frontiers | Frontiers in Energy Research

TYPE Technology and Code
PUBLISHED 31 October 2024
DOI 10.3389/fenrg.2024.1496523

OPEN ACCESS

EDITED BY
Gabriele Croci,
University of Milano-Bicocca, Italy

REVIEWED BY
Zhang Chunyu,
Sun Yat-sen University, China
Antonio Jiménez-Carrascosa,
Paul Scherrer Institut (PSI), Switzerland

*CORRESPONDENCE
Zhaoyuan Liu,
liuzy7@mail.tsinghua.edu.cn

RECEIVED 14 September 2024
ACCEPTED 23 October 2024
PUBLISHED 31 October 2024

CITATION
Wang W, Jia C, Liu Z and Wang K (2024) Design and implementation of HDF5-format neutron nuclear database in RMC code. *Front. Energy Res.* 12:1496523. doi: 10.3389/fenrg.2024.1496523

Design and implementation of HDF5-format neutron nuclear database in RMC code

Wu Wang, Conglong Jia, Zhaoyuan Liu* and Kan Wang

Reactor Engineering Computational Analysis Laboratory, Department of Engineering Physics, Tsinghua University, Beijing, China

This study developed a new structured nuclear database to improve the readability and extensibility of the ACE (A Compact ENDF) nuclear database, save disk space, reduce memory usage, and enhance the computational efficiency of the Reactor Monte Carlo (RMC) code. A Python package was developed to store nuclear data in HDF5 format. Compared to the ACE database, the HDF5-format database shows significant improvements: an 80% reduction in disk space for continuous energy neutron data and a 60% reduction for neutron thermal scattering data. The HDF5-format database was implemented in RMC's criticality calculation mode and validated through VERA benchmark problem 2B, demonstrating perfect agreement with the ACE results in k_{eff} and neutron flux counts. The Computational results indicate a 7.7% reduction in memory usage and a 20.1% improvement in computational efficiency with the HDF5-format database. Additional tests show that using the database at a single temperature point reduces memory usage by 5.4% and running time by 13.2%. At two temperature points, memory usage decreases by 35.2% and running time by 18.0%. The new data structure reduces temperature-independent redundant data and improves indexing efficiency, leading to greater savings with more temperature points. This development enhances performance of criticality calculation of RMC and addresses ACE database limitations.

KEYWORDS

nuclear database, RMC, HDF5, ACE, VERA

1 Introduction

With the rapid development of supercomputers and the refinement of Evaluated Nuclear Data Files (ENDF), the Monte Carlo (MC) method is increasingly being used for particle transport simulations and is considered the most accurate approach for solving particle transport problems. The high computational accuracy achieved by the Monte Carlo method can be attributed to its sophisticated geometric modeling capabilities and the use of a large number of particle histories. Additionally, the availability of sufficiently detailed nuclear databases plays a crucial role. Most Monte Carlo particle transport simulation software currently relies on the ACE nuclear data developed by Los Alamos National Laboratory (LANL). The ACE was initially designed for the MCNP (Monte Carlo N-Particle) code (Forster and Godfrey, 2006) developed by the U.S. Department of Energy. Subsequently, numerous MC programs such as Geant4 (Agostinelli et al., 2003), Fluka (Battistoni et al., 2016), Serpent (Leppänen, 2013), and RMC (Wang et al., 2015) have also adopted the ACE for their nuclear databases.

The ACE nuclear database contains various data related to simulating nuclear reactions, including microscopic cross-sections for all possible reactions, secondary particle yields, angular and energy distributions, and so on. These data are typically derived from the ENDF evaluated nuclear data files (Trkov and Brown, 2018). However, ENDF data cannot be directly used in MC programs and must undergo processing to enable sampling of reaction channels and secondary particle distributions. The NJOY program (Macfarlane et al., 2017) is commonly used to process ENDF-formatted nuclear data into the ACE. In the ACE, the large amounts of nuclear reaction data and their index data are stored in three one-dimensional arrays (NXS, JXS, and XSS). MC programs require complex indexing based on the fixed ACE (Conlin and Romano, 2019; Sweezy et al., 2003) to access the various nuclear data stored within the database. Due to the large amount of data, it is impractical to review or extend the nuclear data. Taking the example of the ACE continuous-energy neutron nuclear database for 235U at 293.6K, which is processed from ENDF-B/VIII.0 and provided on the LANL nuclear data website, it contains as many as 76,027 main energy grid points, and the entire XSS array has a staggering 7,168,374 data points, making it practically legible and non-extensible. However, with the continuous development of Monte Carlo software capabilities, there is a growing need for nuclear data that is not available in existing ACE nuclear databases. For instance, energy release data for fine-grained energy deposition and covariance data for sensitivity and uncertainty analysis are required but not present in ACE nuclear data. As a consequence, the lack of extensibility in ACE nuclear database necessitates the creation of separate databases to incorporate the required additional nuclear data. This fragmentation and lack of centralization present challenges in effectively managing and organizing nuclear data resources.

To make it easier for researchers to understand the nuclear database, the Working Party on International Nuclear Data Evaluation Cooperation (WPEC) has proposed a new, simplified, and universally applicable nuclear data structure called General Nuclear Data Structure (GNDS) (Mattoon et al., 2012). The GNDS defines only the structure required for storing nuclear data and the types of data to be stored, without specifying how the data is stored in a file. The GNDS hierarchical structure can be stored in any format that supports nested data structures, such as XML, JSON, or HDF5. Inspired by the GNDS, Paul Romano et al. have developed a well-structured HDF5-format nuclear database based on ENDF and ACE and applied it to the OpenMC code (Romano and Harper, 2017). The HDF5-format nuclear database developed in this study features a more detailed hierarchical division of secondary particle energy and angle distributions, enhancing the overall readability of the data. Additionally, a thorough analysis of computational efficiency and memory usage is provided, with a comparison to the ACE database.

The objective of this study is to design and manufacture a nuclear database with a clear hierarchical structure that enables readability of any nuclear data, facilitates easy data extension, and is compatible with the RMC code to save memory and improve efficiency. In this study, a Python code package was developed to parse the ACE nuclear databases and written nuclear data into HDF5-format files according to the new data structure designed in this study. The manufacturing process of the HDF5-format nuclear database is described in Section 2.1. Sections 2.2, 2.3 present the design of the new structures specifically tailored for continuous-energy neutron and thermal nuclear data, respectively. Section 3 provides a description of the application development of the HDF5-format nuclear database within the criticality calculation mode of RMC. Section 4 conducted tests and validations on the storage space requirements of the HDF5-format neutron nuclear database, as well as on the correctness, memory usage, and computational efficiency of the criticality calculations performed by RCM using the HDF5-format neutron nuclear database.

2 HDF5-format neutron nuclear database design and manufacture

This study presents a Python-based framework for constructing HDF5-format nuclear databases. The developed code package enables complete parsing of ACE nuclear databases, effectively translating ACE nuclear data into an in-memory hierarchy of Python objects. With this package, the generation of HDF5-format nuclear data files is facilitated, ensuring an identical hierarchical structure as the Python objects. The framework consists of the following two components: (1) Xsdir file parsing module: This module is responsible for extracting crucial parameters required for parsing ACE nuclear databases from the xsdir file. (2) ACE nuclear data parsing module: This module focuses on parsing the nuclear data within ACE nuclear databases and generating the HDF5-format nuclear database. It processes the data files, extracts the required information, and organizes it into HDF5-format files according to the designed hierarchical structure.

The subsequent sections will begin by presenting an overview of the comprehensive steps involved in constructing an HDF5-format nuclear database. Subsequently, the data structure of the HDF5-format continuous energy neutron and HDF5-format thermal neutron scattering nuclear databases will be elucidated based on the data structure of Python class objects.

2.1 Manufacturing process

The ACE nuclear database consists of a database directory file called "xsdir" and various nuclear data files. Each nuclear data file may contain one or more nuclear data tables. The xsdir file contains crucial parameters for guiding the parsing of each nuclear data table. Each nuclear data table corresponds to a specific data table identifier in the xsdir file. Taking the example of the continuous-energy neutron ACE nuclear database processed from ENDF-B/VIII.0, provided by the LANL Nuclear Data website, the data table identifiers "92235.00c" and "92235.01c" represent the continuous-energy neutron nuclear data tables for 235U at temperatures of 293.6 K and 600 K, respectively.

The manufacturing flow of the HDF5-format nuclear database is illustrated in Figure 1. First, the xsdir file is obtained by taking the user input of the xsdir file path. Within the `XSDIRparser` class, the `Parser()` function and the `Format()` function are constructed to parse and format the xsdir file into a nested dictionary

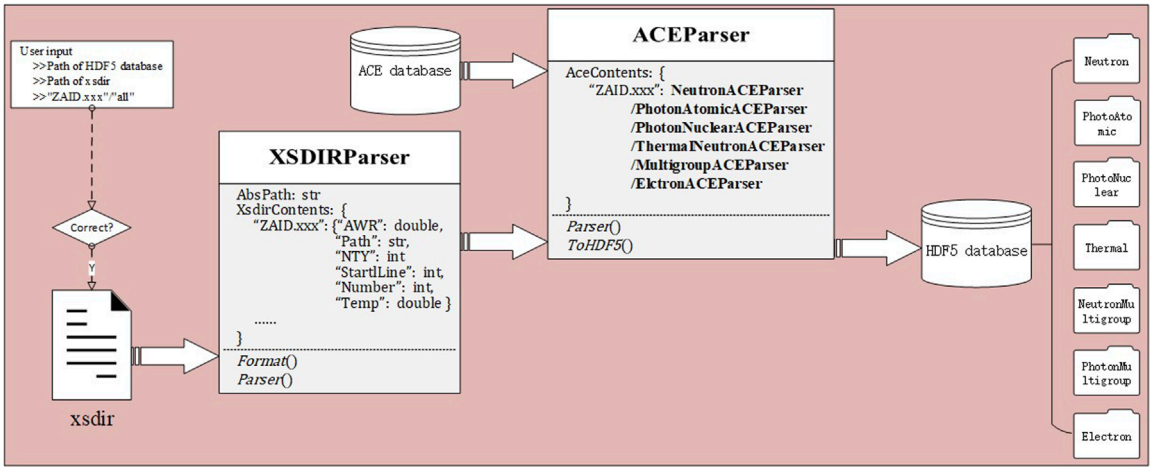

FIGURE 1
The manufacturing flow of the HDF5-format nuclear database.

called XsdirContents which contain the crucial parameters required for guiding the parsing of the ACE nuclear database. In the XsdirContents, the data table identifiers are represented by strings in the format "ZAID.xxx". The parsing parameters include the atomic weight ratio (AWR) of the nuclide, the relative path (Path) to the data file containing the data table, the starting line (StartLine) of the nuclear data in the data file, the number of data points (Number), and the temperature parameter (Temp) for continuous-energy neutron and thermal nuclear databases. Next, the user-provided data table identifier string is used to retrieve the necessary parameters for parsing the specific data table from XsdirContents. If the user inputs "all" as the data table identifier string, a loop will be used to iterate through XsdirContents, retrieving the necessary parameters for parsing each data table. The obtained parameters are then passed to the ACEParser class. Within the ACEParser class, the Parser() function is constructed to read the data files of the ACE nuclear database and perform the parsing. The nuclear data from each data table is parsed and stored in a dictionary called ACEContents, where the data table identifier strings are used as keys, and the parsed nuclear data is stored in corresponding classes. Finally, the parsed nuclear data, organized in the designed structure formed during the parsing process, is efficiently stored in an HDF5-format file by recursively invoking the ToHDF5() function within each class.

Due to the identical secondary particle data, precursor data, and average fission neutron data for continuous energy neutron nuclear data at different temperatures, it is sufficient to parse these data once for each nuclide at various temperatures. As a result, the Python package developed in this study exhibits high efficiency in manufacturing HDF5-format nuclear databases. The computational experiments were conducted on a computing platform featuring an AMD Ryzen 95,900 processor. It takes only 0.5 core-hours to generate a set of HDF5-format continuous energy neutron databases based on the ACE database provided by the Los Alamos Nuclear Data official website. The continuous energy neutron ACE database, based on ENDF-B/VIII.0, contains 3,897 data files and occupies 35.5 GB of storage.

Sections 2.2, 2.3 present the hierarchical design of the continuous energy neutron ACE parser class and thermal neutron scattering ACE parser class, respectively. The design of these parser classes reflects the nuclear data structure of the HDF5-format neutron nuclear database.

2.2 Continuous energy neutron nuclear database

This subsection presents the hierarchical design of the continuous energy neutron ACE parser class, which reflects the data structure of the HDF5-format database. Additionally, it elucidates various efforts undertaken to conserve disk space.

2.2.1 Design of the continuous energy neutron ACE parser class

In ACE nuclear database, all nuclear data are stored in XSS array, and need to use two integer arrays called NXS and JXS for data indexing. As shown in Figure 2, five main data blocks are designed based on the data contained in the continuous energy neutron ACE nuclear database, namely, Energy, Reactions, Nu, Precursors, and ProbabilityTable. The array Energy serves as a unidimensional repository for the main energy grid. The data blocks Nu, Precursors, ProbabilityTable, and Reactions individually encapsulate data pertaining to the average fission neutron number, delayed neutron precursor nuclei, unresolved region probability tables, and reaction channel details. Each of these data blocks is stored utilizing a Python class objects. Each class in the hierarchy generally corresponds to a group within the HDF5 file.

The Reactions data block is encapsulated within the class NeutronReaction. Reactions block encompasses all reaction channel information for nuclear reactions. Each reaction channel includes reaction cross-sections, associated energy grids, and data on secondary particle products. The secondary particle product data block comprises yield data and distribution data for the secondary

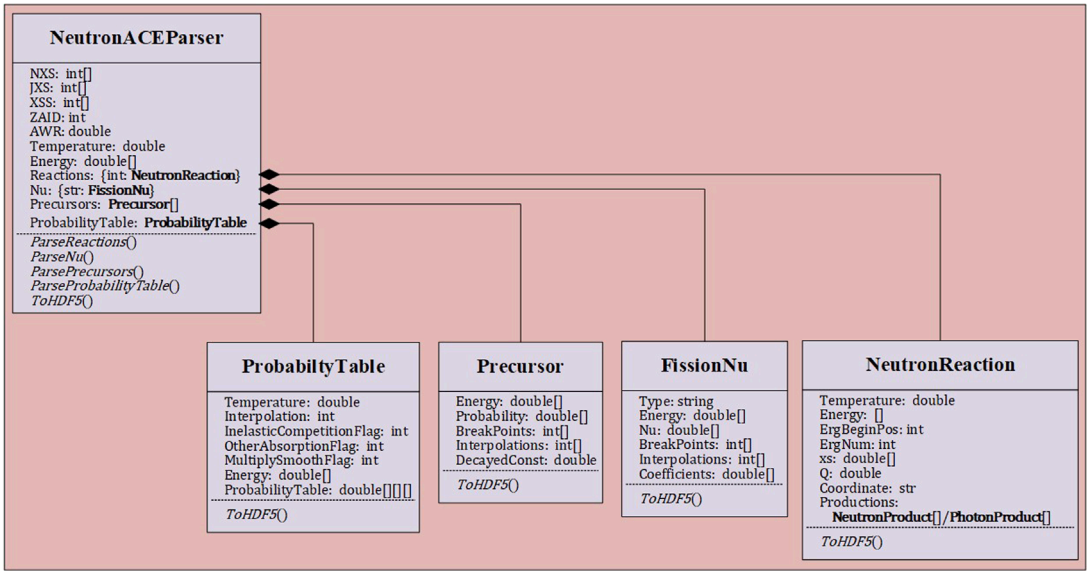

FIGURE 2
Top of continuous energy neutron ACE parser class hierarchy.

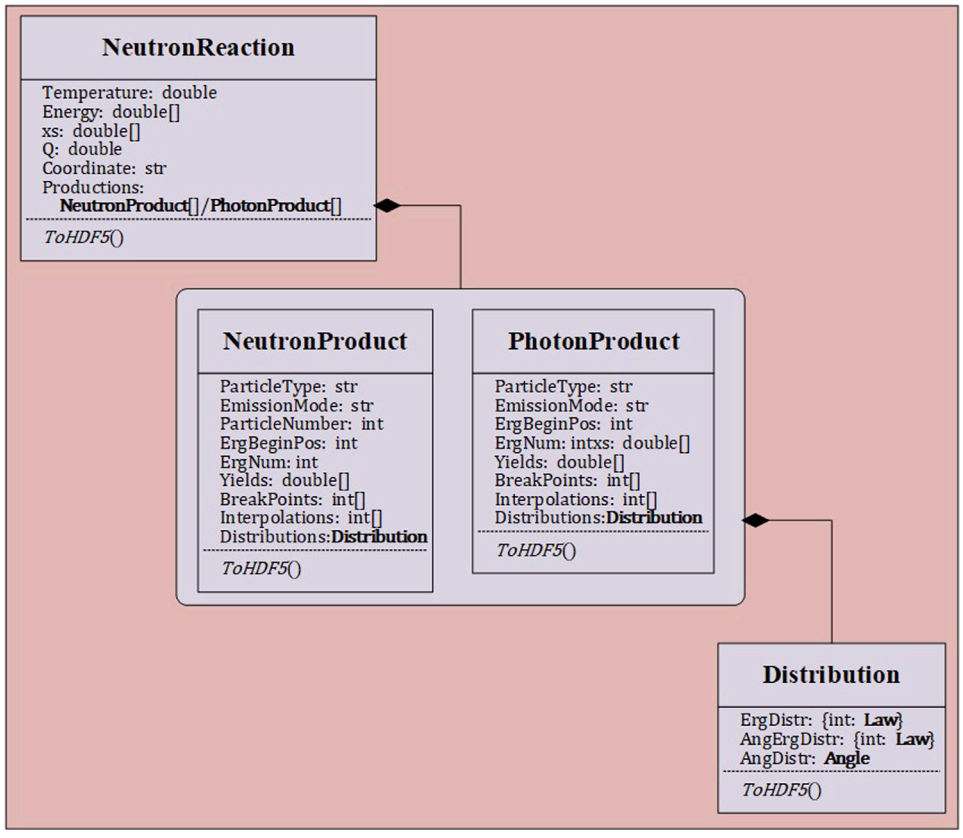

FIGURE 3
Neutron reaction channel class hierarchy.

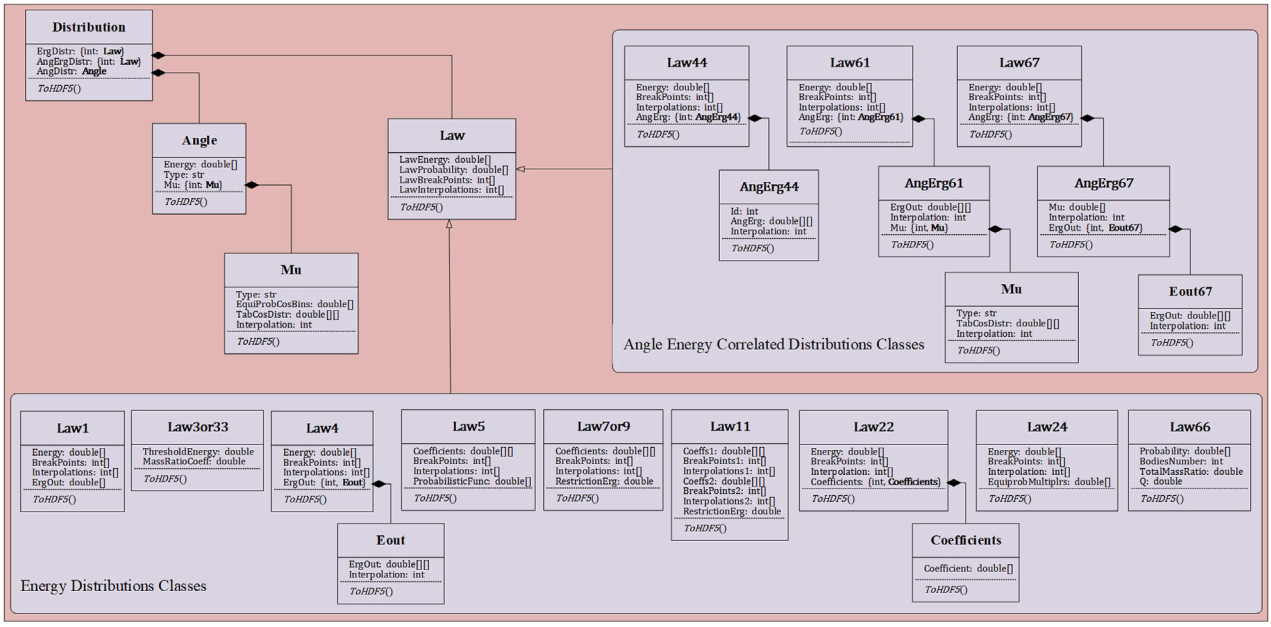

FIGURE 4
Distribution class hierarchy.

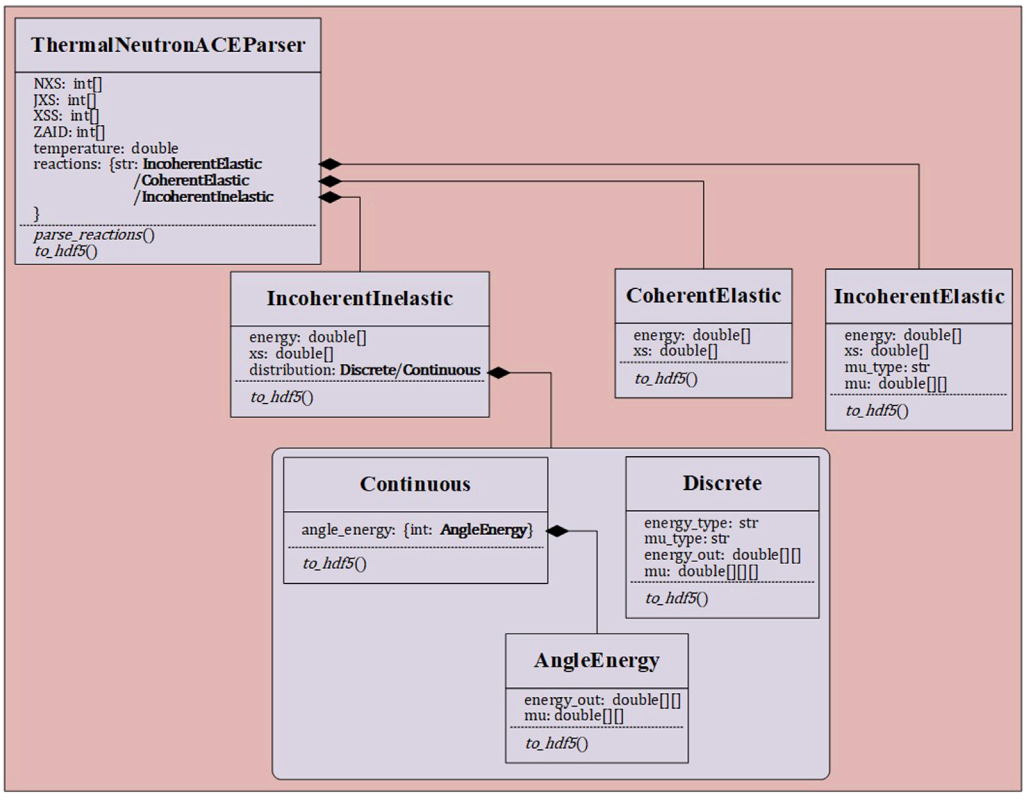

FIGURE 5
Thermal neutron scattering ACE parser class hierarchy.

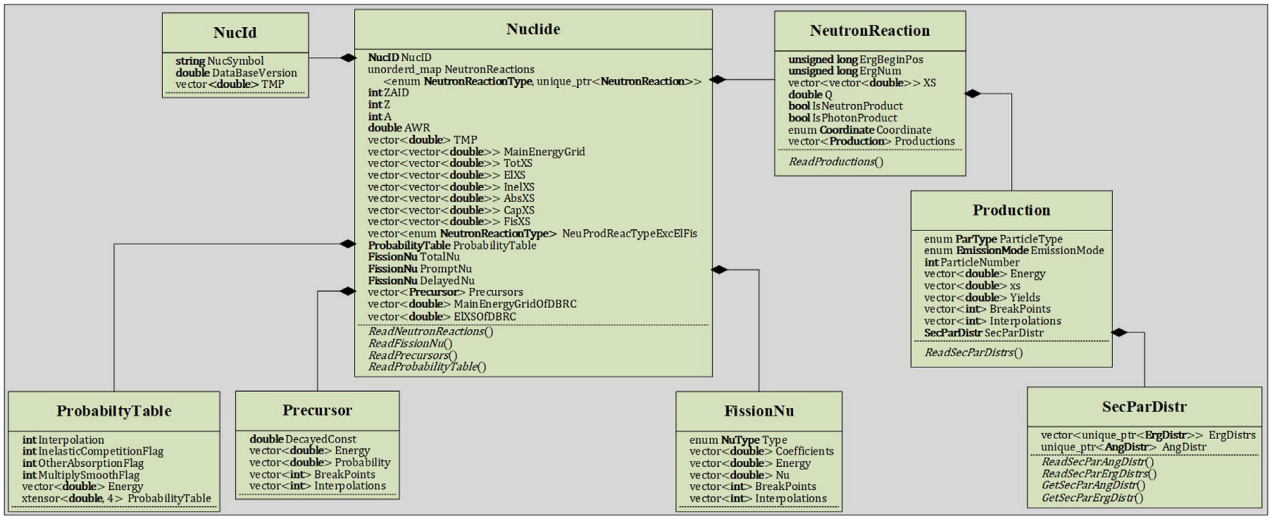

FIGURE 6
Top of nuclide class hierarchy for continuous energy neutron.

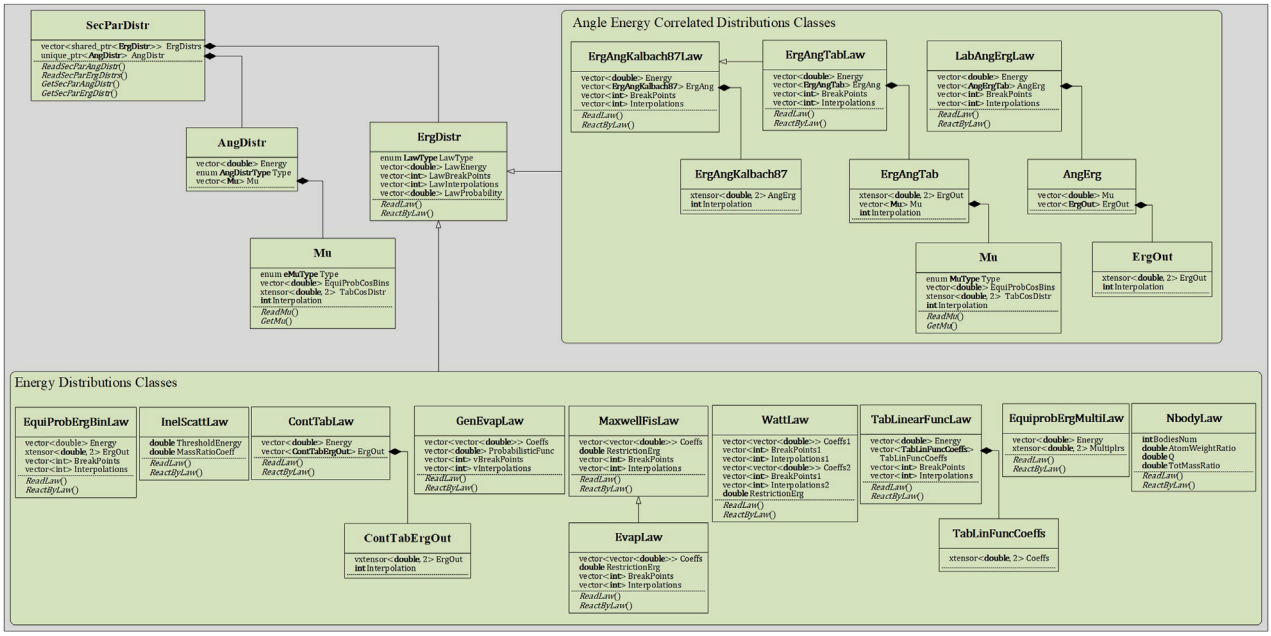

FIGURE 7
Secondary particle distribution class hierarchy for continuous energy neutron.

particles. The `NeutronReaction` class exhibits the highest level of reaction hierarchy, as illustrated in Figure 3.

The reaction product distribution data block incorporates angular and energy distributions. In addition to the elastic scattering neutrons, each reaction product has one or more energy distributions, which are represented by the abstract class `Law`. These distributions can be either angle uncorrelated or correlated, and they are named in Python code using the energy distribution law identifiers from the ACE nuclear database. If the energy distribution is angle correlated, then there will be no angular distribution data in distribution data, otherwise there will be angular distribution data. As depicted in Figure 4, all distribution data have been fragmented. The purpose of this is to facilitate readers in easily locating any desired data within the massive amounts of distribution data. Using `Law61` as an example, the `Energy` list contains incident neutron energy points. Each incident neutron energy point has an associated angle-energy correlated distribution, with this data stored in the dictionary `AngErg`. The keys in `AngErg` are indices for the incident energy points, and the values are instances of the distribution data class corresponding to the incident energies. This

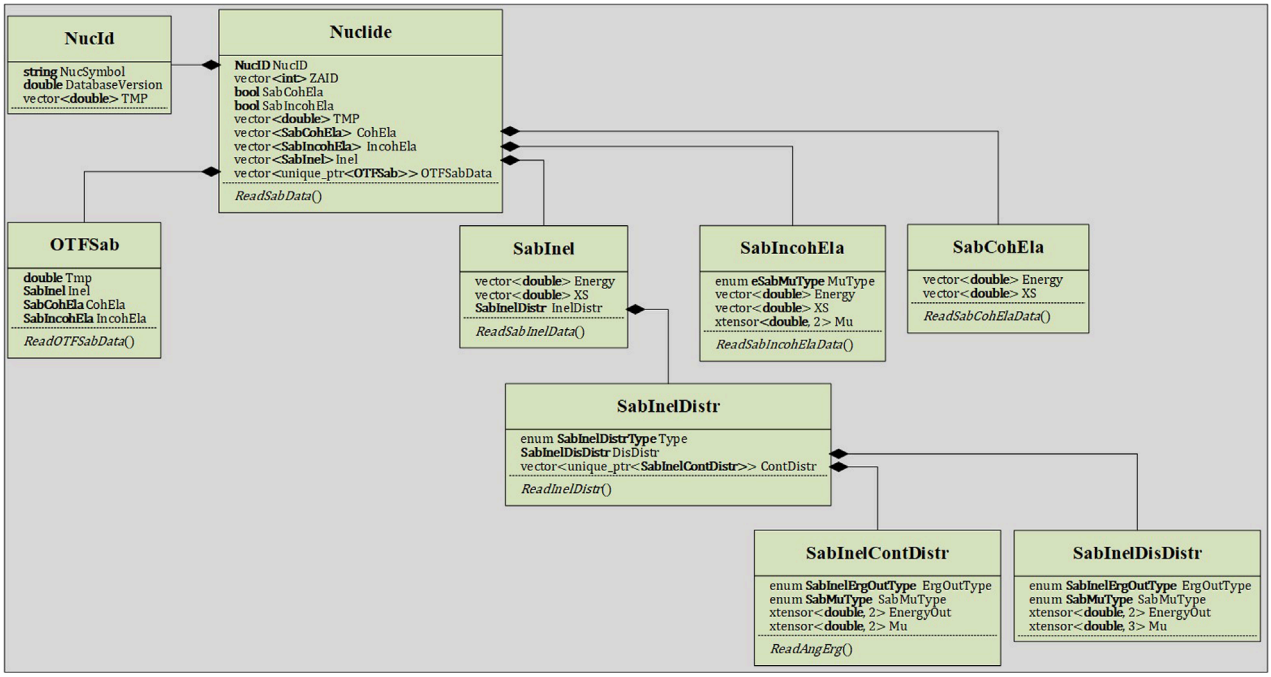

FIGURE 8
Nuclide class hierarchy for thermal neutron scattering.

class is named `AngErg61`. Within `AngErg61`, the `ErgOut` list holds outgoing energy points, and the `Interpolation` specifies the method for interpolating between these outgoing energy points. Each outgoing energy point is associated with a set of outgoing angle distribution data, stored in the dictionary `Mu`. The keys in Mu are indices for the outgoing energy points, and the values are instances of the angular distribution data class corresponding to outgoing energies. This class is named `Mu`, where the `Type` attribute denotes the type of angular distribution, which could be either tabular or isotropic. The `TabCosDistr` list contains outgoing angle cosine values along with their corresponding PDF (Probability Density Function) and CDF (Cumulative Distribution Function) values, and the Interpolation specifies the method for interpolating between these outgoing angle cosines. For detailed parameters of each law, please refer to the documentation provided in Conlin and Romano (2019) and Sweezy et al. (2003), and exhaustive details are not reiterated herein for brevity.

2.2.2 Processing for disk space saving

For continuous energy neutron databases spanning multiple temperatures, the sole distinctions lie in the main energy grid, probability tables, as well as cross-sections and energy grids within the reaction channels. All other dataset components remain invariant across different temperatures. Therefore, in HDF5-format nuclear database, the nuclear data at multiple temperatures do not need to be stored in separate instances as in the ACE. Rather, it suffices to store the main energy grid, probability tables, and reaction channel cross-sections and energy grids categorized by temperature, with the remaining data just storing in a singular instance. This will eliminate redundant nuclear data and significantly reduce data volume.

The energy grid corresponding to reaction channel cross-section data is derived by extracting a subset from the main energy grid. Consequently, in the HDF5-format nuclear database, it suffices to store the starting position (`ErgBeginPos` in Figure 3) and the numbers of the energy grid (`ErgNum` in Figure 3) corresponding to reaction channel cross-sections as attributes within the `Energy` data block under different temperature groups within the `Reaction` group. Notably, the `Energy` data block itself remains effectively empty.

Empirical research indicates that, in HDF5-format files, storing certain data as attributes rather than datasets can result in storage space savings (Romano and Harper, 2017). Therefore, some dispersed data, such as nuclide ZA identification (`ZAID` in Figure 2), atomic weight ratio (`AWR` in Figure 2), database temperature (`Temperature` in Figure 2), reaction Q value (`Q` in Figure 3), coordinate (`coordinate` in Figure 3), and frequently occurring interpolation parameters like `breakpoints` and `interpolations`, will be stored as attributes within groups or datasets.

2.3 Thermal neutron scattering nuclear database

This subsection presents the hierarchical design of the thermal neutron scattering ACE parser class, which reflects the data structure of the HDF5-format database. The hierarchy of the thermal neutron scattering ACE parsing class is significantly simpler than that of the continuous-energy neutron ACE parsing class, as depicted in Figure 5.

Thermal neutron scattering reactions encompass incoherent inelastic scattering, coherent elastic scattering, and incoherent elastic scattering. The data within these reaction channels varies with different temperatures. Consequently, in the HDF5-format

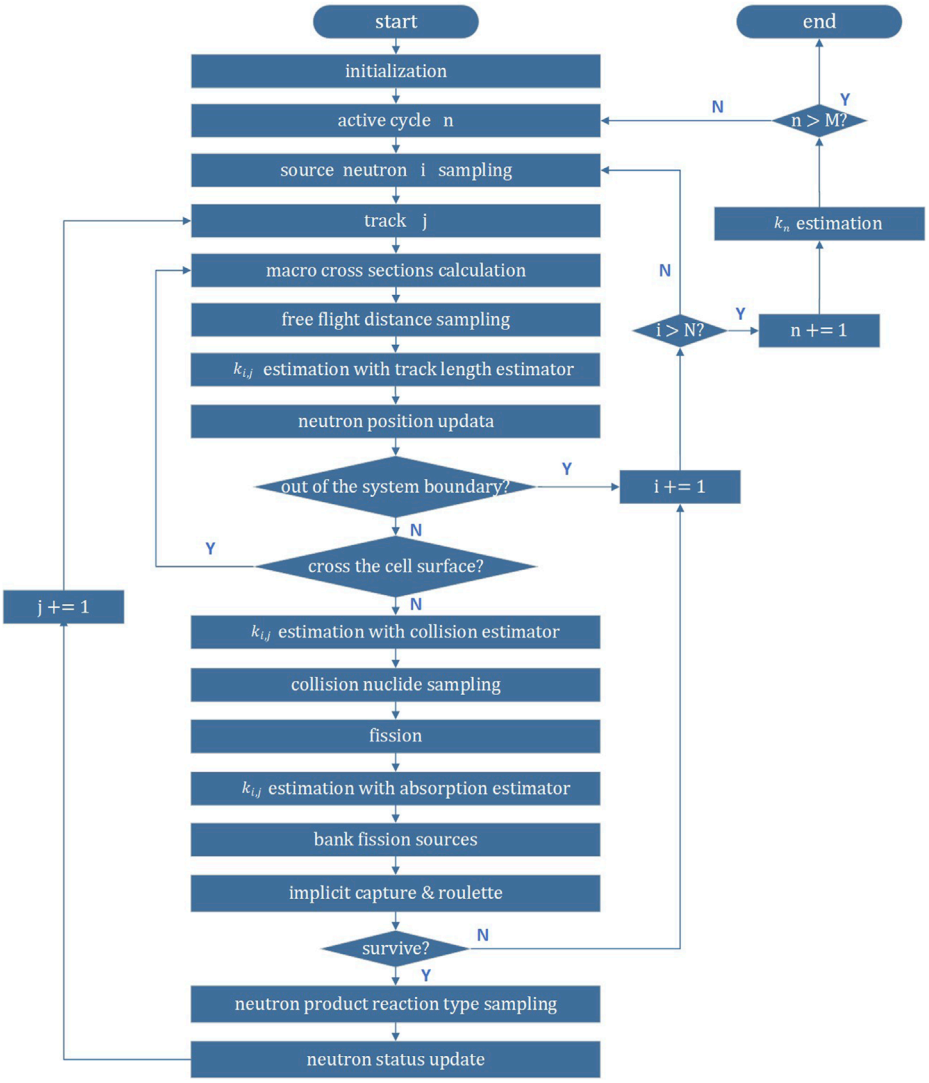

FIGURE 9
Criticality calculation flow in RMC.

thermal nuclear database, the data for each reaction channel at multiple temperatures is stored separately under groups named after their respective temperatures. Consequently, unlike the continuous energy neutron HDF5-format nuclear database, the thermal HDF5-format nuclear database does not exhibit a significant reduction in terms of data volume. The reduction mainly involves excluding some index data in ACE files.

3 Implementation in the RMC criticality module

3.1 Design of neutron nuclear data hierarchy in RMC

In the context of utilizing ACE nuclear databases within RMC code, the data retrieval process involves reading the ACE nuclear database data into three distinct one-dimensional arrays: NXS, JXS, and XSS in the class `Nuclide`. The XSS array serves as the repository for nuclear data utilized in particle transport. However, accessing specific data during the particle transport process involves a highly complex indexing procedure. This not only results in inefficient data indexing but also presents significant challenges for developers in terms of code maintenance and feature development. On the other hand, when the input card contains database identifiers for the same nuclide at different temperatures, using the ACE format nuclear database loads all the nuclear data for different temperatures into memory. However, for continuous energy neutron nuclear databases, the secondary particle data, average fission neutron numbers, and precursor data are independent of temperature. Therefore, using the ACE format nuclear database leads to data redundancy and wastage of memory.

In contrast, the adoption of the HDF5-format nuclear database brings a more direct and clear approach to RMC. Within RMC code, a underlying neutron nuclear data structure is constructed, as illustrated in Figures 6–8, which aligns hierarchically with the

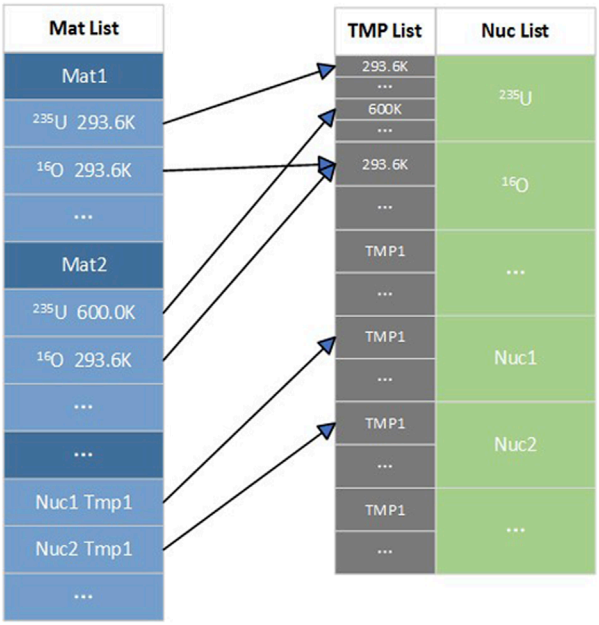

FIGURE 10
Nuclide index mapping based on HDF5-format nuclear database in RMC.

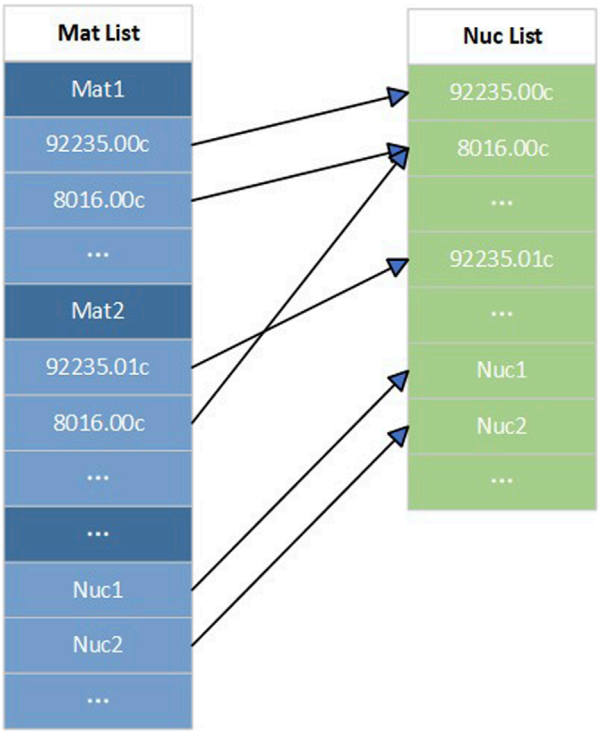

FIGURE 11
Nuclide index mapping based on ACE nuclear database in RMC.

structure of the HDF5-format nuclear database. This design not only allows developers to access all nuclear data within the code but also facilitates easy expansion of the nuclear data structure.

Considering on-the-fly cross-section processing capability in RMC code (e.g., on-the-fly thermal scattering temperature interpolation), neutron transport calculation efficiency, and consistency with computation results using ACE nuclear database, some special designs have been implemented, as described below.

3.1.1 Temperature dependence

As shown in Figures 6, 8, the `NucID` within the nuclear data class is a class object that stores isotope identifiers. For continuous energy neutron nuclear data, this class includes `NucSymbol`, `DatabaseVersion`, and `TMP`, representing the chemical symbol, database version, and temperature point of the database for the nuclide, respectively. For thermal neutron scattering nuclear data, the `NucID` contains `NucSymbol`, `DatabaseVersion`, and `TMP`. The `NucSymbol`, `DatabaseVersion`, and `TMP` in `NucID` are read from the material card in the RMC input file. `NucSymbol` and `DatabaseVersion` serve as unique identifiers for the databases. Whenever the same `NucSymbol` and `DatabaseVersion` appear in the material card, it is considered the same isotope, and all the Kelvin temperatures associated with that isotope in the material card are stored in the `TMP` vector. The RMC code retrieves the corresponding nuclear data from the database based on the `NucSymbol` and `DatabaseVersion`, and then reads the temperature-dependent data based on the Kelvin temperatures in the `TMP` vector. The corresponding MeV temperatures are also read from the database and stored in the `TMP` vector within the `Nuclide` class.

As evident from Figure 6, the temperature-dependent nuclear data in the continuous energy neutron nuclear database includes main energy grid, cross-sections, and probability table. The data dimensions of main energy grid, cross-sections, and probability table correspond to the dimensions of the `TMP`. This correspondence ensures that the data for each temperature point in `TMP` can be directly accessed and associated with the respective main energy grid, microscopic cross-sections, and probability tables. As shown in the `Nuclide` class in Figure 6, the `ProbabilityTable` is not designed as a vector. This is because only the probability table is different in the probability table data at different temperatures, while other data (e.g., `Interpolation`, `InelasticCompetitionFlag`, `OtherAbsorptionFlag`, `MultiplySmoothFlag` and `Energy` in `ProbabilityTable` class in Figure 6) are the same. Therefore, in order to reduce redundant data in the database, the probability table in the `ProbabilityTable` class is designed to be a four-dimensional tensor corresponding to `TMP`. Apart from the main energy grid, reaction cross section, and probability table, other nuclear data are independent of temperature. Therefore, only one copy of these data is stored in memory, significantly reducing memory usage compared to the ACE database when utilizing nuclear data at different temperatures for the same nuclide.

The thermal neutron scattering data is predominantly temperature-dependent. Therefore, in the Nuclide class for thermal scattering, the inelastic scattering (`Inel`), coherent elastic scattering (`CohEla`), and incoherent elastic scattering (`IncohEla`) data are all designed as vectors corresponding to `TMP`, as shown in Figure 8. To accurately calculate the thermal neutron scattering distribution, the RMC code has developed an on-the-fly thermal scattering temperature interpolation feature specifically for thermal neutron

TABLE 1 Continuous energy neutron nuclear database disk occupancy comparison: HDF5 vs. ACE (decimal).

Isotopic symbol	ACE (MB)	HDF5 (MB)	Disk space savings (%)
^{1}H	1.5	0.6	60.00
^{16}O	114.6	10.7	90.66
^{235}U	1,126.4	274.5	75.63
^{238}U	1,126.4	181.8	83.86
^{90}Zr	47.5	10.7	77.47
^{91}Zr	128.5	22.1	82.80
^{92}Zr	60.6	12.8	78.88
^{94}Zr	55.6	13.0	76.62
^{96}Zr	41.2	7.6	81.55

TABLE 2 Continuous energy neutron nuclear database disk occupancy comparison: HDF5 vs. ACE (binary).

Isotopic symbol	ACE (MB)	HDF5 (MB)	Disk space savings (%)
^{1}H	0.6	0.6	0.00
^{16}O	45.4	10.7	76.43
^{235}U	431.4	274.5	36.37
^{238}U	418.3	181.8	56.54
^{90}Zr	18.8	10.7	43.09
^{91}Zr	51.0	22.1	56.67
^{92}Zr	23.7	12.8	45.99
^{94}Zr	22.0	13.0	40.90
^{96}Zr	16.3	7.6	53.37

TABLE 3 Thermal neutron scattering nuclear database disk occupancy comparison: HDF5 vs. ACE (decimal).

Sab isotopic symbol	ACE (MB)	HDF5 (MB)	Disk space savings (%)
h-h2o	563.1	226.5	59.78
zr-zrh	190.1	76.5	59.76
grph	394.0	157.7	59.97
be-beo	339.3	135.7	60.01
o-beo	280.2	112.4	59.89

scattering data. For this purpose, in the `Nuclide` class for thermal scattering, the `OTFSabData` block is designed to store the thermal neutron scattering data at all temperatures, which is used for online temperature interpolation of thermal scattering.

3.1.2 Cross-section data structure design

Within the Nuclide class as shown in Figure 6, the variables `TotXS`, `ElXS`, `InelXS`, `AbsXS`, `CapXS`, and `FisXS` represent the total cross-section, elastic scattering cross-section, cumulative

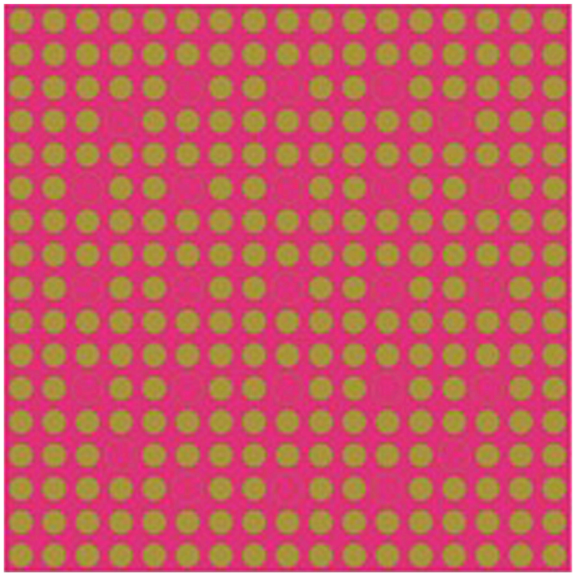

FIGURE 12
RMC assembly model of the VERA benchmark problem 2B.

TABLE 4 k_{eff} results for VERA benchmark problem 2B.

Nuclear database type	Ave	Std
ACE	1.171789	0.000315
HDF5	1.171789	0.000315

cross-section of all neutron-producing reaction channels excluding elastic scattering and fission, neutron disappearance reaction cross-section, capture reaction cross-section, and fission cross-section, respectively. The `InelXS` is obtained by summing the neutron-producing reaction cross-sections during the database retrieval process and does not correspond to a specific reaction channel. The remaining five cross-sections correspond to specific reaction channels. However, due to their frequent usage in the RMC neutron transport process and to enhance data indexing efficiency, these cross-sections are directly stored in the `Nuclide` class instead of their corresponding `NeutronReaction` class.

To enhance the accuracy of resonance elastic scattering simulation in the epithermal energy region of heavy nucleus, RMC has developed the Doppler broadening rejection correction (DBRC) function (Brown, 2019). This feature requires the use of main energy grid and elastic scattering cross-sections of the continuous energy neutron nuclear database at 0 K. Therefore, in the `Nuclide` class, this study has designed the `MainEnergyGridOfDBRC` and `ElXSOfDBRC` vectors to store the main energy grid and elastic scattering cross-sections at 0 K, respectively. Unlike using ACE databases where all nuclear data are loaded into memory, this approach significantly reduces memory consumption.

3.1.3 Calculation consistency design

All data used for criticality calculations in the HDF5-format nuclear database are generated from the ACE nuclear database. Therefore, in order to verify the accuracy of the database construction and ensure the correctness of application development in the RMC code, it is required that the criticality calculation results using the HDF5-format nuclear database match exactly with the results obtained using the ACE nuclear database. To achieve this goal, two specific details should be noted: reaction channel types with secondary neutron emission but exclude elastic scattering and fission must be stored in `NeuProdReacTypeExcElFis` (As shown in the `Nuclide` class in Figure 6) according to the sampling order in the ACE nuclear database to ensure consistency in reaction channel sampling order. In the same way, the secondary particle energy distributions should be stored in `ErgDistrs` following the sampling order in the ACE nuclear database to ensure consistency in the sampling of secondary particle energy distributions.

3.2 Criticality calculation process with HDF5-format nuclear database

Figure 9 shows the criticality calculation flow in RMC. The computational process for inactive cycles is identical to that of active cycles, except for the absence of count statistics. Therefore, the flow chart only depicts the computational process for active cycles.

During the initialization process, RMC constructs an index map from the material list to the nuclide list and reads the HDF5-format nuclear database. Figure 10 illustrates the index mapping from the material list to the nuclide list based on the HDF5-format nuclear database, while Figure 11 shows the index mapping from the material list to the nuclide list based on the ACE nuclear data. A comparison reveals that the nuclide list based on the ACE nuclear database may contain duplicate entries for the same nuclide, such as 92235.00c and 92235.01c, which differ only in the database temperature. However, 92235.00c and 92235.01c contain the same secondary particle data. Therefore, during the calculation process, two identical copies of secondary particle data for 235U will be stored in the memory, and the more temperature points there are, the more redundant data there will be. To eliminate this redundancy, the HDF5-format nuclear database in RMC integrates nuclear data for the same nuclide at different temperatures. Consequently, a temperature list is added for temperature-related data, and during the RMC initialization process, an index mapping from the material list to the nuclide temperature lists is constructed. Once the construction of the index mapping from the material list to the nuclide list is completed, the reading of the HDF5-format nuclear database begins. Simultaneously, the construction of the underlying nuclear data structure, as described in Section 2.2, is performed. The database reading process is implemented by invoking the read functions within the c++ classes shown in Figures 6–8.

Subsequently, for each neutron history, ray-tracing is executed along each track until a collision event occurs. During the ray-tracing process, the flight distance of particle is sampled based on the macroscopic total cross-section of the material particle located. Prior to each flight distance sampling, calculations of the macroscopic total cross-sections of all nuclides in the material are performed, and the `TotXS`, `ElXS`, `InelXS`, `AbsXS`, `CapXS`, and `FisXS` are all used to calculate the total cross-section and inelastic cross-section when using unresolved region probability table. Due to the typically large number of particle tracks per

TABLE 5 Neutron flux tally results for VERA benchmark problem 2B.

Region	Ave-ACE ($/(cm^2 * s)$)	Re-ACE ($\times 10^{-4}$)	Ave-HDF5 ($/(cm^2 * s)$)	Re-HDF5 ($\times 10^{-4}$)
Fuel	14.155	2.2278	14.155	2.2278
Moderator in fuel pins	4.2948	2.2178	4.2948	2.2178
Cladding in fuel pins	23.574	2.1075	23.574	2.1075

TABLE 6 Running time for VERA benchmark problem 2B: HDF5 vs. ACE.

ACE (*s*)	HDF5 (*s*)	Run time savings (%)
152.9	122.2	20.1

TABLE 7 Memory usage for VERA benchmark problem 2B: HDF5 vs. ACE.

ACE (GByte)	HDF5 (GByte)	Memory savings (%)
9.1	8.4	7.7

TABLE 8 Running time for case1 and case 2: HDF5 vs. ACE.

Cases	ACE (*s*)	HDF5 (*s*)	Run time savings (%)
Case1	129.7	112.6	13.2
Case2	148.3	121.6	18.0

TABLE 9 Memory usage for case1 and case 2: HDF5 vs. ACE.

Cases	ACE (GByte)	HDF5 (GByte)	Memory savings (%)
Case1	5.6	5.3	5.4
Case2	10.8	7.0	35.2

history, cross-section calculations are frequently executed during the ray-tracing process. Furthermore, on-the-fly cross-section processing technologies are employed to handle these cross-sections dynamically. Therefore, due to the higher frequency of these cross-section invocations, the cross-section design presented in subsection 3.1.2 becomes imperative.

After ray-tracing, processing of collision type sampling, forced fission, implicit capture, and neutron-producing reactions exclude elastic scattering and fission sampling are performed. During the generation of fission sources and sampling reaction type with neutron production, it is necessary to sample the energy and angular distributions of secondary neutrons. The sampling process no longer relies on the complex data indexing used in the ACE nuclear database. Instead, it involves invoking the `GetSecParAngDistr()` function and the `GetSecParErgDistr()` function within the `SecParDistr` class to retrieve the angular and energy distributions for secondary neutrons, respectively. Subsequently, the `GetMu()` function in the `AngDistr` class is invoked to determine the emission angle of the secondary neutron, and the `ReactByLaw()` function in the sampled energy distribution class object is used to calculate the emission energy of the secondary neutron.

4 Results

4.1 Test of disk space occupancy of HDF5-format nuclear database

In this study, HDF5-format nuclear databases of continuous energy neutron and thermal neutron scattering nuclear databases are manufactured based on the ACE nuclear database based on ENDF-B/VIII.0, which is provided by LANL database website. Compared to the decimal format continuous energy neutron ACE nuclear database, the HDF5-format database achieved a significant 80% reduction in disk space requirements, while the thermal neutron scattering database demonstrated an equivalent 60% reduction. Furthermore, Table 1, Tables 2, 3 provide detailed listings of disk space savings for several isotopes and S(α,β) nuclides.

Compared to the decimal format ACE nuclear database, the HDF5-format nuclear database significantly reduces the disk space. This is primarily benefit from the data compression capabilities of the HDF5-format file. Furthermore, the disk space savings for the HDF5-format continuous energy neutron nuclear database are much greater than the disk space savings for the HDF5-format thermal neutron scattering nuclear database. This is because, in the case of the continuous energy neutron database, the secondary particle production data at different temperatures of the same isotope is identical. Therefore, only one instance of particle production data is stored in the HDF5-format nuclear database, resulting in a significant reduction in redundant data. To demonstrate this, this study also compared the disk space usage of the binary continuous energy neutron ACE nuclear database with that of the HDF5-format nuclear database. Compared with the binary format continuous energy neutron ACE nuclear database, the HDF5-format database achieved a 50% reduction in disk space requirements.

For the thermal neutron scattering nuclear database, the secondary particle data for different temperatures is different. Thus, the disk space savings for the thermal neutron scattering HDF5-format nuclear database are mainly attributed to the data compression capabilities of the HDF5-format files, without the contribution of reducing redundant data.

Tables 1, 2 indicate that the disk space savings for the continuous energy neutron database of 1H are comparatively lower than those of other isotopes. This is due to the fact that 1H has secondary particle production data only for elastic scattering and capture reaction channels. With a limited amount of secondary particle production data available, the disk space savings for 1H are reduced in comparison to other isotopes.

4.2 Verification of correctness and testing of efficiency

In this study, the assembly model for VERA benchmark problem 2B was modeled using RMC. The RMC geometry model is illustrated in Figure 12, and the material parameters for the model can be found in reference (Godfrey, 2014). The temperature of all lattice cell is 600 K.

The criticality simulations of the assembly model of VERA benchmark problem 2B were conducted with ACE nuclear database and HDF5-format nuclear database. Neutron flux counts in the fuel, moderator, and cladding regions were compared. The functions of unresolved region probability table, on-the-fly thermal scattering temperature interpolation and DBRC were enabled in the calculation. The number of inactive cycles was set to 100, the number of active cycles was set to 400, and the number of histories per cycle was set to 10,000. As presented in Tables 4, 5, the computational results of k_{eff} and tally scores were found to be completely consistent, demonstrating the accuracy of the HDF5-format nuclear database as well as the correctness of the criticality calculation process with HDF5-format nuclear database.

The parallel computations were performed using both the ACE nuclear database and the HDF5-format nuclear database with 15 MPI processes on the same computing platform, and this experiment was repeated 5 times. As shown in Tables 6, 7, the criticality calculations based on the HDF5-format nuclear database in RMC demonstrated a 20.1% improvement in computational efficiency compared to the criticality source calculations based on the ACE format nuclear database. Furthermore, it achieved a significant reduction in computer memory usage by 7.7%.

As described in Section 3.1.1 and Section 3.2, the design of the continuous energy neutron nuclear data structure ensures that temperature-independent nuclear data for the same nuclide is stored only one copy in memory, which significantly reduce memory usage. To demonstrate this, two models were constructed based on the VERA benchmark problem 2B, labeled as case1 and case2. In case 1, the temperature of all fuel rods is set to 293.6 K. In case 2, the temperature of some fuel rods is set to 293.6 K, while the temperature of the other fuel rods is set to 600 K. Consequently, all nuclides in case 1 utilize the database at a single temperature point of 293.6 K, whereas all nuclides in case 2 utilize the database at two temperature points: 293.6 K and 600 K. The parallel computations were performed with 15 MPI processes on the same computing platform. In the calculation, the on-the-fly thermal scattering temperature interpolation and DBRC features were disabled. As shown in Table 8, the code running time with the HDF5 format nuclear database is significantly reduced. Moreover, the time savings are greater in case2 compared to case1. As shown in Table 9, the memory savings in case2 are more significant compared to case1, demonstrating that eliminating temperature-independent redundant data greatly reduces memory usage. For both case1 and case2, the code running time was compared using the ACE format database and the HDF5 format nuclear database. The calculation for each case was repeated 5 times.

The above tests demonstrate that the new data structure design markedly enhances both program memory usage and efficiency. Furthermore, when utilizing databases at various temperature points for the same nuclide, the savings in memory and improvements in efficiency are notably enhanced.

5 Summary

In this study, a newly structured neutron nuclear database was designed for implementation in Monte Carlo program RMC. Additionally, a Python code package was developed to parse ACE nuclear databases and manufacture HDF5-format nuclear databases based on the designed nuclear data structure. The resulting HDF5-format nuclear database demonstrates exceptional readability. Furthermore, by eliminating redundant data in ACE nuclear database and leveraging the compression characteristics of HDF5 files, a significant reduction in disk space usage is achieved compared to ACE nuclear databases. The thermal neutron scattering nuclear database exhibits a 60% reduction, and the continuous energy neutron nuclear database achieves an 80% reduction in disk space consumption.

A redesign of the underlying neutron nuclear data structure in RMC was conducted using the HDF5-format nuclear database. Additionally, application development was implemented for the criticality calculation module of RMC. Criticality simulations were performed on the assembly model of the VERA benchmark problem 2B using both the ACE nuclear database and the HDF5-format nuclear database. The computed values of k_{eff} and cell neutron flux tally were found to be entirely consistent, which confirms the accuracy of the HDF5-format nuclear database and the correctness of the application development in RMC. Additionally, the computational efficiency and memory usage of RMC criticality calculations were tested and compared using the ACE nuclear database and the HDF5-format nuclear database. The test results indicate that when using the database at a single temperature point, memory usage with the HDF5-format nuclear database is reduced by 5.4%, and the running time is reduced by 13.2%. When using the database at two temperature points, memory usage is reduced by 35.2% and running time by 18.0%. This demonstrates that the new data structure design reduces temperature-independent redundant nuclear data and improves data indexing efficiency. Consequently, the more temperature points used for the same nuclide, the greater the savings in memory and running time.

In summary, this study demonstrates that the HDF5-format nuclear data implemented in the Monte Carlo code RMC surpasses the ACE format nuclear database in terms of readability, extensibility, disk space utilization, computational efficiency, and memory usage.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found here: https://nucleardata.lanl.gov/.

Author contributions

WW: Conceptualization, Data curation, Methodology, Software, Validation, Visualization, Writing–original draft. CJ: Validation, Writing–review and editing. ZL: Funding acquisition, Writing–review and editing. KW: Project administration, Supervision, Writing–review and editing.

Funding

The author(s) declare that financial support was received for the research, authorship, and/or publication of this article. Supported by the National Natural Science Foundation of China (No. 12105150).

Acknowledgments

I would like to express my sincere gratitude to the Reactor Engineering Computational Analysis Laboratory for providing the research platform. I am deeply thankful to my advisor, KW, for his invaluable support throughout my research. My heartfelt appreciation also goes to CJ and ZL for their invaluable contributions and insightful guidance.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Agostinelli, S., Allison, J., Amako, K., Apostolakis, J., Araujo, H., Arce, P., et al. (2003). Geant4—a simulation toolkit. *Nucl. Instrum. Methods Phys. Res. Sect. A Accel. Spectrom. Detect. Assoc. Equip.* 506, 250–303. doi:10.1016/S0168-9002(03)01368-8

Battistoni, G., Bauer, J., Boehlen, T. T., Cerutti, F., Chin, M. P., Dos Santos Augusto, R., et al. (2016). The fluka code: an accurate simulation tool for particle therapy. *Front. Oncol.* 6, 116. doi:10.3389/fonc.2016.00116

Brown, F. B. (2019). *Doppler broadening resonance correction for free-gas scattering in MCNP6. 2*. Technical report. Los Alamos, NM (United States): Los Alamos National Laboratory.

Conlin, J. L., and Romano, P. (2019). *A compact endf (ace) format specification*. Los Alamos (NM): Los Alamos National Laboratory.

Forster, R., and Godfrey, T. (2006). "Mcnp-a general Monte Carlo code for neutron and photon transport," in Monte-Carlo Methods and Applications in Neutronics, Photonics and Statistical Physics: Proceedings of the Joint Los Alamos National Laboratory-Commissariat à l'Energie Atomique Meeting, Cadarache Castle, Provence, France, April 22–26, 1985 (Springer), 33–55.

Godfrey, A. T. (2014). *VERA core physics benchmark progression problem specifications*. Technical report. Oak Ridge, TN (United States): Oak Ridge National Laboratory.

Leppänen, J. (2013). *Serpent–a continuous-energy Monte Carlo reactor physics burnup calculation code*. Finland: VTT Technical Research Centre of Finland.

Macfarlane, R., Muir, D. W., Boicourt, R., Kahler III, A. C., and Conlin, J. L. (2017). *The NJOY nuclear data processing system, version 2016*. Technical report. Los Alamos, NM (United States): Los Alamos National Laboratory. doi:10.2172/1338791

Mattoon, C., Beck, B., Patel, N., Summers, N., Hedstrom, G., and Brown, D. (2012). Generalized nuclear data: a new structure (with supporting infrastructure) for handling nuclear data. *Nucl. Data Sheets* 113, 3145–3171. doi:10.1016/j.nds.2012.11.008

Romano, P., and Harper, S. (2017). Nuclear data processing capabilities in openmc. *EPJ Web Conf.* 146, 06011. doi:10.1051/epjconf/201714606011

Sweezy, J. E., Booth, T., Brown, F., Bull, J., Forster III, R., Goorley, J., et al. (2003). *Mcnp a general Monte Carlo n-particle transport code, version 5*. Los Alamos, NM: Los Alamos National Laboratory.

Trkov, A., and Brown, D. A. (2018). *ENDF-6 formats manual: data formats and procedures for the evaluated nuclear data files*. Technical report. Upton, NY (United States): Brookhaven National Laboratory. doi:10.2172/142511

Wang, K., Li, Z., She, D., Xu, Q., Qiu, Y., Yu, J., et al. (2015). Rmc–a Monte Carlo code for reactor core analysis. *Ann. Nucl. Energy* 82, 121–129. doi:10.1016/j.anucene.2014.08.048