Vol.13, No.2 2014

全国共同利用版

広報

センター共同研究報告（平成25年度）

ISSN 1347-3581

京都大学学術情報メディアセンター
Academic Center for Computing and Media Studies, Kyoto University

Vol. 13, No.2 号の発刊に当たって

京都大学学術情報メディアセンター
深沢　圭一郎

本号では、学術情報メディアセンターの平成 25 年度共同研究報告について特集いたします. 平成 25 年度では若手研究者の支援事業に関して 12 件，プログラム高度化支援事業について 5 件，大規模計算の支援事業について 1 件の共同研究が実施され，それぞれの報告が掲載されています. 数値計算に関する様々な報告が掲載されていますので, 計算機を利用される研究者や学生の方の良い参考になればと思います.

若手研究者の支援事業は，40 歳未満の若手研究者(学生を含む)を対象とした利用者向けの奨励研究制度で, 優れた研究提案でスーパーコンピュータを利用することで学術的にインパクトがある成果を創出できると期待される課題に対して計算機利用負担金の全部または一部を本センターが補助しています．今回報告のあった 12 件には化学物性や流体計算，行列分解，経済，物理など多様な研究分野が含まれており，また，北海道大学，一橋大学，山梨大学，京都大学、大阪大学，岡山大学と関西だけで無く全国の若手研究者との共同研究となりました.

プログラム高度化支援事業では, スーパーコンピュータをグループコースまたは専用クラスタコースでご利用の研究グループを対象にプログラムの制御構造･データ構造の改良による性能チューニングや並列化手法の改良による高度化・高性能化(共有メモリ型プログラムの分散メモリ並列化や共有メモリ/分散メモリ階層並列化など)，問題分割・負荷分散方式などの改良による高度化・高性能化といった大規模計算プログラムの高度化・高性能化を支援しています．平成 25 年度では，流体計算、行列分解やコンクリート・地面の構造解析といった研究分野のアプリケーション高度化を支援させていただきました．ノード間通信の効率化や GPU での計算効率化、ハイブリッド並列実行の効率化など様々な高度化･高性能化があり，数値計算研究者にとって最適化の良い指標となると思います.

大規模計算の支援事業ですが，スーパーコンピュータをグループコースまたは専用クラスタコースでご利用の研究グループを対象に最大で 128 ノード×2 週間の大規模ジョブコース利用支援を行っています．今回は，MD（Molecular Dynamics）計算では計算量が膨大になるため現在は現実的に複雑な流体解析に利用が難しい問題に対して，流体計算と MD 計算を繋ぐような Synchronized Molecular Dynamics 法を開発し, 適用した計算への支援報告となっています. 大規模並列計算機と親和性の高い手法となっており，今後の発展がさらに期待されます.

また，本号では並列プログラミングチュートリアルについても掲載されています．Cray Reveal と呼ばれるクレイ社製コンパイラがソースコードを解析して得られた情報と Cray Pat と呼ばれる性能解析ツールによるプロファイル情報を組み合わせて OpenMP プログラミングを支援する開発ツールの利用方法が紹介されています． GUI で性能解析、OpenMP 指示行を自動生成することができるツールとなっており, OpenMP 初心者から OpenMP 性能最適化を行うエキスパートの方まで広く利用ができます.

今後も皆様の研究，教育にご活用いただけるようにセンター教職員も尽力していきますので，今後ともご利用，ご支援のほど、よろしくお願いいたします.

円形液膜内非定常マランゴニ対流の解明

山本　卓也

大阪大学大学院基礎工学研究科物質創成専攻

1　緒言

マランゴニ対流は表面張力勾配によって駆動される流れであり、工学プロセスの一部である乾燥、半導体結晶成長等で見られる。この流れは多くの場合は浮力対流と共存することによってその流れの影響が小さくなるが、浮力対流の存在しない微小重力環境下で、マランゴニ対流は重要となるため、数多くの研究が行われている[1-3]。

宇宙飛行士の Donald Pettit 博士は 2003 年に"Saturday Morning Science"という一般向けの実験を国際宇宙ステーション内で行った[4]。その中に、円形の液膜を用いたマランゴニ対流の実験が行われたが、その実験結果は今まで説明されてきた現象と異なっていたため数々の研究が行われてきた[5-7]。さらに、2012 年に再び Donald Pettit 博士によって国際宇宙ステーション内で実験が行われた[8]。この実験では、円形液膜内マランゴニ対流に関する複数の実験が行われ、非定常振動マランゴニ対流が液膜内で発達すると報告された。しかし、このように液膜内で振動流が報告された例は少なく、その振動流に遷移するメカニズムは分かっていない。

本研究では、液膜内でマランゴニ対流の振動流へと遷移した原因を解明するため、数値流体解析(CFD)を行った。

2　数値解析手法

数値計算に用いた計算領域の概略図を Fig. 1 に示す。液膜形状は固-液-気の三重点で固定されているとした。ここで、液膜形状の静的変形が大きい場合に、特に液膜内マランゴニ対流が発達する[7]ので、本研究では凹み形状が大きい場合に着目した。また、気液界面形状は Young-Laplace 式によって決まるとし、流れによる動的な気液界面変形は考慮しない。支配方程式は Navier-Stokes 式、連続式、energy 式であり、計算にはフリーソースである OpenFOAM[9]を用いて計算を行った。気液界面上の境界条件として温度勾配により惹起されるマランゴニ対流の境界条件と断熱条件を課した。物性値としては 293 K の水の値を用い、表面張力を除き物性値は温度に依存しないとする。その他の詳細な数値解析手法については著者らの文献[7]を参照されたい。

3　結果と考察

振動流に遷移した数値計算結果を Fig. 2 に示す。温度分布より左右対称な振動流が発達したことがわかる。次に、その振動流が発生したメカニズムを議論する。水を用いた場合の液膜内マランゴニ対流では、Pr 数が大きいため($Pr = 6.96$)、液膜の縁付近で温度勾配が大きくなる。そして、液膜の縁で駆動された流れは液膜中心に向かって流れるが、液膜中心部では液膜厚みが薄いため、そこで流体が加速される[7]。そして、液膜中心部で加速された流れは、加熱点と逆の液膜端に到達し、液膜の縁に沿って流れが加熱点に再び戻る(Fig. 2 (a))。そして、高温な流体が加熱点に戻る(Fig. 2 (c))と振動流が発生する。このようにして、振動流が発達したと考えられる。高温部に流体が戻ってきたときに、液膜中心向きの流れが絞られる(Fig. 2 (d))。そして、再び液膜中心に向かう流れが押し戻す。これらを繰り返すことによって振動流が発生したと考えられる。このような不安定性の発現は V. M. Shevtsova and J. C. Legros の液柱内マランゴニ対流の数値計算結果で"cold finger"が気

液界面にぶつかることによって生じると不安定性[10]と類似していると考えられる。また、この振動流が左右対称であったのは左右対称に加熱をしたため、加熱点と反対の液膜端に到達したときに左右対称に分岐し、左右対称に加熱点まで戻ってきたためであると考えられる。さらに、境界で与える熱量を増加させる、もしくは、液膜の凹み具合がさらに大きい場合に、非線形性が増加し、左右対称性が崩れると期待されるが、現段階ではその数値計算をするに至っていない。

4　結言

液膜内でのマランゴニ対流の振動流に遷移するメカニズムを解明するため数値計算を行い、左右対称な振動流が熱的な不安定性により生じることが分かった。宇宙飛行士の Donald Pettit 博士が行ったように、左右非対称な振動流に遷移するメカニズムに関しては今後更なる研究が必要である。

引用文献

[1] M. Lappa, *Fluids, Materials and Microgravity: Numerical Techniques and Insights into Physics*, 1st edition (Elsevier Science, Oxford, U. K., 2005).

[2] H. Kawamura *et al.*, Int. J. Heat Transfer, **134**, 031005 (2012).

[3] S. Matsumoto *et al.*, Int. J. Microgravity Sci. Appl., **31**, S51-S79 (2014).

[4] D. Pettit, Saturday Morning Science Videos, 2003 see http://mix.msfc.nasa.gov/IMAGES/QTVR/0601211.mov.

[5] H. Kawamura *et al.*, J. Jpn. Soc. Microgravity Appl., **23**, 157-160 (2006).

[6] I. Ueno *et al.*, Proc. of the second Conference on Microfluidics, Toulouse, 10-203 (2010).

[7] T. Yamamoto *et al.*, Phys. Fluids, **25**, 082108 (2013).

[8] D. Pettit, Thin Film Physics, 2012 see http://physicscentral.com/explore/sots/episode3.cfm.

[9] The open source CFD toolbox, OpenFOAM (see http://www.openfoam.com).

[10] V. M. Shevtsova and J. C. Legros, Phys. Fluids, **10**, 1621-1634 (1998).

謝辞

本共同研究制度(若手研究者奨励枠)を活用させて頂いたことを、この場を借りて厚く御礼申し上げる。また、本研究の一部は科学研究費補助金・基盤 B（課題番号 23360343, 25289087）の援助を受けた。

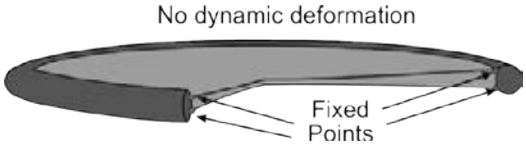


Fig. 1 Schematic of thin liquid concave film.

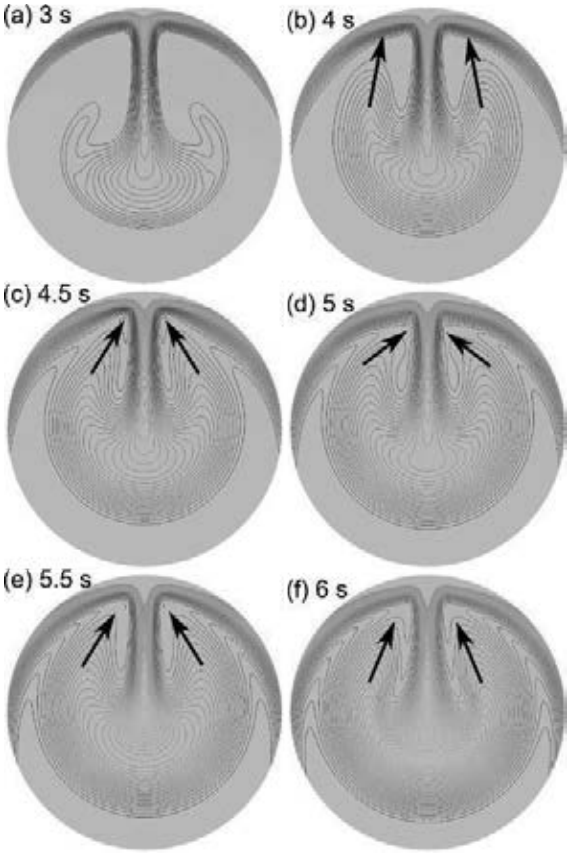


Fig. 2 Time development of temperature distribution along a free surface: (a) 3 s, (b) 4 s, (c) 4.5 s, (d) 5 s, (e) 5.5 s, and (f) 6 s, the temperature increment is 0.333 K and the minimum temperature contour expresses 293.1 K.

大規模スケールでのブロック共重合体薄膜の自己組織化形状予測

吉元　健治[1,2]

[1]京都大学学際融合教育研究推進センター、[2]京都大学大学院工学研究科 化学工学専攻 材料プロセス工学研究室

1　緒言

近年、半導体製造分野では、ブロック共重合体を用いた誘導自己組織化(DSA: Directed Self-Assembly)とよばれる新たなパターニング技術が注目されている。ブロック共重合体は、化学特性が異なるポリマーが末端で化学結合した化合物で、基板表面に塗布・熱処理するだけで、ナノメートルオーダーの規則パターンにミクロ相分離(自己組織化)するという利点を持つ(図 1)。この自己組織化したジブロック共重合体のドメインの位置や配向を、予め基板上に施した物理的・化学的なパターンによって制御する手法がDSA である。

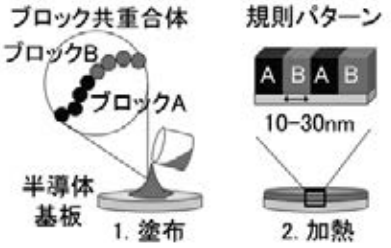


図 1 基板上でのブロック共重合体の自己組織化

一方で、ブロック共重合体が自己組織化する際、大規模な欠陥構造が生成されることが大きな問題となっている(図２)。そこで本研究では、欠陥構造の生成メカニズムを解明するために、基板上でのブロック共重合体の自己組織化を大規模かつ動的にシミュレーションできる簡易化モデルの構築に取り組んだ。

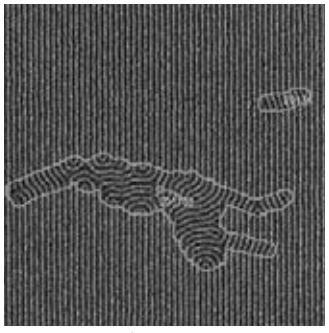


図 2 誘導自己組織化プロセスで基板上に形成したジブロック共重合体の欠陥構造(SEM 画像、IBM 提供)

2　モデルとシミュレーション手法

シミュレーションには、ブロック共重合体の濃度場を変数とする Ohta-Kawasaki モデルを用いた[1]。バルク系の自由エネルギーに、ブロック共重合体と基板との相互作用エネルギーを加えて系全体のエネルギーを表現し、そのエネルギーが最小となるように、ジブロック共重合体の濃度場を Time Dependent Ginzburg Landau 式を用いて時間発展させた。その濃度場の時間発展に用いた逐次計算式を以下に示す。

$$\eta(\boldsymbol{r},t+\Delta\tau) \approx \eta(\boldsymbol{r},t) + \Delta\tau\left|\nabla^2\mu_s(\boldsymbol{r},\tau) \quad \alpha\eta(\boldsymbol{r},\tau)\right| \quad (1)$$

where

$$\mu_s(\boldsymbol{r},t) \quad \frac{1}{\xi^2}\eta(\boldsymbol{r},t) + g\eta(\boldsymbol{r},t)^3 \quad \nabla^2\eta(\boldsymbol{r},t) + \Lambda(\boldsymbol{r}). \quad (2)$$

ここで、$\eta(\boldsymbol{r},t)$は、時刻 t での、位置 $\boldsymbol{r}$ の局所体積内のブロック A とブロック B の体積分率差 (=オーダーパラメータ)、$\Delta\tau$ は時間のステップ幅を表している。３つの係数(α, ξ, g)はジブロック共重合体の物性などから計算される。式(2)は化学ポテンシャルμ_sの計算に用いられ、Λは基板とブロックとの相互作用強度を表す。逐次計算では、(ⅰ)現在の濃度場$\eta(\boldsymbol{r},t)$に対して、化学ポテンシャル$\mu_s(\boldsymbol{r},t)$を式(2)から計算し、(ⅱ)式(1)より$\eta(\boldsymbol{r},t)$を時間ステップ幅$\Delta\tau$分アップデートする。(ⅲ)$\eta(\boldsymbol{r}, t+\Delta\tau)$を現時刻の濃度場して、(i)と(ii)の計算を繰り返す。

これらの逐次計算は比較的容易に並列化できる(図３)。系を計算機コアの数に分割し、各分割ブロック内で式(i)~(iii)の逐次計算を行う。ただし、式(1)と式(2)に含まれる空間２回微分∇^2の計算には中心差分法を用いるため、隣接する分割ブロック間でηとμ_sの境界値を交換する必要が生じる。したがって、系が比較的小さい場合は、境界値のデータを交換する時間の割合が大きくなり、並列化の計算効率が減少してしまう。

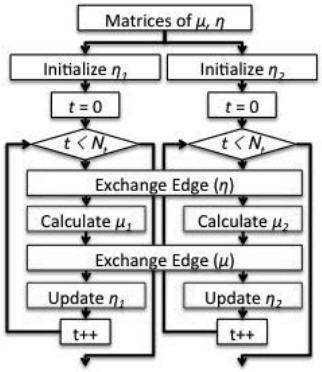


図 3 逐次計算の並列化の概略フロー図[2]

3 結果

3.1 大規模計算へのスケーラビリティ

まず、64 コア（8 コア/ノード x 8 ノード）を用いて、2 次元バルク系での対称性ジブロック共重合体の自己組織化シミュレーションを行った。並列化計算には MPI を使用した。図 4 に、10^5回の逐次計算を行うのに、64 コアで要した計算時間と 1 コアで要した時間の比（＝スピードアップ比）をまとめている。系のサイズが大きくなるに従い、各分割グリッドでの「η と μ_sの計算時間」が「境界値の交換に要する時間」よりも大きくなり、スピードアップ比が向上している。グリッド数が 2048x2048 の系では、スピードアップ比は約 55 で、理想値（=64）に近づいている。従って、大規模な系に対しても、計算機のコア数を増やせば、約コア数分計算速度も上げられる。

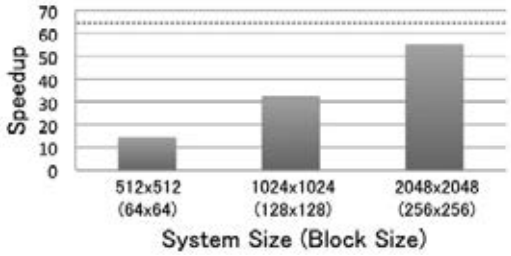


図 4 並列化（64 コア）による計算速度の向上[2]

3.2 誘導自己組織化のシミュレーション

次に、基板上で対称性ジブロック共重合体の自己組織化シミュレーションを行った。基板と各ブロックとの親和性が同等な場合、対称性ジブロック共重合体は、自然周期 L_0のラメラ構造を形成した。一方で、片方のブロックとの親和性が高い化学ガイドパターン（幅 $L_0/2$, 周期 $2L_0$）を基板上に予め作成しておくと、最初は無秩序なラメラ構造が形成されるが、徐々に化学ガイドパターンに引き寄せられ、最終的には全てのラメラ構造が同じ方向に規則的に配列する様子が観測された(図 5)。この欠陥構造の動的挙動は、化学ガイドパターンとジブロック共重合体との相互作用によっても大きく変化するため、現在は実験データ（例：図２）との比較検証を行っている。

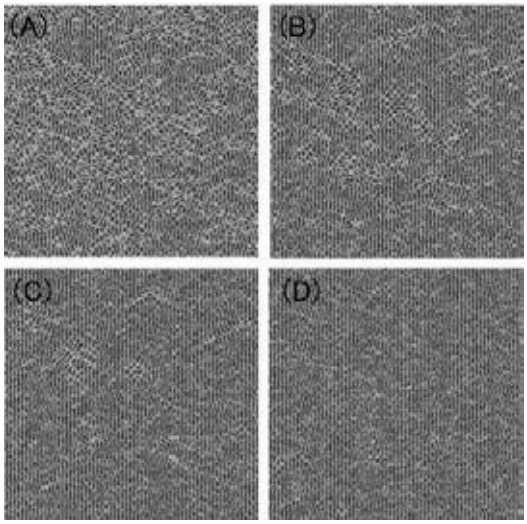


図５ 誘導自己組織化プロセスにおける欠陥パターンの時間変化[2]：(A) t=4, (B) t=8, (C) t=12, (D) t=24 [τ]。

4 結言

本研究では、ジブロック共重合体の誘導自己組織化プロセスで発生する大規模欠陥の発生メカニズムを解明するために、簡易化シミュレーションモデルを構築した。並列化により計算が高速化され、大規模な欠陥構造の動的挙動も予測できることが確認できた。今後は実験データとの比較検証等を予定している。

＜謝辞＞本共同研究制度（若手奨励枠）を活用させて頂きましたことに大変感謝致します。

[参考文献]

[1] T. Ohta and K. Kawasaki, “Equilibrium morphology of block copolymer melts,” *Macromolecules*, **19**(10), 2621-2632 (1986).

[2] K. Yoshimoto and T. Taniguchi, “Large-Scale Simulations of Directed Self-Assembly with Simplified Model,” *J. Photopolym. Sci. Technol.*, **26**(6), 809-816 (2013).

ジブロックコポリマーの誘導自己組織化シミュレーション

深渡瀬 健[1]　　吉元 健治[1,2]

[1]京都大学大学院 工学研究科 化学工学専攻 材料プロセス工学研究室、[2]京都大学学際融合教育推進センター

1 緒言

近年、半導体回路パターンのさらなる微細化・高密度化を実現するため、新たなパターニング技術の開発が進んでいる。その中で、「ブロックコポリマーを用いた誘導自己組織化」によるパターニングは、その簡便さとコスト面からも注目されている[1]。ブロックコポリマーは、化学特性の異なる２つのポリマーから構成され、ナノオーダーの周期構造を自発的に形成する（自己組織化）。周期構造の配向や場所を制御するために、予め基板表面を加工しておく方法が誘導自己組織化と呼ばれ、従来のリソグラフィでは作成することが困難な 10 ~ 50 nm の周期パターンを比較的容易に作成できるようになる。しかし、誘導自己組織化には欠陥構造が生成されやすいという欠点がある。この欠陥構造は実験による検出が困難なために、シミュレーションによる予測が有効と考えられる。そこで、本研究では、ホールシュリンクと呼ばれる誘導自己組織化プロセス[2]に対してシミュレーションを行い、欠陥構造の生成を抑制するためのプロセス条件を検討した。

2 ホールシュリンクと欠陥構造

ホールシュリンクプロセスの概要を図１に示す。(1) 従来のリソグラフィ工程により、基板上に直径約 60-80nm の円柱状の穴(ガイドホール)を作成する。(2) ガイドホール上に、ジブロックコポリマー(PS-b-PMMA) を塗布・熱処理する。ジブロックコポリマーの PMMA ブロック比率は 0.3 で、ガイドホール内で相分離後、ガイドホール中心付近で PMMA の円柱状ドメインを形成する。(3) PMMA ドメインを選択除去後、ガイドホールよりも小さな穴（直径約 20nm）が形成される。

一方で、(2)の工程の後、PMMA の円柱状ドメインがガイドホール底面まで到達できず、円柱ドメインと底面との間に PS 残膜がしばしば形成されてしまう（図２）。PS 残膜の厚さは、ガイドホールの形状、壁の PS と PMMA に対する親和性、などにより大きく変化すると考えられる。そこで、本研究では、ガイドホールの形状（高さ、直径、テーパー角）と壁の親和性を変化させて、各ケースに対して、PS 残膜の厚さを調べた。

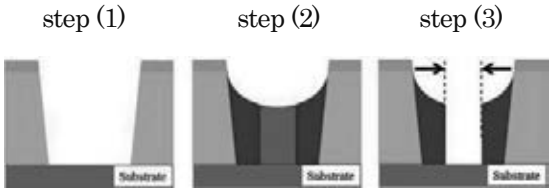


図 1 ホールシュリンクプロセス

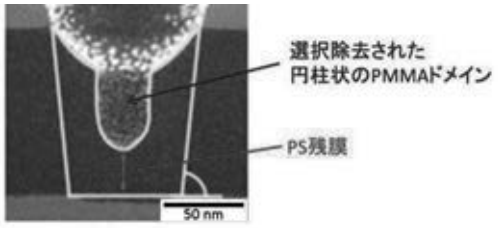


図 2 PMMA ドメイン除去後の断面 TEM 画像

3 シミュレーション

3.1 モデル

本研究では、ジブロックコポリマーのバルクエネルギーには、汎関数理論から導かれる、比較的簡易な Ohta-Kawasaki (OK) モデルを用いた[3]。ガイドホールとブロックコポリマーとの相互作用は、モデルパタメータを２つ含む簡単な制約項で表現し、各パラメータの値は実験データとの会わせ込みによって決定した[4]。また、動的シミュレーションを行うために、密度場の時間発展として時間発展型 Ginzburg-Landau 式を用いている。

3.2 ガイドホール内でのモルフォロジー

まず、ガイドホールの直径を変化させた場合、ガイドホール内でのジブロックコポリマーのモルフォロジーがどのように変化するかをシミュレーションにより調べた（図３）。ここでは、テーパー角とガイドホールの高さを 0° と 3.18 *Lo* に固定し、ホール直径を 1.59 *Lo* から 2.27 *Lo* まで段階的に大きくしている（*Lo*：バルク系でのミクロ相分離ドメインの１周期分の長さ）。ガイドホールの直径が大きくなるに従い、PMMA ドメインの形状が円柱状、円盤状、リング状へと変化する様子が観測される。このシミュレーション結果は、自己無頓着場理論(SCFT)を用いた先行研究の結果[5]と精度よく一致している。なお、本研究で用いた OK モデルでは、図３の各ケースの計算時間は全て数十秒～数分(16 コア並列化計算)であるが、SCFT では、数時間～数十時間を要する。したがって、本研究で対象とするホール系では、簡易化モデルを用いることで、短時間で比較的精度の高いモルフォロジー予測ができることがわかった。

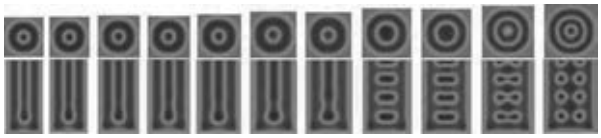


図３ガイドホールの直径とジブロックコポリマーのモルフォロジーの関係

3.3 PS 残膜の最小化

次に、ガイドホールの直径・高さ・テーパー角を変えて、PS 残膜の厚さを調べた。図４にガイドホール直径だけを変えた時のホール断面図を示す。ガイドホールの直径を大きくすると、PS 残膜厚さが徐々に小さくなるが、やがて PMMA シリンダーが断裂することが確認できる。図 5 にガイドホールの直径とテーパー角を変化させた場合の PS 残膜の等高線図を示す。PS 残膜厚さにはガイドホールの直径とテーパー角度により変化するものの、その最小値は約 15nm と依然大きな値であった。したがって、ガイドホールの形状を変えるだけでは PS 残膜を減少させることは難しいと考えられる。一方で、通常のプロセスでは、ガイドホールの側面と底面は PMMA への親和性がより高いが、底面のみ PMMA への親和性と PS への親和性を同等とした場合（中性化）、図 6 に示すように、PMMA ドメインがガイドホール底面に接触することが観測された。ただし、実プロセスでガイドホールの底面だけ化学特性を変化させるのは困難なため、現在は材料変数（例：ジブロックコポリマーのブロック組成比）を調節することで、PS 残膜を効果的に減少させる条件を探索している。

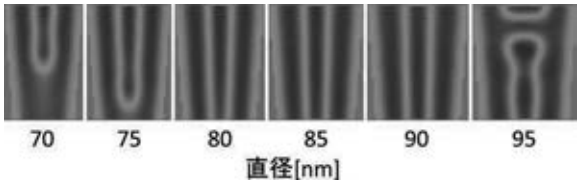


図４ガイド直径を変化させた時のホール断面図
（高さは 100nm、テーパー角は 4° に固定）

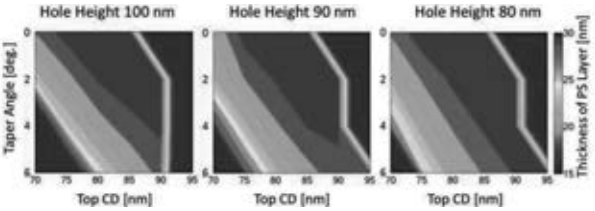


図５残膜厚さの等高線図。ガイドホールの高さ：（左）100nm,（中央）90nm、(右)80nm。

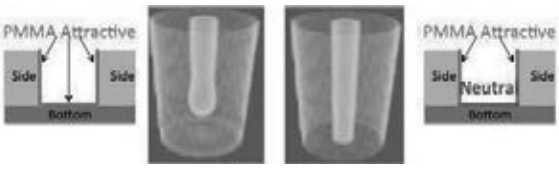


図６ガイドホール底面の親和性によるモルフォロジー変化。(左)PMMA 親性大（右）中性。

4 結言

本研究では、ジブロックコポリマーの自己組織化を利用したホールシュリンクプロセスで問題となっている PS 残膜の最小化をシミュレーションで試みた。まず、本研究で用いた OK モデルの計算精度を SCFT との計算結果と比較し、高速かつ高精度でガイドホール内でのジブロックコポリマーのモルフォロジーを予測できることを検証した。次に、ホールシュリンクの実験結果との合わせ込みによりモデルのパラメータを決定し、ガイドホールの形状やガイド壁の親和性を変えることによって、PS 残膜の最小化をシミュレーションによって行った。その結果、ガイドホール形状の最適化だ

けではPS 残膜厚さはほとんど変わらず、ガイド底面の親和性を中性化することで PS 残膜が抑制される可能性が見つかった。

［参考文献］

[1] R. Gronheid et al., "Readying Directed Self-Assembly for Patterning in Semi-Conductor Manufacturing," *J. Photopolym. Sci. Technol.* **26**, 779 –791 (2013).

[2] Y. Seino et al., "Contact hole shrink process using graphoepitaxial directed self-assembly lithogra- phy," *J. Micro/Nanolith. MEMS MOEMS.* **12**, 033011 (2013).

[3] K. Yamada et al., “Fddd structure in AB-type diblock copolymers,” *J. Phys.*: *Condens. Matter*, **18**, L421 (2006).

[4] K. Yoshimoto et al., “Optimization of directed self-assemble hole shrink process with simplified model,” *J. Micro/Nanolith. MEMS MOEMS.* **13(3)**, 031305 (2014)

[5] N. Laachi et al., "The hole shrink problem: Theoretical studies of directed self-assembly in cylindrical confinement," *Proc. SPIE* **8680**, 868014 (2013).

最後に、本共同研究制度（若手奨励枠）を活用させて頂きましたことに感謝致します。

色素増感太陽電池を指向した新奇ポルフィリン系色素の構造と電子構造の解明

梅山有和

京都大学大学院工学研究科分子工学専攻

1 緒言

現在、化石燃料の枯渇と環境へ与える影響が深刻な問題になってきており、太陽電池活用の拡充が期待されている。しかしシリコンに代表される無機太陽電池は、製造コストが高く、広範に普及させることに障害が生じている。一方、有機太陽電池は低コスト、着色による意匠性、柔軟性、軽量という利点が指摘されている。中でも色素増感太陽電池はエネルギー変換効率（η）が高く、次世代のエネルギー源として期待されている。現在までにルテニウム錯体と多孔性酸化チタンを用いた系において11%を越える高いη値が報告されている。しかし、ルテニウムは高価で、資源制約もある。そのため、ルテニウム錯体に変わる安価で高性能な色素の開発が求められている。そこで我々は、安価な金属を利用できるポルフィリン錯体に注目し、新規ポルフィリン系色素の開発を行ってきた [1]。取り分け、メソ位に導入されたジアリールアミノ基の数と導入位置の効果を検討し、その相関を解明している [2,3]。本研究では、高い太陽電池性能を示す既報の YD-2（Figure 1）より優れた光捕集能を有する色素分子として、対称性が低くなるようにジアリールアミノ基を導入した ZnPBAT（Figure 1）を設計した [3]。また、参照化合物として、エチニレン基を持たない ZnPBA（Figure 1）も合わせて設計し、その合成と物性評価を行った。

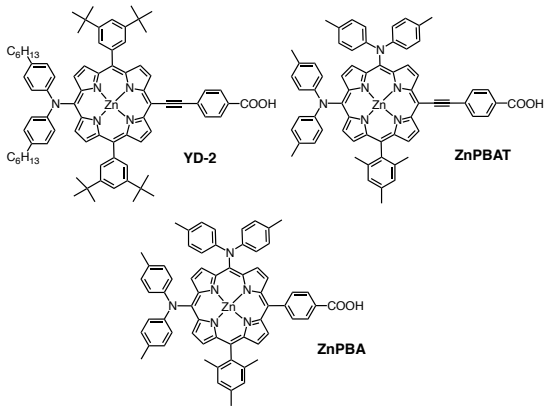


Figure 1. Structures of porphyrin compounds.

2 結果と考察

2.1 新規ポルフィリン色素の諸物性と色素増感太陽電池への応用

ZnPBAT、YD2、およびZnPBAのエタノール溶液中の吸収スペクトルをFigure 2に示す。ZnPBATおよびZnPBAはYD-2に比べて幅広い範囲で吸収を示し、2 つのジアリールアミノ基の導入により光捕集能が向上することがわかった。また、電気化学測定により求めたZnPBATおよびZnPBAの第一酸化電位（E_{ox} = 0.85 V vs. NHE）はYD2（E_{ox} = 0.93 vs. NHE）と比較して高くなった。

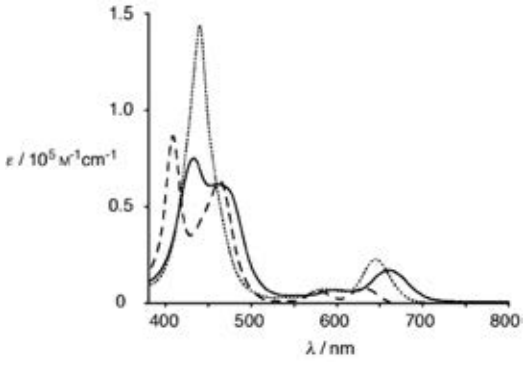


Figure 2. UV/Vis absorption spectra of ZnPBAT (solid line), YD2(dotted line), and ZnPBA (dashed line) in EtOH.

ZnPBAT を酸化チタンに吸着させ、色素増感太陽電池を作製して性能評価を行ったところ、η= 10.1%（J_{SC} = 19.33 mA cm^{-2}、V_{OC} = 0.719 V、ff = 0.724）であった。これは、YD-2（η= 9.1%、J_{SC} = 17.05 mA cm^{-2}、V_{OC} = 0.742 V、ff = 0.718）や ZnPBA（η= 8.3%、J_{SC} = 16.26 mA cm^{-2}、V_{OC} = 0.713 V、ff = 0.719）と比べて高い値となっている。さらに、最適条件下での作用スペクトルを測定したところ、吸収の長波長化を反映し、700-800 nm で ZnPBAT の IPCE 値は他と比較して高くなった。

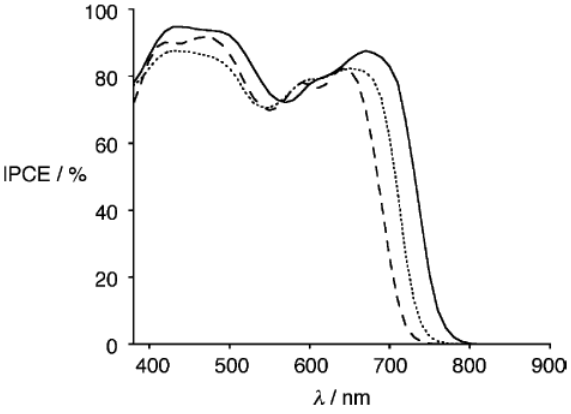


Figure 3. Action spectra of ZnPBAT (solid line), YD2(dotted line), and ZnPBA (dashed line) cells.

2.2 理論計算

ポルフィリンの再安定化構造およびそのフロンティア軌道の電子構造について知見を得るために、密度汎関数法（DFT）による理論計算（B3LYP）を行った。基底関数系に 3-21G(d)を選択し、Gaussian03 プログラムを用いた。分子軌道の可視化には Molstudio3.0 プログラムを用いた。Figure 3 に ZnPBAT、YD2、および ZnPBA の最高被占軌道（HOMO）と最低空軌道（LUMO）における電子密度分布を示す。一般に、アンカー部位の LUMO の電子密度が大きいほど、励起状態の色素と酸化チタンの 3d 軌道との間の電子カップリングが大きくなることが知られている。ZnPBAT は YD-2 と同様にアンカー部位の LUMO の電子密度が大きくなることがわかり、ZnPBAT の電子注入効率は YD-2 と同程度に高いと考えられる。そのため、ZnPBAT の高い光捕集能を反映し、YD-2（17.05 mA cm^{-2}）と比べて ZnPBAT（19.33 mA cm^{-2}）の短絡電流（J_{SC}）値が向上したと考えられる。このように、本計算結果は実験結果の理論的解釈の一助となった点で意義がある。

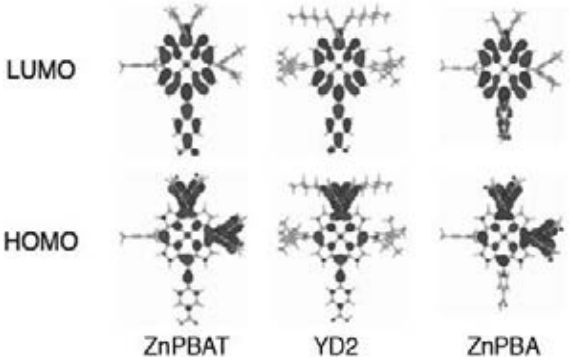


Figure 3. Some sets of molecular orbital diagrams for ZnPBAT, YD-2, and ZnPBA obtained by DFT calculations with B3LYP/6-31G(d).

3 参考文献

[1] H. Imahori, T. Umeyama, K. Kurotobi and Y. Takano, “Self-Assembling Porphyrins and Phthalocyanines for Photoinduced Charge Separation and Charge Transport”, *Chem. Commun.*, **48**, 4032-4045 (2012).

[2] M. Simon, H. Iijima, Y. Toude, T. Umeyama, Y. Matano, S. Ito, N. V. Tkachenko, and H. Lemmetyinen and H. Imahori, “Optical, Electrochemical, and Photovoltaic Effects of an Electron-Withdrawing Tetrafluorophenylene Bridge in a Push-Pull Porphyrin Sensitizer used for Dye-Sensitized Solar Cells”, *J. Phys. Chem. C*, **115**, 14415-14424 (2011).

[3] H. Hayashi, A. S. Touchy, Y. Kinjo, K. Kurotobi, Y. Toude, S. Ito, H. Saarenpää, N. V. Tkachenko, H. Lemmetyinen, and H. Imahori, “Triarylamine-Substituted Imidazole- and Quinoxaline-Fused Push-Pull Porphyrin for High Performance Dye-Sensitized Solar Cell”, *ChemSusChem*, **6**, 508-517 (2013).

[4] K. Kurotobi, Y. Toude, K. Kawamoto, Y. Fujimori, S. Ito, P. Chabera, V. Sundström, and H. Imahori, “Highly Asymmetrical Porphyrins with Enhanced Push-Pull Character for Dye-Sensitized Solar Cells”, *Chem. Eur. J.*, **19**, 17075-17081 (2013).

流水中の自然石に働く流体力の評価

吉田 圭介[1], 牛島 省[2], 田中 龍二[1], 宮木 伸[3]

[1]岡山大学大学院 環境生命科学研究科, [2]京都大学 学術情報メディアセンター, [3]大成建設株式会社

1 諸言

近年，河川のインフラ整備では構造物の堅牢さに加えて自然環境への調和が要求されており，現地で調達した石や木などの自然素材を用いた構造物が河道内に設置される場合がある．例えば，適度な空隙を有する水制工，落差工（**図 1** 参照），根固工および水深寸法以上の巨石を河道に配置した工法は，川の流れに多様性をもたせて河川景観を創造したり，水生生物の生息環境を保全する効果がある．

一方，従前のコンクリート製構造物とは異なり，自然石を用いた工法では材料の要素は必ずしも画一的ではない．そのため，構造物の流水に対する力学的応答が明らかでない場合があり，その結果，自然石からなる構造物の治水機能を検討することは一般に困難である．現在，河川実務では技術者の経験や模型実験[1]などに基づいてこうした構造物の安全度を照査したり推定するのが通例であるが，これには労力や費用が多くかかる上，射流などの急変流場では計測が困難であるため，設計の信頼性は必ずしも十分ではないと指摘されている．

そこで，本研究では固気液多相場の数値解析法[2]を利用して，一様な開水路の底面に設置された自然石に働く流体力を解析し，併せて行った室内模型実験との比較から数値解析法の妥当性を検討することとした．本研究の最終目標は，流水中の自然石に働く流体力を精度良く評価できる解析手法を確立して，水工学の実務に資することである．

2 実験

本研究では長さ 16m，幅 0.6m，高さ 0.4m，勾配 1/1000 の可変勾配型循環水路（**図 2** 参照）を用いた．座標系は図に示す通りである．実験では強化プラスチック製の自然石の模型（**図 3** 参照）をアクリル製の水路底面に接着して通水し，流水中の流体力と周辺の流れ場を測定した．

図 1 落差工（円山川）

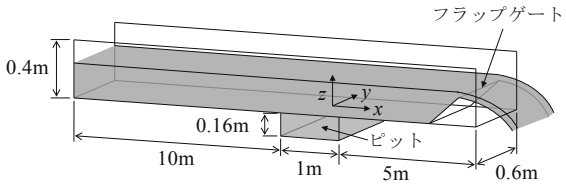


図 2 実験水路と座標系

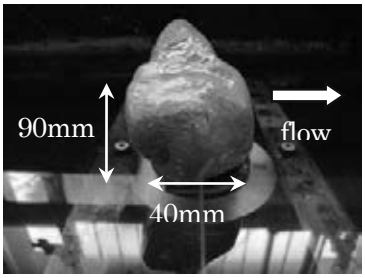


図 3 実験で用いた模型の設置状況（側面）

表 1 実験条件

Case	流量 Q [l/s]	断面平均の主流速 $\overline{U}$ [m/s]	フルード数 Fr
1	40	0.33	0.24
2	60	0.50	0.36
3	80	0.67	0.48

流体力の計測では水路のピット内中央部に四分力計(東京計測社製，Y116M2AG4)を設置し，自然石に働く抗力・揚力を 30Hz の時間解像度で測定した．水流の流速計測には 2 次元電磁流速計(KENEK 社製，VM-802H 型)を用い，開水路の底面境界層が

表 2　数値解析条件表

時間刻み Δt [s]	1.0×10^{-3}
計算時間 T [s]	30.0
空間格子幅 $(\Delta x, \Delta y, \Delta z)$ [mm]	(5.0, 5.0, 5.0)
計算領域 (L_x, L_y, L_z) [m]	(3.0, 0.6, 0.2)
流入条件	一様流速
流出条件	自由流出
水面条件	Slip 条件
側壁・底面条件	Non-slip 条件

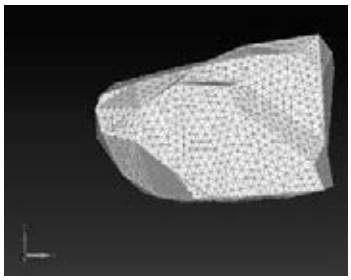


図 4　自然石模型の 4 面体分割（yz 平面）

十分に発達して，主流速の鉛直分布が対数則分布を示すことを確認した. 実験条件は**表 1** の通りである.

3　数値解析

本研究では牛島ら[2]の数値解析手法を用いて，自然石に働く流体力を解析した．牛島らの手法では固気液多相場を一流体モデルとして扱い，混合体に対する連続式と運動方程式を非圧縮性条件の下で DNS（直接数値計算）により解く．また，任意形状の固体は四面体要素の集合として表現され，固体領域に働く流体力は抗力係数などの経験的なパラメータからではなく，直接的に適切に算定される．

本研究では実験で用いた模型の 3 次元形状測定を行って，数値解析に必要なデータを作成した（**図 4** 参照）．数値解析の条件は**表 2** に示す通りである．解析では自然石を剛体とみなした．また，解析では直方体格子を用い，x, y, z 各方向に均一に格子分割を行っている．なお，負荷軽減のため，解析では領域分割による並列計算を行っている．

4　結果と考察

図 5 は数値解析結果の一例であり，模型石を通る鉛直断面（xz 平面）における渦度コンターを示した．模型石周辺では準周期的な渦構造が生成され，それらが移流された後に，上面境界付近で拡散する様子が観察される．

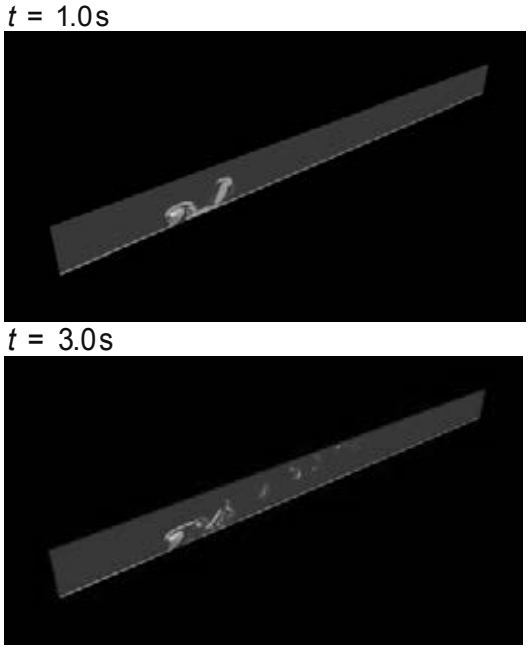


図 5　模型石を通る鉛直断面の渦度のコンター（計算結果，Case3，t ：計算初期からの経過時間）

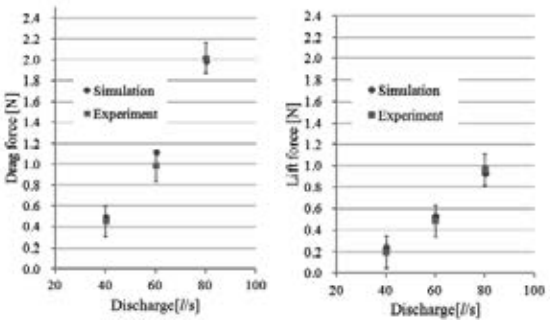


図 6　流量と抗力・揚力との関係

図 6 には水理実験と数値解析から求めた，流量毎の抗力と揚力の算定値を示す．図中，実験値にはエラーバーを付した．定性的には流量と抗力には比例関係が認められる．また，定量的には実験値と解析値は概ね一致する．今後は条件の詳細を検討するなどして，流体力評価の再現性を向上させたい．

5　参考文献

[1] (財)土木研究センター：護岸ブロックの水理特性試験法マニュアル（第 2 版），2003.

[2] 牛島省，福谷彰，牧野統師：3 次元自由水面流中の接触を伴う任意形状物体運動に対する数値解法,土木学会論文集 B，Vol.64，pp.128-138，2008.

エノン写像の周期軌道展開

斉木 吉隆 *

一橋大学大学院商学研究科

yoshi.saiki@r.hit-u.ac.jp

1 エノン写像の不安定周期軌道

ここでは実 2 次元のエノン写像

$$\begin{cases} x_{n+1} = a - x_n^2 + by_n \\ y_{n+1} = x_n \end{cases}$$

を考察する．a, b はパラメタで, $a = 1.4, b = 0.3$ の場合には非双曲的となる．この系から不安定周期軌道を検出し , それぞれに対してリアプノフ指数, 安定多様体と不安定多様体のなす最小角度 (以下最小角度とよぶ) を， Ginelli ら (2007) [1] の手法を用いて計算した．図 1 は周期を $p = 12$ から $p = 26$ まで増加させたとき, 各不安定周期軌道に関して周期とリアプノフ指数をプロットしたものである．図 2 は周期を $p = 12$ から $p = 26$ まで増加させたとき , 各不安定周期軌道に関して周期と最小角度をプロットしたものである．幅広い最小角度を持ち, 特に 0 付近のものも存在する．図 3 は, 周期 $p = 12$ から $p = 18$ の各不安定周期軌道に対して最小角度とリアプノフ指数を同時にプロットしたものである．$a = 1.4$ の場合には, 最小角度が小さい不安定周期軌道はリアプノフ指数も小さいことが確認できる．

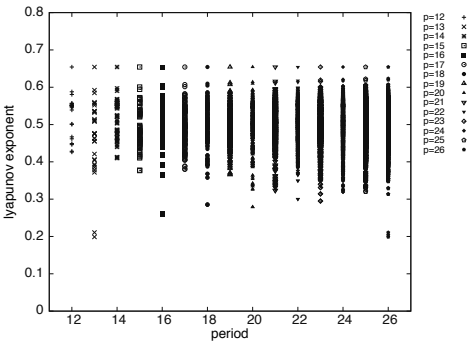


図 1: 周期軌道の周期とリアプノフ指数.

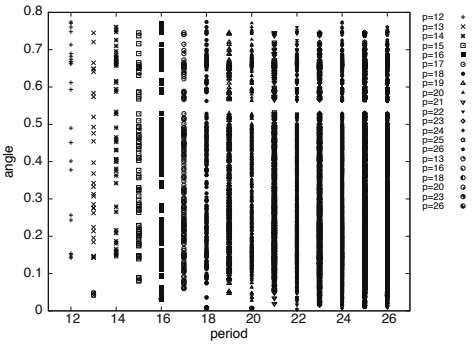


図 2: 周期軌道の周期と 周期軌道上の安定多様体と不安定多様体がなす最小角度.

2 周期軌道展開: 不安定周期軌道によるカオス統計量の近似

Grebogi, Ott, Yorke (1988) [2] を参考に, カオスアトラクタを使って計測した natural measure を不安定周期軌道を使って近似する方法を述べる．まず， 領域 C_k の natural measure の近似値は $\mu_k = \lim_{T\to\infty} \frac{f(x_0,T,\epsilon_k)}{T}$ と定義される．但し $f(x_0, T, \epsilon_k)$ は吸引領域内のランダムな初期値 x_0 から始まる軌道が時間 T の間に一辺の長さ ϵ_k の領域 C_k に滞在する時間とする．これを不安定周期軌道を使って近似する．領域 C_k の natural measure $\mu_k(p)$ を $\mu_k(p) = \sum_{x_{kp}\in C_k} \frac{1}{L_1(x_{kp})}$ で近似する．$L_1(x_{kp})$ はヤコビ行列 $DM^p(x_{kp})$ の伸びる方向の固有値である．但し $M(x)$ は d 次元写像 (エノン写像の場合は 2 次元) とする．x_{kp} は p 回写

*本研究は二瓶 浩輝 氏 (北海道 NS ソリューションズ, 元北海道大学) との共同研究に基づく .

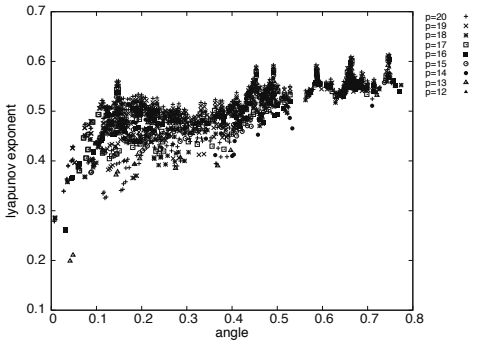


図 3: 周期軌道上の安定多様体と不安定多様体がなす最小角度とリアプノフ指数.

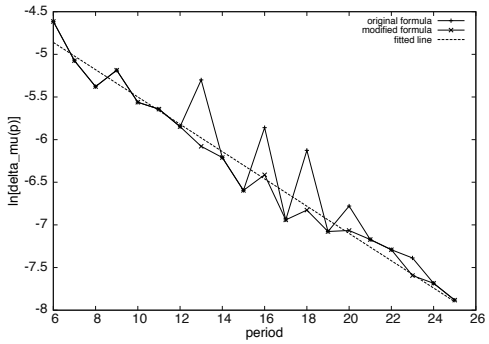


図 4: 周期 p (p は 6 から 25) の不安定周期軌道を使った natural measure の近似の誤差 (+ :[2] に基づく近似, ×:リアプノフ指数が低い (多様体がなす最小角度が小さい) 周期軌道を無視した近似).

像の不動点, 言い換えると p または p の約数の周期点である. この議論を領域 C_k 内の全ての不動点において行うと, 領域 C_k の natural measure の近似値は $\mu(C_k) = \lim_{n\to\infty}[\sum_{\text{fixed point in } C_k} \lambda_{1j}^{-1}]$ と表される. 不安定周期軌道によって構成された natural measure の近似値が, 真の natural measure に近付けばよい. 近似値が真の値にどれだけ近付いたかは, 誤差 $\Delta\mu(p) = \sqrt{\sum_{k=1}^{N} \frac{[\mu_k(p)-\mu_k]^2}{N}}$ により判断する. 但し N は natural measure の近似値がゼロではない領域 C_k の数である. natural measure の定義より, 吸引領域内に入っている限り全ての領域の natural measure の和は 1 になる. しかし, 不安定周期軌道で近似する場合はリアプノフ指数のみを使った式であるため natural measure の近似値の和が 1 になるとは限らない. そこで $\Delta\mu(p) = \sqrt{\sum_{k=1}^{N} \frac{[((\mu_k(p)/\mu_s(p))-\mu_k]^2}{N}}$ として正規化を行う. 但し $\mu_s(p) = \sum_{C_k} \mu_k(p)$, つまり不安定周期軌道で生成した natural measure の近似値の総和である. まずはこの式を使って Lai ら [3] の結果を再現した (図 4 の+). 以上により, 双曲的な状況を想定して構成された周期軌道展開が, ヘノン写像の非双曲パラメタ $a = 1.4$ の場合にも有効であることがわかる. 一方, 近似精度が悪い周期も存在していることも確認できる.

3 周期軌道展開の改良

上でおこなった周期軌道展開において近似精度が悪い周期 (例えば周期 $p = 13, 16, 18$) の周期軌道の中にはリアプノフ指数が非常に小さく, 安定多様体と不安定多様体がなす最小角度も非常に小さいものを含んでいる. このようなリアプノフ指数が小さい周期軌道の点を含む領域 C_k において, 近似値がカオス軌道統計量よりも大きくなり過ぎているため, リアプノフ指数が 0.3 以下の不安定周期軌道を無視して同様の計算を行った (図 4). ここで, + が修正前, × が修正後の誤差であり, 従来の方法によって精度が落ちている周期での近似精度の改良に成功していることがわかる. 周期軌道展開は元来, 不安定周期軌道の安定多様体と不安定多様体が十分大きな角度をもって横断的に交わる状況を想定されて構築された公式である. しかし, 安定多様体と不安定多様体のなす最小角度が小さい不安定周期軌道に関しては, 周期軌道展開においてその周期軌道が担当する領域が小さくなる. リアプノフ指数が非常に小さい周期軌道と安定多様体と不安定多様体のなす最小角度の小さい周期軌道は対応している (図 3) ため, そのような周期軌道の貢献度を落とす修正を行うことにより周期軌道展開の改良が可能となったと解釈される.

参考文献

[1] F. Ginelli, P. Poggi, A. Turchi, H. Chate, R. Livi, and A. Poloti, Phys. Rev. Lett. 99, 130601:1-4, 2007.

[2] C. Grebogi, E. Ott, and J. A. Yorke, Phys. Rev. A 37, 1711-1724, 1988.

[3] Y. Lai, Y. Nagai, and C. Grebogi, Phys. Rev. Lett. 70, 649-652, 1997.

行列分解タイルアルゴリズムのスーパーコンピュータシステムでの実装

小嶋弘樹

山梨大学

1 はじめに

コンピュータの性能が大きく向上したことでコンピュータ・シミュレーションは実験に代わる検証法手法として自然科学分野で成長しつつある.これらで扱われる巨大な行列を何らかの扱いやすい形式に変換することを前処理という.この前処理にかかる時間が全体に占める割合は非常大きいので,これを高速に行うことがシミュレーション全体の高速化に繋がる.我々は,前処理に用いられる行列分解の 1 つである QR 分解の高速化を行っている.

QR 分解のタイルアルゴリズムに対して,これまで RightLooking,StaticPipeline,動的スケジューリングなどのスケジューリング手法により、行列分解の行方向の並列化による高速化は行われてきた。クラスタシステムでは,より多くの計算資源が利用できるので,更なる並列化によって行列分解の高速化が見込まれる.

本研究では、列方向の並列化である CAQR[1]を Cray XE6 上に実装し,行列分解計算の高速化を目的とする.

2 QR 分解

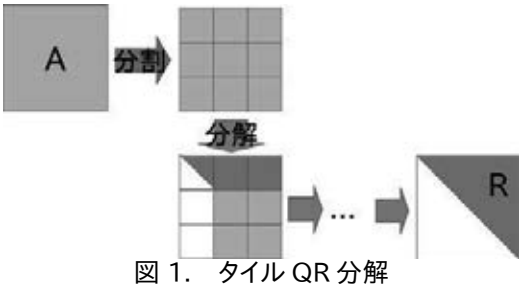


図 1. タイル QR 分解

QR 分解とは, $m \times n (m \geq n)$ 行列 A を式(1)となるような $m \times n$ 直交行列 Q と R の積に変換する行列分解の 1 つである.密行列の三重対角化など前処理で多用される重要な行列分解の 1 つである.

$$A = QR \qquad (1)$$

本研究では,タイルアルゴリズムで QR 分解を行う.タイルアルゴリズムとは,行列を行方向と列方向に分割し,文化いつした小行列(タイル)ごとに計算を行うことで分解を進めていく. (図 1)

タイル QR 分解には 4 つの計算カーネルがある.GEQRT カーネルと TSQRT カーネルは分解カーネルと呼ばれ,タイルを分解し,変換行列を生成する.LARFB カーネルと SSRFB カーネルは更新カーネルと呼ばれ,分解カーネルで生成した変換行列を適用して後続タイルの更新を行う.

計算カーネルは,依存関係を満たしたうえでなら任意の順序で実行可能である.依存関係は i,j,k の 3 方向存在し(図 2),全ての依存関係を満たしたタイルは計算が可能になる.

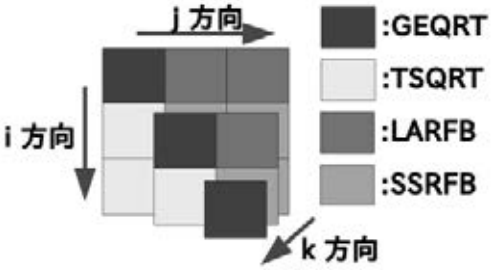


図 2. 依存関係

3 行方向の並列化

依存関係を満たした上でなら任意の順序で計算を行えるため,タイル QR 分解には様々なスケジューリングが考案されている.同一タイル行の更新カーネルは並列に実行可能である.これを利用して,行方向の並列化を行ったスケジューリングに RightLooking や StaticPipeline[2]がある.

RightLooking(以下,RL と呼ぶ)は,上のタイルから順番に分解カーネルを適用した後,同一行の更新カーネルを並列に適用し,分解を進めるスケジューリングである.

StaticPipeline(以下,SP と呼ぶ)は,タイル列ごとに担当するスレッドを割り当てるスケジューリングである.ステップを跨いでサイクリックにスレッドを割り当てることでスレッド間の負荷分散を行うという特徴がある.

4 列方向の並列化

タイル化,行方向並列化により QR 分解の並列度が高まったが,依存関係のため,列方向には逐次実

行となってしまう.列方向並列化の手法にCAQR[1]がある.CAQRとは行列を複数のドメインに分割し,各ドメインでタイルQR分解を行い,結果をマージするQR分解である.

マージとは,複数ドメインの分解の結果を1つのドメインにまとめる操作である.この操作にあたる計算カーネルがTTQRTである.また,マージに対する後続タイルの更新をTTMQRで行う.

本研究で実装したCAQRは,ドメイン間の並列化をMPI,ドメイン内の並列化をOpenMPで行うハイブリット実装である.ドメイン内の分解は,分解カーネル,同一行タイルの更新カーネルの順でカーネルを実行し,各ドメインの下端まで順番に分解と更新を繰り返す.(図3)

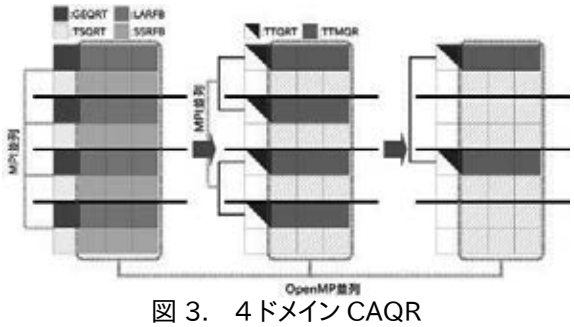


図3.　4ドメインCAQR

5　実験

5.1 実験環境

実験には京都大学スーパーコンピュータのシステムA(Cray XE6)の1ノードを使用した.システムの概要を表1に示す.

表1.　実験環境

ノード	プロセッサ数	2
	メモリ	64GB
CPU	プロセッサ名	AMD Opteron
	クロック	2.5GHz
	コア数	16

5.2 実験内容

RL,SP,CAQRのQR分解速度を比較した.行列は行:列=10:1の長方形行列である.RL,SPは32スレッド,CAQRは4ドメインで実行し,1ドメインあたり8スレッドを割り当てた.タイルサイズは200とした.

5.3 実験結果

CAQRはRLの約2倍の速度であるが,SPの約半分の速度という結果であった。(図4)

SPと比較して,分解速度が大きく劣っていた理由として,実装したCAQRのドメイン内分解スケジューリングに原因があると考えられる.ドメイン内はRLに似たスケジューリングを行っており,十分な並列化が行われていないという問題がある.しかし,RLと分解速度を比較すると,CAQRはRLの2倍の速度が出ていることから,列方向並列化による効果が現れていると考えられる.

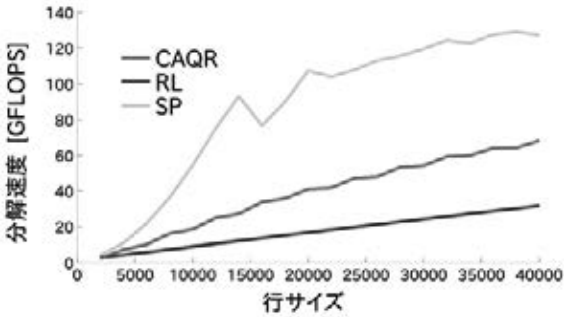


図4.　速度比較

6　今後の課題

今回実装したCAQRは,ドメイン内の処理をRLに似た形で行っていたが,この部分の処理をSPや先行研究の動的スケジューリング[3]を適用することで更なる高速化が見込める.

参考文献

[1]J.Kurzak,H.Ltaief,J.Dongarra,Rosa M.Badia,“ScheDuling Linear Algebra Operations on Multicore Processors”,LAPACK Working Note 213,(2009).

[2]B.Hadri,H.Ltaief,E.Agullo,J.Dongarra,“Enhancing Parallelism of Tile QR Factorization for Multicore Architectures”,LAPACK Working Note 222,(2009).

[3]Tomohiro Suzuki,Hideki Miyashita,“OpenMP/MPI implementation of tile QR factorization on T2K open supercomputer”. Proceedings of IEEE 7th International Symposium on Embedded Multicore/Manycore SoCs (MCSoC-13),Special Session on Auto-Tuning for Multicore and GPU(ATMG),(2013).

固体表面特性の違いによる乱流構造の変化と抵抗低減効果の解明

中本　真義

大阪大学大学院基礎工学研究科物質創成専攻

1　緒言

固体表面上での流体との摩擦抵抗を低減することは、多くの工業分野において非常に重要な課題となっている。壁に接する流れ（特に乱流域の流れ）において、壁面特性は摩擦抵抗に多大な影響を及ぼす。そのため、摩擦抵抗の低減させる手段として物体表面の特性や形状を変化させることにより、物体周りの流れを制御し、流体抵抗を低減させる研究が数多く報告されている。その一例として、船舶においては、粘弾性物質であるヒドロゲルポリマーを船底に塗布することで、海水との粘性摩擦抵抗が低減されることが知られている。ヒドロゲルポリマーは様々な物理的性質を持っており、その周囲の流れ場が複雑化しているため、抵抗低減のメカニズムを実験のみで解明することは非常に難しい。本研究では、数値流体解析を用いてヒドロゲルポリマーの乱流抵抗低減効果のメカニズムの解明することを目的とする。本報告では、ヒドロゲルの多孔質体としての性質に着目し、壁の多孔性が周囲の流体に及ぼす影響と乱流抵抗との関連性について記述する。

2　数値解析手法

支配方程式は、非圧縮性非定常の 3 次元ナビエ・ストークス方程式と連続式である。数値計算に用いた計算領域の概略図を Fig. 1 に示す。本研究で用いる計算領域は、上下に透過性壁を持つ乱流チャネルである。チャネル内部は、流体のみが流れる流体領域と、透過率 φ をもつ多孔質領域の二つに分かれており、厚みはそれぞれ、$2\delta_f$ と δ_p である。本研究では、流体領域と多孔質体領域は連続であるので、多孔質体の表面の粗さは考慮しない。境界条件は、x、z 方向に周期境界条件を用い、壁では no-slip 条件とした。計算領域の大きさは、x、y、z、方向にそれぞれ、$4\pi\,\delta_f$、$(2\,\delta_f + 2\,\delta_p)$、$2\pi\,\delta_f$ である。多孔質の厚み δ_p は流体領域の 5%、10%、および 20%とした。初期値として、ハーゲン・ポアズイユ型の速度分布に微小の攪乱を加えた速度分布を与えた。計算格子は、x、z 方向には等間隔格子を用い、y 方向については、多孔質領域には等間隔格子を、流体領域は、二つの領域の境界付近に格子点を集中させるために、双曲線正接関数を用いて不等間隔格子とした。

3　多孔質体モデル

本研究で用いた多孔質体モデルとしてダルシー・フォルヒハイマー則に従う圧力勾配を、ナビエ・ストークス式の外力項として加えた。

$$\boldsymbol{F} = \nabla \frac{p}{\rho} = -\frac{\nu}{\sqrt{K}}\boldsymbol{u} - \frac{c_F}{K}|\boldsymbol{u}|\boldsymbol{u}$$

ここで K は透過係数、c_F はフォルヒハイマー係数、$\boldsymbol{u}$ は流量平均速度である。本研究に用いる各物性値は、既往の研究[1]において使用されたセラミックの多孔質体のものを使用し、それぞれ $K = 0.020\ \mathrm{mm}^2$、$c_F = 0.17$ である。本研究では FIK 恒等式[4]から算出される抵抗係数 C_f の多孔質厚み δ_p に対する依存性を検証し、各パラメータが乱流構造、乱流抵抗へ及ぼす影響について検討した。

4　結果と考察

多孔質壁による乱流抵抗の増減効果への影響を評価するために、抵抗低減率（Drag reduction Ratio : DR）を、非透過性壁条件に対する多孔質壁条件での抵抗係数の比から算出した。

$$DR[\%] = \left(1 - \frac{{C_f}^p}{{C_f}^0}\right) \times 100$$

ここで、上付きの 0、p はそれぞれ非透過壁条件と多孔質壁条件での抵抗係数を表す。また、抵抗係数 C_f は完全に発達した平行平板チャネル乱流における FIK 恒等式[2]より算出した。

ここで、全ての変数は、チャネルの幅、$2(\delta_f+\delta_p)$、と流量平均速度、 $\boldsymbol{u_m}$ 、で無次元化されている。多孔質体の厚みを流体領域の 5%、10%、20%（D005、D010、D020）と変化させたときの DR を Table 2 に示す。本研究では、壁厚さが最も小さいケース（D005）で顕著な抵抗低減効果が得られた。多孔質体の厚みを流体領域の 5%、10%、20%（D005、D010、D020）と変化させたときの DR を Table 2 に示す。本研究では、壁厚さが最も小さいケース（D005）で顕著な抵抗低減効果が得られた。Figure 2 は、多孔質壁近傍におけるストリーク構造を表している。ストリーク構造は平均速度からのずれ（$u' \equiv u - u_{average}$）を可視化したものであり、 $u'>0$（図中の赤）の領域は高速ストリーク、$u'<0$（図中の青）の領域は低速ストリーク構造を表している。多孔質壁の厚さが増大するにつれて、主流方向に長いストリーク構造が見られなくなっていることが分かる。この結果は、壁の多孔性によって壁近傍の乱流構造が弱くなるという従来の結果と同じ傾向[3]を示しており、このような乱流構造の崩壊がエネルギー散逸を促進させ、抵抗が増大したと考えられる。

5 結言

本研究では、多孔質の壁の厚さが、乱流抵抗、乱流構造に及ぼす影響について、多孔質体をダルシー則に従う連続体と近似する数値計算により検討した。

- 多孔質壁の厚さが最も小さい系において、抵抗低減効果が得られた。この時の壁の厚さは、境界層と同程度の大きさであった。
- 乱流変動は、多孔質の壁厚みが増大するとともに大きくなり、壁近傍の乱流構造は、弱まった。

本研究では、多孔質体を特徴づける長さスケールのうち、とくに多孔質壁厚さの影響について検討したが、表面粗度に境界層の変化や、その他の長さスケールとの関連についても検討すべきである。

引用文献

[1] Suga, K. *et al.*, International Journal of Heat and Fluid Flow, 31 (2010), pp. 974-984.

[2] Fukagata, K., Iwamoto, K. and Kasagi, N., Phys. Fluids, 14 (2002), pp.73-76.

[3] Hahn, S. J. and Choi, H., J. Fluids Mech., 450 (2002), pp. 259-285.

謝辞

本共同研究制度(若手研究者奨励枠)を活用させて頂いたことを、この場を借りて厚く御礼申し上げる。

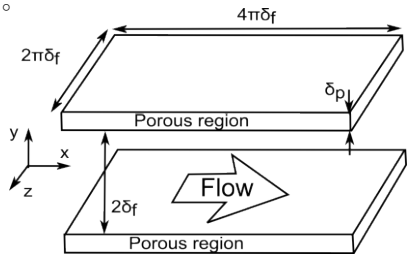


Fig. 1 A schematic of the channel flow bounded with two parallel porous walls.

Table 1 Drag reduction ratio of each case.

Case	C_f (×10^3)	DR	$\delta_p{}^+$
D005	6.55	9.6	9
D010	7.16	1.3	18
D020	7.56	-4.3	37
Flat	7.25	-	-

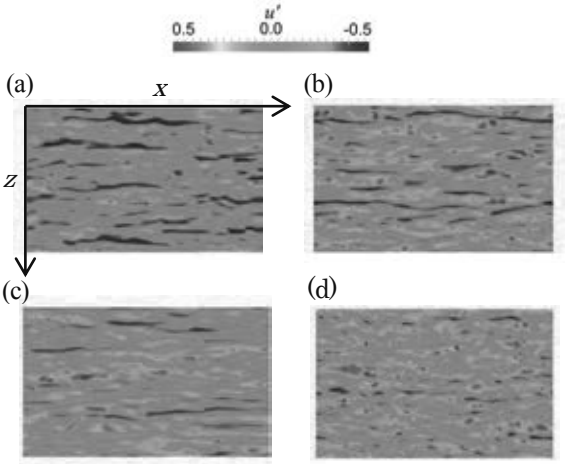


Fig. 2 The streaky structures at y^+=20 : (a) impermeable wall, (b) Case D005, (c) Case D010, (d) Case D020.

構造がゆらぐ希土類錯体を用いる反応の生成物選択性・立体選択性はいかにして制御されているのか

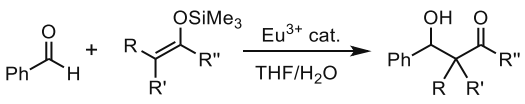
畑中美穂

京都大学福井謙一記念研究センター

構造が揺らぎやすい希土類錯体を触媒として用いる反応の機構を、人工力誘起反応(AFIR)法を用いて明らかにした。柔らかい触媒系における立体選択性の定量的な議論には、遷移状態を決め打ちで求めるだけでは不十分で、AFIR 法を用いる遷移状態の網羅的探索を行うことが必要であると分かった。また、不斉 Eu^{3+}-DOTA 誘導体を触媒とする向山アルドール反応の場合、よりイオン半径の大きい Pr^{3+}などに置換することで、触媒の構造は揺らぎにくくなるため、立体選択性が向上することが分かった。

1 はじめに

有機化学の分野で、これまで多くの立体選択的反応を可能にする不斉触媒が開発されてきており、その反応機構や遷移状態(TS)の解析に、計算化学は大きく貢献してきた。これまで、高い立体選択性を実現するためには、触媒自体の構造も TS の構造も硬くなければならないという考えが常識であり、計算化学による立体選択性の研究でも、立体異性体毎に一つの TS だけを決め打ちで求めて、それらを比較するという方法で議論されてきた。しかし、近年盛んに研究が行われている希土類イオンを含む不斉触媒の場合、開殻 4f 電子が閉殻 5s・5p 電子に外側から遮蔽されるため、配位子と共有結合を結べず、希土類イオン周りの配位構造が柔らかく揺らいでしまうという問題があった。このような揺らぎの問題があるにも関わらず、一体どのようにして、高い立体選択性が発現するのだろうか？ 本研究では、水中で希土類(Eu^{3+})触媒を用いる二種の向山アルドール反応(Scheme 1)に着目し、自動反応経路探索の一つ、人工力誘起反応(AFIR)法を用いた反応経路の解析を行った。



Scheme 1. 水中での向山アルドール反応

2 人工力誘起反応(AFIR)法とは

本研究で用いる AFIR 法について、簡単に説明する。[1] Figure 1 に示すのは、原子 A と B の距離(r_{AB})に対するポテンシャルエネルギー$E(r_{AB})$である。通常、A と B の解離極限と分子 AB の形成の間には、反応障壁がある。通常の衝突シミュレーションでは、この反応障壁を超えて AB を形成する過程が起こることは非常に稀である。

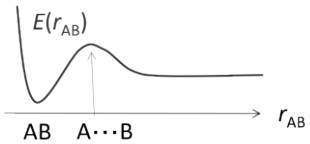


Figure 1. 原子 AB の分子衝突におけるポテンシャルエネルギー$E(r_{AB})$

ここで、A-B 間距離に比例する人工力項 (αr_{AB})をポテンシャルエネルギー$E(r_{AB})$に加えると、Figure 2 のように、(αが十分に大きければ) 障壁のない関数 $F(r_{AB})$に変換することができる。この関数 $F(r_{AB})$上であれば、関数値の最小化から AB に簡単にたどり着くことができる。最小化の過程で得た $F(r_{AB})$関数値から人工力項αr_{AB}を除くことで$E(r_{AB})$を得、この関数上の極大点を、近似 TS 構造ととらえ、近似構造を初期構造とする TS の構造最適化を行うことで、簡便に TS を得ることができる。

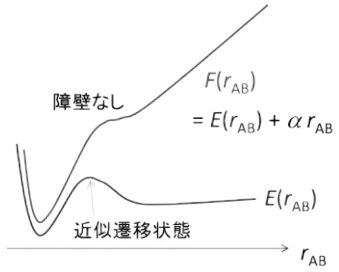


Figure 2. 人工力項αr_{AB}を加えることで、反応障壁のない形状になった$F(r_{AB})$

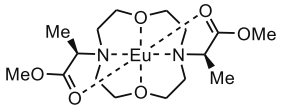
3 $Eu^{3+}(H_2O)_8$触媒を用いる反応

まず初めに、$Eu^{3+}(H_2O)_8$ 触媒を用いる向山アルドール反応に着目した。この反応では、(i)水存在下でのみ反応が進行すること、(ii)立体選択性が、水添加量を増やすことで、*anti* 体優勢から *syn* 体優勢に逆転することが実験的に知られていた。[2] まず、AFIR 法を用いて反応機構を調べることで、この反応が、C-C 結合生成、水からアルデヒドへのプロトン移動、水の求核付加によるトリメチルシリル基の脱離の順に段階的に起きていること、(i)の理由が、無水条件下では、生成物が大幅に安定化するトリメチルシリル基の脱離過程が起きず、逆反応が速く進行してしまうためであることを明らかにした。[3]

次に、(ii)の立体選択性を調べるために、それを決定する C-C 結合生成段階の TS に着目した。前述の通り、Ln^{3+}の配位構造は柔らかく、決め打ちによる TS の決定では、重要な TS を見逃す危険性がある。そこで、基質同士の配向や接近方向をランダムに変えた 400 個の初期構造(反応前の構造)を用意し、そこから、AFIR 法を用いて C-C 結合生成反応経路を探索し、それらの TS を求めてみた。その結果、C-C 結合生成段階の TS が *syn*、*anti* 体を合わせて 164 個得られ、そのうち、安定な 17 個の TS が、*syn* : *anti* 比に影響していることが分かった。*syn*、*anti* 体を与える TS のうち最も安定なものだけを考慮すると、*syn* : *anti* 比は 64 : 37 で、実験値の 73 : 27 を定性的に再現したが、得られた全ての TS を考慮すると、*syn* : *anti* 比が 75 : 25 となり、定量的に再現できた。得られた TS の解析から、(ii)の水の有無で立体選択性が逆転する理由は、水中での最安定 *anti*-TS が、基質と Eu^{3+}の配位水の間で水素結合を形成するために、構造の柔軟性が失われ、不安定化するためであると説明できた。[3,4]

4 不斉 Eu^{3+}DOTA 誘導体を用いる反応

次に、不斉 DOTA 誘導体[5]を配位子にもつ Eu^{3+} 触媒(Scheme 2)を用いる同じ反応の立体選択性の発現機構に着目した。まず、この触媒の安定構造を検討したところ、3 つの conformer (A, B, C)が共存していることが分かった。(Figure 3)



Scheme 2. 不斉 Eu^{3+}-DOTA 誘導体

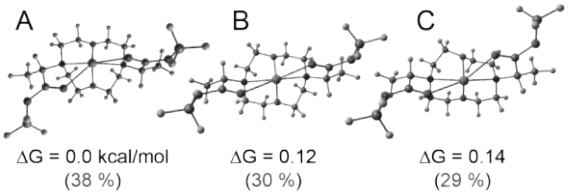


Figure 3. 安定な 3 つの触媒構造とその自由エネルギー(kcal/mol)、及び存在確率

得られた 3 つの触媒構造を用い、§3 と同様に、C-C 結合生成段階の TS を、AFIR 法により網羅的に探索した。その結果、構造 B が最も安定な TS を形成し、主立体異性体の生成に寄与することが分かった。これに対し、主生成物とは異なる立体異性体の生成には、構造 C が主に寄与していた。そこで、構造 C の存在確率を下げれば、より高い立体選択性が発現するはずであると考えた。種々の検討の結果、希土類金属を Eu^{3+}よりイオン半径の大きい Pr^{3+}に変更することで、構造 B、C の存在確率がそれぞれ増加、減少することが明らかになった。このように、柔軟な構造の触媒設計に必要なことは、触媒の構造揺らぎを抑えることだと提案することができた。

[1] S. Maeda, K. Ohno, K. Morokuma., *Phys. Chem. Chem. Phys*. **2013**, 15, 3683.

[2] S. Kobayashi, *Synlett* **1994**, 689.

[3] M. Hatanaka, K. Morokuma, *J. Am. Chem. Soc.* **2013**, *135*, 13972.

[4] M. Hatanaka, S. Maeda, K. Morokuma, *J. Chem. Theory Comput.* **2013**, *9*, 2882.

[5] Y. Mei, P. Dissanayake, M.J. Allen, *J. Am. Chem. Soc*. **2010**, *132*, 12871.

多地域景気循環の同期現象に関する大規模数値解析

江刺 邦彦*
北海道大学大学院理学院数学専攻†

1 はじめに

同期現象は物理学や生物学で頻繁に研究され, 経済現象とりわけ景気循環においても重要な現象の一つである. 同期現象に関連して, カオス的遍歴やヘテロ次元サイクルと呼ばれる現象が時折観察される. これらは高次元力学系特有の現象で, 前者は軌道が複数のカオス的状態を経巡り続ける様子を指し, 後者は不安定方向の次元が時間発展により変化する現象である.

本研究では多地域景気循環を記述する数理モデルを力学系の観点から分析し, カオス的遍歴やヘテロ次元サイクルが生じる原因や内在する数理的構造の解明を目的とする. 本モデルでは一地域の生産量を一つの素子 (変数) とみなし, それぞれの地域が全体場に影響を受けることによって互いに影響を及ぼしながら時間発展をする大域結合写像 (Globally Coupled Map, GCM) であり, アトラクターに到達するまでの計算時間が非常に長くなることがしばしばある. そのため, 通常の計算機では, 扱える次元 (素子の数) がごく低次元に限られるだけではなく, パラメータ同定等の網羅的な計算はほぼ不可能であることから, カオス的遍歴やヘテロ次元サイクルの本質的特徴を捉えるのは困難であった.

本研究ではスーパーコンピュータを利用することによってはじめて実行可能な高次元力学系の網羅的解析を行い, 多地域景気循環の同期現象をカオス的遍歴やヘテロ次元サイクルの観点で解明した.

2 多地域景気変動モデル

多地域景気変動に関する大域結合写像 (Globally Coupled Map, GCM) モデル [3] を簡潔に説明する. i 番目の地域の生産量 $x_(t)(i)$ は

$$\begin{cases} x_{t+1}(i) = (1-\varepsilon) f(x_t(i)) + \frac{\varepsilon}{N} \sum_{j=1}^{N} f(x_t(j)) \\ f(x_t(i)) = (1-\phi) x_t(i) + \frac{\phi}{(x_t(i))^{\eta}} \end{cases} \tag{1}$$

で記述される. 尚, N は地域数, ϕ は調整速度, η は価格弾力性の逆数を表す. 各要素の内生的力学は写像 $f(x)$ で表現され, $f(x)$ に関してはカオス的振舞いを示す非可逆な写像を使用する. ϕ は式 (1) の上式右辺第 2 項は平均場を通した各要素の大域的な相互作用で, 言い換えると全体から全体への相互作用である. それによって, 2 つの正反対の影響が共存する. 全体から全体への相互作用は全要素を同期させる傾向へ導き, 一方カオス的不安定性は同期しない傾向へ導く. これらの 2 つの影響のバランスである $\varepsilon \in (0,1)$ の値によって, この GCM は様々な複雑現象を生む.

本モデルは非線形性の影響による引き込みによって同期の程度が高くなり, 経済はラミナー状態として 1 クラスタ状態にしばらく滞在するが, その状態を抜け出しバースト状態としていろいろな状態の間をさまよい, それが長時間繰り返される. このような系の振舞いをカオス的遍歴と呼ぶ.

3 カオス的遍歴の次元依存性

ここでは, 多地域景気変動モデル (式 (1)) が系の次元に依らず, 同様にカオス的遍歴を引き起こすことを示す. 図 1 はモデルの次元がそれぞれ, $N = 2$ と 100 の場合のカオス的遍歴を引き起こすパラメタ領域である. これらの次元では類似したパラメタ領域でカオス的遍歴が起きることを確認した. τ はトランジエント時間, T はカオス的遍歴であるか否かを判定するために用いた時間を表している.

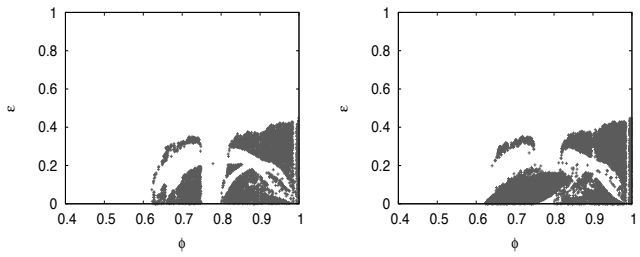


図 1: パラメタ空間 (ϕ, ε) でカオス的遍歴を起こす領域 $(N = 2, \tau = 10^5, T = 10^4)$(左), $(N = 100, \tau = 10^5, T = 10^5)$(右)

次に, カオス的遍歴がアトラクタとして実現されるかど

*本研究は立正大学経済学部の小野崎保氏ならびに一橋大学大学院商学研究科の斉木吉隆氏との共同研究に基づく.
†2014 年度より (株)NTT データ

うかを確かめるためにパラメタ領域におけるカオス的遍歴を起こすパラメタの割合がトランジエント時間を長くしていくとどうなるかを調べたところ, 図 2 のようにトランジエント時間を長くすると正の有限値に収束する傾向が見られた. このことより, 次元に依らずカオス的遍歴がアトラクタとして実現される示唆を得た.

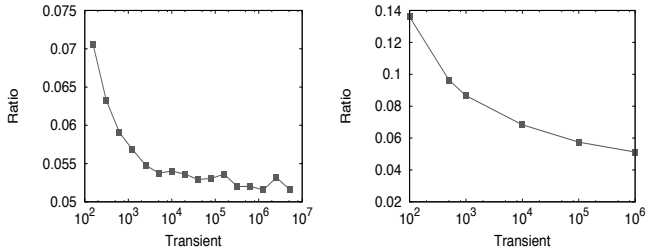


図 2: (ϕ, ε) を $(0,1) \times (0,1)$ の領域の中からランダムに選び振舞いがカオス的遍歴であったパラメタの割合とトランジエント時間の関係 ($N = 2$(左), $N = 100$(右)). トランジエント時間を長くすると正の有限値に収束する傾向が見られ, カオス的遍歴がアトラクタとして実現されることを示唆.

4 カオス的遍歴のヘテロ次元性

ヘテロ次元サイクルは, ホモクリニック接触と並び双曲力学系を破壊する代表的な要因であり, カオス軌道が不安定方向の個数が異なる構造を経巡る. 従来よりカオス的遍歴の要因の 1 つの可能性として挙げられてきた [4]. カオス的な不変集合には無限個の不安定周期軌道が埋め込まれていることが知られており, ヘテロ次元サイクルの存在は, カオス軌道が不安定方向の個数が異なる構造を経巡ること, すなわち, 不安定方向の異なる周期軌道の存在とその近傍を渡り歩くカオス軌道の存在に対応する [1].

ここでは, 不安定方向の異なる周期軌道 (サドル, ソース) がカオスアトラクタのどれだけ近くにいるのかを調べるために, カオス軌道とそれらの周期軌道の距離を測り, 一番近くを通過した時の距離を図 3 に描いた. 図 3 より, どの周期軌道にもカオス軌道がいくらでも近づくことが予想され, カオスアトラクタはこれらのサドル UPO とソース UPO を含んでいることが示唆される. これによって不安定方向の個数が異なる周期軌道の存在とそれらをつなぐ軌道の存在を数値的に確認できた. つまり, この系のカオス的遍歴の背後にヘテロ次元サイクルが存在していることが確認された.

5 まとめ

カオス的遍歴を起こすパラメタ領域を高次元で調べたところ, 低次元の場合と大きな違いはなかった. また, そ

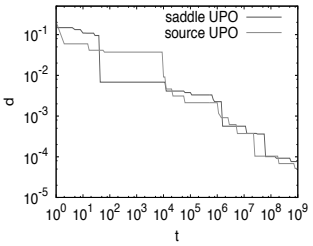


図 3: カオス遍歴軌道とバースト周期軌道の最短距離. (ϕ=0.7, η=3.5, ε=0.315)
$d(t) = \min_{\tau \leqslant t} dist(x^\tau, x_{UPO})$. x^τ は現在までの時間発展の値, x_{UPO} は周期軌道の点の値を表す.
赤: サドル周期軌道との距離, 緑: ソース周期軌道との距離. 周期軌道のいくらでも近くをカオス的遍歴軌道が通過することを示唆.

のパラメタ領域におけるカオス的遍歴を起こす比率を, トランジエント時間を変えて調べたところ, トランジエント時間を長くすると正の有限値に収束する傾向が見られたことから, カオス的遍歴がアトラクタとして実現される示唆を得た.

一方, 2 次元での詳細な解析を行った. 2 次元ではカオス的遍歴をオンオフ間欠性とみなすことができ, 同期状態 (ラミナー状態) と非同期状態 (バースト状態) を長時間行き来する. 従来よりカオス的遍歴の要因の 1 つの可能性として, カオス軌道が不安定方向の個数が異なる構造を経巡るヘテロ次元サイクルが挙げられてきた. カオス的な不変集合には無限個の不安定周期軌道が埋め込まれており, 本研究では不安定方向の異なる周期軌道の存在と, その近傍を渡り歩くカオス軌道の存在を確認した. この系におけるカオス的遍歴の背後にはヘテロ次元サイクルの構造が存在していることが確認された [2].

参考文献

[1] Bonatti, C., Díaz L. J., and Viana, M., 2005, "Dynamics Beyond Uniform Hyperbolicity", Springer.

[2] Esashi, K., Onozaki, T. and Saiki, Y., Chaotic Itinerancy in Regional Business Cycle Synchronization, submitted to Economics Letters.

[3] Onozaki, T., Yanagita, T., Kaizoji, T. and Toyabe, K., 2007, Regional business cycle synchronization through expectations. Physica A 383, 102-107.

[4] Tsuda, I., 2009, Hypotheses on the functional roles of chaotic transitory dynamics, CHAOS 19, 015113.

軌道分散の評価時間依存性から見る Lorenz system の軌道の予測可能性

中野直人 *

*独立行政法人科学技術振興機構さきがけ，北海道大学大学院理学研究院

1　はじめに

Lorenz system はカオス力学系として有名な系である．その初期値鋭敏性により初期時刻における僅かな初期値の誤差も時間が経つとともに拡大する性質を持っている．すなわち，空間内の限りなく小さな領域に限定してもその領域を通過する軌道は拡散してしまうため，pointwise の予測は原理的に不可能であるということを意味している．一方で，このように予測が不可能な系であっても，近い初期値を持つアンサンブルメンバーをいくつか取って，ある基準時刻における解の分散を計算し，初期値の分散からの拡大率（誤差拡大率に相当する）を調べることにより，その系の予測可能性の指標を相空間内にプロットすることができる．もちろん設定する評価時間や相空間の位置によってその予測可能性の指標は異なる値を取り得る．その指標のプロットを評価時間を変えながら眺めてみると，相空間のどの位置から軌道が拡散していくかを見ることができる興味深い量であると言える．ここでは Lorenz system の平面射影データに対する拡散の強さに着目し，その拡散行列を計算するための評価時間を変えることで軌道の拡散の様子を探ることにする．

2　データ

ここで用いるデータは Lorenz system

$$x' = p(y-x),\ y' = -y + x(r-z),\ z' = -bz + xy$$

の平面射影データである．ただしパラメータは $p = 10,\ r = 28,\ b = 8/3$ である．Lorenz system のデータは，初期値を $(x_0, y_0, z_0) = (1, 1, 1)$ とし，時間間隔 $\Delta t = 1.0 \times 10^{-4}$ で 4 次ルンゲ＝クッタ法で計算する．$0 \le t \le 100$ までは遷移状態とみなし，アトラクタ上の軌道である $t_0 := 100 < t \le 1100$ までの 10^7 個のデータを採取する (図 1 上)．さらにこれを主成分分析によって得られる第 1，第 2 主成分が張る平面に射影し，これによって得られた時系列データ $\{\boldsymbol{X}_{t_n} = (u_{t_n}, v_{t_n})\}_{n=1}^{10^7}$ $(t_n = 100 + n\Delta t)$ を解析する時系列データとする (図 1 下)．

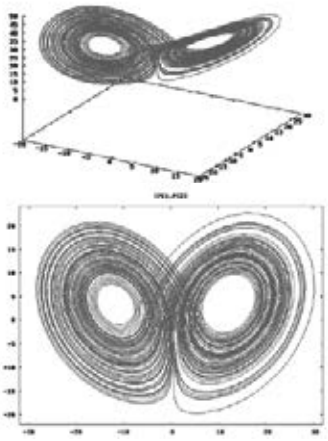


図 1: (上) Lorenz system の数値解による軌道，(下) 平面に射影された Lorenz system の軌道．

3　手法

平面を長さ 0.1 の正方形のセルに分割し，各セル $\boldsymbol{x}_m$ を通過するデータ点に対して，ある基準時刻 τ 後のデータ点の分散の変化率を計算する：

$$B(\tau)_m = \frac{1}{|T_m|}\sum_{t_n \in T_m} \frac{|\boldsymbol{X}_{t_n+\tau} - \boldsymbol{X}_{t_n}|^2}{2\tau} - \frac{\tau}{2}\left(\frac{1}{|T_m|}\sum_{t_n \in T_m} \frac{\boldsymbol{X}_{t_n+\tau} - \boldsymbol{X}_{t_n}}{\tau}\right)^2.$$

ただし，T_m はセル $\boldsymbol{x}_m$ を通過する軌道の時刻の集合である．したがって，$B(\tau)$ はセル $\boldsymbol{x}_m$ を通過するデータ点に対するアンサンブル平均を用いて計算される分散となっている．この量を全てのセルで計算してプロットすることで，相空間内の分散の多寡から予測可能性を解析する．

4　結果

$B(\tau)$ を $\tau = 0.01$ 刻みで 5.00 までの 500 通りで計算し，その変化を観察した．平面データ軌道は予め数値計算によって作成してあるため，$B(\tau)$ の計算は MPI による並列計算を用いている．ここでは，紙面の都合上，図 2 に $\tau = 0.01, 0.40, 0.80, 1.20, 1.60$ のときの $B(\tau)$ の平面プロットを示す．

ここからわかるように，各評価時間において，軌道の分散の度合いが強い（予測可能性の低い）領域は相空間の中で偏在していることがわかる（赤の領域が分散の度合いが強く，青が弱い）．例えば，$tau = 0.01$ のケースでは，非線型性が効く前のごく短い時間の後の軌道の分散を計算しているため，軌道交差領域のみが分散の度合いが強い．

また，$\tau = 0.80$ のケースでは，軌道は少なくとも半周は廻っており，カオス性の影響を受けたデータの分散の度合いを計っていることになる．実際，0.80 後の時間に軌道が大きく分散する領域というのは，軌道交差部だけでなく，Lorenz バタフライの翅の内側部にも緑色の筋として現れている．

このことより，軌道がカオス性により発散していく性質は特定の領域を通過することで生じることがこの図から示唆される．予測可能性の善し悪しが相空間に部分的（筋的）に表現されていることが興味深いことである．そして，τ を大きくしてその緑の筋を辿っていくことにより，軌道が分散していく経路を見出すことができるため，Lorenz system の軌道の葉層構造の解析につながることが期待される．∎

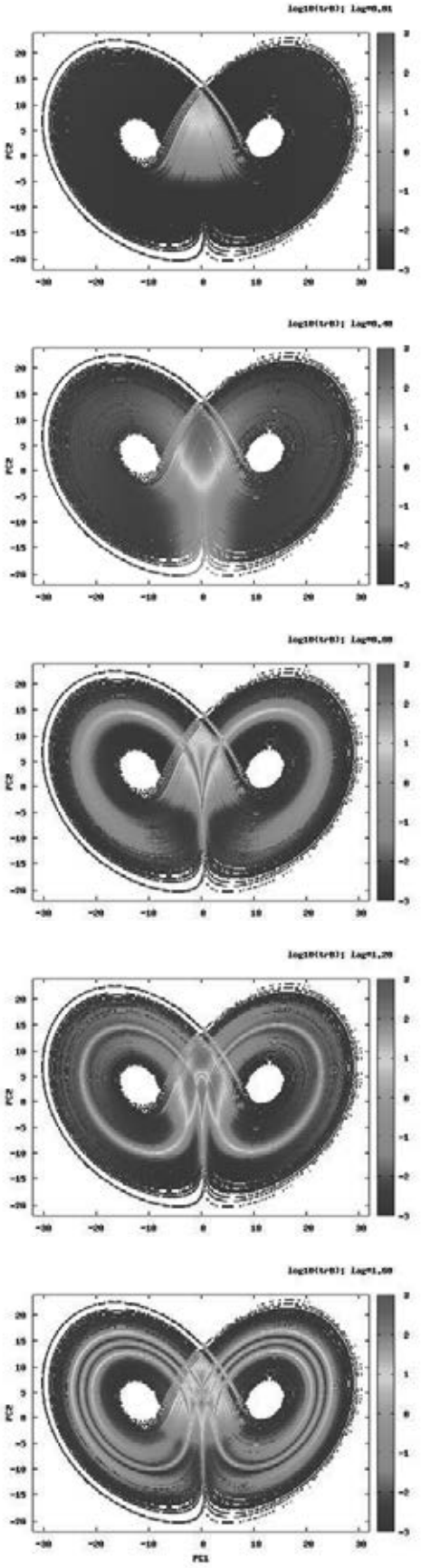


図 2: $B(\tau)$ のプロット．上から，$\tau = 0.01, 0.40, 0.80, 1.20, 1.60$.

参考文献

[1] 中野 直人, 稲津 將, 楠岡 誠一郎, 齊木 吉隆, 「時系列データに対する確率微分方程式モデルの統計的係数決定公式と軌道の予測可能性について」, 京都大学数理解析研究所講究録 **1919**, pp 38–60, (2014).

溶融紡糸工程におけるドローレゾナンス現象のマルチスケールシミュレーション

高瀬 和大*, 谷口 貴志*

*京都大学大学院工学研究科 化学工学専攻

1 はじめに

高分子流体では、高分子鎖の配向や絡み合いなどのミクロスケールの構成要素の特徴が，マクロスケールの流動特性に強く影響を及ぼすことが知られている．今までの高分子材料の研究で行われてきた多くの流動シミュレーションでは，流体の応力算出のために経験論的な構成方程式が用いられてきた．しかし，構成方程式を用いたマクロスケールの流動解析ではミクロな構成要素の特徴とそれらの集合体のマクロな性質を” 直接” 関係づけることができない．一方，分子動力学に基づくミクロスケールのシミュレーションを巨視的な空間スケールで，かつ，マクロスケールの特徴時間まで行うことは，計算負荷の観点で現実的ではない．

実際，高性能な高分子材料を開発するには，マクロな流動挙動と高分子のミクロな状態という非常に異なるスケールの問題を同時に扱わなければならない．そのため，スケール間の関係をより詳しく扱えるようにミクロモデル・メソモデル・マクロモデルを相互に組み合わせて行う Multi-Scale Simulation(MSS) 法の確立が強く望まれてきている．MSS では, その方法から明らかになように莫大な量の計算が必要となっている．そのため，計算の並列化が必要不可欠であり，高性能な CPU や GPU が利用できる京都大学学術情報メディアセンターの共同研究制度を利用させて頂いた．

本研究では高分子溶融紡糸工程にマルチスケールシミュレーション法を適用し，巨視的なレオロジー特性と微視的な構造変化との関係を明らかにすることを目的として研究を行った．

2 溶融紡糸工程と数値計算手法

紡糸工程では，ノズルから押し出された高分子溶融体を巻き取り部で高速で巻き取ることで張力を与えており，エアギャップと呼ばれる短い空気間隔で伸長させた後，水などの媒体に入れることで冷却し成形が行われる (Fig.1)．ノズル出口の速度 V_o と巻き取り速度 V_w の比 V_w/V_o はドロー比と呼ばれている．ドロー比がある一定の値より大きくなると伸長流動の不安定性が起こり，糸の断面積が周期的に変動する現象 (ドローレゾナンス現象) が発生することが知られている．また，エアギャップ長や分子量分布も安定性・材料特性に影響を及ぼす [1,2].

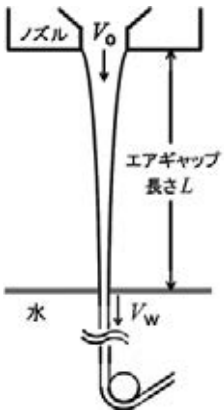


図 1: Schematic view of a polymer melt spinning process.

この紡糸過程は以下で述べる式で表現できる．これらの式では糸の状態を糸の速度 V，断面積 A を用いて表現しており，断面積は連続の式により，速度場は力の釣り合いの式から求めることができる．

$$\frac{\partial A(x,t)}{\partial t} = -\frac{\partial}{\partial x}\Big(A(x,t)V(x,t)\Big) \quad (1)$$

$$\frac{\partial}{\partial x}\Big(A(x,t)\sigma(x,t)\Big) = 0 \quad (2)$$

$$\sigma \equiv \sigma_{xx} - (\sigma_{yy} + \sigma_{zz}) \quad (3)$$

ここで，x は座標，t は時間，σ は応力差である．

応力は高分子の微視的な状態から決定され，その値は高分子鎖の過去の変形履歴に依存するため，ラグランジュ的描像に基づき，流動点上で応力を算出した (Fig.2)．応力の算出には，高分子の微視的モデ

ルの 1 つである Slip-link モデル [3] を用いた．Slip-link モデルは，絡み合いによる束縛のために各高分子鎖がチューブ状の領域を運動するという理論を表現している (Reptation 運動)．各高分子鎖は絡み合い点 (Slip-link) を結んだ線 (Primitive path) に沿って自由にスライドすることができ，各 Slip-link はバーチャルな絡み合い情報をもつ．具体的には以下の 5 つの手順で計算を行った [3](Fig.2)．

1. 流れ場によるアフィン変形
2. 式 (4) に基づく高分子の長さの緩和
3. Primitive path に沿った鎖の Reptation 運動
4. 絡み合い点 (Slip-link) の生成・消失
5. 式 (5) による応力算出

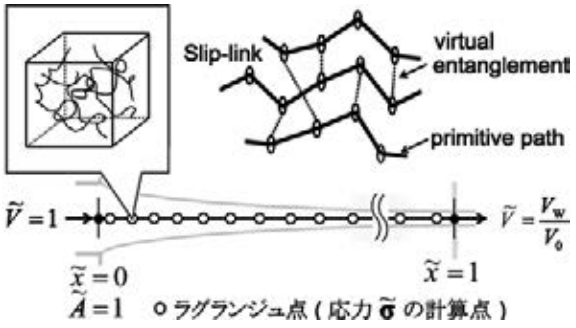


図 2: Lagrange points on a spinline and schematic view of polymer chains expressed by the slip-link model

$$\frac{dL}{dt} = -\frac{1}{\tau_{\mathrm{R}}}(L(t) - L_{\mathrm{eq}}) + \left(\frac{dL}{dt}\right)_{\mathrm{affine}} + g(t) \quad (4)$$

$$\sigma_{\alpha\beta} = \sigma_{\mathrm{e}}\left\langle \frac{r_\alpha r_\beta}{a|\boldsymbol{r}|} \right\rangle \qquad \alpha,\ \beta \in \{x, y, z\} \qquad (5)$$

ここで，L は高分子鎖 (primitive path) の全長であり，(平衡状態では L_{eq}=$Z_{\mathrm{eq}}a$, a は Slip-link 間の平均距離) τ_{R} は Rouse 緩和時間，$g(t)$ は揺動散逸定理を満たす熱揺動力，$\boldsymbol{r}$ は隣接する Slip-link の結合ベクトル，σ_{e} は Slip-link モデルにおける応力の単位である．$\langle(\cdots)\rangle$ は 1 つの Lagrange 点中の高分子鎖の全てのストランドについての統計平均である．Z_{eq} は 1 本の高分子の平衡時の絡み合い数であり，絡み合い数は分子量 M と絡み合い点間分子量 M_{e} を用いて $Z_{\mathrm{eq}} \equiv M/M_{\mathrm{e}}$ と表される．また，応力は，各ラグランジュ点での高分子鎖の統計平均から計算される．具体的には，式 (5) に示すように，高分子鎖上で隣接する Slip-link 間に働く張力から算出される．式 (1)〜(5) を無次元化して計算を行う際，無次元パラメータであるデボラ数 $\mathsf{De} = \tau_{\mathrm{R}}V_{\mathrm{o}}/L$ が現れる．デボラ数は，紡糸系の代表滞留時間 $t_{\mathrm{o}} = L/V_{\mathrm{o}}$ と高分子鎖の伸びの緩和時間 τ_{R} との比を示している．

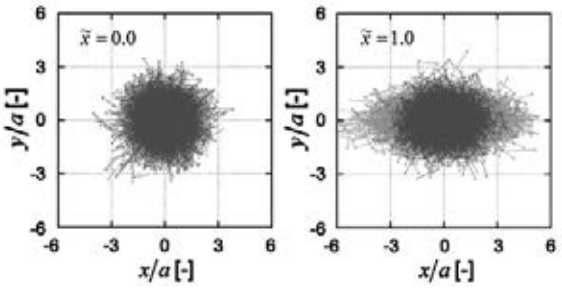


図 3: Configuration of polymer chains located at $\tilde{x}$ = 0.0 (left), 1.0 (right) for the draw ratios=3.0 (blue), 6.0 (green), 8.0 (orange). Number of polymer chains on a Lagrange point n_{p}=1000 and the average number of entanglements at equilibrium is $Z_{\mathrm{eq}} = 10.4$. (x : flow direction, y : cross sectional direction)

3 結果および考察

系にラグランジュ点を約 3000 点導入し，各ラグランジュ点上に 1000 本の高分子鎖が存在するとしてシミュレーションを行った．各高分子鎖は平均約 10 個の絡み合い点を持つため，系全体では約 9000 万の自由度を同時に扱うことになる．Fig.3 に 1 つのラグランジュ点が紡糸線上を流動する際のミクロな高分子鎖の構造変化についての結果を示す．まず，高分子鎖の伸びと配向に関する結果として，1 つのラグランジュ点上に存在する全ての高分子鎖の構造を重ね合わせた図を Fig.3 に示す．それぞれドロー比 3.0, 6.0, 8.0 について，座標 $\tilde{x} = 0$ と $\tilde{x} = 1$ 付近を通過する際の高分子鎖の状態を示している．グラフから，初期状態ではランダムに配向していた高分子鎖が次第に流れ方向に伸びながら配向していくことがみてとれる．加えて，ドロー比が大きくなるほど変形が顕著になっている．次に，同様の条件の絡み合いの状態の変化を Fig.4 に示す．

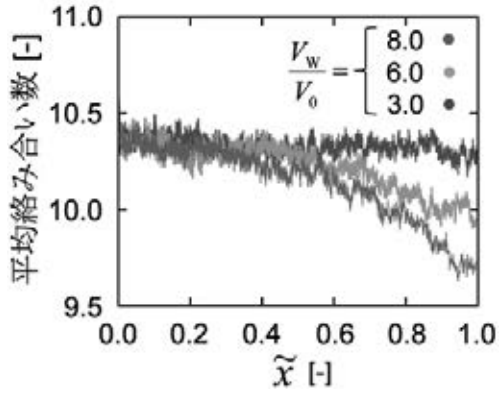


図 4: Change in the averaged entanglement number of polymer chains in a Lagrange point as a function of its position on a spinning line.

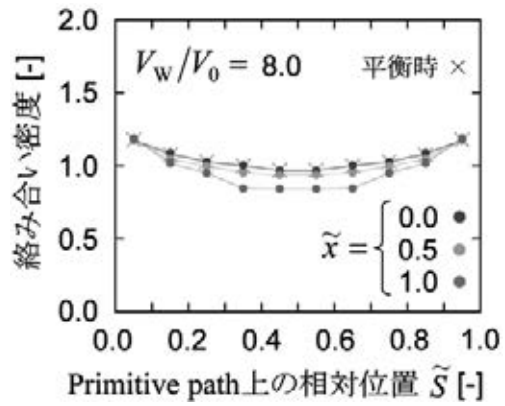


図 5: Entanglement density along primitive path at $\tilde{x} = 0.0, 0.5, 1.0$ for $V_w/V_o = 8.0$.

Fig.4 は，1 つのラグランジュ点上の高分子鎖の平均絡み合い数をラグランジュ点の座標の関数として示したものである．Fig.4 から，ドロー比が大きくなると絡み合い数がエアギャップ末端部で減少することがみてとれる．また，Fig.5 には各々の高分子鎖の Primitive path 上における絡み合い密度を示す．絡み合い密度は Primitive path を 10 分割した各区間の平均絡み合い数であり，絡み合い数が減少する際には，高分子鎖の中央付近の絡み合い点が解放されていることがわかる．このことは，絡み合いが消失した後，中央付近は末端部よりも絡み合いが再生成されにくいことに起因していると考えられる．

4　結言

Slip-link モデルを用いたマルチスケールシミュレーションを紡糸工程に適用することによって，巨視的な流動状態の変化と微視的な配向・絡み合い構造との関係を明らかにした. 先に述べたように，本研究で用いた手法は，多数のミクロ系を同時に計算するため，処理する計算量が膨大となる．そのため本共同研究制度で並列計算機を利用させて頂けたことで並列計算のプログラム開発を滞りなく進めることができ，また，計算結果を修士論文としてまとめることができた．ここに感謝の意を表したい．今後は，並列計算を更に効率よく行うことで計算負荷を低減させ, 力のバランスを計算する部分のアルゴリズムを改善することで，現在まだ計算することできていないドロー比が大きな領域（紡糸系が流動不安定を示す領域）でマクロ・ミクロの相互関係を検討することや，分子量分布をもつ系などのさらに複雑なミクロな構造を検討していくことが課題である．以上のように，ミクロとマクロを直接つなぐマルシスケールシミュレーション法は，現在も発展段階にあり，更なる発展のためには並列計算機の大規模な利用が必要不可欠なため，スーパーコンピュータの役割は重要となるだろう．

参考文献

[1] 石原英昭：成形加工, **23**(5), 276-285 (2011).

[2] 石原英昭：成形加工, **23**(6), 336-346 (2011).

[3] M. Doi and J. Takimoto, *Phil. Trans. R. Soc. Lond.* **A 361**, 641-652 (2003).

高次精度差分法による高レイノルズ数乱流場における大規模構造の直接数値シミュレーション

山本　義暢

山梨大学大学院総合研究部

1　はじめに

近年，高レイノルズ数壁面乱流場に出現する大規模構造に注目が集まっている．この大規模構造は境界層スケールでスケーリングでき，時空間大きな揺らぎを伴うこと特徴であり，平均量及び乱流統計量分布にも大きな影響を与え，乱流抵抗・輸送・制御においてその解明は重要である．

大規模構造の統計的影響は壁面摩擦速度と境界層厚に基づくレイノルズ数(Re_τ)で 10^3 程度から出現するが，その影響が顕著となるのは，Re_τ = 7000 程度と実験的には報告されている[1]．この大規模構造を解析する手法としては，直接数値計算 (Direct Numerical Simulation, DNS) が最も有力であるものの，これまでに実行された壁面乱流場の最高レイノルズ数は 4000 程度[2], [3]であり，実験的に報告されているレイノルズ数領域での実行例は未だ存在しない．

本研究では世界最大規模の高レイノルズ数乱流場の直接数値計算を目標として，高速かつ大規模直接数値計算コード開発を行う．

2　直接数値計算の概要

本研究における直接数値計算の解析対象は，図 1 に示す，一定の圧力勾配で駆動される 2 次元チャンネル乱流場である．ベースとした直接数値計算コードはの基礎方程式は，非圧縮性流体の運動方程式，連続式及びパッシブスカラを仮定した温度場の輸送方程式である[4]．

空間離散化手法は，主流及びスパン方向に 10 次精度の中心差分法[5]，壁垂直方向に 2 次精度の中心差分を使用する．時間進行は，対流項及び粘性項に 3 次精度の低容量型 Runge-Kutta 法，圧力項に Euler 陰解法を適用し，Fractional step 法により解く．この際圧力ポアソン解法には，高速フーリエ変換 (x, z 方向) と 3 重対角行列解法 (y 方向)による直接解法を使用する．境界条件は，主流及びスパン方向に周期境界条件，壁面で no-slip 条件を課した．温度場に対しては壁温一定 (上壁温度:θ_{top} > 下壁温度:θ_{bed}) とし，主流及びスパン方向には周期境界条件を課した．

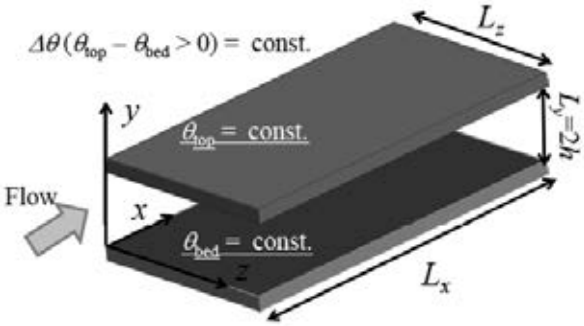


Fig.1 Computational domain and coordinate system.

3　並列化方法の概要

3.1　領域分割/MPI

本研究では，壁垂直方向 (y 方向) への MPI (Message Passing Interface) による領域分割による並列化を適用する. この際に必要となる通信は，壁垂直方向の差分演算時に必要となる袖通信，圧力ポアソン方程式の 3 重対角行列解法時の all-to-all 通信である．袖通信においては，隣接間

通信となるが，通信データを連続データとするために，プログラミング上，y 軸方向は，3 次元配列の最外ループとした (fortran の場合).

また all-to-all 通信は，3 重対角行列解法時に，分割方向 (y 方向) に再起参照関係となるために，これを回避するために分割軸の入れ替えを行うが，この際に必要となる．従来の並列化においては分割軸の入れ替えを z 軸方向に行っていたが，チャンネル乱流場では y 軸と z 軸方向の格子数が異なり，各プロセス間での通信量及び演算量のアンバランスが生じる可能性があった. そこで今回，分割軸の入れ替え時に転送配列を(x, z, y) の 3 次元配列から，(xz,y) の 2 次元配列とし，N_xN_z の格子数を MPI プロセスにより領域分割した.

3.2 OpenMP

また通信量の削減と並列ノード数を稼ぐことを目的として，ノード内は，OpenMP による並列化を計算コードのほぼ全領域に適用した．また 3 次元計算の最外ループ (y 軸) で領域分割を行っている関係上，最外ループをさらに OpenMP で分割すると総並列数が結局 1 次元方向の格子数に制限されるため，OpenMP における collapse 機能を使用して，これを回避した．また 2D-FFT 部分においては，FFTW3.3 の OpenMP 版を適用した.

3.3 メモリ量削減

10 次精度差分法では，1 点の微係数を計算する際に， 10 点（拡散項では，20 点）のステンシルを要する．これに対処するために境界部分にステンシルを拡張すると大幅なメモリ増加をもたらすため，境界部分についてはリスト配列を使用し，メモリ量は高次精度差分法においても，2 次精度差分法の場合と同等となるようにした.

4 DNSに要する計算機資源と演算速度の見積

4.1 時間積分長

乱流の DNS においては発達乱流場を得るための助走計算と，その後長時間平均による統計量計算を行うためのプロダクト RUN を実行する．この助走計算に要する時間積分長は，渦の turn over time (h/u_τ, h チャンネル半幅，u_τ 壁面摩擦速度) の N_0 倍程度，プロダクト RUN には，N_1 倍程度とすると．1 タイムステップあたりの時間刻(Δt^+ 上付き添え字は摩擦速度と動粘性係数により無次元化された値であることを示す)をもちいて，必要タイムステップ数(N)は，次式(1)で求められる.

$$N = (N_0 + N_1)\,\mathrm{Re}_\tau / \Delta t^+ \tag{1}$$

4.2 必要演算速度

直接数値計算コードにおいて，1 タイムステップの 1 格子点当たりに RG[Gflop/grid] の演算量とする．さらに Re_τ =1000 の場合の格子点数を主流，壁垂直，スパン方向にそれぞれ，N_x, N_y, N_z の格子点数を必要とし，各方向にレイノルズ数の増加に伴い，Re_τ に比例して格子点数が増えるものとすると，必要演算量[Gflop]は，式(2)で求められる.

$$R = NR_G\left\{N_xN_yN_z(\mathrm{Re}_\tau/1000)^3\right\} \tag{2}$$

この計算を約 100 日で終了させるとすると，この計算に必要な実効演算速度 Rmax[Tflops]は，次式(3)となる.

$$R_{\max} = R/(100\times 24\times 60\times 60) \tag{3}$$

5 DNSに必要な格子分解能の算定

DNS においては，最大スケールから最小スケールを解像できる計算領域と格子分解能を適用する必要がある．最小スケールは一般にはコルモゴロフスケール程度と見積もられる．しかし乱流エネルギの散逸は，コルモゴロフスケールの 10 倍程度により担われていることから，コルモゴロフスケールの数倍程度の解像度を適用した DNS の適用例も多い．また差分法に基づく DNS の場合は，分解解像内においても，差分近似精度に由来する近似誤差が発生しその影響を加味する必要が生じる．そこで本研究では，大規模構造が出現する最低限度のレイノルズ数である，Re_τ = 1000 を対象とし，渦運動のフルスケールを解像する FTS(Full Turbulence Simulation) と低次統計量の再現性を確保した最低限度の DNS の 2 種類の基準データをスペクトル法(Pseudo-Spectral Method, PSM)コード

[6]を用いて作成した．次に 10 次精度差分法(10th order accuracy Finite Difference Method, 10th FDM)の場合に主流方向 (x, z 方向) への解像度を変化させ格子分解能の算定を行った．この計算条件を表 1 に示す.

Table 1 Numerical conditions

Method	Re_τ	Grid number (N_x, N_y, N_z)	Resolution ($\Delta x^+, \Delta y^+, \Delta z^+$)
PSM	1000	1920, 1032, 1280	6.7, 0.5-2.0, 5.0
PSM	1000	768, 512, 768	16.7, 0.5-8.0, 8.3
10th FDM	1000	512, 512, 768	25.0, 0.5-8.0, 8.3
10th FDM	1000	640, 512, 768	20.0, 0.5-8.0, 8.3
10th FDM	1000	768, 512, 768	16.7, 0.5-8.0, 8.3
10th FDM	1000	960, 512, 768	13.3, 0.5-8.0, 8.3
10th FDM	1000	960, 512, 1024	13.3, 0.5-8.0, 6.3
10th FDM	1000	1152, 512, 768	11.1, 0.5-8.0, 8.3
10th FDM	1000	1280, 512, 1024	10.0, 0.5-8.0, 6.3

5.1 基準データベースの精度確認

図 2 に本レイノルズ数における壁方向へのコルモゴロフ長(l_K^+)解像に必要な格子分解能を示す．コルモゴロフ波数を解像するのに必要な空間解像度は壁近傍で小さく($\Delta^+ = 4.5$)，壁面から離れると次第に大きくなることがわかる($\Delta^+ = 17$)．FTS として設定したケースの最大格子幅は 6.7 程度であるため，壁面近傍ではコルモゴロフスケールの 70%程度 ($k_x l_K = 0.7$) の解像度となる．しかし図 3 に示す通り，壁面近傍では，主流及びスパン方向への散逸スペクトルは，$k_x l_K$=0.4 程度でほとんどゼロとなっていることから，本解像度において十分な信頼性を有していると結論付けられる．

また解像度が低いケースの結果(図 3 中, 赤色表示)においてもコルモゴロフスケールまでは解像できていないものの，散逸スペクトルのピークは完全に解像できていることが確認できる．実際スペクトル法においては，本解像程度 ($\Delta x^+ = 18$, $\Delta z^+ = 9$) 程度の分解能を確保した場合は，低次統計量においてはFTSの結果とほとんど差異は見受けられない(図省略).

以上より，スペクトル法の高解像ケース($\Delta x^+ = 6.7, \Delta z^+ = 5.0$) を FTS データとし, 低解像度ケース ($\Delta x^+ = 17, \Delta z^+ = 8$) のケースを DNS としての最低限の条件ケースとして使用する.

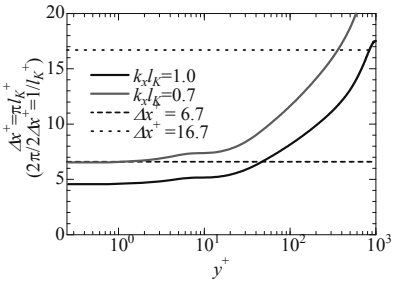


Fig. 2 Spatial resolution to resolve the Kolmogorov's wave number

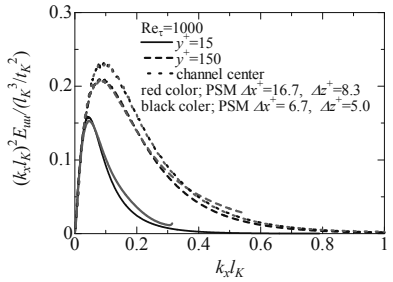


Fig. 3 Dissipation spectra in cases of PSM.

5.2 必要格子分解能の算出

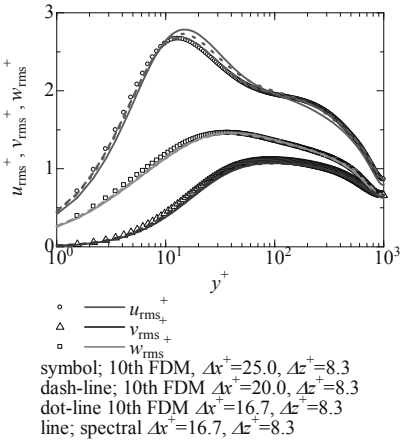


Fig. 4 Grid dependency of turbulent intensities in 10th FDM coarse grid cases.

次に 10 次精度差分法における乱流強度分布における格子依存性の影響を図 4 及び 5 に示す．図 4 に示す低解像度のスペクトル法($\Delta x^+ = 17$, $\Delta z^+ = 8$)の以下の格子分解能では，主流方向の乱流強度分布を過小評価することがわかる．但し，低解像度と同等の分解能では多少改善している.

図 5 に示す低解像度のスペクトル法($\Delta x^+ = 17$, $\Delta z^+ = 8$)の以上の格子分解能を用いた場合は，主流方向ピーク値は解像度が($\Delta x^+ = 13, \Delta z^+ = 8$)の場合，若干低めに算出されるものの，($\Delta x^+ = 11.1, \Delta z^+ = 8$)

では，ほぼ同等の結果となる．またスパン方向に高解像度格子を適用した($\Delta x^+ = 13$, $\Delta z^+ = 6$)のケースにおいても，主流方向乱流強度のピーク値に関しては，($\Delta x^+ = 13$, $\Delta z^+ = 8$)のケースとほとんど変化しなかった(図省略).

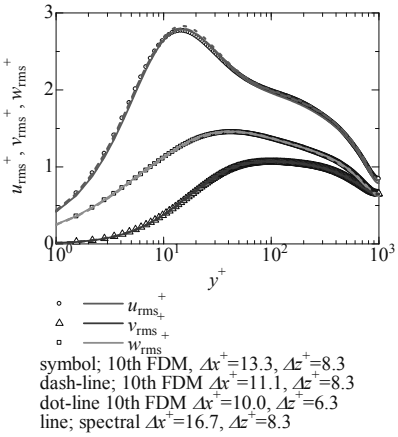


Fig. 5 Grid dependency of turbulent intensities.

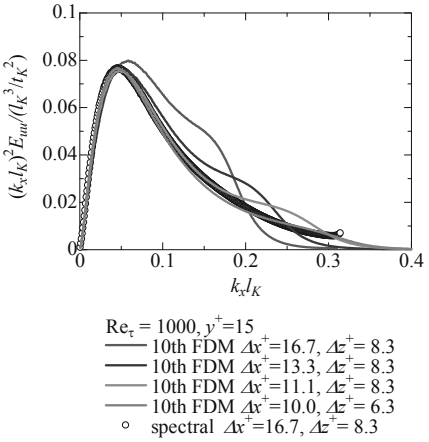


Fig. 6 Grid dependency of dissipation spectra at buffer layer.

以上より，10 次精度差分法を適用する場合は，低次統計量の観点からは，スペクトル法と同程度以上の分解能は必要であり，その解像度の影響は主流方向の格子解像度の依存することがわかる．

図 6 に主流方向乱流強度分布のピーク位置 (y^+ = 15) 付近における主流方向の散逸スペクトルの比較を示す．スペクトル法と同等の解像度(Δx^+ = 17, Δz^+ = 8)の場合(図中の赤線), 散逸ピーク付近から解像度不足による過剰評価が見られる．一方その他のケースでは，散逸ピーク近傍ではスペクトル法の結果と良好に一致し，乱流強度分布の結果をよく説明できる．

以上の乱流強度分布及びスペクトル解析の結果より，10 次精度中心差分法で必要な格子分解能は，最低($\Delta x^+ = 13$, $\Delta z^+ = 8$)程度が必要と見積もれる．

5.2 必要演算速度の算出

時間積分長，演算速度，格子分解能について，$Re_\tau = 1000$ の場合に実測した値を表 2 にまとめる．

Table 2 Numerical costs in case of 10th FDM DNS code

N_0	N_1	Δt^+	$N_x N_y N_z$	R[Gflop]
10	3	0.036	960x512x768	1.6×10^{-6}

そして表 2 の値を用いて評価した，高レイノルズ数乱流場の DNS に必要な実効演算速度を図 7 に示す．現世界最高レイノルズ数の 4000 は，10 [Tflops]，目標とする 8000 では 100[Tflps]程度の実効演算速度が必要となることが見積もれる．

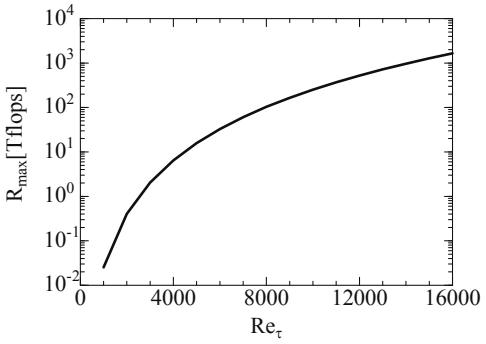


Fig. 7 Estimation of effective computational operation speed.

6 開発コードの並列性能

6.1 10 次精度差分法（CRAY・XE6）

図 8 に京都大学学術情報メディアセンタの CRAY・XE6 おいて，計測した 10 次精度中心差分法 DNS コードの演算パフォーマンスを示す．なお計算条件は，表 1 に示す，Re_τ =1000, 1152x768x512 のケースで強スケーリングによる測定結果である．本計算条件に必要なメモリ量より最小ノード数は 2 (64 cores)となる．また測定結果は，2 ノード使用時の全体計算時間で規格化している．

本コードにおける主要演算部分については，良好な並列性能が確認できる．

ただし，赤点線で示した all-to-all 通信部分については，192 コア使用時(6 ノード)に通信時間の増加が見られ，並列数の増加に比例する通信時間の短縮は得られていない．

しかし，all-to-all 通信部分のコスト比率は現並列数においては，比較的低いため，青点線で示した全体計算時間は，理論並列性能 (並列化率 100% ，黒点線)と同等の値を示している．2 ノードと 8 ノードの結果により算出した並列化率は，99.975%，並列化率は 93.9%であった．また 8 ノード時の実効演算速度は，約 222 [Gflops]であり，これは理論性能の約 8.7%に相当し，十分な並列性能及び実効効率が得られていると結論付けられる．

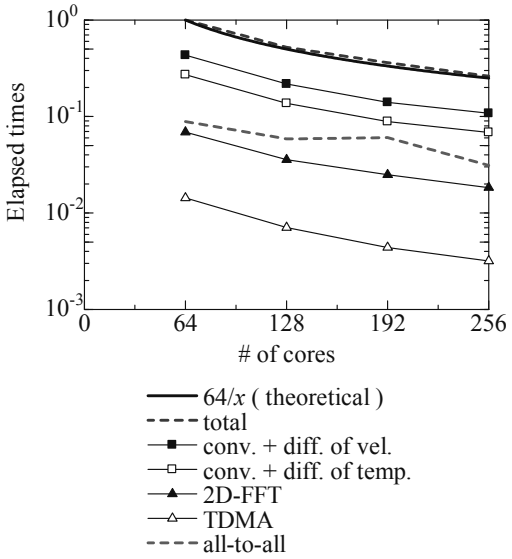


Fig.8 Elapsed times by CRAY XE6, in case of 10th FDM. Re_τ =1000, 1152x768x512 grids

6.2 6 次精度差分法（CRAY・XE6）

また本研究では，6 次精度差分法のコード開発も併せて実施した．上記と同一条件における 6 次精度差分法コードの並列性能を図 9 に示す．全体的な傾向は 10 次精度差分法の場合と同様であり，192 並列時の all-to-all 通信において性能劣化が確認できる．また 10 次から 6 次への演算コストの低減に伴い，全体に占める通信コストが若干増加している．2 ノードと 8 ノードの結果により算出した並列化率は，99.958%，並列化率は 90.2%であった．また 256 並列時の 1 タイムステップあたりの計算時間は 10 次精度の場合の約 70%程度であった．

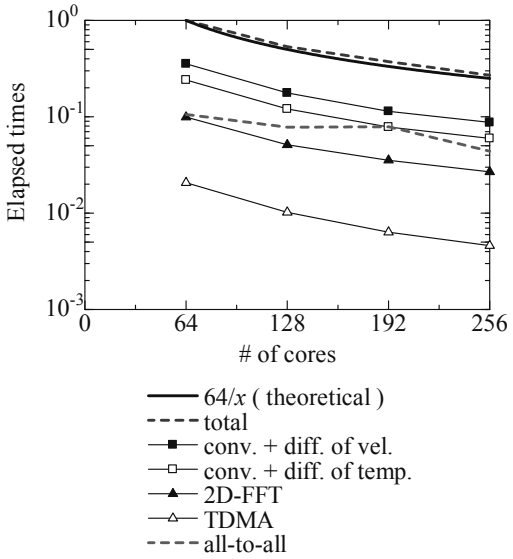


Fig.9 Elapsed times by CRAY XE6, in case of 6th FDM. Re_τ =1000, 1152x768x512 grids

6.3 10 次精度差分法（Fujitsu・CX400）

本計算条件では，高々8 ノードであるため，ノード数の増加に伴う all-to-all 通信のコスト増加が懸念される．そこで，九州大学情報基盤センタの Fujitsu・CX400/64 ノード(1024 cores)を使用したベンチマークを併せて行った．計算条件は CRAY・XE6 の場合と同様である．図 9 にその計測結果を示す．CX400 は 1 ノードあたりのメモリ量が 128GB であるため，1 ノードにおけるベンチマークが可能である．また計算結果は，1 ノード使用時の全体計算時間により規格化している。並列数が増加した場合においても，本コードにおける主要演算及び通信部分について全てにおいて良好な並列性能が確認できる．また all-to-all 通信部分については，効率はやや落ちるもののノード数の増加に伴い，通信時間の減少が確認できる．CX400 システムのノード間接続はは Fat-Tree によるネットワークトポロジであり，3D トーラスタイプの XE6 に比べ，all-to-all 通信において優位性があるものと考えられる．

従って，青点線で示した全体計算時間は，理論並列性能 (並列化率 100% , 黒点線)と同等の値を示している．1 ノードと 64 ノードの結果により算出した並列化率は，99.983%，並列化率は 85%であった．また 64 ノード時の実効演算速度は，約

1.54 [Tflops]であり，これは理論性能の約 7%に相当する．

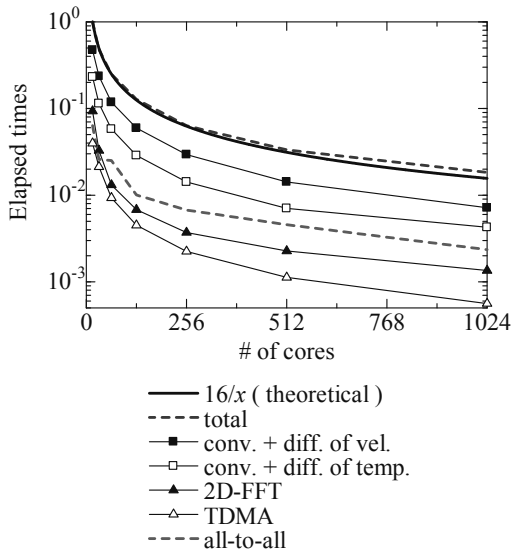


Fig.10 Elapsed times by Fujitsu CX400, in case of 10th FDM. Re_τ =1000, 1152x768x512 grids

7 まとめ

本研究では，高レイノルズ数乱流場における大規模構造解析を目的として，高次精度差分法(10次精度)に基づく直接数値計算コードの開発とその性能評価を行った．その結果，

(1) 実験的手法により報告されている高レイノルズ数乱流場を対象とした DNS データベースの構築には，Re_τ =4000 の場合に 10[Tflops], Re_τ = 8000 の場合に, 100 [Tflops] 程度の実効演算速度を確保する必要があることが分かった．

(2) 本開発コードにおいては，10 次精度差分法コードの場合，京都大学 CRAY・XE6 を使用した場合に，強スケーリングベンチマークにおいて，並列化効率 94%, 並列化率 99.975% の高効率計算を実現した．

(3) 同様に 6 次精度差分法コードにおいても，並列化効率 90%，並列化率 99.958%を達成した．

(4) さらに 10 次精度差分法コードの場合，九州大学 CX400・1024 並列使用時において，約 64 倍の強スケーリングベンチマークにおいて，並列化効率 85%, 並列化率 99.983% の高効率計算を実現し，並列数が増加した場合においても十分な並列性能を確保できることを確認した．

本開発コードは，高次精度差分法に基づくもので有り，スペクトル解析等においても，スペクトル法と同等の精度が確保できており，有用であると考えている．

今後は開発コードを用いて，実際に世界最大規模の高レイノルズ数領域における大規模直接数値計算を実施する予定である．

謝辞：プログラム開発においては京都大学学術情報メディアセンタ及び CRAY JAPN によるサポートをいただきました．また本スペクトル法の DNS データは，東北大学サイバーサイエンスセンタ SX-9 を利用して得られたものです．記して謝意を表します．

参考文献

(1) Hutchins, N., & Marusic, I. (2007), Evidence of very long meandering features in the logarithmic region of turbulent boundary layers, J. Fluid Mech., 579, 1-28.

(2) Lozano-Durana, A. and Jimenez, J. (2014), Effect of the computational domain on direct simulations of turbulent channels up to Re_τ = 4200, Phys. Fluids, 26, 011702.

(3) Bernardini, M., Pirozzoli, S., and Orlandi, P. (2014), Velocity statistics in turbulent channel flow up to Re_τ =4000, J. Fluid Mech., 742, 171-191.

(4) Yamamoto, Y. Kunuig, T., and Serizawa, A., Turbulence statistics and scalar transport in an open-channel flow. J. Turbulence, 2(10), 1-16.

(5) Morinishi, Y., Lund, T. S, Vasilyev, O.V., and Moin, P. (1998), Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow, J. Comp. Phys. 143, 90-124.

(6) Yamamoto, Y., & Kunugi, T. (2011), Discussion on heat transfer correlation in turbulent channel flow imposed wall-normal magnetic field, Fusion Eng. Des., 86(12), 2886-2890.

動的スケジューリング版タイル QR 分解の MPI/OpenMP ハイブリッド実装

鈴木 智博 *

*山梨大学

1 はじめに

密行列の数値解法では，前処理によって元の行列を何らかの望ましい性質を持つ行列に変形した後に処理が行われることが多い．これを前処理と呼ぶ．さらに，計算時間全体に占める前処理（と後処理）の時間が支配的であることが多い．近年の科学技術計算の大規模化，高速化の要請に応えるために，マルチコア，メニーコアなど最新の計算資源を有効に活用できる前処理，後処理のための行列分解アルゴリズムが求められている．行列分解に対するタイルアルゴリズムは，行列を小行列に分割して処理することで細粒度のタスクを多数生成できるので，近年の高並列環境向けアルゴリズムとして注目されている．

我々はこれまでに QR 分解のタイルアルゴリズムをマルチコアクラスタシステムに実装した [4]．この実装は，OpenMP によるタスク並列プログラミングモデル，動的タスクスケジューリングなどの特徴を持つ．平成 25 年度プログラム高度化共同研究支援事業によって，この実装のノード間通信における問題点が明らかになり，これを改良することができた．本報告では，ノード間通信の問題点とその改良法，改良の性能向上について報告する．

2 タイル QR 分解

QR 分解は数値線形代数の基本的かつ重要なアルゴリズムである．数値線形代数ライブラリ LAPACK の QR 分解はブロックアルゴリズムを採用している．ブロックアルゴリズムは，行列を縦ブロックに分割し，分解と後続行列更新を縦ブロック毎に繰り返すことで行列全体を分解する．後続行列更新に L3 BLAS 演算が使用できるので，L2 BLAS 演算を多用する逐次アルゴリズムと比較して高い性能を発揮することができる．

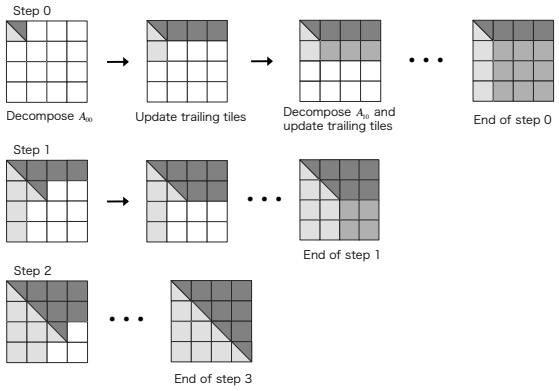


図 1: タイル QR 分解

しかし，ブロックアルゴリズムは基本的には縦ブロックの分解と後続行列更新を順に繰り返す逐次アルゴリズムであり，fork-join 型の並列計算モデルである．fork-join 型計算モデルでは原理上必ずストールするスレッドが発生するため，近年の高並列な計算資源を効率的に利用することはできない．

行列分解に対するタイルアルゴリズム [1] は対象とする行列を小行列（**タイル**）に分割し，分解，更新操作を 1 または 2 タイル毎に行う（図 1）．タイルサイズを適切に選択することで，実行プロセス/スレッド数に対して十分な数のタスクを生成することができるため，高並列な計算資源を有効に活用できる手法として近年注目されている．

$m \times n$ 行列 A の QR 分解は以下で定義される．

$$A = QR \tag{1}$$

ただし，$m \geq n$，Q は $m \times m$ 直交行列，R は $m \times n$ 上三角行列である．以下では，タイルサイズを $b \times b$（正方タイル）とし，行列 A は $p \times q$ 個のタイル $A_{ij}(i = 0, \ldots, p-1, j = 0, \ldots, q-1)$ から成るものとする．ただし，$p = \lceil m/b \rceil$，$q = \lceil n/b \rceil$ とする．

タイル QR 分解は以下の 4 つの基本計算（**カーネル**）で構成される．

- GEQRT：対角タイル $A_{kk}\,(k = 0, \ldots, q-1)$ のブロック化 QR 分解（ブロックサイズ s）を行ない，上三角行列 R_{kk} を生成する．変換行列は，単位下三角行列 V_{kk} と上三角行列 T_{kk} に格納される．
- TSQRT：上三角行列 $R_{kk}\,(k = 0, \ldots, p-1)$ とその下方にあるタイル $A_{ik}\,(0 \leq k < i < q)$ の組に対してブロック化 QR 分解を行い，R_{kk} を更新する．変換行列は正方行列 V_{ik}，上三角行列 T_{ik} に格納される．
- LARFB：GEQRT によって生成された変換行列 V_{kk}，T_{kk} をその右側タイル $A_{kj}\,(0 \leq k < j < q)$ に適用し，これを更新する．
- SSRFB：TSQRT によって生成された変換行列 V_{ik}，T_{ik} をその右側タイル A_{kj}，$A_{ij}\,(0 \leq k < i < p, 0 \leq k < j < q)$ に適用し，これを更新する．

GEQRT と TSQRT を**分解カーネル**，LARFB と SSRFB を**更新カーネル**と呼ぶ．

2.1 カーネル実行の依存性

我々はタイル QR 分解を，前項で示した 4 つのカーネル実行をタスクとしたタスク並列モデルとして実装する．タイルの上から下を “i 方向”，左から右を “j 方向”，図 1 のステップ数に相当する方向を “k 方向” として，タイル QR 分解のタスクには 3 方向の依存関係が存在する．

同一タイル列の分解または更新カーネルは上から順に実行しなければならない．この i 方向の逐次性を i **方向依存**と呼ぶ．

同一タイル行の更新カーネルは並列に実行することができるが，分解カーネルが変換行列を生成した後でなければ実行できない[1]．これを j **方向依存**と呼ぶ．

ステップ 0 を除いて，ステップ k のすべてのカーネルは，ステップ $k-1$ における同一タイル上のカーネルの終了後でなければ実行できない．これを k **方向依存**と呼ぶ．

2.2 動的スケジューリング

タイルアルゴリズムにはタスク実行の依存性を考慮したいくつかのスケジューリング方法が存在する [3, 2]．実行可能なタスクがスケジューリングの不備により実行待ちとなることを極力排除するために，我々はタイル QR 分解のタスクスケジューリング方式として**動的スケジューリング** [5] を採用した．動的スケジューリングでは，プロセス/スレッドが**タスクキュー**から実行可能タスクの情報を取り出し，これを実行する．動的スケジューリングによるタスクスケジューリングは，多くの静的スケジューリング方式よりも最適に近い可能性が高い．タスク間の依存性の有無は，**プログレステーブル**で管理する．

タスクの依存性の有無は，タイルの位置 $\{i, j, k\}$ に対応したプログレステーブルにその進捗を記録することで判定される．タスクを実行したプロセス/スレッドは，その完了によって解消される依存性をただちにプログレステーブルに記録し，すべての方向依存が解消された位置 $\{i, j, k\}$ をタスクキューに投入する．

3 クラスタシステム向け実装

ここではタイル QR 分解のクラスタシステム向け実装 [4]（以降，**SM2013** と参照する）について述べる．

実装の基本方針を以下に示す．

- タイルサイズをブロックサイズとした縦方向一次元ブロックサイクリックデータ分散
- 1 ノード 1 MPI プロセス
- OpenMP によるワークシェアリング
  - 1 ノード t スレッド

[1] タイル列を逐次的に分解する場合．並列に分解する手法も存在する．

– 通信スレッド (thread id (tid) = 0)

– 計算スレッド (tid = 1 ··· $t-1$)

∗ 動的タスクスケジューリング

クラスタシステムに対するタイル QR 分解におい縦方向一次元ブロックサイクリックデータ分散を用した場合，タイル列分解を担当するノードが生する変換行列を他のノードへ送信するブロードキスト通信が発生する．SM2013 では第 k タイル列分解を担当するノードは，i 方向すべての分解カネルが終了するまで送信元のままであり，他のノドからの送信は行われなかった．しかし，第 k タル列の分解が完了する前に第 k+1 タイル列の一の分解タスクは実行可能である．第 k タイル列の解がある程度進んだ後では，第 k+1 タイル列を当するノードは送信可能な変換行列データを持つ送信できずにいる状況にある．

ランク k のノードで第 k タイル列の分解を行っいるとき，一つの分解カーネルが終了すると，ランク k のノードから他のノードへ変換行列のブロードキャストが行われ，このノードを含めたすべてのノードでステップ k の更新カーネルが実行される．ここで，第 k+1 タイル列を担当するランク k+1 ノードはステップ k の更新カーネルを実行した後，k 方向依存がなくなったステップ k+1 の分解カーネルを同じタイル列で実行することが可能である．第 k タイル列の分解が完了するまでこのノードからは変換行列のブロードキャストが行われないので，これ以外のノードではステップ k のタスクのみが実行可能である．この連続的に発生するノード間の負荷不均衡が SM2013 の問題点である．

このように SM2013 では，ランク k の第 k タイル列の分解カーネルがすべて終了するまで，他のノードからの通信は発生しないが，負荷不均衡を解消するためには第 k タイル列分解の完了を待たずに，ステップ k+1 の分解カーネルが生成した変換行列のブロードキャストを開始すればよい．これにより第 k+1 タイル列担当以外のノードにおいてステップ k+1 の更新タスクを実行することが可能となる．この目的のため，通信スレッドが常に送信可能な変換行列の有無を確認し，送信可能な変換行列データを持つノードが随時これをブロードキャストするよう改良を加えた [6].

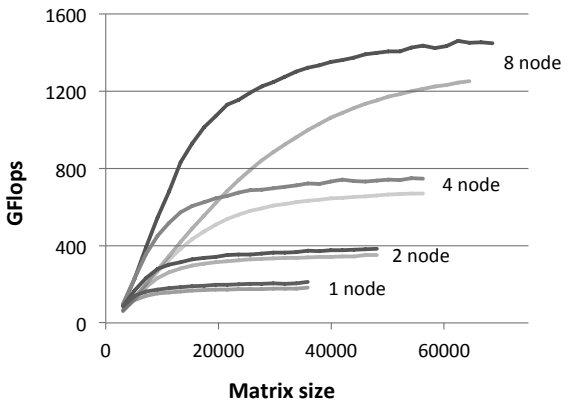


図 2: タイル QR 分解の性能

4 性能評価

この節では Cray XE6 上で行った性能評価の結果を示す．表 1 に使用したコンパイラ，ライブラリを示す．ノードあたりのスレッド数は $t = 32$ とし，プログラム実行時のオプションを `numactl -i all` とした．

表 1: 開発環境

コンパイラ	Cray C コンパイラ 8.2.2
BLAS/LAPACK	Cray LibSci 12.1.3
MPI	Cray MPICH 6.3.0

図 2 に通信改良前後の実装の性能を示す [6]．この図で，各ノードに 2 本づつ性能曲線が描かれているが，低速な方が SM2013，高速な方が今回作成した改良版である．ノード数の増加に伴って，SM2013 よりも特に小さい行列サイズで性能差が大きくなっていることが分かる．同図において 1 node の性能においても差が生じている理由は，改良版の実装のカーネルに PLASMA のカーネルを使用しているためである．

図 3 に行列サイズ $m = n = 15360$ と $m = n = 35840$ における台数効果（強スケール）を示す [6]．今回作成した実装は SM2013 と比べて，比較的小さい行列に対してもよくスケールしていることが分かる．

図 4 に 1 ノードあたりの行列サイズを $m = n = 24320$ としたときの台数効果（弱スケール）を示す

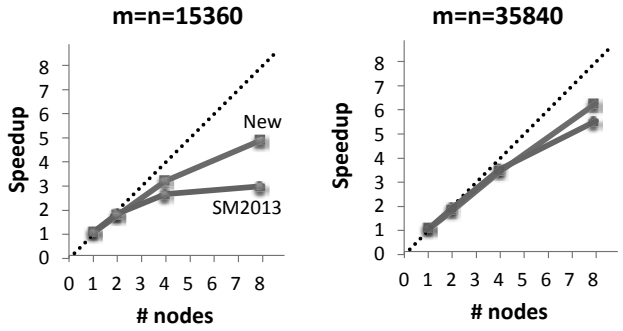


図 3: 台数効果（強スケール）

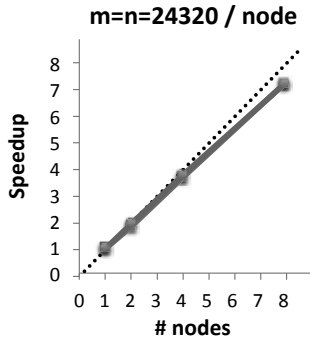


図 4: 台数効果（弱スケール）

[6]．こちらの尺度では SM2013 も新しい実装もよくスケールしている．

性能評価実験より，PLASMA カーネルを使用し，通信処理を改良した実装は 8 ノード実行時に SM2013 と比較して最大二倍の性能向上が見られる（$m = n =$ 12000 付近）ことが分かった．更に，台数効果を見ると行列サイズが小さい場合でも強スケールしていると言える．これは超並列マシン向けの実装として好ましい性質である．

5　おわりに

行列分解に対するタイルアルゴリズムは細粒度のタスクを多く生成することで，高い並列性を持つ最近のマルチコア, メニーコアアーキテクチャの性能を発揮できるアルゴリズムとして注目されている．以前の我々のクラスタシステム向けのタイル QR 分解には通信処理に起因して，負荷不均衡が生ずる問題があることを明らかにした．これを改良することで各ノードに対して十分な数のタスクが割り当てられ，比較的小さい規模の問題でも強スケールする実装が得られたことを実験で示した．

今後の課題として，まず，タイルサイズ，内部ブロック幅のパラメータチューニング方法を確立することが挙げられる．

参考文献

[1] Buttari, A., Langou, J., Kurzak, J. and Dongarra, J.: Parallel tiled QR factorization for multicore architectures, *Concurrency and Computation: Practice and Experience*, Vol. 20, No. 13, pp. 1573 – 1590 (2008).

[2] Kurzak, J., Ltaief, H., Dongarra, J. and Badia, R. M.: Scheduling Linear Algebra Operations on Multicore Processors, *Concurrency and Computation: Practice and Experience*, Vol. 21(1), pp. 15 – 44 (2009).

[3] Kurzak, J., Buttari, A. and Dongarra, J.: Solving systems of linear equations on the CELL processor using Cholesky factorization, Technical report, Trans. Parallel Distrib. Syst (2007).

[4] Suzuki, T. and Miyashita, H.: OpenMP/MPI implementation of tile QR factorization on T2K open supercomputer, *Proceedings of IEEE 7th International Symposium on Embedded Multicore/Many-core SoCs (MCSoC-13), Special Session on Auto-Tuning for Multicore and GPU (ATMG)* (2013).

[5] 鈴木智博：タイルアルゴリズムのための動的スケジューラの OpenMP 実装，情報処理学会研究報告ハイパフォーマンスコンピューティング (HPC), No. 2013-HPC-139(13), pp. 1 – 6 (2013).

[6] 鈴木智博：クラスタシステム向けタイル QR 分解のノード間通信の改良，(投稿中) (2014).

大規模GPU計算による3Dイメージベース FITの効率化

中畑 和之 *

*愛媛大学大学院理工学研究科

1 はじめに

社会インフラの構造部材の健全度を非破壊的に評価するために，超音波や電磁波を用いた検査が行われている．これらの波動は不可視であるため，シミュレーションによってその伝搬経路や特性を明らかにできれば検査の信頼性は格段に向上する．著者らは，これまでに有限積分法 [1](Finite Integration Technique: FIT) とイメージベース処理 [2] を組み合わせた超音波・電磁波のシミュレーション手法 [3] について研究を行ってきた．FIT は，支配方程式 (偏微分方程式) をセルと呼ばれる微小正方形 (2 次元)or 立方体 (3 次元) で領域積分した後，その積分方程式を離散化する方法である．時間方向には，陽的に解の更新を行う．イメージベース処理によって，2 次元解析ではベクター図やデジカメ等の画像から，3 次元解析では CAD，CT 写真，表面凹凸データ等のデジタル画像から数値モデルを作成できる．このプリプロセスの労力を減らすことによって，非破壊検査の多様な対象に対して現場で実時間シミュレーションを行うことが本シミュレーションの最終目的である．

近年，グラフィックボード上の集積回路である Graphics Processing Unit(GPU) を科学技術計算に応用する動きが年々加速している．GPU を画像処理専用のプロセッサとしてだけではなく，並列計算ユニットとして利用する試みは General purpose computing on GPU (GPGPU) と呼ばれている．京都大学においても GPU 計算機が整備され，サブシステム B では 1 ノードあたり 1 基の GPU(NVIDIA 社の Tesla M2090) がマウントされており，最大で 64 基の GPU が並列で利用できる．GPU1 基あたり 512 個の並列処理コアが搭載されているため，計算のアルゴリズムによっては CPU 並列計算よりも GPU 計算の方が，FLOPS が大きくなることが報告されている [4]．平成 24 年度のプログラム高度化支援事業では，2 次元イメージベース FIT を GPU で並列計算した場合の効率改善についてアドバイスを頂いた．この高度化支援事業では，コード中の分岐命令を有効に減らして GPU のカーネル計算自体を高速化すること，また，CPU 計算と GPU のカーネル計算部分を同時に進行させ，時間を要する部分を効果的に隠蔽することで，計算効率の大幅な改善がなされた．これを受けて H25 年度の高度化支援事業では，3 次元イメージベース FIT の GPU 計算のためのチューニングをして頂いたので，その報告を行う．3 次元弾性波動場では，応力 6 成分 τ_{ij} $(i,j=1\sim3)$ と粒子速度 3 成分 v_i $(i=1\sim3)$ を交互に更新する陽的アルゴリズムであり，2 次元波動場と比較して，1 ステップあたりのカーネルの実行量は多くなる．また，FIT はステンシル計算であり，2 次元版と同様に 3 次元版もメモリアクセス律速となる．高度化前に，CUDA Fortran を用いた著者自身のコードによって，ある程度の性能が出現できていたが，分析依頼の結果，以下のような性能向上の余地があることが分かった．

- Fortran 記述のコードに分岐処理が多用されている．その理由の一つとして，PML(波動吸収層) のために複雑なネスト構造になっているためである．
- 粒子速度 3 成分 (v_i) を数値積分して変位 3 成分 (u_i) を計算し，それを出力する必要がある．これは，デバイスーホスト間のメモリ転送後にホスト側で計算をしている．この処理に多くの時間を要している．
- スレッド数の調整やキャッシュメモリの使用など，大規模計算のために京都大学の GPU デバイス (Tesla M2090) に合わせたカーネルコードの最適化の必要がある．

ここでは，以上のような要因に対する改善策，およびその成果について報告する．

2 計算機の仕様と開発環境

2.1 検証に用いた GPU 計算機

本研究では，サブシステム B を利用した．GPU ボードとして NVIDIA 社の Tesla M2090 が 1 ノードあたり 1 基マウントされている．M2090 には，512 個の CUDA コアとよばれる最小単位の演算処理ユニットが内蔵されている．この CUDA コアが 32 個集まって 1 つのストリーミング・マルチプロセッサ (SM) を形成する．SM 内部には，高速メモリであるレジスタに加え，シェアードメモリ，L1 キャッシュ等のオンチップメモリ (計 64KB) がある．GPU のメモリ階層は，L2 キャッシュを経由して，グローバルメモリ，ローカルメモリ，テクスチャ，コンスタントメモリ等のデータを記憶する DRAM(GDDR5) に繋がっている．メモリバンド幅は 177GB/s(公称値) であり，サブシステム B 上の CPU(DDR3) のそれと比べると 1.7 倍程度大きい．従って，この大きいバンド幅を有効利用することが GPU 計算の高速化の 1 つのアプローチとなる．しかし，GPU1 基あたりの DRAM は 6GB であり，大規模な計算を行う場合には，複数の GPU を使用することになる．本研究では，最大 8 基の GPU を用いて検証を行った．

2.2 開発環境

2.2.1 CUDA Fortran

ここでは，プログラミング環境として CUDA Fortran を用いた．CUDA Fortran を提供するのは，現在のところ PGI 社のみであり，ここでは PGI Accelerator Fortran Ver.13.3 for Linux(CUDA Toolkit 5.0) を用いた．CUDA Fortran のプログラムは，CPU を動作させるホストコードと GPU を動作させるカーネルコードの 2 つのコードから成り立つ．ホスト用コードには，通常のプログラミングの他に CUDA のランタイム API，Built-in のベクトル型宣言，およびカーネル用コードを呼ぶ call 文等が含まれている．

2.2.2 CUDA の階層構造

CUDA はグリッド，ブロック，スレッドという並列計算の粒度単位を持っている．スレッドが計算を行う最小単位であり，そのスレッドの集まりがブロック，ブロックの集まりがグリッドとなる．ブロックとスレッドは 3 次元的な構造をしており，それぞれ最大数が決められている．1 ブロック内で使用できるスレッドの最大数は，x_1 方向が 1024 個，x_2 方向が 1024 個，x_3 方向が 64 個である．ただし，1 ブロック内の合計スレッド数は 1024 個までという制限がある．なお，ブロックやスレッドの ID は構造体で管理される．

プログラム実行時に，グリッド上にカーネルを生成し，スレッドそれぞれがそのカーネルを実行する．従って，GPU 上の各プロセッサは SIMD 型のデータ処理を実行することになるが，CUDA におけるカーネルは if 文等の分岐命令も実行できるため，厳密な SIMD 処理ではなく，NVIDIA 社は SIMT(Single Instruction Multiple Thread) と呼んで区別して表記している．

3 FIT への GPU 計算の導入

3.1 有限積分法 (FIT)

超音波版の 3 次元 FIT の詳細な定式化については，過去の論文 [3] で記載されているので，ここでは概略について述べる．3 次元直交座標系の位置ベクトルを $\boldsymbol{x}$=(x_1, x_2, x_3) とし，粒子速度を $\boldsymbol{v}(\boldsymbol{x}, t)$，応力を $\boldsymbol{\tau}(\boldsymbol{x}, t)$ とする．運動方程式を領域 V(その境界は S) で積分したものは

$$\int_V \rho(\boldsymbol{x}) \frac{\partial v_i}{\partial t}(\boldsymbol{x}, t) \mathrm{d}V = \int_S \tau_{ij}(\boldsymbol{x}, t) n_j(\boldsymbol{x}) \mathrm{d}S \quad (1)$$

と表される．ここで，ρ は密度，$\boldsymbol{n}$ は S 上の法線ベクトルである．また，応力と速度の構成関係を積分する．

$$\int_V \frac{\partial \tau_{kl}}{\partial t}(\boldsymbol{x}, t) \mathrm{d}V = \int_S c_{ijkl}(\boldsymbol{x}) v_i(\boldsymbol{x}, t) n_j(\boldsymbol{x}) \mathrm{d}S \quad (2)$$

上式で，$\boldsymbol{c}$ は弾性スティフネスで 4 階のテンソルであり，等方性の場合，以下のように書き表すことができる．

$$c_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu(\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk}) \quad (3)$$

上式で，λ と μ は Lamé 定数，δ はクロネッカーのデルタである．この定数を用いれば，縦波 (P 波) と横波 (S 波) の音速はそれぞれ，$c_P = \sqrt{(\lambda + 2\mu)/\rho}$, $c_S = \sqrt{\mu/\rho}$ と表される．いま，λ，μ，ρ は V 内で一定とし，V を微小な立方体 Δx^3(ボクセル) として，式 (2) と (3) を空間方向に離散化する．この V を積分セルという．離散化後の物理量の節点配置を図 1 に示す．非均質材料の場合，各セルに材料定数を設定

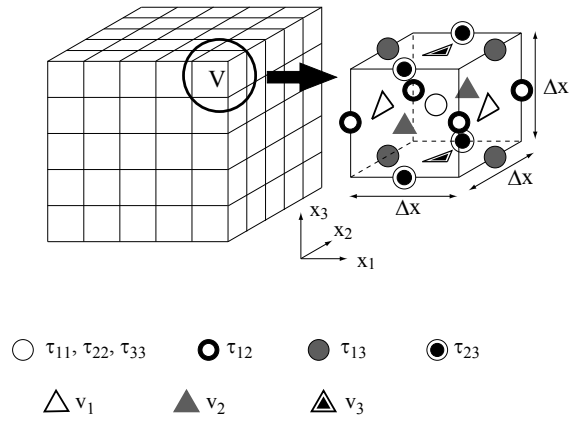


図 1: 3 次元 FIT における積分セル V と物理量の節点配置.

することで異種材料の界面を簡単に模擬できる．また，領域 V の大きさを 3 次元の画像データのボクセルサイズと等しくすれば，例えば CT 画像などを簡単に数値シミュレーションに取り込めるのが特徴である．

時間軸については，以下のような中心差分法で近似する．

$$\{\boldsymbol{\tau}\}^{z+\frac{1}{2}} = \{\boldsymbol{\tau}\}^{z-\frac{1}{2}} + \Delta t\{\dot{\boldsymbol{\tau}}\}^{z}, \tag{4}$$

$$\{\boldsymbol{v}\}^{z} = \{\boldsymbol{v}\}^{z-1} + \Delta t\{\dot{\boldsymbol{v}}\}^{z-\frac{1}{2}} \tag{5}$$

ここで，z は整数，Δt は時間ステップ幅である．上式は，速度 $\boldsymbol{v}$ と応力 $\boldsymbol{\tau}$ を交互に時間更新する陽解法であることを示す．式 (5) は，実際には $\tau_{11}(i,j,k)$ や $v_1(i,j,k)$ といった 3 次元配列を確保して，時間更新する毎にメモリを上書きしながら利用することになる．ここで，i は x_1 方向，j は x_2 方向，k は x_3 方向のインデックスを表す.

3.2 スレッドと配列

Tesla M2090 デバイスの場合，DRAM は 6GB 程度であり，これより大きな配列サイズを確保できない．従って，計算領域が大きい場合には，計算領域を分割して複数のデバイスに副領域を割り当てることが必要になる．なお，GPU はメモリバンド幅が非常に大きいので，1 つのデバイスで計算可能である場合は，副領域に分割することは避けた方がよい．図 2(a) に配列 τ_{11} の割り当ての例を示す．この例では，計算領域を 4 つの副領域に分割している．副領域を 1 つのグリッドとし，その中で 2 つのブロックに分かれているとする．グリッド内部のブロックの大きさは CUDA Fortran の構造体表記を用いて表される，この場合，gridDim%x=1, gridDim%y=1, gridDim%z=2 である．本研究では，1 つの配列の計算 (例えば式 (4) における $\tau_{11}(1,1,1)$ の更新) に 1 つのスレッドを割り当てる．図 2(a) の Grid 4 における 1 つのブロックでは，x_1 方向のスレッド数は 4，x_2 方向のスレッド数は 3，x_3 方向のスレッド数は 2 となる．これは，blockDim%x=4, blockDim%y=3, blockDim%z=2 と記述される．従って，1 つのグリッドでは，x_1 方向に (gridDim%x) × (blockDim%x) 個，x_2 方向に (gridDim%y) × (blockDim%y)，x_3 方向に (gridDim%z) × (blockDim%z) 個の粒度でスレッドが作られることになる.

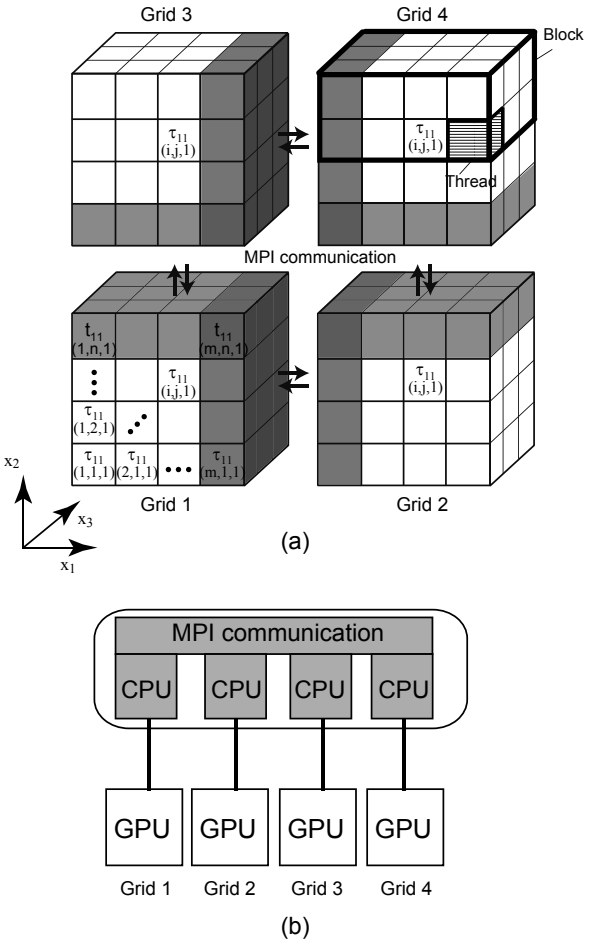


図 2: CUDA Fortran における計算の階層．各配列の更新計算はスレッド単位で行われる.

Fortran の特徴として，配列は左のインデックスが先に動く順番でメモリ上に並ぶ．従って，ブロック内でメモリが連続するのは，x_1 方向に並ぶ場合であることに注意する．これは，CUDA Fortran でも同様である．また，スレッド処理は 1 ワープ (32 スレッ

ド) 単位で行われるため，スレッド数は 32 の倍数が推奨されている．

3.3 マルチ GPU 計算

ここでは，図 2(b) に示すように，1 つの副領域の計算を 1 つのデバイスにタイアップする．すなわち，マルチプロセスに分割後，1 つの CPU プロセスが 1 つの副領域を担当する．その 1 つのプロセスに 1 つの GPU デバイスを割当てて，マルチ GPU 計算を行う．本研究では，3 次元 MPI 通信を用いて，プロセス間のデータ通信を行っている．3 次元 MPI 通信時には，隣接する副領域の間で袖部 (図 2(a) に示す色づけした帯部分) を交換する必要がある．本研究では，上述のように 1 プロセスと 1 デバイスをタイアップさせているため，まずホスト側のメモリにデバイス側からデータを転送し，次にホストメモリを介して交換したデータをデバイスメモリに送るという段階を踏んでいる．ここでは，できるだけ最小限の通信となるように，各配列の袖部のみを通信に用いている．なお，通信ライブラリとして MPI-1 と MPI-2 をサポートする Open MPI 1.6 を用いた．ネットワークインターコネクトは，InfiniBand 4x FDR Dual-rail である．

図 3 にプログラム高度化前のフローチャートを示す．基本的に，ホストとの通信が無く，カーネルだけを実行するのが理想であるが，FIT では MPI 通信やファイル出力等でデバイスからホストへメモリ転送をする必要がある．また，図 3 に示すように，v_i は数値積分によって変位 u_i に変換してファイル出力している．

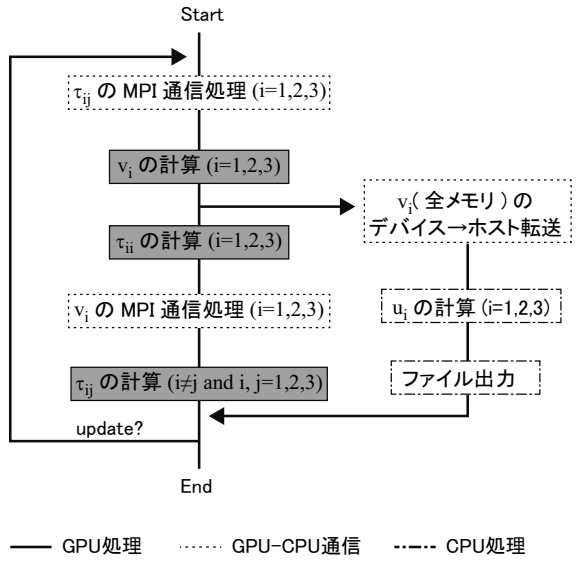


図 3: オリジナルプログラムのフローチャート．

4 高速化・効率化のための改良

4.1 分岐処理の見直し

高度化前のコードには分岐が多用されていた．その理由の一つとして，PML(波動吸収層) が実装されているためであり，結果として複雑なネスト構造になっていた．ここでは，応力 (τ_{ij}) と粒子速度 (v_i) の計算において，PML 処理を別のカーネルとして完全に分離した．カーネルコードを単純化することで，スレッドあたりのレジスタ使用量が減り，Occupancy(最大 warp 数) が向上し，約 30%の速度向上が得られた．

4.2 出力処理の最適化

FIT は陽解法であるので，定期的に計算結果を write する必要がある．非破壊検査への応用を考えて，粒子速度 (v_i) を数値積分して変位 (u_i) に換算し，それを出力する．高度化前のプログラム (図 3) では，粒子速度 v_i を計算した後に，GPU のデバイスメモリからホストメモリに転送し，ホスト側で変位 u_i の計算を行っていた．ホスト処理がカーネル処理よりも多くかかっており，1 ステップ終了時にホストとカーネル実行の同期をとる際にこの部分がネックとなり，全体として多くの時間を要していた．高度化後は，図 4 に示すようにカーネル処理として変位計算を追加した．また，変位の全メモリを転送するのではなく，出力用に間引いた変位データをホストに転送することによって，計算時間と転送時間の双方を短縮できた．また，デバイスーホスト間の通信を別 stream として非同期通信とした．以上のプロセスによって，出力処理とカーネル処理をオーバーラップすることができ，本申請の最も効果的な改善となった．しかし，この方法はデバイスメモリを多く必要とするため，京都大学のシステム B のようにメモリが豊富にある場合には有効であるが，個人 PC のような環境でメモリを節約したい場合には再考が必要である．

4.3 カーネルコードの見直し

デバイスーホスト間転送の性能向上のために，Pinned Memory の適用を検討した．この結果，Pinned Memory を使わない場合と比較して，転送時間は 50%程度減少した．また，カーネルメモリの配列に対してパディングを行った．Fermi 用に 128Byte に揃えるこ

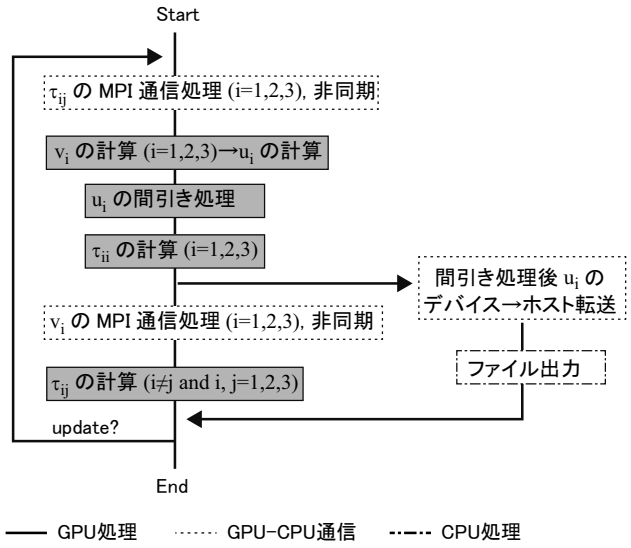


図 4: プログラム高度化後のフローチャート.

とで，3%程度の計算時間の減少が見られた．さらに，スレッドブロックサイズも検討を行った．スレッドブロックサイズを blockDim%x=64, blockDim%y=2, blockDim%z=1 とした場合が良い性能が得られることが分かった．スレッドブロックサイズを色々変化させた場合，20%程度の速度向上が見られた．また，L1 キャッシュを占有する CUDA オプションも効果的であった．このローカルメモリ分割仕方については，cudaDeviceSetCacheConfig() 関数を呼び出す事で切り替えることが出来る．

以上の検討結果を含め，オリジナルコードと高度化コードのスケーラビリティの比較を行ったものを図 5 に示す．MPI 分割によってノード数毎に最速となる領域分割が異なるが，それぞれの最速値を選択してスケーラビリティを示したのが図 5 である．全体の問題サイズを変えずに，Strong scaling で評価し

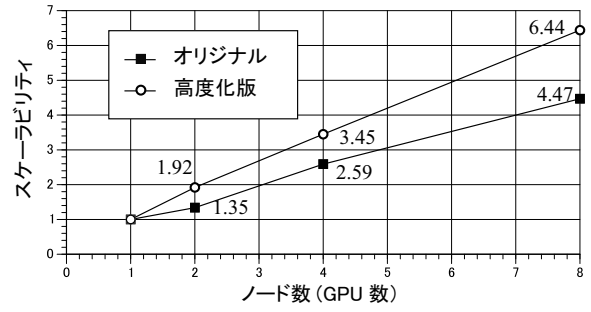


図 5: ノード数を増やした場合のスケーラビリティの比較 (オリジナル，高度化版).

ている．高度化版は，8GPU 使用時で 6.44 のスケーラビリティが得られている．

5 コンクリート中の超音波伝搬のモデル化と計測実験との比較

ここでは，3 次元 FIT によって計算した波形と計測実験で得られた波形の比較を行い，本シミュレーションの妥当性を示す．計測用の供試体として，骨材含有率 (V.F.) が 10，30，50%の供試体 (直径 100mm, 高さ 150mm の円柱形) をそれぞれ 3 体づつ，計 9 体作成した．供試体中の粗骨材の最大粒径は 10mm であり，水セメント比は 40%で統一した．円柱の上部に直径 20mm の円形探触子 (中心周波数 400kHz) を設置し，透過波を底面で受信した．一方，数値モデルも供試体と同サイズとし，同様の骨材粒径の分布とした．ただし，数値モデルでは骨材は大きさの異なる球形とし，ランダムに配置した．なお微小空隙はモデル化していない．ばらつきを考慮し，数値モデルは V.F.=10，30，50%のモデルをそれぞれ 10 個，計 30 個用意した．セメント硬化体は，c_P=3841m/s, ρ=2014kg/m^3 であり，これは供試体と同時に作成したセメント硬化体の小片から得たデータである．骨材は，c_P=5113m/s，ρ=2570kg/m^3 であり，実際の骨材から 1 辺 10mm の立方体を切り出して計測した．V.F.=50%のときの，FIT によるシミュレーション例を図 6 に示す．

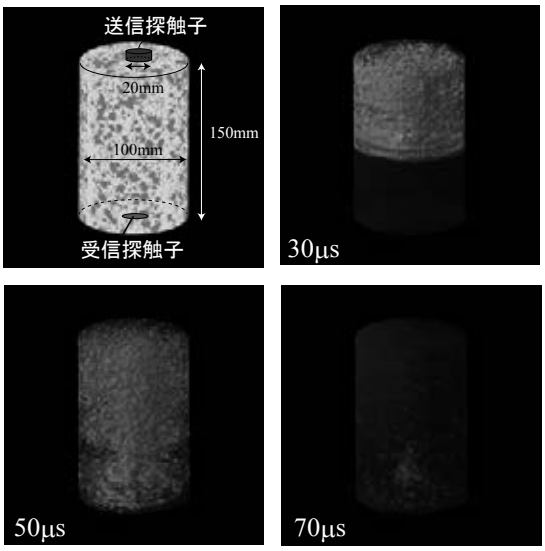


図 6: 骨材含有率が 50%のコンクリート中を伝搬する超音波の可視化 (3 GPU 並列).

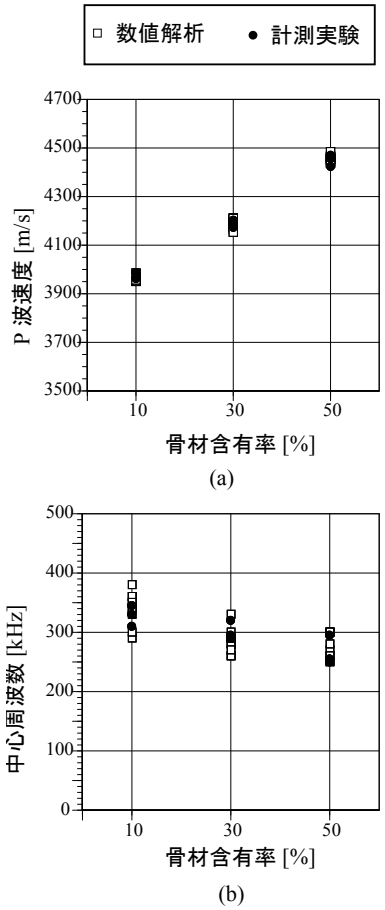


図 7: 骨材含有率が 50%のコンクリート中を伝搬する超音波の可視化 (3GPU 並列).

次に，V.F. を変化させた場合の音速と受信波の周波数の変化をプロットしたものを図 7 に示す．実験では，送信探触子を供試体の上部に，受信探触子を下部に設置し，超音波を波数 1 波で励起した．数値解析でも同様の送受信配置・励起周波数とし，表面の垂直応力を変動することで超音波を発生した．ボクセルサイズは Δx=0.02mm，Δt=20ns とし，5000 回の時間更新を行った．本計算は，研究室所有のワークステーション (NVIDIA Tesla C2075, 3GPU，PCI-Express 2.0 で接続) で行い，1 ケースあたりの計算時間は 20 分程であった．図 7(a) から，V.F. が増加すると縦波音速は増加することがわかる．これは，V.F. が増加すると速度の大きな粗骨材の割合が増えるためである．また，V.F. の増加に伴い，探触子で得られる受信波の周波数は減少している (図 7(b)) が，これは骨材の増加によって近接する骨材同士の波動散乱が増加するためである．波長の短い高周波成分が骨材で散乱され，結果として低周波成分が透過することになる．以上の結果から，計測実験とシミュレーションは超音波の音速，周波数の観点では良好な一致を示した．従って，3 次元 FIT は，骨材の含有具合による超音波伝搬の変化を定量的にシミュレーションできることを示した．

6 まとめ

今回，分岐処理の見直し，出力処理の最適化，カーネルコードの見直しについてご指導を頂いた．特に，出力処理の最適化が最も効果があり，ホスト処理（ファイル出力）を隠蔽することができた．1 ステップあたりの実行速度が改善されただけで無く，スケーラビリティも向上した．この成果を非破壊検査に応用し，コンクリート中の超音波伝搬シミュレーションが実時間で実施できることを示した．スパコンの高度化で得られた技術が PC ワークステーションに移転でき，性能が発揮されたことは非常に意義がある．

今後の課題は，XeonPhi のコプロセッサを利用した場合の FIT の高速化についても検討を行い，ヘテロ計算環境において性能が得られるように工夫していくことである．

謝辞： 有益なご助言を頂いた「スーパーコンピュータシステム共同研究企画委員会」の皆様，並びに，チューニングコードを提供して頂いたクレイ・ジャパン・インクの武田大輔氏に感謝を申し上げます．

参考文献

[1] P. Fellinger, R. Marklein, K.J. Langenberg and S. Klaholz, “Numerical modeling of elastic wave propagation and scattering with EFIT –elastodynamic finite integration technique”, *Wave Motion*, **21**, pp.47-66 (1995).

[2] K. Terada, T. Miura and N. Kikuchi, “Digital image-based modeling applied to the homogenization analysis of composite materials”, *Computational Mechanics*, **20**(4), pp.331-346 (1997).

[3] K. Nakahata, K. Terada, T. Kyoya, M. Tsukino and K. Ishii, Simulation of ultrasonic and electromagnetic wave propagation for nondestructive testing of concrete using image-based FIT, *Journal of Computational Science and Technology*, Vol.6, No.1, pp.28-37, 2012.

[4] T. Okamoto, H. Takenaka, T. Nakamura and T. Aoki, “Accelerating large-scale simulation of seismic wave propagation by multi-GPUs and three-dimensional domain decomposition”, *Earth, Planets and Space*, **62**, pp.939-942 (2010).

動的／静的水～土骨格連成有限変形解析コードの高度化

野田利弘

名古屋大学　減災連携研究センター

1　平成 25 年度事業の概要

平成 24 年度（下半期）に続き、平成 25 年度も京都大学学術情報メディアセンターのプログラム高度化支援事業に採択いただいた。平成 25 年度事業の報告とともに、計算事例を紹介する。

平作 24 年度は、筆者らのオリジナルの解析コード（GEOASIA® [1], [2]）において、OpenMP と MPI によるハイブリッド並列化計算を可能にするため、それまでの PARDISO に代わり、直接法ライブラリ MUMPS 及び反復法ライブラリ PETSc を適用した。なお、GEOASIA は、地盤の静的・動的を問わないで砂から粘土までの広範な土からなる地盤の変形・破壊挙動を調べることを目的としている。より詳細に興味がある方は「広報」[3]をご覧いただきたい。

平成 25 年度事業では、このハイブリッド並列化した解析コードの高速化を行うため、スレッド並列化を有効化したスレッド並列検討版 PETSc を GEOASIA に適用し、計測および性能検証を行った。具体的には、GEOASIA 実行時の係数行列の実行時スレッド数、CSR 形式データ対角要素からの零要素の排除 Affinity 設定等の影響に加え、PETSc のフラット MPI による収束傾向について検証した。以下では、この中から一部を報告する。

2　スレッド並列化を適用した PETSc の GEOASIA への適用

各種検証を行うため、テストデータとして、三次元解析（3D ケース）と有限要素数が多い二次元解析（2DLarge ケース）を使用した。各テストデータ係数行列の詳細を表 1に、計測に使用した PETSc の実行時オプションを表 2 に示す。

表 1　テストデータ(係数行列)ケース一覧

テストデータ名	行列次元	非零要素として確保された要素数	左記のうち、実際の値が厳密に 0 である要素数
2DL	144,241 x 144,241	2,709,993	19,810
3D	55,380 x 55,380	3,612,682	47,748

表 2　組み込みテスト時の PETSc オプション

オプション	値
収束判定閾値	1.0e-8
最大繰返し回数	10000 回
前処理	ASM(ILU)
ILU fill-in Level	ILU(1)
ソルバ	TFQMR
置換関数	GEMV,TRSV,DOT

本報では、3D ケースの検討結果の一部として、図 1 と図 2 にそれぞれ Solver 部分と全体の PETSc の 1 プロセス 1 スレッド(1p1t)に対する速度向上比を示す。また、図中の「default」はスレッド並列化を行っていない PETSc を利用した計測結果である。「gemv+dot」はスレッド並列版 PETSc を組み込んだ GEOASIA の計測結果である。「csr 削除」が記載されている結果は、スレッド並列検討版 PETSc 及び CSR 形式からの零要素の排除の両方を適用した結果である。これは、GEOASIA において、PETSc に渡されるデータが CSR 形式で定義されているが、本 CSR 形式のデータの中には零要素も多く含まれており、本零要素に起因して「zero divide」等の不安定な動作が発生していると推察されたため、CSR 形式のデータの零要素の内、特に影響が大きいと推察される対角要素内の零要素を排除した。また、「前

処理再計算」が記載されている結果は、CSR 形式の零要素の排除を行うことで「SAME_NONZERO_PATTERN」が使用できなくなることから CSR 形式の零要素の排除時の計測では「SAME_PRECONDITIONER」への切り替えを行っていたが、収束性能が一定の閾値を超えて悪化した場合には再度前処理を計算しなおす修正を加えた。これは今回、「SAME_PRECONDITIONER」使用時には STEP 数の増加に伴い solver の収束性能の悪化が見込まれためである。

スレッド並列検討版 PETSc を組み込んだ結果は、Solver 部分の速度向上はやや見られるものの、全体実行時間としては微小な差異にとどまった。一方で、CSR 形式のデータの対角要素からの零要素の排除時および前処理再計算時には、Solver 部分で約 30%、全体でも約 10%の速度向上が見られた。なお、いずれの場合も精度は「default」での実行と比較しほぼ差異は見られなかった。2DLarge に関する計測結果は省略するが、このケースでは、16 スレッド実行時に反って速度低下が見られたため、係数行列のみ 8 スレッドで実行した結果、その速度低下は回避できることが分かった。

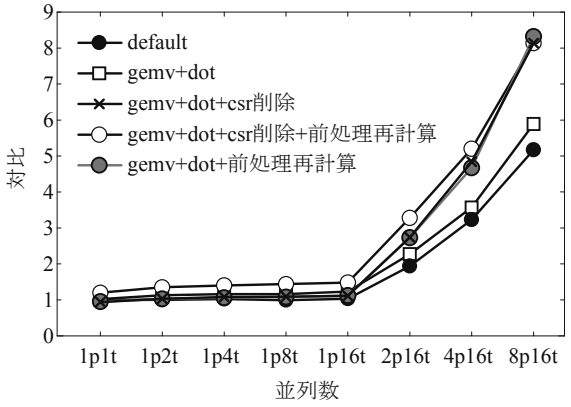


図 1　修正を加えた Solver 部分の実行時間の変化

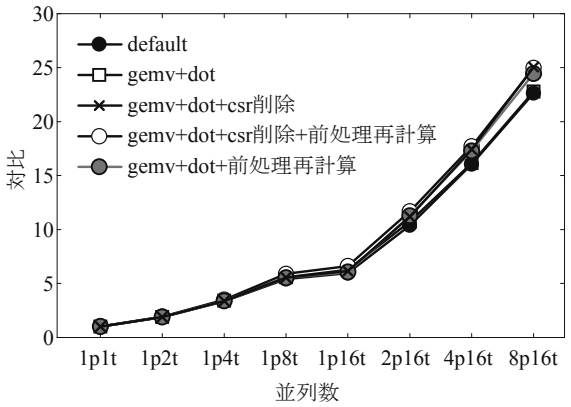


図 2 修正を加えた全体の実行時間の変化

次に、今後の更なる並列数の増加及びスレッド並列化のための検討材料として、図 3 はフラット MPI による PETSc の単体実行を行った結果を示す。1 プロセス~16 プロセスまでは 1 ノードで実施し、その他は 2 ノード、4 ノード、8 ノードにて実施した。この結果として、並列数の増加に伴い反復回数も増加した。しかしながら、計算時間においても並列数の増加に伴い短縮が見られた。このことから、反復回数・計算時間という点では PETSc における更なるスレッド並列化も可能であると考えられる。

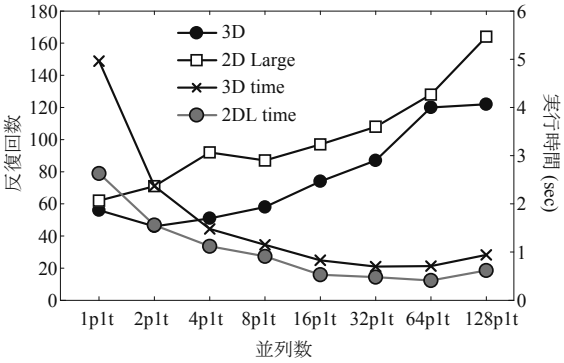


図 3　MPI 並列数増加に伴う反復回数と実行時間

3　計算事例の紹介 - 横ずれ断層上の表層地盤におけるフラワー構造を伴うリーデルせん断帯群生成シミュレーション [4]

横ずれ断層上では、被覆層を介して地表面にリーデルせん断帯と呼ばれる雁行状のせん断面群が表れることが知られている。図 4 [5]の模式図に示すように、右横ずれ断層の場合、地表面には左雁行配列のせん断面群が表れる。1997 年兵庫県南部地震においても、野島江崎地区の棚田において、リーデルせん断帯が確認されている（写真 1 [6]）。Naylor et al. [7] に代表されるように、リーデルせん断帯の再現を目的とした模型実験は多数行われている。上田[8]は、X 線 CT スキャンにより、横ずれ断層に伴い、基盤の断層面から被覆層内を花弁状に拡がるフラワー構造が地表面に達しリーデルせん断帯が形成される様子を可視化している。また、澤田・上田[9]は、構成式にモール・クーロン破壊規準に基づく弾完全塑性体を用いた大変形解析を実施し、上記模型実験にて確認したリーデルせん断等の形成過程をシミュレートしている。

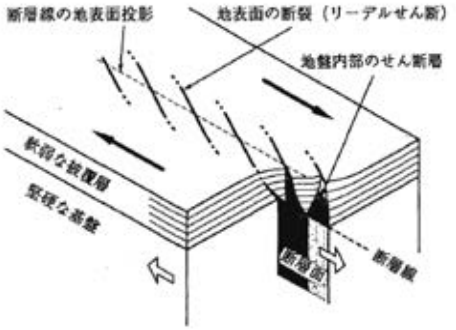


図４　リーデルせん断帯の模式図[5]

写真 1 兵庫県南部地震で観察されたリーデルせん断帯[6]

(1) 計算条件

以下では、今回実施した計算事例として、横ずれ断層運動に伴い表層地盤で観測されるフラワー構造を伴うリーデルせん断帯の３次元生成シミュレーション結果を紹介する。なお、間隙水と連成させない一相系で計算を実施した。

計算スケール等は上田[8]による模型実験を参考に決定した。図５は有限要素メッシュと境界条件を示す。y-z 側面は摩擦のない壁を想定して、x 方向の変位を固定した。底面は z 方向の変位を固定し、断層部分を挟んで一様な y 方向の強制変位（定率速度 10^{-6}m/s）を逆向き（右横ずれ断層を想定した向き）に与えた。x-z 側面は周期境界を設けて、同様なパターンが y 方向に連続する状況を設定した。また、初期不整に伴う分岐モードの発現とひずみの局所化の誘発[10]を狙って、下端の断層部分の土要素のみ材料定数の一部（正規圧密線の切片）に違い（結果的に初期過圧密比の違い）を与えることで材料の初期不整を与えた。

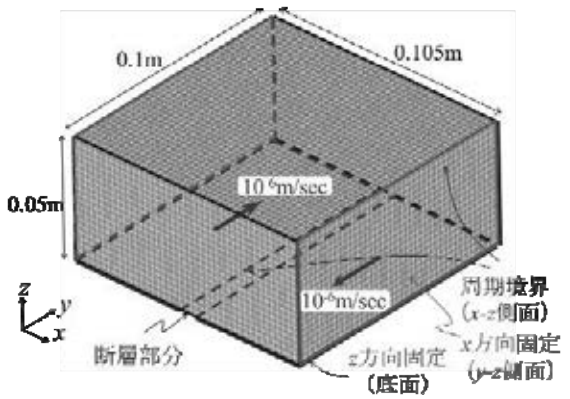


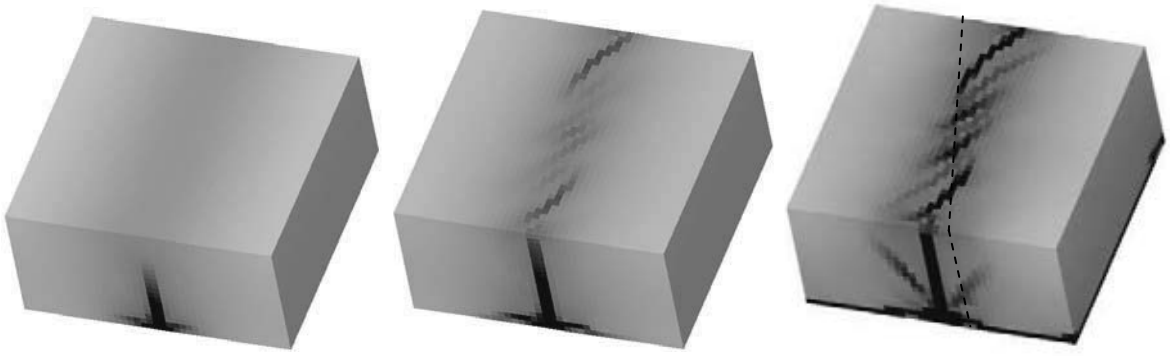
図５ 有限要素メッシュと境界条件

(2) 計算結果

図６に地盤の解析領域表面におけるせん断ひずみ分布を示す。底面の相対変位の増大に伴って、地表面にリーデルせん断帯群が表れることが分かる。図７ に地盤内部における等せん断ひずみ分布を示す。地盤内部に断層上を斜めに横切りながらねじれて発達してゆくひずみの局所化領域を確認できる。図７よりこの花弁状のひずみの局所化領域が地表面に達することで、左雁行配列のリーデルせん断帯群が形成されていることが分かる。また，例えば図６(c)および図７(c)に破線で示した断面から見えれば、複数のフラワー構造が地表面に向かって発達していることを確認できる（図省略）。以上で示した通り、GEOASIA により、ひずみの局所化を伴う進行性破壊現象としてフラワー構造を伴うリーデルせん断帯の生成を数値的に再現できる。なお、底面に初期不整のない場合は、図を省略するが、底面にずれが生じても平板状のせん断帯が形成されるのみで、フラワー構造やリーデルせん断帯は現れなかった。このことから、リーデルせん断帯の発生要件として、分岐を誘発するための初期不整を挙げることができる。

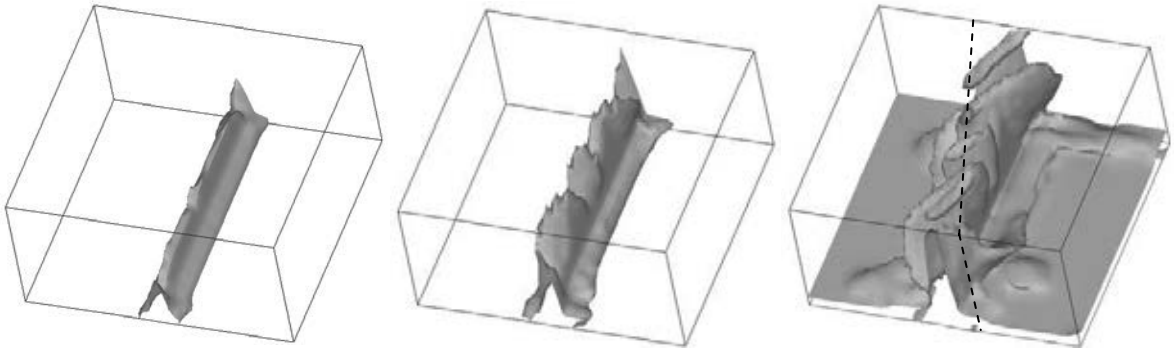
4　おわりに

平成 25 年度では、スレッド並列検討版 PETSc を GEOASIA に適用して性能計測を行った。結果は、Solver 部分の時間で若干の性能向上が見られたが、全体の実行時間としては優位な性能向上を見ることはできなかった。一方で、係数行列実行スレッド数の調整や PETSc の実行時オプション等の検討結果から、今後のアプローチとしては PETSc の前処理部分の更なるスレッド並列化や係数行列等を含む領域分割などが考えられることが分かった。幸いなこ



(a) 相対変位 0.09mm　(b) 相対変位 0.12mm　(c) 相対変位 0.45mm

0 0.007　図 6　せん断ひずみ分布（初期不整あり）



(a) 相対変位 0.09mm　(b) 相対変位 0.12mm　(c) 相対変位 0.45mm

図 7　等せん断ひずみ(ε_s=0.005)面（初期不整あり）

とに、H26 年度の支援事業にも採択いただき、領域分割法の適用などが継続的に進んでいる。

謝辞　京都大学学術情報メディアセンターの中島浩教授および牛島省教授ならびにクレイ・ジャパン・インクの方々には、今も非常にご尽力いただいている。ここに謝意を表します。

5　参考文献

[1] Asaoka, A. and Noda, T. (2007): All soils all states all round geo-analysis integration, Int. Workshop on Constitutive Modelling - Development, Implementation, Evaluation, and Application, Hong Kong, China, 11-27.

[2] Noda, T, Asaoka, A. and Nakano, M.(2008): Soil-water coupled finite deformation analysis based on a rate-type equation of motion incorporating the SYS Cam-clay model, S&F, 48(.6), 771-790.

[3] 京都大学学術情報メディアセンター全国共同研究利用版広報，Vol.12, No.2, 33-35.

[4] 川合裕太，野田利弘，山田正太郎，浅岡顕，澤田義博(2014)，横ずれ断層に伴うフラワー構造を伴うリーデルせん断帯生成の数値シミュレーション，土木学会第 69 年次学術講演会，Ⅲ-371，741-742.

[5] 谷和夫, 上田圭一, 阿部信太郎, 仲田洋文, 林泰幸 (1997): 野島地震断層で観察された未固結な表層地盤の変形構造, 土木学会論文集, 568/Ⅲ-39, 21-39.

[6] 中田高, 岡田篤正編 (1999): 野島断層【写真と解説】兵庫県南部地震の地震断層, 東京大学出版.

[7] Naylor, M. A., Mandl, G. and Sijpesteijn, C. H. K. (1986): Fault geometries in basement-induced wrench faulting under different initial stress states, J. Struct. Geol., 8, 737-752.

[8] 上田圭一 (2003): 横ずれ断層系の発達過程ならびに変位地形の形成過程，電力中央研究所研究報告, U03021.

[9] 澤田昌孝，上田圭一 (2009): 横ずれ断層の進展に伴う地盤の破壊領域評価のための数値シミュレーション，電力中央研究所研究報告, N08028.

[10] Asaoka, A. and Noda, T. (1995): Imperfection-sensitive bifurcation of Cam-clay under plane strain compression with undrained boundaries, S&F, 35(1), 83-100.

コンクリート材料の物質拡散・非線形力学を連成した経年劣化シミュレータの高度化

浅井 光輝

九州大学大学院工学研究院　社会基盤部門

コンクリートの経年劣化は，中性化・アルカリシリカ反応・塩害等さまざまな要因がある．その多くは力学的要因と化学的要因が複雑に連成しており，劣化メカニズムを詳細に把握したうえで合理的な対策することは困難である．そこで，コンクリート内部での物質の浸透・拡散現象と同時に，亀裂・はく離損傷等の現象を再現可能なシミュレータを開発してきた． 同シミュレータを用いれば，コンクリートアリカリシリカ反応時に観測される自由表面での複雑なひび割れが定性的に再現することができる．材料劣化の促進実験結果との比較検証することで計算の信頼性を確保するためにも大規模計算が不可欠となり，京都大学情報メディアセンターが企画しているプログラム高度化共同研究支援事業を利用させていただき，その高速化・効率化を図った．

1　緒言

コンクリート構造物の経年劣化の要因は，中性化・アルカリシリカ反応(ASR)・塩害等のさまざまな現象が考えられる．その多くは力学的要因と化学的要因が複雑に連成しており，劣化メカニズムを詳細に把握したうえで合理的な対策を行うことは非常に困難である．コンクリート内部での物質の浸透・拡散現象は，材料内部の膨張あるいは劣化の起点となり，微視的な亀裂，材料界面でのはく離等の損傷を招く．また逆に，微視的な損傷を受けた領域は，拡散現象において高速に物質が拡散する経路となり，物質が局所的な領域に拡散しやすくなる．以上のように，拡散問題と力学の損傷問題は両者が双方向に連成する現象であり，材料劣化の微視的なメカニズムを解明していくには，連成現象までを再現可能なシミュレータが必要となる．

そこで本研究では，まずコンクリート材料のさまざまな劣化要因の中から ASR を取りあげ，連成解析を実施することで，ASR 現象の特徴的な損傷である亀甲状のひび割れパターン（Fig.1 を参照）を再現することを目標とし，コンクリート材料をモルタルと骨材，そして両者の界面の 3 相に分けたメゾコンクリート材料モデルを用いた数値解析モデルを開発し，その並列計算効率の向上を図った．

写真 1　ASR による損傷事例

2　拡散膨張･不連続面進展の連成解析

ASR とはコンクリート材料の母材であるモルタル内部のアルカリイオンと介在物である骨材が反応し，骨材表面にゲル状の物質が生成されることで内部膨張が生じ，この内部膨張に伴い材料界面のはく離，あるいは母材内部に微視的なひび割れが進展していくものと考えられている．この一連の流れをモデル化するものとした．

コンクリート中へのイオンの浸透には拡散，移

流，吸着などの現象が考えられるが，本研究においては拡散現象のみを考慮するものとした．基本的には，非定常拡散解析と内在物膨張に伴う不連続面進展解析とも有限要素法で解析を行い，この両者を組み合わせることで，拡散膨張を考慮した不連続面進展解析手法の構築を行うことにした．このコンクリートのアルカリ骨材反応を想定した連成解析の手順を図 1 に示す．

解析手順としては，まず非定常拡散問題を解き拡散物質の空間分布を予測する．次に，濃度の空間分布の結果を基に膨張力を決定する．この際，膨張力によって引張り応力が基準値を超え不連続面と判定される領域（要素）は，拡散係数の高い仮想空隙領域に置換し，再び拡散問題を解く．以上の手順を繰り返すことで，浸透・拡散に伴う不連続面進展解析を実施した[1)]．以下ではその手法を概説する．

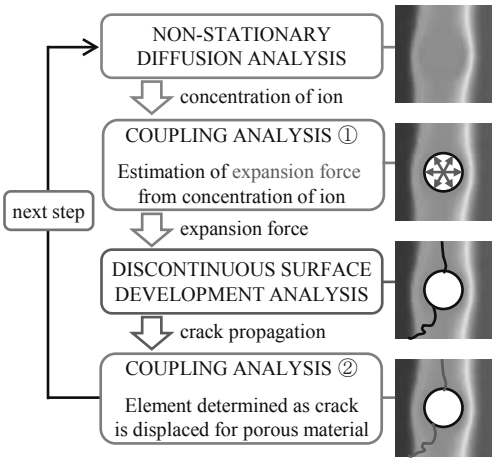


図1　拡散膨張・不連続面進展の連成解析フロー

2.1 ASR 反応に伴う骨材膨張のモデル化

アルカリシリカ反応では，コンクリート内のアルカリ反応性骨材と微細空隙中のアルカリ溶液が化学反応することにより生成するアルカリシリカゲルが吸水，膨張し骨材周辺からひび割れが進展する．本研究ではアルカリシリカゲルの吸水による体積膨張を骨材の体積膨張として表現した．具体的には，骨材要素のみに次式に示すように濃度に比例した膨張ひずみを与えることにした．

$$\begin{aligned}\left\{\varepsilon^c\right\} &= \left\{\varepsilon_x^c \quad \varepsilon_y^c \quad \varepsilon_z^c \quad \gamma_{xy}^c \quad \gamma_{yz}^c \quad \gamma_{zx}^c\right\}^T \\ &= \left\{\alpha c \quad \alpha c \quad \alpha c \quad 0 \quad 0 \quad 0\right\}^T\end{aligned} \tag{1}$$

ここで，α は膨張係数，cはイオン濃度を表す．この膨張ひずみをもとに膨張力を評価し，力学計算においてはこの力を駆動力とした非線形問題を解く．

2.2 損傷モデルによる不連続面の表現

損傷モデルでは連続体損傷力学を導入し，損傷変数 D を用い，要素剛性（材料割線係数）を段階的に減少させる．損傷変数は $0 \leqq D \leqq 1$ であり，D=0 のときには健全な状態を示し，D=1 は最終的な破壊状態を表す．損傷に伴う微小空隙の発達は材料の剛性低下を引き起こすため本研究では弾性係数の低下によって損傷状態を表わす．

$$D = \frac{E_0 - E}{E_0} \tag{2}$$

E_0 は非損傷状態の弾性係数，E は損傷後の弾性係数である．$\boldsymbol{C}_e$ を一般的な等方弾性テンソルとすると，損傷を考慮した材料構成則は以下のように規定される．

$$\boldsymbol{\sigma} = (1 - D)\mathbf{C}^e : \boldsymbol{\varepsilon}^e \tag{3}$$

コンクリートは引張り応力に弱い材料であり，圧縮強度に比べて引張強度は，約 1/10 程度である．そこで，圧縮と引張強度の相対的な影響度を考慮した相当ひずみを用いて破壊損傷を定義することが望ましい．本研究では Peerlings ら [2)]が提案した圧縮域と引張域の強度比を考慮した相当ひずみ(4)の両者を用いることにした．

$$\begin{aligned}\varepsilon_{eq} &= \frac{m-1}{2m(1-2\nu)} I_1 \\ &\quad + \frac{1}{2m}\sqrt{\left(\frac{m-1}{1-2\nu} I_1\right)^2 + \frac{2m}{(1+\nu)^2} J_2}\end{aligned} \tag{4}$$

ここで，I_1 はひずみの 1 次不変量，J_2 は偏差ひずみの 2 次不変量であり，m はコンクリートの圧縮強度と引張強度の比，ν はコンクリートのポアソン比を示す．

この相当ひずみを用い，損傷の進展則に従うものとした．

$$D = 1 - \frac{\kappa_0}{\kappa}\left[1 - A + A\exp\left\{-B(\kappa - \kappa_0)\right\}\right] \tag{5}$$

ここで A，B は損傷の進展を表すパラメータ，κ は材料が過去に受けた最大の相当ひずみである．κがκ_0 となると損傷が開始するものとし，対応して損傷変数 D は非ゼロの値を持ち始め，最終的には完全な損傷状態を示す 1.0 に漸近する．

2.3 損傷係数による拡散係数の変動

先に説明した損傷モデルを用いれば，微視的なクラックが成長することで段階的に剛性を低下していく過程を表現できる．その損傷係数を参照しながら，拡散係数も空隙に相当した大きな値c_Bへと段階的に変化させることにした．具体的には，損傷係数を用いて以下の式により損傷後の拡散係数c_Dを与えるものとした．

$$c_D = c + D(c_B - c) \tag{6}$$

3　メモリ分散化並列計算手法

解析コードは，材料の断面画像から解析モデルを作成することを想定し，ボクセルと呼ばれる構造格子状に規則配置した要素を用いた有限要素解析 [3]を前提としている．この特徴を活かし，単純に各計算ノードが担当する要素数を均等に分割するように，要素ベースで３次元領域分割を行った(図２参照)．なお，計算過程においては，最大で，面で接し合う領域間（６面），線で接し合う領域間（各稜線の 12 線），点で接し合う領域間（各頂点の８点）での 26 領域間での通信が必要となる（図３参照）．共同研究制度の支援を受けてチューニングした結果として，並列化効率を図４に示し，またメモリ分散化したことにより各計算ノードが負担するメモリ使用量が軽減した結果を図５に示す．なお，計測には 50×50×50 要素分割した立方体形状以のモデルを用いた．ここでは最大 64 プロセス×4 スレッド並列のハイブリッド並列まで行ったが，良好な並列化効率・省メモリ対策が実施できていることが確認できる．

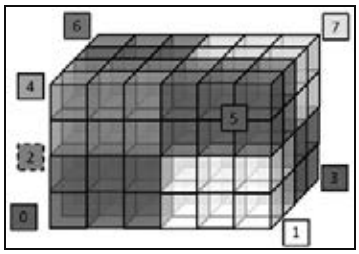


図２　領域分割概念図

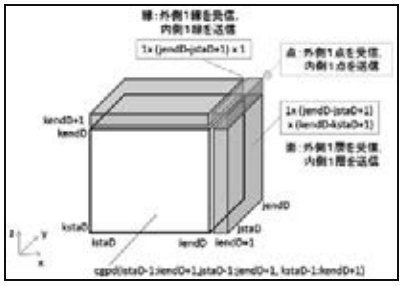


図３　近接領域とのデータ通信

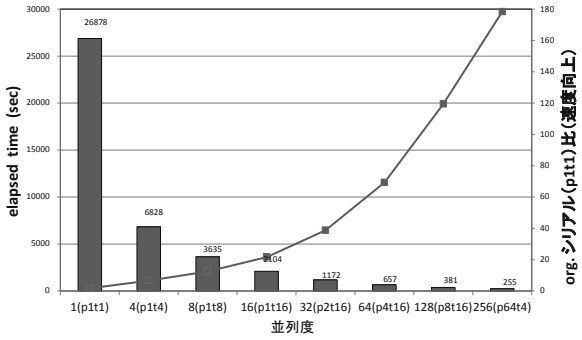


図４　並列化効率

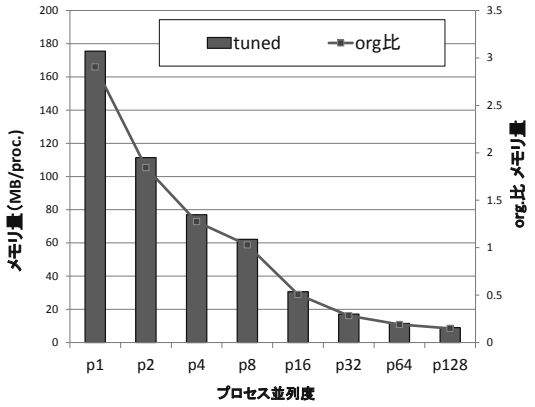
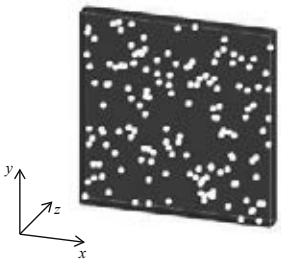


図５ 省メモリ化

4　解析事例

上記で説明した解析手法を用い，3 次元コンクリートモデルを解析した結果を報告する．図６に示す解析モデルは 500×500×50㎜ のコンクリート板を想定したものであり，内部に完全球形の骨材をランダムに配置したモデルである．なお，要素間隔を 1㎜ のボクセル要素によりモデルを作成したため，総要素数は 1250 万である．



サイズ	： x=500(mm), y=500(mm), z=50(mm)
1ボクセル	： 1mm^3（12,500,000要素）
ランダムに球形骨材を配置	
拡散係数	
モルタル （青）	： K=1.0×10^{-8} cm^2/sec
骨材 （白）	： K=1.0×10^{-12}cm^2/sec

図 6　3D コンクリート解析モデル

解析に用いた物性値は表-1 にまとめて示す．その他，損傷モデルのパラメータは A=0.99, B=100, m=10 とした．境界条件としては，z=0 の xy 平面以外，全面の変位を拘束し，非拘束面である xy 平面(z=0)から拡散物質を浸透させるものとした．

表-1　材料パラメーター一覧

	弾性定数 (GPa)	ポアソン比	κ_0(損傷開始相当ひずみ)
モルタル	20	0.25	2.0×10^{-4}
界面	20	0.25	5.0×10^{-5}
骨材	60	0.25	—

解析結果として，表面上の損傷分布と内部の損傷分布を図 7 に示す．同図から，変位を拘束せず，アルカリイオンを与えた表面付近の骨材から膨張が始まり，500 日後程度からその周辺で界面割れが生じ，1250 日後程度からは界面割れが母材であるモルタルへと進展し，骨材表面での割れを連結するようにひび割れが進展し始め，2000 日後には，実際にも観測されるような亀甲状の割れが定性的に表現することができた．

5　結語

これまでに，非定常拡散解析と内在物膨張に伴う不連続面進展解析とを組み合わせた連成解析手法を開発し，コンクリートのアルカリ骨材反応の損傷メカニズム解明を試みてきた．今回は京都大学学術メディアセンターの企画しているプログラム高度化共同研究支援事業によりその高度化・高速化を実施した． 高度化の成果として，数千万から億単位の要素を用いた解析が可能となり，実験と一対一のスケールにて比較することが現実的になってきた．

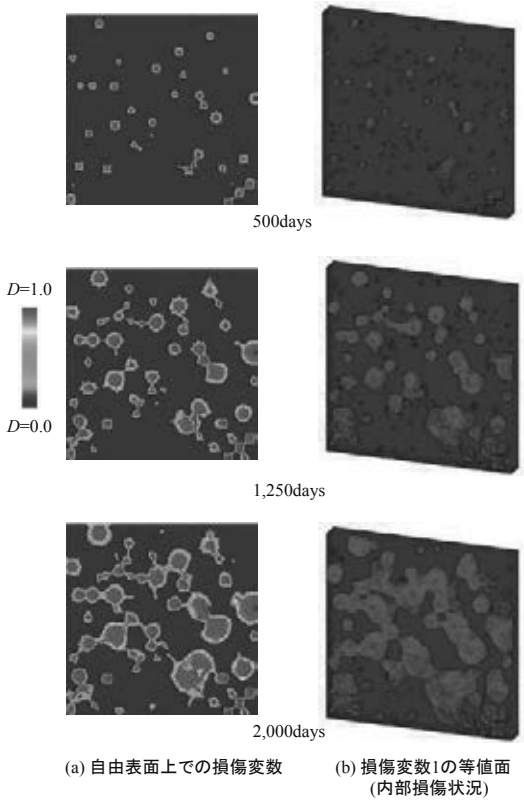


図 7　解析結果

参考文献

1) 渡邉茜，浅井光輝，損傷モデルの概念を導入したボクセル FEM による物質拡散とひび割れ進展問題の連成解析，*計算工学論文集*,No.20130007，2013.
2) R. H. J. Peerlings, R. de Borst, W. A. M. Brekelmans and M. G. D. Geers, Gradient-enhanced damage modeling of concrete fracture, Mech. Cohes.-Frict. Mater., 3, pp.323-342, 1998.
3) 浅井光輝, 山岸道弘, 寺田賢二郎, 永井学志, 非局所型有限要素法の開発とその破壊挙動解析への適用，土木学会論文集，No.759, pp.233-245, 2004.

Synchronized Molecular Dynamics 法による高分子潤滑の解析

安田修悟

兵庫県立大学大学院シミュレーション学研究科

本稿では、最近、著者らが開発した Synchronized Molecular Dynamics（SMD）法 [S.Yasuda and R. Yamamoto, Phys. Rev. X **4**, 041011 (2014)] についての解説とそれを用いて 2 平板間の高分子液体の潤滑挙動を解析した結果について紹介する。SMD 法では、多数の分子動力学（MD）セルを流体の要素に割り当て、各流体要素おける局所的な物理量を MD 法によって計算する。しかし、ある一定の時間間隔で、システム全体のマクロ物理量（エネルギーや運動量）の輸送方程式が満たされるように、隣接する MD セル間において運動量とエネルギーの交換を行う。これによって、任意の分子モデルを基に、巨視的な移動現象を解析することができるようになる。SMD 法による高分子潤滑の解析では、潤滑中に生じる粘性発熱が高分子液体のレオロジー特性と内部構造にどのように影響するのかを詳細に調べている。せん断流・発熱・内部構造が互いに複雑に関係し合った結果として、せん断流による変形が支配的となる領域と温度上昇による影響が支配的となる領域との間に、興味深い転移挙動があることが明らかにされた。

1　はじめに

分子動力学（MD）シミュレーションは、複雑液体の種々の輸送係数や内部構造などを解析するのに有効な手段である。MD 法では、液体を構成する分子をモデル化し、多数の分子の運動を計算する。[1,2] 液体の物性値を求める場合には、通常、1 辺が数ナノメートル程度の周期境界条件を考えた微小な MD セルが用いられる。また、最近では、ナノ流路（流路幅が数 10 ナノメートル程度）における複雑液体の流動を直接 MD 法によってシミュレーションする研究も行われている。[3]

MD 法は、任意の分子モデルに基づいて解析することが可能であるので、新奇機能性材料の開発やナノ流体技術における種々の解析に役立つと期待されている。しかしながら、マイクロスケールを超えるシステムに対しては、計算量が膨大になるために、計算機性能が発達した今日においても、まだまだ現実的には MD 法を適用することは不可能である。複雑な分子構造を持つ、高分子や生体分子からなる複雑液体の巨視的な流動を、任意の分子モデルを用いてシミュレーションする方法の開発は、工学的にも理学的にも重要な課題である。このような課題に対して、分子シミュレーションと連続体シミュレーションを双方向に接続してシミュレーションをするマルチスケールモデリングが一つの有効な解決策として期待されている。[4]

本稿では、最近、著者らが開発した Synchronized Molecular Dynamics（SMD）法[5]を用いて高分子潤滑の問題を解析した結果について紹介する。

以下では、具体的な問題設定を行い、その具体的な解法として SMD 法の解説を行う。高分子潤滑の解析では、2 平板間のせん断流れによる粘性発熱が、高分子液体のレオロジー特性や内部構造にどのように影響するのかについて、文献[5]で得られた結果を中心に紹介する。最後に、全体のまとめと SMD 法の適用における技術的、概念的な注意点について言及する。

2　問題設定

2 平板間にある高分子液体を考える。平板は一様一定の温度 T_0 に保たれており、時刻 $t = 0$ で片側の平板にせん断応力 σ_0 を加える。図 1 参照。

高分子液体は、Kremer-Grest の高分子モデル[6]を用いた、ビーズを 10 個だけ繋げた短い高分子鎖から構成される。

平板間の距離が十分に大きいと、平板の動きによって駆動されるせん断流による粘性発熱によって、平板間に不均一な温度分布（温度上昇）が生じる。本稿では、平板間の距離が、10 マイクロメートル程度の場合を考える。このとき、一般的な潤滑油に対して、歪速度が 10^5 s^{-1} 程度の高速せん断流において、油の粘度が有意に変化するだけの温度上昇が生じることが、後に定義する無次元パラメータによって予測される。

発熱を伴う場合の、2 平板間の高分子液体に対するマクロな輸送方程式は、以下のように書くことができる。以下では、xを平板の移動する方向、yを平板に垂直な方向とする。

$$\rho_0 \frac{\partial \dot{\gamma}}{\partial t} = \frac{\partial^2 \sigma_{xy}}{\partial y^2} \tag{1}$$

$$\rho_0 \frac{\partial e}{\partial y} = \sigma_{xy}\dot{\gamma} - \lambda \frac{\partial^2 T}{\partial y^2} \tag{2}$$

ここで、$\dot{\gamma}$は歪速度、σ_{xy}はせん断応力、Tは温度、λは熱伝導係数、eは単位質量当たりの内部エネルギー、ρ_0は密度（一様一定）である。また上式では、高分子液体のマクロ物理量は平板に沿う方向には一様一定であると仮定している。

粘性発熱による温度上昇は、式(2)の右辺第 1 項と第 2 項の大きさの比によって決定される。従って、粘性発熱によるレオロジー特性の変化を調べる場合には、次の Nahme-Griffith 数 *Na* が重要な無次元パラメータとなる。

$$Na = \frac{\sigma_0 \dot{\Gamma} H^2}{\lambda |\partial \log(\eta_0)/\partial T_0|^{-1}} \tag{3}$$

ここで、η_0は温度T_0での高分子液体の粘度を表しており、式(3)の分母に含まれる$\Delta T = |\partial \log(\eta_0)/\partial T_0|^{-1}$は高分子液体の粘性が十分に変化するのに必要な温度上昇を表す。即ち、$Na>1$ ではせん断流による粘性発熱によって、高分子液体の粘度が有意に変化することが予測される。$\dot{\Gamma}$は、両平板の速度差を平板間距離で除したせん断速度を表す。即ち、$\dot{\Gamma} = v_w/H$、v_wは上平板の速度。

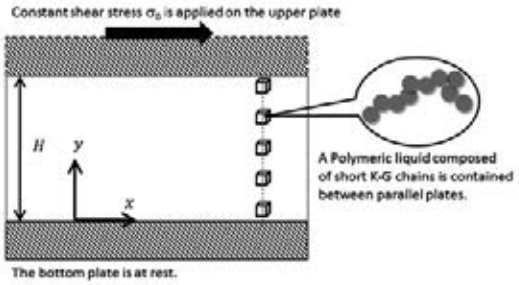


図１：問題設定

3 Synchronized MD 法

本節では、SMD 法について解説する。図 2 参照。前節で述べた問題では、平板に垂直な方向に温度、応力、流速といった巨視的な物理量が空間的に変化する。SMD 法では、高分子液体の局所的な温度と応力を MD シミュレーションによって計算する。即ち、複数の MD セルを、物理量を求めるべき各流体要素に割当て、各流体要素における密度と歪速度を拘束条件として、非平衡 MD シミュレーション（SLLOD 法[2]）によって各流体要素の物理量を計算する。当然、非平衡 MD シミュレーションでは、高分子の内部構造も同時に求められる。このとき、計算領域に疎らに割当てられた MD セルは、ある一定の時間間隔で、運動量とエネルギーの輸送方程式が局所的に満たされるように、隣接する MD セル間で、運動量とエネルギーの交換を行う。即ち、一定の時間間隔Δtで、各セルの局所的な歪速度と温度が運動量とエネルギー方程式を満たすように補正される。

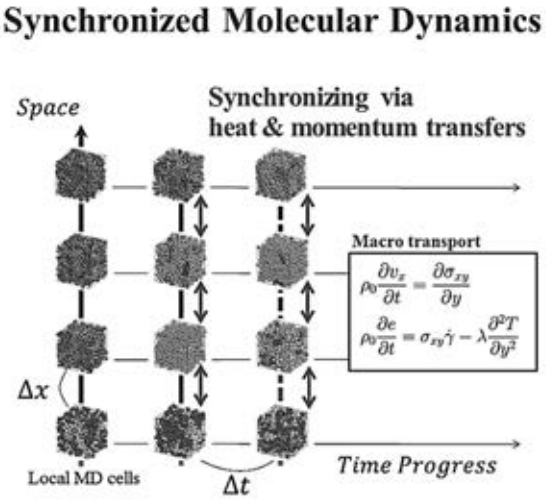


図２：SMD 法の概略図。

各 MD セルによって計算される応力Σと温度Θは次のように巨視的な輸送方程式と関係し、各 MD セルの歪速度と温度補正（運動エネルギーの補正）が計算される。式(1)を時間ステップΔtだけ積分する。

$$\dot{\gamma}^n(y) = \dot{\gamma}^{n-1}(y) + \frac{1}{\rho_0}\frac{\partial^2}{\partial y^2}\int_{(n-1)\Delta t}^{n\Delta t} \Sigma_{xy}\,[\tau; \dot{\gamma}^{n-1}(y)]d\tau \quad (4)$$

ここで、式 (1) 右辺のせん断応力σ_{xy}は MD セルによって$\dot{\gamma}^{n-1}$の歪速度の下に非平衡 MD によって計算される応力Σ_{xy}と置換えられる。

隣接する MD セルの温度からフーリエ則にしたがって各 MD セルに流れ込む熱量を計算し、それを用いて各 MD セルにおける運動エネルギーの補正を行う。即ち、一定の時間間隔Δtで、各 MD セルの運動エネルギーを次のように計算されるδK だけ補正する。

$$\delta K(y) = -\frac{\lambda}{\rho_0}\frac{\partial^2}{\partial y^2}\int_{(n-1)\Delta t}^{n\Delta t} \Theta[\tau\,; \dot{\gamma}^{n-1}(y)]d\tau \quad (5)$$

式(5)は、エネルギー輸送方程式(2)の右辺第二項の時間積分に対応しており、Θ[τ]は各 MD セルで計算される瞬時の温度を表している。一方、式(2)の右辺第 1 項は、非平衡 MD シミュレーションによって自動的に計算される。(このエネルギー補正の方法は、式 (2)で記述される内部エネルギーの変化に対して、熱流量によってもたらされる部分は、分子の内部構造によって決まるポテンシャルエネルギーよりも分子の運動エネルギーの変化により早く伝わることを前提としている。)

以上、式(4)、(5)によって計算される歪速度とエネルギー補正を一定の時間間隔で各 MD セルに課すことで、全ての MD セルは、系の運動量とエネルギーの輸送方程式を満たすように同期される。

4　計算結果

以下では、全ての物理量を Lennard-Jones ポテンシャルの特徴量を用いて無次元化された無次元量で表す。また本稿では、定常問題のみを扱うため、結果で示される物理量は全て、定常状態における長時間の時間平均が取られた値が示されている。物理量は平板間で空間的に変化するが、本稿では、局所的な物理量の空間平均についての結果のみを紹介する。

平板温度を$T_0 = 0.2$、平板間距離を$H = 2500$、熱伝導率をλ = 150とし、平板に加えるせん断応力σ_0を様々に変化させて計算を実行した。図 3 に、平板で測定されるレオロジー特性と高分子液体の温度変化の関係を示す。図3(a)には温度一定の MD シミュレーションによって計算される粘性係数が比較のため表示されている。温度一定の MD は、熱伝導係数が無限に大きい場合に対応する。Naが小さいところでは、両者に大きな差が現れないが、$Na > 1$で、シアシニングの様子が大きく異なることが分かる。また、図 3(b)から、$Na > 1$で歪速度$\dot{\Gamma}$の増加とともに急激な温度上昇が起こることが分かる。$Na > 1$では、粘性発熱による温度上昇がレオロジー特性に大きく影響していることが考えられる。

図 4 に$\dot{\Gamma}$と高分子液体の内部構造の関係を示す。$Q_{\alpha\beta}$は高分子の配向テンソルのαβ成分を表す。$Na < 1$では、$\dot{\Gamma}$が大きくなると伴にせん断方向に引伸ばされ、特定の方向に配向することが分かる。しかし、$Na > 1$では、この傾向が完全に逆転する。これは、$Na > 1$では、高分子液体の温度上昇の影響が支配的になるためと考えられる。即ち、$Na > 1$では、$\dot{\Gamma}$が大きくなると伴に温度が上昇し、高分子鎖の構造は分子の熱運動によってランダムで一様な構造を回復する。

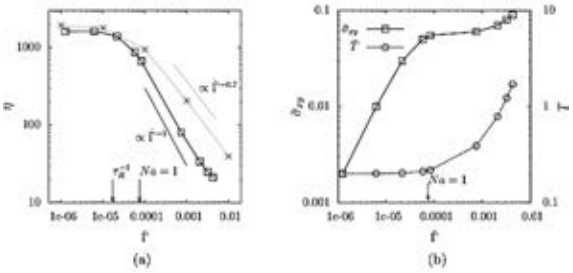


図３（文献[5]からの引用）：粘性係数と温度の変化。図の下矢印は、*Na* 数が 1 になる歪速度を示す。また、図(a)の*印は、温度一定の MD で計算した結果、□は SMD で計算した結果を示す。

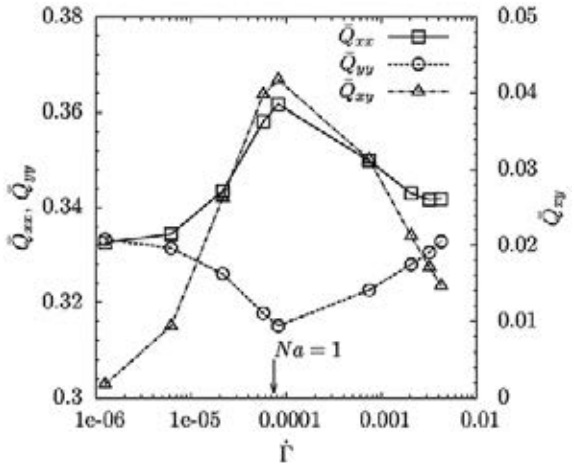


図４（文献[5]からの引用）：内部構造（配向テンソル）の歪速度に対する変化。

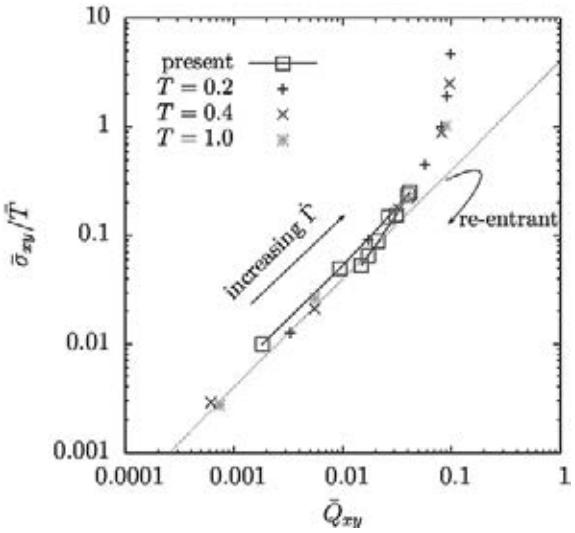


図５（文献[5]からの引用）：応力光学特性。+, x, *印は、温度一定の MD で計算した結果を表す。SMD 計算の結果では、歪速度$\dot{\Gamma}$を大きくしたときに線形の応力光学特性への re-entrant な転移があることが分かる。

図 3 で示される温度上昇と、図 4 で示される構造変化の結果をもとに、図 5 において、高分子液体の応力光学特性を表示する。ここで、温度一定の MD シミュレーションの結果を比較のために示す。温度一定の MD シミュレーションは、系の温度が T=0.2、0.4、1.0 に対して実行し、それぞれの温度に対する結果についてプロットしている。温度一定の MD では、温度が異なる場合にも同一の応力光学特性曲線を与えることが分かる。この曲線は、Q_{xy}の値が小さい場合には線形、Q_{xy}の値が大きい場合には非線形であり、歪速度の増加とともに単調に右肩上がりの曲線を描く。

一方、SMD 法の結果は、従来の温度一定の MD シミュレーションとは、大きく異なる振舞いを示す。$Na < 1$の場合には、温度一定の MD と同様に歪速度の増加とともに、右肩上がりに線形の関係を示す。しかし、$Na > 1$の領域では、歪速度の増加とともに温度上昇の影響が支配的になり、縦軸の値（σ_{xy}/T）も横軸の値（Q_{xy}）も共に減少し、右肩下がりに線形の関係を示す。即ち、$Na = 1$を境に、せん断変形が支配的な領域と温度上昇が支配的な領域の転移が起こるが、応力光学特性においては線形関係がいずれの領域においても成立する。

5　まとめ

本稿では、最近、著者らが開発した SMD 法を用いて高分子液体の粘性発熱を伴う潤滑問題の解析に適用した例を紹介した。潤滑において発熱は、基本的に重要な問題である。しかし、強いせん断流下において、高分子液体の内部構造が、せん断による変形と発熱による温度上昇にどのように影響されるのかはこれまでの解析方法では取扱うことが難しい問題であった。Na 数の定義式(3)からも分かるように、粘性発熱による温度上昇の効果は、システムサイズが大きいときに高分子液体のレオロジー特性と内部構造に強い影響を残す。本研究で解析した問題は、マクロな移動現象とミクロな分子運動が強く相互作用する問題の代表的な一例と言える。本研究では、このようなミクロとマクロが強く相互作用する物理的な難問題に対して、SMD が有効であることを具体的に示すことができた。実際に、図 5 で得られた結果は、一見、複雑に見える複雑なレオロジーと温度と内部構造の関係の背後に、単純な線形の応力光学特性が成り立つことを示すものであり、物理的に興味深い結果である。

また SMD 法では、複数の MD セルがある一定の時間間隔において互いに独立に計算が実行され

るため、各 MD セルに一つの CPU ソケットを割当てるようなプロセス並列コードを組むことによって、ほぼ理想的な並列化効率を得ることができる。さらに、各 MD セルでは、スレッド並列を実行することによって、効果的なハイブリット並列計算も可能である。このように SMD 法は最近の大規模並列計算機との親和性が非常に高いという大きな利点がある。

本稿では、文献[5]の研究成果をもとに、SMD 法の計算手法と解析結果についてここまで紹介してきた。文献[5]では、これらの結果に加えて、SMD 法の適用における技術的な注意点や適用範囲を調べるための具体的な数値試験も行われている。本稿では、その詳細な結果は省略するが、本稿の最後に、数値試験によって明らかにされた SMD 法の技術的・概念的な注意点を３つ述べる。

一つ目は、時間間隔Δtの設定に関しての条件である。時間間隔Δtは、MD セル間の距離Δxに対して粘性力が伝達される時間よりも小さくとる必要がある。即ち、$\Delta t < \Delta x^2/2\eta$が満たされる必要がある。これは、運動量輸送方程式を陽解法で数値的に解く際の安定性問題と関係しているが、SMD 法では粘性係数ηが自律的に変化するため、この条件が満たされないと、数値計算が破たんし実行不可能になるのではなく、むしろ上の条件を満たすように局所的な粘性係数が “偽の値” を持つことになる。二つ目は、SMD 法で生じるノイズが巨視的な温度分布に与える影響についての問題である。せん断速度が小さい場合、せん断応力の値に比してノイズの割合が大きくなる。このとき、定常状態において、流速分布は、長時間の平均を取ることによってノイズの影響を取り除くことができるが、温度分布は、長時間の時間平均ではノイズを取り除くことができない。このことは、せん断が小さい場合の解析解との比較によって明らかにされた。従って、せん断が小さい場合に精確に温度場を求めるためには、MD セルでの粒子数を多くすることが必要となる。三つ目は、MD セルの大きさl_{MD}とセル間Δxについての点である。SMD 法では、各 MD セル内では流体の慣性や対流の無い一様な状況を考える。このため、MD セルのスケールでの局所レイノルズ数は、十分に小さくする必要がある。一方で、粘性散逸を含む流体の巨視的な振舞いは、メッシュ間隔Δxの精度で巨視的な輸送方程式によって数値計算される。このため、メッシュ間隔Δxは、流体の巨視的な振舞いを十分精確に数値計算できるように小さくとらなくてはいけない。

SMD 法の適用にあたっては幾つかの注意点も必要であるが、SMD 法は、ミクロとマクロが強く相互作用する複雑流体の問題に対して有効な新しいシミュレーション法であり、また、今後ますます普及することが予想される大規模並列計算機との親和性が高いという重要な特徴を持っている。これらの利点を活かし、SMD 法が今後の複雑流体のシミュレーションにおける一つの重要な方法となるように、更に開発を続けていきたいと考えている。

文献

[1] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford University Press, New York, 1989).

[2] D. J. Evans and G. Morris, *Statistical Mechanics of Nonequilibrium Liquids* (Cambridge University Press, New York, 2008).

[3] B. H. Kim, A. Beskok, and T. Cagin, “Viscous Heating in Nanoscale Shear Driven Liquid Flows”, Microfluid. Nanofluid. **9**, 31 (2010).

[4] G. Karniadakis, A. Beskok, and N. Aluru, *Microflows and Nanoflows Fundamentals and Simulation* (Springer, New York, 2000).

[5] S. Yasuda and R. Yamamoto, “Synchronized Molecular-Dynamics Simulation via Macroscopic Heat and Momentum Transfer: An appriction to Polymer Lubrication”, Phys. Rev. X **4**, 041011 (2014).

[6] K. Kremer and G. S. Grest, “Dynamics of Engangled Linear Polymer Melts: A mMolecular-Dynamics Simulation”, J. Chem. Phys. **92**, 5057 (1990).

Cray Reveal によるスレッド並列化

武田 大輔

クレイ・ジャパン・インク

1 はじめに

近年は CPU のマルチコア化が進み、スレッド並列処理は HPC の分野に限らず広く一般的にその重要性を増しています。HPC 分野でも、MPI 等で既に分散メモリ並列化が実装されているアプリケーションに対して、その通信オーバーヘッドを低減するなどの目的でスレッド並列化を加えた、いわゆる「ハイブリッド」並列処理が一般的になってきていますし、GPU やインテル社の MIC アーキテクチャ等の多数のコア（演算装置）を備えた「アクセラレータ」と呼ばれるハードウェアにおいては、その性能を引き出すためにスレッド並列化を行うことが必須要件となっています。

本稿でご紹介する Cray Reveal（以下、単に Reveal とします）は、クレイ社製コンパイラがソースコードを解析して得られた情報と、Cray Pat と呼ばれる性能解析ツールによるプロファイル情報を組み合わせ、OpenMP プログラミングを支援するための開発ツールです。前述のスレッド並列化の潮流の中で開発された比較的新しいツールでまだ発展途上のものですが、OpenMP に不慣れな方の補助としてお使いいただけるだけでなく、既に OpenMP をよく使用されている方が時間のかかる作業を自動化するツールとしても活用していただけるものと考えられます。以下では、スーパーコンピュータシステム・サブシステム D を想定して、Reveal を使うための準備から実際に OpenMP 指示行を挿入するまでの一連の手順を解説します。なお、Reveal はサブシステム A および E でも利用可能です。

2 準備

本節では、Reveal を使用するために必要な準備についてご説明します。対象となるソースプログラムが既に存在する（ここでは myprog.[c|cc|f] とします）ものと仮定します。

2.1 ソースコードのバックアップ

Reveal は自動でソースコードに変更を加えて OpenMP 指示行を追加するソフトウェアですので、失敗したときにいつでも元に戻せるよう、ソースコードのバックアップを取っておくことをお勧めします。

2.2 X Window System 環境の準備

Reveal は GUI によって操作を行うアプリケーションですので、X Window System が必要です。サブシステム D で X Window System を使用する方法については [1] をご参照ください。

2.3 modulefile のロード

サブシステム D では module というコマンドで modulefile と呼ばれる設定ファイルをロードすることによって、使用するコンパイラやライブラリへのパスや、一部のコマンドでは自動で指定されるオプション等を制御するようになっています。したがって、まずはこれから使用するソフトウェアに合わせて module コマンドの設定を行うことが重要です。

Reveal は、クレイ社製コンパイラ（Cray Compiling Environment、以下「CCE」とします）を使用しますので、PrgEnv-cray モジュールがロー

ドされていることをご確認ください。通常の設定では、サブシステム D にログインした時点でデフォルトでロードされています。

```
$ module list
Currently Loaded Modulefiles:
(中略)
22) PrgEnv-cray/5.2.25
```

もし、PrgEnv- で始まる名前のモジュールが PrgEnv-cray でない場合は、module swap コマンドで切り替えることが可能です。例えば、いま PrgEnv-intel (インテルコンパイラ用のプログラミング環境) がロードされている状態であるとすると、

```
$ module swap PrgEnv-intel PrgEnv-cray
```

とすることで、CCE のプログラミング環境に移行することができます。さらに、Cray Pat と Reveal を使用するために、perftools モジュールをロードします。

```
$ module load perftools
```

なお、本稿で前提とするバージョンは、CCE 8.3.0、perftools 6.2.0、Reveal 2.0.0 であり、本稿執筆時点ではサブシステム D のデフォルトとなっていますが、他のサブシステムや将来のソフトウェア更新によってバージョンが異なることもありますので、ご利用の際にはご確認ください。

2.4 Cray Pat によるプロファイル

プロファイルとは、プログラムの各部分でどのくらい実行時間がかかっているかを調べたもので、通常は関数単位や行単位で計測した実行時間や IP サンプリング数で表されます。Reveal を使用するために必須ではありませんが、最適化や並列化を行うにあたって「どこに時間がかかっているか?」というのは重要な情報ですので、通常はこれらの作業を始める前にプロファイルを取るという手順が行われます。

サブシステム D では、Cray Pat というプロファイラが用意されていますので、これを使用します。Cray Pat でプロファイルを取るには再ビルド (コンパイル) が必要ですが、特に Reveal 用のプロファイルを取る際にはコンパイル時に -h profile_generate というオプションを指定してください。なお、-h の後のスペースはあってもなくてもかまいませんので、-hprofile_generate と書いても同じ意味になります。他の –h オプションについても同様です。

```
$ cc -h profile_generate myprog.c  # C
$ CC -h profile_generate myprog.cc # C++
$ ftn -h profile_generate myprog.f  # Fortran
```

このオプションは、通常の Cray Pat の使用方法では不要ですが、これを指定することによって「ループ単位の」実行時間が記録されるようになり、この情報に基づいてどのループを並列処理するかを決定します (3.5 節でご説明します)。但し、このオプションによってループの最適化が変化し、通常実行時の実行時間とは差異が生じる可能性がありますので、あくまで目安程度の値とお考えください。

なお、分割コンパイルをされている場合は、生成された *.o ファイル (オブジェクトファイル) を必ずとっておくようにしてください。これらは以下でご説明する pat_build / pat_report の際に必要となります。

次に、pat_build というコマンドで実行形式プログラムを「プロファイル取得用」に変換します。その際、-w オプションを指定してください。これは「トレーシング (tracing) を行う」という意味ですが、詳細は [4] [5] をご参照ください。

```
$ pat_build -w myprog
```

すると、myprog+pat という別の実行形式ファイルが生成されます。これを実行することでプロファイルを取得します。実行は通常通りの方法で行ってください。システム D での実行に不慣れな方は [2] をご参照ください。

もし、入力データの大きさ等を調整できる場合、あまり小さすぎるデータでプロファイルを取ることは避け、できるだけ実用的なデータでプロファイルを取ることをお勧めします。データのサイズに

よって実行に時間のかかる部分が大きく変わってしまうことがあり得るからです。また、それでは実行時間が長くなりすぎてしまう、という場合には、たとえば繰り返し回数、実行ステップ数等が調整できるものであれば、これを調整して実行時間が適切になるようにしてください。これも、あまりステップ数を小さくしてしまうと初期化と終了処理ばかりが目立ってしまうということになるかもしれませんので、それなりの回数は実行するようにした方がよいでしょう。

さて、実行が終了すると、.xf という拡張子を持つファイルが生成されているはずです。これを pat_report というプログラムで処理します。

```
$ pat_report *.xf > patreport.txt
```

これを実行すると、patreport.txt というファイルにテキスト形式のレポート（本稿では参照しません）が出力され、さらに .xf というファイルが .ap2 という拡張子のファイルに変換されます。この ap2 というファイルが Reveal への入力となります。

2.5 プログラムライブラリ

対象となるプログラムを Reveal に読み込ませるためには、「プログラムライブラリ (program library)」というものを作成する必要があります。ライブラリというと *.a とか *.so とかの拡張子を持ったファイルを連想しがちですが、ここでいう「プログラムライブラリ」とは、コンパイラがプログラムを解析して得られた情報を保存しておくディレクトリのことで、CCE 専用の用語です。プログラムライブラリを作成するには、コンパイラオプションとして -h pl=(ディレクトリ名) を指定します。

```
$ cc -h pl=myprog.pl -h wp myprog.c  # C
$ CC -h pl=myprog.pl -h wp myprog.cc # C++
$ ftn -h pl=myprog.pl -h wp myprog.f # Fortran
```

この例では、myprog.pl という名前のディレクトリが作成され、その中にコンパイル情報が格納されます。.pl という拡張子が慣習で使用されますが、特に必要はありませんので、自由に名前を付けてください。ふだん他のコンパイルオプションを指定している場合には、それらも同じように指定してください。

-h wp (whole program) というオプションは、分割コンパイルをしている際に、ファイルをまたがった手続き間解析 (Inter-Procedural Analysis, IPA) を行うというオプションで、主に関数・サブルーチンのインライン展開に効果があります。後述するように、ループ中の関数・サブルーチン呼び出しをインライン展開することは重要ですので、分割コンパイルによってビルドされるプログラムに対して Reveal を使用する際には -h wp を指定することをお勧めします。

先に進む前に、プログラムライブラリについての注意点をいくつかご説明しておきます。

まず、プログラムライブラリは、ソースコードが変更されたら必ず作り直す必要があります。もし make 等でビルドを自動化している場合には、clean 時にプログラムライブラリも削除するようにしておいた方がよいでしょう。また、分割コンパイルしている場合には、必ずすべてのコンパイルおよびリンクにおいて同じプログラムライブラリを指定する必要があります。すべてのソースファイルが単一のディレクトリに含まれている場合にはあまり問題ないと思いますが、ファイルによってディレクトリが分かれている場合にはプログラムライブラリの指定は絶対パスで行う必要があります。

前節の手順で「Cray Pat によるプロファイル」を取られた方は、その際のビルドでプログラムライブラリを作成してしまえばよい、とお考えになるかもしれませんが、これはお勧めできません。その理由は、先述した通り -h profile_generate オプションによって最適化が変化してしまい、特に関数・サブルーチンがインライン展開されなくなってしまうことがあるためです。お手数ですが、プロファイル用のビルドとプログラムライブラリ生成用のビルドは分けて行うことをお勧めします。

3 Reveal を使う

3.1 Reveal の起動

さて、以上で Reveal を使う準備は調いましたの

で、実際に Reveal を起動してみましょう。Reveal を起動するには、reveal コマンドにプログラムライブラリを指定して実行します。

```
$ reveal myprog.pl
```

プロファイルを取得している場合には、ここで同時に ap2 ファイルを指定することもできます。

```
$ reveal myprog.pl myprog+pat+3218-201t.ap2
```

但し、ap2 ファイルは後で読み込ませることも可能です。ここでは、前者の方法で起動し、ap2 ファイルは後で読み込ませることにします。正常に起動できると、 図 1 のような画面が表示されます。

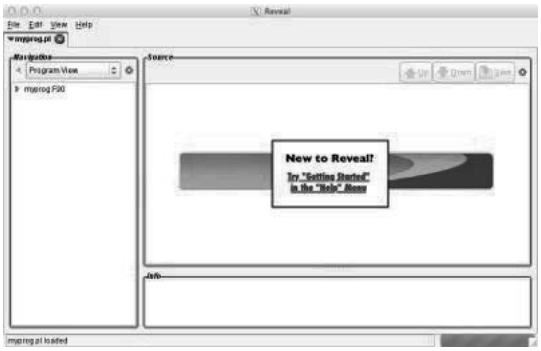


図 1　Reveal 起動直後の画面

3.2 ナビゲーション

まずは、Reveal 上でソースコードの内容を確認してみましょう。画面の左側に Navigation というパネルがあり、起動時には "Program View" と表示されています (プロファイル情報を読み込ませていない場合)。これは、プログラム全体を、ソースファイル、関数・サブルーチン、ループという階層で整理してナビゲーションするという意味です。各項目の左にある小さな三角をクリックすると、その項目が展開されて子項目が表示されるようになります (図 2)。そして、項目をクリックすると、その項目に該当するソースコード上の内容が右側に表示されます。

"Program View" と表示されているところを "Function List" に変更すると、ソースファイルの階層が表示されなくなり、関数・サブルーチン単位で整理されます。また、"Array List" を選択すると、関数ごとに使用している配列名がリストアップされ、配列名を選択するとソースコード上でその配列がハイライトされるようになります。"Compiler Message" を選択すると、CCE がフィードバックメッセージを生成した行がリストアップされます。

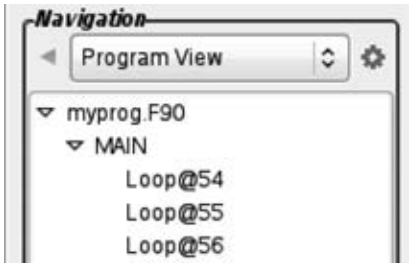


図 2　Navigation パネル

3.3 ループマーク

ループマーク (loopmark) とは、コンパイラがループをどのように最適化したかを簡潔に表す記号表記です。例えば、図 3 の 81 行目のループの左側には FVr2 と書かれていますが、これはそれぞれ「F := このループはフラット (サブルーチン呼び出しがない) である」「V := ベクトル化された」「r2 := 2 回アンロールされた」という意味です。ループマークの上にマウスカーソルをホバーさせると簡単な意味 ("Vectorized" のような) がツールチップに表示されます。ループマークの凡例は、メニューの Help → Loopmark Legend で確認することができます。

実は、ループマークは Reveal を使わなくても確認できます。コンパイラオプション -h list=m (C/C++ の場合) または -rm (Fortran の場合) を指定すると、.lst という拡張子のファイルが生成され、その中にループマークと説明が付けられたソースコードが出力されます。Reveal では、それを GUI 上で確認することができます。

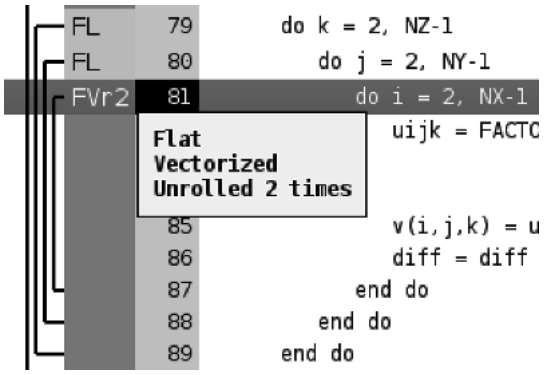


図 3　ループマークの例

3.4 コンパイラフィードバック

Navigation パネルでループを選択すると、画面下方にメッセージが表示されます。これは、コンパイラが生成したフィードバックメッセージで、ループマークの内容を説明したものになっています。メッセージをダブルクリックすると、新しいウィンドウが開き、そのメッセージの意味のさらに詳しい説明が表示されます。

3.5 プロファイル情報の追加

ここまでで、Reveal 上でソースコードをナビゲーションし、特にコンパイラの解析情報を確認することができるようになりました。次に、このプログラムのどこに実行時間がかかっているかを確認してみましょう。それには、プロファイル情報を追加する必要があります。メニューの File → Attach Performance Data を選択し、先ほど pat_report で生成した ap2 ファイルを選択してください。すると、Navigation が "Loop Performance" に変わり、各ループが実行時間順にソートされて表示されます。すなわち、いちばん上に表示されているループがいちばん時間のかかっているループということです。但し、ここで注意していただきたいのは、これらの時間は "inclusive" である、ということです。すなわち、多重ループがある場合、外側のループと内側のループの実行時間は重複して計上されます。したがって、全体の実行時間はそれらの和ではなく、最外ループの実行時間ということになります。図 4 の場合、79 行目、80 行目、81 行目は 3 重ループになっていて、同じくらいの実行時間が 3 回表示されています。全体の実行時間はもちろんこれらの合計ではなく、79 行目のループの実行時間として計上されている 54.7308 秒ということになります。

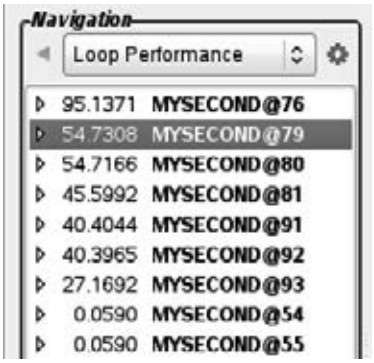


図 4　プロファイル情報の表示例

3.6 変数のスコーピング

前節で時間のかかっているループが特定できましたので、これを OpenMP で並列化したいと思います。そのために必要なことは「変数のスコーピング」です。

OpenMP をお使いになったことのある方はおわかりかと思いますが、ここで少し補足しておきます。OpenMP では for/do ループを並列化するために parallel for（C/C++ の場合）または parallel do（Fortran の場合）という指示行を使用します。これは、例えば リスト 1 のようなコードに対して、ループ変数（i, j, k）で指定されるループ本体の処理を一つのタスクと見なし、これを各スレッドに割り当てるという意味です。このときに、ループ本体で使用されている変数の「データ共有属性（data sharing attribute）」を決定する必要があります。例えば、ループ変数 i, j, k や一時変数 uijk のようなものは、スレッドごとに独立した値を持たなければなりません。このような変数は private であるといいます。一方、配列 u, v のようなものは、すべてのスレッドからアクセスできなければなりません。このような変数は shared であるといいます。diff は、同一のスカラー変数をすべてのスレッドから更新することになり、後述する「排他制御」が必要なように見えるかもしれませんが、このパターンの時は特別に「リダクション（reduction）演算」として並列に処理することができます。このように、ループ中の変数の共有属性を決定する作業を、Reveal で

は「変数のスコーピング」と呼んでいます。

リスト 1　Jacobi 法カーネルの例

```
do k = 2, NZ-1
do j = 2, NY-1
do i = 2, NX-1
   uijk = FACTOR * (u(i,j,k+1) + u(i,j,k-1) &
                  +  u(i,j+1,k) + u(i,j-1,k) &
                  +  u(i+1,j,k) + u(i-1,j,k))
   v(i,j,k) = uijk
   diff = diff + abs(uijk - u(i,j,k))
end do
end do
end do
```

OpenMP の文法自体はあまり難しくありませんが、実際に使用して難しいと感じたり時間がかかったりすることの一つは、このような変数の共有属性をひとつずつ確認しなければならないことでしょう。特にループが大きくなり変数の数が増えると、これらを確認するだけでかなりの時間がかかることがあります。また、指定を間違えると、コンパイルは通るが実行時におかしな挙動をするというデバッグしにくいバグの原因になることがあります。特に、default(shared) 等として private に指定すべき変数を忘れてしまう、というのは典型的なバグの温床です。

Reveal は、この「変数のスコーピング」を自動で行うことができます。実際にやってみましょう。まず、スコーピングしたいループを Navigation パネル上で右クリックし、"Scope Loop" を選択します。すると、"Scoping Tool" という別の画面が開き、選択したループがリストアップされているはずです。そこで、左下にあるボタン "Start Scoping" を押すとスコーピングが実行され、ループ本体で使用されている各変数の種類（スカラー変数か配列変数か）と共有属性（private か shared か）が自動的に決定され、その結果が "Scoping Result" というウィンドウに表示されます（図 5）。

後述するように、スコーピングは常に成功するとは限りません。スコーピングが成功したループは Navigation パネル上で緑色の四角で、失敗したループは赤色の丸で、それぞれマークされます。

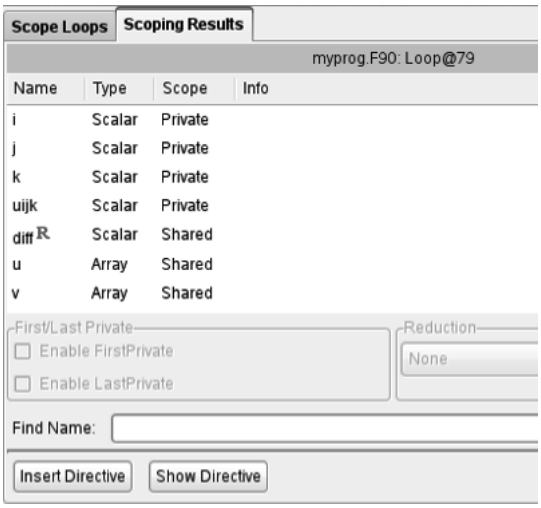


図 5　スコーピングの結果

3.7 OpenMP 指示行の生成

さて、先ほどの例のスコーピングは成功しましたので、OpenMP 指示行を生成してみましょう。Scoping Tool の Scoping Result が表示されている下の方に "Insert Directive" と "Show Directive" と書かれたボタンがあります。まず、Show Directive ボタンを押すと、新しいウィンドウが開いて対応する OpenMP 指示行が表示されます。これで確認して問題がなければ、Insert Directive ボタンを押すと、ソースエディタ上に指示行が挿入されます。

リスト 2　自動生成された OpenMP 指示行

```
! Directive inserted by Cray Reveal.
! May be incomplete.
!$OMP  parallel do default(none)          &
!$OMP&   private (i,j,k,uijk)             &
!$OMP&   shared  (u,v)                    &
!$OMP&   reduction (+:diff)
```

注意点は、この時点ではまだ実際の（ファイルシステム上の）ソースファイルには指示行は書き込まれていない、ということです。メインの画面に戻ると、右上の "Save" というボタンが有効になっていると思います。このボタンを押して初めてファイルに指示行が保存されます。但し、ソースファイルを変更してしまうと、コンパイラが生成した解析情報とソースファイルの整合性がとれなくなってしまうので、再コンパイルしてプログラムライブラリを

作り直すことが必要となります。したがって、変更点をソースファイルに出力するのは一連の作業の最後に行う必要があります。なお、スコーピングや Insert Directive 等の作業の途中経過は、ソースファイルに出力しなくてもプログラムライブラリ中に保存されていますので、ソースファイルに出力しなくても失われることはありません。

また、プログラムが複数のソースファイルから成り立っている場合、Save ボタンによって変更点を出力するためにはひとつひとつファイルを選択して行わなければならず、やや面倒です。このような場合は、メニューの File → Save All を選択するのがよいでしょう。この方法では出力対象とするソースファイルを複数選択して一括保存できますので、より簡単な操作で実行でき、出力漏れも防げると思います。

3.8 複数のループを一度に処理する

上記ではひとつのループを選択してスコーピングと OpenMP 指示行の挿入を行いましたが、プログラムが大きくなるとこれでは非効率的と感じるかもしれません。逆に、一度にすべてのループを処理することもできます。そのためには次のような手順を行います。

1. Navigation パネルの上方にある歯車のアイコンをクリックし、"Scope All Loops" を選択する (図 6)。
2. Scoping Tool が立ち上がるので "Start Scoping" をクリックする。
3. スコーピングが終了したらメニューから Edit → Insert All Valid Directives を選択する。

これで、スコーピングが正常終了したすべてのループを対象として、OpenMP 指示行を挿入できます。多重ループの場合は最外のループにのみ指示行が挿入されます。

図 6 "Scope All Loops" を選択する

このやり方は簡便ですが、実行時間の短すぎるループも区別なく対象としてしまうのが難点です。Insert All Valid Directives で対象とするファイルやループを指定したい場合は、スコーピングの時点でそれらのファイル・ループを選択しておく必要があります。上記手順 (2) で、Start Scoping をクリックする前にそのような選択をしておきましょう (図 7)。もし、少数のファイル・ループだけでよいということであれば、一度 Edit List → Uncheck All Items ですべてのループを選択解除してから必要なものだけ選択すると便利です。"Apply Filter" ボタンを使用すると、実行時間やループ長等の条件を与えてループを選択することも可能です。

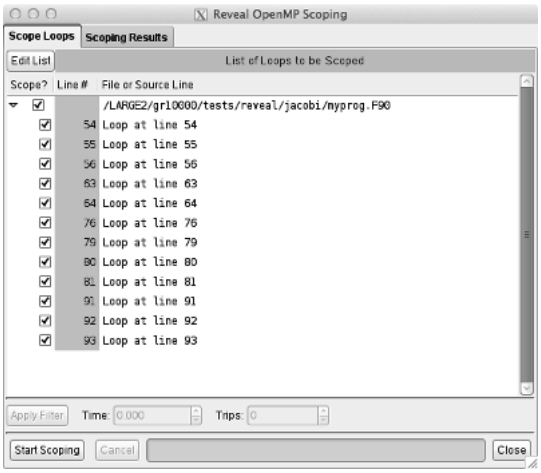


図 7 スコーピング対象のループを選択する

また、いったんスコーピングしてしまった後でも、スコーピング情報をクリアして (Edit → Clear All Scoping Information) やり直すことができます。

4 スコーピングに失敗する場合

次に、Reveal がスコーピングに失敗するケース

を見ていきます。これらのケースには、Reveal が対応していない場合と、そのままのプログラムでは本質的に並列化できない場合とがあります。

4.1 Reveal では扱えないケース

以下の場合には、Reveal ではループの解析ができません：

- ループ中に early exit が含まれる場合、
- ループ中に I/O が含まれる場合、
- ループ中に（インライン展開されない）関数・サブルーチン呼び出しが含まれる場合。

ここで、"early exit" というのは、ループの途中で break（C/C++）、exit（Fortran）、goto 等でループの外に出てしまうことを言います。early exit は、Reveal だけでなく OpenMP の仕様として禁止されていますので、OpenMP で並列化を行う際にはいずれにしてもコードの修正が必要です。I/O は、一般にはスレッドセーフではありませんので、OpenMP 並列領域内で I/O を行う場合には critical 指示行等による排他制御が必要です。関数・サブルーチン呼び出しは、それがスレッドセーフであれば OpenMP 仕様上は問題ありませんが、Reveal では残念ながら対応していません。

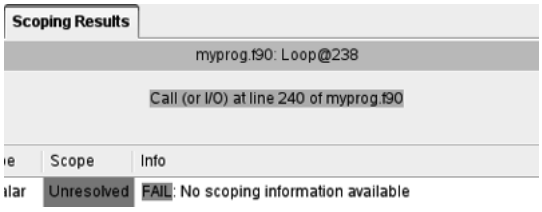


図 8　サブルーチン呼び出しによる失敗例

4.2 インライン展開

ループ中の関数・サブルーチン呼び出しが「インライン展開されていない場合」は上述のように Reveal では解析できませんが、インライン展開されている場合には問題なく解析することができます。この場合には、スコーピングを実行することができるだけでなく、呼び出し側の関数・サブルーチンにインライン展開されたコードを擬似コードで表示したり、コールサイト（呼び出し元）を追跡したりすることができます。

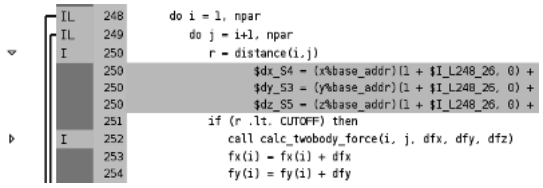


図 9　インライン展開された疑似コード

2.5 節で述べたように、関数・サブルーチンが複数のファイルにまたがって定義されている場合、これらのインライン展開を行うには -h wp オプションが必要です。-h wp オプションはさらにプログラムライブラリの作成を必要としますが、Reveal を利用するときは必ず作成しますので、問題なく -h wp オプションを追加することができます。Reveal を使用する際には -h wp オプションを指定することをお勧めします。

4.3 本質的に並列化できない場合

対象となるループが本質的に並列化できない場合は、もちろんスコーピングは失敗します。本質的に並列化できない場合の例として、スレッド間で共有されている変数への書き込みが存在している場合があります。この場合は、この処理を並列化すると共有変数に対する更新が競合し、処理のタイミングによって計算結果が異なるものになってしまう可能性があります。また、ループに依存性がある場合も失敗します。ループの依存性とは、ループ反復のある回の計算が他の回の計算結果に依存することを言います。例えば、配列要素のうち (i-1) 回目で更新した値を用いて i 回目の値を計算する、といったケースがこれに該当します。この場合も計算結果がループ反復の実行順序に依存してしまうため、並列処理を行うことができません。このようなループを OpenMP によって並列化するには、threadprivate 指示行を挿入する、critical/atomic 指示行で排他制御を行う、ループのアルゴリズムを変更する等の対策が必要となりますが、残念ながらこれらを Reveal で自動で行うことはできません。

このようなループを Reveal で処理するとどのような結果が得られるのか試してみましょう。例として、リスト 3 のようなループを考えます。これは、

分子動力学法のプログラムでよく見られるパターンで、粒子 i と、それと相互作用する粒子 j に働く力 (fx, fy, fz) を更新するループです。作用反作用の法則のために力は必ずペアで更新されますが、そのためこの i のループを並列化すると粒子 j に対する更新が他のスレッドによる更新と競合する可能性があります。

リスト 3　自動で並列化できないコード例

```
do i = 1, npar
do j = i+1, npar
  r = distance(i,j)
  if (r .lt. CUTOFF) then
    call
calc_twobody_force(i,j,dfx,dfy,dfz)
    fx(i) = fx(i) + dfx
    fy(i) = fy(i) + dfy
    fz(i) = fz(i) + dfz
    fx(j) = fx(j) - dfx
    fy(j) = fy(j) - dfy
    fz(j) = fz(j) - dfz
  end if
end do
end do
```

このループを Reveal によってスコーピングすると、図 10 のように fx, fy, fz に対して "Unresolved" と判定されます。これは、この変数には依存性が疑われるため、スコーピングできない、という意味です。Unresolved がある場合、Reveal は正しい並列化ができませんので、先述した通り手動で問題を解決する必要があります。

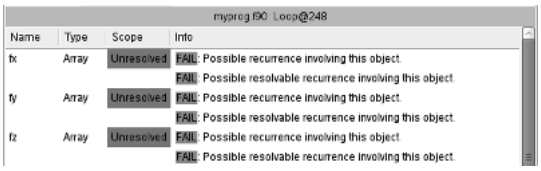


図 10　Unresolved 変数の例

しかし、この場合でも Reveal によって指示行を生成することは可能です。実際に Insert Directive を行うと、リスト 4 のような指示行が挿入されます。unresolved という節が指定されていますが、もちろんこれは OpenMP ではサポートされていないものですので、このままコンパイルするとエラーになります。これは、「ここを手動で修正する必要がある」というヒントだとお考えください。ここでは、力 (fx, fy, fz) の更新について競合の発生する可能性がありますので、atomic 指示行を挿入することによってこれを排他制御します。その上で、unresolved を shared に変更します。これで、正しくコンパイル・実行できるプログラムになりました (リスト 5)。

リスト 4　unresolved を含む指示行

```
! Directive inserted by Cray Reveal.
! May be incomplete.
!$OMP  parallel do default(none)          &
!$OMP&   unresolved (fx,fy,fz)            &
!$OMP&   private (dfx,dfy,dfz,i,j,r)      &
!$OMP&   shared  (npar)
```

リスト 5　手動で修正したコード例

```
!$OMP  parallel do default(none)          &
!$OMP&   shared  (fx,fy,fz)               &
!$OMP&   private (dfx,dfy,dfz,i,j,r)      &
!$OMP&   shared  (npar)
do i = 1, npar
do j = i+1, npar
  r = distance(i,j)
  if (r .lt. CUTOFF) then
    call
calc_twobody_force(i,j,dfx,dfy,dfz)
!$omp atomic update
    fx(i) = fx(i) + dfx
!$omp atomic update
    fy(i) = fy(i) + dfy
!$omp atomic update
    fz(i) = fz(i) + dfz
!$omp atomic update
    fx(j) = fx(j) - dfx
!$omp atomic update
    fy(j) = fy(j) - dfy
!$omp atomic update
    fz(j) = fz(j) - dfz
  end if
end do
end do
```

このように、依存性等の解決は Reveal を使用しても自動ではできず、手動で行う必要があります。しかし、Unresolved をヒントとすることで並列化

を阻害する要因の特定が容易となり、コンパイラの自動並列化では並列化できないループを選んで手動で並列化することが可能です。また、本来並列化してはいけないループを並列化してしまう等の誤りを回避する助けにもなるでしょう。

4.4 C/C++ で複数のポインタを使用している場合

C/C++ を使用していて、ループ中でポインタを複数使用している場合には、スコーピングが失敗します。これは、ポインタに関していわゆる「エイリアス問題」があるためです。あるコードブロックを実行中に複数の異なるポインタ変数が同一の実体を参照することがある場合に、これらのポインタ（参照）は互いに「エイリアス（alias、別名）」であると言います。ループ中で使用されているポインタについてこのようなことが発生すると依存性の原因となるため、並列化ができなくなります。コンパイラは、コンパイル時にエイリアスが存在するかどうか判断できない場合、安全のため、エイリアスがあっても正常に動作するような（保守的な）コードを生成します。結果として、エイリアスが解決されていないポインタがあると、コンパイラは（Reveal も）そのループは並列化できないと判断してしまうのです。実際には並列化だけでなく、ベクトル化のようなループに対する最適化の多くを適用不可と判断してしまうため、大きな性能上の損失を引き起こす可能性があります。

例として、リスト 1 と同じ処理を C 言語で記述した リスト 6 のようなコードをスコーピングすると、図 11 のように u および v が Unresolved となりスコーピングに失敗してしまいます。処理内容としては Fortran の場合と全く同一ですから並列化できるはずですが、ポインタのエイリアスについてコンパイラが理解できなかったために、並列化できないと判断されてしまいました。

リスト 6　ポインタを使用した Jacobi 法

```
float jacobi_kernel(
  float *u, float *v, int nx, int ny, int
nz
)
{
    (中略)
    for (idx = is; idx <= ie; ++idx) {
        (中略)
        uijk = FACTOR *
            (u[idx+1]      + u[idx-1]
          +  u[idx+nz]     + u[idx-nz]
          +  u[idx+nz*ny] + u[idx-nz*ny]);
        v[idx] = uijk;
        diff += FABS(uijk - u[idx]);
    }

    for (idx = is; idx <= ie; ++idx) {
        u[idx] = v[idx];
    }

    return diff;
}
```

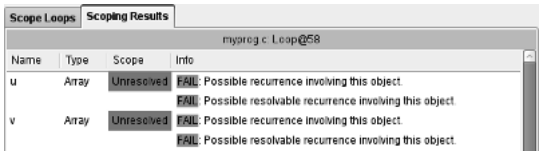


図 11　ポインタ u,v のスコーピング結果

これを解決するには、次のうちのいずれかの方法を適用します。

1. restrict ポインタを使用する。

「restrict ポインタ」とは、ポインタに対して「エイリアスが存在しない」ことを宣言したものです。C99 規格で標準化されている他、多くの C/C++ コンパイラで拡張機能としてサポートされています。ポインタ u, v を restrict であると宣言するには、以下のように restrict または __restrict__ キーワードを追加します。

```
float * restrict u, // C99 の場合
float * __restrict__ v // C++ の場合
```

これによって、コンパイラは u や v にはエイリアスが存在しないと仮定でき、これらを参照しているループに並列化を含めたより積極的な最適化を適用できます。但し、「本当にエイリアスが発生しない

ようにプログラミングする」のはプログラマの責任です。もし、restrict と宣言したポインタに対してエイリアスが発生してしまった場合には、計算結果が不正になるなどの問題が生じる可能性がありますのでご注意ください。

2. -h restrict オプションを使用する。

CCE では、ポインタを暗黙に restrict とみなすオプションが用意されています。-h restrict=a と指定するとすべてのポインタが restrict ポインタであると仮定されます。-h restrict=f と指定すると、関数引数として与えられたすべてのポインタが restrict ポインタであると仮定されます。リスト 6 の場合、u および v は関数引数ですので、-h restrict=f と指定することでエイリアス問題を解決できます。ソースコードを修正しなくてよいという便利さはありますが、先述した通り、実際にエイリアスが発生してしまうとプログラムの実行に支障をきたす可能性がありますので、慎重に使用することをお勧めします。

なお、2 つの配列をポインタを経由せず直接アクセスする場合には、コンパイラは 2 つの参照が異なるメモリ領域を参照していることを認識できるため、エイリアス問題は発生しません。しかし、C/C++ では、配列をポインタとして関数の引数に渡したり、malloc 等で動的にメモリ確保を行うことは一般的ですから、ポインタを避けて通ることはできないでしょう。したがって、ポインタのエイリアスに注意を払うことは、C/C++ プログラムの性能改善にとって非常に重要であると言えます。

5 まとめ

以上、本稿では Reveal の使用方法を概観しました。すべてのループが自動並列化できれば簡単なのですが、それができない場合に、自動並列化と手動による OpenMP 指示行の挿入との間を埋めるツールが Reveal です。OpenMP 初心者の方は指示行挿入の補助として、エキスパートの方は時間のかかるスコーピング作業の自動化のために、それぞれの目的に合わせてお使いいただけると思います。本稿の内容が、皆様のプログラムの性能改善の一助となれば幸いです。

6 参考文献・リンク

[1] X Window System (Exceed onDemand) の使用方法
- http://web.kudpc.kyoto-u.ac.jp/manual/ja/install/eod
- http://web.kudpc.kyoto-u.ac.jp/manual/ja/login/eod

[2] サブシステム D でのジョブ実行方法
- http://web.kudpc.kyoto-u.ac.jp/manual/ja/run
- http://web.kudpc.kyoto-u.ac.jp/manual/ja/run/batchjob/systema

[3] クレイコンパイラ (CCE) 使用方法
man craycc, crayCC, crayftn をご参照ください。(サブシステム D にログインし、PrgEnv-cray モジュールがロードされていることをご確認ください。)

[4] Cray Pat 使用方法
man intro_craypat, pat_build, pat_report, pat_help をご参照ください。(サブシステム D にログインし、module load perftools を実行した後にご利用ください。)

[5] 武田大輔: CrayPat による性能解析, 京都大学学術情報メディアセンター全国共同利用版広報, Vol . 12 No. 1 (2013)

[6] Reveal 使用方法
man reveal をご参照ください。(サブシステム D にログインし、module load perftools を実行した後にご利用ください。)

システム A 運転状況 （2014 年 4 月 ～ 2014 年 9 月 ）

1）保守作業に伴うサービス休止およびシステムダウン障害発生状況

保守作業に伴うサービス休止

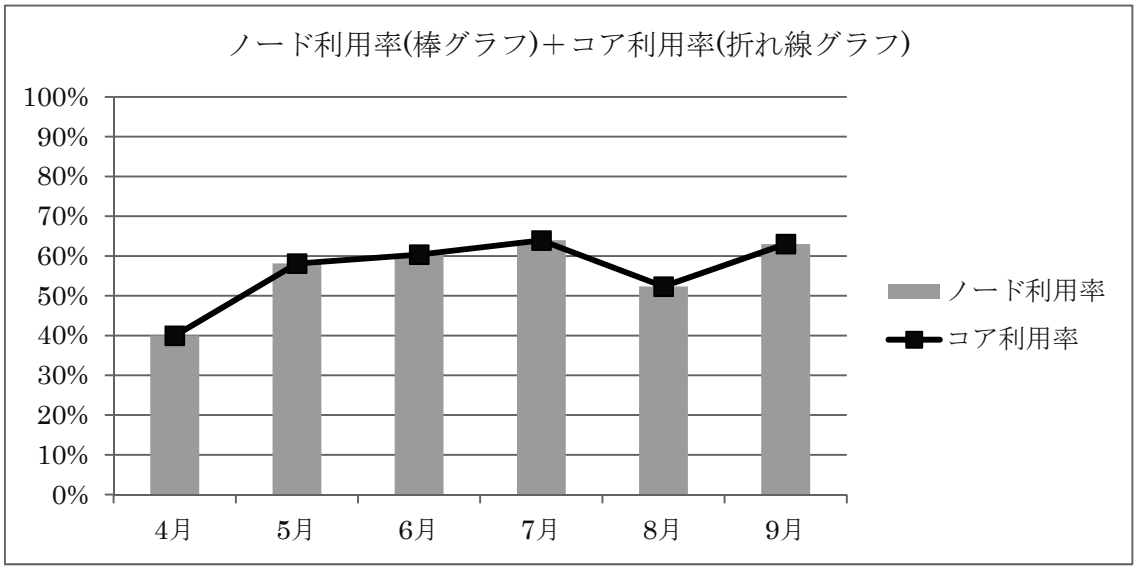
保守開始日時	サービス再開日時	保守時間[h]
2014/04/01 0:00	2014/04/02 9:15	33.15
2014/06/11 9:00	2014/06/13 9:00	48.00
2014/08/05 9:00	2014/08/06 9:00	24.00

システムダウン障害発生状況

障害発生日時	サービス再開日時	ダウン時間[h]

2）サービス状況

	サービス時間[h]	バッチ					
		処理件数	経過時間[h]	占有時間[h]	CPU時間[h]	平均稼動ノード数	ノード利用率
4 月	686.45	13,902	62,959	8,375,500	7,566,480	939.9	40 %
5 月	744.00	16,422	82,176	12,399,200	10,634,600	939.7	58 %
6 月	672.00	14,925	71,699	11,567,500	9,993,420	923.7	60 %
7 月	744.00	17,208	96,989	14,221,200	12,781,000	939.9	64 %
8 月	720.00	17,260	90,000	12,009,000	10,760,900	939.9	52 %
9 月	720.00	18,581	80,206	13,600,100	11,779,400	940.0	63 %
計	4286.45	98,298	484,030	72,172,500	63,515,800	937.2	56 %



- 占有時間 = 合計(経過時間×占有コア数)
- 平均稼動ノード数 = 電源 ON 状態のノード数の月平均 (10 分間隔のサンプリングデータより算出)
- ノード利用率 = 稼動ノードに対するジョブが実行されているノードの割合

システム B 運転状況 （2014 年 4 月 ～ 2014 年 9 月 ）

1）保守作業に伴うサービス休止およびシステムダウン障害発生状況

保守作業に伴うサービス休止

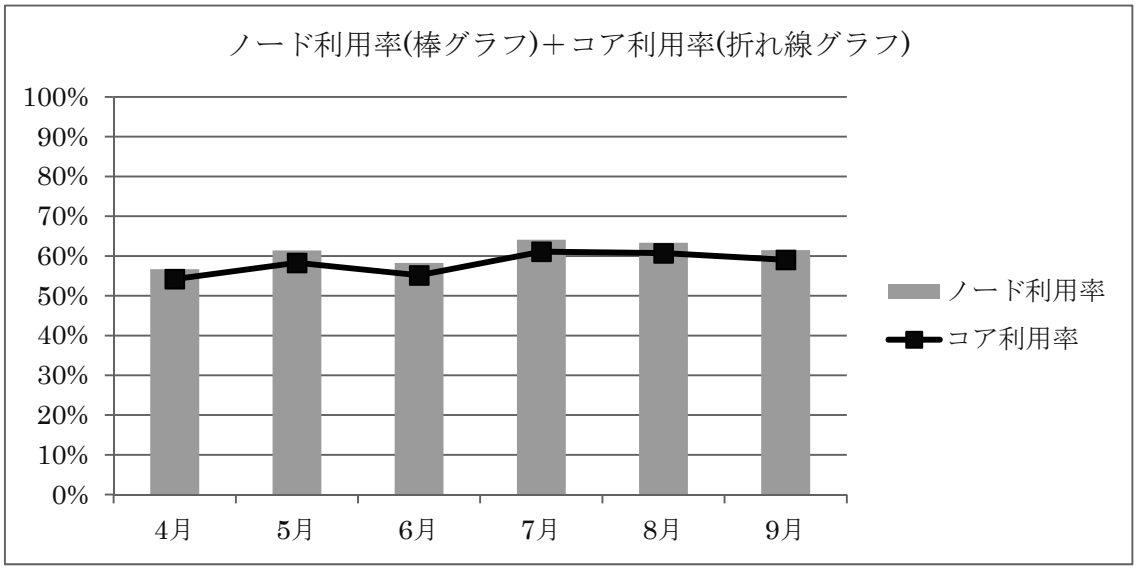
保守開始日時		サービス再開日時		保守時間[h]
2014/04/01	0:00	2014/04/02	9:15	33.15
2014/06/11	9:00	2014/06/13	9:00	48.00
2014/08/05	9:00	2014/08/06	9:00	24.00

システムダウン障害発生状況

障害発生日時	サービス再開日時	ダウン時間[h]

2）サービス状況

	サービス時間[h]	バッチ					
		処理件数	経過時間[h]	占有時間[h]	CPU時間[h]	平均稼動ノード数	ノード利用率
4 月	686.45	678,823	216,022	2,944,350	268,242	511.5	57 %
5 月	744.00	348,782	509,213	3,810,850	312,566	533.9	61 %
6 月	672.00	527,419	613,903	3,122,830	264,409	501.2	58 %
7 月	744.00	313,035	585,352	3,746,890	312,799	533.0	64 %
8 月	720.00	253,088	526,942	3,932,670	325,228	532.4	63 %
9 月	718.10	257,749	597,627	3,648,760	312,089	532.9	62 %
計	4284.55	2,378,896	3,049,059	21,206,350	1,795,333	524.1	61 %



- 占有時間 = 合計(経過時間×占有コア数)
- 平均稼動ノード数 = 電源 ON 状態のノード数の月平均 (10 分間隔のサンプリングデータより算出)
- ノード利用率 = 稼動ノードに対するジョブが実行されているノードの割合

システム C 運転状況 （2014 年 4 月 ～ 2014 年 9 月 ）

1）保守作業に伴うサービス休止およびシステムダウン障害発生状況

保守作業に伴うサービス休止

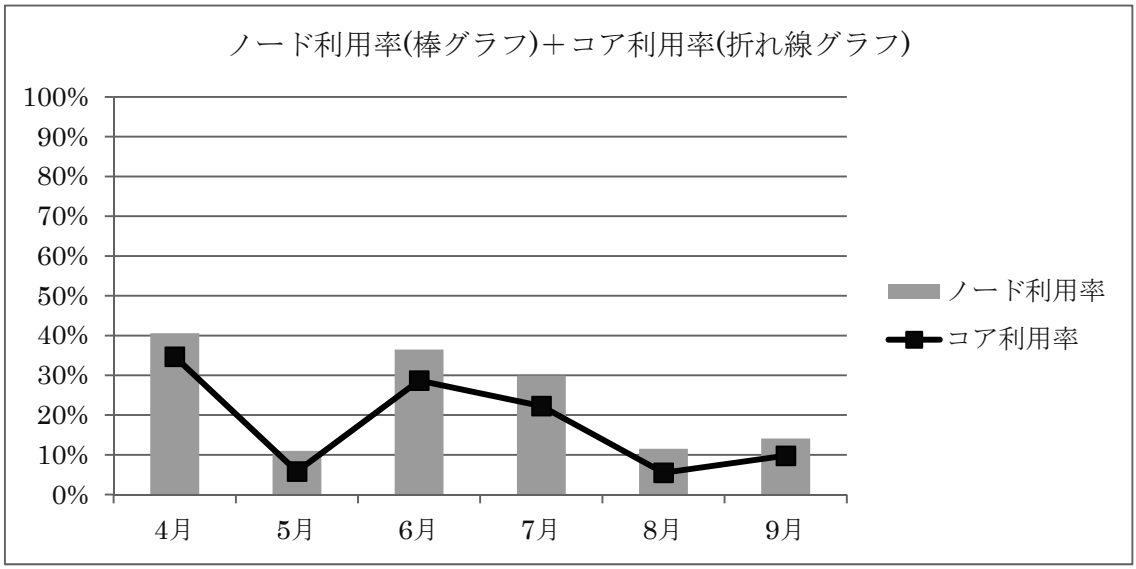
保守開始日時		サービス再開日時		保守時間[h]
2014/04/01	0:00	2014/04/02	9:15	33.15
2014/06/11	9:00	2014/06/13	9:00	48.00
2014/08/05	9:00	2014/08/06	9:00	24.00

システムダウン障害発生状況

障害発生日時	サービス再開日時	ダウン時間[h]

2）サービス状況

	サービス時間[h]	バッチ					
		処理件数	経過時間[h]	占有時間[h]	CPU時間[h]	平均稼動ノード数	ノード利用率
4 月	686.45	107,479	5,537	117,188	88,871	15.2	41 %
5 月	744.00	1,032	3,625	26,137	13,595	16.0	11 %
6 月	672.00	1,892	4,018	96,660	65,508	15.6	37 %
7 月	744.00	12,664	5,014	89,189	53,933	16.0	30 %
8 月	720.00	2,454	3,759	20,222	11,201	16.0	12 %
9 月	718.10	1,988	3,236	37,013	25,663	16.0	14 %
計	4284.55	127,509	25,188	386,409	258,771	15.8	24 %



- 占有時間 = 合計(経過時間×占有コア数)
- 平均稼動ノード数 = 電源 ON 状態のノード数の月平均 (10 分間隔のサンプリングデータより算出)
- ノード利用率 = 稼動ノードに対するジョブが実行されているノードの割合

システム D 運転状況 （2014 年 7 月 ～ 2014 年 9 月 ）

1）保守作業に伴うサービス休止およびシステムダウン障害発生状況

保守作業に伴うサービス休止

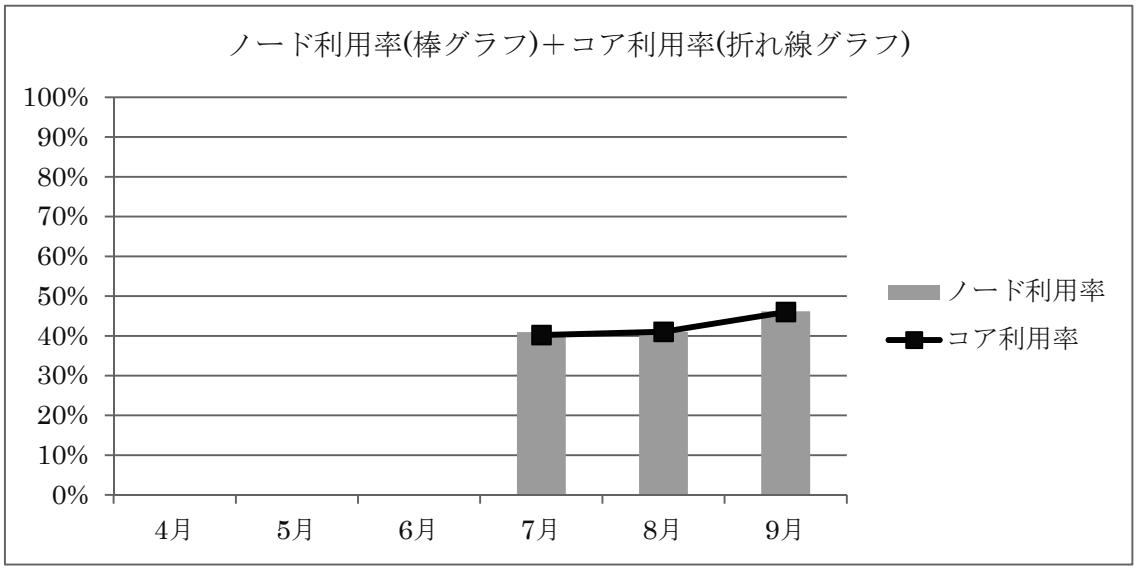
保守開始日時	サービス再開日時	保守時間[h]
2014/08/05 9:00	2014/08/06 9:00	24.00

システムダウン障害発生状況

障害発生日時	サービス再開日時	ダウン時間[h]
2014/08/23 7:00	2014/08/25 8:45	49.45

2）サービス状況

	サービス時間[h]	バッチ					
		処理件数	経過時間[h]	占有時間[h]	CPU時間[h]	平均稼動ノード数	ノード利用率
7 月	744.00	6,488	6,083	2,090,390	144,777	256.0	41 %
8 月	670.15	5,064	9,872	2,102,190	813,454	253.7	41 %
9 月	720.00	2,840	11,308	2,394,910	206,649	255.5	46 %
計	2134.10	14,392	27,262	6,587,490	1,164,880	255.0	43 %



- 占有時間 = 合計(経過時間×占有コア数)
- 平均稼動ノード数 = 電源 ON 状態のノード数の月平均 (10 分間隔のサンプリングデータより算出)
- ノード利用率 = 稼動ノードに対するジョブが実行されているノードの割合

システム E 運転状況 （2014 年 4 月 ～ 2014 年 9 月 ）

1）保守作業に伴うサービス休止およびシステムダウン障害発生状況

保守作業に伴うサービス休止

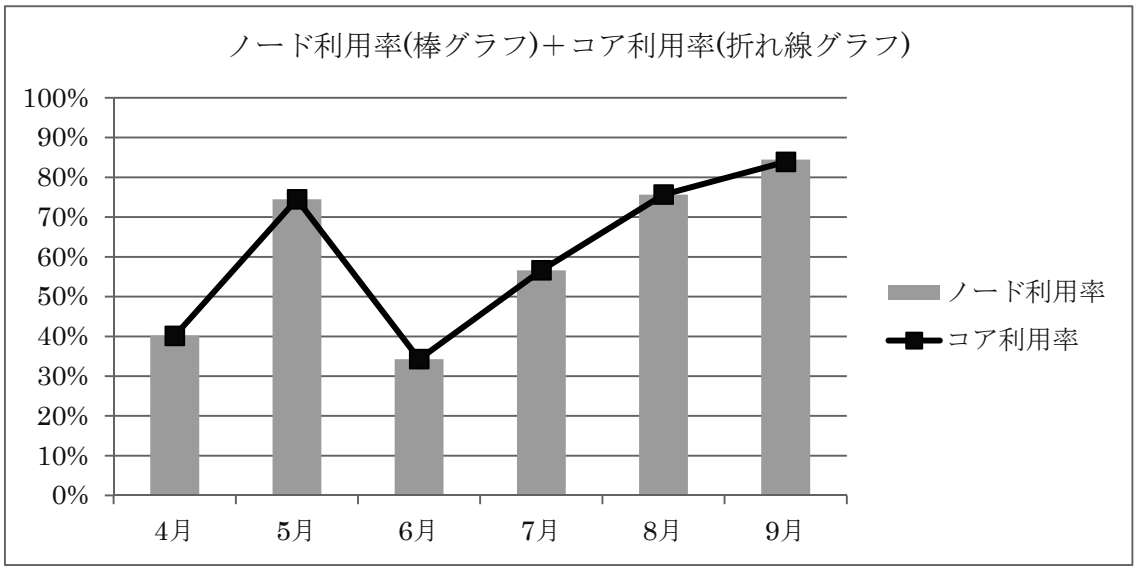
保守開始日時		サービス再開日時		保守時間[h]
2014/04/01	0:00	2014/04/02	9:15	33.15
2014/04/17	12:00	2014/04/17	14:30	2.30
2014/06/11	9:00	2014/06/13	10:10	49.10
2014/08/05	9:00	2014/08/06	9:00	24.00

システムダウン障害発生状況

障害発生日時		サービス再開日時		ダウン時間[h]
2014/04/14	9:35	2014/04/14	11:30	1.55
2014/08/23	7:00	2014/08/25	8:45	49.45

2）サービス状況

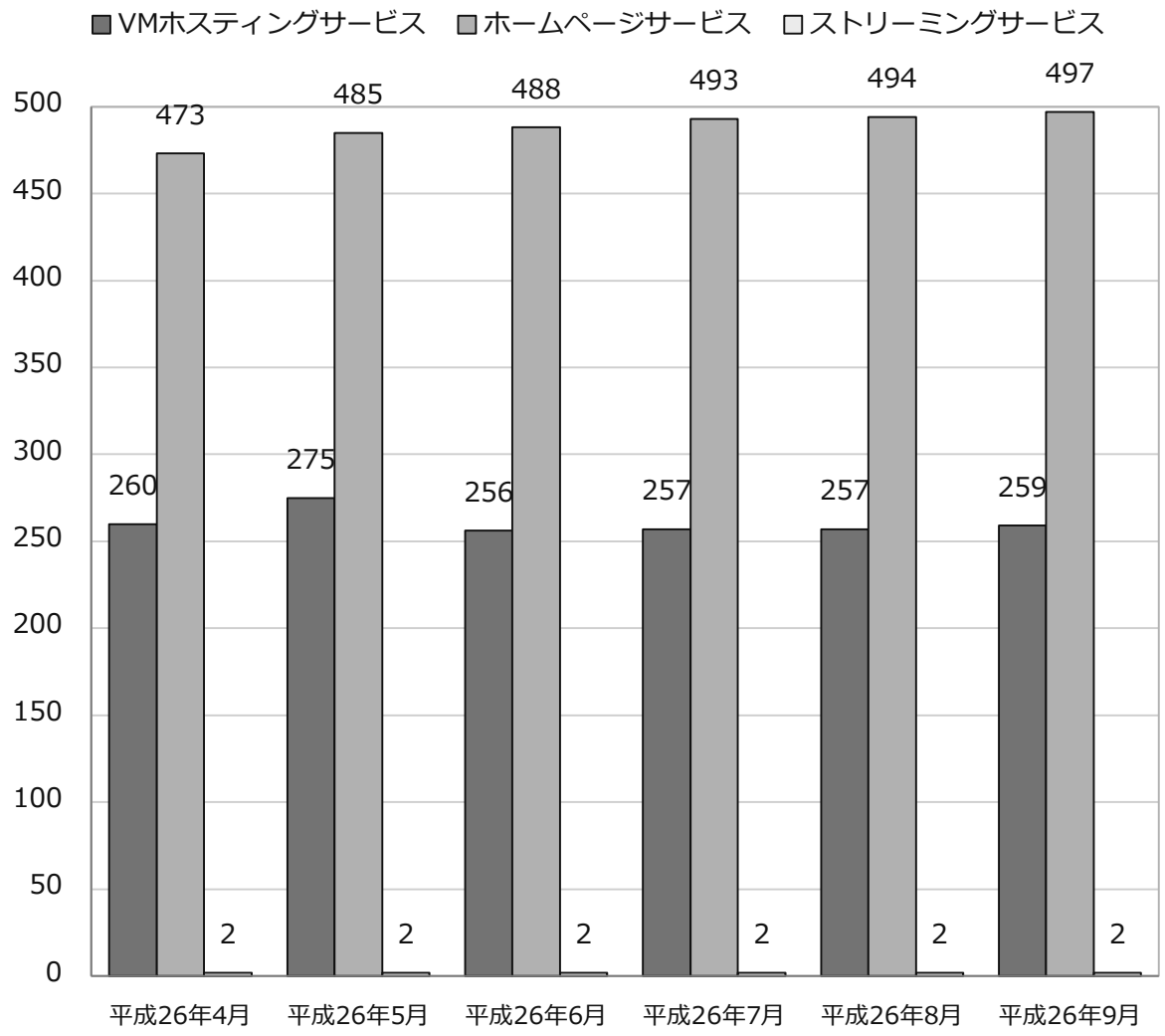
	サービス時間[h]	バッチ					
		処理件数	経過時間[h]	占有時間[h]	CPU時間[h]	平均稼動ノード数	ノード利用率
4 月	682.20	1,335	17,052	972,087	26,835	353.1	40 %
5 月	744.00	1,827	22,554	1,963,930	36,267	353.9	75 %
6 月	670.50	2,850	13,632	815,709	28,495	338.6	34 %
7 月	744.00	3,944	38,283	1,450,290	36,077	354.0	57 %
8 月	670.15	2,481	40,563	1,862,810	43,723	353.1	76 %
9 月	720.00	3,762	46,981	2,280,770	43,544	357.5	85 %
計	4231.25	16,199	179,065	9,345,596	214,941	351.7	61 %



- 占有時間 = 合計(経過時間×占有コア数)
- 平均稼動ノード数 = 電源 ON 状態のノード数の月平均 (10 分間隔のサンプリングデータより算出)
- ノード利用率 = 稼動ノードに対するジョブが実行されているノードの割合

汎用コンピュータシステムのサービス状況

1. ホスティング・ホームページサービス利用状況



(平成26年4月から平成26年9月)

大型計算機システム利用承認件数について

平成26年9月末現在、大型計算機システムの利用件数は、2,777件となっています。

センター利用による研究成果

―平成 25 年度計算機利用結果報告書から―

【1000　情報学】

1. Nakada, S., Y. Ishikawa, T. Awaji , T. In, K. Koyamada, M. Watanobe, H. Okumura, Y. Nishida, S. Saitoh : An integrated approach to the heat and water mass dynamics of a large bay: High-resolution simulations of Funka Bay, Japan : Journal of Geophysical Research Ocean, Vol.118, No.7, pp.3530 – 3547, 2013
2. Zhao K., S. Nakada, N. Sakamoto, K. Koyamada, C. Bajaj, Y. Ishikawa, T. Awaji, T. In, S. Saitoh : A Vasualization for the Dynamic Behaviors of the Mixture of Water Mass for Northwestern Pacific Near Japan : International Journal of Modeling, Simulation, and Scientific Computing, Vol.04, No.1341002, pp.1 – 18, 2013
3. Xun Zhang, Sei-Ichi Saitoh, Toru Hirawake, Satoshi Nakada, Koji Koyamada, Toshiyuki Awaji, Yoichi Ishikawa, Hiromichi Igarashi : An attempt of dissemination of potential fishing zones prediction map of Japanese common squid in the coastal water, southwestern Hokkaido, Japan : Proceedings of the Asia-Pacific Advanced Network, Vol.36, pp.132 – 141, 2013
4. Kazuki Semba, Koji Tani, Takashi Yamada, Takeshi Iwashita, Yasuhito Takahashi, Hiroshi Nakashima : Parallel Performance of Multi-threaded ICCG Solver Based on Algebraic Block Multi-color Ordering In Finite Element Electromagnetic Field Analyses : IEEE Trans. Magnetics, Vol.49, No.5, pp.1581 – 1584, 2013
5. Takeshi Minami, Motoharu Hibino, Tasuku Hiraishi, Takeshi Iwashita, Hiroshi Nakashima : Performance Improvement of Three-Dimensional Tiled FDTD Kernel Based on Automatic Parameter Tuning : Proc. Intl. Conf. Computation on Electromagnetic Fields, pp.1 – 2, 2013
6. Masaru Ueno, Tasuku Hiraishi, Motoharu Hibino, Takeshi Iwashita, Hiroshi Nakashima : Multilingualization Based on RPC for Job Level Parallel Script Language Xcrypt : IPSJ Trans. Programing, Vol.6, No.2, pp.54 – 68, 2013
7. Kazuki Semba, Koji Tani, Takashi Yamada, Takeshi Iwashita, Yasuhito Takahashi, Hiroshi Nakashima : Parallel Performance of Multi-threaded ICCG Solver in Electromagnetic Finite Element Analyses on the Latest Processors : Proc. Progress In Electromagnetics Research Symp., pp.979 – 983, 2013
8. Yohei Miyake, Chris M. Cully Hideyuki Usui, Hiroshi Nakashima : Plasma Particle Simulations of Wake Formation behind a Spacecraft with Thin Wire Booms : J. Geophysics Research, Vol.118, pp.1 – 14, 2013
9. Yohei Miyake, Hiroshi Nakashima : Low-Cost Load Balancing for Parallel Particle-in-Cell Simulations with Thick Overlapping Layers : Proc. Intl. Symp. Parallel and Distributed Processing with Applications, 2013

【1300　人間医工学】

10. 郭永明、 松山金寛、 石堂康弘：人工股関節ステムの初期固定性の有限要素解析：日本機械学会九州支部鹿児島講演会講演論文集, No.138-3, pp.13 – 14, 2013

【4100　数学】

11. 新谷誠, 綱川隆司, 梶博行：日本語と英語 Wikipedia のカテゴリー構造の整合性について：言語処理学会第 20 回年次大会, No.B6-3, 2014

【4300　物理学】

12. 間嶋拓也, 西尾達哉, 北島謙生, 今井誠, 土田秀次, 伊藤秋男 : MeV イオンと微小液滴の衝突

I：液滴サイズ分布測定： 日本物理学会講演概要集, Vol. 69, No.1, pp.196 – 196, 2014

13. Mitsuru TANAKA : Inverse Transverse Migration of Small Bubbles in Turbulence : Journal of the Physical Society of Japan, Vol.82, No.4, 044401-1 - 044401-14, 2013
14. Takashi Miyake, Kiyoyuki Terakura, Yosuke Harashima, Hiori Kino and Shoji Ishibashi : First-principles study of magnetocrystalline anisotropy and magnetization in NdFe12, NdFe11Ti and NdFe11TiN： J. Phys. Soc. Jpn., Vol. 83, pp.043702-1 - 043702-4, 2014

【4400 地球惑星科学】

15. 武村一史：一般座標系を用いた数値モデルによる山岳波のシミュレーション： 日本気象学会 2013 年度秋季大会講演予稿集, pp.373 – 373, 2013
16. Miyake, Y., C. M. Cully, H. Usui, and H. Nakashima : Plasma Particle Simulations of Wake Formation Behind a Spacecraft with Thin Wire Booms： Journal of Geophysical Research, Vol.118, No.9, pp.5681 – 5694, 2013
17. Miyake, Y., and H. Nakashima : Low-Cost Load Balancing for Parallel Particle-In-Cell Simulations with Thick Overlapping Layers : Proc. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp.1107 – 1114, 2013

【4500 プラズマ科学】

18. Paul P. Hilscher, Kenji Imadera, Jiquan Li and Yasuaki Kishimoto : The effect of weak collisionality on damped modes and its contribution to linear mode coupling in gyrokinetic simulation： Physics of Plasmas, Vol.20, pp.082127-1 - 082127-14, 2013
19. Paul P. Hilscher, Kenji Imadera, Jiquan Li and Yasuaki Kishimoto : Role of Stable Modes in the ITG-Driven Instability in a Mode-Coupled System： Plasma and Fusion Research, Vol.8, pp.1303151-1 - 1303151-3, 2013
20. Paul P. Hilscher, Kenji Imadera, Jiquan Li and Yasuaki Kishimoto : Gyrokinetic Simulations of Short-Wavelength ITG Instability in the Presence of a Static Magnetic Island： Plasma and Fusion Research, Vol.8, pp.2403040-1 - 2403040-4, 2013
21. Aohua Mao, Jiquan Li, Yasuaki Kishimoto and Jinyuan Liu : Stability of Double Tearing Mode in the Presence of Shear Flows : Plasma and Fusion Research, Vol.8, pp.2403121-1 - 2403121-5, 2013

【5000 機械工学】

22. 堀司, 柴垣大貴, 長崎茜, 白神洋輔, 毛笠明志, 西家隆行, 渡邊裕章, 黒瀬良一： Flamelet/progress-variable 法による管状火炎バーナの数値解析： 第 51 回燃焼シンポジウム講演論文集, Vol. 51, pp.184 – 185, 2013
23. 上田桂, 西川雅章, 北條正樹, 武田一朗：繊維よれを含む繊維強化複合材料の圧縮強度に関する座屈分岐解析：第４回日本複合材料合同会議講演論文集, 2013
24. 西川雅章, 北條正樹：繊維強化複合材料における三次元微視変形解析手法の検討：日本材料学会 第６２期通常総会・学術講演会 講演論文集, pp.(No. 418), 2013
25. 松田慎平, 西川雅章, 北條正樹：分子モデルを用いた高分子粘弾性特性評価のためのシミュレーション：第 18 回計算工学講演会 講演論文集, Vol.18, pp.(D-2-3), 2013
26. 西川雅章, 北條正樹：樹脂の非線形性を考慮した繊維強化複合材料の微視力学解析法：日本機械学会 2013 年度年次大会 講演論文集, pp.(J046034), 2013
27. 塚本匠, 西川雅章, 北條正樹：プリフォーム複合材の力学微視構造と疲労損傷累積過程との相関に関する解析： 第 38 回複合材料シンポジウム 講演要旨集, pp.(B1-4-2), 2013
28. 上田桂, 西川雅章, 北條正樹, 武田一朗：圧縮せん断混合負荷下における繊維強化複合材料のキンク発生応力に関する有限要素解析：第 38 回複合材料シンポジウム 講演要旨集, pp.(A1-1-3), 2013
29. 関谷優太, 西川雅章, 北條正樹：形状記憶複合材料を用いたモーフィングスキンにおける材料特性分布の最適化：日本機械学会 M&M2013 材料力学カンファレンス 講演論文集,

pp.(OS1613), 2013

30. Masaaki Nishikawa, Masaki Hojo : Application of Three-Dimensional Fiber-Based Simulation to Strength Prediction of Fiber Reinforced Composite with Process-Induced Microstructural Heterogeneities : Proceedings of the 13th Euro-Japanese Symposium on Composite Materials, 2013
31. 後藤聡, 西川雅章, 北條正樹：繊維基材を用いた複合材料の硬化に伴う残留ひずみの評価：日本機械学会 第 21 回機械材料・材料加工技術講演会 講演論文集, pp.(111), 2013

【5200 土木工学】

32. 川合裕太, 野田利弘, 山田正太郎, 浅岡顕, 澤田義博：横ずれ断層に伴うフラワー構造を伴うリーデルせん断帯生成の数値シミュレーション：土木学会第 69 年次学術講演会, No.Ⅲ-371, 741~742, 2014

【5400 材料工学】

33. T. Yokoyama, F. Oba, A. Seko, H. Hayashi, Y. Nose, and I. Tanaka : Theoretical photovoltaic conversion efficiencies of ZnSnP2 , CdSnP2 , and Zn1-x Cdx SnP2 alloys : Applied Physics Express, Vol.6, No.6, pp.061201-1 - 061201-3, 2013
34. Y. Hinuma, F. Oba, Y. Kumagai, and I. Tanaka : Band offsets of CuInSe2/CdS and CuInSe2/ZnS (110) interfaces: A hybrid density functional theory study : Physical Review B, Vol.88, No.3, pp.035305-1 - 035305-12, 2013
35. Y. Hinuma, F. Oba, Y. Nose, and I. Tanaka : First-principles study of valence band offsets at ZnSnP2/CdS, ZnSnP2/ZnS, and related chalcopyrite/zincblende heterointerfaces : Journal of Applied Physics, Vol.114, No.4, pp.043718-1 - 043718-6, 2013
36. Y. Hinuma, F. Oba, and I. Tanaka : Valence band offsets at zinc-blende heterointerfaces with misfit dislocations: A first-principles study : Physical Review B, Vol.88, No.7, pp.075319-1 - 075319-8, 2013
37. R. Ishikawa, A. R. Lupini, F. Oba, S. D. Findlay, N. Shibata, T. Taniguchi, K. Watanabe, H. Hayashi, T. Sakai, I. Tanaka, Y. Ikuhara, and S. J. Pennycook : Atomic structure of luminescent centers in high-efficiency Ce-doped w-AlN single crystal : Scientific Reports, Vol.4, pp.3778-1 - 3778-5, 2014
38. A. Gruneis, G. Kresse, Y. Hinuma, and F. Oba : Ionization potentials of solids: The importance of vertex corrections : Physical Review Letters, Vol.112, No. 9, pp.096401-1 - 096401-5, 2014

【5500 プロセス工学】

39. T. Yamamoto, Y. Takagi, Y. Okano, and S. Dost : Numerical investigation for the thermocapillary flow observed in "Saturday-Morning-Science" conducted by Dr. Donald Pettit : proceedings of ASGSR/ISPS-5, 2013
40. T. Yamamoto, Y. Takagi, Y. Okano and S. Dost : A three-dimensional simulation of thermocapillary flow in a circular thin liquid film under zero gravity : Trans. JSASS Aerospace Tech. Japan,Vol.12, pp.Pe_19~Pe_27, 2014
41. 山本卓也、高木洋平、岡野泰則：液膜内 thermocapillary 対流の流動方向に対する数値計算： 第 51 回日本伝熱シンポジウム講演要旨集, pp. A343, 2014
42. T. Yamamoto, Y. Takagi, Y. Okano and S. Dost : A three-dimensional simulation of thermocapillary flow in a circular thin liquid film under zero gravity : Trans. JSASS Aerospace Tech. Japan, Vol.12, pp.Pe_19 - Pe_27, 2014
43. T. Yamamoto, Y. Takagi, Y. Okano and S. Dost : Numerical investigation for the direction of thermocapillary flow in a cooled circular water film : Proc. IMA7 (7th Conference of the International Marangoni Association), Vol.126, 2014
44. 山本卓也、高木洋平、岡野泰則：液膜内 Marangoni 対流の流動方向に対する液膜形状と渦の関係性の影響に関する数値解析：化学工

学会第 46 回秋季大会講演要旨集, pp.O117, 2014

【6800　薬学】

45. Takumi Azuma, Yusuke Kobayashi, Ken Sakata, Takahiro Sasamori, Norihiro Tokitoh, Yoshiji Takemoto : Synthesis and Characterization of Binary-Complex Models of Ureas and 1,3-Dicarbonyl Compounds: Deeper Insights into Reaction Mechanisms Using Snap-Shot Structural Analysis : The Journal of Organic Chemistry, Vol.79, No.4, pp.1805 – 1817, 2014

46. Yusuke Kobayashi, Tsubasa Inokuma, Yoshiji Takemoto : Development of Innovative Hydrogen-Bond-Donor Catalysts Based on Heterocyclic Scaffolds and Their Applications to Asymmetric Reactions : Journal of Synthetic Organic Chemistry, Japan, Vol.71, No.5, pp.491 – 502, 2013

大型計算機システム利用負担金

別表1　スーパーコンピュータシステム

コース	タイプ	セット	利用負担額	提供サービス					
				システム	バッチ	システム資源	経過時間（時間）	ディスク（GB）	利用者番号
エントリ	−	基本	12,600 円/年	B	共有	最大1ノード相当((16コア、64GBメモリ)×1)	1	60	−
パーソナル	タイプA	基本	100,000 円/年	A	共有	最大4ノード相当((32コア、64GBメモリ)×4)	168	1,000	−
	タイプB	基本	100,000 円/年	B	共有	最大4ノード相当((16コア、64GBメモリ)×4)	168	1,000	−
	タイプC	基本	100,000 円/年	C	共有	最大2ソケット相当((8コア、384GBメモリ)×2)	168	1,000	−
	タイプD	基本	100,000 円/年	D	共有	最大2ノード相当((28コア、64GBメモリ)×2)	168	1,000	−
	タイプE	基本	100,000 円/年	E	共有	最大2ノード相当((10コア、32GBメモリ + 1MIC)×2)	168	1,000	−
	タイプG	基本	100,000 円/年	B (GPU)	共有	最大2ノード相当((16コア、64GBメモリ + 1GPU)×2)	168	1,000	−
グループ	タイプA1	最小	200,000 円/年	A	優先	4ノード((32コア、64GBメモリ)×4)	336	8,000	8
		追加単位	200,000 円/年			4ノード((32コア、64GBメモリ)×4)	−	8,000	8
	タイプA2	最小	240,000 円/年		準優先	8ノード((32コア、64GBメモリ)×8)	336	9,600	16
		追加単位	120,000 円/年			4ノード((32コア、64GBメモリ)×4)	−	4,800	8
	タイプA3	最小	600,000 円/年		占有	8ノード((32コア、64GBメモリ)×8)	336	16,000	16
		追加単位	300,000 円/年			4ノード((32コア、64GBメモリ)×4)	−	8,000	8
	タイプB1	最小	250,000 円/年	B	優先	4ノード((16コア、64GBメモリ)×4)	336	8,000	8
		追加単位	250,000 円/年			4ノード((16コア、64GBメモリ)×4)	−	8,000	8
	タイプB2	最小	300,000 円/年		準優先	8ノード((16コア、64GBメモリ)×8)	336	9,600	16
		追加単位	150,000 円/年			4ノード((16コア、64GBメモリ)×4)	−	4,800	8
	タイプB3	最小	750,000 円/年		占有	8ノード((16コア、64GBメモリ)×8)	336	16,000	16
		追加単位	375,000 円/年			4ノード((16コア、64GBメモリ)×4)	−	8,000	8
	タイプC1	最小	400,000 円/年	C	優先	4ソケット((8コア、384GBメモリ)×4)	336	8,000	16
		追加単位	200,000 円/年			2ソケット((8コア、384GBメモリ)×2)	−	4,000	8
	タイプC2	最小	240,000 円/年		準優先	4ソケット((8コア、384GBメモリ)×4)	336	4,800	16
		追加単位	120,000 円/年			2ソケット((8コア、384GBメモリ)×2)	−	2,400	8
	タイプD1	最小	300,000 円/年	D	優先	4ノード((28コア、64GBメモリ)×4)	336	8,000	8
		追加単位	150,000 円/年			2ノード((28コア、64GBメモリ)×2)	−	4,000	4
	タイプD2	最小	360,000 円/年		準優先	8ノード((28コア、64GBメモリ)×8)	336	9,600	16
		追加単位	90,000 円/年			2ノード((28コア、64GBメモリ)×2)	−	2,400	4
	タイプD3	最小	900,000 円/年		占有	8ノード((28コア、64GBメモリ)×8)	336	16,000	16
		追加単位	450,000 円/年			4ノード((28コア、64GBメモリ)×4)	−	8,000	8
	タイプE1	最小	280,000 円/年	E	優先	4ノード((10コア、32GBメモリ + 1MIC)×4)	336	8,000	8
		追加単位	140,000 円/年			2ノード((10コア、32GBメモリ + 1MIC)×2)	−	4,000	4
	タイプE2	最小	336,000 円/年		準優先	8ノード((10コア、32GBメモリ + 1MIC)×8)	336	9,600	16
		追加単位	84,000 円/年			2ノード((10コア、32GBメモリ + 1MIC)×2)	−	2,400	4
	タイプE3	最小	840,000 円/年		占有	8ノード((10コア、32GBメモリ + 1MIC)×8)	336	16,000	16
		追加単位	420,000 円/年			4ノード((10コア、32GBメモリ + 1MIC)×4)	−	8,000	8
	タイプG1	最小	250,000 円/年	B (GPU)	優先	2ノード((16コア、64GBメモリ + 1GPU)×2)	336	4,000	8
		追加単位	250,000 円/年			2ノード((16コア、64GBメモリ + 1GPU)×2)	−	4,000	8
大規模ジョブ	タイプA	最小	20,000 円/週(7日)	A	占有	8ノード((32コア、64GBメモリ)×8)	−	−	−
		追加単位	5,000 円/週(7日)			2ノード((32コア、64GBメモリ)×2)	−	−	−
	タイプB	最小	24,000 円/週(7日)	B	占有	8ノード((16コア、64GBメモリ)×8)	−	−	−
		追加単位	6,000 円/週(7日)			2ノード((16コア、64GBメモリ)×2)	−	−	−
	タイプC	最小	20,000 円/週(7日)	C	占有	4ソケット((8コア、384GBメモリ)×4)	−	−	−
		追加単位	10,000 円/週(7日)			2ソケット((8コア、384GBメモリ)×2)	−	−	−
	タイプD	最小	30,000 円/週(7日)	D	占有	8ノード((28コア、64GBメモリ)×8)	−	−	−
		追加単位	7,500 円/週(7日)			2ノード((28コア、64GBメモリ)×2)	−	−	−
	タイプE	最小	28,000 円/週(7日)	E	占有	8ノード((10コア、32GBメモリ + 1MIC)×8)	−	−	−
		追加単位	7,000 円/週(7日)			2ノード((10コア、32GBメモリ + 1MIC)×2)	−	−	−
	タイプG	最小	24,000 円/週(7日)	B (GPU)	占有	4ノード((16コア、64GBメモリ + 1GPU)×4)	−	−	−
		追加単位	12,000 円/週(7日)			2ノード((16コア、64GBメモリ + 1GPU)×2)	−	−	−
専用クラスタ	—	最小	750,000 円/年	B	−	8ノード((16コア、64GBメモリ)×8)	−	16,000	16
		追加単位	375,000 円/年			4ノード((16コア、64GBメモリ)×4)	−	8,000	8
ライセンスサービス			20,000 円/年	可視化ソフト(AVS,ENVI/IDL)およびプリポストウェアの1ライセンスにつき					

備考

1. 利用負担額は、年度単位で算定している。また、総額表示である。
2. 大型計算機システムの全ての利用者は、上記表のサービスの他、次のサービスを受けることができる。
   1） 大判プリンタサービス
   2） その他、大型計算機システムが提供するサービス、機器の利用
3. 上記表の大規模ジョブコース、ライセンスサービスの申請には、大型計算機システムの利用者であることが必要である。
4. 「共有」: 当該カテゴリのユーザ間で一定の計算資源を共有するベストエフォートのスケジューリングを行う。
   「準優先」: 定常稼働状況において記載値（以上）の計算資源が確保されるように優先スケジューリングを行う。
   また、稼働状況によらず記載値の1/4の計算資源が確保されることを保証する。
   「優先」: 定常稼働状況において記載値（以上）の計算資源が確保されるように優先スケジューリングを行う。
   また、稼働状況によらず記載値の1/2の計算資源が確保されることを保証する。
   「占有」: 稼働状況によらず記載値（以上）の計算資源が確保されることを保証する。
5. ディスク容量はバックアップ領域（最大で総容量の1/2)を含む。

6. グループコース及び専用クラスタコースのシステム資源は、下記の負担額を支払うことにより増量することができる。
なお増量は各月1日に実施し、増量した資源は当該年度末までの期間にわたって利用されるものとする。

コース	タイプ	追加負担金額（増量単位あたり）	システム資源増量単位	ディスク増量（GB）
グループ	タイプA1	20,000 円/月	4ノード((32コア、64GBメモリ)×4)	8,000
	タイプA2	12,000 円/月	4ノード((32コア、64GBメモリ)×4)	4,800
	タイプA3	30,000 円/月	4ノード((32コア、64GBメモリ)×4)	8,000
	タイプB1	25,000 円/月	4ノード((16コア、64GBメモリ)×4)	8,000
	タイプB2	15,000 円/月	4ノード((16コア、64GBメモリ)×4)	4,800
	タイプB3	37,500 円/月	4ノード((16コア、64GBメモリ)×4)	8,000
	タイプC1	20,000 円/月	2ソケット((8コア、384GBメモリ)×2)	4,000
	タイプC2	12,000 円/月	2ソケット((8コア、384GBメモリ)×2)	2,400
	タイプD1	15,000 円/月	2ノード((28コア、64GBメモリ)×2)	4,000
	タイプD2	9,000 円/月	2ノード((28コア、64GBメモリ)×2)	2,400
	タイプD3	45,000 円/月	4ノード((28コア、64GBメモリ)×4)	8,000
	タイプE1	14,000 円/月	2ノード((10コア、32GBメモリ + 1MIC)×2)	4,000
	タイプE2	8,400 円/月	2ノード((10コア、32GBメモリ + 1MIC)×2)	2,400
	タイプE3	42,000 円/月	4ノード((10コア、32GBメモリ + 1MIC)×4)	8,000
	タイプG1	25,000 円/月	2ノード((16コア、64GBメモリ + 1GPU)×2)	4,000
専用クラスタ	—	37,500 円/月	4ノード((16コア、64GBメモリ)×4)	8,000

7. グループコース及び専用クラスタコースを通年でなく利用する場合には、下記の負担額を支払うものとする。
ただし、利用期間は当該年度内に限るものとする。

利用期間			3ヶ月	6ヶ月	9ヶ月
グループコース	タイプA1	最小	80,000 円	120,000 円	180,000 円
		追加単位	80,000 円	120,000 円	180,000 円
	タイプA2	最小	96,000 円	144,000 円	216,000 円
		追加単位	48,000 円	72,000 円	108,000 円
	タイプA3	最小	240,000 円	360,000 円	540,000 円
		追加単位	120,000 円	180,000 円	270,000 円
	タイプB1	最小	100,000 円	150,000 円	225,000 円
		追加単位	100,000 円	150,000 円	225,000 円
	タイプB2	最小	120,000 円	180,000 円	270,000 円
		追加単位	60,000 円	90,000 円	135,000 円
	タイプB3	最小	300,000 円	450,000 円	675,000 円
		追加単位	150,000 円	225,000 円	337,500 円
	タイプC1	最小	160,000 円	240,000 円	360,000 円
		追加単位	80,000 円	120,000 円	180,000 円
	タイプC2	最小	96,000 円	144,000 円	216,000 円
		追加単位	48,000 円	72,000 円	108,000 円
	タイプD1	最小	75,000 円	150,000 円	225,000 円
		追加単位	37,500 円	75,000 円	112,500 円
	タイプD2	最小	90,000 円	180,000 円	270,000 円
		追加単位	22,500 円	45,000 円	67,500 円
	タイプD3	最小	225,000 円	450,000 円	675,000 円
		追加単位	112,500 円	225,000 円	337,500 円
	タイプE1	最小	70,000 円	140,000 円	210,000 円
		追加単位	35,000 円	70,000 円	105,000 円
	タイプE2	最小	84,000 円	168,000 円	252,000 円
		追加単位	21,000 円	42,000 円	63,000 円
	タイプE3	最小	210,000 円	420,000 円	630,000 円
		追加単位	105,000 円	210,000 円	315,000 円
	タイプG1	最小	100,000 円	150,000 円	225,000 円
		追加単位	100,000 円	150,000 円	225,000 円
専用クラスタ コース	—	最小	300,000 円	450,000 円	675,000 円
		追加単位	150,000 円	225,000 円	337,500 円

8. グループコース及び専用クラスタコースの利用者番号は利用者あたり年額5,000円を負担することで追加できる。

9. 機関・部局定額制度

他機関又は学内における部局(『国立大学法人京都大学の組織に関する規程』第3章第2節から第11節で定める組織をいう。)の組織が、その組織単位でグループコースサービス(年間)の利用を申請する場合、料金表(年間)に掲載額の1.5倍を利用負担金とする。なお、利用負担金額が150万円未満の場合は100人、150万円を超える場合は、150万円毎に100人までの利用者を認める。

10. スパコン連携サービス

学術情報メディアセンターのスーパーコンピュータシステムと密な連携により、学内における部局の組織が計算サーバ等を設置する場合、下記の負担額を支払うものとする。

冷却方式	利用負担額	利用負担額算定単位
水冷	10,300 円/月	水冷冷却方式の計算サーバ等の定格電力 1kWにつき
空冷	12,900 円/月	空冷冷却方式の計算サーバ等の定格電力 1kWにつき

別表2（汎用コンピュータシステム）

区分	利用負担額	単位
VMホスティングサービス	72,000円／年	1仮想マシンにつき
ホームページサービス	6,000円／年	1ドメイン名につき
ストリーミングサービス	6,000円／年	1申請につき

備考

１．利用負担額は、総額表示である。

２．上記表の汎用コンピュータシステムのサービスを利用するためには、大型計算機システムの利用者であることが必要である。

３．VM ホスティングサービスにおいて、下記の負担額を支払うことにより CPU、メモリ、ディスクを増量することができる。

区分	利用負担額	単位
CPU増量	18,000円／年	2コアにつき（最大8コアまで）
メモリ増量	18,000円／年	8GBにつき（最大64GBまで）
ディスク増量	18,000円／年	200GBにつき（最大1,000GBまで）

４．VM ホスティングサービスにおいて VMware を用いる場合は、下記の負担額を支払うことにより VMware の利用及び CPU、メモリ、ディスクを増量することができる。ただし、システム資源が限られているためサービスの提供を限定することがある。

区分	利用負担額	単位
VMware利用	72,000円／年	1仮想マシンにつき
CPU増量	36,000円／年	2コアにつき（最大8コアまで）
メモリ増量	36,000円／年	8GBにつき（最大64GBまで）
ディスク増量	18,000円／年	200GBにつき（最大1,000GBまで）

５．ホームページサービス及びストリーミングサービスにおいて、下記の負担額を支払うことにより公開スペースの上限を拡大することができる。

区分	利用負担額
公開スペース20GBプラン	3,000円／年
公開スペース50GBプラン	9,000円／年

６．利用負担額は、当該年度（4月から翌年3月まで）の利用に対して年額として算定するが、年度途中から利用を開始する場合には月数に応じて減額する。

別表3　スーパーコンピュータシステム（民間機関利用）

システム	システム資源	経過時間（時間）	ディスク（GB）	利用者番号	利用負担額
A	8ノード(32コア、64GBメモリ)×8)	336	9,600	16	960,000 円/年
	12ノード(32コア、64GBメモリ)×12)	336	14,400	24	1,440,000 円/年
	16ノード(32コア、64GBメモリ)×16)	336	19,200	32	1,920,000 円/年
B	8ノード(16コア、64GBメモリ)×8)	336	9,600	16	1,200,000 円/年
	12ノード(16コア、64GBメモリ)×12)	336	14,400	24	1,800,000 円/年
	16ノード(16コア、64GBメモリ)×16)	336	19,200	32	2,400,000 円/年
D	8ノード(28コア、64GBメモリ)×8)	336	9,600	16	1,440,000 円/年
	12ノード(28コア、64GBメモリ)×12)	336	14,400	24	2,160,000 円/年
	16ノード(28コア、64GBメモリ)×16)	336	19,200	32	2,880,000 円/年
E	8ノード(10コア、32GBメモリ＋1MIC)×8)	336	9,600	16	1,344,000 円/年
	12ノード(10コア、32GBメモリ＋1MIC)×12)	336	14,400	24	2,016,000 円/年
	16ノード(10コア、32GBメモリ＋1MIC)×16)	336	19,200	32	2,688,000 円/年

備考

1. 利用負担額は、年度単位で算定している。また、総額表示である。
2. ディスク容量はバックアップ領域(最大で総容量の1/2)を含む。
3. 通年でなく利用する場合には、下記の負担額を支払うものとする。
   ただし、利用期間は当該年度内に限るものとする。

システム	システム資源	利用期間		
		3ヶ月	6ヶ月	9ヶ月
A	8ノード	240,000 円	480,000 円	720,000 円
	12ノード	360,000 円	720,000 円	1,080,000 円
	16ノード	480,000 円	960,000 円	1,440,000 円
B	8ノード	300,000 円	600,000 円	900,000 円
	12ノード	450,000 円	900,000 円	1,350,000 円
	16ノード	600,000 円	1,200,000 円	1,800,000 円
D	8ノード	360,000 円	720,000 円	1,080,000 円
	12ノード	540,000 円	1,080,000 円	1,620,000 円
	16ノード	720,000 円	1,440,000 円	2,160,000 円
E	8ノード	336,000 円	672,000 円	1,008,000 円
	12ノード	504,000 円	1,008,000 円	1,512,000 円
	16ノード	672,000 円	1,344,000 円	2,016,000 円

― サービス利用のための資料一覧 ―

1. スーパーコンピュータシステム・ホスト一覧

- システム A：camphor.kudpc.kyoto-u.ac.jp
- システム B・C：laurel. kudpc.kyoto-u.ac.jp
  - システム B（SAS 利用時）：sas.kudpc.kyoto-u.ac.jp
- システム D：magnolia.kudpc.kyoto-u.ac.jp
- システム E：camellia. kudpc.kyoto-u.ac.jp

※ ホストへの接続は SSH(Secure SHell) 鍵認証のみ、パスワード認証は不可

2. 問い合わせ先 & リンク集

- 情報環境機構のホームページ
  http://www.iimc.kyoto-u.ac.jp/

- 学術情報メディアセンターのホームページ
  http://www.media.kyoto-u.ac.jp/

- スーパーコンピュータシステムに関する問い合わせ先
  - 利用申請などに関する問い合わせ先

  【情報環境支援センター】
  E-mail：zenkoku-kyo@media.kyoto-u.ac.jp / Tel：075-753-7424
  URL: http://www.iimc.kyoto-u.ac.jp/ja/services/comp/
  - システムの利用など技術的な問い合わせ先

  【スーパーコンピューティング掛】
  E-mail：consult@kudpc.kyoto-u.ac.jp / Tel：075-753-7426
  URL: http://www.iimc.kyoto-u.ac.jp/ja/services/comp/contact.html

- ホームページ・ホスティングサービスに関する問い合わせ先

  【クラウドコンピューティング掛】
  E-mail：whs-qa@media.kyoto-u.ac.jp / Tel：075-753-7494
  URL: http://www.iimc.kyoto-u.ac.jp/ja/services/whs/

- コンテンツ作成支援サービスに関する問い合わせ先

  【コンテンツ作成室】
  E-mail：cpt@media.kyoto-u.ac.jp / Tel：075-753-9012
  URL: http://www.iimc.kyoto-u.ac.jp/ja/services/content/

編 集 後 記

京都にやってきて半年近くが経ち、京都の地理や交通にもやっと慣れてきたような気がする。今まで関西には縁が無かったが、今後は京都だけで無く周辺にも足をのばしてみたいと思う。縁が無いと言えば雪が積もる土地にも縁が無かったが、今年は正月に京都で大雪の洗礼にあった。おかげで出かけられなかったが、逆に家でゆっくりでき、久しぶりに寝正月を満喫できた。

逍遥人

先日、リアル脱出ゲームと呼ばれるイベントに参加しました。リアル脱出ゲームは、１時間の制限時間内に謎（パズルやクイズのようなもの）を解き明かし、閉じ込められた場所から脱出するというものです。と言っても実際閉じ込められているわけではなく、そういう体です。会場に着くと参加者は１チーム６人に振り分けられ、いよいよゲームは始まります。解いたと思ったら新たな謎、迫る制限時間の中、チーム一丸となり協力した結果・・・残念ながら脱出失敗！！でした。失敗の原因は「チーム内で情報共有が出来ていなかった」「作業分担がうまく出来ていなかった」「思ったこと、感じたことをもっと自由に発言すればよかった」「謎がとけた喜びから細かい見落としがあった」などがあったと思います。あれれ？これって普段の仕事でも起こりうることなのでは？

ペーペー

京都大学学術情報メディアセンター全国共同利用版広報　Vol. 13, No. 2

2015 年　2 月　 25 日発行

編集者　京都大学学術情報メディアセンター
　　全国共同利用版広報編集部会
発行者　〒606-8501　京都市左京区吉田本町
　　京都大学学術情報メディアセンター
　　Academic Center for Computing and Media Studies
　　Kyoto University
　　Tel. 075-753-7400
　　http://www.media.kyoto-u.ac.jp/
印刷所　〒616-8102 京都市右京区太秦森ヶ東町 21-10
　　株式会社エヌジーピー

目次

Cover : Designed by TANI Takushi / T&T Design Lab. (February,2015)