社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－HPC－111 (28)
2007／8／2

複素非対称行列向け固有値解法のCSX600による高速化

宮田考史† 山本有作† 中村佳正††

本論文では，複素非対称行列に対するヘッセンベルグ QR 法を ClearSpeed 社の浮動小数点コプロセッサ CSX600 を用いて高速化した結果について報告する．ベースとなるアルゴリズムとしては，Braman らにより提案された small-bulge マルチシフト QR 法を用いる．このアルゴリズムは，計算の大部分を行列乗算の形で実行でき，高性能アーキテクチャに適している．しかし，CSX600 の性能を引き出すには，シフト数を 200 以上に増やす必要があり，その場合，行列乗算以外の部分の実行時間が無視できなくなる．そこで本研究では，行列乗算以外で最も大きな時間を占める 2 つの部分，すなわち対角ブロック内部でのバルジ追跡と分離した小行列の固有値計算に着目し，それぞれに対して再帰型アルゴリズムへの変換とブロック化という技法を適用した．これにより，この 2 つの部分の実行時間を大幅に短縮できる．これらの最適化の結果，12,000×12,000 の複素ヘッセンベルグ行列の固有値とシューア標準形を求める場合，CSX600 を使うことで 3.2GHz Xeon に比べ 3.8 倍の高速化を達成できた．

Acceleration of a Complex Nonsymmetric Eigensolver with the CSX600 Floating-Point Coprocessor

TAKAFUMI MIYATA,† YUSAKU YAMAMOTO† and YOSHIMASA NAKAMURA††

In this paper, we show how to speed up the Hessenberg QR algorithm for computing the eigenvalues of a dense complex nonsymmetric matrix using the CSX600 floating-point coprocessor. Our approach is based on the small-bulge multishift QR algorithm proposed by Braman et al. This algorithm can perform a major part of the computation in the form of matrix multiplication and is therefore very suited to high performance architectures. However, to exploit the potential of the CSX600, the number of shifts must be increased to more than two hundreds. In that case, the parts other than matrix multiplication begin to occupy a considerable percentage of the execution time, thereby limiting the speedup. To solve this problem, we focus on two most time-consuming parts in the algorithm except for matrix multiplication: accumulation of bulge-chasing transformations and solution of a decoupled small eigenproblem. We reconstruct the former as a recursive algorithm and apply a blocking technique to the latter. This greatly reduces the execution time of these parts. As a result of these optimizations, we obtained up to 3.8 times speedup with the CSX600 processor over a 3.2GHz Xeon processor when computing the eigenvalues and the Schur canonical form of a 12,000 × 12,000 complex Hessenberg matrix.

1. はじめに

複素非対称密行列の固有値問題は，様々な応用分野で重要であり[2]，近年では 1 万元を超える大型の問題も解かれるようになっている．この問題を解くための標準的なアルゴリズムは，次の 4 ステップからなる：(1) ユニタリ変換によるヘッセンベルグ行列への変形，(2) ユニタリ変換の蓄積，(3) ヘッセンベルグ QR 法による固有値とシューア標準形の計算[8)9)12)]，(4) 固有ベクトルの計算．多くの場合，(3) のヘッセンベルグ QR 法が計算時間の大部分を占めるため，この部分の高速化が重要である．

最近，1 チップに多数の浮動小数点演算器を集積した専用プロセッサが開発され，大規模計算を高速に行う手段として注目を集めている．たとえば GRAPE-DR[11)] は 1 チップに 512 個の演算コアを集積し，最大 512GFLOPS の性能を持つ．また，ClearSpeed 社

† 名古屋大学大学院工学研究科計算理工学専攻
Department of Computational Science and Engineering, Graduate School of Engineering, Nagoya University
†† 京都大学大学院情報学研究科数理工学専攻 / JST SORST
Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University / JST SORST

の CSX600[5] は 1 チップに 96 個の演算コアを集積し，最大 48GFLOPS の性能を持つ．

本論文では，複素行列に対するヘッセンベルグ QR 法を CSX600 を用いて高速化することを目的とする．ヘッセンベルグ QR 法には様々な変種があるが，我々は Braman, Byers, Mathias により提案された small-bulge マルチシフト QR 法[4] をベースとして用いる．このアルゴリズムでは，複数のシフトを導入することで QR 法の基本演算であるバルジ追跡演算を複数個同時に行い，それにより並列粒度とデータ参照の局所性を高めている．さらに，複数個のバルジ追跡演算をまとめて行列乗算として実行することを可能にしている．これらの工夫により，small-bulge マルチシフト QR 法は，様々な高性能計算機上で高速に実行できるアルゴリズムとなっている．

CSX600 を使う場合，small-bulge マルチシフト QR 法において行列乗算の部分のみを CSX600 に担当させ，それ以外を CPU で実行することにより，ある程度の高速化は可能である．しかし，CSX600 の性能を引き出すには，行列乗算 $C = AB$ において，A, B, C のすべてのサイズを 1000 以上に取る必要がある．この場合，シフト数は 200 以上となるが，シフト数の増加に伴って行列乗算以外の演算の割合が増えるため，その部分がボトルネックとなる．

この問題を解決するため，我々は行列乗算以外の部分について最適化を行う．具体的には，「対角ブロック内部でのバルジ追跡」と「分離した小行列の固有値計算」と呼ぶ演算量の多い部分を取り上げ，前者に対して再帰的アルゴリズムへの変換，後者に対してブロック化を適用する．これにより，両部分の実行時間が大幅に減少し，行列乗算の占める割合が増加するため，CSX600 による高速化の効果が向上する．これらの最適化により，12,000×12,000 の複素ヘッセンベルグ行列の固有値とシューア標準形を求める場合，CSX600 を使うことで 3.2GHz Xeon に比べて 3.8 倍の高速化を達成できた．

以下では，2 節で CSX600 プロセッサとその性能について紹介する．3 節で small-bulge マルチシフト QR 法について述べ，CSX600 向けの最適化手法を提案する．4 節で数値実験の結果を述べ，5 節でまとめを行う．

2. CSX600 の特徴と性能

2.1 CSX600 プロセッサ

ClearSpeed 社の CSX600[5] は，数値計算専用に開発された浮動小数点コプロセッサである．チップ上に 1 個の主プロセッサと 96 個の計算専用プロセッサ，キャッシュメモリ，メモリコントローラ等が集積されており，250MHz で動作する．各プロセッサの仕様を表 1 に示す．チップの理論ピーク性能は倍精度演算で 48GFLOPS である．

表 1　CSX600 の仕様
Table 1　Specification of the CSX600.

種別	個数	仕様
主プロセッサ	1	64 ビット 64 バイト レジスタファイル 8K バイト 命令キャッシュ 4K バイト データキャッシュ
計算専用プロセッサ	96	64 ビット 2 演算同時実行可能 128 バイト レジスタファイル 6K バイト SRAM IEEE754 準拠 浮動小数点演算 0.5GFLOPS（倍精度）

CSX600 を PC 上で利用するため，ClearSpeed Advance というボードが製品化されている．本ボードは 2 個の CSX600 チップと最大 4G バイトの DRAM を搭載し，PCI-X バスにより PC と接続される．ボードの消費電力は約 25W である．本研究ではこのボードを使用する．

2.2 ソフトウェア開発環境

CSX600 の利用法としては，専用言語 C^n を用いてプログラム全体を移植する方法と，ベンダー提供のライブラリを用いてプログラムの一部のみを CSX600 で加速する方法とがある．本研究では後者の方法を採用し，BLAS（Basic Linear Algebra Subprograms）[7] の機能を提供する CSXL ライブラリから，長方行列どうしの乗算を行う DGEMM ルーチンを用いる．このルーチンは，ホスト PC のプログラムからコールする形で使うことができ，PC の主記憶から ClearSpeed Advance ボード上へのデータ転送，ボード上での CSX600 による演算，ボードから PC の主記憶へのデータの再転送を行う．

2.3 行列乗算の性能

本項では，CSXL の DGEMM ルーチンの基礎的な性能を報告する．DGEMM では，$M \times K$ 行列 A, $K \times N$ 行列 B, $M \times N$ 行列 C, スカラー α, β に対して $C := \alpha AB + \beta C$ を計算する[7]．Small-bulge マルチシフト QR 法では，次節で述べるように，正方行列と長方行列の積を計算するのに DGEMM を用いる．そのため，$M = K$ あるいは $N = K$ である．$M = K$, $N = K$ の場合の性能をそれぞれ図 1，図 2

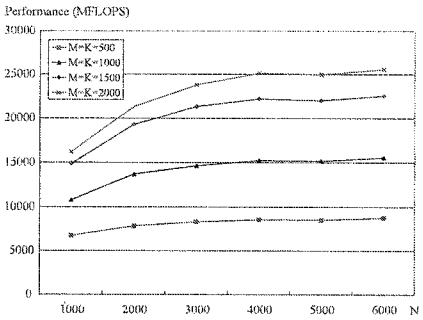

図 1 $M = K$ の場合の CSXL の DGEMM の性能
Fig. 1 Performance of CSXL DGEMM ($M = K$).

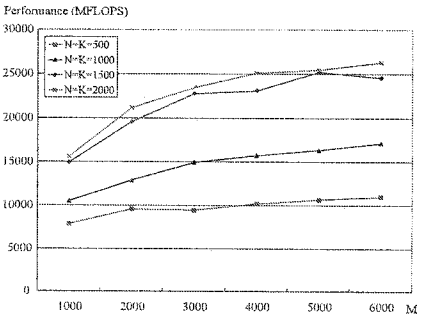

図 2 $N = K$ の場合の CSXL の DGEMM の性能
Fig. 2 Performance of CSXL DGEMM ($N = K$).

に示す．ここで，計算機環境は 4 節で述べる通りである．また，実際の使用条件に従い，$M = K$ の場合は A, B とも非転置，$N = K$ の場合は B のみ転置とした．図より，以下のことが言える．

- 行列乗算の性能は，長方行列の長い辺のサイズが 5,000 を超えるとほぼ飽和する．
- 正方行列のサイズが 500 の場合，CSXL の性能は 10GFLOPS 程度であり，標準的なプロセッサの性能（3.2GHz Xeon では 6.4GFLOPS）と大きな差はない．十分な高速化の効果を得るには，正方行列のサイズを少なくとも 1,000 程度にすることが必要である．

次節では，これらの結果に基づき，small-bulge マルチシフト QR 法の最適化を行う．

3. Small-bulge マルチシフト QR 法と CSX600 向け最適化

3.1 Small-bulge マルチシフト QR 法

Small-bulge マルチシフト QR 法[4] は，標準的なダブルシフト QR 法[6)10)] の自然な拡張である．いま，

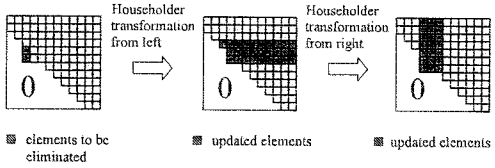

図 3 ハウスホルダー変換によるバルジ追跡
Fig. 3 Bulge chasing by Householder transformations.

$A = A_0$ を $n \times n$ の複素ヘッセンベルグ行列とし，第 l 回目の QR 反復後の行列を A_l とする．Small-bulge マルチシフト QR 法では，A_l から A_{l+m} を計算するため，まず A_l の右下隅の $m \times m$ 行列の固有値 $\sigma_1, \sigma_2, \ldots, \sigma_m$ を求め，次のように QR 分解と行列 Q によるユニタリ変換を $m/2$ 回繰り返す．

$$
\begin{aligned}
&(A_l - \sigma_2 I)(A_l - \sigma_1 I) = Q_l R_l, \\
&A_{l+2} = Q_l^{-1} A_l Q_l, \\
&\qquad \vdots \\
&(A_{l+m-2} - \sigma_m I)(A_{l+m-2} - \sigma_{m-1} I) \\
&\quad = Q_{l+m-2} R_{l+m-2}, \\
&A_{l+m} = Q_{l+m-2}^{-1} A_{l+m-2} Q_{l+m-2}. \qquad (1)
\end{aligned}
$$

ただし，それぞれの QR 分解とユニタリ変換は陽的に行うのではなく，ダブルシフト QR 法と同様，ハウスホルダー変換によるバルジ追跡[6)10)] の形で陰的に行う（図 3）．大きさ 4×4 のバルジ $m/2$ 個をそれぞれ行列の左上隅から右下隅まで追跡することにより，A_l から A_{l+m} の計算が完了する．

反復が進むにつれて A_l の副対角要素は減少し，ある時点で第 $(n - \bar{m} + 1, n - \bar{m})$ 要素が 0 と見なせるようになる．ここで，$\bar{m}$ は通常，m に近い整数である．この時点で A_l の右下隅の $\bar{m} \times \bar{m}$ の小行列を分離し，これに対してダブルシフト QR 法を行って固有値とシューア標準形を求める．このとき，ダブルシフト QR 法のバルジ追跡で使うハウスホルダー変換は，右下 $\bar{m} \times \bar{m}$ 小行列のみでなく，A_l の最後の $\bar{m}$ 列全体に作用させる必要がある．これらの処理をまとめて「分離した小行列の固有値計算」と呼ぶ．これが終了したら，残りの行列に対して再び small-bulge マルチシフト QR 法を行う．

3.2 行列乗算の利用

Small-bulge マルチシフト QR 法における中心演算は，$m/2$ 個のバルジをそれぞれ行列の左上から右下まで追跡する演算である．バルジどうしは，更新する領域が重ならないよう，互いに 3 行ずつ離す必要があるため[4]，$m/2$ 個のバルジは $(m/2) * 3 + 1$ 行を占める．いま，これらのバルジをそれぞれ k 行ずつ追跡するこ

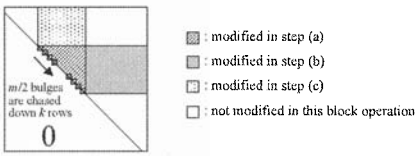

図 4 $m/2$ 個のバルジをそれぞれ k 行追跡することにより影響を受ける領域

Fig. 4 Regions affected by chasing $m/2$ bulges each by k rows.

とを考えると，これにより影響を受ける領域は，図 4 における色付きの $(m/2)*3+k$ 行，$(m/2)*3+k$ 列の領域である．これを，対角ブロック（斜線部），非対角行ブロック（灰色の部分），非対角列ブロック（点々の部分）に分ける．すると，バルジ追跡によるハウスホルダー変換は対角ブロック内での演算のみから定まるから，行列全体の更新を次の 3 つのステップに分けて実行できることがわかる[4]．

(a) 対角ブロック内部で，$(m/2)*3$ 個のバルジをそれぞれ k 行追跡する．また，そこで使ったハウスホルダー変換 $(m/2)*k$ 個を，$((m/2)*3+k)\times((m/2)*3+k)$ のユニタリ行列 U に蓄積する．

(b) 非対角行ブロックに左から U をかける．

(c) 非対角列ブロックに右から U^* をかける．

以下，ステップ (a) を「対角ブロック内部でのバルジ追跡」と呼ぶ．ステップ (b)，(c) は，明らかに正方行列と長方行列の乗算として実行できる．したがって，もし $(m/2)*3+k \ll n$ ならば，small-bulge マルチシフト QR 法は，演算のほとんどを行列乗算として実行できる．なお，Braman らは，演算量最小化のためには $k=(m/2)*3$ とすることが最適であり，このとき本アルゴリズムの演算量は通常のマルチシフト QR 法に比べて約 2 倍であることを示している．行列乗算はバルジ追跡演算と比べ，様々なアーキテクチャで数倍以上高速に実行できるため，多くの場合，演算量の増加があっても，small-bulge マルチシフト QR 法はダブルシフト QR 法や large-bulge マルチシフト QR 法[3] などの従来手法と比べて高速になる．

3.3 CSX600 の利用における問題点

本研究では，small-bulge マルチシフト QR 法を CSX600 により高速化するため，上記のステップ (b)，(c) を CSXL で実行する．これらは複素数の行列乗算であるが，本研究で用いた CSXL ver. 2.23 では実数用の DGEMM しか用意されていないため，実部と虚部を別の要素に格納して DGEMM を利用する．このため，正方行列のサイズは縦横 2 倍となり，Braman らにしたがって $k=(m/2)*3$ とすると，U のサイズ

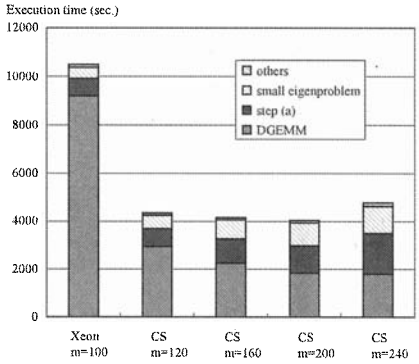

図 5 m を変えた場合の各部分の実行時間（$n=6000$）

Fig. 5 Execution time of each part as a function of m ($n=6000$).

は実行列で $6m\times 6m$ となる．2.3 項の結果から，正方行列のサイズは 1000 以上とすることが望ましく，$m\geq 166$ とする必要がある．

しかし，m を大きくするにつれ，行列乗算以外の部分の演算量が増大する．特に，3.1 で述べた「分離した小行列の固有値計算」と，ステップ (a) の「対角ブロック内部でのバルジ追跡」は，両方とも全体で $O(mn^2)$ の演算量を占める．さらに，これらの部分はデータ参照の局所性が低く，演算性能が低いため，ボトルネックになりやすい．

例として，$n=6000$ の場合の実行時間を図 5 に示す．グラフは左より，CSX600 なしの場合，CSX600 を使って m を増やしていった場合の性能を示す．m が増えるにつれて DGEMM の時間は減少するが，それ以外の部分が増大し，$m=200$ のときには，DGEMM 以外が 50%を占める．CSX600 の性能を引き出すには，これらの部分を高速化することが必要である．

3.4 対角ブロック内部でのバルジ追跡の高速化

Small-bulge マルチシフト QR 法におけるステップ (a) では，数多くの小さなハウスホルダー変換を対角ブロックおよび行列 U に作用させる．このため，データ参照の局所性が低く，演算性能が出にくい．そこで本研究では，この部分を再帰的アルゴリズムとして定式化し直す．

いま，q を m と k の公約数とし，$m'=m/q$，$k'=k/q$ とする．そして，$m/2$ 個のバルジを m' 個ずつ q 個の群に分ける．また，バルジの追跡も k' 行ずつの追跡 q 回に分ける．いま，第 i 番目のバルジ群の第 j 回目の追跡（$1\leq i\leq q, 1\leq j\leq q$）を考えると，これらは対角ブロックの $3*(m'/2)+k'$ 本の行と列，および U の $3*(m'/2)+k'$ 本の行に作用する．この

作用を直接行うのではなく，図4と同様にして対角ブロック内の作用を受ける領域を小対角ブロック（大きさ $(3*(m'/2)+k') \times (3*(m'/2)+k')$），小非対角行ブロック，小非対角列ブロックの3つに分け，次のように行うことにする．

(a') 小対角ブロック内部で，$(m'/2)*3$ 個のバルジをそれぞれ k' 行追跡する．また，そこで使ったハウスホルダー変換 $(m'/2)*k'$ 個を，$((m'/2)*3+k') \times ((m'/2)*3+k')$ のユニタリ行列 ${U'}_{ij}$ に蓄積する．

(b') 小非対角行ブロックに左から ${U'}_{ij}$ をかける．

(c') 小非対角列ブロックに右から ${U'^*}_{ij}$ をかける．

(d') U の該当する行ブロックに左から ${U'}_{ij}$ をかける．

これにより，ステップ (a) のうち，上記のステップ (a') 以外の部分は行列乗算により実行可能となる．ただし，行列乗算はサイズが小さくなるため，CPU で実行する．この変形によりステップ (a) の演算量は約2倍に増加するが，CPU においても行列乗算は高速に実行できるため，実行時間は短縮できると考えられる．

このアルゴリズムにおいて，ステップ (a') での演算はステップ (a) での演算と相似のため，同様の変形を再帰的に適用し，ステップ (a') をさらに行列乗算を用いて行うようにできる．これにより，このアルゴリズムは自然に再帰的アルゴリズムとして解釈できる．最適な再帰の段数と q の値は，CPU のアーキテクチャと m，k により定まる．

3.5 分離した小行列の固有値計算の高速化

3.1 項で述べた通り，「分離した小行列の固有値計算」のステップにおいては，ダブルシフト QR 法におけるハウスホルダー変換を A_l の最後の $\bar{m}$ 列全体に作用させる必要がある．そこで，この部分でもまず右下隅の $\bar{m} \times \bar{m}$ 小行列に対してダブルシフト QR 法を適用して固有値とシューア標準形を求め，そのとき使ったハウスホルダー変換をすべて $\bar{m} \times \bar{m}$ の行列 U に蓄積する．その後，最後の $\bar{m}$ 列の非対角ブロックに右から U^* をかける．これにより，非対角ブロックへの作用を行列乗算で行うことができ，性能が向上する．

4. 性能評価

4.1 評価環境

前節の検討結果に基づいて改良した small-bulge マルチシフト QR 法のプログラムを作成し，性能評価を行った．言語は FORTRAN である．使用した計算機環境は表2の通りである．

評価では，実部と虚部がそれぞれ $[0,1]$ の一様乱数に従う $m \times n$ 行列を生成し，それを LAPACK[1] の ZGEHRD によりヘッセンベルグ形に変換して small-bulge マルチシフト QR 法の入力として用いた．

表2 実験に使用した計算機環境
Table 2 Computational environments used in the experiments.

項目	条件
CPU	Xeon 3.2GHz
メモリ	8GB
OS	Red Hat Enterprise Linux
コンパイラ	Intel Fortran Ver. 9.1
BLAS	Intel Math Kernel Library Ver. 8.1
コプロセッサ	ClearSpeed Advance ボード × 1 （CSX600 チップ 2 個搭載）

4.2 改良の効果

まず，前節で述べた改良の CSX600 上での効果を評価した．そのため，$n = 3000$，6000 とし，改良を行わない場合と行った場合の実行時間を比較した．ただし，$n = 3000$ のときは $m = 160$，$n = 6000$ のときは $m = 200$ とした．また，改良版での q の値は，$m' = m/q = 40$ となるように定めた．CSX600 は行列乗算部分のみで利用した．CSXL は行列乗算の M，N，K すべてが 480 以上でないと動作しないため，ステップ (b)，(c) および 3.5 項で述べた行列乗算の部分のみを CSXL で実行し，ステップ (b')，(c')，(d') などそれ以外の部分は CPU で実行した．

実行時間の結果を図6に示す．改良により，「分離した小行列の固有値計算」，「対角ブロック内部でのバルジ追跡」の両方の時間が減少していることがわかる．特に，前者は劇的に減っているが，これはハウスホルダー変換を行列 U にまとめて非対角ブロック作用させることで，演算量の削減もできたためだと考えられる．後者は前者に比べると効果が大きくないが，$n = 6000$ の場合で実行時間が 2/3 に減少している．これより，提案した改良は効果があると言える．

4.3 Xeon 単体との性能比較

次に，改良版のアルゴリズムを用い，CSX600 あり／なしの場合の実行時間を比較した結果を図7に示す．CPU のみの場合は，$m = 100$，$q = 5$ とした．また，CSX600 を使った場合は，$n = 6000$ のとき $m = 200$，$q = 5$，$n = 12{,}000$ のとき $m = 240$，$q = 6$ とした．CSX600 の利用法は前項と同様である．図より，CSX600 を使うことで $n = 6000$，12,000 の場合にそれぞれ 3.5 倍，3.8 倍の高速化が得られていることがわかる．改良なしの場合，$n = 6000$ での高速化は 2.6 倍（図5参照）であるため，改良により CSX600 により性能を引き出すことが可能になったと言える．

なお，以上では高速化効果を測る基準として Xeon

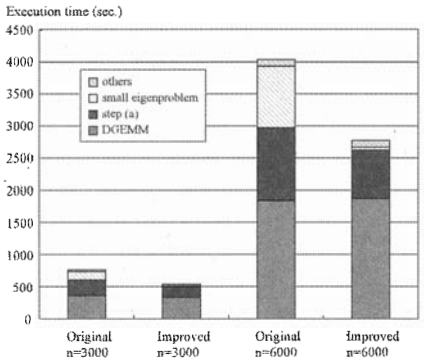

図 6 CSX600 上での改良の効果

Fig. 6 Effect of the improvements on the CSX600 processor.

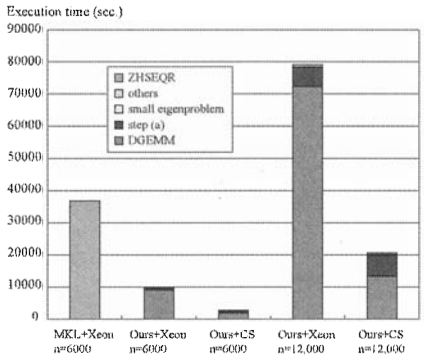

図 7 CSX600 あり／なしの場合の性能比較

Fig. 7 Comparison of execution times with and without the CSX600.

上での本研究のコードを用いたが，参考のため，標準的な QR 法ルーチンである LAPACK ZHSEQR の性能も図 7 に示した．本研究のコードは，Xeon 上でも，ZHSEQR に比べて格段に高速であることがわかる．

5. おわりに

本論文では，浮動小数点コプロセッサ CSX600 による複素ヘッセンベルグ QR 法の高速化について述べた．本研究では，ベースとなるアルゴリズムとして small-bulge マルチシフト QR 法を用い，CSX600 を使う場合にボトルネックとなる 2 つの部分を同定して，それらを行列乗算を使う形に定式化し直すことで高速化を図った．性能評価の結果，$12,000 \times 12,000$ の複素ヘッセンベルグ行列に対し，CSX600 を使うことで 3.2GHz Xeon の 3.8 倍の高速化を達成できた．

発表では，m，k，q などの性能パラメータの最適化や，本アルゴリズムに対する性能予測モデルについても報告する予定である．

謝辞 日頃からご指導頂いている名古屋大学大学院工学研究科の張紹良教授と，CSX600 の環境設定にご協力頂いた京都大学大学院理学研究科物理学・宇宙物理学専攻の畝山多加志氏に感謝いたします．なお，本研究は名古屋大学 21 世紀 COE プログラム「計算科学フロンティア」，科学研究費補助金基盤（C）一般（課題番号 18560058）および科学研究費補助金特定領域研究「i-explosion」（課題番号 19024018）の補助を受けている．

参 考 文 献

1) E. Anderson, Z. Bai, C. Bischof, J. Demmel, J.Dongarra, J.D. Croz, A.Greenbaum, S.Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide*. SIAM, 1992.
2) Z. Bai, D. Day, J. W. Demmel, and J. J. Dongarra. A test matrix collection for non-Hermitian eigenvalue problems (release 1.0). Technical Report CS-97-355, Department of Computer Science, University of Tennessee, Knoxville, 1997.
3) Z. Bai and J. Demmel. On a block implementation of Hessenberg QR iteration. *Int. J. of High Speed Computing*, 1:97–112, 1989.
4) K.Braman, R.Byers, and R.Mathias. The multishift QR algorithm. part I: Maintaining well-focused shifts and level 3 performance. *SIAM J. Matrix Anal. Appl.*, 23(4):929–947, 2002.
5) http://www.clearspeed.com/.
6) J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.
7) J. Dongarra, J. D. Croz, S. Hammarling, and R. J. Hanson. An extended set of FORTRAN basic linear algebra subprograms. *ACM Trans. Math. Software*, 14(1):1–17, 1988.
8) J. G. F. Francis. The QR transformation. a unitary analogue to the LR transformation. I. *Comput. J.*, 4:265–271, 1961/1962.
9) J. G. F. Francis. The QR transformation. II. *Comput. J.*, 4:332–345, 1961/1962.
10) G.H. Golub and C.F. van Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996.
11) GRAPE-DR Project. http://grape-dr.adm.s.u-tokyo.ac.jp/.
12) V.N. Kublanovskaya. On some algorithms for the solution of the complete eigenvalue problem. *U.S.S.R. Comput. Math. and Math. Phys.*, 3:637–657, 1961.