JAEA-Data/Code
2010-030

JAEA-Data/Code

次世代炉心解析システム MARBLE の開発

Development of the Next Generation Reactor Analysis Code System, MARBLE

横山 賢治　巽 雅洋　平井 康志　兵頭 秀昭
沼田 一幸　岩井 武彦　神 智之　羽様 平
長家 康展　千葉 豪　久語 輝彦　石川 眞

Kenji YOKOYAMA, Masahiro TATSUMI, Yasushi HIRAI, Hideaki HYOUDOU
Kazuyuki NUMATA, Takehiko IWAI, Tomoyuki JIN, Taira HAZAMA
Yasunobu NAGAYA, Go CHIBA, Teruhiko KUGO and Makoto ISHIKAWA

原子力基礎工学研究部門
核工学・炉工学ユニット

Division of Nuclear Data and Reactor Engineering
Nuclear Science and Engineering Directorate

March 2011

Japan Atomic Energy Agency　日本原子力研究開発機構

本レポートは独立行政法人日本原子力研究開発機構が不定期に発行する成果報告書です。
本レポートの入手並びに著作権利用に関するお問い合わせは、下記あてにお問い合わせ下さい。
なお、本レポートの全文は日本原子力研究開発機構ホームページ（http://www.jaea.go.jp）
より発信されています。

独立行政法人日本原子力研究開発機構　研究技術情報部　研究技術情報課
〒319-1195　茨城県那珂郡東海村白方白根 2 番地 4
電話 029-282-6387, Fax 029-282-5920, E-mail:ird-support@jaea.go.jp

This report is issued irregularly by Japan Atomic Energy Agency
Inquiries about availability and/or copyright of this report should be addressed to
Intellectual Resources Section, Intellectual Resources Department,
Japan Atomic Energy Agency
2-4 Shirakata Shirane, Tokai-mura, Naka-gun, Ibaraki-ken 319-1195 Japan
Tel +81-29-282-6387, Fax +81-29-282-5920, E-mail:ird-support@jaea.go.jp

© Japan Atomic Energy Agency, 2011

次世代炉心解析システムMARBLEの開発

日本原子力研究開発機構
原子力基礎工学研究部門 核工学・炉工学ユニット
横山 賢治、巽 雅洋＊1、平井 康志＊1、兵頭 秀昭＊1、沼田 一幸＊2、岩井 武彦＊2、
神 智之＊2、羽様 平、長家 康展、千葉 豪、久語 輝彦、石川 眞

(2010年12月17日受理)

高速炉核特性解析のための次世代炉心解析システムMARBLEを開発した。MARBLEは、これまでにJUPITER標準解析手法と呼ばれる高速炉詳細解析手法として開発されてきたJOINT-FR、SAGEP-FRと呼ばれる解析システム（従来システム）の後継である。MARBLEでは従来システムに含まれている解析コードを修正せずに用いることができるため、従来システムと同等の解析機能を有する。これに加えて、燃料交換、制御棒操作等のモデルを導入することで燃焼を伴う高速炉実機の核特性解析に関する機能を向上させている。また、燃焼計算ソルバーや核設計精度評価ソルバーを新規に開発しており、従来システムにはなかったクリロフ部分空間法に基づく燃焼計算や拡張バイアス因子法に基づく核設計精度評価等の機能も追加されている。

MARBLEの開発では、柔軟性・拡張性等のソフトウェア品質の向上やモジュール化による検証性の向上、共同開発への対応の観点からオブジェクト指向技術を採用した。また、スクリプト言語とシステム言語による二階層システムと呼ばれるソフトウェア構造を適用している。この結果として、MARBLEは一定の入力を受けて出力を返すような独立した解析コードではなく、解析システムを構築するための部品の集まり（フレームワーク）となっている。一方で、MARBLEは構築済みの解析システムも有しており、従来システムに相当する高速炉核特性解析システムSCHEME、高速炉実機燃焼解析システムORPHEUSを利用することができる。

原子力科学研究所（駐在）：〒319-1195 茨城県那珂郡東海村白方白根2番地4
＊1 原子燃料工業株式会社
＊2 株式会社NESI

Development of the Next Generation Reactor Analysis Code System, MARBLE

Kenji YOKOYAMA, Masahiro TATSUMI* [1], Yasushi HIRAI* [1], Hideaki HYOUDOU* [1], Kazuyuki NUMATA* [2], Takehiko IWAI* [2], Tomoyuki JIN* [2], Taira HAZAMA, Yasunobu NAGAYA, Go CHIBA, Teruhiko KUGO and Makoto ISHIKAWA

Division of Nuclear Data and Reactor Engineering
Nuclear Science and Engineering Directorate
Japan Atomic Energy Agency
Tokai-mura, Naka-gun, Ibaraki-ken

(Received December 17, 2010)

A next generation reactor analysis code system, MARBLE, has been developed. MARBLE is a successor of the fast reactor neutronics analysis code systems, JOINT-FR and SAGEP-FR (conventional systems), which were developed for so-called JUPITER standard analysis methods. MARBLE has the equivalent analysis capability to the conventional system because MARBLE can utilize sub-codes included in the conventional system without any change. On the other hand, burnup analysis functionality for power reactors is improved compared with the conventional system by introducing models on fuel exchange treatment and control rod operation and so on. In addition, MARBLE has newly developed solvers and some new features of burnup calculation by the Krylov sub-space method and nuclear design accuracy evaluation by the extended bias factor method.

In the development of MARBLE, the object oriented technology was adopted from the viewpoint of improvement of the software quality such as flexibility, expansibility, facilitation of the verification by the modularization and assistance of co-development. And, software structure called the two-layer system consisting of scripting language and system development language was applied. As a result, MARBLE is not an independent analysis code system which simply receives input and returns output, but an assembly of components for building an analysis code system (i.e. framework). Furthermore, MARBLE provides some pre-built analysis code systems such as the fast reactor neutronics analysis code system, SCHEME, which corresponds to the conventional code and the fast reactor burnup analysis code system, ORPHEUS.

Keywords: Fast Reactor, Neutronics, Nuclear Design, Object-oriented Technique, MARBLE, SCHEME, ORPHEUS

---

* [1] Nuclear Fuel Industries, Ltd.

* [2] NESI incorporation

目 次

Contents

List of Tables

List of Figures

1. はじめに

近年、高速炉核設計に関する研究開発に対するニーズが多様化し、解析対象がより複雑化しているため、解析対象に応じて工学的なモデルや解析手法を柔軟に変更したり、新たに開発した手法を取り入れて容易に拡張したりできることが、解析コードに求められる重要な要素となってきている。また、コード検証と妥当性確認（V&V: Verification and Validation）の重要性も年々高まってきており、解析コードの検証性や最新知見を迅速に導入できることも重要な要素となってきている。このような背景を受けて、原子力機構では柔軟性、再利用性、拡張性、保守性といった性能を備えた「工学系モデリング言語としての次世代解析システム（以下、次世代解析システム）」に関する検討を行ってきた。これまでに、課題と要素技術の調査[1)]、プロトタイプによる検討[2,3)]、オブジェクト指向言語を用いた中性子拡散ソルバーの開発[4)]、既存コードシステムからの移行方法の検討[5)]、オブジェクト指向言語を用いた炉定数調整・核設計精度評価ソルバーの開発[6)]等を行った。また、これらの検討で得られた要素技術を使って燃焼感度解析コードのシステム化整備[7–9)]を行った。

本研究では、これまでの検討で得られた次世代解析システムの要素技術を用いて、高速炉核特性解析のための次世代解析システムとして、次世代炉心解析システム MARBLE (Multi-purpose Advanced Reactor Physics Analysis System Based on Language of Engineering）を開発した。MARBLE は高速炉の臨界実験解析をベースに開発されてきた JUPITER 標準解析手法と呼ばれる高速炉核特性の詳細解析手法[10,11)]に基づく解析システムである。この詳細解析手法を実現する解析システムとしては既に JOINT-FR や SAGEP-FR と呼ばれる解析システム（以下、従来システム[12)]）が存在しているが、MARBLE はこれらの従来システムの後継となることを目指して開発されたものである。

JUPITER 標準解析手法自体は 1997 年の時点でほぼ完成しており、その後の大きな変更点としては 2004 年に公開された SLAROM-UF コード[15,16)]を用いた超微細群補正が加えられたことが挙げられる。MARBLE は、この超微細群補正を含む JUPITER 標準解析手法として開発された「高速炉核特性詳細解析手法」を高速炉実機に対しても容易に適用できるようにし発展させていくための基盤技術となることを目標として開発された。このため、MARBLE は将来にわたって継続的に解析手法やモデルを高度化していけるように拡張性や柔軟性を重視しており、このための方策として次世代解析システムの基本概念を適用して設計・開発した。

MARBLE の特徴としては以下の３つの項目に要約することができる。

① 現代的なソフトウェア開発手法を導入した解析システムの再設計・再構築
② 従来システム JOINT-FR の基本解析機能の再現と利便性の向上
③ 燃料交換や燃焼を伴う実機への高速炉核特性詳細解析手法の適用

これらの項目は、それぞれ、①MARBLE フレームワーク、②高速炉核特性解析システム SCHEME、

③高速炉実機燃焼解析システム ORPHEUS として実現されている。本報告書ではこれらのフレームワーク、解析システムの内容と使い方について説明する。

第 2 章では MARBLE の基本概念と設計について述べ、MARBLE が採用しているフレームワークの概要について説明する。続いて、第 3 章では MARBLE のフレームワークについて詳しく説明する。第 4 章では、MARBLE フレームワークを使って開発された 2 つの解析システムについて説明する。第 5 章では MARBLE の使い方を手順を追って説明することで、MARBLE を使い始める際に必要となる情報をまとめた。なお、第 5 章は他の章から独立させてあるので、MARBLE をすぐに利用したい場合には第 5 章から読み始めることができる。その他、MARBLE を利用する上で必要となるリファレンス的な情報は付録にまとめた。最後に第 6 章で全体を総括する。

2. 基本概念と設計

本章では次世代炉心解析システム MARBLE を開発することになった背景やそこから導かれた基本概念と設計について述べる。MARBLE は工学系モデリング言語としての次世代解析システムとして検討された新しい解析システムの概念に基づいた高速炉用の炉心解析システムである。本章のまとめとして工学系モデリング言語としての次世代解析システムとの関係を整理する。

2.1 背景

最初に次世代炉心解析システム MARBLE の開発を行った背景として、高速炉詳細核特性解析手法とその手法を実現するための従来システムの概要を述べる。

2.1.1 高速炉詳細核特性解析手法

現在の高速炉核特性詳細解析手法は、原子力機構の前身である動力炉・核燃料開発事業団と米国エネルギー省（DOE）との共同研究として米国の ZPPR 臨界実験装置を使って行われた大型高速炉の臨界実験である JUPITER の解析評価を進める中で、発展・確立してきたものであり「高速炉核設計基本データベース」の一部として集大成されている [10,11]。その後、解析対象は JUPITER 臨界実験だけにとどまらず、日本、フランス、ロシアの臨界実験データや高速炉実機の性能試験データ等を取り込み、拡張された高速炉核設計基本データベースは統合炉定数 ADJ2000R [13,14] の開発のための基本データとなっており、高速炉設計研究で利用されている。

この高速炉核特性詳細解析手法は決定論的な解析手法に基づいている。近年、連続エネルギーモンテカルロ法が数値解法の参照解とされる場合が増えているが、複雑な燃料交換や燃料組成分布の評価が必要な高速炉核設計においては、今後も決定論的な解析手法が主体であり続けると予想される。

高速炉核特性詳細解析手法は、現状でも解析手法として十分に確立したものとなっているが、前述のように近年も超微細群格子計算コード SLAROM-UF の導入等により手法は高度化されており、今後も解析手法やモデルを改良していく必要がある。

2.1.2 従来の高速炉核特性解析システム

前節の高速炉核特性詳細解析手法を実現するものとして、最終的に JOINT-FR[17–27]、SAGEP-FR[28,30–32] と呼ばれることになった複数の公開解析コード群からなる解析システム（前述の従来システム）が存在している。この従来システムは、1970 年代に基本設計された JOINT[18] と呼ばれるインターフェースコードシステム [18] をベースとしており、複数の解析コード群の入力データ

や出力データを変換しながらコードを連携して動作させることが可能なものとなっている。

高速炉の核特性解析の際に重要かつ膨大となるデータとして中性子と原子核の反応確率を表す実効断面積があるが、従来システムではこの実効断面積の取扱いにはPDSファイルと呼ばれる独自のファイル形式が採用されている。このPDSファイルは解析コードから完全に独立しており、解析コード群のモジュール性を高める上で非常に優れた基本設計となっていたと考えられる。しかしながら、モジュール性を確保するのが難しいFORTRAN77以前のFORTRANと機械語で実装された従来システムは、設計当時の計算機環境を考慮して設定された核種数や反応数の制限等もあり、長年の保守・改良の中で複雑化し、品質を保証しつつ保守・改良を続けていくのが事実上困難な状況になってきていた。また、不自然な拡張オプションの追加等により、ユーザがシステム上の制限を理解して慎重に入力データを作成しなければ正しく計算できないようなケースも増えてきた。

このような問題は計算機利用が比較的早く始まった炉物理解析の分野では特別なことではないようであり、特にフランスでは同様の問題意識から軽水炉・高速炉の炉心解析システムを全面的にオブジェクト指向言語C++で書き換えて統合化するプロジェクト[34,35]が進行している。

一方で、従来システムはその発展とともに大きくなっており、ごく少数の開発者だけによる開発には限界が生じてきている。また、各ユーザが独自に作成した周辺ツールも多く存在するがそれらのツールを統合する仕組みがないため、各ユーザが同じようなツールを繰り返し作ってしまうような非効率さも生じている。

2.2 要件

前述のような背景を踏まえて、以下のような要件を満たす新しい解析システムとしてMARBLEを開発することを目標とした。

- 検証された従来システムの結果を再現できること。
- 今後のニーズに柔軟に対応し高度化し続けられること。
- 複数の開発者による共同開発に対応できること。
- ユーザの入力データ作成に関する負担を低減すること。

従来システムは既に多くの関係者に利用されており、高速炉核特性詳細解析手法を適用する際の解析システムとしての信頼を得ている。このため、従来システムの結果を再現できることは既存ユーザ、新規ユーザのどちらにとっても非常に重要な意味を持つと考えた。その上で今後の拡張に対応できるようなソフトウェア技術上の高度化を実現することを目標とした。また、複数の開発者による共同開発に対応することは、開発速度の向上だけでなく、開発・保守管理が継続されることへの信頼感、トレーサビリティの確保による信頼性向上のためにも重要と考えた。ユーザの入力データ作成に関する負担の低減は単なる作業効率だけの問題ではなく、計算結果の品質保証に係わる問題である。

2.3 基本概念

MARBLE は、次世代解析システムに対する課題と要素技術調査 [1] の段階で検討された「二階層システム」の概念に基づいて開発されている。二階層システムでは、図 2.3.1 に示すように、(1) システムを特定のアルゴリズムやモデルに基づいて物理的な計算を実行する「計算層」と、(2) これらの計算層に含まれる要素・機能をつなぎ合わせて使う「制御層」に分けて考える。このように２つの階層に分けることで、解析に必要な機能とデータを部品化しておくことが可能となる。このことにより、必要に応じて解析手順を組み立てることが可能となるため様々な解析のニーズに対応することができる。更に、組み立てた解析手順は再部品化でき、システムの継続的な高度化を支援することができる。

このように MARBLE では二階層システムを採用しているため、ひとつひとつの部品の持つ機能は限られており、従来システムでいうところの「解析システム」を作り上げるための基盤技術（フレームワーク）と考えることができる。また、部品だけでなく、部品を使って組み立てられた解析システム自体も MARBLE に含まれていることになる。

二階層システムの概念は、次世代解析システムに関する要素技術の調査 [1] で提案した「工学系モデリング言語」の考え方にも対応している。工学系モデリング言語とは、原子炉の解析で必要になる物理量・解析手法等の概念を人間が理解しやすく記述でき、かつ、計算機にも実行できるプログラムやダイアグラムのことである。計算層に含まれる部品の持つ意味を物理量や解析手法に対応させておくことで、制御層で記述されたプログラムは工学系モデリング言語として機能することになる。

また、二階層システムの概念はプログラミング言語を「システム言語」と「スクリプト言語」に分類する考え方 [36] とも対応している。実行速度が重視される計算層の開発には主に FORTRAN や C++ 等のシステム言語を適用し、要素を結合することが重要な制御層の開発には主に Python 等のスクリプト言語を適用する。このようにすることで、プログラミング言語の特徴を活かしてシステムを効率的に開発することができるとともに、機能を効果的に再利用する仕組みを作ることができる。

2.4 基本設計

2.4.1 オブジェクト指向技術

MARBLE の開発では、従来システムでは利用されていなかったオブジェクト指向技術を導入することにした。これは、オブジェクト指向技術によってもたらされるソフトウェアの内的品質（モジュール性、プログラムの読みやすさ）の向上に伴うソフトウエアの外的品質（正確さ、頑丈さ、拡張性、再利用性、互換性）の向上 [37] を図ることが大きな動機となっている。また、工学系モデリング言語を実現するための要素技術として、物理上の概念に基づいてプログラムを作成することが必要であり、この点でもオブジェクト指向技術の導入が必要であった。

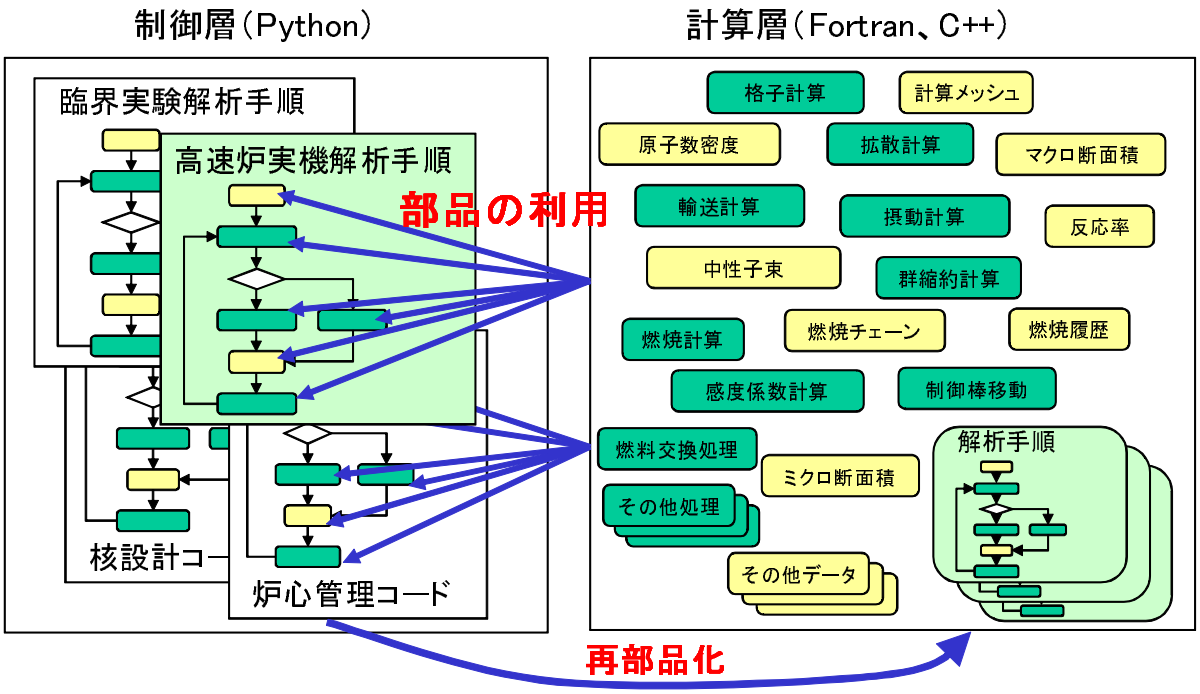

図 2.3.1　MARBLE の基本構造

また、オブジェクト指向技術に基づくモジュール化は、モジュール単位の検証（Verification）を可能とするため、近年、原子力関連の解析コードに求められている検証性に対して強みを持つことができると考えられる。他にも、複数の開発者による共同開発に対応するためには、作業の分担が必要があり、開発者が担当しているプログラムどうしが干渉しないようにする必要がある。このためにもモジュール化の技術が必要である。これらの観点からもオブジェクト指向技術の採用が不可欠であると判断した。

2.4.2　スクリプト言語とシステム言語

MARBLE の基本概念である二階層システムを構成するためにはスクリプト言語の採用が必要になる。スクリプト言語としては、一般的なテキストベースのプログラミング言語だけでなくダイアグラムを利用するようなものも考えられる。しかしながら、開発の柔軟性や効率性の点から一般的なテキストベースのスクリプト言語の方が良いと考えた。これらの観点から前節で述べたオブジェクト指向技術とスクリプト言語の両方の特性を持つオブジェクト指向スクリプト言語を採用することにした。

結果として MARBLE の開発では、スクリプト言語として Python[38]、システム言語として FORTRAN と C/C++を採用した。スクリプト言語の候補としては Python の他に Ruby、OCaml 等があったが、スクリプト言語として Python を採用した主な理由は以下のようなものである。

- 書き方が統一されるように設計されており習得しやすい。
- 手続き型とオブジェクト指向型の両方に対応している。
- 完成度の高い数値計算ライブラリが整備されている。
- 数値解析分野への適用例が豊富で信頼性が高い。
- 仕様・実装が公開されており自由に入手できる。

MARBLE のスクリプト言語はユーザが直接利用する可能性があるため、FORTRAN しか使ったことのないユーザにとって比較的習得しやすくオブジェクト指向を必ずしも強制しないという特徴を持つ点を重視した。また、開発開始時点で既に FORTRAN と親和性の高い数値計算ライブラリ[1]と FORTRAN との結合ツール[2]が存在していた点も大きな理由となった。その他、Python は個人が開発したオープンソースから始まっているが、Python Software Foundation という非営利団体によって管理されており、今後の開発の継続性が確保されている点も重視した。

なお、次世代解析システムにおける二階層システムとスクリプト言語・システム言語の役割のより詳細な内容については、文献 [1)] を参照することができる。

2.4.3 解析コードのカプセル化

MARBLE の目標のひとつとして、解析手法そのものを高度化し続けていけるということがあるので、本来、すべての機能を MARBLE の基本概念にそってオブジェクト指向技術が適用できる Python または C++で書き直すのが良いと考えられる。しかしながら、時間と費用の制約から従来システムに含まれるすべての解析コードを最初からすべて書き直すのは非現実的と判断した。また、従来システムの結果を再現する必要があることから、従来システムに含まれる解析コードの再利用は避けられないと考えた。一方で、FORTRAN で書かれたコードを段階的にオブジェクト指向言語で書き換える方法もある程度確立できていた [5)]。

このため、最初の開発ステップでは従来システムに含まれる解析コードを書き換えずに基本的にすべて再利用する方針を採用した。解析コードをカプセル化する技術としては、燃焼感度解析コードのシステム化整備 [8,9)] で PSAGEP コードシステムを開発した際に適用した技術を発展させることにした。具体的には、入力ファイルの書き出しのみに対応していた PSAGEP のカプセル化技術を、MARBLE のカプセル化技術では出力ファイル書き出しと読み込みの両方に対応させた。更に、PSAGEP のカプセル化技術では入力ファイルの生成しか考慮していなかったが、MARBLE のカプセル化技術では、出力ファイルに含まれるデータを別の解析コードや後処理プログラムの入力データとして利用することを想定して、出力ファイルにも対応させた。これにより解析コードのデータの受け渡しの順番に関する制限を完全になくすことができ、従来システムの解析コードがオブジェクト指向言語で記述されているかのように完全に制御できるようになった。

ただし、従来システムの解析コードはファイルをインターフェースとして用いているため、MARBLE のカプセル化技術を実現するためには、FORTRAN のファイル入出力処理を Python 上で模

[1] Numeric、Numarray が存在しており、MARBLE で採用することになった Numpy[39)] の開発も始まっていた
[2] 当時不安定バージョンの f2py[40)] が存在しており、最終的に Numpy に統合された。

擬する必要があった。しかしながら、Python にはそのような機能がなかったため、FORTRAN のファイル入出力処理を Python 上で模擬するライブラリを独自に開発する必要があると判断した。このため、MARBLE の一部として FortranUtils と呼ばれる Python モジュールを独自に開発した（付録 G 参照）。

2.4.4 ユーザマニュアル

ソフトウェアはハードウェアに比べて変更が格段に容易なことから、新たなニーズへの対応や問題点修正のために書き換えられていくことは避けられない事実である。このため、ソフトウェアのマニュアルもソフトウェア本体に追従して更新されていく必要がある。このような観点から、MARBLE では基本的なマニュアルはなるべくソースコードの中に記述し、ユーザはオンラインでこのマニュアルを参照することを原則としている。この方法にはソースコードの修正とマニュアルの修正を同時に進めやすいという利点がある。また、ユーザは必ずソースコードとマニュアルをセットで入手することになるので、実装とマニュアルの不整合が生じにくいという利点もある。

なお、Python は基本機能としてソースコード内部にコメントとしてマニュアルを記載する機能を持っており、MARBLE ではこの機能を利用している。このため、MARBLE に含まれるクラスやモジュールの詳細についてはこのマニュアルを参照することができる。後述するように MARBLE はインタラクティブに利用することも可能であるため、この方式を採用していることでオンラインマニュアルのような使い方も可能となっている。

2.4.5 ユーザインターフェース

次世代解析システムの要素技術の調査ではデータの保存形式として XML(Extensible Markup Language) 等の検討を行ったが、MARBLE では更に調査を進め、XML を簡略化した YAML と呼ばれるファイル形式 [41] を採用した。YAML は名前付けと字下げ（インデント）による構造化が可能なファイル形式であり、人が見たときの読みやすさとコンピュータ処理のしやすさの両方を兼ね備えている。Python で扱うデータ構造との親和性が良いので YAML 形式ファイルの処理プログラムを書くのは容易である。また、YAML 形式ファイルは通常のエディタを使って簡単に編集することができる。

MARBLE でユーザがデータを準備する場合には基本的に YAML 形式ファイルを使うようにしている。また、工学系モデリング言語の観点からは場合によっては Python が持つ一部の機能を直接インターフェースとして用いることも想定している。

2.4.6 開発プロセス

MARBLE の開発では、オブジェクト指向技術やスクリプト言語といった技術だけでなく、現代的なソフトウェア開発プロセスを積極的に導入した。また、オープンソース開発で用いられてい

る手法も積極的に取り入れ、開発過程のコードをすべて記録・管理することが可能なバージョン管理システム（Subversion[42)]）や複数の開発者がインターネット経由で開発課題の管理やコミュニケーションを行うことが可能な問題点追跡システム（Trac[43)]）等を利用することにした。

開発プロセスとしてはアジャイルソフトウェア開発[44)]と呼ばれる開発手法を積極的に導入した。この開発手法は反復型開発と呼ばれる手法の一種であり、計画、要求分析、設計、実装、テスト、文書化といった一連の作業をなるべく短い期間で行う。この一連の作業を繰り返し行うことで柔軟かつ迅速にソフトウェアを開発する。MARBLE の開発では特にテスト駆動開発[45)]と呼ばれる開発手法を重点的に採用した。テスト駆動開発では、プログラムを作成する前に動作確認のためのテストを書き、そのテストが正常に動作するように実装するという作業を繰り返す。この手法では、プログラムの外部的振る舞いを保ったままで、内部の構造を改善していくリファクタリング[46)]と呼ばれる作業を繰り返しながら開発を進めていく。このリファクタリングを行うことで、コードの品質を内部から高めていくことができるという特徴を持っている。また、先にテストを書くことで自然と分析・設計を行うことになるとともに、開発するプログラムの機能を文書化することにもなる。この開発手法に厳密に従うとプログラムが常に動作する状態を保つことができる。

この手法は燃焼感度解析コードのシステム化整備でも利用され、当時の経験から、複数の開発者による同時開発を効率的に進めるためや従来システムの動作を保証する観点からも非常に有効であることが分かっていた。MARBLE はこれらの開発手法と不可分なものとなっている。

2.5 まとめ

次世代炉心解析システム MARBLE は、基本的に工学系モデリング言語に基づく次世代解析システムの開発で検討された基本概念を採用している。本章のまとめとして、当時抽出された共通課題について MARBLE がどのような解決方法を採用したかを整理することで本章のまとめとする。

2.5.1 共通ニーズ

当時議論された共通課題としてのニーズは以下の 3 点である[1)]。

① 自作コードの組み込み
② 参照解としての利用
③ 異なる専門分野間での統合解析

自作コードの組み込みは、MARBLE では FORTRAN のコードについてはカプセル化技術を利用することで実現している（付録 C 参照）。また、C++のコードについても、既存の公開ライブラリと補助ツールを提供することで自作コードを組み込めるような仕組みを提供している[47)]。

参照解としての利用という点については、これまで臨界実験解析で妥当性確認（Validation）が行われてきている従来システムを再利用することで実現できていると考えられる。また、テスト

駆動開発の手法を適用することでモジュール単位で計算結果を完全にすることを確認しつつ新しいシステムに移行したため、段階的なコード検証（Verification）も行われたと考えられる。

最後の点については現状の MARBLE は高速炉の炉心解析のみを対象としているため、異なる専門分野間での統合解析については次期開発ステップ以降に取り組むべき課題である。

2.5.2 機能上の共通課題

機能上の共通課題として挙げられたのは以下の３点である。

① 物理的意味を伴った中小規模モジュールの集合体としてのシステム
② システムの信頼性確保
③ モジュラー型コードシステム

１点目については、MARBLE フレームワークでオブジェクト指向技術を適用し炉心解析で用いられる概念に対応する多くのクラスを定義することで実現している。２点目については、テスト駆動開発手法を採用することで実装コードとほぼ同量の膨大な自動実行可能な単体テストを同時に作ることで信頼性を確保している。３点目については、解析に必要な機能やデータを部品に分解することとスクリプト言語による組み合わせによりある程度実現されたと考えることができる。

2.5.3 実装上の共通課題

実装上の共通課題として挙げられたのは以下の５点である。

① 異なるプログラミング言語間での結合
② 自律したモジュール（オブジェクト指向技術の適用）
③ ネットワーク対応・携帯性
④ オープンシステムによる共同開発
⑤ モジュールの結合関係の記述

１点目については、オブジェクト指向スクリプト言語 Python を中核に置き、FORTRAN、C++との連携技術を採用・確立することで、解析コードの開発でよく使われる FORTRAN と C++を同じフレームワーク上で同等に扱えるようになった。２点目についてはオブジェクト指向技術を採用したことで、非常に粒度の小さいレベルで自律したモジュールを作成している。ただし、従来システムの解析コードをカプセル化して再利用している場合には、解析コードのレベルではモジュール化されていない。今後、解析コードの持つ機能を更にオブジェクト指向に対応させていく必要がある。３点目については現状では対応しているとはいえないが、Python が対応している PC クラスタのような環境であれば、今後対応していくことは可能と考えられる。ただし、大型計算機のような環境では Python の利用はまだ一般的ではないので今後の課題が残る。４点目については、当時の検討では明確な回答がなかったが、結果的には問題追跡システムとバージョン

管理システム、テスト駆動開発等を融合的に利用することで実現していると考えられる。５点目についてはスクリプト言語 Python と YAML 形式の採用により、データと機能の結合関係をすべて記述することは技術的に可能となっている。

3. フレームワーク

前章で述べたようにMARBLEは従来型の解析システムでいうところの解析システムを開発するためのフレームワークである。ここでは、MARBLEフレームワークの全体概要と機能について説明する。

3.1 全体概要

MARBLEフレームワークに含まれる主なパッケージを図3.1.1に示す。MARBLEフレームワークを機能で分類すると以下の３つに分類することができる。

① 炉心解析用データ管理機能
② 炉心解析機能
③ 二階層システム化支援機能

図では、ほぼこの機能の順番で整理されており、１番目の炉心解析用データ管理機能は主にModelパッケージ（marble.model）に含まれるモジュールによって実現されている。炉心解析で用いるデータはオブジェクト指向技術を使ってなるべく物理的な概念に基づくようにモデル化されている。

２番目の炉心解析機能については、現状のMARBLEでは大部分の機能を従来システムに含まれる解析コードを再利用することで実現している。解析コードの再利用にはカプセル化技術を用いており、図のSlaromパッケージ（marble.capsule.slarom ）からNewPDSパッケージ（marble.capsule.newpds）までが各解析コードのカプセル化のためのモジュールとなっている。その他、一部の解析機能はMARBLE固有のソルバーによって実現されており、これらのソルバーはSolverCoreパッケージ（marble.solvercore）に含まれている。MARBLEの炉心解析機能はこれらのパッケージによって実現されている。なお、解析コードのカプセル化技術については付録Cを参照することができる。

３番目の二階層システム化機能はスクリプト言語（Python）とシステム言語（C++、Fortran）による二階層システムを構築するためのツール群によって実現されている。具体的には、Fortranコードの入出力ファイルを処理するためのツールのFortranUtilsパッケージ（marble.utils.fortran）、Fortranコードの詳細な実行制御を行う低レベルカプセル化層のCapsuleパッケージ（marble.capsule）PythonとC++を直接結合するためのツールであるFrameworkパッケージ（marble.framework）、からなる。

marble/	MARBLE のトップレベルのパッケージ
model/	データモデルのパッケージ
material/	核種・組成・断面積データ
quantity/	基本物理量と単位変換
coordinates/	炉心・計算メッシュの座標系
geometry/	詳細幾何形状
mesh/	３次元計算メッシュ構造
pattern/	集合体の装荷パターン
info/	計算体系全体の情報
procedure/	解析手順
meshflux/	解析コードの計算メッシュと中性子束データ
solver/	解析コード・ソルバーの高レベルカプセル化層
runner/	解析コードの低レベルカプセル化層
solvercore/	新規開発ソルバーのパッケージ
burnup/	燃焼計算ソルバー BURNUP
perturbation/	摂動計算ソルバー PERTURBATION
uncertainty/	炉定数調整・設計精度評価ソルバー UNCERTAINTY
capsule/	低レベルカプセル化層のパッケージ
slarom/	格子計算コード SLAROM の低レベルカプセル化層
casup/	格子計算コード CASUP の低レベルカプセル化層
slaromuf/	格子計算コード SLAROM-UF 及び UFLIB の低レベルカプセル化層
citation/	多次元拡散計算コード CITATION-FBR の低レベルカプセル化層
twotran/	2 次元輸送計算コード TWOTRAN の低レベルカプセル化層
tritac/	3 次元 XYZ 体系輸送計算コード TRITAC の低レベルカプセル化層
nshex/	3 次元 Hex-Z 体系輸送計算コード NSHEX の低レベルカプセル化層
perky/	拡散摂動計算コード PERKY の低レベルカプセル化層
snpert/	2 次元輸送摂動計算コード SNPERT の低レベルカプセル化層
snpert3d/	3 次元輸送摂動計算コード SNPERT3D の低レベルカプセル化層
able/	炉定数調整コード ABLE の低レベルカプセル化層
accept/	設計精度評価コード ACCEPT の低レベルカプセル化層
jfs/	JFS-3 型炉定数セットの低レベルカプセル化層
pds/	PDS（SLAROM、CASUP 実効断面積）ファイルの低レベルカプセル化層
newpds/	NewPDS（SLAROM-UF 実効断面積）ファイルの低レベルカプセル化層
utils/	各種ユーティリティプログラムのパッケージ
fortran/	FORTRAN 形式ファイルの I/O のための FortranUtils パッケージ
python/	Python 関連のユーティリティプログラム
exception/	MARBLE フレームワークの例外（エラー）定義
framework/	C++の二階層システム化のためのツール
scheme/	高速炉核特性解析システム SCHEME
orpheus/	高速炉実機燃焼解析システム ORPHEUS
test/	単体テスト及び結合テスト
data/	単体テスト及び結合テストのためのデータ保存場所

図 3.1.1　MARBLE のパッケージ構造

3.2 炉心解析用データ管理機能

3.2.1 Material パッケージ

炉心解析では物質データを均質の媒質で核種と原子数密度で取り扱うため、物質データを扱うための核種や原子数密度、それらに関連する中性子核反応断面積や遅発中性子に関するデータを取り扱う機能が必要である。このためのパッケージが Material パッケージ（marble.model.material）である。このパッケージには主に以下のようなモジュールが含まれている。

- Nuclide モジュール（核種情報）
- NuclideProperty モジュール（核種依存データ）
- DelayedNeutronData モジュール（遅発中性子関連データ）
- Material モジュール（原子数密度）
- ReactionType モジュール（反応種別情報）
- CrossSection モジュール（ミクロ・マクロ実効断面積データ）

これらは炉心解析おいて非常に基本的なデータとあるので、MARBLE に慣れたユーザが組成データを加工したい場合に直接利用することを想定した設計となっている。このため、他のパッケージに比べると単独で取り出して使うのが容易である。詳細については文献 [48,50] を参照できる。

3.2.2 Quantity パッケージ

Quantity パッケージ（marble.model.quantity）は MARBLE が物理データを保持する際に必要な単位変換を行うためのパッケージである。MARBLE では物理データを保持する際に実装にあわせて先に単位変換するのを避け、なるべくオリジナルの文献に記載されている単位で直接数値データを記録するようにしている。これは単位換算に使った換算係数が後になって変更される可能性があるため、換算係数による影響を後から再評価できるようにトレーサビリティを確保するための配慮である。このパッケージは基本的に物理データを単位付きで正確に記述するのが目的であるが、単独で取り出して単位換算の機能を利用することも可能である。このパッケージには主に以下のようなモジュールが含まれている。

- Mass（質量）
- SubstanceQuantity（物質量）
- Energy（エネルギー）
- Period（時間）
- DecayConstant（崩壊定数）

3.2.3 MeshFlux パッケージ

MeshFlux パッケージ（marble.model.meshflux）は従来システムと同様の概念で炉心計算用のメッシュや中性子束データを取り扱うためのパッケージである。MARBLE では本来、計算メッシュや中性子束のデータは後述のパッケージで取り扱うが、従来システムでは CITATION コード等で用いられている計算メッシュの概念を直接ユーザが取り扱うため、従来システムに慣れたユーザにとっては同様の取扱いができた方が便利なことがある。このため、このパッケージが用意されている。このパッケージには以下のモジュールが含まれている。

- Cell（１次元セル形状）
- MeshFlux（１～３次元計算メッシュ）

１次元のセル形状は１次元の計算メッシュと同様に処理することができるため、このパッケージに含まれている。計算メッシュとしては以下の座標系を取り扱うことができる。

- １次元球座標系（RMesh クラス）
- １次元直交座標系（XMesh クラス）
- ２次元円筒（RZ）座標系（RzMesh クラス）
- ３次元直交（XYZ）座標系（XyzMesh クラス）
- ３次元六角（Hex-Z）座標系（HexzMesh クラス）
- ３次元三角（Tri-Z）座標系（TrizMesh クラス）

このパッケージは SCHEME で利用されており、データの指定方法については付録 H を参照することができる。

3.2.4 Coordinates パッケージ

Coordinates パッケージ（marble.model.coordinates）は典型的な高速炉体系における集合体炉心装荷位置（集合体アドレス）に基づく座標系及び計算メッシュの座標系のデータを管理する。このパッケージが対象とする座標系は３次元六角座標系（Hex-Z）、３次元三角座標系（Tri-Z）、3次元直交座標系（XYZ）である。

このパッケージには主に以下のようなモジュールが含まれている。

- AssemblyAddress（集合体アドレス）
- CoreProperty（炉心形状と炉心特性）
- HexZCoordinates（Hex-Z 座標系）
- TrizCoordinates（Tri-Z 座標系）
- XyzCoordinates（XYZ 座標系）
- PseudoXyzCoordinates（Hex-Z 座標系を XYZ 座標系に変換した XYZ 座標系）

- Zone（ゾーン：同一のミクロ断面積が与えられる領域）
- ZoneSet（ゾーンの集合）

詳細については文献[48)]を参照できる。

3.2.5 Geometry パッケージ

Geometry パッケージ（marble.model.geometry）は集合体の３次元詳細幾何形状及びメッシュ幾何形状をモデル化したものである。このパッケージの主な機能は、幾何形状を定義する機能と定義された幾何形状の任意の領域の面積を算出する機能（レイトレース処理）を有する。このパッケージには主に以下のようなモジュールが含まれている。

- HexShape（正六角形）
- TriangleShape（正三角形）
- CircleShape（正円）
- SquareShape（四角形）
- GeomComponent（幾何形状の構成単位）
- Region（GeomComponent で囲まれた領域）
- SegmentSetGeometry（GeomComponent を Z 軸方向に重ねた３次元幾何形状）
- OverlapRegion（領域の重ね合わせで形成される領域）
- RayTraycer（レイトレース処理機能）

詳細については文献[48)]を参照できる。

3.2.6 Mesh パッケージ

Mesh パッケージ（marble.model.mesh）は炉心全体の３次元メッシュ構造をモデル化したものである。このパッケージでは、前述の Coordinates パッケージで定義された３次元座標系情報と Geometry パッケージで定義された幾何形状情報を組み合わせることで、炉心全体の３次元メッシュ構造を定義することができる。また、異なる座標系で定義された３次元メッシュ構造を相互に変換する機能も持っている。このパッケージには主に以下のようなモジュールが含まれている。

- HexzMeshGeometry（六角メッシュによる３次元メッシュ構造）
- TrizMeshGeometry（三角メッシュによる３次元メッシュ構造）
- XyzMeshGeometry（四角メッシュによる３次元メッシュ構造）
- MeshNumberGlobalConverter（異なるメッシュ体系間のメッシュ番号変換）

詳細については文献[48,50)]を参照できる。

3.2.7 Pattern パッケージ

Pattern パッケージ（marble.model.pattern）は集合体の装荷パターンをモデル化したものであり、集合体アドレスとそこに配置される集合体に関する情報（名前、種類、装荷された向き、装荷履歴等）を管理することができる。このパッケージに含まれる主なモジュールは以下のとおりである。

- Pattern（装荷パターン）
- CellData（集合体情報）

詳細については文献 [48,50] を参照できる。

3.2.8 Info パッケージ

Info パッケージ（marble.model.info）は計算体系全体の情報を管理するためのパッケージである。炉心の中性子束分布や核種の数密度分布、ミクロ及びマクロ断面積等、ソルバーが計算を実施するために必要な情報やソルバーが算出した計算結果等を管理する。

- CoreInfo（計算体系全体の情報の管理）
- CoreFluxInfo（計算体系全体の中性子束分布の管理）
- FluxRegionInfo（中性子束メッシュ：同一の中性子束が与えられる領域）
- CoreMaterialInfo（計算体系全体の組成分布の管理）
- MaterialRegionInfo（マテリアルメッシュ：同一の組成が与えられる領域）
- CrossSectionInfo（計算体系全体のミクロ／マクロ断面積の管理）
- InitialComposition（初期組成の管理）
- CellModelConverter（格子計算用の集合体モデル）

詳細については文献 [48,50] を参照できる。

3.2.9 Procedure パッケージ

Procedure パッケージ（marble.model.procedure）は計算手順を定義するためのパッケージである。このパッケージは計算手順を追加・拡張することを前提としたパッケージとなっており、MARBLE フレームワークに新しい計算手順を追加する場合にはこのパッケージを拡張する。現状定義されている計算手順としては格子計算、炉心計算、燃焼計算等がある。また、実効断面積のエネルギー群縮約処理や制御棒操作に関する処理等の計算手順も定義されている。各計算手順に応じて以下のようなモジュールが含まれている。詳細については文献 [50] を参照できる。

- CellCalculationProcedure（格子計算手順）
- MacroXSCalculationProcedure（マクロ断面積生成処理手順）

- CoreCalculationProcedure（炉心計算手順）
- BurnupCalculationProcedure（燃焼計算手順）
- CollapseGroupCalculationProcedure（エネルギー群縮約処理手順）
- ControlRodOperationProcedure（制御棒操作に関する処理手順）

3.2.10 Solver パッケージ

Solver パッケージ（marble.model.solver）は MARBLE フレームワークの中で高レベルカプセル化層と呼ばれているソフトウェア階層に対応する。MARBLE のカプセル化は 2 段階で行われており、解析コードに依存したインターフェース（解析コード特有のファイル形式やデータ構造等）を持つカプセル化部分である低レベルカプセル化層と、これらのインターフェースを統一的に扱う高レベルカプセル化層（入力データの設定、実行、出力データの取得等）に分かれている。高レベルカプセル化層では、カプセル化された FORTRAN の解析コードだけでなく、Python、C++で開発されたソルバーも同様に取り扱うことが可能であり、MARBLE で扱う解析コード・ソルバーを統一的に制御するためのパッケージである。

このパッケージについては解析コード・ソルバー毎に以下のようなモジュールが含まれている。詳細については文献 [48,50] を参照できる。

- SlaromUfSolver（SLAROM-UF コードの高レベルカプセル化層）
- CitationSolver（CITATION-FBR コードの高レベルカプセル化層）
- TritacSolver（TRITAC コードの高レベルカプセル化層）
- NshexSolver（NSHEX コードの高レベルカプセル化層）
- BurnupSolver（BURNUP ソルバーの高レベルカプセル化層）

3.3 炉心解析機能

前述のようにMARBLEは高速炉核特性詳細解析システムの後継であり、基本的な解析手法は高速炉核特性解析手法[10,11]と同じであるのでここでは概要だけを述べる。なお、現状のMARBLEが内蔵している炉心解析機能は一部に限られており、大部分の機能は前章で述べた解析コードのカプセル化で既存の解析コードを利用することで実現している。

3.3.1 核データライブラリ

評価済み核データライブラリを高速炉解析用に処理して作成されたJFS-3型炉定数セット[19]及び超微細群格子計算コードSLAROM-UF用に開発されたUFLIB型炉定数セット[16]が利用可能である。JFS-3、UFLIBともに現在、以下の評価済み核データライブラリに基づく炉定数セットが利用できる。

- JENDL-4.0 [51]
- JENDL-3.3 [52]
- JENDL-3.2 [53]
- ENDF/B-VII.0 [54]
- JEFF-3.1 [55]

3.3.2 格子計算

格子計算コードとしては、従来のCASUP[20,21]、SLAROM[19]コードに加えて、両コードの機能を包含した超微細群格子計算コードSLAROM-UF[15,16]を用いることができる。SLAROM-UFに組み込まれておりMARBLEで利用可能な主な解析手法（機能）としては、(1) 東稔法に基づく背景断面積の計算機能[56]、(2) 超微細群計算機能、(3) 反応率比割合保存法（RRRP法[57]）が挙げられる。また、輸送断面積としては中性子束重み輸送断面積、カレント重み輸送断面積（SLAROM-UFのみ対応可能）、EXAPANDA方式輸送断面積（SLAROM、CASUPでカレント重み輸送断面積と呼ばれていたもの）に対応している。なお、SLAROM-UFはCASUP、SLAROMの機能を包括しているのでMARBLEでは格子計算コードとしてSLAROM-UFを利用することを推奨している。

3.3.3 炉心計算

炉心計算コードとしては解析対象の座標系や中性子輸送の取扱いにあわせて複数の解析コードを利用することができる。２次元及び３次元拡散計算コードとしてはCITATIONコード[22]をベースとしたCITATION-FBRコードが利用できる。輸送計算コードとしては、2次元輸送計算コードTWOTRAN[25]、３次元XYZ体系輸送計算コードTRITAC[23,24]と３次元Hex-Z体系ノード

法輸送計算コードNSHEX[58–61]が利用できる。

また、従来システムには組み込まれていなかったため、現状のMARBLEでは低レベルカプセル化層のみの実験的な実装であるが、多次元拡散計算コードDIF3D[62]や、C++で開発されたCBG[63]の有限差分拡散計算ソルバーPLOSやSn輸送計算ソルバーSNTも組み込まれている。

炉心計算コードと連携して利用可能な摂動計算コードとしては、拡散摂動計算コードPERKY[26]、２次元輸送摂動計算コードSNPERT[27]、３次元輸送摂動計算コードSNPERT3Dを利用できる。ただし、これらの摂動計算コードは従来システムと同様にそれぞれ、CITATION-FBR、TWOTRAN、TRITACコードと連携して使う必要がある。また、現状は３次元XYZ体系の反応度計算にしか対応していないが、PERTURBATIONと呼ばれるMARBLE固有のソルバーを利用することもできる。詳細については付録Eを参照できる。

3.3.4 燃焼計算

燃焼計算機能については、BURNUPと呼ばれるMARBLE固有のソルバーを利用することができる。BURNUPは行列指数法及びクリロフ部分空間法に基づく燃焼計算[64]ソルバーである。詳細については付録Dを参照することができる。また、現状のMARBLEでは燃焼計算機能としてORIGEN2コード[65]を利用することはできないが、ORIGEN2ライブラリに対する低レベルカプセル化層を実装しているので、ORIGEN2ライブラリを利用したり作成したりすることができる。

3.3.5 感度計算

核データに対する感度係数を計算する機能として一般化摂動論に基づく感度解析コードSAGEP[28]を利用することができる。SAGEPコードを用いることで高速炉解析に必要な基本核特性（実効増倍率、反応度価値、反応率比、反応率分布）に対して核種、反応、エネルギー群毎の寄与（感度係数）を求めることができる。

なお、従来システムSAGEP-FRには燃焼の効果を考慮して感度係数を計算することが可能なSAGEP-BURN[30–32]が含まれているが、SAGEP-BURNの機能は現状のMARBLEでは利用できない。燃焼感度解析を行う場合は、PSAGEPと呼ばれる別の解析システム[7–9]を利用する必要がある。

3.3.6 炉定数調整・核設計精度評価

炉定数調整・核設計精度評価については、UNCERTAINTYと呼ばれるMARBLE固有のソルバーを利用することができる。UNCERTAINTYは、従来システムのABLE、ACCEPTコード[33]に相当する機能を有しており、従来バイアス因子法、炉定数調整法、拡張バイアス因子法、一般化バイアス因子法を用いた核設計精度評価を行うことができる。UNCERTAINTYは炉定数調

整法に基づく核設計精度評価ソルバー[6)]として開発されたものを拡張したものである。変更点については付録 F を参照することができる。

なお、MARBLE では ABLE、ACCEPT をソルバーとして直接利用する機能は提供していないが、UNCERTAINTY は ABLE、ACCEPT の入力データ形式にも対応しているため、既存の ABLE、ACCEPT の入力データを利用することができる。

3.4 二階層システム化機能

MARBLE フレームワークではプログラミング言語 Python、FORTRAN、C++を連携して利用できるようにするためのツールを提供している。

3.4.1 Python と FORTRAN の二階層システム化機能

FORTRAN で書かれた解析コードを MARBLE フレームワークに組み込んで使う場合には以下のパッケージを利用することができる。

- FortranUtils パッケージ（marble.utils.fortran）: FORTRAN ファイルの入出力処理
- Capsule パッケージ（marble.capsule）: FORTRAN コードの実行制御

FortranUtils パッケージは FORTRAN の READ 文と WRITE 文を使った入出力処理を Python 上で模擬するためのパッケージである。FORTRAN のテキストファイル（固定形式、自由形式、NAMELIST 形式）及びバイナリファイルのほとんどの入出力処理に対応しており、FORTRAN とよく似た構文で Python から入出力処理できるようになっている。このパッケージを使うことで FORTRAN コードの入出力ファイルを読み書きする低レベルカプセル化層を簡単に作成することができる。MARBLE に組み込まれている解析コードの低レベルカプセル化層はこのパッケージを使っている。FortranUtils のパッケージの利用方法については付録 G を参照できる。

3.4.2 Python と C++の二階層システム化機能

C++で書かれた解析コードを MARBLE フレームワークに組み込んで使う場合には、以下のパッケージを利用することができる。

- Framework パッケージ（marble.framework）

MARBLE で C++と Python を結合して利用する場合には高機能 C++ライブラリ Boost[66] の一部である Boost Python Library を利用する。Boost Python Library は完成度の高いライブラリであるが、数値計算で取り扱う配列データを受け渡すためには補助的なツールが別途必要となる。このため、MARBLE フレームワークでは Python の Numpy ライブラリの配列データ（numpy.ndarray）、C++の標準テンプレートライブラリの配列データ（std::vector）、C++の高機能数値計算ライブラリ Blitz++[67] の配列データ（blitz::array）を相互に受け渡すためのツールを提供している。

このパッケージの詳細については文献 [47] を参照できる。

3.5 自己テスト機能

この機能は開発者向けの機能であるが、以下のパッケージにはMARBLEのテスト駆動開発を支えている単体テスト（モジュール単位での動作確認）や結合テスト（モジュールを組み合わせた場合の動作確認）が収められている。

- Testパッケージ（marble.test）：単体テストと結合テスト
- Dataパッケージ（marble.data）：テスト用データの保存場所

Testパッケージには各パッケージのモジュール毎に用意された単体テストが含まれている。また。このパッケージにはSCHEMEの結合テスト（marble.test.scheme.join）とORPHEUSの結合テスト（marble.test.orpheus.join）も含まれている。

なお、これらのパッケージの目的は動作確認（Verification）であって、妥当性確認（Validation）ではない。このため、動作確認テストに必要な時間を短くするために物理的に不自然なデータやオプションを設定をしているものも多い。すなわち、入出力データそのものに物理的な正確さは不要であり、各モジュールにおける入力と出力の間の正しい関係（モジュールの動作）を示すことに重点が置かれている。

したがって、TestパッケージやDataパッケージに含まれているデータは、以下に述べるようなMARBLEの動作確認やモジュールの動作を理解する以外の目的に使用してはならない。

3.5.1 利用方法 (1)：動作確認

Testパッケージに含まれるモジュールはすべて単体で実行できるようになっており、実行することで対応するモジュールの単体テストが行われる。単体テストや結合テストをまとめて実行することも可能であり、このような場合には以下のようなスクリプトを利用することができる。

- marble/test/alltest.py（すべての単体テストの実行）
- marble/test/scheme/join/run_all.sh（SCHEMEのすべての結合テストの実行）
- marble/test/orpheus/join/run_all.sh（ORPHEUSのすべての結合テストの実行）
- marble/alltest.sh（すべての単体テスト及び結合テストの実行）

なお、これらの単体テストや結合テストは従来システムによる計算結果を参照解としてテストするものが多く、事前に参照解を準備しておかなければならないものが多い。このようなテスト用データを準備するのが、Dataパッケージである。テスト用データを準備するには以下のスクリプトを利用することができる。

- marble/data/run_all.sh（すべてのテスト用データの生成）

3.5.2 利用方法 (2)：サンプル問題としての利用

単体テストと結合テストは本来、開発者が拡張・改修したモジュールが正常に動作するかどうかを確認するために実行するものであるが、テストには結果も含まれているので、どのように動作するかを示すサンプルプログラムと見ることもできる。MARBLE には実装と同程度の量の単体テストが付属しており、単体テストを記述するためのユニットテストフレームワークの基本機能を理解すれば、一般ユーザにとってはサンプルプログラムとして利用することができる。

このため、ユニットテストフレームワークの使い方を簡単に説明する。以下に単体テストの例を示す。

```
class NuclideTest(unittest.TestCase):

   def testPu239(self):
       pu239 = Nuclide("Pu-239")
       self.assertEquals(pu239.anumber(), 94)
       self.assertEquals(pu239.amass(), 239)
       self.assertEquals(pu239.element(), "Pu")
       self.assertEquals(pu239.name(), "Pu-239")
       self.assertEquals(pu239.id(), 942390)
       self.assertEquals(pu239.jfsid(), 949)

   def testH1(self):
       h1 = Nuclide("H-1")
       self.assertEquals(h1.anumber(), 1)
       self.assertEquals(h1.amass(), 1)
       self.assertEquals(h1.element(), "H")
       self.assertEquals(h1.name(), "H-1")
       self.assertEquals(h1.id(), 10010)
       self.assertEquals(h1.jfsid(), 1)
```

単体テストはオブジェクト指向言語 Python のクラス（class 文で定義）とメソッド（def 文で定義）を使って定義されるが、メソッドがひとつのテストでクラスはそれらのテストを束ねるものとなっている。

この例は Nuclide クラスの単体テストとして作成されたものであるが、Pu-239 と H-1 を例にした２つのテストが定義されている。どちらのテストも実際に Nulicde クラスのインスタンスを作ってどのように動作すべきかをテストしている。メソッドの引数 self はこの例では意味を持たないので無視してよい。`assertEquals()` は括弧内の二つの結果が等しい (equals) ことを宣言（assert）しておりこれが正しいかどうかをチェックしている。amass()、anumber()、element()、name()、id()、jfsid() は、それぞれ、原子番号、質量数、元素、核種名、MARBLE 核種 ID、JFS ライブラリでの核種 ID を返すメソッドであり、テストを実行しなくてもテストを読むだけでこれらの動作を理解することができる。

`assertEquals()` のようなチェック（宣言）文はいくつかバリエーションがあるが、次に示すものを理解しておけばほとんどの単体テストの意味を理解できる。

`assertEquals(x, y)`	x と y は等しい
`assertNotEquals(x, y)`	x と y は等しくない。
`assertTrue(x)`	x の実行結果は真値（True）である。
`assertFalse(x)`	x の実行結果は偽値（False）である。
`assertAlmostEquals(x, y)`	x と y はほぼ等しい[1]。
`assertTrue(allclose(x, y))`	配列 x と配列 y に含まれる値はすべてほぼ等しい[1]。
`assertRaises(x, y)`	y を実行すると例外（エラー）x が発生する。

[1]浮動小数点数どうしを比較する際に利用する。

4. 解析システム

前述のようにMARBLE本体はフレームワークであるため、フレームワークを利用して解析システムに相当する新たな機能を作成し、内部に含めることができる。本章ではMARBLE上で構築された２つの解析システム、高速炉核特性解析システムSCHEMEと高速炉実機燃焼解析システムORPHEUSについて説明する。

4.1 高速炉核特性解析システムSCHEME

SCHEMEは従来システムJOINT-FRに相当する（燃焼計算機能を含まない）高速炉核特性解析システムである。JOINT-FRと同様に主に臨界実験体系の詳細解析を対象としているが、六角メッシュを取り扱うことができるので　実機体系の特定の時点における核特性の詳細解析にも利用できる。解析手法・モデルはJOINT-FRと同等であるが、入力データの作成を簡略化して使いやすさを向上することを目的としている。また、MARBLEフレームワークを利用することで柔軟性・拡張性を持たせたシステムとなっている。

4.1.1 基本設計

SCHEMEは従来システムJOINT-FRを既に利用しているユーザや、新たにJOINT-FRを利用して核特性解析を実施したいと考えている人が利用することを想定した解析システムである。このため、以下のような基本方針で作成した。

① 従来システムの計算結果を再現する。
② 従来システムのユーザになじみのある概念を使う。
③ 自動的に決定できるデータの設定は自動化する。

１点目についてはMARBLE本体と同じ要件である。２点目については、従来システムのユーザは従来システムの入力データの形式で種々のデータを保管していることが多く、入力データを生成するツールを保持していることも多い。このため、既存の入力データや関連ツールを再利用できるように、入力データの基本概念を変更しないようにした。また、MARBLEの基盤技術となっているオブジェクト指向技術について詳しく知らなくても使い始められることを目標にした。このため、FORTRANのサブルーチン・関数の概念に似たPythonの関数のみを利用することにした。３点目については従来システムのソフトウェア設計上の制限に由来する解析コードのオプション設定方法や入力データの準備方法に関する経験や知識を反映して処理を自動化した。ただし、完全自動化によるブラックボックス化は避け、必要に応じてどのようなオプションやデータが生成されているのかを確認できる仕組みを設けた。

4.1.2 解析機能

SCHEME ではデータの準備、コードの実行等すべて関数を使って処理する。このため、SCHEME の解析機能の概要は SCHEME で利用可能な関数を知ることで理解できる。以下に SCHEME で利用できる主な関数を示す。

- 格子計算
  - slaromuf()：超微細群格子計算コード SLAROM-UF の実行
- 炉心計算
  - citation()：拡散計算コード CITATION-FBR の実行
  - tritac()：３次元 XYZ 体系輸送計算コード TRITAC の実行
  - nshex()：３次元 Hex-Z 体系輸送計算コード NSHEX の実行
- 摂動計算
  - citation_perky_for_reactivity()：拡散摂動理論による反応度計算（PERKY）
  - tritac_snpert3d()：輸送摂動理論による反応度計算（SNPERT3D）
  - perturbation_diffusion()：拡散摂動理論による反応度計算（PERTURBATION）
  - perturbation_transport()：輸送摂動理論による反応度計算（PERTURBATION)
  - citation_perky_for_beta_effective()：実効遅発中性子割合の計算 (PERKY)
  - citation_perky_for_prompt_neutron_lifetime()：即発中性子寿命の計算（PERKY）
  - citation_perky_for_worth_mapping()：反応度価値空間分布の計算（PERKY）
- 縮約計算
  - condense()：実効断面積の縮約計算（任意の群数）
  - condense7g()：実効断面積の縮約計算（70 群→ 7 群）
  - condense18g()：実効断面積の縮約計算（70 群→ 18 群）
- 実効断面積処理
  - xmix()：マクロ断面積の生成と平均化処理
- 組成データ作成
  - material()：組成データの作成
  - materialset()：組成データの集合の作成
- 格子計算モデル作成
  - homocell()：均質セルモデルの作成
  - slabcell()：１次元スラブセルモデルの作成
  - ringcell()：１次元リングセルモデルの作成
- 炉心計算メッシュ作成

- rzmesh()：２次元 R-Z 体系の計算メッシュの作成
- xyzmesh()：３次元 XYZ 体系の計算メッシュの作成
- hexzmesh()：３次元 Hex-Z 体系の計算メッシュの作成
- trizmesh()：３次元 Tri-Z 体系の計算メッシュの作成

- 従来システムからのデータ変換
  - homocell()：SLAROM、SLAROM-UF 入力データからの変換
  - slabcell()：CASUP、SLAROM-UF 入力データからの変換
  - ringcell()：CASUP、SLAROM-UF 入力データからの変換
  - rzmesh()：CITATION-FBR、TWOTRAN 入力データからの変換
  - xyzmesh()：CITATION-FBR、TRITAC 入力データからの変換
  - hexzmesh()：CITATION-FBR、NSHEX 入力データからの変換
  - trizmesh()：CITATION-FBR 入力データからの変換
  - material_from_newpdsfile()：新 PDS ファイルからの組成データ作成
  - material_from_pdsfile()：旧 PDS ファイルからの組成データ作成
  - macro_from_newpdsfile()：新 PDS ファイルからのマクロ断面積作成
  - macro_from_pdsfile()：旧 PDS ファイルからのマクロ断面積作成
  - micro_from_newpdsfile()：新 PDS ファイルからのミクロ断面積作成
  - micro_from_pdsfile()：旧 PDS ファイルからのミクロ断面積作成

これらの関数の詳細については、付録 H を参照することができる。

4.1.3 検証計算

SCHEME の検証計算として ZPPR-9 の臨界性の解析を行った。この検証計算では高速炉核特性詳細解析手法に基づき、メッシュ補正、輸送補正、群縮約補正を行い、これらの補正係数についても SCHEME 上で正しく計算できることを確認した。

表 4.1.1 に検証計算結果を示す。この表から分かるように MARBLE/SCHEME と JOINT-FR の計算結果は有効数字の範囲内で完全に一致している。基本的に両者で SLAROM-UF、CITATION、TRITAC 等の解析コードは全く同じものを利用しているが、JOINT-FR のインターフェースコード JOINT の機能等については MARBLE 上で再実装している。このため、浮動小数点データの計算順序やテキストファイル上の有効桁数等が原因で浮動小数点データとしては完全再現しないが、実用上問題のない精度で JOINT-FR の計算結果を再現することが確認できる。

表 4.1.1　SCHEME の検証計算（ZPPR-9 臨界性）

		JOINT-FR	MARBLE	差 (pcm)
基準計算値		0.994211	0.994209	−0.2
補正	メッシュ補正	−0.001000	−0.001003	−0.3
	輸送補正	+0.002335	+0.002342	+0.6
	群縮約補正（超微細群＋175群）	+0.000087	+0.000089	+0.2
補正後計算値		0.995634	0.995637	+0.4

4.1.4　まとめ

以上のように SCHEME は従来システム JOINT-FR に相当する高速炉核特性解析システムである。SCHEME を含む MARBLE の段階的な利用方法については第 5 章を参照することができる。また、SCHEME のユーザマニュアルは付録 H を参照することができる。なお、SCHEME の関数は MARBLE フレームワークの一部でもあるので、MARBLE フレームワークの部品と同様にオンラインマニュアルを参照することもできる。

4.2 高速炉実機燃焼解析システムORPHEUS

ORPHEUSは高速炉実機の燃焼を考慮して詳細核特性解析を行うための解析システム[68–72]である。従来システムJOINT-FRを用いて高速炉実機を対象とした燃焼解析を行う場合、入力データが非常に複雑になってしまうため、詳細なモデルを適用した解析の実施がきわめて困難であった。このため、簡易モデルによる解析が中心にならざるを得ず、燃焼を伴わない核特性解析と同等の詳細モデルによる解析を行うことができなかった。高速炉実機燃焼解析システムORPHEUSはこれらの問題を解決するためにMARBLE上で開発された。

4.2.1 基本設計

ORPHEUSは燃焼を伴う高速炉実機の核特性解析を目的とした解析システムであり、JOINT-FR相当の高速炉核特性詳細解析手法を実機解析に適用したいと考えている人が利用することを想定した解析システムである。このため、ある程度定められた炉心体系に対して繰り返しパラメータを変更して計算することを想定している。このため、以下のような要件を満たすように基本設計を行った。

① 従来システム相当の詳細な核特性解析手法を実機の燃焼解析に適用できる。
② 必要最低限の入力データを準備するだけで実行できる。
③ ソフトウェア上の新しい概念を理解する必要がない。

１点目はORPHEUSの開発の主目的である。例えば、従来システムが輸送理論に基づく燃焼計算に対応していなかったことから、詳細解析で重要な輸送・メッシュ補正計算を実施できるようにする必要があった。また、燃焼に伴うミクロ断面積の変化、サイクル毎の燃料交換、制御棒操作等を扱っても入力データの取扱いが複雑にならないようにして、ユーザが炉物理の解析手法だけに注力して解析モデルの効果等を評価できるようにすることを基本設計とした。

このため、図4.2.1に示すように、基本的にユーザは計算コードや計算条件に依存しない情報のみを与えればよいようにした。具体的には集合体の詳細幾何形状や装荷パターン、組成などで定義されるas-builtな体系情報を入力する。これらの情報から計算に必要なモデルに自動変換されて計算が行われる。

なお、ORPHEUSでは従来型の解析システムと同様に、ある一定の入力形式に従って入力データを準備するだけで実行できるようにした。これは、SCHEMEのようにユーザが自由に手順を変えたり途中に独自の処理を加えたりする機能を提供するとかえってユーザを混乱させると判断した結果である。すなわち、ORPHEUSでは新しい解析手順を組み込んだり新しい解析手法を組み込んだりするようなユーザは想定していない[1]。

[1]MARBLEフレームワークからみればこのような拡張性を想定しているが、開発者としてORPHEUS本体に修正を加える必要がある。

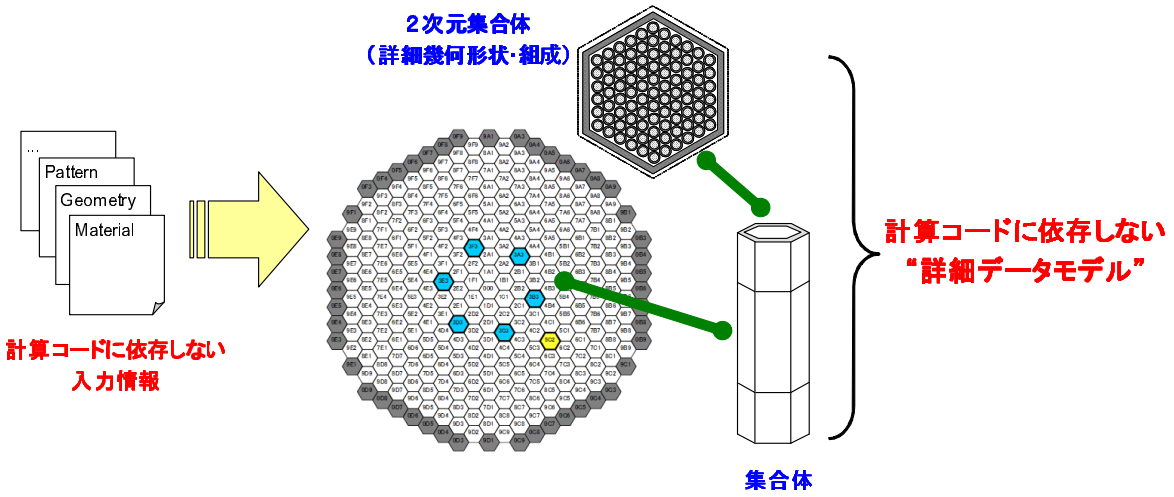

図 4.2.1　ORPHEUS における解析データの取扱い

4.2.2　特徴

ORPHEUS では自動化できる処理はなるべく自動化し、作業の効率化・省力化と解析作業における入力ミスの低減による品質向上を目指した。また、従来システムでは実施困難であった燃焼を考慮した詳細な核特性解析を容易にすることを目的とした。

4.2.2.1　機能上の特徴

機能上の特徴としては以下のような項目が挙げられる。

- 各計算ステップの一連の処理の自動実行機能
- 燃焼に伴うミクロ断面積の自動更新機能
- 燃焼効果を考慮した群縮約計算機能
- 制御棒操作機能

ORPHEUS では、図 4.2.2 に示すような炉心解析で必要な格子計算、炉心計算、燃焼計算の繰り返し計算を含む一連の処理を自動的に実行する。また、一連の処理を自動化することで、燃焼に伴うミクロ断面積の更新も自動的に行うことができる。群縮約計算機能としては、ユーザが任意の燃焼時点のスペクトルを指定して群縮約計算する方法と、ミクロ断面積の更新を考慮して各燃焼ステップで自動的に群縮約計算する方法を使うことができる。制御棒操作機能としては、任意の軸方向位置を指定することで制御棒の引き抜き位置をモデル化することができるようになっている。

その他、ユーザが本来必要な炉物理的な解析に集中できるように解析モデルの作成作業等の多くが自動化されている。特に格子計算と炉心計算における計算体系のモデル作成は手作業で行う

と煩雑になりやすいが、ORPHEUS では以下の自動生成機能を利用することができる。

- 格子計算モデルの自動生成機能
- 炉心計算メッシュの自動生成機能

格子計算モデルについては、図 4.2.3 に示すようにユーザが定義した集合体詳細モデルから均質モデルや 1 次元リングモデルに自動変換することができる。炉心計算メッシュについては、図 4.2.4 に示すように Hex-Z 体系、XYZ 体系、Tri-Z 体系を相互に自動変換することができる。

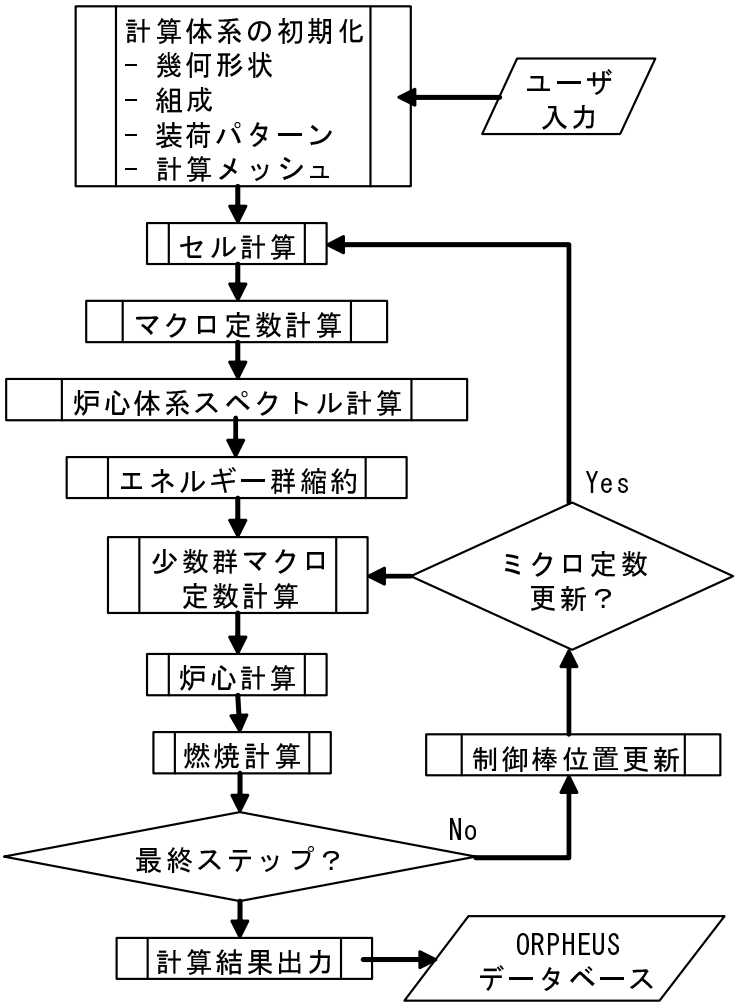

図 4.2.2　ORPHEUS における計算の流れ

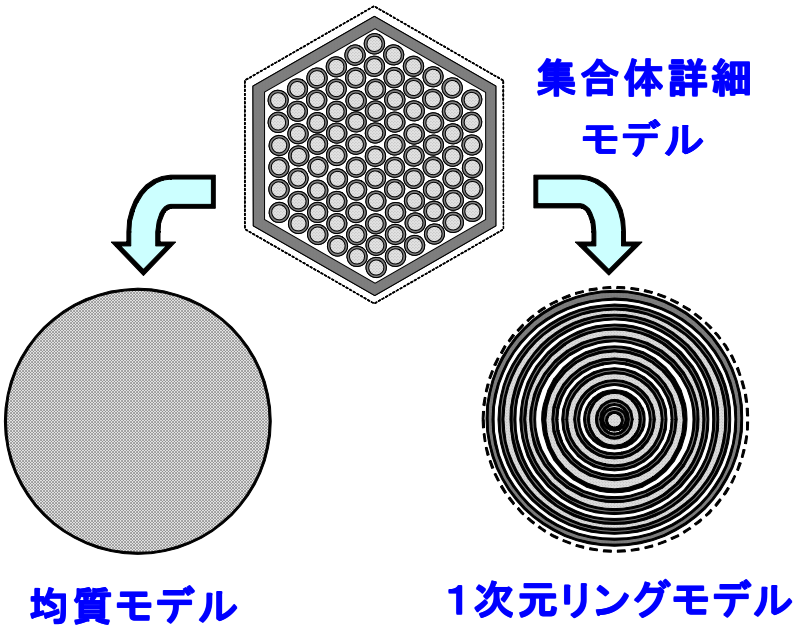

図 4.2.3　ORPHEUS の格子計算モデル自動変換機能

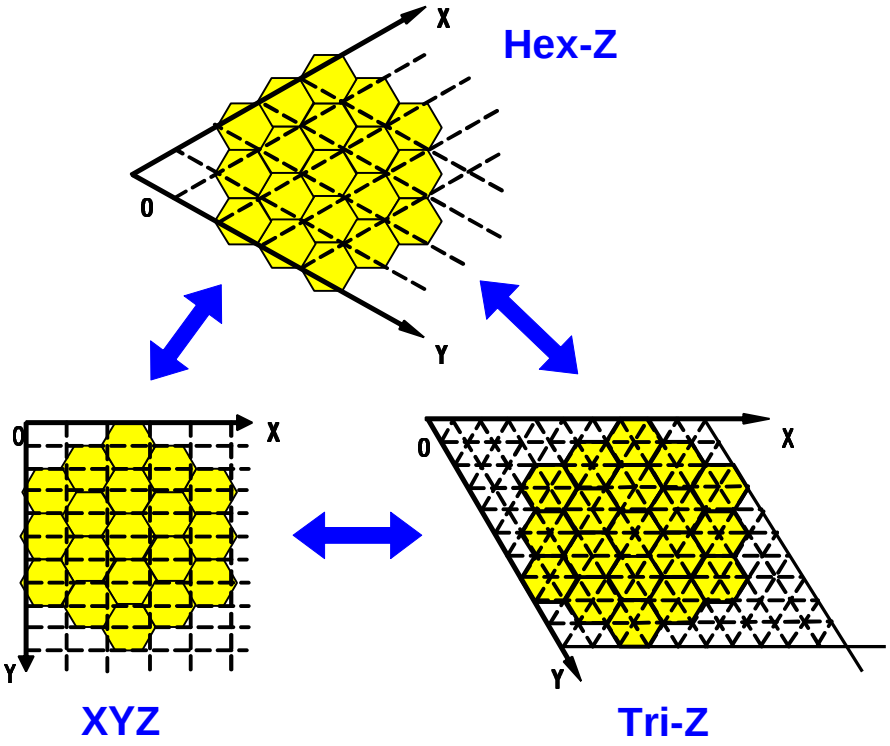

図 4.2.4　ORPHEUS の計算メッシュ自動変換機能

4.2.2.2　ユーザインターフェース上の特徴

ユーザインターフェース上の特徴としては以下の項目が挙げられる。

- 理解が容易な入力ファイル
- 計算結果のデータベース保存

入力ファイルは構造化テキストファイル（YAML）形式を利用して、データに名前を付け階層化することで入力ファイルの意味が理解しやすくなるようにしている。また、計算結果についてはユーザが柔軟に後処理して利用できるようにデータベースに保存するようにしている。

4.2.3 解析機能

ORPHEUS ではデータの準備、コードの実行を含めてすべて入力ファイルで指定する。ORPHEUS の入力ファイルは以下の５つに分かれている。

- 計算条件ファイル（ユーザ入力ファイル）
- 物質組成ファイル
- 幾何形状ファイル
- 装荷パターンファイル
- 炉心特性ファイル

これらのファイルを使うことで標準的な高速炉実機の六角格子状の燃料集合体の形状を含めて炉心解析に必要な炉心体系の情報を定義することができるようになっている。解析対象の燃料組成、幾何形状、装荷パターン、炉心基本特性を一度だけ定義しておけば、前述の格子計算モデルや炉心計算モデルの自動生成機能により一連の計算処理を実行することができる。解析手法として ORPHEUS では以下のキーワードを使うことで解析コード・ソルバーを利用することができる。

- 格子計算
  - slarom_uf: 超微細群格子計算コード SLAROM-UF
- 炉心計算
  - citation：拡散計算コード CITATION-FBR
  - tritac：３次元 XYZ 体系輸送計算コード TRITAC
  - nshex：３次元 Hex-Z 体系輸送計算コード NSHEX
- 燃焼計算
  - burnup：燃焼計算ソルバー BURNUP

格子計算のための解析モデルとしては、(1) 均質モデル、(2) 燃料集合体を対象とした１次元リングモデルを用いた通常均質化、(3) 制御棒集合体を対象とした１次元リングモデルを用いた反応率比保存法（RRRP 法）による均質化の３種類を使うことができる。これらのモデルは前述の格子計算モデルの自動作成機能によりユーザが定義した集合体の詳細形状の情報から自動的に解析モデルを作成して解析コードを実行する。

炉心計算に用いる解析コードは以下に示したキーワードを指定することで自動的に切り替えることができる。例えば、TRITAC コードは XYZ 体系にしか対応していないが、前述の計算メッシュ体系の自動作成機能によりユーザが定義した六角形状の炉心体系の情報から自動的に計算メッシュを作成して解析コードを実行する。また、このリストから明らかなように ORPHEUS では TRITAC、NSHEX を炉心計算に用いることができるので従来システムではできなかった輸送理論に基づく燃焼計算を行うことができる。

4.2.4 検証計算

ORPHEUSの検証計算として高速実験炉「常陽」MK-I炉心を対象とした計算を行い、従来システムの結果との比較を行った。表4.2.1、4.2.2にそれぞれ、実効増倍率と原子数密度（EOC）の比較結果を示す。なお、従来システムは燃焼計算に対応しているのは拡散計算コードのCITATION-FBRのみであるので、この結果は拡散計算での比較である。格子計算コード（SLAROM-UF）と炉心計算コード（CITATION-FBR）は両者で同じものを利用している。燃焼計算コードについては従来システムではCITATION-FBRに組み込まれた行列指数法のソルバー、ORPHEUSではMARBLEの燃焼計算ソルバーBURNUPを用いている。また、各コードを連携するインターフェースコードとして従来システムではJOINTコードを用いている。ORPHEUSではMARBLEフレームワークを使って各コードの連携をしている。

これらの表からORPHEUSは従来システムの結果を完全に再現していることが確認できる。

表 4.2.1　ORPHEUSの検証計算結果（実効増倍率）

	従来システム	ORPHEUS	Diff (%)
BOC	0.99942	0.99942	0.00
EOC	0.99151	0.99151	0.00

表 4.2.2　ORPHEUSの検証計算結果（原子数密度）

	従来システム	ORPHEUS	Diff (%)
U-235	7.2989E-4	7.2989E-4	0.00
U-238	5.3316E-3	5.3316E-3	0.00
Pu-239	1.7657E-3	1.7657E-3	0.00
Pu-241	7.4911E-5	7.4911E-5	0.00
Am-241	8.5448E-5	8.5448E-5	0.00

4.2.5 まとめ

以上のようにORPHEUSは高速炉核特性詳細解析手法に基づく高速炉実機用の新しい燃焼解析システムである。ORPHEUSを含むMARBLEの段階的な利用方法については第5章を参照することができる。また、ORPHEUSのユーザマニュアルについては付録Iを参照することができる。

5. 利用方法

本章はMARBLEを使いたい人が最初に読めるように他の章から独立しており単独で読むことができるようになっている。第2章で述べたようにMARBLEは従来システムとは少し異なった基本概念を持っている。また、MARBLEは非常に幅広いユーザを想定して設計されており、具体的には、(1) 炉物理解析手法やモデルの開発・高度化等で必要な非定常的な解析を実施するユーザ、(2) 新型炉の設計研究等で必要な準定常的な解析を実施するユーザ、(3) 実機の運転管理や燃料設計等で必要な定常的な解析を実施するユーザを想定している。このため、MARBLEはユーザの目的にあわせていろいろな使い方ができるようになっている。このMARBLEが持つ多目的性のためどこから使い始めて良いか分からないというユーザは多いと考えられるので、本章ではMARBLEの使い方をいくつか説明する。

5.1 MARBLEの基本機能

5.1.1 MARBLEの特徴

MARBLEを使い始める上で最初に知っておくべき特徴として以下の2点が挙げられる。

- とても小さい（機能の限られた）「部品」の集まりで構成されている
- 対話的に使うことができる

これらの特徴により、最初は電卓のように対話的にコマンドを打ち込みながら使ってみるということが可能となっている。以下では実際にMARBLEの小さな部品を試しに使った例を示す。

5.1.2 部品の利用

高速炉の炉心解析の入力データとして必要になる基本情報のひとつとして、組成のデータがある。MARBLEでは組成データは「Material」と呼ばれる部品になっている。このMaterialを構成する更に小さな部品として、組成データを入力するときに指定する核種の情報を示す「Nuclide」が存在する。ここではこの「Nuclide」という部品の利用例を示す。

「Nuclide」はその名が示すとおり核種に関する基本情報を管理している。例えば、従来システムでは、U-238を「928」、Pu-239を「949」といった整数に対応させる核種ID（JFSID）を使ってユーザが核種の情報を管理する必要があった。MARBLEでは核種IDは核種に関する基本情報なので「Nuclide」が核種IDについても管理している。

MARBLEはPythonと呼ばれる一般的なプログラミング言語で開発されているため、MARBLEを対話的に使う場合はPythonを直接利用する。コマンドラインに「python」と打ち込むと以下のようになる。

```
% python
Python 2.5.2 (r252:60911, Jan 24 2010, 14:53:14)
[GCC 4.3.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

環境によってバージョンの情報などは異なる可能性があるが、同様の情報が画面に表示されるはずである。最後に表示されている「>>>」は Python のコマンドプロンプトで、ここにコマンドを入力することで MARBLE を対話的に使うことができる。

```
>>> from marble.model.material.Nuclide import Nuclide
>>> pu239 = Nuclide("Pu-239")
>>> pu239.jfsid()
949
```

「>>>」で始まる行が実際にユーザが入力するコマンドである。この例からは「Pu-239」という核種の名前から「949」という核種 ID を取り出していることが分かる。「Nuclide」は核種 ID 以外にも核種に関する基本的な情報を保持している。

```
>>> pu239.element()
'Pu'
>>> pu239.anumber()
94
>>> pu239.amass()
239
```

「anumber」と「amass」がそれぞれ「atomic number」と「atomic mass」の省略形であることが分かれば、Pu-239 の元素名 Pu、原子番号 94、質量数 239 という情報を取り出していることは明確である。これが MARBLE で「部品」と呼ばれているものである[1]。

「Nuclide」は核種変換に関する情報も管理しているので、もう少し複雑な例として、以下に Pu の増殖に関する核変換を確認する方法を示す。

```
>>> u238 = Nuclide("U-238")
>>> u238.transform("capture")
Nuclide('U-239')
>>> u238.transform("capture").transform("beta_decay")
Nuclide('Np-239')
>>> u238.transform("capture").transform("beta_decay").transform("beta_decay")
Nuclide('Pu-239')
```

この例では「transform」が核変換を表すことが分かれば以下の核変換過程を意味していることが分かる。

$$^{238}\mathrm{U} \xrightarrow{(\mathrm{n},\gamma)} {}^{239}\mathrm{U} \xrightarrow{\beta^-} {}^{239}\mathrm{Np} \xrightarrow{\beta^-} {}^{239}\mathrm{Pu}$$

5.1.3 まとめ

ここで紹介した「部品」の機能があまりに単純過ぎて（部品が小さすぎて）存在意義を感じない読者も多いはずである。そのような読者はこのような小さい部品を直接使う必要がない可能性が高いので、このような使い方について知る必要はない。

[1] 厳密には、Nuclide がクラスであり、pu239 が Nuclide クラスから生成されたインスタンスである。

MARBLE がこのようなとても小さい部品を提供している理由は、非定常的な解析作業をするユーザを想定しているからである。例えば、非定常的な解析作業をするユーザは、手作業では処理できないほどの大量のデータについて核種 ID の変換作業をしたり、数百核種からなる燃焼チェーンの定義ファイルに間違いがないかチェックをしたりすることがある。このようなユーザにとってはこのような小さい部品を活用できる可能性がある。このような小さな部品に MARBLE の利用価値を見いだす読者は以下のような部品を活用できる可能性がある。

- Material（組成データ）
- CrossSection（中性子核反応断面積データ）

前述のように「Material」は均質組成データを管理するための部品である。「Material」は組成データの保持だけでなく、均質化操作などで必要になる組成データの足し算や割り算等にも対応している。このため、入力データの作成時の均質組成の計算等に活用できる可能性がある。「CrossSection」は実効断面積データを管理するための部品であり、従来システムの PDS ファイルや New PDS ファイルの読み込み、エネルギー群縮約等の機能を持っている。また、「Material」と「CrossSection」を組み合わせて使うことで、ミクロ断面積と組成データからマクロ断面積を計算することもできる。

これらの小さい部品はなるべく特定の利用目的を設定しないように意図して設計されている。これはユーザ自身が利用目的を決めるということを意味している。

5.1.4 補足：オンラインマニュアル

MARBLE が対話的に使える利点のひとつとしてオンラインマニュアルが使えるという点が挙げられる[2]。オンラインマニュアルを見るためには Python の help() というコマンド[3]を利用する。例えば、前述の「Nuclide」という部品のオンラインマニュアルを見る場合には以下のようにする。

```
>>> help(Nuclide)
Help on class Nuclide in module marble.model.material.Nuclide:

class Nuclide
 | This class encapsulate nuclide information.
 |
 | An argument must stands for a valid isotop such as "U-235" or
 | "Pu-239", etc.
 |
 | Ex1: Generation of an instance of "Pu-239" and showing its information
 | >>> p9 = Nuclide('Pu-239')
 | >>> print p9
 | Pu-239
 | ....
```

ここでは表示を省略しているが先の例で使った amass() や transform() 等の意味なども記載されている。help コマンドでオンラインマニュアルを参照した後は、「q」を押すとプロンプトに戻る

[2]Python では docstring、Pydoc と呼ばれているものである。
[3]厳密には関数と呼ばれている。

ことができる。なお、このオンラインマニュアルはPythonを起動しなくても「pydoc」というコマンドを使うことでコマンドラインから直接参照することもできる[4]。

```
% pydoc marble.model.material.Nuclide
```

pydocコマンドを使う場合は、直接モジュールの階層を指定する必要がある。この例ではmarble.model.material.Nuclideというモジュールのオンラインマニュアルを参照している[5]。例えば、前述の「Material」や「CrossSection」という部品のオンラインマニュアルを参照するには以下のようにする。

```
% pydoc marble.model.material.Material
% pydoc marble.model.material.CrossSection
```

5.2 従来システムユーザのための機能

本節は、従来システム（JOINT-FR、SAGEP-FR等）のユーザを想定した説明であるので、従来システムを利用したことのない読者は本節を読み飛ばしてもよい。

5.2.1 MARBLEの特徴（追加）

前節ではMARBLEの特徴を２つ挙げたが、従来システムを利用しているユーザにとって重要と思われる特徴が追加で２点ある。

- 従来システムの解析コードを修正せずに再利用している
- 従来システムの入出力データを自由自在に制御できるツールが揃っている

従来システムは長年使い込まれてきた中で問題点が修正され、多くの検証がなされてきた。MARBLEではこのような資産を活用できるように、既存の解析コードを一切変更することなく自由自在に制御できるようにしている[6]。この方法は既存の解析コードには一切変更を加えないので、長年検証・改良されてきた既存のコードに新たなプログラム上の誤りを混入させる危険性がないという利点がある。また、新たな手法を開発した場合に結果を比較するという検証の目的もある。一方、従来システムのユーザが新しいシステムであるMARBLEへの段階的な移行を支援するという意味もある。

5.2.2 従来システムの解析コード制御機能

従来システムに含まれている解析コード（CITATION、TRITAC、PERKY、SNPERT3D等）は、MARBLEの機能を使うことで、入出力ファイルの内容を自由に読み込んだり、変更して書き出

[4] UNIXのmanコマンドに相当する。
[5] MARBLEのソースコードに含まれるmarble/model/material/Nuclide.pyというファイルに相当する。
[6] この方法をカプセル化と呼んでいる。

したりすることができる。また、SLAROM や CASUP が出力する PDS ファイルや SLAROM-UF が出力する NewPDS ファイルにも対応している。その他、SLAROM や CASUP で利用される JFS-3 型炉定数セットや、SLAROM-UF で利用される炉定数セット UFLIB にも対応している。

これらの機能の実際の利用例を示す。なお、以下の例では既に python コマンドを起動してある状態から始めている。

```
>>> from marble.capsule.casup.CasupUtils import CasupFort5
>>> cas5 = CasupFort5(path="casup.ft5")
>>> cas5.mcode()
array([[ 8, 24, 26, 28, 42, 11, 925, 928, 948, 949, 940, 941, 942,
       951]])
>>> cas5.an()
array([[ 1.65356006e-02, 3.26604000e-03, 1.18917003e-02,
         2.09279009e-03, 2.35314001e-04, 9.34839994e-03,
         1.57985999e-03, 5.22520021e-03, 1.45700994e-06,
         1.11473002e-03, 2.77843996e-04, 3.82819999e-05,
         7.88083980e-06, 9.19956983e-06]], dtype=float32)
```

この例では、カレントディレクトリにある「casup.ft5」という名前の CASUP コードの入力ファイル（入力機番 5）のファイルを読み込み、核種 ID を表す変数「MCODE」と対応する原子数密度を表す変数「AN」のデータを取り出している。MARBLE では従来システムに含まれている解析コードの変数名を使ってデータを管理するというルールを使っている。このため、解析コードのマニュアルを参照すれば詳細が分かるようになっている[7]。

先に述べたように入出力ファイルからデータを読み込むだけでなく、データを変更して書き出すこともできる。例えば、ナトリウムボイド反応度の解析でボイド領域の実効断面積を計算するために、ナトリウムの原子数密度をゼロと見なせる値に変更する場合には以下のようにしてデータを変更したファイルを作成することができる。

```
>>> cas5 = CasupFort5(path="casup.ft5")
>>> mcode = cas5.mcode()
>>> mcode[0, 5]
11
>>> an = cas5.an()
>>> an[0, 5]
0.0093483999
>>> an[0, 5] = 1.0e-19
>>> cas5.set_an(an)
>>> print cas5
JOYO MK-1 70fuel Reg.FUELH
  1 1 1 0 1 0 0 0 1 0 0 1 0 1 99 0 0 0 0 0
  1 1 1 1 1 0 70 0
  1
 5.00000
0.000E+00
  523.150
 14
HOMO
   8 1.65356E-02 24 3.26604E-03 26 1.18917E-02 28 2.09279E-03
  42 2.35314E-04 11 1.00000E-19 925 1.57986E-03 928 5.22520E-03
 948 1.45701E-06 949 1.11473E-03 940 2.77844E-04 941 3.82820E-05
 942 7.88084E-06 951 9.19957E-06
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1 1 1 1 1 0 0
  2
```

[7]先述のオンラインマニュアルにも簡単な説明が加えられている。

```
 6 37
 1.000E-04 1.000E-05 10 600
FUELH
```

上記の操作を完全に理解するためには、Python の基本的な知識が必要になるが、ここでは核種 ID が 11 のナトリウムに対応する原子数密度を 1.0e-19 に変更し、CASUP の入力ファイルのフォーマットで書き出している（表示している）ということを理解できれば十分である。実際にはこの程度の処理であればエディタを使って手作業で修正した方が早いことが多いが、大量に処理しなければならないデータがある場合やファイルの読み書き以外に少し計算が必要になる場合にはプログラムに自動処理させた方が便利なことがある。

例えば、読み書き以外の計算処理が必要な具体的なケースとしては、JENDL-3.2 で天然組成として扱われていた核種を JENDL-3.3 や JENDL-4.0 で解析するために同位体ごとに分解して入力を作り直すような処理が考えられる。この場合、同位体にあわせて組成比を乗じるという計算が必要になるため、エディタと電卓を使うよりも簡単なプログラムを書く方が便利になると期待できる。また、MARBLE のカプセル化はバイナリファイルにも対応しているので CITATION の中性子束の計算結果を取り出すような場合にも利用することができる。

5.2.3 まとめ

一度でも従来システムの入力ファイルを作成するプログラムを書いたり、出力ファイルから特定のデータを読み出すプログラムを書いたりしたことのあるユーザは、MARBLE のこれらの機能を活用できる可能性がある。一方、従来システムを使っていても、このような処理を必要としないユーザも多いと考えられる。そのようなユーザは、MARBLE ではこのような方法で従来システムがそのまま再利用されているということを理解するだけで十分である。

本節で紹介した MARBLE の機能に興味のある読者は marble.capsule 以下に含まれている部品について調査する価値があると思われる。現在、MARBLE が対応している主な解析コードは以下のとおりである。

- 格子計算コード SLAROM
- 格子計算コード CASUP
- 超微細群格子計算コード SLAROM-UF
- 拡散計算コード CITATION-FBR
- 輸送計算コード TRITAC
- 拡散摂動計算コード PERKY
- 輸送摂動計算コード SNPERT3D

また、主なファイル形式として以下のものにも対応している。

- 高速炉用炉定数セット JFS-3

- 高速炉用超微細群炉定数セット UFLIB
- SLAROM/CASUP の実効断面積ファイル PDS
- SLAROM-UF の実効断面積ファイル NewPDS

例えば、MARBLE から SLAROM-UF を制御する機能について知りたい場合は以下のオンラインマニュアルを参照することができる。

```
% pydoc marble.capsule.slaromuf.SlaromUfUtils
```

なお、本節で紹介した機能は PDS ファイルと NewPDS ファイルも対応しているが、PDS ファイルや NewPDS ファイルが扱う断面積データについては前節で紹介した「CrossSection」の方がより高度な機能を持っているため「CrossSection」を使う方が簡単である。また、PDS ファイルや NewPDS ファイルの内容を確認したいといった単純な処理であれば、後述する高速炉核特性解析システム SCHEME で整備されている。

5.2.4 補足：テスト駆動開発

これまでに紹介したような小さい部品を使うユーザにとっては、これらの部品が作られる過程で作成されたテスト問題を参照することが役に立つ可能性がある。MARBLE はテスト駆動開発と呼ばれる方法で開発されているため、部品には単体テストと呼ばれるテスト問題が用意されている。単体テストでは部品がどのように動作するかをチェックしているので、単体テストを読むことで部品が具体的にどのように動作するかを知ることができる。単体テストは marble/test 以下に保存されている。

5.3 MARBLEの高速炉核特性解析機能

5.3.1 MARBLEフレームワークと解析システム

前節までに説明した小さな部品を直接利用する方法はMARBLEの本質的な利用方法のひとつであるが、実際にはこのような利用方法を必要としないユーザも多いと考えられる。このため、MARBLEでは、従来システムに含まれる解析コードの機能に相当するような「実効断面積を計算する」、「拡散近似で炉心計算をする」、「厳密摂動理論で反応度計算をする」といったよりまとまった機能を提供する部品も提供している。これらの機能は部品と呼ぶよりも解析システムと呼んだ方がわかりやすいと考えられるので、ここでは「MARBLEに含まれる解析システム」あるいは単に「解析システム」と呼ぶことにする。

現在のバージョンのMARBLEには以下の２つの解析システムが含まれている。

- 高速炉核特性解析システムSCHEME
- 高速炉実機燃焼解析システムORPHEUS

SCHEMEは従来システムJOINT-FRと同等の解析機能を持った解析システムである。SCHEMEはJOINT-FRと同様に高速炉の燃焼を伴わない核特性（臨界性、反応度等）に対する詳細解析を目的とする解析システムである。JOINT-FRは解析コードをつなぎあわせることで様々な核特性の解析に柔軟に対応できるという特徴を持っているため、SCHEMEでもこの特徴を活かしている。具体的には、SCHEMEでは解析手順全体を自動化することは避け、ユーザが解析コードをつなぎあわせて使う機能を提供しつつ、入力作成を簡略化して解析を容易に実施できるようにしている。

一方のORPHEUSは従来システムJOINT-FRが苦手としていた高速炉実機の燃焼を伴う詳細解析を得意とする解析システムである。ORPHEUSは実機解析で不可欠な燃料交換や制御棒操作にも対応しており、高速炉実機の核特性解析に対してJUPITER標準解析手法[10,11]と呼ばれる詳細解析手法を適用することができる。

MARBLEを使って核特性解析をする場合、通常はSCHEMEかORPHEUSのどちらかを選択して使うことになる。これらの解析システムについては第4章で述べる。

5.4 高速炉核特性解析システム SCHEME の利用方法

5.4.1 断面積データファイルの利用方法

SCHEME も MARBLE の部品のひとつなのでこれまでの紹介のように対話的に使うことができる。最初に第 3 節の最後に紹介した NewPDS ファイルから断面積データを取り出す機能を使う方法を示す。この機能は従来システムのユーティリティプログラム PDSDUMP の機能に相当する[8]。SCHEME では `macro_from_newpdsfile` というコマンド（関数）[9]を使う。

```
% python
>>> from marble.scheme.jointfr import macro_from_newpdsfile
>>> macro = macro_from_newpdsfile(path="./NEWPDS", name="FUEL", icase=-1)
>>> macro.rtypeids()
['nu_fission', 'n2n', 'n3n', 'capture', 'total_scattering', 'fission_spectrum', 'current_weight_transport'
    , 'n4n', 'parallel_diffusion_coefficient', 'fission', 'current_weight_total', '
    perpendicular_diffusion_coefficient', 'current_weight_p1_total_scattering', 'absorption', '
    total_scattering_matrix', 'flux_weight_total', 'inelastic_scattering', 'elastic_scattering', '
    current_weight_p1_elastic_scattering', 'average_diffusion_coefficient', 'nu']
>>> macro.get("fission")
array([ 0.01123506, 0.00958261, 0.0068759 , 0.00724979, 0.0073198 ,
       0.00757654, 0.00759632, 0.00646707, 0.00476829, 0.00430156,
       0.00392999, 0.00377523, 0.00371826, 0.00373923, 0.00377256,
       0.00381343, 0.00392227, 0.00412102, 0.00425642, 0.00444188,
       0.0046351 , 0.00484825, 0.00509995, 0.00520333, 0.00558723,
       0.00589231, 0.00631141, 0.00681495, 0.00749956, 0.00802208,
       0.00913359, 0.01056371, 0.01205111, 0.01238233, 0.01417928,
       0.01612744, 0.02020642, 0.01974429, 0.0223255 , 0.03771554,
       0.02705657, 0.03514275, 0.0524177 , 0.0477461 , 0.04940087,
       0.0512013 , 0.07749395, 0.0735968 , 0.13782591, 0.06540119,
       0.0686902 , 0.08391147, 0.10715213, 0.13758065, 0.14604248,
       0.11710739, 0.10353771, 0.05237408, 0.02629122, 0.05765083,
       0.03443268, 0.03913053, 0.0454754 , 0.09032088, 0.16150197,
       0.15474665, 0.22565073, 0.42018044, 1.42285526, 1.62224531], dtype=float32)
```

この例では、カレントディレクトリにある「NEWPDS」という名前の NewPDS ファイル[10]の「FUEL」というメンバー名のデータを読み込んでいる。ここで指定している「icase」は NewPDS ファイルを作成したときの SLAROM-UF コードの入力オプション ICASE である[11]。最終的に NewPDS ファイルに含まれている反応種別を確認して、核分裂断面積データを表示している。反応種別については付録 B を参照することができる。

5.4.2 解析コードの実行方法

次に実際に解析コードを実行する方法を示す。SCHEME は対話的に利用できるが、同様の処理をファイルに保存しておいてバッチ処理的に利用することも可能である。本格的な解析作業では多くのデータを扱うので、先にファイルを用意するのが良いと考えられる。ここでは格子計算コード SLAROM-UF を使って実効断面積を計算する方法を示す。以下は内側炉心の実効断面積を均質モデルで計算するための入力データである。SCHEME ではこのように入力データと計算

[8]実際には PDSDUMP が対応しているのは NewPDS ファイルではなく PDS ファイルである。従来システムでは PDS2PDS で旧形式の PDS ファイルに変換してから PDSDUMP を使う必要がある。

[9]() の中が関数の引数（入力）で左辺が返値（出力）を表す。

[10]NewPDS の構造上、実体はディレクトリである。

[11]NewPDS ファイルに格納されている輸送断面積の種類を特定するために指定する必要がある。

手順（利用する解析コードやファイルの処理）をひとまとめに記述する[12]。

```
1  from marble.scheme.jointfr import (
2      materials,
3      homocell,
4      slaromuf)
5
6  mats_inner_core = materials(yaml="""
7  - Material:
8      name: inner_core_material
9      data:
10       U-235: 1.09804e-05
11       U-238: 5.41001e-03
12       Pu-239: 9.01750e-04
13       Pu-240: 3.71581e-04
14       Pu-241: 2.15854e-04
15       Pu-242: 6.14172e-05
16       O-16: 1.37340e-02
17       Na-23: 8.75367e-03
18       Cr-nat.: 3.87561e-03
19       Mn-55: 3.66807e-04
20       Fe-nat.: 1.38604e-02
21       Ni-nat.: 2.72666e-03
22       Mo-nat.: 3.08888e-04
23 """)
24
25 cell_inner_core = homocell(yaml="""
26 HomoCell:
27   name:
28     inner_core
29   matnames:
30     [inner_core_material]
31 """)
32
33 mac70g_inner_core, mic70g_inner_core = slaromuf(
34     cell=cell_inner_core, materials=mats_inner_core,
35     prep_ibsw=-1,
36     prep_te=473.15)
37
38 dump_to_file(mac70g_inner_core, "WORK/mac70g_inner_core.pkl")
39 dump_to_file(mic70g_inner_core, "WORK/mic70g_inner_core.pkl")
```

最初の 1〜4 行目は SCHEME のどのコマンド（関数）を使うかを伝えている。ここでは組成データを定義するコマンド `materials`、均質セルを定義するコマンド `homocell`、SLAROM-UF コードを実行するコマンド `slaromuf` を使うということを伝えている[13]。

6〜23 行目では `materials` コマンドを使って内側炉心の均質組成データを定義している。入力として指定している「yaml」は YAML 形式で書かれたファイルを使うことを意味している。YAML 形式は「字下げ（インデント）」と「キーワード（`:`）」を使うことで名前と階層を持ったデータを定義することができる。また、「`-`（ハイフン）」や「`[]`（カギ括弧）」と「`,`（コンマ）」を使うことでデータの列挙ができる。YAML 形式は、インデントやリスト、キーワードのような一般的な概念でデータ構造を表現するので、特にルールを知らなくてもある程度意味を理解できるようになっている。この例では「inner_core」という名前の均質組成データ（Material）をひとつだけ（最初のハイフン）定義している。

[12] SCHEME は Python のプログラムであるので、Python の仕組みを利用すれば分割することもできる。
[13] 厳密には marble.scheme.jointfr パッケージに定義された関数を import している。

25〜31 行目では `homocell` コマンドを使って均質セルモデルを定義している。ここでも YAML 形式のデータが利用されており、「inner_core」という名前のセルを定義している。この例は均質モデルなので組成はひとつだけ必要であり、組成データとして先に定義した「inner_core_material」を使うように指示している。

33〜36 行目では `slaromuf` コマンドを使って実際に SLAROM-UF を実行している。入力には先に定義した組成データと均質セルモデルを使っている。`slaromuf` コマンドの入力として指定している「prep_ibsw」と「prep_te」は SLAROM-UF の入力データであり、それぞれ、SLAROM-UF コードの PREP モジュールの入力オプションである IBSW と TE を意味している[14]。この例から分かるように、SCHEME では解析コードの入力オプションをすべて入力する必要はない。推奨オプションが自動的に設定されるようになっており、ユーザが変えたいオプションを変数名とともに指定することでオプションを変更することができる。

最後に `dump_to_file` コマンドを使って、SLAROM-UF で計算したマクロ断面積データとミクロ断面積データをファイルに保存している[15]。

以上で SCHEME を使って SLAROM-UF を動かす準備は整ったことになる。この入力データを「myinput.py」という名前で保存すれば以下のようにして計算を実行することができる。

```
$ python myinput.py
```

これまでの説明からも分かるように、SCHEME は先に紹介した小さい部品と同じものである。ここでコマンドと呼んでいるものは Python の関数オブジェクトに相当する。SCHEME の入力データが Python プログラムであることを理解すれば、より柔軟に使うことができるが、通常は単なる入力データ形式であると理解しておくだけで十分である。

5.4.3 まとめ

SCHEME の基本的な使い方を説明した。実際に解析を進めていくためには、SLAROM-UF で作成した実効断面積データを炉心計算コードや摂動計算コードに受け渡しながら解析コードを実行していくことになるが、基本的な操作は SLAROM-UF の例で示したものと同じである。CITATION、TRITAC、PERKY、SNPERT3D を使うためのコマンドとしては、それぞれ、`citation`、`tritac`、`citation_perky_for_reactivity`、`tritac_sneprt3d` 等が用意されている[16]。また、詳細については付録 H を参照することができる。

[14] 詳細は SLAROM-UF のマニュアルを参照することができる。
[15] このファイルは Python の pickle というモジュールで作成されるバイナリファイルになっているため直接エディタ等では読みとることができない。再度読み込むには load_from_file コマンドを使う。
[16] オンラインマニュアルは「pydoc marble.scheme.jointfr」で参照できる。

5.4.4　補足：解析コードのオプション指定方法と入出力ファイル

5.4.4.1　オプション指定方法の特殊なケース

先述のように解析コードのオプションを変更したい場合には、対象とする解析コードの入力オプションの変数名を使って指定するが、SCHEME ではPERKY、SNPERT3D はそれぞれCITATION、TRITAC とあわせて実行するようになっている[17]ため、どちらのコードのオプションかが曖昧になる。このようなコマンドでは２つの解析コードを同時に実行するため、それぞれの解析コードに対してオプションを指定できるように、オプション（変数）名の前に「citation_」、「perky_」をつけるようになっている。例えば、`citation_perky_for_reactivity` コマンドで、CITATION の NUAC18 オプションを 1 にしたい場合は「citation_nuac18=1」のように指定する。

5.4.4.2　解析コードの入出力ファイルの参照方法

従来システムに慣れたユーザは自動設定された解析コードのオプションや自動生成された入出力ファイルを確認したい場合があると考えられる。

このような場合には、解析コードを実行するコマンドに `debug` オプションを渡せばよい。SCHEME（MARBLE）が解析コードを実行する際には、MARBLE の WORK ディレクトリに解析コードの入出力ファイルを生成しており、必要に応じてこれらファイルを残すことができるようになっている。また、SCHEME の解析コードを実行するコマンドの `debug` オプションに WORK ディレクトリを指定することで一時的に WORK ディレクトリの場所を変更することもできる。例えば、「debug=”.”」と指定するとカレントディレクトリに「marble.username. コード名. 実行日時」のようなディレクトリが生成される。このディレクトリの中に解析コードの入出力ファイル一式が残るので、必要に応じてこれらのファイルを参照することができる。

また、ほとんどの解析コードでは重要な情報は通常ファイル機番 6 にまとめて出力し（SCHEME ではこのファイルのことを「outlist」と呼ぶ）、多くの場合、入力データの打ち返しがあるので、outlist ファイルを参照するという方法もある。解析コードの出力ファイルを保存するには、解析コードを実行するコマンドに `outlist` オプションに outlist のファイル名を指定する。例えば、出力ファイルをカレントディレクトリの「codename.outlist」という名前で保存したい場合は、「outlist=”codename.outlist”」と指定すればよい。

5.5　高速炉実機燃焼解析システム ORPHEUS の利用方法

本章の最後として ORPHEUS の利用方法を説明する。ただし、ORPHEUS についてはこれまでのような MARBLE 特有の説明をする必要がほとんどない。ORPHEUS ではシステムで決められた一連の処理を自動的に実行するので、SCHEME のようにユーザが計算手順を定義する必要

[17]PERKY の入力データは CITATION の入力データと整合させる必要があり、入力データを自動生成するためにこのような形式にしている。SNPERT3D については分離できる可能性があるが、PERKY にあわせて同じ形式にしている。

はない。基本的にユーザは入力データを準備して実行するだけで解析を実施することができるので、使い方は従来システムと同じである。

ユーザが準備しなければならない ORPHEUS の入力データファイルは以下の 5 種類である。

- 炉心特性ファイル（CorePropertyFile）
- 幾何形状ファイル（GeometryFile）
- 物質組成ファイル（MaterialFile）
- 装荷パターンファイル（PatternFile）
- 計算条件ファイル（CalcConditionFile）

これらのファイルはすべて YAML 形式になっており、データ項目に名前が付いているため、解析対象や解析手法を知っていればある程度内容が想像できるようなものとなっている。以下では実際にファイルの中身を示して概要を説明する。

5.5.1 炉心特性ファイル

まず最初は炉心特性ファイルである。

```
name : MONJU
power: 714.0 ## MWth

assembly:
  layer : 15
  except: [15A1+]

...

control_rod:
  address:
    ["000",
      3A2+,
      5A5+,
      6A3+]
  bank_group:
    fcr: [5A5, 5C5, 5E5]
    ccr: ["000", 3A2, 3C2, 3E2, 6A3+]
    bcr: [3B2, 3D2, 3F2, 5B5, 5D5, 5F5]
```

名前のとおり、炉心特性ファイルでは炉心の熱出力、大きさ、制御棒の装荷位置等の炉心の基本的な情報を定義する。炉心位置の指定には、常陽、もんじゅで利用されている「1A1」のような炉心アドレス方式が採用されている。

5.5.2 幾何形状ファイル

続いて幾何形状ファイルの例を示す。

```
primitive:
  - name : assembly_outer_boundary
    type : hex
    angle: 0.0
```

```
    value: 11.56

segment:
  - name: inner_core_fuel_segment
    boundary: assembly_outer_boundary
    composition:
      - name : inner_core_fuel_homogenized_material
        region: [+inner_core_fuel_segment]

....

assembly:
  - name: inner_core_fuel_assembly
    composition:
      - axial_range: [198.00, 238.00]
        segment : upper_core_reflector_segment
      - axial_range: [168.00, 198.00]
        segment : axial_blanket_segment
      - axial_range: [ 75.00, 168.00]
        segment : inner_core_fuel_segment
      - axial_range: [ 40.00, 75.00]
        segment : axial_blanket_segment
      - axial_range: [ 0.00, 40.00]
        segment : lower_core_reflector_segment
```

このファイルでは、燃料ピッチ 11.56cm の六角形の内側燃料集合体を均質モデルで定義している。まず最初に「primitive」では燃料集合体の断面の幾何形状を指定するために対面間距離 11.56cm の六角形を定義している。続いて「segment」では燃料集合体の水平断面を定義しており、先ほど定義した六角形に当てはめる組成を設定している。最後に「assembly」において定義したセグメントを使って燃料集合体の軸方向の構造を決定している。この例では簡単のため均質モデルを使っているが、円や六角形を組み合わせて燃料集合体の構造を忠実にモデル化することができるようになっている。

5.5.3 物質組成ファイル

次は物質組成ファイルであるが、先ほどの幾何形状ファイルで指定した組成データを定義するファイルである。物質組成ファイルについては SCHEME で使われる組成データと同じであるので説明を省略する。

5.5.4 装荷パターンファイル

続いてパターンファイルの例を示す。

```
core_name : MONJU High-Burnup Core
cycle_name : cycle 1
cycle_number: 1

fuels:
  - label: ICF?????
    type : inner_core_fuel_assembly

  - label: OCF?????
    type : outer_core_fuel_assembly

reflectors:
```

```
  - label: RDF?????
    type : radial_reflector_assembly

control_rods:
  - label: FCR?????
    type : fine_control_rod_assembly

  - label: CCR?????
    type : coarse_control_rod_assembly

  - label: BCR?????
    type : backup_control_rod_assembly

reactor_outside: radial_reflector_assembly

load_fuels:
  - { address: 1A1, label: ICF01A01 }
  - { address: 1B1, label: ICF01B01 }
  - { address: 1C1, label: ICF01C01 }
  - { address: 1D1, label: ICF01D01 }
  - { address: 1E1, label: ICF01E01 }
  - { address: 1F1, label: ICF01F01 }
  - { address: 2A1, label: ICF02A01 }
  - { address: 2A2, label: ICF02A02 }
  - { address: 2B1, label: ICF02B01 }
....
```

装荷パターンファイルは集合体の装荷パターンを定義するためのファイルでサイクル毎に与える必要がある。このように集合体毎の装荷状況を個別に管理して解析することができるようになっている。また、装荷パターンファイルでは、集合体の装荷パターンに加えて炉心計算で使うミクロ断面積の指定（zone）も行う。

5.5.5 計算条件ファイル

最後に計算条件ファイルの例を示す。

```
calc_mode : standard

title : MONJU Burnup Calculation
core_name : MONJU High-Burnup Core
case_name : 70g Calculaion for Group Condensation

cycle:
  name : 0 cycle
  number : 0

file:
  input:
    core_property: $PWD/monju.coreproperty
    pattern : $PWD/monju_1cyc.pattern
    geometry : $PWD/monju.geometry
    material : $PWD/monju.material
  output:
    raytrace : $PWD/output/monju_70g.out.raytrace
    database : $PWD/output/monju_70g.out.database
    summary : $PWD/output/monju_70g.out.summary
    list : $PWD/output/monju_70g.out.list
    debug : $PWD/output/monju_70g.out.debug

calc_system:
  coordinates:
    core : hexz
```

```
    burnup: hexz

....

solver:
  xs:
    name : slarom_uf
    library: JENDL-3.2
    ng : 70
    options:
      index: $PWD/Index.g70
      ibsw : 2 ## buckling search: on
      itpe : 0 ## iteration of background cross section (0/1=No/Yes)
      te : 473.15

  core:
    name : marble_citation
    options:
      fs_zone: 1
      region_wise_fission_spectrum: on
      ngc : [0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
      iedg : [1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
      itmx : [900, 900]
      isodf: 1
      ixdct: 1 ## average_diffusion_coefficient
      iydct: 1 ## average_diffusion_coefficient
      izdct: 1 ## average_diffusion_coefficient
      ipunf: 7

  burnup:
    name: burnup
    options:
      chain: standard2006
      decay: standard2006

step:
  - period: 0.00 ## day
    power : 100.00 ## %
    control_rod:
      fcr: 100.00
      ccr: 100.00
      bcr: 110.00

....
```

計算条件ファイルではまず最初に計算モードを指定し、他の入力ファイルと出力ファイルを指定することで計算体系の情報を定義する。続いて、実効断面積計算、炉心計算、燃焼計算で使うオプションと燃焼計算ステップについて指定する。これにより計算条件をすべて指定することができる。

上記の５つのファイルが揃えば実行の準備が完了したことになる。

5.5.6 実行方法

ORPHEUSの実行するためにはORPHEUSのmainパッケージに含まれるControllerモジュールを使う。このモジュールは計算条件ファイル名を引数として直接実行することが可能である。例えば、カレントディレクトリにMARBLE本体（marbleディレクトリ）とORPHEUSの計算条件ファイル（myinput.calccondition）がある場合、以下のようにすることでORPHEUSを実行することができる。

```
\$ python marble/orpheus/main/Controller.py
```

計算が正常に終了すると計算条件ファイルに指定した出力ファイルが生成される。ORPHEUS のユーザマニュアルは付録 I に添付されている。また、ORPHEUS については開発内容を詳細に記載した一連の報告書が発行されているので、ORPHEUS の詳細についてはこれらの報告書[68–72]を参照することができる。

5.6 まとめ

本章では MARBLE の使い方をいくつか紹介した。高速炉の実機体系を対象とした燃焼を含む核特性解析をしたいユーザは ORPHEUS を利用することができる。また、従来システムをある程度使用した経験のある場合には SCHEME から使い始めることで MARBLE の方式に慣れることができると考えられる。従来システムの詳細を熟知して使いこなしているユーザには、SCHEME のようなシステムでもまだ大きすぎて使いにくい場合があると考えられる。このような場合には MARBLE フレームワークの中の小さな部品を直接取り出して必要な機能を組み立てて使うということが考えられる。

6. おわりに

高速炉核特性詳細解析手法に基づく新しい解析システムとして次世代炉心解析システムMARBLEを開発した。

MARBLEの開発では、オブジェクト指向技術やインターネットを利用した共同開発環境を導入し、アジャイル開発と呼ばれる現代的なソフトウェア開発手法を適用した。MARBLEの開発を通してこれらの技術は炉心解析の分野においても有効であることを確認できた。これらの技術は柔軟性や拡張性といったソフトウェアの性能の向上に効果があるだけでなく、互いに有機的に連携して機能することでコードの検証を確実にし、複数の開発者による共同作業を効率的に進める効果も持っている。また、MARBLEの開発に先立って行われた工学系モデリング言語としての次世代解析システムの検討に基づき、スクリプト言語としてPython、システム言語としてFORTRAN、C/C++を採用した二階層システムを適用した。これにより今後の開発ではスクリプト言語やオブジェクト指向言語の特徴を利用した開発を進めることができる。これらの結果として、MARBLEは解析システムや解析手順を構築するための再利用可能な部品の集まり（フレームワーク）となっている。

MARBLEフレームワークでは、従来システムに組み込まれている炉心解析で必要となるデータ管理機能を抽出し、再利用可能な部品として整備した。また、従来システム（JOINT-FR、SAGEP-FR）に含まれるほぼすべての解析コードを書き換えずに再利用する仕組みを確立した。これにより従来システムに含まれる各解析コードの結果を完全に再現できるようにするとともに、従来システムの解析機能を解析コード毎に分解してより柔軟に利用できるようになった。

より具体的な解析機能としては、MARBLEフレームワーク上で高速炉核特性解析システムSCHEMEを開発することにより、JOINT-FRに相当する高速炉核特性の詳細解析をより簡便に行うことが可能となった。また、従来システムでは提供されていなかった高速炉実機燃焼解析に必要な部品をMARBLEフレームワーク上で新たに作成し、これらの部品を適用して高速炉実機燃焼解析システムORPHEUSを開発した。ORPHEUSを使うことで高速炉実機の運転状況を考慮した体系への詳細解析手法の適用が可能となった。

以上のように、MARBLEは従来システムの基本解析機能と拡張性に優れたソフトウェア基盤を持った解析システムである。今後は、従来システムの後継として、MARBLEフレームワークの拡張性を活用してより高度な解析手法やモデルを導入していくことができると考えられる。

参考文献

1) 横山賢治, 細貝広視, 宇都成昭, 笠原直人, 名倉文則, 大平正則, 加藤雅之, 石川 眞: “工学系モデリング言語としての次世代解析システムの開発 (I) – 課題および要素技術の調査 –”, JNC TN9420 2002-004 (2002 年 11 月).

2) 横山賢治, 細貝広視, 宇都成昭, 笠原直人, 石川眞: “工学系モデリング言語としての次世代解析システムの開発 (II) – プロトタイプ作成による検討 –”, JNC TN9400 2003-021 (2003 年 4 月).

3) 横山賢治, 細貝広視, 宇都成昭, 笠原直人, 石川眞: “工学系モデリング言語としての次世代解析システムの開発 (III) – プロトタイプ作成による検討 (その 2) –”, JNC TN9400 2004-022 (2004 年 4 月).

4) 千葉豪: “工学系モデリング言語としての次世代解析システムの開発 (IV) – 境界要素法に基づく中性子拡散ソルバーの開発 –”, JNC TN9400 2004-055 (2004 年 10 月).

5) 横山賢治: “工学系モデリング言語としての次世代解析システムの開発 (V) – 二階層システム移行のための既存コード再構築手法の検討 –”, JAEA-Research 2006-055 (2006 年 10 月).

6) 横山賢治: “工学系モデリング言語としての次世代解析システムの開発 (VI) – 炉定数調整・核設計精度評価ソルバーの開発 –”, JAEA-Data/Code 2007-023 (2008 年 1 月) .

7) K. Yokoyama, M. Ishikawa, M. Tatsumi and H. Hyoudou: “Restructuring of Burnup Sensitivity Analysis Code System by using an Object-Oriented Design Approach,” Proc. Conf. on Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications, Avignon, France, September 12–15, 2005 (M&C2005).

8) 巽 雅洋, 兵頭秀昭 : ” 燃焼感度解析コードのシステム化整備 “, JNC TJ9400 2003-012 (2004 年 2 月).

9) 巽 雅洋, 兵頭秀昭 : ” 燃焼感度解析コードのシステム化整備 (II)“, JNC TJ9410 2004-002 (2005 年 2 月)

10) M. Ishikawa, “Consistency Evaluation of JUPITER Experiment and Analysis for Large FBR Cores,” Int. Conf. on the Phyiscs of Reactor, PHYSOR96, September 16-20, 1996, Mito, Ibaraki, Japan (1996).

11) 石川 眞, 佐藤若英, 杉野和輝, 他: “核設計基本データベースの整備（VIII） – JUPITER 実験解析結果の集大成 –”, PNC TN9410 97-099 (1997 年 11 月).

12) 横山賢治, 羽様 平, 千葉 豪, 大木 繁夫, 石川 眞: “– 高速炉サイクルの研究開発を支える解析システム – V. 核特性解析コードシステムの開発と利用”, JNC Technical Review, No. 17, JNC TN1340 2002-003 (2002 年 12 月).

13) M. Ishikawa, K. Sugino, W. Sato and K. Numata: “Development of a Unified Cross-section Set ADJ2000 based on Adjustment Technique for Fast Reactor Analysis,” Proc. Int. Conf. on Nuclear Data for Science and Technology Oct. 7-12, 2001, Tsukuba, Ibaraki, Japan (ND2001), J. Nucl. Sci. Technol., Suppl. Vol.2, pp.1077-1080 (Aug. 2002).

14) 羽様 平, 千葉 豪, 沼田一幸, 佐藤 若英: “高速炉用統合炉定数 ADJ2000R の作成”、JNC TN9400 2002-064 (2002 年 11 月).

15) T. Hazama, G. Chiba and K. Sugino: “Development of a Fine and Ultra-Fine Group Cell Calculation Code SLAROM-UF for Fast Reactor Analyses,” J. Nucl. Sci. Technol. 43[8], pp.908-918 (2006).

16) T. Hazama, G. Chiba, W. Sato and K. Numata: “SLAROM-UF: Ultra Fine Group Cell CalculationCode for Fast Reactor – Version 20090113 –,” JAEA-Review 2009-003 (2009).

17) 石川 眞, 斎藤正幸, 三田敏男: “核設計基本データベースの整備 (IV) –核特性解析コードシステムの整備–”, PNC TN9440 94-004 (1994 年 3 月).

18) 中川正幸, 阿部純一, 佐藤若英: “高速炉の核特性解析コードシステム”, JAERI-M 83-066 (1983 年 4 月)

19) M. Nakagawa and K. Tsuchihashi,: “SLAROM : A Code for Cell Homogenization Calculation of Fast Reactor”, JAERI 1294 (Dec. 1984).

20) 竹田敏一, 谷本浩一, 小野俊治: “大型高速臨界集合体での中性子ストリ-ミング効果に関する研究 (II)“, PNC TJ265 82-01 (1982 年 3 月).

21) S. Ono, E. Wachi, T. Takeda and T. Sekiya: “CASUP: Cell Calculation Code for Fast Reactor Analysis,” Technology Reports of The Osaka University, Vol. 33, No. 1708, pp.207-219 (1983).

22) T. B. Fowler, D. R. Vondy and G. W. Cunningham: “Nuclear Reactor Core Analysis Code: CITATION”, ORNL-TM-2496, Rev.2, ORNL (Jul. 1971).

23) M. Bando, T. Yamamoto, Y. Saito and T. Takeda: “Three-Dimensional Transport Calculation Method for Eigenvalue Problems Using Diffusion Synthetic Acceleration,” J. Nucl. Sci. Technol. 22[10], pp.841-850 (1985).

24) 山本敏久: “炉心核特性詳細解析コード TRITAC の改良” PNC TN9410 95-069 (1995 年 4 月)

25) K. D. Lathrop and F. W. Brinkley: “TWOTRAN-II : An Interfaced, Exportable Version of the TWOTRAN Code for Two-Dimensional Transport,” LA-4848-MS, LANL (Jul. 1973).

26) 飯島 進, 吉田弘幸, 桜木廣隆: “高速炉設計用計算プログラム・２（2 次元・3 次元拡散摂動理論計算コード: PERKY”, JAERI-M 6993 (1977 年 2 月).

27) 佐々木誠, 市川真一: “「常陽」輸送コードシステムの作成（Sn 輸送コード・使用マニュアル）”, PNC TN952 81-08 (1981 年 8 月).

28) 原 昭浩, 竹田敏一, 菊地康之: “SAGEP: 一般化摂動論に基づく二次元感度解析コード”, JAERI-M 84-027 (1984 年 1 月).

29) 林 秀行, 永田 敬, 森山正敏, 石川 眞, 中大路道彦, 黒木修二, 山岡光明: “大型高速炉設計研究成果報告書 -60 万 KWe 級プラントの設計研究-”, PNC TN9410 92-137 (1992 年 5 月).

30) T. Takeda and T. Umano, “Burnup Sensitivity Analysis in a Fast Breeder Reactor - Part I: Sensitivity Calculation Method with Generalized Perturbation Theory,” Nucl. Sci. Eng., Vol.91, pp.1-10 (1985).

31) 花木 洋, 澤田周作, 三田敏男: “燃焼核特性に対する感度解析コードの整備”, PNC TJ9124 93-009 (1993 年 3 月).

32) 花木 洋, 三田敏男, 大橋正久: “燃焼核特性に対する感度解析コードの整備 (II)”, PNC TJ9124 94-007 (1994 年 3 月).

33) T. Takeda, A. Yoshimura, T. Kamei and K. Shirakata, “Prediction Uncertainty Evaluation Methods of Core Performance Parameters in Large Liquid-Metal Fast Breeder Reactors,” Nucl. Sci. Eng., Vol.103, pp.157-165 (1989).

34) C.Calvin: “DESCARTES: A New Generation System for Neutronic Calculations,” Int. Conf. on Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications (M&C 2005), Avignon, France, September 12-15, 2005, on CD-ROM (2005).

35) H. Golfier, R. Lenain, C. Calvin, J-J. Lautard, A-M. Baudron, Ph. Fougeras, Ph. Magat, E. Martinolli and Y. Dutheillet: “APOLLO3: A Common Project of CEA, AREVA and EDF for the Development of a New Deterministic Multi-Purpose Code for Core Physics Analysis,” Int. Conf. on Mathematics, Computational Methods & Reactor Physics (M&C 2009), New York, May 3-7, 2009, on CD-ROM (2009).

36) J. K. Ousterhout: “Scripting: Higher Level Programming for the 21st Century,” IEEE Computer Magazine, (Mar. 1998).

37) B. Meyer 著, 二木厚吉 監訳, 酒匂 寛, 酒匂順子 共訳: “オブジェクト指向入門”, アスキー出版局, ISBN4-7561-005-3 (1990).

38) Python Software Foundation: “The Python Programming Language”, available from `http://www.python.org/` (accessed 2010-11-25).

39) T. E. Oliphant, “Guide to NumPy”, available from `http://numpy.scipy.org/` (accessed 2010-11-25).

40) P. Peterson, “F2PY: Fortran to Python Interface Generator,” available from `http://cens.ioc.ee/projects/f2py2e/` (accesed 2010-11-25).

41) C. Evans, “YAML Ain’t Markup Language”, available from `http://www.yaml.org/` (accessed 2010-11-25).

42) Apache Software Foundation: “Apache Subversion: Enterprise-class centralized version control for the masses,” available from `http://subversion.apache.org` (accessed 2010-11-25).

43) Edgewell Software: “An Enhanced Wiki and Issue Tracking System for Software Development Projects,” available from `http://trac.edgewall.org/` (accessed 2010-11-25).

44) K. Beck, et al: “Manifesto for Agile Software Development,” available from `http://www.agilemanifesto.org/` (accessed 2010-11-25).

45) K. Beck 著, 長瀬嘉秀 監訳: “テスト駆動開発入門”, ピアソン・エデュケーション (2003 年).

46) M. Fowler 著, 児玉公信, 他 翻訳: “リファクタリング – プログラムの体質改善テクニック”, ピアソン・エデュケーション (2000 年).

47) 巽 雅洋, 横山賢治: “次世代炉物理解析システムのためのフレームワーク開発”, JAEA-Data/Code 2007-020 (2007 年 11 月).

48) 平井康志、兵頭秀昭、巽雅洋、神智之、横山賢治: “次世代炉物理解析システムのためのフレームワーク開発（その 2）”, JAEA-Data/Code 2008-020 (2008 年 10 月).

49) 横山賢治, 平井康志, 兵頭秀昭, 巽雅洋: “次世代炉物理解析システムのためのフレームワーク開発（その 3）”, JAEA-Data/Code 2009-012 (2010 年 2 月).

50) 横山賢治, 平井康志, 巽雅洋: “次世代炉物理解析システムのためのフレームワーク開発（その 4）”, JAEA-Data/Code 2010-015 (2010 年 11 月).

51) K. Shibata, O. Iwamoto, T. Nakagawa, N. Iwamoto, A. Ichihara, S. Kunieda, S. Chiba, K. Furutaka, N. Otuka, T. Ohsawa, T. Murata, H. Matsunobu, A. Zukeran, S. Kamada, and

J. Katakura: “JENDL-4.0: A New Library for Nuclear Science and Engineering” (to be published to J. Nucl. Sci. Technol.).

52) K. Shibata, T. Kawano, T. Nakagawa, O. Iwamoto, J. Katakura, T. Fukahori, S. Chiba, A. Hasegawa, T. Murata,H. Matsunobu, T. Ohsawa, Y. Nakajima, T. Yoshida, A. Zukeran, M. Kawai, M. Baba, M. Ishikawa, T. Asami, T. Watanabe, Y. Watanabe, M. Igashira, N. Yamamuro, H. Kitazawa, N. Yamano and H. Takano: “Japanese Evaluated Nuclear Data Library Version 3 Revision-3: JENDL-3.3,” J. Nucl. Sci. Technol. 39[11], p.1125-1136 (2002).

53) T. Nakagawa, K. Shibata, S. Chiba, T. Fukahori, Y. Nakajima, Y. Kikuchi, T. Kawano, Y. Kanda, T. Ohsawa, H. Matsunobu, M. Kawai, A. Zukeran, T. Watanabe, S. Igarashi, K. Kosako and T. Asami: “Japanese evaluated nuclear data library, version 3 revision-2; JENDL-3.2,” J. Nucl. Sci. Technol. 32[12], pp.1259-1271 (1995).

54) M. B. Chadwick, et al.: “ENDF/B-VII.0: Next Generation Evaluated Nuclear Data Library for Nuclear Science and Technology,” Nuclear Data Sheets, Volume 107, Issue 12, pp.2931-3060 (2006).

55) A. Koning, R. Forrest, M. Kellett, R. Mills, H. Henriksson and Y. Rugama: “The JEFF-3.1 Nuclear Data Library,” JEFF Report 21, NEA Data Bank, ISBN 92-64-02314-3 (2006).

56) T. Tone: “Numerical Study of Heterogeneity Effects in Fast Reactor Critical Assemblies,” J. Nucl. Sci. Technol., 12[8], pp.467-481 (1975).

57) T. Kitada, S. Kosaka and T. Takeda: “New Control Rod Homogenization Method for Fast Reactors,” J. Nucl. Sci. Technol. 31[7], pp.647-653 (1994).

58) H. Ikeda and T. Takeda: “A New Nodal S_N Transport Method for Three-Dimensional Hexagonal Geometry,” J. Nucl. Sci. Technol. 31[6], pp.497-509 (1994).

59) K. Sugino and T. Takeda: “An Iprovement of the Transverse Leakage Treatment for the Nodal S_N Transport Calculation Method in Hexagonal-Z Geometry,” J. Nucl. Sci. Technol. 33[8], pp.620-627 (1996).

60) 杉野和輝: “3 次元 HexZ 体系用ノード法輸送計算コード NSHEX の実用性向上のための改良”, PNC TN9410 98-064 (1998 年 7 月).

61) K. Sugino and T. Kugo: “Effect of Polynomial Expansion Order of Intra-node Flux Treatment in Nodal Sn Transport Calculation Code NSHEX for Large-size Fast Power Reactor Core Analysis,” J. Nucl. Sci. Technol. 48[3], pp.1-8 (2011).

62) K. L. Derstine: "DIF3D: A Code to Solve One-, Two-, and Three-Dimensional Diffusion Theory Problems," Argonne-82-64, Argonne National Laboratory (1984).

63) G. Chiba and Y. Shimazu: "Sodium void reactivity worth calculations for fast critical assemblies without whole-lattice homogenization. J. Nucl. Sci. Technol. 44[12], pp.1526-1534 (2007).

64) A. Yamamoto, et al.: "Numerical Solution of Stiff Burnup Equation with Short Half Lived Nuclides by the Krylov Subspace Method," J. Nucl. Sci. Technol., Vol. 44, No.2, pp.147-154 (2007).

65) A. G. Croff: "ORIGEN2 - A Revised and Updated Version of the Oak Ridge Isotope Generation and Depletion Code," ORNL-5621, Oak Ridge National Laboratory (Jul. 1980).

66) B. Dawes, D. Abrahams, et.al: "Boost C++ Library", available from `http://www.boost.org` (accesed 2010-11-25).

67) T. Veldhuizen: "Blitz++ User's Guide," available from `http://www.oonumerics.org/blitz/` (accessed 2010-11-25).

68) 兵頭秀昭, 巽 雅洋, "高速炉実機燃焼解析システムの開発", JAEA-Data/Code 2006-018 (2006 年 8 月).

69) 平井康志, 兵頭秀昭, 巽 雅洋, "高速炉実機燃焼解析システムの開発（その 2）", JAEA-Data/Code 2007-019 (2007 年 11 月).

70) 平井康志, 兵頭秀昭, 巽雅洋, 横山賢治, "高速炉実機燃焼解析システムの開発（その 3）", JAEA-Data/Code 2008-021 (2008 年 10 月).

71) 横山賢治, 平井康志 兵頭秀昭, 巽雅洋, "高速炉実機燃焼解析システムの高度化", JAEA-Data/Code 2009-016 (2010 年 2 月).

72) 横山賢治, 平井康志, 巽雅洋, "高速炉実機燃焼解析システムの高度化（その 2）", JAEA-Data/Code 2010-016 (2010 年 10 月).

73) R. B. Sidje: "Expokit: A Software Package for Computing Matrix Exponentials", ACM Trans. Math. Softw., 24(1): pp.130-156, (1998).

74) The SciPy community: "SciPy Reference Guide," available from `http://www.scipy.org/` (accesed 2010-11-25).

75) K.Okumura, T.Kugo, K.Kaneko and K.Tsuchihashi: "SRAC2006: A Comprehensive Neutronics Calculation Code System," JAEA-Data/Code 2007-004 (Feb. 2007).

76) J. B. Dragt, J. W. Dekker, H. Gruppelaar and A. J. Janssen: "Method of Adjustment and Error Evaluation of Neutron Capture Cross Sections; Application to Fission Product Nuclides", Nucl. Sci. and Eng. Vol. 62, p. 117 (1977).

77) T. Takeda, A. Yoshimura, T. Kamei and K. Shirakata: "Prediction Uncertainty Evaluation Methods of Core Performance Parameters in Large Liquid-Metal Fast Breeder Reactors", Nucl. Sci. and Eng. Vol.86, p.121 (1984).

78) T. Sano and T. Takeda: "Generalized Bias Factor Method for Accurate Prediction of Neutronics Characteristics", J. Nucl. Sci. Technol., 43[12], pp.1465-1470 (2006).

79) T. Kugo, T. Mori and T. Takeda: "Theoretical Study on New Bias Factor Methods to Effectively Use Critical Experiments for Improvement of Prediction Accuracy of Neutronic Characteristics," J. Nucl. Sci. Technol. 44[12], pp.1509-1517 (2007).

80) K. Yokoyama, K. Numata, T. Hazama and M. Ishikawa: "Development of Neutronics Design Accuracy Evaluation Solver for Next Generation Reactor Physics Analysis Code System MARBLE," Proc. of 16th Pacific Basin Nuclear Conference (16PBNC), Aomori, Japan, Oct. 13-18, 2008, P16P1168 (2008).

付録A　MARBLEにおける核種の指定方法

従来システムでは核種を表す識別子（ID）として、JFSIDと呼ばれる3桁の整数が用いられている。解析で取り扱う核種が少ないうちは3桁で十分であるが、評価済み核データライブラリJEDNL-3.3以降では天然組成の元素を同位体毎に陽に扱うようになったり、燃焼チェーンを詳細化するようになったりしたことで、取り扱う核種が増え、3桁ではわかりやすい核種IDを定義するのが難しくなってきている。このため、MARBLEではORIGEN2で用いられている核種IDに似た最大7桁の整数による核種IDを採用している。また、一般ユーザ向けには文字列を使った原子記号による表記法を核種IDとして使えるようになっている。なお、JFSIDも核種IDとして利用することはできるが、JFSIDは炉定数セットによって一部の核種IDが変わるのでJFSIDをMARBLEの核種IDとして使うことは推奨されない。

A.1　MARBLEの核種ID

MARBLEにおける核種IDは以下のように定義される。

```
Nuclide ID = 1000000 * F
           +   10000 * Z
           +      10 * A
           +       1 * M
```

Z: 原子番号
A: 質量数（天然組成の場合は0）
M: 核異性体インデックス
   (stable, m1, m2, m3, ... = 0, 1, 2, 3, ...)
F: ランプ化FPインデックス
   (FP_GAS_RELEASE, FP = 7, 8)

Nuclide ID > 2000000(=2E+6)を疑似核種用のIDとする。なお、ランプ化FPは疑似核種の一種であるのでこのルールに従っていることになる。

以下に典型的な例を示す。

この核種IDは処理速度（核種IDの検索）の向上のために導入されたものであり、実際にはユーザはこの核種IDを直接利用する必要はない。次節で説明する核種名をユーザは実質的な核種IDとして利用することができる。

A.2 MARBLEの核種名

上記の整数を使ったIDは定義がはっきりしているので、3桁のJFSIDよりは使いやすいと考えられるが、核種の原子番号やランプ化FPインデックスを記憶しておく必要がある。このため、MARBLEでは、文字列を使った原子記号による表記法を核種IDとして使えるようになっている。ただし、解析システムの入力データ等で用いるテキストデータでは、上付き文字、下付き文字を表現するのが難しいのでハイフンを用いて「元素記号-質量数」で表記する。

表A.2.1に典型的な核種の核種名と核種IDを示す。表に示されているように例えば、水素1は「H-1」、プルトニウム239は「Pu-239」となる。準安定状態の核種についても「m」を添えることで表現でき、例えば、準安定状態のアメリシウム242は「Am-242m」となる。天然組成元素については、「質量数」の代わりに「-nat.」とする。例えば、鉄の天然組成を表す核種名は「Fe-nat.」となる。ランプ化FPについては、JFS-3-J3.2R等で使われている表記を踏襲し、核種名の後に「FP」（ガス放出を考慮しないモデル）または「FPGR」（ガス放出モデル）を付加する。例えば、Pu-239から生成する核分裂生成物の場合、ガス放出を考慮しないモデルでランプ化した疑似核種は「Pu-239FP」、ガス放出モデルでランプ化した疑似核種については「Pu-239FPGR」と表記される。

表 A.2.1　MARBLEにおける核種名と核種IDの例

核種	核種名	核種ID
$^{1}_{1}$H	H-1	10010
$^{23}_{11}$Na	Na-23	110230
$^{56}_{26}$Fe	Fe-56	260560
$^{235}_{92}$U	U-235	922350
$^{239}_{94}$Pu	Pu-239	942390
$^{241}_{95}$Am (metastable)	Am-241m	952411
$^{255}_{100}$Fm	Fm-255	1002550
天然組成Fe	Fe-nat.	260000
$^{239}_{94}$Puのランプ化FP	Pu-239FP	8942390
$^{239}_{94}$Puのランプ化FP(ガス放出モデル)	Pu-239FPGR	7942390

付録B　MARBLEにおける反応種別の指定方法

本章ではMARBLEで利用する中性子核反応の反応種別を表す識別子（ID）についてまとめる。従来システムでは反応種別は整数のIDで表されており、解析コードによってその定義が異なるため、解析コードにあわせて反応種別IDを使い分ける必要があった。MARBLEでは、従来システム（SLAROM、CASUP、SLAROM-UF、SAGEP、ABLE、ACCEPT）で使われている反応種別IDを整理し、反応種別IDをシステム全体で統一している。

B.1　反応種別IDと別名

MARBLEでは反応種別はReactionTypeというクラスで管理している。ReactionTypeの識別にはプログラム内部ではidを使っているが、ReactionTypeの利用者はID以外にも名前（name）、短縮名（shortname）、別名（alias）を利用することができる。これらは以下のような意図で作成されているので、新しい反応種別を追加する際にはこれらのルールに従う必要がある。

ID 反応種別を識別するためのID。英語の小文字と数字、アンダースコアのみで表現される。

Name 反応の詳細名。一般的な英文で表現される。説明的な表示が必要な場合に用いることを想定している。

Shortname 反応の短縮名。英語の大文字と数字、ピリオドのみで表現される（8文字以内）。断面積の表や図を作成する場合に用いることを想定している。

Aliases 反応IDの別名。同じ反応種別でも異なる呼び名を持つ場合があるため、これらを表現するために利用する。（別名をたくさん作るとプログラムがわかりにくくなるので多用すべきではない。）

B.2　反応種別IDと定義

表B.2.1〜表B.2.3に従来システムの旧形式のPDSファイルに格納されているミクロ断面積の定義を示す[17]。SLAROMの生成する旧PDSファイルはオプションによる変更が特に多いので、別途、表B.2.2に示す。新形式のPDSファイルについてはSLAROM-UFのマニュアル[16]を参照することができる。。

MARBLEの反応種別IDは文字列を使っており、詳細名（Name）も定義しているので、ほとんどのものは多くのユーザにとって自明であると思われる。例えば、核分裂はfission、中性子捕獲はcapture、(n, 2n)反応はn2n、カレント重みの全断面積はcurrent_weight_totalのように定義

されている。しかしながら、輸送断面積のように定義が複数あるものもあるため、反応種別の ID の定義について以下にまとめる。

flux_weight_total 中性子束重みの全断面積（σ_t^0）。SLAROM が旧 PDS ファイルのマクロ断面積の 7 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=9 (total) に格納するデータ。

current_weight_total カレント重みの全断面積（σ_t^1）。SLAROM-UF が S0 メンバー、SB メンバーの mt=10 (total(current-weighted)) に格納するデータ。

fission 核分裂断面積（σ_f）。SLAROM、CASUP が旧 PDS ファイルのマクロ断面積の 8 番目、ミクロ断面積の 4 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=2 (fission) に格納するデータ。

nu_fission 生成断面積（$\nu\sigma_f$）。SLAROM、CASUP が旧 PDS ファイルのマクロ断面積、ミクロ断面積の 2 番目に格納するデータ。または、SLAROM-UF の新 PDS ファイルの S0 メンバー、SB メンバーの mt=1 (production) に格納するデータ。

nu 核分裂あたりの中性子発生数（ν）。新旧 PDS ファイルには含まれていないので、nu_fission と fission から算出。

$$\nu = \nu\sigma_f/\sigma_f$$

capture 中性子捕獲断面積（σ_c）。SLAROM が旧 PDS ファイルのミクロ断面積の 10 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=3 (capture) に格納するデータ。

fission_spectrum 核分裂スペクトル（χ）。SLAROM、CASUP が旧 PDS ファイルの#メンバーに格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=16(averaged fission spectrum) に格納するデータ。

mu_average 中性子平均散乱角余弦（$\bar{\mu}$）。SLAROM、CASUP が旧 PDS ファイルのミクロ断面積の 8 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバーの mt=12 (p1 component of elastic scattering(current-weighted))、mt=4 (elastic) に格納するデータから算出。

$$\bar{\mu} = \sigma_{el,1,g}^1/\sigma_{el,0,g}^0$$

n2n (n, 2n) 反応断面積。SLAROM、CASUP が旧 PDS ファイルのマクロ断面積の 11 番目、ミクロ断面積の 7 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=6 ((n,2n)) に格納するデータ。

n3n (n, 3n) 反応断面積。SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=7 ((n,3n)) に格納するデータ。

n4n (n, 4n) 反応断面積。SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=8 ((n,4n)) に格納するデータ。

ngamma (n, γ) 反応断面積。UFLIB の R メンバーの MT=102 に格納されているデータ。

nproton (n, p) 反応断面積。UFLIB の R メンバーの MT=103 に格納されているデータ。

ndeuteron (n, d) 反応断面積。UFLIB の R メンバーの MT=104 に格納されているデータ。

ntritium (n, t) 反応断面積。UFLIB の R メンバーの MT=105 に格納されているデータ。

nhelium (n, He) 反応断面積。UFLIB の R メンバーの MT=106 に格納されているデータ。

nalpha (n, α) 反応断面積。UFLIB の R メンバーの MT=107 に格納されているデータ。

absorption 吸収断面積（σ_a）。SLAROM、CASUP が旧 PDS ファイルのマクロ断面積、ミクロ断面積の 1 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバーの mt=2 (fission)、mt=3 (capture) に格納するデータから算出。

$$\sigma_a = \sigma_f + \sigma_c$$

average_diffusion_coefficient 等方拡散係数（$\bar{D}$）。CASUP、SLAROM が旧 PDS ファイルのマクロ断面積の 3 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバーの mt=13 (average diffusion coefficient) に格納するデータ。CASUP では、

$$\bar{D} = \frac{2D_{\parallel} + D_{\perp}}{3}$$

（スラブ体系）で計算される。SLAROM-UF でも上式が成り立つため、同じ名前で定義した。

perpendicular_diffusion_coefficient 垂直方向の非等方拡散係数（$D_{\perp}$）。CASUP、SLAROM が旧 PDS ファイルのマクロ断面積の 4 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバーの mt=14 (pependicular diffusion coefficient) に格納するデータ。

parallel_diffusion_coefficient 平行方向の非等方拡散係数（$D_{\parallel}$）。CASUP、SLAROM が旧 PDS ファイルのマクロ断面積の 5 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバーの mt=15 (parallel diffusion coefficient) に格納するデータ。

flux_weight_transport 中性子束重みの輸送断面積（σ^0_{tr}）。SLAROM が旧 PDS ファイルのマクロ断面積の 6 番目または 7 番目、ミクロ断面積の 3 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=9 (total)、mt=11 (p1 component of total scattering (current-weighted)) に格納されるデータから算出。

$$\sigma^0_{tr} = \sigma^0_{t,g} - \sigma^0_{s,1,g}$$

expanda_style_transport EXPANDA 方式の輸送断面積（$\sigma_{tr}^{\mathrm{expanda}}$）。SLAROM、CASUP では従来「カレント重みの輸送断面積」と呼ばれていた輸送断面積。 SLAROM-UF のマニュアルでは「EXPANDA 方式」と呼ばれている輸送断面積。MARBLE では両者を区別するために別の名前で定義した。SLAROM、CASUP が旧 PDS ファイルのマクロ断面積の 6 番または 7 番、ミクロ断面積の 9 番に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=18 (Transport(expanda)) に格納するデータ。

$$\sigma_{tr}^{\blacktriangledown} = \sigma_{t,g}^{1} - \bar{\mu}(\sigma_{el,0,g}^{0} + (\sigma_{t,g}^{1} - \sigma_{t,g}^{0}))$$

current_weight_transport カレント重みの輸送断面積（σ_{tr}^{1}）。SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=10 (total(current-weighted))、mt=11 (p1 component of total scattering(current-weighted)) に格納するデータから算出。SLAROM-UF ではこの断面積をカレント重みの輸送断面積と定義し、従来の SLAROM、CASUP の「カレント重みの輸送断面積」を EXPANDA 方式と定義しているため、MARBLE でも SLAROM-UF の定義にあわせた。

$$\sigma_{tr}^{1} = \sigma_{t,g}^{1} - \sigma_{s,1,g}^{1} = \sigma_{t,g}^{1} - \bar{\mu}\sigma_{s,0,g}^{0}$$

dphi_weight_transport $D\phi$ 重みで縮約した輸送断面積。JOINT が旧 PDS ファイルのミクロ断面積の 11 番目に格納する縮約断面積。旧 PDS ファイルとの互換性のために定義した。（注 : CrossSection クラスの縮約では、縮約の重みに応じて反応名を変更しないので、旧 PDS ファイルのデータを読み込みたいとき以外には使う必要はない。）

total_scattering 全散乱断面積（σ_s）。SLAROM が旧 PDS ファイルのマクロ断面積の 9 番目、ミクロ断面積の 5 番目に格納するデータ。CASUP が旧 PDS ファイルのミクロ断面積の 5 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバーの mt=9 (total)、mt=2 (fission)、mt=3 (capture) に格納するデータから算出。

$$\sigma_s = \sigma_t - (\sigma_f + \sigma_c) = \sigma_t - \sigma_a$$

total_scattering_matrix 全散乱マトリックス（$\sigma_{s,g\to g'}$）。SLAROM が旧 PDS ファイルのマクロ断面積、ミクロ断面積の 12 番目に格納するデータ。CASUP が旧 PDS ファイルのミクロ断面積の 12 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバーの sigs、SM メンバーから算出。

$$\sigma_{s,g\to g'} = \sigma_{el,g\to g'} + \sigma_{in,g\to g'} + \sigma_{n2n,g\to g'}$$

total_scattering_for_flux_weight_transport 中性子束重みの輸送断面積用に自群散乱を補正した後の全散乱断面積。SLAROM がマクロ断面積の 9 番に格納するデータ。本来、中性子束重みの輸送断面積と吸収断面積が用意されていれば計算できるので定義する必要はない。

しかしながら、旧 PDS ファイルに格納されるデータから補正前の値を計算することができないため、旧 PDS ファイルとの互換性を保持し、SLAROM-UF の全散乱断面積との差を明確にするために定義した。

$$\Sigma_s^{\mathrm{flux}} = \Sigma_{tr}^0 - \Sigma_a$$

total_scattering_matrix_for_flux_weight_transport 中性子束重みの輸送断面積用に自群散乱を補正した後の全散乱マトリックス。SLAROM がマクロ断面積の 12 番に格納するデータ。

$$\Sigma_{s,g\to g}^{\mathrm{flux}} = \Sigma_{s,g\to g} - (\Sigma_t - \Sigma_{tr}^0)$$

total_scattering_for_expanda_style_transport EXPANDA 方式の輸送断面積用に自群散乱断面積を補正した後の全散乱断面積。SLAROM、CASUP がマクロ断面積の 9 番に格納するデータ。本来、中性子束重みの輸送断面積と吸収断面積が用意されていれば計算できるので定義する必要はない。しかしながら、旧 PDS ファイルに格納されるデータから補正前の値を計算することができないため、旧 PDS ファイルとの互換性を保持し、SLAROM-UF の全散乱断面積との差を明確にするために定義した。

$$\Sigma_s^{\mathrm{expanda}} = \Sigma_{tr}^{\blacktriangledown} - \Sigma_a$$

total_scattering_matrix_for_expanda_style_transport EXPANDA 方式の輸送断面積用に自群散乱を補正した後の全散乱マトリックス。SLAROM、CASUP がマクロ断面積の 12 番に格納するデータ。

$$\Sigma_{s,g\to g}^{\mathrm{expanda}} = \Sigma_{s,g\to g} - (\Sigma_t - \Sigma_{tr}^{\blacktriangledown})$$

current_weight_p1_total_scattering 全散乱断面積の P1 成分（$\sigma_{s,1}^1$）。SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=11 (p1 component of total scattering(current-weighted)) に格納するデータ。

elastic_scattering 弾性散乱断面積（σ_{el}）。SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=4 (elastic) に格納するデータ。

current_weight_p1_elastic_scattering 弾性散乱断面積の P1 成分（$\sigma_{el,1}^1$）。SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=12 (p1 component of elastic scattering(current-weighted)) に格納するデータ。

inelastic_scattering 非弾性散乱断面積（σ_{in}）。SLAROM、CASUP が旧 PDS ファイルのマクロ断面積の 10 番目、ミクロ断面積の 6 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの S0 メンバー、SB メンバーの mt=5 (inelastic) に格納するデータ。

elastic_scattering_matrix 弾性散乱マトリックス（$\sigma_{el,g\to g'}$）。感度解析用 SLAROM（SLAROM-SNS）が旧 PDS ファイルのマクロ断面積、ミクロ断面積の 13 番目に格納するデータ。または、SLAROM-UF が新 PDS ファイルの SM メンバーの itype=1 に格納するデータ。

inelastic_scattering_matrix 非弾性散乱マトリックス（$\sigma_{in,g\to g'}$）。感度解析用SLAROM（SLAROM-SNS）が旧PDSファイルのマクロ断面積、ミクロ断面積の14番目に格納するデータ。または、SLAROM-UFが新PDSファイルのSMメンバーのitype=2に格納するデータ。

n2n_matrix (n, 2n) マトリックス（$\sigma_{n2n,g\to g'}$）。感度解析用SLAROM（SLAROM-SNS）が旧PDSファイルのマクロ断面積、ミクロ断面積の15番目に格納するデータ。または、SLAROM-UFが新PDSファイルのSMメンバーのitype=3に格納するデータ。

elastic_removal 散乱除去断面積（σ_r）。SLAROM-UFが新PDSファイルのSBメンバーのmt=13 (elastic removal) に格納するデータ。

total_n2n 全(n, 2n)反応断面積。UFLIBのMT=-16に格納されているデータ。

total_n3n 全(n, 3n)反応断面積。UFLIBのMT=17に格納されているデータ。

delayed_nu 核分裂あたりの遅発中性子発生数（ν_d）。UFLIBのMT=455に格納されているデータ。

ffactor_gradient_capture 中性子捕獲断面積の自己遮蔽因子の勾配（温度依存性）（$\frac{1}{f}\frac{\mathrm{d}f_c}{\mathrm{d}T}$）。SAGEP、ABLE、ACCEPTが反応ID=21として扱うデータ。

ffactor_gradient_fission 核分裂断面積の自己遮蔽因子の勾配（温度依存性）（$\frac{1}{f}\frac{\mathrm{d}f_f}{\mathrm{d}T}$）。SAGEP、ABLE、ACCEPTが反応ID=22として扱うデータ。

ffactor_gradient_elastic_scattering 弾性散乱断面積の自己遮蔽因子の勾配（温度依存性）（$\frac{1}{f}\frac{\mathrm{d}f_{el}}{\mathrm{d}T}$）。SAGEP、ABLE、ACCEPTが反応ID=25として扱うデータ。

ffactor_gradient_inelastic_scattering 非弾性散乱断面積の自己遮蔽因子の勾配（温度依存性）（$\frac{1}{f}\frac{\mathrm{d}f_{in}}{\mathrm{d}T}$）。SAGEP、ABLE、ACCEPTが反応ID=26として扱うデータ。

inelastic_scattering_level_1〜34 励起レベル別の非弾性散乱断面積（Level=1〜34）。SAGEP、ABLE、ACCEPTが反応ID=51〜84として扱うデータ。

表 B.2.1　旧形式の PDS ファイル内のマクロ断面積の定義

コード名	モデル	PDS格納位置識別IDとPDSファイル内断面積内容														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLAROM	均質	Σ_a	$\nu\Sigma_f$	D_{av}^*	D_{av}^*	D_{av}^*	Σ_{tr}	Σ_t^*	Σ_f	Σ_s^*	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}^*$	$\Sigma_{s,1,g\to j}$	$\Sigma_{s,2,g\to j}$	$\Sigma_{s,3,g\to j}$
SLAROM	非均質	Σ_a	$\nu\Sigma_f$	D_{av}^*	$D_{\perp}^*$	$D_{//}^*$	Σ_{tr}	Σ_t^*	0.0	Σ_s^*	0.0	0.0	$\Sigma_{s,g\to j}^*$	$\Sigma_{s,1,g\to j}$	$\Sigma_{s,2,g\to j}$	$\Sigma_{s,3,g\to j}$
感度係数用 SLAROM	均質	Σ_a	$\nu\Sigma_f$	D_{av}^*	D_{av}^*	D_{av}^*	Σ_{tr}	Σ_t^*	Σ_f	Σ_s^*	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}^*$	$\Sigma_{d,g\to j}$	$\Sigma_{ind,g\to j}$	$\Sigma_{n2n,g\to j}$
CASUP	非均質	Σ_a	$\nu\Sigma_f$	$D^{\blacktriangledown}_{av}$	$D^{\blacktriangledown}_{\perp}$	$D^{\blacktriangledown}_{//}$	$\Sigma^{\blacktriangledown}_{tr}$	Σ_t	Σ_f	Σ_s	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}$	–	–	–
JOINT	縮約定数	一般に、D（拡散係数が中性子束重み）のとき ϕ 重みで縮約、$D^{\blacktriangledown}$（拡散係数がカレント重み）のとき $\phi^{\blacktriangledown}$ 重みで縮約する。														

$\Sigma^{\blacktriangledown}_{tr}$　　$\Sigma^{\blacktriangledown}_{tr}-\Sigma_a$　　$\Sigma_{s,g\to g}-(\Sigma_t-\Sigma^{\blacktriangledown}_{tr})$

表 B.2.2　旧形式の PDS ファイル内のマクロ断面積の定義（SLAROM、均質体系）

ICASE	IBSW	PDS格納位置											
		1	2	3	4	5	6	7	8	9	10	11	12
-1	-1	Σ_a	$\nu\Sigma_f$	$D^{\blacktriangledown}_{av}$	←	←	Σ_{tr}	Σ_t	Σ_f	Σ_s	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}$, $\Sigma_s^{g\to g}$
	0	Σ_a	$\nu\Sigma_f$	$D^{\blacktriangledown}_{av}$	←	←	Σ_{tr}	$\Sigma^{\blacktriangledown}_{tr}$	Σ_f	$\Sigma^{\blacktriangledown}_{tr}-\Sigma_a$	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}$, $\Sigma_s^{g\to g}-(\Sigma_t-\Sigma^{\blacktriangledown}_{tr})$
	1	Σ_a	$\nu\Sigma_f$	$D^{\blacktriangledown}_{av}$	←	←	Σ_{tr}	$\Sigma^{\blacktriangledown}_{tr}$	Σ_f	$\Sigma^{\blacktriangledown}_{tr}-\Sigma_a$	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}$, $\Sigma_s^{g\to g}-(\Sigma_t-\Sigma^{\blacktriangledown}_{tr})$
0	-1	Σ_a	$\nu\Sigma_f$	$D^{\blacktriangledown}_{av}$	←	←	Σ_{tr}	Σ_t	Σ_f	Σ_s	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}$, $\Sigma_s^{g\to g}$
	0	Σ_a	$\nu\Sigma_f$	D_{av}	←	←	Σ_{tr}	Σ_{tr}	Σ_f	$\Sigma_{tr}-\Sigma_a$	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}$, $\Sigma_s^{g\to g}-(\Sigma_t-\Sigma_{tr})$
	1	Σ_a	$\nu\Sigma_f$	D_{av}	←	←	Σ_{tr}	Σ_{tr}	Σ_f	$\Sigma_{tr}-\Sigma_a$	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}$, $\Sigma_s^{g\to g}-(\Sigma_t-\Sigma_{tr})$
1	-1	Σ_a	$\nu\Sigma_f$	$D^{\blacktriangledown}_{av}$	←	←	Σ_{tr}	Σ_t	Σ_f	Σ_s	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}$, $\Sigma_s^{g\to g}$
	0	Σ_a	$\nu\Sigma_f$	D_{av}	←	←	Σ_{tr}	Σ_t	Σ_f	Σ_s	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}$, $\Sigma_s^{g\to g}$
	1	Σ_a	$\nu\Sigma_f$	D_{av}	←	←	Σ_{tr}	Σ_t	Σ_f	Σ_s	Σ_{in}	Σ_{n2n}	$\Sigma_{s,g\to j}$, $\Sigma_s^{g\to g}$

表 B.2.3　旧形式の PDS ファイル内のミクロ断面積の定義

コード名	モデル	PDS格納位置識別IDとPDSファイル内断面積内容														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLAROM	均質	σ_a	$\nu\sigma_f$	σ_{tr}	σ_f	σ_s	σ_{in}	σ_{n2n}	μ	$\sigma^{\blacktriangledown}_{tr}$	σ_c	0.0	$\sigma_{s,g\to j}$	$\sigma_{s,1,g\to j}$	$\sigma_{s,2,g\to j}$	$\sigma_{s,3,g\to j}$
SLAROM	非均質	σ_a	$\nu\sigma_f$	σ_{tr}	σ_f	σ_s	σ_{in}	σ_{n2n}	μ	$\sigma^{\blacktriangledown}_{tr}$	σ_c	0.0	$\sigma_{s,g\to j}$	$\sigma_{s,1,g\to j}$	$\sigma_{s,2,g\to j}$	$\sigma_{s,3,g\to j}$
感度係数用 SLAROM	均質	σ_a	$\nu\sigma_f$	σ_{tr}	σ_f	σ_s	σ_{in}	σ_{n2n}	μ	$\sigma^{\blacktriangledown}_{tr}$	σ_c	0.0	$\sigma_{s,g\to j}$	$\sigma_{el,g\to j}$	$\sigma_{ind,g\to j}$	$\sigma_{n2n,g\to j}$
CASUP	非均質	σ_a	$\nu\sigma_f$	$\sigma^{\blacktriangledown}_{tr}$	σ_f	σ_s	σ_{in}	σ_{n2n}	μ	$\sigma^{\blacktriangledown}_{tr}$	0.0	0.0	$\sigma_{s,g\to j}$	—	—	—
JOINT	縮約定数	ϕ重み	ϕ重み	ϕ重み	ϕ重み	ϕ重み	ϕ重み	ϕ重み	ϕ重み	ϕ重み	ϕ重み	σ_{tr}をDϕ重み*	ϕ重み	ϕ重み	ϕ重み	ϕ重み
		$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み	$\sigma^{\blacktriangledown}_{tr}$をD$^{\blacktriangledown}\phi^{\blacktriangledown}$重み*	$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み	$\phi^{\blacktriangledown}$重み

付録C　解析コードのカプセル化

MARBLEでは既存の解析コードのカプセル化という概念がシステム構成上、重要な役割を果たしている。従来システムに含まれている解析コードのほとんどがカプセル化されており、MARBLEではカプセル化したものを利用する。カプセル化はオブジェクト指向における基本的な概念のひとつであるが、ここでいう「カプセル化」は既存の外部コードの入力データ、出力データ、実行ファイルをPython言語で「すべて」制御できるようにするための機能を指す。

このカプセル化により既存のコードに手を加えることなく、MARBLE上で既存のコードの入力データの作成、計算コードの実行、出力結果の抽出を制御できる。また、データのひとつひとつをプログラム上で簡単に設定できるため、自動化するのが簡単になるという利点がある。

今後、解析手法やモデルを高度化していく場合には、オブジェクト指向技術の利用できるPythonやC++を用いて新規にソルバーを開発するのが理想的であるが、従来システムに含まれていない他の公開コードをMARBLE上で利用する可能性がある。この場合には、MARBLEのカプセル化方法に準拠してカプセル化を行う必要がある。ここではMARBLEのカプセル化の基本方針についてまとめる。

C.1　背景

従来コードの入出力ファイルは逐次アクセスが前提となっており、ランダムアクセスが難しいことが多い。また、従来コードの入出力ファイルが自己記述的でなく、オブジェクト指向的な取り扱いが難しいことが多い。PSAGEPの開発においても従来コードの入力ファイルを生成するInputGeneratorの開発に工数を要した。このため、開発者がこの作業を繰り返し行わなくても済むように、従来コードの入出力データに対して、オブジェクト指向プログラミングと親和性の高いランダムアクセスをサポートする階層が必要である。

従来システムの既存コードの入出力ファイルのファイル形式を隠蔽し、データ項目へのランダムアクセス機能を提供するものとしてMARBLEのカプセル化の方針を決定した。

C.2　基本設計

従来システムに含まれている解析コードは基本的に入出力にファイルを使い、制御にシェルスクリプトを使う。このことを考慮して、従来コードのカプセル化の準備として、従来コードの入出力ファイルとシェルスクリプトに相当する機能を持つ階層を設ける。この階層では、従来コードの入出力ファイルのデータ構造をそのまま使い、シェルスクリプトの動作をそのまま模擬する。MARBLEの物理データや物理モデルに基づいた抽象化は行わない。この階層を「低レベルカプセル化層」と呼び、高度な抽象化は「高レベルカプセル化層」と呼ぶ別の階層を新たに作成して

対応する。

従来コードの入出力ファイルを表現するクラスを ConventionalCodeFile、従来コードのシェルスクリプトを表現するクラスを ConventionalCodeRunner と呼ぶことにする。

C.3 詳細設計と実装方法

C.3.1 インターフェース

入出力ファイルのカプセル化のインターフェースを決める ConventionalCodeFile クラスは以下のようなインターフェースを持たせることにする。

```
================================
 ConventionalCodeFile
--------------------------------
 - data
--------------------------------
 + __init__(self, path)
 + __init__(self, fin)
 + __str__(self)
 + read_file(self, fin)
 + write_file(self, fout)
 + handler(self)
 + strio(self)

 # _initialize(self, path, fin)
 # _read_file(self, fin)
 # _write_file(self, fout)

 + set
 + get
================================
```

インスタンス変数 `data` は、Python のリストとディクショナリで構成される。従来コードの入出力ファイルに含まれるすべてのデータをリストとディクショナリの混合体として表現するように実装する。従来コードのマニュアルに変数名が明示されている場合は、変数名をキー（すべて小文字にする）としたディクショナリとし、データの意味として順番が必要な場合はリストで表現する。

`read_file()`、`write_file()` は Python のファイルオブジェクト（file や StringIO 等）を受け取りそのファイルオブジェクトに、従来コードのフォーマットでデータを入出力する。この２つのメソッドを使うことで、自由にファイルからデータを読み出したり、ファイルに書き出したりすることができるようになっている。

なお、このクラスでは汎用性を高めるために、ファイルパスや文字列データを直接処理せずに、常に Python ファイルオブジェクトに対して操作するように実装しなければならない。ファイルパスであれば open 文で file オブジェクトに変換してから、文字列データであれば StringIO オブジェクトに変換してから処理を行う。

ファイルパスを受け取るコンストラクタ `__init__()` と、文字列データを返す `__str__()` メソッドは、カプセル化したファイルを対話的に使い易く（統一的な方法で使用）するために用意され

ているものであるので、サブクラスで安易にオーバーライドするべきではない。

C.3.2 実装方法

既存コードの入出力ファイルをカプセル化する際には、ConventionalCodeFile クラスを継承し、「ConventionalCode」の部分を既存コードの名前にしたサブクラスを作成する。例えば、CITATION の入力機番 5 のファイルの場合、クラス名は「CitationFort5」となる。サブクラスでは「_（ひとつのアンダースコア）」で始まる以下の protected メソッドと、解析コードの入出力データに付けられている変数名に対応して「set_変数名 ()」という名前のデータ設定用のメソッドと「変数名 ()」という名前のデータ取得用のメソッド（両方をまとめてアクセッサメソッドと呼ぶ）を実装することになる。例えば、CITATION のエネルギー群数を表す入力データ（変数名 KMAX）に対するアクセッサメソッドとして、「set_kmax()」と「kmax()」というメソッドを実装する。

サブクラスで実装する各 protected メソッドの役割は以下のとおりである。

① `_initialize()`: ファイルによってはデータの順番を保持した方が良い場合があるため、self.data を list にする等、インスタンス生成時に別の処理をしたい場合にこのメソッドに記述する。

② `_read_file()`: `read_file()` の既存コードの入出力ファイル毎の実装を記述する。`read_file()` では、各コードの入出力ファイルで共通化できる fin の前処理・後処理等を記述する。

③ `_write_file()`: `_read_file()` と同様、共通化できない `write_file()` の実装部分を記述する。

前述のようにサブクラスではこれらの protected メソッドに加えて、アクセッサメソッドを実装する必要があるが、すべてのデータはインスタンス変数 data に納められているので、アクセッサメソッドの実装はインスタンス変数 data へのデータの設定と取得だけの非常に簡単なものになるはずである。

アクセッサメソッドの実装は単純な作業となるが、変数毎に定義しておくことで以下のような利点が得られる。

- 解析コードで使われている変数名と一対一対応するので、カプセル化作業に伴って曖昧さが混入する確率が少ない。また、解析コードのマニュアルを参照すればデータの意味が明らかになる。
- 変数名に対応した各メソッドに docstring を埋め込めるので、変数名の説明を詳細に記述すれば解析コードのオンラインマニュアルとして利用できる。
- カプセル化を利用する際、変数名を間違えて入力した場合にエラーになるため、ユーザが誤った入力をして誤動作させる確率が下がる。

解析コードの入出力ファイルのカプセル化はインターフェースを統一することで再利用性を高めている。低レベルカプセル化層は一度作成すれば再利用性が非常に高いので、一貫性を守るようにして丁寧に作成する必要がある。

付録D　燃焼計算ソルバーBURNUP

従来システムには単独で利用できる燃焼計算コードがなかったため、MARBLE の一部として燃焼計算ソルバー（BURNUP ソルバー）を新規に開発した。高速炉の炉心燃焼解析ではランプ化 FP を使った簡易の燃焼チェーンを用いることが多いため、行列指数法でも十分な計算速度が得られる。実際、従来システムに組み込まれている燃焼計算ソルバーは行列指数法に基づくものである。

一方で、より詳細な燃焼解析を行うためには FP 核種を個別に扱った詳細な燃焼チェーンが必要となる。しかしながら、短半減期の FP を含むような詳細な燃焼チェーンを使うと、行列指数法では計算速度が遅くなることが知られている。このため、従来システムの燃焼計算ソルバーと同等の行列指数法と、詳細な燃焼チェーンでも高速な計算が可能なクリロフ部分空間法に基づく燃焼計算ソルバーを開発した。

D.1　燃焼方程式の解法

一般に燃焼計算は次に示す燃焼方程式を解くことで行われる。

$$\frac{\mathrm{d}\vec{n}(t)}{\mathrm{d}t} = \mathbf{A}\vec{n}(t) \tag{D.1}$$

ここで、$\vec{n}$ は時刻 t における組成ベクトルを、$\mathbf{A}$ は遷移（燃焼）行列を表す。この方程式を解く方法のひとつとして、ORIGEN 等で利用されている行列指数法がある。行列指数法ではこの式を次のように変形して解く。

$$\vec{n}(t) = \exp(\mathbf{A}t) \cdot \vec{v} \tag{D.2}$$

ただし、ここで $\vec{v} = \vec{n}(0)$ である。上式において、$\mathbf{I}$ を単位行列とすると exp の項は以下のように Taylor 展開することができる。

$$\exp(\mathbf{A}t) = \mathbf{I} + \mathbf{A}t + \frac{1}{2}(\mathbf{A}t)^2 + \cdots = \sum_{m=0}^{\infty} \frac{(\mathbf{A}t)^m}{m!} \tag{D.3}$$

すなわち、以下の式を使って数値計算することで燃焼方程式を解くことができる。

$$\vec{n}(t) = \vec{v} + \frac{(\mathbf{A}t)}{1!}\vec{v} + \frac{(\mathbf{A}t)^2}{2!}\vec{v} + \cdots + \frac{(\mathbf{A}t)^{m-1}}{(m-1)!}\vec{v} \tag{D.4}$$

しかしながら、この方法で数値計算上の精度を確保するためには、Lapidus と Luus によって定義された行列 $\mathbf{A}t$ のノルムが一定の条件を満たすように時間ステップを短くする必要がある。この方法を使えば行列指数の計算精度を確保できることが示されているが、取り扱う同位体数が非常に多い場合には、単純に時間ステップを短くしていくだけでは計算時間の点で問題となる。こ

のため、ORIGEN2 コードでは、指数行列の計算の誤差が一定になるように遷移行列から同位体を取り除き、別途解析的に取り扱う等の工夫がなされている [65]。

一方、クリロフ部分空間法を利用する方法 [64] では、(D.4) 式に対して以下の式で定義される m 次のクリロフ部分空間 K_m を考える。

$$K_m(\mathbf{A}t, \vec{n}(t))) = \mathrm{Span}\{\vec{v}, (\mathbf{A}t)\vec{v}, (\mathbf{A}t)^2\vec{v}, \cdots, (\mathbf{A}t)^{m-1}\vec{v}\} \tag{D.5}$$

このとき、燃焼方程式の解は近似的に以下の式で求められる。

$$\vec{n}(t) = \beta \mathbf{V}_{m+1} \exp(t\mathbf{H}_{m+1})\vec{e}_1 \tag{D.6}$$

ここで、

$$\begin{aligned} \vec{e}_1 &= [1, 0, \cdots, 0] \\ \beta &= ||\vec{v}||_2 \\ \mathrm{V}_{m-1} &= [\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_{m+1}] \\ \mathrm{H}_{m-1} &= [h_{ij}] \end{aligned}$$

である。$\vec{e}_1$ は 1 番目の要素が 1 の単位ベクトル、β は L2 ノルムを表す。V_{m-1}、H_{m-1} は、それぞれ、アーノルディ法（Arnoldi process）で得られる正規直交基底（orthonormal basis）、上ヘッセンベルグ行列（upper Hessenberg matrix）である。上ヘッセンベルグ行列は密行列となるため、$\exp(t\mathbf{H}_{m+1})$ は通常の行列指数計算手法（パーデ近似等）で高速に計算することが可能である。このようにクリロフ部分空間法では、粗行列の大きな問題を密行列の小さな問題に変換することで大幅な高速化を図ることができる。

D.2 実装

燃焼計算ソルバー BURNUP は MARBLE の Burnup パッケージで実装されており、中核となるモジュールとして Burnup モジュールと BurnupUtils モジュールを作成した。行列指数の計算には Python の数値計算ライブラリを使っており、実際の処理としては、遷移行列の作成、遷移行列のノルムを使った時間ステップの決定、計算オプションの処理等を行っている。

D.2.1 行列指数法ソルバーの実装

行列指数法に基づく燃焼計算では、行列指数の計算部分が主な計算負荷となるので行列指数の計算部分の計算効率を無視することはできない。Python では行列計算を効率的に行うことができるライブラリ（Numpy[39]、Scipy[74]）が開発されており、指数行列の計算にはこれらのライブラリを利用した。このため、今回の燃焼計算ソルバーの開発では MARBLE 独自の C++ と Python による二階層システムを使う必要はなく Python のみで実装することができた[1]。

[1] 実際には、Numpy、Scipy の内部で C や Fortran が使われているので内部的には二階層化されている。

Scipy ライブラリの行列指数関数としては、パーデ近似を用いるもの、固有値分解を用いるもの、テイラー展開した式を直接計算するものが用意されており、Burnup クラスでは、オプションでこれらを使い分けられるようにした。

なお、デフォルトではテイラー展開に基づいた関数を使うようになっており、必要な計算精度を確保できるように時間ステップを自動的に設定するようにテイラー展開の次数を決定する。BURNUP ソルバーでは、ORIGEN2 コードで採用されている条件と同様に行列指数の計算精度を 0.1%以下とするようにしており、テイラー展開の次数は 54 に制限している。

D.2.2 クリロフ部分空間法ソルバーの実装

クリロフ部分空間法による燃焼計算は EXPOKIT[73] の expv 関数を使うことで実現できる。EXPOKIT には MATLAB による実装と FORTRAN による実装が含まれており、Python ベースの MARBLE で利用できるように、(1)Python を使ってアルゴリズムを実装した Python 版 expv 関数と、(2)EXPOKIT の FORTRAN ソースコードを Python から呼び出せるようにした Fortran 版 expv 関数を作成した。今回作成したクリロフ部分空間法に基づく BURNUP ソルバーでは両者を使い分けることができるようになっており、ここでは前者を Python 版、後者を FORTRAN 版と呼ぶことにする。どちらも同じアルゴリズムであるため同じ計算結果となる。Python 版は FORTRAN 版に比べてプログラムが簡潔で読みやすくなる利点がある一方で実行速度は遅くなる。

FORTRAN 版を使うためには、Python の拡張モジュール（`_fexpv.so`）をコンパイルしておく必要がある。コンパイルするには、Burnup パッケージのディレクトリで make コマンドを実行する。BURNUP ソルバーは FORTRAN 版の Python 拡張モジュールが存在すれば FORTRAN 版 expv 関数（fexpv.py）、存在しなければ Python 版 expv 関数（expv.py）を自動的に呼び出す。すなわち、Python 版を使いたい場合は`_fexpv.so` を消去しておけば良い。

D.3 検証計算

D.3.1 行列指数法ソルバーの検証

BURNUP ソルバーの行列指数法による燃焼計算機能の検証として、以下のような高速炉燃料組成を使った計算ケースを用いて ORIGEN2 コードの計算結果との比較を行った。表 D.3.1 に検証計算の結果を示す。

- 高速炉燃料組成
  - Pu/HM=18.5wt%
  - Pu 同位体比: ^{238}Pu/^{239}Pu/^{240}Pu/^{241}Pu/^{242}Pu = 3/53/27/10/7(wt%)
- 計算条件:
  - 中性子束一定（3.73×10^{15} n/cm^2/sec）

– 燃焼期間 375 日

MARBLE 燃焼計算ソルバーでは高速炉の燃焼解析で標準的に用いられている燃焼チェーン（重核種のみを考慮し、核分裂生成物はランプ化 FP で取り扱う）を用いた。なお、ORIGEN2 の燃焼チェーンは非常に詳細であるため、MARBLE ソルバーの燃焼チェーンでは陽に取り扱われていない核種の生成まで考慮されてしまう。このため、両者を同等に比較できるように ORIGEN2 の結果を補正している。表から分かるように主要な核種では 0.1%以下の差で ORIGEN2 の結果と一致しており、MARBLE 燃焼計算ソルバーが十分な計算精度を持っていることが確認できた。

D.3.2 クリロフ部分空間法ソルバーの検証

BURNUP ソルバーのクリロフ部分空間法による燃焼計算機能の検証として、SRAC[75] コードの計算結果との比較を行った。検証計算には以下のような計算条件を適用した。なお、両者の結果を直接比較できるようにするため、SRAC の燃焼計算で利用される 1 群断面積を出力し、BURNUP ソルバーの入力として利用した。

- 燃料組成：60 万 kWe 級酸化物燃料炉心の内側炉心の燃料組成
- 燃焼チェーン：SRAC の詳細チェーン（th2cm6fp193bp6F）
- 燃焼期間：300 日燃焼後、600 日冷却（燃焼期間中の中性子束一定）
- 中性子束：3.6×10^{15}n/cm^2/sec
- 断面積：JENDL-3.2
- MARBLE/BURNUP の設定：`set_undefined_depletion(True)` を設定
- SRAC の設定：燃焼ステップ中の中性子束を一定にするオプション（IBC3=3）を設定

燃焼・冷却後の原子数密度の比較を表 D.3.2〜表 D.3.7 に示す。これらの表から、クリロフ部分空間法を導入する前の結果と同様に ^{135}Ba、^{155}Gd については約 2%の差が見られるが、大半の核種は 0.1% 以内の差で一致していることが分かる。冷却後の原子数密度が小さい核種では相対差が大きくなっているものがあるが、崩壊でほとんどなくなっている核種であるので実用上問題ないと考えられる。

表 D.3.1　検証計算（行列指数法）

	初期重量	燃焼後の重量 [g]		差 [%]
	[g]	ORIGEN2(O)	MARBLE(M)	(M-O)/O
U-234	0.000e+00	2.425e+03	2.467e+03	1.7
U-235	1.533e+05	1.150e+05	1.150e+05	-0.0
U-236	0.000e+00	8.608e+03	8.529e+03	-0.9
U-238	5.096e+07	4.895e+07	4.895e+07	-0.0
Np-237	0.000e+00	1.057e+04	1.061e+04	0.4
Np-239	0.000e+00	1.503e+04	1.497e+04	-0.4
Pu-238	3.550e+05	2.922e+05	2.924e+05	0.1
Pu-239	6.153e+06	6.152e+06	6.147e+06	-0.1
Pu-240	3.195e+06	3.195e+06	3.192e+06	-0.1
Pu-241	1.124e+06	9.105e+05	9.097e+05	-0.1
Pu-242	8.283e+05	8.162e+05	8.160e+05	-0.0
Am-241	1.775e+05	1.834e+05	1.834e+05	-0.0
Am-242m	0.000e+00	5.674e+03	5.650e+03	-0.4
Am-243	0.000e+00	3.754e+04	3.741e+04	-0.4
Cm-242	0.000e+00	1.178e+04	1.176e+04	-0.2
Cm-243	0.000e+00	2.416e+02	2.402e+02	-0.6
Cm-244	0.000e+00	2.206e+03	2.191e+03	-0.7
Cm-245	0.000e+00	6.694e+01	6.618e+01	-1.1
Cm-246	0.000e+00	6.186e-01	6.091e-01	-1.5
Cm-247	0.000e+00	3.255e-03	3.192e-03	-1.9

表 D.3.2　検証計算結果（クリロフ部分空間法）: 重核種

核種	燃焼前 [10^{24}/cm^3]	燃焼直後[10^{24}/cm^3]		M/S-1	冷却後[10^{24}/cm^3]		M/S-1
		SRAC	MARBLE		SRAC	MARBLE	
Th-232	---	3.823E-14	3.820E-14	-0.08%	8.652E-14	8.645E-14	-0.08%
Pa-231	---	2.023E-18	2.021E-18	-0.07%	2.022E-18	2.021E-18	-0.07%
Pa-233	---	1.546E-16	1.545E-16	-0.08%	6.923E-20	6.918E-20	-0.08%
U-232	---	1.735E-13	1.735E-13	0.00%	6.632E-13	6.629E-13	-0.05%
U-233	---	4.032E-12	4.029E-12	-0.07%	4.032E-12	4.029E-12	-0.07%
U-234	---	2.131E-07	2.130E-07	-0.07%	4.170E-07	4.167E-07	-0.07%
U-235	1.721E-05	1.354E-05	1.354E-05	-0.01%	1.356E-05	1.356E-05	-0.01%
U-236	1.190E-06	1.966E-06	1.966E-06	-0.01%	2.000E-06	2.000E-06	-0.01%
U-237	---	2.618E-08	2.617E-08	-0.01%	3.868E-12	3.865E-12	-0.09%
U-238	7.161E-03	6.932E-03	6.931E-03	-0.01%	6.932E-03	6.931E-03	-0.01%
Np-236	---	2.001E-12	2.000E-12	-0.01%	2.001E-12	2.000E-12	-0.01%
Np-237	---	7.144E-07	7.143E-07	-0.01%	7.564E-07	7.563E-07	-0.01%
Np-239	---	2.238E-06	2.238E-06	-0.01%	4.148E-12	4.145E-12	-0.08%
Pu-236	---	2.729E-12	2.729E-12	0.00%	2.236E-12	2.236E-12	0.01%
Pu-238	3.700E-05	3.127E-05	3.127E-05	0.00%	3.144E-05	3.144E-05	0.00%
Pu-239	8.296E-04	8.407E-04	8.407E-04	0.00%	8.429E-04	8.429E-04	0.00%
Pu-240	3.909E-04	3.995E-04	3.995E-04	0.00%	3.995E-04	3.995E-04	0.00%
Pu-241	1.465E-04	1.248E-04	1.247E-04	-0.02%	1.199E-04	1.199E-04	-0.01%
Pu-242	1.051E-04	1.042E-04	1.042E-04	-0.01%	1.042E-04	1.042E-04	-0.01%
Am-241	5.746E-06	9.503E-06	9.500E-06	-0.04%	1.434E-05	1.434E-05	-0.05%
Am-242	---	4.367E-09	4.365E-09	-0.04%	3.165E-12	3.162E-12	-0.09%
Am-242m	---	2.462E-07	2.461E-07	-0.03%	2.452E-07	2.452E-07	-0.03%
Am-243	---	4.736E-06	4.736E-06	0.00%	4.736E-06	4.735E-06	0.00%
Cm-242	---	5.209E-07	5.207E-07	-0.03%	1.467E-07	1.466E-07	-0.03%
Cm-243	---	1.330E-08	1.330E-08	-0.02%	1.304E-08	1.304E-08	-0.02%
Cm-244	---	3.698E-07	3.698E-07	0.00%	3.584E-07	3.584E-07	0.00%
Cm-245	---	8.439E-09	8.439E-09	0.00%	8.438E-09	8.438E-09	0.00%
Cm-246	---	7.413E-11	7.409E-11	-0.05%	7.412E-11	7.408E-11	-0.05%

表 D.3.3　検証計算結果（クリロフ部分空間法）: 核分裂生成物 (1/5)

核種	燃焼前 [10^{24}/cm^3]	燃焼直後[10^{24}/cm^3]		M/S-1	冷却後[10^{24}/cm^3]		M/S-1
		SRAC	MARBLE		SRAC	MARBLE	
Ge-73	---	1.305E-09	1.305E-09	-0.02%	1.305E-09	1.305E-09	-0.02%
Ge-74	---	3.142E-09	3.141E-09	-0.02%	3.142E-09	3.141E-09	-0.02%
Ge-76	---	9.649E-09	9.648E-09	-0.01%	9.649E-09	9.648E-09	-0.01%
As-75	---	4.417E-09	4.417E-09	-0.02%	4.417E-09	4.417E-09	-0.02%
Se-76	---	9.686E-11	9.685E-11	-0.02%	9.686E-11	9.685E-11	-0.02%
Se-77	---	2.702E-08	2.701E-08	-0.03%	2.702E-08	2.701E-08	-0.03%
Se-78	---	6.995E-08	6.993E-08	-0.02%	6.995E-08	6.993E-08	-0.02%
Se-79	---	1.196E-07	1.195E-07	-0.03%	1.196E-07	1.195E-07	-0.03%
Se-80	---	2.160E-07	2.160E-07	-0.03%	2.160E-07	2.160E-07	-0.03%
Se-82	---	4.723E-07	4.721E-07	-0.03%	4.723E-07	4.721E-07	-0.03%
Br-81	---	3.071E-07	3.070E-07	-0.03%	3.071E-07	3.070E-07	-0.03%
Kr-82	---	4.819E-09	4.818E-09	-0.02%	4.819E-09	4.818E-09	-0.02%
Kr-83	---	6.872E-07	6.869E-07	-0.03%	6.872E-07	6.869E-07	-0.03%
Kr-84	---	1.242E-06	1.241E-06	-0.04%	1.242E-06	1.241E-06	-0.04%
Kr-85	---	2.773E-07	2.772E-07	-0.03%	2.630E-07	2.630E-07	-0.03%
Kr-86	---	1.894E-06	1.893E-06	-0.04%	1.894E-06	1.893E-06	-0.04%
Rb-85	---	1.010E-06	1.009E-06	-0.04%	1.024E-06	1.024E-06	-0.04%
Rb-86	---	2.199E-09	2.198E-09	-0.03%	3.125E-14	3.124E-14	-0.03%
Rb-87	---	2.420E-06	2.419E-06	-0.04%	2.420E-06	2.419E-06	-0.04%
Sr-86	---	1.148E-08	1.147E-08	-0.03%	1.368E-08	1.367E-08	-0.03%
Sr-87	---	4.590E-11	4.585E-11	-0.11%	4.590E-11	4.585E-11	-0.11%
Sr-88	---	3.183E-06	3.181E-06	-0.04%	3.183E-06	3.181E-06	-0.04%
Sr-89	---	9.991E-07	9.987E-07	-0.04%	1.631E-08	1.630E-08	-0.04%
Sr-90	---	4.872E-06	4.870E-06	-0.04%	4.776E-06	4.774E-06	-0.04%
Y-89	---	3.203E-06	3.202E-06	-0.04%	4.186E-06	4.184E-06	-0.04%
Y-90	---	1.300E-09	1.299E-09	-0.11%	1.214E-09	1.213E-09	-0.11%
Y-91	---	1.611E-06	1.611E-06	-0.04%	4.610E-08	4.608E-08	-0.04%
Zr-90	---	4.985E-08	4.980E-08	-0.11%	1.454E-07	1.453E-07	-0.11%
Zr-91	---	4.303E-06	4.302E-06	-0.04%	5.869E-06	5.866E-06	-0.04%
Zr-92	---	7.125E-06	7.122E-06	-0.04%	7.125E-06	7.122E-06	-0.04%
Zr-93	---	8.690E-06	8.686E-06	-0.04%	8.690E-06	8.686E-06	-0.04%
Zr-94	---	9.630E-06	9.626E-06	-0.04%	9.630E-06	9.626E-06	-0.04%
Zr-95	---	3.106E-06	3.104E-06	-0.04%	1.206E-07	1.206E-07	-0.04%
Zr-96	---	1.120E-05	1.120E-05	-0.04%	1.120E-05	1.120E-05	-0.04%
Nb-93	---	2.695E-13	2.694E-13	-0.04%	5.391E-13	5.386E-13	-0.08%
Nb-93m	---	1.560E-12	1.558E-12	-0.11%	4.526E-12	4.521E-12	-0.11%
Nb-94	---	8.605E-12	8.606E-12	0.01%	8.605E-12	8.606E-12	0.01%
Nb-95	---	1.615E-06	1.614E-06	-0.04%	1.397E-07	1.396E-07	-0.04%
Mo-92	---	8.791E-21	8.793E-21	0.02%	8.791E-21	8.793E-21	0.02%
Mo-94	---	1.584E-14	1.584E-14	0.02%	1.608E-14	1.608E-14	0.01%
Mo-95	---	5.725E-06	5.723E-06	-0.04%	1.019E-05	1.018E-05	-0.04%
Mo-96	---	1.012E-07	1.012E-07	-0.04%	1.012E-07	1.012E-07	-0.04%
Mo-97	---	1.182E-05	1.182E-05	-0.04%	1.182E-05	1.182E-05	-0.04%
Mo-98	---	1.267E-05	1.266E-05	-0.04%	1.267E-05	1.266E-05	-0.04%
Mo-99	---	1.790E-07	1.789E-07	-0.05%	2.417E-40	-3.950E-17	-1.6E+23
Mo-100	---	1.479E-05	1.479E-05	-0.04%	1.479E-05	1.479E-05	-0.04%
Tc-99	---	1.305E-05	1.305E-05	-0.04%	1.323E-05	1.323E-05	-0.04%

表 D.3.4　検証計算結果（クリロフ部分空間法）：核分裂生成物 (2/5)

核種	燃焼前 [10^{24}/cm^3]	燃焼直後[10^{24}/cm^3]		M/S-1	冷却後[10^{24}/cm^3]		M/S-1
		SRAC	MARBLE		SRAC	MARBLE	
Ru-100	---	3.663E-07	3.661E-07	-0.04%	3.663E-07	3.661E-07	-0.04%
Ru-101	---	1.410E-05	1.409E-05	-0.04%	1.410E-05	1.409E-05	-0.04%
Ru-102	---	1.531E-05	1.530E-05	-0.04%	1.531E-05	1.530E-05	-0.04%
Ru-103	---	2.837E-06	2.836E-06	-0.04%	1.421E-08	1.420E-08	-0.04%
Ru-104	---	1.444E-05	1.443E-05	-0.04%	1.444E-05	1.443E-05	-0.04%
Ru-105	---	1.086E-08	1.085E-08	-0.07%	0.000E+00	-3.077E-18	---
Ru-106	---	7.607E-06	7.604E-06	-0.03%	4.361E-06	4.361E-06	0.01%
Rh-103	---	1.196E-05	1.196E-05	-0.04%	1.478E-05	1.478E-05	-0.04%
Rh-105	---	8.621E-08	8.617E-08	-0.06%	0.000E+00	-1.093E-18	---
Rh-106	---	7.020E-12	7.014E-12	-0.10%	4.024E-12	4.022E-12	-0.06%
Pd-104	---	2.753E-07	2.752E-07	-0.04%	2.753E-07	2.752E-07	-0.04%
Pd-105	---	1.158E-05	1.157E-05	-0.04%	1.168E-05	1.167E-05	-0.04%
Pd-106	---	2.834E-06	2.832E-06	-0.09%	6.080E-06	6.075E-06	-0.09%
Pd-107	---	6.832E-06	6.828E-06	-0.06%	6.832E-06	6.828E-06	-0.06%
Pd-108	---	4.932E-06	4.929E-06	-0.06%	4.932E-06	4.929E-06	-0.06%
Pd-110	---	1.630E-06	1.629E-06	-0.06%	1.630E-06	1.629E-06	-0.06%
Ag-107	---	2.995E-13	2.991E-13	-0.12%	8.982E-13	8.971E-13	-0.12%
Ag-109	---	3.963E-06	3.961E-06	-0.04%	3.963E-06	3.961E-06	-0.04%
Ag-110m	---	1.797E-08	1.796E-08	-0.05%	7.815E-09	7.811E-09	-0.05%
Cd-110	---	1.095E-07	1.095E-07	-0.04%	1.197E-07	1.196E-07	-0.04%
Cd-111	---	8.625E-07	8.620E-07	-0.06%	8.625E-07	8.620E-07	-0.06%
Cd-112	---	4.970E-07	4.965E-07	-0.11%	4.970E-07	4.965E-07	-0.11%
Cd-113	---	2.930E-07	2.928E-07	-0.08%	2.930E-07	2.928E-07	-0.08%
Cd-113m	---	5.118E-09	5.108E-09	-0.20%	4.915E-09	4.905E-09	-0.19%
Cd-114	---	2.099E-07	2.099E-07	-0.02%	2.099E-07	2.099E-07	-0.02%
Cd-116	---	1.454E-07	1.453E-07	-0.06%	1.454E-07	1.453E-07	-0.06%
In-113	---	9.944E-11	9.902E-11	-0.43%	3.019E-10	3.009E-10	-0.32%
In-115	---	1.579E-07	1.578E-07	-0.05%	1.579E-07	1.578E-07	-0.05%
Sn-116	---	4.617E-09	4.615E-09	-0.03%	4.617E-09	4.615E-09	-0.03%
Sn-117	---	1.668E-07	1.668E-07	-0.04%	1.668E-07	1.668E-07	-0.04%
Sn-118	---	1.453E-07	1.452E-07	-0.05%	1.453E-07	1.452E-07	-0.05%
Sn-119	---	1.418E-07	1.417E-07	-0.05%	1.418E-07	1.417E-07	-0.05%
Sn-119m	---	5.259E-10	5.258E-10	-0.03%	2.587E-10	2.586E-10	-0.03%
Sn-120	---	1.428E-07	1.428E-07	-0.05%	1.428E-07	1.428E-07	-0.05%
Sn-121	---	7.896E-10	7.890E-10	-0.07%	5.151E-13	5.147E-13	-0.09%
Sn-121m	---	9.535E-09	9.533E-09	-0.03%	9.412E-09	9.410E-09	-0.03%
Sn-122	---	1.671E-07	1.670E-07	-0.04%	1.671E-07	1.670E-07	-0.04%
Sn-123	---	6.046E-09	6.045E-09	-0.02%	1.209E-09	1.209E-09	-0.02%
Sn-124	---	2.621E-07	2.621E-07	-0.03%	2.621E-07	2.621E-07	-0.03%
Sn-126	---	5.696E-07	5.695E-07	-0.02%	5.696E-07	5.695E-07	-0.02%
Sb-121	---	1.428E-07	1.427E-07	-0.05%	1.437E-07	1.436E-07	-0.05%
Sb-123	---	1.790E-07	1.789E-07	-0.04%	1.838E-07	1.838E-07	-0.04%
Sb-124	---	1.043E-09	1.043E-09	-0.04%	3.298E-11	3.296E-11	-0.04%
Sb-125	---	2.478E-07	2.477E-07	-0.01%	2.015E-07	2.015E-07	0.00%
Sb-126	---	8.710E-10	8.709E-10	-0.01%	2.727E-14	2.725E-14	-0.09%
Sb-126m	---	3.481E-13	3.481E-13	0.02%	2.075E-16	2.075E-16	-0.01%
Te-122	---	2.907E-09	2.905E-09	-0.05%	2.907E-09	2.905E-09	-0.05%

表 D.3.5　検証計算結果（クリロフ部分空間法）: 核分裂生成物 (3/5)

核種	燃焼前 [10^{24}/cm^3]	燃焼直後[10^{24}/cm^3]		M/S-1	冷却後[10^{24}/cm^3]		M/S-1
		SRAC	MARBLE		SRAC	MARBLE	
Te-123	---	2.305E-11	2.303E-11	-0.06%	2.493E-11	2.492E-11	-0.06%
Te-123m	---	2.289E-12	2.288E-12	-0.05%	4.030E-13	4.028E-13	-0.05%
Te-124	---	1.539E-09	1.539E-09	-0.04%	2.549E-09	2.548E-09	-0.04%
Te-125	---	2.410E-08	2.408E-08	-0.09%	7.001E-08	6.996E-08	-0.08%
Te-125m	---	2.305E-09	2.303E-09	-0.08%	2.631E-09	2.629E-09	-0.07%
Te-126	---	1.411E-08	1.411E-08	0.00%	1.499E-08	1.499E-08	0.00%
Te-127m	---	6.130E-08	6.129E-08	-0.02%	9.098E-09	9.096E-09	-0.02%
Te-128	---	1.839E-06	1.838E-06	-0.02%	1.839E-06	1.838E-06	-0.02%
Te-129m	---	5.358E-08	5.357E-08	-0.02%	1.100E-10	1.099E-10	-0.02%
Te-130	---	5.219E-06	5.217E-06	-0.03%	5.219E-06	5.217E-06	-0.03%
Te-132	---	1.741E-07	1.740E-07	-0.05%	1.133E-35	1.028E-16	9.1E+18
I-127	---	9.247E-07	9.245E-07	-0.02%	9.769E-07	9.767E-07	-0.02%
I-129	---	2.921E-06	2.920E-06	-0.02%	2.974E-06	2.973E-06	-0.02%
I-130	---	2.718E-10	2.718E-10	-0.02%	0.000E+00	-1.405E-22	---
I-131	---	3.191E-07	3.190E-07	-0.04%	1.756E-18	5.878E-17	3.2E+01
I-135	---	1.885E-08	1.884E-08	-0.08%	0.000E+00	-3.791E-20	---
Xe-126	---	2.037E-15	2.037E-15	0.02%	2.037E-15	2.037E-15	0.02%
Xe-128	---	2.407E-08	2.406E-08	-0.02%	2.407E-08	2.406E-08	-0.02%
Xe-129	---	1.935E-10	1.933E-10	-0.07%	1.936E-10	1.934E-10	-0.07%
Xe-130	---	5.764E-08	5.762E-08	-0.03%	5.791E-08	5.789E-08	-0.03%
Xe-131	---	7.890E-06	7.887E-06	-0.04%	8.209E-06	8.206E-06	-0.04%
Xe-132	---	1.164E-05	1.164E-05	-0.04%	1.182E-05	1.181E-05	-0.04%
Xe-133	---	3.920E-07	3.919E-07	-0.04%	2.418E-24	3.596E-17	1.5E+07
Xe-134	---	1.704E-05	1.703E-05	-0.04%	1.704E-05	1.703E-05	-0.04%
Xe-135	---	3.038E-08	3.036E-08	-0.06%	0.000E+00	-3.427E-20	---
Xe-136	---	1.566E-05	1.566E-05	-0.04%	1.566E-05	1.566E-05	-0.04%
Cs-133	---	1.490E-05	1.490E-05	-0.04%	1.530E-05	1.529E-05	-0.04%
Cs-134	---	2.941E-07	2.940E-07	-0.04%	2.232E-07	2.231E-07	-0.02%
Cs-135	---	1.653E-05	1.653E-05	-0.04%	1.658E-05	1.658E-05	-0.04%
Cs-136	---	3.324E-08	3.323E-08	-0.02%	4.563E-15	4.561E-15	-0.03%
Cs-137	---	1.460E-05	1.459E-05	-0.04%	1.433E-05	1.432E-05	-0.04%
Ba-134	---	2.728E-08	2.725E-08	-0.11%	9.818E-08	9.809E-08	-0.10%
Ba-135	---	1.541E-10	1.564E-10	1.50%	1.582E-10	1.605E-10	1.46%
Ba-136	---	3.395E-07	3.394E-07	-0.01%	3.727E-07	3.726E-07	-0.01%
Ba-137	---	1.488E-07	1.487E-07	-0.10%	4.228E-07	4.224E-07	-0.11%
Ba-137m	---	2.290E-12	2.287E-12	-0.10%	2.191E-12	2.188E-12	-0.10%
Ba-138	---	1.387E-05	1.387E-05	-0.04%	1.387E-05	1.387E-05	-0.04%
Ba-140	---	7.505E-07	7.502E-07	-0.04%	6.215E-14	6.205E-14	-0.15%
La-139	---	1.311E-05	1.310E-05	-0.04%	1.311E-05	1.310E-05	-0.04%
La-140	---	9.923E-08	9.919E-08	-0.04%	9.418E-15	9.525E-15	1.14%
Ce-140	---	1.154E-05	1.153E-05	-0.04%	1.239E-05	1.238E-05	-0.04%
Ce-141	---	1.838E-06	1.837E-06	-0.04%	3.060E-09	3.059E-09	-0.04%
Ce-142	---	1.093E-05	1.092E-05	-0.04%	1.093E-05	1.092E-05	-0.04%
Ce-143	---	6.626E-08	6.622E-08	-0.06%	0.000E+00	1.020E-16	---
Ce-144	---	6.278E-06	6.275E-06	-0.05%	3.026E-06	3.024E-06	-0.05%
Pr-141	---	9.966E-06	9.962E-06	-0.04%	1.180E-05	1.180E-05	-0.04%
Pr-143	---	6.532E-07	6.529E-07	-0.04%	1.609E-13	1.607E-13	-0.08%

表 D.3.6　検証計算結果（クリロフ部分空間法）: 核分裂生成物 (4/5)

核種	燃焼前 [10^{24}/cm^3]	燃焼直後[10^{24}/cm^3]		M/S-1	冷却後[10^{24}/cm^3]		M/S-1
		SRAC	MARBLE		SRAC	MARBLE	
Pr-144	---	2.649E-10	2.648E-10	-0.05%	1.275E-10	1.274E-10	-0.05%
Nd-142	---	6.356E-08	6.353E-08	-0.05%	6.356E-08	6.353E-08	-0.05%
Nd-143	---	9.237E-06	9.233E-06	-0.05%	9.957E-06	9.952E-06	-0.05%
Nd-144	---	2.719E-06	2.718E-06	-0.05%	5.972E-06	5.969E-06	-0.05%
Nd-145	---	7.015E-06	7.011E-06	-0.06%	7.015E-06	7.011E-06	-0.06%
Nd-146	---	6.188E-06	6.185E-06	-0.05%	6.188E-06	6.185E-06	-0.05%
Nd-147	---	2.537E-07	2.536E-07	-0.05%	1.512E-15	1.504E-15	-0.48%
Nd-148	---	4.018E-06	4.016E-06	-0.05%	4.018E-06	4.016E-06	-0.05%
Nd-150	---	2.404E-06	2.403E-06	-0.05%	2.404E-06	2.403E-06	-0.05%
Pm-147	---	3.918E-06	3.916E-06	-0.04%	3.359E-06	3.358E-06	-0.03%
Pm-148	---	5.193E-09	5.191E-09	-0.04%	1.806E-12	1.805E-12	-0.05%
Pm-148m	---	4.040E-08	4.039E-08	-0.05%	2.626E-10	2.624E-10	-0.05%
Pm-149	---	3.257E-08	3.255E-08	-0.06%	0.000E+00	5.464E-16	---
Pm-151	---	1.052E-08	1.051E-08	-0.06%	0.000E+00	3.372E-18	---
Sm-147	---	4.106E-07	4.101E-07	-0.11%	1.222E-06	1.221E-06	-0.11%
Sm-148	---	1.934E-07	1.933E-07	-0.05%	2.387E-07	2.386E-07	-0.05%
Sm-149	---	2.734E-06	2.733E-06	-0.06%	2.767E-06	2.765E-06	-0.06%
Sm-150	---	3.030E-07	3.029E-07	-0.05%	3.030E-07	3.029E-07	-0.05%
Sm-151	---	1.670E-06	1.669E-06	-0.05%	1.670E-06	1.669E-06	-0.05%
Sm-152	---	1.530E-06	1.529E-06	-0.04%	1.530E-06	1.529E-06	-0.04%
Sm-153	---	1.013E-08	1.012E-08	-0.06%	0.000E+00	-1.844E-16	---
Sm-154	---	6.397E-07	6.394E-07	-0.05%	6.397E-07	6.394E-07	-0.05%
Eu-151	---	4.843E-09	4.837E-09	-0.12%	1.545E-08	1.543E-08	-0.12%
Eu-152	---	3.440E-10	3.435E-10	-0.13%	3.298E-10	3.294E-10	-0.13%
Eu-153	---	9.422E-07	9.418E-07	-0.04%	9.523E-07	9.519E-07	-0.04%
Eu-154	---	1.031E-07	1.030E-07	-0.05%	9.646E-08	9.641E-08	-0.04%
Eu-155	---	4.799E-07	4.797E-07	-0.05%	4.258E-07	4.256E-07	-0.04%
Eu-156	---	2.875E-08	2.874E-08	-0.05%	3.261E-14	3.261E-14	-0.01%
Eu-157	---	7.587E-10	7.582E-10	-0.07%	0.000E+00	-2.120E-21	---
Gd-152	---	1.516E-10	1.515E-10	-0.12%	1.555E-10	1.553E-10	-0.12%
Gd-154	---	2.316E-09	2.313E-09	-0.12%	8.926E-09	8.916E-09	-0.11%
Gd-155	---	2.778E-08	2.721E-08	-2.07%	8.191E-08	8.127E-08	-0.78%
Gd-156	---	3.357E-07	3.363E-07	0.18%	3.645E-07	3.651E-07	0.16%
Gd-157	---	2.457E-07	2.456E-07	-0.04%	2.464E-07	2.463E-07	-0.04%
Gd-158	---	1.747E-07	1.746E-07	-0.05%	1.747E-07	1.746E-07	-0.05%
Gd-160	---	5.616E-08	5.614E-08	-0.05%	5.616E-08	5.614E-08	-0.05%
Tb-159	---	8.578E-08	8.574E-08	-0.05%	8.578E-08	8.574E-08	-0.05%
Tb-160	---	3.580E-09	3.578E-09	-0.05%	2.017E-10	2.016E-10	-0.05%
Dy-160	---	3.935E-09	3.933E-09	-0.05%	7.313E-09	7.309E-09	-0.05%
Dy-161	---	2.083E-08	2.082E-08	-0.07%	2.083E-08	2.082E-08	-0.07%
Dy-162	---	1.708E-08	1.708E-08	-0.05%	1.708E-08	1.708E-08	-0.05%
Dy-163	---	8.245E-09	8.241E-09	-0.05%	8.245E-09	8.241E-09	-0.05%
Dy-164	---	5.073E-09	5.070E-09	-0.05%	5.073E-09	5.070E-09	-0.05%
Ho-163	---	1.347E-14	1.347E-14	0.02%	1.347E-14	1.347E-14	0.02%
Ho-165	---	2.139E-09	2.138E-09	-0.04%	2.139E-09	2.138E-09	-0.04%
Ho-166m	---	1.040E-11	1.040E-11	-0.02%	1.040E-11	1.039E-11	-0.02%
Er-162	---	0.000E+00	0.000E+00	---	0.000E+00	0.000E+00	---

表 D.3.7　検証計算結果（クリロフ部分空間法）: 核分裂生成物 (5/5)

核種	燃焼前 [10^{24}/cm^3]	燃焼直後[10^{24}/cm^3]		M/S-1	冷却後[10^{24}/cm^3]		M/S-1
		SRAC	MARBLE		SRAC	MARBLE	
Er-164	---	6.459E-14	6.460E-14	0.02%	6.459E-14	6.460E-14	0.02%
Er-166	---	1.478E-09	1.477E-09	-0.03%	1.478E-09	1.477E-09	-0.03%
Er-167	---	6.287E-10	6.285E-10	-0.03%	6.287E-10	6.285E-10	-0.03%
Er-168	---	2.422E-10	2.421E-10	-0.05%	2.422E-10	2.421E-10	-0.05%
Er-170	---	2.452E-11	2.451E-11	-0.04%	2.452E-11	2.451E-11	-0.04%
Hf-176	---	0.000E+00	0.000E+00	---	0.000E+00	0.000E+00	---
Hf-177	---	0.000E+00	0.000E+00	---	0.000E+00	0.000E+00	---
Hf-178	---	0.000E+00	0.000E+00	---	0.000E+00	0.000E+00	---
Hf-179	---	0.000E+00	0.000E+00	---	0.000E+00	0.000E+00	---
Hf-180	---	0.000E+00	0.000E+00	---	0.000E+00	0.000E+00	---

付録E　摂動計算ソルバーPERTURBATION

従来システムには、摂動計算のソルバーとして拡散近似に基づく PERKY と輸送理論に基づく SNPERT、SNPERT3D がある。これらのコードは炉心計算コードと連携して利用することを想定しており、それぞれ、CITATION、TWOTRAN、TRITAC と結合して使う。しかしながら、炉心計算ソルバーは今後変更される可能性が高いので摂動計算ソルバーを炉心計算ソルバーと分離しておく方が拡張性の観点から望ましい。このため、拡散と輸送の両方に対応した摂動計算ソルバーの原型として PERTURBATION ソルバーを作成した。現状の PERTURBATION ソルバーは Python で開発されており 3 次元 XYZ 体系の反応度価値計算にのみ対応している。

E.1　摂動理論に基づく反応度計算式

E.1.1　輸送摂動理論

PERTURBATION ソルバーの輸送摂動計算としては、SNPERT3D の報告書 [?] で「輸送摂動式２」と呼ばれている以下の式を採用している。

$$\frac{\delta k}{kk'} = \frac{F + A + S + L}{D} \tag{E.1}$$

$$D = \int dr \int dE\ \phi^+(E,r)\chi(E,r) \int dE'\ \nu\Sigma_f(E',r)\phi'(E',r) \tag{E.2}$$

$$F = \lambda' \int dr \int dE\ \phi^+(E,r)\chi(E,r) \int dE'\delta\nu\Sigma_f(E',r)\phi'(E',r) \tag{E.3}$$

$$A = \int dr \int dE\ \phi^+(E,r)\delta\Sigma_a(E,r)\phi'(E,r) \tag{E.4}$$

$$S = \int dr \int dE\ \phi^+(E,r) \int dE'\ \delta\Sigma_s(E \to E',r)\phi'(E',r) \tag{E.5}$$

$$L = -\int dr \int dE \int d\Omega\phi^+(E,r,\Omega)\delta\Sigma_t(E,r)\left[4\pi\phi'(E,r,\Omega) - \phi'(E,r)\right] \tag{E.6}$$

この式に多群近似を導入すると以下のように変形することができる。

$$\frac{\delta k}{kk'} = \sum_r \sum_g ([F]^{g,r} + [A]^{g,r} + [S]^{g,r} + [L]^{g,r}) \tag{E.7}$$

$$[D] = \sum_r \sum_g \sum_h \chi^{g,r} < \phi_g^+, \phi_h' > \nu\Sigma_f^{h,r} \tag{E.8}$$

$$[F]^{g,r} = \lambda' \sum_h \chi^{g,r} < \phi_g^+, \phi_h' >_r \nu\Sigma_f^{g,r}/[D] \tag{E.9}$$

$$[A]^{g,r} = \delta\Sigma_a < \phi_g^+, \phi_h' >^r /[D] \tag{E.10}$$

$$[S]^{g,r} = \sum_{h>g} \delta\Sigma_s^{g\to h} \left[< \phi_h^+, \phi_g' >^r - < \phi_g^+, \phi_g' >^r \right] /[D] \tag{E.11}$$

$$[L]^{g,r} = \delta\Sigma_t^{g,r} \left[< \phi_g^+, \phi_g' >^r -4\pi < \phi^+, \phi' >^{g,r} \right] /[D] \tag{E.12}$$

この式を使うことで領域、エネルギー群に分けて反応度を計算することができる。

E.1.2 拡散摂動理論

拡散摂動では漏洩項のみが異なり、漏洩項は以下の式で計算することができる。

$$\frac{\delta k}{kk'} = \frac{[F] + [A] + [S] + [L_d]}{[D]} \tag{E.13}$$

$$[L_d] = -\frac{1}{D} \int dr \int dE \int \delta D(E,r) \nabla\phi^+(E,r) \nabla\phi'(E,r) \tag{E.14}$$

多群近似を導入すると以下の変形できる。

$$\frac{\delta k}{kk'} = \sum_r \sum_g ([F]^{g,r} + [A]^{g,r} + [S]^{g,r} + [L]^{g,r}) \tag{E.15}$$

$$[L_d]^{g,r} = -\delta D^{g,r} < \nabla\phi_g^+, \nabla\phi_g' >^r /[D] \tag{E.16}$$

この式を領域、エネルギー群に分けて反応度を計算することができる。PERTURBATION ソルバーの拡散摂動計算ではこの式が用いられている。

E.2 実装

PERTURABATION は Python と Numpy ライブラリを使って実装した。Python で明示的に for 文によるループを使って計算すると実行効率が悪く実用的でなくなってしまうが、Numpy を用いて明示的なループを避けたプログラムにすることで、ある程度実用的な計算速度を実現している。また、拡散摂動と輸送摂動でコードを共通化し、Python で開発したことで従来システムに比べてプログラムは小さくなっている。

E.3 検証計算

実装した摂動計算ソルバーを検証するため、従来システムの摂動計算コード SNPERT3D と PERKY による計算結果との比較を行った。検証計算に用いた問題は ZPPR-9 のナトリウムボイド反応度（STEP 6) であり、エネルギー群数は 7 群である。

輸送摂動、拡散摂動ソルバーの検証計算結果をそれぞれ表 E.3.1、E.3.2 に示す。この結果から分かるように相対差で 0.01 %の桁で若干の差が見られるが絶対値では $10^{-7}\Delta k/kk'$ であり、実用上問題ないと考えられる。

表 E.3.1　PERTURBATION ソルバー検証計算結果（輸送摂動）

	PERTURBATION (Δk/kk')	SNPERT-3D (Δk/kk')	相対差 (%)	絶対差 (Δk/kk')
摂動分母	4.582E-06	4.582E-06	0.00	8.7E-12
生成項	-4.620E-05	-4.623E-05	-0.07	3.1E-08
吸収項	5.207E-04	5.208E-04	-0.01	-3.7E-08
散乱項	1.873E-03	1.873E-03	0.00	-5.2E-09
漏洩項	-8.927E-04	-8.928E-04	0.00	2.6E-09
合計	1.455E-03	1.455E-03	0.00	-8.0E-09

表 E.3.2　PERTURBATION ソルバー検証計算結果（拡散摂動）

	PERTURBATION (Δk/kk')	PERKY (Δk/kk')	相対差 (%)	絶対差 (Δk/kk')
摂動分母	3.708E-06	3.708E-06	0.00	2.5E-11
生成項	-6.494E-04	-6.494E-04	0.00	4.4E-09
吸収項	9.020E-04	9.017E-04	0.03	2.9E-07
散乱項	2.020E-03	2.020E-03	0.00	-9.7E-08
漏洩項	-1.490E-03	-1.490E-03	0.00	4.6E-09
合計	7.827E-04	7.825E-04	0.03	2.0E-07

付録F　核設計精度評価ソルバーUNCERTAINTY

MARBLEの炉定数調整・核設計精度評価ソルバーについては、先行して開発が進められ詳細は文献[6]にまとめられている。このソルバーはその後も開発を継続し、基本炉定数法、従来バイアス因子法、炉定数調整法[76,77]に加えて、一般化バイアス因子法[78]及び拡張バイアス因子法[79]を追加した[80]。

F.1　実装

新たな手法が追加されたことで、従来のCrossSectionAdjustmentモジュール、CrossSectionAdjustmentUtilsモジュールはモジュールの名称として不適切になったため、それぞれ、Uncertaintyモジュール、UncertaintyUtilsモジュールに変更されている。しかしながら、一般化バイアス因子法と拡張バイアス因子法の追加において、基本的なクラス構造を変更する必要はなかったのでクラス構造に関する本質的な違いはない。従来と同様に、Uncertaintyパッケージ（marble.solvercore.uncertainty）に含まれる以下のクラスを使うことで各手法による核設計精度評価を行うことができる。新たに追加されたクラスについても、すべてUncertaintyPredictionクラスのサブクラスであるので基本的なインターフェースは同じであり、入力データの作成方法は従来からあるクラスと同様である。

① ReferenceMethod（基準炉定数法）
② BiasMethod（従来バイアス因子法）
③ AdjustmentMethod（炉定数調整法）
④ GeneralizedBiasMethod（一般化バイアス因子法）
⑤ ExtendedBiasLCMethod（拡張バイアス因子法(LC法)）
⑥ ExtendedBiasPEMethod（拡張バイアス因子法(PE法)）

F.2　検証計算

UNCERTAINTYソルバーに新たに追加された一般化バイアス因子法と拡張バイアス因子法の検証計算としては、拡張バイアス因子法に関する文献[79]の計算で用いられたツールを参照解として比較を行った。計算ケースとしては、ZPPR-9、ZPPR-10A、ZPPR-10Bの臨界性の3核特性を使って、動燃60万kWe炉心[29]の臨界性に対する設計予測精度を評価した。UNCERTAINTYソルバーによる計算結果を表F.2.1に示す。この表に示された有効数字の範囲内で参照解と完全一致することを確認した。この結果からUNCERTAINTYソルバーは一般化バイアス因子法、拡張バイアス因子法についても正しく計算できていると判断することができる。

表 F.2.1　UNCERTAINTY ソルバーの検証計算結果

	一般化バイアス因子法	拡張バイアス因子法	
		LC 法	PE 法
ZPPR-9 の重み	0.41717	0.41717	0.12780
ZPPR-10A の重み	0.25983	0.25983	0.23478
ZPPR-10B の重み	0.32299	0.32299	0.26414
設計予測精度（分散）	1.409E-5	1.409E-5	1.727E-6
バイアス因子	1.00514	1.00514	1.00324

すべての値が表示された有効数字範囲内で参照解と完全一致した。

付録G　FortranUtilsユーザマニュアル

従来システムに含まれる解析コードはFORTRANで開発されている。このため、従来システムで使われているファイル形式は、FORTRANのテキスト形式（A、I、F、E形編集記述子等）やバイナリ形式を使うことが前提となっている。Pythonにも書式を指定してファイルを扱う方法はあるが、FORTRANの入出力機能（I/O）は数値データを取り扱うための便利な機能を数多く持っているため、Pythonの書式指定方法を使ってFORTRANと同じ動作をさせるのはそれほど簡単ではない。

MARBLEの開発でPythonを採用する場合、従来システムで使われているFORTRAN形式のファイルをどのように扱い、PythonとFORTRANを連携させるかが大きな課題であった。FORTRANのI/Oを模擬するオープンソースライブラリ（ScientificPython）なども存在するが、機能的に不十分であることから、結果としてMARBLEではFORTRANのI/Oを模擬する独自のモジュールをPythonで開発することにした。このモジュールがFortranUtilsである。

FORTRANで開発された外部コードをMARBLE上で利用する場合には、FortranUtilsを利用することで、FORTRANのI/Oと同等の動作をPython上で簡単に実現できる。

G.1　基本機能

FortranUtilsモジュールは、FORTRANのREAD、WRITE、FORMAT、REWINDに対応する機能を提供する。テキスト形式（書式付き入出力）、バイナリ形式（書式なし入出力）ともに対応している。また、フリーフォーマット（並びによる書式）、NAMELIST（名前付き書式）にも対応している。

以下の関数及びクラスを使うことでFORTRANのI/Oを模擬することができる。

- fortran_read関数
- fortran_write関数
- fortran_rewind関数
- fortran_backspace関数
- Namelistクラス

G.2　使用方法

G.2.1　fortran_read関数

FORTRANのREAD文と同じ処理をしたい場合には、fortran_read関数を用いる。テキスト形式の場合、FORTRANのフォーマット文を使うことができる。また、FORTRANと同様にフォー

マット文を省略することでバイナリ形式として処理される。NAMELIST 形式で処理したい場合には、fortran_read 関数の nml 引数に後述する Namelist クラスのインスタンスを渡す。

```
fortran_read(*args, **kwargs)
    FORTRAN の READ 文に相当する FORTRAN 形式でのファイル入力を行う。

    テキスト（書式付き）ファイルの場合：
      fortran_read(fin, format, lenvars) --> numpy.ndarray
        fin     -- Python ファイルオブジェクト (file, StringIO, ...)
        format  -- FORTRAN の書式を表す文字列

                   並びによる書式（フリーフォーマット）の場合"*"
        lenvars -- データの数と型を表すタプルのリスト

    バイナリ（書式なし）ファイルの場合：
      fortran_read(fin, lenvars) --> numpy.ndarray
        fin     -- Python ファイルオブジェクト (file, StringIO, ...)
        lenvars -- データの数と型を表すタプルのリスト

    NAMELIST の場合：
      fortran_read(fin, nml=Namelist オブジェクト) --> Namelist オブジェクト
        fin -- Python ファイルオブジェクト (file, StringIO, ...)
        nml -- Namelist オブジェクト

    補足：
       読み込むデータの数と型は [(1, "i")]（1 個の整数配列）のように
       numpy.zeros() の引数と同じ形式で指定すると、読み込んだデータは
       numpy.ndarray のインスタンスを返す。

    利用例：
        以下に Python の fortran_read 関数で FORTRAN の READ 文を模擬
        した例を示す。

        Fortran:
          INTEGER*4 IDATA(6)
          REAL*4 ADATA(7)
          INTEGER*4 III(3)
          NAMELIST /MYDATA/ III
          READ(5, '(6I3)') IDATA
          READ(1) ADATA
          READ(10, NML=MYDATA)
        <=>
```

```
    Python:
      mydata = Namelist(iii=(3, "i"))
      idata = fortran_read(sys.stdin, "6I3", [(6, "i")]))
      adata = fortran_read(open("fort.1"), [(7, "f")]))
      mydata = fortran_read(open("fort.10"), nml=mydata)
```

G.2.2 fortran_write 関数

FORTRAN の WRITE 文と同じ処理をしたい場合には、fortran_write 関数を用いる。テキスト形式の場合、FORTRAN のフォーマット文を使うことができる。また、FORTRAN と同様にフォーマット文を省略することでバイナリ形式として処理される。バイナリ形式の場合はデータの型が必要となるが、fortran_read 関数は Numpy の配列のデータ型に応じて処理を行う。すなわち、単精度の配列であれば単精度で、倍精度の配列であれば倍精度でファイルに書き出す。NAMELIST 形式で処理したい場合には、fortran_write 関数の nml 引数に後述する Namelist クラスのインスタンスを渡す。

```
fortran_write(*args, *kwargs)
    FORTRAN の WRITE 文に相当する FORTRAN 形式でのファイル出力を行う。

    テキスト（書式付き）ファイルの場合：
      fortran_write(fout, format, variables) --> None
        fout      -- Python ファイルオブジェクト (file, StringIO, ...)
        format    -- FORTRAN の書式を表す文字列
        variables -- 書き出すデータをまとめた Python のリスト

    バイナリ（書式なし）ファイルの場合：
      fortran_write(fout, variables) --> None
        fout      -- Python file object (file, StringIO, ...)
        variables -- 書き出すデータをまとめた Python のリスト
                     書き出すデータは numpy.ndarray を用いることで
                     データ型を指定する。

    NAMELIST の場合：
      fortran_write(fout, nml=Namelist) --> None
        fout -- Python ファイルオブジェクト (file, StringIO, ...)
        nml  -- Namelist オブジェクト

    利用例：
        以下に Python の fortran_write 関数で FORTRAN の WRITE 文を模擬
        した例を示す。
```

```
Fortran:
  INTEGER*4 IDATA(6)
  REAL*4 ADATA(7)
  INTEGER*4 III(3)
  NAMELIST /MYDATA/ III
  DATA IDATA/1,2,3,4,5,6/
  DATA ADATA/1.,2.,3.,4.,5.,6.,7./
  DATA III/1,2,3/
  WRITE(6, '(6I3)') IDATA
  WRITE(1) ADATA
  WRITE(10, NML=MYDATA)
<=>
Python:
  mydata = Namelist(iii=(3, "i"))
  idata = numpy.array([1, 2, 3, 4, 5, 6], dtype="i")
  adata = numpy.array([1., 2., 3., 4., 5., 6.], dtype="f")
  mydata.set("iii", [1, 2, 3])
  fortran_write(sys.stdout, "6I3", [idata])
  fortran_write(open("fort.2", "w", [adata]))
  fortran_write(open("fort.11", "w"), nml=mydata)
```

G.2.3 fortran_rewind 関数

FORTRAN の REWIND 文と同じ処理をしたい場合には、fortran_rewind 関数を用いる。FortranUtils では Python のファイルオブジェクトを用いるので、Python のファイルオブジェクトを直接操作しても問題ない（fileobj.seek(0) と同等である）。

```
fortran_rewind(fileobj)
    FORTRAN の REWIND 文に相当するファイルの頭出しを行う。
```

G.2.4 fortran_backspace 関数

FORTRAN の BACKSPACE 文と同じ処理をしたい場合には、fortran_backspace 関数を用いる。FortranUtils では Python のファイルオブジェクトを用いるので Python のファイルオブジェクトを直接操作しても問題ないが、バイナリ形式の場合は FORTRAN のレコードを適切に扱う必要があるため、fortran_backspace 関数を使う方が簡単である。

```
fortran_backspace(fileobj)
    FORTRAN の BACKSPACE 文に相当するファイルの巻き戻し操作を行う。
```

```
    ファイル（ファイルオブジェクト：fileobj）が指しているファイル中の位
    置を 1 レコード分巻き戻す。

    注意：
      バイナリファイルの場合、file オブジェクトが bainary モードで opne され
      ていなければならない。
```

G.2.5 Namelist クラス

FORTRAN の NAMELIST 文に相当する処理を行いたい場合には、前述の fortran_read 関数、fortran_write 関数と Namelist クラスのインスタンス（オブジェクト）を組み合わせて利用する。

FORTRAN で NAMELIST を使う場合は、データを読み書きする前に NAMELIST 文を使って NAMELIST のデータ構造を定義する。FortranUtils ではこの NAMELIST のデータ構造を定義するために Namelist クラスを利用する。

```
class Namelist
   __init__(self, name, **data)
      Namelist インスタンスを生成する。

      引数：
        name -- NAMELIST の名前
        data -- NAMELIST の変数名とデータ型の定義

      利用例：
        以下に Python の Namelist クラスで FORTRAN の NAMELIST 文を模擬
        した例を示す。

        FORTRAN:
          INTEGER IARRAY(5)
          REAL A
          REAL*8 D
          CHARACTER*3 C
          REAL XARRAY(2,3)
          NAMELIST /MYDATA/ IARRAY, A, D, C, XARRAY
        <=>
        Python:
          mydata = Namelist("mydata", iarray=(5, "i"),
                                      a=(1, "f"),
                                      d=(1, "d"),
                                      c=(1, "S3"),
```

```
                                    xarray=((3, 2), "f"))
```

Python 側で２次元以上の配列を使う場合は C 言語の配列の並び（C-contiguous）を使う必要がある。FORTRAN の配列の並び（Fortran-contiguous）とは添え字の順番が逆になるので注意が必要である。

```
get(self, varname)
```

NAMELIST のデータの値を返す。

```
    Arguments:
      varname -- NAMELIST の変数名
```

```
set(self, varname, data, at=None)
```

NAMELIST のデータの値を設定する。

```
    Arguments:
      varname -- NAMELIST の変数名
      data    -- NAMELIST の変数に設定するデータ
      at      -- 設定する配列データの開始位置
                 （省略した場合は C 言語の配列の並びでの先頭）
```

利用例：

以下に Python の Namelist クラスで FORTRAN の NAMELIST 文を模擬した例を示す。

```
      FORTRAN:
        REAL XARRAY(2, 3)
        NAMELIST /MYDATA/ XARRAY
        XARRAY(1,2) = 3.0
      <=>
      Python:
        mydata = Namelist("mydata", xarray=((3, 2), "f"))
        mydata.set("xarray", 3.0, at=(1, 0))
```

Python 側で２次元以上の配列を使う場合は C 言語の配列の並び（C-contiguous）を使う必要がある。FORTRAN の配列の並び（Fortran-contiguous）とは添え字の順番が逆になるので注意が必要である。また、Python では C 言語と同様に配列の添え字は 0 番目から数える。

付録H　SCHEMEユーザマニュアル

次世代炉心解析システム MARBLE に含まれる高速炉核特性解析システム SCHEME の利用方法について説明する。

H.1　SCHEMEの入力ファイル

SCHEME では MARBLE のオブジェクトを生成・操作するための関数を使って、高速炉核特性解析に必要なデータの作成や解析コードの実行等の一連の解析手順を定義する。このため、入力ファイルには入力データと計算手順の両方を記述しなければならない。

SCHEME の入力ファイルは、オブジェクト指向プログラミング言語 Python のごく一部の機能を使ったものであるが、Python プログラムとして実行することが可能である。このため、入力ファイルを Python プログラムとして実行することで高速炉核特性解析を行うことができる。具体的には入力ファイルを python コマンドの引数に渡せばよい。

以下では入力データの作成、解析コードの実行、データの保存と復元のための SCHEME 関数について説明する。

H.2　入力データの作成

H.2.1　組成データの作成

MARBLE では解析に必要な組成データを MaterialSet というオブジェクトで管理する。MaterialSet オブジェクトを生成するための関数として materialset() が用意されている。

```
materialset(yaml=None, slaromuf=None, casup=None)
    組成データの集合（MaterialSet）オブジェクトを生成する。

    引数：
      yaml -- YAML 形式のテキストデータまたはファイルパス
      slaromuf -- SLAROM-UF の入力ファイルのテキストデータまたはファイルパス
      casup -- CASUP の入力ファイルのテキストデータまたはファイルパス
```

新規に組成データを作成する場合は通常 YAML 形式のテキストデータを使うが、SLAROM-UF コードや CASUP コードの入力ファイルからデータを抽出して作成することもできる。図 H.2.1 に YAML 形式での入力例を示す。最初の行はデータ形式の指定であるので「MaterialSet」で固定である。各セクションの内容については表 H.2.1 に示した。

```
MaterialSet:
  - name: inner_core
    data:
      U-235:    1.09804e-05
      U-238:    5.41001e-03
      Pu-239:   9.01750e-04
      Pu-240:   3.71581e-04
      Pu-241:   2.15854e-04
      Pu-242:   6.14172e-05
      O-16:     1.37340e-02
      Na-23:    8.75367e-03
      Cr-nat.:  3.87561e-03
      Mn-55:    3.66807e-04
      Fe-nat.:  1.38604e-02
      Ni-nat.:  2.72666e-03
      Mo-nat.:  3.08888e-04
  - name: sodium_follower
    data:
      Na-23:   2.04623e-02
      Cr-nat.: 1.37677e-03
      Mn-55:   1.30305e-04
      Fe-nat.: 4.92377e-03
      Ni-nat.: 9.68620e-04
      Mo-nat.: 1.09730e-04
```

図 H.2.1　組成データの入力例

表 H.2.1　MaterialSet のセクション一覧

セクション	内容
name	均質組成の名前を指定する。
data	数密度データを指定する。 核種名: 原子数密度（10^{24} cm^{-3}）

H.2.2 格子計算モデルの作成

格子計算用の形状モデルの情報は、HomoCell、SlabCell、RingCell オブジェクトを使って管理する。SCHEME ではこれらのオブジェクトを作成するための関数として、それぞれ、homocell()、slabcell()、ringcell() が用意されている。

```
homocell(yaml=None, slaromuf=None, casup=None)
    均質セル（HomoCell）オブジェクトを生成する。

    引数:
      yaml -- YAML 形式のテキストデータまたはファイルパス
      slaromuf -- SLAROM-UF の入力ファイルのテキストデータまたはファイルパス
      casup -- CASUP の入力ファイルのテキストデータまたはファイルパス

slabcell(yaml=None, slaromuf=None, casup=None, **kwargs)
    1 次元スラブセル（SlabCell）オブジェクトを生成する。

    引数 :
      yaml -- YAML 形式のテキストデータまたはファイルパス
      slaromuf -- SLAROM-UF の入力ファイルのテキストデータまたはファイルパス
      casup -- CASUP の入力ファイルのテキストデータまたはファイルパス

ringcell(yaml=None, slaromuf=None, casup=None, **kwargs)
    1 次元リングセル（RingCell）オブジェクトを生成する。

    引数 :
      yaml -- YAML 形式のテキストデータまたはファイルパス
      slaromuf -- SLAROM-UF の入力ファイルのテキストデータまたはファイルパス
      casup -- CASUP の入力ファイルのテキストデータまたはファイルパス
```

組成データと同様に YAML 形式の入力ファイルで定義することが可能である。図 H.2.2 に YAML 形式での入力例を示す。各セクションの内容については表 H.2.2 に示した。

```
SlabCell:
  name:
    SCF00
  widths:
    [ 0.22225, 0.635 ,  0.3175 , 0.0381 , 1.1938 , 0.0381 , 0.0381 ,
      0.5588 , 0.0381 , 0.0381 , 0.5588 , 0.0381 , 0.0381 , 0.5588 ,
      0.0381 , 0.3175 , 0.635 ,  0.22225]
  matnames:
    ['REG01', 'REG02', 'REG03', 'REG04', 'REG05', 'REG06', 'REG07', 'REG08',
     'REG09', 'REG10', 'REG11', 'REG12', 'REG13', 'REG14', 'REG15', 'REG16',
     'REG17', 'REG18']
  boundary_condition:
    periodic
```

図 H.2.2　スラブセルモデルの入力例

表 H.2.2　セルモデル（HomoCell、SlabCell、RingCell）のセクション一覧

セクション	内容
name	セルモデルの名前
bounds	領域の境界位置 (cm) widths が指定されていれば入力不要である。
widths	領域の幅 (cm) bounds が指定されていれば入力不要である。
matnames	領域に対応する組成データの名前
boundary_condition	境界条件（isotropic/periodic/isolated）

H.2.3 炉心計算用メッシュの作成

炉心計算用の計算メッシュの情報は、RzMesh（２次元 RZ メッシュ）、XyzMesh（３次元 XYZ メッシュ、HexzMesh（３次元 Hex-Z メッシュ）、TrizMesh（３次元 Tri-Z メッシュ）オブジェクトを使って管理する。これらのオブジェクトを生成するための関数として、それぞれ、rzmesh()、xyzmesh()、hexzmesh()、trizmesh() が用意されている。

```
rzmesh(yaml=None, citation=None)
    ２次元 RZ メッシュ（RzMesh）オブジェクトを生成する。

    引数:
      yaml -- YAML 形式のテキストデータまたはファイルパス
      citation -- CITATION の入力ファイルのテキストデータまたはファイルパス

xyzmesh(yaml=None, citation=None, tritac=None)
    ３次元 XYZ メッシュ（XyzMesh）オブジェクトを生成する。

    引数:
      yaml -- YAML 形式のテキストデータまたはファイルパス
      citation -- CITATION の入力ファイルのテキストデータまたはファイルパス
      tritac -- TRITAC の入力ファイルのテキストデータまたはファイルパス

hexzmesh(yaml=None, citation=None, nshex=None)
    ３次元 Hex-Z メッシュ（HexzMesh）オブジェクトを生成する。

    引数:
      yaml -- YAML 形式のテキストデータまたはファイルパス
      citation -- CITATION の入力ファイルのテキストデータまたはファイルパス
      nshex -- NSHEX の入力ファイルのテキストデータまたはファイルパス

trizmesh(yaml=None, citation=None)
    ３次元 Tri-Z メッシュ（TrizMesh）オブジェクトを生成する。

    引数:
      yaml -- YAML 形式のテキストデータまたはファイルパス
      citation -- CITATION の入力ファイルのテキストデータまたはファイルパス
```

炉心計算用メッシュについても YAML 形式の入力ファイルで定義することが可能である。図 H.2.3 に YAML 形式での入力例を示す。各セクションの内容については表 H.2.3 に示した。

```
RzMesh:
  bounds:
    r:
      [0.0, 3.117, 9.351, 18.959, 30.698, 88.872, 119.949, 142.594, 159.448]
    z:
      [0.0, 20.396, 50.876, 76.276, 81.359, 91.516, 104.81]
  divs:
    r:
      [1, 1, 2, 2, 12, 6, 5, 3]
    z:
      [4, 6, 5, 1, 2, 2]
  matnames:
    ['SCF01', 'OTC02', 'RBL03', 'RBUHO', 'ABL04', 'ABU05', 'RDRHO', 'AXRHO', 'MTXHO']
  matmap:
    [[0, 0, 0, 0, 0, 1, 2, 6],
     [0, 0, 0, 0, 0, 1, 2, 6],
     [4, 4, 4, 4, 4, 4, 2, 6],
     [5, 5, 5, 5, 5, 5, 2, 6],
     [5, 5, 5, 5, 5, 5, 3, 6],
     [7, 7, 7, 7, 7, 7, 7, 8]]
  boundary_condition:
    rmin: 'reflection'
    rmax: 'vacuum'
    zmin: 'reflection'
    zmax: 'vacuum'
```

図 H.2.3　２次元 RZ 体系の計算メッシュ情報の入力例

表 H.2.3　炉心計算用計算メッシュの入力データのセクション

セクション	内容
name	セルモデルの名前
bounds	領域の境界位置 (cm) widths が指定されていれば入力不要である。 軸方向（r/x/y/z）毎にサブセクションを指定して入力する。
widths	領域の幅 (cm) bounds が指定されていれば入力不要である。 軸方向（r/x/y/z）毎にサブセクションを指定して入力する。
divs	領域の分割数 軸方向（r/x/y/z）毎にサブセクションを指定して入力する。
matnames	領域に対応する組成データの名前
matmap	各領域に対応する組成データ番号の割り当て 組成データ番号は matnames で定義した順番で最初を 0 番目と数える。 bounds や widths で定義した領域数に対応した配列でなければならない。
boundary_condition	境界条件（reflection/vacuum） RzMesh の場合、rmin は reflection でなければならない。

H.3 解析コードの実行

H.3.1 格子計算コードの実行

格子計算コードSLAROM-UFを実行するための関数としてslaromuf()が用意されている。前節で説明した関数で作成した組成データと格子計算モデルを入力としてSLAROM-UFコードを実行することができる。

```
slaromuf(cell, matset, peaco=False, peaco_cond=False, rrrp=False,
         library='UFLIB.J32-090423', egroup=70,
         outlist=None, debug=False, **options)
    SLAROM-UFコードを実行する。

    引数:
      cell -- セル（HomoCell, SlabCell, RingCell）オブジェクト
      matset -- 組成データMaterialSetオブジェクト
      peaco -- PEACOモジュールの利用オプション（True/False）
      peaco_cond -- PEACOモジュールとCONDモジュールの併用
                    オプション（True/False）
      rrrp -- RRRPモジュールの利用オプション（True/False）
      library -- 利用するUFLIBの名前
      egroup -- 利用するUFLIBのエネルギー群数
      outlist -- 出力ファイル（機番6）のファイルパス
      debug -- WORKファイルの保存ディレクトリ
               （指定したディレクトリに実行時の入出力ファイルを保存）
      **options -- SLAROM-UFコードのオプションの指定

    補足:
      SLAROM-UFコードのオプションの指定では、引数の名前を
      「モジュール名_変数名」で指定する。例えば、PATHモジュール
      のIBSWオプション（変数名）を1にしたい場合には、path_ibsw=1
      のように指定すればよい。

    返値:
      実効マクロ断面積（MacroscopicCrossSection）オブジェクトと
      実効ミクロ断面積（MicroscopicCrossSection）オブジェクトのタプル
```

H.3.2 実効断面積に関する処理の実行

JOINT-FR には実効断面積に関する処理を行うコードとして XMIX が含まれている。XMIX コードに相当する処理を行うための関数として xmix() が用意されている。

```
xmix(name,
     micro=None, material=None,
     macros=None, weights=None,
     macset=None, weightset=None)
    XMIX コードの処理をエミュレートする。

    XMIX コードの主な機能は以下の２つがある
      (1) ミクロ断面積と組成データからマクロ断面積を計算する。
      (2) 複数のマクロ断面積に重みをかけて平均化する。

    引数:
      (1) の場合は以下の引数が必要となる。
        name -- 計算結果のマクロ断面積（MacroscopicCrosssection）オ
                ブジェクトの名前
        micro -- ミクロ断面積（MicroscopicCrossSection）オブジェクト
        material -- 組成データ（Material）オブジェクト

      (2) の場合は以下の引数が必要となる。
        name -- 計算結果のマクロ断面積（MacroscopicCrosssection）オ
                ブジェクトの名前
        macros -- マクロ断面積（MacroscopicCrosssection）オブジェク
                  トのリスト
        weights -- macros の順番に対応した重みのリスト
        または、
        name -- 計算結果のマクロ断面積（MacroscopicCrosssection）オ
                ブジェクトの名前
        macset -- マクロ断面積セット（MacroscopicCrosssectionSet）オ
                  ブジェクト
        weightset -- macset の各マクロ断面積の名前に対応した重みのディ
                     クショナリ
```

ミクロ断面積と組成データからマクロ断面積を計算する機能の典型的な用途は、JUPITER 実験解析の AMM（All Master Model）補正である。AMM 補正では組成がわずかに異なる大量の燃料セルのマクロ断面積を求める必要があるが、組成の変化に伴う実効ミクロ断面積の変化が無視できると考えられるので、代表的な組成で一度だけ格子計算を行って実効ミクロ断面積を求めておけば、すべてのマクロ断面積を計算することができる。

複数のマクロ断面積に重みをかけて平均化する機能の典型的な用途はJUPITER実験解析で縮約計算用の2次元RZ体系のマクロ断面積を作成することである。ひとつの領域に2種類以上の燃料ドロワが分散して装荷されている場合に燃料ドロワの数を重みにしてマクロ断面積を平均化する。

H.3.3 群縮約計算の実行

断面積データの群縮約計算を実行するための関数としてcondense()が用意されている。

```
condense(xsec, boundary, weight, weight_transport=None)
    断面積の群縮約計算を行う。

    引数:
      xsec -- 実効マクロまたはミクロ断面積
              (Macroscopic/MicroscopicCrossSection) オブジェクト
      boundary -- 縮約後のエネルギー群構造の指定
                  (縮約前のエネルギー群の下限を指定する。)
      weight -- 縮約に用いる重み
      weight_transport -- 輸送断面積の縮約に用いる重み
                          (省略した場合はD φ 重みで縮約される。)

    補足:
      縮約後のエネルギー群構造の指定方法はJOINTコードの入力ファイル
      のカード5 (IA) と同じである。

      輸送断面積の縮約はJOINTコードとは異なり、特にオプションを指定
      しなくてもD φ 重みで縮約される。拡散係数Dには断面積オブジェク
      トに含まれている平均拡散係数 (average_diffusion_coefficient)
      を用い、中性子束 φ には引数weightで入力された値を用いる。
```

また、JFS-3の70群から18群への縮約、70群から7群への縮約については頻繁に行われるで、縮約後のエネルギー群構造の指定を省略できる関数として、condense18g()とcondense7g()が用意されている。

```
condense18g(xsec70g, weight, weight_transport=None)
    断面積の群縮約計算 (70群→18群) を行う。

    引数:
      xsec -- JFS-3の70群実効マクロまたはミクロ断面積
              (Macroscopic/MicroscopicCrossSection) オブジェクト
```

```
        weight -- 縮約に用いる重み
        weight_transport -- 輸送断面積の縮約に用いる重み
                            (省略した場合はD φ重みで縮約される。)

  condense7g(xsec70g, weight, weight_transport=None)
      断面積の群縮約計算（70 群→ 7 群）を行う。

      引数:
        xsec -- JFS-3 の 70 群実効マクロまたはミクロ断面積
                  (Macroscopic/MicroscopicCrossSection) オブジェクト
        weight -- 縮約に用いる重み
        weight_transport -- 輸送断面積の縮約に用いる重み
                            (省略した場合はD φ重みで縮約される。)
```

H.3.4 炉心計算コードの実行

炉心計算コードとしては、CITATION-FBR コードと TRITAC コードが利用可能である。

H.3.4.1 拡散計算コード CITATION-FBR の実行

CITATION-FBR コードを実行するための関数として citation() が用意されている。

```
citation(mesh, macset,
         representative_fission_spectrum,
         region_wise_fission_spectrum,
         r=None, x=None, y=None, z=None,
         black_absorber=None,
         adjoint=False, outlist=None, debug=False, **options)
    CITATION-FBR コードを実行する。

    引数 :
      mesh -- 計算メッシュ (RzMesh, XyzMesh, HexzMesh, TriZMesh) オブジェクト
      macset -- 実効マクロ断面積 (MacroscopicCrossSectionSet) オブジェクト
      representative_fission_spectrum -- 代表核分裂スペクトルの領域名
      region_wise_fission_spectrum -- 領域別核分裂スペクトル (True/False)
      r, x, y, z -- 各軸方向に適用する拡散係数の種類
                    (例 'average_diffusion_coefficient',
                        'perpendicular_diffusion_coefficient',
                        'parallel_diffusion_coefficient',)
      black_absorber -- 黒体吸収を適用する場合の領域番号
```

（計算メッシュオブジェクトの matmap の番号）
adjoint -- 随伴計算オプション（True/False）
outlist -- 出力ファイル（機番 51）のファイルパス
debug -- WORK ファイルの保存ディレクトリ
（指定したディレクトリに実行時の入出力ファイルを保存）
**options -- CITATION-FBR コードのオプションの指定

返値:
実効増倍率 (float) とメッシュ中性子束（MeshFlux）オブジェクト
のタプル

補足:
CITATION-FBR の入力形式の制限のため、領域別核分裂スペクトルを使う
場合であっても代表核分裂スペクトルの領域名を入力する必要がある。

なお、上記の CITATION-FBR コードの実行に限らず、炉心計算コードや摂動計算コードを実行する際には、マクロ断面積やミクロ断面積を入力データとして指定することになる。格子計算等でマクロ断面積やミクロ断面積を準備する段階では、領域毎の断面積データを扱うが、炉心計算や摂動計算では全領域の断面積データをまとめて渡す必要がある。

MARBLE ではマクロ断面積（MacroscopicCrossSection）オブジェクトとミクロ断面積（MicroscopicCrossSection）オブジェクトをまとめるために、それぞれ、マクロ断面積セット（MacroscopicCrossSectionSet）オブジェクトとミクロ断面積セット（MicroscopicCrossSectionSet）オブジェクトというオブジェクトがある。これらのオブジェクトを生成する関数として、それぞれ、macroset() と microset () が用意されている。

macroset(*macros)
マクロ断面積セット（MacroscopicCrossSectionSet）オブジェクトを生
成する。

引数:
macros -- マクロ断面積オブジェクト

microset(*micros)
ミクロ断面積セット（MicroscopicCrossSectionSet）オブジェクトを生
成する。

引数:
micros -- ミクロ断面積オブジェクト

典型的な使い方としては、図 H.3.1 に示すように slaromuf() 関数で作成した領域毎のマクロ断面積とミクロ断面積をまとめるときに使う。このようにしてまとめられたマクロ断面積セットは前述の citation() 関数の macset 引数に渡すことができる。

```
mac_core, mic_core = slaromuf(...)
mac_blanket, mic_blanket = slaromuf(...)
mac_reflector, mic_reflector = slaromuf(...)

macset = macroset(mac_core, mac_blanket, mac_blanket)
micset = microset(mic_core, mic_blanket, mic_blanket)
```

図 H.3.1　マクロ断面積セットの作成例

H.3.4.2　輸送計算コード TRITAC の実行

TRITAC コードを実行するための関数として tritac() が用意されている。

```
tritac(mesh, macset, sn,
       representative_fission_spectrum,
       pms=True, pt=False,
       adjoint=False,
       outlist=None, debug=False, **options)
    TRITAC コードを実行する。

    引数:
      mesh -- 計算メッシュ（XyzMesh）オブジェクト
      macset -- 実効マクロ断面積（MacroscopicCrossSectionSet）オブジェクト
      sn -- Sn 次数
      representative_fission_spectrum -- 代表核分裂スペクトルの領域名
      pms -- 正負項分離（PMS）版 TRITAC の利用（True/False）
      pt -- 摂動計算（PT）版 TRITAC の利用（True/False）
      adjoint -- 摂動計算オプション（True/False）
      outlist -- 出力ファイル（機番 51）のファイルパス
      debug -- WORK ファイルの保存ディレクトリ
               （指定したディレクトリに実行時の入出力ファイルを保存）
      **options -- TRITAC コードのオプションの指定

    返値:
      実効増倍率 (float) とメッシュ中性子束（MeshFlux）オブジェクト
      のタプル
```

H.3.4.3 輸送計算コード NSHEX の実行

NSHEX コードを実行するための関数として nshex() が用意されている。

```
nshex(mesh, macset, sn,
      representative_fission_spectrum,
      adjoint=False,
      outlist=None, debug=False, **options)
    NSHEX コード（全炉心版）を実行する。

    引数:
      mesh -- 計算メッシュ（HexzMesh）オブジェクト
      macset -- 実効マクロ断面積（MacroscopicCrossSectionSet）オブジェクト
      sn -- Sn 次数
      representative_fission_spectrum -- 代表核分裂スペクトルの領域名
      adjoint -- 随伴計算オプション（True/False）
      outlist -- 出力ファイル（機番 9）のファイルパス
      debug -- WORK ファイルの保存ディレクトリ
               （指定したディレクトリに実行時の入出力ファイルを保存）
      **options -- NSHEX コードのオプションの指定

    返値:
      実効増倍率 (float) とメッシュ中性子束（MeshFlux）オブジェクト
      のタプル
```

H.3.5 摂動計算コードの実行（反応度価値）

摂動計算コードとしては PERKY コードと SNPERT3D コードが利用可能である。JOINT-FR では PERKY コード、SNPERT3D コードはそれぞれ、CITATION-FBR コード、TRITAC コードと連携して利用するようになっている。SCHEME ではこの連携を自動化し 2 つのコードをひとつの関数の中で連続して実行する。

H.3.5.1 拡散摂動計算コードの実行（反応度価値）

拡散計算コード CITATION-FBR と拡散摂動計算コード PERKY を使って反応度価値を計算するための関数として citation_perky_for_reactivity() が用意されている。

```
  citation_perky_for_reactivity(
    mesh_before, mesh_after, macset,
    representative_fission_spectrum_before_for_citation,
```

```
representative_fission_spectrum_after_for_citation,
region_wise_fission_spectrum_for_citation,
representative_fission_spectrum_for_perky,
r=None, x=None, y=None, z=None,
black_absorber_before=None,
black_absorber_after=None,
first_order=False,
outlist_citation_before=None,
outlist_citation_after=None,
outlist_perky=None,
debug=False, **options)
  CITATION-FBR コードと PERKY コードを使って一次摂動または厳密摂動理
  論に基づき反応度を計算する。

  引数：
    mesh_before -- 非摂動体系の計算メッシュ（RzMesh, XyzMesh,
                   HexzMesh）オブジェクト
    mesh_after  -- 摂動体系の計算メッシュ（RzMesh, XyzMesh,
                   HexzMesh）オブジェクト
    macset -- 実効マクロ断面積セット（MacroscopicCrossSectionSet）
              オブジェクト
    representative_fission_spectrum_before_for_citation
      -- 非摂動体系の CITATION-FBR 実行時の代表核分裂スペクトルの領域名
    representative_fission_spectrum_after_for_citation
      -- 摂動体系の CITATION-FBR 実行時の代表核分裂スペクトルの領域名
    region_wise_fission_spectrum_for_citation
      -- CITATION-FBR 実行時の領域別核分裂スペクトル利用オプション
           （True/False）
    representative_fission_spectrum_for_perky
      -- PEKRY 実行時の代表核分裂スペクトルの領域名
    r, x, y, z -- 各軸方向に適用する拡散係数の種類
                  （例 'average_diffusion_coefficient',
                       'perpendicular_diffusion_coefficient',
                       'parallel_diffusion_coefficient',）
    black_absorber_before -- 黒体吸収を適用する場合の非摂動体系の
                             領域番号（計算メッシュオブジェクトの
                             matmap の番号）
    black_absorber_after -- 黒体吸収を適用する場合の摂動体系の領域
                            番号（計算メッシュオブジェクトの matmap
                            の番号）
    first_order -- 一次摂動適用オプション（True/False）
    outlist_citation_before -- 非摂動体系の CITATION の出力ファイル
```

```
                                    （機番 51）のファイルパス
      outlist_citation_after -- 摂動体系の CITATION の出力ファイル
                                    （機番 51）のファイルパス
      outlist_perky -- PEKRY の出力ファイル（機番 6）のファイルパス
      debug -- WORK ファイルの保存ディレクトリ
               （指定したディレクトリに実行時の入出力ファイルを保存）
      **options -- CITATION-FBR/PERKY コードのオプションの指定

    返値:
      非摂動体系の実効増倍率、摂動体系の実効増倍率、摂動計算で得られ
      た反応度のタプル

    補足:
      CITATION-FBR コード及び PERKY コードのオプションを指定する際には、
      どちらのコードのオプションであるかを明示するために、それぞれ、
      オプション名の前に「citation_」、「perky_」を付ける必要がある。
```

H.3.5.2　輸送摂動計算コードの実行（反応度価値）

輸送計算コード TRITAC と輸送摂動計算コード SNPERT3D を使って反応度価値を計算するための関数ととして tritac_snpert3d() が用意されている。

```
tritac_snpert3d(
  mesh_before, mesh_after, macset, sn,
  representative_fission_spectrum_before_for_tritac,
  representative_fission_spectrum_after_for_tritac,
  pms=True,
  first_order=False,
  outlist_tritac_before=None,
  outlist_tritac_after=None,
  outlist_snpert3d=None,
  debug=False,
  **options)
    TRITAC コードと SNPERT3D コードを使って一次摂動または厳密摂動理
    論に基づき反応度価値を計算する。

    Arguments:
      mesh_before -- 非摂動体系の計算メッシュ（XyzMesh）オブジェクト
      mesh_after  -- 摂動体系の計算メッシュ（XyzMesh）オブジェクト
      macset -- 実効マクロ断面積セット（MacroscopicCrossSectionSet）
```

```
            オブジェクト
      sn -- Sn 次数
      representative_fission_spectrum_before_for_tritac
        -- 摂動体系の TRITAC 実行時の代表核分裂スペクトルの領域名
      representative_fission_spectrum_after_for_tritac
        -- 摂動体系の TRITAC 実行時の代表核分裂スペクトルの領域名
      pms -- 正負項分離（PMS）版 TRITAC の利用（True/False）
      first_order -- 一次摂動適用オプション（True/False）
      outlist_tritac_before -- 非摂動体系の TRITAC の出力ファイル
                               （機番 6）のファイルパス
      outlist_tritac_after -- 非摂動体系の TRITAC の出力ファイル
                               （機番 6）のファイルパス
      outlist_snpert3d -- SNPERT3D の出力ファイル（機番 6）のファイルパス
      debug -- WORK ファイルの保存ディレクトリ
               （指定したディレクトリに実行時の入出力ファイルを保存）
      **options -- TRITAC/SNPERT3D コードのオプションの指定

    返値:
      非摂動体系の実効増倍率、摂動体系の実効増倍率、摂動計算で得られ
      た反応度価値のタプル

    補足:
      PERKY コードと異なり、SNPERT3D コードは核分裂スペクトルを自動的
      に読み込むため、代表核分裂スペクトルを指定する必要はない。

      TRITAC コード及び SNEPRT3D コードのオプションを指定する際には、
      どちらのコードのオプションであるかを明示するために、それぞれ、
      オプション名の前に「tritac_」、「snpert3d_」を付ける必要がある。
```

H.3.6 摂動計算コードの実行（実効遅発中性子割合）

拡散計算コード CITATION-FBR と拡散摂動計算コード PERKY を用いて実効遅発中性子割合の計算を行うための関数として、citation_perky_for_beta_effictive() が用意されている。

```
citation_perky_for_beta_effective(
  mesh, macset, micset, matset,
  representative_fission_spectrum_for_citation,
  region_wise_fission_spectrum_for_citation,
  representative_fission_spectrum_for_perky,
  nuclides, delayed_neutron_data,
```

```
r=None, x=None, y=None, z=None,
black_absorber=None,
outlist_citation=None,
outlist_perky=None,
debug=False, **options)
  CITATION-FBR コードと PERKY コードを使って実効遅発中性子割合を計算
  する。

  引数:
    mesh -- 計算メッシュ（RzMesh, XyzMesh,HexzMesh）オブジェクト
    macset -- 実効マクロ断面積セット（MacroscopicCrossSectionSet）
              オブジェクト
    micset -- 実効ミクロ断面積セット（MicroscopicCrossSectionSet）
              オブジェクト
    matset -- 組成データセット（MaterialSet）オブジェクト
    representative_fission_spectrum_for_citation
      -- CITATION-FBR 実行時の代表核分裂スペクトルの領域名
    region_wise_fission_spectrum_for_citation
      -- CITATION-FBR 実行時の領域別核分裂スペクトル利用オプション
          （True/False）
    representative_fission_spectrum_for_perky
      -- PEKRY 実行時の代表核分裂スペクトルの領域名
    nuclides -- 実効遅発中性子割合の計算で考慮する核種名のリスト
                （PERKY コードの入力データの SECTION 005 に対応する。）
    delayed_neutron_data -- 遅発中性子データ（DelayedNeutronData）
                            オブジェクト
    r, x, y, z -- 各軸方向に適用する拡散係数の種類
                  （例 'average_diffusion_coefficient',
                       'perpendicular_diffusion_coefficient',
                       'parallel_diffusion_coefficient',）
    black_absorber -- 黒体吸収を適用する場合の 領域番号
                      （計算メッシュオブジェクトの matmap の番号）
    outlist_citation -- CITATION の出力ファイル（機番 51）のファイルパス
    outlist_perky -- PEKRY の出力ファイル（機番 6）のファイルパス
    debug -- WORK ファイルの保存ディレクトリ
             （指定したディレクトリに実行時の入出力ファイルを保存）
    **options -- CITATION-FBR/PERKY コードのオプションの指定

  返値:
    実効増倍率と実効遅発中性子割合のタプル

  補足:
```

CITATION-FBR コード及び PERKY コードのオプションを指定する際には、
どちらのコードのオプションであるかを明示するために、それぞれ、
オプション名の前に「citation_」、「perky_」を付ける必要がある。

実効遅発中性子割合の計算で入力として必要になる遅発中性子データは、MARBLE では遅発中性子データオブジェクト（DelayedNeutronData）で管理されている。このオブジェクトを生成するための関数として delayed_neutron_data() が用意されている。

```
delayed_neutron_data(version=None,
                     nuclides=None,
                     yield_=None,
                     spectrum=None)
    遅発中性子データ（DelayedNeutronData）オブジェクトを生成する。

    データの作成方法は以下の２通りがある
      (1) MARBLE に登録されているデータを利用する。
      (2) ユーザが直接データを入力する。

    引数:
       (1) の場合は以下の引数が必要となる。
         version -- データのバージョン名
                    （データは NuclideProperty クラスで定義されており、
                     例えば、'JENDL-4.0' 等が指定できる。）
       (2) の場合は以下の引数が必要となる。
         nuclides -- 核種名のリスト
                     （PERKY の入力データ MNCC に対応する。）
         yield_ -- 遅発中性子収率
                   （PERKY の入力データ DNUF に対応する。）
         spectrum -- 遅発中性子スペクトル
                     （PERKY の入力データ DXKAI に対応する。）

    補足:
       yield_ と spectrum は（核種数，ファミリー数，エネルギー群数）の
       ３次元配列でなければならない。
```

H.3.7　摂動計算コードの実行（即発中性子寿命）

拡散計算コードCITATION-FBRと拡散摂動計算コードPERKYを用いて即発中性子寿命の計算を行うための関数として、citation_perky_for_prompt_neutron_lifetime() が用意されている。

```
citation_perky_for_prompt_neutron_lifetime(
  mesh, macset,
  representative_fission_spectrum_for_citation,
  region_wise_fission_spectrum_for_citation,
  representative_fission_spectrum_for_perky,
  lethargy,
  r=None, x=None, y=None, z=None,
  black_absorber=None,
  outlist_citation=None,
  outlist_perky=None,
  debug=False, **options)
    CITATION-FBR コードと PERKY コードを使って即発中性子寿命を計算する。

    引数:
      mesh -- 計算メッシュ（RzMesh, XyzMesh,HexzMesh）オブジェクト
      macset -- 実効マクロ断面積セット（MacroscopicCrossSectionSet）
                オブジェクト
      representative_fission_spectrum_for_citation
        -- CITATION-FBR 実行時の代表核分裂スペクトルの領域名
      region_wise_fission_spectrum_for_citation
        -- CITATION-FBR 実行時の領域別核分裂スペクトル利用オプション
           （True/False）
      representative_fission_spectrum_for_perky
        -- PEKRY 実行時の代表核分裂スペクトルの領域名
      lethargy -- 計算に利用するエネルギー群構造の各群の低エネルギー
                  側境界に対応したレサジー
      r, x, y, z -- 各軸方向に適用する拡散係数の種類
                    （例 'average_diffusion_coefficient',
                         'perpendicular_diffusion_coefficient',
                         'parallel_diffusion_coefficient',）
      black_absorber -- 黒体吸収を適用する場合の 領域番号
                        （計算メッシュオブジェクトの matmap の番号）
      outlist_citation -- CITATION の出力ファイル（機番 51）のファイルパス
      outlist_perky -- PEKRY の出力ファイル（機番 6）のファイルパス
      debug -- WORK ファイルの保存ディレクトリ
               （指定したディレクトリに実行時の入出力ファイルを保存）
      **options -- CITATION-FBR/PERKY コードのオプションの指定
```

```
    Return:
      実効増倍率と即発中性子寿命のタプル

    補足:
      CITATION-FBR コード及び PERKY コードのオプションを指定する際には、
      どちらのコードのオプションであるかを明示するために、それぞれ、
      オプション名の前に「citation_」、「perky_」 を付ける必要がある。
```

即発中性子寿命の計算ではレサジーを入力する必要がある。このレサジーの値をユーザが計算して入力してもよいが、SCHEME ではユーティリティ関数として lethargy() が用意されている。標準的な JFS-3 の 70 群、18 群、7 群で計算している場合、citation_perky_for_prompt_neutron_lifetime() の引数 lethargy に渡す値は、それぞれ、lethargy(jfs70g())、lethargy(jfs18g())、lethargy(jfs7g()) とすればよい。

```
lethargy(structure)
    各エネルギー群の低エネルギー側境界でのレサジーを返す。

    引数:
      structure -- エネルギー群境界エネルギー（eV）のリストまたは、
                   エネルギー群構造（EnergyGroupStructure）オブジェ
                   クト

jfs70g()
    JFS-3 の 70 群構造のエネルギー群構造（EnergyGroupStructure）オブジェ
    クトを生成する。

jfs18g()
    JFS-3 の 18 群構造のエネルギー群構造（EnergyGroupStructure）オブジェ
    クトを生成する。

jfs7g()
    JFS-3 の 7 群構造のエネルギー群構造（EnergyGroupStructure）オブジェ
    クトを生成する。
```

H.3.8 摂動計算コードの実行（反応度価値空間分布）

拡散計算コードCITATION-FBRと拡散摂動計算コードPERKYを用いて反応度価値空間分布の計算を行うための関数として、citation_perky_for_worth_mapping() が用意されている。

```
citation_perky_for_worth_mapping(
  mesh, macset, micset, matset,
  representative_fission_spectrum_for_citation,
  region_wise_fission_spectrum_for_citation,
  representative_fission_spectrum_for_perky,
  r=None, x=None, y=None, z=None,
  black_absorber=None,
  outlist_citation=None,
  outlist_perky=None,
  debug=False, **options)
    CITATION-FBR と PERKY を使って反応度価値空間分布（反応度マップ）を
    計算する。

    引数:
      mesh -- 計算メッシュ（RzMesh, XyzMesh, HexzMesh）オブジェクト
      macset -- 実効マクロ断面積セット（MacroscopicCrossSectionSet）
                オブジェクト
      micset -- 実効ミクロ断面積セット（MicroscopicCrossSectionSet）
                オブジェクト
      matset -- 組成データセット（MaterialSet）オブジェクト
      representative_fission_spectrum_for_citation
        -- CITATION-FBR 実行時の代表核分裂スペクトルの領域名
      region_wise_fission_spectrum_for_citation
        -- CITATION-FBR 実行時の領域別核分裂スペクトル利用オプション
            （True/False）
      representative_fission_spectrum_for_perky
        -- PEKRY 実行時の代表核分裂スペクトルの領域名
      r, x, y, z -- 各軸方向に適用する拡散係数の種類
                    （例 'average_diffusion_coefficient',
                        'perpendicular_diffusion_coefficient',
                        'parallel_diffusion_coefficient',）
      black_absorber -- 黒体吸収を適用する場合の 領域番号
                        （計算メッシュオブジェクトの matmap の番号）
      perky_nucl1 -- 反応度を計算する核種の JFS-ID の指定
                     （PERKY 入力ファイルの SECTION 007 の NUCL1 に対応）
      perky_nucla -- 核分を起こす核種の JFS-ID の指定
                     （PERKY 入力ファイルの SECTION 007 の NUCLA に対応）
```

perky_npln -- 反応度を計算するプレーンの数
（PERKY 入力ファイルの SECTION 007 の NPLN に対応）
perky_mlft -- 反応度を計算する左端のメッシュ点
（PERKY 入力ファイルの SECTION 007 の MLFT に対応）
perky_mrit -- 反応度を計算する右端のメッシュ点
（PERKY 入力ファイルの SECTION 007 の MRIT に対応）
perky_mtop -- 反応度を計算する上端のメッシュ点
（PERKY 入力ファイルの SECTION 007 の MTOP に対応）
perky_mbot -- 反応度を計算する下端のメッシュ点
（PERKY 入力ファイルの SECTION 007 の MBOT に対応）
outlist_citation -- CITATION の出力ファイル（機番 51）のファイルパス
outlist_perky -- PEKRY の出力ファイル（機番 6）のファイルパス
debug -- WORK ファイルの保存ディレクトリ
（指定したディレクトリに実行時の入出力ファイルを保存）
**options -- CITATION-FBR/PERKY コードのオプションの指定

返値:
実効増倍率と反応度マップのタプル

補足:
CITATION-FBR コード及び PERKY コードのオプションを指定する際には、
どちらのコードのオプションであるかを明示するために、それぞれ、
オプション名の前に「citation_」、「perky_」を付ける必要がある。

H.4　データの保存と復元

計算の途中でデータをファイルに保存したり、ファイル保存されているデータを復元して利用したりする場合の関数として、それぞれ、dump_to_file() と load_from_file() が用意されている。

典型的な利用方法としては、繰り返し利用する可能性があり、作成にも時間のかかる実効マクロ断面積セットオブジェクトのようなデータを保存しておき、炉心計算や摂動計算を実行する際に復元して使うといった方法がある。

```
dump_to_file(obj, path)
    オブジェクトのデータをファイルに保存する。

    引数:
      obj -- MARBLE オブジェクト
      path -- 保存するファイルのパス

load_from_file(path)
    ファイルに保存されたオブジェクトを復元する。

    Arguments:
      path -- オブジェクトが保存されたファイルのパス
```

また、保存したファイルやディレクトリに対して移動や削除といった処理をしたい場合に利用する関数として mv()、rm()、mkdir()、rmdir() が用意されている。

```
mv(old_path, new_path)
    ファイルを移動する（ファイル名を変更する)。

    引数:
      old_path -- 移動元のファイルのパス
      new_path -- 移動先のファイルのパス

rm(path)
    ファイルを削除する。

    引数:
      path -- 削除するファイルのパス

mkdir(path)
    ディレクトリを作成する。
```

```
    引数:
      path -- 作成するディレクトリのパス

rmdir(path, force=False)
    ディレクトリを削除する。

    引数:
      path -- 削除するディレクトリのパス
      force -- ディレクトリの中身があっても強制的に削除するためのオ
               プション（True/False）
```

付録I　ORPHEUSユーザマニュアル

次世代炉心解析システム MARBLE に含まれる高速炉実機燃焼解析システム ORPHEUS の利用方法について説明する。

I.1　ORPHEUSの入力ファイル

高速炉実機燃焼解析システム ORPHEUS は MARBLE を利用して開発された解析システムであるが、解析を実施するためにユーザがプログラムを書いて MARBLE のオブジェクトを直接操作する必要がないようにシステム化されている。すなわち、ユーザは従来型の解析システムと同様に入力ファイルを準備して実行するというユーザインターフェースを採用している。

ORPHEUS を動作させるためには以下の５種類のテキストファイルを入力として与える必要がある。なお、計算モードよっては、ORPHEUS データベースファイルとミクロ断面積ファイルを指定する必要がある。

- 計算条件ファイル（ユーザ入力ファイル）
- 物質組成ファイル
- 幾何形状ファイル
- 装荷パターンファイル
- 炉心特性ファイル

これらの入力ファイルは YAML 形式で記述され、キーワードとそれに対応する値が階層構造で定義される。以下では、この階層構造のトップにあるキーワードとそのキーワードで与えられる値全体をセクションと呼ぶことにする。本節では、これらの入力ファイルで定義されたセクションに関して特に重要な点に絞って説明する。

I.2　計算条件ファイル（ユーザ入力ファイル）

計算条件ファイルでは、計算の対象となる炉心やサイクルの指定、各種の詳細な計算条件等、計算ケースに固有な情報を与える。また、計算条件ファイル以外の入力ファイルの指定も行うため、計算条件ファイルはユーザが実行したい計算内容を定義するための根幹となるファイル（ユーザ入力ファイル）と考えることができる。本ファイルのセクションの一覧は表 I.2.1 のとおりである。

以下では各セクションについて入力内容を説明する。

表 I.2.1　計算条件ファイルのセクション一覧

セクション	内容
calc_mode	システムの計算モードを指定する。
file	入出力ファイルを指定する。
title	計算タイトルを指定する。
core_name	炉心名を指定する。
case_name	計算ケース名を指定する。
cycle	サイクルに関する情報を指定する。 サイクル名、積算サイクル数、起動・停止日等。
calc_system	計算体系を指定する。
solver	使用する計算コードとコードに与えるオプションを指定する。
step	燃焼ステップ毎の種々の条件を指定する。
condensed_group	群縮約計算の条件を指定する（群縮約計算モードのときのみ必要)。
raytrace	レイトレース処理実行時のオプションを指定する。

I.2.1　calc_mode セクション

ORPHEUS では解析対象に応じて解析フローを変更するために計算モードを指定する。calc_mode セクションの入力例を図 I.2.1 に示す。calc_mode セクションに与えることができる値の一覧を表 I.2.2 に示す。

```
calc_mode:
  standard
```

図 I.2.1　計算条件ファイルの calc_mode セクションの入力例

表 I.2.2　calc_mode セクションの値一覧

値	意味
standard	標準解析モード
constant_micro	ユーザ指定のミクロ断面積を用いた解析モード
collapse	群縮約計算モード（ミクロ断面積更新なし）
collapse_micro_update	群縮約計算モード（ミクロ断面積更新を含む）
control_rod	制御棒価値計算モード

I.2.2　file セクション

file セクションの入力例を図 I.2.2 に示す。図に示すようにシステムへの入力として与えるファイルを input キーワード、システムが出力するファイルを output キーワードの階層下に指定する。input キーワード以下に指定するファイルについては表 I.2.3 に示した。前述の 5 つの入力ファイルのうちユーザ入力ファイルを除く 4 つのファイルは必ず指定しなければならない。

```
file :
  input:
    core_property: $PWD/monju.coreproperty
    pattern      : $PWD/monju_1cyc.pattern
    geometry     : $PWD/monju.geometry
    material     : $PWD/monju.material
  output:
    raytrace     : $PWD/output/monju_70g.out.raytrace
    database     : $PWD/output/monju_70g.out.database
    summary      : $PWD/output/monju_70g.out.summary
    list         : $PWD/output/monju_70g.out.list
    debug        : $PWD/output/monju_70g.out.debug
```

図 I.2.2　計算条件ファイルの file セクションの入力例

表 I.2.3　計算条件ファイルの input キーワード以下に指定するファイル

キーワード	ファイルの内容
core_property	炉心特性ファイル（必須）
pattern	装荷パターンファイル（必須）
geometry	幾何形状ファイル（必須）
material	物質組成ファイル（必須）
raytrace	レイトレースファイル（オプション）
database	ORPHEUS データベース（colapse、control_rod 計算モード時に必要）
cross_section	断面積情報ファイル（constant_micro 計算モード時に必要）

output キーワード以下に指定するファイルについては表 I.2.4 に示した。ORPHEUS データベースはリスタートファイルとしての機能も持っており、リスタート計算を行う場合は、input キーワード以下に入力として使用する ORPHEUS データベースファイルを必要な数だけサイクル番号をキーワードにして指定する。

ORPHEUS では計算メッシュの初期物質組成を得るため、集合体の核物質領域の計算メッシュに対するウェイト（面積）を集合体詳細幾何形状に対するレイトレース処理を行って算出する。レイトレースは比較的時間を要する処理であるが、集合体の詳細幾何形状の種類は限られているた

表 I.2.4　計算条件ファイルの output キーワード以下に指定するファイル

キーワード	ファイルの内容
database	ORPHEUS データベース
raytrace	レイトレースファイル
summary	計算結果を要約した出力ファイル（現在使用されていない）
list	計算結果出力ファイル
debug	デバッグ用出力ファイル

め、一度実施したレイトレース処理の結果を保存し２回目以降は保存された結果を使用するようにすれば処理時間を短縮できる。

なお、図 I.2.2 に示されているように、「$」記号を使うことで環境変数（この例では$PWD）を用いることもできる。また、相対パスによる指定も可能である。

I.2.3　title セクション

計算タイトルを指定する。ユーザによる情報の管理に利用されることを想定して準備されているセクションであるが、現在の実装では使用されていない。

I.2.4　core_name セクション

炉心名を指定する。入力のエラーチェックのために使用することを想定して準備されているセクションであるが、現在の実装では使用されていない。

I.2.5　case_name セクション

解析ケース名を指定する。ユーザによる情報の管理に利用されることを想定して準備されているセクションであるが、現在の実装では使用されていない。

I.2.6　cycle セクション

cycle セクションの入力例を図 I.2.3 に示す。解析対象運転サイクルの名前（name）と積算サイクル数（number）を指定する。

```
cycle:
  name: The 2nd dash cycle
  number: 3
```

図 I.2.3　計算条件ファイルの cycle セクションの入力例

I.2.7 calc_system セクション

calc_system セクションの入力例を図 I.2.4 に示す。炉心計算（core）及び燃焼計算（burnup）のそれぞれについて、coordinates キーワード以下に計算メッシュ体系を指定し、axial_mesh キーワード以下に軸方向のメッシュ長を指定する。計算メッシュの体系には、hexz、triz、xyz のいずれかを指定する[1]。軸方向メッシュ長は、CITATION-FBR コードの入力と同様に、分割数と長さのペアを必要な数だけ指定する。例えば、[12, 60.0] という指定は 60.0cm を 12 等分することを表す。

なお、ここで指定されたペアの数は後述する装荷パターンファイルのプレーンに相当する。炉心計算と燃焼計算のプレーンは同じ形状にしなければならない。

```
calc_system:
  coordinates:
    core   : triz
    burnup: hexz

  axial_mesh:
    core   : [[10,   60.0],    ## reflector
              [12,   60.0],    ## core
              [10,   60.0]]    ## reflector

    burnup: [[1,    60.0],    ## reflector
              [12,   60.0],    ## core
              [1,    60.0]]    ## reflector
```

図 I.2.4　計算条件ファイルの calc_system セクションの入力例

[1]計算コードによって使用できる計算メッシュ体系が異なる。例えば、CITATION-FBR コードでは hexz、triz、xyz が指定できるが、TRITAC コードでは xyz しか指定できない。

I.2.8　solver セクション

solver セクションの入力例を図 I.2.5 に示す。格子計算（xs）、炉心計算（core）、燃焼計算（burnup）のそれぞれについて、使用する計算コードの名称と計算コードに固有の入力オプションを指定する。この例では、炉心計算コードとして CITATION-FBR、燃焼計算コードとして BURNUP を使用する。オプション指定が省略された場合はシステムが適当なデフォルト値を与える。

表 I.2.5 に solver セクションに指定可能なソルバーの一覧を示す。格子計算コードとしては、現状 SLAROM-UF のみ利用可能である。炉心計算については、CITATION-FBR（Hex-Z 体系、Tri-Z 体系、XYZ 体系拡散）、TRITAC（XYZ 体系輸送）、NSHEX （Hex-Z 体系輸送）が利用可能である。CITATION-FBR と TRITAC については入力フォーマットの制限から領域数が制限されているため、ORPHEUS の燃焼計算では領域数が足りなくなるという問題が発生することがある。CITATION-FBR、TRITAC の入力フォーマットを変更すると JOINT-FR が動作しなくなるため、MARBLE では MARBLE 専用に拡張した MARBLE 版 CITATION-FBR、MARBLE 版 TRITAC を利用できるようにしている。なお、MARBLE 版 TRITAC では断面積入力フォーマットをバイナリ化してデータ入力の高速化も行われている。通常は MARBLE 版を使えばよいが、JOINT-FR との厳密な比較を行いたい場合などのために JOINT-FR 版も利用できるようになっている。

表 I.2.5　solver セクションに指定可能なソルバー一覧

キーワード	ソルバー名	特徴
xs	slarom_uf	SLAROM-UF コード
core	citation	JOINT-FR 標準の CITATION-FBR
	marble_citation	領域数等を拡張した MARBLE 用 CITATION-FBR
	tritac	JOINT-FR 標準の TRITAC（通常版）
	tritac_pms	JOINT-FR 標準の TRITAC（正負項分離版）
	marble_tritac	領域数等を拡張した MARBLE 用 TRITAC（標準版）
	marble_tritac_pms	領域数等を拡張した MARBLE 用 TRITAC（正負項分離版）
	nshex	NSHEX コード
burnup	burnup	MARBLE 組み込みの BURNUP ソルバー

I.2.8.1　炉心計算のオプション

CITATION-FBR と TRITAC では、代表核分裂スペクトルの領域を指定する必要があるが、fs_zone キーワードで、後述する装荷パターンファイルで指定されたゾーン番号を指定する。また、CITATION-FBR では region_wise_fission_spectrum キーワードに True を指定することで領域依存の核分裂スペクトルを使った計算を行うことができる。

I.2.8.2　燃焼計算のオプション

燃焼計算ソルバー BURNUP については、chain キーワードと decay キーワードでそれぞれ燃焼チェーンと崩壊定数のデータセットを指定する。現状では「standard2002（図 I.2.6 参照）」と「standard2006（図 I.2.7 参照）」が利用可能である。

```
solver :
  xs:
    name   : slarom_uf
    library : UFLIB.J32−090423
    ng      : 70
    options:
      ibsw : 2   ## buckling search
      itpe : 0   ## iteration of background cross section  (0/1=No/Yes)
      te   : 473.15

  core:
    name   : marble_citation
    options:
      fs_zone : 1
      region_wise_fission_spectrum :  True
      ngc  : [0,  0,  0,  0,  0,  1,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
      iedg : [1,  0,  0,  1,  1,  1,  1,  1,  1,  0,  0,  0,  0,  0,  0,  0,  0]
      itmx : [900,  900]
      isodf : 1
      ixdct : 1   ## average_diffusion_coefficient
      iydct : 1   ## average_diffusion_coefficient
      izdct : 1   ## average_diffusion_coefficient
      ipunf : 7

  burnup:
    name: burnup
    options:
      chain: standard2006
      decay: standard2006
```

図 I.2.5　計算条件ファイルの solver セクションの入力例

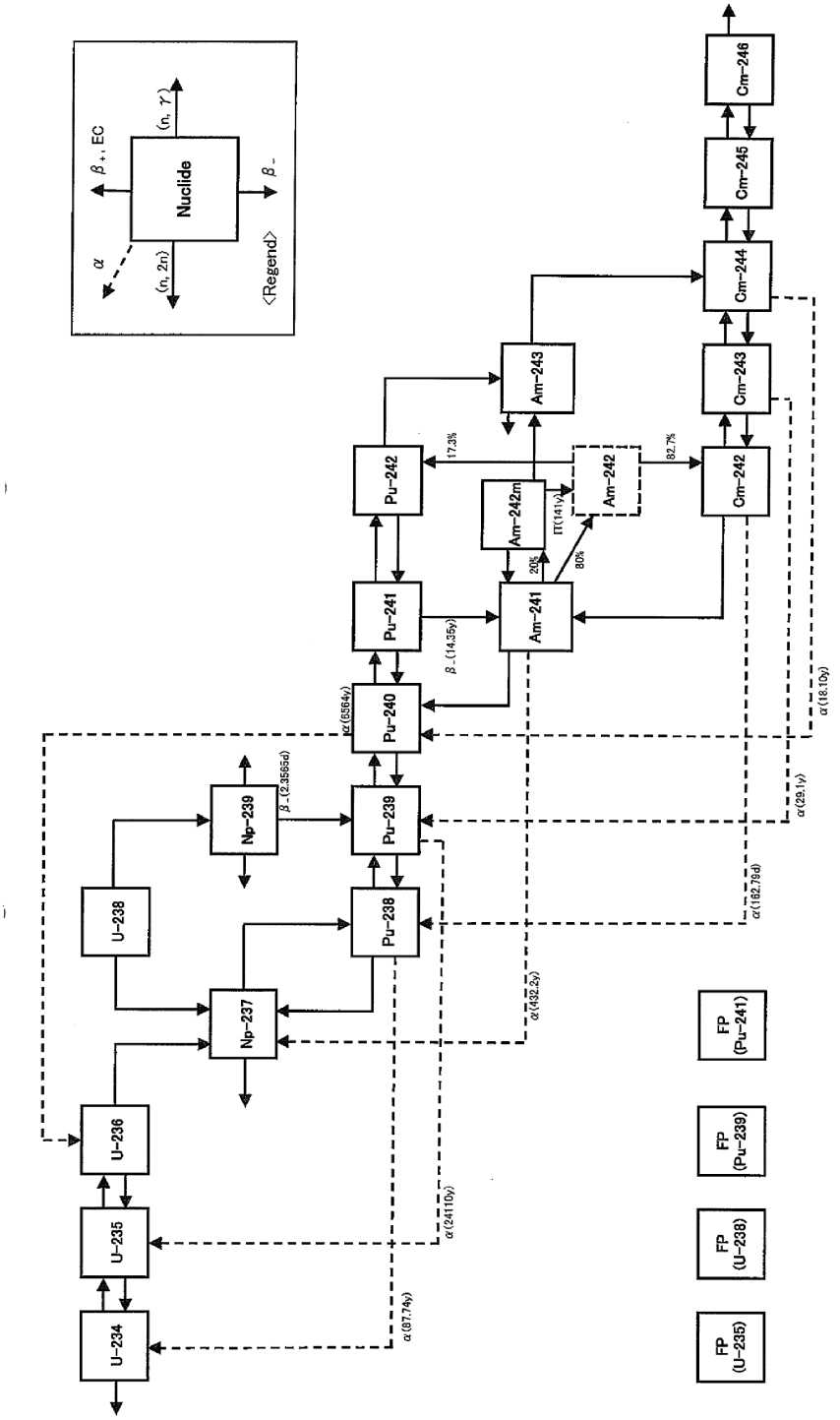

図 I.2.6　燃焼チェーン（standard2002）

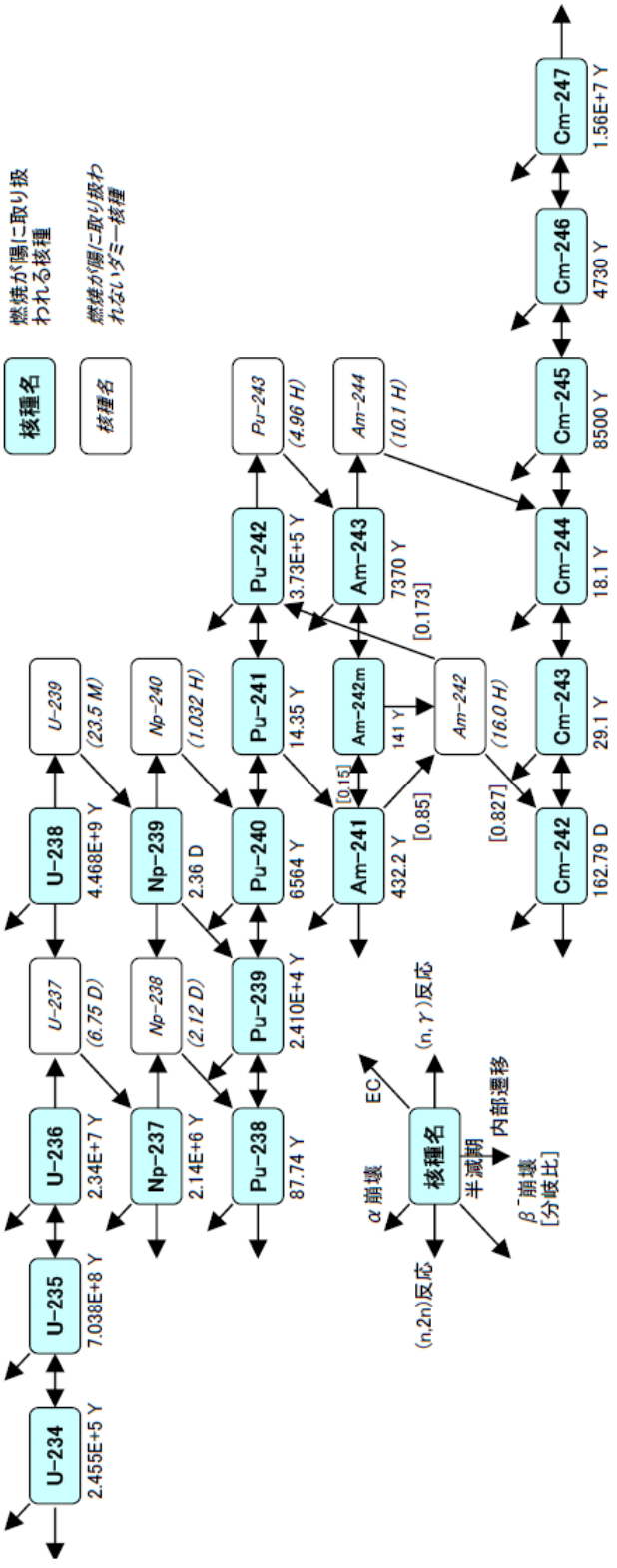

図 I.2.7　燃焼チェーン（standard2006）

I.2.9　step セクション

step セクションの入力例を図 I.2.8 に示す。ここでは燃焼ステップ毎に period キーワードと power キーワードを使ってそれぞれ、燃焼期間（日数）と相対出力（%）を指定する。燃焼を行うステップ数分だけ指定する。

また、control_rod キーワードを使うことで各燃焼計算ステップにおける制御棒位置を指定することができる。制御棒集合体位置として「all」を指定した場合、すべての制御棒を表す。集合体位置アドレス（1B1 等）を指定した場合は当該制御棒のみを表す。バンクグループ名（後述の炉心特性ファイルで定義する）を指定した場合は、当該バンクグループに含まれるすべての制御棒を表す。

なお、実効ミクロ断面積を燃焼ステップ毎に更新する計算モード（standard、collapse_micro_update）では、特に何も指定しない場合、燃焼ステップの最初に格子計算を実行して実効ミクロ断面積を更新する。更新しないようにする場合は、xs_update キーワードに False を指定する。

```
step:
  - period:  25.00  ## day
    power : 100.00  ## %
    control_rod:
      all :  120.0

  - period:  25.00
    power : 100.00
    control_rod:
      all :  120.0
    xs_update: False

  - period:  30.00
    power :   0.00
    control_rod:
      all :  120.0
    xs_update: False
```

図 I.2.8　計算条件ファイルの step セクションの入力例

I.2.10　condensed_group セクション

このセクションは、計算モード（calc_mode）として「collapse」または「collapse_micro_update」が指定されたときに必要となる。図 I.2.9 にミクロ断面積の更新がない群縮約計算の場合（calc_mode: collapse）の例を示す。

```
file :
  input:
    database :
      0: $MARBLE_TEST_DATA_PATH/orpheus/minimal/0cyc.out.odb
      1: $MARBLE_TEST_DATA_PATH/orpheus/minimal/1cyc.out.odb
      2: $MARBLE_TEST_DATA_PATH/orpheus/minimal/2cyc.out.odb

solver :
  xs:
    ng: 7

condensed_group:
  boundary: [4, 8, 19, 28, 37, 46 ,70]
  target:
    cycle: 2
    step: 5
```

図 I.2.9　ミクロ断面積の更新がない場合の群縮約計算入力例

この図に示すように condensed_group セクションの boundary キーワードで縮約する際の少数群の構造を指定する。この例では 70 群の詳細群構造を 1〜4 ／ 5〜8 ／ 9〜19 ／ 20〜28 ／ 29〜37 ／ 38〜46 ／ 47〜70 の 7 群に縮約している[2]。縮約計算に用いる中性子スペクトルとミクロ断面積は target キーワードで指定する。この例ではサイクル 2 の計算ステップ 5 の計算結果を使用している。縮約計算に用いる詳細群の中性子束や断面積の計算結果は ORPHEUS データベースから取り出すようになっており、利用する ORPHEUS データベースを file セクションの database キーワードで指定する。縮約後の少数群の群数は solver セクションの ng キーワードで指定する。

次にミクロ断面積の更新がある場合（calc_mode: collapse_micro_update）の例を図 I.2.10 示す。ミクロ断面積の更新がある場合は、各燃焼ステップで詳細群の中性子スペクトルを求めてから縮約計算を行うので、どの時点の中性子スペクトルとミクロ断面積を使うかを指定する必要はない。したがって、例のように condensed_group セクションの boundary キーワードのみ指定すればよい。

[2]JOINT コードの入力方式と同一である。

```
condensed_group:
  boundary: [4, 8, 19, 28, 37, 46 ,70]
```

図 I.2.10　ミクロ断面積の更新がある場合の群縮約計算入力例

I.2.11　raytrace セクション

raytrace セクションの入力例を図 I.2.11 に示す。raytrace セクションではレイトレース処理の際の Gauss 積分の積分点の個数（order）を指定する。order キーワードに指定できる最大値は 30 である。

```
raytrace:
  order: 30
```

図 I.2.11　計算条件ファイルの raytrace セクションの入力例

I.3 物質組成ファイル

図 I.3.1 に物質組成ファイルの入力例を示す。物質組成ファイルのセクションは material のみである。定義する組成データの数だけ配列の形で与える。各組成データは name キーワードで名前を定義し、type キーワードで物質の種類を指定する。composition キーワードで核種毎の原子数密度データ（10^{24} cm^{-3}）を入力する。なお、燃焼させる燃料データについては、燃焼チェーンで利用する核種にあわせて原子数密度データを指定しておかなければならない。

表 I.3.1 に material セクションに入力する type 一覧を示す。この type の情報は格子計算モデルを作成する際に利用されるので正しく入力する必要がある。

表 I.3.1 type キーワードの値一覧

値	意味
fuel	燃料
sus	構造材
coolant	冷却材
absorber	吸収材
other	上記の物質の混合

```
material:
  − name: fuel_pellet_material
    type: fuel
    composition:
      U−234:     0.00000e+00
      U−235:     1.68531e−03
      U−236:     0.00000e+00
      U−238:     5.57394e−03
      Np−237:    0.00000e+00
      Np−239:    0.00000e+00
      Pu−238:    1.42106e−05
      Pu−239:    1.08722e−02
      Pu−240:    2.70988e−03
      Pu−241:    3.73374e−04
      Pu−242:    7.68636e−05
      Am−241:    8.97256e−05
      Am−242m: 0.00000e+00
      Am−243:    0.00000e+00
      Cm−242:    0.00000e+00
      Cm−243:    0.00000e+00
      Cm−244:    0.00000e+00
      Cm−245:    0.00000e+00
      Cm−246:    0.00000e+00
      Cm−247:    0.00000e+00
      O−16:      4.28603e−02
      U−235FP: 0.00000e+00
      U−238FP: 0.00000e+00
      Pu−239FP: 0.00000e+00
      Pu−241FP: 0.00000e+00

  − name: b4c_pellet_material
    type: absorber
    composition:
      B−10: 8.868160e−02
      B−11: 8.922750e−03
      C−12: 2.467800e−02

  − name: sodium_material
    type: coolant
    composition:
      Na−23: 2.295440e−02

  − name: sus_material
    type: sus
    composition:
      Cr−nat.:  1.531820e−02
      Mn−55:    1.261930e−03
      Fe−nat.:  5.945620e−02
      Ni−nat.:  7.551710e−03
      Mo−nat.: 1.289120e−04
```

図 I.3.1　物質組成ファイルの入力例

I.4　幾何形状ファイル

幾何形状ファイルでは、集合体の詳細な幾何形状の定義を行う。本ファイルのセクションの一覧は表 I.4.1 のとおりである。

表 I.4.1　幾何形状ファイルのセクション一覧

セクション	内容
primitive	六角形・円等の基礎的な幾何形状を定義する。
pin	燃料ピンの幾何形状等、primitive セクションにおいて定義された形状の 組み合わせによる単純形状を定義する。
lattice	燃料ピンの格子配列等、pin セクションにおいて定義された単純形状の適当な配置を定義する。
segment	lattice、primitive セクションで定義された形状を用いて、集合体の２次元断面の幾何形状を定義する。
assembly	segment セクションにおいて定義された形状を Z 軸方向に積み重ねて、３次元集合体詳細幾何形状を定義する。

I.4.1　primitive セクション

primitive セクションの入力例を図 I.4.1 に示す。ORPHEUS では六角形・円等の基礎的な形状を組み合わせることで複雑な集合体詳細幾何形状を構成する。ここではもっとも基礎となる幾何形状の定義をおこなう。type、angle、value キーワードでそれぞれ、形状、回転角、大きさを定義する。ここで与えるベースとなる形状の定義を図 I.4.2 に示した。name キーワードで、定義された形状に対して固有の名称が与えられる。ここで与えた名称は pin、segment 等のキーワードの中で参照される。

```
primitive:
  - name : circle1        # pellet  outer boundary (= clad inner boundary)
    type :  circle
    angle:  0.0
    value:  0.23240

  - name : circle2        # clad outer boundary
    type :  circle
    angle:  0.0
    value:  0.27607
```

図 I.4.1　primitive セクションの入力例

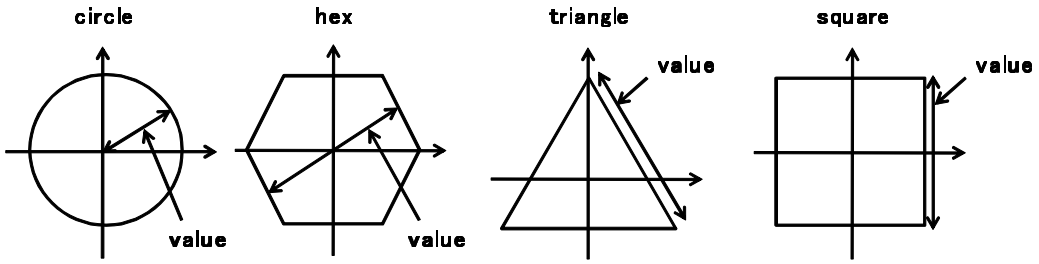

図 I.4.2　ベースとなる幾何形状の定義（回転角 0[rad]）

I.4.2　pin セクション

pin セクションの入力例を図 I.4.3 に示す。ここでは primitive セクションで定義された形状を組み合わせて燃料ピンや吸収対等を定義する。name キーワードでピンの名称を、boundary キーワードで外枠となる形状を与え、composition キーワードで primitive な形状の組み合わせで生じる領域に関する定義を与える。この入力例では、circle1 という形状の内部領域の名称が fuel1 であること、及び、pin1 の内部に circle1 が存在し、pin1（自分自身の外枠）と circle1 との間の領域の名称が clad1 であることを定義している。形状の前に付与されたプラス記号はその形状の内側を表し、マイナス記号は外側を表す。なお、ここで与えられる領域の名称を元にこの領域の物質組成が決定される。

```
pin:
  − name: pin1                                  # fuel
    boundary: circle2
    composition:
      − name : fuel1
        region: [+circle1 ]
      − name : clad1
        region: [−circle1 , +pin1]
```

図 I.4.3　pin セクションの入力例

I.4.3 lattice セクション

lattice セクションの入力例を図 I.4.4 に示す。ここでは pin セクションで定義された形状を適当に配列させた形状を定義する。この入力例では arrangement キーワードで番号 1 の燃料ピンを格子状に配列させている。そして pin_index キーワードで番号 1 の燃料ピンを pin1 と定義している（pin1 は事前に pin セクションで定義されている幾何形状である）。pitch キーワードは、ピンを配置する際のピン間隔を与える。name および boundary キーワードについては pin セクションにおけるものと同様で、定義した lattice の名称と lattice の外枠となる形状を与える。

```
lattice :
  - name: lattice1
    boundary: hex1
    pitch: 0.647
    arrangement:
      - [           1,  1,  1,  1,  1,  1,  1          ]
      - [         1,  1,  1,  1,  1,  1,  1,  1        ]
      - [       1,  1,  1,  1,  1,  1,  1,  1,  1      ]
      - [     1,  1,  1,  1,  1,  1,  1,  1,  1,  1    ]
      - [   1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1  ]
      - [ 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1 ]
      - [1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1]
      - [ 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1 ]
      - [   1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1  ]
      - [     1,  1,  1,  1,  1,  1,  1,  1,  1,  1    ]
      - [       1,  1,  1,  1,  1,  1,  1,  1,  1      ]
      - [         1,  1,  1,  1,  1,  1,  1,  1        ]
      - [           1,  1,  1,  1,  1,  1,  1          ]
      - [             1,  1,  1,  1,  1,  1,  1        ]
    pin_index:
      - number: 1
        name:   pin1
```

図 I.4.4 lattice セクションの入力例

I.4.4 segment セクション

segment セクションの入力例を図 I.4.5 に示す。ここでは primitive，pin，lattice セクションで定義された幾何形状を用いて集合体の XY 平面断面の幾何形状を定義する。キーワードの構成は pin セクションのものと同一であり、領域の定義方法についても pin セクションと同様である。

この入力例では、lattice1 の内側で定義される fuel_lattice という名前の領域、lattice1 の外側と hex2 の内側で定義される wrapper1 という名前の領域、hex2 の外側から外側境界の内側で定義される coolant1 という名前の領域の 3 つの領域で構成される core0 という名前のセグメントが定義されている。

```
segment:
  − name: core0
    boundary: hex3
    composition:
      − name : fuel_lattice
        region: [+lattice1 ]
      − name : wrapper1
        region: [−lattice1 , +hex2]
      − name : coolant1
        region: [−hex2, +core0]
```

図 I.4.5 segment セクションの入力例

I.4.5 assembly セクション

assembly セクションの入力例を図 I.4.6 に示す。ここでは segment セクションで定義された 2 次元幾何形状を Z 軸方向に積み重ねることで詳細集合体幾何形状を定義する。すなわち（assembly セクションの）segment キーワードを使って、segment セクションで定義された 2 次元集合体幾何形状を指定し、axial_range キーワードでその幾何形状が割り当てられる Z 軸方向の範囲を指定する。name キーワードで与えられる名称がこの幾何形状をもつ集合体の集合体タイプ名となる。

制御棒集合体の幾何形状を定義する場合には、制御棒集合体が全挿入された位置で定義し、下側に炉心計算モデルの有効長と同じ長さの幾何形状を定義しておく。制御棒集合体が引き抜かれたときにこの幾何形状に置換される。この例では、炉心計算モデルの有効長は 180.0cm であり、炉心モデルの下端位置（0.0cm）から下側に-180.0cm まで幾何形状が定義されている。このような幾何形状定義しておくことで、制御棒集合体を引き抜いていくと吸収材領域がナトリウムフォロワーに置換されるというモデルを表現することができる。

```
assembly:
  − name: fuel_assembly
    composition:
      − axial_range: [120.00,  180.00]
        segment    : reflector_segment
      − axial_range: [ 60.00,  120.00]
        segment    : fuel_segment
      − axial_range: [   0.00,   60.00]
        segment    : reflector_segment

  − name: control_rod_assembly
    composition:
      − axial_range: [120.00,  180.00]
        segment    : na_follower_segment
      − axial_range: [ 60.00,  120.00]
        segment    : b4c_segment
      − axial_range: [   0.00,   60.00]
        segment    : na_follower_segment
      − axial_range: [−180.00,    0.00]
        segment    : na_follower_segment
```

図 I.4.6　assembly セクションの入力例

I.5　装荷パターンファイル

装荷パターンファイルでは、燃料集合体・反射体・制御棒等の炉心への装荷パターンの定義及びゾーン（同一のミクロ断面積が与えられる領域）の定義を行う。装荷パターンファイルはサイクル毎に個別に作成する必要がある。

本ファイルのセクションの一覧は表のとおりである。

表 I.5.1　装荷パターンファイルのセクション一覧

セクション	内容
core_name	炉心名を指定する。
cycle_name	サイクル名を指定する。
cycle_number	積算サイクル数を指定する。
fuels	今サイクルで新たに装荷される燃料集合体を定義する。
load_fuels	燃料集合体の装荷アドレスを指定する。
reflectors	今サイクルで新たに装荷される反射体を定義する。
load_reflectors	反射体の装荷アドレスを指定する。
control_rods	今サイクルで新たに装荷される制御棒を定義する。
load_control_rods	制御棒の装荷アドレスを指定する。
sources	今サイクルで新たに装荷される中性子源を定義する。
load_sources	中性子源の装荷アドレスを指定する。
reactor_outside	炉外（外側反射体の外側）に仮想的に存在する集合体を定義する。
zone_set	ゾーンを定義する。

I.5.1　fuels セクションと load_fuels セクション

fuels セクション及び load_fuels セクションの入力例を図 I.5.1 に示す。fuels セクションでは今サイクルで新たに装荷される新燃料集合体を定義する。label キーワードで集合体ラベルを与え、type キーワードで集合体タイプを指定する。ここで指定される集合体タイプは、幾何形状ファイルの assembly セクションで定義されたものである。このサンプルでは集合体ラベルが PFD066～PFD068 で集合体タイプが fuel_pfd1 の燃料集合体 3 体と、集合体ラベルが PFC010 で集合体タイプが fuel_pfb1 という集合体 1 体を定義している。なお、新燃料集合体を定義する際に任意の一文字を表す「?」を使うことで定義を省略することができる。例えば「REF」で始まり続く任意の 3 文字の反射体集合体を定義したい場合には、集合体ラベルを「REF???」とすることができる。

load_fuels セクションでは fuels セクションで定義した新燃料と前サイクルからの引継ぎ燃料、さらに再使用燃料の装荷情報を定義する。例では、fuels セクションで定義した集合体ラベル PFD066～PFD068 の新燃料をそれぞれ集合体アドレス 1A1，1C1，1E1 に装荷し、同じく新燃料 PFC010 を 2E2 に装荷している。また集合体アドレス 2B1 には前サイクルからの引継ぎ燃料である PFB010 を装荷している。

集合体を装荷する際の回転を定義するには、load_fuels セクションで maxKey キーワードを使用し、集合体の MaxKey の位置を指定する。例えば PFB010 を回転させる場合は`{address:2B1, label:PFB010, maxKey:2}` 等と指定する[3]。なお、集合体の MaxKey は集合体の装荷方向を決めるための概念であり、回転していない状態を 0 とし、反時計回りに 60°、120°、180°、240°、300° 回転した状態はそれぞれ 1、2、3、4、5 で表現される。MaxKey の定義については文献 [48] を参照することができる。

引継ぎ燃料や再使用燃料の装荷情報の指定は、集合体ラベル名のみならず移動情報を用いることでも可能である。例えば前々サイクルの 2B1 に装荷されていた集合体を 1A1 に装荷する場合は `{address:1A1, pre_location: [-2, 2B1]}` 等と指定する。ここで、-2 は前々サイクルを意味する。前サイクルの場合は-1 である。もしくは今サイクルをサイクル 4 として、サイクル 3 の 2C1 に装荷されていた燃料を 1D1 に引き継ぐ場合、`{address:1D1, pre_location: [3, 2C1]}` 等と指定することも可能である。

```
fuels :
  – label: [PFD066, PFD068]
    type: fuel_pfd1

  – label: PFC010
    type: fuel_pfb1

load_fuels :
  – { address: 1A1, label: PFD066 }
  – { address: 1C1, label: PFD067 }
  – { address: 1E1, label: PFD068 }
  – { address: 2B1, label: PFB010 }
  – { address: 2E2, label: PFC010 }
```

図 I.5.1　fuels セクション及び load_fuels セクションの入力例

同様に、reflectors と load_reflectors セクション、control_rods と load_control_rods セクション、sources と load_sources セクションでそれぞれ、反射体、制御棒、中性子源の集合体と装荷位置を定義する。

I.5.2　reactor_outside セクション

reactor_outside セクションは、計算モデル上、炉心として定義されていない外側反射体の外側（炉外）の領域に仮想的な物質が存在することを指定するために使う。ここで指定された仮想的な集合体（通常は均質化されたマテリアル）が炉外領域を埋め尽くし、この炉外領域全体がひとつ

[3] maxKey キーワードは小文字で始まり K のみ大文字であることに注意が必要である。

のゾーンとして自動的に取り扱われる。

I.5.3 zone_set セクション

zone_set セクションの入力例を図 I.5.2 に示す。zone_set セクションによりゾーン（同一のミクロ断面積が与えられる領域）の定義を行う。ゾーンは径方向には集合体単位、軸方向にはプレーン単位で炉心を分割したものである。したがって、ゾーンの定義は集合体アドレスのセットとプレーン番号のセットの組み合わせによって行う。すなわち、address キーワードで集合体アドレスのセットを定義し、plane キーワードで軸方向のプレーン番号のセットを定義する。そして、zone キーワードでそれら 2 つのキーワードで与えられた情報と組み合わせてゾーンを定義する。zone はゾーン番号とあわせて定義される。なお、ゾーン番号は正の整数とする[4]。

この例では径方向の集合体のセットとして rid1 と rid2 が定義され、軸方向プレーンのセットとして zid1、zid2、zid3 が定義されている。そして、これらの組み合わせとして計 6 個のゾーンが定義されている。ここで rid1 は 000、1B1、1D 1、･･･ 、5E4、5F4 の計 64 の集合体アドレスを表し、rid2 は 1A1、1C1、1E1 の 3 つの集合体アドレスを表している。

なお、address キーワードにおいて集合体アドレスを定義する際、例えば「2A1+」と記述した場合は 2A1 の 60° 回転対称位置を含めたアドレス（2A1, 2B1, 2C1, 2D1, 2E1, 2F1 の 6 つ）を表している。

各ゾーンの領域を指定する際、address と plane というキーワードを使用する。address では径方向の集合体範囲を指定し、plane では軸方向のプレーン範囲を指定する。次に各ゾーンにおける実効ミクロ定数を求める格子計算を実施する際に集合体をどのようにモデル化するかを指定する。指定は cell キーワードにて行う。cell キーワードに指定できる値を表 I.5.2 に示す。また、各ゾーン毎に当該ゾーンが燃焼計算の対象となるのか否かを指定する。指定は burnable キーワードで行い、値は True または False のいずれかである。指定を省略した場合は、False を指定したものとみなされ、燃焼計算の対象とならない。

なお、制御棒領域のゾーンについては、制御棒が全挿入しているものとして定義する。炉心下側に存在するゾーンについては定義する必要はない。なお、同ゾーンの軸方向分割はマテリアルメッシュの軸方向分割と一致させておく必要がある（ここではマテリアルメッシュが軸方向に 3 分割されているものとする）。制御棒領域のゾーンは非燃焼とする[5]。

[4] ゾーン番号 0（零）は炉外領域（reactor_outside セクションで指定した領域）を表す。

[5] 現状の ORPHEUS の計算モデルでは、制御棒の燃焼と制御棒の移動を同時に扱うことができない [72)]。制御棒集合体を通常の燃料集合体として定義し「burnable: True」とすることで制御棒領域を燃焼するものとしてモデル化することは可能である。当然のことながら、この場合、制御棒の引き抜き操作に関する機能を利用できないので、特定の引き抜き状態をモデル化しておく必要がある。

```
zone_set:
  address:
    rid1:  [ 000 ,  1B1 ,  1D1 ,  1F1 ,  2A1+, 2A2+, 3A1+, 3A2+, 4A1+,
             4A2+, 4A3+, 4A4+, 5A3+, 5A4+]
    rid2:  [ 1A1 ,  1C1 ,  1E1 ]

  plane:
    zid1:  [ 0,  0]          # plane 0
    zid2:  [ 1,  1]          # plane 1
    zid3:  [ 2,  2]          # plane 2

  zone:
    1: {address: rid1,  plane: zid1,  cell :  homo}
    2: {address: rid1,  plane: zid2,  cell :  ring,  burnable: True}
    3: {address: rid1,  plane: zid3,  cell :  homo}
    4: {address: rid2,  plane: zid1,  cell :  homo}
    5: {address: rid2,  plane: zid2,  cell :  control_rod}
    6: {address: rid2,  plane: zid3,  cell :  homo}
```

図 I.5.2　zone_set セクションの入力例

表 I.5.2　cell キーワードの値一覧

値	内容
homo	格子計算を均質モデルで行う。
ring	格子計算を 1 次元リングモデルで行う。
control_rod	格子計算を 1 次元リングモデルを用いた RRRP 法で行う。

I.6 炉心特性ファイル

炉心特性ファイルでは、炉心名称や熱出力等の各炉心に固有のデータを定義する。また炉心全体の形状、すなわち燃料集合体、反射体、制御棒等の配置情報（集合体アドレス）を定義する。本ファイルのセクション一覧は表のとおりである。

表 I.6.1　炉心特性ファイルのセクション一覧

セクション	内容
name	炉心名を指定する。
power	定格熱出力 [MWth] を指定する。
assembly	集合体の配置情報を指定する。
fuel	燃料集合体の配置情報を指定する。
reflector	反射体の配置情報を指定する。
control_rod	制御棒の配置情報を指定する。
source	中性子源の配置情報を指定する。

I.6.1 assembly セクション

assembly セクションの入力例を図に示す。assembly セクションでは燃料・反射体等を含めた全集合体の配置情報（集合体アドレス）を指定する。指定方法は 2 種類ある。ひとつは体系に含まれる集合体の層数を与えてから実際に体系に含まれない集合体アドレスを除外する方法であり、もうひとつは体系に含まれる全ての集合体アドレスを指定する方法である。ここでは前者について説明する。後者については後述の fuel セクションや reflector セクションでの定義に使用される方法と同じである。

まず体系に含まれる集合体の層数を layer キーワードで指定する。ここで層数とは最外周の集合体アドレスから定義される。例えば最外周の反射体の集合体アドレスが 10A3 等である場合は層数を 10 と定義する。次に体系に含まれない集合体アドレスを except キーワードで指定する。サンプルでは 10A1，10A2 および 10A10 とその 60° 回転対称位置の計 18 体を除外している（プラス記号を用いたアドレス表記については装荷パターンファイルの zone_set セクションで用いられているものと同じである）。

```
assembly:
  layer : 10
  except: [10A1+, 10A2+, 10A10+]
```

図 I.6.1　assembly セクションの入力例

I.6.2　fuel、reflector、control_rod、source セクション

これらのセクションでは燃料、反射体、制御棒、中性子源の配置情報（集合体アドレス）を指定する。基本的に fuel、reflector、control_rod、source セクションは同一の構造を持っており、ここでは control_rod セクションの入力例を図 I.6.2 に示す。指定は address キーワードに集合体アドレスを羅列することで行う。この例では、制御棒の集合体アドレスとして 3A3 とその 60° 回転対称位置の 6 つの集合体アドレスを指定している（プラス記号を用いたアドレス表記については装荷パターンファイルの zone_set セクションで用いられているものと同じである）。

なお、これらのセクションのうち、control_rod セクションだけは、バンクグループを定義するための bank_group キーワードが指定可能となっている。この例では炉心アドレス 3A3、3C3、3E3 の 3 本の制御棒、3B3、3D3、3F3 の 3 本の制御棒ををまとめて、それぞれ「bank1」と「bank2」というバンクグループ名で 2 つのバンクを定義している。ここで定義したバンクグループは、計算条件ファイルの step セクションで制御棒位置を指定する際に利用することができる。

```
control_rod:
  address: [3 A3+]
  bank_group:
    bank1: [3A3, 3C3, 3E3]
    bank2: [3B3, 3D3, 3F3]
```

図 I.6.2　control_rod セクションの入力例

I.7　断面積情報ファイル

断面積情報ファイルは、ユーザ指定のミクロ断面積を用いた解析を行う計算モードである constant_micro を用いた場合にのみ必要となる入力ファイルである。本ファイルのセクション一覧は以下のとおりである。

表 I.7.1　断面積情報ファイルのセクション一覧

セクション	内容
newpds_dir	NewPDS ファイルが存在するディレクトリを指定する。
nuclides	ミクロ断面積が定義されている核種の一覧を指定する。
xs_table	各ゾーンに割り当てる NewPDS ファイルのメンバー名を指定する。
fission_spectrum	実効マクロ断面積に与える核分裂スペクトルを指定する。
fs_zone	代表的な核分裂スペクトルを与えるゾーン番号を指定する。
energy_group	エネルギー群数を指定する。

I.7.1　xs_table セクション

xs_table セクションの入力例を図に示す。xs_table セクションでは各ゾーンに割り当てられる実効ミクロ断面積を指定する。すなわち、ゾーン番号をキーとして対応する PDS ファイル名を指定する。なお、ゾーン番号は装荷パターンファイルで定義したゾーン番号である。ゾーン番号 0 は炉外（装荷パターンファイルの reactor_outside）で定義する領域を表す。

```
xs_table:
  0:  RDR
  1:  FUEL
  2:  AXR
  3:  AXR
  4:  RDR
  5:  B4C
  6:  NA
```

図 I.7.1　xs_table セクションの入力例

I.7.2　fission_spectrum セクション

実効マクロ断面積作成時の核分裂スペクトルを外部から与えるようになっており、fission_spectrum セクションで、各ゾーンの核分裂スペクトルのデータを指定する。

国際単位系（SI）

表１．SI 基本単位

基本量	SI 基本単位	
	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd

表２．基本単位を用いて表されるSI組立単位の例

組立量	SI 基本単位	
	名称	記号
面積	平方メートル	m^2
体積	立法メートル	m^3
速さ，速度	メートル毎秒	m/s
加速度	メートル毎秒毎秒	m/s^2
波数	毎メートル	m^{-1}
密度，質量密度	キログラム毎立方メートル	kg/m^3
面積密度	キログラム毎平方メートル	kg/m^2
比体積	立方メートル毎キログラム	m^3/kg
電流密度	アンペア毎平方メートル	A/m^2
磁界の強さ	アンペア毎メートル	A/m
量濃度(a)，濃度	モル毎立方メートル	mol/m^3
質量濃度	キログラム毎立法メートル	kg/m^3
輝度	カンデラ毎平方メートル	cd/m^2
屈折率(b)	（数字の） 1	1
比透磁率(b)	（数字の） 1	1

(a) 量濃度（amount concentration）は臨床化学の分野では物質濃度（substance concentration）ともよばれる。
(b) これらは無次元量あるいは次元１をもつ量であるが、そのことを表す単位記号である数字の１は通常は表記しない。

表３．固有の名称と記号で表されるSI組立単位

組立量	SI 組立単位			
	名称	記号	他のSI単位による表し方	SI基本単位による表し方
平面角	ラジアン(b)	rad	1 (b)	m/m
立体角	ステラジアン(b)	sr(c)	1 (b)	m^2/m^2
周波数	ヘルツ(d)	Hz		s^{-1}
力	ニュートン	N		m kg s^{-2}
圧力，応力	パスカル	Pa	N/m^2	m^{-1} kg s^{-2}
エネルギー，仕事，熱量	ジュール	J	N m	m^2 kg s^{-2}
仕事率，工率，放射束	ワット	W	J/s	m^2 kg s^{-3}
電荷，電気量	クーロン	C		s A
電位差（電圧），起電力	ボルト	V	W/A	m^2 kg s^{-3} A^{-1}
静電容量	ファラド	F	C/V	m^{-2} kg^{-1} s^4 A^2
電気抵抗	オーム	Ω	V/A	m^2 kg s^{-3} A^{-2}
コンダクタンス	ジーメンス	S	A/V	m^{-2} kg^{-1} s^3 A^2
磁束	ウエーバ	Wb	Vs	m^2 kg s^{-2} A^{-1}
磁束密度	テスラ	T	Wb/m^2	kg s^{-2} A^{-1}
インダクタンス	ヘンリー	H	Wb/A	m^2 kg s^{-2} A^{-2}
セルシウス温度	セルシウス度(e)	℃		K
光束	ルーメン	lm	cd sr(c)	cd
照度	ルクス	lx	lm/m^2	m^{-2} cd
放射性核種の放射能(f)	ベクレル(d)	Bq		s^{-1}
吸収線量，比エネルギー分与，カーマ	グレイ	Gy	J/kg	m^2 s^{-2}
線量当量，周辺線量当量，方向性線量当量，個人線量当量	シーベルト(g)	Sv	J/kg	m^2 s^{-2}
酸素活性	カタール	kat		s^{-1} mol

(a)SI接頭語は固有の名称と記号を持つ組立単位と組み合わせても使用できる。しかし接頭語を付した単位はもはやコヒーレントではない。
(b)ラジアンとステラジアンは数字の１に対する単位の特別な名称で、量についての情報をつたえるために使われる。実際には、使用する時には記号rad及びsrが用いられるが、習慣として組立単位としての記号である数字の１は明示されない。
(c)測光学ではステラジアンという名称と記号srを単位の表し方の中に、そのまま維持している。
(d)ヘルツは周期現象についてのみ、ベクレルは放射性核種の統計的過程についてのみ使用される。
(e)セルシウス度はケルビンの特別な名称で、セルシウス温度を表すために使用される。セルシウス度とケルビンの単位の大きさは同一である。したがって、温度差や温度間隔を表す数値はどちらの単位で表しても同じである。
(f)放射性核種の放射能（activity referred to a radionuclide）は、しばしば誤った用語で"radioactivity"と記される。
(g)単位シーベルト（PV,2002,70,205）についてはCIPM勧告2（CI-2002）を参照。

表４．単位の中に固有の名称と記号を含むSI組立単位の例

組立量	SI 組立単位		
	名称	記号	SI 基本単位による表し方
粘度	パスカル秒	Pa s	m^{-1} kg s^{-1}
力のモーメント	ニュートンメートル	N m	m^2 kg s^{-2}
表面張力	ニュートン毎メートル	N/m	kg s^{-2}
角速度	ラジアン毎秒	rad/s	m m^{-1} s^{-1}=s^{-1}
角加速度	ラジアン毎秒毎秒	rad/s^2	m m^{-1} s^{-2}=s^{-2}
熱流密度，放射照度	ワット毎平方メートル	W/m^2	kg s^{-3}
熱容量，エントロピー	ジュール毎ケルビン	J/K	m^2 kg s^{-2} K^{-1}
比熱容量，比エントロピー	ジュール毎キログラム毎ケルビン	J/(kg K)	m^2 s^{-2} K^{-1}
比エネルギー	ジュール毎キログラム	J/kg	m^2 s^{-2}
熱伝導率	ワット毎メートル毎ケルビン	W/(m K)	m kg s^{-3} K^{-1}
体積エネルギー	ジュール毎立方メートル	J/m^3	m^{-1} kg s^{-2}
電界の強さ	ボルト毎メートル	V/m	m kg s^{-3} A^{-1}
電荷密度	クーロン毎立方メートル	C/m^3	m^{-3} sA
表面電荷	クーロン毎平方メートル	C/m^2	m^{-2} sA
電束密度，電気変位	クーロン毎平方メートル	C/m^2	m^{-2} sA
誘電率	ファラド毎メートル	F/m	m^{-3} kg^{-1} s^4 A^2
透磁率	ヘンリー毎メートル	H/m	m kg s^{-2} A^{-2}
モルエネルギー	ジュール毎モル	J/mol	m^2 kg s^{-2} mol^{-1}
モルエントロピー，モル熱容量	ジュール毎モル毎ケルビン	J/(mol K)	m^2 kg s^{-2} K^{-1} mol^{-1}
照射線量（X線及びγ線）	クーロン毎キログラム	C/kg	kg^{-1} sA
吸収線量率	グレイ毎秒	Gy/s	m^2 s^{-3}
放射強度	ワット毎ステラジアン	W/sr	m^4 m^{-2} kg s^{-3}=m^2 kg s^{-3}
放射輝度	ワット毎平方メートル毎ステラジアン	$W/(m^2$ sr)	m^2 m^{-2} kg s^{-3}=kg s^{-3}
酵素活性濃度	カタール毎立方メートル	kat/m^3	m^{-3} s^{-1} mol

表５．SI 接頭語

乗数	接頭語	記号	乗数	接頭語	記号
10^{24}	ヨタ	Y	10^{-1}	デシ	d
10^{21}	ゼタ	Z	10^{-2}	センチ	c
10^{18}	エクサ	E	10^{-3}	ミリ	m
10^{15}	ペタ	P	10^{-6}	マイクロ	μ
10^{12}	テラ	T	10^{-9}	ナノ	n
10^9	ギガ	G	10^{-12}	ピコ	p
10^6	メガ	M	10^{-15}	フェムト	f
10^3	キロ	k	10^{-18}	アト	a
10^2	ヘクト	h	10^{-21}	ゼプト	z
10^1	デカ	da	10^{-24}	ヨクト	y

表６．SIに属さないが、SIと併用される単位

名称	記号	SI 単位による値
分	min	1 min=60s
時	h	1h =60 min=3600 s
日	d	1 d=24 h=86 400 s
度	°	1°=(π/180) rad
分	’	1’=(1/60)°=(π/10800) rad
秒	”	1”=(1/60)’=(π/648000) rad
ヘクタール	ha	$1ha=1hm^2=10^4m^2$
リットル	L, l	$1L=11=1dm^3=10^3cm^3=10^{-3}m^3$
トン	t	$1t=10^3$ kg

表７．SIに属さないが、SIと併用される単位で、SI単位で表される数値が実験的に得られるもの

名称	記号	SI 単位で表される数値
電子ボルト	eV	$1eV=1.602\ 176\ 53(14)\times10^{-19}J$
ダルトン	Da	$1Da=1.660\ 538\ 86(28)\times10^{-27}kg$
統一原子質量単位	u	1u=1 Da
天文単位	ua	$1ua=1.495\ 978\ 706\ 91(6)\times10^{11}m$

表８．SIに属さないが、SIと併用されるその他の単位

名称	記号	SI 単位で表される数値
バール	bar	$1\ bar=0.1MPa=100kPa=10^5Pa$
水銀柱ミリメートル	mmHg	1mmHg=133.322Pa
オングストローム	Å	$1\ Å=0.1nm=100pm=10^{-10}m$
海里	M	1 M=1852m
バーン	b	$1\ b=100fm^2=(10^{-12}cm)2=10^{-28}m^2$
ノット	kn	1 kn=(1852/3600)m/s
ネーパ	Np	SI単位との数値的な関係は、対数量の定義に依存。
ベル	B	
デジベル	dB	

表９．固有の名称をもつCGS組立単位

名称	記号	SI 単位で表される数値
エルグ	erg	$1\ erg=10^{-7}$ J
ダイン	dyn	$1\ dyn=10^{-5}$N
ポアズ	P	1 P=1 dyn s cm^{-2}=0.1Pa s
ストークス	St	1 St =$1cm^2$ s^{-1}=$10^{-4}m^2$ s^{-1}
スチルブ	sb	1 sb =1cd cm^{-2}=10^4cd m^{-2}
フォト	ph	1 ph=1cd sr cm^{-2} 10^4lx
ガル	Gal	1 Gal =1cm s^{-2}=$10^{-2}ms^{-2}$
マクスウェル	Mx	1 Mx = 1G cm^2=10^{-8}Wb
ガウス	G	1 G =1Mx cm^{-2} =10^{-4}T
エルステッド(c)	Oe	1 Oe ≙ $(10^3/4\pi)$A m^{-1}

(c) ３元系のCGS単位系とSIでは直接比較できないため、等号「 ≙ 」は対応関係を示すものである。

表10．SIに属さないその他の単位の例

名称	記号	SI 単位で表される数値
キュリー	Ci	$1\ Ci=3.7\times10^{10}Bq$
レントゲン	R	$1\ R=2.58\times10^{-4}C/kg$
ラド	rad	$1\ rad=1cGy=10^{-2}Gy$
レム	rem	$1\ rem=1\ cSv=10^{-2}Sv$
ガンマ	γ	1γ=1 nT=10-9T
フェルミ		1フェルミ=1 fm=10-15m
メートル系カラット		1メートル系カラット = 200 mg = 2×10-4kg
トル	Torr	1 Torr = (101 325/760) Pa
標準大気圧	atm	1 atm = 101 325 Pa
カロリー	cal	1cal=4.1858J（「15℃」カロリー），4.1868J（「IT」カロリー）4.184J（「熱化学」カロリー）
ミクロン	μ	1 μ =1μm=10^{-6}m

（第8版，2006年改訂）

この印刷物は再生紙を使用しています