J. Plasma Fusion Res. Vol.89, No.3 (2013) 171-179

講座 核融合プラズマシミュレーションの技法ー大規模並列計算環境の活用ー

4．Vlasovシミュレーションのコーディング技法

渡邉智彦，井戸村泰宏[1)]
核融合科学研究所・総合研究大学院大学，[1)]日本原子力研究開発機構

（原稿受付：2013年2月8日）

本章では，磁場閉じ込め核融合プラズマの乱流輸送解析で用いられるジャイロ運動論的シミュレーションにおけるコーディング技法について解説する．前半は，局所フラックス・チューブ・モデルを用いたGKVコードにおける並列化方針と集団通信について述べる．後半は，グローバル配位において全分布関数の時間発展を求めるGT5Dコードを例に，核融合プラズマの大規模シミュレーション研究で用いられている最先端の技法を概説する．

Keywords:
gyrokinetic simulation, turbulence, parallel computing

4.1 ジャイロ運動論的シミュレーションについて

数keVにも及ぶ高温の磁場閉じ込め核融合プラズマでは，粒子の平均自由行程は装置サイズの数百倍にも及び，通常の流体近似の限界を超えて，運動論的な取り扱いが不可欠である．特に，異常輸送の原因となる微視的乱流については，ジャイロ運動論[1]に基づいたシミュレーションが輸送フラックスの定量的な評価に必須の手段となっている．このため，大規模なジャイロ運動論的シミュレーション研究が精力的に進められ，輸送機構の解明と理解，さらに閉じ込め性能の予測に関する研究が世界各国で展開されている[2]．磁場閉じ込め核融合プラズマにおける微視的乱流シミュレーションについては，学会誌の講座としてもまとめられているので，ご参照いただきたい[3]．

さて，ジャイロ運動論的シミュレーションでは，実空間3次元と速度空間2次元の計5次元からなる位相空間上の分布関数の時間発展方程式を数値的に解くことになる．その解法には，粒子（マーカー）を用いたPIC（Particle-in-Cell）法に基づくものと，分布関数の発展方程式を数値格子や関数展開などを使って離散化して近似的に解く，いわゆるVlasovシミュレーション法とがある．本章では，後者の手法，すなわちジャイロ運動論的Vlasovシミュレーションのコーディング技法について解説する．

分布関数を連続体として取り扱うVlasovシミュレーションは，数値手法の観点から3章で述べられた流体・MHDシミュレーションと多くの共通点を持つ．実際，差分に伴う袖通信やFFTなど共通する課題が多い．一方，5次元という高次元性や，速度空間積分を含む微積分方程式であることなどから，コーディング技法としてユニークなトピックも含んでいる．これらを踏まえ，次節では局所的な乱流輸送を扱うGKVコード[4]を用いたフラックス・チューブ・シミュレーションで使われる手法について述べ，4.3節ではグローバルな乱流輸送問題を全分布関数によって扱うGT5Dコード[5]での技法について概説する．

第1章で簡単に紹介されたように，5次元位相空間上の分布関数を取り扱うジャイロ運動論的シミュレーションは多大な計算コストを必要とする．このため，京コンピュータや国際核融合エネルギー研究センターのHelios，核融合科学研究所（核融合研）のプラズマ・シミュレータなどにおいて大規模なシミュレーションが進められている．ここでは，そうした大規模シミュレーションで使われている手法の一端を紹介したい．読者諸賢の今後の研究の一助となれば幸いである．

4.2 局所ジャイロ運動論的Vlaosvシミュレーションの技法

4.2.1 フラックス・チューブ・モデルとその数値解法

トカマクやヘリカル型の磁場閉じ込め装置において発生する微視的乱流は，熱や粒子の異常輸送を引き起こす．ジャイロ半径（ρ）程度の大きさをもつ乱流渦は，装置サイズ（L）に比べ十分小さいと考え，局所的な乱流輸送シミュレーションが行われている．そのスケール比は，乱流揺らぎの磁力線平行方向（z）と垂直方向の特性長の比と同じオーダーとなり，実空間において強い非等方性をもつ揺動成分を取り扱う必要がある．そのために，磁力線に沿ってシミュレーション領域を設定した，フラックス・チューブ・モデル[6]が有用である．文献[4]に挙げたGKVコードでは，実空間座標を(x,y,z)とし，さらに速度空間について$(v_\parallel,\mu)$の座標を選ぶ．ここで，xは動径方向，yは磁力線ラベル方向，$v_\parallel$は磁力線平行方向速度，μは磁気モーメントを表す．

ここでは5次元位相空間$(x,y,z,v_\parallel,\mu)$上で定義されたジャイロ中心の揺動分布関数δfの時間発展を，静電近似（低β近似）のもと，ジャイロ運動論的方程式

4. Coding Techniques for Vlasov Simulations
WATANABE Tomo-Hiko and IDOMURA Yasuhiro
corresponding author's e-mail: watanabe.tomohiko@nifs.ac.jp

$$\left[\frac{\partial}{\partial t}+v_{\parallel}\hat{\boldsymbol{b}}\cdot\nabla+\boldsymbol{v}_{\mathrm{d}}\cdot\nabla-\mu(\hat{\boldsymbol{b}}\cdot\nabla\Omega)\frac{\partial}{\partial v_{\parallel}}\right]\delta f+\frac{c}{B_0}\{\phi,\delta f\}$$
$$=(\boldsymbol{v}_{*}-\boldsymbol{v}_{\mathrm{d}}-v_{\parallel}\hat{\boldsymbol{b}})\cdot\frac{e\nabla\phi}{T_{\mathrm{i}}}F_{\mathrm{M}}+C(\delta f) \quad (1)$$

にしたがって求める場合を考えよう．ここで，左辺の[…]内の項は線形微分演算からなり，{ , }は$\boldsymbol{E}\times\boldsymbol{B}$ドリフトによる非線形対流項をポアッソン括弧式の形式で表したものである．右辺の$C(\delta f)$は速度空間座標についての2階微分を含むモデル衝突項を意味する．また，マクスウェル分布F_{M}に比例する項は，δfについての式としては非斉次項にあたり，ジャイロ半径効果を含むポテンシャル揺動ϕに依存する．このポテンシャル揺動は，分布関数揺動δfを速度空間上で積分したジャイロ中心密度の揺動成分

$$\delta n_{\boldsymbol{k}}=\int J_0(\rho k)\delta f_{\boldsymbol{k}}B\,\mathrm{d}v_{\parallel}\mathrm{d}\mu \quad (2)$$

から準中性条件を使って求められる．ここで下付き添字$\boldsymbol{k}$は，(x,y)座標に関してフーリエ変換を行った揺動の波数ベクトル$\boldsymbol{k}=(k_x,k_y)$を意味する．ゼロ次ベッセル関数J_0は，揺動波数に対する有限ジャイロ半径効果を表す．また，磁気面上で一定のポテンシャルをもつ，いわゆるゾーナルフロー成分に対して電子の密度揺動がゼロになることを取り扱うために，磁気面平均を求めることが必要な場合がある．これは，ポテンシャル揺動の$k_y=0$成分に関して磁力線平行方向(z)座標について次式の積分操作を施すことで得られる．

$$\langle\phi\rangle_{kx}=\frac{\int\sqrt{g}\phi_{kx,ky=0}\mathrm{d}z}{\int\sqrt{g}\mathrm{d}z} \quad (3)$$

次にGKVコードで用いる数値手法を簡単にまとめよう．上記の基礎方程式の性質から，(x,y)座標について波数空間(k_x,k_y)へとフーリエ変換し，スペクトル法を用いるのが自然である．そのため，非線形項については，実空間へと2次元FFTを用いて逆変換した上で2つの項の積を取り，それを再度波数空間へとFFTにより変換する．一方，線形項については，(x,y)座標に関する空間微分は波数ベクトルを乗ずるだけで簡単に計算できる．磁力線平行方向の移流項とミラー項，さらに衝突項には，座標成分$(z,v_{\parallel},\mu)$に関する微分が現れるが，これらは差分近似によって評価する．時間積分には，4次精度をもつルンゲ･クッタ･ギル法を適用し，ジャイロ運動論的方程式を数値的に解く．こうして次の時間ステップで得られた$\delta f_{\boldsymbol{k}}$を数値積分してジャイロ中心密度を求め，ポテンシャル揺動を計算するという手順を取っている．以下では，大規模並列計算機における実装技法についてみていこう．

4.2.2 GKVコードでの並列化方針

オリジナルのGKVコードは，核融合研で稼働していたSX-7/5M（NEC）において開発され，その後，地球シミュレータや核融合研のプラズマ・シミュレータ（日立SR16000）をはじめ京コンピュータにいたるまで，様々な超並列計算機に移植されてきた．まず，それらにおける領域分割と並列化の指針をまとめてみよう．

低並列度における実装

SX-7は，1ノードが32台のベクトル・プロセッサで構成され，それらを一つの筐体に搭載した計算機であり，1ノードあたり約282GFlopsという当時としては強力な演算性能を有していた．このSX-7の5ノード構成をフルに利用して，トカマクにおけるイオン温度勾配（ITG）乱流のシミュレーションが行われた[4]．SX-7では，ノード内の32台のCPUでスレッド（共有）並列を行うことができ，また，コンパイラによる自動並列化機構が有効に働いていた．この環境を利用するため，GKVコードでは，分散並列化は5つのノードそれぞれに1つのMPIプロセスを割り当てるのみにとどめ，ノード内は自動並列を活用するという方針をとった．わずか5つの領域に分割するだけなので，1次元領域分割を磁力線平行方向(z)座標に適用してコーディングを行った．同様の手法は，やはり比較的少数の強力な計算ノードからなるSX-9においても有効である．

この場合に必要となるMPI通信は，磁力線平行方向の移流項を扱う差分演算にともなう袖通信と，式(3)の磁気面平均におけるz方向の総和演算に伴う集団通信である．前者については，第3章のMHDシミュレーションで解説されたものと同様に行えばよい．後者については，数値積分において現れる総和演算（リダクション演算の一種）を領域分割された変数に対して行うため，`MPI_Allreduce`を利用する（具体的なコーディング例については4.2.3小節で解説する）．

ノード内の自動並列化については，ほとんどのdoループに対してソースコードの変更なしに動作した．スレッド並列は，主に最外側ループとなるμ（磁気モーメント）方向に対して行った．ただし，2次元FFTを構成する個々の1次元FFTの手続きをまるごと自動並列で呼び出すための指示行を利用する．もちろん，同様なスレッド並列は，OpenMPを用いても可能である．

高並列度における実装

地球シミュレータやプラズマ・シミュレータ（日立SR16000）においては，SX-7／5Mに比べて演算性能は数十倍程度増大し，これを活用して大型ヘリカル装置におけるITG乱流輸送のシミュレーション[7]を実行することができた．この時，利用できるノード数は256から512程度へと増大し，個々のノードには8から32のスレッド並列を行うCPUまたは演算コアが実装されていた．領域分割数が増大するのに伴い，一般に通信コストも上昇する．ただし，袖通信に関しては，分割領域の表面積を小さくすることで，MPIプロセスあたりの計算量に対する通信負荷を減らすことができるので（通信回数の増加による起動時のオーバーヘッドが問題にならない範囲では），多次元の領域分割が有利となる．そこでこれらの計算機に対しては，$(z,v_{\parallel},\mu)$の3次元について領域分割を行うことにした．一方，2次元の波数空間(k_x,k_y)については分割せず，同一のMPIプロセスに納めるようにした．これは第3章で議論された2次元FFTに伴う転置処理とその通信コストの負荷を避けるためである．

速度空間座標($v_\parallel, \mu$)についても分割されたため，それらに関する微分項（ミラー力と衝突演算子）の部分には袖通信が発生する．加えて，式(2)の速度空間積分に伴う総和演算をMPIプロセス毎に並列で行った後，リダクション演算と集団通信を行う必要がある．すなわち，磁気面平均と速度空間積分という2つの異なるリダクション演算を実行することになる．また，μ方向も領域分割するためにループ長が短くなることから，スレッド並列の並列度は4ないし8程度が適度な設定である．

4.2.3 積分演算の実装法

この小節では，上述した数値積分に伴うリダクション演算の実装についてみていこう．並列化していない場合，台形公式に基づいた数値積分の核となる部分は

```
int_phi(:) = (0.d0, 0.d0)
do i = 1, global_nz
  do m =1, nm
    int_phi(m) = int_phi(m) + phi(i,m) * dz
  end do
end do
```

のdoループからなる．ここでglobal_nzは，z方向の全格子点数を，nmは他の次元についての配列サイズを表す．z方向をnproczに領域分割したとすると，MPIプロセス内の和を

```
int_phi(:) = (0.d0, 0.d0)
do i = 1, local_nz
  do m =1, nm
    int_phi(m) = int_phi(m) + phi(i,m) * dz
  end do
end do
```

として求めた後（ここで，global_nz=local_nz*nproczとする），nprocz個のプロセスについて次のようにMPIライブラリを呼び出してint_phiについて総和をとる．

```
call MPI_Allreduce( int_phi, int_phi_all, &
       nm,MPI_DOUBLE_COMPLEX, MPI_SUM, &
       z_comm_world, ierr )
```

ここで，総和の結果はint_phi_allに格納され，z方向分割したすべてのMPIプロセスにおいて同じ値となる．MPI_DOUBLE_COMPLEXは，int_phi，int_phi_allが倍精度複素数型の変数であることを意味する（もちろん，int_phiの型に応じてここは変更する）．ここでは，MPI_SUMというパラメータ変数を指定することで，種々のリダクション演算の中から総和を選んでいる．z_comm_worldは，z方向の分割に伴って定義したコミュニケータである．こうして領域分割された座標方向への積分を求めることができる．

ここで，領域分割の方法に応じて，どのようにコミュニケータを使い分けるかをもう少しみていこう．コミュニケータは，全MPIプロセスのなかの部分集合を定義するものであり，特にその中での集団通信を行うのに有効である．今回の例では，磁気面平均と速度空間積分について異なるコミュニケータを利用することになる．4.2.2小節前半で述べたz方向への1次元領域分割の場合，上の例に挙げたz_comm_worldは，全MPIプロセスからなる既定のコミュニケータMPI_COMM_WORLDに等しい．一方，($z, v_\parallel, \mu$)での3次元領域分割を行った場合，速度空間積分用のコミュニケータも別に用意する必要がある（**図1参照**）．それをv_comm_worldとしよう．それぞれのコミュニケータは，全ランク番号（rank），z方向へのランク番号（zrank）とz方向の分割数（nprocz）から，次のように定義できる．

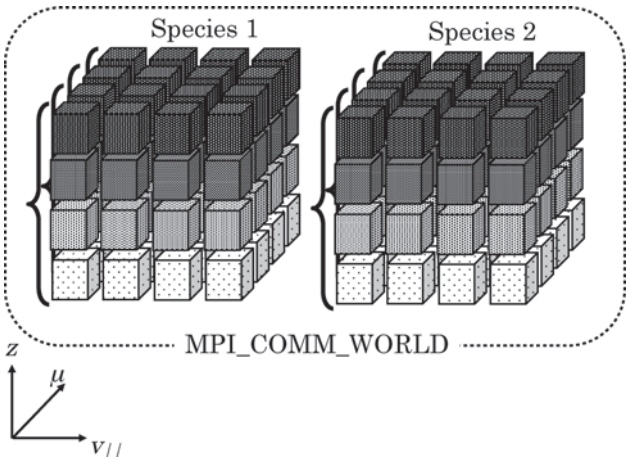

図1 領域分割とコミュニケータの概念図．z方向への分割された領域は異なるパターンで示され，それぞれが2次元速度空間積分を計算するためのコミュニケータ（v_comm_world）を作る．一方，磁気面平均は，z方向のパターンの異なる立方体一つずつからなるカラムでできたコミュニケータ（z_comm_world）内のリダクション演算で計算される．また，それぞれの粒子種が一つのコミュニケータを使って定義されている．

```
rankz = mod( rank, nprocz )
zcolor = rank / nprocz
vcolor = rankz
call MPI_Comm_split( MPI_COMM_WORLD, &
         zcolor, rank,z_comm_world, ierr )
call MPI_Comm_split( MPI_COMM_WORLD, &
        vcolor, rank,v_comm_world, ierr )
```

ここで，それぞれのコミュニケータは，zcolorまたはvcolorが同じ値をもつMPIプロセスのグループから構成される．このようにして，3次元領域分割された変数の数値積分を簡単に計算することができる．

4.2.4 より進んだ計算モデルとその並列化

4.2.1小節では，静電的なジャイロ運動論的方程式にもとづくシミュレーション技法について説明した．現在さらに，電磁揺動，多粒子種，マルチ・フラックス・チューブなどを取り入れた拡張が進められている．これらの拡張で必要となる技法を以下で簡単にみていこう．

ジャイロ運動論ではゆっくりした電磁場変動のみを扱うため，変位電流を無視したアンペールの式を解く．そこで

は，揺動分布関数の1次モーメントで与えられる電流密度をソースとして，磁場揺動のベクトルポテンシャルを求めることになる．コーディング技法としては，静電場を解く場合と本質的な違いはない．

一方，ジャイロ（またはドリフト）運動論的電子や，多成分イオンなどを扱う場合には，2つの方法が考えられる．一つは，分布関数の配列の定義において，粒子種を新たな次元として追加する方法である．この場合，粒子種については並列化が難しくなるとともに，コード全体にわたる改訂が必要となる．もう一つは，粒子種についても分散並列化する方法である．この場合，MPIのランク番号によってどの粒子種を担当するかを決めておき，さらに，それぞれの粒子種をコミュニケータによってグループ分けすればよい．こうすることで，コードの大幅な変更は不要となり，かつ，並列性能を損なうことなく，分散可能な並列数を大きくすることができる．この手法は，また，径電場効果を取り入れるためにヘリカル系プラズマのシミュレーションにおいて進められている，マルチ・フラックス・チューブ・シミュレーションに対しても有効である．

これまでの例では，2次元FFTの効率的な計算のため，波数空間は分割せず一つのMPIプロセスに納めていた．しかし，イオンおよび電子温度勾配乱流が共存する場合など，より幅広い波数空間を扱う際には，波数空間についても領域分割が必要となる．この際に問題となる波数空間内でのデータ転置にともなう通信コストを隠蔽するため，OpenMPを利用した通信部分と演算部分の並列実行手法が新たに開発され，京コンピュータのように10万コアを超える最先端の超並列環境においても，高い実行性能を達成することができるようになった[8]．この新技法を用いた大規模なジャイロ運動論的シミュレーションにより，イオンスケールから電子スケールまでを同時に含んだ乱流揺動のスペクトル構造が明らかにされ，異常輸送現象における複数の物理機構の協調および競合過程を詳らかにすることができるであろう．

4.3 大域的ジャイロ運動論的Vlaosvシミュレーションの技法

4.3.1 大域的モデルとその数値解法

前節で紹介したフラックス・チューブ・モデルと異なり，大域的モデルはトーラスプラズマ全体を計算領域とし，5次元位相空間$(\mathbf{R}, v_\parallel, \mu)$上で定義されたジャイロ中心の全分布関数$f$をジャイロ運動論方程式系

$$\frac{\partial f}{\partial t}+\{f,H\}=\frac{\partial f}{\partial t}+\dot{\mathbf{R}}\cdot\nabla f+\dot{v}_\parallel\frac{\partial f}{\partial v_\parallel}=C(f)+S_{\mathrm{src}} \qquad (4)$$

$$-\nabla_\perp\cdot\frac{\rho_{\mathrm{ti}}^2}{\lambda_{\mathrm{Di}}^2}\nabla_\perp\phi+\frac{1}{\lambda_{\mathrm{Di}}^2}[\phi-\langle\phi\rangle_{\mathrm{f}}]= 4\pi e\left[\int f\delta([\mathbf{R}+\rho]-\mathbf{x})\,d^6Z-n_{0\mathrm{e}}\right] \qquad (5)$$

に従って発展させる．式(4)の左辺は5次元位相空間のポアッソン括弧式$\{,\}$とジャイロ中心のハミルトニアン

$$H=\frac{1}{2}mv_\parallel^2+\mu B+e\langle\phi\rangle_\alpha \qquad (6)$$

を用いて与えられる．ここでは前節と同様に静電近似，断熱的電子応答のイオン系乱流を考えており，$\mathbf{R}+\rho$は粒子位置，eとmはイオンの電荷と質量，$n_{0\mathrm{e}}$は平衡電子密度，ρ_{ti}は熱速度で評価した軌道半径，λ_{De}，λ_{Di}は電子とイオンのデバイ長，$\langle\phi\rangle_\alpha(=\phi)$は軌道平均した静電ポテンシャル，$\langle\phi\rangle_{\mathrm{f}}(=\langle\phi\rangle_{kx})$は磁気面平均した静電ポテンシャルをそれぞれ示す．式(4)は衝突項Cおよび熱源S_{src}がない極限では，リウヴィル方程式となり，fはジャイロ中心の軌道に沿って保存する．式(5)は準中性条件から導かれるポアッソン方程式であり，左辺第1項は軌道効果によるイオン分極密度，第2項は電子断熱応答を示す．

大域的モデルではフラックス・チューブ・モデルで仮定している磁力線に垂直な平面での背景プラズマ分布，背景磁場パラメータの一様性と周期境界条件を利用できないため，様々な問題を検討する必要がある．まず，準定常状態のプラズマ分布における揺動分布関数δfと背景分布関数$f_0(\sim F_{\mathrm{M}})$の時空間スケールの分離を仮定し，前者のみを発展させるδfモデルを計算する場合，フラックス・チューブ・モデルでは，周期境界条件の下で式(1)のように解析的にδfを分離できるのに対し，大域的モデルでは乱流輸送による背景プラズマ分布の緩和や時間発展を打ち消して準定常状態を維持する熱源モデルが必要となる．このような分布固定熱源は，分布補正の時定数の任意性や実験とは異なるソース・シンク分布（例えば，プラズマ中に現れるシンク）など，その取り扱いに注意が必要である．一方，実験条件と同様に一定の熱源とプラズマ境界におけるシンクを与えてプラズマ分布を自由に発展させるfull-fモデルの場合にはプラズマ分布が準定常状態に達するまで長時間のシミュレーションが必要になる．

full-fモデルの長時間乱流シミュレーションでは粒子数保存の誤差がシミュレーションに大きな影響を与えるので，全分布関数fの高い保存精度が要求される．一方，高温の核融合プラズマは粒子衝突による散逸効果が小さく非線形性が強いため，非線形移流項の数値的安定性を維持することが要求される．フラックス・チューブ・モデルの場合には擬スペクトル法の採用によってこの問題を解決しているが，大域的モデルでは背景プラズマ分布と背景磁場の非

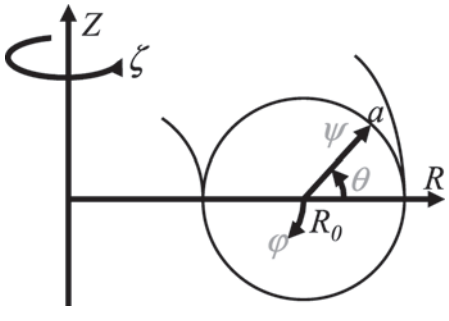

図2 大域的モデルで使用する円柱座標系(R,ζ,Z)および磁束座標系(ψ,θ,φ).

一様性（軸対象トカマクの場合にはトロイダル角についての対称性は存在）および非周期的な境界条件のために，スペクトル法を用いることができない．このため，大域的モデルの場合には精度と安定性を両立する数値手法の選択がきわめて重要になる．

フラックス・チューブ・モデルとのもう一つの大きな違いは，磁気軸の取り扱いが必要になることである．これについては磁束座標系（**図 2** 参照）の構造格子を用いる多くのコードにおいて，磁気軸近傍のクーラン条件の発散を回避するために磁気軸近傍を解かずに境界条件を設定する中空トーラス配位を採用している．一方，円柱座標系を採用して軸の問題を回避するコードも存在する．前者においては，プラズマ中心領域に設定された人工的な境界条件の影響に注意が必要であるのに対し，後者の場合には計算格子が磁気面に沿っていないので，磁気面量や磁気面平均の取り扱いに注意が必要である．

4. 3. 2　GT5D コードの数値解法

次に GT5D コード[5]で用いる数値手法を簡単にまとめよう．式（4）は位置 **R** に関しては円柱座標系において取り扱い，左辺の移流項に 4 次精度の森西差分スキーム[9]を 5 次元一般曲線座標として与えられる案内中心座標系に拡張したスキーム適用する．森西差分スキームはポアッソン括弧式が持っている f および f^2 という 2 つの保存則を同時に満たすように中心差分を修正することによって，数値的な散逸無しに数値安定性を保証し，精度と安定性を両立する．一方，右辺の衝突項は線形フォッカー・プランク演算子に 6 次精度の中心差分を適用し，かつ，衝突演算子の基本的特性である粒子数，運動量，エネルギーの保存則を厳密に満たす手法[10]を採用する．このように，GT5D では full-f モデルの長時間シミュレーションを実現するために式（4）全体を保存型スキームで実装している．一方，式（5）は磁束座標系において取り扱い，軸対称トカマクにおける演算子のトロイダル角 φ についての対称性を利用して，トロイダル方向のフーリエモード展開，および，ポロイダル断面上で 2 次元有限要素近似を組み合わせて静電ポテンシャルを解く．

時間積分には半陰的ルンゲ・クッタ法を適用し，式（4）において磁力線方向の熱運動を円柱座標系で表現することに起因するスティフな線形移流項を陰的ステップとして取り扱う．この陰的ステップの連立一次方程式は～10^{10}次元の巨大な非対称疎行列となるため，非対称行列に対する基礎的なクリロフ部分空間解法として知られている共役残差法[11] に基づく反復法ソルバーを適用する．ここで，フラックス・チューブ・モデルでは磁力線方向の座標 z で熱運動を表現することにより陽解法で時間積分を行えたのに対し，大域的モデルではトロイダル角 ζ で熱運動を表現したために陰的ステップで大規模行列を取り扱うというデメリットが発生する．一方，円柱座標系の採用には，磁気軸の問題を回避できる，トロイダル角 ζ（$=-\varphi$）に中心差分を適用することによって離散的なトロイダル対称性が満たされてトロイダル角運動量が厳密に保存される，あるいは，トロイダル対称性を生かしたフーリエモード展開およびその並列化が可能になるといったメリットもある．このように，大域的モデルの設計においては解きたい問題の特性を考慮した上で各手法のメリット，デメリットを総合的に勘案し，現実的な計算コストの範囲内で最適な座標系や数値手法を選択することが重要である．

4. 3. 3　GT5D コードの並列化方針

GT5D は原子力機構に導入された Altix3700Bx2（Itanium2，2048 コア，13TFlops），あるいは，BX900（Nehalem-EP，17192コア，200TFlops）といった超並列スカラー計算機で開発され，最近では，Helios（SandyBridge-

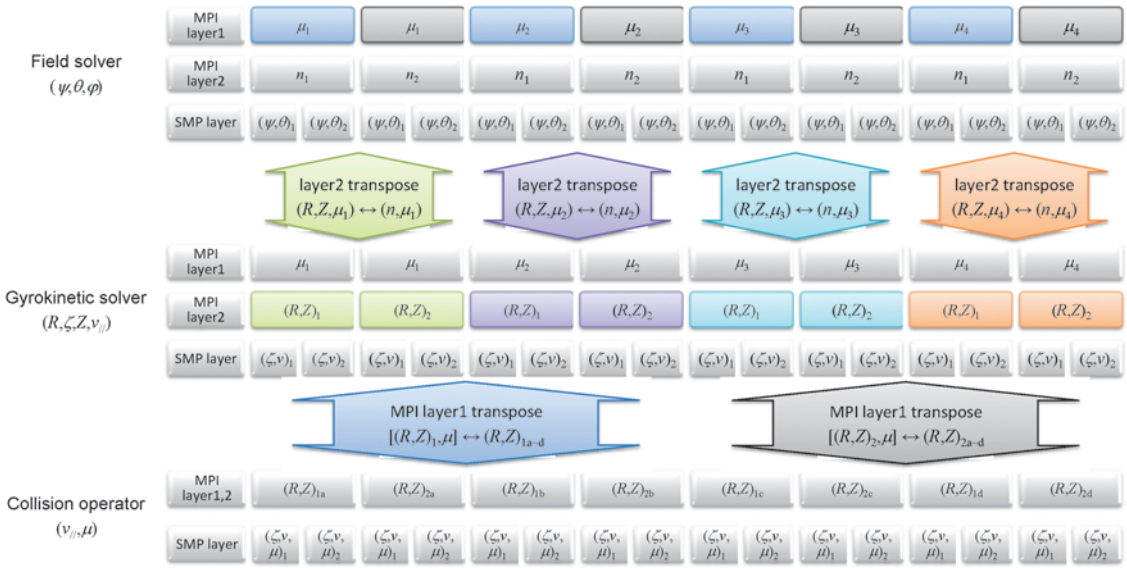

図 3　GT5D における Gyrokinetic solver（式（4）左辺），Collision operator（式（4）右辺），および，Field solver（式（5））の並列化モデル．(R, Z, μ)方向の 3 つの MPI コミュニケータによって構成される 2 階層の MPI ネットワークとマルチコアのスレッド並列による SMP 層から構成される 3 階層の通信ネットワーク上で各演算子の対称性を利用した 3 次元ないし 2 次元の領域分割を実装している．図は μ 方向 4 並列，R 方向 2 並列，Z 方向 1 並列，SMP2 並列の例を示し，通信処理を行うコミュニケータは色のついたボックスで示している．ここで，色の異なる通信処理は独立に処理される．

EP，70560コア，1.5 PFlops）や京（SPARC64VIIIfx，705024コア，11.3 PFlops）といったペタスケール計算機にも移植されている．このような超並列環境における並列化で最も重要になるのが，問題の対称性を利用して階層的な並列化モデルを構築することである（**図 3**）．例えば，式（4）左辺の移流演算子（Gyrokinetic solver）は断熱不変量 μ をパラメータとする 4 次元（$\mathbf{R}, v_\parallel$）の移流項となり，式（4）右辺の衝突演算子（Collision operator）は位置 $\mathbf{R}$ をパラメータとする 2 次元（$v_\parallel, \mu$）の移流 - 拡散項となる．一方，トロイダル方向にフーリエモード展開を適用すると，式（5）（Field solver）はトロイダルモード数 n をパラメータとする 2 次元ポアソン方程式となる．これらの演算子を対称性パラメータについて分割すると，分割後の部分空間の演算は完全に独立に実行できる．このため，まずはこれらの対称性パラメータを用いて粒度の粗い並列化を行うことで，並列処理における通信コストを大幅に削減する．ただし，上記のように演算子毎に異なる対称性パラメータを用いて並列化を行う場合には，ソルバー間でデータ転置処理が発生するため，各ソルバーの演算コストと転置処理コストのバランスに注意が必要である．

次に検討する点が対称性パラメータで分割した部分空間の問題に対する細かい粒度の並列化である．この並列化は通信処理を伴うため，演算量と通信量のバランスを考慮した分割軸と並列度の選択が必要となる．差分法のようなステンシル演算の場合，演算量と通信量は領域分割した部分空間の体積と表面積にそれぞれ比例するため，できるだけ多次元の並列分割軸を利用し，かつ，分割された領域が球（あるいは，直方体）になるべく近くなる並列度の選択を行うことで与えられた並列度に対する通信量を最小化できる．ただし，多次元並列化を行う場合には通信相手の数が増大するため，通信ネットワークのハードウェア機構によっては処理コストが増大する場合もある．

上記の考察に基づいてGT5Dでは各ソルバーに 3 次元ないし 2 次元の領域分割モデルを適用し，第 2 章で説明した MPI 並列化と OpenMP による SMP（Symmetric Multi-Processing）並列化を組み合わせて構成する 3 階層の通信ネットワーク上に実装している．ここで重要な点は，2 階層の MPI ネットワークの一方の分割軸に対称性パラメータを適用することにより，ソルバー内の通信処理，および，ソルバー間のデータ転置処理を 1 階層の部分空間に閉じ込めている点である．これによって，通信処理で使用するコミュニケータのサイズ（通信に参加するノード数）が一桁以上小さくなり，通信コストが大幅に削減されている．

4.3.4 通信マスク手法

上記の階層的並列化モデルによって BX900 上で数千～1 万コアを用いた並列処理が実現し，JT-60U 規模の問題サイズ（～10^{10}格子）に対して 99.9969% という並列化率が達成された．ここで，並列化率 α はアムダールの法則において全体処理に占める並列化されている部分の割合と定義され，1 コアによる逐次処理に対する n コアの並列加速率 S_n は

$$S_n = \frac{1}{1-\alpha+\alpha/n} \qquad (7)$$

と与えられる．逆に，並列化率 α は 1 コアと n コアでの処理時間 T_1 および T_n で与えられる並列加速率 $S_n = T_1/T_n$ から算出できる．ただし，メモリサイズや処理時間の制約により 1 コアと n コアで同じ問題を処理して並列加速率 S_n を求めることが難しい場合には，n コアと m コアでの処理時間 T_n および T_m から得られる並列加速率の比 $S_n/S_m = T_m/T_n$ に式（7）を代入して並列化率 α を算出する．式（7）はコア数 n を増やしていくと，並列化部分 α/n とそれ以外 $1-\alpha$ が釣り合う点 $n_b = \alpha/(1-\alpha)$ が現れ，それを超えると並列加速率はコア数無限大で得られる上限値 $1/(1-\alpha)$ に向かってゆっくりと漸近していくことを示す．このため，あるプログラムおよび問題サイズで実質的に利用可能なコア数の目安は n_b となる．このことは，例えば，100万コアの並列処理を実現するには，プログラムの並列処理性能として 99.9999%（six nine）の並列化率が要求されることを示している．上の GT5D の例では，α = 99.9969% に対して n_b =32,257 となるが，実際に 16,384 コアでの処理コスト分布をみると，通信処理のオーバーヘッドが 3 ～ 4 割を占めて並列加速率が頭打ちになりつつあった．このボトルネックを解決し，10万コア以上のペタスケール計算に十分な並列化率を実現するために，GT5D では通信マスク手法の開発を行った．

図 4 に領域分割した差分演算の例を示す．通常の実装方法は袖領域のデータを隣接ノードから転送し，通信処理の

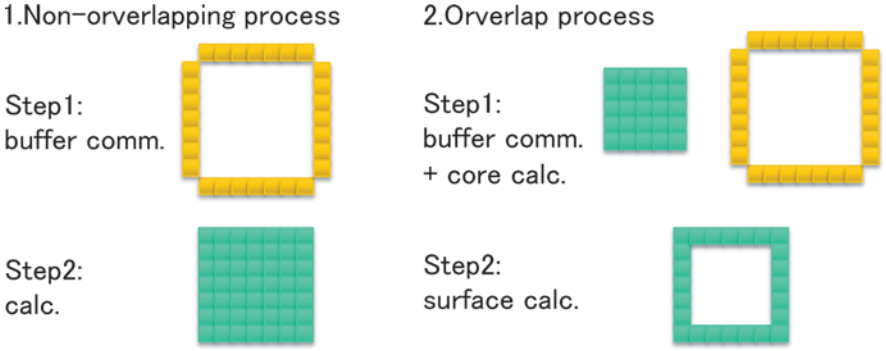

図 4　領域分割した差分演算における通信マスク手法の概念図．通常は袖領域通信を実行してから差分演算を実行するが，通信マスク手法では袖領域通信と内部領域の差分演算を同時処理することによって通信処理のオーバーヘッドを隠蔽する．

完了後に差分演算を実行する．一方，通信マスク手法の場合には，袖領域通信と袖領域を参照しない領域の差分演算を同時実行することによって通信コストのオーバーヘッドを隠蔽する．このような演算と通信の同時処理は，従来，MPI_Isend/MPI_Irecv のような非同期通信関数を用いることで実装可能と考えられてきたが，MPI ライブラリや計算環境によってはこのような標準的な実装が機能しない場合があるので注意が必要である．多くの MPI ライブラリではある程度のデータサイズ以上の転送に RendezVouz プロトコルと呼ばれる通信方式を採用しており，ライブラリの内部ではデータ転送の開始前に通信相手ノードと制御通信を確立するような処理を実行している．しかしながら，通信マスク手法では MPI 関数のコール直後に演算処理を開始するため，通信相手のノードが演算処理に占有されて制御通信に応答できない状況が発生しうる[12]．このような場合，演算終了まで通信処理が中断するため，演算と通信の同時処理が行われない．GT5D の開発では，BX900，FX1，Helios，京等の多くの計算環境でこの問題に直面したため，これを回避する 2 つの方法を開発した[13]．

図 5 は**図 3** における Gyrokinetic solver の差分演算と袖領域通信の実装例を示す．手法 B では，演算開始後に MPI_Test（非同期通信の状態をチェックする関数）のようなダミー MPI 関数を定期的にコールしてノードを一時的に MPI 処理に復帰させることによって RendezVouz プロトコルの制御通信を確立し，通信処理を開始させる．一方，手法 C では，OpenMP のマスター構文を使用して，通信処理の完了までマスタースレッドを通信専用に割り当てることによって通信処理が中断する問題を回避する．手法 B では演算と通信の同時処理が実現するのに対し，手法 C ではマスタースレッドが通信に占有されるため，通信処理のオーバーヘッドがスレッド間で共有されて，例えば，8 スレッド中 1 スレッドを通信に割り当てる場合には通信のオーバーヘッドが 1/8 に減少する．現在のプロセッサは 8 コア，16コアとコア数が増加する傾向にあるため，この方法でも十分にオーバーヘッドの削減が見込める．また，手法 C は非同期 1 対 1 通信だけでなく同期通信，さらには，集団通信にも適用できるため汎用性が高い．

実際，GT5D では**図 3** の Collision operator を計算する前後に発生する MPI_Alltoall によるデータ転置にもこの手法を適用し，通信処理のオーバーヘッドを削減した．**図 6** に示す実装例では，データ転置前後の分割軸に関係しない ζ 方向に処理全体を分割し，さらに，データの転置と逆転置の処理順序を入れ替えて，データ依存性のない ζ 方向インデックスの転置，差分演算，逆転置を組み合わせて同時処理することでデータ転置のオーバーヘッドを削減する．この手法は，前述の GKV の並列化 FFT のデータ転置処理にも応用されている[8]．これらの通信マスク手法の開発によって京において10万コア以上を用いるペタスケール計算が実現し，ITER 規模の問題サイズ（$\sim 10^{11}$格子）の計算で $\alpha = 99.9998\%$（$n_b = 500000$）を達成した．

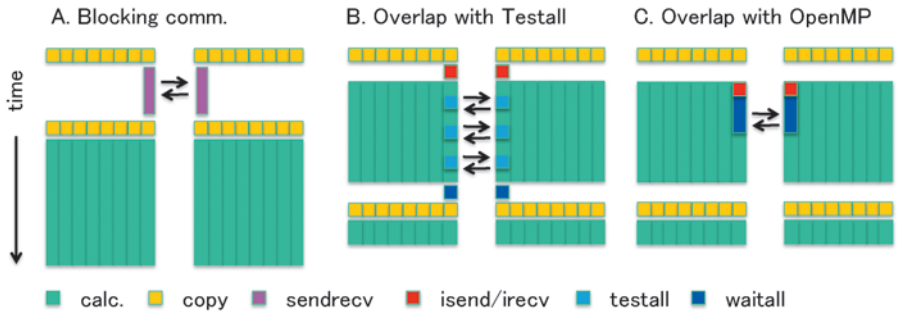

図 5　8 コアの環境において Gyrokinetic solver の差分計算と袖領域通信を実装した例．A は同期通信を用いるオリジナルの実装．B は演算中に MPI_test をコールすることによって非同期通信を促進する通信マスク手法．C はマスタースレッドで通信処理を実行する通信マスク手法．

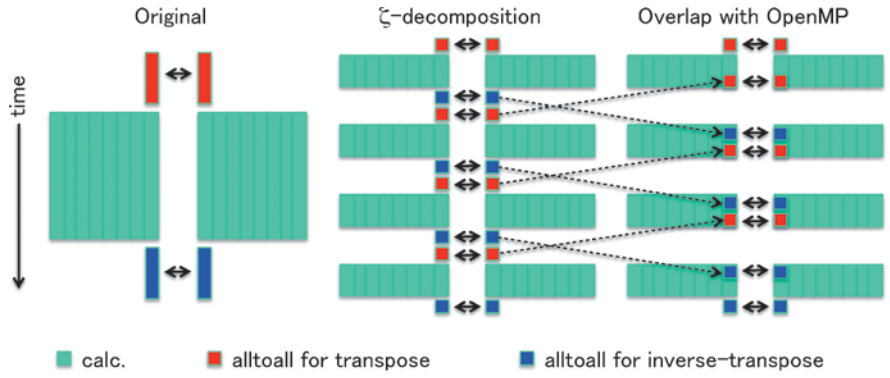

図 6　8 コアの環境において Collision operator の差分演算とデータ転置を実装した例．データ転置前後の分割軸に関係しない ζ 方向に処理全体を分割し（中図），データの転置と逆転置の処理順序を入れ替えて，データの依存性のないインデックスの転置，差分演算，逆転置を組み合わせて通信マスク手法を構築する（右図）．

4.3.5 マルチコアプロセッサにおける最適化

第2章で現在のスカラー計算機におけるメモリの壁の問題が指摘されたが，この問題はしばしばルーフラインモデル[14]に基づいて議論される．このモデルでは，理論演算性能 F (Flops)，理論メモリ帯域幅 B (Byte/s)の計算機上で演算数 f(Flop)，メモリ参照数 b (Byte)のプログラムを実行する際の実効演算性能 S (Flops)が

$$S = \frac{f}{f/F + b/B} \qquad (8)$$

と与えられる．この関係式を変形するとプログラムの対ピーク性能比 S/F がプログラムの演算密度 f/b および計算機の B/F 比によって

$$\frac{S}{F} = \frac{f/b}{f/b + F/B} \qquad (9)$$

と与えられる．**図7**に式(9)を示すが，この関係式は f/b が低い領域ではメモリ帯域幅 B が性能を律速し，f/b が高い領域では演算性能 F によって性能が頭打ちになる状況を示している．図中で $B/F=4$ は地球シミュレータ，$B/F=1$ は Altix3700Bx2，$B/F=0.5$ は BX900 や京に相当する性能であるが，ステンシル計算の演算密度の領域に着目すると，ベクトル・プロセッサでは50%以上を誇っていた CFD (数値流体力学) 計算の対ピーク性能比がシングルコアスカラープロセッサでは20%程度まで低下し，現在のマルチコアスカラープロセッサでは10%程度といわれる実効処理性能の変遷を理解することができる．

このモデルから，現在のマルチコアスカラー機の性能を引き出すには演算密度を向上する最適化がきわめて重要であることがわかる．本節では，そのようなマルチコアプロセッサにおける演算密度を向上する最適化技術を1つ紹介したい．ここで，以下のような4次元の4次精度中心差分を考える．

```
    real*8 f(-1:nx+2,-1:ny+2,-1:nz+2,-1:nv+2)
    real*8 df(-1:nx+2,-1:ny+2,-1:nz+2,-1:nv+2)
!$OMP DO ←デフォルトではブロック分割となる
    do l=1,nv
    do k=1,nz
    do j=1,ny
    do i=1,nx
       fx=a*(f(i+1,j,k,l)-f(i-1,j,k,l))+ &
              b*(f(i+2,j,k,l)-f(i-2,j,k,l))
       fy=a*(f(i,j+1,k,l)-f(i,j-1,k,l))+&
              b*(f(i,j+2,k,l)-f(i,j-2,k,l))
       fz=a*(f(i,j,k+1,l)-f(i,j,k-1,l))+&
              b*(f(i,j,k+2,l)-f(i,j,k-2,l))
       fv=a*(f(i,j,k,l+1)-f(i,j,k,l-1))+&
              b*(f(i,j,k,l+2)-f(i,j,k,l-2))
       df(i,j,k,l)=f(i,j,k,l)+ &
              c*(fx+fy+fz+fv)
    enddo
```

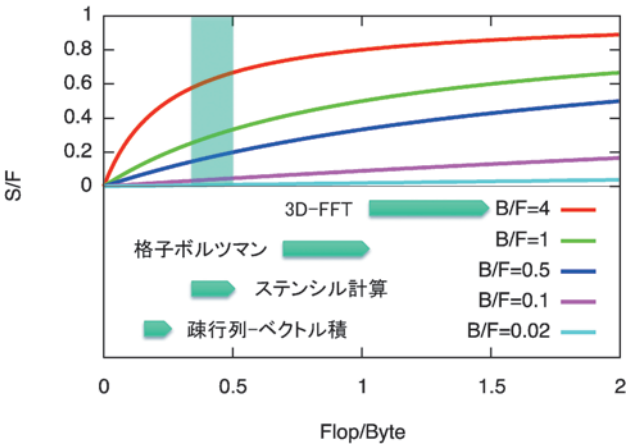

図7 ルーフラインモデルにおける対ピーク性能比 S/F のプログラムの演算密度 Flop/Byte および計算機の B/F 比に対する依存性．図中に示すアプリケーション特性データは文献[14]に基づいている．

```
    enddo
    enddo
    enddo
```

ここで，a，b，c はスキーム，格子幅等によって決まる定数を示す．この差分カーネルの演算数はソースコードより25 Flop となる．一方，メモリ参照数の見積りは少し複雑な計算を必要とする．

第2章でも紹介されたように，ほとんどのスカラー機はキャッシュを搭載した階層的なメモリ機構をもっている．例えば，京の場合には8コアで共有する6 MByte のL2キャッシュを搭載しており，メインメモリのデータ転送速度が64GByte/s となるのに対し，L2キャッシュのデータ転送速度は256GByte/s となる．このため，メインメモリへのデータ参照コストに比べてキャッシュへのデータ参照はほぼ無視できると仮定して上記差分カーネルのデータ参照数を見積もる．ここでは，問題規模として nx=ny=nz=nv=40のケースを考える．Fortran における配列のメモリアドレスは N(i,j,k,l)=i+1+(j+1)*(nx+4)+(k+1)*(nx+4)*(ny+4)+(l+1)*(nx+4)*(ny+4)*(nz+4)のように変化する．このため，i 方向差分では N(i,j,k,l)-4〜N(i,j,k,l)+4 を参照するのに対し，j 方向差分では N(i,j,k,l)-2*(nx+4)〜N(i,j,k,l)+2*(nx+4)を参照し，k 方向，l 方向ではさらに参照するアドレスの範囲が拡大する．このデータサイズを計算すると，i 方向：40 Byte，j 方向：1.68 kByte，k 方向：70 kByte，l 方向：2.96 MByte となる．一方，コアあたりのL2キャッシュサイズは6 MByte/8コア＝750kByte となるため，k 方向まではオンキャッシュとなるのに対し，l 方向はメモリアクセスとなる．この議論に基づいて上のソースコードではキャッシュからロードするデータを実線，メモリからロードするデータを破線，メモリにストアする（正確にはロードしてからストアする）データを点線で表している．この判定条件でメモリ参照数を見積もるとロード5，ロード1/ストア1となり，倍精度配列で8 Byte×7＝56 Byte のメモリ参

照となる．したがって，このカーネルの演算密度は f/b = 25 Flop / 56 Byte=0.45 となる．

これに対して OpenMP のループ分割方法をサイクリック分割に変更したのが，以下の例である．

```
    real*8 f(-1:nx+2,-1:ny+2,-1:nz+1,-1:nv+2)
    real*8 df(-1:nx+2,-1:ny+2,-1:nz+1,-1:nv+2)
!$OMP DO schedule(static,1)←サイクリック分割を指定
    do l=1,nv
    do k=1,nz
    do j=1,ny
    do i=1,nx
       fx=a*(f(i+1,j,k,l)-f(i-1,j,k,l))+ &
              b*(f(i+2,j,k,l)-f(i-2,j,k,l))
       fy=a*(f(i,j+1,k,l)-f(i,j-1,k,l))+ &
              b*(f(i,j+2,k,l)-f(i,j-2,k,l))
       fz=a*(f(i,j,k+1,l)-f(i,j,k-1,l))+ &
              b*(f(i,j,k+2,l)-f(i,j,k-2,l))
       fv=a*(f(i,j,k,l+1)-f(i,j,k,l-1))+ &
              b*(f(i,j,k,l+2)-f(i,j,k,l-2))
       df(i,j,k,l)=f(i,j,k,l)+ &
              c*(fx+fy+fz+fv)
    enddo
    enddo
    enddo
    enddo
```

前のケースでは OpenMP（あるいは，多くの自動並列化）でデフォルトのブロック分割（$n=1,2,\cdots,5$ として m 番目のスレッドに $l=(m-1)*5+n$ を割り当て）となっていたため，各スレッドで全く独立な領域を計算していたのに対し，このケースではサイクリック分割（m 番目のスレッドに $l=(n-1)*5+m$ を割り当て）となる．このため，隣接スレッドがロードしたキャッシュ上のデータを再利用することによって，キャッシュ上のデータだけで 1 方向の差分を実行できる．この結果，差分カーネル全体のメモリ参照数は 24 Byte にまで削減され，演算密度も f/b =25 Flop / 24 Byte=1.04 に向上する．この手法は共有キャッシュを有するマルチコアプロセッサに特有の最適化技術である．GT5D でもこの手法によって Helios および京における差分カーネルの演算密度および実効演算性能を大幅に向上した．

参考文献

［1］洲鎌英雄：プラズマ・核融合学会誌 **79**, 107 (2003).
［2］X. Garbet *et al.,* Nucl. Fusion **50**, 043002 (2010).
［3］渡邉智彦 他：プラズマ・核融合学会誌 **81**, 534 (2005); ibid 581 (2005); ibid 686 (2005).
［4］T.-H. Watanabe and H. Sugama, Nucl. Fusion **46**, 24 (2006).
［5］Y. Idomura *et al.,* Comput. Phys. Commun. **179**, 391 (2008).
［6］M.A. Beer *et al.,* Phys. Plasmas **2**, 2687 (1995).
［7］T.-H. Watanabe *et al.,* Phys. Rev. Lett. **100**, 195002 (2008).
［8］S. Maeyama *et al., submitted to* Int. J. High Perform Comput. Appl. (2012).
［9］Y. Morinishi *et al.,* J. Comput. Phys. **143**, 90 (1998); J. Comput. Phys. **197**, 686 (2004).
［10］S. Satake *et al.,* Plasma Fusion Res. **3**, S1062 (2008).
［11］Y. Saad, *Iterative Methods for Sparse Linear Systems,* Second Edition (SIAM, 2003).
［12］S. Sayantan *et al.,* in Proceedings of PPoPP06 (2006).
［13］Y. Idomura *et al., submitted to* Int. J. High Performance Comput Appl. (2012).
［14］S. Williams *et al.,* Commun. ACM **52**, 65 (2009).

渡邉智彦（わたなべともひこ）

気がつけば，シミュレーションをやるようになってからの人生の方が，その前よりも長くなりました．次はどんなスーパーコンピュータを使えるようになるのか楽しみに，そろそろ，また新しい研究を始めたいと思っています．

井戸村泰宏（いどむらやすひろ）

日本原子力研究開発機構システム計算科学センター所属．ジャイロ運動論シミュレーションによる乱流輸送研究，および，関連する HPC 技術の開発に取り組んでいます．この分野に従事して10年以上になりますが，計算機の目覚ましい進歩にいつも驚かされます．シミュレーション手法やモデルも計算機パワーに負けないように高度化できればと考えています．