Vol.2011-HPC-130 No.42
2011/7/28

QR 分解アルゴリズムに対する自動チューニング –性能モデルに関する考察–

深谷 猛[†1] 山本有作[†2] 張 紹良[†1]

これまで，QR 分解をはじめとする密行列計算アルゴリズムのブロック化に対する自動チューニング手法の研究を行ってきた．実用的なコストでチューニングを行うためには，チューニングの過程で使用する性能データの全てを実際に測定することは難しく，モデルを用いた予測が必要となる．そこで，本稿では，これまでに提案した手法の内部で使用する性能モデルについて検討を行う．数値実験の結果，適切なモデルを用いることで，チューニングのコストを大幅に削減しつつ，十分な効果を得られることが確認された．

Automatic Tuning for the Algorithm of QR Decomposition – an investigation into performance models –

TAKESHI FUKAYA,[†1] YUSAKU YAMAMOTO[†2]
AND SHAO-LIANG ZHANG[†1]

Recently, we have studied the automatic performance tuning for algorithms of basic matrix computations such as the QR decomposition. In terms of the tuning cost, it is not practical to measure all values required in the tuning process. Therefore, values estimated by performance models are used, instead of exact values. In this paper, we discuss the performance models used in our proposed auto-tuning method. Numerical experiments show that some models decrease tuning cost and keep the effect of tuning.

†1 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
†2 神戸大学大学院システム情報学研究科計算科学専攻
Department of Computational Science, Graduate School of System Informatics, Kobe University

1. はじめに

計算機環境の多様化・複雑化に伴い，計算機の性能を引き出すためのソフトウェアのチューニング作業は困難かつ手間のかかる作業になっている．そのため，機械的にチューニングを行う仕組み（自動チューニング）の研究が注目されている[1)]．

このような背景のもと，我々は QR 分解などの基本的な密行列計算アルゴリズムに対する自動チューニング手法の研究を行っている．特に，これらの計算で重要となるブロック化の方法を機械的に決定する手法の研究を行ってきた[2)3)]．これまでに，我々は，様々なブロック化の方法を二分木で系統的に表現し，動的計画法に基づいたアルゴリズムにより，最適な二分木を効率的に決定する手法を開発した．

本稿では，QR 分解アルゴリズムを対象として，これまでに開発した自動チューニング手法の内部で用いる性能モデルについて検討する．チューニングに要するコストを抑えるためには，手法の内部で使用する性能値を全て実測することは難しく，何らかのモデルによる予測が必要となる．しかし，予測の際に生じる誤差による影響も重要な問題となり得る．本稿では，実際に測定した値を用いてチューニングをした場合と，モデルを用いて予測した値を用いてチューニングをした場合について，チューニングのコストと成果について比較し，実用的な自動チューニング手法の構築のための課題を明らかにする．

2. QR 分解アルゴリズムとブロック化

本研究で扱う QR 分解アルゴリズムについて紹介する．

2.1 QR 分解

$m \times n$ の行列 A を

$$A = QR,$$

と分解することを QR 分解と呼ぶ．ここで，Q は $m \times m$ の直交行列，R は $m \times n$ の上三角行列である．ただし，Q を一つの行列として陽的に求める必要がない場合も多い．

2.2 ハウスホルダー QR 分解

QR 分解を計算する代表的な手法の一つに，ハウスホルダー変換（$H := I - t\boldsymbol{y}\boldsymbol{y}^T$）を用いたものがある[4)]．この手法では，行列 A に対して，

$$H_n \cdots H_1 A = R,$$

と逐次的にハウスホルダー変換を作用させて R を得る（**図 1**）．また，Q はハウスホルダー変換の積として陰的に保持することが一般的である．

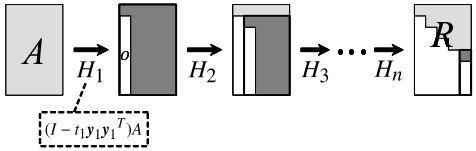

図 1 ハウスホルダー変換による QR 分解の計算
Fig. 1 QR decomposition using the Householder transformations.

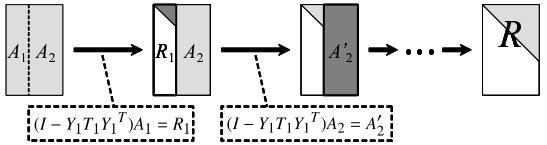

図 2 ブロック化されたハウスホルダー QR 分解の計算
Fig. 2 Blocked Householder QR decomposition.

計算の大半は，ハウスホルダー変換を行列に作用させる部分が占めるが，この部分は，

- $\boldsymbol{y}^T A \rightarrow \boldsymbol{w}^T$（行列ベクトル積）
- $A - t\boldsymbol{y}\boldsymbol{w}^T$（rank-1 更新）

と Level-2 BLAS に相当する．そのため，近年の計算機では高い実行性能を得ることが難しく，次節で紹介するブロック化を行うことが一般的である．

2.3 ブロック化

複数個のハウスホルダー変換は，

$$(I - t_p \boldsymbol{y}_p {\boldsymbol{y}_p}^T) \cdots (I - t_1 \boldsymbol{y}_1 {\boldsymbol{y}_1}^T) = I - YTY^T,$$

と行列の形にまとめることができる[5]．なお，$Y = [\boldsymbol{y}_1 \cdots \boldsymbol{y}_p]$，$T$ は下三角行列で $t_i, \boldsymbol{y}_i$ $(i = 1, \ldots, p)$ から計算される．

この性質を用いてハウスホルダー QR 分解をブロック化した場合の計算は，

(1) $A \rightarrow [A_1 A_2]$（行列を分解）
(2) $(I - Y_1 T_1 Y_1^T) A_1 = R_1$（$A_1$ の QR 分解）
(3) $(I - Y_1 T_1 Y_1^T) A_2 \rightarrow A_2'$（$A_2$ の更新）

という流れで行われる（**図 2**）．

このようにすることで，(3) の部分が行列乗算で計算できるようになるので，実行性能の向上が期待される．一方，ハウスホルダー変換を合成する際の T の計算により，QR 分解全体の計算量は増加する．

2.4 ブロック分割法のチューニング

実際にブロック化をして QR 分解を計算するためには，以下の点を決める必要がある．

- ブロック幅
- 部分的な QR 分解の計算方法（再帰的にブロック化が可能なため）

本論文では，上記の 2 点をまとめて,「ブロック分割法」と呼ぶことにする．

すでに述べたように，ブロック化をする場合，行列乗算を使うことによる性能向上と，ハウスホルダー変換の合成に伴う計算量の増加がトレードオフの関係にある．そこで，使用する計算機環境や問題サイズに応じて，できるだけ計算時間が短くなるようなブロック分割法を決定することが必要になる．なお，現状では，上記の 2 点のうち，部分的な QR 分解の計算方法を決めた上で，ブロック幅をチューニングすることが一般的である[6][7]．

3. ブロック分割法の自動チューニング

我々がこれまでに行ってきた，ブロック分割法の自動チューニングの研究について説明する．

3.1 方　　針

研究の目的は，ブロック分割法の自動チューニング手法の構築である．つまり，計算機環境と問題の行列が与えられたときに，できるだけ計算時間が短くなるようなブロック分割法を機械的に決定する手法を開発することである．なお，本研究では，行列ベクトル積や行列乗算などの基本演算は BLAS ライブラリを使用することを想定しており，計算機環境にこれを含める．

これまでに述べたように，再帰的なブロック化が可能なため，ブロック化における自由度は非常に大きいと言える．本研究では，この自由度をできるだけ維持した自動チューニング手法を構築する．つまり，これまでのように，経験的に自由度を削減してからチューニングを行うのではなく，どのような状況にも対応できるような非経験的なチューニングを行うことを目指す．

3.2 これまでに提案した手法

3.2.1 二分木を用いたブロック分割法の表現

自動チューニングを行うためには，まず，チューニングされるパラメータを導入する必要がある．つまり，ブロック分割法をパラメータを用いて表現することことになる．

ブロック化の基本原理は，行列を 2 つのブロック（ハウスホルダー変換をまとめるブロッ

Vol.2011-HPC-130 No.42
2011/7/28

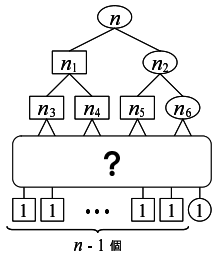

図 3 二分木による表現
Fig. 3 Representation with a binary tree.

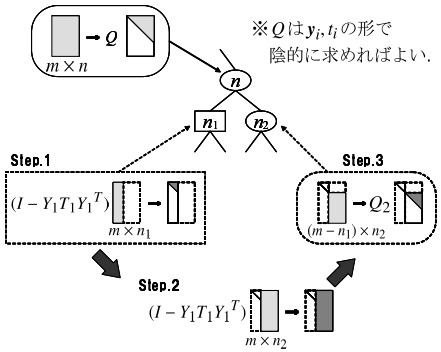

図 4 二分木のノードで行う計算手順 (1)
Fig. 4 Computation on each node of a binary tree (1).

クと作用されるブロック）に分割することと，再帰的にブロック化ができることである．そこで，**図 3** に示したような二分木を用いて，ブロック分割法を表現することを考える．

なお，二分木と QR 分解の計算の対応は**図 4** と**図 5** に示した通りである．

3.2.2 動的計画法に基づく二分木の最適化

詳細は文献 2) や文献 3) に委ねるが，動的計画法を用いて二分木を最適化する手法について紹介する．二分木の最適化は，まず，部分木を最適化して，そのうえで最適な部分木の組合せを決定すればよい．そのため，小さい部分木から順番に，過去の最適化の結果を再利用して最適化を行うという，動的計画法による最適化が可能である．

動的計画法を用いて二分木の最適化を行うアルゴリズムを Algorithm 1 に示す．なお，

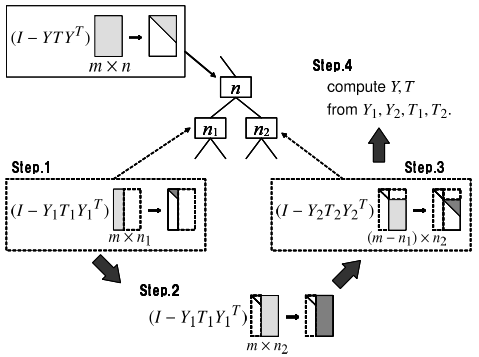

図 5 二分木のノードで行う計算手順 (1)
Fig. 5 Computation on each node of a binary tree (2).

- $T_{\mathrm{QR}}[m,n]$：最適なブロック分割法で $m \times n$ の行列を QR 分解（Q は陰的に計算）した場合の計算時間.
- $T_{\mathrm{WY}}[m,n]$：最適なブロック分割法で $m \times n$ の行列を QR 分解（$Q = I - YTY^T$）した場合の計算時間.
- $B_{\mathrm{QR}}[m,n]$：二分木の丸のノード（サイズが $m \times n$）に対する最適な分割幅（図 4 の n_1 に相当）.
- $B_{\mathrm{WY}}[m,n]$：二分木の四角のノード（サイズが $m \times n$）に対する最適な分割幅（図 5 の n_1 に相当）.
- $T_{\mathrm{App}}(,,)$：$I - YTY^T$ で行列を更新する計算時間（図 4，5 の Step.2 に相当）.
- $T_{\mathrm{Agg}}(,,)$：$I - YTY^T$ に合成する計算時間（図 5 の Step.4 に相当）.

となっている．

4. 性能モデルについて

Algorithm 1 を用いてブロック分割法（二分木）の最適化を行う際に，$T_{\mathrm{App}}(), T_{\mathrm{Agg}}()$ の値が必要となる．これらは，引数によってサイズが決まる行列に対するいくつかの BLAS ルーチンの実行時間の総和である．これらの値を得る方法として，そのルーチンを実際に実行して時間を測定する，ということが考えられる．しかし，全ての値を測定することになると，チューニングに要する時間が膨大になり，現実的とは言い難い．そのため，性能モデルを用いて値を予測するということが必要になる．しかし，予測値を用いた場合は，予測の際に生

Vol.2011-HPC-130 No.42
2011/7/28

Algorithm 1 Dynamic Programming (DP)

Input: m, n

Output: $B_{\mathrm{QR}}[,], B_{\mathrm{WY}}[,]$

```
for h = 1 to m do
  T_QR[h,1] = T_WY[h,1] ← House(h)
end for
for k = 2 to n do
  for h = k to m do
    T_QR[h,k] = T_WY[h,k] ← +∞
    for i = 1 to k − 1 do
      t_QR ← T_WY[h,i] + T_App(h,i,k−i)
           +T_QR[h−i,k−i].
      t_WY ← T_WY[h,i] + T_App(h,i,k−i)
           +T_WY[h−i,k−i] + T_Agg(h,i,k−i).
      if t_QR < T_QR[h,k] then
        T_QR[h,k] ← t_QR, B_QR[h,k] ← i
      end if
      if t_WY < T_WY[h,k] then
        T_WY[h,k] ← t_WY, B_WY[h,k] ← i
      end if
    end for
  end for
end for
```

じる誤差により最適化の結果が影響を受けることも十分考えられる．そこで，性能モデルを用いて最適化を行った場合と，実際に計算時間を測定して最適化を行った場合を比較し，結果について考察する．

4.1 今回使用する性能モデル

$T_{\mathrm{App}}(), T_{\mathrm{Agg}}()$ の部分の計算では，ともに DGEMM や DTRMM などの BLAS ルーチンを複数回実行する．そこで，個々の BLAS ルーチンの実行時間を予測し，それらの足し合わせとして，$T_{\mathrm{App}}(), T_{\mathrm{Agg}}()$ の値を予測する．

個々の BLAS ルーチンの実行時間の予測は，サンプル点の値を使った多重線形補間を用いる．以下では，DGEMM などサイズパラメータが 3 個の場合を例にして説明する．

サイズパラメータを m, n, k とする．サンプル点を，

$$m = m_1, m_2, \cdots, m_i, \cdots, m_M \quad (m_1 < m_2 < \cdots < m_M),$$
$$n = n_1, n_2, \cdots, n_j, \cdots, n_N \quad (n_1 < n_2 < \cdots < n_N),$$
$$k = k_1, k_2, \cdots, k_h, \cdots, k_K \quad (k_1 < k_2 < \cdots < k_K),$$

として，$M \times N \times K$ 個の点 (m_i, n_j, k_h) で BLAS ルーチンの FLOPS 値：$f(m_i, n_j, k_h)$ を測定する．

補間により求めたい点を $(\bar{m}, \bar{n}, \bar{k})$ とする．まず，

$$m_{\bar{i}} < \bar{m} \le m_{\bar{i}+1},$$
$$n_{\bar{j}} < \bar{n} \le n_{\bar{j}+1},$$
$$k_{\bar{h}} < \bar{k} \le k_{\bar{h}+1},$$

を満たす $(\bar{i}, \bar{j}, \bar{h})$ を求める．次に，$(\bar{m}, \bar{n}, \bar{k})$ における予測値：$\bar{f}(\bar{m}, \bar{n}, \bar{k})$ を以下のようにして求める．

$$\begin{aligned}
\bar{f}(\bar{m}, \bar{n}, \bar{k}) = \frac{1}{\Delta m_{\bar{i}}\, \Delta n_{\bar{j}}\, \Delta k_{\bar{h}}} \Big\{ & |m_{\bar{i}+1} - \bar{m}||n_{\bar{j}+1} - \bar{n}||k_{\bar{h}+1} - \bar{k}| f(m_{\bar{i}}, n_{\bar{j}}, k_{\bar{h}}) \\
& + |m_{\bar{i}+1} - \bar{m}||n_{\bar{j}} - \bar{n}||k_{\bar{h}+1} - \bar{k}| f(m_{\bar{i}}, n_{\bar{j}+1}, k_{\bar{h}}) \\
& + |m_{\bar{i}} - \bar{m}||n_{\bar{j}+1} - \bar{n}||k_{\bar{h}+1} - \bar{k}| f(m_{\bar{i}+1}, n_{\bar{j}}, k_{\bar{h}}) \\
& + |m_{\bar{i}} - \bar{m}||n_{\bar{j}} - \bar{n}||k_{\bar{h}+1} - \bar{k}| f(m_{\bar{i}+1}, n_{\bar{j}+1}, k_{\bar{h}}) \\
& + |m_{\bar{i}+1} - \bar{m}||n_{\bar{j}+1} - \bar{n}||k_{\bar{h}} - \bar{k}| f(m_{\bar{i}}, n_{\bar{j}}, k_{\bar{h}+1}) \\
& + |m_{\bar{i}+1} - \bar{m}||n_{\bar{j}} - \bar{n}||k_{\bar{h}} - \bar{k}| f(m_{\bar{i}}, n_{\bar{j}+1}, k_{\bar{h}+1}) \\
& + |m_{\bar{i}} - \bar{m}||n_{\bar{j}+1} - \bar{n}||k_{\bar{h}} - \bar{k}| f(m_{\bar{i}+1}, n_{\bar{j}}, k_{\bar{h}+1}) \\
& + |m_{\bar{i}} - \bar{m}||n_{\bar{j}} - \bar{n}||k_{\bar{h}} - \bar{k}| f(m_{\bar{i}+1}, n_{\bar{j}+1}, k_{\bar{h}+1}) \Big\}.
\end{aligned}$$

ただし，

$$\Delta m_{\bar{i}} = |m_{\bar{i}+1} - m_{\bar{i}}|, \quad \Delta n_{\bar{j}} = |n_{\bar{j}+1} - n_{\bar{j}}|, \quad \Delta k_{\bar{h}} = |k_{\bar{h}+1} - k_{\bar{h}}|,$$

である．このようにして予測された FLOPS 値から，BLAS ルーチンの実行時間を予測する．

DTRMM などサイズパラメータが 2 個の場合も同様にして行う．また，行列の転置などを指定するパラメータが異なる場合は，それぞれ区別して予測を行うこととする．

5. 数 値 実 験

$T_{\mathrm{App}}, T_{\mathrm{Agg}}$ に実測値および線形補間による予測値を用いて，動的計画法でブロック分割法の最適化を行う．そして，得られたブロック分割法を使って QR 分解の計算をした際の計算時間の評価を行う．

5.1 計算機環境

使用した計算機環境は**表 1** の通りである．

表 1 性能評価に使用した計算機環境
Table 1 Computational environments used in experiments.

項目	条件
CPU	Intel Xeon X5670 (2.93GHz) 1 コアのみ使用
メモリ	24GB
OS	Yellow Dog Enterprise Linux
コンパイラ	gcc ver. 4.1.2 (オプション -O3)
BLAS	ATLAS ver. 3.8.0

5.2 実 験 方 法

$T_{\mathrm{App}}, T_{\mathrm{Agg}}$ の値を以下の 3 種類の方法で取得する．

model 1：実際に BLAS ルーチンを実行して，その実行時間を用いる．

model 2：線形補間で予測する．サンプル点は $1, 2, 4, 8, 16, 32, 64, 128, 256, 512$.

model 3：線形補間で予測する．サンプル点は $1, 3, 9, 27, 81, 243, 512$.

比較したブロック分割法は以下の 6 種類である．

b： recursive bisection

f： fixed-size blocking

h： hybrid blocking (f と b)

d1：model 1 を用いて動的計画法により最適化された分割法

d2：model 2 を用いて動的計画法により最適化された分割法

d3：model 3 を用いて動的計画法により最適化された分割法

なお，f と h は，ブロック幅を 1 刻みで変えて計算時間を測定し，最小となった場合を採用した．

5.3 実 験 結 果

まず，最適化に要した時間を**表 2** に示す．なお，表 2 のサンプリングは，model 2,3 で用いる BLAS ルーチンの実行時間のサンプルデータの測定にかかった時間である．

表 2 最適化に要した時間（秒）
Table 2 Time for optimization (sec.).

モデル	サンプリング	動的計画法	合計
model 1	–	1.95×10^5	1.95×10^5
model 2	5.81	374	380
model 3	2.73	373	376

次に，行列サイズを変えて QR 分解の計算時間を測定した結果を**表 3** に示す．なお，表中の f と h における括弧内の値はブロック幅を示す．

表 3 QR 分解の計算時間（ミリ秒） ※括弧内の数字はブロック幅を示す
Table 3 Execution time for computing QR decomposition (msec.). ※ Parenthetic number means the optimal block length.

		execution time (msec.)					
m	n	b	f	h	d1	d2	d3
128	128	1.42	1.13 (12)	1.12 (12)	1.10	1.53	1.41
256	128	2.48	2.05 (12)	2.04 (12)	2.03	2.35	2.66
256	256	6.97	5.58 (12)	5.47 (48)	5.32	6.66	7.28
384	128	3.59	3.01 (12)	2.96 (12)	2.92	3.29	4.00
384	256	10.4	8.61 (12)	8.29 (48)	8.20	9.16	11.1
384	384	17.8	15.5 (18)	14.5 (48)	14.4	16.9	20.2
512	128	4.77	3.93 (16)	3.89 (12)	3.77	4.15	5.08
512	256	13.8	11.8 (12)	11.3 (48)	11.1	12.2	14.7
512	384	24.1	21.8 (18)	20.4 (48)	20.2	22.3	28.9
512	512	39.8	33.3 (18)	30.6 (48)	30.8	34.6	42.6

また，$m = n = 512$ の場合についての詳しい結果を**図 6** に示す．

5.4 考 察

予測値を使って最適化を行った結果得られたブロック分割法は，実測値を使った場合に得

Vol.2011-HPC-130 No.42
2011/7/28

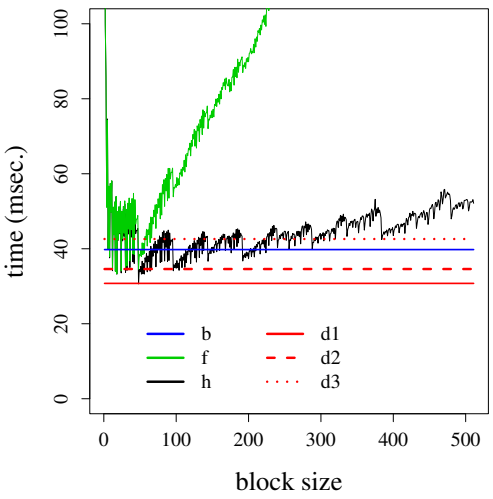

図 6 $m = n = 512$ の場合の，ブロック分割法と QR 分解の計算時間の詳細
Fig. 6 Detail of xecution time for computing QR decomposition when $m = n = 512$.

られたものより性能が劣っている．しかし，今回の model2 については，図 6 を見る限りでは，大きく性能が劣っているわけではなく，一方で，表 2 から分かるようにチューニングの時間が大幅に削減できている．ただし，model3 のようにチューニングの時間は削減できても，最適化の精度も大きく低下する可能性も考えられる．

これらの結果から，適切な性能モデルを用いて予測を行えば，チューニングの時間を削減しながら，十分な最適化を行える可能性がある．ただし，性能モデルの構築（例えば，今回のケースではサンプル点の設定）は経験的であるため，今後，与えられた環境において，性能モデルを非経験的に構築する手法が必要と言える．

6. おわりに

本稿では，ハウスホルダー変換を用いた QR 分解アルゴリズムのブロック化における，ブロック分割法の自動チューニングを考えた．特に，我々がこれまでに構築した手法の内部で使用する性能モデルに注目し，いくつかのモデルの比較を行った．数値実験の結果，適切なモデルを用いれば，チューニングに要する時間を現実的な範囲に抑えつつ，十分な最適化が行える可能性があることが確認できた．一方で，モデルによっては，チューニングの精度が大幅に低下してしまう危険性も示された．

大規模な行列計算を想定した場合，性能モデルによる予測は必要不可欠である．今後は，今回使用した，線形補間以外の予測方法（例えばスプライン補間など）を用いて実験すると同時に，チューニングに適した性能モデルを非経験的に構築する手法について研究を行うことが課題である．

謝辞 日頃からご議論頂いている名古屋大学大学院工学研究科計算理工学専攻の張研究室の皆様，自動チューニングの研究に関して有益な助言を頂いている自動チューニング研究会の皆様に感謝いたします．なお，本研究は日本学術振興会特別研究員奨励費および科学研究費補助金を補助を受けている．

参 考 文 献

1) Naono, K., Teranishi, K., Cavazos, J. and Suda, R.: *Software Automatic Tuning*, Springer, 1 edition (2010).
2) Fukaya, T., Yamamoto, Y. and Zhang, S., L.: A Dynamic Programming Approach to Optimizing the Blocking Strategy for the Householder QR Decomposition, *Proceedings of IEEE Cluster 2008*, pp.402–410 (2008).
3) 深谷 猛，山本有作，張 紹良：動的計画法を用いたブロックハウスホルダー QR 分解アルゴリズムの性能最適化，情報処理学会論文誌：コンピューティングシステム，Vol.35 (採録決定).
4) Golub, G. and VanLoan, C.: *Matrix Computations*, Johns Hopkins University Press, 3 edition (1996).
5) Schreiber, R. and Van Loan, C.: A storage-efficient WY representation for products of Householder transformations, *SIAM Journal on Scientific and Statistical Computing*, Vol.10, pp.53–57 (1989).
6) Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Croz, D., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S. and Sorensen, D.: *LAPACK User's Guide*, SIAM (1992).
7) Elmroth, E. and Gustavson, F.: Applying Recursion to Serial and Parallel QR Factorization Leads to Better Performance, *IBM Journal of Research and Development*, Vol.44, p.605 (2000).