

キャッシュマシン向け三重対角化アルゴリズムの性能予測方式

山 本 有 作†

本論文では、level-3 BLAS を用いて構成される三重対角化アルゴリズムである Bischof & Wu のアルゴリズムに対し、性能予測モデルを構築する。本モデルは階層的な性能モデリング手法に基づいており、対象とするプロセッサ上での level-3 BLAS の性能、行列サイズ、2 種の性能パラメータを与えることで、三重対角化の実行時間を高精度に予測する。Opteron および Alpha 21264A 上で数値実験を行ったところ、本モデルは 1920 から 7680 までの行列サイズに対し、10% 以下の誤差で実行時間を予測できた。本モデルは、性能パラメータの最適値を自動的に推定する自動チューニングに利用できるほか、グリッド環境での三重対角化において、利用可能なマシンのうちで実行時間が最短となるマシンを自動的に選ぶためにも利用可能である。

Performance Prediction of a Tridiagonalization Algorithm for Cache-based Microprocessors

YUSAKU YAMAMOTO†

We construct a performance model for Bischof & Wu's tridiagonalization algorithm that is fully based on the level-3 BLAS. The model has a hierarchical structure, which reflects the hierarchical structure of the original algorithm, and given the matrix size, the two block sizes and the performance data of the underlying BLAS routines, predicts the execution time of the algorithm. Experiments on the Opteron and Alpha 21264A processors show that the model is quite accurate and can predict the performance of the algorithm for matrix sizes from 1920 to 7680 and for various block sizes with relative errors below 10%. The model will serve as a key component of an automatic tuned library that selects the optimal block sizes itself. It can also be used in a Grid environment to help the user find which of the available machines to use to solve his/her problem in the shortest time.

1. はじめに

実対称行列の固有値計算は広い分野で使われる基本的な線形計算の一つであり、分子計算、統計計算などに幅広い応用を持つ¹⁰⁾。行列が密行列である場合、もっともよく使われるアルゴリズムは、入力行列 A をハウスホルダー法により実対称三重対角行列 T に変換し、 T の固有値を二分法あるいは QR 法により求める方法である¹⁰⁾。このアルゴリズムにおいては、行列サイズを N とするとき、前半の三重対角化の演算量が約 $\frac{4}{3}N^3$ と大きく、演算時間の大部分を占めることが知られている。そこで、この三重対角化の計算を高速化することが重要な課題となる。

Intel Pentium 4, AMD Opteron, Alpha 21264A など、最近のマイクロプロセッサ上で計算を行う場合、キャッシュメモリを効率的に利用することが高速化の

鍵となる。このための方法としては、アルゴリズムをブロック化し、演算をできる限りデータ再利用性の高い level-3 BLAS (行列乗算) を用いて行うことが有効である。この方針に従った三重対角化のアルゴリズムとして、Dongarra のアルゴリズムがある⁹⁾。このアルゴリズムでは、ハウスホルダー法において演算量の半分を占める行列の rank-2 更新の部分、複数段分の更新をまとめて行うことにより、level-3 BLAS を用いて行うことを可能にする。このアルゴリズムは LAPACK¹⁾、ScaLAPACK⁴⁾ などに実装され、広く利用されている。しかし、Dongarra のアルゴリズムでは、演算量の残り半分を占める行列ベクトル積の部分はオリジナルのハウスホルダー法と同じである。そのため、この部分でのデータ再利用性の低さがネックとなり、最近のマイクロプロセッサ上ではピークの 10 ~ 25% 程度の性能しか達成できないことが知られている¹³⁾¹⁵⁾。

三重対角化のためのもう 1 つの方法としては、Bischof のアルゴリズムがある²⁾³⁾。このアルゴリズム

† 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,
Nagoya University

では、行列 $\mathbf{A}$ をまず適当な半帯幅 L を持つ帯行列 $\mathbf{B}$ に変換し、次に $\mathbf{B}$ を三重対角行列に変換する。この2つのステップのうち、前半部の演算量は約 $\frac{4}{3}N^3$ とハウスホルダー法と同じであり、かつ演算のほとんどすべてが level-3 BLAS を用いて実行できる。一方、後半部は level-3 BLAS の形では実行できないが、演算量は約 $6N^2L$ と、前半部に比べてオーダーが小さい。さらに、前半部では、ハウスホルダー法から Dongarra のアルゴリズムを導いたのと同様にして、更新演算を複数段 (M 段) 分まとめて行うことができ、性能をより向上させることが可能である。これを Bischof & Wu のアルゴリズムと呼ぶ¹⁴⁾。このアルゴリズムは、Opteron や Alpha 21264A などのプロセッサ上で、ピークの 50% 以上の高い性能を発揮できることが示されている¹⁵⁾。ただし、そのためには、行列サイズ N に応じて、性能に関係する2つのパラメータ L , M を最適な値に選ぶことが必要である。

本論文では、Bischof & Wu のアルゴリズムに対する性能予測モデルを提案する。本モデルは Cuenca et al. によって提案された階層的な性能予測モデル⁵⁾⁶⁾の考え方に基いており、プロセッサ上での level-3 BLAS の性能と N , L , M が与えられると、Bischof & Wu のアルゴリズムの実行時間を精度良く予測する。これにより、プロセッサと N に応じた最適な L , M の値を事前に推定することが可能となる。さらに、本モデルを使うことにより、グリッド環境での三重対角化において、利用可能なマシンのうちで実行時間が最短となるマシンを選ぶことや、与えられたマシンでの実行時間を予測してジョブの最適スケジューリングに役立てることも可能となる。

以下では、まず2章で Bischof & Wu の三重対角化アルゴリズムを説明する。次に3章で我々の提案する性能予測モデルの詳細を述べる。4章では Opteron と Alpha 21264A の2種のマシン上で数値実験を行い、本性能予測モデルの有効性を検証する。最後に5章でまとめを行う。

2. Bischof & Wu の三重対角化アルゴリズム

2.1 入力行列の帯行列への変換

Bischof の提案した三重対角化アルゴリズムは、入力行列 $\mathbf{A}$ の帯行列 $\mathbf{B}$ への変換と、帯行列 $\mathbf{B}$ の三重対角行列への変換という2つの部分からなる²⁾³⁾。このうち、前半部のアルゴリズムをアルゴリズム2に示す²⁾³⁾。ここで、 N , L はそれぞれ行列サイズ、 $\mathbf{B}$ の半帯幅であり、簡単のため、 N は半帯幅 L で割り切れると仮定する。

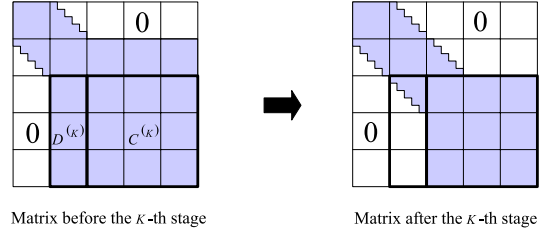


図1 Bischof のアルゴリズムの第 K 段における行列
Fig.1 matrix at the K -th stage of Bischof's algorithm.

[アルゴリズム 1: Bischof のアルゴリズム]
do $K = 1, N/L - 1$
 [ブロックハウスホルダ変換の作成]
 $\mathbf{D}^{(K)}$ を第1ブロックが上三角行列で
 第2ブロック以下がゼロ行列であるブロック
 ベクトルに変換するブロックハウスホルダー
 変換 $\mathbf{I} - \mathbf{U}^{(K)}\boldsymbol{\alpha}^{(K)}\mathbf{U}^{(K)t}$ を求める。
 [行列-ブロックベクトル積]
 $\mathbf{P}^{(K)} = \mathbf{C}^{(K)}\mathbf{U}^{(K)}\boldsymbol{\alpha}^{(K)}$
 $\boldsymbol{\beta}^{(K)} = \boldsymbol{\alpha}^{(K)t}\mathbf{U}^{(K)t}\mathbf{P}^{(K)}/2$
 $\mathbf{Q}^{(K)} = \mathbf{P}^{(K)} - \mathbf{U}^{(K)}\boldsymbol{\beta}^{(K)}$
 [行列の rank-2L 更新]
 $\mathbf{C}^{(K)} := \mathbf{C}^{(K)} - \mathbf{U}^{(K)}\mathbf{Q}^{(K)t} - \mathbf{Q}^{(K)}\mathbf{U}^{(K)t}$
end do

計算は $K = 1$ から $K = N/L - 1$ までの $N/L - 1$ 段からなる。各段における処理の流れは通常のハウスホルダー法とほぼ同一であり、ベクトルが幅 L のブロックベクトル（すなわち行列）に、スカラーがサイズ $L \times L$ の行列に置き換わる形となる。特に、ベクトル $\mathbf{d}^{(k)}$, $\mathbf{u}^{(k)}$, $\mathbf{p}^{(k)}$, $\mathbf{q}^{(k)}$ はそれぞれ幅 L のブロックベクトル $\mathbf{D}^{(K)}$, $\mathbf{U}^{(K)}$, $\mathbf{P}^{(K)}$, $\mathbf{Q}^{(K)}$ に、スカラー $\alpha^{(k)}$, $\beta^{(k)}$ はそれぞれ $L \times L$ 行列 $\boldsymbol{\alpha}^{(K)}$, $\boldsymbol{\beta}^{(K)}$ に置き換わる。第 K 段 ($1 \leq K \leq N/L - 1$) の処理では、図1に示すとおり、まず全体行列の第 K ブロック列の第 $K + 1$ 番目以降の要素からなるブロックベクトル $\mathbf{D}^{(K)}$ に着目し、 $\mathbf{D}^{(K)}$ を、第1ブロックが上三角行列でそれ以外の部分がゼロであるようなブロックベクトルに変換するブロックハウスホルダー変換 $\mathbf{H}^{(K)} = \mathbf{I} - \mathbf{U}^{(K)}\boldsymbol{\alpha}^{(K)}\mathbf{U}^{(K)t}$ を求める。次に、第 $K - 1$ 段での行列に対して $\mathbf{H}^{(K)}$, $(\mathbf{H}^{(K)})^{-1}$ をそれぞれ左側、右側からかけることにより、帯行列化が1段分進む(1)。これを $K = N/L - 1$ まで繰り返すことにより、帯行列化が完了する。²⁾³⁾

上記の処理においては、行列-ブロックベクトル積と行列の rank-2L 更新が、それぞれ演算量のほぼ半分を占める。これらは両方とも level-3 BLAS を用い

て行うことができ、 L が大きいほど 1 回の計算におけるデータの再利用性が高くなる。したがって、キャッシュサイズに応じて L を適切に取ることで、キャッシュマシン上で高い性能が期待できる。

2.2 Wu による改良

上記のアルゴリズムの性能を更に向上させるため、Wu は Dongarra による多段同時更新の考え方をアルゴリズム 1 に適用することを提案した¹⁴⁾。この結果得られるアルゴリズムをアルゴリズム 2 に示す。このアルゴリズムでは、アルゴリズム 1 において、1 段ごとに行列の rank- $2L$ 更新を行うのではなく、ある整数 M を決め、 M 段に 1 回、 M 段分の更新をまとめて行う。これにより得られるアルゴリズムをアルゴリズム 2 に示す。ここで、 $U^{(K)}$ など Sans Serif 体の記号は行列を要素とする行列を表し、 $(U^{(K)})_i$ はその第 i ブロック列のみからなる行列を表す。ただし、 K のみは整数を表すとする。

```
[アルゴリズム 2: Wu のアルゴリズム]
do K = 1, N/(LM) - 1
  U((K-1)*LM) =  $\phi$ , Q((K-1)*LM) =  $\phi$ 
  do K = (K-1)*M + 1, K*M
    [部分ハウスホルダー変換]
    D(K) := D(K)
      - U(K-1) (Q(K-1))t K-(K-1)*M
      - Q(K-1) (U(K-1))t K-(K-1)*M
    [ブロックハウスホルダー変換の作成]
    D(K) を第 1 ブロックが上三角行列で第 2
    ブロック以下がゼロ行列であるブロック
    ベクトルに変換するブロックハウスホルダ
    ー変換 I - U(K)  $\alpha^{(K)}$  U(K)t を求める。
    [行列-ブロックベクトル積]
    P(K) = (C(K) - U(K-1) Q(K-1))t
      - Q(K-1) U(K-1)t U(K)  $\alpha^{(K)}$ 
     $\beta^{(K)}$  =  $\alpha^{(K)}$ t U(K)t P(K) / 2
    Q(K) = P(K) -  $\beta^{(K)}$  U(K)
    U(K) = [U(K-1) | U(K)]
    Q(K) = [Q(K-1) | Q(K)]
  end do
  [行列の rank-2LM 更新]
  C(K*M) := C((K-1)*M) - U(K*M) Q(K*M)t
    - Q(K*M) U(K*M)t
end do
```

多段化の適用により、行列の rank- $2L$ 更新の部分は、rank- $2LM$ 更新となる。したがって Wu のアルゴ

リズムでは、Bischof のアルゴリズムこの部分に対して L の値を M 倍にしたのと同様の効果があり、後半の三重対角化部分の演算量を増やさずに帯行列化部分の性能を向上できると考えられる。

2.3 帯行列の三重対角行列への変換

アルゴリズム 1 または 2 によって得られた帯行列を三重対角化するには、村田法¹²⁾を用いる。村田法では、演算を通常の level-3 BLAS を用いて行うことはできないが、演算量が $O(6N^2L)$ であるため、 $L \ll N$ であれば、全演算量の中でこの部分が占める割合は小さい。

2.4 帯行列化で用いる BLAS ルーチン

アルゴリズム 2 を実装するには、様々な level-3 BLAS ルーチンが必要である。いま、アルゴリズム 2 中の行列はすべて列方向が連続アドレスになるように (BLAS の記法で言えば 'N' または 'Non-transposed' の形で) 格納されているとする。このとき、アルゴリズム中では AB , AB^t , $A^t B$ の 3 つのタイプの行列乗算が存在するため、これらに対応して、転置オプションそれぞれ 'N' 'N', 'T' 'N', 'N' 'T' となる 3 通りの DGEMM (一般密行列どうしの乗算) ルーチンのコールが必要である。また、行列-ブロックベクトル積では DSYMM (対称行列と一般密行列との積) ルーチン、rank- $2LM$ 更新では DSYR2K (行列の rank- $2K$ 更新) ルーチンのコールが必要である。さらに、ブロックハウスホルダー変換を行うルーチン (以下 BlockHouse と呼ぶ) のコールも必要である。以上より、アルゴリズム 2 の性能を予測するには、6 種の異なる BLAS ルーチンの性能を知る必要があることがわかる。

3. 性能予測モデル

3.1 原 理

アルゴリズム 2 の性能を予測するための手法としては、階層的な性能モデリング手法⁵⁾⁶⁾を用いる。この手法では、まず各 BLAS ルーチンに対し、行列サイズや転置オプションなどを入力として対象とするプロセッサ上での実行時間を高精度に予測する経験的モデルを構築する。次に、アルゴリズム中での BLAS ルーチンのコール 1 つ 1 つについて、行列サイズと転置オプションなどのパラメータより、モデルを用いて実行時間を予測する。そしてこれらをすべて足し上げることで、全実行時間を予測する。この手法は、LU 分解や QR 分解などの基本的な行列分解の性能予測に適用され、精度の良い性能予測ができることが確かめられている⁵⁾⁶⁾⁷⁾。

3.2 BLAS 性能のモデリング

2.4 節で述べたように、アルゴリズム 2 では 5 種類の level-3 BLAS ルーチン（3 種類の DGEMM と DSYMM, DSYR2K）および BlockHouse のコールが必要である。そこで、まずこれら 6 種のルーチンに対し、性能モデルを構築する必要がある。本節では、転置オプションが 'N' 'N' の DGEMM を例に取り、その構築手法を述べる。このルーチンは、 $m \times k$ 行列 A と $k \times n$ 行列 B に対し、 $m \times n$ 行列 $C = AB$ を計算する。モデル構築の目的は、 m, n, k の関数としての実行時間 $f_{\text{DGEMM,NN}}(m, n, k)$ の式を求めることである。

アルゴリズム 2 の解析より、DGEMM（転置オプション 'N' 'N'）のコールにおいて、 N は常に半帯幅 L に等しく、一方、 m, k は最大 N までの値を取ることがわかる。この観察に基づき、本研究では、異なる n の値に対する実行時間を異なる関数で近似することにする。なぜなら、 L は 1 から 100 程度の比較的小さい値を取るが、このような小さいサイズの範囲では level-3 BLAS の性能は不規則な変化を示すことが多いため、1 つの関数で近似すると、誤差が大きくなると考えられるからである。一方、 n を固定したときの実行時間は、 m, k の双一次式で近似する。すなわち、実行時間の式は次のようになる。

$$\begin{aligned} f_{\text{DGEMM,NN}}(m, n, k) \\ &= f_{\text{DGEMM,NN}}^n(m, k) \\ &= (a_{11}^n m + a_{10}^n)k + (a_{01}^n m + a_{00}^n). \end{aligned} \quad (1)$$

係数 $a_{11}^n, a_{10}^n, a_{01}^n, a_{00}^n$ は実測性能から最小二乗法により定める。なお、実際には n の値は $\{3, 6, 12, 24, 48, 96\}$ の 6 通りのみで $f_{\text{DGEMM,NN}}^n(m, k)$ を求め、それ以外の n の値での実行時間が必要な場合は一時補間により計算する。残りの 2 つの DGEMM と DSYMM, DSYR2K の性能モデリングも、同様の方針で行う。

3.3 BlockHouse の性能モデリング

サブルーチン BlockHouse は、ブロックサイズ（半帯幅） L とブロックベクトルの長さ n の 2 つのサイズパラメータを持つ。BlockHouse に対しては、前節で用いたような双一次式による近似は良い精度を与えない。なぜなら、BlockHouse は level-3 BLAS で記述されておらず、キャッシュ利用効率が十分高くないため、 L または n が大きい領域で性能が落ちるからである。そこで、BlockHouse の性能モデルを作るにあたっては、 (L, n) 空間上の十分多数の格子点で性能を測定し、これを双一次補間する。

3.4 アルゴリズム 2 の性能モデリング

以上で作成した level-3 BLAS および BlockHouse

の性能モデルを用いて、アルゴリズム 2 の性能モデルを構築する。この場合、従来採用されてきた手法としては、BLAS の各ルーチンで実行される演算の量を解析的に求め、これから BLAS の性能モデルにより、各ルーチンごとの総実行時間を求めて、これを足し合わせるという手法がある。しかしこの手法では、演算量を単純な足し合わせにより求めるのでは、演算性能の入力サイズ依存性が反映されず、精度が低くなるという問題点がある。一方、行列サイズ依存性を反映させようとすると、解析的な式が複雑になり、モデル構築の手間が大きくなるという問題点がある。

そこで本研究では、演算量の解析的な式を求めることをやめ、より簡単なアプローチを取ることにする。具体的には、まず DGEMM.TIME, DSYMM.TIME, DSYR2K.TIME, BlockHouse.TIME などの関数を新たに作成する。これらの関数は、それぞれ DGEMM, DSYMM, DSYR2K, BlockHouse と同じ引数を持つが、演算結果を返す代わりに、与えられた引数に対する実行時間を性能モデルにより予測し、その結果を返す。一方、アルゴリズム 2 のプログラムも、各 BLAS ルーチンの代わりに .TIME のついたルーチンをコールし、その結果の予測時間を累積するよう書き換える。これにより、 N, L, M を与えると、アルゴリズム 2 の実行時間を予測するプログラムが生成できる。

本方式では、各 BLAS ルーチンの性能の入力サイズ依存性が正確に反映されるので、高精度な予測ができると期待される。また、ある (N, L, M) の組に対して予測を行うための演算量は、アルゴリズム 2 から容易にわかるように $O(N/L)$ であり、実際に実行した場合の演算量 $\frac{4}{3}N^2$ に比べて無視できるほど小さい。

4. 実験結果

4.1 BLAS 性能のモデリング

以上で示した性能予測手法の有効性を検証するため、Opteron (1.6GHz, メモリ 2GB) と Alpha 21264A (750MHz, メモリ 512MB) 上で実験を行った。コンパイラは g77 (最適化レベル-O4)、BLAS は GOTO BLAS である。サブルーチン BlockHouse は自作であり、特にチューニングは行っていない。

最初に、3.2 で述べた方法に従い、BLAS の性能モデリングを行った。DGEMM（転置オプション 'N' 'N'）のモデリングを行うに当たっては、 n, m, k がそれぞれ集合 $\{3, 6, 12, 24, 48, 96\}$, $\{3, 6, 12, 24, 48, 96\}$, $\{100, 200, 400, 800, 1000\}$ のうちの 1 つの値を取るようにし、 $6 \times 6 \times 5 = 180$ 回の測定を行った。そして、 n の値を固定したときの実行時間を式 (1) のように $m,$

k の双一次式で表し、係数 a_{11}^n , a_{10}^n , a_{01}^n , a_{00}^n を最小二乗法により求めた。Opteron の場合の結果を表 1 に示す。また、 $n = 12$ の場合に、モデル式により得られた実行時間と実測した実行時間との比較を表 2 に示す。表より、モデル式は実際の実行時間を良く近似しており、特に k が大きいとき精度が良いことがわかる。他の level-3 BLAS ルーチンと BlockHouse についても、同様にしてモデルを作成したところ、同程度の精度が得られた。

表 1 The coefficients of eq. (1) for DGEMM('N','N') on the Opteron.

n	a_{11}^n	a_{10}^n	a_{01}^n	a_{00}^n
3	8.75E-9	5.54E-8	-1.54E-9	2.21E-6
6	9.83E-9	7.99E-8	1.08E-8	3.18E-6
12	1.19E-8	1.73E-7	3.63E-8	4.92E-6
24	1.98E-8	2.14E-7	7.64E-8	8.33E-6
48	3.55E-8	3.33E-7	1.57E-7	1.60E-5
96	6.73E-8	4.77E-7	3.11E-7	3.29E-5

表 2 Actual (above) and estimated (below) execution times (s) of DGEMM('N'*N') (Opteron, $n = 12$).

k	$m = 100$	$m = 200$	$m = 400$	$m = 800$
3	1.39E-05	2.17E-05	3.80E-05	7.05E-05
	1.26E-05	1.98E-05	3.42E-05	6.29E-05
6	1.63E-05	2.68E-05	4.88E-05	9.25E-05
	1.67E-05	2.74E-05	4.89E-05	9.19E-05
12	2.36E-05	3.99E-05	7.53E-05	1.44E-04
	2.48E-05	4.27E-05	7.84E-05	1.50E-04
24	3.97E-05	6.84E-05	1.31E-04	2.53E-04
	4.12E-05	7.32E-05	1.37E-04	2.66E-04
48	7.18E-05	1.31E-04	2.53E-04	4.95E-04
	7.38E-05	1.34E-04	2.55E-04	4.97E-04
96	1.40E-04	2.55E-04	4.98E-04	9.77E-04
	1.39E-04	2.56E-04	4.91E-04	9.61E-04

4.2 アルゴリズム 2 の性能予測

次に、3.4 節の方法に従い、アルゴリズム 2 に対するモデルを作成し、 N , L , M を変えたときの実行時間を予測して実測値と比較した。行列サイズ N は 1920, 3840, 7680 の 3 通りの値 (7680 は Opteron のみ) を取り、 L , M はそれぞれ $\{3, 6, 12, 24, 48, 96\}$, $\{1, 2, 4, 8, 16, 32, 64\}$ のどれかの値を取るようにした。ただし、 $M \times L \leq 384$ という条件を加えた。

Opteron の $N = 1920$ での結果を表 3 に示す。表中で、上段の数字は実測時間、下段の数字は予測時間を示す。表より、予測誤差は最大でも 10% と小さいことがわかる。他の N についても同様の結果が得られた。特に、 $N = 7680$ の場合は極めて精度が高く、予

測誤差は最大 2% であった。

Alpha 21264A での $N = 1920$ の結果を表 4 に示す。この場合、最大予測誤差は 4% である。 $N = 3840$ の場合は、最大予測誤差は 5% であった。これらの結果から、本研究で構築したモデルはどちらのマシンにおいても、実行時間を精度良く予測することが可能であると言える。

表 3 Actual (above) and estimated (below) execution times (s) of reduction to a band matrix (Opteron, $N = 1920$).

$M \setminus L$	3	6	12	24	48	96
1	16.63	9.57	6.15	4.98	4.65	4.96
	15.36	8.90	5.83	4.73	4.46	4.86
2	14.28	8.51	5.73	4.88	4.70	5.20
	12.97	7.88	5.33	4.62	4.53	5.07
4	13.23	8.11	5.66	4.95	4.94	5.67
	11.98	7.44	5.25	4.70	4.75	5.49
8	12.89	8.10	5.76	5.22	5.49	
	11.61	7.41	5.37	4.96	5.22	
16	13.05	8.38	6.14	5.90		
	11.75	7.70	5.71	5.51		
32	13.67	9.10	7.04			
	12.36	8.36	6.43			
64	15.07	10.75				
	13.68	9.69				

表 4 Actual (above) and estimated (below) execution times (s) of reduction to a band matrix (Alpha, $N = 1920$).

$M \setminus L$	3	6	12	24	48	96
1	42.01	23.04	14.27	11.02	9.98	10.55
	41.59	23.81	14.54	11.11	9.95	10.31
2	37.27	20.83	13.55	10.86	10.08	11.02
	38.41	21.42	13.81	11.03	10.16	10.90
4	35.26	20.16	13.42	11.04	10.60	11.96
	36.09	20.76	13.77	11.28	10.79	11.76
8	34.66	20.24	13.68	11.70	11.71	
	35.63	20.91	14.15	12.04	11.78	
16	35.02	20.81	14.63	13.04		
	36.23	21.72	15.21	13.32		
32	36.36	22.67	16.59			
	37.93	23.61	17.06			
64	39.94	26.36				
	41.56	27.05				

5. おわりに

本研究では、キャッシュマシン向けの三重対角化アルゴリズムである Bischof & Wu のアルゴリズムを対象として、性能予測モデルを構築した。本モデルは階層的な性能予測手法に基づき、三重対角化の実行時間を高精度に予測する。Opteron および Alpha 21264A で

の評価の結果、本モデルは $N = 1920$ から $N = 7680$ までの行列サイズに対し、10%以下の誤差で実行時間を予測できることを明らかにした。発表では、このモデルを用いて性能パラメータの最適化を行った結果についても報告する予定である。

今後の課題としては、より多くのプロセッサに対して本手法を適用し、有効性を検証すること、共有メモリ型および分散メモリ型並列計算機上での三重対角化に対し、本手法を拡張することが挙げられる。また、固有値計算プログラム全体に対する性能予測モデルの構築と、その自動チューニング型ライブラリへの適用も課題である。

謝辞 日頃から有益な議論をして頂いている自動チューニング研究会のメンバーに感謝いたします。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」および科学研究費補助金若手研究 (B) (課題番号 16760053) の補助を受けている。

参 考 文 献

- 1) Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S. and Sorensen, D.: *LAPACK Users' Guide*, Second edition, SIAM, Philadelphia, 1995.
- 2) Bischof, C., Marques, M. and Sun, X.: Parallel Bandreduction and Tridiagonalization, Technical Report 8, *PRISM Working Note*, 1993.
- 3) Bischof, C., Lang, B. and Sun, X.: Parallel Tridiagonalization through Two-step Band Reduction, Technical Report 17, *PRISM Working Note*, 1994.
- 4) Blackford, L. S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D. and Whaley, R. C.: *ScaLAPACK Users' Guide*, SIAM, Philadelphia, 1997.
- 5) J. Cuenca, L. -P. Garcia and D. G. Gimenez: Empirical Modeling of Parallel linear Algebra Routines, in *Proceedings of the 5th International Conference on Parallel Processing and Applied Mathematics (PPAM2003)*, Lecture Notes in Computer Science, No. 3019, pp. 169-174, Springer-Verlag, 2004.
- 6) J. Cuenca, D. Gimenez and J. Gonzalez: Architecture of an automatically Tuned Linear Algebra Library, *Parallel Computing*, Vol. 30, pp. 187-210 (2004).
- 7) K. Dackland and B. Kågström: A Hierarchical Approach for Performance Analysis of ScaLapack-based Routines Using the Linear Algebra Machine, in *proceedings of Workshop on Applied Parallel Computing in Industrial Computation and Optimization (PARA96)*, Lecture Notes in Computer Science, No. 1184, pp. 187-195, Springer-Verlag, 1996.
- 8) J. Dongarra and V. Eijkhout: Self-Adapting Numerical Software for Next Generation Applications, *International Journal of High Performance Computing Applications*, Vol. 17, No. 2, pp. 125-131 (2003).
- 9) Dongarra, J. J., Hammarling, S. J. and Sorensen, D. C.: Block Reduction of Matrices to Condensed Forms for Eigenvalue Computations, *Journal of Computational and Applied Mathematics*, Vol. 27, pp. 215-227 (1989).
- 10) Golub, G.H. and van Loan, C.F.: *Matrix Computations*, Third edition, Johns Hopkins University Press, 1996.
- 11) T. Katagiri, H. Kuroda and Y. Kanada: A Methodology for Automatically Tuned Parallel Tridiagonalization on Distributed Memory Parallel machines, in *Proceedings of Vec-Par2000*, pp.265-277, Faculdade de Engenharia da Universidade do Porto, Portugal, June 2000.
- 12) 村田健郎, 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.
- 13) Salvini, S. A. and Mulholland, L. S.: The NAG FORTRAN Library, in *Proceedings of the Ninth SIAM Conference on Parallel Processing and Scientific Computing 1999*, SIAM, Philadelphia (1999).
- 14) Wu, Y.-J. J., Alpatov, P. A., Bischof, C. H. and van de Geijn, R.A.: A Parallel Implementation of Symmetric Band Reduction Using LAPACK, in *Proceedings of the Scalable Parallel Libraries Conference*, 1996.
- 15) 山本有作: キャッシュマシン向け対称密行列固有値解法の性能・精度評価, 情報処理学会論文誌 ACS, Vol. 46, No. SIG3 (ACS8), pp. 81-91 (2005).