PyDrag, A User-Friendly Approach to Dragon Deterministic Code

Vivian Salino

PYDRAG, A USER-FRIENDLY APPROACH TO DRAGON DETERMINISTIC CODE

V. Salino
Institut de Radioprotection et de Sûreté Nucléaire (IRSN),
PSN-RES/SNC/LN,
BP 17, Fontenay-aux-Roses, 92262, France

vivian.salino@irsn.fr

ABSTRACT

DRAGON is a deterministic code for neutron transport and, more generally, for nuclear reactor simulation. It is completely open-source from the processing of nuclear data evaluations to the simulation of full cores, and therefore without equivalent in the open world. It is powerful, fast and versatile. However, this flexibility comes at a price: it is an expert code, and its use requires constant learning of the underlying methods. The proposed PyDrag software is a complementary tool, facilitating its use to cover a rather common need: the simulation of pressurized water reactors. Its simplicity derives from the use of the Python language and also a pursuit of the highest minimalism in the interface offered to the user: a simple case can be described in less than 30 lines, based on information directly available in typical industrial or scientific documentation. Potential users are in fields related to nuclear reactor physics, encouraging interdisciplinarity: nuclear data, fuel cycle and reactor physics experts and analysts (fuel, thermal-hydraulics), etc. Like DRAGON, PyDrag is open source and available today.

1. INTRODUCTION: DRAGON, A CODE WITH MANY ADVANTAGES AND CORRESPONDING DRAWBACKS

DRAGON [1,2], continuously developed at École Polytechnique de Montréal since 1981 [3], is a deterministic code for nuclear reactor simulation with many advantages.

1) It delivers state-of-the-art methods [1]. Self-shielding can be treated with equivalence in dilution or subgroups methods that can be supplied with many different types of probability tables or, since recently, with resonance spectrum expansion method. Neutron transport can be solved with P_{ij}, S_N or MoC methods. Leakage models, on top of providing a realistic spectrum in infinite geometries, allows to determine the diffusion coefficient respecting formal equivalence between transport and diffusion. Finally, depletion is treated through a combination of a high order Bateman solver and a saturation model for short-lived isotopes.

2) Being deterministic, DRAGON is a fast code, offering low computation times (an example will be given in section 3) even for relatively complex cases. This is useful for evaluating the difference between two similar situations, where similar deterministic bias can be expected and will cancel out in differences. It turns out to be a very common question: rather than evaluating an absolute result, it is common to wonder whether this or that aspect is important (such as cladding temperature, grain size in MOX fuel, changing the nuclear data evaluation, etc.). The closer the two situations are, the more difficult and costly it is to obtain the answer in Monte-Carlo (due to statistical uncertainties) and even more in full core, coupled or transient

calculations. For a deterministic code, on the other hand, its deterministic bias can be expected to be even more similar for closer situations, therefore gaining precision in computing the differences. For example, negligible effects can be quickly identified and excluded from investigation scopes, where a Monte-Carlo code user will live on with doubts (within the statistical uncertainties). From a safety perspective, it is important to quickly distinguish the significant and insignificant effects (with a low cost for additional questioning) in order to focus and further investigate on the important ones, if necessary with reference tools.

3) DRAGON is an open-source software, down to the nuclear data pipeline that feeds it [4]. Therefore, we are offered the freedom to study the source code, modify it for a specific need and even redistribute it. It allows sharing without legal barriers, enabling smooth international collaboration (for example, within the JEFF community, often hindered by legal restrictions on various codes). To our knowledge, DRAGON is the world's only full-featured open-source deterministic lattice code. In particular, it can handle the difficult self-shielding problem and can therefore cover the full range from nuclear data to full core diffusion.

4) DRAGON is as versatile and flexible as a deterministic code can be. This ambition of universality, present from its very inception, allows its users to address a wide range of cases, such as PWR, PHWR, VVER, FBR (sodium, lead, ...), HTR (including TRISO particles), molten salts, etc. BWR-specific functionalities are currently under development.

But this versatility and flexibility, which stems from a high degree of modularity, comes at a price and with drawbacks corresponding to these advantages. As pointed out by its developers, DRAGON is composed of "collections of independent computational modules that can be linked together […]. The drawback of this approach is that users need to learn the capability to build their own computational schemes to use the code. This requires much more know-how than using competing codes such as CASMO." [3]

For example, its user must learn [5] and know which self-shielding method is most suitable and, depending on that choice, which probability tables should be selected, know what's a slowing down model, how to cylindrize cells and its corresponding boundary approximation, select the normalization of P_{ij} matrices, the order of anisotropy between cells, the leakage model, the depletion solver options, a wide variety of discretizations, etc. Therefore, a level of expertise akin to that of a developer is required.

Also, being an expert code or rather toolbox (in a manner similar to Geant4's toolbox), long debates ensue between users, on this or that approximation or discretization, to reach a consensus on the relevance of better precision at a higher cost (due to engineering time or to calculation time required for an improvement). Conversely, on a more closed code (with strong or unchangeable defaults), such debates cannot take place, which is also both an advantage and a drawback, depending on the circumstances.

Flexibility and modularity also ineluctably imply that input data are presented in a mixed form, where each module can receive both:

- nuclear and technological data relating to the case and

- modelling choices (methods, approximations, discretizations, etc.).

This is easy to manage for a numerical benchmark, simple in geometry and composition, for one single nuclear data evaluation. However, it is much more complicated for more realistic assemblies, while coping with any nuclear data evaluation (different isotopes availability, etc.). For realistic cases, the input data becomes much longer (thousands or tens of thousands of lines, constantly alternating between data and choices of methods) and thus complicated to modify and to adapt to another case. Of course, this is partly due to the complexity of real reactors (as opposed to simplified numerical benchmarks) and to the variability of nuclear data. Monte-Carlo users also suffer from such complexities, albeit less, as they do not have to manage self-shielding of isotopes according to their existence in a nuclear data evaluation (such as elemental zirconium and/or its isotopic breakdown).

Finally, this flexibility also implies the use of a programming language, to provide instructions. In the case of DRAGON, this language is CLE-2000, developed specifically for such needs. It is a complete language, including the declaration of typed variables, conditional branching, loops, etc. The DRAGON user cannot therefore escape learning a specific language, with all its specific features, of which reverse Polish notation is the most emblematic example.

2. PRINCIPLES BEHIND PYDRAG INCEPTION

PyDrag is a Python 3 overlay, a wrapper around DRAGON's Fortran, simplifying its use and improving accessibility, while maintaining Fortran's speed and incidentally keep its massive historic codebase (i.e. not spending one or two decades rewriting it). OpenMC [6] and TRIPOLI-5 [7] have a similar approach: a C++ core for speed and a Python interface for easy, efficient usage. The combination of Serpent [8] and SerpentTools [9] also allows a somewhat similar experience. The visible consensus and ubiquity of Python comes from its ease of access, offered by its proximity to natural language through many English keywords, together with a simple syntax and a dynamic typing.

PyDrag provides a highly desirable separation between methods and data:

(1) Choices of deterministic methods (along with approximations and discretizations) are built in PyDrag, on a first side.

(2) On a second side, technological data (related to the case) can be copied exactly as written in reference documents (whether industrial or scientific). The verbatim aspect simplifies error checking, since no intermediate steps are required. Without such dispositions, users are forced to perform intermediate calculations (such as an isotopic composition spreadsheet) that rarely accompany the dataset itself, at least not in the same file. Comments may help, but since they are not interpreted by the code, they may also be outdated and therefore even more misleading. The information ends up being lost, and the next user no longer has the intermediate step, casting unbridgeable doubt and reducing its reusability, since the initial data can no longer be seen and changed. For instance, users of traditional Monte-Carlo codes often encounter this type of difficulty.

(3) Finally, and on a third side, nuclear data can be managed in a single line, pointing to the desired nuclear data evaluation. In the same spirit as the first point above (1), PyDrag includes rules to relieve the user by coping with the inherent variability of nuclear data evaluations.

PyDrag is intended to provide a single, fixed and unchangeable calculation scheme (see Ref. [10] for details), in a somewhat similar fashion to CASMO [11]. Here however, users wanting more have the freedom to dig into PyDrag's codebase, or use DRAGON directly for utmost flexibility.

Remarkably, many technological data are of little significance or vary only slightly. Like CASMO [11], PyDrag provides reasonable default values for natural abundances, primary circuit pressure, power density, default structural materials (densities and mass percent compositions of stainless steel 304, Inconel 718, Zircalloy 4, M5 alloy, AIC, B_4C, hafnium, air...). Such defaults help the user to focus on the main parameters, which are enrichment in uranium 235, fuel density, burnable poisons, radii, pitches, pin layout, temperatures and boron concentration. By default, water density is computed from its temperature and pressure, using water tables included in DRAGON through implementation of the IAPWS-IF97 industrial standard. Of course, if available, the user may overload any default data, since they are case-specific.

PyDrag benefits from the extensive IRSN experience with DRAGON, including:

- code-to-code comparisons with APOLLO2 and CASMO5 over many cycles of numerous fuel managements, both at assembly and full core scale, including for example gadolinium and MOX fuel [12,13,14],

- code-to-code comparisons with SERPENT, also at assembly and full core scale [15],

- comparisons to experimental measurements of boron critical concentrations, isothermal coefficients, control rods worths, detector responses and power distributions that were performed on BEAVRS [16], Tihange-1, Almaraz-2, Bugey-2, Fessenheim-1 and 2 [15],

- non-regression testing, continuously covering absence of changes for more than a decade on all above cases.

PyGan interface [17], linking Python and CLE-2000, is extensively used to control DRAGON's execution. PyDrag's code is relatively short and of a maintainable size: currently 4000 lines of Python code. For comparison, an equivalent dataset written purely in CLE-2000 language, covering the same usages and cases, is about 18 000 lines long.

The main current limit is its restriction to standard PWRs. Such a restriction is the obvious price of simplification. VVERs could be supported (at least partially) in a not-too-distant future[1]. Finally, all these advantages come with a drawback when it comes to education: because PyDrag is simple, users do not need to learn and understand reactor physics subtleties or the arcane world of nuclear data, although that accessible first approach may encourage them on that path.

3. A SIMPLE PYDRAG USE EXAMPLE

To illustrate its simplicity, a practical example is shown on Figure 1.

[1] Both to provide help for the safety of facilities in Eastern Europe [19] and also to inform European populations in the event of radionuclide dispersion between Central Europe and the Ural Mountains [20,21,22,23].

In line 4, nuclear data library is provided in a format supported by DRAGON [1], such as Draglib format [18,4]. The number of energy groups can be arbitrary: PyDrag will automatically adapt the self-shielding model accordingly to achieve satisfactory results. For production calculation, we currently recommend 295 neutron energy groups [24] and JEFF-3.1.1 nuclear data evaluation [25], which is a conservative choice.

In lines 5 to 9, important material properties are defined, such as fuel density and enrichment, water boron concentrations and temperatures. These can be in Celsius, Kelvin or Fahrenheit, the latter being supported due to its presence in numerous references coming from the USA, where PWR technology was born.

One advantage of Python can be seen in line 5: the user can directly provide an operation instead of a result.

Then, pins are described, alternating materials (as predefined) and radii (in centimeters). In line 11, UO_2 fuel occupies the center up to a radius of 0.4096 cm, enveloped by Zircalloy-4 up to 0.47 cm. Finally, pins are indeed surrounded (implicitly) by borated water.

```
1  # Materials and their temperatures
2  path = ('https://github.com/IRSN/PyNjoy2016/releases'
3          '/download/JEFF-3.x/drglibJEFF-3.1.1')
4  materials = pydrag.Materials(NuclearData = path)
5  materials.UO2.set_density(10.96*0.95) # g/cm3
6  materials.UO2.set_enrichment('U235', 3.7/100)
7  materials.UO2.set_temperature(600, 'C') # Celsius
8  materials.water.set_temperature(286, 'C') # Celsius
9  materials.water.set_boron(600) # ppm
10 # Geometry
11 F = ['UO2', 0.4096, 'Zr4', 0.47] # Fuel pin
12 T = ['water', 0.57, 'Zr4', 0.61] # Empty tube
13 PinLayout = [[T, F, F, T, F, F, T, F, F],
14                [F, F, F, F, F, F, F, F],
15                   [F, F, F, F, F, F, F],
16                      [T, F, F, T, F, F],
17                         [F, F, F, F, F],
18                            [T, F, F, F],
19                               [F, F, F],
20                                  [F, F],
21                                     [F]]
22 geometry = pydrag.Geometry(PinLayout,
23                            PinPitch = 1.26, # cm
24                            AssemblyPitch = 21.5) # cm
25 # Calculation and plot
26 burnup, kinf = pydrag.Deplete(materials, geometry)
27 plt.plot(burnup, kinf)
```

Figure 1: Application case example of PyDrag [26].

These Fuel (F) pins and Tubes (T) are then used to build an assembly, using a pin layout specified here in lines 13 to 21. An assembly can be described as an eighth (default and most common case, like here), a quarter or in full, which are useful options for asymmetric assemblies [27]. To complete the geometry, pin and assembly pitches are given on lines 23 and 24, implying a thin layer of borated water surrounding the assembly.

The calculation itself is triggered on a single line (here, 26), in that case depleting the assembly up until its end of life (by default, 72 000 $MW \cdot d \cdot t^{-1}_{initial\ heavy\ metal}$). The calculation time for this example is two minutes on a standard computer (with a CPU such as Intel® Xeon® E5-2667, clocked at 2.90 GHz).

The purest possible simplicity and even minimalism is achieved here. Not a single piece of information nor a single line could be discarded without greatly affecting the results.

The last line, plotting the results with Matplotlib and producing Figure 2, illustrates the advantages of interfacing with Python: all its power and diversity are at fingertips, connecting a wide range of communities with Python as glue. To give another example, PyDrag calls can be included in an interactive Jupyter Notebook, for easy reworking of outputs or plots without having to run the calculations again.

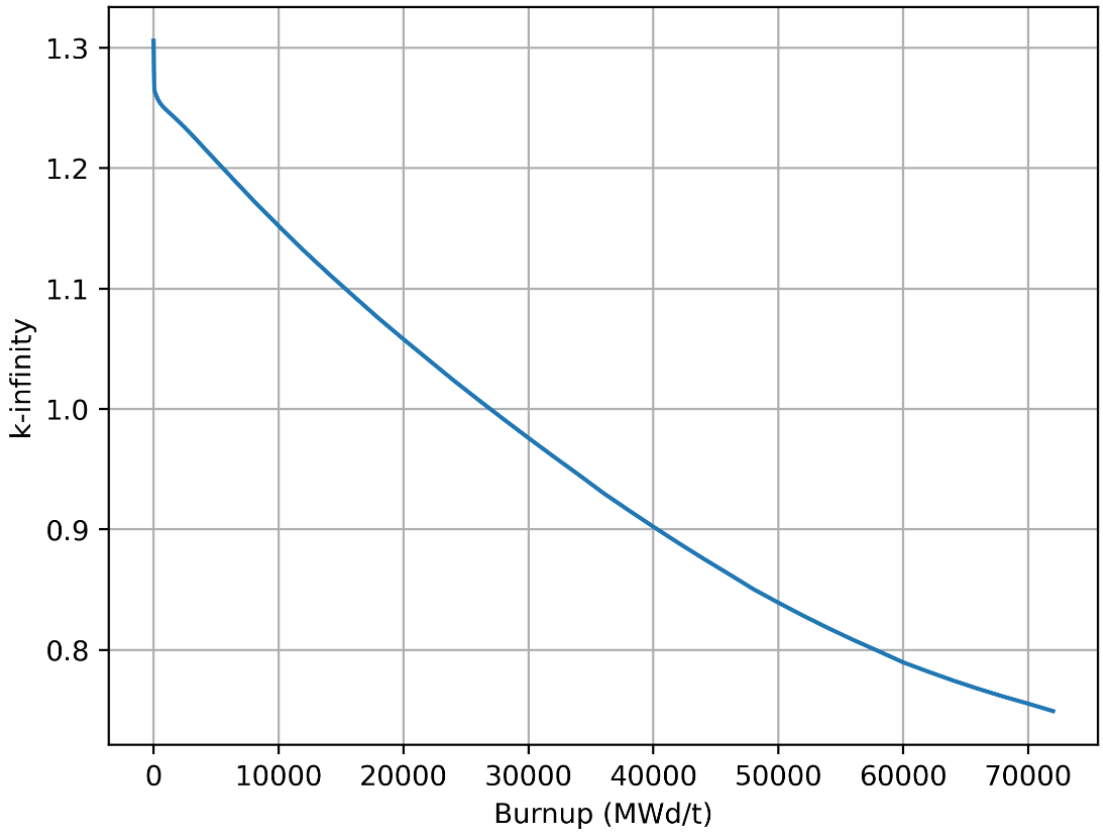

Figure 2: Example of an infinite multiplication factor (k_∞) curve obtained from PyDrag.

Beyond this minimal working example, other important features include grid modelling (axially diluted in water, around pins), thermal expansion (activated by default) and depletion data input (core total power, active height and number of fuel assemblies). More details are available in PyDrag's documentation [26] which is provided through web pages, in line with modern standards.

4. CONCLUSIONS AND PERSPECTIVES

PyDrag is an open-source [26], easy-to-use tool dedicated to nuclear reactor modeling. It leverages the power of Python and DRAGON to deliver an optimized experience to non-expert users. PyDrag and DRAGON are complementary tools, aiming different applications and

specialists: as illustrated in Table 1, the first is dedicated to ease of use, while the latter offers greater flexibility.

Table 1: Recommendations for choosing most adapted tool, between PyDrag and DRAGON.

Is my case a standard PWR?	Yes	No
Do I want to build and experiment my own calculation scheme, with all the benefits and risks included?	No	Yes
Do I want to learn the different ways of computing probability tables for self-shielding models and its consequences? Learn what are Gelbard and Helios normalizations? Selecting cell cylindrization? Etc.	No	Yes
Do I want to understand and deal with the specificities of each nuclear data evaluation?	No	Yes
How do I want to provide the materials' definition?	Material density $[g \cdot cm^{-3}]$	Atomic concentration $[at \cdot b^{-1} \cdot cm^{-1}]$
You should rather use…	PyDrag	DRAGON

Future perspectives include:

- to provide more examples for retrieving data already computed such as reaction rates (for various isotopes, energies, volumes…), neutron flux, homogenized and condensed cross sections, diffusion coefficients, spent fuel composition, decay heat, etc.,

- to extend validation with interpretation of post-irradiation experiments (in particular, comparisons to radiochemical analysis of spent fuel compositions [28]) and more comparisons to reactor physics measurements available in open data such as boron critical concentrations, detector responses, reactivity coefficients, etc. (see Table 2),

Table 2: Reactor physics measurements available as open data.

Type	Reactor	Number of cycles	Total	Ref.	Comments
PWR	Tihange-1	1	16	[29,30]	157 assemblies, 15×15 pin layout
	Fessenheim-2	1		[31,32]	Identical specifications between both, 157 assemblies, 17×17 pin layout
	Almaraz-2	2		[33]	
	Turkey Point-3	3		[34]	Identical specifications between both, 157 assemblies, 15×15 pin layout
	Surry-1	3		[35]	
	Three Miles Island-1	2		[36,37]	177 assemblies, 15×15 pin layout "Cycle 2 poor quality data" [38,39,40]
	Zion-2	2		[41]	193 assemblies, 15×15 pin layout
	BEAVRS (Catawba)	2		[42]	193 assemblies, 17×17 pin layout Large power tilt at first start-up
VVER	X2 (Khmelnitski-2)	4	13	[43,44,45]	
	Kozloduy-2	3		[46]	
	Kalinin-1	3			Identical specifications between both
	Kozloduy-5	3			
BWR	Monticello	3	8	[47]	

	Peach Bottom-2	2		[48]
	Quad Cities-1	3		[49,50]
Total		37		

- to implement automatic comparison functions with OpenMC [6] that could be dynamically launched on the case under consideration (useful only when computationally feasible) for validation of detailed reaction rates [51], which require a full consistency [4] between Draglib multigroup libraries [4] and HDF5 OpenMC libraries [6],

- to add modelling of the radial and axial reflectors,

- to add full core calculations with DONJON code [2],

- to add the functionality to perform dynamic calls to PyNjoy2016 [4] from PyDrag, allowing to provide directly a complete set of files in ENDF-6 format instead of a multigroup Draglib (in line 4 of Figure 1 ; similar functionality exists in OpenMC [6]), in order to facilitate its use by the nuclear data community and enable validation with experimental data (see Ref. [28] and Table 2) as early as possible during the nuclear data evaluation process,

- to add automatic propagation of nuclear data uncertainties [10] through randomly sampled nuclear data [4],

- to prepare a PyPI package for an even easier installation.

5. ACKNOWLEDGMENTS

Our warmest thanks go to Pr. Alain Hébert and Pr. Guy Marleau for their fruitful comments on this article. Together with Pr. Robert Roy, these three people from École Polytechnique de Montréal wrote and released DRAGON, without which PyDrag would not exist. Many thanks to them.

Also, very special thanks to Benjamin Benedet who, as IRSN subcontractor, is a major contributor to PyDrag code development and documentation.

6. REFERENCES

[1] G. Marleau, A. Hébert and R. Roy, "A User Guide for DRAGON Version5", Report IGE-335, École Polytechnique de Montréal, Institut de Génie Nucléaire, 2023. http://merlin.polymtl.ca/downloads/IGE335.pdf

[2] A. Hébert, "DRAGON5 and DONJON5, the contribution of École Polytechnique de Montréal to the SALOME platform", *Annals of Nuclear Energy*, Volume 87, Part 1, Pages 12-20, 2016. https://doi.org/10.1016/j.anucene.2015.02.033

[3] A. Hébert, "Dragon and Donjon: a Legacy Open-Source Reactor Physics Project at Polytechnique Montréal", Proceedings of the *IAEA Technical Meeting on the Development and Application of Open-Source Modelling and Simulation Tools for Nuclear Reactors*, Milano, Italy, June 2022. https://conferences.iaea.org/event/247/contributions/19918/attachments/10630/19919/IAEA_Hebert.pdf

[4] V. Salino and A. Hébert, "PyNjoy2016: an Open Source System for Producing Cross Sections Libraries for DRAGON5 and SERPENT2", Proceedings of *The International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering* (M&C 2023), American Nuclear Society, Niagara Falls, Ontario, Canada, Aug 2023. https://hal.science/irsn-04095482/

[5] A. Hébert, "Applied Reactor Physics", Third Edition, Presses internationales Polytechnique, 2020.

[6] P. Romano, N. Horelik, B. Herman, A. Nelson, B. Forget and K. Smith, "OpenMC: A State-of-the-Art Monte Carlo Code for Research and Development", *Annals of Nuclear Energy*, Volume 82, pages 90-97, 2015. https://doi.org/10.1016/j.anucene.2014.07.048

[7] C. Larmier, A. Jinaphanh, V. Franchini, D. Q. D. Nguyen, F. Malvagi, D. Mancusi and A. Zoia, "Preliminary Code-to-Code Comparisons for the Implementation of Neutron and Photon Physics in the new Monte Carlo Transport Code TRIPOLI-5", Proceedings of *The International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering* (M&C 2023), American Nuclear Society, Niagara Falls, Ontario, Canada, Aug 2023.

[8] J. Leppänen, M. Pusa, T. Viitanen, V. Valtavirta, and T. Kaltiaisenaho, "The Serpent Monte Carlo code: Status, development and applications in 2013", *Annals of Nuclear Energy*, Volume 82, Pages 142-150, 2015. https://doi.org/10.1016/j.anucene.2014.08.024

[9] A. Johnson, D. Kotlyar, S. Terlizzi and G. Ridley, "serpentTools: A Python Package for Expediting Analysis with Serpent", *Nuclear Science and Engineering*, Volume 194, Pages 1016-1024, 2020. https://doi.org/10.1080/00295639.2020.1723992

[10] V. Salino, D. Rochman, E. Dumonteil, F. Malvagi and A. Hébert, "Nuclear Data Uncertainties and Adjustments Using Deterministic and Monte-Carlo Methods along with PWR Measurements", Proceedings of *The International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering* (M&C 2023), American Nuclear Society, Niagara Falls, Ontario, Canada, Aug 2023. https://irsn.hal.science/irsn-04095487/

[11] J. Rhodes, K. Smith and D. Lee, "CASMO-5 development and applications", American Nuclear Society's *Topical Meeting on Reactor Physics* (PHYSOR), Vancouver, BC, Canada, 2006. https://www.researchgate.net/publication/228528270_CASMO-5_development_and_applications

[12] A. Hébert, J. Taforeau, V. Salino and A. Bruneau, "Evaluation des codes neutroniques pour le projet HEMERA V3", IRSN, Technical Report PSN-EXP/SNC/2014-161, 2014.

[13] G. Tixier, "Validation du schéma de calcul assemblage DRAGOR-V1", IRSN, Technical Report PSN-EXP/SNC/2017-178, 2017.

[14] G. Tixier and J.-J. Ingremeau, "Projet ORION : Calcul du cycle à l'équilibre de la gestion combustible GEMMES", IRSN, Technical Report PSN-EXP/SNC/2017-179, 2017.

[15] V. Salino, "Incertitudes et ajustements de données nucléaires au moyen de méthodes déterministes, probabilistes et de mesures effectuées sur des réacteurs à eau sous

pression", Ph.D. thesis, École Polytechnique de Montréal, Canada, 2022. https://publications.polymtl.ca/10545/

[16] J. Taforeau and V. Salino, "Analysis of the MIT BEAVRS Benchmark using the DRAGON-5/PARCS code sequence", Proceedings of *Physics of Reactors 2016 – PHYSOR 2016 – Unifying Theory and Experiments in the 21st Century*, American Nuclear Society, Sun Valley, Idaho, USA, May 2016.

[17] A. Hébert and R. Roy, "The Ganlib5 kernel guide", Report IGE-332, École Polytechnique de Montréal, Institut de Génie Nucléaire, 2023. http://merlin.polymtl.ca/downloads/IGE332.pdf

[18] Draglib download page. http://merlin.polymtl.ca/libraries.htm

[19] D. Verrier *et al.*, "Codes and Methods Improvements for VVER Comprehensive Safety Assessment: The CAMIVVER H2020 Project", Proceedings of the *28th International Conference on Nuclear Engineering (ICONE 28)*, American Society of Mechanical Engineer, Virtual Conference, August 2021. https://doi.org/10.1115/ICONE28-64169

[20] "Report on the IRSN's investigations following the widespread detection of ^{106}Ru in Europe early October 2017", IRSN, January 2018. https://www.irsn.fr/sites/default/files/documents/actualites_presse/actualites/IRSN_Report-on-IRSN-investigations-of-Ru-106-in-Europe-in-october-2017.pdf

[21] O. Masson *et al.*, "Airborne concentrations and chemical considerations of radioactive ruthenium from an undeclared major nuclear release in 2017", *Proceedings of the National Academy of Sciences (PNAS)*, Volume 116 (34), Pages 16750-16759, 2019. https://doi.org/10.1073/pnas.1907571116

[22] "Informations de l'IRSN sur l'incident nucléaire ayant affecté une installation militaire dans la région d'Arkhangelsk (Russie) le 8 août 2019", IRSN, 13th of August, 2019. https://www.irsn.fr/sites/default/files/documents/actualites_presse/actualites/IRSN_Note-Incident-Russie_20190813.pdf

[23] "Détection en Europe du Nord d'une élévation des niveaux de radioactivité dans l'air – mise à jour du 22 juillet 2020", IRSN, 22nd of July, 2020. https://www.irsn.fr/sites/default/files/documents/actualites_presse/actualites/IRSN_NI-2-Detection-Radioactivite-Europe-Nord_22072020.pdf

[24] A. Hébert, "Development of the Subgroup Projection Method for Resonance Self-Shielding Calculations", *Nuclear Science and Engineering*, Volume 162:1, Pages 56-75, 2009. https://doi.org/10.13182/NSE162-56

[25] A. Santamarina *et al.*, "The JEFF-3.1.1 Nuclear Data Library", OECD NEA Data Bank, JEFF Report 22, 2009. https://www.oecd-nea.org/jcms/pl_14470

[26] PyDrag's homepage. https://gitlab.extra.irsn.fr/PyDrag/PyDrag

[27] N.Horelik, B.Herman, B.Forget and K.Smith, "MIT BEAVRS: Benchmark for Evaluation and Validation of Reactor Simulations", Proceedings of *The International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering* (M&C 2013), American Nuclear Society, Sun Valley, Idaho, USA, May 2013.

[28] Michel-Sendis *et al.*, "SFCOMPO-2.0: An OECD NEA database of spent nuclear fuel isotopic assays, reactor design specifications, and operating data", *Annals of Nuclear Energy*, Volume 110, Pages 779-788, 2017. https://doi.org/10.1016/j.anucene.2017.07.022

[29] H. Panek, "Qualification du système Neptune - interprétation d'expériences critiques, calculs de cœur de réacteur de puissance - ébauche d'une nouvelle chaîne de calcul basée sur les codes APOLLO et TORTISE - premiers tests sur cœur de PWR", CEA-N-2092, Ph.D. thesis, Paris-Sud University, 1980. https://inis.iaea.org/collection/NCLCollectionStore/_Public/11/511/11511367.pdf

[30] D. Tournier, "Contribution à la qualification du système Neptune - application au suivi de Tihange", CEA-N-2155, Ph.D. thesis, Paris-Sud University, 1980. https://inis.iaea.org/collection/NCLCollectionStore/_Public/12/578/12578167.pdf

[31] E. Kamha, "Contribution à l'élaboration et à la qualification d'un schéma de calcul pour la gestion des réacteurs PWR, à l'aide du système Neptune - suivi du réacteur Fessenheim 2", Ph.D. thesis, Paris-Sud University, 1981. https://inis.iaea.org/collection/NCLCollectionStore/_Public/18/076/18076909.pdf

[32] A. Hassini, "Établissement de bibliothèques à contre-réactions pour le suivi du 1er cycle du réacteur Fessenheim", Ph.D. thesis, Paris-Sud University, 1982. https://inis.iaea.org/collection/NCLCollectionStore/_Public/14/776/14776891.pdf

[33] "In-core fuel management code package validation for PWRs," IAEA, Technical report IAEA-TECDOC-815, 1995. https://inis.iaea.org/collection/NCLCollectionStore/_Public/26/077/26077395.pdf

[34] "Reactor core physics design and operating data for cycles 1, 2, and 3 of the Turkey Point No. 3 PWR power plant", Technical report EPRI-NP-827, 1978. https://www.osti.gov/biblio/6803027

[35] R. Carlson, "Reactor core physics design and operating data for cycles 1, 2, and 3 of Surry unit 1 PWR power plant", Technical report EPRI-NP-79-2-LD, Georgia Institute of Technology, 1979. https://smartech.gatech.edu/handle/1853/38968

[36] "Reactor core physics design and operating data for cycles 1 and 2 of TMI unit 1 power plant", Technical report EPRI-NP-1410, Babcock and Wilcox, 1980. https://www.osti.gov/biblio/5082638

[37] G. Neely, D. Napolitano et M. Erighin, "Core nuclear design codes and methods qualification", Technical report R003-03-002106, Babcock and Wilcox, 2010. https://www.nrc.gov/docs/ML1024/ML102450302.pdf

[38] A. Ward, "Overview of GENPMAXS/PARCS Capabilities Utilizing SCALE Cross Sections", SCALE Users Group Meeting, September 2017. https://www.ornl.gov/sites/default/files/2_Andrew_Ward.pdf

[39] P. Yarsky, "Recent Progress in Validating POLARIS for USNRC Use", SCALE Users Group Meeting, September 2017. https://www.ornl.gov/sites/default/files/3_Peter_Yarsky.pdf

[40] P. Yarsky, "Use of POLARIS and PARCS to Predict Cycle Depletion for Three Mile Island Unit One Cycles One and Two", Proceedings of *Physics of Reactors 2016 – PHYSOR 2016 – Unifying Theory and Experiments in the 21st Century*, American Nuclear Society, Sun Valley, Idaho, USA, May 2016.

[41] A. J. Impink, Jr. et B. A. Guthrie, III, “Reactor core physics design and operating data for cycles 1 and 2 of the Zion unit 2 PWR power plant”, Carnegie-Mellon University, Technical report EPRI-NP-1232, 1979. https://www.osti.gov/biblio/5303509

[42] N. Horelik, B. Herman, M. Ellis, S. Kumar, J. Liang, B. Forget and K. Smith, “Benchmark or Evaluation And Validation of Reactor Simulations (BEAVRS)”, version 3.0.1, 8th of December, 2020. https://github.com/mit-crpg/BEAVRS/blob/master/beavrs_specifications.pdf

[43] T. Lötsch, V. Khalimonchuk and A. Kuchin, "Proposal of a benchmark for core burnup calculations for a VVER-1000 reactor core", 2009. https://inis.iaea.org/collection/NCLCollectionStore/_Public/41/035/41035568.pdf

[44] T. Lötsch, V. Khalimonchuk and A. Kuchin, "Corrections and additions to the proposal of a benhcmark for core burnup calculations for a VVER-1000 reactor", 2010. https://inis.iaea.org/collection/NCLCollectionStore/_Public/41/131/41131407.pdf

[45] Y. Bilodid, E. Fridman and T. Lötsch, "X2 VVER-1000 benchmark revision: Fresh HZP core state and the reference Monte Carlo solution", Annals of Nuclear Energy, Volume 144, 2020. https://doi.org/10.1016/j.anucene.2020.107558

[46] "In-core fuel management code package validation for WWERs", IAEA, Technical report IAEA-TECDOC-847, 1995. https://inis.iaea.org/collection/NCLCollectionStore/_Public/28/026/28026025.pdf

[47] “Reactor core physics design and operating data for cycles 1, 2, and 3 of the Monticello BWR nuclear generating plant”, Technical report EPRI-NP-472, 1977. https://www.osti.gov/biblio/7209095

[48] N.H. Larsen, “Core design and operating data for Cycles 1 and 2 of Peach Bottom 2”, Technical report EPRI-NP-563, 1978. https://www.osti.gov/biblio/6561294 or https://inis.iaea.org/collection/NCLCollectionStore/_Public/10/449/10449546.pdf

[49] N.H. Larsen, G.R. Parkos and O. Raza, “Core design and operating data for Cycles 1 and 2 of Quad Cities 1”, Technical report EPRI-NP-240, 1976. https://www.osti.gov/biblio/7255696

[50] N.H. Larsen, “Core design and operating data for Quad Cities 1 Cycle 3”, Technical report EPRI-NP-552, 1983. https://www.osti.gov/biblio/6356541

[51] K. Fröhlicher, “Mise en œuvre d’un calcul du cœur BEAVRS en transport 3D cellule par cellule avec le code déterministe APOLLO3”, Master’s thesis, École Polytechnique de Montréal, Canada, 2019. https://publications.polymtl.ca/3926/