CBZ/Burnerの使用マニュアル

千葉豪

2021年10月15日

目 次

1 はじめに

ウラン、プルトニウムといった重核種のいくつかの同位体は、中性子と衝突することによって高い確率で核分裂反応を起こし、一反応あたりおおよそ 200MeV のエネルギーを発生します。核分裂の結果、重核種は通常、ふたつの核分裂生成物（fission product、FP）とよばれる原子核に分かれます。核分裂エネルギーのうち大部分（8 割程度）は FP の運動エネルギーとなり、原子力発電プラントではそれが最終的に熱エネルギー、電気エネルギーへと変換されていきます。

核分裂反応により重核種が FP に変わっていくわけですから、原子力発電プラントを運転していくに従って、燃料内の重核種の数はどんどん減少していくことになります。核分裂性の重核種は中性子の連鎖反応に寄与しますから、それらが減少していくと、核分裂の連鎖反応を継続することが難しくなります。従って、原子力発電所では、新しい燃料を原子炉に装荷して運転を開始する際には、核分裂の連鎖反応が過剰に起こりやすいように「多めの」燃料を装荷します。このように、核分裂の連鎖反応が過剰に起こりやすい状態を、専門用語では「正の反応度が入っている」などと言います。このように過剰な核燃料を装荷した場合は、そのままでは核分裂の連鎖反応がバランスよく継続される「臨界」状態を越えてしまうことになりますので、中性子吸収材である制御棒を挿入したり、軽水炉であれば冷却水中に中性子を吸収するボロンを希釈したり、燃料に中性子吸収体であるガドリニウムを混合したりして、過剰な正の反応度を抑制し、臨界状態を保ちます。原子力発電プラントの運転に伴って、核分裂する核燃料は減少し、反応度も減少していくため、制御棒を少しずつ引き抜いたり、ボロンの希釈量を小さくしたりします[1]。

また、核燃料に多く含まれるウラン 238（U-238）は、中性子を吸収することによって U-239 となり、これが 2 度のベータ崩壊を経て核分裂性核種のプルトニウム 239（Pu-239）に変わります。つまり、原子力発電プラントを運転していくに従って、U-235 や Pu-239 といった核分裂性核種は減少しますが、U-238 の中性子吸収により Pu-239 が新たに生成され、核分裂連鎖反応に寄与することになります。

さらに、核分裂により発生した FP の量も重要となります。FP の種類は 1,000 を越えますが、このなかでは強い放射線を発するもの、半減期が長く、いつまでも残留してしまうものなどがあります。使用済み核燃料の再処理、中間貯蔵、深地層処分を考えるうえで、使用済み核燃料に含まれる FP が、どのようなエネルギーの放射線を、どの程度の期間出しつづけるかを正確に把握しておく必要があります。さらには、万が一、原子炉で事故が起こった場合、外部環境に放出されてしまう FP の量を評価することも重要です。FP は、ある確率分布をもって核分裂により発生します[2] が、FP の大部分は中性子過剰の不安定核なので、固有の半減期に従って崩壊し、異なる核種に変換されます。また、原子炉内で中性子を吸収することによっても異なる核種となります。

以上のように、原子力発電プラントで原子炉に装荷された核燃料は、運転の経過とともに、その組成が大きく変わっていきます。このことを**核燃料の燃焼**と呼び、燃焼に伴う核燃料の組成変化を正確に評価することは極めて重要です。

コードシステム CBZ には、核燃料の燃焼を計算するためのパッケージがいくつかありますが、本稿では燃料ピンセルの燃焼計算を行うパッケージ Burner について説明します。

[1] 燃料にガドリニウムを混ぜ込んだ場合は、ガドリニウムは強中性子吸収体であるため、原子力発電プラントが運転を始めると、比較的短い期間でガドリニウムは中性子を吸収し、中性子の吸収断面積が小さい別な同位体に変わります。その結果、ガドリニウムによる「負の反応度」は小さくなっていきます。このような働きをするものを**可燃性毒物**（Burnable poison）と呼びます。

[2] 例えば、U-235 が熱中性子によって核分裂した場合、ある FP は 0.3%の確率で発生し、違うものは 0.2%発生し、、、という具合。

2 必要なデータファイル

Burner では、中性子核反応断面積を収納する多群断面積に加えて、FP の崩壊データ、核分裂収率データといった、燃焼計算に特有のデータが必要となります。多群断面積データは CBGLIB ディレクトリに、燃焼計算に必要なデータは CBGLIB_BURN ディレクトリに収納されることになります。

Burner を使用するための一般的なディレクトリ構造は以下のようなものになります。

/home/
- CBG/src/ （CBZ のソースプログラム収納）
- CBGCAL/Burner/ （CBZ の計算用ファイル収納）
- CBGLIB/
  - j4.107g.iwt4/ (JENDL-4.0 ベースの 107 群ライブラリ）
  - Bell-107g/ (107 群ライブラリ用の最適化された Bell 因子データ）
- CBGLIB_BURN/
  - CBG_CHAIN/ （燃焼チェーンデータ収納）
  - CBG_FY/ （燃焼チェーン毎の累積核分裂収率データ収納）
  - DDFile/ （崩壊に関するデータ収納）
  - ReactionEnergy/ （核分裂反応あたりに発生するエネルギーに関するデータ収納）
  - atomic_mass/ （原子重量に関するデータ収納）
  - decay_constant/ （崩壊定数に関するデータ収納）

このチュートリアルでは Burner ディレクトリの main.burn_tutorial.cxx に基づいた説明が行われます。

この main.burn_tutorial.cxx をコンパイルし実行させることで燃焼計算が行われますが、Mac 環境では、Segmentation fault で計算が正常に行えない場合があることが報告されています。そのような場合には、まずは千葉に相談して下さい。

3 入力データの作成

原子炉には複数の燃料集合体が配置され、その燃料集合体は複数の燃料ピンから構成されます。燃料集合体は縦に長い構造となっており、それが原子炉の鉛直方向に挿入され配置されますが、燃料集合体を上から見ると、現在の商用原子炉である軽水炉では正方形状、「常陽」「もんじゅ」といった高速炉では六角形状となっています。また、集合体内における燃料ピンの配置も、軽水炉では正方配列、高速炉では六角配列となっています。この違いの理由ですが、熱伝達特性が良好なナトリウムを冷却材として用いる高速炉では、極力、冷却材の割合を小さくし効率的に核燃料を「詰める」ためです（六角配列にすると燃料をより稠密に配置することができます）[3]。

原子炉内に装荷された核燃料はその装荷位置によって中性子束レベルとエネルギースペクトルが異なるため、原子炉内の全ての核燃料の燃焼挙動を評価するためには、原子炉全体の計算を行う必要があります。ところが、燃料集合体は数十本を越える燃料ピンから構成され、さらに原子炉は百体を越える燃料集合体から構成されます。さらに、同一の燃料ピンであっても、そのピン中の中性子束は軸方向位置によっても異なるため、これら全てを評価することは極めて困難な作業となります。

そこで、主に軽水炉では、燃料ピンと被覆管、さらにはそれを囲む冷却材からなる**燃料ピンセルモデル**で、簡易的に燃料の燃焼挙動を評価する、ということが行われます。冷却材の外側には適切な境界条件を課すため、径方向については同様のセルが無限に配列されていると仮定することになります。また、軸方向については無限として扱うため、ピンセルモデルは二次元となります（仮に冷却材領域の外側も円周で近似するならば、さらに簡易的な一次元円筒でモデル化できます）。BWR の燃料集合体と、それに対応するピンセルモデルの例について **Fig. 1** に示します。

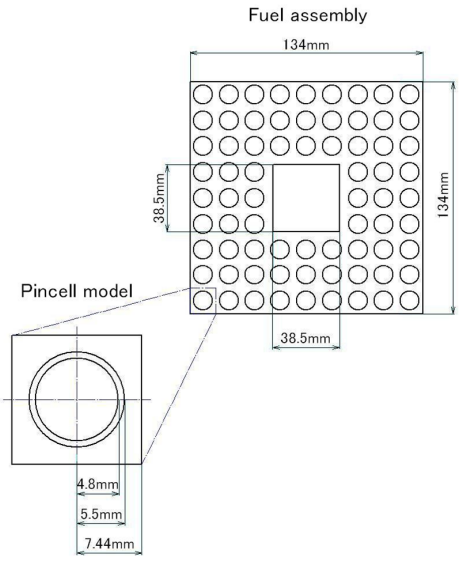

Fig. 1: BWR 燃料集合体とピンセルモデル（川本君作成の図を拝借）

さて、それでは、Burner を使用するためのプログラム `main.burn_tutorial.cxx` を眺めながら、Burner の使い方について説明していきます。

[3]軽水炉でも、中性子の減速を緩和させて、U-238 から Pu-239 への転換を促進させる型式が提案されています（高転換炉、低減速スペクトル炉などと呼びます）。このような場合には、燃料ピン配置は六角配列となります。

3.1 多群断面積データ、燃焼チェーンデータの指定

燃焼計算では、中性子と原子核との種々の反応のうち、原子核が自身とは異なる原子核に変換するような反応がどれだけ起こっているかを知る必要があります。そのような核反応として、核分裂反応（2 つの FP に変換)、中性子捕獲反応（中性子数がひとつ増加した核種に変換)、(n,2n) 反応（中性子数がひとつ減少した核種に変換）が挙げられます。その反応の量（反応率）を知るためには、まずはそれらの断面積データが必要となります。また、中性子束の大きさと、そのエネルギースペクトルの情報も必要となります [4] ので、ピンセル体系で中性子束の計算を行います（Burner では衝突確率法を使用)。この中性子束計算の際にも断面積データが必要となります。

また、中性子との特定の核反応により、原子核は異なる原子核に変換するわけですが、個々の原子核について、ある核反応が起こった際にどの原子核に変換するかの情報が必要となります。加えて、崩壊でも原子核は異なるものに変換されるため、崩壊後にどの原子核に変換するかの情報も必要です。このような情報をまとめたものを**燃焼チェーン**と呼びます（燃焼チェーンの詳細については `BurnupChainGenerator` のマニュアルに述べられています)。なお、燃焼チェーンの図は、計算コードシステム SRAC-2006（JAEA-Data/Code 2007-004）のマニュアルの p.186 以降に記載されていますので、ホームページ [5] から印刷して手元に置いておいて下さい。今回の計算で用いているものは、重核種については p.186 の Fig.3.3-1 のもの、FP 核種については p.189 と p.190 のものに該当しています。SRAC-2006 で用いられている主要な重核種の燃焼チェーンについて **Fig. 2** に示します。

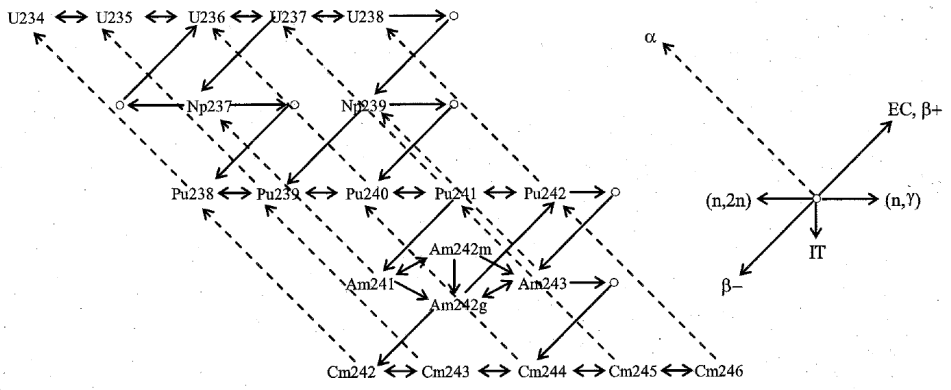

Fig. 2: SRAC-2006 の主要重核種の燃焼チェーン

CBZ では、これら断面積データ、燃焼チェーンデータは外部ファイルに定義されているので、それを読み込む必要があります。それでは以下に、断面積データ、燃焼チェーンデータを読み込む箇所を示します。

Listing 1: 断面積データ、燃焼チェーンデータの指定

```
1   Burner bn;
2   Burnup bu;
3
4   string cbglibdir("../../");
5
6   // (common)
7   bu.ReadReactionEnergyFromFile(cbglibdir,"sractype");
8   //bu.ReadReactionEnergyFromFile(cbglibdir,"fr_standard");
9   bu.GetBurnupChain().ReadDecayConstantFromFile(cbglibdir,"srac_org");
10  bu.ReadAtomicMassDataFromFile(cbglibdir,"jeff311");
11
12  // (FP197)
13  bn.SetLibrary(cbglibdir,"jendl-4.0");
14  bu.GetBurnupChain().Set21HeavyMetalChain();
15  bu.GetBurnupChain().ReadFPYieldDataFromFile(cbglibdir,"fp197.j2011t");
16  bu.GetBurnupChain().OverWritingChainData(cbglibdir,"fp197.j2011t");
17
18  // (Actinide decay heat)
```

[4] エネルギースペクトルの情報が必要なのは、核分裂などの種々の反応断面積が中性子の入射エネルギーに依存するから。
[5] http://jolissrch-inter.tokai-sc.jaea.go.jp/pdfdata/JAEA-Data-Code-2007-004.pdf

```
19  bn.AddActinideDecayHeatDataToBurnupChain(cbglibdir,"dd.jeff311",bu);
20
21  // (Dose coefficient data)
22  bu.GetBurnupChain().ReadDoseCoefficientData(cbglibdir);
```

ピンセルの燃焼計算を行う Burner は、C++の「クラス」です。そこで、実際の計算を行うため、Burner クラスの「インスタンス」を生成する必要があります。1行目において、Burner クラスのインスタンス bn を作成しています。以降の実際の計算はインスタンス bn が行うことになります。

Burner クラスはピンセル体系の燃焼計算を行うためのクラスですが、実際に燃焼数値計算を行うのは Burnup クラスです。Burnup クラスは、燃焼チェーンデータを定義する BurnupChain クラスのインスタンスをメンバとして保持するとともに、核種の一群断面積データなどを保持し、燃焼計算を行います。そこで、2行目で、Burnup クラスのインスタンス bu を作成しています。

Burner では、計算に必要な断面積データは CBGLIB、燃焼に関するデータは CBGLIB_BURN という名前のディレクトリに置かれるルールとなっています。4行目では、これらのディレクトリの位置を指定します。この例では、今いるディレクトリからふたつ上位のところに CBGLIB、CBGLIB_BURN ディレクトリが存在することになっています。

多群断面積データの読み込みは 13 行目の bn.SetLibrary で行っています。この例で読み込んでいる断面積データは、評価済み核データファイル JENDL-4.0 に基づいたエネルギー 107 群の多群断面積ライブラリとなります。なお、Burner を用いる場合、多群断面積ライブラリに対応する XSLibrary クラスは Burner 内部で定義されるため、直接ユーザが扱うことはありません。断面積データを表示するなど、XSLibrary クラスのメソッドを用いたい場合は、Burner クラスの GetXSLibrary メソッドを用いることで XSLibrary クラスのインスタンスを取り出すことが出来ます。

燃焼チェーンデータは Burnup クラスが保有するメンバー（BurnupChain クラスのインスタンス）として定義されます。14 行目では重核種のためのチェーンとしてウランからキュリウムまでを含む 21 核種のものを使うことを宣言しています。15、16 行目では、核分裂による FP の生成確率分布（FP yield）のデータ、FP のチェーンデータを外部ファイルから読み込んでいます [6]。燃焼チェーンに含まれるデータは、反応断面積と同様に評価済み核データファイルによって異なり、この例では JENDL FP decay data file 2011 に基づくデータファイルを読み込んでいます。

7 行目では核反応あたりに発生するエネルギーデータを、9 行目では崩壊定数データを、10 行目では原子質量データをそれぞれ読み込んでいます。

また、以上のデータではアクチニド核種（ウラン、プルトニウムといった重核種）の崩壊熱データが定義されないため、19 行目でそのデータを外部ファイルから読み込んでいます。最後に 20 行目では線量換算係数データを外部ファイルから読み込んでいます（これは吸入もしくは経口線量を計算する際に利用します）。

3.2 燃焼チェーンデータの出力

前節で定義したデータは Burnup クラスのインスタンス bu などに格納されていますが、それらについてユーザが知るためのメソッドがいくつか用意されています。本節では燃焼チェーンデータを定義するクラス（BurnupChain クラス）が有するメソッドを説明します。なお、燃焼チェーンデータの詳しい解説については、「CBZ/BurnupChain と NuclideChainData の使用マニュアル」にも記載されているので、必要であればそちらも参考にして下さい。

以下に燃焼チェーンデータを出力するための例を示します。

Listing 2: 燃焼チェーンデータの出力

```
1  //bu.GetBurnupChain().ShowHalfLife();
2  //bu.GetBurnupChain().ShowDoseCoefficient();
3  //bu.GetBurnupChain().ShowChainData();
4  //bu.GetBurnupChain().ShowChainDataToOrigin("Ba137");
5  //bu.GetBurnupChain().ShowNGBranchingRatio();
```

[6] bu.GetBurnupChain() では、Burnup クラスのインスタンス bu が保有する BurnupChain クラスのインスタンスを呼び出しています。従って、それに続くメソッド ReadFPYieldDataFromFile は、BurnupChain クラスのメソッドになります。

```
6   //bu.GetBurnupChain().ShowFPYieldForFP("Cs137");
7   //bu.GetBurnupChain().ShowFPYield("U235");
```

ここで示されている各メソッドについて、下記で説明します。

- ShowHalfLife：燃焼チェーンに含まれる原子核の崩壊定数と半減期データを出力させるためのものです。コメント（「//」は、その行のそれ以降をコメントアウトする役割があります）を外して、コンパイル、実行してみて下さい[7]。
- ShowDoseCoefficient：燃焼チェーンに含まれる原子核の経口線量変換係数と吸入線量変換係数を出力させるためのものです。
- ShowChainData：燃焼チェーンデータを出力します。引数として何も与えなければ、燃焼チェーンに含まれる全ての核種について、中性子反応、崩壊後にどの核種に変換するかが出力されることと思います。特定の核種のデータのみを出力したい場合には、核種名の文字列か、核種 ID を引数として与えて下さい。
- ShowChainDataToOrigin：引数で指定した核種（この例では Ba-137）が、どの核種の、どの反応（崩壊）で生成するかを出力してくれます。下に出力例を示します。

Listing 3: ShowChainDataToOrigin メソッドによる燃焼チェーン情報の出力例

```
1 # Origin of Ba137 (561370)
2 #   Cs137 (551370) : Decay      0.053
3 #   Ba136 (561360) : Capture    0.972094
4 #   Ba137m (561371) : Decay      1
5 #   Ba138 (561380) : (n,2n)     0.875913
```

各行の最後に数値が出力されていますが、これは「分岐比」を指しています。この例でいうと、Ba-136 が中性子を吸収した場合、97.2%の確率で Ba-137 になることを示しています（ちなみに、残りは Ba-137m（準安定状態 meta stable）になります）。

- ShowNGBranchingRatio：(n,γ) 反応により生成される核種が ground と meta-stable に分岐する核種について、分岐比を一覧にして示します。
- ShowFPYieldForFP：引数で名前を与えた FP の核分裂収率を核分裂性核種毎に示します。
- ShowFPYield：引数で名前を与えた核種が核分裂の結果発生する FP の収率を一覧で示します。

なお、個々の原子核の燃焼に関するデータ（数値）を取り出したいときには、引数にデータを取り出したい原子核の ID を与える以下のようなメソッドを使うとよいでしょう。

Listing 4: 個々の燃焼チェーンデータの取り出し例

```
1   int id=942390;
2   real dc=bu.GetBurnupChain().GetDecayConstant(id);
3   real coef1=bu.GetBurnupChain().GetDoseCoefficientIngestion(id);
4   real coef2=bu.GetBurnupChain().GetDoseCoefficientInhalation(id);
5   cout<<id<<" : "<<dc<<" "<<coef1<<" "<<coef2<<"\n"; exit(0);
```

3.3 組成データの入力

前述の通り、燃料ピンセルモデルは一般的に、燃料、被覆管、冷却材の三つの媒質で構成されます。従って、それら三媒質それぞれについて、含まれる原子核の種類とその数密度を指定する必要があります。Burner ではそれぞれの媒質について、含まれる原子核の Index と数密度を入力します。main.burn_tutorial.cxx の該当部分を以下に示します。

[7]ただしこの場合、ShowHalfLife メソッドにより半減期データを出力した後に、それ以降の処理が継続して行われ計算結果がどんどん出力されてしまうため、この行の次に exit(); を打ち込んでおき、そこで処理を終了させるようにしましょう。

Listing 5: 組成データの指定

```
1  // +++ fuel composition +++
2  // (UO2)
3  int nuc0=3;
4  int mat0[]={922350,922380,80160};
5  real den0[]={7.753e-4, 2.175e-2, 4.505e-2}; // 3.4% U5-enrichment
6  real temp0=968.8;
7   // (clading)
8  int nuc1=3;
9  int mat1[]={400000,260000,240000};
10 real den1[]={3.786e-2, 2.382e-4, 6.770e-5};
11 real temp1=604.;
12 // (moderator)
13 int nuc2=6;
14 int mat2[]={10010,80160,50100,280000,240000,260000};
15 real den2[]={5.572e-2, 2.786e-2, 4.592e-6, 3.688e-4, 1.609e-4, 1.306e-4};
16 real temp2=574.2;
17
18 bn.PutFuelData(nuc0,mat0,den0,temp0);
19 bn.PutCladData(nuc1,mat1,den1,temp1);
20 bn.PutModeratorData(nuc2,mat2,den2,temp2);
```

2 行目から 6 行目では燃料媒質の組成データを与えています。3 行目の `nuc0` にはその媒質に含まれる原子核の種類数を与えます。4 行目の整数型（`int`）配列 `mat0` には含まれる原子核の Index を与えます。5 行目では個々の原子核の数密度を実数の配列 `den0` で与えます。この順番は `mat0` 配列の並びに対応します。また、数密度の単位は $10^{24}/\mathrm{cm}^3$ です。最後の 6 行目では燃料領域の温度を指定します（単位は K）。このように指定した燃料領域の数密度データは、18 行目の `PutFuelData` メソッドにより、`Burner` クラスのインスタンス `bn` に入力されます。被覆管領域、冷却材領域についても同様にして数密度データを与えます。

3.4 形状データの入力

Burner では燃料ピンセルモデルを扱います。燃料ピン自体は、円筒形のペレットと被覆管で構成されますが、その外側の冷却材領域の外周は、燃料ピンが正方配列であるとすると正方形、六角配列であるとすると正六角形となります。ちなみに、ピンセルの外部境界条件には**周期条件**（periodic boundary）を用いています[8]。

燃料ピンセルモデルは三つの媒質で構成されますが、中性子束分布の計算では燃料領域と冷却材領域を複数の空間メッシュに分割します[9]。**図 1** に正方ピンセルのメッシュ分割例を示します。この例では、中心から 3 つまでの円が燃料領域、その次が被覆管領域、それ以外が冷却材領域となります。

`main.burn_tutorial.cxx` で形状データを指定している部分を以下に示します。

Listing 6: 形状データの指定

```
1  // +++ Geometry Data +++
2  real pin_pitch=0.6325*2.; // [cm]
3  real rr[]={0.25,0.35,0.412,0.476,0.55,0.65,0.75};
4  int rmed[]={0,0,0,1,2,2,2};
5  bn.PutGeometryData(pin_pitch,7,rr,rmed);
```

2 行目の `pin_pitch` ですが、これは燃料ピンのピッチ（ある燃料ピンと隣の燃料ピンの中心間距離）を示します。燃料ピンセルモデルの場合は、冷却材領域外側の正方形の辺の長さとなります。単位は cm です。3 行目では上述したそれぞれの円環領域の半径を、小さい順から指定します。4 行目ではそれぞれの円環領域がどの媒質に対応するかを、内側の領域から指定します。`0` が燃料、`1` が被覆管、`2` が冷却材に、それぞれ対応します。5 行目では、`Burner` クラスの `PutGeometryData` メソッドにより、上記で指定したデータを `Burner` クラスのインスタンス `bn` に与えています。なお、2 つ目の引数は、円筒の配置数に対応しています。

[8] 周期的にピンセルが配置される場合にはこの境界条件が厳密となります。この他に、ピンセルの外部境界に達した中性子は等方に反射され体系に戻るという等方反射 White reflection を設定することができます。等方反射を用いた場合には計算時間を短縮することが出来ますが、あくまで近似であることに注意が必要です。等方反射条件への変更は `Burner` クラスのメソッド `PutWhiteBoundary()` で行うことが出来ます。

[9] 実際の燃料ピンでは、燃料（ペレット）領域と被覆管領域の間にはギャップが存在しますが、ここではそれを無視しています。従って、ギャップを無視している分、燃料がギャップ領域に「希釈」されている、という考え方になります。

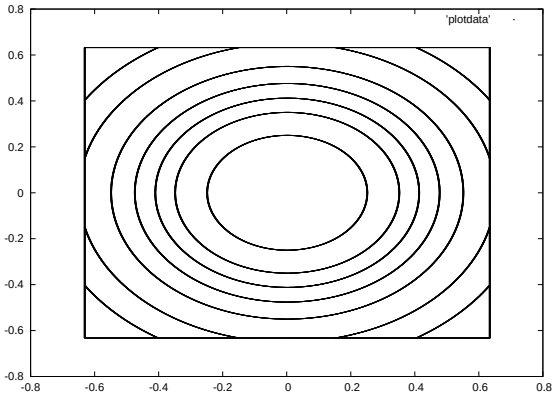

Fig. 3: ピンセルのメッシュ分割

3.5 燃焼条件の入力

核燃料の燃焼計算では、中性子束レベルの大きさと燃焼期間の情報が必要になります。中性子束レベルのかわりに、燃料ピン 1cm 長さあたりの出力（線出力）が与えられる場合もあります。`main.burn_tutorial.cxx` の燃焼条件の入力に該当する部分を以下に示します。

Listing 7: 燃焼条件データの入力

```
1   // +++ Burnup history data +++
2   bn.PutBurnStep(21);
3   real power_density_list[]={
4     179., 179., 179., 179., 179.,
5     179., 179., 179., 179., 179.,
6     179., 179., 179., 179., 179.,
7     179., 179., 179., 179., 179.,
8     0.
9   }; // [W/cm]
10  real burn_time[]={
11     0.1, 1.0, 2.5, 5.0, 7.5,  10., 12.5, 15., 17.5, 20.,
12     22.5, 25., 27.5, 30., 32.5,  35., 37.5, 40., 42.5, 45., // GWd/t
13     4.*365., // day
14  };
15  bn.PutPowerDensityList(power_density_list);
16  bn.PutBurnTime(burn_time,true,true); // [GWd_t/day][accumulate/not]
```

燃焼計算は「ステップ」に区切って行います。各ステップのはじめに中性子束分布の計算を行い、それぞれの原子核について反応率を計算します。中性子束分布は燃料の組成に依存して変化する（すなわち燃焼時間に依存して変化する）ため、燃焼時間を適切なステップに分割する必要があります。このステップ数を指定しているのが 2 行目です。3 行目から 9 行目までは、それぞれの燃焼ステップにおける線出力を W/cm 単位で指定しています。出力を「比出力（MW/tU）」で指定することも可能で、この場合には `PutPowerDensityList` メソッドの二つ目の引数に `string` 型で `''MW_t''` を追加します (`PutPowerDensityList(power_density_list,''MW_t'');` とします)。この例では 21 ステップ目の線出力がゼロになっていますが、このステップは燃料が原子炉から取り出され、冷却されていることを想定しています [10]。10 行目から 14 行目までは各ステップの燃焼期間を指定しており、単位は「GWd/t」となります [11]。なお、ここでは積算値を与えていることに注意して下さい (例えば 2 ステップ目では 1.0GWd/t となっていますが、これは 2 ステップ終了時にその程度燃料が燃えたことを意味しています。１ステップ目で 0.1GWd/t まで燃やしているため、２ステップ目では 0.9GWd/t 燃やしていることになります)。ただし、線出力がゼロのステップでは、単位は「日」になります（`Burner` クラスが自動的に判別してくれます。この場合は積算値ではないことに注意)。ちなみに燃焼初期で細かくステップを分割しているのは、短半減期の FP の挙動を正確に評価するためです。

[10] 冷却期間では、中性子と原子核との反応は起こりませんが、崩壊は起こるので、燃料の組成は時間とともに変動します。
[11] はじめに装荷した燃料に含まれる重核種の重量あたりに発生した熱出力積算値。

15、16 行目では、上記のように設定した燃焼条件を、Burner クラスのインスタンス bn に渡しています。

上の例では、各燃焼ステップの時間長さを GWd/t 単位の燃焼度で与えていますが、日数で与えることも可能です。その場合には Burner クラスの PutBurnTime メソッドの二つ目の引数である boolean 変数を false にする必要があります。また、燃焼ステップの長さを積算値ではなく各ステップの値として指定することも出来ます。その場合は PutBurnTime メソッドの三つ目の引数である boolean 変数を false にして下さい [12] 。

また、各燃焼ステップの時間長さが全てのステップで同一となる場合は、PutBurnTime メソッドの一つ目の引数を配列ではなく実数で与えることも可能です。

燃焼計算では、各燃焼ステップをさらにサブステップに分割して計算を行います。線出力が燃焼（中性子束規格化）条件として与えられている場合、各サブステップでは線出力により中性子束の規格化を行います。線出力一定の場合、燃焼ステップ間では核燃料が減損するため中性子束レベルを燃焼とともに大きくする必要があります。燃焼ステップをサブステップに分割し、各サブステップで中性子束の規格化を行うことにより、その効果を考慮することが出来るわけです。Burner ではこのサブステップ数のデフォルトを 20 と指定していますが、このサブステップ数を変更したい場合には Burner クラスの PutSubstep(int i) メソッドを使用します。引数には新たなサブステップ分割数を渡してあげます。燃焼ステップとサブステップの概念について **Fig. 4** にまとめます。

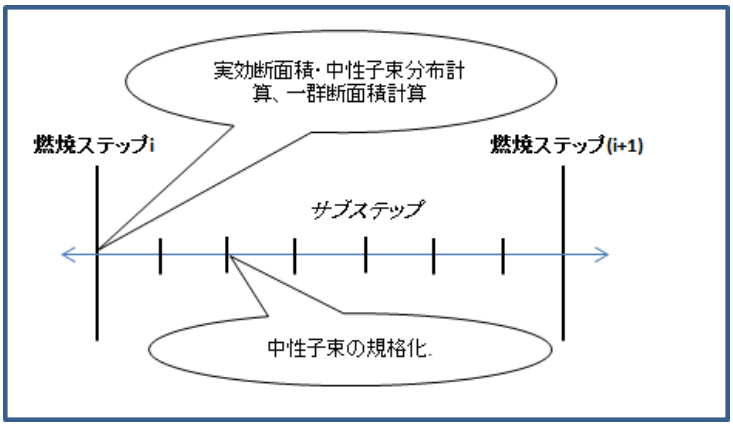

Fig. 4: 燃焼ステップとサブステップの概念

さらに、線出力ではなく、中性子束レベルを燃焼条件として指定することも可能です。その場合は PutPowerDensityList メソッドではなく、PutFluxLevelList メソッドを用います。中性子束レベルを直接指定する場合の燃焼条件データの例を以下に示します。なお、中性子束レベルの単位は「/cm^2/s」となります。

Listing 8: 燃焼条件データの入力（中性子束レベルを直接指定する場合）

```
1  // +++ Burnup history data +++
2  bn.PutBurnStep(10);
3  real flux_level_list[]={
4    5.5856e15, 0.,
5    5.5856e15, 0.,
6    5.5856e15, 0.,
7    5.5856e15, 0.,
8    5.5856e15, 0.
9  }; // [/cm2/s]
10 real burn_time[]={
11   148, 60,
12   148, 60,
13   148, 60,
14   148, 60,
15   148, 4*365
16 }; // [day]
```

[12] 燃焼と冷却を繰り返す計算を行う際に、燃焼ステップの長さを日数で、かつ積算値で与える場合には、冷却期間を積算値として与えないこと、燃焼期間の積算値には冷却期間は含まれないことに注意して下さい。

```
17    bn.PutFluxLevelList(flux_level_list);
18    bn.PutBurnTime(burn_time, false, false); // [GWd_t/day][accumulate/not]
```

4 燃焼計算の実行と結果の出力

4.1 燃焼計算の実行

以上で燃焼計算に必要な情報がインスタンス bn に与えられました。あとは計算を実行するのみです。main.burn_tutorial.cxx において計算を実行させる部分を以下に示します

Listing 9: 計算の実行

```
1  bn.Calculation(bu);
```

これにより計算が行われ、計算結果がインスタンス bn のメンバ関数として保持されます。

4.2 計算結果の画面上への出力

計算結果にアクセスするためには、計算が終了したあとにインスタンス bn の出力メソッドを実行することになります。main.burn_tutorial.cxx において計算結果を画面上に出力させる部分を以下に示します。

Listing 10: 計算結果の画面上への出力

```
1  bn.ShowEigenvalue();
2  bn.ShowNeutronFluxHistory();
3  //bn.ShowNuclideList();
4
5  bn.ShowNumberDensityChange();
6  string nuc_name1[]={"Xe135","Xe137"};
7  bn.ShowNumberDensityChange(2,nuc_name1);
8
9  string prt_nuc_name[]={"Mo098","Mo099","Mo100"};
10 string prt_nuc_name2[]={"FP","HM","ALL"};
11 bn.ShowNumberDensityHistory(3,prt_nuc_name,bu,"nd");
12 bn.ShowNumberDensityHistoryAtomWise("Mo",bu,"nd");
13
14 //bn.ShowHeavyMetalWeightRatio(bu);
15 //bn.ShowCrossSection(3,prt_nuc_name);
16 //bn.ShowCrossSection(0); // for burnup step 0
17 //bn.ShowFuelNeutronFlux(0);
18 //bn.ShowFuelNeutronFluxExcel(0);
19 int prt_nuc2=5;
20 string prt_nuc_nam2[]={
21   "U235","U238","Pu239","Pu240","Pu241"
22 };
23 //bn.ShowNuclideWiseContributionForFission(prt_nuc2,prt_nuc_nam2);
24 //bn.ShowCaptureToDecayRatio(prt_nuc2,prt_nuc_nam2,bu);
25 //bn.ShowGroupDependentCrossSection("U235");
26 //bn.ShowGroupDependentCrossSectionExcel("U235");
```

この例では以下のような計算結果を出力するためのメソッドが示されています。

- ShowEigenvalue：燃焼ステップ毎の体系の中性子無限増倍率と転換比、燃料領域の中性子束レベル（ただし最初のサブステップのみ）を出力します。
- ShowNeutronFluxHistory：燃焼ステップ毎の中性子束レベルを、燃料領域、被覆管領域について出力します。
- ShowNuclideList：燃料領域に含まれる原子核の一覧を出力します。
- ShowNumberDensityChange：燃焼前後の全ての原子核の数密度を出力します。ただし、燃焼前、燃焼後の数密度がともにゼロの核種については出力されません。なお、引数として実数型のデータを与えた場合には、燃焼初期もしくは末期の数密度がその値より大きい核種のみを出力します。また、引数として核種数と核種名のリストを与えると、指定した核種のみの情報を出力します。
- ShowNumberDensityHistory：一つ目の引数で指定した数の核種についてさまざまなデータを出力します。なお、核種の指定は二つ目の引数に核種名のリストを与えることで行います。核種名は、「Xe135」等同位体で与えることに加えて、「Xe」のように元素で与えることも可能です。さらに、「FP」「HM」

「ALL」とすることで、それぞれ FP の総和、重核種の総和、全核種の総和を指定することも出来ます[13]。また、４番目の引数として、どのようなデータを出力するか指定するキーワードを与えます。以下にキーワードと出力されるデータの対応を示します。

キーワード	出力されるデータ
nd	全体積中の原子数密度 [10^{24}]
nd_per_vol	単位体積あたりの原子数密度 [$10^{24}/\mathrm{cm}^3$]
bq	全体積中の放射能 [Bq]
bq_per_thm	装荷燃料トンあたりの放射能 [Bq/tHM]
kg_per_thm	装荷燃料トンあたりの重量 [kg/tHM]
w_per_thm	装荷燃料トンあたりの崩壊熱 [W/tHM]
w_gamma_per_thm	装荷燃料トンあたりの崩壊熱（ガンマ線成分）[W/tHM]
w_beta_per_thm	装荷燃料トンあたりの崩壊熱（ベータ線成分）[W/tHM]
sv_per_thm_ing	装荷燃料トンあたりの経口吸収線量 [Sv/tHM]
sv_per_thm_inh	装荷燃料トンあたりの吸入吸収線量 [Sv/tHM]

このメソッドの出力において、最後の欄に数値が与えられますが、これは表示されている全ての値の総和になります。なお、５番目の引数に true を指定することにより、総和のみを表示させることも可能となっています。

- ShowNumberDensityHistoryAtomWise：元素名を指定することにより、その全ての同位体についてのデータを出力します。出力するデータは３番目の引数で与えます。キーワードは ShowNumberDensityHistory メソッドと同様です。４番目の引数に true を指定することにより、総和のみを表示させることも可能です。

- ShowHeavyMetalWeightRatio：燃焼前後の重核種重量などを出力します。

- ShowCrossSection：燃焼中の１群断面積（後述の付録 A を参照のこと）を出力します。引数として、出力する核種の数と核種名のリストを与えた場合には、それらの核種について燃焼度依存の１群断面積が出力されます。一方、引数として整数値を１つのみ与えた場合は、その数値に対応する燃焼ステップでの全ての核種の１群断面積が出力されます。さらに、隠れオプションとして、３つ目の引数として true を与えると、１群断面積ではなく反応率が出力されます。また、４つ目の引数（整数）を与えることが出来、これは表示される小数点以下の桁数に対応します。

- ShowFuelNeutronFlux：燃料領域の中性子束のエネルギースペクトルを出力します。引数には表示したい中性子束スペクトルの燃焼ステップを指定します。この例では「0」が与えられているので、燃焼初期の中性子スペクトルが出力されます。ペアの数字が 107 個出力されますが、これは各エネルギー群の中性子束に対応しており、左側の数値がそのエネルギー群の上限エネルギー、右側の数値が中性子束となります。なお、中性子束はエネルギー群のレサジー（$\ln(E_0/E)$ で定義）幅で除されているため、このデータをプロットする場合には X 軸を対数表示にする必要があります。

- ShowFuelNeutronFluxExcel：ShowFuelNeutronFlux とほぼ同一のメソッドですが、Excel でヒストグラム状のグラフを作成するのに適したフォーマットで出力します。

- ShowNuclideWiseContributionForFission：燃焼ステップ毎の全核分裂反応に対する核種毎の寄与割合を出力します。引数で指定した核種の情報のみを表示し、それらの総和を括弧内に示します。

- ShowCaptureToDecayRatio：燃焼中の中性子捕獲反応率と崩壊定数の比を出力します[14]。

[13] HM は原子番号が 80 以上のものとしています。

[14] ある核種の数密度について、中性子捕獲による減少率は $N\sigma\phi$、崩壊による減少率は λN と書けます。このメソッドでは $\sigma\phi/\lambda$ を出力しますので、着目核種の減少における中性子捕獲と崩壊の割合を定量的に知ることが出来ます。

- ShowGroupDependentCrossSection：燃焼後時点でのエネルギー群依存の断面積を、引数で与えた核種について出力します。
- ShowGroupDependentCrossSectionExcel：ShowGroupDependentCrossSection とほぼ同一ですが、Excel での編集に適したフォーマットで出力します。

4.3 計算結果の外部ファイルへの出力

ランダムサンプリング法を用いた核種生成量の不確かさ解析等、多ケースの燃焼計算の結果を利用するような場合には、各ケースの計算結果を外部ファイルに保存しておくことが想定されます。このような場合のために、WriteFileNumberDensity メソッドが実装されています。このメソッドでは、指定した燃焼サイクル初期における無限中性子増倍率と核種数密度を、指定した名前の外部ファイルに出力することが可能です。WriteFileNumberDensity メソッドの使用例を以下に示します。

Listing 11: 計算結果の外部ファイルへの出力

```
1  int cyc_num=2;
2  int cyc[]={0,20};
3  bn.WriteFileNumberDensity(cyc_num,cyc,"./","out_nd",10);
```

一行目の cyc_num で情報を出力するサイクル数を指定し、二行目の cyc で燃焼サイクルを指定します。ここで、0 は一番最初のサイクルに、サイクル数を 20 として計算した際にその同数である 20 とした場合には燃焼後の時点に、それぞれ対応することになります。また、WriteFileNumberDensity メソッドの 3 つ目、4 つ目の引数は、それぞれ外部ファイルを出力するディレクトリ名（スラッシュまで必要です）とファイル目に対応します。最後の 5 つ目の引数は、出力する数値の小数点以下の桁数に対応します。

出力されるデータは、サイクル毎に、燃焼サイクル、中性子無限増倍率が記載され、その後、燃料領域における核種数密度が全ての核種に対して記載されます。核種数密度の並びに対応する核種 ID については、例えば ShowNumberDensityChange メソッドで引数を-1 とすれば全ての核種が出力されるので、それを参考にするとよいでしょう。

4.4 計算結果の数値データとしての取り出し

Burner を用いた燃焼計算を繰り返し行い、特定の核種についての燃焼後の数密度データを繰り返し毎に保存させておき、繰り返しが終わったあとに結果をまとめて表示させる、などという操作が必要になる場合があるかもしれません。そのような場合には、計算結果の数値データに直接アクセスすることになります。数密度データに対しては、GetNuclideDensity メソッドを利用することが出来ます。このメソッドでは、一つ目の引数に数密度データを取り出したい核種の名前（文字列型）もしくは ID を、二つ目の引数に取り出したい燃焼ステップを指定します。

5 練習問題

以下の条件について、入力データを作成し、燃焼計算を行って下さい。

- ピンピッチ：1.265 cm
- 燃料ペレット半径：0.412 cm
- 被覆管外側半径：0.476 cm
- 燃料/被覆管/冷却材温度：968.8 / 604.0 / 574.2 K
- 線出力：179 W/cm
- 燃焼期間：45 GWd/t まで燃焼、その後 4 年冷却

燃料としては、UO_2 燃料と MOX 燃料の二種類について計算して下さい。U-235 の濃縮度は前者で 4.1 wt%、後者で 0.2 wt%、Pu 含有率は MOX 燃料で 10.0 wt%となっています。燃料の数密度を **Table 1** に、被覆管、冷却材の数密度を **Table 2** にまとめます。なお、これらの表中の「Index」は CBZ で用いている、個々の原子核に対して定義される Index を示しています [15]。

Table 1: Initial fuel composition

Nuclide	Index	Number density (10^{24}/cm^3)	
		UO_2	MOX
^{235}U	922350	9.349E-4	4.104E-5
^{238}U	922380	2.159E-2	2.022E-2
^{238}Pu	942380		4.731E-5
^{239}Pu	942390		1.223E-3
^{240}Pu	942400		5.586E-4
^{241}Pu	942410		2.069E-4
^{242}Pu	942420		1.418E-4
^{241}Am	952410		6.007E-5
^{16}O	80160	4.505E-2	4.500E-2

また、燃焼計算の結果を用いて、以下の設問に答えて下さい。燃焼チェーンの図が必要になると思うので、前述したように SRAC のマニュアルの必要部分を印刷しておいて下さい。また、「このような情報が欲しい」「このような計算をしたい」という場合には連絡して下さい。

問１：UO_2、MOX 燃料について、燃焼に伴う無限増倍率の変動（燃焼（欠損）反応度と呼びます）を図を用いて比較せよ。また、`ShowFuelNeutronFlux` メソッドを用いて、UO_2、MOX 燃料での中性子束エネルギースペクトルの図を作成せよ（縦軸を線形とした場合と対数とした場合の 2 種類の図を作成するものとする）。

問２：UO_2、MOX 燃料について、燃焼に伴う U-235、-238、Pu-238、Pu-239、-240、-241、-242、Am-241 の数密度の変動を図に示すとともに、傾向を簡単に説明せよ。

[15] CBZ における原子核に対する ID の与え方ですが、原子核の原子番号を Z、質量数を A、励起レベルを L としたとき、その原子核の ID を $Z \times 10000 + A \times 10 + L$ と定義します。基本的に大部分の原子核は基底状態にあるため $L = 0$ となりますが、一部の原子核で励起状態で準安定となるものがある（例えば Am-242 など）ため、その場合は $L > 0$ となります。また、天然同位体組成の原子核については、質量数を 0 として定義します。例えば天然同位体組成の鉄については、ID は「260000」として与えられます。

Table 2: Number densities of clading and moderator regions

Region	Nuclide	Index	Number density (10^{24}/cm^3)
Clading	Zr	400000	3.786E-2
	Fe	260000	2.382E-4
	Cr	240000	6.770E-5
Moderator	^{1}H	10010	5.572E-2
	^{16}O	80160	2.786E-2
	^{10}B	50100	4.592E-6
	Ni	280000	3.688E-4
	Cr	240000	1.609E-4
	Fe	260000	1.306E-4

問３：UO_2、MOX 燃料について、燃焼に伴う核分裂反応における核種毎の寄与割合の変化を図に示せ（`Burner` クラスのメソッド `ShowNuclideWiseContributionForFission` を使うこと）。

問４：UO_2 燃料について、燃焼に伴う Cs-133、-134、-137 の数密度の変動を図に示せ。また、それぞれの曲線を多項式で近似するとした場合にはどの程度の次数が必要となるか見積り、その次数が核種によって異なるならばその原因について考えよ。

問５：Xe-135 は強い中性子吸収体であり、I-135 のベータ崩壊により生成する。UO_2 燃料について、燃焼に伴う Xe-135、I-135 の数密度の変動を図に示せ。Cs の同位体とは異なる挙動を示すが、その理由について述べよ。

問６：Pu-238 の生成メカニズムについて、UO_2、MOX 燃料それぞれについて述べよ。

問１では中性子束のエネルギースペクトルが UO2 と MOX 燃料とで大きく異なることが分かるかと思います。これは熱中性子エネルギー領域における U、Pu 同位体の中性子吸収断面積を観察することで理解できると思いますので、千葉のサイトの項目「核データ」にあるテキスト「炉物理計算で重要となる核データ」を眺めて下さい。

また、問６における核種の生成メカニズムを評価する方法として、燃焼チェーンを「分断」する方法があります。つまり、「この核種の生成にはこの流れが重要だな」と思ったとすると、その流れを分断した計算を行ってみることで、その直感が正しいかを定量的に評価できるわけです。チェーンを分断するメソッドとして、`BurnupChain` クラスに `CutChainForDecay`、`CutChainForCapture`、`CutChainForN2N` なるメソッドがありますので、それを利用して下さい。このメソッドの引数は文字列データとなっていますが、ここではチェーンを切断する核種の名前を与えてあげて下さい。例えば、Pu-241 の崩壊チェーンを分断したい場合には `bu.GetBurnupChain().CutChainForDecay("Pu241");` を適切な場所に追加してあげれば OK です。もちろん、計算する前（`bn.Calculation(bu);` を行う前）でないと意味がありませんので注意して下さい。

また、チェーンを分断する方法とは別に、特定の核種の核分裂収率をゼロにする方法もあります。この場合は、`BurnupChain` クラスの `SetZeroFissionYield` メソッドを利用して下さい。このメソッドの引数はふたつの文字列データであり、ひとつめは核分裂性核種に、ふたつめは FP に対応します。例えば、U-235 の核分裂で発生する Xe-135 の収率をゼロにしたい場合は `bu.GetBurnupChain().SetZeroFissionYield("U235","Xe135");` を適切な場所に追加してあげれば OK です。

6 Burnup モジュールを用いた一点炉での燃焼計算

第 3 章で触れているように、実際の燃焼計算を行っている（すなわち燃焼方程式を解く役割を果たしている）ものが Burnup モジュールです。Burnup モジュールでは、与えられた燃焼条件（燃焼チェーン、初期数密度、中性子束レベル、燃焼期間、各核種の一群反応断面積）を入力として、燃焼後の核種数密度を種々の数値計算法により求めることができます。従って、これら、特に一群反応断面積の数値などが既知であり、また重核種のみなど限られた数の核種の燃焼計算を行う場合には、Burner モジュールよりもむしろ Burnup モジュールを直接使ったほうが便利です（Burner モジュールは、燃焼時点に応じたピンセル体系の一群断面積を計算し、Burnup モジュールを適宜走らせるためのもの、と考えるとよいでしょう）。

それでは Burnup モジュールの使用例として、5 つの重核種（U-238、Np-237、Pu-238、-239、-240）の燃焼計算例を以下に示します。この例では U-238 のみに初期数密度 1.0 を与えて、一定の中性子束レベルで 365 日間燃焼させたときの数密度の変化を計算しています。

Listing 12: Burnup モジュールの使用例 (1)

```
1  Burnup bu;
2
3  string cbglibdir("../../");
4  bu.SetDefaultChain();
5  bu.ReadReactionEnergyFromFile(cbglibdir,"sractype");
6  bu.GetBurnupChain().ReadDecayConstantFromFile(cbglibdir,"srac_org");
7  bu.ReadAtomicMassDataFromFile(cbglibdir,"jeff311");
8
9  //++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
10
11 int nucnum=7;
12 bu.PutNucnum(nucnum);
13 //                ID    NUCID    Init.      One-group cross section
14 //                                ND      (fission)   (capture)    (n,2n)
15 bu.PutNuclideData( 0,  922380,   1.0,    7.244e-2,   2.066e-1,   1.932e-3);
16 bu.PutNuclideData( 1,  932370,   0.0,    4.58339e-1, 1.20346,    2.53620e-4);
17 bu.PutNuclideData( 2,  942380,   0.0,    0.,         0.,         0.);
18 bu.PutNuclideData( 3,  942390,   0.0,    1.72322,    3.60183e-1, 5.14009e-4);
19 bu.PutNuclideData( 4,  942400,   0.0,    0.,         0.,         0.);
20 bu.PutNuclideData( 5, 9990000,   0.0,    0.,         0.,         0.);
21 // Pseudo FP from Pu-239 fission
22 bu.PutNuclideData( 6, 9980000,   0.0,    0.,         0.,         0.);
23 // Pseudo FP from U-238 fission
24
25 bu.CalTransitionMatrixFluxDependentPart();
26 bu.CalTransitionMatrixFluxInDependentPart();
27
28 //++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
29
30 bu.BurnupCalculation(5.0e14, 60*60*24*365, "mmpa", true);
31  // Flux level & Burnup period in sec
32
33 real sum=0;
34 for(int i=0;i<nucnum;i++){
35   sum+=bu.GetDensity(i);
36 };
37 cout<<"\n  Total number density after burnup : "<<sum<<"\n";
```

燃焼チェーンの指定方法については第 3.1 節ですでに説明していますので、そちらを参照して下さい。基本的には、現在行っている計算で用いている重核種のチェーン（例えば ADS 計算を行っている場合には BurnupChain クラスの「SetHeavyMetalChainForADS」メソッドで定義されるチェーン）を用いるとよいでしょう。主要な重核種のみを扱い、かつ核分裂生成物を「擬似核種」として簡略的に扱う場合には、この例のように「SetDefaultChain」メソッドによりデフォルトのチェーンを指定するとよいでしょう。

Burnup クラスのインスタンスに対して計算対象とする核種数を PutNucnum メソッドにより与えたあとに、個々の核種の情報（核種 ID、初期数密度、1 群反応断面積）を PutNuclideData メソッドにより与えます。なお、この PutNuclideData メソッドにより入力されない核種については、燃焼チェーン上にその核種が存在したとしても、数密度が計算されないことに注意して下さい。例えば、デフォルトチェーンには Pu-239、-240、-241 が含まれますが、Pu-240 が PutNuclideData メソッドで Burnup クラスのインスタンスに与えられないと、Pu-240 の数密度は計算されませんので、Pu-239 からの Pu-240 を介した Pu-241 の生成は計算されないことになります。

必要なデータを与えたあとには、燃焼方程式に現れる燃焼行列の断面積に依存する成分と依存しない成分を計算する必要があるので、CalTransitionMatrixFluxDependentPart メソッド、CalTransitionMatrixFluxInDependentPart メソッドを行います。

その後、燃焼計算を行うことが可能となります。燃焼計算のためのメソッド BurnupCalculation では、中性子束レベル、燃焼期間（秒単位）を１つ目と２つ目の引数で指定します。３つ目の引数は数値計算手法を指定しており、問題がない限り「mmpa」と指定して MMPA 法を使うとよいでしょう。４つ目の引数は結果の出力オプションに対応し、true にすると、燃焼前後の数密度が出力されます。

また、燃焼期間をいくつかのステップに分割し、各ステップ中での数密度を出力させたい場合には、以下のようなプログラムとすればよいでしょう。

Listing 13: Burnup モジュールの使用例 (2)

```
1  int cycle=40;
2  real delta_t=10; // [days]
3
4  real t=0.;
5  for(int i=0;i<cycle+1;i++){
6    cout<<"  "<<t<<" ";
7    for(int j=0;j<nucnum;j++){
8      cout<<bu.GetDensity(j)<<" ";
9    };
10   cout<<"\n";
11   if(i!=cycle){
12     bu.BurnupCalculation(5.0e14, delta_t*24*60*60, "mmpa", false);
13     t+=delta_t;
14   };
15 };
```

GetDensity メソッドでは、その時点の数密度（核種は引数で指定）を取り出すことができます。

A 反応率と一群断面積

燃焼計算では、中性子と個々の原子核が燃焼中にどれだけ反応（捕獲、核分裂、(n,2n) など）したかを定量的に知る必要があります。中性子と原子核との反応量は、一般的に**反応率**と呼ばれます（単位時間、単位体積あたりに起こる反応の数が厳密な反応率の定義となります)。

本稿では、中性子エネルギーを 107 メッシュ（群）で扱っているため、反応断面積、中性子束ともに 107 群で与えられます。この場合、核種 i の（微視的）中性子捕獲反応率は、$\sum_{g=1}^{107}\sigma_{c,g}^{i}\phi_g$ で与えられます。ここで、$\sigma_{c,g}^{i}$、ϕ_g はそれぞれ核種 i の g 群の中性子捕獲断面積、g 群の中性子束を示します。

燃焼計算では、一般的に全中性子束 ϕ（$=\sum_{g=1}^{107}\phi_g$）を入力として与えますので、反応率を計算するためには、反応率を保存するような「平均」断面積をあらかじめ計算しておく必要があります。この平均断面積を**一群断面積**（エネルギーを 1 群として扱った場合の断面積）と呼び、$\sigma_c^i=\sum_{g=1}^{107}\sigma_{c,g}^{i}\phi_g/\sum_{g=1}^{107}\phi_g$ で計算されます。このようにして決められた一群断面積を用いることで、任意の大きさの中性子束 $\tilde{\phi}$ に対する反応率を $\sigma_c^i\tilde{\phi}$ と計算することが出来ます。

4.2 節で述べたメソッド `ShowCrossSection` では、この一群断面積が指定された核種について出力されます。

なお、4.2 節で述べられているように、メソッド `ShowCrossSection` では反応率も出力することが出来ます。ここで出力される反応率は、微視的反応率に数密度が乗ぜられたもの（すなわち、反応 x の場合は $N^i\sigma_x^i\phi$）となっており、「巨視的断面積を核種毎に表示したもの」とも見做すことができます。ここでは中性子束は中性子吸収反応率（中性子の総吸収数）の総和 A が 1.0 となるように規格化されています。従って、核分裂による中性子の総発生数を F としたとき、$k_\infty=F/A=F$ の関係が得られます。核種 i の吸収反応率が $C^i=N^i\sigma_c^i\phi$ であるとき、大雑把に言うと、核種 i の中性子吸収は k_∞ に対して $-k_\infty C^i$ の影響を有している、ということになります。また、$k_\infty\approx 1.0$ と見做すならば、単純に $-C^i$ の影響を有する、とすることも可能です。

B 燃焼後の冷却計算

出力ゼロの状態では核種の崩壊のみを考慮すればよく、そのような計算を「冷却計算」と呼ぶ。Burner の燃焼計算を行うメソッド「Calculation」では、燃焼ステップ毎に中性子核反応断面積を計算し、衝突確率法を用いて固有値（無限増倍率）や中性子束分布などを計算しているが、冷却計算ではそのような情報は基本的には不要であるため、計算を省略して計算時間を短縮することが出来る。Burner には冷却計算のためのメソッドとして「CoolingCalculation」が実装されている。その計算例を以下に示す。

Listing 14: 冷却計算の例（main.burn_cooling.cxx）

```
1  // +++ Burnup history data +++
2  bn.PutBurnStep(21);
3  real power_density_list[]={
4    179., 179., 179., 179., 179.,
5    179., 179., 179., 179., 179.,
6    179., 179., 179., 179., 179.,
7    179., 179., 179., 179., 179.,
8    0.,
9  }; // [W/cm]
10 real burn_time[]={
11    0.1, 1.0, 2.5, 5.0, 7.5,  10., 12.5, 15., 17.5, 20.,
12    22.5, 25., 27.5, 30., 32.5,  35., 37.5, 40., 42.5, 45., // GWd/t
13    4.*365., // day
14 };
15 bn.PutPowerDensityList(power_density_list);
16 bn.PutBurnTime(burn_time,true,true); // [GWd_t/day][accumulate/not]
17
18 bn.Calculation(bu);
19
20 +++ cooling calculation
21
22 bn.PutBurnStep(4);
23 real burn_time2[]={4*365, 6*365, 8*365, 10*365};
24 bn.PutBurnTime(burn_time2);
25
26 bn.CoolingCalculation(bu,1);
27
28 int prt_nuc=3;
29 string prt_nuc_nam[]={"FP","HM","ALL"};
30 bn.ShowNumberDensityHistory(prt_nuc,prt_nuc_nam,bu,"w_per_thm");
```

この例では、まず 45GWd/t までの燃焼と 4 年間の冷却を「Calculation」メソッドにより行っている。その後、冷却計算のため、冷却計算での燃焼ステップと燃焼ステップの情報を新たに Burner クラスのインスタンス bn に与えてやり（例の 22 から 24 行目に該当）、その後、「CoolingCalculation」メソッドにより冷却計算を行っている。CoolingCalculation メソッドの一つ目の引数は Calculation メソッドと同様に Burnup クラスのインスタンスである。また、二つ目の引数は燃焼ステップのサブステップ分割数である。ここで、**冷却期間は各々のステップの長さであり、「累積長さ」ではない**ことに注意が必要である。また、前述している通り、PutBurnTime メソッドで単一の実数を与えた場合は、全ての燃焼ステップで同一の冷却時間が設定される。

なお、冷却計算では固有値などは計算されないため、「ShowEigenvalue」メソッド等は使用できず、数密度が関係する物理量（数密度や崩壊熱など）のみ出力するメソッドを用いて計算結果を得なければならない。この例では「ShowNumberDensityHistory」メソッドを用いて、FP 核種、重核種、これら両方に由来する崩壊熱を出力させている。

B.1 冷却計算におけるリスタート機能

同一の燃焼後組成からさまざまな条件での冷却計算を行う場合、毎回、同一の初期燃料組成から同じ燃焼計算を行うことは時間の無駄である。そのため、燃焼後組成をファイルに出力しておき、それを利用して条件を変えた冷却計算を行う「リスタート」計算が可能となっている。燃焼後組成をファイルに出力するためには、Burner クラスの「WriteFileFuelNumberDensity」メソッドを用いる。ひとつめの引数はディレクトリ位置、ふたつめはファイル名に対応する。例を以下に示す。

Listing 15: 燃焼後組成のファイル保存例（main.restart1.cxx）

```
// +++ Burnup history data +++
bn.PutBurnStep(21);
real power_density_list[]={
  179., 179., 179., 179., 179.,
  179., 179., 179., 179., 179.,
  179., 179., 179., 179., 179.,
  179., 179., 179., 179., 179.,
  0.,
}; // [W/cm]
real burn_time[]={
   0.1, 1.0, 2.5, 5.0, 7.5,  10., 12.5, 15., 17.5, 20.,
   22.5, 25., 27.5, 30., 32.5,  35., 37.5, 40., 42.5, 45., // GWd/t
   4.*365., // day
};
bn.PutPowerDensityList(power_density_list);
bn.PutBurnTime(burn_time,true,true); // [GWd_t/day][accumulate/not]

bn.Calculation(bu);
bn.WriteFileFuelNumberDensity("./","fuel_den");
```

ファイルから組成を読み込んで冷却計算を行う場合は、まず燃焼初期の組成データを Burner クラスのインスタンスに入力し、その後、Burner クラスのメソッド「CalHeavyMetalInitialWeight」を実行しなければならない。これは、初装荷燃料の重核種重量が必要となる場合があるからである。その後は通常の処理と同様である。例を以下に示す。

Listing 16: 燃焼後組成のファイルからの読み込みと冷却計算例（main.restart2.cxx）

```
bn.CalHeavyMetalInitialWeight(bu);

bn.ReadFileFuelNumberDensity("./","fuel_den");

bn.PutBurnStep(4);
real burn_time2[]={4*365, 6*365, 8*365, 10*365};
bn.PutBurnTime(burn_time2); // [GWd_t/day][accumulate/not]

bn.CoolingCalculation(bu,4);
```

B.2 再処理を想定した冷却計算

使用済み燃料の放射能の減衰を評価したい際にも冷却計算を行うことになるが、使用済み燃料の再処理を想定した場合には、例えば、燃料取り出し４年後にウランとプルトニウムの同位体が除去されるというようなことを考えなくてはならない。そのような場合の計算例を以下に示す。

Listing 17: 使用済み燃料の再処理を想定した冷却計算例（main.cool_reprocessing.cxx）

```
// +++ cooling

real *burn_time2=new real[120];

// +++ 4-year cooling (abount 1,420 days)
bn.PutBurnStep(31);
real un=1.;
for(int i=0;i<31;i++){
  burn_time2[i]=un;
  un*=1.2;
};
bn.PutBurnTime(burn_time2); // day
bn.CoolingCalculation(bu,1);

// +++ reprocessing
bn.NumberDensityReset(920000,929999,0.); // U
bn.NumberDensityReset(940000,949999,0.); // Pu

// +++ furtherr cooling
bn.PutBurnStep(90);
for(int i=0;i<90;i++){
  burn_time2[i]=un;
  un*=1.2;
};
bn.PutBurnTime(burn_time2); // day
bn.CoolingCalculation(bu,1);

int prt_nuc=9;
string prt_nuc_nam[]={
```

```
30      "ALL","FP","HM","Pu238","Pu240","Pu241","Am241","Am243","Np237"
31    };
32    bn.ShowNumberDensityHistory(prt_nuc,prt_nuc_nam,bu,"bq_per_thm");
```

この例では、通常の燃焼計算を行ったあと、およそ 4 年間の冷却計算を行い（5 から 13 行目）、その後、燃料再処理を想定してウランとプルトニウム同位体の数密度をゼロとし（15 から 17 行目）、再度冷却計算を行っている（19 から 26 行目）。冷却計算の場合は、計算結果は全ての履歴が保存されるため、32 行目の「ShowNumberDensityHistory」メソッドでは、全冷却期間（再処理前後）の計算結果が出力される。

なお、Burner クラスの「NumberDensityReset」メソッドでは、核種 ID が一つ目の引数以上で二つ目の引数以下の核種の数密度を、三つ目の引数の値（この例ではゼロ）に再定義する。**ここでは数密度の絶対値を指定しているが、例えば数密度を定数倍したいときには、その定数にマイナスを乗じたものを三つ目の引数に指定するとよい。**

B.3 冷却期間中の崩壊による組成変動の核種毎反応度効果の評価

原子炉停止後の冷却期間中に、原子核の崩壊により組成が変動し、系の反応度が変化することがよく知られている（代表的なものとしては、Xe-135 や Sm-149 が挙げられるであろう）。Burner には、冷却計算開始時の組成を基準とし、冷却期間中における各核種の基準状態に対する変動が与える反応度効果を評価するメソッド「NuclideWiseReactivityEffectCalculationDuringCooling」が実装されている。このメソッドでは、冷却計算開始時点の系の中性子束、随伴中性子束を計算し、冷却計算中の各計算点において、上記の反応度を一次摂動により評価する。入力例を以下に示す。

Listing 18: 冷却期間中の崩壊による組成変動の核種毎反応度効果の評価計算例（main.cool_reactivity.cxx）

```
1   bn.Calculation(bu);
2
3   //
4   bn.PutBurnStep(100);
5   real *burn_time2=new real[100];
6   real un=0.01;
7   for(int i=0;i<100;i++){
8     burn_time2[i]=un;
9     un*=1.2;
10  };
11  bn.PutBurnTime(burn_time2);
12
13  int prt_nuc=8;
14  string prt_nuc_nam[]={"Xe135","Sm149","Pu239","Pu241","Am241","Pm147","Sm147","Pm148m"};
15  bn.NuclideWiseReactivityEffectCalculationDuringCooling(bu,prt_nuc,prt_nuc_nam);
```

C リスタート機能を利用した燃焼後数密度を用いた固有値計算

異なるコード間で燃焼後の固有値の比較を行う際、比較的大きな差異が観察されたときに、その差異がどの核種数密度の差異に由来するかを知ることが必要となる場合がある。このようなときには、あるコードの計算結果に基づく燃焼後数密度で基準となる固有値を計算し、核種毎に異なるコードで得られた燃焼度数密度に置換して固有値を再計算し、基準計算値からの変動量を観察することで、固有値の差異の原因を特定することが可能となる。そのような場合には、予め計算しておいた燃焼後数密度を外部ファイルに保存しておき、逐次それを読み込んで、必要な変更を加えて固有値計算を行えばよい。

燃焼後数密度の外部ファイルへの保存については、B.1 節で説明されているリスタート機能を活用すればよい。

燃焼後数密度を計算しておけば、以後の計算では改めて燃焼計算を行う必要はなく、外部ファイルに与えられている燃焼後数密度データを読み込み、Burner クラスのインスタンスに与えてやればよい。そのためのメソッド「ReadFileFuelNumberDensity」についても B.1 節に説明されている。数密度を読み込んだあとは、Burner クラスの「EigenvalueCalculation」メソッドにより固有値を計算することが出来る。なお、読み込んだ数密度に変更を加えたい場合は、Burner クラスのインスタンスが保有している Medium クラスのインスタンス med[0] の数密度データを書き換えればよい。下記に、U-235 の数密度を外部ファイルで与えられている数値から 1.1 倍して固有値計算を行うための例を示す。

Listing 19: リスタート機能を利用した燃焼後数密度を用いた固有値計算の例

```
1   /*
2   bn.Calculation(bu);
3
4   bn.WriteFileFuelNumberDensity("./","fuel_den");
5   */
6
7   bn.ReadFileFuelNumberDensity("./","fuel_den");
8
9   real den=bn.GetMedium(0).GetNuclide(922350).GetDensity();
10  bn.GetMedium(0).GetNuclide(922350).PutDensity(den*1.1);
11
12  bn.EigenvalueCalculation();
```

D Co-59 の放射化計算

被覆管材料や冷却材に微量に含まれる不純物の放射化量を評価しなければならない場合がある。Burner では、計算した燃焼履歴を用いて、被覆管領域に含まれる Co-59 の放射化量を計算することが出来る。

Co-59 は (n,g) 反応により Co-60 に変換するが、一定の割合で、準安定状態（meta-stable）のものが生成される。ただし、Co-60 の準安定状態の半減期は約 10 分であり、原子炉運転後の放射能計算等の時間オーダーを考えると、実用上は、Co-59 の (n,g) 反応により全て基底状態の Co-60（半減期 5.27 年）が生成されると考えて問題ない。この場合、Co-59 と Co-60 の数密度 $N_9(t)$、$N_0(t)$ は以下の微分方程式に従う。

$$\frac{dN_9(t)}{dt} = -\sigma_c \phi N_9(t), \tag{1}$$

$$\frac{dN_0(t)}{dt} = \sigma_c \phi N_9(t) - \lambda_0 N_0(t) \tag{2}$$

ここで、σ_c は Co-59 の微視的捕獲断面積である。また、Co-60 の中性子との反応は無視している。

この微分方程式は解析的に解けて、以下の解が得られる。

$$N_9(t) = N_9(0)\exp(-\sigma_c \phi t), \tag{3}$$

$$N_0(t) = \frac{N_9(0)\sigma_c \phi}{\lambda_0 - \sigma_c \phi}\{\exp(-\sigma_c \phi t) - \exp(-\lambda_0 t)\} + N_0(0)\exp(-\lambda_0 t) \tag{4}$$

Burner では、各燃焼ステップにおける中性子束レベル、中性子束のエネルギースペクトルの情報が保存されているため、それらを用いることにより、被覆管領域の Co-59、Co-60 の数密度の変動を計算することが出来る [16]。

計算には Burner クラスのメソッド「CobaltActivationCalculation(real den_init, real flux_factor)」を用いる。ここで、ひとつめの引数「den_init」には、Co-59 の数密度初期値を指定する。単位は「$10^{24}/\mathrm{cm}^3$」である。また、ふたつめの引数「flux_factor」には、中性子束レベルに乗じる因子を指定する（デフォルトは 1.0 なので、普通は無視して構わない）。これは、例えば、燃料集合体の上部、下部構造材を想定した計算を便宜的にする場合に、中性子束レベルを調整するために用いるものである [17]。

なお、このメソッドは燃焼中の中性子束情報を用いるため、一度、Burner クラスのインスタンスに燃焼計算をさせたあとに行わなければならないことに注意が必要である。

SUS に含まれる Fe-54 は (n,p) 反応により放射性核種である Mn-54 となる（Mn-54 の半減期は約 312 日）。Burner では、Co-59 と同様に Fe-54 の放射化計算を行うことが出来る。

この計算のためには Fe-54 の (n,p) 反応の多群断面積データが必要となるが、Burner では JENDL-4.0 評価値を多群に処理したものを内蔵している。対応するメソッドは「Mn54ActivationCalculation(real den_init, real flux_factor);」である。引数の意味は「CobaltActivationCalculation」と同一であるが、ここで「den_init」としてゼロを指定した場合には、被覆管に含まれている Fe-54 の数密度がそのまま用いられる。

[16] 中性子束のエネルギースペクトルの情報は Co-59 の (n,g) 反応の一群断面積を計算するために必要となる。このメソッドは簡単なプログラムであるので、ソースを眺めてみることを推奨する。

[17] ただし、このような計算を行う場合は、中性子束のエネルギースペクトルを被覆管領域のものと同一と仮定していることになる。

E 燃焼感度の計算

E.1 燃焼後数密度に対する核データの感度の計算

重核種の核分裂反応により FP は蓄積していく。核分裂量に比例して数密度が増加していく FP がある一方、核分裂により生成した FP が中性子捕獲、崩壊などいくつかの反応を経て異なる FP となって蓄積する場合もある。当然のことながら、個々の FP はその半減期に従って崩壊もする。FP には上記のように複雑なメカニズムを経て生成するものが多数あり、そのメカニズムを詳細に知ることは非常に重要である（言わずもがなであるが、アクチニドの数密度も同様である）。その際に大きな威力を発揮するのが燃焼後数密度の核データに対する感度係数である。

核種 i の燃焼後数密度を N^i_{EOC} とし、核データを p（各種反応断面積、半減期、核分裂収率などが対応する）とした場合、燃焼後数密度の感度係数 S は以下のように定義される。

$$S = \frac{dN^i_{EOC}/dp}{N^i_{EOC}/p} = \frac{dN^i_{EOC}/N^i_{EOC}}{dp/p} \tag{5}$$

つまり S は、核データ p が倍になったときの N^i_{EOC} の相対変動率と言い換えることが出来る。すなわち、S が 1.0 の場合は、核データ p が倍になった場合に N^i_{EOC} も同様に倍になることになる。

燃焼後数密度の感度は、例えば着目した核データ p_j を微小に変動させ、その際の N^i_{EOC} の変動を計算することで、通常の燃焼計算コードで得ることができる。しかし、核データが例えば 100 種あった場合には、燃焼計算を 101 回（1 回は基準の計算、100 回は 100 個の核データをそれぞれ微小変動させた計算）行なわなければならず、莫大な計算量が必要となる。この問題を避けるため、**一般化摂動論**と呼ばれる方法が開発され、それに基づいて少ない計算負荷で燃焼感度係数を計算することができる。ここではその詳細は省くが、この理論を適用した例が論文「G.Chiba, et al.,'Sensitivity analysis of fission product concentrations for light water reactor burned fuel,' *J. Nucl. Sci. Technol.,* **47**[7], p.652 (2010)」に記載されている。

CBG/Burner で燃焼後数密度の感度を計算させる方法は非常に簡単で、`Calculation` メソッドのかわりに `SensitivityCalculation` メソッドを用いればよい。以下にその例を示す。`SensitivityCalculation` メソッドの引数の二つ目は燃焼後数密度の感度を計算する核種の数、三つ目はその核種名のリストである。

Listing 20: 燃焼感度の計算例（`main.burn_sns.cxx`）

```
1  // +++ Burnup history data +++
2  bn.PutBurnStep(22);
3  real power_density_list[]={
4    179., 179., 179., 179., 179.,
5    179., 179., 179., 179., 179.,
6    179., 179., 179., 179., 179.,
7    179., 179., 179., 179., 179.,
8    179., 179.,
9  }; // [W/cm]
10 real burn_time[]={
11    0.1, 1.0, 2.5, 5.0, 7.5,  10., 12.5, 15., 17.5, 20.,
12    22.5, 25., 27.5, 30., 32.5,  35., 37.5, 40., 42.5, 44.,
13    44.9, 45., // GWd/t
14 };
15 bn.PutPowerDensityList(power_density_list);
16 bn.PutBurnTime(burn_time,true,true); // [GWd_t/day][accumulate/not]
17
18 string target[]={"Gd155","Cm242"};
19 bn.SensitivityCalculation(bu,2,target);
```

ひとつ注意しなければならないのは、短半減期核種の燃焼後数密度の感度を計算する場合には、燃焼末期の燃焼ステップを燃焼初期と同様に細かくとる必要がある点である [18]。この例でも、45GWd/t の燃焼期間のうち、44、44.9GWd/t で燃焼ステップを分割していることが分かるであろう。なお、燃焼のあとに冷却期間を設定する場合は、同様に冷却期間の末期を詳細に分割する必要がある。例えば、4 年冷却後の数密度に対する感度を計算する場合は、4 年 1 ステップの分割とするのではなく、4 年から 16 日を引いた日数（3 年と 349 日）、10 日、5 日、1 日というように冷却期間を 4 分割するようなことを行わなければならない。

[18] 燃焼感度の計算で必要となる随伴数密度の時間変化が燃焼末期で極めて大きい場合があることがその理由である。

また、燃焼感度の計算は、高い精度で行う必要がないので、等方反射境界条件（White boundary condition）を用いて計算時間を短縮させることを推奨する（境界条件の設定方法については 3.4 節の脚注を参照のこと）。

計算された燃焼後数密度の感度は CBZ の SensitivityData クラスのインスタンスとしてメソッド内部で定義され、そのクラスが有する WriteFile メソッドによりファイルに書き出される。ファイル名は sns. に感度を計算した核種名が追加されたものとなる（例えば Gd-155 の数密度の感度は sns.Gd155 となる）。この感度の情報を見るためには、SensitivityData クラスの ShowSelf メソッドを用いる（この詳細については、千葉のホームページの項目「CBZ（マニュアル）」の「CBZ/SensitivityData の使用マニュアル」を参照のこと）。例を以下に示す。

Listing 21: 燃焼後数密度の感度ファイルの出力例（main.burn_sns_read.cxx）

```
1  SensitivityData sns;
2  //sns.ReadFile("./","sns.Cm242");
3  sns.ReadFile("./","sns.Gd155");
4  sns.ShowSelf(0.001);
```

ここでは、まず SensitivityData クラスのインスタンス sns を生成して、これにファイルのデータを読ませ（ReadFile メソッド）、データを出力させている（ShowSelf メソッド）。ShowSelf メソッドの引数は、表示する感度の（絶対値に対する）下限値を示しており、この例では感度の絶対値が 0.001 以上となるもののみを出力させている。捕獲断面積、核分裂断面積といった群断面積データに対する感度は同様にエネルギー群に依存したデータとなるが、ShowSelf メソッドでは、全ての群の総和（すなわち 1 群断面積に対する感度）が出力される。

なお、計算される感度係数ファイルに sns.XXX とは異なる名前をつけることも可能である。その例を以下に示す。SensitivityCalculation メソッドの 5 番目の引数に指定する文字列がファイル名に対応し、この例では sns_uo2_XXX なるファイルが生成されることになる。4 つ目の引数は随伴数密度データを出力するためのオプションであり false とすればよい。

Listing 22: 燃焼感度の計算例 (2)（main.burn_sns.cxx）

```
1  bn.SensitivityCalculation(bu,2,target,false,"sns_uo2_");
```

E.2 燃焼後の無限増倍率に対する核データの感度の計算

燃焼後の核特性に対する感度も一般化摂動論を応用することで容易に計算が可能である。ここでは燃焼後無限増倍率に対する核データの感度の計算方法について解説する。なお、一般核特性に対する感度の計算方法等については「G.Chiba, et al., 'Uncertainty quantification of neutronic parameters of light water reactor fuel cells with JENDL-4.0 covariance data,' *J. Nucl. Sci. Technol.,* **50**[7], p.751 (2013)」に記載されている。

以下に、燃焼後の無限増倍率に対する核データの感度の計算例を示す。

Listing 23: 燃焼後の無限増倍率の感度の計算例

```
1  bn.SensitivityCalculationKeffEOC(bu);
```

Burner クラスの SensitivityCalculationKeffEOC メソッドにより燃焼後の無限増倍率に対する感度がファイル名 sns.k_EOC として計算される。また、燃焼効果を考慮しない感度と、燃焼効果に由来する感度[19]もそれぞれ異なるファイルで出力される（それぞれのファイル名は sns.k_EOC_dir、sns.k_EOC_indir）。

[19]前者は、無限増倍率を計算する時点での数密度が不確かではないとしたときの感度で**直接感度**とも呼ばれる。一方、後者は無限増倍率を計算する時点での数密度を介した感度で**燃焼間接感度**とも呼ばれる。巨視的断面積に関して、微視的断面積の項を介した影響が直接感度、数密度の項を介した影響が燃焼間接感度と考えてもよいだろう。

E.3 燃焼後数密度に対する核データの感度の計算（直接法）

本章でこれまでに述べた感度計算は燃焼核特性に対する一般化摂動理論に基づいて行うが、計算された感度の妥当性を確認するため、直接断面積を変動させて計算する感度と比較する場合がある。このような数値微分に基づく感度の計算を**直接法**と呼称したりするが、Burner には直接法に基づいて感度を計算するメソッド SensitivityCalculationDirect が実装されている。使用例を以下に示す。

Listing 24: 直接法による燃焼後数密度の感度の計算例

```
1  int target_num=4;
2  string target[]={"Pu239","Cm242","Cm245","Gd156"};
3  bn.SensitivityCalculationDirect(bu,target_num,target,942390,18,"sns.dir.p9f");
4  bn.PutFuelData(nuc0,mat0,den0,temp0);
5  bn.SensitivityCalculationDirect(bu,target_num,target,922350,18,"sns.dir.u5f");
```

このメソッドでは、数密度感度を計算する対象の核種の情報を２つ目と３つ目の引数で指定する。また、４つ目と５つ目の引数ではそれぞれ、感度を計算する核種 ID と断面積の種類 ID を指定する。現時点では、核分裂反応（18）と捕獲反応（102）のみが利用可能である。６つ目の引数では計算される感度のファイル名を指定する。上の例における３行目の感度計算では sns.dir.p9f と指定しているため、例えば Pu-239 燃焼後数密度に対する感度は sns.dir.p9f.Pu239 という名前のファイルで書き出される。また、このメソッドを連続して行う場合には、一度、燃料領域の核種数密度情報をリセットする必要があるので、この例の 4 行目のように PutFuelData メソッドにより燃料領域の初期組成情報を与える必要がある。

E.4 崩壊熱に対する核データの感度の計算

サンプルプログラム main.decayheat_sns.cxx を参照のこと。

F 燃焼行列の取り出し

Burner の Calculation メソッドにより燃焼計算を行う際、各燃焼ステップの一番始めのサブステップの燃焼行列を外部ファイルに出力することが可能である。これを行うためには、Burner クラスのインスタンスに対して予め「SetMatrixExtraction」メソッドを実行する必要がある。これにより、各燃焼ステップの燃焼行列からなる「matdat_stepXX」という名前のファイルが生成される。なお、SetMatrixExtraction メソッドの引数に string 型変数を与えてやることによって、ファイル名の「matdat」の部分を変更することが可能となる。また、二番目の引数では燃焼行列の各要素を出力する際の詳細度（precision）を指定できる（デフォルトは 16）以下に例を示すが、この場合には、各要素が小数点以下 10 桁で記述される燃焼行列のデータが「matrix_stepXX」という名前のファイルが生成されることになる。

Listing 25: 燃焼行列の外部ファイルへの出力例

```
1  bn.SetMatrixExtraction("matrix",10);
2  bn.Calculation(bu);
```

燃焼行列を格納するファイルの中身であるが、まずは行列のサイズ（すなわち核種数）が最初の行に与えられる。二行目以降は、含まれる核種の ENDF-ID と名前が核種数だけ交互に出力される。この核種の並びが、以後に出力される燃焼行列と対応する。その後、核種数 × 核種数のサイズの燃焼行列が出力される。燃焼行列を A_{ij} とした場合、はじめに $i = 1$ に対して $j = 1, 2, 3, ..$ というように与えられ、その後、$i = 2$ について同様に与えられる、というような並びとなっている（行列の行毎にデータが与えられる）。

G 均質 · 縮約断面積データのファイル出力

マクロ燃焼モデルでの使用を想定し、ピンセル全体を均質化し、かつエネルギー群を縮約した媒質データを、燃焼ステップ毎にファイル出力するメソッドが利用可能である。なお、現時点では XMAS-172 群構造のライブラリを使用していることを想定し、かつ出力されるものは 9 群構造に縮約されたものとなっている。

この機能を使用する場合は、`Calculation` メソッドを実行する前に、`GroupCollapsingDuringBurnup` メソッドを行えばよい。以下に使用例を示す。引数の一つ目が均質化かつ縮約された媒質データのテキストファイルを出力するディレクトリに、二つ目がファイル名に対応する。テキストファイルは計算する燃焼ステップ毎に出力され、引数で指定したファイル名に「`_stXX`」が付加される（`XX` は 0 から始まるステップ番号である）。以下の例では、`./MED_BN_DIR` ディレクトリに、`case1_stXX` という名前のファイルが燃焼ステップ数だけ作成されることになる。

Listing 26: 直接法による燃焼後数密度の感度の計算例

```
1  bn.GroupCollapsingDuringBurnup("MED_BN_DIR","case1");
```