



ISSN 1349-1113  
JAXA-RR-06-004

# 宇宙航空研究開発機構研究開発報告

## JAXA Research and Development Report

---

非構造格子Euler/Navier-StokesソルバJTASの計算性能最適化

坂下 雅秀, 松尾 裕一, 村山 光宏

2006年11月

宇宙航空研究開発機構  
Japan Aerospace Exploration Agency



# 非構造格子 Euler/Navier-Stokes ソルバ JTAS の計算性能最適化

## 目 次

概要 (Abstract) .....	1
1. はじめに .....	3
2. JTAS 概要 .....	4
2.1 支配方程式 .....	4
2.2 空間方向の離散化 .....	4
2.3 時間方向の離散化 .....	6
2.4 ベクトル化 .....	8
2.4.1 LU-SGS 法の非構造格子へのベクトル化を考慮した適用 .....	9
2.4.2 数値流束の評価及び総和演算のベクトル化 .....	10
2.5 並列化 .....	12
2.6 その他 .....	13
2.6.1 主なサブルーチンの計算内容 .....	13
2.6.2 乱流モデル .....	13
3. 計算性能最適化チューニングのための変更 .....	17
3.1 スカラ版変更内容 .....	17
3.1.1 LU-SGS 法における節点番号の付け替え .....	17
3.1.2 色分けの削除 .....	17
3.1.3 勾配の計算における配列定義順序の連続化 .....	18
3.2 スレッド版変更内容 .....	18
4. 計算性能測定条件 .....	23
4.1 ハードウェア .....	23
4.2 ソフトウェア .....	24
4.3 JTAS 実行条件 .....	25
4.3.1 初期条件 .....	25
4.3.2 格子点数 .....	26
5. スカラ版計算性能測定結果及び評価 .....	27
5.1 計算結果の確認 .....	27
5.2 測定方法 .....	28
5.2.1 タイマの挿入 .....	28
5.2.2 プロファイラによる解析 .....	28
5.3 測定結果及び評価 .....	33
5.3.1 全体の測定結果及び評価 .....	33
5.3.2 測定区間 “SPVCOR” における測定結果及び評価 .....	38
5.3.3 測定区間 “SPDRF 1” における測定結果及び評価 .....	44
5.3.4 測定区間 “LIMITER” における測定結果及び評価 .....	49
5.3.5 測定区間 “LU-SGS” における測定結果及び評価 .....	49
5.3.6 測定区間 “COMMUNICATION” における測定結果及び評価 .....	55

6. スレッド版計算性能測定結果及び評価 .....	65
6.1 計算結果の確認 .....	65
6.2 測定方法 .....	65
6.3 測定結果及び評価 .....	65
6.3.1 全体の測定結果及び評価 .....	66
6.3.2 測定区間“SPVCOR”における測定結果及び評価 .....	72
6.3.3 測定区間“SPDRF1”における測定結果及び評価 .....	72
6.3.4 測定区間“LIMITER”における測定結果及び評価 .....	80
6.3.5 測定区間“LU-SGS”における測定結果及び評価 .....	82
7. 今後の課題 .....	92
APPENDIX A スカラ版チューニング詳細 .....	93
A.1 勾配 $\nabla q_i$ の計算 .....	95
A.2 Venkatakrishnanの制限関数 $\Psi_i$ の計算 .....	104
A.3 LU-SGS法ソルバ部分 .....	106
A.4 HLLW法による数値流束ベクトル $\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n}$ の計算 .....	127
A.5 タイマの挿入 .....	128
APPENDIX B スレッド版チューニング詳細 .....	137
B.1 スレッド版変更内容 .....	138
B.2 LU-SGS法における色分け削除の効果確認のための変更 .....	153
APPENDIX C 要素と辺及び節点の関係 .....	161
APPENDIX D 実行スクリプト .....	164
APPENDIX E メモリ使用量 .....	176
参考文献 .....	176

# 非構造格子 Euler/Navier-Stokes ソルバ JTAS の計算性能最適化\*

坂下 雅秀\*1, 松尾 裕一\*1, 村山 光宏\*2

## Performance Optimization of an Unstructured Grid Euler/Navier-Stokes Solver JTAS

Masahide Sakashita\*1, Yuichi Matsuo\*1 and Mitsuhiro Murayama\*2

**Abstract** : A scalar version of the three-dimensional hybrid-unstructured-grid finite-volume method Euler/Navier-Stokes solver JTAS (JAXA Tohoku-university Aerodynamic Simulation code) is developed to improve the scalar computing performance by optimizing the memory access pattern because the original JTAS code was developed for a vector system. The efficiency of this optimization is confirmed on a large scale SMP cluster system consisting of Fujitsu PRIM-POWER HPC2500. The speedup ratio is more than 1.8 from the performance measurement with practical data. In addition, a thread parallel version of JTAS is also developed to confirm the possibility of the unstructured grid CFD solver. The thread parallel execution of the JTAS original version can be done naturally. But, under the 2 processes and 8 threads hybrid execution condition, the thread parallel speedup ratio is only about 5, which is less than 70% of the theoretical speedup for the time-integration part. Generally speaking, a scalar version with some recursive references cannot be efficiently executed by the thread parallel. Thus, a thread version without recursive references is developed to get a higher performance under the thread parallel execution. The thread parallel acceleration ratio for the time-integration part of this version is about 6.2, which is more than 70% of the theoretical speedup. Then, the efficiency of thread parallel optimization is confirmed. However, total elapsed time is longer than a scalar version under execution used same number of CPUs, and only 1.5 times of performance gain as compared with a JTAS original version and this. Through the study, the understanding to the features of the unstructured flow solver like JTAS is deepened and the coding strategy for the performance speedup to JTAS is obtained.

**Key Words** : Super computing, SMP, MPI, Thread parallel, Unstructured grid, Finite-volume method, Navier-Stokes solver, CFD solver, LU-SGS method,

---

\* 平成18年9月29日受付

\*1 情報・計算工学センター 計算機運用・利用技術チーム  
(JAXA's Engineering Digital Innovation Center, Computing Resource Management Team)

\*2 航空プログラム 国産旅客機チーム  
(Aviation Program Group, Civil Transport Team)

## 概 要

3次元ハイブリッド非構造格子有限体積法 Euler/Navier-Stokes ソルバ JTAS (JAXA Tohoku university Aerodynamic Simulation code) は、元来、ベクトル計算機用に開発されたものであるが、ここでは主にメモリアクセスの効率化によるスカラ性能の向上を図る変更を加えることにより、JTAS スカラ版を開発した。このスカラ版について、大規模 SMP (Symmetric Multiple Processor) クラスタ計算機である JAXA スーパーコンピュータシステム (富士通製 PRIME-POWER HPC2500) を使用してテストデータによる計算性能測定を行い、JTAS オリジナル版と比較して約1.8から1.9倍の性能が得られることを確認した。次に、スレッド並列化の可能性を確認する目的で、JTAS スレッド並列版を開発した。JTAS オリジナル版は、スレッド並列による実行が可能ではあるが、時間積分計算の部分で、8スレッド実行によるスレッド並列加速率が約5倍と、理論値(8倍)の7割を下回る性能しか得られていなかった。また、スカラ版は再帰参照を含むため、スレッド並列化は困難であった。そこで、再帰参照を含まないスレッド並列化可能な JTAS スレッド並列版を開発した。このスレッド版について、同様にテストデータによる性能測定を行った結果、時間積分計算部分で、8スレッド実行によるスレッド並列化加速率が約6.2倍と、理論値の7割を越える性能が得られることが確認された。一方で、全体の実行時間(経過時間)については、同じCPU数を使用した場合のスカラ版の性能に及ばず、JTAS オリジナルに比べて約1.5倍の性能向上にとどまることも確認された。本研究を通じて、JTAS のような非構造格子ソルバの特性に対する知見と計算性能向上に対するコーディング指針が得られた。

## 1. はじめに

本報告書は、3次元非構造格子有限体積法 Euler/Navier-Stokes ソルバ JTAS (JAXA Tohoku university Aerodynamic Simulation code) について、主にスカラ性能向上を図ることを目的に変更を加え、性能測定を行って効果を確認した結果について報告するものである。

現在、宇宙航空研究開発機構 (JAXA) では、次世代超音速機技術の基礎研究として小型超音速実験機 (NEXST-1) に関するプロジェクトが進められている [1] [2]。このプロジェクトにおいては、複雑な形状の回りにおける剥離や再付着を伴う複雑な流れ場に対する CFD (Computational Fluid Dynamics) 解析技術が求められている。このような解析には非構造格子法 (Unstructured Grid Method) が良く用いられる。非構造格子法は、

- (1) 構造格子に比べて格子生成が比較的容易
- (2) 流れ場の重要な場所に格子を細分化して局所的重点的に配置し精度向上を図ることが可能
- (3) 最適設計時における形状変化に伴う計算格子の修正が容易

といった特徴を持つ。JAXA においては、非構造格子ソルバとして、主に JTAS が用いられている [3] [4] [5]。JTAS は、東北大学で開発された TAS (Tohoku university Aerodynamic Simulation code) [6] をもとに JAXA に導入されている CeNSS (Central Numerical Simulation System) と呼ばれる大規模 SMP (Symmetric Multiple Processor) クラスタシステム (富士通製 PRIMEPOWER HPC2500) に適合するように若干の変更が加えられたコードであり、オリジナルの TAS と区別する意味で JTAS と呼ばれている。

JTAS は、CeNSS 向けに変更が加えられているものの、その変更は配列の次元入れ替え等限定的なものであり、CeNSS の性能を十分有効に活用出来ていないという問題があった。そこで本報告中で示す作業において、CeNSS に対する適合性を高め、効率の良い解析を可能にすることを目的として、より内容に踏み込んだ変更を行った。また、テストデータを用いた性能測定を実施し、変更による性能向上を確認した。なお、変更は、スカラ性能向上のためメモリアクセスの効率化を主眼においた再帰参照ループを含むスカラ版と、FORTRAN コンパイラによる自動並列化のより効率的な活用を促進することを主眼においた再帰参照ループを含まないスレッド版の二つに分けて行った。

本報告書では、次章「2. JTAS 概要」において JTAS の計算手法の概要について簡単に示す。次に、「3. 変更内容」において、スカラ版及びスレッド版のそれぞれについて JTAS に加えた変更の具体的な内容を説明する。さらに「4. 計算条件」において、性能測定に使用したテストデータの概要を示す。「5. スカラ版測定結果及び評価」において、スカラ版による性能測定の結果を示し、オリジナル版 JTAS との性能比較等、測定結果に対する若干の評価を行う。同様に、「6. スレッド版測定結果及び評価」において、スレッド版による性能測定の結果を示し、スカラ版同様、若干の評価を行う。最後に、「7. 今後の課題」において、今後取り組むべき課題について触れる。

## 2. JTAS 概要

ここでは、3次元ハイブリッド非構造格子有限体積法 Euler/Navier-Stokes ソルバ JTAS が採用している数値解法及びベクトル化と並列化の手法について、その概略を示す。より詳しくは、参考文献 [7] を参照されたい。

### 2.1. 支配方程式

性能評価に使用した支配方程式は Euler 方程式であるが、ここでは三次元圧縮性 Navier-Stokes 方程式を元に、JTAS における数値解法の概要を示す。Navier-Stokes 方程式は、その粘性流束ベクトルを 0 と置く（非粘性流体）か、或いは粘性流束ベクトルにおいて剪断応力による寄与を 0 と置く（完全流体）ことにより Euler 方程式が得られるので、数値解法は共通である。

今、三次元非定常圧縮性 Navier-Stokes 方程式の積分形表示は、

$$\frac{\partial}{\partial t} \int_{\Omega} \mathbf{Q} dV + \int_{\partial\Omega} [\mathbf{F}(\mathbf{Q}) - \mathbf{G}(\mathbf{Q})] \cdot \mathbf{n} ds = 0 \quad (2.1)$$

で与えられる。ここに、 $\mathbf{Q} = [\rho, \rho u, \rho v, \rho w, e]^T$  は保存量 (Conservative variables) ベクトルであり、 $\rho$  は密度、 $\mathbf{u} = [u, v, w]^T$  は流速、 $e$  はエネルギーである。また、 $\mathbf{F}(\mathbf{Q})$  及び  $\mathbf{G}(\mathbf{Q})$  はそれぞれ非粘性流束ベクトル及び粘性流束ベクトルである。

式 (2.1) の連立方程式は、以下の完全気体に関する状態方程式によって閉じられる。

$$p = (\gamma - 1) \left[ e - \frac{1}{2} \rho (u^2 + v^2 + w^2) \right] \quad (2.2)$$

但し、 $\gamma$  は比熱比であり、理想気体の仮定のもと定数 1.4 を与える。

### 2.2. 空間方向の離散化

(2.1) 式は、各物理量の無次元化を行った後、セル節点有限体積法によって空間方向が離散化される。有限体積法における検査体積は、図 2.1 に示したように、各格子節点周りに要素の中心 A、要素表面の中心 B、D 及び要素を構成する辺の中点 C を結んで出来る面を境界とする多面体として定義される。このような検査体積の

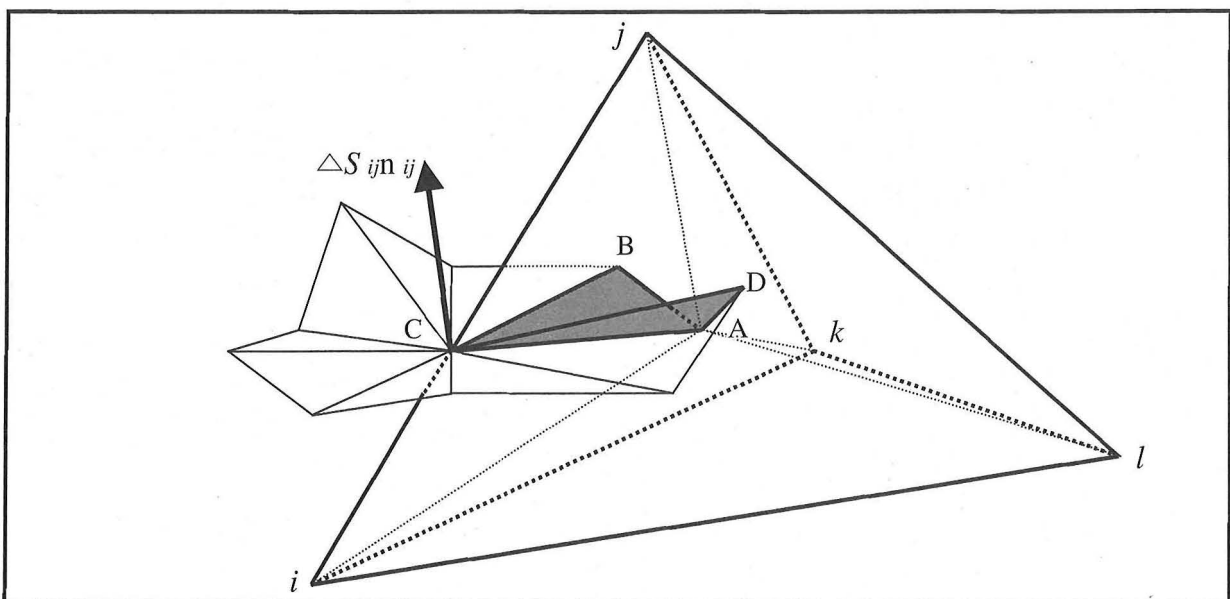


図 2.1 三角錐要素における検査体積境界面



定義方法は、非重合二重格子 (non-overlapping dual cell) と呼ばれる。

この空間離散化によって (2.3) 式が得られる。

$$\frac{\partial \mathbf{Q}_i}{\partial t} = -\frac{1}{V_i} \left[ \sum_{j(i)} \Delta S_{ij} \mathbf{F}(\mathbf{Q})_{ij} \cdot \mathbf{n}_{ij} - \frac{1}{\text{Re}} \sum_{j(i)} \Delta S_{ij} \mathbf{G}(\mathbf{Q})_{ij} \cdot \mathbf{n}_{ij} \right] \quad (2.3)$$

ここに、 $V_i$  は節点  $i$  の周りの検査体積の体積であり、 $\Delta S_{ij}$  と  $\mathbf{n}_{ij}$  はそれぞれ節点  $i$  とそれに隣接する節点  $j$  との間の検査体積表面の面積及びその単位法線ベクトル (点  $i$  から見て外向きが正) である。また、 $\Sigma_{j(i)}$  は節点  $i$  の周りの多面体検査体積において、それを構成する全ての面についての総和を取ることを意味する。Re はレイノルズ数である。隣接する二節点  $ij$  間の検査体積境界面は、複数の平面から構成されていることに注意されたい。従って、面ベクトル  $\Delta S_{ij} \mathbf{n}_{ij}$  は、節点  $i$  及び  $j$  を両端とする辺を共有する全ての要素について、要素毎に二つ存在する面ベクトルの全てを足し合わせるによって求める必要がある。

(2.3) 式の右辺第一項の流束を評価するためには、近似リーマン解法を用いたスキームが適用される。JTAS では、HLLEW (Harter-Lax-Van Leer-Einfeldt-Wada) 法 [8] が用いられている。この時、流束  $\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n}$  は、(2.4) 式で与えられる。

$$\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n} = \frac{1}{2} \left[ \mathbf{f}(\mathbf{Q}_L, \mathbf{n}) + \mathbf{f}(\mathbf{Q}_R, \mathbf{n}) - \left| \mathbf{A}(\tilde{\mathbf{Q}}, \mathbf{n}) \right| (\mathbf{Q}_R - \mathbf{Q}_L) \right] \quad (2.4)$$

但し、 $\mathbf{f}(\mathbf{Q}, \mathbf{n}) \equiv \mathbf{F}(\mathbf{Q}) \cdot \mathbf{n}$  は数値流束ベクトルであり、 $\mathbf{A} \equiv \partial \mathbf{f} / \partial \mathbf{Q}$  はヤコビ行列である。また、添え字  $L$  及び  $R$  は検査体積境界面の両側における物理量を、 $\tilde{\cdot}$  (チルダ) は Roe 平均を表す。この数値流束ベクトルは、後に示す (2.25) 式において“残差ベクトル”  $\mathbf{R}_i$  が含む流束ベクトルとして使用される。JTAS において、この“残差ベクトル”  $\mathbf{R}_i$  はファイル名“necns2.f”に含まれるサブルーチン“SPVCOR”で計算されている。

ここで、検査体積の両面  $L$  及び  $R$  における物理量  $\mathbf{Q}_L$  及び  $\mathbf{Q}_R$  を高次精度で求めるために、基礎物理変数 (Primitive variables)  $\mathbf{q} = [\rho, u, v, w, p]^T$  に対して、その勾配  $\nabla \mathbf{q}_i$  を用いた (2.5) 式で与えられる区分的一次関数を導入する。

$$\mathbf{q}(\mathbf{r}) = \mathbf{q}_i + \Psi_i \nabla \mathbf{q}_i \cdot (\mathbf{r} - \mathbf{r}_i) \quad (2.5)$$

ここに、 $\mathbf{r}$  は位置ベクトルであり  $\Psi$  は流束制限関数である。

勾配  $\nabla \mathbf{q}_i$  は節点  $i$  を共有する全ての要素について、その勾配  $\nabla \mathbf{q}_e$  を体積平均することにより求める。即ち、

$$\nabla \mathbf{q}_i = \frac{\sum_{e(i)} \nabla \mathbf{q}_e \cdot V_e}{\sum_{e(i)} V_e} \quad (2.6)$$

によって与えられる。但し、 $e(i)$  は節点  $i$  を共有する全ての要素を表し、 $V_e$  は要素  $e$  の体積である。要素  $e$  の勾配  $\nabla \mathbf{q}_e$  は、三角錐の場合、以下のようにして求めることができる。

$$\nabla \mathbf{q}_e = \frac{\partial \mathbf{q}}{\partial x} \mathbf{i} + \frac{\partial \mathbf{q}}{\partial y} \mathbf{j} + \frac{\partial \mathbf{q}}{\partial z} \mathbf{k} \quad (2.7)$$

$$\frac{\partial \mathbf{q}}{\partial x} = \frac{1}{d} \begin{vmatrix} \Delta \mathbf{q}_{ji} & \Delta y_{ji} & \Delta z_{ji} \\ \Delta \mathbf{q}_{ki} & \Delta y_{ki} & \Delta z_{ki} \\ \Delta \mathbf{q}_{li} & \Delta y_{li} & \Delta z_{li} \end{vmatrix}$$

$$\frac{\partial \mathbf{q}}{\partial y} = \frac{1}{d} \begin{vmatrix} \Delta x_{ji} & \Delta \mathbf{q}_{ji} & \Delta z_{ji} \\ \Delta x_{ki} & \Delta \mathbf{q}_{ki} & \Delta z_{ki} \\ \Delta x_{li} & \Delta \mathbf{q}_{li} & \Delta z_{li} \end{vmatrix} \quad (2.8)$$

$$\frac{\partial \mathbf{q}}{\partial z} = \frac{1}{d} \begin{vmatrix} \Delta x_{ji} & \Delta y_{ji} & \Delta \mathbf{q}_{ji} \\ \Delta x_{ki} & \Delta y_{ki} & \Delta \mathbf{q}_{ki} \\ \Delta x_{li} & \Delta y_{li} & \Delta \mathbf{q}_{li} \end{vmatrix}$$

但し,  $i, j, k$  及び  $l$  は三角錐の四頂点を表し (図2.1参照),  $\mathbf{i}, \mathbf{j}$  及び  $\mathbf{k}$  はそれぞれ,  $x, y$  及び  $z$  方向の単位ベクトルである. また,

$$\begin{aligned} \Delta \mathbf{q}_{ji} &= \mathbf{q}_j - \mathbf{q}_i, \Delta \mathbf{q}_{ki} = \mathbf{q}_k - \mathbf{q}_i, \Delta \mathbf{q}_{li} = \mathbf{q}_l - \mathbf{q}_i \\ \Delta x_{ji} &= x_j - x_i, \Delta x_{ki} = x_k - x_i, \Delta x_{li} = x_l - x_i \\ \Delta y_{ji} &= y_j - y_i, \Delta y_{ki} = y_k - y_i, \Delta y_{li} = y_l - y_i \\ \Delta z_{ji} &= z_j - z_i, \Delta z_{ki} = z_k - z_i, \Delta z_{li} = z_l - z_i \end{aligned} \quad (2.9)$$

$$d = \begin{vmatrix} \Delta x_{ji} & \Delta y_{ji} & \Delta z_{ji} \\ \Delta x_{ki} & \Delta y_{ki} & \Delta z_{ki} \\ \Delta x_{li} & \Delta y_{li} & \Delta z_{li} \end{vmatrix}$$

である. 三角柱及び四角錐の場合, これをそれぞれ三つ及び二つの三角錐に分割することで, 同様に計算することが可能である. 分割方法の詳細については「APPENDIX C. 要素と辺及び節点の関係」を参照されたい. JTASにおいて, 勾配の計算はファイル名 “der. f” に含まれるサブルーチン “SPDRF1\_tet” (三角錐要素の勾配の計算), サブルーチン “SPDRF1\_pri” (三角柱要素の勾配の計算) 及びサブルーチン “SPDRF1\_pyr” (四角錐要素の勾配の計算) で行われる. これらの, サブルーチンでは, 粘性流束ベクトル  $\mathbf{G}(\mathbf{Q})$  に含まれる温度及び速度の勾配の計算も行われる. これらの勾配は, 辺  $ed$  を共有する全ての要素  $e(ed)$  について, その勾配の平均として計算される. 即ち,  $\mathbf{q}$  を温度及び速度とした場合に, 辺における勾配  $\nabla \mathbf{q}_{ed}$  は,

$$\nabla \mathbf{q}_{ed} = \frac{\sum_{e(ed)} \nabla \mathbf{q}_e \cdot V_e}{\sum_{e(ed)} V_e} \quad (2.10)$$

で計算される.

また, 制限関数  $\Psi_i$  ( $0 \leq \Psi_i \leq 1$ ) は, 高次精度差分において non-physical peak を除去し, スキームの単調性を保持するための関数である. JTAS では, 制限関数として Venkatakrishnan の制限関数 [9] が用いられている. Venkatakrishnan の制限関数は, 以下の式で定義される.

$$\Psi_i = \min_{j(i)} \begin{cases} \frac{\Delta_{\max}^2 + \varepsilon^2 + 2\Delta_{\max} \Delta_{\min}}{\Delta_{\max}^2 + 2\Delta_{\min}^2 + \Delta_{\max} \Delta_{\min} + \varepsilon^2}, & \text{if } \Delta_{\min} > 0 \\ \frac{\Delta_{\min}^2 + \varepsilon^2 + 2\Delta_{\min} \Delta_{\max}}{\Delta_{\min}^2 + 2\Delta_{\max}^2 + \Delta_{\min} \Delta_{\max} + \varepsilon^2}, & \text{if } \Delta_{\min} < 0 \end{cases} \quad (2.11)$$

ここに,

$$\begin{aligned} \Delta_{\min} &= \nabla \mathbf{q}_i \cdot (\mathbf{r} - \mathbf{r}_i) / 2 \\ \Delta_{\max} &= \max_{j(i)} (\mathbf{q}_j - \mathbf{q}_i) \\ \Delta_{\min} &= \min_{j(i)} (\mathbf{q}_j - \mathbf{q}_i) \\ \varepsilon^2 &= (K \Delta l)^3, K = \text{const} \end{aligned} \quad (2.12)$$

であり,  $K$  は通常0.1から0.3程度とする. また,  $\Delta l$  は, 要素の平均長さである. JTASにおいて,  $\Delta_{\max}$  及び  $\Delta_{\min}$  の計算はファイル名 “der. f” に含まれるサブルーチン “LIMITER1” で, 制限関数  $\Psi_i$  の計算は同じくファイル名 “der. f” に含まれるサブルーチン “LIMITER2” で行っている.

### 2.3. 時間方向の離散化

時間方向の離散化は Euler 陰解法により行われ, 非構造格子に拡張された LU-SGS (Lower-Upper Symmetric Gauss-Sidel) 陰解法 [10] [11] により時間積分が行われる.

今, (2.3) 式に Euler 陰解法を適用して時間方向の離散化を行うと,

$$\Delta \mathbf{Q}^n = -\frac{\Delta t}{V_i} \left[ \sum_{j(i)} \Delta S_{ij} \mathbf{f}(\mathbf{Q})_{ij}^{n+1} - \sum_{j(i)} \Delta S_{ij} \mathbf{g}(\mathbf{Q})_{ij}^{n+1} \right] \quad (2.13)$$

$$\Delta \mathbf{Q}^n \equiv \mathbf{Q}_i^{n+1} - \mathbf{Q}_i^n$$

$$\mathbf{f}(\mathbf{Q})_{ij} = \mathbf{F}(\mathbf{Q})_{ij} \cdot \mathbf{n}_{ij}$$

$$\mathbf{g}(\mathbf{Q})_{ij} = \frac{1}{\text{Re}} \mathbf{G}(\mathbf{Q})_{ij} \cdot \mathbf{n}_{ij}$$

を得る．ここに， $\Delta t$  は時間刻み幅を，上付き添え字  $n$  は時間ステップを表す．(2.13) 式において，数値流束ベクトル  $\mathbf{f}$  及び  $\mathbf{g}$  は保存量ベクトル  $\mathbf{Q}$  の非線形関数であるため，これらの項をテイラー展開し二次以降を無視することにより線形化を行う．この時，ヤコビ行列を  $\mathbf{A} \equiv \partial \mathbf{f} / \partial \mathbf{Q}$  及び  $\mathbf{M} \equiv \partial \mathbf{g} / \partial \mathbf{Q}$  とすれば，数値流束ベクトル  $\mathbf{f}$  及び  $\mathbf{g}$  は，

$$\mathbf{f}_{ij}^{n+1} = \mathbf{f}(\mathbf{Q})_{ij}^n + \mathbf{A}_i^n \Delta \mathbf{Q}_i^n \quad (2.14)$$

$$\mathbf{g}_{ij}^{n+1} = \mathbf{g}(\mathbf{Q})_{ij}^n + \mathbf{M}_i^n \Delta \mathbf{Q}_i^n$$

となる．さらに，数値流束ベクトル  $\mathbf{f}$  において，その保存量ベクトル  $\mathbf{Q}$  への依存性について，風上差分的な取り扱いをすることにすれば，

$$\mathbf{f}_{ij}^{n+1} = \mathbf{f}_{ij}^n + \left( \mathbf{A}^+ \Delta \mathbf{Q}_i + \mathbf{A}^- \Delta \mathbf{Q}_i \right)^n \quad (2.15)$$

$$\mathbf{g}_{ij}^{n+1} = \mathbf{g}_{ij}^n + \left( \mathbf{M}^+ \Delta \mathbf{Q}_i + \mathbf{M}^- \Delta \mathbf{Q}_i \right)^n$$

となる．ここに， $\mathbf{A}^\pm$  はヤコビ行列  $\mathbf{A}$  を正の固有値を持つ行列と，負の固有値を持つ行列に分割したものである．

(2.15) 式を (2.13) 式に代入し，ヤコビ行列  $\mathbf{M}$  の掛かる項を無視すれば，

$$\left[ \frac{V_i}{\Delta t} \mathbf{I} + \sum_{j(i)} \Delta S_{ij} (\mathbf{A}_i^+)^n \right] \Delta \mathbf{Q}_i^n + \sum_{j(i)} \Delta S_{ij} (\mathbf{A}_j^-)^n \Delta \mathbf{Q}_j^n = \mathbf{R}_i^n \quad (2.16)$$

$$\mathbf{R}_i^n = - \sum_{j(i)} \Delta S_{ij} (\mathbf{f} - \mathbf{g})_{ij}^n$$

を得る．因みに，Euler 陽解法を適用した場合，(2.16) 式は，

$$V_i \frac{\Delta \mathbf{Q}_i^n}{\Delta t} = \mathbf{R}_i^n \quad (2.17)$$

となり  $\mathbf{R}_i$  が残差 (residual) に相当することがわかる．

(2.16) 式に，LU-SGS 陰解法を適用するためには，節点  $i$  に隣接する節点の集合  $j(i)$  を下三角形要素 (lower) に属する節点の集合  $j \in L(i)$  と上三角形要素 (upper) に属する節点の集合  $j \in U(i)$  に分割する．分割の具体的な方法については，「2.4 ベクトル化」を参照されたい．この分割により，

$$\left[ \frac{V_i}{\Delta t} \mathbf{I} + \sum_{j(i)} \Delta S_{ij} (\mathbf{A}_i^+)^n \right] \Delta \mathbf{Q}_i^n + \sum_{j \in L(i)} \Delta S_{ij} (\mathbf{A}_j^-)^n \Delta \mathbf{Q}_j^n + \sum_{j \in U(i)} \Delta S_{ij} (\mathbf{A}_j^-)^n \Delta \mathbf{Q}_j^n = \mathbf{R}_i^n \quad (2.18)$$

を得るので，これに LU-SGS 法を適用すれば，

$$\Delta \mathbf{Q}_i^* = \Delta \mathbf{Q}_i^{n-1}$$

$$\text{前進スイープ: } \Delta \mathbf{Q}_i^* = \frac{1}{\mathbf{D}_i^n} \left[ \mathbf{R}_i^n - \sum_{j \in L(i)} \Delta S_{ij} (\mathbf{A}_j^-)^n \Delta \mathbf{Q}_j^* \right]$$

$$\text{退進スイープ: } \Delta \mathbf{Q}_i^n = \Delta \mathbf{Q}_i^* + \frac{1}{\mathbf{D}_i^n} \sum_{j \in U(i)} \Delta S_{ij} (\mathbf{A}_j^-)^n \Delta \mathbf{Q}_j^n \quad (2.19)$$

$$\mathbf{R}_i^n = - \sum_{j(i)} \Delta S_{ij} (\mathbf{f} - \mathbf{g})_{ij}^n$$

$$\mathbf{D}_i^n = \frac{V_i}{\Delta t} \mathbf{I} + \sum_{j(i)} \Delta S_{ij} (\mathbf{A}_i^+)^n$$

となる．

ここで，行列  $\mathbf{D}$  は一般には対角行列でないため，その逆行列  $1/\mathbf{D}$  を求める演算量は少なくない．よって，行列  $\mathbf{D}$  を対角行列で近似し演算量の削減を図る．JTAS では，この対角化に Jameson-Turkel によるヤコビ行列の分割法 [12] が用いられている．

今,  $\rho_A$  をヤコビ行列  $\mathbf{A}$  のスペクトル半径  $|U| + a$  に, 粘性ベクトルのヤコビ行列  $\mathbf{M}$  に関する項を無視した代わりの効果を付け加えたものとする。即ち,

$$\rho_A = \chi(|U| + a) + 2 \frac{\mu}{\text{Re} \rho h} \quad (2.20)$$

とする。但し,  $U$  は, 検査体積表面の法線方向速度  $U = \mathbf{u} \cdot \mathbf{n}$  である。また,  $a$  は音速,  $\mu$  は粘性係数であり,  $\chi$  は経験定数 1.01 である。この時, 分割されたヤコビ行列  $\mathbf{A}^\pm$  は,

$$\mathbf{A}^\pm = \frac{\mathbf{A} \pm \rho_A \mathbf{I}}{2} \quad (2.21)$$

として, 与えられる。このことと, ヤコビ行列の性質 [13],

$$\sum_{j(i)} \Delta S_{ij} \mathbf{A} = 0 \quad (2.22)$$

より, 行列  $\mathbf{D}$  は対角化され,

$$\mathbf{D}_i^n = \left[ \frac{V_i}{\Delta t} + \sum_{j(i)} \Delta S_{ij} (\rho_a)_j^n \right] \mathbf{I} \quad (2.23)$$

となる。

さらに, ヤコビ行列  $\mathbf{A}$  の計算を省くため,

$$\mathbf{A} \Delta \mathbf{Q} \cong \Delta \mathbf{f} \equiv \mathbf{f}(\mathbf{Q} + \Delta \mathbf{Q}) - \mathbf{f}(\mathbf{Q}) \quad (2.24)$$

なる近似を行う。

以上の近似により, 最終的に (2.19) 式は,

$$\begin{aligned} \Delta \mathbf{Q}_i^* &= \Delta \mathbf{Q}_i^{n-1} \\ \text{前進スイープ: } \Delta \mathbf{Q}_i^* &= \frac{1}{\mathbf{D}_i^n} \left[ \mathbf{R}_i^n - \frac{1}{2} \sum_{j \in L(i)} \Delta S_{ij} \left( \Delta \mathbf{f}_{ij}^* - (\rho_a)_j \Delta \mathbf{Q}_j^* \right) \right] \\ \text{退進スイープ: } \Delta \mathbf{Q}_i^n &= \Delta \mathbf{Q}_i^* + \frac{1}{2 \mathbf{D}_i^n} \sum_{j \in U(i)} \Delta S_{ij} \left( \Delta \mathbf{f}_{ij}^n - (\rho_a)_j \Delta \mathbf{Q}_j^n \right) \\ \mathbf{R}_j^n &= - \sum_{j(i)} \Delta S_{ij} (\mathbf{f} - \mathbf{g})_{ij}^n \\ \mathbf{D}_i^n &= \left[ \frac{V_i}{\Delta t} + \sum_{j(i)} \Delta S_{ij} (\rho_a)_j^n \right] \mathbf{I} \\ \Delta \mathbf{f}_{ij}^* &= \mathbf{f}(\mathbf{Q} + \Delta \mathbf{Q}^*)_{ij}^n - \mathbf{f}(\mathbf{Q})_{ij}^n \end{aligned} \quad (2.25)$$

となる。ここで, HLLEW 法により評価されるのは, “残差ベクトル”  $\mathbf{R}_i$  が含む流束ベクトルについてのみであることに注意されたい。

JTAS において, 対角行列  $\mathbf{D}$  の計算はファイル名 “lut2.f” に含まれるサブルーチン “DIAGONAL” で, 前進スイープの右辺に含まれる和を取る部分の計算は  $\Delta \mathbf{f}^*$  の評価を含めて同じくサブルーチン “SIGMALEFT” で, 前進スイープの計算はサブルーチン “SWEEPPL” で, 後退スイープの右辺に含まれる和を取る部分の計算は  $\Delta \mathbf{f}^n$  の評価を含めサブルーチン “SIGMARIGHT” で, 後退スイープの計算はサブルーチン “SWEEPPR” で, それぞれ行っている。

LU-SGS 法の非構造格子法への適用方法については, 「2.4 ベクトル化」を参照されたい。

## 2.4. ベクトル化

ここでは, JTAS で採用されているベクトル化手法について解説する。まず, LU-SGS 法をベクトル化可能な形で非構造格子に適用する方法について説明した後, 数値流束ベクトルの評価や総和演算を効率良く行うために使用されているベクトル化手法について説明する。

### 2.4.1. LU-SGS 法の非構造格子へのベクトル化を考慮した適用

(2.21) 式を LU-SGS 法によって解くためには、節点  $i$  に隣接する節点の集合  $j(i)$  を下三角形要素に属する節点の集合  $j \in L(i)$  と上三角形要素に属する節点の集合  $j \in U(i)$  に分割する必要がある。構造格子では、三次元  $x$ ,  $y$  及び  $z$  方向の節点番号をそれぞれ  $i$ ,  $j$  及び  $k$  とし、 $i+j+k = \text{一定}$  となる超平面 (Hyper plane) 上に属する節点について同時に計算を行うことで、下三角形要素と上三角形要素への分割が可能であると共に、ベクトル計算が可能となる。一方、非構造格子では、格子線が存在しないため、別の方法で超平面を構成する必要がある。JTAS では、以下のように仮の面を構成した後、実際の面を構成するという二段階の過程を経ることにより、超平面を構成している [10] [11].

- (1) 適当な一節点に仮の超平面番号  $N_i=1$  を与える。
- (2) 「現在」の仮超平面番号を  $n_i=1$  とする。
- (3) 仮の超平面番号  $N_i=n_i$  を持つ超平面に隣接する節点の内、仮の超平面番号を持たない全ての節点に仮の超平面番号  $N_i=n_i+1$  を与える。
- (4) 「現在」の仮超平面番号を  $n_i=n_i+1$  とする。
- (5) 全ての節点に仮の超平面番号が与えられるまで、(3) 及び (4) を繰り返す。

以上により、仮の超平面が構成された。JTAS において、ここまでの処理はファイル名 “lut2.f” に含まれるサブルーチン “MARKPLANE” で行われる。ところで、この仮の超平面内には、お互いに隣接する節点が含まれており、このままではベクトル計算が出来ない。従って、この仮の超平面の中で色分けを行い、実際の超平面を構成することによりベクトル計算を可能なものとする。

即ち、

- (1) 超平面番号カウンタ  $m_{i-1} = 0$  とする。
- (2) 仮の超平面番号  $n_i = 1$  とする。
- (3) 仮の超平面番号  $N_i = n_i$  において、隣接する節点にそれぞれ別の色番号  $n_c (=1 \sim)$  を与える。
- (4) 各節点の超平面番号を  $N_p = m_{i-1} + n_c$  とする。
- (5)  $m_i = m_{i-1} + n_{c, \max}$  とする。但し、 $n_{c, \max}$  は仮の超平面  $N_i = n_i$  における色番号の最大値。
- (6) 仮の超平面番号  $n_i = n_i + 1$  とする。
- (7) 全ての仮の超平面について (3) から (6) を繰り返す。

以上により、同じ超平面番号を持つ節点については互いに隣接することがなく、ベクトル計算が可能な超平面が構成された。JTAS では、この処理をファイル名 “lut2.f” に含まれるサブルーチン “COLORPLANE” (上記 (3) の処理) 及びサブルーチン “MARKEDG11” (上記 (4) 及び (5) の処理) において行っている。

この超平面は、同時に LU-SGS 法における下三角形要素と上三角形要素への分離にもなっていることに注意されたい。今、ある節点  $i$  が超平面番号  $N_p(i)$  を持つとする。この時、隣接する節点  $j$  が持つ超平面番号  $N_p(j)$  が  $N_p(i) > N_p(j)$  であれば節点  $j$  は節点  $i$  の下三角形要素  $j \in L(i)$  であり、 $N_p(i) < N_p(j)$  であれば、上三角形要素  $j \in U(i)$  であるとすれば良い。この関係を改めてまとめれば、(2.26) 式となる。

$$\begin{aligned} j \in L(i) & \quad , \text{if } N_p(j) < N_p(i) \\ j \in U(i) & \quad , \text{if } N_p(j) > N_p(i) \end{aligned} \tag{2.26}$$

これにより、前進スイープでは超平面番号 1 から超平面番号最大の方向にスイープを行い、後退スイープでは逆

に、超平面番号最大から超平面番号 1 に向かってスイープを行うことで非構造格子に対する LU-SGS 法が実現される。

実際のプログラミング上、前進（後退）スイープは、ある超平面に属する節点  $i$  についてではなく、その超平面に属する節点  $i$  とその節点について下三角形（上三角形）要素である節点  $j \in L(i)$  ( $j \in U(i)$ ) を結ぶ辺  $ed$  について DO ループが構成される。節点  $i$  についてのループでは、その節点  $i$  を含む辺  $ed$ 、その辺の他端の節点  $j$  と二段階のインデックス参照になるのに対して、辺  $ed$  のループでは両端の節点  $i$  及び  $j$  を直接インデックス参照することが可能だからである。このため、以下の (2.27), (2.28) 式で与えられるような色番号  $m_c$  が辺に対して定義される。即ち、前進スイープでは、辺  $ed$  の両端の節点  $i$  及び  $j$  が持つ二つの超平面番号  $N_p(i)$  及び  $N_p(j)$  の内、大きい方の超平面番号がその辺の色番号  $m_c$  となり、後退スイープでは、小さい超平面番号が色番号  $m_c$  となる。JTAS において、辺  $ed$  に対する色番号の設定は、ファイル名 “lut2.f” に含まれるサブルーチン “MARKEDGES” で行われる。そして、同じ色番号  $m_c$  を持つ辺が同時に計算される。

(1) 前進スイープ

$$m_c = N_p(k)$$

$$k = \begin{cases} i, & \text{if } N_p(i) > N_p(j) \\ j, & \text{if } N_p(i) < N_p(j) \end{cases} \quad (2.27)$$

(2) 後退スイープ

$$m_c = N_p(k)$$

$$k = \begin{cases} i, & \text{if } N_p(i) < N_p(j) \\ j, & \text{if } N_p(i) > N_p(j) \end{cases} \quad (2.28)$$

以上により、非構造格子に適用可能であり、かつベクトル化可能な LU-SGS 法が構成された。但し、ここで注意しなければならないのは、以上のような超平面の構成によってベクトル化の障害となる再帰参照を回避できるのは、LU-SGS 法の最終形である (2.24) 式の両辺に含まれる  $\Delta Q^*$  (前進スイープの場合) 及び  $\Delta Q^*$  (後退スイープの場合) の関係においてのみ、ということである。

実際にプログラムを作成する際に、計算の効率化を考慮すると前進スイープ及び後退スイープの右辺に含まれる総和演算  $\Sigma$  の計算においても、ベクトル化を疎外する要因である再帰演算が問題となってくる。JTAS において、この問題にどのように対処しているのかについては、他の計算において表れる総和演算と共に、「2.4.2 数値流束ベクトルの評価及び総和演算のベクトル化」において説明する。

#### 2.4.2. 数値流束ベクトルの評価及び総和演算のベクトル化

ここでは、数値流束ベクトルの評価や総和演算を効率良く行うために JTAS において使用されているベクトル化手法について説明する。

まず最初に、JTAS において数値流束ベクトルがどのように評価されているのかについて説明する。HLEW 法による数値流束ベクトル  $\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n}$  を与える (2.4) 式を再掲すれば、

$$\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n} = \frac{1}{2} \left[ \mathbf{f}(\mathbf{Q}_L, \mathbf{n}) + \mathbf{f}(\mathbf{Q}_R, \mathbf{n}) - \left| \mathbf{A}(\tilde{\mathbf{Q}}, \mathbf{n}) \right| (\mathbf{Q}_R - \mathbf{Q}_L) \right] \quad (2.4)$$

である。この流束ベクトルは、ある検査体積  $i$  において、それを構成する面の法線方向ベクトル  $\mathbf{n}$  に依存するため、各面ごとに求める必要がある。ところで、ある面は異なる二つの検査体積の境界を構成するのであるから、その面を通る流束はその二つの検査体積に対して同じ量でかつ符号が反対であるように寄与することになる。従って、ファイル名 “necns2.f” に含まれるサブルーチン “SPVCOR” で計算される “残差ベクトル”  $\mathbf{R}_i$

$$\mathbf{R}_i^n = - \sum_{j(i)} \Delta S_{ij} (\mathbf{f} - \mathbf{g})_{ij}^n$$

において、数値流束ベクトル  $\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n}$  を検査体積ごとにそれを構成する全ての面に対して計算することは、流束ベクトルを二重に計算することとなり効率的ではない。面ごとに計算すれば流束ベクトルの計算を最小限に抑えることが出来る。ところで、検査体積がその唯一含む節点の番号で指定されるのと同様に、面はそれが唯一含む辺の番号で指定されるので、実際のプログラムにおける“残差ベクトル”  $\mathbf{R}_i$  の計算は、辺 IE が含む両端の節点の番号 N1 及び N2 を保持する配列の名前を“NEDG2ND”とした場合、例えば以下ようになる。

```

DO 100 IE = 1, Nedges
  N1 = NEDG2ND(1, IE)
  N2 = NEDG2ND(2, IE)
  HLLW_FLUX = FLUX_FUNC( Some arguments required )
  Resid(N1) = Resid(N1) + HLLW_FLUX
  Resid(N2) = Resid(N2) - HELLW_FLUX
100 CONTINUE

```

リスト 2.1 “残差ベクトル”  $\mathbf{R}_i$  計算プログラム例

ここで、この DO ループは配列“Resid”に対して再帰参照になっていることに注意されたい。なぜならば、検査体積は複数の面から構成されているため、異なる面（辺）IE において同じ検査体積番号（節点番号）が N1 又は N2 に表れるからである。

JTAS では、この再帰参照を回避しベクトル計算を行うため、色分けによるグループ化の手法が用いられている。これは、ある検査体積（節点） $i$  に含まれる全ての面（辺） $e(i)$  は必ず別の色を持つように前もって色分けしておき、DO ループ内では同じ色を持った面（辺）のみを計算することにより、再帰参照を避ける方法である。この場合、実際のプログラムは、例えば以下のような二重ループになる。外側の DO ループが全ての色に対する処理のループであり、内側の DO ループがその内のある一つの色に対する処理を行う DO ループである。色分けを適切に行うことにより、内側のループで一度に処理される面（辺）は必ず異なる検査体積（節点）を構成するものとなり、再帰参照が回避されベクトル化される [14]。

```

DO 100 IC = 1, MAX_Colors
*VOCL LOOP, NOVREC
  DO 110 IE = 1, Num_edges(IC)
    N1 = NEDG2ND(1, IE, IC)
    N2 = NEDG2ND(2, IE, IC)
    HLLW_FLUX = FLUX_FUNC( Some arguments required )
    Resid(N1) = Resid(N1) + HLLW_FLUX
    Resid(N2) = Resid(N2) - HELLW_FLUX
  110 CONTINUE
100 CONTINUE

```

リスト 2.2 “残差ベクトル”  $\mathbf{R}_i$  計算ベクトル化例

ここで、DO110ループの前に挿入された行 “\*VOCL LOOP, NOVREC” は、直後の DO ループが再帰参照を含まないことをコンパイラに指示するための行である。この指定の仕方は、コンパイラを作成したメーカーによって異なり、この例は富士通製のコンパイラに指示をするためのものである。DO110のループでは、配列 “Resid” を定義参照するためのインデックス N1 及び N2 が配列 “NED2ND” によって定義されており、コンパイラはこれらの値が同じ値をとり得るか否かについて判断できないため、このような指示行を挿入しないとベクトル化されない。

同様のベクトル化手法は、勾配の計算、制限関数の計算、対角行列  $\mathbf{D}$  の計算及び LU-SGS 法における前進・後退スweepにおいても使用されている。但し、勾配の計算 (2.6) 式、

$$\nabla \mathbf{q}_i = \frac{\sum_{e(i)} \nabla \mathbf{q}_e \cdot V_e}{\sum_{e(i)} V_e} \quad (2.6)$$

において、勾配の計算は面 (辺) ごとではなく要素  $e$  毎に行われることが異なる。

## 2.5. 並列化

JTAS には、MPI を利用したプロセス並列化がなされている。ここでは、その並列化手法の概要について説明する。

プロセス並列化を行うためには、まず計算に先立って各プロセスに割り当てるために、格子空間を領域分割しなければならない。JTAS では、通常、この処理にミネソタ大学で開発された PDS (Public Domain Software) である METIS [15] をもとに三角錐のみならず三角柱や四角錐の混在したハイブリッド格子も取り扱えるように変更された前処理プログラムが利用される。そして、この分割された領域をそれぞれのプロセスが分担して計算する。この時、分割された領域の境界上では、同一の節点が複数の領域にまたがって存在することとなる。そこで、JTAS では、以下に示す手順によりそれらの節点における計算の分担を決め、異なる領域間においてその結果の送受信を可能としている [16] [17]。

- (1) 最も少ない節点数を持つ領域の節点  $i$  を “Sending Vertices”  $S_i$  と定義 (図2.2)
- (2) 他の領域の対応する節点  $i$  を “Receiving Vertices”  $R_j$  と定義 (図2.2)
- (3) “Receiving Vertices” に隣接する節点  $j$  を “Sending Vertices”  $S_j$  と定義 (図2.3)
- (4) 節点  $j$  に対応する節点を “Sending Vertices”  $S_i$  に隣接する節点  $j$  として追加 (図2.3)
- (5) (4) で追加した節点  $j$  を “Receiving Vertices”  $R_j$  と定義 (図2.3)

“Sending Vertices”  $S_j$  では、その節点における物理量や勾配等が計算され、対応する “Receiving Vertices”  $R_i$  を持つ他のプロセスにその情報が送信される。“Receiving Vertices”  $R_j$  では、対応する節点の情報を “Sending Vertices”  $S_j$  から受信する。最も節点数の少ない領域に計算を担当させることにより、負荷バランスが均質化することが期待できる。

尚、JTAS において LU-SGS 法は、この分割された領域ごとに適用され前進スweep終了時点及び後退スweep終了時点で、それぞれ計算結果の送受信が行われる。このため、計算に使用するプロセス数が変わると領域分割方法が変わることから、計算の途中経過は必ずしも一致しない。但し、十分な時間発展のもとに結果が定常状態に至れば、プロセス数によらず同じ結果が得られる [7]。



## 2.6. その他

### 2.6.1. 主なサブルーチンの計算内容

この節で説明した主な計算と、その計算を行っているサブルーチンの対応を表2.1に示す.

### 2.6.2. 乱流モデル

現在 JTAS において用意されている乱流モデルは, Goldberg-Ramakrishnan モデル [18] 及び Spallart-Allmaras モデル [19] である. これらのモデルを含めた性能評価は行わなかったため, ここでは特に触れない.

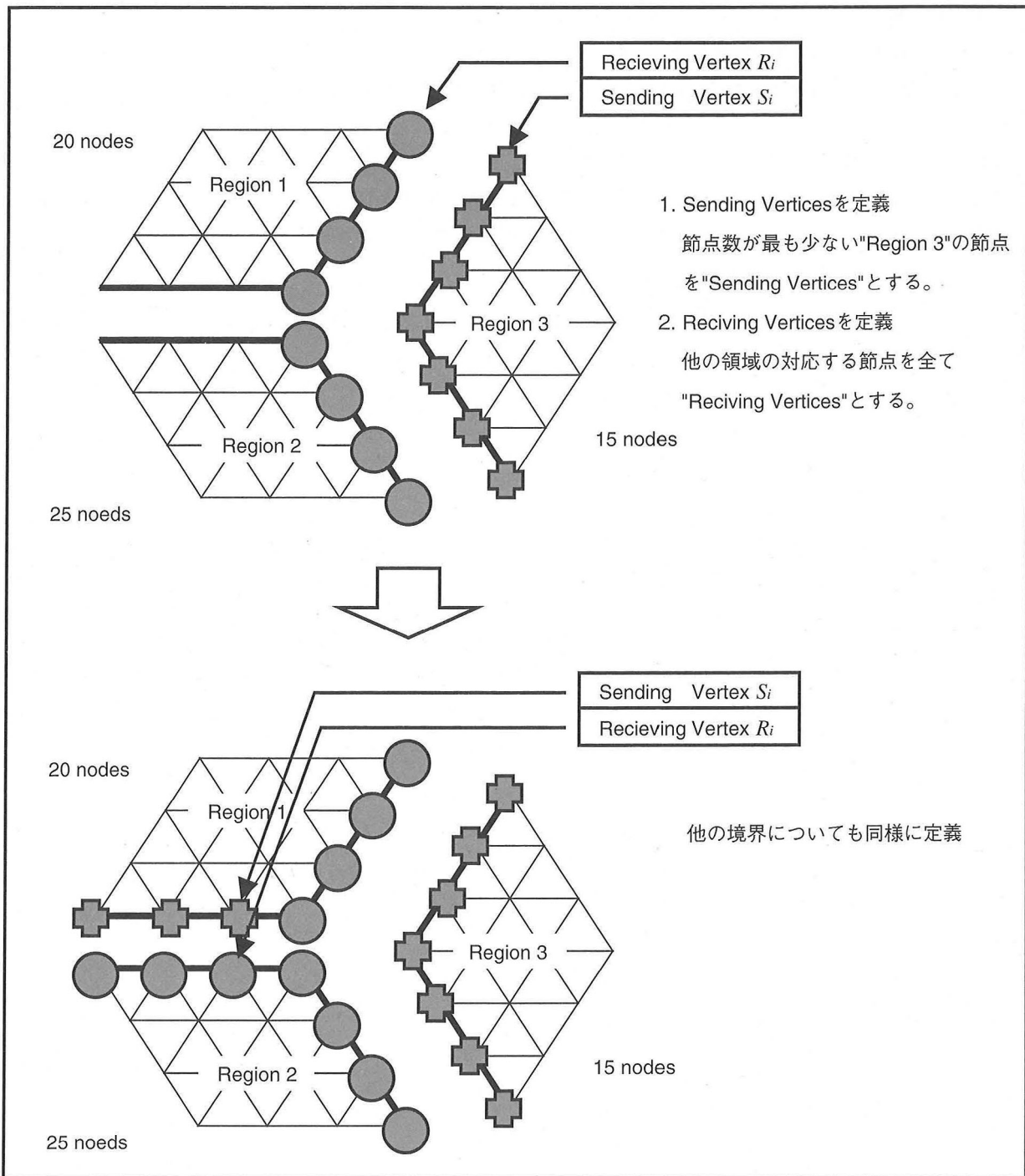


図 2.2 “Sending/Receiving Vertices” の定義

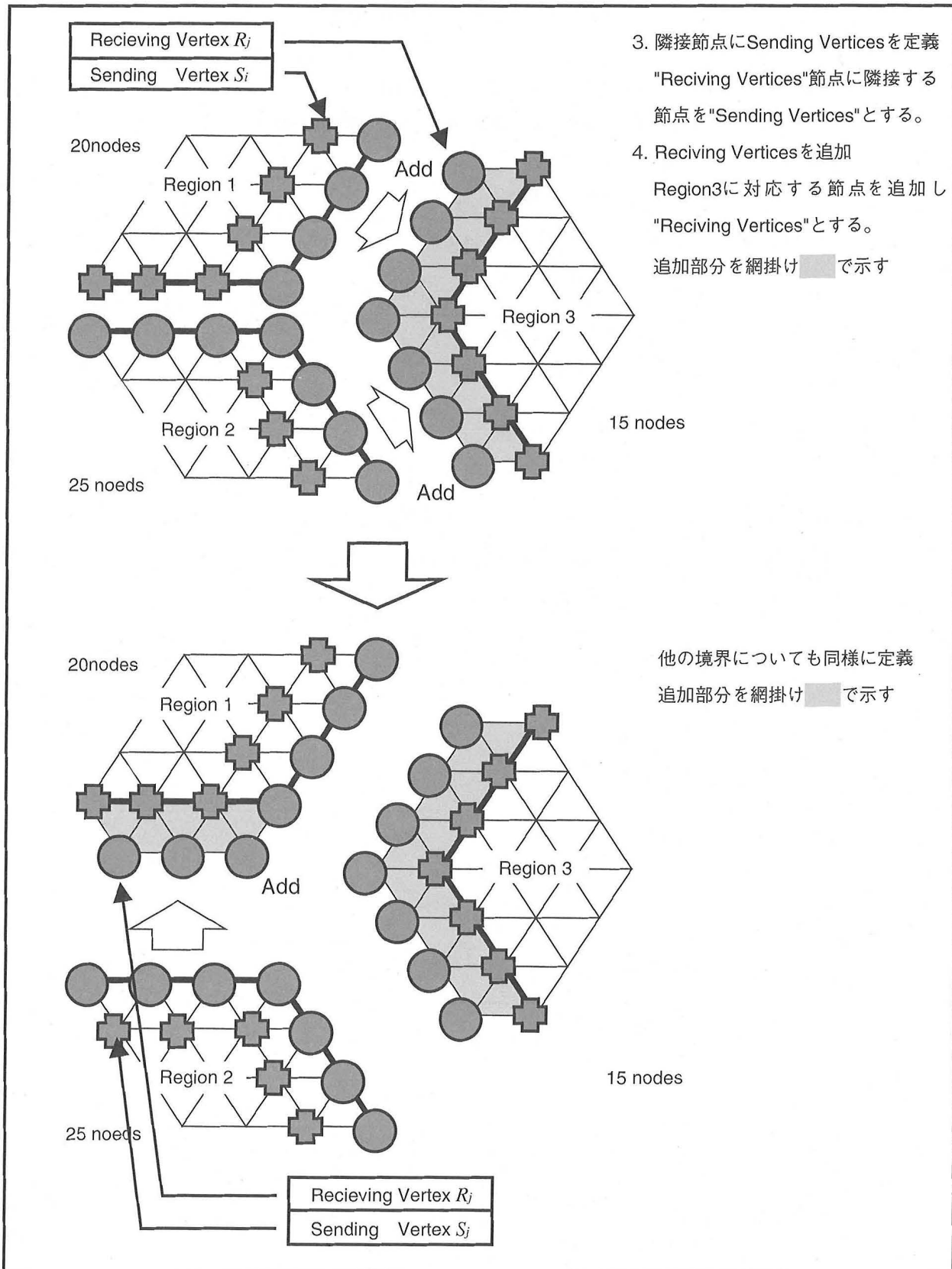


図 2.3 "Sending/Receiving Vertices" の追加

表 2.1 主なサブルーチンの処理内容

ファイル名	Subroutine 名	計 算 内 容
necns2.f	SPVCOR	<p>HLLEW 法による数値流束ベクトル <math>\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n}</math></p> $\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n} = \frac{1}{2} \left[ \mathbf{f}(\mathbf{Q}_L, \mathbf{n}) + \mathbf{f}(\mathbf{Q}_R, \mathbf{n}) - \left  \mathbf{A}(\tilde{\mathbf{Q}}, \mathbf{n}) \right  (\mathbf{Q}_R - \mathbf{Q}_L) \right] \quad (2.4) \text{ 式}$ <p>“残差ベクトル” <math>\mathbf{R}_i</math></p> $\mathbf{R}_j^n = - \sum_{j(i)} \Delta S_{ij} (\mathbf{f} - \mathbf{g})_{ij}^n \quad (2.25) \text{ 第 4 式}$
der.f	SPDRF1_tet	<p>三角錐要素における勾配</p> $\nabla \mathbf{q}_i = \frac{\sum_{e(i)} \nabla \mathbf{q}_e \cdot V_e}{\sum_{e(i)} V_e} \quad (2.6) \text{ 式}$
	SPDRF1_pri	<p>三角柱要素における勾配</p> $\nabla \mathbf{q}_i = \frac{\sum_{e(i)} \nabla \mathbf{q}_e \cdot V_e}{\sum_{e(i)} V_e} \quad (2.6) \text{ 式}$
	SPDRF1_pyr	<p>四角錐要素における勾配</p> $\nabla \mathbf{q}_i = \frac{\sum_{e(i)} \nabla \mathbf{q}_e \cdot V_e}{\sum_{e(i)} V_e} \quad (2.6) \text{ 式}$
	LIMITER1	$\Delta_{\max}, \Delta_{\min}$ <span style="float:right">(2.12) 第 2・3 式</span>
	LIMITER2	<p>Venkatakrishnan の制限関数 <math>\Psi_i</math> (<math>0 \leq \Psi_i \leq 1</math>)</p> $\Psi_i = \min_{j(i)} \begin{cases} \frac{\Delta_{\max}^2 + \varepsilon^2 + 2\Delta_- \Delta_{\max}}{\Delta_{\max}^2 + 2\Delta_-^2 + \Delta_{\max} \Delta_- + \varepsilon^2}, & \text{if } \Delta_- > 0 \\ \frac{\Delta_{\min}^2 + \varepsilon^2 + 2\Delta_- \Delta_{\min}}{\Delta_{\min}^2 + 2\Delta_-^2 + \Delta_{\min} \Delta_- + \varepsilon^2}, & \text{if } \Delta_- < 0 \end{cases} \quad (2.11) \text{ 式}$
lut2.f	DIAGONAL	<p>対角行列 <math>\mathbf{D}_i</math></p> $\mathbf{D}_i^n = \left[ \frac{V_i}{\Delta t} + \sum_{j(i)} \Delta S_{ij} (\rho_a)_j^n \right] \mathbf{I} \quad (2.25) \text{ 第 5 式}$
	SIGMALEFT	<p>前進スイープ (一部)</p> $\sum_{j \in L(i)} \Delta S_{ij} \left( \Delta \mathbf{f}_{ij}^* - (\rho_A)_j \Delta \mathbf{Q}_j^* \right) \quad (2.25) \text{ 第 2 式}$
	SWEEPL	<p>前進スイープ (一部) <math>\Delta \mathbf{Q}^*</math></p> $\Delta \mathbf{Q}_i^* = \frac{1}{\mathbf{D}_i^n} \left[ \mathbf{R}_j^n - \frac{1}{2} \sum_{j \in L(i)} \Delta S_{ij} \left( \Delta \mathbf{f}_{ij}^* - (\rho_A)_j \Delta \mathbf{Q}_j^* \right) \right] \quad (2.25) \text{ 第 2 式}$
	SIGMARIGHT	<p>後退スイープ (一部)</p> $\sum_{j \in U(i)} \Delta S_{ij} \left( \Delta \mathbf{f}_{ij}^n - (\rho_A)_j \Delta \mathbf{Q}_j^n \right) \quad (2.25) \text{ 第 3 式}$
	SWEEPR	<p>後退スイープ (一部) <math>\Delta \mathbf{Q}^n</math></p> $\Delta \mathbf{Q}_i^n = \Delta \mathbf{Q}_i^* + \frac{1}{2\mathbf{D}_i^n} \sum_{j \in U(i)} \Delta S_{ij} \left( \Delta \mathbf{f}_{ij}^n - (\rho_A)_j \Delta \mathbf{Q}_j^n \right) \quad (2.25) \text{ 第 3 式}$
	UPDATEQ	<p>時間ステップ <math>n+1</math> における保存量ベクトル <math>\mathbf{Q}^{n+1}</math></p> $\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n + \Delta \mathbf{Q}_i^n$

### 3. 計算性能最適化チューニングのための変更

ここでは、CeNSS に対する適合性を高めるために行った変更の内容について説明する。変更は、スカラ並列計算機を対象に再帰演算による計算手順を含み MPI によるプロセス並列による並列計算が可能なスカラ版と、再帰演算を含まず SMP クラスタにおけるスレッド並列と MPI によるプロセス並列とによるハイブリッド並列計算が可能なスレッド版に分けて行った。スレッド版については再帰参照を含まないため、オリジナルの JTAS 同様、ベクトル実行も可能である。以下に、それぞれの変更内容を示す。

#### 3.1. スカラ版変更内容

ここでは、スカラ並列計算機を対象に再帰演算による計算手順を含み MPI によるプロセス並列のみによる並列計算が可能なスカラ版について、JTAS に加えた変更内容を示す。変更内容は、以下の三点である。

- (1) LU-SGS 法における節点番号の付け替え
- (2) 色分けの削除
- (3) 勾配計算における配列定義順序の連続化

これらの変更は、いずれも配列の定義・参照パターンを整理し、メモリアクセスをより局所的にすることで、メモリアクセスにおけるミス率を軽減することを目的として行ったものである。以下、それぞれの変更内容について説明する。

##### 3.1.1. LU-SGS 法における節点番号の付け替え

「2.4.1 LU-SGS 法の実行効率化」で示した通り、JTAS では非構造格子に LU-SGS 法を適用するために超平面が構成されている。ところが、節点における各物理量等を配列に保存する際には、格子生成時に付されたオリジナルの節点の番号でインデックスされる場所に保存されている。このような方法は、或る一つの超平面内に存在する節点の番号が不連続であるため、効率的なメモリアクセスを疎外する要因となることが予想された。そこで、LU-SGS 法の計算を行う部分では、ハイパー面を考慮した節点番号の付け替えを行うこととし、新たな節点番号として超平面内でオリジナルの節点番号の昇順に連続な番号を与えた。この時、超平面  $N_p+1$  に含まれる節点の番号は、超平面  $N_p$  で与えた節点番号の続きから連続になるようにし、物理量等はこの超平面を考慮した新たな節点番号でインデックスされる場所に保存するように変更した。同時に、LU-SGS 法に関する部分（サブルーチン“LUSGS”及びそこから呼び出されるサブルーチン）以外では従来通りの番号付けとし、LU-SGS 法で必要となる保存量ベクトル等のデータは、従来の番号付けで保存されている配列から新たな番号付けで保存される配列に一旦コピーした後 LU-SGS 法の計算を行い、サブルーチン“UPDATEQ”で、新しい時間ステップでの値  $Q^{n+1}$  を計算する際に従来の番号付けの配列に戻す方法をとった。より詳細な変更内容については、「APPENDIX A.3 LU-SGS 法ソルバ部分」を参照されたい。

##### 3.1.2. 色分けの削除

先に、「2.4.2 数値流束ベクトルの評価及び総和演算のベクトル化」で示したように、JTAS ではベクトル計算機による実行に配慮して、色分けの手法により再帰演算を回避している。ところが、この色分けの手法は、同じ節点を含む辺については同時に計算しないようにするものであり、この結果、配列の定義・参照、即ちメモリアクセスのパターンが複雑となり、キャッシュミスの頻発による性能低下を引き起こすことが予想された。一方で、スカラ計算機においては、再帰演算が許容されるため、色分けを行わなければならない必然性はない。そこ

で、色分けの削除を行い、配列の定義・参照をより局所化し、キャッシュミスの低減を図った。即ち、オリジナルの JTAS ではリスト2.2で示したコーディングがなされていたものを、リスト2.1で示したコーディング例に変更した。より詳細な変更内容については、「APPENDIX A. スカラ版チューニング内容詳細」を参照されたい。

以下では、この変更によるコーディング自体を「色分けの削除」または「再帰参照を含む DO ループ」等と記述する。

### 3.1.3. 勾配計算における配列定義順序の連続化

「3.1.2 色分けの削除」で行った変更では、計算結果を保存するための配列定義順序は、完全には連続とはならない。そこで、勾配の計算について、これを連続としメモリキャッシュのミス率を低減するために、DO ループの分割を行った。勾配の値は、節点（検査体積）ごと及び辺（検査体積境界面）ごとに必要となるので、DO ループを節点及び辺の DO ループとすれば、計算結果の保存について、配列のアクセス順序は連続とすることが出来る。一方で勾配は、先ず要素ごとに計算され、節点または辺を共有する要素全てについての体積平均が取られる。ひとつの要素は、複数の節点及び辺を含むために、節点ごと及び辺ごとに DO ループを構成すると同じ要素の勾配の計算を何度もすることになり、効率的とは言えない。そこで、計算の効率化と配列定義の順序の連続化が同時に満たされるように、要素ごとに計算した勾配の値を一度作業用配列に保存し、その後、節点ごと及び辺ごとに和を取るように変更し、計算量の効率化とメモリアクセスの連続化の両立を図った。この変更のプログラミング例をリスト3.1に示す。より詳細な変更内容については、「APPENDIX A. スカラ版チューニング内容詳細」を参照されたい。尚、勾配の計算以外にも LU-SGS 法及び数値流束ベクトルの計算等において色分けの削除を行ったが、これらについては、この変更は適用しなかった。その理由は、スカラ版とスレッド版の計算性能比較において明らかである。以下では、このコーディング方法を「色分けの削除及び DO ループの分割」または単に「DO ループの分割」等と記述する。

なお、リスト3.1に「変更後」として示されるような作業用配列を使用したコーディング方法は、構造格子法では良く用いられている方法である。但し、構造格子の場合、全ての検査体積の持つ面の数が一定であるため、リスト3.1「変更後」の DO200及び DO300に相当する部分は、二重ループではなく、DO210及び DO310が展開された一重ループとしてコーディングすることができる。一例として、代表的な海洋モデルのひとつである POM (Princeton Ocean Model) [20] でのコーディング例をリスト3.2に示す。この例は、移流拡散方程式を解いて海洋の塩分濃度及び温度の分布を求めるサブルーチン（サブルーチン名“advt1”）の一部である。POMでは、三次元六面体構造格子に有限体積法が適用されており、その  $x$  及び  $y$  方向表面における流束ベクトル“xflux”及び“yflux”がループ1で、 $z$  方向表面における流束ベクトル“zflux”がループ2でそれぞれ計算された後、検査体積  $(i, j, k)$  における正味の流束“ff”がループ3で計算されている。リスト3.1「変更後」の DO200及び DO300に相当する POM のループ3（リスト3.2）は六面全ての流束の和を取る部分が書き下されて、一重ループとしてコーディングされている。

## 3.2. スレッド版変更内容

ここでは、再帰演算を含まず SMP クラスタにおいてスレッド並列と MPI によるプロセス並列とによりハイブリッド並列計算が可能なスレッド版について、JTAS に加えた変更内容を示す。スレッド版は再帰参照を含まないため、オリジナルの JTAS 同様、ベクトル実行も可能である。

オリジナルの JTAS は、既にベクトル化されており、一般に、ベクトル化可能な DO ループは FORTRAN の持つ自動並列化機能によりスレッド並列化することが可能である。事実、オリジナルの JTAS は一切の変更を加えることなしに、スレッド・MPI ハイブリッド並列化が可能である。ところで、ベクトル化は基本的に最内側 DO ループに対してなされる。一方で、スレッド並列化では、スレッド生成のオーバーヘッドをなるべく小さくする

リスト 3.1 配列定義順序の連続化変更例

変 更 前	変 更 後
<pre> DO 100 IELM=1, N_EL_F   IF (IDS_EL (ICL_EL, IELM). GE. 1) THEN     IEDG=IDS_EL (IE1_EL, IELM)     :     IED6=IDS_EL (IE6_EL, IELM)     IND1=IDS_ED (IN1_ED, IEDGE1)     :     IND4=IDS_ED (IN2_ED, IEDGE4)     :     :     :     :     :     :     :     :     IP=I1     pole(krx_pl, ip)=pole(krx_pl, ip)+drx     pole(kry_pl, ip)=pole(kry_pl, ip)+dry     pole(krz_pl, ip)=pole(krz_pl, ip)+drz     :     IP=IND4     pole(krx_pl, ip)=pole(krx_pl, ip)+drx     pole(kry_pl, ip)=pole(kry_pl, ip)+dry     pole(krz_pl, ip)=pole(krz_pl, ip)+drz     :     :     :     :     :     IED=IED1     ds_ed(ktx_ed, ied)=ds_ed(ktx_ed, ied)+dtx     ds_ed(kty_ed, ied)=ds_ed(kty_ed, ied)+dty     ds_ed(ktz_ed, ied)=ds_ed(ktz_ed, ied)+dtz     :     IED=IED6     ds_ed(ktx_ed, ied)=ds_ed(ktx_ed, ied)+dtx     ds_ed(kty_ed, ied)=ds_ed(kty_ed, ied)+dty     ds_ed(ktz_ed, ied)=ds_ed(ktz_ed, ied)+dtz     :     :     :     END IF 100 CONTINUE </pre>	<pre> DO 100 IELM=1, N_EL_F   IF (IDS_EL (ICL_EL, IELM). GE. 1) THEN     IEDG=IDS_EL (IE1_EL, IELM)     :     IED6=IDS_EL (IE6_EL, IELM)     IND1=IDS_ED (IN1_ED, IEDGE1)     :     IND4=IDS_ED (IN2_ED, IEDGE4)     :     :     elm_wk( 1, ielm)=v0*drx     elm_wk( 2, ielm)=v0*dry     elm_wk( 3, ielm)=v0*drz     END IF 100 CONTINUE  DO 200 IP=1, N_P_F   NELM=IND_TBL (0, IP)   DO 210 IE=1, NELM     IELM=IND_TBL (IE, IP)     pole(krx_pl, ip)=pole(krx_pl , ip)     &amp; + elm_wk( 1, ielm)     pole(kry_pl, ip)=pole(kry_pl , ip)     &amp; + elm_wk( 2, ielm)     pole(krz_pl, ip)=pole(krz_pl , ip)     &amp; + elm_wk( 3, ielm)     :   210 CONTINUE 200 CONTINUE  DO 300 IED=1, N_ED_F   NELM=IED_TBL (0, IED)   DO 310 IE=1, NELM     IELM=IED_TBL (IE, IED)     ds_ed(ktx_ed , ied)=ds_ed(ktx_ed , ied)     &amp; + elm_wk(20, ielm)     ds_ed(kty_ed , ied)=ds_ed(kty_ed , ied)     &amp; + elm_wk(21, ielm)     ds_ed(ktz_ed , ied)=ds_ed(ktz_ed , ied)     &amp; + elm_wk(22, ielm)     :   310 CONTINUE 300 CONTINUE </pre>

リスト 3.2 POM (Princeton Ocean Model) コーディング例

```

C
C Do advective fluxes:
C
  do k=1, kbm1
    do j=2, jm
      do i=2, im
        xflux(i, j, k) = 25e0 * ((dt(i, j) + dt(i-1, j))
$          * (f(i, j, k) + f(i-1, j, k)) * u(i, j, k))
        yflux(i, j, k) = 25e0 * ((dt(i, j) + dt(i, j-1))
$          * (f(i, j, k) + f(i, j-1, k)) * v(i, j, k))
      end do
    end do
  end do

(snip "Add diffusive fluxes")

C
C Do vertical advection:
C
(snip vertical advection at sea surface and bottom)
C
  do k=2, kbm1
    do j=2, jmm1
      do i=2, imm1
        zflux(i, j, k) = 5e0 * (f(i, j, k-1) + f(i, j, k)) * w(i, j, k) * art(i, j)
      end do
    end do
  end do

C
C Add net horizontal fluxes and then step forward in time:
C
  do k=1, kbm1
    do j=2, jmm1
      do i=2, imm1
        ff(i, j, k) = xflux(i+1, j, k) - xflux(i, j, k)
$          + yflux(i, j+1, k) - yflux(i, j, k)
$          + (zflux(i, j, k) - zflux(i, j, k+1)) / dz(k)
      end do
    end do
  end do

(snip "step forward in time")

```

} ループ 1

} ループ 2

} ループ 3



ために、スレッド生成回数の少ない、より外側の DO ループで並列化することが望ましい。JTAS では、色分けによるベクトル化が行われているため、例えばリスト 2.2 に示したコーディング例では、内側の DO ループである DO110 がベクトル化、即ちスレッド並列化されることとなり、スレッド生成のオーバーヘッドによる性能低下が予想された。加えて、スレッド並列化される DO ループの回転数は、生成されたスレッドの中でなるべく多くの演算が行われるように、ベクトル化における場合同様なるべく多い方が望ましいが、色分けによるベクトル化では、一度に並列実行されるのは一つの色に属する辺（勾配の計算の場合要素）のみであり、全ての辺を一度に処理するのに比べて性能が低下することが予想される。一方で、全ての辺について同時に計算することにすれば、リスト 2.1 に示すように DO ループは一重ループとなり、外側かつ回転数の多い理想的なループの構成となるが、既に示したようにリスト 2.1 の DO ループは再帰参照を含み、ベクトル化もスレッド並列化も行うことが出来ないため、スカラ版の方法は採用できない。

ところで、「3.1.3 勾配計算における配列定義順序の連続化」において、リスト 3.1 「変更後」のループ構成は、DO100、DO200 及び DO300 のいずれもが再帰参照を含まず、スレッド並列可能であり、色分けによる方法で問題になると予想される点が改善されている。また、DO100 は当然として、DO210 及び DO310 も基本的な総和演算であるから、通常ベクトル化可能である。そこで、スカラ版では適用しなかった数値流束ベクトル及び制限関数の計算において、勾配の計算の場合同様、この変更を適用してスレッド並列版を作成した。この変更のプログラミング例をリスト 3.3 に示す。これは、HLEW 法による数値流束ベクトルを計算するサブルーチン “SPVCOR” のプログラミング例であり、DO100 において、辺ごとに計算された数値流束ベクトルは、一旦、作業用配列 “EDG\_WK” に辺ごとの値として保存された後、DO110 において、節点（検査体積）ごとの配列 “POLE” に足し込まれている。DO100 における配列 “EDG\_WK” 及び DO110 における配列 “POLE” は、インデックス参照ではなく直接参照となっていることに注意されたい。以下では、このコーディング方法を「再帰参照を回避するための DO ループの分割」または単に「DO ループの分割」等と記述する。

また、LU-SGS 法の計算に関係するサブルーチンである “DIAGONAL”、“SIGMALEFT” 及び “SIGMARIGHT” については、色分けを行った方が良い性能が得られた（「6. スレッド版計算性能測定結果及び評価」参照）。このためスレッド版では、サブルーチン “DIAGONAL” については JTAS オリジナルのままとし、サブルーチン “SIGMALEFT” 及び “SIGMARIGHT” については色分けを削除せずに、超平面を考慮した節点番号の付け替えのみを導入した。

より詳細な変更内容については、「APPENDIX B. スレッド版チューニング内容詳細」を参照されたい。

リスト 3.3 DO ループの分割による配列定義順序の連続化変更例

```

DO 100 IEDGE=1, N_ED_F
  IF (IDS_ED (ICL_ED, IEDGE) .GE. 1) THEN
    I2=IDS_ED (IN1_ED, IEDGE)
    I3=IDS_ED (IN2_ED, IEDGE)
    :
    EDG_WK (1, IEDGE) =F1
    :
    EDG_WK (5, IEDGE) =F5-G5
    :
  END IF
100 CONTINUE
C
DO 110 IP=1, N_P_F
  NEDG=IEDND_TBL (0, IP)
  DO 111 IE=1, NEDG
    IEDGE=IEDND_TBL (IE, IP)
    XSIGN=SIGN (1.0D0, IEDGE)
    IEDGE=ABS (IEDGE)
    R23=DS_ND (IRO_ND, IP)
    F1 =EDG_WK (1, IEDGE)
    :
    FG5=EDG_WK (5, IEDGE)
    :
    POLE (1, IP) =POLE (1, IP) +XSIGN*F1
    :
    POLE (5, IP) =POLE (5, IP) +XSIGN*FG5
    :
  111 CONTINUE
110 CONTINUE

```

#### 4. 計算性能測定条件

ここでは、計算性能の測定をどのような条件のもとに行ったかについて説明する。最初に、性能の測定に使用した CeNSS のハードウェア諸元を、続いて、コンパイラ、リンケージエディタ及び MPI ライブラリ等のソフトウェアについて説明する。最後に、初期条件、各種パラメータ、格子サイズ等、JTAS の入力データに関わる実行条件について説明する。

##### 4.1. ハードウェア

ここでは、性能測定に使用した大規模クラスタ SMP システム CeNSS のハードウェア諸元について簡単に示す。

表 4.1 CeNSS ハードウェア諸元

ハードウェア	Fujitsu PRIMEPOWER HPC2500
論理ピーク性能	9.3TFLOPS 5.2GFLOPS/CPU
メモリ容量	3.6TBytes 64GBytes/node
ノード数	56
CPU 数	1,792 32/node
ノード内アーキテクチャ	SMP (Symmetric Multiple Processor)
CPU	SPARC64V
L2 キャッシュ容量	2 MBytes, on chip
インターコネクタ形状	Full crossbar
インターコネクタ性能	4 GBytes × 2 (送受信)

## 4.2. ソフトウェア

ここでは、測定に使用したコンパイラ、リンケージエディタ及びMPIライブラリ等のソフトウェアに関して説明する。表4.2に使用したコンパイラ、リンケージエディタ及びMPIライブラリのバージョン情報を示す。また、表4.3に翻訳時に指定したコンパイルオプション及び実際に有効になったコンパイルオプションの一覧を示す。尚、表4.3中の“f90ns コマンド”とは、CeNSS上に用意された固有のコンパイル用シェルスクリプトであり、このシェルスクリプト内でコンパイルコマンドが呼び出される。

表 4.2 ソフトウェアバージョン情報

ソフトウェア	バージョン情報
コンパイラ	Fujitsu Fortran Version 5.6
リンケージエディタ	Software Generation Utilities - Solaris Link Editors : 5.8-1.296
MPI ライブラリ	MPI Patchlevel 2.21.0 MPLib version MPLIB-sr2.3.1

表 4.3 コンパイルオプション一覧

	コンパイルオプション
指定したオプション	-Umpi -Qi,p -NRtrap
f90ns コマンド(*)により指定されたオプション	-I/opt/NS/LANG/mpif_old1/FJSVmpi2/bin/./include/sparcv9 -f2004,1321 -Kfast_GP2=3,V9,ocl,largepage=2,hardbarrier -Kmfunc=2 -Kbcopy -IU0 -Ar -Am
有効となったオプション	-fi -A2 -Am -Ar -Fixed -O5 -Qi, p -X7 -x0 -KV9 -KVIS1 -KSPARC64_GP2 -KNOFMADD -Knoauto -Kbcopy -Kdalign -Keval -Knofrecipro -Kfsimple -Kfuse -Knohardbarrier -Knoifunc -Klargepage=2 -Kloop -Kns -Kmfunc=2 -Kocl -Kpreex -Kprefetch=3 -Kprefetch_cache_level=2 -Kprefetch_noinfer -Kprefetch_model=L -Kprefetch_nostrong -Knostriping -Ktiling=40 -Kunroll -KNOOMP -Knothreadsafesafe -Knoarray_private -Knomregion_extension -Nnoallextput -Nnoautoobjstack -Nnocalleralloc -Nnocompdisp -Nnocopyarg -Nfreealloc -Nnof90move -Nnof95 -Nnomallocfree -Nnoobsfun -NRtrap -Nnorecursive -Nnosave
実行時間計測用に追加指定したオプション	-I../include.TIMER
Profiler 情報取得用に追加指定したオプション	-Ktl_trt -I../include.DUMMY
スレッド並列実行用に追加指定したオプション	-Kparallel -Kvppocl -Et

(\*) f90ns コマンド：CeNSS 固有のコンパイル用シェルスクリプト

### 4.3. JTAS 実行条件

ここでは、性能測定のために JTAS を実行するに際して設定した、初期条件、各種パラメータ、格子サイズ等、JTAS の入力データに関して説明する。

#### 4.3.1. 初期条件

性能測定のために設定した初期条件は、以下の通り。この初期条件では、空力係数（揚力  $C_l$  及び抗力  $C_d$ ）の計算を行うこと及び翼面に相当する部分に非スリップ境界が設定されている。

表 4.4 性能測定に使用した主な初期条件

設定データ	設定値	
時間積分ステップ数	500ステップ	
クーラン数	1.0x10 <sup>5</sup>	
迎角	0.0度	
レイノルズ数	1.0x10 <sup>6</sup>	
比熱比 $\gamma$	1.4	
自由流温度	273.15K	
壁面温度	0.5	
自由流マッハ数	0.8	
流れの種別	層流	
空力係数計算	あり (subroutine cldmns)	
境界流入条件	初期密度 (RO)	1.0
	流入速度 (UX, UY, UZ) <sup>T</sup>	(9.465727652959386x10 <sup>-1</sup> , 0.0, 0.0) <sup>T</sup>
	初期圧力 (P)	1.0
	初期渦動粘性係数 (RT)	20.0
	非スリップ境界	あり

## 4.3.2. 格子点数

性能測定のために設定した計算格子点の数は以下の通り。MPIによる並列実行時には、「2.5 並列化」で示したように、分割された領域の間で情報の授受を行うための“Receiving Vertices”  $R_i$  が付加される。ここでは、2プロセス実行の場合と、8プロセス実行の場合について、この情報授受のためにオーバーラップして設けられた“Receiving Vertices”  $R_i$  を含めた格子点数が示されている。実際に解析が行われる格子点数は「正味」として示した。

表 4.5 性能測定に使用した計算格子点の数

要素名	要素			辺	格子点	境界	
	三角錐	三角柱	四角錐			三角形	矩形
	tetrahedra	prisms	pyramids			triangles	rectangles
変数名	N_EL_F	N_PRISM	N_PYR	N_ED_F	N_P_F	N_BD_F	N_RECT
2 process							
process 0	716,199	0	0	861,171	128,802	32,399	0
process 1	481,662	141,636	879	870,779	160,656	25,808	4,145
小計	1,197,861	141,636	879	1,731,950	289,458	58,207	4,145
合計	1,340,376			1,731,950	289,458	62,352	
8 process							
process 0	182,352	0	0	222,698	33,990	13,050	0
process 1	134,190	35,096	230	237,522	44,051	10,536	2,304
process 2	120,333	36,132	219	222,736	42,006	9,829	2,349
process 3	123,079	34,981	269	223,935	41,992	10,051	2,387
process 4	121,467	40,658	189	233,670	44,689	10,269	2,413
process 5	186,599	0	0	227,823	34,763	13,076	0
process 6	186,367	0	0	227,621	34,749	13,181	0
process 7	190,362	0	0	232,919	35,654	13,864	0
小計	1,244,749	146,867	907	1,828,924	311,894	93,856	9,453
合計	1,392,523			1,828,924	311,894	103,309	
正味							
小計	1,173,840	141,636	879	1,690,955	280,969	40,604	4,145
合計	1,316,355			1,690,955	280,969	44,749	

## 5. スカラ版計算性能測定結果及び評価

ここでは、スカラ版 JTAS による性能測定の結果を示し、JTAS オリジナルとの性能比較等、測定結果に対する若干の評価を行う。まず、「5.1 計算結果の確認」で、変更が正しく行われたことをどのように確認したかについて説明し、続いて「5.2 測定方法」で性能測定を行った方法について説明する。最後に、「5.3 測定結果及び評価」において、性能測定を行った結果を示し、その簡単な評価を行う。

### 5.1. 計算結果の確認

スカラ版を作成するにあたり JTAS オリジナルの計算結果と比較し変更が正しいものであることを確認した。この確認は、スカラ版において、時間積分500ステップの計算終了時における揚力“ $Cl$ ”、抗力“ $Cd$ ”及びピッチモーメント“ $Tpm$ ”の値が、オリジナルと十分な精度を持って一致しているか否かをもって行った。表5.1に確認を行った計算結果を示す。表5.1によれば、各プロセスごとのスカラ版計算結果はオリジナルの計算結果と良く一致している。従って、変更が正しく行われたことが確認された。尚、プロセス数の異なる計算結果の間では、オリジナルにおいても相互の計算結果が良く一致しているとは言えない。この不一致は、「2.5 並列化」で示したように JTAS における LU-SGS 法の導入方法によるものであると考えられる。時間積分500ステップでは、残差 (residual) が $10^{-4}$ 程度と十分に収束しているとは言いがたい。このために、この不一致が発生していると考えられる。

表 5.1 計算結果の確認

version	$Cl$	$Cd$	$Tpm$
2 process			
オリジナル	-2.510437870811829e-03	2.606253245212966e-03	5.925822252922501e-04
スカラ版	-2.510437870811722e-03	2.606253245212965e-03	5.925822252921988e-04
4 process			
オリジナル	-2.524233286685411e-03	2.606823218510942e-03	6.011854742564876e-04
スカラ版	-2.524233286685662e-03	2.606823218510942e-03	6.011854742566543e-04
8 process			
オリジナル	-2.504319943187164e-03	2.606454561021094e-03	5.876676847528363e-04
スカラ版	-2.504319943187180e-03	2.606454561021095e-03	5.876676847528665e-04
16process			
オリジナル	-2.517598774349860e-03	2.607755371615025e-03	5.932925432830958e-04
スカラ版	-2.517598774349668e-03	2.607755371615025e-03	5.932925432829816e-04
32process			
オリジナル	-2.516358401408989e-03	2.609155705632342e-03	5.942460079979562e-04
スカラ版	-2.516358401409026e-03	2.609155705632342e-03	5.942460079979813e-04
64process			
オリジナル	-2.533700724012261e-03	2.610311052882490e-03	6.031517823320042e-04
スカラ版	-2.533700724012269e-03	2.610311052882491e-03	6.031517823319886e-04

## 5.2. 測定方法

ここでは、どのような方法で性能測定を行ったかについて説明する。最初に、経過時間 (Elapsed Time) 及び CPU 時間を測定するためのタイマ挿入の方法と、挿入したタイマによって測定した区間について説明する。続いて、プログラムの実行状況の解析を支援するツールであるプロファイラを実行するにあたって指定したオプション等について説明する。

### 5.2.1. タイマの挿入

経過時間 (Elapsed Time) 及び CPU 時間を測定するために、JTAS のソースプログラムにタイマを挿入した。ここでは、そのタイマの挿入方法と挿入によって計測を行った区間について説明する。より具体的な挿入方法については、「APPENDIX A.5 タイマの挿入」を参照されたい。

経過時間を測定するためには、ある時点からの経過時間をマイクロ秒単位で取得するサービスルーチン “GETTOD” を使用した。測定区間の前後で “GETTOD” により経過時間を測定し、その差をもって、その区間を処理するのに要した経過時間とした。また、CPU 時間を計測するためには、実行開始からの CPU 時間を返すサービス関数 “ETIME” を使用した。CPU 時間も、経過時間の場合同様、測定区間の前後で “ETIME” により CPU 時間を計測し、その差をその区間の処理に要した CPU 時間とした。また、MPI によるプロセス並列実行であることを考慮して、測定前には基本的にバリア同期を取った。そして、MPI のルートプロセス (ランク 0) における経過時間及び CPU 時間を測定結果とした。但し、LU-SGS 法における前進、後退スイープにおいては、各プロセスごとに、超平面の数が異なることから、それを処理するサブルーチンの呼び出し回数も異なる。このため、この区間においてはバリア同期は取らなかった。バリア同期を取らない区間については各プロセスごとにその区間を処理するのに要する経過時間及び CPU 時間が異なる。そこで、全てのプロセスの経過時間及び CPU 時間の内、最大値と最小値を測定結果とした。リスト 5.1 に経過時間及び CPU 時間の測定例を示す。リスト 5.1 において、測定区間ごとに整数型変数 “IND\_TIMER” に異なる値 (“測定区間 ID”) を与えることにより全ての区間の測定を同時に行った。また、バリア同期が取れない区間においては、MPI ライブラリ “MPI\_barrier” の呼び出しを抑止するため、論理型配列 “barrier (n, IND\_TIMER)” (n=1, 2) に論理値 “false.” を与えた。測定区間は、主な処理のまとまりごとに設定した。測定区間を具体的にどのように設定したか及びどの区間においてバリア同期を取らなかったかについて、その一覧を表 5.2 に示す。

尚、タイマ挿入に際しては、サービス関数 “ETIME” 呼び出しのオーバーヘッドが比較的大きいことから、測定区間の直前ではサービス関数 “ETIME” を呼び出した後にサービスルーチン “GETTOD” の呼び出しを行い、計測区間の直後では、サービスルーチン “GETTOD” を呼び出した後に、サービス関数 “ETIME” の呼び出しを行った。これにより、経過時間の計測結果にサービス関数 “ETIME” の呼び出し時間が含まれてしまうことを避けることができる。

### 5.2.2. プロファイラによる解析

ここでは、プロファイラによる実行状況の解析を行うための方法について説明する。プロファイラによる実行状況解析を行うためには、コンパイルオプション “-Kil\_trt” を指定してコンパイルを行った。この時、JTAS のソースプログラムへのタイマの挿入は行わない状態でコンパイルを行った。タイマは、リスト 5.1 に示すように全てインクルードファイル内に置き、これを JTAS のソースプログラム内にインクルードすることによりタイマ挿入を行うとともに、プロファイラ実行時には、このインクルードファイルを空のダミーファイルで置き換えることによりタイマの挿入を抑止し、そのオーバーヘッドも含めて解析してしまうことを避けた。また、実行を行う際には、いくつかの環境変数にプロファイラに指示するためのオプションを設定した。この時設定した環境変数の変数名及び、その環境変数に設定した値を表 5.3 に示す。



リスト 5.1 タイマ挿入例 (続く)

タイマ挿入例	
<pre> : include "timer_header.h" : : IND_TIMER= "測定区間 ID" include "timer_begin.h"          測 定 区 間  IND_TIMER= "測定区間 ID" include "timer_end.h" : : </pre>	
include file "timer_header.h" の内容	(説明は「APPENDIX A.5 タイマの挿入」参照)
<pre> parameter ( MAXPROCESS = 128 ) parameter ( MAXTIMER   = 50 )  real*4    ETIME real*4    WRK_TIMER(2)  real*8    ELB_TIMER, ELE_TIMER, TTL_TIMER, PES_TIMER real*4    CPB_TIMER, CPE_TIMER, CPU_TIMER integer   CNT_TIMER, CNT_PES character CAP_TIMER*64  logical timer_on logical cpu_on logical barrier_on(2, MAXTIMER)  common /TIMER_AREA/ &amp;      TTL_TIMER(4, MAXTIMER), &amp;      PES_TIMER(0:MAXPROCESS-1, 4, MAXTIMER), &amp;      ELB_TIMER( MAXTIMER), ELE_TIMER( MAXTIMER), &amp;      CPB_TIMER(2, MAXTIMER), CPE_TIMER(2, MAXTIMER), &amp;      CPU_TIMER(2, MAXTIMER), &amp;      CNT_TIMER( MAXTIMER), &amp;      CNT_PES  (0:MAXPROCESS-1,  MAXTIMER) common /TIMER_CAPT/ CAP_TIMER(MAXTIMER) common /TIMER_FLAG/ timer_on, cpu_on, barrier_on  common /MPI_PARAM/ npes, mype, iwcomm </pre>	

リスト 5.1 タイマ挿入例 (続き)

<pre>include file "timer_begin.h"の内容 IF( timer_on ) THEN   CNT_TIMER(IND_TIMER) = CNT_TIMER(IND_TIMER) + 1   IF(barrier_on(1, IND_TIMER)) &amp;    call MPI_barrier(iwcomm, ierror)   IF( cpu_on ) THEN     CPU_TIMER(1, IND_TIMER) = ETIME( WRK_TIMER )     CPB_TIMER(1, IND_TIMER) = WRK_TIMER(1)     CPB_TIMER(2, IND_TIMER) = WRK_TIMER(2)   END IF   CALL GETTOD( ELB_TIMER(IND_TIMER) ) END IF</pre>	(説明は「APPENDIX A.5 タイマの挿入」参照)
<pre>include file "timer_end.h"の内容 IF( timer_on ) THEN   IF(barrier_on(2, IND_TIMER) ) &amp;    call MPI_barrier(iwcomm, ierror)   CALL GETTOD( ELE_TIMER(IND_TIMER) )   IF( cpu_on ) THEN     CPU_TIMER(2, IND_TIMER) = ETIME( WRK_TIMER )     CPE_TIMER(1, IND_TIMER) = WRK_TIMER(1)     CPE_TIMER(2, IND_TIMER) = WRK_TIMER(2)   END IF   TTL_TIMER(1, IND_TIMER) = TTL_TIMER(1, IND_TIMER) &amp;    + ELE_TIMER( IND_TIMER) &amp;    - ELB_TIMER( IND_TIMER)   TTL_TIMER(2, IND_TIMER) = TTL_TIMER(2, IND_TIMER) &amp;    + CPE_TIMER(1, IND_TIMER) &amp;    - CPB_TIMER(1, IND_TIMER)   TTL_TIMER(3, IND_TIMER) = TTL_TIMER(3, IND_TIMER) &amp;    + CPE_TIMER(2, IND_TIMER) &amp;    - CPB_TIMER(2, IND_TIMER)   TTL_TIMER(4, IND_TIMER) = TTL_TIMER(4, IND_TIMER) &amp;    + CPU_TIMER(2, IND_TIMER) &amp;    - CPU_TIMER(1, IND_TIMER) END IF</pre>	(説明は「APPENDIX A.5 タイマの挿入」参照)

測定結果の累計処理

表 5.2 測定区間の名称一覧 (続く)

測定区間の名称	説明
TOTAL	JTAS 全体 MPI ライブラリ “MPI_initialize” が呼び出された直後から MPI ライブラリ “MPI_finalize” が呼び出される直前まで
PREPROCESS	初期化処理 MPI ライブラリ “MPI_initialize” が呼び出された直後から サブルーチン “SPV0SV” が呼び出される直前まで
INTEGRATION	時間積分ループ サブルーチン “SPV0SV” が呼び出された直後から MPI ライブラリ “MPI_finalize” が呼び出される直前まで
SPVCOR	HLEW 法による数値流束ベクトル $\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n}$ の計算 サブルーチン “SPVCOR” を呼び出す 直前から サブルーチン “SPVCOR” を呼び終わった直後まで
SPDRF 1	勾配 $\nabla \mathbf{q}_i$ の計算 サブルーチン “SPDRF1_tet” を呼び出す 直前から サブルーチン “SPDRF1_pyr” を呼び終わった直後まで
__tet	三角錐要素の勾配 $\nabla \mathbf{q}_i$ の計算 サブルーチン “SPDRF1_tet” を呼び出す 直前から サブルーチン “SPDRF1_tet” を呼び終わった直後まで
__prism	三角柱要素の勾配 $\nabla \mathbf{q}_i$ の計算 サブルーチン “SPDRF1_pri” を呼び出す 直前から サブルーチン “SPDRF1_pri” を呼び終わった直後まで
__pyr	四角錐要素の勾配 $\nabla \mathbf{q}_i$ の計算 サブルーチン “SPDRF1_pyr” を呼び出す 直前から サブルーチン “SPDRF1_pyr” を呼び終わった直後まで
LIMITER	Venkatakrishnan の制限関数 $\Psi_i$ ( $0 \leq \Psi_i \leq 1$ ) の計算 測定区間 “LIMITER1” と “LIMITER2” の和
LIMITER 1	$\Delta_{\max}$ , $\Delta_{\min}$ の計算 サブルーチン “LIMITER1” を呼び出す 直前から サブルーチン “LIMITER1” を呼び終わった直後まで
LIMITER 2	Venkatakrishnan の制限関数 $\Psi_i$ ( $0 \leq \Psi_i \leq 1$ ) の計算 サブルーチン “LIMITER2” を呼び出す 直前から サブルーチン “LIMITER2” を呼び終わった直後まで

表 5.2 測定区間の名称一覧 (続き)

測定区間の名称	説 明
LU-SGS	LU-SGS 法ソルバ部 サブルーチン “LUSGS” の最初から最後まで
DIAGONAL	対角行列 $D_i$ の計算 サブルーチン “DIAGONAL” を呼び出す 直前から サブルーチン “DIAGONAL” を呼び終わった直後まで
SIGMALEFT	前進スイープの一部 サブルーチン “SIGMALEFT” を呼び出す 直前から サブルーチン “SIGMALEFT” を呼び終わった直後まで この区間の測定においてバリア同期は取られない
SWEEPL	前進スイープの計算 サブルーチン “SWEEPL” を呼び出す 直前から サブルーチン “SWEEPL” を呼び終わった直後まで この区間の測定においてバリア同期は取られない
SIGMARIGHT	後退スイープの一部 サブルーチン “SIGMARIGHT” を呼び出す 直前から サブルーチン “SIGMARIGHT” を呼び終わった直後まで この区間の測定においてバリア同期は取られない
SWEEP R	後退スイープの計算 サブルーチン “SWEEP R” を呼び出す 直前から サブルーチン “SWEEP R” を呼び終わった直後まで この区間の測定においてバリア同期は取られない
UPDATEQ	$Q^{n+1} = Q^n + \Delta Q^n$ の計算 サブルーチン “UPDATEQ” を呼び出す 直前から サブルーチン “UPDATEQ” を呼び終わった直後まで
COMMUNICA- TION	プロセス間通信 サブルーチン “SUB_MPICAL L” の最初から最後まで
Turbulent	乱流モデル計算後に行われるプロセス間通信 IFLAG = 0 が設定されて呼び出されたサブルーチン “SUB_MPICAL L” の最初から最後まで
Gradient	勾配 $\nabla q_i$ 計算後に行われるプロセス間通信 IFLAG = 1 が設定されて呼び出されたサブルーチン “SUB_MPICAL L” の最初から最後まで
Limiter	Venkatakrishnan の制限関数 $\Psi_i$ ( $0 \leq \Psi_i \leq 1$ ) の計算後に行われるプロセス間通信 IFLAG = 2 が設定されて呼び出されたサブルーチン “SUB_MPICAL L” の最初から最後まで
Control Vol.	検査体積の表面積、体積等の諸量計算後に行われるプロセス間通信 IFLAG = 3 が設定されて呼び出されたサブルーチン “SUB_MPICAL L” の最初から最後まで
Flux	HLLW 法による数値流束ベクトル [9] $F(Q) \cdot n$ の計算後に行われるプロセス間通信 IFLAG = 4 が設定されて呼び出されたサブルーチン “SUB_MPICAL L” の最初から最後まで
Forward	LU-SGS 法前進スイープ計算後に行われるプロセス間通信 IFLAG = 5 が設定されて呼び出されたサブルーチン “SUB_MPICAL L” の最初から最後まで
Backward	LU-SGS 法後退スイープ計算後に行われるプロセス間通信 IFLAG = 6 が設定されて呼び出されたサブルーチン “SUB_MPICAL L” の最初から最後まで
SUBTOTAL	測定区間 “SPVCOR”, “SPDRF 1”, “LIMITER”, “LU-SGS” 及び “COMMUNICATION” の和

表 5.3 プロファイラ実行時に設定した環境変数

環境変数	設定値
TRT_ENV	PMP=on
PROF_STATS	7
PROF_PA	cov, ins, sta
PROF_MEMORY_SIZE	512000

### 5.3. 測定結果及び評価

ここでは、JTAS の性能測定の結果及びその評価について示す。測定は、プロセス数 2, 4, 8, 16, 32 及び 64 の 6 ケースについて行った。1 プロセスによる実行は、それに必要なデータが存在しないため、これを行わなかった。また、測定は他のジョブも存在する通常の運用状態の元で行った。従って、特に経過時間の測定結果には変動要因が含まれていることに注意されたい。最初に、全体の測定結果について概観した後、主な処理ごとの測定結果について示し、簡単な評価を行う。

#### 5.3.1. 全体の測定結果及び評価

ここでは、JTAS 全体に関する測定結果について概観する。表 5.4 及び図 5.1 に主な測定区間における経過時間を測定した結果及びその区間における経過時間が全体の経過時間（測定区間“TOTAL”）に占めるコスト（割合）を示す。各欄の上段が経過時間であり、下段がコストである。また、経過時間の単位は秒、コストの単位は%である。測定区間“SUBTOTAL”は、測定区間“SPVCOR”以下の測定区間について、その経過時間の合計を取ったものである。各測定区間の具体的な設定については、表 5.2 を参照されたい。JTAS オリジナルのいずれのプロセス数での実行においても、測定区間“SUBTOTAL”によって全体の 90% 以上のコストが把握できており、測定区間の設定は妥当である。特徴的なのは、勾配  $\nabla \mathbf{q}_i$  を計算する部分である測定区間“SPDRF1”におけるコストが、オリジナルで全体の三分の一以上、スカラ版でも四分の一前後と大きな割合を占めていることである。このコストは、測定区間“SPVCOR”における数値流束フラックスの計算や測定区間“LU-SGS”における LU-SGS 法の計算のコストよりも高い割合を占めている。従って、勾配  $\nabla \mathbf{q}_i$  を如何に効率よく計算するかということが、JTAS の性能を大きく左右することがわかる。

各測定区間におけるより詳細な測定結果及び評価は、「5.3.2 測定区間“SPVCOR”における測定結果及び評価」から「5.3.6 測定区間“COMMUNICATION”における測定結果及び評価」において示す。

ところで、測定区間“COMMUNICATION”におけるプロセス間通信のコストは、特にスカラ版の 32 プロセス及び 64 プロセスで 15% 以上とかなり大きな割合を占めている。但し、通常の解析に使用されているデータの格子点数は、測定に使用したデータの百倍程度あり、これを 100 プロセス程度の並列度で実行している。従って、この測定に使用したデータにおいて、1 プロセス当たりの格子点数が実解析の規模と同程度になるのは、高々数プロセス程度の並列度で実行した時であり、32 及び 64 プロセスによってこのデータを実行するには、データが小規模すぎることが言える。よって、ここでは、プロセス間通信によるオーバーヘッドについての詳しい検討は行わない。

測定は、他のジョブも存在する通常の運用状態の元で行った。従って、経過時間は他のジョブの影響で変動することが考えられる。そこで、他のジョブの影響を比較的受け難い CPU 時間についても同時に測定を行った。表 3.4 に、主な測定区間における CPU 時間を測定した結果及びその区間における CPU 時間が全体の CPU 時間（測定区間“TOTAL”）に占めるコストを示す。各欄の上段が CPU 時間であり、下段がコストである。また、CPU 時間の単位は秒、コストの単位は%である。測定区間“SUBTOTAL”は、測定区間“SPVCOR”以下の測定区間

表 5.4 JTAS 全体の経過時間測定結果

経過時間 時間積分500ステップ		上段：経過時間 [秒] 下段：コスト [%]				
オリジナル						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
TOTAL	9694.77	4597.24	2491.43	1460.14	848.91	495.31
PREPROCESS	78.45 0.81	30.45 0.66	17.27 0.69	5.77 0.40	3.06 0.36	2.22 0.45
INTEGRATION	9616.30 99.19	4566.77 99.34	2474.15 99.31	1454.35 99.60	845.84 99.64	493.08 99.55
SPVCOR	1466.09 15.12	725.14 15.77	376.80 15.12	202.55 13.87	112.77 13.28	57.02 11.51
SPDRF1	3304.28 34.08	1532.20 33.33	894.27 35.89	569.08 38.97	311.82 36.73	172.20 34.77
LIMITER	1673.68 17.26	837.68 18.22	439.68 17.64	239.21 16.38	126.88 14.95	50.94 10.28
LU-SGS	2425.57 25.02	1069.97 23.27	515.90 20.71	278.49 19.07	166.60 19.63	109.47 22.10
COMMUNI- CATION	25.47 0.26	37.80 0.82	47.40 1.90	56.43 3.86	66.06 7.78	67.36 13.60
SUBTOTAL	8895.09 91.75	4202.79 91.78	2276.05 91.36	1345.76 92.17	784.13 92.37	456.99 92.26
スカラ版						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
TOTAL	5360.71	2574.41	1364.26	760.21	455.53	312.92
PREPROCESS	56.33 1.05	19.31 0.75	8.96 0.66	4.09 0.54	2.65 0.58	2.46 0.79
INTEGRATION	5304.37 98.95	2555.10 99.25	1355.29 99.34	756.12 99.46	452.87 99.42	310.45 99.21
SPVCOR	857.97 16.00	397.66 15.45	195.26 14.31	92.93 12.22	47.44 10.41	26.10 8.34
SPDRF1	1572.96 29.34	739.90 28.74	397.72 29.15	206.16 27.12	105.09 23.07	57.83 18.48
LIMITER	882.50 16.46	376.67 14.63	172.41 12.64	75.69 9.96	38.52 8.46	20.17 6.45
LU-SGS	1243.24 23.19	632.95 24.59	341.65 25.04	206.88 27.21	132.50 29.09	97.95 31.30
COMMUNI- CATION	25.82 0.48	40.49 1.57	44.72 3.28	63.01 8.29	70.13 15.40	75.97 24.28
SUBTOTAL	4582.49 85.48	2187.67 84.98	1151.76 84.42	644.67 84.80	393.68 86.42	278.02 88.85

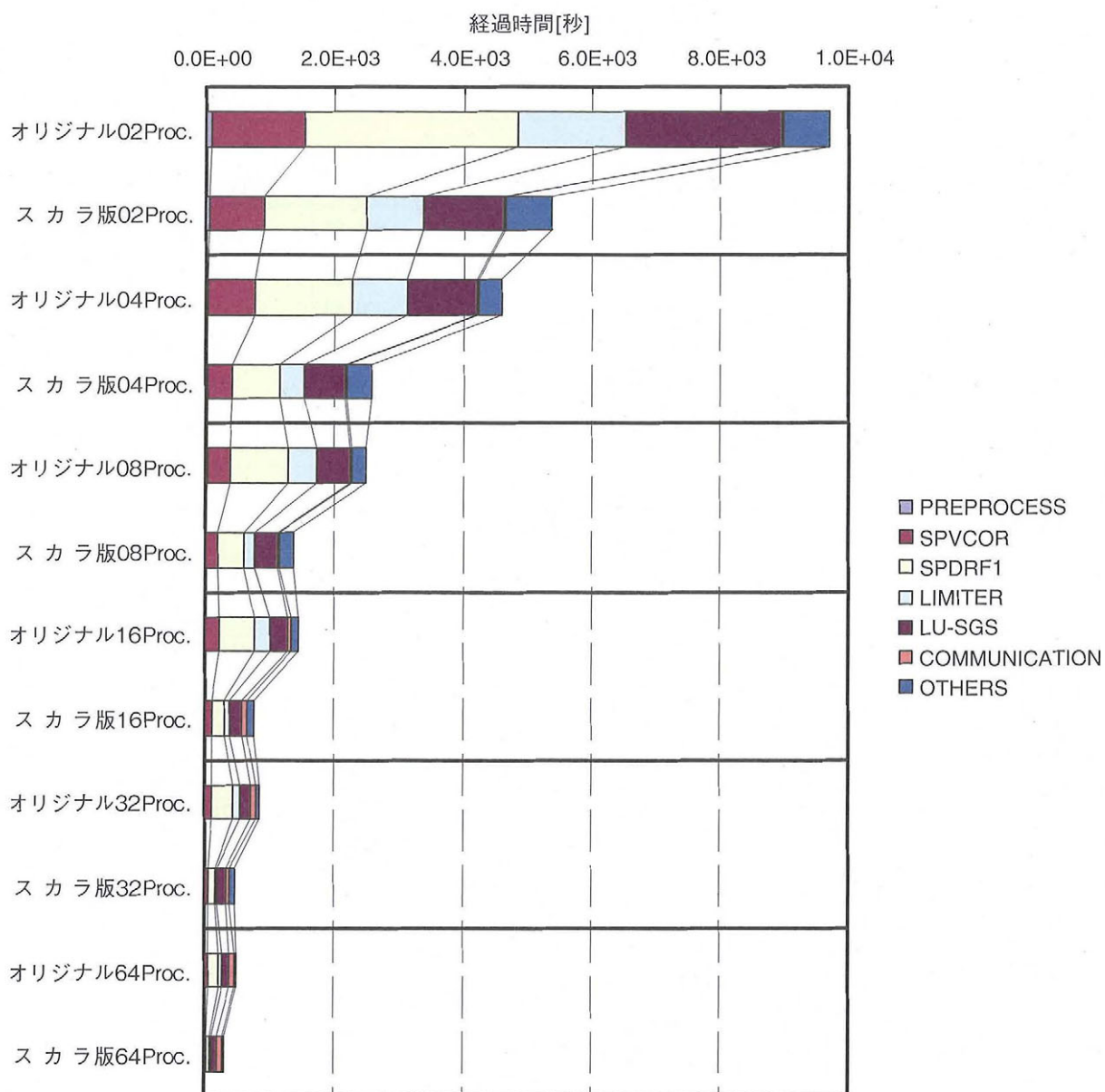


図 5.1 JTAS 全体の経過時間測定結果

表 5.5 JTAS 全体の CPU 時間測定結果

CPU 時間							上段：CPU 時間 [秒]
時間積分500ステップ							下段：コスト [%]
オリジナル							
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.	
TOTAL	8615.10	4083.92	2222.38	1195.37	677.65	387.55	
PREPROCESS	70.18 0.81	27.64 0.68	10.21 0.46	5.18 0.43	2.35 0.35	1.49 0.38	
INTEGRATION	8544.90 99.19	4056.27 99.32	2212.16 99.54	1190.18 99.57	675.29 99.65	386.05 99.61	
SPVCOR	1465.73 17.01	698.88 17.11	369.90 16.64	186.40 15.59	96.86 14.29	44.68 11.52	
SPDRF1	2544.82 29.54	1198.66 29.35	675.07 30.38	340.80 28.51	175.55 25.91	93.89 24.23	
LIMITER	1670.67 19.39	804.62 19.70	433.61 19.51	222.08 18.58	112.01 16.53	39.07 10.08	
LU-SGS	2168.53 25.17	1069.97 26.20	487.84 21.95	276.72 23.15	164.81 24.32	107.13 27.64	
COMMUNI- CATION	25.89 0.30	37.77 0.92	47.34 2.13	56.47 4.72	64.65 9.54	66.21 17.08	
SUBTOTAL	7875.64 91.42	3809.9 93.29	2013.76 90.61	1082.47 90.56	613.88 90.59	350.98 90.56	
スカラ版							
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.	
TOTAL	5199.58	2513.38	1272.16	716.53	436.37	304.54	
PREPROCESS	54.85 1.05	17.56 0.70	7.73 0.61	3.62 0.51	2.05 0.47	1.76 0.58	
INTEGRATION	5144.72 98.94	2495.72 99.30	1264.42 99.39	712.91 99.49	434.32 99.53	302.77 99.42	
SPVCOR	857.94 16.50	397.65 15.82	185.14 14.55	90.90 12.69	47.42 10.87	26.13 8.58	
SPDRF1	1500.14 28.85	709.06 28.21	348.01 27.36	166.36 23.22	89.83 20.59	54.74 17.97	
LIMITER	882.54 16.97	376.77 14.99	165.26 12.99	75.78 10.58	38.65 8.86	20.24 6.65	
LU-SGS	1203.62 23.15	613.23 24.40	323.99 25.47	206.38 28.80	131.13 30.05	96.05 31.54	
COMMUNI- CATION	25.84 0.50	40.66 1.62	44.81 3.52	63.16 8.81	68.50 15.70	74.53 24.47	
SUBTOTAL	4470.08 85.97	2137.37 85.04	1067.21 83.89	602.58 84.10	375.53 86.06	271.74 89.23	



について、その CPU 時間の合計を取ったものである。各測定区間の具体的な設定については、表5.2を参照されたい。ここで、CPU 時間の測定結果においても把握率、コスト分布等、経過時間で見た場合と同程度の測定結果が得られた。従って、以降では実行時間として、経過時間を使用することとする。

表5.6及び図5.2に、経過時間を元に算出した並列化による性能向上の程度をあらわす加速率を示す。加速率はオリジナル、スカラ版共に2プロセスによる実行を基準としていることに注意されたい。従って、4プロセス実行では、論理的な加速率の上限を超えたスーパーリニアな加速が達成されている。16プロセス実行でも90%に近い(スカラ版で88%)高い加速率が得られている。一方、32及び64プロセス実行では、加速率の低下が見られる。この原因としては、プロセス間通信のコストが高いことが、その一つとして考えられる。また、非構造格子の特性上、全てのプロセスに均等に格子を分配することが難しいため、計算負荷の不均衡(ロードインバランス)も一因となっていると考えられる。しかし、既に述べたように測定用データの格子点数がさほど多くないことから、ここでは、詳しい原因について検討することはしない。

表 5.6 並列実行による加速率

測定区間：TOTAL						
算定根拠：経過時間						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	1.00	2.11	3.89	6.64	11.42	19.57
スカラ版	1.00	2.08	3.93	7.05	11.77	17.13

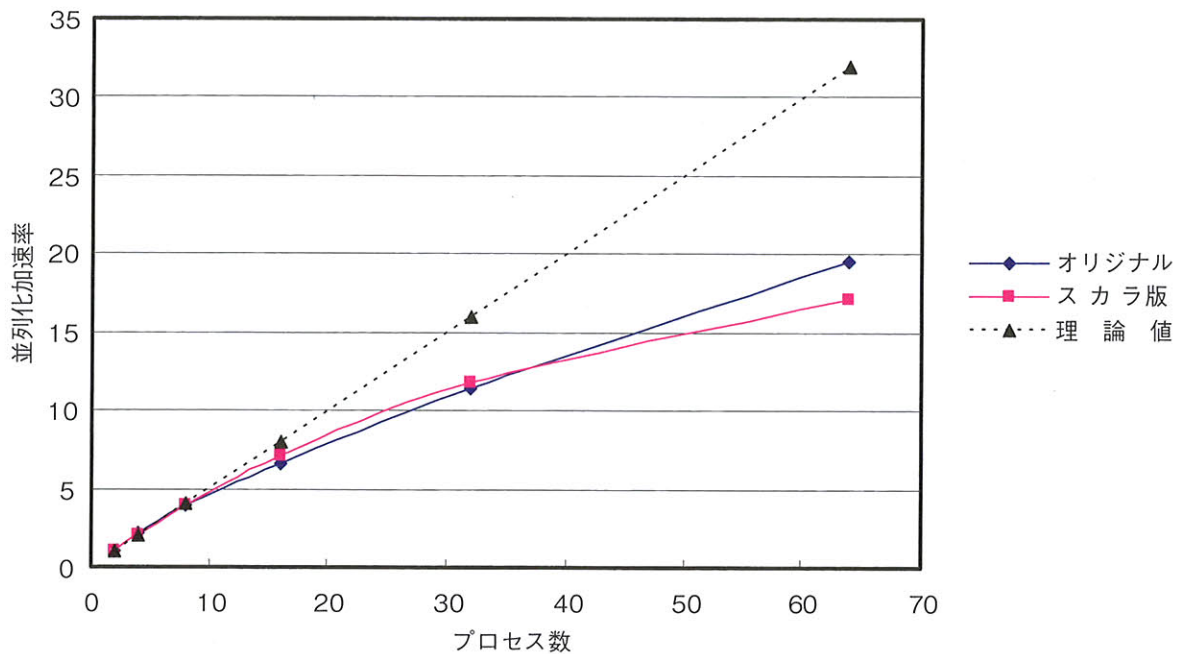


図 5.2 並列実行による加速率

表5.7及び図5.3に、経過時間を元に算出した JTAS オリジナルに対してスカラ版でどの程度性能が向上したかを表す性能向上率を示す。性能向上率は、オリジナルの実行に要した経過時間をスカラ版の実行に要した経過時間で除したのとして計算した。64プロセス実行を除き、いずれのプロセス数による実行でも約1.8倍またはそれ以上の性能向上を得ることが出来た。これによりスカラ版における最適化の効果が確認された。

表 5.7 最適化による性能向上率

測定区間：TOTAL						
算定根拠：経過時間						単位：[-]
プロセス数	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
性能向上率	1.81	1.79	1.83	1.92	1.86	1.58

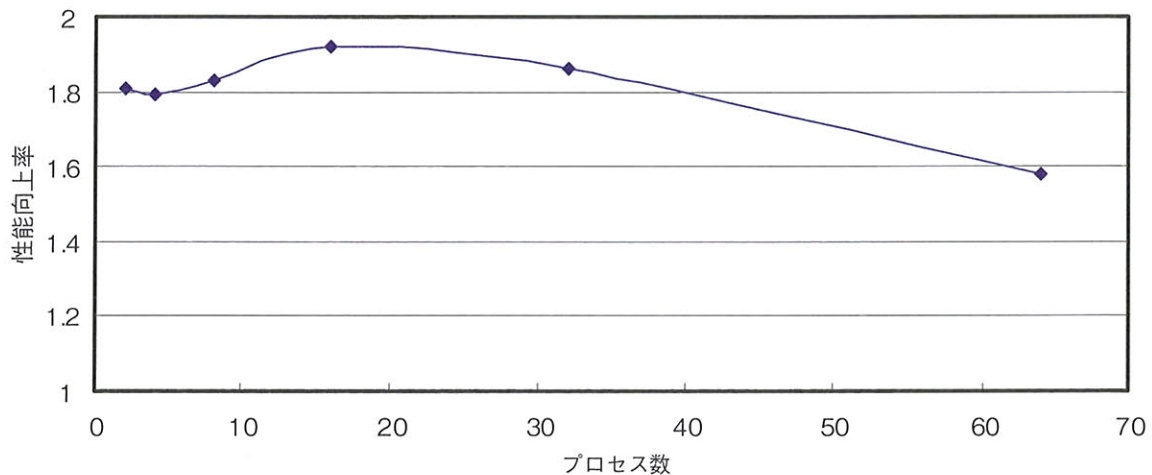


図 5.3 最適化による性能向上率

表5.8及び図5.4にプロファイラによって得られた実行状況解析結果の抜粋を示す。表中，“Min.”はプロファイラより得られた各プロセスの解析結果の内の最小値を，“Max.”は同様に最大値をそれぞれ示す。（ ）内は、その最大値または、最小値が測定されたプロセスのランクである。また，“Total”はプロファイラが全プロセスに亘って解析し“Total”として表示した値であり，“Ave.”は“Total”の値をプロセス数で割って得られたプロセス当たりの単純平均値である。

表5.8に明らかなように、浮動小数点演算性能（FLOPS）、L2 キャッシュ・ミス率、アドレス変換バッファ・ミス率、“Memory Write Back”と全ての項目において改善が見られており、スカラ版で行った最適化の効果が確認された。但し、1 CPU の論理ピーク性能に対しては高々3から4%程度の性能に留まった。

### 5.3.2. 測定区間“SPVCOR”における測定結果及び評価

測定区間“SPVCOR”では、HLLW 法による数値流束ベクトル  $\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n}$  の計算を行っているサブルーチン“SPVCOR”について、経過時間の測定及び実行状況の解析を行なった。表5.9及び図5.5から5.7にこの区間における測定及び解析結果を示す。経過時間は、サブルーチン“SPVCOR”の実行に要した時間であり、バリア同期を取ってルートプロセス（ランク0）における測定結果のみを示している。スカラ版の経過時間において、下段は性能向上率であり、オリジナルの実行に要した経過時間をスカラ版の実行に要した経過時間で除したものと計算した。

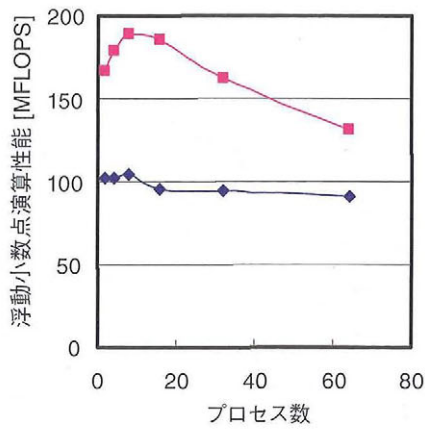
経過時間は2倍前後と、JTAS 全体と同程度の性能向上が見られる。L2 キャッシュ・ミス率と“Memory Write Back”が改善しており、その結果、浮動小数点演算性能 FLOPS が向上したと考えられる。その一方、2 プロセス実行の場合に、L2 キャッシュ・ミス率が約2%と未だ高い状態に留まっている。そのため、浮動小数点演算性能の論理ピーク性能比は6%弱に留まっている。

表 5.8 プロファイラ情報抜粋 (続く)

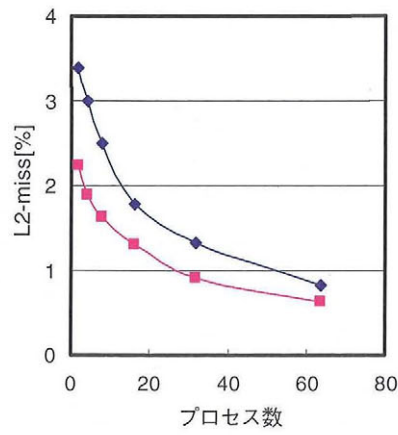
測定区間：TOTAL <span style="float: right;">() 内はランク</span>						
浮動小数点演算性能 FLOPS [MFLOPS]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
Min.	103.7 (00)	104.8 (01)	103.3 (00)	93.06 (07)	90.6 (16)	85.8 (34)
Max.	109.6 (01)	107.8 (02)	114.4 (04)	111.6 (11)	112.6 (30)	116.5 (49)
Total	205.2	407.1	834.9	1536.1	3027.9	5800.6
Ave.	102.6	101.8	104.4	96.0	94.6	90.6
スカラ版						
Min.	168.2 (00)	182.8 (00)	189.1 (00)	185.0 (01)	153.4 (10)	123.2 (34)
Max.	179.7 (01)	188.3 (02)	210.4 (01)	206.7 (00)	195.8 (30)	169.5 (52)
Total	333.8	714.7	1513.5	2973.8	5180.5	8366.6
Ave.	166.9	178.7	189.2	185.9	161.9	130.7
命令処理性能 MIPS [MIPS]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
Min.	282.2 (00)	276.1 (01)	290.3 (00)	314.7 (09)	408.6 (12)	513.8 (24)
Max.	305.6 (01)	340.1 (03)	392.7 (04)	499.1 (13)	623.4 (17)	818.8 (22)
Total	565.6	1138.8	2599.3	6165.1	14822.1	35861.4
Ave.	282.8	284.7	324.9	385.3	463.2	560.3
スカラ版						
Min.	331.5 (00)	373.9 (00)	429.8 (06)	497.9 (05)	615.2 (05)	688.1 (56)
Max.	420.2 (01)	469.9 (03)	572.3 (02)	686.4 (10)	791.6 (09)	1027.9 (48)
Total	720.9	1632.2	3787.5	8705.9	20204.8	48016.6
Ave.	360.5	408.1	473.4	544.1	631.4	750.3
L2 キャッシュ・ミス率 L2-miss [%]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
Min.	3.27 (01)	2.57 (03)	2.20 (04)	1.31 (10)	1.00 (16)	0.49 (34)
Max.	3.54 (00)	3.19 (02)	2.84 (00)	2.52 (09)	1.95 (30)	1.21 (33)
Total	3.40	3.01	2.49	1.79	1.33	0.83
スカラ版						
Min.	1.89 (01)	1.62 (03)	1.35 (02)	1.08 (10)	0.73 (09)	0.41 (34)
Max.	2.67 (009)	2.29 (00)	1.92 (06)	1.55 (05)	1.09 (30)	0.82 (01)
Total	2.24	1.90	1.63	1.30	0.91	0.62

表 5.8 プロファイラ情報抜粋 (続き)

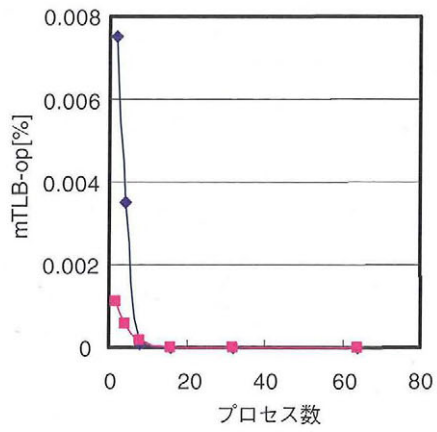
測定区間：TOTAL <span style="float: right;">() 内はランク</span>						
アドレス変換バッファ・ミス率 mTLB-op [%]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
Min.	0.0053 (01)	0.0026 (04)	0.0001 (00)	0.0000 (—)	0.0000 (—)	0.0000 (—)
Max.	0.0098 (00)	0.0044 (00)	0.0002 (01)	0.0000 (—)	0.0000 (—)	0.0000 (—)
Total	0.0075	0.0035	0.0001	0.0000	0.0000	0.0000
スカラ版						
Min.	0.0006 (00)	0.0005 (01)	0.0001 (01)	0.0000 (—)	0.0000 (—)	0.0000 (—)
Max.	0.0016 (01)	0.0007 (00)	0.0002 (00)	0.0000 (—)	0.0000 (—)	0.0000 (—)
Total	0.0011	0.0006	0.0002	0.0000	0.0000	0.0000
Memory Write Back [%]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
Min.	1.0696 (01)	0.9746 (03)	0.9109 (04)	0.5786 (10)	0.4299 (16)	0.2129 (34)
Max.	1.1347 (00)	1.1923 (01)	1.1334 (01)	1.0361 (09)	0.8127 (22)	0.5211 (33)
Total	1.1009	1.1190	1.0047	0.7553	0.5834	0.3595
スカラ版						
Min.	0.5340 (01)	0.5174 (03)	0.5040 (02)	0.4126 (07)	0.3164 (10)	0.1599 (34)
Max.	0.6442 (00)	0.6246 (00)	0.6315 (01)	0.5963 (11)	0.4802 (23)	0.3909 (12)
Total	0.5831	0.5692	0.5666	0.5205	0.4056	0.2455
浮動小数点演算命令の割合 Float [%]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
Min.	35.87 (01)	31.49 (03)	29.38 (02)	19.14 (07)	15.62 (16)	10.50 (34)
Max.	36.73 (00)	37.97 (01)	35.85 (01)	33.36 (05)	28.23 (30)	22.39 (33)
Total	36.29	35.75	32.12	24.92	20.43	16.17
スカラ版						
Min.	42.76 (01)	40.01 (03)	34.75 (02)	28.25 (07)	21.05 (10)	12.36 (34)
Max.	50.73 (00)	48.88 (00)	44.85 (06)	39.72 (05)	30.57 (12)	22.19 (01)
Total	46.31	43.79	39.96	34.16	25.64	17.42
Load/Store 命令の割合 [%]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
Min.	34.13 (00)	33.56 (03)	32.60 (05)	29.89 (07)	28.61 (07)	26.52 (03)
Max.	34.42 (01)	34.31 (02)	33.81 (01)	32.83 (11)	31.21 (25)	30.95 (12)
Total	34.28	33.99	33.02	31.33	29.95	28.82
スカラ版						
Min.	32.77 (01)	32.41 (01)	31.77 (02)	30.07 (07)	28.07 (09)	25.04 (48)
Max.	33.34 (00)	33.19 (00)	32.75 (07)	32.03 (11)	30.74 (24)	29.58 (41)
Total	33.02	32.73	32.23	31.24	29.47	



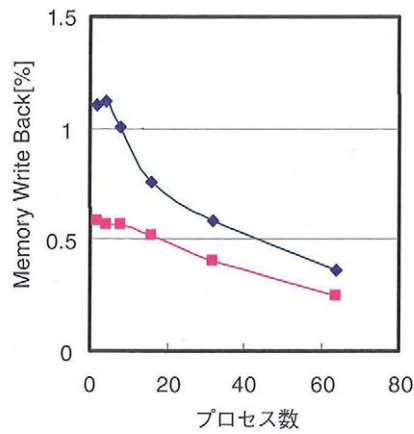
(a) 浮動小数点演算性能



(b) L2 キャッシュ・ミス率



(c) アドレス変換バッファ・ミス率



(d) Memory Write Back

図 5.4 プロファイル情報抜粋

表 5.9 測定区間“SPVCOR”における測定結果

測定区間：SPVCOR 時間積分500ステップ						
経過時間 [秒]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	1466.09	725.14	376.80	202.55	112.77	57.02
スカラ版	857.97	397.66	195.26	92.93	47.44	26.10
性能向上率	1.71	1.82	1.93	2.18	2.38	2.18
浮動小数点演算性能 FLOPS [MFLOPS]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	178.5	180.5	189.0	190.6	191.5	204.1
スカラ版	306.1	334.0	383.7	409.6	410.4	375.3
命令処理性能 MIPS [MIPS]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	311.6	319.2	339.4	348.7	365.0	429.5
スカラ版	517.9	575.2	650.4	702.2	734.3	757.9
L2 キャッシュ・ミス率 L2-miss [%]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	2.68	2.50	2.32	2.11	1.83	1.25
スカラ版	1.91	1.12	0.89	0.65	0.50	0.33
アドレス変換バッファ・ミス率 mTLB-op [%]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	0.0001	0.0001	0.0000	0.0000	0.0000	0.0000
スカラ版	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Memory Write Back [%]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	0.5875	0.6080	0.5932	0.5402	0.4904	0.3336
スカラ版	0.3259	0.3618	0.3004	0.1922	0.1716	0.1257
浮動小数点演算命令の割合 Float [%]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	57.28	56.56	55.68	54.67	52.47	47.51
スカラ版	59.10	59.10	59.00	58.34	55.89	49.51
Load/Store 命令の割合 [%]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	29.70	29.69	29.54	29.32	20.03	28.62
スカラ版	30.11	30.21	30.17	30.00	29.73	28.65

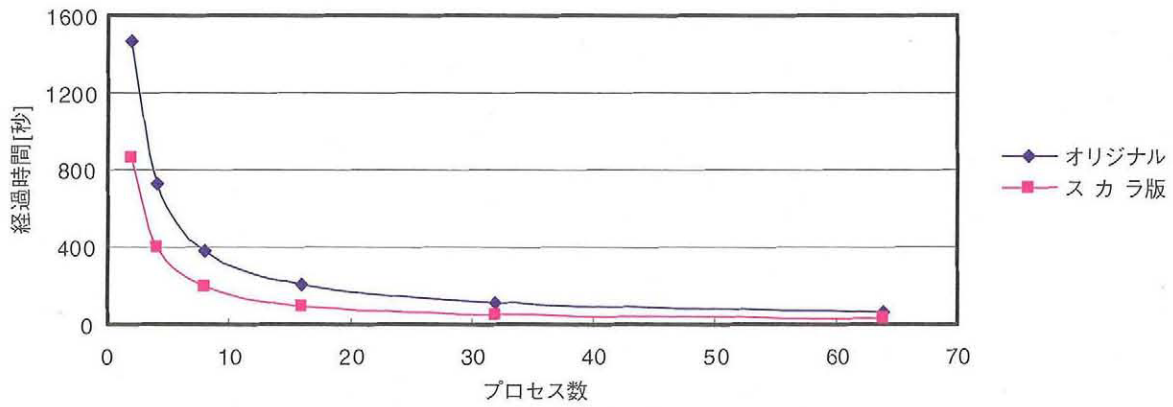


図 5.5 測定区間 “SPVCOR” における経過時間測定結果

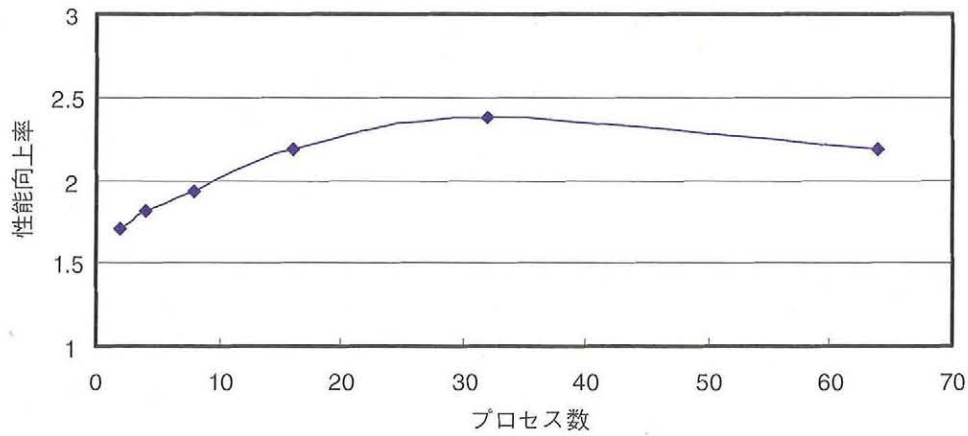
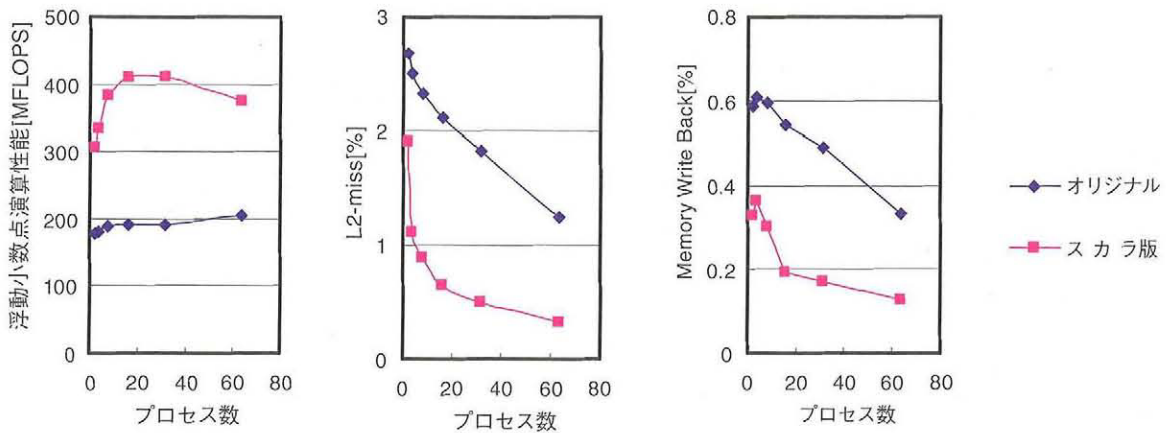


図 5.6 測定区間 “SPVCOR” における最適化による性能向上率



(a) 浮動小数点演算性能 (b) L2キャッシュ・ミス率 (c) Memory Write Back

図 5.7 測定区間 “SPVCOR” におけるプロファイラ情報抜粋

### 5.3.3. 測定区間“SPDRF1”における測定結果及び評価

測定区間“SPDRF1”では、各節点及び辺における勾配 $\nabla q_i$ の計算について、実行時間の測定及び実行状況の解析を行なった。表5.10、図5.8及び5.9にこの区間における測定及び解析結果を示す。測定区間“\_tet”、“\_prism”及び“\_pyr”は、それぞれ、サブルーチン“SPDRF1\_tet”、“SPDRF1\_pri”及び“SPDRF1\_pyr”における測定及び解析結果である。経過時間は、バリア同期を取ってルートプロセス（ランク0）における測定結果のみを示している。スカラ版の経過時間において、下段は性能向上率であり、オリジナルの実行に要した経過時間をスカラ版の実行に要した経過時間で除したものとして計算した。スカラ版における“SUM”は、リスト3.1の「変更後」に示されたDO200及びDO300の処理に要した経過時間であり、“node”は節点における勾配の計算に対応するDO200、“edge”は辺における勾配の計算に対応するDO300におけるプロファイラの測定結果である。DO200及びDO300は、サブルーチン“SPDRF1\_tet”、“SPDRF1\_pri”及び“SPDRF1\_pyr”を呼び出しているサブルーチン“DR”で、一括して実行される。従って、これら3つのサブルーチンにおける経過時間及び解析結果には、これらのDOループでの処理に関する部分は含まれない。このことから、オリジナルにおける測定区間“\_tet”、“\_prism”及び“\_pyr”とスカラ版における当該測定区間を直接比較することは出来ないことに注意されたい。

性能向上率は全てのプロセス数で2倍を越えており、高い最適化の効果が得られた。但し、“SUM”即ち、リスト3.1の「変更後」、DO200及びDO300の部分が実際の勾配の計算の倍以上かかっており、非常に効率が悪い。この部分では、L2キャッシュ・ミス率が3から5%と非常に高い値を示しており、メモリアクセスが効率悪化の原因となっていることがわかる。このため、DOループを分割せず、リスト2.1に示されているような再帰参照を含むコーディングを行うことも考えられる。そこで、実際にサブルーチン“SPDRF1\_tet”、“SPDRF1\_pri”及び“SPDRF1\_pyr”において、色分けの削除のみを行い再帰参照を含むループとした場合とスカラ版との簡単な比較を試みた。その結果を、表5.11及び図5.10に示す。表中、スカラ版はリスト3.1に示される3つのループで構成されたコーディングについての測定結果であり、再帰参照版がリスト2.1に示すようなループの分割を行わずに再帰参照を含むコーディングについて測定した結果である。再帰参照版は、サブルーチン“SPDRF1\_tet”、“SPDRF1\_pri”及び“SPDRF1\_pyr”以外の部分はオリジナルのままである。表には、オリジナル、スカラ版及び再帰参照版の経過時間及び8プロセス実行によるプロファイラ解析の内の主な解析結果を示した。再帰参照版については、参考のため各サブルーチンの経過時間も示してある。プロファイラの解析結果は、サブルーチン“SPDRF1\_tet”に対するものであり、浮動小数点演算性能 FLOPS、命令処理性能 MIPS、L2キャッシュ・ミス率 L2-miss 及びアドレス変換バッファ・ミス率 mTLB-op を示した。スカラ版については、表5.10同様、DO200及びDO300に対応する解析結果である“SUM”の“node”及び“edge”についても示した。再帰参照版における測定区間“SPDRF1”の処理に要する経過時間は、いずれのプロセス数においてもオリジナルに比べて短縮されているが、スカラ版ではさらに性能が向上していることがわかる。また、キャッシュ・ミス率は再帰参照版において優れるものの、浮動小数点演算性能ではスカラ版が優れる。このことから、スカラ版は総和演算部分でキャッシュ・ミスの頻発により性能が低下するものの、勾配の計算（リスト3.1 DO100）においてキャッシュ・ミスが低減され、全体としては再帰参照版に比べても性能が向上することが確認できた。この結果をもとに、スカラ版ではリスト3.1に示すようなループ分割を行う変更を採用した。他のサブルーチン“SPVCOR”等では、再帰参照版の方が良い性能を得られたため、ループ分割は行わなかった。このことについては、「6. スレッド版計算性能測定結果」においてより詳細を示す。

尚、サブルーチン“SPDRF1\_pri”の計算（測定区間“\_prism”）に関して2から16プロセスによる実行において900MFLOPSを越える高い実効性能が得られている。このサブルーチンが、三角柱要素の勾配を三つの三角錐要素に分割して計算していることを考えれば、サブルーチン“SPDRF1\_tet”において、いくつかの三角錐要素を融合させて同時に計算することによって、より高い性能が得られる可能性が示唆されたと言える。



表 5.10 測定区間“SPDRF1”における測定結果(続く)

測定区間：SPDRF1 時間積分500ステップ						
経過時間 [秒]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	3304.28	1532.20	894.27	569.08	311.82	172.20
_tet	2528.71	1182.35	672.64	366.58	205.63	106.33
_prism	769.68	343.53	215.97	196.74	100.89	60.30
_pyr	3.03	1.05	0.74	0.62	0.43	0.27
スカラ版	1572.96	739.90	397.72	206.16	105.09	57.83
	2.10	2.07	2.25	2.76	2.97	2.98
_tet	480.08	206.77	100.89	49.88	27.06	13.64
_prism	82.70	38.91	26.25	21.47	10.35	6.74
_pyr	0.46	0.18	0.14	0.11	0.08	0.05
SUM	1009.72	494.04	270.44	137.40	67.60	37.40
浮動小数点演算性能 [MFLOPS]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
_tet	80.5	77.4	79.3	79.4	81.3	86.9
_prism	135.6	129.5	132.1	132.7	137.4	148.2
_pyr	-----	-----	-----	-----	-----	-----
スカラ版						
_tet	275.3	324.9	381.4	395.5	366.9	290.1
_prism	984.7	960.6	962.6	943.0	831.4	671.9
_pyr	-----	-----	-----	-----	-----	-----
SUM node	90.4	105.0	126.8	148.3	196.7	280.9
edge	54.1	58.4	64.1	69.6	78.6	89.5
命令処理性能 [MIPS]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
_tet	176.7	170.3	174.7	175.9	181.6	200.3
_prism	278.3	266.5	271.8	272.9	282.7	304.9
_pyr	-----	-----	-----	-----	-----	-----
スカラ版						
_tet	420.0	496.4	588.0	625.3	613.9	560.3
_prism	1551.3	1508.8	1508.7	1474.8	1289.8	1184.8
_pyr	-----	-----	-----	-----	-----	-----
SUM node	228.1	250.2	287.8	321.9	392.2	495.0
edge	171.6	184.6	201.3	216.9	240.8	260.5
L2 キャッシュ・ミス率 L2-miss [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
_tet	5.08	4.84	4.68	4.60	4.36	3.52
prism	2.57	2.57	2.59	2.56	2.39	2.04
_pyr	-----	-----	-----	-----	-----	-----
スカラ版						
_tet	1.91	1.51	1.50	2.07	2.67	1.39
_prism	0.29	0.54	0.39	0.44	1.07	1.59
_pyr	-----	-----	-----	-----	-----	-----
SUM node	5.99	4.90	3.85	2.87	1.61	1.39
edge	5.47	4.81	4.16	3.59	2.99	2.37

表 5.10 測定区間“SPDRF1”における測定結果(続き)

測定区間：SPDRF 1						
アドレス変換バッファ・ミス率 mTLB-op [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
_tet	0.0115	0.0008	0.0001	0.0000	0.0000	0.0000
_prism	0.0002	0.0001	0.0000	0.0000	0.0000	0.0000
_pyr	-----	-----	-----	-----	-----	-----
スカラ版						
_tet	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
_prism	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
_pyr	-----	-----	-----	-----	-----	-----
SUM node	0.0024	0.0000	0.0000	0.0000	0.0000	0.0000
edge	0.0008	0.0000	0.0000	0.0000	0.0000	0.0000
Memory Write Back [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
_tet	2.8802	2.8846	2.9182	2.9772	2.9104	2.3781
_prism	1.7428	1.7487	1.7569	1.7467	1.6265	1.4421
_pyr	-----	-----	-----	-----	-----	-----
スカラ版						
_tet	0.5948	0.7408	1.0894	1.7989	2.4133	1.2030
_prism	0.1699	0.2090	0.2353	0.2950	0.9290	1.3468
_pyr	-----	-----	-----	-----	-----	-----
SUM node	0.4794	0.4491	0.4297	0.4343	0.4591	0.9211
edge	0.9564	0.9587	0.9483	0.9204	0.8543	0.7537
浮動小数点演算命令の割合 Float [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
_tet	45.57	45.43	45.38	45.17	44.77	43.39
_prism	48.71	48.61	48.64	48.62	48.61	48.61
_pyr	-----	-----	-----	-----	-----	-----
スカラ版						
_tet	65.55	65.44	64.86	63.25	59.77	51.78
_prism	63.48	63.67	63.80	63.94	64.46	56.71
_pyr	-----	-----	-----	-----	-----	-----
SUM node	40.51	41.99	44.05	46.06	50.16	56.75
edge	31.55	31.64	31.84	32.11	32.64	34.35
Load/Store 命令の割合 [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
_tet	46.08	46.11	46.29	46.57	46.62	45.41
_prism	45.45	45.42	45.45	45.46	45.55	45.80
_pyr	-----	-----	-----	-----	-----	-----
スカラ版						
_tet	27.61	28.31	29.81	32.96	35.89	31.49
_prism	33.25	32.32	31.79	31.36	31.57	34.16
_pyr	-----	-----	-----	-----	-----	-----
SUM node	48.27	47.38	45.79	43.61	38.47	34.76
edge	49.96	50.07	50.17	50.24	50.28	47.43

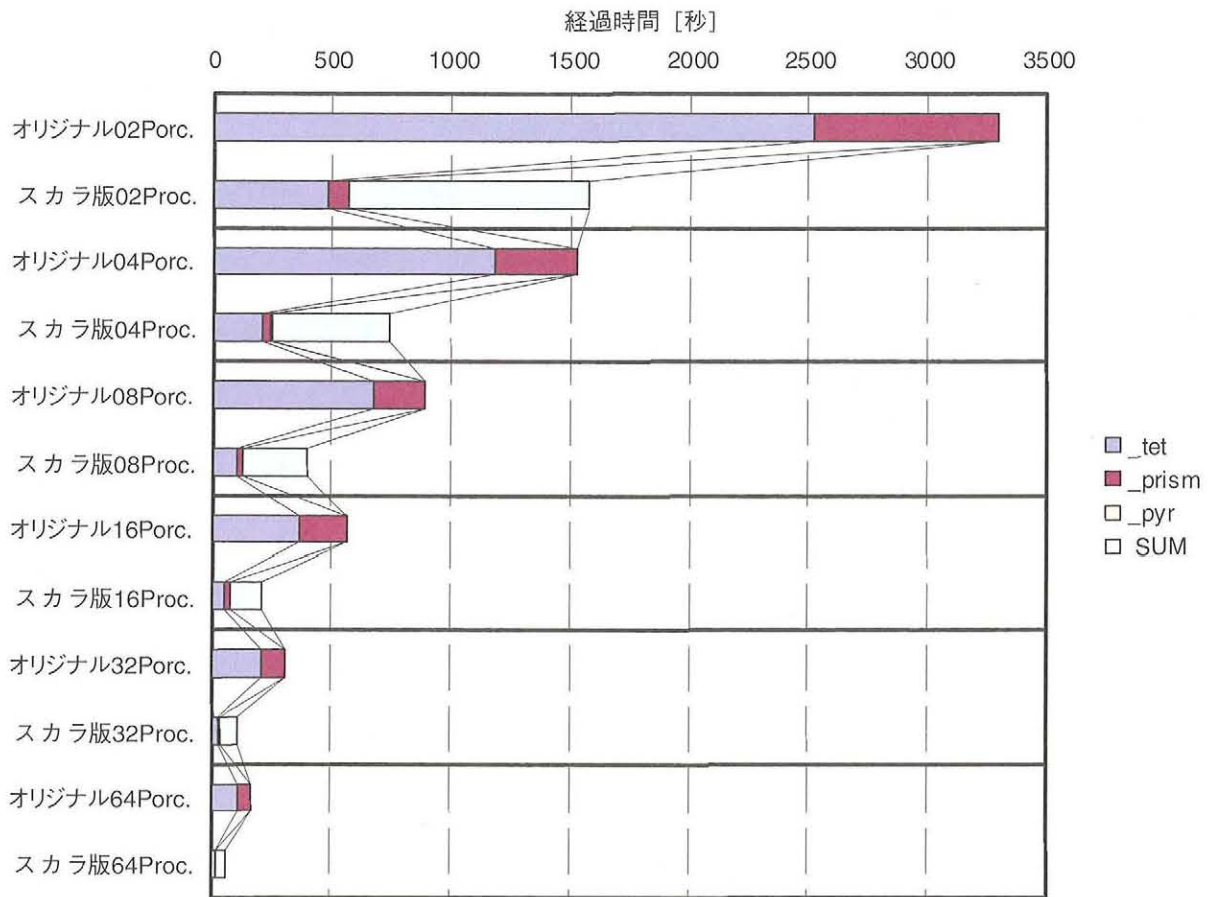


図 5.8 測定区間 “SPDRF1” における測定結果

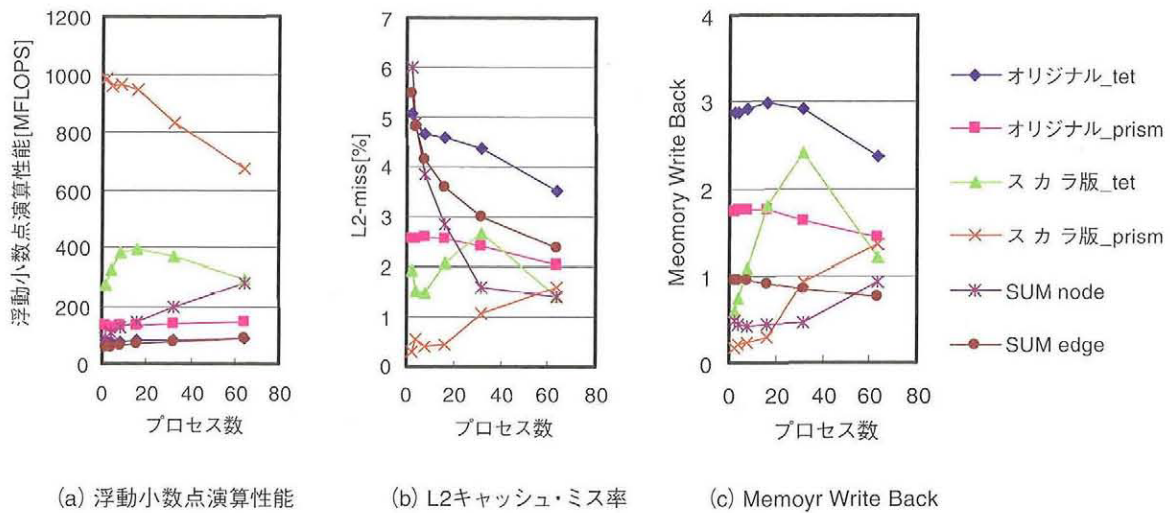


図 5.9 測定区間 “SPDRF1” におけるプロファイラ情報抜粋

表 5.11 測定区間“SPDRF1”における再帰参照版とスカラ版の性能比較

経過時間 [秒]						
測定区間：SPDRF1						
時間積分500ステップ						
Version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	3304.28	1532.20	894.27	569.08	311.82	172.20
スカラ版	1572.96	739.90	397.72	206.16	105.09	57.83
再帰参照版	2394.84	1105.34	518.20	272.34	117.07	65.04
_tet	2124.20	980.82	446.11	201.07	83.48	40.20
_prism	265.67	118.21	66.74	65.82	26.24	19.48
_pyr	2.22	0.82	0.58	0.46	0.28	0.21

プロファイラの主な解析結果				
解析対象：サブルーチン “SPDRF1_tet”				
8 process 実行				
Version	FLOPS [MFLOPS]	MIPS [MIPS]	L2-miss [%]	mTLB-op [%]
オリジナル	79.3	174.7	4.68	0.0001
スカラ版	254.7	580.8	1.53	0.0000
Sum node	115.1	254.7	4.79	0.0000
edge	55.3	171.0	5.60	0.0000
再帰参照版	120.8	256.8	2.83	0.0000

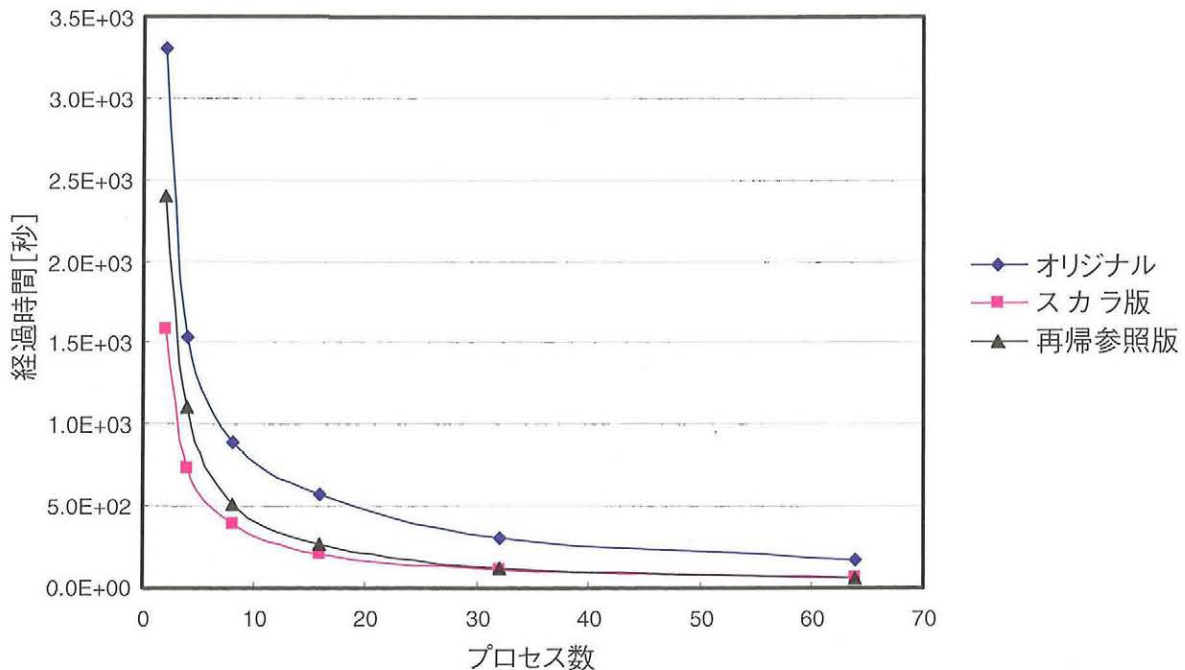


図 5.10 測定区間“SPDRF1”における再帰参照版とスカラ版の性能比較

### 5.3.4. 測定区間 “LIMITER” における測定結果及び評価

測定区間 “LIMITER” では、Venkatakrishnan の制限関数  $\Psi_i$  の計算について、経過時間の測定及び実行状況の解析を行なった。この測定区間は、二つのサブルーチン “LIMITER1” 及び “LIMITER2” によって構成される。表5.12及び図5.11から5.13にこの区間における測定及び解析結果を示す。測定区間 “LIMITER1” は、サブルーチン “LIMITER1” に関する測定及び解析の結果であり、測定区間 “LIMITER2” は、サブルーチン “LIMITER2” に関する測定及び解析の結果である。経過時間は、サブルーチン “LIMITER1” 及び “LIMITER2” の実行に要した時間の和であり、バリア同期を取ってルートプロセス（ランク0）における測定結果のみを示している。スカラ版の経過時間において、下段は性能向上率であり、オリジナルの実行に要した経過時間をスカラ版の実行に要した経過時間で除したものとして計算した。

性能向上率は、2プロセス実行で1.9倍と2倍を下回ったものの、他のプロセス数においては全て2倍を上回る結果が得られた。測定区間 “LIMITER1” に比べ測定区間 “LIMITER2” における性能向上率は若干下回るが、これは測定区間 “LIMITER2” では JTAS オリジナルにおいて既に、200MFLOPS を越える性能が出ていることによる。スカラ版では、さらなる性能向上により2プロセスによる実行を除き400MFLOPS を越える性能を得ることが出来た。一方で測定区間 “LIMITER1” では性能が向上したにもかかわらず、100MFLOPS を越えないケースが半数にのぼる。この原因の一つとしては、測定区間 “LIMITER1” において浮動小数点演算の割合が高々35%程であることがあげられる。また、L2 キャッシュ・ミス率がスカラ版においてすら3%前後と高いことも影響していると考えられる。

### 5.3.5. 測定区間 “LU-SGS” における測定結果及び評価

測定区間 “LU-SGS” では、LU-SGS 法により解を求める計算について、経過時間の測定及び実行状況の解析を行なった。表5.13及び図5.14から5.20にこの区間における測定及び解析結果を示す。各測定区間は、それぞれ同じ名前を持つサブルーチンに対応する。経過時間の測定において、サブルーチン “DIAGONAL” の経過時間を測定する区間 “DIAGONAL” 及びサブルーチン “UPDATEQ” の経過時間を測定する区間 “UPDATEQ” に対してのみバリア同期を取ってルートプロセス（ランク0）における測定結果を示した。それ以外の測定区間ではバリア同期が取れないため、全プロセスの経過時間を個別に測定し、その最小値を上段に、最大値を下段にそのプロセスのランクと共に示した。経過時間においてオリジナルの下段及びスカラ版の中段には、それぞれの2プロセス実行の経過時間を1とした場合の並列化による加速率を示す。また、スカラ版の経過時間において、下段は性能向上率であり、オリジナルの実行に要した経過時間をスカラ版の実行に要した経過時間で除したものとして計算した。測定区間 “COMMUNICAT.” は、測定区間 “FORWARD” と “BACKWARD” の和であり、それぞれ前進スイープ後及び後退スイープ後に行われるプロセス間通信に要した時間の合計である。

測定区間 “LU-SGS” において、スカラ版における最適化による効果は2プロセス実行で1.95倍と全体を上回る結果となったものの、プロセス数が増加するとともにその効果は低下し、64プロセスによる実行では1割程度の性能向上に留まった。この原因の一つに、オリジナルの2プロセス実行において、測定区間 “DIAGONAL” を除くとアドレス変換バッファ・ミス率  $mTLB-op$  が非常に大きいことがあげられる。オリジナルでは、プロセス数が多くなるに従い1プロセスに割当たる格子点数が減少するため、アドレス変換バッファ・ミス率は低下し、このため2プロセス実行と比較した場合の相対的な性能向上が著しい。事実、オリジナルの加速率は、16プロセスまで理論値を上回るスーパーリニア値を示している。32プロセスにおいても、理論値比91%の加速率である。一方、スカラ版では、最適化の効果により2プロセス実行におけるアドレス変換バッファ・ミス率の低減が図られたことにより、プロセス数の増大によりミス率が低減する割合も減少し、この結果加速率はオリジナルに比べると低い。これにより、プロセス数の増加とともに、スカラ版における最適化の効果が減少したと考えられる。

表 5.12 測定区間 “LIMITER” における測定結果 (続く)

測定区間：LIMITER 時間積分500ステップ						
経過時間 [秒]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	1673.68	837.70	439.68	239.21	126.88	50.94
LIMITER1	525.33	268.22	141.60	77.51	38.81	12.07
LIMITER2	1148.35	569.46	298.08	161.70	88.07	38.87
スカラ版	882.50	376.67	172.61	75.69	38.52	20.17
性能向上率	1.90	2.22	2.55	3.16	3.30	2.53
LIMITER1	219.65	83.64	33.80	13.72	6.51	3.19
性能向上率	2.39	3.21	4.19	5.65	5.96	3.78
LIMITER2	662.85	293.03	138.61	61.97	32.01	16.98
性能向上率	1.73	1.94	2.15	2.61	2.75	2.29
浮動小数点演算性能 [MFLOPS]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
LIMITER1	40.6	39.3	40.0	40.4	43.4	48.7
LIMITER2	205.2	205.6	213.3	215.8	227.2	304.4
スカラ版						
LIMITER	94.2	124.6	164.5	171.7	98.8	64.0
LIMITER2	359.3	419.8	496.2	541.6	547.2	469.0
命令処理性能 [MIPS]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
LIMITER1	122.7	119.6	122.9	125.0	149.2	321.2
LIMITER2	332.6	333.4	346.2	351.5	373.4	530.6
スカラ版						
LIMITER1	265.8	352.6	476.0	513.2	417.9	501.6
LIMITER2	575.3	672.6	798.0	881.6	921.2	910.1
L2 キャッシュ・ミス率 L2-miss [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
LIMITER1	8.61	8.12	7.96	7.52	4.85	0.80
LIMITER2	2.10	1.96	1.88	1.87	1.75	0.83
スカラ版						
LIMITER1	3.34	2.68	3.30	4.89	1.70	0.52
LIMITER2	0.82	0.54	0.34	0.29	0.53	0.33

表 5.12 測定区間 “LIMITER” における測定結果 (続き)

測定区間：LIMITER						
アドレス変換バッファ・ミス率 mTLB-op [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
LIMITER1	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
LIMITER2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
スカラ版						
LIMITER1	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000
LIMITER2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
LIMITER Write Back [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
LIMITER1	3.2075	3.2874	3.4205	3.3200	2.2490	0.4167
LIMITER2	0.5887	0.5977	0.5932	0.6460	0.6278	0.3201
スカラ版						
LIMITER1	1.3937	1.3277	1.8420	2.6606	0.9278	0.3121
LIMITER2	0.2356	0.1633	0.1100	0.1096	0.2629	0.1833
浮動小数点演算命令の割合 FLOAT [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
LIMITER1	33.09	32.83	32.58	32.32	29.05	15.15
LIMITER2	61.70	61.65	61.62	61.39	60.84	57.37
スカラ版						
LIMITER1	35.43	35.35	34.56	33.46	23.65	12.76
LIMITER2	62.46	62.41	62.17	61.43	59.40	51.53
Load/Store 命令の割合 [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
LIMITER1	46.30	46.51	46.75	46.07	42.58	32.92
LIMITER2	33.46	33.51	33.64	33.93	34.60	35.45
スカラ版						
LIMITER1	48.50	49.39	50.31	46.95	35.86	30.34
LIMITER2	33.98	34.21	34.66	35.57	36.52	34.98

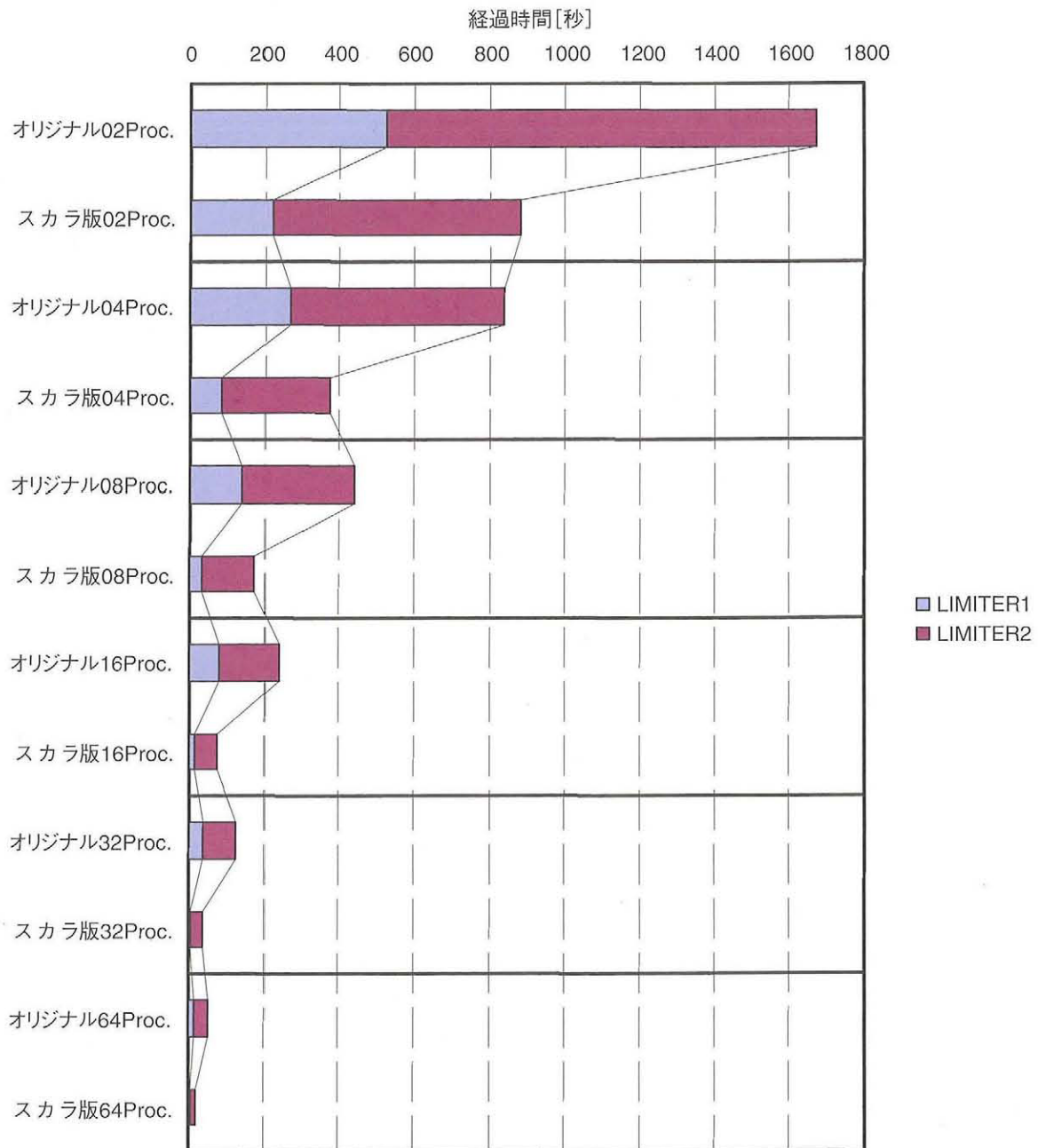


図 5.11 測定区間“LIMITER”における測定結果



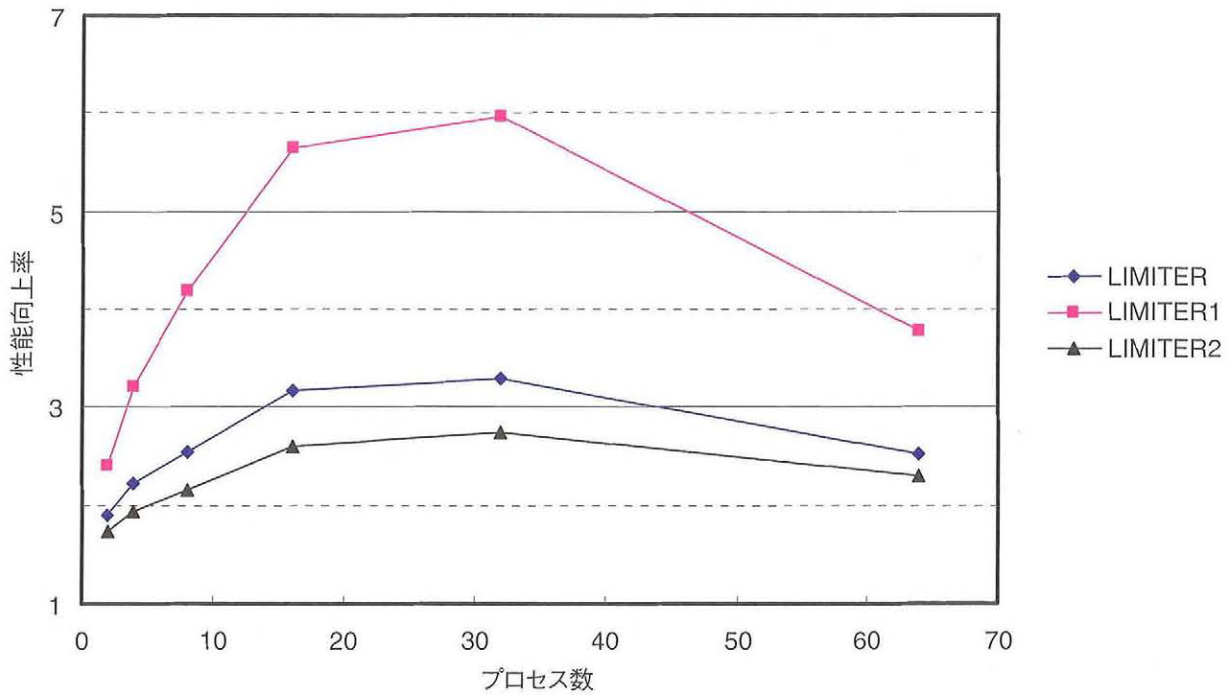


図 5.12 測定区間 “LIMITER” における最適化による性能向上率

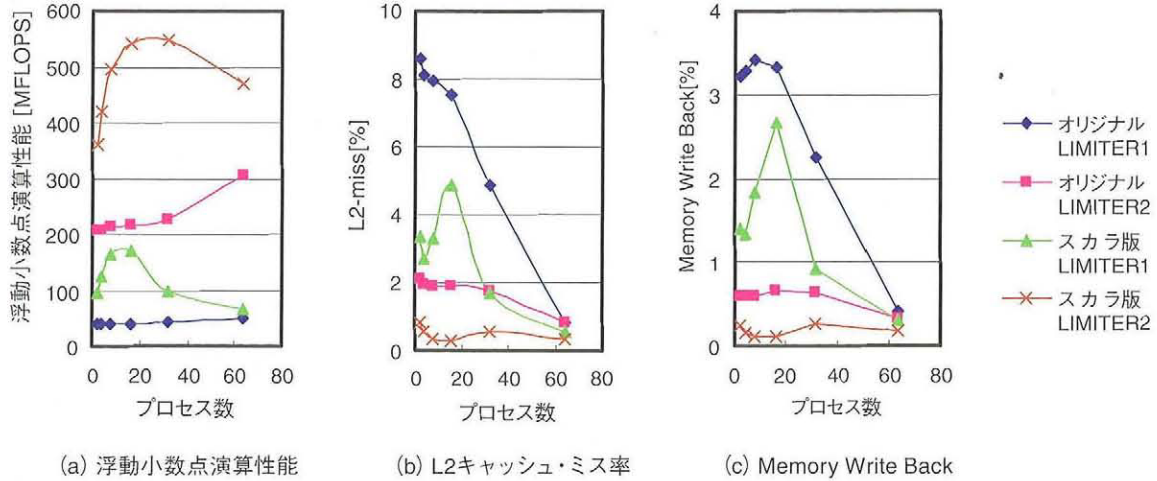


図 5.13 測定区間 “LIMITER” におけるプロファイラ情報抜粋

全般的には、アドレス変換バッファ・ミス率の低減が性能向上の要因であると言えるが、測定区間“DIAGONAL”においては、オリジナルにおいてもアドレス変換バッファ・ミス率は高くない。この区間では、L2 キャッシュ・ミス率 L2-miss が半減しており、これが性能向上の要因である。

測定区間“SWEPL”及び“SWEPR”における性能向上は、特に顕著である。サブルーチン“SWEPL”及び“SWEPR”は、プログラミング上は単純な総和演算を行うルーチンであるため、配列の定義参照を連続化したことが極めて大きな性能向上に繋がった。一方で、これら2つのサブルーチンにおいても浮動小数点演算性能は160MFLOPS程度に留まった。

8 プロセス実行において、スカラ版がオリジナル比1.51倍の性能を得ている理由をプロファイラの解析結果から説明することは難しい。特に、測定区間“SIGMALEFT”及び“SWEPL”において、スカラ版のL2 キャッシュ・ミス率及びアドレス変換バッファ・ミス率は、オリジナルのそれと比べて大きい。それにも関わらず、いずれの区間においても処理に要した経過時間は短縮されており、浮動小数点演算性能 FLOPS も向上している。この性能向上は、浮動小数点演算命令の割合が増加していることから、実際には、Load/Store 命令数が減少していることにより得られたものであると考えられる。スカラ版の最適化において、浮動小数点演算命令を増減させるような変更は加えていない。従って、浮動小数点演算命令の割合が増加するためには、他の命令が減少していなければならない。オリジナルとスカラ版の Load/Store 命令の割合は、僅かに減少しているもののほぼ同じであるから、オリジナルに比べてスカラ版における Load/Store 命令の絶対数は半減していると言える。L2 キャッシュ・ミス率は Load/Store 命令に対するキャッシュ・ミスを起こした割合であるから、同じミス率で命令数が半減すれば、キャッシュ・ミスの回数も半減していることになる。このことから、8 プロセスにおける性能向上は、キャッシュ・ミス率の改善ではなく、キャッシュ・ミスの起こる回数が減少したことによるものと推定できる。プロファイラによって得られる情報からは、これ以上の性能向上要因の確定は困難である。

尚、測定区間“UPDATEQ”においては、スカラ版の性能がオリジナル版に比べ低下している。これは、サブルーチン“UPDATEQ”において、新しいタイムステップでの保存量ベクトルの計算

$$\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n + \Delta\mathbf{Q}_i^n$$

をする際に、変化分  $\Delta\mathbf{Q}^n$  は超平面を考慮して節点番号を変更した配列に保存されているのに対して、 $\mathbf{Q}^{n+1}$  は、オリジナルのままであることによる。このため、オリジナルでは連続であった配列の参照が、スカラ版ではインデックス参照となっており、L2 キャッシュ・ミス率やアドレス変換バッファ・ミス率が増加し、性能が低下している。この性能低下は、LU-SGS 法に関わる部分のみならず、JTAS 全体においても配列の保存方法を超平面に対応した方法に変更することで、避けることが可能である。しかし、サブルーチン“UPDATEQ”の処理に要する時間は LU-SGS 法に関わる部分の計算時間の高々数%程度であり、JTAS 全体の実行時間に比べれば僅かである。従って、JTAS 全体において配列の保存方法を変更するために必要となる作業量を勘案すれば、このオーバーヘッドによる性能低下は許容範囲であると考えられる。尚、超平面を考慮した節点番号の付け替えに対しては、これ以外にも LU-SGS 法の計算において参照される保存量ベクトル  $\mathbf{Q} = [\rho, \rho u, \rho v, \rho w, e]^T$  や検査体積の体積及び表面積等を、計算に先立ってオリジナルの配列から超平面を考慮した配列にコピーする必要がある。そのオーバーヘッドも存在する。このオーバーヘッドについては、「6. スレッド版測定結果及び評価」において触れる。ここでは、参照する配列をコピーせずに、オリジナルの配列をそのまま参照することとした場合と、コピーした場合で性能がどの程度変わるか調べた結果を示す。表5.14及び図5.21に参照される配列をコピーして節点番号を付け替えた場合を“reordering”として、コピーはせずオリジナルの配列をそのまま参照した場合を“not”として、測定区間“LU-SGS”の処理に要した経過時間を示す。参考のために、オリジナルについても同様に経過時間を示す。表5.14及び図5.21より、参照される保存量ベクトル  $\mathbf{Q}$  等をコピーするかしないかに関わらず、定義される配列の保存方法を超平面を考慮した形に変更すれば、性能が向上することがわかる。また、8 プロセスま

での比較的少数のプロセスによる実行では、配列をコピーした (“reordering”) 方がオリジナルのまま (“not”) より若干ではあるが性能が良いという結果が得られた。16プロセス以上の測定結果では、“reordering” と “not” の差は高々数%であり、通常運用状態における測定であることも考え合わせればこの差は測定誤差の範囲であり、この結果からどちらの性能が良いかを判断することは適当ではない。従って、この測定に使用したデータの1プロセス当たりの格子点数が、実際の解析に使用されるその1プロセス当たりの格子点数と同程度になるのが、数プロセス程度の低並列度による実行においてであることを踏まえれば、参照する配列についてもコピーした方が良いと言える。

最後に、あるハイパー面内での節点の計算順序と性能の関係について調べた結果を表5.15及び図5.22に示す。表中、“sorted” ではサブルーチン “SIGMALEFT” 及び “SIGMARIGHT” において、その辺が含む節点の番号が連続となるように辺の計算順序をソートして計算を行ったものであり、“not sorted” はオリジナルの辺番号の昇順に計算を行ったものである。また、“as is” はオリジナルで計算されている順序を再現したものである (表 A.3.15参照)。但し、色分けのための二重ループは再現していない。測定区間は “LU-SGS” であり、その処理に要した経過時間が示されている。参考値としてオリジナルにおける経過時間の測定値も示した。この測定結果では、2プロセスにおける計算順序 “as is” において若干の性能低下が見られるものの、その他のケースにおいて経過時間の差は僅かであり、有意な差は見られなかった。

### 5.3.6. 測定区間 “COMMUNICATION” における測定結果及び評価

測定区間 “COMMUNICATION” では、プロセス間通信に要する経過時間の測定を行なった。この測定区間には、さらに7つの測定区間が含まれる。これらの測定区間の具体的範囲については、表5.2を参照されたい。表5.16及び図5.23にこの区間において通信に要した経過時間を測定した結果を示す。全ての測定区間についてバリア同期を取って、ルートプロセス (ランク0) における測定結果のみを示している。スカラ版における最適化では、プロセス間通信に関する変更は行っていない。オリジナルとスカラ版で経過時間が異なるのは測定誤差によるものである。

既に、「5.3.1全体の測定結果及び評価」で示した通り、ここでは、プロセス間通信によるオーバーヘッドについての詳しい検討は行わない。参考値として示すにとどめる。

表 5.13 測定区間“LU-SGS”における測定結果 (1/3)

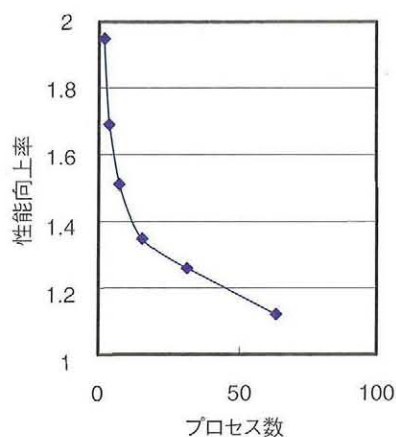
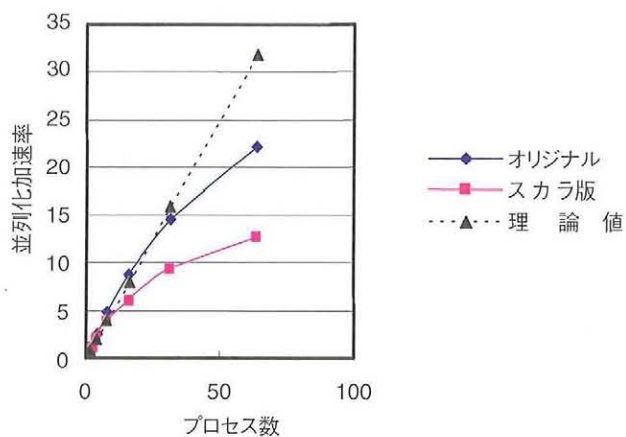
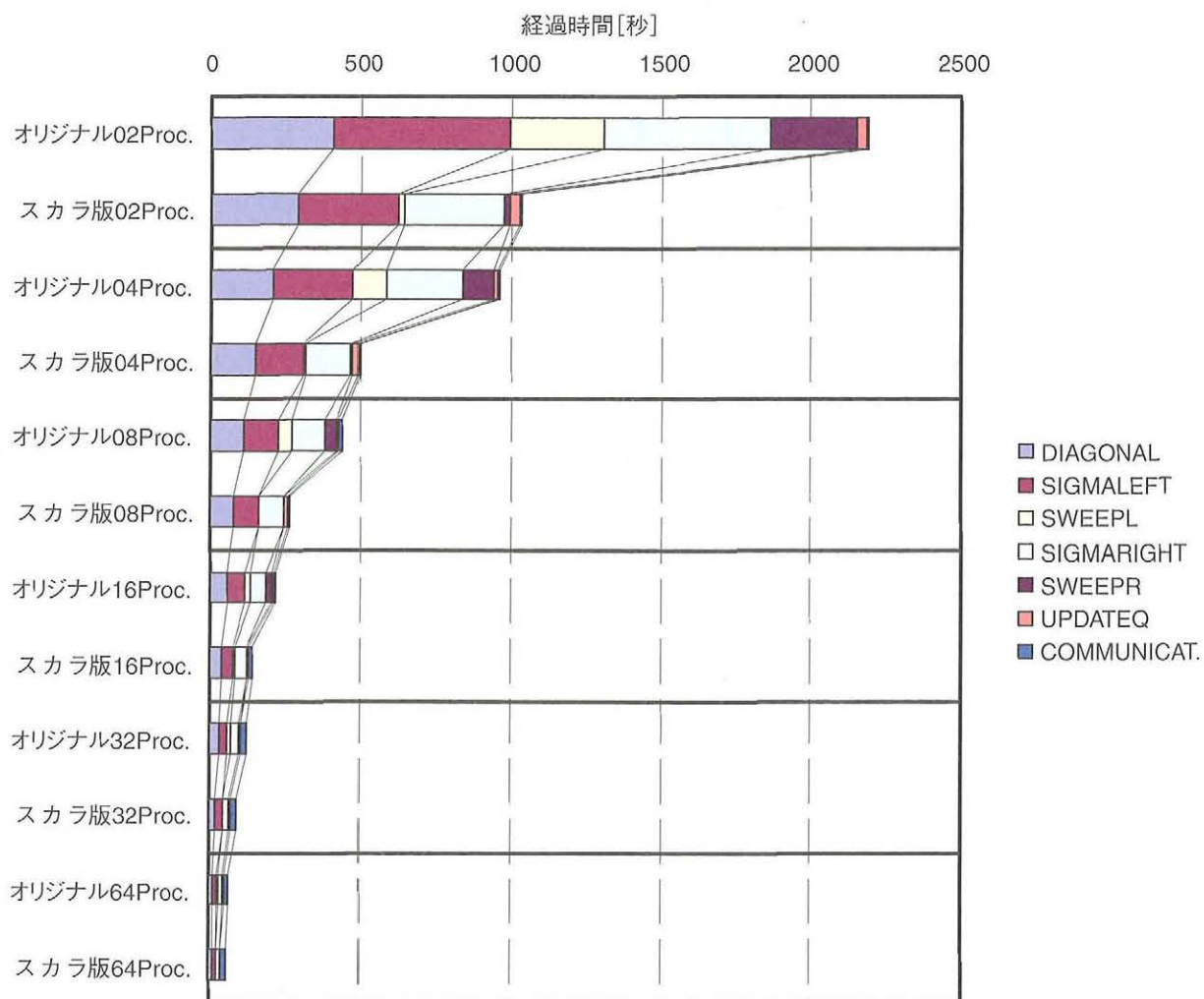
測定区間：LU-SGS 時間積分500ステップ						
経過時間 [秒]						上段：最小値 下段：最大値 ( ) 内：ランク
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル 加速率	2425.57 1.00	1069.97 2.27	515.90 4.70	278.49 8.71	166.60 14.56	109.47 22.16
DIAGONAL	411.22	210.55	110.16	59.75	32.19	15.24
SIGMALEFT	560.63 (00)	241.05 (00)	113.78 (02)	45.00 (03)	20.71 (10)	9.69 (22)
	585.04 (01)	266.67 (02)	120.07 (01)	55.64 (09)	28.52 (30)	13.96 (33)
SWEEPL	212.63 (00)	76.21 (00)	28.10 (00)	10.36 (03)	4.27 (16)	1.69 (27)
	307.43 (01)	108.20 (03)	39.73 (01)	19.23 (10)	8.03 (19)	3.88 (01)
SIGMARIGHT	532.52 (00)	227.94 (00)	109.55 (06)	43.69 (03)	20.38 (10)	9.45 (27)
	558.69 (01)	252.46 (02)	115.70 (01)	54.20 (09)	28.13 (03)	14.21 (52)
SWEEPR	201.78 (00)	71.21 (00)	25.97 (00)	9.58 (03)	4.00 (16)	1.57 (39)
	295.80 (01)	101.49 (03)	37.24 (01)	18.21 (10)	7.73 (19)	3.77 (13)
UPDATEQ	28.33	13.96	7.85	4.50	2.50	1.48
COMMUNICAT.	5.41	7.85	10.21	11.37	14.27	15.08
スカラ版 加速率 性能向上率	1243.24 1.00 1.95	632.95 1.96 1.69	341.65 3.63 1.51	206.88 6.01 1.35	132.50 9.38 1.26	97.95 12.69 1.12
DIAGONAL	293.04	146.45	78.06	39.09	21.34	11.85
SIGMALEFT	321.50 (00)	148.71 (03)	73.04 (00)	35.20 (03)	17.10 (10)	8.36 (34)
	331.99 (01)	162.08 (02)	81.29 (04)	41.35 (09)	22.66 (27)	12.45 (33)
SWEEPL	13.36 (00)	6.34 (00)	3.18 (00)	1.63 (03)	0.81 (10)	0.40 (34)
	16.94 (01)	7.63 (02)	4.35 (04)	2.38 (10)	1.26 (27)	0.77 (01)
SIGMARIGHT	319.34 (00)	148.52 (03)	72.93 (00)	35.05 (03)	16.65 (10)	7.56 (34)
	331.86 (01)	149.26 (01)	81.29 (06)	41.25 (09)	22.18 (27)	11.60 (09)
SWEEPR	14.37 (00)	6.78 (00)	3.39 (00)	1.74 (03)	0.85 (16)	0.41 (27)
	17.82 (01)	8.20 (02)	4.57 (04)	2.51 (10)	1.34 (27)	0.78 (01)
UPDATEQ	35.78	17.06	9.91	5.72	3.06	1.92
COMMUNICAT.	5.53	8.45	9.50	12.35	16.99	18.10
浮動小数点演算性能 FLOPS [MFLOPS]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
DIAGONAL	71.1	68.9	69.9	68.8	66.1	59.1
SIGMALEFT	72.2	80.5	98.5	120.2	140.6	156.8
SWEEPL	68.9	85.6	108.4	129.6	144.2	154.6
SIGMARIGHT	77.4	85.0	100.0	111.9	115.7	104.2
SWEEPR	68.6	77.6	108.4	104.2	108.7	102.9
UPDATEQ	173.6	135.3	98.1	83.3	10.3	25.8
スカラ版						
DIAGONAL	96.5	100.6	104.5	101.3	88.6	63.8
SIGMALEFT	154.5	163.7	169.4	169.0	165.1	162.1
SWEEPL	153.8	163.1	169.4	168.1	165.1	162.5
SIGMARIGHT	154.8	163.5	168.4	166.6	151.3	121.5
SWEEPR	153.6	161.9	165.6	163.9	144.5	114.9
UPDATEQ	20.1	18.0	18.4	24.5	29.6	55.1

表 5.13 測定区間 “LU-SGS” における測定結果 (2/3)

測定区間：LU-SGS						
命令処理性能 MIPS [MIPS]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
DIAGONAL	130.7	127.5	130.6	131.9	142.4	259.9
SIGMALEFT	341.0	345.1	353.7	391.7	406.2	399.2
SWEEPL	480.6	445.9	451.3	442.7	418.6	393.8
SIGMARIGHT	356.1	364.5	371.2	392.1	410.6	442.9
SWEEPR	500.9	459.7	457.4	442.9	452.3	474.1
UPDATEQ	169.3	223.4	357.8	538.2	682.5	768.4
スカラ版						
DIAGONAL	174.4	182.8	192.8	194.2	200.9	269.9
SIGMALEFT	265.8	283.0	296.2	300.3	294.0	290.9
SWEEPL	264.6	281.5	295.1	298.6	292.7	290.9
SIGMARIGHT	266.2	282.3	298.8	323.2	351.1	411.3
SWEEPR	264.2	280.0	297.1	323.0	353.5	429.8
UPDATEQ	139.8	199.9	349.8	523.8	677.1	779.1
L2 キャッシュ・ミス率 L2-miss [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
DIAGONAL	12.10	11.46	11.31	10.73	7.44	1.58
SIGMALEFT	4.12	3.23	2.89	2.60	2.73	2.65
SWEEPL	3.10	2.69	2.89	3.11	3.09	2.67
SIGMARIGHT	3.85	3.24	2.80	1.82	1.25	0.68
SWEEPR	3.54	3.47	2.63	1.58	1.14	0.70
UPDATEQ	6.11	2.76	1.02	0.53	0.43	0.52
スカラ版						
DIAGONAL	6.14	5.23	4.85	4.24	2.88	1.05
SIGMALEFT	3.47	3.39	3.71	3.35	2.84	2.67
SWEEPL	3.48	3.46	3.67	3.44	2.83	2.71
SIGMARIGHT	3.33	2.87	2.38	1.78	1.17	0.59
SWEEPR	3.41	2.97	2.38	1.75	1.11	0.56
UPDATEQ	9.48	4.71	1.73	1.07	0.75	0.68
アドレス変換バッファ・ミス率 mTLB-op [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
DIAGONAL	0.0009	0.0007	0.0002	0.0000	0.0000	0.0000
SIGMALEFT	0.0270	0.0189	0.0007	0.0000	0.0000	0.0000
SWEEPL	0.0196	0.0132	0.0005	0.0000	0.0000	0.0000
SIGMARIGHT	0.0251	0.0170	0.0010	0.0000	0.0000	0.0000
SWEEPR	0.0192	0.0133	0.0007	0.0000	0.0000	0.0000
UPDATEQ	0.0002	0.0003	0.0000	0.0000	0.0000	0.0000
スカラ版						
DIAGONAL	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
SIGMALEFT	0.0093	0.0061	0.0018	0.0001	0.0001	0.0001
SWEEPL	0.0091	0.0061	0.0018	0.0001	0.0001	0.0001
SIGMARIGHT	0.0082	0.0053	0.0012	0.0000	0.0000	0.0000
SWEEPR	0.0079	0.0052	0.0012	0.0000	0.0000	0.0000
UPDATEQ	0.0038	0.0014	0.0001	0.0000	0.0000	0.0000

表 5.13 測定区間“LU-SGS”における測定結果(3/3)

測定区間：LU-SGS						
Memory Write Back [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
DIAGONAL	2.0273	2.2597	2.8169	3.0636	2.1197	0.4409
SIGMALEFT	0.2420	0.2058	0.2364	0.2998	0.3686	0.4142
SWEEPL	0.2045	0.2179	0.2998	0.3999	0.4323	0.4578
SIGMARIGHT	0.2625	0.5543	0.5714	0.2952	0.2672	0.1858
SWEEPR	1.0072	1.5494	1.0383	0.3067	0.2365	0.1867
UPDATEQ	1.9736	0.9346	0.3399	0.1370	0.0730	0.0735
スカラ版						
DIAGONAL	0.9250	0.9691	1.2634	1.3606	0.9780	0.3001
SIGMALEFT	0.2469	0.3008	0.4212	0.4503	0.3916	0.4365
SWEEPL	0.2432	0.3066	0.4049	0.4699	0.3952	0.4470
SIGMARIGHT	0.4160	0.4160	0.3338	0.2866	0.2200	0.1021
SWEEPR	0.4644	0.4378	0.3608	0.3044	0.2173	0.1018
UPDATEQ	3.3164	1.5809	0.4888	0.2294	0.1221	0.1099
浮動小数点演算命令の割合 FLOAT [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
DIAGONAL	54.37	54.00	53.49	52.16	46.43	22.72
SIGMALEFT	21.16	11.77	27.84	30.70	34.61	39.27
SWEEPL	14.34	19.20	24.01	29.28	34.46	39.26
SIGMARIGHT	21.73	23.33	26.92	28.54	28.18	23.54
SWEEPR	13.70	16.88	19.84	23.52	24.02	21.70
UPDATEQ	14.33	6.05	2.74	1.55	1.51	3.36
スカラ版						
DIAGONAL	55.32	55.02	54.19	52.16	44.12	23.62
SIGMALEFT	58.11	57.85	57.19	56.28	56.15	55.70
SWEEPL	58.14	57.94	57.41	56.30	56.38	55.87
SIGMARIGHT	58.16	57.92	56.35	51.55	43.08	29.56
SWEEPR	58.14	57.81	55.73	50.73	40.88	26.75
UPDATEQ	14.36	8.98	5.26	4.68	4.37	7.07
Load/Store 命令の割合 [%]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル						
DIAGONAL	23.65	24.22	25.39	25.99	25.78	25.88
SIGMALEFT	31.28	30.90	29.98	29.28	28.22	26.90
SWEEPL	31.98	30.97	29.92	28.91	27.92	27.07
SIGMARIGHT	31.17	31.21	30.60	29.87	28.98	27.78
SWEEPR	32.81	32.60	31.42	30.08	28.67	27.74
UPDATEQ	37.58	30.51	27.75	25.97	24.94	25.05
スカラ版						
DIAGONAL	23.63	24.29	25.52	25.93	26.03	25.06
SIGMALEFT	26.53	27.20	28.52	28.76	27.71	26.56
SWEEPL	26.53	27.16	28.26	28.56	27.78	26.60
SIGMARIGHT	26.24	26.31	26.11	26.01	25.94	25.20
SWEEPR	26.32	26.35	26.21	26.02	25.98	24.87
UPDATEQ	35.32	29.48	25.21	23.92	23.48	23.46



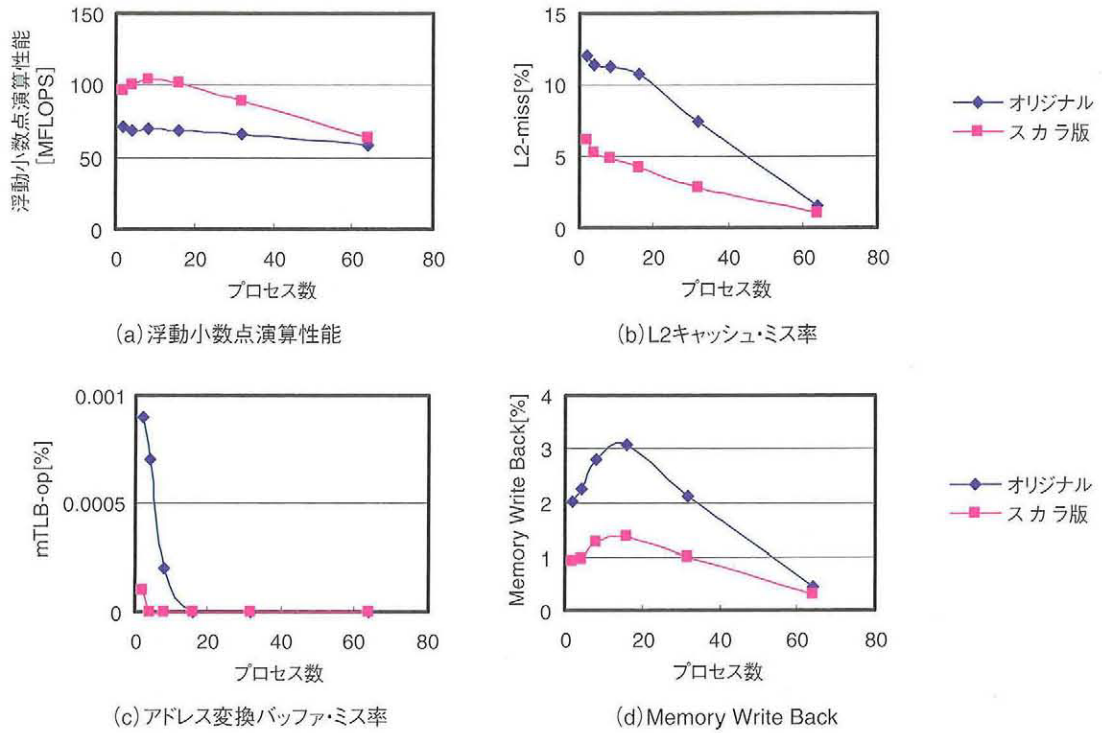


図 5.17 測定区間“DIAGONAL”におけるプロファイル情報抜粋

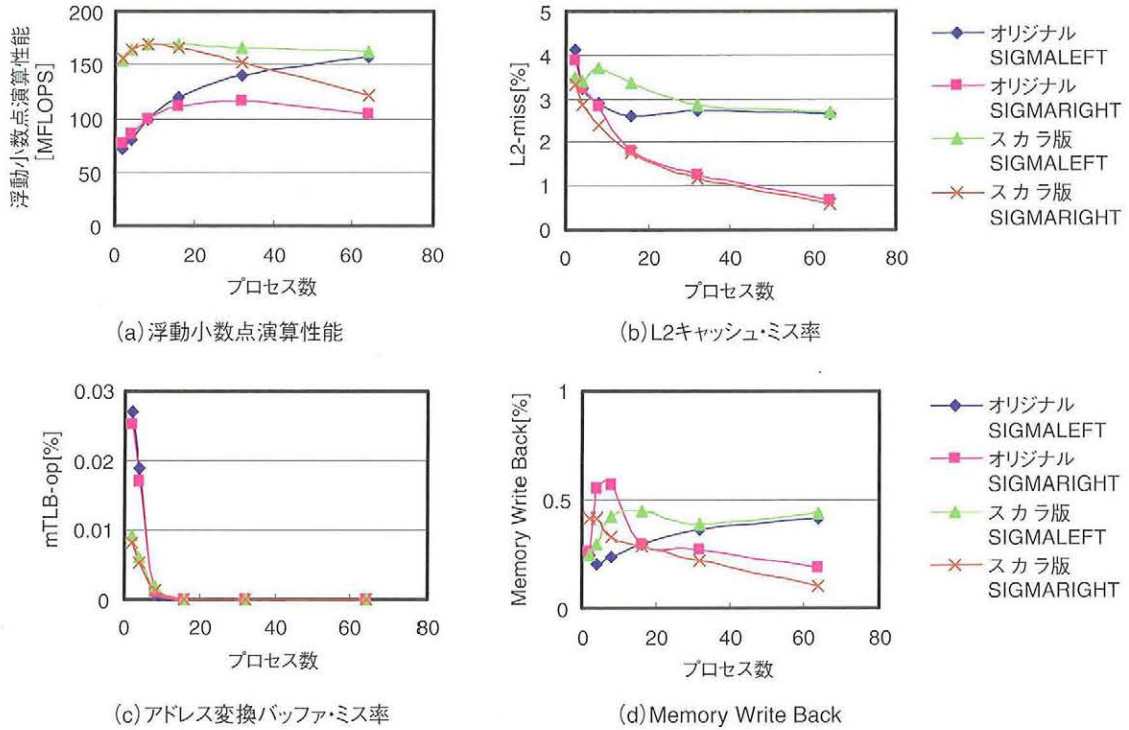


図 5.18 測定区間“SIGMALEFT”及び“SIGMARIGHT”におけるプロファイル情報抜粋



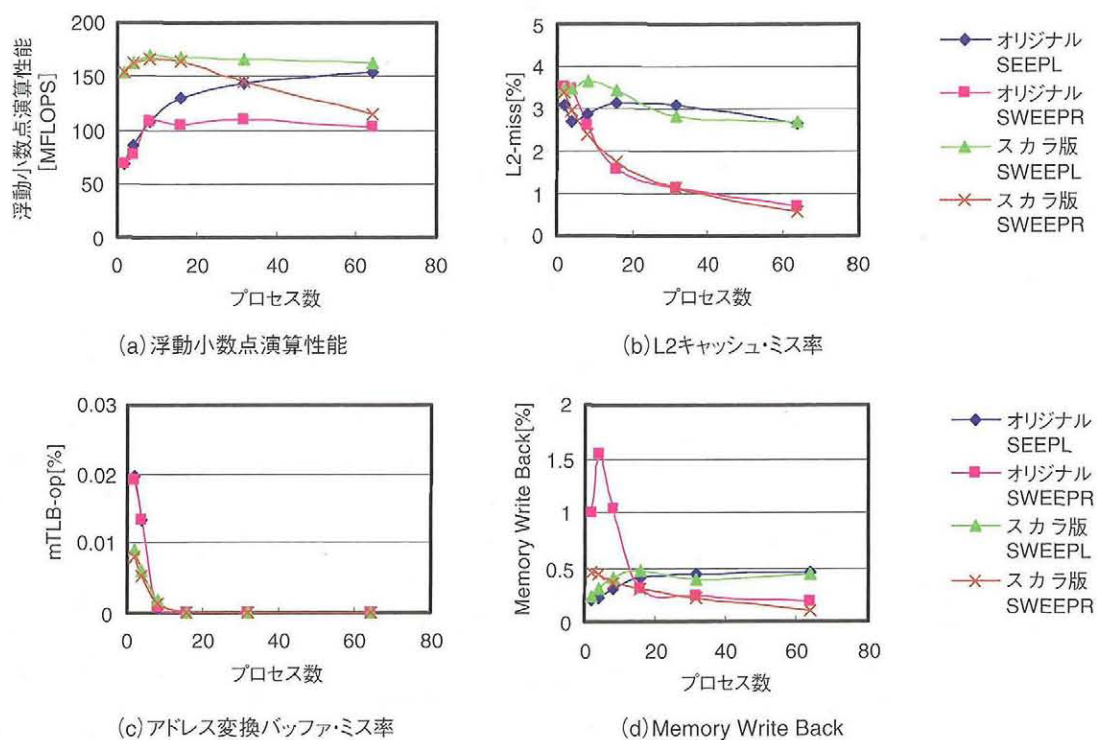


図 5.19 測定区間 “SWEPL” 及び “SWEPR” におけるプロファイラ情報抜粋

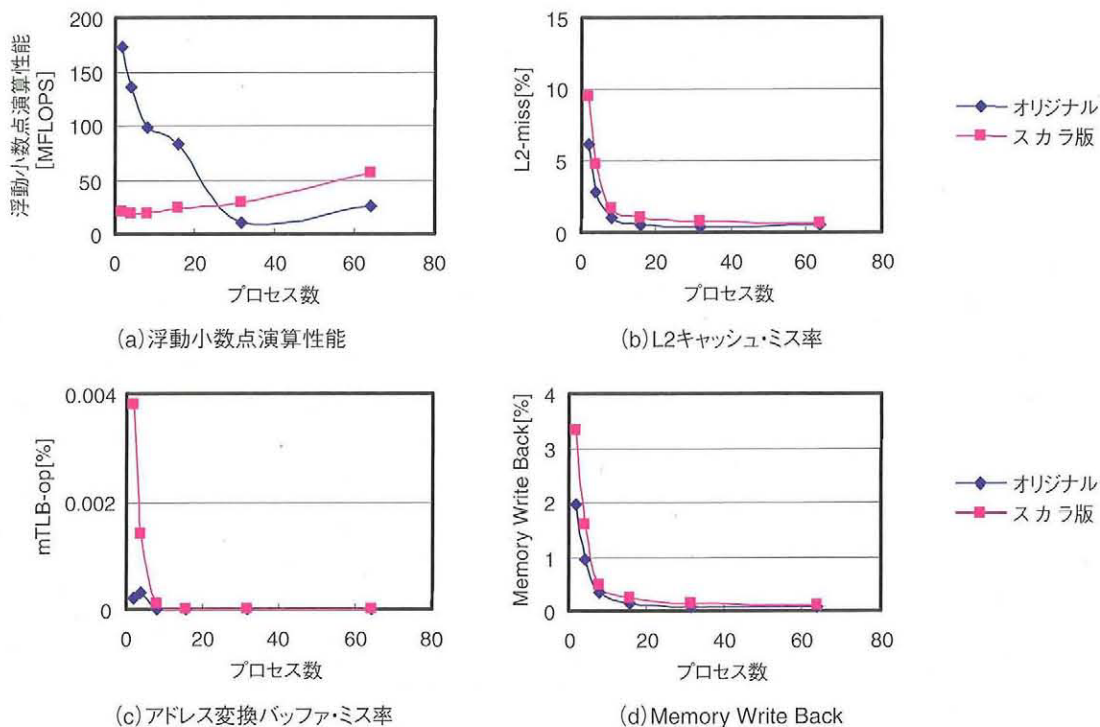


図 5.20 測定区間 “UPDATEQ” におけるプロファイラ情報抜粋

表 5.14 参照される配列の reordering による効果

測定区間：LU-SGS						
経過時間 [秒]						
version	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	2425.57	1069.97	516.90	278.49	166.60	109.47
reordering	1203.62	613.23	323.99	206.38	131.13	96.05
not	1371.25	723.19	388.82	211.36	134.90	92.96

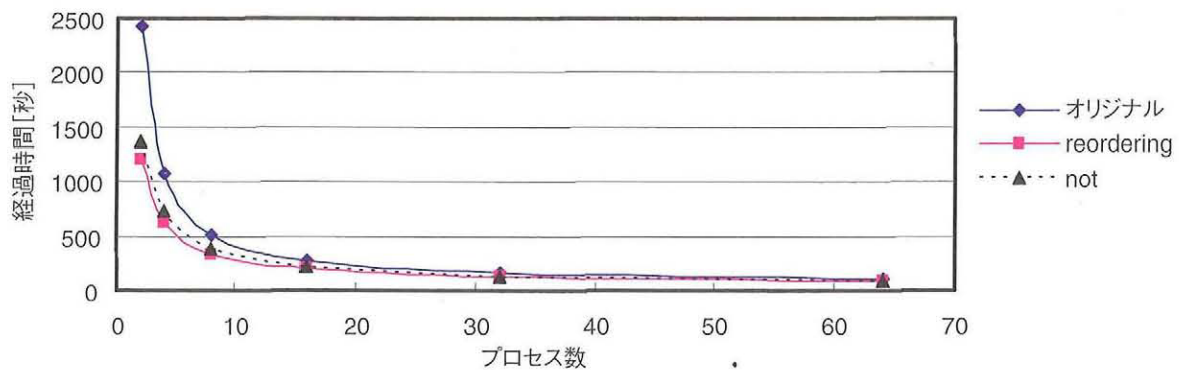


図 5.21 参照される配列の reordering による効果

表 5.15 ソートの効果

測定区間：LU-SGS						
経過時間 [秒]						
計算順序	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	2425.57	1069.97	516.90	278.49	166.60	109.47
sorted	1243.24	632.95	341.65	206.88	132.50	97.95
not sorted	1189.46	632.52	350.68	213.03	134.33	97.81
as is	1308.16	661.68	353.35	204.71	130.85	94.48

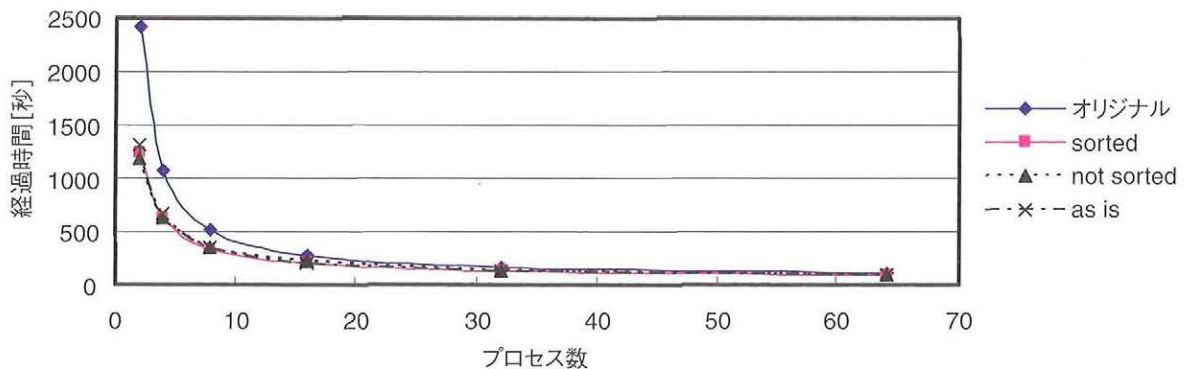
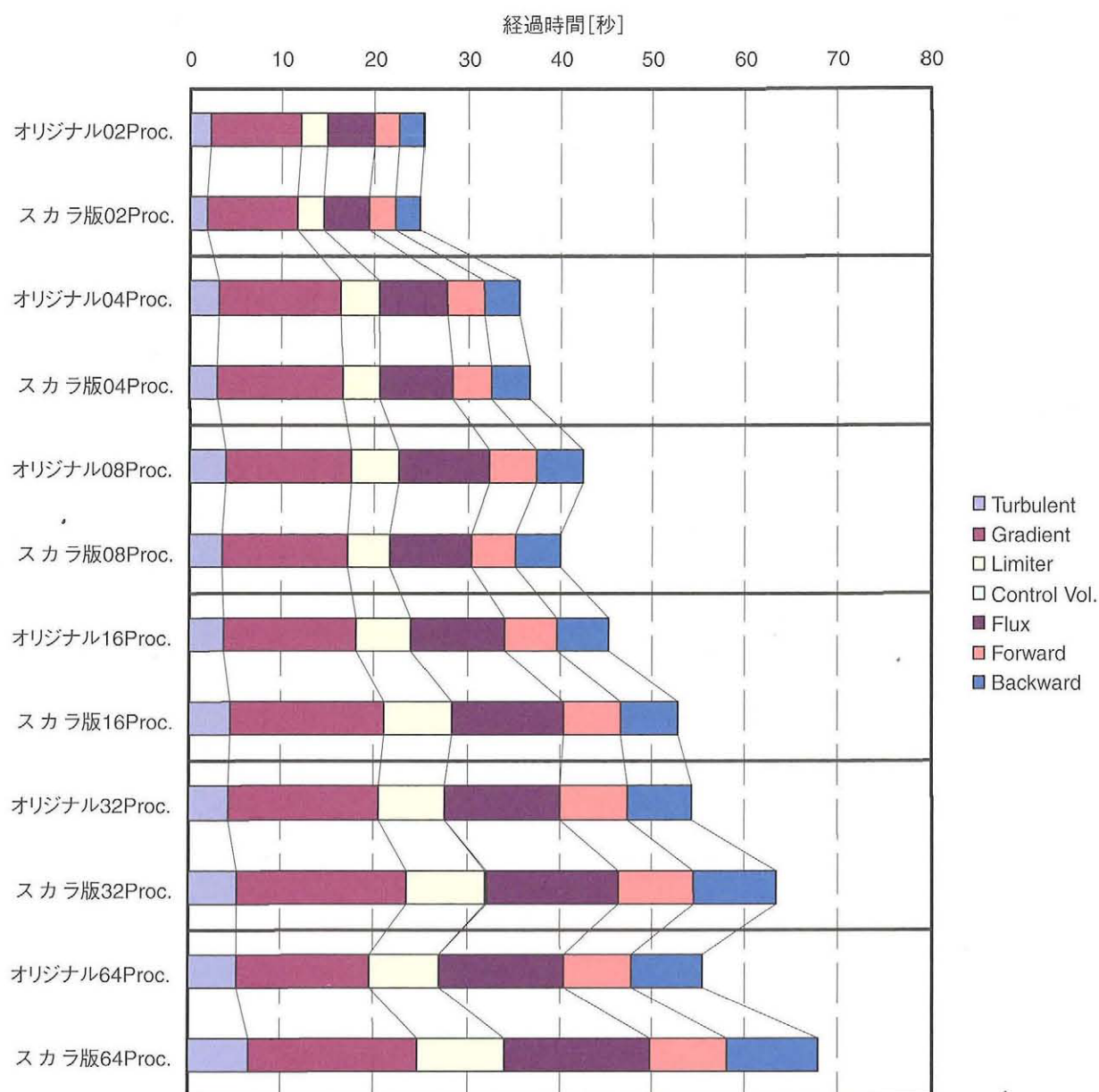


図 5.22 ソートの効果

表 5.16 測定区間 “COMMUNICATION” における測定結果

測定区間：COMMUNICATION						
経過時間 [秒]						
測定区間	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.
オリジナル	25.89	37.77	47.34	56.47	64.65	66.21
Turbulent	2.31	3.21	3.91	3.73	4.28	5.19
Gradient	9.85	13.29	13.67	14.42	16.22	14.52
Limiter	2.76	4.00	5.12	5.79	7.12	7.22
Control Vol.	0.00	0.00	0.00	0.00	0.00	0.02
Flux	5.00	7.18	9.48	9.94	12.36	13.50
Forward	2.77	3.95	5.16	5.80	7.38	7.23
Backward	2.64	3.90	5.05	5.57	6.90	7.85
スカラ版	25.82	40.49	44.72	63.01	70.13	75.97
Turbulent	1.90	3.01	3.58	4.52	5.19	6.58
Gradient	9.88	13.51	13.51	16.68	18.32	18.09
Limiter	2.71	4.02	4.68	7.11	8.43	9.35
Control Vol.	0.01	0.01	0.01	0.02	0.06	0.01
Flux	4.95	7.67	8.65	12.1	14.41	15.71
Forward	2.76	4.16	4.62	6.10	8.21	8.42
Backward	2.77	4.29	4.88	6.25	8.78	9.68



## 6. スレッド版計算性能測定結果及び評価

ここでは、スレッド版 JTAS による性能測定の結果を示し、測定結果に対する若干の評価及び JTAS オリジナルやスカラ版との性能比較を行う。まず、「6.1 計算結果の確認」で、変更が正しく行われたことをどのように確認したかについて説明する。続いて「6.2 測定方法」で性能測定を行った方法について説明する。最後に、「6.3 測定結果及び評価」において、性能測定を行った結果を示し、その簡単な評価を行う。

### 6.1 計算結果の確認

スレッド版を作成するにあたり JTAS オリジナルの計算結果と比較し変更が正しいものであることを確認した。この確認は、スレッド版において、時間積分500ステップの計算終了時における揚力“ $C_l$ ”、抗力“ $C_d$ ”及びピッチモーメント“ $T_{pm}$ ”の値が、オリジナルと十分な精度を持って一致しているか否かをもって行った。表6.1に確認を行った計算結果を示す。表中、“Proc. only”とは、コンパイルオプション“-Kparallel”を指定せずにコンパイルし MPI プロセスのみにより並列実行した場合の結果である。表6.1によれば、スレッド版の計算結果はオリジナルの計算結果と良く一致しており、これにより変更が正しく行われたことが確認された。

表 6.1 計算結果の確認

2 process 実行 時間積分500ステップ				
version		$C_l$	$C_d$	$T_{pm}$
オリジナル		$-2.510437870811829e-03$	$2.606253245212966e-03$	$5.925822252922501e-04$
ス レ ッ ド 版	Proc. only	$-2.510437870811889e-03$	$2.606253245212966e-03$	$5.925822252922761e-04$
	01Thread	$-2.510437870811476e-03$	$2.606253245212966e-03$	$5.925822252920433e-04$
	02Thread	$-2.510437870811476e-03$	$2.606253245212966e-03$	$5.925822252920433e-04$
	04Thread	$-2.510437870811476e-03$	$2.606253245212966e-03$	$5.925822252920433e-04$
	08Thread	$-2.510437870811476e-03$	$2.606253245212966e-03$	$5.925822252920433e-04$
	16Thread	$-2.510437870811476e-03$	$2.606253245212966e-03$	$5.925822252920433e-04$

### 6.2 測定方法

ここでは、性能測定をどのような方法で行ったかについて説明する。

経過時間 (Elapsed Time) を測定するためのタイマ挿入の方法、挿入したタイマによって測定した区間、プログラムの実行状況の解析を支援するツールであるプロファイラを実行するために指定したブション等は、全てスカラ版の測定を行ったときの方法に準ずる。従って、これらについては、「5.2 測定方法」を参照されたい。尚、スレッド版の測定においては、CPU 時間の計測は行わなかった。これは、CPU 時間を計測するサービス関数“ETIME”を呼び出すことによるオーバーヘッドが大きく、スレッド並列による並列化効率の測定に影響を与えたためである。従って、スレッド版の測定においては、論理変数“cpu\_on”に論理値“.false.”を与えてサービス関数“ETIME”の呼び出しを抑制した。

### 6.3 測定結果及び評価

ここでは、JTAS の性能測定の結果及びその評価について示す。測定は、MPI による並列化におけるプロセス数を2で固定とし、スレッド並列については MPI プロセスのみ (Process only)、1 プロセス当たりのスレッド

数1, 2, 4, 8及び16の6ケースについて行った。MPI プロセスのみの場合とスレッド数1の場合の実行はいずれも2CPUによる実行であるが、MPI プロセスのみの場合はコンパイルオプションに“-Kparallel”を指定せずにコンパイルし、スレッド数1の場合は指定したことが異なる。この違いにより、MPI プロセスのみの場合には実行可能モジュールにスレッド並列化のためのライブラリ呼び出しは組み込まれていないが、スレッド数1の場合にはその他のスレッド数の場合と同じ実行可能モジュールを使用しており、従って、ライブラリが組み込まれている。尚、オリジナルでは運用上のCPU時間制限により16スレッドによる実行は行えなかった。また、プロファイラによる実行状態の解析を行うには、プロファイラのために専用のスレッドを用意する必要があるが、CeNSSでは運用上のスレッド数上限が16であるため、16スレッド実行におけるプロファイラによる解析は行えなかった。加えて、測定は他のジョブも存在する通常の運用状態の元で行った。このため、特に経過時間の測定結果には変動要因が含まれていることに注意されたい。以下、先ず全体の測定結果について概観した後、主な処理ごとの測定結果について示し、簡単な評価を行う。

### 6.3.1 全体の測定結果及び評価

ここでは、JTAS全体に関する測定結果について概観する。表6.2, 図6.1及び6.2に主な測定区間における経過時間を測定した結果及びその区間におけるスレッド並列化により得られた加速率を示す。各欄の上段が経過時間であり、下段が加速率である。また、経過時間の単位は秒である。測定区間“SUBTOTAL”は、測定区間“SPVCOR”以下の測定区間について、その経過時間の合計を取ったものである。各測定区間の具体的な設定については、表5.2を参照されたい。各測定区間におけるより詳細な測定結果及び評価は、「6.3.2 測定区間“SPVCOR”における測定結果及び評価」から「6.3.5 測定区間“LU-SGS”における測定結果及び評価」に示す。測定区間“COMMUNICATION”におけるプロセス間通信のコストについてはスカラ版と同様であり、その詳しい検討はここでは行わない。

表6.2より、いずれのスレッド数による実行においても、最適化を行った全ての測定区間でオリジナルに比べてスレッド版における経過時間が短縮されており、最適化の効果が確認できた。また、8スレッド実行において、加速率が全体では6倍を下回っているものの、時間積分(測定区間“INTEGRATION”)においては6倍を上回っており、当初の目標を達成する十分な加速率が得られた。

「3.2 スレッド版変更内容」で示した通り、オリジナルが色分けによる再帰参照の回避を行っているのに対して、スレッド版ではLU-SGS法に関する部分を除いて色分けを削除しDOループを分割するコーディング方法を採用した。このコーディング方法は既に示したように、構造格子法において良く用いられている方法であり、非構造格子法においても同様のコーディング方法を用いた場合に良い性能が得られることが確認されたことは、興味深い結果である。一方で、LU-SGS法では「6.3.5 測定区間“LU-SGS”における測定結果及び評価」に示すように、オリジナルにおける色分けをそのままスレッド並列化することにより、色分けを削除しDOループを分割するコーディング方法より良い性能が得られた。加えて、MPIによるプロセス並列化のみでスレッド並列を行わない場合には、基本的に再帰参照を含むリスト2.1に示すコーディング方法において良い性能が得られたが、例外的に勾配 $\nabla \mathbf{q}$ の計算(測定区間“SPDRF1”)においてはDOループの分割によって良い性能が得られた。即ち、再帰参照を含むコーディング方法よりDOループを分割するコーディング方法において良い性能が得られる場合が存在し、同時にDOループを分割するコーディング方法より色分けを用いたコーディング方法においてより良い性能が得られる場合が存在することがわかった。このことは、インデックス参照を行うと共に再帰参照を含むJTASに類似のDOループに適用できる一般的なスレッド並列化の手法が存在しないことを意味する。また、スカラ版において測定区間“SPDRF1”はスレッド並列化可能であり、結局、非構造格子法やそれと類似のインデックス参照を多用するアプリケーションにおいては、MPIによるプロセス並列のみによる並列化を

表 6.2 JTAS 全体の経過時間測定結果

経過時間 2 process 実行 時間積分500ステップ		上段：経過時間 [秒] 下段：加 速 率 [-]				
オリジナル						
測定区間	Process only	Hybrid				
		01Thread	02Thread	04Thread	08Thread	16Thread
TOTAL	8843.54	8803.71 1.00	4978.60 1.77	2770.26 3.18	1814.81 4.85	
PREPROCESS	68.30	70.13	69.99	66.36	66.18	
INTEGRATION	8775.23	8733.57 1.00	4908.60 1.78	2703.89 3.23	1748.62 4.99	
SPVCOR	1410.40	1141.31 1.00	773.14 1.48	374.34 3.04	205.75 5.55	
SPDRF1	3069.13	2679.61 1.00	1432.21 1.87	689.44 3.89	362.70 7.39	
LIMITER	1598.96	1599.02 1.00	861.88 1.86	410.54 3.89	211.96 7.54	
LU-SGS	2009.07	2297.07 1.00	1414.62 1.62	964.67 2.38	773.07 2.97	
COMMUN.	54.91	21.65	28.11	23.01	21.80	
SUBTOTAL	8142.47	7738.66 1.00	4509.96 1.72	2462.00 3.14	1575.28 4.91	
スレッド版						
測定区間	Process only	Hybrid				
		01Thread	02Thread	04Thread	08Thread	16Thread
TOTAL	5910.70	6012.07 1.00	3229.03 1.86	1810.94 3.32	1075.91 5.59	704.99 8.53
PREPROCESS	112.33	122.66	122.76	122.80	121.15	120.80
INTEGRATION	5798.36	5889.41 1.00	3106.27 1.90	1688.14 3.49	954.76 6.17	584.19 10.08
SPVCOR	941.18	1036.89 1.00	531.88 1.95	273.57 3.79	139.73 7.42	71.83 14.44
SPDRF1	1725.38	1660.64 1.00	862.60 1.93	454.79 3.65	237.36 7.00	130.09 12.77
LIMITER	1009.35	1087.35 1.00	581.07 1.87	310.65 3.50	161.96 6.71	83.83 12.97
LU-SGS	1356.23	1374.72 1.00	699.87 1.96	389.80 3.53	238.55 5.76	165.61 8.30
COMMUN.	52.27	29.99	58.68	39.00	36.66	29.63
SUBTOTAL	5084.41	5189.59 1.00	2734.10 1.90	1304.65 3.98	814.26 6.37	480.99 10.79

行うか、スレッド並列を併用したハイブリッド並列化を行うかについても、事前に予測することは非常に困難であるということが言える。表6.3にJTASの主な計算と、その計算を最も効率よく行えたコーディング方法との関係をまとめておく。表中の網掛け部分は、その他の部分とは異なるコーディング方法により良い性能が得られた部分である。

表6.4及び図6.3にプロファイラによって得られた実行状況の解析結果について、その抜粋を示す。表中、各解析項目の“Rank0”及び“Rank1”は、MPIプロセスのランク0及びランク1における解析結果である。“Min.”及び“Max.”は、それぞれのプロセスに含まれるスレッドにおける測定値のうち最小値及び最大値であり、( )内はその値が測定されたスレッドの番号である。“Average”は、ランク0とランク1の測定値を単純平均した値であり、“Total”はプロファイラが“Process Total”として出力した結果である。L2キャッシュ・ミス率、アドレス変換バッファ・ミス率共にオリジナルに比べて改善しているもの十分とは言えず、この結果、浮動小数点演算性能も150MFLOPSを若干上回る程度に留まった。

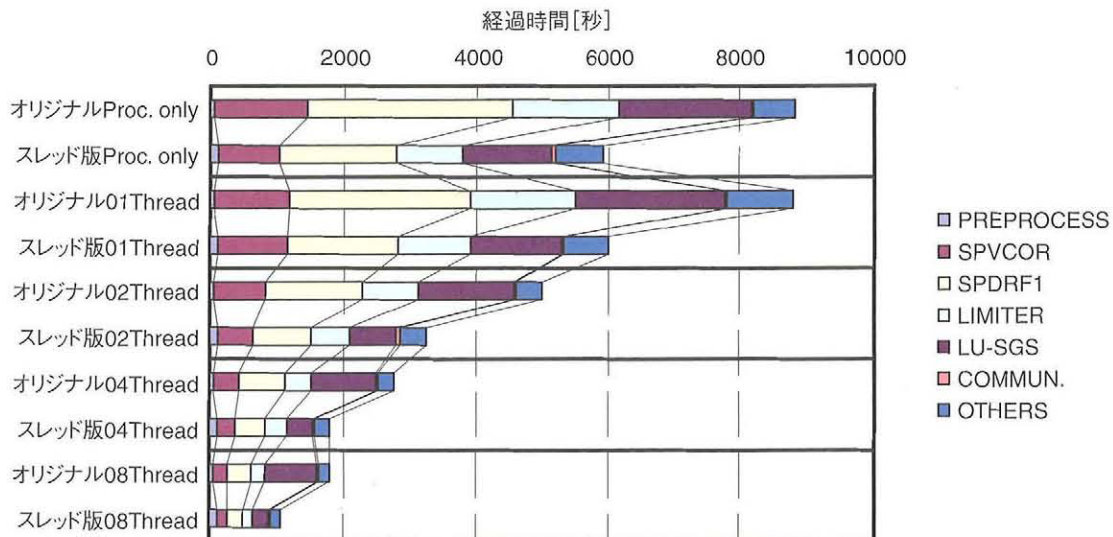


図 6.1 JTAS 全体の経過時間測定結果

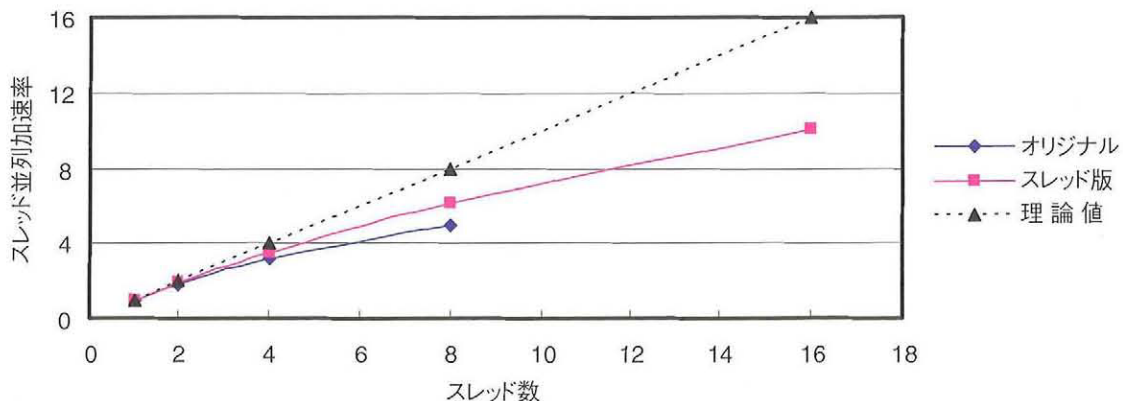
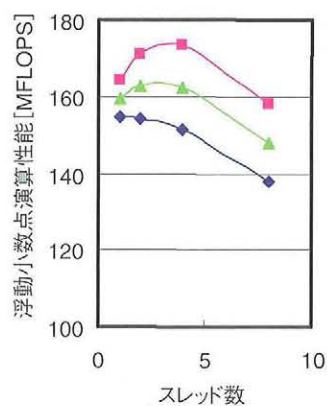


図 6.2 測定区間“INTEGRATION”におけるスレッド並列実行による加速率

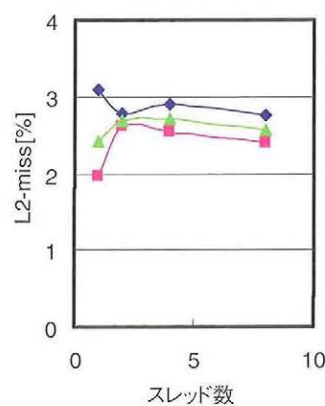


表 6.3 JTAS の主な計算と最適なコーディング方法の関係

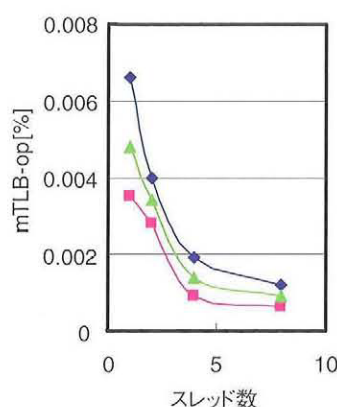
計算内容	測定区間	関係するサブルーチン	最も性能が良かったコーディング方法	スレッド並列化可能で性能が良かったコーディング方法
数値流束の計算	SPVCOR	SPVCOR	色分けの削除 再帰参照を含む	色分けの削除 DO ループ分割
勾配の計算	SPDRF1	SPDRF1_pri SPDRF1_pyr SPDRF1_tet	色分けの削除 DO ループ分割 再帰参照を含まず	色分けの削除 DO ループ分割
制限関数の計算	LIMITER	LIMITER1 LIMITER2	色分けの削除 再帰参照を含む	色分けの削除 DO ループ分割
LU-SGS 法	LUSGS	DIAGONAL SIGMALEFT SIGMALIGHT	色分けの削除 再帰参照を含む	色分け



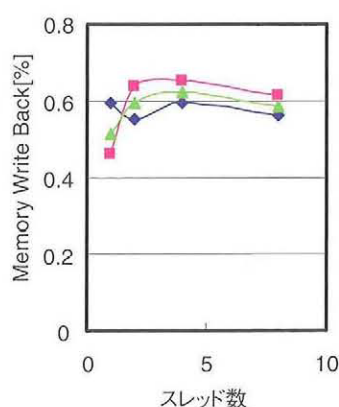
(a) 浮動小数点演算性能



(b) L2キャッシュ・ミス率



(c) アドレス変換バッファ・ミス率



(d) Memory Write Back

図 6.3 プロファイラ情報抜粋

表 6.4 プロファイラ情報抜粋 (続く)

測定区間：TOTAL 2 process 実行							( ) 内はスレッド番号
浮動小数点演算性能 FLOPS [MFLOPS]							
Rank	オリジナル	スレッド版					
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread	
Rank0	103.7	155.9	154.8	154.0	151.4	137.7	
Min.				143.7 (00)	131.7 (00)	99.8 (00)	
Max.				167.1 (01)	174.1 (03)	169.0 (07)	
Rank1	109.6	165.5	164.4	171.1	173.1	158.1	
Min.				153.0 (00)	141.6 (00)	108.5 (00)	
Max.				197.8 (01)	216.2 (03)	203.8 (07)	
Average	106.7	160.7	159.6	162.6	162.3	147.9	
命令処理性能 MIPS [MIPS]							
Rank	オリジナル	スレッド版					
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread	
Rank0	282.2	307.2	317.8	351.5	327.9	310.3	
Min.				309.3 (01)	265.0 (01)	268.5 (02)	
Max.				384.9 (00)	358.0 (00)	421.7 (00)	
Rank1	305.6	440.7	456.0	353.8	371.1	350.8	
Min.				347.6 (00)	282.8 (01)	285.4 (01)	
Max.				363.1 (01)	368.5 (03)	424.7 (00)	
Average	293.9	747.9	386.9	352.7	349.5	330.55	
L2 キャッシュ・ミス率 L2-miss [%]							
Rank	オリジナル	スレッド版					
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread	
Rank0	3.54	3.21	3.08	2.78	2.91	2.75	
Min.				2.60 (00)	2.48 (00)	1.86 (00)	
Max.				3.06 (01)	3.47 (01)	3.41 (02)	
Rank1	3.27	2.05	1.96	2.61	2.53	2.39	
Min.				2.22 (01)	2.05 (03)	1.84 (07)	
Max.				2.89 (00)	3.13 (01)	3.23 (01)	
Total	3.40	2.53	2.42	2.69	2.71	2.57	
アドレス変換バッファ・ミス率 mTLB-op [%]							
Rank	オリジナル	スレッド版					
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread	
Rank0	0.0098	0.0060	0.0066	0.0040	0.0019	0.0012	
Min.				0.0021 (01)	0.0004 (02)	0.0002 (06)	
Max.				0.0053 (00)	0.0039 (00)	0.0024 (00)	
Rank1	0.0053	0.0031	0.0035	0.0028	0.0009	0.0006	
Min.				0.0014 (01)	0.0002 (02)	0.0001 (06)	
Max.				0.0038 (00)	0.0021 (00)	0.0017 (00)	
Total	0.0075	0.0043	0.0048	0.0034	0.0014	0.0009	

表 6.4 プロファイラ情報抜粋 (続き)

測定区間：TOTAL 2 process 実行		() 内はスレッド番号				
Memory Write Back [%]						
Rank	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
Rank0	1.1347	0.6251	0.5923	0.5518	0.5960	0.5581
Min.				0.4935 (00)	0.4770 (00)	0.3369 (00)
Max.				0.6438 (01)	0.6772 (02)	0.6486 (07)
Rank1	1.0696	0.4836	0.4576	0.6349	0.6509	0.6108
Min.				0.5832 (00)	0.5122 (00)	0.3884 (00)
Max.				0.7081 (01)	0.7958 (03)	0.7443 (05)
Total	1.1009	0.5421	0.5133	0.5923	0.6239	0.5848
浮動小数点演算命令の割合 [%]						
Rank	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
Rank0	36.73	50.75	48.70	43.81	46.15	44.39
Min.				37.35 (00)	33.97 (00)	23.67 (00)
Max.				54.02 (01)	53.83 (01)	52.67 (02)
Rank1	35.87	37.56	36.06	48.35	46.64	45.06
Min.				44.01 (00)	35.20 (00)	25.55 (00)
Max.				54.49 (01)	53.84 (03)	52.64 (05)
Total	36.29	43.01	41.29	46.02	46.40	44.73
Load/Store 命令の割合 [%]						
Rank	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
Rank0	34.13	29.25	29.93	29.93	30.28	30.35
Min.				29.74 (01)	29.88 (01)	29.87 (01)
Max.				30.05 (00)	30.85 (00)	31.38 (00)
Rank1	34.42	29.27	29.73	30.47	30.63	30.61
Min.				29.85 (01)	30.05 (03)	29.80 (06)
Max.				30.91 (00)	31.30 (00)	31.68 (00)
Total	34.28	29.26	29.81	30.19	30.46	30.45

表 6.5 最適化による性能向上率

測定区間：TOTAL 算定根拠：経過時間		単位：[ratio]			
Process only	Hybrid				
	01Thread	02Thread	04Thread	08Thread	
1.50	1.46	1.54	1.53	1.69	

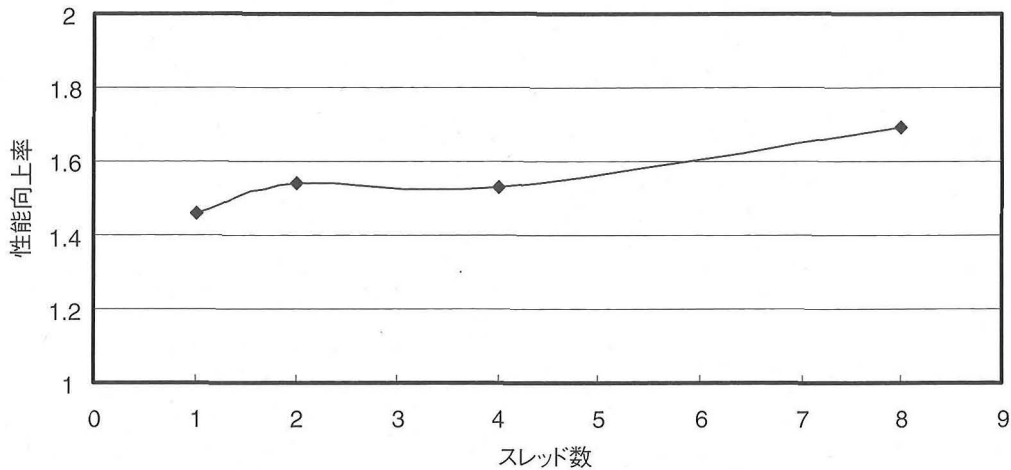


図 6.4 最適化による性能向上率

表6.5及び図6.4に経過時間を元に算出したJTAS オリジナルに対してスレッド版でどの程度性能が向上したかを表す性能向上率を示す。性能向上率は、オリジナルの実行に要した経過時間をスレッド版の実行に要した経過時間で除したものとして計算した。いずれのスレッド数による実行でも性能が向上することが確認出来た。一方で、スカラ版における最適化の効果（表5.7参照）には及ばないことも確認された。

### 6.3.2 測定区間“SPVCOR”における測定結果及び評価

測定区間“SPVCOR”では、HLLW法による数値流束ベクトル  $\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n}$  の計算を行っているサブルーチン“SPVCOR”について、経過時間の測定及び実行状況の解析を行なった。表6.5及び図6.5から6.7にこの区間における測定及び解析結果を示す。経過時間は、サブルーチン“SPVCOR”の実行に要した時間であり、バリア同期を取ってルートプロセス（ランク0）における測定結果のみを示している。“Loop 1”及び“Loop 2”は、それぞれリスト6.1に示すサブルーチン“SPVCOR”の対応する部分についての経過時間及び解析結果である。スレッド並列による実行では、スレッドが生成される部分がサブルーチン化されてしまうため、サブルーチン“SPVCOR”全体の解析結果を得ることが出来なかった。そこで、ループごとの解析結果を示した。

オリジナルに比べて処理に要する経過時間が短縮されると同時に、表6.2及び図6.6に示したように8スレッド（2プロセス、16CPU）実行においてオリジナルで5.55倍だったスレッド並列による加速率がスレッド版において7.42倍に向上しており、その最適化の効果が確認できた。一方で、スカラ版と同じCPU数による実行における経過時間を比較すると、スカラ版が16プロセスで92.92秒（表5.9参照）であるのに対してスレッド版では8スレッド（2プロセス）で139.73秒を要しており、性能面ではプロセス並列のみによる実行が優位であるという結果が得られた。また、スレッド数の増加によりL2キャッシュ・ミス率が低減することが期待されたが、その傾向を確認することは出来なかった。

### 6.3.3 測定区間“SPDRF1”における測定結果及び評価

測定区間“SPDRF1”では、各節点及び辺における勾配  $\nabla \mathbf{q}_i$  の計算について、経過時間の測定及び実行状況の解析を行なった。表6.6及び図6.8から6.10にこの区間における測定及び解析結果を示す。測定区間“\_tet”、“\_prism”及び“\_pyr”は、それぞれ、サブルーチン“SPDRF1\_tet”、“SPDRF1\_pri”及び“SPDRF1\_pyr”における測定及び解析結果である。経過時間は、バリア同期を取ってルートプロセス（ランク0）における測定結果のみを示している。スレッド版における“SUM”は、リスト3.1の「変更後」に示されたDO200及びDO300の処理に要した経過時間であり、“node”は節点における勾配の計算に対応するDO200、“edge”は辺における勾

表 6.5 測定区間 “SPVCOR” における測定結果

測定区間：SPVCOR 2 process 実行						
経過時間 [秒] 時間積分500ステップ						
version	Proc. only	01Thread	02Thread	04Thread	08Thread	16Thread
オリジナル	1410.40	1141.31	773.14	374.34	205.75	
スレッド版	941.18	1036.89	531.88	273.57	139.73	71.83
浮動小数点演算性能 FLOPS [MFLOPS]						
Loop	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
Loop 1	178.5	283.7	304.5	302.4	303.2	290.9
Loop 2			108.1	102.8	92.0	90.8
命令処理性能 MIPS [MIPS]						
Loop	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
Loop 1	311.6	488.0	537.3	534.1	537.2	520.9
Loop 2			241.7	225.1	236.3	224.07
L2 キャッシュ・ミス率 L2-miss [%]						
Loop	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
Loop 1	2.68	1.65	1.29	1.36	1.37	1.33
Loop 2			4.47	4.37	4.03	3.54
アドレス変換バッファ・ミス率 mTLB-op [%]						
Loop	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
Loop 1	0.0001	0.0000	0.0000	0.0000	0.0000	0.0001
Loop 2			0.0002	0.0002	0.0003	0.0005
Memory Write Back [%]						
Loop	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
Loop 1	0.5875	0.2843	0.2537	0.3221	0.3396	0.3102
Loop 2			0.3137	0.3102	0.2900	0.2632
浮動小数点演算命令の割合 [%]						
Loop	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
Loop 1	45.57	58.15	56.68	56.63	56.45	55.84
Loop 2			44.73	43.73	42.49	40.51
Load/Store 命令の割合 [%]						
Loop	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
Loop 1	29.70	29.48	28.56	28.62	28.64	28.67
Loop 2			28.33	28.35	28.51	28.52

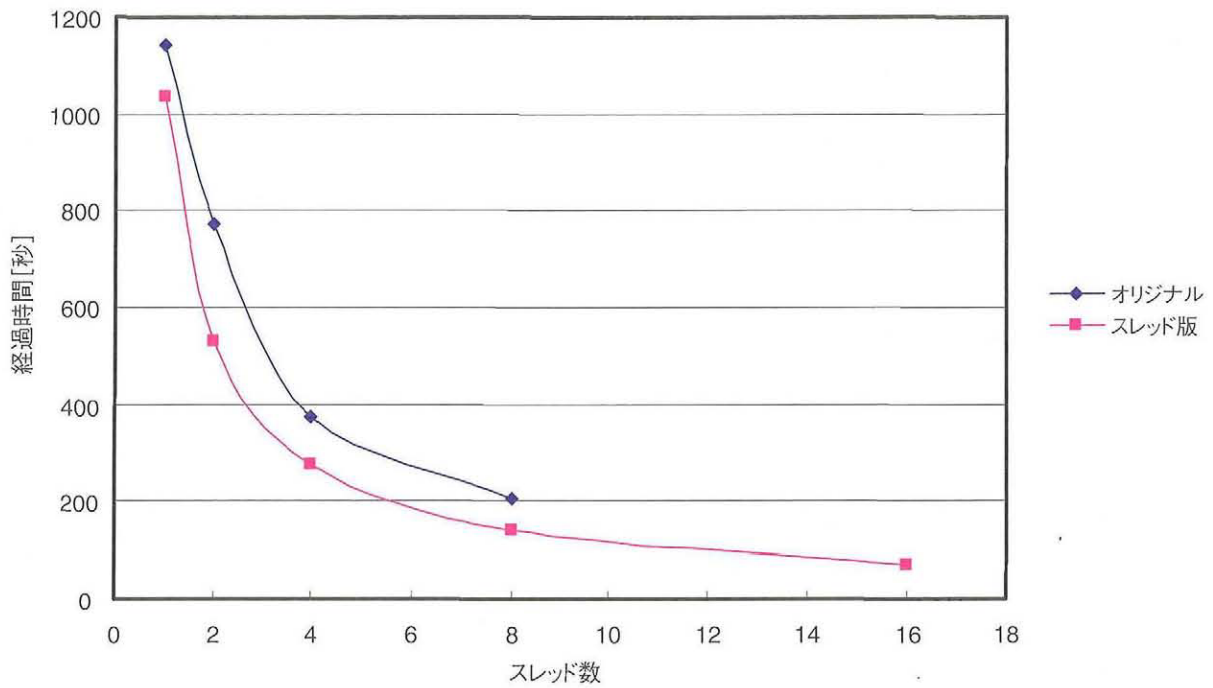


図 6.5 測定区間“SPVCOR”における測定結果

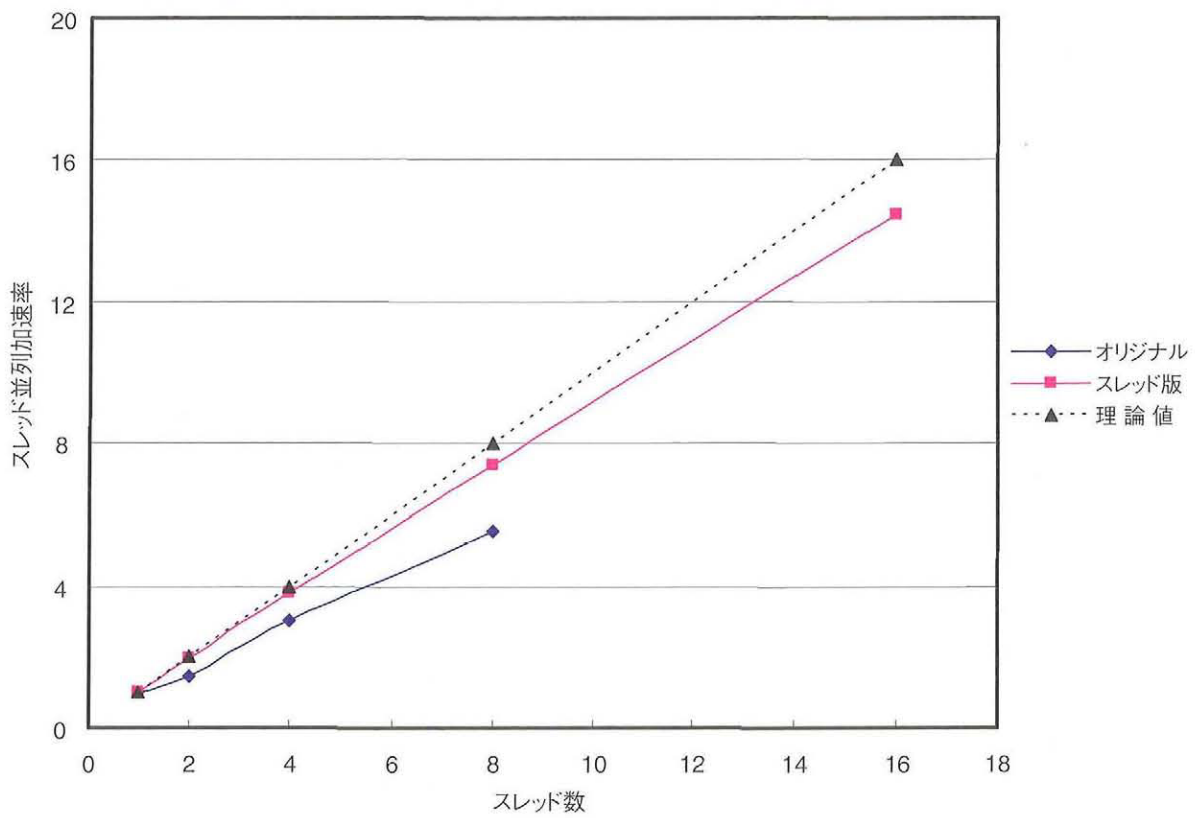
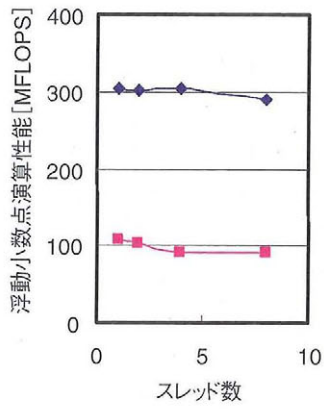
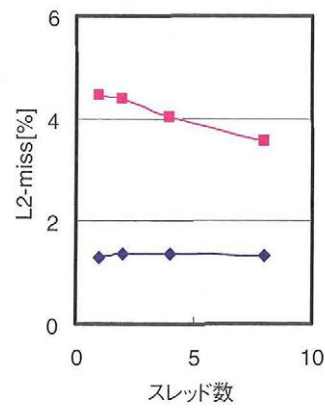


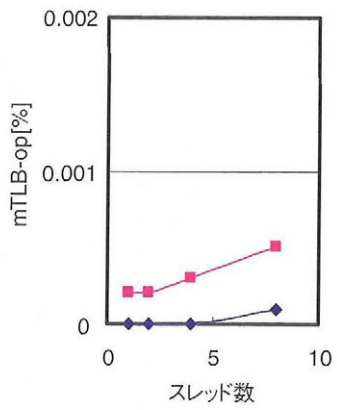
図 6.6 測定区間“SPVCOR”におけるスレッド並列実行による加速率



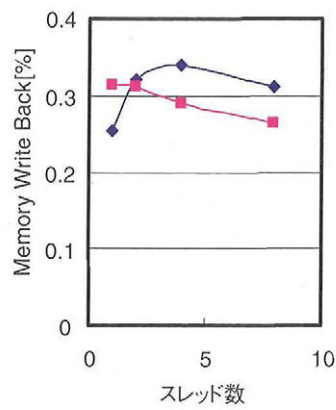
(a) 浮動小数点演算性能



(b) L2キャッシュ・ミス率





(c) アドレス変換バッファ・ミス率



(d) Memory Write Back

図 6.7 測定区間 “SPVCOR” におけるプロファイラ情報抜粋

リスト 6.1 サブルーチン “SPVCOR” (抜粋)

<pre> DO 100 IE DGE=1, N_ED_F   IF (IDS_ED (ICL_ED, IE GDE). GE. 1) THEN     IND1=IDS_ED (IN1_ED, IEDGE)     IND2=IDS_ED (IN2_ED, IEDGE)     :     EDG_WK( 1, IEDGE)=F1     :     EDG_WK( 8, IEDGE)=G7   END IF 100 CONTINUE </pre>		Loop 1
<pre> DO 110 IP=1, N_P_F   NEDG=IEDND_TBL (0, IP)   DO 111 IE=1, NEDG     IEDGE=IEDND_TBL (IE, IP)     XSIGN=SIGN (1. 0DO, IEDGE)     IEDGE=ABS (IEDGE)     F1 =EDG_WK ( 1, IEDGE)     :     G7 =EDG_WK ( 8, IEDGE)     POLE (IR1_PL, IP)=POLE (IR1_PL, IP)+XSIGN*EDG_WK ( 1, IEDGE)     :     POLE (IP1_PL, IP)=POLE (IP1_PL, IP)+XSIGN*EDG_WK ( 5, IEDGE)   : 111 CONTINUE 110 CONTINUE </pre>		Loop 2

配の計算に対応する DO300におけるプロファイラの測定結果である。DO200及び DO300は、サブルーチン “SPDRF1\_tet”, “SPDRF1\_pri” 及び “SPDRF1\_pyr” を呼び出しているサブルーチン “DR” で、一括して実行される。従って、これら3つのサブルーチンにおける経過時間及び解析結果には、これらの DO ループでの処理に関する部分は含まれない。このことから、オリジナルにおける測定区間 “\_tet”, “\_prism” 及び “\_pyr” とスレッド版における当該測定区間を直接比較することは出来ないことに注意されたい。

この測定区間は、スカラ版と同じでありスレッド版の MPI のみ (Proc. only) による実行の測定結果が表5.10と異なるのは、測定誤差によるものである。

この測定区間においては、8スレッド実行の場合にオリジナルで7.39倍あったスレッド並列化による加速率がスレッド版では7.00倍に低下している (表6.2参照)。ここで、加速率  $S$ 、並列化率  $\alpha$  及びスレッド数  $n$  としてアムダールの法則、

$$S = \frac{1}{\frac{\alpha}{n} + (1-\alpha)} \quad (6.1)$$

を適用し、スレッド並列化によるオーバーヘッド  $O$  を

$$O = (1-\alpha)T \quad (6.2)$$

で評価するものとする。但し、 $T$  は当該測定区間の計算に要した経過時間であり、従って、オーバーヘッドは、当該区間の計算に要する時間の内、並列化されていない部分の処理に要した時間である。この時、オリジナルに



において98.8%であった並列化率 $\alpha$ がスレッド版では98.0%に低下し、スレッド並列化によるオーバーヘッドは、オリジナル版の4.2秒に対してスレッド版では4.9秒に増加している。しかも、オーバーヘッドの7割以上に当たる3.5秒が測定区間“SUM”において費やされている。また、スレッド数が2、4及び8と増加するに従い、測定区間“SUM”におけるオーバーヘッドが、それぞれ1.3秒、2.8秒及び3.5秒と順次増加している。表6.6及び図6.10に示したように、測定区間“SUM”においては、スレッド数の増加とともにアドレスバッファ変換ミス率も増加していることから、これがスレッド加速率の低下の一因であると考えられる。また、スレッド数の増加により各スレッドのメモリアクセスの競合が発生している可能性もあるが、これに関してはプロファイラの情

表 6.6 測定区間“SFDRF1”における測定結果(続く)

測定区間：SPDRF1 2 process 実行						
経過時間 [秒] 時間積分500ステップ						
測定区間	Proc. only	01Thread	02Thread	04Thread	08Thread	16Thread
オリジナル	3069.13	2679.61	1432.21	689.44	362.70	
_tet	2402.33	2116.35	1152.98	546.24	286.84	
_prism	661.51	558.44	275.42	139.98	72.87	
_pyr	2.63	2.33	1.17	0.71	0.48	
スレッド版	1725.38	1660.64	862.60	454.79	237.36	130.09
_tet	431.39	512.66	261.88	139.02	70.20	37.73
_prism	83.03	81.39	41.54	21.10	10.43	5.14
_pyr	0.46	0.47	0.31	0.19	0.12	0.08
SUM	1205.74	1061.13	531.88	273.57	154.03	71.83
浮動小数点演算性能 FLOPS [MFLOPS]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
_tet	80.5	279.5	287.1	282.8	277.1	255.0
_prism	135.6	1009.0	1034.5	945.8	886.2	763.3
Sum node		93.8	87.6	88.4	94.3	88.59
edge		55.5	54.2	53.5	53.2	49.73
命令処理性能 MIPS [MIPS]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
_tet	176.7	426.4	441.2	436.5	431.4	405.4
_prism	278.3	1589.1	1637.9	1528.8	1469.9	1311.7
Sum node		230.6	224.5	225.0	233.7	220.1
edge		175.8	173.0	171.3	172.4	166.0
L2 キャッシュ・ミス率 L2-miss [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
_tet	5.08	1.92	1.89	1.99	2.19	2.35
_prism	2.58	0.29	0.29	0.34	0.44	0.65
Sum node		5.94	6.85	6.63	6.12	5.59
edge		5.48	5.65	5.64	5.56	5.43

表 6.6 測定区間“SFDRF1”における測定結果(続き)

測定区間：SPDRF1 2 process 実行						
アドレス変換バッファ・ミス率 mTLB-op [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
_tet	0.0115	0.0000	0.0000	0.0000	0.0000	0.0001
_prism	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000
Sum node		0.0026	0.0026	0.0025	0.0026	0.0028
edge		0.0009	0.0007	0.0008	0.0009	0.0012
Memory Write Back [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
_tet	2.8802	0.5923	0.5904	0.6936	0.9278	1.1273
_prism	1.7428	0.1701	0.1710	0.1552	0.1568	0.1824
Sum node		0.4735	0.4126	0.4109	0.3965	0.3873
edge		0.9572	0.9412	0.9269	0.8996	0.8462
浮動小数点演算命令の割合 [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
_tet	45.57	65.54	65.07	64.78	64.23	62.92
_prism	48.71	63.49	63.16	61.87	60.29	58.19
Sum node		40.70	39.04	39.30	40.35	40.24
edge		31.60	31.32	31.21	30.87	29.97
Load/Store 命令の割合 [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
_tet	46.08	27.65	27.60	28.07	29.04	29.98
_prism	45.45	33.20	33.41	33.14	32.43	31.21
Sum node		48.23	48.88	48.38	47.04	45.44
edge		49.99	48.38	48.27	47.92	47.18

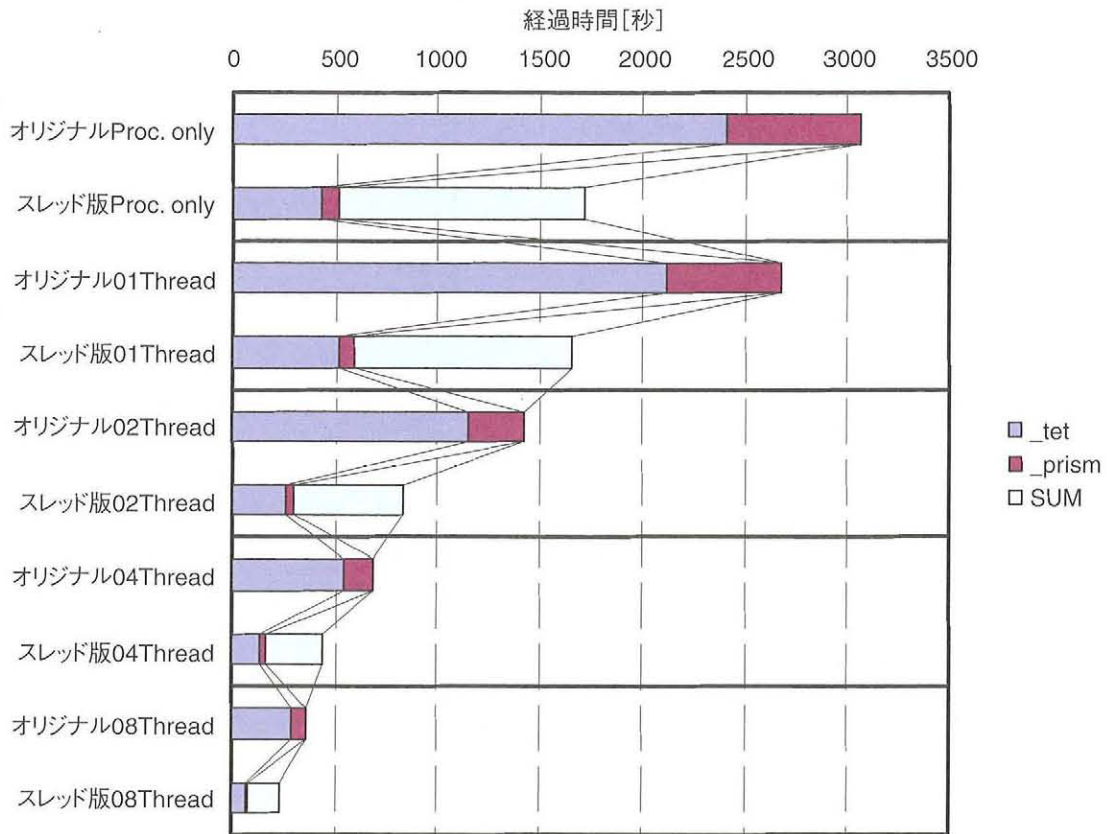


図 6.8 測定区間 "SFDRF1" における測定結果

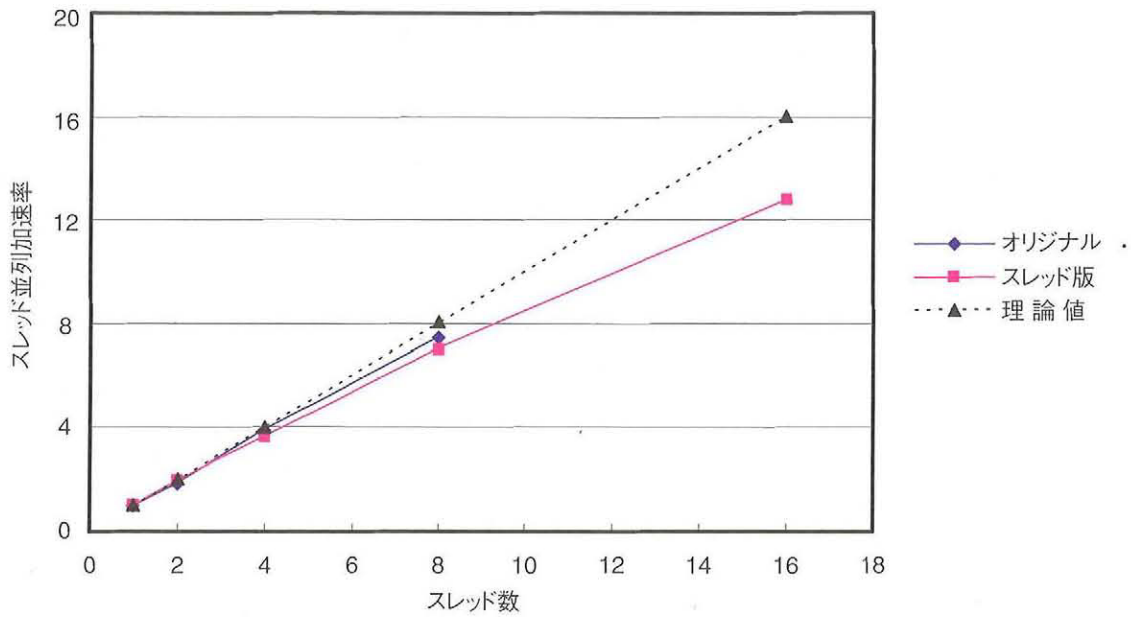


図 6.9 測定区間 "SFDRF1" におけるスレッド並列実行による加速率

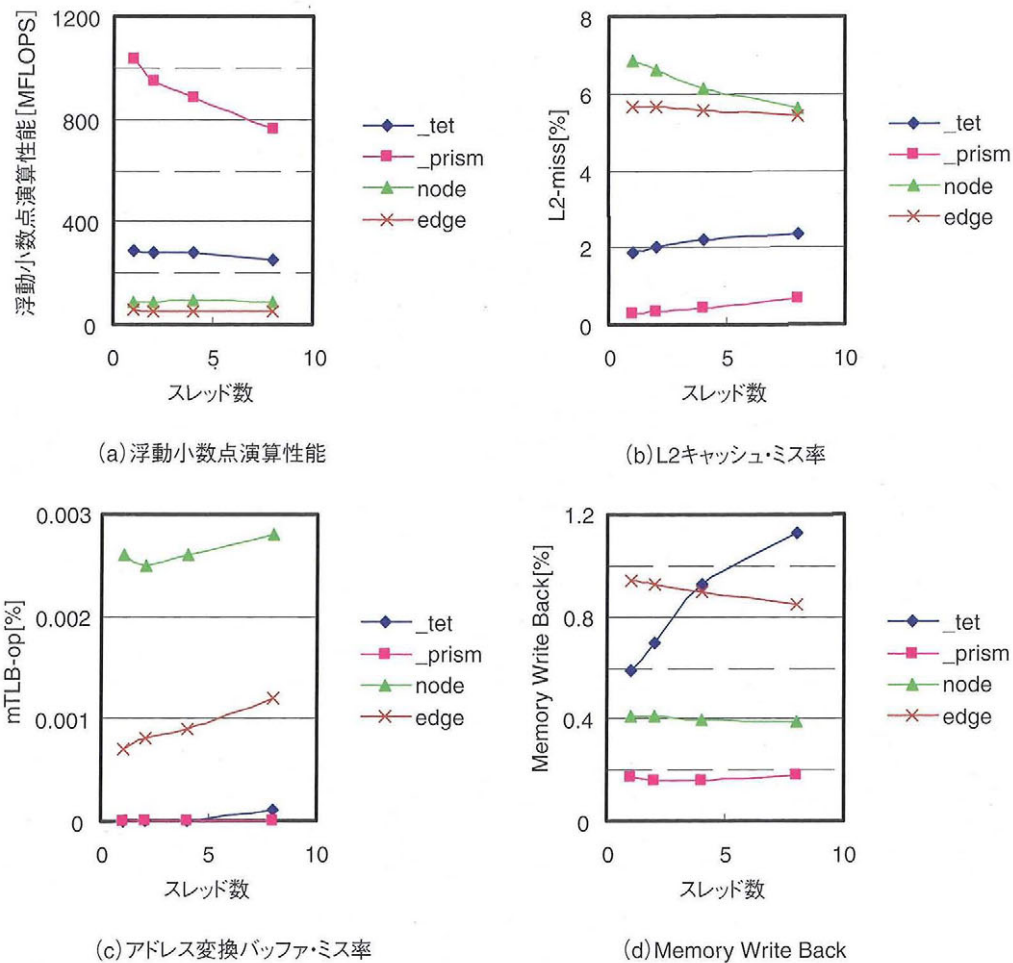


図 6.10 測定区間“SFDRF1”におけるプロファイラ情報抜粋

報からは確認できなかった。しかし、処理に要した経過時間は362.7秒から237.36秒に短縮されており最適化による計算性能向上の効果は確認された。一方で、スカラ版と同じCPU数による実行における経過時間を比較すると、スカラ版では16プロセスで206.16秒(表5.10参照)であるのに対してスレッド版では8スレッド(2プロセス)で237.36秒と1割以上多くの時間を要しており、性能面ではプロセス並列のみによる実行が優位であるという結果が得られた。また、スレッド数の増加によりL2キャッシュ・ミス率が低減することが期待されたが、その傾向を確認することは出来なかった。

### 6.3.4 測定区間“LIMITER”における測定結果及び評価

測定区間“LIMITER”では、Venkatakrisnanの制限関数 $\Psi_i$ の計算について、経過時間の測定及び実行状況の解析を行なった。この測定区間は、二つのサブルーチン“LIMITER1”及び“LIMITER2”によって構成される。表6.7にこの区間における測定及び解析結果を示す。測定区間“LIMITER1”は、サブルーチン“LIMITER1”に関する測定及び解析の結果であり、測定区間“LIMITER2”は、サブルーチン“LIMITER2”に関する測定及び解析の結果である。また、“オリジナル”及び“スレッド版”として示した経過時間は、サブルーチン“LIMITER1”及び“LIMITER2”の実行に要した時間の和である。いずれの経過時間も、バリア同期を取ってルートプロセス(ランク0)における測定結果のみを示している。測定区間“LIMITER2”における解析結果の上段及び下段は、測定区間“SPVCOR”における場合同様2つに分割されたDOループのLoop1及びLoop2それぞれにおける解

析結果である。

この測定区間においては、8スレッド実行の場合にオリジナルで7.54倍あったスレッド並列化による加速率がスレッド版では6.71倍に低下している（表6.2参照）。ここで、測定区間“SPDRF1”で行ったのと同様にアムダールの法則を適用し、スレッド並列化による並列化率及びオーバーヘッドを評価すると、オリジナルにおいて99.1%であった並列化率はスレッド版では98.7%に低下し、オーバーヘッドは、オリジナル版の1.8秒に対してスレッド版では4.4秒に増加している。オーバーヘッドの7割弱である3.0秒は測定区間“LIMITER1”で費やされている。表6.7及び図6.13に示したように、測定区間“LIMITER1”においては、スレッド数の増加とともにL2キャッシュミス率、アドレスバッファ変換ミス率及びMemory Write Backのいずれもが増加しており、このことがスレッド並列加速率低下の一因であると考えられる。しかし、処理に要した経過時間は、211.96秒から

表 6.7 測定区間“LIMITER”における測定結果（続く）

測定区間：LIMITER							上段：Loop 1
2 process 実行							下段：Loop 2
経過時間 [秒]							
時間積分500ステップ							
version	Proc. only	01Thread	02Thread	04Thread	08Thread	16Thread	
オリジナル	1598.96	1599.02	861.88	410.54	211.96		
LIMITER1	494.72	473.17	260.22	124.48	64.50		
LIMITER2	1104.24	1125.85	601.66	286.06	147.46		
スレッド版	1009.35	1087.35	581.07	310.65	161.96	83.83	
LIMITER1	276.12	311.51	173.60	102.51	54.18	28.10	
LIMITER2	733.23	775.84	407.47	208.14	107.78	55.73	
浮動小数点演算性能 FLOPS [MFLOPS]							
測定区間	オリジナル	スレッド版					
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread	
LIMITER1	40.6	76.7	78.5	77.1	75.2	69.8	
LIMITER2	205.2	314.0	411.8 65.0	407.8 60.4	405.9 60.8	384.2 57.6	
命令処理性能 MIPS [MIPS]							
測定区間	オリジナル	スレッド版					
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread	
LIMITER1	122.7	256.6	266.3	263.7	260.9	248.1	
LIMITER2	332.6	431.1	520.2 225.7	516.4 219.9	517.6 221.8	499.2 212.8	
L2 キャッシュ・ミス率 L2-miss [%]							
測定区間	オリジナル	スレッド版					
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread	
LIMITER1	8.61	2.89	2.88	3.20	3.76	3.95	
LIMITER2	2.10	1.80	1.38 4.51	1.37 4.47	1.36 4.16	1.35 3.67	

表 6.7 測定区間“LIMITER”における測定結果(続き)

測定区間：LIMITER 2 process 実行		LIMITER 2 上段：Loop 1 LIMITER 2 下段：Loop 2				
アドレス変換バッファ・ミス率 mTLB-op [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
LIMITER1	0.0001	0.0000	0.0000	0.0000	0.0000	0.0001
LIMITER2	0.0000	0.0000	0.0000 0.0001	0.0000 0.0001	0.0000 0.0002	0.0001 0.0003
Memory Write Back [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
LIMITER1	3.2075	0.5294	0.5240	0.7883	1.1210	1.2396
LIMITER2	0.5887	0.2667	0.2392 0.4356	0.2347 0.4327	0.2282 0.4053	0.2183 0.3676
浮動小数点演算命令の割合 [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
LIMITER1	33.09	29.91	29.47	29.22	28.83	28.13
LIMITER2	61.70	72.84	79.15 28.79	78.97 27.47	78.43 27.42	76.96 27.05
Load/Store 命令の割合 [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
LIMITER1	46.30	42.14	41.91	42.66	42.83	42.00
LIMITER2	33.46	16.96	15.80 24.49	15.87 24.77	16.10 24.74	16.63 24.67

161.96秒に短縮されており最適化による計算性能向上の効果は確認された。一方で、スカラ版と同じCPU数による実行における経過時間を比較すると、スカラ版が16プロセスで75.69秒(表5.12参照)であるのに対してスレッド版では8スレッド(2プロセス)で161.96秒を要しており、性能面ではプロセス並列のみによる実行が優位であるという結果が得られた。また、スレッド数の増加によりL2キャッシュ・ミス率が低減することが期待されたが、その傾向を確認することは出来なかった。

### 6.3.5 測定区間“LU-SGS”における測定結果及び評価

測定区間“LU-SGS”では、LU-SGS法により解を求める計算について、経過時間の測定及び実行状況の解析を行なった。表6.8及び図6.14から6.16に、この区間における測定及び解析結果を示す。各測定区間は、それぞれ同じ名前を持つサブルーチンに対応する。経過時間の測定において、サブルーチン“DIAGONAL”の経過時間を測定する区間“DIAGONAL”及びサブルーチン“UPDATEQ”の経過時間を測定する区間“UPDATEQ”に対してのみバリア同期を取ってルートプロセス(ランク0)における測定結果を示した。それ以外の測定区間では

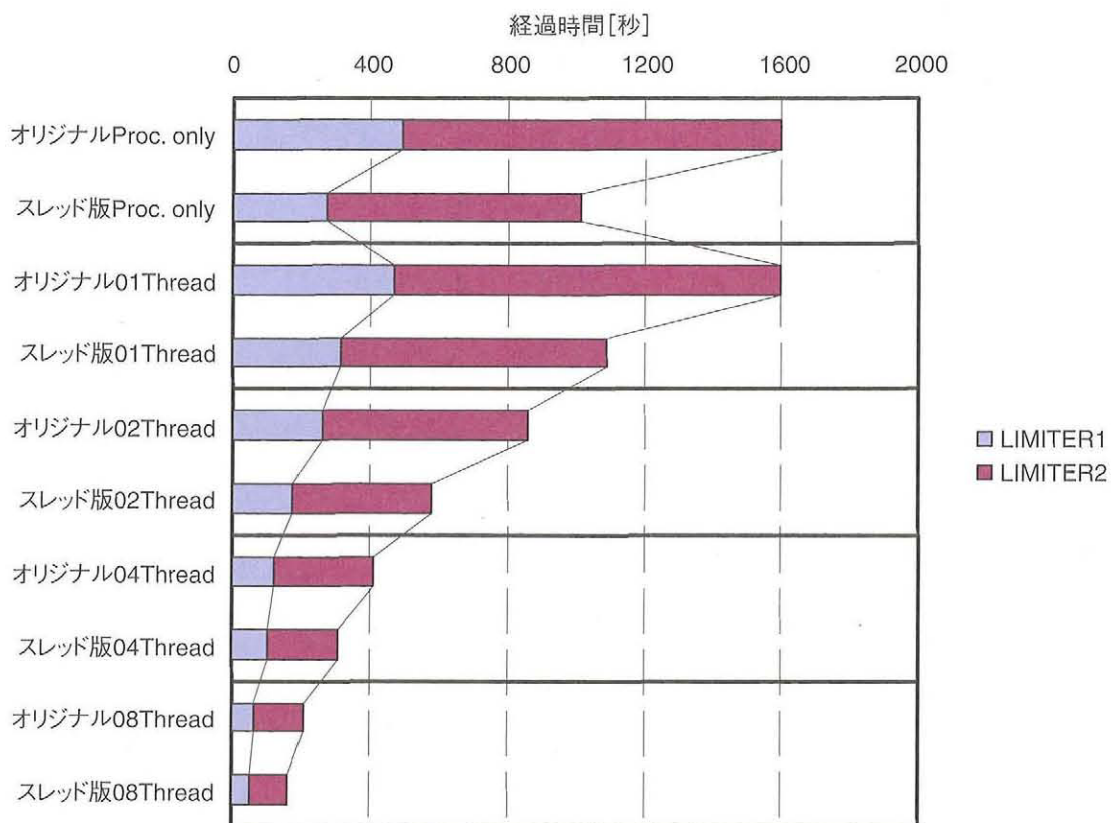


図 6.11 測定区間 “LIMITER” における測定結果

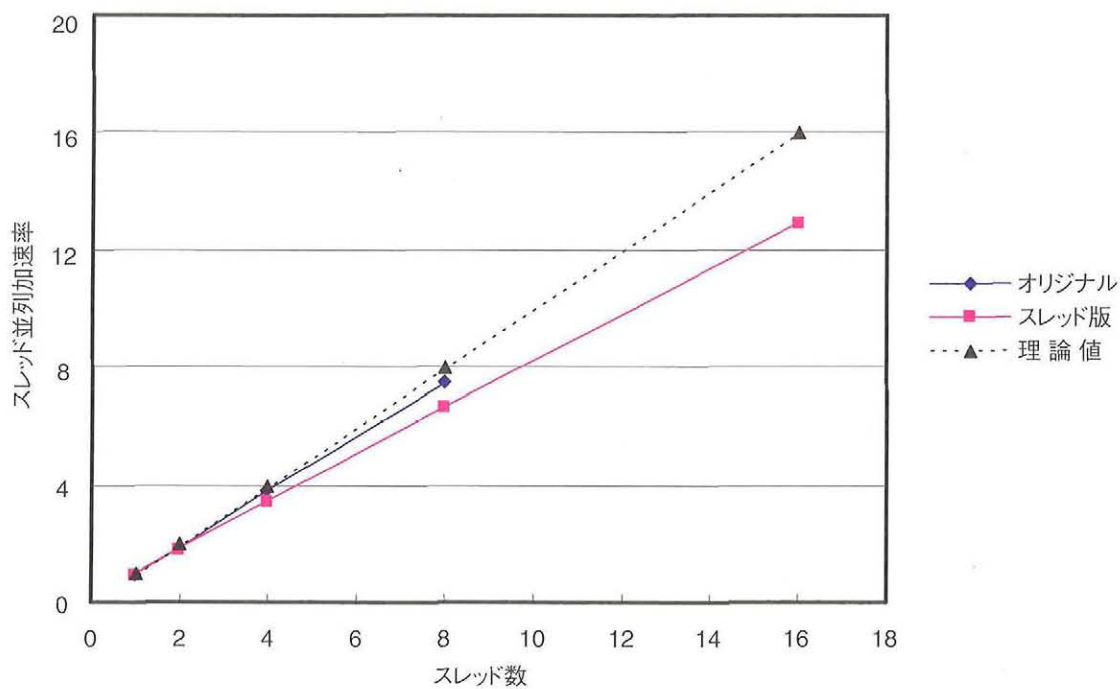


図 6.12 測定区間 “LIMITER” におけるスレッド並列実行による加速率

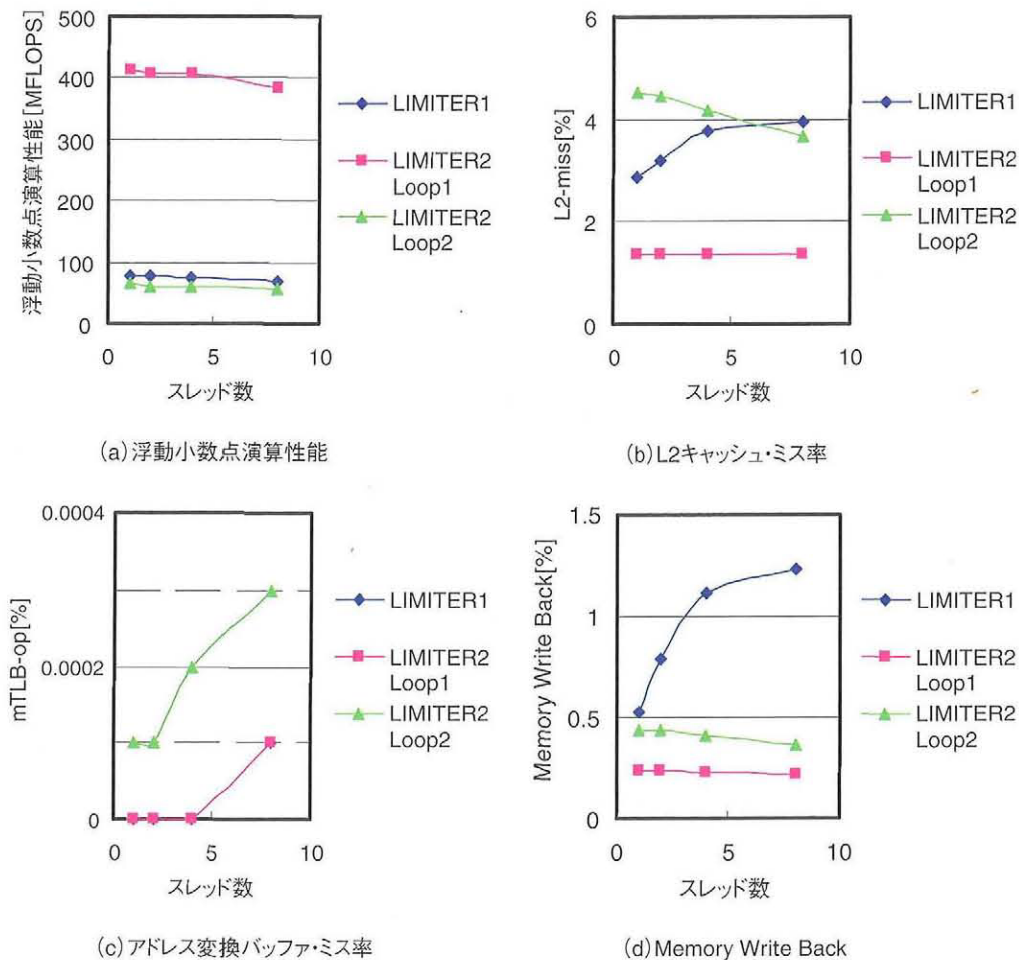


図 6.13 測定区間“LIMITER”におけるプロファイラ情報抜粋

バリア同期が取れないため、2つのプロセスそれぞれの経過時間を個別に測定し、ランク0のプロセスについてその値を上段に、ランク1のプロセスについてその値を下段に示した。測定区間“Copy”には、LU-SGS法において参照される保存量ベクトル  $\mathbf{Q} = [\rho, \rho u, \rho v, \rho w, e]^T$  や検査体積の体積及び表面積等を、計算に先立ってオリジナルの配列から超平面を考慮した配列にコピーする処理に要する経過時間及び解析結果を示した。また、“Copy [%]”にはそのコピー処理がLU-SGS法全体の経過時間に占める割合を百分率(%)で示した。

この測定区間においては、8スレッド実行の場合にオリジナルで2.97倍であったスレッド並列化による加速率がスレッド版では5.76倍に向上した。同時に、処理に要した経過時間も773.07秒から238.55秒に短縮されており最適化の効果は確認された(表6.2参照)。一方で、スカラ版と同じCPU数による実行における経過時間を比較すると、スカラ版が16プロセスで206.88秒(表5.13参照)であるのに対してスレッド版では8スレッド(2プロセス)で238.55秒を要しており、性能面ではプロセス並列のみによる実行が優位であるという結果が得られた。また、スレッド数の増加に伴いL2キャッシュ・ミス率が低減することによる性能向上が期待されたが、その効果が得られたのは、測定区間“Copy”においてのみであり、その他の測定区間で性能向上を確認することは出来なかった。さらに、測定区間“Copy”の処理に要する時間はLU-SGS法の計算全体の高々4%強であり、性能への影響は低いことがわかった。



表 6.8 測定区間 “LU-SGS” における測定結果 (続く)

測定区間：LU-SGS 2 process 実行						
経過時間 [秒] 時間積分500ステップ						上段：Rank 0 下段：Rank 1
Version	Proc. only	01Thread	02Thread	04Thread	08Thread	16Thread
オリジナル	2009.07	2297.07	1414.62	964.67	773.07	
DIAGONAL	384.80	411.35	221.48	107.63	56.77	
SIGMALEFT	531.20	559.95	292.73	138.92	82.01	
SWEEPL	508.20	570.60	279.05	145.20	89.31	
SIGMARIGHT	198.16	243.52	232.09	203.68	196.35	
SWEEPR	257.51	330.47	295.99	273.37	263.85	
UPDATEQ	504.31	528.77	274.72	131.39	78.59	
スレッド版	1356.23	1374.72	699.87	389.80	238.55	165.61
DIAGONAL	485.27	545.61	267.25	139.23	86.66	
SIGMALEFT	187.71	227.94	217.60	191.60	184.36	
SWEEPL	248.31	313.33	282.99	261.41	251.93	
SIGMARIGHT	23.46	26.95	13.27	6.84	3.65	
SWEEPR	400.85	432.14	214.32	112.79	60.66	34.37
UPDATEQ	355.07	370.87	177.25	99.22	61.55	48.71
Copy	367.46	364.69	181.51	103.04	65.35	50.12
Copy [%]	12.47	13.11	8.54	6.90	5.32	4.44
スレッド版	16.08	15.98	10.86	8.21	6.03	4.65
DIAGONAL	356.11	371.57	176.93	98.71	60.46	47.20
SIGMALEFT	374.29	372.11	184.10	103.51	65.64	49.25
SWEEPL	13.19	14.11	8.82	6.92	5.23	4.37
SIGMARIGHT	17.36	17.54	12.49	9.19	6.46	4.82
SWEEPR	35.22	34.07	17.26	8.95	5.12	3.20
UPDATEQ	57.92	58.20	29.45	16.07	9.65	6.30
Copy	4.27	4.23	4.21	4.12	4.05	3.8
浮動小数点演算性能 FLOPS [MFLOPS]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
DIAGONAL	71.1	71.1	70.5	65.9	63.2	52.3
SIGMALEFT	72.2	146.9	147.7	140.8	127.5	90.6
SWEEPL	68.9	85.9	149.8	141.4	128.9	91.3
SIGMARIGHT	77.4	144.7	145.5	138.9	125.5	88.5
SWEEPR	68.6	83.5	147.5	139.1	124.9	89.2
UPDATEQ	17.4	18.5	20.5	17.8	17.2	11.8
Copy		8.0	9.0	16.6	30.6	29.5
命令処理性能 MIPS [MIPS]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
DIAGONAL	130.7	130.9	133.4	131.1	130.2	122.9
SIGMALEFT	340.1	248.9	258.0	257.1	250.0	222.6
SWEEPL	480.6	187.1	263.1	259.4	255.9	228.9
SIGMARIGHT	356.1	249.9	253.8	253.2	246.8	224.4
SWEEPR	500.9	183.7	259.4	256.3	250.2	227.1
UPDATEQ	169.3	154.7	158.3	169.9	181.3	167.6
Copy		79.5	86.3	92.0	107.2	100.4

表 6.8 測定区間“LU-SGS”における測定結果(続き)

測定区間: LU-SGS		2 process 実行				
L2 キャッシュ・ミス率 L2-miss [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
DIAGONAL	12.10	12.08	11.75	11.40	11.32	9.85
SIGMALEFT	4.12	3.68	3.54	3.14	3.12	2.61
SWEEPL	3.10	5.34	3.33	3.00	3.00	2.57
SIGMARIGHT	3.85	3.66	3.51	2.96	2.65	2.14
SWEEPR	3.54	5.22	3.51	3.00	2.57	2.09
UPDATEQ	6.11	8.57	8.79	5.36	3.16	2.50
Copy		23.99	22.81	18.37	13.30	10.62
アドレス変換バッファ・ミス率 mTLB-op [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
DIAGONAL	0.0009	0.0009	0.0009	0.0010	0.0010	0.0012
SIGMALEFT	0.0270	0.0478	0.0533	0.0395	0.0224	0.0168
SWEEPL	0.0196	0.0093	0.0535	0.0419	0.0255	0.0181
SIGMARIGHT	0.0251	0.0429	0.0478	0.0364	0.0223	0.0148
SWEEPR	0.0192	0.0078	0.0457	0.0392	0.0235	0.0157
UPDATEQ	0.0002	0.0045	0.0207	0.0065	0.0053	0.0052
Copy		0.0056	0.0057	0.0055	0.0036	0.0032
Memory Write Back [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
DIAGONAL	2.0273	1.9320	1.8776	1.9503	2.1204	1.8175
SIGMALEFT	0.2420	0.2547	0.2429	0.2243	0.2378	0.2168
SWEEPL	0.2045	0.7912	0.2126	0.2074	0.2437	0.2050
SIGMARIGHT	0.2625	0.4238	0.4009	0.3121	0.2621	0.2163
SWEEPR	1.0072	1.0050	0.4258	0.3751	0.3096	0.2323
UPDATEQ	1.9736	3.0363	3.0209	1.7398	0.7999	0.4838
Copy		3.1544	2.9946	2.3878	1.7320	1.3060
浮動小数点演算命令の割合 [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
DIAGONAL	54.37	54.32	52.83	50.31	48.54	42.54
SIGMALEFT	21.16	59.05	57.26	54.74	51.02	40.71
SWEEPL	14.34	45.89	56.96	54.50	50.36	39.88
SIGMARIGHT	21.73	59.05	57.31	54.84	50.84	39.42
SWEEPR	13.70	45.47	56.86	54.27	49.94	39.26
UPDATEQ	10.25	11.99	12.96	10.45	9.47	7.04
Copy		10.20	10.43	18.02	28.58	29.44
Load/Store 命令の割合 [%]						
測定区間	オリジナル	スレッド版				
	Proc. only	Proc. only	01Thread	02Thread	04Thread	08Thread
DIAGONAL	23.65	23.52	23.53	24.24	25.04	25.52
SIGMALEFT	31.28	25.11	25.99	26.07	26.79	27.86
SWEEPL	31.98	26.72	25.61	26.00	26.95	28.04
SIGMARIGHT	31.17	24.83	25.70	25.66	25.89	26.85
SWEEPR	32.81	26.90	25.64	25.84	26.27	27.43
UPDATEQ	37.58	33.21	34.20	30.15	27.54	26.69
Copy		56.82	55.02	46.99	35.53	31.56

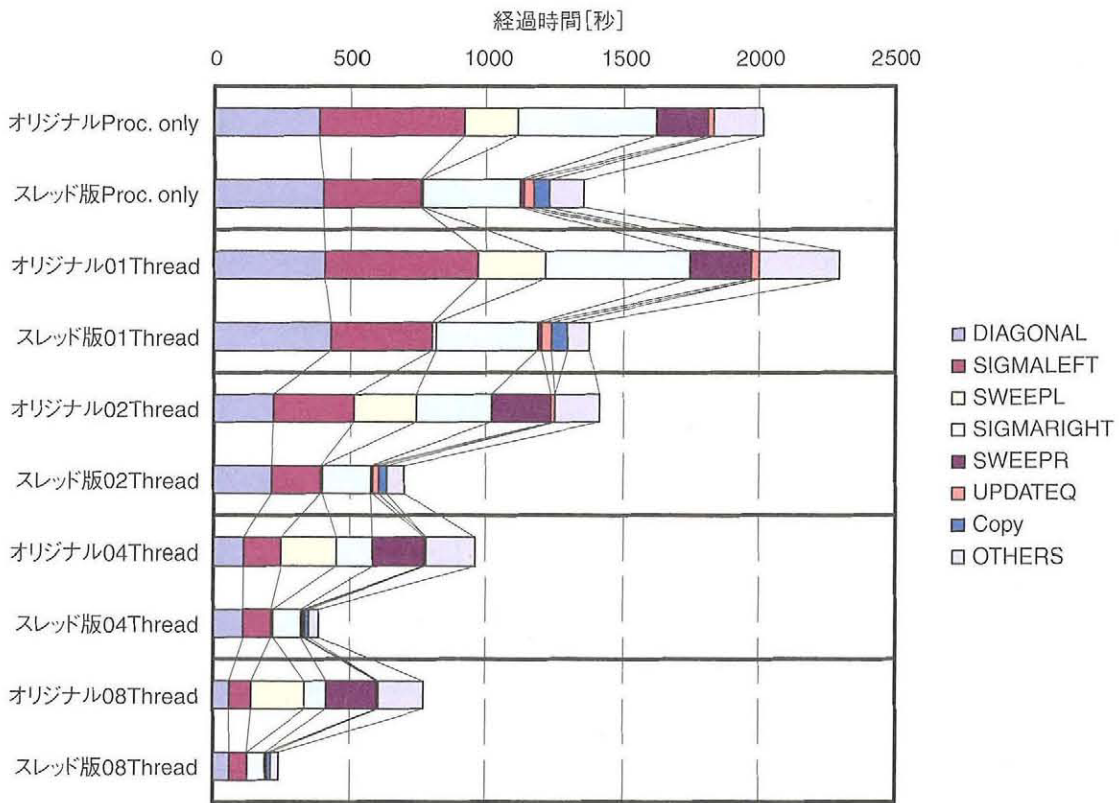


図 6.14 測定区間 "LU-SGS" における測定結果

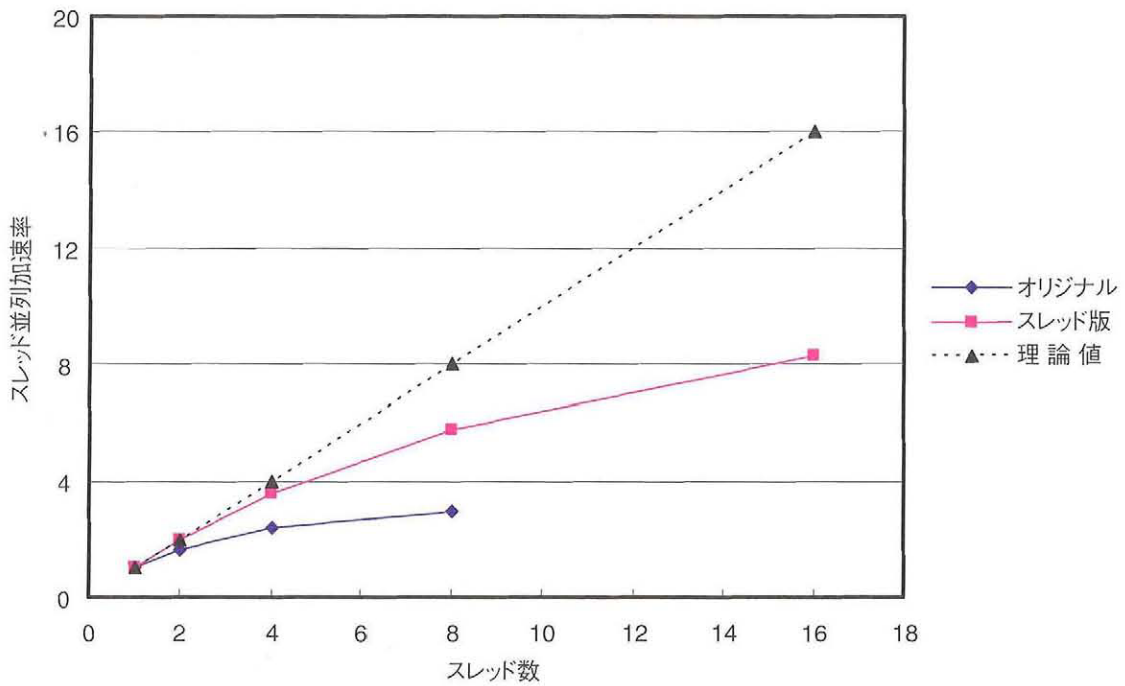


図 6.15 測定区間 "LU-SGS" におけるスレッド並列実行による加速率

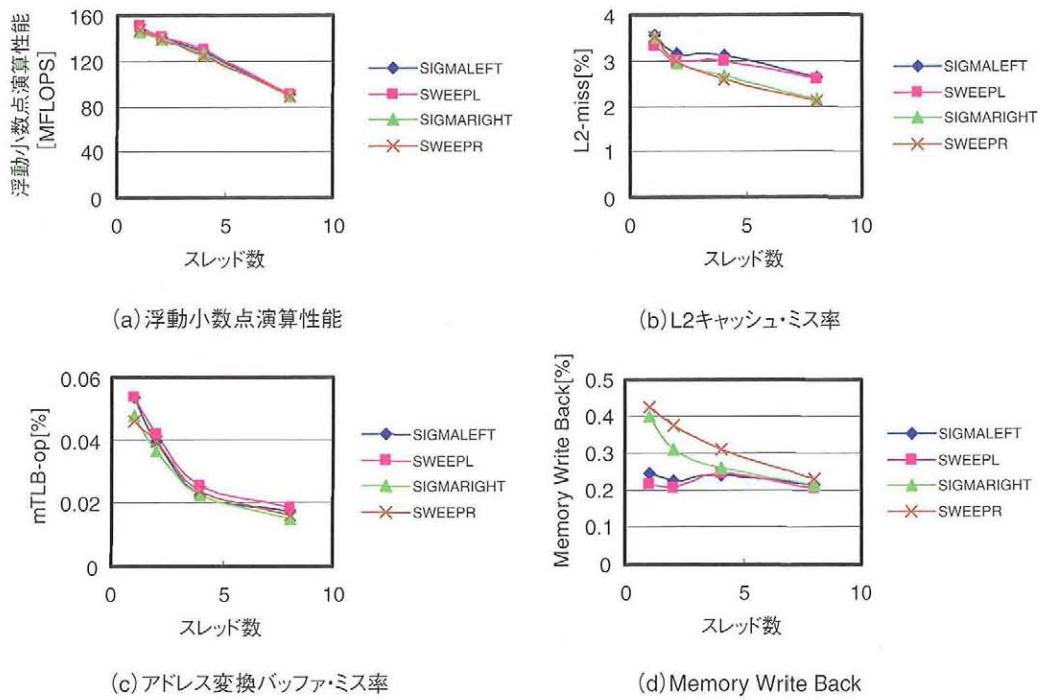


図 6.16 -1 測定区間“LU-SGS”におけるプロファイラ情報抜粋 (その1)

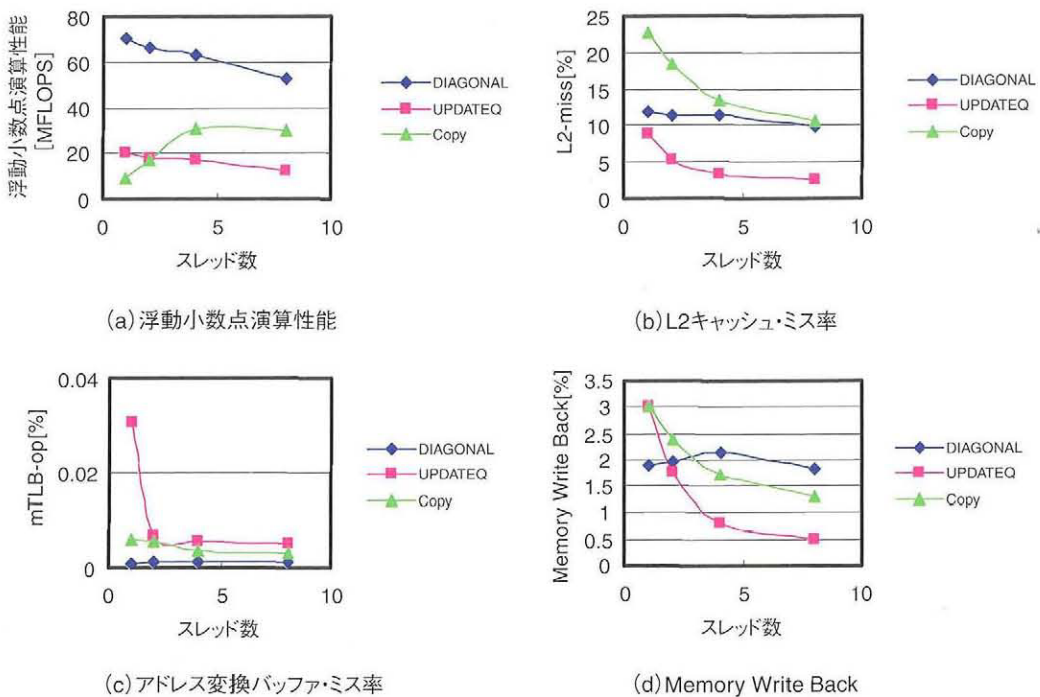


図 6.16 -2 測定区間“LU-SGS”におけるプロファイラ情報抜粋 (その2)

表6.9及び図6.17に測定区間“LU-SGS”においてコーディング方法を変更した場合に、性能がどのように変化するかを示す。コーディング方法としては、

- (1) 色分けにより再帰参照を回避した場合（オリジナル，リスト2.2参照）
- (2) 色分けを削除し DO ループを分割した場合（リスト3.1参照）
- (3) 色分けにより再帰参照を回避し，超平面を考慮した節点番号の付け替えを行った場合（スレッド版）
- (4) 色分けを削除し DO ループを分割するとともに，節点番号の付け替えを行った場合

の四種類を検討した。LU-SGS法における色分けの削除方法についての詳細は「APPENDIX B.2 LU-SGS法における色分け削除の効果確認のための変更」を参照されたい。ここで、JTASオリジナルにおいて色分けによるベクトル化が行われているのは、測定区間“DIAGONAL”，“SIGMALEFT”及び“SIGMARIGHT”である。これらの測定区間は、それぞれ同名のサブルーチンに対応している。表6.9及び図6.17では、それぞれの測定区間において各コーディング方法別に、スレッド並列による計算に要した経過時間を示した。表及び図中では、各コーディング方法を、順に「(1) 色分け（オリジナル）」、「(2) 色分け削除」、「(3) 色分け+Reo.（スレッド版）」及び「(4) 色除+Reo.」と略記した。また、測定区間“DIAGONAL”はLU-SGS法における超平面を考慮した節点番号の付け替えには関連しないため、この測定区間における「(3) 色分け+Reo.（スレッド版）」及び「(4) 色除+Reo.」の測定結果は示されていない。参考値として、スカラ版における経過時間も同時に示した。表6.9及び図6.17より明らかなように、色分けの削除を行うとオリジナルに比べて性能が低下すること、節点番号の付け替えのみを行った場合に最も良い性能が得られることが確認できた。色分けの削除を行った場合について測定区間（サブルーチン）“DIAGONAL”において調べてみると、MPI並列のみによる実行の場合に近似対角行列の計算を行うリスト3.3 DO100に相当する部分が284.47秒（経過時間）で計算されているのに対して、単に節点毎の配列に足し込むのみであるDO110に相当する部分に176.63秒（同）と、その60%以上もの時間を要しており、このことが色分けを削除しDOループを分割した場合に性能が低下する原因であると考えられる。以上のことから、スレッド版では節点番号の付け替えのみを行い色分けの削除は行わなかった。

表 6.9 測定区間“LU-SGS”におけるコーディング方法と性能の関係

経過時間 [秒]							2 process 実行 時間積分500ステップ
測定区間：LU-SGS							
Coding	Proc. only	01Thread	02Thread	04Thread	08Thread	16Thread	
(1) 色分け (オリジナル)	2009.07	2297.07	1414.62	964.67	773.07		
(2) 色分け削除	2408.36	2671.32	1950.94	1638.87	1269.93	724.49	
(3) 色分け+Reo. (スカラ版)	1356.23	1374.72	699.87	389.80	238.55	165.61	
(4) 色 除+Reo.	1844.72	1789.36	896.81	485.75	278.96	162.59	
スカラ版	1243.24						
測定区間：DIAGONAL							
Coding	Proc. only	01Thread	02Thread	04Thread	08Thread	16Thread	
(1) 色分け (オリジナル)	384.80	411.35	221.48	107.63	56.77		
(2) 色分け削除	465.22	446.98	262.27	144.70	66.60	22.52	
スカラ版	293.04						
測定区間：SIGMALEFT							
Coding	Proc. only	01Thread	02Thread	04Thread	08Thread	16Thread	
Rank 0							
(1) 色分け (オリジナル)	531.20	559.95	292.73	138.92	82.01		
(2) 色分け削除	683.63	712.90	522.14	416.40	328.90	193.25	
(3) 色分け+Reo. (スカラ版)	355.07	370.87	177.25	99.22	61.55	48.71	
(4) 色 除+Reo.	566.37	562.56	257.71	121.78	66.44	39.73	
スカラ版	321.50						
Rank 1							
(1) 色分け (オリジナル)	508.20	570.60	279.05	145.20	89.31		
(2) 色分け削除	662.26	745.93	516.54	432.97	336.57	191.97	
(3) 色分け+Reo. (スカラ版)	367.46	364.69	181.51	103.04	65.35	50.12	
(4) 色 除+Reo.	561.61	526.14	265.15	138.62	76.28	45.77	
スカラ版	331.99						
測定区間：SIGMARIGHT							
Coding	Proc. only	01Thread	02Thread	04Thread	08Thread	16Thread	
Rank 0							
(1) 色分け (オリジナル)	504.31	528.77	274.72	131.39	78.59		
(2) 色分け削除	613.03	682.52	508.45	403.83	322.47	190.24	
(3) 色分け+Reo. (スカラ版)	356.11	371.57	176.93	98.71	60.46	47.20	
(4) 色 除+Reo.	569.76	568.61	261.28	125.85	69.12	41.83	
スカラ版	319.34						
Rank 1							
(1) 色分け (オリジナル)	485.27	545.61	267.25	139.23	86.66		
(2) 色分け削除	590.36	708.19	497.91	418.38	328.99	189.03	
(3) 色分け+Reo. (スカラ版)	374.29	372.11	184.10	103.51	65.64	49.25	
(4) 色 除+Reo.	567.24	533.03	278.14	147.52	84.88	50.15	
スカラ版	331.86						

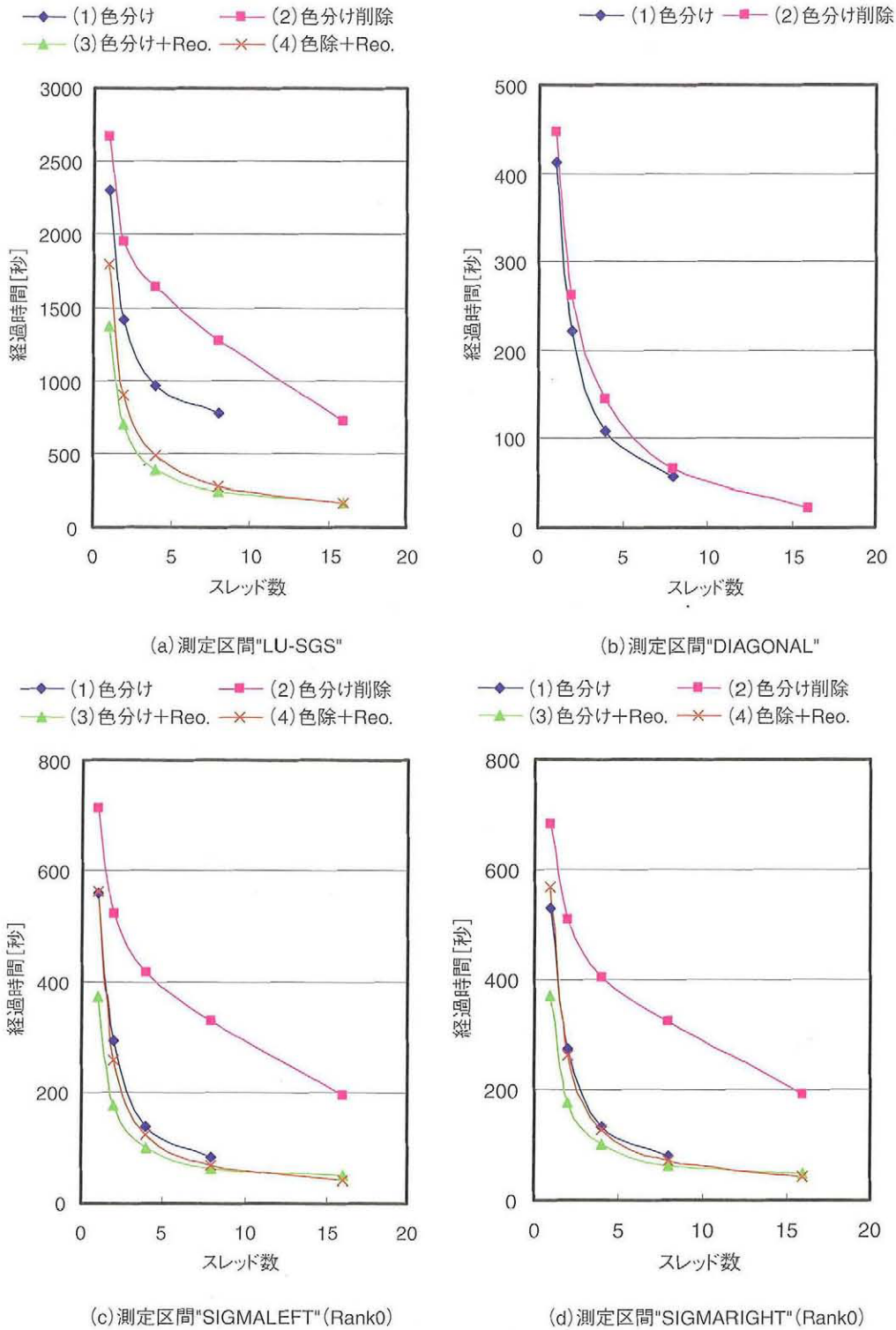


図 6.17 測定区間 "LU-SGS" におけるコーディング方法と性能の関係

## 7. 今後の課題

今回加えた変更により、JTAS オリジナルに比べてスカラ版で約1.8倍から1.9倍の性能向上が見られ最適化の効果が確認出来た。一方で、論理ピーク性能に対しては高々3から4%程度の性能に留まった。しかし、サブルーチン“SPDRF1\_pri”の計算（測定区間“\_prism”）に関して2から16プロセスによる実行において900MFLOPSを越える高い実効性能が得られている（表5.10参照）。このサブルーチンが、三角柱要素の勾配を三つの三角錐要素に分割して計算していることを考えれば、サブルーチン“SPDRF1\_tet”において、いくつかの三角錐要素を融合させて同時に計算することで、より高い性能が得られる可能性が示唆されたと言える。

また、配列の保存方法が性能に与える影響についても検討の余地があると考えられる。現在のJTASでは、例えば、数値流束の計算結果、検査体積における勾配等多くの値がひとつの配列“POLE”に保存されている。この配列をいくつかの配列に細分化することによりメモリキャッシュをより有効に活用できる可能性があり、今後の検討課題である。

最後に、スカラ版における測定区間“SPDRF1”及びスレッド版における測定区間“LU-SGS”において、他の測定区間とは異なるコーディング方法を採用することで良い性能が得られたが、その原因については明らかではない。この原因を究明することは、より良い性能を得ることが出来るコーディング方法を事前に知る手がかりになると考えられる。



## APPENDIX A. スカラ版チューニング内容詳細

ここでは、スカラ版で適用した変更内容について、その詳細を説明する。スカラ版においては、表 A.1 に示す五項目について変更を行った。以下では、その項目毎に変更内容の詳細を示す。

表 A.1 スカラ版において変更が加えられたルーチン一覧（続く）

(1) 勾配 $\nabla q_i$ の計算		
File 名	Subroutine 名	変 更 内 容
der.f	DR	要素毎に計算された勾配の節点毎及び辺毎ごとの配列への足し込み
	SPDRF1_pri	色分けの削除, DO ループの分割
	SPDRF1_pyr	色分けの削除, DO ループの分割
	SPDRF1_tet	色分けの削除, DO ループの分割
forfast.f	FORFAST	インデックステーブルの設定
	SET_ELEMENT_TABLE	インデックステーブルの設定 (新規作成ルーチン)
ACCUM2	<i>include file</i>	作業用配列からの足し込みに変更
dsed	<i>include file</i>	作業用配列からの足し込みに変更
DS3D	<i>include file</i>	作業用配列及びインデックステーブルの配列宣言追加
(2) Venkatakrisnan の制限関数 $\Psi_i$ の計算		
File 名	Subroutine 名	変 更 内 容
der.f	LIMITER1	色分けの削除
	LIMITER2	色分けの削除

表 A.1 スカラ版において変更が加えられたルーチン一覧 (続き)

(3) LU-SGS 法ソルバ部分		
File 名	Subroutine 名	変 更 内 容
lut2. f	LUSGS	作業用配列へのコピー
	DIAGONAL	色分けの削除
	DIAGONALT	色分けの削除 (変更のみ・未テスト)
	DIAGONALT_sa	色分けの削除 (変更のみ・未テスト)
	SIGMALEFT	色分けの削除, 節点番号の付け替え
	SIGMALEFTTT	色分けの削除, 節点番号の付け替え (変更のみ・未テスト)
	SIGMALEFTTT_sa	色分けの削除, 節点番号の付け替え (変更のみ・未テスト)
	SWEEPL	節点番号の付け替え
	SIGMARIGHT	色分けの削除, 節点番号の付け替え
	SIGMARIGHTTT	色分けの削除, 節点番号の付け替え (変更のみ・未テスト)
	SIGMARIGHTTT_sa	色分けの削除, 節点番号の付け替え (変更のみ・未テスト)
	SWEEPR	節点番号の付け替え
	UPDATEQ	節点番号の付け替えによる参照配列の変更
	ZEROSIGMA	節点番号の付け替えによる参照配列の変更
	ZERODELTA	節点番号の付け替えによる参照配列の変更
	SIGMAPREPARE	節点番号の付け替えによるインデックステーブルの変更, 新規作成
	LUSGSREAD	節点番号の付け替えにより変更されたインデックステーブルの入力
LUSGSWRITE	節点番号の付け替えにより変更されたインデックステーブルの出力	
mpi. f	SETS_SUB_MPICALL	節点番号の付け替えによる参照配列の変更
	SETR_SUB_MPICALL	節点番号の付け替えによる参照配列の変更
necns2. f	NMAIN	配列 “deltQ” のゼロクリア
DS3D	<i>include file</i>	インデックステーブルの配列宣言追加
LU	<i>include file</i>	作業用配列及びインデックステーブルの配列宣言追加
(4) HLEW 法による数値流束ベクトル $\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n}$ の計算		
File 名	Subroutine 名	変 更 内 容
necns2. f	SPVCOR	色分けの削除
(5) タイマの挿入		
File 名	Subroutine 名	変 更 内 容
timer_header.h	<i>include file</i>	タイマ計測に必要な変数及び配列の宣言
timer_init.h	<i>include file</i>	タイマ計測の初期化
timer_begin.h	<i>include file</i>	タイマ計測区間の開始
timer_end.h	<i>include file</i>	タイマ計測区間の終了
timer_print.h	<i>include file</i>	タイマ計測結果のまとめ及び標準出力ファイルへの出力
necns2.f	SPVOCL	変数名 “ETIME” の単精度実数宣言削除

A.1 勾配 $\nabla q_i$ の計算

ここでは、勾配 $\nabla q_i$ の計算に関して、スカラ版で適用した変更内容について以下に示す。

表 A.1.1 サブルーチン “DR” の変更内容

File 名	der.f
Subroutine 名	DR
変 更 内 容 の 説 明	
<p>サブルーチン “SPDRF1_tet”, “SPDRF1_pri” 及び “SPDRF1_pyr” において、配列の定義・参照の連続化を図るために分割された DO ループの内、節点及び辺の配列への足し込みを行う DO ループを、サブルーチンの呼び出し元である本サブルーチンに追加し、足し込みの一括処理を行うように変更した。          (「3.1.2 色分けの削除」及び「3.1.3 勾配計算における配列定義順序の連続化」参照)</p>	
変 更 内 容 リ ス ト	
<pre> call spdrf1_tet call spdrf1_pri call spdrf1_pyr  DO IP=1, N_P_F   NELM=IND_TBL(0, IP)   DO IE=1, NELM     IELM=IND_TBL(IE, IP)     include 'ACCUM2.ELM'   END DO END DO DO IED=1, N_ED_F   NELM=IED_TBL(0, IED)   DO IE=1, NELM     IELM=IED_TBL(IE, IED)     include 'dsed.ELM'   END DO END DO  call spdrf2_n </pre>	<p>足し込み処理を追加</p>

表 A.1.2 サブルーチン “SPDRF1\_pri” の変更内容

File 名	der.f	
Subroutine 名	SPDRF1_pri	
変 更 内 容 の 説 明		
<p>色分けを削除するとともに、DO ループを分割し要素毎の勾配を一旦作業用配列に保存することで、配列の定義・参照の連続化を図った。作業用配列は、三角錐要素の勾配の保存に利用した配列と同じものを使用し、これに追加することとした（「変更後」下線部参照）。</p> <p>節点及び辺の配列への足し込みは、サブルーチン “SPDRF1_tet” 及び “SPDRF1_pyr” と統合して、これらのサブルーチンの呼び出し元であるサブルーチン “dr” にて一括して行うように変更した。（「3.1.2 色分けの削除」及び「3.1.3 勾配計算における配列定義順序の連続化」参照）</p>		
変 更 内 容 リ ス ト		
	変 更 前	変 更 後
	<pre> DO 101 ICOLOR=1, MC_pri_N   jmin0=itc_pri2(1, icolor)   jmax =itc_pri2(2, icolor)   jminq=jmin0+1 *VOCL LOOP, NOVREC   DO 100 J=jminq, JMAX     IELM=itc_pri1(J)     ie1=jprism(1, ielm)       :     ie9=jprism(9, ielm)     l1=ids_ed(in1_ed, ie7)       :     l6=ids_ed(in2_ed, ie9)       :     ip=i1     include 'ACCUM2'       :     ip=i6     include 'ACCUM2'       :     ied=ie1     include 'dsed'       :     ied=ie9     include 'dsed' 100 CONTINUE 101 CONTINUE </pre>	<pre> DO 100 IELM=1, N_prism   IF (icol_pri (IELM).GE.1) THEN     ie1=jprism(1, ielm)       :     ie9=jprism(9, ielm)     l1=ids_ed(in1_ed, ie7)       :     l6=ids_ed(in2_ed, ie9)       :     IELMM=N_EL_F+IELM     elm_wk( 1,ielmm)= &amp;          drx0*v0+drx1*v1+drx2*v2       :     elm_wk(22,ielmm)= &amp;          dtz0*v0+dtz1*v1+dtz2*v2     END IF 100 CONTINUE </pre> <p>Subroutine “DR”に追加した処理</p> <pre> DO IP=1, N_P_F   NELM=IND_TBL (0, IP)   DO IE=1, NELM     IELM=IND_TBL (IE, IP)     include 'ACCUM2. ELM'   END DO END DO DO IED=1, N_ED_F   NELM=IED_TBL (0, IED)   DO IE=1, NELM     IELM=IED_TBL (IE, IED)     include 'dsed. ELM'   END DO END DO </pre>

表 A.1.3 サブルーチン “SPDRF1\_pyr” の変更内容

File 名	der.f
Subroutine 名	SPDRF1_pyr
変 更 内 容 の 説 明	
<p>色分けを削除するとともに、DO ループを分割し要素毎の勾配を一旦作業用配列に保存することで、配列の定義・参照の連続化を図った。作業用配列は、三角錐要素及び三角柱要素の勾配の保存に利用した配列と同じものを使用し、これに追加することとした（「変更後」下線部参照）。</p> <p>節点及び辺の配列への足し込みは、サブルーチン “SPDRF1_tet” 及び “SPDRF1_pri” と統合して、これらのサブルーチンの呼び出し元であるサブルーチン “dr” にて一括して行うように変更した。（「3.1.2 色分けの削除」及び「3.1.3 勾配計算における配列定義順序の連続化」参照）</p>	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> DO 101 ICOLOR=1,MC_pyr_N   jmin0=itc_pyr2(1,icolor)   jmax =itc_pyr2(2,icolor)   jminq=jmin0+1 *VOCL LOOP,NOVREC   DO 100 J=jminq,JMAX     IELM=itc_pyr1(J)     I1=ipyr(1,IELM)     :     I5=ipyr(5,IELM)     ie1=jpyr(1,ielm)     :     ie8=jpyr(8,ielm)     :     ip=i1     include 'ACCUM2'     :     ip=i5     include 'ACCUM2'     :     ied=ie1     include 'dsed'     :     ied=ie9     include 'dsed' 100 CONTINUE 101 CONTINUE </pre>	<pre> DO 100 IELM=1,N_pyr   IF(Icol_pyr(IELM).GE.1) THEN     I1=ipyr(1,IELM)     :     I5=ipyr(5,IELM)     ie1=jpyr(1,ielm)     :     ie8=jpyr(8,ielm)     :     IELMM=N_EL_F+N_prism+IELM     elm_wk(1,ielmm)=drx0*v0+drx1*v1     :     elm_wk(22,ielmm)=dtz0*v0+dtz1*v1   END IF 100 CONTINUE </pre> <p>Subroutine “DR”に追加した処理</p> <pre> DO IP=1,N_P_F   NELM=IND_TBL(0,IP)   DO IE=1,NELM     IELM=IND_TBL(IE,IP)     include 'ACCUM2.ELM'   END DO END DO DO IED=1,N_ED_F   NELM=IED_TBL(0,IED)   DO IE=1,NELM     IELM=IED_TBL(IE,IED)     include 'dsed.ELM'   END DO END DO </pre>

表 A.1.4 サブルーチン “SPDRF1\_tet” の変更内容

File 名	der.f
Subroutine 名	SPDRF1_tet
変 更 内 容 の 説 明	
<p>色分けを削除するとともに、DO ループを分割し要素毎の勾配を一旦作業用配列に保存することで、配列の定義・参照の連続化を図った。</p> <p>節点及び辺の配列への足し込みは、サブルーチン “SPDRF1_pri” 及び “SPDRF1_pyr” と統合して、これらのサブルーチンの呼び出し元であるサブルーチン “dr” にて一括して行うように変更した。 (「3.1.2 色分けの削除」及び「3.1.3 勾配計算における配列定義順序の連続化」参照)</p>	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> DO 101 ICOLOR=1, MC_E_N   jmin0=itc_el2(1, icolor)   jmax =itc_el2(2, icolor)   jminq=jmin0+1 *VOCL LOOP, NOVREC   DO 100 J=jminq, JMAX     IELM=itc_el1(J)     IEDGE1=IDS_EL(IE1_EL, IELM)     :     IEDGE6=IDS_EL(IE6_EL, IELM)     I1=IDS_ED(IN1_ED, IEDGE1)     :     I4=IDS_ED(IN2_ED, IEDGE4)     :     ip=i1     include 'ACCUM2'     :     ip=i4     include 'ACCUM2'      ied=iedge1     include 'dsed'     :     ied=iedge6     include 'dsed' 100 CONTINUE 101 CONTINUE </pre>	<pre> DO 100 IELM=1, N_EL_F   IF (IDS_EL(ICL_EL, IELM).GE.1) THEN     IEDGE1=IDS_EL(IE1_EL, IELM)     :     IEDGE6=IDS_EL(IE6_EL, IELM)     I1=IDS_ED(IN1_ED, IEDGE1)     :     I4=IDS_ED(IN2_ED, IEDGE4)     :     elm_wk( 1, ielm)=v0*drx     :     elm_wk(22, ielm)=v0*dtz   END IF 100 CONTINUE </pre> <p>Subroutine “DR”に追加した処理</p> <pre> DO IP=1, N_P_F   NELM=IND_TBL(0, IP)   DO IE=1, NELM     IELM=IND_TBL(IE, IP)     include 'ACCUM2.ELM'   END DO END DO DO IED=1, N_ED_F   NELM=IED_TBL(0, IED)   DO IE=1, NELM     IELM=IED_TBL(IE, IED)     include 'dsed.ELM'   END DO END DO </pre>

表 A.1.5 サブルーチン “FORFAST” の変更内容

File 名	forfast. f
Subroutine 名	FORFAST
変更内容の説明	
<p>節点-要素対応テーブル “IND_TBL” 及び辺-要素対応テーブル “IED_TBL” を設定する処理を追加</p> <p><u>節点-要素対応テーブル IND_TBL</u>  <math>nelm = IND\_TBL(0, nd)</math> : 節点 <math>nd</math> を共有する要素の数 <math>nelm</math>  <math>ielm = IND\_TBL(ie, nd)</math> : 節点 <math>nd</math> を共有する <math>ie</math> 番目の要素の要素番号 <math>ielm</math></p> <p><u>辺-要素対応テーブル IED_TBL</u>  <math>nelm = IED\_TBL(0, ed)</math> : 辺 <math>ed</math> を共有する要素の数 <math>nelm</math>  <math>ielm = IED\_TBL(ie, ed)</math> : 辺 <math>ed</math> を共有する <math>ie</math> 番目の要素の要素番号 <math>ielm</math></p>	
変更内容リスト	
<p>(サブルーチン “FORFAST” の最後に以下の処理を追加)</p> <pre> DO J=1, IMX_ED DO I=0, IMX_ELED   IED_TBL(I, J) = 0 END DO END DO DO J=1, IMX_ND DO I=0, IMX_ELND   IND_TBL(I, J) = 0 END DO END DO C DO IELM=1, N_EL_F   IF (IDS_EL( ICL_EL, IELM).GE.1) THEN     :     CALL SET_ELEMENT_TABLE( IED_TBL, IMX_ELED, IMX_ED , &amp;                          IEDGE1, IELM , &amp;                          'Fail FORFAST EL in ED ' )     :     CALL SET_ELEMENT_TABLE( IED_TBL, IMX_ELED, IMX_ED , &amp;                          IEDGE6, IELM , &amp;                          'Fail FORFAST EL in ED ' )     I1=IDS_ED(IN1_ED, IEDGE1)     :     I4=IDS_ED(IN2_ED, IEDGE4) C     CALL SET_ELEMENT_TABLE( IND_TBL, IMX_ELND, IMX_ND , &amp;                          I1, IELM , &amp;                          'Fail FORFAST EL in ND ' )     :     CALL SET_ELEMENT_TABLE( IND_TBL, IMX_ELND, IMX_ND , &amp;                          I4, IELM , &amp;                          'Fail FORFAST EL in ND ' ) C   END IF END DO </pre> <p>(以下、三角柱要素及び四角錐要素についても同様)</p>	

表 A.1.6 新規作成サブルーチン “SET\_ELEMENT\_TABLE”

File 名	forfast. f
Subroutine 名	SET_ELEMENT_TABLE( INDX_TBL, MXSIZ1, MXSIZ2, INDX, IELM, MSG )
変更内容の説明	
<p>新規作成ルーチン          節点-要素対応テーブル “IND_TBL” 及び辺-要素対応テーブル “IED_TBL” を設定するサブルーチンを新規作成</p> <p>処理内容</p> <ol style="list-style-type: none"> <li>1) INDX_TBL (0, INDX) に 1 を加算した値を “II” とする.</li> <li>2) II &gt; MXSIZ1 ならば, “MSG” を標準出力に出力した後, “MPI_abort” を呼び出しジョブを異常終了させる.</li> <li>3) II ≤ MXSIZ1 ならば, IED_TBL (0, INDX) = II, IND_TBL (II, INDX) = IELM とする.</li> </ol>	
変更内容リスト	
<pre> SUBROUTINE SET_ELEMENT_TABLE( INDX_TBL, MXSIZ1, MXSIZ2, &amp;                               INDX      , IELM  , MSG  )   USE MPI_communication   DIMENSION INDX_TBL(0:MXSIZ1, MXSIZ2)   CHARACTER MSG*(*)   II = INDX_TBL(0, INDX) + 1   IF (II.GT. MXSIZ1) THEN     WRITE(*,*) MSG, MXSIZ1, II     CALL MPI_abort(MPI_COMM_WORLD, 1)     STOP   END IF   INDX_TBL(0, INDX) = II   INDX_TBL(II, INDX) = IELM END </pre>	



表 A.1.7 インクルードファイル “ACCUM2” の変更内容

File 名	ACCUM2		
Subroutine 名	include file		
変 更 内 容 の 説 明			
<p>サブルーチン “SPDRF1_tet”, “SPDRF1_pri” 及び “SPDRF1_pyr” において, DO ループを分割したことによる変更である. 要素毎の勾配を, 節点の配列に直接足し込まず, 計算結果が保存された配列から足し込むように変更した.</p> <p>(「3.1.3 勾配計算における配列定義順序の連続化」参照)</p>			
変 更 前		変 更 後	
File 名	ACCUM 2	File 名	ACCUM2. ELM
変更内容リスト	<pre>pole(krx_pl , ip)=pole(krx_pl , ip)+drx pole(kry_pl , ip)=pole(kry_pl , ip)+dry pole(krz_pl , ip)=pole(krz_pl , ip)+drz pole(kux_pl , ip)=pole(kux_pl , ip)+duxx pole(kuy_pl , ip)=pole(kuy_pl , ip)+duxy pole(kuz_pl , ip)=pole(kuz_pl , ip)+duxz pole(kvz_pl , ip)=pole(kvz_pl , ip)+duyz pole(kvx_pl , ip)=pole(kvx_pl , ip)+duyx pole(kvy_pl , ip)=pole(kvy_pl , ip)+duyy pole(kwz_pl , ip)=pole(kwz_pl , ip)+duzz pole(kwx_pl , ip)=pole(kwx_pl , ip)+duzx pole(kwy_pl , ip)=pole(kwy_pl , ip)+duzy pole(kpz_pl , ip)=pole(kpz_pl , ip)+dpz pole(kpx_pl , ip)=pole(kpx_pl , ip)+dpz pole(kpy_pl , ip)=pole(kpy_pl , ip)+dpy pole(krtx_pl , ip)=pole(krtx_pl , ip)+drtx pole(krty_pl , ip)=pole(krty_pl , ip)+drty pole(krtz_pl , ip)=pole(krtz_pl , ip)+drtz pole(ip1_pl , ip)=pole(ip1_pl , ip)+v</pre>	<pre>pole(krx_pl , ip)=pole(krx_pl , ip)+elm_wk( 1, ielm) pole(kry_pl , ip)=pole(kry_pl , ip)+elm_wk( 2, ielm) pole(krz_pl , ip)=pole(krz_pl , ip)+elm_wk( 3, ielm) pole(kux_pl , ip)=pole(kux_pl , ip)+elm_wk( 4, ielm) pole(kuy_pl , ip)=pole(kuy_pl , ip)+elm_wk( 5, ielm) pole(kuz_pl , ip)=pole(kuz_pl , ip)+elm_wk( 6, ielm) pole(kvz_pl , ip)=pole(kvz_pl , ip)+elm_wk( 7, ielm) pole(kvx_pl , ip)=pole(kvx_pl , ip)+elm_wk( 8, ielm) pole(kvy_pl , ip)=pole(kvy_pl , ip)+elm_wk( 8, ielm) pole(kwz_pl , ip)=pole(kwz_pl , ip)+elm_wk( 9, ielm) pole(kwx_pl , ip)=pole(kwx_pl , ip)+elm_wk(10, ielm) pole(kwy_pl , ip)=pole(kwy_pl , ip)+elm_wk(11, ielm) pole(kwz_pl , ip)=pole(kwz_pl , ip)+elm_wk(12, ielm) pole(kpx_pl , ip)=pole(kpx_pl , ip)+elm_wk(13, ielm) pole(kpy_pl , ip)=pole(kpy_pl , ip)+elm_wk(14, ielm) pole(kpz_pl , ip)=pole(kpz_pl , ip)+elm_wk(15, ielm) pole(krtx_pl , ip)=pole(krtx_pl , ip)+elm_wk(16, ielm) pole(krty_pl , ip)=pole(krty_pl , ip)+elm_wk(17, ielm) pole(krtz_pl , ip)=pole(krtz_pl , ip)+elm_wk(18, ielm) pole(ip1_pl , ip)=pole(ip1_pl , ip)+elm_wk(19, ielm)</pre>	

表 A.1.8 インクルードファイル “dsed” の変更内容

File 名	dsed		
Subroutine 名	include file		
変 更 内 容 の 説 明			
<p>サブルーチン “SPDRF1_tet”, “SPDRF1_pri” 及び “SPDRF1_pyr” において, DO ループを分割したことによる変更である. 要素毎の勾配を, 辺の配列に直接足し込まず, 計算結果が保存された配列から足し込むように変更した.</p> <p>(「3.1.3 勾配計算における配列定義順序の連続化」参照)</p>			
変 更 前		変 更 後	
File 名	dsed	File 名	dsed. ELM
変更内容リスト	<pre> ds_ed(ktx_ed , ied)=ds_ed(ktx_ed , ied)+dtx ds_ed(kty_ed , ied)=ds_ed(kty_ed , ied)+dty ds_ed(ktz_ed , ied)=ds_ed(ktz_ed , ied)+dtz ds_ed(kux_ed , ied)=ds_ed(kux_ed , ied)+duxx ds_ed(kuy_ed , ied)=ds_ed(kuy_ed , ied)+duyx ds_ed(kuz_ed , ied)=ds_ed(kuz_ed , ied)+duxz ds_ed(kvz_ed , ied)=ds_ed(kvz_ed , ied)+duyz ds_ed(kvx_ed , ied)=ds_ed(kvz_ed , ied)+duyx ds_ed(kvy_ed , ied)=ds_ed(kvy_ed , ied)+duyy ds_ed(kwx_ed , ied)=ds_ed(kwx_ed , ied)+duzx ds_ed(kwy_ed , ied)=ds_ed(kwy_ed , ied)+duzy ds_ed(kwz_ed , ied)=ds_ed(kwz_ed , ied)+duzz ds_ed(kvl_ed , ied)=ds_ed(kvl_ed , ied)+v ds_ed(krtx_ed , ied)=ds_ed(krtx_ed , ied)+drtx ds_ed(krty_ed , ied)=ds_ed(krty_ed , ied)+drty ds_ed(krtz_ed , ied)=ds_ed(krtz_ed , ied)+drtz </pre>		
			<pre> ds_ed(ktx_ed , ied)=ds_ed(ktx_ed , ied)+elm_wk(20, ielm) ds_ed(kty_ed , ied)=ds_ed(kty_ed , ied)+elm_wk(21, ielm) ds_ed(ktz_ed , ied)=ds_ed(ktz_ed , ied)+elm_wk(22, ielm) ds_ed(kux_ed , ied)=ds_ed(kux_ed , ied)+elm_wk( 4, ielm) ds_ed(kuy_ed , ied)=ds_ed(kuy_ed , ied)+elm_wk( 5, ielm) ds_ed(kuz_ed , ied)=ds_ed(kuz_ed , ied)+elm_wk( 6, ielm) ds_ed(kvz_ed , ied)=ds_ed(kvz_ed , ied)+elm_wk( 7, ielm) ds_ed(kvx_ed , ied)=ds_ed(kvx_ed , ied)+elm_wk( 8, ielm) ds_ed(kvy_ed , ied)=ds_ed(kvy_ed , ied)+elm_wk( 9, ielm) ds_ed(kwx_ed , ied)=ds_ed(kwx_ed , ied)+elm_wk(10, ielm) ds_ed(kwy_ed , ied)=ds_ed(kwy_ed , ied)+elm_wk(11, ielm) ds_ed(kwz_ed , ied)=ds_ed(kwz_ed , ied)+elm_wk(12, ielm) ds_ed(kvl_ed , ied)=ds_ed(kvl_ed , ied)+elm_wk(19, ielm) ds_ed(krtx_ed , ied)=ds_ed(krtx_ed , ied)+elm_wk(16, ielm) ds_ed(krty_ed , ied)=ds_ed(krty_ed , ied)+elm_wk(17, ielm) ds_ed(krtz_ed , ied)=ds_ed(krtz_ed , ied)+elm_wk(18, ielm) </pre>

表 A.1.9 インクルードファイル “DS3D” の変更内容

File 名	DS3D		
Subroutine 名	include file		
変 更 内 容 の 説 明			
<p>サブルーチン “SPDRF1_tet”, “SPDRF1_pri” 及び “SPDRF1_pyr” において, DO ループを分割したために必要となった, パラメーター文の宣言, 作業用配列及びインデックステーブルの配列宣言を追加した. 追加した作業用配列及びインデックステーブルは以下の通り.          (「3.1.3 勾配計算における配列定義順序の連続化」参照)</p>			
変数名	型	サイズ	説 明
IMX_ELND	I4	1	同一の節点を共有する要素の数の最大値
IMX_ELED	I4	1	同一の辺を共有する要素の数の最大値
ELM_WK	R8	(IMX_EL+IMX_PRISM+IMX_PYR)	各要素における勾配を保存するための作業用配列
IND_TBL	I4	(0:IMX_ELND,IMX_ND)	節点-要素対応テーブル IND_TBL (0, nd) : 節点 nd を共有する要素の数 IND_TBL (ie, nd) : 節点 nd を共有する ie 番目の要素の要素番号
IED_TBL	I4	(0:IMX_ELED,IMX_ED)	辺-要素対応テーブル IED_TBL (0, ed) : 辺 ed を共有する要素の数 IED_TBL (ie, ed) : 辺 ed を共有する ie 番目の要素の要素番号
IDS_TBL	I4	( 2,IMX_ED)	辺-節点对応テーブル IDS_TBL ( , ed) : 辺 ed が含む新しい節点の番号 LU-SGS 法における変更により追加
変 更 内 容 リ ス ト			
<pre>COMMON/ITLEFT/ itleft COMMON/TILEFT/ tileft  PARAMETER(IMX_ELND =50, &amp;          IMX_ELED =10) COMMON/EL_WRK/ &amp;          ELM_WK(22, IMX_EL+IMX_PRISM+IMX_PYR) COMMON/ND_WRK/IND_TBL(0:IMX_ELND, IMX_ND), COMMON/ED_WRK/IED_TBL(0:IMX_ELED, IMX_ED) &amp;          IDS_TBL( 2, IMX_ED)</pre>			
			} 作業用配列及びインデックステーブルを追加

A.2 Venkatakrishnan の制限関数  $\Psi_i$  の計算

ここでは、Venkatakrishnan の制限関数  $\Psi_i$  の計算に関して、スカラ版で適用した変更内容について以下に示す。

表 A.2.1 サブルーチン “LIMITER1” の変更内容

File 名	der.f
Subroutine 名	LIMITER1
変更内容の説明	
色分けを削除し、配列の定義・参照の局所化を図った。(「3.1.2 色分けの削除」参照)	
変更内容リスト	
変更前	変更後
<pre> DO 10 ICOLOR=1, MCL_ED   jmin0=itc_ed2(1, icolor)   jmax =itc_ed2(2, icolor)   jminq=jmin0+1 *VOCL LOOP,NOVREC   DO 1100 J=jminq, JMAX     IEDGE=itc_ed1(J)     I2=IDS_ED(IN1_ED, IEDGE)     I3=IDS_ED(IN2_ED, IEDGE)     :     pole(i11_pl , i2)= &amp;      min(pole(i11_pl , i2), r3)     pole(i12_pl , i2)= &amp;      max(pole(i12_pl , i2), r3)     :     pole(irt2_pl, i3)= &amp;      min(pole(irt2_pl, i3), rt2)     pole(irt3_pl, i3)= &amp;      max(pole(irt3_pl, i3), rt2) 1100 CONTINUE 10 CONTINUE </pre>	<pre> DO 100 IEDGE=1, N_ED_F   IF (IDS_ED(ICL_ED, IEDGE). GE. 1) THEN     I2=IDS_ED(IN1_ED, IEDGE)     I3=IDS_ED(IN2_ED, IEDGE)     :     pole(i11_pl , i2)= &amp;      min(pole(i11_pl , i2), r3)     pole(i12_pl , i2)= &amp;      max(pole(i12_pl , i2), r3)     :     pole(irt2_pl, i3)= &amp;      min(pole(irt2_pl, i3), rt2)     pole(irt3_pl, i3)= &amp;      max(pole(irt3_pl, i3), rt2)   END IF 100 CONTINUE </pre>

表 A.2.2 サブルーチン “LIMITER2” の変更内容

File 名	der. f
Subroutine 名	LIMITER2
変 更 内 容 の 説 明	
色分けを削除し、配列の定義・参照の局所化を図った。(「3.1.2 色分けの削除」参照)	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> DO 10 ICOLOR=1,MCL_ED   jmin0=itc_ed2(1,icolor)   jmax =itc_ed2(2,icolor)   jminq=jmin0+1 *VOCL LOOP,NOVREC   DO 1100 J=jminq,JMAX     IEDGE=itc_ed1(J)     I2=IDS_ED(IN1_ED, IEDGE)     I3=IDS_ED(IN2_ED, IEDGE)     :     del2=r1     qmax=rmax2     qmin=rmin2     q =r2     kq =kq1_pl     ll =i2     include 'lim'     :     : 1100 CONTINUE 10 CONTINUE </pre>	<pre> DO 100 IEDGE=1,N_ED_F   IF (IDS_ED(ICL_ED, IEDGE).GE.1) THEN     I2=IDS_ED(IN1_ED, IEDGE)     I3=IDS_ED(IN2_ED, IEDGE)     :     del2=r1     qmax=rmax2     qmin=rmin2     q =r2     kq =kq1_pl     ll =i2     include 'lim'     :     :   END IF 100 CONTINUE </pre>
<p>【参考】 <i>Include file</i> “lim” の内容</p> <pre> plus = (1.+sign(1.DO,del2))/2. delmax=(qmax-q)*sign(1.DO,del2) psip = (delmax**2+eps2+2.*del2*delmax)/(delmax**2+2.*del2**2+delmax*del2+eps) delma =-(qmin-q)*sign(1.DO,del2) psim = (delma **2+eps2+2.*del2*delma )/(delma **2+2.*del2**2+delma *del2+eps) psi = psip*plus+psim*(1.-plus) c pole(kq,ll)= min (pole(kq,ll),psi) </pre>	

### A.3 LU-SGS 法ソルバ部分

ここでは、LU-SGS 法ソルバ部分に関して、スカラ版で適用した変更内容について以下に示す。

表 A.3.1 サブルーチン “LUSGS” の変更内容

File 名	lut2.f
Subroutine 名	LUSGS
変 更 内 容 の 説 明	
<p>ハイパー面を考慮した節点番号の付け替えのため、オリジナルの節点番号で保存されている配列から新しい節点番号で保存される作業用配列にコピーする処理を追加した。          (「3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え」参照)</p>	
変 更 内 容 リ ス ト	
<pre> if(iturb.eq.2)call diagonal_t_sa if(iturb.eq.1)call diagonal_t if(iturb.ne.1.and.iturb.ne.2)call diagonal  DO 11=1,NPF   I          =NDINV(11)   WPOLE(NR1_PL,11) =POLE(IR1_PL,1)   WPOLE(NU1_PL,11) =POLE(IU1_PL,1)   WPOLE(NV1_PL,11) =POLE(IV1_PL,1)   WPOLE(NW1_PL,11) =POLE(IW1_PL,1)   WPOLE(NP1_PL,11) =POLE(IP1_PL,1)   WPOLE(NTURB,11) =TURB(1)   WPOLE(NDIAG,11) =POLE(IDIAG,1)   WDS_ND(NRO_ND,11)=DS_ND(IRO_ND,1)   WDS_ND(NUO_ND,11)=DS_ND(IUO_ND,1)   WDS_ND(NVO_ND,11)=DS_ND(IVO_ND,1)   WDS_ND(NWO_ND,11)=DS_ND(IWO_ND,1)   WDS_ND(NPO_ND,11)=DS_ND(IPO_ND,1)   WDS_ND(NRT_ND,11)=RT(1)   WDS_ND(NVL_PL,11)=POLE(IVL_PL,1)   WDS_ND(NSF_PL,11)=SURF(1)   IWDS_ND(11)      =IDS_ND(I16_ND,1) END DO           </pre>	
	<p>作業用配列にコピーする処理を追加</p>

表 A.3.2 サブルーチン “DIAGONAL” の変更内容

File 名	lut2.f
Subroutine 名	DIAGONAL
変 更 内 容 の 説 明	
色分けを削除し, 配列の定義・参照の局所化を図った. (「3.1.2 色分けの削除」参照)	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> DO 10 ICOLOR=1, MCL_ED   jmin0=itc_ed2(1, icolor)   jmax =itc_ed2(2, icolor)   jminq=jmin0+1 *VOCL LOOP, NOVREC   DO 1100 J=jminq, JMAX     IEDGE=itc_ed1(J)     I2=IDS_ED(IN1_ED, IEDGE)     I3=IDS_ED(IN2_ED, IEDGE)     :     pole(idiag, I2)=pole(idiag, I2)     &amp;      +(0.5*abs(zf2)+aa2)*capa+unu     :     pole(idiag, I3)=pole(idiag, I3)     &amp;      +(0.5*abs(zf3)+aa3)*capa+unu   1100 CONTINUE   10 CONTINUE </pre>	<pre> DO 10 IEDGE=1, N_ED_F   IF (IDS_ED(ICL_ED, IEDGE).GE.1) THEN     I2=IDS_ED(IN1_ED, IEDGE)     I3=IDS_ED(IN2_ED, IEDGE)     :     pole(idiag, I2)=pole(idiag, I2)     &amp;      +(0.5*abs(zf2)+aa2)*capa+unu     :     pole(idiag, I3)=pole(idiag, I3)     &amp;      +(0.5*abs(zf3)+aa3)*capa+unu   END IF   10 CONTINUE </pre>

表 A.3.3 サブルーチン “DIAGONALT” の変更内容

File 名	lut2.f
Subroutine 名	DIAGONALT
変 更 内 容 の 説 明	
色分けを削除し, 配列の定義・参照の局所化を図った. (「3.1.2 色分けの削除」参照) 変更は行ったがテストは行っていないので注意されたい.	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.2 サブルーチン “DIAGONAL” の変更内容」に同じ)	

表 A.3.4 サブルーチン “DIAGONALT\_sa” の変更内容

File 名	lut2.f
Subroutine 名	DIAGONALT_sa
変 更 内 容 の 説 明	
色分けを削除し, 配列の定義・参照の局所化を図った. (「3.1.2 色分けの削除」参照) 変更は行ったがテストは行っていないので注意されたい.	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.2 サブルーチン “DIAGONAL” の変更内容」に同じ)	

表 A.3.5 サブルーチン “SIGMALEFT” の変更内容

File 名	lut2.f
Subroutine 名	SIGMALEFT
変更内容の説明	
<p>ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。            (「3.1.1 LU-SGS法における節点番号及び辺番号の付け替え」参照)            色分けを削除し、配列の定義・参照の局所化を図った。(「3.1.2 色分けの削除」参照)</p>	
変更内容リスト	
変更前	変更後
<pre> if(ilevel.eq.1) then   jmin=1 else   if(mcl_lu(ilevel-1).le.0) return   jmin=jsigma(mcl_lu(ilevel-1),ilevel-1)+1 end if DO 10 ICOLOR=1,MCL_lu(ilevel)   jmax=jsigma(icolor,ilevel) *VOCL LOOP,NOVREC   DO 1100 J=jmin,JMAX     IEDGE=ITMPI(J)     I2=IDS_ED(IN1_ED,IEDGE)     I3=IDS_ED(IN2_ED,IEDGE)     :     itp=markedg(iedge)     if(itp.lt.0) then       ii=i2       i2=i3       i3=ii       sx=-sx       sy=-sy       sz=-sz     end if     R =DS_ND(IRO_ND,I2)     :     P =DS_ND(IPO_ND,I2)     RTT=rt(i2)     deltr =pole(idlt1,i2)     :     deltrt=pole(idlt6,i2)     :     unu=fmu/r/re/pole(ivl_pl,ia)*surf(ia)     :     pole(mq1_pl,ia)=pole(mq1_pl,ia)     &amp; -delf1-specr*deltr     :     pole(mq6_pl,ia)=pole(mq6_pl,ia)     -delf6-specr*deltrt 1100 continue     jmin=jmax+1 10 continue </pre>	<pre> if(ilevel.ne.1.and. &amp; mcl_lu(ilevel-1).le.0) return jmin=jsigma(ilevel-1)+1 jmax=jsigma(ilevel)  DO 1100 J=jmin,JMAX   IEDGE=ITMPI(J)   I12 =IDS_TBL(1,IEDGE)   I1A =IDS_TBL(2,IEDGE)   :   itp =markedg(iedge)   if(itp.lt.0) then      sx=-sx     sy=-sy     sz=-sz   end if   R =WDS_ND(NRO_ND,I12)   :   P =WDS_ND(NPO_ND,I12)   RTT=WDS_ND(NRT_ND,I12)   deltr =deltQ(idlt1,i12)   :   deltrt=deltQ(idlt6,i12)   :   unu =fmu/r/re/wds_nd(nvl_pl,i1a)   &amp; *wds_nd(nsf_pl,i1a)   :   deltrQ(mq1_pl,i1a)=deltQ(mq1_pl,i1a)   &amp; -delf1-specr*deltr   :   deltrQ(mq6_pl,i1a)=deltQ(mq6_pl,i1a)   &amp; -delf6-specr*deltrt 1100 continue </pre>



表 A.3.6 サブルーチン “SIGMALEFTT” の変更内容

File 名	lut2. f
Subroutine 名	SIGMALEFTT
変 更 内 容 の 説 明	
<p>ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。          (「3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え」参照)          色分けを削除し、配列の定義・参照の局所化を図った。(「3.1.2 色分けの削除」参照)          変更は行ったがテストは行っていないので注意されたい。</p>	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.5 サブルーチン “SIGMALEFT” の変更内容」に同じ)	

表 A.3.7 サブルーチン “SIGMALEFTT\_sa” の変更内容

File 名	lut2. f
Subroutine 名	SIGMALEFTT_sa
変 更 内 容 の 説 明	
<p>ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。          (「3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え」参照)          色分けを削除し、配列の定義・参照の局所化を図った。(「3.1.2 色分けの削除」参照)          変更は行ったがテストは行っていないので注意されたい。</p>	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.5 サブルーチン “SIGMALEFT” の変更内容」に同じ)	

表 A.3.8 サブルーチン “SWEEPL” の変更内容

File 名	lut2. f
Subroutine 名	SWEEPL
変 更 内 容 の 説 明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。 (「3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え」参照)	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> j=0 do i=1, n_p_f   itp=marknd1(i)   itp1=ids_nd(i16_nd, i)   if(itp.eq. iplane.and. itp1.lt. 10) then     j      =j+1     itmp(j)=i   end if end do jmax=j *VOCL LOOP, NOVREC do j=1, jmax   i=itmp(j)    dia=1./pole(idiag, i)   pole(idlt1, i)=(-0.5*pole(mq1_pl, i) &amp;                +pole(ir1_pl, i))*dia   :   pole(idlt6, i)=(-0.5*pole(mq6_pl, i) &amp;                +turb(i))*dia end do </pre>	<pre> do i=ndsigma(iplane-1)+1, ndsigma(iplane)    itp=iwds_nd(i)   if(itp.lt. 10) then     dia      =1./wpole(ndiag, i)     deltQ(idlt1, i)=(-0.5*deltQ(mq1_pl, i) &amp;                  +wpole(nr1_pl, i))*dia     :     deltQ(idlt6, i)=(-0.5*deltQ(mq6_pl, i) &amp;                  +wpole(nturb, i))*dia   end if end do </pre>

表 A.3.9 サブルーチン “SIGMARIGHT” の変更内容

File 名	lut2.f
Subroutine 名	SIGMARIGHT
変更内容の説明	
<p>ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。                  (「3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え」参照)                  色分けを削除し、配列の定義・参照の局所化を図った。(「3.1.2 色分けの削除」参照)</p>	
変更内容リスト	
変更前	変更後
<pre> if(ilevel.eq.1) then   jmin=1 else   if(mcl_lu1(ilevel-1).le.0) return   jmin=jsigma1(mcl_lu1(ilevel-1),ilevel-1)+1 end if DO 10 ICOLOR=1,MCL_lu1(ilevel)   jmax=jsigma1(icolor,ilevel) *VOCL LOOP,NOVREC   DO 1100 J=jmin,JMAX     IEDGE=ITMPR(J)     I2=IDS_ED(IN1_ED,IEEDGE)     I3=IDS_ED(IN2_ED,IEEDGE)     :     itp=marked1(iedge)     if(itp.gt.0) then       ii=i2       i2=i3       i3=ii       sx=-sx       sy=-sy       sz=-sz     end if     R =DS_ND(IRO_ND,I2)     :     P =DS_ND(IPO_ND,I2)     RTT=rt(i2)     deltr =pole(kdlt1,i2)     :     deltrt=pole(kdlt6,i2)     :     unu=fmu/r/re/pole(ivl_pl,ia)*surf(ia)     :     pole(mq1_pl,ia)=pole(mq1_pl,ia)     &amp; -delf1-specr*deltr     :     pole(mq6_pl,ia)=pole(mq6_pl,ia)     &amp; -delf6-specr*deltrt 1100 continue     jmin=jmax+1 10 continue                 </pre>	<pre> if(ilevel.ne.1.and. &amp; mcl_lu1(ilevel-1).le.0) return jmin=jsigma1(ilevel-1)+1 jmax=jsigma1(ilevel) DO 1100 J=jmin,JMAX   IEDGE=ITMPR(J)   I12 =IDS_TBL(2,IEEDGE)   I1A =IDS_TBL(1,IEEDGE)   :   itp =markedg1(iedge)   if(itp.gt.0) then     sx=-sx     sy=-sy     sz=-sz   end if   R =WDS_ND(NRO_ND,I12)   :   P =WDS_ND(NPO_ND,I12)   RTT=WDS_ND(NRT_ND,I12)   deltr =deltQ(idlt1,ii2)   :   deltrt=deltQ(idlt6,ii2)   :   unu =fmu/r/re/wds_nd(nvl_pl,ia)   &amp; *wds_nd(nsf_pl,ia)   :   deltrtQ(mq1_pl,ia)=deltQ(mq1_pl,ia)   &amp; -delf1-specr*deltr   :   deltrtQ(mq6_pl,ia)=deltQ(mq6_pl,ia)   &amp; -delf6-specr*deltrt 1100 continue                 </pre>

表 A.3.10 サブルーチン “SIGMARIGHTT” の変更内容

File 名	lut2.f
Subroutine 名	SIGMARIGHTT
変 更 内 容 の 説 明	
<p>ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。          (「3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え」参照)          色分けを削除し、配列の定義・参照の局所化を図った。(「3.1.2 色分けの削除」参照)          変更は行ったがテストは行っていないので注意されたい。</p>	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.9 サブルーチン “SIGMARIGHT” の変更内容」に同じ)	

表 A.3.11 サブルーチン “SIGMARIGHTT\_sa” の変更内容

File 名	lut2.f
Subroutine 名	SIGMARIGHTT_sa
変 更 内 容 の 説 明	
<p>ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。          (「3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え」参照)          色分けを削除し、配列の定義・参照の局所化を図った。(「3.1.2 色分けの削除」参照)          変更は行ったがテストは行っていないので注意されたい。</p>	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.9 サブルーチン “SIGMARIGHT” の変更内容」に同じ)	

表 A.3.12 サブルーチン “SWEEP” の変更内容

File 名	lut2.f
Subroutine 名	SWEEP
変 更 内 容 の 説 明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。 (「3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え」参照)	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> j=0 do i=1, n_p_f   itp=marknd1(i)   if(itp.eq. iplane.and. itp1.lt.10) then     j=j+1     itmp(j)=i   end if end do jmax=j *VOCL LOOP, NOVREC do j=1, jmax   i =itmp(j)    dia=1./pole(idiag, i)   pole(kdlt1, i)=pole(idlt1, i) &amp;      -.5*pole(mq1_pl, i)*dia   :   pole(kdlt6, i)=pole(idlt6, i) &amp;      -.5*pole(mq6_pl, i)*dia  end do </pre>	<pre> do i=ndsigma(iplane-1)+1, ndsigma(iplane)    itp=iwds_nd(i)   if(itp.lt.10) then     dia      =1./wpole(ndiag, i)     delTQ(idlt1, i)=(-0.5*delTQ(mq1_pl, i) &amp;      +wpole(nr1_pl, i))*dia     :     delTQ(idlt6, i)=(-0.5*delTQ(mq6_pl, i) &amp;      +wpole(nturb, i))*dia   end if end do </pre>

表 A.3.13 サブルーチン“UPDAEQ”の変更内容

File 名	lut2.f
Subroutine 名	UPDATEQ
変 更 内 容 の 説 明	
<p>ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。          ([3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え] 参照)</p> <p>このサブルーチンにおいて、ハイパー面を考慮した新しい節点の番号付けによる配列からオリジナルの配列に書き戻しが行われている。従って、「変更前」の DO ループは、連続アクセスであるのに対して、「変更後」ではインデックス参照となっており、若干のオーバーヘッドが発生する。</p>	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> do i=1,n_p_f   if(ids_nd(imr_nd,i).eq.dltd) go to 10    dtt=dtloc(i)   v =pole(ivl_pl,i)/dtt   pole(ir1_pl,i)=pole(kdlt1,i)*v   pole(iu1_pl,i)=pole(kdlt2,i)*v   pole(iv1_pl,i)=pole(kdlt3,i)*v   pole(iw1_pl,i)=pole(kdlt4,i)*v   pole(ip1_pl,i)=pole(kdlt5,i)*v   turb(i)      =pole(kdlt6,i)*v  10  continue end do </pre>	<pre> do i=1,n_p_f   if(ids_nd(imr_nd,i).eq.dltd) go to 10   ii      =nddir(i)   dtt     =dtloc(i)   v       =pole(ivl_pl,i)/dtt   pole(ir1_pl,i)=deltQ(kdlt1,ii)*v   pole(iu1_pl,i)=deltQ(kdlt2,ii)*v   pole(iv1_pl,i)=deltQ(kdlt3,ii)*v   pole(iw1_pl,i)=deltQ(kdlt4,ii)*v   pole(ip1_pl,i)=deltQ(kdlt5,ii)*v   turb(i)  =deltQ(kdlt6,ii)*v  10  continue end do </pre>

表 A.3.14 サブルーチン“ZEROSIGMA”の変更内容

File 名	lut2.f
Subroutine 名	ZEROSIGMA
変 更 内 容 の 説 明	
<p>ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。          ([3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え] 参照)</p>	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> do i=1,npf   pole(mq1_pl,i)=0.   pole(mq2_pl,i)=0.   pole(mq3_pl,i)=0.   pole(mq4_pl,i)=0.   pole(mq5_pl,i)=0.   pole(mq6_pl,i)=0. end do </pre>	<pre> do i=1,npf   delTQ(mq1_pl,i)=0.   delTQ(mq2_pl,i)=0.   delTQ(mq3_pl,i)=0.   delTQ(mq4_pl,i)=0.   delTQ(mq5_pl,i)=0.   delTQ(mq6_pl,i)=0. end do </pre>

表 A.3.15 サブルーチン “ZERODELT” の変更内容

File 名	lut2. f	
Subroutine 名	ZERODELT	
変 更 内 容 の 説 明		
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。 (「3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え」参照)		
変 更 内 容 リ ス ト		
変 更 前	変 更 後	
<pre> do i=1, npf    if(ids_nd(imr_nd, i).eq.dltd) goto 10   pole(idlt1, i)=0.   pole(idlt2, i)=0.   pole(idlt3, i)=0.   pole(idlt4, i)=0.   pole(idlt5, i)=0.   pole(idlt6, i)=0.    pole(kdlt1, i)=0.   pole(kdlt2, i)=0.   pole(kdlt3, i)=0.   pole(kdlt4, i)=0.   pole(kdlt5, i)=0.   pole(kdlt6, i)=0.    dtt=dtloc(i)   pole(idiag, i)=pole(ivl_pl, i)/dtt 10  continue end do </pre>	<pre> do i=1, npf   ii=ndinv(i)   if(ids_nd(imr_nd, ii).ne.dltd) then     deltQ(idlt1, i)=0.     deltQ(idlt2, i)=0.     deltQ(idlt3, i)=0.     deltQ(idlt4, i)=0.     deltQ(idlt5, i)=0.     deltQ(idlt6, i)=0.      deltQ(kdlt1, i)=0.     deltQ(kdlt2, i)=0.     deltQ(kdlt3, i)=0.     deltQ(kdlt4, i)=0.     deltQ(kdlt5, i)=0.     deltQ(kdlt6, i)=0.   end if end do do i=1, npf   if(ids_nd(imr_nd, i).ne.dltd) then     dtt=dtloc(i)     pole(idiag, i)=pole(ivl_pl, i)/dtt   end if end do </pre>	

表 A.3.16 サブルーチン “SIGMAPREPARE” の変更内容 (1/4)

File 名	lut2. f
Subroutine 名	SIGMAPREPARE
変 更 内 容 の 説 明	
色分けを削除し、配列の定義・参照の局所化を図るため、インデックステーブル “ITMPI” 及び “ITMPPr” の設定方法を変更した。(「3.1.2 色分けの削除」参照)	
変 更 内 容 リ ス ト ( 続 く )	
変 更 前	変 更 後
<pre> if(numplane.gt.2000) then   write(*,*) "STOP!! sigmaprep1"   call MPI_abort(MPI_COMM_WORLD, 1)   stop end if J=0  do ilevel=1,numplane-1   DO 10 ICOLOR=1,MCL_lu(ilevel)     if(icolor.gt.64) then       write(*,*) "STOP!! sigmaprep1"       call MPI_abort(MPI_COMM_WORLD, 1)       stop     end if     DO 333 IEDGE=1,N_ED_F       ICOL=markedgcol(IEDGE)       itp =abs(markedg(iedge))       IF(ICOL.EQ.ICOLOR.and. &amp;      itp.eq.ilevel ) THEN         J      =J+1         ITMPI(J)=IEDGE       END IF     333    CONTINUE       jsigma(icolor,ilevel)=j     10    continue   end do   J=0    do ilevel=1,numplane-1     DO 11 ICOLOR=1,MCL_lu1(ilevel)       DO 334 IEDGE=1,N_ED_F         ICOL=markedgcol1(IEDGE)         itp =abs(markedg1(iedge))         IF(ICOL.EQ.ICOLOR.and. &amp;        itp.eq.ilevel ) THEN           J      =J+1           ITMPPr(J)=IEDGE         END IF       334    CONTINUE         jsigma1(icolor,ilevel)=j       11    continue     end do </pre>	<pre> if(numplane.gt.imx_plane) then   write(*,*) "STOP!! sigmaprep1"   call MPI_abort(MPI_COMM_WORLD, 1)   stop end if jsigma(0)=0 J      =0 do ilevel=1,numplane-1    DO IEDGE=1,N_ED_F      itp=abs(markedg(iedge))     IF(itp.eq.ilevel) THEN       J      =J+1       ITMPI(J)=IEDGE     END IF   END DO    jsigma(ilevel)=j end do  jsigma1(0)=0 J      =0 do ilevel=1,numplane-1    DO IEDGE=1,N_ED_F      itp=abs(markedg1(iedge))     IF(itp.eq.ilevel) THEN       J      =J+1       ITMPPr(J)=IEDGE     END IF   END DO    jsigma1(ilevel)=j end do </pre>



表 A.3.16 サブルーチン “SIGMAPREPARE” の変更内容 (2/4)

File 名	lut2. f		
Subroutine 名	SIGMAPREPARE		
変 更 内 容 の 説 明			
<p>ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更するため、以下のインデックステーブルを設定する処理を追加した。          (「3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え」参照)</p>			
変数名	型	サイズ	説 明
NDDIR	I4	(IMX_ND)	オリジナル-新節点番号対応テーブル $nd^{new} = NDDIR(nd^{old})$ $nd^{new}$ : ハイパー面を考慮した新節点番号 $nd^{old}$ : オリジナルの節点番号
NDINV	I4	(IMX_ND)	オリジナル-新節点番号逆対応テーブル $nd^{old} = NDINV(nd^{new})$ $nd^{old}$ : オリジナルの節点番号 $nd^{new}$ : ハイパー面を考慮した新節点番号
NDSIGMA	I4	(0:IMX_PLANE)	ハイパー面 $N_p$ が含む節点番号の最大値 $nd_{max}$ $nd_{max} = NDSIGMA(N_p)$
IDS_TBL	I4	(2,IMX_ED)	辺-節点対応テーブル $nd^{new} = IDS\_TBL(ed)$ : 辺 $ed$ が含む新しい節点の番号 $nd^{new}$
変 更 内 容 リ ス ト ( 続 く )			
<pre> nds sigma(0)=0 j      =0 do iplane=1,numplane   do i=1,n_p_f     itp1=marknd1(i)     if(itp1.eq. iplane) then       j      =j+1       nmdir(i)=j       ndinv(j)=i     end if   end do   ndsigma(iplane)=j end do do iplane=1,numplane   do i=n_p_f+1,npf     itp1=marknd1(i)     if(itp1.eq. iplane) then       j      =j+1       nmdir(i)=j       ndinv(j)=i     end if   end do end do DO IEDGE=1, N_ED_F   ITP=MARKEDG1(IEEDGE)   I2 =IDS_ED(IN1_ED, IEDGE)   I3 =IDS_ED(IN2_ED, IEDGE)   I12=NDDIR(I2)   I13=NDDIR(I3)   IF(ITP. GE. 0) THEN     IDS_TBL(1, IEDGE)=I12     IDS_TBL(2, IEDGE)=I13   ELSE     IDS_TBL(1, IEDGE)=I13     IDS_TBL(2, IEDGE)=I12   END IF END DO END DO </pre>			

表 A.3.16 サブルーチン “SIGMAPREPARE” の変更内容 (3/4)

File 名	lut2.f
Subroutine 名	SIGMAPREPARE
変 更 内 容 の 説 明	
<p>サブルーチン “SIGMALEFT” 及び “SIGMARIGHT” において節点番号が連続に処理されるようにハイパー面に含まれる辺の番号を保持するインデックステーブル “ITMPI” 及び “ITMPr” をその辺が含むハイパー面内の節点番号が昇順になるようにソートする処理を追加した。(「3.1.1 LU-SGS 法における節点番号及び辺番号の付け替え」参照)</p> <p>この処理は、論理変数 “SORT” に .false. を与えることでスキップすることが可能である。 .true. を与えてソートを行った場合が、表 5.15 の “sorted” となり、 .false. を与えてソートをスキップした場合が “not sorted” となる。“as is” については、表 A.3.16 (4/4) を参照されたい。</p>	
変 更 内 容 リ ス ト ( 続 く )	
<pre> LOGICAL SORT C***** sort by ascending order of node number SORT = .true. c for forwording   IF( SORT ) THEN     DO ILEV=1, NUMPLANE1       DO I =JSIGMA(ILEV-1)+1, JSIGMA(ILEV)-1         DO J =I+1, JSIGMA(ILEV)           IEDGE1=ITMPL(I)           IIA1 =IDS_TBL(2, IEDGE1)           IEDGE2=ITMPL(J)           IIA2 =IDS_TBL(2, IEDGE2)           IF( IIA2.LT. IIA1 ) THEN             IITMP = ITMPL(I)             ITMPL(I) = ITMPL(J)             ITMPL(J) = IITMP           END IF         END DO       END DO     END DO   END IF c for backwording   IF( SORT ) THEN     DO IPLANE=1, NUMPLANE1       DO I =JSIGMA1(IPLANE-1)+1, JSIGMA1(IPLANE)-1         DO J =I+1, JSIGMA1(IPLANE)           IEDGE1=ITMPR(I)           IIA1 =IDS_TBL(1, IEDGE1)           IEDGE2=ITMPR(J)           IIA2 =IDS_TBL(1, IEDGE2)           IF( IIA2.LT. IIA1 ) THEN             IITMP = ITMPR(I)             ITMPR(I) = ITMPR(J)             ITMPR(J) = IITMP           END IF         END DO       END DO     END DO   END IF </pre>	

表 A.3.16 サブルーチン “SIGMAPREPARE” の変更内容 (4 / 4)

File 名	lut2.f
Subroutine 名	SIGMAPREPARE
変 更 内 容 の 説 明	
<p>サブルーチン “SIGMALEFT” 及び “SIGMARIGHT” においてオリジナルと同じ順番で節点が処理されるようにハイパー面に含まれる辺の番号を保持するインデックステーブル “ITMPI” 及び “ITMPr” の並べ替えを行う処理を追加した。この処理は、ソートの効果を確認するために用意したものであり、最初の if 文に .true. を与えることにより、表5.15の “as is” の測定が再現される。通常は、.false. を与えておけばよい。</p>	
変 更 内 容 リ ス ト ( 続 き )	
<pre> !      set .true., if same order of original LU-SGS calculation       if( .false. ) then         J=0         do ilevel=1,numplane-1           DO ICOLOR=1,MCL_lu(ilevel)           DO IEDGE =1,N_ED_F             ICOL=markedgcol(IEEDGE)             itp =abs(markedg(iedge))             IF(ICOL.EQ.ICOLOR.and.itp.eq.ilevel) THEN               J      =J+1               ITMPI(J)=IEEDGE             END IF           END DO         END DO       end do  c       J=0       do ilevel=1,numplane-1         DO ICOLOR=1,MCL_lu1(ilevel)         DO IEDGE =1,N_ED_F           ICOL=markedgcol1(IEEDGE)           itp =abs(markedg1(iedge))           IF(ICOL.EQ.ICOLOR.and.itp.eq.ilevel) THEN             J      =J+1             ITMPr(J)=IEEDGE           END IF         END DO       END DO     end do   end if </pre>	

表 A.3.17 サブルーチン“LUSGSREAD”の変更内容

File 名	lut2.f	
Subroutine 名	LUSGSREAD	
変更内容の説明		
ハイパー面を考慮した節点番号の付け替えに伴い追加・変更されたインデックステーブルの入力処理を追加・変更した。		
変更内容リスト		
変更前	変更後	
<pre> REWIND MT READ(MT) IDUMMY, IDUMMY, NUMPLANE READ(MT) (MARKND1 (I) , I=1, NPF) READ(MT) (MARKEDG (I) , I=1, N_ED_F) READ(MT) (MARKEDG1 (I) , I=1, N_ED_F) READ(MT) (MCL_LU (I) , I=1, NUMPLANE-1) READ(MT) (MCL_LU1 (I) , I=1, NUMPLANE-1) READ(MT) ((JSIGMA (I, J), I=1, MCL_LU(J)), &amp; J=1, NUMPLANE-1) READ(MT) ((JSIGMA1 (I, J), I=1, MCL_LU1 (J)), &amp; J=1, NUMPLANE-1) ITMPLMAX = JSIGMA (MCL_LU (NUMPLANE-1), &amp; NUMPLANE-1) &amp; ITMPRMAX = JSIGMA1 (MCL_LU1 (NUMPLANE-1), &amp; NUMPLANE-1) READ(MT) (ITMPL (I), I=1, ITMPLMAX) READ(MT) (ITMPR (I), I=1, ITMPRMAX) </pre>	<pre> REWIND MT READ(MT) IDUMMY, IDUMMY, NUMPLANE READ(MT) (MARKND1 (I), I=1, NPF) READ(MT) (MARKEDG (I), I=1, N_ED_F) READ(MT) (MARKEDG1 (I), I=1, N_ED_F) READ(MT) (MCL_LU (I), I=1, NUMPLANE-1) READ(MT) (MCL_LU1 (I), I=1, NUMPLANE-1) READ(MT) (JSIGMA (I), I=0, NUMPLANE-1) READ(MT) (JSIGMA1 (I), I=0, NUMPLANE-1) ITMPLMAX = JSIGMA (NUMPLANE-1) ITMPRMAX = JSIGMA1 (NUMPLANE-1) READ(MT) (ITMPL (I), I=1, ITMPLMAX) READ(MT) (ITMPR (I), I=1, ITMPRMAX) READ(MT) (NDSIGMA (I), I=0, NUMPLANE) READ(MT) (NDDIR (I), I=1, NPF) READ(MT) (NDINV (I), I=1, NPF) READ(MT) (IDS_TBL (1, I), I=1, N_ED_F) READ(MT) (IDS_TBL (2, I), I=1, N_ED_F) </pre>	
REWIND MT	REWIND MT	

表 A.3.18 サブルーチン “LUSGSWRITE” の変更内容

File 名	lut2.f
Subroutine 名	LUSGSWRIGHT
変 更 内 容 の 説 明	
ハイパー面を考慮した節点番号の付け替えに伴い追加・変更されたインデックステーブルの出力処理を追加・変更した。	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> REWIND MT WRITE (MT) NPF, N_ED_F, NUMPLANE WRITE (MT) (MARKND1 (1) , I=1, NPF) WRITE (MT) (MARKEDG (1) , I=1, N_ED_F) WRITE (MT) (MARKEDG1 (1) , I=1, N_ED_F) WRITE (MT) (MCL_LU (1) , I=1, NUMPLANE-1) WRITE (MT) (MCL_LU1 (1) , I=1, NUMPLANE-1) WRITE (MT) ((JSIGMA (I, J) , I=1, MCL_LU(J)) , &amp; J=1, NUMPLANE-1) WRITE (MT) ((JSIGMA1(I, J) , I=1, MCL_LU1 (J)) , &amp; J=1, NUMPLANE-1) WRITE (MT) (ITMPL (1) , I=1, ITMPLMAX) WRITE (MT) (ITMPR (1) , I=1, ITMPRMAX) </pre>	<pre> REWIND MT WRITE (MT) NPF, N_ED_F, NUMPLANE WRITE (MT) (MARKND1 (1) , I=1, NPF) WRITE (MT) (MARKEDG (1) , I=1, N_ED_F) WRITE (MT) (MARKEDG1 (1) , I=1, N_ED_F) WRITE (MT) (MCL_LU (1) , I=1, NUMPLANE-1) WRITE (MT) (MCL_LU1 (1) , I=1, NUMPLANE-1) WRITE (MT) (JSIGMA (1) , I=0, NUMPLANE-1) WRITE (MT) (JSIGMA1 (1) , I=0, NUMPLANE-1) WRITE (MT) (ITMPL (1) , I=1, ITMPLMAX) WRITE (MT) (ITMPR (1) , I=1, ITMPRMAX) WRITE (MT) (NDSIGMA (1) , I=0, NUMPLANE) WRITE (MT) (NDDIR (1) , I=1, NPF) WRITE (MT) (NDINV (1) , I=1, NPF) WRITE (MT) (IDS_TBL(1, 1) , I=1, N_ED_F) WRITE (MT) (IDS_TBL(2, 1) , I=1, N_ED_F) REWIND MT </pre>
REWIND MT	REWIND MT

表 A.3.19 サブルーチン “SETS\_SUB\_MPICALL” の変更内容

File 名	mpi. f
Subroutine 名	SETS_SUB_MPICALL
変 更 内 容 の 説 明	
<p>LU-SGS 法において、ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更したため、前進スイープ (iflag=5) 及び後退スイープ (iflag=6) の後に行うプロセス間通信についても、その対象とする配列を変更した。iflag=7 は、サブルーチン “SIGMALEFT” 及び “SIGMARIGT” における計算結果に関するプロセス間通信であり、これも同時に修正したが、現在の JTAS においてこのプロセス間通信が実行されることはない。</p>	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> i = 1 do n=1, nd_send   jmin =      dom_s(3, n)   jmax = jmin + dom_s(2, n) - 1   do j=jmin, jmax     l = nod_s(2, j)     if(iflag.eq.0) then       :     else if(iflag.eq.5) then        do m=1, 6         sbuf(i) = pole(ktl(m), l)         i = i + 1       enddo     else if(iflag.eq.6) then        do m=1, 6         sbuf(i) = pole(kul(m), l)         i = i + 1       enddo     else if(iflag.eq.7) then        do m=1, 6         sbuf(i) = pole(kvl(m), l)         i = i + 1       enddo     end if   end do end do </pre>	<pre> i = 1 do n=1, nd_send   jmin =      dom_s(3, n)   jmax = jmin + dom_s(2, n) - 1   do j=jmin, jmax     l = nod_s(2, j)     if(iflag.eq.0) then       :     else if(iflag.eq.5) then       ll = nmdir(l)       do m=1, 6         sbuf(i) = deltQ(ktl(m), ll)         i = i + 1       enddo     else if(iflag.eq.6) then       ll = nmdir(l)       do m=1, 6         sbuf(i) = deltQ(kul(m), ll)         i = i + 1       enddo     else if(iflag.eq.7) then       ll = nmdir(l)       do m=1, 6         sbuf(i) = deltQ(kvl(m), ll)         i = i + 1       enddo     end if   end do end do </pre>

表 A.3.20 サブルーチン “SETS\_SUB\_MPICALL” の変更内容

File 名	mpi.f
Subroutine 名	SETS_SUB_MPICALL
変更内容の説明	
<p>LU-SGS 法において、ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更したため、前進スイープ (ifalg=5) 及び後退スイープ (iflag=6) の後に行うプロセス間通信についても、その対象とする配列を変更した。iflag=7 は、サブルーチン “SIGMALEFT” 及び “SIGMARIGT” における計算結果に関するプロセス間通信であり、これも同時に修正したが、現在の JTAS においてこのプロセス間通信が実行されることはない。</p>	
変更内容リスト	
変更前	変更後
<pre> i = 1 do n=1, nd_send   jmin =      dom_s(3, n)   jmax = jmin + dom_s(2, n) - 1   do j=jmin, jmax     l = nod_s(2, j)     if(iflag.eq.0) then       :     else if(iflag.eq.5) then        do m=1, 6         sbuf(i) = pole(ktl(m), l)         i = i + 1       enddo     else if(iflag.eq.6) then        do m=1, 6         sbuf(i) = pole(kul(m), l)         i = i + 1       enddo     else if(iflag.eq.7) then        do m=1, 6         sbuf(i) = pole(kvl(m), l)         i = i + 1       enddo     end if   end do end do </pre>	<pre> i = 1 do n=1, nd_send   jmin =      dom_s(3, n)   jmax = jmin + dom_s(2, n) - 1   do j=jmin, jmax     l = nod_s(2, j)     if(iflag.eq.0) then       :     else if(iflag.eq.5) then       ll = nmdir(l)       do m=1, 6         sbuf(i) = deltQ(ktl(m), ll)         i = i + 1       enddo     else if(iflag.eq.6) then       ll = nmdir(l)       do m=1, 6         sbuf(i) = deltQ(kul(m), ll)         i = i + 1       enddo     else if(iflag.eq.7) then       ll = nmdir(l)       do m=1, 6         sbuf(i) = deltQ(kvl(m), ll)         i = i + 1       enddo     end if   end do end do </pre>

表 A.3.21 サブルーチン “NMAIN” の変更内容

File 名	necns2.f
Subroutine 名	NMAIN
変 更 内 容 の 説 明	
一ステップ当たりの変化分 $\Delta Q_i$ を保存するための配列 “deltQ” を新設したため、この配列のゼロクリア処理を追加した。	
変 更 内 容 リ ス ト	
<pre> call mirror do i=1, npf do j=1, isz_pl     pole(j, i)=0.0 enddo enddo do j=1, npf do i=1, isz_deltQ     deltQ(i, j)=0.0 enddo enddo call prepare </pre>	<p>} 配列“deltQ”のゼロクリア処理を追加</p>

表 A.3.22 インクルードファイル “DS3D” の変更内容

File 名	DS3D
Subroutine 名	<i>include file</i>
変 更 内 容 の 説 明	
辺-節点对応テーブル“IDS_TBL”の配列宣言を追加した。 $nd^{new} = IDS\_TBL(, ed)$ : 辺 $ed$ が含む新しい節点の番号 $nd^{new}$	
変 更 内 容 リ ス ト	
「表 A.1.9 インクルードファイル“DS3D”の変更内容」参照	



表 A.3.23 インクルードファイル “LU” の変更内容 (続く)

File 名	LU
Subroutine 名	<i>include file</i>
変 更 内 容 の 説 明	
ハイパー面を考慮した節点番号を付け替えて保存するための配列の宣言を追加した.	
配 列 名	説 明
ndsigma(0:imax_plane)	ハイパー面 $N_p$ が含む節点番号の最大値 $nd_{\max}$ $nd_{\max} = \text{NDSIGMA}(N_p)$
nmdir(imx_nd)	オリジナル-新節点番号対応テーブル $nd^{\text{new}} = \text{nmdir}(nd^{\text{old}})$ $nd^{\text{new}}$ : ハイパー面を考慮した新節点番号 $nd^{\text{old}}$ : オリジナルの節点番号
ndinv(imx_nd)	オリジナル-新節点番号逆対応テーブル $nd^{\text{old}} = \text{ndinv}(nd^{\text{new}})$ $nd^{\text{old}}$ : オリジナルの節点番号 $nd^{\text{new}}$ : ハイパー面を考慮した新節点番号
deltQ (idltm , i), (m =1,6)	前進スweep: $\Delta \mathbf{Q}_i^*$
deltQ (kdltm , i), (m =1,6)	退進スweep: $\Delta \mathbf{Q}_i^n$
deltQ (mqm, _pl, i), (m =1,6)	$\sum_{j \in L(i) \text{ or } j \in U(i)} \Delta S_{ij} \left( \Delta \mathbf{f}_{ij}^* - (\rho_A)_j \Delta \mathbf{Q}_j^* \right)$
wpole (nxx_pl, i), (xx=r1, u1, v1, w1, p1, turb)	HLLEW 法による数値流束ベクトル $\mathbf{F}(\mathbf{Q}_i) \cdot \mathbf{n}$
wpole (ndiag , i)	$\mathbf{D}_i$
wds_nd (nxx_nd, i), (xx=r0, u0, v0, w0, p0, rt)	$\Delta \mathbf{Q}_i^n$
iwds_nd (nxx_nd, i), (xx=r0, u0, v0, w0, p0, rt)	節点タイプ

表 A.3.23 インクルードファイル“LU”の変更内容(続き)

File 名	LU
Subroutine 名	include file
変 更 内 容 リ ス ト	
変 更 前	変 更 後
parameter (imx_plane=10000)	parameter (isz_deltQ= 18)
:	parameter (imx_plane=10000)
:	parameter (imx_wpl = 7)
parameter (idlt1 =krx_pl, idlt2 =kux_pl,	parameter (imx_wnd = 8)
+ idlt3 =kvx_pl, idlt4 =kwx_pl,	:
+ idlt5 =kpx_pl, idlt6 =krtx_pl,	parameter (idlt1 = 1, idlt2 = 2,
+ kdlt1 =kry_pl, kdlt2 =kuy_pl,	+ idlt3 = 3, idlt4 = 4,
+ kdlt3 =kvy_pl, kdlt4 =kwy_pl,	+ idlt5 = 5, idlt6 = 6,
+ kdlt5 =kpy_pl, kdlt6 =krty_pl,	+ kdlt1 =13, kdlt2 =14,
+ mq1_pl=krz_pl, mq2_pl=kuz_pl,	+ kdlt3 =15, kdlt4 =16,
+ mq3_pl=kvz_pl, mq4_pl=kwz_pl,	+ kdlt5 =17, kdlt6 =18,
+ mq5_pl=kpz_pl, mq6_pl=krtz_pl	+ mq1_pl= 7, mq2_pl= 8,
:	+ mq3_pl= 9, mq4_pl=10,
:	+ mq5_pl=11, mq6_pl=12,
:	:
common/sig/ jsigma (64, 2000),	parameter (nr1_pl= 1, nu1_pl= 2,
+ jsigma1(64, 2000),	+ nv1_pl= 3, nw1_pl= 4,
+ itmpl (imx_ed),	+ np1_pl= 5, nturb = 6, ndiag = 7)
+ itmpr (imx_ed)	parameter (nr0_nd= 1, nu0_nd= 2,
	+ nv0_nd= 3, nw0_nd= 4,
	+ np0_nd= 5, nrt_nd= 6,
	+ nvl_pl= 7, nsf_pl = 8)
	parameter (n16_nd = 1)
	common/ sig / jsigma (0:imx_plane),
	+ jsigma1(0:imx_plane),
	+ itmpl ( imx_ed),
	+ itmpr ( imx_ed),
	+ ndsigma(0:imx_plane),
	+ nddir ( imx_nd),
	+ ndinv ( imx_nd)
	common/deltQ/ deltQ (isz_deltQ, imx_nd)
	common/qwork/ wpole (imx_wpl , imx_nd),
	+ wds_nd (imx_wnd , imx_nd),
	+ iwds_nd( imx_nd)

**A.4 HLEW 法による数値流束ベクトル  $F(Q) \cdot n$  の計算**

ここでは、HLEW 法による数値流束ベクトル  $F(Q) \cdot n$  の計算に関して、スカラ版で適用した変更内容について以下に示す。

表 A.4 サブルーチン “SPVCOR” の変更内容

File 名	necns2.f		
Subroutine 名	SPVCOR		
変更内容の説明			
色分けを削除し、配列の定義・参照の局所化を図った。(「3.1.2 色分けの削除」参照)			
変更内容リスト			
変更前	変更後		
<pre> DO 10 ICOLOR=1, MCL_ED   jmin0=itc_ed2(1, icolor)   jmax =itc_ed2(2, icolor)   jminq=jmin0+1 *VOCL LOOP, NOVREC   DO 1100 J=jminq, JMAX     IEDGE=itc_ed1(J)     I2=IDS_ED(IN1_ED, IEDGE)     I3=IDS_ED(IN2_ED, IEDGE)     :     POLE(IR1_PL, I2)=POLE(IR1_PL, I2)-F1     :     POLE(IP1_PL, I2)=POLE(IP1_PL, I2)-F5+g5     :     POLE(IR1_PL, I3)=POLE(IR1_PL, I3)+F1     :     POLE(IP1_PL, I3)=POLE(IP1_PL, I3)+F5-g5 1100 CONTINUE 10 CONTINUE         </pre>	<pre> DO 100 IEDGE=1, N_ED_F   IF (IDS_ED(ICL_ED, IEDGE).GE.1) THEN     I2=IDS_ED(IN1_ED, IEDGE)     I3=IDS_ED(IN2_ED, IEDGE)     :     POLE(IR1_PL, I2)=POLE(IR1_PL, I2)-F1     :     POLE(IP1_PL, I2)=POLE(IP1_PL, I2)-F5+g5     :     POLE(IR1_PL, I3)=POLE(IR1_PL, I3)+F1     :     POLE(IP1_PL, I3)=POLE(IP1_PL, I3)+F5-g5   END IF 100 CONTINUE         </pre>		

### A.5 タイマの挿入

ここでは、実行時間計測のためにタイマを挿入するに際して行った変更の内容について説明する。タイマの挿入は、ソースプログラムに表 A.5.1 で示される 5 つのインクルードファイルを適宜挿入することにより行った。それぞれのインクルードファイルの説明及び使用方法を表 A.5.2 から A.5.6 に示す。挿入した位置については、表 A.5.7 に示す。表中、“ID” は設定した“測定区間 ID”である。同時に挿入位置として、当該測定区間を設定するためにインクルードファイル“timer\_begin.h”及び“timer\_end.h”を挿入したファイル名及びサブルーチン名を示す。サブルーチン名が二つある場合には、上段がインクルードファイル“timer\_begin.h”を挿入したサブルーチンであり、下段がインクルードファイル“timer\_end.h”を挿入したサブルーチンである。「バリア同期」の欄には、当該測定区間においてバリア同期を取った場合には「あり」を、取らなかった場合には「なし」を記述した。測定区間“LIMITER”にはタイマの挿入は行なわなかった。この測定区間では、測定区間“LIMITER 1”と“LIMITER2”の測定結果の和を測定結果とした。

また、JTAS では既にファイル名“necns2.f”に含まれるサブルーチン“SPVOCL”において、サービス関数“ETIME”が呼び出されているため、変数“ETIME”について単精度実数型の宣言がされていた。タイマ計測のためインクルードファイル“timer\_header.h”においても、変数“ETIME”について同様の宣言がされているため、これを挿入する必要から表 A.5.8 に示すようにサブルーチン“SPVOCL”における当該宣言は削除した。

表 A.5.1 タイマ計測用インクルードファイルの内容

File 名	内 容
timer_header.h	タイマ計測に必要な変数及び配列の宣言
timer_init.h	タイマ計測の初期化
timer_begin.h	タイマ計測区間の開始
timer_end.h	タイマ計測区間の終了
timer_print.h	タイマ計測結果のまとめ及び標準出力ファイルへの出力

尚、タイマ計測を行わない場合には、各 include file の代わりに同名の空ファイルを用意して、これをインクルードしコンパイルすればよい。但し、この場合でも表 A.5.2 に示すようにインクルードファイル“timer\_header.h”においては、変数“ETIME”の単精度実数型の宣言をしなければならない。

表 A.5.2 インクルードファイル “timer\_header.h” の使用方法

File 名	timer_header.h
Subroutine 名	<i>include file</i>
説 明	
タイマ計測に必要な変数及び配列の宣言を行う。	
使 用 方 法	
タイマ計測を行うためのインクルードファイル “timer_init.h”, “timer_begim.h”, “timer_end.h” 及び “timer_print.h” をインクルードしたルーチンにおいて、その宣言部に一度だけインクルードする。	
プ ロ グ ラ ム リ ス ト	
<pre> parameter ( MAXPROCESS = 128 ) parameter ( MAXTIMER   = 50 )  real*4  ETIME      ! タイマ計測を行わない場合の空ファイルにおいてもこの宣言は削除しないこと。 real*4  WRK_TIMER (2) real*8  ELB_TIMER, ELE_TIMER, TTL_TIMER, PES_TIMER real*4  CPB_TIMER, CPE_TIMER, CPU_TIMER  integer  CNT_TIMER, CNT_PES character CAP_TIMER*64 logical timer_on logical cpu_on logical barrier_on(2, MAXTIMER)  common /TIMER_AREA/ &amp;      TTL_TIMER (4, MAXTIMER), &amp;      PES_TIMER (0:MAXPROCESS-1, 4, MAXTIMER), &amp;      ELB_TIMER ( MAXTIMER), ELE_TIMER ( MAXTIMER), &amp;      CPB_TIMER (2, MAXTIMER), CPE_TIMER (2, MAXTIMER), &amp;      CPU_TIMER (2, MAXTIMER), &amp;      CNT_TIMER ( MAXTIMER), &amp;      CNT_PES  (0:MAXPROCESS-1, MAXTIMER) common /TIMER_CAPT/ CAP_TIMER (MAXTIMER) common /TIMER_FLAG/ timer_on, cpu_on, barrier_on  common /MPI_PARAM/ npes, mype, iwcomm </pre>	

表 A.5.3 インクルードファイル “timer\_init.h” の使用方法

File 名	timer_init.h
Subroutine 名	include file
説 明	
タイマ測定環境の初期化を行う。	
使 用 方 法	
以下の変数及び配列に適宜値を設定し、全ての測定区間の開始位置よりも前で、かつ MPI 環境が初期化された後で一度だけインクルードする。設定すべき変数及び配列と設定すべき内容は、以下の通り。	
変 数 名	設 定 す べ き 内 容
mype	その MPI プロセスのランクを設定
npes	MPI プロセスの数を設定
iwcomm	MPI コミュニケータの値を設定
timer_on	タイマ計測を 行 う 場合.true. 行わない場合.false.
cpu_on	CPU 時間を測定 する 場合.ture. 測定しない場合.false. (経過時間のみが測定される)
barrier_on(1, "測定区間 ID ")	測定区間開始位置でバリア同期を 取 る 場合.ture. 取らない場合.false.
barrier_on(2, "測定区間 ID ")	測定区間終了位置でバリア同期を 取 る 場合.ture. 取らない場合.false.
cap_timer ( "測定区間 ID ")	測定区間の説明 (空白の場合、測定結果は出力されない)
プ ロ グ ラ ム リ ス ト ( 抜 粋 )	
<pre> mype = myrank npes = npe_max iwcomm = MPI_COMM_WORLD timer_on = .true. cpu_on = .true. DO l=1, MAXTIMER   barrier_on(1, l) = .true.   barrier_on(2, l) = .true. END DO barrier_on(1, 33) = .false. barrier_on(2, 33) = .false. barrier_on(1, 34) = .false. barrier_on(2, 34) = .false. barrier_on(1, 35) = .false. barrier_on(2, 35) = .false. barrier_on(1, 36) = .false. barrier_on(2, 36) = .false. : c necns2.f CAP_TIMER( 1) = 'TOTAL'           ! sub. NMAIN CAP_TIMER( 2) = 'PREPROCESS'      ! sub. NMAIN CAP_TIMER( 3) = 'TIME INTEGRATION' ! sub. NMAIN c necns2.f CAP_TIMER(11) = 'SPVCR'           ! sub. SPVCRP c der.f CAP_TIMER(21) = 'SPDRF1 TOTAL'    ! sub. DR CAP_TIMER(22) = '      _Tet'      ! sub. DR : </pre>	

表 A.5.4 インクルードファイル “timer\_begin.h” の使用方法

File 名	timer_begin.h
Subroutine 名	include file
説 明	
測定区間開始位置を設定する。	
使 用 方 法	
<p>測定区間の開始位置で “測定区間 ID” を設定した後インクルードする。                  同じ “測定区間 ID” が設定されてインクルードファイル “timer_end.h” がインクルードされた位置までが測定区間となる。                  測定区間は、他の測定区間を一部分又は全て含んでも良い。                  また、複数の測定区間に同じ “測定区間 ID” を与えても良い。但し、同じ “測定区間 ID” を持つ測定区間は、一部でも重複してはならない。</p>	
プ ロ グ ラ ム リ ス ト	
<pre> cfj&gt;&gt;   IF( timer_on ) THEN     CNT_TIMER(IND_TIMER) = CNT_TIMER(IND_TIMER) + 1     IF( barrier_on(1, IND_TIMER) ) &amp;      call MPI_barrier(iwcomm, ierror)     IF( cpu_on ) THEN       CPU_TIMER(1, IND_TIMER) = ETIME( WRK_TIMER )       CPB_TIMER(1, IND_TIMER) = WRK_TIMER(1)       CPB_TIMER(2, IND_TIMER) = WRK_TIMER(2)     END IF     CALL GETTOD( ELB_TIMER(IND_TIMER) )   END IF cfj&lt;&lt;                 </pre>	
使 用 例	
<pre> IND_TIMER = 22 include "timer_begine.h" IND_TIMER = 21 include "timer_begin.h"  測定区間 } "測定区間 ID"=21  IND_TIMER = 21 include "timer_end.h"  測定区間 } "測定区間 ID"=22  IND_TIMER = 22 include "timer_end.h"                 </pre>	

表 A.5.5 インクルードファイル “timer\_end.h” の使用方法

File 名	timer_end.h
Subroutine 名	<i>include file</i>
説 明	
測定区間終了位置を設定する.	
使 用 方 法	
<p>測定区間の終了位置で“測定区間 ID”を設定した後インクルードする。          同じ“測定区間 ID”が設定されてインクルードファイル“timer_begin.h”がインクルードされた位置から、このインクルードファイルがインクルードされた位置までが測定区間となる。          測定区間は、他の測定区間を一部分又は全て含んでも良い。          また、複数の測定区間に同じ“測定区間 ID”を与えても良い。但し、同じ“測定区間 ID”を持つ測定区間は、一部でも重複してはならない。</p>	
プ ロ グ ラ ム リ ス ト	
<pre> cfj&gt;&gt;   IF( timer_on ) THEN     IF( barrier_on(2, IND_TIMER) )     &amp;    call MPI_barrier(iwcomm, ierror)     CALL GETTOD( ELE_TIMER(IND_TIMER) )     IF( cpu_on ) THEN       CPU_TIMER(2, IND_TIMER) = ETIME( WRK_TIMER )       CPE_TIMER(1, IND_TIMER) = WRK_TIMER(1)       CPE_TIMER(2, IND_TIMER) = WRK_TIMER(2)     END IF      TTL_TIMER(1, IND_TIMER) = TTL_TIMER(1, IND_TIMER)     &amp;    + ELE_TIMER( IND_TIMER)     &amp;    - ELB_TIMER( IND_TIMER)     TTL_TIMER(2, IND_TIMER) = TTL_TIMER(2, IND_TIMER)     &amp;    + CPE_TIMER(1, IND_TIMER)     &amp;    - CPB_TIMER(1, IND_TIMER)     TTL_TIMER(3, IND_TIMER) = TTL_TIMER(3, IND_TIMER)     &amp;    + CPE_TIMER(2, IND_TIMER)     &amp;    - CPB_TIMER(2, IND_TIMER)     TTL_TIMER(4, IND_TIMER) = TTL_TIMER(4, IND_TIMER)     &amp;    + CPU_TIMER(2, IND_TIMER)     &amp;    - CPU_TIMER(1, IND_TIMER)   END IF cfj&lt;&lt; </pre>	
使 用 例	
「表 A.5.4 インクルードファイル “timer_begin.h” の使用方法」参照	



表 A.5.6 インクルードファイル “timer\_print.h” の使用方法

File 名	timer_print.h
Subroutine 名	include file
説 明	
測定結果を標準出力に出力する。	
使 用 方 法	
全ての測定区間終了位置よりも後で、かつ MPI 環境が終了される前に一度だけインクルードする。	
プ ロ グ ラ ム リ ス ト	
<pre> IF( timer_on ) THEN   DO I=1, MAXTIMER     IF( CAP_TIMER(I).NE.' ' ) THEN       call MPI_gather( CNT_TIMER( I), 1, MPI_INTEGER &amp;                    , CNT_PES (0, I), 1, MPI_INTEGER &amp;                    , 0, iwcomm, ierr)       j_end = 1       if( cpu_on ) j_end = 4       DO J=1, j_end         call MPI_gather &amp;          ( TTL_TIMER( J, I), 1, MPI_DOUBLE_PRECISION &amp;          , PES_TIMER(0, J, I), 1, MPI_DOUBLE_PRECISION, 0, iwcomm, ierr)       END DO       IF( MYPE.EQ.0 ) THEN         IF( barrier_on(2, I) ) THEN           lp_end = 0         ELSE           lp_end = npes-1         END IF         WRITE(6, fmt='(a)') CAP_TIMER(I)         DO lp=0, lp_end           WRITE(6, fmt='(a, i2.2, a, l10)') &amp;          'PE NO. : ', lp, ' EXE. COUNT = ', &amp;          CNT_PES (lp, I)           WRITE(6, fmt='(a, i2.2, a, f10.2, a)') &amp;          'PE NO. : ', lp, ' ELAPS TIME = ', &amp;          PES_TIMER(lp, 1, I)*1.d-6, ' SEC.'           IF( cpu_on ) THEN             WRITE(6, fmt='(a, i2.2, a, f10.2, a)') &amp;            'PE NO. : ', lp, ' USER TIME = ', &amp;            PES_TIMER(lp, 2, I) , ' SEC.'             WRITE(6, fmt='(a, i2.2, a, f10.2, a)') &amp;            'PE NO. : ', lp, ' SYS TIME = ', &amp;            PES_TIMER(lp, 3, I) , ' SEC.'             WRITE(6, fmt='(a, i2.2, a, f10.2, a)') &amp;            'PE NO. : ', lp, ' TOTCPU TIME = ', &amp;            PES_TIMER(lp, 4, I) , ' SEC.'           END IF         END DO       END IF     END DO   END IF END IF END DO END IF </pre>	

表 A.5.7 測定区間一覧 (続く)

ID	測定区間の名称	挿入位置		説明	バリア同期
		file	routine		
01	TOTAL	necns2.f	NMAIN SPV0CL	JTAS 全体 MPI ライブラリ “MPI_initialize” が呼び出された直後から MPI ライブラリ “MPI_finalize” が呼び出される直前まで	あり
02	PREPROCESS	necns2.f	NMAIN	初期化処理 MPI ライブラリ “MPI_initialize” が呼び出された直後から サブルーチン SPV0SV が呼び出される直前まで	あり
03	INTEGRATION	necns2.f	NMAIN SPV0CL	時間積分ループ サブルーチン SPV0SV が呼び出された直後から MPI ライブラリ “MPI_finalize” が呼び出される直前まで	あり
11	SPVCOR	necns2.f	SPVCRP	HILLEW 法による数値流束ベクトル $\mathbf{F}(\mathbf{Q}) \cdot \mathbf{n}$ の計算 サブルーチン SPVCOR を呼び出す直前から サブルーチン SPVCOR を呼び終わった直後まで	あり
21	SPDRF 1	der.f	DR	勾配 $\nabla q_i$ の計算 サブルーチン SPDRF1_tet を呼び出す直前から サブルーチン SPDRF1_pyr を呼び終わった直後まで	あり
22	_tet	der.f	DR	三角錐要素の勾配 $\nabla q_i$ の計算 サブルーチン SPDRF1_tet を呼び出す直前から サブルーチン SPDRF1_tet を呼び終わった直後まで	あり
23	_prism	der.f	DR	三角柱要素の勾配 $\nabla q_i$ の計算 サブルーチン SPDRF1_pri を呼び出す直前から サブルーチン SPDRF1_pri を呼び終わった直後まで	あり
24	_pyr	der.f	DR	四角錐要素の勾配 $\nabla q_i$ の計算 サブルーチン SPDRF1_pyr を呼び出す直前から サブルーチン SPDRF1_pyr を呼び終わった直後まで	あり
--	LIMITER	-----	-----	Venkatakrisnan の制限関数 $\Psi_i (0 \leq \Psi_i \leq 1)$ の計算 区間 LIMITER 1 と区間 LIMITER2 の和	-----
25	LIMITER1	der.f	SPDRB	$\Delta_{\max}$ , $\Delta_{\min}$ の計算 サブルーチン LIMITER1 を呼び出す直前から サブルーチン LIMITER1 を呼び終わった直後まで	あり
26	LIMITER2	der.f	SPDRB	Venkatakrisnan の制限関数 $\Psi_i (0 \leq \Psi_i \leq 1)$ の計算 サブルーチン LIMITER2 を呼び出す直前から サブルーチン LIMITER2 を呼び終わった直後まで	あり

表 A.5.7 測定区間一覧 (続き)

ID	測定区間の名称	挿入位置		説明	バリア 同期
		file	routine		
31	LU-SGS	lut2.f	LUSGS	LU-SGS 法ソルバ サブルーチン LUSGS の最初から最後まで	あり
32	DIAGONAL	lut2.f	LUSGS	対角行列 $D_i$ の計算 サブルーチン DIAGONAL を呼び出す直前から サブルーチン DIAGONAL を呼び終わった直後まで	あり
33	SIGMALEFT	lut2.f	LUSGS	前進スイープの一部 サブルーチン SIGMALEFT を呼び出す直前から サブルーチン SIGMALEFT を呼び終わった直後まで	なし
34	SWEEPL	lut2.f	LUSGS	前進スイープの計算 サブルーチン SWEEPL を呼び出す直前から サブルーチン SWEEPL を呼び終わった直後まで	なし
35	SIGMARIGHT	lut2.f	LUSGS	後退スイープの一部 サブルーチン SIGMARIGHT を呼び出す直前から サブルーチン SIGMARIGHT を呼び終わった直後まで	なし
36	SWEEPR	lut2.f	LUSGS	後退スイープの計算 サブルーチン SWEEPR を呼び出す直前から サブルーチン SWEEPR を呼び終わった直後まで	なし
37	UPDATEQ	lut2.f	LUSGS	$Q^{n+1} = Q^n + \Delta Q^n$ の計算 サブルーチン UPDATEQ を呼び出す直前から サブルーチン UPDATEQ を呼び終わった直後まで	あり
41	COMMUNICA- TION	mpi.f	SUB_MPICALL	プロセス間通信 サブルーチン SUB_MPICALL の最初から最後まで	あり
42	Turbulent	mpi.f	SUB_MPICALL	乱流モデル計算後に行われるプロセス間通信 IFLAG=0 として呼び出された サブルーチン SUB_MPICALL の最初から最後まで	あり
43	Gradient	mpi.f	SUB_MPICALL	勾配 $\nabla q_i$ 計算後に行われるプロセス間通信 IFLAG=1 が設定されて呼び出された サブルーチン SUB_MPICALL の最初から最後まで	あり
44	Limiter	mpi.f	SUB_MPICALL	Venkatakrishnan の制限関数 $\Psi_i (0 \leq \Psi_i \leq 1)$ の計算後に行われるプロセス間通信 IFLAG=2 が設定されて呼び出された サブルーチン SUB_MPICALL の最初から最後まで	あり
45	Control Vol.	mpi.f	SUB_MPICALL	検査体積の表面積, 体積等の諸量計算後に行われるプロセス間通信 IFLAG=3 が設定されて呼び出された サブルーチン SUB_MPICALL の最初から最後まで	あり
46	Flux	mpi.f	SUB_MPICALL	HLEW 法による数値流束ベクトル $F(Q) \cdot n$ の計算後に行われるプロセス間通信 IFLAG=4 が設定されて呼び出された サブルーチン SUB_MPICALL の最初から最後まで	あり
47	Forward	mpi.f	SUB_MPICALL	LU-SGS 法前進スイープ後に行われるプロセス間通信 IFLAG=5 が設定されて呼び出された サブルーチン SUB_MPICALL の最初から最後まで	あり
48	Backward	mpi.f	SUB_MPICALL	LU-SGS 法後退スイープ後に行われるプロセス間通信 IFLAG=6 が設定されて呼び出された サブルーチン SUB_MPICALL の最初から最後まで	あり

表 A.5.8 サブルーチン “SPVOCL” の変更内容

File 名	necns2.f	
Subroutine 名	SPVOCL	
変 更 内 容 の 説 明		
タイマ挿入により二重宣言となる変数 “ETIME” についての単精度実数型宣言を削除した.		
変 更 内 容 リ ス ト		
変 更 前	変 更 後	
real :: wtime1, dtime1 (2), etime	real :: wtime1, dtime1 (2)	

**APPENDIX B. スレッド版チューニング内容詳細**

ここでは、スレッド版で適用した変更内容について、「B.1 スレッド並列版変更内容」にその詳細を示す。加えて、LU-SGS 法における色分けの削除の効果を評価するために行った変更について「B.2 LU-SGS 法における色分け削除の効果確認のための変更」に示す。

### B.1 スレッド版変更内容

ここでは、スレッド版で適用した変更内容について、その詳細を説明する。スレッド版においては、表B.1.1に示すサブルーチン及びインクルードファイルに対して色分けの削除，それによる再帰参照を回避するためのDO ループの分割及び節点番号の付け替え等の変更を加えスレッド並列の最適化を図った。

表 B.1.1 スレッド版において変更が加えられたルーチン一覧

File 名	Subroutine 名	変 更 内 容
der.f	DR	要素毎に計算された勾配の節点毎及び辺毎ごとの配列への足し込み (スカラ版に同じ)
	SPDRF1_pri	色分けの削除, DO ループの分割 (スカラ版に同じ)
	SPDRF1_pyr	色分けの削除, DO ループの分割 (スカラ版に同じ)
	SPDRF1_tet	色分けの削除, DO ループの分割 (スカラ版に同じ)
	LIMITER1	色分けの削除, DO ループの分割
	LIMITER2	色分けの削除, DO ループの分割
forfast.f	FORFAST	インデックステーブルの設定
	SET_ELEMENT_TABLE	インデックステーブルの設定 (スカラ版に同じ)
lut2.f	LUSGS	オリジナルに同じ
	DIAGONAL	オリジナルに同じ
	DIAGONALT	オリジナルに同じ
	DIAGONALT_sa	オリジナルに同じ
	SIGMALEFT	節点番号の付け替え (色分けは削除せず)
	SIGMALEFTT	節点番号の付け替え (色分けは削除せず) (変更のみ・未テスト)
	SIGMALEFTT_sa	節点番号の付け替え (色分けは削除せず) (変更のみ・未テスト)
	SWEEPL	節点番号の付け替え (スカラ版に同じ)
	SIGMARIGHT	節点番号の付け替え (色分けは削除せず)
	SIGMARIGHTT	節点番号の付け替え (色分けは削除せず) (変更のみ・未テスト)
	SIGMARIGHTT_sa	節点番号の付け替え (色分けは削除せず) (変更のみ・未テスト)
	SWEEPR	節点番号の付け替え (スカラ版に同じ)
	UPDATEQ	節点番号の付け替えによる参照配列の変更 (スカラ版に同じ)
	ZEROSIGMA	節点番号の付け替えによる参照配列の変更 (スカラ版に同じ)
	ZERODELT	節点番号の付け替えによる参照配列の変更 (スカラ版に同じ)
	SIGMAPREPARE	節点番号の付け替えによるインデックステーブルの変更, 新規作成
mpi.f	SETS_SUB_MPICALL	節点番号の付け替えによる参照配列の変更 (スカラ版に同じ)
	SETR_SUB_MPICALL	節点番号の付け替えによる参照配列の変更 (スカラ版に同じ)
necns.f	NMAIN	節点番号の付け替えによるゼロクリアの追加
necns.f	SPVCOR	色分けの削除, DO ループの分割
ACCUM2	<i>include file</i>	作業用配列からの足し込みに変更 (スカラ版に同じ)
dsed	<i>include file</i>	作業用配列からの足し込みに変更 (スカラ版に同じ)
DS3D	<i>include file</i>	インデックステーブルの配列宣言追加 (スカラ版に同じ)
lim	<i>include file</i>	計算結果を作業用配列へ保存するように変更
LU	<i>include file</i>	作業用配列及びインデックステーブルの配列宣言追加 インデックステーブル "jsigma" 及び "sigma1" のみスカラ版と異なる
visclam	<i>include file</i>	計算結果保存部分を削除

表 B.1.2 サブルーチン “DR” の変更内容

File 名	der.f
Subroutine 名	DR
変 更 内 容 の 説 明	
<p>サブルーチン “SPDRF1_tet”, “SPDRF1_pri” 及び “SPDRF1_pyr” において, 配列の配列の定義・参照の連続化を図るために分割された DO ループの内, 節点及び辺の配列への足し込みを行う DO ループを, サブルーチンの呼び出し元である本サブルーチンに追加し, 足し込みの一括処理を行うように変更した.</p>	
変 更 内 容 リ ス ト	
省 略 (「表 A.1.1 サブルーチン “DR” の変更内容」に同じ)	

表 B.1.3 サブルーチン “SPDRF1\_pri” の変更内容

File 名	der.f
Subroutine 名	SPDRF1_pri
変 更 内 容 の 説 明	
<p>色分けを削除するとともに, DO ループを分割し要素毎の勾配を一旦作業用配列に保存することで, 配列の定義・参照の連続化を図った. 作業用配列は, 三角錐要素の勾配の保存に利用した配列と同じものを使用し, これに追加することとした.</p> <p>節点及び辺の配列への足し込みは, サブルーチン “SPDRF1_tet” 及び “SPDRF1_pyr” と統合して, これらのサブルーチンの呼び出し元であるサブルーチン “dr” にて一括して行うように変更した.</p>	
変 更 内 容 リ ス ト	
省 略 (「表 A.1.2 サブルーチン “SPDRF1_pri” の変更内容」に同じ)	

表 B.1.4 サブルーチン “SPDRF1\_pyr” の変更内容

File 名	der.f
Subroutine 名	SPDRF1_pyr
変 更 内 容 の 説 明	
<p>色分けを削除するとともに, DO ループを分割し要素毎の勾配を一旦作業用配列に保存することで, 配列の定義・参照の連続化を図った. 作業用配列は, 三角錐要素及び三角柱要素の勾配の保存に利用した配列と同じものを使用し, これに追加することとした.</p> <p>節点及び辺の配列への足し込みは, サブルーチン “SPDRF1_tet” 及び “SPDRF1_pri” と統合して, これらのサブルーチンの呼び出し元であるサブルーチン “dr” にて一括して行うように変更した.</p>	
変 更 内 容 リ ス ト	
省 略 (「表 A.1.3 サブルーチン “SPDRF1_pyr” の変更内容」に同じ)	

表 B.1.5 サブルーチン “SPDRF1\_tet” の変更内容

File 名	der.f
Subroutine 名	SPDRF1_tet
変 更 内 容 の 説 明	
<p>色分けを削除するとともに, DO ループを分割し要素毎の勾配を一旦作業用配列に保存することで, 配列の定義・参照の連続化を図った.</p> <p>節点及び辺の配列への足し込みは, サブルーチン “SPDRF1_pri” 及び “SPDRF1_pyr” と統合して, これらのサブルーチンの呼び出し元であるサブルーチン “dr” にて一括して行うように変更した.</p>	
変 更 内 容 リ ス ト	
省 略 (「表 A.1.4 サブルーチン “SPDRF1_tet” の変更内容」に同じ)	

表 B.1.6 サブルーチン “LIMITER1” の変更内容

File 名	der.f	
Subroutine 名	LIMITER1	
変更内容の説明		
<p>スカラ版を元に DO ループの分割を行い再帰参照を回避することにより、スレッド並列を可能とした。          (「3.2 スレッド版変更内容」参照)</p> <p>このサブルーチン “LIMITER1” においては、各節点においてその節点に隣接する全ての節点との間で、保存量の大小比較を行うのみなので、変更後のプログラムにおいてもリスト3.3 DO 100に相当する計算結果を作業用配列に保存する DO ループは存在しない。</p>		
変更内容リスト		
変更前 (スカラ版)	変更後	
<pre> DO 100 IEDGE=1, N_ED_F   IF (IDS_ED (ICL_ED, IEDGE). GE. 1) THEN     I2 =IDS_ED (IN1_ED, IEDGE)     I3 =IDS_ED (IN2_ED, IEDGE)      R2 =DS_ND (IRO_ND, I2)     UX2=DS_ND (IUO_ND, I2)     UY2=DS_ND (IVO_ND, I2)     UZ2=DS_ND (IWO_ND, I2)     P2 =DS_ND (IPO_ND, I2)     rt2=rt (i2)     R3 =DS_ND (IRO_ND, I3)     UX3=DS_ND (IUO_ND, I3)     UY3=DS_ND (IVO_ND, I3)     UZ3=DS_ND (IWO_ND, I3)     P3 =DS_ND (IPO_ND, I3)     rt3=rt (i3)     POLE (I11_PL , I2)= &amp;      MIN (POLE (I11_PL , I2), R3)     POLE (I12_PL , I2)= &amp;      MAX (POLE (I12_PL , I2), R3)     :     POLE (I11_PL , I3)= &amp;      MIN (POLE (I11_PL , I3), R2)     POLE (I12_PL , I3)= &amp;      MAX (POLE (I12_PL , I3), R2)     :   END IF 100 CONTINUE </pre>	<pre> DO 110 IP=1, N_P_F   NEDG=IEDND_TBL (0, IP)   DO 111 IE=1, NEDG     IEDGE=IEDND_TBL (IE, IP)     IF (IEDGE. GE. 0) THEN       I23=IDS_ED (IN1_ED, ABS (IEDGE))     ELSE       I23=IDS_ED (IN2_ED, ABS (IEDGE))     END IF     R23 =DS_ND (IRO_ND, I23)     UX23=DS_ND (IUO_ND, I23)     UY23=DS_ND (IVO_ND, I23)     UZ23=DS_ND (IWO_ND, I23)     P23 =DS_ND (IPO_ND, I23)     RT23=RT (I23)      POLE (I11_PL , IP)= &amp;      MIN (POLE (I11_PL , IP), R23)     POLE (I12_PL , IP)= &amp;      MAX (POLE (I12_PL , IP), R23)     :   111 CONTINUE 110 CONTINUE </pre>	



表 B.1.7 サブルーチン “LIMITER2” の変更内容

File 名	der.f
Subroutine 名	LIMITER2
変更内容の説明	
<p>スカラ版を元に DO ループの分割を行い再帰参照を回避することにより、スレッド並列を可能とした。            (「3.2 スレッド版変更内容」参照)</p>	
変更内容リスト	
変更前 (スカラ版)	変更後
<pre> DO 100 IEDGE=1, N_ED_F   IF (IDS_ED (ICL_ED, IEDGE). GE. 1) THEN     I2=IDS_ED (IN1_ED, IEDGE)     I3=IDS_ED (IN2_ED, IEDGE)     :     del2=r1     qmax=rmax2     qmin=rmin2     q =r2     kq =kq1_pl     ll =i2     include 'lim'     :     del2=rr     qmax=rmax3     qmin=rmin3     q =r3     kq =kq1_pl     ll =i3     include 'lim'     :   END IF 100 CONTINUE           </pre>	<pre> DO 100 IEDGE=1, N_ED_F   IF (IDS_ED (ICL_ED, IEDGE). GE. 1) THEN     I2=IDS_ED (IN1_ED, IEDGE)     I3=IDS_ED (IN2_ED, IEDGE)     :     del2=r1     qmax=rmax2     qmin=rmin2     q =r2     ii =1     ll =i2     include 'lim.EDG'     :     del2=rr     qmax=rmax3     qmin=rmin3     q =r3     ii =7     ll =i3     include 'lim.EDG'     :   END IF 100 CONTINUE   DO 110 IP=1, N_P_F     NEDG=IEDND_TBL (0, IP)     DO 111 IE=1, NEDG       IEDGE= IEDND_TBL (IE, IP)       IIST =(SIGN (1, IEDGE)+1)*3       IEDGE= ABS (IEDGE)       POLE (KQ1_PL , IP)=       &amp; MIN (POLE (KQ1_PL , IP),       &amp; EDG_WK (IIST+1, IEDGE))     111 CONTINUE   110 CONTINUE           </pre>

表 B.1.8 サブルーチン “FORFAST” の変更内容

File 名	forfast.f
Subroutine 名	FORFAST
変 更 内 容 の 説 明	
<p>スカラ版に節点-辺対応テーブル “IEDNE_TBL” の設定処理を追加した。  <u>節点-辺対応テーブル “IEDNE_TBL”</u>  <math>ned = \text{IEDND\_TBL}(0, nd)</math> : 節点番号 <math>nd</math> を含む辺の数 <math>ned</math>  <math>ed = \text{IEDND\_TBL}(ied, nd)</math> : 節点番号 <math>nd</math> を含む <math>ied</math> 番目の辺番号 <math>ed</math>  <math>ed &lt; 0</math> : <math>nd = \text{IDS\_ED}(\text{IN1\_ED}, ed)</math>  <math>ed &gt; 0</math> : <math>nd = \text{IDS\_ED}(\text{IN2\_ED}, ed)</math></p>	
変 更 内 容 リ ス ト	
<p>(スカラ版のサブルーチン “FORFAST”) の最後に以下の処理を追加  スカラ版の変更内容については、表 A.1.5 「サブルーチン」 FORFAST” の変更」参照)</p> <pre> DO J=1, IMX_ND DO I=0, IMX_EDND   IEDND_TBL(I, J) = 0 END DO END DO DO IEDGE=1, N_ED_F   IF (IDS_ED( ICL_ED, IEDGE).GE.1) THEN     I2=IDS_ED( IN1_ED, IEDGE)     I3=IDS_ED( IN2_ED, IEDGE)     II       = IEDND_TBL(0, I2) + 1     IF (II.GT. IMX_EDND) THEN       WRITE(*,*) 'Fail FORFAST ED in ND ', IMX_EDND, II       CALL MPI_abort(MPI_COMM_WORLD, 1)       STOP     END IF     IEDND_TBL(0, I2) = II     IEDND_TBL(II, I2) = -IEDGE     II       = IEDND_TBL(0, I3) + 1     IF (II.GT. IMX_EDND) THEN       WRITE(*,*) 'Fail FORFAST ED in ND ', IMX_EDND, II       CALL MPI_abort(MPI_COMM_WORLD, 1)       STOP     END IF     IEDND_TBL(0, I3) = II     IEDND_TBL(II, I3) = IEDGE   END IF END DO </pre>	

表 B.1.9 新規作成サブルーチン “SET\_ELEMENT\_TABLE”

File 名	forfast.f
Subroutine 名	SET_ELEMENT_TABLE( INDX_TBL, MXSIZ1, MXSIZ2, INDX, IELM, MSG )
変更内容の説明	
新規作成ルーチン 節点-要素対応テーブル “IND_TBL” 及び辺-要素対応テーブル “IED_TBL” を設定するサブルーチンを新規作成	
処理内容 表 A.1.6 「新規作成サブルーチン “SET_ELEMENT_TABLE”」に同じ	
変更内容リスト	
省 略 (「表 A.1.6 「新規作成サブルーチン “SET_ELEMENT_TABLE”」に同じ)	

表 B.1.10 サブルーチン “LUSGS” の変更内容

File 名	lut2.f
Subroutine 名	LUSGS
変更内容の説明	
ハイパー面を考慮した節点番号の付け替えのため、オリジナルの節点番号で保存されている配列から新しい節点番号で保存される作業用配列にコピーする処理を追加した。	
変更内容リスト	
省 略 (「表 A.3.1 「サブルーチン “LUSGS” の変更」に同じ)	

表 B.1.11 サブルーチン “DIAGONAL” の変更内容

File 名	lut2.f
Subroutine 名	DIAGONAL
変更内容の説明	
オリジナルに同じ (変更なし).	

表 B.1.12 サブルーチン “DIAGONALT” の変更内容

File 名	lut2.f
Subroutine 名	DIAGONALT
変更内容の説明	
オリジナルに同じ (変更なし).	

表 B.1.13 サブルーチン “DIAGONALT\_sa” の変更内容

File 名	lut2.f
Subroutine 名	DIAGONALT_sa
変更内容の説明	
オリジナルに同じ (変更なし).	

表 B.1.14 サブルーチン “SIGMALEFT” の変更内容

File 名	lut2.f	
Subroutine 名	SIGMALEFT	
変更内容の説明		
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。 色分けの削除は行わなかった。		
変更内容リスト		
変更前	変更後	
<pre> DO 10 ICOLOR=1, MCL_lu(i level)   jmax=jsigma(icolor, i level) *VOCL LOOP, NOVREC CDIR\$ IVDEP *VDIR NODEP   DO 1100 J=jmin, JMAX     IEDGE=ITMPI(J)     I2 =IDS_ED(IN1_ED, IEDGE)     I3 =IDS_ED(IN2_ED, IEDGE)     itp=markedg(iedge)     if(itp.lt.0) then       ii=i2       i2=i3       i3=ii     end if     :     R =DS_ND(IRO_ND, I2)     :     P =DS_ND(IPO_ND, I2)     RTT =rt(i2)     deltr =pole(idlt1, i2)     :     deltrt=pole(idlt6, i2)     :     unu=fmu/r/re/pole(ivl_pl, ia)*surf(ia)     :     &amp; pole(mq1_pl, ia)=pole(mq1_pl, ia)     &amp; -delf1-specr*deltr     :     &amp; pole(mq6_pl, ia)=pole(mq6_pl, ia)     &amp; -delf6-specr*deltrt 1100 continue   jmin=jmax+1 10 continue </pre>	<pre> DO 10 ICOLOR=1, MCL_lu(i level)   jmax=jsigma(icolor, i level) *VOCL LOOP, NOVREC CDIR\$ IVDEP *VDIR NODEP   DO 1100 J=jmin, JMAX     IEDGE=ITMPI(J)     I12 =IDS_TBL(1, IEDGE)     I1A =IDS_TBL(2, IEDGE)     :     R =WDS_ND(NRO_ND, I12)     :     P =WDS_ND(NPO_ND, I12)     RTT=WDS_ND(NRT_ND, I12)     deltr =deltQ(idlt1, ii2)     :     deltrt=deltQ(idlt6, ii2)     :     &amp; unu =fmu/r/re/wds_nd(nvl_pl, ia)     &amp; *wds_nd(nsf_pl, ia)     :     &amp; delTQ(mq1_pl, ia) =deltQ(mq1_pl, ia)     &amp; -delf1-specr*deltr     :     &amp; delTQ(mq6_pl, ia) =deltQ(mq6_pl, ia)     &amp; -delf6-specr*deltrt 1100 continue   jmin=jmax+1 10 continue </pre>	

表 B.1.15 サブルーチン “SIGMALEFTT” の変更内容

File 名	lut2.f
Subroutine 名	SIGMALEFTT
変 更 内 容 の 説 明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。 色分けの削除は行わなかった。 変更は行ったがテストは行っていないので注意されたい。	
変 更 内 容 リ ス ト	
省 略 (「表 B.1.14 サブルーチン “SIGMALEFT” の変更内容」に同じ)	

表 B.1.16 サブルーチン “SIGMALEFTT\_sa” の変更内容

File 名	lut2.f
Subroutine 名	SIGMALEFTT_sa
変 更 内 容 の 説 明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。 色分けの削除は行わなかった。 変更は行ったがテストは行っていないので注意されたい。	
変 更 内 容 リ ス ト	
省 略 (「表 B.1.14 サブルーチン “SIGMALEFT” の変更内容」に同じ)	

表 B.1.17 サブルーチン “SWEEPL” の変更内容

File 名	lut2.f
Subroutine 名	SWEEPL
変 更 内 容 の 説 明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.8 サブルーチン “SWEEPL” の変更内容」に同じ)	

表 B.1.18 サブルーチン “SIGMARIGHT” の変更内容

File 名	lut2.f	
Subroutine 名	SIGMARIGHT	
変更内容の説明		
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。 色分けの削除は行わなかった。		
変更内容リスト		
変更前	変更後	
<pre> DO 10 ICOLOR=1, MCL_lu1(ilevel)   jmax=jsigma1(icolor, ilevel) *VOCL LOOP, NOVREC CDIR\$ IVDEP *VDIR NODEP   DO 1100 J=jmin, JMAX     IEDGE=ITMPPr(J)     I2=IDS_ED(IN1_ED, IEDGE)     I3=IDS_ED(IN2_ED, IEDGE)     itp=markedg1(iedge)     if(itp.gt.0) then       ii=i2       i2=i3       i3=ii     end if       :       R =DS_ND(IRO_ND, I2)       :       P =DS_ND(IPO_ND, I2)       RTT =rt(i2)       deltr =pole(kdlt1, i2)       :       deltrt=pole(kdlt6, i2)       :       unu=fmu/r/re/pole(ivl_pl, ia)       *surf(ia)       :       pole(mq1_pl, ia)=pole(mq1_pl, ia)       -delf1-specr*deltr       :       pole(mq6_pl, ia)=pole(mq6_pl, ia)       -delf6-specr*deltrt 1100 continue   jmin=jmax+1 10 continue </pre>	<pre> DO 10 ICOLOR=1, MCL_lu1(ilevel)   jmax=isigma1(icolor, ilevel) *VOCL LOOP, NOVREC CDIR\$ IVDEP *VDIR NODEP   DO 1100 J=jmin, JMAX     IEDGE= ITMPPr(J)     I12 = IDS_TBL(2, IEDGE)     I1A = IDS_TBL(1, IEDGE)       :       R =WDS_ND(NRO_ND, I12)       :       P =WDS_ND(NPO_ND, I12)       RTT=WDS_ND(NRT_ND, I12)       deltr =deltQ(kdlt1, i12)       :       deltrt=deltQ(kdlt6, i12)       :       unu =fmu/r/re/wds_nd(nvl_pl, i1a)       &amp; *wds_nd(nsf_pl, i1a)       :       deltrQ(mq1_pl, i1a)=deltQ(mq1_pl, i1a)       &amp; -delf1-specr*deltr       :       deltrQ(mq6_pl, i1a)=deltQ(mq6_pl, i1a)       &amp; -delf6-specr*deltrt 1100 CONTINUE   jmin=jmax+1 10 CONTINUE </pre>	

表 B.1.19 サブルーチン “SIGMARIGHTT” の変更内容

File 名	lut2.f
Subroutine 名	SIGMARIGHTT
変 更 内 容 の 説 明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。 色分けの削除は行わなかった。 変更は行ったがテストは行っていませんので注意されたい。	
変 更 内 容 リ ス ト	
省 略 (「表 B.1.18 サブルーチン “SIGMARIGHT” の変更内容」に同じ)	

表 B.1.20 サブルーチン “SIGMARIGHTT\_sa” の変更内容

File 名	lut2.f
Subroutine 名	SIGMARIGHTT_sa
変 更 内 容 の 説 明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。 色分けの削除は行わなかった。 変更は行ったがテストは行っていませんので注意されたい。	
変 更 内 容 リ ス ト	
省 略 (「表 B.1.18 サブルーチン “SIGMARIGHT” の変更内容」に同じ)	

表 B.1.21 サブルーチン “SWEEP” の変更内容

File 名	lut2.f
Subroutine 名	SWEEP
変 更 内 容 の 説 明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.12 サブルーチン “SWEEP” の変更内容」に同じ)	

表 B.1.22 サブルーチン “UPDATEQ” の変更内容

File 名	lut2.f
Subroutine 名	UPDATEQ
変 更 内 容 の 説 明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。 このサブルーチンにおいて、ハイパー面を考慮した新しい節点の番号付けによる配列からオリジナルの配列 に書き戻しが行われている。従って、「変更前」の DO ループは、連続アクセスであるのに対して、「変更後」 ではインデックス参照となっており、若干のオーバーヘッドが発生する。	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.13 サブルーチン “UPDATEQ” の変更内容」に同じ)	

表 B.1.23 サブルーチン “ZEROSIGMA” の変更内容

File 名	lut2.f
Subroutine 名	ZEROSIGMA
変更内容の説明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。	
変更内容リスト	
省 略 (「表 A.3.14 サブルーチン “ZEROSIGMA” の変更内容」に同じ)	

表 B.1.24 サブルーチン “ZERODELT” の変更内容

File 名	lut2.f
Subroutine 名	ZERODELT
変更内容の説明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更した。	
変更内容リスト	
省 略 (「表 A.3.15 サブルーチン “ZERODELT” の変更内容」に同じ)	

表 B.1.25 サブルーチン “SIGMAPREPARE” の変更内容

File 名	lut2.f
Subroutine 名	SIGMAPREPARE
変更内容の説明	
ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更するため、インデックステーブルを設定する処理を追加した。 スカラ版に含まれる他の変更は、スレッド版では加えていない。	
変更内容リスト	
省 略 (「表 A.3.16 サブルーチン “SIGMAPREPARE” の変更内容 (2/4)」に同じ)	

表 B.1.26 サブルーチン “SETS\_SUB\_MPICALL” の変更内容

File 名	mpi.f
Subroutine 名	SETS_SUB_MPICALL
変更内容の説明	
LU-SGS 法において、ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更したため、前進スイープ (ifalg=5) 及び後退スイープ (iflag=6) の後に行うプロセス間通信についても、その対象とする配列を変更した。iflag=7 は、サブルーチン “SIGMALEFT” 及び “SIGMARIGHT” における計算結果に関するプロセス間通信であり、これも同時に修正したが、現在の JTAS においてこのプロセス間通信が実行されることはない。	
変更内容リスト	
省 略 (「表 A.3.19 サブルーチン “SETS_SUB_MPICALL” の変更内容」に同じ)	



表 B.1.27 サブルーチン “SETR\_SUB\_MPICALL” の変更内容

File 名	mpi.f
Subroutine 名	SETR_SUB_MPICALL
変 更 内 容 の 説 明	
<p>LU-SGS 法において、ハイパー面を考慮して節点番号を付け替えた配列を用いて計算を行うように変更したため、前進スイープ (ifalg=5) 及び後退スイープ (iflag=6) の後に行うプロセス間通信についても、その対象とする配列を変更した。iflag=7 は、サブルーチン “SIGMALEFT” 及び “SIGMARIGT” における計算結果に関するプロセス間通信であり、これも同時に修正したが、現在の JTAS においてこのプロセス間通信が実行されることはない。</p>	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.20 サブルーチン “SETR_SUB_MPICALL” の変更内容」に同じ)	

表 B.1.28 サブルーチン “NMAIN” の変更内容

File 名	necns2.f
Subroutine 名	NMAIN
変 更 内 容 の 説 明	
<p>一ステップ当たりの変化分 <math>\Delta Q</math> を保存するための配列 “deltQ” を新設したため、この配列のゼロクリア処理を追加した。</p>	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.21 サブルーチン “NMAIN” の変更内容」に同じ)	

表 B.1.29 サブルーチン “SPVCOR” の変更内容

File 名	necns2. f
Subroutine 名	SPVCOR
変 更 内 容 の 説 明	
スカラ版を元に DO ループの分割を行い再帰参照を回避することにより，スレッド並列を可能とした。 (「3.2 スレッド版変更内容」参照)	
変 更 内 容 リ ス ト	
変 更 前 (スカラ版)	変 更 後
<pre> DO 100 IEDGE=1, N_ED_F   IF (IDS_ED (ICL_ED, IEDGE). GE. 1) THEN     :     POLE (IR1_PL, I2)=POLE (IR1_PL, I2) -F1     :     POLE (IR1_PL, I3)=POLE (IR1_PL, I3) +F1     :   END IF 100 CONTINUE </pre>	<pre> DO 100 IEDGE=1, N_ED_F   IF (IDS_ED (ICL_ED, IEDGE). GE. 1) THEN     :     EDG_WK ( 1, IEDGE)=F1     :     EDG_WK ( 8, IEDGE)=G7   END IF 100 CONTINUE DO 110 IP=1, N_P_F   NEDG=IEDND_TBL (0, IP)   DO 111 IE=1, NEDG     IEDGE=IEDND_TBL (IE, IP)     XSIGN=SIGN (1. 0D0, IEDGE)     IEDGE=ABS (IEDGE)     R23=DS_ND (IRO_ND, IP)     F1 =EDG_WK ( 1, IEDGE)     :     G7 =EDG_WK ( 8, IEDGE)     POLE (IR1_PL, IP)=POLE (IR1_PL, IP)     &amp;                               +XSIGN*F1     :     POLE (IP1_PL, IP)=POLE (IP1_PL, IP)     &amp;                               +XSIGN*FG5     TURB (IP)=TURB (IP)     &amp;                               +XSIGN* (F6-G6*R23)     TURB1 (IP)=TURB1 (IP)     &amp;                               -XSIGN* G7   111 CONTINUE 110 CONTINUE </pre>

表 B.1.30 インクルードファイル “ACCUM2” の変更内容

File 名	ACCUM2
Subroutine 名	include file
変 更 内 容 の 説 明	
サブルーチン “SPDRF1_tet”, “SPDRF1_pri” 及び “SPDRF1_pyr” において，DO ループを分割したことによる変更である．要素毎の勾配を，節点の配列に直接足し込まず，計算結果が保存された配列から足し込むように変更した．	
変 更 内 容 リ ス ト	
省 略 (「表 A.1.7 インクルードファイル “ACCUM2” の変更内容」に同じ)	

表 B.1.31 インクルードファイル “DS3D” の変更内容

File 名	DS3D		
Subroutine 名	include file		
変更内容の説明			
<p>サブルーチン “SPDRF1_tet”, “SPDRF1_pri” 及び “SPDRF1_pyr” において, DO ループを分割したことによる変更である. 要素毎の勾配を, 辺の配列に直接足し込まず, 計算結果が保存された配列から足し込むように変更した.</p>			
変更内容リスト			
省略 (「表 A.1.8 インクルードファイル “DS3D” の変更内容」に同じ)			
変更内容の説明			
上記変更に加え, 以下のパラメータ, 作業用配列及びインデックステーブルを追加した.			
変数名	型	サイズ	説明
IMX_EDND	I4	1	一つの節点を共有する辺の最大値
EDG_WK	R8	(12,IMX_ED)	辺ごとに計算された結果を保存するための作業用配列
IEDND_TBL	I4	(0:IMX_EDND,IMX_ND)	節点-辺対応テーブル $ned = IEDND\_TBL(0, nd)$ : 節点番号 $nd$ を含む辺の数 $ned$ $ed = IEDND\_TBL(ied, nd)$ : 節点番号 $nd$ を含む $ied$ 番目の辺番号 $ed$ $ed < 0$ : $nd = IDS\_ED(IN1\_ED, ed)$ $ed > 0$ : $nd = IDS\_ED(IN2\_ED, ed)$
変更内容リスト			
変更前 (スカラ版)		変更後	
<pre>COMMON/ITLEFT/ itleft COMMON/TILEFT/ tileft  C PARAMETER( &amp;      IMX_ELND  =50, &amp;      IMX_ELED  =10) : COMMON/ND_WRK/IND_TBL( 0:IMX_ELND, IMX_ND) COMMON/ED_WRK/  &amp;      IED_TBL( 0:IMX_ELED, IMX_ED), &amp;      IDS_TBL(      2, IMX_ED)</pre>		<pre>COMMON/ITLEFT/ itleft COMMON/TILEFT/ tileft  C PARAMETER( &amp;      IMX_ELND  =50, &amp;      IMX_EDND  =25, &amp;      IMX_ELED  =10) : COMMON/ND_WRK/IND_TBL( 0:IMX_ELND, IMX_ND) COMMON/ED_WRK/  &amp;      EDG_WK(12, IMX_ED), &amp;      IEDND_TBL( 0:IMX_EDND, IMX_ND), &amp;      IED_TBL( 0:IMX_ELED, IMX_ED), &amp;      IDS_TBL(      2, IMX_ED)</pre>	

表 B.1.32 インクルードファイル“lim”の変更内容

File 名	lim		
Subroutine 名	include file		
変 更 内 容 の 説 明			
サブルーチン“LIMITER2”において、DO ループを分割したことによる変更である。辺毎の制限関数 $\Psi_i$ を、節点の配列に直接設定せず、作業用配列に保存するように変更した。 (「3.2 スレッド版変更内容」参照)			
変 更 前		変 更 後	
File 名	lim	File 名	lim, EDG
変更内容リスト	<pre> plus=(1.+sign(1.D0, del2))/2. : psi=psip*plus+psim*(1.-plus) pole(kq, ll) = min (pole(kq, ll), psi) </pre>		
		<pre> plus =(1.+sign(1.D0, del2))/2. : psi=psip*plus+psim*(1.-plus) EDG_WK(ll, IEDGE)=PSI </pre>	

表 B.1.33 インクルードファイル“LU”の変更内容

File 名	LU
Subroutine 名	include file
変 更 内 容 の 説 明	
ハイパー面を考慮した節点番号を付け替えて保存するための配列の宣言を追加した。	
変 更 内 容 リ ス ト	
省 略 (「表 A.3.23 インクルードファイル“LU”の変更内容」に同じ)	

表 B.1.34 インクルードファイル“visclam”の変更内容

File 名	visclam		
Subroutine 名	include file		
変 更 内 容 の 説 明			
このインクルードファイルをインクルードしているサブルーチン“SPVCOR”にて、計算結果を作業用配列に保存するように変更し、このインクルードファイルにおける再帰参照を含む計算結果保存部分を削除した。			
変 更 内 容 リ ス ト			
変 更 前		変 更 後	
<pre> : g7      =(drtx*sx+drtysy+drtz*sz)*rey turb1(i2)=turb1(i2)+g7 turb1(i3)=turb1(i3)-g7 g6      =-g7*fnut*sigm </pre>		<pre> : g7      =(drtx*sx+drtysy+drtz*sz)*rey c      turb1(i2)=turb1(i2)+g7 c      turb1(i3)=turb1(i3)-g7 g6      =-g7*fnut*sigm </pre>	

## B.2 LU-SGS 法における色分け削除の効果確認のための変更

ここでは、LU-SGS 法における色分け削除の効果確認のために行った変更について、その詳細を示す。表 B.2.1 に LU-SGS 法に関して色分け削除のために変更が必要となるファイル及びサブルーチンの一覧を示す。サブルーチン “DIAGONALT” 等、乱流モデルとして Goldberg-Ramakrishnan モデルまたは Spallart-Allmaras モデルが指定された場合に呼び出しの対象となるサブルーチン群は変更の対象としなかった。また、LU-SGS 法に関わる部分以外は、スレッド版のソースプログラムを使用した。

表 B.2.2 以下に、JTAS オリジナルから LU-SGS 法に関する部分の色分けを削除する場合に必要な変更内容の詳細を示す。

表 B.2.1 LU-SGS 法における色分け削除のために変更が必要なルーチン一覧

File 名	Subroutine 名	変 更 内 容
forfast. f	FORFAST	スレッド版に同じ
lut2.f	LUSGS	サブルーチン “SIGMALEFT” 及び “SIGMARIGHT” 呼び出し時の引数追加
	DIAGONAL	色分けの削除及び DO ループ分割
	SIGMALEFT	色分けの削除及び DO ループ分割
	SIGMARIGHT	色分けの削除及び DO ループ分割
	SIGMAPREPARE	インデックステーブルの設定追加
DS3D	<i>include file</i>	スレッド版に同じ
LU	<i>include file</i>	インデックステーブルの配列宣言変更及び追加

表 B.2.2 サブルーチン “FOFAST” の変更内容

File 名	forfast.f
Subroutine 名	FORFAST
変 更 内 容 の 説 明	
色分けを削除し、DO ループの分割に必要な各種インデックステーブルの設定を追加した。	
変 更 内 容 リ ス ト	
省 略 (「表 B.1.8 サブルーチン “FORFAST” の変更内容」に同じ)	

表 B.2.3 サブルーチン “LUSGS” の変更内容

File 名	lut2.f
Subroutine 名	LUSGS
変 更 内 容 の 説 明	
サブルーチン “SIGMALEFT” 及び “SIGMARIGHT” に仮引数を追加したことにともない、これらの呼び出し文において対応する実引数を追加した。	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> do iplane=1,numplane   :   if(iturb.ne.1.and.iturb.ne.2) &amp;   call sigmaleft(ilevel)   : end do do iplane=numplane,1,-1   :   if(iturb.ne.1.and.iturb.ne.2) &amp;   call sigmaright(ilevel)   : </pre>	<pre> do iplane=1,numplane   :   if(iturb.ne.1.and.iturb.ne.2) &amp;   call sigmaleft(ipplane,ilevel)   : end do do iplane=numplane,1,-1   :   if(iturb.ne.1.and.iturb.ne.2) &amp;   call sigmaright(ipplane,ilevel)   : </pre>

表 B.2.4 サブルーチン “DIAGONAL” の変更内容

File 名	lut2.f
Subroutine 名	DIAGONAL
変 更 内 容 の 説 明	
<p>スカラ版の変更に加え、DO ループの分割を行い再帰参照を回避することにより、スレッド並列を可能とした。</p> <p>スカラ版の変更内容については、「表 A.3.2 サブルーチン “DIAGONAL” の変更内容」参照のこと。</p>	
変 更 内 容 リ ス ト	
変 更 前 (スカラ版)	変 更 後
<pre> DO 10 IEDGE=1, N_ED_F   IF (IDS_ED(1CL_ED, IEDGE).GE.1) THEN     I2=IDS_ED(IN1_ED, IEDGE)     I3=IDS_ED(IN2_ED, IEDGE)     :     pole(idiag, I2)=pole(idiag, I2)     &amp;      +(0.5*abs(zf2)+aa2)*capa+unu     :     pole(idiag, I3)=pole(idiag, I3)     &amp;      +(0.5*abs(zf3)+aa3)*capa+unu   END IF 10 CONTINUE </pre>	<pre> DO 10 IEDGE=1, N_ED_F   IF (IDS_ED(1CL_ED, IEDGE).GE.1) THEN     I2=IDS_ED(IN1_ED, IEDGE)     I3=IDS_ED(IN2_ED, IEDGE)     :     EDG_WK(1, IEDGE)=     &amp;      (0.5*abs(zf2)+aa2)*capa+unu     :     EDG_WK(2, IEDGE)=     &amp;      (0.5*abs(zf3)+aa3)*capa+unu   END IF 10 CONTINUE DO 110 IP=1, N_P_F   NEDG=IEDND_TBL(0, IP)   DO 111 IE=1, NEDG     IEDGE=IEDND_TBL(IE, IP)     IEDGA=ABS(IEDGE)     IF (IEDGE.GE.0) THEN       POLE(IDIAG, IP)=POLE(IDIAG, IP)       &amp;      +EDG_WK(2, IEDGA)     ELSE       POLE(IDIAG, IP)=POLE(IDIAG, IP)       &amp;      +EDG_WK(1, IEDGA)     END IF   111 CONTINUE 110 CONTINUE </pre>

表 B.2.5 サブルーチン “SIGMALEFT” の変更内容

File 名	lut2.f
Subroutine 名	SIGMALEFT
変更内容の説明	
色分けを削除するとともに、DO ループの分割を行うことで再帰参照を回避しスレッド並列を可能とした。サブルーチン “SIGMALEFTT” 及び “SIGMALEFTT_sa” にも同様の変更を加えれば、インデックステーブル “isigma” 及び “JTMPI” を新規作成しなくても “jsigma” 及び “ITMPI” の再利用が可能である。	
変更内容リスト	
変更前	変更後
<pre> subroutine sigmaleft(ilevel) : if(ilevel.eq.1) then   jmin=1 else   if(mcl_lu(ilevel-1).le.0) return   jmin=jsigma(mcl_lu(ilevel-1),ilevel-1)+1 end if DO 10 ICOLOR=1,MCL_lu(ilevel)   jmax=jsigma(icolor,ilevel) *VOCL LOOP, NOVREC CDIR\$ IVDEP *VDIR NODEP   DO 1100 J=jmin, JMAX     IEDGE=ITMPI(J) :     pole(mq1_pl, ia)=pole(mq1_pl, ia) &amp;     -delf1-specr*deltr :     pole(mq6_pl, ia)=pole(mq6_pl, ia) &amp;     -delf6-specr*deltrt 1100 continue   jmin=jmax+1 10 continue </pre>	<pre> subroutine sigmaleft(iplane,ilevel) : if(ilevel.ne.1 .and. &amp; mcl_lu(ilevel-1).le.0) return jmin=isigma(ilevel-1)+1 jmax=isigma(ilevel) : DO 100 J=jmin,JMAX   IEDGE=JTMPI(J) :   EDG_WK(1, J)=delf1+specr*deltr :   EDG_WK(6, J)=delf6+specr*deltrt : 100 continue DO 110 IIP=NDSIGMA(IPLANE-1)+1, &amp; NDSIGMA(IPLANE)   IP =JTMP(IIP)   NEDG=IEDND_TBL(0, IP) DO 111 IE=1, NEDG   IEDGE=IEDND_TBL(IE, IP)   IEDGA=ABS(IEDGE)   ITP =MARKEDG(IEDGA)   IF( IEDGE*ITP.GE.0 ) THEN     J=ITMPW(IEDGA)     POLE(MQ1_PL, IP)=POLE(MQ1_PL, IP) &amp;     -EDG_WK(1, J) :     POLE(MQ6_PL, IP)=POLE(MQ6_PL, IP) &amp;     -EDG_WK(6, J)   END IF 111 CONTINUE 110 CONTINUE </pre>

表 B.2.6 サブルーチン “SIGMARIGHT” の変更内容

File 名	lut2.f
Subroutine 名	SIGMARIGHT
変 更 内 容 の 説 明	
<p>色分けを削除するとともに、DO ループの分割を行うことで再帰参照を回避しスレッド並列を可能とした。  サブルーチン “SIGMARIGHTT” 及び “SIGMARIGHTT_sa” にも同様の変更を加えれば、インデックステーブル “isigma1” 及び “JTMP1” を新規作成しなくても “jsigma1” 及び “ITMP1” の再利用が可能である。</p>	
変 更 内 容 リ ス ト	
変 更 前	変 更 後
<pre> subroutine sigmaright(ilevel) : if(ilevel.eq.1) then jmin=1 else if(mcl_lu1(ilevel-1) .le. 0) return jmin=jsigma1(mcl_lu1(ilevel-1), ilevel-1)+1 end if DO 10 ICOLOR=1, MCL_lu1(ilevel) jmax=jsigma1(icolor, ilevel) *VOCL LOOP, NOVREC CDIR\$ IVDEP *VDIR NODEP DO 1100 J=jmin, JMAX IEDGE=ITMP1(J) : pole(mq1_pl, ia)=pole(mq1_pl, ia) &amp; -delf1-specr*deltr : pole(mq6_pl, ia)=pole(mq6_pl, ia) &amp; -delf6-specr*deltrt 1100 continue jmin=jmax+1 10 continue </pre>	<pre> subroutine sigmaright(iplane, ilevel) : if(ilevel.ne.1 .and. &amp; mcl_lu1(ilevel-1) .le. 0) return jmin=isigma1(ilevel-1)+1 jmax=isigma1(ilevel) : DO 100 J=jmin, JMAX IEDGE=JTMP1(J) : EDG_WK(1, J)=delf1+specr*deltr : EDG_WK(6, J)=delf6+specr*deltrt 100 continue c DO 110 IIP=NDSIGMA(IPLANE-1)+1, &amp; NDSIGMA(IPLANE) IP =JTMP1(IIP) NEDG=IEDND_TBL(0, IP) DO 111 IE=1, NEDG IEDGE=IEDND_TBL(IE, IP) IEDGA=ABS(IEDGE) ITP =MARKEDG1(IEDGA) IF( IEDGE*ITP.LE.0 ) THEN J=ITMPE(IEDGA) POLE(MQ1_PL, IP)=POLE(MQ1_PL, IP) &amp; -EDG_WK(1, J) : POLE(MQ6_PL, IP)=POLE(MQ6_PL, IP) &amp; -EDG_WK(6, J) &amp; END IF 111 CONTINUE 110 CONTINUE </pre>



表 B.2.7 サブルーチン “SIGMAPREPARE” の変更内容 (続く)

File 名	lut2.f
Subroutine 名	SIGMAPREPARE
変 更 内 容 の 説 明	
<p>色分けの削除及び DO ループの分割を行うために必要な各種インデックステーブルの設定を追加した。  サブルーチン “SIGMALEFTT”, “SIGMALEFTT_sa”, “SIGMARIGHTT” 及び “SIGMARIGHTT_sa” にも同様の変更を加えた場合には、インデックステーブル “isigma”, “JTMP1”, “isigma1” 及び “JTMPr” を新規作成しなくても “jsigma”, “ITMP1”, “jsigma1” 及び “ITMPr” の設定方法を「変更内容リスト」に示す方法に変更すればよい。</p>	
配 列 名	説 明
isigma(0:imax_plane)	ハイパー面 $N_p$ が含む辺の数の最大値 $nd_{max}$ (前進スイープ用) 但し、辺の数はハイパー面を超えた累計 $nd_{max} = \text{isigma}(N_p)$
JTMP1(0:imax_ed)	ハイパー面 $N_p$ に属する $j$ 番目 ( $j$ はハイパー面を越えて通番) の辺の番号 $ed$ (前進スイープ用) $ed = \text{JTMP1}(j)$
ITMPw(0:imax_ed)	ハイパー面 $N_p$ に属する $j$ 番目 ( $j$ はハイパー面を越えて通番) の辺の番号 $ed$ (前進スイープ用) インデックステーブル “JTMP1” の逆テーブル $j = \text{ITMPw}(ed)$
isigma1(0:imax_plane)	ハイパー面 $N_p$ が含む辺の数の最大値 $nd_{max}$ (後退スイープ用) 但し、辺の数はハイパー面を超えた累計 $nd_{max} = \text{isigma}(N_p)$
JTMPr(0:imax_ed)	ハイパー面 $N_p$ に属する $j$ 番目 ( $j$ はハイパー面を越えて通番) の辺の番号 $ed$ (後退スイープ用) $ed = \text{JTMPr}(j)$
ITMPe(0:imax_ed)	ハイパー面 $N_p$ に属する $j$ 番目 ( $j$ はハイパー面を越えて通番) の辺の番号 $ed$ (後退スイープ用) インデックステーブル “JTMPr” の逆テーブル $j = \text{ITMPe}(ed)$
JTMP(0:imax_plane)	ハイパー面 $N_p$ に属する $i$ 番目 ( $i$ はハイパー面を越えて通番) の節点番号 $ip$ $ip = \text{JTMP}(i)$
ndsigma(0:imax_plane)	ハイパー面 $N_p$ が含む節点の数の最大値 $nd_{max}$ 但し、節点の数はハイパー面を超えた累計 $nd_{max} = \text{NDSIGMA}(N_p)$

表 B.2.7 サブルーチン “SIGMAPREPARE” の変更内容 (続き)

File 名	lut2.f
Subroutine 名	SIGMAPREPARE
変 更 内 容 リ ス ト	
(サブルーチン“SIGMAPREPARE”の最後に以下の処理を追加した)	
<pre> i sigma(0)=0 J      =0 do i level=1, numplane-1   DO IEDGE=1, N_ED_F     itp=abs(markedg(iedge))     IF(itp.eq. i level) THEN       J      =J+1       JTMP1(J)  =IEDGE       ITMPw(IEDGE)=J     END IF   END DO   i sigma(i level)=j end do c i sigma1(0)=0 J      =0 do i level=1, numplane-1   DO IEDGE=1, N_ED_F     itp=abs(markedg1(iedge))     IF(itp.eq. i level) THEN       J      =J+1       JTMP1(J)  =IEDGE       ITMPe(IEDGE)=J     END IF   END DO   i sigma1(i level)=j end do c ndsigma(0)=0 j      =0 do i plane=1, numplane   do i=1, n_p_f     itp =marknd1(i)     itp1=ids_nd(i16_nd, i)     if(itp.eq. i plane. and. itp1.lt. 10) then       j      =j+1       JTMP(j)=i     end if   end do   ndsigma(i plane)=j end do </pre>	

表 B.2.8 インクルードファイル “DS3D” の変更内容

File 名	DS3D
Subroutine 名	<i>include file</i>
変更内容の説明	
色分けを削除し、DO ループの分割に必要な各種インデックステーブルの設定を追加した。	
変更内容リスト	
省略（「表 B.1.31 インクルードファイル “DS3D” の変更内容」に同じ）	

表 B.2.9 インクルードファイル“LU”の変更内容

File 名	LU	
Subroutine 名	include file	
変 更 内 容 の 説 明		
色分けを削除し、DO ループの分割に必要な各種インデックステーブルの配列宣言を追加した。		
配 列 名	説 明	
isigma(0:imax_plane)	ハイパー面 $N_p$ が含む辺の数の最大値 $nd_{max}$ (前進スイープ用) 但し、辺の数はハイパー面を超えた累計 $nd_{max} = \text{isigma}(N_p)$	
JTMP1(0:imax_ed)	ハイパー面 $N_p$ に属する $j$ 番目 ( $j$ はハイパー面を越えて通番) の辺の番号 $ed$ (前進スイープ用) $ed = \text{JTMP1}(j)$	
ITMPw(0:imax_ed)	ハイパー面 $N_p$ に属する $j$ 番目 ( $j$ はハイパー面を越えて通番) の辺の番号 $ed$ (前進スイープ用) インデックステーブル” JTMP1” の逆テーブル $j = \text{ITMPw}(ed)$	
isigma1(0:imax_plane)	ハイパー面 $N_p$ が含む辺の数の最大値 $nd_{max}$ (後退スイープ用) 但し、辺の数はハイパー面を超えた累計 $nd_{max} = \text{isigma1}(N_p)$	
JTMPr(0:imax_ed)	ハイパー面 $N_p$ に属する $j$ 番目 ( $j$ はハイパー面を越えて通番) の辺の番号 $ed$ (後退スイープ用) $ed = \text{JTMPr}(j)$	
ITMPe(0:imax_ed)	ハイパー面 $N_p$ に属する $j$ 番目 ( $j$ はハイパー面を越えて通番) の辺の番号 $ed$ (後退スイープ用) インデックステーブル” JTMPr” の逆テーブル $j = \text{ITMPe}(ed)$	
JTMP(0:imax_plane)	ハイパー面 $N_p$ に属する $i$ 番目 ( $i$ はハイパー面を越えて通番) の節点番号 $ip$ $ip = \text{JTMP}(i)$	
ndsigma ( 0 : imax _ plane)	ハイパー面 $N_p$ が含む節点の数の最大値 $nd_{max}$ 但し、節点の数はハイパー面を超えた累計 $nd_{max} = \text{NDSIGMA}(N_p)$	
変 更 内 容 リ ス ト		
変 更 前	変 更 後	
common/sig/ jsigma (64, 2000) , + jsigma1 (64, 2000) ,  + itmpl ( imx_ed) , + itmpr ( imx_ed)	common/sig/ jsigma (64, 2000) , + jsigma1 (64, 2000) , + isigma (0: imx_plane) , + isigma1 (0: imx_plane) , + itmpl ( imx_ed) , + itmpr ( imx_ed) , + jtmp1 ( imx_ed) , + jtmp1 ( imx_ed) , + itmpw ( imx_ed) , + itmpe ( imx_ed) , + jtmp ( imx_nd) , + ndsigma (0: imx_plane)	

## APPENDIX C. 要素と辺及び節点の関係

JTAS では、ある要素が含む辺及び節点とその要素のどの部分を構成するものであるかについて幾何学的な仮定をおいている。要素が含む辺及び節点の番号は、その仮定の下に配列に保持されている。ここでは、その仮定について説明する。

図 C.1 に三角錐要素における辺及び節点との関係を示す。三角錐要素において辺 4 は必ず辺 1 の反対側、即ち辺 1 と共有する節点を持たない辺でなければならない。同様に辺 5 は辺 2 の、辺 6 は辺 3 のそれぞれ反対側の辺でなければならない。節点 1 及び 2 は辺 1 に含まれ、節点 3 及び 4 は辺 4 に含まれなければならない。三角錐要素が含む辺の番号は配列 “IDS\_EL” に保持されており、辺が含む節点の番号は配列 “IDS\_ED” に保持されている。この要素と辺及び節点の関係から三角錐要素 IELM に含まれる 4 つの節点 N1, N2, N3 及び N4 は、リスト C.1 に示すように辺 1 (IED1) 及び 4 (IED4) を介して全て知ることが出来る。

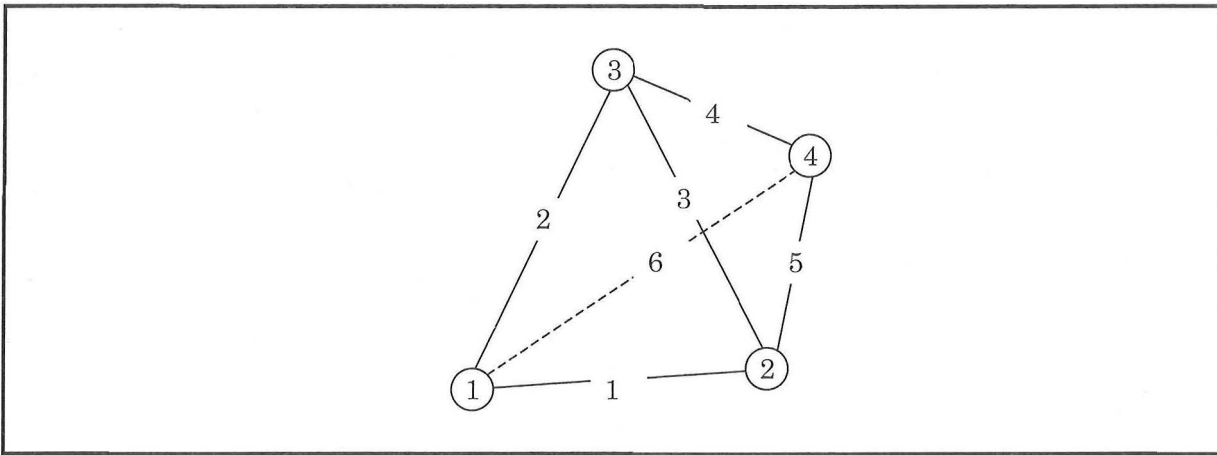


図 C.1 三角錐要素における辺及び節点との関係

リスト C.1 三角錐要素に含まれる全ての節点

```

IED1=IDS_EL(IE1_EL, IELM)
IED4=IDS_EL(IE4_EL, IELM)
N1 =IDS_ED(IN1_ED, IED1)
N2 =IDS_ED(IN2_ED, IED1)
N3 =IDS_ED(IN1_ED, IED4)
N4 =IDS_ED(IN2_ED, IED4)

```

図 C.2 に三角柱要素における辺及び節点との関係を示す。三角柱要素において辺 1, 2 及び 3 が一方の底面を構成し、辺 4, 5 及び 6 が他方の底面を構成しなければならない。そして辺 1 と辺 4, 辺 2 と辺 5, 辺 3 と辺 6 がそれぞれの底面の対応する辺を構成していなければならない。辺 7, 8 及び 9 は三角柱の側面を構成する辺であり、辺 7 は辺 1 と辺 3 が共有する節点 1 と辺 4 と辺 6 が共有する節点 4 を結ぶ辺でなければならない。同様に辺 8 は、辺 1 と辺 2 が共有する節点 2 と辺 4 と辺 5 が共有する節点 5 を結ぶ辺でなければならず、辺 9 は、辺 2 と辺 3 が共有する節点 3 と辺 5 と辺 6 が共有する節点 6 を結ぶ辺でなければならない。そして、辺 7, 8 及び 9 の終端は必ず同じ方向でなければならない。三角柱要素が含む辺の番号は配列 “JPRISM” に保持されており、辺が含む節点の番号は配列 “IDS\_ED” に保持されている。この要素と辺及び節点の関係から三角柱要素 IELM に含まれる 6 つの節点 N1 から N6 は、リスト C.2 に示すように辺 7 (IED7), 8 (IED8) 及び 9 (IED9)

を介して全て知ることが出来る．三角柱要素における勾配は，節点1，2，3及び5を4つの頂点とする三角錐，節点1，3，5及び6を頂点とする三角錐，節点1，4，5及び6を頂点とする三角錐の3つに分割されて計算される．

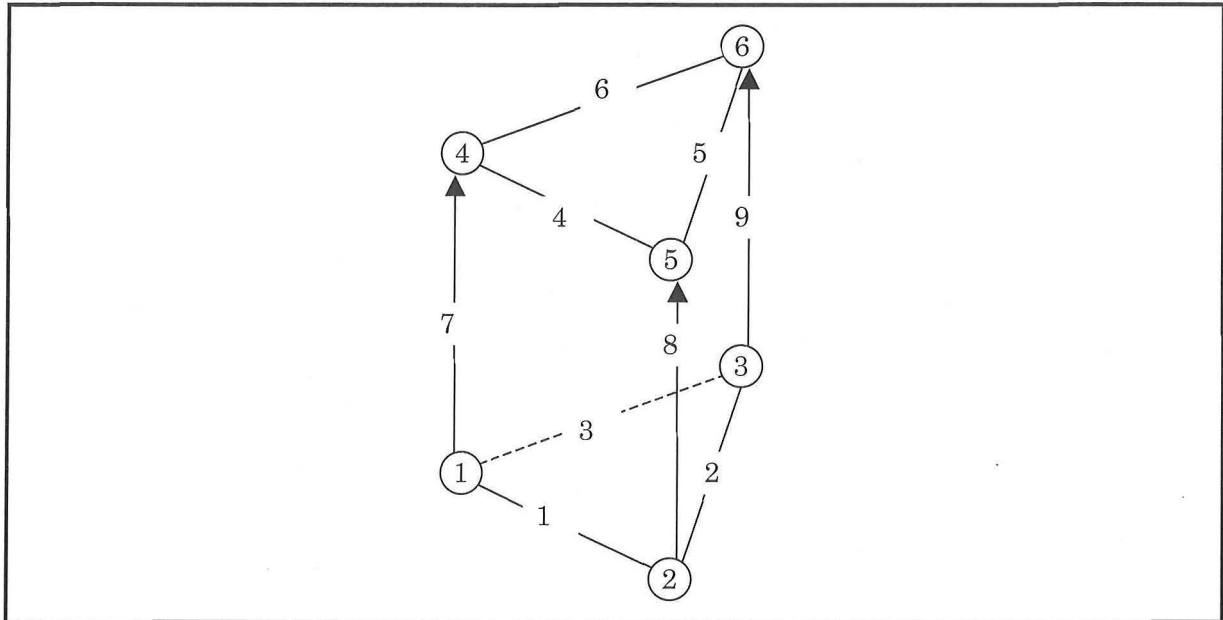


図 C.2 三角柱錐要素における辺及び節点との関係

リスト C.2 三角柱錐要素に含まれる全ての節点

```

IED7=JPRISM(7, IELM)
IED8=JPRISM(8, IELM)
IED9=JPRISM(9, IELM)
N1  =IDS_ED(IN1_ED, IED7)
N2  =IDS_ED(IN2_ED, IED7)
N3  =IDS_ED(IN1_ED, IED8)
N4  =IDS_ED(IN2_ED, IED8)
N5  =IDS_ED(IN1_ED, IED9)
N6  =IDS_ED(IN2_ED, IED9)

```

図 C.3に四角錐要素における辺及び節点との関係を示す．四角錐要素において辺1，2，3及び4が底面を構成し，辺5，6，7及び8が側面を構成しなければならない．辺5は辺1と辺4が共有する節点1と四角錐の頂点を構成する節点5を結ぶ辺でなければならない．同様に辺6は，辺1と辺2が共有する節点2と頂点を構成する節点5を結ぶ辺でなければならず，辺7は，辺2と辺3が共有する節点3と頂点5を，辺8は辺3と辺4が共有する節点4と頂点5を，それぞれ結ぶ辺でなければならない．四角錐要素が含む辺の番号は配列“JPYR”に保持されている．また，四角錐要素が含む節点の番号は，他の要素の場合と異なり辺を介することなく配列“IPYR”に直接保持されている（リスト C.3）．四角錐要素における勾配は，節点1，2，3及び5を4つの頂点とする三角錐と節点1，3，4及び5を頂点とする三角錐との2つに分割されて計算される．

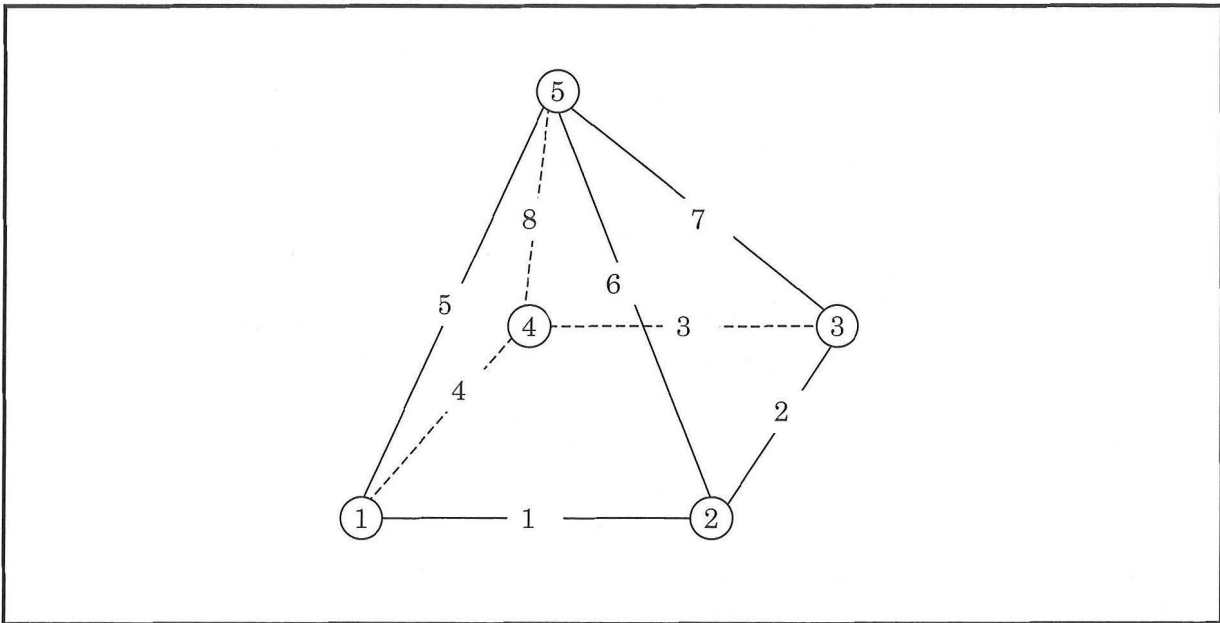


図 C.3 四角錐要素における辺及び節点との関係

リスト C.3 四角錐要素に含まれる全ての辺及び節点

```

IED1=JPYR (1, IELM)
IED2=JPYR (2, IELM)
IED3=JPYR (3, IELM)
IED4=JPYR (4, IELM)
IED5=JPYR (5, IELM)
IED6=JPYR (6, IELM)
IED7=JPYR (7, IELM)
IED8=JPYR (7, IELM)
N1  =IPYR (1, IELM)
N2  =IPYR (2, IELM)
N3  =IPYR (3, IELM)
N4  =IPYR (4, IELM)
N5  =IPYR (5, IELM)
    
```

## APPENDIX D. 実行スクリプト

ここでは、JTAS による性能測定をより容易に実施することを目的に作成した実行用シェルスクリプトについて簡単に説明する。

リスト D.1 に作成した実行用シェルスクリプトのリストを示す。この実行用シェルスクリプトは、表 D.1 に示す環境変数に適宜値を設定した後に実行することにより、計算条件ファイルを生成し JTAS を実行した後、計算結果を指定されたディレクトリに保存する。環境変数を設定し、この実行用シェルスクリプトを起動するシェルスクリプトの例をリスト D.2 に示す。実行は、TSS による会話型実行と NQS (Network Queuing System) によるバッチ処理のいずれかを選択することが可能である。表 D.2 に実行用シェルスクリプトで使用される主なディレクトリ及びファイルについてその一覧を示す。

尚、この実行用シェルスクリプトによる JTAS の実行においては、生成される計算条件ファイル、JTAS が使用するファイル “fort. nn”，標準出力に出力した内容を保存するファイル（環境変数 “\$STDOUT” に設定、現在の値は “output”）及び NQS の標準出力を保存するファイル（環境変数 “\$NQSOUT” に設定、現在の値は “stdout.\$VERSION”）のファイル名が固定されている。従って、同じ実行ディレクトリで複数のジョブを同時に実行することは出来ないので注意されたい。



表 D.1 実行用シェルスクリプトを起動する際に指定する環境変数一覧

環境変数	説明
TEST_MOD	<p>“YES” を設定すると実行されるコマンドのリストが標準出力に表示される。 この時、実行はされない。 “YES” 以外を設定すると実行が行われる。</p>
PROC_NUM	MPI 並列におけるプロセス数を指定する。
VERSION	<p>実行可能モジュールのバージョンを指定する。 実行可能モジュール名は、“\$BIN_NAM\$DOT\$VERSION” となる。 ここで、環境変数 “\$BIN_NAM” には現在 “jtas” が設定されている。また、環境変数 “\$DOT” は、環境変数 “\$VERSION” がヌルの場合ヌルであり、ヌル以外の場合には、“.” (ドット) が設定される。</p>
CASE	<p>計算ケースの名称を設定する。 格子データ等 JTAS の入力データが存在するディレクトリパスの一部として使用される。</p>
DATA_SAVE	<p>計算結果を保存するか否かを指定する。 “YES” を設定した場合、計算結果は環境変数 “\$SAVE_DIR” で指定されるディレクトリに保存される。 現在、環境変数 “\$SAVE_DIR” には、“\$EXEC_DIR/RUN\$DOT\$VERSION” が設定されている。 “YES” 以外を設定した場合、計算結果は環境変数 “\$EXEC_DIR” で指定される JTAS が実行されたディレクトリに残される。</p>
TSS_MODE	<p>実行を TSS で行うか NQS によるバッチジョブとして行うかを指定する。 “YES” を設定した場合、TSS で実行される。 “YES” 以外を設定した場合、NQS のバッチジョブとして実行される。</p>
PROFILE	<p>プロファイラによる実行状況の解析を行うか否かを指定する。 “YES” を指定した場合、プロファイラによる解析に必要な環境変数を設定した後、JTAS の実行を行う。 “YES” 以外を指定した場合、環境変数の設定を行わずに実行を行う。 プロファイラによる解析を行う場合には、実行可能モジュールを作成する際にコンパイルオプション “-Ktl-trt” が指定されている必要がある。このコンパイルオプションが指定されずにコンパイルした実行可能モジュールを実行した場合、この環境変数に “YES” を設定してもプロファイラによる解析は行われない。 設定される環境変数については、表5.3を参照のこと。</p>
SUFFIX	<p>環境変数 “\$VERSION” を修飾するための環境変数。 一例として、同じバージョンの実行可能モジュールをプロファイラによる解析を行う場合と行わない場合で区別する時に使用する。 起動用シェルスクリプト (リスト D.2) において VERSION=\$VERSION\$SUFFIX として環境変数 “\$VERSION” を修飾している。</p>

表 D.2 実行用シェルスクリプトで 사용되는主なファイル及びディレクトリ

計算条件ファイル		
項目	環境変数	値
場所	\$EXEC_DIR	実行ディレクトリの項参照
ファイル名	\$DIMD_NAM	"DIMDAT"
非構造格子データファイル		
項目	環境変数	値
場所	\$DATA_DIR	"Dom\$PROC_ZP" 但し, \$PROC_ZP=nnn であり, nnn は 3 桁のゼロ パディングされた MPI プロセス数
ファイル名	\$DATA_NAM	"\$DATA_DIR/cube"
実行ディレクトリ		
項目	環境変数	値
場所	\$EXEC_DIR	"\$ENV_DIR/\$CODE_NAM/\$CASE/D\$PROC_ZP" 但し, \$ENV_DIR= "large 領域ディレクトリ/BMT" \$CODE_NAM= "jtas" \$CASE: 実行用シェルスクリプト起動時に設定 \$PROC_ZP=nnn であり, nnn は 3 桁のゼロパディングされた MPI プロセス数
実行可能モジュール		
項目	環境変数	値
場所	\$BIN_DIR	"\$SRC_DIR/\$CODE_NAM/bin" 但し, \$SCR_DIR= "\$HOME/BMT" \$CODE_NAM= "jtas"
モジュール名	\$BIN_NAM\$DOT\$VERSION	\$BIN_NAM=\$CODE_NAME (= "jtas") \$DOT=null \$VERSION=null の時 "." (ドット) ≠null の時 \$VERSION: 実行用シェルスクリプト起動時に設定
計算結果が保存されるディレクトリ		
項目	環境変数	値
場所	\$SAVE_DIR	"\$EXEC_DIR/RUN\$DOT\$VERSION" 但し, \$EXEC_DIR: 実行ディレクトリの項参照 \$DOT=null \$VERSION=null の時 "." (ドット) ≠null の時 \$VERSION: 実行用シェルスクリプト起動時に設定
JTAS 標準出力ファイル名	\$STDOUT	"output"
NQS 標準出力ファイル名	\$NQSOUT	"stdout \$DOT \$VERSION" 但し, \$DOT=null \$VERSION=null の時 "." (ドット) ≠null の時 \$VERSION: 実行用シェルスクリプト起動時に設定

## リスト D.1 実行用シェルスクリプト (1/8)

```
#!/bin/sh
#
# TEST MODE FLAG #####
#
TEST_MODE=${TEST_MODE:="NO"}
#
if [ "$TEST_MODE" = "YES" ]
then
    set -x
fi
#
# ENVIRONMENT VARIABLES REQUIRED SETTING AT EXECUTION TIME #####
#
# number of MPI process (no zero padding)
#
PROC_NUM=${PROC_NUM:="2"}
#
# version number of the programs
#
VERSION=${VERSION:=""}
#
# execution case (directory) name
#
CASE=${CASE:="case1"}
#
# saving or not result files
#
DATA_SAVE=${DATA_SAVE:="YES"}
#
# execution mode selector; ="YES": execute by TSS mode
#
#                               else : execute as NQS job
#
TSS_MODE=${TSS_MODE:="NO"}
#
# execute profiler or not; ="YES":    execute
#
#                               else : not execute
#
PROFILE=${PROFILE:="NO"}
#
# ENVIRONMENT VARIABLES REQUIRED SETTING AT INSTALLING TIME #####
#
# directory name contained installed source files
#
SRC_DIR="$HOME/BMT"
#
# directory name contained installed execution environment files
#
ENV_DIR="/large/l/138/BMT"
#
```

## リスト D.1 実行用シェルスクリプト (2/8)

```
#
# SYSTEM DEPENDENT PARAMETERS #####
# default file name of FORTRAN INPUT/OUTPUT
FFILES="fort"
#
# cpu time measurement function in "DIMDAT"
TIME_FNC="0          -itleft(=0:etime,=1:chktime(censs))"
#TIME_FNC="1          -itleft(=0:etime,=1:chktime(censs))"
#
# SET NUMBER OF MPI PROCESS WITH 3 DIGIT ZERO PADDING #####
PROC_ZP=`echo $PROC_NUM | awk '{printf "%03¥dn", $0}'`
#
# set dot "." discripeter, if the version number "VERSION" is not null
if [ "$VERSION" = "" ]
then
    DOT=
else
    DOT="."
fi
#
# SET ENVAIRONMENT VARIABLES #####
# installed code name (as directory name)
CODE_NAM="jtas"
#
# directory name for execution
EXEC_DIR="$ENV_DIR/$CODE_NAM/$CASE/D$PROC_ZP"
#
# file name of dimensional data
DIMD_NAM="DIMDAT"
if [ "$TEST_MODE" = "YES" ]
then
    DIMD_NAM="/dev/null"
fi
#
# directory name contained grid data et. al
DATA_DIR="Dom$PROC_ZP"
#
# base name of grid data files
DATA_NAM="$DATA_DIR/cube"
#
# directory name for saving result files
SAVE_DIR="$EXEC_DIR/RUN$DOT$VERSION"
#
# directory name of the executable binaly file
BIN_DIR="$SRC_DIR/$CODE_NAM/bin"
#
# file name of the executable binaly file
BIN_NAM="$CODE_NAM"
```

## リスト D.1 実行用シェルスクリプト (3/8)

```
#
# SET SOME COMMANDS #####
#
# null command should be do nothing
#
if [ "$TEST_MODE" = "YES" ]
then
    NULL_CMD="echo"
else
    NULL_CMD=""
fi
#
# name of qsub command
#
if [ "$TSS_MODE" = "YES" ]
then
    QSUB_CMD=""
else
    QSUB_CMD="/bin/qsub"
fi
if [ "$TEST_MODE" = "YES" ]
then
    QSUB_CMD="$NULL_CMD $QSUB_CMD"
fi
#
# name of MPI command
#
MPI_CMD="$NULL_CMD timex mpiexec"
#
# name of "mkdir" command
#
MKD_CMD="$NULL_CMD mkdir"
#
# name of "rm" command
#
RM_CMD="$NULL_CMD rm -f"
#
# name of "mv" command
#
MV_CMD="$NULL_CMD mv"
```

## リスト D.1 実行用シェルスクリプト (4 / 8)

```

#
# MAKE DIMENSIONAL DATA NAMED "DIMDAT" #####
#
PWD=`pwd`
cd $EXEC_DIR
#
# directory name for gridded data
#
GNAME=' "'"$DATA_NAM"' "'
CNAME=' "'"$DATA_NAM"' "'
#
# case dependent data
#
if [ "$CASE" = "case1" ]
    then
#
# for "case1"
#
INTG_STEP="1000"                : Auto stop steps"
BOND_TP01="2"                  -1 : boundary type of first boundary"
                                # 1: entry flow      condition
                                # 2: slip      boundary condition
                                # 3: exit flow      condition
                                # 5: symmetry boundary condition
                                # 22: nonslip boundary condition
                                #
TIME_LIMIT="1000"              # Job time [s]
CLDM_SWCH="0"                  ISWITCH(=1:on, =0:off)"
CLDM_TMNG="2"                  ISWITCH2(Computation timing)"
CLDM_AREA="0.0"                AREA"
CLDM_AL_="0.0"                AL"
CLDM_RLS_="0.0"                RLS"
CLDM_RLB_="0.0"                RLB"
CLDM_XMC_="0.0"                XMC"
CLDM_YMC_="0.0"                YMC"
CLDM_ZMC_="0.0"                ZMC"
#

```

リスト D.1 実行用シェルスクリプト (5/8)

```

#
# for "case2"
#
elif [ "$CASE" = "case2" ]
    then
INTG_STEP=" 500                : Auto stop steps"
BOND_TP01="22                 -1      : boundary type of first boundary"
TIME_LMIT="10000"            # Job time [s]"
CLDM_SWCH="1                 ISWITCH(=1:on, =0:off)"
CLDM_TMNG="1                 ISWITCH2(Computation timing)"
CLDM_AREA="1.0               AREA"
CLDM_AL__="0.0               AL"
CLDM_RLS_="1.0               RLS"
CLDM_RLB_="1.0               RLB"
CLDM_XMC_="0.0               XMC"
CLDM_YMC_="0.0               YMC"
CLDM_ZMC_="0.0               ZMC"
#
fi # if [ "$CASE" = "case name" ] ; then
#
cat << DATA_END > $DIMD_NAM
3      1      version edition
1      2      1      nns      nmuscl ntinteg
$GNAME      gname
$CNAME      cname
1.0000000000000000e+005 1.0000000000000000e+000: CFL number
20000                : Auto save interval
1.4000000000000000e+000 : Gamma
1                    : 0 = reading "fort.45"; 1 = init. cond.
1                    : 0 = reading "fort.9"; 1 = coloring for LUSGS
$INTG_STEP
1.0000000000000000e+006 : Reynolds number
2.7315000000000000e+002 : Freestream temperature
5.0000000000000000e-001 : Wall Temperature
8.0000000000000000e-001 : Freestream Mach number
6                    : boundary surfaces
$BOND_TP01
1      -2
3      -3
2      -4
2      -5
2      -6
1.0000000000000000e+000 -init. density
9.465727652959386e-001 -init. Ux (AOA = 0.0000000000000000e+000)
0.0000000000000000e+000 -init. Uy
0.0000000000000000e+000 -init. Uz
1.0000000000000000e+000 -init. pressure
2.0000000000000000e+001 -init. rt

```

## リストD.1 実行用シェルスクリプト (6/8)

```

2          ----- boundary surface number
1.000000000000000e+000          -init. density
9.465727652959386e-001          -init. Ux (AOA = 0.000000000000000e+000)
0.000000000000000e+000          -init. Uy
0.000000000000000e+000          -init. Uz
1.000000000000000e+000          -init. pressure
2.000000000000000e+001          -init. rt
0          ----- termination
0
$TIME_LMIT
0          : 0 = laminar; 1 = turbulent
$CLDM_SWCH
$CLDM_TMNG
$CLDM_AREA
$CLDM_AL_
$CLDM_RLS_
$CLDM_RLB_
$CLDM_XMC_
$CLDM_YMC_
$CLDM_ZMC_
$TIME_FNC
100          -tileft(Time left for output and end)
DATA_END
#
# cd $PWD
# SET NQS PARAMETERS #####
# file name for standard output of FORTRAN program
#   STDOUT="output"
# file name for standard output of NQS system
#   NQSOUT="stdout$DOT$VERSION"
# queue name of the NQS job
#   QUE_NAME="QJOB"
# job name used NQS jobs
#JOB_NAME="$CASE$DOT${VERSION}_$PROC_ZP"
#   JOB_NAME="jtascode.01-01"          # conform to CeNNS naming rule
#           # column 1 : alphabet
#           #       2-8: alpha-numeric
#           #       9 : dot "."
#           #       10-11: job step number N (numeric)
#           #       12 : hyphen
#           #       13-14: total job steps M (numeric)
#           #           N <= M
# memory size limit
#   MEM_SIZE="2048mb"
# cpu time limit
#   if [ "$TIME_LMIT" = "" ]
#       then
#           CPU_LMIT="1500"
#       else
#           CPU_LMIT=`expr $TIME_LMIT + 300`
#   fi

```



## リスト D.1 実行用シェルスクリプト (7/8)

```

#
# SET COMMANDS FOR SAVING RESULT DATA FILES #####
#
if [ "$DATA_SAVE" = "YES" ]
then
#
# make directory if not exist
if [ !-d $SAVE_DIR ]
then
$MKD_CMD $SAVE_DIR
fi
SAVE_CMD1="$MV_CMD $EXEC_DIR/$STDOUT $SAVE_DIR"
SAVE_CMD2="$MV_CMD $EXEC_DIR/$FFILES '*' $SAVE_DIR"
SAVE_CMD3="$MV_CMD ${DATA_NAM}'*' .rslt $SAVE_DIR"
SAVE_CMD4="$MV_CMD ${DATA_NAM}'*' .mark $SAVE_DIR"
if [ "$PROFILE" = "YES" ]
then
SAVE_CMD5="$RM_CMD $SAVE_DIR/'*' .pri"
SAVE_CMD6="$MV_CMD $EXEC_DIR/'*' .pri $SAVE_DIR"
else
SAVE_CMD5=
SAVE_CMD6=
fi
else
SAVE_CMD1=
SAVE_CMD2=
SAVE_CMD3=
SAVE_CMD4=
SAVE_CMD5=
SAVE_CMD6=
fi
#
# SET ENVIRONMENT VARIABLES FOR PROFILER #####
#
if [ "$PROFILE" = "YES" ]
then
TRT_ENV__=' eval TRT_ENV="PMP=on" ; export TRT_ENV'
PROF_STATS=' eval PROF_STATS="7" ; export PROF_STATS'
PROF_PA__=' eval PROF_PA="cov, ins, sta" ; export PROF_PA'
PROF_MEM__=' eval PROF_MEMORY_SIZE=512000 ; export PROF_MEMORY_SIZE'
else
TRT_ENV__=' #'
PROF_STATS=' #'
PROF_PA__=' #'
PROF_MEM__=' #'
fi

```

## リスト D.1 実行用シェルスクリプト (8/8)

```

#
# EXECUTION #####
# EXECUTION BY TSS MODE
if [ "$TSS_MODE" = "YES" ]
then
    cd $EXEC_DIR
    if [ "$PROFILE" = "YES" ]
    then
        $TRT_ENV__
        $PROF_STATS
        $PROF_PA__
        $PROF_MEM__
    fi
    $MPI_CMD -n $PROC_NUM $BIN_DIR/$BIN_NAME$VERSION
#
    $SAVE_CMD1
    $SAVE_CMD2
    $SAVE_CMD3
    $SAVE_CMD4
    $SAVE_CMD5
    $SAVE_CMD6
# EXECUTION AS NQS JOB
else
    $QSUB_CMD << SCRIPT_END

#!/bin/sh
#@%$-s /bin/sh
#@%$-q $QUE_NAME
#@%$-r $JOB_NAME
#@%$-IP $PROC_NUM
#@%$-lp 1
#@%$-IM $MEM_SIZE
#@%$-IT $CPU_LIMIT
#@%$-o $EXEC_DIR/$NQSOUT
#@%$-oi
#@%$-eo
$TRT_ENV__
$PROF_STATS
$PROF_PA__
$PROF_MEM__
cd $EXEC_DIR
$MPI_CMD -n $PROC_NUM $BIN_DIR/$BIN_NAME$VERSION > $STDOUT
$SAVE_CMD1
$SAVE_CMD2
$SAVE_CMD3
$SAVE_CMD4
$SAVE_CMD5
$SAVE_CMD6
SCRIPT_END
#
fi
#
exit

```

## リスト D.2 実行用シェルスクリプト起動シェルスクリプト例

```
#!/bin/sh
#
# TEST MODE FLAG #####
#
TEST_MODE=
#
# PARAMETERS REQUIRED AT EXECUTION TIME #####
#
# number of MPI process (no zero padding)
#
PROC_NUM=2
#
# version number of the programs
#
VERSION="v01.26"
#
# execution case (directory) name
#
CASE="case2"
#
# saving or not result files; ="YES" or NULL : will be saved
#                               else           : not   saved
#
DATA_SAVE="YES"
#
# execution mode selector; ="YES" : execute by TSS mode
#                               else : execute as NQS job
#
TSS_MODE=
#
# execute profiler or not; ="YES" :   execute
#                               else : not execute
#
PROFILE=
#
# suffix of executable for profiling compiled with-Ktl_trt
#
SUFFIX=
#
# modify version number, if profiling is efficient
#
if [ "$PROFILE"="YES" ]
then
VERSION=$VERSION$SUFFIX
fi
#
. ./goscript
#
exit
#
```

## APPENDX E. メモリ使用量

表 E.1 に計算に要したメモリの使用量を示す。このデータは、ジョブ終了後に標準出力に出力されるセンター固有情報を元としている。“Total” は、全プロセスの合計メモリ使用量の最大値である。“Root proc.” は、MPI のルートプロセス（ランク 0）が使用したメモリ量であり、上段が平均使用量（Average）、下段が最大使用量（Max.）である。“Others” は、ルートプロセス以外のプロセスが使用したメモリ量であり、ルートプロセスの場合同様、上段が平均使用量、下段が最大使用量である。ルートプロセス以外のプロセスにおいては、メモリ使用量は全て同じであった。

表 E.1 メモリ使用量

メモリ使用量 [MB]							上段：Average 下段：Max.
オリジナル							
Process 数	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.	
Total	1088	1984	3776	7360	14528	28864	
Root proc.	574	574	572	573	570	562	
	640	640	640	640	640	640	
Others	447	447	446	446	444	376	
	448	448	448	448	448	448	
スカラ版							
Process 数	02Proc.	04Proc.	08Proc.	16Proc.	32Proc.	64Proc.	
Total	1600	3008	5824	11456	22720	45248	
Root proc.	829	829	826	826	819	806	
	896	896	896	896	896	896	
Others	703	702	701	699	694	625	
	704	704	704	704	704	704	
スレッド版							
Process 数	Proc. only	01Thread	02Thread	04Thread	08Thread	16Thread	
Total	1856	1856	1856	1856	1984	2112	
Root proc.	956	956	951	949	1005	1060	
	1024	1024	1024	1024	1088	1152	
Others	831	831	830	830	892	954	
	832	832	832	832	896	960	

## 参考文献

- [1] Iwamiya, T., “NAL SST Project and Aerodynamic Design of Experimental Aircraft”, Proceedings of the 4th ECCOMAS Computational Fluid Dynamics Conference, Wiley, Chichester, England, U.K., pp. 580–585, 1998.
- [2] 高木正平, 坂田公夫 他., “[特集] 超音速実験機計画について”, 日本流体力学会誌ながれ18-5, pp.275–307, 1999.
- [3] 藤田健, 松島紀左, 中橋和博., “非構造格子 CFD を用いた逆問題設計システムの高度化”, 第15回数値流体力学シ

ンポジウム予稿集 D05-3, 2001.

- [4] 高橋克倫, 藤田健 他., “NAL 小型超音速実験機 NEXST-1 の結合分離金具形状修正の CFD 解析”, 第17回数値流体力学シンポジウム予稿集 F2-3, 2003.
- [5] “小型超音速実験機 (NEXST-1) の舵角変化時における空力特性変化の数値解析”  
[http://www.ista.jaxa.jp/res/c02/a06\\_01.html](http://www.ista.jaxa.jp/res/c02/a06_01.html)
- [6] Nakahashi, K., Ito, Y., and Togashi, F., “Some challenges of realistic flow simulations by unstructured grid CFD”, *Int. J. for Numerical Methods in Fluids*, Vol.43, pp.769-783, 2003.
- [7] “並列型の非構造格子ソルバプログラム ユーザーズマニュアル” 平成15年2月28日 財団法人青葉工業振興会.
- [8] Obayashi, S. and Guruswamy, G. P., “Convergence Acceleration of a Navier-Stokes Solver for Efficient Static Aeroelastic Computation”, *AIAA Journal*, Vol. 33, No. 6, pp. 1134-1141, 1995.
- [9] Venkatakrishnan V., “On the Accuracy of Limiters and Convergence to Steady State Solutions.”, *AIAA Paper*, 93-0880, 1993.
- [10] Sharov, D. and Nakahashi, K., “Reordering of 3-D Hybrid Unstructured Grids for Vectorized LU-SGS Navier-Stokes Computations”, *AIAA 97-2102*, 1997.
- [11] Sharov, D. and Nakahashi, K., “Reordering of 3-D Hybrid Unstructured Grids for Lower-Upper Symmetric Gauss-Seidel Computations”, *AIAA Journal*, Vol. 36, No. 3, 1998
- [12] Jameson, A. and Turkel, E., “Implicit Scheme and LU Decompositions”, *Mathematics of Computations*, Vol. 37, No. 156, pp. 385-397, 1981.
- [13] Men'shov, I. and Nakamura, Y., “Implementation of LU-SGS Method for an Arbitrary Finite Volume Discretization”, *Proc. of Japanese 9th CFD Symposium*, pp. 123-124, 1995.
- [14] Sharov, D. , Luo, H. and Baum. J. D., “Implementation of Unstructured Grid GMRES+LU-SGS Method on Shared-Memory, Cache-based parallel Computers.”, *AIAA 2000-0927*, 2000.
- [15] METIS - Family of Multilevel Partitioning Algorithms  
<http://glaros.dtc.umn.edu/gkhome/views/metis>
- [16] 中橋和博, 藤田健, “非構造格子法に基づく流体数値計算法の並列化”, 平成13年度東北大学情報シナジーセンター共同研究成果, 2001.
- [17] Fujita, T, et. al, “Evaluation of Parallelized Unstructured-grid CFD for Aircraft Applications”, *Proc. of Parallel CFD 2002*.
- [18] Goldberg. U. and Ramakrishnan. V., “A Pointwise Version of Baldwin-Barth Turbulence Model.”, *Comp. Fluid Dyn.*, 1, 321-338, 1993.
- [19] pallart, P. R. and Allmaras, S. R., “A One-Equation Turbulence Model for Aerodynamic Flows.”, *AIAA Paper 92-0439*, 1992.
- [20] Princeton Ocean Model (POM)  
<http://www.aos.princeton.edu/WWWPUBLIC/htdocs.pom/index.html>



宇宙航空研究開発機構研究開発報告 JAXA-RR-06-004

---

発行日 平成 18 年 11 月 30 日  
編集・発行 宇宙航空研究開発機構  
〒182-8522 東京都調布市深大寺東町 7-44-1  
URL : <http://www.jaxa.jp/>  
印刷・製本 藤原印刷株式会社

---

本書及び内容についてのお問い合わせは、下記にお願いいたします。  
宇宙航空研究開発機構 情報システム部 研究開発情報センター  
〒305-8505 茨城県つくば市千現 2-1-1  
TEL : 029-868-2079 FAX : 029-868-2956

---

©2006 宇宙航空研究開発機構

※本書の一部または全部を無断複写・転載・電子媒体等に加工することを禁じます。

