# DRAGON5 and DONJON5, the contribution of École Polytechnique de Montréal to the SALOME platform

Alain Hébert[1]

[1] *École Polytechnique de Montréal, Institut de Génie Nucléaire,*
*P. O. Box 6079, Station "Centre-Ville", Montréal, Q. C., Canada,*
*Email: alain.hebert@polymtl.ca*

**Abstract.** The DRAGON lattice code and DONJON full core simulation code are in active development since 1990. A lattice code is used to solve the neutron/photon transport equation using a deterministic approach over a domain corresponding to a subpart or over a complete 2D or 3D nuclear reactor. A full core simulation code includes steady state and transient neutron flux solution capabilities together with reactor operation, micro depletion, and simplified thermal-hydraulics models. The DRAGON and DONJON codes are designed around a scripting language, known as CLE-2000, to automate the flow of information used in computational schemes

DRAGON5 and DONJON5 are the latest version of these codes and features characteristics that are required to design advanced nuclear reactors. First, a new kernel, named GANLIB5, was implemented entirely in ANSI C to facilitate the coupling of the code in multi-physics environments and to permit the embedding in control systems. Second, we added new geometric capabilities to permit the representation of complex geometries that are typical of advanced applications. The first release of DRAGON5 and DONJON5 was available in October 2013. Subsequent releases are permitting couplings with the geometry module (*Geometry*) and supervisor module (YACS) of the SALOME platform.

**Keywords:** Lattice code, Neutron transport, Deterministic solution, Computational scheme, Computer-aided design.

## INTRODUCTION

A lattice code is the main software component for describing the behavior of neutrons in a nuclear reactor.[1] Most important phenomena are precisely described, including effects from material temperature and density, resonance self-shielding, fuel burnup, neutron transport and leakage, etc. These effects are described using a deterministic numerical approach. Traditionally, the lattice code is applied over a spatial domain called a *unit cell*, a small 2D region of the reactor assumed to repeat itself by symmetry or translation. In a production pressurized water reactor (PWR), the unit cell is a fuel assembly or a colorset of four fuel assemblies. A full core simulation code is a collection of components, related to reactor physics, and dedicated to the simulation of fuel management, reactor operation or accident scenarios. The VERSION5 distribution in an open-source framework including the lattice code DRAGON5 and the full core simulation code DONJON5.[2,3] This distribution is the latest release of the reactor physics codes developed at École Polytechnique de Montréal (ÉPM). One of the original characteristics of this fifth distribution is its inherent capability to be integrated to the open-source SALOME platform.

The SALOME platform provides the required computer-aided design (CAD) capabilities, including geometry treatment and surface analysis. The SALOME platform also includes supervision and visualization capabilities.[4] It is in active development since 2001 by three sponsors: the Commissariat à l'Énergie Atomique et aux Énergies Alternatives (CEA), Électricité de France (EDF) and EURIWARE/Open Cascade.

SALOME and VERSION5 offer a powerful ecosystem for designing advanced reactors, starting from a CAD-oriented description of the geometry and up to power map calculations or multi-physics applications. The actual implementation is limited to 2D geometries, but the approach could be extended to general 3D cases.

The SALOME distribution, represented by the central circle in Figure 1, can be extended by application codes, specific to each discipline of nuclear engineering. These discipline are related to reactor physics (APOLLO3, DRAGON5, DONJON5, COCAGNE), thermal-hydraulics (THYC), fuel thermo-mechanics (ALCIONE, CYRANO3), etc. Some codes, required in reactor physics, may be kept outside the SALOME platform (NJOY-2012 and MCNP in our example). It is the responsibility of each application code to develop the application programming interfaces (API) making possible the coupling with SALOME. In this paper, we will focus our attention to the integration of codes DRAGON5 and DONJON5 in the SALOME platform. Two modules of the SALOME platform were investigated during this exercise: the *Geometry* module for defining an arbitrary 2D geometry, and the YACS supervisor module for defining multi-physics computational schemes involving many application codes.

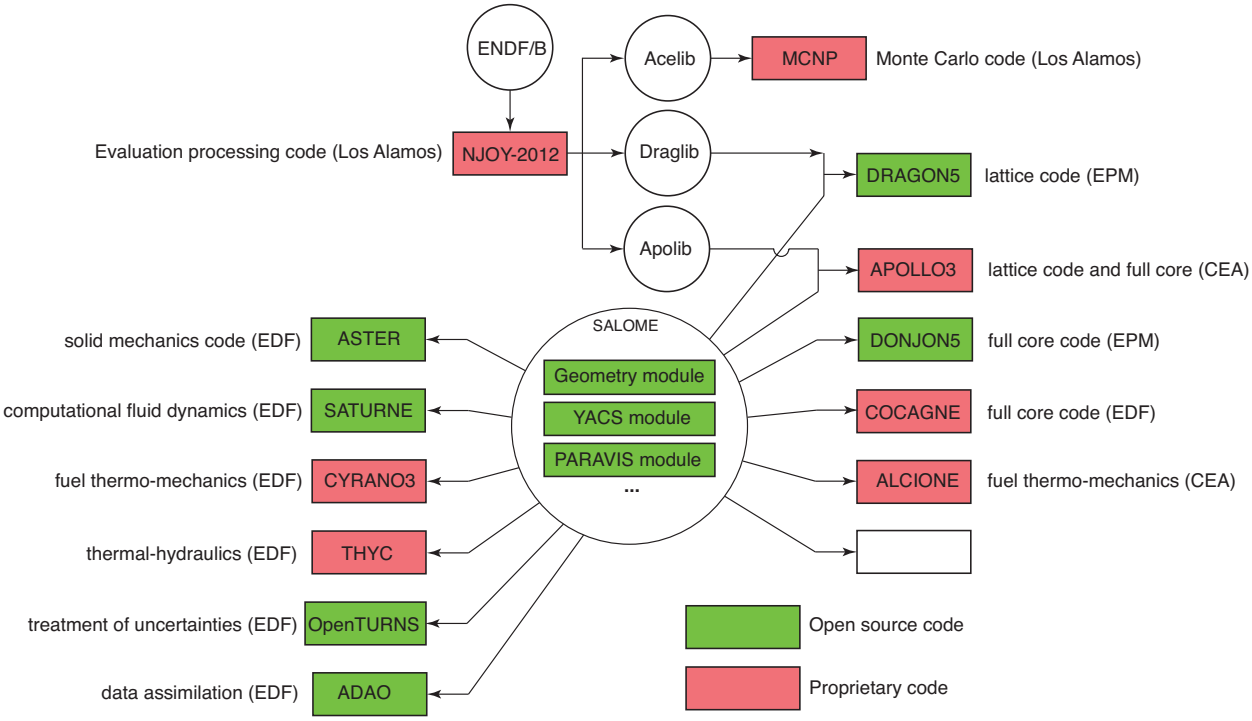


**FIGURE 1.** The SALOME platform

## GEOMETRIC SOFTWARE COMPONENTS

### The Geometry module in SALOME

The *Geometry* module in the open-source SALOME platform provides a rich set of commands to create, edit, import or modify a complex CAD model. The module is powered by a geometry kernel based on the Open CASCADE Technology which provides a boundary representation of the model (BREP) and maintains the topological structure required by the subsequent meshing operations. It provides a powerful set of shape-healing functionalities that can be used to simplify the model or to repair poorly defined imported models. The *Geometry* module functionalities can be accessed through the graphical user interface (GUI). They can also be accessed programmatically in the SALOME Python execution engine that allows building complex automated scripts. In solid modeling and CAD, the boundary representation—often abbreviated BREP—is a method for representing solids is as a collection of connected surface elements. We are promoting the 2D BREP format for representing geometries that are typical of advanced reactors.

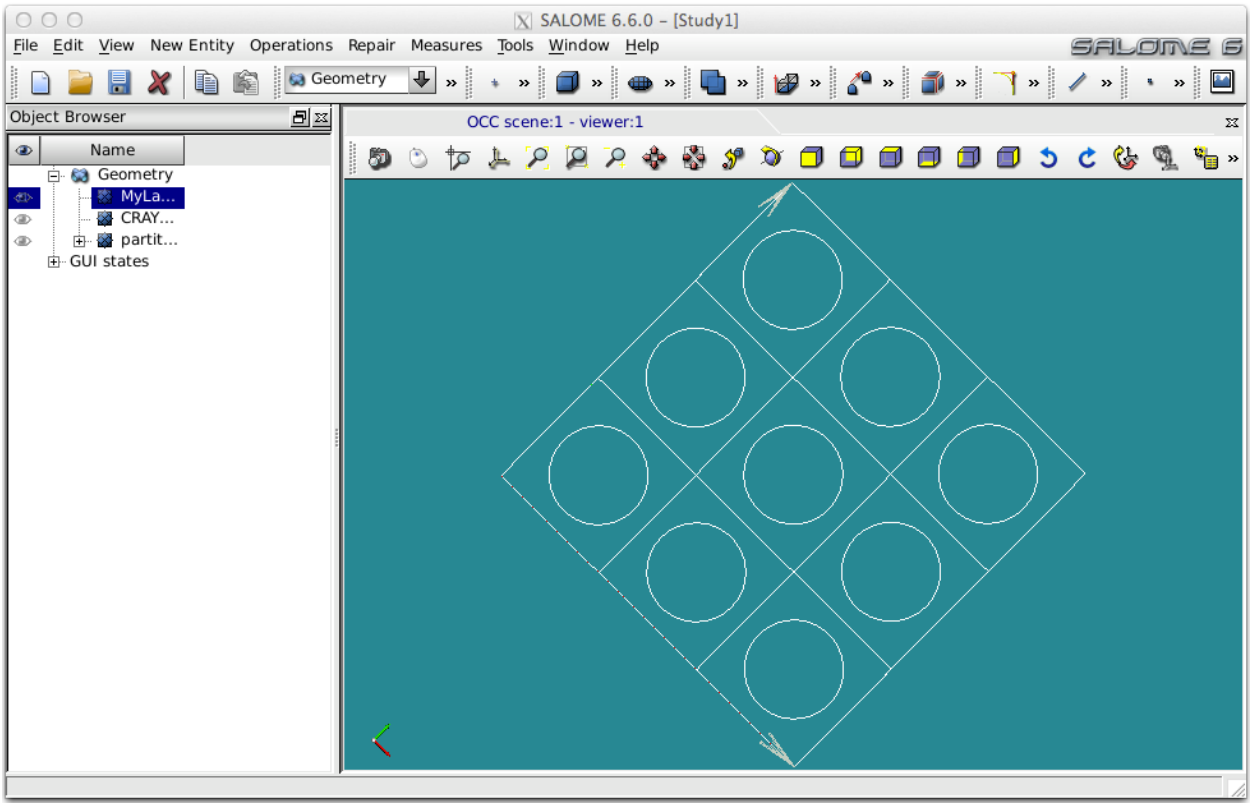


**FIGURE 2.** A simple 3x3 fuel assembly in 2D.

In our current implementation, module *Geometry* of the SALOME platform is used to manage and export an arbitrary 2D geometry as a collection of surface elements of three types: straight line segments, circular arc segments and circles. Figure 2 depicts a simplified 3x3 fuel assembly, used as typical lattice geometry in DRAGON5. A post-processing Python script generates a *surfacic file* containing the collection of surface elements.[5] A more complex geometry will be presented later in this paper.

## The TDT tracking algorithm

The surfacic file is the output of the BREP geometry model and the input of the TDT tracking algorithm. A *tracking* process is applied over the lattice geometry to span a sufficiently large number of neutron trajectories. In a 2D domain, the tracking parameters are the number of azimuthal angles ε and the number of parallel tracks per centimeter. Sets of tracks are drawn over the complete 2D domain. Each set is characterized by a given angle and contains parallel tracks covering the domain. Each track is divided into a number of straight segments of finite length.

The track generation procedure can be performed in a number of ways. We are promoting the TDT tracking algorithm for the specific case of arbitrary geometries in 2D.[6] This algorithm has the unique capability to construct the complete tracking information from the set of 2D surface elements.

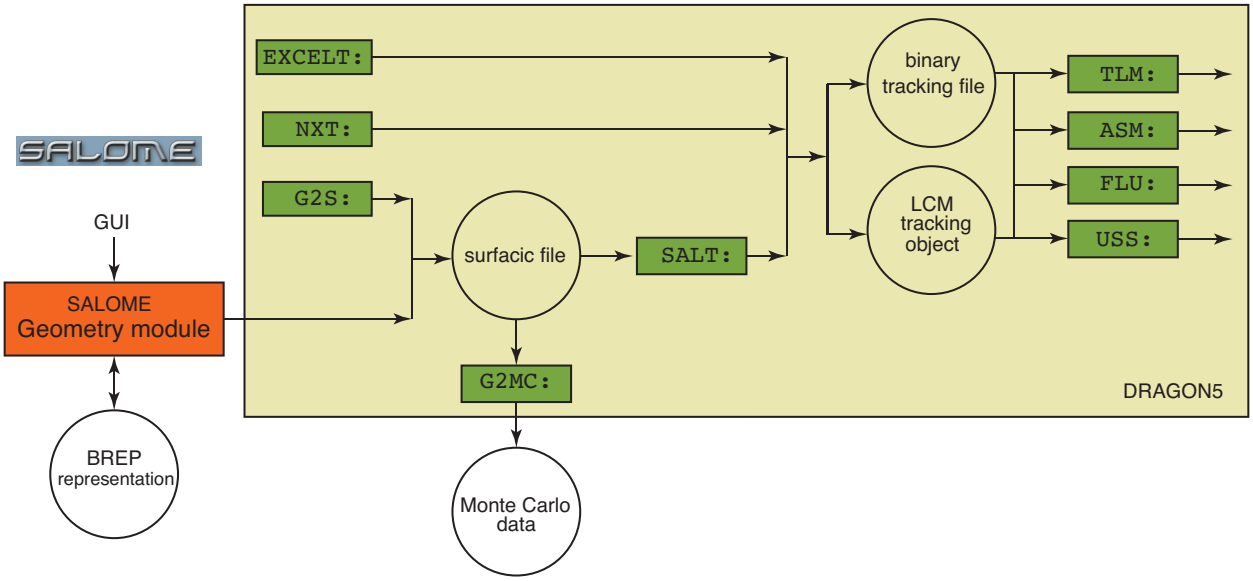


**FIGURE 3.** Data flow of geometric information.

The `SALT:` module available in DRAGON5 is the result of a complete reprogramming of the TDT tracking algorithm as an alternative of the existing `EXCELT:` and `NXT:` tracking modules. `SALT:` is implemented as 6000 lines of Fortran 2003 statements, using application programming interfaces already available in the `NXT:` module of DRAGON3 and DRAGON4.[7] As depicted in Fig. 3., `SALT:` is using a *surfacic file* as input and produces two objects at output: a LCM object containing general information about the geometry and quadratures, and the binary sequential file containing the tracks. Tracks corresponding to a given azimuthal angle ε are separated by a constant distance $\Delta h$ (`DELX` variable in `SALT:`). These tracks are depicted by the `TLM:` module of DRAGON5 for two different values of $\Delta h$ in Fig. 4. We see that tracks become indistinguishable with `DELX = 0.01`. Each track is used forward and backward, corresponding to angles ε and −ε.

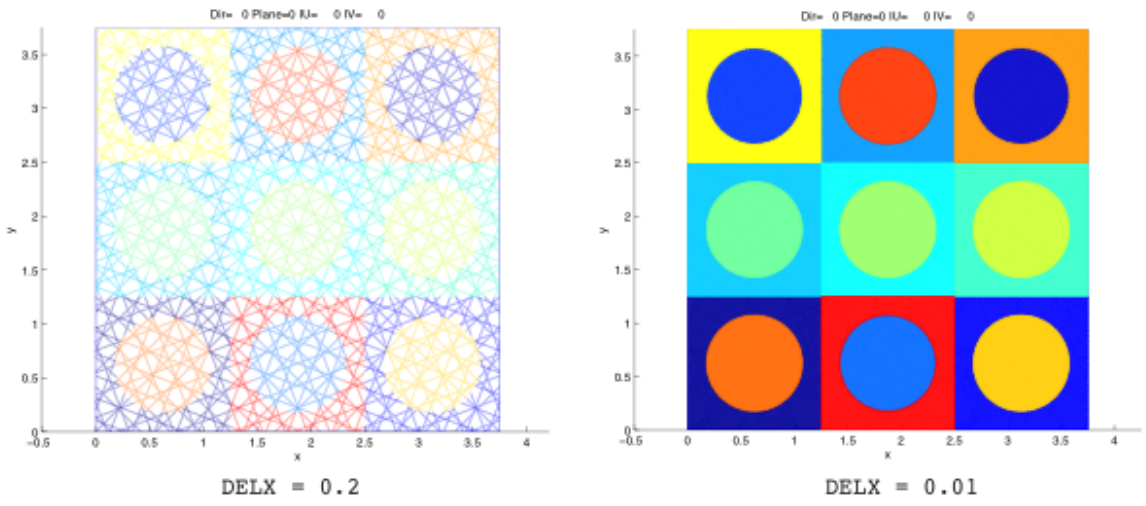


**FIGURE 4.** Tracking of the geometry in Fig. 2 using the SALT: module.

A companion module, named `G2MC:`, can be used to generate Monte Carlo data for codes such as MCNP, TRIPOLI3 and SERPENT, using information available in the surfacic file. Another module, named `G2S:`, allows the production of the surfacic file for simple geometries that can be defined via the `GEO:` module of DRAGON5.

# COMPUTER SCIENCE COMPONENTS

## Designing and executing computational schemes

A presentation of a lattice code is not complete without the introduction of a software component for designing *computational schemes*. A lattice code is a complex software package with numerous data-flow options, numerical techniques and input databases. This extreme flexibility must be managed with a software component to design computational schemes. Such a software component was introduced in the lattice code DRAGON3, back in 1996. This scripting language is known as CLE-2000 and is currently used to program computational schemes in Canada.[8]

## The GANLIB5 kernel

The GANLIB is a small library that is linked to a software application in order to facilitate modularity, interoperability, and to bring generic capabilities in term of data transfer.[9] The GANLIB is an application-programming interface (API) made of subroutines that are called by the software application (e.g., a lattice code) or by the multi-physics surrounding application. In other words, the GANLIB acts as a standardized interface between the software application and the multi-physics applications.

The GANLIB is made of two distinct and inter-related components:

1. CLE-2000 is a compact supervisor responsible for the free-format recovery of input data, for the modularization of the software application and for the insertion of loops and control statements in the input data flow. CLE-2000 permits the conception of computational schemes, dedicated to specific engineering studies, without any need for recompilation of the software application.
2. LCM objects are data structures used to transfer data between modules of the software application and towards the multi-physics application. LCM objects are structures made of *associative table*s and *heterogeneous lists*. These structures are either *memory resident* or *persistent* (i.e., stored in a file). The LCM object API is implemented with access efficiency as its first requirement.

## The Skin++ classes

Skin++ is a C++ wrapper class system giving full access to the GANLIB5 API for a program written in C++. These classes are currently used for coupling GANLIB5-based codes with the SALOME platform. Skin++ allows the complete DRAGON5 capabilities to be available as method calls selected in the three classes depicted in the UML diagram of Fig. 5.

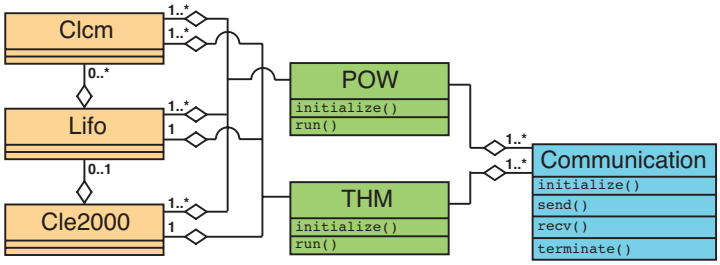


**FIGURE 5.** The UML representation of the C++ system of classes

The three classes have the following capabilities:

- An instance of class `Cle2000` encapsulates a unique CLE-2000 script, generally corresponding to an implementation of a computational scheme.
- An instance of class `Lifo` manage a "Last In First Out" stack used as container for the input/output parameters and objects exchanged with the root CLE-2000 script.

- An instance of class `Clcm` encapsulates a unique LCM object used as container inside the CLE-2000 script or inside DRAGON5 modules called by the root CLE-2000 script.

## SALOME–DRAGON5/DONJON5 APPLICATIONS

### A two-level computational scheme

The first application of the SALOME–DRAGON5 coupling is an implementation of a two-level computational scheme for a 17 x 17 PWR fuel assembly. The fuel assembly is described in the *Geometry* module of SALOME in eighth-of-assembly symmetry, using windmill discretization and sub meshing in the moderator region. The result of this discretization is depicted in Fig. 6.

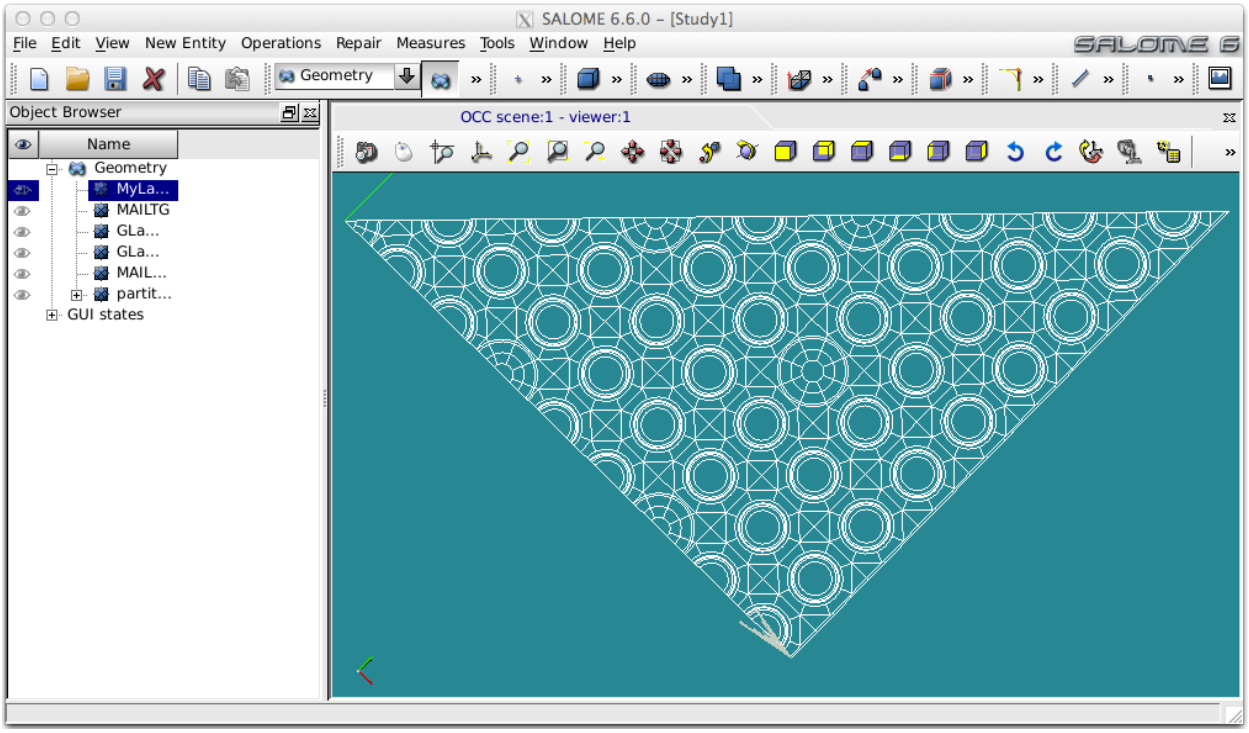


**FIGURE 6.** A 17x17 fuel assembly with windmill discretization.

This computation scheme is similar to the REL-2005 scheme proposed by the CEA for generating multi-parameter assembly databases with the APOLLO2 code.[10] The dataflow of the proposed two-level scheme is depicted in Fig. 7 and is implemented in the CLE-2000 language available in the GANLIB5 kernel. A computational scheme is a collection of CLE-2000 script files (with suffix "`.c2m`").

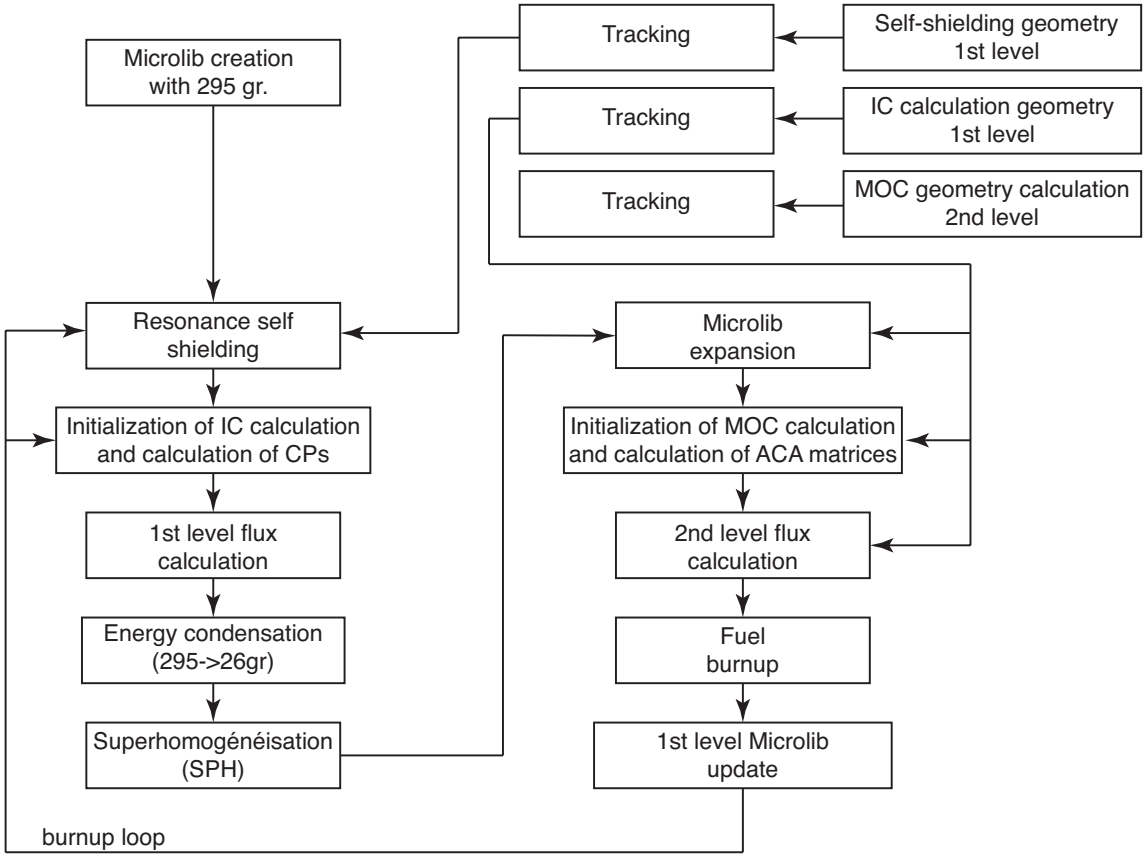


**FIGURE 7.** Data flow for the two-level computational scheme

The two-level computational schemes REL-2005 have been proposed in Ref. 10 as technique to reduce computational costs of PWR assembly calculation for daily production calculations, while maintaining acceptable accuracy. These schemes are based on a series of transport calculations of increasing accuracy with intermediate energy condensation steps. In the case of a 17 x 17 PWR assembly, our implementation in DRAGON5 can be summarized as follows:

- A 37-mixture and 295-group microscopic cross-section library, based in the SHEM–295 energy mesh[11], is defined by procedure `Mix_UOX_32.c2m.`
- A resonance self-shielding calculation is performed using the interface current formalism with double–$P_1$ current approximation and 295 groups. The 37-mixture simplified geometry used in the self-shielding calculation is defined by procedure `Geo_SS_32.c2m`.
- The resonance self-shielding calculation is based on the Subgroup Projection Method (SPM)[11], using probability tables obtained with the CALENDF formalism. The transport equation in each energy group is solved using a flux-current iterative approach, as described in Sect. 3.8.1 of Ref. 1.
- The first-level flux calculation is performed using the interface current formalism with double–$P_1$ current approximation and 295 groups. The 37-mixture simplified geometry used in the self-shielding calculation is defined by procedure `Geo_N1_32.c2m`.
- The microscopic cross sections are condensed to 26 groups.
- A transport-transport SPH equivalence procedure is performed, as explained in Sect. 4.4 of Ref. 1. The macro-calculation is performed using the interface current formalism with double–$P_1$ current approximation. Not performing this SPH correction introduces an error of the order of 190 pcm in the computational scheme.
- The microlib containing microscopic cross-section information is expanded from 37 to 161 mixtures. This task is performed by procedure `MultLIBEQ_32.c2m`.
- The neutron flux in the second level is computed using the method of characteristics (MOC) available in DRAGON5.[12] Specular reflection is represented on assembly boundary using infinite cycling tracks in the

domain. Scattering anisotropy is limited to the transport-corrected $P_0$ approximation. Finally, the MOC method is accelerated using the Algebraic Collapsing Acceleration (ACA) technique presented in Ref. 13. The tracking used by the ACA technique and by the MOC is based on the SALOME surfacic file. A Macrolib with 26 energy groups is used.

Results obtained using our two-level computational scheme ate validated with respect to the SERPENT2 Monte Carlo code.[14] SERPENT2 is using a continuous-energy representation of the microscopic cross sections. SERPENT2 is used to obtain a reference effective multiplication factor ($K_{eff}$) and power distribution in the PWR assembly as a function of assembly type (UOX, MOX, presence of poisons, etc.) and burnup. Care should be taken to use the same geometry and the same ENDF/B evaluation in SERPENT2 and DRAGON5. The Monte Carlo simulation is using 2,000,000 neutrons and 1,000 cycles at each burnup step. The Bateman equations are solved using the Chebyshev Rational Approximation Method (CRAM).

**Table 1.** Zero-burnup calculations

| | SERPENT2 | | DRAGON5 | | |
|---|---|---|---|---|---|
| | analog | implicit | 295-group | 295/26–grp (no sph) | 295/26–grp (sph) |
| Pin cell | 1.37572 ± 0.00093 | 1.37539 ± 0.00036 | 1.374360<br>-103 pcm | 1.372913<br>-248 pcm | 1.374826<br>-56 pcm |
| 17x17 assembly | 1.37053 ± 0.00094 | 1.36876 ± 0.00038 | 1.368348<br>-41 pcm | 1.367015<br>-175 pcm | 1.368985<br>22.5 pcm |

Zero-burnup results obtained with the two codes are presented in Table 1, corresponding to a single pin cell and to the complete 17 x 17 PWR assembly. We notice the important reduction in CPU time obtained with the two-level scheme.

**Table 2.** Burnup-dependent calculations

| | SERPENT2 | 295-group | 95/26–grp (no sph) | 295/26–grp (sph) |
|---|---|---|---|---|
| CPU time | 173 hour | 138 hour | 4 hour | 8 hour |

Burnup dependent results are presented only for the effective multiplication factor and are limited to 40,000 Mw-day/tonne. They are presented in Table 2 and Fig. 8.

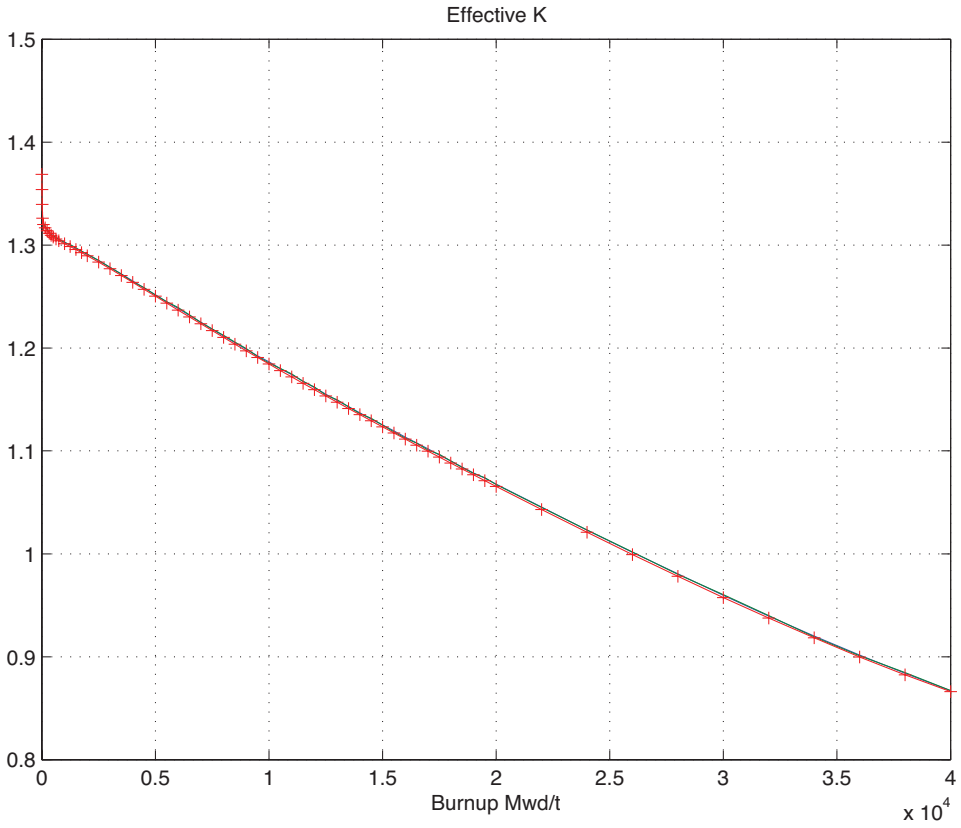


**FIGURE 8.** Evolution of the effective multiplication factor in SERPENT2 and DRAGON5 (two-level)

## A multiphysics application

The second application of the SALOME–DONJON5 coupling is an implementation of a simple steady-state multi-physics iteration involving reactor physics and simplified thermal-hydraulics of a 900 MWe PWR. This application converges the assembly power distributions and thermal-hydraulics values for the fuel temperature, coolant temperature and density. This problem is first solved within DONJON5, without using SALOME, in order to obtain a reference convergence history. The data-flow corresponding to this problem is depicted in Fig. 9 and was programmed entirely in the CLE-2000 control language available in DONJON5.

Three CLE-2000 procedures were implemented. `IniPowCompo.c2m` is responsible for the initialization of the calculation and for setting the initial conditions of the iterative procedure. The Fmap object contains the parameter distributions in power, burnup, fuel temperature, coolant temperature and coolant density. `PowComponent.c2m` is a neutron flux and power calculation involving a single finite-element full-core calculation in DONJON5. `ThmComponent.c2m` is a simplified thermal-hydraulics calculation, applied over each assembly, assuming steady-state conditions. These three CLE-2000 procedures are exchanging information via data containers named *LCM objects* and implemented in ANSI C. `PowComponent` and `ThmComponent` are iterated using a while loop until convergence of the coolant temperature. The while loop can be implemented in CLE-2000, C or C++ language.

As a final exercice, we have chosen to construct a computational scheme using the coupling capabilities of the YACS supervisor module. A YACS computational scheme is made of a set of calculation nodes corresponding to single actions such as a calculation step made with a CLE-2000 or Python script, a post-treatment service, etc. A calculation node can also be a more complex operation such as a *while* loop, a *for each* distribution over many processors, a *block* node defining a sub-scheme, etc. Nodes are communicating together using input and output *ports* corresponding to variables of different types, and are implemented using the CORBA protocol. We are differencing *dataflow* ports transmitting dummy variables between different nodes of a YACS scheme and *datastream* ports

transmitting information via *Calcium* ports, making possible data synchronization during the execution of the node.[4] This coupling approach was previously presented in Ref. 15.

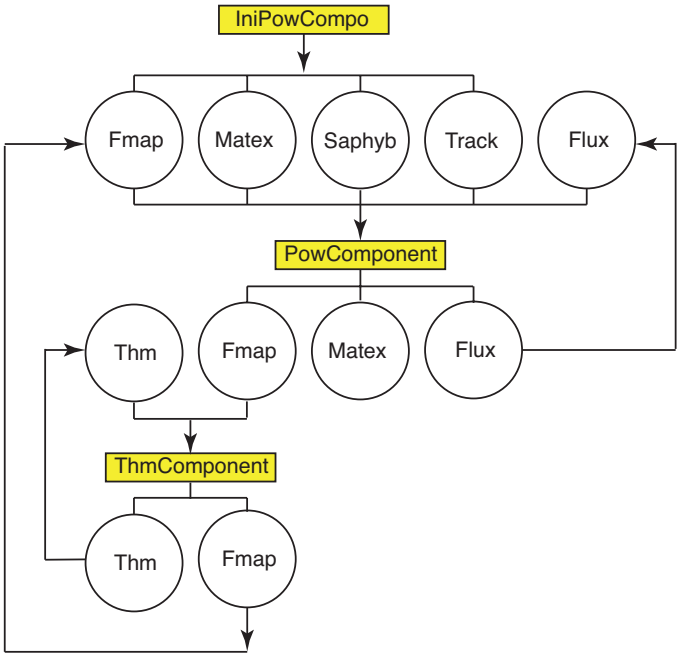


**FIGURE 9.** Data flow for a multiphysics iteration in DONJON5

The SALOME coupling between `PowComponent` and `ThmComponent` consists to program two *while* loops in C++, one embedding each of these two procedures and to synchronize them using the Calcium API. This approach is now presented.

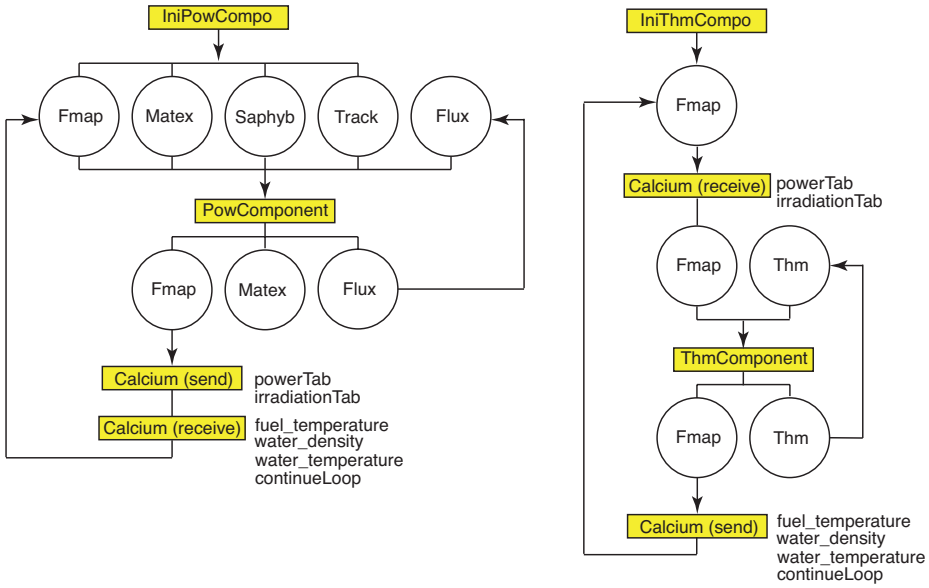


**FIGURE 10.** Data flow for a multiphysics iteration in SALOME

Each CLE-2000 procedure, `POWcomponent.c2m` and `THMcomponent.c2m`, is encapsulated into a single *while* loop, programmed in C++ using the classes presented in Figure 5 and implemented in a `run()` method of two classes: `POW` and `THM`, respectively. These *while* loops are depicted in Fig. 10. In this case, the synchronization of the two loops is based on the iteration index, using a mechanism available in Calcium. Moreover, the `recv()` and `send()` methods off the `Communication` class are used to exchange scalar field information between the two loops. In YACS terminology, these scalar fields are using *datastream* ports. This means that `POW` and `THM` classes can execute on different CPUs, introducing a form of parallelism in this computational scheme.

The SALOME module based on classes `POW` and `THM` is constructed using YACSGEN, a Python package available in the SALOME distribution. YACSGEN construct the SALOME module automatically from a short description of its two components `POW` and `THM`, including descriptions of dataports and datastreams they use. A Python script is used to provide the required information. A view of the graphical user interface obtained using YACSGEN is presented in Fig. 11.

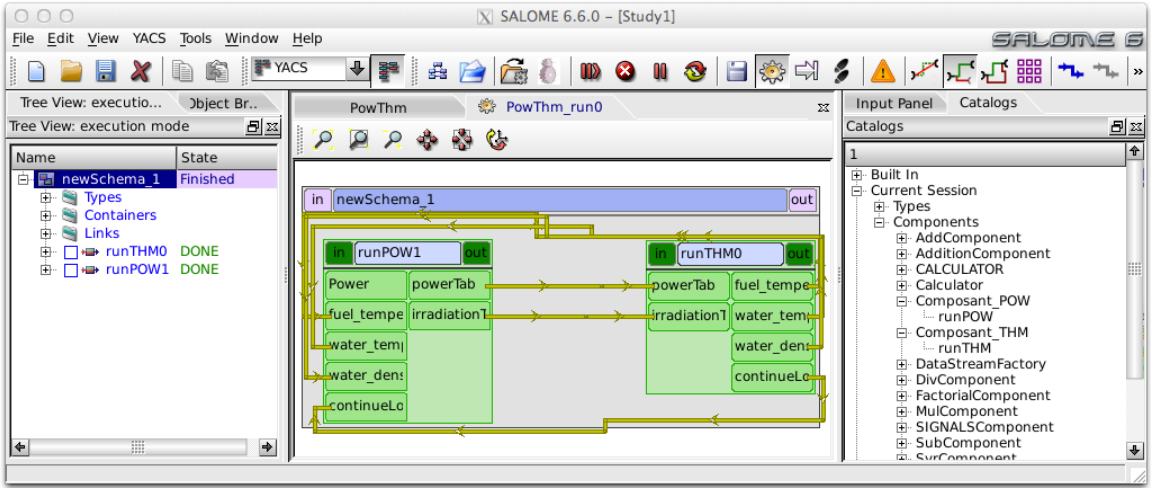


**FIGURE 11.** Graphical User Interface of the YACS module. The datastreams are depicted in yellow.

The multi-physics study is a simple fixed-point iteration for obtaining stable steady-state temperature and coolant density distributions over the core. We have selected a typical reactor of the 900 MWe CP1 model with inlet coolant temperature of 559 K, uniform coolant pressure of 155.1 bar and coolant density of 0.7160 g/cc. As preliminary step, a multi-parameter reactor database in Saphyb format is constructed using the DRAGON5 lattice code. A typical 17 x 17 fuel assembly of the CP1 model is analyzed using a lattice-code computational scheme similar to REL-2005[10], collecting *delta-sigma branch points* for the following global parameters:

- fuel burnup for $0 \leq B \leq 40{,}000$ MW-day/tonne
- Boron concentration for $0 \leq C_B \leq 2000$ ppm
- fuel temperature for $550 \leq T_f \leq 1{,}150$ K (nominal value = 800 K)
- coolant temperature for $500 \leq T_c \leq 700$ K (nominal value = 600 K)
- coolant density for $0.5 \leq N_c \leq 0.8$ g/cc (nominal value = 0.659 g/cc)

The computational scheme for obtaining the stable temperature and coolant density distributions over the core is now available in YACS and is made of two coupled components:

- The `POW` component implements a reactor physics DONJON5 calculation. Three scalar fields are recovered from a previous `THM` calculation (or are set as initial-condition values): the fuel temperature, the coolant temperature and the coolant density in core. The calculation starts with a multi-parameter interpolation of the cross section and diffusion coefficient information available in the Saphyb, as explained in Sect. 4.6 of Ref. 1. Next, a solution of the 3D diffusion equation over the full core is obtained using the finite element

method, available in DONJON5. Finally, the resulting neutron flux distribution is used to compute the thermal power distribution over the core.

- The thermal-hydraulics calculation uses two scalar fields at input: the thermal power distribution and the burnup distribution in core. The `THM` component implements a simplified thermo-hydraulics calculation performed in parallel for each assembly in the core, neglecting any interaction with neighbor assemblies and assuming constant pressure. This simplified model is an homogeneous model that remains valid only at liquid or subcooled boiling conditions. The conduction equations for a complete assembly are replaced by a solution of the conduction equation of an averaged fuel pin. Finally, four output scalar field is computed: the fuel temperature, the coolant temperature, the fuel surface temperature and the coolant density in core.

The overall convergence process takes 14 iterations to converge, with a stopping criterion of 1K on the coolant temperature. The resulting coolant temperature distribution is depicted in Fig.12.

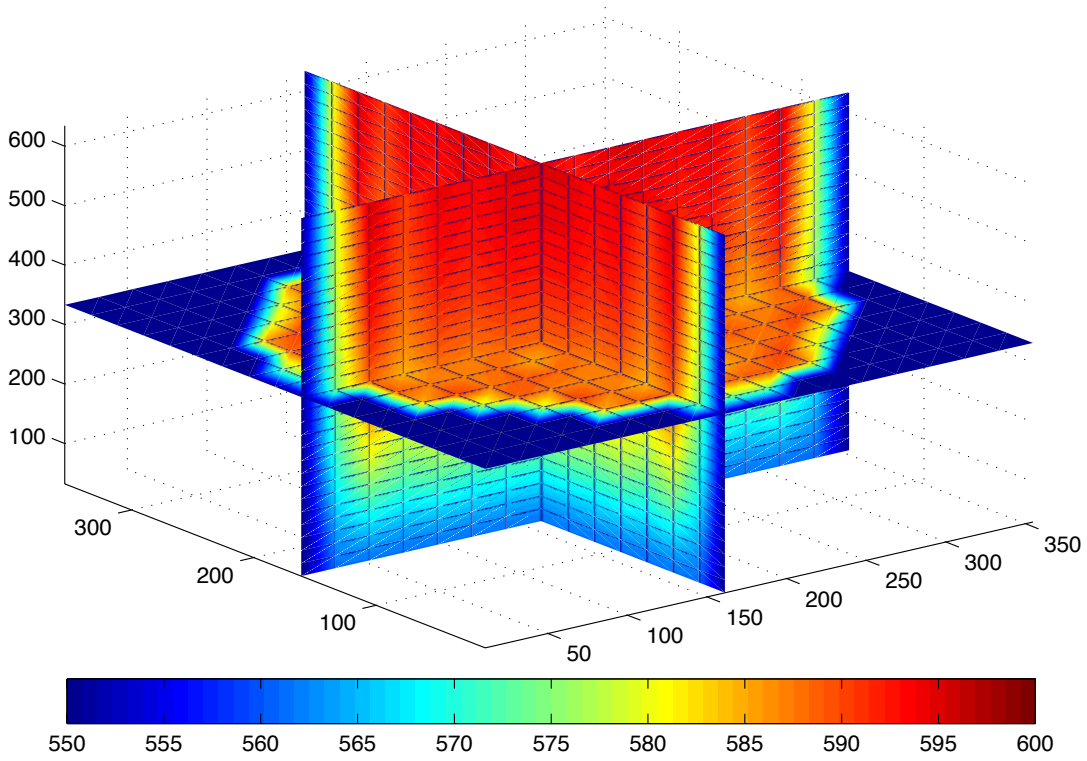


**FIGURE 12**. Coolant temperature distribution in core

## CONCLUSION

The SALOME and DRAGON5/DONJON5 codes are proposed as tools to implement computational schemes dedicated to advanced nuclear reactor. This system offers computer-efficient solutions consistent with daily calculations. The new tracking module `SALT:` is making the interface between the *Geometry* module of SALOME and the solution techniques of DRAGON5. The current implementation of the interface is limited to 2D geometric models but without any limitation about the complexity of the geometry. Finally, a system of C++ classes allow the integration of any CLE-2000 computational scheme as a YACS component in SALOME. Such integration in the SALOME platform using the YACS supervisor module allows for the realization of multiphysics simulations.

## ACKNOWLEDGMENTS

The author is grateful to the development teams of projects SALOME and COCAGNE at EDF/R&D division for providing a constant support during the realization of this work.

## REFERENCES

1. Hébert, A., *Applied Reactor Physics*, Presses Internationales Polytechnique, ISBN 978-2-553-01436-9, 424 p., Montréal, 2009. See the web site at https://moodle.polymtl.ca/course/view.php?id=1233
2. Marleau, G., Hébert, A. and Roy R., "New Computational Methods Used in the Lattice Code Dragon," *Proc. Int. Topl. Mtg. on Advances in Reactor Physics*, Charleston, USA, March 8–11, 1992, American Nuclear Society. DRAGON can be downloaded from the web site at http://www.polymtl.ca/merlin/
3. A. Hébert, "DRAGON5: Designing Computational Schemes Dedicated to Fission Nuclear Reactors for Space," *Int. Conf. on Nuclear and Emerging Technologies for Space*, Albuquerque, NM, February 25–28, 2013.
4. Ribes A. and Caremoli. C., "The Salome platform component model for numerical simulation," *In COMPSAC 07: Proceeding of the 31st Annual International Computer Software and Applications Conference*, pages 553–564, Washington, DC, USA, 2007, IEEE Computer Society. SALOME can be downloaded from the web site at http://www.salome-platform.org
5. Pora, Y., personnal communication.
6. Lyoussi-Charrat, N., "Calcul du transport neutronique dans le code APOLLO2 par la méthode des probabilités de collision dans une géométrie cartésienne générale," Thèse de doctorat, Université de Clermont-Ferrand 2, France, Mars 1994.
7. Prabha, H., Marleau, G. and Hébert, A., "Tracking algorithms for multi-hexagonal assemblies (2D and 3D)," *Ann. Nucl. Energy,* **69**, 175 (2014).
8. Roy, R., "The CLE-2000 toolbox," École Polytechnique de Montréal, Report IGE-163, December 1999.
9. Hébert, A. and Roy, R., "The GANLIB5 kernel guide (64–bit clean version)," École Polytechnique de Montréal, Report IGE-332, October 2012.
10. J-F. Vidal, R. Tran, O. Litaize, D. Bernard, A. Santamarina, C. Vaglio-Gaudard, "New modelling of LWR assemblies using the APOLLO2 code package," *Proc. Joint Int. Top. Mtg. on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007)*, Monterey, CA, USA, April 15-19, 2007.
11. Hébert, A., "Development of the Subgroup Projection Method for Resonance Self-Shielding Calculations," *Nucl. Sci. Eng.* **162**, 56–75 (2009).
12. Le Tellier, R. and Hébert, A., "On the integration scheme along a trajectory for the characteristics method", *Ann. nucl. Energy*, **33**, 1260–1269 (2006).
13. Le Tellier, R. and Hébert, A., "An improved Algebraic Collapsing Acceleration with General Boundary Conditions for the Characteristics Method," *Nucl. Sci. Eng.,* **156**, 121–138 (2007).
14. Leppänen, J., "Development of a New Monte Carlo Reactor Physics Code," D.Sc. Thesis, Helsinki University of Technology (2007), VTT Publications 640, 2007. See the web site at http://montecarlo.vtt.fi
15. Hébert, A., "Integration of the DRAGON5/DONJON5 codes in the SALOME platform for performing multi-physics calculations in nuclear engineering" *Joint Int. Conf. on Supercomputing in Nuclear Applications and Monte Carlo 2013 (SNA + MC 2013),* La Cité des Sciences et de l'Industrie, Paris, France, October 27–31, 2013.