高速炉核設計ツールCBZ/FRBurnerの使用マニュアル

千葉豪

目 次

1 FRBurner の概要

FRBurner は高速中性子炉の炉心燃焼計算を行なうためのモジュールである。当初は二次元円筒体系にモデル化するものとしていたため FRBurnerRZ と呼称していたが、将来的に三次元体系への拡張を行う予定であることから FRBurner と呼ぶこととした。

現状では、三次元で構築される原子炉炉心は二次元円筒にモデル化する。三次元体系の二次元円筒体系へのモデル化の概念を **Fig. 1** に示す（詳細は Appendix を参照のこと）。

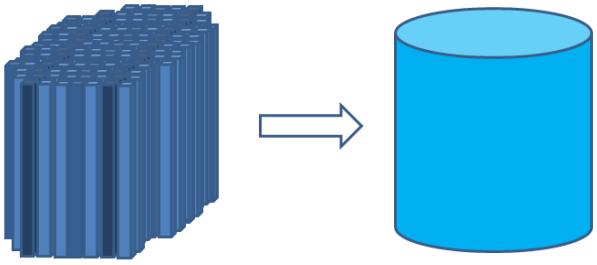

Fig. 1: 三次元体系の二次元円筒へのモデル化

高速増殖炉では一般に、ある一定期間運転した後、運転を止めて、一部の炉心燃料集合体、ブランケット集合体を取り出し、新しいものに交換する、ということを繰り返す。運転する期間を「サイクル」と呼ぶ。炉心燃料集合体、ブランケット集合体は、決められたサイクルだけ利用し、その後、取り出す。例えば、全ての集合体を 5 サイクル利用するとするならば、毎サイクル終了時には、炉心中の 1/5 の集合体が取り出されることになる。このような場合、「燃料交換バッチ数が 5 である」という言い方がされる[1]。

各領域の実効断面積は多群の CBZLIB を用いて計算され、そのエネルギー群の巨視的断面積をそのまま炉心内の中性子束分布計算に用いる。中性子束分布は、中性子拡散方程式を数値的に解くことによって得られ、CBZ の輸送ソルバー SNRZ や拡散ソルバー PLOS を用いる（デフォルトでは後者となる）。

核燃料の燃焼は、燃焼計算モジュール Burnup を内部で利用する。用いる燃焼チェーンはユーザが指定するが、本テキストの例では、Burnup クラスに内蔵されているデフォルトチェーンを利用している[2]。

2 入力データの作成

このマニュアルでは、サンプル入力 main.rz.halfcore.cxx（もしくは main_frburner_sample.cxx）を例に解説する。この入力は、高速増殖原型炉の簡易モデルを計算するためのものである。

2.1 燃料ペレットの組成データの作成

高速炉の燃料ペレットとしては、一般的にウランとプルトニウムの混合酸化物燃料（MOX 燃料）が使われている。ペレットの組成データ（どのような原子核で構成されているか、各々の原子核の数密度はどの程度か、といったデータ）は FRDTFuelComposition クラスで定義する。以下に、FRDTFuelComposition クラスのインスタンスの作成例を示す。

Listing 1: FRDTFuelComposition クラスのインスタンス作成例

```
1  // +++ Fuel composition +++
```

[1] このような燃料交換を二次元円筒炉心では厳密にモデル化することは出来ない。従って、FRBurner では、二次元円筒炉心の各領域について、仮想的なバッチ毎の核種数密度を評価しておき、燃焼計算はバッチ毎の数密度について行なわれる。また、炉心の中性子束分布計算では、各領域について、全てのバッチで平均した数密度を用いて計算した巨視的断面積を用いている。

[2] 重核種としては 17 核種（U-235、-236、-238、Np-237、Pu-238、-239、-240、-241、Am-241、-242m、-243、Cm-242、-243、-244、-245、-246）を扱い、FP は擬似核種として扱っている。重核種のチェーン例については、後述の「練習問題」に示されている。

```
2   MATIDTranslator midt;
3
4   // [Inner core]
5   FRDTFuelComposition fc_ic;
6   fc_ic.PutPuFissileEnrichment(16.1); // [wt%]
7   //fc_ic.PutPuEnrichment(20.); // [wt%]
8   fc_ic.PutU5Enrichment(0.2); // [wt%]
9   fc_ic.PutOM(1.97);
10  fc_ic.PutPelletTheoreticalDensity(85.); // %
11  string matnum[]={"Pu238","Pu239","Pu240","Pu241","Pu242","Am241"};
12  real fc_ic_inp[]={0., 58., 24., 14., 4., 0.};
13  fc_ic.PutTRUComposition(6,matnum,fc_ic_inp,midt);
14  /*
15  int matid[]={942380,942390,942400,942410,942420,952410};
16  fc_ic.PutTRUComposition(6,matid,fc_ic_inp);
17  */
```

FRBurner では燃料ペレットとして UO_2 と PuO_2 の混合燃料を想定しているが、Pu に Np や Am といった MA を添加することも可能である。

6 行目の PutPuFissileEnrichment メソッドでは、（初装荷）燃料に含まれる核分裂性 Pu（Pu-239 と Pu-241 の和）の、全アクチニドに占める重量比（核分裂性 Pu 富化度）をパーセント単位で指定する。なお、核分裂性 Pu ではなく、全ての Pu 同位体の総和の、全アクチニドに占める重量比（Pu 富化度）を指定する場合には、PutPuEnrichment メソッドを用いる。

8 行目の PutU5Enrichment メソッドでは、UO_2 中のウランに占める U-235 の重量比をパーセント単位で指定する（なお、UO_2 に含まれるウラン同位体は U-235 と U-238 としている）。

9 行目の PutOM メソッドでは、酸素のアクチニド核種に対する数密度比を指定する[3]。

10 行目の PutPelletTheoreticalDensity メソッドでは、ペレット中の酸化物燃料の体積理論密度（ペレット全体積中に燃料が占めている割合）を指定する。この例ではペレット全体の 85%を酸化物燃料が占め、それ以外の 15%は微細な空隙が占めていることになる。

最後に、13 行目の PutTRUComposition メソッドでは、PuO_2 燃料に含まれる超ウラン元素（TRU 元素）の重量比を指定する。2 つ目の引数では核種名（文字列）の配列（matnum）を指定し、その並び順の重量比を 3 つ目の引数で指定しているが、このように核種名を文字列で与える場合には 4 つ目の引数に核種名から ID に変換する MATIDTranslator クラスのインスタンスを指定しなければならない。なお、2 つ目の引数で直接核種 ID を指定する場合には 4 つ目の引数は不要となる（16 行目）。

高速炉では、炉心燃料領域からの中性子の漏洩を抑制しつつ、U-238 からの Pu-239 の転換を行なう「ブランケット」も、炉心燃料と並んで重要である。FRDTFuelComposition クラスによるブランケット燃料のペレットのデータ作成例を以下に示す。

Listing 2: FRDTFuelComposition クラスのインスタンス作成例（ブランケット燃料）

```
1   // [Blanket]
2   FRDTFuelComposition fc_b;
3   fc_b.PutPuFissileEnrichment(0);
4   fc_b.PutU5Enrichment(0.2);
5   fc_b.PutOM(2.01);
6   fc_b.PutPelletTheoreticalDensity(92.79);
7   fc_b.PutTRUComposition(6,matnum,fc_ic_inp,midt);
```

ブランケット燃料は UO_2 のみで構成されることから、核分裂性 Pu 富化度は 0 と設定されている（3 行目）。また、PuO_2 を全く含まない燃料であっても、手続き上、PutTRUComposition メソッドを実行しておく必要がある。

金属燃料を用いる場合には、FRDTFuelComposition クラスを継承した FRDTMetalFuelComposition クラスのインスタンスを活用すればよい。このクラスによる金属燃料のペレットのデータ作成例を以下に示す。このクラスのコンストラクタに引数として順に、%単位のスミア密度、%単位の Pu 富化度、wt%単位の Zr 含有率、TRU 同位体数、TRU 同位体核種 ID の配列、TRU 同位体の重量比の配列、AtomicMassData

[3]酸素のアクチニドに対する数密度比を「O/M 比」と呼ぶ。この値が 2.0 を超えると被覆管の酸化が始まるため、一般的に 2.0 より小さい値が設定される。なお、燃料が燃焼するとアクチニド核種の数密度が減少するため、O/M 比は増加することになる。

クラスのインスタンスを与えればよい。なお、AtomicMassData クラスのインスタンスに外部ファイルからデータを与える ReadFile メソッドの１つ目の引数は CBGLIB_BURN ディレクトリの相対パスを指定する。

Listing 3: 金属燃料のための FRDTMetalFuelComposition クラスのインスタンス作成例

```
1  AtomicMassData amd;
2  amd.ReadFile("../../","jeff311");
3  FRDTMetalFuelComposition fc_ic(70., pu_enrichment, 10.0, tru_nuc_nam, matnam, wgtin, amd);
4  FRDTMetalFuelComposition fc_oc(75., pu_enrichment,  6.0, tru_nuc_nam, matnam, wgtin, amd);
5  FRDTMetalFuelComposition fc_abic(70., 0., 10.0, tru_nuc_nam, matnam, wgtin, amd);
6  FRDTMetalFuelComposition fc_aboc(75., 0., 10.0, tru_nuc_nam, matnam, wgtin, amd);
```

ペレット領域の核種数密度は ShowSelf メソッドにより画面に出力させることが出来る。

2.2 被覆管の組成データの作成

中型高速増殖炉の燃料集合体の概念図を **Fig. 2** に示す。燃料ペレットは被覆管中に配置され、ペレットと被覆管で構成される 100 本を超える燃料ピンは「ラッパ管」と呼ばれる六角形状の管に配置される。なお、この図における最外周の六角形は、燃料集合体の外部境界に対応する。

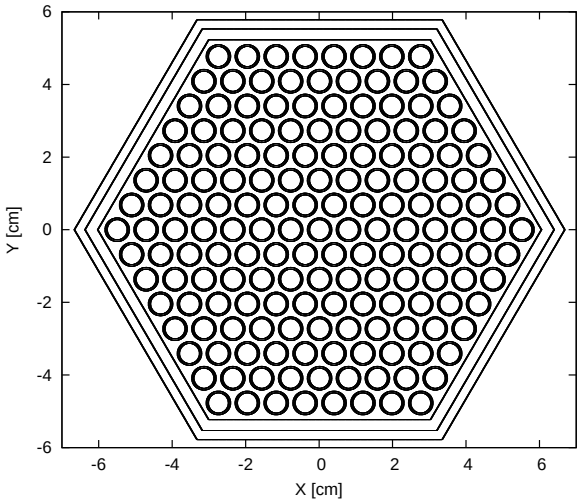

Fig. 2: 中型高速増殖炉の燃料集合体概念図

FRBurner では、被覆管、ラッパ管は同一の材質からなるものとする。これらのデータは FRDTSUSComposition クラスにより定義する。そのクラスのインスタンス作成例を以下に示す。

Listing 4: FRDTSUSComposition クラスのインスタンス作成例

```
1  // +++ SUS composition +++
2  FRDTSUSComposition sus316;
3  real sus_inp2[]={1.7, 13.5, 17., 2.5, 65.3, 0., 0.,}; // Mn Ni Cr Mo Fe W Co-59
4  sus316.PutDensityAndRatio(7.98,sus_inp2);
```

FRDTSUSComposition クラスでは、Mn、Ni、Cr、Mo、Fe、W、Co-59 の重量比を指定することで数密度データを作成する[4]。この例では SUS316 のデータを作成している。

2.3 燃料集合体の形状データの作成

燃料集合体の形状を定義するための FRDTSubAssemblyGeometry クラスのインスタンスの作成例を以下に示す。なお、サンプル入力では MonjuFuelSubAssembly メソッドでこれをソース内部で実行させている。

[4]Co-59 は、放射化の原因となる Co-60 のインベントリを計算する際に、その値を指定する。

Listing 5: FRDTSubAssemblyGeometry クラスのインスタンス作成例

```
1  // +++ Fuel subassembly geometry +++
2  FRDTSubAssemblyGeometry sa_geom;
3  sa_geom.PutAssemblyPitch(115.6);
4  sa_geom.PutDuctOutersize(110.6);
5  sa_geom.PutDuctThickness(3.);
6  sa_geom.PutPinOuterDiameter(6.5);
7  sa_geom.PutPinThickness(0.47);
8  sa_geom.PutNumberOfPin(169);
9  sa_geom.PutSpacerwireDiameter(1.32);
10 sa_geom.PutSpacerwirePitch(307.);
11 sa_geom.PutPelletOuterDiameter(5.4);
12 sa_geom.PutPelletInvoidDiameter(0.);
13 sa_geom.PutPinPitch(7.9);
14
15 //sa_geom.ShowVolumeInformation();
```

上のサンプルでは種々のパラメータの指定を行っているが（長さの単位は [mm]）、その定義を **Fig. 3** に示す。なお、ピンピッチは隣接する 2 つのピンの中心間距離であり、「もんじゅ」では約 7.9mm とされている。燃料集合体の最外周はラッパ管（ダクト）で囲まれるが、集合体間には冷却材が流れる隙間があるため、集合体ピッチ（「Assembly pitch」）とダクト面間距離（「Duct outer size」）は一致しないことに注意が必要である。また、燃料ペレットの中央部にボイド領域を設ける燃料（中空燃料）も扱うことが可能となっている。9、10 行目ではスペーサワイヤに関するデータを定義している。スペーサワイヤは隣接する燃料ピンが接触することを避けるため、各燃料ピンの外周に螺旋状に巻かれるワイヤであり、10 行目の SpacerwirePitch では巻き付けピッチ（螺旋状にピン外周を一周したときの軸長）を指定しているが、**現在の版では非均質モデルを用いるときには考慮することができないためゼロとしている**。

燃料ピン数は、燃料リングの層数を N とすると、$3N(N-1)+1$ として与えられる。「もんじゅ」の燃料集合体における燃料リングの層数は 8 であるので、燃料ピン数は 169 となる。燃料ピン数は PutNumberOfPin メソッドで与える。

なお、ダクト形状と燃料ピン数、ピンピッチの関係が物理的にありえない場合には、エラーメッセージが出力される。

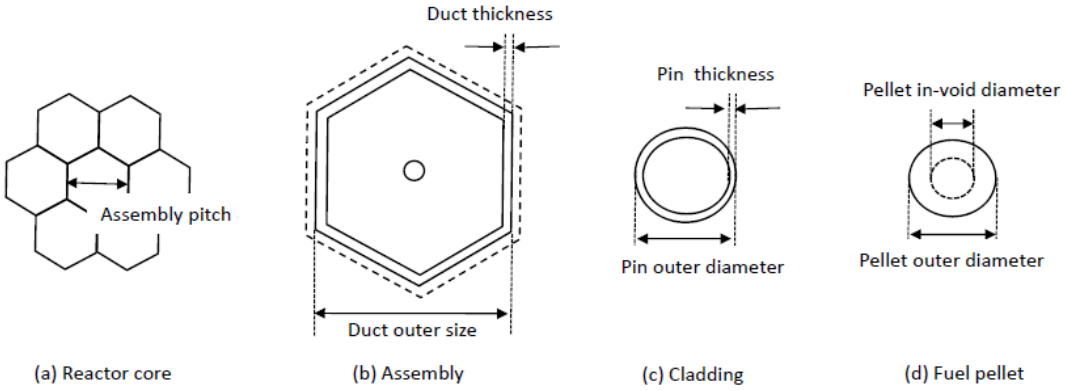

Fig. 3: FRDTSubAssemblyGeometory クラスで定義する種々の幾何形状パラメータ

サンプル入力例の 15 行目では ShowVolumeInformation メソッドがコメントアウトされている。このメソッドは燃料集合体の各領域の体積を表示するためのもので、「もんじゅ」の燃料集合体に対しては以下のような出力が与えられる。

Listing 6: FRDTSubAssemblyGeometry クラスの ShowVolumeInformation メソッドの使用例

```
1  ##############################################################################
2  # Region-wise volume of subassembly [mm^2] and fraction to total.
3  #
4  #   Total                    : 1.15730e+04
5  #   Fuel pellet              : 3.87047e+03 3.34440e-01
6  #   Void in pellet           : 0.00000e+00 0.00000e+00
7  #   Void at pellet-clad gap  : 2.32759e+02 2.01123e-02
8  #   Cladding                 : 1.50471e+03 1.30019e-01
```

```
#   Spacer wire                : 2.31784e+02 2.00280e-02
#   Wrapper tube               : 1.11821e+03 9.66224e-02
#   Coolant                    : 4.61507e+03 3.98779e-01
#   (inside of wrapper tube) : 3.63560e+03
#################################################################################
```

2.4 燃料集合体データの作成

これまでに説明した、燃料集合体の形状データ、燃料ペレットの組成データ、被覆管とラッパ管の組成データから、燃料集合体のデータを定義する FRDTSubAssembly クラスのインスタンスを定義する。インスタンスの作成例を以下に示す。

Listing 7: FRDTSubAssembly クラスのインスタンス作成例

```
1  // +++ Fuel subassembly +++
2  FRDTSubAssembly sa_ic;
3  sa_ic.PutSubAssemblyGeometry(&sa_geom);
4  sa_ic.PutFuelComposition(&fc_ic);
5  sa_ic.PutSUSComposition(&sus316);
6  sa_ic.PutSodiumDensity(0.838008);
7  sa_ic.PutTemperature(1097.5+273.15, 472.5+273.15); // [K] non-sodium/sodium
```

3 から 5 行目において、形状データ、燃料ペレットおよび構造材（被覆管とラッパ管）の組成データのインスタンスを渡している。6 行目では、冷却材流路を流れるナトリウムの密度（単位は [g/cm^3]）を、7 行目では燃料集合体に含まれる冷却材（ナトリウム）核種を除いた全ての原子核の温度と冷却材核種の温度（単位は [K]）を、それぞれ指定している[5]。

2.5 制御棒データの作成

制御棒については、基本的には燃料集合体と同様の方法で種々のデータを作成する。サンプル main.rz.halfcore.cxx の計算では制御棒は考慮していないので、ここは読み飛ばしてもらって構わない。

パッケージでは main.cr.cxx というファイルが対応しており、これに基づいて説明を行う。

まずは、吸収体（B_4C）ペレットの組成データの作成例を以下に示す。一つ目の引数が B-10 濃縮度に、二つ目の引数がペレットの理論密度に、それぞれ対応する。この例では、2 行目が調整棒、3 行目が後備炉停止棒のデータとなっている。

Listing 8: 制御棒データの作成例（FRDTB4CComposition クラスのインスタンス作成例）

```
1  // +++ B4C composition +++
2  FRDTB4CComposition b4c_cr(0.39, 0.951632); // B-10 ratio, theoretical density
3  FRDTB4CComposition b4c_bcr(0.90, 0.9506);  // B-10 ratio, theoretical density
```

次に、制御棒の幾何形状データの作成例を以下に示す。燃料集合体と同様のいくつかの項目に値を入力することに加えて、制御棒特有の「案内管（guide tube）」、「保護管（shield tube）」の形状データを定義する必要がある。

Listing 9: 制御棒データの作成例（FRDTControlRodGeometry クラスのインスタンス作成例）

```
1  FRDTControlRodGeometry cr_geom;
2  cr_geom.PutAssemblyPitch(115.6);
3  cr_geom.PutPinOuterDiameter(16.9);
4  cr_geom.PutPinThickness(2.0);
5  cr_geom.PutNumberOfPin(19);
6  cr_geom.PutSpacerwireDiameter(1.2);
7  cr_geom.PutSpacerwirePitch(239.);
8  cr_geom.PutPelletOuterDiameter(12.2);
9  cr_geom.PutPelletInvoidDiameter(0.);
10 // The following are specific for control rod
11 cr_geom.PutGuidetubeOuterDiameter(110.6);
12 cr_geom.PutGuidetubeThickness(3.);
13 cr_geom.PutShieldtubeOuterDiameter(94.);
14 cr_geom.PutShieldtubeThickness(2.);
```

[5]通常、燃料ペレット、被覆管、冷却材の順に温度は低下していくため、それぞれの構造物に含まれる原子核について異なる温度を設定する必要があるが、FRBurner では計算の簡略化のため、冷却材に含まれる核種とそれ以外の核種とで温度設定を分けている。

「もんじゅ」に用いられている制御棒の幾何形状を **Fig. 4** に示す。19 本の吸収ピンが保護管内に収納されており、吸収ピンと保護管が一体となって案内管内部を上下するという構造となっている。制御棒が抜けた状態（フォロワ）は、案内管のみが存在する状態に対応する。加えて、制御棒の吸収ピンの構造について

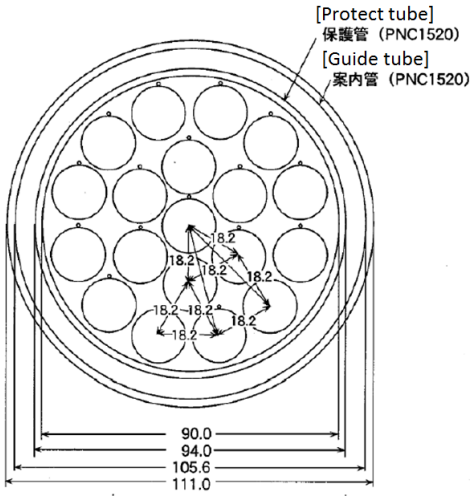

Fig. 4: 「もんじゅ」制御棒の形状

Fig. 5 に示す。調整棒と後備炉停止棒とでは構造が異なっていることに注意が必要である。

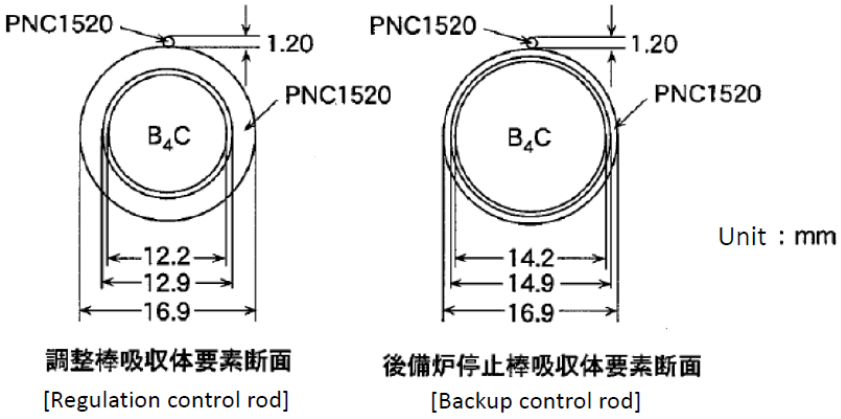

Fig. 5: 「もんじゅ」制御棒の吸収ピンの構造

最後に、制御棒に対応するクラス `FRDTControlRod` のインスタンスの作成例と、それを用いた `Medium` クラスのインスタンスへの数密度情報追加の例を以下に示す。

Listing 10: 制御棒データの作成例（`FRDTControlRod` クラスのインスタンス作成例）

```
1   FRDTControlRod cr;
2   cr.PutControlRodGeometry(&cr_geom);
3   cr.PutB4CComposition(&b4c_cr);
4   cr.PutSUSComposition(&sus316);
5   cr.PutSodiumDensity(0.856195); // 400oC
6
7   Medium med_cr;
8
9   med_cr.PutImax(group);
10  med_cr.PutPL(1);
11
12  cr.PutHomogenizedNumberDensity(med_cr);
13
14  med_cr.PutTemperatureForAllNuclide(25.+273.15);
```

2.6 燃料、ブランケット、制御棒以外の媒質データの作成

中性子反射体など、上記以外の媒質データについては、媒質に含まれる原子核とその数密度、及び原子核の温度の情報を直接 FRBurner クラスのインスタンスに渡すことになる。これについては後述する。

2.7 炉心形状データの作成

二次元円筒炉心の形状、空間メッシュ分割、物質（媒質）配置は `CartMeshInfo` クラスのインスタンスにより定義する。以下にその例を示す。なお、`CartMeshInfo` クラスについては、CBZ のチュートリアル応用編に記載されているのでそちらを参照のこと。

Listing 11: `CartMeshInfo` クラスによる炉心形状データの作成例

```
// +++ CartMeshInfo
CartMeshInfo cmi;
real xl[]={
  6.06943,  9.98878, 10.39784,  2.32216,  3.61336,
  4.52736, 10.48487,  7.73198,  3.87278,  9.39049,
 10.50360, 10.50570, 10.50715, 10.50818,  9.59162,
 30.0
};
real yl[]={
  15., 15., 16.5, 15., 20.,   26.5
};
int xm[]={1,2,2,1,1, 1,2,2,1,2, 2,2,2,2,2, 6};
int ym[]={3,3,3,3,4, 3};
int mat[]={
  40, 0, 3, 6,40, 6, 9,12,40,15,18,21,24,29,31,39,
  40, 1, 4, 7,40, 7,10,13,40,16,19,22,25,29,31,39,
  40, 2, 5, 8,40, 8,11,14,40,17,20,23,26,29,31,39,
  40,32,32,32,40,32,33,33,40,33,35,35,27,30,31,39,
  40,34,34,34,40,34,34,34,40,34,36,36,28,30,31,39,
  40,37,37,37,40,37,37,37,40,37,37,37,38,38,38,39,
};
cmi.PutMeshInfo(16,6,xm,ym,xl,yl,mat);
cmi.PutBoundaryCondition
("Reflective","Vacuum","Reflective","Vacuum");
```

媒質の配置は配列 `mat` で定義されており、この計算例では計 41 種の媒質データが定義されることが分かる。これらの媒質はそれぞれが異なる媒質として扱われ、燃焼計算もそれぞれの媒質で行われることになる。

高速炉の炉心は、一般的に、いくつかの領域に区別することが出来、これらの領域に装荷される燃料は同一のものとなる[6]。`FRBurner` ではこの領域を「Material zone」として定義している。Material zone は上述の `CartMeshInfo` クラスのインスタンスを定義する際に設定した径方向と軸方向分割のもとで定義される（上記の例ではそれぞれが 16、6 に分割されている）。従って、Material zone を定義するための情報（配列）の大きさは、`CartMeshInfo` クラスのインスタンスに与える媒質の配置情報と同一となる。Material zone を定義する場合には、`CartMeshInfo` クラスのインスタンスに渡した配列データ `mat` をコピーし、それに Material zone の情報を上書きすることで対応すればよいであろう。なお、核燃料が含まれない（燃焼計算の対象としない）媒質に対しては Material zone を定義する必要がないため、Material zone に使われる ID 以外の数を指定する必要がある。基本的に、核燃料が含まれない媒質の ID は Material zone の最大 ID よりも大きくなることから、媒質 ID をそのまま定義してやればよい。以下に Material zone を指定する配列の作成例を示す。

Listing 12: Material zone の定義例

```
// +++ Material zone definition +++
int materialzone[]={
  40, 0, 0, 0,40, 0, 0, 0,40, 0, 1, 1, 2, 2, 2,39,
  40, 0, 0, 0,40, 0, 0, 0,40, 0, 1, 1, 2, 2, 2,39,
  40, 0, 0, 0,40, 0, 0, 0,40, 0, 1, 1, 2, 2, 2,39,
  40, 3, 3, 3,40, 3, 3, 3,40, 3, 4, 4, 2, 2, 2,39,
  40, 3, 3, 3,40, 3, 3, 3,40, 3, 4, 4, 2, 2, 2,39,
  40,37,37,37,40,37,37,37,40,37,37,37,38,38,38,39,
};
```

[6] パッケージに与えられているサンプルでは、「内側炉心」「外側炉心」「径方向ブランケット」「内側炉心に対応する軸方向ブランケット」「外側炉心に対応する軸方向ブランケット」の 5 つの領域が定義されている。

2.8 FRBurner クラスのインスタンスの生成と各種データの引き渡し

はじめに、FRBurner クラスのインスタンスの作成例を示す。

Listing 13: FRBurner クラスのインスタンスの作成例

```
1  FRBurnerRZ frbn(group,fuel_nuc_num,matno);
2  frbn.PutNumberofMedium(41);
3
4  // (core geometry information)
5  frbn.PutCartMeshInfo(cmi);
```

コンストラクタでは、エネルギー群数、燃料媒質に含まれる核種数、これらの核種の Index のリストを引数として渡す。これらはデフォルトのデータと考えて良い。また、2 行目ではこのインスタンスで定義される媒質数を引数として渡している。5 行目では、炉心体系情報を示す CartCoreInfo クラスのインスタンスを渡している。

次に、Material zone に関する情報の FRBurner クラスのインスタンスへの渡し方の例を示す。

Listing 14: FRBurner クラスのインスタンスへの Material zone の情報の引き渡し例

```
1  // (material zone)
2  frbn.PutNumberMZone(5);
3  frbn.PutMZoneInfo(materialzone);
4  string name_mzone[]={"IC","OC","RB","ABIC","ABOC"};
5  frbn.PutNameMZone(name_mzone);
6  int batch_mzone[]={5,5,5,5,5};
7  frbn.PutBatchMZone(batch_mzone);
8  frbn.PutSADataToMZone(sa_ic,0,xslib);
9  frbn.PutSADataToMZone(sa_oc,1,xslib);
10 frbn.PutSADataToMZone(sa_rb,2,xslib);
11 frbn.PutSADataToMZone(sa_ab,3,xslib);
12 frbn.PutSADataToMZone(sa_ab,4,xslib);
```

2 行目の PutNumberMZone メソッドではこの体系で定義される Material zone の数を渡している。それに続く 3 行目では PutMZoneInfo メソッドにより、前述で説明した Material zone の配置情報（int 型の配列）を渡している。各 Material zone には名称を定義することが出来、4、5 行目ではそれを与えている。また、燃焼交換バッチ数も Material zone 毎に定義されるため、その情報を 6、7 行目で与えている[7] 。8 から 12 行目においては、PutSADataToMZone メソッドにより、二つ目の引数で指定された Material zone に、集合体の情報を有する FRDTSubAssembly クラスのインスタンスを一つ目の引数として渡している。

最後に、非燃料媒質、すなわち燃焼計算の対象としない媒質情報の FRBurner クラスのインスタンスへの渡し方の例を示す。

Listing 15: FRBurner クラスのインスタンスへの非燃料媒質の情報の引き渡し例

```
1  // +++ Non-fuel assembly composition information +++
2  // (axial shielding (core))
3  int mat_nf[]={110230,240000,280000,420000,260000,250550};
4  real den3[]={8.94364e-3, 3.87561e-3, 2.72666e-3, 3.08888e-4, 1.38604e-2, 3.66807e-4};
5  // (axial shielding (blanket))
6  real den30[]={7.60948e-3, 3.07270e-3, 2.16178e-3, 2.44896e-4, 1.09889e-2, 2.90816e-4};
7  // (radial shielding)
8  real den4[]={5.02191e-3, 1.33748e-2, 6.18204e-3, 1.20994e-4, 4.59545e-2, 1.32492e-3};
9  // (sodium follower)
10 real den5[]={2.04623e-2, 1.37677e-3, 9.68620e-4, 1.09730e-4, 4.92377e-3, 1.30305e-4};
11
12 // (non-fuel)
13 frbn.PutNonfuelInfo(37,6,mat_nf,den3,673.15);
14 frbn.PutNonfuelInfo(38,6,mat_nf,den30,673.15);
15 frbn.PutNonfuelInfo(39,6,mat_nf,den4,673.15);
16 frbn.PutNonfuelInfo(40,6,mat_nf,den5,673.15);
```

PutNonfuelInfo メソッドを用いて、一つ目の引数で指定した媒質 ID に対して、媒質に含まれる原子核の種類数、核種 ID、数密度データ、温度といった情報を引数として渡す。なお、非燃料媒質に対して FRDTSubAssembly クラスのインスタンスを用いて物質情報を定義することも可能であり、その場合は PutSADataToNonfuel メソッドを用いる。

[7] 交換バッチとは、燃料集合体が取り出されるまでに炉心に装荷されるサイクル数を示す。換言すると、各サイクル毎に、1/（交換バッチ数）の燃料集合体が炉心から取り出されることになる。

3 燃焼計算の実行と結果の取り出し

ある熱出力のもとで、決められた期間、原子炉を運転した場合、燃料ペレット中の核種の組成は時々刻々と変化し、それに伴い原子炉としての特性も変化していく。原子炉の運転に伴うこういった変化を計算することを**原子炉（炉心）の燃焼計算**と呼ぶ。FRBurner クラスは、二次元円筒体系に高速炉の炉心をモデル化して燃焼計算を行なう機能を持つ。ここでは、FRBurner クラスのインスタンス frbn を用いた計算例を示す。

3.1 燃焼計算条件の指定

燃焼計算条件の指定例を以下に示す。

Listing 16: 燃料計算条件の指定例

```
1  frbn.PutReactorPower(714000000*0.5); // [Wth]
2  // A factor of 0.5 is used because we are treating a half-core model.
3  frbn.PutTraceCycle(6);
4  frbn.PutCycleDiv(2);
5  frbn.PutCycleLength(148); // day
6  frbn.PutRefuelDay(0.); // days required for refueling
```

PutReactorPower メソッドでは原子炉熱出力を W 単位で設定する。なお、この計算例では、計算の対象としている原子炉について、軸方向の対称性を利用して炉心の半分のみを扱っているので、熱出力を全炉心のもの（714 MWth）の半分に設定している。

PutTraceCycle メソッドでは燃焼計算を行う全サイクル数を指定する。

PutCycleDiv メソッドでは、各燃焼サイクルのステップ分割数を指定する。この場合は 2 を指定しているので、サイクルは 2 分割される。従って、各燃焼サイクルにおいては、3 つの燃焼点（サイクル始点、中央点、終点）で、中性子束分布、固有値等の計算が行なわれる。

PutCycleLength メソッドでは、各燃焼サイクルのサイクル長を日単位で指定する。なお、FRBurner では各燃焼サイクルのサイクル長は同一となる。

PutRefuelDay メソッドでは、各燃焼サイクル間に（燃料集合体の取り替えのため）経過する日数を指定する。

3.2 計算実行前の情報の取り出し

燃焼計算の実行前に、以下のメソッドによりいくつかの情報を取り出すことができる。計算実行前の情報の取り出しのためのメソッドの一覧を以下に示す。

Listing 17: 計算実行前の情報の取り出しのためのメソッド一覧

```
1  frbn.PrintMacroXS();
2  frbn.PrintInitialND();
```

- PrintMacroXS：初装荷燃料の巨視的断面積を、内側炉心、外側炉心、径ブランケット、軸ブランケットについて出力する。
- PrintInitialND：初装荷燃料の核種数密度を、内側炉心、外側炉心、径ブランケット、軸ブランケットについて出力する。

3.3 計算の実行と標準出力

計算の実行は、FRBurner クラスに実装されている Run メソッドで行なう。このメソッドには 2 つの引数が必要とされており、一つ目が計算に使用する核データライブラリ（XSLibrary クラスのインスタンス）、二つ目が計算に使用する燃焼計算モジュール（Burnup クラスのインスタンス）に対応する。

Run メソッドを実行すると、画面に以下のような出力が行なわれる。

Listing 18: Run メソッドの出力例

```
1  ################################################################################
2  #C Step   Day   Keff    Max. line    C.R.   Material zone-wise
3  #y                     power[W/cm]          Power Dist. [%]
4  #                      (pos:r,z)
5  #                                            (0)   (1)   (2)   (3)   (4)
6    0    0     0 1.09190 344.9 (1,0) 1.095 56.5  38.3  3.0   1.3   0.6
7    0    1    74 1.07501 333.7 (1,0) 1.110 55.7  38.1  3.5   1.7   0.8
8    0    2   148 1.05891 323.4 (1,0) 1.125 54.8  38.0  4.0   2.1   0.9
9    1    0   148 1.06558 327.8 (1,0) 1.119 55.2  38.0  3.8   1.9   0.8
10   1    1   222 1.04991 318.0 (1,0) 1.133 54.4  37.8  4.4   2.3   0.9
11   1    2   296 1.03501 309.0 (1,0) 1.147 53.6  37.6  4.9   2.6   1.1
12   2    0   296 1.04774 317.2 (1,0) 1.134 54.2  37.8  4.5   2.3   1.0
13   2    1   370 1.03292 308.2 (1,0) 1.148 53.4  37.5  5.0   2.7   1.1
14   2    2   444 1.01884 299.9 (1,0) 1.161 52.7  37.3  5.6   3.0   1.2
15   3    0   444 1.03702 311.3 (1,0) 1.143 53.7  37.5  4.9   2.6   1.0
16   3    1   518 1.02269 302.7 (1,0) 1.156 52.9  37.3  5.5   3.0   1.2
17   3    2   592 1.00909 294.7 (1,0) 1.168 52.1  37.0  6.0   3.3   1.3
18   4    0   592 1.03212 308.9 (1,0) 1.146 53.4  37.4  5.1   2.7   1.1
19   4    1   666 1.01801 300.4 (1,0) 1.159 52.6  37.1  5.7   3.1   1.2
20   4    2   740 1.00462 292.6 (1,0) 1.171 51.9  36.8  6.3   3.4   1.3
21   5    0   740 1.03197 309.1 (1,0) 1.146 53.4  37.4  5.1   2.7   1.1
22   5    1   814 1.01787 300.6 (1,0) 1.159 52.6  37.1  5.7   3.1   1.2
23   5    2   888 1.00449 292.7 (1,0) 1.171 51.9  36.8  6.3   3.4   1.3
```

各行において、左から、燃焼サイクル、サイクル中の燃焼ステップ（燃焼サイクルともに 0 から始まることに注意）、経過日数、実効増倍率、最大線出力、転換比、出力分担率が示される。この例では、各燃焼サイクル毎に 3 点で実効増倍率等の計算が行なわれている（ステップ分割数が 2 に設定されているため）。また、燃焼サイクル長は 148 日と設定されている。

最大線出力には括弧内でふたつの整数値のペアが示されているが、これは、最大線出力を示した詳細メッシュの位置を示している。メッシュは「0」から定義されるため、「(1,0)」となっている場合には、R 方向については 2 つ目、Z 方向については 1 つ目のメッシュを示していることになる。

転換比は核分裂性核種の巨視的中性子吸収反応率に対する親核種の巨視的中性子捕獲反応率の比で定義される。親核種は Th-232、U-234、U-238、Pu-240、核分裂性核種は U-233、U-235、Pu-239、Pu-241 としている。また、Th-232 から U-233 が生成される過程における Pa-231 の中性子捕獲は U-233 の生成を小さくすることから、Pa-231 の巨視的中性子捕獲反応率を転換比の分子から除く操作をしている[8]。

それに引き続く実数値は Material zone 毎の出力分担率を%単位で示す。

3.4 計算オプションの指定

Run メソッドの実行前に、以下のメソッドにより計算オプションを指定することができる。計算オプション指定のためのメソッドの一覧を以下に示す。

Listing 19: 計算オプション指定のためのメソッド一覧

```
1  frbn.PrintLinepowerMap();
2  frbn.VoidCalculationOn();
3  frbn.DopplerCalculationOn();
4  frbn.ShowFissionInfoOn();
5  frbn.CMFDOff();
```

[8] ここで定義している転換比を数式で表すと、

$$C.R. = \frac{(N^{Th-232}\sigma_c^{Th-232} + N^{U-234}\sigma_c^{U-234} + N^{U-238}\sigma_c^{U-238} + N^{Pu-240}\sigma_c^{Pu-240} - N^{Pa-233}\sigma_c^{Pa-233})\phi}{(N^{U-233}\sigma_a^{U-233} + N^{U-235}\sigma_a^{U-235} + N^{Pu-239}\sigma_a^{Pu-239} + N^{Pu-241}\sigma_a^{Pu-241})\phi} \quad (1)$$

となる。ただし、$\sigma_a = \sigma_c + \sigma_f$ である。

- PrintLinePowerMap：Run メソッド実行時に、線出力の空間分布を標準出力に追加させる。さらに、線出力空間分布の時間推移をプロットするためのデータを linepower_map という名前の外部ファイルに出力させる（この使用方法については後述する）。
- VoidCalculationOn：燃焼計算中の各計算点において、**燃焼領域（媒質）について**冷却材であるナトリウムの数密度を 10^{-10}[1/cm/barn] に低下させた際の反応度の計算を行う。
- DopplerCalculationOn：燃焼計算中の各計算点において、**燃焼領域（媒質）に含まれる**重核種の温度を 1,500 K にした際の反応度の計算を行う。
- ShowFissionInfoOn：Run メソッド実行時に、燃焼計算中の各計算点において中性子増倍に関する情報を標準出力に追加させる（詳細は CBZ のチュートリアル（応用編）参照のこと）。
- CMFDOff：粗メッシュ有限差分加速法をオフにする。デフォルトではこの加速法を使用しているが、実効増倍率が 1.5 を越えるような体系の計算を行なう場合には計算が正常に実施されない場合があるのでこのメソッドを用いる。なお、FRBurner クラスの PreCalculation メソッドよりも前に行なう必要がある。

3.5 計算実行後の結果の取り出し

Run メソッドにより燃焼期間中の実効増倍率や最大線出力といった情報が標準出力として画面に表示されるが、それ以外の情報については、Run メソッド実行後に FRBurner クラスのインスタンスのメソッドにより取り出すことが出来る。計算実行後の結果の取り出しのためのメソッドの一覧を以下に示す。

Listing 20: 計算実行後の結果の取り出しのためのメソッド一覧

```
1  frbn.PrintND(bu,"nd");
2  frbn.PrintBurnup();
3  frbn.PrintFluxLevelHistory();
4  frbn.Print1groupXS("Pu239",0);
5  frbn.PrintNuclideWeightPerBatch(bu); // Discharged HM weights are printed.
6  frbn.PrintNuclideWeightPerBatch(bu,true); // Loaded HM weights are printed.
```

- PrintND：燃焼計算後の各媒質のバッチ毎のデータを出力する。一つ目の引数が Burnup クラスのインスタンスで、二つ目の引数が出力する情報を指定する文字列を示す。この例では二つ目の引数で nd が指定されているので、核種数密度を出力する[9]。なお、内側炉心、外側炉心、径ブランケット、軸ブランケットといった「マクロな」領域毎のデータも併せて出力される。
- PrintBurnup：各媒質の平均燃焼度とバッチ間の最大燃焼度を出力する。バッチ毎の燃焼度が知りたい場合は PrintND メソッドで取り出すことが出来る。
- PrintFluxLevelHistory：非燃料媒質を含む全ての媒質における平均中性子束レベルを、燃焼ステップ、サブステップ毎に出力する。
- Print1groupXS：一つ目の引数で指定した核種の微視的 1 群断面積[10] を、二つ目の引数で指定した ID の媒質（ただし燃料媒質のみ）について、燃焼ステップ、サブステップ毎に出力する。
- PrintNuclideWeightPerBatch：1 バッチあたりの原子炉からの取り出し燃料の核種毎、領域毎の重量を出力する。1 バッチあたりの原子炉への装荷燃料の重量を出力させる場合には、2 つ目の引数として true を指定すれば良い。このメソッドを利用することにより、平衡状態の原子炉における物量収支を計算することが出来る。

[9] FRBurner では、ある核分裂性核種の核分裂反応により生成する核分裂生成物核種は擬似的な単一の核種とする（**擬似ランプ化核分裂生成物核種**、**ランプ化 FP** とも呼ぶ）。FRBurner ではランプ化 FP として「FP.U235」「FP.U238」「FP.Pu239」「FP.Pu241」の 4 核種があり、U-235、U-236 が核分裂した場合には「FP.U235」が、U-238、Np-237、Pu-238 が核分裂した場合には「FP.U238」が、Pu-239、-240 が核分裂した場合には「FP.Pu239」が、Pu-241、-242、Am 同位体、Cm 同位体が核分裂した場合には「FP.Pu241」が生成されるようになっている。

[10] 中性子核反応断面積は入射中性子のエネルギーに依存する物理量であるが、中性子束のエネルギー分布を荷重として平均化するこ

3.6 計算実行後の任意の燃焼点における計算

燃焼計算実行後に、任意の燃焼サイクル、計算点の炉心状態に対して種々の計算を行なうことが可能である。この場合、まずは、炉内の全ての媒質に対して、指定した燃焼サイクル、計算点における核種数密度にデータを書き換える必要があるので、以下の例に示す PutNumberDensity メソッドを用いる。

Listing 21: 任意の燃焼時点の数密度に書き換えるためのメソッド使用例

```
1  frbn.PutNumberDensity(0,0);
```

このメソッドでは、一つ目の引数で燃焼サイクル、二つ目の引数で計算点を指定し、その時点の核種数密度に「時間を戻す」操作を行なう。

全媒質の核種数密度を指定した燃焼時点に戻したあと、以下に示すメソッドで、その燃焼時点に対して種々の計算を行い、情報を取り出すことが出来る。なお、PutNumberDensity メソッドを行なわないで以下のメソッドを実行した場合には、それらは燃焼計算終了時点の原子炉に対して計算を行なうことになる。

Listing 22: 任意の燃焼時点の数密度に書き換えたあとのメソッド使用例

```
1  frbn.CalNeutronFluxEnergySpectrum(); // forward fluxes for all mediums
2  frbn.CalNeutronFluxEnergySpectrum(true); // adjoint fluxes for all mediums
3  frbn.CalNeutronFluxEnergySpectrum(false,8); // forward flux for medium 8
4  frbn.CalNeutronFluxEnergySpectrum(false,0,15); // forward fluxes for mediums 0 to 15
5  frbn.CalVoidReactivity();
6  frbn.CalDopplerReactivity();
7  frbn.Cal1groupXS();
8  frbn.ShowNeutronMultiplicationInfo();
```

- CalNeutronFluxEnergySpectrum：各媒質における中性子束のエネルギースペクトルを表示する。一つ目の引数を true にすると、随伴中性子束を表示する（デフォルトは false）。二つ目の引数は（随伴）中性子束を表示させる媒質の ID を示す（これを省略した場合は、すべての媒質の中性子束が表示される）。三つ目の引数も指定した場合は、二つ目の引数の ID から三つ目の引数の ID の媒質について、（随伴）中性子束を表示する（4 行目の例では媒質 0 から媒質 15 までの中性子束が表示される）。

- CalVoidReactivity：**燃焼領域（媒質）において**冷却材であるナトリウムの数密度を 10^{-10}[1/cm/barn] に減少させた際の反応度を厳密摂動により計算する。一つ目、二つ目の引数を指定することによって、冷却材密度を減少させる媒質を指定することが出来る。その方法は CalNeutronFluxEnergySpectrum と同様である。例えば、CalVoidReactivity(0, 16); とした場合は、媒質 ID が 0 から 16 の領域について含まれるナトリウムの数密度を変化させたときの反応度が計算される。

- CalDopplerReactivity：**燃焼領域（媒質）において**重核種の温度を増加させた際の反応度を厳密摂動により計算する。一つ目、二つ目の引数を指定することによって、重核種の温度を増加させる媒質を指定することが出来る。その方法は CalNeutronFluxEnergySpectrum と同様である。

- Cal1groupXS：非燃料媒質を含む全ての媒質における 1 群断面積を出力する。現在整備中なので、Pring1groupXS メソッドを替わりに利用のこと。

- ShowNeutronMultiplicationInfo：現在の原子炉の状態における核分裂連鎖反応の詳細な情報を出力する（詳細は CBZ のチュートリアル（応用編）参照のこと）。例えば、Nuclide-wise fission contribution では、原子炉全体の核分裂反応における核種毎の寄与割合を知ることが出来る。

とによって「1 群の」断面積を計算することが出来る。具体的には、断面積を $\sigma(E)$、中性子束を $\phi(E)$ としたとき、1 群断面積 $\bar{\sigma}$ は

$$\bar{\sigma} = \frac{\int \sigma(E)\phi(E)dE}{\int \phi(E)dE} = \frac{\sum_{g=1}^{G} \sigma_g \phi_g}{\sum_{g=1}^{G} \phi_g} \tag{2}$$

と定義される。

ここで、CalVoidReactivity、CalDopplerReactivity メソッドによる摂動計算に基づく反応度計算例を示す。

Listing 23: 摂動計算による反応度の計算例

```
#************************************************
#* System CBG
#*    Perturbation calculation
#*    Solver : PLOS
#************************************************
#    Energy      Yield       Absorption  Scattering  Leakage     (n,2n)      Total
   0 1.000e+07  0.000e+00  1.428e-04  1.809e-04  -1.124e-04 0.000e+00 2.113e-04
   1 7.788e+06  0.000e+00  1.352e-04  2.724e-04  -2.618e-04 0.000e+00 1.457e-04
   2 6.065e+06  0.000e+00  5.023e-05  1.559e-04  -5.031e-04 0.000e+00 -2.970e-04
   3 4.724e+06  0.000e+00  5.277e-06  3.263e-04  -7.248e-04 0.000e+00 -3.933e-04

  67 5.316e-01  0.000e+00  1.825e-08  4.210e-09  1.521e-08 0.000e+00 3.767e-08
  68 4.140e-01  0.000e+00  1.493e-08  4.087e-09  7.619e-09 0.000e+00 2.664e-08
  69 3.224e-01  0.000e+00  9.862e-10  0.000e+00  2.235e-10 0.000e+00 1.210e-09
#
# +++ Summary (energy-integrated value) +++
#
# Yield       : 0.000e+00
# Absorption  : 1.698e-03
# Scattering  : 2.242e-02
# N2N         : 0.000e+00
# (Non-Leak)  : 2.412e-02
# Leakage-r   : -1.045e-02
# Leakage-z   : -3.055e-02
# (Leakage)   : -4.100e-02
#
# ** Perturbation Cal.   : -1.688e-02
# ** Direct Cal.         : -1.687e-02
```

参照すべき反応度は 27 行目（下から 2 行目）の Perturbation Cal. に示されている値である。反応度は要因別にも示されており、それが 18 から 25 行目に対応する。19 行目の Absorption は体系の吸収断面積が、20 行目の Scattering は体系の散乱断面積が変動した影響をそれぞれ示している。また、25 行目の (Leakage) は体系からの中性子漏洩が変動した影響を示している。16 行目の Summary 以前は、それぞれの要因成分について、エネルギー群毎に示しているものである。

また、体系に含まれている全ての媒質のデータ（Medium クラスのインスタンスのデータ）をファイルとして出力することが出来る。これは FRBurner クラスの WriteFileMediumData メソッドにより行なう。例を以下に示す。

Listing 24: 全媒質データの外部ファイルへの出力例

```
frbn.WriteFileMediumData("./MED_DATA/","med");
```

ひとつめの引数で出力されるファイルのディレクトリを、ふたつめの引数でファイル名を、それぞれ指定する。媒質のファイル名は、指定された名前に ID 番号が付加されて出力される。この例では med0、med1 といったファイルが MED_DATA ディレクトリに出力されることになる。

3.7 動特性パラメータの計算

動特性パラメータの実効遅発中性子割合 β_{eff}、即発中性子寿命 l を計算するための入力例を以下に示す（パッケージの main.rz.halfcore.cxx にコメントアウトされている）。

Listing 25: 動特性パラメータ計算のための入力例

```
  DelayedNeutronData Del;
  string mdir2("../../CBGLIB/DELAYED/J4.70G/");
  Del.PutDataFromFile(mdir2,"Th232");
  Del.PutDataFromFile(mdir2,"U233");
  Del.PutDataFromFile(mdir2,"U234");
  Del.PutDataFromFile(mdir2,"U235");
  Del.PutDataFromFile(mdir2,"U238");
  Del.PutDataFromFile(mdir2,"Pu239");
  Del.PutDataFromFile(mdir2,"Pu240");
  Del.PutDataFromFile(mdir2,"Pu241");
```

```
11    Del.PutDataFromFile(mdir2,"Pu242");
12    frbn.CalDelayedNeutronParameters(Del);
13    //Del.ShowSelf();
```

また、重核の遅発中性子データの詳細については、DelayedNeutronData クラスの ShowSelf メソッドにより画面上に出力することが出来る。出力例を以下に示す。最初の#Nu_d に、入射中性子エネルギー群毎の核分裂あたりの遅発中性子発生数（ν_d）が核種毎に示されている。非常に高いエネルギー領域では ν_d は大きな値をとるが、それより低い領域ではほぼ一定の値となり、例えば Th-232（MATID が 902320）だと 0.0490、U-238（MATID が 922380）だと 0.0463 となっていることが分かる。

Listing 26: 遅発中性子データの出力例

```
1  # Nu_d
2  # 902320 922330 922340 922350 922380 942390 942400 942410 942420
3  0 3.34e-02 4.80e-03 8.37e-03 1.01e-02 3.05e-02 4.80e-03 7.79e-03 1.15e-02 1.30e-02
4  1 3.66e-02 4.86e-03 9.05e-03 1.04e-02 3.71e-02 4.94e-03 8.21e-03 1.31e-02 1.36e-02
5  2 4.85e-02 6.22e-03 1.03e-02 1.31e-02 4.31e-02 5.56e-03 8.94e-03 1.56e-02 1.49e-02
6  3 4.90e-02 7.69e-03 1.05e-02 1.55e-02 4.63e-02 6.08e-03 9.10e-03 1.60e-02 1.53e-02
7  4 4.90e-02 7.93e-03 1.05e-02 1.69e-02 4.63e-02 6.48e-03 9.11e-03 1.60e-02 1.53e-02
8  5 4.90e-02 7.80e-03 1.05e-02 1.68e-02 4.63e-02 6.53e-03 9.11e-03 1.60e-02 1.53e-02
9  6 4.90e-02 7.70e-03 1.05e-02 1.67e-02 4.63e-02 6.46e-03 9.11e-03 1.60e-02 1.53e-02
10 7 4.90e-02 7.63e-03 1.05e-02 1.66e-02 4.63e-02 6.41e-03 9.11e-03 1.60e-02 1.53e-02
11 8 4.90e-02 7.56e-03 1.05e-02 1.65e-02 4.63e-02 6.37e-03 9.11e-03 1.60e-02 1.53e-02
12
13 65 4.90e-02 6.70e-03 1.05e-02 1.59e-02 4.63e-02 6.22e-03 9.11e-03 1.60e-02 1.53e-02
14 66 4.90e-02 6.70e-03 1.05e-02 1.59e-02 4.63e-02 6.22e-03 9.11e-03 1.60e-02 1.53e-02
15 67 4.90e-02 6.70e-03 1.05e-02 1.59e-02 4.63e-02 6.22e-03 9.11e-03 1.60e-02 1.53e-02
16 68 4.90e-02 6.70e-03 1.05e-02 1.59e-02 4.63e-02 6.22e-03 9.11e-03 1.60e-02 1.53e-02
17 69 4.90e-02 6.70e-03 1.05e-02 1.59e-02 4.63e-02 6.22e-03 9.11e-03 1.60e-02 1.53e-02
18 #
19 # Family-wise abundance
20 #
21 902320 3.642e-02 1.259e-01 1.501e-01 4.406e-01 1.663e-01 8.078e-02
22 922330 8.593e-02 2.292e-01 1.781e-01 3.516e-01 1.142e-01 4.089e-02
23 922340 5.497e-02 1.964e-01 1.803e-01 3.877e-01 1.324e-01 4.818e-02
24 922350 3.500e-02 2.170e-01 2.120e-01 3.850e-01 1.260e-01 2.500e-02
25 922380 1.300e-02 1.370e-01 1.620e-01 3.880e-01 2.250e-01 7.500e-02
26 942390 3.300e-02 2.850e-01 2.190e-01 3.260e-01 8.600e-02 5.100e-02
27 942400 3.197e-02 2.529e-01 1.508e-01 3.301e-01 1.795e-01 5.471e-02
28 942410 1.805e-02 2.243e-01 1.426e-01 3.493e-01 1.976e-01 6.821e-02
29 942420 1.964e-02 2.314e-01 1.256e-01 3.262e-01 2.255e-01 7.160e-02
30 #
31 # Family-wise decay constant
32 #
33 902320 1.310e-02 3.500e-02 1.272e-01 3.287e-01 9.100e-01 2.820e+00
34 922330 1.290e-02 3.470e-02 1.193e-01 2.862e-01 7.877e-01 2.442e+00
35 922340 1.310e-02 3.370e-02 1.210e-01 2.952e-01 8.136e-01 2.572e+00
36 922350 1.244e-02 3.054e-02 1.114e-01 3.014e-01 1.136e+00 3.014e+00
37 922380 1.323e-02 3.209e-02 1.386e-01 3.591e-01 1.415e+00 4.030e+00
38 942390 1.277e-02 3.014e-02 1.238e-01 3.254e-01 1.122e+00 2.697e+00
39 942400 1.330e-02 3.050e-02 1.152e-01 2.974e-01 8.477e-01 2.880e+00
40 942410 1.360e-02 3.000e-02 1.167e-01 3.069e-01 8.701e-01 3.003e+00
41 942420 1.360e-02 3.020e-02 1.154e-01 3.042e-01 8.272e-01 3.137e+00
```

3.8 線出力空間分布の時間推移

前述した通り、燃焼計算の前に PrintLinePowerMap メソッドを実行しておくと、線出力空間分布の時間推移データが linepower_map に出力され、これを用いて簡単な動画を作成することが出来る。

線出力の空間分布は、各燃焼サブステップ毎に出力されるため、燃焼ステップ分割数を n_1、サブステップ分割数を n_2 とすると、$n_1 \times (n_2 + 1)$ 点での線出力空間分布データが出力されることになる。この数値は linepower_map ファイルの 1 行目にも出力されている。

この linepower_map ファイルを元に、GNUPLOT を用いて動画を作成するためにシェルスクリプトの例を以下に示す。3 行目はデータ点数に応じたループを定義しており、この例では 18 点のデータがあるため 0 から 17 までのループとなっている。X 軸、Y 軸のプロット領域については 20、21 行で定義する（この例ではコメントアウトされている）。表示しているデータの範囲については 25 行目で定義する。また、ページの切り替え速度は 35 行目の-delay のあとの数値を変更することで調整可能である。最終的に pmap.gif というファイルが生成される。

Listing 27: 線出力空間分布の時間推移を簡単な動画として出力するためのシェルスクリプト例

```
#!/bin/sh

for loop in `seq 0 17`
do

loop10=`expr $loop + 10`

if test $loop -lt 10 ; then
  output=pmap00$loop.png
elif test $loop -lt 100 ; then
  output=pmap0$loop.png
else
  output=pmap$loop.png
fi
echo "writing file : $output"

gnuplot -persist <<EOF
#set size 0.7,0.7
set pm3d map
#set xrange[50:100]
#set yrange[30:60]
set xlabel 'R [cm]'
set ylabel 'Z [cm]'
#set logscale cb
set cbrange[:350]
#set format cb "10^{%L}"
set terminal png
set output '$output'
set title '$loop'
splot "linepower_map" i $loop:$loop ti ''
EOF

done

convert -loop 0 -delay 20 pmap*.png pmap.gif
rm pmap*.png
```

4 練習問題

Fig. 6 に中型の高速増殖炉の燃料配置図（原子炉を上から見た図に対応する）を示す。

白抜きの燃料集合体が内側炉心燃料集合体、「#」のマークが付されているものが外側炉心燃料集合体である。両者の主な差異は Pu 富化度が異なる点にある。また、内側炉心領域に配置されている「C」「F」「B」が付されている集合体は制御棒を示し、それぞれ「粗調整棒」「微調整棒」「後備炉停止棒」に対応する。外側炉心燃料の周囲を囲む 3 層の領域には径ブランケット集合体が、さらにその外側には遮蔽体が配置されている。前述したように、各々の集合体は均質[11] として扱っており、炉心計算ではその詳細な内部構造は無視される。

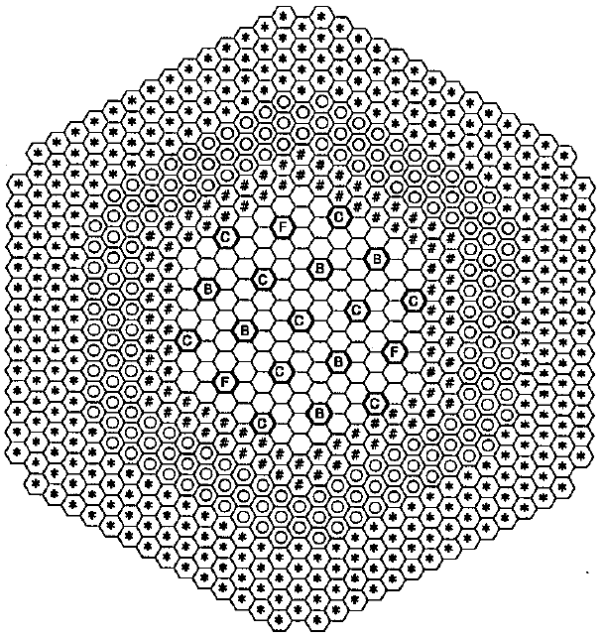

Fig. 6: 中型高速増殖炉の燃料配置図

この炉心体系を、**Fig. 7** に示すように、円筒体系の二次元で近似してモデル化する。なお、軸方向は厳密には対称ではないが、簡略化のため、軸対称を仮定し、1/2 炉心モデルとして扱う。

この計算では、はじめは全て新燃料が装荷されるとする。このような炉心を**初装荷炉心**と呼び、反応度が高い。その後、サイクルが進むにつれて、（5 バッチの場合は）1/5 ずつ燃料が交換される。初装荷炉心から 5 サイクルが経過すると、全ての燃料集合体は一度ずつ交換されることになり、平衡状態に達する。このような炉心を**平衡炉心**（equilibrium core）と呼ぶ。

今回の演習では、平衡サイクルの初期（beginning of equilibrium cycle、BOEC）、中期（MOEC）、末期（EOEC）において、実効増倍率と最大線出力等を計算する。

この中型高速炉の 2 次元モデルに対して FRBurner を用いて炉心燃焼計算を行い、以下の設問に対する解答を PPT 形式にまとめよ。図等の完成度は問わない（凝った図を作成する必要は無い）。なお、各原子核の多群断面積を観察したい場合もあるだろう。この場合は、「CBZ のチュートリアル（応用編）」に、多群断面積ライブラリ CBZLIB からの断面積データの取り出し方法が記載されているので、そちらを参照するとよい。

また、下記の演習問題を取り組むにあたっては、核燃料の燃焼に伴う重核種の変換過程を理解する必要がある。**Fig. 8** に、SRAC-2006 コードで用いられている主要重核種の燃焼チェーンを示す。今回の例で用いている燃焼チェーンはこれとは微妙に異なるが、同様と考えてよいので、これを参考にしてもらいたい。

[11] 集合体の構成物質をすべて溶かして混ぜ込み、一種類の単一な媒質とした、とイメージすれば良い。

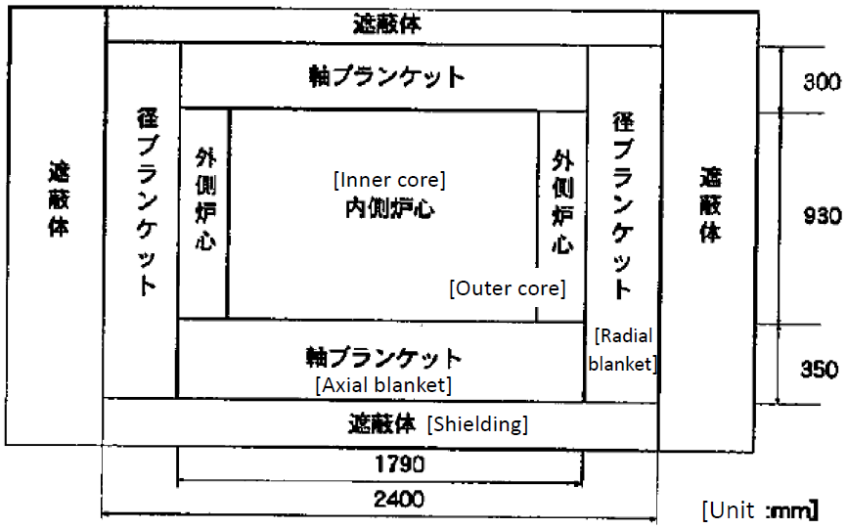

Fig. 7: 中型高速増殖炉の円筒モデル

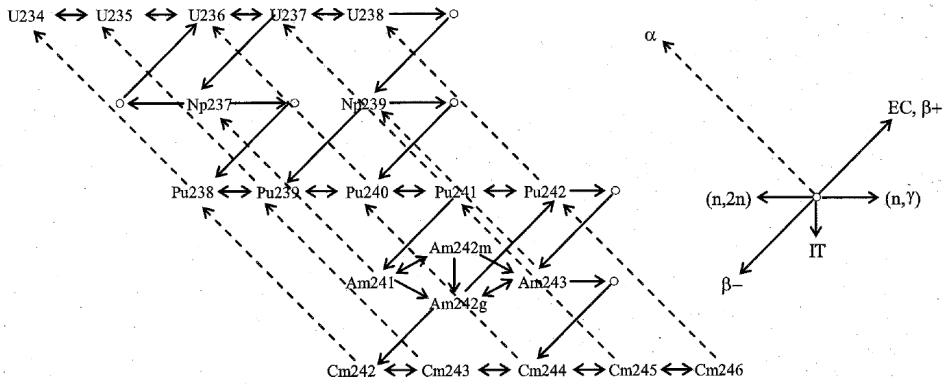

Fig. 8: SRAC-2006 の主要重核種の燃焼チェーン

1. 内側炉心、外側炉心、径ブランケット、軸ブランケットの平均巨視的核分裂生成断面積（Production cross section, $\nu\Sigma_f$）を、`PrintMacroXS` メソッドにより取り出し、横軸を中性子エネルギー、縦軸を断面積として図にプロットせよ。なお、両軸とも対数で表すこととする。また、内側炉心と外側炉心の違いを整理せよ。核分裂生成断面積は内側炉心よりも外側炉心で大きいが、その炉心設計上の理由を考えよ。さらに、内側（もしくは外側）炉心とブランケット領域の核分裂生成断面積を比較し、差異の原因を述べよ。

2. 燃焼に伴う実効増倍率の変動、最大線出力の変動を、横軸を運転経過日数として図示せよ。ただし、炉心、ブランケットともに交換バッチを 5 とする。また、最大線出力の燃焼に伴う挙動の特徴について説明せよ（`PrintLinepowerMap` メソッドを使うことで画面に出力される線出力の空間分布を観察するとよいであろう）。

3. 炉心、ブランケットともに交換バッチを 4、さらに 3 に変更したときの実効増倍率の燃焼挙動の変化を調べよ。また、その変化の理由について説明せよ（交換バッチの変更は `PutBatchMZone` メソッドに与える引数の内容を変えることにより行える）。

4. EOEC の実効増倍率が 1.0 をわずかに越える程度（1.000 以上かつ 1.003 未満）になるように、かつ BOEC、MOEC、EOEC における最大線出力が最も小さくなるように、Pu 富化度を内側炉心、外側

炉心で調整せよ。燃料交換バッチ数は 5 とする。なお、ここで構築した炉心を「基準炉心」とする（Pu 富化度の調整は `PutPuFissileEnrichment` メソッドで行える）。

5. 基準炉心について、EOEC の中性子束エネルギースペクトルを、内側炉心、外側炉心、径ブランケット、軸ブランケットの間で比較せよ。全ての媒質について比較するのは大変なので、各々の領域について代表的な媒質を選べばよい。また、領域間の中性子スペクトルの特徴的な違いと、その違いが生じる理由について述べよ。

6. 基準炉心について、内側炉心、外側炉心、ブランケット領域における U-238、Pu-239、Pu-241 の核種数密度の燃焼サイクル依存性を図示せよ。なお、横軸には燃焼サイクル、縦軸には数密度をとることとする。

7. 基準炉心に対して、U-235 濃縮度を、内側炉心、外側炉心ともに 5.0wt%および 10.0wt%として Pu 富化度を調整せよ。ただし、基準炉心作成時と同様に、EOEC の実効増倍率が 1.0 をわずかに超える程度とし、最大線出力を最小化させるものとする。さらに、それらの炉心の実効増倍率と最大線出力の燃焼挙動を基準炉心と比較せよ。

8. 基準炉心に対して、Pu-241 の一部が崩壊して Am-241 が生成した燃料（Pu-241、Am-241 の TRU 組成比をぞれぞれ 7%に変更する）を使った場合を考え、Pu 富化度を調整せよ。なお、炉心成立条件は前問と同様とする。さらに、それらの炉心の実効増倍率と最大線出力の燃焼挙動を基準炉心と比較し、その理由を考察せよ。

9. 基準炉心に対して Pu を全く含まない燃料を用いた場合に原子炉が成立するか調べよ（U-235 濃縮度については内側炉心と外側炉心で異なる値にしてもよい）。成立するならば、実効増倍率と最大線出力について基準炉心と比較し、差異を説明せよ。ただし、ブランケット燃料の組成は不変とする。

10. U-235 濃縮度、Pu 富化度、TRU 組成比を調整し、サイクル長を最大化する炉心を構築せよ。なお、ブランケット燃料の組成は不変とし、平衡炉心の最大線出力は 360 W/cm 以下とする。炉心を構築したならば、どのような要因でその炉心のサイクル長を基準炉心と比較して大きくできたか説明せよ。また、平衡炉心における燃焼反応度（BOEC と EOEC の反応度差）を 0.04Δk 以下とする制約条件を付加した場合についても同様のことを実施せよ。

11. 基準炉心の EOEC に対して、内側炉心、外側炉心、径ブランケット、軸ブランケット各々の領域の冷却材が沸騰した場合に炉心に印加される反応度を計算せよ（つまり反応度が 4 種計算されることになる）。各々の結果について、どのようなメカニズムで、正もしくは負の反応度が印加されるか、説明せよ。

A トリウム燃料装荷炉心の解析

FRBurner は、基本的には U-235 と U-238 からなる UO_2 と PuO_2 の混合酸化物燃料を扱うことを想定しているが、ThO_2 や、U-235、-238 以外のウラン同位体を含む UO_2 も扱うことが可能である。このような場合は、まずは多群断面積ライブラリ CBZLIB と燃焼計算チェーンの指定方法を通常と変える必要がある。パッケージには、「main.rz.halfcore.new_fuel_inp.cxx」として、入力例が収納されている。

このマニュアルの本文中で使われている燃焼チェーンは 17 個の重核種で構成されている簡略化チェーンである。この場合の断面積ライブラリクラス CBZLIB と燃焼チェーンを含む燃焼計算クラス Burnup のインスタンスの作成方法は以下の通りである。

Listing 28: BurnupChain クラスのデフォルト燃焼チェーンを使う場合の入力例

```
// +++ Library
string libdir("/home/chiba/CBGLIB/j4.70g.iwt7/");
XSLibrary xslib(libdir,"N-ENERGY");

// Default chain (17 nuclides)
string filename[]={
  "U235","U236","U238","Np237","Pu238",
  "Pu239","Pu240","Pu241","Pu242","Am241",
  "Am242m","Am243","Cm242","Cm243","Cm244",
  "Cm245","Cm246","FP.U235","FP.U238","FP.Pu239",
  "FP.Pu241","O016","Na023","Cr000","Mn055",
  "Fe000","Ni000","Mo000","B010","B011",
  "C000",
};
int matno[]={
  922350,922360,922380,932370,942380,  942390,942400,942410,942420,952410,
  952421,952430,962420,962430,962440,  962450,962460,9950000,9980000,9990000,
  9910000,80160,110230,240000,250550,  260000,280000,420000,50100,50110,
  60000,
};
xslib.ReadFile(31,libdir,filename);
int fuel_nuc_num=28; // fuel composition
int burn_nuc_num=21; // burnup calculation

// +++ Burnup chain
Burnup bu;
bu.SetDefaultChain();
bu.ReadReactionEnergyFromFile("../../","fr_standard");
bu.GetBurnupChain().ReadDecayConstantFromFile("../../","srac_org");
```

これに対して 4 つの重核種（U-234、-237、Np-239、Am-242）を追加した燃焼チェーンを用いる場合は、以下の入力を用いる。

Listing 29: BurnupChain クラスの 21 重核種燃焼チェーンを使う場合の入力例

```
// +++ Library
string libdir("/home/chiba/CBGLIB/j4.70g.iwt7/");
XSLibrary xslib(libdir,"N-ENERGY");

// 21 chain
string filename[]={
  "U234","U235","U236","U237","U238",
  "Np237","Np239","Pu238","Pu239","Pu240",
  "Pu241","Pu242","Am241","Am242","Am242m",
  "Am243","Cm242","Cm243","Cm244","Cm245",
  "Cm246","FP.U235","FP.U238","FP.Pu239","FP.Pu241",
  "O016","Na023","Cr000","Mn055","Fe000",
  "Ni000","Mo000","B010","B011","C000",
};
int matno[]={
  922340,922350,922360,922370,922380,
  932370,932390,942380,942390,942400,
  942410,942420,952410,952420,952421,
  952430,962420,962430,962440,962450,
  962460,9950000,9980000,9990000,9910000,
  80160,110230,240000,250550,260000,
  280000,420000,50100,50110,60000,
};
xslib.ReadFile(35,libdir,filename);
int fuel_nuc_num=32; // fuel composition
int burn_nuc_num=25; // burnup calculation

// +++ Burnup chain
Burnup bu;
```

```
30  bu.SetDefaultChain();
31  bu.GetBurnupChain().Set21HeavyMetalChain(true); // "true" is for fast reactor analysis
32  bu.GetBurnupChain().AddPseudoFP();
33  bu.ReadReactionEnergyFromFile("../../","fr_standard");
34  bu.GetBurnupChain().ReadDecayConstantFromFile("../../","srac_org");
```

さらに、これに対して 7 つの重核種（Th-232、Pa-231、Pa-233、U-232、-233、Np-236、Pu-236）を追加した燃焼チェーンを用いる場合は、以下の入力を用いる。このチェーンには Th-232 や U-233 が含まれているため、ThO_2 燃料の計算が可能となる。

Listing 30: BurnupChain クラスの 28 重核種燃焼チェーンを使う場合の入力例

```
1   // +++ Library
2   string libdir("/home/chiba/CBGLIB/j4.70g.iwt7/");
3   XSLibrary xslib(libdir,"N-ENERGY");
4
5   // 28 chain
6   string filename[]={
7     "Th232","Pa231","Pa233","U232","U233",
8     "U234","U235","U236","U237","U238",
9     "Np236","Np237","Np239","Pu236","Pu238",
10    "Pu239","Pu240","Pu241","Pu242","Am241",
11    "Am242","Am242m","Am243","Cm242","Cm243",
12    "Cm244","Cm245","Cm246","FP.U235","FP.U238",
13    "FP.Pu239","FP.Pu241","O016","Na023","Cr000",
14    "Mn055","Fe000","Ni000","Mo000","B010",
15    "B011","C000",
16  };
17  int matno[]={
18    902320,912310,912330,922320,922330,
19    922340,922350,922360,922370,922380,
20    932360,932370,932390,942360,942380,
21    942390,942400,942410,942420,952410,
22    952420,952421,952430,962420,962430,
23    962440,962450,962460,9950000,9980000,
24    9990000,9910000,80160,110230,240000,
25    250550,260000,280000,420000,50100,
26    50110,60000,
27  };
28  xslib.ReadFile(42,libdir,filename);
29  int burn_nuc_num=28+4; // burnup calculation
30  int fuel_nuc_num=28+4+7; // fuel composition
31
32  // +++ Burnup chain
33  Burnup bu;
34  bu.GetBurnupChain().Set28HeavyMetalChain(true); // "true" is for fast reactor analysis
35  bu.GetBurnupChain().AddPseudoFP();
36  bu.ReadReactionEnergyFromFile("../../","fr_standard");
37  bu.GetBurnupChain().ReadDecayConstantFromFile("../../","srac_org");
```

ThO_2 燃料等を用いる場合には、燃料組成情報を定義するクラス FRDTFuelComposition を、本文中の 1.1 節とは異なる使い方で用いることになる。以下に入力例を示す。

Listing 31: UO_2、PuO_2、ThO_2 で構成される燃料組成の定義例

```
1   FRDTFuelComposition fc_ic;
2
3   fc_ic.PutOM(1.97);
4   fc_ic.PutPelletTheoreticalDensity(85.);
5   int uo2_id[]={922350,922380};
6   real uo2_wgt[]={0.002, 0.998};
7   fc_ic.PutUO2Composition(2,uo2_id,uo2_wgt);
8   int puo2_id[]={942380,942390,942400,942410,942420,952410};
9   real puo2_wgt[]={0., 58., 24., 14., 4., 0.};
10  fc_ic.PutPuO2Composition(6,puo2_id,puo2_wgt);
11  fc_ic.CalNumberDensity(100.-22.3611,22.3611,0.); // UO2/PuO2/ThO2
```

まずは、燃料の O/M 比と理論密度を設定する（3、4 行目）。その後、5 から 7 行目で、UO_2 燃料における同位体組成とその重量比を設定する。この例では U-235 が 0.2wt%、U-238 が 99.8wt%となっている。さらに、8 から 10 行目で、PuO_2 燃料における同位体組成とその重量比を設定する。なお、PuO_2 には Pu 同位体に加えて Am 同位体も加えることが出来る。最後に、11 行目で、UO_2、PuO_2、ThO_2 の重金属重量比を CalNumberDensity メソッドに引数として渡してやることにより、燃料ペレット中の核種数密度を決定

する[12]。なお、重量比はコード内で規格化されるため、どのような単位でも構わない（総和が 1.0 や 100.0 になる必要が無い）。

[12] UO_2、PuO_2、ThO_2 の密度はそれぞれ 10.96、11.44、9.87（[g/cm^3]）と仮定している。また、ThO_2 には Th-232 のみが含まれるとしている。

B　二次元円筒体系へのモデル化の詳細

Fig. 9 に示す高速炉心を円筒体系にモデル化する方法の詳細について述べる。この図では、燃料リング第１層位置に粗調整制御棒（「C」）が配置され、それを取り囲むように第２層（水色の線）、第３層（赤の線）、第４層（緑の線）、第５層（紫の線）が配置されている。第６層以降については、本節では説明を省略する。

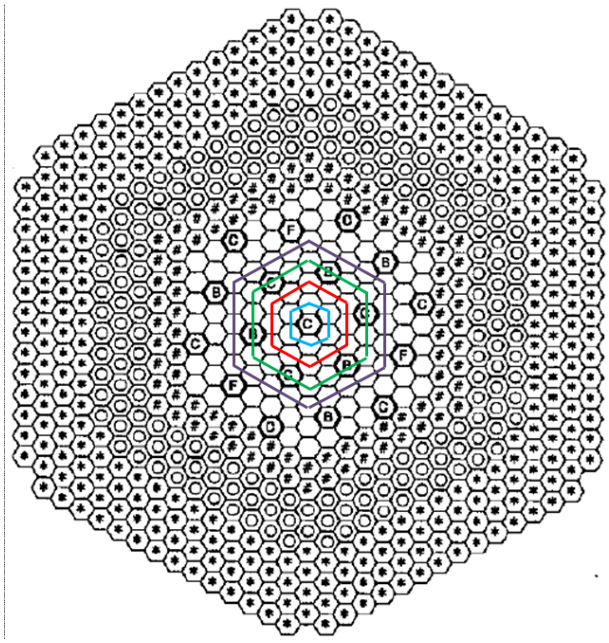

Fig. 9: 円筒にモデル化する高速炉心

この炉心を円筒にモデル化する場合には、各層毎に面積が保存されるように等価な円筒を構築していく。その概念を示したものを **Fig. 10** に示す。この図では第５層まで考慮した円筒炉心モデルが示されている。第４層には燃料集合体に加えて制御棒集合体も配置されているが、このような層は燃料、制御棒、燃料というようにさらに３つの層に区別してモデル化を行っている。この図で黒く示されているのが制御棒集合体をモデル化したものである。制御棒集合体に対応する層の内側と外側にそれぞれ燃料集合体の層が配置されるが、その割合は任意であるので、通常は面積が 1:1 となるように配置している。

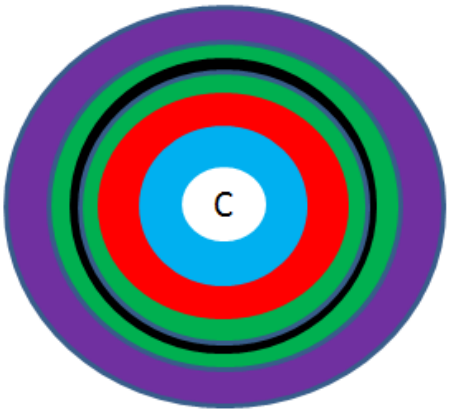

Fig. 10: 円筒状にモデル化された高速炉心

最後に、円筒状にモデル化した炉心の物質配置マップ（配列 [mat]）を **Fig. 11** に示す。配列 [xl] は各層の幅を示す。対応する燃料リングと同一の色の破線で物質 Index を囲っている。燃料集合体と制御棒集合体が含まれる第４層では、円筒モデルで制御棒リングの内側に配置される燃料と外側に配置あｓれる燃料とに対して同一の媒質 ID（6、7、8）が割り当てられている。FRBurner では媒質毎に燃焼追跡計算が行われるので、同じ燃料リングに属する媒質は同一として扱ったほうがよいであろう。

```
real xl[]={
  6.06943,  9.98878, 10.39784,  2.32216,  3.61336,
  4.52736, 10.48487,  7.73198,  3.87278,  9.39049,
 10.50360, 10.50570, 10.50715, 10.50818,  9.59162,
 30.0
};
int mat[]={
  40, 0, 3, 6,40, 6, 9,12,40,15,18,21,24,29,31,39,
  40, 1, 4, 7,40, 7,10,13,40,16,19,22,25,29,31,39,
  40, 2, 5, 8,40, 8,11,14,40,17,20,23,26,29,31,39,
  40,32,32,32,40,32,33,33,40,33,35,35,27,30,31,39,
  40,34,34,34,40,34,34,34,40,34,36,36,28,30,31,39,
  40,37,37,37,40,37,37,37,40,37,37,37,38,38,38,39,
};
```

Fig. 11: 円筒状にモデル化された炉心の物質配置マップ

C 加速器駆動未臨界炉（ADS）の計算

FRBurner では JAEA が提案している鉛ビスマス冷却の ADS [13] の計算が可能である。

C.1 燃料集合体の均質数密度データの作成

2020 年 4 月 13 日現在の FRBurner では、燃料集合体計算において 1 次元もしくは 2 次元非均質モデルを利用可能であるが、ADS の計算においてはこれらはまだテストしていないため、燃料集合体計算は全て均質モデルを用いて行う。燃料集合体の均質組成データの作成方法は、高速炉については 2 章に記載しているが、ADS の場合には形式が大きく異なるため、ここで説明する。

JAEA が設計した ADS では、MA 窒化物（MAN）、Pu 窒化物（PuN）、Zr 窒化物（ZrN）の混合燃料を用いる。従って、ペレットの組成は、これらの混合割合（重量比）と、それぞれにおける核種の同位体比から求められることになる。

以下に、燃料ペレット中の MAN、PuN の同位体比を定義するための例を示す。燃料集合体の組成情報は FRDTADSFuelComposition クラスで定義され、燃料ペレット中の MAN、PuN の重核種同位体比は PutMAData、PutPuData メソッドで指定することが出来る。これらのメソッドでは、引数に MAN、PuN それぞれが含む重核種の種類数、重核種の核種 ID の配列、同位体比の配列を渡す。なお、窒素については N-15 が 100%として扱われる。また、ZrN 中のジルコニウムについては、天然の同位体組成が仮定される。

Listing 32: MAN、PuN の重核種同位体比の定義例

```
1   FRDTADSFuelComposition fic;
2   int pu_id[]={
3     922340,922350,922360,922370,922380,
4     942380,942390,942400,942410,942420,
5     952410,
6   };
7   real pu_vector[]={
8     0.0004, 0.,    0.0001, 0.,     0.,
9     0.0238, 0.5447,0.2419, 0.1085, 0.0696,
10    0.0109
11  };
12  fic.PutPuData(11,pu_id,pu_vector);
13
14  int ma_id[]={
15    932370,932390,
16    942400,
17    952410,952420,952421,952430,
18    962420,962430,962440,962450,962460,962470,962480,962490,962500,
19    972490,
20  };
21  real ma_vector[]={
22    0.4965, 0.,
23    0.0032,
24    0.3210, 0.,     0.0006, 0.1337,
25    0.,     0.0003, 0.0404, 0.0039, 0.0004, 0.,0.,0.,0.,
26    0.,
27  };
28  fic.PutMAData(17,ma_id,ma_vector);
```

ADS 計算の場合、2 章で述べたように詳細な集合体の幾何形状データは入力せず、以下の手続きで集合体の均質数密度データを作成する。

集合体は、ペレット、被覆管、冷却材の 3 種類の媒質で構成されるものと仮定し、それぞれの集合体中の体積比を volume_fraction_pellet、volume_fraction_cladding、volume_fraction_coolant とする。被覆管、冷却材領域に含まれる核種の均質数密度はユーザが直接入力する。以下に、これらの領域に含まれる核種の均質数密度の指定例を示す。この操作は PutNonfuelData メソッドで行う。このメソッドの引数は PutMAData、PutPuData メソッドと同様である。ここで、「均質の（集合体平均の）」数密度を直接入力する点に注意が必要である。例えば、この例では 7 から 10 行目で配列 str_den を指定しているが、これは被覆管領域と冷却材領域に含まれる核種の数密度に対応している（最初の 5 つが被覆管領域、その他が冷

[13] K. Tsujimoto, et al., Neutronics design for lead-bismuth cooled accelerator-driven system for transmutation of minor actinoide, J. Nucl. Sci. Technol., 41[1], p.21-36 (2004).

却材領域である）。その後、11 から 16 行目で体積比が乗ぜられているが、この部分は、集合体平均の数密度への変換に対応している。

Listing 33: 被覆管、冷却材領域に含まれる核種数密度の定義例

```
1   int str_id[]={
2     240000, 250550, 260000, 280000, 420000, // Cladding
3     822040, 822060, 822070, 822080, 832090, // Coolant
4   };
5
6   real pb_den=1.31228E-02;
7   real str_den[]={
8     1.52695E-02,  8.50214E-04,  5.64500E-02,  9.54662E-03, 1.21728E-03,
9     pb_den*0.014, pb_den*0.241, pb_den*0.221, pb_den*0.524,  1.62272e-2,
10  };
11  for(int i=0;i<5;i++){
12    str_den[i]*=volume_fraction_cladding;
13  };
14  for(int i=5;i<10;i++){
15    str_den[i]*=volume_fraction_coolant;
16  };
17
18  fic.PutNonfuelData(10,str_id,str_den);
```

以上のように、燃料ペレット中の PuN、MAN の同位体組成を指定し、さらに被覆管、冷却材領域に含まれる核種の数密度を指定したあとに、以下の例で示すように、ペレット中の MAN、PuN、ZrN の重量比を `PutFuelWeightRatio` メソッドにより指定する。引数の３つ目までが、ZrN、PuN、MAN の重量比に対応し、４つ目が集合体におけるペレット領域の体積比に対応する。このメソッドにより、`FRDTADSFuelComposition` クラスのインスタンスが燃料集合体の均質数密度を計算し、内部に保持する。

Listing 34: 被覆管、冷却材領域に含まれる核種数密度の定義例

```
1   fic.PutFuelWeightRatio(zrn_inp, (1.-zrn_inp)*pun_inp_ic, (1.-zrn_inp)*(1.-pun_inp_ic),
2     volume_fraction_pellet); // Zr-N, Pu-N, MA-N
```

`FRBurner` では、集合体の均質数密度データは `Medium` クラスのインスタンスにより入力する必要があるため、次のステップとしては、`FRDTADSFuelComposition` クラスのインスタンスから `Medium` クラスのインスタンスに均質数密度データを与える必要がある。その例を以下に示す。まずは、`FRBurner` クラスのインスタンスに与えられる媒質クラスのインスタンスとして `min_ic` を生成する。4 行目では、`Burnup` クラスのメソッド `AddNuclideToMediumFromBurnupChain` により、`min_ic` が引数として引き渡されているが、これにより、燃焼チェーンに含まれている核種が `min_ic` に追加で定義される（この操作を行わないと、燃焼により新たに生成される核種の数密度データを管理できないため）。`FRDTADSFuelComposition` クラスのインスタンスからのデータの引き渡しは 5 行目の `PutNumberDensity` メソッドにより行われている。これにより必要なデータは全て `Medium` クラスのインスタンス `min_ic` に渡されたことになる。

Listing 35: `FRDTADSFuelComposition` クラスのインスタンスから `Medium` クラスのインスタンスへの均質数密度データの受け渡しの例

```
1   Medium min_ic;
2   min_ic.PutImax(group);
3   min_ic.PutPL(1);
4   bu.AddNuclideToMediumFromBurnupChain(min_ic); // !!
5   fic.PutNumberDensity(min_ic);
6   min_ic.PutTemperatureForAllNuclide(983.);
7   //min_ic.ShowNumberDensity(false);  exit(0);
```

C.2 燃料交換の記述

ADS の燃焼計算では、サイクル終了後、燃料を全て取り出して、そこから FP 核種を取り除き、取り除かれた FP と同重量の MA（LWR 使用済み燃料由来）を追加して次のサイクルに再装荷する、という手順を踏む。ここでは全ての使用済み燃料を一括して処理すると想定するため、再装荷燃料の組成は単一（全ての集合体で同一）となる。CBZ では、燃焼感度計算を想定して、燃料交換を行列形式で記述して扱っているため、ここではそれについて説明する。

サイクル終了後の燃料交換前の領域（媒質）j、核種 k の数密度を $N_k^j[/\mathrm{cm}^3]$、領域 j の体積を $V^j[\mathrm{cm}^3]$、核種 k の 1 個あたりの重量を w_k[g] と記述する。w_k は核種 k の原子量をアボガドロ数で除した値となる。

さて、核種 k の物質ゾーン（Material zone）z での総重量 $W_{k,z}$[g] は以下のように書ける。

$$W_{k,z} = \sum_{j \in z} N_k^j V^j w_k \tag{3}$$

そして、重核種の物質ゾーン z での総重量 $W_{HM,z}$[g] は以下のように書ける。

$$W_{HM,z} = \sum_{k \in HM} W_{k,z} = \sum_{k \in HM} \sum_{j \in z} N_k^j V^j w_k \tag{4}$$

ここで、初装荷炉心の物質ゾーン z での重核種の総重量を $W_{HM,z,0}$[g] とする（プログラム中の配列変数 `hm_w_total_mzone` に対応）。各サイクル初期の物質ゾーン毎の重核種の総重量は常にこの値になる。

物質ゾーン z におけるサイクル燃焼後の重核種重量の減少量 $\Delta W_{HM,z}$ は以下のように書ける。

$$\Delta W_{HM,z} = W_{HM,0,z} - W_{HM,z} = W_{HM,0,z} - \sum_{k \in HM} \sum_{j \in z} N_k^j V^j w_k \tag{5}$$

さて、燃料交換において燃料に新たに追加する、LWR から受け入れる MA 燃料に含まれる核種 k の全炉心平均での数密度を、ここで仮に $\Delta\tilde{N}_k$ とする（プログラム中では変数 `matden` に対応）。この数密度に対応する MA 燃料の、物質ゾーン z での総重量 $\tilde{W}_z$ は

$$\tilde{W}_z = \sum_k \Delta\tilde{N}_k V_z w_k \tag{6}$$

と書ける（プログラム中では変数 `hm_w_newfuel_mzone` に対応）。ここで V_z は物質ゾーン z の総体積を示す。サイクル燃焼後の物質ゾーン z における重核種重量の減少量が $\Delta W_{HM,z}$ であるため、同重量の MA 燃料が追加されなければならない。従って、物質ゾーン z において燃料交換で追加されるべき MA 燃料の核種 k の数密度 $\Delta N_{k,z}$ は

$$\Delta N_{k,z} = \Delta\tilde{N}_k \times \frac{\Delta W_{HM,z}}{\tilde{W}_z} \tag{7}$$

と書ける。

以上より、物質ゾーン z に属する媒質での燃料交換後の核種 k の領域平均数密度 $N'_{k,z}$（ただし $k \in HM$）は以下のように書ける。

$$\begin{aligned}
N'_{k,z} &= \frac{\sum_{j \in z} N_k^j V^j}{V_z} + \Delta N_{k,z} = \frac{\sum_{j \in z} N_k^j V^j}{V_z} + \Delta\tilde{N}_k \frac{\Delta W_{HM,k}}{\tilde{W}_k} \\
&= \frac{\sum_{j \in z} N_k^j V^j}{V_z} + \Delta\tilde{N}_{k,z} \times \frac{1}{\tilde{W}_k} \times \left(W_{HM,0,k} - \sum_{k \in HM} \sum_{j \in z} N_k^j V^j w_k \right) \qquad (8) \\
&= \frac{\sum_{j \in z} N_k^j V^j}{V_z} + \Delta\tilde{N}_{k,z} \times \frac{W_{HM,0,k}}{\tilde{W}_k} - \Delta\tilde{N}_{k,z} \times \frac{1}{\tilde{W}_z} \times \left(\sum_{k \in HM} \sum_{j \in z} N_k^j V^j w_k \right) \qquad (9)
\end{aligned}$$

ここで、右辺第一項は使用済み燃料中の重核種が物質ゾーン毎に混合されて再装荷されることを示し、第二項と第三項は新たに追加される MA 燃料に対応する。この式は行列形式で以下のように書ける。

$$\mathbf{N}' = \mathbf{R}\mathbf{N} + \mathbf{S} \tag{10}$$

ここで、$\mathbf{N}'$ は、燃料交換における数密度変化を示すベクトルであり、そのサイズは総領域（媒質）数と総核種数の積となる。$\mathbf{R}$、$\mathbf{S}$ はプログラム中ではそれぞれ `mat_reload`、`vec_reload` に対応する。

また、重核種の全炉心の総重量 W_{HM} は、$V^j w_k$ からなるベクトル $\mathbf{p}$ を用意することで $W_{HM} = \mathbf{p}^T\mathbf{N}$ として計算することができる。プログラム中では $\mathbf{p}$ は `vec_weight_cal` に対応する。

C.3 計算例

FRBurner を用いた ADS の燃焼計算の例として JAEA 設計の ADS を対象とした計算を行った。核データとして JENDL-3.2 ベースの CBZLIB を用いる、1 サイクルの燃焼ステップ数を 3 とするなど、可能な限り計算条件を同一とするものとした[14]。はじめに実効増倍率の計算結果を **Fig. 12** に示す。FRBurner の計算結果は 0.4%程度系統的に小さめに評価しているものの、燃焼に伴う推移は概ね再現できていると考えてよいだろう。次に、最大出力密度の計算結果を **Fig. 13** に示す。FRBurner の計算結果は数%程度系統的に大きめに評価しているもの[15] 、実効増倍率と同様に燃焼に伴う推移は概ね再現できていると考えてよいだろう。

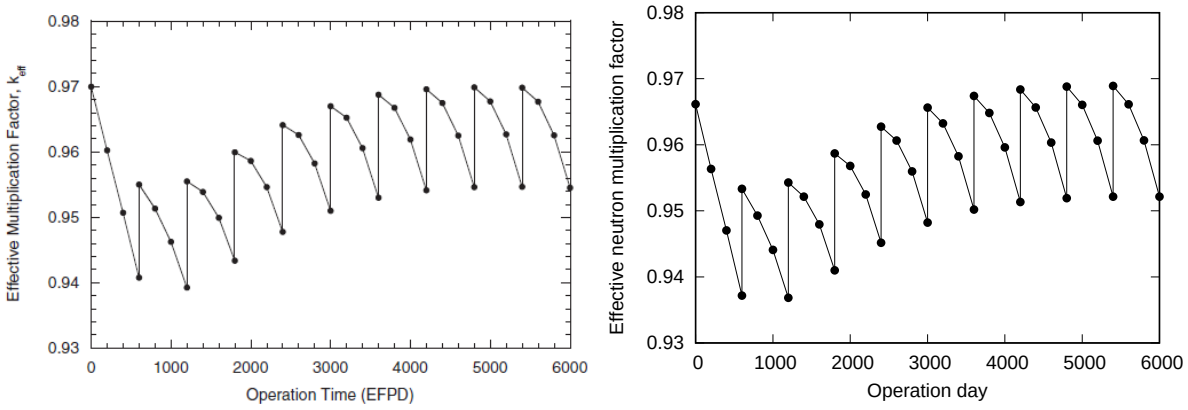

Fig. 12: 中性子実効増倍率の比較（左：文献より引引用、右：CBZ）

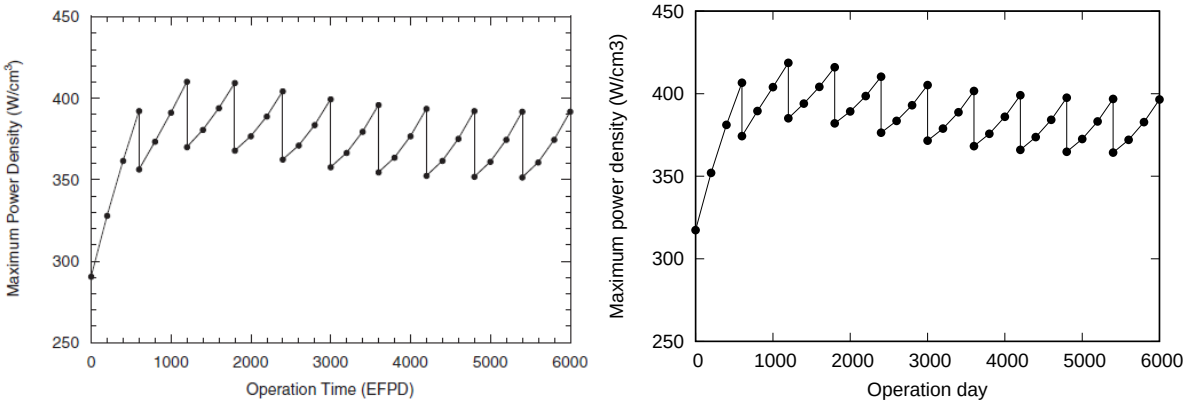

Fig. 13: 最大出力密度の比較（左：文献より引用、右：CBZ）

C.4 外部中性子源の設定

FRBurner を用いた ADS の燃焼計算では、計算結果としては体系の固有値が表示されるが、燃焼計算に用いる中性子束の空間、エネルギー分布は、核破砕中性子源を外部源とした固定源方程式によって得てい

[14]付録の D 節で述べている Predictor-corrector 法を用いることにより計算結果は有意に変わる。つまり、燃焼ステップ数が 3 では不十分ということになるが、計算条件を揃えるという意味で、このような設定で計算を行った。なお、Predictor-corrector 法はデフォルトでは off になっており、使用したい場合は FRBurner クラスの RunADS メソッド中に変更を施す（メソッド冒頭部の boolean 変数の pc_cal を true にする）必要がある。

[15]実効増倍率を 0.4%程度過小評価しているため、未臨界度が深くなり、出力ピークがより立ちやすい条件となっている。

る。核破砕中性子源自体を計算することは CBZ では出来ないため、何らかの計算コードで求めた外部中性子源分布を用いることを想定している。現在の版では、JAEA が提案している鉛ビスマス冷却の ADS を想定した外部中性子源のみが利用可能となっている。

ソースで該当しているのは、クラス `FRBurner` のメソッド `ExternalSourceReading` と `ExternalSourceSetting` である。ソースを眺めてもらえば分かるが、現時点では、JAEA 提案の ADS の核計算を想定しており、径方向、軸方向メッシュ数がそれぞれ 38、54 に、またエネルギー群数も 70 に固定されている。従って、サンプル入力として与えられている ADS 体系以外の計算を行いたい場合は、これらのメソッドの修正が必要となる。

D 燃焼サイクルのステップ分割数

高速炉の燃焼計算では、サイクル中の中性子実効増倍率の時間推移を評価する目的で、燃焼サイクルをいくつかのステップに分割するという操作を行うことがある。分割されたステップ毎に実効増倍率と中性子束分布が計算され、それに基づいてステップ中の燃焼計算を行われるが、高速炉では燃焼期間中の中性子束の空間分布の変動はそれほど大きいものではないため、ステップ分割が計算結果に大きく影響することはない。

一方、ADS では、中性子束の空間分布は未臨界度に強く依存する。従って、燃焼サイクル中で未臨界度（中性子実効増倍率）が大きく変動する場合には、中性子束の空間分布もそれに伴って変動することになり、燃焼サイクルのステップ分割数が計算結果に大きく影響する可能性がある。

Fig. 14 に高速炉（もんじゅ）と ADS の燃焼に伴う中性子実効増倍率を示すが、ADS では計算結果が燃焼ステップ分割数に大きく影響していることが分かる（ただし、燃焼サイクルの長さが両者で異なることから、それが原因とも考えられる）。

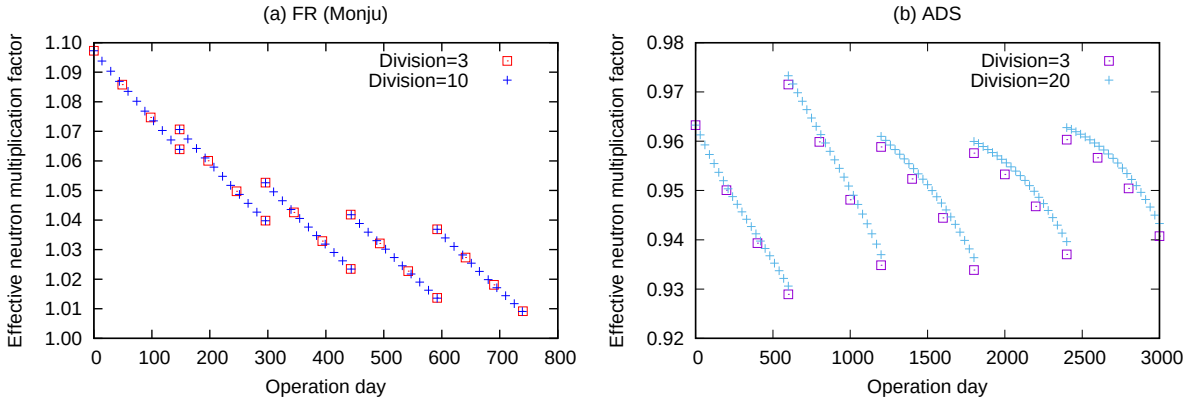

Fig. 14: 中性子実効増倍率の燃焼ステップ分割数に対する依存性

燃焼ステップ分割数の増加は計算負荷の増加に直結するため、ADS の燃焼計算のためのメソッド RunADS に予測子-修正子法（Predictor-Corrector 法、PC 法）を実装した。実装したアルゴリズムは以下の通りである。

1. 燃焼ステップ初期の組成で中性子束分布を計算し、出力により規格化する。
2. 上記ステップ 1 で計算した中性子束分布に基づき燃焼計算を行い、燃焼ステップ末期の組成を計算する（Predictor 計算）。
3. 燃焼ステップ末期の組成で中性子束分布を計算し、出力により規格化する。
4. 全てのメッシュ、エネルギー群の中性子束を、上記ステップ 1 とステップ 3 で計算された中性子束の平均値に置き換え、出力により規格化する。
5. 上記ステップ 4 で計算した中性子束分布に基づき燃焼計算を行い、燃焼ステップ末期の組成を再計算する（Corrector 計算）。

ADS の燃焼計算について、PC 法を適用しない場合とした場合とで実効増倍率を比較した。参照解は、サイクルを 20 のステップに分割して得られた結果とした（ただし PC 法は適用していない）。結果を **Fig. 15** に示すが、PC 法の適用により、少ないステップ分割数で高い精度の計算が可能となっていることが分かる。

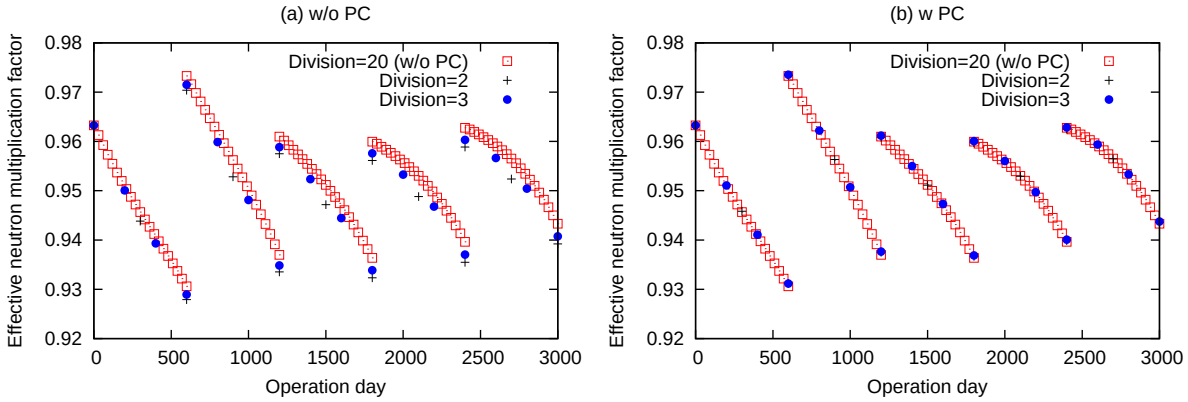

Fig. 15: 中性子実効増倍率の燃焼ステップ分割数に対する依存性（予測子-修正子法の効果）

E 実効断面積の計算について

媒質の多群実効断面積は核種組成に依存して決まるため、核種数密度が変動する燃焼計算においては、燃焼ステップ毎に実効断面積を計算する必要がある。ただし、高速炉の解析においては、核特性計算結果の共鳴実効断面積への依存性は熱中性子炉ほど大きくはないため、例えば燃焼開始時点の組成で実効断面積を求め、燃焼期間中はそれを流用する、という方法がしばしば採用される。FRBurner でも同様の扱いをしている。

FRBurner では、燃焼計算に先立って行われる「PreCalculation」メソッドにおいて、内側炉心、外側炉心、径ブランケット、軸ブランケットの初期組成に対して実効断面積を計算し、それがその後の燃焼計算で用いられる。なお、PreCalculation メソッドでは、初期組成での中性子拡散固有値方程式[16] を解いて各組成領域での平均中性子束を求めて、それを各媒質データ（Medium クラスのインスタンス）に中性子束データとして持たせている。燃焼計算においては各燃焼サブステップ毎にその時点の数密度と実効断面積から巨視的断面積を計算させているが、その際の平均核分裂スペクトルの計算でこの各媒質データが持つ中性子束データが荷重関数として用いられる。

なお、燃焼計算中は、各媒質が持つ中性子束データは逐次更新されていく。燃焼計算中に実効断面積を再計算するように変更した場合は、**媒質がもつ中性子束データとしてライブラリの中性子束データが上書きされてしまう**[17]ことに注意が必要である。

ADS の燃焼計算を、通常の方法で行った場合（FRBurner クラスの RunADS メソッドを用いて行った場合）と、サイクル毎に新たな FRBurner クラスのインスタンスを生成して計算させるという方法で行った場合（つまり燃料交換を FRBurner クラスを用いずに行った場合）について、3 サイクルまでの実効増倍率を比較した（燃焼ステップ分割数は 3)。なお、この 2 つの方法の間には、燃料交換に伴う炉心燃料数密度の計算に微妙な差異が存在することもあり、以下の 4 ケースの計算を実施した。

- Case 1：通常の方法
- Case 2：サイクル毎に FRBurner クラスのインスタンスを生成
- Case 3：Case 2 と同様だが、各サイクルの初期組成は Case 1 のものと同一と設定
- Case 4：Case 2 と同様だが、2 サイクル目の初期組成を Case 1 のものと同一と設定

Case 1 では 1 サイクル初期に計算した実効断面積がその後の計算で利用されるが、その他では各サイクル初期にその時点の数密度に基づいて実効断面積が再計算される。計算結果を **Table 1** に示す。なお、1 サイクル目は同一の計算結果となるため、2、3 サイクルのみについて示す。Case 1 と Case 3 では各サイクル初期の数密度が同一となるため、実効断面積の違いのみが抽出可能となるが、3 サイクル目で 10pcm 程度の差異が生じていることが分かる。また、2 サイクル目の初期組成が異なる Case 2 と Case 3（および 4）とで、2 サイクル目の計算結果がほぼ同様であることから、2 サイクル目の Case 2、3、4 の計算結果が Case 1 と異なるのは実効断面積計算方法の違いに起因しているということが言える。すなわち、3 サイクル目において、Csae 2 および 4 は Case 1 との間に比較的大きな差異が見られているが、これは前述した燃料交換に伴う炉心燃料数密度計算の微妙な差異によるものではなく、実効断面積計算方法の違いに起因していることになる。実際に、各サイクルの最初の燃焼ステップで実効断面積を再計算させると、Case 1 と Case 2 の実効増倍率の差異は 0.00003 以内に収まり、上記の結論が適切であることが確認した。

[16] ADS の計算では、厳密には固定源の方程式を解くべきではあるが、その影響は小さいことを確認している。
[17] すなわち、実効断面積の計算精度は向上するが、平均核分裂スペクトルの計算精度が低下する可能性があることになる。

Table 1: Effective neutron multiplication factors in ADS burnup calculations

Cycle	Step	Case 1	Case 2	Case 3	Case 4
2	1	0.96855	0.96858 (+0.00003)	0.96857 (+0.00002)	0.96857 (+0.00002)
	2	0.95754	0.95758 (+0.00004)	0.95758 (+0.00004)	0.95758 (+0.00004)
	3	0.94631	0.94632 (+0.00001)	0.94631 (+0.00001)	0.94631 (+0.00001)
	4	0.93345	0.93343 (-0.00002)	0.93343 (-0.00002)	0.93343 (-0.00002)
3	1	0.95752	0.95741 (-0.00011)	0.95753 (+0.00001)	0.95741 (-0.00011)
	2	0.95140	0.95131 (-0.00009)	0.95139 (-0.00001)	0.95131 (-0.00009)
	3	0.94378	0.94367 (-0.00011)	0.94372 (-0.00006)	0.04367 (-0.00011)
	4	0.93338	0.93325 (-0.00013)	0.93328 (-0.00010)	0.93325 (-0.00013)