Proceedings of Nuclear and Emerging Technologies for Space 2013
Albuquerque, NM, February 25-28, 2013
Paper 6722

DRAGON5: Designing Computational Schemes Dedicated to Fission Nuclear Reactors for Space

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 is 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. The DRAGON code is designed around a scripting language, known as CLE-2000, to automate the flow of information used in production computational schemes

DRAGON5 is the latest version of the code and features characteristics that are required to design nuclear reactors for space. First, a new kernel, named GANLIB5, was designed 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 space applications. The first release of DRAGON5 was available in October 2013. Subsequent releases may include couplings with the geometry module (*Geometry*) and supervisor module (YACS) of the SALOME platform. Computer-aided design capabilities will be first offered in 2D and will permit the interoperability between BREP, IGES, STEP and ACIS formats.

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 leakage 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. The idea of using a lattice code to model a full-core geometry is a valuable idea for describing small reactors, typical of those used in space applications. This paper is a first presentation of the improvements made in the latest release of the lattice code DRAGON in order to support this full-core approach.

The proposed software components are selected in two open-source frameworks. 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.[2] 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. The second open-source framework is DRAGON5, a lattice code used for solving the neutron transport equation. The DRAGON family of lattice codes is in development since 1991 and will be the main subject of this paper.[3]

SALOME and DRAGON5 offer a powerful ecosystem for designing spatial 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. We will now present the main components of this system.

GEOMETRIC SOFTWARE COMPONENTS

A BREP geometry model

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 spatial reactors. 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 1 depicts a BREP-formatted fuel assembly from a boiling water reactor (BWR), used as typical lattice geometry in DRAGON5. More complex geometries have recently been studied.

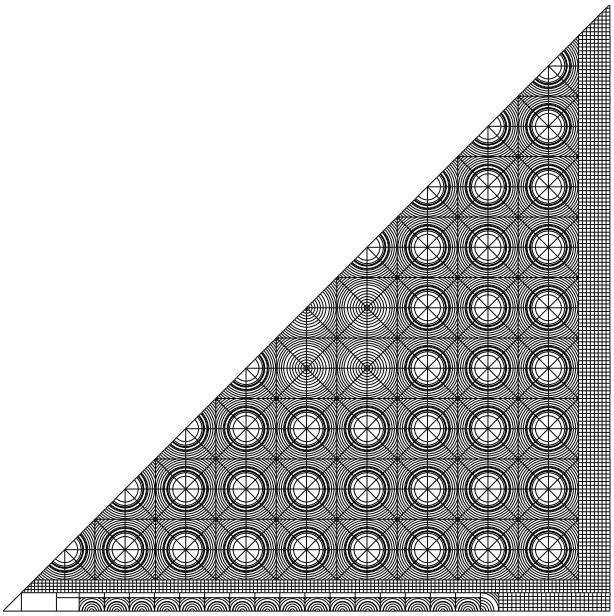

FIGURE 1. A BREP-formatted boiling water assembly in 2D.

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. The *Geometry* module can import geometry from IGES, STEP, in BREP and ACIS format. It also 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.

The TDT tracking algorithm

The output of the BREP geometry model is 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. Tracks in a set are separated by a constant distance Δh as depicted in Fig. 2 for the particular case of a 2D square pin-cell. Each track is used forward and backward, corresponding to angles ε and −ε.

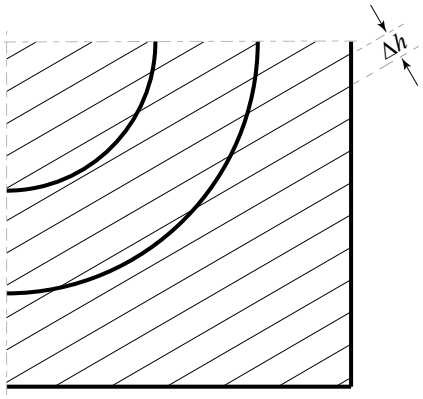

FIGURE 2. Tracking of a simple pin-cell.

The track generation procedure can be performed in a number of ways.[1] We are promoting the TDT tracking algorithm for the specific case of nuclear reactors in space.[4] This algorithm has the unique capability to construct the complete tracking information from the set of 2D BREP surface elements. The TDT tracking approach was recently applied over a detailed 2D eight-of-core PWR geometry with exact boundary conditions. This geometry features 142,872 regions and 263,366 surface elements. We are currently using an implementation of the TDT tracking algorithm designed by Warin in 2002 and implemented in module `SALT:` of the DRAGON5 lattice code.[5]

NUMERICAL SOFTWARE COMPONENTS

The core of the DRAGON5 calculation is a deterministic solution of the neutron transport equation. We are promoting two numerical methods for the specific case of nuclear reactors in space: the algebraic collapsing (AC) method and the method of characteristics (MOC). The neutron transport equation governs the distribution of neutron flux $\phi(r,E,\Omega)$ in phase space. It is written

$$\Omega \cdot grad\,\phi(r,E,\Omega) + \Sigma(r,E)\phi(r,E,\Omega) = Q(r,E,\Omega) \tag{1}$$

where $\Sigma(r,E)$ is the macroscopic total cross section and $Q(r,E,\Omega)$ is the scattering and fission source.

The Algebraic Collapsing method

The second order algebraic collapsing (AC) method was introduced by Suslov as a general technique for the acceleration of the method of characteristics (MOC) in 2D and 3D.[6] The core of the ACA is the solution of a simplified transport equation, referred to as the ACA operator, leading to even-parity fluxes similar to those obtained with the diffusion theory. The discretization process is based on a weighted summation of all neutron trajectories over the global angular domain, as recovered from the TDT tracking. The resulting sparse matrix system can be scattering reduced and solved for the even-parity flux and boundary currents. The solution of the sparse matrix system requires only small computer resources and is therefore a good candidate for synthetic acceleration or for obtaining a first estimate of the neutron flux. The second-order AC method was introduced in DRAGON4 in 2007 and successfully used as synthetic acceleration technique for the MOC.[7]

The first-order AC method is a new class of simplified characteristics schemes for solving the transport equation for neutral particles with scattering anisotropy. These algorithms are similar to the second-order AC synthetic acceleration method introduced by Suslov. However, our implementation differs in noticeable ways: The proposed schemes are based on a first-order formulation consistent with a treatment of scattering anisotropy whereas Suslov's scheme is a second-order scheme limited to isotropic scattering. Moreover, the proposed scheme is valid in voided regions and can be extended to linear-characteristics and quadratic-order diamond differencing, leading to a consistent treatment of linear discontinuous sources. The first-order AC method is intended to provide approximate solutions of the transport equation with limited CPU resources and could also be implemented within synthetic acceleration schemes of the MOC. Every proposed scheme is compatible with existing TDT tracking files.

The Method of Characteristics

The method of characteristics (MOC) solves the characteristic form of the transport equation by following the straight neutron paths of the neutral particle as it moves across the domain.[8] This approach is based on an iterative calculation of the particle flux by solving the transport equation over tracks crossing the complete domain. The MOC is generally applied to the multi-group form of the transport equation and to spatial domains made of regions with piecewise-uniform nuclear properties. The scalar flux per region and energy group is constructed by collecting all mean angular fluxes in terms of the entering angular flux and the source inside the region. Interestingly, the MOC has the capability to use standard TDT tracking information.

The MOC offers an alternative to the collision probability (CP) method in order to overcome its two main limitations:

1. The CP method produces full square matrices of order equal to the number of regions in the domain, and

2. The CP method is limited to isotropic sources in the LAB. In particular, the MOC is to be preferred in cases where the number of regions exceeds a few hundred.

The application of the MOC to reactor physics was first reported by Askew in 1972, followed by a first production implementation, known as the CACTUS module of WIMS-E.[8,9] The CACTUS code is based on the concept of *cyclic tracking* where the characteristics are infinite in length and periodic with respect to the lattice. The same idea of periodic characteristics has also been implemented in the MOCC module of DRAGON3 and MCCG module of DRAGON4.[10,11]

Another class of characteristic solvers is based on the explicit representation of boundary conditions and on the use of finite-length characteristics. This class of solvers is noticeably faster than those based on the cyclic tracking approach and is used in production lattice calculations. The most widely known implementations are those of codes MCCG3D and CASMO-4.[6,12] Similar techniques were subsequently implemented in codes APOLLO2 and DRAGON.[13,14]

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. A breakthrough achievement was made in 1987. A software component, named GIBIANE, was first introduced in the APOLLO2 lattice code, designed by the CEA in France.[15] GIBIANE is en embedded scripting language designed specifically to program computational schemes for a variety of nuclear reactor types. It is important to understand that a computational scheme must be validated against experiments or against Monte Carlo reference calculations before being used in daily production calculations. In other words, this is the computational scheme that is validated, not the lattice code alone. A similar 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. CLE-2000 could be a very useful component for spatial reactor applications.[16]

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.[17] 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 application, as depicted in Fig. 3.

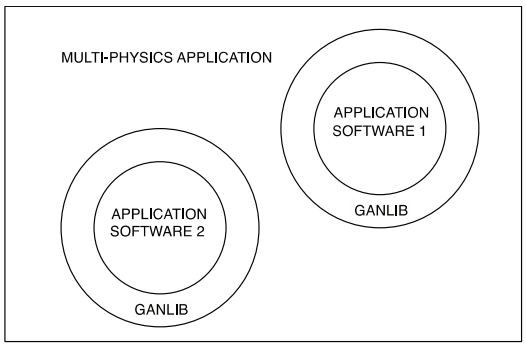

FIGURE 3. Implementing a multi-physics application.

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, even for frequent calls with small chunks of data.

The GANLIB Version 5 is implemented in the ANSI C programming language in order to maximize its compatibility in a multi-physics environment where different components are implemented in various programming languages (C++, Fortran, Java, etc.). The GANLIB Version 5 is *64-bit clean*, another benefit of using an ANSI C implementation. This last property allows the execution of software applications with 32-bit integers and 64-bit addresses. Specific Fortran APIs are also available and are implemented according to the C interoperability mechanism, available in Fortran 2003 and standardized by the International Organization for Standardization (ISO). Consequently, the computational modules in DRAGON5 are programmed in Fortran-2003.

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 and ROOT[18] platforms. Skin++ allows the complete DRAGON5 capabilities to be available as method calls selected in the three classes depicted in the UML diagram of Fig. 4.

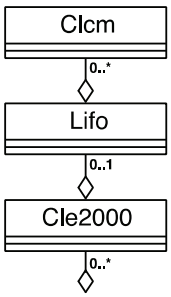

FIGURE 4. The UML representation of Skin++

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. Only the root CLE-2000 script needs to be encapsulated. A typical implementation takes parameter values at input and produces database objects at output. This setup is used in cases where a CLE-2000 computational scheme is to be executed from a C++ framework. It is always possible to execute a CLE-2000 script in stand-alone mode without using Skin++.
- 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.

In the following example, a computational scheme named "`pwr2010`" is executed using input parameters ("`draglib`", "`type`", "`fuel`", etc.) and produces an output database in XSM format named "`saphyb`".

```
#include "Cle2000.hxx"
using namespace boost;  using namespace std;  using namespace ganlib;

int main(int argc, char **argv) {
  // construct the lifo stack
  LifoPtr ipLifo = LifoPtr(new Lifo());
  ipLifo->pushEmpty("saphyb", "XSM", "./saphyb_UOX_AICN");
  ipLifo->push("draglib", string("draglibendfb7r1SHEM295"));
  ipLifo->push("type", string("FRA"));
  ipLifo->push("fuel", string("UOX"));
  ipLifo->push("rod", string("AICN"));
  ipLifo->push("homoge", string("HOMOGENE"));
  ipLifo->push("u235", float_32(3.7));
  ipLifo->push("burnup", float_32(85000.));
  ipLifo->push("nrgoup", int_32(2));
  ipLifo->lib();

  // call the procedure with in-out CLE-2000 variables
  Cle2000Ptr ipCle2000 = Cle2000Ptr(new Cle2000("pwr2010", 0, ipLifo));
  ipCle2000->exec();

  // recover the saphyb
  ClcmPtr xsm_object; ipLifo->node("saphyb", xsm_object);
  xsm_object->open("READ-ONLY");
  xsm_object->lib();
  xsm_object->close("KEEP");

  // erase the lifo stack
  while (ipLifo->getMax() > 0) ipLifo->pop();

  // end of case ------------------------------------------------------------
  cout << "Skin++: successful end of execution" << endl;
  return 0;
}
```

The Skin++ classes are also permitting the interoperability of DRAGON5 with the ROOT package from CERN.[18] The first bricks for a very fast and intuitive way to analysis the DRAGON5 results have been put together in the graphic user interface DDGUI.[19] Based on the extensive ROOT C++ package, the possible features are numerous. For this first version of the software, we have programmed the fundamental tools which may be the more useful on an everyday basis: view the data structures content, draw the geometry and draw the flux or power from a DRAGON5 computation. A typical DDGUI window is represented in Fig. 5. The tests show how amazingly fast the user can get the information needed for a general overview or more precise analyses. Several other features will be implemented in the near feature.

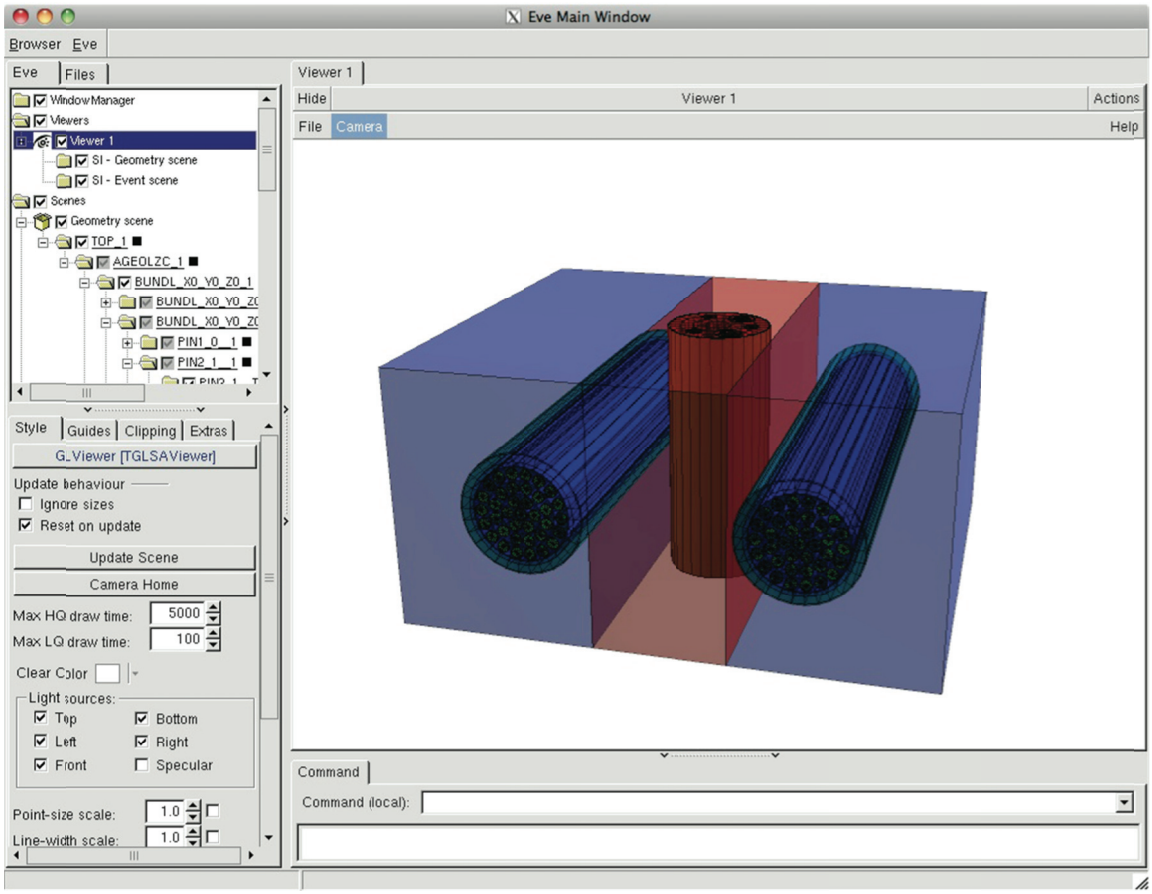

FIGURE 5. A DDGUI typical window

CONCLUSION

The SALOME and DRAGON5 codes are proposed as tools to implement computational schemes dedicated to nuclear reactors in space. This system is purely deterministic and offers computer-efficient solutions consistent with daily calculations. The current implementation is limited to 2D geometric models but without any limitation about the complexity of the geometry.

In the future, this system could be extended to 3D geometric models provided that the TDT tracking technology could be generalized. Such an improvement would require major R&D efforts, but would open the way towards innovative reactor designs.

ACKNOWLEDGMENTS

This work was supported by a grant from the Natural Science and Engineering Research Council of Canada.

REFERENCES

1. Hébert, A., *Applied Reactor Physics*, Presses Internationales Polytechnique, ISBN 978-2-553-01436-9, 424 p., Montréal, 2009. Complementary information can be downloaded from the web site at https://moodle.polymtl.ca/course/view.php?id=1233
2. 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
3. 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/
4. 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.
5. Warin, X., "Notice théorique de la méthode des caractéristiques 2D et du générateur de trajectoires SALT," École Polytechnique de Montréal, Report IGE-329, Mars 2002.
6. I. R. Suslov, I. R., "Solution of Transport Equation in 2– and 3–Dimensional Irregular Geometry by the Method of Characteristics," *Int. Conf. Math. Methods and Supercomputing in Nuclear Applications*, Karlsruhe, April 19–23, 1993.
7. Le Tellier, R., and Hébert, A., "An Improved Algebraic Collapsing Acceleration with General Boundary Conditions for the Characteristics Method," *Nucl. Sci. Eng*., **156**, 121 (2007).
8. Askew, J. R., "A Characteristics Formulation of the Neutron Transport Equation in Complicated Geometries," Report AEEW-M 1108, United Kingdom Atomic Energy Establishment, Winfrith (1972).
9. Halsall, M. J. "CACTUS, A Characteristics Solution to the Neutron Transport Equation in Complicated Geometries," Report AEEW-R 1291, United Kingdom Atomic Energy Establishment, Winfrith (1980).
10. Roy, R., "The Cyclic Characteristics Method," *Int. Conf. Physics of Nuclear Science and Technology*, Long Island, New York, October 5–8, 1998.
11. Le Tellier, R., and Hébert, A., "On the Integration Scheme Along a Trajectory for the Characteristics Method," *Ann. Nucl. Energy*, **33**, 1260 (2006).
12. Knott, D. and Edenius, M., "Validation of the CASMO-4 Transport Solution," *Int. Conf. Math. Methods and Supercomputing in Nuclear Applications*, Karlsruhe, April 19–23, 1993.
13. Sanchez, R. and Chetaine, A., "A Synthetic Acceleration for a Two-Dimensional Characteristic Method on Unstructured Meshes," *Nucl. Sci. Eng.*, **136**, 122 (2000).
14. Wu, G. J. and Roy, R., "A New Characteristics Algorithm for 3D Transport Calculations," *Ann. Nucl. Energy*, **30**, 1 (2003).
15. Loubière, S., Sanchez, R., Coste, M., Hébert, A., Stankovski, Z., Van Der Gucht, C. and Zmijarevic, I., "APOLLO2, Twelve Years Later," paper presented at the *Int. Conf. on Mathematics and Computation, Reactor Physics and Environmental Analysis in Nuclear Applications*, Madrid, Spain, September 27–30, 1999.
16. Roy, R., "The CLE-2000 toolbox," École Polytechnique de Montréal, Report IGE-163, December 1999.
17. Hébert, A. and Roy, R., "The GANLIB5 kernel guide (64–bit clean version)," École Polytechnique de Montréal, Report IGE-332, October 2012.
18. O. Couet, "Multiple data sets visualization in ROOT," J. of Physics: Conference Series **119** (2008).
    ROOT can be downloaded from the web site at http://root.cern.ch/
19. Chambon, R. and Marleau, G., "DDGui, a new and fast way to analyse DRAGON and DONJON code results," PHYSOR 2012 Advances in Reactor Physics Linking Research, Industry, and Education Knoxville, Tennessee, USA, April 15–20, 2012, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2012).