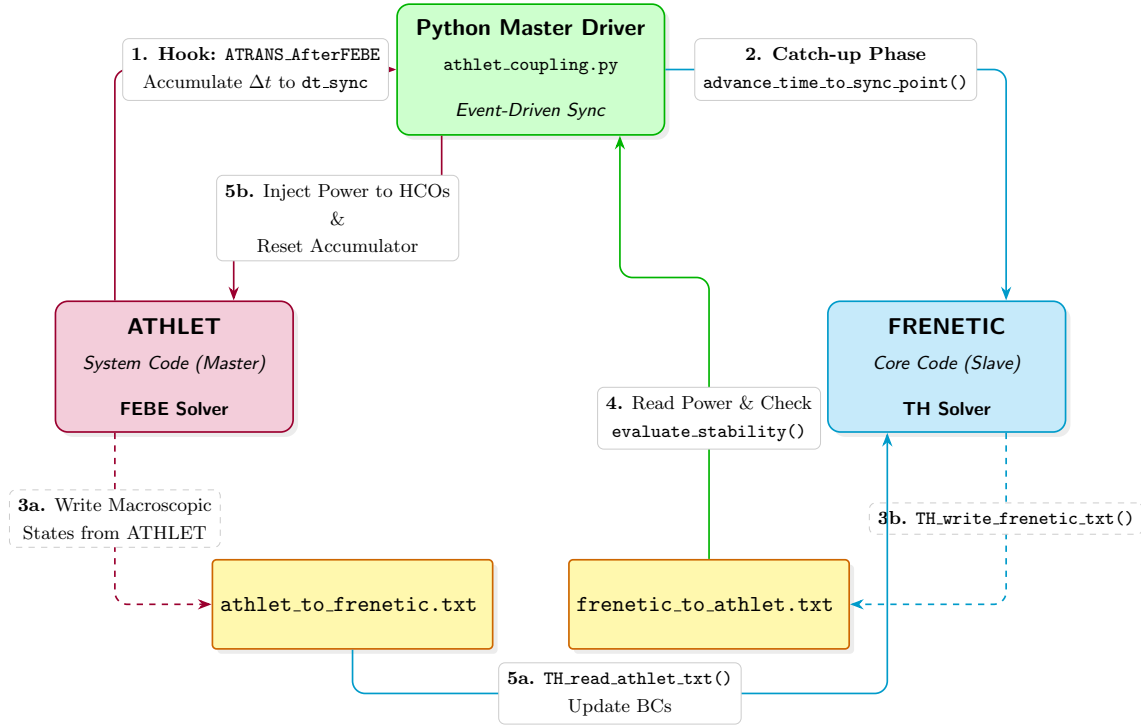


Figure 6.2: Workflow of the explicit coupling strategy and Event-Driven synchronization between ATHLET and FRENETIC.

6.3 Architecture of the Thermal-Hydraulic Driver Interface (*driver.py*)

To correctly bridge the gap between the high-level Python commands invoked in the `main.py` script and the actual mathematical routines computed by the Fortran backend, an intermediate Python interface module, named `driver.py`, was developed. This file acts as the core API (Application Programming Interface) for the thermal-hydraulic module, translating Python method calls into memory-safe execution commands directed to the compiled Fortran shared library (`_native.frenetic_th`).

The necessity of restructuring the `driver.py` file arose from the fundamental differences between executing a standalone simulation and performing an explicit coupling. In a

standard standalone run, the thermal-hydraulic module handles its own time integration continuously from $t = 0$ to t_{end} , i.e. the final simulation time. Conversely, during an explicit coupling with ATHLET, the system code takes over as the "master clock," dictating the time advancement. Rather than demanding iterative recalculations and interface convergence loops, the explicit scheme requires FRENETIC to autonomously advance its physical time to reach the imposed ATHLET macro-step synchronization point, executing the core data exchange only at discrete intervals.

Consequently, the monolithic Fortran DO loops that traditionally governed the standalone time advancement had to be fragmented and exposed to Python as individual, step-by-step controllable routines. The `driver.py` script was therefore organized into two distinct operational blocks to accommodate both scenarios:

1. **Standalone Execution Block:** This block retains the legacy logic required to initialize the solver (`initialise`), run the continuous transient internally (`execute`), and safely close the output files and deallocate memory resources at the end of the calculation (`finalise`). These functions ensure full backward compatibility when the `isCoupledAthlet` flag is set to `False`.
2. **Explicit Coupling Block (ATHLET Hooks):** This section represents the core API development required to enable the explicit coupling architecture via text-file data exchange, bridging the Python supervisor with the Fortran backend. It is composed of three specific routines:
 - `advance_time_to_sync_point(sync_interval)`: Representing the core of the time integration logic, this function is invoked by the FEBE hook at the macro-step synchronization point. It commands FRENETIC to advance its physical time to cover the accumulated `sync_interval`. To achieve this, FRENETIC utilizes its internal adaptive sub-stepping logic, ensuring that mathematical stability is maintained within the sub-channel mesh while perfectly aligning with the ATHLET master clock.
 - `read_athlet_bc()`: This function updates the internal boundary conditions of the core solver. It triggers the Fortran backend to explicitly open the `athlet_to_frenetic.txt` file, reading the respective macroscopic temperature and pressure values evaluated by ATHLET, thereby preparing FRENETIC for its forthcoming time advancement step.
 - `write_frenetic_results()`: Functioning as the primary data extraction interface, this routine commands the FRENETIC core to evaluate the integrated absolute thermal power (W) generated across the 3D topology, alongside the bulk fluid outlet temperature. These values are immediately written into the `frenetic_to_athlet.txt` file, effectively bypassing direct array manipulation in RAM and establishing a resilient data transfer pathway.

6.4 Architecture of the API Thermal-Hydraulic Driver Interface of FRENETIC (TH_driver_API.f90)

To establish a functional and memory-safe bridge between the high-level Python supervisor and the core numerical solvers of FRENETIC, a dedicated Fortran API module (`th_driver_api`) was developed starting from the previous version (characterized only by the presence of the subroutines related only to standalone calculations). This module is subsequently compiled into a shared library via `f2py`, enabling seamless cross-language interoperability.

The fundamental refactoring of this Fortran driver was necessitated by the intrinsic operational differences between standalone simulations and explicit coupled execution. In its native configuration, the FRENETIC thermal-hydraulic solver handles the transient phase monolithically: once the `Coupling_Run_Transient()` subroutine is invoked, an internal DO loop assumes total control, advancing the physical time continuously from $t = 0$ to t_{end} . While highly efficient for isolated simulations, this monolithic approach is entirely incompatible with an explicit coupling scheme, where the ATHLET system code acts as the master clock and requires precisely controlled synchronization points.

To resolve this, the monolithic time integration loop was fragmented. The Fortran engine's capabilities were decoupled and exposed as individual, highly controllable subroutines, while the underlying time-marching strategy was entrusted to a newly developed macro-stepping routine within the `TH_Coupling_Manager_Mod_modified.f90` module. The resulting API architecture is divided into two distinct operational blocks:

- **Standalone Execution Block:** This section preserves the legacy capabilities of FRENETIC. It includes standard routines such as `TH_initialise`, `TH_initialise_from_files`, and `TH_finalise`. The `TH_execute` subroutine retains the traditional continuous execution logic, calculating the steady-state and running the uninterrupted transient phase.
- **Explicit Coupling Block:** This newly introduced section exposes the step-by-step numerical functions required by the Python `driver.py` script to synchronize with ATHLET. By delegating the raw file input/output operations to the internal `TH_IO_Manager_Mod_modified.f90` module, the API effectively acts as a clean, intermediate wrapper. The structure has been streamlined into three strategic subroutines:
 1. `TH_advance_time_to_sync_point(sync_interval)`: A routine that manages both the time advancement. It delegates the actual execution to the newly implemented `TH_advance_MacroStep` subroutine. This routine effectively overrides the native continuous execution by forcing the internal `transient` solver

to perform adaptive sub-stepping strictly up to the requested synchronization horizon, thus ensuring mass and energy conservation within the macro-step.

2. `TH_read_athlet_txt()`: A wrapper subroutine designed to interface with the Input/Output manager. Upon invocation from Python, it instructs the Fortran solver to securely open the text file generated by ATHLET, extracting the appropriate variables and mapping them directly to the internal boundary conditions arrays (e.g., `bc%inletPress(:)` and `bc%TINL(:)`).
3. `TH_write_frenetic_txt()`: A corresponding wrapper subroutine that triggers the evaluation of the total integral heat generation across the core channels. Rather than exposing multidimensional Fortran arrays directly to Python, which could pose severe memory management constraints, this routine performs the domain integration internally. The resulting scalar integral power and the calculated other variables are cleanly printed to an external text file, completing the cross-code interface exchange.

Chapter 7

Coupling architecture FRENETIC-ATHLET Test Cases & Results

The post-processing phase was carried out to verify that the coupling between the FRENETIC thermal-hydraulic module and the ATHLET system code was correctly implemented, with the ultimate goal of assessing the entire coupling architecture. As already discussed in Chapter 5 regarding FRENETIC, the results are exported using the Hierarchical Data Format version 5 (HDF5) [21].

To test the coupling architecture, it was decided to perform simulations, with a dedicated python script called `run_simulation.py`, focusing on the following test cases:

- `TH_1D_standalone_transient`
- `ALFRED`

The successful execution and reliable data extraction for all the examined cases confirmed that the coupling structure and its numerical implementation were handled correctly.

7.1 `TH_1D_standalone_transient`

This test case, named *SimpleLoop 1D*, was used for the preliminary analysis and verification of the explicit coupling interface between the ATHLET system-level code and the FRENETIC TH module. This model allows to isolate the data exchange dynamics in a controlled environment, ensuring numerical convergence before moving on to more complex coupled simulations.

The system models a one-dimensional thermal-hydraulic loop, specifically designed to reproduce the steady-state and transient behavior of Lead. The geometric configuration of the loop features a total height of 6 m and an overall width of 4 m. The hydraulic diameter of the channel, calculated as the ratio between the hexagonal section area and

its perimeter ($D_h = 4A/P$), is approximately 0.16 m. Furthermore, the discretization of the thermal-hydraulic domain was designed to accurately reflect the core physics:

- The core is located at an elevation of 1.2 m from the bottom reference.
- The heat removal is simulated through 5 control volumes discretized over a total length of 0.8 m, positioned at a height of 4 m.

The loop topology is defined by specific boundary components characterizing the two computational domains (see Fig. 7.1).

Regarding the ATHLET domain:

1. **ATHLET INLET:** Defined as the fluid entry point into the macroscopic loop, implemented via a **FILL** component that strictly imposes the inlet mass flow rate ($\dot{m}_{in}$) and the inlet fluid temperature (T_{in}).
2. **ATHLET OUTLET:** Implemented via a *Time Dependent Volume* (TDV), which acts as a pressure boundary condition imposing the system outlet pressure (P_{out}), thus ensuring the hydraulic stability of the overall circuit.

Regarding the FRETIC domain:

1. **FRETIC INLET:** Represents the bottom of the active channel assembly. Here, the inlet mass flow rate and the inlet coolant temperature (provided by the ATHLET loop calculations) are imposed to evaluate the upward flow.
2. **FRETIC OUTLET:** Represents the top of the channel assembly, where the outlet pressure boundary condition (provided by ATHLET) is applied to correctly compute the internal pressure drops.

Imposed Parameter	Value
Core Mass Flow Rate	173.3 kg/s (Constant)
Thermal Power Generated	8.0 kW
Core Outlet pressure	6.3 bar
Working Fluid	Lead (Pb)

Table 7.2: *Boundary Conditions and setup parameters in the FRENETIC domain.*

The coupling between the codes occurs through an explicit exchange of variables that relies on a *Partitioned Domain Decomposition* approach, where the computational fluid domain is split at the core boundaries (inlet and outlet plenums). Specifically, ATHLET provides the inlet temperature (extracted from the TDV) and outlet pressure conditions (extracted from the PIPE) to FRENETIC. In return, FRENETIC calculates and transfers back the outlet temperature, the inlet pressure, and the integral thermal power (distributed over the extraction volumes), thus effectively closing the thermal-hydraulic feedback loop (see Tab. 7.3).

Exchanged Quantity	Location in ATHLET	Location in FRENETIC	Exchange Direction
Core outlet pressure	Inlet Pressure (PIPE)	Outlet Pressure	AT → FR
Core inlet pressure	Outlet Pressure (TDV)	Inlet Pressure	FR → AT
Core inlet temperature	Outlet Temperature (TDV)	Inlet Temperature	AT → FR
Core outlet temperature	Inlet Temperature (PIPE)	Outlet Temperature	FR → AT
Core integral power	Removed Heat (GCSM)	Total Core Heat	FR → AT

Table 7.3: *Variables dynamically exchanged between the two codes at synchronization intervals.*

Figure 7.2 illustrates the time evolution of the main thermal-hydraulic variables exchanged between ATHLET and FRENETIC during the coupled simulation, characterized by a total simulation time (dictated by ATHLET) of 300 s.

Evolution of Coupled Variables

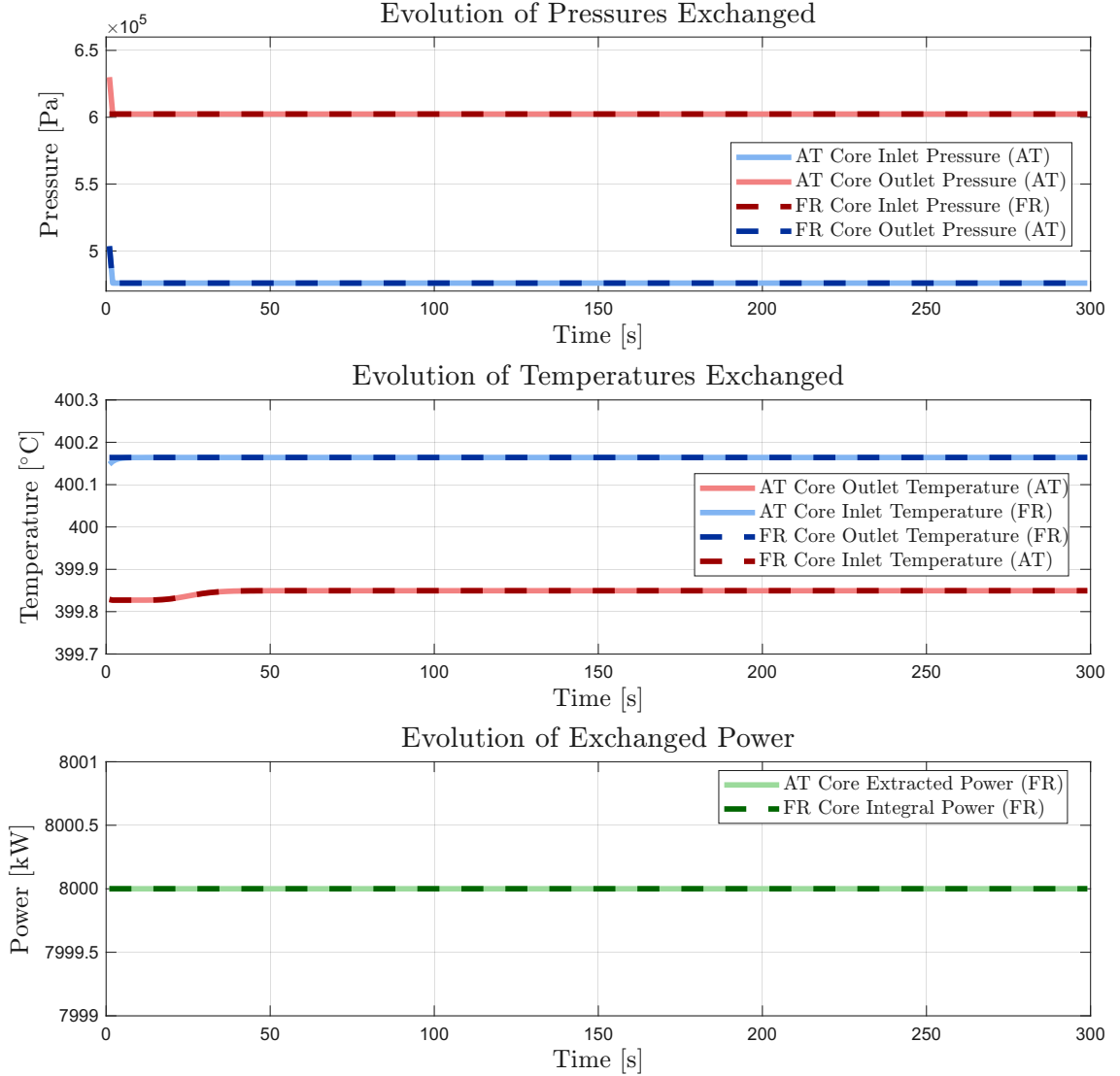


Figure 7.2: Time evolution of the exchanged variables during the coupled transient (the acronym at the beginning indicates the domain to which the variable belongs, while the acronym at the end indicates the domain from which it comes).

The analysis of the three subplots allows us to draw the following conclusions:

- **Pressures:** A perfect overlap is observed between the ATHLET inlet pressure ($P_{in,ATH}$) and the FRENETIC outlet pressure ($P_{out,FRE}$), which is consistent with fluid continuity requirements. The same holds true for the ATHLET outlet pressure ($P_{out,ATH}$) and the FRENETIC inlet pressure ($P_{in,FRE}$). The stability of the signals highlights that the pressure drop calculation within the loop is numerically

consistent between the two solvers.

- **Temperatures:** The graph clearly shows thermodynamic consistency across the domain interfaces. The ATHLET core outlet temperature ($T_{out,ATH}$), which acts as a boundary condition, perfectly overlaps with the FRENETIC core inlet temperature ($T_{in,FRE}$). Conversely, the FRENETIC core outlet temperature ($T_{out,FRE}$) is successfully fed back into ATHLET, completely overlapping with the corresponding ATHLET core inlet temperature ($T_{in,ATH}$). The exact matching of these curve pairs confirms that the data exchange takes place without any significant time lags, thus confirming that the synchronization strategy based on the physical time of the simulation is appropriate.
- **Integrated Power:** The power trend directly reflects the energy deposited into the fluid. In this 1D *standalone* test case, the power remains constant at 8 kW, confirming that the ATHLET GCSM controller correctly reads and processes the power signal from the exchange files generated by FRENETIC.

The observed behavior ultimately confirms the robustness of the explicit coupling approach. Even though the interface relies on a text file exchange mechanism, the complete absence of spurious oscillations in the exchanged variables proves that the chosen synchronization time step ($\Delta t_{sync} = 1.0$ s) is adequately sized relative to the time constants of the investigated transient.

7.2 ALFRED Reactor

This ultimate test case was utilized to perform a more comprehensive and structured analysis of the coupling interface between ATHLET and the FRENETIC thermal-hydraulic module. Initially, the objective was to adopt the exact same domain discretization strategy successfully applied in the 1D case. However, since the ALFRED core features a multi-assembly configuration, a direct replication of the 1D approach was not feasible. Specifically, the parallel-channel layout inherently entails localized flow dynamics and different pressure evaluations at the inlet of each individual channel, which cannot be consistently represented by a single, lumped boundary condition. Despite this necessary methodological adaptation, the current setup allows for a robust comparison with the results obtained in previous works [1], highlighting the enhanced simulation capabilities provided by the integration of a system-level code like ATHLET.

7.2.1 ATHLET Model

The system models a one-dimensional thermal-hydraulic loop based on the geometry detailed in [27] before the design revision done in [1], and is specifically designed to reproduce the steady-state and transient behaviour of the liquid lead primary coolant. The geomet-

ric configuration of the loop features approximately a total height of 10 m and an inner diameter value of 8.3 m.

The loop topology (see Fig. 7.3) follows a forced circulation regime defined by the following main components and flow path:

1. The liquid lead flows from the reactor pool bottom (RPB), where it is collected after being discharged from the Steam Generators, and reaches the core lower vessel (CLV) through a lower pool connection (LPC).
2. From the CLV, the coolant enters the active core (RC). The core channel is modelled considering a first unpinned region of 1.0 m (characterized by a larger flow area), followed by a 1.0 m region where the fuel pins are located and the heat transfer to the liquid lead occurs through an active length of 0.6 m. This core region is modelled considering 469 channels in parallel.
3. Upon exiting the core, the flow splits: a portion of the lead enters the core upper vessel (CUV) where it stagnates, while the remaining part is drawn into the reactor coolant pump (RCP). The 8 pumps are lumped into a single equivalent channel with a hydraulic diameter of 0.5 m. The pump head is regulated by a Proportional-Integral-Derivative (PID) controller, which continuously adjusts the operational parameters to maintain the target total mass flow rate of 25974.45 kg/s.
4. After passing through the pump component, the liquid lead goes into a plenum (consisting of the branch PL and the pipe PL.P) and then flows downwards into the Steam Generators (SG). The heat extraction within the SG consists of a uniform heat sink along the component, which is automatically adjusted by the simulation in order to maintain a target outlet temperature (T_{out}) of approximately 400°C.
5. Exiting the Steam Generators, the cooled liquid lead is discharged into the reactor pool upper (RPU), flowing downwards to the RPB to complete the primary circuit.

Additionally, the model incorporates upper piping sections (CUV.C, PL.C, RPUC.C) that represent the Argon cover gas, and are connected by specific junctions (RPUC, PLC). Thermal structures (labeled with a .W suffix, e.g., CUV.W, SG.W) are also included to account for the thermal inertia and the boundary heat losses of the structural materials (e.g., Austenitic stainless steel). Finally, it should be noted that the current 1D system code approach intrinsically limits the representation of complex 3D recirculation and fluid mixing phenomena; however, this simplified macroscopic assumption is considered acceptable pending further detailed CFD studies.

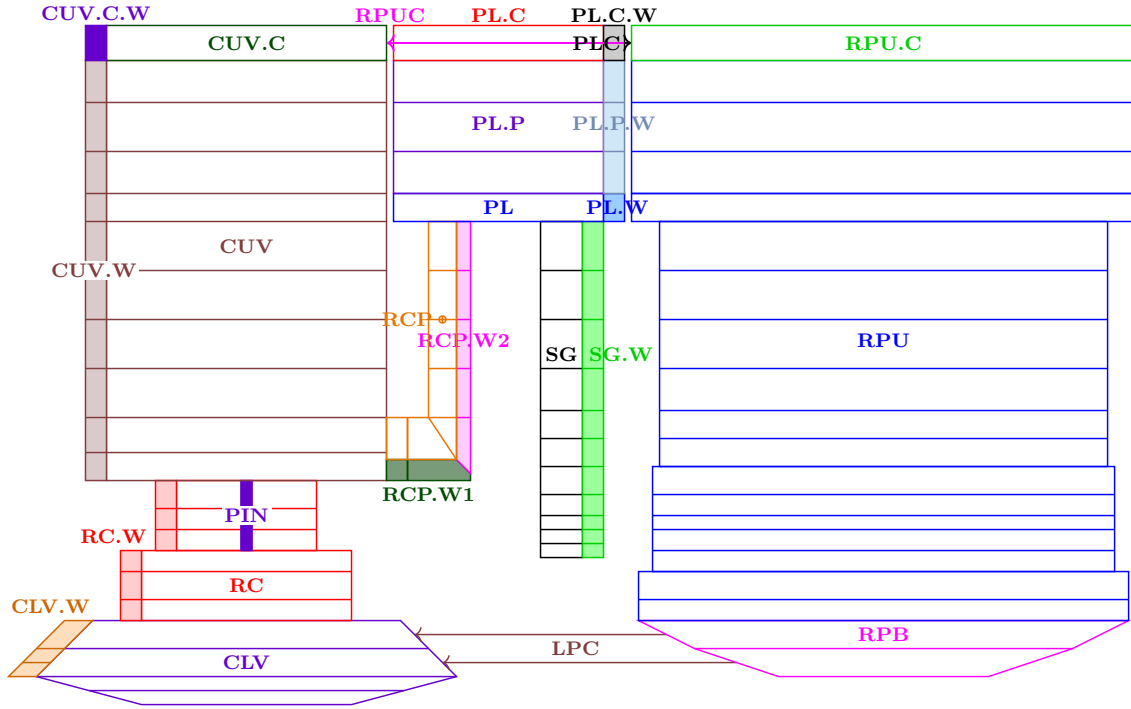


Figure 7.3: *ATHLET* detailed nodalization scheme for the *ALFRED* test case (reference in Chapter 1 in Fig. 1.1).

Under nominal full-power conditions, the primary thermal-hydraulic circuit is governed by forced circulation induced by the primary pumps submerged in the lead pool. The operational parameters obtained at the steady state equilibrium (time for steady state simulation: 100000 s) are reported in Table 7.4 and are consistent with the reference [2]:

Variable Name	Value
Extracted Thermal Power	300 MW _{th}
Core Inlet Temperature	401.31 °C
Core Outlet Temperature	480.45 °C
Core Inlet Pressure	6.58 bar
Core Outlet Pressure	4.79 bar
Mass Flow Rate	25974.45 kg/s
Working Fluid	Lead (Pb)

Table 7.4: *ATHLET* Steady-state operational parameters.

Furthermore, to evaluate the numerical stability of the *ATHLET*'s model, a nodalization convergence study was conducted, proving the appropriateness of the 64-volume mesh utilized for the coupled simulations (see Appendix A).

7.2.2 FRENETIC Model

Regarding the general characteristics of the test case, the most relevant geometric and numerical parameters [1] are:

- *Active length*: This is defined by the variables `zPowStart` and `zPowEnd`, which represent the axial positions of the beginning and the end of the heated zone, respectively. For this specific ALFRED test case, this value is set to 0.6 m.
- *Channel Number*: This is defined by the variable `nChan` and represents the total number of channels considered in the core configuration, including the dummy elements. For the ALFRED case, this value is equal to 469.
- *Total Length of the FA*: This is defined by the variable `xLengt` and accounts for the entire axial extension of the Fuel Assembly, including the non-heated structural zones beyond the active length. For the ALFRED case, this value is equal to 3.4 m, but to be consistent with the core dimensions of ALFRED it was set to 2.0 m.
- *Hexagonal Pitch*: This value is defined by the variable `HexPitch` and represents the center-to-center distance between adjacent Fuel Assemblies (modeled as individual macro-channels) in the reactor core. For the ALFRED case, this value is equal to 0.171 m.

Furthermore, to properly perform the simulation, it was necessary to account for the specific average mass flow rate assigned to each channel group across the core configuration (reported in Tab. 7.5) (see Fig. 7.4). The sum of these individual mass flow rates equals the total mass flow rate of 25974.45 kg/s:

Cooling Group	Average Mass Flow Rate per Simulated Channel [kg/s]
 Fuel SA - I	172.3
 Fuel SA - II	145.2
 Fuel SA - III	117.5
 Fuel SA - IV	93.4
 Control Assemblies (CR)	261.0
 Reflector	143.0
Bypass between fuel SA	110.0

Table 7.5: Average mass flow rate per channel for the ALFRED core configuration [1].

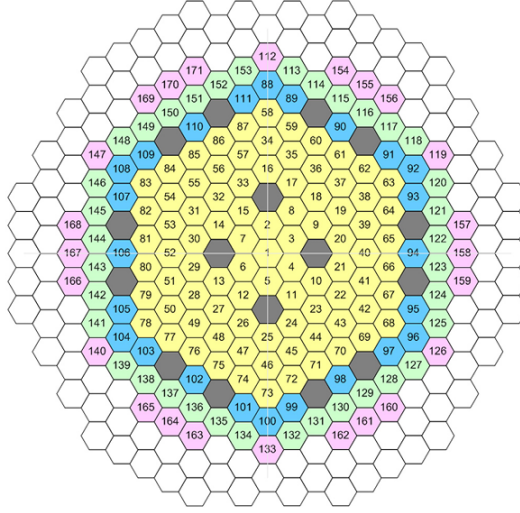


Figure 7.4: *ALFRED core configuration per cooling groups* [1].

This total mass flow rate value differs from the one reported in [1] because the number of simulated channels for the Fuel SA-I Cooling group is 91 instead of 87. Consequently, the resulting overall mass flow rate slightly differs from the nominal value of 25460 kg/s, although it is consistent with other references that state the mass flow rate could have a value around 25900 kg/s [28].

Subsequently, to provide a realistic radial power distribution within the core, a `power.inp` file was generated. The power generated in each channel was calculated proportionally to its respective mass flow rate, simulating the core orificing and ensuring a uniform temperature increase. Then the value was divided by the active length:

$$P_{channel} = \frac{P_{tot} \cdot \frac{\dot{m}_{channel}}{\dot{m}_{tot}}}{0.6} \quad (7.1)$$

Furthermore, since this is a multi-channel case, FRENETIC must perform a fully converged steady-state calculation at time $t = 0$ prior to the coupling with ATHLET to provide the necessary initial values for the transient execution. It is worth noting why this dedicated initialization is strictly necessary here, whereas it was not required in the previous 1D test cases. In a 1D single-channel geometry, the hydrodynamics are straightforward: the entire inlet mass flow rate is forced through the single assembly, allowing the solver to compute the initial thermodynamic state without ambiguity. Conversely, in a complex multi-channel core like ALFRED, the total mass flow must be properly partitioned across hundreds of parallel assemblies based on their specific hydraulic resistances to achieve an equalized core pressure drop. Without resolving this initial flow distribution, the feedback variables passed to ATHLET at $t = 0$ would be physically meaningless.

To ensure this proper initialization, a new subroutine named `TH_init_steady_state` was implemented in `TH_driver_API.f90`. This subroutine calls the `SteadyState` and `MemSwap`

routines (from FRENETIC) to compute the initial values, and subsequently initializes the exchange variables for the transient using the `InitiaTran` subroutine. This new function was then invoked within the FRENETIC Python modules `driver.py` and `__init__.py`, and successfully integrated into the coupling supervisor (`ATHLET_coupling`) right before the activation of the ATHLET engine.

Table 7.6 reports all the steady-state operational parameters used to initialize the FRENETIC domain:

Variable Name	Value
Extracted Thermal Power	300 MW _{th}
Inlet Temperature	673.15 K
Outlet Pressure	2.0 bar
Total Mass Flow Rate	25974.45 kg/s
Working Fluid	Lead (Pb)

Table 7.6: *FRENETIC Steady-state operational parameters.*

7.2.3 Coupling

The coupling between the codes occurs in the following way: ATHLET provides the inlet temperature and outlet pressure conditions to FRENETIC. In return, FRENETIC calculates and transfers back only the integral thermal power, closing the thermal-hydraulic feedback loop.

Although FRENETIC, in this specific case, does not provide ATHLET with any boundary conditions other than the integral thermal power, observing an average value, across all 469 channels, for the outlet temperature variable remains of interest.

$$T_{out,FRENETIC} = \frac{H_{out,total}}{\dot{m}c_{p,total}} - 273.15 \quad (7.2)$$

Regarding ATHLET, since a more complex case is being analyzed as opposed to a single-channel case, the data extraction procedure changes. As specified in the previous section, the ATHLET outlet temperature was extracted from the TDV (Time Dependent Volume), while the inlet pressure was extracted from the PIPE. For ALFRED, the extraction of these data occurs through a single component defined as RC, which represents the ALFRED's core in the ATHLET domain. Another difference is that, for ALFRED, it was also possible to observe the total generated power value in ATHLET (retrieved from FRENETIC) by using the command `engine_Athlet.gcsn['S.POW']['value'].value`, subsequently multiplied by 22050, that represents the total number of pins; and also the value of the mass flow rate using the command `engine_Athlet.TFO_map['RC'].glj[0]`. Finally, for the execution of the transient, a steady-state initialization file was also created on the

ATHLET side, which was called within the `run_simulation.py` script using the `options` command. Tab. 7.7 reports all the variables observed during the coupled simulation:

Exchanged Quantity	Location in ATHLET	Location in FRENETIC	Exchange Direction
Core outlet pressure	Outlet Pressure (RC)	Outlet Pressure	AT → FR
Core inlet temperature	Inlet Temperature (RC)	Inlet Temperature	AT → FR
Core outlet temperature	Outlet Temperature	Outlet Temperature	FR (Not exchanged)
Core integral power	Heat Source (S.POW)	Total Core Power	FR → AT
Core mass flow rate	Core Mass Flow (RC)	Core Inlet Mass Flow	Not exchanged
Coolant specific heat	Coolant Specific Heat	Coolant Specific Heat	FR (Not exchanged)

Table 7.7: Variables dynamically exchanged and monitored between the two codes at synchronization intervals.

Figures 7.5 and 7.6 illustrate the time evolution of the main thermal-hydraulic variables exchanged between ATHLET and FRENETIC during the coupled simulation of 300 s.

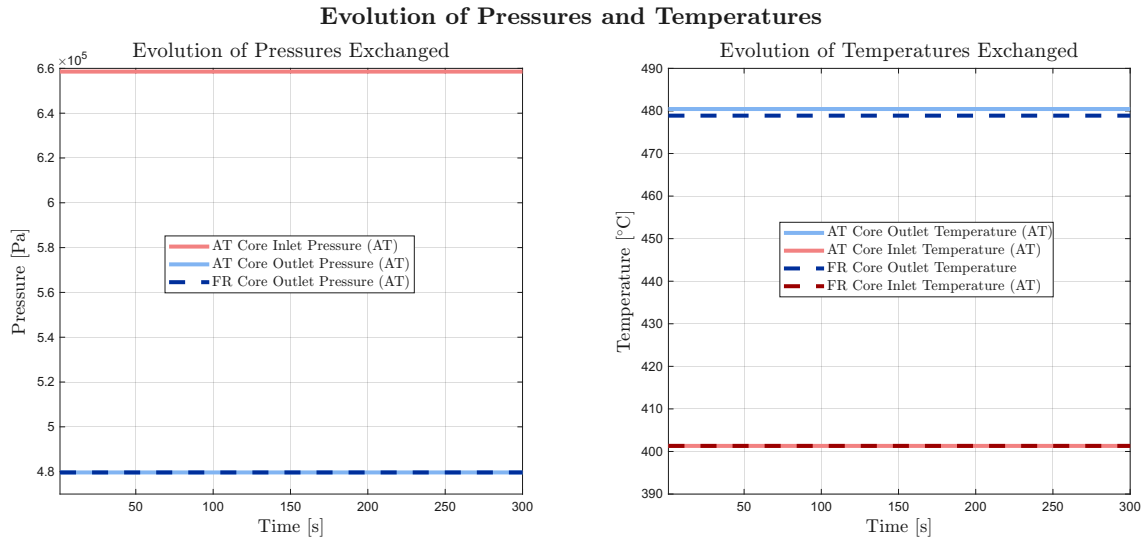


Figure 7.5: Time evolution of the exchanged variables during the coupled transient for the ALFRED case (the acronym at the beginning indicates the destination domain, while the acronym at the end indicates the origin domain).

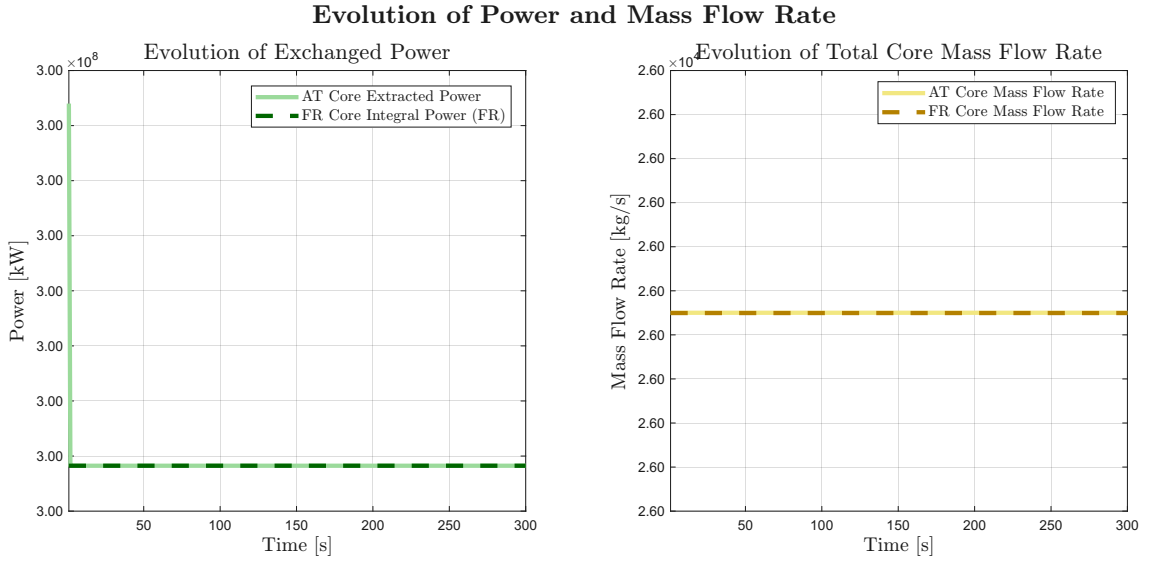


Figure 7.6: Time evolution of the exchanged variables during the coupled transient for the ALFRED case (the acronym at the beginning indicates the destination domain, while the acronym at the end indicates the origin domain).

The analysis of the coupled simulation results, evaluated over progressively wider time windows (10 s, 60 s, 100 s, and 300 s), allows to draw the following conclusions regarding the steady-state stability of the ALFRED model:

- **Pressures:** The exchanged pressure signals exhibit absolute stability throughout all the investigated time frames. The core inlet pressure provided by ATHLET ($P_{in,ATH}$, approximately 6.58 bar) and the core outlet pressure calculated by FRENETIC ($P_{out,FRE}$, approximately 4.80 bar) maintain a constant pressure drop. The complete absence of numerical drifts confirms that the hydraulic coupling and the momentum balance at the interface are perfectly handled by the two solvers.

As observed in the pressure evolution results, the core inlet pressure evaluated by FRENETIC is deliberately omitted from the analyzed variables. An in-depth analysis of the multi-channel core mechanics revealed a fundamental architectural incompatibility in the current coupling scheme regarding pressure distribution.

Unlike the previously analyzed 1D single-channel case, where a single pressure drop is straightforwardly mapped between the two solvers, the ALFRED multi-FA model features diverse flow regimes. FRENETIC independently calculates the pressure drop across each individual channel (or cooling sub-group), which naturally varies depending on the specific mass flow rate allocated to that group. Consequently, FRENETIC yields a heterogeneous spatial distribution of inlet pressures. Physically, this distribution represents the localized conditions immediately downstream of the lower plenum and the specific orificing devices at the entrance of the Fuel Assemblies. However, this distributed behavior is strictly incompatible with the current macroscopic ATHLET nodalization, which expects to receive a single, homogenized inlet pressure value from the core domain to assign as the global lower

plenum pressure.

Despite this structural discrepancy, the uncoupled FRENETIC inlet pressure does not compromise the validity of the current coupled simulation. In the present modeling scheme, the ATHLET loop still incorporates the macroscopic Reactor Core (RC) component, meaning that the global core pressure drops and the main loop thermal-hydraulics are independently governed and stabilized by the system code.

To resolve this mapping inconsistency and achieve a physically rigorous multi-channel coupling, two revised strategies are envisioned for future developments:

1. **Boundary Condition Swap:** The data exchange architecture could be inverted. ATHLET would impose the inlet and outlet core mass flow rates and provide the global upper and lower plenum pressures as boundary conditions to FRENETIC. In turn, FRENETIC, utilizing internally adjustable friction factors to manage the flow partitioning, would compute the detailed channel hydraulics without needing to collapse the inlet pressure into a single scalar value for ATHLET, and then would have to change the inlet and outlet mass flow rates back to ATHLET.
 2. **ATHLET Topology Refinement:** Alternatively, the nodalization of the system code could be enhanced by assigning multiple Time Dependent Volumes (TDVs) within ATHLET, corresponding exactly to the number of FA sub-groups defined in FRENETIC, immediately downstream of the lower plenum. Using this approach, the same coupling methodology outlined for the 1D standalone case can be applied.
- **Temperatures:** The graphs clearly demonstrate the thermodynamic consistency of the coupling interface. The ATHLET core inlet temperature ($T_{out,ATH}$), which acts as a boundary condition for FRENETIC ($T_{in,FRE}$), perfectly overlaps at the nominal ALFRED cold value of 400°C. Similarly, the FRENETIC core outlet temperature ($T_{out,FRE}$), shows an exact overlap at 480°C. This behaviour confirms that the data exchange takes place without any significant time lags or numerical diffusion, validating the synchronization strategy.
 - **Integrated Power:** The power trend directly reflects the thermal energy deposited into the fluid. Aside from a negligible initialization adjustment at the very first computational step, the power remains perfectly constant at the nominal value of 300 MW. This confirms that the ATHLET GCSM controller correctly reads and processes the power signal from the text exchange files generated by FRENETIC.
 - **Mass Flow Rate:** The primary coolant mass flow rate exchanged between the codes shows a virtually perfect match, stabilizing at 25974.45 kg/s. The infinitesimal discrepancy between the signals (in the order of 10^{-4} kg/s) is strictly related to the text file formatting and machine precision limits during the I/O operations, proving that the mass conservation is rigorously respected across the coupling boundary.

The observed behaviour ultimately confirms the robustness of the explicit coupling approach applied to the full-loop ALFRED model. Even though the interface relies on a text file exchange mechanism, the complete absence of spurious oscillations in the exchanged variables over 300s of simulated time proves that the chosen synchronization time step is adequately sized relative to the time constants of the system.

7.2.4 FRENETIC Grid Independence Analysis

To ensure the numerical reliability of the coupled ATHLET-FRENETIC simulations, a spatial grid independence study on the axial discretization of the core channels was performed. Four different axial meshes were tested, characterized by an increasing number of elements ($nElms$): a coarse mesh (100 nodes), a nominal mesh (200 nodes), a fine mesh (400 nodes), and an ultra-fine mesh (600 nodes).

Throughout all simulations, the **total integrated power**, taken from FRENETIC, was successfully kept constant at 300 MW, confirming the consistency of the coupling boundary conditions across the different spatial grids.

The evaluation of the spatial convergence was based on two primary safety parameters: the **maximum coolant temperature** ($T_{cool,max}$) and the **maximum cladding surface temperature** ($T_{clad,max}$). As shown in the following graphs (Fig. 7.7 and 7.8), the transition from the coarse to the nominal mesh yielded negligible differences. However, a noticeable refinement in the resolution of the thermal gradients occurred when increasing the grid to 400 elements. Specifically, $T_{cool,max}$ decreased by approximately 0.31°C , while $T_{clad,max}$ increased by about 0.81°C . This behaviour indicates that a lower number of nodes is insufficient to fully capture the peak local heat transfer dynamics within the core. Further refining the mesh to 600 axial elements demonstrated a clear asymptotic stabilization of the numerical solution. The differences between the 400-node and 600-node meshes are minimal: the coolant temperature variation was limited to just 0.089°C , and the cladding temperature delta was minimized to 0.051°C .

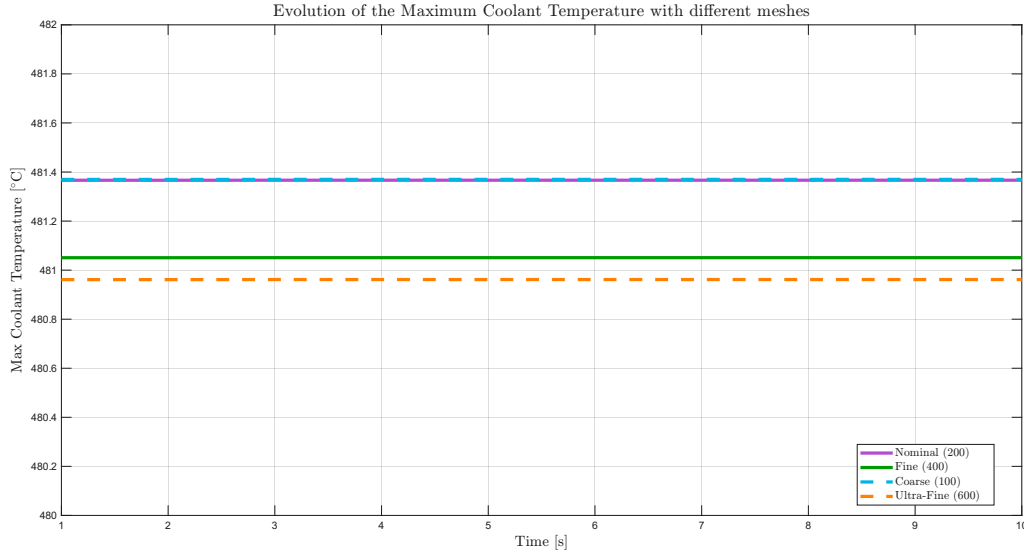


Figure 7.7: *Time Evolution (10 s) of the Maximum Coolant Temperature with different meshes.*

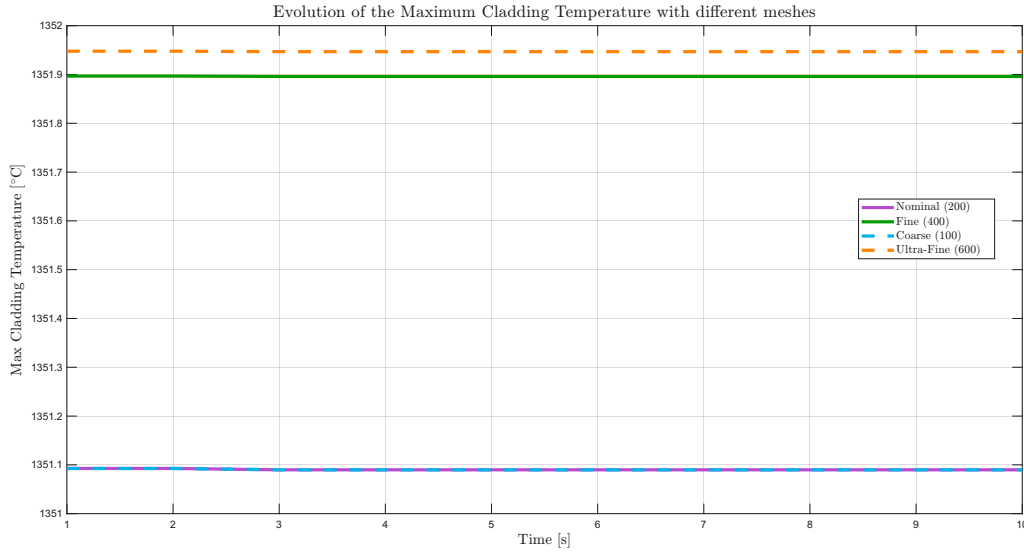


Figure 7.8: *Time Evolution (10 s) of the Maximum Cladding Temperature with different meshes.*

To rigorously quantify the spatial convergence observed in the temperature profiles, a relative error analysis was performed. In the absence of an exact analytical solution, the results obtained with the ultra-fine mesh (600 axial elements) were assumed as the reference asymptotic solution (T_{ref}). The relative error ϵ_{rel} for each evaluated grid was computed using the following standard formulation:

$$\epsilon_{rel} = \left| \frac{T_i - T_{ref}}{T_{ref}} \right| \times 100 \quad (7.3)$$

where T_i represents the peak temperature recorded for the i -th mesh configuration.

As summarized in Table 7.8, the discrepancy between the spatial discretizations rapidly decreases as the grid is refined. The relative error for both the coolant and cladding temperatures drops well below 0.05% when comparing the 400-element mesh to the reference configuration. These extremely low percentage values confirm that the numerical solution is firmly stabilized and that the spatial grid independence has been successfully achieved.

Mesh	$T_{\text{cool,max}}$ [°C]	Rel. Error [%]	$T_{\text{clad,max}}$ [°C]	Rel. Error [%]
Coarse	481.3700	0.0849	1351.0898	0.0634
Nominal	481.3666	0.0842	1351.0898	0.0634
Fine	481.0508	0.0185	1351.8960	0.0038
Ultra-Fine	480.9616	<i>Reference</i>	1351.9468	<i>Reference</i>

Table 7.8: *Relative error analysis for the spatial grid independence study.*

The temperatures variations reported in Fig. 7.9 and 7.10, are physically negligible and do not represent any macroscopic thermodynamic or heat transfer phenomenon occurring within the core. Instead, this behaviour is a direct artifact of the numerical coupling methodology. The explicit nature of the coupling scheme, combined with the data exchange performed via the text files, introduces intrinsic truncation errors and floating-point round-offs at the domain interface. Consequently, the two independent solvers continuously adjust their internal thermodynamic balances at each synchronization step, oscillating around the exact steady-state analytical solution within the limits of their respective convergence tolerances.

The fact that these micro-oscillations do not diverge over time, but rather remain strictly confined within an ultra-fine numerical band, provides further confirmation of the unconditional stability of the synchronization algorithm and the robustness of the achieved steady-state.

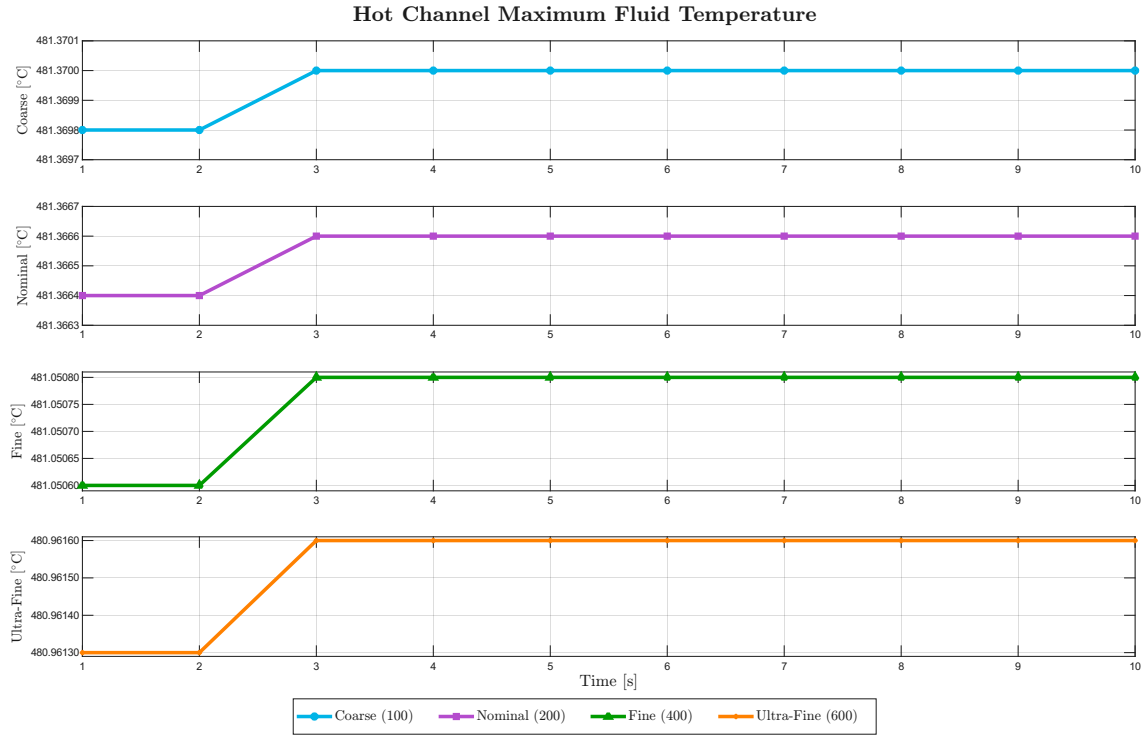


Figure 7.9: Time Evolution (10 s) of the Maximum Coolant Temperature taken from FRENETIC using different meshes.

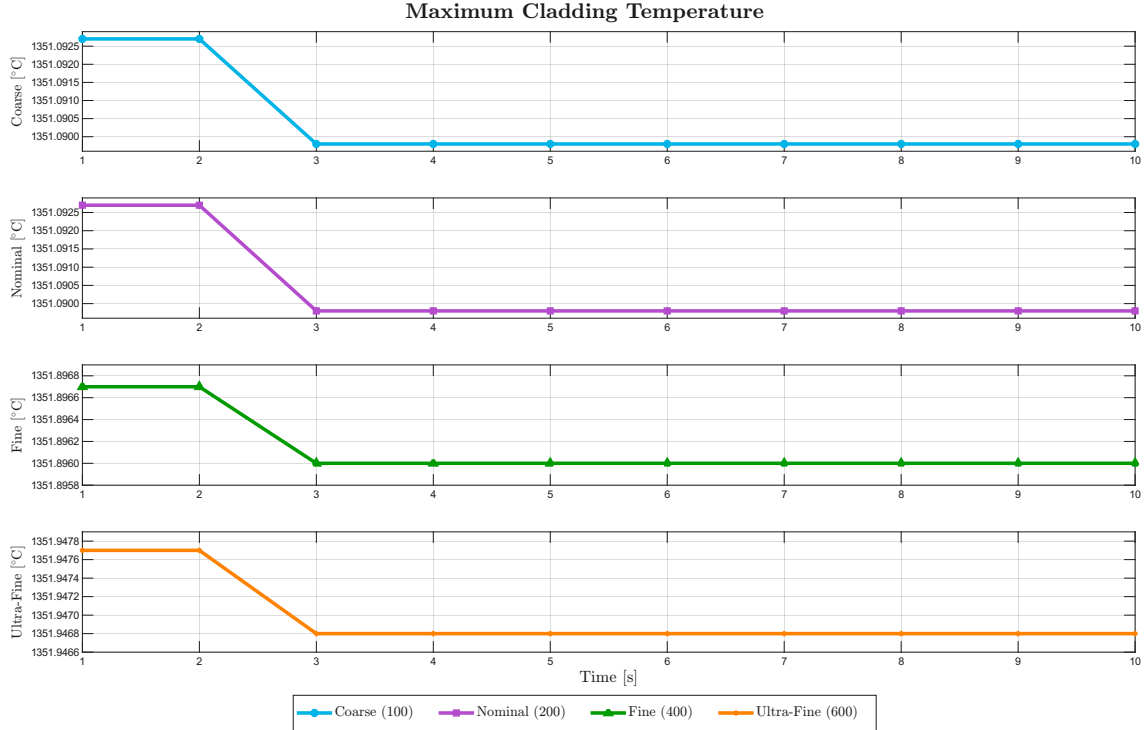


Figure 7.10: Time Evolution (10 s) of the Maximum Cladding Temperature taken from FRENETIC using different meshes.

In conclusion, in the context of liquid metal thermal-hydraulics, temperature variations

below 1°C are considered negligible numerical noise; therefore, the grid independence is considered successfully achieved. Although the 600-element mesh provides the most rigorous spatial discretization, the 200-element configuration demonstrated highly consistent and physically reasonable results. Given the necessity to simulate an extended transient of 300 s, adopting the 600-element mesh would have resulted in an excessively high and prohibitive computational burden. Consequently, to ensure an optimal trade-off between numerical accuracy and computational efficiency, the 200-element mesh was selected as the reference discretization for all the subsequent transient analyses.

Chapter 8

Conclusions and Future Developments

The deployment and safety assessment of Generation IV Liquid Metal Fast Reactors (LMFRs) rely fundamentally on the availability of robust, multi-scale computational tools. Fission reactors are governed by deeply interconnected multi-physics phenomena, requiring simulation frameworks that can seamlessly bridge the gap between macroscopic plant dynamics and localized core thermal-hydraulics. In this context, the primary objective of this thesis was the extensive modernization of the thermal-hydraulic (TH) module of the FRENETIC code and the development of a novel multi-scale coupling interface with the ATHLET system code.

The first major phase of this work was dedicated to the structural refactoring of the FRENETIC TH solver. Moving away from the legacy monolithic Fortran structure, the code was transitioned into a highly modular architecture. This restructuring strictly adhered to modern software engineering principles, improving memory management, flexibility, and portability. To ensure that this deep code refactoring did not compromise physical accuracy, the *new* TH module was subjected to a rigorous validation matrix. The successful execution of 1D and 3D standalone transients, alongside the experimental benchmark against the CIRCE pool facility (ICE campaign), definitively proved that the solver is numerically stable and capable of accurately capturing the thermal inertia and the specific heat transfer mechanics of heavy liquid metals.

Following the successful standalone validation, the project focused on the core objective: the coupling strategy. Rather than forcing a rigid, hard-coded merge between FRENETIC and ATHLET, a flexible, *Event-Driven* architecture was developed. By exploiting the Py-ATHLET shared library and specific Fortran callback functions (Hooks), a Python Master Driver was implemented to orchestrate the execution. For this work, a fully explicit coupling scheme was adopted, relying on a secure text-file data exchange at predefined macroscopic synchronization intervals. This methodology allowed the two solvers to maintain their optimized internal time-advancement schemes without the computational overhead

of interface iterations.

The testing of this architecture on the ALFRED simplified loop demonstrated that the coupling is physically consistent. The codes successfully exchanged macroscopic boundary conditions (inlet temperature, outlet pressure) and core feedback parameters (integral extracted power), proving that the macroscopic BOP dynamics simulated by ATHLET can effectively drive the high-fidelity sub-assembly resolution provided by FRENETIC.

While the primary objectives of this thesis have been successfully met, laying a solid foundation for multi-scale LMFR simulations, the developed architecture opens the door to several necessary enhancements. The current limitations encountered during the testing phase highlight a clear roadmap for future developments:

- **Fully Implicit Coupling Scheme:** The explicit approach proved stable for the transients analysed in this work. Nonetheless, to tackle severe accidental scenarios characterized by violent feedbacks and to allow for larger, more efficient time-steps, the coupling architecture should evolve towards a fully implicit, iterative scheme. This will require the implementation of relaxation algorithms within the Python supervisor to ensure numerical convergence at the interface.
- **In-Memory Data Exchange:** The text-file-based coupling methodology adopted in this work was instrumental in ensuring data tracking, avoiding segmentation faults, and simplifying the debugging process during the initial implementation. However, writing and reading `.txt` files at every macro-step introduces an inevitable I/O computational overhead. The natural next step is to upgrade the Python Master Driver to perform a direct *In-Memory* data injection using shared arrays, fully exploiting the capabilities of the `libathlet.so` interface.
- **Multi-Channel Boundary Condition Mapping:** During the transition from the 1D single-channel test to the multi-channel ALFRED core representation, challenges emerged regarding the proper distribution of ATHLET's 1D boundary conditions across FRENETIC's 3D domain. Specifically, mapping a single system outlet pressure to multiple core channels with varying flow regimes requires a more sophisticated mathematical interface. Future work should implement a dedicated mapping matrix to properly distribute mass flow rates and pressure drops dynamically across the different core cooling groups (Fuel, Control Rods, Reflector).
- **Integration of the Neutronic Module:** The ultimate goal of the FRENETIC code is full multi-physics simulation. With the TH-System coupling now established, future efforts will focus on re-integrating the FRENETIC Neutronic (NE) module into the coupled loop. This will enable the simulation of fully coupled Neutronic/Thermal-Hydraulic/System transients, allowing for the comprehensive study of spatial kinetic effects and passive reactivity feedbacks driven by the primary circuit dynamics.

Ultimately, the work conducted in this thesis successfully delivered a modernized, modular, and coupled simulation framework. Despite the inherent complexities of synchronizing independent numerical solvers, the updated FRENETIC architecture now possesses the structural maturity required to support the next generation of advanced nuclear safety analyses.

Appendix A

ATHLET System Nodalization Study

To properly evaluate the spatial convergence and the numerical stability of the primary circuit, a dedicated nodalization study was performed on the ATHLET standalone model. The analysis compared four different spatial grids, characterized by a progressively increasing total number of fluid volumes: 64, 128, 173, and 210.

The study assessed the absolute values and the relative discrepancies of the steady-state temperature and pressure fields at key circuit locations: Core Inlet, Core Outlet, Steam Generator (SG) Inlet, and Steam Generator Outlet. The coarsest mesh (64 volumes) was assumed as the reference baseline for the discrepancy calculations.

Regarding the **temperature field** (Fig. A.1), the results demonstrate a generally stable behaviour. The relative temperature discrepancy increases with the grid refinement but remains globally bounded below 2.5% for all evaluated nodes. Notably, the SG Inlet temperature exhibits almost zero discrepancy across all meshes, highlighting the strong robustness of the energy balance in that specific section of the loop (the mixing of the Pb is so large that it is guaranteed that the same T is always reached at the SG inlet).

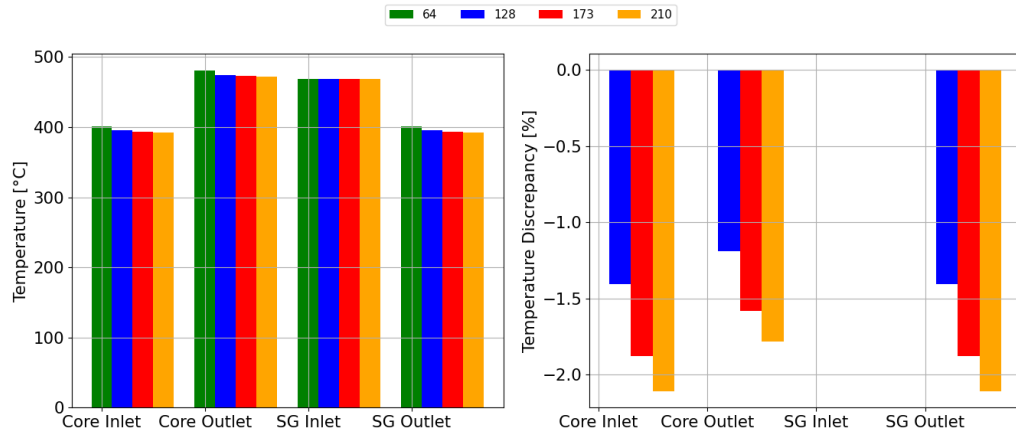


Figure A.1: *Nodalization Study on Temperature.*

Conversely, the **pressure field** (Fig. A.2) analysis reveals a numerical anomaly. While the 128-volume and 210-volume grids show a consistent and acceptable pressure discrepancy (ranging between 1% and 4%), the 173-volume configuration exhibits a severe deviation, peaking at a negative discrepancy of approximately 11% at the SG Inlet. This localized instability strongly suggests that the 173-volume nodalization introduces an ill-posed geometric transition which numerically perturbs the momentum balance.

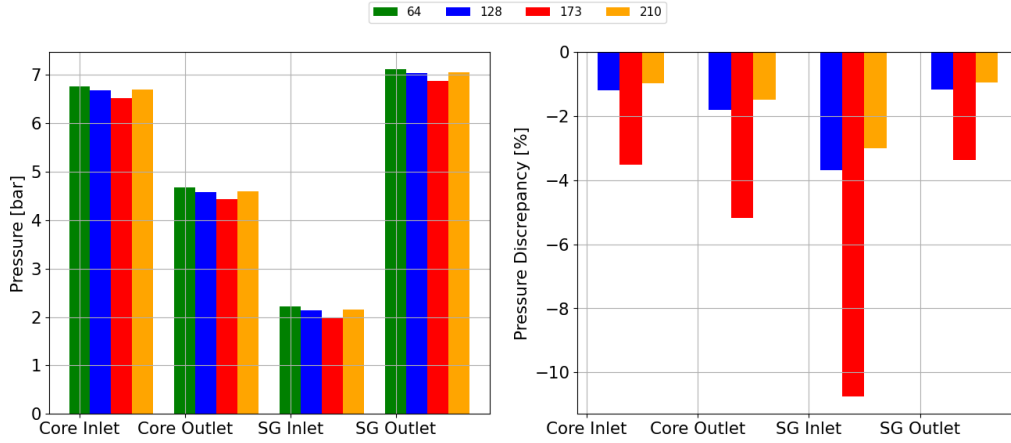


Figure A.2: *Nodalization Study on Pressure.*

Finally, the **computational cost** (Fig. A.3) was evaluated in terms of relative elapsed time. The transition up to 173 volumes results in a moderate increase in the computational effort. However, adopting the 210-volume mesh causes a dramatic spike, bringing the relative elapsed time to over 350%. This exponential growth in computational cost is directly tied to the Courant-Friedrichs-Lewy (CFL) condition: the presence of highly refined, extremely small fluid volumes forces the ATHLET implicit solver to drastically reduce the internal time step to preserve numerical stability.

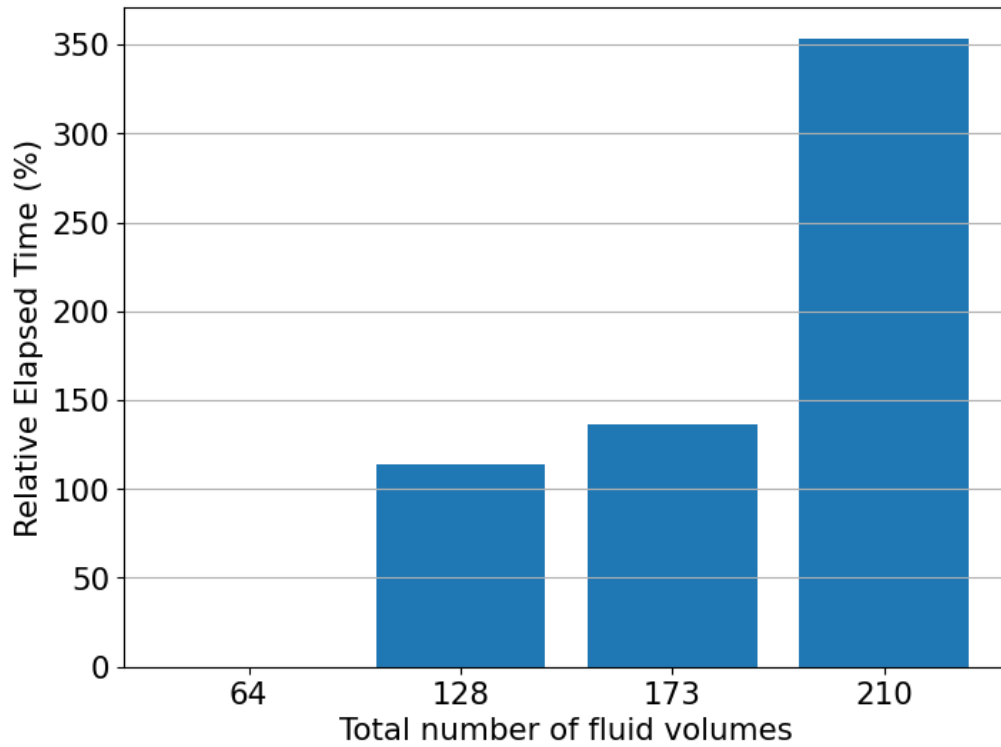


Figure A.3: *Nodalization Study on the Computational Cost.*

In conclusion, the nodalization study clearly demonstrates that the 128-volume mesh would represents the optimal choice. It ensures a stable and accurate evaluation of both thermal and hydraulic parameters, avoiding the pressure anomalies seen in the 173-volume mesh, while maintaining highly efficient computational performance compared to the 210-volume setup. Generally, the 64-volume configuration has been defined adopted as the nominal ATHLET grid for the subsequent coupled simulations, because the data are considered generally acceptable and because in current and future analyses it is important to have an ATHLET model with a low computational cost in view of performing transients.

Appendix B

Python Master Driver Script

The following script, `athlet_coupling.py`, contains the logic for the explicit multi-scale coupling between the ATHLET system code and the FRENETIC core solver.

```
1 # MODULAR FUNCTION FOR ATHLET COUPLING (EXPLICIT APPROACH)
2
3 # I have inserted some comments, in specific parts, to distinguish the
4 # command that will be used to run
5 # a single channel case or a multi-channel case (like ALFRED). The user can
6 # comment/uncomment the lines
7 # to adapt the code to the specific case.
8
9 def ATHLET_coupling(athlet_input: str, TH: Any) -> None:
10     # INSERTED FOR THE MULTI-CHANNEL CASE:
11     # def ATHLET_coupling(athlet_input: str, TH: Any, options: str) -> None:
12     """
13     This function manages the entire execution of ATHLET and the explicit
14     coupling with FRENETIC-TH.
15     It receives as input the ATHLET input file and the initialized TH
16     module.
17     """
18     print("Initialization of ATHLET for EXPLICIT coupled simulation...")
19
20     sys.path.append("/home/giorgiaversino/ATHLET/AC2-2023.4/extensions/
21     python")
22     sys.path.append("/home/giorgiaversino/ATHLET/AC2-2023.4/extensions/fde"
23     )
24
25     from Athlet._controller import Controller
26
27     # Start the ATHLET engine linking the Fortran shared library
28     engine_Athlet = Controller(lib="/home/giorgiaversino/ATHLET/AC2-2023.4/
29     bin/libathlet.so")
30
31     # VARIABLES TO MANAGE THE SYNCHRONIZATION STATE AND TO PERFORM
32     STABILITY MONITORING:
33     sync_manager = {
```

```

26     "dt_sync": 1.0, # Target synchronization interval (in seconds) for
coupling
27     "accumulated_dt": 0.0, # Variable to save the sum of the ATHLET's
dt
28     "is_synced": False, # Boolean flag to indicate if the
synchronization point has been reached
29     "history": {} # Dictionary to store the history of exchanged
variables for stability monitoring
30 }
31
32 # CREATION OF A CSV FILE TO SAVE THE EVOLUTION OF THE EXCHANGED
VARIABLES AND THEIR DERIVATIVES FOR POST-PROCESSING:
33 with open("coupling_history.csv", "w") as f_hist:
34     f_hist.write("Time_s,T_in_core_athlet_C,P_out_core_athlet_Pa,
Power_frenetic_W,T_out_frenetic_C,P_in_frenetic_Pa,T_out_core_athlet_C,
P_in_core_athlet_Pa,Cp_frenetic\n")
35     # FOR THE MULTI-CHANNEL CASE:
36     # f_hist.write("Time_s,T_in_core_athlet_C,P_out_core_athlet_Pa,
Power_frenetic_W,Power_from_Athlet,T_out_frenetic_C,P_in_frenetic_Pa,
T_out_core_athlet_C,P_in_core_athlet_Pa,Cp_frenetic,mdot_frenetic,
mdot_athlet,T_cool_max_C,T_clad_max_C\n")
37
38 # STABILITY MONITORING FUNCTION
39 def evaluate_stability(var_name: str, current_val: float, current_time:
float, buffer_size: int = 5):
40     """
41     Generic function to evaluate oscillations and instabilities for any
exchanged variable.
42     It uses a rolling window (buffer) to compute smoothed derivatives
and avoid single-step numerical noise.
43     """
44
45     # DYNAMIC INITIALIZATION OF THE HISTORY BUFFER: Check if the passed
variable (like T_inlet) already has a tracking history.
46     # If not, creates a sub-dictionary with empty lists to store its
past states.
47     if var_name not in sync_manager["history"]:
48         sync_manager["history"][var_name] = {
49             "times": [], # List of past time steps
50             "values": [], # List of past physical values
51             "local_derivs": [] # List of past instantaneous derivatives
to check for oscillations
52         }
53
54     hist = sync_manager["history"][var_name]
55
56     # UPDATE THE HISTORY BUFFER WITH THE CURRENT VALUE AND TIME: Append
the newest time and value at the end of
57     # the lists (which becomes index [-1]).
58     hist["times"].append(current_time)

```

```

59     hist["values"].append(current_val)
60
61     # ROLLING WINDOW MAINTENANCE: If the buffer exceeds the defined
62     # size, remove the oldest element (index [0]) to maintain a fixed-size
63     # history.
64     if len(hist["times"]) > buffer_size:
65         hist["times"].pop(0)
66         hist["values"].pop(0)
67
68     # MINIMUM DATA CHECK: A derivative requires at least two points. If
69     # the buffer only contains one point,
70     # bypass the mathematical evaluation and return safe/default values
71     # with a status indicating that the system is still initializing.
72     if len(hist["values"]) < 2:
73         return 0.0, 0.0, "INITIALIZING"
74
75     # SMOOTHED DERIVATIVE CALCULATION: Calculate a broader derivative
76     # using the oldest point [0] and the newest point [-1] in the buffer.
77     # This helps to smooth out short-term numerical noise and provides
78     # a more stable indication of the overall trend.
79     dt_broad = hist["times"][-1] - hist["times"][0]
80     dv_broad = hist["values"][-1] - hist["values"][0]
81     smoothed_deriv = dv_broad / dt_broad if dt_broad != 0 else 0.0
82
83     # RELATIVE ERROR CALCULATION: Evaluates the magnitude of the broad
84     # change relative to the current absolute value.
85     rel_change = abs(dv_broad) / abs(current_val) if current_val != 0
86     else 0.0
87
88     # OSCILLATION DETECTION: To detect sudden oscillations, calculate
89     # the instantaneous derivative between the last two points in the buffer
90     # (newest point [-1] and strictly previous point [-2]).
91     dt_local = hist["times"][-1] - hist["times"][-2]
92     dv_local = hist["values"][-1] - hist["values"][-2]
93     local_deriv = dv_local / dt_local if dt_local != 0 else 0.0
94
95     # Save the local derivative in the derivative buffer and maintain
96     # its size with the same rolling window logic.
97     hist["local_derivs"].append(local_deriv)
98     if len(hist["local_derivs"]) > buffer_size:
99         hist["local_derivs"].pop(0)
100
101     # Default status assumption:
102     status = "MONOTONIC"
103
104     # SIGN CHANGE CHECK:
105     if len(hist["local_derivs"]) >= 2:
106         # If the product of the current slope [-1] and the previous
107         # slope [-2] is strictly negative,

```

```

97         # it mathematically proves that the signal inverted its
          direction, and therefore an oscillation is occurring.
98         if hist["local_derivs"][-1] * hist["local_derivs"][-2] < 0:
99             status = "OSCILLATION DETECTED"
100
101     return smoothed_deriv, rel_change, status
102
103     # CALLBACK FUNCTIONS (HOOKS) DEFINITION
104     def on_input_start():
105         print("ATHLET is parsing the .in file...")
106
107     def on_input_done():
108         print("ATHLET input reading completed. Memory allocated.")
109
110     def on_after_febe():
111         """
112         Triggered only when ATHLET has successfully converged its
          macroscopic time step.
113         This is the core of the explicit coupling synchronization and
          stability monitoring.
114         """
115         actual_dt = engine_Athlet.dt.value
116         actual_time = engine_Athlet.t.value
117
118         # SUM OF THE TIME STEPS:
119         sync_manager["accumulated_dt"] += actual_dt
120
121         # CONTROL ON THE TRIGGER CONDITION:
122         if sync_manager["accumulated_dt"] >= sync_manager["dt_sync"]:
123             sync_manager["is_synced"] = True
124
125             print("Before sync point")
126             # Both codees must arrive at the same time before exchanging
          data.
127             TH.th_advance_time_to_sync_point(sync_interval=sync_manager["
          accumulated_dt"])
128             print("After sync point")
129
130             # The alignment is done!
131
132             # VARIABLES DEFINED BY ATHLET FOR A SINGLE CHANNEL CASE::
133             T_in_core_athlet_C = engine_Athlet.TF0_map["TDV"].tl[-1]
134             P_out_core_athlet_Pa = engine_Athlet.TF0_map["PIPE"].press[0]
135
136             T_out_core_athlet_C = engine_Athlet.TF0_map["PIPE"].tl[0]
137             P_in_core_athlet_Pa = engine_Athlet.TF0_map["TDV"].press[-1]
138
139             # # VARIABLES DEFINED BY ATHLET FOR A MULTI-CHANNEL CASE::
140             # T_in_core_athlet_C = engine_Athlet.TF0_map["RC"].tl[0]
141             # P_out_core_athlet_Pa = engine_Athlet.TF0_map["RC"].press[-1]

```

```

142
143     # T_out_core_athlet_C = engine_Athlet.TFO_map["RC"].t1[-1]
144     # P_in_core_athlet_Pa = engine_Athlet.TFO_map["RC"].press[0]
145
146     # mdot_from_athlet = engine_Athlet.TFO_map["RC"].glj[0]
147
148
149     # Writing to text file for FRENETIC:
150     with open("athlet_to_frenetic.txt", "w") as f:
151         f.write(f"{T_in_core_athlet_C}\n")
152         f.write(f"{P_out_core_athlet_Pa}\n")
153
154
155     #STABILITY MONITORING LOGIC:
156     smoothed_deriv_T, rel_change_T, status_T = evaluate_stability(
157         var_name="T_inlet_frenetic",
158         current_val=T_in_core_athlet_C,
159         current_time=actual_time,
160         buffer_size=5
161     )
162
163     # Print the stability report:
164     print(f"\n[SYNC TRIGGER] Reached the interval at t={actual_time
165           :.2f} s")
166     print(f"   --- STABILITY MONITORING REPORT ---")
167     print(f"   T_outlet: {T_in_core_athlet_C:.2f} C | Relative
168           Change: {rel_change_T*100:.4f}%")
169     print(f"   Derivative T: {smoothed_deriv_T:.4f} C/s | Status: {
170           status_T}")
171
172     # Example of fixing a safety threshold: warn if the relative
173     error is greater than 5%
174     # (this can be used to trigger a relaxation in the future)
175     if rel_change_T > 0.05:
176         print("   [WARNING] High temperature variation detected!
177               Possible instability.")
178
179     # FRENETIC writes its file (from Fortran) to provide the values
180     to ATHLET:
181     TH.th_write_frenetic_txt()
182
183     # READING AND UPDATING THE VALUES FOR THE NEXT MACROSTEP:
184     FRENETIC reads
185     # the ATHLET results to update the boundary conditions
186     TH.th_read_athlet_txt()
187
188     integral_power_W = 0.0
189     T_out_frenetic_C = 0.0
190     try:

```

```

185         with open("frenetic_to_athlet.txt", "r") as f:
186             lines = f.readlines()
187             integral_power_W = float(lines[0].strip())
188             T_out_frenetic_C = float(lines[1].strip())
189             P_in_frenetic_Pa = float(lines[2].strip())
190             cp_frenetic_J_kgK = float(lines[3].strip())
191             # VALUES INSERTED ONLY FOR THE MULTI-CHANNEL CASE:
192             # mdot_tot = float(lines[4].strip())
193             # T_cool_max_C = float(lines[5].strip())
194             # T_clad_max_C = float(lines[6].strip())
195
196             # INJECTION OF THE NEW VALUES IN ATHLET THROUGH GCSM::
197             def temp_in_from_frenetic(t, dt, v_t0):
198                 return T_out_frenetic_C # (T_out of FRENETIC
199 becomes T_in of ATHLET)
200             def press_out_from_frenetic(t, dt, v_t0):
201                 return P_in_frenetic_Pa # (P_in of FRENETIC
202 becomes P_out of ATHLET)
203             engine_Athlet.connect_gcsms_std("IN.TEMP",
temp_in_from_frenetic)
204             engine_Athlet.connect_gcsms_std("P.TDV",
press_out_from_frenetic)
205
206             except Exception as e:
207                 print(f" [ERROR] Python was unable to read the file: {e}")
208
209             # POWER INJECTION: UPDATING THE POWER MAP IN ATHLET
210             try:
211                 # Power injection through GCSM signal (to adapt to the
212 SimpleLoop.in file)
213                 def power_from_frenetic(t, dt, v_t0):
214                     return integral_power_W
215                     # VALUE INSERTED ONLY FOR THE MULTI-CHANNEL CASE:
216                     # return integral_power_W/22050 # --> 22050 is the
217 total number of fuel pins of ALFRED
218                 engine_Athlet.connect_gcsms_std("S.POW", power_from_frenetic
219 )
220                 print(f" [INFO] Power injected into ATHLET HCOs with
221 success.")
222
223                 # VALUES INSERTED ONLY FOR THE MULTI-CHANNEL CASE:
224                 # integral_power_from_athlet = engine_Athlet.gcsms["S.POW
225 "]["value"].value
226                 # integral_power_from_athlet = integral_power_from_athlet *
227 22050
228             except KeyError:
229                 # If the .in file does not have HCOs, skip the injection
230 without crashing
231                 print(f" [WARNING] No HCO objects found in ATHLET. Power
232 not injected (Test Mode).")

```

```

222         # SAVE OF THE EXCHANGED VARIABLES IN THE HISTORY CSV FILE FOR
223         POST-PROCESSING:
224         try:
225             with open("coupling_history.csv", "a") as f_hist:
226                 f_hist.write(f"{actual_time:.4f},{T_in_core_athlet_C:.4f},{P_out_core_athlet_Pa:.4f},{integral_power_W:.4f},{T_out_frenetic_C:.4f},{P_in_frenetic_Pa:.4f},{T_out_core_athlet_C:.4f},{P_in_core_athlet_Pa:.4f},{cp_frenetic_J_kgK:.4f}\n")
227                 # FOR THE MULTI-CHANNEL CASE:
228                 f_hist.write(f"{actual_time:.4f},{T_in_core_athlet_C:.4f},{P_out_core_athlet_Pa:.4f},{integral_power_W:.4f},{integral_power_from_athlet:.4f},{T_out_frenetic_C:.4f},{P_in_frenetic_Pa:.4f},{T_out_core_athlet_C:.4f},{P_in_core_athlet_Pa:.4f},{cp_frenetic_J_kgK:.4f},{mdot_tot:.4f},{mdot_from_athlet:.4f},{T_cool_max_C:.4f},{T_clad_max_C:.4f}\n")
229         except Exception as e:
230             print(f" [WARNING] Impossible to open coupling_history.csv : {e}")
231
232         print(f"\n[TXT-SYNC] Alignment a t={actual_time:.2f} s. Exchange completed.")
233
234         sync_manager["accumulated_dt"] = 0.0
235         sync_manager["is_synced"] = False
236     else:
237         # If the simulation is not at "dt_sync" yet, FRENETIC is not called.
238         # ATHLET will continue using the power calculated at the last sync point.
239         pass
240
241     # Steady State a t=0 for FRENETIC:
242     print("\n[INIT] Steady State Execution of FRENETIC (at t = 0)...")
243     TH.th_init_steady_state_mod()
244
245     print("[INIT] Write the initial conditions for ATHLET...")
246     # This is done in order to avoid the possibility that ATHLET runs with zeros
247     TH.th_write_frenetic_txt()
248
249     # HOOKS REGISTRATION
250     engine_Athlet.hooks.connectCallback("ATHLET_AfterPreset", on_input_start)
251     engine_Athlet.hooks.connectCallback("ATHLET_InputDone", on_input_done)
252     engine_Athlet.hooks.connectCallback("ATRANS_AfterFEBE", on_after_febe)
253
254     try:
255         # engine_Athlet.start(args=[athlet_input, "SimCoupled", options])
256         engine_Athlet.start(args=[athlet_input, "SimCoupled"])

```

```
257     finally:
258         print("Terminating ATHLET engine...")
259         engine_Athlet.terminate()
```

Listing B.1: *Modular Python script for the explicit coupling between ATHLET and FRETIC-TH.*

Appendix C

Material properties

The user will be able to select one among the available correlations for the liquid metal density, viscosity, specific heat and thermal conductivity as already implemented. It should be noticed that adding further materials or different databases in the future is relatively straightforward.

C.1 Coolant properties

The user can choose between lead, LBE and sodium correlations according to [29]. With respect to the previous implementation, the selection now occurs via keywords rather than via numbers, thus simplifying the code utilization. The following nomenclature was adopted:

- 'Pb'
- 'LBE'
- 'Na'
- 'Na-RELAP'

C.2 Fuel rod properties

The user can choose between two different kinds of fuel and non-fuel materials. The following keywords are available:

- 'UO2'
- 'U5Fs'
- 'B4C'
- 'SS'

C.3 Wrapper properties

Wrapper material properties can be chosen by selection from the input file. They are listed inside cladding materials properties since cladding and wrapper materials are typically selected from the same set. In addition to the previously implemented stainless steels, other possible austenitic steels and ferritic-martensitic steels are adopted, going towards HLMCR constraints and design choices.

The labels for the newly introduced correlations are:

- '15-15Ti'

$$c_p = 431.0 + 0.177 \cdot T + \frac{8.72 \cdot 10^{-5}}{T^2} \left[\frac{\text{J}}{\text{kg K}} \right] \quad (\text{C.1})$$

$$k = 8.826 + 1.707 \cdot 10^{-2} \cdot T - 2.315 \cdot 10^{-6} \cdot T^2 \left[\frac{\text{W}}{\text{m K}} \right] \quad (\text{C.2})$$

- 'T91'

$$c_p = 244.0 + 4.677 \cdot T - 1.103 \cdot 10^{-2} \cdot T^2 + 1.151 \cdot 10^{-5} \cdot T^3 - 3.979 \cdot 10^{-9} \cdot T^4 \left[\frac{\text{J}}{\text{kg K}} \right] \quad (\text{C.3})$$

$$k = 17.94 + 2.51 \cdot 10^{-2} \cdot T - 1.45 \cdot 10^{-5} \cdot T^2 \left[\frac{\text{W}}{\text{m K}} \right] \quad (\text{C.4})$$

- 'SS316'

$$\rho = 8084 - 0.4209 \cdot T - 3.894 \cdot 10^{-5} \cdot T^2 \left[\frac{\text{kg}}{\text{m}^3} \right] \quad (\text{C.5})$$

$$c_p = 462.0 - 0.134 \cdot T \left[\frac{\text{J}}{\text{kg K}} \right] \quad (\text{C.6})$$

$$k = 9.248 + 1.157 \cdot 10^{-2} \cdot T \left[\frac{\text{W}}{\text{m K}} \right] \quad (\text{C.7})$$

- 'He'

New temperature-dependent values for the modelling of the helium gap are also given, providing isobaric properties at 5 bar and a conduction correlation only as a function of temperature, as the one used in the MATPRO code:

$$k = 2.639 \cdot 10^{-3} \cdot T^{0.7085} \left[\frac{\text{W}}{\text{m K}} \right] \quad (\text{C.8})$$

Due to the lack of available correlations in references for a wide range of temperatures for T91 and 15-15Ti materials, their densities are implemented as the one for stainless steels and need to be updated in the future.

Appendix D

Constitutive relations

Constitutive correlations of empirical nature are employed with the possibility for the user to choose among the available correlations for the friction factor and the Nusselt number. The user can choose between them in the input file using a more readable nomenclature through string labels, calling the adopted correlation properly, as for the materials. Here they are listed with the new label, type of correlation, and validity range.

D.1 Friction factor

- 'CONST': constant $f = 0.005$
- 'BUNDLE': laminar/turbulent along rod bundles.

$$f_T = \frac{C_{f,T}}{Re^{0.18}}, \quad f_L = \frac{C_{f,L}}{Re} \quad (D.1)$$

$$C_{f,L|T} = a + b_1(P/D - 1) + b_2(P/D - 1)^2 \quad (D.2)$$

Where coefficients for $C_{f,L}$ and $C_{f,T}$ are reported in Table D.1.

	1.0 ≤ P/D ≤ 1.1			1.1 < P/D ≤ 1.5		
Subchannel	a	b ₁	b ₂	a	b ₁	b ₂
<i>Laminar flow</i>						
Interior	26.00	888.2	-3334	62.97	216.9	-190.2
Edge	26.18	554.5	-1480	44.40	256.7	-267.6
Corner	26.98	1636	-10050	87.26	38.59	55.12
<i>Turbulent flow</i>						
Interior	0.09378	1.398	-8.664	0.1458	0.03632	-0.03333
Edge	0.09377	0.8732	-3.341	0.1430	0.04199	0.04428
Corner	0.1004	1.625	-11.85	0.1499	0.006706	-0.009567

Table D.1: Coefficients for bare rod subchannel friction factor constants in hexagonal array.

- ‘BLASIUS’: Blasius correlation. Blasius has proposed a simple correlation for the friction factor in a smooth circular tube:

$$f = \frac{0.316}{Re_{D_h}^{0.25}} \quad (D.3)$$

where $Re_{D_h} = (\rho v D_h)/\mu$ is the Reynolds number computed using the hydraulic diameter as the characteristic length. This correlation is applicable in turbulent regime up to $Re < 10^5$.

- ‘WIRED’: Pressure drop in wire-wrapped hexagonal array pin bundles.

$$f_T = \frac{C_{fT}}{Re^{0.18}}, \quad f_L = \frac{C_{fL}}{Re} \quad (D.4)$$

$$C_{fL} = [-974.6 + 1612.0(P/D) - 598.5(P/D)^2] (H/D)^{0.06-0.085(P/D)} \quad (D.5)$$

$$C_{fT} = [0.8063 - 0.9022(\log_{10}(H/D))] + 0.3256[\log_{10}(H/D)]^2 (P/D)^{9.7} (H/D)^{1.78-2.0(P/D)} \quad (D.6)$$

Correlations 1 and 3 are a function of Reynolds number, rod pitch and diameters, of rods and wire wrapper.

D.2 Nusselt number

The heat-transfer correlations for liquid metals are derived from experiments performed in dedicated facilities or CFD simulations, and are specific for the kind of rod bundles in terms of shape (e.g. triangular lattice or squared) and extension (7-19-37 pins etc.). Due to the significant spread of these experimental data, care should be taken when applying such

correlations, also bearing in mind that none was developed specifically for wire-wrapped pins.

- 'BiB': Seban-Shimazaki, liquid metals flow in smooth pipe, uniform axial wall temperature and uniform radial heat flux.

$$Nu = 5.0 + 0.025Pe^{0.8} \quad (D.7)$$

- 'SCHAD': Schad-modified.

$$Nu = \begin{cases} 4.496 [-16.15 + 24.96(P/D) - 8.55(P/D)^2] & \text{for } Pe \leq 150 \\ [-16.15 + 24.96(P/D) - 8.55(P/D)^2] Pe^{0.3} & \text{for } 150 < Pe < 1000 \end{cases} \quad (D.8)$$

Valid for $1.3 \leq P/D \leq 2$.

- 'USHAKOV': Ushakov.

$$Nu = 7.55(P/D) - 20.0(P/D)^{-13} + 0.041(P/D)^{-2}Pe^{0.56+0.19(P/D)} \quad (D.9)$$

Valid for $150 \leq Pe \leq 4000$ and $1.3 \leq P/D \leq 2$.

- 'EXPERIMENTAL': Interpolation based on experimental data obtained at the Brasi-mone research center.
- 'MIKITYUK': Mikityuk Fully Developed Flow in Bare Bundles in triangular lattice.

$$Nu = 0.047 \left(1 - e^{-3.8(P/D)-1.0}\right) (Pe^{0.77} + 250.0) \quad (D.10)$$

Valid for $30 \leq Pe \leq 5000$ and $1.1 \leq P/D \leq 1.95$.

- 'KAZIMI': Westinghouse Carelli-Kazimi.

$$Nu = 4.0 + 0.33(P/D)^{3.8}(Pe/100.0)^{0.86} + 0.16(P/D)^5 \quad (D.11)$$

Valid for $10 \leq Pe \leq 5000$ and $1.1 \leq P/D \leq 1.4$.

A comparison of experimental results of wired pins against several correlations concluded that the average Nu was lower than that predicted by the majority of the correlations for bare rod bundles at the same Pe. Results showed values between the Carelli-Kazimi and the Mikityuk correlations, and the slope of the experimental trend is very similar to the correlations. For wire-wrapped fuel, the Carelli-Kazimi correlation being more conservative fits near to the available data so is suggested. For fully developed flow in bare bundles, the derived correlation by Mikityuk shows good agreement with data. To appreciate the differences between correlations a comparison in the prediction of an average heat transfer inside an assembly as a function of Pe number, at 700 K and $P/D = 1.32$, in their validity range, is shown in Figure.

Appendix E

Acronyms

ALFRED	Advanced Lead-cooled FR European Demonstrator
ANL	Argonne National Laboratory
ATHLET	Analysis of THERmal-hydraulics of LEaks and Transients
BDBA	Beyond Design Basis Accident
BiB	Box-in-the-Box
BOP	Balance Of Plant
BWR	Boiling Water Reactor
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy
CLV	Core Lower Vessel
CRP	Coordinated Research Project
CUBBOX	CUBic BOX
CUV	Core Upper Vessel
CUV.W	Core Upper Vessel Wall
CV	Control Volume
DHR	Decay Heat Removal
DOCs	Design-Oriented Codes
DPA	Displacement Per Atom
ENEA	Ente Nazionale Energia Atomica
FA	Fuel Assembly
FEBE	Forward-Euler, Backward-Euler
FRENETIC	Fast REactor Neutronics/Thermal-hydraulICS
FVM	Finite Volume Method
GCSM	General Control Simulation Module
GFR	Gas-Cooled Fast Reactor
GIF	Generation IV International Forum
GRS	Gesellschaft für Anlagen- und Reaktorsicherheit
HA	Hexagonal Assembly

HCOs	Heat Conduction Objects
HECU	Heat Conduction and Heat Transfer
HLMCR	Heavy Liquid Metal Cooled Reactor
I/O	Input/Output
IAEA	International Atomic Energy Agency
ICE	Integral Circulation Experiment
IW	Inter-Wrapper
LBE	Lead-Bismuth Eutectic
LPC	Lower Pool Connection
LFR	Lead-Cooled Fast Reactor
LM	Liquid Metal
LMFR	Liquid Metal Fast Reactor
LWR	Light Water Reactor
MSR	Molten Salt Reactor
NE	Neutronics
NEUKIN	Neutron Kinetics
ODEs	Ordinary Differential Equations
PARCS	Purdue Advanced Reactor Core Simulator
PID	Proportional Integral Derivative
PK	Point Kinetics
PL	Plenum
PLC	Plenum Connection
PL.C	Plenum Cover Gas
PL.P	Plenum Pipe
PWR	Pressurized Water Reactor
QUABOX	QUAdratic BOX
RC	Reactor Core
RCP	Reactor Coolant Pump
RPB	Reactor Pool Bottom
RPU	Reactor Pool Upper
RPUC	Chiedere
SA	Sub-Assembly
SCWR	SuperCritical-Water-cooled Reactor
SFR	Sodium-Cooled Fast Reactor
SG	Steam Generator
SG.W	Steam Generator Wall
TFD	Thermo-Fluid Dynamics
TFOs	Thermo-Fluid dynamic Objects
TH	Thermal-Hydraulics
TM	Thermo-Mechanics
ULOF	Unprotected Loss of Flow

ULOHS	Unprotected Loss of Heat Sink
UTOP	Unprotected Transient Overpower
VHTR	Very-High-Temperature Reactor
VOCs	Verification-Oriented Codes

Bibliography

- [1] G. Grasso. The core design of alfred, a demonstrator for the european lead-cooled reactors. *Nuclear Engineering and Design*, 2014. (cit. on pp. ix, xii, 72, 75, and 76).
- [2] Cristiano Ciurluini. Preliminary neutron kinetic – thermal hydraulic coupled analysis of the alfred reactor using phisics/relap5-3d. *Journal of Physics: Conference Series*, 2020. (cit. on pp. xi, 5, 6, and 74).
- [3] Francesco Lodi. *Development of Core Design Methods and Tools for Gen-IV Heavy Liquid Metal Cooled Reactors*. Ph.d. thesis, Alma Mater Studiorum - Università di Bologna, Bologna, Italy, 2017. (cit. on pp. xi, 8, and 16).
- [4] A. Lombardo, Giuseppe F. Nallo, N. Abrate, Y. Liu, and S. Dulla. Development of a New Thermal-Hydraulic Module for FRENETIC, a Code for the Multiphysics Analysis of Liquid Metal–Cooled Reactors. *Nuclear Technology*, 211:2490–2507, 2024. (cit. on pp. xi, 13, 17, and 18).
- [5] R. Bonifetto, S. Dulla, P. Ravetto, and R. Zanino. Thermal-hydraulic modeling of liquid metal cooled fast reactor sub-assemblies: The FRENETIC single-channel model. *Nuclear Engineering and Design*, 249:249–258, 2012. (cit. on pp. xi, 50, and 51).
- [6] United Nations Economic Commission for Europe (UNECE). Technology Brief: Nuclear Power. Technical report, United Nations, Geneva, Switzerland, 2021. (cit. on p. 1).
- [7] Generation IV International Forum. GIF R&D Outlook for Generation IV Nuclear Energy Systems. Technical report, Generation IV International Forum (GIF), August 2009. (cit. on pp. 1, 2, and 4).
- [8] Dominique M Warin. Developments in the partitioning and transmutation of radioactive waste. In *Advanced Separation Techniques for Nuclear Fuel Reprocessing and Radioactive Waste Treatment*, pages 363–376. Elsevier, 2011. (cit. on p. 3).
- [9] Igor L. Pioro, editor. *Handbook of Generation IV Nuclear Reactors*. Woodhead Publishing, 2nd edition, 2022. (cit. on p. 3).

- [10] Carolyn Howlett and Richard Underhill. Nuclear Heat Transfer and Passive Cooling. Volume 5: Liquid Metal Thermal Hydraulics. Technical report, Frazer-Nash Consultancy, March 2023. (cit. on pp. 3 and 4).
- [11] The LFR provisional System Steering Committee. Safety Design Criteria for Generation IV Lead-cooled Fast Reactor System. Technical report, Generation IV International Forum (GIF), March 2021. Revision 1. (cit. on p. 8).
- [12] A Del Nevo, A Cervone, G Grasso, M Tarantino, T Barani, A Cammi, M Cerini, S Cervino, L Cognini, L De Luca, et al. Development of best estimate numerical tools for lfr design and safety analysis. 2017. (cit. on p. 8).
- [13] R. Bonifetto, S. Dulla, P. Ravetto, L. Savoldi Richard, and R. Zanino. A full-core coupled neutronic/thermal-hydraulic code for the modeling of lead-cooled nuclear fast reactors. *Nuclear Engineering and Design*, 261:85–94, 2013. (cit. on pp. 9 and 15).
- [14] G. F. Nallo, N. Abrate, S. Dulla, P. Ravetto, and D. Valerio. Neutronic benchmark of the FRENETIC code for the multiphysics analysis of lead fast reactors. *EPJ Web of Conferences*, 2018. EPJ manuscript. (cit. on pp. 9 and 13).
- [15] P. Schöffel, A. Di Nora, D. Eckert, M. Junk, D. von der Cron, F. Weyermann, and A. Wielenberg. *ATHLET 3.4.2 User’s Manual*. Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH, Garching, Germany, October 2024. GRS-P-1/Vol. 1, Rev. 13. (cit. on pp. 9, 19, 26, and 58).
- [16] Antonino Lombardo. Development and validation of a thermal-hydraulic model for the analysis of fast nuclear reactors in the FRENETIC code. Master’s thesis, Politecnico di Torino, Turin, Italy, July 2022. Tesi di Laurea Magistrale in Nuclear and Risk Engineering. (cit. on pp. 15 and 16).
- [17] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab®*, volume 113 of *Fluid Mechanics and Its Applications*. Springer, 2016. (cit. on p. 16).
- [18] S Langenbuch and K Velkov. Overview on the development and application of the coupled code system athlet-quabox/cubbox. 2005. (cit. on p. 25).
- [19] Nicolò Abrate. COREutils: A python library for FRENETIC pre- and post-processing, 2024. (cit. on pp. 31 and 42).
- [20] JJ Grudzinski and C Grandy. Fuel assembly bowing and core restraint design in fast reactors. In *ASME International Mechanical Engineering Congress and Exposition*, volume 46521, page V06BT07A026. American Society of Mechanical Engineers, 2014. (cit. on p. 35).
- [21] The HDF Group. *HDF5 User Guide*. The HDF Group, 2024. (cit. on pp. 41 and 67).

- [22] NumPy Developers. F2PY Users Guide and Reference Manual. <https://numpy.org/doc/stable/f2py/>, 2026. Accessed: 2026-07-09. (cit. on pp. 42 and 56).
- [23] Roberto Zanino, Roberto Bonifetto, A Ciampichetti, I Di Piazza, Laura Savoldi, M Tarantino, et al. First validation of the frenetic code thermal-hydraulic model against the enea integral circulation experiment. *Transactions of the American Nuclear Society*, 107:1395–1398, 2012. (cit. on p. 50).
- [24] M. Tarantino, P. Gaggini, I. Di Piazza, P. Agostini, N. Forgione, D. Martelli, and G. Barone. CIRCE Experimental Report. Technical report, ENEA, 2013. (cit. on p. 51).
- [25] Aseem Daga, Sergio de Cesare, and Mark Lycett. Separation of concerns: techniques, issues and implications. *Journal of Intelligent Systems*, 15(1-4):153–176, 2006. (cit. on p. 57).
- [26] V. Jacht, M. Junk, J. Scheuer, P. Schöffel, and A. Wielenberg. *ATHLET 3.4.2 Programmer’s Manual*. Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH, Garching, Germany, October 2024. GRS-P-1/Vol. 2, Rev. 13. (cit. on p. 57).
- [27] G Bandini and M Polidori. Report on the results of analysis of dec events for the etdr (alfred). *Deliverable DEL022-2013, EURATOM FP7 “LEADER” Project*, 2013. (cit. on p. 72).
- [28] R. Ponciroli, A. Cammi, S. Lorenzi, and L. Luzzi. A preliminary approach to the alfred reactor control strategy. *Politecnico di Milano, Department of Energy*, 2014. (cit. on p. 76).
- [29] Vitaly Sobolev. Database of thermophysical properties of liquid metal coolants for GEN-IV: Sodium, lead, lead-bismuth eutectic (and bismuth). Technical Report SCK CEN-BLG-1069, SCK CEN, Mol, Belgium, December 2011. (cit. on p. 103).