ROM で捉える炉物理

名古屋大学 遠藤 知弘

1. はじめに

原子炉物理学の数値解析では、設計対象の実効増倍率k_{eff}や全中性子束分布$\phi(\vec{r},E)$などの炉心特性を数値計算により予測することが重要なテーマとなっている。これら予測を行うため、ボルツマンの中性子輸送方程式[1], [2]に基づくことで、体系内を飛び交う中性子の集団的な挙動をモデル化し、さまざまな数値計算手法と計算機を駆使することで炉心核特性の数値解を得ることができる。

仮に、計算コストを無限に費やすことができれば、高い正確度(accuracy)の予測も可能であろう。しかし現実世界では、その時々で利用可能な計算機の性能に限りがあり、予測のために割くことができる時間にも制限がある。原子炉物理学の数値解析分野では、比較的よい正確度で高速に中性子輸送計算の解を得ることができるよう、決定論的な数値解析手法(空間メッシュや中性子エネルギー・飛行方向を離散化、あるいは関数展開して数値解を得る手法)において、様々な工夫がなされてきた。このような過去に編み出されてきた工夫は、炉物理計算コードの開発を通じて研究者が培ってきた「工学的知見に基づいた縮約モデル(Reduced Order Model, ROM)」と呼んでもよいだろう。

ROM については、過去の炉物理夏期セミナーのトピック[3]として既に扱われているが、本資料による講義・演習では以下を達成することを主目的とする。

①	原子炉物理学分野で過去から現在までに活用されてきた Reduced Order Model について概要を学ぶ。
②	近年、原子炉物理学分野で利用されつつある Reduced Order Model について学ぶ。
③	演習問題を通じて、②の基礎技術を習得する。

2. 懸賞問題

Figure 1 で示すように、厚さH (cm)の 1 次元平板の均質体系を考えたとしよう。

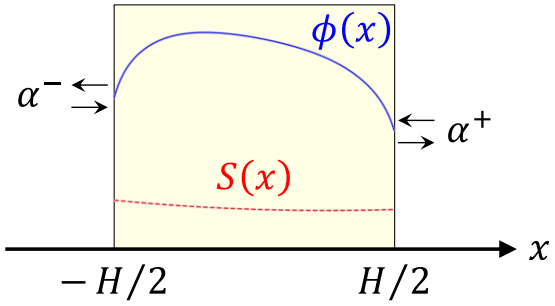

Figure 1 計算体系

この体系に対する計算条件として、以下で挙げる入力パラメータを与えたとする。

✓ 巨視的吸収断面積：Σ_a (1/cm)。

✓ 巨視的散乱断面積：Σ_s (1/cm)。
✓ 等方中性子源：$S(x)$ (neutrons/cm^3)、ただし 2 次関数で表現できる形状とする。
✓ 領域左側と右側の境界条件：アルベド境界条件α^-, α^+。

以上の入力パラメータ$\Sigma_a, \Sigma_s, S(x), \alpha^-, \alpha^+$に対して任意の値が与えられた時に、領域内の位置$x$における全中性子束$\phi(x)$の数値を高速かつ正確に求める方法を開発せよ。

以上の課題を解決するため、以下 3 名のエンジニアが名乗りをあげた。
成田さん：数学がそこそこ得意。紙と鉛筆が友達。→第 3 章
海野さん：プログラミングが得意。→第 4 章
辻さん：ROM を鋭意勉強中。→第 5 章

以降では、各 3 名のエンジニアの問題解決の軌跡を追いながら、原子炉物理分野における「ROM」について様々な観点から学ぶこととしよう。勘所となる点については、💡にて強調しておく。

3. 解析的なアプローチ

3.1. 中性子輸送方程式の解析解

成田は問題を一瞥すると、おもむろに近くにあった裏紙に数式を記述し始めた。

「中性子が飛んでいる飛行方向ベクトル$\vec{\Omega} = (\mu, \eta, \xi)$に沿って、原子核と衝突した中性子は吸収や別の方向への散乱で減少し、源から等方向に放出された中性子のぶんだけ増加する。」

成田は、体系内に飛び交う中性子の様子を頭に思い描きながら、中性子エネルギー1 群のボルツマン中性子輸送方程式を、以下のように記述した。

$$\mu \frac{\partial \psi}{\partial x} + \eta \frac{\partial \psi}{\partial y} + \xi \frac{\partial \psi}{\partial z} + \Sigma_t \psi(x, y, z, \mu, \eta, \xi) = \frac{\Sigma_s \phi(x, y, z) + S(x, y, z)}{4\pi} \quad (1)$$

$\psi(x, y, z, \mu, \eta, \xi)$ ：位置(x, y, z)における飛行方向(μ, η, ξ)の角度中性子束 (neutrons/cm^2/s/sr)。
$\phi(x, y, z)$ ：位置(x, y, z)における全中性子束 (neutrons/cm^2/s/)。
Σ_t ：巨視的全断面積$\Sigma_t = \Sigma_a + \Sigma_s$ (1/cm)。

「今回の問題は、1 次元平板体系なので空間変数はxのみで済む。x軸周りの回転対称性もあるから、中性子の飛行方向もx軸と成す極角θのみで済むだろう。飛行方向の依存性についても、x軸と対する方向余弦$\mu = \cos\theta$のみ考えれば良いはず。」

成田は、解くべき計算体系の対称性に注目して、解くべき中性子輸送方程式の変数の次元を減らして、以下のように記述し直した。

$$\mu\frac{\partial\psi}{\partial x}+\Sigma_{\mathrm{t}}\psi(x,\mu)=\frac{\Sigma_{\mathrm{s}}\phi(x)+S(x)}{2} \tag{2}$$

$\psi(x,\mu)$: 位置(x)におけるx軸との方向余弦μの角度中性子束、x軸周りで対称性があるため、方位角$0\le\varphi\le2\pi$で角度積分した量。Figure 2 のような円錐形状を思い描くとよい。

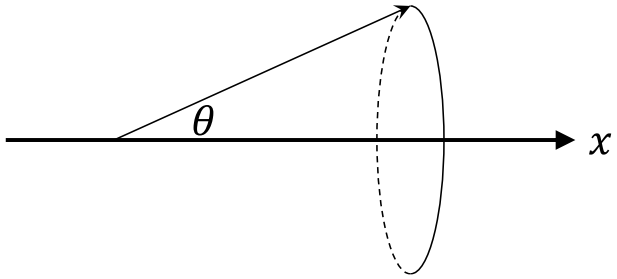

Figure 2 角度中性子束$\psi(x,\mu)$の概念図

> 対称性に応じて、変数の数そのものを減らすことが大切である。1 次元体系で解ける問題を、わざわざ 2 次元体系や 3 次元体系で解くべきではない。
>
> 今回のように 1 次元平板体系の場合には分かりやすいが、例えば「対称性のある球体系」を考えた場合には、位置が$\vec{r}=(x,y,z)$と 3 次元で表現できるからといって、短絡的に「3 次元xyz座標系」の問題を考えるのではなく、原点からの距離rのみに依存すると考え「1 次元の球体系」に基づいて問題を解き始めよう。

式(2)であれば、紙と鉛筆で解析解が導出できそうだ。中性子がx方向の正/負側どちら向きに飛行しているかによって、場合分けする必要がある点に注意しながら、成田は角度中性子束$\psi(x,\mu)$の解析解を以下のように導出した。

$$\psi(x,\mu)=\begin{cases}\psi(-H/2,\mu)\mathrm{e}^{-\frac{\Sigma_{\mathrm{t}}(x+H/2)}{\mu}}+\displaystyle\int_{-H/2}^{x}\left(\Sigma_{\mathrm{s}}\phi(x')+S(x')\right)\mathrm{e}^{-\frac{\Sigma_{\mathrm{t}}(x-x')}{\mu}}\frac{1}{\mu}dx' & \mu>0\\ \dfrac{\Sigma_{\mathrm{s}}\phi(x)+S(x)}{2\Sigma_{\mathrm{t}}} & \mu=0\\ \psi(H/2,\mu)\mathrm{e}^{-\frac{\Sigma_{\mathrm{t}}(H/2-x)}{|\mu|}}+\displaystyle\int_{x}^{H/2}\left(\Sigma_{\mathrm{s}}\phi(x')+S(x')\right)\mathrm{e}^{-\frac{\Sigma_{\mathrm{t}}(x'-x)}{|\mu|}}\frac{1}{|\mu|}dx' & \mu<0\end{cases} \tag{3}$$

全中性子束$\phi(x)$の解析解は、角度中性子束$\psi(x,\mu)$を全立体角、今回の場合には$-1\le\mu\le1$の範囲で積分すればよい。

$$\phi(x)\equiv\int_{-1}^{1}\psi(x,\mu)\,d\mu \tag{4}$$

ここまで数式を書き下すと、成田は気がついた。「アルベド境界条件α^{-},α^{+} を考慮できていなかった」と。x方向負(左)向きあるいは正(右)向きの部分中性子流を$J^{-}(x),J^{+}(x)$とすれば、アルベド境界条件は以下のように表現できる。

$$\alpha^{-}=\frac{J^{+}(-H/2)}{J^{-}(-H/2)} \tag{5}$$

$$\alpha^{+} = \frac{J^{-}(H/2)}{J^{+}(H/2)} \tag{6}$$

$$J^{-}(x) \equiv \int_{-1}^{0} |\mu|\, \psi(x,\mu)\, d\mu \tag{7}$$

$$J^{+}(x) \equiv \int_{0}^{1} \mu\, \psi(x,\mu)\, d\mu \tag{8}$$

成田は筆を擱き、もう一度問題に目を配り、深く溜息をついた。「はたして、領域内の全中性子束$\phi(x)$の数値を計算できるのか？」と。式(3), (5)–(8)を何度も眺めたが、自己撞着(self-consistent)な問題となっており、成田が有している数学力では、これ以上の式変形により厳密な解析解を得ることはできなかった。

3.2. 中性子拡散方程式の解析解

成田は、厳密さへのこだわりをいったん捨て、全中性子束$\phi(x)$を近似的だが簡便に解くことができないか、これまで学んだ知識を掘り起こすこととした。「拡散近似」という言葉を思い出すと、おもむろに角度中性子束$\psi(x,\mu)$を、なにやら奇怪な関数で展開し始めた。

$$\psi(x,\mu) = \sum_{l=0}^{\infty} \frac{2l+1}{2} f_l(x) P_l(\mu) \tag{9}$$

上式において、$P_l(\mu)$はl次の Legendre 多項式、$f_l(x)$は対応する展開係数をそれぞれ表している。Legendre 多項式$P_l(\mu)$は$-1 \le \mu \le 1$の範囲で定義される関数であり、0 次から 3 次までの多項式を以下に例示するとともに、それら多項式の分布を Figure 3 に図示する。

$$P_0(\mu) = 1 \tag{10}$$

$$P_1(\mu) = \mu \tag{11}$$

$$P_2(\mu) = \frac{1}{2}(3\mu^2 - 1) \tag{12}$$

$$P_3(\mu) = \frac{1}{2}(5\mu^3 - 3\mu) \tag{13}$$

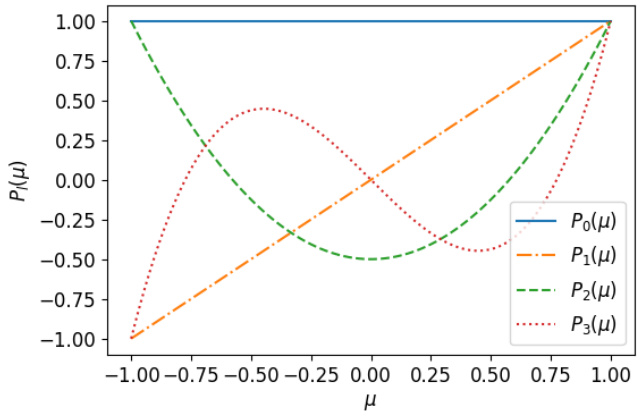

Figure 3 Legendre 多項式$P_l(\mu)$

Fourier 級数展開において三角関数が満足している性質と同じように、Legendre 多項式$P_l(\mu)$も以下で示すような「直交条件」を満足している。

$$\int_{-1}^{1} P_l(\mu)P_{l'}(\mu)d\mu = \frac{2}{2l+1}\delta_{ll'} \tag{14}$$

$\delta_{ll'}$：クロネッカーのデルタ

式(14)の直交性を利用することで、l次の展開係数$f_l(x)$は以下のように求めることができる。

$$f_l(x) = \int_{-1}^{1} P_l(\mu)\psi(x,\mu)d\mu \tag{15}$$

ここで成田は、式(15)において$l = 0,1$の時を考えることで、車輪の再発明ではあるものの、以下の閃きを得たのであった。

$$f_0(x) = \int_{-1}^{1} P_0(\mu)\psi(x,\mu)d\mu = \int_{-1}^{1} \psi(x,\mu)d\mu \equiv \phi(x) \tag{16}$$

$$f_1(x) = \int_{-1}^{1} P_1(\mu)\psi(x,\mu)d\mu = \int_{-1}^{1} \mu\,\psi(x,\mu)d\mu \equiv J(x) \tag{17}$$

すなわち、角度中性子束$\psi(x,\mu)$を$l = 0,1$までの Legendre 多項式で展開すれば、全中性子束$\phi(x)$と正味の中性子流$J(x)$を使って、以下のように表現することができる。

$$\psi(x,\mu) \approx \frac{1}{2}\phi(x) + \frac{3}{2}J(x)\mu \tag{18}$$

式(18)から分かることは、①正味の中性子流$J(x) = 0$であれば等方的に中性子が飛び交っており、②正/負向きの中性子流$J(x)$が生じている場合には前方/後方に飛行方向分布が偏っている、ということだ(Figure 4)。さらに、式(7), (8)で定義した部分中性子流$J^-(x), J^+(x)$に、式(18)の近似を代入してμについて積分すれば、$J^-(x), J^+(x)$も以下のように表現できる。

$$J^-(x) \approx \frac{1}{4}\phi(x) - \frac{1}{2}J(x) \tag{19}$$

$$J^+(x) \approx \frac{1}{4}\phi(x) + \frac{1}{2}J(x) \tag{20}$$

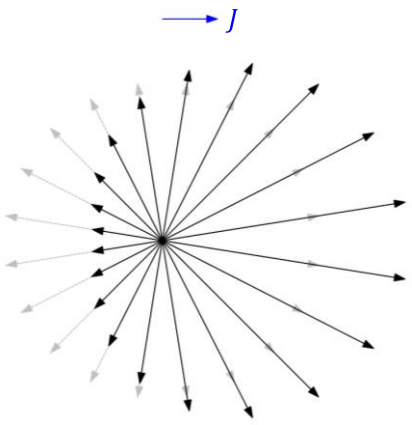

Figure 4 正味の中性子流Jによる角度中性子束$\psi(\mu)$の偏り

さて、下拵えは十分だ。本来解きたい厳密な中性子輸送方程式(2)に、式(18)を代入し両辺を$-1 \leq \mu \leq 1$の範囲で積分すれば、以下で示す関係式(中性子バランス式)を得ることができる。

$$\frac{dJ}{dx} + \Sigma_{\mathrm{a}}\phi(x) = S(x), \quad \therefore \frac{dJ}{dx} + \Sigma_{\mathrm{t}}\phi(x) = \Sigma_{\mathrm{s}}\phi(x) + S(x) \tag{21}$$

さらに、中性子輸送方程式(2)に式(18)を代入した上で、両辺に$P_1(\mu) = \mu$をかけた上で$-1 \leq \mu \leq 1$の範囲で積分することで、以下の関係式も得ることができる。

$$\frac{1}{3}\frac{d\phi}{dx} + \Sigma_{\mathrm{t}} J(x) = 0 \tag{22}$$

式(22)を「$J(x) =$」の形で変形すれば、全中性子束の勾配に比例して正味の中性子流が生じるという、いわゆる「Fick の拡散法則[4]」と呼ばれる関係式を導出ことができる。

$$J(x) = -D\frac{d\phi}{dx} \tag{23}$$

$$D = \frac{1}{3\Sigma_{\mathrm{t}}} \tag{24}$$

D：拡散係数 (cm)

あとは式(23)を式(21)に代入することで、中性子の飛行方向に関する変数を明示的に取り扱うことなく、簡易に全中性子束を求めることができる「中性子拡散方程式」を導出することができる。

$$-D\frac{d^2\phi}{dx^2} + \Sigma_{\mathrm{a}}\phi(x) = S(x) \tag{25}$$

💡 角度中性子束のルジャンドル多項式P_1成分までを考慮した「中性子拡散方程式」では、角度中性子束$\psi(\mu)$の飛行方向μ依存性を、①全中性子束ϕと②正味の中性子流Jの 2 成分のみで、効率良く次元削減している。炉物理計算における核反応率や中性子バランスを考える上で重要な量となっているϕ, Jが、2 つのルジャンドル多項式$P_0(\mu), P_1(\mu)$ の展開係数に相当しており、実に巧妙な方法と言えよう。

これに気を良くした成田は、「直交関数系」を過信してしまう。「全中性子束$\phi(x)$も Fourier 級数展開、すなわち三角関数で展開すれば、中性子拡散方程式(25)を効率良く解けるのではないか？」と。

$$\phi(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\left(\frac{n\pi x}{H/2}\right) + b_n \sin\left(\frac{n\pi x}{H/2}\right)\right) \tag{26}$$

式(26)を式(25)に代入し、三角関数をかけてxについて積分してみたものの、計算が複雑すぎる

上に、少ない展開次数では$\phi(x)$の空間分布に物理的に不自然な振動が生じてしまい、残念ながら成田の手には負える手法とはならなかった。

> 💡 三角関数のように、数学的には汎用性のある直交関数系が、効率良い計算を実現するために必ずしも有益とは限らない。展開することで効率良く次元圧縮したい場合には、解くべき対象体系の特徴を適切に捉えた関数系(ないし基底)を利用する必要がある。

これに懲りた成田は、中性子拡散方程式(25)を「2 階の非斉次常微分方程式」と捉え直して、再度、裏紙に解析解を解き始めた。まず固定中性子源$S(x)$は任意の 2 次関数で与えられることから、式(25)を以下のように書き直した。

$$-D\frac{d^2\phi}{dx^2}+\Sigma_{\mathrm{a}}\phi(x)=ax^2+bx+c \tag{27}$$

非斉次方程式(27)の特殊解$\phi^*(x)$は、発見的な方法ながら以下のように求めることができる。

$$\phi^*(x)=\frac{ax^2+bx+2\frac{D}{\Sigma_{\mathrm{a}}}a+c}{\Sigma_{\mathrm{a}}} \tag{28}$$

また、式(27)の斉次方程式

$$-D\frac{d^2\phi}{dx^2}+\Sigma_{\mathrm{a}}\phi(x)=0 \tag{29}$$

の独立な解を求めることで、非斉次方程式(27)の一般解は以下のように求めることができる。

$$\phi(x)=A\mathrm{e}^{x/L}+C\mathrm{e}^{-x/L}+\frac{ax^2+bx+2aL^2+c}{\Sigma_{\mathrm{a}}} \tag{30}$$

$$L\equiv\sqrt{D/\Sigma_{\mathrm{a}}} \tag{31}$$

L：拡散距離 (cm)

以下で示すように、式(30)右辺の定数A,Cは、左/右側の境界面のアルベド境界条件を与えることで求めることができる。

$$\alpha^-=\frac{\frac{1}{4}\phi(-H/2)+\frac{1}{2}J(-H/2)}{\frac{1}{4}\phi(-H/2)-\frac{1}{2}J(-H/2)} \tag{32}$$

$$\alpha^+=\frac{\frac{1}{4}\phi(H/2)-\frac{1}{2}J(H/2)}{\frac{1}{4}\phi(H/2)+\frac{1}{2}J(H/2)} \tag{33}$$

具体的な計算は省略するが、上式に Fick の拡散法則(式(23))と式(30)を代入することで、定数A,Cに関する連立方程式を得ることができる。結果として、以下で示す連立方程式の解析解に入力パラメータ($\Sigma_{\mathrm{a}},D,\alpha^-,\alpha^+$など)を代入することで定数$A,C$を決定することができる。

$$\begin{pmatrix} A \\ C \end{pmatrix} = \begin{pmatrix} \left\{(1-\alpha^-) - 2(1+\alpha^-)\frac{D}{L}\right\}\mathrm{e}^{-\frac{H}{2L}} & \left\{(1-\alpha^-) + 2(1+\alpha^-)\frac{D}{L}\right\}\mathrm{e}^{\frac{H}{2L}} \\ \left\{(1-\alpha^+) + 2(1+\alpha^+)\frac{D}{L}\right\}\mathrm{e}^{\frac{H}{2L}} & \left\{(1-\alpha^+) - 2(1+\alpha^+)\frac{D}{L}\right\}\mathrm{e}^{-\frac{H}{2L}} \end{pmatrix}^{-1} \cdot \begin{pmatrix} -(1-\alpha^-)\dfrac{\frac{aH^2}{4} - \frac{bH}{2} + (2aL^2 + c)}{\Sigma_\mathrm{a}} + 2(1+\alpha^-)L^2(-aH+b) \\ -(1-\alpha^+)\dfrac{\frac{aH^2}{4} + \frac{bH}{2} + (2aL^2 + c)}{\Sigma_\mathrm{a}} - 2(1+\alpha^+)L^2(aH+b) \end{pmatrix} \tag{34}$$

こうして、手計算で解析解が得ることができるよう解くべき問題を簡略化することで、成田は中性子拡散方程式の数値解を高速に得ることができる解析解を導出することができた。

https://colab.research.google.com/github/hyd3nekosuki/RPDsummer2024/blob/main/1DSlabNodalDiff2024RPDsmr.ipynb

💡3 次元炉心計算で活用されている「近代ノード法[5]」では、中性子拡散方程式の解析解を利用することで、粗い空間メッシュ分割でも離散化誤差の少ない数値解が得られるよう工夫している。このとき、連立方程式で解くべき未知数としては境界面で与えられる「境界条件の数」まで次元削減できる。

なお、3 次元体系に対して近代ノード法を利用する場合には、3 次元のまま解析解を求めるわけではない。例えば、注目している方向以外の方向(本資料の場合にはy,z方向)の中性子漏洩項を隣接メッシュの情報に基づいて 2 次関数形で表現し、注目するx,y,z方向のみで次元削減した 1 次元体系の拡散方程式の形に変形する。こうして次元削減したx,y,z方向の解析解を利用し、それらの解を合成することで効率化を図っている。

4. 炉物理計算手法を駆使したアプローチ

4.1. 確率論的手法

海野は問題を見るや、おもむろにノート PC を起動し、インターネットから Google Colaboratory [6]にアクセスした。

1 次元平板体系のモンテカルロ計算コード：https://colab.research.google.com/github/hyd3nekosuki/RPDsummer2024/blob/main/1DSlabMonteCarlo2024RPDsmr.ipynb

「これからの時代は、モンテカルロ計算を習得しておく必要があるよな」。海野は、夜な夜な OpenMC の計算理論[7]を勉強しながら、自身の理解を深めるために、Python を用いて 1 次元平板体系のエネルギー1 群の非アナログ・モンテカルロ固定源計算コード[8]を自作していた。「この前作成したモンテカルロコードを少し手直しすれば、この問題も楽に解決できるだろう」と、以下 2 つのコード修正を行った。

① 2 次関数の固定中性子源$S(x)$に基づいて、棄却法[9]により初期中性子源を発生するように修正した。まず、中性子源$S(x)$を以下式のような 2 次関数で表現したとする。

$$S(x) = ax^2 + bx + c \tag{35}$$

なお、$-H/2 \le x \le H/2$の範囲で、常に中性子源$S(x) \ge 0$が正となるように、係数a, b, cを与える点に注意する必要がある。棄却法では、まず$-H/2 \le x \le H/2$の範囲における$S(x)$の最大値$S_{\max}$を調べる。今回の場合には、①両端のx座標$x = \pm H/2$と、②2 次関数の頂点x座標$x_v = -b/(2a)$が$-H/2 \le x \le H/2$に存在するか調べ、それらx座標位置における$S(x)$の最大値を調べることで$S_{\max}$を調べることができる。その後、2 つの独立な一様乱数$0 \le \xi \le 1$と$-H/2 \le x_s \le H/2$を発生させて、以下の条件式を満足するようなx_sを採用することで、2 次関数$S(x)$に基づいて中性子の発生位置x_sをサンプリングすることができる。

$$ax_s^2 + bx_s + c \le \xi\, S_{\max} \tag{36}$$

② アルベド境界条件については、簡略化して取り扱うことした。具体的には、「境界面から流出した中性子が、アルベド値の割合だけ正反対向きに反射される」とみなして、境界面まで到達した中性子の重み$w(x, \mu)$を以下のように変化させる処理を実装した。

$$w(-H/2, -\mu) = \alpha^{-} w(-H/2, \mu), \qquad \text{if } \mu < 0 \tag{37}$$

$$w(H/2, -\mu) = \alpha^{+} w(H/2, \mu), \qquad \text{if } \mu > 0 \tag{38}$$

「試しの計算として、まずは追跡する中性子数は 100,000 個で良いかな？」と𝕏で呟くと、海野は Google Colab 上で改良したモンテカルロ計算コードの計算を流し始めた。Python で作成したプログラムだから致し方ないのだろうか、なかなか計算結果が返ってこない。ようやく返ってきた collision estimator による全中性子束$\phi(x)$ の計算結果(Figure 5)を見て、海野は落胆した。

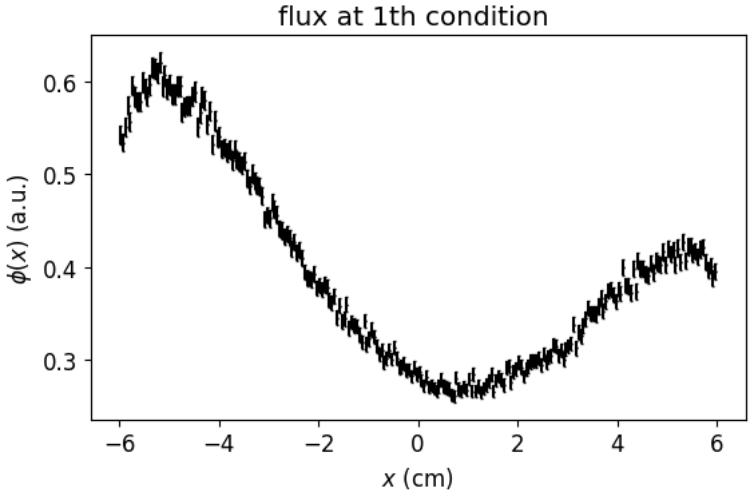

Figure 5 モンテカルロ固定源計算による全中性子束の計算例
(誤差バーは統計的不確かさ 1σ に相当)

「統計的不確かさによるバラつきが、想像以上に大きいな…」。海野は、過去に学んだ「中心極限定理[10]」について思い出した。今回のようなモンテカルロ固定源計算の場合には、中性子源から発生した 1 個 1 個の中性子は互いに独立である。この場合、collision estimator で得られた中性子束の平均値$\bar{\phi}$の相対統計的不確かさ$\sigma_{\bar{\phi}}$は、中心極限定理に基づいて以下のように推定することができる。

$$\sigma_{\bar{\phi}} = \frac{\sigma_{\phi}}{\sqrt{N}} \tag{39}$$

$$\sigma_{\phi} = \frac{1}{N-1}\sum_{i=1}^{N}\left(\phi_i - \frac{1}{N}\sum_{j=1}^{N}\phi_i\right) \tag{40}$$

ϕ_i：i番目のヒストリーで得られた中性子束。
N：ヒストリー数。先の例では$N = 100{,}000$。

式(39)に基づいて考えれば、モンテカルロ計算の統計的不確かさを 10 分の 1 まで低減したければ、100 倍多くしたヒストリー数$100 \times N$ものモンテカルロ計算を実施しなければならない。もう一度、$\phi(x)$の計算結果と問題に目を配ると、ノート PC を閉じて深く溜息をついた。「はたして、細かく分割した全中性子束$\phi(x)$を統計精度良く高速に計算できるのか？」と。

💡 位置$\vec{r}$や飛行方向$\vec{\Omega}$の取り扱いについて近似の無い、モンテカルロ計算は高い正確度(high accuracy)の数値解を得ることができる計算手法である。しかし、乱数を利用した計算手法の宿命として、統計的不確かさが生じざるを得ない。追跡する中性子ヒストリー数Nが不十分な場合には、正確度が高くても低い統計精度(low precision)の計算結果となることを肝に銘じよう。

全中性子束の空間分布$\phi(x)$を細かい空間メッシュで微細に調べようとすればするほど、統計的不確かさも大きくならざるを得ない。モンテカルロ計算を利用する場合には、「許容可能な統計精度」を念頭においた上で、中心極限定理に基づいて考え、実用的な時間内で計算結果を得ることができるのか、良く考えよう。

4.2. 決定論的手法による中性子拡散計算

海野は、モンテカルロ計算に固執することは止め、何か良い方法がないか、JAEA から入手した過去の夏期セミナーテキスト[11]に目を通すこととした。

「なになに、高速に炉物理計算を実施したい場合には『決定論的手法』を利用すれば良いのか📝」。決定論的手法のうち、中性子拡散計算コードについては、過去の炉物理夏期セミナーの講義テキスト(第 36 回[12] 第 50 回[13]、第 52 回[14])で何度も解説がなされている。海野は、これまでに公開されてきた集合知を最大限活用して、中性子拡散計算コードの作成に取り組み始めた。

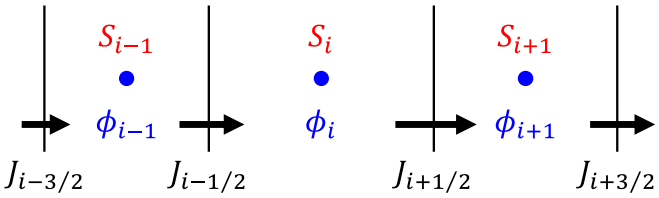

Figure 6 有限体積法による 1 次元平板体系の離散化

固定中性子源$S(x)$を与えた中性子拡散方程式(25)に対して、Figure 6 で示したように空間変数xについて空間メッシュで細かく分割した「有限体積法」に基づくことで、以下の差分式を得ることができる。

$$\frac{J_{i+1/2}-J_{i-1/2}}{\Delta x_i}+\Sigma_{\mathrm{a},i}\phi_i=S_i \tag{41}$$

$$J_{i+1/2}\approx\frac{-2D_iD_{i+1}}{D_i\Delta x_{i+1}+D_{i+1}\Delta x_i}(\phi_{i+1}-\phi_i) \tag{42}$$

上式において、下添字iはメッシュ位置(整数値)、境界面位置(分数値)をそれぞれ表している。

ただし、$i=1,NX$番目の空間メッシュ(体系の一番左/右側の空間メッシュ)については、それら空間メッシュの左隣/右隣にこれ以上空間メッシュがない、特殊な状況となっている。今回の場合には、アルベド境界条件に基づいて正味の中性子流$J_{1/2},J_{NX+1/2}$を求めれば良い。そこで海野は、まずは$i=1$番目の空間メッシュについて、拡散計算で与えるアルベド境界条件を導出することとした。$i=1$番目の空間メッシュにおける全中性子束ϕ_1と、左側アルベド境界面の表面中性子束$\phi_{1/2}$を用いて、Fick の拡散法則(式(23))の中性子束勾配$d\phi_1/dx$を近似することで、左側境界面における正味の中性子流$J_{1/2}$は以下のように求めることができる。

$$J_{1/2}\approx-D_1\frac{\phi_1-\phi_{1/2}}{\Delta x_1/2} \tag{43}$$

ここで、式(32)で示したように、左側アルベド境界条件が表面中性子束$\phi_{1/2}$と正味の中性子流$J_{1/2}$を用いて表現できることに注目する。

$$\alpha^-=\frac{\frac{1}{4}\phi_{1/2}+\frac{1}{2}J_{1/2}}{\frac{1}{4}\phi_{1/2}-\frac{1}{2}J_{1/2}}=\frac{\phi_{1/2}+2J_{1/2}}{\phi_{1/2}-2J_{1/2}} \tag{44}$$

式(44)を式変形することで、$\phi_{1/2}$は$J_{1/2}$とα^-を用いて以下のように表現することができる。

$$\phi_{1/2}=-2\frac{1+\alpha^-}{1-\alpha^-}J_{1/2} \tag{45}$$

さらに、式(45)右辺に式(43)を代入して整理することで、$\phi_{1/2}$はϕ_1とα^-を用いて以下のように表現することもできる。

$$\phi_{1/2}=\frac{2(1+\alpha^-)D_1}{(1-\alpha^-)\frac{\Delta x_1}{2}+2(1+\alpha^-)D_1}\phi_1 \tag{46}$$

結果として、正味の中性子流$J_{1/2}$は以下のように計算することができる。

$$J_{1/2} \approx -\frac{(1-\alpha^-)D_1}{(1-\alpha^-)\frac{\Delta x_1}{2}+2(1+\alpha^-)D_1}\phi_1 \tag{47}$$

同様の手順により、右側のアルベド境界面についても、正味の中性子流$J_{NX+1/2}$は以下のように計算することができる。

$$J_{NX+1/2} \approx \frac{(1-\alpha^+)D_{NX}}{(1-\alpha^+)\frac{\Delta x_{NX}}{2}+2(1+\alpha^+)D_{NX}}\phi_{NX} \tag{48}$$

各空間メッシュの全中性子束ϕ_iとS_iを、それぞれ列ベクトル$\vec{\phi} \in \mathbb{R}^{NX\times 1}, \vec{S} \in \mathbb{R}^{NX\times 1}$の形に並べてみよう。以上で示した差分式(41), (42), (47), (48)に基づくことで、$\vec{\phi}$の数値解を得るための連立方程式を、以下で示すような行列・ベクトル形式で書き直すことができる。

$$\mathbf{A}\vec{\phi} = \vec{S} \tag{49}$$

$$\vec{\phi} = (\phi_1 \quad \phi_2 \quad \cdots \quad \phi_{NX})^{\mathrm{T}} \tag{50}$$

$$\vec{S} = (S_1 \quad S_2 \quad \cdots \quad S_{NX})^{\mathrm{T}} \tag{51}$$

上式において、上添字Tは転置を表す。また係数行列$\mathbf{A} \in \mathbb{R}^{NX\times NX}$は、以下で示すような「三重対角行列」と呼ばれる疎行列の形となっている。

$$\mathbf{A} = \begin{pmatrix} A_{0,1} & A_{x+,1} & 0 & 0 & \cdots & 0 & 0 & 0 \\ A_{x-,2} & A_{0,2} & A_{x+,2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & A_{x-,3} & A_{0,3} & A_{x+,3} & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & \cdots & A_{x-,NX-1} & A_{0,NX-1} & A_{x+1,NX-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & A_{x-,NX} & A_{0,NX} \end{pmatrix} \tag{52}$$

$$A_{x-,i} = \begin{cases} \dfrac{-2D_{i-1}D_i}{(D_{i-1}\Delta x_i + D_i\Delta x_{i-1})\Delta x_i} & \text{if } 2 \le i \le NX \\ \dfrac{-(1-\alpha^-)D_i}{\left((1-\alpha^-)\frac{\Delta x_i}{2}+2(1+\alpha^-)D_i\right)\Delta x_i} & \text{if } i = 1 \end{cases} \tag{53}$$

$$A_{x+,i} = \begin{cases} \dfrac{-2D_iD_{i+1}}{(D_i\Delta x_{i+1} + D_{i+1}\Delta x_i)\Delta x_i} & \text{if } 1 \le i \le (NX-1) \\ \dfrac{-(1-\alpha^+)D_i}{\left((1-\alpha^+)\frac{\Delta x_i}{2}+2(1+\alpha^+)\right)\Delta x_i} & \text{if } i = NX \end{cases} \tag{54}$$

$$A_{0,i} = \Sigma_{\mathrm{a},i} - \left(A_{x-,i} + A_{x+,i}\right) \tag{55}$$

「さて、この連立方程式(49)の数値解をどのように計算しようか？」。海野は、実際に手を動かしてプログラミングする前に情報収集をし始めた。連立方程式の数値解法としては、①直接解法、②定常反復解法、③非定常反復解法、など様々な方法がある。時間をかけてサーベイし

た結果、海野は「係数行列が式(52)のような三重対角行列の場合には、LU 分解による直接解法[15]で良い」との結論に至った。海野がこの判断に至った理由を、以下でフォローしてみよう。

三重対角行列**A**を LU 分解することで、下三角行列**L**と上三角行列**U**の行列積の形に変換することができる。

$$\mathbf{A} = \mathbf{LU} \tag{56}$$

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ L_2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & L_3 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \boxdot & \boxdot & \ddots & \boxdot & \boxdot & \vdots \\ 0 & 0 & 0 & 0 & \cdots & L_{NX} & 1 \end{pmatrix} \tag{57}$$

$$\mathbf{U} = \begin{pmatrix} U_1 & A_{x+,1} & 0 & 0 & \cdots & 0 & 0 \\ 0 & U_2 & A_{x+,2} & 0 & \cdots & 0 & 0 \\ 0 & 0 & U_3 & A_{x+,3} & \cdots & 0 & 0 \\ \vdots & \boxdot & \boxdot & \ddots & \boxdot & \boxdot & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & U_{NX} \end{pmatrix} \tag{58}$$

$$L_{i,g} = \begin{cases} 0 & \text{if } i = 1 \\ \dfrac{A_{x-,i}}{U_{i-1}} & \text{else} \end{cases} \tag{59}$$

$$U_{i,g} = \begin{cases} A_{0,1} & \text{if } i = 1 \\ A_{0,i} - L_i A_{x+,i-1} & \text{else} \end{cases} \tag{60}$$

このように係数行列**A**を LU 分解できれば、以下の手順により全中性子束$\vec{\phi}$の数値解を計算することができる。

$$\vec{z} = (z_1, z_2, \cdots, z_{NX})^T = \mathbf{L}^{-1}\vec{S} \tag{61}$$

$$z_i = \begin{cases} S_1 & i = 1 \\ S_i - L_i z_{i-1} & i \geq 2 \end{cases} \tag{62}$$

$$\vec{\phi} = \mathbf{U}^{-1}\mathbf{L}^{-1}\vec{S} = \mathbf{U}^{-1}\vec{z} \tag{63}$$

$$\vec{\phi} = \begin{cases} \dfrac{z_{NX}}{U_{NX}} & i = NX \\ \dfrac{z_i - A_{x+,i}\phi_{i+1}}{U_i} & i \leq (NX-1) \end{cases} \tag{64}$$

行列形式で書くことで一見複雑に見えるが、要は①「前進消去(逆行列$\mathbf{L}^{-1}$を左からかける演算)」により中性子源ベクトル$\vec{S}$の前処理をし、②「後退代入((逆行列$\mathbf{U}^{-1}$を左からかける演算)」により$\vec{\phi}$の数値解を順次計算しているだけである。

こうして海野が作り上げた、1 次元平板体系における中性子拡散計算を実施する Python 関数の例を以下に示す。

https://colab.research.google.com/github/hyd3nekosuki/RPDsummer2024/blob/main/1DSlabDiffusion2024RPDsmr.ipynb

```
import numpy as np
from numba import njit

@njit(cache=True)
def calcDiffusion1DX(D, SigA, s, dx, matid, albedo):
    NX  = len(s)
    L   = np.zeros(NX)
    U   = np.zeros(NX)
    Axp = np.zeros(NX)

    for i in range(NX):
        D0  = D[matid[i]]
        dx0 = dx[i]
        if i==0:
            a = albedo[0]
            Dxm = -(1-a)*D0/(2*(1+a)*D0 + (1-a)*dx0/2)
        else:
            Dm  = D[matid[i-1]]
            dxm = dx[i-1]
            Dxm = -2*Dm*D0/(Dm*dx0 + D0*dxm)
            Axm = Dxm/dx0
        if i==(NX-1):
            a = albedo[1]
            Dxp = -(1-a)*D0/(2*(1+a)*D0 + (1-a)*dx0/2)
        else:
            Dp  = D[matid[i+1]]
            dxp = dx[i+1]
            Dxp = -2*D0*Dp/(D0*dxp + Dp*dx0)
            Axp[i] = Dxp/dx0

        A0 = SigA[matid[i]] - ( (Dxm + Dxp)/dx0 )

        if i==0:
            L[i] = 0.0
            U[i] = A0
        else:
            L[i] = Axm/U[i-1]
            U[i] = A0 - Axp[i-1]*L[i]

    B = np.zeros(NX)
    BBefore = 0.0
    for i in range(NX):
        B[i] = s[i] - BBefore*L[i]
        BBefore = B[i]

    flux = np.zeros(NX)
    fluxAfter = 0.0
    for i in range(NX-1, -1, -1):
        fluxAfter = (B[i] - Axp[i]*fluxAfter) / U[i]
        flux[i] = fluxAfter

    return flux
```

なお、Python で作成したプログラムの計算を高速にするため、海野は Numba[16]を利用している。Numba とは Python 用の実行時コンパイラ(Just-In-Time (JIT) Compiler)であり、NumPy の配列や関数、for ループを使ったプログラムを機械語に「実行時コンパイル」することで計算速度の向上を図ることができる。特に、繰り返し処理(for 文)の中に更に複数の for 文や条件式(if 文)含まれている場合には、Numba を利用することで高速な Python プログラムを実装することができる。

4.3. 決定論的手法による中性子輸送計算

かくして、海野は中性子拡散計算コードを作成することで、高速に全中性子束の数値解を求めることができるようになった。ただ、海野の脳裏には、「全中性子束$\phi(x)$の数値を高速かつ正確に求める方法を開発せよ」、この言葉がこだましていた。この問題を十分には解決できていないのではないか、という不安がよぎっていた。有限体積法による中性子拡散計算の場合、以下で挙げるような「計算モデル起因の誤差」が発生することで、正確さ(accuracy)が損なわれている。

- ✓ 空間メッシュの差分化による離散化誤差。これを削減するためには、空間メッシュの幅を非常に細かくすればよいが、空間メッシュ数NXの増加、計算時間の増加に繋がる。
- ✓ 拡散近似による誤差。中性子の飛行方向を正確に考慮できていないため、拡散近似が成り立たない条件(例：角度中性子束の非等方性が強く、$P_1(\mu)$成分までででは近似できない場合)では、誤差が大きくなる。
- ✓ 極端な条件として、巨視的全断面積$\Sigma_t = 0$の真空領域を考えた場合には、式(24)で定義した拡散係数Dの値は無限大に発散してしまう。

この不安を払拭するため、海野は決定論的手法による中性子輸送計算コードの試作に着手し始めるのであった。まず海野は、1 次元平板体系で利用されている決定論的な中性子輸送計算手法について文献サーベイを開始した。候補となる手法として、衝突確率法[17]、S_N 法[18]、MOC(Method Of Characteristics, MOC)[19]、をピックアップすることができた。

最初に調べた「衝突確率法」とは、積分型の中性子輸送計算手法の一種である。ある領域iで生じた中性子が別の領域jで衝突する確率P_{ij}を数値計算で求め、各領域の全中性子束ϕ_iを求める方法であり、例えば、燃料セル格子体系を対象とした共鳴計算で使用されることが多い。ただ文献を調べるにつれ、1 次元平板体系の衝突確率P_{ij}を計算するには、「指数積分関数$E_{in}(x)$」と呼ばれる特殊関数が必要[17]のようであり、数学が得意でない海野の手に負える代物ではなかった。

そこで、海野にとって直観的に理解しやすい方法であった「MOC」を利用して、1 次元中性子輸送計算コードを試作することとした。MOC では、中性子の飛行方向をまるで針鼠🦔や毬栗✷の針のように細かく分割(離散化)して、各代表的な飛行方向について角度中性子束を計算する。拡散計算の場合と同様に、Figure 6 で示したように空間メッシュで分割し、i番目の空間メッシュについて、ある飛行方向μ_mの角度中性子束$\psi_m(x)$が従う中性子輸送方程式を考えると、以下のような形となる。

$$\mu_m \frac{d\psi_m}{dx} + \Sigma_t \psi_m(x) = \frac{d\psi_m}{ds} + \Sigma_t \psi_m(s) = q_i \tag{65}$$

$$q_i \equiv \frac{\Sigma_s \phi_i + S_i}{2} \tag{66}$$

ただし、$s = x/\mu$は飛行方向に沿った空間座標位置を意味する。式(65)右辺の等方中性子源q_iが

既知であれば、i番目の空間メッシュ内の角度中性子束$\psi_m(x)$の解析解を以下のように求めることができる。

$$\psi_m(x)=\begin{cases}\psi_{m,i-1/2}\mathrm{e}^{-\frac{\Sigma_\mathrm{t}(x-x_{i-1/2})}{\mu}}+\dfrac{q_i}{\Sigma_\mathrm{t}}\left(1-\mathrm{e}^{-\frac{\Sigma_\mathrm{t}(x-x_{i-1/2})}{\mu}}\right) & \mu>0\\ \psi_{m,i+1/2}\mathrm{e}^{-\frac{\Sigma_\mathrm{t}|x-x_{i+1/2}|}{|\mu|}}+\dfrac{q_i}{\Sigma_\mathrm{t}}\left(1-\mathrm{e}^{-\frac{\Sigma_\mathrm{t}|x-x_{i+1/2}|}{|\mu|}}\right) & \mu<0\end{cases} \tag{67}$$

ただし式(67)の解析解を導出するにあたって、空間メッシュ内でq_iが一定であると近似している。式(67)に基づくことで、空間メッシュに流入する角度中性子束$\psi_{m,i}^{\mathrm{in}}$から流出する角度中性子束$\psi_{m,i}^{\mathrm{out}}$を、以下の式に基づいて飛行方向に沿って逐次計算することができる。

$$\psi_{m,i}^{\mathrm{out}}=\psi_{m,i}^{\mathrm{in}}\mathrm{e}^{-\frac{\Sigma_\mathrm{t}\Delta x_i}{|\mu|}}+\frac{q_i}{\Sigma_\mathrm{t}}\left(1-\mathrm{e}^{-\frac{\Sigma_\mathrm{t}\Delta x_i}{|\mu|}}\right) \tag{68}$$

$$\psi_{m,i}^{\mathrm{in}}=\begin{cases}\psi_{m,i-1/2} & \mu>0\\ \psi_{m,i+1/2} & \mu<0\end{cases} \tag{69}$$

$$\psi_{m,i}^{\mathrm{out}}=\begin{cases}\psi_{m,i+1/2} & \mu>0\\ \psi_{m,i-1/2} & \mu<0\end{cases} \tag{70}$$

さらに、式(65)の両辺をi番目の空間メッシュの飛跡に沿って全積分し、飛跡の長さ$s_i=\Delta x_i/|\mu|$を割ることで、i番目の空間メッシュ内の平均的な角度中性子束$\overline{\psi}_{m,i}$も以下のように計算できる。

$$\overline{\psi}_{m,i}=\frac{1}{\Sigma_\mathrm{t}}\left(q_i-\frac{\psi_{m,i}^{\mathrm{out}}-\psi_{m,i}^{\mathrm{in}}}{\Delta x_i/|\mu|}\right) \tag{71}$$

こうして得られたm番目飛行方向の平均角度中性子束$\overline{\psi}_{m,i}$について、全飛行方向$(1\le m\le NM)$で数値積分することにより、i番目の空間メッシュ内の平均全中性子束ϕ_iも計算することができる。

$$\phi_i=\sum_{m=1}^{NM}w_m\overline{\psi}_{m,i} \tag{72}$$

$$\sum_{m=1}^{NM}w_m=2 \tag{73}$$

w_m：m番目飛行方向の重み。今回の場合、m番目飛行方向の方向余弦$\mu=\cos\theta$積分値 $\Delta\mu_m=\int_{\mu_{m-1/2}}^{\mu_{m+1/2}}d\mu$に相当する。

なお、式(66), (68)–(72)を見て分かるように、①全中性子束ϕ_iを計算するためには、角度中性子束$\overline{\psi}_{m,i}$の数値が必要となるが、②角度中性子束$\overline{\psi}_{m,i}$を計算するためには全中性子束ϕ_iの数値に基づいた中性子源q_iの数値も必要となっており、自己撞着な問題となっている。

とはいえ、炉物理計算プログラマーの海野にとっては、それほど厄介な問題ではない。「適当な全中性子束の初期値$\phi_i^{(0)}$を与えて、数値解が収束するまで反復計算すればよいだけだ」と、海野は反復処理を行うプログラミングを作成し始めた。

ただし、単純に体系を空間メッシュで細かく分割したり、角度中性子束$\overline{\psi}_{m,i}$の飛行方向を細かく分割したりするだけでは、正確な数値解を求めることはできても、中性子拡散計算に比較

すると多くの計算時間を要するプログラムとならざるを得ないことが分かった。特に、巨視的散乱断面積Σ_sが大きな計算条件(散乱比Σ_s/Σ_tが 1 に近い計算条件)の場合、えいやっと設定した散乱中性子源$\Sigma_s\phi_i$が収束するまでに多くの反復計算が必要となることが判明した。そこで海野は、これまでの炉物理分野での知見をざっとサーベイし、以下に挙げるような改良を施すこととした。

① Gauss-Legendre 分点[15]

式(72)の全中性子束ϕ_iに関する数値積分において、少ない飛行方向分割数NMで正確にϕ_iの数値積分が計算できるように、Gauss-Legendre(ガウス-ルジャンドル)分点に基づいて、離散化した飛行方向μ_mおよび重みμ_mを設定することとした。詳しい説明は文献[1], [2], [15]などを参照いただきたいが、角度中性子束の飛行方向依存性$\psi(\mu)$がμに関する$(2n-1)$次の多項式で表現できる場合には、n次のルジャンドル多項式$P_n(\mu)$がゼロとなる点で離散化すれば、正確にその数値積分を計算することができる。

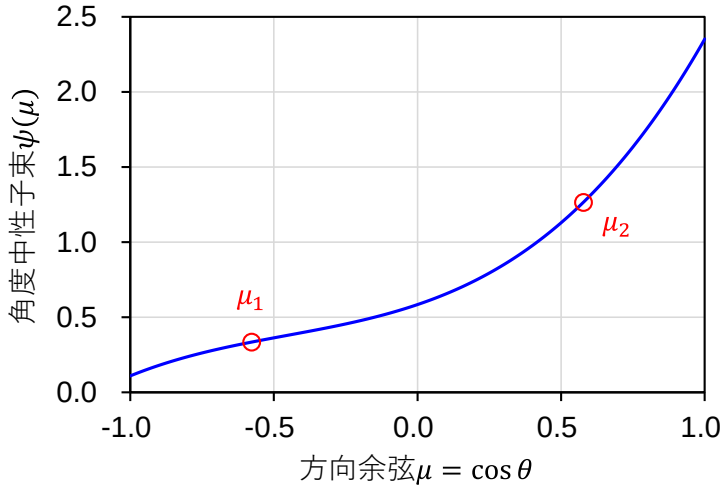

Figure 7 Gauss–Legendre 分点による角度中性子束の離散化例

例えば、角度中性子束$\psi(\mu)$の角度依存性が、飛行方向の方向余弦μに対して 3 次関数で表現できる場合を考えてみよう(Figure 7)。

$$\psi(\mu) = f_0 + f_1\mu + f_2\mu^2 + f_3\mu^3 \tag{74}$$

上式において、f_0, f_1, f_2, f_3は各次数の展開係数に相当する。この場合、$-1 \le \mu \le 1$の範囲で積分した全中性子束の正確な値ϕ_{ref}は以下のようになる。

$$\phi_{\mathrm{ref}} = \int_{-1}^{1} \psi(\mu)d\mu = 2f_0 + \frac{2f_2}{3} \tag{75}$$

一方、2 次のガウス・ルジャンドル分点を用いた MOC 計算の場合には場合には、$-1 \le \mu \le 1$の範囲を 2 つの代表的な飛行方向μ_1, μ_2で離散化することになる。ここで、μ_1, μ_2と対応する重みw_1, w_2の具体的な値は以下のとおりとなる。

$$\mu_1 = -\frac{1}{\sqrt{3}}, \mu_2 = -\frac{1}{\sqrt{3}}, \tag{76}$$

$$w_1 = w_2 = 1 \tag{77}$$

離散化された飛行方向μ_1, μ_2に対して、MOC 計算で正確にその方向の角度中性子束$\psi_1 = \psi(\mu_1), \psi_2 = \psi(\mu_2)$を求めることができたとすると、式(72)のような重み付き和による数値積分により全中性子束ϕ_{calc}は以下のように計算できる。

$$\begin{aligned}\phi_{\mathrm{calc}} &= w_1\psi_1 + w_2\psi_2 \\ &= \left\{f_0 + f_1\left(-\frac{1}{\sqrt{3}}\right) + f_2\left(-\frac{1}{\sqrt{3}}\right)^2 + f_3\left(-\frac{1}{\sqrt{3}}\right)^3\right\} \\ &+ \left\{f_0 + f_1\left(\frac{1}{\sqrt{3}}\right) + f_2\left(\frac{1}{\sqrt{3}}\right)^2 + f_3\left(\frac{1}{\sqrt{3}}\right)^3\right\} = 2f_0 + \frac{2f_2}{3}\end{aligned} \tag{78}$$

すなわち、Gauss-Legendre 分点による数値積分の結果は、全中性子束の正確な値ϕ_{ref}と一致する。「たった 2 つの点でしか離散化していないため、せいぜい直線状に近似した場合と変わらないのでは？」と考えていた海野にとって、これは驚嘆すべき結果であった。

以上の例から分かるように、角度中性子束$\psi(\mu)$の角度依存性が低次の多項式で近似できる条件となっていれば、Gauss-Legendre 分点を活用することで、MOC で計算する代表的な離散化方向μ_mの総数NMが少なくても、高い正確度で全中性子束を数値積分できるだろう。仮に各空間メッシュの全中性子束ϕ_iが正確に計算できれば、散乱中性子源$\Sigma_{\mathrm{s}}\phi_i$も正確に与えることができ、式(66), (67)による中性子輸送計算の結果の誤差も小さくなることが期待できる。

💡 MOC や S_N 法では、代表的な飛行方向で離散化して角度中性子束の飛行方向分布を調べる必要があるため、基本的には多くの計算コストを要する。計算時間を少しでも短縮したい場合には、「全中性子束の数値積分を正確に計算する」ことを念頭に置いて、1 次元平板体系であれば Gauss-Legendre 分点など、計算対象の体系に応じて適切な角度分点を使用して、計算すべき角度中性子束の総数NMを削減することが重要となる。

② 詳細メッシュ拡散加速法[20], [21]

式(41), (42)で示したように、中性子拡散計算を考えた場合には、自群散乱中性子源を陽に取り扱う必要がなく、巨視的吸収断面積$\Sigma_{\mathrm{a}} = \Sigma_{\mathrm{t}} - \Sigma_{\mathrm{s}}$を用いて、正味で吸収された中性子のみを対象として効率良く全中性子束ϕ_iを求めることができる。もしも、「中性子輸送計算で得られる正味の中性子流J_{tr}の結果を再現できるように、通常の拡散計算で使用している正味の中性子流J_{dif}の計算式(42)を補正」できさえすれば、高速かつ正確な計算が実現できるはずである。

海野が最新の文献調査したところ、以下のように補正項$D_{\mathrm{cor},i+1/2}(\phi_{i+1} + \phi_i)$を付け加えた正味の中性子流の計算式を用いることで、中性子輸送計算の反復回数(transport sweep 回数)を大幅に削減できることが分かった。

$$J_{i+1/2} = \frac{-2\widetilde{D}_i\widetilde{D}_{i+1}}{\widetilde{D}_i\Delta x_{i+1} + \widetilde{D}_{i+1}\Delta x_i}(\phi_{i+1} - \phi_i) + D_{\mathrm{cor},i+1/2}(\phi_{i+1} + \phi_i) \tag{79}$$

上式において、補正係数$D_{\mathrm{cor},i+1/2}$は、輸送計算で得られた正確な正味の中性子流$J_{\mathrm{tr},i+1/2}$に基づいて以下のように求めることができる。

$$D_{\mathrm{cor},i+1/2} = \frac{J_{\mathrm{tr},i+1/2} - \left\{\frac{-2\widetilde{D}_i\widetilde{D}_{i+1}}{\widetilde{D}_i\Delta x_{i+1} + \widetilde{D}_{i+1}\Delta x_i}(\phi_{i+1} - \phi_i)\right\}}{\phi_{i+1} + \phi_i} \tag{80}$$

実際の MOC 輸送計算では、数値解が収束するまで多数回の反復計算(transport sweep)を実施する必要があるが、n回目の外部反復計算で得られた収束途中の中性子流$J^{(n)}_{\mathrm{tr},i+1/2}$を近似値として、式(80)により補正係数$D^{(n)}_{\mathrm{cor},i+1/2}$の近似値を求める。こうして推定された$D^{(n)}_{\mathrm{cor},i+1/2}$を用いることで、輸送計算結果を極力再現できるような中性子拡散計算を実施することができ、自群散乱中性子源を効率良く更新できる、すなわち transport sweep の回数を大幅に削減することができる。

ただし、海野が拡散加速法を試しに実装してみたところ、特定の計算条件(光学距離$\Sigma_t\Delta x$が大きい場合)において収束解が得られず反復計算が不安定になる、といった悩ましい問題が生じることも明らかとなった。この場合の処方箋として、様々な方法がこれまでに編み出されており、例えば、補正項$D_{\mathrm{cor},i+1/2}$計算時や拡散加速計算時に用いる拡散係数$\widetilde{D}_i$として、「artificial grid diffusion (AGD) coefficient [20]」と呼ばれる以下の拡散係数を代わりに使用することで安定に加速することができる。

$$\widetilde{D}_i = \frac{1}{3\Sigma_t} + \frac{\Delta x_i}{4} \tag{81}$$

以上で述べたような試行錯誤の結果、海野は比較的高速に中性子輸送計算の数値解を得ることができる計算プログラムを作成することができた。

1 次元平板体系の MOC 輸送計算コード：
https://colab.research.google.com/github/hyd3nekosuki/RPDsummer2024/blob/main/1DSlabMOC2024RPDsmr.ipynb

💡中性子の飛行方向も厳密に取り扱う「中性子輸送計算」は正確な数値解法ではあるが、何の工夫も無く実装しても、非常に多くの計算コストを要してしまい、実用的な方法とはならない。この場合には、飛行方向に関する変数を削減した、より低次元なモデルである「中性子拡散計算」を有効活用することで、中性子輸送計算の反復回数を大幅に削減することができる。

近年の決定論的中性子輸送計算コードでは、拡散加速法を実装することが当たり前となっている。炉心解析や計算コード開発にたずさわることとなる研究者・エンジニアには、①中性子拡散計算の基礎を十分に理解した上で、②どのような計算条件で拡散加速計算の不安定性が生じるのか把握しておくことを強く推奨する。

5. ROM によるアプローチ

5.1. 特異値分解

今回解くべき問題は、「計算の正確さを求めれば計算時間が長くなり、近似や簡略化で計算時間を短くすれば正確さを損なう」というジレンマを抱えた問題であり、取り組むエンジニアの腕の見せどころとなっている。

辻は、上述したような、これまでの炉物理計算手法における課題の打開を目指し、炉物理分野に対する ROM の応用についてこれまで勉強に励んできた。

まず辻は、計算の入力として与えるべきパラメータ$(\Sigma_a, \Sigma_s, S(x), \alpha^-, \alpha^+)$について、乱数を用いて様々な組み合わせを考え、職場の先輩が整備して下さった「1 次元 MOC 中性子輸送計算コード」を用いて計算を多数回実施してみた。しかし、得られた全中性子束$\phi(x)$の空間分布(Figure 8)を眺めてみたものの、与える入力パラメータの変化に応じて$\phi(x)$が変わることは確認できたが、規則性を見いだすことは、辻にとって容易では無かった。

そこで辻は、過去に流体解析の分野で産声をあげ、近年では原子炉物理の分野でも活用されつつある固有直交分解(Proper Orthogonal Decomposition, POD)[22]–[24]を利用して、計算条件を変えた$\phi(x)$のデータに含まれている「主要な成分」をデータ駆動型の方法論で推定することを試みた。

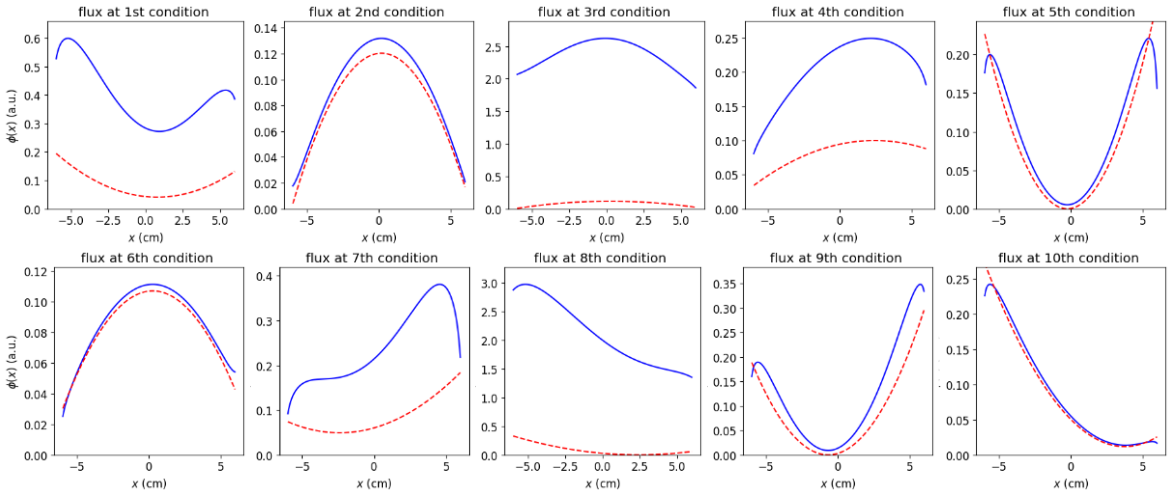

Figure 8 パラメータを変えた時の全中性子束$\phi(x)$の変化例
(青実線：$\phi(x)$、赤破線：中性子源$S(x)$)

POD を実施するため、まずは、ある計算条件cで得られた全中性子束の空間分布を、1 次元の列ベクトル$\vec{\phi}_c = (\phi_{1,c} \quad \phi_{2,c} \quad \cdots \quad \phi_{NX,c})^{\mathrm{T}} \in \mathbb{R}^{NX\times 1}$の形に並べる。その後、計算条件を$NC$回だけ変化させて得られた$NC$個の列ベクトル$\vec{\phi}_c$を横に並べて、中性子束のデータ行列$\boldsymbol{\Phi} \in \mathbb{R}^{NX\times NC}$を用意する。

$$\boldsymbol{\Phi} = \left(\vec{\phi}_1 \quad \vec{\phi}_2 \quad \cdots \quad \vec{\phi}_{NC}\right) = \begin{pmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,NC} \\ \phi_{2,1} & \phi_{2,2} & \square & \phi_{2,NC} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{NX,1} & \phi_{NX,2} & \cdots & \phi_{NX,NC} \end{pmatrix} \tag{82}$$

このデータ行列$\mathbf{\Phi}$に対して特異値分解(Singular Value Decomposition, SVD)[15]を適用することで、行列$\mathbf{\Phi}$を、以下の 3 つの行列の積で表現することができる。

$$\mathbf{\Phi} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} = \sum_{i=1}^{r} \vec{u}_i \sigma_i \vec{v}_i^{\mathrm{T}} \tag{83}$$

$$\mathbf{U} = (\vec{u}_1 \quad \vec{u}_2 \quad \cdots \quad \vec{u}_r) = \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,r} \\ u_{2,1} & u_{2,2} & \ddots & u_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ u_{NX,1} & u_{NX,2} & \cdots & u_{NX,r} \end{pmatrix} \tag{84}$$

$$\mathbf{V} = (\vec{v}_1 \quad \vec{v}_2 \quad \cdots \quad \vec{v}_r) = \begin{pmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,r} \\ v_{2,1} & v_{2,2} & \ddots & v_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ v_{NC,1} & v_{NC,2} & \cdots & v_{NC,r} \end{pmatrix} \tag{85}$$

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{pmatrix} \tag{86}$$

上式において、rは中性子束データ行列$\mathbf{\Phi}$のランク$r \leq \min(NX, NC)$、$\mathbf{U} \in \mathbb{R}^{NX \times r}$は左特異ベクトル$\vec{u}_i$を並べたセミユニタリ行列、$\mathbf{V} \in \mathbb{R}^{NC \times r}$は右特異ベクトル$\vec{v}_i$を並べたセミユニタリ行列、$\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$は対角成分の要素として特異値$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$を並べた対角行列、をそれぞれ表す。ここでベクトル$\vec{u}_i, \vec{v}_i$は、以下で示す固有値方程式を満足する正規直交ベクトルである。

$$(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}})\vec{u}_i = \sigma_i^2 \vec{u}_i \tag{87}$$

$$(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})\vec{v}_i = \sigma_i^2 \vec{v}_i \tag{88}$$

すなわち、①行列$\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} \in \mathbb{R}^{NX \times NX}$の固有ベクトルが左特異ベクトル$\vec{u}_i$に対応し、②$\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} \in \mathbb{R}^{NC \times NC}$の固有ベクトルが右特異ベクトル$\vec{v}_i$に対応し、③これら行列$\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}, \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$の固有値$\lambda_i = \sigma_i^2$が特異値の 2 乗に対応している。以上で説明した数学的性質より、$\vec{u}_i$と$\vec{v}_i$はそれぞれ正規直交条件を満足するため、$\mathbf{U}$あるいは$\mathbf{V}$の左からそれらの転置行列$\mathbf{U}^{\mathrm{T}}$あるいは$\mathbf{V}^{\mathrm{T}}$をかけると単位行列$\mathbf{I}_r \in \mathbb{R}^{r \times r}$となる。

$$\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}_r \quad \because \vec{u}_i \vec{u}_j = \delta_{ij} \tag{89}$$

$$\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}_r \quad \because \vec{v}_i \vec{v}_j = \delta_{ij} \tag{90}$$

辻は、1 次元平板体系の空間メッシュを$NX = 240$に分割した上で、乱数を用いて入力パラメータ組み合わせ$(\Sigma_a, \Sigma_s, S(x), \alpha^-, \alpha^+)$を$NC = 400$回変化させて、240 行 400 列の中性子束データ行列$\mathbf{\Phi}$を準備した。Python/Numpy の `numpy.linalg.matrix_rank()` 関数[25]を用いて、$\mathbf{\Phi}$のランクrを調べてみると、驚くべきことに行列$\mathbf{\Phi}$のランクは$r = 39$であり、NX, NCの大きさと比べて約 1 桁小さいことが分かった。この理由について理解を深めるため、辻は Python/Numpy の `numpy.linalg.svd()` 関数[26]を用いて、行列$\mathbf{\Phi}$の特異値σ_iを調べてみたところ Figure 9 のような結果となった。Figure 9 から分かるように、特異値の次数iが大きくなるにつれて特異値

σ_2の大きさは指数関数的に減少し続け、次数が十分に大きくなると極めて小さな値に特異値が収束している。なお、`numpy.linalg.matrix_rank()`関数のマニュアル[25]を調べたところ、デフォルトでは「行列の特異値最大値×(行数あるいは列数の最大値)×計算機イプシロン」を判定値として行列ランクrを求めている、とのことだった。数値計算では、浮動小数点の小数で取り扱うことが可能な計算精度に限界があるため、今回の場合には、$i > 39$の特異値については数値計算上、無視可能なオーダーとみなせるだろう。

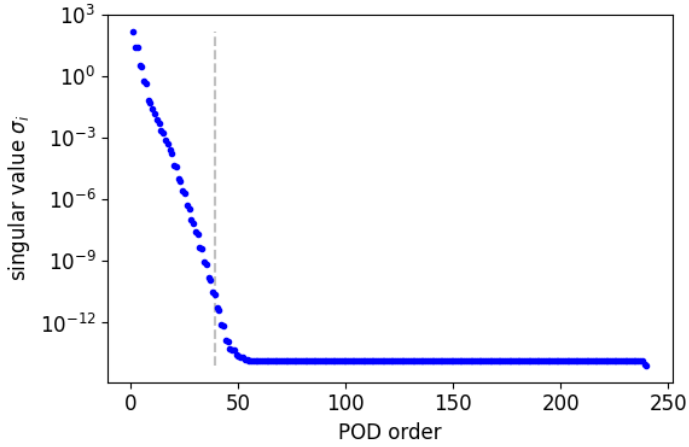

Figure 9 特異値の分布例

以上で述べた特異値分解(式(83))による行列$\mathbf{\Phi}$の圧縮について、それを可視化した例を Figure 10 に示す。行列$\mathbf{\Phi}$のもともとの行列サイズ(NX, NC)と比べて、ランクrが小さいほど、特異値分解による圧縮効果も大きいことが視覚的に理解できるだろう。

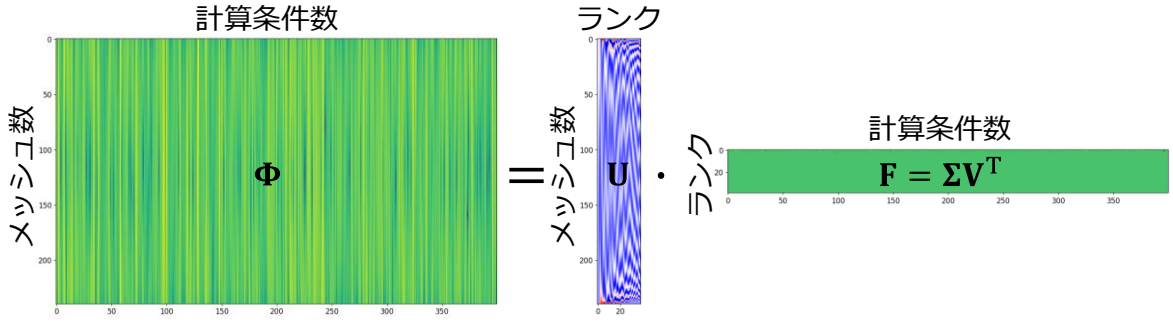

Figure 10 特異値分解による行列の圧縮

💡行数や列数が多い、サイズの大きなデータ行列$\mathbf{\Phi}$があったとしても、その行列ランクrが小さい場合には、特異値分解を利用することでデータ量を大幅に削減することができる。

なお、様々な計算条件を取り得る場合には、本資料では乱数を利用して入力パラメータ$(\Sigma_a, \Sigma_s, S(x), \alpha^-, \alpha^+)$の組み合わせを決め、データ行列$\mathbf{\Phi}$を準備していた。データ行列のサイズがさらに大規模なデータとなる場合には、効率良く特異値分解を実施するための手法として Randomized SVD[27]というアルゴリズムも提案されているので記憶に留めておくと良い。

5.2. 固有直交分解

前節で説明した特異値分解と、POD との関係について理解を深めるため、式(83)を各計算条件cで得られた全中性子束の列ベクトル$\vec{\phi}_c$に注目して、以下のように書き直してみよう。

$$\mathbf{\Phi} = \begin{pmatrix} \vec{\phi}_1 & \cdots & \vec{\phi}_c & \cdots & \vec{\phi}_{NC} \end{pmatrix} = \mathbf{UF} \tag{91}$$

$$\mathbf{F} = \begin{pmatrix} \vec{f}_1 & \cdots & \vec{f}_c & \cdots & \vec{f}_{NC} \end{pmatrix} = \begin{pmatrix} f_{1,1} & \cdots & f_{1,c} & \cdots & f_{1,NC} \\ f_{2,1} & \cdots & f_{2,c} & \cdots & f_{2,NC} \\ \vdots & \boxdot & \vdots & \ddots & \vdots \\ f_{r,1} & \cdots & f_{r,c} & \cdots & f_{r,NC} \end{pmatrix} = \mathbf{\Sigma V}^{\mathrm{T}} \tag{92}$$

$$\vec{\phi}_c = \mathbf{U}\vec{f}_c = \sum_{i=1}^{r} f_{i,c}\,\vec{u}_i = \sum_{i=1}^{r} \sigma_i v_{c,i} \vec{u}_i \tag{93}$$

$$\vec{f}_c = \begin{pmatrix} f_{1,c} & f_{2,c} & \cdots & f_{r,c} \end{pmatrix}^{\mathrm{T}} = \begin{pmatrix} \sigma_1 v_{c,1} & \sigma_2 v_{c,2} & \cdots & \sigma_r v_{c,r} \end{pmatrix}^{\mathrm{T}} \tag{94}$$

$$f_{i,c} = \sigma_i v_{c,i} \tag{95}$$

上式において、$f_{i,c}$は$\vec{\phi}_c$をi番目のベクトル$\vec{u}_i$で展開した時の展開係数、列ベクトル$\vec{f}_c \in \mathbb{R}^{r\times 1}$は展開係数$f_{i,c}$を縦方向に並べた列ベクトル、行列$\mathbf{F} \in \mathbb{R}^{r\times NC}$は$\vec{f}_c$を横方向に並べた展開係数行列、をそれぞれ表す。

すなわち、$\mathbf{\Phi}$を特異値分解する行列演算は、計算条件cの全中性子束$\vec{\phi}_c$を基底ベクトル$\vec{u}_i$と対応する展開係数$\sigma_i v_{c,i}$の積和で展開していることと等価と言える。以上より、$\mathbf{\Phi}$を展開する左特異ベクトル$\vec{u}_i$を POD 基底と呼び、$\vec{u}_i$を横方向に並べた行列$\mathbf{U}$を POD 基底行列、対応する展開係数を並べた行列$\mathbf{F}$を展開係数行列と呼ぶ。なお、展開係数行列$\mathbf{F}$については、上述した式(92)のように特異値分解で得られた$\mathbf{\Sigma}$と$\mathbf{V}^{\mathrm{T}}$の行列積により求めることができるが、式(91)両辺に左から転置行列$\mathbf{U}^{\mathrm{T}}$をかけることで分かるように、中性子束データ行列$\mathbf{\Phi}$に左から$\mathbf{U}^{\mathrm{T}}$をかけても求めることができる。

$$\mathbf{F} = \mathbf{U}^{\mathrm{T}}\mathbf{\Phi} \tag{96}$$

以上の計算理論を踏まえ、辻は計算条件を$NC = 400$回変化させて得られた中性子束データ行列$\mathbf{\Phi}$を特異値分解することで推定した、POD 基底ベクトル$\vec{u}_i$の空間分布を調べてみた(Figure 11)。Figure 11 から分かるように、特異値分解を利用することで、「解きたい計算体系で現れる固有の空間分布」を、得られたデータ行列のみに基づいたデータ駆動型の方法論により抽出できることが分かる。例えば、特異値σ_1が最も大きく、各計算条件で得られた全中性子束に含まれる主要な 1 次 POD 基底$\vec{u}_1$の空間分布形状は、三角関数$\cos(x)$のように体系中心でピークを持つ形状となっている。また、POD 基底の次数iが大きくなるにつれて、正/負の振動回数が増加しており、Figure 3 で示したルジャンドル多項式$P_l(x)$のように、左/右端の境界面($x = \pm H/2$)付近における振動が顕著となる空間分布形状となっている。

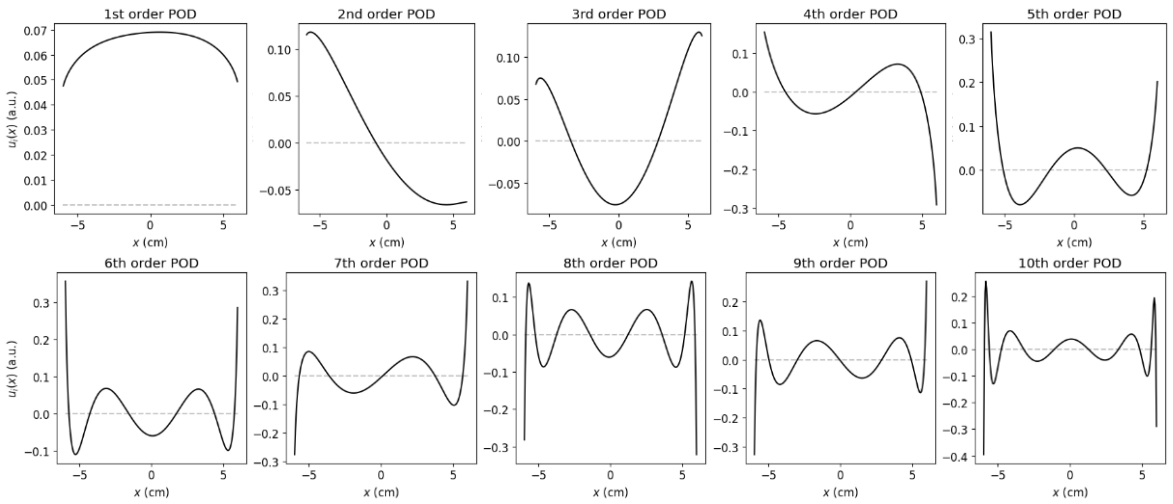

Figure 11 POD 基底ベクトルの推定結果

ただ、Figure 11 の POD 基底ベクトル$\vec{u}_i$の空間分布を眺めて、辻は少し違和感を抱いた。「入力パラメータの組み合わせを乱数により設定したせいかもしれないが、今回の問題は 1 次元平板の均質体系にもかかわらず、POD 基底ベクトル$\vec{u}_i$の空間分布が中心$x = 0$に対して非対称となっているのだろうか？」と。辻が色々と条件を変えて検討したところ、計算条件の数NCを大きくすることで POD 基底ベクトル$\vec{u}_i$の結果が対称な形状となることが、経験的に分かった。

そこで辻は、先行研究[28]で活用されていた「データ拡張 (Data Augmentation)」を利用して、POD 基底を推定することとした。例えば、ある計算条件で得られた全中性子束$\phi_c(x)$であったとする。今回の計算条件の場合には幾何形状に対称性があるため、アルベド境界条件(α^-, α^+)と$S(x)$を$x = 0$に対して反転させた入力条件を考えた場合、元々の計算条件で得られた空間分布を反転した$\phi_{\mathrm{flip},c}(x) = \phi_c(-x)$となることが自明であろう。すなわち、アルベド境界条件と$S(x)$を反転させた入力条件で改めて中性子輸送計算を実施しなくても、$\phi_c(x)$を$x = 0$に対して反転させた$\phi_{\mathrm{flip},c}(x)$を中性子束データ行列$\mathbf{\Phi}$に追加することで、POD 基底推定に用いるデータ量を倍に増やすことができる。例えば、データ拡張を利用した POD 基底を推定する Python プログラム例は以下のように作成できる。

```
snapshotF = np.append(snapshotF, snapshotF[::-1,:], axis=1)
rankF = np.linalg.matrix_rank(snapshotF)
print("rank of flux snapshot data= {}".format(rankF))
UF, sF2, VFh = np.linalg.svd(snapshotF, full_matrices=False)
coefF = np.diag(sF2[:rankF]).dot(VFh[:rankF,:])
UF = UF[:,:rankF]
for i in range(rankF):
    Umin = UF[:,i].min()
    Umax = UF[:,i].max()
    if Umax < abs(Umin):
        UF[:,i]*=-1
        coefF[i,:]*=-1
```

上記プログラム例では、Numpy 配列のスライシングの機能を駆使して、配列添字を`::-1` とすることで、始点 0 番目から終点$NX - 1$番目の要素を、始点$NX - 1$番目から終点 0 番目となるように反転させている。加えて、POD 基底ベクトル$\vec{u}_i$の符号を反転させても対応するi次の展開係数$f_{i,c}$の符号が反転するだけなので、各次数の POD 基底ベクトル$\vec{u}_i$の最大値符号が正となる

ように符号も反転させている。Figure 11 の場合と同じ全中性子束のデータに対して、行列データ拡張を利用して POD 基底ベクトルを推定した結果を Figure 12 に示す。Figure 12 から分かるように、データ拡張を利用することで、中心$x = 0$に対して対称性を満足した POD 基底ベクトル$\vec{u}_i$を求めることができる。

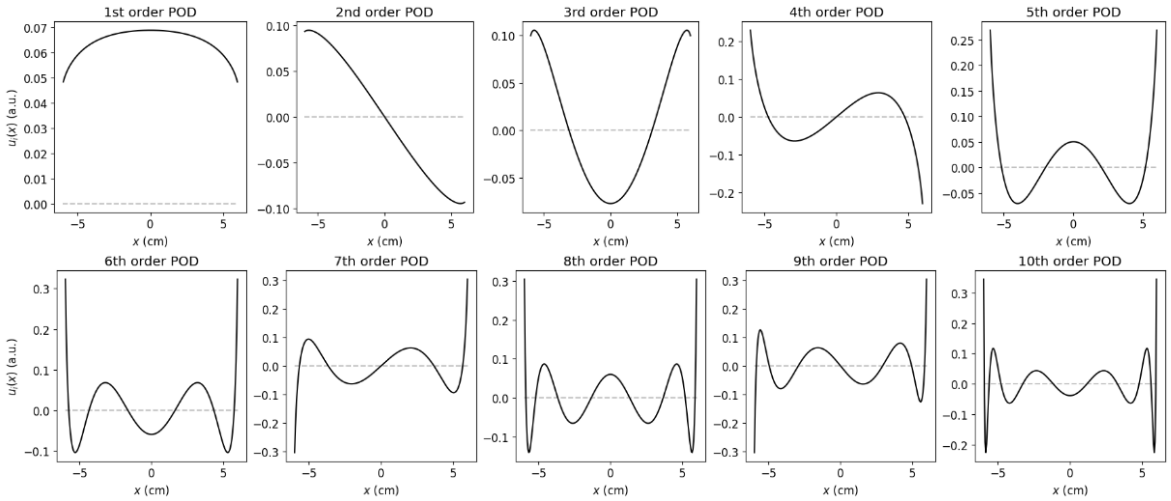

Figure 12 データ拡張を適用した POD 基底ベクトルの推定結果

💡データ行列$\mathbf{\Phi}$を特異値分解することで得られた POD 基底ベクトル$\vec{u}_i$は、特異値σ_iの大きさ順に並んでおり、特異値σ_iが大きな低い次数の$\vec{u}_i$ほど、データ行列を構成する各計算条件の全中性子束$\vec{\phi}_c$に含まれる「主要な成分」に対応している。言い換えると、もともとのデータ行列$\mathbf{\Phi} = \mathbf{U\Sigma V^{T}\Phi}$を構成する多数の成分のうち、特異値$\sigma_i$が非常に小さな次数以降の成分$\vec{u}_i\sigma_i\vec{v}_i^{\mathrm{T}}$は寄与割合が小さく無視できる(次元削減できる)ことに繋がる。

筆者が伝え聞いたノウハウとして、「多項式展開ノード法の重み付き残差法で用いられる展開関数(0 次～4 次までの多項式)[5]は、ルジャンドル多項式と一見似ているが、展開誤差が小さくなるよう経験的に設定した多項式」であるとのことだった。

こうした工学者の優れた経験・知見に基づくことなく、POD ではデータ駆動型の方法論により、解くべき計算体系・計算条件に応じた形で、汎用的な数学関数では表現しきれないような、特徴的な体系固有の分布$\vec{u}_i$を推定することができる。なお、Figure 12 が示唆しているように、POD 基底ベクトル$\vec{u}_i$の分布は必ずしも解析的な正規直交関数系(三角関数、ルジャンドル多項式など)とは一致しない。

5.3. POD 展開係数のテーブル化

前節で述べたような手順により、対象とする計算体系の全中性子束を、少ない次数で効率良く展開できそうな POD 基底ベクトル$\vec{u}_i$を、辻はようやく推定することができた。

「さて、中性子輸送計算を利用して得られた POD 基底ベクトル$\vec{u}_i$を用いて、どのように ROM を構築しようか」と、辻は暫く悩んだ。最初から問題を難しく過ぎると解決策も浮かばないため、問題を簡略化し、ある入力パラメータpが 1 個のみ変化した場合に、POD 基底の展開係数

$f_i(p)$がどのように変化するのか調査することとした。

例えば、ある特定の1個の入力パラメータpを変化させて中性子輸送計算を実施することで、全中性子束の空間分布$\vec{\phi}(p) \in \mathbb{R}^{NX\times 1}$が計算できたとする。このとき、$\vec{\phi}(p)$を POD 基底ベクトル$\vec{u}_i$で展開した展開係数$f_i(p)$は、2 つのベクトル$\vec{u}_i$と$\vec{\phi}(p)$の内積、すなわち「$\vec{\phi}(p)$を基底ベクトル$\vec{u}_i$方向に投影した長さ」により求めることができる。

$$f_i(p) = \vec{u}_i^{\mathrm{T}}\vec{\phi}(p) \tag{97}$$

あるいは、$1 \le i \le r$の展開係数を纏めた列ベクトル$\vec{f}(p) \in \mathbb{R}^{r\times 1}$を求めたければ、$\vec{\phi}(p)$に左から POD 基底ベクトル行列の転置行列$\mathbf{U}^{\mathrm{T}}$をかけることで一斉に求めることもできる。

$$\vec{f}(p) = (f_1(p) \quad f_2(p) \quad \cdots \quad f_r(p))^{\mathrm{T}} = \mathbf{U}^{\mathrm{T}}\vec{\phi}(p) \tag{98}$$

入力パラメータpを様々に変化させて展開係数を調べ、例えば、任意の関数でフィッティングしたり、テーブル化したりすることで、展開係数$f_i(p)$を「何らかの関数」でモデル化できたとする。ひとたび展開係数のモデル化さえ構築できれば、あとは解きたい計算条件で与える入力パラメータp_{target}をモデル化した展開係数に代入することで、対応する展開係数$f_i(p_{\mathrm{taget}})$を低コストで計算できる。こうして推定した展開係数$f_i(p_{\mathrm{taget}})$を用いて、POD では以下のように全中性子束を再構成することができる。

$$\vec{\phi}_{\mathrm{POD}}(p) \approx \mathbf{U}\vec{f}(p) \tag{99}$$

試計算として、辻は注目する入力パラメータ以外はある条件で固定し、ある 1 つの入力パラメータ(中性子源$S(x)$の 2 次関数係数a, b、左側境界面のアルベド値α^-)のみを様々な範囲で変化させて POD 展開係数$f_i(p)$の変化を調べ、Figure 13 のように図示してみた。Figure 13 から分かるように、1 個の入力パラメータのみが変化する場合には、POD 展開係数$f_i(p)$は直線や滑らかに変化する関数形で表現できそうな形状となっていることが確認できた。

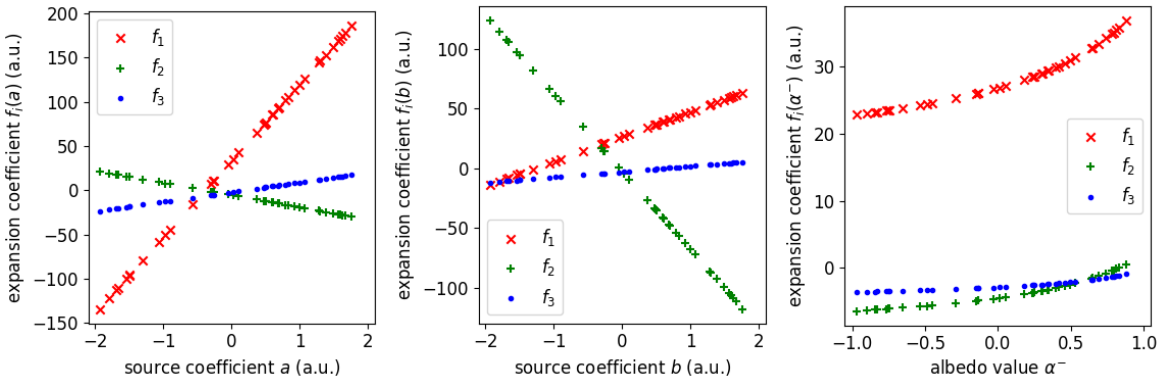

Figure 13 入力パラメータ 1 個のみ変化させたときの POD 展開係数の変化

しかし、2 つの入力パラメータが変化する場合、3 つの入力パラメータが変化する場合、…

とさらに複雑な状況を考えるにつれて、辻は深く溜息をついた。「今回の場合には、次元の呪い[29]を解決できそうにないかも？」と。例えば、仮に POD 基底ベクトルの展開次数がたかだか$r \approx 40$までで済んだとしても、1 つの入力パラメータ$f_i(p)$のみ変化する形で済めば、40 個の POD 展開係数$f_i(p)$をテーブル化すればよいだけだが、今回の問題では入力パラメータ$(\Sigma_\mathrm{a}, \Sigma_\mathrm{s}, a, b, c, \alpha^-, \alpha^+)$の総数が 7 つあり、これら 7 つのパラメータが同時に変化する組み合わせも考えて、7 変数のテーブル化$f_i(\Sigma_\mathrm{a}, \Sigma_\mathrm{s}, a, b, c, \alpha^-, \alpha^+)$について考えねばならない…。

流行りの「機械学習」が得意な後輩であれば、この難問をサクッと解決できるかもしれないが、辻は機械学習スキルを十分習得していなかったため、別の代替案を模索することとした。

💡そもそも、変化する入力パラメータの数が少ない場合には、手段は問わないので、POD 展開係数$f_i(p)$を「入力パラメータpの変化に対してテーブル化」するだけでよい。こうしてテーブル化された POD 展開係数$f_i(p)$を利用することで、求めるべき対象体系の展開係数$f_i(p_\mathrm{target})$を計算でき、求めたい出力(本資料の場合には全中性子束$\vec{\phi}(p_\mathrm{target})$)を高速に計算可能な ROM を容易に構築することができる。

余談となるが、原子炉物理における偉大な先人が編み出した「巧妙なテーブル化による ROM」としては、「等価原理に基づく共鳴計算[2], [30]」を挙げることができるだろう。決定論的手法による格子計算では、非均質体系における燃料棒内の中性子束エネルギースペクトル$\phi(E)$を真面目に解く代わりに、その計算と等価な条件となるよう「背景断面積σ_b」を与えた簡易式(例 : $\phi(E) \propto \left(\sigma_{\mathrm{t},r}\,(E) + \sigma_\mathrm{b}\right)^{-1} E^{-1}$)によって、微視的断面積の共鳴核反応による$\phi(E)$のエネルギースペクトルを極めて効率良く推定する方法論が編み出されている。これは計算機性能がが低かった時代において、燃料棒内の核種組成や燃料棒セル配置の非均質性が$\phi(E)$に及ぼす効果を「背景断面積σ_b」というパラメータを介して縮約したモデルであり、偉大な先人達が考案した「共鳴計算に対する工学的な ROM」の先駆けであると言えるだろう。

5.4. POD を適用した中性子拡散計算

中性子輸送計算に対して POD を適用するといった挑戦的な問題に取り組む前に、辻は先行研究[23]で検討された「POD を適用した中性子拡散計算」を実装してみることとした。まず、計算体系を細かく空間メッシュで分割して離散化することにより、中性子拡散方程式は式(49)で示すような行列・ベクトル形式の連立方程式で記述することができる。

$$\mathbf{A}\vec{\phi} = \vec{S} \tag{49}$$

再掲

上式において、$\mathbf{A} \in \mathbb{R}^{NX \times NX}$は漏れや吸収による中性子消滅を表す係数行列、$\vec{\phi} \in \mathbb{R}^{NX \times 1}$は全中性子束の列ベクトル、$\vec{S} \in \mathbb{R}^{NX \times 1}$は中性子源の列ベクトルをそれぞれ表す。式(49)の連立方程式は、空間メッシュ数NXが大きくなるにつれて係数行列**A**のサイズが大きな問題となるため、数値計算手法に関する十分なノウハウを有していなければ(今回の場合には、係数行列**A**が三重対角行列であることを利用して効率的な数値解法を選択しなければ)、数値解$\vec{\phi}$を得るのに非常に多くの計算コストが必要となってしまう。

そこで、式(49)のような大規模な連立方程式を解く代わりに、次元圧縮して行列サイズを小

さくした、小規模な連立方程式を解くことで効率良く$\vec{\phi}$の数値解を得ることができないだろうか？このような次元削減を実現するため、中性子拡散計算に POD を適用してみよう。まず、全中性子束$\vec{\phi}$が、少数個$(1 \le i \le r)$の POD 基底ベクトル$\vec{u}_i \in \mathbb{R}^{NX\times 1}$を用いて以下のように展開できると仮定しよう。

$$\vec{\phi} = \mathbf{U}\vec{f} \tag{100}$$

$$\mathbf{U} = (\vec{u}_1 \quad \vec{u}_2 \quad \cdots \quad \vec{u}_r) = \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,r} \\ u_{2,1} & u_{2,2} & \ddots & u_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ u_{NX,1} & u_{NX,2} & \cdots & u_{NX,r} \end{pmatrix} \tag{84}$$

再掲

上式において、$\vec{f} = (f_1 \quad f_2 \quad \cdots \quad f_r)^{\mathrm{T}} \in \mathbb{R}^{r\times 1}$は POD 基底展開係数を並べた列ベクトルを表す。式(100)を式(49)に代入し、左から POD 基底ベクトル行列の転置$\mathbf{U}^{\mathrm{T}}$をかけることで、POD 基底展開係数ベクトル$\vec{f}$に関する、小規模な連立方程式を以下のように得ることができる。

$$\begin{gathered} \tilde{\mathbf{A}}\vec{f} = \vec{S} \\ \because \mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{U}\vec{f} = \mathbf{U}^{\mathrm{T}}\vec{S} \end{gathered} \tag{101}$$

$$\tilde{\mathbf{A}} \equiv \mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{U} \tag{102}$$

$$\vec{S} = \mathbf{U}^{\mathrm{T}}\vec{S} \tag{103}$$

上式において、$\tilde{\mathbf{A}} \in \mathbb{R}^{r\times r}$は POD 展開係数の消滅項を圧縮した係数行列、$\vec{S} \in \mathbb{R}^{r\times 1}$は中性子源$\vec{S}$の POD 展開係数ベクトル、をそれぞれ表す。

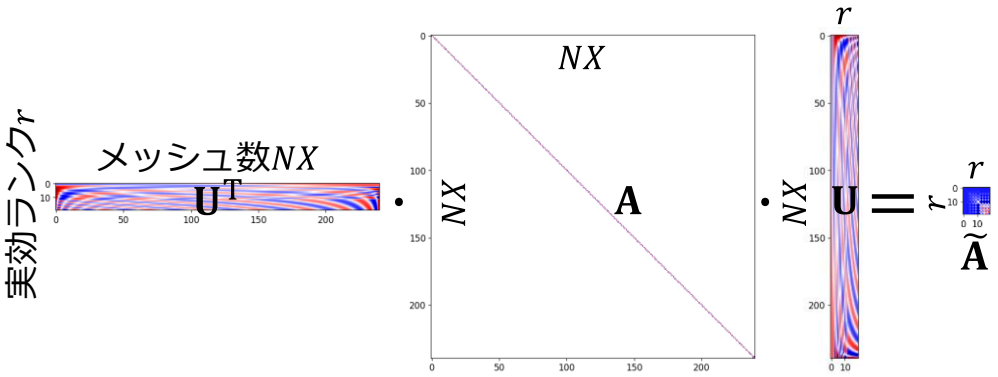

Figure 14 POD による係数行列**A**の圧縮例

式(101)でどれだけ次元圧縮することができるのか視覚的にイメージを掴むため、今回の問題を中性子拡散計算で解くことを想定して、空間メッシュ数$NX = 240$から、POD 基底の展開次数が$r = 23$まで次元圧縮した場合について、係数行列**A**の圧縮を可視化した例を Figure 14 に示す。圧縮前の係数行列**A**はサイズが大きな三重対角の疎行列であったが、行列**A**の左右からサンドイッチのように POD 基底ベクトル行列**U**で挟み込むことによって、遥かにサイズが小さな密行列に圧縮できることが分かるだろう。

例えば、n元連立方程式(係数行列サイズが$n \times n$)の数値解を何の工夫もない単純な直接解法(Gauss の消去法)で解いた場合には、n^3に比例した計算量が必要となる。従って、係数行列$\mathbf{A}$のサイズが240×240から、$\tilde{\mathbf{A}} = 23 \times 23$まで次元削減できる場合には、直接解法による連立方程式の数値計算量を$(23/240)^3 \approx 0.0009$、約 1000 分の 1 まで抑えることができると期待できる。こうして、対象とする計算条件について、次元圧縮した連立方程式(101)を解くことで POD 展開係数ベクトル$\vec{f}$の数値解が得ることができれば、式(100)のように$\vec{f}$の左から$\mathbf{U}$をかけることで、全中性子束$\vec{\phi}$の空間分布を再構成することができる。

辻は「POD を適用した中性子輸送計算手法」の実現に向け更に理解を深めるために、圧縮係数行列$\tilde{\mathbf{A}}$を求めるための演算式(101)の意味について、辻なりの言葉で考察することにした。

まず、次元圧縮前の全中性子束$\vec{\phi}$の列ベクトルに対して左から係数行列$\mathbf{A}$を作用させて得られる$\mathbf{A}\vec{\phi} \in \mathbb{R}^{NX \times 1}$は、「単位時間当たりに消滅する中性子数(neutrons/cm^3/s)の空間分布」を物理的に表している。それを踏まえれば、全中性子束$\vec{\phi}$の代わりにi次の POD 基底ベクトル$\vec{u}_i$を与え、左から係数行列$\mathbf{A}$を作用させた場合には、$\mathbf{A}\vec{u}_i \in \mathbb{R}^{NX \times 1}$という列ベクトルは「全中性子束分布の形状が$\vec{u}_i$であった時に、単位時間当たりに消滅する中性子数の空間分布」に相当するだろう。

さらに、既に式(98),(103)により示したように、全中性子束$\vec{\phi}$や中性子源$\vec{S}$の列ベクトルに左から$\mathbf{U}^{\mathrm{T}}$をかける演算は「各次数の POD 展開係数を一度に求める」操作に相当している。あるいは、成分を分解して、$\vec{\phi}$や$\vec{S}$の列ベクトルに左からj次の POD 基底ベクトルを転置した行ベクトル$\vec{u}_j^{\mathrm{T}}$をかける演算(ベクトルの内積)は、「i次数の POD 展開係数を個別に求める」操作に相当している。

従って、$\vec{u}_j^{\mathrm{T}}\mathbf{A}\vec{u}_i$のように、係数行列$\mathbf{A}$を左/右から$\vec{u}_j^{\mathrm{T}}$と$\vec{u}_i$をかける演算は「全中性子束分布の形状が$\vec{u}_i$であった時に、単位時間当たりに消滅する$j$次の POD 展開係数」に相当するようなスカラー量と考えることができるだろう。つまり、上述した POD 基底ベクトル$\vec{u}_i$の消滅率がj次の POD 展開係数に与える影響を、一度の行列演算で簡潔に表現した式が$\mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{U}$に他ならない。あるいは、「高次元空間上の全中性子束$\vec{\phi}$に関する関係式を真面目に考える代わりに、より低次元空間上の POD 展開係数$\vec{f}$に関する関係式に変換する操作」と言い換えても良いかもしれない。以上で考察したような演算により、POD 展開係数ベクトル$\vec{f}$を求めるための支配方程式(101)を得ることができる。

💡決定論的手法において空間離散化誤差を減らすためには、空間メッシュを詳細に分割する必要がある。しかし、空間メッシュ数が増加するほど、離散化された中性子拡散/輸送計算において解くべき連立方程式のサイズは大規模とならざるを得ない。この場合、各空間メッシュの全中性子束$\vec{\phi}$を直接求めるのではなく、POD 基底ベクトル$\vec{u}_i$が張る低次元の空間上に解くべき問題を投影することで、取り扱うデータ量を大幅に削減でき、効率良く問題を解くことができる。こうして低次元空間上で求められた数値解に、POD 基底ベクトル行列$\mathbf{U}$を作用させることで、もともとの高次元空間上の解を再構成する流れとなる。

なお、高次元の問題を低次元に投影して効率良く解くという考え方は、大規模疎行列の固有値・固有関数を求める数値計算手法の 1 つである「Implicitly Restarted Arnoldi Method

(IRAM)[31]」にも通ずる考え方であるため、記憶に留めておくと良い。

5.5. PODを適用した中性子輸送計算

辻が習得しているROMに関する知識も発展途上であり、この問題解決に費やすことができる時間にも制限があったため、残念ながら「PODを適用した中性子輸送計算手法」そのものを辻は実装することはできなかった。

辻が文献調査した結果として、以下のような方法論により POD を適用した中性子輸送計算手法も実装できそうな見込みがあることも分かったので、備忘録として残しておく。

① 行列演算形式による中性子輸送計算に対するPODの適用

例えば、S_N法やMOCにおいて、各空間メッシュの境界面において離散化した角度中性子束$\psi_{m,i-1/2}$を全てメモリーに保持して、$\psi_{m,i-1/2}$に関する大規模な連立方程式を解くことで$\psi_{m,i-1/2}$の数値解を求める計算プログラムを作成したとしよう。この時、全メッシュ境界面における全飛行方向の角度中性子束を並べた列ベクトル$\vec{\psi} \in \mathbb{R}^{\{NM(NX+1)\}\times 1}$に関して、式(49)と同様に、行列演算形式で連立方程式を表すことができるだろう。

$$\mathbf{A}_{\mathrm{tr}}\vec{\psi} = \frac{1}{2}\vec{S} \tag{104}$$

上式において、$\mathbf{A}_{\mathrm{tr}} \in \mathbb{R}^{\{NM(NX+1)\}\times\{NM(NX+1)\}}$は角度中性子束に対する係数行列、$\vec{S}$は外部中性子源の列ベクトル、をそれぞれ表す。上述のように行列演算形式で記述することができれば、中性子拡散計算の場合と同様に、角度中性子束$\vec{\psi}$に関する POD 基底ベクトルを求めて、POD 展開係数に関する小規模な連立方程式へと変換して効率良く解けば良い。

この方法は、PODを適用した中性子拡散計算の方法を素直に中性子輸送計算に適用しており非常に分かりやすいが、角度中性子束$\vec{\psi}$を陽に取り扱うため、保持すべきデータ量が膨大となってしまう点が課題となる。

② 全中性子束を未知数とした中性子輸送計算に対するPODの適用[24]

積分型中性子輸送方程式の数値解法の一種である「衝突確率法」を用いた場合、今回の問題で解くべき連立方程式は以下のような連立方程式の形式で表現することができる。

$$\mathbf{T}\vec{\phi} = \mathbf{P}\vec{q} \tag{105}$$

$$\vec{q} \equiv \frac{\Sigma_{\mathrm{s}}\vec{\phi} + \vec{S}}{2} \tag{106}$$

上式において、$\mathbf{T} \in \mathbb{R}^{NX\times NX}$は衝突を表す係数行列、$\mathbf{P} \in \mathbb{R}^{NX\times NX}$はある領域$j$で発生した中性子が領域$i$で衝突する確率を表す係数行列、$\vec{q}$は外部中性子源と散乱中性子源からなる中性子源の列ベクトル、をそれぞれ表す。衝突確率法の場合には、中性子拡散計算(式(49))で扱う係数行列**A**とは異なっているものの、結局のところ解くべき未知数そのものは全中性子束$\vec{\phi}$となっている。従って、式(105)に現れる2つの係数行列$\mathbf{T}, \mathbf{P}$の左/右からPOD基底ベクトル行列$\mathbf{U}^{\mathrm{T}}, \mathbf{U}$をかけることで、次元圧縮した POD 展開係数に関する方程式を得ることができる。この方法の場

合、衝突確率行列**P**を求めるために指数積分関数が絡んだ複雑な計算が必要となる点や、衝突確率行列**P**が大規模な密行列となっている点、が注意すべき点となる。特に、本問題に適用する場合には、外部境界面におけるアルベド境界条件の取り扱い方法が難しく、現時点の辻には実装方法のイメージが湧かなかった。

あるいは式(79)で示されるように、中性子拡散計算における正味の中性子流$J_{i+1/2}$計算式を補正することで、中性子輸送計算の結果を再現するような全中性子束$\vec{\phi}$に関する連立方程式を立てることができる。この連立方程式のサイズは、通常の中性子拡散計算の場合と同じであり、前節で述べた方法論を、そっくりそのまま利用して次元削減することができる。ただし、この手法では、「解くべき計算条件に対して補正係数$D_{\mathrm{cor},i+1/2}$をどのように推定するか」という点が肝となる。先行研究[24]では、時間依存の中性子輸送計算に対して POD を適用しているが、粗い時間ステップの MOC 動特性計算結果を活用して POD 基底ベクトル行列を求めるのに加えて、各時刻における補正係数$D_{\mathrm{cor},i+1/2}$も拡散加速法を適用することで求めておき、より詳細な時間ステップにおける動特性計算を実施する際に、補正係数$D_{\mathrm{cor},i+1/2}$(より正確には、粗い時間ステップで得られた圧縮係数行列)を内挿する、といった巧妙な手法を採用することで極めて効率的な時間依存 POD 中性子輸送計算を実現している。

③　次元圧縮演算と等価な transport sweep の活用[32]

本手法については、辻も未だ勉強中であるため、詳細については先行研究[32]を参考いただきたい。

辻が理解した範囲で、以下で計算手順の要点のみ纏めておく。まず、全中性子束$\vec{\phi}$を効率良く展開可能な POD 基底ベクトル$\vec{u}_i$を求めた後、「通常の全中性子束$\vec{\phi}$の代わりに$\vec{u}_i$を与えた散乱中性子源」「$\vec{\phi}$の代わりに$\vec{u}_i$を与えた核分裂中性子源」をそれぞれ計算する。こうして得られた POD 基底ベクトル$\vec{u}_i$に対応する固有の散乱/核分裂中性子源を「仮想的な固定源」として与えた中性子輸送計算(transport sweep)を実施し、得られた全中性子束の計算結果に対して$\vec{\phi}$基底ベクトル行列**U**を作用させることで、次元圧縮した連立方程式を得る(係数行列を推定する)方法論となっている。

直観的理解としては、「POD を適用した中性子拡散計算の場合で考えると、行列演算$\mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{U}$と等価な演算となるよう、特殊な中性子拡散計算を各基底ベクトル$\vec{u}_i$に対して実施する方法」と表現しても良いかもしれない。

6.　おわりに

本資料では、筆者が過去に報告した口頭発表[33]の内容「炉物理計算に対する ROM の適用例」の補完として、原子炉物理分野において過去に適用されてきた「工学的知見に基づいた ROM」や、「近年、活用されつつある ROM 適用例」について、筆者が知る限りの範囲内で理論や計算手法の概説を試みた。

各章の語り部として仮想的に設定した 3 名のエンジニアは、それぞれが培ってきた専門的な

知識を武器にして、3 者 3 様の方法論で解くべき問題を簡略化(次元削減、縮約)することで、問題解決に取り組んだ、と言える。炉物理計算手法は、偉大な先人達が積み上げてきた「工学的知見に基づいた ROM」に関するノウハウの塊であり、これを習得するには理論・数値解析手法に関する知識を深める必要がある。一方で、高速かつ正確な炉物理計算を実現するためには、既存の炉物理計算手法を鵜呑みにするだけでなく、本資料で題材として取り上げた「POD による ROM」のような、打開策となり得る方法論を貪欲に取り入れていく必要もある。言うまでもなく、炉物理計算に関する基礎知識が無ければ、「POD による ROM」といった方法論も適切に機能せず、宝の持ち腐れとなりかねない。

すなわち、「ある特定領域の専門的知識のみだけでは、達成可能な水準にも限界がある」という点も、本資料を通じて筆者が伝えたかった内容となっている。例えば、仮に、彼ら 3 名のエンジニアが互いに協力して補うことができたならば、例えば「pin-by-pin 全炉心体系を対象とした高速な中性子輸送計算コードの開発」といった、挑戦的課題についても、解決の糸口をつかむことができると期待している。

本資料では、1 次元平板の均質体系における固定源計算を題材として ROM の説明を試みたため、十分に盛り込むことができなかった説明もある。例えば、「工学的知見に基づいた ROM」については、以下に挙げるような、本資料内で語り尽くせなかった巧みな手法が幾つも存在している。興味がある読者は、各自文献調査してみて欲しい。

- ✓ 決定論的手法における多段階計算手順[2]：例えば、「①燃料棒単位の 1 次元 or2 次元セル計算→②2 次元集合体計算→③3 次元炉心計算」の順に、詳細条件で離散化しやすい小規模体系から、粗い条件で離散化した大規模体系へと、段階的に計算を実施する方法論。
- ✓ Simplified P3(SP3)近似[34], [35] : 3 次元xyz体系の場合、角度中性子束$\psi(\vec{\Omega})$は球面調和関数$Y_{lm}(\vec{\Omega})$で展開することになるが、(l,m)が高次になるにつれて多く展開係数が必要となる。SP3 法は、$l=3$次までの角度中性子束展開係数のうち一部のみを考慮した ROM であり、pin-by-pin 中性子輸送計算手法[36]として活用されている。
- ✓ 一点炉動特性計算[37] :「全中性子束の相対的な空間分布やエネルギー分布が、k_{eff}固有値方程式の基本モード形状(過渡変化前の臨界時)と同じである」と近似し、中性子束の大きさ(∝基本モード成分の展開係数)のみが時間変化すると簡略化した、極めて実用的な ROM。
- ✓ 一次摂動論に基づいた感度係数評価[38]:未摂動条件におけるk_{eff}固有値計算および随伴固有値計算を 2 回だけ実施することで、核データの変化量に対するk_{eff}変化量を効率良く計算する手法。

「近年、活用されつつある ROM 適用例」についても全てを網羅することは容易ではないが、以下のような方法が既に検討されていることを頭に入れておくと、新たな閃きを生むかもしれない。

- ✓ 動的モード分解(Dynamic Mode Decomposition)による加速法[39]：過去の炉物理計算手法で言うと、チェビシェフ加速法に相当する手法と表現しても良い。反復計算で得られた全中

性子束$\vec{\phi}^n$について、過去から現在まで遡って得られた結果$(\vec{\phi}^{n-r}\cdots,\vec{\phi}^{n-2},\vec{\phi}^{n-1})$を時空間データ行列として纏める。こうして整理したデータ行列に対して特異値分解と固有値分解を駆使することで、$\vec{\phi}^n$を複数の指数関数成分の和で表現し、反復回数$n\to\infty$に相当する収束解を近似的に推定する。

- ✓ RSE 法(Resonance calculation using energy Spectrum Expansion method)[40], [41]：共鳴計算における中性子束エネルギー分布$\phi(E)$に対して POD を応用した例。
- ✓ モンテカルロ計算における POD 展開係数タリー[42]：collision estimator などにより、モンテカルロ計算の枠組みで POD 展開係数の統計量を求める手法

7. 参考文献

[1] 小林 啓祐, “原子炉物理,” コロナ社 (1996).

[2] D. G. Cacuci (Ed.), “Handbook of Nuclear Engineering,” Springer New York, NY (2010); DOI: 10.1007/978-0-387-98149-9

[3] 山本 章夫, “Reduced Order Model とシミュレーション計算,” 第 50 回炉物理夏期セミナーテキスト (2018).

[4] A. Fick, “Ueber Diffusion,” *Annalen der Physik*, **170**(1), pp. 59‐86 (1855); DOI: 10.1002/andp.18551700105

[5] R. D. Lawrence, “Progress in Nodal Methods for the Solution of the Neutron Diffusion and Transport Equations,” *Prog. Nucl. Energy*, **17**(3), pp.271–301 (1986); DOI: 10.1016/0149-1970(86)90034-X

[6] Google Colaboratory, https://colab.research.google.com/notebooks/intro.ipynb?hl=ja (2024/6/18 アクセス).

[7] OpenMC: Theory and Methodology, https://docs.openmc.org/en/stable/methods/index.html (2024/6/18 アクセス)

[8] 長家 康展, “Python を利用した核計算 (2)確率論手法,” 第 50 回炉物理夏期セミナーテキスト (2018).

[9] L. Devroye, “Non-Uniform Random Variate Generation,” Springer New York, NY (1986); DOI: 10.1007/978-1-4613-8643-8

[10] H. Fischer, “A History of the Central Limit Theorem: From Classical to Modern Probability Theory,” Springer New York, NY (2011); https://doi.org/10.1007/978-0-387-87857-7

[11] 原子炉物理分野の教育用資料集, https://rpg.jaea.go.jp/else/rpd/others/study/index.html (2024/6/19 アクセス).

[12] 山本 章夫, “ 拡散方程式の数値解法の基礎,” 第 36 回炉物理夏期セミナーテキスト (2004).

[13] 遠藤 知弘, “Python を利用した核計算 (1)決定論的手法,” 第 50 回炉物理夏期セミナーテキスト (2018).

[14] 遠藤 知弘, “Google Colaboratory を用いた 拡散計算コード演習,” 第 52 回炉物理夏期セミナーテキスト (2021).

[15] W. H. Press et al., Numerical Recipes: The Art of Scientific Computing, 3rd Ed, Cambridge University Press, Cambridge, UK (2007).

[16] Numba, https://numba.pydata.org/ (2024/6/19 アクセス).

[17] 奥村 啓介, 他, “SRAC2006; 総合核計算コードシステム,” JAEA-Data/Code 2007-004, pp. 221–245; DOI: 10.11484/jaea-data-code-2007-004

[18] R. E. Alcouffe et al., “PARTISN: A Time-Dependent, Parallel Neutral Particle Transport Code System,” LA-UR-08-07258 (Revised: November 2008).

[19] OpenMOC: Theory and Methodology, https://mit-crpg.github.io/OpenMOC/methods/index.html (2024/6/19 アクセス).

[20] M. Jarrett et al., “Analysis of Stabilization Techniques for CMFD Acceleration of Neutron Transport Problems,” *Nucl. Sci. Eng.*, **184**(2), pp. 208–227 (2016); DOI: 10.13182/NSE16-51.

[21] A. Zhu et al., “An Optimally Diffusive Coarse Mesh Finite Difference Method to Accelerate Neutron Transport Calculations,” *Ann. Nucl. Energy*, **95**, pp. 116–124 (2016); DOI: 10.1016/j.anucene.2016.05.004

[22] G. Berkooz et al., “The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows,” *Annu. Rev. Fluid Mech.*, **25**(1), pp. 539-575 (1993); DOI: 10.1146/annurev.fl.25.010193.002543

[23] K. Tsujita et al., “Fast Reproduction of Time-Dependent Diffusion Calculations Using the Reduced Order Model Based on the Proper Orthogonal and Singular Value Decompositions,” *J. Nucl. Sci. Technol.*, **58**(2), pp. 173–183 (2021); DOI: 10.1080/00223131.2020.1814891

[24] K. Tsujita et al., “Efficient Reduced Order Model Based on the Proper Orthogonal Decomposition for Time-Dependent MOC Calculations,” *J. Nucl. Sci. Techonol.*, **60**(3), pp. 343–357 (2023); DOI: 10.1080/00223131.2022.2097963

[25] NumPy v2.0 Manual: numpy.linalg.matrix_rank, https://numpy.org/doc/stable/reference/generated/numpy.linalg.matrix_rank.html; (2024/6/23 アクセス).

[26] NumPy v2.0 Manual: numpy.linalg.svd, https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html; (2024/6/23 アクセス).

[27] N. Halko et al., “Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions,” *SIAM Rev.*, **53**(2), pp. 217–288 (2011); DOI: 10.1137/090771806

[28] M. Ito et al. “Preliminary Study on Two-Dimensional SP3 Calculation Based on POD-Local/Global Iterations,” *Proc. PHYSOR2024*, San Francisco, CA, April 21–24, 2024, pp. 1612–1621; https://www.ans.org/pubs/proceedings/article-55630/

[29] R. Bellman, “Dynamic Programming,” Princeton University Press, Princeton, NJ (1957).

[30] R. J. J. Stamm’Ler and M. J. Abbate, “Methods of Steady-State Reactor Physics in Nuclear Design,” *Academic Press*, London (1983).

[31] R. B. Lehoucq et al., “Deflation Techniques for an Implicitly Restarted Arnoldi Iteration,” *SIAM J. Matrix Anal. & Appl.*, **17**(4), pp. 789–821 (1996); DOI: 10.1137/S0895479895281484

[32] P. Behne et al., “Minimally-Invasive Parametric Model-Order Reduction for Sweep-Based Radiation Transport,” *J. Comput. Phys.*, **469**, 111525 (2022); https://doi.org/10.1016/j.jcp.2022.111525

[33] 遠藤 知弘, “炉物理計算に対する ROM の適用例,” 日本原子力学会 2022 年秋の大会, 1F PL03, 茨城大学日立キャンパス, 2022 年 9 月 7 日; https://confit.atlas.jp/guide/event/aesj2022f/subject/1F_PL03/advanced

[34] E. W. Larsen et al., “Asymptotic Derivation of the Multigroup P_1 and Simplified P_N Equations with Anisotropic Scattering,” *Nucl. Sci. Eng.*, **123**(3), pp. 328–342 (1996); https://doi.org/10.13182/NSE123-328

[35] A. Yamamoto et al., “Recent Developments in the GENESIS Code Based on the Legendre Polynomial Expansion of Angular Flux Method,” *Nucl. Eng. Technol.*, **49**(6), pp. 1143–1156 (2017); https://doi.org/10.1016/j.net.2017.06.016

[36] M. Tatsumi et al., “Advanced PWR Core Calculation Based on Multi-group Nodal-transport Method in Three-dimensional Pin-by-Pin Geometry,” *J. Nucl. Sci. Technol.*, **40**(6), pp. 376–387 (2003); DOI: 10.1080/18811248.2003.9715369

[37] A. F. Henry, “Nuclear-Reactor Analysis,” MIT Press, Cambridge, MA (1975).

[38] “SCALE 6.3.1 User Manual: 6.6. SAMS: Sensitivity Analysis Module for SCALE,” https://scale-manual.ornl.gov/sams.html (2024/6/26 アクセス).

[39] J. A. Roberts et al., “Acceleration of the Power Method with Dynamic Mode Decomposition,” *Nucl. Sci. Eng.*, **193**(12), pp. 1371–1378 (2019); https://doi.org/10.1080/00295639.2019.1634928

[40] R. Kondo et al., “A New Resonance Calculation Method Using Energy Expansion Based on a Reduced Order Model,” *Nucl. Sci. Eng.*, **195**(7), pp. 694–716 (2021); DOI: 10.1080/00295639.2020.1863066

[41] R. Kondo et al., “Improvements in Computational Efficiency for Resonance Calculation Using Energy Spectrum Expansion Method,” *Nucl. Sci. Eng.*, **196**(7), pp. 769–791 (2022); DOI: 10.1080/00295639.2021.2025297

[42] R. Kondo et al., “Flux Distribution Tallies Using Proper Orthogonal Decomposition in Monte Carlo Calculations,” *J. Nucl. Sci. Technol.*, (2024); DOI: 10.1080/00223131.2024.2365445