

Fortran プリプロセッサの開発と

球 MHD シミュレーション

細山田 真也

要旨

エネルギーが高い状態から低い状態に系が自発的に遷移する現象、緩和は様々な物理系で見られる興味深い現象である。それは磁気流体力学（MagnetoHydroDynamics, MHD）のモデルにおいても例外ではない。MHD 流体の緩和については Taylor 理論と呼ばれる緩和理論が確立されているが、Taylor 理論では緩和状態において磁気エネルギーだけが存在し、流体の運動エネルギーは無視できると仮定している。MHD 系の自然現象では流れを持つ緩和状態が存在するので、この理論は一般的に適応されない。本研究の最終的な目的は、古典的な Taylor 理論を拡張し、磁気エネルギーだけでなく流れのエネルギーを持つ緩和状態を説明する MHD 緩和理論を構成することである。本研究では球状容器内の MHD 流体の振る舞いについて計算機シミュレーションを行った。計算モデルは以下の通りである。半径 1 の球面に囲まれた球領域に MHD 流体が満たされているとし、球面調和関数で表される初期磁場を与える。境界条件は磁場については完全導体境界、流れ場については剛体境界、温度場については断熱境界を仮定した。全球でシミュレーションを行う際に問題となる格子点集中問題を解決するための離散化手法として Yin–Yang–Zhong 格子を用いた。またシミュレーションコード実装を効率的に行うために Fortran プリプロセッサ efpp を開発した。シミュレーションの結果、球面調和関数特有の緩和構造を発見することができた。

目次

1	序論	1
1.1	球 MHD シミュレーション	1
1.2	緩和現象	2
1.3	磁気流体力学と MHD 方程式	3
1.4	Taylor 理論	4
1.5	先行研究	4
2	球の数値計算手法	5
2.1	時間積分法	6
2.2	空間離散化手法	7
2.3	球座標格子の問題点	7
2.4	Yin–Yang 格子の利点と Zhong 格子の導入	9
2.5	Yin–Yang–Zhong 格子の座標変換	11
2.6	Yin–Yang–Zhong 格子のベクトル変換と補間	12
3	Fortran プリプロセッサの開発	15
3.1	現代的 Fortran の問題点	15
3.2	eFortran と efpp	16
3.3	具体例	17
3.4	サンプルコード	21
4	計算モデルとコードの実装	22
4.1	計算手法	22
4.2	初期条件	23
4.3	境界条件	25
4.4	パラメータ設定	25
4.5	eFortran 版球 MHD コードの概観	26
5	シミュレーション結果	27
5.1	エネルギーのグラフ	27

5.2	緩和状態の三次元構造	29
6	結論	33
	謝辞	33
	参考文献	34

1 序論

1.1 球 MHD シミュレーション

シミュレーションとは現実上で行われる実験をコンピュータ上で疑似的に行うことである。その利点は低コストであること、安全であることなどが挙げられ、今や理論・実験科学に続く「第3の科学」として多くの学術・応用分野で必要不可欠な研究手法として位置づけられている [プラズマシミュレーション]。本研究で扱うシミュレーションは球状容器中の MHD 流体の振る舞いについてである。MHD 流体とは磁気流体力学 (MagnetoHydroDynamics) 方程式に従う流体のこと、別名を電気伝導性流体という。例えば、プラズマは MHD 流体の代表格である。プラズマは、固体、液体、気体に続く第4の物質状態と呼ばれ、地上で意識することは少ないが、宇宙や天体で物質のとる普遍的な状態である [プラズマシミュレーション]。また、液体鉄も MHD 流体の一つである。地球内部の大部分を占めるマントル層の下には外核と呼ばれる液体鉄の領域があるが、その流れによって地球の磁場が形成・維持されていると考えられている [地球ダイナモ研究のこれまでとこれから]。

つまり、MHD 流体のシミュレーションを扱うことによって、将来的に MHD 流体そのものだけでなく地球やその他天体の振る舞いが理解できるようになり、宇宙に関する多くの謎が解明される可能性がある。球を想定している理由は、天体の多くは球状であるから、また角運動量が保存する、境界に流れが妨げられないなど球体ならではの特徴があるからである。

数値シミュレーションに必要な材料は以下の通りである。

- (1) 対象にする問題
- (2) 問題が従う微分方程式
- (3) 微分方程式の時間微分および空間微分を離散的に近似する手法

(1) 本研究が対象にする問題は MHD 流体の基礎的な性質とは何か、特に MHD 流体の緩和現象 (1.2 参照) に見られる特徴は何かということである。そして、(2) MHD 流体が従う方程式は MHD 方程式 (1.3 参照) であり、(3) その方程式の時間微分と空間微分を離散的に近似する手法はそれぞれ4次ルンゲ・クッタ法 (2.1 参照) と有限差分法 (2.2 参照) を用いた。

1.2 緩和現象

緩和 (relaxation) とは一般的に系が不安定な状態から安定な状態に移り変わることを言う。散逸を持つ系が閉じている (=系が外部とエネルギーや物質を交換しない) 場合、その系は最終的に熱力学的な平衡状態となるが、そこには至るには散逸時間程度の時間がかかる。そして多くの場合、その散逸時間のスケールは系に存在する他の時間スケールと比べて極めて大きい。つまり、ある系が完全に緩和するには時間がかかるのが一般的である。しかし、多くの物理系で散逸時間よりも短い時間スケールで系がほぼ定常状態に落ち着く現象が観察されている。それは完全な定常状態ではないが、散逸時間程度の時間スケールで維持される準定常状態である。そのような物理系は、どのような初期状態から出発しても数少ない特定の準定常状態に落ち着く。そしてこの状態は何らかの特徴的な構造を持っている場合が多い。

プラズマの系においても例外でなく、自然界や実験室でこの緩和現象が確認され、多くの研究者を惹きつけてきた。この緩和は 2 次元系でも起き得るが、本質的で興味深い点はその 3 次元性にある。スーパーコンピュータ (スパコン) が登場した 80 年代半ばより、3 次元磁気流体モデルを用いた緩和のシミュレーション研究がようやく可能になった。当初は $50 \times 50 \times 50$ 程度の空間メッシュを用いた計算が限度であったが、その後のスパコンの能力の飛躍的増加により、現在では 1 次元方向に 1,000 メッシュ以上をもつ 3 次元計算が可能となっている。これまでに、様々な系で発生するプラズマの緩和現象が計算機シミュレーションにより解明されてきている [1]。

磁気流体力学 (Magnetohydrodynamics, MHD) モデルに基づくプラズマの緩和理論は Woltjer [2] と Taylor [3] によって主に構築された。MHD モデルでは、磁気エネルギー、運動エネルギー、熱エネルギーの 3 つのエネルギーが存在する。Woltjer-Taylor 理論ではこのうち磁気エネルギーだけに注目しその緩和状態を考えている。運動エネルギーは無視する、すなわち緩和状態でプラズマの流れは止まっているという仮説は Taylor が想定していたプラズマの閉じ込めを目的とした核融合の実験装置では妥当なものであるが、全てのプラズマ (あるいは MHD) 系で成り立つわけではない。むしろ、緩和状態でも流れが生じているほうが自然界的に一般的である。そこで本研究では、流れの運動エネルギーを含む MHD 流体の動きを計算機シミュレーション [4] で調べることによって、より詳細な MHD 流体の緩和理論を発展させることを目標とする。

1.3 磁気流体力学と MHD 方程式

ここでは本研究で扱われる方程式、MHD 方程式について説明する。

MHD 方程式とは MHD 流体の振る舞いを表した式のことで、圧縮性の磁気流体の運動は以下の MHD 方程式で記述される [5]。

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathbf{f}, \quad (1)$$

$$\frac{\partial \mathbf{f}}{\partial t} = -\nabla \cdot (\mathbf{v}\mathbf{f}) - \nabla p + \mathbf{j} \times \mathbf{b} + \mu \left[\nabla^2 \mathbf{v} + \frac{1}{3} \nabla(\nabla \cdot \mathbf{v}) \right] + 2\mathbf{f} \times \boldsymbol{\Omega}, \quad (2)$$

$$\frac{\partial p}{\partial t} = -\mathbf{v} \cdot \nabla p - \gamma p \nabla \cdot \mathbf{v} + (\gamma - 1) [\kappa \nabla^2 T + \eta \mathbf{j}^2 + \Phi], \quad (3)$$

$$\frac{\partial \mathbf{a}}{\partial t} = \mathbf{v} \times \mathbf{b} + \eta \nabla^2 \mathbf{a} + \nabla(\phi - \eta \nabla \cdot \mathbf{a}), \quad (4)$$

ここで、

$$\mathbf{f} = \rho \mathbf{v}, \quad (5)$$

$$p = C_v(\gamma - 1)\rho T, \quad (6)$$

$$\mu_0 \mathbf{j} = \nabla \times \mathbf{b}, \quad (7)$$

$$\mathbf{b} = \nabla \times \mathbf{a}, \quad (8)$$

$$\Phi = 2\mu \left[\epsilon_{ij} \epsilon_{ij} - \frac{1}{3} (\nabla \cdot \mathbf{v}) \right], \quad (9)$$

$$\epsilon_{ij} = \frac{1}{2} \left[\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right], \quad (10)$$

である。

ρ は質量密度、 \mathbf{f} は質量流束、 p は圧力、 \mathbf{a} は磁場のベクトルポテンシャル、 \mathbf{v} は速度場、 \mathbf{b} は磁場、 \mathbf{j} は電流密度、 T は温度、 γ は比熱比、 C_v は定積比熱、 $\boldsymbol{\Omega}$ は角運動量、 μ は粘性係数、 η は電気抵抗、 κ は熱伝導率、 Φ は散逸関数、 ϕ は任意のスカラーフィールである。本研究では $\phi = \eta \nabla \cdot \mathbf{a}$ というゲージを用いた。

式 (1) は質量保存則、式 (2) は運動方程式、式 (3) は圧力の式、式 (4) は磁場の誘導方程式を表している。

1.4 Taylor 理論

MHD 方程式より、電気抵抗 $\eta = 0$ のとき磁気ヘリシティ

$$K = \int \mathbf{a} \cdot \mathbf{b} \, dV \quad (11)$$

は保存することがわかる。磁気ヘリシティとは磁力線のねじれ具合、繋がり具合を示す指標である。J.B.Taylor は、緩和状態での領域全体の磁気エネルギー

$$W_m = \int \frac{b^2}{2} \, dV \quad (12)$$

が最小となる場合、

$$\nabla \times \mathbf{b} = \mu \mathbf{b} \quad (13)$$

を満たすことを示した [3][6]。このとき式 (26)、式 (7) より磁場中に働くローレンツ力 $\mathbf{j} \times \mathbf{b}$ は

$$\mathbf{j} \times \mathbf{b} = \frac{\mu}{\mu_0} \mathbf{b} \times \mathbf{b} = \mathbf{0} \quad (14)$$

となる。これは緩和状態において領域内の磁場による力は全く働くことを示している。この状態のことを force-free 状態という。

この理論においては、MHD 流体の流れのエネルギーと熱エネルギーは考慮されていない。本研究では流れがある MHD 流体の緩和を考える。

1.5 先行研究

本研究の先行研究として、初期磁場がトロイダル磁場 (Fig.) の時の緩和状態を本研究室の山本らが調べた。その結果は以下の通りである。

- 初期流れの有無に関わらず、磁気エネルギーの緩和過程において磁気ヘリシティがほぼ保存される。
- 初期状態に剛体回転流れを与えた場合には、運動エネルギーが磁気エネルギーに変換されるダイナモ効果がみられ、初期速度場がゼロの場合と比べて緩和状態での磁気エネルギーが高くなった。
- 初期速度場がゼロの場合においては、緩和状態に正方形の各頂点に渦ができるという特徴的な構造が見られた。

本研究と先行研究との大きな違いは初期磁場である。先行研究ではリング磁場を用いていたが、本研究では球面調和関数を用いた。その理由はリング磁場は構造が単純であるが、時間経過で生じる流れが速く、シミュレーションの発散の原因となるためである。球面調和関数を用いてリングを複数個用意するとその構造は複雑になるものの、長期的なシミュレーションが可能となる。

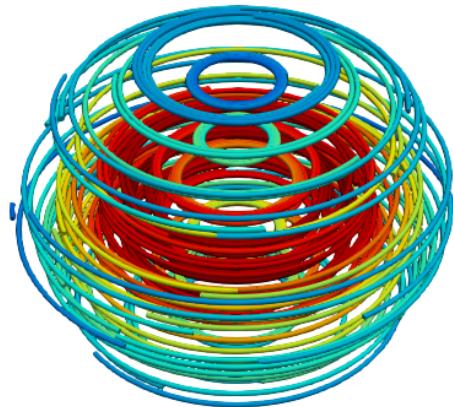


Fig.1 Troidal magnetic field in Yamamoto's simulation.

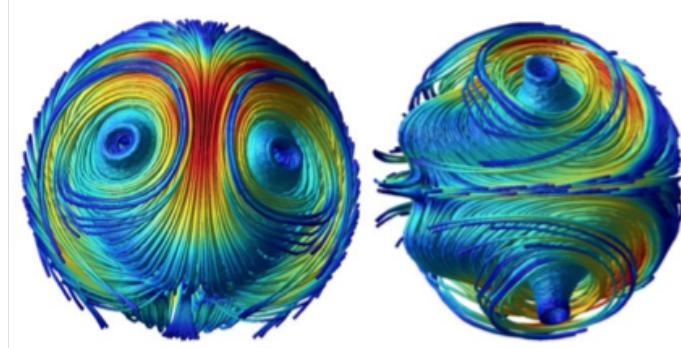


Fig.2 Relaxed magnetic field in Yamamoto's simulation.

2 球の数値計算手法

本研究は球体のシミュレーションを対象としているので、球ジオメトリの計算手法が必要不可欠である。この章では球の数値計算手法と本研究に用いられる座標格子「Yin–Yang–Zhong 格子」について説明する。

2.1 時間積分法

メモリーが有限であるコンピュータでは連続的な微分方程式を解くことができないので、微分方程式を離散化して近似する必要がある。そのうち時間微分を近似する代表的な手法に1次オイラー法と4次ルンゲ・クッタ法がある。以下よりそれについて説明する。

まず解きたい微分方程式が以下のように表せるとする。

$$\frac{dq(t)}{dt} = f(q, t) \quad (15)$$

ここで時間を以下のように離散化させる。

$$t_j = \Delta t \times j \quad (16)$$

$$q_j = q(t_j) \quad (17)$$

$$f_j = f(q_j, t_j) \quad (18)$$

j は 0 以上の整数である。1次オイラー法は () 式を以下のように近似する方法である。

$$\frac{q_{j+1} - q_j}{\Delta t} = f_j \quad (19)$$

この式を変形させると

$$q_{j+1} = q_j + \Delta t \times f_j \quad (20)$$

となる。また4次ルンゲ・クッタ法は () 式を以下のように近似する。

$$q_{j+1} = q_j + \frac{\Delta t}{6} k_1 + \frac{\Delta t}{3} k_2 + \frac{\Delta t}{3} k_3 + \frac{\Delta t}{6} k_4 \quad (21)$$

$$k_1 = f(q_j, t_j) \quad (22)$$

$$k_2 = f(q_j + \frac{\Delta t}{2} k_1, t_j + \frac{\Delta t}{2}) \quad (23)$$

$$k_3 = f(q_j + \frac{\Delta t}{2} k_2, t_j + \frac{\Delta t}{2}) \quad (24)$$

$$k_4 = f(q_j + \Delta t k_3, t_j + \Delta t) \quad (25)$$

1次オイラー法は単純だが1次精度 $\mathcal{O}(\Delta t)$ なので誤差が大きくなる。それに対し4次ルンゲ・クッタ法は4次精度 $\mathcal{O}(\Delta t^4)$ なので十分に高い。4次ルンゲ・クッタ法の問題点は計算量やメモリ量が多くなることだが、それらを削減する高速4次ルンゲ・クッタ法（ルンゲ・クッタ・ギル法など）のアルゴリズムが考案されているので大きな問題にならない。よって本研究では4次ルンゲ・クッタ法を用いることにする。

2.2 空間離散化手法

球内部の三次元空間を離散化する方法は大きく分けて 2 つある。1 つは球面調和関数（4.2.2 参照）などを用いて離散的に式を展開するスペクトル法、もう 1 つは空間を有限個の格子（グリッド）で分割する有限差分法である。流体のシミュレーションはスペクトル法を用いた研究が盛んである[]。なぜなら、スペクトル法は球ジオメトリへの適応が容易であり、比較的少ない展開モード数だと高い精度の解が得られるからである。しかし、球面調和関数を用いたスペクトル法には以下の 2 つの問題点がある：

- (1) 展開モード数 L, M での計算量は $\mathcal{O}(L^2M^2)$ であるため、高い空間解像度を得るために展開モード数を増やすと計算量が非線形に増大してしまう。
- (2) 数値計算の各ステップ毎に実空間とスペクトル空間の間で球面調和関数変換と逆変換を行う必要があるが、これらの処理にはプロセッサ間のグローバル通信が必要であり、並列計算に向かない。

有限差分法は上記 2 つの問題点を解消することができる。例えば、(1) に関して系の自由度を L, M とすると有限差分法の計算量は $\mathcal{O}(LM)$ と表されるので、指數関数的に計算量が増えることはない。また (2) に関して陽的な（未知の変数を既知の変数のみで求められることを陽的という）時間積分法を採用する限り、グリッドベースの方法が並列計算に適していることはよく知られている。

よって有限差分法を用いてシミュレーションを行いたいのだが、有限差分法を球ジオメトリに適応させるには計算上の問題がある。次節では球座標格子の問題点について説明する。

2.3 球座標格子の問題点

従来において、球殻系の流体シミュレーションには球座標格子 (r,θ,ϕ) が広く使われてきた。ここで r は原点からの距離（半径）、 θ は余緯度、 ϕ は経度である。球座標格子の低緯度表面付近は格子間隔がほぼ均一であり理想的なシミュレーション格子といえる。しかし、球座標格子には以下の 2 つの特異点がある [Fig.3]。

- (1) 極 ($\theta = 0, \pi$)
- (2) 球の中心 ($r = 0$)

これらの特異点によって、シミュレーションにおいて2つの問題が生じる。

(1) 特異点でのゼロ除算の問題

(2) 特異点に格子が集中していることによる計算コストの増加

(1) の問題については、シミュレーションにおいて座標特異点上に格子点を置く必要はなく、ずらして置くことが可能なので回避することができる。またロピタルの定理を利用し方程式を特異的でない形にして解くという方法もある。(1) の問題よりむしろ(2) の格子点が集中している問題のほうが深刻である。その理由について以下より説明する。

本研究では MHD 方程式を陽的な時間積分法である4次ルンゲ＝クッタ法を用いて計算する。陽的な計算手法を用いて計算する場合、以下の CFL(Courant-Friedrichs-Lowy) 条件を満たす必要がある。CFL 条件とはシミュレーション内の情報が伝播する速さは物理的に計算される波の伝播速度を超えてはいけないという制約条件である(この制約を破るとシミュレーションで得られる数値に物理的な意味がなくなる)。例えば、音波モードのみが存在する1次元の系における CFL 条件は以下の式で表される。

$$\Delta t < \frac{\Delta x}{C} \quad (26)$$

ここで Δt はシミュレーションの時間刻み幅、 Δx はシミュレーションの格子間隔、 C は音速である。この式からシミュレーションの格子間隔が小さいほど、より短い時間刻み幅で数値解析を行う必要があることがわかる。つまり、格子点が集中している特異点付近は時間刻み幅を小さくする必要があり、全ての格子点はその時間刻み幅に合わせなければならない。よって、格子点の集中は計算コストの増大につながりシミュレーションの時間発展を妨げる障害となる。

この格子点の集中を避けるために考案されたのが Yin-Yang-Zhong 格子である。

[?]

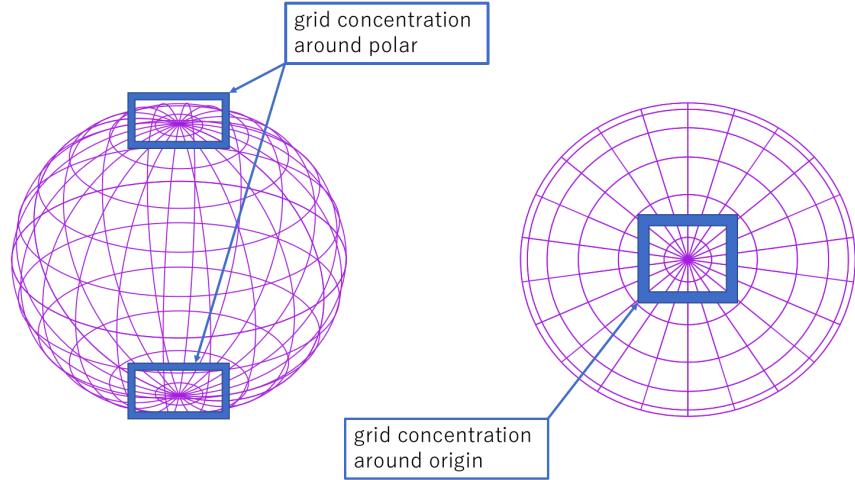


Fig.3 Spherical polar coordinates. Grid points concentrated around the poles and the origin.

2.4 Yin–Yang 格子の利点と Zhong 格子の導入

ここでは、Yin–Yang–Zhong 格子の前身となる「Yin–Yang 格子」(Kageyama & Sato [7])について説明する。Yin–Yang 格子とは、球座標格子の低緯度部分を切り取った格子を 2 つ組み合わせて球全体を覆う重合格子法の 1 つである。Yin–Yang 格子の構成要素は、球座標系の以下の余緯度 θ 、経度 ϕ を切り取ったものである。

$$\frac{\pi}{4} \leq \theta \leq \frac{3\pi}{4} \quad (27)$$

$$-\frac{3\pi}{4} \leq \phi \leq \frac{3\pi}{4} \quad (28)$$

Yin 格子は、球座標系の式 (2),(3) の範囲の格子である。Yang 格子は Yin 格子と同形であり、軸を回転させたものである。Yin 格子と Yang 格子の二つを組み合わせて球を覆うことによって外核を表現する。Yin–Yang 格子は球座標格子の利点である低緯度部分を用いているので、極付近への格子の集中を避けることができる。シミュレーションでは、Yin 格子、Yang 格子ともに球座標格子系で計算を行い、それぞれの境界値を求める場合には Yang 格子が本来、Yin 格子の軸を回転させたものであるこ

とを考慮し、それぞれの格子で補間を行う。Yin-Yang 格子を用いた球殻シミュレーションは様々な研究に応用されている。一方、Yin-Yang 格子を用いても球の原点付近での格子の集中は回避できない。そこで新たに考案されたのが「Zhong 格子」である。Fig.4 に Yin-Yang-Zhong 格子の構造を示す。Zhong 格子は、Yin-Yang 格子で覆われた球中心の空洞部分に導入されるカーテシアン座標格子である。Yin-Yang 格子による球殻部分を Shell、Zhong 格子による球中心部分を Core と呼ぶ。Shell の内縁半径を r_c とし、その内側に Core を配置する。Yin-Yang-Zhong 格子を用いることで、ほぼ均一な格子間隔で全球シミュレーションを行うことが可能となる。Yin-Yang-Zhong 格子を用いたシミュレーションでは、Yin 領域、Yang 領域、Zhong 領域で別々の並列プロセスが割り当てられる。Core 領域である $r = r_c$ の球全体を囲む、一辺が $2r_c$ の立方体を l-cube と呼ぶ。ユーザーは、l-cube の x-y-z 方向の分割数を指定する。分割したうちの 1 つの区画を s-cube と呼ぶ。Core 領域に少しでも重なっている部分を持つ s-cube には並列プロセスを割り当て、Core 領域と重なった部分を持たない s-cube には並列プロセスを割り当てない。これにより、Shell 領域、Core 領域それぞれに効率よく並列プロセスを割り当てることができる。

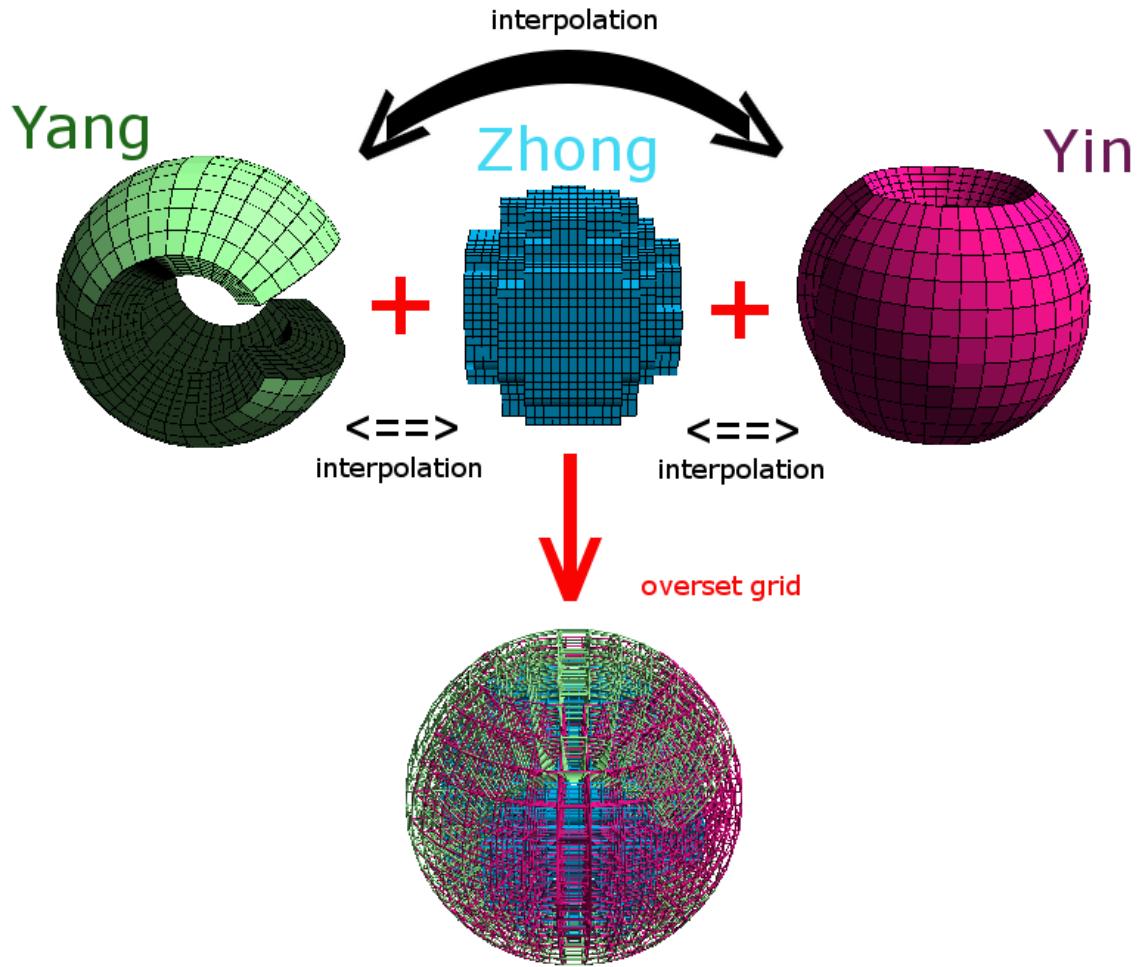


Fig.4 Construction of Yin-Yang-Zhong grid.

2.5 Yin-Yang-Zhong 格子の座標変換

Yin 格子と Yang 格子では、座標系が異なるが、元は同じ球座標系であるため、簡潔に座標変換を行うことができる。Yin 領域における直交座標系 (x^e, y^e, z^e) と Yang 領域における直交座標系 (x^n, y^n, z^n) には、

$$(x^e, y^e, z^e) = (-x^n, z^n, y^n) \quad (29)$$

$$(x^n, y^n, z^n) = (-x^e, z^e, y^e) \quad (30)$$

という関係がある。これを行列で表現すると、

$$\begin{pmatrix} x^e \\ y^e \\ z^e \end{pmatrix} = M \begin{pmatrix} x^n \\ y^n \\ z^n \end{pmatrix} \quad (31)$$

ここで、

$$M = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (32)$$

であり、この変換行列 M は $M^{-1} = M$ の関係を満たす。また式 (29)-(32) より球座標形式での座標の関係は、

$$r^e = r^n, \quad (33)$$

$$\sin \theta^e \cos \phi^e = -\sin \theta^n \cos \phi^n, \quad (34)$$

$$\sin \theta^e \sin \phi^e = \cos \phi^n, \quad (35)$$

$$\cos \phi^e = \sin \theta^n \sin \phi^n, \quad (36)$$

で与えられることがわかる。ここでの添字の n, e はそれぞれ Yin 座標と Yang 座標に対応する。

次に Zhong 格子について示す。Zhong 格子は、一般的なカーテシアン座標系なので、Yin 格子、Yang 格子と Zhong 格子の間の座標変換はよく知られた極座標とカーテシアン座標の座標変換の式、

$$x = r \sin \theta \cos \phi, \quad (37)$$

$$y = r \sin \theta \sin \phi, \quad (38)$$

$$z = r \cos \theta, \quad (39)$$

$$(40)$$

で与えられる。

2.6 Yin–Yang–Zhong 格子のベクトル変換と補間

Shellにおいて球殻が Yin 領域と Yang 領域の 2 つの領域に分割されているため、Yin 格子と Yang 格子の境界部分で値の相互補完を行う必要がある。スカラー値の補

間は単純だが、ベクトル値の補間はやや複雑である。これは Yang 格子は Yin 格子を回転させたものであり、ベクトルの方向がそれぞれの格子で異なるためである。

Fig.5 に Yin 格子と Yang 格子のベクトルの向きの関係を示す。図の緑色の格子が Yin 格子、青色の格子が Yang 格子を示している。図中の $\hat{\theta}^n$ と $\hat{\phi}^n$ は、それぞれ Yin 格子の緯度方向、経度方向のベクトルであり、 $\hat{\theta}^e$ と $\hat{\phi}^e$ はそれぞれ Yang 格子の緯度方向、経度方向の単位ベクトルである。Yin 格子上でのベクトル場を $(v_r^n, v_\theta^n, v_\phi^n)^t$ 、Yang 格子上でのベクトル場を $(v_r^e, v_\theta^e, v_\phi^e)^t$ とする。ここで t は転置を意味する。Yin-Yang 格子間の回転角を $\psi(\theta, \phi)$ とすると、Yin 座標から Yang 座標へのベクトル変換は、

$$\begin{pmatrix} v_r^e \\ v_\theta^e \\ v_\phi^e \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \psi & -\sin \psi \\ 0 & \sin \psi & \cos \psi \end{pmatrix} \begin{pmatrix} v_r^n \\ v_\theta^n \\ v_\phi^n \end{pmatrix} \quad (41)$$

と表すことができる。 r 方向のベクトルは不変で、 θ, ϕ 方向のベクトルの回転変換を行っている。Fig.5 から、 $\cos \psi$ と $\sin \psi$ は

$$\cos \psi = \hat{\phi}^n \cdot \hat{\phi}^e, \quad (42)$$

$$\sin \psi = -\hat{\phi}^n \cdot \hat{\theta}^e, \quad (43)$$

と表される。また、 $\hat{\phi}^n, \hat{\phi}^e, \hat{\theta}^e$ は、

$$\hat{\phi}^n = -\sin \phi^n \hat{x}^n + \cos \phi^n \hat{y}^n, \quad (44)$$

$$= -\sin \phi^n \hat{x}^e + \cos \phi^n \hat{z}^e, \quad (45)$$

$$\hat{\phi}^e = -\sin \phi^e \hat{x}^e + \cos \phi^e \hat{y}^e, \quad (46)$$

$$\hat{\theta}^e = \cos \theta^e \cos \phi^e \hat{x}^e + \cos \theta^e \sin \phi^e \hat{y}^e - \sin \theta^e \hat{z}^e, \quad (47)$$

と表すことができる。ここで、Yin 格子、Yang 格子でのカーテシアン座標系の各単位ベクトルをそれぞれ $\hat{x}^n, \hat{y}^n, \hat{z}^n, \hat{x}^e, \hat{y}^e, \hat{z}^e$ とした。よって、 $\cos \psi$ と $\sin \psi$ は、式(50)-(55) より、

$$\cos \psi = -\sin \phi^e \sin \phi^n, \quad (48)$$

$$\sin \psi = \frac{\cos \phi^n}{\sin \theta^e} \quad (49)$$

とかける。式(22),(23)を用いると、Yin-Yang 格子におけるベクトル変換式は、変換行列を P とし、

$$\begin{pmatrix} v_r^e \\ v_\theta^e \\ v_\phi^e \end{pmatrix} = P \begin{pmatrix} v_r^n \\ v_\theta^n \\ v_\phi^n \end{pmatrix}, \quad (50)$$

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\sin \phi^e \sin \phi^n & -\cos \phi^n / \sin \theta^e \\ 0 & \cos \phi^n / \sin \theta^e & -\sin \phi^e \sin \phi^n \end{pmatrix}, \quad (51)$$

と書くことができる。これは、Yin 座標系から Yang 座標系への変換行列である。また、Yang 座標系から Yin 座標系への変換行列は P の逆行列 P^{-1} で表すことができ、 P^{-1} は P の添字を入れ替えることで表現できる。

$$P^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\sin \phi^n \sin \phi^e & -\cos \phi^e / \sin \theta^n \\ 0 & \cos \phi^e / \sin \theta^n & -\sin \phi^n \sin \phi^e \end{pmatrix}, \quad (52)$$

Zhong 格子上のベクトル場を v_x, v_y, v_z とすると、Zhong 座標系から Yin 座標系へのベクトル変換は

$$\begin{pmatrix} v_r^n \\ v_\theta^n \\ v_\phi^n \end{pmatrix} = \begin{pmatrix} \sin \theta \cos \phi & \sin \theta \sin \phi & \cos \theta \\ \cos \theta \cos \phi & \cos \theta \sin \phi & -\sin \theta \\ -\sin \phi & \cos \phi & 0 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}, \quad (53)$$

また Zhong 座標系から Yang 座標系へのベクトル変換は

$$\begin{pmatrix} v_r^e \\ v_\theta^e \\ v_\phi^e \end{pmatrix} = \begin{pmatrix} -\sin \theta \cos \phi & -\sin \theta \sin \phi & -\cos \theta \\ -\sin \phi & \cos \phi & 0 \\ \cos \theta \cos \phi & \cos \theta \sin \phi & -\sin \theta \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}, \quad (54)$$

と表すことができる。Shell と Core の間の補間にはトリリニア補間を用いる。つまり、Shell の境界値を Core から補間する場合は、半径 $r = r_c$ での Shell の座標値を Core の座標系に変換、補間点を囲む 8 個の格子点を探索し 8 点から線形補間を行う。逆に、Core の値を Shell から補間する場合には、Shell の格子での、

$$r_c \leq r \leq r_c + 2\Delta r \quad (55)$$

の範囲内で Core 領域に含まれる格子を探索する。そして、その座標値を Core から Shell に変換し、線形補間に利用する 8 つの格子点を探索し補間する。

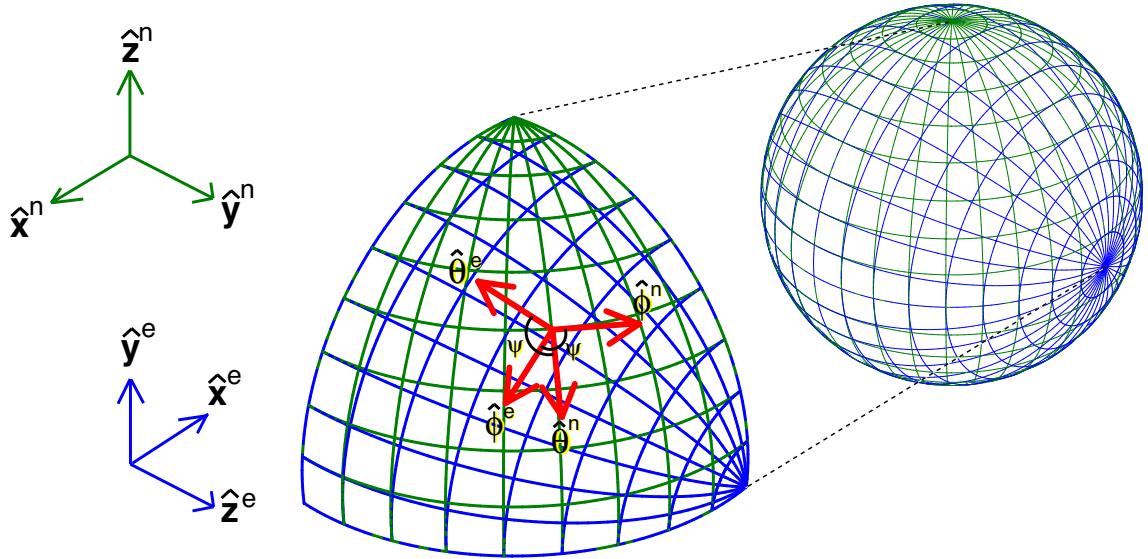


Fig.5 Vector components of Yin and Yang.

3 Fortran プリプロセッサの開発

本研究のシミュレーションプログラムの実装に関して、より便利で簡潔なコーディングを目指すべくプログラミング言語の1つであるFortranのプリプロセッサを開発した。その詳細をこれより説明する。

3.1 現代的 Fortran の問題点

Fortranとは1950年代に誕生した世界初の高級プログラミング言語の一つであり、その特徴はHPC(High Performance Computing:高性能計算)に特化していることである。FortranがHPCとして優れている点は以下の通りである。

- 多次元配列の扱いが得意であること。
- 組み込み関数が豊富であること。
- 数式の取り扱いが得意であること。
- モジュールと内部副プログラムによる階層的名前空間が容易であること。
- 数学ライブラリを含めて科学技術関係のコードが豊富であること。

Fortranは現在でもバージョンアップし続け、特にFortran 2003では大きな仕

様変更がされた。よって Fortran 2003 以降のバージョンを現代的 Fortran(Modern Fortran) と呼ばれることが多い。この現代的 Fortran には様々な便利な機能を備えているものの、シミュレーションにおいては使いづらいところがある。例えば、C や Java に標準的に用意されている複数行をまとめてコメントアウトする機能（ブロックコメントアウト）は現代的 Fortran に備わっていないので、一行一行の行頭に「！」(exclamation mark) をつける必要がある (Fig.6)。これはトライ & エラーを繰り返す数値シミュレーションを行うものにとって大きな手間となり負担となる。このようなストレスがかかるコーディングを避けるために、新しい Fortran の方言 eFortran とそれを標準 Fortran に文字列変換するプリプロセッサ efpp を開発した。

```
!
!      This routine calculates ...
!
!      History: ...
!
!      Author: ...
!
```

Fig.6 There is no block comments in standard Fortran.

3.2 eFortran と efpp

プリプロセッサとはプログラムをコンパイルする前にそのプログラムになんらかの処理をするプログラムのことである。本研究で開発したプリプロセッサ efpp は通常コンパイルできない特殊な文法のプログラムを文字列置換によってコンパイル可能なプログラムに書き換える。我々はこの特殊な文法のプログラミング言語を eFortran と名付けた。つまり、eFortran で書かれたプログラムを efpp によってコンパイル可能なプログラムに書き換えるという仕組みである (Fig.7)。efpp は Python というプログラミング言語で書かれている。その理由はコンパイルが不要で実行が簡単であるから、文字列変換に用いる正規表現が得意であるからなどである。

eFortran(.e03)

Fortran(.f90)

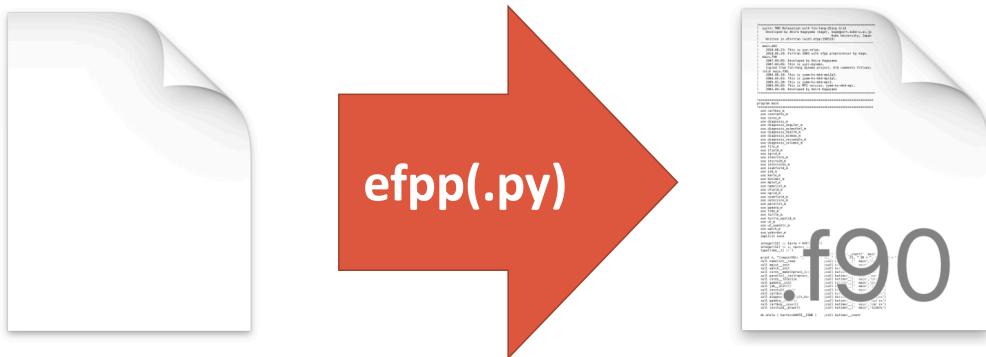


Fig.7 Conversion from eFortran to Fortran by preprocessor.

efpp の利点は数多くある。

- シミュレーションでよく使う構文を簡潔に書くことができる。(繰り返し、初期化、変数定義など)
- efpp は実質文字列を変換しているだけなので、ユーザーが新しい文法を追加したり、不要ない文法を無視したりすることができる。
- 文字列変換によって行番号を変えないようにしている、これはデバッグの際に混乱することを避けるためである。

3.3 具体例

ここでは 3 つの具体例を通して、eFortran の詳細を説明する。

- (1) ブロックコメント
- (2) 変数定義の簡略化
- (3) マクロ定義

3.3.1 ブロックコメント

先ほど述べた通り、標準の Fortran にはブロックコメント機能がないので複数行のコメントアウトには不便が生じていた。そこで eFortran では簡単に複数行のコメント

アウトができるような新しい文法を追加した。具体的にはコメントアウトしたい行を3つ以上の「=」(equal)の行で挟むことで実現する (Fig.8)。この機能は入れ子にすることも可能で、eFortran コード内で同じ数の「=」の行のペアがあればそのペアに挟まれている行が自動的にコメントアウト文に変換される (Fig.9)。

```
=====
This routine calculates ...

History: ...
Author: ...

=====
```

Fig.8 Syntax of block comment. Sandwiched by a pair of "=" sequence in eFortran.

<pre>abc def ghijklmn opq ===== abc def ghijklmn opq ===== abc def ghijklmn opq abc def ghijklmn opq ===== abc def ghijklmn opq ===== abc def ghijklmn opq ===== abc def ghijklmn opq</pre>		<pre>abc def ghijklmn opq ! ===== ! abc def ghijklmn opq !! ===== !! abc def ghijklmn opq !! abc def ghijklmn opq !! ===== ! abc def ghijklmn opq ! ===== abc def ghijklmn opq</pre>
---	--	--

Fig.9 A block of comment can contain another block of comment inside it.

3.3.2 変数定義の簡略化

Fortran の変数定義は C や Java と違って、非常に冗長である。例えば、整数型の変数 i を定義するときは C では

```
int i;
```

と書くのに対し Fortran では

```
integer :: i
```

と書く必要がある。また Fortran では変数に様々なオプションをつける場合があり、例えば読み取り専用のオプショナル（省略可能な）引数である倍精度実数 p は以下のように記述する。

```
real(DR), intent(in), optional :: p
```

シミュレーションにおいて、このような変数定義は頻繁に行うためコーディングが面倒なだけでなくタイプミスによるエラーが起こりやすい。eFortran では簡潔な変数定義を実現している。(Fig.10)



```
integer(SI), intent(inout) :: m  
real(DR), intent(in), optional :: p  
integer(SI), parameter :: N = 100
```

```
inte(SI) <io> :: m  
real(DR) <optin> :: p  
inte(SI) <const> :: N = 100
```

Fig.10 In standard Fortran, declarations to be long lines (top). But in eFortran, it can be write simply (bottom).

3.3.3 マクロ定義

eFortran には特定の文字列をモジュール名や関数名などに自動的に変換する機能がある。例えば、eFortran コードに「_MODULE_」と入力すれば、その文字列は「_MODULE_」と書かれたモジュールの名前に変換される。(Fig.11) この機能は主にデバグの際に用いられる。さらに、このような文字列変換のルールは eFortran の

ユーザーが新しく作ることができる。その方法は efpp_alias.list というファイルに以下の文を追加すれば良い。

```
"string_before" => "string_after"
```

これにより、eFortran コードで「string_before」と書かれた箇所は標準 Fortran コードで「string_after」に変換される。

<u>__MODULE__</u>	⇒ Module name
<u>__FUNC__</u>	⇒ Subroutine / Function name
<u>__LINE__</u>	⇒ Line number of the source code

Fig.11 String substitution in source code

3.4 サンプルコード

最後に eFortran のサンプルコードと変換後の標準 Fortran コードを以下に示す (Fig.12,Fig.13)

```
1 module test_m
2   use constm
3   implicit none
4   contains
5     subroutine testr(j,val)
6       integer(SI) :: j
7       real(DR) :: val
8
9       =====
10      This is in a block comment.
11      =====
12      Comments can be nested.
13      =====
14      =====
15
16      integer(SI) :: ctr = 0
17      integer(SI) :: NN = 1
18      logical :: just_once = .true.
19
20      j += 1
21      call mhd.sub.update(mhd.main)
22      print *, "Hello from __FUNC__ in __MODULE__"
23      ! ==> Hello from testr in test_m
24
25      ==<just_once>==
26      print *, "This line is executed only once."
27      ==>/just_once>==
28
29      ==<skip ctr:10>==
30      print *, "This line is executed every 10 times."
31      ==>/skip ctr>==
32
33      contains
34        subroutine inter
35          print *, "Hello from __FUNC__"
36          ! ==> Hello from testr/inter
37          **debug_print** "i = ", i
38        end subroutine inter
39      end subroutine testr
40 end module test_m
```

Fig.12 Sample eFortran code.

```

1 module test_m
2   use constm
3   implicit none
4   contains
5     subroutine testr(j,val)
6       integer(SI), intent(inout) :: j
7       real(DR), intent(in), optional :: val
8
9 !      =====
10 !      This is in a block comment.
11 !!      =====
12 !!      Comments can be nested.
13 !!      =====
14 !
15
16   integer(SI) :: ctr = 0
17   integer(SI), parameter :: NN = 1
18   logical :: just_once = .true.
19
20   j = j + 1
21   call mhd%sub%update(mhd%main)
22   print *, "Hello from testr in test_m"
23                                     ! ==> Hello from testr in test_m
24
25   if (just_once) then
26     print *, "This line is executed only once."
27   just_once = .false. ; end if
28
29   if(mod(ctr,10)==0) then
30     print *, "This line is executed every 10 times."
31   end if; ctr = ctr + 1
32
33   contains
34     subroutine inter
35       print *, "Hello from testr/inter"
36                                     ! ==> Hello from testr/inter
37       **debug_print** "i = ", i
38     end subroutine inter
39   end subroutine testr
40 end module test_m

```

Fig.13 Standard Fortran code converted from eFortran

4 計算モデルとコードの実装

4.1 計算手法

Yin–Yang–Zhong 格子を用いて単位球体内部 (半径 $r = 1$) での MHD 方程式 (式 (1)-(4)) の数値計算を行う。基本変数を質量流束 \mathbf{f} 、ベクトルポテンシャル \mathbf{a} 、質量密度 ρ 、圧力 p の4つとし、計算領域は全球 ($0 \leq r \leq 1$) とする。離散化の手法として、空間微分は有限差分法の1つである2次精度中心差分法、時間積分は4次ルンゲ

=クッタ法を用いた。

4.2 初期条件

4.2.1 圧力・質量密度

本研究において圧縮性 MHD 方程式を解くものの、シミュレーションの精度上、圧縮性の物理的な効果は可能な限り小さくするものとする。よって、代表的な流速を U 、音速を C とすると

$$\mathcal{M} = \frac{U}{C} \quad (56)$$

で定義されるマッハ数 \mathcal{M} が最大 $\mathcal{O}(10^{-1})$ の流れを考える。このとき流速を $\mathcal{O}(10^0)$ とすると、音速

$$C = \sqrt{\gamma \frac{p}{\rho}} \quad (57)$$

は $\mathcal{O}(10)$ 程度となるので、本研究の全て初期状態において圧力 p と質量密度 ρ に

$$p = 100.0, \quad (58)$$

$$\rho = 1.0, \quad (59)$$

を与えた。

4.2.2 球面調和関数

本研究のシミュレーションの初期条件として球面調和関数を用いた。ここではまず、球面調和関数の概要を説明する。

球面調和関数とは球面状の関数 $f(\theta, \phi)$ を展開するのに用いられる関数群のことをいう。つまり、 $f(\theta, \phi)$ は次式のように球面調和関数 Y_l^m の線形和で表すことができる。

$$f(\theta, \phi) = \sum_{l=0}^L \sum_{m=-l}^l c_l^m Y_l^m(\theta, \phi) \quad (60)$$

ここで、 l, m はそれぞれ $0 \leq |m| \leq l$ を満たす整数、 c は定数である。この式より球面調和関数はフーリエ級数展開を球面に対応させたものといえる。また球面調和関数自体は次式で表される。

$$Y_l^m(\theta, \phi) = P_l^m(\cos\theta) e^{im\phi} \quad (61)$$

ここで、 $P_l^m(\cos\theta)$ はルジャンドル陪関数であり緯度方向の展開を表している ($e^{im\phi}$ は経度方向の展開を表す)。以下の表 (Table.1) は異なる (l, m) におけるルジャンドル陪関数と球面調和関数を示している。

Table1 Associated Legendre polynomials (left), and spherical harmonics (right).

$P_0^0(x) = 1$	$Y_0^0 = \sqrt{\frac{1}{4\pi}}$
$P_1^1(x) = -(1 - x^2)^{1/2}$	$Y_1^1 = -\sqrt{\frac{3}{8\pi}} \sin^2\theta e^{i\phi}$
$P_1^0(x) = x$	$Y_1^0 = \sqrt{\frac{3}{4\pi}} \cos\theta$
$P_2^2(x) = 3(1 - x^2)$	$Y_2^2 = \frac{1}{4} \sqrt{\frac{15}{2\pi}} \sin^2\theta e^{2i\phi}$
$P_2^1(x) = -3(1 - x^2)^{1/2}x$	$Y_2^1 = -\sqrt{\frac{15}{8\pi}} \sin\theta \cos\theta e^{i\phi}$
$P_2^0(x) = \frac{1}{2}(3x^2 - 1)$	$Y_2^0 = \sqrt{\frac{5}{4\pi}} (\frac{3}{2} \cos^2\theta - \frac{1}{2})$

シミュレーションではこの球面調和関数の $(l, m) = (3, 1), (3, 2), (3, 3)$ の 3 パターンを初期条件として行った。具体的に初期の磁場のベクトルポテンシャル \mathbf{a} と質量流速 \mathbf{f} はそれぞれ以下のように設定する。 C は定数であり全ての (l, m) において $C = 0.2$ とした。

$$\begin{pmatrix} a_r \\ a_\theta \\ a_\phi \end{pmatrix} = \begin{pmatrix} C \times r \times \sin^2(\pi r) P_l^m(\cos\theta) \cos(m\phi) \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{f} = \mathbf{0} \quad (62)$$

以下にそれぞれの (l, m) のときの初期磁力線を示す (Fig.14)。それぞれ左から $(l, m) = (3, 1)$ 、 $(l, m) = (3, 2)$ 、 $(l, m) = (3, 3)$ の磁場を表している。

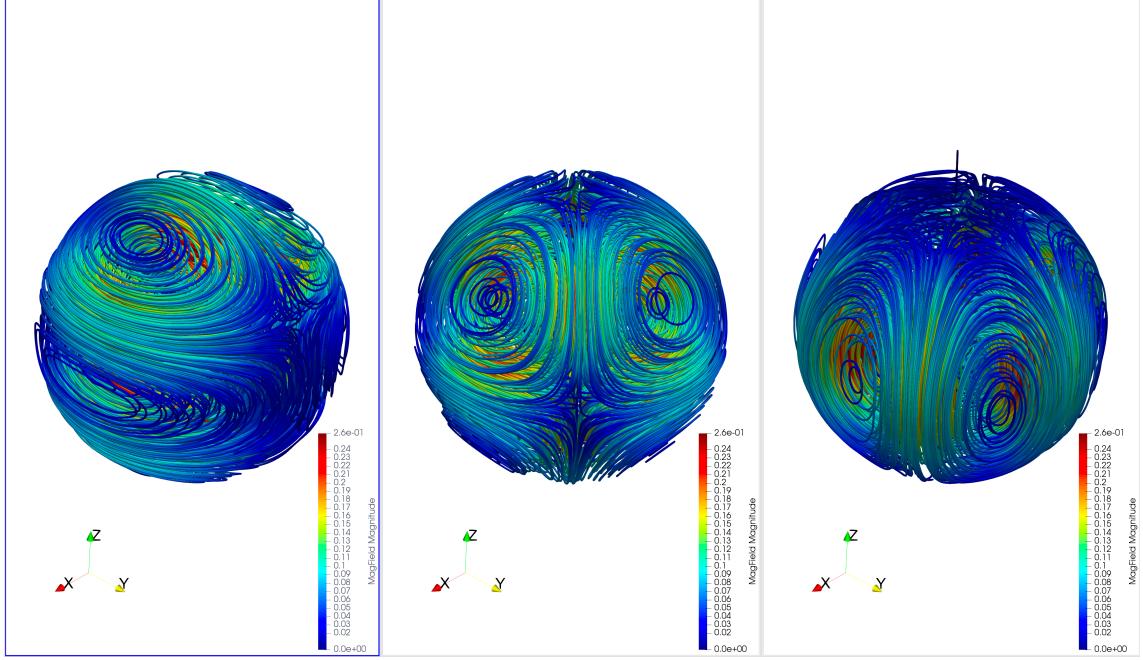


Fig.14 Initial magnetic field in $(l, m) = (3, 1)$ (left), $(3, 2)$ (middle), $(3, 3)$ (right).

4.3 境界条件

本研究では、磁場に関しては完全導体壁境界条件（磁力線が境界を貫かない）、速度場に関しては rigid 境界条件（流れがない）、温度に関しては断熱境界条件（熱の出入りがない）を用いた。つまり境界面を S 、境界面に垂直な単位ベクトルを \mathbf{n} とすると

$$\mathbf{v} = \mathbf{0} \text{ (on } S\text{)}, \quad (63)$$

$$\mathbf{a} \times \mathbf{n} = \mathbf{0}, \nabla \cdot \mathbf{a} = \mathbf{0} \text{ (on } S\text{)}, \quad (64)$$

$$\frac{\partial T}{\partial r} = 0 \Leftrightarrow \frac{\partial}{\partial r} \left(\frac{p}{\rho} \right) = 0 \text{ (since } p = \rho T\text{) (on } S\text{)}, \quad (65)$$

を意味する。

4.4 パラメータ設定

粘性係数 μ 、電気抵抗 η 、熱拡散率 κ はそれぞれ空間的に一様であり規格化された値とする。全ての初期条件で $\mu = 1.0 \times 10^{-4}$, $\eta = 1.0 \times 10^{-4}$, $\kappa = 1.0 \times 10^{-4}$ と設定した。計算の空間解像度は、すべての場合において Yin-Yang 格子は動径方向に 201、

緯度方向に 204、経度方向に 608、Zhong 格子は x,y,z 方向にそれぞれ 222 である。計算機は地球シミュレータ (NEC SX-ACE, JAMSTEC) を使用した。また 3 次元可視化ツールとして Paraview を用いた。

4.5 eFortran 版球 MHD コードの概観

本研究のシミュレーションコードの実装にあたって、eFortran を使用した。ここではそのプログラムコードの概観について説明する。

main.e03

メインモジュール

constants.e03

定数モジュール。MHD 方程式を解くための定数を定義している。

cores.e03

Yin–Yang–Zhong 格子モジュール。Yin–Yang–Zhong それぞれの格子内で MHD 方程式を解く。

diagnosis.e03

診断モジュール。数値的なエラーが見つかればプログラムを終了させる。

ut.e03

ユーティリティモジュール。標準出力や型変換などを扱う。

mpiut.e03

MPI ユーティリティモジュール。本シミュレーションは MPI を用いて並列計算をしている。

namelist.e03

ユーザーがシミュレーションごとに定義する定数が書かれたファイルを読み込むモジュール。

ppdata.e03

データ出力モジュール。MHD シミュレーションで得られた生データを出力する。

turtle.e03

2 次元可視化モジュール。MHD データを等高線やベクトルを使って可視化表現する。

5 シミュレーション結果

5.1 エネルギーのグラフ

以下に球面調和関数 Y_l^m の

- (a) $(l, m) = (3, 1)$ の場合
- (b) $(l, m) = (3, 2)$ の場合
- (c) $(l, m) = (3, 3)$ の場合

それぞれの運動エネルギーおよび磁気エネルギーのグラフを示す (Fig.15-17)。なおグラフ化ツールには gnuplot を用いた。

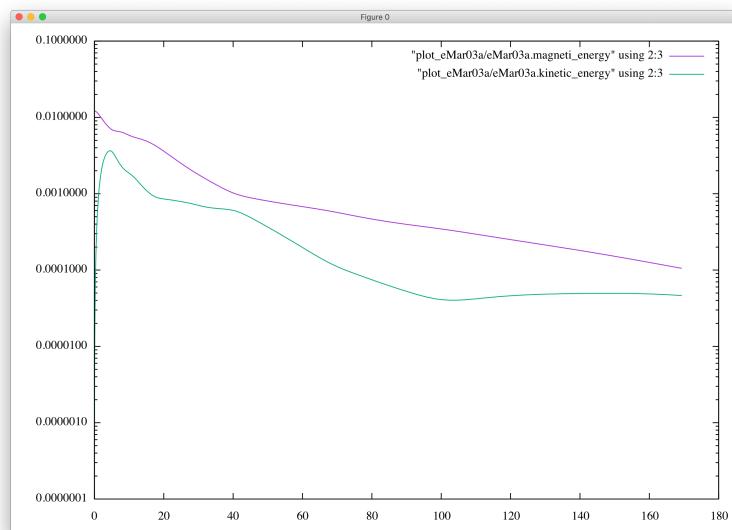


Fig.15 Relaxation of total magnetic(purple line) and flow(green line) energy in $(l, m) = (3, 1)$.

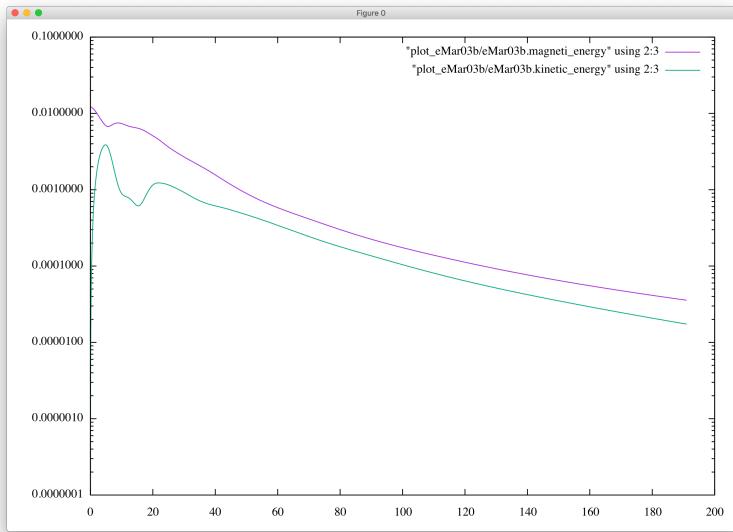


Fig.16 Relaxation of total magnetic(purple line) and flow(green line) energy in $(l, m) = (3, 2)$.

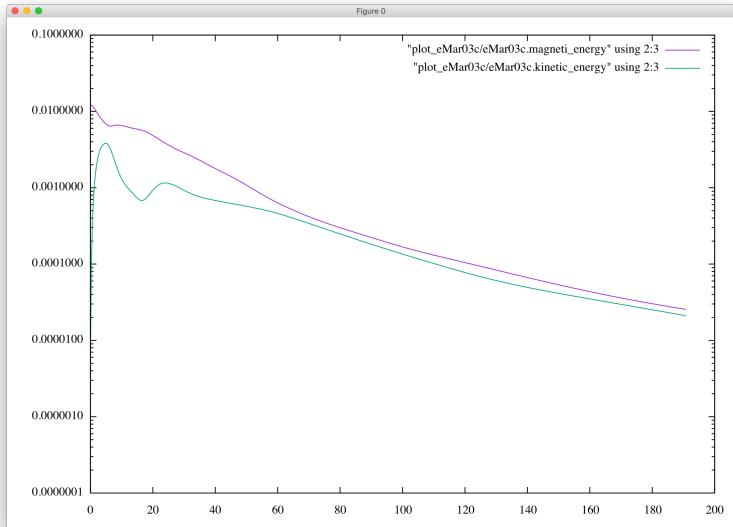


Fig.17 Relaxation of total magnetic(perple line) and flow(green line) energy in $(l, m) = (3, 3)$.

いずれの場合も磁気エネルギーと運動エネルギーがお互いにエネルギーを交換しあい振動していることがわかる。そして最終的に振動は落ち着き、双方が線形的に減衰

している。一番興味があるのはこのエネルギーが線形的に減衰している時で、それぞれグラフがプロットしている時刻の最大値付近では系が十分緩和していると考えられる。

5.2 緩和状態の三次元構造

以下に緩和状態の磁力線および流線の3次元構造を示す (Fig.18-20)。流線は流れのベクトル図に重ね合わせている。

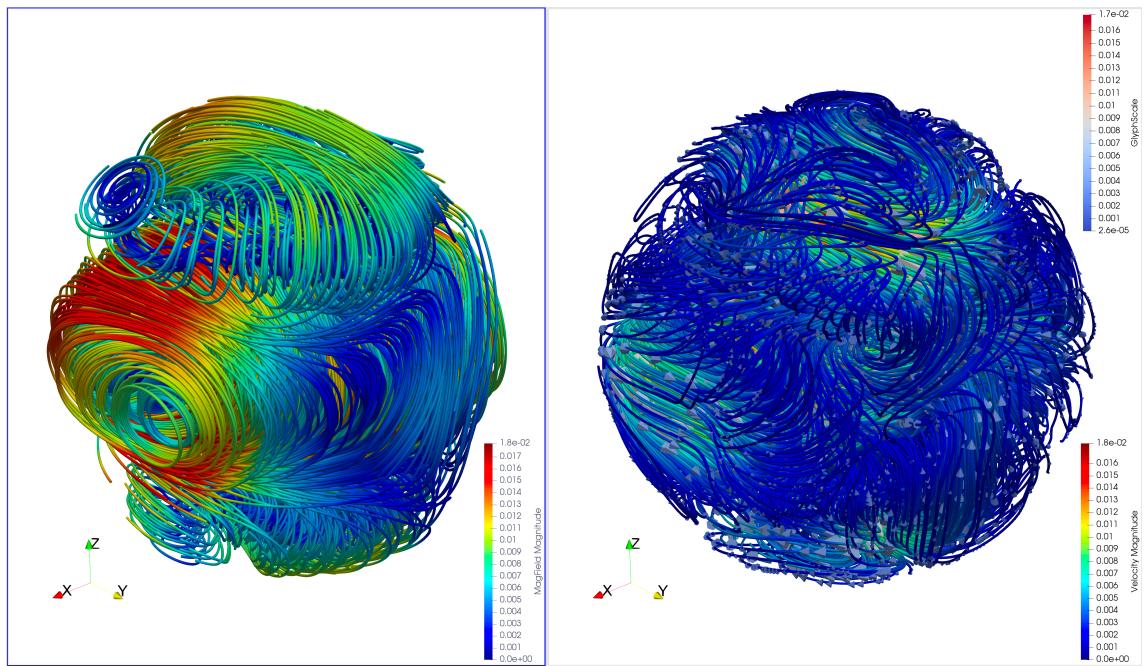


Fig.18 Stream tracer of relaxed magnetic field(left), and flow(right) in $(l, m) = (3, 1)$.

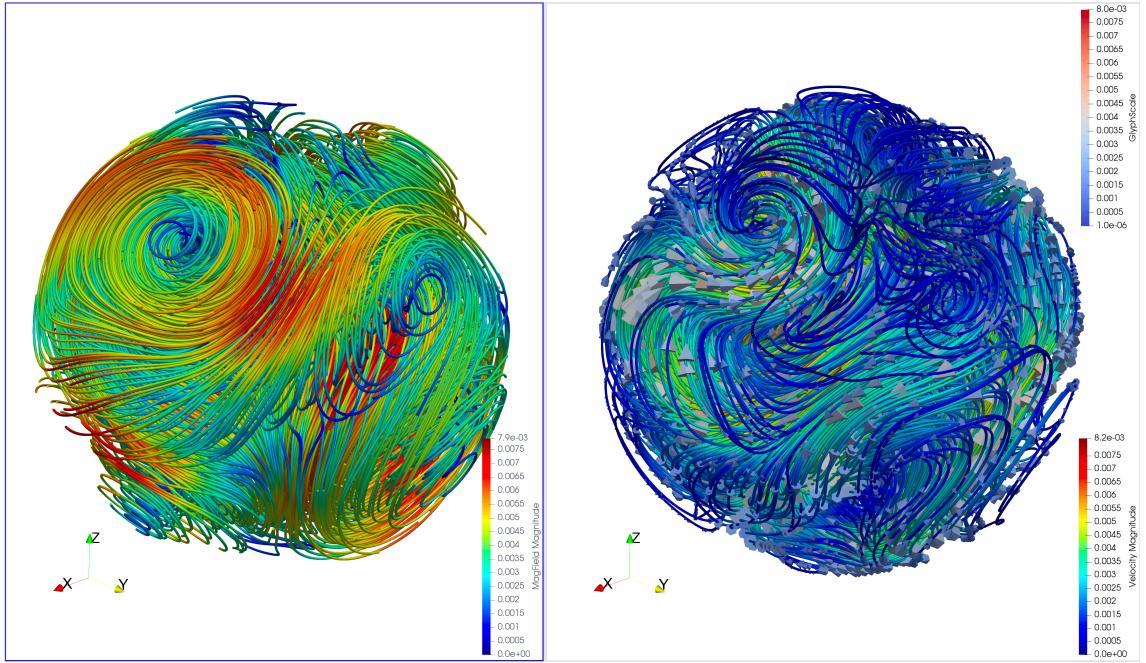


Fig.19 Stream tracer of relaxed magnetic field(left), and flow(right) in $(l, m) = (3, 2)$.

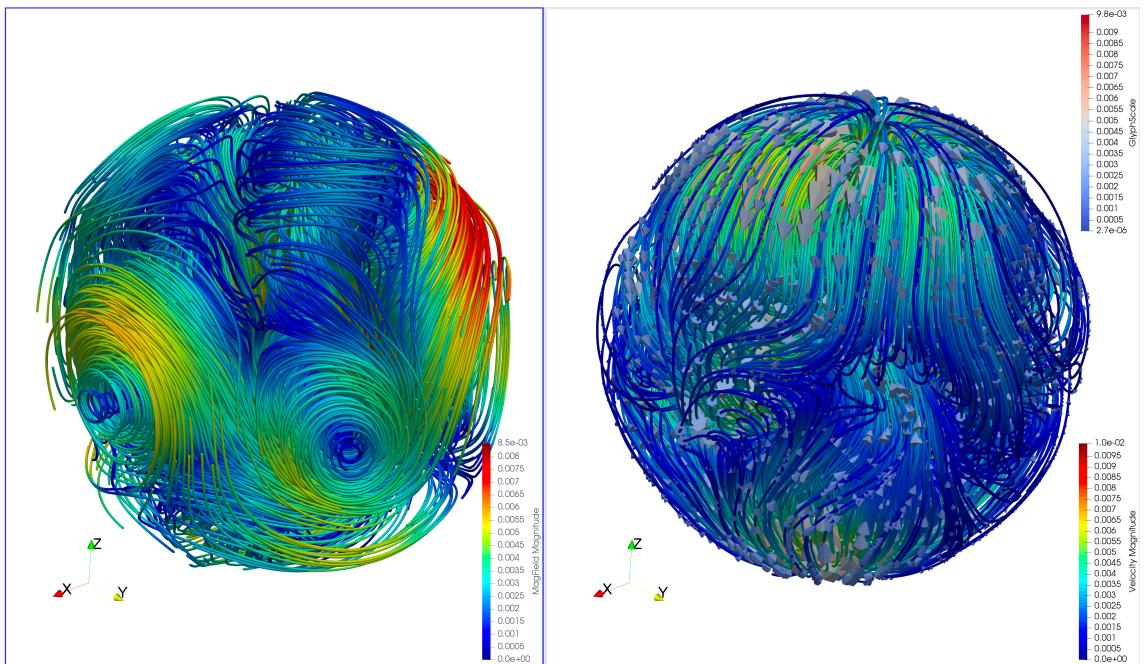


Fig.20 Stream tracer of relaxed magnetic field(left), and flow(right) in $(l, m) = (3, 3)$.

これらのうち流れ場の構造の特徴が特に顕著だと思われる $(l, m) = (3, 3)$ を重点的

に解析する。

5.2.1 $(l, m) = (3, 3)$ のときの緩和状態

以下に Fig.20 の赤道断面 (Fig.21 上) と子午断面 (Fig.21 下) を示す。

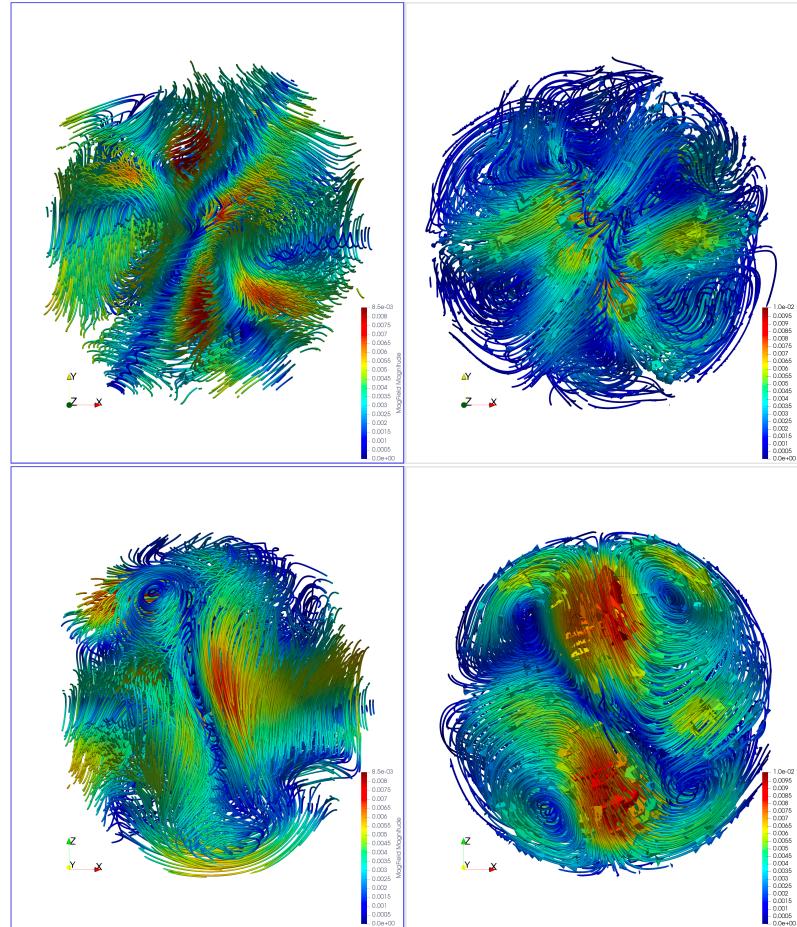


Fig.21 Equator plane (top), and meridian plane (bottom) of stream tracer:
relaxed magnetic field (left) and flow (right) in $(l, m) = (3, 3)$.

これより緩和状態の構造は以下のようない模式図 (Fig.22) で表される。赤色は磁力線、青色は流れ場を表している。

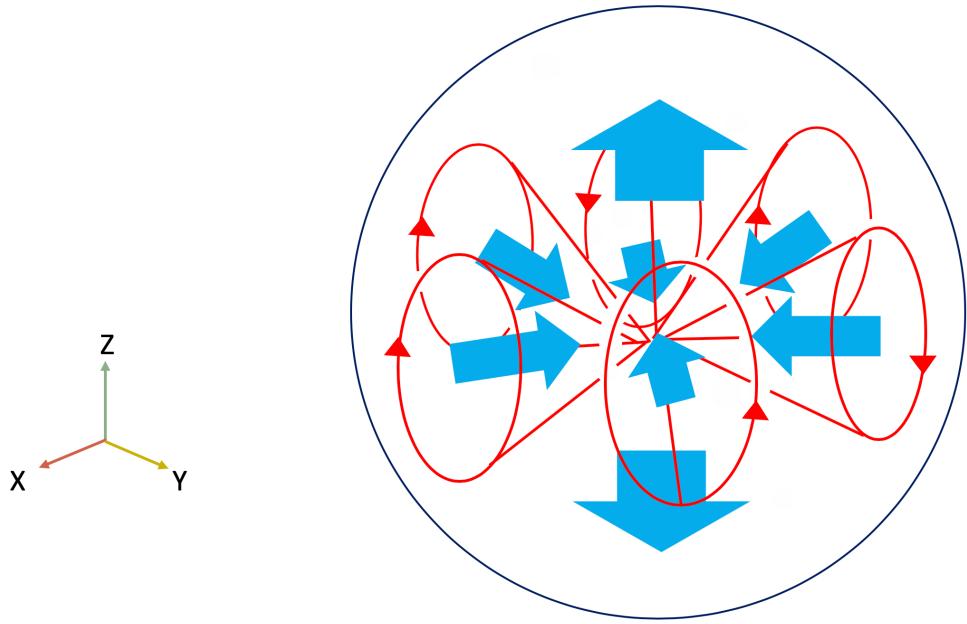


Fig.22 3D Structure of relaxed magnetic field(red), and flow(blue) in $(l, m) = (3, 3)$.

この図より速度場の遷移が以下のようになることがわかる。

1. 磁力線のリングから中心に向かう流れが起きる。
2. 球の中心付近で流れが衝突する。
3. 衝突した流れは両極方向へ向きが変わる。
4. 極にたどり着いた流れは境界に当たることで、また磁力線のリングの中心へ向かう（1に戻る）。

また磁場の遷移は以下のようになる。

1. 球表面付近の磁力線のリングが縮まり流れの駆動力になる。
2. 縮んだ磁力線のリングは流れに運ばれ中心付近に集まる。
3. 集まった磁力線のリングは流れ場によって両極に引き伸ばされる。
4. 引き伸ばされた磁力線のリングは球の表面付近を通り、元の磁力線のリングの中心へ向かう（1に戻る）。

6 結論

エネルギーが不安定な状態から安定な状態に落ち着くという緩和は自然界の多くに見られ、MHD 流体も例外ではない。MHD 流体の緩和状態は領域内で力が全く働くかない force-free 状態であるという Taylor 理論は多くの実験に適用されてきた。しかし、Taylor 理論は運動エネルギーが無視され、緩和状態では流れが存在しないと仮定されている。そこで本研究では球ジオメトリ内部での MHD 流体の緩和について計算機（スパコン）上で調べた。

本研究の設定として、完全導体かつ外部から力が働くない断熱の境界である球に MHD 流体が入っている状況を想定した。また初期条件として球面調和関数を磁場を与えた。球面調和関数で表されるリング上の磁場は、磁場の輪が縮む方向にローレンツ力が働きそれが起動力となって MHD 流体が動き出した。

計算機でシミュレートするにあたって球座標格子を用いた場合、格子点の集中が避けられないため計算にコストがかかってしまう。そこで本研究では重合格子である Yin-Yang-Zhong 格子を用いることでコストがかからない計算を行うことができた。

本研究では弱い磁場を想定していたので、これからは強い磁場についても考察していく必要がある。

謝辞

本研究を進めるにあたって、熱心に指導していただいた陰山聰教授に深く感謝いたします。

参考文献

- [1] 堀内利得. “プラズマ・核融合シミュレーションの発展と将来への期待”. *J. Plasma Fusion Res.*, Vol. 80, No. 5, pp. 401–403, 2004.
- [2] Lodewijk Woltjer. A theorem on force-free magnetic fields. *Proceedings of the National Academy of Sciences*, Vol. 44, No. 6, pp. 489–491, 1958.
- [3] J. Brian Taylor. Relaxation of toroidal plasma and generation of reverse magnetic fields. *Physical Review Letters*, Vol. 33, No. 19, p. 1139, 1974.
- [4] 水谷幸夫, 香月正司. “コンピュータによる熱移動と流れの数値解析”. 森北出版, 2 1985.
- [5] 坂下志郎, 池内了. “宇宙流体力学”. 培風館, 1996.
- [6] Sergio Ortolani and Dalton D Schnack. “*Magnetohydrodynamics of Plasma Relaxation*”. World Scientific.
- [7] Akira Kageyama and Tetsuya Sato. ”Yin-Yang grid”: An overset grid in spherical geometry. *Geochemistry, Geophysics, Geosystems*, Vol. 5, No. 9, 2004.