ISSN 1344-9567

スーパーコンピューティング ニュース

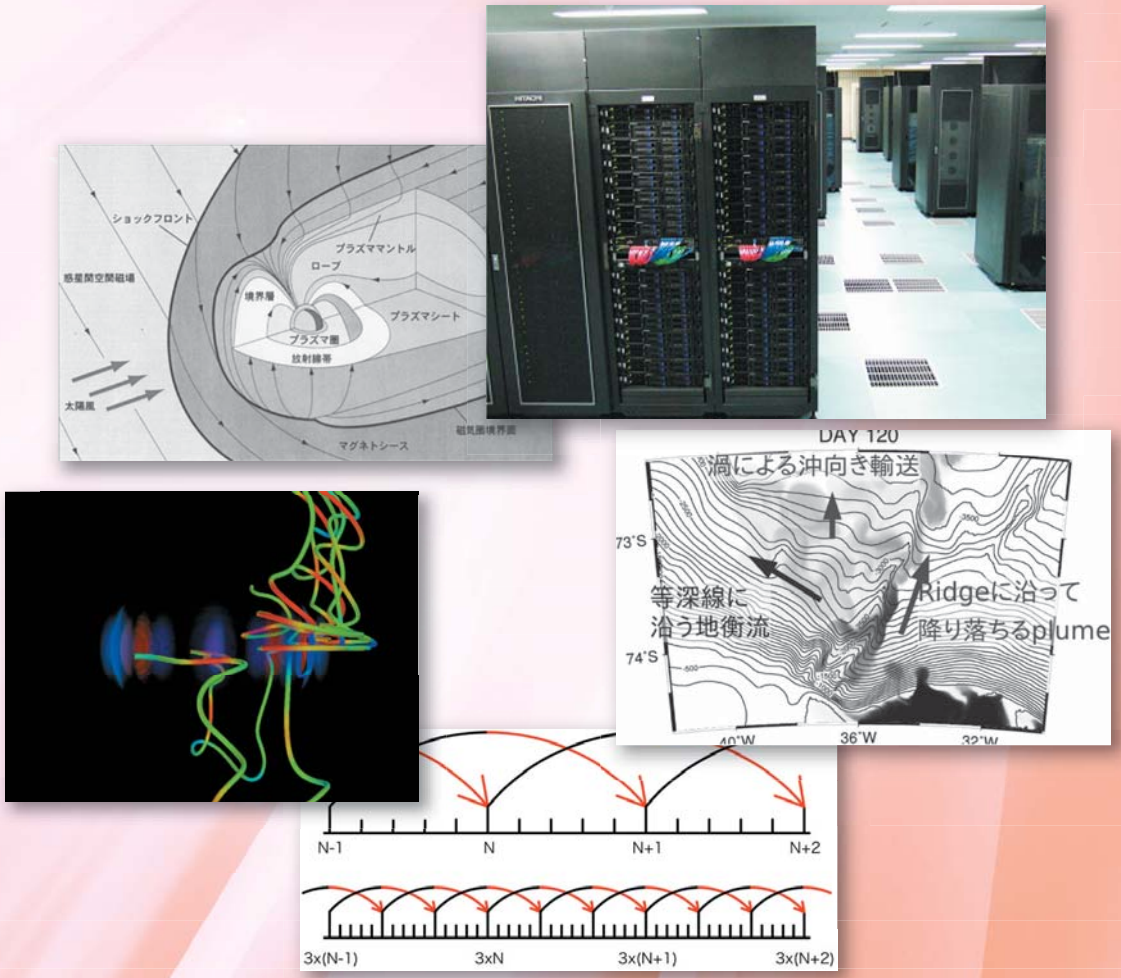
SUPERCOMPUTING NEWS

東京大学情報基盤センター スーパーコンピューティング部門

Vol.12, No.Special Issue 1

2010.2

特集：平成20年度公募型プロジェクト報告



東京大学情報基盤センター
Information Technology Center, The University of Tokyo

目　　次

特集：平成 20 年度公募型プロジェクト報告

平成 21 年度特集号発行にあたって

中 島 研 吾

東京大学情報基盤センター

1. はじめに

本特集号は平成 20 年度に東京大学情報基盤センター（以下　本センター）で実施した 2 種類の公募プロジェクト（T2K オープンスパコン（東大）共同研究プロジェクト，スーパーコンピュータ若手利用者推薦（試行）（後期））で実施された計 9 課題の報告をまとめたものです。本特集号に掲載されている各課題については，2009 年 5 月 11 日（月）に東京大学小柴ホールで開催された『東京大学情報基盤センター平成 20 年度公募型プロジェクト報告会：ペタ/エクサスケールコンピューティングへの道 2009 』で報告されました[1]。

2. T2K オープンスパコン（東大）共同研究プロジェクト

「T2K オープンスパコン（東大）共同研究プロジェクト」は平成 20 年 6 月より稼動を開始した「T2K オープンスパコン（東大)」の利用環境を向上することを目的として，利用者グループ（アプリケーション開発者）とセンタースタッフが共同で研究を実施するものです。「T2K オープンスパコン」64 ノード（1,024 コア）を無料で利用できます。平成 21 年 1 月より開始されたプロジェクトでは以下に示す 5 件の課題が採択されました。

課題名	代表者氏名	代表者所属
電磁流体コードによる大規模惑星磁気圏シミュレーション	深沢圭一郎	九州大学大学院理学研究院地球惑星科学部門
海洋大循環のマルチスケール連結階層モデリング	羽角博康	東京大学気候システム研究センター
津波発生伝播の大規模 3 次元シミュレーション	古村孝志	東京大学大学院情報学環
地球ダイナモの新しいシミュレーションコード開発とその応用	陰山　聡	神戸大学大学院工学研究科
前処理行列の改良による反復法ソルバーの高速化について	久田俊明	東京大学大学院新領域創成科学研究科

本共同研究プロジェクトでは「64 ノード（1,024 コア)」程度を使用する大規模計算を大量に行う研究を対象としました。この共同研究プロジェクトで採択された研究グループは，様々なシミュレーションのアルゴリズムの開発，プログラムの高速化に関する研究を本センターのスタッフと共同で実施し，研究成果については「T2K オープンスパコン（東大)」上でのその成果をライブラリ，HPC ミドルウェア等のアプリケーション開発環境整備にフィードバックするこ

[1] http://www.cc.u-tokyo.ac.jp/publication/sympo/03/

とにより，利用環境の向上に資することを最終的な目標とします。研究可能な協力分野としては以下のような例が挙げられます：

- スカラープロセッサ向けチューニング
- 線形ソルバー（密行列，疎行列）の高速化，チューニング
- 反復法前処理手法
- ハイブリッド並列化
- 並列適応格子法，動的負荷分散
- 細粒度タスク並列化、並列分散プログラミング言語
- ファイル転送効率化
- ユーザー所有クラスタや他のスパコンとの連携

3. スーパーコンピュータ若手利用者推薦（試行）

「スーパーコンピュータ若手利用者推薦（試行)」は平成 19 年度から開始された制度で，概ね 35 歳以下の若手研究者（学生を含む）を対象として，本センターの日立 SR11000，または HA8000 クラスタシステム（T2K オープンスパコン（東大)）を審査の上無料で利用できます。年 2 回公募を実施し，年間 4 件程度の優れた研究提案を採択します。継続申請と再審査を経ることにより，最大 1 年間の無料利用が可能です。採択者は SR11000 コース 3（最大 4 ノード）またはコース 4（同 16 ノード，月末 64 ノード)，HA8000 クラスタシステム コース 4（同 32 ノード，月末 64 ノード)，コース 5（同 64 ノード，月末 256 ノード）を利用できます。平成 20 年度後期は以下に示す 4 件の課題が採択されました。

課題名	氏　名	所　属
実対称固有値問題に対する多分割の分割統治法の分散メモリ型並列計算機への一実装	田村 純一	埼玉大学大学院理工学研究科
企業間取引の大規模ネットワーク構造からみた企業の特徴	大西 立顕	キヤノングローバル戦略研究所, 東京大学大学院法学政治学研究科
厳密な理論波形計算を用いた高解像度地球内部構造推定	竹内　希	東京大学地震研究所
数値モデルによる海洋微小スケールプロセスの解明	松村 義正	東京大学気候システム研究センター

T2K オープンスパコン（東大）共同研究プロジェクト

電磁流体コードによる大規模惑星磁気圏シミュレーション

深沢圭一郎

海洋大循環のマルチスケール連結階層モデリング

羽角　博康

津波発生伝播の大規模 3 次元シミュレーション

古村　孝志

地球ダイナモの新しいシミュレーションコード開発とその応用

陰 山　　聡

前処理行列の改良による反復法ソルバーの高速化について

久田　俊明

電磁流体コードによる大規模惑星磁気圏シミュレーション

深沢圭一郎

九州大学大学院理学研究院地球惑星科学部門

梅田隆行、荻野瀧樹

名古屋大学太陽地球環境研究所

１．はじめに

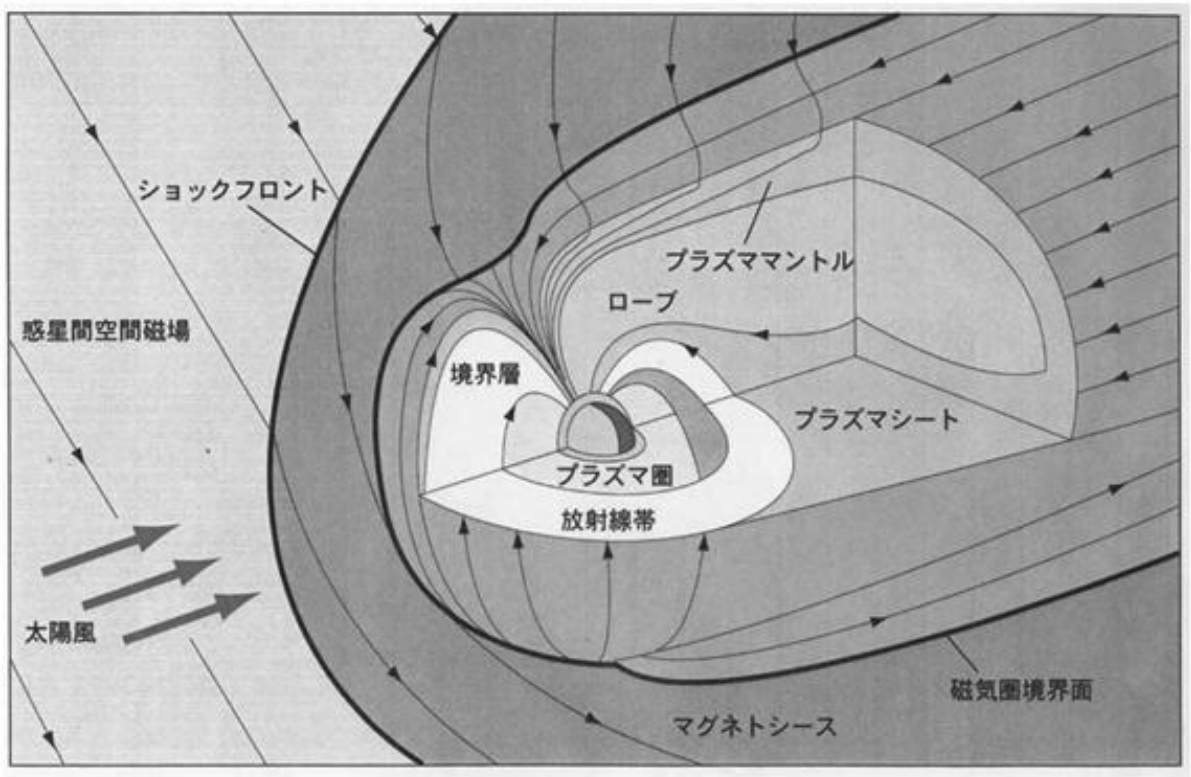
１．１　太陽地球惑星系科学分野の紹介

宇宙空間は真空と思われているが、その 99%はプラズマで満たされている。プラズマとは電離した気体のことであり、帯電している電子とイオンが分かれて存在する状態である。しばしば物質の第 4 の状態とも呼ばれている。宇宙空間、特に我々の暮らす太陽系においては太陽から太陽風と呼ばれるプラズマの風が常時吹き出しており、太陽系全体にそのプラズマが充満している。宇宙プラズマは導電率が高いため、プラズマは磁力線に沿って動きやすく、また磁力線を横切る動きを取りにくい特徴がある。そのため、太陽風プラズマは太陽の磁場を伴って超音速で吹き出しており、地球のような磁化惑星に衝突すると、その磁場を伴ったプラズマの風が惑星の固有磁場と相互作用する。その結果、惑星磁場が変形し、磁気圏という第１図に示すような形をとる。惑星磁気圏の太陽側は太陽風の圧力により圧縮された形をしており、反太陽側は太陽風によって引き延ばされた形をしている。図の左側から太陽風が流れ込み、磁気圏の前面には、弓形の冠衝撃波(bow shock)が形成され、その内側にはマグネトシースが存在する。磁気圏は磁場構造により、内部磁気圏(中低緯度に根ざす閉じた磁力線からなる領域)と外部磁気圏(高緯度側に根ざす開いた磁力線からなる領域)の 2 つに分けられる。その内部磁気圏と外部磁気圏の昼側境界にあたるのが、カスプ領域である。ローブ領域は外側磁気圏で開いた磁力線領域であり、希薄なプラズマが存在している。ローブに挟まれた閉じた領域がプラズマシートと呼ばれる部分で地球の極域電離圏と磁力線を通してつながっている。その境界領域はエネルギーが高くなっている。特に、このプラズマシートは、南と北のローブ領域の磁場で囲まれているため、エネルギーが高く、また、地球に向かう高速の流れが存在し、かつ密度粒子が高く、地球の極域電離層で起こる様々な現象の源となっている。磁気圏境界と呼ばれる部分が、地球磁気圏の殻である。その内側において、尾部の子午面では、カスプからの延長のプラズママントルが、赤道面では、低緯度境界層が存在し、これらの領域では、100 - 150km/s の粒子の流れが観測されている。この磁気圏は基本的に太陽風プラズマに対するシールドとして働いているが、いくつかの条件下では太陽風プラズマが磁気圏内に侵入することがある。その結果、例えば身近な現象としては極域地方でのオーロラ発光という形で表れる。より詳細な紹介については、参考文献[1]などを参考にしていただきたい。

宇宙プラズマ研究において、我々は主にこのような太陽から吹いてくる磁場を伴ったプラズマの風(太陽風)と地球の磁場が相互作用して起こる様々な現象を研究ターゲットにしている。これらは宇宙空間で起きる現象であるため探査機を打ち上げて観測を行うが、基本的に“その場”の観測しか行えない（立体空間情報を得ることができない)。そのため、3 次元空間構造、

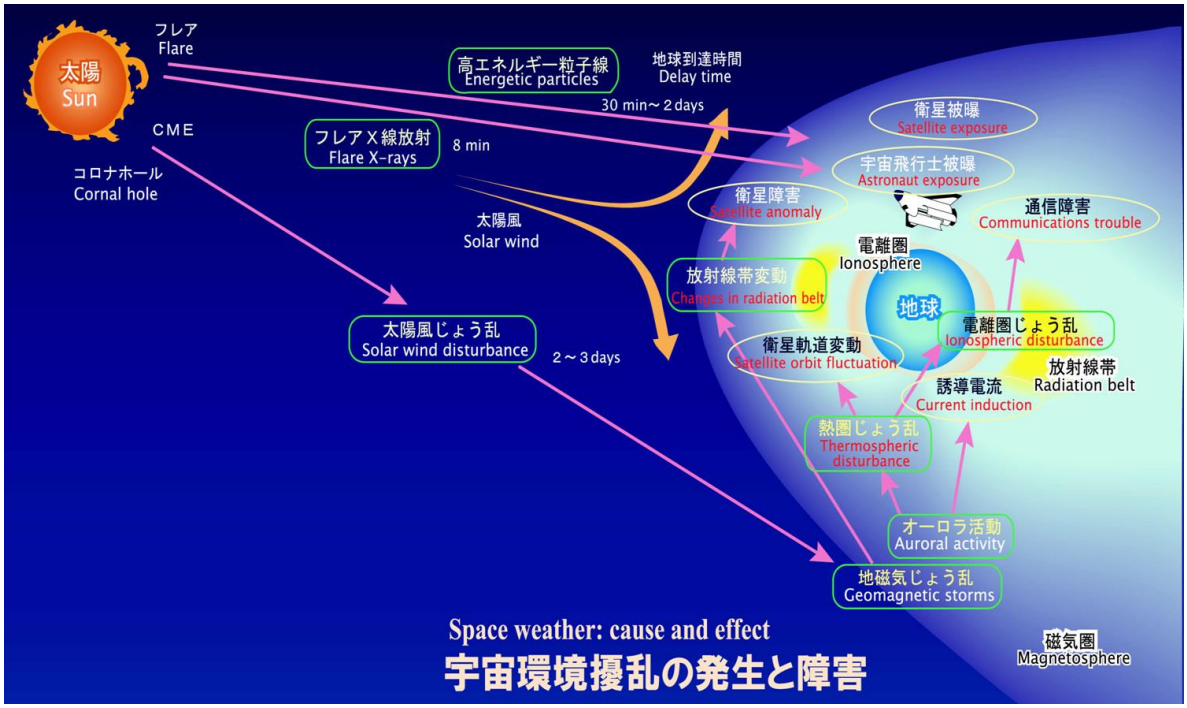
さらにその時間発展などを調べることのできる宇宙プラズマ計算機シミュレーションがこの分野の理論の発展、また観測結果の理解の促進に非常に重要な役割を果たしてきている。

地球周辺の宇宙空間において、前述のような宇宙プラズマに起因する様々な現象を気象にならい、宇宙天気と呼んでいる。その宇宙天気と呼ばれる現象を第2図にまとめた。基本的にこれらの現象は太陽の活動により生じる。例えば、太陽表面でのフレア爆発、コロナから高速にガスの固まりが放出される現象（コロナ質量放出：CME）等がある。それら太陽の活動の結果、身近なものとしては、衛星に障害が生じ、衛星放送に不具合が生じることや、国際宇宙ステーションなどで活動している宇宙飛行士が被爆することなどがある。また磁気圏より地球に近いところに存在する電離圏という領域では太陽活動により、電子密度が変動し、地上でGPS衛星からの電波をうまく受信できないことも起こる。このような現象を引き起こす太陽活動だが、実はその活動度は11年周期で変動している。つまり11年毎に高い活動度（極大期）、低い活動度（極小期）を繰り返している。現在（2009年）は極小期が数年続く珍しい時期に入っており、宇宙天気現象を耳にする日も少ない。しかしながら、今後太陽活動度が上昇していくことが見込まれるので、宇宙天気という言葉を目にする機会も増えるであろう。



第1図：地球磁気圏の構造。

図の左側から超音速の太陽風が惑星間空間磁場を伴って吹いており、それが地球に達すると、地球の前面には衝撃波面が形成される。その衝撃波面より地球側に磁気圏の境界を表す磁気圏境界面が存在し、その中が磁気圏になる。磁気圏は惑星固有磁場の勢力範囲であり、太陽風から地球をシールドする働きを持つ。磁気圏内は、いろいろな領域に分けられており、それぞれ図中にあるような名前が付いている。



第 2 図：様々な宇宙天気現象（情報通信研究機構提供）。

太陽で起きる、フレア、CME（コロナ質量放出）や太陽風により、様々な現象が地球周辺で起こる。例えば、高エネルギー粒子線であれば、宇宙空間にいる宇宙飛行士に被爆をもたらし、太陽風の擾乱により、磁気圏で擾乱が起こり、それに伴い地球周辺の様々な領域で擾乱が起こり、オーロラ活動が活発になったり、衛星の軌道に影響を与えたりする場合がある。日本の宇宙天気研究、予測は歴史的に独立行政法人情報通信研究機構で行われている。

このような宇宙天気現象は地球だけでなく、磁場を持つ木星や土星でも起こる。第 1 表に太陽系内の代表的な磁化惑星の特徴を示すが、惑星に固有磁場が存在すると、前述のように太陽風プラズマとの相互作用が起こり、磁気圏が形成される。そのため、木星、土星でもオーロラが観測されている（第 3 図を参照）。また第 1 表にあるように木星、土星は高速自転しており、地球とは異なった磁気圏構造をしていると考えられる。木星は巨大な磁場を持ち、更に磁気圏内に大量のプラズマを保持したまま高速自転しているために、磁気圏が遠心力によりディスク状に伸びていると考えられている（第 4 図参照）。また固有磁場が非常に強力なためその勢力範囲である磁気圏も非常に大きく、土星磁気圏が木星磁気圏の尾部（反太陽側）に入っている観測結果もある。土星はその特徴からよく地球と木星の中間の惑星と言われる。それは、地球程度の磁場しか無く、一方でガス惑星であり、木星と同様に高速自転し、大きさも太陽系では木星の次の規模であることからきている。しかしながら、最近の研究では二つの惑星の中間という特徴ではなく、土星固有の現象、特徴も見つかってきている。

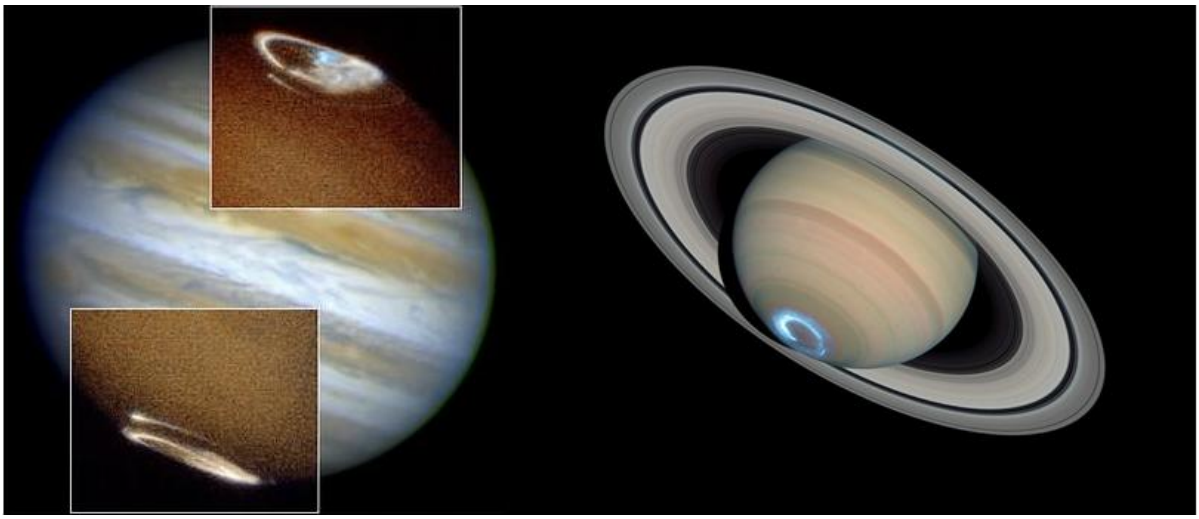
地球においては今までに、また現在もたくさんの衛星が磁気圏を観測し、宇宙天気現象について理解が進んできているが、その他の惑星ではそれほど進んでいない。それでも木星においては、今まで 8 つの探査機（Voyager 1、2、 Pioneer 10、11、 Ulysses、 Galileo、 Cassini、New Horizon）が観測を行い、その中でも Galileo 探査機は 1997 年から 2002 年まで木星を周回観測し、さまざまな情報を与えてくれた。我々のグループにおいてもこの Galileo 探査機の結果に対するシミュレーションを行い、その構造、現象の理解につなげている[2][3][4]。一方で

土星においては、今までに 4 機の探査機（Voyager 1、 Pioneer 10、11、Cassini）が観測を行っており、2002 年から今現在も Cassini 探査機が土星を周回観測している。Cassini 探査機は Galileo 探査機以上に様々な現象を観測しており、現在我々もその理解のため、シミュレーションを行っている[5][6]。第 5 図に我々の土星磁気圏シミュレーション結果を載せる。シミュレーション自体は 3 次元で行っているが、ここでは土星磁気圏赤道面におけるプラズマ対流の磁力線方向に対して垂直（上図)、平行（下図）の渦度を表している。左から右に時間発展を示しており、磁気圏境界面で激しく乱れるプラズマ対流が見て取れる。これは地球、木星磁気圏では見られない構造であり、土星磁気圏の特徴とも考えられている。近年 Cassini 探査機がこの対流構造に似た観測を行い、シミュレーションの正当性が強くなってきている。

第 1 表：地球、木星、土星の特徴。

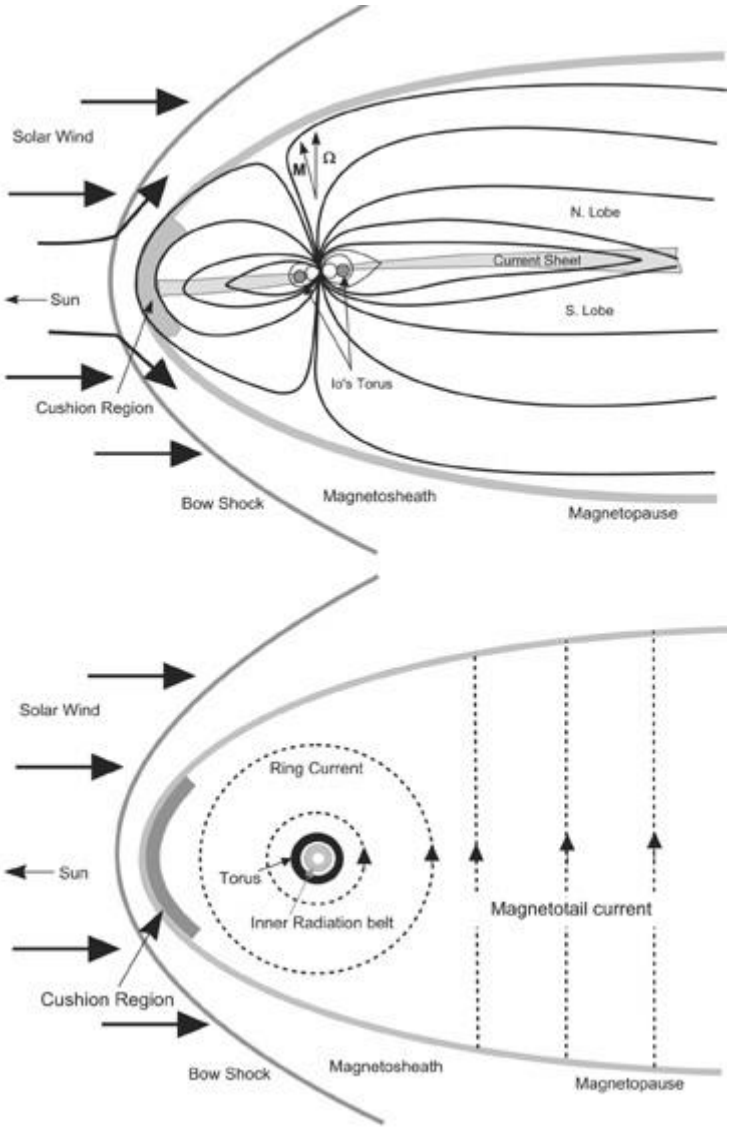
固有磁場は木星が飛び抜けて大きく、地球と土星ではそれほど変わらない。自転速度、赤道半径は木星、土星で同程度であるが、地球に比べると遙かに大きい。プラズマ源は各惑星磁気圏内に広がるプラズマの発生源を示している。イオ、エンケラダスは衛星であり、地球で言うところの月に当たるが、大気を持っている等、その特徴は異なる。木星はおよそ太陽から、地球と比べて 5 倍以上離れており、土星はさらに 9.5 倍離れている。

	固有磁場 [nT]	磁極	自転周期 [hr]	プラズマ源	赤道半径 [km]	太陽からの距離 [A.U.]
地球	31,000	N 極が南	24	電離圏	6,378	1
木星	420,000	N 極が北	10	イオ、電離圏	71,492	5.2
土星	21,000	N 極が北	10.65	エンケラダス、電離圏	60,268	9.55



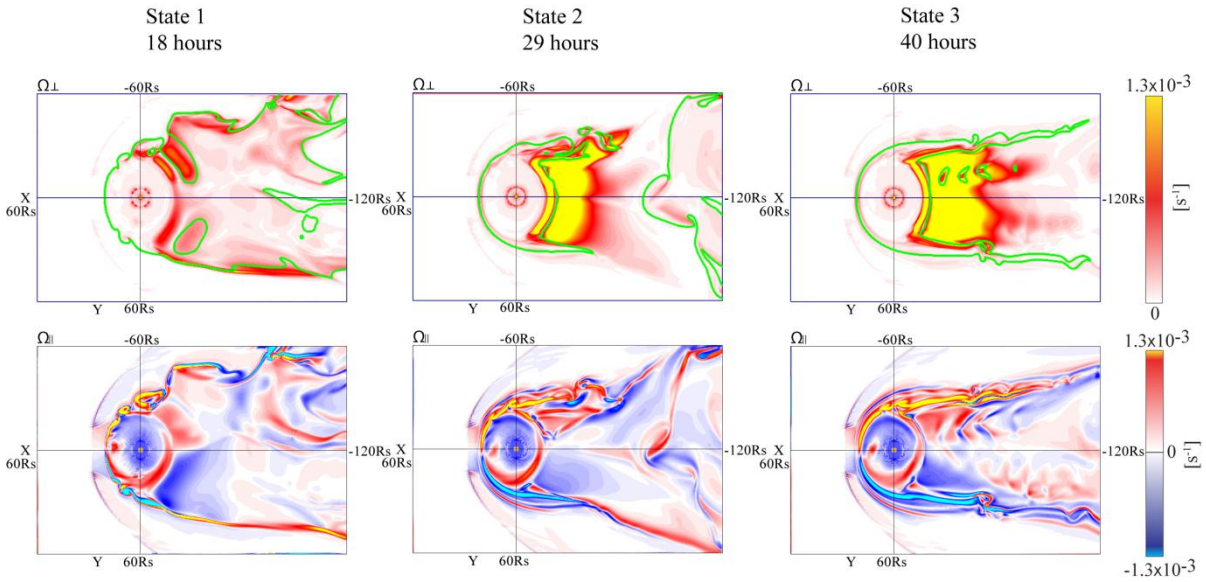
第 3 図：木星と土星におけるオーロラ発光[NASA 提供]。

左側がハッブル宇宙望遠鏡で撮像された木星におけるオーロラ発光（紫外線領域）の写真であり、右図が土星におけるオーロラ（紫外線領域）の写真である。青白く光って見えるのは紫外線領域での撮像のためである。



第 4 図：木星磁気圏の概略図[7]。

上図が子午面（縦に切った面）における木星磁気圏であり、遠心力によって磁力線が左右に伸びているのがわかる。この構造を磁気ディスクと呼ぶ。下図は赤道面における磁気圏構造を示している。木星の衛星イオから吹き出したプラズマが作るイオトーラスが木星の囲むように輪を描いている。これらは地球にはない木星の特徴である。これらによって木星磁気圏独特の性質ができあがっている。



第 5 図: 土星磁気圏シミュレーション結果[6]。

赤道面における磁場に垂直なプラズマ対流の渦度（上図）、平行な渦度。各図の左側から太陽風が吹いてきている。特に下の図で Y 軸の負側でプラズマ対流が乱れているのが見える。上図内の緑色の線は磁力線が閉じているか開いているかの境界を表す線である。つまり磁気圏境界面を書いている。

１．２　プロジェクトの目標

一般に磁気圏の大規模構造をシミュレーションする場合、プラズマを電磁流体と近似した、電磁流体力学（MagnetoHydroDynamics：MHD）シミュレーションを行う。このシミュレーションでは数値計算的な意味においてスケールフリーであり、解像度は計算機の資源によって決まる。現在我々がシミュレーションでは 600×400×400（×MHD 変数:8）の直行格子を使い、解像度は 0.3R_S（1R_S は土星半径を表す：60,000km）であるが、近年の土星磁気圏シミュレーション結果に表れるプラズマ対流の構造（第 4 図参照）を表すには解像度が足りないことが明らかになっている。そこで、本共同研究プロジェクトでは高精細の土星磁気圏グローバルシミュレーションを行うことを目的としている。最終的な目標としては、0.1R_S の解像度（現在の 1/3）で土星磁気圏を解くことである。前述のように土星磁気圏は地球磁気圏と違い自転速度が早く、惑星付近では磁気圏プラズマが自転とともに回転しており、太陽風との相互作用の結果が準定常状態に達するまでに非常に時間がかかり、意味のある結果を得るためには、シミュレーションを長時間走らすことが必要となる。そのため、計算機の実行効率が非常に重要になってくる。

平成 20 年度共同研究では、シミュレーションを始めるに当たって、いわゆる T2K 型、PC クラスタ型の計算機に対する我々の二つのシミュレーションコードの性能評価を行う。一つは今述べた土星磁気圏グローバル MHD コードであり、もう一つは次（々）世代惑星磁気圏シミュレーションコードと言われる Vlasov コードである。共同研究で使用できる T2K オープンスパコンの資源は 64node、1024core までであるので、その core 数までの性能評価を行う。

MHD 方程式は Vlasov 方程式において速度空間を落とすような近似、更には単一流体近似により導かれる（詳細は次のセクションを参照）。この近似により扱えなくなった現象についてはモデル化されたパラメータを使い、現象論として組み込んでいる。現在磁気圏の大規模構造を調べるには、計算資源から考えても MHD シミュレーションを使うしかなく、またその結果も観測、

理論とよく合う。しかしながら、マクロスケールからミクロスケールに、またはその逆を含んだ現象までも調べるためには、MHD シミュレーションでは不可能であり、将来の計算資源の向上を見込んで、Vlasov シミュレーションによる新しい磁気圏コードの開発が行われている。

これら両コードともにベクトル機である地球シミュレータ、NEC SX シリーズ、またスカラ機である富士通 HPC2500 でも最適化により、良い性能を示しており[8]、本稿では、その性能評価について詳しく述べる。

２．磁気圏シミュレーションモデルの概要と特徴

２．１　電磁流体方程式

宇宙プラズマの密度はとても低いために、その平均自由行程が非常に長くなる。例えば、太陽プラズマの平均自由行程は１天文単位（太陽と地球の距離）にも達する。そのため宇宙プラズマは基本的に衝突が無いと見なされる。その無衝突プラズマの振る舞いは以下の Vlasov（無衝突 Boltzmann）方程式によって記述される。

$$\frac{\partial f_s}{\partial t} + \vec{v}\frac{\partial f_s}{\partial \vec{r}} + \frac{q_s}{m_s}(\vec{E} + \vec{v} \times \vec{B})\frac{\partial f_s}{\partial \vec{v}} = 0 \tag{1}$$

ここで $\vec{E}$ 、 $\vec{B}$ 、 $\vec{r}$ と $\vec{v}$ はそれぞれ電場、磁場、距離、速度を表す。また、 $f_s(\vec{r},\vec{v}_s,t)$ は位置−速度位相空間における分布関数であり、 s はイオンや電子など種類を示す。 q_s は電荷を m_s は質量を表す。電磁場の時空間発展は以下のように Maxwell 方程式によって記述される。

$$\begin{aligned}
\nabla \times \vec{B} &= \mu_0 \vec{J} + \frac{1}{c^2}\frac{\partial \vec{E}}{\partial t} \\
\nabla \times \vec{E} &= -\frac{\partial \vec{B}}{\partial t} \\
\nabla \cdot \vec{E} &= \frac{\rho}{\varepsilon_0} \\
\nabla \cdot \vec{B} &= 0
\end{aligned} \tag{2}$$

ここで $\vec{J}$ は電流密度、 ρ は電荷密度、 μ_0 は真空中の透磁率、 ε_0 は真空中の誘電率、 c は光速を示す。電流密度 $\vec{J}$ は帯電した粒子の動きによって以下のように記述できる。

$$\vec{J} = \sum_s q_s \int f_s \vec{v} d\vec{v} \tag{3}$$

また、電流密度 $\vec{J}$ は以下の電荷保存則を満たしている。

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \vec{J} = 0 \tag{4}$$

ここまでにあげた式が無衝突プラズマの第一原理の方程式である。

次に電磁流体力学（MHD）方程式を求めていく。MHD 方程式は Vlasov 方程式(1)の 0 次、1 次、2 次のモーメントをとり、運動論的効果を無視することで得られる。まず Vlasov 方程式の 0 次のモーメントをとる（Vlasov 方程式(1)を速度空間について積分する）と連続の式が求まる。

$$\frac{\partial n_s}{\partial t} + \nabla \cdot (n_s \vec{u}_s) = 0 \tag{5}$$

$\vec{u}_s$ は各プラズマの平均速度を表す。次に Vlasov 方程式(1)に $m\vec{v}$ を掛けてから速度空間で積分する（1 次のモーメントをとる）と、運動方程式が求まる。

$$\frac{\partial}{\partial t}(m_s n_s \vec{u}_s) + \nabla \cdot (m_s n_s \vec{u}_s \vec{u}_s + \vec{P}_s) - \rho_s \vec{E} - \vec{J}_s \times \vec{B} = 0 \tag{6}$$

$\vec{P}_s$ は圧力テンソルを表す。更に、粒子の運動エネルギー $\frac{1}{2} m \left|\vec{v}\right|^2$ を Vlasov 方程式(1)に掛けて速度空間で積分する（2 次のモーメントをとる）と、エネルギー方程式が求まる。

$$\frac{\partial}{\partial t}(\frac{1}{2} m_s n_s \left|\vec{u}\right|_s^2 + \frac{3}{2} p_s) + \nabla \cdot (\frac{1}{2} m_s n_s \left|\vec{u}\right|_s^2 \vec{u}_s + \frac{3}{2} p_s \vec{u}_s + \vec{P}_s \cdot \vec{u}_s + \vec{h}_s) - \vec{E} \cdot \vec{J}_s = 0 \tag{7}$$

p_s は $p_s \equiv \frac{1}{3} \sum_{i=1\sim 3} P_{i,i,s}$ で定義される圧力である。$\vec{h}_s$ は熱流束密度である。ここで以下の操作を行って、単一流体近似をこれらの流体方程式に適応すると、

$$\sum_s m_s n_s \equiv \xi \text{、} \frac{\sum_s m_s \vec{u}_s}{\sum_s m_s} \equiv \vec{U} \text{、} \sum_s \rho_s \equiv 0 \text{、} \sum_s \vec{J}_s \equiv \vec{J} \tag{8}$$

さらに、いくつかのテンソル計算を行って、以下の式を得る。

$$\frac{\partial \xi}{\partial t} + \nabla \cdot (\xi \vec{U}) = 0 \tag{9}$$

$$\xi \frac{\partial \vec{U}}{\partial t} + \xi (\vec{U} \cdot \nabla) \vec{U} + \nabla p - \vec{J} \times \vec{B} = 0 \tag{10}$$

$$\frac{\partial p}{\partial t} + (\vec{U} \cdot \nabla) p + \gamma p \nabla \cdot \vec{U} = 0 \tag{11}$$

ここで $\gamma = \frac{5}{3}$ は比熱比であり、等方対角テンソル $\sum_s \vec{P}_s \equiv pI$ （I は単位テンソル）と仮定したため、熱流束密度は無視できる。更に、磁場凍結条件により、

$$\vec{E} + \vec{U} \times \vec{B} = 0 \tag{12}$$

が求まり、Darwin 近似を用いて、

$$\nabla \times \vec{B} = \mu_0 \vec{J} \tag{13}$$

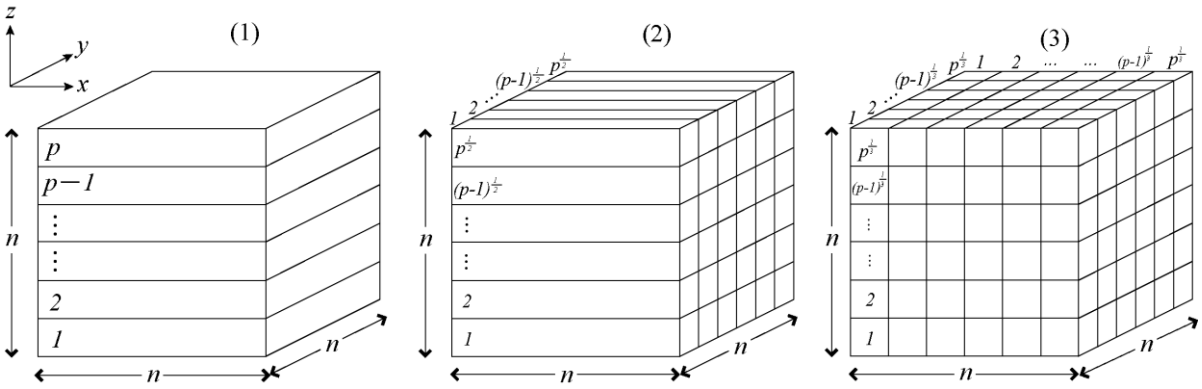
が求まる。これら(9)～(13)式が MHD 方程式となる。より詳細な導出は参考文献[9]を参考にされたい。

２．２　シミュレーションモデル

２．２．１　MHD モデル

MHD 方程式を解く数値計算法としては、Ogino et al. [10]によって開発された Modified Leap Frog 法を使用する。これは最初の１回を two step Lax-Wendroff 法で解き、続く （l-1）回を leap-frog 法で解き、その一連の手続きを繰り返す。l の値は数値的に安定の範囲で大きい方が

望ましいので、２次精度の中心空間差分を採用するとき、数値精度の線形計算と予備的シミュレーションから l =8 に選んでいる。Modified leap-frog 法は、two step Lax-Wendroff 法の数値的安定化効果を一部取り入れて、leap-frog 法の数値的減衰と分散の小さい効果をより多く取り入れた、数値的減衰と分散にバランスの良くとれた一種の組み合わせ計算方法となっている。また、パラメータを変化させることによって、性質の良く分かった２つの計算方法に一致させることができるので、結果に与える数値誤差の影響も理解し易い利点を持っている。更にこの手法を用いた計算で、今まで様々な計算機で性能評価を行ってきたこともあり、同様の手法をもちいることで、過去の結果と比較できる利点もある。



第 6 図: 3 種類の領域分割法。

左から 1 次元領域分割、2 次元領域分割、3 次元領域分割の概要図を示す。n^3 の配列を並列 p で分割している。全並列数を p としているため、2 次元領域分割では各次元で $p^{1/2}$ 並列、3 次元領域分割の場合、x, y, z 方向に $p^{1/3}$ 並列を適用している[11]。

並列化にはMPI を使用する。並列化手法としては3次元空間を分割する領域分割法を用いる。領域分割には、第 6 図に示すように、1 次元、2 次元、3 次元分割が考えられ、本性能評価ではこれらすべての評価を行う。分散メモリ型の並列計算機を用いた並列計算では、3 次元配列に対して領域分割を用いるのが通常である。3 次元モデルの場合、領域分割の次元を 1 次元、2 次元、3 次元に選ぶことができる。その場合の計算時間(T_{S1}, T_{S2}, T_{S3})と通信時間(T_{C1}, T_{C2}, T_{C3})は大まかに次の様に見積もることができる。

i)1 次元領域分割

$$T_{S1} = \frac{k_1 n^3}{p} 、 T_{C1} = k_2 n^2 (p-1) \tag{14}$$

ii)2 次元領域分割

$$T_{S2} = \frac{k_1 n^3}{p} 、 T_{C2} = 2k_2 n^2 (p^{\frac{1}{2}} - 1) \tag{15}$$

iii)3 次元領域分割

$$T_{S3} = \frac{k_1 n^3}{p} 、 T_{C3} = 3k_2 n^2 (p^{\frac{1}{3}} - 1) \tag{16}$$

ここに、k_1 と k_2 は一定の係数、n は 3 次元配列における 1 方向の変数量、p は総並列数である。ここでは簡単のため、領域分割は第 5 図に示すように x, y, z 方向に同じ数(n)の配列を使用し、

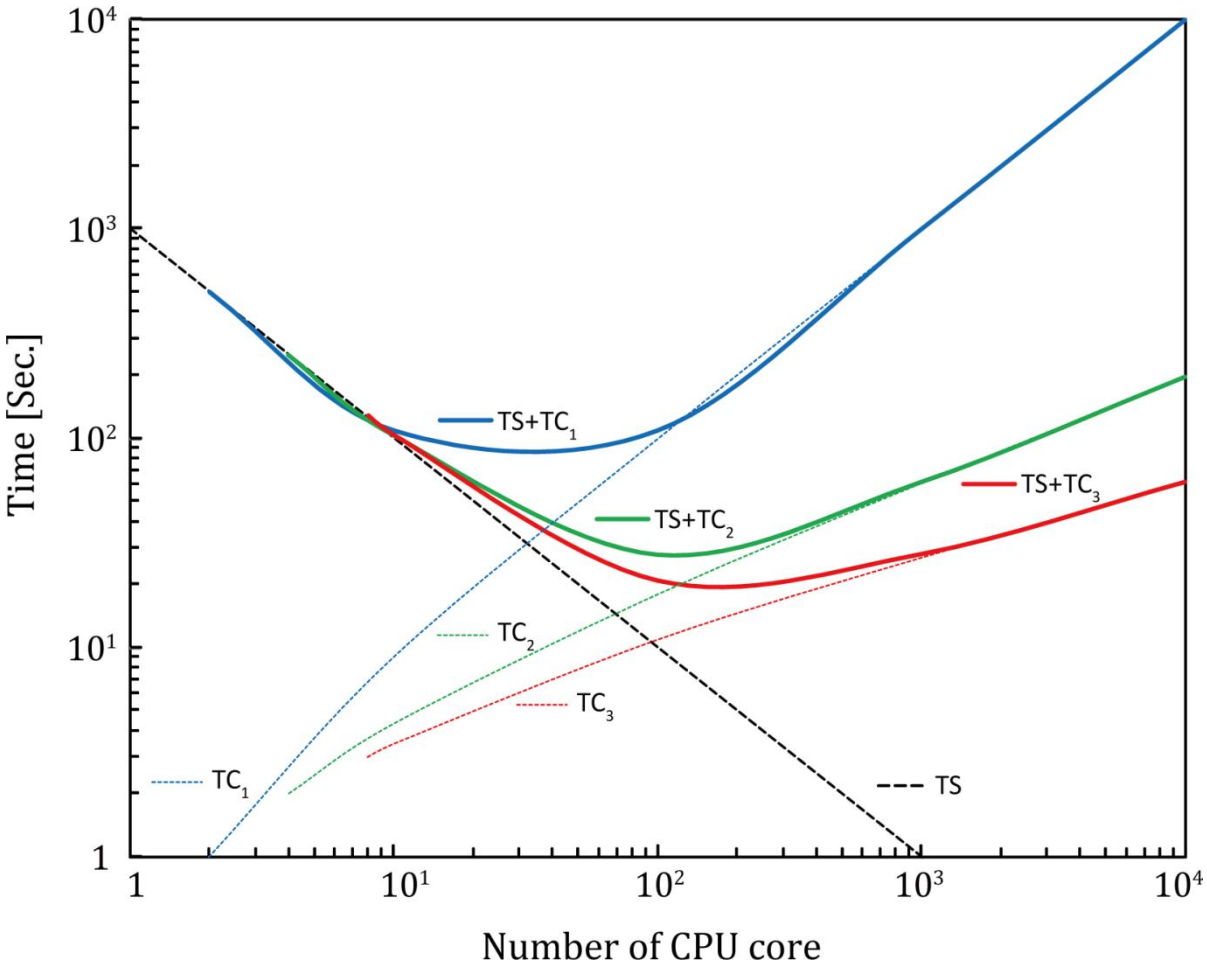
各次元を並列化する場合は等しい並列化数を設定している。計算時間と通信時間の和が並列計算に要する時間と考えられる。これらの式を第7図にグラフで示した。式、図より明らかに計算時間 T_{S1} T_{S2} T_{S3}は並列数 p に反比例して短くなるが、通信時間 T_{C1}, T_{C2}, T_{C3}は p の増加に伴って長くなる。 しかし、その通信時間の長くなる様子は、T_{C1}, T_{C2}, T_{C3}によって大きく異なる（第7図参照)。即ち、3次元領域分割が最も通信時間を短くでき、また、1次元と2次元領域分割の間でも通信時間の差は大きくなることが理解できる。ただし、この比較では簡単のため、通信時間を決める係数 k_2が同じであると仮定した。これは通信部分のプログラムの工夫によりある程度小さくすることが可能である。こうして、スカラ並列機では3次元領域分割が、一方ベクトル並列機では、1つの次元方向はベクトル化に利用する必要があるために2次元領域分割が最も効率的であろうと予想できる。

スカラ機で性能を出すにはキャッシュの有効活用が重要である。基本的な動作としてはデータアクセス時に、その前後含めて数KBのデータをキャッシュに格納する。キャッシュの量や、一度にキャッシュに格納するデータ量はCPUアーキテクチャ毎に変わるので、最高のパフォーマンスを出すにはそれぞれの調整が必要である。MHD シミュレーションにおいては、物理変数がプラズマ密度、速度3成分、圧力、磁場3成分の計8変数となる。そのため、配列を u(i,j,k,m)と定義し（Type A)、m=8としている。数値計算時に、同じ場所の物理変数を何度も使うことになるので、一般に u(m,i,j,k,)と定義した方がキャッシュヒット率は上がると考えられる(Type B)。そのため、本性能評価においてもこの配列定義を使った評価も行う。

２．２．２ Vlasovモデル

今回の性能評価では Umeda et al.によって開発された 5 次元 Vlasov コード[12]を使う。5次元のうち2次元は位置空間(x, y)であり、3次元は速度空間(v_x, v_y, v_z)である。この Vlasov コードは 4 次精度の無振動、正値性のある、保存型スキームであり、多次元保存型 semi-Lagrangian 法の1種である。

Vlasovモデルでは7つの物理変数を取り扱う。分布関数F、電場（Ex, Ey, Ez）と磁場（Bx, By, Bz）である。電荷密度ρと電流密度（Jx, Jy, Jz）も使用するが、これらは分布関数のモーメントを取ることで得られる。この分布関数の計算上の量は電場、磁場の比べて非常に大きい。この Vlasov コードでは電磁場の配列定義には Type B の定義法を使用し（u(m, nx, ny))、分布関数ではモーメントを計算するとき（$\sum v_x \sum v_y \sum v_z F$）にキャッシュヒット率が最大になるような定義を用いた（F(nvx, nvy, nvz, nx, ny))。このため2次元領域分割を今回は適用した。一方で速度空間はモーメント計算時間の短縮のため、分割は行わなかった。



第 7 図:並列数に対する各領域分割における計算時間の変化[11]。

式(14)～(16)によって描かれたグラフ。横軸は並列数、縦軸は時間を示す。点線はそれぞれ計算時間、通信時間を示し、実線が各領域分割にかかる時間（T_S+T_C）を示している。ここでは簡単のため、k_1=1、k_2=0.01 としている。k_n の値に依るが、ある並列数から、計算時間の上昇が見られる。しかしながら、3 次元領域分割の場合、その上昇を最小限に抑えられている。

３．性能評価結果

今回 T2K オープンスパコンでは 3 つの Fortran Compiler が利用できたので、その中から日立最適化 Fortran Compiler と Intel Fortran Compiler Ver. 11.0 を使用した。PGI 製コンパイラではうまく最適化する時間が足りなかった。コンパイラオプションとしては、日立コンパイラには、

```
-Oss -noparallel -autoinline=2
```

を使い、Intel コンパイラには、

```
-O3 -msse3 -xSSE3 -ipo
```

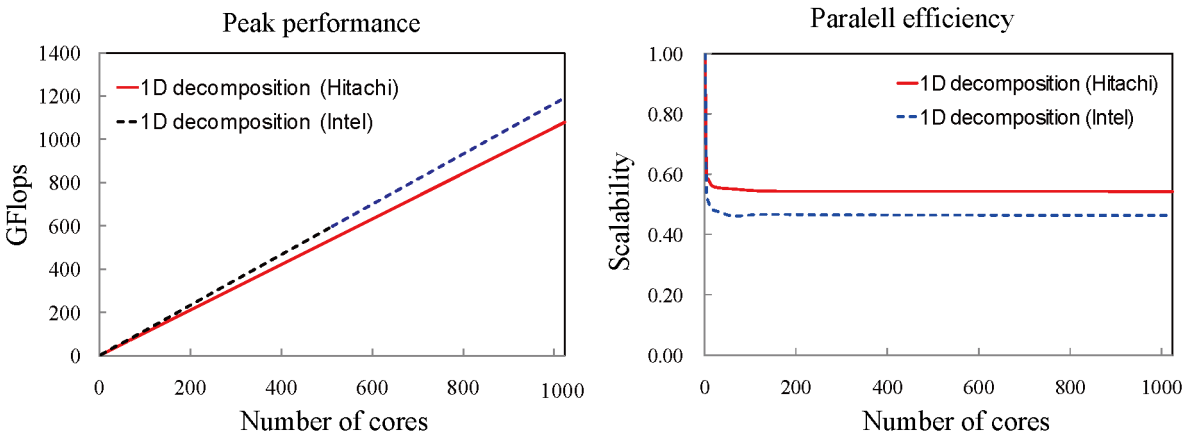
を使用した。これらからわかるように並列化はすべて MPI だけで行った。ここで Intel コンパイラは SIMD Extensions (SSE)をサポートしているが、日立コンパイラはサポートしていない。

並列化時の通信において、通信時間を最小限にするために、すべての境界値を入れるためのバッファー配列を用いた。また、実際の MPI で通信する際には“MPI_sendrecv”を用いた。これは送受信を一括で行う関数になり、送受信に伴うプロセスが一つで済む。例えば、MPI_send/MPI_isend”と”MPI_recv/MPI_irecv”などを使用するとそれぞれに 1 プロセスが必要となる。さらに、blocking と non-blocking の send/receive がよく大容量通信時にバッファオーバーフローするのに対して、MPI_sendrecv はほとんどのスーパーコンピュータシステムにおいて最適化されており、安定した大容量通信が行える。

３．１　MHD シミュレーション

MHD シミュレーションでは 64MB/core（1GB/node）サイズの配列を計算するが、MHD 方程式を Modified LeapFrog 法で解くためのワーク配列として 192MB/core（3GB/node）を追加で使用する。ここでは、1 次元、2 次元、3 次元領域分割の結果を順に述べる。まず、1 次元領域分割の結果について、述べる。基本的に 1 次元領域分割は、小並列のベクトル計算機に最適な並列化手法と言われている。ベクトル化のために do ループを長くとる取る必要があったためと、並列化数が少ない分、通信コストも低いので、次に述べる 2 次元、3 次元領域分割の場合のように通信用のバッファーにデータを集める動作の方が、コストがかかるからである。この手法は Flat MPI を使う上では超並列計算には対応していない（並列化される次元の配列数を並列化数が越えてしまう場合）が、自動並列や OpenMP を併用することで、超並列計算に対応することが可能である。ただし、性能が出るかどうかは現状では不明瞭である（確実なデータが無い)。そのため 1 次元領域分割は昔のベクトル計算機（富士通 VPP5000、NEC SX-6 など）によく使われていた手法で、今回 1 次元領域分割の評価に使用した我々のコードベクトル計算機向けに最適化されたコードでもある。

第 8 図に 1 次元領域分割の結果を載せる。左側の図が並列化数（使用 core 数）に対する実効性能を表しており、右図が並列化効率を示している。赤色の実線が日立製コンパイラを使用した結果を示し、青い破線が Intel 製コンパイラを使用した結果を示している。実効性能は図を見ると明らかなように、グラフが直線を描いており、並列化数に対する理想的な性能向上を示している。最終的な実効性能としては、1024core（64node）使用時に日立製コンパイラで 1,077.8GFlops を達成し、Intel 製コンパイラでは 1,192.8GFlops を達成した。このときの理論性能に対する実効性能の割合を表す実行効率は日立製コンパイラで 11%、Intel 製コンパイラで 13%であった。このようにそれぞれ最高で実効性能 1TFlops（1,000GFlops)、実行効率 10%以上の結果になった。コンパイラでは常に Intel 製コンパイラの方が良い実効性能を出していた。一方、右図の並列化効率を見てみると、日立製コンパイラの方が Intel 製コンパイラよりも高い効率を出している。両コンパイラの結果ともに 4 並列で一気に効率が下がり、16 並列以上では効率は変わらず、図においても横一直線になっている。効率自体は 1024core 使用時に日立製コンパイラで約 56%、Intel 製コンパイラで約 46%となっている。

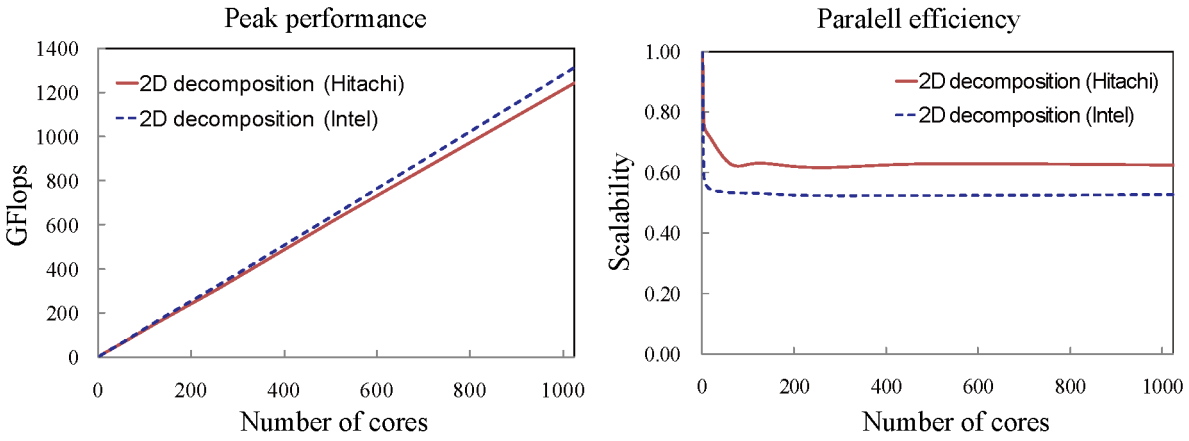


第 8 図:並列数に対する 1 次元領域分割の実効性能と並列化効率[11]。

左図が実効性能を表し、横軸が並列化数（使用 core 数)、縦が GFlops 値を表す。右図は並列化効率を表し、横軸が並列化数（使用 core 数）で縦軸がスケーラビリティを示す。赤色の実線が日立コンパイラの結果を示し、青色の破線が Intel コンパイラの結果を表す。実効性能としては Intel コンパイラの性能が良いが、並列化効率では日立コンパイラに劣る。

次に 2 次元領域分割の結果を述べる。2 次元領域分割は 1 次元領域分割と同様にベクトル計算機向けの手法と考えられているが、特に並列化数の多いベクトル計算機向けである。理由は 1 次元領域分割の場合とベクトル化向けにある次元向けの do ループを長くとることができることと、並列化数が多くなりすぎて、通信コストが無視できなくなっているからである。この場合には、並列化プロセスで通信データをまとめて、各プロセス間で一気に通信する方がコストを低くできると考えられる。この手法は地球シミュレータで特に有効で、2 次元領域分割の評価に使うコードは地球シミュレータで最適化されたコードでもある。

2 次元領域分割の評価結果を第 9 図に載せる。基本的に図の書式は第 7 図と同じであり、左図が実効性能を表し、右図が並列化効率を表している。まず、実効性能だが、ここでも 1 次元領域分割の結果と同様にきれいな右肩上がりの直線を示しており、並列化数の上昇に対する良い性能上昇を表している。1024core における実効性能としては、日立製コンパイラで 1,241.6GFlops を達成し、Intel 製コンパイラで 1,313.0GFlops を達成した。このときの実行効率は日立製コンパイラで 13%、Intel 製コンパイラで 14%となった。これらの結果は日立製コンパイラ、Intel 製コンパイラのなかでそれぞれ最高の結果である。また、1024core 時の日立製コンパイラ、Intel 製コンパイラの性能差が、1 次元領域分割の場合に比べて、少なくなっている。次に並列化効率だが、これも 1 次元領域分割と同様に日立製コンパイラの方が Intel 製コンパイラよりも良い効率を出した。基本的に並列化効率の下がり方などの変化は 1 次元領域分割と似ており、4 並列ですぐに性能が落ち、その後はフラットな直線を描いている。ただし、並列化効率自体は各コンパイラで 1 次元領域分割よりも 10%程度上昇しており、1024core 使用時に日立製コンパイラで 62%、Intel 製コンパイラで 53%の効率が出た。これらも今回の評価のなかで最高の結果である。



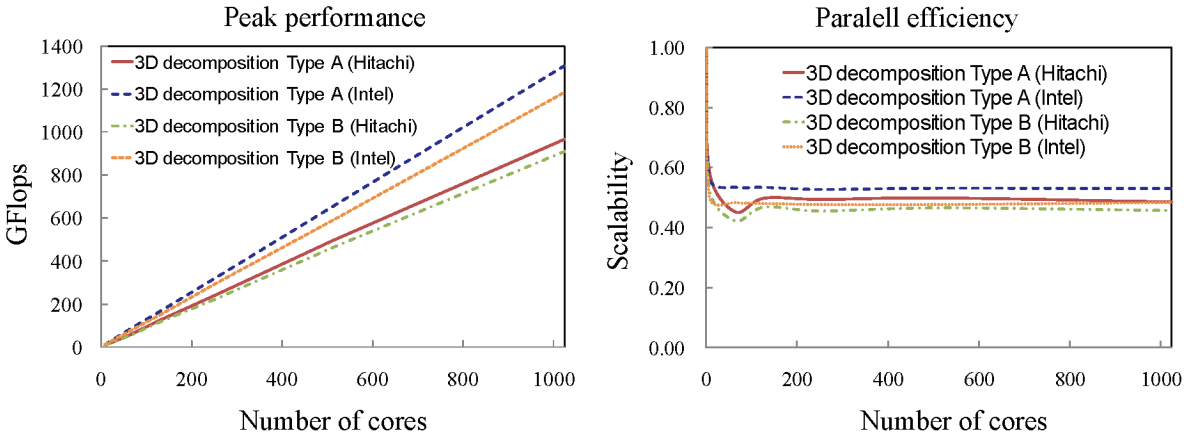
第 9 図:並列数に対する 2 次元領域分割の実効性能と並列化効率[11]。

左図が実効性能を表し、横軸が並列化数（使用 core 数)、縦が GFlops 値を表す。右図は並列化効率を表し、横軸が並列化数（使用 core 数）で縦軸がスケーラビリティを示す。赤色の実線が日立コンパイラの結果を示し、青色の破線が Intel コンパイラの結果を表す。両コンパイラの結果でも並列化効率が他の領域分割よりも良い結果になっている。

次に 3 次元領域分割の性能評価について述べる。3 次元領域分割は、1 次元、2 次元領域分割と比べて、並列化数の上昇に対する通信コストが低いために、超並列計算機向けの手法と言われる。3 次元領域をすべて並列化するために、基本的にはスカラ計算機向けの手法である。われわれは、一昔前にベクトル計算機からスカラ計算機（富士通 HPC2500）へのシフトを経験し、そのときに 3 次元領域分割による最適化を行い良い性能を得た。また、最近ではプレペタスケールコンピュータとも言われる富士通 FX1、日立 SR11000、SR16000 においても 3 次元領域分割で良い性能評価結果を得ている。これらはスカラ計算機ではあるが、T2K オープンスパコンのような一般的な x86 系のプロセッサーを使った PC クラスタタイプの計算機ではないので、同じ傾向が出るとは限らない。

第 10 図に 3 次元領域分割の結果を載せる。前述のように 3 次元領域分割では二つの配列定義を用いたため、図中に 4 つの結果が並んでいる。Type A の配列定義は f(i, j, k, m)の順番であり、Type B の配列定義は f(m, i, j, k)と MHD 変数を示す m を配列の初めに持ってきている。図の書式自体は第 7 図、第 8 図と同様であり、左図が実効性能、右図が並列化効率を示す。まず、実効性能を見ると、Intel 製コンパイラの Type A、Type B の方が、日立製コンパイラの結果より、明らかに性能が出ている。この性能差は前述の 1 次元、2 次元領域分割よりも大きい。実効性能自体は、1024core において、日立製コンパイラでは 1,000GFlops に達せず、Type A で 965.2GFlops（実行効率 10%)、Type B で 908.4GFlops（同 9.6%）であった。一方 Intel 製コンパイラでは、Type A で 1,306.6GFlops（同 14%)、Type B で 1,183.5GFlops（12.6%）を達成した。Intel 製コンパイラでの Type A は 2 次元領域分割の結果と同程度であり、最高の性能を出している。また、配列定義の結果に注目してみると、日立、Intel 製の両コンパイラで Type A の結果が Type B を上回っている。この結果は今までのスカラ計算機(HPC2500、FX1、SR11000、SR16000）とは異なる結果であった。この手法でのキャッシュチューニングは T2K オープンスパコン HA8000 には適さないようだ。並列化効率は並列化数増加に対する変化の様子は、1 次元領域分割、2 次元領域分割と同様であった（すぐに効率が下がり、その後はフラット)。Intel 製

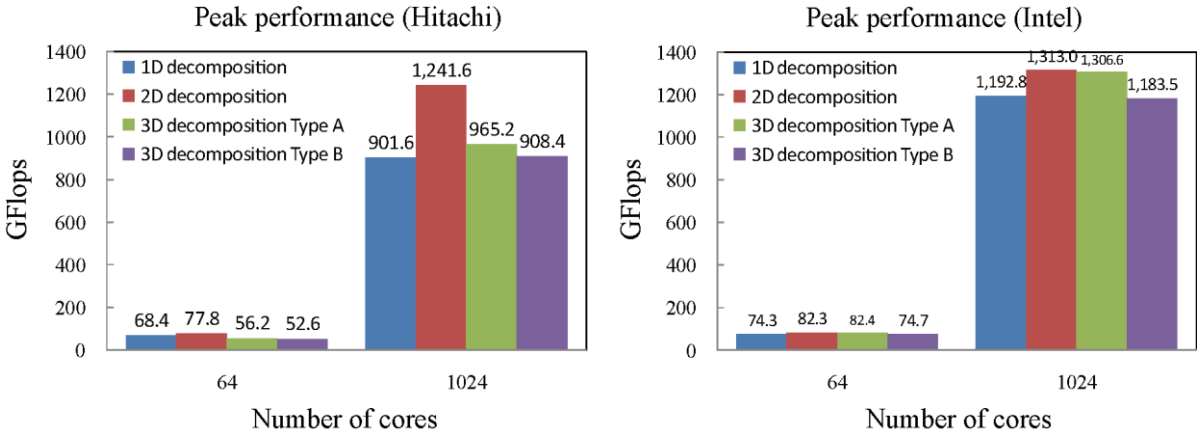
コンパイラで Type A の場合以外は、並列化効率が他の領域分割と比べて良くない。



第 10 図:並列数に対する 3 次元領域分割の実効性能と並列化効率[11]。

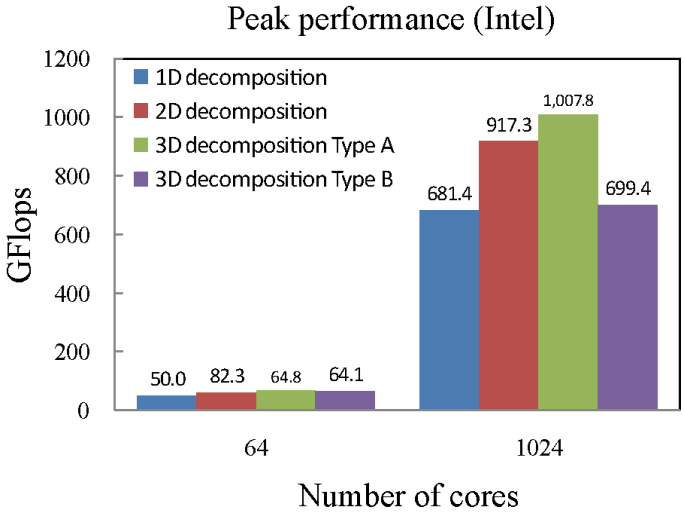
左図が実効性能を表し、横軸が並列化数（使用 core 数）、縦が GFlops 値を表す。右図は並列化効率を表し、横軸が並列化数（使用 core 数）で縦軸がスケーラビリティを示す。ここではキャッシュヒットを考えて、2 種類の配列定義をテストした。Type A が 1 次元、2 次元領域分割と同じ配列定義（f(i,j,k,m)）であり、Type B は MHD 変数を配列の 1 次元目に移動させた定義（f(m,i,j,k,)）になる。他のスカラー計算機と比べて、この性能評価では Type B の結果があまり良くない。

ここで、64core と 1024core の結果を使い、各領域分割の性能を第 11 図に載せ、比べてみる。左図では日立製コンパイラ、右図では Intel 製コンパイラの結果を載せている。まず、全体を見て明らかなように、2 次元領域分割が日立製コンパイラでも Intel 製コンパイラでも良い性能を出しており、HA8000 には 2 次元領域分割が適しているとわかる。特に日立製コンパイラの結果では、2 次元領域分割以外は明らかな性能下降が見られる。一方 Intel 製コンパイラでは、全体的な差は少なく、その中では 2 次元領域分割、3 次元領域分割の Type A が良い性能を出していることが分かる。Intel 製コンパイラの結果が全体的に良い理由は SSE が有効に使えていることが考えられる。事実、SSE を有効にしない場合の結果は、全体的に日立製コンパイラより 1%程度実行効率が悪く、3 次元領域分割の Type A でやっと 1TFLops を越える程度であった（第 12 図参照）。また、Type B の性能が良くない理由はキャッシュサイズに問題があると考えられる。今までに Type B により良い結果を得た計算機は基本的に 1core に対して大きな 2 次キャッシュ（2MB 以上）を有している。一方で HA8000 の CPU である AMD Opteron 8356 は L2 が 512 KB/core、L3 が 2 MB/cpu つまり、512KB/core である。このため、計算上使用する MHD 変数分の配列がうまくキャッシュに収まらないと考えられる。



第 11 図:各領域分割の実効性能。

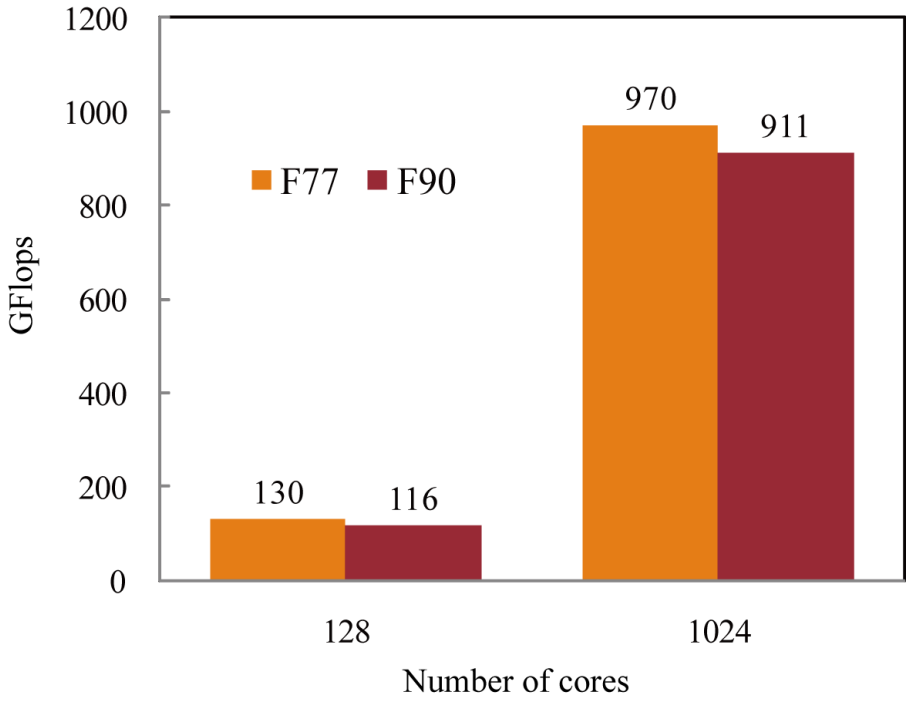
左図が日立コンパイラの実効性能を表し、横軸が並列化数（使用 core 数）、縦が GFlops 値を表す。右図は Intel コンパイラの結果を表し、書式は左図と同じである。Intel コンパイラは全体的に性能が出ているが、日立コンパイラでは 2 次元領域分割以外はあまり性能が出ていない。



第 12 図:SSE を無効にした各領域分割の実効性能。

横軸が並列化数（使用 core 数）、縦が GFlops 値を表す。コンパイラオプションにより、SSE を有効にしない場合、Intel 製コンパイラの性能は著しく劣化する。

ここからは、MHD コードの性能評価時に調べたいくつかの結果を紹介する。まず、Fortran のバージョンで性能が変わるかを調べた。よく Fortran77 の方が、Fortran90 以降に比べて、シンプルな分コンパイラが最適化しやすく、性能が出るといわれる。最近ではプログラムのメンテナンスなどの理由により、Fortran90 を使われる人が多いが、われわれのコードで違いが出るかを調べた。第 13 図にその結果を載せる。これは日立製コンパイラを使った、1 次元領域分割の結果であるが、1024core 使用時は明らかな（誤差範囲ではない）差が見えている。おおよそ 6%ほど Fortran77 の方が良い性能を出している。今回の性能評価全体では Fortan90 を使ったが、最高の性能を出すには Fortran77 を使用する方が良いと考えられる。

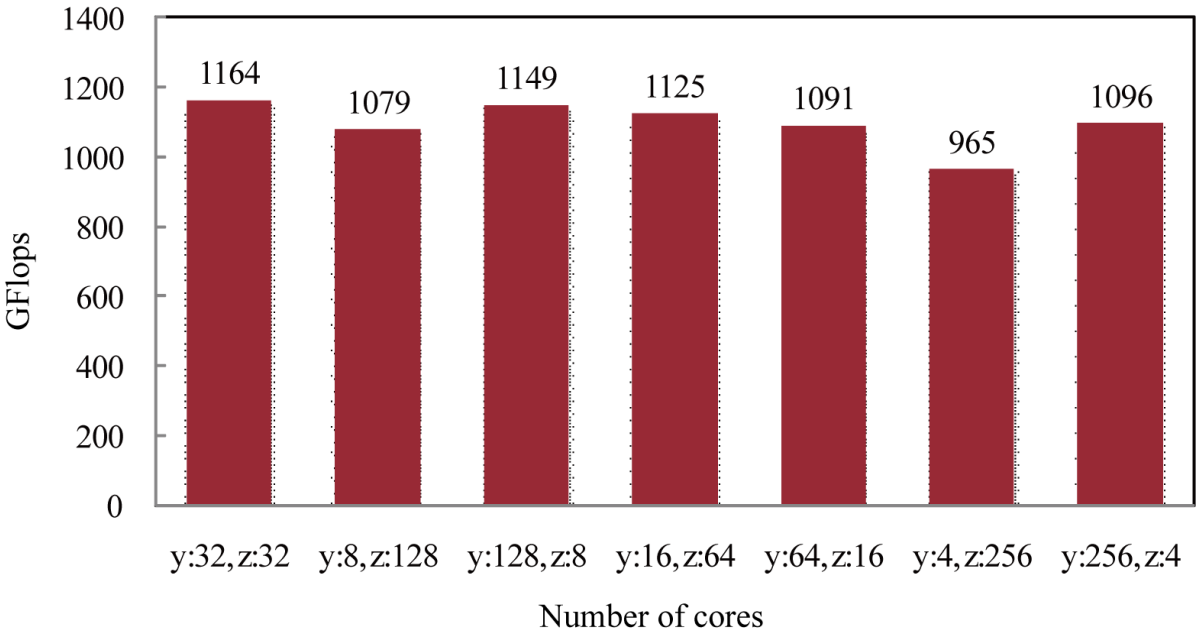


第 13 図:Fortran77 と Fortran90 の性能比較。

オレンジが Fortran77 の結果であり、赤が Fortran90 の結果である。左図が実効性能を表し、横軸が並列化数（使用 core 数)、縦が GFlops 値を表す。これは日立コンパイラの結果である。少しではあるが、Fortran77 の結果が良い性能を示した。

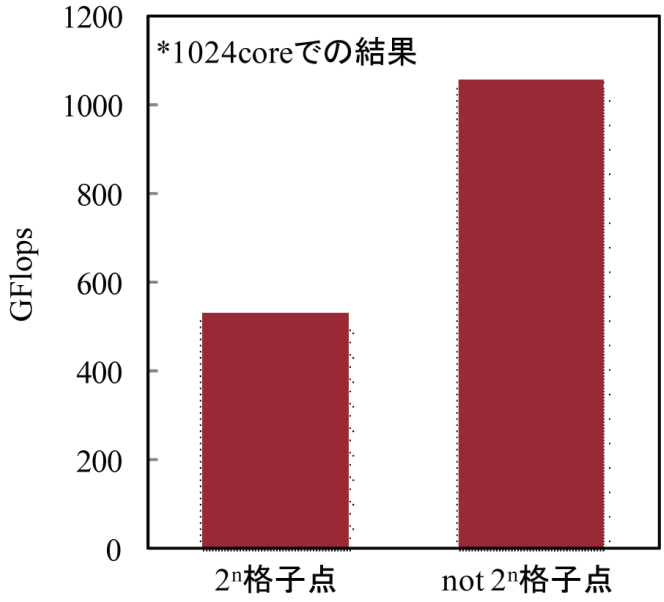
次に 2 次元領域分割における各次元の変列数を変化させた場合、どのように性能が変わるのかを調べた。第 14 図に 1024core 使用時の y 方向と z 方向の並列化数を変化させた場合の結果を示す。これらはすべて日立製コンパイラの結果になる。図を見て気付くのは、z 方向に 256 並列を行った場合の性能劣化である。Y 方向に 256 並列させた場合に比べて 100GFlops 程度性能が下がっている。Z 方向に 128 並列の場合もそれほど良い性能ではなく、極端に z 方向に並列化数を増やすと性能が落ちる傾向が見られた。おそらくメモリアクセスの最適化に問題が出るためと考えられる。この結果から 2 次元領域分割では 32 並列×32 並列が最も良い性能が出ることが分かったので、前述の 2 次元領域分割では 32×32 並列を使用した。

最後に使用配列が 2 の n 乗であるとき、キャッシュミスによる性能劣化があるかどうかを調べた。これは配列のサイズにより、メモリアクセス時に毎回キャッシュミスを起こしてしまうような状況を想定している。第 15 図にその結果を載せる。左側が配列のサイズを 2 の n 乗に定義した場合、右側が 2 の n 乗ではない場合の結果である。明らかに 2 の n 乗に定義した場合で性能の劣化が見られる。これは基本的にどの CPU アーキテクチャでも起こる問題ではあるが、コンパイラオプションにより回避可能なことも多い。



第 14 図:2 次元領域分割における各次元の並列数の変化に対する実効性能。

ここでは、合計 1024core を用いて、y 方向の分割数、z 方向の分割数を変化させた。横軸が y, z 方向における並列化数、縦が GFlops 値を表す。y、z 共に等しい数の並列数が良い性能を出しているが、y の分割数が多い場合より、z の分割数が多い方が性能が落ちやすい結果になっている。これは日立コンパイラによる結果である。



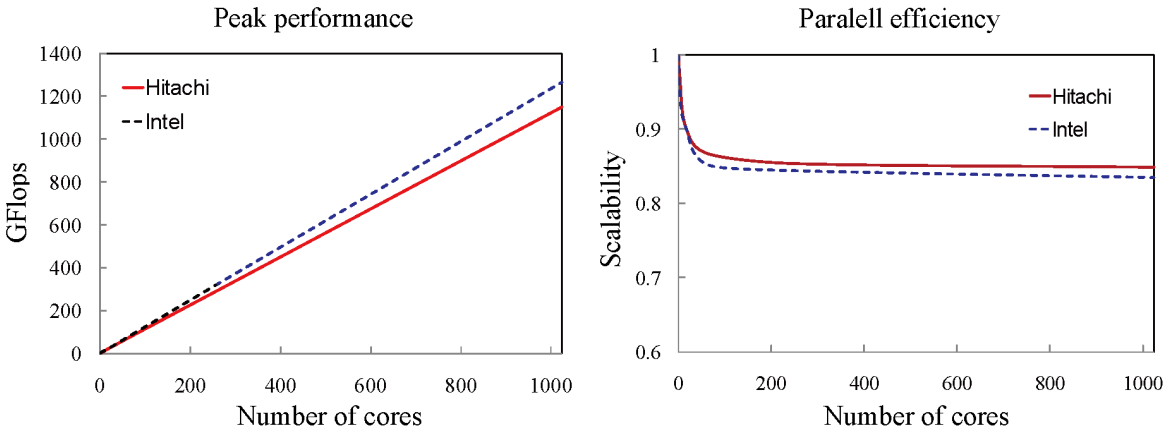
第 15 図: 2 のべき乗個の格子点を使った場合の性能劣化

左が 2^n 個の格子点数を使った場合、右側が 2^n 個ではない格子点を使った場合の結果。縦軸は GFlops 値を表す。明らかな性能劣化が確認できる。この結果は日立コンパイラによるものである。

３．２　Vlasov シミュレーション

Vlasov シミュレーションでは、1GB/core（16GB/node）の配列を使用した。ここで使用した無振動保存型スキームでは境界値が前後で 6 グリッド分必要のため、並列化時の通信が MHD コ

ードよりも多くなる。また前述のように 2 次元領域分割を位置空間に適用する。これは MHD コードの結果を受けて、2 次元領域分割が最適だと考えられるからである。第 16 図に Vlasov シミュレーションの性能評価を載せる。図の書式は第 8 図、第 9 図と同じで、左図が実効性能、右図が並列化効率を示す。赤色の実線が日立製コンパイラを使用した結果を示し、青い破線が Intel 製コンパイラを使用した結果を示している。まず実効性能をみると、リニアに性能が上がっており、MHD コードの場合と同様に良い性能上昇を示した。また、やはり Intel 製コンパイラが日立製コンパイラよりも良い性能を出しており、1024core において、日立製コンパイラで 1,149.3GFlops (実行効率 12%)、Intel 製コンパイラで 1,265,7GFlops (同 13%) を達成した。これは MHD コードの結果とほとんど変わらない性能である。また、並列化効率をみると、これも MHD コードの結果と同様に日立製コンパイラの方が Intel 製コンパイラよりも良い効率を出す。さらに、Vlasov コードの場合、並列化効率の下降が MHD 比べて著しく少ない。日立製コンパイラであれば、1024core 時に 85%の並列化効率を出しており、Intel 製コンパイラでも 83%の並列化効率を出している。これは、MHD コードの結果と比べて、20%程度良い結果である。つまり、MHD コードは非並列（単体）での性能が高いが、並列化により性能が落ちている。一方で Vlasov コードは非並列時の性能は MHD より低いが並列化効率が良く、性能の劣化が少ないため、最終的に MHD コードと同じ程度の実効性能を達成していると言える。



第 16 図:並列数に対する Vlasov コードの実効性能と並列化効率[11]。

左図が実効性能を表し、横軸が並列化数（使用 core 数）、縦が GFlops 値を表す。右図は並列化効率を表し、横軸が並列化数（使用 core 数）で縦軸がスケーラビリティを示す。赤色の実線が日立コンパイラの結果を示し、青色の破線が Intel コンパイラの結果を表す。Vlasov コードは MHD コードに比べて良い並列化効率を持っている。

４．まとめと今後の展望

H20 年度では、HA8000 を使い土星磁気圏シミュレーションを始めるに当たって、我々のコードの性能評価を行った。性能評価は MHD コードと Vlasov コードの二つを用いて行った。MHD コード、Vlasov コードをあわせて、実効性能では Intel 製コンパイラの方が性能が出るが、並列化効率は日立製コンパイラの方が良いという結果になった。MHD コードでは、2 次元領域分割を用いた場合が最も性能が良く、約 1.3TFlops（実行効率 14%）を達成し、Vlasov コードでは 1.27TFlops（同 13%）の性能を達成した。両コードとも 16core 以上は並列化することによる性能劣化は見えなかったため、このまま 1024 並列を越えてもリニアに性能が出ると考えられる。

別のシステムでは10,000並列でのベンチマークをとり、性能が出ることが実証されている。ただし、T2Kオープンスパコンのような汎用PCクラスタ型ではない、スパコンシステムの結果であり、実際にHA8000などで大規模並列を調べる必要性は十分ある。

MHDコードの問題点として、並列化を行うと性能が6割近くに下がってしまうことが今回わかり、そこを改善することで実行効率20%近く、最低でも15%を越える性能を目指す。それらの最適化が終わり次第、高精細の土星磁気圏シミュレーションを始める。目標としては $0.1R_S$ の格子幅を用いたシミュレーションを行い、現在問題になっている土星磁気圏境界面上の乱れた対流構造をうまく取り扱えているかを調べる。また、次（々）世代磁気圏コードのVlasovコードでは、単coreでの性能上昇を考える。並列化効率は良いので、単体での性能を上げる事で、並列化後の性能を上げることが可能である。こちらでも15%以上の実行効率を目指していく。

参 考 文 献

[1] Margaret G. Kivelson, Christopher T. Russell 編『Introduction to space physics』, Cambridge University Press, 1995.

[2] Ogino, T., R. J. Walker, and M. G. Kivelson, A global magnetohydrodynamic simulation of the Jovian magnetosphere, *J. Geophys. Res., 103*, 225, 1998.

[3] Fukazawa, K., T. Ogino, and R.J. Walker, “Dynamics of the Jovian magnetosphere for northward interplanetary magnetic field (IMF)”, Geophys. Res. Lett., 32, doi:10.1029/2004GL021392, 2005.

[4] Fukazawa, K., T. Ogino, and R.J. Walker, ″The Configuration and Dynamics of the Jovian Magnetosphere″, J. Geophys. Res., 111, A10207, 2006.

[5] Fukazawa, K., T. Ogino, and R.J. Walker, ″Magnetospheric Convection at Saturn as a Function of IMF B_Z″, Geophys. Res. Lett., 34, L01105, 2007a.

[6] Fukazawa, K., T. Ogino, and R.J. Walker, ″Vortex-associated reconnection for northward IMF in the Kronian magnetosphere″, Geophys. Res. Lett., 34, L23201, 2007b.

[7] Khurana, K. K., M. G. Kivelson, V. M. Vasyliunas, N. Krupp, J. Woch, A. Lagg, B. H. Mauk and W. S. Kurth, The Configuration of Jupiter's Magnetosphere, In:Bagenal, F., T. Dowling and W. McKinnon (Ed.), Jupiter, Cambridge University Press, New York, 2004.

[8] 様々なスパコンにおけるMHDコードの実効性能.
http://center.stelab.nagoya-u.ac.jp/web1/simulation/hpfja/comput04.html

[9] R. O. Dendy,『Plasma Dynamics』, Oxford University Press, 1990.

[10] T. Ogino, R. J. Walker, M. Ashour-Abdalla, A global magnetohydrodynamic simulation of the magnetopause when the interplanetary magnetic field is northward, IEEE Trans. Plasma Sci.20, 817.828, 1992.

[11] Fukazawa, K., T. Umeda and, T. Ogino, Performance measurement of electromagnetic fluid codes for space plasma on the T2K open supercomputer system, submitted to

parallel computing.

[12] T. Umeda, K. Togano, T. Ogino, Two-dimensional full-electromagnetic Vlasov code with conservative scheme and its application to magnetic reconnection 180, 365-374, 2009.

海洋大循環のマルチスケール連結階層モデリング

羽角博康

東京大学気候システム研究センター

黒木聖夫

独立行政法人海洋研究開発機構

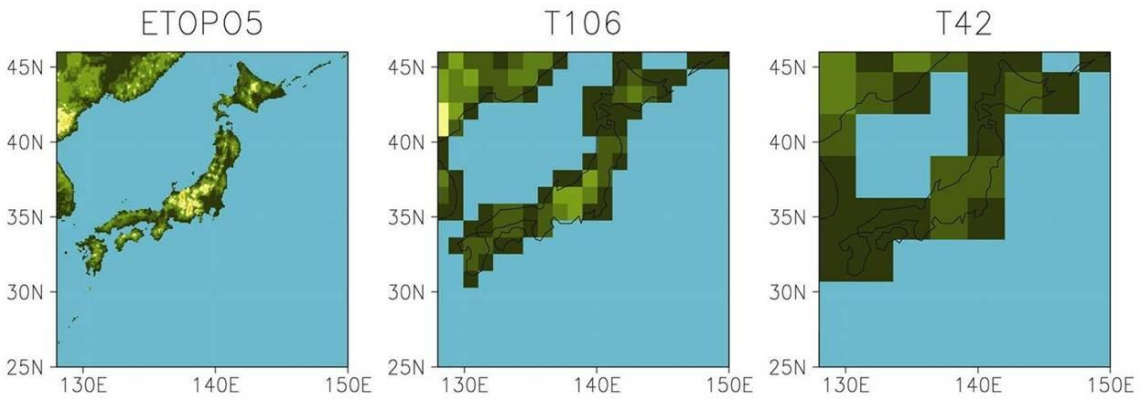
１．はじめに：気候変動予測シミュレーションの現状

気候変動に関する政府間パネル（IPCC: Intergovernmental Panel on Climate Change）は，地球温暖化問題に対する政策決定への判断材料に供することを目的に，気候変動の現状と将来予測に関する科学的知見を集約する評価報告書を，1990 年から数年ごとに出版している．2007 年に出版された第 4 次評価報告書では，気候の温暖化が現実に進行していること，そしてその原因が人間活動による温室効果気体（特に二酸化炭素）の排出であることが，これまでにない強い確信をもって述べられている（IPCC, 2007）.

地球温暖化は海洋の状態とも密接に関わる．地球温暖化の中で生じるであろう海洋の変化の代表的なものとしては，陸上の氷床や氷河の融解と海水熱膨張を主な原因とする海水位上昇，人為起源二酸化炭素の吸収に伴う海水の酸性化，および海氷の減少や深層循環の停滞などを挙げることができる．これらのうちいくつかの側面については既に顕著な影響が認められ，マスメディア等でも取り上げられて社会的関心事となっている．また，海洋は気候の温暖化に対して単に受動的に変化するのではなく，海洋の変化が温暖化の進行そのものに大きな影響を及ぼす. 例えば, 海氷の減少は太陽光の反射量の減少を通して温暖化を加速する働きを持ち, また, 深層循環の停滞は温暖化の大小の地理的分布を大きく変化させ得る．そして，温暖化以前の問題として，そもそも気候の状態の決定において，海洋は本質的な役割を果たしている.

地球温暖化をはじめとする気候の将来変動を予測するための唯一の手段は数値シミュレーションである．我々が気候の状態として直接的に意識するものは地表付近の気温・降水量・風などであるが，それを含めた大気の物理的状態のシミュレーションは，日々の天気予報という場面でも行われている．大気の物理的状態には，積雪や植生分布といった陸上の状態や海面水温が大きな影響を及ぼすが，日々の天気を予報する範囲ではそれらの要素に大きな変化が生じることを考慮する必要はなく，観測した状態が与えられるのが通常である．しかし，地球温暖化の予測においてはそれらの要素が大規模に変化することを同時に予測することが必要とされる. その中でも特に，大気と比べて大量の熱を蓄え，流れに伴ってその熱を輸送する海洋の状態を正しく予測することの重要性は高い.

地球温暖化の予測シミュレーションは世界各国で行われており, IPCC 第 4 次評価報告書は 20 を越える研究機関が行った予測結果に基づいている．第 1 次評価報告書からおよそ 20 年の間, シミュレーションに用いられる数値気候モデルの開発は各研究機関で継続的に進められるとともにその比較検討の国際的プロジェクトも組織され，計算機の性能向上を背景とした高解像度化とも相俟って，シミュレーションの信頼性向上が図られてきた．それに伴い，予測に求められる内容も，初期には全地球的な気温上昇の様相が中心であったが，近年は地域的な現象へと



第 1 図：水平格子サイズによる地形表現の違い.

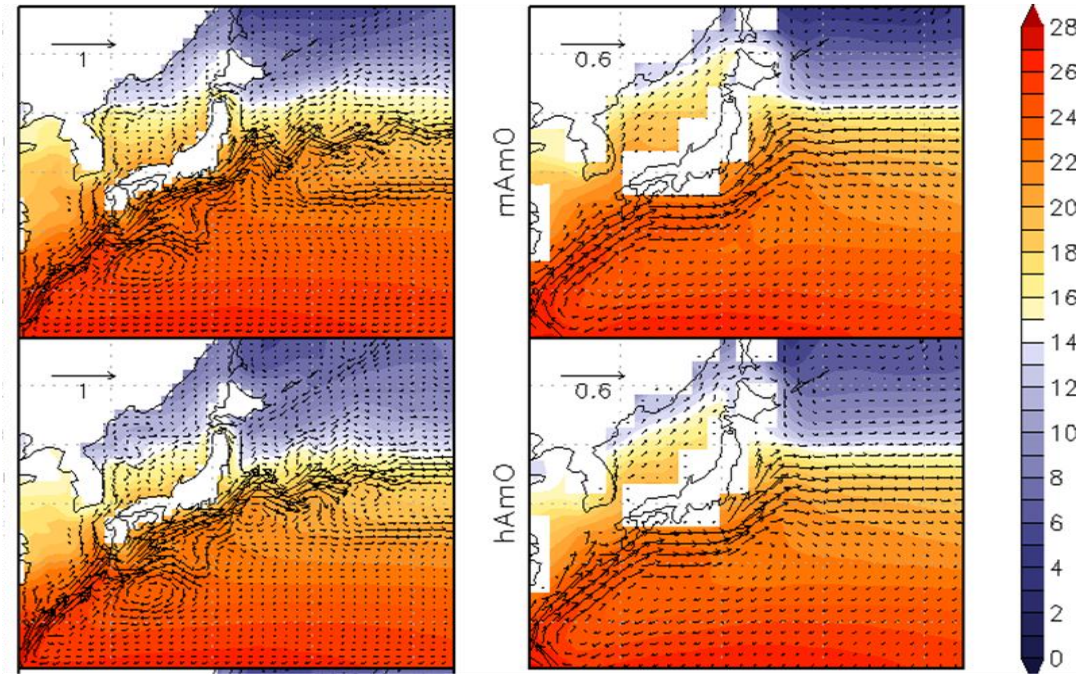
左から順に水平格子サイズ 10 km，100 km，300 km の場合の，日本付近の地形表現.

シフトしてきた．豪雨や旱魃等の極端現象と呼ばれるものの強度や頻度に関することがその代表である．そうした予測は，各国・地域における具体的な影響緩和策や適応策を模索する上では欠かせないものである.

筆者自身も，東京大学気候システム研究センター・国立環境研究所・海洋研究開発機構からなる研究コンソーシアムの一員として,地球温暖化予測シミュレーションに携わっている.我々のグループは,地球シミュレータの利用を背景として,大気の水平格子サイズがおよそ 100 km，海洋の水平格子サイズがおよそ 20 km という，IPCC 第 4 次評価報告書に予測結果を提出した中でも最高解像度のシミュレーションを行った．なお，日々の天気予報や，短期的・局地的な海況予測シミュレーションにおいては，これよりも格段に高い解像度が採用されるのが通常である．しかし，地球温暖化予測においては，過去 100 年以上の再現と複数の将来シナリオ[1]に基づく予測が要求されるとともに，それに付随する各種実験やスピンアップと呼ばれる初期状態作成計算が必要となり，極めて長期間に渡るシミュレーションを行うことになる．与えられた計算資源で必要な期間のシミュレーションを遂行できることが，解像度選択における最大の制約となる．実際のところ，IPCC 第 4 次評価報告書における他の予測シミュレーションでは，典型的な水平格子サイズは，大気については 200～300 km，海洋については 100 km 程度である．この差がどの程度であるかは,第 1 図をご覧いただけば一目瞭然であろう.水平格子サイズ 100 km というのは，例えば日本をそれなりに表現するために最低限必要な解像度と言える.

海洋のシミュレーションにおいて，水平格子サイズ 100 km と 20 km の間には，地形表現の問題だけにとどまらない本質的な違いがある．海洋という非線型系では，現象の再現性は解像度とともに連続的に向上するわけではなく，ある特徴的な空間スケールを表現するかどうかに依存して質的な違いが現れる．20 km という水平スケールは，中緯度海洋における変形半径と呼ばれる空間スケールに相当し，例えば中規模渦と呼ばれる現象が表現されるかどうかがこのスケールで分かれる．海洋の中規模渦とは，基礎的な力学が似ている大気で言えば温帯の移動性高・低気圧に相当するものであり，大気の場合と同様に南北方向の熱輸送に大きな役割を果た

[1] 地球温暖化予測シミュレーションでは，大気中の温室効果気体濃度の将来的な値が必要となる．そのためには，まず将来の産業活動に関してシナリオ（経済成長優先，新エネルギーへの移行，など）を設定し，それに応じた化石燃料消費量（および土地利用状態変化など）を予測し，さらにモデルを通してそれを大気中温室効果気体濃度に変換する.



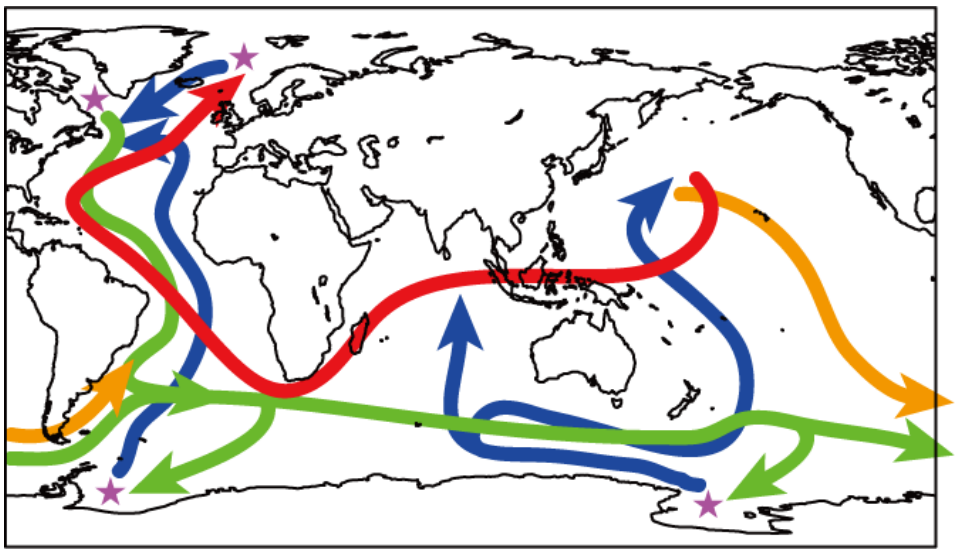
第 2 図: 水平格子サイズによる黒潮の再現性の違い.

大気水平格子(上)約 300 km, (下)約 100 km, および海洋水平格子(左)約 20 km, (右)約 100 km の場合の, 日本付近における長時間平均の海面付近流速 (矢印: m/s) と水温 (色). 同じ長さの矢印が表す流速が左右で大きく異なることに注意.

す.

また, 変形半径の解像の有無は, 海洋の西岸境界流の表現に対しても顕著な影響を及ぼす. 太平洋・大西洋・インド洋という大洋はすべて, その西端に南北方向の強い海流を持っており, それらを総称して西岸境界流と呼ぶ. 北太平洋であれば日本の南岸を通る黒潮と北海道から本州東岸を通る親潮がそれにあたる. こうした西岸境界流は, その強い流れによって大量の熱を輸送して気候の状態に大きな影響を与え, また, 沿岸海況や水産環境の面からも重要な働きを持つ. 第 2 図は水平格子 100 km の場合と 20 km の場合で黒潮の再現性にどのような差が現れるかを示したものである. 水平 20 km 格子の場合, 黒潮の幅, 日本南岸の流路 (南岸にはりついて直進する流路と大きく蛇行する流路の 2 種類が長期間平均によって同時に見えている) や本州からの離岸位置, さらにその先で 2 回定常的な蛇行をもって東進する流路など, 平均的な構造は良く再現されている. 一方, 水平 100 km 格子の場合, 日本南岸に沿う比較的強い流れは存在するが, 幅が広くなるとともに流速・流量ともに小さく, また本州から離岸する位置も大幅に北にずれている.

地球温暖化の中で黒潮をはじめとした西岸境界流がどのように変化するのかは重要な予測・研究対象であるが, この水平 100 km 格子の場合のように, そもそも元々の平均的な状態がうまく再現されていないようなシミュレーションでは, 当然のことながらその将来変化を信頼性高く予測することはできない. 計算機能力の発達を背景として, やっとそのような予測が現実的になってきたところである (Sakamoto et al., 2005; Sakamoto and Hasumi, 2008). そして, より信頼性の高い地球温暖化予測とその影響評価を今後行っていくためには, 西岸境界流のみならず, 様々な中小規模の海洋現象をこれまで以上に適切に扱う必要がある.



第 3 図：海洋大循環の概略図.

ただし，黒潮のような各大洋内での水平的な循環は無視しており，大洋間のつながりに特化している．矢印は赤・橙・緑・青の順からそれぞれ，表層（海面付近），中層（深さ 1000 m 程度），深層（深さ 3000 m 程度），底層（深さ 5000 m 程度）の流れを，星印は主な海水の沈降（深層水形成）領域を示す.

２．本研究の目的：海洋の連結階層モデリングの必要性と技術課題

黒潮の例に代表されるように，地球温暖化予測シミュレーションの精度や信頼性は着実に高まってきているが，残念ながら現状ではまだ十分とは言えず，その中でも海洋の解像度不足に起因する部分が少なからず存在する．同じ黒潮に関して言えば，平均的な構造は水平 20 km 格子程度でよく再現されるものの，短期的な変動性についてはもう一段階高い解像度を適用しなければ十分に再現することができない．そして，地球温暖化に伴う黒潮の変化が自然環境および人間社会に対してどのような影響を及ぼすのかを評価する上では，その変動性がどのように変わるのかを知ることまでが重要となる．これは喩えて言うなら，地球温暖化の結果として台風の頻度や強度がどのように変わるのかを知ることが防災上重要であるのと同様である.

黒潮は海洋の表層に存在する海流の代表例であるが，海洋の深層にも流れは存在する．それは我々の生活環境からは非常に遠くに位置し，しかもその流れの速さは黒潮に比べれば何十分の一に過ぎないのであるが，実は根本的なところで我々の生活環境を大きく左右している．深層海洋の流れは深層だけに閉じておらず，表層海洋の流れとリンクしている．海水の密度は温度と塩分に依存し，海面付近で冷却されて低温化（もしくは蒸発等により高塩分化）した海水は，高密度のために深層へ沈む傾向が強くなる．現在の海洋の状態において，深さ数千 m の深海に存在する水はいたるところで 0℃近くの低温であり，これは高緯度の海面付近にある低温水が沈降して深層を占めていることを示す.

第 3 図はそうした海洋の深層と表層をつなぐ循環の概略を模式的に示したものであるが，その大きな特徴は，海面付近から深層海洋への沈降が北大西洋高緯度と南極周囲の極めて限られた場所でのみ生じていることである．深層水形成と呼ばれるこの沈降過程は水平 1 km 程度のスケールで生じる対流現象に端を発している．その意味では，極めて限られた領域における微小規模の現象が全海洋規模の循環をコントロールしている．そしてこの循環は，地球上の熱を大

規模に再配分する．その働きがあればこそ，例えばヨーロッパ北部は 70 度を越えるような高緯度にも関わらず人が居住できる環境にある．氷期等の過去に生じた大規模な気候変動はこの海洋大循環の変動と大きな関わりがあることが知られており，地球温暖化においてもこの循環が大幅に弱まるであろうことが予測のひとつの焦点になっている．しかして，当然と言うべきか，水平 1 km 程度スケールのプロセスを含む深層水形成過程は水平 20 km 格子程度では適切に表現されているとは言えず，この点に関しても海洋現象の表現不足のために地球温暖化の将来予測シミュレーションに不確実性が残っている．

上述の問題の単純な解決方法は，もちろん解像度を十分に高めることである．しかし，例えば全海洋を水平1 km格子で表現した上で意味のある地球温暖化予測シミュレーションを行うことは，現在開発中の次世代スーパーコンピュータを用いてもほぼ無理である．その一方で，地球温暖化に大きな影響を及ぼす，あるいは地球温暖化から大きな影響を受けるという意味で，海洋の中で水平10 km程度以下の微小規模現象が重要になる場所は，幸いにして限られている．したがって，現実的な解決策としては，それらの領域および現象を選択的に高解像度で表現したシミュレーションと比較的低解像度での全海洋シミュレーションとの連結階層化が考えられる．

こうした連結階層化は，もちろん新しい概念ではなく，海洋の数値モデリングにおいても既に様々なモデル開発および適用例が存在する．とはいえ，そうした過去の試みのほぼ全ては一方向のダウンスケーリングに限られており，小規模スケールの現象が海洋の大規模構造を左右するプロセスを扱う実例はほとんど存在しない．その理由は，ひとつにはやはり計算資源の制約にある．空間的に小規模スケールの現象は，それ自身が持つ特徴的な時間スケールが短い．したがって，小規模スケール現象をだけを目的とする場合には，空間解像度を高めなければならない代わりに，長期間の時間積分は必要でなくなる．ダウンスケーリングの連結階層シミュレーションを行う場合，対象とする高解像度領域の周囲での境界条件を与える目的で，より広い領域のシミュレーションが行われることになるが，領域は広いものの必要とされる時間スケールは短いため，高解像度対象領域に比べて計算負荷がさほど高くないのが普通である．一方，双方向の連結を行う場合に目的とされるのは，高解像度で表現する小規模スケールの現象が大規模スケールの現象にどのような影響を及ぼすかである．大規模スケールの現象の持つ特徴的な時間スケールは長く，この目的に対しては，その長い時間のシミュレーションを高計算負荷である高解像度領域に対して実行しなければならない．その実現のために大きな計算資源が必要であることはもちろんだが，利用できる計算資源の制約を前提として，高並列環境で高速動作させるための努力も不可欠である．双方向の連結階層シミュレーションがあまり行われてきていないもうひとつの理由として，モデル開発の困難さも挙げることができる．各階層モデル間での物理量の連続性や保存性の厳密さなどという重要な制約に関して，すべてを同時に満足させるような標準的なスキームは確立されていないのが現状である．さらには，そうした数値モデルを高並列環境で効率よく動作させる枠組みを用意する必要もある．

以上のことを背景として，我々は双方向に連結階層化した海洋大循環モデルの開発とその高並列チューニングを重要課題として研究を行っている．本共同研究では，その高並列チューニングの部分に関して，HA8000 の計算資源の提供およびプログラム開発に関する助言を受け，研究開発を進めてきた．

３．海洋大循環モデルとその連結階層化

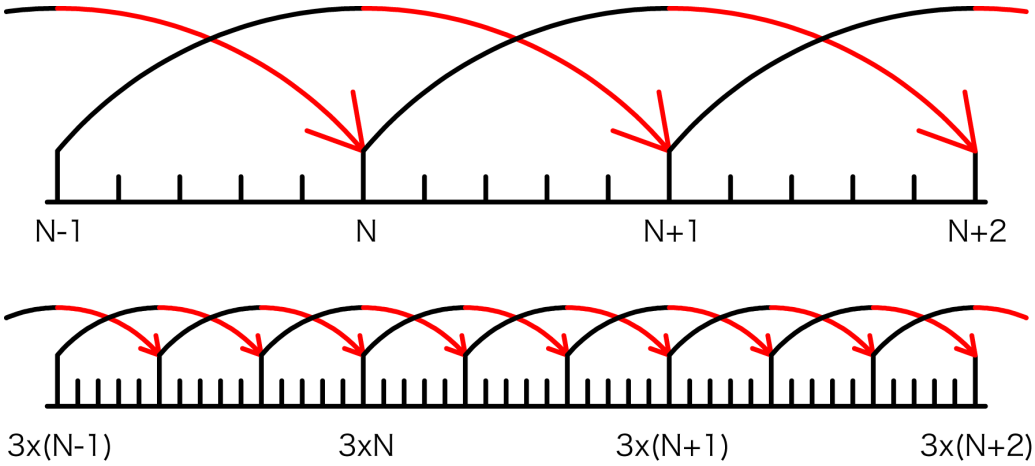
（１）海洋大循環モデル

海洋の循環を記述する方程式系は回転系の Navier-Stokes 方程式（線型粘性流体の運動方程式）と熱・溶存物質（塩分）の輸送・拡散方程式を基礎としている．しかし，当然のことながら分子粘性・拡散を直接表現する解像度を海洋という巨大系に対して適用できるわけはない．全海洋規模の循環を扱う場合には，対象とする運動の時空間スケールに基づいたいくつかの近似を導入して方程式系を簡略化するとともに，解像されないスケールの現象の影響はパラメータ化によって表現する．これを離散化した数値モデルは，海洋大循環モデルと呼ばれる．離散化は一般に構造格子上の差分法により行われる．本課題では，東京大学気候システム研究センターで開発されている海洋大循環モデルである COCO（CCSR Ocean Component Model）をベースとし，その連結階層化および並列計算の効率化を行った．本小節の残りの部分では，COCO の仕様のうち，後述する連結階層化に関わる部分について簡単に解説する．方程式系や差分化手法に関する詳細については Hasumi（2006）などを参考にされたい．

COCO では基礎方程式系を，水平方向には球面上の一般曲線直交座標系上で，鉛直方向には深度座標系上で定式化している．地球上の問題を扱うにあたっては，水平座標として地理座標（緯度・経度）を用いるのが最も簡便であるが，地理座標系で全海洋を扱うと北極海に経度座標が集中する特異点が存在し，数値計算上の困難をもたらす．したがって，全海洋を扱う場合には，特異点を陸上に置くような座標系を選ぶことが都合がよい．一方，北極海以外の限定された海域に対するシミュレーションを行う場合には，地理座標を用いることが最も一般的である．ただし，その場合でも，ターゲットとする海域を連結階層化によらずに集中的に高解像度で表す目的などで，地理座標とは大きく異なる座標系が採用されることが少なからずある．その一方で，座標系の選択では特異点を陸上に置くことが前提となるため，この方法だけでは望むような格子系を得ることが必ずしも可能ではない．特に，具体例を後述するように，大きな海域の中で比較的広い領域を高解像度化する目的にはこの方法は向かないことが多く，そのような場合には連結階層化の方が効率良くターゲット海域を高解像度化することが可能となる．

モデル変数の格子配置について，COCO では Arakawa-B と呼ばれるタイプの staggered 格子を採用している（Mesinger and Arakawa, 1976）．すなわち，水平面上で，格子の中央に温度や塩分（あわせてトレーサと呼ぶ）が，格子の頂点で水平流速が定義される．詳しくは次小節を参照されたい．一方，海洋大循環モデルでは鉛直流速は予報量ではなく診断量であり，水平流速を元に，非圧縮流体に対する連続の式から求められる．

運動方程式を数値的に解く際，ほぼ全ての海洋大循環モデルにおいて共通して，モード分離という手法が用いられる．鉛直に積分した水平流速の収束はその場での海面昇降に対応し，この海面昇降に重力が復元力として働くことで，海面には重力波という波動が存在する．この海面重力波の典型的速度は 100 m/s 程度であり，海洋中の流速や海面昇降と関係しない波動の伝播速度と比べて 2 桁程度大きい．海面重力波は海洋大循環モデルがシミュレート対象とする現象に対してあまり重要な意味をもたないため，運動方程式全体を海面重力波による CFL 条件によって制約される時間ステップを用いて解くことは，甚だ効率が悪い．一方，鉛直積分した水平流速と海面昇降を記述する方程式は単純な形をしており，それ自体を解くのに必要とされる計算負荷は比較的小さい．そこで，運動方程式の水平流速を，鉛直平均成分（順圧成分と呼ぶ）とそこからのずれ（傾圧成分と呼ぶ）に分解し，それぞれに対する予報方程式を別個に解くこ



第 4 図: モード分離における時間積分の進行の模式図.
(上)連結階層しないモデル（および連結階層した場合の外側モデル）における進行.（下)連結階層した場合の内側モデルのおける進行. 長い縦線は傾圧モードの，短い縦線は順圧モードの時間ステップを表す.

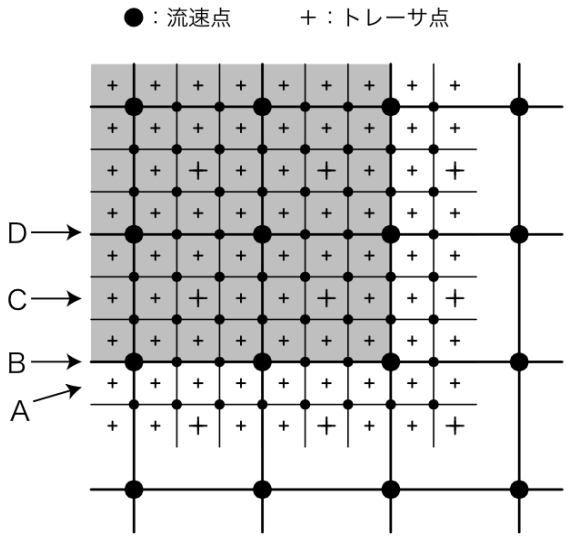
とを行う[2]. この際，時間積分に関しては，傾圧成分を時間 1 ステップ進める間に，順圧成分はその $1/N_S$ の時間ステップを用いて N_S 回の刻み前進を行うという，タイムスプリット手法を用いる[3]. 例えば COCO の場合，標準的には時間積分スキームとして leap-frog を採用しているが，傾圧成分に対して時間が第 N-1 ステップから第 N+1 ステップに進行するとき，順圧成分では $2N_S$ 回の刻み積分を行い（第 4 図参照），両者の最終結果を足し合わせたものが第 N+1 ステップの水平流速となる. なお，順圧成分の刻み積分における時間積分スキームには，Euler-backward を採用している.

方程式系の時間積分は，水平流速傾圧成分，水平流速順圧成分，トレーサの順で，それぞれ第 N-1 ステップから第 N+1 ステップへ進行する. トレーサの輸送計算では，第 N-1 ステップから第 N+1 ステップへの進行において，第 N ステップの水平流速が必要になる. この水平流速の傾圧成分については第 N ステップの値そのものでよいが，順圧成分については，連続の式との整合性のため，第 N-1 ステップから第 N+1 ステップまでの平均を用いる必要がある. したがって，運動方程式が表現する流速とトレーサ輸送に用いられる流速は厳密には一致しないことになる. また，順圧成分自身は leap-frog によらないものの，傾圧成分の leap-frog の時間進行の重なりに対応して，順圧成分の計算は重複して行われる.

COCO は MPI を用いた並列化コーディングが施されている. 並列化は水平方向の 2 次元領域分割によって行われ，すべてのプロセスで同一のプログラムを用いる SPMD の手法を用いている. 分割された各領域の計算では，周囲の 2 格子分の幅のデータが必要とされ，その分を通信によって隣接する領域と交換する. モデル計算における通信はこのタイプのものに限定され，初期条件・境界条件の読み込みや結果出力の I/O における gather，scatter を除き，全体通信は存在しない.

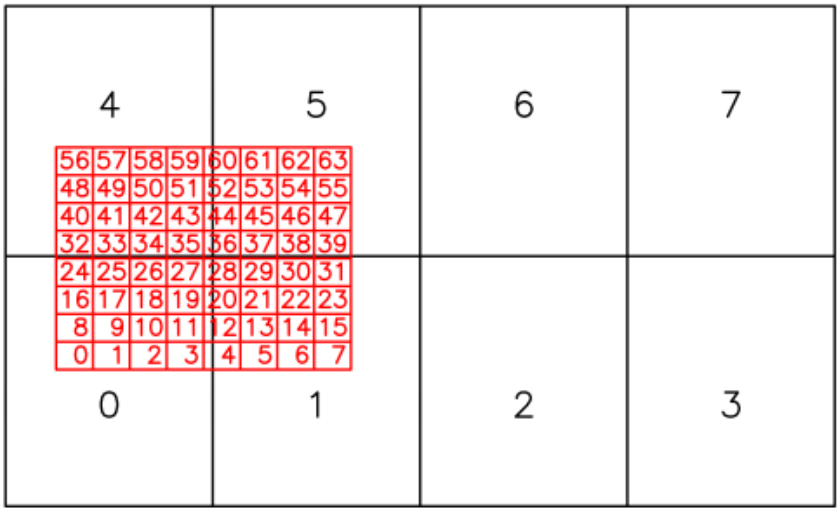
（２）連結階層化

[2] ただし，海面昇降が 0 でない場合には厳密なモード分離は不可能であり，近似的を伴う.
[3] N_S の典型的な値は数十.



第 5 図：内側モデルの境界付近における nested-grid モデルの格子配置.

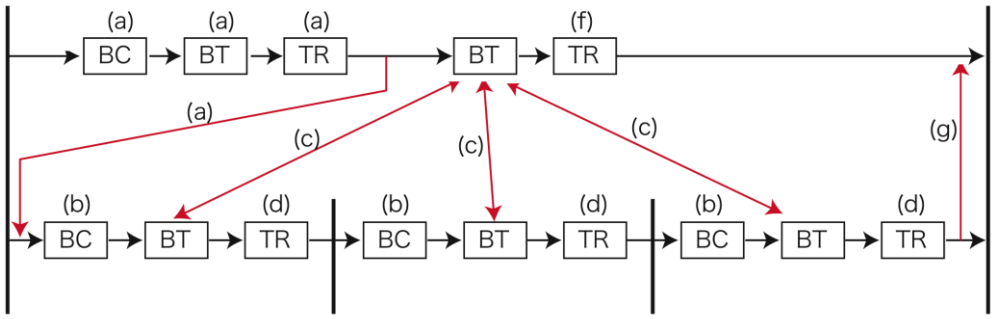
外側モデルの 1 格子を内側モデルで 3×3 分割した場合の例．影のついた領域が内側モデルを表す．●，＋ともに，大きい印は外側モデルの変数（およびそこに重なって存在する内側モデルの変数），小さい印は内側モデルの変数の配置を表す．A，B はトレーサと流速それぞれについて，外側モデルから内側モデルへ境界条件を与える位置．C，D は内側モデルから外側モデルへ境界条件を与える位置．



第 6 図：外側モデルと内側モデルの領域分割および通信イメージ.

外側モデル 8 プロセス（黒），内側モデル 64 プロセス（赤）で並列計算する場合の例．数字はプロセスのランクを示す．外側モデルと内側モデルの間では，重なりがある部分の間のみで通信が発生する．

ここで採用する連結階層化の手法は，高解像度化領域を固定した双方向 nested-grid である．モデルの全海域（外側モデルと呼ぶ）の中に矩形領域（内側モデルと呼ぶ）を設定し，鉛直格子は両者で共通，水平格子については外側モデルの 1 グリッドを 5×5 などの奇数個分割とする．境界付近における格子配置を模式的に示したものが第 5 図である．内側モデルの境界に位置する流速点からその外側 2 格子分のトレーサ点までの領域に，外側モデルの格子から線型補間し



第 7 図: 外側モデルと内側モデルの時間積分進行および通信の模式図.

外側モデルの時間 1 ステップを内側モデルで 3 分割している例を示す. 上段は外側モデル, 下段は内側モデルの計算過程であり, 上下段間の矢印は通信の方向を示す. BC, BT, TR はそれぞれ傾圧成分, 順圧成分, トレーサの計算を表し, (a)～(g)は本文中の各手続きを示す.

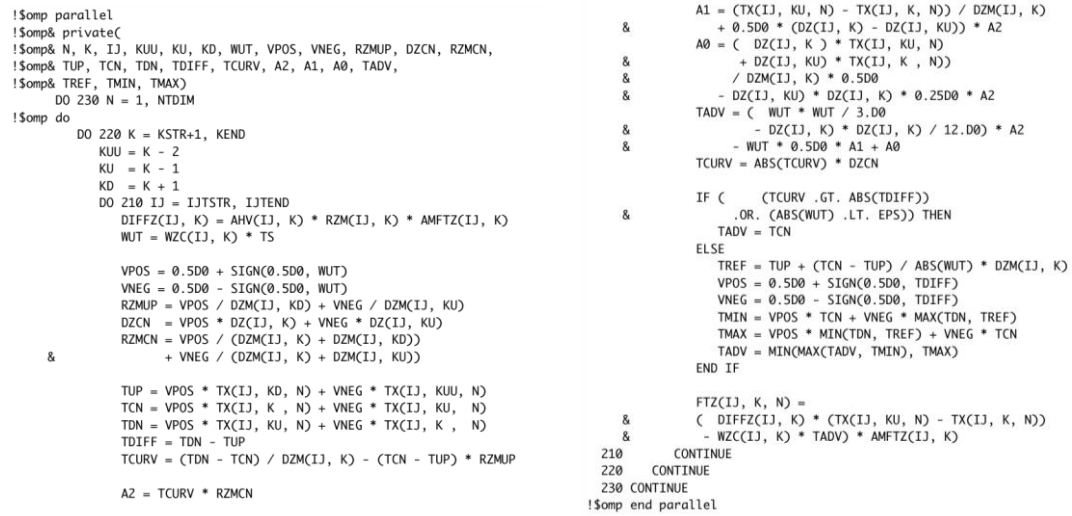
た流速やトレーサを与え, 内側モデルの境界条件としている. また, 内側モデルの領域内については, 内側モデルの変数を空間平均したもので外側モデルの変数を置き換えることで, 双方向ネストを行っている. なお, 内側モデルの海岸・海底地形については, 境界での整合性を保つために, 内側モデル境界から外側 2 格子分の領域で外側モデルと一致させている.

連結階層化モデルの実行においては, MPMD 方式を採用している. 第 6 図に示すように, 外側モデルと内側モデルはそれぞれが水平 2 次元で領域分割されている. 各モデルの内部では, それぞれのモデル用に作成されたコミュニケータを用いて通信が行われる. 一方, 外側モデルと内側モデルの間の通信はコミュニケータ MPI_COMM_WORLD を用いて行われるが, 大規模並列化に対応するために, 領域に重なりが存在するプロセス間でのみ通信が行われるようにコーディングされている. 第 6 図の例では, 例えば外側モデルのランク 4 のプロセスは, 内側モデルのランク 32-36, 40-44, 48-52, 56-60 のプロセスのみとの間で通信が行われる.

外側モデルと内側モデルのデータ交換は第 7 図に示される手順で行われる. すなわち

(a) 外側モデルの傾圧成分, 順圧成分, トレーサを更新し, 境界条件となる傾圧成分とトレーサのデータを内側モデルに送る. これを時間方向に線型補間して, 内側モデルの境界条件を得る.
(b) 内側モデルの傾圧成分を更新する.
(c) 内側モデルと外側モデルの順圧成分を, 内側・外側モデルで通信しながら計算する.
(d) 内側モデルのトレーサを更新する.
(e) (b)～(d)の手続きを, 外側モデルの第 N+1 ステップの時刻になるまで繰り返す.
(f) (c)で得た順圧成分を用いて, 外側モデルのトレーサを再計算する.
(g) 内側モデル領域の順圧成分とトレーサの空間平均値を外側モデルに送信し, 外側モデルの変数を置き換える.

上記の(c)の過程については, 順圧成分計算で時刻が重複する部分があるため (第 4 図参照), 取り扱いに工夫が必要である. ここでは, 外側モデルと内側モデルの通信は, 常に第 4 図の矢印の赤で示される後半部分で行うようにしている. また, 前半部分 (矢印の黒) の計算については, ひとつ前の順圧成分の後半部分で保存しておいたデータを利用する.



```
!$omp parallel
!$omp& private(
!$omp& N, K, IJ, KUU, KU, KD, WUT, VPOS, VNEG, RZMUP, DZCN, RZMCN,
!$omp& TUP, TCN, TDN, TDIFF, TCURV, A2, A1, A0, TADV,
!$omp& TREF, TMIN, TMAX)
      DO 230 N = 1, NTDIM
!$omp do
         DO 220 K = KSTR+1, KEND
            KUU = K - 2
            KU  = K - 1
            KD  = K + 1
            DO 210 IJ = IJTSTR, IJTEND
               DIFFZ(IJ, K) = AHV(IJ, K) * RZM(IJ, K) * AMFTZ(IJ, K)
               WUT = WZC(IJ, K) * TS

               VPOS = 0.5D0 + SIGN(0.5D0, WUT)
               VNEG = 0.5D0 - SIGN(0.5D0, WUT)
               RZMUP = VPOS / DZM(IJ, KD) + VNEG / DZM(IJ, KU)
               DZCN  = VPOS * DZ(IJ, K) + VNEG * DZ(IJ, KU)
               RZMCN = VPOS / (DZM(IJ, K) + DZM(IJ, KD))
     &               + VNEG / (DZM(IJ, K) + DZM(IJ, KU))

               TUP = VPOS * TX(IJ, KD, N) + VNEG * TX(IJ, KUU, N)
               TCN = VPOS * TX(IJ, K , N) + VNEG * TX(IJ, KU,  N)
               TDN = VPOS * TX(IJ, KU, N) + VNEG * TX(IJ, K ,  N)
               TDIFF = TDN - TUP
               TCURV = (TDN - TCN) / DZM(IJ, K) - (TCN - TUP) * RZMUP

               A2 = TCURV * RZMCN
               A1 = (TX(IJ, KU, N) - TX(IJ, K, N)) / DZM(IJ, K)
     &            + 0.5D0 * (DZ(IJ, K) - DZ(IJ, KU)) * A2
               A0 = (  DZ(IJ, K ) * TX(IJ, KU, N)
     &               + DZ(IJ, KU) * TX(IJ, K , N))
     &              / DZM(IJ, K) * 0.5D0
     &            - DZ(IJ, KU) * DZ(IJ, K) * 0.25D0 * A2
               TADV = (  WUT * WUT / 3.D0
     &                 - DZ(IJ, K) * DZ(IJ, K) / 12.D0) * A2
     &              - WUT * 0.5D0 * A1 + A0
               TCURV = ABS(TCURV) * DZCN

               IF (     (TCURV .GT. ABS(TDIFF))
     &             .OR. (ABS(WUT) .LT. EPS)) THEN
                  TADV = TCN
               ELSE
                  TREF = TUP + (TCN - TUP) / ABS(WUT) * DZM(IJ, K)
                  VPOS = 0.5D0 + SIGN(0.5D0, TDIFF)
                  VNEG = 0.5D0 - SIGN(0.5D0, TDIFF)
                  TMIN = VPOS * TCN + VNEG * MAX(TDN, TREF)
                  TMAX = VPOS * MIN(TDN, TREF) + VNEG * TCN
                  TADV = MIN(MAX(TADV, TMIN), TMAX)
               END IF

               FTZ(IJ, K, N) =
     &         (  DIFFZ(IJ, K) * (TX(IJ, KU, N) - TX(IJ, K, N))
     &          - WZC(IJ, K) * TADV) * AMFTZ(IJ, K)
  210       CONTINUE
  220    CONTINUE
  230 CONTINUE
!$omp end parallel
```

第 8 図：OpenMP 指示行挿入の例.

第 1 表：Flat MPI 並列とハイブリッド並列の実行時間比較（括弧内は相対的速度上昇度）.

PGI fortran コンパイラ

コア数 （コア数÷64）	Flat MPI 実行時間 （速度上昇率）	OpenMP+MPI 実行時間 （速度上昇率）
64　(1)	6333s (1.00)	6965s (1.00)
216 (3.37)	2545s (2.49)	2528s (2.76)
648 (10.125)	1517s (4.18)	1257s (5.54)

Intel fortran コンパイラ

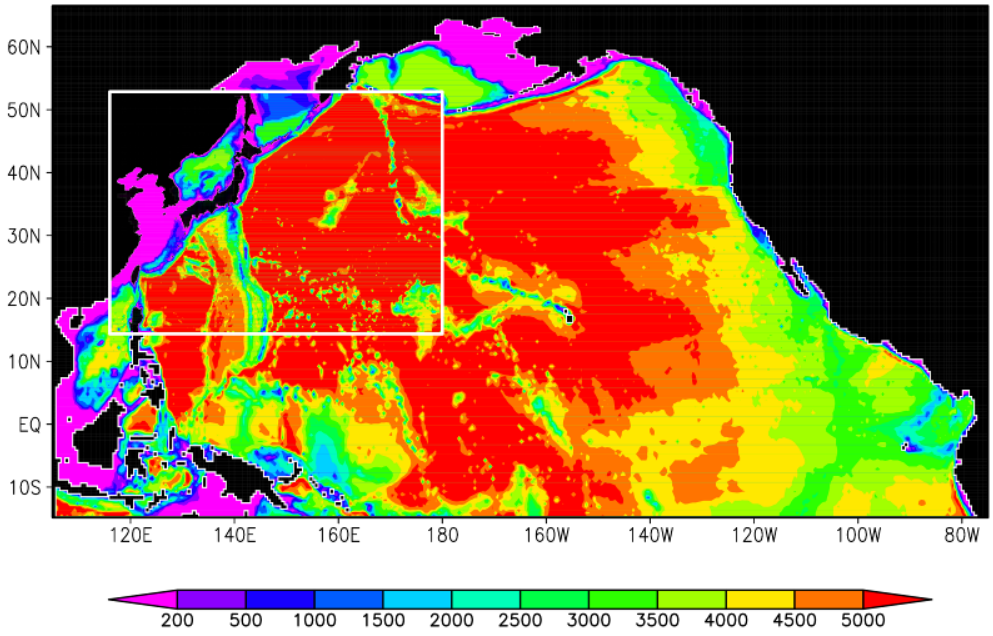
コア数 （コア数÷64）	Flat MPI 実行時間 （速度上昇率）	OpenMP+MPI 実行時間 （速度上昇率）
64　(1)	10417s (1.00)	12477s (1.00)
216 (3.37)	4075s (2.56)	4476s (2.79)
648 (10.125)	2270s (4.59)	2215s (5.63)

Hitachi fortran コンパイラ

コア数 （コア数÷64）	Flat MPI 実行時間 （速度上昇率）	自動並列+MPI 実行時間 （速度上昇率）	OpenMP+MPI 実行時間 （速度上昇率）
64　(1)	6123s (1.00)	6073s (1.00)	6473s (1.00)
216 (3.37)	2590s (2.36)	2243s (2.71)	2345s (2.76)
648 (10.125)	1646s (3.72)	1304s (4.66)	1311s (4.94)

（3）ハイブリッド並列化

本共同研究課題では，MPI による並列に加えて，OpenMP によるスレッド並列を併用したハイブリッド並列を行うよう，プログラムの書き換えとチューニングを行うことを主たる目的としている．第 8 図は OpenMP の指示行挿入例である．この例では，鉛直方向を表す K のループについての並列化が行われている．こうした DO ループを対象にした単純な機械的指示行挿入をまず



第 9 図: モデル領域.

図の全領域が外側モデルに対応し，白枠内が内側モデル．色は海底地形を表す.

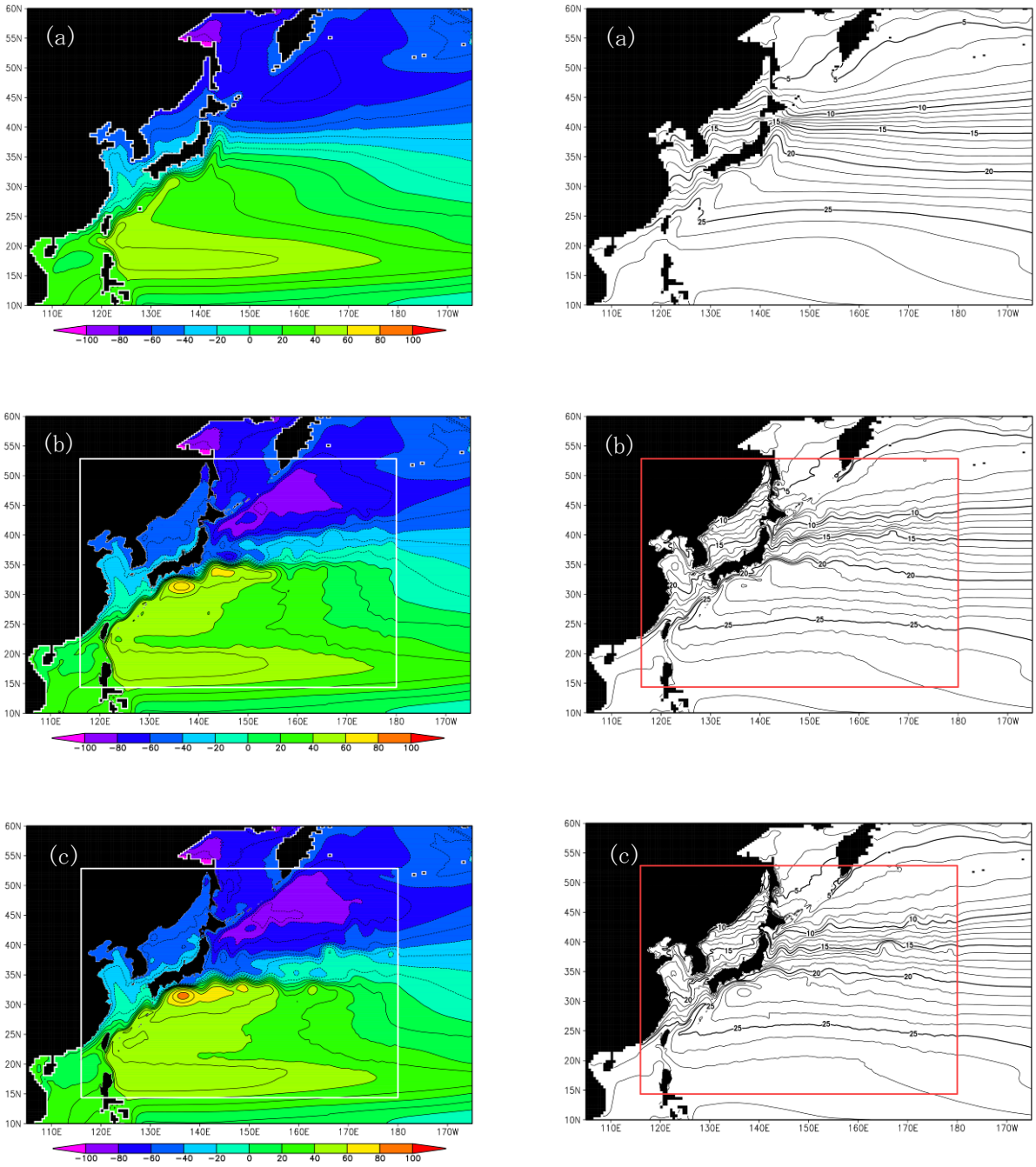
は完了した.

第 1 表に，後述するシミュレーション実行設定の外側モデル（水平 360×216 格子，鉛直 40 格子）において，flat MPI の場合と OpenMP と MPI を併用したハイブリッド並列の実行時間の比較を示す．スレッド並列は HA8000 上で 1 プロセスあたり 4 スレッドとしており，各ソケットに対応する 4 コアをスレッド並列化している．また，fortran コンパイラは HA8000 上で利用できた PGI，Intel，Hitachi すべてについて調べ，Hitachi の場合には独自の自動並列によるハイブリッド並列についても調べた．それぞれ 1 回ずつの測定結果であることに注意が必要であるが，いずれのコンパイラを用いた場合でも，問題サイズを固定した上での使用コア数が増加するにつれ，すなわち 1 コアあたりの格子数が小さくなるにつれ，ハイブリッド並列の方で実行時間･実行速度上昇度ともに flat MPI を上回る性能が実現されるようになっている.ただし，Hitachi コンパイラの場合には，OpenMP を用いたハイブリッド並列よりも，Hitachi コンパイラ独自の自動並列によるハイブリッド並列の方が，コア数増加に対する実行時間減少および実行速度上昇度について良い結果が得られている.

今回行った性能測定は 640 程度の比較的少ないコア数までだが，その範囲でもハイブリッド並列の優位性を示す結果が得られている．さらなる高並列のもとでは，ハイブリッド並列の優位性がさらに高まるものと考えられる.今回の OpenMP 指示行挿入は単純に機械的に行える部分だけを対応したものであり，さらに細かいチューニングを実施することで，さらなる性能向上も期待される．本共同研究課題は平成 21 年度も継続しており，その中でそれを実現していく予定である.

４．連結階層化モデルの実行例

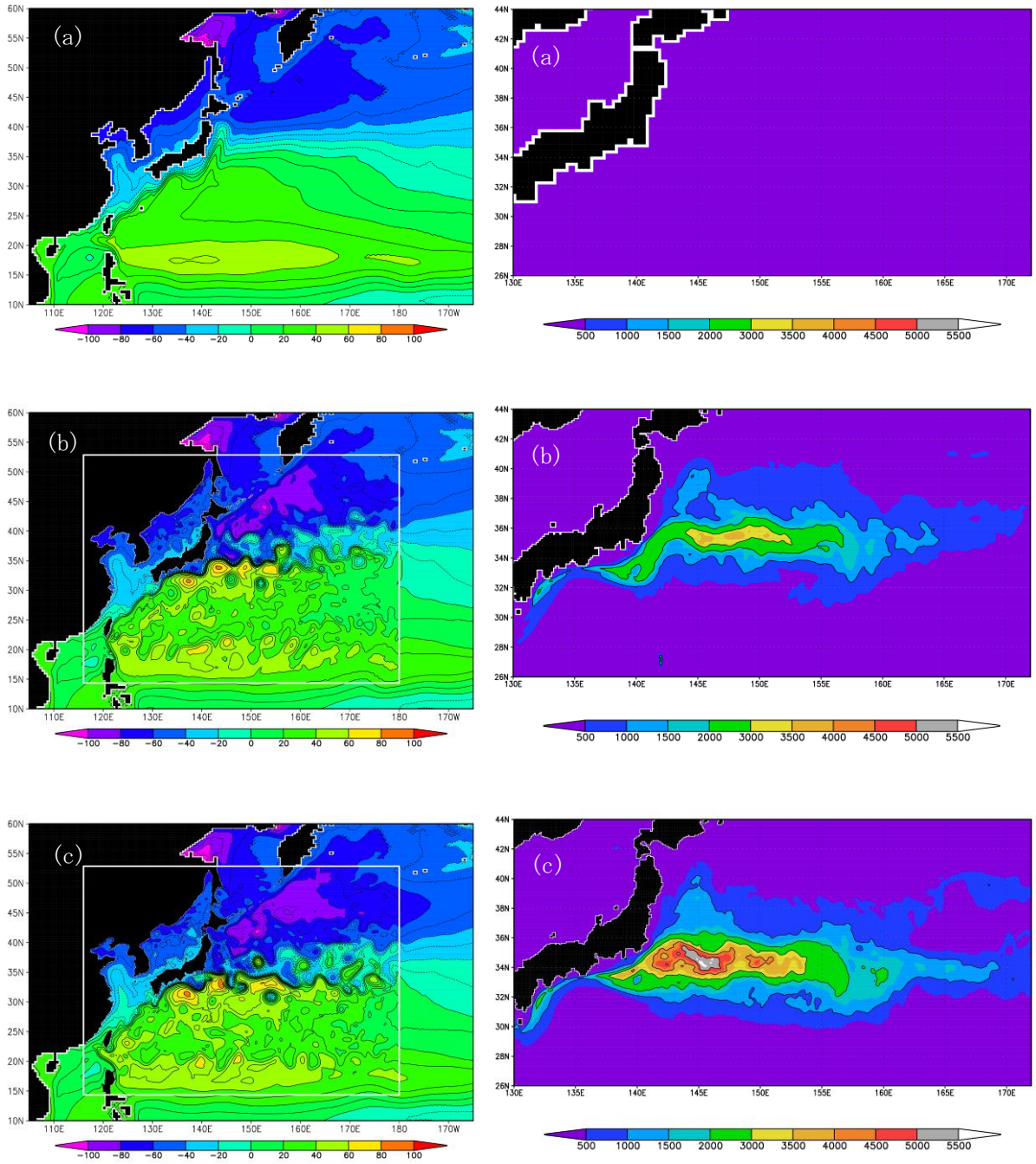
上述の連結階層化モデルを第 9 図に示す領域設定のもとで実行した結果を紹介する．外側モデルは北太平洋全域を含む領域で，格子数は水平 360×216，鉛直 40 である．座標系は通常の



第 10 図：5 年平均(左)海面高度(cm)，および(右)海面水温(℃)．

(a) 全領域を外側モデルの解像度で計算，(b) 内側モデル解像度 1/6×1/6cos φ，(c) 内側モデル解像度 0.1×0.1cos φ．海面高度の等値線は海面付近の流線におおよそ対応し，等値線が密集している場所ほど強い流れが存在することを示す．

地理座標であり，水平格子サイズは，東西方向が 0.5 度，南北方向が 0.5cos φ 度（φ は緯度）である．なお，一定経度幅の長さは cos φ に比例するので，この設定でほぼ正方形の水平格子となる．内側モデルは外側モデルの水平格子を 3×3 分割および 5×5 分割した場合を実行しており，それぞれ水平格子数 384×288 および 640×480，水平格子サイズ 1/6×1/6cos φ 度および 0.1×0.1cos φ 度である．外側モデルと内側モデルの間での使用 CPU 数の比は，3×3 分割の場合で 1:1，5×5 分割の場合で 1:4 である．この比は格子数の比にほぼ一致しているが，内側モデルの方が短い時間ステップを適用しなければならないため，現状の設定では外側モデルの



第11図: (左)海面高度の瞬間値(cm), および(右)表層渦運動エネルギーの5年平均値(cm^2/s^2). (a), (b), (c)の並びは第 10 図と同様. 渦運動エネルギーは, 水平流速の長時間平均からのずれによる運動エネルギーとして定義.

計算進行に待ちが生じている. この待ちをできるだけ少なくするような効率化が今後もちろん必要である.

それぞれのシミュレーションは, 観測された平均的な水温・塩分分布を初期状態とし, 海面で平均的な年周期を持つ大気境界条件を与え, 15 年分行った. 以下に示す結果では, その最後 5 年を対象としている. 第 10 図に, 海面付近の長期間平均の構造を表すものとして, 5 年平均の海面高度と海面水温を示す. 現実の黒潮は, 日本南岸を北東向きに流れた後, 房総半島付近で離岸し, 黒潮続流と呼ばれる強い東向き流となって太平洋内部へ流れることが知られている.

全領域を外側モデルの解像度で計算した場合，黒潮に相当する流れは日本東岸に沿って三陸付近まで北上した後に離岸するという，現実とはかけ離れたものになっている．また，離岸後も現実の黒潮続流のような強い流れではなく,弱い東向き流として太平洋内部へ流れる.さらに，この非現実的な黒潮の北上に伴い，三陸沖での水温が現実よりも顕著に高くなってしまっている．内側モデルの水平解像度を 3 倍および 5 倍に高めた場合，黒潮の離岸位置および黒潮続流の再現性は格段に高まり，海面水温も観測されたものにかなり近づく.

第 11 図は，海面付近流速の時間変動に関わる場として，海面高度の瞬間値と海面付近の渦運動エネルギー分布を描いたものである．中緯度海洋には中規模渦と呼ばれる半径 100 km 程度の渦が存在しており，流れの変動や水温分布等に大きな影響を及ぼすことが知られている．黒潮続流域は全海洋の中でも中規模渦の活動が最も強い領域のひとつであり，海面の渦運動エネルギーが局所的には 4500 cm^2/s^2 を超えるものと推定されている．全領域を外側モデルの解像度とした結果では，顕著な渦活動は認められない．一方，内側モデルを高解像度化した結果では，黒潮続流域で多くの中規模渦が表現されている．黒潮続流の平均的な流速や流路の面では，水平解像度 1/6×1/6cosφ 度および 0.1×0.1cosφ 度の間に顕著な違いは認められなかったが，渦運動エネルギーで表した渦活動度の面では大きな違いが認められる．水平解像度 1/6×1/6cosφ 度では 4000 cm^2/s^2 に満たないが，0.1×0.1cosφ 度では 5500 cm^2/s^2 を超える.

５．おわりに

以上の例から，黒潮および黒潮続流を妥当にシミュレートするためには，少なくとも 0.1×0.1cosφ 度程度の水平解像度を実現する必要があることがわかる．このような解像度を北太平洋全域に，あるいは全海洋に適用し，気候の問題を扱うのに必要とされる数十年分の計算を行うのは，全くもって容易ではない．また，高解像度化が必要とされる領域が陸からかなり離れた場所にまで及ぶため，座標系をうまく選択することで効率よくこの領域だけを高解像度化することも容易ではない.

全海洋のシミュレーションの中で,黒潮は間違いなく重要な高解像度化ターゲットであるが，他にも高解像度化すべき重要なターゲットはいくつか存在する．それらすべてを同時に高解像度化するという方向も考えられれば，計算効率および科学的解釈の面からは別々に高解像度化するという利点も存在する．いずれにせよ，こうした状況を鑑みるに，ここで紹介したような連結階層化は非常に有効であるとともに，並列計算効率をさらに高めるべく，チューニング等をすすめる必要がある．本共同研究課題は平成 21 年度も継続しており，ハイブリッド並列化についてプログラムのチューニングを引き続き行うとともに，余裕があればスカラープロセッサ向けチューニングも行いたいと考えている.

本共同研究プロジェクトをはじめ，稼動開始直後の特別プロジェクトや通常に取得したアカウントを用いて,この 1 年あまりの間に HA8000 上である程度大規模な並列計算を様々に行ってきた．それ以前にはスカラー機での大規模並列計算の経験はあまりなく，使い始めた当初の感想は，もともと感じていた不安に比べれば意外とすんなりと使えたというものであった．しかし，使い続け，あるいは様々な情報に触れるにつれ，現状よりも格段の高速化を何とかして実現しなければという思いが強くなってきている．直接測定したことはないが（そして是非そのためのツールを整備していただきたいが），間接的な見積もりから，COCO の HA8000 での実行性能はピーク比で 5 %程度と捉えている．何もせずにこれだけの性能があれば良い方だとも聞く

が，別口で行っている別のプログラムのスカラープロセッサ向けチューニングで，キャッシュの有効利用ができる形のループであれば 20～30 %の性能は十分に出せるという例も目の当たりにしている．プログラム全体に渡ってそこまでの性能を出すことが可能とまでは期待していないが，チューニング努力によって二倍以上という速度に現実性があるのならば，それなりの方策は講じなければと感じている．

しかし,また一方で,そうしたスカラープロセッサ向けチューニングの実作業を垣間見るに，こんなことは自分ではできないというのも正直な感想である.筆者は 20 年近く前から東大のスーパーコンピュータをヘビーに使い続けており，ベクトルプロセッサを対象としたものであれば，速度向上を意識したプログラミングやある程度のチューニングを自分の手でやってきたつもりである．しかし，それが可能だったのも，ベクトルプロセッサでは，ループ長をできるだけ長くなど，かなり単純なルールに従っていれば自動的にある程度の効率が保証されたからだと言える．キャッシュメモリの挙動を理解して，それに合わせたチューニングを行うという作業は，とても出来るともやりたいとも思わない．多くの人々のノウハウが蓄積してくれれば話は違うのかもしれないが，自分にとって不可欠な道具の使い方やその将来について，不安や無力感を覚えつつある．とはいえ，スカラープロセッサによる大規模並列は，もはや引き返しようのない方向であるのもまた事実であり，チューニングの専門家を研究室なり研究所なりにとりこむという道を真剣に考えなければならないものかと思案したりもする．大学という場でそうした人材を確保し続けるのは，これまた容易なことではないのだが．

参 考 文 献

Hasumi, H. (2006): CCSR Ocean Component Model (COCO) Version 4.0, CCSR Report No. 25, 103pp. (http://www.ccsr.u-tokyo.ac.jp/~hasumi/COCO/coco4.pdf)

IPCC (2007): Climate Change 2007: The Physical Science Basis, Cambridge University Press, 996 pp.

Mesinger, F., and A. Arakawa (1976): Numerical methods used in atmospheric models, GARP Publication Series, 17, World Meteorological Organization, Geneva, 64pp.

Sakamoto, T. T., et al. (2005): Responses of the Kuroshio and the Kuroshio Extension to global warming in a high-resolution climate model, *Geophy. Res. Lett.*, **32(17)**, L14617.

Sakamoto, T. T., and H. Hasumi (2008): Pacific upper ocean response to global warming -climate modeling in an eddying ocean regime-, in *Eddy-Resolving Ocean Modeling*, Geophysical Monograph Series 177, American Geophysical Union, 265-279.

津波発生伝播の大規模3次元シミュレーション

齊藤竜彦

防災科学技術研究所

古村孝志

東京大学大学院情報学環

片桐孝洋

東京大学情報基盤センター

中島研吾

東京大学情報基盤センター

１．はじめに

海に囲まれた日本では1896年明治三陸地震（死者22000人）や、1944年東南海地震（死者1200名）のように大津波による被害が頻発している。緊迫する東海地震や宮城県沖地震に備え、津波災害の軽減のための津波発生と被害予測シミュレーションの高精度化が急務の課題である。

津波の発生と伝播のシミュレーションでは、１）断層運動が海底地殻変動を起こす過程、２）これにより津波が発生する津波波源形成過程、そして３）複雑な海底を津波が伝播する過程の３つを、厳密に評価することが不可欠である。ところが、1970年代に開発された、現在一般的に用いられている津波評価計算法では、計算の簡単化のために、基本方程式を直接計算するかわりに、近似式が使用される。たとえば、津波波源形成では、均質な地下構造モデルを用いて海底地殻変動を計算し、これを初期津波とするのが一般的である。また、津波の伝播過程の計算では、３次元流体式の計算のかわりに、2次元線形長波方程式が多用されている。この結果、これらの近似の適用条件を越える地震と津波において、津波高や継続時間の過大評価や過小評価など津波警報の問題をたびたび起こしてきた。

2006年千島列島の地震（M7.0）では、地震発生から６時間を過ぎて津波警報が解除された後に、大きな津波が到来した。また、2007年千島列島の地震（M7.2）では、北海道～関東の太平洋岸に津波警報が発令されたが、実際に観測された津波高は1/2以下に過ぎなかった。これらの事例は、従来の簡便な津波評価の限界を示しており、これら少数の事例とはいえ例外事象が津波警報の信頼性を損なう原因となる。

近年の計算機の急速な性能向上によって、津波シミュレーションが行われ始めた1970年代当初に比べて1000万倍以上高速な計算が現実化した。この結果、わざわざ近似計算を用いなくとも、３次元流体運動方程式の直接計算により高精度に評価することが十分可能になった。次世代の高精度津波予測シミュレーション法の確立のためには、３次元ナビエ・ストークス方程式に基づく差分計算の高度化が必要不可欠である。

２．Ｈ２０年度共同研究の目標

本研究では、将来発生が危惧される巨大地震による津波被害の軽減を目的に、大規模津波シミュレーションのための大規模計算コードを開発する。既に開発した３次元ナビエ・ストークス式の差分法計算に基づく津波計算コードを改良し、T2K オープンスパコン（東大）に代表される、スカラー型超並列計算機において高い並列化スケーラビリティと実効性能を引き出すことのできる実用化コードを整備する。また、本津波計算コードを用いて、津波発生過程の詳細シミュレーションを行うとともに、近年の津波地震の大規模・高精度津波シミュレーションを実施し、室戸沖の海底下に設置された高分解能の沖合ケーブル津波計で記録した津波データとの比較からモデルの有効性と精度を検証する。

３．Ｈ２０年度共同研究の成果

（１）３次元津波シミュレーションによる高精度津波計算

[３次元津波計算の概要]

３次元の津波生成と伝搬をシミュレーションするためのナビエ・ストークス方程式を差分法で離散化したものである。自由表面をもつ流体の一般的な解析手法である SOLA-SURF 法をもとにしている。数値計算アルゴリズム上の分類は、差分法の陽解法となる。求めるべき変数（3次元配列となる）は、x、y、z 方向に対応する流体の速度 u、 v、 w と、流体中の圧力 p である。

[津波発生過程のシミュレーション]

津波予測に一般的に用いられている２次元津波シミュレーションでは､水平流のみを計算し、鉛直流を直接計算することができない。このため、地震による海底地殻変動が海水を持ち上げることにより発生する、初期津波波高の形成はシミュレートできず、海水面に初期津波波高分布を初期条件として強制的に与えていた。このとき、従来の研究の多くでは、海底における地殻上下動変動の分布と海面における初期津波波高の分布が等しいという簡単な仮定が使用される場合が多かった。一方、ここで用いる３次元津波シミュレーションコードは、水平流および鉛直流を直接計算するため、地震による海底地殻変動から初期津波波高分布の形成過程をシミュレートすることが可能となる。

津波発生過程のシミュレーション結果を確認すると､たとえば､プレート境界地震のように、地震断層の面積が広い範囲に広がり、かつ、断層の傾斜角がなだらかとなる場合には、地震により海底面に生じる上下動変動成分と海底面に生じる初期津波の波高分布がほぼ一致する。この場合､従来の津波発生シミュレーションで用いられてきた仮定は妥当なものとなる｡一方で、プレート内地震の場合のように、地震の規模に比べて断層面積が狭く、大きな断層滑りが発生する場合、もしくは、断層の傾斜角度が急なために断層の真上の狭い範囲にのみ海底地殻変動が発生する場合には、初期津波波高は海底面の上下動成分に比べて、有意に小さくなる。このことは、プレート境界地震の場合には、従来の海底変動＝初期津波という近似がそのまま適用できるが、プレート内部地震の場合には、この近似が成立せず津波波高分布を津波発生過程シミュレーションにより導出する必要があることを示している［Saito and Furumura 2009b］。

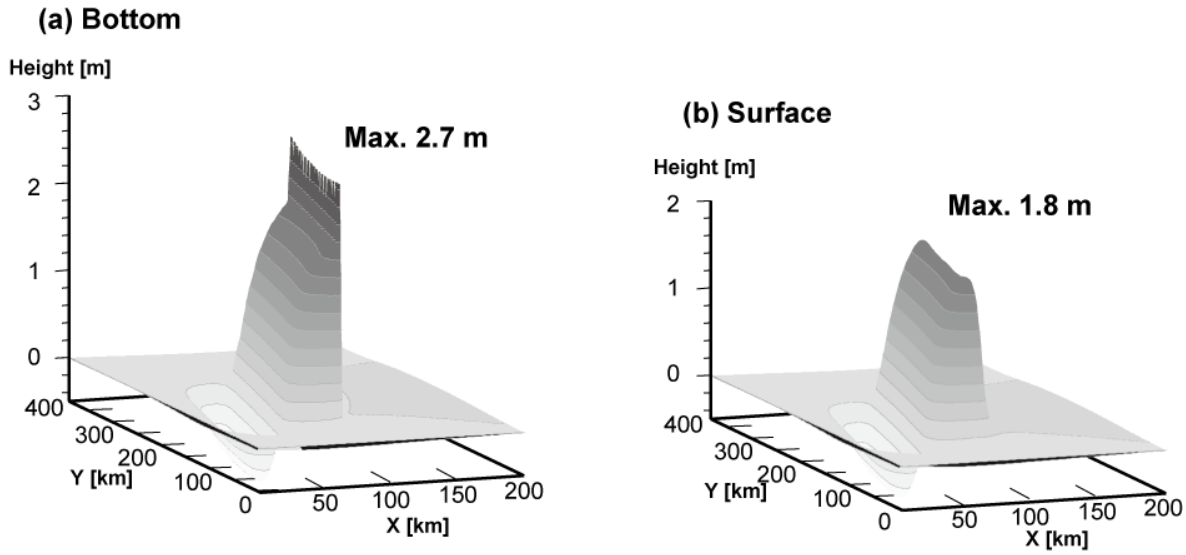


図１　津波発生過程シミュレーション。(a)海底における上下動変位分布と(b)海面に形成される初期津波波高分布。

[高精度伝播過程のシミュレーション]

３次元ナビエ・ストークス方程式より導かれる２次元津波方程式には、さまざまな近似方法が存在し [Saito and Furumura 2009c など]、一般に、近似精度が高くなるにつれ、方程式は複雑になり、また、計算時間も長くなる。それゆえ、津波シミュレーションでは、水深や海底面形状の複雑さ、津波の周期特性など、津波発生伝播の状況に応じて適切な２次元津波方程式を使用する必要がある。例えば、津波伝播過程評価の多くでは、非分散性の津波方程式が使用されてきたが、津波の波長が短く水深が深い場合には、地震断層走向に直行する方向に生まれる、分散性を持つ特異な津波波形を評価することができない。

2004 年に発生した紀伊半島南東沖の地震による津波観測記録に顕著に見られる津波の分散波は、分散波理論に基づく津波シミュレーション、または３次元ナビエ・ストークス方程式の直接計算する津波シミュレーションによって初めて再現できるなど、津波シミュレーションの高度化の重要性が指摘されている [Saito and Furmura 2009a]。

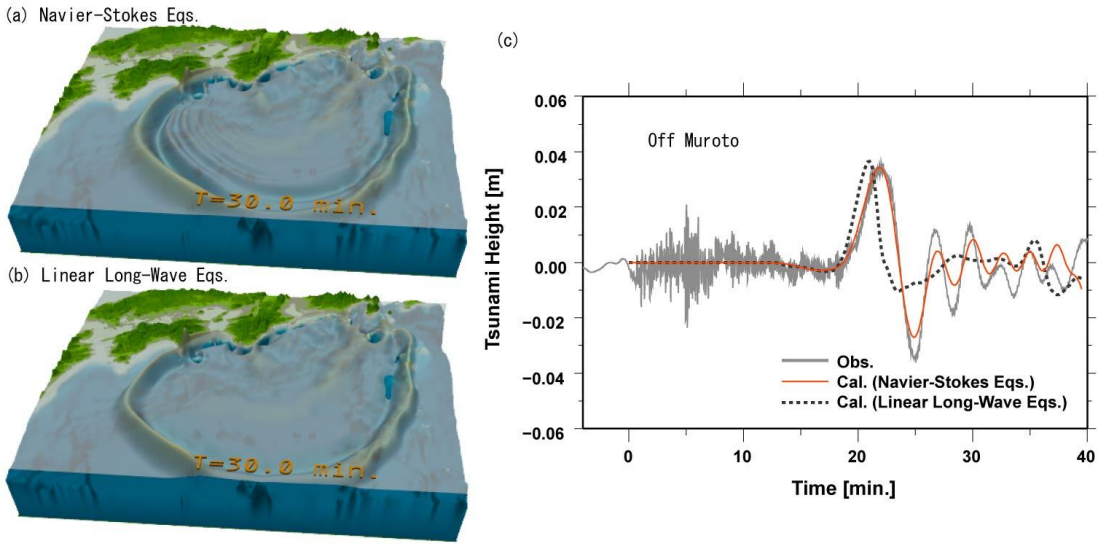


図 2 2004 年紀伊半島南東沖地震時の津波。

(a) 3 次元ナビエ・ストークス方程式に基づくシミュレーション結果と(b)従来の 2 次元津波シミュレーション法（線形長波法方程式）に基づくシミュレーション結果。(c)JAMSTEC 室戸沖観測点で観測された津波波形記録（灰線）とシミュレーション結果との比較。3 次元津波シミュレーション（実線）は、従来の 2 次元津波シミュレーション法（破線）に比べて、津波の分散波形を良く再現する。

[津波波源推定への応用]

こうして、津波の第一波だけでなく、津波波源形成と伝播過程において複雑化する後続波形も含めて良く再現出来るようになると、この津波波形の特徴を積極的に利用することで津波波源である地震断層に関してより多くの情報を抽出できるようになる。例えば、伝播過程で現れる津波の分散現象は、地震断層の方向に関して強い方位依存性をもつことから、地震波を用いた震源過程解析からは断層走向が決定できない場合でも、津波記録に現れる分散性の方位依存性を調査することで地震断層の走向を正確に推定することができる。さらに,津波だけでなく,地震動を併せてモデリングすることによって，より短周期成分の地震断層運動を理解することが可能となる［Furumura and Saito 2009，古村・齊藤 2009］.

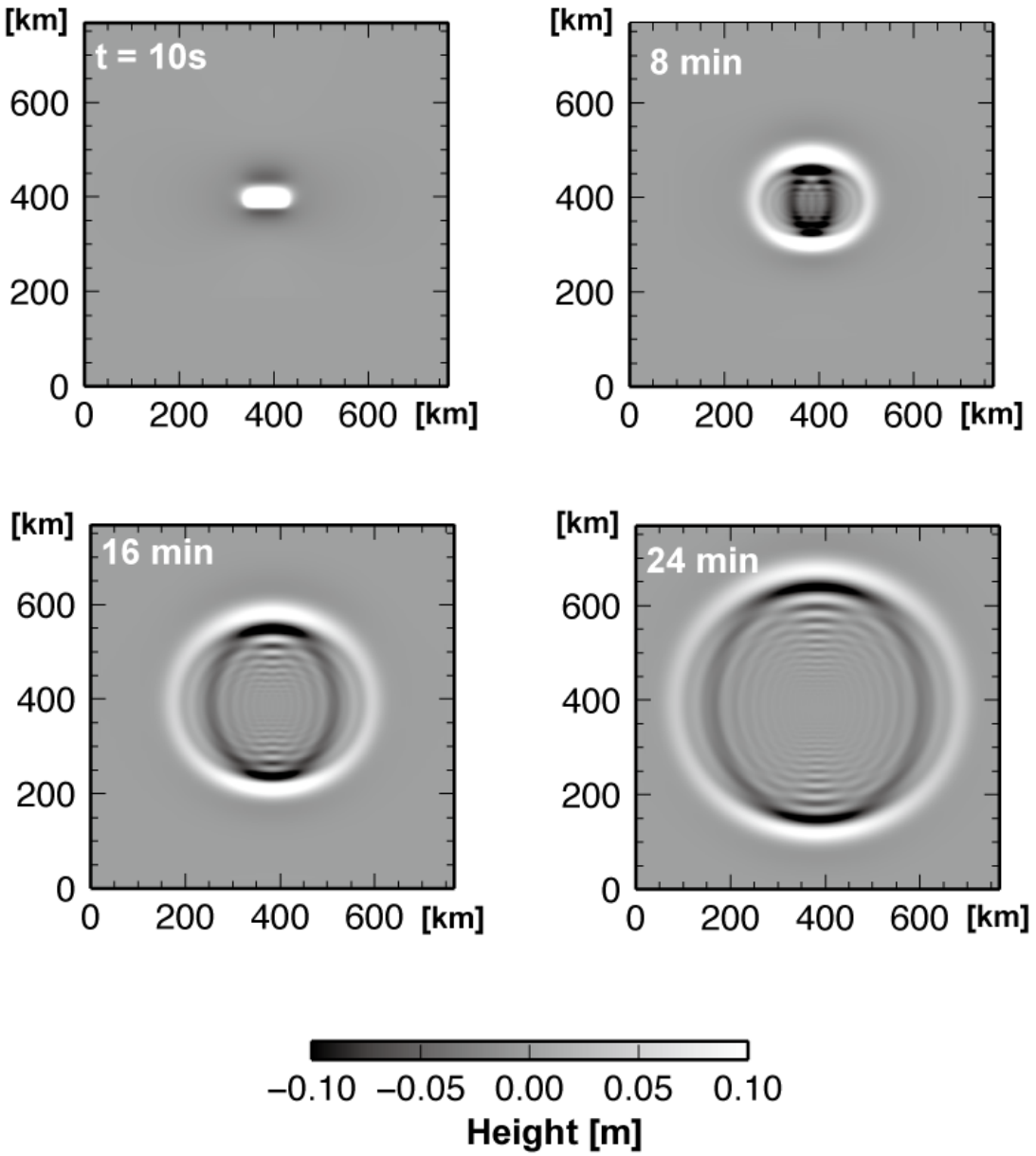


図 3　津波シミュレーション結果。

波源から周囲に伝播する際、津波は分散する。特に、断層走向に垂直な断層の短軸方向（図中の上下方向）へ伝播するときに、強い分散を示す。いっぽう、断層の長軸方向（図中の左右方向）では津波の分散は小さい。

(2) 大規模３次元津波計算におけるコード最適化

津波シミュレーションでは、現実的な海底地形における海水の流動を３次元の流体式（ナビエ・ストークス式）を用いて評価する。本計算では、津波の発生と伝播を、非圧縮性流体を考慮した SOLA 法［Hirt et al. 1975］に基づく FDM 計算に基づいて行う.

(i)ホットスポット部分と演算カーネル

このコードの最も時間のかかる部分（ホットスポット）を、図 4 に示す。

```
      do itl = 1, itlmax
        err = 0.0
        演算カーネル部分 (err)
        if (err. lt. eps) goto OUT
      enddo
OUT   continue
```

図 4 ホットスポット部分の構成

ここでは、演算カーネル部分で計算した精度 err が要求値である eps よりも小さくなるか、最大反復回数 itmax まで、演算カーネル部分が呼ばれる。ここで、演算カーネルの具体的な計算内容は、以下に示すｘ、ｙ、ｚ軸に関する 3 重ループである（図 5)。

```
        do k = 1, nz
          do j = 1, ny
            do i = 1, nx
              if (mos(i,j,k)) then
                dd = (u(i,j,k)-u(i-1,j,k))*dxinv
     &             + (v(i,j,k)-v(i,j-1,k))*dyinv
     &             + (w(i,j,k)-w(i,j,k-1))*dzinv
                dp = beta*dd
                u(i,j,k)   = u(i,j,k) + dtdx*dp
                u(i-1,j,k) = u(i-1,j,k) - dtdx*dp
                v(i,j,k)   = v(i,j,k) + dtdy*dp
                v(i,j-1,k) = v(i,j-1,k) - dtdy*dp
                w(i,j,k)   = w(i,j,k) + dtdz*dp
                w(i,j,k-1) = w(i,j,k-1) - dtdz*dp
                p(i,j,k)   = p(i,j,k) + dp
                err = max (err, abs(dd))
              endif
            enddo
          enddo
        enddo
```

基本演算コード

図 5 演算カーネル部分（論理マスクコード）

3 重ループの中央に津波に関する演算の実行を判断する論理マスク mos(i、j、k)が存在し、予め計算を行う範囲を定め、論理マスクを.true.か.false.に設定しておく。３次元のシミュレーション領域には、固体（陸地）部分と液体（海洋）部分の２つがあり、津波計算は液体部分のみ行えば良いためである。このような計算コードの最適化を考えた場合、以下の問題点がある。

（１） IF 文がループの中央にあるため比較演算の回数が多く、この分岐予測は事前に困難のため命令発行が先行して行えない。

（２） k ループ長一定ではなく、i、j ループの値に依存して k ループ長が決まる。このため、k ループを連続とするコード最適化（データのプリロードなど）が計算機アーキテクチャによっては難しい。

上記問題（１）を解決するには、i、 j 毎に k ループ長を記憶した変数 kb(i、j)と kt(i、j)を導入するこよにより、IF 文を消去することができる。以下の図 6 にコードを載せる。

```
do j = 1, ny
  do i = 1, nx
    kb2 = kb(i,j)+1
    kt2 = kt(i,j)-1
    if(kt2 .gt. kb2 ) then
      do k = kb2,kt2
        図 2 の基本演算コード
      enddo
    endif
  enddo
enddo
```

図 6 IF 文除去コード

ここで、最内ループ中から IF 文を除去することができたが、最内ループが k に対するループであることに注意が必要である。計算で用いる、u、 v、 w、 p 配列において、連続してデータが入っている方向が i ループの方向である（Fortran の場合）ため、これらの配列は連続アクセスとならず、キャッシュメモリ上にあるデータの利用が妨げられ、演算の激しい性能低下を引き起こす。

以上のように、ここで述べる 3 次元津波伝搬シミュレーションコードは単純な 3 重ループ(つまり、i、 j、 k ループの範囲がすべて自明）で無いためにコード最適化は簡単ではない。

(ii)高速化に向けたアルゴリズムの改良

もし、海の深さが一様な場合には、図 5 から IF 文を取り除くことができ（図７)、コードの最適化は容易である。当然、実際の複雑な海底地形の津波シミュレーションには利用できないが、ベンチマークコードとしてよい例題となるため、以降これについて考察を深める。

```
do k = 1, nz
  do j = 1, ny
    do i = 1, nx
       図 5 の基本演算コード
    enddo
  enddo
enddo
```

図 7 単純 k、j、i ループコード

ここで、k ループ長は座標(x、y)の地点における海の深さにより決まる。すなわち、陸地の部分では k ループ長は 0 もしくは極めて短い長さとなり、深海の部分では長い k ループ長となる。津波シミュレーションの多くでは、陸地よりも深海の割合が圧倒的に大きいため、ある kblk*kblk の区間 [x:x+kblk, y:y+kblk]で深海が存在すれば、k ループ長を長くとることができる。この特徴を考慮したコードは、以下の図 8 ようになる

```
do jjj = 1, ny/kblk
   jj = 1 + (jjj-1)*kblk
   do iii = 1, nx/kblk
     ii = 1 + (iii-1)*kblk
     do k = kbmin(iii,jjj), ktmax(iii,jjj)
       do j = jj, jj+kblk-1
         do i = ii, ii+kblk-1
            図 5 の基本演算コード
         enddo
       enddo
     enddo
   enddo
enddo
```

図 8 単純ブロック化コード

ここで、配列 kbmin(iii, jjj)と ktmax(iii, jjj)は、区間 [iii:iii+kblk, jjj:jjj+kblk]において k ループの開始値と終了値を記したものである。

(iii) 一般化したブロック化コード (提案手法)

図 8 のブロック化コードは、kblk*kblk の区間内で、k-ループの開始値と終了値が一定でないと使えない。もちろん、実際の海底地形データには凹凸があり水深は一定でない。そこで、計算領域全体にわたって、水深の一定幅の部分の演算をブロック化し、それより浅い部分と、深い部分の演算を別に行うのが有効である、

図 9 のコードは、先の図 8 のコードをもとに、k ループについて、1)ブロック化しない、海の浅い部分、2) 中間のブロック化する部分、3) ブロック化しない、海の深い部分、の 3 つに

分割コードである。

```
c          ====  １）ブロック化しない、浅海部分
           do jjj = 1, ny/kblk
             jj = 1 + (jjj-1)*kblk
             do iii = 1, nx/kblk
               ii = 1 + (iii-1)*kblk
               if (kbmin(iii,jjj) < ktmax(iii,jjj)) then
                 do k = 1, kbmin(iii,jjj)-1
                   do j = jj, jj+kblk-1
                     do i = ii, ii+kblk-1
                       図５の基本演算コード
                     enddo
                   enddo
                 enddo
               endif
             enddo
           enddo
c          ==== ２）ブロック化する中心部分
           do jjj = 1, ny/kblk
             jj = 1 + (jjj-1)*kblk
             do iii = 1, nx/kblk
               ii = 1 + (iii-1)*kblk
               do k = kbmin(iii,jjj), ktmax(iii,jjj)
                 do j = jj, jj+kblk-1
                   do i = ii, ii+kblk-1
                     図５の基本演算コード
                   enddo
                 enddo
               enddo
             enddo
           enddo
c          ==== ３）残りの深海部分
           do jjj = 1, ny/kblk
             jj = 1 + (jjj-1)*kblk
             do iii = 1, nx/kblk
               ii = 1 + (iii-1)*kblk
               if (ktmax(iii,jjj) < kbmin(iii,jjj)) then
                 kstart = 1
               else
```

```
          kstart = ktmax(iii,jjj)+1
        endif
        do k = kstart, nz
          do j = jj, jj+kblk-1
            do i = ii, ii+kblk-1
              図5の基本演算コード
            enddo
          enddo
        enddo
      enddo
    enddo
```

図9　一般化ブロック化コード（提案手法）

図9の1)と3)の演算では、先に述べた論理マスクを使ったコードでも、IF文を除去したコードでもどちらも利用できる。なお、各kblk*kblkの区間に連続した部分が存在しない場合には、3）の部分でブロック化なしのコードが実行すれば良く、最悪でもブロック化を行わない、従来の論理マスクを使ったコードと演算の内容は同一になる。

［性能評価］

(1) 評価環境

T2Kオープンスパコン（東大版）を利用して、これらの演算の性能評価を行った。T2Kの各ノードは、AMD Opteron 8356（2.3　GHz、 4コア)を4台（4ソケット）搭載しており、メモリは32　GBである。理論最大演算性能は、ノードあたり147.2　GFLOPSである。通信性能は運用クラスタ群で異なるが、ここではMyri-10G通信ネットワークがノード間に4本実装され、最大5GB/secの双方向性能を有するA群を利用して計算を行った。用いたFortranコンパイラは、日立最適化Fortran90 V01-00-/Bであり、コンパイラオプションとして、変数の倍精度化、自動インライン展開、最適化、自動並列化なし （-precexp=4 -autoinline -opt=ss -noparallel）を指定した。

(2) ベンチマークプログラム

3次元津波伝搬シミュレーションコードは、MPIによる並列化が行われているが、ベンチマークテストでは、ここからMPI通信部分を除去し1コアで実行できるプログラムに修正した。シミュレーションの問題サイズは、nx=256、 ny=256、 nx=50 である。ここでは海底の深さ（水深）は均一とした。水深に関するループ長はk=10～48である。先に示した津波シミュレーションの主要カーネルについて9回の反復計算を行い、演算時間を計測した。

ブロック化コードのブロック幅kblkについては、32、64、128、256と変化させた場合の時間を計測し、256 が最速であったことから、以降すべての計算でこの値を採用した。ベンチマークプログラムでは海底の深さ（nx）を一定としたため、kblk = nx = 256 の場合が最速となるのは当然とも言える。

図10に、津波伝播シミュレーションコードに対して各種のチューニングを行った結果の実行

時間を載せる。なお、「w 配列局所」とは、3 次元配列アクセスの際の 3 次元目のキャッシュミスヒットを防止するため、計算のカーネル部分の計算に入る前に、用意した一次元配列にデータをコピーし、そして、カーネル部分の計算後に計算結果をもとの 3 次元配列に書き戻すようにした修正したコードである。

図 10 のベンチマークテストの結果から以下のことがわかる。

（1） 「IF 文除去コード」は、「論理マスクコード」より実行速度が遅い。理由は、IF 文除去コードは最内ループが k ループとなり、これは配列の第 3 要素であることから、演算においてメモリアクセスが不連続となりデータアクセス時間が増大するためである。

（2） 「単純 k、i、j ループコード」は、論理マスクコードに比べて演算量が増えるにもかかわらず、論理マスクコードより高速である。このことは、ループ中での論理マスク計算に必要な IF 文の実行オーバヘッドが大きいことを意味している。

（3） 「単純ブロック化コード」は、単純 k、j、i ループコードより高速である。これは、余分な演算を行わない効果である。

（4） 「IF 文除去コード」において、配列のならびを u(i, j, k) から u(k, i, j)に変更してメモリアクセスを連続化したコード（10 の最右列）は論理マスクコードより実行速度が遅い。この理由は、配列変数の大きさが 50 に定義されているのに対して、k ループの変化する範囲が 10～48 であり、配列全体への連続アクセスの効率が悪いことによる。

以上ベンチマークテストの結果より、最初の論理マスクコード（60.3 秒）に対して、「一般化 BLK+W 局所化」の最適化を行ったコード（40.6 秒）では 48%の速度向上が達成された。また、この最適化コードは、最も演算速度の遅い IF 文除去コード（111 秒）に対しては 270 %の速度向上となる。なお、一般化 BLK+W 局所化コードのカーネル部分の実効速度は、論理ピーク性能（9.2 GFLOPS）の 14.5%（1.34 GFLOPS）であった。

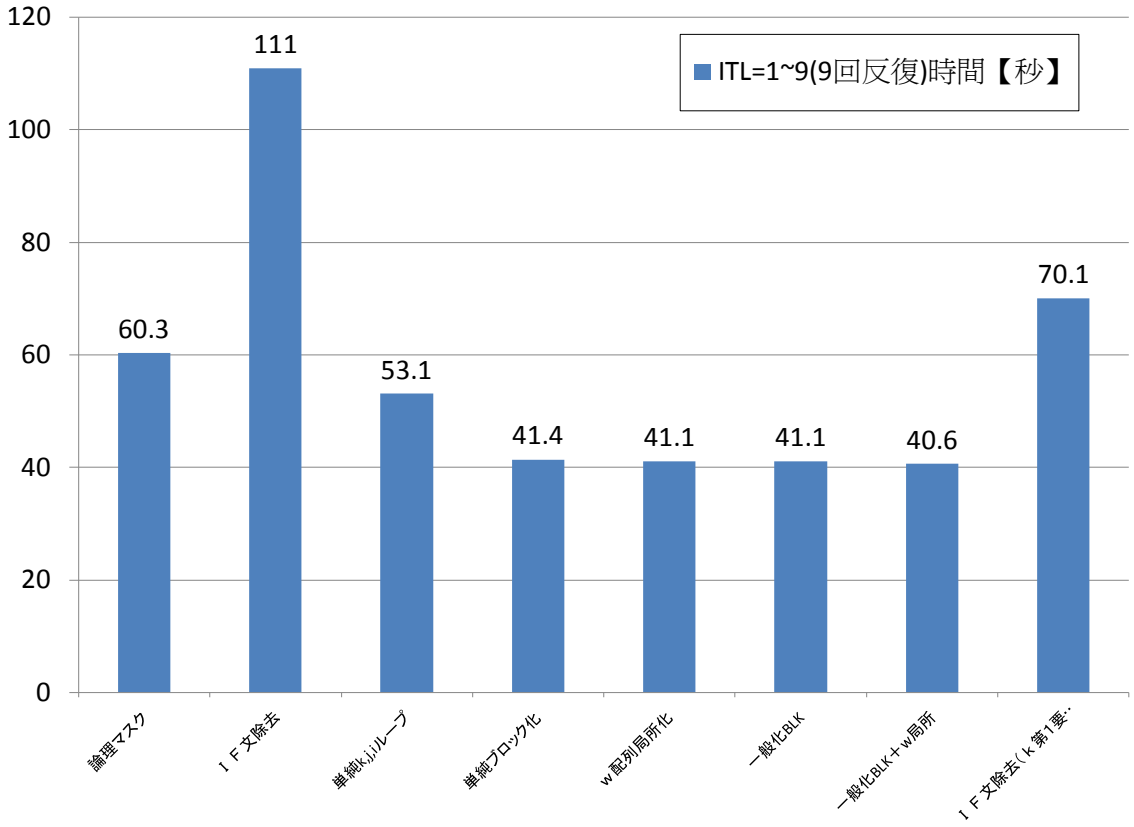


図 10　各最適化コードの実行時間（秒）

(3) MPI プログラム

次に、問題サイズを nx=2048、 ny=1600、 nx=104 に拡大し、MPI で領域分割して並列化したプログラム[4]に、本提案手法を実装して性能評価を行った。並列計算では、T2K のノード内の 16 個のコアとノード間の通信にどちらも MPI を用いる、ピュア MPI 実行を行い、numactl コマンドを用いて MPI プロセスをランク番号の順に、各コアに一つづつ割り当てた。こうすることにより、隣接プロセス通信が、同じソケット内のコア同士で行えるようになり、通信の効率化が期待できる。

並列計算では、CPU core（プロセス）数に応じて３次元計算領域を水平（x、y）方向に領域分割する。たとえば 8 ノード(16cores/node)を用いた並列計算では、64core(64 プロセス)となるがこのとき、X 方向に 8 プロセス、Y 方向に 8 プロセスを割り当てる。したがって各 core の担当する領域は nx/8×ny/8×nz である。次に 16 ノードの場合は、X 方向に 16 プロセス、Y 方向に 8 プロセスというように、X 方向から優先的に領域分割を進める。

なお、ブロックサイズ kblk は、各プロセスにおける領域サイズと同じ大きさに設定した。

実行結果を表 1 にのせる。表 1 では問題サイズを固定して、core 数を増やした場合の並列計算の実行速度の増加の性能評価の結果（Strong Scaling Test）を示している。ノード数（core 数）が増加するにつれ、ここで提案した最適化手法の効果が表れる。従来の演算手法（論理マスクの利用）に対して、本最適化手法の採用により最大で 5.8 %の高速化が達成した。また、各ノードの 16 core のうち､8 core のみを用いた並列計算では本提案手法の効果はより大きく、従来のコードに比べて 9.2 %の高速化が達成された。このように、本最適化手法は、T2K におい

ては 8core/node による実行での効果が大きい。また、64 ノード（512～1024 core）を用いた並列計算では、1024 core（ノードあたり 16 core を使用）を用いた計算時間は 1302 秒、また 512 core（8cores/node）を用いた計算時間が 1262 秒となり、core 数が少ない場合のほう高速化するという興味深い結果が得られた。同様の計算を従来のコードを用いて行ったところ、1024 core（16cores/node）実行のほうが高速であった。これは、最適化を行ったコードではメモリアクセスの連続化が進んだ結果、演算効果を得るためには広いメモリバンド幅が必要となったこと、ところが、T2K ではメモリバンド幅の要求に十分応えることができず、16 cores/node 実行次には多数の core からの同時メモリアクセスに伴なう速度低下が起きてしまったものと考えられる。なお、本件については、通信性能と演算性能に関するモニタリングと性能モデル化を行ない、現象の解析が必要である。

また、表 1 より、ノード数（core 数）が 8～32 に増加するほど提案する最適化手法の効果が大きくなることがわかる。これは、計算領域全体に占める、海の領域の計算量が陸の領域の計算量より大きいため、一般にノード数が増加すると並列計算における core 毎の負荷分散が悪くなり並列計算のバランスが崩れる。ところが提案する最適化手法を取り入れると、海の領域で k ループが連続的に取れることができるようになり、計算が高速化される。このことにより、これまで計算時間が大きくかかっていた core の計算が高速化し、各 core の負荷分散が改善され、並列計算全体の性能が大きく向上する。並列計算の Strong Scaling Test では、core 数が増えると各 core のデータアクセス領域が縮小する。したがって、演算を行う変数のデータがキャッシュに乗りやすくなり、メモリバンド幅の問題が解決して高速化が進むことも考えられる。このことから、本最適化手法は、より多数の core を用いた超並列計算の実行時にさらなる速度向上が期待できる。

表 1　MPI プログラムでの実行時間[秒]。括弧内の数値は 8 ノード計算を 1 とした場合の、計算速度の向上を示す台数効果。

ノード数	従来（16cores/node）	提案（16cores/node）	提案法速度向上	従来（8cores/node）	提案（8cores/node）	提案法速度向上
8	9183 (1x)	9151 (1x)	0.34%	10122 (1x)	10161 (1x)	−0.38%
16	4152 (2.21x)	3942 (2.32x)	5.3%	4570 (2.21x)	4628 (2.19x)	−1.2%
32	2334 (3.93x)	2205 (4.15x)	5.8%	2527 (4.00x)	2314 (4.39x)	9.2%
64	1354 (6.78x)	1302 (7.08x)	3.9%	1377 (7.35x)	1262 (8.05x)	9.1%

４．今後の展望

津波発生伝播を評価するナビエ・ストークス式の数値計算に用いる SOLA 法［Hirt et al. 1975］は、次世代スパコンなどのスカラー型並列計算機において高い性能を発揮することが期待される。この計算では、流体の非圧縮性条件の計算において、逐次的に修正が行われ、その収束計算に計算時間全体の７割の時間を要している。大規模な津波伝播シミュレーションの実用化に向けて、この計算時間を短縮するために、キャッシュメモリの活用などのコードチューニング作業をさらに進めることが必要である。計算効率を上げる別の有効な方法として、流体計算で良く用いられる MAC 法によるポアソン方程式の半陰解法の利用も有効である。MAC 法では、収束計算が不要であり、陰解法の長所である数値計算の安定性向上も期待できる。ポアソン方程式の計算の効率化に関してこれまで研究が進んでおり、各種の並列ポアソン方程式ソルバーが活用できるほか、マルチグリッド法などのより高度な手法を用いた並列計算効率の向上も期待できる。

今後、大規模な３次元津波発生伝播シミュレーションの実現化を急ぎ、将来の南海トラフ巨大地震の連動発生等による津波の高精度評価に活用したい。そして、過去の大地震による津波とその災害を再現するとともに、将来の津波被害を正しく予測し、災害軽減に役立てるための高精度シミュレーションを行いたい。

参　考　文　献

Furumura, T. and T. Saito, An integrated simulation of ground motion and tsunami for the 1944 Tonankai earthquake using high-performance super computers, Journal of Disaster Research, 4, No 2, 118-126, 2009.

古村孝志・齊藤竜彦、地震―津波連成シミュレーション、日本計算工学会編、超ペタスケール・コンピューティング、第 15 章、丸善、出版予定。

Hirt, C. W., B. D. Nichols and N. C. Romero, SOLA - A numerical solution algorithm for transient fluid flows, Los Alamos National Laboratory report LA-5852, 1975

Saito T. and T. Furumura, Three-dimensional simulation of tsunami generation and propagation: Application to intraplate events, J. Geophys. Res., 114, B02307, doi:10.1029/2007JB005523., 2009a.

Saito T. and T. Furumura, Three-dimensional tsunami generation simulation due to sea-bottom deformation and its interpretation based on the linear theory, Geophys. J. Int., doi:10.1111/j.1365-246X.2009.04206.x, 2009b.

Saito T. and T. Furumura, Scattering of linear long-wave tsunamis due to randomly fluctuating sea-bottom topography: coda excitation and scattering attenuation, Geophys. J. Int., doi:10.1111/j.1365-246X.2009.04206.x, 2009c.

地球ダイナモの新しいシミュレーションコード開発とその応用

陰山　聡

神戸大学大学院 工学研究科

大野暢亮

海洋研究開発機構 地球シミュレータセンター

１．はじめに

地球の内部は二層に分かれている。外側は岩石でできたマントル層、内側は主に鉄でできた核（コア）の層である。核自体も二層に分かれており、外側は外核、内側は内核と呼ばれる。外核は液体、内核は固体状態であることが地震波の観測から確認されている。

地球磁場の源は外核中の液体鉄である。この外核の液体鉄が対流運動するために、その運動エネルギーが MHD（Magnetohydrodynamics）ダイナモ作用によって磁場のエネルギーに変換されることで地球の双極子磁場が生まれている。

外核の液体鉄の流れを駆動している主な要因として熱対流が考えられているが、それ以外にも組成対流、歳差運動による流れなど様々な可能性が検討されており、未だ確定はしていない。現実にはこれらが複合して外核に流れを作り出しているのであろう。

双極子磁場の自発的な生成に加えて、地球磁場に関してもう一つ確認されている不思議な事実は、双極子モーメントの向き、即ちN極とS極の位置が短い時間に（非周期的に）突然反転するという現象である。最近500万年ほどの間では平均すれば2，30万年に一度、地球の磁気的な南北は逆転してきた。一番最近の逆転は78万年前に起きた。今の双極子磁場もいつか逆転することは確実である。双極子磁場の自発的な生成とその逆転を理解することが地球ダイナモ研究の大きな目標である。

２．シミュレーションモデル

地球の外核を想定し、二つの同心球面に挟まれた球殻状の領域を考える。その中に電気伝導性流体（MHD 流体）が入っている。内側の球面（半径 $r = r_i$）は高温、外側の球面（半径 $r = r_o$）は低温に保たれている。球の中心方向に重力がはたらき、二つの球殻は同じ角速度 Ω で回転する。温度差が十分に大きければ（レイリー数 Ra が十分高ければ）内部の流体は熱対流運動し、MHD ダイナモ機構によって、対流の運動エネルギーが磁場のエネルギーに変換され、磁場が生成される。

比較的低い解像度で十分であればこの問題を数値的に解くのはそれほど難しいものではない。我々も含めて世界中のグループが地球ダイナモシミュレーションに挑戦し、双極子磁場の自発的生成やその非周期的逆転現象を計算機の中で再現することに成功した。こうした研究を通じて、双極子磁場の自発的な生成とその逆転という現象は、マントルの影響など外部に複雑な要因を求める必要はなく、MHD 方程式に内在する性質であるということが示されたことは、計算科学としての地球ダイナモシミュレーション研究の大きな成果であると我々は考えている。

スーパーコンピュータの進歩に伴い、地球ダイナモシミュレーションも着実に進歩してきた。

その進歩の指標の一つとしてよく使われるのは、無次元量の一つであるエクマン数 $E_k = \nu / 2\Omega r_o^2$ である。（νは動粘性率。）エクマン数 E_kは、運動方程式の中の粘性力とコリオリ力の比である。地球外核の E_kは $O(10^{-15})$程度と見積もられる。つまり外核は回転の影響を非常に強く受けた回転対流系である。これほど小さな無次元量を持つ系を直接数値シミュレーション（DNS）で解くのは現在のスーパーコンピュータでも難しい。エクマン数がνに比例することからもわかるように、エクマン数を小さくした計算を実行するには高い空間解像度が要求される。回転流体系に現れる境界層（エクマン層）の厚さがエクマン数の平方根に比例することからもそれはわかる。

1990 年代に我々が地球ダイナモシミュレーション研究を始めたとき、計算で用いたエクマン数は $E_k = O(10^{-4})$であった。それでも回転の効果が卓越する対流系や、MHD ダイナモ系の特徴は十分に再現することができた。そして、スーパーコンピュータの進歩と共にエクマン数を始めとする新たなパラメータを開拓することで、双極子磁場の自発的生成や、その逆転など、地球磁場の特徴的な性質を計算機の中で少なくとも定性的に再現することに成功した。

スーパーコンピュータの進歩と共に、地球ダイナモシミュレーションで使われるエクマン数はゆっくりとではあるが着実に小さくなっていった。地球ダイナモシミュレーション研究の世界は、最先端のスーパーコンピュータを駆使していかに小さなエクマン数を実現するかという競争という面もある。我々は 2003 年頃から、地球シミュレータを限界まで使った最高解像度の（即ち最小エクマン数の）地球ダイナモシミュレーションを実行することを目標とし、研究を進めてきた。

解くべき基本方程式は規格化された圧縮性 MHD 方程式である。計算格子にはインヤン格子[1]を用いた。格子サイズは 511（r）x 514（θ）x 1538（ϕ）x 2 （Yin/Yang）である。ここで r は半径を、θは余緯度（赤道を中心に南北 45 度）を、ϕ は経度（270 度）を表す。この格子間隔は外核表面の赤道面上で約 11km に相当し、これは地球ダイナモシミュレーションでは世界最高解像度である。

プラントル数 Pr と磁気プラントル数 Pm はどちらも 1 とした。圧縮性流体のレイリー数 Ra は深さの関数である。ここでは $Ra(r_i)=1.5 \times 10^{10}$ の結果を示す。これは対流の臨界レイリー数の 300 倍以上 1000 倍以下であることを数値的に確認した。エクマン数は $E_k = 2.3 \times 10^{-7}$ である。

速度場の境界条件は粘着、ベクトルポテンシャルに対しては、磁場が境界上で半径方向成分しか持たないように設定した。温度は境界上で固定、質量密度に対しては境界面上で連続の式を片側差分により解く。

初期条件は対流と磁場のない、熱伝導だけの（不安定）平衡状態である。シミュレーション開始時に温度場とベクトルポテンシャル場にランダムな弱い攪乱を加える。二つの球面間の温度差が十分に大きければ（レイリー数が十分に高ければ）熱対流運動が開始し、さらにその流れの運動エネルギーを消費して MHD ダイナモ作用により磁場が生成される。

磁場の成長と比較して対流場の時間発展は比較的早く、速度場が先に飽和する。その後、磁場が成長し、磁場のエネルギーが対流の運動エネルギーの約 4 倍になったところで磁場のエネルギーは飽和した。

３．対流と磁場の構造

図 1 は対流が発生し、十分に飽和した後の渦度場の様子を示している。この図の可視化には次の章で説明する Armada を用いた。この図が示すように対流は薄いシート状のプルームの集まりになっている。赤道面上の流れの構造を詳しく見ると、経度方向に薄い構造をもったジェット状の上昇流と下降流が交互に並んでいる。ある時刻のスナップショットを見ると、ジェット状の流れが内核から離れるにつれて枝分かれする構造が見える。枝分かれの結果、経度方向のジェット流の幅は回転軸からの距離によらずほぼ一定になっている。この図の子午面断面が示すように渦度場（流れ場）は回転軸方向にほぼ一様になっている。この 2 次元性は強いコリオリ力の影響である。

一般に、回転球殻中の熱対流運動は円柱状の対流胞の集まりとして組織化されると考えられている。実際、エクマン数が 10^{-5} 以上の計算機シミュレーションでは円柱状の対流胞が生成される。一方、エクマン数が 10^{-6} になると対流が円柱状ではなく、プルーム状になることは、隅田による水を使った回転（半）球殻対流の一連の実験によって示されていた。我々の計算機シミュレーションは彼らの実験結果を確認するものである。

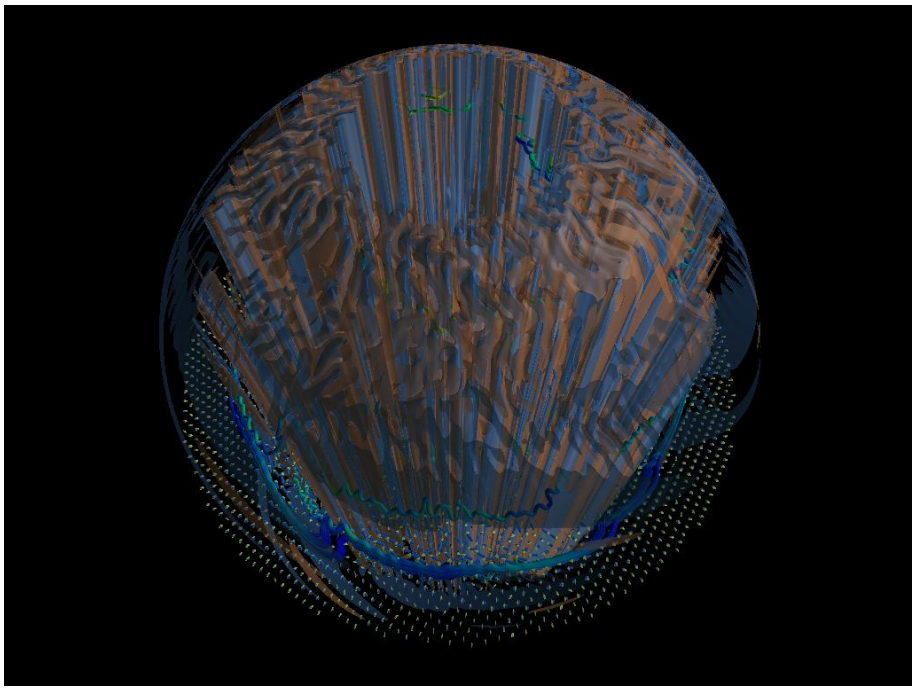


図１： シート状対流構造

エクマン数が 2.3×10^{-7} の対流構造は、これよりもエクマン数（粘性率）の高い回転球殻対流とは異なる構造を持つ。自転軸方向にまっすぐに伸びた薄いシート状のプルームが多数集まっている。

このシート状の対流構造はダイナモ効果をもつ。磁場のエネルギーが作られる場所を調べたところ、磁場が生成される場所は内核から上昇する細いプルームの中に局在することがわかった。そして磁場が生成される領域の周りを取り囲むようにして電流が螺旋状に流れている。このような螺旋型の電流構造は普遍的で、電流の場は螺旋構造の集合体として組織化されている。多数の点から出発させた電流場の力線、つまり電流線を描くと、電流場は多数の螺旋型の構造の集合体となっていることがわかった。この螺旋型電流のそれぞれの中心部分で、螺旋を貫くようにしてほぼまっすぐな磁力線が上昇プルームの中に伸びている。その周囲の流れ場を詳しく見たところ、この磁力線はプルームの流体が内核近くで加熱され、浮力によって加速しなが

ら上昇するところで引き延ばされていることがわかった。つまり上昇プルームが磁力線を半径方向（回転軸から離れる方向）に引き延ばすことで磁場のエネルギーが生成されている。

４．可視化手法

この高解像度ダイナモシミュレーションでは、大量のデータが生産されるので、ポストプロセスとして可視化を行うことは効率が悪い。そこで我々はインヤン格子で定義されたデータをレイキャスティングで直接可視化する並列可視化プログラム Armada を開発し[2]、本年度も引き続きその拡張をおこなった。Armada はグラフィックス関係のハードウェア（GPU）を一切使用せずに描画するので、HA8000 システムを含めたスーパーコンピュータ上でも動作する。このプログラムは並列化されており、並列化には、ノード間は MPI、ノード内は OpenMP を用いている。可視化機能は、スカラーデータの可視化には、等値面・カラースライス・ボリュームレンダリング、ベクトルデータの可視化には、流線・矢印表示が使用できる（図 2、図 3)。

Armada は一度の実行で、多数の可視化画像を生成するようにデザインされている。つまり、一度の実行で多数の視点、多数の可視化パラメータ（等値面レベル、流線の出発点など）による可視化画像を出力できる。例えば、等値面とスライスをそれぞれ 100 個の視点から描画することができる。また可視化機能も単独ではなく、等値面＋スライスなど組み合わせて使用することができる。

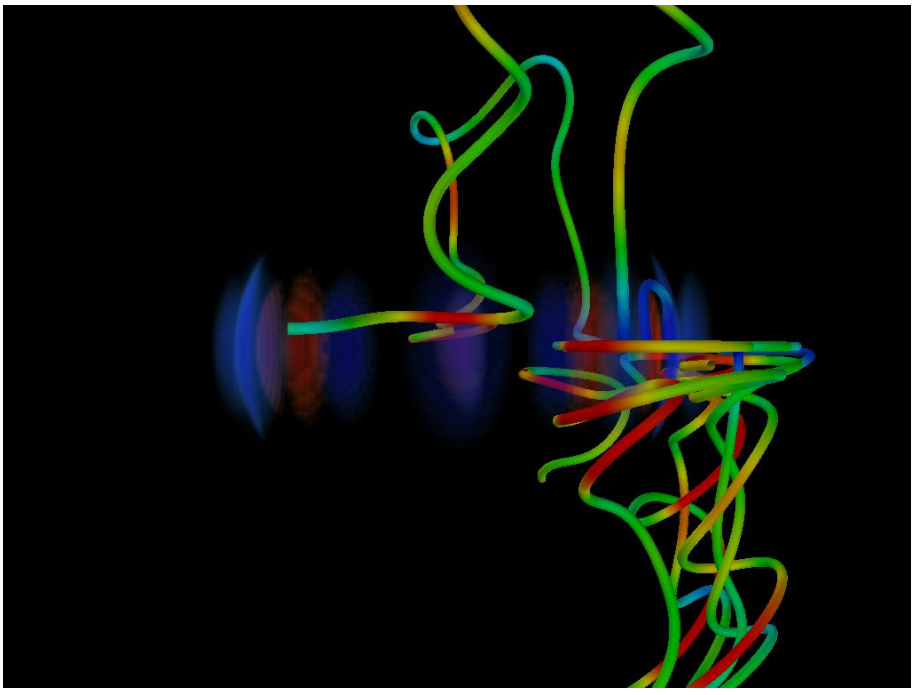


図 2： Armada で描いた流線と渦度のボリュームレンダリング

ボリュームレンダリングにより渦度の自転軸方向成分を可視化している。流線はその構造がわかりやすいようにチューブ状に表示している。この図はレイキャスティングアルゴリズムに基づくソフトウェアレンダリングによって HA8000 システム上で計算されたものである。

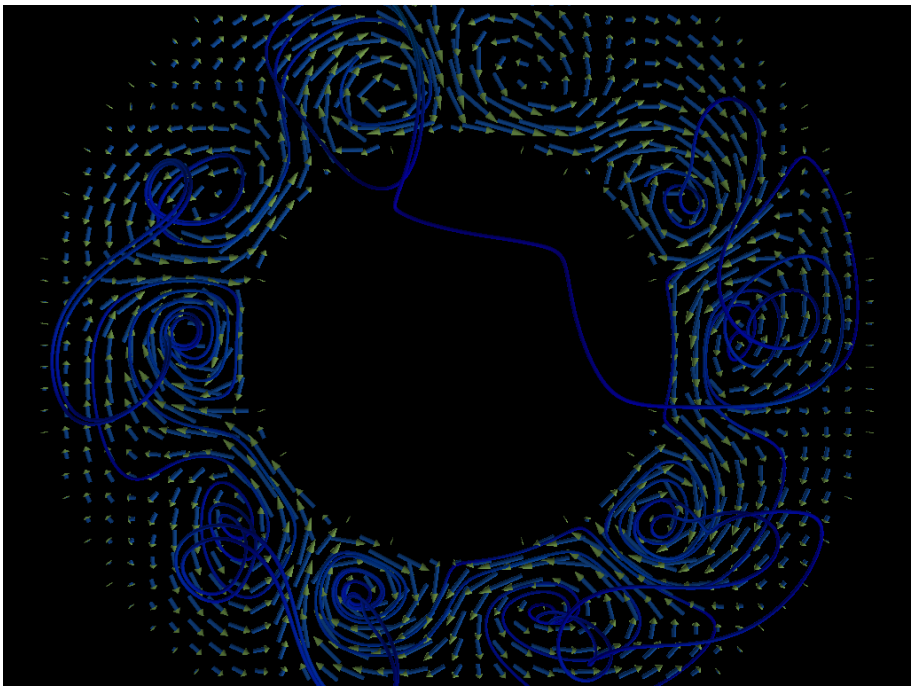


図 3： HA8000 上で Armada を用いて描いた流線と矢印による流れベクトル

矢印の向きと長さがそれぞれの位置における流れベクトルを示している。これらの矢印もポリゴンベースの（OpenGL を用いた）ものではなく、レイキャスティングで計算し、可視化したものである。

我々は、このプログラムで、大量の可視化画像を生成・出力し、解析する予定であるが、解析の途中では、いろいろな可視化手法を組み合わせて可視化した画像（例えば、等値面とスライスを同時に描画した画像）が必要になることが考えられる。このプログラムには、既に述べたように可視化手法を組み合わせて使用する機能もあるが、すべての組み合わせを出力するのは効率がよいとは言えない。そこで我々は、可視化画像と同時に Z 値をファイルに出力する機能を Armada に組み込んだ。これにより、描画済みの複数の画像を合成することができるようになった。複数の画像が合成できるので、例えば、等値面、スライス、流線、それぞれ単独で可視化した画像があれば、それらを合成することにより、等値面＋スライス、等値面＋流線の画像が得られることを意味する。

Z 値とは、（色のついた）ピクセルの視点からの距離である。このデータがあると、複数の画像を前後の矛盾なく重ね合わせることができる。重ね合わせる方法は単純である。視点や視線方向が同じ画像を 2 枚重ね合わせるとき、それぞれの Z 値を比較して、近い方のピクセルの色を新たな画像のピクセルの色とすればよい。手前に物体があると、その後ろに何か物体があっても隠れて見えないということを意味する。これは、Z バッファ法あるいはデプスバッファ法と呼ばれ、コンピュータグラフィックスの世界で広く使われている手法である。2 枚の画像を重ね合わせた後、新たな画像の Z 値のデータを作れば、さらにその画像と別な画像と重ね合わせることも可能である。本プログラムでは、Z 値は 4 バイトの実数を用いている。また、Z 値のデータは、ランレングス・エンコーディングを用いて圧縮したあと出力する。ただし、この手法では、可視化画像に、ボリュームレンダリング等、半透明な可視化オブジェクトが含まれている場合は、合成できない。

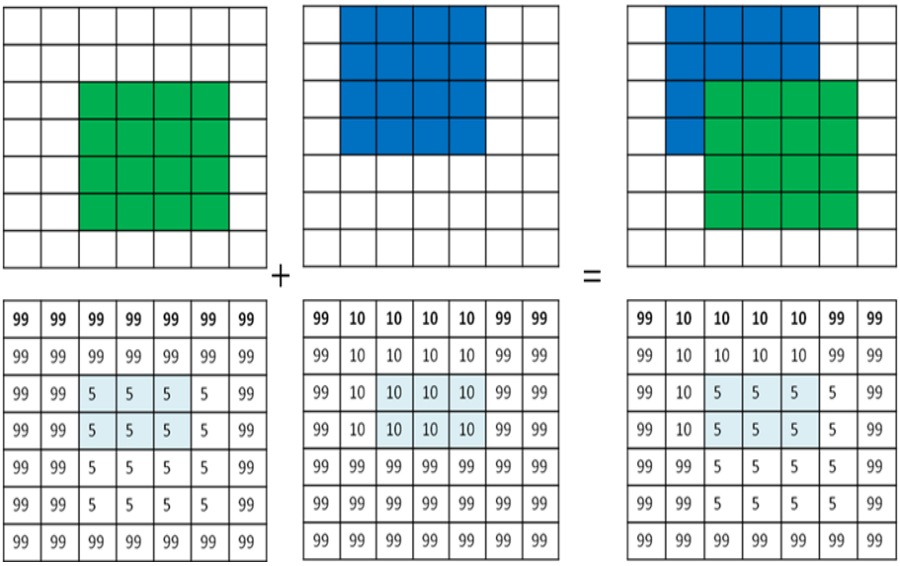


図 4： Armada に組み込まれたソフトウェア的 Z バッファ法

上はピクセルの色、下は Z 値を表す。合成するさい、水色にした部分の Z 値が、オブジェクトの前後関係を調べるために使用された。また、簡単のため、この図では、Z 値の最大値は 99 とした。

５．まとめ

HA8000 システムで行った地球ダイナモシミュレーションとそのデータ可視化の概要を述べた。地球ダイナモシミュレーションは本質的に高解像度が要求される計算である。我々はインヤン格子を利用した高並列化ダイナモシミュレーションコードを開発している。地球ダイナモシミュレーションの難しさの一つはその可視化である。解像度が高いため、出力データサイズが大きく、また、計算される現象も複雑な空間構造をもっているため、ポストプロセスとしての可視化は効率的でない。そこで我々は計算しながら可視化を行う方法を模索しつつあり、そのための第一歩として、我々は並列計算機上で実行する並列レンダリング可視化プログラム Armada を開発している。

参　考　文　献

[1] Akira Kageyama and Tetsuya Sato, “Yin-Yang Grid”: An Overset Grid in Spherical Geometry. Geochmi. Geophys. Geosyst. vol.5, doi:10.1029/2004GC000734 (2004)

[2] 陰山聡、大野暢亮、HA8000 システムでの地球ダイナモシミュレーションと可視化、スーパーコンピューティングニュース Vol.11, No. Special Issue 2, pp.33-42 (2009)

前処理行列の改良による反復法ソルバーの高速化について

細井聡　　鷲尾巧　　岡田純一　　久田俊明*

1　はじめに

我々は，マルチスケール・マルチフィジックス心臓シミュレータを開発している。そのシミュレーション時間の大半をミクロ部（心筋細胞）の計算が占め，さらにその中でもソルバー部の占める割合が大きい。このようなわけで，ソルバーの高速化はマルチスケール・マルチフィジックス心臓シミュレータにとって極めて重要である。

GMRES[1] を始めとする Krylov 部分空間法の収束性に関する問題点は，生成される Krylov 部分空間が初期誤差ベクトルの低周波成分を含むようになるまで収束が非常に緩やかであることである [2]。そこで，係数行列 A の小さな固有値に対応する反復解の誤差を素早く除去して反復回数を削減することを考える（前処理行列 M が適用される場合は AM^{-1} を係数行列と見なす)。

一方，非線形問題においては，連立 1 次方程式を繰り返し解くが，充分短期間にはその左辺の係数行列は大きくは変化しないと考えられる。そこで，既に解いた方程式から得られる情報を用いて前処理行列を改良し，新規にこれから解く方程式に対する反復回数を削減することを試みる。GMRES の場合

$r_0 = b - Ax_0;\quad \beta = \|r_0\|_2;\quad v_1 = r_0/\beta$
for　$j = 1, \ldots$
　$w_j = M^{-1}v_j$
　$v_{j+1} = Aw_j$
　for　$i = 1, j$
　　$h_{i,j} = v_{j+1}^T v_i;\quad v_{j+1} = v_{j+1} - h_{i,j}v_i$
　endofor
　$h_{j+1,j} = \|v_{j+1}\|_2;\quad v_{j+1} = v_{j+1}/h_{j+1,j}$
　if (収束条件を満たす) $m = j$ として j ループを終了
endfor
$V_m = [v_1, \cdots, v_m],\quad \bar{H}_m = \{h_{i,j}\}_{1\le i\le j+1, 1\le j\le m}$とする
$\|\beta e_1 - \bar{H}_m\|_2$を最小にする y を求める
$x = x_0 + M^{-1}V_m y$ が解となる

図 1　通常の前処理付きフル GMRES

* 東京大学新領域創成科学研究科人間環境学専攻

- Hessenberg 行列
- 直交基底ベクトル
- 固有値
- 固有ベクトルの張る空間

などを有効に使って低周波モードの素早い除去が可能である [3][4][5][6]。また，一度に 1 つの方程式だけしか解くことができないので，いわゆるグローバル GMRES[7] のような手法は使えない。本原稿では，図 1 に示すような前処理付きフル GMRES の前処理行列の改良を試みる。また、対象は制約条件付きの問題であるので、係数行列は負定値となり通常の ILU 前処理では解くのが困難な問題である。以下、 2 章では基底ベクトルを用いた前処理行列の改良手法とその効果について述べ， 3 章では粗いグリッドを用いた前処理行列の改良とその評価を行ない，最後にまとめを行なう。

2 基底ベクトルを用いた前処理行列の改良

ある部分空間基底ベクトルを用いて，元の前処理行列 M を $\tilde{M}$ に改良することを考える。ここで，ある部分空間とは $(AM^{-1})^T AM^{-1}$ の小さな固有値で張られる部分空間である。具体的には，図 2 のように通常の GMRES（図 1）を呼び出す前に (a)(b) の処理を行ない，GMRES のメインループ（j ループ）中の $w_j = M^{-1}v_j$ を $w_j = \tilde{M}^{-1}v_j$（図 2(c)）に変更する。ここで，$\tilde{M}$ に相当する処理を図 3 に示す。これは，陽に $\tilde{M}$ という行列を生成するわけではないが図 2(c) において v_j から w_j を生成する際に元の前処理行列 M の代わりに $\tilde{M}$ を適用したと見なすことができる。以上が，前処理行列を改良するという意味である。

なお，前処理行列を改良した効果を評価する指標として以下の 2 つを用いる。

$$\text{反復回数比} = \frac{\text{改良した前処理行列}\tilde{M}\text{を用いた場合の反復回数}}{\text{元の前処理行列 } M \text{ を用いた場合の反復回数}} \tag{1}$$

$$\text{スピードアップ} = \frac{\text{元の前処理行列 } M \text{ を用いた場合の実行時間}}{\text{改良した前処理行列}\tilde{M}\text{を用いた場合の実行時間}} \tag{2}$$

反復回数比の値が小さい程，反復回数が減少したことを表す。

2.1 実装についての補足

- 図 3(c_2): $W = AM^{-1}\hat{U}$（ただし $\hat{U} = [\hat{\mu}_1 \cdots \hat{\mu}_L](1 \le k \le L)$）を計算し
$$W^T W[\alpha_1 \cdots \alpha_L]^T = W^T v_j \tag{3}$$
を満たす $\alpha_k(1 \le k \le L)$ を求める。$W^T W$ を GMRES メインループ（j ループ）に突入する前に LU 分解あるいは特異値（SVD）分解しておけば，メインループ中では前進・後退代入あるいは V, w, U 行列との乗算により解を求めることができる。
- 図 3(c_3): $\hat{v}_j = \sum_{k=1}^{L} \alpha_k M^{-1}\hat{\mu}_k = M^{-1}[\hat{\mu}_1 \cdots \hat{\mu}_L][\alpha_1 \cdots \alpha_L]^T = (M^{-1}\hat{U})[\alpha_1 \cdots \alpha_L]^T$ として計算する。
- 反復法ソルバーにおいて疎行列ベクトル積は実行時間の大きな割合を占める。そこで、$M^{-1}\hat{U}$ および $AM^{-1}\hat{U}$ の計算においては複数の疎行列ベクトル積を同時に実行するような実装として高速化した。同時に実行することにより、間接参照が減り、BLAS Level3 の利用が可能となり，また，効率的な SIMD 命令の生成が可能となる。

(a) 前回の $GMRES$が m 回で収束した場合, $\bar{H}_m^T \bar{H}_m$の
$s = min(m, l)$ 個の小さな固有値に対応した
基底ベクトル$\hat{\mu}_1, \ldots, \hat{\mu}_s$を選ぶ
ただし $\bar{H}_m : (m+1) \times m$ の $Hessenberg$行列
l : あらかじめ定めた正定数
$\hat{\mu}_k = V_m \mu_k (1 \leq k \leq s)$
$\mu_k : \bar{H}_m^T \bar{H}_m$の固有ベクトル
$(b)(a)$ で選択した s 個の基底ベクトル
をこれまで収集した基底ベクトルの
集合 P に追加

$r_0 = b - Ax_0; \quad \beta = \|r_0\|_2; \quad v_1 = r_0/\beta$
$for \quad j = 1, \ldots$
$w_j = \tilde{M}^{-1} v_j \ (c)$
$v_{j+1} = Aw_j$
$for \quad i = 1, j$
$h_{i,j} = v_{j+1}^T v_i; \quad v_{j+1} = v_{j+1} - h_{i,j} v_i$
$endofor$
$h_{j+1,j} = \|v_{j+1}\|_2; \quad v_{j+1} = v_{j+1}/h_{j+1,j}$
if (収束条件を満たす) $m = j$ として j ループを終了
$endfor$
$V_m = [v_1, \cdots, v_m], \quad \bar{H}_m = \{h_{i,j}\}_{1 \leq i \leq j+1, 1 \leq j \leq m}$とする
$\|\beta e_1 - \bar{H}_m\|_2$を最小にする y を求める
$x = x_0 + M^{-1} V_m y$ が解となる

図 2　改良された前処理行列を用いたフル GMRES

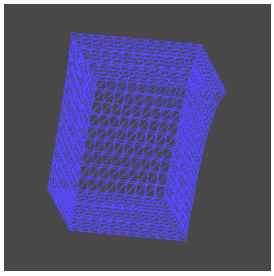


図 4　超弾性体

(c_1) これまでに収集してきた基底ベクトルの集合 P
から L 個の基底ベクトル$\hat{\mu}_k (1 \leq k \leq L)$ を選ぶ[a]
$(c_2)\tilde{v}_j = AM^{-1} \sum_{k=1}^{L} \alpha_k \hat{\mu}_k$とおき
$\|v_j - \tilde{v}_j\|_2$を最小にする$\alpha_k (1 \leq k \leq L)$ を求める
$(c_3)\hat{v}_j = M^{-1} \sum_{k=1}^{L} \alpha_k \hat{\mu}_k$
$(c_4)\tilde{r} = v_j - \tilde{v}_j$
$(c_5) M \Delta v_j = \tilde{r}_j$を解く
$(c_6) w_j = \hat{v}_j + \Delta v_j$

図 3　図 2(c) の $\tilde{M}^{-1}$ に相当する処理

[a] L 個の選び方としては 2.2 で示すように，1 回前の方程式の情報のみ用いる，過去に解いた全ての方程式の情報を用いるなど幾つかのヴァリエーションが考えられる。

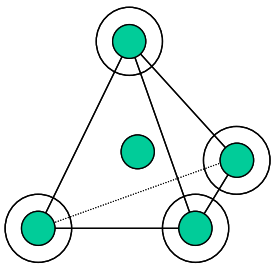


図 5　四面体有限要素

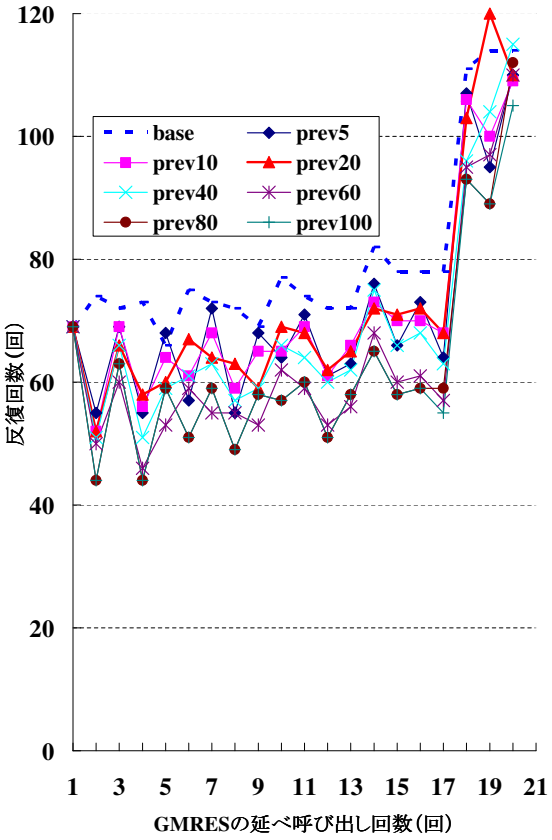


図 6　基底ベクトルを用いた前処理行列改良の効果
（1 回前に解いた方程式の情報のみを用いた場合）

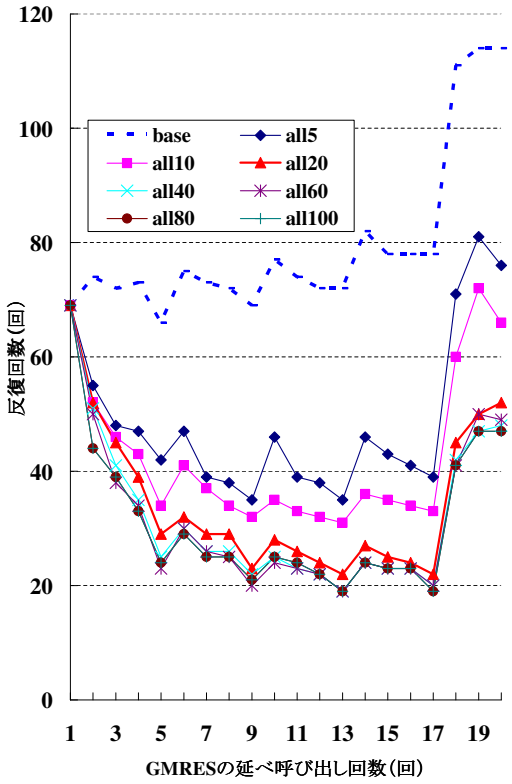


図 7　基底ベクトルを用いた前処理行列改良の効果
(過去に解いた全ての方程式の情報を用いた場合)

2.2　数値実験例

図 4 のような 3 次元超弾性体 (節点数 $16 \times 16 \times 16$ + バブル節点) の伸縮シミュレーションプログラムの線形ソルバーに対して本手法を適用した．離散化は, 4 面体 MINI 要素を用いて行った. すなわち, 4 面体の頂点に加えその重心に変位節点を追加し, 各頂点に静水圧点を加えたものである（図 5)。混合要素タイプの有限要素離散化を適用しているので係数行列は対称ではあるが静水圧の自由度の数だけ負の固有値を有し，通常の ILU 前処理では解くことが困難な問題である．本例題においては, 5 段階で境界応力を増加させたので, 全部で 5 回の非線形問題を解いている.

ここでは, 元の前処理行列M として文献 [8] により提案されたものを用い，変位自由度を消去する際に圧力対角に生じる fill-in を取り込むように ILU 分解する。そして，相対残差が 10^{-8} 以下となるまでの反復回数を比較する。

図 6 の base は元の前処理行列 M のみを用いた場合の反復回数である。一方，prev$k$ は $l = k$ とし，1 回前に解いた方程式のみの基底ベクトルを用いて前処理行列を改良した場合の反復回数で，反復回数比は大半の GMRES の呼び出しにおいて 0.7〜1.0 超であり，反復回数を充分削減できたとは言い難い。

図 7 の all$k$ は $l = k$ とし，過去 t 回解いた方程式に対して毎回 $min(m_k, l)$ $(1 \leq k \leq t)$ 個の小さな固有値

に対応した基底ベクトルを継続して収集し，合計 $L = \sum_{k=1}^{t} min(m_k, l)$ 個の基底ベクトルを用いて前処理行列を改良した場合の反復回数を表す（ただし，$m_k (1 \leq k \leq t)$ は，k 回目の GMRES 呼び出しの反復回数)。これから，以下のことが言える。

- $l \geq 20$ の場合の反復回数比に大きな差はない
- GMRES の述べ呼び出し回数が 3 回以下の場合は反復回数比がまだ充分小さくはない。この時点ではまだ充分なスピードアップが得られていないと予想される。
- 実行が進み，用いる基底ベクトル数が増えるに従い反復回数比は小さくなる。$l = 20$ の場合，GMRES の延べ呼び出し回数が 7 回目で反復回数比が始めて 0.4 以下となるが，そのためには基底ベクトルが既に 140 個必要となっている。

また，図 8 において以下のことが言える。

- first20 と first20×4 は，それぞれ最初の 1 回の GMRES 呼び出しのみの 20 個，最初の 4 回の GMRES 呼び出しのみの 20×4 個の基底ベクトルのみを用いた場合の反復回数である。いずれも次第に反復回数比は増加してしまうので，これでは最終的には充分なスピードアップが得られない。
- periodic1 は，4 回の GMRES 呼び出しにに 1 回収集した基底ベクトルを全て捨て再度収集を開始する場合である。定期的に反復回数比が 1 に戻ってしまうのが最大の問題であり，平均のスピードアップが充分に得られない。
- periodic2 は，最初の 4 回の GMRES の間基底ベクトルを収集し続け，次の 4 回は収集を中断し（ただし，これまでに収集した基底ベクトルは保持し続ける)，また次の 4 回は収集を継続 $\cdots$ とした場合である。periodic1 のように定期的反復回数比が 1 に戻ってしまうことはないし，all に比べれば用いる基底ベクトル数は少なくて済む。しかし，やはり反復回数比は増加傾向にあり，基底ベクトル数の増加も避けられないので，結局は十分なスピードアップは得られない。

以上のように，過去に解いた方程式のうちの一部の情報のみを用いれば反復回数は次第に増加してしまう。一方，全ての情報を用いれば基底ベクトル数が増大してしまい，いずれにしろ充分なスピードアップが得られないと考えられる。

一部の方程式の情報を用いるだけでは不充分で，過去に解いた方程式の情報を収集し続けなければならない理由の 1 つとして，右辺の違いの影響が考えられる。たとえば，$Ax = b$ を解いた結果得られる全ての基底ベクトルを用いて前処理行列を改良する場合を考える。もし，この前処理行列を全く同じ連立 1 次方程式に再び適用すると，丸め誤差の影響を無視すれば 1 回の反復で GMRES は収束する。しかし，右辺のみを変更した $Ax = b'$ に適用すると反復回数比はせいぜい 2/3 程度にしかならない。すなわち，1 つの連立 1 次方程式を解いた結果得られる基底ベクトルを全て用いても任意の右辺に対する誤差の低周波成分を表現するのにはまだ充分ではないと考えられる。

また，図 3 の前処理行列を改良する処理は，GMRES メインループ (j ループ) 中で毎回実行しないと反復回数をほとんど削減できないか場合によっては返って反復回数を増やしてしまう結果となる（図 9 の onlyInit)。これも，1 回だけでは初期誤差の低周波成分を全て除くことはできないためと考えられる。

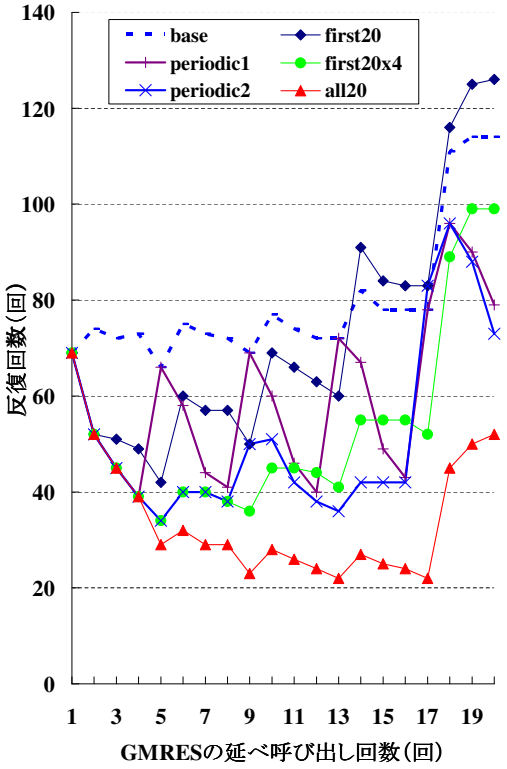


図 8　最初の基底ベクトルのみ用いる場合

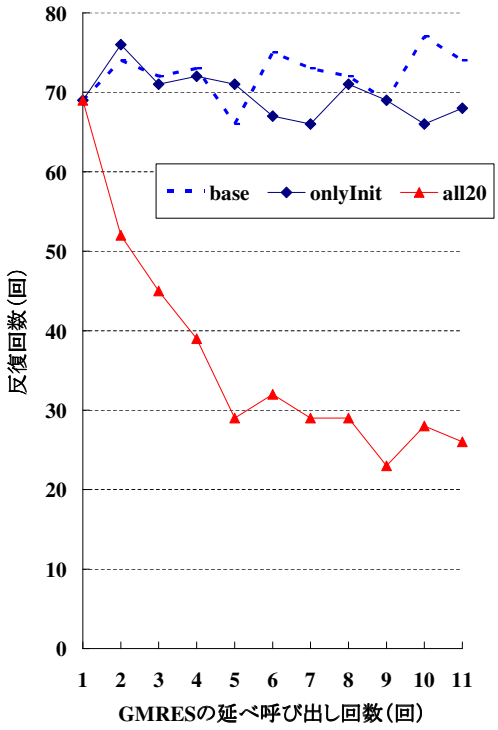


図 9　初期化時にのみ前処理行列の改良を行なう場合

2.3　基底ベクトルの線形結合を用いて基底ベクトル数を削減

2.2 の手法は，過去に解いた全ての方程式の情報を用いて始めて反復回数比を 0.3 程度にできるが，用いなければならない基底ベクトル数が多過ぎるという問題があった。そこで，極力少数の基底ベクトルで低周波部分空間を効率良く表現するために，収集した基底ベクトルの線形結合を用いることを考える。具体的には，t 回目の GMRES の呼び出しの反復回数を削減するための基底ベクトルを以下のようにして求める。なお，以下で m_k は k 回目の GMRES 呼び出しの反復回数である。

- $t = 1$ の場合，利用可能な基底ベクトルがないので通常のフル GMRES を実行
- $t = 2$ の場合，$t = 1$ 回目の GMRES の呼び出しの結果得られた基底ベクトルのうち小さな固有値に対応した $L_1 = min(m_1, l)$ 個を用いて前処理行列を改良
- $t \geq 3$ の場合，$t-2$ 回目の GMRES の呼び出しに用いた L_{t-2} 個の基底ベクトルを $\hat{\mu}_{t-2,k}(1 \leq k \leq L_{t-2})$，$t-1$ 回目の GMRES の呼び出しの結果得られる $L_{t-1} = min(m_{t-1}, l)$ 個の基底ベクトルを $\hat{\mu}_{t-1,k}(1 \leq k \leq L_{t-1})$ として，以下により t 回目の GMRES の呼び出し時に用いる $L_t = min(L_{t-2} + L_{t-1}, l)$ 個の基底ベクトルを求める。
  1. $\hat{\mu}_{t-1,k}(1 \leq k \leq L_{t-1})$ を $\hat{\mu}_{t-2,k}(1 \leq k \leq L_{t-2})$ に対して直交化
  2. $\hat{U} = [\hat{\mu}_{t-2,1} \cdots \hat{\mu}_{t-2,L_{t-2}}\ \hat{\mu}_{t-1,l} \cdots \hat{\mu}_{t-1,L_{t-1}}]$ とし $(AM^{-1}\hat{U})^T AM^{-1}\hat{U}$ の固有値を求め，そのうちの小さな L_t 個に対応した固有ベクトル $\mu_k(1 \leq k \leq L_t)$ を選ぶ

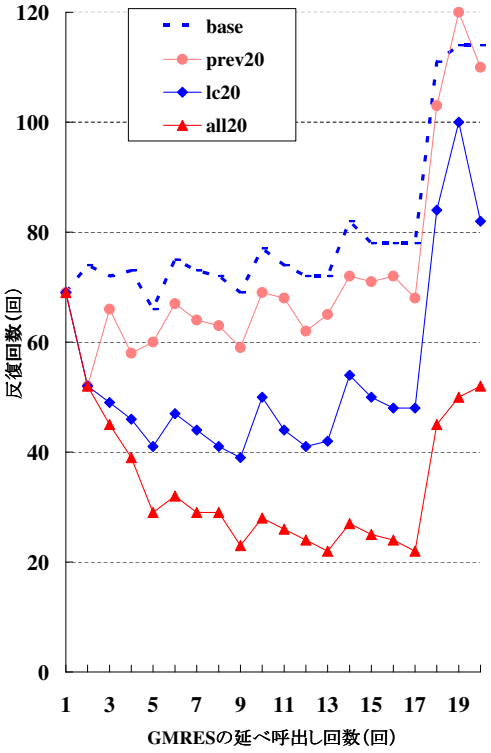


図 10　基底ベクトルを線形結合した場合の反復回数

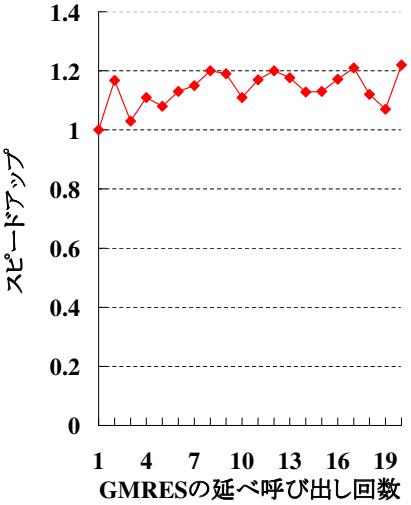


図 11　基底ベクトルの線形結合を用いた場合のスピードアップ

3. $\hat{\mu}_{t,k} = \hat{U}\mu_k (1 \leq k \leq L_t)$ を新たな基底ベクトルとし，t 回目の GMRES 呼び出しの反復回数削減に用いる

図 10 の lc20 は $l = 20$ として線形結合した基底ベクトルを用いた場合の反復回数である。1 つ前の方程式の情報のみを用いる場合（prev20）より反復回数比は充分小さくなっており，最小で 0.6 弱となっている。しかし，過去に解いた全ての方程式の情報を用いた場合（all20）に比べると反復回数比はまだ充分小さくなっているとは言えない。

次に，基底ベクトル線形結合を用いた場合のスピードアップを図 11 に示す。最大でも 20% のスピードアップしか得られておらず，実行環境により優位な差となって現れない場合もあり得ると考えられる。

3　粗いグリッドを用いた前処理行列の改良

2 では元の自由度と同じベクトルを用いたので、たとえそれらの線形結合を用いても初期誤差の低周波成分を完全に表現するには多くのベクトルが必要であった。一方、粗いグリッドを用いれば少数のベクトルでそれを表現できると考えられる。そこで、前処理行列の改良に粗いグリッドを併用することを考える [1][9][10]。マルチスケール・マルチフィジックス心臓シミュレータのミクロ部では実際の心筋細胞を模擬した図 12 のような数値細胞の動きをシミュレートする。図 12 の右端の図は 2 つの心筋細胞を連結したモデルである。図 12 の数値細胞に対して細かいメッシュ (fine メッシュ) と粗いメッシュ（coarse メッシュ）を作り，これら 2 つのメッシュを用いて前処理行列の改良を試みる。しかし，問題の規模や複雑さを考えると，実際の数値細胞に

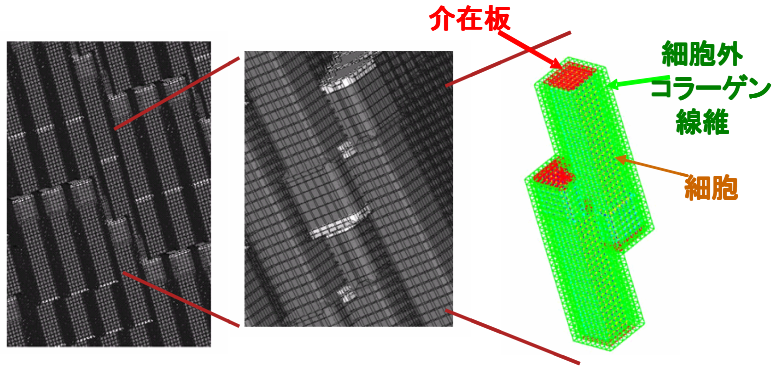


図 12　数値細胞

表 1　マテリアル情報

ギャップ・ジャンクション	$\{(x,y,z)\|2 \leq x \leq 15, 2 \leq y \leq 15, z = 1 \ or \ 76\}$
筋原繊維	$\{(x,y,z)\|2 \leq x \leq 15, 2 \leq y \leq 15, 2 \leq z \leq 75\}$
細胞外セル	上記以外の範囲の要素

最初から取り組むのは必ずしも得策ではない。そこで，まずは図 4 のような単純な直方体を 1 個の心筋細胞に見立て，六面体有限要素を用いて細かいメッシュと粗いメッシュを作成する。六面体要素の各頂点には変位節点（自由度 3），重心には圧力節点（自由度 1）を配置する。そして，このような簡単な例題に対して粗いグリッドを用いて前処理行列を改良する効果を確かめることにする。

図 4 に対して以下のような fine メッシュを作成して，心筋細胞 1 個を模擬する。

- 直方体の x, y, z 方向の長さの比を 1.0 : 1.0 : 5.0 とし，fine メッシュはこれを $15 \times 15 \times 75$ 個の六面体有限要素で分割する（節点数は $16 \times 16 \times 76$ 個）。
- fine メッシュの各有限要素に対して表 1 のようにマテリアル情報を与える。なお，ここで (x, y, z) は直方体の節点座標を表す。

3.1　coarse メッシュから fine メッシュへの変換

coarse メッシュから fine メッシュの生成は、異なるマテリアル情報を跨がないように行なう。また、coarse メッシュから fine メッシュへの変換に際しては，変位節点と圧力節点とを以下のように分けて考える。

- fine メッシュ上における変位は，coarse メッシュ上での変位から空間補完して求める。たとえば図 13 は coarse メッシュ上の節点の中間点に fine メッシュ上の節点を生成した場合に、fine メッシュでの変位を求めた様子を示している。
- fine メッシュ上の圧力は，coarse メッシュにおける圧力値の恒等写像により求める。図 14 は図 13 と同様な fine メッシュを生成した場合に、coarse メッシュ上の各有限要素の重心に位置する圧力を恒等写像した場合である。

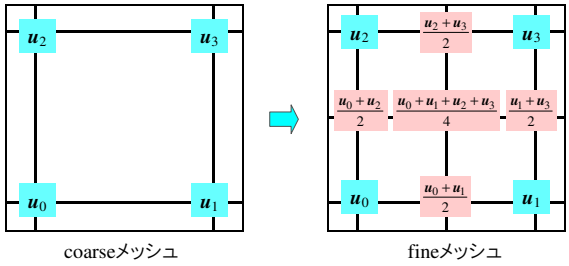


図 13　coarse メッシュから fine メッシュへの変換例（変位）

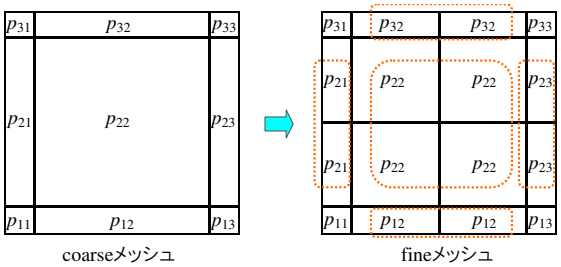


図 14　coarse メッシュから fine メッシュへの変換例（圧力）

$(a) B = AP, A_c = P^T B$ を計算
$(b) A_c$の LU 分解結果を改めて A_cとする

$r_0 = b - Ax_0;\quad \beta = \|r_0\|_2;\quad v_1 = r_0/\beta$
for　$j = 1, \ldots$
　$w_j = \tilde{M}^{-1} v_j$ (c)
　$v_{j+1} = Aw_j$
　for　$i = 1, j$
　　$h_{i,j} = v_{j+1}^T v_i;\quad v_{j+1} = v_{j+1} - h_{i,j} v_i$
　endofor
　$h_{j+1,j} = \|v_{j+1}\|_2;\quad v_{j+1} = v_{j+1}/h_{j+1,j}$
　if (収束条件を満たす) $m = j$ として j ループを終了
endfor
$V_m = [v_1, \cdots, v_m],\quad \bar{H}_m = \{h_{i,j}\}_{1 \le i \le j+1, 1 \le j \le m}$とする
$\|\beta e_1 - \bar{H}_m\|_2$を最小にする y を求める
$x = x_0 + M^{-1} V_m y$ が解となる

図 15　粗いグリッドを用いて改良された前処理行列を用いたフル GMRES

$(c_1) A_c q_c = P^T v_j$を前進・後退代入により解く
$(c_2) q_0 = P q_c$
$(c_3) \tilde{v}_j = v_j - Aq_0 = v_j - Bq_c$
$(c_4) M q_1 = \tilde{v}_j$を解く
$(c_5) w_j = q_1 + q_0$

図 16　図 15(c) の $\tilde{M}^{-1}$ に相当する処理

今 coarse メッシュから fine メッシュへの変換を P とするとき[*1]，fine メッシュ上での連立 1 次方程式の左辺の係数行列 A に対応する coarse メッシュ上での係数行列 A_c は

$$A_c = P^T A P \tag{4}$$

で与えられる。今，GMRES のメインループ中において $Aq = r$ を解く場合を考える。両辺に左から P^T を掛けて

$$P^T A q = P^T r \tag{5}$$

[*1] この P は図 14 の p_{ij} とは異なり，変位と圧力の自由度とを合わせた coarse メッシュ上のベクトルを fine メッシュ上へのベクトルへ変換する行列を表す。

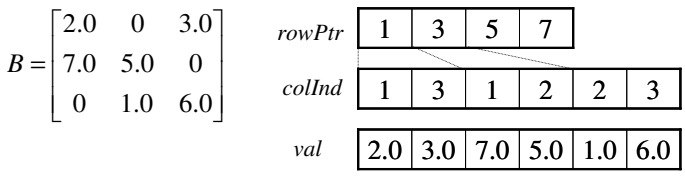


図 17　CRS 形式による疎行列の格納

ここで

$$q = Pq_c \tag{6}$$

とおけば

$$A_c q_c = P^T r \tag{7}$$

となり，これが coarse メッシュ上で解く連立 1 次方程式となる。以上を用いた前処理行列を改良するアルゴリズムの概要を図 15 および図 16 に示す。

- A_c の LU 分解は GMRES のメインループに突入する前に実行しておき（図 15(b)），メインループ内では前進・後退代入のみで coarse メッシュ上での連立 1 次方程式を解く（図 16(c_1)）。
- coarse メッシュ上での連立 1 次方程式の解 q_c を変換 P を用いて fine メッシュ上に写像する（図 16(c_2)）。

3.2　実装についての補足

- 粗いグリッドを用いた前処理行列の改良においては，coarse メッシュから fine メッシュへの変換行列 P は全実行中不変であることを仮定した。よって，P は最初に一度だけ生成すれば良い。
- A の非零要素の位置は全実行中不変。よって，$B = AP$ の非零要素の位置も全実行中不変。
  - 具体的には B の各要素は

$$B_{ij(1\leq N\leq i, 1\leq j\leq N_c)} = \begin{cases} 0 & \forall_{k(1\leq k\leq N)}\ \ A_{ik} = 0 \text{ あるいは } P_{kj} = 0 \\ \displaystyle\sum_{k=1}^{N} A_{ik}P_{kj} & otherwise \end{cases} \tag{8}$$

    となる（ここで，N は A の行数，N_c は A_c の行数)。なお，$\exists_{k(1\leq k\leq N)}\ \ A_{ik} \neq 0$ かつ $P_{kj} \neq 0$ ではあるが $\displaystyle\sum_{k=1}^{N} A_{ik}P_{kj} = 0$ となる B_{ij} は非零要素として処理する。
  - B に対していわゆる CRS 形式（図 17）の疎行列用のデータ構造を用いる場合
    - B の各行の最初の非零要素へのポインタを表す配列 $rowPtr$，および，各非零要素の列位置を表す配列 $colInd$ の領域確保と値の設定は最初に一度だけ行なえば良い。
    - 非ゼロ要素の値を保持する配列 val の領域確保は最初に一度だけ行なえば良く，次回以降は値のみ変更すれば良い。
- 図 16(c_3) における Aq_0 の計算は，$(AP)q_c$ により計算した方が効率的である。よって，AP の計算結果 B を保持しておく。

表 2　fine メッシュおよび coarse メッシュ

	変位節点座標 $\{(x,y,z)\|x \in N_x, y \in N_y, z \in N_z\}$	総変位 節点数	総圧力節点数 および 総要素数	総自由度 （総自由度比）
(a)fine	$N_x = \{x\|1 \le x \le 16\}, N_y = \{y\|1 \le y \le 16\},$ $N_z = \{z\|1 \le z \le 76\}$	19456	16875	75243 (1.000)
(b)coarse4×4×4	$N_x = N_y = \{1, 2, 15, 16\}$ $N_z = \{1, 2, 75, 76\}$	64	27	219 (0.003)
(c)coarse5×5×5	$N_x = \{1, 2, 8, 15, 16\}, N_y = \{1, 2, 9, 15, 16\},$ $N_z = \{1, 2, 38, 75, 76\}$	125	64	439 (0.006)
(d)coarse7×7×7	$N_x = N_y = \{1, 2, 5, 8, 12, 15, 16\}$ $N_z = \{1, 2, 20, 38, 55, 75, 76\}$	343	216	1245 (0.017)
(e)coarse7×7×11	$N_x = N_y = \{1, 2, 5, 8, 12, 15, 16\}$ $N_z = \{1, 2, 11, 20, 29, 38, 48, 57, 66, 75, 76\}$	539	360	1977 (0.026)
(f)coarse7×7×13	$N_x = N_y = \{1, 2, 5, 8, 12, 15, 16\}$ $N_z = \{1, 2, 9, 16, 23, 30, 37, 47, 54, 61, 68, 75, 76\}$	637	432	2343 (0.031)

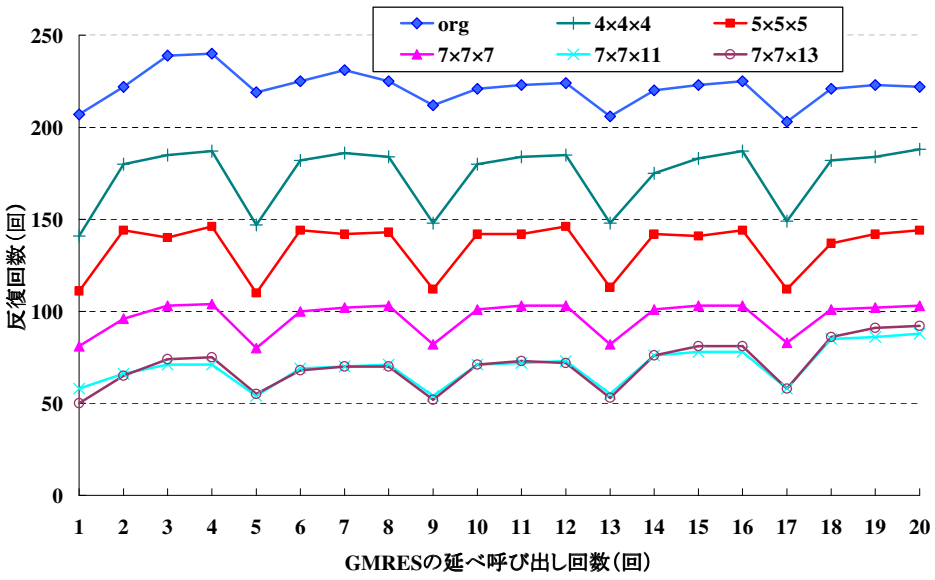


図 18　粗いグリッドを用いて前処理行列を改良した場合の反復回数

- $A_c = P^T AP$ も非零要素の位置は常に不変である。
- A_c の LU 分解と前進・後退代入には，Lapack(dgetrt+dgetrs) と後藤 BLAS を用いた。

3.3　数値実験例

2.2 と同様な伸縮シミュレーションを行ない，表 2(a) に示す fine メッシュに対して元の前処理列 M を適用した場合と，その fine メッシュと表 2(b)〜(f) のいずれか 1 つの coarse メッシュとを併せて用いて前処理行列を改良した場合の性能や反復回数を比較する。なお、表 2 中の N_x, N_y, N_z はそれぞれ x, y, z 方向の節点

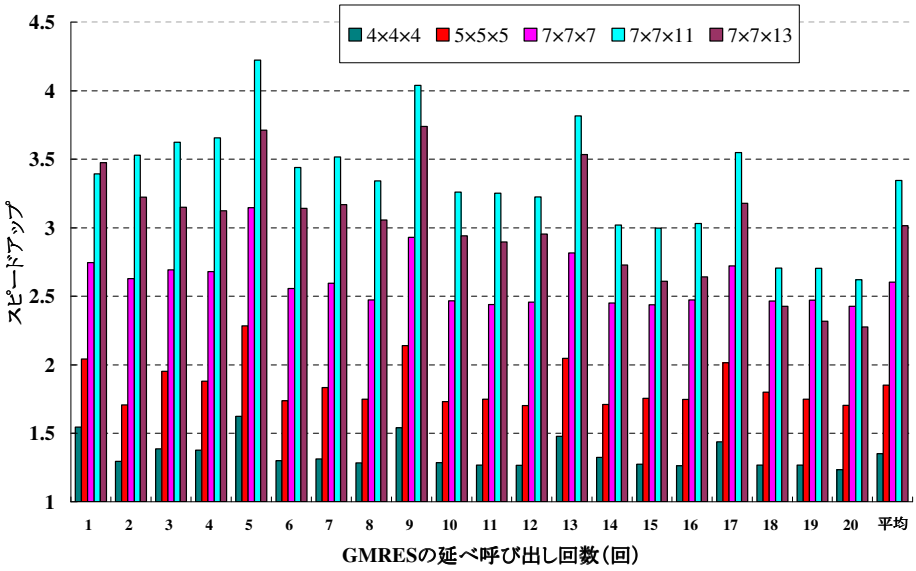


図 19　粗いグリッドを用いて前処理を改良した場合のスピードアップ

の集合を表す。また、総自由度比とは，fine メッシュの総自由度数に対する各メッシュの総自由度数である。

図 18 において，org は fine メッシュに対して元の前処理列 M を適用した場合の反復回数，4×4×4 は fine メッシュと表 2(b) の coarse4×4×4 を併せて用いて前処理行列を改良した場合の反復回数を表す（以下，5×5×5 などについても同様)。また，図 19 において，各 GMRES の呼び出しに対する 5 本の棒のうち最左は coarse4×4×4 を併せて用いた場合のスピードアップを表している（以下順に，coarse5×5×5 を併用した場合 ··· で，最右が coarse7×7×13 を併用した場合のスピードアップである)。なお，最右の 5 本は 20 回の GMRES の呼び出しのスピードアップの平均値である。図 18 および図 19 より以下のことが言える。

- 表 2 の中で最も粗いメッシュである coarse4×4×4 を用いた場合，反復回数比は 0.7〜0.8 程度であるが最大で 1.6 倍以上のスピードアップが得られている。基底ベクトルを用いた手法と比べると，反復回数比の割にはスピードアップが大きい。しかし，平均のスピードアップは 1.35 倍でありまだ充分ではない。
- 一方，coarse7×7×7 や coarse7×7×11, $7 \times 7 \times 13$ を用いた場合は平均で 2.5 倍以上のスピードアップが得られている。特に coarse7×7×11 の場合最大で 4.2 倍，平均で 3.3 倍のスピードアップとなっている。
- coarse7×7×11 と coarse7×7×13 との反復回数ほとんど変わらず，ごく僅かではあるがむしろ後者の方が少ない時もあるのに，スピードアップは常に前者の方が大きく最大で 15% 以上，平均でも 10% 以上高速である。以上の理由については現在調査中である。

4　まとめ

問題の非線形性を利用した反復法ソルバーの高速化についての検討を簡単な例題に対して行なった。

1. まず，既に解いた連立 1 次方程式から得られる基底ベクトルを用いて前処理行列を改良し反復回数の削減を試みた。過去に解いた全ての方程式から得られる基底ベクトルを収集し続けた場合，最初の数回の GMRES 呼び出し以降は反復回数比を 0.3 前後とすることができた。しかし，実行が進むに従い用いる

基底ベクトル数は単調増加してしまうので，次第に前処理行列を改良するコストも増加し充分なスピードアップが得られないようになってしまう。逆に，どこかで基底ベクトルの収集を中止してしまうと次第に反復回数比は多くなってしまうので，やはり充分なスピードアップは得られない。
繰り返し解く連立 1 次方程式の右辺の違いに対処しなければならないこと，および，非線形問題とはいえ充分時間が経てば左辺の係数行列も大きく変化してしまうと考えられることが基底ベクトルの収集を継続しなければならない理由の一因であると考えられる。

2. 次に，基底ベクトルの収集を継続する一方でその数の増加を抑制するために，得られた基底ベクトルの線形結合を用いることを試みた。一部の方程式から得られる情報のみを用いた場合と比べると，用いる基底ベクトル数を一定以下に抑えつつ，反復回数比を小さくし続けることが可能となった。しかし，基底ベクトル数に制限を設けないで収集し続けた場合に比べるとまだ反復回数比が大きく，実際のスピードアップも高々 1.2 程度であった。
3. さらに，粗いグリッドを用いて前処理行列を改良することを試した。非常に簡単な例題に対してではあるが，最も効果があった場合には反復回数比を 0.25 前後に保つことができ，また，併用する coarse メッシュにより平均で 2.5〜3 倍強のスピードアップが得られた。

今回用いた例題に比べると細胞はずっと複雑であるが，今後，粗いグリッドを用いて前処理行列を改良する手法をベースにミクロ部のソルバーの高速化に取り組む予定である。

謝辞

本研究は，JST 産学協同シーズイノベーション化事業育成ステージ「心臓シミュレータの医療への実用化研究」，並びに，平成 20 年度 T2K オープンスパコン (東大) 共同研究プロジェクトの一環として行なわれました。東京大学情報基盤センター・中島研吾先生を始めとする関係各位に感謝の意を表します。また，本論文で用いた 3 次元超弾性体伸縮シミュレーションプログラムは，東京大学大学院新領域創成科学研究科の渡邊浩志先生からご提供頂きました。ここに厚く御礼申し上げます。

参考文献

[1] Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, second edition, 2003.

[2] H. A. van der Vorst and C. Vuik. The superlinear convergence behaviour of GMRES. *J. Comput. Appl. Math.*, Vol. 48, No. 3, pp. 327–341, 1993.

[3] Ronald B. Morgan. GMRES with deflated restarting. *SIAM J. Sci. Comput.*, Vol. 24, No. 1, pp. 20–37, 2002.

[4] Andrew Chapman and Yousef Saad. Deflated and augmented Krylov subspace techniques. *Numer. Linear Algebra Appl*, Vol. 4, pp. 43–66, 1996.

[5] Kevin Burrage. On the performance of various adaptive preconditioned GMRES strategies, 1997.

[6] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel. Adaptively preconditioned GMRES algorithms. *SIAM J. Sci. Comput*, Vol. 20, pp. 243–269.

[7] Gerard. A. Meurant. *The Lanczos and Conjugate Gradient Algorithms:From Theory to Finite Precision Computations*. SIAM, 2006.

[8] T.Washio, T.Hisada, H.Watanabe, and T.E.Tezduyar. A robust preconditioner for fluid–structure interaction problems. *Comput. Methods Appl. Mech. Engrg.*, 2005.

[9] George Em Karniadakis and Robert M. Kirby. *Parallel Scientific Computing in C++ and MPI.* CAMBRIDGE UNIVERSITY PRESS, 2003.

[10] Thoman. *Multigrid Methods on GPUs.* VDM Velag, 2008.

スーパーコンピュータ若手利用者推薦（試行）

実対称固有値問題に対する多分割の分割統治法の分散メモリ型並列計算機への一実装

田村　純一

企業間取引の大規模ネットワーク構造からみた企業の特徴

大西　立顕

厳密な理論波形計算を用いた高解像度地球内部構造推定

竹 内　　希

数値モデルによる海洋微小スケールプロセスの解明

松村　義正

実対称固有値問題に対する多分割の分割統治法の分散メモリ型並列計算機への一実装

田村 純一　桑島 豊　重原 孝臣

埼玉大学大学院 理工学研究科

坪谷 怜

コンピュートロン株式会社

1　はじめに

実対称固有値問題は量子物理や金融工学などの分野で現れ，その高速な解法が必要とされている。実対称固有値問題を解く際には，前処理として実対称行列を三重対角化してから計算するのが一般的である。この処理には，数値的に安定な *Householder* **法**がよく用いられる。

実対称三重対角行列の固有値問題に対する解法は，今まで多くのアルゴリズムが提案されている。古典的な手法である**二分法・逆反復法**(以下 BII) や，行列の性質により高速化できる高精度な**二分割の分割統治法**(以下 DC2)[1] などがある。また近年，桑島らにより多分割の分割統治法 (以下 DCK)[3, 4] が提案された。DCK は DC2 を拡張し，$k > 2$ の分割数をとることが可能な手法で，DC2 では遅延が生じる行列においても固有分解を高速に計算することができる。他にも，二分法を改良した *Multiple Relatively Robust Representations(MRRR)* **法**[2] もあるが，並列化に難点があると言われている。

DCK の逐次実装および共有メモリ型並列計算機での実装は既に行われている [6]。しかし，分散メモリ型並列計算機での実装は未だ行われていない。そこで本稿では，DCK の分散並列化手法の提案と，HITACHI HA8000 と HITACHI SR11000 を用いて行った数値実験の結果を述べる。DCK のアルゴリズムには，最大で n 次行列の積をとるステップ，また必要であれば計算した固有ベクトルの再直交化を行うステップがあり，これらの計算ステップの負荷が大きくなることがある。本稿で提案する並列化手法は，これらのステップの負荷を低減することに主眼を置いている。とくに行列積は *deflation* が少ない場合に負荷が大きくなるため，そのような場合に負荷をうまく低減することを目的とした並列化手法を提案する。

以下では，2 章で多分割の分割統治法のアルゴリズムを簡単に説明し，3 章で分散並列化の手法を述べる。4 章では数値実験について述べ，5 章でまとめを述べる。

2　DCK のアルゴリズム

多分割の分割統治法 (DCK) は，n 次実対称三重対角行列 T と分割数 $k \ll n$ を入力として，T の固有分解 $T = Q\Lambda Q^T$ を与える対角行列 $\Lambda \equiv \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ と直交行列 $Q \equiv (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n)$ を出力する。

DCK のアルゴリズムは大雑把に次の 3 ステップからなる。まず，入力行列 T を，直交行列 Q_p を用いて実対角行列 D と低階数摂動 UU^T の和 (U は $n \times (k-1)$ 行列) へ相似変換する: すなわち $T = Q_p(D + UU^T)Q_p^T$。次に，この $D + UU^T$ の固有分解 $D + UU^T = Q_u \Lambda Q_u^T$ を計算する。最後に行列積 $Q_p Q_u$ を計算することで，T の全固有ベクトルを得ることができる。各ステップの詳細は以下に示す。いくつかのステップには，数値実験で参照するために `Tj` や `group` といった名前をつけている。

1. T を $T = Q_p(D + UU^T)Q_p^T$ と相似変換する。$D = \mathrm{diag}(d_1, \ldots, d_n)$ は対角行列，
   U は $n \times (k-1)$ 行列，Q_p は k 個の直交行列の直和。
   (a) T を三重対角行列の直和と摂動の和 $T = \bigoplus T_j + VV^T$ $(j = 1, \ldots, k)$ に分割する。
   (b) T_j の固有分解 $T_j = Q_j \Lambda_j Q_j^T$ を (再帰的に) 計算する (`Tj`)。
   (c) $Q_p \equiv \bigoplus Q_j$ を用いて $T = Q_p(D + UU^T)Q_p^T$ と相似変換する。
2. 固有分解 $D + UU^T = Q_u \Lambda Q_u^T$ を計算する。
   (a) U の第 j 行が全て 0 のとき，deflation により $D + UU^T$ の自明な固有対 $(d_j, \boldsymbol{e}_j)$ を除去する。ただし n 次単位行列の第 j 列ベクトルを $\boldsymbol{e}_j$ とする。
   ここで $n - r$ の自明解が得られたとして，以降のステップでは自明でない r 個の固有対を計算する。
   (b) $D + UU^T$ の非自明な固有値 $\lambda'_1, \ldots, \lambda'_r$ を求める (`eval`)。これらは T の固有値に一致する。
   (c) $D + UU^T$ の非自明な固有ベクトル $\boldsymbol{q}'_1, \ldots, \boldsymbol{q}'_r$ を求める (`evec`)。
   (d) 必要ならば固有ベクトルを再直交化する。
      i. 固有ベクトル同士の直交性を判定する (`group`)。
      ii. 直交しないベクトル同士をグループ化する (`group`)。
      iii. グループ毎に直交化する (`reortho`)。
3. 行列積 $Q_p Q_u = Q$ を計算する (`prod`)。

DCK は，分割数 k を大きくすることで直交行列 Q_p の非零要素数を減少させることができる。これによりステップ 3. の行列積の演算量を削減できる。DCK のステップの中で常に $O(n^3)$ かかり，かつ最も大きい演算量となるのはこの行列積のみである。よって，分割数 k を大きくして行列積の演算量を削減することで，全体として高速化できるという点が DCK の特徴である。

ステップ 2a. の deflation は，自明な固有対を事前に取り除き，解くべき固有値問題のサイズを小さくする操作である。deflation で得た固有対を用いると，ステップ 3. の行列積を計算せずに T の固有対が得られる。deflation が多く起こると，高速に T の固有分解が計算できる。

ステップ 2b. では，問題を対角行列と階数 1 の摂動の和 $D + uu$ の固有分解に帰着させる。これを $k - 1$ 回逐次的に解くことで $D + UU^T$ の固有値を得る。$D + uu$ の固有分解の計算は DC2 の内部計算に現れ，並列性の高さが知られている。ステップ 2c. も同様に，$k - 1 + \alpha_\lambda$ 次 (通常 α_λ は n に対して十分小さい) の行列 $\tilde{F}(\lambda)$ の核の計算に帰着させる [4] が，この計算は λ ごとに独立に行うことができる。核の計算には，逆反復法を用いる。

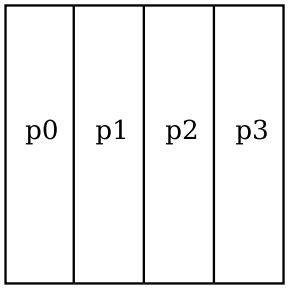


図 1　列ブロック分割の例 ($p = 4$)

ステップ 3. は，DCK のアルゴリズム中で唯一 $O(n^3)$ かかるステップである。しかし Q_p が直和型行列でありゼロ要素が多く存在することを利用して，効率良く計算できる。

以上のアルゴリズムの中で，プロセス間の通信が頻発するのはステップ 2d. およびステップ 3. であり，両ステップにおける通信負担の低減が重要になる。

3　分散並列化

一般に，プログラムの並列化には，プロセス内の複数のスレッドで処理を並行に実行する方式と，複数プロセスが処理を並行に実行する方式がある。それぞれの方式は OpenMP[8] と MPI[9] という規格で標準化されている。前者は共有メモリ型並列計算機で用いられ，並列化したい処理にコンパイラに対する指示を記述するだけで並列化できる。ただし効率はコンパイラに依存する。後者は主に分散メモリ型並列計算機向けであり，プロセス間の通信を明示的に記述する必要があり繁雑さが増すが，他方詳細なチューニングが可能である。一般に通信はネットワークを経由するため，CPU がメモリにアクセスする時間と比較してコストが高い。そこで，MPI を用いて並列化する際はできるだけ通信を避けることが望ましい。

分散メモリ型並列計算機向けに並列化するにあたり，今回は MPI のみを使用し，OpenMP によるスレッドを利用した並列化は行っていない。以降では，使用するプロセス数を p とし，各プロセスを $p_0, p_1, \ldots, p_{p-1}$ と呼ぶ。また，本実装では，分割数 k は p の倍数であることを仮定する。

分散並列化を行うにあたり，データ分散の手法はスケーラビリティや効率に大きく関わるため重要である。今回は，アルゴリズム中で演算量最大のステップ 3.(行列積) と並列化手法が自明ではないステップ 2d.(再直交化) を分散並列化するため，行列の列ブロック分割を選択した。

列ブロック分割とは，行列をベクトルの並びと見て，あるプロセスがいくつかの隣接するベクトルをまとめて保持することである。図 1 のように，n 次行列を p 個のプロセスが分散して持つ場合，各プロセスは n/p 本の隣り合うベクトルを持つ。DCK では，各プロセスは出力する直交行列 Q を列ブロック分割で保持する。このデータ分散方式に合うよう，プロセスグリッドは $1 \times p$ としている。

入力の T と出力の Λ は $O(n)$ 領域であり，全プロセスが保持する。

DCK のアルゴリズムにおいて，1b.(T_j の固有分解)，2b.($D+UU^T$ の固有値)，2c.($D+UU^T$ の固有ベクトル)，2d.(固有ベクトルの再直交化)，3.(行列積 Q_pQ_u) の 5 つが主要な計算ステップである。そこで，以下では各ステップにどのような分散並列化を行うかを述べる。

■T_j の固有分解　ステップ 1b.(T_j の固有分解) を T_j ごとにプロセスに割り振ることで並列化する。各プロセスは k/p 個の T_j の固有分解を行うが，各固有分解は逐次版 LAPACK の DC2 を用いて計算する。

入力行列 T を全プロセスが保持しているため，このステップ中に通信は必要ない。T_j の固有分解が全て終わると，通信を行い D,U を全プロセスで共有する。

■$D+UU^T$ の固有値　ステップ 2b. は実対角行列と階数 1 の摂動の和 $D+uu^T$ の固有分解を繰り返し逐次的に解くことで計算できる。DC2 の内部計算に現れる $D+uu^T$ の固有分解は，並列性が高いことが知られており，この計算を並列化することで，本ステップを並列化している。

$D+UU^T$ の全固有値を求めたら，固有値を昇順に整列して全体に放送する。これは，固有ベクトル計算後の再直交化の効率を上げるための準備である。

■$D+UU^T$ の固有ベクトル　ステップ 2c. では，$D+UU^T$ の固有値 λ に属する固有ベクトルを $\tilde{F}(\lambda)$ の核を利用して計算する。核の計算は逆反復法による。各固有ベクトルは固有値ごとに独立に求められるため，各プロセスに割り振ることで並列化する。Q_u を列ブロック分割形式でデータ分散するため，プロセス p_j は $\boldsymbol{q}'_{jn/p+1},\ldots,\boldsymbol{q}'_{(j+1)n/p}$ のみを計算する。

固有ベクトル計算時には正規化前の長さとレイリー商を保存しておき，全固有ベクトル計算後にそれらを全プロセスで共有する。これは固有ベクトルの直交性検査 (後述の簡易検査) に用いられる。

以上の 3 つのステップは，各ステップの持つ並列性を利用して分散並列化しており，手法としては自明なものである。残りの再直交化と行列積計算に関しては，手法が自明ではないため，詳しく述べる。

3.1　再直交化

再直交化はベクトル間の直交性の検査，非直交グループの構成，直交化計算の 3 つのステップからなる。本節では，提案する分散並列化手法をステップごとに述べる。

3.1.1　直交性の検査

直交性の検査は以下の手順で行う。固有値を事前にソートし，また，列ブロック分割を用いたことで，非直交な固有ベクトルはプロセス番号が近い (または同じ) プロセスに局在化しており，その結果，後述の [内積による検査] におけるデータ通信量を抑制することができている。

固有ベクトル同士の直交性検査の結果は 2 値行列 B に保存する。行列 B は Q_u と同様に，列ブロック分割形式で各プロセスにデータを分散させ，プロセス p_j が持つ部分を $B_{:j}$ と表記する。行

列 B の (i,j) 成分 b_{ij} は，ステップ 2c. で求めた固有ベクトル $\boldsymbol{q}'_i, \boldsymbol{q}'_j$ と微小パラメータ ϵ を元に次のように計算する (具体的手順は後述)。

$$b_{ij} = \begin{cases} 0 & (|(\boldsymbol{q}'_i, \boldsymbol{q}'_j)| \leq \epsilon) \\ 1 & (otherwise) \end{cases}$$

固有ベクトルの組み合わせで要素の値が決まるため，B は対称行列となる。また非直交な組み合わせは隣り合った固有ベクトルで多く起こるため，B は対角成分近辺に 1 が多い行列となる。

行列 B の成分を計算するための，固有ベクトル同士の直交性判定は 2 段階で行う。まず，簡易検査を行い，直交性が不十分と判定された場合のみさらに詳細な内積を用いた検査を行う。内積の前に簡易検査を行うことで，出来るだけ通信を避けている。各検査は以下の通り。

[簡易検査] 固有ベクトルを求める際に得たそのベクトルの長さとレイリー商を利用して，固有ベクトルの内積を過大評価する。必要なデータは全プロセスが保持しているため，簡易検査中に通信は不要である。簡易検査の詳細は [4] に譲る。

[内積による検査] 簡易検査で直交性が不十分とされた場合，内積により直交性を検査する。内積を計算するにはベクトルデータが必要であるため，通信を要する。

以下，まず 6 プロセス用いる場合 $(p = 6)$ の例を述べたあとで，アルゴリズムとして直交性検査のステップを示す。図 2 はこの例での計算ステップの進行を示す。以降，特に明記しない限り，ベクトルは $D + UU^T$ の固有ベクトルを意味する。また $D + UU^T$ の固有ベクトルを並べた Q_u のうちプロセス p_j が計算した部分を $Q_{u;j} = (\boldsymbol{q}'_{jn/p+1}, \ldots, \boldsymbol{q}'_{(j+1)n/p})$ と表記する。まず，プロセス p_0 の持つベクトル $Q_{u;0}$ と p_2 の持つベクトル $Q_{u;2}$ との直交性を次のように調べる。

1. p_2 が，$Q_{u;0}$ と $Q_{u;2}$ の間で簡易検査を行い，$Q_{u;2}$ の各ベクトルと非直交となる $Q_{u;0}$ のベクトルの対の候補を求める。候補が無ければ，B の (1,3) ブロックに 0 を代入し，直交性の検査を終了する。候補がある場合，以下の 2.∼ 4. のステップに進む。
2. 1. で求めた非直交なベクトルの対の候補の中で，$Q_{u;0}$ の中の最初と最後のベクトルの番号 $f_{0,2}, \ell_{0,2}$ を p_2 から p_0 へ送信する。
3. p_0 は，$Q_{u;0}$ のうち $f_{0,2}$ 番目から $\ell_{0,2}$ 番目までのベクトルを p_2 に送信する。
4. p_2 は，受信したベクトルのうち非直交の可能性のあるベクトルの対に対して内積計算を行い，2 値行列 B の $(1,3)$ ブロックの値を得る。

これで p_2 は，p_0 の持つベクトルとの直交性検査を終え，2 値行列 B の一部を計算できた。同じ処理を p_0 と p_4 の間でも行うことで，図 2(a) の最上行のうちプロセス名の記載されているブロックの計算ができたことになる。また p_1 と p_0，p_1 と p_3，p_1 と p_5 の間でも同じように行う (いずれも p_1 がベクトルデータを送信する) と，図 2(b) のように，上から 2 行目を計算できる。図 2 の四角の中の文字は，その部分の直交性検査を行うプロセスを示している。上の矢印はベクトルデータの流れを示しており，背景が赤のプロセスから送信することを表す。ここで，上から 2 ブロック分の部分を構成するステップは，通信がほとんど競合せず，各プロセスの行う処理は独立なため並列に実行できると考えられる。

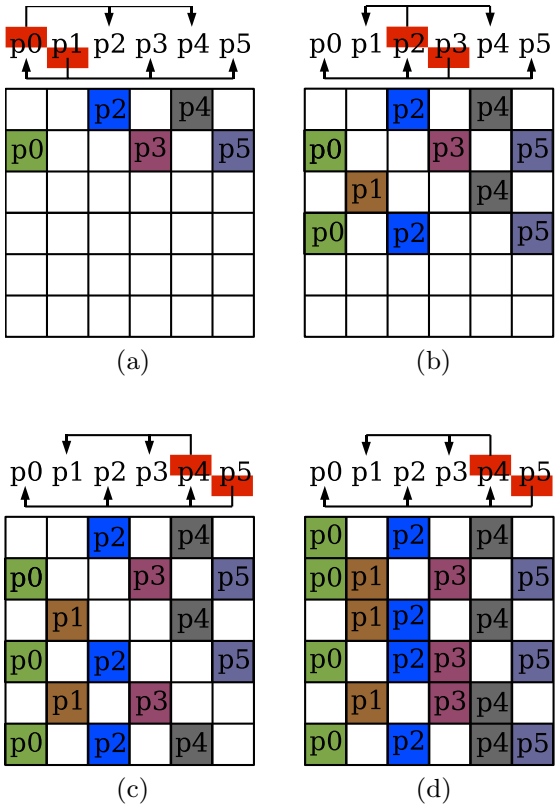


図 2　直交性検査 (行列 B の構成) の進行

2 値対称行列 B の構成は，図 2(a) から (b)，(c) へと進行する。先程述べたように，このステップは行ごとに並列に実行可能である。

最後に各プロセス p_j が持つベクトル $Q_{u;j}$ 同士の直交性を調べる (図 2(d))。固有値の並びと固有ベクトル計算の分配方法から，ある一つのプロセスが持つベクトル同士の直交性検査には内積計算が必要になることが多いと考えられる。この部分を最後に行うのは，図 2 のステップ (a) から (c) のように，通信が必要な時にその部分を計算すると遅延が発生する恐れがあり，これを避けるためである。

一般的には，以下のアルゴリズムに従い直交性検査を行う。ここで，整数 s を自分のプロセス番号とする。

1: **for** $i = 0$ to $p - 1$ **do**
2: **if** $s - i$ が正の偶数 or $i - s$ が正の奇数 **then**
3: $Q_{u;i}$ と $Q_{u;s}$ に対して簡易検査を行う
4: 簡易検査の結果に基づき，$Q_{u;s}$ と非直交な $Q_{u;i}$ の最初と最後のベクトルの番号を $f_{i,s}$, $\ell_{i,s}$ とする
5: 直交性の悪い組合せが無いならば $f_{i,s} = \ell_{i,s} = -1$ とする
6: p_i に $f_{i,s}$, $\ell_{i,s}$ を送信する
7: **if** $Q_{u;i}$ と $Q_{u;s}$ に直交性の悪い組合せがある **then**

8: p_i から $Q_{u;i}$ の $f_{i,s}$ 列目以降 $\ell_{i,s}$ 列目までを受信するまで待つ
9: 受信後，直交性の悪い組合せについてのみ内積計算を行い，直交性を判定する
10: **end if**
11: **else if** $s = i$ **then**
12: **for** $j = 0$ to $p - 1$ **do**
13: **if** $j - s$ が正の偶数 or $s - j$ が正の奇数 **then**
14: p_j から $f_{i,j}$, $\ell_{i,j}$ を受信するまで待つ
15: 受信後，$f_{i,j} \geq 0$ ならば，p_j へ $Q_{u;s}$ の $f_{i,j}$ 列目以降 $\ell_{i,j}$ 列目までを送信する
16: **end if**
17: **end for**
18: **end if**
19: **end for**
20: $Q_{u;s}$ のベクトル同士の直交性を検査する

3.1.2 グループ化

前節で作成した 2 値対称行列 B に基づき，非直交なベクトル同士をまとめたグループを構成する。グループ情報は互いに素な集合 (Disjoint set) を扱うデータ構造 Union-Find[10] を用いて管理する。

まず各プロセスは，2 値対称行列 B のうち自分が計算した部分のみを参照してグループを作る。つまりプロセス p_j が B の部分行列 $B_{:j}$ を参照する。グループ情報としては，各固有ベクトルがどのグループに属するかをグループごとに整数値で表せばよいため，n 次整数ベクトル一つの領域があれば十分である。次に，p_{p-1} は作成したグループ情報を p_{p-2} に送信する。p_{p-2} は，受信した p_{p-1} のグループ情報を自身の持つグループ情報と合併させ，それを p_{p-3} に送信する。これをプロセス番号の降順に繰り返すことで，最終的に全プロセスからのグループ情報が p_0 に集まる。p_0 は最終結果を全プロセスに送信する。

本ステップでプロセスが通信する情報は毎回 $O(n)$ であり，これを $p - 1$ 回繰り返して最後に全体に送信するため，$O(pn)$ の通信が必要となる。

3.1.3 直交化計算

グループごとにレイリー・リッツ法を用いてベクトルを直交化する。直交化は，グループ内のベクトル数があらかじめ設定した閾値以下であれば，単一のプロセスが行う。そうでなければ，全てのプロセスを用いて行う。

グループの直交化の手順を述べる。一般にグループ内のベクトルは異なるプロセスが保持しているため，単一プロセスを用いる場合は，そこにグループ内の全てのベクトルを集めて該当プロセスが直交化計算を行う。全プロセスを用いる場合は，適切なプロセスにベクトルを送信してデータ分散を行い，分散並列ルーチンを用いて直交化計算を行う。最後に，直交化前のベクトルを保持して

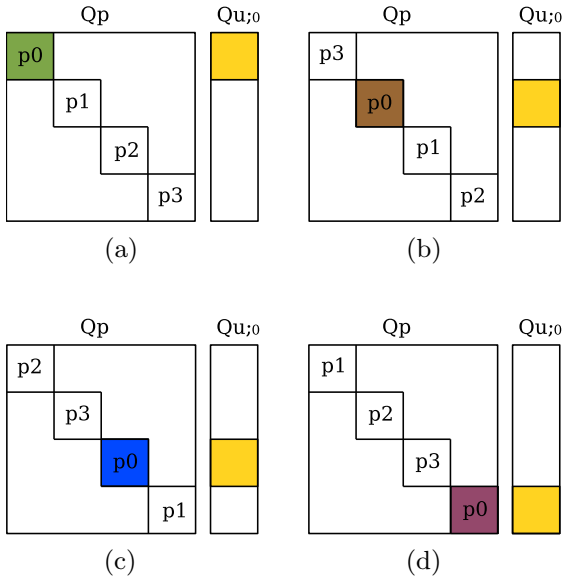


図 3　DGEMM+MPI による行列積の計算の進行 (p_0 の例)

いたプロセスに直交化後のベクトルを送信する。なお直交化計算には，単一プロセスを用いる場合は逐次版 LAPACK の `DSYEV` を，全プロセスの場合は ScaLAPACK の `PDSYEV` を使用する。

3.2　行列積

このステップは，分散メモリ型並列計算機向け線形代数ライブラリ ScaLAPACK の下位ライブラリ PBLAS に含まれる行列積を計算するルーチン `PDGEMM` を用いる方法と，プロセス間で Q_p の対角ブロック行列を MPI により通信し，各プロセスは数値線形代数ライブラリ LAPACK の下位ライブラリ BLAS の `DGEMM` を用いて逐次的に行列積を計算する方法のいずれかにより計算することができる。`PDGEMM` を用いると通信に関する事項をライブラリに任せることができるが，通信を自分で書き `DGEMM` を用いると通信を制御できるため性能を上げられる可能性がある。

まず，`DGEMM+MPI` でどのように通信と計算を行うかを述べる。概要としては，Q_p の対角ブロックを隣接プロセス間で交換し合いながら，行列積自体はプロセス内で `DGEMM` により計算を行うというものである。以下に，プロセス数 $p = 4$ のときに p_0 が対角ブロックを交換しながら行列積を行う様子を示す。図 3(a) の Q_p には対角ブロックを所持しているプロセスを記し，p_0 が所持している $Q_{u;0}$ を図の $Q_{u;0}$ の下にある長方形で示している。まず p_0 は，今所持している Q_p の対角ブロックと $Q_{u;0}$ の色で示す部分の積を取る。次に，所持している Q_p の対角ブロックを p_3 に送信し，p_1 の持つブロックを受け取る。隣接するプロセスへ対角ブロック行列を渡し，逆隣から受け取るというように，バケツリレー的にブロック行列を循環させている。今度は図 3(b) のように，対角部分と，今度は一ブロック下の色付きの部分とで行列積を行う。これを (c)，(d) と進めていき，他のプロセスでも同様に計算することで，Q_pQ_u が計算できる。各プロセスが対角ブロック行列を k 回交換するため通信量は $O(n^2/k)$，計算量は $O(n^3/(kp))$ である。

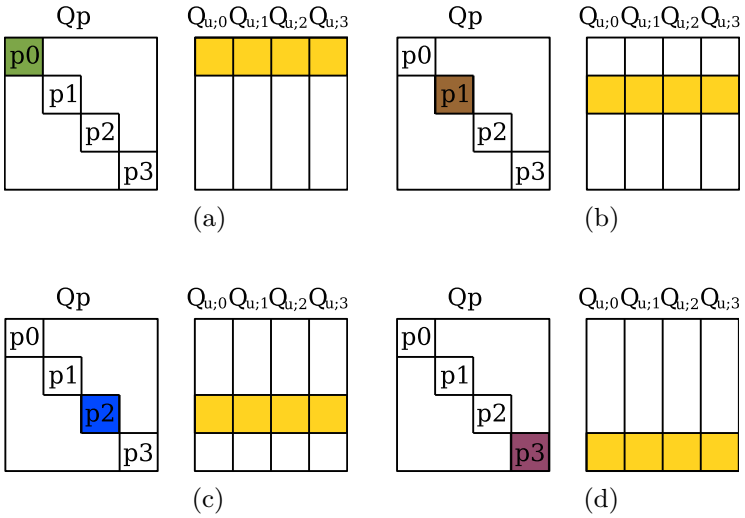


図 4　PDGEMM による行列積の計算の進行

次に，PDGEMM を用いた計算方法を述べる。図 4 に PDGEMM を用いた行列積計算 ($p = 4$ の例) の進行の様子を示す。DGEMM+MPI では対角ブロックをプロセス間で明示的に通信しながら行列積を計算したが，こちらでは通信をルーチン側に任せている。またこちらの方式でも，ブロック行列が対角に並ぶという Q_p の形式を生かすため，Q_p の i 番目の対角ブロック行列と全プロセス上に分散した $Q_{u;j}$ の部分行列 (図 4 の色付き部分で示している) との積を PDGEMM により計算する。この積を対角ブロックごとに行うことで，Q_p と Q_u の積を計算できる。

今回は，HITACHI SR11000 における次の予備実験の結果を元に計算方法を決定した。DCK のアルゴリズムに現れる n/k 次ブロックが k 個対角に並ぶ直和行列 (全体で n 次行列) と n 次密行列の積を，PDGEMM と DGEMM+MPI でベンチマークとして実装し，それぞれの所要時間を計測した。実験結果を図 5 に示す。横軸にプロセス数，縦軸に所用時間の比を取り，行列サイズ n と分割数 k を変えて得たグラフを示している。ここで，グラフは PDGEMM による行列積の計算時間に対する DGEMM+MPI による計算時間の比であり，1 より下にあるほど DGEMM+MPI の方が速いことを表す。図を見ると，どのグラフもほぼ 1 に近いかそれより下にあり，また分割数 k が大きいグラフほど下の方にある。以上より，本実装では，行列積には DGEMM+MPI を用いた。

4　数値実験

分散メモリ型並列計算機 HITACHI HA8000 上で数値実験を行い，今回提案した DCK の分散並列化の効率を確かめる。また，他の実対称三重対角固有値問題の解法と速度を比較する。

数値計算ライブラリとして LAPACK(BLAS), その分散並列版である ScaLAPACK[5](PBLAS) を用いている。ScaLAPACK と PBLAS は，線形代数ライブラリ用メッセージ交換 (通信) ライブラリ BLACS を経由して MPI 通信を行うが，DCK で通信を行う際も同様に BLACS を用いている。MPI による通信には，MPI1.2 通信ライブラリ MPICH-MX を利用している。BLAS に自前

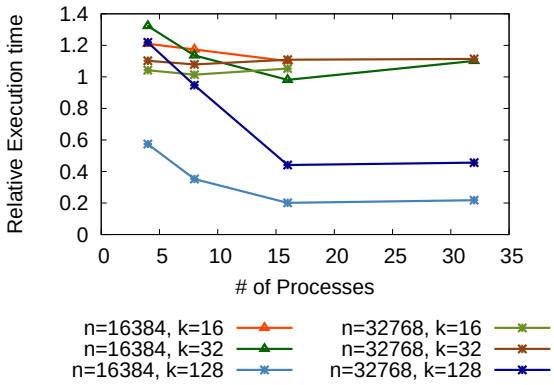


図 5　PDGEMM に対する DGEMM+MPI の相対実行時間

行列 DS	$a_j = j \times 10^{-6}$,　$b_j = 1$。deflation はあまり起こらない。
行列 QC	平均 0，分散 1 の正規乱数を要素とする実対称行列を三重対角化した行列。 量子カオス系のモデルとして用いられる。deflation はあまり起こらない。
行列 DL	$a_j \in (2, 4]$,　$b_j \in (1, 2]$ の一様乱数を持つ実対称三重対角行列。deflation が多く起こる。

表 1　実験に使用する行列

でコンパイルした GotoBLAS 1.26 を用いている以外は，いずれのライブラリも並列計算機に備え付けのものを使用している。

プログラムは C 言語で実装し，HITACHI 最適化 C コンパイラでコンパイルした。MPI を用いて通信するため，コンパイルは `mpicc -64 -Wc,-V -O4 +Op -noparallel` のようにした。

4.1　対象行列

表 1 に示す 3 種類の実対称三重対角行列を用いた。サイズは $n = 10000, 20000, 30000$ の 3 種類である。表 1 内では，主対角成分を a_j，副対角成分を b_j としている。

4.2　実験条件

分散並列化した DCK を用いて，n 次実対称三重対角行列の全固有対 $(\lambda_j, \boldsymbol{q}_j)$ を計算する時間を計測する。解の相対残差 ϵ_r，直交誤差 ϵ_o は次の式で評価する。

$$\epsilon_r = \max_{1 \le j \le n} \frac{||T\boldsymbol{q}_j - \lambda_j \boldsymbol{q}_j||_2}{||T||_2}$$
$$\epsilon_o = \max_{1 \le i \le j \le n} |\boldsymbol{q}_i^T \boldsymbol{q}_j - \delta_{ij}|$$

ただし δ_{ij} はクロネッカーのデルタである。

また BII，DC2 を用いて同様の計算を行い，DCK と速度を比較する。どちらも ScaLAPACK

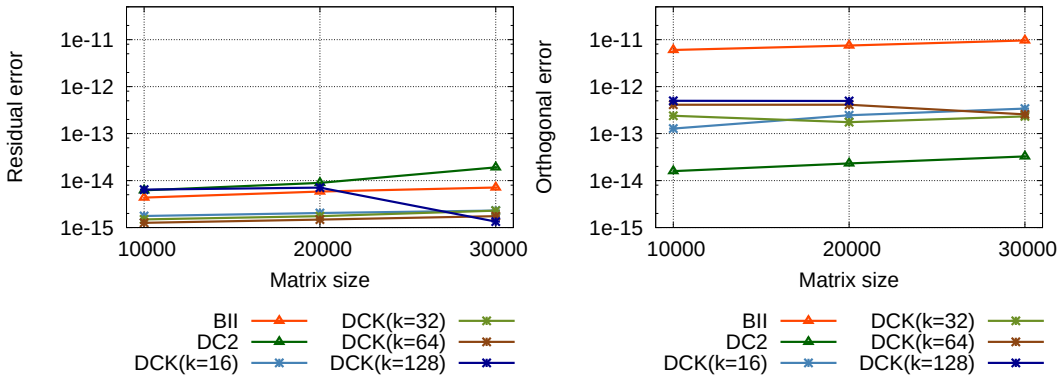


図 6　相対残差 ϵ_r と直交誤差 ϵ_o

に実装されているルーチン (pdstebz/pdstein, pdstedc) を用いている。

なお，実験時は 1 プロセスに CPU コアを 1 つ割り当てている。

4.3　実験結果

まず始めに，精度に関する実験結果を述べる。DCK は精度に関するパラメータを持っており，これが全体の性能にも影響を与えるためである。プロセス数 $p = 16$ として行列 DS を用いた。図 6 に示すように，DCK の計算精度は BII と DC2 の中間程度である。以降の実験では，特に指定のない限り，精度に関するパラメータはここで指定したものと同じである。

次に，分散並列化した DCK の評価を行う。プロセス数を増加させた時に，DCK の各計算ステップにかかる時間を図 7 に示す。30000 次の行列 DS を分割数 $k = 32$ で計算している。なお図 7 から図 12 まで，右の図は左の図を両対数にしたものである。all が全体時間を意味し，他は 2 に記載したアルゴリズムでの名前付けに従う。計算時間が最も大きい prod とそれに次いで大きい eval，evec，Tj がプロセス数の増加に応じて高速化できており，これらの計算ステップを分散並列化できていることが分かる。group と reortho は所用時間の絶対値が小さいことで並列化の効果が見えづらいため，追補実験を次に示す。

再直交化に関する処理の詳しい計測を HA8000 で取りきれなかったため，SR11000 の 1 ノード (16CPU) における結果を代わりに示す。

共有並列化を行った DCK[6] と今回提案する分散並列化を行った DCK とを比較し，グループ化 (group) と直交化計算 (reortho) の分散並列化の効率を見る。一般に，共有並列の方が通信を必要としないため並列効率は高い。表 2 に行列サイズ $n = 20000$，16 並列 (共有並列は 16 スレッド，分散並列は 16 プロセス) の元で行列 QC を用いたときの計算時間を計算ステップごとに分割数 $k = 16, 32$ について示す。項目名は 2 章のアルゴリズムでの名前付けに従う。なお再直交化の負荷を上げるため，直交精度に関するパラメータを本章の冒頭で示した値より 1 桁小さくしている。また共有並列ではスレッド並列化済みの LAPACK(BLAS) を利用している。

表 2 によると，分散の group は共有よりも約 30% 速く計算できており，3.1.1 節に述べた手法

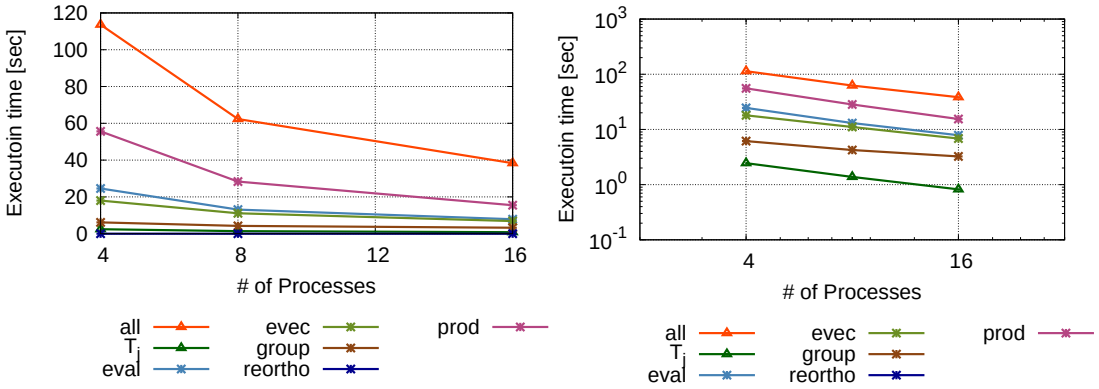


図 7　プロセス数に対する各計算ステップの所要時間 (行列 DS，$n = 30000$，$k = 32$)

k	並列化手法	`all`	`recur`	`eval`	`evec`	`group`	`reortho`	`prod`
16	共有	27.8	6.8	3.8	1.8	3.5	1.6	9.9
	分散	25.0	0.8	3.7	1.8	2.4	2.8	12.1
32	共有	23.8	5.0	4.2	2.2	3.7	3.2	5.2
	分散	21.7	0.3	3.9	2.2	2.6	4.7	6.5

表 2　共有並列と分散並列の比較 (行列 QC，$n = 20000$，単位:秒)

が効果を上げていることを示している。一方，分散の `reortho` のみを見ると，$k = 16$ では 75%，$k = 32$ では 47% の遅れが生じているが，`group` と合わせて再直交化としてまとめると共有並列に対して数パーセントの遅れとなり，十分な速度で計算できていると考えられる。

次に，分割数を変えて行列 DS の固有分解を計算したときの結果を図 8 に示す。凡例の k の値は分割数を表す。どの分割数でも，プロセス数を増やすことで高速化している。また行列 DS は deflation が起こらないため，分割数をある程度大きくした方が性能が良く，この場合には分割数 $k = 32, 64$ の時が最も高速に計算できている。

次に，プロセス数を $p = 16$ とし，行列サイズを変えて実験した結果を図 9 に示す。$k = 16$ は $n = 30000$ で大きく時間がかかっているが，図 9 右の両対数グラフを見ると他はいずれも n^3 よりは傾きが小さい。他誌で報告されている [6] ように，DCK の計算量は $O(n^3)$ であるが，deflation が少ない行列に対しては実際の並列実行時間はそれを下回ることがここでも確認できる。

次に，図 9 に他の解法のグラフを重ねたものを図 10 に示す。行列 DS の場合，分割数を大きくすると有利であり，DCK が DC2 より高速に計算できる。また，適切な分割数であれば BII より高速に計算できていることがわかる。実際の演算量としては，図 10 右で DCK より BII がやや傾きが緩やかであることから，サイズを大きくしていくと BII が有利となる可能性がある。

行列 QC に対して図 10 と同等な実験を行った結果を図 11 に示す。deflation の起こる頻度は行列 DS と似ているため，図 10 と同様の結果が得られた。行列 QC では，$\tilde{F}(\lambda)$ の最大サイズが行

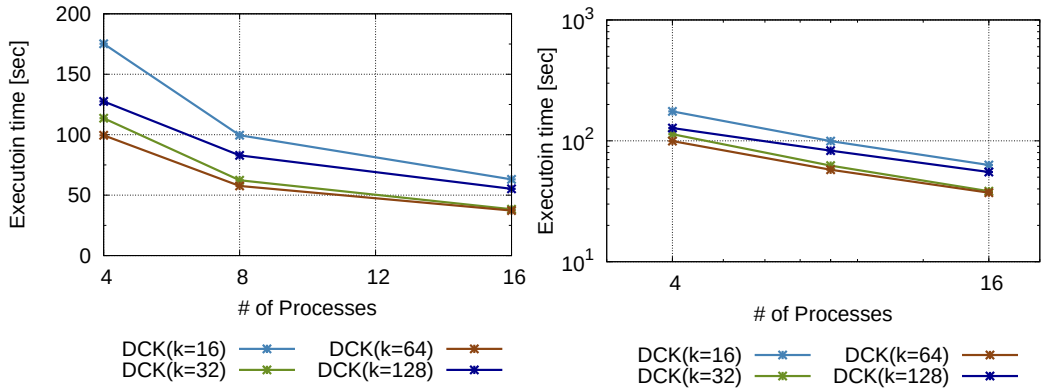


図 8　分割数毎のプロセス数に対する所要時間の変化 (行列 DS，$n = 30000$)

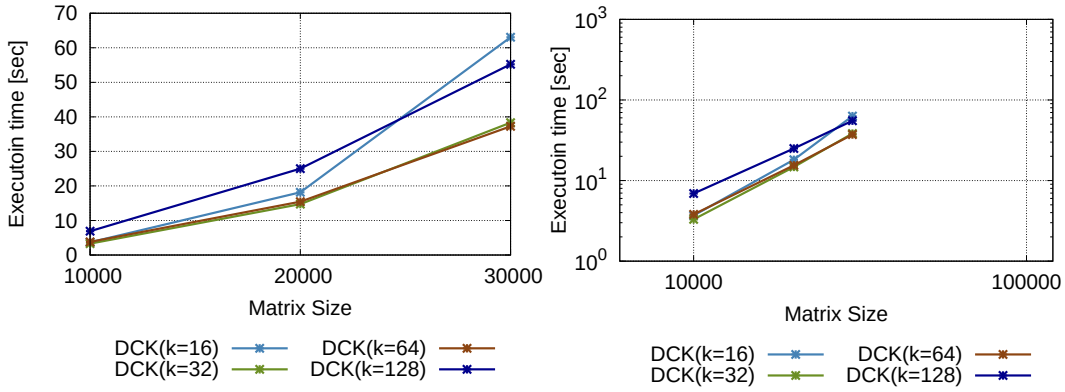


図 9　分割数毎の行列サイズに対する所要時間の変化 (行列 DS，$p = 16$)

列 DS での 1/4 程度になっていることで，固有ベクトル計算にかかる時間が短くなり，DCK で最速となる分割数が行列 DS より大きくなっていた。

行列 DL を用いた結果を図 12 に示す。この行列は上の 2 つの行列とは違い deflation が多く起こり，行列積の計算負荷が削減されるため，DC2 が最も高速となっている。また適切な分割数であれば DCK は BII と同程度の速度で計算可能である。

5　まとめ

今回，多分割の分割統治法 (DCK) の分散並列化手法を提案，実装し，HITACHI HA8000 と HITACHI SR11000 上で評価を行った。deflation が多く起こる行列に対しては二分割の分割統治法 (DC2) が有効だが，deflation が少ない行列においては DCK の方が有利となる。後者の行列では DCK のアルゴリズム中の行列積と再直交化の負荷が大きくなることがあるため，これらの負荷を低減するような手法を提案した。

DCK の主要な計算ステップのうち T_j の固有分解 (`Tj`)，$D + UU^T$ の固有値計算 (`eval`)，$D + UU^T$ の固有ベクトル計算 (`evec`) には高い並列性があり，それらを利用することで分散並列

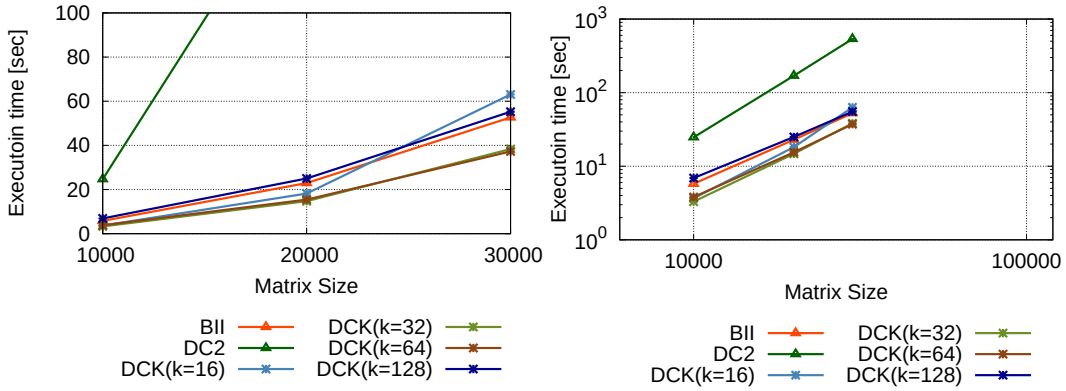


図 10　解法ごとの行列サイズに対する所要時間の変化 (行列 DS，$p = 16$)

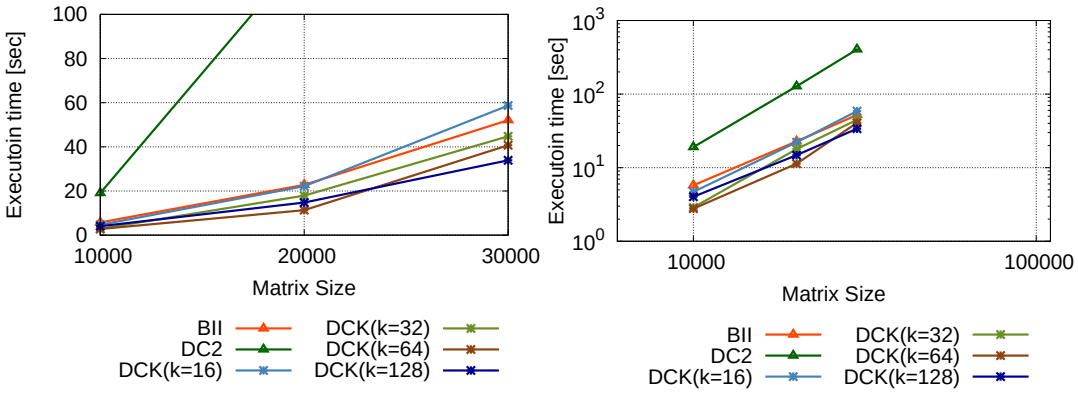


図 11　解法ごとの行列サイズに対する所要時間の変化 (行列 QC，$p = 16$)

化が行えた。一方，行列積 (`prod`) と再直交化 (`group`，`reortho`) の並列化手法は自明ではないが，本稿で提案した手法により良好な結果を得られた。

数値実験によると，deflation が少ない行列において DCK は ScaLAPACK の DC2 と比較して最短で 1/10 程度の時間で計算できており，本稿で提案した並列化手法が有効であることを示している。

なお，数値実験で他の解法と比較する際には適切な分割数を選択して行っているが，現実的にはいくつかの分割数で試してみなければ適切な分割数は分からない。そこで，適切な分割数を事前に求めることは課題の一つである。逐次ではこの問題に対して解決策が提案されている [7]。また，直交化計算 (`reortho`) に関してはまだ改善の余地がある。たとえば，本稿の実装ではグループごとに逐次的に行っている直交化を，プロセスに振り分けて行うことで小さなグループの直交化を全プロセスに分散させることができる。現在この作業は完了している。また再直交化において，グループ情報をまとめる操作を行う際の通信を木構造にすることで高速化できる可能性がある。このような再直交化の分散並列化の改善も今後の課題である。

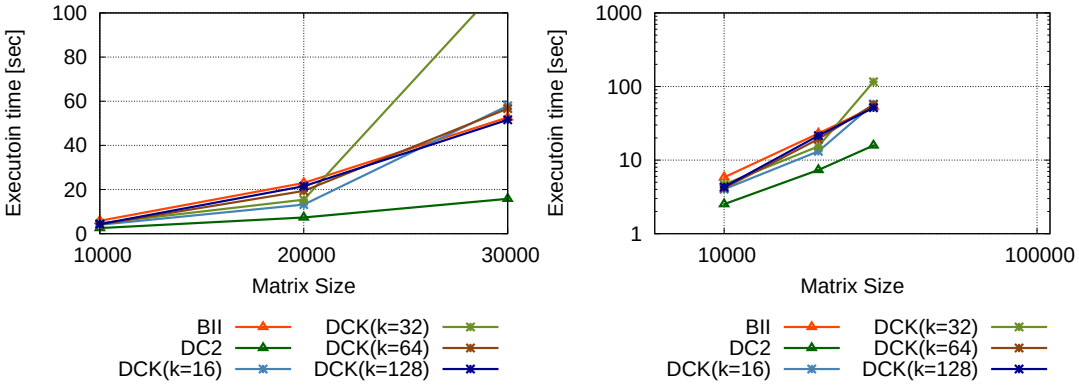


図 12　解法ごとの行列サイズに対する所要時間の変化 (行列 DL，$p = 16$)

6　謝辞

本研究は，東京大学情報基盤センターの若手利用者推薦制度の下で遂行し，HITACHI HA8000 を半年に渡り利用させていただきました。ここに謝意を表します。

7　情報基盤センターへの要望

HITACHI HA8000 では `ls` でファイルリストを表示したり，`vim` で小さなファイルを編集して保存するなどというときに，結果が表示されるまで時間がかかることがあった。そこでファイルの編集や列挙などの性能を改善してほしい。この要望に対しては，既にセンターは増強計画に基づき対策を進行しているため，センターの対処に感謝したい。

参考文献

[1] J. J. M. Cuppen, A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem, *Numerische Mathematik*, 36, pp. 177–195 (1981).

[2] I. S. Dhillon, A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem, University of California at Berkeley, Berkeley, CA(1998).

[3] 桑島豊, 重原孝臣, 実対称三重対角固有値問題の分割統治法の拡張, 日本応用数理学会, Vol.15, No.2 (2005), pp. 89–115.

[4] 桑島豊, 重原孝臣, 実対称三重対角固有値問題に対する多分割の分割統治法の改良, 日本応用数理学会論文誌, Vol.16, No.4 (2006), pp. 453–480.

[5] L. S. Blackford et. al., ScaLAPACK Users' Guide, Society for Industrial and Applied Mathematics, Philadelphia, PA (1997).

[6] 田村純一, 坪谷怜, 桑島豊, 重原孝臣, 実対称固有値問題に対する多分割の分割統治法の共有メ

モリ型並列計算機における有効性, HPCS2009 論文集, pp.97–104 (2009).
[7] 石川祐輔, 田村純一, 桑島豊, 重原孝臣, 実対称固有値問題に対する多分割の分割統治法における準最適分割数の自動決定について, 第 38 回数値解析シンポジウム 講演予稿集, pp.17–20 (2009).
[8] OpenMP ARB, http://openmp.org/.
[9] Message Passing Interface Forum, http://www.mpi-forum.org/.
[10] T.H. Cormen, 浅野 哲夫 et.al., アルゴリズムイントロダクション 改訂 2 版 第 2 巻, 近代科学社 (2007).

企業間取引の大規模ネットワーク構造からみた企業の特徴

大 西 立 顕

一般財団法人 キヤノングローバル戦略研究所，東京大学大学院法学政治学研究科

1　超並列計算による大規模経済データの分析

近年の情報技術の進展により様々な分野で電子化が進み，詳細で膨大な情報が日々蓄積されるようになってきた。その結果，超大量多様な経済データが利用可能になり，これらの分析の学術的・社会的ニーズが高まっている。伝統的な経済学は，このような大規模な実証データが存在しない時代に発展してきた。そのため，まず最初になんらかの仮定を行ない，その仮定を出発点にして理論を構築しているが，その仮定・理論の正当性は十分に検証されていない。しかし，今では現実の大規模データを用いることで，前提となる仮定の妥当性や理論の正当性を実証的に検証しながら，理論構築できる状況になってきている。物理学の概念や手法を用いて，このような大規模な実証データを科学的に分析するのが**経済物理学**である。

経済活動は売り手・買い手の間のモノ・金の交換 (相互作用) と捉えられる。相互作用があっても個々がランダムにゆらげば，全体のゆらぎは正規分布に従う。しかし，ほとんどの経済現象では常に競争が働くため相互作用が強く，全体として大きなゆらぎを伴なうため**ベキ分布**に従う。平均値±標準偏差で現象を捉えられる正規分布と異なり，ベキ分布は桁違いの値を取りうるので，平均・分散は意味をなさず，正規分布を基本にしている多くの統計手法は，厳密には適用できず有効ではない。さらに，経済現象では要素 (経済主体) の異質性・多様性が強く，物質科学の現象と異なり作用・反作用の法則が成立しないため，多体の方向つき相互作用 (**複雑ネットワーク**) を考える必要がある。したがって，統計的有意な関係性や法則性を見つけ出すには，**ノンパラメトリックな統計手法**や現実の分布を利用した**モンテカルロ法** (適切なランダムシャッフルデータとの比較) による分析が必要になる。これらは，計算量が膨大かつ並列計算に適した計算のため，スーパーコンピュータによる超並列計算が必要かつ有効である。

企業間のお金と製品・サービスの流れは経済活動の根幹に関わる現象である。本研究では，有向リンク解析 (ページランクとオーソリティ・ハブ度) とネットワークモチーフの二つの手法を用いて，企業間取引の大規模な有向ネットワークを分析する。

2　企業間取引ネットワーク

2.1　解析したデータ

東京商工リサーチ社が提供する 2005 年時点での日本企業約 100 万社についての企業属性 (売上高，申告所得，利益金，従業員数，業種，地域など) [1]と主要取引先データ (仕入先，販売先) を分析した。取引先データから，企業をノード，取引関係を有向リンクとして，各企業がどの企業と取引しているかに関する大規模な有向ネットワーク (企業間取引ネットワーク) [2]を構築し，分析した [1, 2]。有向リンクの方向は，お金の流れの向き (買い手 → 売り手) にとる。つまり，

[1]売上高，申告所得，利益金については，2003 年と 2004 年時点のデータも入っている。

[2]ただし，個人消費者，政府，海外の会社との取引関係はデータに含まれていないため，本研究は国内の企業間取引に限定した分析になる。

反対向きは製品・サービスの流れになる。ネットワークのサイズは 961,318 ノード，3,667,521 リンクになる。

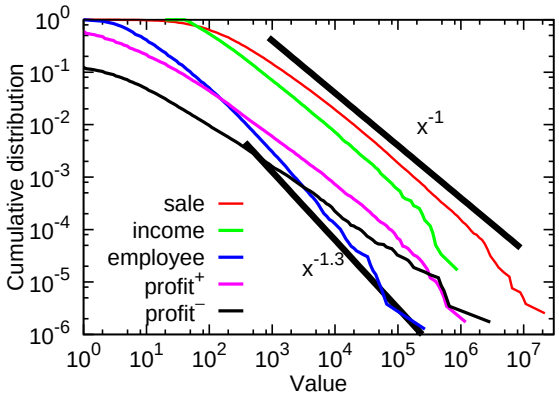
2.2 企業の規模・成長率の統計性

企業規模 x の分布は，$x > x_{min}$ でベキ分布

$$P(> x) \propto x^{-\alpha}$$

に従う (第 1 図)。最尤法と Kolmogorov-Smirnov 検定から算出した指数は　~ 1 である (第 1 表)。企業の成長率 $r = x(t+1)/x(t)$ (企業規模の前年比) は，ガウス分布よりずっと裾野が広く，ベキ分布に近い大きなゆらぎを示す (第 2 図)。

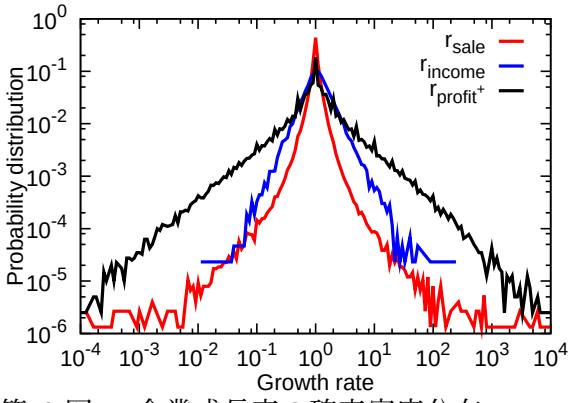
つまり，企業のダイナミクスは大きなゆらぎを伴なう複雑な系であり，現象を無相関な確率変数の単純な和として理解することはできない。そのため，現象の解明には，単一の企業そのものだけでなく，企業間の相互作用を理解することが重要になる。



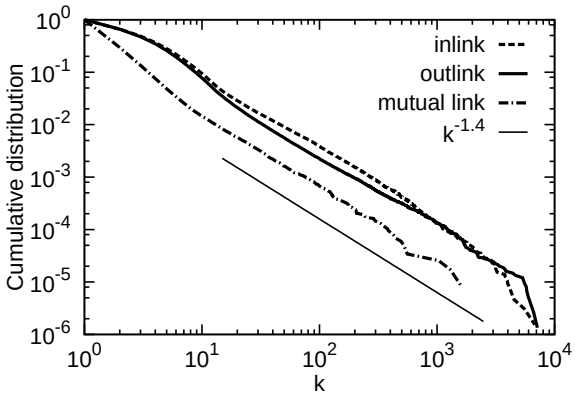
第 1 図 ： 企業規模の累積確率分布。売上高 (sale)，申告所得 (income)，正の利益金 (pro t$^+$)，負の利益金 (pro t$^-$) の単位は 100 万円。従業員数 (employee) の単位は人。

第 1 表 ： 最尤法と Kolmogorov-Smirnov 検定から算出 [3] した指数　と下限 x_{min}。

	x_{min}	
売上高	13800	1.025 ± 0.010
申告所得	424	0.967 ± 0.010
従業員数	344	1.298 ± 0.014
入リンク数	87	1.371 ± 0.025
出リンク数	85	1.249 ± 0.028
相互リンク数	34	1.350 ± 0.076



第 2 図 ： 企業成長率の確率密度分布。$r_{\mathbf{sale}}$，$r_{\mathbf{income}}$，$r_{\mathbf{profit}^+}$ はそれぞれ売上高，申告所得，正の利益金の前年比。



第 3 図 ： リンク数の累積確率分布。

2.3　ネットワークの統計性

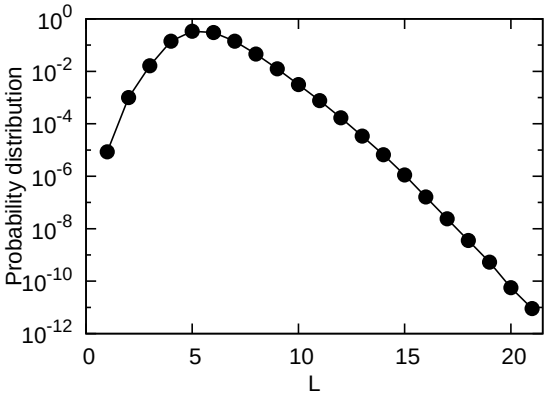
有向ネットワークでは，リンクは入リンク (◯←)，出リンク (◯→)，相互リンク (◯↔) の三種類に分けられる。入リンク数，出リンク数，相互リンク数はどれもベキ分布し [1, 2, 4, 5, 6]，このネットワークは**スケールフリーネットワーク**になっている (第 3 図)。最尤法と Kolmogorov-Smirnov 検定から算出した指数は，入リンクと相互リンクは　 ~ 1.4，出リンクは　 ~ 1.2 である (第 1 表)。

ノード間の距離 (任意の二つのノードをつなぐのに必要な最小リンク数) は指数分布し，平均距離は 5.62，最大距離は 21 リンクである (第 4 図)。

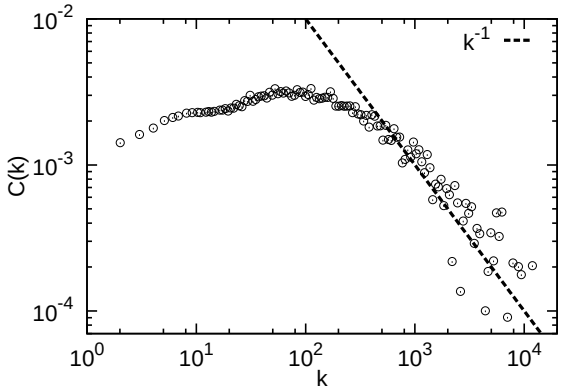
いま，リンクの方向を無視して無向ネットワークとして考えたとき，ノード i の**クラスター係数**は，そのノードを含む三角形の個数÷可能な三角形の個数，つまり，

$$C_i \;=\; \frac{2\ t_i}{k_i(k_i \quad 1)}$$

で与えられる。ただし，t_i はノード i を含む三角形の個数，k_i はノード i のリンク数である。同じリンク数をもつノードについての C_i の平均値 $C(k)$ は，$k > 500$ で $C(k) \sim k^{-1}$ の減衰を示し (第 5 図)，リンク数の多い企業ほどクラスター係数が小さくつながりが疎で，リンク数の少ない企業ほどクラスター係数が大きくつながりが密である。同様の特徴は現実の様々なネットワークでも観測されている [7]。

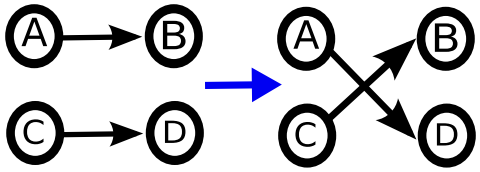


第 4 図：距離 L の確率密度分布。



第 5 図：クラスター係数のリンク数依存性。

2.4　リンク数保存のランダムネットワーク



第 6 図：リンク数を保存するようなリンクの組の入れ替え。

単にリンク数の情報だけからでは得られない特徴を抽出するため，実ネットワークを**リンク数保存のランダムネットワーク**と比較する。リンク数保存のランダムネットワークは，各企業の入リンク数 (◯←) と出リンク数 (◯→) と相互リンク数 (◯↔) を保存するようなリンクの組の入れ替え操作 (第 6 図) をネットワークがランダムになるまで繰り返すことで得られる [8]。実ネットワークとランダムネットワークで，各ノードの二体相互作用は同じになる。本研究では，

リンク数保存のランダムネットワークを 1000 個作成し，実ネットワークとの比較から統計的有意性を評価する。

2.5 並列計算の実装

解析する大規模ネットワークは疎なグラフなため，隣接リスト表現 (ノードに隣接するすべてのノードを隣接リストに列挙する) によりデータ構造を保持する。

本研究での主な計算は，同じ計算をパラメータや乱数の種を変えて何回も行うものであり，各コアが独立な計算をすればよいため，実装は容易である。MPI による並列化で 1024 コアを用いて計算を行った。

3 ページランクとオーソリティ・ハブ度

3.1 ページランク

有向ネットワークの構造は，ノード i からノード j へのリンクがあれば $A_{ij} = 1$，リンクがなければ $A_{ij} = 0$ と表現した隣接行列 A_{ij} から理解できる。大規模な隣接行列をそのまま扱うのは困難が伴う。そこで，隣接行列を確率行列に変換し，この確率行列の最大固有ベクトルとして各企業の重要度を定義したのが**ページランク**である [9]。ページランクは Web ページの重要度の指標として Google で利用されている。

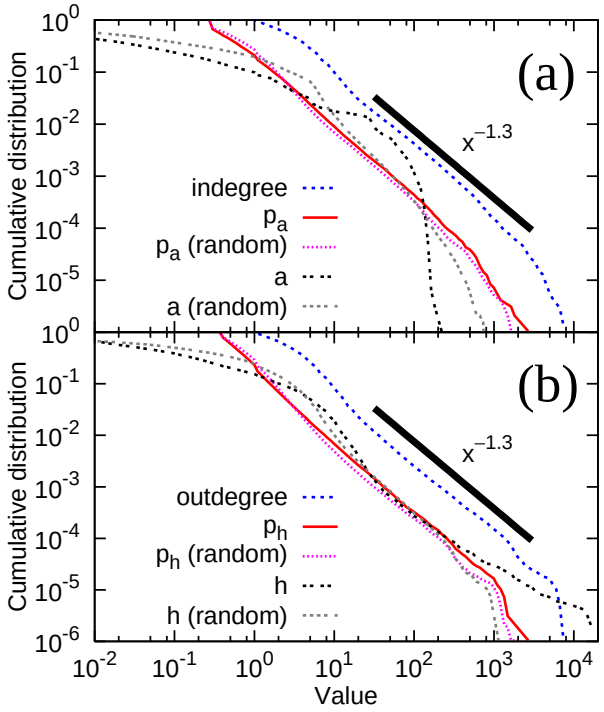
ページランクは，ランダムジャンプつきのネットワーク上のランダムウォークの定常状態 (滞在時間) として定義される。このとき，ランダムウォーカーは，確率 $1 \quad \varepsilon$ でランダムに選んだ一つの出リンクに沿って進み，確率 ε で全ノードの中からランダムに選んだ一つノードにジャンプする (ただし，出リンクがない場合は，常に全ノードからランダムに選んだ一つノードにジャンプする) ものとする。この過程はマルコフ連鎖で記述でき，確率行列の性質とペロン・フロベニウスの定理から最大固有値 1 の一意な解が保証されるので，ページランク p_a は連立一次方程式

$$x_i \;=\; (1 \quad \varepsilon) \sum_j M_{ij} x_j \;+\; 1/n$$

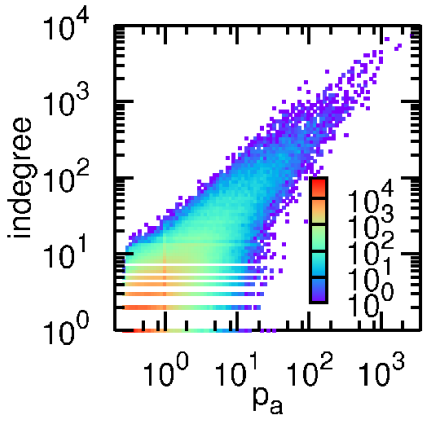
の解として求まる [10]。ただし，$M_{ij} = A_{ji}(\sum_k A_{jk})^{-1}$ (ただし，ノード j に出リンクがない場合は $M_{ij} = 0$ とする)，$p_{ai} = n x_i (\sum_j x_j)^{-1}$，$\varepsilon = 0.15$ とする。ページランクの値が大きくなるためには，単に入リンク数が多いだけでなく，自分へリンクしているノードのページランクの値が大きいことも重要になる。有向リンクの方向はお金の流れの向きにとっているので，p_a は各企業に流れるお金の流量と解釈できる。

ページランクは，ネットワークのリンクの方向を逆向きにしても同様に定義できる。このように定義したページランク p_h は，各企業に流れる製品・サービスの流量と解釈できる。

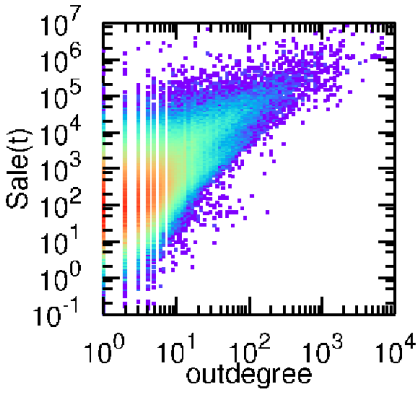
企業間取引ネットワークでは，p_a は入リンク数と，p_h は出リンク数と同じような分布をしている (第 7 図)。Kendall の順位相関係数は第 2 表のようになり，互いに強く相関している (第 8 図)。つまり，全体としてみればページランクはリンク数と同じような量になっている。



第 7 図：(a) 入リンク数 (indegree)，p_a，a と (b) 出リンク数 (outdegree)，p_h，h の累積確率分布。比較のため，リンク数保存のランダムネットワークで求めた結果も示している。



第 8 図：p_a と入リンク数の同時分布。



第 9 図：出リンク数と売上高の同時分布。

第 2 表：Kendall の順位相関係数 τ。　は p-value $< 10^{-8}$ を示す。

	入リンク数	出リンク数	p_a	p_h	a	h
出リンク数	0.23	–	–	–	–	–
p_a	0.75	0.14	–	–	–	–
p_h	0.14	0.71	0.08	–	–	–
a	0.67	0.10	0.53	0.05	–	–
h	0.16	0.62	0.09	0.47	0.09	–
売上高	0.25	0.36	0.21	0.34	0.18	0.20
申告所得	0.14	0.24	0.13	0.26	0.08	0.16
正の利益金	0.12	0.15	0.13	0.16	0.08	0.10
従業員数	0.25	0.31	0.21	0.29	0.20	0.07
r_{sale}	0.06	0.03	0.07	0.03	0.06	0.03
r_{income}	0.04	0.01	0.06	0.01	0.05	0.03
r_{profit^+}	0.00	0.00	0.03	0.01	0.00	0.00

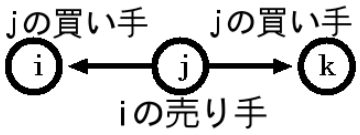
3.2　オーソリティ・ハブ度

オーソリティ度 a と**ハブ度** h は隣接行列の最大特異ベクトル (それぞれ A^TA，AA^T の最大固有ベクトル) として定義され，$a_i = \sum_j A_{ji}h_j$，$h_i = \sum_j A_{ij}a_j$ の計算を再帰的に繰り返すことで算出できる (HITS (hypertext induced topic selection) アルゴリズム)[11]。ただし，

$\sum_i a_i = 1$, $\sum_i h_i = 1$ と規格化するものとする。一般に，オーソリティ・ハブ度は解の一意性が保証されない [12]。一意性を保証するため，本研究では A_{ij} の代わりに $(1 \quad \varepsilon)A_{ij} + \varepsilon/n$ を用いる。ただし，n は全ノード数とする。

$a_i = \sum_{jk} A_{ji}A_{jk}a_k$ となるので，オーソリティ度は，自分の売り手にとっての買い手のオーソリティ度の総和になる (第 10 図)。よって，オーソリティ度は競争相手がどれだけいるかを表す売り手としての劣位性，同様に，ハブ度は買い手としての劣位性と解釈できる。

企業間取引ネットワークでは，a は入リンク数と，h は出リンク数と強く相関しているため (第 2 表)，全体としてみれば，オーソリティ・ハブ度もリンク数と同じような量になっている。



第 10 図：オーソリティ度。

3.3 ネットワーク構造と企業の特徴

Kendall の順位相関係数から，リンク数と企業規模の間には有意な正の相関が認められる (第 2 表)。つまり，リンク数の多い企業ほど企業規模が大きい (第 9 図)。一方，リンク数と企業成長率の間の相関は非常に弱い。これらの企業規模・成長率との相関関係は，ページランク，オーソリティ・ハブ度でみてもほぼ同様である。つまり，企業属性との関係でみても，ページランク，オーソリティ・ハブ度はリンク数と同じような量になっている。

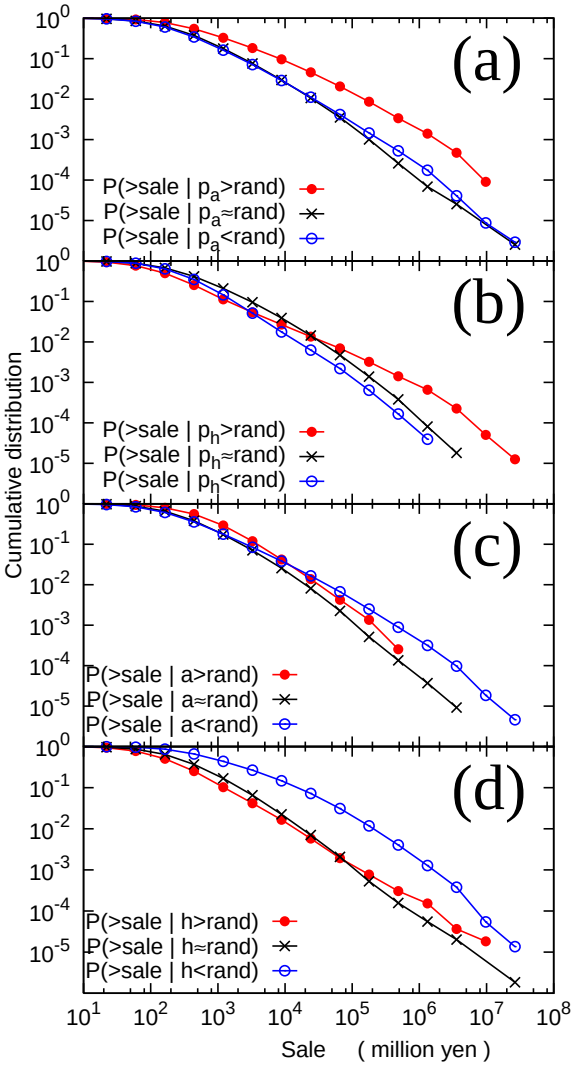
3.4 ランダムネットワークとの比較

リンク数が企業の特徴に与える影響を除去した上で，有向リンク構造と企業の特徴の関係を調べるため，リンク数保存のランダムネットワーク (1000 個) でのページランク，オーソリティ・ハブ度を求め，実際のネットークでの値と比較した。

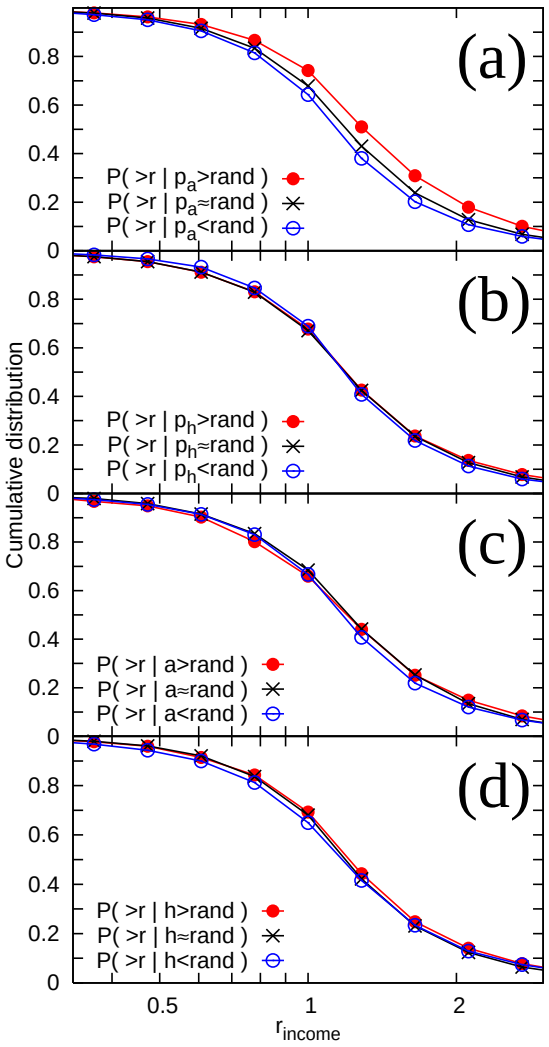
実際のネットワークとランダムネットワークとで，ページランクの分布にほとんど違いはない (第 7 図)。一方，ランダムネットワークと比較して実際のネットワークでは，オーソリティ度の分布の裾野は狭く指数的に減衰し，ハブ度の分布は裾野が広い。これは，売り手として競争相手の多い企業 (オーソリティ度の大きい企業) は少なく，買い手として競争相手の多い企業 (ハブ度の大きい企業) が多いことを示している。

ページランク，オーソリティ・ハブ度のそれぞれについて，ランダムネットワークでの値と比較して，実際のネットワークでの値が統計的有意に「大きい値をとる企業 (>rand)」,「同じような値をとる企業 (∼rand)」,「小さい値をとる企業 (<rand)」の三つのタイプに企業を分類した。各タイプ別にみた売上高の累積分布は第 11 図のようになる。売上高の大きい企業ほど，ページランクを大きく (p_a >rand，p_h >rand)，オーソリティ・ハブ度を小さく (a <rand，h <rand) する，つまり，お金や製品・サービスの流量を大きく，売り手・買い手として優位になるようなリンク構造をとる傾向がある。他の企業規模についてもこの特徴を確認できる。また，各タイプ別にみた申告所得の成長率の累積分布は第 12 図のようになる。成長企業は，ページランク p_a を大きく (p_a >rand) する，つまり，お金の流量を大きくするようなリンク構造をとる傾向がある。他の企業成長率についてもこの特徴を確認できる。

これらの特徴は，入リンク数，出リンク数，ページランク，オーソリティ・ハブ度の値自体だけからでは得られない，ランダムネットワークとの比較によって得られた知見である。特に，企業成長率は単独の企業そのものの情報だけから得られる特徴量とは無相関であるにもかかわらず，お金の流量とは関係しており，有向ネットワーク構造としてシステム全体をみることではじめて捉えられる結果である。



第 11 図：(a) p_a, (b) p_h, (c) a, (d) h で条件つけした売上高の累積分布。他の企業規模についても同様の結果が得られる。



第 12 図：(a) p_a, (b) p_h, (c) a, (d) h で条件つけした申告所得の成長率の累積分布。他の企業成長率についても同様の結果が得られる。

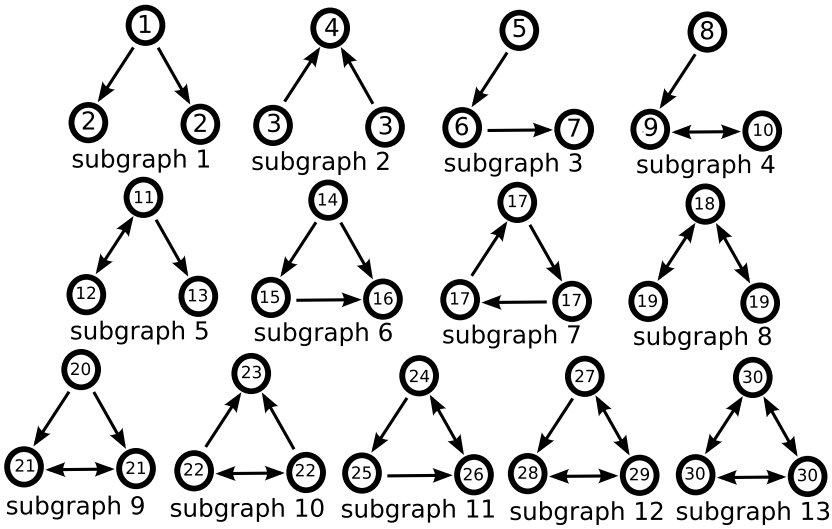
4 ネットワークモチーフ

4.1 特徴的な三体相互作用

リンクをランダムにつなぎ変えたネットワークと比較して，ネットワーク中で統計的有意に出現する特徴的な部分グラフを**ネットワークモチーフ**という [13, 14]。ネットワークモチーフを調べることで，ネットワークを構成する部品を明らかにし，ボトムアップ的にネットワーク構造を調べることができる。

三ノードの部分グラフ (三ノードのつながり方) は全部で 13 種類ある (第 13 図)。有向ネットワークでは，三角形の構成の仕方は一通りではないため，クラスター係数の定義は様々に考えら

れる。三ノードの部分グラフの出現頻度を調べることは，有向ネットワーク上のクラスター係数を調べることに対応する。無向ネットワーク上のクラスター係数と異なり，これらの三ノードの部分グラフはループ・仲介・合流・分岐などの流れ構造を表現している。



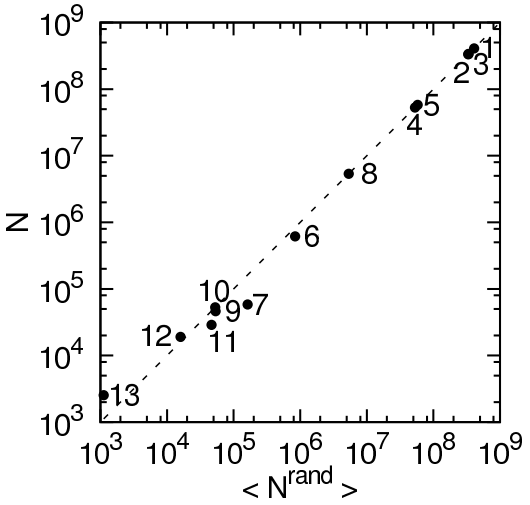
第 13 図：三ノードの部分グラフ。各部分グラフは 1~3 種類の role を持つため，各ノードは 30 種類の role に分類できる。

4.2 ネットワークモチーフ

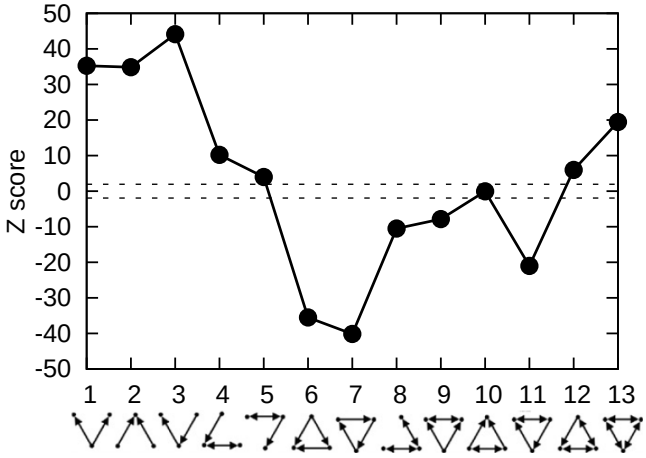
各部分グラフ i の実際のネットワーク上での出現回数 N_i とリンク数保存のランダムネットワーク上での出現回数 $N_i^{\mathbf{rand}}$ は第 14 図のようになる。出現回数は，(1) リンクを一つ取り出す (2) 取り出したリンクに関係する部分グラフを数える (3) そのリンクを除去するという操作を繰り返すことで算出できる [15]。出現回数の差の統計的有意性は

$$Z_i = \left(N_i \quad \left\langle N_i^{\mathrm{rand}} \right\rangle \right) \Big/ \sqrt{\left\langle \left(N_i^{\mathrm{rand}} \quad \langle N_i^{\mathrm{rand}} \rangle \right)^2 \right\rangle}$$

で評価できる (第 15 図)。



第 14 図：各部分グラフの実ネットワーク上での出現回数 N とランダムネットワーク上での出現回数 $\left\langle N^{\mathbf{rand}} \right\rangle$。



第 15 図：各部分グラフの出現回数の統計的有意性。

言語構造や二部グラフでみられるV字構造 (部分グラフ 1~5) と web や友人関係などの社会ネットワークでみられるクリーク (部分グラフ 13) がネットワークモチーフになっている。一方，feedforward ループ (部分グラフ 6) と feedback ループ (部分グラフ 7) はアンチモチーフになっている。つまり，生産・消費活動のネットワークでは，ループ構造は少なくV字型とクリークが重要な構造になる。

4.3 role

ネットワーク構造をより詳細に調べるため，構造同値により **role** を定義する。たとえば，feedback ループ (部分グラフ 7) は対称性からどのノードも同じ立場にあり，各ノードは同じ一つの role を持つが，feedforward ループ (部分グラフ 6) は三つの role を持つ。このようにして，すべてノードは 30 種類の role に分類できる (第 13 図) [3]。

4.4 role による業種の特徴づけ

日本標準産業分類の中分類により，各企業は 96 業種に分類できる。ある一つの業種に注目し，業種間の違いをみることで，たとえば製造業の企業はどの role に位置しやすいかなどを調べることができる。

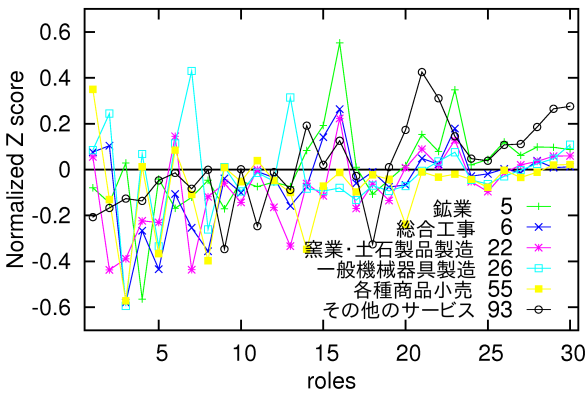
各業種別に，各 role i の実際のネットワーク上での出現回数 N_i とランダムネットワーク上での出現回数 $N_i^{\mathbf{rand}}$ を算出し，出現回数の統計的有意性を

$$Z_i \;=\; \Big(N_i \quad \big\langle N_i^{\mathrm{rand}} \big\rangle \Big) \Big/ \sqrt{\Big\langle \big(N_i^{\mathrm{rand}} \quad \langle N_i^{\mathrm{rand}} \rangle \big)^2 \Big\rangle}$$

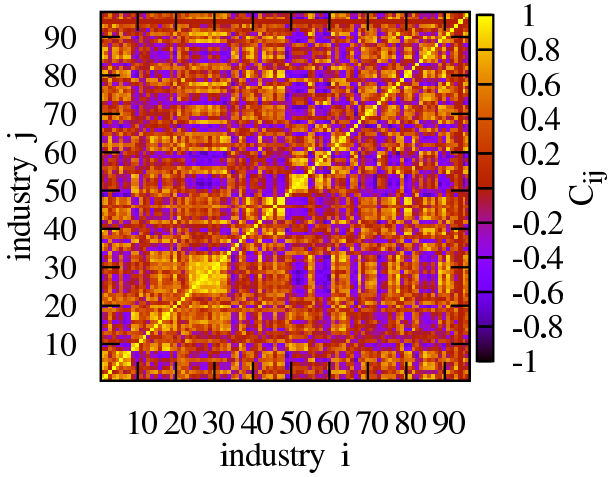
で評価する。異なる業種間を比較するため，Z を規格化した

$$NZ_i \;=\; Z_i \Big/ \sqrt{\sum_i Z_i^2}$$

を考える。各 role i について NZ_i をプロットした role プロファイルは，一般に，業種により異なる (第 16 図)。業種間の NZ の相関行列は第 17 図のようになる。近い業種 ($i \sim j$) 同士は role プロファイルが似ていることが分かる。



第 16 図：業種別の role プロファイル (各 role i について NZ_i をプロットしたもの)。

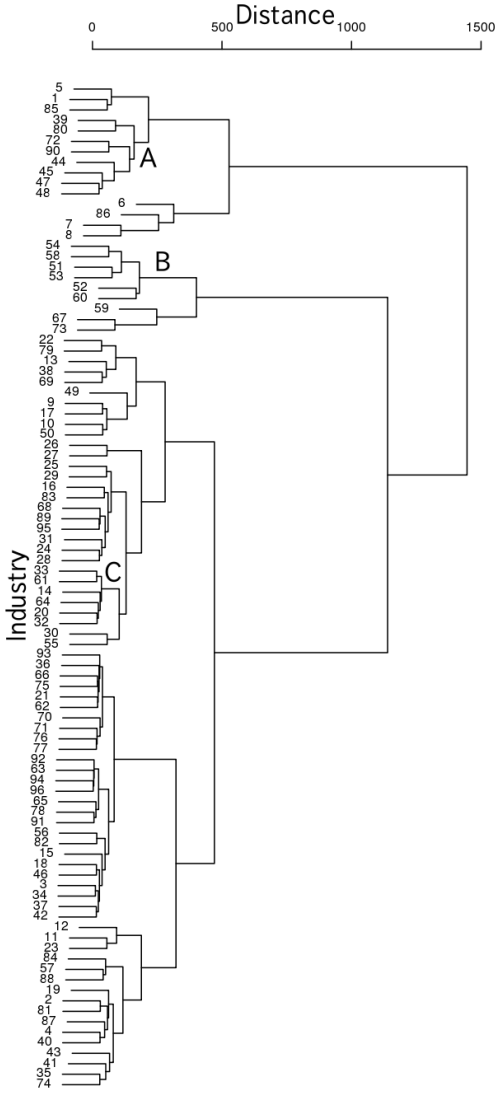


第 17 図： 業種 i の NZ と業種 j の NZ の間の相関係数で定義される相関行列。

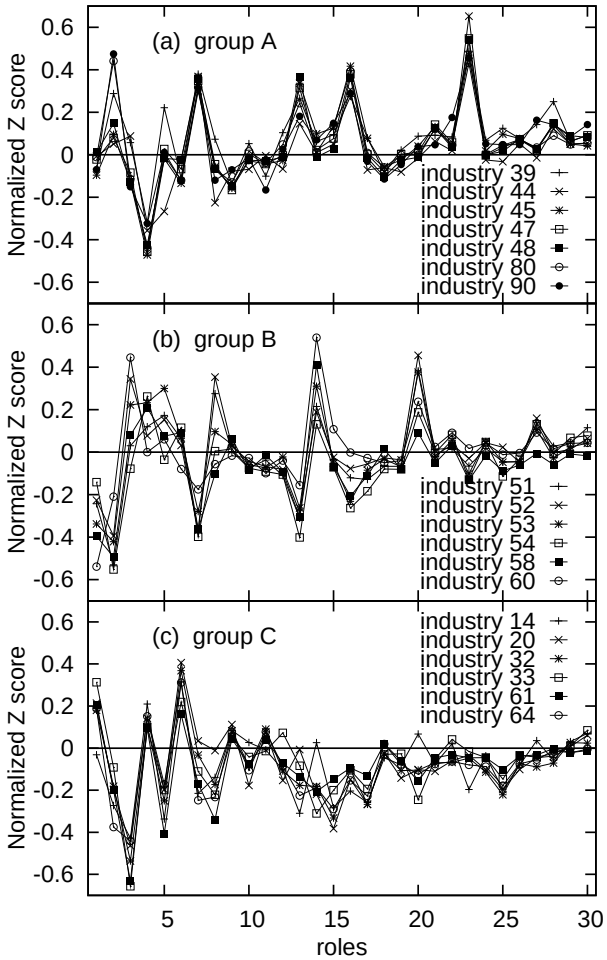
[3]ただし，一般に各ノードは複数の部分グラフに属すため，一つ以上の role を持ちうる。

相関行列で定義される業種間の距離を用いてクラスター分析 (ユークリッド距離によるウォード法) を行い，全 96 業種をグループに分類した。得られたデンドログラム (第 18 図) は，業種間の階層構造を示し，様々なレベルでグループを定義している。

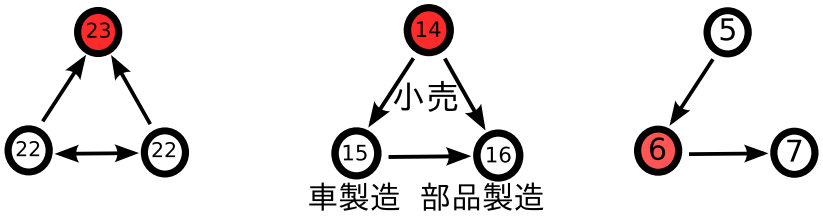
典型的な例として，グループA (39 情報サービス，44 道路貨物運送，45 水運，47 倉庫，48 運輸附帯サービス，80 専門サービス，90 その他の事業サービス)，グループB (51 飲食料品卸売，52 建築・鉱物・金属材料卸売，53 機械器具卸売，54 その他の卸売，58 自動車・自転車小売，60 その他の小売)，グループC (14 家具・装備品製造，20 ゴム製品製造，32 その他の製造，33 電気，61 銀行，64 非預金信用機関) の三つのグループに注目する (第 19 図)。グループAは role 23 で特徴づけられ，互いに取引関係にある二つの企業に製品・サービスを販売するような構造になっており，仲介・代理・取次業務を表現している (第 20 図)。グループBは role 14 で特徴づけられ，卸売業・小売業の典型的な生産・販売の過程を表現している。グループCは role 6 で特徴づけられ，製品を加工し商品を販売する過程を表現し，付加価値を与える加工産業に対応している。



第 18 図：業種のデンドログラム。



第 19 図：各グループの role プロファイル。



第 20 図：各グループを特徴づける role (左：グループA，中央：グループB，右：グループC)。

このように，得られたデンドログラムは業種の事業内容と関係しており，role による業種の分類は経済学的に意味ある分類になっている。また，role プロファイルのみを使って業種を分類することで，有向ネットワーク構造の観点からの分類も可能である。有向ネットワーク構造を表現する role プロファイルは，経済学的に価値ある情報を提供する道具になるものと考えられる。

5　まとめ

日本企業約 100 万社の取引関係・企業属性データを，100 万ノード・400 万リンクの有向ネットワークとして実証分析した。まず，有向ネットワーク上のノードの重要度をページランク，オーソリティ・ハブ度として算出し，企業規模・企業成長率との関係性を調べた。その結果，大企業はページランクを大きく，オーソリティ度・ハブ度を小さくするリンク構造を，成長企業はページランクを大きくするリンク構造を取る傾向を明らかにした。また，ネットワーク中で統計的有為に出現する三ノードの部分グラフを抽出した結果，企業間取引ではV字構造とクリークがモチーフに，feedforward ループと feedback ループがアンチモチーフになっていることを明らかにした。さらに，role による有向リンク構造の特徴づけを行った。これらの結果は，単独の企業そのものの情報だけからでは得られない，有向ネットワーク構造としてシステム全体をみてはじめて捉えられる特徴であり，企業成長・衰退の理解，企業の健全性の把握，操業停止・連鎖倒産のリスク評価などに役立てる上で重要な知見になると考えられる。

近年のグローバル化により，要素間の相互作用が強くなり，世界金融危機・世界的資源インフレ・連鎖破綻に代表される連鎖危機が緊急課題になっている。また，情報化により，経済現象のみならず人間活動に関係するあらゆるデータが利用可能になってきている。このような複雑なシステムを理解するには，要素レベルの網羅的な大規模データから相互作用ネットワークを同定し，ネットワーク構造の特徴抽出・意味づけ・機能推定を行うことで，システム全体としての挙動・特徴を解明する必要がある。したがって，今後，あらゆる分野で本研究のような超並列計算を用いた大規模データの実証分析が重要になってくると考えられる。

謝辞

データ提供いただいた独立行政法人経済産業研究所に感謝する。本研究は，高安秀樹氏 (ソニーＣＳＬ) と高安美佐子准教授 (東京工業大学大学院総合理工学研究科) との共同研究である。本研究の一部は科学研究費補助金若手研究 (B)20760053 と全国銀行学術研究振興財団の助成を受けて行った。数値計算は，スーパーコンピューター若手利用者推薦の下，T2K オープンスパコン HITACHI HA8000 クラスタシステム (東京大学情報基盤センター) を用いて行った。

参考文献

[1] T. Ohnishi, H. Takayasu, and M. Takayasu. Hubs and Authorities on Japanese Inter-Firm Network: Characterization of Nodes in Very Large Directed Networks. *Progress of Theoretical Physics Supplement*, 179:157–166, 2009.

[2] T. Ohnishi, H. Takayasu, and M. Takayasu. Network Motifs in Inter-Firm Networks. *Journal of Economic Interaction and Coordination, under review.*

[3] A. Clauset, C.R. Shalizi, and M.E.J. Newman. Power-law distributions in empirical data. *Arxiv preprint arXiv:0706.1062*, 2007.

[4] Y.U. Saito, T. Watanabe, and M. Iwamura. Do larger rms have more inter rm relationships? *Physica A: Statistical Mechanics and its Applications*, 383(1):158–163, 2007.

[5] Y. Fujiwara and H. Aoyama. Large-scale structure of a nation-wide production network. *Arxiv preprint arXiv:0806.4280*, 2008.

[6] H. Fan, Z. Wang, T. Ohnishi, H. Saito, and K. Aihara. Multicommunity weight-driven bipartite network model. *Physical Review E*, 78(2):26103, 2008.

[7] E. Ravasz and A.L. Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):26112, 2003.

[8] R. Milo, N. Kashtan, S. Itzkovitz, M.E.J. Newman, and U. Alon. On the uniform generation of random graphs with prescribed degree sequences. *Arxiv preprint cond-mat/0312028*, 2003.

[9] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[10] C.P.C. Lee, G.H. Golub, and S.A. Zenios. Partial state space aggregation based on lumpability and its application to pagerank. *Technical report, Stanford University*, 2003.

[11] J. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632, 1999.

[12] A. Farahat, T. Lofaro, J.C. Miller, G. Rae, and L.A. Ward. Authority Rankings from HITS, PageRank, and SALSA: Existence, Uniqueness, and Effect of Initialization. *SIAM Journal on Scienti c Computing*, 27(4):1181, 2006.

[13] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827, 2002.

[14] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon. Superfamilies of evolved and designed networks. *Science*, 303(5663):1538–1542, 2004.

[15] R. Itzhack, Y. Mogilevski, and Y. Louzoun. An optimal algorithm for counting network motifs. *Physica A: Statistical Mechanics and its Applications*, 381:482–490, 2007.

厳密な理論波形計算を用いた高解像度地球内部構造推定

竹　内　　希
東京大学地震研究所

１．本研究の背景

地球内部を伝播する地震波を解析しグローバル地球内部構造を推定する研究は，地震学の中心的課題の一つと考えられている．これまでの研究では，計算機資源の限界から（1）（Ｐ波やＳ波の到達時刻など)観測地震波形データから抽出された一部の情報をデータとし，(2)（無限小波長近似などの）近似を用いて，内部構造モデルパラメータ推定を実施していた．近年，(1)観測地震波形データそのものをデータとし，(2)これと厳密な（有限波長の効果を厳密に考慮して計算された）理論波形を直接比較することにより内部構造パラメータを推定する，「波形インバージョン」が注目されている．地震波形データに含まれる全情報を正確に抽出できるようになるため，内部構造モデルの精度と解像度の改善が期待できる．しかし膨大な計算量が問題となるため，従来の研究では理論波形計算の際に粗い近似を導入していて，厳密な理論波形を用いた「波形インバージョン」はまだ実現されていなかった．

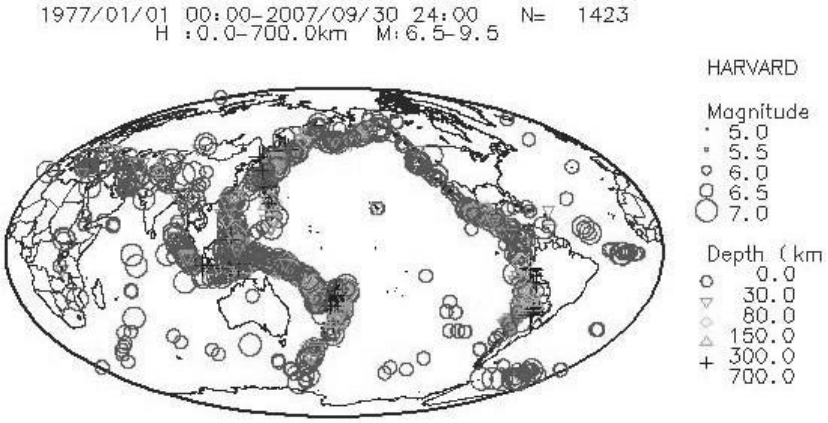
２．本研究の目的・意義

本研究では，HA8000 クラスターシステム及び地球シミュレーターを駆使し，厳密な理論波形を用いた波形インバージョンを実現する．また，得られた地球内部構造モデルから，マントル対流上昇流の様式を制約する特徴的な不均質構造を検出する．

波形インバージョンは，これまでサンプリングの乏しかった地域のモデル解像度を改善することに大いに役に立つ．大きな地震の大部分はプレート境界で発生（第１図）し，観測点の大部分は陸域に存在する．このため地震波のサンプリングには，必然的に大きな偏りが生じる（第２図右）．太平洋の下や，インド洋・アフリカの下は，地震・観測点数ともに乏しい領域なので，これまでの内部構造モデルの解像度が低かった．これらの領域は偶然にもマントル対流上昇流域にあたるので，上昇流のダイナミクスを制約するような内部構造の情報が乏しかったとも言える．

実体波の到達時刻は(他のフェーズとの干渉のない)孤立した実体波フェーズ以外は計測が困難である．地震波には様々なフェーズがあり，後続波になるほど到達するフェーズの種類が多くなる．このため走時解析による内部構造推定では，初動Ｐ波及びその近傍のフェーズのみをデータとして用いてきた．従って得られた内部構造モデルでは，使用可能な情報量の限界から，上昇流域の解像度は乏しかった．一方で波形インバージョンを実施すれば，フェーズの干渉があっても解析上の困難は生じず，後続波の徹底活用が可能になり，上昇流域におけるサンプリングが改善される（第２図左）．実際に本研究の解析では，地震発生から約４時間長のデータを活用できており，同種の解析では類を見ない時間長のデータを活用できている．

上昇流ダイナミクスを制約するために特に注目すべき特徴は，上昇流に起因すると思われる高温異常域（地震波速度構造では低速度異常域に対応する）の広がりが，深さ 670 km の地球

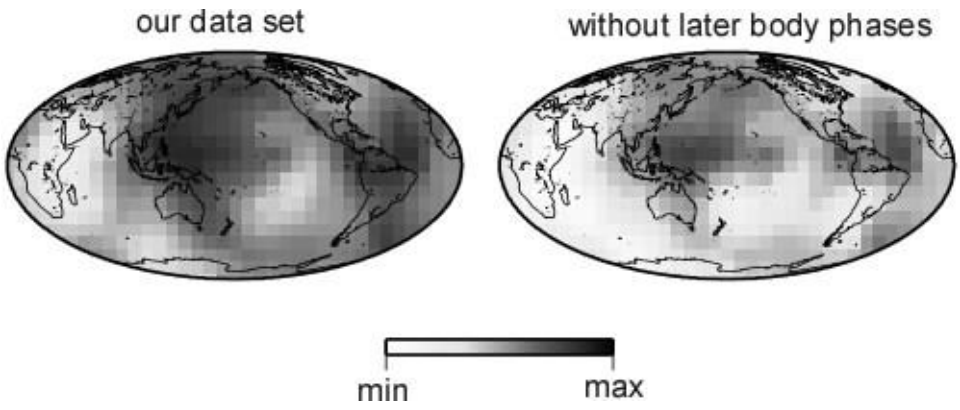


第 1 図：大地震の発生する地域。

世界の標準地震カタログの１つである，Harvard 地震カタログの中で，モーメントマグニチュードが 6.5 以上のイベントの分布．1977/01/01-2007/09/30 に起こったイベントをプロットしてある．

内部不連続面の上下でどのようになっているかである．670 km 不連続面は，マントルを構成する主要鉱物が相変化を起こしている面であり，この面によってマントル対流上昇流・下降流がともに抵抗力を受けると考えられている．実際（比較的高い解像度の内部構造モデルが得られている）下降流域では，670 km において下降流（つまり沈み込むスラブ）が抵抗力を受け，不連続面上で一旦横たわるという描像が確立されている．それに対し，上昇流が 670km 不連続面で抵抗力を受け不連続面直下で停滞するのか，あるいは抵抗力を受けつつも不連続面を突き抜けるのかは，未だ良くわかっていない．

本研究では，後続波を徹底活用した波形インバージョンを実施することにより，上昇流域のモデル解像度を改善した．得られたモデルから，670 km 不連続面を突き抜ける筒状の低速度異



第２図：後続波データを徹底活用した場合としない場合のサンプリングの比較。

我々のデータセット（地震発生から約４時間長のデータ）のマントル最下部におけるサンプリング（左）と，我々のデータセットから後続実体波（Ｇ１フェーズ以降のフェーズ）を除いたデータセットのサンプリング（右）の比較．

常が確認された．さらに，670 km 不連続面の上下で，筒の傾きが大きく変化することが確認された（第７図；後述）．このことは，上昇流は 670 km 不連続面で抵抗力を受け，その流れの向きを大きく変えつつも，そのまま突き抜けているという描像を示唆する．

３．波形インバージョンによる内部構造推定問題の概要

波形インバージョンによる内部構造推定の中で，最も計算時間を要するのは，理論波形のモデルパラメータに対する偏微分係数計算である．詳細の説明は省略するものの，偏微分係数計算の主要部分は，非零要素をおよそ 5×10^{11} 個もつ疎行列の掛け算である．実際に内部構造推定を実現するには，この掛け算をそれぞれの周波数及び地震に対して(本研究では 512×191 回)実施する必要がある．

「厳密な理論波形計算」とは，この疎行列の掛け算を近似なしで計算することに対応している．この膨大な掛け算が計算機資源の限界から困難であったことが，波形インバージョンの実現を困難にしていた理由の１つである．（本研究ではこの他にも，適切な基底関数の選定や，微分演算の離散化手法を工夫により計算時間の短縮を実現している．）

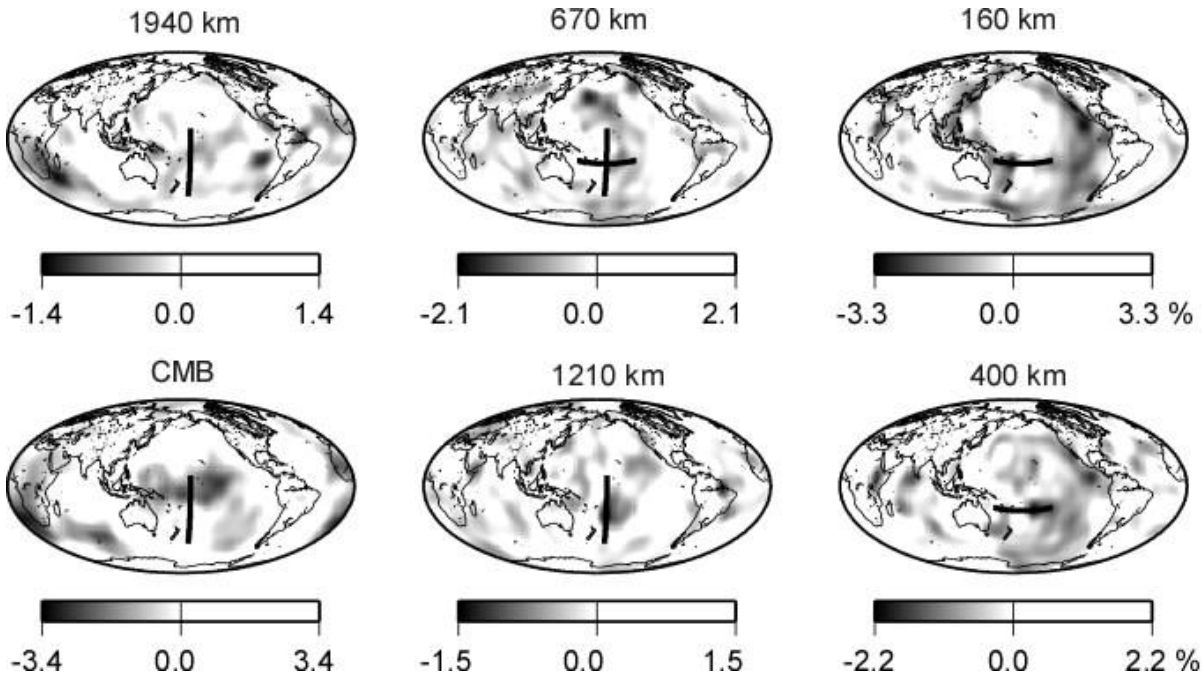
並列化手法は単純であり，疎行列を行毎に分割・グループ分けし，それぞれの行グループの掛け算を別々のＣＰＵに割り振って計算する．各行グループ内の非零成分の個数がほぼ等しくなるように行の分割を実施する．本計算に要する時間は１地震あたり，地球シミュレーターでは８時間（実時間，8 ノード 64CPU を使用），HA8000 では７時間（実時間，64CPU 256 コアを使用）であった．地球シミュレーターでは平均ベクトル長をできるだけ長くするように，HA8000 ではメモリ参照ができるだけ連続になるようにループの順番を入れ替えてあるが，計算に必要な演算回数は同じである．

４．平成２０年度の成果

平成１９年度の研究で，波形インバージョンによる全マントルＳ波速度構造推定を実施し，南西太平洋の下の特徴的な上昇流域の低速度異常構造を検出していた(Takeuchi, 2007 によるモデル SH18CE；第３図及び第７図左)．本年度はこの描像の妥当性を，特異値分解(Wiggins, 1972)を用いて検証した．具体的には，全マントル速度構造を表現するのに用いる基底の数を様々に変化させてモデル推定を実施し，注目している構造が，用いる基底の数によらず普遍的にみられる特徴であるかどうかを検討する．多くの基底を用いて推定される構造モデルは，解像度は高いが精度は低いモデルであり，少ない基底を用いて推定されるモデルは，解像度は低いが精度は高いモデル（ロバスト成分を表したモデル）である．基底の数によらず見られる特徴は，解像度・精度ともに十分な信頼性の高い特徴であると言える．

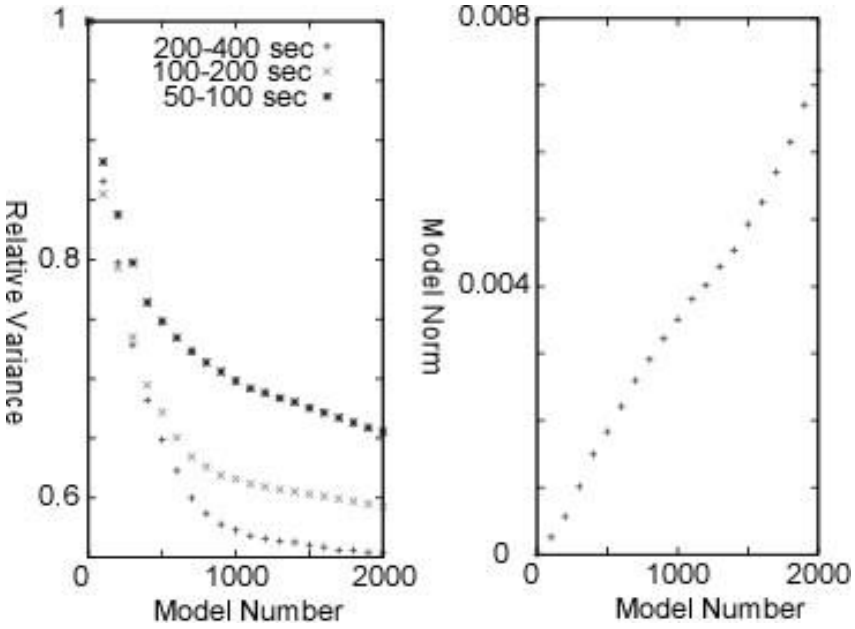
基底の数とデータ残差の関係をプロットしてみると，全体的な傾向として，基底の数を増やせば増やすほど残差は相対的に改善されるが，モデルのノルム(不均質構造の RMS 振幅に対応する)は増大するという特徴が見られる（第４図）．つまり，基底の数を増やすほど，大きな不均質を導入しつつ残差を改善していると言える．

第４図をさらに良く見てみると，基底の数が 800 から 1200 のモデルにかけて，モデルのノルムの増大が緩やかであるのに，残差が改善され続けることがわかる．これから，ノルムをあまり増やさずに大きな残差改善を実現するモデルとして，基底の数が 1200 の時のモデルを最適モデル(MODEL-1200；第５図)として選定することとする．



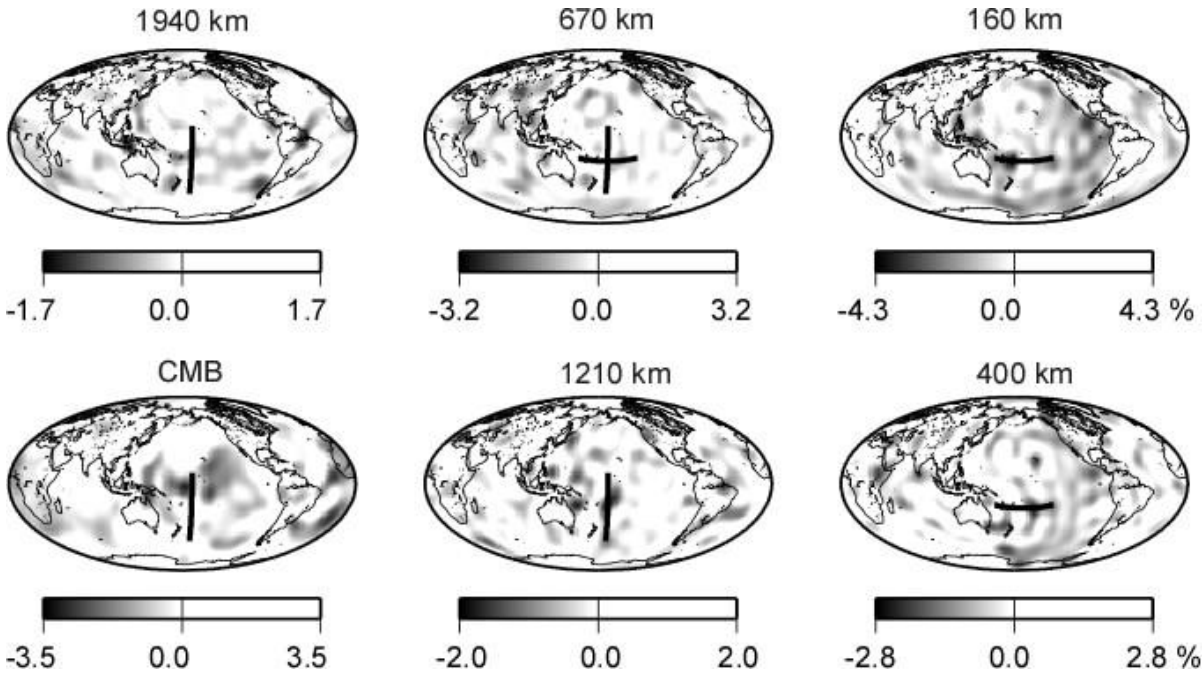
第３図:平成１９年度に推定された構造モデル(SH18CE)。

Takeuchi (2007) によって推定された全マントルＳ波速度構造モデル (SH18CE). CMB 及び様々な深さにおける標準球対称モデル(PREM)からのずれをプロットしてある. 色がついている領域が高温異常に対応すると思われる低速度異常域を表す. 太い実線における断面図を第７図に示してある.



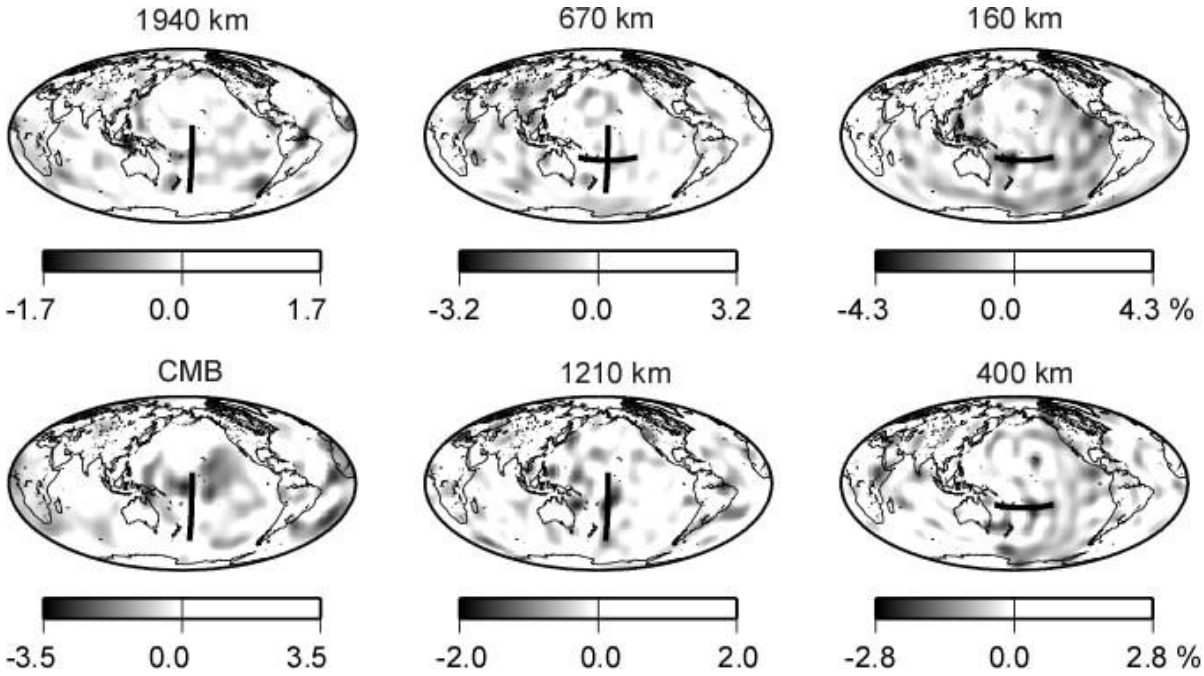
第４図: 基底数(Model Number)と残差及びモデルノルムの関係。

用いる基底数を様々に変化させて推定した場合の, それぞれの構造モデルに対するデータ残差(Variance)とモデルノルム(Model Norm). Variance は周波数帯域毎に別々に計算し, 初期モデル(Model Number = 0)に対する残差により規格化した値をプロットしてある.



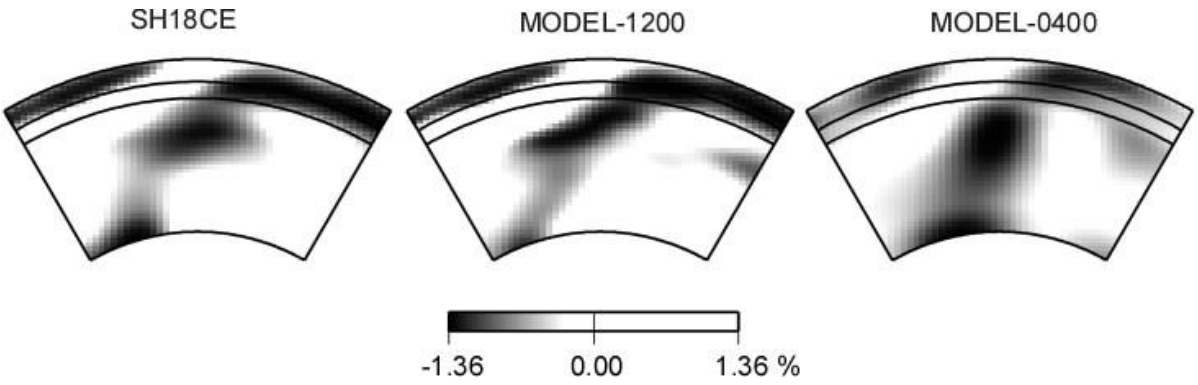
第５図: 平成２０年度に推定された最適モデル(MODEL-1200)。

1200 の基底を用いて推定された最適モデル. その他の詳細は第３図と同様.



第６図: 平成２０年度に推定されたロバストモデル(MODEL-0400)。

400 の基底を用いて推定されたロバストモデル. その他の詳細は第３図と同様.



第７図：本研究で推定されたモデル間の南西太平洋下低速度異常構造の比較。
SH18CE(第３図)，MODEL-1200（第５図)，MODEL-0400（第６図)における太線における断面図の比較．各断面図中の太い実線は，深さ 400km 及び 670km にある不連続面の位置を表す．

一方で，データの残差を周波数別に見てみると，長周期データ（200-400 sec）では 400 程度の基底数でデータ残差を大きく減らすが，それ以上基底数を増やしても残差のさらなる改善はあまりない．短周期データ（50-100 sec）では，400 程度の基底数でデータの残差は相当減るが，それ以降も緩やかな残差改善が見られることがわかる．このことから，すべての周波数帯域のデータに対して相当の残差改善を実現する最低限のモデルとして，基底数 400 のモデルをロバストモデル（MODEL-0400；第６図)として選定する．

平成１９年度のモデル（SH18CE；第３図）及び，今回推定した最適モデル（MODEL-1200；第５図），ロバストモデル（MODEL-0400；第６図）を比較すると，前者２つのモデルは高い相関を持つが，ロバストモデルは他の２つのモデルと相関が低いことがわかる．実際に SH18CE と MODEL-1200 の相関係数は 0.72 であるが，SH18CE と MODEL-0400 の相関係数は 0.42 に過ぎない． しかしながら南西太平洋の下の着目している低速度異常域は，どのモデルでも同様な広がりが見られる（第７図）．このことからこの特徴は信頼性の高い特徴であると言うことができる．

５．まとめと今後の課題

平成１９年度の研究で，波形インバージョンによる全マントルＳ波速度構造推定を実施し，南西太平洋の下に，筒状の低速度異常構造を検出した．平成２０年度に HA8000 を用い，着目している低速度異常域の信頼性について検討し，データから良く制約された信頼性の高い特徴であるとの結論を得た．

地球シミュレーター上で開発した内部構造推定ソフトウェアを HA8000 に移植をし，正しく動かすことには成功した．現状では，各ＣＰＵあたり(4 コアあたり)およそ 10^{14} 回の浮動小数点演算を要する計に約７時間要している．つまり実行性能は 4GFLOPS 程度であり，理論上のピーク性能(36.8GFLOPS)の 10%強であり，まだ性能改善が可能であると考えられる．メモリ参照の衝突が性能劣化の主たる原因であると考えられるので，今後改善に努めたいと思う．

本研究で実施したのはＳ波速度構造推定であり，３成分ある地震波形記録のうちＳ波速度に敏感な１成分（トランスバース成分）のみを用いている．３成分波形記録を用いたＰ波速度・Ｓ波速度構造の同時推定が次なる課題となるが，現在の計算よりもさらに１桁以上多くの計算

量を要する．今後コーディングの改善とともに，さらなる計算機の性能改善に期待して，同時推定を実現したい．

参 考 文 献

Takeuchi, N.『Whole mantle SH velocity model constrained by waveform inversion based on three-dimensional Born kernels』, Geophysical Journal International, vol.169. 1153-1163, 2007

Wiggins, R.A.『The general linear inverse problem: implication of surface waves and free oscillations for earth structure』, Rev. Geophys. Space Phys., vol.10, 251-285, 1972

数値モデルによる海洋微小スケールプロセスの解明

松 村 義 正

東京大学気候システム研究センター

１．はじめに

海洋には数千年のタイムスケールで地球を巡る大規模な循環場が存在する。この大循環は大まかには次のような全体像をなしている。すなわち、まず南極沿岸やラブラドル海など高緯度の幾つかの領域において，大気から冷却されることによって海面付近の海水が高密度化し，深層まで沈み込む。このようにして海洋深層に供給された冷たく重い水は海底地形の制約を受けながらゆっくりと低緯度域に向かい，そこで暖かい水と混合することにより浮力を得、表層まで湧昇する。海洋表層には主に風によって形成される強い流れ(例えば黒潮やメキシコ湾流等)が存在しており，深層を巡って湧昇してきた水はこの表層の循環によって再び高緯度の沈み込み域まで運ばれる。以上のような海洋の大循環は低緯度域から高緯度域に向けて大量の熱を輸送するため，地球の気候形成に極めて重要な役割を担っている。したがって例えば数値モデルによって温暖化予測を行おうといった際には，この海洋大循環をモデル内で如何に現実的に再現するかが肝となる。海洋大循環の再現を主要な目的とする数値モデルを一般に海洋大循環モデルと呼ぶが，これは基本的には流体の運動を記述する Navier-Stokes 方程式と，温度・塩分及び圧力と海水の密度の関係を記述した状態方程式を支配方程式とする一般的な数値流体コードである。ただし数千年，数万 km という時間的にも空間的にも非常に大きなスケールの現象を扱うという点が特徴であり，このため大循環モデルでは格子幅が非常に粗く，また計算効率を高めるためいくつかの大胆な近似による簡略化を行っている。最も特徴的なのは圧力を静水圧，すなわち“大気圧+その格子より上に存在する海水の重さ”で置き換える静水圧近似でり，鉛直加速度を無視して水平的な収束発散のみから鉛直速度を求めることに相当する。これは海水の運動が水平方向数万kmのスケールに大して深さが高々数千mしかない薄い層の中での現象であり，さらに深層が冷たく重く，表層に行くほど暖かく軽いという密度成層がより海水の鉛直運動を妨げることを鑑みれば，ある程度は正当な仮定と言えよう。

近年の計算機資源を用いれば，上記のような海洋大循環モデルに海面での境界条件として観測に基づく風の場や日射・降水等を与えつつ水平格子幅数 10km 程度の解像度で数千年のモデル時間走らせることによって，観測と整合的な大循環場を再現することが可能となっている。(ただし特に海洋深層の流れの強さやその流路は直接観測が困難なこともあり未だ良く把握されておらず，直接観測とモデリング研究の両面から活発な議論が続いている。）このような海洋大循環モデルと大気大循環モデル，海水の相変化と氷の力学を扱う海氷モデル，さらには陸面での水循環や植生を扱う陸面モデル等が互いに影響を与え合う結合モデルの計算結果は気候変動に関する政府間パネル(Intergovermental Paanel on Climate Change, IPCC)による評価報告書における温暖化予測の主要な根拠の一つともなっており，したがって海洋大循環モデルの信頼性向上は海洋物理学上の要請にとどまらず，人類共通の課題の一つといっても過言ではない。

先に海洋大循環モデルにおける静水圧近似が，海洋の全球規模大循環を構成する海水の運動

の水平スケールが鉛直スケールよりもはるかに大きいために妥当であるという旨を述べたが，実はこれはあまり正確でない。大循環の構造の水平と鉛直のスケールのアスペクト比自体は確かに十分大きいのであるが，そもそもこの大循環は，内部重力波の砕波によって生じる乱流混合や高緯度域の海面で浮力を失った水が局所的に海洋中深層まで沈み込む過程いった，時間的・空間的に小さいスケールの現象によって駆動・維持されている。これら海洋の微小スケールプロセスでは運動のアスペクト比は１に近くもはや静水圧近似は成り立たない。ではなぜ微小スケールの非静水圧な運動によって駆動・維持されている海洋大循環が，静水圧近似を行っている大循環モデルで再現されているかとうと，誤解を恐れずに言えば，そもそも現状の大循環モデルでは格子が粗いために大循環を駆動する微小スケールプロセスは解像することができず，経験的なパラメタリゼーションによって代替しているからに他ならない。パラメタリゼーションとは簡単に言えばモデルで解像されないサブグリッドスケールの現象が実際にはどうあるべきかを，物理的な考察や観測や室内実験に基づく経験的な知識からモデルで解像されている物理量を元に推定し，モデル計算を逐次補正することである。 もちろん用いられるパラメタリゼーションが確固たる物理的根拠に基づいており，その妥当性が十分検証されたものであるならば，パラメタリゼーションを含む大循環モデルの結果も信頼できるものであるのだが，実際には必ずしもそう理想的ではない。特にパラメタリゼーション内で具体的な定数が必要な場合には，最も観測と整合するような結果を導くパラメータ値をアドホックなチューニングによって決定することになり，往々にしてある物理量を観測と合わすようにチューニングすると他の物理量がよりずれてくるということになりがちである。また最悪の場合には，現実に起きている現象とはまったく異なるメカニズムによって大循環がモデル内で実現されているということもあり得る。例えば低緯度域での深層水の浮力の獲得は潮汐を起源とする乱流混合によって為されるはずが，精度の低い移流スキームを用いたモデルではこれを強い数値拡散が上書きし，非現実的な循環をモデル内で実現してしまう。このような状況では，パラメータチューニングの結果現在の深層循環が現実的に再現できたとしても，古気候や温暖化予想といった現在とは大きく異なる気候システムを対象とした場合においてそれらのパラメータが適切である保証は無く，気候変動予測における大循環モデルの信頼性には疑問が残ると言わざるを得ない。

以上のような海洋大循環モデルの現状を考えると，その信頼性の向上は究極的にはできるだけ近似を行わない方程式系に基づき，かつ海洋で起きているあらゆるスケールの現象を解像するのに十分な格子幅での実験をすることによって実現されるべきなのであるが，そのような計算は例え計算機の発展が現在のペースで数十年続いたとしても困難である。したがってまず我々にできることは，既存の大循環モデルで解像されない微小スケールプロセスの物理的な詳細を理解し，それらのプロセスが定量的に深層水の形成やその湧昇にどの程度の貢献をしているのかを見積り，その知見をより良いパラメタリゼーションの開発と改良という形で大循環モデルにフィードバックしていくことである。微小スケールプロセスの詳細な理解のために最も重要なのは実際に何が起きているかを直接観測することであるのは疑う余地が無いが，海洋中深層での現象を時空間的に密な観測によって把握することは極めて困難である。そこで特定の現象，あるいは特定の海域のみを対象に，利用可能な計算機資源の制限内で可能な限り高解像度かつ近似を廃した領域モデルを構築し，大循環モデルでは決して表現され得ない微小スケールプロセスをモデルで再現することができれば，そのメカニズムの理解やより大規模な海洋循環に与える影響を評価することが可能となる。

筆者は東大情報基盤センターの 2008 年度若手研究者推薦でいただいた計算機リソースを活用し，まず海洋微小スケールプロセスを扱うことに特化した数値モデルを開発し，その計算効率の向上に取り組んだ(Matsumura and Hasumi, 2008)。次にその数値モデルを用いて南極沿岸での局所的な深層水形成を再現する高解像度の数値実験を行うことにより，当該海域でどれくらいの量の高密度水がどのくらいの深さまで沈み込んでいるのかをシミュレートした(Matsumura and Hasumi, 2009a)。またその結果の解析から，特に海底地形の小規模な起伏の影響に着目して力学的な考察に行い,大陸斜面上における等深線の曲率や勾配の空間変化といった地形的な特徴と,高密度水の流路との関係を定式化した(Matsumura and Hasumi, 2009b)。これら一連の研究を紹介する。詳細な記述は各原著論文に任せ，本稿では概要を述べるに留めるが，本誌の読者の主要な興味の対象である数値計算手法とその効率に関してはできるだけ詳しい記述を行うよう心がけた。なお本稿第 3 節の内容は 2009 年特集号 2 に掲載された T2K オープンスパコン HPC 特別プロジェクト成果報告(羽角ら)と一部重複することを了承願いたい。

２．海洋非静力学モデル開発

前節に述べたとおり，既存の海洋大循環モデルは静水圧近似を行っているが，本研究の対象である海洋微小スケールプロセスにおいては静水圧平衡は成り立たない。そのため既存の大循環モデルを単に高解像度すのでは不十分で，静水圧近似を行わなず圧力場を陽に計算する数値モデルが必要となる。ただし流体運動の時間発展を計算する際に圧力を陽に予報するということは，密度の粗密波によって圧力偏差が波動として伝播する現象，すなわち音波を扱う必要があることを意味する。海水中の音波の伝播速度は 1500 m s^{-1} 程度であるが，これは最も速い海流と比較しても 1000 倍程度大きい。したがって CFL 条件を考慮すると，興味の対象であるところの海水の運動に積極的な影響を与えないにも関わらず，それよりもはるかに速く伝播し時間刻み幅を制約する音波を真面目に扱うのは賢明とは言えない。幸いにして海水の圧縮率はさほど高くなく，圧縮運動による密度の変化は，非一様な日射や降水等によって生じる温度・塩分偏差による密度変化に比べ十分小さい。そこで流速ベクトル $\boldsymbol{u}$ に対して非発散条件 $\nabla \cdot \boldsymbol{u} = 0$ を仮定することにより，Navier-Stokes 方程式中の圧力 p を，この非発散性を満たすように Poisson 方程式を解くことによって求めることができる(Marshall et al, 1997a)。なお壁面を貫く流速は 0 であるべきとの要請から，壁面においてそれに垂直な方向の圧力の微分が 0 となるノイマン境界条件が課せられる。このように流速の非発散を仮定して Poisson 方程式から圧力を求めることは音波の伝播速度を無限大とみなすことに相当し，海洋物理の対象となる現象では妥当な仮定である。このような近似は非圧縮近似とも呼ばれるが，温度や水圧による密度の変化の影響は海水の状態方程式に含まれる形で考慮されている。

以上のように，静水圧近似をしない非静力学海洋モデルでは圧力の導出のために 3 次元 Poisson 方程式をタイムステップ毎に解く必要があるが，これは計算コストの観点から容易でない。離散化された系では解くべき Poisson 方程式は $\boldsymbol{q} - L\,\boldsymbol{p} = 0$ と表現される。ここでモデルの総格子数を N(通常最低でも 10^6 以上のオーダーである)とすると $\boldsymbol{p}, \boldsymbol{q}$ は N 次元ベクトル，L は離散化されたラプラス演算子を表す $N \times N$ の正方行列である。$\boldsymbol{p}$ が解くべき圧力（正確には圧力から静水圧及び大気圧を除いたものをリファレンス密度で割った modified pressure と呼ばる量）である。一方 $\boldsymbol{q}$ は Navier-Stokes 方程式中の $\boldsymbol{p}$ 以外の項から導出される各格子での収束発散を表し，タイムステップ毎に評価される。圧力 $\boldsymbol{p}$ をこの方程式を満たすように求めれ

ば， Navier-Stokes 方程式が非発散性を保ったまま時間積分を行えることになる。我々のモデルは水平方向には曲線直交座標,鉛直方向には格子の厚みが水平一様な座標系を採用しており，速度は各格子面に対して垂直な成分がその面の中央にて定義され，圧力及び温度塩分等の海水の特性は各格子点中央にて定義されている。このような格子配置においてはある格子での ∇p はその格子とそれに隣接する 6 格子の計 7 格子における p から求まるので，L の各行は 7 つの非ゼロ要素を持つことになる。$\boldsymbol{p}$ は L の逆行列を用いて $\boldsymbol{p} = L^{-1}\boldsymbol{q}$ と書けるが，実際に直接 L^{-1} を求めることは計算時間の観点からもメモリ容量の観点からも不可能であり，その他の解法を用いることになる。離散化された Poisson 方程式の最も高速な解法は高速フーリエ変換(FFT)を用いたスペクトル法であるが，これは複雑な海底地形が存在する状況では適用できない。そこで既存の海洋非静力学モデルでは共役勾配(conjugate gradient，以下 CG)法によって反復的に圧力を求める手法が広く採用されている(Marshall et al.， 1997b)。ただし CG 法の収束性能は前処理行列の選択によって大きく左右されるが，最適な前処理行列は問題設定やその規模に依存しており一意に決められない。またより重要な点として，一般に CG 法その他の Krylov 部分空間法における収束までの反復回数は総格子数に依存して増加するため，モデルを高解像度化したり，より広い領域を覆うために格子数が増加すると，総計算コストは指数関数的に増大してしまう。

このような問題から，非静力学モデルを用いた海洋微小スケールプロセスの研究は狭い領域・短い積分期間のみを扱った理想化実験や，地形が存在しない表層のみを対象としたものに留まってきた。そこで我々はまず，近年工学分野で大きな成果を挙げている多重格子法を導入し，海洋非静力モデル内における Poisson 方程式の計算コスト低減に取り組んだ。多重格子法は反復解法の一種であるが，解像度の異なる複数の格子を用意し，高波数の誤差は細かい格子上で，一方より低波数の誤差は粗い格子上でというように階層的に誤差の減衰を行うという特徴がある。この手法の著しい利点は，理論的には収束までの反復回数が格子数によらずほぼ一定であるというスケーラビリティにある。したがって非静力学海洋モデル内の Poisson 解法に多重格子法を用いることができれば，その計算コストは総格子数に対して線形となり，上に述べたような非静力学モデルを大規模実験に使用する際の欠点を 解消できる。ただ導入にあったて幾つか海洋モデルに特徴的な問題があり，それらについて対処をする必要があった。以下ではそれら問題点とその対処法を述べる。ただし係数行列の形状やその詳細な定義をここで逐一説明して解法の完全なる記述を行うことは控え，概念的な記述に留める。詳細に興味があれば Matsumura and Hasumi(2008)を参照にされたい。

まず海底地形の表現と境界条件について述べる。境界で圧力微分が 0 になるというノイマン型境界条件は，最も単純には境界の外側（つまり地中に相当する格子）での圧力場を仮想的に領域内部の圧力場の鏡像対称として与えることによって実現できる。境界が平面であればそれで問題ないのだが，斜面や海嶺等の凹凸がある地形においては単純な鏡像対称によって領域外での仮想圧力を与えることはできない。海底地形に隣接する格子での圧力を一定値に固定するという方法も考えられるが，これでは例えば斜面を降る高密度水の流れにおける圧力傾度が表現できない。そこで各格子においてそこが領域内であれば 1，地中に相当する場合は 0 をとるマスク関数を定義し，係数行列 L 内の各係数に対して，ある格子面での微分を表す項にその面を挟む隣接 2 格子におけるこのマスク値の積を乗じることにする。これにより地中に相当する格子における仮想的な値を参照することなく，境界面での微分は自動的に 0 となってノイマン

境界条件が満たされる。ただし領域を囲む境界すべてでノイマン境界条件を適用すると p が一意に定まらず，L の行列式が 0 となって解けなくなることに留意する必要がある。この特異性は例えば p の平均値を明示的に指定する等の拘束条件を新たに課すことによって解消できる。圧力 p はモデルの支配方程式において常に空間微分の形でのみ登場するので，その平均値は物理的意味を持たず， 上記の拘束条件を課すことによって一般性は失われない。

次に多重格子法における粗い格子の構築方法を述べる。我々のモデルは直交する座標軸上で定義された構造化格子を採用しているので，粗い格子系は細かい座標系において隣接する複数の格子を結合することにより実現される。このため非構造格子を用いるようなコードに比べはるかに容易かつ簡潔に多重格子法を実装できた。ただし海洋モデルでは通常水平方向の格子幅に比べ鉛直方向の格子幅が著しく小さい通称パンケーキ型の格子形状となっている。このような格子幅に非等方性がある場合，単純に各方向同数の細かい格子を結合して粗い格子を構築するのでなく，粗い格子において各方向の格子幅が同程度となるように，すなわち粗い格子上での係数行列の非ゼロ要素の大きさが均一となるように結合させる方が高い収束率を実現できることが知られている（より一般的には代数的多重格子法と称される)。海洋モデルのようにパンケーキ状格子であれば，まず鉛直方向にのみ幾つかの格子を結合し，立方体に近い格子形状となった格子レベル以降で 3 次元的に結合することによって粗い格子を構築すればよい。また複雑な海底地形における境界条件の扱いは上に記した通りだが，粗い格子上での海底地形をどのように表現するかには任意性がある。色々試行錯誤をした結果，ある粗い格子を構成する細かい格子全てが地中に相当する場合にのみその粗い格子も地中として扱い，わずかでも海洋内部領域を含むような格子は全て，粗い格子系においては海洋内部を表す格子とみなすようにするのが最適であった。

多重格子法では，各格子レベルにおいて古典的な定常反復法と同様の緩和演算を行うことにより対応する波長の誤差を減衰させるが，その際にどのような手法を用いるかも収束率や計算効率を左右する重要な要素である。 広く採用されているのは単純な Gauss-Seidel 法による緩和演算を複数回行う手法であるが,これは計算順序が 規定されベクトル化や並列化に不適である。 その欠点を補う手法として red-black 法(格子をチェッカーボードパターンによって二つのグループに分割し，各々のグループ内では各格子で並列に緩和演算を実行する）も多く採用されているが，収束率はあまり良くない。3 色以上のグループ分けを行うことによってさらなる効率向上を目指す多色スキームも提案されているが，本研究では異なるアプローチを採用した。緩和演算法を選択するということは緩和演算子として係数行列 L の近似逆行列 M をどのように設定するかに他ならないのだが,我々はこの M の決定に Sparse Approximate Inverse (SAI) と呼ばれるテクニックを用いた。SAI ではまず M の非ゼロ要素の配置を適当に定めておき，その値をノルム $|| I - ML ||^2$ が最小となるように最小二乗問題を解いて決定する。M の非ゼロ要素の配置の選択には任意性があるが，ここでは L と同じパターンを持つとした。これは Grote and Hucke (1997)において SPAI-1 と呼ばれるアルゴリズムである。M が L と同じ非ゼロ要素の配置を持つと仮定することは，ある格子での緩和演算において近傍 7 点の情報のみしか用いないということを意味する。多重格子法における緩和演算は各格子レベルで表現される最も高波数の誤差のみを効率的に減衰されることが要求されるのであるから，M の構造に上記のような局所性を持たせることは理にかなっている。さらにこの特性は並列化時にも通信を最小に留めるという点で大きな利点である。

SPAI-1 アルゴリズムによる緩和演算子は収束率及び並列化効率の双方において満足な結果が得られたが，初期化に時間がかかるという欠点がある。海洋モデルが扱うのは時間発展問題であり，同じ係数行列の線形問題を何度も繰り返し解くのであるが，海面の昇降が時々刻々変化する自由表面モデルでは鉛直格子幅が時間変化するため，係数行列 L もタイムステップ毎に変化する。よってタイムステップ毎に緩和演算子 M の初期化も必要なように思えるが，海面昇降の時間変化は高々数 10cm と格子幅に比べ短く，初期の海面昇降から計算した M が任意の時刻においても L の近似逆行列として十分機能する。したがって自由表面による鉛直格子幅の時間発展を伴う海洋モデルにおいても，タイムステップ毎に L の再構築は必要であるが M の初期化は初回に一度行うだけで良く，緩和演算子に SAI を採用することにおける初期化コストの欠点は隠蔽される。

先には明示しなかったが，境界条件としてノイマン条件を用いている場合，係数行列 L は対称となる。この対称性を生かし，反復解法として多重格子法を単体で用いるのではなく，これを共役勾配法の前処理として用いることができる。この手法は MGCG と呼ばれ，多重格子法単体よりも収束率が高い。元来 CG 法は並列計算に適したアルゴリズムであるが，多重格子法を前処理に用いることによって収束までの反復回数が格子数に伴い増加するという欠点も解消される。したがって粗い格子の構築方法に上述のような工夫をし，緩和演算子として SAI を採用した多重格子法を前処理として用いた CG 法は非静力学海洋モデルで用いるのに理想的な Poisson 解法であるといえる。この手法を用いれば 収束判定に用いる相対残差の閾値を 10^{-8} とした場合の収束までの反復回数は矩形領域の場合で 5 回以内，複雑な地形が存在する状況においても 8 回程度に留まっており，しかもそれは格子数が増加しても変化しない。この Poisson 解法を用いる我々が開発した非静力学モデルの計算コストは，静力学近似を行うモデルのコストの２倍以下に留まっており，海洋非静力学モデルの適用可能性が著しく広がることになる。

若手利用者推薦とは異なるが，HA8000 導入に合わせて昨年度に実施された T2K オープンスパコン HPC 特別プロジェクトに関連して，短時間ではあるが HA8000 の 512 ノードを使用してテストを行う機会が得られたので，上記の Poisson ソルバのスケーラビリティをチェックした。HA8000 は 4 コア CPU が 4 つの計 16 コア搭載したノードが 512 台連結されているので，最大 8192 コアによる並列計算ができる．解いた問題サイズは $N = 2880 \times 3840 \times 160 \sim 1.8 \times 10^9$ である。結論から言うと，並列ノードを増やすことによる性能劣化は全く見られず，少なくともこの程度の格子数の問題であれば極めて高いスケーラビリティを保っていることが確認できた。ただし HA8000 上で詳細なプロファイル測定を行ったところ，多重格子法内での行列ベクトル積を計算するルーチンで並列分割数によらず一様な実効性能の低下が見られた。ここでは演算命令とロードストア命令の比が 1 に近く，メモリ帯域が飽和して性能を制限してしまっていた。係数行列 L 及びその緩和演算子 M はそれぞれ $N \times N$ 行列であり，各行 7 の比ゼロ要素があるから 7N の独立した変数から成っている。しかし我々が採用している格子系においては，例えば鉛直格子幅が一定かつ海岸線が断崖絶壁のように鉛直に一様であれば，モデルの鉛直各層での係数の値が等しく，L 及び M の要素中に同じ値をとるものが多く存在することになる．この共通部分を適切に抽出して演算中に再利用するようにすれば，キャッシュを有効活用してパフォーマンスが改善されることが期待される。

なお開発したソルバのコードは Fortran90 の module によって海洋モデルとは完全に分離されており，他の用途にも利用可能である．もしその利用に興味のある読者がいれば，筆者

<ymatsu@ccsr.u-tokyo.ac.jp>に連絡いただければコードの提供が可能である。線形ソルバのコードはフリーなものも含め数多く流通しているが，完全に並列化された多重格子法による 3 次元 Poisson ソルバというのはそう何処にでも落ちているものでないと思われるので，有用な場合もあろうかと考えている。利用は直交する座標系(水平の座標軸は曲がっていても良く，格子幅も一定である必要もない。もちろん単純な FFT 実装のように格子数を 2 のべき乗に制限する必要もない)による構造化格子に制限されているが，任意の領域形状においてノイマン型境界条件を許容するという特徴もあるので,もしこれらの条件に適合する問題を解くためのソルバを探しているならば是非利用して頂いて感想等をお聞かせ願えると光栄である。もちろん筆者のような数値計算の素人が書いたコードであるので，その道の専門家によるものに比べれば性能的にも遥かに劣るであろうが，利用者からのフィードバックによって洗練させることができれば我々の本業にとっても有意義であり，幸いである。

３．ウェッデル海での棚氷水の沈降に関するシミュレーション

（１）はじめに

南極沿岸では活発な海氷生成によるブライン排出を起源とする高密度水が深層に沈み込んでおり，これが南極底層水の起源になっている。このような深層水形成は海洋大循環を駆動する要因の一つであるから，定量的な理解は地球の気候を論ずる上でも重要である。沿岸で形成された高密度水は，まず沿岸付近の大陸棚上に溜まる。これが全世界の海洋の深層を占める高密度水になるまでには，大陸棚上から大陸斜面上へと流出し，大陸斜面上を深層まで下降しなければならない。このような高密度水が重力の作用によって斜面を下る流れのことを重力流と呼ぶが，地球という回転系においてはコリオリ力が作用するため，高密度水は重力にまかせて単に斜面を下るとのではなく，斜面を降る向きの重力と岸向きのコリオリ力がバランスすることによって，斜面上を等深線に沿って（南半球では岸を左に見ながら）流れる傾向がある。したがってただ大陸棚上で高密度水が生成されるというだけでは，深層水が形成されるとは言えない。重力流の沈降のためには，コリオリ力と重力の間のバランスを崩す何らかの作用が必要となる。

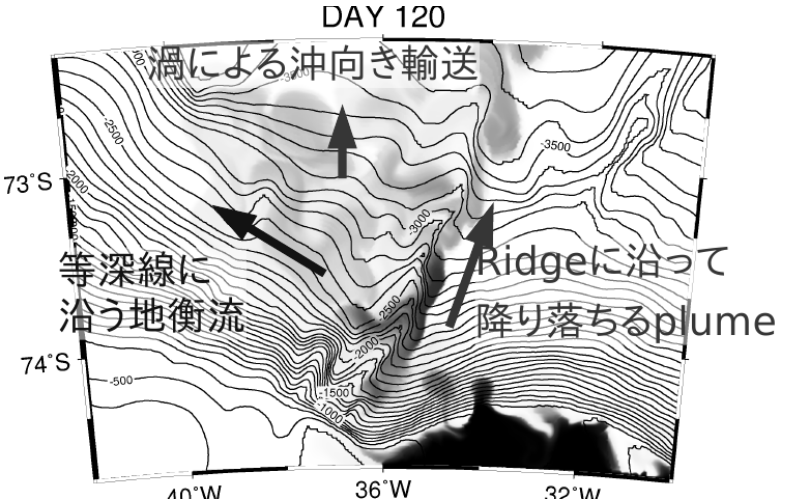
そのバランスを崩す要因として海底摩擦の効果，密度フロントでの傾圧不安波動の発達に伴う渦による輸送効果，斜面上の地形起伏の影響など様々なものがこれまでに指摘されている。しかしこれまでの研究ではそれらの個々の要素が理論的あるいは理想化設定の数値シミュレーションとして扱われるばかりであり，実際の海洋においてどのプロセスがどの程度働くのかについてはほとんど調べられていない。

本研究では，ウェッデル海における深層水形成の主要な起源となっているフィルヒナー流出を対象に，現実的設定のもとで高解像度数値シミュレーションを実施する。ウェッデル海の奥部は棚氷によって覆われており，その棚氷の端には冬季に沿岸ポリニヤが形成される。ここでは活発な海氷生成に伴って高密度水が生成されるが，直下の大陸棚上に落ちた高密度水は即座に外洋に向かうのではなく，一度棚氷の下にもぐりこみ，そこを循環して再び棚氷の外に出る。棚氷の下面は結氷温度に保たれているが，海面よりも数百 m 以上深くに存在するため，圧力の影響によって海面で実現される結氷温度よりも低くなる。そのため棚氷下から再び外に出た海水は非常に低温であるという特徴を持ち，このようにして形成される低温の高密度水は棚氷水と呼ばれる。この棚氷水は海底地形のくぼみに沿って大陸棚の端へと達し，大陸斜面上へと

局所的に流出する。流出した棚氷水はコリオリ力の作用のために大陸斜面上をほぼ等深度線に沿って流れるが，西経 36 度付近に存在する小規模な海嶺に当たって進路を沖向きに曲げられ，等深度線を横切って沈降することが観測によって知られている(Foldvik et al., 2004)。本研究では海底地形起伏の棚氷水の沈降にどの程度重要かを明らかにし，この海域での深層水形成量を定量的に評価することを目的とする。

（2）シミュレーションの設定と結果

用いる数値モデルは第 2 節で述べた非静水圧海洋モデルである。シミュレーションの対象領域は，ウェッデル海の南岸付近，西経 30-42 度，南緯 71-75 度の領域である。水平解像度はおよそ 900 m，鉛直解像度は 25 m である。初期状態として観測に基づく水温・塩分の鉛直分布を水平一様に与える。 領域南東の棚氷水の流出口となる窪地内 500 m 以深を水温-2℃・塩分 34.7 psu に緩和し，継続的な棚氷水供給を表現する。また沈降する棚氷水の流路および流量を追跡するために，この緩和領域において値を常に 1 とし，それ以外の場所では初期値を 0 とす



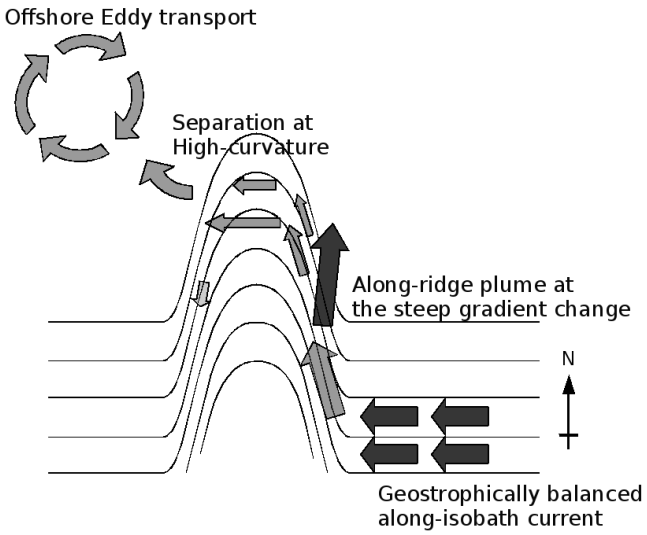
第 1 図：120 日目での最下層格子での仮想トレーサー濃度.

る仮想トレーサーを導入する。この仮想トレーサーは水温や塩分と同じ輸送・拡散方程式を用いて値が予報される。初期状態から 240 日の計算を行い，解析に用いる。

実験 120 日目における海底の仮想トレーサー濃度を第 7 図に示す。フィルヒナー陥没の出口で供給された高密度水は，大陸斜面上をまず西向きに流れ，海嶺にぶつかる場所で手前にある谷線に沿って下降している。こうした全体的な様相に加え，直接観測が存在する場所での水温・塩分分布や流速は，観測とよく整合している(Foldvik et al., 2004)。仮想トレーサー分布からわかる通り，高密度水の主要な流路は海嶺の手前を下降するものだが，海嶺の先端を回り込んで等深度線に沿ってさらに西向きに流れる成分も存在する。また海嶺の先端からは渦が切り離され，これもまた高密度水を沈降させる働きを持つことがわかる。

（3）海底地形の起伏の効果

斜面上において重力とコリオリ力の間のバランスだけを考慮する場合，コリオリパラメータ

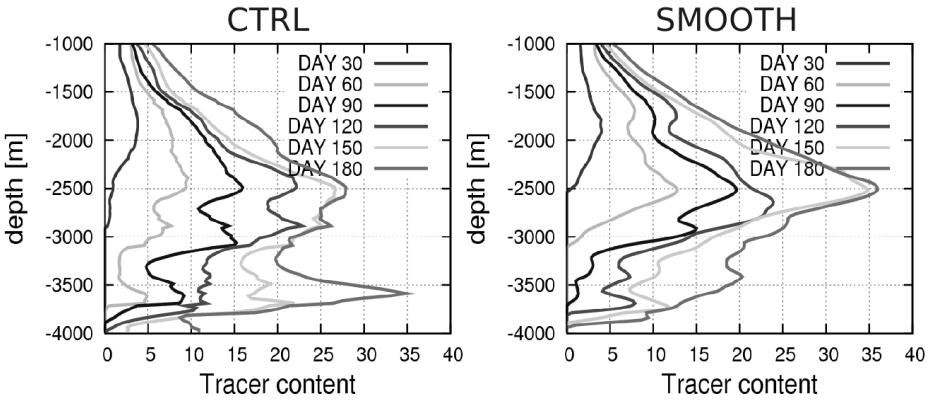


第 2 図：高密度水が等深度線を横切るメカニズムの概念図.

f, 等深度線に沿う流速 U, 斜面の勾配 α, 重力加速度 g, 高密度水の密度 ρ, 高密度水と周囲水の密度差 $\Delta\rho$ に対して $fU = -g\alpha\Delta\rho/\rho$ が成り立つ。この場合，任意の f, α, ρ, $\Delta\rho$ に対して，バランスを満たす U が存在する。しかし海嶺が存在するなどし等深度線が曲率を持っている場合には，等深度線に沿う流れのバランスの中に遠心力の項が加わる。曲率半径を R とするとき(等深度線が沖に向かって凸の場合を正とする)，バランスの式は $fU + U^2/R = -g\alpha\Delta\rho/\rho$ となる。このバランスを満たす U が存在するためには，R に下限値 $R_c = 4g\alpha\Delta\rho/\rho f$ が存在する。これよりも小さい正の曲率半径を持つ場合，等深度線に沿う流れは斜面に沿うことができず，剥離して外洋側へと向かう。外洋に向かう過程では高密度水層が上下に引き伸ばされることになるため，ポテンシャル渦度保存より高密度水層は f と同じ符号の渦度を獲得する。フィルヒナー流出における典型的な条件で R_c を見積もるとおよそ 0(1) km となり，海嶺先端の曲率半径とよく対応する。したがって，第 1 図に見られた海嶺先端から切離する渦は，このような理由によってできたものだと考えられる。

一方海嶺の手前の部分は曲率が負であり，遠心力は流れを斜面に押し付ける方向に働くため，高密度水の沈降を阻害する要因として働く。しかしこの領域では局所的に等深度線が密集しているため，等深度線に沿って進むにつれて斜面の勾配が急になる。したがってその場所を通過する高密度水に作用する重力の斜面下向き成分は短時間で急激に大きくなる。一方コリオリ力が重力とバランスするには 1/f の時間がかかることから，斜面勾配の微分が局所的に十分大きければ，コリオリ力がバランスするより前に増大する重力の影響で高密度水は斜面を降り落ちていくことになる。ここでは具体的な式は割愛するが，その結果として等深度線の曲率・コリオリパラメータ・斜面勾配それぞれについて等深度線に沿って考えた微分が重要となり，それらの大小の兼ね合いによって沈降が生じるかどうかが決まる。これについてもフィルヒナー流出における典型的な条件で評価したところ，海嶺の手前は確かに高密度水の沈降が生じるべき場所にあたっていた。フィルヒナー流出の沈降について，地形起伏の力学的影響を概念的にまとめたものを第 2 図に示す。

仮想トレーサーの量を各深度で水平積分することにより，フィルヒナー流出で与えられた高密度水がどの深度に供給されるかを調べた結果を第 3a 図に示す。分布は 2500–3500 m にわたって幅広いピークを持ち，高密度水の一部は 4000 m の深さまで輸送されていることがわかる。一方モデルの地形を平滑化し， 海嶺を取り除いて行った実験の結果では，高密度水は深度 2500



第 3 図：各深度で水平積分した仮想トレーサー量.

a）標準のシミュレーション結果，b）海嶺を取り除いたシミュレーション結果.

m を中心に分布し，3000 m より下にはほとんど輸送されていない(第 3b 図)。海嶺が存在しない場合に高密度水の沈降を担うものは，海底摩擦と傾圧不安定の影響によるものと考えられる。今回の対象領域においては，高密度水が深層に沈降する要因として，海底起伏の効果が海底摩擦や傾圧不安定の効果を大きく上回ることが示された。

４．おわりに

海洋モデルの力学コアは，考慮すべき支配方程式の複雑さの観点で見れば，例えば水蒸気の相変化や微小なエアロゾルによる光学的影響まで扱う必要のある大気モデルに比べ，ずいぶん簡単な問題だといえる。また工学分野で用いられる数値流体コードと比較しても，単純な構造化格子で事足りるという点で間違いなく易しい部類の問題だと言っていい。一方解くべき問題の規模は膨大であり，計算機資源は幾らあっても十分ということはない。支配方程式や格子配置が単純であればこそ，アルゴリズムの選択や明示的・非明示的な最適化チューニング，　並列分割方法の変更等によるパフォーマンスの向上は著しく，そういう意味では計算機ユーザーあるいはプログラマーにっとて非常にチャレンジングな問題であるとも言える。利用可能な計算機資源が限られている中で，コードの処理速度が向上するということは，より高い解像度・広い領域・長い積分時間のシミュレーションが可能となるということを意味し，即ち理学的な成果にも直結する。

ところで海洋学を含む地球物理学という学問は，地球シミュレータの存在に象徴されるように，先端の高性能計算機利用者の少なくない部分を占めている。しかし多くの地球物理研究者はモデルのユーザーとしてシミュレーション結果を解析したり，あるいは自分の興味対象である物理現象をコードに追加したりはするが，コードの数値的パフォーマンス自体にはあまり注意を払ってこなかったように思える。他の分野と比して膨大な計算機資源を使用しているにも関わらず，である。本稿で述べた研究においては，多重格子法による Poisson ソルバーという，工学分野では 10 年以上前に導入され成果を上げているいわば枯れた手法を導入するだけで劇的に処理性能が向上し，これまで困難と思われていた実験が可能となった。同様のことは他にも数多くあり得るはずで，我々地球物理の研究者がもう少し視野を広げて計算工学に興味を持って接するだけで，今まで計算機資源を食いつぶしていたボトルネックが解消され，不可能だった計算が可能となり，結果として新たな理学的発見をもたらすことにもなるのではないかと

期待している。特に今後の大型計算機の性能向上は未曾有の超高並列化によって実現されていく事を鑑みれば，地球物理分野が今後も継続して使用する計算機資源に見合った成果を挙げていくためには，積極的に最新の工学的知見を取り入れて，大幅なアルゴリズムの変更まで考慮しながら並列化に適するようモデルコードを改良していかねばならないとの決意を新たにする次第である。

謝辞：まず原稿の入稿が大幅に遅延してしまい，関係者各位に多大な迷惑をお掛けしましたことをお詫び致します。若手利用者推薦に採用していただいた事に関しては，本研究で博士論文を執筆したということもあり，大変有難く利用させていただき，有効に活用することができました。また中島研吾教授にはモデル開発時から多重格子法に関するアドバイスや，HA8000 導入時にはシステムに関する詳細な情報を頂くなど大変お世話になりましたので改めて謝辞を述べさせていただきます。

参 考 文 献

Grote, M. J. and T. Huckle (1997)：Parallel Preconditioning with sparse approximate invers, *SIAM J. Sci. Comp.*, **18**, 838-853

Foldvik, A., T. Gammelsrød, S. Østerhus, E. Fahrbach, G. Rohardt, M. Schröder, K. W. Nicholls, L. Padman and R. A. Woodgate: Ice shelf water overflow and bottom water formation in the southern Weddell Sea, *J. Geophys. Res.*, **109(C2)**, C02015

Marshall, J., H. Chris, L. Perelman and A. Adcroft (1997a)：Hydrostatic, quasi-hydrostatic, and nonhydrostatic ocean modeling, *J. Geophys. Res.*, **102**, 5333-5752

Marshall, J., A. Adcroft, C. Hill, L. Perelman and C. Heisey (1997b)：A finit-volume, incompressible Navier-Stokes model for studies of the ocean on parallel computers, *J. Geophys. Res.*, **102**, 5753-5766

Matsumura, Y. and H. Hasumi (2008)：A non-hydrostatic ocean model with a scalable multigrid Poisson solver, *Ocean Modelling,* **24**, 15-28.

Matsumura, Y. and H. Hasumi (2009a)：Modeling Ice Shelf Water overflow and bottom water formation in the southern Weddell Sea, *J. Geophys. Res,* under review.

Matsumura, Y. and H. Hasumi (2009b)：Dynamics of cross-isobath dense water transport induced by slope topography, *J. Phys. Oceanogr.,* under review.

東京大学情報基盤センター・スーパーコンピューティングニュース
Vol.12 No. Special Issue 1 (2010.2)

編集・発行　　東京大学情報基盤センター
スーパーコンピューティング部門

〒113-8658　東京都文京区弥生 2-11-16
（電話）　03-5841-2717（ダイヤルイン）
（FAX）　03-5841-2708