csg2csg: A tool to assist Validation & Verification

Andrew Davis[1], **Steven Lilley**[2], **and Jonthan Shimwell**[1]
[1] United Kingdom Atomic Energy Authority
Culham
Abingdon
Oxfordshire
OX14 3DB

[2] ISIS Neutron and Muon Source,
Science and Technology Facilities Council,
Rutherford Appleton Laboratory
Didcot OX11 0QX

andrew.davis@ukaea.uk, steven.lilley@stfc.ac.uk, jonathan.shimwell@ukaea.uk

ABSTRACT

The *csg2csg* tool is a minimal dependency Python3 based program to facilitate the comparison of radiation transport Monte Carlo codes. The *csg2csg* can currently parse MCNP and OpenMC geometry descriptions, and it can subsequently export MCNP, FLUKA, PHITS, OpenMC and Serpent geometry files. This translation is achieved by reading the input geometry into a class based system of objects corresponding to surfaces, cells, materials and so on. The objects are then translated into a 'generic' form. This form once generalised can be converted to any of the supported code formats and then radiation transport calculations can be performed. The code has been tested on several geometries; most notably a complex JET model and a 70k cell ITER C-Model geometry.

KEYWORDS: CSG, convert, neutronics

1. INTRODUCTION

For a number of years the fusion community has been investigating alternative Monte Carlo (MC) codes for use in fusion nuclear analysis. An EFDA study [1] considered several codes for use, but the study stopped short of investigating detailed geometries due to the difficulties in model conversion. The process of validating a new MC code for use depends on a number of factors, but availability of benchmarks is a limiting factor, fortunately for fusion a collection of benchmarks exist, Shielding INtegral Benchmark Archive and Database (SINBAD) [2] is the collection of available experiments. However, due the variable level of agreement seen in SINBAD Fusion it is thought that any code suitable for fusion, must also be able to complete the suite of International Criticality Safety Benchmark Evaluation Project (ICSBEP) [3] benchmarks. Thus, the ultimate target for *csg2csg* is to be able to robustly translate any geometry, however the geometries present in both SINBAD and ICSBEP are the key geometries that must be translatable.

Surface Type	MCNP	Serpent2	FLUKA	OpenMC	PHITS
General Plane	Y	Y	Y	Y	Y
X/Y/Z Plane	Y	Y	Y	Y	Y
X/Y/Z Cylinder	Y	Y	Y	Y	Y
X/Y/Z Sphere	Y	Y	Y	Y	Y
X/Y/Z Cone[1]	Y	Y	Y	Y	Y
X/Y/Z Torus	Y	Y	N	N	Y
General Quadratic	Y	Y	Y	Y	Y
Surface by Points	Y	Y	N	N	Y
Macrobodies[2]	Y	Y	Y	N	Y

Table 1: List of MC codes and their supported surfaces. [1] MCNP definition of sided cones supported by a small subset. [2]The sets of macrobodies that are supported are not equivalent

The MC codes initially targeted for conversion are: MCNP [4], FLUKA [5], OpenMC [6], PHITS [7] and Serpent2 [8]. The reasons for the set of codes is manifold, but this range of codes is thought to stress the range of geometry and material descriptions possible. It should be noted, that there is no attempt to translate source or tally descriptions, it already known that sources will prove near impossible to translate directly.

2. *csg2csg* METHOD

Firstly a survey of the different codes mentioned in section 1 and the number of different surface types were compared to find the common set of supported surfaces amongst the codes, the code by code comparison can be found in Table 1. Most codes support the same sets of basic surfaces, the exceptions are: toroidal surfaces, surface by points and macrobodies. It is clear from the common set of surfaces, that surfaces defined by points (which largely serve as helper functions in MCNP) and macrobodies are the features that must get the most attention. Therefore all surfaces must be expressed as their most basic forms, e.g. an RCC should be converted to a cylinder bounded by two planes.

The concept of cell is common amongst all the codes, all be it with different names (e.g. Region, Cell), they all point to some description of the cell as defined by the surfaces that bound it, that is all that can be described as common between the codes, some codes have the cell indicate material either by name or id, others have no indication on the cell card and hold that information separately. Some MC codes associate their density information with the material description and not the cell description, some include additional information like temperature. In terms of the detail of material description all of the codes are equivalent in the detail of the isotopic description, with the exception of FLUKA which has a multigroup mode wherein only some materials retain their special isotopic descriptions of cross sections as opposed to element averaged. *csg2csg* takes the information contained within the cell/material description and instantiates a new material for each material/density pair, currently temperature is ignored.

3. *csg2csg* CODE STRUCTURE

During the development of *csg2csg* a strong Object Oriented (OO) focus as taken to ensure that there was the appropriate inheritance between classes. For instance, take for example the relationship between the `Card`, `CellCard`, and `MCNPCellCard` classes as shown in Figure 1. The base class `Card` defines the access patterns from objects which inherit from it, in this case `CellCard`. When processing is done `CellCard` contains the generalised description of cell cards. During read time the process which operates on the `MCNPCellCard` class populates the text description of `Card`, and generalises the description of the cell card to Surface objects and Boolean operations.

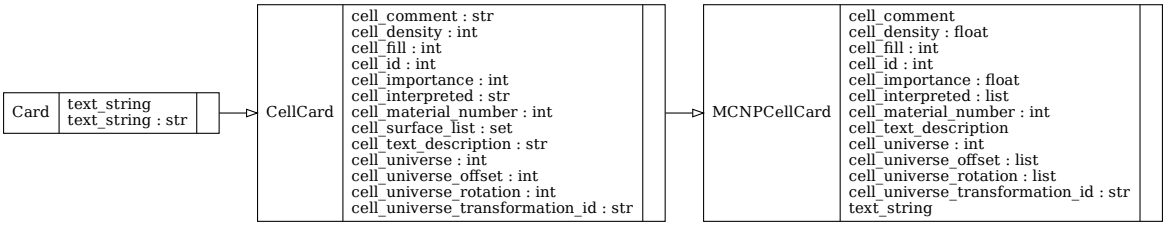

Figure 1: UML inheritance diagram for the `Card`, `CellCard`, and `MCNPCellCard` classes

Similarly the surface cards are processed and sorted in a sequence of classes `Card`, `SurfaceCard`, and `MCNPSurfaceCard`. The surfaces are input initially into their text based description, for example the string "*4 CX 4" in MCNP format refers to a surface with id 4, which is a cylinder lying along the x axis with radius 4 cm also with a reflecting condition. This is stored directly in the `Card.text_description` member variable. When processed the boundary condition information is stored in generic form `Boundary.REFLECTING` and the surface type is processed into generic form, in this case `Surface.CYLINDER_X`. The only special thing of note here is that macrobodies when processed are converted into explicit half space representation, for example a BOX is turned into 6 general planes, with an inside (negative) sense and an outside (positive) sense.

The next stage in programmatic flow is to 'explode nots', this procedure updates the cell descriptions which in MCNP can use the # Boolean operator to negate that cell description. Many MC codes support negation, but not with reference to the cell number, so an explicit description of the negated cell is inserted. In MCNP parlance the description `(3:4:5) #4` would be exploded to form `(3:4:5) #(7 -9 12)`, however in *csg2csg* this is done internally using generic operations and data structures.

The following stage, is to apply explicit surface transformations, MCNP input allows in the surface description an integer representing the translation to apply. All MC codes allow cell transformations but surface transformations are unique to the MCNP taxonomic group (MCNP, PHITS, MARS) . The procedure to allow this feature to be fed into other MC codes is to explicitly transform the surfaces, this is done following [9] but is enumerated here due to some implementation differences. First, the surface is turned explicitly from its simplified form `surface.CYLINDER_X` into `surface.GENERAL_QUADRATIC`. Then the rotation matrix as defined by the transform

card can be applied.

First, the surface coefficients are instanced into A and the rotation coefficient into B.

$$A = \begin{bmatrix} k & g/2 & h/2 & j/2 \\ g/2 & a & d/2 & f/2 \\ h/2 & d/2 & b & e/2 \\ j/2 & f/2 & e/2 & c \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & v1_i & v1_j & v1_k \\ 0 & v2_i & v2_j & v2_k \\ 0 & v3_i & v3_j & v3_k \end{bmatrix} \tag{1}$$

Thus the resultant rotated surface equation can be described as in Equation 2.

$$R_1 = B^T A B \tag{2}$$

The result of Equation 2 is the fully rotated form of the general quadratic form of the surface, however this does not include the appropriate Cartesian shift. We must now factor in the shift and its appropriate rotation, we do this constructing a unitary diagonal matrix, which includes the Cartesian shift and the unit directions.

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ dx & 1 & 0 & 0 \\ dy & 0 & 1 & 0 \\ dz & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The final fully rotated and shifted form is described by Equation 4.

$$R_3 = C^T R_1 C \tag{4}$$

Thus the contents of R_3 now contain the fully rotated and translated surface. The reason for the deviation relative to [9] is that it was found that doing the compound rotation and translation in a single stage resulted in the incorrect surface description.
The last few remaining core functions are largely book keeping;

1. ensure that there is a unique material object for each material number/density pair - most MC codes do not support MCNP's shorthand
2. define all universe translations as a 3 vector matrix rotation form
3. apply cell based boundary conditions to the surfaces which belong to that cell - e.g. some MC codes associate importance 0 regions with surfaces
4. generate bounding boxes for all the surfaces in the problem - this is to support translation to Serpent2. Serpent2 does not support the union operator in the outermost cell - so *csg2csg* generates an additional cell bounded by a sphere to ensure the outermost cell is simple

4. EXAMPLES

Without showing detailed validation of MC codes, which is beyond the scope of this paper, it is difficult to prove the validity or utility of the code. Included are a number of pictorial examples produced by the code, in these cases they were chosen as they either demonstrate complexity in terms of the cell descriptions translated (JET), shear number of cells and surfaces (C-Model) and ESS.

4.1. JET Model

The JET model is the work of some 10's of people over the last 20 years. It represents the complete torus hall of the JET experiment, including the full torus, external heating systems, and diagnostics, slices are shown in Figure 2

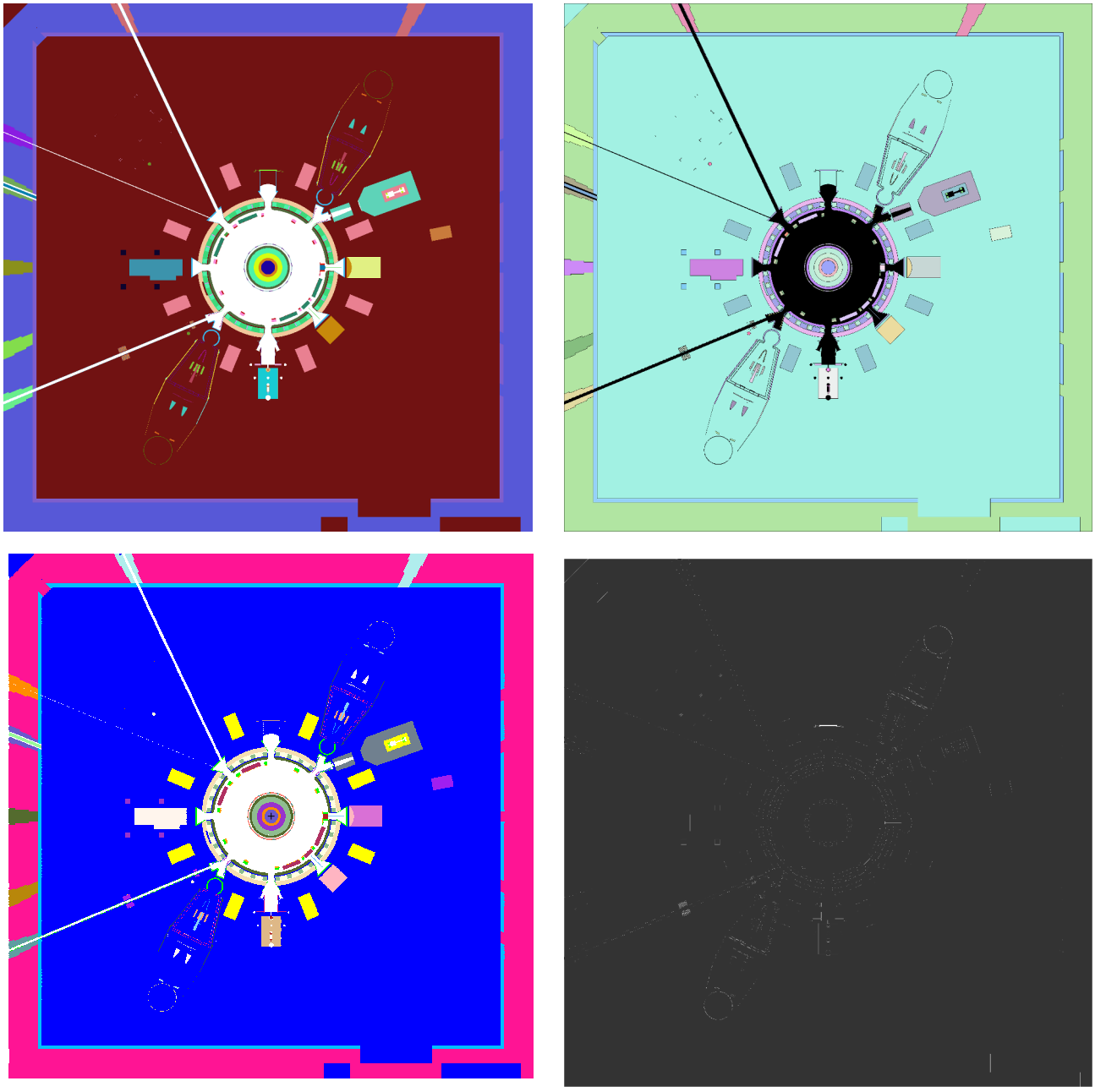

Figure 2: Plot slices of the JET geometry showing OpenMC, Serpent2 and MCNP plots and the diff (black no difference - white difference) of the Serpent2 and OpenMC images respectively (z = 0 cm)

Due to its varied history its an excellent test case to show the range of combinations of methods of defining cells, universes surfaces containing; mixtures of surfaces and macrobodies, cells defined with large numbers of unions and complements, a number of different universe specifications and an extensive collection of materials. This model is geometrically more complex that the ITER C-Model, in that ITER has strict limits on the number of unions, parentheses and nested universes that are allowed.

As can be seen qualitatively in Figure 2, the geometry can be represented in each of the MC codes demonstrated. This is further confirmed by the diff of Serpent2 and OpenMC geometries, where only single pixel cell boundary differences can be seen, here black indicates no difference and white indicates a difference. This diff plot is enabled by using the same RGB values for each unique material, and the same query resolution for the geometry, then these plots can easily *diffed*. Single pixel diffs must be tolerated since the method by which each MC determines what spatial coordinate to query varies, one could either stride through the mesh in some order and the centroid of each mesh element is used to determine the spatial coordinate, alternatively one could start at the boundary of the mesh and ray trace across boundaries and use the centroid of the ray, certainly some codes perform some numerical tolerance 'bumping' to ensure that no queries occur directly on cell boundaries.

4.2. C-Model

The C-Model geometry (C-Model V1 R2.1 - 15/12/2016) is a large and complex MCNP input that describes the ITER reference analysis geometry, it represents the investment of several person years of effort to produce and involves the work of most of the European fusion laboratories. It has several layers of nested universes, large complex multi-union base level universe descriptions, with some 108454 surfaces and 70374 cells. A model of this complexity has only been possible in the last decade with the progress of CAD translation tools like SuperMC & MCAD.

The same diff method as used in Section cannot be used in this case since the C-Model geometry has more than 256 unique material/density combinations, Serpent2 cannot currently reproduce this many colours and thus the diff becomes meaningless, but the geometry slices are shown in Figure 3. A diff of OpenMC & MCNP was produced and showed similar agreement to the JET model, i.e. differences only on surfaces.

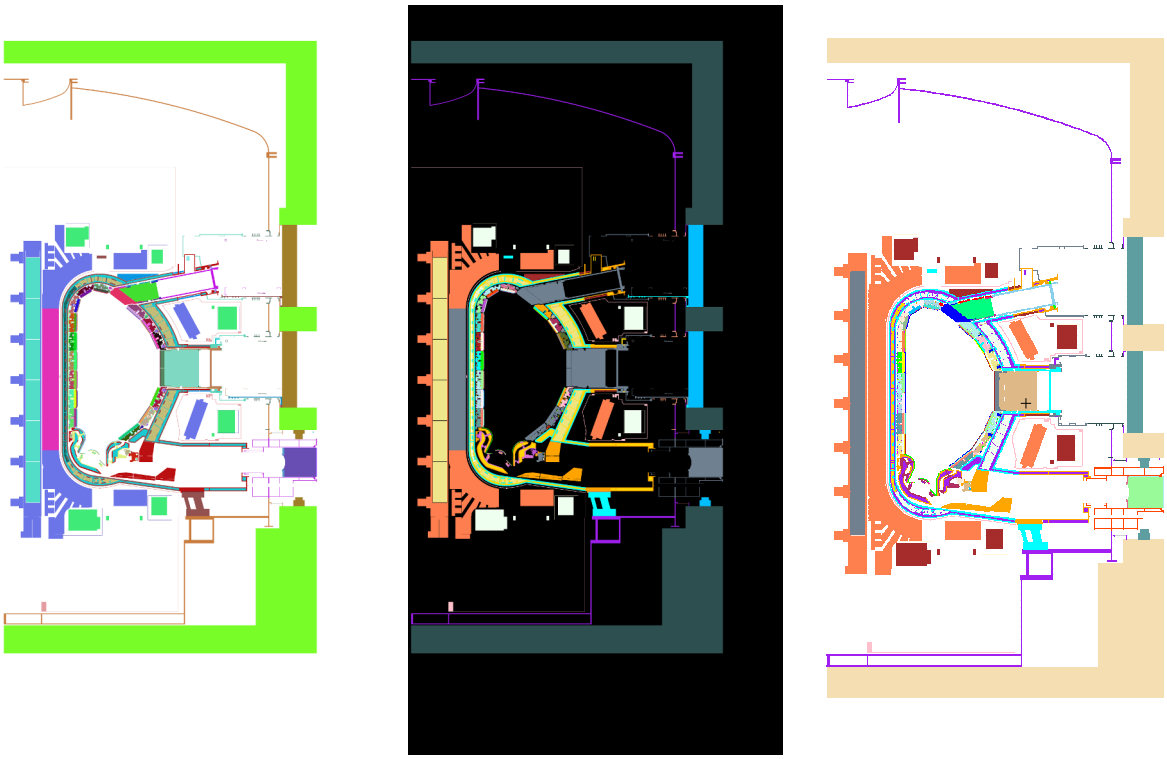

Figure 3: Plot slices of the C Model geometry showing OpenMC, and Serpent2 images respectively (y = 2 cm)

5. CONCLUSIONS

The *csg2csg* tool has been developed for the purpose of aiding radiation transport benchmarking, facilitating cross code comparison and helping to reduce monoculture. The sister paper to this, "Validation of OpenMC for a range of fusion benchmarks" made broad use of *csg2csg* to convert the geometries in question. Furthermore, its ability to translate complex geometry correctly as demonstrated qualitatively in Section 4 for the JET and ITER geometry cases.

In terms of missing features there are a number of near term goals;

- Support for hierarchical geometry formats like ROOT [10] or Geant4 [11] would open up the use of Geant4 on routine fusion problems and benchmarks. This feature is not thought to be straightforward or performant due to the key differences between the half space based geometry inherited from MCNP geometry and the hierarchical manifold volume based geometry of ROOT/Geant4.
- Some MC codes do not support the Boolean union operator, but instead do so using some implicit ordering of the cells. In order to support those codes the cell description must be refactored into a form which allows trivial splitting of the unions into many small manifold regions. It is thought that this will be done through expansion of the Boolean expressions and the laws of Boolean arithmetic.

ACKNOWLEDGEMENTS

The authors would like to thank Jonathan Naish for the supply of the JET model for testing & conversion.

This work was partially funded under Horizon 2020 grant 800999 as part of the SAGE2 grant. This work was supported by EPSRC Grant EP/T012250/1.

REFERENCES

[1] A. Davis. "Validation report of a subset of available monte carlo codes using ENDF/B VIIR0 data." Technical Report WP12-DTM04-T11-D12, EFDA (2012).

[2] I. Kodeli, A. Milocco, P. Ortego, and E. Sartori. "20 years of SINBAD (Shielding integral benchmark archive and database)." *Prog Nucl Sci Technol*, **volume 4** (2014).

[3] J. B. Briggs, L. Scott, and A. Nouri. "The International Criticality Safety Benchmark Evaluation Project." *Nuclear Science and Engineering*, **volume 145**(1), pp. 1–10 (2003).

[4] C.J. Werner, et al. ""MCNP6.2 Release Notes"." Technical Report LA-UR-18-20808.

[5] T. Böhlen, F. Cerutti, M. Chin, A. Fassò, A. Ferrari, P. Ortega, A. Mairani, P. Sala, G. Smirnov, and V. Vlachoudis. "The FLUKA Code: Developments and Challenges for High Energy and Medical Applications." *Nuclear Data Sheets*, **volume 120**, pp. 211 – 214 (2014). URL http://www.sciencedirect.com/science/article/pii/S0090375214005018.

[6] P. K. Romano and B. Forget. "The OpenMC Monte Carlo particle transport code." *Annals of Nuclear Energy*, **volume 51**, pp. 274 – 281 (2013). URL http://www.sciencedirect.com/science/article/pii/S0306454912003283.

[7] H. IWASE, K. NIITA, and T. NAKAMURA. "Development of General-Purpose Particle and Heavy Ion Transport Monte Carlo Code." *Journal of Nuclear Science and Technology*, **volume 39**(11), pp. 1142–1151 (2002). URL https://doi.org/10.1080/18811248.2002.9715305.

[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**, pp. 142 – 150 (2015). URL http://www.sciencedirect.com/science/article/pii/S0306454914004095. Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013, SNA + MC 2013. Pluri- and Trans-disciplinarity, Towards New Modeling and Numerical Simulation Paradigms.

[9] J. Durkee. "MCNP Geometry Transformation and Plotter Equations." *Progress in Nuclear Energy*, **volume 61**, pp. 26–40 (2012).

[10] M. B. I. Antcheva and B. B. et al. "ROOT — A C++ framework for petabyte data storage, statistical analysis and visualization." *Computer Physics Communications*, **volume 180**(12), pp. 2499 – 2512 (2009). URL http://www.sciencedirect.com/science/article/pii/S0010465509002550. 40 YEARS OF CPC: A celebratory issue focused on quality software for high performance, grid and novel computing architectures.

[11] M. Asai, A. Dotti, M. Verderi, and D. H. Wright. "Recent developments in Geant4." *Annals of Nuclear Energy*, **volume 82**, pp. 19 – 28 (2015). URL http://www.sciencedirect.com/science/article/pii/S030645491400406X. Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013, SNA + MC 2013. Pluri- and Trans-disciplinarity, Towards New Modeling and Numerical Simulation Paradigms.