情報処理学会論文誌　コンピューティングシステム　Vol. 4　No. 4　146–157 (Oct. 2011)

動的計画法を用いたブロックハウスホルダ QR 分解アルゴリズムの性能最適化

深谷　猛[†1]　山本有作[†2]　張　紹良[†1]

密行列計算においては，高性能化のためにアルゴリズムのブロック化が必須である．その際に，ブロック化の方法次第で性能が大きく変化するため，その最適化が重要な課題となっている．しかしながら，ブロック化の自由度が大きいため，従来は限定された範囲内で最適化を行うことがほとんどである．本論文では，QR 分解アルゴリズムを対象として，二分木を使うことで従来より格段に広いクラスのブロック化の方法を系統的に扱い，その中から動的計画法により最適なブロック化の方法を決定する手法を提案する．数値実験の結果，提案手法がブロック分割法に対する自動チューニング手法として有望であることが示された．

Performance Optimization for the Blocked Householder QR Decomposition Using the Dynamic Programming

Takeshi Fukaya,[†1] Yusaku Yamamoto[†2] and Shao-Liang Zhang[†1]

Blocking techniques are widely used in high performance matrix computations. When using them, it is important to optimize a blocking way, which influences the performance of computations. However, because of the high degree of freedom in blocking techniques, such optimization is generally done in a limited class of blocking ways. In this paper, we propose a framework to determine the efficient blocking way for the algorithm of QR decomposition. In our framework, various kinds of blocking ways are represented systematically with binary trees and an optimal one is determined by dynamic programming. Results of numerical experiments show that our framework has good possibilities in the view of the automatic performance tuning.

1. はじめに

計算機環境の多様化・複雑化にともない，高性能計算を行うにはソフトウェアのチューニングが不可欠となっている．しかし，パラメータの自由度が大きかったり，専門知識や経験が必要だったりと人の手で効果的なチューニングをすることが困難な場合が多くなっている．そのような背景から，計算機自身が機械的にチューニングを行う自動チューニングの研究が注目されている[1)]．

我々が研究を行っている密行列計算アルゴリズムのブロック化の方法を最適化する問題[2),3)]では，ブロック化の際の自由度が非常に大きいため，従来は自由度を減らして数個のパラメータのみの最適化を行うことが一般的であった．そこで，我々は，基本的な密行列計算の 1 つであるハウスホルダ QR 分解のブロック化について，従来より格段に広いクラスから最適なブロック化の方法を機械的に探し出すことを目指す．本論文では，この目的を達成するために，二分木を用いてブロック化の方法を系統的に扱い，動的計画法により最適な二分木を機械的に決定する手法を提案する．そして，数値実験により，提案手法を評価するとともに，より実用的な手法の構築に向けた課題を明らかにする．

本論文の構成は以下のとおりである．まず，2 章で QR 分解とそのブロック化について説明をする．3 章で本研究の目的を述べたうえで，4 章で動的計画法を用いた最適化方法の詳細を説明する．5 章で数値実験の結果を示して，その考察を行う．6 章で関連研究を紹介して，最後に 7 章でまとめと今後の課題を述べる．

2. ハウスホルダ QR 分解とブロック化

本章では，ハウスホルダ変換を用いた QR 分解とそのブロック化について説明する．

2.1 QR 分解

$m \times n$ $(m \geq n)$ の行列 A を

$$A = QR,$$

と分解することを QR 分解と呼ぶ[4)]．ここで，Q は $m \times m$ の直交行列，R は $m \times n$ の上

†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

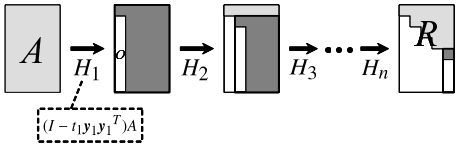


図 1　ハウスホルダ変換による QR 分解の計算
Fig. 1　QR decomposition using the Householder transformations.

三角行列である．ただし，Q を行列として陽的に求めることはしない場合も多い．

2.2　ハウスホルダ変換を用いた QR 分解

ハウスホルダ変換は，スカラー t とベクトル $\boldsymbol{y}$ により，

$$H := I - t\boldsymbol{y}\boldsymbol{y}^T,$$

と書かれる H を用いた直交変換である．ハウスホルダ変換による QR 分解の計算[4] では，**図 1** に示したように，

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

と行列 A にハウスホルダ変換を逐次的に作用させて R を得る．なお，Q はハウスホルダ変換の積として $\boldsymbol{y}_i$，t_i（$i = 1, \ldots, n$）の形で保持することが一般的であり，本研究でも同様とする．

この計算の大半はハウスホルダ変換を行列に作用させる部分で，

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

と行われる．上記の 2 種類の演算はともに Level-2 BLAS[5] で，近年の計算機では高い実効性能を得ることが難しい．そのため，高性能計算を行う際には次節で述べるブロック化を行うことが一般的となっている[6]．

2.3　ハウスホルダ QR 分解のブロック化

複数（ここでは p 個）のハウスホルダ変換の積は，

$$(I - t_p\boldsymbol{y}_p{\boldsymbol{y}_p}^T)\cdots(I - t_1\boldsymbol{y}_1{\boldsymbol{y}_1}^T) = I - YTY^T, \quad (1)$$

と書くことができる[7]．ここで，$Y = [\boldsymbol{y}_1 \ldots \boldsymbol{y}_p]$，$T$ は対角要素が $t_1, \ldots, t_p$ の下三角行列で，その狭義下三角部分は $\boldsymbol{y}_i$，t_i（$i = 1, \ldots, p$）から計算される．

式 (1) を用いてハウスホルダ QR 分解をブロック化した場合の計算は**図 2** に示したように，

(1)　$A \to [A_1 A_2]$（計算対象の行列を分割）
(2)　$(I - Y_1 T_1 {Y_1}^T)A_1 = R_1$（$A_1$ の QR 分解）

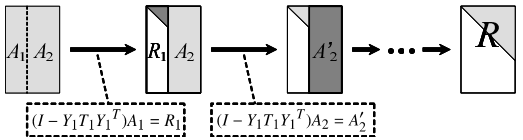


図 2　ブロック化されたハウスホルダ QR 分解の計算
Fig. 2　Blocked Householder QR decomposition.

(3)　$(I - Y_1 T_1 {Y_1}^T)A_2 \to A_2'$（$A_2$ の更新）

という流れで行われる[4),6)]．この結果，(3) の計算で高い実効性能が期待できる行列乗算の使用が可能になる．一方，(2) で T_1 の計算が必要となるため，ブロック化しない場合と比べて演算量が増加する．(2) の計算方法は主に次の 2 つがある．

(a) 2.2 節で述べた方法を用いて，

$$(I - t_p\boldsymbol{y}_p{\boldsymbol{y}_p}^T)\cdots(I - t_1\boldsymbol{y}_1{\boldsymbol{y}_1}^T)A_1 = R_1,$$

と A_1 の QR 分解を計算して，$\boldsymbol{y}_i$，t_i（$i = 1, \ldots, p$）から T_1 を計算する．

(b) 再帰的にブロック化アルゴリズムを用いて，

$$(I - Y_{1q}T_{1q}{Y_{1q}}^T)\cdots(I - Y_{11}T_{11}{Y_{11}}^T)A_1 = R_1,$$

と A_1 の QR 分解を計算して，Y_{1i}，T_{1i}（$i = 1, \ldots, q$）から T_1 を計算する．

(a) の計算は 2.2 節で説明したように Level-2 BLAS で行うことになるが，(b) の場合は行列乗算の使用が可能になる．ただし，(b) は (a) に比べて T の計算回数が多いため，演算量が多くなる．

ブロック化されたハウスホルダ QR 分解の計算では，ハウスホルダ変換の合成にともなう演算量の増加よりも行列乗算の使用による性能向上の方が大きくなるようにブロック化を行うことで，QR 分解の計算の高性能化が実現できる．

2.4　ブロック分割法の最適化

2.3 節で述べたように，ブロックハウスホルダ QR 分解の計算を高性能化するためには，適切なブロック化を行うことが重要な課題となる．ブロック化における自由度は以下の 2 種類がある．

- **行列を分割する際の分割幅**
  行列サイズが大きくなるほど行列乗算の実効性能の向上が期待できるが，T の計算コストも増加．
- **部分的な QR 分解の計算方法**
  再帰的にブロック化を行うほど行列乗算で計算する部分の割合が増加するが，T の計算

回数も増加.

ここで，行列乗算を含む BLAS ルーチン[5] の実効性能は計算機環境（BLAS ライブラリの実装を含む）ごとに異なる．また，T の計算コストは対象の行列サイズ（特に m）によって変化する．

本論文では上記の 2 つを合わせて**ブロック分割法**と呼び，使用する計算機環境と対象とする問題サイズに応じてブロック分割法を最適化する問題について検討する．

3. 研 究 目 的

2.4 節で述べたようにブロック分割法を最適化する際の自由度は非常に大きく，これを人間の手で扱うのは困難といえる．そのため，現状ではブロック分割法の自由度を削減して，ブロック幅等の数個のパラメータの最適化を行うことが一般的である．一方，近年注目されている自動チューニング[1] では，計算機自身がパラメータの最適化を機械的に行うことを目指しているので，今回のような膨大な自由度も処理できる可能性がある．

そこで，本論文では，QR 分解におけるブロック分割法の自由度をできるだけ維持したうえで，その最適化を機械的に行う仕組みを提案し，実際に得られた分割法に対して性能評価を行う．

4. 動的計画法によるブロック分割法の最適化

本章では，本論文で提案する，動的計画法を用いてブロック分割法を最適化する手法について説明する．

4.1 二分木による分割法の表現

計算機上でブロック分割法を扱うためには分割法を何らかの形で系統的に扱うことが必要である．そこで，本研究では**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．

以下の条件を満たす二分木を考える．

- 根のノードの数字は計算対象の行列の列の数とし，丸で囲む
- 2 つの子ノードの数字の和はそれらの親ノードの数字と一致
- 葉のノードの数字はすべて 1
- 丸で囲まれたノードの左の子ノードは四角，右の子ノードは丸
- 四角で囲まれたノードの子ノードはともに四角

各ノードの数字をそのノードで計算対象とする部分行列の列の数に対応させる．一方，行の数は二分木の根の部分で与えられて，その後は列の数が決まることで一意に決まっていくので特に明記しない．また，各ノードの丸と四角の囲みをそれぞれ，

- **丸**：Q を $\boldsymbol{y}_i$，t_i で求める QR 分解
- **四角**：$Q = I - YTY^T$ の形の QR 分解

に対応させる．

二分木の各ノードで行う計算は以下のようにする．

- **丸で囲まれたノード**

  親ノードで扱う行列のサイズを $m \times n$ とした場合の計算手順を**図 4** に示す．

  step.0 $m \times n$ の行列を列方向に $n_1 : n_2$ に分割する．

  step.1 n_1 以下の部分木に従ってブロック分割を行い，左側の $m \times n_1$ の部分行列を

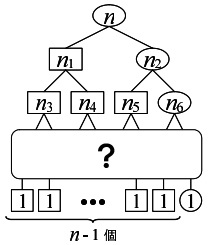


図 3　二分木による表現
Fig. 3　Representation with a binary tree.

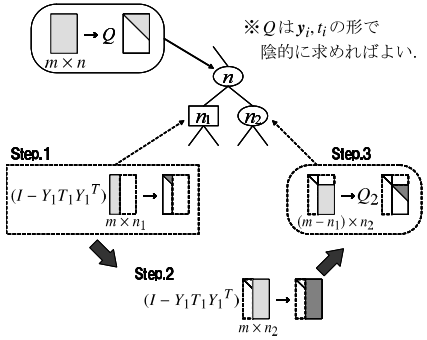


図 4　二分木のノードで行う計算手順（1）
Fig. 4　Computation on each node of a binary tree (1).

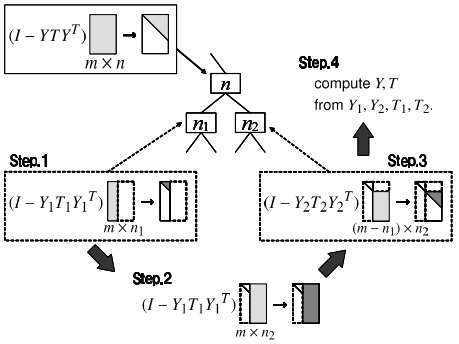


図 5　二分木のノードで行う計算手順（2）
Fig. 5　Computation on each node of a binary tree (2).

QR 分解する．

step.2 step.1 で得られた Y_1，T_1 を使って，右側の部分行列を更新する．

step.3 n_2 以下の部分木に従ってブロック分割を行い，右側の部分行列の一部（サイズは $(m - n_1) \times n_2$）を QR 分解する．

- **四角で囲まれたノード**
  親ノードで扱う行列のサイズを $m \times n$ とした場合の計算手順を**図 5** に示す．step.0 から step.3 の計算は，step.3 の部分の QR 分解の Q の形が異なる以外は丸のノードの場合と同じである．最後に，
  step.4 step.1 と step.3 で得られた Y_1，Y_2，T_1，T_2 から Y，T を計算する．
- **葉のノード**
  数字が 1 になった場合は Y がベクトル，T がスカラーとなるので，丸と四角は同じ意味になる．ここでは，計算対象のベクトルを $\boldsymbol{a}$ として，
  $(I - t\boldsymbol{y}\boldsymbol{y}^T)\boldsymbol{a} = [r, 0, \ldots, 0]^T$,
  を満たす $\boldsymbol{y}$，t，r を求める．

葉の数が n の二分木の総数は 2^n 以上であることが知られている[8] ので，この方法により非常に広いクラスのブロック分割法を系統的に扱うことが可能となる．また，二分木を使って従来のブロック分割法を表現できるので，その代表例を**図 6** に示す．

- **non-blocking**
  ハウスホルダ変換をブロック化せず，1 本ずつ後ろの部分全体に作用させる．
- **fixed-size blocking**
  一定の幅（図では L）ごとにブロック化を行う．ブロック内の計算はブロック化しない．LAPACK[9] で用いられている手法である．
- **recursive bisection**
  再帰的に行列を半分に分割してブロック化を行う[10]．
- **hybrid blocking**
  fixed-size blocking のブロック内の計算に recursive bisection を用いる[10]．

4.2　目的関数の定式化

本節では，ブロック分割法を決定する際に用いる目的関数について説明する．

本研究では QR 分解の計算時間を短縮することを目指すので，

- $\mathfrak{b}_n$：葉の数が n の二分木
- $\mathfrak{B}_n$：葉の数が n の二分木の集合

として，$m \times n$ の行列の QR 分解（Q は $\boldsymbol{y}_i$，t_i）を，$\mathfrak{b}_n$ で表現されるブロック分割法で計算した際の計算時間を

$T_{\mathrm{QR}}(m, n, \mathfrak{b}_n)$

として，これを目的関数とする．つまり，

$$\mathfrak{b}_n^* := \arg \min_{\mathfrak{b}_n \in \mathfrak{B}_n} T_{\mathrm{QR}}(m, n, \mathfrak{b}_n) \tag{2}$$

が我々が求めるべき二分木となる．

なお，実際の計算機上での計算では，行列をメモリに格納する際の整合寸法も計算時間に影響を与える．また，直前の計算内容によってキャッシュメモリの状態が異なるため，計算時間はその影響を受ける可能性もある．ただし，本論文ではこれらの影響を考えずに計算時間は行列サイズとブロック分割法によって決まるとして議論をする．

4.3　動的計画法による最適化

本節では，動的計画法を用いて式 (2) を解くアルゴリズムについて説明する．

まず，T_{WY}，T_{App}，T_{Agg} を以下のように定義する．

- $T_{\mathrm{WY}}(m, n, \mathfrak{b}_n)$：
  $m \times n$ の行列の QR 分解（$Q = I - YTY^T$）の計算を，二分木 $\mathfrak{b}_n$ で表現されるブロック分割法を用いて行うときの計算時間
- $T_{\mathrm{App}}(m, n_1, n_2)$：
  $m \times n_2$ の行列に $I - YTY^T$（Y は $m \times n_1$，T は $n_1 \times n_1$）を作用させる時間．4.1 節

図 6　従来のブロック分割法を二分木で表現した例
Fig. 6　Binary trees which represents a conventional blocking ways.

で説明した step.2 の計算時間に相当する．

- $T_{\mathrm{Agg}}(m,n_1,n_2)$：
  $m\times n_1$ の Y_1，$(m-n_1)\times n_2$ の Y_2，$n_1\times n_1$ の T_1，$n_2\times n_2$ の T_2 から $m\times(n_1+n_2)$ の Y と $(n_1+n_2)\times(n_1+n_2)$ の T を計算する時間．4.1 節で説明した step.4 の計算時間に相当する．

すると，図 4，図 5 から分かるように，

$$\begin{aligned}T_{\mathrm{QR}}(m,n,\mathfrak{b}_n) &= T_{\mathrm{WY}}(m,n_1,\tilde{\mathfrak{b}}_{n_1})\\ &\quad + T_{\mathrm{App}}(m,n_1,n_2)\\ &\quad + T_{\mathrm{QR}}(m-n_1,n_2,\tilde{\mathfrak{b}}_{n_2}),\end{aligned}\tag{3}$$

$$\begin{aligned}T_{\mathrm{WY}}(m,n,\mathfrak{b}_n) &= T_{\mathrm{WY}}(m,n_1,\tilde{\mathfrak{b}}_{n_1})\\ &\quad + T_{\mathrm{App}}(m,n_1,n_2)\\ &\quad + T_{\mathrm{WY}}(m-n_1,n_2,\tilde{\mathfrak{b}}_{n_2})\\ &\quad + T_{\mathrm{Agg}}(m,n_1,n_2),\end{aligned}\tag{4}$$

の関係式が成り立っている．ただし，ここで，$\tilde{\mathfrak{b}}_{n_1}$，$\tilde{\mathfrak{b}}_{n_2}$ はそれぞれ n_1，n_2 を頂点とする左右の部分木とする．

$$T^*_{\mathrm{QR}}(m,n)=\min_{\mathfrak{b}_n\in\mathfrak{B}_n} T_{\mathrm{QR}}(m,n,\mathfrak{b}_n),$$
$$T^*_{\mathrm{WY}}(m,n)=\min_{\mathfrak{b}_n\in\mathfrak{B}_n} T_{\mathrm{WY}}(m,n,\mathfrak{b}_n),$$

として，式 (3) を用いると

$$\begin{aligned}&\min_{\mathfrak{b}_n\in\mathfrak{B}_n} T_{\mathrm{QR}}(m,n,\mathfrak{b}_n)\\ &\quad=\min_{1\le n_1\le n-1}\Big[\min_{\mathfrak{b}_{n_1},\mathfrak{b}_{n_2}}\big\{T_{\mathrm{WY}}(m,n_1,\mathfrak{b}_{n_1})\\ &\qquad + T_{\mathrm{App}}(m,n_1,n_2)+T_{\mathrm{QR}}(m-n_1,n_2,\tilde{\mathfrak{b}}_{n_2})\big\}\Big]\\ &\quad=\min_{1\le n_1\le n-1}\Big[\min_{\mathfrak{b}_{n_1}}\big\{T_{\mathrm{WY}}(m,n_1,\mathfrak{b}_{n_1})\big\}\\ &\qquad + T_{\mathrm{App}}(m,n_1,n_2)+\min_{\mathfrak{b}_{n_2}}\big\{T_{\mathrm{QR}}(m-n_1,n_2,\tilde{\mathfrak{b}}_{n_2})\big\}\Big]\end{aligned}$$

となり，

$$\begin{aligned}T^*_{\mathrm{QR}}(m,n)=\min_{1\le n_1\le n-1}\Big[&T^*_{\mathrm{WY}}(m,n_1)+T_{\mathrm{App}}(m,n_1,n_2)\\ &+T^*_{\mathrm{QR}}(m-n_1,n_2)\Big],\end{aligned}\tag{5}$$

が導かれる．なお，$n_2=n-n_1$ である．

同様にして式 (4) からは，

$$\begin{aligned}T^*_{\mathrm{WY}}(m,n)=\min_{1\le n_1\le n-1}\Big[&T^*_{\mathrm{WY}}(m,n_1)+T_{\mathrm{App}}(m,n_1,n_2)\\ &+T^*_{\mathrm{WY}}(m-n_1,n_2)+T_{\mathrm{Agg}}(m,n_1,n_2)\Big],\end{aligned}\tag{6}$$

が導かれる．

式 (5)，(6) はベルマン方程式と呼ばれる再帰方程式で，動的計画法を用いて解くことができる[11]．動的計画法では，再帰的に定義された最適値（T^*_{QR}，T^*_{WY}）をボトムアップの形で計算し，計算過程で得られた情報から最適解（$\mathfrak{b}^*_n$）を構成する．

動的計画法を用いて二分木を決定するアルゴリズムを Algorithm 1 に示す．このアルゴリズムでは親ノードがとりうるすべての数字に対して最適な子ノードへの分割の比率を $B_{\mathrm{QR}}.B_{\mathrm{WY}}$ に記録していくことで，陰に二分木の最適化を行っている．したがって，実際に QR 分解の計算を行う際には，得られた $B_{\mathrm{QR}}.B_{\mathrm{WY}}$ を再帰的に参照して行列の分割を行う

Algorithm 1 Dynamic Programming (DP)

Input: m, n
Output: $B_{\mathrm{QR}}[,], B_{\mathrm{WY}}[,]$
  for $h = 1$ to m **do**
    $T_{\mathrm{QR}}[h,1] = T_{\mathrm{WY}}[h,1] \leftarrow \mathrm{House}(h)$
  end for
  for $k = 2$ to n **do**
    for $h = k$ to m **do**
      $T_{\mathrm{QR}}[h,k] = T_{\mathrm{WY}}[h,k] \leftarrow +\infty$
      for $i = 1$ to $k-1$ **do**
        $t_{\mathrm{QR}} \leftarrow T_{\mathrm{WY}}[h,i] + T_{\mathrm{App}}(h,i,k-i)$
            $+T_{\mathrm{QR}}[h-i,k-i]$.
        $t_{\mathrm{WY}} \leftarrow T_{\mathrm{WY}}[h,i] + T_{\mathrm{App}}(h,i,k-i)$
            $+T_{\mathrm{WY}}[h-i,k-i] + T_{\mathrm{Agg}}(h,i,k-i)$.
        if $t_{\mathrm{QR}} < T_{\mathrm{QR}}[h,k]$ **then**
          $T_{\mathrm{QR}}[h,k] \leftarrow t_{\mathrm{QR}},\ B_{\mathrm{QR}}[h,k] \leftarrow i$
        end if
        if $t_{\mathrm{WY}} < T_{\mathrm{WY}}[h,k]$ **then**
          $T_{\mathrm{WY}}[h,k] \leftarrow t_{\mathrm{WY}},\ B_{\mathrm{WY}}[h,k] \leftarrow i$
        end if
      end for
    end for
  end for

（丸のノードの分割については B_{QR}，四角については B_{WY} を参照する）[⋆1]．

Algorithm 1 中で，House，T_{App}，T_{Agg} の 3 種類の値が必要となる．House(h) は 4.1 節で触れた二分木の葉のノードでの計算時間（ベクトルの長さが h）である．また，T_{App}，T_{Agg} はともに 3 つの引数でサイズが決まる行列に対するいくつかの BLAS ルーチン（主に GEMM と TRMM）の実行時間の和となっている．これらの値は，QR 分解におけるカーネルルーチンの実行時間に相当し，計算を行う環境（CPU や用いる BLAS ライブラリ等）によって異なる．したがって，実際に Algorithm 1 を実行する場合，これらの値は実行時間予測モデル等により計算することになる．

⋆1 たとえば，二分木のあるノード（丸）で行列サイズが $m \times n$ であった場合，$B_{\mathrm{QR}}[m,n]$ を参照して，その値（n' とする）に従って，$n \to n' + (n-n')$ と分割をする．次の段階は，行列サイズが $m \times n'$ の四角のノードと，$(m-n') \times (n-n')$ の丸のノードになるので，それぞれ，$B_{\mathrm{WY}}[m,n']$ と $B_{\mathrm{QR}}[m-n', n-n']$ を参照して，同様に分割をしていく．

4.4 ま と め

本論文では，膨大な数のブロック分割法を二分木を用いて表現し，動的計画法により最適な二分木の決定をする手法を提案した．この手法では，対象とする計算機環境での QR 分解内のカーネルルーチンの実行時間のみを用いて，分割法の決定を機械的に行うことが可能である．以下では，本手法の特徴について簡単にまとめる．

得られるブロック分割法の性能

提案手法により得られるブロック分割法の性能は，使用するカーネルルーチンの実行時間の予測精度に依存する．

カーネルルーチンの実行時間の予測が正確であるとすると，

- 既存の計算機環境では，従来の分割法を含めた膨大な数の二分木を選択の候補としているため，従来の分割法よりも QR 分解の計算時間を短縮する分割法を得ることが期待できる．
- 計算機環境の特性が大幅に変化しても，二分木の決定の際に経験的な情報を使っていないため，適切な分割法を得ることが可能といえる．

一方，カーネルルーチンの実行時間の予測精度が不十分な場合は，実際の計算において最適（もしくはそれに近い）分割法を必ずしも得られるとは限らない．

最適化に要するコスト

提案手法では，ある m，n に対して動的計画法を行うと，そのサイズ以下のすべての場合について，最適化な分割法を得ることになる．つまり，ライブラリのインストール時等に 1 度，最適化を実行すればよいということである．また，そのコストは，行列サイズとカーネルルーチンの実行時間の予測コストに依存する．

5. 数 値 実 験

実際に提案手法を用いてブロック分割法の最適化を行い，さらに，得られた分割法を用いて QR 分解を行うことで，提案手法の評価をする．

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

No.	項目	条件
1	CPU	Intel Xeon X5670 (2.93 GHz) 1 コアのみ使用
	メモリ	24 GB
	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
2	CPU	Intel Xeon X5355 (2.66 GHz) 1 コアのみ使用
	メモリ	16 GB
	OS	Red Hat Linux
	コンパイラ	Intel icc ver. 9.1（オプション -O3）
	BLAS	Intel MKL ver. 9.0

5.1　計算機環境

使用した計算機環境は**表 1** の 2 種類である．

5.2　実 験 方 法

まず，QR 分解のカーネルルーチンの実行時間の予測に以下の 3 種類のモデルを用いて，提案手法により分割法の最適化を行った．

model 1：実際にカーネルルーチンを実行して，その実行時間を用いる．

model 2：カーネルルーチン内で呼ばれる BLAS ルーチンの実行時間をサンプル点の値から多重線形補間により予測し，その値を足し合わせることでカーネルルーチンの実行時間を予測する．サンプル点は，BLAS ルーチン内のサイズパラメータ（DGEMM の場合は 3 つ，DTRMM の場合は 2 つ）を，それぞれ 1，2，4，8，16，32，64，128，256，512 と変化させて，すべての組合せとする．また，行列の転置等に関するパラメータが異なる場合は，それぞれに個別に予測を行う．

model 3：model 2 で，サンプル点を 1，3，9，27，81，243，512 と変化させた場合．

次に，以下のブロック分割法を用いて QR 分解の計算を行い，その計算時間を測定した．

b： recursive bisection

f： fixed-size blocking

h： hybrid blocking（f と b）

d1：model 1 を用いて提案手法により最適化された分割法

d2：model 2 を用いて提案手法により最適化された分割法

d3：model 3 を用いて提案手法により最適化された分割法

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

環境	モデル	サンプリング	動的計画法	合計
1	model 1	–	1.95×10^5	1.95×10^5
	model 2	5.81	374	380
	model 3	2.73	373	376
2	model 1	–	2.68×10^5	2.68×10^5
	model 2	11.6	208	220
	model 3	5.04	192	197

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．

また，実験に関する主な事項は以下のとおりである．

- 用いた行列は $[-0.5, 0.5]$ の乱数行列．
- QR 分解の計算時間は 10 回の平均を採用．
- 配列の整合寸法は行列サイズにかかわらず 512 とした[⋆1]．
- 最大の行列サイズ（Algorithm 1）は $m = n = 512$．
- 動的計画法の結果（B_{QR}，B_{WY}）はファイルに出力し，QR 分解の計算時にメモリ上に読み込み，それを参照して計算を進めた．
- d1，d2，d3 で動的計画法の結果を読み込む時間は計算時間に含めていない．

5.3　実 験 結 果

まず，提案手法による最適化の実行時間を**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．

次に，それぞれの計算機環境で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** と**表 4** に示す．また，それぞれに対して，3 つのモデル内で予測される QR 分解の計算時間もあわせて記載する．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．

$m = n = 512$ の場合について，と h のブロック幅を 1 刻みで変化させたときの，QR 分解の計算時間と各モデル内での予測時間の変化のグラフを**図 7**，**図 8** に示す．なお，比較のために f と h 以外の値もあわせてグラフに示した（ブロック幅に依存しないため，横軸に平行な直線で示した）．

⋆1 model 1 の予測精度をできるだけ高めるために，このような措置をとった．

表 3　計算機環境 1 における QR 分解の計算時間と各モデルでの予測時間（ミリ秒）　※括弧内の数字はブロック幅を示す
Table 3　Execution time and estiamted time for computing QR decomposition on the environment 1 (msec.).　※ Parenthetic number means the optimal block length.

		execution time (msec.)						estimeted time (msec.)																	
								in model 1						in model 2						in model 3					
m	n	b	f	h	d1	d2	d3	b	f	h	d1	d2	d3	b	f	h	d1	d2	d3	b	f	h	d1	d2	d3
128	128	1.42	1.13 (12)	1.12 (12)	1.10	1.53	1.41	2.75	2.60	2.49	1.77	2.42	2.54	8.78	5.16	6.94	5.73	4.00	4.23	6.39	5.34	5.85	5.29	4.74	4.68
256	128	2.48	2.05 (12)	2.04 (12)	2.03	2.35	2.66	3.24	3.03	2.88	2.33	2.84	3.14	9.11	5.49	7.24	5.76	4.54	4.77	6.14	5.08	5.86	4.85	4.59	4.47
256	256	6.97	5.58 (12)	5.47 (48)	5.32	6.66	7.28	9.01	7.92	7.57	6.23	8.10	8.78	21.1	13.6	17.2	14.5	11.5	12.0	16.5	14.8	15.5	14.0	13.5	12.9
384	128	3.59	3.01 (12)	2.96 (12)	2.92	3.29	4.00	4.42	3.91	3.86	3.20	3.75	4.49	10.2	6.59	8.39	7.21	5.86	6.44	7.42	6.39	7.20	6.20	6.03	5.83
384	256	10.4	8.61 (12)	8.29 (48)	8.20	9.16	11.1	11.9	10.3	10.1	8.75	10.2	12.1	23.8	16.3	20.0	17.4	14.4	15.6	18.9	18.3	18.4	16.3	17.1	15.4
384	384	17.8	15.5 (18)	14.5 (48)	14.4	16.9	20.2	20.7	18.5	17.4	15.4	18.7	22.0	35.4	27.1	32.6	29.1	24.6	25.9	32.9	32.9	32.8	30.3	29.3	28.1
512	128	4.77	3.93 (16)	3.89 (12)	3.77	4.15	5.08	5.71	4.83	4.96	4.10	4.76	5.70	11.4	7.26	9.48	8.31	6.88	7.45	8.68	7.52	8.42	7.54	7.38	7.08
512	256	13.8	11.8 (12)	11.3 (48)	11.1	12.2	14.7	15.6	13.6	13.3	11.7	13.4	15.9	27.5	19.8	23.6	21.1	18.0	19.7	23.0	23.3	22.5	21.0	22.2	19.6
512	384	24.1	21.8 (18)	20.4 (48)	20.2	22.3	28.9	26.8	24.8	23.1	21.0	24.0	30.5	42.0	33.2	38.8	35.5	30.8	32.8	39.1	42.1	40.2	37.7	37.3	34.8
512	512	39.8	33.3 (18)	30.6 (48)	30.8	34.6	42.6	43.2	37.4	34.6	31.8	36.8	44.4	67.6	49.6	56.8	52.0	45.5	49.8	59.8	65.9	61.7	58.5	55.9	52.5

表 4　計算機環境 2 における QR 分解の計算時間と各モデルでの予測時間（ミリ秒）　※括弧内の数字はブロック幅を示す
Table 4　Execution time and estiamted time for computing QR decomposition on the environment 2 (msec.).　※ Parenthetic number means the optimal block length.

		execution time (msec.)						estimeted time (msec.)																	
								in model 1						in model 2						in model 3					
m	n	b	f	h	d1	d2	d3	b	f	h	d1	d2	d3	b	f	h	d1	d2	d3	b	f	h	d1	d2	d3
128	128	1.81	1.56 (12)	1.63 (16)	1.67	1.85	2.01	2.00	1.80	1.80	1.49	1.98	2.07	18.4	4.20	18.2	7.34	2.54	2.54	20.4	5.18	20.2	9.00	3.02	2.92
256	128	2.70	2.38 (16)	2.48 (16)	2.49	2.78	2.99	2.62	2.50	2.41	2.19	2.78	2.87	19.3	4.22	19.1	9.58	3.41	3.49	21.3	5.16	21.1	11.7	4.11	3.96
256	256	7.73	6.40 (16)	6.57 (16)	6.54	7.03	7.68	7.47	6.55	6.36	5.73	6.96	7.47	40.9	10.2	39.8	19.0	8.46	8.60	45.4	13.1	44.9	23.5	10.4	10.2
384	128	3.57	3.13 (12)	3.26 (12)	3.17	3.69	3.93	3.48	3.25	3.28	2.89	3.62	3.81	20.0	5.70	13.5	9.83	4.31	4.41	22.3	6.99	16.5	11.9	5.14	5.09
384	256	10.7	9.01 (16)	9.11 (24)	9.04	9.67	10.2	9.98	9.00	8.81	8.12	9.53	9.81	43.5	12.8	29.8	23.3	11.2	11.5	48.5	16.4	36.3	28.6	13.7	13.4
384	384	20.3	16.7 (20)	16.6 (24)	16.7	18.3	19.5	19.0	16.3	15.9	14.7	17.4	18.2	49.9	22.2	47.1	36.8	20.0	20.5	59.7	28.2	58.1	45.2	24.3	23.8
512	128	4.35	3.89 (16)	4.01 (12)	3.89	4.44	4.57	4.23	4.05	4.10	3.61	4.43	4.58	20.9	5.73	14.3	11.3	5.19	5.19	23.1	6.89	17.3	13.6	5.82	5.80
512	256	13.7	11.6 (20)	11.6 (24)	11.5	12.5	13.0	12.7	11.7	11.3	10.6	12.3	12.6	46.3	14.9	32.4	26.1	14.0	14.3	51.5	18.1	39.2	31.6	16.5	16.5
512	384	26.2	22.3 (20)	22.1 (24)	21.9	24.7	25.7	24.3	21.7	21.0	19.7	23.2	24.3	55.7	28.0	52.6	44.1	25.6	26.5	65.8	34.8	64.3	53.8	30.5	29.8
512	512	43.3	35.0 (24)	34.6 (32)	34.6	37.1	39.9	39.3	33.3	32.0	30.3	34.4	37.0	107	43.9	103	63.5	39.9	40.6	119	52.5	116	76.6	46.8	45.7

最後に，$m = n = 512$ の場合について，提案手法で得られた二分木の一部分を**図 9**，**図 10** に示す．

5.4 考　　察

得られた実験結果について考察する．

5.4.1 model 1 を用いた最適化の結果について

まず，表 3，表 4 に示した QR 分解の計算時間と model 1 内での予測時間を比較すると，model 1 の予測精度が高いことが確認できる．特に，行列のサイズが大きい場合は，より精度の高い予測となっており，図 7，図 8 におけるグラフを比べてみても，model 1 での予測と実際の様子は非常に似たものになっていることが確認できる．また，誤差の原因の一部と

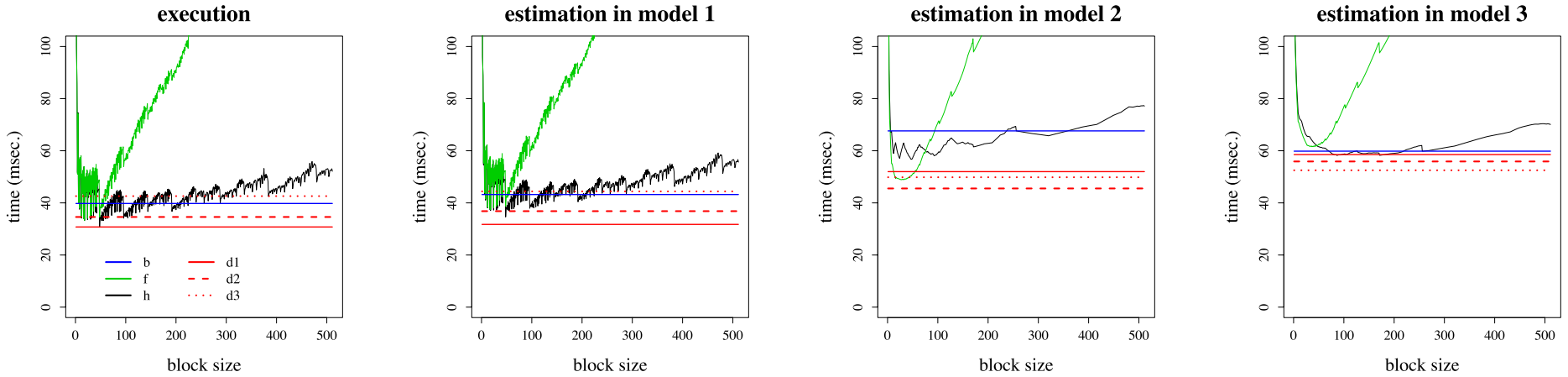


図 7　環境 1 で $m = n = 512$ の場合の，ブロック分割法と QR 分解の計算時間（あるいは予測時間）の詳細
Fig. 7　Detail of xecution time (and estimeted time) for computing QR decomposition when $m = n = 512$ on the environment 1.

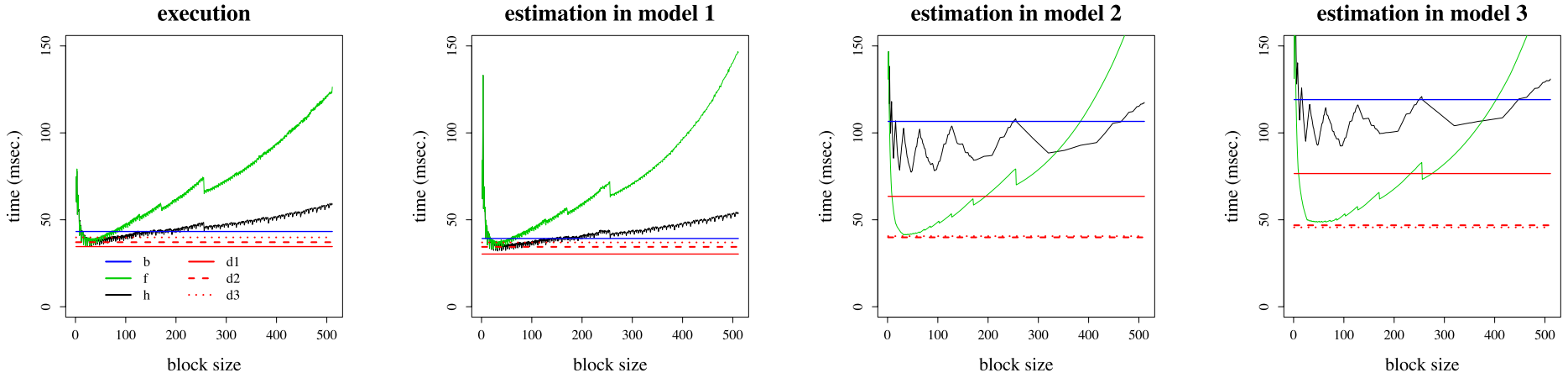
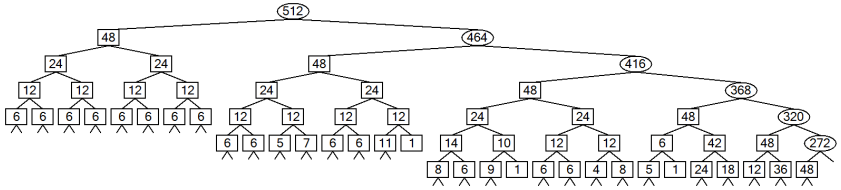
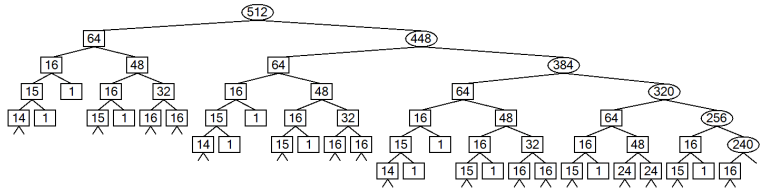


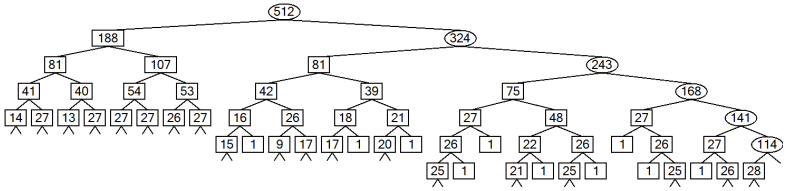
図 8　環境 2 で $m = n = 512$ の場合の，ブロック分割法と QR 分解の計算時間（あるいは予測時間）の詳細
Fig. 8　Detail of xecution time (and estimeted time) for computing QR decomposition when $m = n = 512$ on the environment 2.



(using model 1)

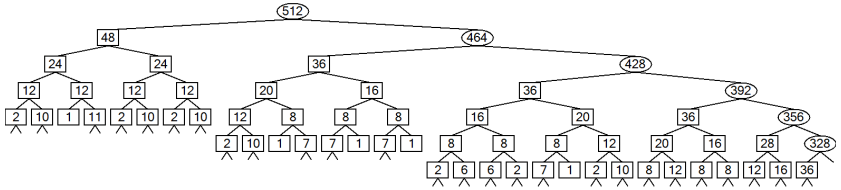


(using model 2)

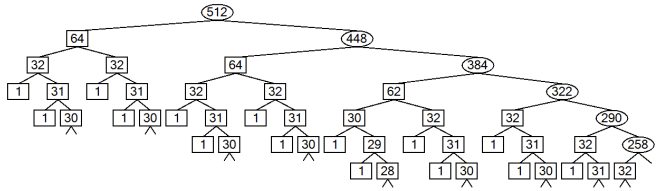


(using model 3)

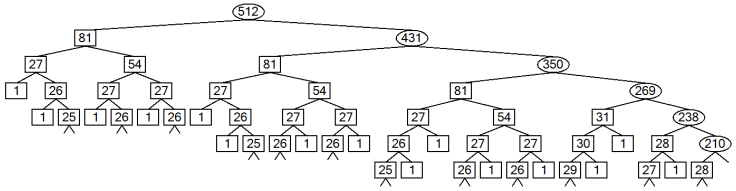
図 9　環境 1 で $m = n = 512$ の場合に対して得られた二分木
Fig. 9　Obtained binary trees for the case of $m = n = 512$ on the environment 1.



(using model 1)



(using model 2)



(using model 3)

図 10　環境 2 で $m = n = 512$ の場合に対して得られた二分木
Fig. 10　Obtained binary trees for the case of $m = n = 512$ on the environment 2.

して，キャッシュメモリの影響が考えられる．このモデルではカーネルルーチンを単独で実行して計算時間を測定しているのに対して，実際の計算では，QR 分解の一連の計算の中で同じ計算をしている．そのため，両者はキャッシュメモリの状態が異なり，その結果，実行時間にずれが生じている可能性がある．ただし，誤差の原因については，詳しい調査が必要である．

得られた分割法の性能は，表 3，表 4 の結果から，従来の分割法（b，f，h）の中で最適なものと同等であることが分かる．これは，model 1 がかなり精度の高いモデルであったため，モデル内で最適であった分割法が，実際の計算においても，最適に近いものになっていたからであると考えられる．

また，model 1 の予測精度を考慮すると，図 7，図 8 における model 1 のグラフから，実際の計算においても，従来の分割法（特に h）の性能が，そもそも非常に高いといえる．実際，図 9，図 10 に示した二分木の構造を見てみると，h の分割法に近い構造となっている．このことから，今回，実験で使用した環境では，理想的なモデルを用いて提案手法で最適化を行ったとしても，劇的な性能向上が見込めないことが予想される．

以上の点と，表 2 に示した最適化に要したコストから，このままの形での実用化は難しいといえる．

5.4.2　model 2，3 を用いた最適化の結果について

まず，表 3，表 4 の結果から，model 2 と 3 の両者とも，予測精度が高いとはいえない．図 7，図 8 のグラフも，実際の計算時間の様子と離れたものになっている．ただし，グラフを見ると f の予測は他と比べて比較的正しく予測できていることが確認できるので，その理

由を調査することが必要である．

実際に得られた分割法も，表 3，表 4 の結果から，従来の分割法の中で最適なものと比べて劣っていることが確認できる．ただし，図 7，図 8 のグラフを見ると，model 2 を使った場合に得られた分割法の性能は，従来の分割法に比べて大幅に劣っているわけではないことが分かる．

model 2 と 3 を比較すると，model 2 を使った最適化で得られた分割法の性能が，model 3 の場合よりも優れていることが，表 3，表 4 から読み取れる．これは，表 3，表 4 から分かるように，model 2 の方が予測精度が高くなっているからであると思われる．

なお，表 2 に示した最適化に要したコストについては，最適化を 1 度だけ実行すればよいことを考えると，実用上，許容範囲であるといえる．

5.4.3　今後の課題について

上記の考察から分かるように，カーネルルーチンの実行時間の予測モデルの精度と，提案手法により得られるブロック分割法の性能との間には強い関係がある．しかし，両者の間にどのような関係があるかについては，詳細がまだ明らかになっていない．

BLAS に代表されるカーネルルーチンに対して，精度の高い実行時間の予測方法を構築することは，本研究に限らず，高性能計算において重要な課題となっている．そのため，精度の高い性能予測モデルを構築する系統的な手法，つまり，性能予測モデルに対する自動チューニング手法等の開発が重要な課題となる．

また，本研究で提案した動的計画法の実行コストは，カーネルルーチンの実行時間の予測コストを定数とすると，$O(mn^2)$ となっている．これを削減するための工夫として，今まで得られた知識（たとえば，ハイブリッド型の分割法が優れている）の利用等を検討する必要がある．

6. 関 連 研 究

Bischof らにより fixed-size blocking のブロック幅を動的計画法を用いて可変にすることで性能が向上することが報告されている[12]．ただし，再帰的にブロック分割することは考慮されていない．

一方で，FLAME と呼ばれる研究では再帰的に行列を 2 分割することで，様々なアルゴリズムのバリエーションが表現できることが報告されている[13]．しかし，その分割の仕方をどのように決定すべきか，については特に触れられていない．

7. お わ り に

本研究では，ハウスホルダ変換による QR 分解をブロック化して計算する際のブロック分割法を，できるだけ分割の自由度を維持したうえで，機械的に最適化する仕組みを構築することを目指した．そして，

(1)　二分木を用いて非常に広いクラスのブロック分割法を系統的に扱う，

(2)　動的計画法を用いて最適な二分木を決定する，

という手順により，QR 分解におけるカーネルルーチンの実行時間予測モデルのみを用いて，機械的に二分木を決定する仕組みを提案した．

数値実験により，用いる予測モデルの精度が十分高ければ，従来，経験的に良いとされていたブロック分割法の性能と同等の性能を示す分割法を機械的に発見できることが確認できた．一方で，予測モデルの精度が十分でない場合は，得られた分割法の性能も十分高いとはいい難い．そのため，今後の課題として，

- 提案手法内で用いる予測モデルの精度と得られる分割法の性能の関係についての詳しい調査，
- できるだけ少ないコストで精度の高い予測モデルを構築する手法の開発，

といったことがあげられる．

謝辞　本論文を丁寧にお読みくださり，有益な助言をくださった査読者の方々に感謝いたします．また，日頃からお世話になっている名古屋大学大学院工学研究科の張研究室の皆様，自動チューニング研究会の皆様に感謝いたします．

本研究は科学研究費補助金特別研究員奨励費（課題番号：22・8599），基盤研究（A）（課題番号：23240005），基盤研究（B）（課題番号：21300013），基盤研究（C）（課題番号：21560065），新学術領域研究（課題番号：22104004）の補助を受けている．

参 考 文 献

1) Naono, K., Teranishi, K., Cavazos, J. and Suda, R.: *Software Automatic Tuning*, 1st edition, Springer (2010).
2) Fukaya, T., Yamamoto, Y. and Zhang, S.L.: A Dynamic Programming Approach to Optimizing the Blocking Strategy for the Householder QR Decomposition, *Proc. IEEE Cluster 2008*, pp.402–410 (2008).
3) 深谷　猛，山本有作，張　紹良：密行列計算アルゴリズムに対するブロック分割法の最適化と性能評価，情報処理学会研究報告，Vol.2010, No.33, pp.1–6 (2010).

4) Golub, G. and Van Loan, C.: *Matrix Computations*, 3rd edition, Johns Hopkins University Press (1996).
5) Dongarra, J., Du Croz, J., Hammarling, S. and Duff, I.: A Set of Level 3 Basic Linear Algebra Subprograms, *ACM Trans. Mathematical Software* (*TOMS*), Vol.16, pp.1–17 (1990).
6) Dongarra, J., Duff, I., Sorensen, D. and van der Vorst, H.: *Numerical Linear Algebra for High-Performance Computers*, 2nd edition, SIAM (1998).
7) 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).
8) Stanley, P.: *Enumerative Combinatorics*, 2nd edition, Cambridge Univ. Press (1999).
9) 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).
10) 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).
11) Bellman, R.: *Dynamic Programming*, Dover Publications (2003).
12) Bischof, C. and Lacroute, P.: An Adaptive Blocking Strategy for Matrix Factorizations, *Proc. Joint International Conference on Vector and Parallel Processing*, pp.210–221 (1990).
13) Bientinesi, P., Gunnels, J., Myers, M., Quintana-Orti, E. and van de Geijn, R.: The Science of deriving Dense Linear Algebra Algorithms, *ACM Trans. Mathematical Software*, Vol.31, No.1, pp.1–26 (2005).

(平成 23 年 1 月 28 日受付)
(平成 23 年 5 月 25 日採録)

深谷　　猛（学生会員）

2009 年名古屋大学大学院工学研究科計算理工学専攻（博士前期課程）修了．現在，同大学院工学研究科計算理工学専攻（博士後期課程）在学中．2010 年 4 月より日本学術振興会特別研究員（DC2）．行列計算の高性能アルゴリズムとその性能最適化や自動チューニングの研究に従事．日本応用数理学会会員．

山本　有作（正会員）

1992 年東京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．2001〜2002 年コロンビア大学ビジネススクール客員研究員．2003 年名古屋大学大学院工学研究科助手．同年名古屋大学博士（工学）．同講師，助教授，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教授．並列計算機向け行列計算アルゴリズムおよび金融工学向け高速計算アルゴリズムの研究開発に従事．日本応用数理学会，SIAM，INFORMS 各会員．

張　　紹良（正会員）

1990 年筑波大学大学院工学研究科博士課程修了．工学博士．東京大学大学院工学系研究科助教授を経て，2005 年より名古屋大学大学院工学研究科教授．大規模行列計算における反復解法の開発および並列計算アルゴリズムの研究に従事．日本応用数理学会，SIAM 各会員．