JAERI-Data/Code
96-012

非圧縮性NS方程式における圧力方程式の並列解法と評価

1996年3月

市原　潔*・横川三津夫・蕪木英雄

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所技術情報部情報資料課（〒319-11 茨城県那珂郡東海村）あて，お申し越しください。なお，このほかに財団法人原子力弘済会資料センター（〒319-11 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Information Division, Department of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.

編集兼発行　日本原子力研究所
印　　刷　㈱原子力資料サービス

JAERI－Data/Code 96－012

非圧縮性ＮＳ方程式における圧力方程式の並列解法と評価

日本原子力研究所計算科学技術推進センター
市原　潔*・横川三津夫・蕪木　英雄

（1996年２月９日受理）

３次元非圧縮性ナビエ・ストークス方程式（ＮＳ方程式）のうち，最も計算時間を要する圧力 Poisson 方程式の解法部分について，赤黒逐次過緩和法（red-dlack ＳＯＲ法）および前処理付き共役勾配法（ＰＣＧ法）をとりあげ，分散メモリ型ベクトル並列計算機富士通ＶＰＰ500でＤＯループ分割による並列化を試みた結果を報告する。red-black ＳＯＲ法については，分散メモリ型スカラ並列計算機インテル Paragon でも実行し，スケーラビリティの比較を行った。

Red-black ＳＯＲ法では，ＰＥ数の増大とともにスケーラビリティが失われる傾向がＶＰＰ500と Paragon で共通している。しかし，Paragon の場合は通信時間の増大のみがその原因であるのに対し，ＶＰＰ500の場合はそのほかにＤＯループ分割によるベクトル効率の低下という原因があることが判明した。したがって，ベクトル化効率の著しい低下を招かないような大規模な問題の並列化がＶＰＰ500での並列実行に適する。

また，ＰＣＧ法の並列化にあたって，前処理行列の不完全ＬＵ分解法に red-black ＳＯＲ法を適用した。この方法では不完全ＬＵ分解を解くために，１回の前進代入・後退代入を複数回の反復に置き換えるために，全体の浮動小数点演算数が著しく増大するにもかかわらず１回のＤＯループに含まれる演算数は少ないままである。したがって，並列化のＤＯループ分割を行うと，ベクトル化効率低下が red-black ＳＯＲ法よりも顕著に現れ，本報で述べたケースのうち格子数の少ないものについては，red-black ＳＯＲ法に比べて並列化のメリットが現れにくいこが判明した。

日本原子力研究所：〒153　東京都目黒区中目黒2-2-54
＊業務協力員　㈱三菱総合研究所

JAERI－Data/Code 96－012

Parallelization of Pressure Equation Solver for Incompressible N-S Equations

Kiyoshi ICHIHARA* ,Mitsuo YOKOKAWA and Hideo KABURAKI

Center for Promotion of Computational Science and Engineering
Japan Atomic Energy Research Institute
Nakameguro, Meguro-ku, Tokyo

(Received February 9, 1996)

A pressure equation solver in a code for 3-dimensional incompressible flow analysis has been parallelized by using red-black SOR method and PCG method on Fujitsu VPP500, a vector parallel computer with distributed memory. For the comparison of scalability, the solver using the red-black SOR method has been also parallelized on the Intel Paragon, a scalar parallel computer with a distributed memory.

The scalability of the red-black SOR method on both VPP500 and Paragon was lost, when number of processor elements was increased. The reason of non-sacalability on both systems is increasing communication time between processor elements. In addition, the parallelization by DO-loop division makes the vectorizing efficiency lower on VPP500. For an effective implementation on VPP500, a large scale problem which holds very long vectorized DO-loops in the parallel program should be solved.

PCG method with red-black SOR method applied to incomplete LU factorization (red-black PCG) has more iteration steps than normal PCG method with forward and backward substitution, inspite of same number of the floating point operations in a DO-loop of incomplete LU factorization. The parallelized red-black PCG method has less merits than the parallelized red-black SOR method when the computational region has fewer grids, becaues the low vectorization efficiency is obtained in red-black PCG method.

Keywords: Vector Parallel Calculation, Navier-stokes Equations, Pressure Poisson Equation

---

* on loan to Mitsubishi Research Institute Inc.

目　　次

Contents

1. 序論

次世代計算機の技術のひとつとして並列処理に期待が寄せられている。流体解析のように大規模かつ高精度な数値シミュレーションを実用レベルで活用するためには、これまで以上に処理速度の向上に努力が払われる必要があり、並列処理はひとつの重要なキーとなるものである。特に、各プロセッサにベクトルユニットと局所メモリを持つ分散メモリ型ベクトル並列計算機は、ベクトル処理と並列処理の両方による性能向上が注目される。

３次元非圧縮性ナビエス・トークス方程式（ＮＳ方程式）のうち、圧力 Poisson 方程式は、連立一次方程式で表現され、最も計算時間を要することが知られている。時間依存のＮＳ方程式の場合には、右辺ベクトルが変化する連立一次方程式を時間ステップ毎に解くので、１時間ステップ前の圧力解を初期値とする反復法を適用する方が、直接法を用いるよりも有利である。本報では、圧力 Poisson 方程式の解法部分について、反復法である赤黒逐次過緩和法（red-black SOR 法）および前処理付き共役勾配法（PCG 法）をとりあげ、分散メモリ型ベクトル並列計算機富士通 VPP500 でＤＯループ分割による並列化を試みた結果を報告する。ケーススタディとして、分散メモリ型スカラ並列計算機であるインテル Paragon 上での結果とスケーラビリティの比較を行った。

結論として、red-black SOR 法について、ＰＥ数の増大とともにスケーラビリティが失われる傾向は共通しているが、Paragon の場合は通信時間の増大のみがその原因であるのに対し、VPP500 の場合はそれ以外にＤＯループ分割によるベクトル効率の低下が原因であることが判明した。したがって、ベクトル化効率の著しい低下を招かないような大規模な問題の並列化が VPP500 での並列実行に適する。

また、PCG 法の並列化にあたって、前処理行列の不完全ＬＵ分解法に red-black SOR 法を適用した。この方法では不完全ＬＵ分解を解くために、１回の前進代入・後退代入を複数回の反復に置き換えるために、全体の浮動小数点演算数が著しく増大するにもかかわらず１回のＤＯループに含まれる演算数は少ないままである。したがって、並列化のＤＯループ分割を行うと、ベクトル化効率低下が red-black SOR 法よりも顕著に現れ、本報で述べたケースのうち格子数の少ないものについては、red-black SOR 法に比べて並列化のメリットが現れにくい。

なお、共有メモリ型ベクトル並列計算機（CRAY Y-MP 8/664）による Multi-color SOR 解法に関する収束性の改善と演算速度の評価の研究が小野ら [1] によってなされている。

2. 対象とする物理モデルと数値解法

2.1 基礎方程式

方程式は時間項を含む非圧縮ＮＳ方程式とする。流れの速度ベクトルを以下のように表すことにする。

$$\vec{u} = (u, v, w)$$

連続の式は、以下のように表される。

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

運動方程式は、以下のように表される。

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial (vu)}{\partial y} + \frac{\partial (wu)}{\partial z} = -\frac{\partial p}{\partial x} + \frac{1}{Re}(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2})$$

$$\frac{\partial v}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial v^2}{\partial y} + \frac{\partial (wv)}{\partial z} = -\frac{\partial p}{\partial y} + \frac{1}{Re}(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2})$$

$$\frac{\partial w}{\partial t} + \frac{\partial (uw)}{\partial x} + \frac{\partial (vw)}{\partial y} + \frac{\partial w^2}{\partial z} = -\frac{\partial p}{\partial z} + \frac{1}{Re}(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2})$$

なお、乱流モデル等を考慮せず、移流項には低次の差分法を用いているため、高レイノルズ数の流れの解析は困難である。本報ではすべてのケースについてレイノルズ数を 100 として計算した。

2.2 境界条件

流路すなわち計算領域は、正方形断面を持つ直方体とした (Fig.2.2.1)。流れ方向をｘ軸、高さ方向をｚ軸とし、右手系をなすようにｙ軸をとる。直方体のｘ、ｙ、ｚ方向の寸法をそれぞれ L_x, L_y, L_z とする。L_x, L_y, L_z の比率は以下の通りである。

$$L_x : L_y : L_z = 15 : 1 : 1$$

入口にて一様流を与え、上面と下面で滑りなしの境界条件、すなわち以下の条件を与える。

入口：$x = 0$ において、一様流を与える。

$$u = 1, v = w = 0$$

上面、下面：$z = 0, z = N$ において、すべり無しとする。

$$u = v = w = 0$$

側面：$y = 0, y = M$ において、側面方向の流速加速度をゼロとする。

$$\frac{\partial u}{\partial y} = \frac{\partial v}{\partial y} = \frac{\partial w}{\partial y} = 0$$

出口：$x = L$ において、出口方向の流速加速度をゼロとする。

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial x} = \frac{\partial w}{\partial x} = 0$$

また、圧力のリファレンス・ポイントも出口で与えることとし、圧力の出口方向の微分値がゼロであるとする。

$$p = 0, \frac{\partial p}{\partial x} = 0$$

流路のx方向の長さが十分長いことから、定常状態に落ち着いたときの出口での流れ方向の軸を含む縦断面での流速プロファイルはほとんど Poiseuille Flow と同じになる。Fig.2.2.2 に計算結果と解析解を対照して示す。両者の一致は良好である。

2.3 離散化と数値解析方法

時間の離散化は、time splitting method に基づく陽的差分である。流れの速度ベクトル $\vec{u} = (u, v, w)$ に対し、以下のように計算される。

$$\vec{u}^{n+1} = \vec{u}^{*} - \triangle t \nabla p^{n+1}$$

ここで、$\vec{u}^{*}$ は、前回のタイムステップで計算された速度ベクトルである。

空間の離散化は、等間隔のスタッガードメッシュによる。すなわち、圧力が定義される格子点が、速度成分の定義される格子点の中点になるように配置される。圧力が定義される格子点の最も外側のものは、物理領域と一致し、速度成分の定義される格子点の最も外側のものは、格子間隔の半分の距離だけ、物理領域から外にはみだすことになる。

3. 並列化の方針

本並列化では、２章に述べた物理モデルおよび数値解析方法に基づいた逐次計算機評価用のプログラム MiFlow をとりあげた。ここでは、既存プログラムの並列化を前提として、その方針を述べる。

3.1 並列化のための分割方法

一般に、複数プロセッサエレメント（ＰＥ）による並列化のための分割方法として、以下が考えられる。

(1) 手続き分割
繰り返し手続き等を複数ＰＥで分担して実行するようにプログラムを書き換えるものである。手続きの独立性（排他性）が高いほど、並列実行の効率は高くなる。

(2) データ分割
手続き分割に対応して、データを分割するものである。データ定義の変更に伴い、モジュール間インターフェースの書き換えが発生する。

上記のうち、複数ＰＥでの実行のために、「手続き分割」は必須である。「データ分割」は、メモリ節約という意味で行うことが望ましい。新規プログラム開発の場合であれば「手続き分割」と「データ分割」の両方を勘案した設計を行うべきであろう。しかし、既存プログラムの改造による並列化の場合は、変更がプログラム全体に及ぶため、現実的でないことが多い。

今回は、既存プログラム MiFlow を並列化するので、「手続き分割」のみを行うことにした。並列化の対象は、圧力方程式の解法ルーチンである。圧力方程式の解法ルーチンのコーディングステップ数が全体の１割に満たないにもかかわらず、実行時には全実行時間の９割程度を費やしているからである。したがって、圧力方程式の解法ルーチンの並列化によって、少ない工数で実行時間の短縮を図れることになる。

3.2 圧力 Poisson 方程式の解法の並列化

圧力 Poisson 方程式は、圧力を未知数とした連立一次方程式の形に表現できる。特に、非定常のＮＳ方程式では、時間ステップを進めながら速度場と圧力場とを交互に求める必要があるので、右辺ベクトルが少しずつ変化していく連立一次方程式を時間ステップ毎に解くことになる。

連立一次方程式の解法には大きく分けて、直接法と反復法がある。非定常流れの計算では、ある時間ステップでの解が次の時間ステップでの解の良好な近似値であるために、反復法が有利である。本報では、２通りの反復解法すなわち、red-black SOR 法と PCG 法をとりあげる。以下にそれぞれについて簡単に説明する。なお、線形方程式 $A\vec{x} = \vec{b}$ を解くものとし、行列 A は、下三角部分 L 、対角部分 D、上三角部分 Uに分けられるものとする。すなわち、

$$A = L + D + U$$

とする。ここで、L と U は、A を LU 分解したものでないことに注意されたい。Fig.3.2.1 に、4個のＰＥによる手続き分割のイメージを示す。ＤＯループ分割により、結果的に圧力の未知数 $\vec{x}_1$ の求解領域がＰＥの個数分に分けられる。３次元領域の７点差分による係数行列は規則的な疎行列で表され、求解の都度、隣接ＰＥ間で情報交換が必要となる。z 方向に変数を分割すると考えれば、情報交換すべき圧力の配列要素の個数は送信、受信のそれぞれについて N_xN_y 個である。

(a) red-black SOR 法

通常の SOR 法のアルゴリズムは、以下のように表される（戸川[2]）。ここで、k 回目の反復におけるベクトル $\vec{x}$ の 第 i 成分を $x_i^{(k)}$ と表すことにする。

$x_i^{(0)}(i=1,2,\ldots,n)$ を初期ベクトルの成分とし、$k=0$ として、

(1) $i=0,\ \Delta x_{max}=0$

(2) $i=i+1$

(3) $\tilde{x}_i^{(k+1)} = a_{ii}^{-1}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} \right)$

(4) $x_i^{(k+1)} = x_i^{(k)} + \omega\ (\tilde{x}_i^{(k+1)} - x_i^{(k)})$

$\Delta x_{max} = Max\ (\ |\tilde{x}_i^{(k+1)} - x_i^{(k)}|\ ,\ \Delta x_{max})$

(5) $i<n$ ならば、(2) へ行く。

(6) Δx_{max} が十分に小さい場合は終了。

そうでない場合は、$k+1$ を k と置き直して、(1) に行く。

ここで、総和記号 $\sum$ を含む項は、総和の下限値 $\leq$ 上限値の場合のみ意味を持つものとする。またω は、過緩和係数であり、本報での計算では 1.8 とした。なお、収束に至るまでに必要な反復回数は、過緩和係数によって敏感に変化する場合があるので、他の過緩和係数の値についても今後ケーススタディが必要である。

上記 (3) を見てわかるように通常の SOR 法では、i 番目の x の値を更新するために、直前の計算によって得られた $i-1$ 番目の x の値を必要としている。すなわち、値の更新が漸化式の形で行われるため、そのままではコンパイル時にベクトル化がなされないし、並列処理のための書き換えもできない。

red-black SOR 法では、偶数番目の変数更新のＤＯループと奇数番目の変数更新のＤＯループに分け、２つのＤＯループを実行することで $\vec{x}$ の更新が一巡するようにするものである（別名 odd-even SOR 法と呼ばれる）。有限差分法における格子点をチェッカーボードになぞらえると、赤の升目を計算するときには黒の升目のみを参照し、黒の升目を計算するときは赤の升目のみを参照することになるので、高いベクトル化効率が得られる。また、並列化のためのプログラムの変更も比較的容易である。

(b) PCG 法（前処理付共役勾配法）

PCG 法は、行列 A に近似逆行列を掛け合わせることで条件数を低下させ、その後に CG

法を適用するものである。アルゴリズムは、以下のように記述される。ここで、M は、A の近似行列であり、本報では、

$$M = (L + D)D^{-1}(D + U)$$

のように不完全ＬＵ分解の形で表現されることを仮定する。

$\vec{x}_0$ ：初期ベクトル，$\vec{r}_0 = \vec{b} - A\vec{x}_0$，$\vec{p}_0 = M^{-1}\vec{r}_0$，$k = 0, 1, 2, ...$ に対して、

(1) $\vec{q} = A\vec{p}_k$

(2) $\alpha_k = (M^{-1}\vec{r}_k, \vec{r}_k)/(\vec{p}_k, \vec{q})$

(3) $\vec{x}_{k+1} = \vec{x}_k + \alpha_k\vec{p}_k$

(4) $\vec{r}_{k+1} = \vec{r}_k - \alpha_k\vec{p}_k$
もし、$(M^{-1}\vec{r}_{k+1}, \vec{r}_{k+1})/(M^{-1}\vec{b}, \vec{b}) < \epsilon^2$ ならば終了。

(5) $\beta_k = (M^{-1}\vec{r}_{k+1}, \vec{r}_{k+1})/(M^{-1}\vec{r}_k, \vec{r}_k)$

(6) $\vec{p}_{k+1} = M^{-1}\vec{r}_{k+1} + \beta_k\vec{p}_k$
$k+1$ を k と置き直して、(1) に行く。

PCG 法の場合、並列化の妨げとなるのは、上記 (2) に含まれる $M^{-1}\vec{r}$ の計算である。その理由は以下の通りである。すなわち、実際は、M^{-1} を直接計算することはなく、$M\vec{c} = \vec{r}$ となる $\vec{c}$ を求める。M が不完全ＬＵ分解されていることを利用して、以下の２ステップを解く。

$$(L + D)\vec{y} = \vec{r}$$

$$(I + D^{-1}U)\vec{c} = \vec{y}$$

これらは、それぞれ前進代入・後退代入によって解くことになり、そのままではベクトル化・並列化ができない。不完全ＬＵ分解の並列化のためのオーダリング方法としては、red-black、four-color、zebra、parallel block、van der Vorst などが知られている [3]。

本報では、不完全ＬＵ分解の解法に red-black SOR を適用した。すなわち、以下のようにして解いた。

$\vec{x}_0$ ：初期ベクトル，$\vec{r}_0 = \vec{b} - A\vec{x}_0$，$\vec{p}_0 = M^{-1}\vec{r}_0$，$k = 0, 1, 2, ...$ に対して、

(1) $(L + D)\vec{y} = \vec{r}$ を red-black SOR で解く。

(2) $(I + D^{-1}U)\vec{c} = \vec{y}$ を red-black SOR で解く。

(3) 収束したら不完全ＬＵ分解終了。
そうでなければ (1) に行く。

通常の不完全ＬＵ分解の解法では１度の前進消去・後退消去で済むところを、red-black SOR による不完全ＬＵ分解の解法では反復を行うために、全体の浮動小数点演算数が増大する。したがって、ベクトル化・並列化による実行時間の短縮が演算数の増大分を上回らなければ意味がないことに注意されたい。

4. 並列実行のケーススタディと実行結果について

4.1 並列実行のケーススタディの分類について

並列実行のケーススタディは、圧力 Poisson 方程式の並列解法、格子数、ＰＥ数について以下の組み合わせのもとで行った。

4.1.1 圧力 Poisson 方程式の並列解法の種類

（ａ） red-black SOR 法

VPP500 および Paragon で実行した。

（ｂ） PCG 法（不完全ＬＵ分解の解法に red-black SOR 法を適用）

VPP500 で実行した。

4.1.2 格子数

格子数としては、以下の３通りを候補とした。なお、N_x, N_y, N_z は、それぞれ x, y, z 方向の格子分割数とする。

（ａ）$N_x = 101, N_y = 5, N_z = 21$ ：圧力の未知数 10,605 個

（ｂ）$N_x = 101, N_y = 9, N_z = 41$ ：圧力の未知数 37,269 個

（ｃ）$N_x = 101, N_y = 9, N_z = 81$ ：圧力の未知数 73,629 個

なお、時間刻み幅は、上記（ａ）、（ｂ）に対し 0.02 とし、上記（ｃ）に対し 0.005 とした。

（ｃ）で時間刻み幅を小さくしているのは、（ａ）、（ｂ）と同じ時間刻み幅では安定性条件が満たされずに収束しなくなるからである。

Red-black SOR 法を VPP500 で実行したケースでは、上記３通りの格子数について測定を行った。その他のケース（red-black SOR 法を Paragon で実行したケースと PCG 法を VPP500 で実行したケース）では、上記（ａ）と（ｂ）のみを測定した。（ｃ）を測定しなかった理由は以下の通りである。

・Red-black SOR 法を Paragon で実行しようとした場合、（ｃ）の格子数では、ＣＰＵ時間の運用上の制約（最大１０時間）を越えてしまうため、そのままでは実行できない。なお、計算途中の状態をファイルに書き出して再スタートするようにプログラムを改造することは可能であるが、今回は改造を見送った。

・PCG 法を VPP500 で実行した場合、（ｃ）の格子数では収束に至らなかった。その理由等については４．３節で検討する。

なお、以下の説明で格子数を引用するときは、煩雑さを避けるため、「（ａ）$101 \times 5 \times 21$」、「（ｂ）$101 \times 9 \times 41$」、「（ｃ）$101 \times 9 \times 81$」と記す。

4.1.3 ＰＥ数

ＰＥ数は、1, 4, 8, 12, 16 とした。上限が 16 である理由は、以下の通りである。

上記「格子分割数」に挙げた格子分割のケースでは、ＰＥ数が 32, 64, … の場合に、３次元７点差分で参照する隣接ＰＥの数が２を越えてしまうためである。すなわち、７点差分の中心にある圧力 $p_{i,j,k}$は、以下の６点を参照する。

$$p_{i-1,j,k},\ p_{i,j-1,k},\ p_{i,j,k-1},\ p_{i,j,k+1},\ p_{i,j+1,k},\ p_{i+1,j,k}$$

いわゆる辞書式順序付けでｘ，ｙ，ｚの順に内側からループが回るとした場合、$p_{i,j,k}$から見た上記６点の一次元配列における相対位置は、以下のようになる。

$$-N_xN_y,\ -N_y,\ -1,\ +1,\ +N_y,\ +N_xN_y$$

あるＰＥが担当する圧力 $p_{i,j,k}$の計算に必要な$\pm N_xN_y$ だけ離れた点は、少なくとも隣のＰＥの担当する範囲にあった方がよい。そうでないと、VPP500 の場合は"OVERLAPFIX 文"と呼ばれる隣接ＰＥ間の通信のためのコマンドが使えないことになり、Paragon の場合は、特定ＰＥから見た通信相手のＰＥ数が増大するので通信時間が増大するとともにプログラミングが煩雑になるからである。上記の条件を成立させるためには、N_xN_y-1 が隣接ＰＥの担当する格子数を越えてはならない。

例えば、「(a) $N_x=101, N_y=5, N_z=21$ ：圧力の未知数 10,605 個」の場合をとりあげると、未知数を各ＰＥに等分配するとし、

ＰＥ数 16 の場合は、以下のように条件が満たされるが、

$$N_xN_y = 505 < 10605/16 \approx 662.8$$

ＰＥ数 32 の場合は、以下のように条件が満たされない。

$$N_xN_y = 505 > 10605/32 \approx 331.4$$

なお、VPP FORTRAN では、"OVERLAPFIX 文"で転送する要素の数（隣接ＰＥ間で情報交換される要素の数）と"LOCAL 文"でＰＥに割り当てられる要素の数（ＤＯループが手続き分割されるときのＰＥが計算を担当する要素の数）の整合性のチェックは行っていないので、アプリケーション・プログラム側でチェックする必要がある。

4.1.4 その他の条件

実行機種とコンパイラは以下の通りである。なお、コンパイルそのものはフロント・エンドのサーバでクロスコンパイルされる。

実行機種	コンパイラ	コンパイラ・オプション
VPP500/42	VPP FORTRAN77 EX/VPP	（単一ＰＥのとき）-Pdt -Wv,-md
		（複数ＰＥのとき）-Pdt -Wv,-md -Wx
Paragon XP/S15	if77	-nx

VPP500の場合は、ベクトル化オプションを有効にしてコンパイルした（"-Wv" オプション）。また、ＰＥが複数／単数によって、ソース・プログラム中のXOCL文の有効／無効を切り替えた("-Wx" オプションの有無)。

Paragonの場合は、プロセッサ間通信にｎｘライブラリを使用することを指定した（"-nx" オプション)。さらにParagonの場合、実行時に仮想記憶を使用しないようにするため、実行コマンドに"-plk"を指定する必要がある。

実行時間の部分的計測方法は、VPP500およびParagonの両方に対して共通である。すなわち、ソースプログラム上の測りたい区間の前後でメーカー提供の実行時間計測ルーチン（VPP500の場合はサブルーチンgettod、Paragonの場合は関数dclock）を呼び、戻り値の区間差分（ラップ・タイム）のループ中の累積をもって、当該部分の実行時間とみなした。

複数ＰＥによる並列実行の場合、メイン・プロセッサにおける計測時間で代表させた。つまり、複数ＰＥでの平均値ではない。また、アイドリング時間を切り出した計測は行っていない。つまり、計測した実行時間には、複数ＰＥ間で同期をとるためのアイドリング時間が含まれたものになっている。

4.2 Red-black SOR 法の並列実行結果

Red-black SOR 法に基づくプログラムを VPP500 および Paragon で実行したときの実行時間（Elapsed time）について計測した。本節で引用するすべての表では次の項目の計測時間を示している。

(1) 事前の収束判定　　(表中'Err.estim.1')

「事前の収束判定」は、圧力方程式の残差ベクトルを計算し、もし収束していない（最も絶対値の大きな成分が所定の値より大きい）場合は (2) に続き、収束している場合は (2)～(7) をスキップして、次の時間ステップでの速度場の計算に抜けるものである。

(2) 赤黒法の「赤－計算部分」(表中'Red calc.')

SOR 法のＤＯループの奇数番目をストライド２で実行する。

(3) 赤黒法の「赤－通信部分」(表中'Red comm.')

上記計算結果を隣接ＰＥ間で交換する。VPP500 では"OVERLAPFIX"を用いた。Paragon では、同期型通信を行う"CSEND"/"CRECEIVE"を用いた。

(4) 赤黒法の「黒－計算部分」(表中'Black calc.')

SOR 法のＤＯループの偶数番目をストライド２で実行する。

(5) 赤黒法の「黒－通信部分」(表中'Black comm.')

上記計算結果を隣接ＰＥ間で交換する（上記 (3) と同様である）。

(6) 事後の収束判定　　(表中'Err.estim.2')

「事後の収束判定」は、圧力方程式の残差ベクトルを計算し、収束していない場合は上記 (2) にループし、収束している場合は、下記 (7) に抜けるものである。

(7) 分割実行した部分の共有化 (表中'Data bind.')

「分割実行した部分の共有化」とは、ＰＥ毎に部分的に求めたデータを互いに通信しあって、全ＰＥが同じデータを持つようにすることである。その理由は、圧力方程式の解法以外の部分では、全ＰＥで同じ計算を重複実行するからである。　VPP500 では"SPREAD MOVE"を用いた。Paragon では、非同期型通信を行う"ISEND"/"IRECEIVE"を用い、メインＰＥとそれ以外のＰＥとの間でスター型のデータ交換を行った。

(8) 上記以外　　(表中'others')

「上記以外」は、圧力方程式の解法以外のプログラム部分を指す。

VPP500 上で、格子数「(ａ) $101\times5\times21$」、「(ｂ) $101\times9\times41$」、「(ｃ) $101\times9\times81$」とした場合の上記部分についての実行時間の内訳とＣＰＵ時間、ベクトルユニット時間（ＶＵ時間）を Table 4.2.1, 4.2.2, 4.2.3 に示す。実行時間およびその内訳をグラフ化したものを Fig.4.2.1, 4.2.2, 4.2.3 に示す。これら３通りの格子数について実行時間の合計値を、Table 4.2.4, Fig.4.2.4 に示す。

Table 4.2.4には、反復回数(iter.)も示した。なお、反復回数は格子分割の仕方（と時間刻み幅）のみに依存し、ＰＥ数や計算機（VPP500,Paragon）に依存せずに共通の値をとっている。

VPP500では、ＰＥ数が、１，４，８…と４ＰＥずつ増えているにもかかわらず、実行時間が短縮化していく比率は１／４にほど遠い状況にある。それどころか、格子数「（ａ）$101\times5\times21$」の場合のＰＥ数8～16、格子数「（ｂ）$101\times9\times41$」の場合のＰＥ数12～16の実行時間は、増加の局面にある。格子数「（ｃ）$101\times9\times81$」の場合は、かろうじて実行時間は減少の傾向にあるが、ＰＥ数が８以上の場合の改善はごくわずかである。スケーラビリティの有無をより明確化するために、スピードアップ率をTable 4.2.5, Fig.4.2.5に示す。ＰＥ数の増大とともにスピードアップ率は飽和し、ＰＥ数が１６のあたりではいずれの格子数の場合も水平または下り坂（右下がり）になってきていることから、「スケーラブルでない」と判断できる。その主要な原因は、以下の２つが考えられる。

(1) 通信時間の増加

第１の原因は、ＰＥ数が増えるにつれて通信時間が増加するためである。すなわち、Table 4.2.1, 4.2.2, 4.2.3において、"Data bind."すなわち、「分割実行した部分の共有化」に要する時間はＰＥ数に対してほぼ線形に変化している。このことは、「分割実行した部分の共有化」に用いた"SPREAD MOVE" 文は、同時通信ではなく、１ＰＥずつのデータ収集・分配を行っていることを伺わせる。この部分の時間が全体の時間に占める割合はＰＥ数が増えるにつれて比較的大きくなるので、たしかにスケーラビリティを損なう原因となっている。

一方、"Red comm."と"Black comm."すなわち、red-black SORを実行した後の更新データを隣接ＰＥ間で交換するために要する時間はＰＥ数によらない。すなわち、格子数（ａ）、（ｂ）、（ｃ）に対して、ＰＥ数にほとんど関係せずそれぞれ約 1.8、7.0、15.0 秒という一定値をとっている。つまり、隣接ＰＥ間での一対一通信が同時並行的に行われていることを表している。これらの時間をTable 4.2.4に示す反復回数（iter.）およびメッセージ長さ（$N_x * N_y * 8$）で割ると"Red comm."または"Black comm."の反復１回あたりの１バイト転送に要する時間は、それぞれ秒単位で $0.98 * 10^{-10}$，$0.96 * 10^{-10}$，$0.78 * 10^{-10}$ となり、類似の値をとる。すなわち、ＰＥ数４～１６程度の範囲では、１反復１バイトあたりの"Red comm."または"Black comm."に要する時間は、10^{-10} 秒程度であり、ＰＥ数への依存性は殆どない。

(2) ベクトル化効率の低下

通信時間（"Red comm."、"Black comm."、"Data bind."）を除外した時間についてのスピードアップ率をTable 4.2.6, Fig. 4.2.6に示す。

この数字、つまり「通信時間を除外した時間についてのスピードアップ率」自体は通常のユーザにとって実用的に意味がないが、これを見ることでスケーラビリティを確保できない原因が通信時間の増大だけではないことが明らかとなる。

この図から、通信時間を除いているにもかかわらず、ＰＥ数が大きくなるにつれてグラフの傾きが小さくなってきていることがわかる。実際、Table 4.2.1, 4.2.2, 4.2.3 で、計算に要した時間（表中"Red calc."、"Black calc."）を見ると、ＰＥ数が４個ずつ増えても計算時間が１／４の比で減少していない。つまり計算の効率そのものが落ちてきている。

次にＣＰＵ時間とＶＵ（ベクトルユニット）時間を見てみる（Table 4.2.1, 4.2.2, 4.2.3）。これらの表で、"CPU-time","VU-time"はいずれも全ＰＥでの合計値を表している。ベクトル化効率"VU/CPU"は"VU-time"と"CPU-time"の比を計算したものである。いずれの格子数の場合も、ベクトル化効率はＰＥ数が増えるごとに劣化している。また、同じＰＥ数に対しては、格子数が大きいほどベクトル化効率は高い。したがって、ＤＯループの分割によって、個々のＰＥの実行するベクトル演算のベクトル長が短くなるが、格子数が多くて余裕があるときはベクトル化効率の低下は緩和されるということがわかる。

次に Paragon での実行結果について述べる。格子数「（ａ）101×5×21」、「（ｂ）101×9×41」の場合の実行時間の内訳を Table 4.2.7, 4.2.8 に示す。また、これらをグラフ化したものを Fig.4.2.7, 4.2.8 に示す。4.1 節で述べたように、格子数「（ｃ）101 × 9 × 81」の場合は、ＣＰＵ時間が運用の上限を越えるため実行できなかった。上の２通りの格子数について、全実行時間をまとめたものを Table 4.2.9, Fig.4.2.9 に示す。Table 4.2.9 に含まれる反復回数 (iter.) は、同一格子分割に対しては Table 4.2.4 と同じ値をとっている。Table 4.2.9 を Table 4.2.4 と比べると、実行時間そのものは、Paragon のほうが VPP500 よりも、ＰＥ数が１のときで１５０倍以上、ＰＥ数が１６のときでも５０倍以上かかっている。

スピードアップ率を Table 4.2.10, Fig.4.2.10 に示す。Fig.4.2.10 を見ると、Paragon でもＰＥ数の増大とともにスピードアップ率が飽和する傾向が現れているが、その劣化の仕方は VPP500 ほどではない。Table 4.2.10 を Table 4.2.5 と比べると、Paragon のほうが VPP500 よりも、ＰＥ数４～１６の範囲で２～３倍程度高いスピードアップ率を持っていることがわかる。

Paragon において、スケーラビリティの阻害要因は、ＰＥ間の通信時間の増大にのみ依っている。すなわち、「通信時間（"Red comm."、"Black comm."、"Data bind."）を除外した時間についてのスピードアップ率」（Table 4.2.11, Fig. 4.2.11）が、ほぼ完全にスケーラブルだからである。

4.3 PCG 法の並列実行結果

PCG 法に基づくプログラムを VPP500 で実行した結果について述べる。なお、ここで用いた PCG 法は、不完全ＬＵ分解に red-black SOR 法を適用したものであり、以下単に「PCG 法」と呼ぶが、普通の PCG 法と特に区別する場合は「red-black PCG 法」と呼ぶことにする。時間の測定位置は、3.2 節に述べた PCG 法のアルゴリズムに対応して下記のようにした。なお、「表中」とは、「本節に取り上げる表中」の意味である。

(1) 事前の収束判定 (表中'Err.estim.1')

「事前の収束判定」は、圧力方程式の残差ベクトルを計算し、もし収束していない（最も

絶対値の大きな成分が所定の値より大きい）場合は最初の不完全ＬＵ分解を解いた後 (2) に続き、収束している場合は (2)～(7) をスキップして、次の時間ステップでの速度場の計算に抜けるものである。

(2) α 等の計算 (表中'alpha,x & r')

下記の諸量の計算に要する時間である。

$\vec{q} = A\vec{p}_k$

$\alpha_k = (M^{-1}\vec{r}_k, \vec{r}_k)/(\vec{p}_k, \vec{q})$

$\vec{x}_{k+1} = \vec{x}_k + \alpha_k\vec{p}_k$

$\vec{r}_{k+1} = \vec{r}_k - \alpha_k\vec{p}_k$

(3) 不完全ＬＵ分解に係る通信 (表中'commu.')

前処理行列 M^{-1} に対し、不完全ＬＵ分解を仮定して $M^{-1}\vec{r}_{k+1}$ を red-black SOR を用いて解く場合の、通信に要する時間である。なお、「赤」と「黒」の通信時間は合計してある。

(4) 不完全ＬＵ分解に係る計算 (表中'incom. LU')

同上の計算に要する時間である。

(5) p 等の計算 (表中'p')

下記の諸量の計算に要する時間である。

$\beta_k = (M^{-1}\vec{r}_{k+1}, \vec{r}_{k+1})/(M^{-1}\vec{r}_k, \vec{r}_k)$

$\vec{p}_{k+1} = M^{-1}\vec{r}_{k+1} + \beta_k\vec{p}_k$

(6) 事後の収束判定　　(表中'Err.estim.2')

「事後の収束判定」は、圧力方程式の残差ベクトルを計算し、収束していない場合は上記 (2) にループし、収束している場合は、下記 (7) に抜けるものである。

(7) 分割実行した部分の共有化 (表中'Data bind.')

「分割実行した部分の共有化」とは、ＰＥ毎に部分的に求めたデータを互いに通信しあって、全ＰＥが同じデータを持つようにすることである。

(8) 上記以外　　(表中'others')

「上記以外」は、圧力方程式の解法以外のプログラム部分を指す。

Red-black PCG 法を VPP500 上で、格子数「(a) $101 \times 5 \times 21$」、「(b) $101 \times 9 \times 41$」として実行した場合の上記部分についての実行時間の内訳とＣＰＵ時間、ベクトルユニット時間（ＶＵ時間）を Table 4.3.1, 4.3.2 に示した。実行時間およびその内訳を Fig.4.3.1, 4.3.2 に示した。また、これら２通りの格子数についての実行時間の合計値を Table 4.3.3, Fig.4.3.3 に示した。

格子数「(a) $101 \times 5 \times 21$」の場合は、並列化のメリットはほとんどない。ＰＥ数が１のときの実行時間の大部分が、不完全ＬＵ分解を解くところで費やされている（Table 4.3.1 の'incom.LU')。ＰＥ数が４以上では、不完全ＬＵ分解を解くことに要する時間はほとんど一律に６０％前後に減

少するが、不完全ＬＵ分解に red-black SOR を適用することによる通信時間の増大が丁度計算時間の短縮分を埋めてしまい、全体の時間はほとんど変わらなくなってしまっている（Table 4.3.1 の'incom.LU'）。ＰＥ数４以上では、ＰＥ数の増大とともにベクトル化率が減少していることと（Table 4.3.1 の'VU/CPU(%)'）、不完全ＬＵ分解を解くことに要する時間があまり変化しないことから（Table 4.3.1 の'incom.LU'）、ベクトル化率減少とＰＥ数増大とが互いに相殺しあっていることがわかる。

格子数「（ｂ）$101 \times 9 \times 41$」の場合は、ＰＥ数が１から４に増えることで実行時間が 553 秒から 301 秒に減少しておりやや効果はあるものの、それ以上のＰＥ数ではほとんど効果が見られない（Table 4.3.2, Fig.4.3.2）。

格子数「（ｃ）$101 \times 9 \times 81$」の場合は、計算が収束にいたらなかった。富士総合研究所（[3]）の報告書でも、不完全ＬＵ分解に red-black SOR を適用して、収束にいたらなかった結果を引用しており、原因として前処理行列が正定値になっていないためであろうという推定を紹介している。しかしながら、この推定では、今回のケースのように、同じ疎行列の構造を持つ前処理行列が格子数の違いによって収束したりしなかったりすることを十分に説明できないため、今後詳細な分析が必要である。

実行時間の大きさそのものは、同じ格子数、ＰＥ数に対して、PCG 法の方が SOR 法よりも２～３倍程度かかっており（Fig.4.3.3）、PCG 法にとって不利な結果となっている。スピードアップ率を Table 4.3.4, Fig.4.3.4 に示す。これを見ると、格子数「（ａ）$101 \times 5 \times 21$」、「（ｂ）$101 \times 9 \times 41$」のいずれの場合も、並列化の効果が薄いことが明白である。通信時間を除いたスピードアップ率（Table 4.3.5, Fig.4.3.5）を見ても改善の度合いがわずかであることから、ＰＥ数４以上での並列化の効果を阻害している原因は、通信時間よりもＤＯループ分割によるベクトル化効率の低下にあることがわかる。

次に Table 4.3.6, Fig.4.3.6 に、ＰＥ数が１に限定された場合の３種類のプログラムすなわち、「並列化を行う前の PCG 法」（Table 4.3.6 中'original'、以下「オリジナル」）、「不完全ＬＵ分解に red-black SOR を導入したがベクトル化のチューニングが十分でないもの」（Table 4.3.6 中'modified(no tune)'、以下「チューニングなし」）、「不完全ＬＵ分解に red-black SOR を導入し、ベクトル化のチューニングを行ったもの」（Table 4.3.6 中'modified(vect.)'、以下、「ベクトル化」）の実行時間を示す。２番目の「チューニングなし」とは、不完全ＬＵ分解を解く red-black SOR のＤＯループ中のベクトル化不能のＩＦ文が全体のベクトル化率の低下を招いていたプログラムである。３番目の「ベクトル化」では、その点を改良した。「オリジナル」に対する「チューニングなし」の実行時間および、「チューニングなし」に対する「ベクトル化」の実行時間はそれぞれ約９倍、約 1/10 になっており、結果的に「オリジナル」に対する「ベクトル化」の実行時間は約 0.9 倍である。

ループ回数を見てみると、PCG 法そのものを呼び出すループ回数（Table 4.3.6 '(A) PCG loop'）は、「オリジナル」が 22107 回であるのに比べ、「チューニングなし」が 16021 回、「ベクトル化」が 16123 回という具合にどちらも７割程度に減っている。この違いは、不完全ＬＵ分解を解くときに前進代入・後退代入という直接法を用いるか、red-black SOR のような反復法を用いるかに

よって誤差のたまりかたが違うせいと推定されるが、その立証のためにはこの部分についての誤差解析が必要である。

前処理行列の不完全ＬＵ分解法に要するループ回数の合計値は（Table 4.3.6 '(B) LU fact.')、「オリジナル」が 23107 回であるのに対し、「チューニングなし」が 141971 回、「ベクトル化」が 126101 回という具合にそれぞれ 6.1 倍、5.5 倍に増えている。つまり、「ベクトル化」は、ＬＵ分解法部分の反復回数が数倍程度に増えているにもかかわらず高いベクトル化効率によって実行時間の短縮（約 0.9 倍）を得ていることになる。

前処理行列の不完全ＬＵ分解法に要するループ回数の合計値を PCG 法そのものを呼び出すループ回数で割ることで、前処理行列の不完全ＬＵ分解法が収束する平均ループ回数がわかる（Table 4.3.6 '(B)/(A)')。前進代入、後退代入による場合は、１度の実行で解が確定するが、「オリジナル」でこの値が１でなく 1.05 となっている。すなわち、「オリジナル」では、初期ベクトルに対して１回不完全ＬＵ分解法を行ったあと、Ｎ回の PCG 法のループの中で各１回の不完全ＬＵ分解法を行い、収束した時点で速度場の計算に抜ける、という処理を繰り返しているので、圧力方程式が１回収束するときの前処理行列の不完全ＬＵ分解法に要するループの数が、PCG 法そのもののループ回数よりも１回多くなり、したがって、比率をとると端数が出るのである。「チューニングなし」、「ベクトル化」では、不完全ＬＵ分解法の red-black SOR が、それぞれ約９回、約８回で収束していることがわかる。

本節をまとめると、ＰＥ数が１のときの PCG 法において、前処理行列の不完全ＬＵ分解を解く場合に、１回の前進代入・後退代入のかわりに red-black SOR 法による反復法を適用することで、浮動小数点演算数が数倍程度に増大するが、高いベクトル化率が確保できるので実行時間は、元よりも若干短縮できる。しかし、ＰＥ数が４以上のときは、並列化による計算負荷の分散とベクトル化率の低下とがほぼ完全にトレードオフの関係にあって、並列化のメリットが少ないことがわかる。

5. むすび

圧力方程式を解く red-black SOR 法のサブルーチン部分を並列化し、分散メモリ型ベクトル並列計算機 VPP500 と分散メモリ型スカラ並列計算機 Paragon で実行時間を計測した。また、PCG 法の不完全ＬＵ分解を解く部分に red-black SOR 法を適用した red-black PCG 法による並列化を行い、分散メモリ型ベクトル並列計算機 VPP500 で実行時間を計測した。

Red-black SOR 法を VPP500 で並列実行する場合、通信時間の増大以外に、ＤＯループの細分化によるベクトル化効率の低下がスケーラビリティを阻害する原因となるため、すでに逐次型のベクトル計算機向けにチューニングされたプログラムの場合は、ＰＥ数の増大が単純に実行時間の減少をもたらすわけではないことが判明した。このことは、実行時間から通信時間を除外した時間に基づくスピードアップ率を VPP500 および Paragon について比較することで、前者が依然としてスケーラブルでないのに対し、後者は完全にスケーラブルであることから明確となった。したがって、ベクトル並列計算機の適用が効果的であるのは、ＤＯループ分割によってもベクトル化効率の低下が著しくないような大規模な問題に限られることがわかった。

Red-black PCG 法は red-black SOR 法に比べ、格子数・ＰＥ数により２～３倍の計算時間を要し、格子数が多い場合でもＰＥ数が４以上ではほとんどスケーラブルでないことから、今回のケーススタディに関する限り、VPP500 での実行に適さないことが判明した。しかしながら、定性的な振る舞いに関して３点ほどの不明点が生じている。１つは、格子数が多い場合に収束に至らない場合があるが、その理由および条件がわからないこと。２つめは、不完全ＬＵ分解の解法で前進代入・後退代入による直接法を用いるよりも red-black SOR 法のような反復法を用いた方が PCG 法を呼び出す反復回数が減るが、その理由がわからないこと（不完全ＬＵ分解法部分で red-black SOR 法を適用することで、全体の浮動小数点演算数自体は増大する）。３つめは、計算時間の大部分が前処理行列の不完全ＬＵ分解法に費やされているが、前処理を行わないことによる高速化の可能性が検証されていないことである。

今後の課題として、圧力 Poisson 方程式の並列解法に関しては、red-black SOR 法や red-black PCG 法以外の解法（たとえば、BiCG-STAB 法[6)7)]や前処理なしの CG 法など）の検討、圧力 Poisson 方程式の並列解法以外に領域分割によるプログラム全体の並列化、実行環境として VPP500、Paragon 以外の計算機での試行などが挙げられる。また、上記 red-black PCG 法の要点でも述べたように、個々の解法の収束性・収束条件のより詳細な分析なども必要である。

参考文献

1) 小野謙二、姫野龍太郎. Multi-color SOR 法による非圧縮性コードの収束性の改善. 第７回数値流体力学シンポジウム、(1993).

2) 戸川隼人. マトリクスの数値計算. オーム社、(1972).

3) 富士総合研究所. 超並列コンピューティングの現状と動向調査（第２．０版）. 富士総合研究所、(1994).

4) I.T.Foster. Designing and Building Parallel Programs. Addison-Wesley,(1995).

5) 岩下英俊　他. ＶＰＰ　Ｆｏｒｔｒａｎ：分散メモリ型並列計算機言語. 情報処理学会誌, Vol.36,No.7,pp.1542-1550,(1995).

6) 稲津隆敏　他. BiCGSTAB 系のいろいろなアルゴリズム. 第２回ＮＥＣ・ＨＰＣ研究会,(1995).

7) 村松一弘　他. Cenju-3 上での流体解析とその実時間可視化システム. 第２回ＮＥＣ・ＨＰＣ研究会,(1995).

Table 4.2.1 E-time, CPU-time, VU-time (VPP500 SOR (a) 101*5*21)
(sec.)

No. of PE	1	4	8	12	16
Err. estim. 1	0.7	2.9	2.1	2.0	2.0
Red calc.	10.9	3.6	2.3	1.9	1.6
Red comm.	0.0	1.8	1.7	1.8	1.7
Black calc.	11.0	3.7	2.4	1.9	1.7
Black comm.	0.0	1.8	1.6	1.8	1.7
Err. estim. 2	10.0	5.1	4.2	4.4	4.2
Poisson's eq.	32.6	18.9	14.3	13.8	12.9
others	2.6	5.1	5.0	5.0	4.7
Total E-time	35.2	28.2	25.4	26.5	26.8
CPU-time	35.8	104.9	188.3	292.7	398.4
VU-time	34.5	42.4	52.8	63.3	74.4
VU/CPU(%)	96.4	40.4	28.0	21.6	18.7
CPU/PE	35.8	26.2	23.5	24.4	24.9

Table 4.2.2 E-time, CPU-time, VU-time (VPP500 SOR (b) 101*9*41)
(sec.)

No. of PE	1	4	8	12	16
Err. estim. 1	1.4	7.7	12.1	8.6	10.7
Red calc.	87.1	23.8	12.9	9.2	7.5
Red comm.	0.0	7.0	7.0	5.8	7.0
Black calc.	87.1	24.0	13.1	9.4	7.7
Black comm.	0.0	6.9	7.0	5.9	6.9
Err. estim. 2	79.7	25.9	16.9	14.4	12.8
Poisson's eq.	255.3	95.3	69.0	53.3	52.6
others	8.7	10.4	10.5	11.2	11.1
Total E-time	264.0	116.1	94.5	83.2	85.9
CPU-time	264.2	452.0	732.2	964.9	1330.5
VU-time	262.0	288.1	323.2	357.2	392.9
VU/CPU(%)	99.2	63.7	44.1	37.0	29.5
CPU/PE	264.2	113.0	91.5	80.4	83.2

Table 4.2.3 E-time, CPU-time, VU-time (VPP500 SOR (c) 101*9*81)

(sec.)

No. of PE	1	4	8	12	16
Err. estim. 1	5.4	22.8	7.5	11.1	9.3
Red calc.	461.1	123.0	64.6	44.9	35.5
Red comm.	0.0	14.5	14.6	19.4	15.0
Black calc.	467.5	123.0	64.8	45.2	35.6
Black comm.	0.0	13.8	14.0	19.1	16.6
Err. estim. 2	433.9	122.1	71.8	55.9	48.4
Poisson's eq.	1367.9	419.2	237.3	195.6	160.4
others	33.9	36.6	36.5	37.1	37.3
Total E-time	1401.8	505.7	346.1	322.8	304.6
CPU-time	1399.4	1994.5	2743.4	3828.4	4811.7
VU-time	1393.9	1496.4	1633.5	1766.5	1907.7
VU/CPU(%)	99.6	75.0	59.5	46.1	39.6
CPU/PE	1399.4	498.6	342.9	319.0	300.7

Table 4.2.4 E-Time (VPP500 SOR) (sec.) (count)

No. of PE	1	4	8	12	16	itr. count
(a) 101*5*21	35.2	28.2	25.4	26.5	26.8	42821
(b) 101*5*41	264.0	116.1	94.5	83.2	85.9	97209
(c) 101*5*81	1401.8	505.7	346.1	322.8	304.6	264685

Table 4.2.5 Speedup (VPP500 SOR)

No. of PE	1	4	8	12	16
(a) 101*5*21	1.00	1.25	1.39	1.33	1.31
(b) 101*5*41	1.00	2.27	2.79	3.17	3.07
(c) 101*5*81	1.00	2.77	4.05	4.34	4.60

Table 4.2.6 Speedup of red-black calc. part (VPP500 SOR)

No. of PE	1	4	8	12	16
(a) 101*5*21	1.00	3.00	4.66	5.76	6.64
(b) 101*5*41	1.00	3.64	6.70	9.37	11.46
(c) 101*5*81	1.00	3.77	7.18	10.31	13.06

Table 4.2.7 E-Time(Paragon SOR (a)101*5*21) (sec.)

No. of PE	1	4	8	12	16
Err. estim.1	121.5	213.9	361.4	532.4	653.6
Red calc.	1877.6	478.6	239.9	160.9	119.9
Red comm.	0.0	20.4	19.5	19.0	19.6
Black calc.	1918.0	475.3	238.7	159.2	118.9
Black comm.	0.0	25.7	23.7	23.6	25.2
Err. estim.2	1636.7	457.1	260.4	196.9	160.3
Data bind.	0.0	3.3	4.9	5.3	6.5
others	321.2	316.5	317.9	320.2	320.8
Total	5875.0	1990.8	1466.4	1417.5	1424.8

Table 4.2.8 E-Time(Paragon SOR (b)101*9*41) (sec.)

No. of PE	1	4	8	12	16
Err. estim.1	418.6	446.4	572.7	721.8	950.7
Red calc.	15044.2	3831.7	1919.2	1282.4	962.4
Red comm.	0.0	74.9	78.0	69.4	78.3
Black calc.	15368.3	3815.3	1911.9	1275.8	958.7
Black comm.	0.0	73.6	67.5	82.2	68.2
Err. estim.2	13044.7	3457.2	1829.2	1283.5	1017.2
Data bind.	0.0	7.7	11.2	12.2	14.6
others	1142.9	1129.6	1130.1	1130.5	1132.0
Total	45018.7	12836.4	7519.8	5857.8	5182.1

Table 4.2.9 E-Time(Paragon SOR) (sec.) (count)

No. of PE	1	4	8	12	16	itr. count
(a)101*5*21	5875	1991	1466	1418	1425	42821
(b)101*5*41	45019	12836	7520	5858	5182	97209

Table 4.2.10 Speedup (Paragon SOR)

No. of PE	1	4	8	12	16
(a)101*5*21	1.00	2.95	4.01	4.14	4.12
(b)101*5*41	1.00	3.51	5.99	7.69	8.69

Table 4.2.11 Speedup of red-black calc. part(Paragon SOR)

No. of PE	1	4	8	12	16
(a)101*5*21	1.00	3.98	7.93	11.86	15.89
(b)101*5*41	1.00	3.98	7.94	11.89	15.83

Table 4.3.1 E-time, CPU-time, VU-time (VPP500 PCG (a) 101*5*21)
(sec.)

No. of PE	1	4	8	12	16
Err. estim.1	8.5	15.8	15.0	16.6	20.6
alpha, x & r	4.0	3.1	2.7	3.4	3.3
commu.	0.0	28.7	27.0	29.8	31.6
incom. LU	100.4	63.7	53.7	68.4	66.9
p	0.9	1.4	1.3	1.6	1.6
Err. estim.2	3.4	2.2	1.7	2.1	2.1
Data bind.	0.0	4.1	6.0	7.6	9.6
others	2.7	4.1	5.5	8.3	6.2
Total	119.9	123.1	112.9	137.8	141.9
CPU-time	119.6	478.9	876.9	1581.8	2204.1
VU-time	117.0	115.3	119.8	149.5	165.2
VU/CPU(%)	97.8	24.1	13.7	9.5	7.5
CPU/PE	119.6	119.7	109.6	131.8	137.8

Table 4.3.2 E-time, CPU-time, VU-time (VPP500 PCG (b) 101*9*41)
(sec.)

No. of PE	1	4	8	12	16
Err. estim.1	37.7	24.7	19.8	10.2	11.4
alpha, x & r	15.5	6.8	5.1	4.7	5.4
commu.	0.0	60.7	61.0	53.5	63.5
incom. LU	474.7	174.5	124.0	110.9	125.6
p	3.2	2.2	2.0	1.9	2.3
Err. estim.2	13.2	5.4	3.7	3.2	3.6
Data bind.	0.0	13.1	18.5	22.1	30.7
others	8.8	13.5	14.0	13.8	14.6
Total E-time	553.1	300.9	248.1	220.3	257.1
CPU-time	540.4	1176.0	1935.3	2584.5	4019.2
VU-time	533.4	545.2	571.5	576.7	720.8
VU/CPU(%)	98.7	46.4	29.5	22.3	17.9
CPU/PE	540.4	294.0	241.9	215.4	251.2

Table 4.3.3 E-Time(VPP500 PCG)

No. of PE	1	4	8	12	16
(a)101*5*21	119.9	123.1	112.9	137.8	141.9
(b)101*5*41	553.1	300.9	248.1	220.3	257.1

Table 4.3.4 Speedup (VPP500 PCG)

No. of PE	1	4	8	12	16
(a)101*5*21	1.00	0.97	1.06	0.87	0.84
(b)101*5*41	1.00	1.84	2.23	2.51	2.15

Table 4.3.5 Speedup of PCG calc. part(VPP500 PCG)

No. of PEs	1	4	8	12	16
(a)101*5*21	1.00	1.56	1.85	1.45	1.49
(b)101*5*41	1.00	2.70	3.80	4.24	3.74

Table 4.3.6 E-Time(VPP500 PCG (b)101*9*41)

	original	modified(no tune)	modified(vect.)
E-time(sec.)	621.9	5401.7	553.1
VU/CPU(%)	8.3	12.1	98.7
(A) PCG loop	22107	16021	16123
(B) LU fact. loop	23107	141971	126101
(B)/(A)	1.05	8.86	7.82

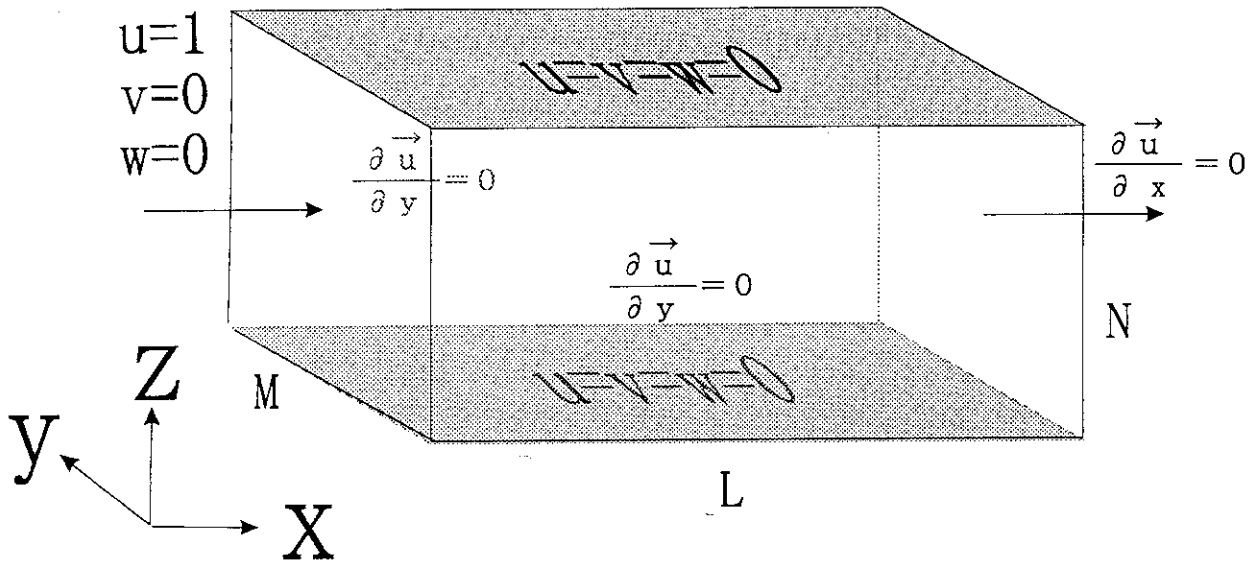

Fig. 2. 2. 1 Computationak region and B. C.

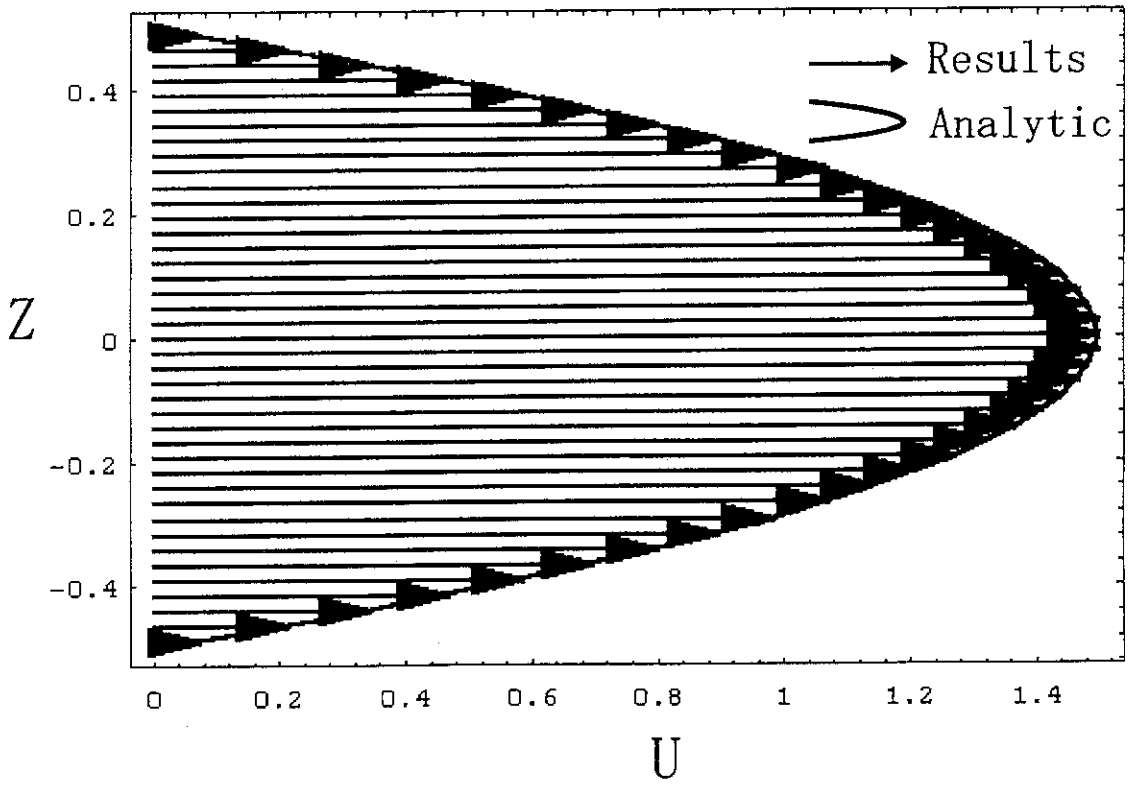

Fig. 2. 2. 2 Velocity profile at outlet

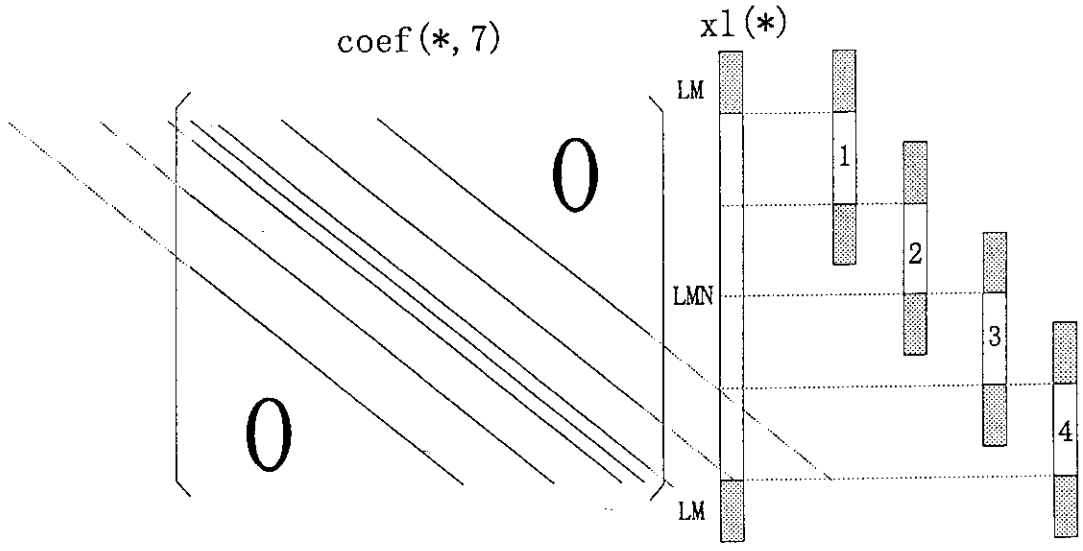

Fig. 3. 2. 1 Pressure array devided into no. of PE.

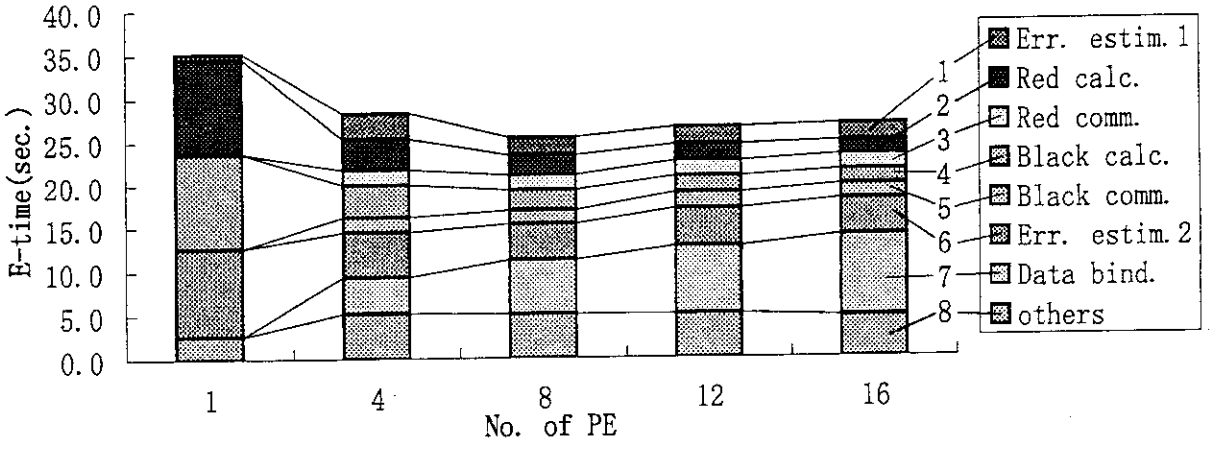

Fig. 4. 2. 1 E-Time (VPP500 SOR (a) 101*5*21)

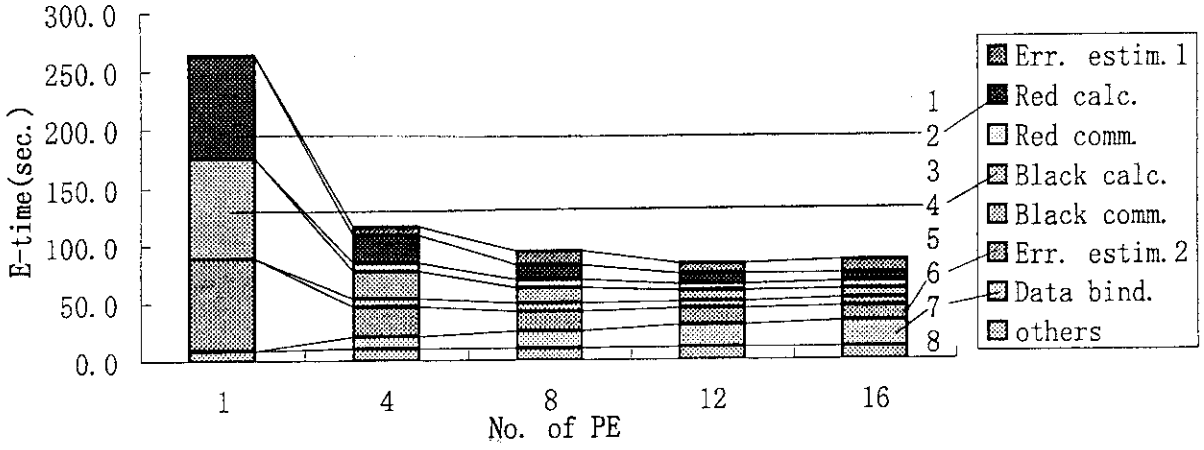

Fig.4.2.2 E-Time (VPP500 SOR (b) 101*9*41)

Sheet1 (2) グラフ 5

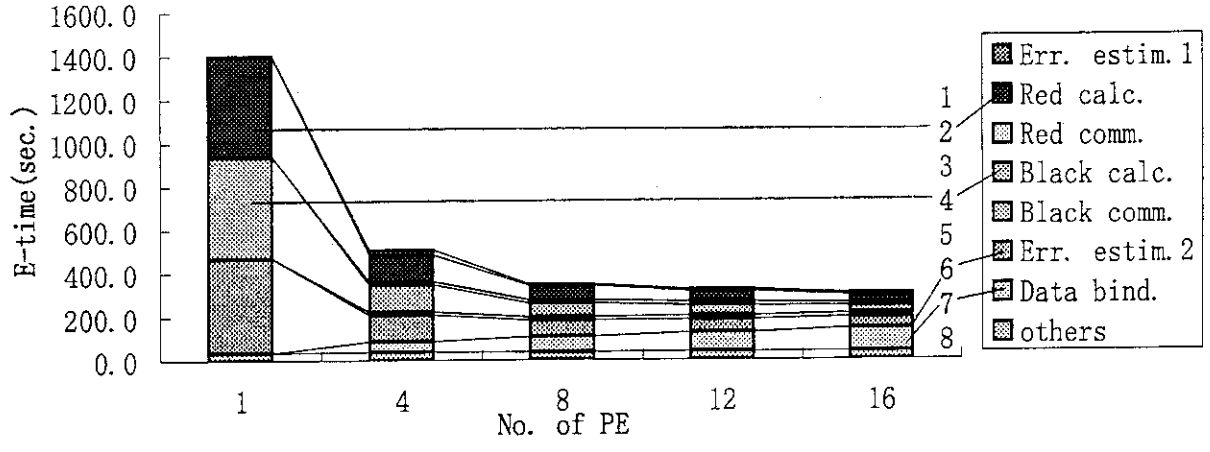

Fig.4.2.3 E-Time (VPP500 SOR (c) 101*9*81)

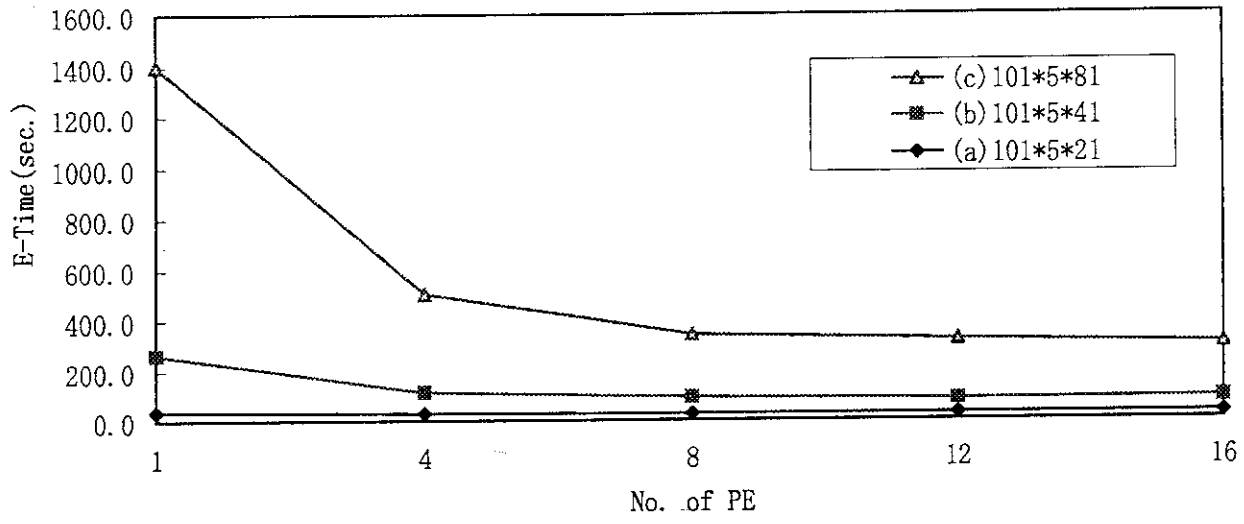

Fig. 4.2.4 E-Time (VPP500 SOR CASE1, 2, 3)

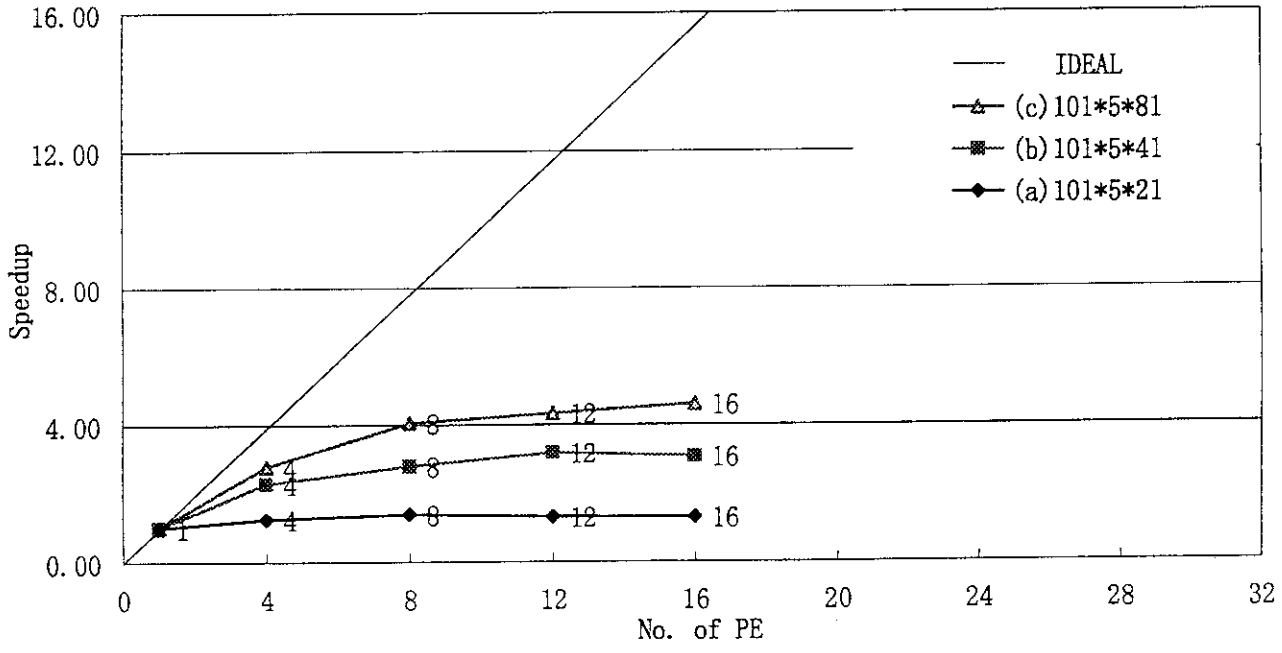

Fig. 4.2.5 Speedup (VPP500 SOR)

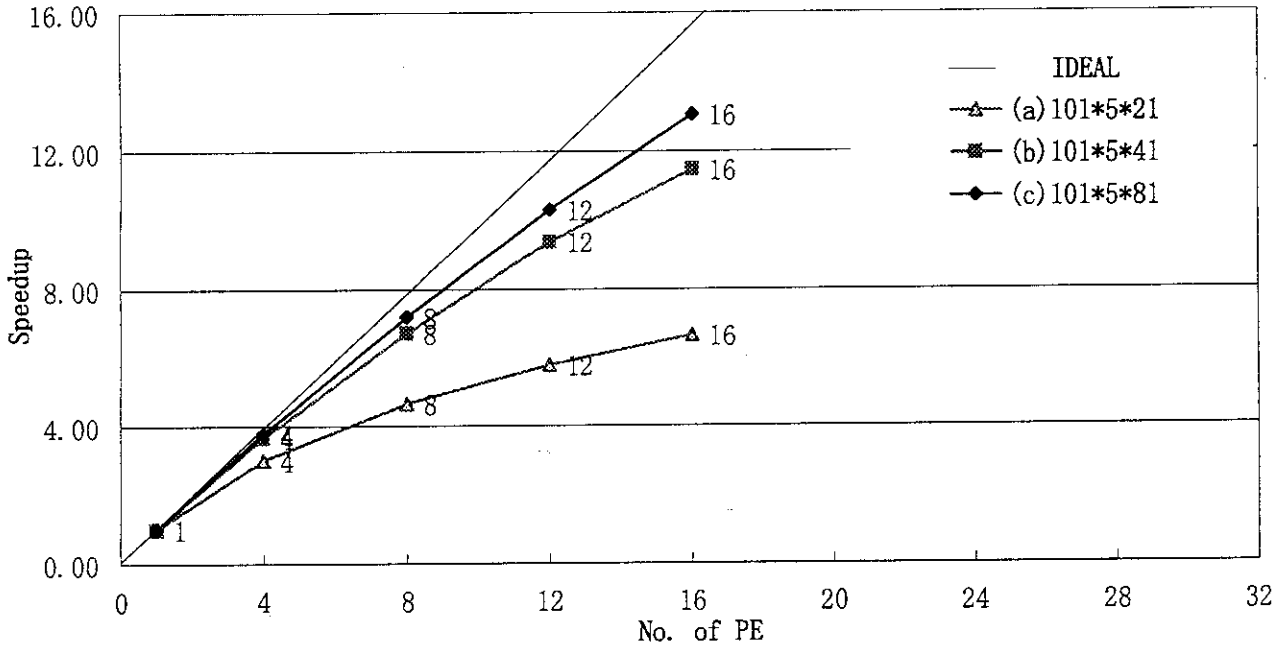

Fig. 4.2.6 Speedup of red-black calc. part (VPP500 SOR)

Sheet1 (3) グラフ 2

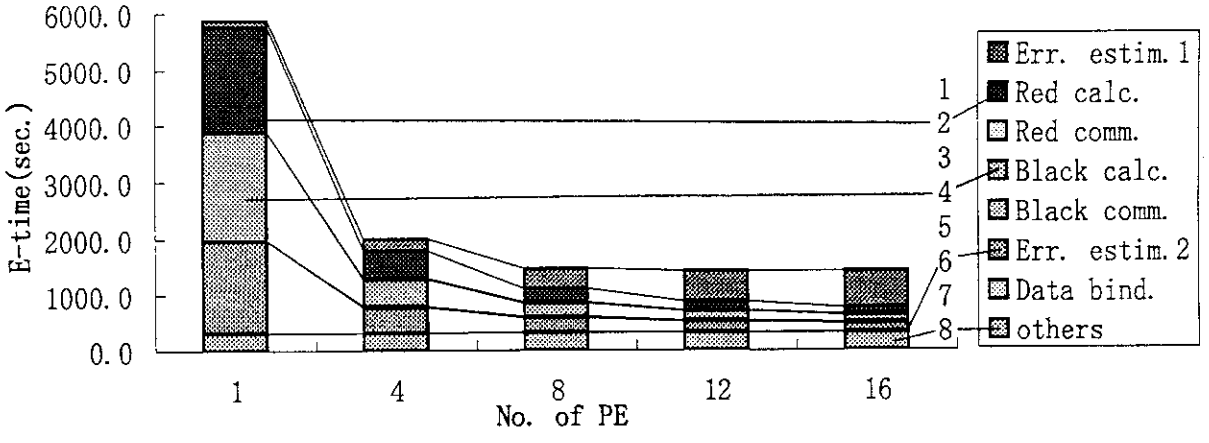

Fig. 4.2.7 E-Time (Paragon SOR (a) 101*5*21)

Sheet1 (3) グラフ 4

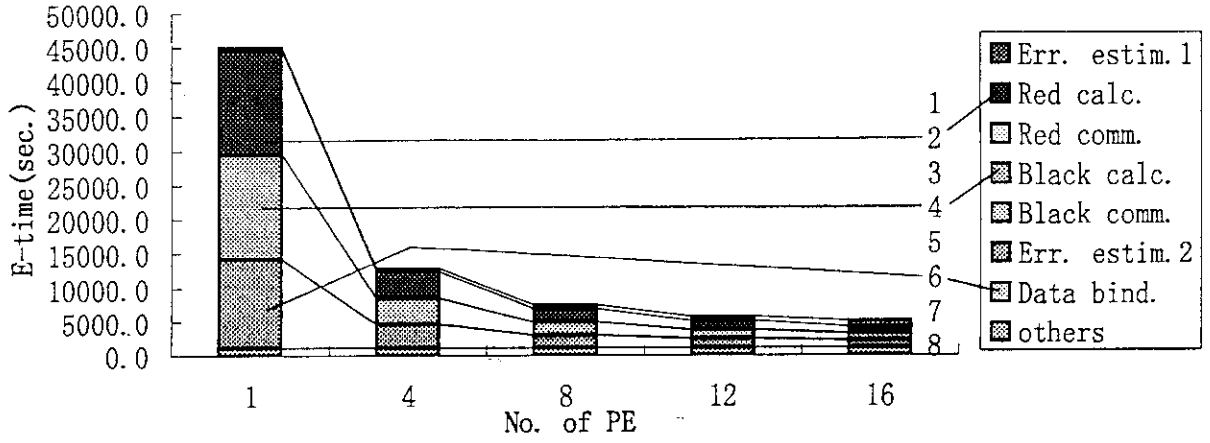

Fig.4.2.8 E-Time (Paragon SOR (b) 101*9*41)

Sheet1 (3) グラフ 6

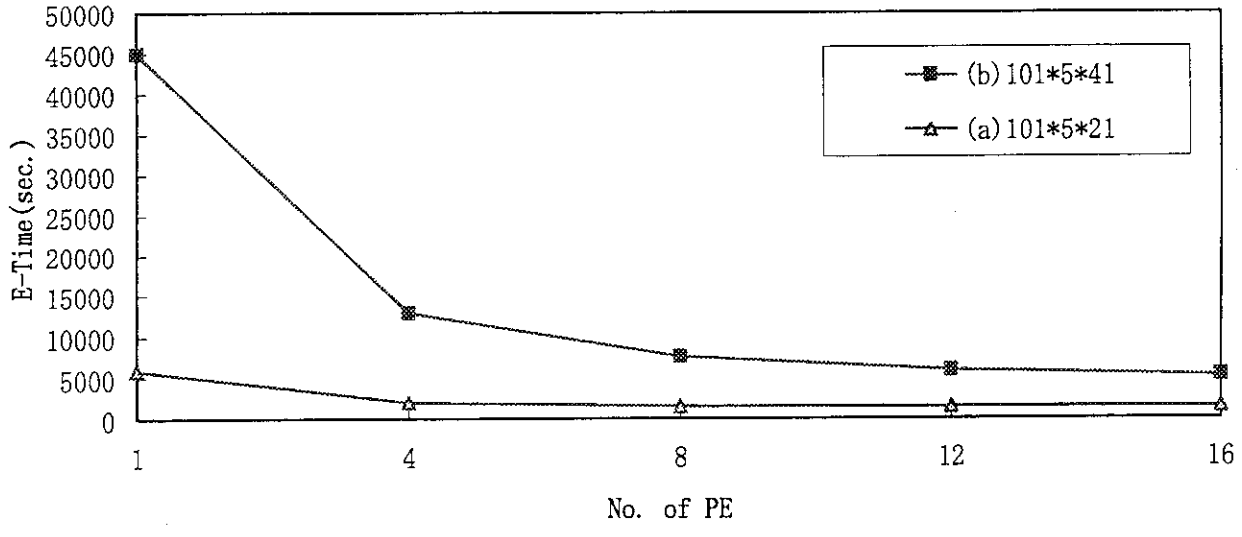

Fig.4.2.9 E-Time (Paragon SOR)

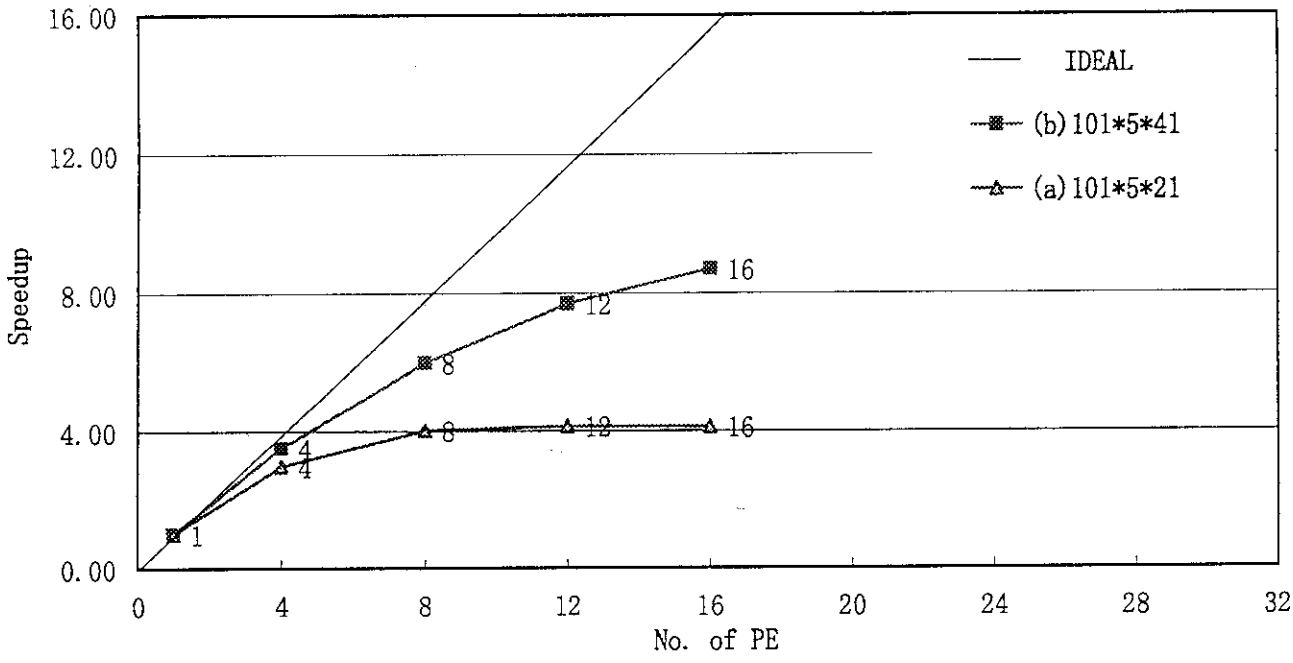

Fig. 4.2.10 Speedup (Paragon SOR)

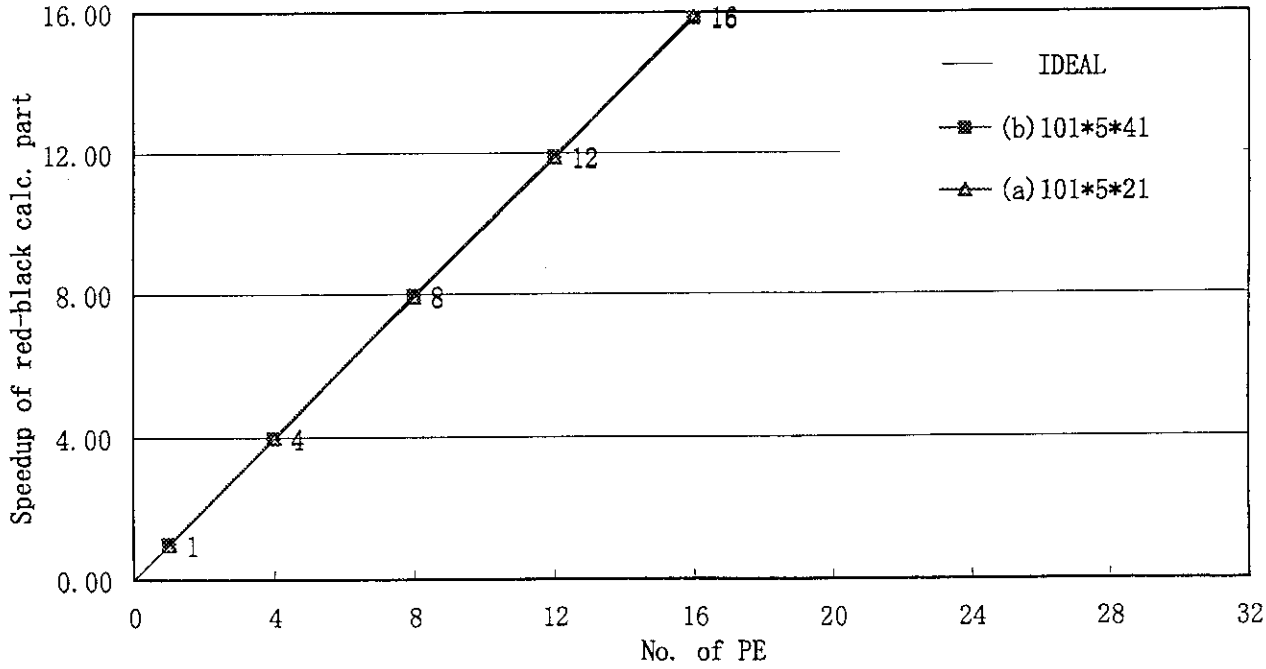

Fig. 4.2.11 Speedup of red-black calc. part (Paragon SOR)

Sheet1 (4) グラフ 2

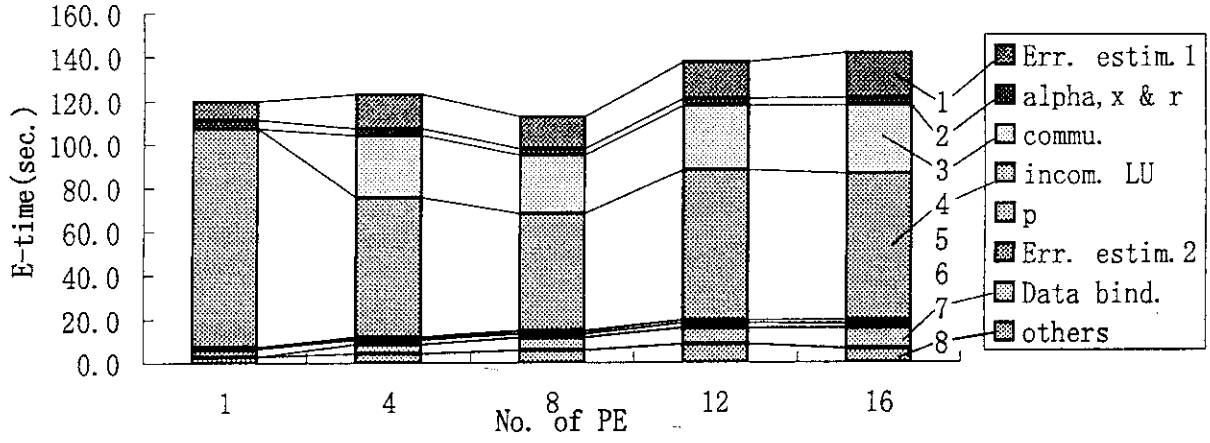

Fig.4.3.1 E-Time (VPP500 PCG (a) 101*5*21)

Sheet1 (4) グラフ 4

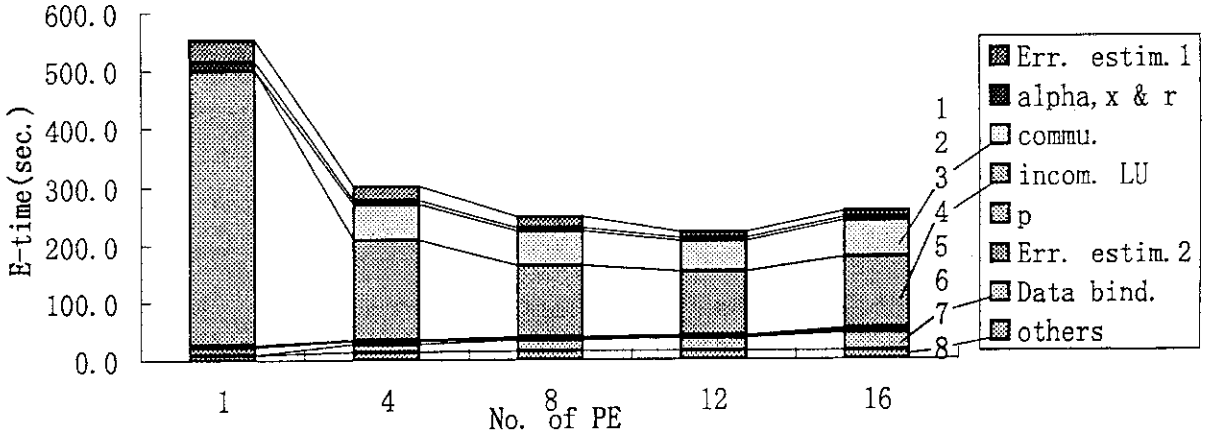

Fig.4.3.2 E-Time (VPP500 PCG (b) 101*9*41)

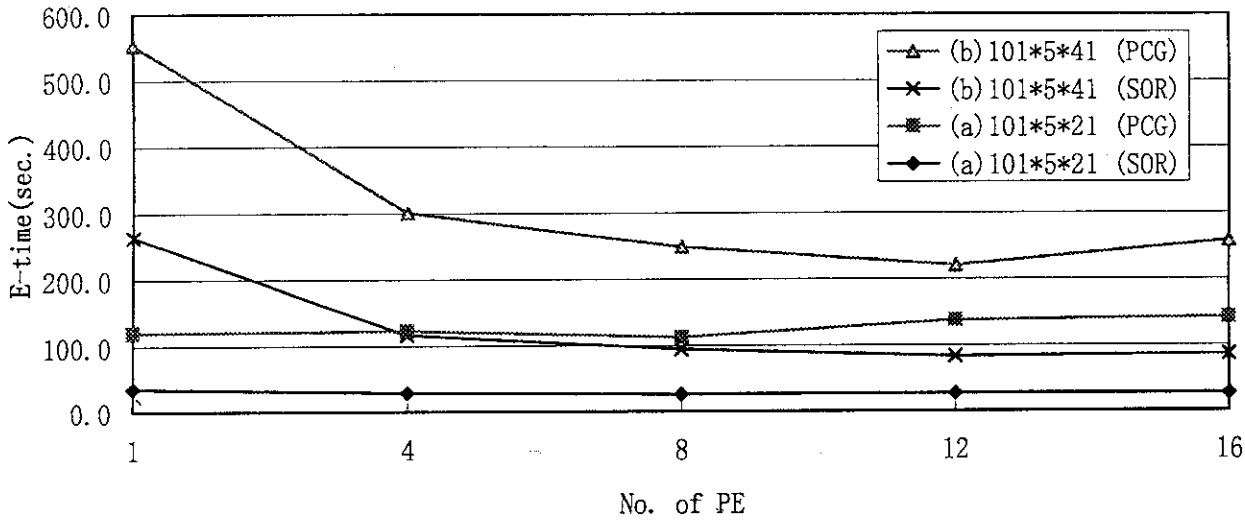

Fig.4.3.3 E-Time (VPP500 PCG & SOR)

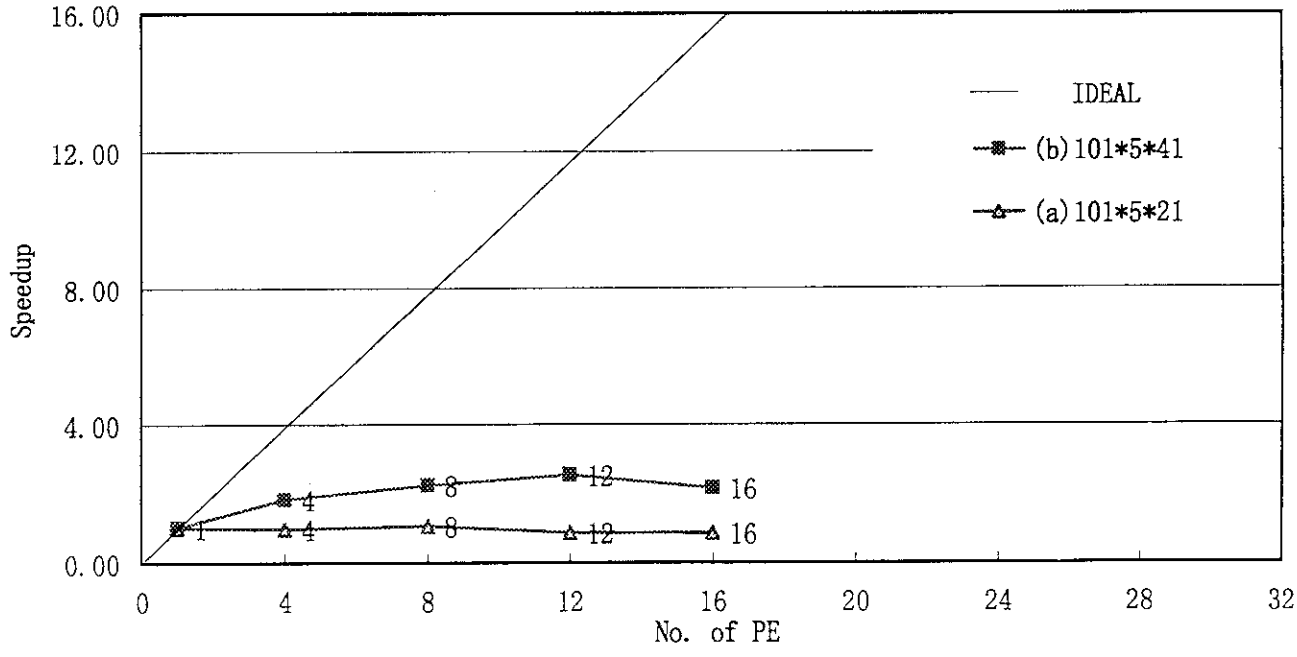

Fig.4.3.4 Speedup (VPP500 PCG)

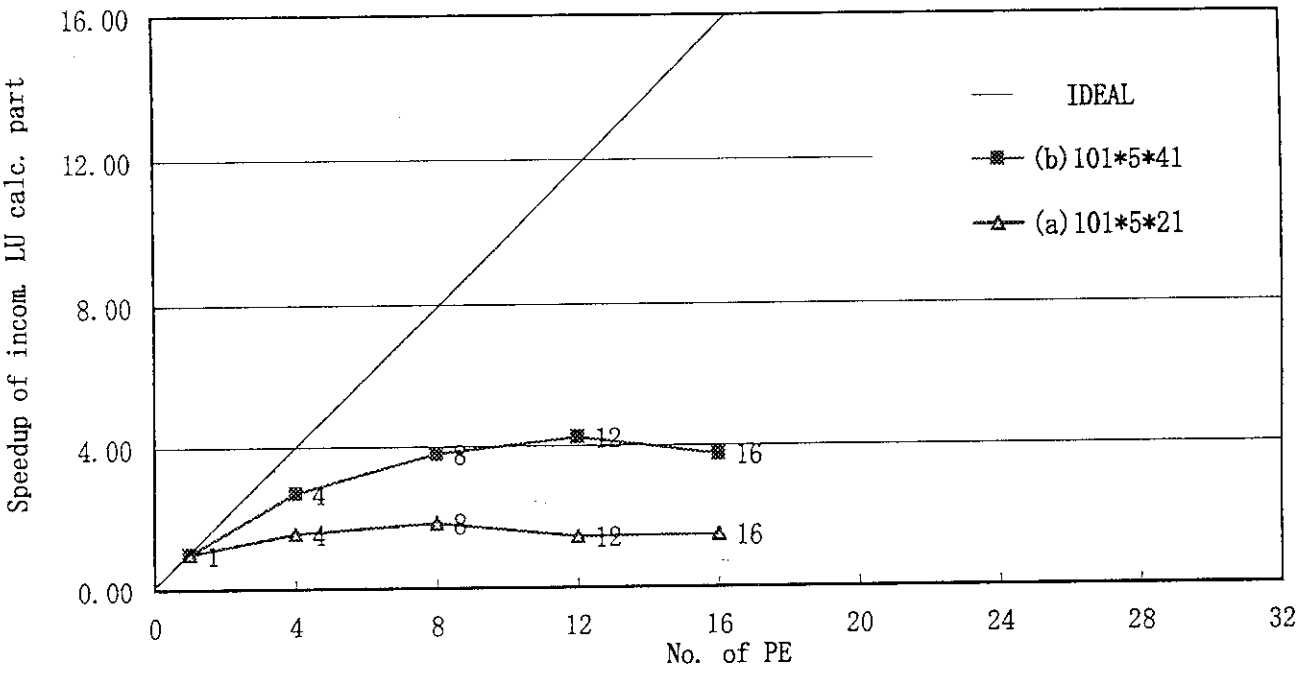

Fig. 4.3.5 Speedup of PCG calc. part (VPP500 PCG)

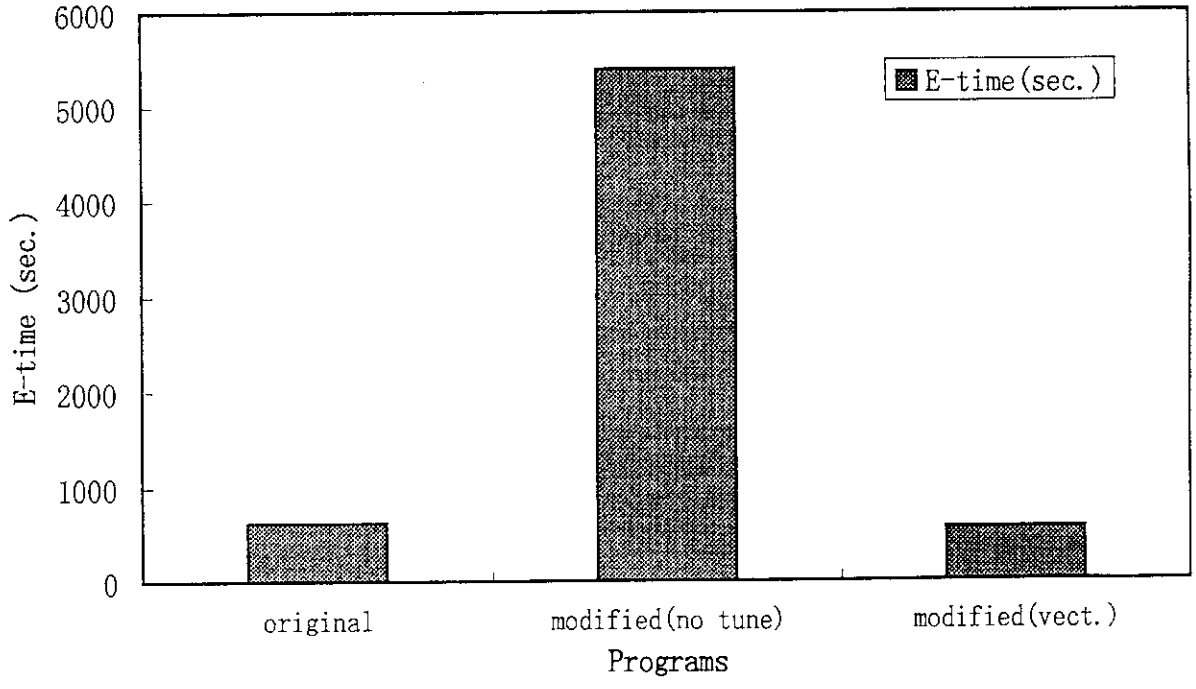

Fig. 4.3.6 Performance comparison of original program and modified ones.

付録：プログラム MiFlow の実行方法

1. モジュール構成

プログラム MiFlow は、以下のモジュールで構成される。

miflow.f　:メインプログラム
+- clearv.f　:ゼロクリア
+- datain.f　:パラメタ設定
+- initvr.f　:よく使われる値ををあらかじめ設定しておく
+- mkcoef.f　:coef(*,7),zcoef(*,7,*) の設定
+- ilvpcg.f　:ハイパープレーンのリストベクトル dd(*) の設定
+- setbnd.f　:境界条件の設定 (u,v,w)
+- calcu1.f　:u1 の計算
+- calcv1.f　:v1 の計算
+- calcw1.f　:w1 の計算
+- setbcv.f　:境界条件の設定 (u1,v1,w1)
+- caluvw.f　:圧力 Poisson 方程式の計算
　+- linsor.f　:SOR natural ordering
　+- lsorrb.f　:SOR red-Black
　+- linpcg.f　:PCG natural ordering
　+- lpcghp.f　:PCG hyperplane
　+- lsor4c.f　:SOR 4-color

なお、以下のファイルは VPP500 および Paragon のための独自仕様を含むため、ファイル識別子を変更して別々に格納している。

［VPP500 向け］
miflow.fvp VPP500 向け
lsorrb.fvp 同上
linpcg.fvp 同上
caluvw.fvp 同上
clearv.fvp 同上
initvr.fvp 同上

［Paragon 向け］
miflow.fpa Paragon 向け（単一ＰＥ、複数ＰＥ共通）
mkcoef.fpa 同上
lsorrb.fpa Paragon 向け（単一ＰＥ専用）
lsorrb.fp1 Paragon 向け（複数ＰＥ専用）

2. インクルードファイル

共通パラメタ、共通変数は、以下のインクルードファイルに定義されている。

(1) ./Include/cmparm (VPP500 と Paragon)
格子分割数および配列の大きさを宣言する。

(2) ./Include/cmvars (VPP500 と Paragon)
流速、圧力などの主要な配列やレイノルズ数などの共通変数を宣言する。

(3) ./Include/cmtime (VPP500 と Paragon)
時間計測のための配列等を宣言する。

(4) ./Include/cmnpe (VPP500 のみ)
ＰＥ数を宣言する。なお、Paragon の場合は、実行時にＰＥ数を指定できるので、このインクルードファイルは不要である。

(5) ./Include/cmlslv1 (VPP500 のみ)
圧力 Poisson 方程式の係数行列と右辺ベクトルを宣言する。なお、Paragon の場合は、この宣言を cmvars に含めているので、このインクルードファイルは不要である。

(6) ./Include/cmlslv2 (VPP500 のみ)
手続き分割のための INDEXPARTITION 文を宣言する。なお、Paragon の場合は、INDEXPARTITION 文が無いので、このインクルードファイルは不要である。

(7) ./Include/cmlslv3 (VPP500 のみ)
PCG 法で用いる作業領域を宣言する。なお、Paragon の場合は、この宣言を cmvars に含めているので、このインクルードファイルは不要である。

3. データの設定方法

(1) 格子数の設定 (VPP500 と Paragon)

インクルードファイル cmparm に含まれるパラメタ l,m,n に x,y,z 方向の格子分割数を指定する。なお、l,m,n は、本文 4.1 節の N_x, N_y, N_z に対応する。

```
parameter ( l= x 方向分割数, m= y 方向分割数, n= z 方向分割数 )
```

List A.1 にインクルードファイル cmparm の例を示す。

(2) ＰＥ数の設定 (VPP500 のみ)

インクルードファイル cmnpe に含まれるパラメタ npe にＰＥ数を指定する。

```
parameter ( NPE =ＰＥ数 )
```

List A.2 にインクルードファイル cmnpe の例を示す。

(3) 手続き分割の設定 (VPP500 のみ)

インクルードファイル cmlslv2 に含まれるパラメタ ia1, ia2, ･･･ に、GLOBAL 文により GLOBAL 配列を各ＰＥに割り当てるときの配列の個数を指定する。また、id1, id2, ･･･ に、ＤＯループをＰＥ毎に分割実行するときのＤＯ変数の範囲に含まれる個数を指定する。また、上記パラメタ ia1, ia2, ･･･ ,id1, id2, ･･･ を用いた INDEX PARTITION 文を宣言する。また、パラメタ ia1, ia2, ･･･ および id1, id2, ･･･ の個数は、ＰＥ数に一致させる。

```
parameter (ia1=GLOBAL 文のインデックス,ia2= ･･･ 以下同様 )
parameter (id1=ＤＯ変数の範囲,id2= ･･･ 以下同様 )
!xocl index partition ind_i = ( pe,index=1:lmn2, &
!xocl part=(ia1,ia2, ･･･ ) )
!xocl index partition ind_do=( pe,index=1:lmn, &
!xocl part=(id1,id2, ･･･ ) )
```

Red-black SOR 法によりストライド２でＤＯループを並列実行するに際しては、全てのＰＥで奇数番目または偶数番目のいずれかに足並みがそろっている必要がある。つまり、ＤＯ変数の範囲に含まれる個数は、「最後のＰＥを除く全てのＰＥにおいて偶数」である必要がある。

次に、GLOBAL 文のインデックスは、以下のようになる。すなわち、

(a) 最初と最後のＰＥについては、上記「ＤＯ変数の範囲に含まれる個数」に $l * m$ の値を加えたものになる。l と m は、上記「格子数の設定」で与えた、「ｘ方向分割数」と「ｙ方向分割数」である。

(b) 最初でも最後でもないＰＥについては、上記「ＤＯ変数の範囲に含まれる個数」と同じ値になる。

具体例で説明する。ＰＥ数が４、格子分割が「(b) l=101, m= 9, n= 41」であるとする。設定すべきパラメタは、ia1, ia2, ia3, ia4 および id1, id2, id3, id4 の各４個ずつとなる。

最初に「ＤＯ変数の範囲」に含まれる個数すなわち、 id1, id2, id3, id4 の設定を考える。格子数は、

$$N = 101 \times 9 \times 41 = 37269$$

であるので、これを４個のＰＥに平均に割り当てるとし、

$$N/4 = 37269/4 = 9317.25$$

ところで、「最後のＰＥを除く全てのＰＥにおいて偶数」である必要があることと、ＰＥごとに割り当てられた格子数の合計が $N = 37269$ である必要があることから、例えば、

$$id1 = 9318,\ id2 = 9318,\ id3 = 9318,\ id4 = 9315$$

のような設定が考えられる。

次に「GLOBAL 文のインデックス」すなわち、ia1, ia2, ia3, ia4 に値を設定する。最初と最後のＰＥについては、「ＤＯ変数の範囲に含まれる個数」に $l * m = 909$ を加えたものとなり、それ以外については同じ値となるので、

$$ia1 = 10227,\ ia2 = 9318,\ ia3 = 9318,\ ia4 = 10224$$

となる。

List A.3 にインクルードファイル cmlslv2 の例として、ＰＥ数が４で格子分割が「(b) l=101, m= 9, n= 41」の場合を示す。紙面の関係で、ＰＥ数が４の場合のみ示しているが、このファイルでは、本報で扱ったＰＥ数と格子数の全ての組み合わせについての定義が含まれており、必要に応じて対応する文のコメントをはずしてコンパイルし直せばよい。

(4) 時間刻み幅、解法ルーチン選択 (VPP500 と Paragon)

サブルーチン datain に含まれる変数に適当な値を代入する。

dt = 時間刻み幅
lpend = ループ回数
ilsolv = 圧力方程式解法ルーチン選択番号

時間刻み幅と空間刻み幅は、以下の式を満たしている必要がある。

$$\frac{u\Delta t}{\Delta x} + \frac{2\alpha\Delta t}{\Delta x^2} \leq 1$$

したがって、

$$\Delta t \leq \frac{1}{\frac{u}{\Delta x} + \frac{2\alpha}{\Delta x^2}}$$

本報のケーススタディでは、以下のようにしている。

「(a) l=101 m=5 n=21」、「(b) l=101 m=9 n=41」の場合、

dt = 2.0e-02

lpend = 20

「(c) l=101 m=9 n=81」の場合、

dt = 0.5e-02

lpend = 40

圧力方程式解法ルーチン選択番号は、以下のように対応している。

1 ：通常の（red-black でない）SOR

2 ：Red-black SOR

3 ：Red-black PCG

4 ：ハイパープレーン・オーダーリングの PCG

5 ：4-color SOR

本報のケーススタディでは、２または３を指定している。

List A.4 にサブルーチン datain での変数設定例を示す。

(5) コンパイル (VPP500 と Paragon)

上記の変更が終了したのち、コンパイルする。

［VPP500 の場合］

```
cp lsorrb.fvp lsorrb.f
cp linpcg.fvp linpcg.f
cp caluvw.fvp caluvw.f
cp clearv.fvp clearv.f
cp initvr.fvp initvr.f
cp miflow.fvp miflow.f
cp Include/cmvars.ivp Include/cmvars
comopt =コンパイルオプション
srcfil =calcu1.f calcv1.f calcw1.f \
       caluvw.f cldata.f clearv.f datain.f ilvpcg.f initvr.f \
       linpcg.f linsor.f lpcghp.f lsor4c.f lsorrb.f \
       miflow.f mkcoef.f setbcv.f setbnd.f
frtpx $(srcfil) -o miflow.exe $(comopt)
```

上記一連のコマンド中、コンパイルオプションは、ＰＥ数が１かそうでないかによって以下のように設定を変える。

ＰＥ数が１のとき、　　comopt =-Pdt -Wv,-md -Wx

ＰＥ数が１より大きいとき、　comopt =-Pdt -Wv,-md

［Paragon の場合］

```
cp miflow.fpa miflow.f
cp mkcoef.fpa mkcoef.f
cp 「lsorrb のソースファイル」 lsorrb.f
srcfil =calcu1.f calcv1.f calcw1.f \
        caluvw.f cldata.f clearv.f datain.f ilvpcg.f initvr.f \
        linpcg.f linsor.f lpcghp.f lsor4c.f lsorrb.f \
        miflow.f mkcoef.f setbcv.f setbnd.f
if77 -nx $(srcfil) -o miflow.exe
```

上記一連のコマンド中、コンパイル対象となる red-black SOR による解法ルーチン lsorrb のソースファイルは、ＰＥ数が１かそうでないかによって以下のように設定を変える。

ＰＥ数が１のとき、　　cp lsorrb.fp1 lsorrb.f

ＰＥ数が１より大きいとき、cp lsorrb.fpa lsorrb.f

(6) 実行　　(VPP500 と Paragon)

以下に、VPP500 で実行する場合のシェル・スクリプトの例を示す。"set npe=ＰＥ数"において、コンパイル時に指定したのと同じＰＥ数を指定する。

qsub コマンドの"-q ジョブクラス" において、ＰＥ数と実行時間に応じた適切なジョブクラスを指定する。また、"-lPv" に続いて、ＰＥ数を格納した変数を指定している。ここに示した例では、ＰＥ数 16、ジョブクラス vpp16 としている。

［VPP500 の場合］

```
set npe=16
set dirnam=miflow
set prg=miflow
set prgnam=$prg$npe
rm $prgnam.txt
echo "y" | rm $prgnam.bat >& /dev/null
echo "y" | rm $prgnam.exe >& /dev/null
cp $prg.exe $prgnam.exe
echo "cd ~/$dirnam " > $prgnam.bat
echo "date " >> $prgnam.bat
echo " $prgnam.exe > " >> $prgnam.bat
qsub -q vpp16 -lPv $npe $prgnam.bat
```

以下に、Paragon で実行する場合のシェル・スクリプトの例を示す。"set nodes＝ＰＥ数"において、ＰＥ数を指定する。

実行オプションの "-plk" は、「仮想記憶を使用しない」ことを示す。また、"-sz" に続いてＰＥ数を格納した変数を指定している。

"qsub -q ジョブクラス" において、ＰＥ数と実行時間に応じた適切なジョブクラスを指定する。ここに示した例では、ＰＥ数 16、ジョブクラス qn128-8 としている。

［Paragon の場合］

```
set nodes=16
set dirnam=miflow
set prgnam=miflow
rm tmp.bat
echo " cd $home/$dirnam " > tmp.bat
echo " date " >> tmp.bat
echo " $prgnam.exe -plk -sz $nodes " >> tmp.bat
echo " date " >> tmp.bat
qsub -q qn128-8 -lT 08:00:00 $home/$dirnam/tmp.bat
```

List A.1 インクルードファイル：cmparm

```
c*
c*      file = Include/cmparm
c*
c*    note.
c*      Remember that you should consider the value of "dt"(datain.f)
c*      if you modify l, m or n value .
c*
case1 parameter ( l=101, m= 5, n= 21 )
      parameter ( l=101, m= 9, n= 41 )
case3 parameter ( l=101, m= 9, n= 81 )
c tst parameter ( l=101, m= 9, n= 48 )

      parameter ( l1=l+1, m1=m+1, n1=n+1, l2=l+2, m2=m+2, n2=n+2 )
      parameter ( lm=l*m, lmn=l*m*n, lmn2=lmn+2*lm, nls=l+m+n+2 )
      parameter ( lm2=lm+2*l )

      implicit real*8 (a-h,o-z)
```

List A.2 インクルードファイル：cmnpe

```
c*
c*      file = Include/cmnpe
c*
c*      note: If you modify the value of NPE ,
c*            remember to modify Include/cmlslv2 too .
c*
      parameter ( NPE =16 )
```

List A.3 インクルードファイル：cmlslv2

```
c*      /* file=cmlslv2.ivp */
cxocl processor pemain(NPE)
cxocl subprocessor pe(NPE)
c*      case 1: l=101, m= 5, n= 21
c*      case 2: l=101, m= 9, n= 41
c*      case 3: l=101, m= 9, n= 81
c*      /* NPE = 4 */
case1 parameter (ia1= 3155,ia2= 2652,ia3= 2652,ia4= 3156)
case1 parameter (id1= 2650,id2= 2652,id3= 2652,id4= 2651)
        parameter (ia1=10227,ia2= 9318,ia3= 9318,ia4=10224)
        parameter (id1= 9318,id2= 9318,id3= 9318,id4= 9315)
case3 parameter (ia1=19317,ia2=18408,ia3=18408,ia4=19314)
case3 parameter (id1=18408,id2=18408,id3=18408,id4=18405)
!xocl index partition ind_i = ( pe,index=1:lmn2, &
!xocl     part=(ia1,ia2,ia3,ia4 ) )
!xocl index partition ind_do=( pe,index=1:lmn, &
!xocl     part=(id1,id2,id3,id4 ) )
c* /* NPE = 8 */
        ...
        （以下、省略）
```

List A.4 インクルードファイル：datain.f

```
      subroutine datain
      include "./Include/cmparm"
      include "./Include/cmvars"
      xlen    =   15.0d0
      ylen    =   1.0d0
      zlen    =   1.0d0
      dx      =   xlen/real(l)
      dy      =   ylen/real(m)
      dz      =   zlen/real(n)
c /* for l=101 m=5 n=21 or l=101 m=9 n=41 */
      dt      =   2.0e-02
      lpend =   20
c    /* for l=101 m=9 n=81 : Attention! Modify "lpend" too ! */
c      dt      =   0.5e-02
c      lpend   =   40
      re = 100.0d0
      uinit = 1.0
      icschm = 1
      ilsolv = 3
c*   & Poisson Solver
c*   &          1 = SOR method with natural ordering
c*   &          2 = SOR method with red-black ordering      (vect.)
c*   &          3 = ILUCG method with natural ordering
c*   &          4 = ILUCG method with hyperplane ordering (vect.)
c*   &          5 = SOR method with 4-color ordering      (vect.)
      lpbgn   =   1
      linner =   50
      maxitr =   199
      ...
      （以下、省略）
```