

データ再分散を行う並列 Gram-Schmidt 再直交化

片桐 孝洋^{†,††}

本報告^aでは Gram-Schmidt 法 (G-S 法) を用いた再直交化処理の並列アルゴリズムにおいて、データ再分散を行うことで列方向分散 (Column-Wise Distribution, CWD) における低い並列性を改善する再直交化方式を提案する。提案方式を用いた並列再直交化を、固有ベクトル計算のための逆反復法に実装した。PC クラスタおよび国産スーパーコンピュータ 3 種 (HITACHI SR8000/MPP, Fujitsu VPP800/63, および NEC SX-5/128M8) を用いて提案方式が実装された並列逆反復法を評価したところ、CWD を用いた従来法に対し、修正 G-S 法を利用した再直交化で最大約 4 倍、古典 G-S 法を利用した再直交化で最大約 3.5 倍の速度向上が得られた。

^a この報告は、ABCLib Working Notes No.5 (2004 年 2 月 10 日登録) であるが、電気通信大学情報システム学研究所技術報告 UEC-IS-2003-7 (2003 年 11 月 6 日登録)、および 2004 年 ハイパフォーマンスコンピューティングと計算科学シンポジウム (HPCS 2004) (2004 年 1 月 15 日発表) 日本科学未来館みらい CAN ホール (臨海副都心)、HPCS 2004 論文集, pp.9-16, にて発表された論文をもとに、加筆・修正・削除を行ったものである。

A Parallel Gram-Schmidt Re-orthogonalization Method Using Data Re-distribution

TAKAHIRO KATAGIRI^{†,††}

In this report, we improve a parallel re-orthogonalization with Gram-Schmidt (G-S) method. A data re-distribution approach is used to solve the low parallelism problem for column-wise distribution (CWD) in the method. The proposed method is implemented to the inverse iteration method for computing eigenvectors. The inverse iteration method using the proposed method is evaluated with a PC cluster and three kinds of super-computers in Japan, which are the HITACHI SR8000/MPP, Fujitsu VPP800/63, and NEC SX-5/128M8. The results of performance evaluation indicated that the maximum speedup factor of 4 in Modified G-S method, and of 3.5 in Classical G-S method with respect to conventional methods using CWD were obtained.

1. はじめに

直交化処理は、固有値計算や QR 分解といったような線形計算を実行する場合においてもっとも基本的、かつ重要な処理の 1 つである¹⁾。それゆえ、多くの研究者が QR 分解などの直交化処理について研究を行っている^{2)~4)}。たとえば S.Oliveira らは QR 分解について、行方向分散方式および列方向分散方式における並列アルゴリズムの性能解析を行っている⁵⁾。一方で D.Vanderstraeten らは、QR 分解において適応的に並列性の高い直交化方式を選択することで並列性の改善をおこなった⁶⁾。

この論文では QR 分解ではなく、Gram-Schmidt

(G-S) 直交化法による再直交化処理の並列化手法について取り扱う。この Gram-Schmidt 再直交化は、データ分散方式、および直交化アルゴリズムの種類 (修正 G-S, 古典 G-S) の違いにより、全く異なる並列性を示すことが知られている⁷⁾。

固有ベクトル計算のための逆反復法 (Inverse Iteration Method, IIM) に G-S 再直交化を適用する場合、対象となる行列の数値特性にも依存するが、全体の実行時間に対し逆反復法の時間が 90%以上を占め、さらにこの逆反復法の時間のうち G-S 再直交化の時間も 90%以上を占めることが報告されており⁸⁾、高速処理が望まれている。この IIM 中の G-S 再直交化を並列化する場合、IIM の処理自体の自然な並列性を抽出するためデータ分散を列方向分散とする方式がよく利用されるが、このデータ分散方式では直交精度と並列性のトレードオフの問題が生じる ことが知られてい

[†] 電気通信大学大学院情報システム学研究所
Graduate School of Information Systems, The University of Electro-Communications

^{††} 科学技術振興機構さきがけ, PRESTO, JST

Twisted Decomposition を利用する、再直交化が不要な逆反

る⁷⁾。

直交精度と並列性のトレードオフは、直交精度が高い修正 G-S 法が列方向分散方式において逐次化されることから生じる。ところが G-S 再直交化アルゴリズム単体で考えると、行方向分散方式では修正 G-S 法、および古典 G-S 法ともに並列性を抽出できることが知られている⁷⁾。しかし残念なことに、行方向のデータ分散を行うと、IIM 中の再直交化以外の処理 (LU 分解、前進・後退代入、および正規化) が逐次化される。このことで、G-S 再直交化における並列性を利用する効果が出ない。

そこで本論文では、IIM が有する自然な並列性を抽出できる列方向分散方式を採用しつつ、並列 G-S 再直交化の直前でデータの再分散を行い行方向分散に変更し、かつこの再分散された行方向分散データを重複して所有することで並列 G-S 再直交化の並列性をも抽出する並列再直交化方式を提案する。

この論文の構成は以下の通りである。第 2 章で、逐次 G-S 再直交化アルゴリズムをデータ依存の観点から説明する。第 3 章では、逐次 G-S 再直交化アルゴリズムに基づく、並列 G-S 再直交化アルゴリズムについて述べる。第 4 章では、データ再分散を行う新しい並列再直交化方式の提案と、実際のアプリケーションの一例である固有ベクトル計算のための IIM に適用した場合の性能解析を行う。第 5 章は、PC クラスタおよび 3 種の国産スーパーコンピュータを用いて性能評価を行う。最後に第 6 章で、この論文で得られた知見をまとめる。

2. G-S 再直交化法の逐次アルゴリズム

G-S 法を用いた再直交化を実行するために、以下の 2 種の方法が広く利用されている。それらは、古典 G-S 法と修正 G-S 法である。古典 G-S 法は G-S 直交化の式を単純に実装した方法であり、修正 G-S 法は高い精度を得るために計算式を修正した方法である。この章では、データ依存の観点からこれらの方法の並列性の違いを説明する。なおこの並列性の違いは、各方法におけるベクトル単位でのデータ依存の違いから生じる。

2.1 古典 Gram-Schmidt (古典 G-S) 法

再直交化をするための古典 G-S 法を図 1 に示す。ここで a, b をベクトルとすると、図 1 中の表記 (a, b) は内積を示している。

図 1 では、最内側の計算 (2)-(4) は並列性を有する。この理由は、内積 (q_j, a_i) が初期ベクトル a_i を得た地点で並列に実行できるからである。

```

(1)  $q_i = a_i$ ;
(2) do  $j = 1, i - 1$ 
(3)    $q_i = q_i - (q_j, a_i)q_j$ ;
(4) enddo
(5)  $q_i$  の正規化;

```

図 1 ベクトル q_i に対する再直交化における古典 G-S 法

2.2 修正 Gram-Schmidt (修正 G-S) 法

再直交化をするための修正 G-S 法を図 2 に示す。

```

(1)  $a_i^{(0)} = a_i$ ;
(2) do  $j = 1, i - 1$ 
(3)    $a_i^{(j)} = a_i^{(j-1)} - (q_j, a_i^{(j-1)})q_j$ ;
(4) enddo
(5)  $q_i = a_i^{(i-1)}$  の正規化;

```

図 2 ベクトル q_i に対する再直交化における修正 G-S 法

図 2 では、最内側の計算 (2)-(4) の内積で並列性がない。この理由は、(3) で定義されるベクトル $a_i^{(j)}$ の計算に関して、一つ前のループで定義されたベクトル $a_i^{(j-1)}$ を参照しているからである。すなわち計算 (3) では、ループ伝搬のフロー依存が存在する。フロー依存が存在するため、そのままでは並列化ができない。この理由から、多くの研究者が修正 G-S 法による再直交化は並列処理に向かないと主張している。

3. G-S 再直交化法の並列化

並列処理をする場合 G-S 法による再直交化は、逐次処理のデータ依存とは別の理由で、参照されるベクトル q_1, q_2, \dots, q_{i-1} のデータ分散方式により、異なる並列性を示すことが知られている⁷⁾。この章では、この並列性について説明する。

3.1 列方向分散

まず、列方向分散 (Column-Wise Distribution, CWD) と呼ばれる単純な分割方式について説明する。CWD は正規化されたベクトル a_i と参照されるベクトル q_1, q_2, \dots, q_{i-1} に関して、その構成要素を分散せずにベクトル全体を PE (Processing Element) に割り当てる分散方式である。以降、CWD に基づく並列アルゴリズムの実装の詳細について議論する。

3.1.1 古典 G-S 法

図 3 は CWD における古典 G-S 法による再直交化方式を示している。図 3 中の “Local” という表記は、以降に示す式がローカルデータ (分散されたデータ) のみを用いて実行されることを表している。図 3 では、計算 (8) (9) において並列性がある。

3.1.2 修正 G-S 法

図 4 は修正 G-S 法による CWD における再直交化

復法が I.Dhillon により提案されている⁹⁾。ところがこの方法は、固有値の重複度に左右される。固有値が重複している場合、本質的に逆反復法中の再直交化は避けられない。

```

(1) if ( $a_i$  を所有) then
(2)    $q'_i = a_i$  ;
(3)   放送 ( $a_i$ ) ;
(4)   else
(5)     受信 ( $a_i$ ) ;
(6)      $q'_i = 0$  ;
(7)   endif
(8)   do  $j = 1, i - 1$ 
(9)     Local  $q'_j = q'_i - (q_j, a_i) q_j$  ; enddo
(10)  if ( $a_i$  を所有) then
(11)   do  $j = 1, i - 1$ 
(12)      $q_j$  を所有する PE からの受信 ( $q'_j$ ) ;
(13)      $q'_i = q'_i + q'_j$  ; enddo
(14)   else
(15)      $q_i$  を所有する PE へ送信 ( $q'_i$ ) ;
(16)   endif
(17)  if ( $q_i$  を所有)  $q_i = q'_i$  の正規化 ;

```

図 3 列方向分散における古典 G-S 法

方式を示している．図 4 中の計算 (4) においては，並

```

(1)  $a_i^{(0)} = a_i$  ;
(2) do  $j = 1, i - 1$ 
(3)    $a_i^{(j-1)}$  を所有する PE からの受信 ( $a_i^{(j-1)}$ ) ;
(4)   Local  $a_i^{(j)} = a_i^{(j-1)} - (q_j, a_i^{(j-1)}) q_j$  ;
(5)    $a_i^{(j+1)}$  を所有する PE へ送信 ( $a_i^{(j)}$ ) ;
(6)   enddo
(7)    $q_i = a_i^{(i-1)}$  の正規化 ;

```

図 4 列方向分散における修正 G-S 法

列性が皆無である点に注意する．これは CWD では本質的に $a_i^{(j-1)}$ と q_{j-1} を所有する PE と， $a_i^{(j)}$ と q_j を所有する PE が違うことによる．このことから，CWD での修正 G-S 法による再直交化は基本的に並列性がない．

3.2 行方向分散

次にもう一つよく使われる分散方式である，行方向分散 (Row-Wise Distribution, RWD) について説明する．RWD は，ベクトル a_i と q_i ，および直交化済ベクトル q_1, q_2, \dots, q_{i-1} に関して，その構成要素を分割して分散する方式である．したがって全ての PE は，これらベクトルの一部を所有している．CWD と RWD の違いは，CWD ではベクトルの構成要素を分割しないのに対して，RWD では構成要素を分割して分散させる点であることに注意する．

RWD では内積 $(q_j, a_i^{(j-1)})$ と (q_j, a_i) に関して，古典 G-S 法でも修正 G-S 法でもリダクション演算が必要になる．なぜなら，各 PE はこれら内積を実行するためのベクトル全てを所有していないからである．

3.2.1 古典 G-S 法

図 5 は古典 G-S 法の再直交化アルゴリズムである．図 5 中の表記の “Global sum” は，リダクション演算であり，分散されたデータを加算してから，その結果を PE 全てに放送する．このリダクション演算は MPI (Message Passing Interface) の MPI_ALLREDUCE 関数を利用して実装できる．図 5 では，最内側ループで

```

(1)  $q_i = a_i$  ;
(2) do  $j = 1, i - 1$ 
(3)   Local ( $q_j, a_i$ ) ; enddo
(4)   Global sum  $\eta_j = (q_j, a_i)$  ;
      ( $j = 1, \dots, i - 1$ )
(5)   do  $j = 1, i - 1$ 
(6)     Local  $q_i = q_i - \eta_j q_j$  ;
(7)   enddo
(8)    $q_i$  の正規化 ;

```

図 5 行方向分散における古典 G-S 法

Global sum 演算が必要でないことがわかる．この理由は，古典 G-S 法では最内側ループで毎回定義される内積値を必要としないことによる．またこのことは，2 章で説明された事項からも導かれる．すなわち j -ループの方向の依存関係が存在しないからである．この内積値は，正規化前に定義された値を参照することで容易に計算が可能となる．

なお Global sum 演算のベクトル長は，直交化するために参照するベクトルの個数 $i - 1$ に依存する．

3.2.2 修正 G-S 法

図 6 は，修正 G-S 法による並列再直交化アルゴリズムである．図 6 では，PE に分散されたデータに関

```

(1)  $a_i^{(0)} = a_i$  ;
(2) do  $j = 1, i - 1$ 
(3)   Local ( $q_j, a_i^{(j-1)}$ ) ;
(4)   Global sum  $\eta = (q_j, a_i^{(j-1)})$  ;
(5)   Local  $a_i^{(j)} = a_i^{(j-1)} - \eta q_j$  ;
(6)   enddo
(7)    $q_i = a_i^{(i-1)}$  の正規化 ;

```

図 6 行方向分散における修正 G-S 法

するリダクション演算が本質的に反復ごとに必要となる．なぜなら，Global sum 演算が最内側ループにあるからである．第 j -反復において修正 G-S 法は，第 $j - 1$ -反復での内積値を必要とするので，Global sum 演算を最内側ループに書くしかない．このことは 2 章での説明において， j -ループの方向でのフロー依存の存在から説明できる．

4. データ再分散を行う並列再直交化方式

4.1 前提となる所有情報

提案する並列再直交化方式の適用例として、固有値・固有ベクトル計算のための IIM に実装することを考える。この並列 IIM がよびだされる前に、二分法などの方法を用いて全固有値の上限と下限が計算されており、全 PE がその情報を所有していると仮定する。たとえば、第 i 番目の固有値の上限と下限は、それぞれ、 $\lambda_i(i)$ と $\lambda_u(i)$ に格納されていると仮定する。またこのデータの所有により、全 PE が計算中の第 k 固有ベクトルに関連する重複固有値の範囲について、独立に算出できると仮定する。

4.2 前提となるデータ分散方式

いま $nprocs$ 台の PE を利用し、その PE 番号を $myid = 0, 1, \dots, nprocs - 1$ とする。このとき、以下に示すデータを所有する PE 番号 P_i は、

- 実対称三重対角行列 T : 全 PE が全データを所有、
 - 固有値 λ_i ($i = 1, \dots, n$):
 $P_i \equiv [(i-1)/\lfloor n/nprocs \rfloor]$ 番 PE が所有、
 - 固有ベクトル x_i ($i = 1, \dots, n$):
 $P_i \equiv [(i-1)/\lfloor n/nprocs \rfloor]$ 番 PE が所有、
- とする。ここで、 $P_i > nprocs - 1$ の場合は、 $P_i \equiv nprocs - 1$ とする。

計算中の第 k 番固有ベクトル、および G-S 再直交化で参照されるデータの分散方式は行方向分散方式 (RWD) であるが、IIM 中で利用する固有ベクトルデータは列方向分散 (CWD) としても重複して所有していると仮定する。作業領域として列方向分散のデータを所持しているので、IIM 中で行われる LU 分解、および前進・後退代入での通信が発生しない。

以下に、再直交化すべき固有ベクトルに関するデータ所有情報をまとめる。なお、 is は現在計算中の固有ベクトル x_k と再直交化すべき固有ベクトル番号のうち、もっとも小さいものの番号をさす。また ie は、現在計算中の固有ベクトル x_k と再直交化すべき固有ベクトル番号のうち、もっとも大きいものの番号をさす。この計算すべき固有ベクトルに関する範囲情報の is と ie は、二分法などでの固有値計算結果から IIM ルーチンが実行される前に静的に与えられる。

- 現在計算中の固有ベクトル x_k :
 - CWD として $P_k \equiv [(k-1)/\lfloor n/nprocs \rfloor]$ 番 PE が x_k を全部所有、かつ、
 - $P_j \equiv [(j-1)/\lfloor n/nprocs \rfloor]$ ($j = is, \dots, ie$) 番 PE が x_k の要素を RWD された形で所有、

ただしこの LU 分解と前進・後退代入は逐次処理である。たとえ行方向分散方式 (RWD) を利用して並列化を試みても、IIM 中で利用される三重対角行列用 LU 分解で逐次化される。また前進代入処理において、枢軸選択結果に起因するベクトル収集のための通信が必要、という問題も生ずる。なお直交化の不要なベクトルについては、全く独立に並列計算可能である。

- x_k と再直交化すべき固有ベクトル x_i ($i = is, \dots, k-1$):
 - CWD として $P_i \equiv [(i-1)/\lfloor n/nprocs \rfloor]$ 番 PE が x_i を全部所有、かつ、
 - $P_j \equiv [(j-1)/\lfloor n/nprocs \rfloor]$ ($j = is, \dots, ie$) 番 PE が x_k の要素を RWD された形で所有、
- ここで上記の PE 番号計算において、 $P_k, P_j, P_i > nprocs - 1$ となる場合は、 $P_k, P_j, P_i \equiv nprocs - 1$ とする。

RWD の分散方式には任意性がある。いま対象の直交化すべきベクトル長を n とし、再直交化済の固有ベクトルを所有する PE 数を $nprocs_{RO}$ とする。このとき $\lfloor n/nprocs_{RO} \rfloor$ 個の連続する再直交化すべきベクトルの構成要素について、再直交化済の固有ベクトルを所有する各 PE が所有する「ブロック分散方式」を仮定する。ここで割り切れず直交化すべきベクトルの構成要素が余る場合、余ったベクトルの構成要素は再直交化済の固有ベクトルを所有する PE のうち、最も大きな番号の PE が所有するものとする。

固有ベクトルに関する出力のデータ分散方式は、列方向分散方式 (CWD) を仮定する。

4.3 提案方式を利用した並列逆復法

図 7 に提案する再直交化方式の概略を示す。図 7 の

- 〈1〉 CWD として格納されているベクトル x_k を、RWD として再分散；
- 〈2〉 RWD として x_k を格納；
- 〈3〉 RWD として格納されている x_k を、すでに RWD として格納されている再直交化すべきベクトル x_i ($i = is, \dots, k-1$) を用いて並列再直交化；
- 〈4〉 再直交化された RWD として格納されている x_k を、CWD として再分散；
- 〈5〉 CWD として x_k を格納；

図 7 提案する並列直交化方式の概略

提案方式は、CWD として分散されているベクトルデータを RWD に再分散するフェーズ (行 (1))、RWD されているベクトルデータを用いて再直交化するフェーズ (行 (3))、および RWD されている再直交化済のベクトルデータを CWD に再分散するフェーズ (行 (4)) から構成される。図 8 に、図 7 で示したデータ再分散を行う並列再直交化方式を利用する並列 IIM のアルゴリズムをのせる。

図 9、図 10、および図 11 に、それぞれ、並列再直交化ルーチン (1)、並列再直交化ルーチン (2)、および並列再直交化ルーチン (3) の概略をのせる。また図中の“行方向分散による並列 G-S 再直交化”には、図 5 の古典 G-S 法か図 6 の修正 G-S 法のいずれかのアルゴリズムが適用される。また図中の枠は、図 7 の提案する再直交化方式が使われていることを意味する。

```

(1)  $i_{start} = np * myid + 1$ ;  $i_{end} = np * (myid + 1)$ ;
(2) if ( $nprocs - 1$  番 PE)  $i_{end} = m$ ;
(3) do  $i = i_{start}, i_{end}$ 
(4)  $\hat{\lambda}_i = (\lambda_l(i) + \lambda_u(i)) / 2$ ;
(5) 乱数で初期ベクトル  $\hat{x}_i$  を生成;
(6)  $T - \hat{\lambda}_i I$  の三重対角行列を LU 分解;
(7) do  $iter = 1, MAX\_ITER$ 
(8) 現在計算中の固有ベクトル番号情報  $k$  を
 $\hat{\lambda}_i$  と重複する固有値を所有する PE から
検索して情報を共有させる;
(9) if ( ( $\hat{\lambda}_i$  は重複固有値).and.( $k < i$ )
.and.( $\hat{\lambda}_i$  は PE 間に重複している) ) then
(10) 並列再直化ルーチン (1);
(11) endif
(12) 前進・後退代入から  $\hat{x}_i$  を求解;
(13)  $\hat{x}_i$  の正規化;
(14) if ( $\hat{\lambda}_i$  は重複固有値) then
(15) 並列再直化ルーチン (2);
(16) endif
(17)  $\hat{x}_i$  の正規化;
(18) Rayleigh 商による  $\hat{\lambda}_i$  の改良;
(19) 残差  $res = \|T\hat{x}_i - \hat{\lambda}_i\hat{x}_i\|$  の計算;
(20) if ( $res < \hat{\epsilon}$ )  $iter$  ループの終了;
(21) enddo
(22) if ( $\hat{\lambda}_i$  は重複固有値) 収束ベクトル  $\hat{x}_i$  の放送;
(23) enddo
(24) if ( ( $\lambda_{np * (\hat{myid} + 1)}$  は重複固有値).and.
( $\lambda_{np * (\hat{myid} + 1)}$  は PE 間に重複している) )
then
(25) 並列再直化ルーチン (3);
(26) endif

```

図 8 データ再分散を行う並列再直化方式を利用した IIM . なお $np \equiv \lfloor m/nprocs \rfloor$ とおいた .

4.4 性能解析

ここでは CWD を用いた従来法と、データ再分散を行う提案法について実行速度の解析を行う . 以下議論を簡単にするために、再直化すべき固有ベクトルの数を m 、固有ベクトルの長さを n とし、同じ個数各 PE に分散されていると仮定する . このとき直化のための修正 G-S 法の演算時間を、列方向分散方式 (従来法) では $C_{MGS_{CB}}$ 、および行方向分散方式 (提案法) では $C_{MGS_{RB}}$ と表記する . また古典 G-S 法の演算時間も同様に、 $C_{CGS_{CB}}$ 、 $C_{CGS_{RB}}$ と表記する .

通信に関する時間、すなわち、長さ n のデータをリダクション演算する時間、転送する時間、放送する時間、および収集する時間を、それぞれ、 $T_{red}(n, p)$ 、 $T_{send}(n)$ 、 $T_{broad}(n, p)$ 、および $T_{gather}(n, p)$ とする . ここで p はプロセッサ台数である .

```

(1) if ( $\hat{\lambda}_i$  は PE 間で重複していない) then
(2) 逐次 修正 G-S 法;
(3) else
(4)  $\hat{x}_i$  の受信;
(5)  $x_i^{rb} = (\hat{x}_i$  を行方向分散として格納);
(6) if ( $\hat{x}_i$  は収束ベクトル) then
(7) 現在計算中の固有ベクトル番号情報  $k$  の検索;
(8) if ( $k$  .eq.  $i$ ) goto (13);
(9) endif
(10) 行方向分散による並列 G-S 再直化 ( $x_i^{rb}$ )
(11) 固有ベクトル  $k$  を所有する PE に  $x_i^{rb}$ 
を転送;
(12) goto (4);
(13) continue
(14) endif

```

図 9 並列直化ルーチン (1) の概略

```

(1)  $\hat{\lambda}_i$  の重複固有値を所有する PE へ  $\hat{x}_i$  を送信;
(2)  $x_i^{rb} = (\hat{x}_i$  を行方向分散として格納);
(3) 行方向分散による並列 G-S 再直化 ( $x_i^{rb}$ );
(4)  $\hat{\lambda}_i$  の重複固有値を所有する PE から
 $tx_i^{rb}$  を受信;
(5)  $\hat{x}_i = (tx_i^{rb}$  を列方向分散として格納);

```

図 10 並列直化ルーチン (2) の概略

```

(1) 現在計算中の固有ベクトル番号情報  $k$  の検索;
(2)  $\hat{x}_i$  の受信;
(3)  $x_i^{rb} = (\hat{x}_i$  を行方向分散として格納);
(4) 行方向分散による並列 G-S 再直化 ( $x_i^{rb}$ );
(5) if ( $\hat{x}_i$  は収束ベクトル) then
(6) if ( $k$  は重複固有値の最後か) goto (10);
(7) endif
(8) 固有ベクトル  $k$  を所有する PE に  $x_i^{rb}$  を転送;
(9) goto (1);
(10) continue

```

図 11 並列直化ルーチン (3) の概略

4.4.1 修正 G-S 法

従来法である列方向分散方式による修正 G-S 法再直化の時間を $T_{MGS_{CB}}$ 、提案手法である行方向分散方式による修正 G-S 法再直化の時間を $T_{MGS_{RB}}$ とする . ここで最悪の時間を考慮し、固有ベクトル m 本と再直化しなくてはならない場合を考える . このとき、従来法の時間 $T_{MGS_{CB}}$ は、

$$T_{MGS_{CB}} \approx p \cdot T_{send}(n) + p \cdot C_{MGS_{CB}} \quad (1)$$

となる .

一方、提案手法の時間 $T_{MGS_{RB}}$ は、

$$T_{MGS_{RB}} \approx T_{broad}(n, p) + C_{MGS_{RB}} + m \cdot T_{red}(1, p) + T_{gather}(n/p, p) \quad (2)$$

となる。このとき従来法よりも提案手法が高速になる条件 $T_{MGS_{CB}} > T_{MGS_{RB}}$ を考慮したプロセッサ台数 p_{MGS} は、

$$p_{MGS} > (TC_{broad} + m \cdot TC_{red1} + TC_{gather} + CC_{RB_CB}) / (TC_{send} + 1) \quad (3)$$

となる。ここで、

$$\begin{aligned} TC_{broad} &\equiv T_{broad}(n, p) / C_{MGS_{CB}}, \\ TC_{red1} &\equiv T_{red}(1, p) / C_{MGS_{CB}}, \\ TC_{gather} &\equiv T_{gather}(n/p, p) / C_{MGS_{CB}}, \\ CC_{RB_CB} &\equiv C_{MGS_{RB}} / C_{MGS_{CB}}, \\ TC_{send} &\equiv T_{send}(n) / C_{MGS_{CB}}, \end{aligned} \quad (4)$$

とおいた。式 (3) のしきい値となるプロセッサ台数 p_{MGS} が小さいと、提案手法がより有効となる。したがって通信時間と演算時間の比である、 TC_{broad} , TC_{red1} , TC_{gather} , および CC_{RB_CB} の値が小さい状況では提案法の効果が期待できる。また TC_{send} の値が大きい状況のほうが、提案法の効果が期待できる。

4.4.2 古典 G-S 法

修正 G-S 法と同様に、従来法である列方向分散方式による古典 G-S 法再直交化の時間を $T_{CGS_{CB}}$ 、提案手法である行方向分散方式による古典 G-S 法再直交化の時間を $T_{CGS_{RB}}$ とする。このとき、従来法の時間 $T_{CGS_{CB}}$ は、

$$T_{CGS_{CB}} \approx T_{broad}(n, p) + C_{CGS_{CB}} + T_{gather}(n, p) + C_{CGS_{ADD}}(n, p) \quad (5)$$

となる。ここで、列方向分散方式における古典 G-S 法で逐次化されるベクトル加算の時間を $C_{CGS_{ADD}}(n, p)$ とおいた。

一方、提案手法の時間 $T_{CGS_{RB}}$ は、

$$T_{CGS_{RB}} \approx T_{broad}(n, p) + T_{red}(m, p) + C_{CGS_{RB}} + T_{gather}(n/p, p) \quad (6)$$

となる。このとき、従来法よりも提案手法が高速になる条件 $T_{CGS_{CB}} > T_{CGS_{RB}}$ は

$$\frac{(TC_{gather}(n) + CC_{ADD_{CB}} + 1)}{(TC_{gather}(n/p) + TC_{redm} + CC_{RB_CB})} > 1 \quad (7)$$

となる。ここで、

$$\begin{aligned} TC_{gather}(n) &\equiv T_{gather}(n, p) / C_{CGS_{CB}}, \\ CC_{ADD_{CB}} &\equiv C_{CGS_{ADD}}(n, p) / C_{CGS_{CB}}, \\ TC_{gather}(n/p) &\equiv T_{gather}(n/p, p) / C_{CGS_{CB}}, \\ TC_{redm} &\equiv T_{red}(m, p) / C_{CGS_{CB}}, \\ CC_{RB_CB} &\equiv C_{CGS_{RB}} / C_{CGS_{CB}}, \end{aligned} \quad (8)$$

とおいた。式 (7) の条件を成立させやすくと、提案手法が従来法より有効となる。したがって、通信時間と演算時間の比である $TC_{gather}(n)$ および $CC_{ADD_{CB}}$ は大きく、 $TC_{gather}(n/p)$, TC_{redm} , および CC_{RB_CB}

は小さい値をもつ状況のほうが、提案法が有効であることが期待できる。

実際の再直交化では、再直交化の時間は各 PE で均一でなく、また再直交化すべき個数 m も逆反復が進むにつれ 1 から m まで 1 ずつ増加していく。したがって単純に式 (3) と式 (7) の条件で決まるわけではない。しかしこれらの条件は、従来法より高速となる目安になるものと考えられる。

5. 性能評価

この章では、4 種の並列計算機を用いて各種再直交化方式を評価する。再直交化方式の評価のため IIM に並列化された再直交化を実装した。

5.1 計算機環境

性能評価のため、以下の並列計算機を用いた。

- PC クラスタ: ノードのハードウェア (Intel Pentium4, 2.0GHz), ノード数 (4 ノード), ノード当たりの搭載メモリ (1GB, Direct RDRAM/ECC 256MB*4), MB (ASUSTek P4T-E+A, Socket478 対応), ネットワークカード (Intel EtherExpressPro100+), OS(Linux 2.4.9-34), 通信ライブラリ (MPICH 1.2.1), コンパイラ (PGI Fortran90 コンパイラ, 4.0-2), コンパイラオプション (-fast)。
- HITACHI SR8000/MPP: 東京大学情報基盤センタ所有。ノード当たりの PE 数 (8PE), ノード当たりの理論性能 (14.4GFLOPS), ノード当たりのメモリ (16GB), 通信ネットワーク (3 次元ハイパークロスバ), 最大通信性能 (片方向で 1.6 Gbytes/s, 双方向で 3.2Gbytes/s), コンパイラ (日立最適化 Fortran90 V01-05-/A), コンパイラオプション (-opt=4 -parallel=0), 通信ライブラリ (日立製 MPI)。
- Fujitsu VPP800/63: 京都大学学術情報メディアセンタ所有。ノードにおける理論性能 (8GFLOPS のベクトルユニット, および 1GOPS のスカラーユニット), ノード当たりのメモリ (8GB), 通信ネットワーク (クロスバ網), 最大通信性能 (3.2Gbytes/s), コンパイラ (Fujitsu UXP/V Fortran/VPP V20L20), コンパイラオプション (-O5 -X9), 通信ライブラリ (富士通製 MPI)。
- NEC SX-5/128M8: 大阪大学サイバーメディアセンタ所有。ノード当たりの PE 数 (16PE), ノード当たりの理論性能 (160GFLOPS), ノード当たりのメモリ (128GB), コンパイラ (NEC FORTRAN90/SX), コンパイラオプション (-C hopt), 通信ライブラリ (MPI/SX)。

表 1 提案する再直化方式の効果 . (実行時間, 単位は秒)

(a) PC クラスタ 4 ノード ($p = 4$)

固有ベクトル数 (%)	500(5%)	1,000(10%)	2,000(20%)	4,000(40%)	5,000(50%)	10,000(100%)
MG- S_{CB} (従来法)	88	264	879	3,165	4,832	18,630
MG- S_{RB} (提案法)	1,151	2,989	6,036	14,302	25,768	83,633
CG- S_{CB} (従来法)	64	167	488	1,607	2,428	8,946
CG- S_{RB} (提案法)	102	237	616	1,794	2,592	8,747

(b) HITACHI SR8000/MPP 1 ノード 8PE ($p = 8$)

固有ベクトル数 (%)	500(5%)	1,000(10%)	2,000(20%)	4,000(40%)	5,000(50%)	10,000(100%)
MG- S_{CB} (従来法)	203	625	2,513	8,757	17,116	65,019
MG- S_{RB} (提案法)	56	208	722	2,985	4,052	17,550
CG- S_{CB} (従来法)	40	147	536	2,027	3,037	12,042
CG- S_{RB} (提案法)	30	91	302	1,067	1,618	6,193

(c) HITACHI SR8000/MPP 4 ノード 32PE ($p = 32$)

固有ベクトル数 (%)	500(5%)	1,000(10%)	2,000(20%)	4,000(40%)	5,000(50%)	10,000(100%)
MG- S_{CB} (従来法)	155	532	2,169	7,718	12,573	50,165
MG- S_{RB} (提案法)	134	417	1,585	6,030	10,176	36,477
CG- S_{CB} (従来法)	38	109	242	698	1,053	3,587
CG- S_{RB} (提案法)	42	94	210	578	793	2,412

(d) Fujitsu VPP800/63 4 ノード ($p = 4$)

固有ベクトル数 (%)	500(5%)	1,000(10%)	2,000(20%)	4,000(40%)	5,000(50%)	10,000(100%)
MG- S_{CB} (従来法)	17	51	171	617	943	3,593
MG- S_{RB} (提案法)	32	110	407	1,556	2,408	9,439
CG- S_{CB} (従来法)	12	32	94	309	461	1,667
CG- S_{RB} (提案法)	10	25	63	182	261	860

(e) Fujitsu VPP800/63 8 ノード ($p = 8$)

固有ベクトル数 (%)	500(5%)	1,000(10%)	2,000(20%)	4,000(40%)	5,000(50%)	10,000(100%)
MG- S_{CB} (従来法)	17	52	173	621	947	3,600
MG- S_{RB} (提案法)	40	140	521	2,009	3,115	12,247
CG- S_{CB} (従来法)	11	26	68	201	292	983
CG- S_{RB} (提案法)	10	22	50	118	160	438

(f) NEC SX-5/128M8 1 ノード 4PE ($p = 4$)

固有ベクトル数 (%)	500(5%)	1,000(10%)	2,000(20%)	4,000(40%)	5,000(50%)	10,000(100%)
MG- S_{CB} (従来法)	48	165	555	2,020	3,066	10,352
MG- S_{RB} (提案法)	93	311	1,254	5,516	5,125	18,738
CG- S_{CB} (従来法)	13	36	106	354	531	1,929
CG- S_{RB} (提案法)	10	22	53	137	199	549

5.2 並列逆反復法 (IIM) の詳細

我々が開発した IIM ルーチンの詳細は以下の通りである。

- 処理行列：実数対称三重対角行列 T
- 計算対象：昇順にソートした固有値に対する固有ベクトル第 1 番から第 m 番
- アルゴリズム：Rayleigh 商改良付き IIM
- 要求される誤差：残差ベクトルのノルム $\|Tx_i - \lambda_i x_i\|_2$ を解からの誤差とみなすとき、誤差 $\|T\|_1 \times \epsilon$, ($i = 1, 2, \dots, m$) 以下を要求する。ここで ϵ はマシンイプシロンで、 $x_i \in \mathbb{R}^n$ はこの固有方程式の第 i 番固有ベクトル、 $\lambda_i \in \mathbb{R}$ は第 i 番固有値である。
- 密集固有値判定法：距離 $eps \equiv \|T\|_1 \times 10^{-3}$ とすると、 $|\lambda_i - \lambda_{i-1}| < eps$, ($i = 2, 3, \dots, n$, かつ λ_i は昇順にソート済) となる固有値全てを密集固有値とみなす (Peters-Wilkinson の方法)

なお IIM では、行列の数値特性に依存する反復回数が全体の実行時間を左右する。ここでは EISPACK の IIM を用いたルーチンである TINVIT ルーチンにおける最大反復回数¹⁰⁾ を考慮し、各固有ベクトル収束までの反復回数を 5 回に固定して評価を行う。

5.3 実装した並列再直交化の説明

我々が実装した並列再直交化を以下に示す。

- CG- S_{CB} (従来法)：CWD による PE 内再直交化として修正 G-S 法を用いた並列古典 G-S 法
- MG- S_{CB} (従来法)：CWD による逐次化される修正 G-S 法
- CG- S_{RB} (提案法)：RWD による PE 内再直交化として修正 G-S 法を用いた並列古典 G-S 法。再直交化前/後にデータ再分散が行われる。
- MG- S_{RB} (提案法)：RWD による並列修正 G-S 法。再直交化前/後にデータ再分散が行われる。

5.4 試験行列

試験行列は、10,000 次元の Frank 行列を三重対角化した行列とする。行列特性は以下の通りである。

- 密集固有値とみなす距離： $eps \approx 58,471$
- 密集固有値のグループ数：8
- 最大密集固有値群：昇順にソート済の場合、第 1 番から第 9,993 番固有値

なお Frank 行列は、次のように定義される：

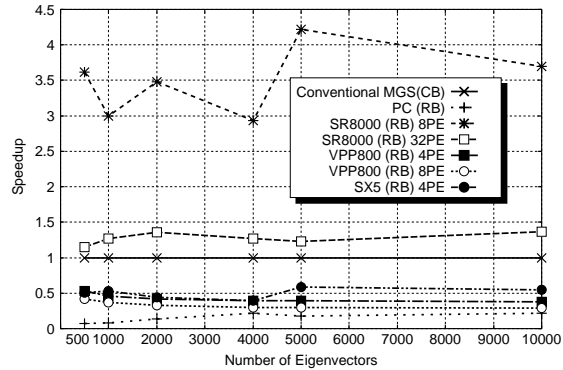
$$A = (a_{ij}) \equiv n - \max(i, j) + 1, (i, j = 1, 2, \dots, n)$$

5.5 実験結果

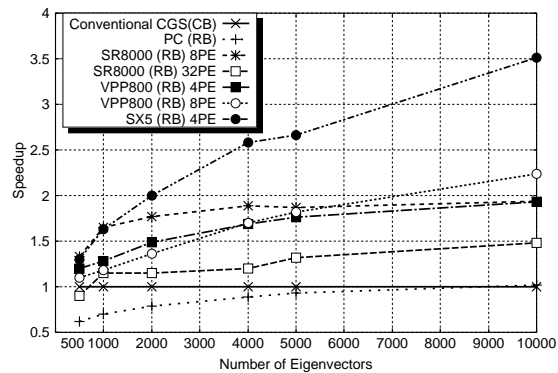
表 1 は、CWD を利用した従来方式と RWD を利用した提案方式とを、各並列計算機上で実行した結果を示している。なおこの試験行列においては、Frobenius ノルムによる直交精度の評価では、修正 G-S 法と古典 G-S 法間での精度差は生じなかった。

図 12 は、各並列計算機上での従来方式の実行時間を 1 としたとき、提案方式でどれだけ高速化されるかの割合を示したものである。表 1 および図 12 から修

正 G-S 法においては、SR8000 を利用したときのみ従来法より高速化され、その最大高速化率は約 4 倍である。また古典 G-S 法では、PC クラスタ以外で速度向上がみられ、その最大高速化率は SX5 の約 3.5 倍である。



(a) 修正 G-S 法における従来法に対する速度向上比



(b) 古典 G-S 法における従来法に対する速度向上比

図 12 各並列計算機上での従来法に対する速度向上比

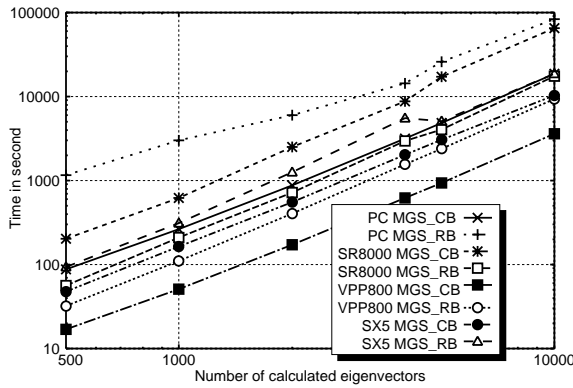
図 13 は、PC クラスタ、SR8000 ($PE = 8$), VPP800 ($PE = 4$), および SX5 での、修正 G-S 法と古典 G-S 法の実行時間を示している。図 13 から、修正 G-S 法においては

- VPP800(従来法) < VPP800(提案法) < SX5(従来法) < SR8000(提案法) < PC クラスタ (従来法) < SX5(提案法) < SR8000(従来法) < PC クラスタ (提案法)

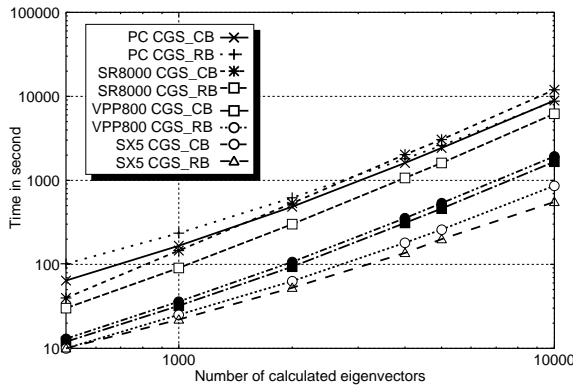
の順番で高速となる。一方、古典 G-S 法においては高速となる順序が、計算すべき固有ベクトル数に応じて変化している。計算すべき固有ベクトル数が 2,000 未満のとき、

- SX5(提案法) < VPP800(提案法) < VPP800(従来法) < SX5(従来法) < SR8000(提案法) < SR8000(従来法) < PC クラスタ (従来法) < PC クラスタ (提案法)

の順番で高速となる。計算すべき固有ベクトル数が



(a) 修正 G-S 法



(b) 古典 G-S 法

図 13 各並列計算機上での実行時間

2,000 以上のとき,

- SX5(提案法) < VPP800(提案法) < VPP800(従来法) < SX5(従来法) < SR8000(提案法) < PC クラスタ (提案法) < PC クラスタ (従来法) < SR8000(従来法)

の順序で高速となる.

5.6 考察

表 2 に式 (3) および式 (7) で示した, 演算時間に対する比のパラメータ値をのせる. なお SR8000 では 8PE 利用時, VPP800 では 4PE 利用時のパラメータをのせている. このパラメータ値は, C_{CGS_ADD} を除く計算時間パラメータは 10 回, C_{CGS_ADD} と全ての通信時間パラメータは 10,000 回の時間を計測し平均を算出後, この平均値を利用し算出した.

表 2 から修正 G-S に関して, SR8000 のみで従来法に対して高速化された理由を考える. その理由として, 以下が推察される.

- 固有ベクトル計算 500 個のとき, 1 対 1 通信時間と計算時間との比 TC_{send} がその他の計算機と比べて大きい. したがって, 従来法の通信時間の遅さが提案法を有利にさせた.
- 固有ベクトル計算 10,000 個のとき, リダクシ

表 2 通信時間と演算時間との比に関するパラメータ. 各パラメータの単位はミリ秒. (問題サイズ $n = 10,000$)

(a) 固有ベクトル 500 個計算時

機種	PC	SR8000	VPP800	SX5
TC_{broad}	.00747	.0795	.00235	.00067
TC_{red1}	.00074	.00422	.00063	.00053
TC_{gather}	.00123	.00983	.00062	.00038
CC_{RB_CB}	1.02	.968	.213	.595
TC_{send}	.00285	.0141	.00129	.00047
$PMGS >$	1.40	3.12	.533	.864
TC_{redm}	.00326	.00960	.0226	.1111
$TC_{gather}(n/p)$.00461	.00960	.0166	.00708
CC_{RB_CB}	3.85	1.001	.732	.851
$TC_{gather}(n)$.0305	.0449	.0580	.0293
CC_{ADD_CB}	.0268	.0578	.0181	.0196
式 (7) 値	.274	1.08	1.39	1.08

(b) 固有ベクトル 10,000 個計算時

機種	PC	SR8000	VPP800	SX5
TC_{broad}	.00036	.00292	.00291	.00386
TC_{red1}	.00003	.00015	.00078	.00196
TC_{gather}	.00005	.00049	.00078	.00127
CC_{RB_CB}	1.001	.942	.330	.431
TC_{send}	.00013	.00090	.00160	.00158
$PMGS >$	1.32	2.51	1.11	2.39
TC_{redm}	.00229	.00508	.01446	.06921
$TC_{gather}(n/p)$.00023	.00047	.00839	.00363
CC_{RB_CB}	3.91	.970	.728	.816
$TC_{gather}(n)$.00158	.00213	.02907	.00363
CC_{ADD_CB}	.00136	.00279	.00904	.0150
式 (7) 値	.255	1.02	1.38	1.14

ン時間と計算時間の比 TC_{red1} がその他の計算機と比べて小さく, かつ計算時間比 CC_{RB_CB} が小さい. したがって大規模な問題では, 問題サイズに比例して増加するリダクション時間と, 従来法に対する計算時間を小さく抑えることができたので, 結果として提案法が有利となった.

古典 G-S 法では, PC クラスタ以外の計算機で速度向上が確認され, 最大の速度向上が得られたのは SX5 である. その理由として, 以下が推察される.

- PC クラスタでは, 計算時間比 CC_{RB_CB} がその他の計算機に比べ大きい. したがって, 提案法の計算時間の遅さが提案法を不利にした.
- SX5 では, ベクトル収集時間と計算時間との比 $TC_{gather}(n/p)$ がその他の計算機と比べて小さい. したがって, 提案法におけるベクトル収集時間の高速さが提案法をより有利にした.

一方, 実際の実行時間で比較した場合 (図 13) を考察する. 図 13 のデータにおける各並列計算機の理論性能は, PC クラスタは 8GFLOPS, SR8000 は 14.4GFLOPS(PC クラスタ比 1.8 倍), VPP800 は 32GFLOPS(PC クラスタ比 4 倍), および SX5 は 40GFLOPS(PC クラスタ比 5 倍) である.

修正 G-S 法では, 以下が推察される.

- SX5 は VPP800 よりも理論性能が高いにもかかわらず

かわならず、VPP800 よりも約 3 倍遅い。したがって、理論性能比に対する効果を得られない。この理由は、SX5 においては、VPP800 よりも従来法との相対的な提案法における計算カーネル速度が遅いことに起因すると推察される (表 2 をみよ)。

- SR8000 の従来法は固有ベクトル数が 2000 以上の場合、PC クラスタの提案法と従来法よりも遅い。これは、計算と通信に関する理論性能比相当の効果を得られないことを意味する。この理由は、計算時間に対する 1 対 1 通信の性能比 TC_{send} が PC クラスタより悪いことに起因していると推察される。
- PC クラスタと VPP800 との理論性能比は 4 倍であるが、双方の従来法における実行時間は約 5 倍以上 VPP800 のほうが高速である。これは、VPP800 の通信性能のよさに起因するものと推察される。したがって、スーパーコンピュータの利用は効果を奏することがある。

古典 G-S 法について、PC クラスタにおける従来法と提案法のうち速いほうの時間を 1 としたとき、その他の計算機における提案法がどれだけ高速化されるかの比を図 14 に示す。

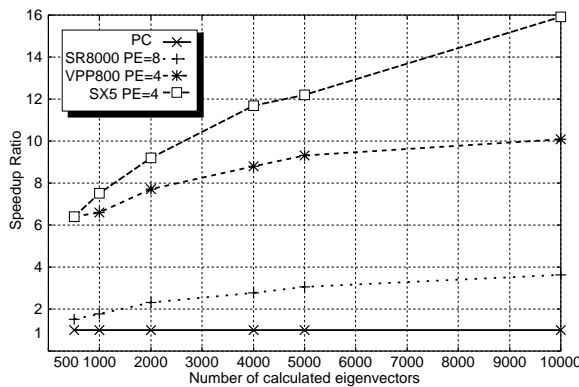


図 14 PC クラスタ上での最速実行時間に対する速度向上比 (提案法)

古典 G-S については、以下が推察される。

- 計算すべき固有ベクトル数が多くなると、SR8000 の従来法は PC クラスタの実行時間より遅くなる。したがって、スーパーコンピュータを利用する効果を奏しない。この理由は PC クラスタに対して、計算時間に対するベクトル収集時間 $TC_{gather}(n/p)$ が悪いことが影響したものと推察される。
- PC クラスタのもっとも高速となる実行時間に対して、提案法では SR8000 で約 4 倍 (PC クラスタとの理論性能比 1.8 倍に対し 2.2 ポイント)、VPP800 で約 10 倍 (PC クラスタとの理論性能比 4 倍に対し 2.5 ポイント)、および SX5 では約 16

倍 (PC クラスタとの理論性能比 5 倍に対し 3.2 ポイント) の高い理論性能比に対する効果を得ている (図 14 をみよ)。すなわちスーパーコンピュータでは、理論性能比以上の効果を奏している。この理由は、提案法の計算カーネルが従来法よりも高速である (表 2 での CC_{RB_CB} 値が小さい) ことから、ベクトル計算機での演算効率の高さ、および PC クラスタに対してスーパーコンピュータでは通信性能がよいこと、などが影響したと推察される。

これらの考察より、提案方式はベクトル並列型のスーパーコンピュータ上では、スカラ並列型の並列計算機に対して理論性能比以上の速度向上が達成できる方式であるといえる。したがってスーパーコンピュータ上では、提案法のより効率的な実行が期待できる。

6. おわりに

本論文では、従来より問題となっていた列方向分散方式 (CWD) による並列 Gram-Schmidt 再直交化の低い並列性を、並列性がより高い行方向分散方式 (RWD) による並列アルゴリズムを利用するようにデータを再分散することで並列性を改良する方式を提案した。性能評価の結果から、修正 G-S 法で最大 4 倍、古典 G-S 法で最大 3.5 倍の高速化を達成した。

提案手法が有効となる状況は、放送時間やベクトル収集時間と計算時間との比である TC_{broad} 、 TC_{gather} が小さく、かつ 1 対 1 通信時間と計算時間との比 TC_{send} が大きくなる並列計算環境である。 TC_{broad} 、 TC_{gather} が小さいという条件から、大規模な PE を有する並列計算機では不利となる。また TC_{send} が大きいという条件から、大規模な問題サイズでの実行は不利となる。したがって性能評価で示したような、4-32 台という比較的小規模な PE 構成で、かつスーパーコンピュータ上で 10,000 次元程度の問題を解く場合に有効となる方式であるといえる。

本提案方式の本質はホットスポットにおいて、その処理に向くデータ分散方式に再分散することにある。本論文で示した固有ベクトル計算のための逆反復法のように、ホットスポットに向くデータ分散方式と、それ以外の処理に向くデータ分散方式が異なるという処理は多いものと推察される。ここで単純なデータ分散方式の変更だけであるから、アルゴリズムの本質と関係ないと解釈されるべきではない。逐次アルゴリズムとデータ構造が切り離せないのと同様に、並列処理ではデータ構造としてのデータ分散方式は切り離せない。本論文の性能評価により、この実例を示した。

今後の課題として数値計算ライブラリにおいて、性能オプションとして本方式を利用するための手法を開発することがあげられる。たとえば、実行対象の並列計算機の特性をモデル化することで、CWD と RWD

でどちらのデータ分散方式が最適となるか固有値求解ライブラリを実行しなくても決定できる手法の開発が必要である。

謝 辞

本研究は、科学技術振興機構 戦略的創造研究推進事業 さきがけプログラム、および平成 14 年度大川情報通信基金研究助成 (02-12) による助成を受けた。

参 考 文 献

- 1) Demmel, J. W.: *Applied Numerical Linear Algebra*, SIAM (1997).
- 2) Dongarra, J. J. and van de Geijn, R. A.: Reduction to Condensed Form for the Eigenvalue Problem on Distributed Memory Architectures, *Parallel Computing*, Vol. 18, pp. 973–982 (1992).
- 3) Hendrickson, B. A. and Womble, D. E.: The Tours-Wrap Mapping for Dense Matrix Calculation on Massively Parallel Computers, *SIAM Sci. Comput.*, Vol. 15, No. 5, pp. 1201–1226 (1994).
- 4) 山本有作, 猪貝光祥, 直野健: 共有メモリ型並列計算機向けの高並列固有ベクトル解法, 2000 年記念並列処理シンポジウム JSPP2000 論文集, pp. 19–26 (2000).
- 5) Oliveira, S., Borges, L., Holzrichter, M. and Soma, T.: Analysis of Different Partitioning Schemes for Parallel Gram-Schmidt Algorithms, *Parallel Algorithm and Applications*, Vol. 14, No. 4, pp. 293–320 (2000).
- 6) Vanderstraeten, D.: A Parallel Block Gram-Schmidt Algorithm with Controlled Loss of Orthogonality, *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing* (1999).
- 7) 片桐孝洋: スーパーコンピュータ環境における Gram-Schmidt 再直交化の性能評価, ハイパフォーマンスコンピューティングと計算科学シンポジウム HPCS2003 論文集, pp. 75–82 (2003).
- 8) 片桐孝洋, 金田康正: 並列固有値ソルバーの実現とその性能, 情報処理学会研究報告, 97-HPC-69, pp. 49–54 (1997).
- 9) Dhillon, I.: A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue / Eigenvector Problem, *Ph.D Thesis, Computer Science Division, University of California Berkeley* (1997).
- 10) Jessup, E. R. and Ipsen, I. C. F.: Improving the Accuracy of Inverse Iteration, *SIAM J. Sci. Stat. Comput.*, Vol. 13, No. 2, pp. 550–572 (1992).