情報処理学会論文誌　コンピューティングシステム　Vol. 3　No. 2　30–38 (June 2010)

密正方行列特異値分解における並列 I-SVD 法の特性を用いた後処理の高速化

豊川 博己[†1]　山本 有作[†2]　木村 欣司[†1]
高田 雅美[†3]　中村 佳正[†1]

本稿では，並列 I-SVD 法を用いて密正方行列の特異値分解を行う．そのために，Bischof らのアルゴリズムと村田法を前処理および後処理として用いる．この際，村田法の逆変換に用いる行列の作成と並列 I-SVD 法を同時に行う．これにより，前処理および後処理を含めた特異値分解の計算時間を短縮することが可能となる．

Acceleration of the Post-processing in the Singular Value Decomposition for Dense Square Matrices Using Properties of the Parallel I-SVD Algorithm

Hiroki Toyokawa,[†1] Yusaku Yamamoto,[†2]
Kinji Kimura,[†1] Masami Takata[†3]
and Yoshimasa Nakamura[†1]

We compute the singular value decomposition for dense square matrices with the parallel I-SVD algorithm. For this purpose, we apply the Bischof's algorithm and Murata's algorithm as pre-processing and post-processing. Furthermore construction of the matrices used for the inverse transformation of Murata's algorithm, and the parallel I-SVD algorithm are performed simultaneously. As a result, the number of idle computation cores decreases, and SVD is achieved faster.

1. はじめに

2 重対角行列の特異値分解アルゴリズムの 1 つとして，I-SVD 法[8] がある．この方法では，行列の特異値は mdLVs 法によって計算され，その特異値に対応する特異ベクトルは dLV 型ツイスト分解によって求められる．I-SVD 法を並列処理するために，特異値計算において split[5] によって生じる行列の分割や，各特異値に対応する特異ベクトルが独立に計算されるという特徴を活かしたものが提案されている[9),10)]．

並列 I-SVD 法は 2 重対角行列向けのアルゴリズムであるため，密正方行列を直接扱うことはできない．そのため，特異値が等しい密正方行列と 2 重対角行列を相互に変換する前処理・後処理が必要となる．代表的なアルゴリズムとしてハウスホルダ法[3] があるが，データの再利用性が低いためキャッシュメモリを活用できない．

ハウスホルダ法を改良したアルゴリズムとして，Dongarra らのアルゴリズム[4] や，Bischof らのアルゴリズム[2] による下三角帯行列化と村田法[6] による 2 重対角化の組合せがある．Bischof らのアルゴリズムと村田法の組合せは，Dongarra らのアルゴリズムに比べて計算量は大きい．しかし level-3 BLAS を用いることが可能となるため，マルチコアプロセッサのための level-3 BLAS を用いることで，Dongarra らのアルゴリズムより高速に 2 重対角化を行うことができる可能性がある．

本研究では，マルチコアプロセッサにおける密正方行列の特異値分解を計算するために，マルチコアプロセッサ環境に有利な Bischof らのアルゴリズムと村田法による前処理および後処理と，並列 I-SVD 法[9),10)] を組み合わせる方法を提案する．提案手法では，村田法の逆変換の処理の一部を並列 I-SVD 法において待機状態のコアで実行する．これによってコアの稼働率が向上し，密正方行列の特異値分解の計算時間が短縮される．

2. 特異値分解と前処理・後処理

2.1 節において，2 重対角行列の特異値分解法を紹介する．2.2 節では，2 重対角行列のための特異値分解法を密正方行列に適用する際に必要となる前処理と後処理について論じる．

†1 京都大学大学院情報学研究科
Graduate School of Informatics, Kyoto University
†2 神戸大学大学院システム情報学研究科
Graduate School of System Informatics, Kobe University
†3 奈良女子大学大学院人間文化研究科
Graduate School of Humanities and Sciences, Nara Women's University

2.3 節において，本研究で用いる密正方行列に対する特異値分解の手順を説明する．

2.1　特異値分解

正方行列 $A \in \mathbf{R}^{N \times N}$ に対する特異値分解（SVD）とは，A を次のように行列 U，Σ，V に分解することである．

$$A = U\Sigma V^T$$

ただし，U，$V \in \mathbf{R}^{N \times N}$ は直交行列，$\Sigma \in \mathbf{R}^{N \times N}$ は非負の要素を持つ対角行列である．Σ の k 番目の要素は，A の k 番目の特異値 σ_k であり，U，V の k 番目の列ベクトルは，それぞれ k 番目の左特異ベクトル，右特異ベクトルである．なお，特異値分解自体は正方行列だけでなく長方行列にも適用できるが，本研究においては正方行列のみを扱うものとする．

代表的な 2 重対角行列の特異値分解法として，QR 法，分割統治法，I-SVD 法がある．これらの方法の性能を比較するために，サイズが 500×500 から 5000×5000 の 2 重対角行列を疑似乱数を用いて 100 個生成する．各方法のルーチンとして，LAPACK [14] で公開されている QR 法のルーチン（DBDSQR），分割統治法のルーチン（DBDSDC），文献 13) において公開されている I-SVD 法のルーチン（DBDSLV）を用いる．計算結果の精度を調べるため，計算された特異値と特異ベクトルを用いて $||BV - U\Sigma||_E$ を計算する．また直交性を調べるため $||U^TU - I||_E$，$||V^TV - I||_E$ を計算する．その結果を**図 1**，**図 2**，**図 3** に示す．これらの図の横軸は行列サイズを表し，縦軸はそれぞれの計算結果を表す．これらの図から，QR 法と I-SVD 法のルーチンはこれら 100 個の行列を正しく特異値分解していることが分かる．一方，分割統治法は 3 つの行列の分解性能が明らかに悪い．原因は不明であるが，これらの行列では特異値分解に失敗しているものと考えられる．以上のことから，QR 法や I-SVD 法は分割統治法と比べて信頼性が高いと考えられる．

図 4 に，各方法の計算時間を示す．なお，いずれの方法も 1 つの計算コアで動作させており，並列動作はさせていない．図 4 から，I-SVD 法や分割統治法は QR 法に比べて明らかに高速であることが分かる．以上より，高速かつ信頼性の高い特異値分解を行うために，

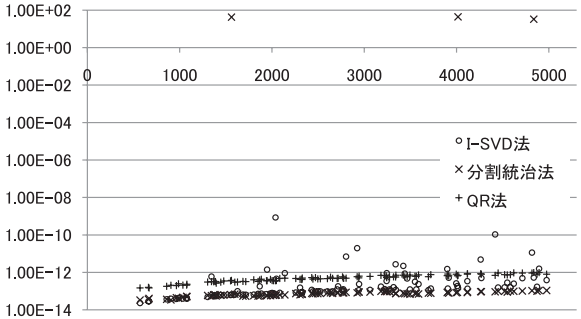


図 1　各アルゴリズムの計算精度
Fig. 1　Accuracy of the computed singular vectors.

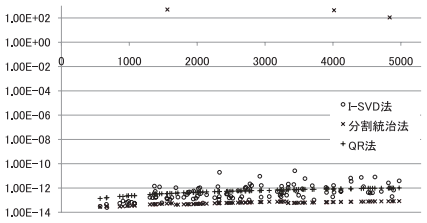


図 2　各アルゴリズムの左特異ベクトルの直交性
Fig. 2　Orthogonality of the computed left singular vectors.

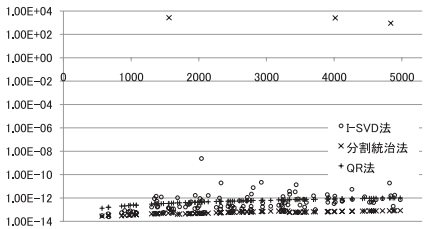


図 3　各アルゴリズムの右特異ベクトルの直交性
Fig. 3　Orthogonality of the computed right singular vectors.

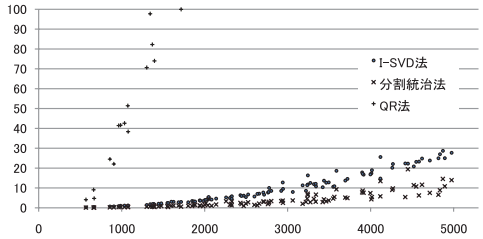


図 4　各アルゴリズムの計算速度
Fig. 4　Execution time for SVD of random bidiagonal matrices.

I-SVD 法を用い，さらにこれを並列化するべきであると考えられる．

2.2　前処理・後処理

密正方行列を特異値分解する場合は，計算速度の観点から通常は特異値分布が等しい 2 重対角行列に変換して特異値分解を行う．そのため密正方行列 $A \in \mathbf{R}^{N\times N}$ を 2 重対角行列 $B \in \mathbf{R}^{N\times N}$ に変換する前処理と，2 重対角行列 B の特異ベクトル U_B，$V_B \in \mathbf{R}^{N\times N}$ を密正方行列 A の特異ベクトル U，$V \in \mathbf{R}^{N\times N}$ に変換する後処理が必要となる．密正方行列を 2 重対角行列に直接変換する代表的なアルゴリズムとしてハウスホルダ法がある．この変換では，データの再利用性が低くキャッシュメモリを効率的に利用することができない．そのため，ハウスホルダ法を改良したアルゴリズムがいくつか考案されている．

2.2.1 項および 2.2.2 項では，ハウスホルダ法を改良したアルゴリズムとして代表的な Dongarra らのアルゴリズムと，Bischof らのアルゴリズムと村田法の組合せを説明する．2.2.3 項において，マルチコアプロセッサ上での計算速度を比較する．

2.2.1　Dongarra らのアルゴリズム

ハウスホルダ法では，鏡像変換を施すことで 1 列ずつ 2 重対角化していく．この処理は行列ベクトル積や rank-1 更新が大半を占め，BLAS を用いても高速化の効果は期待できない．

これを改善するために，Dongarra らのアルゴリズムでは鏡像変換に用いるベクトルを m 本作成し，これらをまとめて作用させることで m 列ずつ 2 重対角化する．これにより，rank-1 更新が rank-m 更新となり，rank-m 更新は行列積を用いて実装可能なため，キャッシュメモリを活用した高速化が可能となる．

2.2.2　Bischof らのアルゴリズムと村田法

この手法ではブロック鏡像変換を用いる Bischof らのアルゴリズムを適用して密正方行列を下三角行列に変換し，その帯行列を村田法によって 2 重対角行列に変換する．

図 5 は，ブロック鏡像変換を用いた Bischof らのアルゴリズムの概念図である．まずブロック鏡像行列 $P_B^{(1)}$ を用意し A の右から作用させることで，最も左上の $L\times L$ ブロックを下三角行列にし，それ以外の最上段のブロックをゼロにする．さらに $P_B^{(1)}$ と同様なブロック鏡像行列 $Q_B^{(1)}$ を A の左から作用させることで，A のうち左側の L 列と上側の L 行を半帯幅 L の下三角帯行列に変換する．これを N/L 回繰り返すことで，密正方行列 A を下三角帯行列 C に変換する．

次に下三角帯行列 $C = \{c_{i,j}\}$ を下 2 重対角行列 B に変換する．**図 6** は，村田法を用いた変換図である．まずハウスホルダ変換と同様の相似変換 $Q_M^{(i,j)}$，$P_M^{(i,j)}$ を考える．ここで $Q_M^{(i,j)}$ は第 j 列の第 $i+1$ 要素以降を消去するハウスホルダ変換であり，これを式で書くと

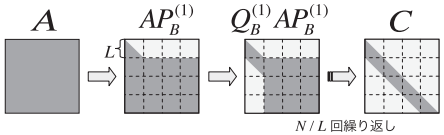


図 5　ブロック鏡像変換による下三角帯行列化
Fig. 5　Transformation to lower banded matrices using block reflector.

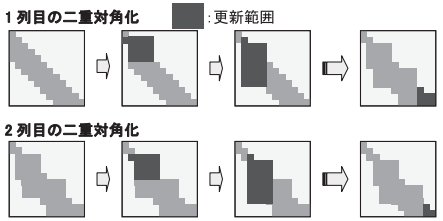


図 6　村田法による 2 重対角化
Fig. 6　Bidiagonalization by Murata's algorithm.

$$
\begin{aligned}
Q_M^{(i,j)} &= I - w_{i,j}w_{i,j}^T/h \\
w_{i,j} &= (0,\ldots,0,c_{i,j} \pm S_{i,j}, c_{i+1,j},\ldots,c_{N,j})^T \\
S_{i,j}^2 &= c_{i,j}^2 + c_{i+1,j}^2 + \ldots + c_{N,j}^2 \\
h &= S_{i,j}^2 + |c_{i,j}|S_{i,j}
\end{aligned}
$$

となる．同様に $P_M^{(i,j)}$ は第 i 行の第 $j+1$ 要素以降を消去するハウスホルダ変換であり，$Q_M^{(i,j)}$ を $w_{i,j} = (0,\ldots,0,c_{i,j} \pm S_{i,j}, c_{i,j+1},\ldots,c_{i,N})^T$，$S_{i,j}^2 = c_{i,j}^2 + c_{i,j+1}^2 + \ldots + c_{i,N}^2$ としたものである．通常のハウスホルダ変換では $i = j+1$ として，C の左右から順次 $Q_M^{(i,j)}$，$P_M^{(i,j)}$ を作用させていくが，これでは帯の外に非ゼロ要素が生成されてしまい，数回の変換で帯行列 C は密行列となる．これを防ぐため，$i = j+1$ ではなく $i = j+k$ とした変換も組み合わせて作用させることを考える．まずは C に $Q_M^{(2,1)}$ を左から作用させて，C の第 1 列 $c_{3,1}, c_{4,1},\ldots$ をゼロにし，次に $P_M^{(1,1)}$ を右から作用させて，第 1 行の $c_{1,2}, c_{1,3},\ldots$ をゼロにする．次に C の第 2 列に注目して，先ほどの $Q_M^{(2,1)}$，$P_M^{(1,1)}$ によって帯の外に生じた非ゼロ要素を消すように変換 $Q_M^{(L+2,2)}$ を行い，$c_{L+3,2}, c_{L+4,2},\ldots$ をゼロにする．さらに，C の第 2 行に注目して $P_M^{(2,2)}$ を作用させて，下三角帯の外に生成された非ゼロ要素 $c_{2,3}, c_{2,4},\ldots$

表 1　各アルゴリズムの計算量
Table 1　Computational cost.

	Dongarra ら	Bischof ら	村田法
順変換	$8N^3/3$	$8N^3/3$	$8LN^2$
逆変換	$4N^3$	$4N^3$	$4N^3$

（$L \ll N$ の場合）

をゼロにする．以降，$Q_M^{(i,j)}$，$P_M^{(i,j)}$ を，帯の外に生じた非ゼロ要素を消すように作用させることで，帯幅が約 2 倍になるだけで帯行列の形は保たれる．この処理を，すべての列が 2 重対角化されるまで行うことで，下三角帯行列 C は 2 重対角行列 B に変換できる．

2.2.3　アルゴリズムの計算速度

表 1 は，先の項で紹介した 2 つのアルゴリズムの理論上の計算量である．Dongarra らのアルゴリズムの方が計算量は少ない．しかし，Dongarra らのアルゴリズムは行列ベクトル積を必要とするため，level-2 BLAS を使う場面がある．一方の Bischof らのアルゴリズムは行列積が処理の大半を占めるため，level-3 BLAS を利用することができる．そのため，マルチコアプロセッサ上で BLAS[11] を用いて計算を行うことで，Bischof らのアルゴリズムが Dongarra らのアルゴリズムに対して計算時間の点で有利になる可能性がある．また，Bischof らのアルゴリズムや村田法は半帯幅 L によって計算量やキャッシュメモリのヒット率が異なるため，帯幅の違いによる計算時間の変化も調べる必要がある．

そこで，マルチコアプロセッサ上で使用するコア数ごとに，これら 2 つのアルゴリズムを比較する．この実験にあたって Dongarra らのアルゴリズムとして，LAPACK に含まれている分割統治法による密行列特異値分解ルーチン DGESDD の内部で用いられている前処理・後処理部分のコードを利用する．BLAS には GotoBLAS2[12] を用いる．**図 7** は，疑似乱数を用いて作成されたサイズが 4000×4000 の密正方行列に対して前処理・後処理を行った結果である．

図 7 より，半帯幅 L を 100〜200 としたときに，Bischof らのアルゴリズム・村田法の計算が早く終了することが分かる．また level-2 BLAS の計算が必要な Dongarra らのアルゴリズムに比べて，level-3 BLAS の計算が多い Bischof らのアルゴリズムでは，コア数を増やした際に計算速度が向上しやすくなっている．

2.3　並列 I-SVD 法の密正方行列への適用

2.1 節および 2.2 節の結果より，本研究では 2 重対角行列の特異値分解アルゴリズムとして I-SVD 法を採用する．また Bischof らのアルゴリズムと村田法の組合せを前処理とし，

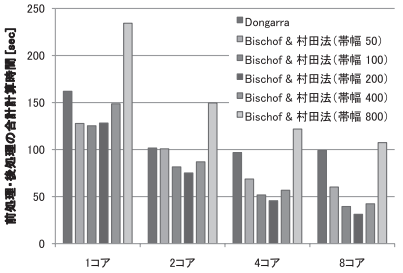


図 7　各アルゴリズムの計算速度
Fig. 7　Execution time for pre-process and post-process.

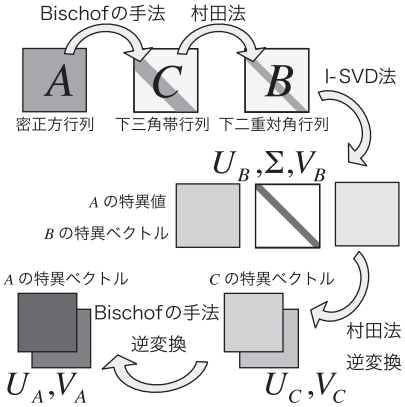


図 8　本研究における密正方行列の特異値分解の流れ
Fig. 8　Process flow of the SVD for a dense square matrix on this article.

その逆変換を後処理とする．**図 8** は密正方行列に対する特異値分解の流れ図である．この流れを以下に示す．

(1)　Bischof らのアルゴリズムを用いて，密正方行列 A を下三角帯行列 C に変換．
(2)　村田法を用いて，下三角帯行列 C を 2 重対角行列 B に変換．
(3)　I-SVD 法を用いて 2 重対角行列 B の特異値分解．
(4)　B の特異ベクトル U_B，V_B に対して村田法の逆変換を行い，C の特異ベクトル U_C，V_C を計算．

(5)　C の特異ベクトル U_C，V_C に対して Bischof らのアルゴリズムの逆変換を行い，A の特異ベクトル U，V を計算．

3. 並列化と高速化の手法

3.1 節において，Bischof らのアルゴリズム，村田法，I-SVD 法をそれぞれ単独で並列化する方法について述べる．3.2 節では，I-SVD 法と逆変換で用いる村田法を 1 つのアルゴリズムと見なし，そのアルゴリズムを並列化することによって全体の計算時間を短縮する方法を提案する．提案方法に GotoBLAS を適用した場合，メモリアクセスの競合が生じる可能性がある．これを回避するための方法について 3.4 節で述べる．

3.1 単純な並列化

Bischof らのアルゴリズムは，処理の大部分を行列計算が占めている．そのため，行列計算の部分に GotoBLAS のような並列 BLAS を用いることで，並列処理が可能となる．

村田法で各列を 2 重対角化する際には，鏡像変換 $Q_M^{(i,j)}$ もしくは $P_M^{(i,j)}$ を作用させていく．このとき，これらの変換は行列 C 全体のうちの図 6 の更新範囲にあたる $L \times L$ または $2L \times L$ の範囲のみに影響を及ぼす．そのため，k 列目の 2 重対角化をあるコアが行っている間に，k 列目の 2 重対角化に必要な変換が影響を及ぼす範囲に重ならないように注意しながら，別のコアが $k+1$ 列目の 2 重対角化を行うことで，村田法の並列化ができる．

I-SVD 法の並列化は，I-SVD 法を

(A)　行列の分割が起こるまで，mdLVs 法を用いて行列を特異値分解．

(B)　すでに計算されている特異値に対応する特異ベクトルを，dLV 型ツイスト分解を用いて計算．

という 2 種類の小さな処理（以下では**ジョブ**と呼ぶ）の集合として考え，各コアがジョブを次々と実行することによって可能となる．この方法を並列 I-SVD 法と呼ぶ．ジョブ (A) において行列の減次によって特異値が求められた場合は，それに対応する特異ベクトルを計算するジョブ (B) が生成される．また行列の分割が起きると，2 つに分かれた小行列それぞれに対してジョブ (A) が生成される．これらのジョブを空いているコアで順次処理することで，すべての特異値と特異ベクトルが計算される．

以上のように Bischof らのアルゴリズムとその逆変換，村田法とその逆変換，I-SVD 法をそれぞれ並列化することで，密正方行列 A の特異値分解を単純に並列化できる．

3.2 提 案 手 法

並列 I-SVD 法では，行列の特異値分布によってはジョブ (A)，(B) が生成される頻度が低い．特に mdLVs 法の初期の段階では分割や減次が発生しにくく，ジョブが十分な頻度で生成されないことがある．実際，行列の特異値分布によっても異なるが，文献 10) の数値実験ではコア数がおよそ 8 個以上の場合に待機状態となるコアが生じ，効率が悪いことが示されている．

後処理の中で行われる村田法の逆変換は，2 重対角化に用いた鏡像変換の逆変換を特異値・特異ベクトルに作用させることにより行う．ただし，1 個の鏡像変換の作用は level-2 BLAS 相当の演算であるため，level-3 BLAS が使えるように複数の鏡像変換をまとめて compact-WY representation [7] と呼ばれるブロック化を行ってから作用させる．したがって，村田法の逆変換は次の 2 つの処理からなる．

(1)　村田法で用いたハウスホルダ変換から compact-WY representation の作成

(2)　B の特異ベクトルに compact-WY representation を作用

このうち compact-WY representation の作成は B の特異値・特異ベクトルの計算が完了する前に行うことができる．

以上をふまえ，密正方行列の並列化効率を向上させるために，並列 I-SVD 法の実行中に待機状態のコアを用いて村田法の逆変換を行う．つまり，「村田法の逆変換で用いる行列（compact-WY representation）を作成する」というジョブを作成し，並列 I-SVD 法のジョブ (A)，(B) を実行していないコアで計算させる．これにより，並列 I-SVD で十分な並列性を確保できない場合でも遊休状態になるコアの数を減らし，密正方行列の特異値分解全体にかかる時間を短縮することができる．

3.3 標準固有値問題との相違

Bischof らのアルゴリズムと村田法を組み合わせる手法は，標準固有値計算問題における 3 重対角化でも利用され，高速化に有効であることが示されている [1),2)]．しかし，標準固有値問題に対する既存研究では，3 重対角化，3 重対角行列の固有値分解，逆変換の 3 つのフェーズをそれぞれ独立に並列化・高速化する手法がとられている．一方，本研究では 2 重対角行列の特異値分解と逆変換の一部を並行して行うことにより，プロセッサコアの有効な利用を目指している．これは，高速・高精度ではあるが並列化時に待機状態となるコアが生じやすいという I-SVD 法の特徴と，compact-WY representation の生成を特異値・特異ベクトルの計算に先行して行えるという村田法逆変換の特徴を組み合わせた並列化手法である．

また，標準固有値問題における 3 重対角化では，内部で DSYMV など対称行列用の BLAS ルーチンを呼び出すことになる．これらのルーチンは行列の上三角または下三角部分のみを

参照して計算を行うため，通常の密行列用ルーチンよりも一般にデータ参照方法が複雑で並列化しにくい．一方で非対称行列の特異値分解のために 2 重対角化する場合は，対称行列用ではなく通常の非対称行列用 BLAS ルーチンを呼び出すため，BLAS を用いた場合に並列化の効果が出やすく，Bischof らのアルゴリズムと村田法の組合せがより有利になると考えられる．

以上の 2 つの点が，並列処理の観点から見た標準固有値問題と特異値分解の違いである．

3.4　メモリアクセスの競合の回避

GotoBLAS では行列積などの計算が並列化されており，並列化されていない BLAS に比べて，時間あたりの CPU–メモリ間データ転送量が多くなると考えられる．このような並列化された BLAS を用いるルーチンと同時に，メモリアクセスが多い他の処理を行った場合，要求されるメモリアクセスが CPU–メモリ間のバスの容量を超え，データ転送に要する時間が大幅に伸びる場合がある．たとえば，提案手法のように並列 I-SVD 法を行いながら，GotoBLAS のルーチンを使用して村田法の逆変換で用いる行列の作成を行った場合は，メモリアクセスの競合が起こる可能性が高い．

これを防ぐため提案手法においては，**図 9** のように村田法の逆変換に用いるコアの数を調整し，メモリアクセスの競合を抑えるようにする必要がある．mdLVs 法で特異値計算を行っている間は，村田法逆変換で用いる行列の作成に必要な GotoBLAS による処理を 1 コアのみを用いて行い，GotoBLAS によるメモリアクセスを抑える．mdLVs 法によってすべての特異値の計算が完了した後，I-SVD 法を行っているスレッドをスリープ状態にして，村田法逆変換の行列作成をすべてのコアを用いて行う．これにより GotoBLAS の性能を最大限に引き出して行列作成を行うことができる．その後，残りの特異ベクトルを計算するために，dLV 型ツイスト分解による特異ベクトル計算を行う．この時点で残っているジョブは，

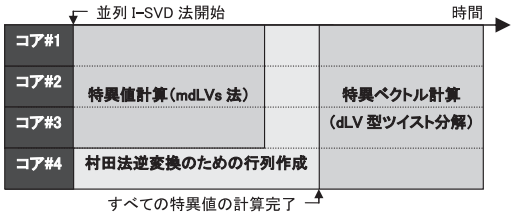


図 9　スレッドの動作の様子
Fig. 9　Behavior of working threads.

実行が完了するまでの時間が短く一定であるジョブ (B) のみなので，効率良く並列計算が行われる．

また，1 つのコアに複数のスレッドが割り当てられた場合，メモリアクセスや計算の効率が悪い．そのためプロセッサアフィニティを設定して，1 つのコアに必ず 1 つのスレッドが割り当てられるようにする．

4. 数 値 実 験

マルチコアプロセッサ上で Dongarra らのアルゴリズムと，Bischof らのアルゴリズムと村田法の組合せを用いた場合の，計算時間の差を調べる．また，Bischof らのアルゴリズムと村田法の組合せを用いた場合に，単純な並列化と提案手法との計算時間の差を調べる．

4.1　実 験 環 境

疑似乱数を用いてサイズが 4000×4000 および 8000×8000 の密正方行列を作成する．下三角帯行列 C の半帯幅 L はそれぞれ 100，200 とする．**表 2** は，実験に用いた計算機の性能を示す．

表 3 は，各方法を開発する際に用いた並列化手法をまとめたものである．実験で用いたプログラムは C++と Fortran で開発し，Intel 社のコンパイラでコンパイルする．Bischof らのアルゴリズムとその逆変換，村田法の逆変換は GotoBLAS を用いて行列計算部を並列実行させる．村田法の順変換は OpenMP を用いて並列化する．並列 I-SVD 法は，I-SVD ライブラリ 0.2.1 を基に，村田法逆変換で用いる行列を作成するジョブも処理するように変更したものを用いる．計算が効率良く行われるようにするため，環境変数 KMP_AFFINITY=scatter を設定し，1 つのコアに複数のスレッドが割り当てられないようにする．

4.2　実 験 結 果

密行列特異値分解の各部分でかかった計算時間を**表 4**，**表 5**，**図 10**，**図 11** に示す．ここ

表 2　実験環境
Table 2　Specification of the test computer.

CPU	Intel Xeon E5430 2.66 GHz （4 コア × 2 プロセッサ）
メモリ	66 GBytes
OS	Fedora Linux 10 （カーネル：2.6.27.5-117.fc10.x86_64）
コンパイラ	icpc & ifort 11.1（最適化オプション：-O3）
BLAS	GotoBLAS2-1.01

表 3　各処理に対する並列化手法
Table 3　Method for parallelization.

処理	並列化手法
Bischof らの手法 順変換	行列計算に GotoBLAS を使用
村田法 順変換	OpenMP を使用
並列 I-SVD 法	ジョブ管理に pthread を使用
村田法 逆変換	行列計算に GotoBLAS を使用
Bischof らの手法 逆変換	行列計算に GotoBLAS を使用

表 4　計算時間（$N = 4{,}000$, $L = 100$）
Table 4　Execution time ($N = 4{,}000$, $L = 100$).

コア数	1 コア		8 コア		
前処理・後処理		Bischof ら		Bischof ら & 村田法	
アルゴリズム	Dongarra ら	& 村田法	Dongarra ら	単純な並列化	提案手法
Dongarra 順変換	127.3 (0.14)		79.6 (0.08)		
Bischof 順変換		20.2 (0.05)		7.4 (0.04)	7.4 (0.03)
村田法 順変換		8.0 (0.04)		1.1 (0.00)	1.1 (0.00)
(a) 並列 I-SVD 法	10.0 (0.00)	10.0 (0.00)	2.2 (0.00)	2.2 (0.02)	
(b) 村田法逆変換 行列作成		2.7 (0.00)		1.8 (0.03)	
(a) & (b)					3.2 (0.10)
村田法逆変換 行列作用		65.1 (0.01)		24.6 (0.04)	24.6 (0.04)
Bischof 逆変換		29.4 (0.01)		4.5 (0.01)	4.5 (0.02)
Dongarra 逆変換	35.3 (0.01)		19.6 (0.05)		
その他の処理	1.7 (0.02)	3.2 (0.00)	1.7 (0.04)	3.3 (0.02)	3.3 (0.03)
合計時間	174.3 (0.14)	138.5 (0.09)	103.1 (0.10)	44.9 (0.05)	44.1 (0.10)

※括弧内の数字は標準偏差を表す　　単位：[sec]

で「その他の処理」は，メモリ確保や内部でのデータコピーなどの処理にかかる時間を表す．また，掲載している計算時間は同じ処理を 10 回行った際の平均値である．また計測値の標準偏差を括弧内に示した．これらの表・図から，前処理・後処理には Bischof らのアルゴリズムと村田法の組合せの方が，Dongarra らのアルゴリズムより高速であることが分かる．これは Dongarra らのアルゴリズムは level-2 BLAS を使用している部分があるため，マルチコアプロセッサの計算性能を十分に発揮させることができないためであると考えられる．

村田法逆変換と並列 I-SVD 法を同時に行う箇所では，実行時間に多少の変動が見られる．しかし，今回の実験でサイズ 4000×4000 および 8000×8000 の行列について 10 回ずつ計測した際には，提案手法の方が単純な並列化を行った場合よりも合計計算時間は短くなっており，提案手法の効果は確認できる．

表 5　計算時間（$N = 8{,}000$, $L = 200$）
Table 5　Execution time ($N = 8{,}000$, $L = 200$).

コア数	1 コア		8 コア		
前処理・後処理		Bischof ら		Bischof ら & 村田法	
アルゴリズム	Dongarra ら	& 村田法	Dongarra ら	単純な並列化	提案手法
Dongarra 順変換	1001.4 (3.06)		585.9 (0.43)		
Bischof 順変換		155.0 (0.22)		39.8 (0.06)	39.7 (0.07)
村田法 順変換		79.2 (0.07)		12.1 (0.04)	12.0 (0.02)
(a) 並列 I-SVD 法	41.3 (0.11)	41.6 (0.09)	10.2 (0.16)	10.5 (0.10)	
(b) 村田法逆変換 行列作成		18.2 (0.01)		6.8 (0.02)	
(a) & (b)					15.3 (0.54)
村田法逆変換 行列作用		461.2 (0.11)		123.2 (0.10)	123.0 (0.11)
Bischof 逆変換		219.5 (0.04)		30.6 (0.03)	30.6 (0.03)
Dongarra 逆変換	282.3 (0.82)		145.9 (0.20)		
その他の処理	6.7 (0.02)	25.7 (0.16)	6.7 (0.09)	26.2 (0.17)	25.5 (1.51)
合計時間	1331.7 (3.65)	1000.3 (0.40)	748.8 (0.56)	249.2 (0.32)	246.1 (2.09)

※括弧内の数字は標準偏差を表す　　単位：[sec]

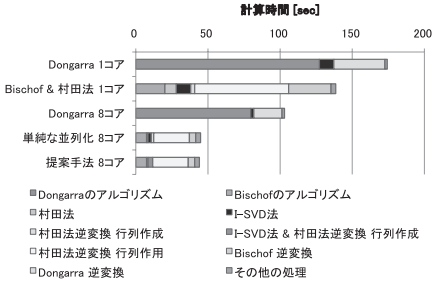


図 10　計算時間（$N = 4{,}000$, $L = 100$）
Fig. 10　Execution time ($N = 4{,}000$, $L = 100$).

村田法逆変換の行列を作用させたときの計算速度の向上率は，サイズ 8000×8000 の行列・8 コア並列のときに約 3.8 倍であり，他の計算部分に比べて並列化性能が出にくくなっている．これは約 $4L \times 2L$ や $2L \times 2L$ といった比較的小さいサイズの行列積を多数回行っているため，大規模行列の積を少数回行う場合と比べて GotoBLAS の性能を引き出しにくいためであると考えられる．

表 6，**図 12** は，並列 I-SVD 法と村田法逆変換のための行列作成にかかる時間のみを，単純な並列化と提案手法で比較した結果を表す．これらから，コア数が 8 個ある場合は並列

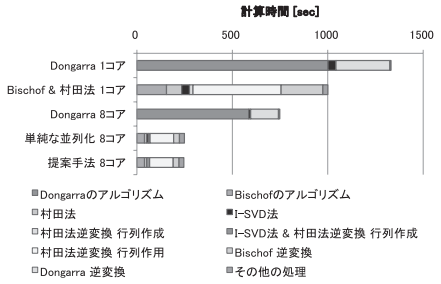


図 11　計算時間（$N = 8{,}000,\ L = 200$）
Fig. 11　Execution time ($N = 8{,}000,\ L = 200$).

表 6　提案手法により削減された計算時間
Table 6　Reduced time by the improvement.

行列サイズ N	4,000	8,000
(a) I-SVD 法	2.2	10.5
(b) 村田法逆変換行列作成	1.8	6.8
(c) = (a) + (b)	4.0	17.3
(d) 提案手法による (a) と (b)	3.2	15.3
提案手法の効果 (c) − (d)	0.8	2.0

単位：[sec]

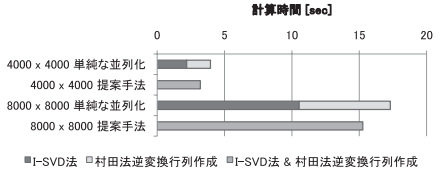


図 12　I-SVD 法と村田法逆変換行列作成の計算時間
Fig. 12　Execution time for the I-SVD method and constructing matrices for Murata's inverse transformation.

I-SVD 法と村田法逆変換のための行列作成にかかる時間の合計が 10〜20%程度削減されたことが分かる．

以上より，密正方行列の特異値分解において，提案された並列化手法は有効であると考えられる．

5. 結　　論

並列 I-SVD 法はジョブが生成される頻度が十分でない場合，多くのコアを持つコンピュータにおいて，いくつかのコアが待機状態になることがある．本研究においては，村田法逆変換で用いる行列を作成する処理を並列 I-SVD 法のジョブの 1 つとして扱い，並列 I-SVD 法と行列作成を同時に行うことで計算時間の短縮を図った．その結果，並列 I-SVD 法と行列作成に必要な合計計算時間を短縮できることが数値実験によって確認された．

参 考 文 献

1) Bischof, C.H., Lang, B. and Sun, X.: A Framework for Symmetric Band Reduction, *ACM Trans. Math. Software*, Vol.26, pp.581–601 (2000).
2) Bischof, C.H., Marques, M. and Sun, X.: Parallel Bandreduction and Tridiagonalization, *Proc. 6th SIAM Conference on Parallel Processing for Scientific Computing*, pp.22–24 (1993).
3) Demmel, J.W.: *Applied Numerical Linear Algebra*, SIAM Philadelphia (1997).
4) Dongarra, J.J., Hammarling, S.J. and Sorensend, D.C.: LAPACK Working Note 2 Block Reduction of Matrices to Condensed Forms for Eigenvalue Computations, *J. Comput. Appl. Math.*, Vol.27, pp.215–227 (1989).
5) Iwasaki, M. and Nakamura, Y.: Accurate Computation of Singular Values in terms of Shifted Integrable Schemes, *Japan J. Indust. Appl. Math.*, Vol.23, pp.239–259 (2006).
6) 村田健郎，堀越清視：対称帯行列を 3 重対角化するための新アルゴリズム，情報処理学会論文誌，Vol.16, pp.93–101 (1975).
7) Puglisi, C.: Modification of the Householder Method Based on the Compact WY Representation, *SIAM J. Sci. Stat. Comput.*, Vol.13, pp.723–726 (1992).
8) 高田雅美，木村欣司，岩崎雅史，中村佳正：高速特異値分解のためのライブラリ開発，情報処理学会論文誌，Vol.47, No.SIG7 (ACS14), pp.91–104 (2006).
9) Toyokawa, H., Kimura, K., Takata, M. and Nakamura, Y.: On Parallelism of the I-SVD Algorithm with a Multi-core Processor, *JSIAM Letters*, Vol.1, pp.48–51 (2009).
10) Toyokawa, H., Kimura, K., Takata, M. and Nakamura, Y.: On Parallelization of the I-SVD Algorithm and its Evaluation for Clustered Singular Values, *Proc. International Conference on Parallel and Distributed Processing Techniques and Application*, pp.711–717 (2009).
11) BLAS. http://www.netlib.org/blas/
12) GotoBLAS2. http://www.tacc.utexas.edu/ resources/software/

13) I-SVD Library. http://www-is.amp.i.kyoto-u.ac.jp/lab/isvd/download/
14) LAPACK. http://www.netlib.org/lapack/

(平成 21 年 10 月 2 日受付)
(平成 22 年 1 月 13 日採録)

豊川　博己

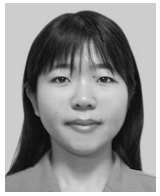
1985 年生．2008 年京都大学工学部情報学科（数理工学コース）卒業．2010 年同大学大学院情報学研究科数理工学専攻修士課程修了．同年新日鉄ソリューションズ（株）入社．在学中は数値計算ライブラリの開発，マルチコアプロセッサ環境を対象とした並列計算に関する研究を行っていた．

山本　有作（正会員）

1966 年生．1990 年東京大学工学部計数工学科（数理工学コース）卒業．1992 年同大学大学院工学系研究科物理工学専攻修士課程修了．同年（株）日立製作所中央研究所入所．2003 年名古屋大学大学院工学研究科助手．同准教授を経て，現在，神戸大学大学院システム情報学研究科計算科学専攻教授．数値解析，高性能計算とその応用に興味を持つ．博士（工学）．SIAM，INFORMS，日本応用数理学会各会員．

木村　欣司（正会員）

1976 年生．2004 年神戸大学大学院自然科学研究科情報メディア科学専攻博士課程修了．2006 年より京都大学大学院情報学研究科特定有期雇用助手．2007 年より新潟大学大学院自然科学研究科助教．2008 年より京都大学大学院情報学研究科特定講師．2009 年より京都大学大学院情報学研究科特定准教授．離散可積分系，計算機代数，数値解析に関する研究に従事．博士（理学）．2009 年日本数式処理学会最優秀奨励賞．日本応用数理学会，日本数式処理学会各会員．



高田　雅美（正会員）

1977 年生．2004 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修了．博士（理学）を同大学より取得．2004 年独立行政法人科学技術振興機構戦略的創造研究推進事業において，京都大学大学院情報学研究科にて委嘱研究員．2006 年奈良女子大学大学院人間文化研究科助手．2007 年奈良女子大学大学院人間文化研究科複合現象科学専攻助教．数値計算ライブラリの開発，分散メモリ環境を対象とする並列プログラムの開発に関する研究に従事．

中村　佳正（正会員）

1955 年生．1983 年京都大学大学院工学研究科博士課程修了．工学博士．岐阜大学助教授，同志社大学教授，大阪大学大学院基礎工学研究科教授等を経て，2001 年より京都大学大学院情報学研究科教授．可積分系，計算数学に関する研究に従事．日本数学会，日本応用数理学会，SIAM 各会員．