CBZ/BurnupChain と NuclideChainData の使用マニュアル

千葉豪

平成 29 年 4 月 27 日

燃焼計算に用いる燃焼チェーンは「BurnupChain」クラスで定義される。この BurnupChain クラスは複数の原子核についての崩壊、生成情報を保持する必要があり、これら個々の原子核の崩壊、生成情報は「NuclideChainData」クラスのインスタンスとして保持されている。具体的には、BurnupChain クラスのインスタンスは複数の NuclideChainData クラスを「map」として保持している。

1 BurnupChain クラス

BurnupChain クラスについては、簡単な解説が Burner モジュールのマニュアルの 3.2 節に記載されている（2016/10/27 時点）ので、ここでは省略する。

なお、個々の原子核の崩壊定数については、理由は今となっては不明だが BurnupChain クラスで定義している。取り出すときは GetDecayConstant メソッドを用いる（引数は取り出したい原子核の核種 ID）。

2 NuclideChainData クラス

BurnupChain クラスのインスタンスからの NuclideChainData クラスのインスタンスの取り出しは「GetNuclideChainData」メソッドにより行う。このメソッドの引数としては、取り出したい原子核の核種 ID（U-235 ならば「922350」）、もしくは核種名（U-235 ならば「“U235”」）を指定する。以下に一例を示す。なお、一般的には BurnupChain クラスのインスタンスは Burnup クラスのインスタンス（この例では「bu」）のメンバとして保持されているので、この例はそのような場合におけるものとしている。

Listing 1: BurnupChain クラスのインスタンスからの NuclideChainData クラスのインスタンスの取り出しの例

```
1  NuclideChainData ncd;
2  ncd=bu.GetBurnupChain().GetNuclideChainData(922350);
```

NuclideChainData クラスについては、メソッドを解説するよりも、それ自身のメンバー変数の構成について説明したほうが早いと考えられる。以下に NuclideChainData クラスのヘッダファイル（ファイル「BurnupChain.h」に記載）のメンバー変数を定義している箇所を示す。

Listing 2: NuclideChainData クラスのヘッダファイルにおけるメンバー変数の定義

```
1  class NuclideChainData{
2   private:
3    int id; // nuclide ID
4    vector<int> ndiv; // number of yielded nuclides
5    // fission, capture, n2n, decay
6    vector< vector<real> > ratio; // ratio of each yielded nuclide
```

```
7   vector< vector<int> > idnext; // ID of yielded nuclide
8   vector<real> decay_energy; // (eV) [elp,eem,ehp]
```

変数「ID」はそれ自身の原子核 ID を示す[1]。

ベクター変数「ndiv」は、崩壊もしくは中性子核反応により他の原子核に遷移するパスの総数を定義する。核分裂反応によるパス数は ndiv[0]、中性子捕獲反応によるパス数は ndiv[1]、(n,2n) 反応によるパス数は ndiv[2]、崩壊によるパス数は ndiv[3] で定義される。例えば、崩壊により単一の核種に変換される場合には ndiv[3] は 1 となるし、Am-241 のように中性子捕獲反応で 2 つの核種（Am-242 と Am-242m）が生成される可能性がある（分岐がある）場合には ndiv[1] は 2 となる。また、核分裂性核種については、ndiv[0] は核分裂反応の結果生成する FP の総数となる。

変数「ratio」は二次元のベクター配列で、ratio[i][j] により、j 番目のパスへの移行確率（分岐比）を定義する。i=0 では核分裂、i=1 では中性子捕獲、i=2 では (n,2n)、i=3 では崩壊に対応する（ndiv の考え方と同様であり、これ以降はその説明を省略する）。

変数「idnext」は ratio と同様に二次元のベクター配列で、idnext[i][j] により、反応 i の j 番目のパスの結果、生成される原子核の核種 ID を定義する。Am-241 の捕獲反応を例にすると、idnext[1][0]=942420（Am-242）、idnext[1][1]=942421（Am-242m）となる。

変数「decay_energy」は崩壊あたりの発生エネルギーを定義し、decay_energy[i] は、i=0 がベータ線、i=1 がガンマ線、i=2 がアルファ線の寄与にそれぞれ対応する。

一例として、核分裂収率の数値を取り出すメソッドについて説明する。核分裂収率は個々の核分裂性核種について定義されるため、まずは該当する核分裂性核種の NuclideChainData クラスのインスタンスにアクセスする必要がある。核分裂収率データは、上記で説明した「ndiv」「ratio」「idnext」で定義されているが、NuclideChainData クラスの「GetFissionYield」メソッドによって取り出すことができる。このメソッドの引数には収率を取り出したい FP 核種の ID を渡す。また、BurnupChain クラス自体にも「GetFissionYield」メソッドが実装されており、この場合は、一つ目の引数に収率を取り出したい核分裂性核種の ID を、二つ目の引数に FP 核種の ID を渡す。以下に、Pu-241 による Pd-109 の核分裂収率のデータを取り出すメソッドの使用例を示す。

Listing 3: Pu-241 による Pd-109 の核分裂収率データの取り出し例

```
1   real val1=bu.GetBurnupChain().GetNuclideChainData(942410).GetFissionYield(461090);
2   real val2=bu.GetBurnupChain().GetFissionYield(942410,461090);
3   cout<<val1<<" "<<val2<<"\n"; exit(0);
```

[1]なお、このインスタンスを保持することになる BurnupChain クラス側でもその原子核 ID を保持しており、BurnupChain 側の ID と NuclideChainData 側の ID とが一致している場合のみ、その核種が「保持」されていると判断される。このあたりは複雑なので理解する必要はない。