Annals of Nuclear Energy 114 (2018) 329–341

Contents lists available at ScienceDirect

Annals of Nuclear Energy

journal homepage: www.elsevier.com/locate/anucene

CAD modeling study on FLUKA and OpenMC for accelerator driven system simulation

Jin-Yang Li [a,b,*], Long Gu [a,*], Hu-Shan Xu [a], Nadezda Korepanova [a,b], Rui Yu [a], Yan-Lei Zhu [a], Chang-Ping Qin [a,b]

[a] Institute of Modern Physics, Chinese Academy of Sciences, Lanzhou 730000, China
[b] University of Chinese Academy of Sciences, Beijing 100049, China

ARTICLE INFO

Article history:
Received 9 June 2017
Received in revised form 16 December 2017
Accepted 20 December 2017

Keywords:
ADS
FLUKA
FreeCAD
OpenMC
Ray-casting technology

ABSTRACT

Accelerator driven system (ADS) is an advanced nuclear system with high inherent safety feature, which can transmute the nuclear waste including minor actinides (MA) and long-lived fission products (LLFPs) into short-lived fission products. In order to accomplish the neutronic simulation in ADS, a conventional way can be divided into two steps, namely the physical process of proton beam bombarding spallation target (FLUKA) and the neutron transport process in subcritical reactor coupling with a spallation neutron source (OpenMC). However, there are some difficulties in the modeling process for ADS, such as the inconsistent modeling process between FLUKA and OpenMC codes, and the error-prone process of constructing complex geometries. Therefore, it is imperative to build a code system that can convert the CAD modeling files into the input cards of Monte Carlo codes. In this context, a code system named CAD-PSFO (FreeCAD based parsing script for FLUKA and OpenMC) has been developed to solve the modeling conversion problem in ADS, in which basic geometry and boolean logic classes have been established with a mapping relationship to FLUKA and OpenMC codes, and ray-casting technology based iteration method has been proposed to solve the problem of higher-order surfaces in complex geometries. Finally, CAD-PSFO has been verified and validated by comparing with a reference model, two benchmarks, and two simulation tests.

1. Introduction

Accelerator driven system (ADS), mainly constructed from three parts (accelerator, spallation target, and subcritical reactor), is one of the most promising nuclear waste transmutation facilities in a near future. Many countries in the world have drawn up special plans for ADS, such as ATW project in the USA (Lawrence et al., 2001), OMEGA project in Japan, and MYRRHA project in Europe (Kurata et al., 2002; Abderrahim et al., 2001). Moreover, the first ADS prototype experimental facility in the world is expected to be established in the year of 2023 under the project of China initiative accelerator driven system (CIADS) (Zhan and Xu, 2012).

It is difficult to simulate the physical process of ADS since it should couple the process of spallation reaction with the neutronic calculation of subcritical reactor. In order to study the behavior of the multi-physics process of ADS, the simulation should be divided into two procedures. The first procedure is simulating high energy protons bombard spallation target, and the second one is subcritical neutron transport coupling with an additional spallation neutron source, which is acquired from the first procedure (Kapoor, 2002). In this study, two Monte Carlo codes FLUKA (Battistoni and Cerutti, 2006) and OpenMC (Romano et al., 2015) have been selected since these two codes are open-source codes, and the coupling them together has a potential to deal with ADS (Li et al., 2017).

Most Monte Carlo codes typically provide native input languages for specifying the spatial domain model or geometry in terms of various boolean combinations of geometric surface or volume primitives. However, it is a challenge to describe and verify the complex three-dimensional geometries by means of native input languages. Moreover, it has been long recognized as an error-prone process (Song et al., 2014). To solve this problem, two approaches have always been taken into consideration. The first approach is translating CAD export files into the native input languages of Monte Carlo codes. The second one is using CAD model directly for Monte Carlo analysis (Wu et al., 2015; Li et al., 2007). In this work, the first approach has been employed, since

* Corresponding authors at: Institute of Modern Physics, Chinese Academy of Sciences, Lanzhou 730000, China (J.-Y. Li).
E-mail addresses: lijinyang@impcas.ac.cn (J.-Y. Li), gulong@impcas.ac.cn (L. Gu).

https://doi.org/10.1016/j.anucene.2017.12.050
0306-4549/

it is a flexible way to adapt to the modeling process without modifying the original codes of FLUKA or OpenMC.

For the modeling process in FLUKA code, there is an optional interactive auxiliary modeling tool for setting up the corresponding input cards. SimpleGeo (Theis et al., 2006, 2011), which developed based on CAD technology, can easily create models via the operations of dragging and dropping. But it has some limitations in describing of higher-order surfaces, and its functions cannot be completely compared with mature CAD modeling packages. On the other hand, as far as we know, there are no optional tools in modeling process for OpenMC code (latest version v0.9.0). Thus, it is imperative to develop an interactive auxiliary modeling tool for FLUKA and OpenMC codes to solve the complex modeling problems in simulation of ADS.

In the context of demand for modeling tool, a compatible interactive modeling program called CAD-PSFO (FreeCAD based parsing script for FLUKA and OpenMC) has been developed. The details of the methods are described in Section 2, which include building of the specification tree-based basic geometry and boolean logic classes, setting up the corresponding mapping relationship to FLUKA and OpenMC codes, and employing ray-casting technology based iteration method to solve the problem of complex models with higher-order surfaces. The verification and application of CAD-PSFO are discussed in Section 3. Finally, conclusions are drawn in Section 4.

2. CAD-PSFO method

In this paper, CAD-PSFO is developed based on the FreeCAD package (Gayer and O'Sullivan, 2016), which is a parametric three-dimensional modeler developed primarily to design real-life objects of any size. The FreeCAD package is open-source, highly customizable, scriptable, extensible, and suitable for secondary development. For most mature CAD modeling packages, the detailed historical procedures for constructing models are stored in a specification tree. The specification tree in FreeCAD code is a multi-way tree structure, which shows a hierarchical view of the project or the state of current task as presented in the left column window of Fig. 1 tagged with "Combo View". All the topology information is included in the specification tree, which has mapping relationship with the graphical user interface in the right column window of Fig. 1, such as the information of constructed solid geometries (CSG) (Requicha and Voelcker, 1983) and the boolean relationship between them.

Fig. 2 is the node expression of the specification tree shown in Fig. 1, which has been split into 5 levels with an index from 0 to 4. In this specification tree, the first level (level-0) is the root node, and each branch path has a node marked with different colors, which have different properties. The blue one represents basic geometry node, where basic geometries are defined as a series of default parameterized solid bodies in CAD-PSFO, including sphere, cylinder, cube and many complicated models listed in Fig. 4. The red one represents logic node including all boolean logic operators listed in Fig. 5, and the black one indicates group node, which is used to store the mutually independent models constructed from basic geometries using boolean operators.

The method of CAD-PSFO is based on the specification tree that simplified code structure is shown in Fig. 3. The CAD-PSFO code can access the data of FreeCAD via the application programming interface (API), and extract all the information from the specification tree. Subsequently, a loop procedure will be used to split the reference model into several groups based on the node expression of this information, where reference model is the complicated model to be studied and groups are mutually independent models depicted in Fig. 2. Each of these groups contains a hierarchical structure of boolean relationship of CSG models, which can be found in the specification tree in Fig. 1 with the index name "Group". Two methods have been proposed to analyze these groups based on whether all CSG models are basic geometries or not. For all models that are completely composed of basic geometry, a basic geometry and boolean logic conversion method is employed, which will be described in Section 2.1. And for complex geometries containing higher-order surfaces, a ray-casting iteration method is introduced (B-rep to CSG), which will be explained in Section 2.2. Finally, conversion interface will be used to organize the output information from the two methods, and to set up the input files for FLUKA and OpenMC codes.

2.1. Basic geometry and boolean logic conversion method

FreeCAD natively supports concepts like B-rep, non-uniform rational B-spline curves and surfaces, a wide range of geometric entities and boolean operations (Gayer and O'Sullivan, 2016). However, the geometric primitives in FLUKA are described by means of solid bodies based on an improved version of the well-known Combinatorial Geometry (CG) package, which can handle even very complex geometries (Battistoni and Cerutti, 2006). OpenMC uses surface half-spaces based on a technique known as CSG to build arbitrarily complex three-dimensional models in Euclidean space (Romano et al., 2015). In order to translate the FreeCAD modeling information into the native languages of FLUKA and OpenMC codes, an intermediate interface should be established to determine what basic geometries and operations are included, and how to set up the mapping relationship to the two Monte Carlo codes. Therefore, two kinds of classes have been proposed in CAD-PSFO code, namely Basic Geometry Classes (BGC) and Boolean Logical Classes (BLC). And a conversion interface has been developed for the translation process.

As for the basic geometry classes (BGC), a hierarchical structure of models has been established as shown in Fig. 4. In Fig. 4, the top graph layer classes are derived from CAD-PSFO basic class including three types of basic geometry classes, namely sphere, cylinder and cuboid. In the second layer, truncated right angle cone and right prism classes are derived from cylinder class, where the right angle cone is a special case included in the truncated right angle, and $n(n \geqslant 3)$ is the number of polygon edges of the upper surface for right prism. The example given in Fig. 4 is the case of $n = 6$. Furthermore, in the third layer, truncated polyhedral cone classes are derived from truncated right cone class, where the polyhedral cone is a special cases included in it, and $n(n \geqslant 3)$ is the number of polygon edges of the upper surface for truncated polyhedral cone.

Fig. 5 shows the boolean logic classes (BLC), which includes the boolean operation class and the logical operation class. The former one consists of the method of union ("+"), subtraction ("−"), and intersection ("∗"), upon which the required model can be constructed with the basic geometry primitives as mentioned above. Another type of class with four methods and one property in the lower level of Fig. 5 is the logical operation class, which can handle the manipulating process to basic geometries and define the corresponding material properties. This logical operation class provides a looping mechanism for dealing with repetitive cell structures. The four methods in logical operation class are the operations of moving, rotating, zooming and mirroring. And one property indicates that a hierarchical data file (HDF Folk and Heber, 2011) with the information of nuclides densities, corresponding composition and temperature stored in it. In CAD-PSFO code, all the basic geometrical models are generated based on the same origin in the global coordinate. Then, these models in the same group will be moved to its real position using logic operation class, and reassembled by means of boolean operation class.