CFD and Neutron codes coupling on a computational platform

Daniele CERRONI, Roberto DA VIÀ, Sandro MANSERVISI, Filippo MENGHINI and Ruben SCARDOVELLI
University of Bologna, DIN, Lab. of Montecuccolino, Via dei colli 16, 40126 Bologna (BO), Italy
E-mail: daniele.cerroni2@unibo.it

Abstract. In this work we investigate the thermal-hydraulics behavior of a PWR nuclear reactor core, evaluating the power generation distribution taking into account the local temperature field. The temperature field, evaluated using a self-developed CFD module, is exchanged with a neutron code, DONJON-DRAGON, which updates the macroscopic cross sections and evaluates the new neutron flux. From the updated neutron flux the new peak factor is evaluated and the new temperature field is computed. The exchange of data between the two codes is obtained thanks to their inclusion into the computational platform SALOME, an open-source tools developed by the collaborative project NURESAFE. The numerical libraries MEDmem, included into the SALOME platform, are used in this work, for the projection of computational fields from one problem to another. The two problems are driven by a common supervisor that can access to the computational fields of both systems, in every time step, the temperature field, is extracted from the CFD problem and set into the neutron problem. After this iteration the new power peak factor is projected back into the CFD problem and the new time step can be computed. Several computational examples, where both neutron and thermal-hydraulics quantities are parametrized, are finally reported in this work.

1. Introduction
The numerical simulation of a nuclear reactor system is a challenging multiphysics problem. One of most important region in these systems, is the active zone of the core where the neutrons interacts with the fissile materials and the coolant flows removing the heat produced by the nuclear chain reaction [1]. During years lot of codes have been developed in order to study the nuclear interaction and the thermal hydraulics evolution of these systems. The first types, neutron codes, usually investigate the evolution of the neutron population at different energy levels considering all the different materials present in the core region [2]. The material properties that determine the interaction with the neutron flux strongly depend on the local temperature field. In order to produce an accurate evaluation of the energy production, the temperature dependencies of the materials properties must be taken into account by the neutron code. The temperature evolution in the core region is usually investigated with thermal hydraulics codes, that solve the energy and momentum balance equation in the core region under the assumption of a constant energy production rate. The mutual interaction between the neutron flux and the local temperature field is a fundamental aspect in the study of a reactor core but, at the same time, a very challenging phenomena to investigate [3]. In this work we couple, using a
numerical platform, the solution of a neutron code with the solution of a CFD code in order to investigate the thermal-hydraulics dynamics of a PWR nuclear reactor evaluating the power generation distribution taking into account the local temperature field. The CFD code that we use is the in-house multigrid finite element code FEMLCORE that has been used and validated for the thermal-hydraulic simulation of a reactor core in [4–6]. The neutron code used in this work is DONJON-DRAGON, an open-source framework developed by the Institute of Nuclear Engineering of the École Polytechnique de Montréal that has been integrated into the numerical platform SALOME. The software is composed by two modules: DRAGON and DONJON. The DRAGON code [7] results from a concerted effort to unify inside a single code various numerical techniques and calculation methods that can be used to solve the neutron transport equation both with the collision probability technique and the method of characteristics. The standard calculation procedure we carry is in two steps. In the first one we perform critical transport calculations on the transport model and generate a consistent set of multigroup properties such as the various cross sections and diffusion coefficients for each different material. In the second step we introduce these nuclear properties in the DONJON full core model and compute the macroscopic flux distribution and the multiplication factor of the core. The DRAGON generated macroscopic properties are stored in COMPO files. The coupling between the neutron and the CFD code is achieved through their inclusion into the numerical platform SALOME, a software developed by the CEA for the design of light water reactor. In this platform all the included codes can exchange data run-time thanks to the MEDmem libraries, a computational toolbox that handles the exchange of data between the codes. During the execution of the codes, the temperature field in the core region, evaluated using the FEMLCORE module, is exchanged, through a MEDmem interface, with the DONJON code. The material macroscopic cross sections are then updated according to the new temperature field using the COMPO object previously generated with the code DRAGON and, the consequent neutron flux, is computed. From the updated neutron flux the new peak factor is evaluated and projected, again through the MEDmem interface, into the FEMLCORE problem that evaluates the new temperature field. The work is divided as follows. After a general description of the SALOME platform structure we describe the coupling procedure of a generic code into the platform. In particular we introduce the principal elements that must be developed in order to exchange data with another code that has been previously integrated into this platform. In the following Section a computational example is presented and the results are analyzed. Finally in the last Section we give some conclusion and we draw lines for future works.

2. Numerical Platform
The collaborative project NURESAFE for the development of reliable software usable for safety nuclear reactor analysis, has been funded by the European community based on the open-source SALOME platform. NURESAFE has been created to improve the nuclear safety by developing high level of expertise and collecting the most recent simulation tools in the nuclear field [4,8]. At the moment the platform is based on computational tools for light water reactors but many of these codes can be adapted to a large range of applications. SALOME integrates a large number of modules each having its own function: KERNEL, GUI, GEOM, SMESH, MED, YACS and Paravis module. Above all the computational tools included into platform, the MED module provides a library for storing and recovering computer data in MED format, associating numerical meshes and fields allowing the exchange between different codes and solvers. Inside the platform these structures are exchanged between solvers at the communication level (CORBA or MPI) offering common read and write functions through MED files. The MED libraries are divided into three groups of functions: MED File, MED Memory and MED CORBA. The first group (MED File) is a C and Fortran api that implements mesh and read/write functions into files with med extension, these files are in HDF5 format: a data model for storing and managing
data. The module supports an unlimited variety of data types, it is designed for flexible and efficient input/output and for handle complex data. The MED Memory group, which is a C++ and Python api, creates mesh and field objects in memory, the mesh creation can be done using set functions, or by loading a file. Fields are also created loading files or by initialization with user-defined functions. Finally the group of functions called MED CORBA

2.1. Data exchange.
This platform has been conceived not only to collect a series of codes that have been extensively used in the field of thermal hydraulics of nuclear reactors but also to harmonize them with the aim of solving complex problems by exchanging information among different codes over a common platform and on large multiprocessor architectures. SALOME can also be used as a platform for the integration of external third-party numerical codes to produce a new application with full pre and post processing management of CAD models. The integration of a code on the SALOME platform is obtained by generating an interface with functions available in the MEDMem library that allows the data transfer from the platform to the code and vice versa. Two different codes both with SALOME MEDMem interface can transfer data to the interface and then from the interface to the other code. MED supports different element shapes such as point, line, triangle, quadrangle, tetrahedron, pyramid, hexahedron, polygon and polyhedron. Each element has a different number of nodes, depending on linear or quadratic interpolation. In order to have a working platform, common input and output formats should be harmonized between different codes. This can be achieved by using SALOME as the basic platform taking care of the data exchange between codes and of the distributed computation between different clusters.

In the following part of the Section we describe the integration procedure of a CFD code into the SALOME platform with the aim of using computational modules generated with this library for multiphysics coupling. The integration of an open-source code into the a numerical platform, can be divided into three major steps: the first is the generation of the code-library from the original code, the second is the generation of the MEDMem interface and finally the generation of SALOME-code interface integration. In a non open-source code, the source code is not available but usually the binary version of the library is provided, in this case the first step of generation of the code-library is no longer needed.

2.2. Generation of the code-library
The generation of the dynamic library from a code is pretty straightforward for modern codes since they are already built as libraries. The main code is simply a collection of call functions to libraries where the algorithms are developed. For old codes, especially in Fortran, sometimes a monolithic main program is developed with a few functions in support with experimental data. In this case it should be straightforward, for developers, to rebuild the code using a library structure.

2.3. MEDMem interface
Simulation studies require the manipulation of meshes and fields for data processing or post processing. Corresponding computer codes can be viewed as software components accessing input meshes and fields, with specific constraints, along with parameters and producing output meshes and fields. The MED module aims at pooling operations on those items, facilitating their use by various codes involved in a simulation process. This includes making codes communicate while preserving as much as possible the integrity of their content. In order to fulfill its objective, the MED module includes methods for: handling meshes and fields to satisfy code input requirements, extraction of field information and projections and serialization to exchange meshes and fields between codes. The structure of the MEDMem libraries is shown in the
left part Figure 1, the fundamental set (blue background) consists in three atomic libraries: MEDCoupling that describes data structures used for cross process exchange of meshes and fields, MEDLoader and ParaMEDLoader that provides input output functions to the med file format and interpolation tools that provides mathematical structures and algorithms for interpolation and localization. The main services offered by MEDCoupling are the manipulation of fields and their support mesh and multidimensional interpolation on nodes, cells, Gauss points and nodes by element. Once the code has been build as a library we need to add methods that can export the results into the MED format, in this way, in order to exchange a computational field with another code, we can just extract the solution from the MED file and project it into the different computational domain still in MED format. For this purpose we build a duplicate of the original computational grid in the med format and create a map that associates each computational node from the original mesh to one of the MED duplicate and vice-versa as shown in the right part of Figure 1. The generated map can be used both for projecting a solution coming from the specific code into the MED support but also for transfer a computational field from the MED mesh to the code specific computational domain. The projection into the MED grid consist on a reordering of the solution according to the generated map. When the solution is prepared as just described, it can be attached to the med support and transferred to any other med support using the methods of the med api. Once the field is transferred into a different support it can be extracted and ordered according to the particular map generated for the second program.

2.4. SALOME interface

The final step in the integration of a generic code into the numerical platform, is the construction of different methods that allow the control of the execution and the data input/output of the selected code. There are basically five classes that must be introduced in the selected code: driver, problem, equations, grids and methods. The driver class is the top level class that interface with the SALOME platform as it is shown in the collaboration diagram in Figure 2, where the basic structure of the code that we want to integrate into the platform is illustrated. This diagram show how the driver class transfers data to the problem class which is specialized for different physics. This class is inherited by all the physical modules used by the selected code. The equations class, which uses only MEDMem functions, inherits the system particular class which contains the assembly and solver of the generic code. The data from the driver class can be transferred into the assembly routine which is user accessible. The data should also be transferred in the opposite direction from the equations to the MEDMem interface. The data flow from parent to son class is ensured by the C++ inheritance rules while the flow in opposite direction is defined by a dynamic_cast command that allows to use son class functions from the parent class. For this reason the parent class should be polymorphic with at least one virtual function. The grids class is an extension of mesh class of the particular CFD code, it contains
all the routines for mesh handling together with MEDMem methods that are used to extract and manage groups of nodes and cells from the original computational grid. In particular the extension class contains the methods that allow the data transfer from one format to another. As in the previous class, data from the driver class can be transferred, by using a dynamic_cast statement, into the assembly routine which is user accessible. Data can also be transferred in the opposite directions from the grids to the MEDMem interface through parent to son class inheritance rules. All the code interfaces must have similar commands to run the program. The mesh boundary names and their flags, used for creating mesh groups, are stored in the driver interface class. Over these groups, by using MEDMem functions and the map between the med support and the specific mesh, we are able to extract field from the med support and project it from the MED support, coming from other SALOME platform codes, into the problem specific mesh format. The basics commands that the problem class must have are: setType, setMesh and solve. The first command sets the problem type (Navies-Stokes, energy, etc.), the second one sets and prepares the mesh which should be available in MED and code specific formats for data exchange and, finally, the solve command controls the solution of the discrete system. The boundary regions of the domain should be controlled for input and output. For this purpose we have to introduce class methods that could set and get analytic and numerical solutions in these particular zones. The interface to the selected code is obtained through the problem class. For this reason it should access to both grids and equations classes as shown in the collaboration diagram of the problem class in Figure 2.

While the grids class can only handle the data the specific code format, the equations class contains two standard maps that associate to any zone of the computational grid a methods so that a volumetric field can be extracted and transferred to a methods with a MEDMem mesh format by using the getSource and the setSource function. If the data transfer is between 3D and 1D the average source functions may be used. When codes are organized as just described exchange data is easy. We just have to create a supervisor that access the driver class from different problems and, after each solving iteration the desired field is extracted from the first driver and projected into the common format. This field is then projected by the second driver into the computational grid of the second problem and can be taken into account during its execution.

3. Numerical Results
In this Section we show some numerical results obtained coupling the neutron code DONJON with the FEM code FEMLCORE. In the example we investigate the temperature field, the velocity profile and the neutron flux evolution in the active core region of a PWR reactor. The codes are able to exchange data through a MED interface, in particular during a time step iteration firstly we evaluate the temperature and the velocity field according to a given distribution of energy production. Those fields are then accessed by the neutron code that update the material properties according to the new temperature field and a new neutron flux distribution is evaluated. The new energy distribution is the taken into account for the evaluation.
of the temperature and velocity field in the next time step iteration. In the following part of the Section we give a general description of the reactor core considered and the some results are shown under different working condition.

3.1. General Description

The geometry of the reactor used in this work, is shown in the left part of Figure 3, we consider an annular reactor characterized by a diameter \( d \) of 3.23m and an height \( a \) of 6.6m. The active part of the core start form \( c = 2.5m \) and ends at \( b = 4.5m \). The coolant (water) flows from the lower inlet region to the outlet on the top. The core that is composed of 221 fuel elements, each of which is made up of 428 fuel pins. The discretization of such complex domain with a realistic geometry would require a number of computational cells that is challenging even on huge clusters and supercomputers. It is therefore mandatory to approximate the velocity, pressure and temperature fields inside the core on a coarser grid of computational cells that somehow homogenize the underlying structure of pins and subchannels. We can consider a multiscale approach, where a fine level and a coarse level are considered. The fine level takes into account for the real geometry of the core. The solution on this level is extended to the whole volume of the coarse cell and projected on the degrees of freedom (dofs) of the coarse cell. At this level all geometrical details pertinent to the single subchannels are ignored, while a mean value of the physical fields are computed like in a porous medium, with physical properties taken as mean values of the coolant and structural materials. We denote the coarse grid fields with a \( \hat{\cdot} \) sign. We therefore define a porosity coefficient \( r = A_C/A_{FA} \) where \( A_C \) is the coolant flow area, while

\[
\begin{align*}
\nabla \cdot (r \rho \hat{v}) &= 0 \\
\frac{\partial (r \rho \hat{v})}{\partial t} + \nabla \cdot (r \rho \hat{v} \hat{v}) &= -\nabla \hat{p} + \nabla \cdot \hat{\tau} + r \rho \hat{g} + \hat{m}_g \\
\frac{\partial (r \rho c_p \hat{T})}{\partial t} + \nabla \cdot (r \rho c_p \hat{T} \hat{v}) &= -\nabla \cdot (r \kappa \nabla \hat{T}) - \hat{q}_g.
\end{align*}
\]

Figure 3. Geometry of the reactor on the left. DONJON computational grid and active zone of reactor core on the center and right part, respectively.

\( A_{FA} \) is the entire section of the fuel assembly. The conservation equations can be written as
The surface subgrid momentum and heat sources $\hat{m}_g$ and $\hat{q}_g$ are independent of $r$. The momentum source $\hat{m}_g$ can be modeled by

$$\hat{m}_g = -\beta \cdot \frac{\rho \vec{v} \cdot \vec{v}}{2D_{eq}}. \quad (2)$$

This term in the momentum equation represents the loss of pressure in the coolant due to the viscous friction in the subchannels. The vector $\beta$ takes into account the direction of the flow. For pure vertical flow the term is set to the standard pressure losses for a standard channel geometry. For cross flow the term should be evaluated case by case. In (1), as a first approximation, $r$ can be assumed constant. In this case even if system (1) is formally very similar to the standard balance equation system but there are large differences hidden in the definition of the stress tensor $\tau$ and of the thermal conductivity $\kappa$. The term $\tau$ can be split in three components: the standard viscous part $\tau^{\text{visc}}$, the turbulent part $\tau^{\text{T}}$ and an additional term $\tau^{\text{eff}}$ that comes from the subgrid (fine) level. The evaluation of $\tau^{\text{eff}}$ is a challenging task. It may be computed with a DNS simulation of a subchannel, or can be obtained by experimental data. In the case of lack of data the term could be modeled with simple algebraic standard relations or ignored. Similarly, the thermal conductivity $\kappa$ can be split in $\kappa^{\text{visc}}, \kappa^{\text{T}}$ and $\kappa^{\text{eff}}$. The 3D-porous model can simulate naturally an open assembly core but also a closed or mixed open-closed assembly core simply by enforcing no cross flow through the assemblies. For details about this model the interested reader can see [4, 5, 9].

The input parameters, used by DRAGON, for the generation of the macroscopic cross section library needed for solving the multigroup diffusion equation, and the particular fuel pin geometry are reported in the left and right part of Figure 4, respectively. The transport equation is integrated over a lot of tiny energy interval and the results are then condensed into the energy groups considered into the multigroup diffusion model. The macroscopic cross section are evaluated for different fuel compositions and for different average temperatures according to a range defined by the user. This operation allows the interpolation of the macroscopic cross section between the range values that has been set. We remark that this first step, the generation of macroscopic cross section library, depends only on the fuel pin composition and geometry, the possibility to evaluate the cross section in a range of composition and temperature is meant for avoiding the generation of this library every time iteration during a simulation. Since DONJON, the code that solves the multigroup diffusion problem, can handle only Cartesian structured mesh we have to embed the computational domain shown in Figure 3 into a Cartesian three-dimensional grid specified by a characteristic length $u$. In our case we choose $u = 0.215m$ and the computational domain is show in central and right part of Figure 3, in particular in the right part of the same Figure we mark the active core region in the domain, in this zone the neutron flux and the peak factor are evaluated considering

![Figure 4](image_url)

**Figure 4.** Fuel pin geometry on the right and composition parameters on the left.
the particular composition of every fuel element of the computational grid. In particular the domain is composed by 31 sections made of $17 \times 17$ cells and the composition of each cell is specified in a $17 \times 17$ matrix. In this example we consider three material types: 0 is a dummy material that mark a zone in which the neutron flux is zero and must not be evaluated, 1 is the fuel material which has the composition shown in the left part of Figure 4 and finally 2 is the reflector material. In the right part of Figure 4 a sketch of the fuel pin element is reported, from that Figure one can appreciate the geometrical complexity that can be considered in the evaluation of the macroscopic cross sections. The firsts and the lasts 10 planes of the neutron computational grid, has the $A$ configuration show in Figure 5, the $11^{th}$ and $22^{th}$ planes has the $B$ configuration show in Figure 5 and finally the remaining planes, the active core, has the $C$ configuration shown in Figure 5. Now that we have specified the composition of all the elements of the computational grid we can solve the two groups diffusion equation evaluating the neutron flux inside the domain. In the next Section we describe the interface operator that is used for the projection of the computational fields in the different computational grids.

![Figure 5](image)

**Figure 5.** Three possible mix (A, B and C) of materials 1, 2 and 3 in a reactor plane.

3.2. Interface

![Figure 6](image)

**Figure 6.** FEMLCORE (solid color) and DONJON (outline box) meshes. Whole meshes on the left and heated zone on the right.

The interface between the neutron and CFD module is created by the construction of a duplicated mesh of the two computational domains in the MEDmem format and creating a map between them. In this case the two meshes have different geometry as we can see in Figure 6. In particular in the left part of that Figure the two computational domains are shown, the bounding box mark the Cartesian structured mesh mesh and the intern grid is the CFD mesh. In the right part of Figure 6 we can see the overlapping between the active core zone in the Cartesian mesh and the CFD domain. The map between the two MEDmem meshes couples all the elements of the CFD computational grid in the nearest cell into the Cartesian grid.
After a solve iteration of the CFD problem the temperature field is taken from the problem and projected, through the MEDmem meshes duplicates and the map operator, to the neutron code. The neutron flux is then calculated by DONJON updating the cross section according to the new temperature field. From the updated neutron flux the new peak factor is evaluated and projected back to the CFD code which can evaluate the new temperature field.

3.3. Core evolution

The temperature field in the initial condition is set to a constant value of $500^\circ C$ while the axial velocity field to a constant value of $1m/s$. The velocity field remains equal to the initial field during the whole simulation because of the boundary condition and the geometry of the problem. For these reasons it is not reported. Due to the energy production in the active zone of the core, the temperature increases from the initial value until the steady state is reached. The temperature field at different time steps $t_1$, $t_2$ and $t_3$ (0.1, 2.5 and 10s, respectively) over the 3D domain is shown in Figure 7. We can notice that the temperature field increases in the active zone of the reactor in the initial time steps ($t_1$ and $t_2$) and then, because of the velocity field, all the upper zone is heated until the steady state is reached ($t_3$). In Figure 8 the peak factor (on the left) and the temperature field (on the right) over the center line of the domain at different time steps ($t_1$, $t_2$ and $t_3$) is shown. We can notice that as the temperature field increase the peak factor changes because of the different cross section. This effect cannot be considered without the usage of the neutron code.

4. Conclusion
In this work we have shown a general procedure for the integration of numerical codes into a computational platform and we have show how this computational tool can be used to investigate
the energy production inside the core region of a nuclear reactor by taking into account the local temperature field. The solution of this complex problem has been achieved by coupling the solution of an in-house finite element code that solves the balance system of equations, with a neutron code that evaluate the energy production in the core region. At every time iteration the material properties, used to evaluate the neutron flux distribution, have been evaluated according to the local temperature field. The exchange of data between the different codes has been obtained through the numerical libraries MEDMem included into the SALOME platform where both of the codes have been included. The numerical results have shown the consistency and the robustness of the coupling algorithm. In future works we would like to integrate into the platform a system code such as Cathare and couple the solution of the multiphysics problem presented in this work, with the simplified model of the whole primary loop of the reactor. An attractive characteristic of the coupling technique shown in this work, is that the execution and the interface of every module included into the platform is independent. This modular approach allows to easily increase the number of codes that can exchange data run-time and, moreover, the substitution of one of this module with any other code already included into the platform.

References
[1] James J Duderstadt L J H 1976 Nuclear Reactor Analysis ISBN 0471223638,9780471223634
[2] Lamarsh J R 1983 Introduction to nuclear engineering 2nd ed Addison-Wesley series in nuclear science and engineering (Addison-Wesley) ISBN 9780201142006,0201142007
[3] Giannetti F, Maio D V D, Naviglio A and Caruso G 2016 Nuclear Engineering and Design 305 168–178 ISSN 0029-5493
[4] Bornia G, Finelli M and Manservisi S 2011 Development and validation of fem-core code for the thermal hydraulics of open cores Tech. rep. Report RdS-2011-002
[5] Bornia G, Cerroni D, Manservisi S, Polidori M and Donato F 2012
[6] Olander D R 1976 Fundamental aspects of nuclear reactor fuel elements Tech. rep. California Univ., Berkeley (USA). Dept. of Nuclear Engineering
[7] Marleau G, Roy R and Hébert A 1994 Report IGE-157, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec
[8] Chanaron B, Ahnert C, Crouzet N, Sanchez V, Kolev N, Marchand O, Kliem S and Papukchiev A 2015 Annals of Nuclear Energy 84 166–177
[9] Cervone A and Manservisi M 2008 A three-dimensional cfd program for the simulation of the thermo-hydraulic behaviour of an open core liquid metal reactor Tech. rep. Report RSE/2009/85