JAEA-Data/Code
2025-021
DOI:10.11484/jaea-data-code-2025-021

JAEA-Data/Code

マルチフィジックス・シミュレーション・プラットフォームJAMPANの開発

Development of the Multi-physics Simulation Platform JAMPAN

神谷 朋宏　近藤 諒一　福田 貴斉　福田 航大
多田 健一　小野 綾子　長家 康展　吉田 啓之

Tomohiro KAMIYA, Ryoichi KONDO, Takanari FUKUDA, Kodai FUKUDA
Kenichi TADA, Ayako ONO, Yasunobu NAGAYA and Hiroyuki YOSHIDA

原子力科学研究所
原子力基礎工学研究センター

Nuclear Science and Engineering Center
Nuclear Science Research Institute

March 2026

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

本レポートは国立研究開発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。
本レポートはクリエイティブ・コモンズ 表示 4.0 国際 ライセンスの下に提供されています。
本レポートの成果（データを含む）に著作権が発生しない場合でも、同ライセンスと同様の
条件で利用してください。（https://creativecommons.org/licenses/by/4.0/deed.ja）
なお、本レポートの全文は日本原子力研究開発機構ウェブサイト（https://www.jaea.go.jp）
より発信されています。本レポートに関しては下記までお問合せください。

国立研究開発法人日本原子力研究開発機構　研究開発推進部　科学技術情報課
〒 319-1112　茨城県那珂郡東海村大字村松 4 番地 49
E-mail: ird-support@jaea.go.jp

This report is issued irregularly by Japan Atomic Energy Agency.
This work is licensed under a Creative Commons Attribution 4.0 International License
(https://creativecommons.org/licenses/by/4.0/deed.en).
Even if the results of this report (including data) are not copyrighted, they must be used under
the same terms and conditions as CC-BY.
For inquiries regarding this report, please contact Library, Institutional Repository and INIS Section,
Research and Development Promotion Department, Japan Atomic Energy Agency.
4-49 Muramatsu, Tokai-mura, Naka-gun, Ibaraki-ken 319-1112, Japan
E-mail: ird-support@jaea.go.jp

© Japan Atomic Energy Agency, 2026

JAEA-Data/Code 2025-021

マルチフィジックス・シミュレーション・プラットフォーム JAMPAN の開発

日本原子力研究開発機構
原子力科学研究所
原子力基礎工学研究センター

神谷　朋宏、近藤　諒一、福田　貴斉、福田　航大+、多田　健一、
小野　綾子、長家　康展、吉田　啓之

(2025 年 12 月 3 日受理)

日本原子力研究開発機構は中性子輸送計算コードや熱流動計算コードなどのシングルフィジックスコードを結合するため、マルチフィジックス・シミュレーション・プラットフォーム JAMPAN を開発している。JAMPAN は、HDF5 形式のデータコンテナとシングルフィジックスコードの入力作成および出力読み取りのためのモジュールから構成されている。ユーザーは結合する計算コードに適合した入出力アクセサモジュールを使用することで、簡単に計算コードを追加・変更することができる。JAMPAN には、中性子輸送計算コード MVP と熱流動計算コード JUPITER、ACE-3D、NASCA に対応したインターフェースが実装されており、炉心設計コードの参照解を提供するための核熱結合計算を行うことができる。ユーザーは必要となる計算精度に応じて熱流動計算コードを選択することが可能である。また、燃料棒の物性値の算出として FEMAXI を利用することが可能である。本報告書では JAMPAN の概要と利用方法について説明する。

原子力科学研究所　〒319-1195　茨城県那珂郡東海村大字白方 2-4
+研究開発推進部

JAEA-Data/Code 2025-021

Development of the Multi-physics Simulation Platform JAMPAN

Tomohiro KAMIYA, Ryoichi KONDO, Takanari FUKUDA, Kodai FUKUDA+, Kenichi TADA, Ayako ONO, Yasunobu NAGAYA and Hiroyuki YOSHIDA

Nuclear Science and Engineering Center,
Nuclear Science Research Institute,
Japan Atomic Energy Agency
Tokai-mura, Naka-gun, Ibaraki-ken

(Received December 3, 2025)

Japan Atomic Energy Agency has developed a high-fidelity multi-physics platform JAMPAN for connecting single-physics codes such as a neutronics code and a thermal-hydraulics code. It consists of the HDF5 formatted data container and input/output data handler modules to generate the input file and read the output file of the single-physics codes. Users can easily add or exchange the code by implementing input and output data handler modules for this code. JAMPAN is equipped with interfaces compatible with the neutronics code MVP and the thermal-hydraulics codes JUPITER, ACE-3D, and NASCA, enabling neutronics and thermal-hydraulics coupling calculations to provide reference solutions for core analysis codes. Users can select the thermal-hydraulics code depending on the required calculation accuracy. In addition, the fuel rod properties can be calculated using FEMAXI. This report explains the overview of JAMPAN.

Keywords: JAMPAN, Neutronics and Thermal-hydraulics Coupling Calculations

+Research and Development Promotion Department

目　次

Contents

表リスト

図リスト

This is a blank page.

1. 序論

近年の計算機性能の向上により大規模で高忠実な計算が可能になっている。マサチューセッツ工科大学(MIT)の Kord Smith 教授は 2003 年に全炉心モンテカルロ計算を提案した[1]。このベンチマーク計算の対象は加圧水型軽水炉(PWR)の全炉心であり、軸方向に 100 分割、径方向に 10 分割している。それぞれの局所出力の標準偏差は 1%以下にする必要があるとしている。Smith 教授は、このような大規模な計算は 2030 年にはワークステーションを用いて実現すると予想した。Smith 教授の予想通り、計算機性能の向上によりこのような大規模な計算が近い将来実現するであろう。

このように、核計算などのシングルフィジックス計算の計算精度は、計算機の進歩に伴ってこの数十年で大幅に向上している。そこで本研究では、これらの高精度なシングルフィジックス計算を組み合わせた、大規模で高忠実なマルチフィジックス計算の実用化に焦点を当てた。炉心設計コードの妥当性確認のため、複数の物理現象の相互作用の取り扱いを可能とするマルチフィジックス・シミュレーション・プラットフォーム **J**AEA **A**dvanced **M**ulti-**P**hysics **A**nalysis platform for **N**uclear systems (JAMPAN)を開発した。JAMPAN は原子炉内の中性子、熱流動、燃料解析といったそれぞれの物理現象を取り扱うシングルフィジックスの解析コードを連結するためのプラットフォームである。そのため、JAMPAN 自体はシングルフィジックスやマルチフィジックスの計算機能を持たず、各シングルフィジックス解析コードを連携するための入出力読み取り機能やデータ変換機能などを備えている。

JAMPAN は Python で実装されている。JAMPAN は、HDF5 形式のデータコンテナとシングルフィジックス解析コードの入力作成および出力読み取りのためのモジュールから構成されている。JAMPAN のコンセプトは米国のアイダホ国立研究所(INL)の MOOSE やフランスの原子力・代替エネルギー庁(CEA)の SALOME など、他のマルチフィジックス・シミュレーション・プラットフォームと同様である。

JAMPAN 開発の最初の計算対象は、炉心設計コードの参照解を提供するための核熱結合計算である。燃料集合体の複雑な体系を取り扱うことによって、より正確な流速や反応率分布を得ることが期待できる。例えば部分長燃料棒の上端部や制御棒の挿入部の複雑な熱流動挙動も考慮できるようになる。JAMPAN を用いたマルチフィジックス解析の最終目標は炉心設計コードの妥当性確認のためのモックアップテストの実施コストを削減することである。この目標を達成するため、経験式を可能な限り含まない高忠実なマルチフィジックス解析を目指している。

2. JAMPAN の特徴とインストール方法

2.1. JAMPAN で取り扱うことのできる体系

JAMPAN の最初の計算対象は核熱結合計算である。現在このプラットフォームは 1 つの中性子輸送計算コード(連続エネルギーモンテカルロ計算コード MVP)と 3 つの熱流動計算コード(多相多成分熱流動計算コード JUPITER、三次元二流体モデル解析コード ACE-3D、サブチャンネル解析コード NASCA)の連結機能を実装している。

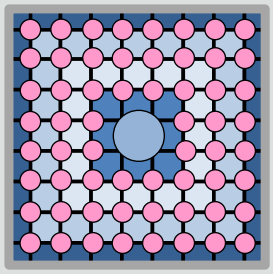
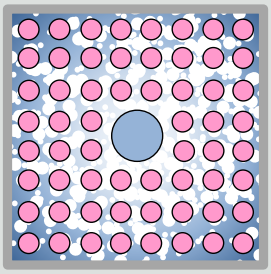
表 2.1 に核熱結合計算対象の例を示す。連結する熱流動計算コードは、計算体系の大きさや求める計算精度によって選択することが可能となっている。どのような計算体系に対しても高忠実な中性子輸送計算コードと熱流動計算コードを用いたマルチフィジックス解析を実施することが理想である。しかし、全炉心体系を JUPITER のような界面捕獲法を採用した詳細な Computational Fluid Dynamics (CFD)計算コードで計算するには、現在のコンピューターの演算性能では十分ではない。そのため、全炉心体系のような大きな体系では、計算コストを削減するため、NASCA や ACE-3D のような平均化処理を行った熱流動計算コードを用いることを想定している。NASCA や ACE-3D はこれまでの炉心設計コードで用いられている熱流動計算コードよりも高精度な熱流動計算コードであるため、MVP/NASCA、MVP/ACE-3D の核熱連結計算によって、炉心設計コードの参照解を与えることが期待できる。なお、現バージョンの JAMPAN で取り扱える中性子輸送計算コードは MVP のみである。今後は、時間依存の解析などを対象とする予定で、時間依存の解析では多群の決定論コードが必要になってくることが予想される。そのため、将来的には GENESIS や CBZ などの多群の決定論コードを JAMPAN で連結することを計画している。

JAMPAN は HDF5 フォーマットの JAMPAN 独自のデータコンテナとアクセサモジュールをもつ。アクセサモジュールはシングルフィジックスコードの入力ファイルの生成、出力ファイルの読み取りを行うことができる。図 2.1 に JAMPAN のシステム構造を示す。 ユーザーは結合するシングルフィジックスコードを指定できる。また、入力生成、出力読込モジュールを調整すれば別の計算コードを追加することができる。 コード連結時の入力生成に関しては JAMPAN が自動的に管理する。連結するコードの依存性を低減するため、出力読込モジュールがコード依存の出力フォーマットを JAMPAN 独自のフォーマットに変換する。入力生成モジュールは JAMPAN 独自のフォーマットを参照し、入力コードのデータ形式に合わせて JAMPAN のデータフォーマットを適切に変換するため、連結するコードの出力フォーマットを気にする必要はない。例えば、NASCA はサブチャンネル単位の減速材の密度を出力する。JUPITER では小さな立方体のセルに分割して計算を実行して、セルごとの減速材の密度を出力する。入出力データの単位は温度(K or ℃)、重さ(g or kg)、圧力(bar or N/m^2)、体積(cc, litter, or m^3)のようにコードに依存している。 データセットクラスはフォーマットや単位を区別するための識別子を持っている。JAMPAN はそれらのフォーマットや単位を

変換する関数を持っており、連結するコードに合わせて単位を適切に変換できる。そのためユーザーは連結するコードのフォーマットや単位を気にせずに、容易に入力生成、出力読込モジュールを導入できる。

JUPITER で単一集合体を解析する際、計算コストを鑑みると立方体のセルの 1 辺はおよそ 1 mm 程度が妥当である。しかし、MVP でコード上 1 mm の立方体のセルを取り扱うことは可能であるが、計算コストが大きすぎる。この問題への対応として JAMPAN では計算コスト低減のための減速材密度を平均化する関数を用意している。この平均化関数により、コードによって平均化する領域のサイズを自由に改変することができる。JAMPAN では、スクリプトによって MVP と JUPITER それぞれのセルサイズをユーザーが任意に指定することができる。

表 2.1 JAMPAN の核熱結合計算の計算対象と計算規模

	Whole core geometry	**Single assembly geometry**
Image of calculation geometry (single assembly)		
Range of statistical averages	Subchannel geometry	Faithfully reproduces the flow states
Calculation scale (The number of mesh divisions in space)	1,000,000 to 10,000,000 (Using workstation)	100,000,000 to 1,000,000,000 (Using supercomputer)
Thermal-hydraulics code	NASCA, ACE-3D	JUPITER
Neutronics code	MVP	MVP

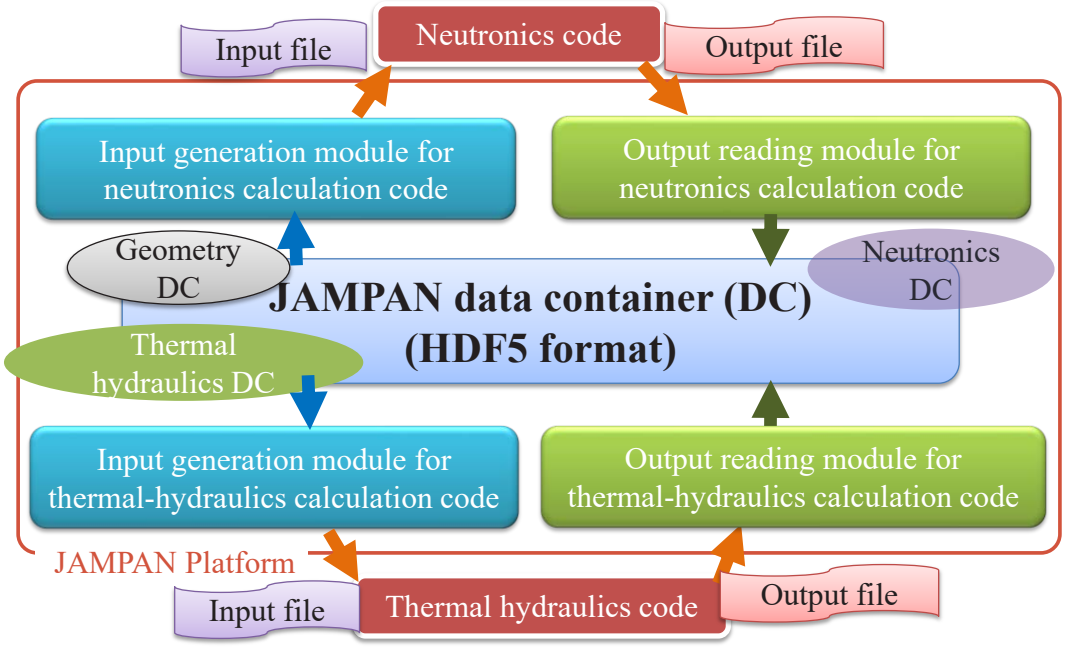


図 2.1 JAMPAN のシステム構造

2.2. JAMPAN のデータコンテナの構造

JAMPAN のデータコンテナは HDF5 フォーマットの計算グループ、時間グループ、データセットの 3 つの層で構成されている。図 2.2 に JAMPAN のデータコンテナの構成を示す。データコンテナは中性子輸送計算(核計算)、熱流動計算、燃料挙動計算といった各シングルフィジックス層のクラスを持ち、それぞれのフィジックス層は時間グループのクラスを持っている。すべてのタイムステップの計算結果をコンテナに格納すると JAMPAN のデータコンテナはファイルサイズが巨大になってしまう。核熱計算では、すべてのタイムステップの計算結果をデータコンテナに格納する必要はない。そこで JAMPAN では、使用するデータ容量を抑えるために、核計算で使用するタイムステップの分だけデータコンテナに格納するようにしている。タイムステップはユーザーが設定することができる。時間グループは連結計算用のインプットパラメーターに使用されるデータセットである。現バージョン（V.1.00.000）では核熱計算のために使用されるデータセットのみ含んでいる。近いうちにユーザーは JAMPAN の入力ファイルを使用してデータセットの選択ができるようになる予定である。

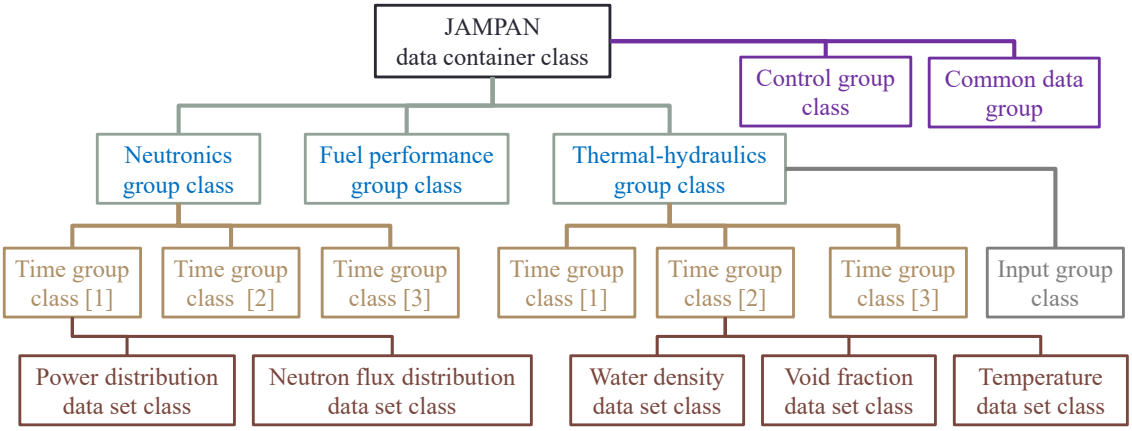


図 2.2 JAMPAN のデータコンテナの構造

2.3. JAMPAN の実行環境

JAMPAN は Python3 で実装されている。サンプルの JAMPAN 実行スクリプトが `jampan/platform/interface/scripts/`に用意されている。計算コードのパスの設定をすればジョブを実行することができるようになっている。別途、実行に必要な python パッケージと開発時のバージョンは以下の通りである。

- Python 3.6.8 以上
- PyYAML 6.0.1
- NumPy 1.26.4 (2.0 以上は未対応)
- Lark 1.1.9
- h5py 3.10.0
- Paramiko 2.12.0 (遠隔 job 投入の場合)
- Nose 1.3.7
- Cython 3.0.8
- Scipy 1.11.4
- Matplotlib 3.6.3

また、必要に応じて次の中性子輸送計算・熱流動計算関連の解析コードを準備する必要がある。

中性子輸送計算

- MVP 3.0 [2]

熱流動計算

- ACE-3D Ver.13
- JUPITER [3]
- NASCA

燃料挙動計算コード

- FEMAXI-8.1 8.1.249j [4]

MVP、ACE-3D、JUPITER、FEMAXI は高度情報科学技術機構の原子力コードセンターから利用申請を行うことができる。なお、NASCA 関連の機能については公開版には含まれていない。NASCA 関連の機能については、ユーザーが NASCA を別途入手した後に個別に配布する。

2.4. JAMPAN のインストール方法

PYTHONPATH の設定と Cython でのコンパイルが必須である。

JAMPAN は、Python モジュールとして開発されているので、標準的な Python モジュールと同様に、ライブラリの検索パス（Python の sys.path）に含まれていればそのまま利用することができる。

JAMPAN では、パッケージファイルを展開してできるディレクトリの直下にある"jampan"というディレクトリが Python モジュールになるようになっている。この jampan ディレクトリがある場所を Python に認識させるために、環境変数 PYTHONPATH を設定すればよい。

PYTHONPATH（必須設定）

Bourne シェル系列のシェル（例えば sh、bash 等）でホームディレクトリ直下に JAMPAN パッケージファイルを設置した場合は、ターミナル上で次のコマンドを実行すればよい。

```
export PYTHONPATH=$PYTHONPATH:$HOME
```

ログイン時に自動反映させるためには上記コマンドを次のファイルに追記すればよい。

```
$HOME/.bashrc
```

C シェル系列のシェル（例えば csh、tcsh 等）では、次のコマンドを実行すればよい。

```
setenv PYTHONPATH $PYTHONPATH:$HOME
```

ログイン時に自動反映させるためには上記コマンドを次のファイルに追記すればよい。

```
$HOME/.cshrc
```

Python3 を実行して import jampan が成功することを確認する。

Cython コンパイル（必須設定）

上記 PYTHONPATH の設定後、ターミナル上で次のコマンドを実行する必要がある。

```
source $HOME/.bashrc                    #C シェル系列は$HOME/.cshrc
cd (jampan_dir)/jampan/platform/capsule/Jupiter/output
python3 setup.py build_ext –inplace
mv
jampan/platform/capsule/Jupiter/output/Jupiter_output_reader_mod.cpython*.so .
```

```
cd (jampan_dir)/jampan/platform/data_container/
python3 setup.py build_ext –inplace
mv
jampan/platform/data_container/th_data_container_interface_mod.cpython*.so .
```

(jampan_dir)は適宜 jampan ディレクトリがある場所を指定すること。setup.py の実行によって生成される.so ファイルは環境によってファイル名が異なる。

これで JAMPAN のインストールは完了である。

また、JAMPAN が使用する環境変数は次の 2 つである。

JAMPAN_CODE_PATH（推奨設定）

JAMPAN_CODE_PATH は設定していなくても JAMPAN は機能するが、パスの簡易化と設計方針より、上記の解析コードやライブラリ等のパスを一つにまとめることを推奨している。Bourne シェル系列のシェルの設定例は次の通りである。

```
export JAMPAN_CODE_PATH=$HOME/local/mvp-
3.0/bin/:$HOME/local/JUPITER/build/bin/
```

C シェル系列のシェルの設定例は次の通りである。

```
setenv JAMPAN_CODE_PATH $HOME/local/mvp-
3.0/bin/:$HOME/local/JUPITER/build/bin/
```

JAMPAN_WORK_PATH（自動設定）

テストケース(現状ごく一部の関数)を実行した際に利用する一時ファイルのための作業ディレクトリである。未設定の場合、JAMPAN の存在するディレクトリと同階層に大文字の JAMPAN ディレクトリが自動で生成される。

2.5. JAMPAN のディレクトリ構造

JAMPAN のディレクトリ構造を図 2.3 に示す。JAMPAN 開発ユーザーでなければ、通常はユーザーが直接アクセスする可能性があるディレクトリは主に次の 3 か所だと想定している。

- data/
- doc/
- platform/interface/scripts/

これらの内、data には基本的なインプットファイルが参考として格納されている。また、doc には JAMPAN のマニュアルなどが格納されており、その詳細は 2.6 節に記載されている。platform/interface/scripts/は JAMPAN 実行スクリプトのサンプルが格納されている。jampan ディレクトリが PYTHONPATH で指定されていれば、JAMPAN 実行 scripts は別の

場所にコピーしても実行することができる。

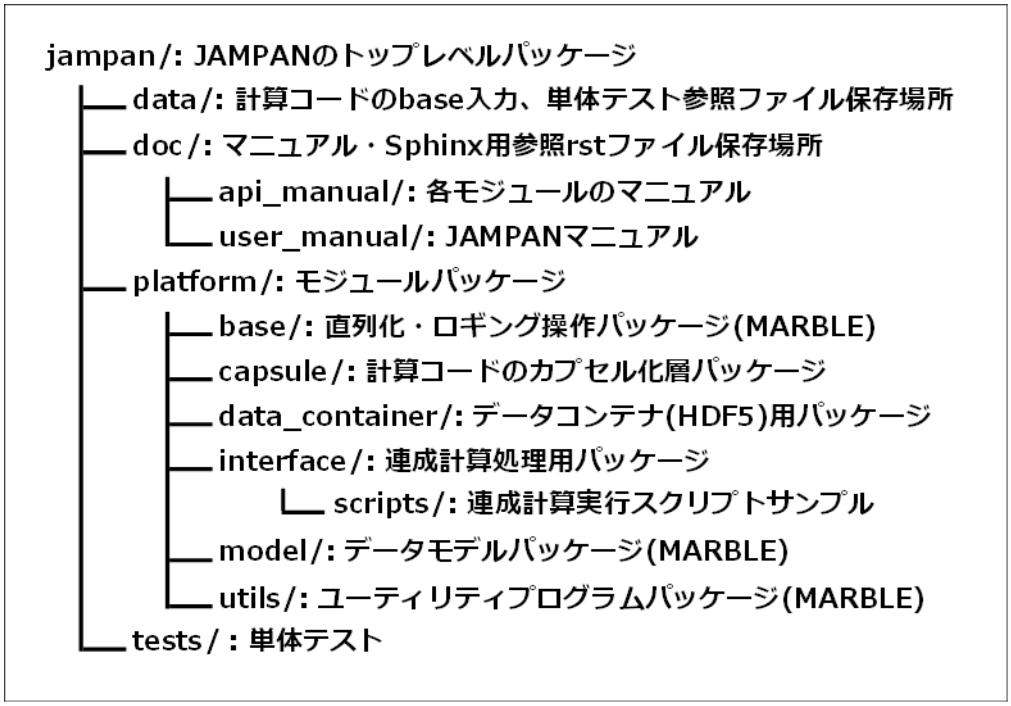


図 2.3 JAMPAN のディレクトリ構造

2.6. JAMPAN のドキュメント

JAMPAN はユーザーマニュアル(jampan/doc/user_manual/_build/html/index.html)と API マニュアル(jampan/doc/api_manual/_build/html/index.html)をパッケージ内に用意している。マニュアルは python パッケージの Sphinx で作成しており、Web ブラウザで閲覧可能である。Web ブラウザで開いたドキュメントのサンプル画像を図 2.4 に示す。

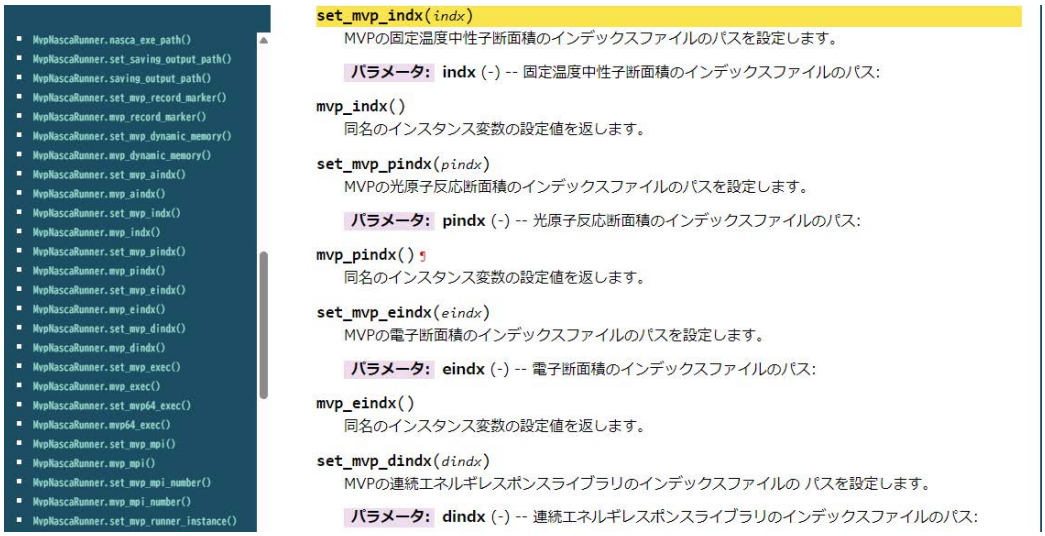


図 2.4 JAMPAN のドキュメントのブラウザ表示例

3. JAMPAN による MVP/JUPITER 連成計算について

3.1. 連成計算フロー

JAMPAN による MVP/JUPITER 連成計算は、単一集合体規模を計算対象としており、JUPITER による熱流動計算と MVP による中性子輸送計算を交互に実施することで進行する。JUPITER は大型計算機で実行され、MVP は JAMPAN が実行されている計算機で実行されることを想定している。前述の通り、JAMPAN では、独自の HDF5 フォーマットのデータコンテナである JAMPAN データコンテナを介して連成が行われる。連成計算の具体的な手続きは下記の通りである。

1) JUPITER の入力を大型計算機にアップロードし、JUPITER の計算を実行。
2) JUPITER の計算結果をダウンロードし、JAMPAN データコンテナに格納。
3) JAMPAN データコンテナにあるデータセットを用いて水密度を算出し、JAMPAN データコンテナに格納。
4) データコンテナにある水密度、温度を反映させた MVP の入力を作成。
5) MVP の計算を実行し、中性子反応率を算出。
6) 発熱量に変換し、JAMPAN データコンテナに格納。
7) JAMPAN データコンテナにある発熱量を反映した JUPITER の入力を作成。
8) 1)～7)を設定した回数繰り返す。

なお、この一連の連成計算は JAMPAN 上で自動的に実行される。なお、MVP/JUPITER 連成計算では、必ず JUPITER から開始する必要があるが、1 回目の JUPITER の計算では、燃料棒が発熱しない断熱計算であり、沸騰も生じない。

3.2. 水密度の外挿

JUPITER では、各セルは固体セルか流体セルかに分類され、図 3.1 に示されるように、燃料棒がボクセルの集合体で表現される。そのため、燃料棒の輪郭がセル境界を横切るが、そのセルは固体と判定される場合、MVP が参照する水密度が得られないといった問題がある。これを解決するために、燃料棒付近の流体セルから個体セルへと水密度を外挿する。各セルの水密度$\rho_{i,j,k}$は VOF（水の体積分率）$f_{i,j,k}$を用いて、

$$\rho_{i,j,k} = f_{i,j,k}\rho_w + \left(1 - f_{i,j,k}\right)\rho_v \tag{3.2.1}$$

と定義される。ここでρ_w、ρ_vはそれぞれ水と水蒸気の密度をそれぞれ示す。JUPITER では、固体の体積分率f^sが導入されており、各セルが固体セルか流体セルかに区別される。f^sは 0 または 1 の値を持ち、1 のときはセルが固体で完全に占められるため、fは 0 となる。したがって、式(3.2.1)から固体セルでは水密度が 0 となる。水密度は固体セルへの外挿を行うこ

とで、

$$\rho_{i,j,k} = \frac{\sum_{i'=i-1}^{i+1}\sum_{j'=j-1}^{j+1}\{f_{i',j',k}\rho_w + (1 - f_{i',j',k})\rho_g\}}{N_k^{fluid}},$$
$$N_k^{fluid} = \sum_{i'=i-1}^{i+1}\sum_{j'=j-1}^{j+1}\frac{1}{2}\left(1 + \mathrm{sgn}\left(f_{i',j',k}^{s} - \frac{1}{2}\right)\right) \quad (3.2.2)$$

と取得される。

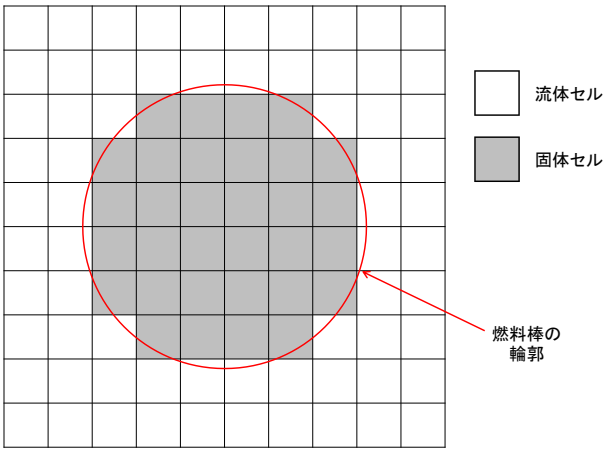


図 3.1 JUPITER における燃料棒の表現

MVP/JUPITER 連成計算では、図 3.2 に示されるように、非定常熱流動計算と定常核計算が交互に繰り返される。熱流動計算は非定常であるため、一度の計算で多数の時刻の出力データが得られる。そのため、式(3.2.2)で示される水密度は出力データ数だけ得られることとなる。JAMPAN では、非定常データを平均する場合と、各熱流動計算において最後に得られた結果をそのまま用いる場合が用意されている。なお、その水密度の導出方法の選択や熱流動計算の出力間隔は JAMPAN の入力で与える（詳しくは 3.4 節参照）。

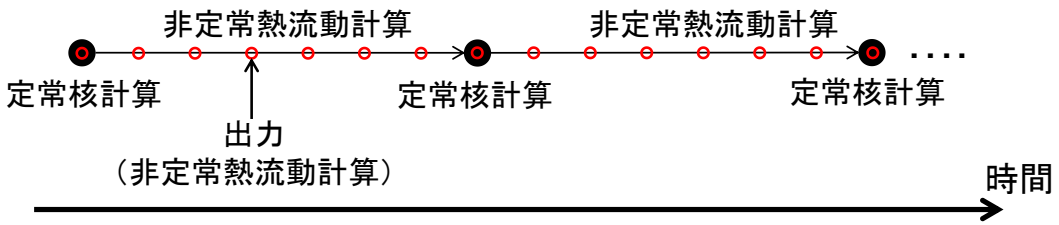


図 3.2 MVP/JUPITER 連成計算の進行

3.3. 発熱量の補間

JUPITER では、総セル数分のデータを持つバイナリファイルを読み込んで、発熱量を設定する。JAMPAN のデータコンテナに格納された MVP の発熱分布を用いて、そのバイナリファイルを作成するが、JUPITER で必要となる発熱分布は MVP で得られる発熱分布より細かいのが一般的である。そのため、MVP の発熱分布から JUPITER で必要な発熱分布を補間する。

水平断面については、MVP のセル内にある JUPITER の固体セルをカウントし、MVP の発熱量をその固体セル数で割ることで、JUPITER の各固体セルに与える発熱量を算出する。鉛直方向については、MVP のセル内で発熱量が一次関数で分布していると仮定する。その傾きは上下のセルを用いた中心差分から算出し、切片はセル内でエネルギー保存を満たすように決定する。一方で、水平断面については、MVP のセル内で発熱量が一様に分布していると仮定し、JUPITER のセルの発熱量を取得する。

3.4. 参照入力

JAMPAN では、あらかじめ用意された JUPITER の参照入力が JAMPAN の実行スクリプトに応じて、自動的に編集される。したがって、ユーザーは JAMPAN の実行スクリプトを編集することで、計算条件を変更することができ、自身で JUPITER 入力を編集する必要がない。現時点で、MVP/JUPITER 連成計算について JAMPAN が対応している体系は図 3.3 に示される BWR8×8step-II 単一集合体である。8×8 で燃料棒が並んでおり、中央には水ロッドが配置される。BWR を想定し、圧力は 7.07 MPa、温度は 549.15 K であり、下面から 2.15 m/s で水が流入する。

MVP/JUPITER 連成計算を実行する際は、実行スクリプトと同じ階層に以下のような参照入力ディレクトリ（下のツリーでは `reference` というディレクトリだが、名前は任意である）を用意する必要がある。

```
reference
├── input
│   └JUPITER
│     └stepII
│       ├── input.txt
│       ├── param.txt
│       ├── geom.txt
│       ├── flags.txt
│       ├── plist.txt
│       ├── phys.vapor_7.07MPa.txt
```

```
│           ├── phys.water_7.07MPa.txt
│           ├── phys.waterrod.txt
│           ├── phys.UO2.txt
│           └── phys.Zry.txt
└──── shell
       └JUPITER
          ├── PBS
          └── r_PBS
```

reference/input/JUPITER/step-II に配置する参照入力は以下のディレクトリに用意されている。

jampan/data/thermal-hydraulics/ref_input/input/JUPITER/stepII

また、reference/shell/JUPITER に配置する原子力機構が保有する大型計算機の PBS 系のジョブスクリプトは以下のディレクトリに用意されている。

jampan/data/thermal-hydraulics/ref_input/shell/JUPITER

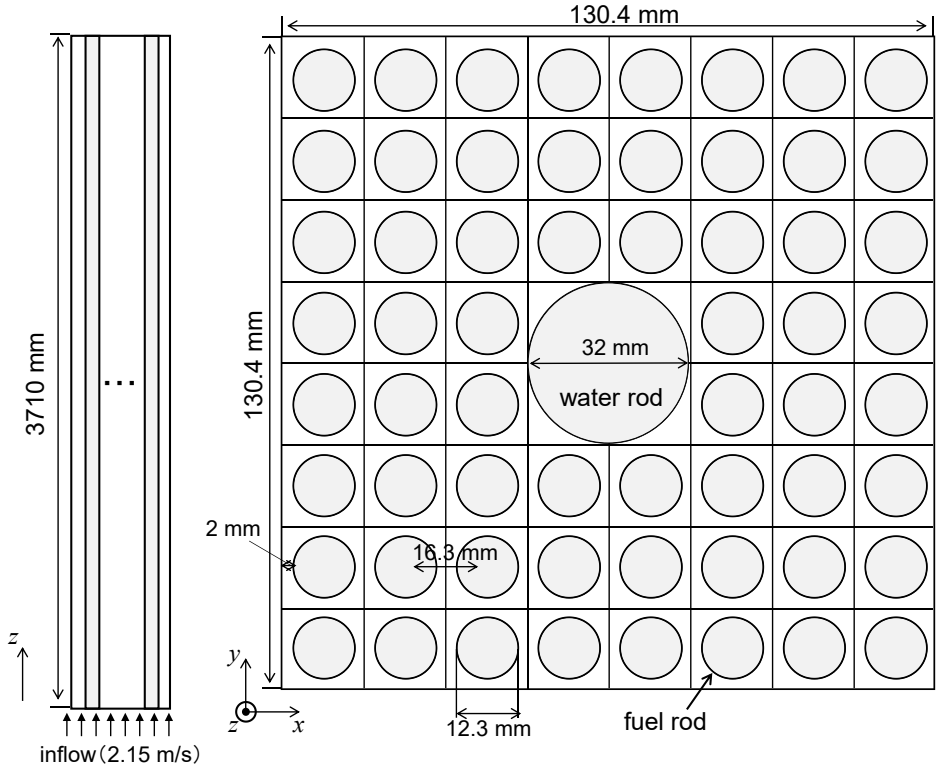


図 3.3 BWR8×8step-II 単一集合体

3.5. 実行スクリプト例

本節の末尾に MVP/JUPITER 連成計算の実行スクリプト例を示す。JAMPAN と MVP はローカル PC で実行し、JUPITER はスーパーコンピュータなどの別の PC で実行される。後者の PC のジョブは PBS 系のジョブスケジューラーを使用して投入される。

MVP/JUPITER 連成計算は `jampan/platform/interface/MvpJupiterRunner.py` にあるクラス `MvpJupiterRunner` によって実施される。`MvpJupiterRunner` は熱流動関連のクラス（`ThRunner`）のインスタンスと MVP 関連（`NeRunner`）のクラスのインスタンスをメンバ変数として持っており、MVP/JUPITER 連成計算を行う際はそれぞれをゲッターによって取得する（サンプルの 016、040 行目）。それぞれのインスタンスのセッターを用いて必要な値を設定した後、メソッド run を呼び出すことで計算が実行される。

以下は `ThRunner` のセッターについての説明であり、すべて必須項目である。

#017: `set_case_name(name)`

ケース名を指定する。ここで指定したケース名のディレクトリが自動生成され、その中に計算結果が格納される。

#018: `set_bundle(name)`

計算体系を指定する。現在（V.1.00.000）のところ“`stepII`”のみ対応している。

#019: `set_lib_path(name)`

参照入力が置かれたディレクトリ（3.4 節で示したフォルダのツリー構造における `reference`）のパスを指定する。

#020: `set_host(name)`

JUPITER を実行する計算機のホスト名を指定する。

#021: `set_user_name(name)`

JUPITER を実行する計算機のユーザー名を指定する。

#022: `set_shell_filename(name)`

PBS を用いて実行するスクリプトファイル名を指定する。

#024: `set_password(name)`

JUPITER を実行する計算機のパスワードを指定する。

#025: `set_workdir_path(name)`

JUPITER を実行する計算機における、JUPITER を実行するディレクトリを指定する。こ

こで指定したディレクトリをあらかじめ、その計算機上に作成しておかなければならない。

#026: `set_simulation_list(list)`

連成回数を示すリストを指定する。例えば、10 回計算するときは[0, 1, 2, 3, 4, 5, 6, 7, 8 ,9]を設定する。

#027: `set_th_condition_list(list)`

JUPITER の計算条件を指定する。二重のリストになっており、内側のリストは 3 つの要素からなり、一つ目の要素は入力項目、二つ目は単位、三つ目は値となっている。表 3.1 は入力項目とその意味を示す。

表 3.1 MVP/JUPITER 連成計算の入力項目

入力項目	意味
`analysis_exe_path`	JUPITER を実行する計算機上における JUPITER の実行ファイルまでのパス
`simulation_time`	連成間隔 （例 0.5 秒ごとに 5 回連成させる場合は[0, 0.5, 1.0, 1.5, 2.0]）
`mpi_process`	x, y, z 方向の並列数
`cell`	MPI による領域分割後のセル数（各方向の総セル数は `mpi_process` と `cell` の積）
`read_accuracy`	JUPITER のアウトプットの精度 （ON：倍精度、OFF：単精度）
`tout`	JUPITER のアウトプットの出力間隔

#036: `set_flag_water_density_timeavg(value)`

MVP が利用する水密度に対して時間平均を施すかを指定する。0 の場合は時間平均を適用せず、1 の場合は適用する。

以下は MVP 関連のセッターについての説明である。すべて必須項目である。

#040: `set_mvp64_exec(name)`

64 ビット版の MVP（mvp64 および mvp64.mpi）の実行ファイルまでのパスを指定する。32 ビット版の MVP（mvp および mvp.mpi）を使用する場合は、アクセッサメソッド `set_mvp_exec(name)`に指定する。

#041: `set_code(name)`

連成計算に使用する計算コードを指定する。MVP/JUPITER 連成計算の場合、[“JUPITER”,

“MVP”]を与える。

#045: set_bundle_type(name)

体系を指定する。BWR8×8step-II では、“8x8S2”を指定する。

#046: set_nhist(value)

MVP の計算のバッチ当たりのヒストリー数を指定する。指定した値は、MVP の入力変数 NHIST の値となる。

#047: set_dynamic_memory(value)

MVP の計算で確保するメモリを指定する。指定した値は、MVP の入力変数 DYNAMIC-MEMORY の値となる。

#050: set_nuclide_id_format(value)

MVP に渡す水の数密度の値の小数点以下の桁数を指定する。整数部分が 1 桁かつ指定した値が小数点以下の桁数となる指数表記に変換される。デフォルト値は 5 であるが、メモリ不足で計算が破綻する場合、小さな値を設定する。

#051: set_tempmt_format(value)

MVP に渡す温度の値の小数点以下の桁数を指定する。整数部分が 1 桁かつ指定した値が小数点以下の桁数となる指数表記に変換される。デフォルト値は 1 であるが、メモリ不足で計算が破綻する場合、小さな値を設定する。

#054: set_fuel_radius(value)

燃料ペレットの半径を指定する。単位は cm である。

#055: set_clad_radius(value)

被覆管の外径を指定する。単位は cm である。

#056: set_fuel_pin_pitch(value)

燃料棒のピッチを指定する。単位は cm である。

#059: set_water_rod_density_h1(value)

水ロッドの水領域の核種 H-1 の数密度を指定する。単位は 10^{24}atom/cm^3 である。

#060: set_water_rod_density_o16(value)

水ロッドの水領域の核種 O-16 の数密度を指定する。単位は 10^{24}atom/cm^3 である。

#061: set_water_rod_temperature(value)

水ロッドの温度を指定する。単位は K である。

#062: set_water_rod_inner_radius(value)

水ロッドの内径を指定する。単位は cm である。

#063: set_water_rod_outer_radius(value)

水ロッドの外径を指定する。単位は cm である。

#068: set_mvp_aindx(name)

MVP の任意温度中性子断面積のインデックスファイルのパスを指定する。

#069: set_mvp_mpi(name)

MPI のコマンド実行ファイルのパスを指定する。

#070: set_mvp_mpi_number(value)

並列数を指定する。

#074: set_fuel_power(value)

燃料棒一本当たりの発熱量を指定する。単位は W である。

#075 set_ne_calculation_class_name(name)

HDF5 の核側の計算グループ名を設定する。現状は `heat_power`（デフォルト値）と設定する必要がある。

#078 set_hdf5_path(name)

データコンテナのパスを指定する。

#079 set_mvp_file_dir(name)

MVP 関連ファイルの格納先のパスを指定する。

#080 set_mesh_num_x(value)

MVP における x 方向分割数を指定する。

#081 set_mesh_len_x(value)

MVP における分割領域の x 方向長さを指定する。単位は cm である。

#082 set_mesh_num_y(value)

MVP における y 方向分割数を指定する。

#083 set_mesh_len_y(value)

MVP における分割領域の y 方向長さを指定する。単位は cm である。

#084 set_mesh_num_z(value)

MVP における z 方向分割数を指定する。

#085 set_mesh_len_z(value)

MVP における分割領域の z 方向長さを指定する。単位は cm である。

```
000 import os
001 import subprocess as sp
002 import sys
003 sys.path.append("jampan のあるディレクトリパス") # path を設定した場合不要
004 from jampan.platform.interface.MvpJupiterRunner import MvpJupiterRunner
005 import getpass
006
007 if __name__ == "__main__":
008     mjr = MvpJupiterRunner()
009
010     dt_interbal = 1.0
011     interval = 11
012     simulation_list = [i for i in range(0,interval)]
013     simulation_time = [dt_interbal*i for i in range(0,interval+1)]
014
015     # ThRunner
016     mti = mjr.th_instance()
017     mti.set_case_name("stepII")
018     mti.set_bundle("stepII")
019     mti.set_lib_path("reference")
020     mti.set_host("*******")
021     mti.set_user_name("*******")
022     mti.set_shell_filename("PBS")
023     pw = getpass.getpass()
024     mti.set_password(pw)
025     mti.set_workdir_path("*********")
```

```
026     mti.set_simulation_list(simulation_list)
027     mti.set_th_condition_list([
028         ["analysis_exe_path",
029          "-",
030          "******/jupiter-double"],
031         ["simulation_time", "s", simulation_time],
032         ["mpi_process", "-", 2, 4, 16],
033         ["cell", "-", 64, 32, 231],
034         ["read_accuracy", "-", "OFF"],
035         ["tout","s",0.01]])
036     mti.set_flag_water_density_timeavg(1)
037 
038     # NeRunner
039     mni = mjr.ne_instance()
040     mni.set_mvp64_exec("/home/code/MVP-GMVP/mvp-
    3.0/bin/LINUXGLIBC/mvp64.mpi")
041     mni.set_code(["JUPITER", "MVP"])
042 
043     # NeInputMaker
044     nimi = mni.ne_input_maker_instance()
045     nimi.set_bundle_type("8x8S2")
046     nimi.set_nhist(20000)
047     nimi.set_dynamic_memory(1500000000)
048 
049     # Accuracy of Water density
050     nimi.set_nuclide_id_format(3)
051     nimi.set_tempmt_format(0)
052 
053     # Fuel Rod
054     nimi.set_fuel_radius(0.529) # [cm]
055     nimi.set_clad_radius(0.529 + 0.086)
056     nimi.set_fuel_pin_pitch(1.63)
057 
058     # Water Rod
059     nimi.set_water_rod_density_h1(5.03583E-02)  # number-density
060     nimi.set_water_rod_density_o16(2.51791E-02)
061     nimi.set_water_rod_temperature(549.1)  # [K]
062     nimi.set_water_rod_inner_radius(3.2/2-0.086)  # [cm]
063     nimi.set_water_rod_outer_radius(3.2/2)  # [cm]
064 
065     # NeCodeRunner
066     mncri = mni.ne_code_runner_instance()
067     mmri  = mncri.mvp_runner_instance()  # mmri = MvpRunner()
068     mmri.set_mvp_aindx("/home/code/MVP-GMVP/MVPlib64/neutron.art.J40.index")
069     mmri.set_mvp_mpi("/opt/intel/oneapi/mpi/2021.8.0/bin/mpiexec")
```

```
070     mmri.set_mvp_mpi_number(32)
071
072     # NeOutputReader
073     mnori = mni.ne_output_reader_instance()
074     mnori.set_fuel_power(71840.)
075     mnori.set_ne_calculation_class_name("heat_power")
076
077     # MvpJupiterRunner
078     mjr.set_hdf5_path(os.getcwd()  +  "/"  +  mti.case_name()  +  "/"  +
    "data_container.h5")
079     mjr.set_mvp_file_dir(os.getcwd() + "/" + mti.case_name())
080     mjr.set_mesh_num_x(16)
081     mjr.set_mesh_len_x(0.815)
082     mjr.set_mesh_num_y(16)
083     mjr.set_mesh_len_y(0.815)
084     mjr.set_mesh_num_z(24)
085     mjr.set_mesh_len_z(15.45833333333333)  # = 371./24
086
087     # Running!
088     sp.run(["mkdir", mti.case_name()])
089     mjr.run()
```

3.6. 出力データ

前節で示した実行スクリプトの実行結果を本節の末尾に示す。set_case_name()で指定したディレクトリが自動生成され（ここでは“stepII”）、その中に計算結果が保存される。data_container.h5 は JAMPAN データコンテナである。set_bundle_type()で指定した名前に”NT1”を加えたファイルが 3 種類出力される。それぞれ、ft06 は標準出力ファイル、ft30 は計算結果出力ファイル、inp は標準入力ファイルを示す。それぞれが連成計算の反復の数だけ出力される。stepII の中には反復の数だけ 1、2…とディレクトリが作成される。その中の input には計算に使用された JUPITER の入力と PBS 系の実行ファイルが格納されている。output には JUPITER の標準出力（STDOUT）と計算結果（vdata）が格納される。計算結果を整理した例は文献[5]に示されている。

```
stepII
├──── data_container.h5
├──── mvp8x8S2NT1.ft06
├──── mvp8x8S2NT1.ft30
├──── mvp8x8S2NT1.inp
├──── mvp8x8S2NT2.ft06
├──── mvp8x8S2NT2.ft30
```

```
├──── mvp8x8S2NT2.inp
├──── …
├──── 1
│      └thermal-hydraulics
│          ├──input
│          │   ├── input.txt
│          │   ├── param.txt
│          │   ├── geom.txt
│          │   ├── flags.txt
│          │   ├── plist.txt
│          │   ├── phys.vapor_7.07MPa.txt
│          │   ├── phys.water_7.07MPa.txt
│          │   ├── phys.waterrod.txt
│          │   ├── phys.UO2.txt
│          │   ├── phys.Zry.txt
│          │   ├── PBS_user
│          │   └── r_PBS_user
│          └ output
│              ├── STDOUT
│              └── vdata
├──── 2
…
```

4. JAMPAN による MVP/ACE-3D 連成計算について

4.1. 連成計算フロー

JAMPAN による MVP/ACE-3D 連成計算は、単一集合体規模を計算対象としており、ACE-3D による熱流動計算と MVP による中性子輸送計算を交互に実施することで進行する。ACE-3D は大型計算機で実行され、MVP は JAMPAN が実行されている計算機で実行される。前述の通り、JAMPAN では独自の HDF5 フォーマットのデータコンテナである JAMPAN データコンテナを介して連成が行われる。連成計算の具体的な手続きは下記の通りである。

1) ACE-3D の入力を大型計算機にアップロードし、ACE-3D の計算を実行。
2) ACE-3D の計算結果をダウンロードし、JAMPAN データコンテナに格納。
3) JAMPAN データコンテナにあるデータセットを用いて水密度を算出し、JAMPAN データコンテナに格納。
4) データコンテナにある水密度、温度を反映させた MVP の入力を作成。
5) MVP の計算を実行し、中性子反応率を算出。
6) 発熱量に変換し、JAMPAN データコンテナに格納。
7) JAMPAN データコンテナにある発熱量を反映した ACE-3D の入力を作成。
8) 1)～7）を設定した回数繰り返す。

なお、この一連の連成計算は JAMPAN 上で自動的に実行される。MVP/ACE-3D 連成計算では、必ず ACE-3D から開始する必要があるが、1 回目の ACE-3D の計算では、燃料棒が発熱しない断熱計算であり、沸騰も生じない。

4.2. 水密度の算出

JUPITER では、計算領域を 0.1 から 1 mm 程度のセルで分割するため、3.2 節で述べたように水密度を個体セルに対して外挿することが必要となるが、二流体モデルを採用している ACE-3D の格子は JUPITER に比べて粗い。そのため、JAMPAN では、ACE-3D 計算においてすべてのセルが流体を含むことを前提としている。そのため、ACE-3D では、すべてのセルにおいて以下のように、水密度$\rho_{i,j,k}$を算出する。

$$\rho_{i,j,k} = f_{i,j,k}\rho_{w,i,j,k} + (1 - f_{i,j,k})\rho_{v,i,j,k} \tag{4.2.1}$$

ここで、$\rho_{w,i,j,k}$、$\rho_{v,i,j,k}$は水および水蒸気の密度を示し、ACE-3D によって出力される。

4.3. 発熱量の補間

ACE-3D では、JUPITER 同様、バイナリファイルを読み込んで、発熱量を設定する。デー

タコンテナに格納された発熱分布から ACE-3D 用の発熱分布を取得するための補間方法は JUPITER の方法と同様である（3.3 節参照）。

4.4. 参照入力

JAMPAN では、あらかじめ用意された ACE-3D の参照入力が JAMPAN の実行スクリプトに応じて、自動的に編集される。したがって、ユーザーが JAMPAN の実行スクリプトを編集することで、計算条件を変更することができる。現時点（V.1.00.000）で、MVP/ACE-3D 連成計算について JAMPAN が対応している体系は図 4.1 に示される BWR8×8 単一集合体である。BWR を想定し、圧力は 7.07 MPa、温度は 549.15 K であり、下面から 2.15 m/s で水が流入する。

MVP/ACE-3D 連成計算を実行する際は、実行スクリプトと同じ階層に以下のような参照入力ディレクトリ（下のツリーでは `reference` というディレクトリだが、名前は任意である）を用意する必要がある。

```
reference
├─── input
│      └ACE3D
│         └ bundle8x8_bwr
│           ├── input
│           │     └ bundle8x8_bwr.txt
│           └ jcl
│              ├── bundle8x8_bwr.jcl
│              └── r_bundle8x8_bwr.jcl
└─── shell
       └ACE3D
          ├── PBS
          └── r_PBS
```

`reference/input/ACE3D/bundle8x8_bwr/input` に配置する参照入力は以下のディレクトリに用意されている。

`jampan/data/thermal-hydraulics/ref_input/input/ACE3D/bundle8x8_bwr`

また、`reference/shell/ACE3D` に配置する原子力機構が保有する大型計算機の PBS 系のジョブスクリプトは以下のディレクトリに用意されている。

`jampan/data/thermal-hydraulics/ref_input/shell/ACE3D`

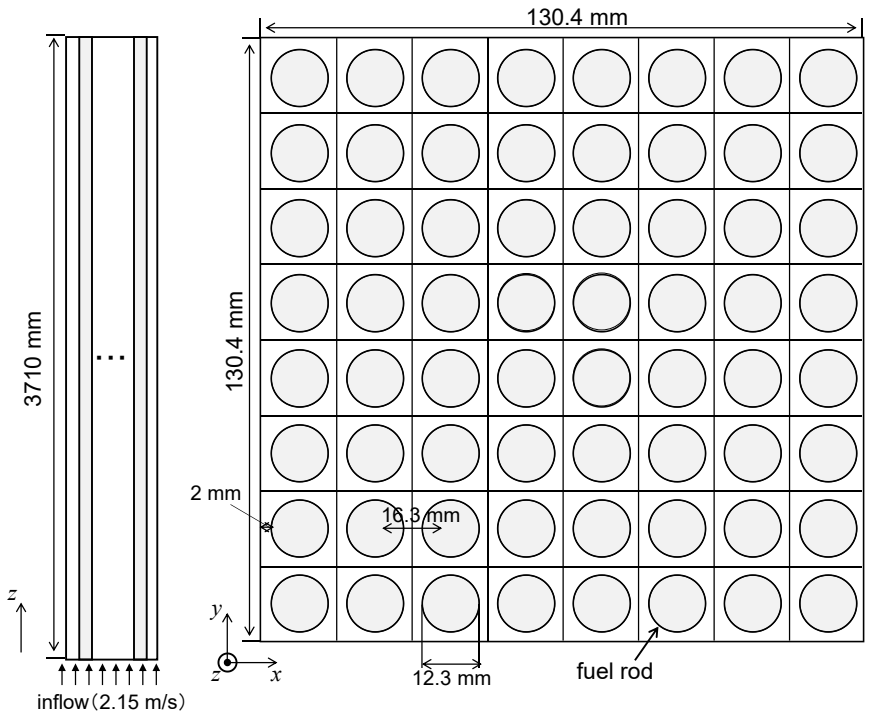


図 4.1 BWR8×8 単一集合体

4.5. 実行スクリプト例

本節の末尾に MVP/ACE-3D 連成計算の実行スクリプト例を示す。JAMPAN と MVP はローカル PC で実行し、ACE-3D はスーパーコンピュータなどの別の PC で実行される。後者の PC のジョブは PBS 系のジョブスケジューラーを使用して投入される。

MVP/ACE-3D 連成計算は `jampan/platform/interface/MvpThydraulicsRunner.py` にあるクラス `MvpThydraulicsRunner` によって実施される。`MvpThydraulicsRunner` は熱流動関連のクラス（`ThRunner`）のインスタンスと MVP 関連のクラス（`NeRunner`）のインスタンスをメンバ変数として持っており、MVP/ACE-3D 連成計算を行う際はそれぞれをゲッターによって取得する（サンプルの 020、044 行目）。それぞれのインスタンスのセッターを用いて必要な値を設定した後、メソッド `run` を呼び出すことで計算が実行される。ほとんどのセッターが MVP/JUPITER 連成計算で呼び出されたものと同様であるため、ここでは、重複しないセッターのみを示す。ただし、ACE3D は実行シェルに応じて MPI によるプロセス並列化が施され、入力に並列数を指定する項目はない。そのため、`set_th_condition_list()` の `mpi_process` には `1, 1, 1` を入力することとする。

#027: set_code ([name])

連成計算に使用するコードを指定。MVP/ACE-3D 連成計算の場合は[“ACE3D”, “MVP”]といったリストを作成する。

#086: set_channel_box_flag (bool)

計算領域の集合体にチャンネルボックスを挿入するのかを指定する。True の場合は挿入し、False の場合は挿入されない。

#087: set_channel_box_thick (bool)

チャンネルボックスの厚さを cm 単位で指定する。

#088: set_reflector_thick_side (value)

水ギャップの側面に定義する完全反射物質の厚さを cm 単位で指定する。

#089: set_reflector_thick_top (value)

水反射体の上部の完全反射物質の厚さを cm 単位で指定する。

#090: set_reflector_thick_bottom (value)

水反射体の下部の完全反射物質の厚さを cm 単位で指定する。

#091: set_water_gap_thick_side (value)

水ギャップの側面の厚さを cm 単位で指定する。

#092: set_water_gap_thick_top (value)

水反射体の上部の厚さを cm 単位で指定する。

#093: set_water_gap_thick_bottom (value)

水反射体の下部の厚さを cm 単位で指定する。

```
000 import os
001 import subprocess as sp
002 import sys
003 sys.path.append("jampan のあるディレクトリパス")   # path を設定した場合不要
004 from jampan.platform.interface.MvpThydraulicsRunner import (
005     MvpThydraulicsRunner
006     )
007 import getpass
008
```

```
009 if __name__ == "__main__":
010     mjr = MvpThydraulicsRunner()
011 
012     dt_interbal = 15
013     interval = 10
014     simulation_list = [i for i in range(0,interval)]
015     simulation_time = [dt_interbal*i for i in range(0,interval+1)]
016     print(simulation_list)
017     print(simulation_time)
018 
019     # ThRunner
020     mti = mjr.th_instance()
021     mti.set_case_name("bundle8x8_bwr")
022     mti.set_bundle("bundle8x8_bwr")
023     mti.set_lib_path("reference")
024     mti.set_host("*****")
025     mti.set_user_name("*****")
026     mti.set_shell_filename("PBS")
027     mti.set_code(["ACE3D", "MVP"])
028     pw = getpass.getpass()
029     mti.set_password(pw)
030     mti.set_pbs_flag(True)
031     mti.set_workdir_path("*****")
032     mti.set_simulation_list(simulation_list)
033     mti.set_th_condition_list([
034         ["simulation_time", "s", simulation_time],
035         ["mpi_process", "-", 1, 1, 1],
036         ["cell", "-", 16, 16, 240],
037         ["read_accuracy", "-", "OFF"],
038         ['output_file_name', '-', 'bundle8x8_bwr'],
039         ["tout", "s", 0.01]])
040     mti.set_solid_flag(1)
041     mti.set_flag_water_density_timeavg(2)
042 
043     # NeRunner
044     mni = mjr.ne_instance()
045     mni.set_mvp64_exec("/home/code/MVP-GMVP/mvp-
            3.0/bin/LINUXGLIBC/mvp64.mpi")
046     mni.set_code(["ACE3D", "MVP"])
047 
048     # NeInputMaker
049     nimi = mni.ne_input_maker_instance()
050     nimi.set_bundle_type("8x8")
051     nimi.set_nhist(20000)
052     nimi.set_dynamic_memory(1200000000)
053 
```

```
    # Accuracy of Water density
    nimi.set_nuclide_id_format(3)
    nimi.set_tempmt_format(0)

    # Fuel Rod
    nimi.set_fuel_radius(0.529) # [cm]
    nimi.set_clad_radius(0.529 + 0.086)
    nimi.set_fuel_pin_pitch(1.63)

    # NeCodeRunner
    mncri = mni.ne_code_runner_instance()
    mmri  = mncri.mvp_runner_instance()  # mmri = MvpRunner()
    mmri.set_mvp_aindx("/home/code/MVP-GMVP/MVPlib64/neutron.art.J40.index")
    mmri.set_mvp_mpi("/opt/intel/oneapi/mpi/2021.8.0/bin/mpiexec")
    mmri.set_mvp_mpi_number(20)

    # NeOutputReader
    mnori = mni.ne_output_reader_instance()
    mnori.set_fuel_power(71840.)
    mnori.set_ne_calculation_class_name("heat_power")

    # MvpJupiterRunner
    mjr.set_hdf5_path(os.getcwd()  +  "/"  +  mti.case_name()  +  "/"  +
"data_container.h5")
    mjr.set_mvp_file_dir(os.getcwd() + "/" + mti.case_name())
    mjr.set_mesh_num_x(16)
    mjr.set_mesh_len_x(0.815)
    mjr.set_mesh_num_y(16)
    mjr.set_mesh_len_y(0.815)
    mjr.set_mesh_num_z(24)
    mjr.set_mesh_len_z(15.45833333333333)  # = 371./24

    # Channel Box
    nimi.set_channel_box_flag(True)
    nimi.set_channel_box_thick(0.25)
    nimi.set_reflector_thick_side(1.)
    nimi.set_reflector_thick_top(0.)
    nimi.set_reflector_thick_bottom(0.)
    nimi.set_water_gap_thick_side(1.0)
    nimi.set_water_gap_thick_top(30.0)
    nimi.set_water_gap_thick_bottom(30.0)

    # Running!
    sp.run(["mkdir", mti.case_name()])
    mjr.run()
```

4.6. 出力データ

前節で示した実行スクリプトの実行結果を本節の末尾に示す。set_case_name()で指定したディレクトリが自動生成され（ここでは“stepII”）、その中に計算結果が保存される。data_container.h5 は JAMPAN データコンテナである。set_bundle_type()で指定した名前に“NT*”を加えたファイルが 3 種類出力される。それぞれ、ft06 は標準出力ファイル、ft30 は計算結果出力ファイル、inp は標準入力ファイルを示す。それぞれが連成計算の反復の数だけ出力される。stepII の中には反復の数だけ 1、2…とディレクトリが作成される。その中の input/input には ACE-3D の入力、input/jcl には ACE-3D の jcl ファイル、input/Shell には PBS 系の実行ファイルが格納される。output には計算結果が格納される。

```
bundle8x8_bwr
├── data_container.h5
├── mvp8x8NT1.ft06
├── mvp8x8NT1.ft30
├── mvp8x8NT1.inp
├── mvp8x8NT2.ft06
├── mvp8x8NT2.ft30
├── mvp8x8NT2.inp
├── …
├── 1
│     └thermal-hydraulics
│        ├input
│        │  ├ input
│        │  │   ├ bundle8x8_bwr.txt
│        │  │   └ heat_distribution
│        │  ├ jcl
│        │  │   └ bundle8x8_bwr.jcl
│        │  ├ Shell
│        │  │   ├ PBS_user
│        │  │   └ r_PBS_user
│        └ output
├── 2
…
```

参考文献

[1] K. Smith, Reactor Core Methods, Invited lecture at the M&C 2003, Apr. 6-10, Gatlinburg, TN, USA, 2003.

[2] Y. Nagaya, K. Okumura, T. Sakurai and T. Mori, MVP/GMVP Version 3 : General purpose Monte Carlo codes for neutron and photon transport calculations based on continuous energy and multigroup methods, JAEA-Data/Code 2016-018, 2017, 421p.

[3] 山下晋、多相多成分詳細熱流動解析コード JUPITER の開発、JAEA-Data/Code 2025-002, 2025, 243p.

[4] 宇田川豊、山内紹裕、北野剛司、天谷政樹、燃料挙動解析コード FEMAXI-8 の開発―軽水炉燃料挙動モデルの改良と総合性能の検証―、JAEA-Data/Code 2018-016, 2019, 79p.

[5] T. Kamiya, T. Nagatake, A. Ono, K. Tada, R. Kondo, Y. Nagaya and H. Yoshida, Neutronics/thermal-hydraulics coupling simulation using JAMPAN in a single BWR assembly, Mechanical Engineering Journal, Vol. 12, No. 4, 24-00461, 2025.