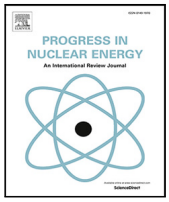
Progress in Nuclear Energy 169 (2024) 105022

Contents lists available at ScienceDirect

Progress in Nuclear Energy

journal homepage: www.elsevier.com/locate/pnucene



Investigating the Functional Mock-up Interface as a Coupling Framework for the multi-fidelity analysis of nuclear reactors

Thomas Guilbaud [a,b,*], Carlo Fiorina [c], Stefano Lorenzi [d], Alessandro Scolaro [a], Federico Carminati [b], Donovan Maire [b], Andreas Pautz [a]

[a] Laboratory of Reactor Physics and Systems Behaviour, EPFL, 1015 Lausanne, Switzerland
[b] Transmutex SA, 8 Chemin de Blandonnet, Vernier, Geneva, Switzerland
[c] Department of Nuclear Engineering, Texas A&M University, 423 Spence St, College Station, TX 77843, USA
[d] Politecnico di Milano, Department of Energy, Nuclear Engineering Division, Via La Masa 34, 20156 Milan, Italy

ARTICLE INFO

Keywords:
Multi-physics
Multi-fidelity
Functional Mock-Up Interface
Coupling standard

ABSTRACT

This paper investigates the use of the Functional Mock-up Interface (FMI) for code coupling in nuclear engineering. The FMI is a standard that defines a container and an interface to exchange dynamic simulation models. It has been developed and refined by several actors over the past two decades. This coupling standard allows seamless integrations of independent objects called Functional Mock-up Units (FMU). Since communication among FMUs is standardized, encapsulating a simulation tool and model within an FMU allows this tool and model to be coupled with any other FMU. This approach is opposed to creating dedicated code-to-code coupling interfaces and enables a more sustainable approach to code coupling in nuclear engineering. The paper showcases the utilization of the FMI standard for the simulation of an operational load-follow scenario in a Lead-cooled Fast Reactor. The primary circuit and balance of the plant are modeled using higher and lower-fidelity codes, respectively, with a third tool employed to model a simplified control system. The paper investigates the pros and cons of the proposed approach by exercising it throughout the following workflow: incorporation of the FMI standard into an existing code; setting up of models using different codes; coupling of these codes based on different architectures; simulation and post-processing of results. As an outcome, implementing an FMI interface presents itself as a judicious long-term investment for simulation software. However, users and developers should be aware of the limited FMI capabilities for the coupling of partial differential equations. In addition, a coupling standard by itself cannot address some difficulties, such as simulation restart, that are associated with the handling of a heterogeneous set of tools.

1. Introduction

The nuclear engineering community has dedicated large efforts to developing advanced multi-physics tools for the high-fidelity analysis of nuclear reactors. Many of these tools are based on MOOSE (Lindsay et al., 2022) or OpenFOAM (Fiorina et al., 2022) and focus on modeling the reactor core or, in some cases, the entire primary circuits. However, limited functionalities are typically available in these codes to model intermediate and secondary loops and the control/reactor protection system. When the behavior of the whole plant is of interest, the community tends to use nuclear-specific system codes like SAM (Hu et al., 2021), TRACE (U.S. Nuclear Regulatory Commission, 2008), and RELAP (Fletcher and Schultz, 1995), with the drawback of a lower fidelity at the core level. Finally, control-oriented studies are often performed with non-nuclear-specific, general-purpose tools like Modelica (OpenModelica, 2022) and Simulink (Simulink Documentation, 2020).

Several efforts have been dedicated to coupling high-fidelity tools with system and control-oriented tools, leading to multi-fidelity simulations where model fidelity is adapted to the impact of the behavior of a system/component on predicted results. For instance, various CFD/system codes coupling have been investigated, including RELAP5/STAR-CCM+ (Jeltsov et al., 2013), OpenFOAM/ATHLET (Herb, 2014), CATHARE/TRIO-U (Bandini et al., 2015), RELAP5/ANSYS Fluent (Angelucci et al., 2017), and RELAP7/MOOSE (Berry et al., 2016).

However, these couplings are typically code-specific, sometimes case-specific, and almost inevitably difficult to generalize to other

* Corresponding author at: Laboratory of Reactor Physics and Systems Behaviour, EPFL, 1015 Lausanne, Switzerland.
E-mail address: thomas.guilbaud@epfl.ch (T. Guilbaud).

https://doi.org/10.1016/j.pnucene.2023.105022
Received 14 September 2023; Received in revised form 28 November 2023; Accepted 11 December 2023
Available online 12 January 2024

codes. In many cases, they are in-house adaptations of one of the coupled tools that are not made available to the community. This hinders widespread adoption within the nuclear engineering community and increases code development and maintenance efforts while being error-prone since each code must be independently debugged. Researchers and engineers face challenges when attempting to explore alternative models or compare different coupling strategies due to the specificity of each coupling. Furthermore, the developed coupling infrastructure often suffers from a lack of documentation. Users may encounter obstacles when implementing or customizing the coupling for their specific needs without proper documentation. It is crucial to address these limitations to establish a sustainable and widely applicable coupling approach within the nuclear engineering community. There is a need for a code-agnostic approach, preferably open-source, transparent, and easy to implement. Modularity should also be prioritized, enabling interchangeable and extensible couplings.

The need for a generalized and comprehensive coupling standard is not exclusive to the nuclear domain. The European MODELISAR project was initiated in 2008 to enhance vehicles' systems and embedded software designs. This collaborative effort involved 29 partners, including prominent European universities and automakers. The project concluded in 2011 with the creation of the Functional Mock-up Interface (FMI) version 1.0 (Modelica Association, 2022). The primary objective behind FMI was to simplify the creation, storage, exchange, and reusability of dynamic system models across diverse simulation systems. The FMI essentially defines a container and an interface for the in-memory exchange of data between dynamic simulation models. Since communication is standardized, incorporation of the FMI standard within a simulation tool would allow this tool to communicate with any other FMI-compatible simulation software, without the need for countless code-to-code coupling interfaces. Of course, when building a coupling framework, there is still a significant workload in determining the coupling algorithm; achieving a high-order coupling when needed; handling multi-code parallelization; etc. However, a coupling standard virtually eliminates the labor-intensive software engineering process associated with the in-memory transfer of information between codes. In addition, once the coupling logic is set up, the process of replacing a code with another code would be essentially effortless. Over time, the core development of the FMI standard has transitioned into a Modelica Association Project, comprising a consortium of organizations and companies including BOSCH, Dassault Systemes, ESI ITI, Modelon, Siemens PLM, ABB, DLR, and the Open Modelica Consortium. The FMI standard has gained significant recognition and is integrated into more than 170 tools across diverse domains. Notably, major general-purpose tools like Matlab/Simulink and OpenModelica, as well as computational fluid dynamics (CFD) tools like ANSYS CFX and Simcenter STAR-CCM+, have incorporated the FMI standard into their frameworks. This widespread adoption of FMI underscores its versatility and applicability in various engineering disciplines.

This paper follows the precursor work on the use of FMI for nuclear applications performed at both the Oak Ridge National Laboratory for the coupling of sub-channel and system codes (Gurecky et al., 2020) and at the Idaho National Laboratory for the simulation of integrated energy systems (Alfonsi et al., 2021). The main objective of the current work is to test and demonstrate the use of an FMI-based methodology to enable the multi-fidelity simulation of nuclear power plants, from core to alternator, using a diverse set of modeling tools. In this approach, the reactor primary circuit, or part of it, is simulated using modern multi-physics tools, while the Balance of Plant (BOP) and control-system are simulated using dedicated, lumped-parameter models (or legacy system codes). In addition to providing information on the use of the FMI coupling standard for this specific set of problems, we hope to showcase the value of an FMI-based methodology and encourage more widespread use of this or other standardized coupling approaches in the field of nuclear engineering.

We simulate a Lead Fast Reactor (LFR) as a test case. We use the OpenFOAM-based GeN-Foam solver for the higher-fidelity representation of the primary pool and OpenModelica and Simulink for the system-level representation of the BOP and control system. The paper is organized as follows. Section 2 describes our methodology, including a description of tools and codes, integration of the FMI standard within GeN-Foam, and some details about simulation restart. In Section 3, we demonstrate the capabilities of the FMI by simulating an operational load-follow scenario using four different coupling architectures. Finally, we provide some discussions and conclusions in Section 4.

2. Methodology and tools

Our approach for a robust multi-fidelity simulation is to rely on the FMI standard to couple heterogeneous sets of tools in a seamless way. As a test case, in this work, we use the high-fidelity solver GeN-Foam for the reactor primary loop and the system tools Modelica and Simulink for the BOP and control system.

2.1. The functional mock-up interface

The FMI is a code coupling standard designed to couple among them objects known as Functional Mock-up Units (FMUs). The idea is that one can seamlessly couple any number of codes, provided that each of these codes can be embedded into an FMU. An FMU is a container that is provided with a standardized way to communicate with other FMUs. If a user is provided with multiple FMUs that perform the same function, switching from one to the other is supposed to be essentially effortless.

The FMI interface addresses the needs mentioned in the introduction: code-agnostic; open-source and transparent; applicable to different codes; easy to implement; and modular. On top of this, approximately 200 simulation tools already feature an FMI interface, providing an already rich ecosystem of coupling possibilities. Some technical key advantages associated with FMUs encompass the unlimited number of inputs or outputs, the versatile representation of these inputs/outputs as ports enabling the transfer of essential quantities such as integers, booleans, strings, floats, or arrays of floats, and the capability to directly couple FMUs to one another. Data exchange can be triggered at any point within a simulation software, which enables tightly coupled simulations.

The FMI exhibits two main coupling options depending on the user's preferences and the software infrastructure being utilized. One approach is the FMU for Model Exchange (ME), where FMUs solely include the model and exclude the solver. Running these ME FMUs requires the pre-installation and accessibility of the relevant software. A more interesting approach in our field, and the one investigated in this article, involves generating all-in-one FMUs that encompass both the model and solvers, referred to as FMI for Co-Simulation (CS) (see Fig. 1).

2.2. System-level codes for BOP and control system

System-level codes are relatively simple to use, fast running, and easier to validate. In this work, we use OpenModelica, an open-source software developed around the Modelica language (OpenModelica, 2022). It allows for solving Differential Algebraic Equations in a wide range of applications. It can simulate electronic, signal processing, fluid, thermal, magnetic, and mechanical systems. OpenModelica allows users to implement their own packages and models to simulate specific systems. One of them is the ThermoPower package (ThermoPower, 2022) that allows performing simulations of power plant turbo-machinery for power generation. In addition to OpenModelica, we use MATLAB/Simulink (Simulink Documentation, 2020) to showcase the combination of multiple tools interchangeably. Both OpenModelica and Simulink are already provided with an FMI interface that allows encapsulating models within exportable FMUs. In the present work, we use the FMI for CS form for both OpenModelica and Simulink.

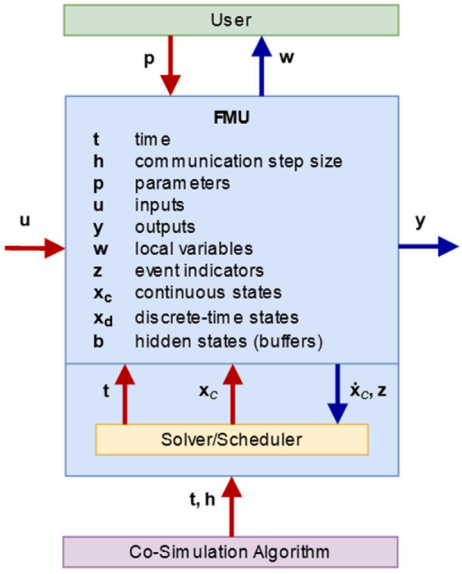


Fig. 1. Schematic view of the FMU for Co-simulation from the FMI project (Modelica Association, 2022).

2.3. High-fidelity simulation code for the primary circuit

High-fidelity, multi-dimensional, and multi-physics codes are increasingly popular for modeling the intricacies of nuclear reactors, notably for advanced and non-traditional reactors with limited calibration data. In this work, we make use of GeN-Foam (Generalized Nuclear Field operation and manipulation), which is an OpenFOAM-based solver for the multi-physics, high-fidelity analysis of nuclear reactors (Fiorina et al., 2015; Fiorina, 2022). It has been developed for steady-state and transient analyses of reactors featuring pin-type, plate-type, or liquid fuel (viz., Molten Salt Reactors). It includes sub-solvers for neutronics, single- and two-phase thermal-hydraulics, and thermal-mechanics, with the choice of the physics to solve and models to use that can be made at runtime. Three different meshes are employed for neutronics, thermal-hydraulics, and thermal-mechanics. The coupling between physics is obtained via fixed-point iterations, and the time derivatives are solved based on a first-order implicit Euler scheme.

There is currently no FMI adapter in the official OpenFOAM framework. To provide GeN-Foam with FMI coupling capabilities, we used the FMU4FOAM library (FMU4FOAM, 2022) developed by the German Aerospace Center (DLR). FMU4FOAM is a compact and extensible open-source project that allows the implementation of the FMI standard in OpenFOAM-based solvers. Through FMU4FOAM, one can either encapsulate an OpenFOAM model in an FMU for ME or allow an OpenFOAM solver to be directly coupled with other FMUs without conversion of the model itself into an FMU. In the latter, FMU4FOAM uses a flexible, FMI-compatible communication layer for exchanging information between OpenFOAM and FMUs, and an FMU Simulator that embeds a Python interpreter to manage the FMU communication and information transfer.

FMU4FOAM comes with several input/output capabilities, including, in particular, the possibility to place sensors in OpenFOAM models to pass point quantities to other FMUs, and the possibility to set boundary conditions such as inlet temperatures or velocities based on the input from FMUs.

New FMI-based features have been introduced in GeN-Foam to enable nuclear reactor analysis (Fiorina et al., 2023). These features have been made publicly available in the GeN-Foam repository (GeN-Foam, 2022). They include new capabilities to allow controlling the reactor based on FMUs, including in particular:

- The setting of the pump momentum source from the FMUs' input, which can be used to model variable-speed pumps. This can be combined with the addition of mass flow rate sensors to simulate closed-loop controllers.
- The setting of an external reactivity input for the point-kinetics sub-solver (Radman et al., 2022), which can be used to model the effect of control rods.
- The modulation of the external neutron source in the point-kinetics sub-solver for sub-critical scenarios (a reactor start-up or the operation of an accelerator-driven system).

In addition, the volume integral of user-selectable fields can be output to FMUs. This is useful, for instance, to pass the core power to an FMU-based control system, though, of course, one could use a standard flux sensor and replicate the actual situation in a nuclear reactor.

The possibility to pass volume integrals to an FMU is also crucial to simulate the heat exchange between a primary loop modeled with GeN-Foam, and a secondary loop modeled with FMUs, thus enabling the set-up of multi-fidelity simulations. From the perspective of the primary circuit, one can simulate the secondary side of a heat exchanger as a fixed-temperature sub-scale "structure" (the fluid in the secondary/intermediate loop), thermally separated from the primary coolant via thermal resistances of the primary coolant, of the heat-exchanger walls, and of the secondary coolant. An FMU of the secondary/intermediate loop will provide GeN-Foam with information about the temperature(s) and the Nusselt number in the secondary/intermediate loop. This required the implementation of new functionalities to:

- Set the temperature of a sub-scale structure in a porous medium based on the input from FMUs;
- Calculate a heat resistance towards a sub-scale structure based on: the Nusselt number of the fluid, a fixed heat conductance, and an additional heat conductance provided by an FMU.

Based on this information, GeN-Foam passes volume integrals to FMUs to calculate the integral power transferred from primary to secondary/intermediate loops and gives this information to the FMU that calculates temperatures in the secondary/intermediate loops. It is possible to split the heat exchanger in GeN-Foam into multiple fixed-temperature zones to adapt to the nodalization of the FMU model of the secondary/intermediate loop. An example of this strategy is detailed in the next Section. All the features presented above can be used in multiple places of a single simulation.

2.4. Coupling architecture and algorithm

To assess the modularity and flexibility of the proposed solution, we have modeled a nuclear power plant using four architectures that differ based on how data transfer is managed among codes. These architectures are summarized in Fig. 2. Architectures a. and b. use GeN-Foam as the master code coupled via FMU4FOAM to FMUs, respectively, Modelica in case a., and Modelica and Simulink in case b.. Cases c. and d. treat all the models as FMUs with a single Python script to perform the coupling. All of these architectures rely on open-source Python modules to simplify the communication and manipulation of FMUs, such as `pyfmi` (Andersson et al., 2016), `fmpy` (Dassault Systèmes, 2023), or `omsimulator` (Ochel et al., 2019). On the one hand, using GeN-Foam as a master allows a somewhat simplified control of the time step, which is usually smaller in multi-physics codes compared to system codes. An FMI-compatible interface is also easier to implement in a code with respect to encapsulating the code into an FMU. On the other hand, a full FMU architecture allows a more modular approach where all the models are coupled and managed in a simple Python script and where the FMUs can easily be exchanged with other FMUs performing the same tasks.

In those cases where GeN-Foam is used as a master code, the FMU4FOAM library uses an FMU Simulator to handle communications

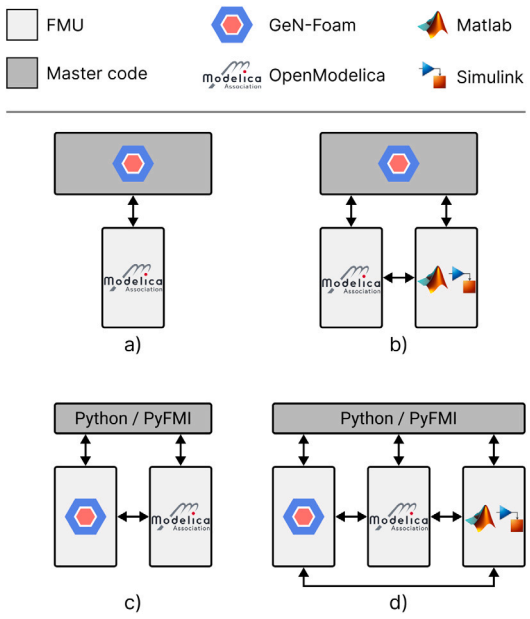


Fig. 2. Multi-fidelity architecture schemes tested during a load-follow scenario.

with FMUs. The FMU Simulator is implemented as an OpenFOAM function object, which is an instance that can be called at the end of each time step by an OpenFOAM-based solver and that often describes a post-processing action to extract results from an OpenFOAM simulation at runtime. The FMU Simulator relies on an embedded Python interpreter that can use open-source Python frameworks such as `pyfmi`, `fmpy`, or `omsimulator` to manage and communicate with the FMUs, and can be instructed to perform specific tasks via standard Python scripting.

At initialization, the FMU Simulator loads the FMUs in memory with the embedded Python interpreter. In the case of multiple FMUs, a connection map is instantiated to instruct how to connect the input/output ports of the various FMUs. When executed, the FMU Simulator transfers the information contained in the communication layer to the FMUs. Then, all FMUs are instructed by the FMU Simulator to perform a time step to align with the GeN-Foam simulated time. Finally, the data from the FMUs are extracted by the FMU Simulator and saved in the communication layer, from where they can be used by the various GeN-Foam classes.

Our current algorithm creates an explicit coupling between GeN-Foam and the FMUs, in the sense that a single bi-directional information exchange is performed at every time step. As shown in the next section, an explicit scheme is sufficient in this work to guarantee a stable solution when performing a coupling between primary and secondary circuits at the level of the heat exchangers. In addition, the default behavior of function objects can be overridden, and the FMU Simulator can be triggered at any point during a simulation. This allows for more implicit coupling schemes that might be necessary for other types of code couplings, such as for a pressure-velocity coupling when simulating the thermal-hydraulics of core and piping/pools using different tools. This latter point will be part of future developments.

When all the models are embedded in FMUs, writing a simple script to couple the units during the simulation, for instance, using the previously mentioned Python packages, is necessary. This approach creates a more homogeneous architecture where all the models are coupled and managed by a unique lightweight script. The Listing 1 is an example of using the PyFMI library for a complete coupled simulation.

Listing 1: Python master script to perform a coupled simulation with the architecture c. in Fig. 2.

```
# Import PyFMI library
import pyfmi

# Load the FMUs
reactor = pyfmi.load_fmu("reactor.fmu")
BOP = pyfmi.load_fmu("BOP.fmu")

# Create connections between FMUs,
# from FMU outputs to FMU inputs
connections = [
    (reactor, "PowCore", BOP, "PowCore"),
    (BOP, "rhoExt", reactor, "rhoExt")
]
# Loop over the remaining FMI ports
# for the SG
for i in range(8):
    connections.append((
        reactor, f"PowSG{i}",
        BOP, f"PowSG{i}"
    ))
    connections.append((
        BOP, f"TempSG{i}",
        reactor, f"TempSG{i}"
    ))

# Create the simulator with models and
# connections
simulator = pyfmi.master.Master(
    [reactor, BOP],
    connections
)
opts = simulator.simulate_options()
opts["step_size"] = 0.01

# Run the coupled simulation
results = simulator.simulate(
    start_time = 0,
    final_time = 1000,
    options = opts
)
```

2.5. Restart of a simulation

The simulation restart is an important capability of any code system that allows, for example, performing transient simulations based on a steady-state solution from a previous calculation. A restart implies that the RAM has been emptied and that essential simulation data needs to be saved somewhere to enable a clean restart. Data can be saved to disk or within a master process. Unfortunately, while the FMI does not have specific limitations in terms of simulation restart, specific codes might have limited restart capabilities and specific restart procedures. This means the user must carefully handle each sub-model of its multi-fidelity (or multi-physics/multi-scale) model. In our specific case, while OpenFOAM includes straightforward restart capabilities, a similar option is missing from OpenModelica or Simulink.

To address this software-specific issue, we instructed the FMU Simulator or the Python script to write all the FMU state variables in a results file at each time step or at the final time step. Then, at the beginning of a restart, the latest FMU state in the results file is extracted, and each variable is initialized using the corresponding values. In the case of Simulink, not all the variables of an FMU are accessible. However, Simulink allows several entry points to initialize its state variables externally via FMI-based inputs.

We observe that we decided to write data to a file since it allows for a restart even when we stop the FMU Simulator or the Python script. However, when employing a Python-driven architecture (architectures c. and d. in Fig. 2), a developer can choose to keep the data in memory within the Python process.

3. Modeling of load-following operations of an LFR

To demonstrate the capabilities of the FMI standard, we have prepared a relatively complex multi-fidelity model of an LFR that couples a high-fidelity model of the primary circuit with lumped-parameter models of the balance of plant (BOP) and of a simplified control system. The LFR considered in this section is based on the design of the ALFRED reactor (Grasso et al., 2014).

3.1. Numerical models

In this section, we provide a brief description of the various models we used. A detailed description is out of the scope of this paper, but an interested reader can refer for instance to Radman et al. (2022) and Ponciroli et al. (2014).

3.1.1. GeN-Foam: primary circuit

The ALFRED reactor is a pool-type LFR developed in the frame of the 7th Euratom Framework Program (Grasso et al., 2014; Alemberti et al., 2020). This critical reactor of 300 MW thermal is cooled with liquid lead at atmospheric pressure, with a minimum temperature of 400 °C (673 K) corresponding to the inlet temperature of the core. The outlet temperature is 480 °C (753 K), leading to an average mass flow rate of ≈25 t/s. In this example, GeN-Foam is used to model the primary pool with a coarse-mesh 2-D geometry using the dimensions from Ref. Grasso et al. (2014). Fig. 3 shows the GeN-Foam model with the core, the primary pump, and the steam generator divided into 8 axial sections. The model makes use of a porous-medium (sub-channel-like) representation of the core and heat exchanger and of a standard RANS model for the pool. While a 2-D model is selected based on the demonstrative nature of this work, a 3-D model could be used without any change in the coupling logic. A porous-medium approach is used to model sub-scale structures such as nuclear fuel pins or steam generators, while we used a RANS CFD approach for clear-fluid regions. A point-kinetics approach is used to model the power evolution of the core with the nuclear feedback coefficients taken from Ref. Grasso et al. (2014).

3.1.2. OpenModelica and Simulink: BOP and control system

The secondary circuit is a superheated pressurized water/steam cycle, and it is modeled using OpenModelica and its ThermoPower library. It includes a steam generator, a high-pressure and two low-pressure steam turbines connected to the same shaft to an alternator, and an electric load acting as the electric grid. The pressure in the steam generator is ≈190 bars with a feedwater mass flow rate of ≈193 kg/s. The feedwater and steam generator outlet temperatures are 335 °C and 450 °C respectively. For cases a. and c. of Figs. 2 and 6, the OpenModelica model also includes a PID-controlled turbine admission valve and a PID-controlled reactor external reactivity (representative of control rods). The PID-controlled turbine admission valve maintains the alternator frequency to the targeted 50 Hz, corresponding to the standard European electrical grid frequency. For cases b. and d., Simulink replaces the PID controllers of OpenModelica as shown in Fig. 7, thus separating the BOP into two models shown in Figs. 4 and 5. In Fig. 4, the feedwater flows into the system through a pressure boundary condition. Then, it flows through the center of the steam generator bayonet, represented as a single pipe, and into the heating sections that receive power from the reactor model. After the steam generator, the steam passes the turbine valve admission which is controlled via one of the PIDs based on the electric grid frequency. Finally, the steam transfers its energy to the high- and low-pressure turbines which are connected to a mechanical inertia, the alternator, and the external electric load. The controllers shown in Fig. 5 use 2 separate PIDs. The first PID compares the electric grid frequency to a fixed target frequency of 50 Hz. The output is connected to the turbine valve admission opening, thus balancing the mechanical and electrical powers of the plant. The second PID compares the core power to the electrical power scaled by the nominal turbomachinery efficiency. The output is connected to the external reactivity insertion that is represented by a single number.

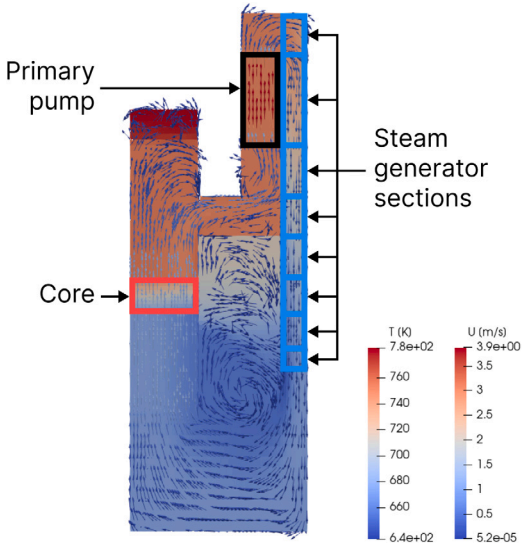


Fig. 3. Primary circuit modeled with GeN-Foam.

3.1.3. Coupling

The coupling between the reactor pool and the BOP is performed at the steam generator level based on the communication scheme presented in Figs. 6 and 7. The steam generator is divided into 8 axial sections, enabling us to partially discretize the axial distribution of the steam/water thermo-physical properties. This discretization can be seen in Figs. 3 and 4. Each section of the steam generator exchanges the power from GeN-Foam and temperature from OpenModelica via FMIs. In addition, the core power can be monitored by a PID that controls the insertion of reactivity in the core. This PID controller can be used to accelerate the convergence of the balance between the core power and the electric demand.

3.2. Results

A simplified load-follow operational transient has been selected as it involves all major nuclear power plant components. This transient is driven by the electricity demand on the alternator and by the response to this change of the PID that drives the turbine admission valve.

During this transient, it is expected that the core power will tend to balance the electric demand thanks to the predominately negative reactivity feedback of the reactor. For instance, if the power demand decreases, the alternator's electric output frequency will increase because of a decreased resistance to rotation. The PID will partly close the turbine admission valve to balance mechanical and electrical powers. The closing of the valve will induce an increase in pressure in the steam generator, which will increase the saturation temperature of the water. As the steam/water mixture heats up, the inlet temperature of the primary circuit will increase, which will decrease the core power thanks to the negative reactivity feedback. Hence, the core outlet temperature and the steam generator temperature will decrease, balancing the pressure increase in the steam generator. This transient is relatively

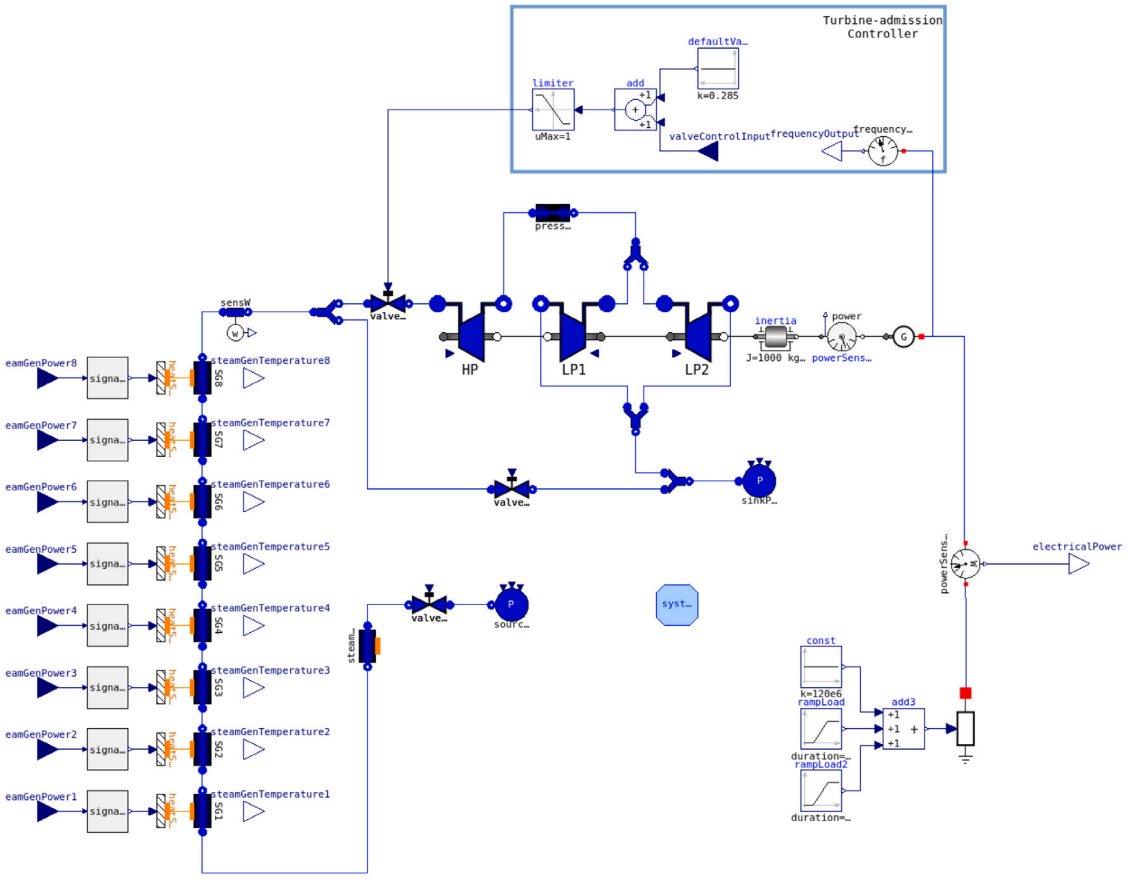


Fig. 4. OpenModelica model of balance of plant without controllers.

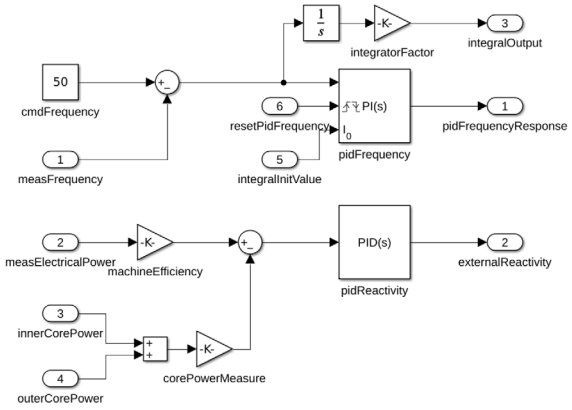


Fig. 5. Simulink model of PID controllers.

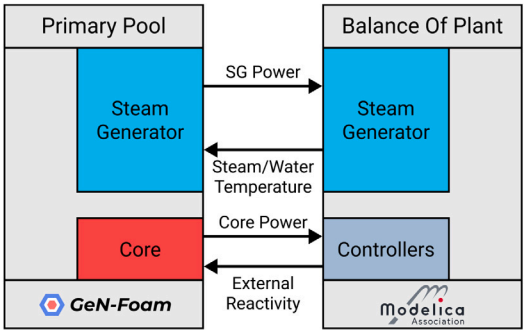


Fig. 6. Information transfers between the GeN-Foam and Modelica models through FMIs corresponding to architectures a. and c. of Fig. 2.

slow to converge to a new steady-state because of the large thermal inertia of the lead in the primary circuit and of the steam/water fluid in the steam generator. The most straightforward to accelerate the power convergence will be to anticipate the transient by adjusting the control rod position. Therefore, an additional PID controller has been included for demonstration purposes in a separate simulation to accelerate the core power convergence. The PID compares the electric load with a measure of the core power and acts on the control rod position to change the reactivity level in the core.

The scenario tested with the four architectures of Fig. 2 is a drop of 20 MWe of the electric power demand from the nominal power at 100 s. Then the demand returns to the starting value after 500 s. The demand variations happen linearly over 10 s. Two cases have been simulated and are reported in Fig. 8: one without any action on the core external reactivity (dashed lines); and one where a PID controller acts on the core external reactivity based on current and desired power levels (plain lines), which represents a more complex scenario where the core is driven towards the desired power output before the feedback from the secondary circuit propagates back to the core. For the first 100 s of the simulation, the system remained in a steady-state, allowing us to test our restart procedure. For brevity, we have only reported in Fig. 8 the results from case a. of Fig. 2. As expected, the other three simulated architectures showed identical behaviors.

In both the simulated cases, the frequency remains well under control with a maximum deviation below 0.2 Hz. In the Continental European power system context, a deviation of 0.2 Hz represents the maximum deviation before the full activation of the frequency containment reserves (Scherer, 2016). During the electric power demand

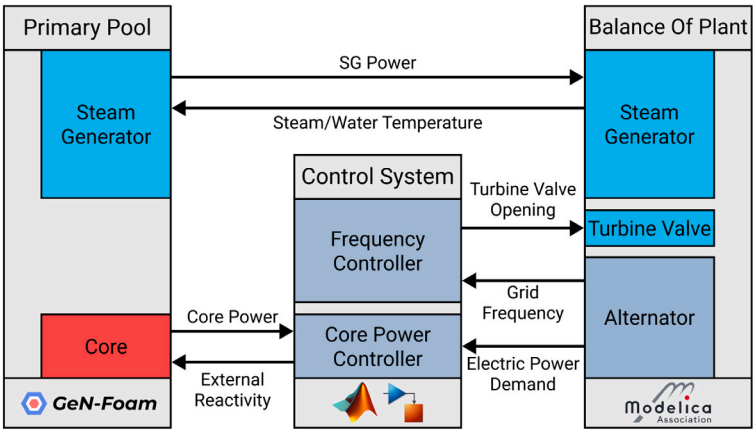


Fig. 7. Information transfers between the GeN-Foam, Modelica, and Simulink models through FMIs corresponding to architectures b. and d. of Fig. 2.

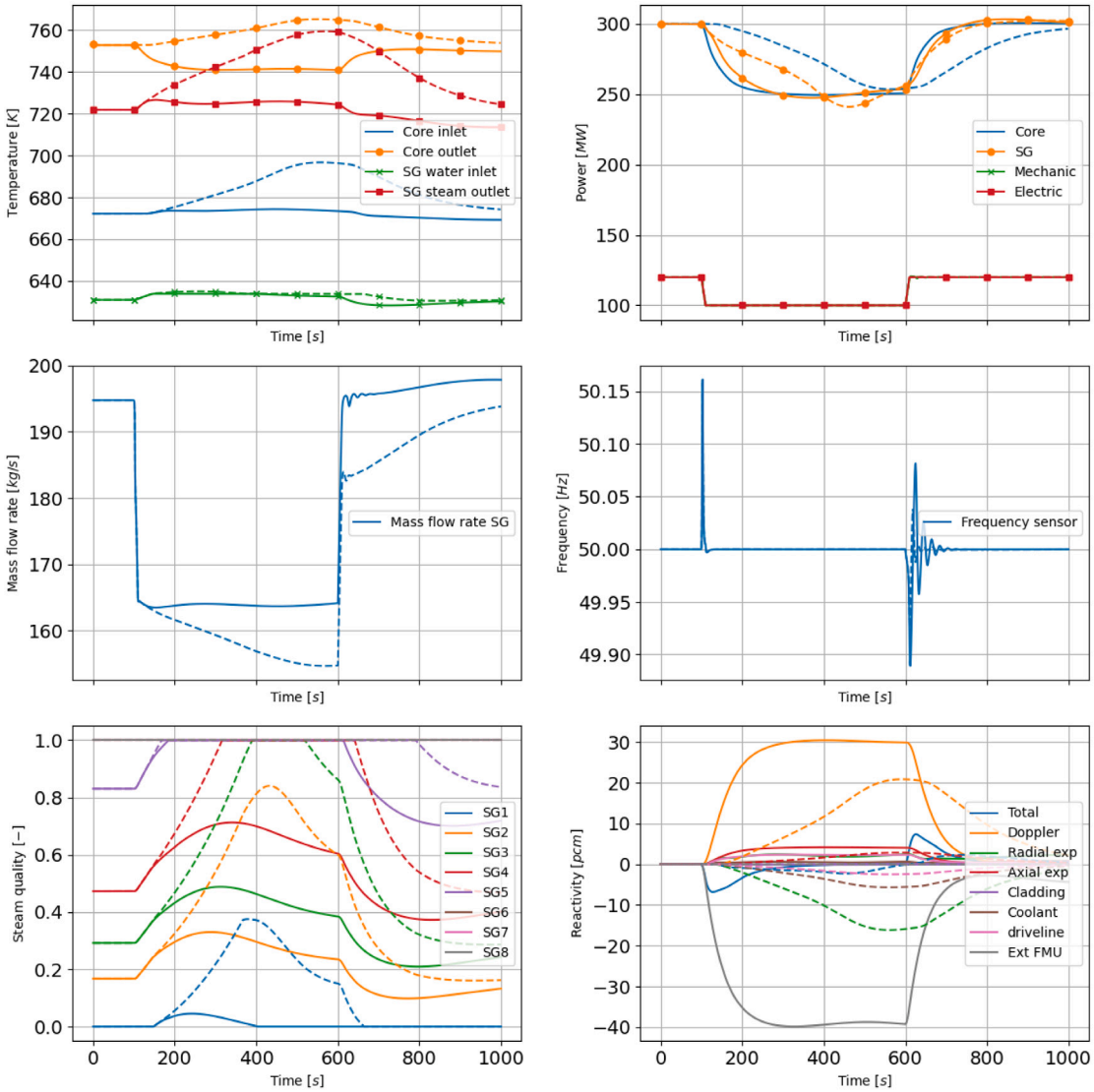


Fig. 8. Response of the system to a load-follow of 20 MWe. Plain lines are the transient response with a PID controller on the external reactivity, and the dashed lines are without the controller.

Table 1
Computational requirements during the 1000 s load-follow transient for different modeling architectures.

Modeling tools	Architecture	Computing time [s]	Ratio [%]
GeN-Foam standalone	–	588	–
GeN-Foam + Modelica (FMU)	Fig. 2.a	643	109
GeN-Foam + Modelica (FMU) + Simulink (FMU)	Fig. 2.b	734	124
GeN-Foam + Modelica (all FMUs)	Fig. 2.c	704	119
GeN-Foam + Modelica + Simulink (all FMUs)	Fig. 2.d	705	119

drop, the core power converges to a new lower power. As expected, the case with the PID controller acting on the core external reactivity converges faster to this new power. We also observe smaller temperature variations and a smaller steam production in the steam generator. It can also be noticed that the electric grid frequency response is asymmetric between the decrease and increase of power demand. In fact, the system behaves non-linearly depending on its state before each transient. Oscillations in frequency and mass flow rate at the point of load increase are thought to be physical since they did not change with different time steps, and since their frequency is significantly smaller than the time steps employed in the simulation.

Regarding computational requirements, Table 1 summarizes the computing time for a single-core run on an 11th Gen Intel® Core™ i9-11900K @ 3.50 GHz. The computational time seems to increase with the number of connected FMUs when using GeN-Foam as the master code, while this is not the case for an FMU-only simulation. We hypothesize that the FMU Simulator and the embedded Python interpreter may affect performance compared to the all-FMUs architecture. As a final comment about simulation results, we report that we have also run the simulation with a parallel GeN-Foam case and verified that parallelism, which is essential for higher-fidelity GeN-Foam cases, did not represent a problem.

4. Discussion and conclusions

This study has demonstrated the use of an FMI-based approach to simulate nuclear reactors, from core to alternator, based on a diverse set of open-source and proprietary tools like GeN-Foam, OpenModelica, and Simulink. Here are some major findings:

- The overhead of implementing an FMI-compatible interface for a new tool does not seem large with respect to implementing a dedicated interface for a specific code-to-code coupling. The additional time dedicated to understanding the FMI standard can be compensated by the availability of work done by the community. In the case of OpenFOAM, we found a library named FMU4FOAM that only needed to be extended to meet our needs. Looking at the library itself, it is clear that it benefitted greatly from the existence of FMU-oriented Python libraries. In essence, FMU4FOAM is configured as a refactoring of some OpenFOAM post-processing utilities, with the additional binding of Python code that allows the wrapping of FMI-oriented libraries. In general, we believe that the strategy of binding Python scripting within software can greatly simplify the task of providing codes with an FMI interface. It also enables seamless interaction with Python and its many libraries for scientific computing.
- Once an FMI-compatible interface and the possibility to wrap a tool into an FMU are implemented, coupling the tool (GeN-Foam in our case) with other FMI-compatible software has been confirmed to be straightforward. We tested two main approaches: GeN-Foam was the master code coupled to FMUs via an FMI-compatible interface and the other where all the models and software, including GeN-Foam, were embedded into FMUs and guided by a Python script. We have found that using a Python-driven all-FMUs architecture greatly simplifies the creation and extension of multi-fidelity or multi-physics models; it provides at all times a general view of the coupling architecture; it enables in-memory restarts; and it allows embedding pre- and post-processing capabilities based on Python libraries e.g. for uncertainty quantification, graphical user interfaces, etc. The drawback is that this approach requires the capability to embed a code into an FMU, which can be more challenging than simply providing a code with an FMI-compatible interface to FMUs.
- Coupling 3 different tools instead of 2 did not result in significant difficulties, underlining the potential of an FMI-based approach for a general, code-agnostic approach to code coupling. An all-FMU coupling strategy also showed no computational overheads when increasing the number of FMUs while maintaining the same complexity in the simulation.
- The issue of parallel performance has only been discussed very briefly. We simply verified that parallelization in GeN-Foam would not introduce problems. However, we believe that proper parallelization can become problematic with an increasing number of FMU objects, especially when two or more of these objects require high-performance computing. In those cases, a coupling architecture similar to cases c. and d. in Fig. 2, where a Python script drives the simulation, could be used for a suitable distribution of resources and set-up of an optimal calculation chain. However, this would make the Python script significantly more complicated with respect to what we reported in Listing 1.
- Simulation restart can represent a challenge in an FMU-based approach, depending on the variables that each code in the simulation environment exposes to the user. This is particularly a concern for proprietary tools. However, it seems reasonable to assume that most proprietary tools will/should expose enough variables to enable a user to perform a restart. This is what happened in our case.
- Regarding the specific case of coupling the primary circuit and BOP via energy conservation in the heat exchanger(s), we observed that an explicit coupling with a single information exchange at every time step was enough to provide a stable solution. However, implicit or semi-implicit couplings could easily be obtained by triggering an information exchange at every, or after several, iterations. Similarly, one could loosen the coupling by triggering the data exchange every several time steps.

Based on the above, implementing an FMI interface presents itself as a judicious and reasonable long-term investment of resources for simulation software. This approach offers the advantage of facilitating the coupling of multiple tools, a capability that would otherwise necessitate the development of multiple dedicated code-to-code-specific coupling interfaces. Furthermore, it is worth noting that numerous prominent codes utilized in the automotive and aeronautics industries already provide FMI adapters and that some nuclear-oriented codes are starting to move in the same direction. This widespread availability enhances the potential for coupling with an expanding array of tools from many engineering domains. By harnessing the power of the FMI standard, we firmly believe that our community can establish a robust and highly modular interdisciplinary modeling framework. This framework could serve as a cornerstone for conducting advanced multi-scale, multi-fidelity, and control-oriented simulations in the field of nuclear engineering.

A caveat of the FMI interface is that it is currently mainly oriented towards system-level, lumped-parameter tools. Although the FMI interface allows for the exchange of any amount of information, it does not provide standards e.g. for mesh-to-mesh projections. The demand for such an extension is increasing within the large FMI community, but the extension does not seem to be planned yet. Other projects exist that

are attempting to create a coupling standard for PDE-oriented tools. A notable example is the preCICE library (Chourdakis et al., 2022), which has a quickly expanding community and support for several major tools for scientific computation as an FMI runner. In this sense, it is a good plug-in for an FMI-based simulation environment.

Our future efforts will be directed towards enhancing the implementation of the FMI standard within GeN-Foam. This aims at ensuring the code's robustness while minimizing the number of dependencies. We also anticipate incorporating implicit coupling capabilities into GeN-Foam to enable stable simulations for specific code-coupling architecture, such as pressure-velocity coupling when simulating the thermal-hydraulics of a circuit with different codes. Additionally, we plan to move in the direction of multi-scale coupling by coupling GeN-Foam with the fuel performance code OFFBEAT (Scolaro et al., 2020) using the presented approach.

CRediT authorship contribution statement

Thomas Guilbaud: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Carlo Fiorina:** Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Software, Supervision, Validation, Writing – review & editing. **Stefano Lorenzi:** Methodology, Writing – review & editing. **Alessandro Scolaro:** Methodology, Project administration, Software, Supervision, Writing – review & editing. **Federico Carminati:** Writing – review & editing. **Donovan Maire:** Writing – review & editing. **Andreas Pautz:** Project administration, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The authors would like to acknowledge TRANSMUTEX for funding part of this study.

References

Alemberti, Alessandro, Caramello, Marco, Frignani, Michele, Grasso, Giacomo, Merli, Fabio, Morresi, Giulia, Tarantino, Mariano, 2020. ALFRED reactor coolant system design. Nucl. Eng. Des. 370, 110884.

Alfonsi, Andrea, Frick, Konor L., Rabiti, Cristian, Bragg-Sitton, Shannon, 2021. Status Report on the INL IES Plug-and-Play Framework. Tech. Rep. INL/EXT-21-63204, Idaho National Engineering Laboratory, Integrated Energy Systems.

Andersson, C., Åkesson, J., Führer, C., 2016. PyFMI: A Python Package for Simulation of Coupled Dynamic Models with the Functional Mock-up Interface. Tech. Rep., Technical Report in Mathematical Sciences LUTFNA-5008-2016, Centre for Mathematical Sciences, Lund University.

Angelucci, M., Martelli, D., Barone, G., Piazza, I. Di, Forgione, N., 2017. STH-CFD codes coupled calculations applied to HLM loop and pool systems. Sci. Technol. Nucl. Install. 2017, 1936894. http://dx.doi.org/10.1155/2017/1936894.

Bandini, G., et al., 2015. Assessment of systems codes and their coupling with CFD codes in thermal–hydraulic applications to innovative reactors. Nucl. Eng. Des. (ISSN: 0029-5493) 281, 22–38. http://dx.doi.org/10.1016/j.nucengdes.2014.11.003, URL: https://www.sciencedirect.com/science/article/pii/S0029549314005792.

Berry, R.A., Zou, L., Zhao, H., Zhang, H., Peterson, J.W., Martineau, R.C., Kadioglu, S.Y., Andrs, D., 2016. RELAP-7 Theory Manual. Tech. Rep., Idaho National Laboratory.

Chourdakis, G, Davis, K, Rodenberg, B, Schulte, M, Simonis, F, Uekermann, B, Abrams, G, Bungartz, HJ, Cheung Yau, L, Desai, I, Eder, K, Hertrich, R, Lindner, F, Rusch, A, Sashko, D, Schneider, D, Totounferoush, A, Volland, D, Vollmer, P, Koseomur, OZ, 2022. preCICE v2: a sustainable and user-friendly coupling library [version 2; peer review: 2 approved]. Open Res. Eur. 2 (51), http://dx.doi.org/10.12688/openreseurope.14445.2, URL: https://doi.org/10.12688/openreseurope.14445.2.

Dassault Systèmes, 2023. FMPy. https://github.com/CATIA-Systems/FMPy.

Fiorina, Carlo, 2022. The gen-foam multiphysics solver: A status update. In: International Conference on Physics of Reactors 2022 (PHYSOR 2022), Pittsburgh, PA.

Fiorina, Carlo, Clifford, Ivor, Aufiero, Manuele, Mikityuk, Konstantin, 2015. Gen-foam: A novel OpenFOAM® based multi-physics solver for 2D/3D transient analysis of nuclear reactors. Nucl. Eng. Des. 294, 24–37.

Fiorina, Carlo, Clifford, Ivor, Kelm, Stephan, Lorenzi, Stefano, 2022. On the development of multi-physics tools for nuclear reactor analysis based on OpenFOAM (R): state of the art, lessons learned and perspectives. Nucl. Eng. Des. 387, 111604. http://dx.doi.org/10.1016/j.nucengdes.2021.111604, URL: http://infoscience.epfl.ch/record/293938.

Fiorina, C., Habtemariam, N., Lorenzi, S., Scolaro, A., Guilbaud, T., Alfonsi, A., 2023. An FMI framework to enable multi-scale, multi-fidelity and control-oriented simulations of nuclear reactors using OpenFOAM/gen-foam and modelica. In: International Conference on Mathematics & Computational Methods Applied Nuclear Science and Engineering. Sheraton on the Falls, Niagara Falls, Canada.

Fletcher, C.D., Schultz, R.R., 1995. RELAP5/MOD3 Code Manual. Tech. Rep., Idaho National Engineering Laboratory.

2022. FMU4foam. https://pypi.org/project/FMU4FOAM/#description.

2022. Gen-foam. https://gitlab.com/foam-for-nuclear/GeN-Foam.

Grasso, G., Petrovich, C., Mattioli, D., Artioli, C., Sciora, P., Gugiu, D., Bandini, G., Bubelis, E., Mikityuk, K., 2014. The core design of ALFRED, a demonstrator for the European lead-cooled reactors. Nucl. Eng. Des. (ISSN: 0029-5493) 278, 287–301. http://dx.doi.org/10.1016/j.nucengdes.2014.07.032, URL: https://www.sciencedirect.com/science/article/pii/S0029549314004361.

Gurecky, William, de Wet, Dane, Greenwood, Michael Scott, Salko, Jr., Robert, Pointer, Dave, 2020. Tech. Rep. ORNL/TM-2020/1872, Oak Ridge National Lab. (ORNL), http://dx.doi.org/10.2172/1763465, URL: https://www.osti.gov/biblio/1763465.

Herb, Joachim, 2014. Coupling OpenFOAM with thermo-hydraulic simulation code ATHLET. In: 9th OpenFOAM Workshop. Zagreb, Croatia.

Hu, Rui, Zou, Ling, Hu, Guojun, Nunez, Daniel, Mui, Travis, Fe, Tingzhou, 2021. SAM Theory Manual. Tech. Rep., Nuclear Science and Engineering Division, Argonne National Laboratory.

Jeltsov, Marti, Kööp, Kaspar, Kudinov, Pavel, Villanueva, Walter, 2013. Development of a domain overlapping coupling methodology for STH/CFD analysis of heavy liquid metal thermal-hydraulics. In: Proc. 15th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-15). KTH, Nuclear Power Safety, URL: http://www.nureth15.org/. QC 20180523.

Lindsay, Alexander D., Gaston, Derek R., Permann, Cody J., Miller, Jason M., Andrš, David, Slaughter, Andrew E., Kong, Fande, Hansel, Joshua, Carlsen, Robert W., Icenhour, Casey, Harbour, Logan, Giudicelli, Guillaume L., Stogner, Roy H., German, Peter, Badger, Jacob, Biswas, Sudipta, Chapuis, Leora, Green, Christopher, Hales, Jason, Hu, Tianchen, Jiang, Wen, Jung, Yeon Sang, Matthews, Christopher, Miao, Yinbin, Novak, April, Peterson, John W., Prince, Zachary M., Rovinelli, Andrea, Schunert, Sebastian, Schwen, Daniel, Spencer, Benjamin W., Veeraraghavan, Swetha, Recuero, Antonio, Yushu, Dewen, Wang, Yaqi, Wilkins, Andy, Wong, Christopher, 2022. 2.0 - MOOSE: Enabling massively parallel multiphysics simulation. SoftwareX (ISSN: 2352-7110) 20, 101202. http://dx.doi.org/10.1016/j.softx.2022.101202, URL: https://www.sciencedirect.com/science/article/pii/S2352711022001200.

Modelica Association, 2022. Functional mock-up interface for model exchange and co-simulation. https://github.com/modelica/fmi-standard/releases/download/v2.0.1/FMI-Specification-2.0.1.pdf.

Ochel, Lennart, Braun, Robert, Thiele, Bernhard, Asghar, Adeel, Rogovchenko-Buffoni, Lena, Eek, Magnus, Fritzson, Peter, Fritzson, Dag, Horkeby, Sune, Hällqvist, Robert, Kinnander, Å ke, Palanisamy, Arunkumar, Pop, Adrian, Sjölund, Martin, 2019. OMSimulator - integrated FMI and TLM-based co-simulation with composite model editing and SSP. pp. 69–78. http://dx.doi.org/10.3384/ecp1915769.

2022. OpenModelica. https://www.openmodelica.org/. [Online; accessed 25-July-2022].

Ponciroli, Roberto, Cammi, Antonio, Lorenzi, Stefano, Luzzi, Lelio, 2014. A preliminary approach to the ALFRED reactor control strategy. Prog. Nucl. Energy (ISSN: 0149-1970) 73, 113–128. http://dx.doi.org/10.1016/j.pnucene.2014.01.016, URL: https://www.sciencedirect.com/science/article/pii/S0149197014000286.

Radman, Stefan, Fiorina, Carlo, Song, Ping, Pautz, Andreas, 2022. Development of a point-kinetics model in OpenFOAM, integration in gen-foam, and validation against FFTF experimental data. Ann. Nucl. Energy (ISSN: 0306-4549) 168, 108891. http://dx.doi.org/10.1016/j.anucene.2021.108891, URL: https://www.sciencedirect.com/science/article/pii/S0306454921007684.

Scherer, Marc, 2016. Frequency Control in the European Power System Considering the Organisational Structure and Division of Responsibilities (Ph.D. thesis). ETH Zurich, http://dx.doi.org/10.3929/ethz-a-010692129.

Scolaro, Alessandro, Clifford, Ivor, Fiorina, Carlo, Pautz, Andreas, 2020. The OFFBEAT multi-dimensional fuel behavior solver. Nucl. Eng. Des. 358, http://dx.doi.org/10.1016/j.nucengdes.2019.110416.

Simulink Documentation, 2020. Simulation and model-based design. URL: https://www.mathworks.com/products/simulink.html.

2022. ThermoPower. https://github.com/casella/ThermoPower.

U.S. Nuclear Regulatory Commission, 2008. TRACE V5.0 User's Manual - Volume 2: Modeling Guidelines. Tech. Rep., U.S. NRC.