# Design Document for the OpenJOY Program

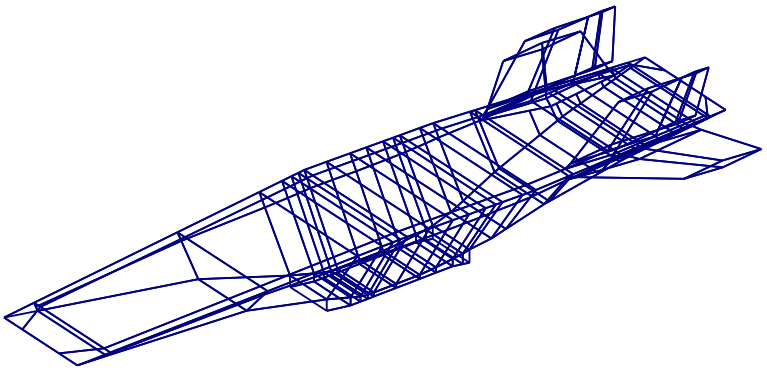
by

OpenJoy Team

A design document for the OpenJOY program
2016

Customer Advisory Committee:

Software Center, IAPCM, Beijing China



---

2016

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF PROGRAMS

# LIST OF APPENDICES

# LIST OF ABBREVIATIONS

# ABSTRACT

**Design Document for the OpenJOY Program**

**by**

**OpenJoy Team**

**Chair: Insert a Chair!**

The OpenJOY program is an open source nuclear data processing framework, which intends to provide the community a research tool that enables them to test new ways of development.

# CHAPTER 1

# Introduction

# CHAPTER 2

# CMS: Constants Maintenance System

We begin with the discussion of a very simple module of the OpenJOY: the constants maintanance system (CMS), which defines the commonly used constants across the computer codes.

## 2.1 List of constants used

The constants commonly used are listed in Table 2.1.

Table 2.1: The constants commonly used in OpenJOY

| Symbol | Meaning | Value |
|---|---|---|
| $k$ | Boltzmann Constant | $8.6173324 \times 10^{-11}$ MeV/K, $8.6173324 \times 10^{-5}$ eV/K |

## 2.2 C++ Data Structures

The data structure used in the computer code is given below.

```
class CMS {
public:
    // Boltzmann Constants
    constexpr static const double BoltzmannMevK = 8.6173324e-11;
    constexpr static const double BoltzmannEvK  = 8.6173324e-5;

};
```

Program 2.1: C++ public APIs for CMS module

# CHAPTER 3

# NIST: NIST Nuclear Data System

## 3.1 Isotopes and Elements

The purpose of the NIST module is for describing all elements and isotopes in an organized format. The ENDF material identifier and ACE ZAID identifier number are mapped to a NIST element or isotope or even its excited meta state. This module serves as a base for linking evaluated nuclear files between different evaluation system.

| Isotopes | ENDF ID | ACE ZAID | Isotopes | ENDF ID | ACE ZAID |
|---|---|---|---|---|---|
| H-$n$ | $125 + 3(n - 1)$ | $1000 + n$ | He-$n$ | $225 + 3(n - 3)$ | $2000 + n$ |
| Li-$n$ | $325 + 3(n - 6)$ | $3000 + n$ | Be-$n$ | $425 + 3(n - 9)$ | $4000 + n$ |
| B-$n$ | $525 + 3(n - 10)$ | $5000 + n$ | C-$n$ | $625 + 3(n - 12)$ | $6000 + n$ |
| U-$n$ | $9225 + 3(n - 234)$ | $92000 + n$ | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |

| | | | | | |
|---|---|---|---|---|---|
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |
| H-1 | 125 | 1001 | | | |

Table 3.1: Meaning for ENDF material id and ACE ZAID

# CHAPTER 4

# ENDF: ENDF Evaluated File System

## 4.1 ENDF-6 Format

The ENDF neutron data-files are stored in text files. Each row of the text file is an array of 80 characters. The first 66 characters are formatted depending on the type of the record. Then it follows by 4 characters of material number (MAT), 2 characters of file number (MF), 3 characters of section number (MT), and 5 characters of line number (NS). An illustration of the line structure is shown in Figure 4.1. The data array at the beginning contains 66 characters whose form depends on the type of the record. The possibilities are:

- A string of 66 characters, or
- Six numbers with each occupying 11 characters.

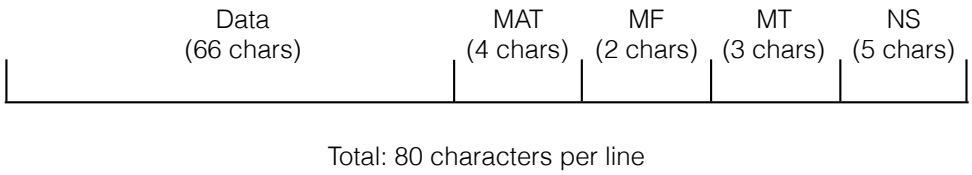



Figure 4.1: An example line of record for ENDF file

The ENDF-6 format specifies five types of commonly used records: TEXT, CONT, LIST, TAB1, and TAB2.

### 4.1.1 TEXT Records

A TEXT records describes some text information, for example the meta information of the cross section.. The data section of the TEXT record is contains only one field: HL, which

takes 66 characters. An illustration of the text record is shown in Figure 4.2.

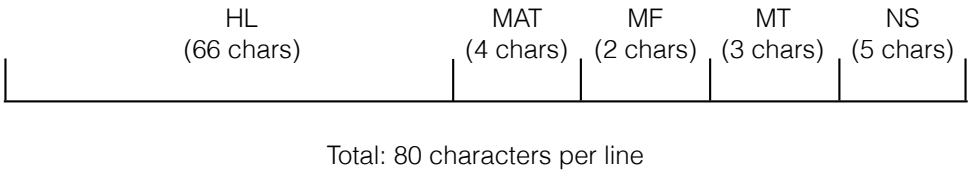



Figure 4.2: An example line of TEXT record for ENDF file

### 4.1.2 CONT Records

A CONT record includes some control information. The data field contains six floating numbers: C1, C2, L1, L2, N1 and N2. An illustration of the structure is shown in Figure 4.3.

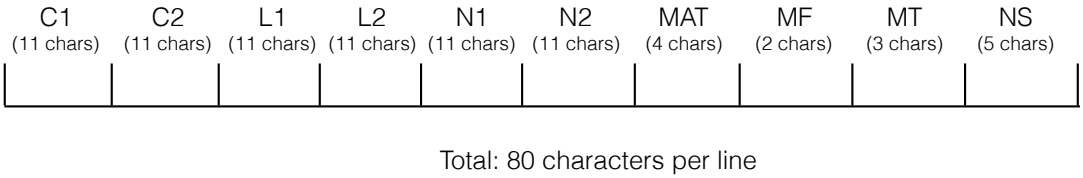



Figure 4.3: An example line of CONT record for ENDF file

### 4.1.3 LIST Records

A LIST record encodes a list of numbers. The records contains multiple lines, where the first line is a CONT record, and followed by a list of numbers organized into packages of 6 numbers. An illustration of these lines are shown in Figure 4.4. The length of list is stored in the 5th argument in the variable NPL in the first line. NPL needs not to be a multiple of 6, since the empty spaces in the list will be filled up with zeros.

### 4.1.4 TAB1 Records

A TAB1 record contains a one-dimensional tabulated functions such as $y(x)$. To describe the table function, four lists of data are presenting: NBT, INT, X, and Y. The dimensions of NBT and INT are NR, and the dimensions of X and Y are NP. This section of data contains

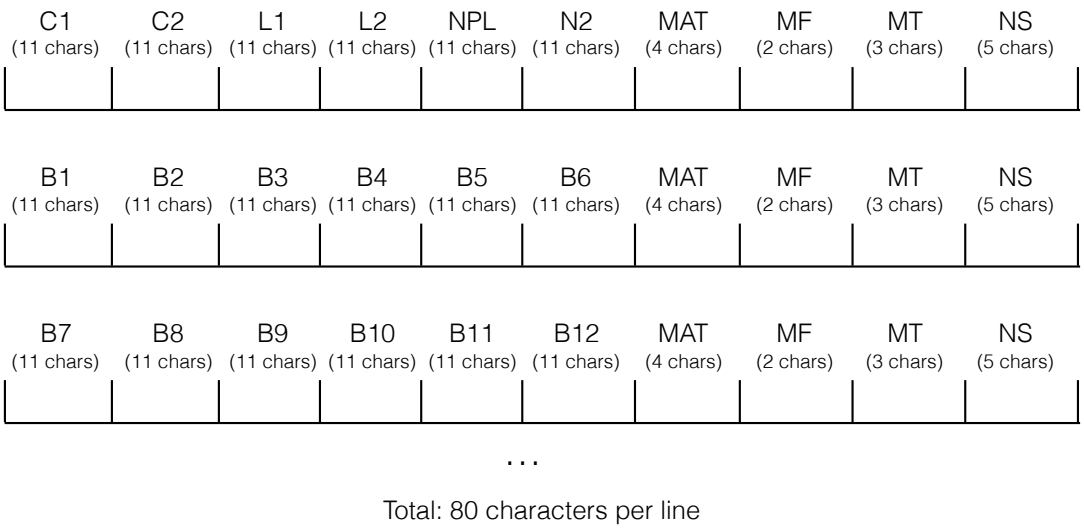



Figure 4.4: An example line of LIST record for ENDF file

a control record at the beginning, then followed by pairs of (NBT, INT) and pairs of (X, Y). An illustration of the storage format is shown in Figure 4.5.

### 4.1.5 TAB2 Records

A TAB2 record contains a two-dimensional tabular functions such as $y(x, z)$. The TAB2 records storages a list of data points for the $z$ variable and the interpolation rules in the NBT and INT array. It follows by NZ records with the C2 designated for the corresponding $z$ value. The possible records followed are either TAB1 or LIST record. An illustration of the format of storage is shown in Figure 5.1.

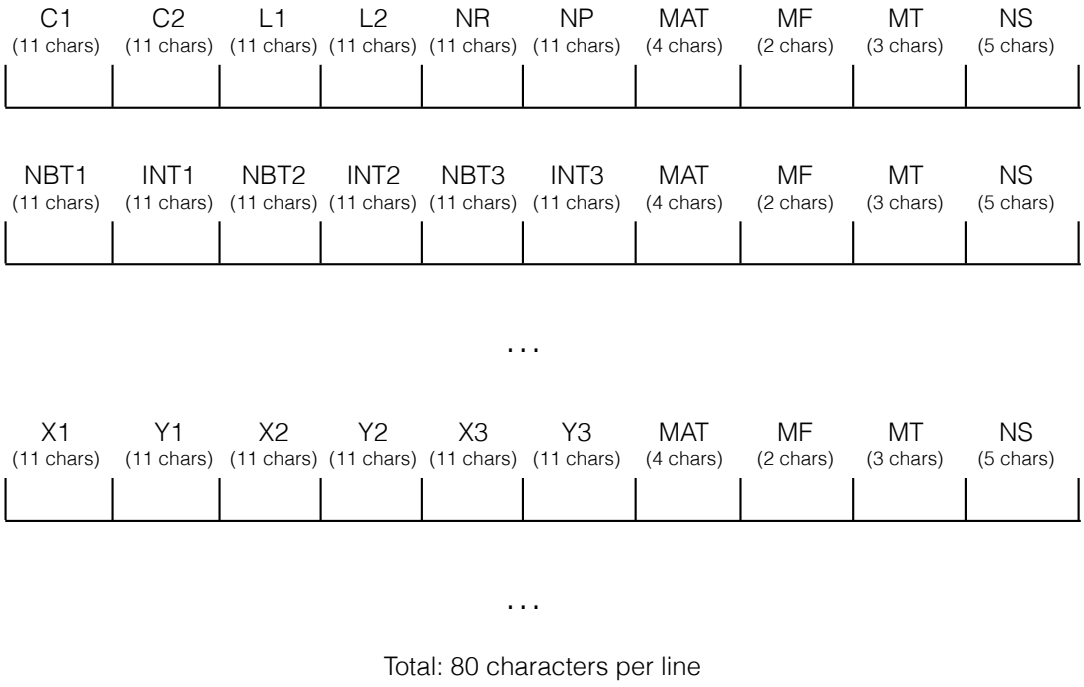



Figure 4.5: An example line of TAB1 record for ENDF file

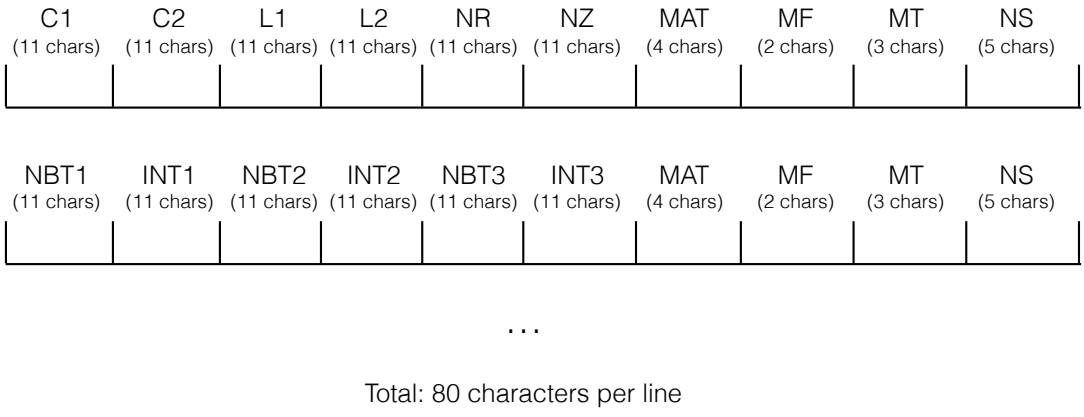



Figure 4.6: An example line of TAB2 record for ENDF file

## 4.2 C++ Data Structure & Application Interfaces

The program is implemented native in C++ programming language, so in this section, we discuss the data structures first then the application interfaces.

The top level neutron data structure contains several major parts: the fission yield data, the reaction data, and the resonance data, as followed.

```
struct ENDFNeutronData {

    ... definitions of support data structures ...

    /* Key data structures for neutron data */

    // The fission yield information, from File 1 and 5
    YieldData                  fissionYield;

    // List of reactions information, from File 3
    std::vector<ReactionData>  reactions;

    // List of resonances information, from File 2
    std::vector<ResonanceData> resonances;
}
```

The discussion begins with lower level data structures, and then discuss each major part in details.

### 4.2.1 Fundamental Computational Data Structures

- *Interpolation function:* The law of interpolation is a structure contains two fields: INT for the interpolation form, and NBT for the interpolation index.

  ```
  struct ENDFInterpLaw {
      long INT = -1;
      long NBT = -1;
  };
  ```

  The law of interpolation is used in a list, i.e.

  ```
  std::vector<ENDFInterpLaw>
  ```

, which is usually combined with a list of scattered function values with $x - y$ pairs. One such pair is called a datapoint, which is defined in a structure as followed:

```
template <typename T>
struct ENDFObjectDataPoint {
    double x = 0.;
    T y;
};
```

$x$ value is a scalar of floating number, and the $y$ value can be any object, and is expressed in a C++ template class parameter. A special case where $y$ is a scalar floating number as well is defined as a special type ENDFDataPoint.

```
typedef ENDFObjectDataPoint<double>   ENDFDataPoint;
```

A tabular function is represented by a list of ENDFInterpLaw and a list of ENDF-DataPoint:

```
struct ENDFInterpolationFunction {
    std::vector< ENDFInterpLaw > interp;
    std::vector< ENDFDataPoint > data;

    // Evaluate the interpolation function at point x
    double evaluate(double x) const;
};
```

The meaning of the integer INT is listed in table 4.1. Next, we illustrate how to

Table 4.1: Definition of interpolation laws

| INT | Interpolation Scheme |
|---|---|
| 1 | $y$ is constant in $x$ |
| 2 | $y$ is linear in $x$ |
| 3 | $y$ is linear in $\ln(x)$ |
| 4 | $\ln(y)$ is linear in $x$ |
| 5 | $\ln(y)$ is linear in $\ln(x)$ |

interpolate the function at any given $x$ with an example, as shown in Figure 4.7, taken from the ENDF manual. In this function, we have 10 data points which splits into 3 regions, the NBT numbers are marked to indicate the upper boundaries of the

regions. The INT numbers indicate the interpolation laws of the regions, as explained in Table 4.1. It is possible that two points share the same $x$-coordinate, and this point relates to a 'jump' in the function. For a given $x$ between two points $(x_l, y_l)$ and $(x_r, y_r)$ with an interpolation law INT. Then, the interpolated $y$ is calculated by:

$$y = y_l, \text{ if INT} = 1, \tag{4.1}$$

$$y = y_l + \frac{x - x_l}{x_r - x_l}(y_r - y_l), \text{ if INT} = 2, \tag{4.2}$$

$$y = y_l + \frac{\log(x/x_l)}{\log(x_r/x_l)}(y_r - y_l), \text{ if INT} = 3, \tag{4.3}$$

$$y = y_l \left(\frac{y_r}{y_l}\right)^{\left(\frac{x-x_l}{x_r-x_l}\right)}, \text{ if INT} = 4, \tag{4.4}$$

$$y = y_l \left(\frac{y_r}{y_l}\right)^{\left(\frac{\log(x/x_l)}{\log(x_r/x_l)}\right)}, \text{ if INT} = 5. \tag{4.5}$$

In the data structure, we have an public function'double evaluate(double x) const', which takes a given value $x$, and use the formula above to calculate the interpolated $y$.

- *Polynomial function:* In the ENDF database, a function can be described by a polynomial with degree $n - 1$ and coefficients $c_0, c_1, \ldots, c_n$. The function value is evaluated by:

$$y = \sum_{i=0}^{n-1} c_i \, x^i \tag{4.6}$$

The data structure for describing a polynomial function is given below:

```
struct ENDFPolynomialFunction {
    std::vector<double> coefficents;

    // Evaluate the polynomial function value at x
    double evaluate(double x) const;
};
```

The 'evaluate' function implements the evaluation of the function value at x.

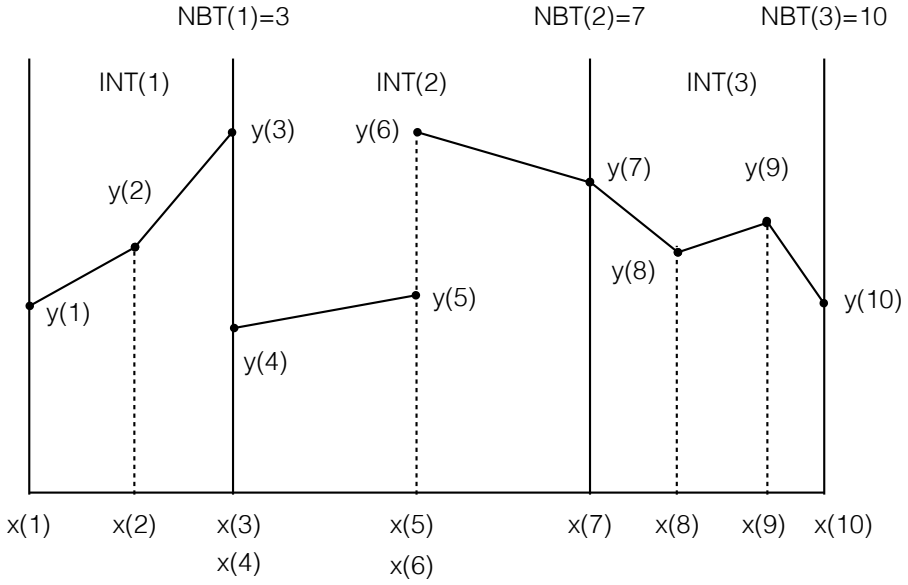



Figure 4.7: An example showing the ENDF interpolation rules of a function, the indices start from 1, but in C++ language, the indices start from 0

- *Function type:* To specify whether a function is an ENDF interpolation or a polynomial, an enumeration is introduced as followed.

```
enum class ENDFFunctionType {
    INTERPOLATION = 1,
    POLYNOMIAL = 2
};
```

### 4.2.2 Fission Yield Data Structures

The fission yield is an important property of the neutron data. For a fission nuclide, a fission can be prompt or delayed. Here, it discusses the data structures to describe the fission yield.

- *Fission spectrum:* The fission spectrum describes the average number of neutron released as a function of the incident neutron energy. Such a function is denoted in $\bar{\nu}(E)$. For prompt fission, it is described as $\bar{\nu}_p(E)$, and for delayed fission, the sum is

described as $\bar{\nu}_d(E)$. If there are $N$ precursor groups, the average number of neutron released from group $i$ is denoted as $\bar{\nu}_i(E)$, and

$$\bar{\nu}_d(E) \;=\; \sum_{i=1}^{N} \bar{\nu}_i(E). \tag{4.7}$$

The average total fission neutron yield is given by $\bar{\nu}(E)$ and

$$\bar{\nu}(E) \;=\; \bar{\nu}_p(E) + \bar{\nu}_d(E). \tag{4.8}$$

The fission spectrum can be described either as an ENDF interpolation function or a polynomial function. So the data structure is listed as followed.

```
struct Spectrum {
    // The function type
    ENDFFunctionType type;
    // The data for interpolation function
    ENDFInterpolationFunction interpFunc;
    // The data for polynomial function
    ENDFPolynomialFunction polyFunc;
};
```

- *Precursor group:* For precursor group $i$, there are the decay constant $\lambda_i$ and the group abundance $\alpha_i$. They can be either constant or energy dependent. So the precursors are described in the data structure as followed.

```
struct Precursor {
    // Decay constant, lambda
    double decayConst = 0.;
    // Decayed-group abundance, alpha
    double groupAbundance = 0.;
};
struct Precursors {
    // List of precursors
    std::vector<Precursor> precursors;
};
```

- *Neutron yield data:* The neutron yield data is packed into a single data structure.

The fission spectra are given in a C++ pointer. When the pointer is nullptr, then no such fission data is provided.

```
struct YieldData {
    ~YieldData();

    // Total neutron yield function
    Spectrum *total = nullptr;
    // Prompt neutron yield function
    Spectrum *prompt = nullptr;
    // Delayed neutron yield function
    Spectrum *delayed = nullptr;

    // Whether precursor is energy dependent
    bool isPrecursorEnergyDependent = false;
    // Energy independent precursors
    Precursors energyIndependentPrecursors;
    // Energy dependent precursors
    ENDFObjectInterpolationFunction< Precursors >
    energyDependentPrecursors;
};
```

### 4.2.3 Resonance Data Structure

- 

### 4.2.4 Reaction Data Structure

The reaction data structures taken from ENDF file 3 describe the 'background' cross sections of reactions, which does not depend on the energy of the incident neutrons. The type of the reaction is determined by its 'MT' number, whose definitions can be found in Table 4.2. The reaction cross sections are represented by the ENDF table with an 'ENDFInterpolationFunction' data structure.

```
struct ReactionData {
    // Reaction number
    long   MT = -1;
    // Mass-difference Q value
```

```
    double QM = 0.;
    // Reaction Q value from the lowest energy state
    double QI = 0.;
    // Complex or "breakup" reaction flag
    long   LR = -1;
    // ENDF interpolated function for cross sections
    ENDFInterpolationFunction xsec;
};
```

| MT | Symbol | Meaning | MT | Symbol | Meaning |
|---|---|---|---|---|---|
| 1 | (n,total) | total | 2 | (n,elastic) | elastic scattering |
| 3 | (n,inelastic) | inelastic scattering | 4 | (n,level) | level scattering |
| 5 | (n,other) | others | 11 | (n,2nd) | two neutrons plus one deuteron |
| 16 | (n,2n) | two neutrons | 17 | (n,3n) | three neutrons |
| 18 | (n,fission) | fission | 19 | (n,f) | f |
| 20 | (n,nf) | one neutron and f | 21 | (n,2nf) | two neutrons and f |
| 22 | (n,na) | one neutron and one alpha | 23 | (n,n3a) | one neutron and three alphas |
| 24 | (n,2na) | two neutrons and one alpha | 25 | (n,3na) | three neutrons and one alpha |
| 28 | (n,np) | one neutron and one proton | 29 | (n,n2a) | one neutron and two alphas |
| 30 | (n,2n2a) | two neutrons and two alphas | 32 | (n,nd) | one neutron and one deuteron |
| 33 | (n,nt) | one neutron and one triton | 34 | (n,n3he) | one neutron and three helium |
| 35 | (n,nd2a) | one neutron, deuteron and two alphas | 36 | (n,nt2a) | one neutron, triton and two alphas |
| 37 | (n,4n) | four neutrons | 38 | (n,3nf) | three neutrons and f |
| 41 | (n,2np) | two neutrons and one proton | 42 | (n,3np) | three neutrons and one proton |
| 44 | (n,n2p) | one neutron and two protons | 45 | (n,npa) | one neutron, proton and alpha |
| $50+i$ | (n,n$i$) | $i$th level, $1 \leq i \leq 40$ | 91 | (n,nc) | continuum scattering |
| 101 | (n,disappear) | | 102 | (n,gamma) | |
| 103 | (n,p) | | 104 | (n,d) | |
| 105 | (n,t) | | 106 | (n,3he) | |

| | | | | |
|---|---|---|---|---|
| 107 | (n,a) | | 108 | (n,2a) |
| 109 | (n,3a) | | 111 | (n,2p) |
| 112 | (n,pa) | | 113 | (n,t2a) |
| 114 | (n,d2a) | | 115 | (n,pd) |
| 116 | (n,pt) | | 117 | (n,da) |
| $600+i$ | (n,p$i$) | $0 \leq i \leq 48$ | 649 | (n,pc) |
| $650+i$ | (n,d$i$) | $0 \leq i \leq 48$ | 699 | (n,dc) |
| $700+i$ | (n,t$i$) | $0 \leq i \leq 48$ | 749 | (n,tc) |
| $750+i$ | (n,3he$i$) | $0 \leq i \leq 48$ | 799 | (n,3hec) |
| $800+i$ | (n,a$i$) | $0 \leq i \leq 48$ | 849 | (n,ac) |

Table 4.2: Meaning of the MT reaction number

The relationship of the cross sections with different MT numbers is summarized in Table.

### 4.2.5 Application Interfaces (APIs)

# CHAPTER 5

# NDLS: Nuclear Data Library System

## 5.1 General Description

The NDLS module stands for the nuclear data library system, which stores all ENDF-6 formatted data in a database. The benefit of this approach instead of traditional text file of ENDF is that it supports multi-processes access or even multi-operating systems access to database. So many applications can share the same database. The database supports regular SQL query interfaces, so it may be familiar to the existing client program.

## 5.2 Database ER Diagram and Schema

The NDLS uses a database management system (DBMS) to store the ENDF data. The basic unit in a DBMS is a table, which contains rows of data with a homogenous column structure, i.e. a spreadsheet in Excel. To make the ENDF data fits in a database, we furthers break the data into more fundamental units. These units are represented in great details in an Entity-Relationship (ER) diagram and the schema of each table, and give the receipt how to store the ENDF information in terms of entries in the tables in DBMS. For one looks for how to use the programming API, one can skip this section.

### 5.2.1 ER Diagram

The design of a database can be described by a graph called Entity-Relationship (ER) diagram. This section describes the design. The database contains 7 entities: Material, Description, Type, Header, Function, List, and Interpolation entities. Next we describe their meaning and the relationship between them in details.

The top level abstraction of objects in the database is a material, which is associated with a material number (MAT). According to ENDF-6 format, an evaluated material is

uniquely defined by the following numbers:

- MAT: a unique identifier for a nuclide whose cross sections are evaluated in the database
- NLIB: an integer indicating from where the file evaluated, such as ENDF/B, CENDL, JENDL,
- NVER: an integer indicating the major version of the evaluation,
- LREL: an integer indicating the minor release version of the evaluation,
- NSUB: an integer indicating the type of collections of the evaluation,
- NMOD: an integer indicating the modification flag,
- LDRV: an integer indicating the derivation flag for a given material,
- TEMP: a real number indicating the temperature of the material of evaluation.

Each material has a description associating with it. The description contains the following fields:

- ZSYMAM: the chemical symbol of the nuclide,
- ALAB: the originating laboratory,
- EDATE: the date of evaluation,
- AUTH: the authors names,
- REF: primary reference information,
- DDATE: original distribution date,
- RDATE: the data and number of last version to this evaluation,
- ENDATE: the master file entry date,
- HSUB: first few rows of evaluation metadata,
- Summary: some conclusive sentences about the evaluation,
- Description: additional comments given in the evaluation file.

Since each description is uniquely associated with a material, and a material evaluation has a description, there is a one-to-one the relationship between the material entity and the description entity.

Each material evaluation contains files the are identified by the file number (MF) and the reaction number (MT). We call the combination of the (MF, MT) pair as the properties of a type entity. There are multiple type entities associating with a material.

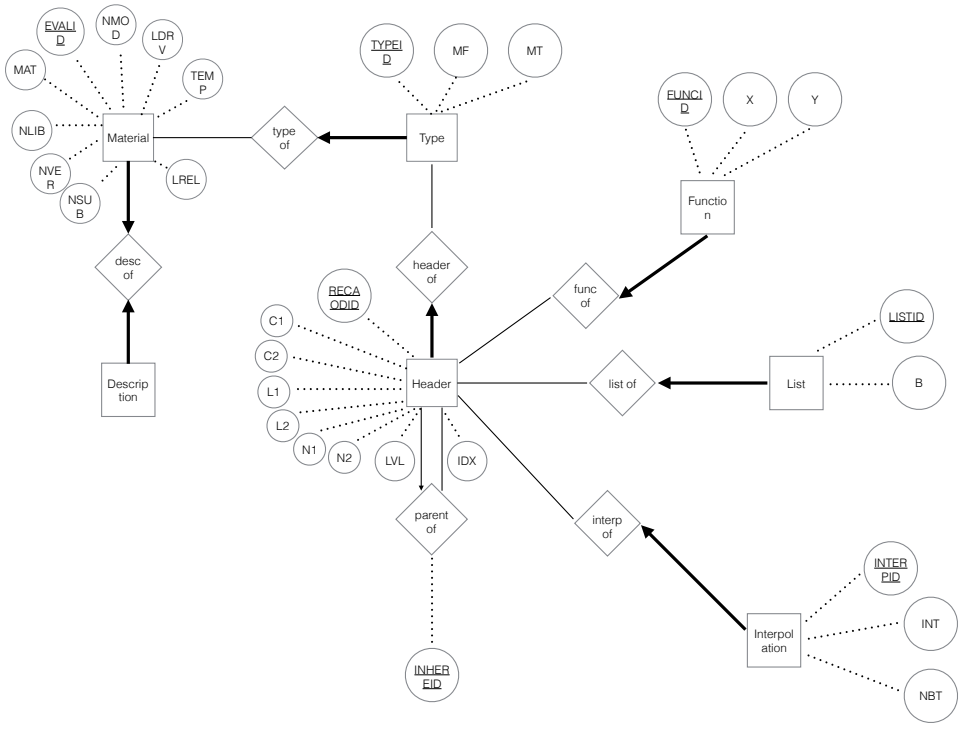



Figure 5.1: The ER diagram of the database

A header entity corresponds to a CONT record in the ENDF file, which contains six properties: C1, C2, L1, L2, N1, N2. In the ENDF text file, each CONT record is followed by any information for a list, a one-dimensional function or a two dimensional function. The abstractions of function, list or interpolation entities, discussed later, are designated to describe a ENDF list, one-dimensional function, or a two dimensional function. Then in the ENDF file, a header entity is followed by a function, a list, an interpolation entity, or another header entity. An illustration of the file structure is shown in Figure 5.2

Since a header entity could follow another header entity, there is an inherence relationship between the header entities. So we assign another two properties: the level (LVL) and the index (IDX) to indicate the parent-children relationship between headers and the brother and sister relationship between the header entities within a level.

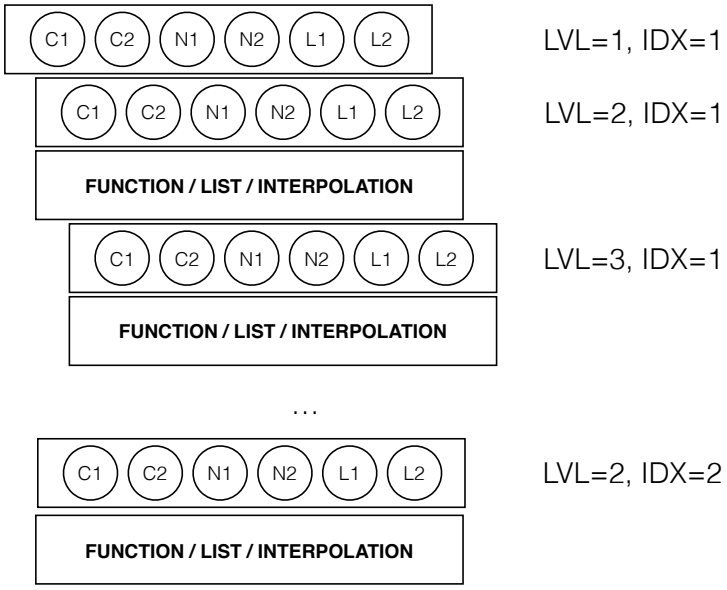



Figure 5.2: The header entities and the entities followed

A function entity contains two properties x and y, which describe a one-dimensional function. A list entity contains a property B, which describes an array of data. An interpolation entity contains two properties: NBT and INT, which is used to describe the ENDF interpolation law.

In the following sections, the database schema are given in more details.

### 5.2.2 Material Table

The material table contains important evaluation conditions for all evaluated data. According to the NJOY document, each evaluated data has a unique combination of MAT, NLIB, NVER, LREL, NSUB, NMOD, LDRV, and TEMP. We chose a self increasing integer EVALID as the primary key, which is generated at the time of inserting data into the database. The schema is shown in Program 5.1.

### 5.2.3 Description Table

The description table includes additional evaluation information of the evaluated data. The EVALID from the material table is the primary key. The schema is shown in Program 5.2.

```
CREATE TABLE `Material_Table` (
`EVALID` INTEGER NOT NULL,
`MAT` INTEGER NOT NULL,
`NLIB` INTEGER NOT NULL,
`NVER` INTEGER NOT NULL,
`LREL` INTEGER NOT NULL,
`NSUB` INTEGER NOT NULL,
`NMOD` INTEGER NOT NULL,
`LDRV` INTEGER NOT NULL,
`TEMP` REAL NOT NULL,
PRIMARY KEY(EVALID)
);
```

Program 5.1: SQL schema for material table

```
CREATE TABLE `Description_Table` (
`EVALID` INTEGER NOT NULL UNIQUE,
`ZSYMAM` TEXT,
`ALAB` TEXT,
`EDATE` TEXT,
`AUTH` TEXT,
`REF` TEXT,
`DDATE` TEXT,
`RDATE` TEXT,
`ENDATE` TEXT,
`HSUB` TEXT,
`Summary` TEXT,
`Description` TEXT,
PRIMARY KEY(EVALID),
FOREIGN KEY(`EVALID`) REFERENCES `Material_Table`(`EVALID`)
);
```

Program 5.2: SQL schema for description table

### 5.2.4 Type Table

The type tables includes all file section paris. A self increasing integer TYPEID is automatically generated at the time of inserting data into the database and serve as the primary key. The schema is shown in Program 5.3.

```
CREATE TABLE `Type_Table` (
`TYPEID` INTEGER NOT NULL,
`EVALID` INTEGER NOT NULL,
`MF` INTEGER NOT NULL,
`MT` INTEGER NOT NULL,
PRIMARY KEY(TYPEID),
FOREIGN KEY(`EVALID`) REFERENCES `Material_Table`(`EVALID`)
);
```

Program 5.3: SQL schema for type table

### 5.2.5 Header Table

The header table gives the header information for each record. A self increasing integer RECORDID is automatically generated at the time of inserting data into the database and serve as the primary key. The schema is shown in Program 5.4.

```
CREATE TABLE `Header_Table` (
`RECORDID` INTEGER NOT NULL,
`TYPEID` INTEGER NOT NULL,
`C1` REAL NOT NULL,
`C2` REAL NOT NULL,
`L1` REAL NOT NULL,
`L2` REAL NOT NULL,
`N1` REAL NOT NULL,
`N2` REAL NOT NULL,
`LVL` INTEGER NOT NULL,
`IDX` INTEGER NOT NULL,
PRIMARY KEY(RECORDID),
FOREIGN KEY(`TYPEID`) REFERENCES `Type_Table`(`TYPEID`)
);
```

Program 5.4: SQL schema for header table

### 5.2.6 Inheritance Table

The inheritance table is designated to describe the inheritance relationship between the header tables, and the schema of which is shown in Program 5.5.

```
CREATE TABLE `Inheritance_Table` (
`INHEREID` INTEGER NOT NULL,
`PARENTID` INTEGER NOT NULL,
`CHILDID` INTEGER NOT NULL,
PRIMARY KEY(INHEREID),
FOREIGN KEY(`PARENTID`) REFERENCES `Header_Table`(`RECORDID`),
FOREIGN KEY(`CHILDID`) REFERENCES `Header_Table`(`RECORDID`)
);
```

Program 5.5: SQL schema for inherence table

### 5.2.7 Function Table

The function table records all one-dimensional functions x-y pairs. The RECORDID of a header table it associates with is the primary key. The schema of the function table is shown in Figure 5.6.

```
CREATE TABLE `Function_Table` (
`FUNCID` INTEGER NOT NULL,
`RECORDID` INTEGER NOT NULL UNIQUE,
`X` BLOB NOT NULL,
`Y` BLOB NOT NULL,
PRIMARY KEY(FUNCID),
FOREIGN KEY(`RECORDID`) REFERENCES `Header_Table`(`RECORDID`)
);
```

Program 5.6: SQL schema for function table

### 5.2.8 List Table

The list table records all arrays of data in a list. The RECORDID of a header table it associates with is the primary key. The schema of the function table is shown in Figure 5.7.

### 5.2.9 Interpolation Table

The interpolation table records all ENDF interpolation rules. The RECORDID of a header table it associates with is the primary key. The schema of the function table is shown in Figure 5.8.

```
CREATE TABLE `List_Table` (
`LISTID` INTEGER NOT NULL,
`RECORDID` INTEGER NOT NULL UNIQUE,
`B` BLOB NOT NULL,
PRIMARY KEY(LISTID),
FOREIGN KEY(`RECORDID`) REFERENCES `Header_Table`(`RECORDID`)
);
```

Program 5.7: SQL schema for list table

```
CREATE TABLE `Interpolation_Table` (
`INTERPID` INTEGER NOT NULL,
`RECORDID` INTEGER NOT NULL UNIQUE,
`INT` BLOB NOT NULL,
`NBT` BLOB NOT NULL,
PRIMARY KEY(INTERPID),
FOREIGN KEY(`RECORDID`) REFERENCES `Header_Table`(`RECORDID`)
);
```

Program 5.8: SQL schema for interpolation table

## 5.3 C++ Data Structure & Application Interfaces

The program is implemented native in C++ programming language, so in this section, we discuss the data structures first then the application interfaces.

### 5.3.1 Data Structure

### 5.3.2 Application Interfaces (APIs)

```
class NDLS {
public:
    NDLS(const std::string& dbFilepath);
    ~NDLS();

    bool open(const std::string& dbFilepath);
    void close();
    bool insertENDFFile(const std::string& endfFilepath);
}
```

Program 5.9: C++ public APIs for NDLS module

# CHAPTER 6

# LS: Linearization System

In this chapter we discuss an important topic in converting complex function into its linearized form. This is important since, the linearized function costs less computations. For the importance in its own, we use a whole chapter to discuss it.

# CHAPTER 7

# XLS: Cross Section Linearization System

The given cross sections in the reactions may not be linearized. As a result, they can be easily summed or accumulated.

## 7.1 Algorithm

The ENDF interpolation tabular data points are divided into regions, each of which uses an integer from 1 to 5 to indicate the method of interpolation.

### 7.1.1 Histogram

If the interpolation law number is 1, the method of interpolation represents a histogram. The points on the region boundaries will remain the same, and each point internal to the region will be accompanied by another point with a $y$ value equal to the previous point.

### 7.1.2 Linear-Linear Interpolation

If the interpolation law number is 2, the method of interpolation is linear to linear. All points will remain the same. No modifications will be applied.

### 7.1.3 Other Laws of Interpolation

If the interpolation law number betweens 3 and 5, additional data points may be necessary to be inserted into the region to make the error of linearization below a certain level of requirement, for example $0.1\%$. The algorithm applied is called the method of inverted stack.

### 7.1.4 Removal of Duplicated Points

Sometimes, the data points after the linearization may have duplications. That is the case that for two points: $(x_1, y_1)$ and $(x_2, y_2)$, we have:

$$x_1 = x_2, \tag{7.1}$$

$$|y_1 - y_2| < \epsilon, \tag{7.2}$$

where $\epsilon$ is small value such as $1 \times 10^{-10}$ barn.

## 7.2 C++ Data Structures & APIs

The XLS class has a static method called 'linearizeENDFInterpFunc', which takes a ENDF interpolation table function and return a vector of cross section data points.

```
struct XLSXsecPair {
    double energyEv  = 0.;
    double sigmaBarn = 0.;
};

// Discontinuity of cross section may present
struct XLSXsec {
    std::vector<XLSXsecPair> xspairs;
};

// The cross section linearization system will take
// an ENDFInterpolationFunction representation
// of a cross section data, and linearlize it
// into an XLSXsec data format, the tolerance in the fraction
// of the relative error is given
class XLS {
public:
    static XLSXsec linearizeENDFInterpFunc
    (const ENDFInterpolationFunction& endf, double tol);
};
```

# CHAPTER 8

# XRS: Cross Section Reconstruction System

The evaluated nuclear data are not easily used by the neutron transport softwares. So further processing are necessary. One important step is to discretize the cross sections on an energy grid with the cross sections linearized on it. In this chapter, we first discuss the topic of resonance cross section reconstruction and then how to combine the resonance with tabulated cross sections.

## 8.1 Resolved Resonance

In this section, we discuss the mathematical theories behind cross section evaluations and the method of linearization. First, we focus on the resolved resonance regions. Second, the unresolved resonance region is discussed.

### 8.1.1 SLBW: Single-level Breit-Wigner

The Single-Level Breit-Wigner (SLBW) is the most simple ways of describing the resonance peaks. First the format of the data provided by the ENDF database is presented. Second, the mathematical theories behind for how to evaluating cross sections at a given energy is discussed. Third, how to write a computer algorithm is discussed.

#### 8.1.1.1 Data Format

The data given from the ENDF database are discussed now. First, we list some scalar variables:

| Variable Name | ENDF Name | Type | Meaning |
|---|---|---|---|
| $I$ | SPI | floating | the total spin |
| $\hat{a}$ | AP | floating | scattering radius in units of $10^{-12}$ cm |
| $N$ | NLS | integer | the number of neutron orbital angular momentum |
| $A$ | AWRI | floating | the ratio of the mass of the isotope to that of a neutron |
| $a$ | N/A | floating | the channel radius |
| $f$ | NAPS | integer | radius choice flag |

There are $N$ angular momenta. For each angular momentum $l_i$ with $1 \leq i \leq N$, there are some variables are defined depending on the index $i$:

| Variable Name | ENDF Name | Type | Meaning |
|---|---|---|---|
| $l_i$ | L | floating | angular momentum at index $i$ |
| $M_i$ | NRS | integer | number of resolved resonances for $l_i$ |
| $Q_i$ | QX | floating | the Q-value used in calculating the penetrability factor |

There are some variables defined for the resonance defined at the resonance index $1 \leq r \leq M_i$ for the angular momentum $l_i$. They are listed in the following table:

| Variable Name | ENDF Name | Type | Meaning |
|---|---|---|---|
| $E_{ir}$ | ER | floating | the resonance energy measured in the laboratory system for $r$th resonance and angular momentum $l_i$ |
| $J_{ir}$ | AJ | floating | the spin for $r$th resonance and angular momentum $l_i$ |
| $\Gamma^0_{ir}$ | GT | floating | the resonance total width evaluated at energy $E_{ir}$ |
| $\Gamma^0_{n,ir}$ | GN | floating | the neutron width evaluated at energy $E_{ir}$ |
| $\Gamma^0_{\gamma,ir}$ | GG | floating | radiation width evaluated at energy $E_{ir}$ |
| $\Gamma^0_{f,ir}$ | GF | floating | fission width evaluated at energy $E_{ir}$ |
| $\sigma_{m,ir}$ | N/A | floating | maximum cross section at energy $E_{ir}$ |

#### 8.1.1.2 Mathematical Theory

There are several commonly used types of representations for describing the resolved resonances. The first one is the Single-Level Breit-Wigner Representation (SLBW), the resonance elastics cross section ($\sigma_n$), the fission cross section ($\sigma_f$), the capture cross section ($\sigma_\gamma$), and the potential scattering cross section ($\sigma_p$) are given by:

$$\begin{aligned}
\sigma_n(E) &= \sigma_p(E) \\
&\quad + \sum_{i=1}^{N}\sum_{r=1}^{M_i} \sigma_{m,ir}(E) \left\{ \left[ \cos 2\phi_{l_i}(E) - (1 - \frac{\Gamma_{n,ir}(E)}{\Gamma_{ir}(E)}) \right] \psi(x_{ir}(E)) \right. \\
&\quad \left. + \sin 2\phi_{l_i}(E)\, \chi(x_{ir}(E)) \right\}, \qquad (8.1) \\
\sigma_f(E) &= \sum_{i=1}^{N}\sum_{r=1}^{M_i} \sigma_{m,ir}(E) \frac{\Gamma^0_{f,ir}}{\Gamma_{ir}(E)} \psi(x_{ir}(E)), \qquad (8.2) \\
\sigma_\gamma(E) &= \sum_{i=1}^{N}\sum_{r=1}^{M_i} \sigma_{m,ir}(E) \frac{\Gamma^0_{\gamma,ir}}{\Gamma_{ir}(E)} \psi(x_{ir}(E)), \qquad (8.3) \\
\sigma_p(E) &= \sum_{i=1}^{N} \frac{4\pi}{k(E)^2} (2l_i + 1) \sin^2 \phi_{l_i}(E), \qquad (8.4)
\end{aligned}$$

where the given quantities are:

$$
\begin{aligned}
E &= \text{incident laboratory energy in units of eV}, && (8.5)\\
A &= \text{isotope weight ratio to neutron}, && (8.6)\\
N &= \text{number of angular momentum}, && (8.7)\\
f &= \text{radius choice flag either 0 or 1}, && (8.8)\\
l_i &= i\text{th angular momentum}, \ 1 \le i \le N, && (8.9)\\
M_i &= \text{number of resonance associated with } l_i, && (8.10)\\
\hat{a} &= \text{scattering radius}, && (8.11)\\
J_{ir} &= \text{spin for angular momentum } l_i \text{ and } r\text{th resonance ,} \\
&\quad 1 \le i \le N, \ 1 \le r \le M_i, && (8.12)\\
I &= \text{total spin}, && (8.13)\\
E_{ir} &= \text{energy at the center of resonance}, \ 1 \le i \le N, \ 1 \le r \le M_i, && (8.14)\\
\Gamma^0_{ir} &= \text{resonance total width at } E_{ir}, \ 1 \le i \le N, \ 1 \le r \le M_i, && (8.15)\\
\Gamma^0_{n,ir} &= \text{resonance neutron width at } E_{ir}, \ 1 \le i \le N, \ 1 \le r \le M_i, && (8.16)\\
\Gamma^0_{\gamma,ir} &= \text{resonance capture width at } E_{ir}, \ 1 \le i \le N, \ 1 \le r \le M_i, && (8.17)\\
\Gamma^0_{f,ir} &= \text{resonance fission width at } E_{ir}, \ 1 \le i \le N, \ 1 \le r \le M_i, && (8.18)
\end{aligned}
$$

The energy dependent resonance widths are:

$$
\begin{aligned}
\Gamma_{n,ir}(E) &= \frac{P_{l_i}(E)\Gamma^0_{n,ir}}{P_{l_i}(|E_{ir}|)}, \ 1 \le i \le N, \ 1 \le r \le M_i, && (8.19)\\
\Gamma_{ir}(E) &= \Gamma_{n,ir}(E) + \Gamma^0_{ir} - \Gamma^0_{n,ir}, \ 1 \le i \le N, \ 1 \le r \le M_i, && (8.20)
\end{aligned}
$$

and the derived quantities are:

$$
\begin{aligned}
k(E) &= (2.196771 \times 10^{-3}) \frac{A}{A+1} \sqrt{E}, && (8.21)\\
g_{ir} &= \frac{2J_{ir}+1}{4I+2},\ 1 \le i \le N,\ 1 \le r \le M_i, && (8.22)\\
\sigma_{m,ir}(E) &= \frac{4\pi}{k(E)^2} g_{ir} \frac{\Gamma_{n,ir}(E)}{\Gamma_{ir}(E)},\ 1 \le i \le N,\ 1 \le r \le M_i, && (8.23)\\
a &= \begin{cases} 0.123\ A^{1/3} + 0.08, & f = 0 \\ \hat{a}, & f = 1 \end{cases}, && (8.24)\\
\rho(E) &= k(E) a, && (8.25)\\
\hat{\rho}(E) &= k(E) \hat{a}, && (8.26)\\
E'_{ir}(E) &= E_{ir} + \frac{S_{l_1}(|E_{ir}|) - S_{l_i}(E)}{2P_{l_i}(|E_{ir}|)} \Gamma^0_{n,ir},\ 1 \le i \le N,\ 1 \le r \le M_i, && (8.27)\\
x_{ir}(E) &= \frac{2(E - E'_{ir}(E))}{\Gamma_{ir}(E)},\ 1 \le i \le N,\ 1 \le r \le M_i, && (8.28)\\
\psi(x) &= \frac{1}{1+x^2}, && (8.29)\\
\chi(x) &= \frac{x}{1+x^2}. && (8.30)
\end{aligned}
$$

The penetration factors are defined here:

$$
\begin{aligned}
P_0(E) &= \rho(E), && (8.31)\\
P_1(E) &= \frac{\rho(E)^3}{1+\rho(E)^2}, && (8.32)\\
P_2(E) &= \frac{\rho(E)^5}{9+3\rho(E)^2+\rho(E)^4}, && (8.33)\\
P_3(E) &= \frac{\rho(E)^7}{225+45\rho(E)^2+6\rho(E)^4+\rho(E)^6}, && (8.34)\\
P_4(E) &= \frac{\rho(E)^9}{11025+1575\rho(E)^2+135\rho(E)^4+10\rho(E)^6+\rho(E)^8}. && (8.35)
\end{aligned}
$$

The phase shifts are defined here:

$$\begin{aligned}
\phi_0(E) &= \hat{\rho}(E), && (8.36)\\
\phi_1(E) &= \hat{\rho}(E) - \tan^{-1}\hat{\rho}(E), && (8.37)\\
\phi_2(E) &= \hat{\rho}(E) - \tan^{-1}\frac{3\hat{\rho}(E)}{3-\hat{\rho}(E)^2}, && (8.38)\\
\phi_3(E) &= \hat{\rho}(E) - \tan^{-1}\frac{15\hat{\rho}(E)-\hat{\rho}(E)^3}{15-6\hat{\rho}(E)^2}, && (8.39)\\
\phi_4(E) &= \hat{\rho}(E) - \tan^{-1}\frac{105\hat{\rho}(E)-10\hat{\rho}(E)^3}{105-45\hat{\rho}(E)^2+\hat{\rho}(E)^4}. && (8.40)
\end{aligned}$$

The shift factors are defined here:

$$\begin{aligned}
S_0(E) &= 0, && (8.41)\\
S_1(E) &= -\frac{1}{1+\rho(E)^2}, && (8.42)\\
S_2(E) &= -\frac{18+3\rho(E)^2}{9+3\rho(E)^2+\rho(E)^4}, && (8.43)\\
S_3(E) &= -\frac{675+90\rho(E)^2+6\rho(E)^4}{225+45\rho(E)^2+6\rho(E)^4+\rho(E)^6}, && (8.44)\\
S_4(E) &= -\frac{44100+4725\rho(E)^2+270\rho(E)^4+10\rho(E)^6}{11025+1575\rho(E)^2+135\rho(E)^4+10\rho(E)^6+\rho(E)^8}. && (8.45)
\end{aligned}$$

Plug in all definitions into the formula for cross sections. We have simplified versions:

$$\begin{aligned}
\sigma_n(E) &= \sigma_p(E) + \frac{\pi}{k(E)^2}\sum_{i=1}^{N}\sum_{r=1}^{M_i} g_{ir} \\
&\quad \frac{\Gamma_{n,ir}(E)^2 - 2\Gamma_{n,ir}(E)\Gamma_{ir}(E)\sin^2\phi_{l_i}(E) + 2(E-E'_{ir}(E))\sin 2\phi_{l_i}(E)}{\frac{1}{4}\Gamma_{ir}^2(E) + (E-E'_{ir}(E))^2} && (8.46)\\
\sigma_\gamma(E) &= \frac{\pi}{k(E)^2}\sum_{i=1}^{N}\sum_{r=1}^{M_i} g_{ir}\frac{\Gamma_{n,ir}(E)\Gamma^0_{\gamma,ir}}{\frac{1}{4}\Gamma_{ir}^2(E) + (E-E'_{ir}(E))^2}, && (8.47)\\
\sigma_f(E) &= \frac{\pi}{k(E)^2}\sum_{i=1}^{N}\sum_{r=1}^{M_i} g_{ir}\frac{\Gamma_{n,ir}(E)\Gamma^0_{f,ir}}{\frac{1}{4}\Gamma_{ir}^2(E) + (E-E'_{ir}(E))^2}, && (8.48)\\
\sigma_p(E) &= \sum_{i=1}^{N}\frac{4\pi}{k(E)^2}(2l_i+1)\sin^2\phi_{l_i}(E). && (8.49)
\end{aligned}$$

### 8.1.2 MLBW: Multilevel Breit-Wigner

#### 8.1.2.1 Data Format

The data required by MLBW calculation are the same as that of SLBW.

#### 8.1.2.2 Mathematical Theory

### 8.1.3 R-M: Reich Moore

#### 8.1.3.1 Data Format

#### 8.1.3.2 Mathematical Theory

# CHAPTER 9

# DBS: Doppler Broadening System

Doppler broadening is an important physical phenomena need to take consideration for neutron transport calculation to get the collision probability correct. We first discuss the mathematical background of this issue, and then give the theory behind the widely used SIGMA1 method.

## 9.1 Mathematical Background of Doppler Broadening

Let $\vec{v}$ be the velocity of the neutron, let $\vec{V}$ be the velocity of the target nucleus, then we can define an averaged collision cross section for target nuclide moving with a probability $f(\vec{V})$.

$$\int_{R^3} f(\vec{V})d\vec{V} \;=\; 1, \tag{9.1}$$

where if $\vec{V} = (x, y, z)$ or $\vec{V} = (r\cos\theta\sin\phi, r\sin\theta\sin\phi, r\cos\phi)$, for a general function $K(\vec{V}) = K(x, y, z)$:

$$\int_{R^3} K(\vec{V})d\vec{V} \;=\; \int_0^\infty\int_0^\infty\int_0^\infty K(\vec{V})dxdydz, \tag{9.2}$$

$$\int_{R^3} K(\vec{V})d\vec{V} \;=\; \int_0^\infty\int_0^\pi\int_0^{2\pi} K(\vec{V})r^2\sin\phi d\theta d\phi dr, \tag{9.3}$$

the averaged cross section is:

$$\bar{\sigma}(\vec{v}) \;=\; \frac{1}{|\vec{v}|}\int_{R^3} |\vec{v}-\vec{V}|\sigma(|\vec{v}-\vec{V}|)f(\vec{V})d\vec{V}. \tag{9.4}$$

In the thermal motion case, the distribution function $f$ follows the free-gas thermal motion law, which at temperature $T$ is:

$$f(\vec{V}) = f^*(|\vec{V}|, T) = (\frac{M}{2\pi kT})^{3/2} e^{-M|\vec{V}|^2/2kT}, \tag{9.5}$$

which depends only on the magnitude of $\vec{V}$. In this case we are able to define a cross section only depends on the magnitude of neutron velocity $\vec{v}$:

$$\bar{\sigma}(|\vec{v}|, T) = \frac{1}{|\vec{v}|} \int_{R^3} |\vec{v} - \vec{V}| \sigma(|\vec{v} - \vec{V}|) f^*(|\vec{V}|, T) d\vec{V}. \tag{9.6}$$

If we define $v_r = |\vec{v} - \vec{V}|$, and denote $v$ and $V$ as the speed of $\vec{v}$ and $\vec{V}$:

$$v = |\vec{v}|, \tag{9.7}$$

$$V = |\vec{V}|. \tag{9.8}$$

It is possible to rewrite the integral in scalar quantities as:

$$\bar{\sigma}(v, T) = \frac{2\pi}{v^2} \int_0^\infty \int_{|v-V|}^{v+V} v_r^2 \sigma(v_r) V f^*(V, T) dv_r dV. \tag{9.9}$$

Change the order of integration, we get:

$$\bar{\sigma}(v, T) = \frac{2\pi}{v^2} \int_0^\infty \int_{|v-v_r|}^{v+v_r} v_r^2 \sigma(v_r) V f^*(V, T) dV dv_r. \tag{9.10}$$

If we define the thermal speed $v_{th} = \sqrt{kT/M}$, $\omega = v/\sqrt{2}v_{th}$, and $\omega_r = v_r/\sqrt{2}v_{th}$:

$$\bar{\sigma}(\omega, T) = \frac{1}{\omega^2\sqrt{\pi}} \int_0^\infty \omega_r^2 \sigma(\omega_r) \left( e^{-(\omega-\omega_r)^2} - e^{-(\omega+\omega_r)^2} \right) d\omega_r. \tag{9.11}$$

In terms of the more frequently used energy quantities of neutron $E = \frac{1}{2}mv^2$, where $m$ is the neutron mass. We define the additional quantities:

$$\alpha = \frac{M}{2kT}, \tag{9.12}$$

$$x = \sqrt{\alpha}\, v = \sqrt{\alpha \frac{2E}{m}} = \sqrt{\frac{ME}{mkT}}, \tag{9.13}$$

where $x$ is dimensionless. We define $\sigma(x,T)$ as the cross section as a function of the variable $x$ and temperature $T$, the formula is rewrite as:

$$x^2\sigma(x,T) = \frac{1}{\sqrt{\pi}}\int_0^\infty x_r^2\sigma(x_r,0)\left(e^{-(x-x_r)^2}-e^{-(x+x_r)^2}\right)dx_r, \tag{9.14}$$

in which we have $x=\omega$ and $x_r=\omega_r$. In a more general case, if we start at temperature $T_1$ and want to get doppler broadened cross section at temperature $T_2$, and define $\alpha = M/2k(T_2-T_1)$, and $x=\sqrt{ME/mk(T_2-T_1)}$, we have:

$$x^2\sigma(x,T_2) = \frac{1}{\sqrt{\pi}}\int_0^\infty x_r^2\sigma(x_r,T_1)\left(e^{-(x-x_r)^2}-e^{-(x+x_r)^2}\right)dx_r. \tag{9.15}$$

For simplicity, if we change the variable $x_r, x$ by $x, y$, we have:

$$y^2\sigma(y,T_2) = \frac{1}{\sqrt{\pi}}\int_0^\infty x^2\sigma(x,T_1)\left(e^{-(y-x)^2}-e^{-(y+x)^2}\right)dx, \tag{9.16}$$

which is the formula we usually see in the textbooks.

## 9.2 Theory of SIGMA1 Method

The SIGMA1 method is probably the most widely used algorithm in the doppler broadening codes. It starts from a linearly interpolated cross section data points, which appears in the PENDF format in the NJOY system. We first define this linearly interpolated data points and derive the algorithm on it.

### 9.2.1 Grid for Data Points

Given the 'cold' cross section on a $x$ grid $(x_n,\sigma_n), n=1\ldots N$ with additional data points:

$$x_0 = 0, \tag{9.17}$$

$$x_{N+1} = \infty. \tag{9.18}$$

Let the cross section have a $1/x$ interpolation below the given data points, i.e. in the interval $(0,x_1]$/ The cross section is constant above the given data points, i.e. in the interval $(x_N,\infty)$. Therefore, the cross sections as a function of $x$ at the cold temperature $T_1$ are

given by:

$$\sigma(x,T_1) \;=\; \begin{cases} \frac{x_1\sigma_1}{x} : 0 < x \le x_1 \\ \sigma_n + s_n(x^2 - x_n^2) : x_n < x \le x_{n+1}, \;\; 1 \le n \le N-1 \\ \sigma_N : x_N < x < \infty \end{cases} , \quad (9.19)$$

where the slope is given by:

$$s_n \;=\; \frac{\sigma_{n+1} - \sigma_n}{x_{n+1}^2 - x_n^2}, \;\; 1 \le n \le N-1. \quad (9.20)$$

### 9.2.2 SIGMA1 Algorithm

The doppler broadened cross section at 'hot' temperature $T_2$ is given by the formula:

$$\sigma(y,T_2) \;=\; \sigma^*(y,T_2) - \sigma^*(-y,T_2), \quad (9.21)$$

$$\sigma^*(y,T_2) \;=\; \frac{1}{\sqrt{\pi}\,y^2}\int_0^\infty x^2\,\sigma(x,T_1)\,e^{-(x-y)^2}dx. \quad (9.22)$$

Plug in the formula Equation (9.19) for interpolated 'cold' cross section formula:

$$\begin{aligned} \sigma^*(y,T_2) \;=\; & x_1\sigma_1\frac{1}{\sqrt{\pi}\,y^2}\int_0^{x_1} x\,e^{-(x-y)^2}dx + \\ & \frac{1}{\sqrt{\pi}\,y^2}\sum_{n=1}^{N-1}\int_{x_n}^{x_{n+1}} x^2\left[\sigma_n + s_n(x^2 - x_n^2)\right]e^{-(x-y)^2}dx + \\ & \sigma_N\frac{1}{\sqrt{\pi}\,y^2}\int_{x_N}^\infty x^2\,e^{-(x-y)^2}dx \end{aligned} \quad (9.23)$$

Next, we define the $H$ function for simplifying the doppler broadening formula:

$$H_m(a,b) \;=\; \frac{1}{\sqrt{\pi}}\int_a^b z^m e^{-z^2}\,dz, \quad (9.24)$$

with the eqivalent representation:

$$H_m(a,b) \;=\; F_m(a) - F_m(b), \quad (9.25)$$

$$F_m(a) \;=\; \frac{1}{\sqrt{\pi}}\int_a^\infty z^m e^{-z^2}dz. \quad (9.26)$$

The Taylor expansions for the case that $a$ close to $b$ are:

$$\begin{aligned} H_m(a,b) &= F_m(a) - F_m(b), && (9.27)\\ &= -F'_m(a)(a-b) + \mathcal{O}((a-b)^2), && (9.28)\\ &= \frac{1}{\sqrt{\pi}} a^m e^{-z^2}(a-b) + \mathcal{O}((a-b)^2). && (9.29) \end{aligned}$$

The recursive formula for the $H$ function can be written as:

$$\begin{aligned} F_0(a) &= \frac{1}{2}\,\mathrm{erfc}(a), && (9.30)\\ F_1(a) &= \frac{1}{2\sqrt{\pi}}\, e^{-a^2}, && (9.31)\\ F_m(a) &= \frac{n-1}{2} F_{m-2}(a) + a^{m-1} F_1(a),\ m \geq 2, && (9.32) \end{aligned}$$

and the formula for lower order $F_n$ are:

$$\begin{aligned} F_2(a) &= \frac{1}{4}\,\mathrm{erfc}(a) + \frac{1}{2\sqrt{\pi}}\, a\, e^{-a^2}, && (9.33)\\ F_3(a) &= \frac{1}{2\sqrt{\pi}}\, \left(1 + a^2\right)\, e^{-a^2}, && (9.34)\\ F_4(a) &= \frac{3}{8}\,\mathrm{erfc}(a) + \frac{1}{2\sqrt{\pi}} \left(\frac{3}{2}a + a^3\right)\, e^{-a^2}. && (9.35) \end{aligned}$$

As a result of this, formula for $\sigma^*$ is simplified as:

$$\sigma^*(y, T_2) = \sum_{n=1}^{N-1} \left[A_n(y)(\sigma_n - s_n x_n^2) + B_i(y) s_n\right] + x_1 \sigma_1 \left[C(y)\right] + \sigma_N \left[B_N(y)\right], \quad (9.36)$$

where we define $A_i(y)$ and $B_i(y)$ as:

$$\begin{aligned} H_{m,n}(y) &= H_m(x_n - y, x_{n+1} - y), && (9.37)\\ A_n(y) &= \frac{1}{y^2} H_{2,n}(y) + \frac{2}{y} H_{1,n}(y) + H_{0,n}(y), && (9.38)\\ B_n(y) &= \frac{1}{y^2} H_{4,n}(y) + \frac{4}{y} H_{3,n}(y) + 6H_{2,n}(y) + 4yH_{1,n}(y) + y^2 H_{0,n}(y), && (9.39)\\ C(y) &= \frac{1}{y^2} H_{1,0}(y) + \frac{1}{y} H_{0,0}(y). && (9.40) \end{aligned}$$

There is a special case for $n = N$:

$$\begin{aligned} H_{m,N}(y) &= H_m(x_N - y, \infty), \\ &= F_m(x_N - y) - F_m(\infty), \\ &= F_m(x_N - y) - \frac{1}{\sqrt{\pi}} \int_{\infty}^{\infty} z^m e^{-z^2} dz, \\ &= F_m(x_N - y). \end{aligned} \tag{9.41}$$

## 9.3 SIGMA1 Code Approximation

In earlier computer implementation, for $y < 64$, there is an additional approximation that used. The key is to assume that:

$$\int_{x_n}^{x_{n+1}} x^2 \left[\sigma_n + s_n(x^2 - x_n^2)\right] e^{-(x-y)^2} dx \approx \int_{x_n}^{x_{n+1}} x^2 \frac{\sigma_n + \sigma_{n+1}}{2} e^{-(x-y)^2} dx. \tag{9.42}$$

With this approximation,

$$\sigma^*(y, T_2) \approx \sum_{n=1}^{N-1} \frac{\sigma_n + \sigma_{n+1}}{2} \left[A_n(y)\right] + x_1\sigma_1 \left[C(y)\right] + \sigma_N \left[B_N(y)\right]. \tag{9.43}$$

No justification is provided here. In the OpenJOY code, the version with no approximations is used.

## 9.4 C++ Data Structure & Application Interfaces

### 9.4.1 Data Structure

The operation of doppler broadening is applied pairs of energy and cross section. A cross section pair is defined in the following data structure:

```
struct ENDFXsecPair {
    double energyEv  = 0.;
    double sigmaBarn = 0.;
};
```

To avoid potential confusion in the units of energies, we explicitly mark the unit at the end of variable names.

### 9.4.2 Application Interfaces (APIs)

The C++ class DBS has a few static methods (currently only one) that take a vector (C++ standard library) of pairs of energy and cross section, the mass ratio of target nuclide to neutron, and a *positive* temperature difference between the broadened ('hot') cross section and the un-broadened ('cold') cross section. The vector of pairs of energy and cross section is taken by reference. All methods will modify the cross section in-place, i.e. the member

```
class DBS {
public:
    static void proceedWithSigma1
    (std::vector<ENDFXsecPair>& xspairs,
     double massRatio, double tempK);
};
```

Program 9.1: C++ public APIs for DBS module

'sigmaBarn' in the elements of 'xspairs' will be modified to the doppler broadened cross sections.

#### 9.4.2.1 Method: proceedWithSigma1

The 'proceedWithSigma1' method will doppler broaden the cross section with the SIGMA1 method as discussed in previous sections.

# CHAPTER 10

# ALS: Angle Linearization System

## 10.1 Emitted Particle Angular Distribution

There are many reactions from which particles are emitted. Refer to Table 4.2 for the meaning of the reactions. The direction of flight of the emitted particle is described by the angular distributions. The scattered particles are usually to be azimuthal symmetric, which means that for incoming direction $\hat{\Omega}$ (a unit vector) and outgoing direction $\hat{\Omega}'$ (a unit vector). The probability of emitting depends on only on the cosine of the angular between $\hat{\Omega}$ and $\hat{\Omega}'$, which is defined as:

$$\mu \;=\; \hat{\Omega}\cdot\hat{\Omega}',\; -1 \le \mu \le 1. \tag{10.1}$$

The distribution function $f(\mu)$ is normalized with the condition:

$$\int_{-1}^{1} f(\mu)\, d\mu \;=\; 1. \tag{10.2}$$

The angular distribution in general depends on the energy of outgoing particle as well. Even, the energy and the angular cosine may couple together. In this chapter, the discussion treats solely the angular distribution. In this chapter, we discuss the approaches to represent the angular distribution, and the linearization approach to make it suitable for computation. First we discuss the representation formats in the ENDF data files. Then, we discuss the algorithm to convert them into computation friendly format.

## 10.2 Representation in ENDF Data File

In ENDF data file, the angular distribution is represented in either a Legendre polynomial expansion or the ENDF interpolation table.

### 10.2.1 Legendre Expansion

For functions $f(\mu)$ defined in the interval $[-1, 1]$, there is an orthogonal expansion (in the sense of mathematics) can be used to approximate the function:

$$f(\mu) \approx \sum_{l=0}^{N} a_l \frac{2l+1}{2} P_l(\mu), \tag{10.3}$$

where $P_l(\mu)$ is the $l$th order Legendre polynomial. There is an recursive definition for $P_l(\mu)$:

$$P_0(\mu) = 1, \tag{10.4}$$

$$P_1(\mu) = \mu, \tag{10.5}$$

$$P_l(\mu) = \frac{2l-1}{l}\mu P_{l-1}(\mu) - \frac{l-1}{l} P_{l-2}(\mu),\ l \geq 2. \tag{10.6}$$

The coefficients $a_l$ can be calculated by the formula:

$$a_l = \int_{-1}^{-1} f(\mu) P_l(\mu)\, d\mu. \tag{10.7}$$

Apply the normalization 10.2, the Legendre coefficients are assumed to satisfy:

$$a_0 = \frac{1}{2}. \tag{10.8}$$

In the ENDF data file, the Legendre coefficients $a_1, a_2, \ldots a_N$ are given.

### 10.2.2 ENDF Interpolation Laws

As discussed in previous chapter, an ENDF interpolation function can provided to describe the angular distribution functions.

## 10.3 Linearization of Angular Distribution

To make the angular distribution suitable for computation, the function needs to be linearized. The linearization used in OpenJOY follows the 'Inverted Stack' algorithm in previous discussions of cross section linearization. At the beginning, two function points evaluated at $\mu = \pm 1$ are inserted in the initial stack at the beginning of the algorithm.

# CHAPTER 11

# CFS: Computational Format System

The data proceeded by previous module are needed to be packed in a computational friendly format. In this chapter, we discuss the design in details. The C++ data structure is described as well. The design goal of the CFS module is to make it easy for programmers to build transport software on top of this module. To achieve that goal, we describe the data with some concepts that we think will give the programmers intuitively access to the data for their transport calculations.

## 11.1 Fundamental Data Structure Design

To make our data structures as intuitive and as expressive as possible, some fundamental data structures need to be designed first. In this section, we discuss them.

### 11.1.1 ENDF Interpolation Function

The CFS module directly inherits the interpolation function from the original ENDF library, which mathematically describe a mapping from a number to an object:

$$f \;:\; x \in \{\text{ number }\} \;\rightarrow\; y \in \{\text{ any data type }\}. \tag{11.1}$$

The function is described the interpolation laws and sampled data points:

$$\text{interp} \;:\; \begin{pmatrix} i_1 \\ \vdots \\ i_K \end{pmatrix} \in \{\text{ integer index }\} \;\rightarrow\; \begin{pmatrix} l_1 \\ \vdots \\ l_K \end{pmatrix} \in \{\text{ interpolation law }\}, \tag{11.2}$$

$$\text{data} \;:\; \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} \in \{\text{ number }\} \;\rightarrow\; \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \in \{\text{ any data type }\}. \tag{11.3}$$

There are a general version, which takes an object as the function $y$ value type, and a specific version, which takes double floating numbers as the function $y$ value type. The data structure are listed here:

```
template<typename T, typename H = ENDFEmpty>
struct CFSObjectFunction :
ENDFObjectInterpolationFunction<T, H> {

    // Additional interfaces

};

struct CFSFunction :
ENDFInterpolationFunction {

    // Additional interfaces

};
```

### 11.1.2 Linearly Interpolation Function

### 11.1.3 Probability Function

## 11.2 Reaction Categorization

## 11.3 Common Energy Grid

All cross sections are based on a common energy grid with cross sections being linearly interpolated between points on the grid. The energy grid is chosen so that the interpolation delivers a maximum relative error of 0.1% from the true value. The CFS module will combine related parts in the neutron data structure and derive the command energy grid. We define the energy grid as:

$$E_i, \;\; i = 1 \ldots N. \tag{11.4}$$

The C++ data structure is described here:

```
struct CFSEnergyGrid {
```

```
    // The energy data points in eV
    std::vector<double> energiesEv;
};
```

## 11.4 Cross Sections

Each reaction has an $MT$ number whose meaning is introduced in Table 4.2. The cross section on the energy grid is given by:

$$I^{MT} = \text{starting index of reaction MT}, \tag{11.5}$$

$$J^{MT} = \text{ending index of reaction MT}, \tag{11.6}$$

$$\sigma_i^{MT}, \qquad i = I^{MT} \ldots J^{MT}. \tag{11.7}$$

The cross section $\sigma_i^{MT}$ may be only defined for a portion of the energy grid, i.e. from index $I^{MT}$ to index $J^{MT}$, and is extended as a function of $1/v$ (or $1/\sqrt{E}$) below the region of interest, and is extended as a constant above the region of interest. The C++ data structure is described here:

```
struct CFSCrossSection {

    // The starting index of the cross section
    // on the energy grid
    long startIndex = 0;

    // The cross section data points in barn
    std::vector<double> sigmasBarn;
};
```

## 11.5 Energy Angular Distributions

In this section we discuss the energy and angle distributions of the particles emitting from a reaction.

### 11.5.1 General Distributions

When neutron or other particles incident on the target particle, particles may emitted. For an outgoing particle $i$, the differential cross section for an incoming particle with energy $E$ and outgoing particle with energy $E'$ and an angular cosine $\mu$ between them is given by:

$$\sigma_i(\mu, E, E') = \frac{1}{2\pi}\,\sigma(E)\,y_i(E)\,f_i(\mu, E, E'), \tag{11.8}$$

where the terms are defined as:

$$\sigma(E) = \text{reaction cross section for incoming particle at energy } E, \tag{11.9}$$
$$y_i(E) = \text{number of particles } i \text{ emitted incoming particle at energy } E, \tag{11.10}$$
$$f_i(\mu, E, E') = \text{a probability density function to describe outgoing particle,} \tag{11.11}$$

and the distribution $f_i(\mu, E, E')$ has the normalization:

$$\int_0^{E'_{max}} dE' \int_{-1}^{1} d\mu\, f_i(\mu, E, E') = 1, \tag{11.12}$$

where $E'_{max}$ is the maximum possible energy of outgoing particle. The coupled distribution is given in ENDF file 6.

### 11.5.2 Decoupled Distributions

There are cases that the energy and angular distributions can be decoupled into the angular part and the energy part. The key is that the energy $E'$ and the angle $\mu$ are independent for each other. In this case, we express the distribution $f_i$ as the product of two functions:

$$f_i(\mu, E, E') = p_i(E, E')\,g_i(\mu, E), \tag{11.13}$$

where the $p_i$ function is the energy distribution part and the $g_i$ function is the angular part. The normalization conditions are:

$$\int_0^{E'_{max}} p_i(E, E')\,dE' = 1, \tag{11.14}$$
$$\int_{-1}^{1} g_i(\mu, E)\,d\mu = 1. \tag{11.15}$$

The differential cross section is given by:

$$\sigma_i(\mu, E, E') \;=\; \left[\frac{1}{2\pi}\, g_i(\mu, E)\right]\, \sigma(E)\, y_i(E)\, p_i(E, E'). \tag{11.16}$$

The decoupled angular distribution is given in ENDF file 4, and the decoupled energy distribution is given in ENDF file 5. Sometimes, the energy law $p_i(E, E')$ is too complicated to be represented in s a single law, we express as the sum of components:

$$p_i(E, E') \;=\; \sum_{k=1}^{K} \alpha_k(E) q_{i,k}(E, E'), \tag{11.17}$$

where the functions $q_{i,k}(E, E')$ are $k$th partial law with a validity function $\alpha_k(E)$:

$$\sum_{k=1}^{K} \alpha_k(E) \;=\; 1. \tag{11.18}$$

Moreover, each partial law is normalized as well:

$$\int_0^{E'_{max}} q_{i,k}(E, E')\, dE' \;=\; 1, \tag{11.19}$$

which keeps the total distribution $p_i$ normalized as equation (11.14).

### 11.5.3 Yield of Reactions

In the differential cross section 11.8, one important factor is the yield $y_i(E)$, or the number of partials emitted from the reaction. In this subsection, we discuss the possible yields from reactions. In ENDF description system, a yield can be given implicit from the type of reaction or from the yield function from File 6. There are two possibilities of the yield function: a constant or a function of energy. Since the neutron is the most important particle in reactor physics calculation, we summarize the neutron yields in Table 11.1.

| Reaction | Possible Constant $y$ | Be Function $y(E)$ | Reaction | Possible Constant $y$ | Be Function $y(E)$ |
|---|---|---|---|---|---|
| MT = 2 | 1 | NO | MT = 5 | $N \geq 0$ | YES |
| MT = 11 | 2 | NO | MT = 16 | 2 | NO |
| MT = 17 | 3 | NO | MT = 18 | N/A | YES |
| MT = 19 | N/A | YES | MT = 20 | N/A | YES |
| MT = 21 | N/A | YES | MT = 22 | 1 | NO |

| | | | | | |
|---|---|---|---|---|---|
| MT = 23 | 1 | NO | MT = 24 | 2 | NO |
| MT = 25 | 3 | NO | MT = 28 | 1 | NO |
| MT = 29 | 1 | NO | MT = 30 | 2 | NO |
| MT = 32 | 1 | NO | MT = 33 | 1 | NO |
| MT = 34 | 1 | NO | MT = 35 | 2 | NO |
| MT = 36 | 1 | NO | MT = 37 | 4 | NO |
| MT = 41 | 2 | NO | MT = 42 | 3 | NO |
| MT = 44 | 1 | NO | MT = 45 | 1 | NO |
| MT = 50+$i$, $1 \leq i \leq 40$ | 1 | NO | MT = 91 | 1 | NO |
| MT $> 100$ | 0 | NO | | | |

Table 11.1: Neutron yields for reactions

# CHAPTER 12

# ACE: ACE Evaluated File System

# CHAPTER 13

# ADLS: ACE Data Library System