擬似スペクトル法を用いた乱流場の直接数値シミュレーションの並列化と性能評価

高田雅美[†] 山本義暢[††] 庄野逸[†,☆]
功刀資彰[††] 城和貴[†]

自由界面における乱流を計算機で解析する場合，直接数値シミュレーションと呼ばれる手法により数値解を求める場合がある．この数値解の精度をあげるためには，しばしば擬似スペクトル法が用いられる．本研究では，擬似スペクトル法を用いた乱流解析プログラムを分散メモリ環境対象の並列プログラムに変換し，その性能評価を行った．擬似スペクトル法は，流体の時間発展方程式を解く際，フーリエ変換を用いて波数空間で計算を行う．この際，通常の並列化で使われる strip mining による実空間の均等分割によって配列データの分割を行った場合，通信時間と通信オーバヘッドが膨大になる可能性がある．本論文では，この擬似スペクトル法を用いた乱流解析プログラムを並列化するために，通信時間と通信オーバヘッドを考慮し，複数の配列を処理する際に，1 つの配列がなるべく少数のプロセッサに割り当てられるように分割し，なおかつ，複数の配列の演算が同時処理できるような方針で実装を行った．提案する並列化手法の有効性を検討するために，並列プログラムを Message Passing Interface を用いて実装し，安価な分散メモリ環境において性能評価を行った．その結果，現実的な実行時間で解が得られることを確認した．

Parallelization and Evaluation of a Direct Numerical Simulation for a Turbulent Flow Using Pseudo Spectral Method

Masami Takata,[†] Yoshinobu Yamamoto,[††] Hayaru Shouno,[†,☆]
Tomoaki Kunugi[††] and Kazuki Joe[†]

A direct numerical simulation (*DNS*) is used for an analysis of a free-surface turbulent flow. To increase the accuracy of this simulation, sometimes pseudo-spectral method (*PSM*) is applied. In this paper, we developed and evaluated a parallel program for the *DNS* program with *PSM* in distributed memory environments. Applying *PSM* means that the calculation is carried out in the frequency domain, so that, the conventional parallelization method for the *DNS* by dividing a loop structure in the calculation homogeneously, that is called strip mining, in spatial domain may require a lot of communication time and communication overhead. In this paper, we propose an improving method for parallelization: In the case of carrying out plural processes with several arrays, a process for an array is assigned to as few processors as possible, and plural processes are carried out as simultaneous as possible. We implemented the parallel program for the *DNS* with Message Passing Interface. As a result, we obtained the *DNS* solution in practical time with an entry level distributed computing system.

1. はじめに

自由表面を有する乱流場は，原子炉，核融合炉・化学プラント等の広範囲な工学分野，さらには海洋・河川等の自然環境内において，特定の条件下で，頻繁に出現する．この乱流構造およびそれに付随する熱物質輸送を解析することは，工学上重要である．これらの乱流場は 3 次元・非定常かつ非線形性が強く，その解析解を求めることが非常に困難である．このような乱流場を解析する手法として，理論に基づく手法，実験に基づく手法，流体運動を記述する方程式の数値解を計算機上で求める直接数値シミュレーション（*DNS*：Direct Numerical Simulation）がある[6),7)]．

DNS で扱う乱流場は非定常の 3 次元上のベクトル場で表現されているため，計算すべき格子点の数はス

† 奈良女子大学大学院人間文化研究科
Graduate School of Human Culture, Nara Women's University
†† 京都大学大学院工学研究科
Department of Nuclear Engineering, Kyoto University
☆ 現在，山口大学工学部
Presently with Faculty of Engineering, Yamaguchi University

ケールの 3 乗に比例して大きくなる．このため，大規模で精密なシミュレーションを高速に行うためには大規模計算可能な高速演算ユニットと大規模メモリが必要となる．我々は，この問題を解決するために，乱流場の *DNS* 計算のための逐次プログラムを並列化することによって，高速化と大規模メモリの確保を実現するというアプローチをとった．

現在，数値計算を実行する並列計算機環境としては 2 種類の環境が考えられる．1 つは複数のプロセッサがメモリを共有する共有メモリ方式と呼ばれるアーキテクチャであり，もう 1 つは複数のプロセッサがそれぞれ固有のメモリを持つ分散メモリ方式と呼ばれるアーキテクチャである．大規模計算の計算時間のみを分散化させることが目的であるならば，並列化にともなう通信等のオーバヘッドが生じにくい共有メモリ方式を対象とした並列プログラムが有効である．しかし，本研究のように，大規模なメモリも必要とする場合，共有メモリにデータが収まらないことがある．このような場合，計算時間のみならず計算データも分散しなければならず，データの分散が可能な分散メモリ型並列計算機に対応した並列プログラムの開発が必要となる．

計算データを分散化させる場合，考慮すべき点は，データを交換するための通信コストとそれによって生じる通信オーバヘッドである．本研究で取り上げた乱流場の *DNS* 計算は数値不安定性を回避し高精度な解を得るために，時間発展を擬似スペクトル法（*PSM*：Pseudo Spectral Method）を用いて波数空間上で行っている[7]．しかしながら，この流体の方程式には非線形演算が含まれるため，更新を行う際にはデータの一部をいったん実空間へ変換する必要が生じる．データを実空間と周波数空間とでやりとりするためには離散フーリエ変換が必要となる．この際，通常の並列化技法として，適用される空間を分割するタイプの並列化手法（strip mining）[2),8)] を適用すると，全プロセッサ間で，分散された全データをやりとりする必要が生じ，通信コストと通信オーバヘッドが膨大になることが予想される．そこで我々は，乱流場の *DNS* 計算を並列化するために，通信コストと通信オーバヘッドをおさえた形で分散メモリ型環境に効率良く対応させるための手法を提案する．

本論文の 2 章において，熱物質輸送の方程式から導かれる自由表面乱流場における *DNS* に関して定式化を行う．3 章では，乱流場の *DNS* 計算を分散メモリ型並列計算機環境へ適用するためのデータ通信方式の選択とデータ分割手法について論じる．4 章では，実際に開発された逐次プログラムと並列プログラムの実行時間を比較する．

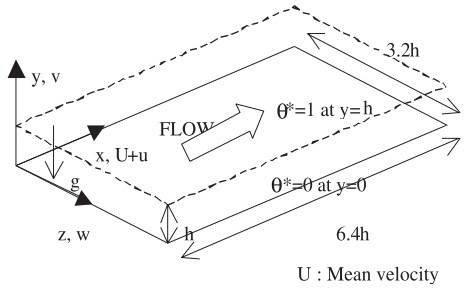

図 1 解析対象と座標系
Fig. 1 Computational domain and coordinate system.

2. 自由表面乱流場における熱物質輸送の直接数値シミュレーション

自由表面を有する乱流は単なるランダムな流れではなく，時空間に対してきわめて高い自由度を持ち，大小様々なスケールを有するため，適切なスケールで 3 次元の非定常計算を行う必要がある．レイノルズ数 $Re = UL/\nu$（U：代表速度，L：代表長さ，ν：流体の動粘性係数）は流れの状態を記述する無次元の数で，対象としている流体中の慣性力と粘性力の比を表す．乱流の完全数値シミュレーション（*FTS*：Full Turbulence Simulation）において，レイノルズ数は重要な指標となる．流れが乱流となる場合，レイノルズ数は大きな値（10^3 以上）となるが，これを数値解として表現するためには，少なくともレイノルズ数の 9/4 乗以上の格子点を要した非定常の 3 次元計算を行わなければならない[5]．

また，流体を記述するもう 1 つの定数であるプラントル数 Pr は，流体の拡散係数 ν と熱拡散係数 α の比（$Pr = \nu/\alpha$）を表すが，流体の拡散係数よりも熱の拡散係数が小さい高プラントル流体における乱流熱輸送の解析を行う場合，3 次元の各方向にプラントル数の $-1/2$ 乗に比例した格子解像度を必要とする．ゆえに，高プラントル自由表面乱流場の *DNS* 計算を考える場合においても，大規模なメモリと高速演算に対処した計算機環境が必要となる．

本研究で対象とした流動場は，無限に広い平板上を一定の外力により駆動され流れる十分に発達した低フルード数開水路乱流場である．**図 1** は，2 次元開水路乱流場の 3 次元座標系を模式的に表したものである．計算条件は，流れ場の平均断面流速 U_m と水深 h においてレイノルズ数を約 2,270，プラントル係数を 1 とし，(x，y，z) 方向にそれぞれ（64，82，64）の格子点を用いた．山本らは，このスケールの格子点で乱流

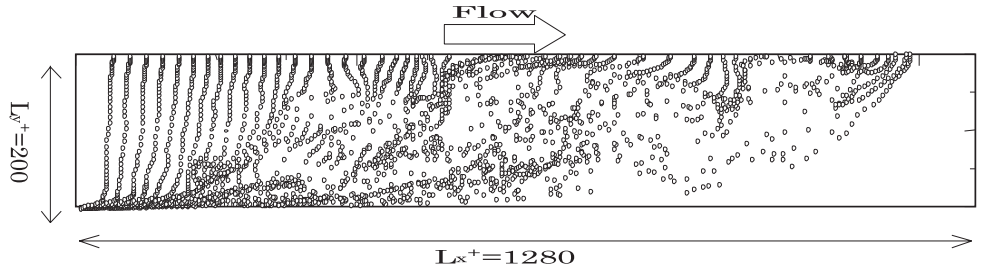

図 2 上流側から仮想的な粒子を注入した場合の流れ
Fig. 2 Visualization of coherent structures obtained from *DNS* database of open-channel flow [6]. Fluid markers are generated along a line to the y-axis (Side view).

場解析を行い，十分な精度の結果を得ている[6),7)]．境界条件として，壁面で no-slip 条件，水面で free-slip 条件，主流およびスパン方向には周期境界条件を適用する．また，この計算結果の例として，上流側から仮想的な粒子を注入して流れを可視化したものを，**図 2** に示す．

数値解析手法を用いるにあたって，乱流場を表現する基本方程式は非圧縮性流体の運動を計算するために，以下のような Navier-Stokes 方程式を用いた．

$$\begin{aligned}\frac{\partial u_i^*}{\partial t}+u_j^*\frac{\partial u_i^*}{\partial x_j}&=F_i-\frac{\partial}{\partial x_i}\left(\frac{p^*}{\rho}\right)+\nu\frac{\partial^2 u_i^*}{\partial {x_j}^2},\\ \frac{\partial u_i^*}{\partial x_i}&=0,\\ \frac{\partial \theta^*}{\partial t}+u_j^*\frac{\partial \theta^*}{\partial x_j}&=\alpha\frac{\partial^2\theta^*}{\partial {x_j}^2}\end{aligned}\qquad(1)$$

ただし，u_i^* は，瞬間的な流速の i（$i=1$，2，3）方向の成分を表し，p^* は瞬間圧力，ρ は流体の密度，θ^* は水面および壁面における温度差により規格化した温度をそれぞれ表すものとする．

通常，流体の *DNS* 計算では，式 (1) で表現される微分方程式を差分化し，有限差分方程式（*FDM*：Finite Difference Method）の形で解く．この系の *FDM* の場合，空間的に近接する格子点の情報のみによって次の時刻の状態が決定されるため，並列化を適用する際，空間方向に対して分割を行い，分割された境界付近の情報のみを隣接部分の計算を行うプロセッサに送信すればよいと考えられる．

しかしながら，この差分方程式の解の精度をあげるためには格子点密度を高くする必要がある．陽解法では，Courant 条件[4)] より，格子点密度の 2 乗に比例させて時間発展幅を小さくしなければならないので，計算実行時間を考えれば，時間発展幅が格子点密度に依存しない陰解法スキームか，*PSM* のような周波数空間での解法を行う方が現実的である．本研究では，*PSM* を用いて *DNS* 計算を行った．式 (1) の周波数空間での表現は以下のように書ける．

$$\begin{aligned}\frac{\partial U_i^*}{\partial t}+\mathrm{j}k_jU_j^*\circ U_i^*&=\mathcal{F}[F_i]-\mathrm{j}k_i\mathcal{F}\left[\frac{p^*}{\rho}\right]\\ &\quad-\nu k_j^2U_i,\\ \mathrm{j}k_iU_i&=0,\\ \frac{\partial\Theta^*}{\partial t}+\mathrm{j}k_jU_j^*\circ\Theta^*&=-\alpha k_j^2\Theta^*\end{aligned}\qquad(2)$$

ただし，U_j^*，Θ^* は，それぞれ u_j^*，θ^* の空間周波数表現とし，演算子 $\mathcal{F}$ はフーリエ変換を表す．また演算子 $\circ$ は畳み込み計算（$K\circ F(\omega)=\int_{-\infty}^{\infty}K(\omega')F(\omega-\omega')d\omega'$）を表すものとする．

式 (2) に関して，周波数空間では微分に関する演算は簡単に表せるようになるが，実空間での積に関する演算が畳み込み表現となるため，この部分は実空間で演算を行った方が効率的である．したがって，時間発展の各時間ステップにおいて，フーリエ変換とその逆変換が必要となる．

ここで考えなければならないのは並列化のための分割方法である．フーリエ変換は実空間から波数空間への変換であり，ある 1 つの波数成分を得るためだけでも全実空間のデータを必要とする．分散メモリ環境において，通常行われるような実空間分割に基づいた並列化（strip mining 等）を行うと，フーリエ変換を行うつど，各プロセッサ間で境界だけでなく全データを送受信する必要が生じるために，通信コストと通信オーバヘッドが膨大になる．したがって，分割方法を工夫する必要がある．この分割手法に関しては，以下の 3 章で述べる．

3. 並列化手法

大規模データを分散メモリ型並列計算機で計算させる場合，通信コストと通信オーバヘッドを考慮したデータ分割を行う必要がある．そこで，まず 3.1 節に

おいて，分散メモリ型並列計算機を対象とする並列化ライブラリ Message Passing Interface（*MPI*）の環境下での同期式通信と非同期式通信のメリットおよびデメリットに関して述べ，3.2 節において，提案する並列化手法に関して簡単な例を用いて論じる．本研究で取り扱う *PSM* を用いた *DNS* 計算に対する並列化手法の適用は，3.3 節において説明する．

3.1 プロセッサ間の通信方式

並列プログラムを分散メモリ型環境において実装した場合，各プロセッサに対して効果的にデータが分割されていたとしても，最小限のデータ通信が必要である．このデータ通信を行うための関数として，並列化ライブラリ *MPI* には，同期式通信と非同期式通信の 2 種類の通信方式がある．

同期式通信の特徴は，送信側プロセッサによる送信用関数（*MPI_SEND*）の呼び出しと，対になる受信側プロセッサによる受信用関数（*MPI_RECV*）の呼び出しによってデータ通信が同期的に行われることである．すなわち，同期式通信ではプログラムの実行は通信が終了するまでブロッキングされる．

これに対して非同期式通信において，送信用関数（*MPI_ISEND*）の呼び出しと，受信用関数（*MPI_IRECV*）の呼び出しは，別々のタイミングに行うことが可能である．これは，計算実行がブロッキングされることはないものの，データ通信のタイミングによっては，間違ったタイミングにデータを書き換えてしまう危険性があることを意味する．したがって，プログラマがプログラム設計の段階でデータの依存性に対して十分に考慮する必要がある．

乱流場の *DNS* 計算のような時間発展方程式の並列化を取り扱う場合には，空間方向に関するループを strip mining によって分割することによりデータ分割を行う方法が一般的である．この場合，空間が均等に分割されていれば各計算部分がホモジニアスな関係となるため，各プロセッサの計算がほぼ同時に終了すると考えられる．したがって，この場合，同期式通信を用いてもブロッキングの影響はほとんどでない．逆に空間が非均等に分割されるほど，最も計算が遅くなるプロセッサに同期しなければならなくなる．また，プロセッサの処理速度が非均等な環境においても，同期式通信のブロッキングの影響は大きくなる．このような場合，非同期式通信を用いた方が，通信オーバヘッドを回避しやすい．

3.2 データ分割手法

do 文に関して，各イタレーション間に依存関係が存在しない場合，並列実行文の *doall* 文に変換することができる．一般的に，*doall* 文は，全プロセッサに対して strip mining されることが理想的とされている．

通常，流体のシミュレーションを *FDM* を用いて行う場合，各配列データを更新する際，考慮しなければならない作用は，空間的に隣接もしくは非常に近い位置からの作用のみである．そのため，並列化のために必要となるデータ通信は，境界部分のみであり，通信コストは非常に小さいといえる．したがって，乱流場の *DNS* 計算の並列化は，strip mining のような手法を用いることが一般的である．

しかしながら，本研究では計算精度をあげるために *PSM* を適用し，波数空間での *DNS* 計算を行っている．式 (1) には非線形成分が含まれていることから，時間発展を行う各ステップにおいて，この非線形成分を計算するために，いったん実空間に戻した形で非線形項の計算を行い，もう一度波数空間に戻すという方法を適用している．

フーリエ変換の性質上，ある配列の一要素を更新するときでさえ，その配列の全データが必要となる．これは，データの依存性が著しく増大することを意味する．すなわち，空間方向に対して strip mining を適用した場合，ブロードキャストによる通信コストと通信オーバヘッドはきわめて大きいものとなることが予想される．

そこで本研究では，strip mining を用いて配列を空間方向に等分割するのではなく，配列データ間の依存関係から推定される通信コストに応じて分割する手法を提案する．本分割手法の効果を検証するために，以下に述べる簡単な例題を対象として，並列化に関する考察を行い，実際のプログラムの実行時間を評価する．

＜例 1＞

```
     do 999 l=1,1000
        do 10  i = 1,100000
 10        x(i) = i
        min = y(1)
        do 20  j = 2,100000
 20        if(min .gt. y(j)) min = y(j)
        do 30  k = 1,100000
 30        z(k) = z(k)*k
        do 40  i = 1,100000
 40        w(i) = x(M(i))*min
 999 continue
```

この例 1 のプログラムにおいて，以下の計算がなされている．

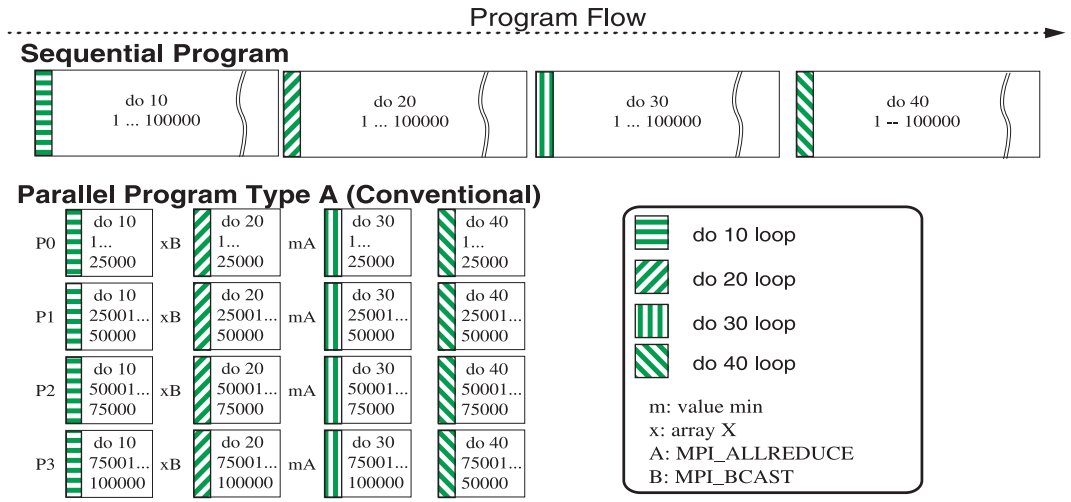

図 3 各 *do* 文に対し，全プロセッサを対象とする strip mining による分割を実装した場合の並列プログラムの概念図（並列化手法 *A*）

Fig. 3 The schematic diagram of the time course of Parallelizing method *A*. Each loop is divided by the strip-mining for whole processors.

(1) *do* 10 で，配列 `x` に値を代入
(2) *do* 20 で，配列 `y` 中の最小値の探索
(3) *do* 30 で，配列 `z` に対する操作を実行
(4) *do* 40 で，step 2 で求めた最小値 `min` と step 1 で計算された配列 `x` の `M(i)` 番目の要素との積を計算

なお `M(i)` は添字をランダムに並べたものとする．これによって，並列化の際，step 4 に，step 1 で計算された配列 `x` のすべてのデータが，すべてのプロセッサにおいて必要とする状況を作り出した．この例 1 に対し，一般的に用いられる手法 *A* と本研究で提案する手法 *B* を用いて並列化し比較する．

並列化手法 *A* は，全プロセッサに対して *doall* 文を strip mining によって均等に分割する一般的な並列化手法である．すなわち，複数の配列を処理していく場合，各配列を逐次的に処理し，配列の処理ごとに複数のプロセッサに均等にデータと演算を分割していく手法である．この場合，3.1 節で述べたように各プロセッサ間に計算時間の差がなくなるので，同期式通信によるデータ通信を行うことを考える．**図 3** は，例 1 に対して手法 *A* を 4 プロセッサに対して実装した場合の実行順序を表す．図中，横軸は時間経過を表し，上部と下部はそれぞれ逐次プログラムおよび並列プログラムの実行を表すものとする．この分割手法は，配列の要素をプロセッサの数に応じて分割する方式である．よって，各ループ番号に関して配列要素を 25,000 個ずつに分割し，そのデータを各プロセッサで処理させる．処理後，計算方法に応じて，計算結果を通信によって交換する．並列化手法 *A* において，*do* 20 に strip mining を施すことによって，各プロセッサは分割された配列要素の中で極小値となるものを検出する．検出後，各プロセッサが検出した変数 *min* のうち最小のものを選択するための同期式通信 *MPI_ALLREDUCE* が必要となる．また，*do* 10 において初期化される配列 *x* は *do* 40 においてランダムにアクセスされる．そのため，各プロセッサは自分の持っていないデータを取得する必要が生じ，ブロードキャストを行う同期式通信 *MPI_BCAST* によってデータを取得しなければならない．

これに対して，我々が提案する並列化手法 *B* は，複数の配列の処理を行う際に，配列の計算におけるデータ依存を考慮し，1 個の配列がなるべく少数のプロセッサに割り当てられるように分割し，複数の配列をなるべく同時に処理する手法である．**図 4** は，例 1 に対して手法 *B* を適用した場合の実行順序を示した図である．例 1 における配列 *x* のデータ依存関係は，*do* 10 と *do* 40 でのみ生じる．これらの配列を分割することによって生じる通信コストを減少させるためには，*do* 10 と *do* 40 を計算するプロセッサ数を減少させればよい．*do* 20 と *do* 40 では，変数 *min* に関してデータ依存関係がある．また，*do* 30 は他のループとデータ依存関係を持たず，*do* 10 と *do* 20 は依存関係を持たない．以上のことをふまえると，*do* 40 を計算する前に実行されるべきループは，*do* 10，*do* 20 である．ゆえに，並列化手法 *B* として，全プロセッサのうち，半数ずつのプロセッサを用いて，同時に *do* 10 と *do*

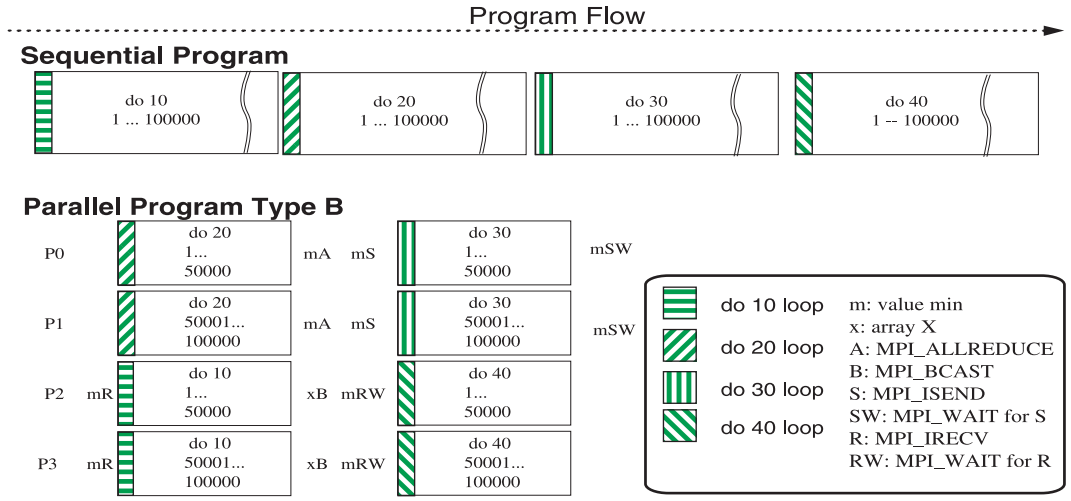

図 4 各 *do* 文に対し，いくつかのプロセッサを対象とする strip mining による分割を実装した場合の並列プログラムの概念図（並列化手法 *B*）
Fig. 4 The schematic diagram of the time course of our proposal Parallelizing method *B*. Each loop is divided to minimize the communication cost.

20 を計算させることとした．例 1 において，*do* 20 を strip mining によって 2 つに分割し，各々をプロセッサ id 0（*P*0）と id 1（*P*1）とに割り当てた．*P*1 は，処理後，計算結果である *min* を *P*0 に送信する．同タイミングに，プロセッサ id 2（*P*2），id 3（*P*3）において，*do* 10 を処理させ，処理後，それぞれ，配列 *x* を送受信する．*P*0 において，*P*1 から送信された *min* を受信後，*P*0 固有の *min* と比較し，小さい方を *P*2，*P*3 に送信する．最後に，各通信終了後，*P*0 と *P*1 において，*do* 30 を処理し，*P*2 と *P*3 において，*do* 40 を処理する．*P*0 と *P*1 に関して，*do* 20 を strip mining を用いて分割しているため，計算時間の差は少ないと考えられる．また，*P*2 と *P*3 に関しても同様である．ゆえに，*P*0 と *P*1 間，または，*P*2 と *P*3 間における通信は，同期式通信とした．しかし，*P*0 と *P*2，*P*3 間における通信は，*do* 20 と *do* 10 の計算時間が異なるため，非同期式通信を適用するものとした．

例 1 の逐次プログラムと並列化手法 *A* および *B* をそれぞれ用いて実装した並列プログラムを実行した場合の性能を**表 1** に示す．user 時間は各プロセッサにおける演算時間および関数呼び出しのための時間の合計を表し，system 時間は通信時間および通信オーバヘッドを表す指標とする．

今回提案した並列化手法 *B* は，並列化手法 *A* と比較して，プログラムの設計の段階で，データ依存関係と計算順序を考慮しなければならず，実装を行う際のポータビリティという面においては劣ると考えられる．しかし，例 1 のプログラムを用いた予備実験の結果より，並列化手法 *B* を用いた並列プログラムの方が，並列化手法 *A* を用いるよりも，通信コストが多いにもかかわらず，関数呼び出しによる user 時間の増加が少なく，また，system 時間が減少していることから通信オーバヘッドが抑えられていることが分かる．したがって，性能面において，並列化手法 *B* は並列化手法 *A* よりも優れていると考えられる．

表 1 例 1 の実行時間
Table 1 Execution time of Example 1.

	user time (s)	system time (s)
逐次プログラム	123	0
並列化手法 *A*	55	73
並列化手法 *B*	43	28

3.3 乱流場の *DNS* 計算への実装

我々が対象とする乱流場の *DNS* 計算のための逐次プログラムは，初期化部分と計算部分に分割することができる．この逐次プログラムの実行時間の大部分は，時間発展の計算部分の繰返しによって消費される．ゆえに，時間発展計算部を最適に並列化することが重要となる．

乱流場の *DNS* 計算で使われる主な配列を**表 2** に示す．

乱流場の *DNS* 計算の時間発展計算部において，次の処理がなされる．

(1) フーリエ変換後の各流速 u，v，w，および熱 t を用いてそれぞれの対流を計算．
(2) Courant 条件[4] の計算．

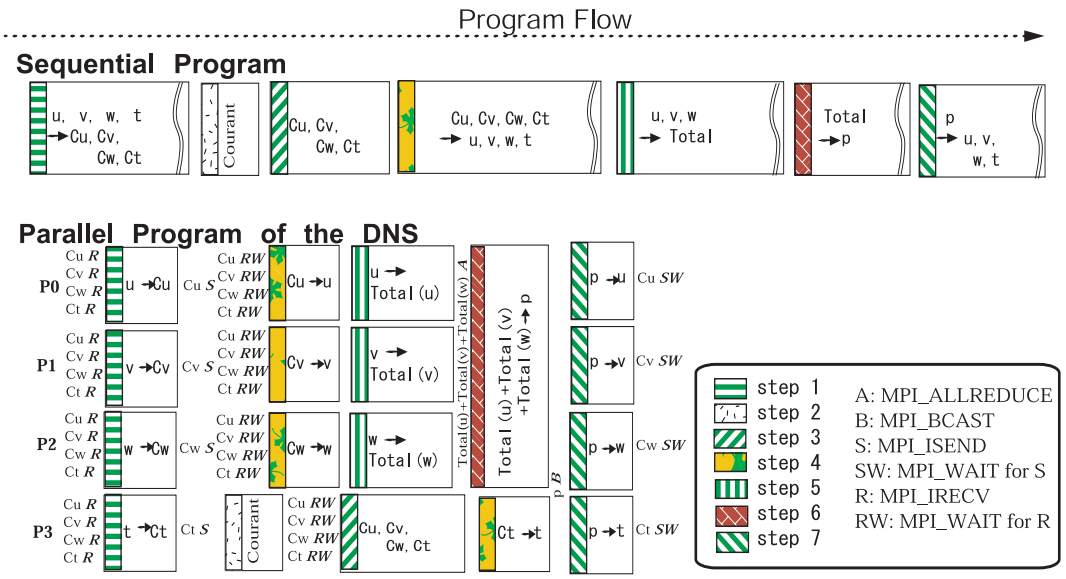

図 5 乱流場の *DNS* 計算に対し，並列化手法 *B* を適用した場合の並列プログラムの概念図
Fig. 5 The schematic diagram of the time course of the *DNS*.

表 2 乱流場 *DNS* 計算で用いる主要配列
Table 2 The list of the main arrays in *DNS*.

x y z	Grid intervals and coordinates in the three dimension
dist_x dist_y dist_z	The temperature at the water surface or wall
u v w	Flow velocities in the x, y, z direction
fu, fuo fv, fvo fw, fwo	(a convertive term) + (a viscous term)
t	The temperature
ft fto	(a convertive term) + (a viscous term) in the energy equation of the temperature
p	(pressure) / (density)

(3) step 1 によって得られた全対流を用いて，流速および熱の全エネルギーを計算．

(4) step 1 によって得られた全対流を用いて，各流速 u，v，w，および熱 t に対する擬スペクトル法[3] とエイリアシング誤差の除去[1] を施す．

(5) 2 次精度の Adams-Bashforth 法[4] による流速 u，v，w の統合．

(6) step (5) によって得られた統合値を用いて，圧力項 p のポアソン解法を行うために，フーリエ変換と 3 重対角行列の連立一次代数方程式の直接解法 *TDMA*（3 重対角行列法）を使った直接解法の計算．

(7) 圧力項 p を用いて，流速 u，v，w，および熱 t の更新．

step (1)，(4)，(7) において，配列 t，u，v，w に関する計算時間はすべて等しい．step (2) と step (3) の計算時間は，step (5) と step (6) の計算時間の 1/3 である．これらの計算時間の関係により，1 つのプロセッサを用いて，配列 t の計算ならびに step (2)，step (3) の計算を行い，3 つのプロセッサを用いて，各配列 u，v，w をそれぞれ計算し，step (5) による統合後，step (6) の計算を strip mining することにより並列実行する．step (1)，(4)，(7) において，各配列要素のデータを処理するにあたり，必要なデータは，その隣接する配列要素であるため，他の配列を計算するプロセッサからデータ通信をする必要性はない．step (3)，(4) において，step (1) の計算によって得られた各配列の対流値がすべてのプロセッサにおいて必要となる．そこで，step (1) の計算終了後，各プロセッサは，計算された各対流値を非同期式に送信し，step (2) の計算終了後，非同期式に受信することとする．step (5) に関して，各流速 u，v，w それぞれにおける統合後，全流速の統合を並列化ライブラリ *MPI* の同期式通信関数 *MPI_ALLREDUCE* を用いて行う．step (7) において必要な圧力項 p は，step (6) において計算される．そこで，step (7) における各配列更新の計算時間は等しいので，同期式通信によって得ることとする．これらのことをふまえて乱流場の *DNS* 計算を並列化すると，**図 5** のようになる．

以上より，並列化手法 B を *DNS* に適用する場合，プロセッサ数が 4 の場合，最も有効になると考えられる．

また，時間発展計算部分の各 step において，各配列計算は，多重ループを用いて行われ，多重ループの最外郭の *do* 文は，イタレーション間に依存関係を持たない．そのため，strip mining することによって等分に分散させた場合，並列化による有効性を失うことはない．ゆえに，プロセッサ数が $4n$（n：自然数）であった場合，プロセッサ数 4 用に分散された計算をそれぞれ n 個の部分に strip mining することによって，並列化手法 B の有効性を失うことなく拡張することが可能である．

4. 計算機実験と性能評価

3.2 節で提案した並列化手法 B を乱流場の *DNS* 計算のためのプログラムに実装し，実際の有効性を検証した．

実験環境として，1 GHz の Pentium III の CPU を持つ計算機を 52 台用意し，OS として Linux 2.4.2（Redhat Linux 7.1），並列化ライブラリとして MPICH 1.2.4 を用いた．通信には，100base-TX を用いた LAN 上の TCP/IP を利用した．

実験に用いたプログラムにおいて，安定した乱流場のシミュレーション結果を得るためには，時間発展の計算部分の繰返し回数を 100,000 回程度に設定しなければならず，このための計算時間は，約 1 週間かかる．しかしながら，各イタレーションにおいて計算手順が変化することはないので，並列化による性能評価を行うためには，1 イタレーションの計算時間が計測できればよいこととなる．ここでは，計測時間の揺らぎを考慮して，イタレーション数を 100 回とした場合の計算時間を評価することとした．

乱流場の *DNS* 計算をするための逐次プログラムの user 時間と system 時間は，それぞれ 2,632 秒，2 秒であった．逐次プログラムを実行した場合も並列プログラムを実行した場合同様，若干の system 時間が生じている．この system 時間は，ファイルの読み書きによるものであると思われる．

並列プログラム（プロセッサ数 4，8，16，32，52）の user 時間と system 時間を**図 6** に示す．

プロセッサ数 4，8，16，32，52 において，user 時間と system 時間を合わせた時間のうち最大のものは，それぞれ，747 秒，462 秒，306 秒，241 秒，230 秒であった．これは，逐次プログラムの時間に対する比で表すと，0.28，0.17，0.11，0.09，0.08 である．比率

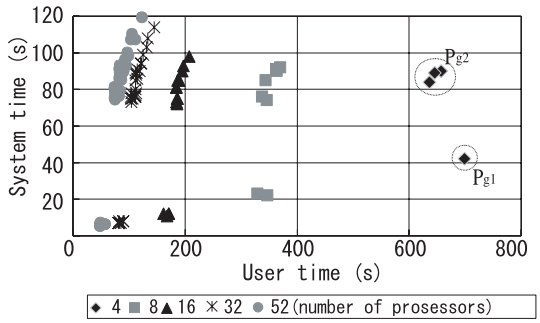

図 6 乱流場 *DNS* の計算を並列プログラムで行った場合の user 時間と system 時間の関係

Fig. 6 Relationships between user time and system time in several multi-processors cases, which is obtained by the calculation of the parallel *DNS* program. The horizontal axis indicates user time, and the vertical indicates system time. We use 4, 8, 16, 32, 52 processors in this simulation.

がリニアにならなかった理由として，system 時間に該当する通信オーバヘッドの増加があげられる．user 時間のみに着目すれば，理想時間（逐次プログラムの時間/プロセッサ使用数）とほぼ等しい．

4 つのプロセッサを用いた並列プログラムに関して，図 6 に表されている一番右下のプロセッサ P_{g1} の user 時間が最も長い．この原因として考えられることは，乱流場の *DNS* 計算における結果の標準出力をプロセッサ P_{g1} に集中させたことである．また，残り 3 つのプロセッサ P_{g2} に関して，system 時間が長い．これは，3.3 節で説明した計算部分に関して，step (5)，(6) で用いたデータ通信後に計算を必要とするタイプの通信 *MPI_ALLREDUCE* と多数のブロードキャスト通信 *MPI_BCAST* によるものであると思われる．

8 つ以上のプロセッサを用いた並列プログラムに関して，図 6 より，4 つのプロセッサを用いた並列プログラムの実行時間が等分に分割されていることが分かる．プロセッサ P_{g1} に対応するプロセッサに関しては，system 時間が減少している．これは，プロセッサ数が増えたことによって，一度に通信されるデータサイズが小さくなったためであると考えられる．一方，3 つのプロセッサ P_{g2} に対応するプロセッサに関して，system 時間があまり変化しなかった原因としては，3 つのプロセッサ P_{g2} によって計算されていたデータを細分化することにより，*MPI_ALLREDUCE* の実行時間が長くかかることと，*MPI_BCAST* の回数が増加したことがあげられる．

5. ま と め

本研究において，逐次プログラムで開発された自由

表面乱流場における熱物質輸送の直接数値シミュレーション *DNS* の特徴を考慮し，分散メモリ型並列計算機用の並列化ライブラリ Message Passing Interface を利用してプロセッサ数 $4n$（n：自然数）用の並列プログラムを開発した．計算機物理学で扱う微分方程式の作用が，時間・空間的に近傍部分の影響ですむという前提条件を持つ場合，従来の並列化手法 A は有効である．しかしながら，計算技法として，本研究が対象とするような *PSM* を用いた乱流場の *DNS* 計算をする場合や，配列をランダムにアクセスしなければならないような計算を行う場合，この前提条件が崩れるため，本研究で提案した分割手法 B が有効になると考えられる．我々は，*PSM* を用いた *DNS* 計算に対する並列化手法 B の性能を評価するために，データ通信コストを考慮して並列プログラムを実装した．実験結果から，提案された並列化手法 B を用いた場合，user 時間に関して，ほぼリニアな結果がえられた．また，system 時間に関して，並列実行時間として現実的な環境で使用しうる結果が得られた．

今回の実験環境は，輻輳が大きく信頼性がそれほど高くない TCP/IP の下で実行を行わざるをえなかった．それにもかかわらず，このような環境下においても有効な結果が得られたことの意味は大きい．本研究の成果は，エントリレベルのネットワークと PC ワークステーションの組合せという比較的性能の劣った環境においても並列計算が有効であることを示したものと考えられる．今後の方針としては，プロセッサとネットワークの質・量をともに向上させていった場合に理論的な性能を出すことが可能であるかを検証することである．

参 考 文 献

1) Canuto, C, Hussaini, M.Y., Quarteroni, A. and Zang, T.A.: *Spectral Methods in Fluid Dynamics*, Springer Verlag (1988).
2) 中田育男：コンパイラの構成と最適化，朝倉書店 (1999).
3) Orszag, S.A.: A Numerical simulation of incompressible flows within simple boundaries, *Galerkin* (*Spectral*) *representations, Stud. Appl. Math.* 50, pp.293–327 (1971).
4) Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T.: *Numerical Recipes in C*, Cambridge University Press (1988).
5) ランダウ・リフシッツ：流体力学（1），東京図書 (1990).
6) Yamamoto, Y., Kunugi, T. and Serizawa, A.: Turbulence Statistic and Scalar Transport in an Open-Channel Flow, *Advances in Turbulence VIII*, Dopazo, C. (Ed.), pp.231–234 (2000).
7) 山本義暢：熱輸送を伴う自由表面混相乱流場の直接数値シミュレーション，京都大学博士論文 (2002).
8) Zima, H. and Chapman, B.: *Supercompilers for Parallel and Vector Computers*, Addison-Wesley Publishing (1991).

(平成 14 年 9 月 26 日受付)
(平成 15 年 1 月 17 日採録)

髙田 雅美（学生会員）

昭和 52 年生．平成 13 年奈良女子大学大学院人間文化研究科情報科学専攻修士課程修了．平成 13 年奈良女子大学大学院人間文化研究科複合領域科学専攻進学．並列計算機を用いたプログラムの開発に関する研究に従事．

山本 義暢

昭和 48 年生．平成 11 年京都大学大学院工学研究科環境地球工学専攻修士課程修了．平成 14 年京都大学大学院工学研究科原子核工学専攻博士課程修了．平成 13 年より日本学術振興会特別研究員．気液混相乱流場の直接数値計算，乱流熱物質移動に関する研究に従事．博士（工学）を京都大学より取得．日本流体力学会，可視化情報学会各会員．

庄野 逸（正会員）

昭和 43 年生．平成 4 年大阪大学大学院基礎工学研究科物理系専攻生物工学分野修了．同年大阪大学基礎工学部助手．平成 9 年大阪大学大学院基礎工学研究科助手．平成 13 年奈良女子大学大学院人間文化研究科助手．平成 14 年より山口大学工学部助教授．神経回路モデル等の確率的情報処理に関する研究に従事．平成 11 年博士（工学）を大阪大学より取得．日本神経回路学会，電子情報通信学会，物理学会各会員．

功刀　資彰

昭和 28 年生．昭和 54 年慶應義塾大学大学院工学研究科応用化学専攻修士課程修了．同年日本原子力研究所入所．平成元年より平成 3 年までの期間米国 UCLA および IPFR 客員研究員．平成 6 年博士（工学）を東京大学より取得．平成 10 年東海大学工学部動力機械工学科教授．平成 11 年京都大学大学院工学研究科原子核工学専攻助教授．現在に至る．熱工学，混相流工学，数値流体力学，核融合炉工学等の研究に従事．平成 5 年可視化情報学会功労賞，平成 14 年国際フランス伝熱学会賞受賞．日本機械学会，日本伝熱学会，日本原子力学会，日本流体力学会，可視化情報学会各会員．

城　　和貴（正会員）

大阪大学理学部数学科卒業．日本 DEC，ATR 視聴覚研究所（日本 DEC より出向），（株）クボタ・コンピュータ事業推進室で勤務．平成 5 年奈良先端科学技術大学院大学情報科学研究科博士前期課程入学．平成 8 年博士（工学）を同大学院大学より取得．平成 8 年同大学院大学情報科学研究科助手．平成 9 年和歌山大学システム工学部情報通信システム学科講師．平成 10 年同学科助教授．平成 11 年奈良女子大学理学部情報科学科教授．画像処理，文字認識，ニューラルネットワーク，並列計算機アーキテクチャ，自動並列化コンパイラ，並列計算機の解析モデル，視覚化等の研究に従事．IEEE，ACM 各会員．