ブロック鏡映変換を用いた実対称行列のブロック三重対角化のOpenMPによる並列化の実験

村上 弘[1,a)]

概要：ブロック鏡映変換を用いた実対称行列のブロック三重対角形への変換処理の，OpenMP による並列化実装の計算機実験を行なった．計算の大部分はブロック小行列（タイル）間の行列積（BLAS3 の xGEMM）になるので，メモリバンド幅を抑えた演算密度の高い計算が行える．簡易な実装でも，たとえば普及品のマルチコア CPU を用いた PC 上での性能が，理論ピーク演算性能の 70%から 80%程度を達成できることが確かめられた．

キーワード：ブロックアルゴリズム，ブロック三重対角化，ブロック鏡映変換

Experiments of OpenMP parallelization of the block tridiagonalization of a real symmetric matrix by the block reflector method

Murakami Hiroshi[1,a)]

Abstract: We made experiments of OpenMP parallelization of the block tridiagonalization of a real symmetric matrix by the block reflector method. Since the most part of the calculation is reduced to matrix multiplications (BLAS3's xGEMM) of small matrices (tiles) of the block size, the calculation is rich in arithmetic operations with low requirement of the memory bandwidth. By experiments, we examined that even by a simple implementation of the method, PCs with multi-cored single CPU for commodities attained about 70% to 80% of their theoretical peak performances for the calculations.

Keywords: blocked algorithm, tiled algorithm, block tridiagonalization, block reflector

1. はじめに

実対称の大規模な（数万元〜）密行列の固有値問題を解く場合，BLAS2 である普通の Householder 法は記憶走査量が多く，メモリバンド幅が計算のボトルネックになる．そこで通常の行列算法での行列要素を通常の数（スカラ）から小行列（タイル）に拡張する算法のブロック化を行うことで，計算の粒度を大きくし，さらに小行列（タイル）同士の積を BLAS3 を用いて行なうと，計算のボトルネックになりがちな要求メモリバンド幅を低減できる．

実対称行列に対して，通常の鏡映変換に自然なブロック化拡張を施して得られるブロック鏡映変換を用いると，通常の Householder 三重対角化法の自然な拡張となるブロック Householder 三重対角化法が得られる．

いま実対称行列の次数を N とするとき，Householder 三重対角化法の場合には，小行列のサイズが b のブロック化では，記憶の走査回数が b に反比例して減るが，演算量の主要項は $(4/3)N^3$ で変わらないので b に比例して演算密度が向上する．

1 首都大学東京 数理情報科学専攻
Department of Mathematics and Information Sciences, Tokyo Metropolitan University
a) mrkmhrsh@tmu.ac.jp

ⓒ 2014 Information Processing Society of Japan

ブロック三重対角化を経由した対称密行列の固有値問題の解法は，通常の三重対角化を経由する方法に類似したものとなり，以下のようになる．まず，ブロック鏡映変換を繰り返し適用することで対称密行列 A を，ブロック三重対角行列 T に変換する．このとき元の A と変換後の T とは固有値の全体が一致する．そうして T の固有対で必要なものを求める．T の固有ベクトルの組に対して，A をブロック三重対角化する際に用いたブロック鏡映変換を逆順に繰り返して作用させると，元の行列 A の固有ベクトルの組が得られる．

全体行列 A はサイズ b の小行列（タイル）に区分けして扱い，タイル内部と対応する記憶領域はメモリ上で連続的になるようにする．A の次数 N がブロックサイズ b の倍数でないときは，行と列の各方向の末尾には行あるいは列の個数が b 未満の長方形のタイルも生じる．

今回の実験で用いたブロック三重対角化法の算法は，論文 [5] の中で既に記述した対称密行列をブロック三重対角行列に変換する操作の中で，ブロック鏡映変換を構成する際に用いる QR 分解を通常の HQR 法から T-S 型行列用の再帰的な処理による方法（文献 [6], [7], [9]）に置き換える改良を加えたものである．また性能測定には，2006 年に論文が書かれた当時よりも演算性能とメモリ容量が共に大幅に向上した最近の計算機（PC）を用いた．

（過去の論文 [5] では，1) 元の対称密行列をブロック鏡映変換の繰り返しでブロック三重対角化する．2) 得られた（帯行列でもある）対称三重ブロック対角行列を，帯内が密の対称帯行列に変換して帯幅を半減させる．3) 村田の帯ハウスホルダ法を用いてその密な帯行列を真の対称三重対角行列へ変換する．4) 真の三重対角行列にスツルム二分法を適用して希望の範囲（固有値分布の端付近）にある比較的少数個の固有値を求める．5) 帯行列の比較的少数個の固有ベクトルだけを対応する各固有値をシフトとする逆反復法で求める．6) 帯内が密の帯行列の固有ベクトルを三重ブロック対角行列の固有ベクトルに逆変換する．7) 最初のブロック三重対角化で用いたブロック鏡映変換を三重ブロック対角行列の各固有ベクトルに逆順に適用することで元の密行列の固有ベクトルを求める．という構成により対称密行列の少数個の固有対を求めるまでの方法と実行例を記述した．

このうち村田の帯ハウスホルダ法を用いて（帯内が密の）対称帯行列の比較的少数個の固有対を求める上記 3) から 5) までのステップの方法は文献 [1] に拠るもので，また同書に掲載されたプログラムを元に修正を加えて計算している．

文献 [2],[10] にも古典的アルゴリズムにより大規模な対称帯行列の固有対の一部を求める方法が記述されている．

大規模な対称行列のブロック三重対角化法には，この論文で紹介するブロック鏡映変換に基づくもの以外にも，通常のハウスホルダ変換の複数個を集積して作る WY 表現を用いてブロック化を実現している Bischof の方法などがある [3], [4], [8].）

2. ブロック三重対角化法

この節は，まず論文 [5] の復習であり，ブロック鏡映変換の構成には QR 分解と特異値分解を用いる．

2.1 ブロック鏡映変換の構成法

いま縦長の $m\times b$ 行列 U が $U^TU = I_b$ を満たせば，$H = I_m - 2UU^T$ は，$H^T = H$，$HH = I_m$ を満たすので，H は（U を軸とする）ブロック版の鏡映変換になる．

いま任意に与えられた縦長の $m\times b$ 行列 C に対して HC の先頭の b 行以外を全て消去するような U が決められれば，その U を利用してブロック三重対角化ができる．つまり $HC = E_b\beta$，ここで E_b は $m\times b$ の拡張単位行列で，β は b 次行列である．そのような U は以下の手順で構成できる（**図 1**，**図 2** を参照）．

(1) $m\times b$ 行列 C の QR 分解を $C \Rightarrow XR$ とする．
$m\times b$ 行列 X の先頭の b 行を集めた b 次行列を $\widehat{X}$ とする．

(2) $\widehat{X}$ の特異値分解を $\widehat{X} \Rightarrow WDV$ とする（この特異値はすべて 1 以下である）．

(3) X の先頭の b 行までが $\widehat{X}$ であったが，X のその位置に b 次行列 WV を加えた $m\times b$ 行列を Y とする．すなわち $Y \Leftarrow X + E_b(WV)$．

(4) $G = V^T\{2(I_b+D)\}^{(-1/2)}$ とし，Y に G を右から乗じた $U \Leftarrow YG$ を作ると，U は $U^TU = I_b$ を満たす．この U が求めるべきブロック変換の軸になる．

(5) $H = I_m - 2UU^T$ を C に作用させると，最初の b 行以外は消去されて，$HC = E_b\beta$ となる．ここで b 次行列 $\beta = -(WV)R$ である．その構成から H は対称な直交行列である．

すると，上記のブロック鏡映変換を繰り返すことで，以下のように任意の実対称行列 A は実対称のブロック三重対角行列へ変換される．

$$H^{(n-2)}\cdots H^{(3)}H^{(2)}H^{(1)}AH^{(1)}H^{(2)}H^{(3)}\cdots H^{(n-2)}$$

$$= \begin{pmatrix} \alpha_1 & \beta_1^T & 0 & \cdots & & & & \\ \beta_1 & \alpha_2 & \beta_2^T & 0 & \cdots & & & \\ 0 & \beta_2 & \alpha_3 & \beta_3^T & 0 & & & \\ \vdots & 0 & \beta_3 & \alpha_4 & \beta_4^T & & & \\ & \vdots & & & & & & \\ & & 0 & \ddots & \ddots & & & \vdots \\ & & & & & \ddots & \ddots & 0 \\ & & & & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1}^T \\ & & & & \cdots & 0 & \beta_{n-1} & \alpha_n \end{pmatrix}$$

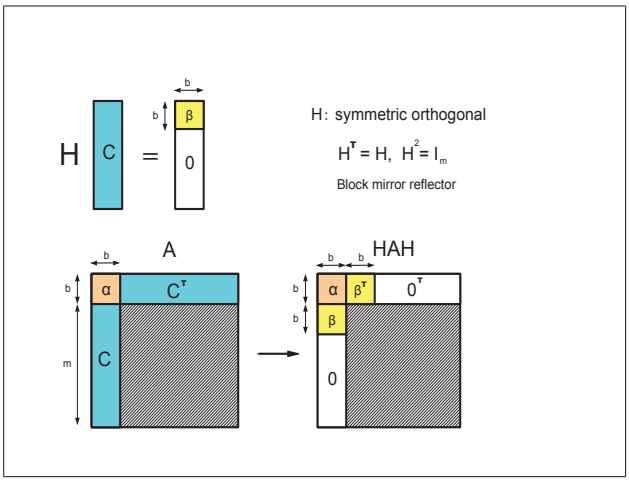

図 1 ブロック両側変換について

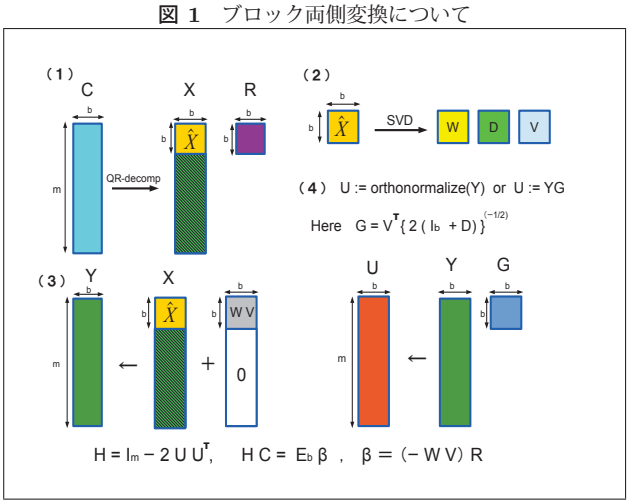

図 2 ブロック鏡映変換の構成法

2.2 HAH の実際の計算方法

先頭のブロック列については変換結果を計算して求める必要はなくて，先頭のブロック列の最初のブロックを α として保持する，そうしてブロック鏡映軸 U を作るときに得られた β も保存する．あとで逆変換に用いるために U を C の部分に重ね書きして保持する．

いま，A が 1 ブロック分だけサイズが小さくなったものを $\widetilde{A}$ で表せば，A のその部分を $A \leftarrow HAH$ と更新する方法は，補助のブロックベクトル P と対称な小行列 γ を使って以下のように書ける；

$$\begin{cases} P \leftarrow \widetilde{A}U, \\ \gamma \leftarrow P^T U, \\ P \leftarrow 2(U\gamma - P), \\ \widetilde{A} \leftarrow \widetilde{A} + UP^T + PU^T. \end{cases}$$

この計算はサイズ b の小行列（タイル）同士の行列積が豊富なので，BLAS3 で実装すれば計算は高速で記憶参照の局所性も高い．ブロック行列 A はその対称性から下半分だけをブロック列ベクトルが並んだ形で保持する．

各段の両側鏡映変換に於いて，ブロック行列 A の小行列は（並列化をしない場合には）列方向に連続的にアクセス

リスト 1 ブロック対称 Householder 三重対角化の擬似コード

```
! NB は行あるいは列のブロック数で，IB，JB などはブロックの添字．
! 行列やベクトルのブロック添字は 1 から NB の範囲をとる．
! 鏡映変換の段数 KB のループの内側には 2 つの計算カーネルがある
DO  KB = 1, NB-2
  「A[KB+1:NB,KB] からブロック鏡映軸 U[KB+1:NB] を作り重ね書きし，
    同時にできる BETA[KB] を格納．ALP[KB]（=A[KB,KB]）も格納．」
  !------------- カーネル 1（ブロック版の行列ベクトル積）
  ! P ← A * U
  P[KB+1:NB] = 0.0
  DO  JB = KB+1, NB
    P[JB] = P[JB] + A[JB,JB] * U[JB]
    DO  IB = JB+1, NB
      P[IB] = P[IB] + A[IB,JB] * U[JB]
      P[JB] = P[JB] + A[IB,JB] * U[IB]
    ENDDO
  ENDDO
  !-------------
  「ブロックベクトル U と P の内積である対称行列 γ を作り，
    P := 2（U γ − P）を γ の対称性も利用して作る．」
  !------------- カーネル 2（ブロック版の A の階数 2 の更新）
  ! A ← A + P * U^{T} + U * P^{T}
  DO  JB = KB+1, NB
    DO  IB = JB, N
      A[IB,JB] = A[IB,JB] + P[IB] * U[JB] + U[IB] * P[JB]
    ENDDO
  ENDDO
  !-------------
ENDDO
「ALP[NB-1]（=A[NB-1,NB-1]），BET[NB-1]（=A[NB,NB-1]），
  ALP[NB]（=A[NB,NB]）を残った A の要素から回収して格納．」
```

される（下図参照）．

$$\begin{pmatrix} * & & & & \\ \downarrow & & & & \\ * & * & & & \\ \downarrow & \downarrow & & & \\ * & * & * & & \\ \downarrow & \downarrow & \downarrow & & \\ * & * & * & * & \\ \downarrow & \downarrow & \downarrow & \downarrow & \\ * & * & * & * & * \end{pmatrix}$$

2.3 固有ベクトルの逆変換 [BCKTRS]

この操作もまた通常の Householder 法と同様にできる．

- 必要な T の固有ベクトルを求めて，それらを逆変換により必要な A の固有ベクトルに戻す．
- T の固有ベクトル $\mathbf{y}$ を A の固有ベクトル $\mathbf{x}$ へ逆変換するには，$\mathbf{y}$ にブロック鏡映軸 $U^{(i)}$ のブロック鏡映変換 $H^{(i)}$ を逆順に作用させる：$\mathbf{x} \leftarrow H^{(1)} H^{(2)} \cdots H^{(n-2)} \mathbf{y}$.
- 効率向上のためには，固有ベクトルを複数個まとめて変換する．ベクトルの組 Y に対するブロック化鏡映変換 $H^{(i)}$ の適用は，$Y \leftarrow H^{(i)} Y$ の形にまとめられ，行列-行列積が豊富でこれにも BLAS3 が使える．T の固有ベクトルの組を対応する A の固有ベクトルの組にブ

ⓒ 2014 Information Processing Society of Japan

ロック鏡映変換を逆順に適用して変換する．

3. T-S 型行列用の QR 分解法について

ブロック鏡映変換の構成には，T-S 型である $m \times b$ 行列の QR 分解が必要である．この QR 分解に記憶参照の局所性が無い BLAS2 である通常の HQR 法を用いると，メモリバンド幅と並列処理の同期が性能のボトルネックとなる．ブロック鏡映変換の軸である U を構成する手順の中で BLAS2 である通常の HQR 法を用いると，サイズ b が $\sqrt{m}$ 付近もしくはそれ以上になるときにブロック鏡映変換の処理全体の性能が低下する．その理由は，各段のブロック鏡映変換の中で，通常の HQR 法が費やす記憶走査量 $(3/2)mb^2$ が，ブロック鏡映変換の記憶走査量 $(1/2)m^2$ と同程度となるからである．そこで，ブロック鏡映変換の中で用いる T-S 型行列用の QR 分解には，（b がある程度大きければ特に）BLAS2 である通常の HQR 法に比べて記憶の走査量が少ない方法を用いるべきである．

今回の実験に用いた T-S 行列に対する QR 分解の方法は，既に文献 [9] に於いて詳細を記述したものであり，T-S 行列に対する Householder 直交変換全体の過程を 2 段階に分割している（多段階の算法も再帰的処理を行なえば自然に得られる）．この T-S 行列用の QR 法とその利点は以下のようになる：計算の大部分が各部分行列に対する粒度の大きく互いに独立に行える処理で構成されているので並列化に適していること，データ参照の局所性を向上させるので階層記憶システムに適し，また PE 間のデータ転送と同期の頻度が少ないことも並列処理に適している．

4. 計算機実験

以下では「換算性能」は，N 次の実対称行列をブロック三重対角化する演算量が $(4/3)N^3$ である，とみなして計算された演算速度のことであるとする．プログラムは Fortran90 言語で書いて，OpenMP により簡単に並列化をしている．xGEMM などの BLAS3 のルーチンには Intel MKL を用いた．より高い性能を引き出すには，用いる小行列の次数 b の値に対して最適に調整された専用の小規模行列の乗算ルーチンを作る方が良いであろう．

最近の CPU では内部回路の温度が低くて余裕がある場合にはクロックの周波数をシステムの基準値よりも上げて動作させ，温度が閾値を越えるとクロックの周波数を下げる機能（インテル製品では Turbo Boost と呼称）を有するものがあり，それを用いれば CPU の実行性能をさらに増大できる可能性があるが，その反面で外部の環境の温度やシステムの冷却能力などに依存して CPU の実行性能が変動することになり，他のマシンとの比較に於いて公平さや明瞭さに困難が生じる．そこで今回のすべての実験では，そのような機能は無効に設定することで，CPU が本来の標準クロック周波数を越えて動作しないようにして測定を行なった．

また，CPU の各コアについてコアあたり複数のスレッドをハードウェアレベルで自動的に実行できる機能（インテル製品では Hyper Thread と呼称し，コアあたり 2 スレッドまで）もまた，メモリの読み書き中心ではない浮動小数点演算中心の計算では有効化すると性能が落ちる場合があるので，今回の実験ではすべて無効に設定して測定を行なった．今回用いたシステムについては，この機能を有効にしても無効にしても（浮動小数点演算についての）CPU の理論ピーク演算性能は変わらない．

ブロック鏡映軸を構成するための QR 分解の方法として，以下の実験ではすべて T-S 型行列用の階層的な方法を，OpenMP による並列化も入れて用いている．

4.1 Intel Core2 Quad Q6600 のシステムの場合

シングル CPU のシステムで，CPU は Intel Core2 Quad Q6600（コア数：4，クロック：2.4GHz，L1(I/D)：32KB，共有 L2:8MB（4MB x 2 ダイ））．主記憶は 16Gbyte（デュアルチャンネル動作，DDR3-1600MHz(PC3-12800）の 4Gbyte の DIMM が 4 本；実際の動作は DDR3-1067MHz で，PC3-8500 であろう）．OS は Fedora 16 for x86_64 で，コンパイラと BLAS ライブラリは Intel Parallel Studio XE 2013 for 64bit Linux （Intel Fortran コンパイラ ver 14.0.0, MKL ライブラリ version 11.1）．並列化には Intel Fortran の OpenMP の機能のみを利用した．用いたコンパイラオプションは”-fast -openmp” である．Core 2-Q6600 は SSSE3 命令により，1 コアあたり CPU クロック毎に倍精度の演算が最大で 4 個実行できるので，CPU の理論ピーク演算性能は 38.4GFLOPS（倍精度演算，SSE3，SSSE3）である．

図 3 は行列サイズ N を 1 万から 6 万まで 1 万ずつ変えながら横軸にはブロックサイズ b をとり，縦軸には換算性能（GFLOPS）の値をとってプロットしたグラフである．**図 4** は，縦軸を変更して，理論ピーク演算性能に対する換算性能の比率をパーセントで表した値にしてプロットしたグラフである．**図 5**）は，ブロックサイズ b を 32，64，96，128，160，192 としたそれぞれの場合について，横軸に行列サイズ N をとり，縦軸に換算性能（GFLOPS）をとってプロットしたグラフである．すべての計算は OpenMP で並列化して，CPU のコア数と同じ 4 スレッドで実行した．

4.2 Intel Core i7-920 のシステムの場合

シングル CPU システムで，CPU は Intel Core i7-920（コア数：4，クロック：2.67GHz，L2：256KB，共有 L3：8MB，HT 機能と TB 機能はオフ）であり，主記憶は 48Gbyte（トリプルチャネル動作，DDR3-1333MHz（PC3-10600）の 8Gbyte の DIMM を 6 本．実際の動作は DDR3-1067MHz，PC3-8500）である．OS は CentOS 6.3 for x86_64 で，コンパイラと BLAS ライブラリは Intel Parallel Studio XE

ⓒ 2014 Information Processing Society of Japan

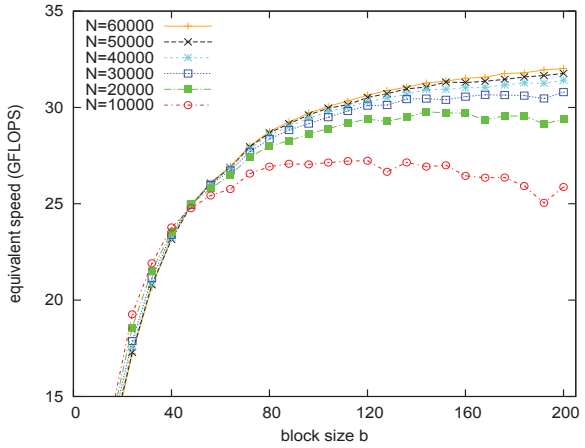

図 3 ブロック三重対角化の換算性能（Core2 Quad Q6600（4 スレッド））

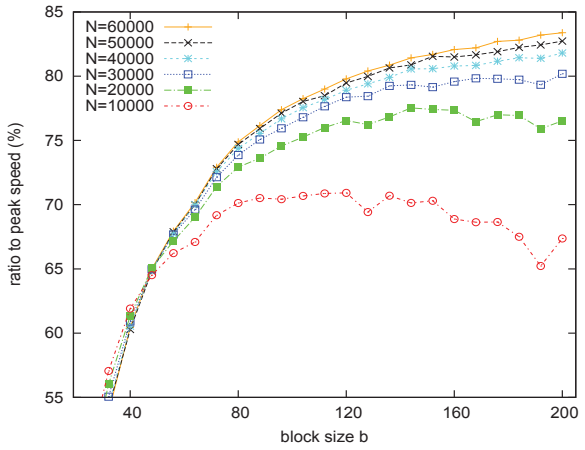

図 4 ピーク性能との比率（Core2 Quad Q6600（4 スレッド））

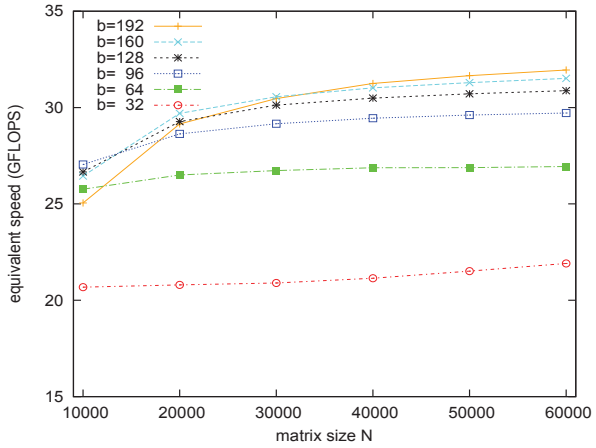

図 5 ブロック三重対角化の換算性能（Core2 Quad Q6600（4 スレッド））

2013 for 64bit Linux （Intel Fortran コンパイラ ver 13.0, MKL ライブラリ ver 11.0）．並列化には Intel Fortran の OpenMP の機能だけを利用した．用いたコンパイラオプションは”-fast -openmp” である．Core i7-920 は Nehalem アーキテクチャで SSE4.2 命令を持ち，1 コアあたり CPU クロック毎に倍精度の演算が最大 4 個実行できるので，TB 機能がオフの場合には CPU の理論ピーク演算性能は 42.72GFLOPS（倍精度演算，SSE4.2）である．

図 6 は行列サイズ N を 1 万から 6 万まで 1 万ずつ変えながら横軸にはブロックサイズ b を 20 から 200 まで 20 刻みでとり，縦軸には換算性能（GFLOPS）の値をとってプロットしたグラフである．**図 7** は，縦軸を変更して理論ピーク演算性能に対する換算性能の比率をパーセントで表した値にしてプロットしたグラフである．

図 8 はブロックサイズ b を 20 から 200 まで 20 ずつ増やしたそれぞれの場合について，横軸に行列サイズを 1 万から 6 万まで 1 万刻みでとり，横軸に換算性能（GFLOPS）をとってプロットしたグラフである．すべての計算は OpenMP を用いて並列化して，CPU のコア数と同じ 4 スレッドで実行した．

4.3 AMD Phenom II X6 1090T のシステムの場合

シングル CPU システムで，CPU は AMD Phenom II X6 1090T B.E. （コア数：6，クロック：3.2GHz，L2：512KB/core，共有 L3：6MB；Socket AM3）であり，主記憶は 32Gbyte（デュアルチャネル動作，DDR3-1333MHz（PC3-10600）CL9 の 8Gbyte DIMM が 4 本）．OS は Fedora 16 for x86_64 であり，コンパイラと BLAS ライブラリは Intel Parallel Studio XE 2013 for 64bit Linux （Intel Fortran コンパイラ ver 13.0，MKL ライブラリ ver 11.0）である．並列化には Intel Fortran の OpenMP の機能のみを利用し，用いたコンパイラオプションは”-fast -openmp” である．AMD Phenom II X6 では SSE4a 命令を持ち，1 コアあたり CPU クロック毎に倍精度の演算が最大 4 個実行できるので，Tubo Core Technology をオフの場合には CPU の理論ピーク演算性能は 76.8GFLOPS（倍精度演算，SSE3, SSE4a）である．

図 9 は行列サイズ N を 1 万から 7 万まで 1 万ずつ変えながら横軸にブロックサイズ b を 20 から 200 まで 20 刻みで，縦軸に換算性能の値をとってプロットしたグラフである．**図 10** は，縦軸を理論ピーク演算性能に対する換算性能の比率をパーセントで表した値に変更してプロットしたグラフである．**図 11**）はブロックサイズ b を 20 から 200 まで 20 ずつ増やしたそれぞれの場合について，横軸に行列サイズを 1 万から 7 万まで，縦軸に換算性能をとってプロットしたグラフである．すべての計算は OpenMP を用いて並列化して，CPU のコア数と同じ 6 スレッドで実行した．

ⓒ 2014 Information Processing Society of Japan

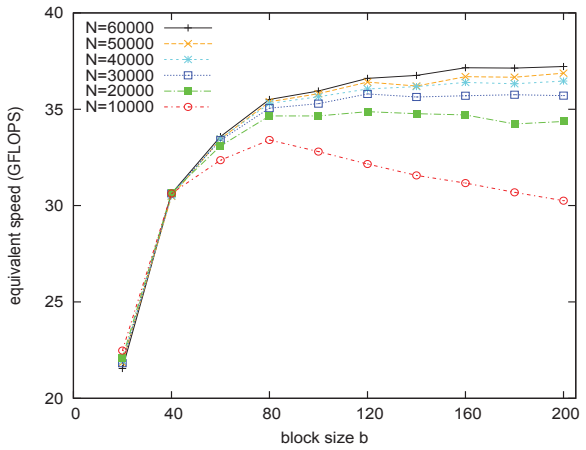

図 6 ブロック三重対角化の換算性能（Core i7-920（4 スレッド））

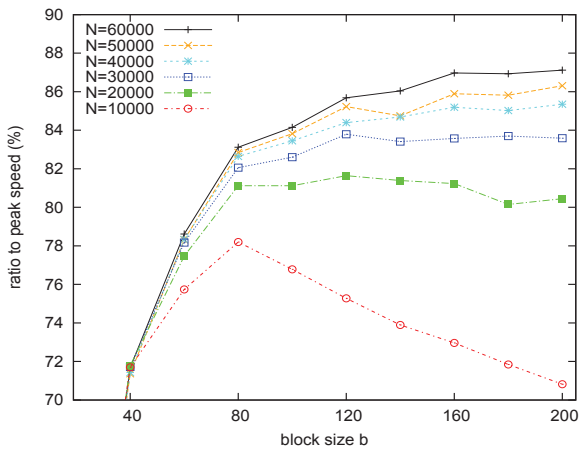

図 7 換算性能のピーク性能に対する比率（Core i7-920（4 スレッド））

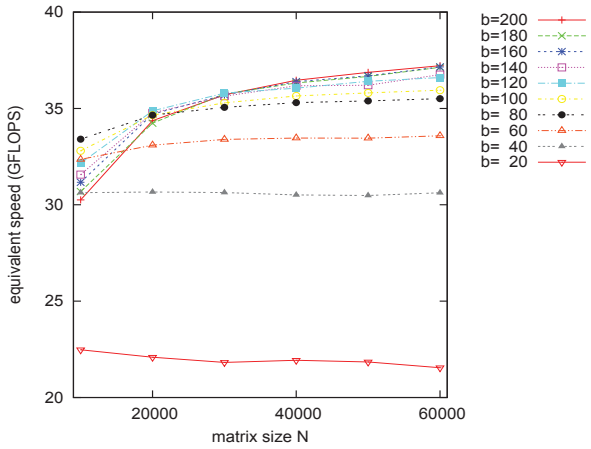

図 8 ブロック三重対角化の換算性能（Core i7-920（4 スレッド））

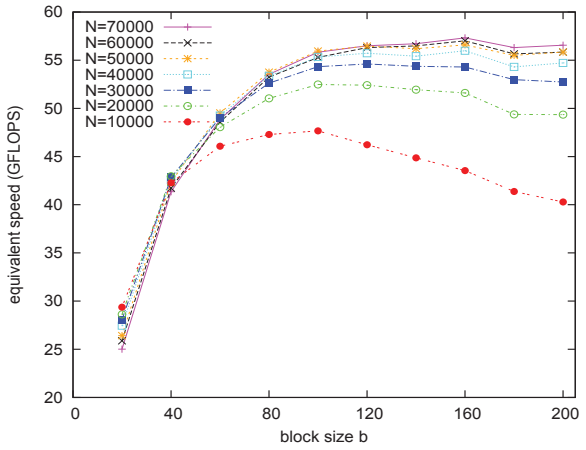

図 9 ブロック三重対角化の換算性能（Phenom II X6 1090T（6 スレッド））

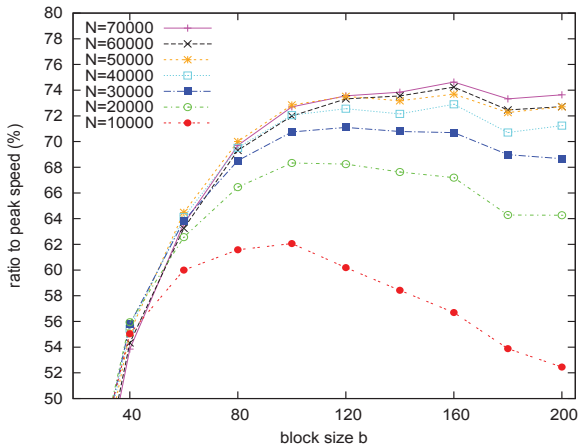

図 10 換算性能のピーク性能に対する比率（Phenom II X6 1090T（6 スレッド））

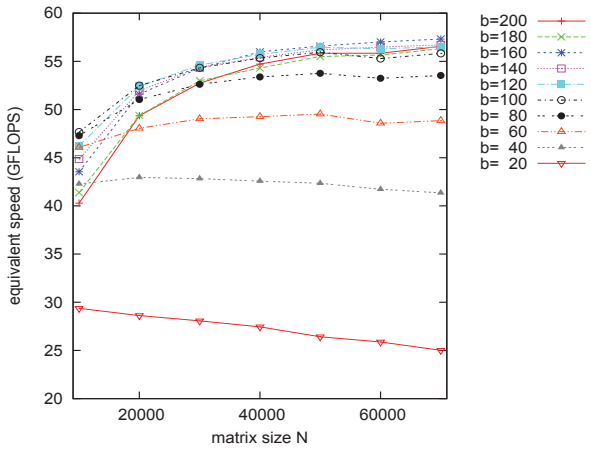

図 11 ブロック三重対角化の換算性能（Phenom II X6 1090T（6 スレッド））

ⓒ 2014 Information Processing Society of Japan

4.4 Intel Core i7-2600K のシステムの場合

シングル CPU システムで，CPU は Intel Core i7-2600K（コア数：4，クロック：3.4GHz，HT 機能と TB 機能はオフに設定，共有 L3：8MB）で，主記憶は 32Gbyte（デュアルチャンネル動作，DDR3-1600MHz（PC3-10600 動作），CL10 の 8Gbyte DIMM を 4 本）．OS は Fedora 16 for x86_64 で，コンパイラと BLAS ライブラリは Intel Parallel Studio XE 2013 for 64bit Linux （Intel Fortran コンパイラ ver 13.0, MKL ライブラリ ver 11.0）を用いた．用いたコンパイラオプションは"-fast -openmp" である．Core i7-2660K は Sandy Bridge アーキテクチャの AVX 命令を持ち，1 コアあたり CPU クロック毎に倍精度の演算を最大 8 個実行できるので，TB 機能がオフの場合には CPU の理論ピーク演算性能は 108.8GFLOPS（倍精度演算，AVX 命令）である．

図 12 は行列サイズ N が 1 万から 9 万までの 1 万刻みでそれぞれの場合について，横軸にブロックサイズ b の値を 20 から 20 刻みで 200 までとり，縦軸に換算性能（GFLOPS）の値をとってプロットしたグラフであり，**図 13** は縦軸を変更して，理論ピーク演算性能に対する換算性能の比率をパーセントで表した値にしてプロットしたグラフである．また**図 14** はブロックサイズ b の値を 20 から 20 刻みで 200 まで変えたそれぞれの場合について，横軸に行列 A の次数 N を 1 万から 9 万まで 1 万刻みでとり，縦軸に換算性能（GFLOPS）の値をとってプロットしたグラフである．すべての計算は OpenMP を用いて並列化し，CPU のコア数と同じ 4 スレッドで実行した．

4.5 Intel Core i7-3770K のシステムの場合

シングル CPU システムで，CPU は Intel Core i7-3770K（コア数：4，クロック：3.5GHz，HT 機能と TB 機能はオフ，L1(I/D)：32KB，L2：256KB，共有 L3：8MB）で，主記憶は 32Gbyte（デュアルチャネル，DDR3-1600MHz（PC3-12800 動作）の 8Gbyte DIMM を 4 本）．OS は CentOS 6.3 for x86_64 で，コンパイラと BLAS ライブラリは Intel Parallel Studio XE 2013 for 64bit Linux （Intel Fortran コンパイラ ver 13.0, MKL ライブラリ ver 11.0）．用いたコンパイラオプションは"-fast -openmp" である．Core i7-3770K は Ivy-Bridge アーキテクチャの AVX 命令を持ち，1 コアあたり CPU クロック毎に倍精度の演算を最大 8 個実行できるので，TB 機能がオフの場合には CPU の理論ピーク演算性能は 112.0GFLOPS（倍精度演算，AVX 命令）である．

図 15 は行列サイズ N を 1 万から 8 万まで 1 万ずつ変えながら横軸にブロックサイズ b を 20 から 200 まで 20 ずつ刻みで，縦軸に換算性能（GFLOPS）の値をとってプロットしたグラフである．**図 16** は，縦軸を理論ピーク演算性能に対する換算性能の比率をパーセントで表した値に変更してプロットしたグラフである．**図 17** は，ブロックサイズ b を 20 から 200 までの 20 ずつ増やした場合のそれぞれ

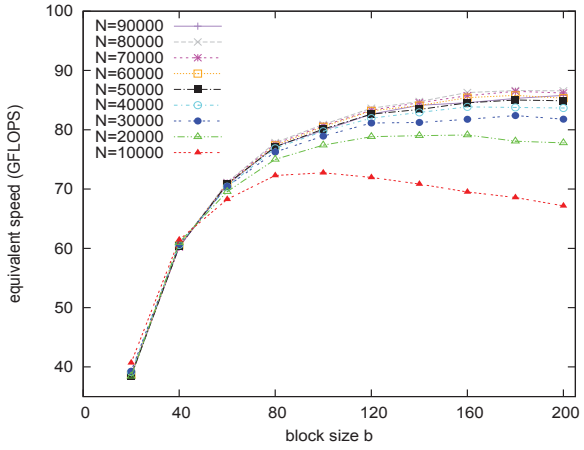

図 12 ブロック三重対角化の換算性能（Core i7-2600K（4 スレッド））

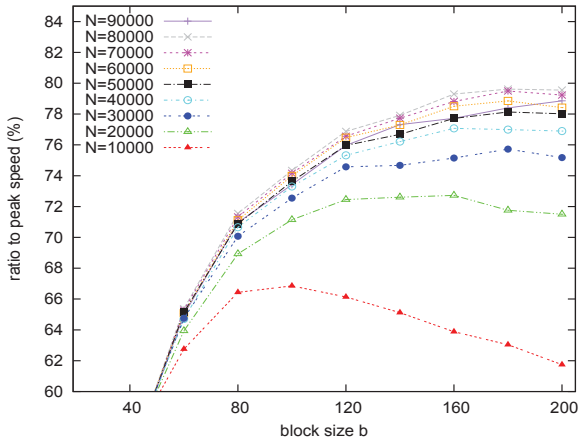

図 13 換算性能のピーク性能に対する比率（Core i7-2600K（4 スレッド））

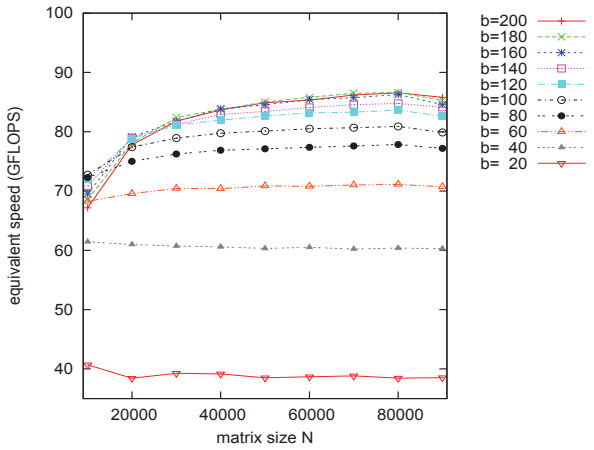

図 14 ブロック三重対角化の換算性能（Core i7-2600K（4 スレッド））

ⓒ 2014 Information Processing Society of Japan

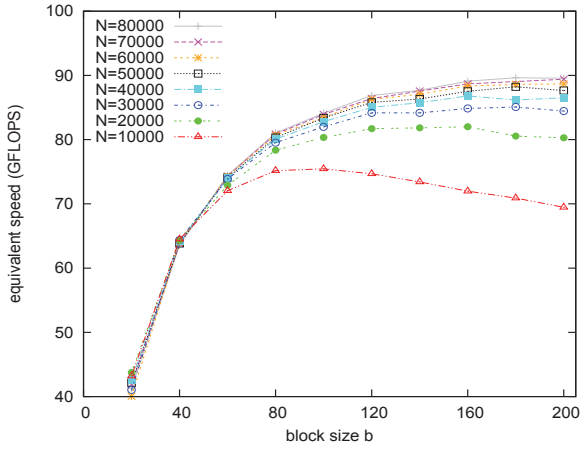

図 15 ブロック三重対角化の換算性能（Core i7-3770K（4 スレッド））

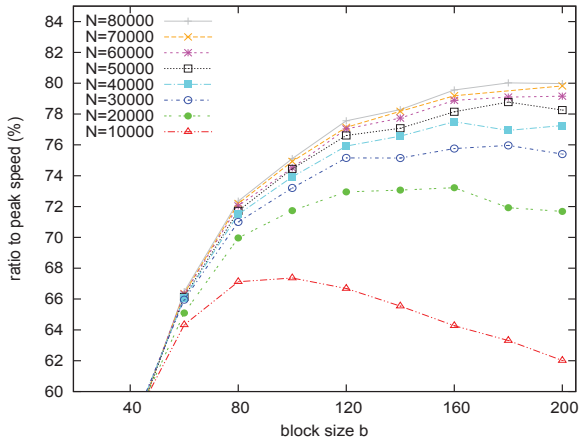

図 16 換算性能に対するピーク性能の比率（Core i7-3770K（4 スレッド））

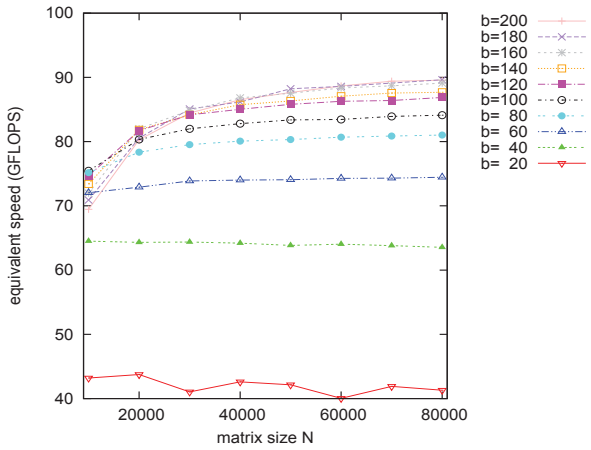

図 17 ブロック三重対角化の換算性能（Core i7-3770K（4 スレッド））

について，横軸に行列サイズを 1 万から 8 万まで 1 万刻みでとって，縦軸には換算性能（GFLOPS）をとってプロットしたグラフである．すべての計算は OpenMP を用いて並列化し，CPU のコア数と同じ 4 スレッドで実行した．

4.6 Intel Core i7-3930K のシステムの場合

シングル CPU システムで，CPU は Intel Core i7-3930K（コア数：6，クロック：3.2GHz，HT 機能と TB 機能はオフ，L1(I/D)：32KB，L2：256KB，共有 L3：12MB）で，主記憶は 64GB（クアドチャネル，DDR3-1600（PC3-12800），CL10 の 8Gbyte DIMM を 8 本）．OS は CentOS 6.3 for x86_64 で，コンパイラと BLAS ライブラリは Intel Parallel Studio XE 2013 for 64bit Linux （Intel Fortran コンパイラ ver 13.0，MKL ライブラリ 11.0）で，用いたコンパイラオプションは”-fast -openmp” である．Core i7-3930K では Ivy-Bridge アーキテクチャの AVX 命令により，1 コアあたり CPU クロック毎に倍精度の演算を最大で 8 個実行できるので，TB 機能がオフの場合には CPU の理論ピーク演算性能は 153.6FLOPS（倍精度演算，AVX 命令）である．

図 18 は行列サイズ N を 1 万から 9 万まで 1 万ずつ増やしたそれぞれの場合について，横軸にブロックサイズ b を 20 から 200 まで 20 ずつ増やしてとり，縦軸には得られた換算性能（GFLOPS）をとってプロットしたグラフであり，**図 19**）は縦軸を変更して，CPU の理論ピーク演算性能に対する換算性能の比率をパーセント値でプロットしたグラフである．**図 20**）ではブロックサイズ b を 20 から 200 まで 20 ずつ増やしたそれぞれの場合について，横軸には行列サイズ N を 1 万から 9 万まで 1 万刻みでとり，縦軸には換算性能（GFLOPS）をとってプロットしたグラフである．図 18 から図 20 はいずれも OpenMP により並列化して，CPU のコア数と等しい 6 スレッドで実行した結果である．

図 21 ではブロックサイズ b を 160 に固定して，行列サイズ N を 1 万から 5 万まで 1 万刻みで変えたそれぞれの場合に対して，横軸には並列計算に用いたスレッド数を 1 から 6 まで 1 ずつ増やしてとり，縦軸には得られた換算性能の値をとってプロットしたグラフである．このグラフから，この実験では換算性能が使用スレッド数にほぼ比例していることが分かる．

4.7 Intel Core i7-4770 のシステムの場合

シングル CPU システムで，CPU は Intel Core i7-4770（コア数：4，クロック：3.4GHz，HT 機能と TB 機能はオフ，L1(I/D)：32KB，L2：256KB，共有 L3：8MB）で，主記憶は 32Gbyte（デュアルチャネル，DDR3-1600MHz（PC3-12800 動作）の 8Gbyte DIMM を 4 本）．OS は CentOS 6.4 for x86_64 で，コンパイラと BLAS ライブラリは

ⓒ 2014 Information Processing Society of Japan

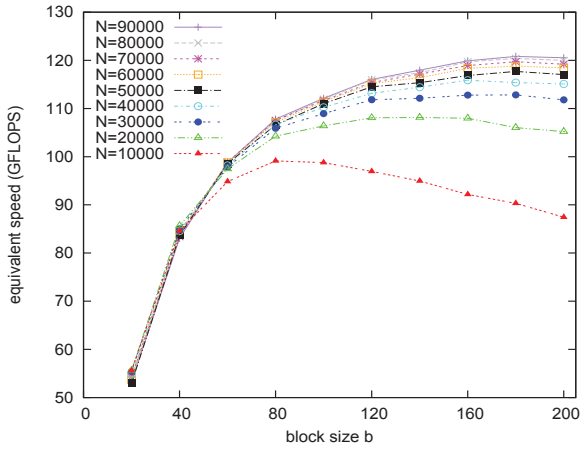

図 18 ブロック三重対角化の換算性能（Core i7-3930K（6 スレッド））

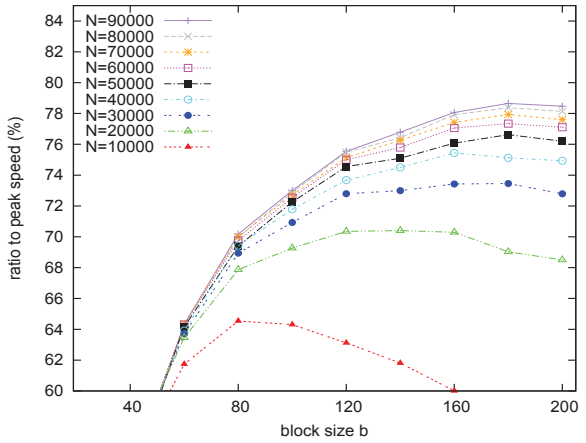

図 19 換算性能に対するピーク性能の比率（Core i7-3930K（6 スレッド））

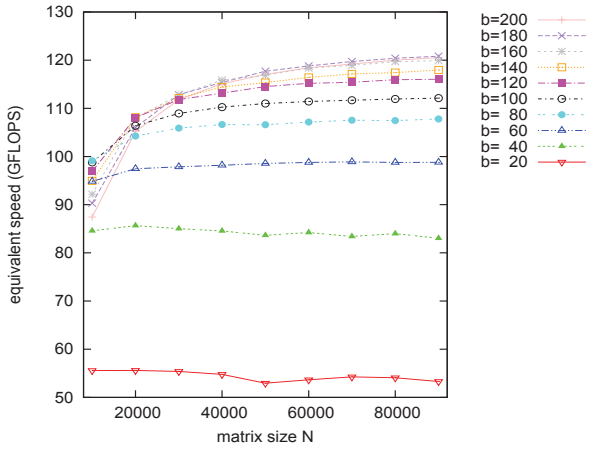

図 20 ブロック三重対角化の換算性能（Core i7-3930K（6 スレッド））

Intel Parallel Studio XE 2013 SP1 update1 for 64bit Linux（Intel Fortran コンパイラ ver 14.0.1, MKL ライブラリ ver 11.1(update1)）で，用いたコンパイラオプションは”-fast -align array256byte -openmp” である．Core i7-4770 では Haswell アーキテクチャーの AVX2 命令を持ち，1 コアあたり CPU クロック毎に倍精度演算を最大 16 個実行できるので，TB 機能がオフの場合には CPU の理論ピーク演算

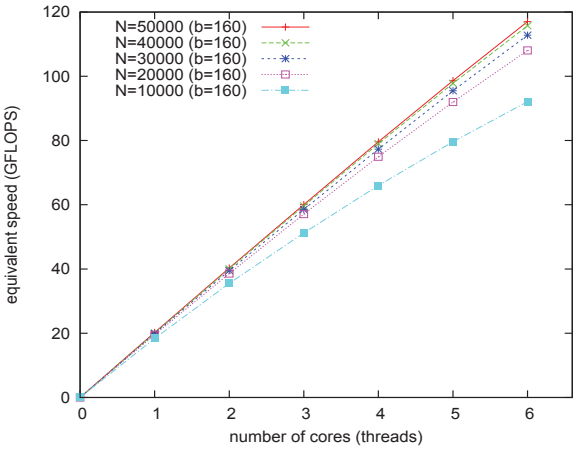

図 21 ブロック三重対角化の換算性能（Core i7-3930K）（使用スレッド数と性能の直線性，$b = 160$）

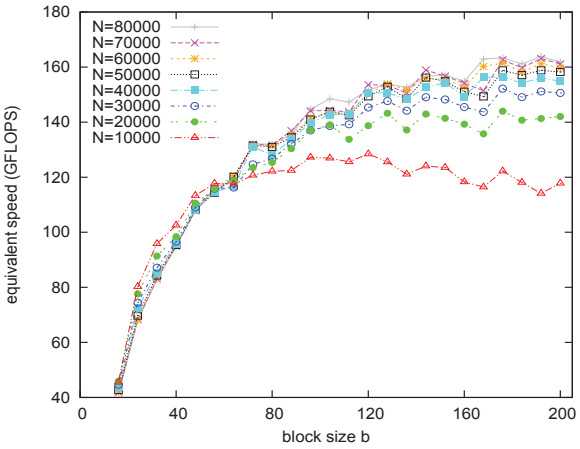

図 22 ブロック三重対角化の換算性能（Core i7-4770（4 スレッド））

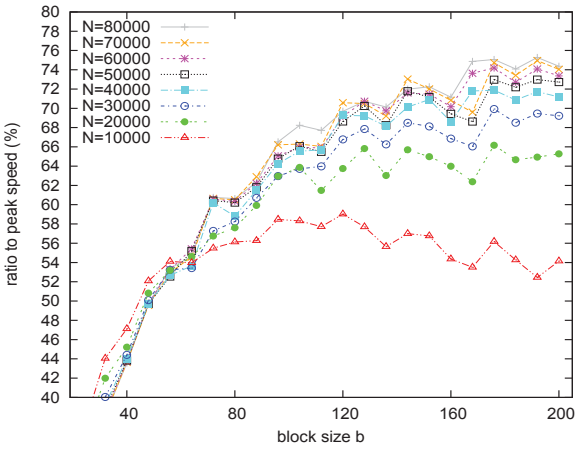

図 23 換算性能に対するピーク性能の比率（Core i7-4770（4 スレッド））

性能は 217.6GFLOPS（倍精度演算，AVX2 命令）である．

図 22 は行列サイズ N を 1 万から 8 万まで 1 万ずつ変えながら横軸にブロックサイズ b を 16 から 200 まで 8 ずつ刻みで，縦軸に換算性能（GFLOPS）の値をとってプロットしたグラフである．**図 23** は，縦軸を理論ピーク演算性

ⓒ 2014 Information Processing Society of Japan

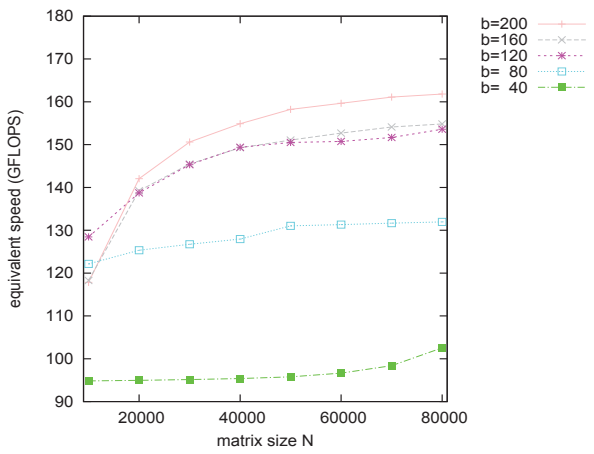

図 24 ブロック三重対角化の換算性能（Core i7-4770（4 スレッド））

能に対する換算性能の比率をパーセントで表した値に変更してプロットしたグラフである．**図 24** は，ブロックサイズ b を 40 から 200 までの 40 ずつ増やした場合のそれぞれについて，横軸に行列サイズを 1 万から 8 万まで 1 万刻みでとって，縦軸には換算性能（GFLOPS）をとってプロットしたグラフである．すべての計算は OpenMP を用いて並列化し，CPU のコア数と同じ 4 スレッドで実行した．

5. 今後に改良すべき点

現状の性能値にはまだ少し向上させる余地が残っている．たとえば，ブロック鏡映軸の構成に於いて T-S 行列の QR 分解に今回用いた算法は階層的ではあるが，内部は純粋な BLAS2 で組み立てられていて，列幅であるブロックサイズ b が大きくなると計算速度は低下する．それを防ぐには列方向もまたブロック分割する処理に置き換えるべきである．

今回のブロック三重対角化のカーネル部分（§2.2 のリスト 1）では，対称性から下三角部分だけを保持したブロック行列に対する鏡映変換 1 段階分の処理は，「ブロック版の行列ベクトル積」（カーネル 1）の箇所では列方向には逐次的とし，対象とする列を複数スレッドで行方向に分割して並列化を行なった．また，「ブロック版の階数 2 の更新」（カーネル 2）の計算では各スレッドがそれぞれ相異なる列を前から後に向かって一つずつ処理していく並列化を行なった．いずれも各スレッドに割り当てる作業量を完全に均等にはできなかったり，変換が進んだ終盤で行列のサイズが小さくなると作業が割り当てられないスレッドができる．特に元の行列のサイズが小さい場合や，計算に用いるスレッドの数が多いときは影響が大きくなる．この影響を減らすには，単純に行あるいは列に基づいてスレッドを割り当てるのではなくて，例えば表を用いて作業の割り当てを管理するなどにより，各スレッドに待ち状態ができるだけ発生しないようにする工夫などが必要であろう．

サイズが固定のタイル同士の行列積の計算には，xGEMM のような汎用ルーチンではなくて，そのサイズ専用にチューニングされたルーチンを用意して使う方が良いであろう．また BLAS3 についても BLAS2 の場合と同様にブロック版の階数 2 の更新 `A[IB,JB] = A[IB,JB] + P[IB] * U[JB] + U[IB] * P[JB]` に含まれているブロック積和 2 回をわけないでまとめて行なう専用ルーチンを作るのが良いのではないかと考えられる．

6. まとめ

今回の実験の結果から，ランダムな実対称密行列のブロック三重対角化について，ブロックサイズ b を適切に大きく選ぶことにより，行列の次数 N が大きい場合には，4 コアから 6 コア程度の小規模なマルチコア型の CPU を 1 個持つ計算機（PC）上で，CPU の理論ピーク演算性能値の 70%から 80%程度の「換算性能」が達成できることを確認した．

参考文献

[1] 村田 健朗，小国 力，唐木 幸比古: 「スーパーコンピュータ　科学技術計算への適用」, 丸善 (1985).

[2] 長谷川 秀彦:「古典的アルゴリズムによる大規模対称帯行列の固有値計算」, 計算工学講演会論文集, Vol.10,No.1 (2005), pp.247–248.

[3] 山本 有作:「キャッシュマシン向け対称帯行列固有値解法の性能・精度評価」, 情報処理学会論文誌, Vol.46, No.SIG3(ACS 8), (2005), pp.81–91.

[4] Yusaku Yamamoto: “Performance modeling and optimal block size selection for a BLAS-3 based tridiagonalization algorithm”, *proceeding of the eighth international conference on High-Performance Computing in Asia-Pacific Region*, (2005).

[5] Hiroshi Murakami: “An Implementation of the Block Householder Method”, 情報処理学会論文誌コンピューティングシステム, Vol.47, SIG 7(*ACS* 14), (2006), pp.61–80.

[6] Julien Langou: “Allreduce Householder factorizations”, Poster Sessions, Jul.11th,2007, at *2007 International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Scientific and Industrial Applications*, Météopole Toulouse,France.
Proceeding paper is available from:J. Langou: “Allreduce Householder factorizations”, ENSEEIHT-IRIT RT/AP0/07/10, pp.103–106. (also CERFACS TR/PA/07/71)

[7] Julien Jangou: “All Reduce Algorithms: Application to Householder *QR* Factorization”, at *Sparse Days at CERFACS Meeting*, Oct.12th, 2007, Toulouse, France.

[8] Yihua Bai and Robert C. Ward: “Parallel block tridiagonalization of real symmetric matrices”, J. Parallel Distrib. Comput., Vol.68, No.5(2008), pp.703–715.

[9] 村上 弘:「マルチコア CPU システムおよび小規模 SMP 並列システム上での Tall Skinny 型 *QR* 分解法の実験」, 情報処理学会論文誌：コンピューティングシステム, Vol.2, No.2(*ACS* 26), (2009), pp.19–29.

[10] 長谷川 秀彦:「古典的アルゴリズムによる大規模対称帯行列の固有値計算」, 計算工学講演会論文集, Vol.14,No.1(2009), pp.205–206.

ⓒ 2014 Information Processing Society of Japan