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

片 桐 孝 洋 †,††

本論文では，性能評価が十分になされてこなかった Gram-Schmidt (G-S) 法による再直交化処理の並列アルゴリズムについて議論する．データ分散や実装方式を検討して，5 種の並列再直交化を実装した．分散メモリ型スーパーコンピュータ HITACHI SR8000/MPP，Fujitsu VPP800/63 による性能評価の結果，従来から利用されている修正 G-S 法では全く速度向上が得られなかった問題に対して，古典 G-S 法を用いると問題の性質や計算機の種類に依存せず約 4 倍の速度向上がさらに得られることが判明した．

Performance Evaluation of Gram-Schmidt Re-Orthogonalization Methods on Super-computer Environments

Takahiro KATAGIRI †,††

Parallel re-orthogonalization with Gram-Schmidt (G-S) method, which performance has not been enough evaluated, is treated in this paper. With taking account of data distribution and implementation methods, we develop the five kinds of parallel re-orthogonalization algorithms. The performance evaluation by using the HITACHI SR8000/MPP and Fujitsu VPP800/63 which are distributed memory super-computers, indicated that the conventional approach of modified G-S method had no speed-up factor. We obtained, however, 4 times speed-up factors in several machine environments and problems by using classical G-S based methods.

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きた[2)~5)]．それにもかかわらず，我々はこの論文で再直交化処理という直交化処理に焦点を当てる．その理由は，この再直交化処理は多くの連立 1 次方程式や固有値計算のための反復解法において，解の精度を保つために必要な処理であることによる．例えば，近年研究されてきた反復解法の 1 つである GMRES 法は，リスタート周期の大きさに依存する基底ベクトルを得るため，再直交処理を必要とする[6]．

近年，並列処理技術の進展により多くの並列計算機が利用できるようになってきている．この理由から，多くの研究者が直交化処理の並列アルゴリズムの研究をおこなっている．ところが本論文の焦点である再直交化処理の並列化アルゴリズムは，従来の QR 分解を基にしたアルゴリズム[4]とは完全に異なる並列性を示す．その理由は，直交化アルゴリズムに内在する並列性が，得られる直交化ベクトル数に大きく依存するからである[7]．それゆえこの論文では，直交化処理を以下の 2 種の処理に分類し区別して参照する：

- QR 分解
  正規化され，直交化されているベクトル q_1, q_2, $\cdots$, q_n を，正規化されているベクトル a_1, a_2, $\cdots$, a_n から求める直交化処理．
- 再直交化
  正規化され，直交化されているベクトル q_i を，既に正規化され直交化されているベクトル q_1, q_2, $\cdots$, q_{i-1} から求める直交化処理．

この論文では，Gram-Schmidt (G-S) 直交化法による再直交化処理の並列化について議論する．焦点は古典 G-S(Classical G-S，CG-S)法の並列化にある．なぜならば古典 G-S 法は，従来から利用されている修正 G-S (Modified G-S，MG-S)法よりも高い並列性を有することによる．ところが残念なことに古典 G-S 法を並列化する場合，精度と実行時間におけるトレードオフの問題が存在する．しかし精度保証という観点では，直交化すべきデータの数値特性を評価しつつ古典 G-S 法と修正 G-S 法を動的に切替える方法[8]が提案されている．ゆえに，古典 G-S 法の並列性能が並列再直交化アルゴリズム全体の性能を決定する．

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
†† 科学技術振興事業団 さきがけプログラム
“Information Infrastructure and Applications”, PRESTO, JST

以上のことを考慮し，本論文での目的は 2 つある．まず第一の目的は，上記で述べたトレードオフ問題を緩和するであろう，新しい並列 G-S アルゴリズムを提案し評価することである．次に第二の目的は，並列再直交化アルゴリズムの性能上限を決定するが，実際の並列計算機を用いて性能評価された報告があまりない，古典 G-S 法の並列アルゴリズムの性能評価を目的にする．

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，新しいアルゴリズムを提案する．第 5 章は，その新しいアルゴリズムを含む，並列化された 5 種の再直交化アルゴリズムの性能評価である．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 (CG-S)法と修正 G-S(MG-S)法である．CG-S 法は G-S 直交化の式を単純に実装した方法であり，MG-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．

2.1 古典 Gram-Schmidt (CG-S) 法

再直交化をするための CG-S 法を図 **1** に示す．ここで図 1 中の表記 $(\cdot,\cdot)$ は，内積を示している．

```
⟨1⟩  q_i = a_i
⟨2⟩  do j = 1, i − 1
⟨3⟩    q_i = q_i − (q_j^T, 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 CG-S 法

図 1 では，最内の計算 ⟨3⟩ は並列性を有する．この理由は，内積 (q_j^T, 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．

2.2 修正 Gram-Schmidt (MG-S) 法

再直交化をするための MG-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^T, 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 MG-S 法

図 2 では，最内の計算 ⟨3⟩ 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 MG-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 $q_1, q_2, ..., q_{i-1}$ 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 $q_1, q_2, ..., q_{i-1}$ に関して，その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．

以降は，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 CG-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．

```
⟨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_i' = q_i' − (q_j^T, 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⟩ q_i = q_i' の正規化
```

図 3　列方向分散における古典 G-S 法

図 3 では，計算 ⟨9⟩ において並列性がある．

3.1.2 修正 G-S 法

図 **4** は MG-S 法による CWD における再直交化方式を示している．

図 4 中の計算 ⟨4⟩ においては，並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

```
⟨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^T, 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

q_{j-1} を所有する PE，および $a_i^{(j)}$ 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 MG-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

次にもう一つの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 $q_1, q_2, ..., q_{i-1}$ に関して，その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，これらベクトルの要素の一部を所有している．CWD と RWD の違いは，CWD ではベクトルの要素を分割しないのに対して，RWD では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 $(q_j^T, a_i^{(j-1)})$ と (q_j^T, a_i) に関して，CG-S 法でも MG-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 CG-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．

```
⟨1⟩  q_i = a_i
⟨2⟩  do j = 1, i − 1
⟨3⟩    Local (q_j^T, a_i)   enddo
⟨4⟩  Global sum η_j = (q_j^T, a_i).
                          (j = 1, ··· , i − 1)
⟨5⟩  do j = 1, i − 1
⟨6⟩    Local q_i = q_i − η_j q_j
⟨7⟩  enddo
⟨8⟩  q_i の正規化
```

図 5　行方向分散における CG-S 法

図 5 では，最内ループで Global sum 演算が必要でないことがわかる．この理由は，CG-S 法では最内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 $i-1$ に依存する点に注意する．

3.2.2　修正 G-S 法

図 **6** は，MG-S 法による並列再直交化アルゴリズムである．

```
⟨1⟩  a_i^(0) = a_i
⟨2⟩  do j = 1, i − 1
⟨3⟩    Local (q_j^T, a_i^(j−1))
⟨4⟩    Global sum η = (q_j^T, a_i^(j−1)).
⟨5⟩    Local a_i^(j) = a_i^(j−1) − η 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 MG-S 法

図 6 では，PE に分散されたデータに関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 演算が最内ループにあるからである．第 j-反復において MG-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．

3.2.3　通信時間の比較と精度

ここでは，Global sum 演算の回数を MG-S 法と CG-S 法で比較してみる．

再直交化の回数 i を k とする．明らかに MG-S 法では $k-1$ 回の Global sum 演算を必要とし，CG-S 法では 1 回である．このように通信時間の観点では，CG-S 法は MG-S 法に比べて優れているといえる．しかしながら CG-S 法は，MG-S 法よりも精度が悪くなる場合がある．したがって精度と速度の観点から，RWD ではもっとも良い方式は決定できない．

これらのことから，RWD は有用な分散方式であるといえるが，RWD はこの論文では議論しない．この理由はベクトルの要素を分割すると，多くの反復解法でユーザにとって使いにくい場合が生じ，実装についても困難となる場合が多いからである☆．

4.　混合 Gram-Schmidt (HG-S) 法の提案

この章では，新しい並列再直交化法である混合 G-S (Hybrid Gram-Schmidt，HG-S) 法を提案する．

4.1　基本的なアイデア

我々は既に，G-S 法では精度と実行時間に関するト

☆ RWD のその他の問題としては，固有ベクトル計算のための逆反復法では，RWD を採用すると逆反復法中の LU 分解が逐次化されてしまうという問題がある．

レードオフが存在することを述べた．以下は CWD における，そのまとめである．

- [**CG-S 法**] 精度：低，並列性：あり
- [**MG-S 法**] 精度：高，並列性：なし

この問題を緩和するため，再直交化について CG-S 法と MG-S 法の両方を利用することは合理的な手段である．我々は，この「混合された」方法を混合 G-S 法 (HG-S 法) とよぶ．

4.2 混合 G-S 再直交化法

図 **7** は，HG-S 法の概要である．図 7 の疑似コードは，本質的な再直交化処理の利用形態を示している点に注意する．なぜなら，多くの反復解法アルゴリズム，例えば固有値計算における逆反復法などは，図 7 のような再直交化処理を必要とするからである．

ここで高い精度を得ることを期待して，CG-S 法の後に MG-S 法を適用するという方式を設定した点に注意する．なぜなら図 7 の HG-S 法では，以前の反復 $i-1$ 中において MG-S 法で再直交化されたベクトルを用いて，現在の反復 i 中で CG-S 法が実行される処理になっているからである．

```
do i = 1,...,max_num_vectors
....
   do iter = 1,...,max_iter
      ....
      CG-S 法を用いた再直交化 (q_i)
      ....
      if (q_i は収束) break iter-ループ
   enddo
   MG-S 法を用いた再直交化 (q_i)
....
enddo
```

図 7 再直交化での HG-S 法．表記 $max_num_vectors$ と max_iter は，直交化するベクトル数，この反復法での最大の反復回数，をそれぞれ示している．

5. 性能評価

この章では，2 種の分散メモリ型スーパーコンピュータ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，固有値計算における逆反復法 (Inverse Iteration Method，IIM) に並列化された再直交化を実装した．

5.1 計算機環境

ここでは性能評価のため，東京大学情報基盤センタの HITACHI SR8000/MPP，および京都大学学術情報メディアセンタの Fujitsu VPP800/63 を用いてその効果を調べる．

HITACHI SR8000/MPP は分散メモリ型の並列計算機であり，メッセージ交換により通信を行うことができる．本実験で使用した SR8000/MPP は，東京大学情報基盤センタが所有する 144 ノード構成のものである．SR8000/MPP の各ノードは 8 つの PE を有し，各ノードにおける理論性能は 14.4GFLOPS，各ノードのメモリは 16 GB である．また通信網のトポロジは 3 次元ハイパークロスバ網であり，最大通信性能は片方向で 1.6 Gbytes/s，双方向で 3.2Gbytes/s である．本実験では，HITACHI Optimized Fortran90 V01-04 コンパイラが使われている．またコンパイラオプションは *-opt=ss -parallel=0* である．通信ライブラリとして，日立製作所が提供している MPI (Message Passing Interface) を用いている．

Fujitsu VPP800/63 はベクトル並列型の並列計算機であり，メッセージ交換により通信を行うことができる．本実験で使用した VPP800/63 は，京都大学学術情報メディアセンタが所有する 63 ノード構成のものである．VPP800/63 の各ノードは 8GFLOPS のベクトルユニット，および 1GOPS のスカラユニットからなり，各ノードのメモリは 8 GB である．また通信網のトポロジはクロスバ網であり，最大通信性能は 3.2Gbytes/s である．本実験では，Fujitsu UXP/V Fortran/VPP V20L20 コンパイラが使われている．またコンパイラオプションは*-O5 -X9* である．通信ライブラリとして，富士通が提供している MPI を用いている．

5.2 並列逆反復法 (IIM) の詳細

我々は，固有値計算における IIM[1] に並列の再直交化を実装した．IIM のアルゴリズム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：全固有ベクトル
- アルゴリズム：Rayleigh 商改良付き IIM
- 最大反復回数：40
- 要求誤差：$||Tx_i - \lambda_i x_i||_2 \leq ||T||_1 \times \epsilon$, $(i = 1, 2, ..., n)$，ここで ϵ はマシンイプシロンで，$x_i \in \Re^n$ はこの固有方程式の第 i 番固有ベクトル，$\lambda_i \in \Re$ 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，$|\lambda_i - \lambda_{i-1}| < eps$, $(i = 2, 3, .., n,$ 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)

我々は既に効率の良い並列固有値ソルバを開発している[9]．IIM はこの固有値ソルバに採用されているので，固有値ソルバにおける IIM ルーチンの実行時間を計測することで性能を評価する．

次に本実験で用いた，IIM の行列のデータ分散について説明する．

いま $nprocs$ 台の PE を利用し，その PE 番号を $myid = 0, 1, ..., nprocs-1$ とする．このとき，各データを所有する PE 番号は以下のようになっている．

- 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：$i/(n/nprocs)$ 番 PE が所有
- 固有ベクトル x_i：$i/(n/nprocs)$ 番 PE が所有

ここで $n/nprocs$ は割り切れるものとした．

我々の並列 IIM ルーチン[9] では，大まかには図 **8** のような流れで固有値計算を行う．

```
⟨1⟩ do   i = np*myid + 1, np*(myid + 1)
⟨2⟩   if ( λ_i は重複固有値でない ) then
⟨3⟩     IIM による x_i の計算;
⟨4⟩   else
⟨5⟩     i 番より前の重複固有値 λ_j (j < i)
            に対応する固有ベクトル x_j (j < i)
            全てに x_i を再直交化しながら
            IIM による x_i の計算;
⟨6⟩   endif
⟨7⟩ enddo
```

図 **8**　並列 IIM の概略．なお $np \equiv n/nprocs$ とおいた．

図 8 中の行 ⟨5⟩ では，当然自分の PE 内に固有ベクトルが無い場合も再直交化するが，この時は通信を必要とする再直交化処理となる．この再直交化に伴う通信回数や通信量は，再直交化の手法により異なる．

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 (1)：PE 内再直交化として MG-S 法を用いた，並列 CG-S 法
- CG-S (2)：PE 内再直交化として CG-S 法を用いた，並列 CG-S 法
- MG-S ： 逐次化される，並列 MG-S 法
- HG-S ： CG-S(1) を利用した混合 G-S 法
- IRCG-S：反復改良 CG-S 法[1]（反復回数 2 ）
- NoOrt：再直交化なし

本実験では CG-S 法の実装方式に関して，2 種の並列実装方式が存在する．まず CG-S(1) は，PE 間は CG-S 法を用いるのだが，PE 内は MG-S 法を用いる実装方式である．CG-S(2) は PE 間は CG-S 法を用い，PE 内も CG-S 法を用いる方式である．

IRCG-S 法は，反復改良 CG-S 法とよばれ，複数回 CG-S 法を実行することで精度改善を行う方法である．この実験では，反復回数は 2 回に固定されている．

5.4　試 験 行 列

試験行列は以下の行列を選んだ．

- 行列 (1)：Frank 行列を三重対角化した行列
  - 行列のサイズ：10,000
  - 密集固有値とみなす距離 eps：58471
  - 密集固有値のグループ数：8
  - 最大密集固有値数：9,993

  Frank 行列は，以下のように定義される．

  $A = (a_{ij}) \equiv n - \max(i, j) + 1$
  $(i, j = 1, 2, ..., n)$
- 行列 (2)：Glued Wilkinson W_{21}^+ 行列, $\delta = 10^{-14}$
  - 行列のサイズ：10,000
  - 密集固有値とみなす距離 eps：64.5
  - 密集固有値のグループ数：1
  - 最大密集固有値数：10,000

  Glued Wilkinson 行列 W_g^+ は $\left[\ \delta,\ \ W_{21}^+,\ \ \delta\ \right]$, ($\delta \in \Re^1$, $W_{21}^+ \in \Re^{21\times 21}$) を対角要素としてもつ行列である．ここで行列 W_{21}^+ は，以下のように定義される．

$$W_{21}^+ \equiv \begin{bmatrix} 10 & 1 & & & & & \\ 1 & 9 & 1 & & & & \\ & 1 & \ddots & \ddots & & & \\ & & \ddots & 0 & \ddots & & \\ & & & \ddots & \ddots & 1 & \\ & & & & 1 & 10 \end{bmatrix}$$

固有値の直交精度を保つために，再直交化が実行される点に注意する．再直交化すべき最大の固有ベクトル数は，最大密集固有値数と同じである．

5.5　実 験 結 果

5.5.1　実 行 時 間

表 **1**，表 **2** は，5 種の並列再直交化方式を用いた IIM の実行時間を示している．スーパーコンピュータ利用環境の制約から，HITACHI SR8000/MPP では $\lfloor 64/\lceil p/8 \rceil \rfloor$ 時間以上のジョブは自動的に実行を中断させられる．ここで p は PE 数である．

5.5.2　直 交 精 度

表 **3**，表 **4** は，IIM での計算された固有ベクトルにおける直交精度を示している．

ここでこの直交精度は，計算された固有ベクトルからなる行列を X とすると，行列 $(X^TX - I)$ の Frobenius ノルムによって評価するものとする．$n \times n$ 行列 $A = (a_{ij})$，$i, j = 1, ..., n$ の Frobenius ノルム $||A||_F$ は以下のように定義される：

$$||A||_F \equiv \sqrt{\sum_{i,j=1}^{n} a_{ij}^2}\ . \qquad (1)$$

しがって $||A||_F \leq n$ であり，k 本の固有ベクトルが直交化しない場合の上限は $||A||_F \leq \sqrt{k(k-1)}$ 程度である．

5.6　考　　察

5.6.1　並列実行時間について

表 1，表 2 では，計算機の種類や問題の性質によらず古典 G-S 法を用いると台数効果が確認できるが，修正 G-S 法を利用すると台数効果がほどんど確認できない．この理由は，理論上修正 G-S 法の並列性がないことによる．

表 1(a)MG-S では，なぜか少々の台数効果が確認できるが，これは直交化が不要な固有ベクトルが存在するため並列性があること，およびキャッシュの効果のため，と推定される．一方で混合 G-S 法は収束後 1 回修正 G-S 法を適用するため，純粋な古典 G-S 法と比べ計算量が多いのだが，古典 G-S 法と実行時間に大差がない．この理由は，反復回数が多いので修正 G-S の時間が無視できることが主な理由と考えられる．

表 1　IIM での各方式の実行時間．(HITACHI SR8000/MPP) 単位は秒．表記 > は，実行時間の制約内で収束しなかったことを意味している．

(a) 行列 (1): 三重対角化された Frank 行列

#PEs(IPs)	CG-S(1)	CG-S(2)	MG-S	HG-S	IRCG-S	NoOrt
8	6,604	6,527	38,854	6,604	12,883	23
16	3,646	3,526	24,398	3,574	6,987	12
32	2,061	2,059	28,050	2,082	3,906	7
64	1,633	1,559	27,960	1,614	3,059	3
128	2,091	2,204	>	2,027	3,978	1

(b) 行列 (2): Glued Wilkinson 行列 ($\delta = 10^{-14}$)

#PEs(IPs)	CG-S(1)	CG-S(2)	MG-S	HG-S	IRCG-S	NoOrt
8	43,005	43,188	>	44,574	>	103
16	22,714	22,534	>	22,969	44,798	50
32	13,306	13,280	>	13,560	25,313	27
64	9,866	10,108	>	10,180	19,378	13
128	13,796	13,990	>	14,056	>	7

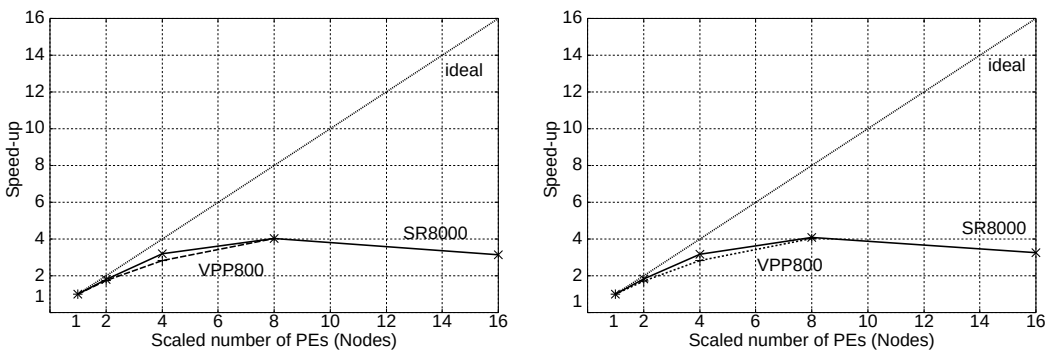
表 2　IIM での各方式の実行時間．(Fujitsu VPP800/63) 単位は秒．

(a) 行列 (1): 三重対角化された Frank 行列

#Node	CG-S(1)	CG-S(2)	MG-S	HG-S	IRCG-S	NoOrt
4	1,950	1,951	3,873	1,946	3,449	317
8	1,124	1,126	3,729	1,125	1,942	166
16	689	689	3,686	690	1,133	88
32	483	483	3,655	484	751	47

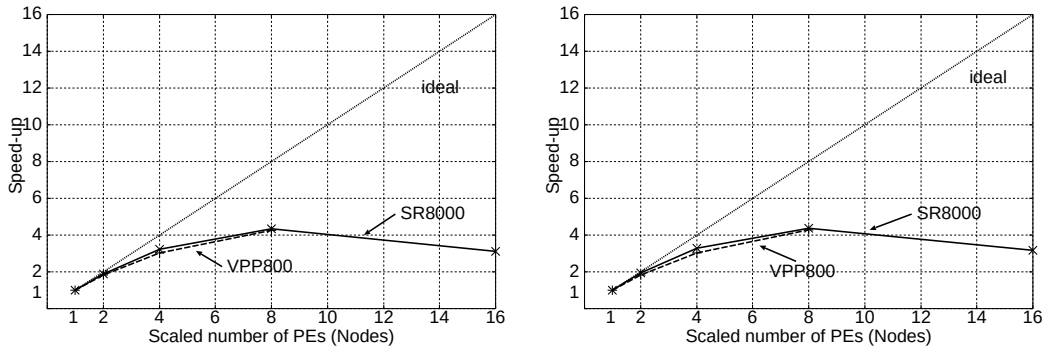
(b) 行列 (2): Glued Wilkinson 行列 ($\delta = 10^{-14}$)

#Node	CG-S(1)	CG-S(2)	MG-S	HG-S	IRCG-S	NoOrt
4	11,011	11,006	26,802	11,219	21,165	259
8	6,049	6,046	26,715	6,149	11,243	130
16	3,634	3,634	26,848	3,691	6,418	65
32	2,590	2,588	27,109	2,621	4,292	32



(a) CG-S(1) 法の台数効果　　(b) HG-S 法の台数効果

図 9　並列再直交化アルゴリズムの台数効果 (Frank 行列)．SR8000/MPP では 8PE，VPP800/63 では 4Node の実行時間で正規化されている．



(a) CG-S(1) 法の台数効果　　(b) HG-S 法の台数効果

図 10　並列再直交化アルゴリズムの台数効果 (Glued Wilkinson 行列) . SR8000/MPP では 8PE , VPP800/63 では 4Node の実行時間で正規化されている .

表 3　IIM での各方式による固有ベクトルの直交精度 . (HITACHI SR8000/MPP) 単位は Frobenius ノルム .

(a) 行列 (1): 三重対角化された Frank 行列

(8PE の MG-S における最大残差ベクトル $\max_i(||Ax_i - \lambda_i x_i||_2) = 1.61\text{E-}7$)

#PEs(IPs)	CG-S(1)	CG-S(2)	MG-S	HG-S	IRCG-S	NoOrt
8	6.48E-13	6.50E-13	6.67E-13	6.48E-13	6.45E-13	1.414
16	6.62E-13	6.62E-13	6.66E-13	6.62E-13	6.61E-13	1.414
32	6.89E-13	6.89E-13	6.66E-13	6.89E-13	6.89E-13	1.414
64	9.42E-13	9.41E-13	6.66E-13	9.41E-13	9.41E-13	1.414
128	1.54E-12	1.54E-12	–	1.54E-12	1.549E-12	1.414

(b) 行列 (2): Glued Wilkinson 行列 ($\delta = 10^{-14}$)

(8PE の CG-S における最大残差ベクトル $\max_i(||Tx_i - \lambda_i x_i||_2) = 2.21\text{E-}11$)

#PEs(IPs)	CG-S(1)	CG-S(2)	MG-S	HG-S	IRCG-S	NoOrt
8	2.26E-11	4.66E-11	–	2.26E-11	–	166
16	1.34E-5	1.34E-4	–	1.34E-5	1.34E-4	163
32	5.12	5.12	–	5.12	5.12	179
64	11.5	11.5	–	11.5	11.5	218
128	22.9	22.9	–	264	–	278

表 4　IIM での各方式による固有ベクトルの直交精度 . (Fujitsu VPP800/63) 単位は Frobenius ノルム .

(a) 行列 (1): 三重対角化された Frank 行列

(4Node の MG-S における最大残差ベクトル $\max_i(||Ax_i - \lambda_i x_i||_2) = 7.59\text{E-}9$)

#Node	CG-S(1)	CG-S(2)	MG-S	HG-S	IRCG-S	NoOrt
4	4.88E-13	4.96E-13	5.07E-13	4.88E-13	4.78E-13	1.414
8	4.85E-13	4.88E-13	5.07E-13	4.85E-13	4.78E-13	1.414
16	4.99E-13	5.01E-13	5.07E-13	5.00E-13	4.95E-13	1.414
32	5.43E-13	5.43E-13	5.07E-13	5.43E-13	5.41E-13	1.414

(b) 行列 (2): Glued Wilkinson 行列 ($\delta = 10^{-14}$)

(4Node の MG-S における最大残差ベクトル $\max_i(||Tx_i - \lambda_i x_i||_2) = 2.32\text{E-}11$)

#Node	CG-S(1)	CG-S(2)	MG-S	HG-S	IRCG-S	NoOrt
4	8.62E-12	4.94E-11	1.92E-12	8.60E-12	8.42E-12	161
8	2.26E-11	3.47E-11	1.94E-12	2.33E-11	2.28E-11	156
16	1.38E-5	1.41E-5	1.91E-12	1.47E-5	1.48E-5	171
32	5.09	5.09	1.88E-12	5.09	5.09	167

また全く直交化しないと，直交化する場合に比べて数十–数百倍程度高速化される．このことから再直交化処理自体の計算量は多く，なるべく避けたい処理であることがわかる．特に NoOrt で表 1(a)128PE の時，10,000 次元の固有ベクトル 10,000 個の求解時間が約 1 秒というのは衝撃的である．

図 **9**，図 **10** は，古典 G-S 法と混合 G-S 法の実行時間に関する正規化された台数効果を示している．図 9，図 10 からこの条件では，計算機の種類や問題の性質に依存せず約 4 倍の高速化が得られるといえる．

5.6.2 直交精度について

表 3(a)，表 4(a) では Frank 行列の場合，どの直交化方式を用いても得られる直交精度には大差がない．したがってこの場合は，古典 G-S 法を利用した直交化をするほうが速度の観点から望ましい．

一方で表 3(b)，表 4(b) では，古典 G-S 法を利用すると PE 数が増加するに従い直交精度が悪化する．この理由は，(1) 乱数を用いた初期ベクトル生成について，PE 数が増えると全く同一の初期ベクトルが利用される実装法，(2) 古典 G-S 法の理論的な直交性の崩れ，および (3) 逐次の古典 G-S 法に対し計算順序が変更される，などの要因が考えられるが，詳細な解析は今後の課題といえる．またこの試験行列については，混合 G-S 法が修正 G-S 法よりも直交精度の観点で優れている結果は得られなかった．

6. おわりに

本論文では，並列再直交化アルゴリズムの全体性能を左右するにもかかわらず性能評価がほとんどなされてこなかった古典 G-S 法の並列アルゴリズムを，固有ベクトル計算における逆反復法に実装して複数のスーパーコンピュータ上で性能評価を行った．また並列性と直交精度とのトレードオフ問題について，新しい方式によりこの問題を緩和する混合 G-S 法を提案した．

性能評価の結果，古典 G-S 法を利用すると修正 G-S 法を用いた方法では全く速度向上できなかった問題に対して，行列の特性に関係なく 4 倍程度の並列性が抽出できることが判明した．また本性能評価の行列に限っては，混合 G-S 法が古典 G-S 法に対して精度が優れている結果は得ることができなかった．したがって精度保証をするために，従来法である数値特性を評価して古典 G-S 法と修正 G-S 法とを切替え精度保証をする方法[8] を利用する必要がある．

今後の課題として，スーパーコンピュータのような高速な通信網を所有していない並列計算環境，たとえば PC クラスタ上での性能評価および古典 G-S 法の適用可能性を，実行時間をモデル化することで評価する必要がある．またこの実行時間予測から古典 G-S の選択を実行時に行い，従来手法を用い精度を保証するため動的に修正 G-S 法に切替えるという並列アルゴリズムも構築できるので，この並列アルゴリズムの評価も必要である．さらに異なる並列化アプローチ，すなわち再直交化に Gram-Schmidt 法を採用するのではなく，Householder 法を採用する方法[5] との比較も重要な課題である．

謝　　辞

日頃議論頂く，電気通信大学情報システム学研究科弓場・本多研究室の諸氏に感謝致します．なお本研究は，科学技術振興事業団戦略的創造研究推進事業さきがけプログラムの助成による．

参考文献

1) Parlett, B. N.: *The Symmetric Eigenvalue Problem*, SIAM (1997).
2) Bischof, C. and van Loan, C.: The WY Representation for Products of Householder Matrices, *SIAM J. Sci. Stat. Comput.*, Vol. 8, No. 1, pp. s2–s13 (1987).
3) 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).
4) 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).
5) Yamamoto, Y., Igai, M. and Naono, K.: A New Algorithm for Accurate Computation of Eigenvectors on Shared-Memory Parallel Processors, *Proceedings of Joint Symposium on Parallel Processing (JSPP)'2000*, pp. 19–26 (2000). in Japanese.
6) Dongarra, J. J., Duff, I. S., Sorensen, D. C. and van der Vorst, H. A.: *Numerical Linear Algebra for High-Performance Computers*, SIAM (1998).
7) Katagiri, T.: A Study on Large Scale Eigensolvers for Distributed Memory Parallel Machines, *Ph.D Thesis, the Department of Information Science, the University of Tokyo* (2000).
8) 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).
9) Katagiri, T. and Kanada, Y.: An Efficient Implementation of Parallel Eigenvalue Computation for Massively Parallel Processing, *Parallel Computing*, Vol. 27, pp. 1831–1845 (2001).