実対称固有値問題に対する多分割の分割統治法の SR11000 への一実装

桑島豊，坪谷怜，田村純一，重原孝臣

埼玉大学理工学研究科

1. はじめに

本稿でとり扱う固有値問題は，量子力学，量子化学，建築，経済など様々な分野で現れる問題である。量子力学，量子化学においてはシュレーディンガー方程式として陽に現れ，建築の分野においては建物の振動を扱う場合などに，経済の分野では市場の安定性に関する指標を与えるためなどに現れ，これら以外の分野においてもその他多様な場面で必要となる。また，基本的な統計解析手法として広く用いられている主成分分析は，固有値計算が主要な部分を占める。大規模な問題を扱う分野も多く，解法の高速化の需要は大きい。そのため，並列計算機を有効に活用できる固有値問題解法が重要である。

固有値問題解法の 1 つである 2 分割の分割統治法は，1 つの CPU 上であっても高速な解法であり，また本質的に並列計算に向く性質を持つ。その 2 分割の分割統治法を，本質的な並列性を損なうことなく，演算回数を減らすことにより，さらなる速度の向上を目指すべく拡張したアルゴリズムが，多分割の分割統治法である。このアルゴリズムは，未だ解決すべき点を残すものの，逐次計算では，LAPACK [*1]に実装されている 2 分割の分割統治法と同等程度の速度で計算できるという結果が得られている。

本稿では，我々が提案した「実対称行列の固有値問題に対する多分割の分割統治法」の並列計算機への実装の第一段階として，SR11000 の 1 ノード上における実装について紹介する。SR11000 において，共有メモリ型の並列化による実装を行い，重点的に並列化を行い効果をあげた箇所や，実行速度による他の固有値問題解法との比較など，現状で得られている結果を報告する。

本原稿の構成は以下の通り。2 節では，実対称固有値問題の数値解法の流れと，代表的な数値解法について述べる。2 節で述べた数値解法の中で本稿で主に取り扱う分割統治法を，3 節では 2 分割について，4 節では多分割について少し詳細に説明する。5 節では並列化の方法の種類と，本研究で用いた並列化の方法について述べる。6 節では数値実験の方法，結果とその考察について述べる。7 節はまとめを行い，8 節で今後の課題について述べる。9 節では，実験中に遭遇した事例などを紹介する。

2. 固有値問題の数値解法

本節では，まず固有値問題を数学的側面から紹介し，その後一般的な固有値問題の数値解法の流れを説明し，代表的な数値解法の簡単な紹介とそれぞれの比較を行う。

[*1] 標準的な行列計算ライブラリの一つ。ソースが無料で公開されている (www.netlib.org/lapack/)。SR11000 では，最適化されたライブラリが提供されている。

2.1 固有値問題の数学的な定式化

この小節では，以降必要となる用語の説明を兼ねて，固有値問題を数学的に述べる。

行列に対する固有値問題とは，n 次複素正方行列 A に対して，

$$A\boldsymbol{q} = \lambda\boldsymbol{q}, \quad \boldsymbol{q} \neq \boldsymbol{o} \tag{1}$$

を満たす複素数 λ と n 次複素ベクトル $\boldsymbol{q}$ の対を求める問題である。λ は固有値，$\boldsymbol{q}$ は固有ベクトルと呼ばれる。

特に，A を実対称行列に限れば，固有値として（重複を含めて）n 実数 $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ を持ち，それぞれに対応して $A\boldsymbol{q}_j = \lambda_j\boldsymbol{q}_j$, $(\boldsymbol{q}_i, \boldsymbol{q}_j) = \delta_{ij}$ を満たす固有ベクトル $\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_n$ が存在する。ただし，δ_{ij} はクロネッカのデルタで，$i = j$ なら 1，$i \neq j$ なら 0 である。

これらを行列形式にまとめると，直交行列 $Q = (\boldsymbol{q}_1\ \boldsymbol{q}_2\ \cdots\ \boldsymbol{q}_n)$，実対角行列 $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$ を用いて，$AQ = Q\Lambda$ であり，従って $A = Q\Lambda Q^T$ という対角化が得られる。逆に，直交行列 $Q = (\boldsymbol{q}_1\ \boldsymbol{q}_2\ \cdots\ \boldsymbol{q}_n)$ と実対角行列 $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$ を用いて，実対称行列 A が $A = Q\Lambda Q^T$ と分解できれば，Λ の対角成分 λ_j が A の固有値で，Q の各列ベクトル $\boldsymbol{q}_j$ が対応する固有ベクトルである。

2.2 実対称行列の実対称三重対角行列化

実対称行列の固有値問題を数値的に解く場合には，実対称行列のまま計算を開始することは実用的でなく，前処理を行い，行列を簡単化した後に固有値計算を行う。一般的な前処理は，実対称行列を実対称三重対角行列に相似変換するというもので，三重対角化と呼ばれる。実対称三重対角行列とは，対角成分とその両隣の成分以外が全て 0 の行列である。

三重対角化には，**ハウスホルダー法**が数値的に安定で，よく用いられる。ハウスホルダー法は，ハウスホルダー行列 $H = I - 2\boldsymbol{u}\boldsymbol{u}^T$, $(\|\boldsymbol{u}\|_2 = 1)$ を用いる方法である。$H = H^T$, $H^TH = H^2 = I$ より，H は対称な直交行列である。ただし，I は単位行列。3 次実対称行列

$$A = \begin{pmatrix} a_1 & b_1 & c_1 \\ b_1 & a_2 & b_2 \\ c_1 & b_2 & a_3 \end{pmatrix} \tag{2}$$

に対して三重対角化を行う場合は，ハウスホルダー行列を $H = I - 2\boldsymbol{u}\boldsymbol{u}^T$,

$$\boldsymbol{u} = \frac{\boldsymbol{u}'}{\|\boldsymbol{u}'\|_2}, \quad \boldsymbol{u}' = \begin{pmatrix} 0 \\ b_1 - b_1' \\ c_1 \end{pmatrix}, \quad b_1' = \sqrt{b_1^2 + c_1^2} \tag{3}$$

と選ぶことで，

$$T = HAH = \begin{pmatrix} a_1 & b_1' & 0 \\ b_1' & a_2' & b_2' \\ 0 & b_2' & a_3' \end{pmatrix} \tag{4}$$

と三重対角化される。4 次以上の実対称行列に対しても，これと同様のハウスホルダー行列による変換を繰り返すことで，三重対角化することができる。

以下で紹介する実対称固有値問題解法は，全て実対称三重対角行列に対して適用される。

表 1　実対称三重対角行列の固有値問題解法の比較

解法	1) 理論	2) 精度	3) 演算量	4) 並列性	5) 行列積	6) 領域
2 分割の分割統治法	○	○	$O(n^2)$ $\sim (4/3)n^3$	○	○	$O(n^2)$
QR 法	○	○	$6n^3$	×	×	$O(n)$
二分法・ 逆反復法	○	○	$O(n^2)$ $\sim O(n^3)$	×	×	$O(n)$
MRRR 法	△	△	$O(n^2)$	△	×	$O(n)$
k 分割の分割統治法	△	○	$O(n^2)$ $\sim (2k/(k^2-1))n^3$	○	○	$O(n^2)$

2.3　代表的な実対称三重対角行列の固有値問題の数値解法の紹介と比較

本節では，代表的な実対称三重対角行列の固有値問題の数値解法を紹介し，それらの解法と多分割の分割統治法をいくつかの観点から比較を行う。

代表的な解法として，本稿で取り扱う分割統治法と，その他に MRRR 法，QR 法，二分法・逆反復法を挙げる。MRRR 法は比較的新しい解法であり，QR 法，二分法・逆反復法は古典的な方法である。

2.3.1　比較の観点

それぞれの観点の意味づけは以下の通り。

1) 理論は，解法に理論的，数学的に確立しているかを意味している。

2) 精度は，各解法で求められる結果の精度を意味している。無限精度演算であれば 0 になるべき，相対残差と直交誤差を評価基準とした。

3) 演算量は，n 次行列の問題を解く際に必要な浮動小数点演算の理論的な回数を表している。現在の計算機において，この値が速度と比例するとは限らない。

4) 並列性は，解法が並列計算機に向いた性質を持っているかを表している。

5) 行列積は，解法に行列積演算が使用されているかを表し，6) 領域は，n 次行列の問題を解く際に必要なメモリ領域を表している。この 2 つは密接な関連があり，行列積が必要な場合には $O(n^2)$ のメモリ領域が必要となる。しかし，並列計算機を含む現在の計算機において，行列積演算は高速に計算可能である。このことは，分割統治法を紹介する上で大きな意味を持つため，次の小々節で少し詳しく述べる。

また，2.3.3 小々節からは，以上の観点を含めて，表 1 にある 5 つの解法を紹介する。

2.3.2　行列積

行列同士の積を，本稿では行列積と呼ぶ。行列積は，本稿で取り扱う固有値問題に対する分割統治法において，演算回数の面で支配的であり，この計算の速度が分割統治法の速度に直結する。

a_{ij} を (i,j) 成分に持つ n 次正方行列 A と，b_{ij} を (i,j) 成分に持つ n 次正方行列 B との積

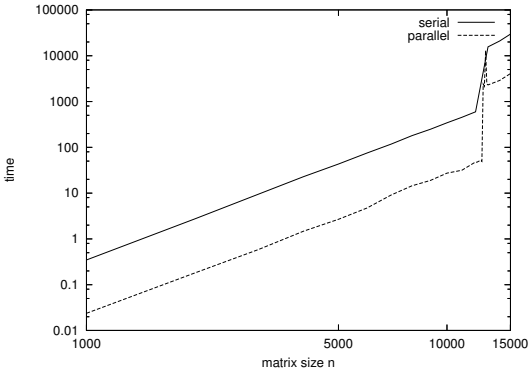

図 1　SR11000 における行列積の計算時間

$C = AB$ の (i, j) 成分 c_{ij} は，

$$c_{ij} = \sum_{l=1}^{n} a_{il} b_{lj} \tag{5}$$

と定義されるので，一成分を求めるための演算は加算と乗算がそれぞれ n 回で，全成分を求めるためには合計 $2n^3$ 回の浮動小数点演算が必要である。これを本稿では，演算量が $2n^3$ であると言う。

これを実際に SR11000 上で検証した結果が図 1 である。このグラフは，n 次行列同士の積の計算にかかる時間を表している。行列積計算には，提供されている BLAS ライブラリ*2を用いた。両対数グラフで，横軸は行列の次数 n で，縦軸は実行時間（単位は秒）である。実線は，1 CPU による逐次実行であり，破線は，16 CPU による並列計算である。このグラフを見ると，逐次，並列ともに $n = 12000$ 程度までは，n^3 に比例する実行時間であり，理論的な演算量評価と一致する。ただし，$n = 12000$ 以降において，実行時間が急激に増加している。この現象の原因は，現在究明中である。

また，$n = 5000$ では，逐次計算で 43.05 秒，並列計算で 2.67 秒で，並列計算は逐次計算の 1/16 程度の時間で終了しており，最大限に並列化がなされている。これは，行列積が並列計算機上で非常に高速であることを示している。ただし，n が 7000 以上，例えば $n = 10000$ では，逐次計算で 342 秒，並列計算で 27 秒で，13 倍弱の速度向上であり，必ずしも並列計算機上で非常に高速とはいえないという結果も得られている。この現象の原因も，現在究明中である。

なお，本原稿で示す数値実験で行列積計算を行う行列は，16 倍の高速化がなされている範囲の次元である。

2.3.3　2 分割の分割統治法と多分割の分割統治法

ここでは，実対称三重対角行列の固有値問題に対する 2 分割の分割統治法と，それを我々が拡張した多分割の分割統治法のアイデアを述べる。詳しい説明は後述する。一般的な意味での 2 分割の分割統治法は，大きな問題を解くために，2 つの小問題に分割し，その小問題の解を利用する，という方法である。実対称三重対角行列の固有値問題に適用した場合，まず 2 つの小さ

*2 LAPACK の下位ルーチンを集めたライブラリ。行列積など基本的な演算を行う。SR11000 では，LAPACK 同様最適化されたライブラリが提供されている。

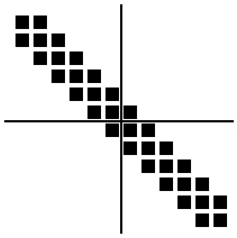

図 2　2 分割の場合の分割の方法

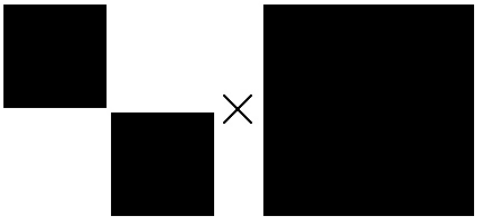

図 3　2 分割の場合の行列積

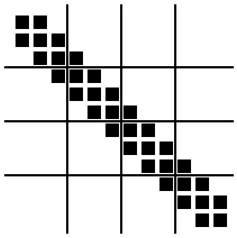

図 4　4 分割の場合の分割の方法

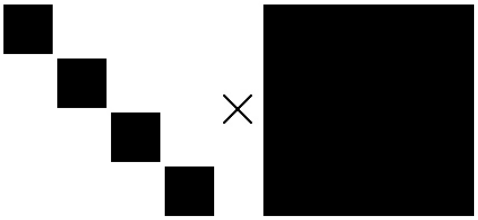

図 5　4 分割の場合の行列積

な実対称三重対角行列の固有値問題とわずかな残りの部分に分割する。図 2 は，分割の方法を模式的に表した図である。この図で，黒い部分はある要素で，白い部分は要素が零を意味する。1 つの実対称三重対角行列を，左上と右下の 2 つの半分の大きさの実対称三重対角行列と，わずかな余りの部分に分けている。

2 つの小さな実対称固有値問題を解を利用することにより，元の問題を別の固有値問題へと変換できる。変換後の問題は，変換前の問題と比較して容易に解くことが可能である。

最後に，元の問題を解くために，図 3 の形をした行列同士の積を計算する。分割統治法では，この行列積計算が演算量の面で支配的な部分を占める。左の行列の半分は零であるから，全てが零でない行列である場合の半分の演算量ですむ。

2 つに分割したために，左の行列の半分の要素が零となったのであるから，分割の数を増やせば，より零の部分を増やすことができ，行列積計算に必要な演算量が減少すると期待される。それは，実際に正しい。すなわち，4 分割ならば，図 5 のような形となり，その行列積は 2 分割の場合の半分の演算量となる。しかし，4 分割にした場合には，図 4 のように，分解時に発生する余りの部分が 2 分割の場合の 3 倍に増大し，その部分を処理するための演算量が増加するという問題は存在する。

2 分割の場合の演算量は，行列積が支配的であるため，行列積の演算量自体の $(4/3)n^3$ となる。k 分割であっても，行列積が支配的な分割数の範囲ならば，同様に $(2k/(k^2-1))n^3$ となる。また，行列積を用いるため，必然的に $O(n^2)$ のメモリ領域が必要となる。

また，2 分割，多分割ともに，いくつか分割した小さな実対称固有値問題は独立なため，この部分は並列に計算できる。

多分割の分割統治法には，最適な分割数の決定法など，未だ理論面での課題が残っている。

2.3.4　QR 法

正則な n 次行列 A に対して，直交行列 Q と右上三角行列 R との積 $A = QR$ への分解は QR 分解と呼ばれる。この分解は，A を構成する列ベクトル $\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n$ に対するグラム・シュ

ミットの直交化法

$\boldsymbol{q}'_1 := \boldsymbol{a}_1$
$\boldsymbol{q}_1 := \boldsymbol{q}'_1/\|\boldsymbol{q}'_1\|_2$
for $i := 2$ **to** n
  $\boldsymbol{q}'_i := \boldsymbol{a}_i - \sum_{j=1}^{i-1}(\boldsymbol{a}_i, \boldsymbol{q}_j)\boldsymbol{q}_j$
  $\boldsymbol{q}_i := \boldsymbol{q}'_i/\|\boldsymbol{q}'_i\|_2$
end for

により構成できる。このアルゴリズムの結果は，

$$(\boldsymbol{a}_1\ \boldsymbol{a}_2\ \cdots\ \boldsymbol{a}_n) = (\boldsymbol{q}_1\ \boldsymbol{q}_2\ \cdots\ \boldsymbol{q}_n)\begin{pmatrix} \|\boldsymbol{q}'_1\|_2 & (\boldsymbol{a}_2,\boldsymbol{q}_1) & (\boldsymbol{a}_3,\boldsymbol{q}_1) & \cdots & (\boldsymbol{a}_n,\boldsymbol{q}_1) \\ 0 & \|\boldsymbol{q}'_2\|_2 & (\boldsymbol{a}_3,\boldsymbol{q}_2) & \cdots & (\boldsymbol{a}_n,\boldsymbol{q}_2) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \|\boldsymbol{q}'_{n-1}\|_2 & (\boldsymbol{a}_n,\boldsymbol{q}_{n-1}) \\ 0 & 0 & 0 & 0 & \|\boldsymbol{q}'_n\|_2 \end{pmatrix} \tag{6}$$

と行列形式で表すことができ，これにより A の QR 分解が完了する。

この QR 分解に基づいて，正則な n 次行列 A の固有値問題を解くための方法が QR 法であり，基本的なアルゴリズムは，擬似コードを用いて以下のように記述できる。$A_1 = A$ を入力とし，

$i := 1$ に初期化
do
  A_i を $A_i = Q_i R_i$ と QR 分解する
  $A_{i+1} := R_i Q_i$ を計算する
while i を 1 づつ増やしながら，収束するまで繰り返す

アルゴリズムで，A_i の QR 分解の逆順の積で計算される A_{i+1} は，構成法より $A_{i+1} = R_i Q_i = Q_i^T A_i Q_i$ であり A_i と同じ固有値をもつ。$A_1, A_2, \cdots$ の極限は，対角線より下の要素が全て 0 に収束し，そのとき対角成分は A の固有値となることが知られている。

A が三重対角行列の場合，$A_i, (i = 1, 2, \cdots)$ も三重対角行列であるため，反復計算が可能である。この他，いくつかの高速化手法を取り入れた，実用的な QR 法の演算量は $6n^3$ である。

2.3.5 二分法・逆反復法

二分法は固有値を求める方法であり，逆反復法は固有ベクトルを求める方法である。これらは独立な方法であるが，併せて用いることが多いため，まとめて紹介する。

二分法　二分法は，実対称三重対角行列 T に対して適用することができ，T の固有値を一部，もしくは全て求める方法である。この解法では，T に依存した関数

$$N(x) \equiv x \text{ 以上の } T \text{ の固有値の数} \tag{7}$$

を定義する。この関数は単調に減少するから，**二分探索**を用いることで，固有値を求めることができる。それぞれの固有値は独立に求められるため，固有値の計算を並列化することができる。$N(x)$ の計算は $O(n)$ の演算量であるので，二分法で n 個の固有値を求めるための演算量は，$O(n^2)$ である。

続いて，二分法の中核をなす $N(x)$ の計算法を述べる。T_k を T の k 次首座小行列とする。k

次首座小行列とは，左上の $k \times k$ を切り出した行列である。また，$p_k(x)$ を

$$\begin{cases} p_0(x) = 1 \\ p_k(x) = \det(xI - T_k), \quad (k = 1, 2, \cdots, n) \end{cases} \tag{8}$$

とする。このとき，$p_0(x), p_1(x), p_2(x), \cdots, p_n(x)$ はスツルム列をなす。従って，スツルム列の理論より，$N(x)$ は $p_0(x), p_1(x), p_2(x), \cdots, p_n(x)$ の符号の変化した数と一致する。

$n = 2$ の場合，

$$T = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \tag{9}$$

を例に挙げる。このとき，$p_0(x) = 1, p_1(x) = x - a, p_2(x) = (x - a)(x - c) - b^2$ であるため，以下の表が書ける。

x	$\cdots$	λ_1	$\cdots$	a	$\cdots$	λ_2	$\cdots$
$p_0(x)$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
$p_1(x)$	$-$	$-$	$-$	0	$+$	$+$	$+$
$p_2(x)$	$+$	0	$-$	$-$	$-$	0	$+$
$N(x)$	2		1	1	1		0

この表より，ここで構成した $N(x)$ は，(7) で定義した関数となっていることがわかる。

逆反復法　一方，逆反復法は，与えられた固有値に対応する固有ベクトルを求める方法である。

逆反復法を説明するための準備として，基本となる，べき乗法を説明する。べき乗法は，正方行列 A に対して，絶対値が最大の固有値に対応する固有ベクトルを求める方法である。擬似コードでは，A を入力として，

初期ベクトル $\boldsymbol{x}_1$ を，長さ 1 のベクトルにとる
$i = 1$ に初期化
do
　$\boldsymbol{x}'_{i+1} := A\boldsymbol{x}_i$
　$\boldsymbol{x}_{i+1} := \boldsymbol{x}'_{i+1} / \|\boldsymbol{x}'_{i+1}\|_2$ と正規化
while $\boldsymbol{x}_i$ が収束するまで繰り返す

となる。このアルゴリズムにおいて，$\boldsymbol{x}_i$ が A の絶対値が最大の固有値に対応する固有ベクトルに収束する。

ここで，A の十分な精度の近似固有値 $\hat{\lambda}$ が求まっているとき，べき乗法を用いて，対応する固有ベクトル $\boldsymbol{q}$ を求めることを考える。$\tilde{A} = (A - \hat{\lambda}I)^{-1}$ とすれば，$\tilde{A}$ の絶対値最大の固有値に対応する固有ベクトルは，$\boldsymbol{q}$ に一致する。そのため，$\tilde{A}$ に対してべき乗法を適用することにより，$\boldsymbol{q}$ を求めることができる。このアルゴリズムが逆反復法である。

実際には，$(A - \hat{\lambda}I)^{-1}$ を陽に求めて，$\boldsymbol{x}_{i+1} = (A - \hat{\lambda}I)^{-1}\boldsymbol{x}_i$ を計算するのではなく，連立一次方程式 $(A - \hat{\lambda}I)\boldsymbol{x}_{i+1} = \boldsymbol{x}_i$ を解くことで $\boldsymbol{x}_{i+1}$ を求める。この解法で固有ベクトルを 1 つ求めるための演算量は $O(n)$，n 本全て求めるためには $O(n^2)$ の演算量である。

逆反復法は，それぞれの固有ベクトルが独立に求められるため，並列計算することができる。しかし，大規模な行列に対する数値計算では，直交するべき固有ベクトルが互いに直交せず，直

交性を確保するために，それらを事後に直交させる，再直交化をしなければならないことが多い。再直交化は並列計算に向かない。そのため，大規模な問題に対しては，$O(n^3)$ の演算量が必要であることが多い。

2.3.6 MRRR 法

MRRR (Multiple Relatively Robust Representations) 法は，紙面の都合で簡単に紹介する。MRRR 法は，最新の固有値問題解法の一つで，演算量 $O(n^2)$ で全ての固有値と固有ベクトルを求めることができることが特長である。新しい解法であるため，未だ理論面に不十分な面がある。各固有ベクトルを並列に求めることが可能であるとされるが，SR11000 のライブラリにある MRRR 法では，並列計算と逐次計算の実行時間の比がほぼ 1 であるため，ここでは並列性が低いとした。[山本，2005] に詳しい。

3. 2 分割の分割統治法

この節では，実対称行列に対する 2 分割の分割統治法について述べる。このアルゴリズムは，本稿でとり扱う多分割の分割統治法の基礎となっているアルゴリズムである。

3.1 メタアルゴリズムとしての分割統治法

一般に，分割統治法は以下の 3 つのステップからなる**メタアルゴリズム**[*3]である。

proc MetaDC ····· 問題 P を分割統治法で解く（k 分割）

MetaDC-1　問題 P を k 個の小さな問題 $P_1, P_2, \cdots, P_k$ に k 分割する。
（ここで，k を分割数と呼ぶことにする。）

MetaDC-2　小さな問題 $P_1, P_2, \cdots, P_k$ をそれぞれ解く。

MetaDC-3　小さな問題の解を利用して，大きな問題 P を解く。

ここで，P_j が P とサイズが異なっているだけで，P と同様の問題であるときにのみ，このメタアルゴリズムを用いることができる。同様の問題であるので，P_j のそれぞれについてもメタアルゴリズム MetaDC を再帰的に用いることができる。ここに，分割統治法の本質的な並列性が存在する。また，高速化のために，P_j の規模が十分小さいときには，再帰的に計算をすることなく，より単純な別の解法を用いて P_j を解くことが一般的である。

整列（ソーティング）問題が代表的な適用例で，その中で**クイックソート**がよく知られている。

3.2 実対称行列に対する 2 分割の分割統治法

前小節で述べた，メタアルゴリズムとしての分割統治法を，分割数が 2 の場合に，実対称三重対角行列の固有値問題に適用した解法を紹介する。このアルゴリズムは，Cuppen によって 1981 年に提案された。

T は対角成分が a_i $(i = 1, 2, \cdots, n)$，副対角成分が b_i $(i = 1, 2, \cdots, n-1)$ であるとする。た

[*3] 問題に依存しない形式で記述されたアルゴリズム。

だし，ある b_i が 0 のとき T は可約のため，一般性を失うことなく $b_i \neq 0$ を仮定する。

まず，[MetaDC-1] のステップとして実対称三重対角行列 T を 2 つの実対称三重対角行列に分割する。$n = 6$ の場合を例にして説明すると，

$$
\begin{aligned}
T &= \left(\begin{array}{ccc|ccc}
a_1 & b_1 & & & & \\
b_1 & a_2 & b_2 & & & \\
 & b_2 & a_3 & b_3 & & \\
\hline
 & & b_3 & a_4 & b_4 & \\
 & & & b_4 & a_5 & b_5 \\
 & & & & b_5 & a_6
\end{array}\right) \\
&= \left(\begin{array}{ccc|ccc}
a_1 & b_1 & & & & \\
b_1 & a_2 & b_2 & & & \\
 & b_2 & a_3 - b_3 & & & \\
\hline
 & & & a_4 - b_3 & b_4 & \\
 & & & b_4 & a_5 & b_5 \\
 & & & & b_5 & a_6
\end{array}\right) + b_3 \left(\begin{array}{c} 0 \\ 0 \\ 1 \\ \hline 1 \\ 0 \\ 0 \end{array}\right) \left(\begin{array}{ccc|ccc} 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right)
\end{aligned} \tag{10}
$$

と分割する。これは，3 次実対称三重対角行列 T_1, T_2 を用いて，

$$
T = \left(\begin{array}{c|c} T_1 & \\ \hline & T_2 \end{array}\right) + b_3 \boldsymbol{v}\boldsymbol{v}^T, \quad \boldsymbol{v} = \left(\begin{array}{ccc|ccc} 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right)^T \tag{11}
$$

と書ける。

次に，[MetaDC-2] のステップとして，分解した T_1, T_2 の固有値問題を解く。すなわち，T_1, T_2 の固有値問題 $T_i = P_i D_i P_i^T$ の解，実対角行列 D_1, D_2，直交行列 P_1, P_2 を求める。

最後に，[MetaDC-3] のステップとして，[MetaDC-2] の結果を利用して，T の固有値問題を解く。T_1, T_2 の固有値問題の解 D_1, D_2，P_1, P_2 を利用して，T は，

$$
\begin{aligned}
T &= \left(\begin{array}{cc} P_1 D_1 P_1^T & \\ & P_2 D_2 P_2^T \end{array}\right) + b_3 \boldsymbol{v}\boldsymbol{v}^T \\
&= \left(\begin{array}{cc} P_1 & \\ & P_2 \end{array}\right) \left\{ \left(\begin{array}{cc} D_1 & \\ & D_2 \end{array}\right) + b_3 \boldsymbol{u}\boldsymbol{u}^T \right\} \left(\begin{array}{cc} P_1^T & \\ & P_2^T \end{array}\right) \\
&\equiv P(D + b_3 \boldsymbol{u}\boldsymbol{u}^T) P^T
\end{aligned} \tag{12}
$$

と相似変形できる。ただし，実対角行列 $D \equiv D_1 \oplus D_2$，直交行列 $P \equiv P_1 \oplus P_2$，

$$
\boldsymbol{u} \equiv \left(\begin{array}{cc} P_1^T & \\ & P_2^T \end{array}\right) \boldsymbol{v} = \left(\begin{array}{c} P_1^T \text{の最終列} \\ P_2^T \text{の第 1 列} \end{array}\right) \tag{13}
$$

このとき，実対角行列と階数 1 の摂動の和 $D + b_3 \boldsymbol{u}\boldsymbol{u}^T$ の固有値問題 $D + b_3 \boldsymbol{u}\boldsymbol{u}^T = Q \Lambda Q^T$ を解くことで

$$
T = PQ \Lambda Q^T P^T \tag{14}
$$

となる。これで，T の全ての固有値と固有ベクトルが求められた。

ここで，P と Q の積は，

$$
PQ = \left(\begin{array}{ccc|ccc}
* & * & * & & & \\
* & * & * & & & \\
* & * & * & & & \\
\hline
 & & & * & * & * \\
 & & & * & * & * \\
 & & & * & * & *
\end{array}\right) \left(\begin{array}{cccccc}
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & * \\
* & * & * & * & * & *
\end{array}\right) \tag{15}
$$

の形式をしている（$*$ はある実数，空白は 0 を意味する）。この行列積は，通常の 6 次行列同士の積の半分の演算量で済む。これは，n 次行列でも同様であり，演算量は n^3 である。

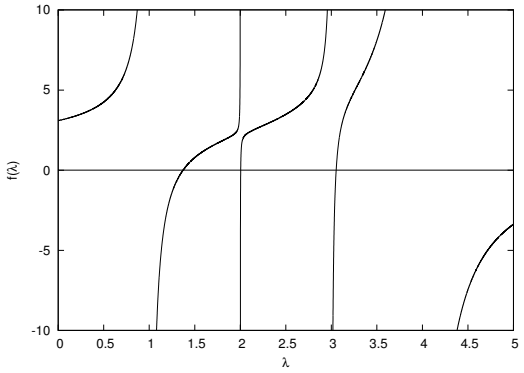

図 6　$f(\lambda)$ の例

実対角行列と階数 1 の摂動の和の固有問題を解くための演算量は，後述するように $O(n^2)$ である。また，メタアルゴリズムとしても述べたとおり，T_1, T_2 の固有値問題は，このアルゴリズムにより（再帰的に）解くことができる。

そのため，この素朴な 2 分割の分割統治法の演算量は，$(4/3)n^3$ である。すなわち，演算量の大部分を行列積が占めている。2.3.2 小節で述べたように，行列積は現在の計算機で高速に計算が可能であるため，2 分割の分割統治法も高速に計算される。

3.3　実対角行列と階数 1 の摂動の和の固有値問題

前小節で述べたように，分割統治法で実対称固有値問題を解くためには，実対角行列と階数 1 の摂動の和 $D + c\boldsymbol{u}\boldsymbol{u}^T$ の固有値問題を解かなければならない。

この行列の固有値は，$D + c\boldsymbol{u}\boldsymbol{u}^T$ の特性方程式を変形した，

$$f(\lambda) \equiv \frac{1}{c} - \sum_{j=1}^{n} \frac{u_j^2}{\lambda - d_j} \tag{16}$$

の零点を求める問題に帰着することができる。この非線形関数 $f(\lambda)$ の $n = 4$ の場合の例が，図 6 である。

この関数 $f(\lambda)$ の零点，すなわち $D + c\boldsymbol{u}\boldsymbol{u}^T$ の固有値は，変形ニュートン法によって求めることができる。n 次の問題に対するこの解法の演算量は，1 つの固有値のために $O(n)$，n 個の固有値を求めるためには $O(n^2)$ である。また，$f(\lambda)$ の零点は，その性質を利用して n 個それぞれを独立に求めることができ，ここに高い並列性が存在する。

一方，固有値 λ に対応する（ノルムが 1 の）固有ベクトルは

$$\frac{(\lambda I - D)^{-1}\boldsymbol{u}}{\|(\lambda I - D)^{-1}\boldsymbol{u}\|_2} \tag{17}$$

で求められる。この方法による，n 次行列の全ての固有ベクトルを求めるために必要な演算量は $O(n^2)$ である。なお，実装においては，数値的に精度を高めるための前処理が追加される。この処理が追加されても演算量が $O(n^2)$ であることは変わらない。

以上により，n 次元の実対称行列と階数 1 の摂動の固有値と固有ベクトルは，$O(n^2)$ の演算量で全て求められる。

3.4 アルゴリズム

ここまでのまとめとして，2 分割の分割統治法を擬似コードで表す。

アルゴリズム　分割統治法 (2 分割) を用いて，n 次実三重対角行列 T の固有値・固有ベクトルを求める:

proc dc_eig(T, Q, Λ) ····· 入力 T から $T = Q\Lambda Q^T$ を満たす直交行列 Q, 実対角行列 Λ を出力する

if T が 1 次
　return $Q = 1, \Lambda = T$
else
　$T = \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix} + c\boldsymbol{v}\boldsymbol{v}^T$ の形式にする
　call dc_eig(T_1, Q_1, Λ_1)
　call dc_eig(T_2, Q_2, Λ_2)
　$\Lambda_1, \Lambda_2, Q_1, Q_2$ から $D + c\boldsymbol{u}\boldsymbol{u}^T$ の形式にする
　固有値問題 $D + c\boldsymbol{u}\boldsymbol{u}^T = Q'\Lambda Q'^T$ を解く
　$Q = \begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix} Q'$ を計算する
　return Q, Λ
endif

なお，実装の際には，分割後の T の次元 n が 1 の場合のみではなく，n がある定数以下の場合に，他のより単純なアルゴリズムを利用することで，再帰の負荷を回避し，高速化を図る。

3.5 デフレーション

前小節で示した分割統治法を，多くの行列に対して，他の固有値問題解法と同等以上の速度に高速化する手法がデフレーションである。デフレーションは，対角行列と階数 1 の摂動の和の固有値問題に存在する，自明な固有値と固有ベクトルを取り除く手法である。

$n = 3$ の場合に，デフレーションが可能になる例を挙げると，

$$\left[\begin{pmatrix} d_1 & & \\ & d_2 & \\ & & d_3 \end{pmatrix} + c\begin{pmatrix} u_1 \\ 0 \\ u_3 \end{pmatrix}\begin{pmatrix} u_1 & 0 & u_3 \end{pmatrix}\right]\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = d_2\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{18}$$

である。すなわち，摂動項の $\boldsymbol{u}$ の第 j 成分が 0 であるときに，自明な固有値は実対角行列の第 j 対角成分であり，対応する固有ベクトルは第 j 単位ベクトルである。ただし，第 j 単位ベクトルは，単位行列の第 j 列とする。

ここで分割統治法に話を限定し，6 次元の固有値問題に対する分割統治法の内部における問題を考える。いま，この行列の第 1 固有値が自明であると判定され，Q の第 1 列が第 1 単位ベクトルとなったとする。Q が直交行列であることより，Q の第 1 行の 2 列目以降が 0 となること

も自動的に決定する。したがって，P と Q の積は

$$PQ = \left(\begin{array}{ccc|ccc} * & * & * & & & \\ * & * & * & & & \\ * & * & * & & & \\ \hline & & & * & * & * \\ & & & * & * & * \\ & & & * & * & * \end{array}\right) \left(\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & * & * & * & * & * \end{array}\right) \tag{19}$$

となる。このとき，この積の第 1 列は計算する必要はないなどの演算量の削減が可能で，この例では，演算回数が 216 回から 150 回に減り，30 パーセント演算量が減少する。

分割統治法の一部として，対角行列と階数 1 の摂動の和の固有値問題を解いている場合，このデフレーションは，分割統治法に与えられた行列の性質によって発生率が変化する。デフレーションが多く発生する問題に対しては，$O(n^2)$ 以下の演算量で済む場合も存在し，平均して $O(n^{2.3})$ の演算量であるという報告もある。

4. 多分割へ拡張した分割統治法

2 分割の分割統治法を多分割に拡張することにより，行列積の演算量を削減することが可能で，アルゴリズム全体の演算量を削減することができる。

分割数が 3 の場合を例に，多分割の統治法のアルゴリズムを説明する。4 以上の分割数であっても，同様に拡張が可能である。

4.1 3 分割の固有値問題

この節では，2 分割の場合の説明と同様に，$n = 6$ を例に説明する。まず，実対称三重対角行列 T を

$$T = \left(\begin{array}{cc|ccc|cc} a_1 & b_1 & & & & & \\ b_1 & a_2 & b_2 & & & & \\ \hline & b_2 & a_3 & b_3 & & & \\ & & b_3 & a_4 & b_4 & & \\ \hline & & & b_4 & a_5 & b_5 & \\ & & & & b_5 & a_6 & \end{array}\right)$$

$$= \left(\begin{array}{cc|ccc|cc} a_1 & b_1 & & & & & \\ b_1 & a_2 - b_2 & & & & & \\ \hline & & a_3 - b_2 & b_3 & & & \\ & & b_3 & a_4 - b_4 & & & \\ \hline & & & & a_5 - b_4 & b_5 & \\ & & & & b_5 & a_6 & \end{array}\right) + b_2 \left(\begin{array}{c} 0 \\ 1 \\ \hline 1 \\ 0 \\ \hline 0 \\ 0 \end{array}\right) \left(\begin{array}{c} 0 \\ 1 \\ \hline 1 \\ 0 \\ \hline 0 \\ 0 \end{array}\right)^T + b_4 \left(\begin{array}{c} 0 \\ 0 \\ \hline 0 \\ 1 \\ \hline 1 \\ 0 \end{array}\right) \left(\begin{array}{c} 0 \\ 0 \\ \hline 0 \\ 1 \\ \hline 1 \\ 0 \end{array}\right)^T$$

と分割する。

このとき，2 次実対称三重対角行列 T_1, T_2, T_3 を用いて，

$$T = \left(\begin{array}{cc|cc|cc} T_1 & & & & & \\ & & & & & \\ \hline & & T_2 & & & \\ & & & & & \\ \hline & & & & T_3 & \\ & & & & & \end{array}\right) + b_2 \boldsymbol{v}_1 \boldsymbol{v}_1^T + b_4 \boldsymbol{v}_2 \boldsymbol{v}_2^T, \quad \boldsymbol{v}_1 = \left(\begin{array}{cc|cc|cc} 0 & 1 & 1 & 0 & 0 & 0 \end{array}\right)^T, \boldsymbol{v}_2 = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right)^T$$

と書ける。

次に，分解した T_1, T_2, T_3 の固有値問題を解く。すなわち，T_1, T_2, T_3 の固有値問題 $T_i = P_i D_i P_i^T$ の解，実対角行列 D_1, D_2, D_3，直交行列 P_1, P_2, P_3 を求める。

最後に，T_1, T_2, T_3 の固有値問題の解 $D_1, D_2, D_3, P_1, P_2, P_3$ を利用して，T は，

$$
\begin{aligned}
T &= \begin{pmatrix} P_1 D_1 P_1^T & & \\ & P_2 D_2 P_2^T & \\ & & P_3 D_3 P_3^T \end{pmatrix} + b_2 \boldsymbol{v}_1 \boldsymbol{v}_1^T + b_4 \boldsymbol{v}_2 \boldsymbol{v}_2^T \\
&= \begin{pmatrix} P_1 & & \\ & P_2 & \\ & & P_3 \end{pmatrix} \left\{ \begin{pmatrix} D_1 & & \\ & D_2 & \\ & & D_3 \end{pmatrix} + b_2 \boldsymbol{u}_1 \boldsymbol{u}_1^T + b_4 \boldsymbol{u}_2 \boldsymbol{u}_2^T \right\} \begin{pmatrix} P_1^T & & \\ & P_2^T & \\ & & P_3^T \end{pmatrix} \\
&\equiv P(D + b_2 \boldsymbol{u}_1 \boldsymbol{u}_1^T + b_4 \boldsymbol{u}_2 \boldsymbol{u}_2^T) P^T
\end{aligned} \tag{20}
$$

と相似変形できる。ただし，実対角行列 $D \equiv D_1 \oplus D_2 \oplus D_3$，直交行列 $P \equiv P_1 \oplus P_2 \oplus P_3$，$\boldsymbol{u}_j \equiv P^T \boldsymbol{v}_j$ である。

このとき，実対角行列と階数 2 の摂動の和の固有値問題 $D + b_2 \boldsymbol{u}_1 \boldsymbol{u}_1^T + b_4 \boldsymbol{u}_2 \boldsymbol{u}_2^T = Q \Lambda Q^T$ を解くことで

$$
T = PQ \Lambda Q^T P^T \tag{21}
$$

となる。

ここで，P と Q の積は，

$$
PQ = \left(\begin{array}{cc|cc|cc} * & * & & & & \\ * & * & & & & \\ \hline & & * & * & & \\ & & * & * & & \\ \hline & & & & * & * \\ & & & & * & * \end{array} \right) \left(\begin{array}{cccccc} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{array} \right) \tag{22}
$$

の形式をしている（$*$ はある実数，空白は 0 を意味する）。この行列積は，通常の 6 次行列同士の積の 1/3 の演算量で済む。これは，n 次行列でも同様であり，演算量は $(2/3)n^3$ である。k 分割の場合には，行列積の演算量は $(2/k)n^3$ である。すなわち，2 分割の分割統治法の $2/k$ の演算量に減少する。

また，実対角行列と低階数の摂動の和の固有値問題は $O(n^2)$ で解くことが可能である。そのため，k 分割の分割統治法は約 $(2/k)n^3$ の演算量であり，2 分割の分割統治法の $3/(2k)$ の演算量に減少する。ただし，実対角行列と低階数の摂動の和の固有値問題は，摂動の階数が増加すると解くために必要な演算量が増加し，分割統治法の中でこの部分が演算量の多くを占めるようになるため，問題に依存して演算量が最小となる分割数が存在する。

4.2 実対角行列と低階数の摂動の和の固有値問題

2 分割の場合には「実対角行列と階数 1 の摂動の和」の固有値問題を解く部分が，分割数が k に拡張された場合には,「実対角行列と階数 $k-1$ の摂動の和」の固有値問題を解くこととなる。

実対角行列と階数 $k-1$ の摂動の和の固有値は，実対角行列と階数 1 の摂動の和の固有値問題を $k-1$ 回逐次的に解くことで，求めることができる。実対角行列と階数 1 の摂動の和の固有値問題は，3.3 小節で述べたように，固有値を並列に求めることができるため，実対角行列と階数 $k-1$ の摂動の和の固有値の計算も，並列性を有する。

一方，固有ベクトルは，同じく 3.3 小節で述べた，実対角行列と階数 1 の摂動の和の固有ベクトル計算を拡張した方法により，求めることができる。その内部では，反復法による計算を行っている。この固有ベクトル計算は，それぞれの固有ベクトルについて独立であり，ここに高い並列性を有する。

両者の方法ともに，n 次元の問題を解くための演算量は $O(n^2)$ である。ただし，摂動の階数が増加したことで，問題の難度が上がり，解くために必要な演算量自体は増加する。

4.3 デフレーション

実対角行列と階数 1 の摂動の和の固有値問題と同様に，摂動の階数が 2 以上の場合でも，デフレーションが発生する。それにより，多分割の分割統治法でも，演算量の支配的な部分を占める行列積の演算量を削減することができる。

しかし，分割数の増加に従って，デフレーション発生率が減少するため，デフレーションによる演算量の削減効果は薄れる。

4.4 アルゴリズム

この節の最後に，3 分割の分割統治法を擬似コードとしてまとめる。

アルゴリズム　3 分割の分割統治法を用いて，n 次実三重対角行列 T の固有値・固有ベクトルを求める:

proc dc_eig3(T, Q', Λ) ‥‥‥ 入力 T から $T = Q'\Lambda Q'^T$ を満たす直交行列 Q', 対角行列 Λ を出力する

if T が 1 次か 2 次
　call eig(T, Q', Λ)　　(* eig() は固有値問題 $T = Q'\Lambda Q'^T$ を解く他の手続き *)
　return Q', Λ
else
　$T = \begin{pmatrix} T_1 & & \\ & T_2 & \\ & & T_3 \end{pmatrix} + c_1\boldsymbol{v}_1\boldsymbol{v}_1^T + c_2\boldsymbol{v}_2\boldsymbol{v}_2^T$　(* T_i は実三重対角行列，$\boldsymbol{v}_1$, $\boldsymbol{v}_2$ は実ベクトル，c_1, c_2 は実数 *)
　call dc_eig3(T_i, P_i, D_i)　$(i = 1, 2, 3)$
　D_i, P_i から $D + c_1\boldsymbol{u}_1\boldsymbol{u}_1^T + c_2\boldsymbol{u}_2\boldsymbol{u}_2^T$ の形式にする $(i = 1, 2, 3)$
　固有値問題 $D + c_1\boldsymbol{u}_1\boldsymbol{u}_1^T + c_2\boldsymbol{u}_2\boldsymbol{u}_2^T = Q\Lambda Q^T$ を解く
　$Q' = \begin{pmatrix} P_1 & & \\ & P_2 & \\ & & P_3 \end{pmatrix} Q$ を計算する
　return Q', Λ
endif

なお，実装の際には，2 分割の場合と同様に，n がある定数以下の場合に，他のより単純なアルゴリズムに切り替えることで，高速化を図る。

5. 共有メモリ型並列計算機への実装

5.1 並列計算

並列計算には大きく分けて，共有メモリ型並列計算と分散メモリ型並列計算の 2 通りがある。この 2 つを SR11000 に当てはめた場合，128 個存在するノードのいくつかを使って並列化する方法が分散メモリ型並列計算であり，1 ノードのみを使い全 16 個のプロセッサの内の複数のプロセッサを用いて並列化する方法が共有メモリ型並列計算である。ノードとは，スーパーコンピュータの構成要素の一つで，単一のメモリ空間と複数のプロセッサからなるものである。

本研究では多分割の分割統治法の並列化の第一段階として，1 ノードを利用した共有メモリ型の並列化を施し高速化を図る。今後の研究では複数ノードを利用した分散メモリ型の並列化を目指している。

次の小節ではいくつかの用語説明を交えながら共有メモリ型並列プログラムの作成方法について説明する。

5.2 共有メモリ型の並列化

共有メモリ型の並列化を行う方法には，共有メモリ型並列化を施されたライブラリをリンクして用いる方法と，指示子と呼ばれる文をソースコード内に挿入して，コンパイラに並列化の指示を与える方法がある。SR11000 上では，LAPACK 等の，既に並列化を施されたライブラリが提供されているため，前者の並列化の方法はこれらをリンクすることで実現できる。一方，指示子を挿入して並列化を行う場合，一般的に OpenMP という API を用いる。OpenMP は，Unix や Windows NT のプラットフォームを含む多くのアーキテクチャでサポートされている API である。本研究においても OpenMP を用いて並列化を行った。

C 言語で書かれたプログラムを OpenMP によって並列化を行う場合，コード中に図 7 の命令を書くことで，波括弧内の命令が複数のプロセッサを用いて実行を行う並列区間であることを指示することができる。並列区間では，各プロセッサの処理は「スレッド」と呼ばれる並列実行単位で表される。

```
#pragma omp parallel
{
  /* 実行文 */
}
```

図 7　並列部の指定

図 8 のように並列区間中の for 文の前に一文 “#pragma omp for” を入れることで，この for 文は複数のスレッドで分担して実行を行うことをコンパイラに指示する。“#pragma omp for” を for 指示文と呼ぶ。本研究では，自作したコードはすべて for 指示文を用いて並列化した。

```
#pragma omp parallel
{
  int i ;
  /* 実行文 */
  #pragma omp for
  for ( i = 0 ; i < 100 ; i++ ) {
    /* 実行文 */
  }
  /* 実行文 */
}
```

図 8　ループ並列 (for 文の並列化)

for 指示文では，スレッドに仕事を割り振る方法を明示的に指示することもできる。チャンクサイズ 10 で静的に割り付けるには “#pragma omp for” の後に “schedule (static , 10)” を付け足すことで指示できる。以下でチャンクサイズ，静的および動的な割り付けについて説明する。

チャンクサイズとは 1 回の処理量のことで，図 8 を例に説明すればチャンクサイズ 10 は変数 i のカウント 10 個分を 1 回として考えることを意味し，5 スレッドを静的に割り付けて並列実行した場合，各スレッドで 2 回分ずつ演算を行う。もしチャンクサイズを指定しなければデフォルトのサイズ (各スレッドで 1 回分ずつ演算を行うサイズ) で割り付けを行う。

スレッドに仕事を割り振ることを割り付けといい，静的な方法と動的な方法がある。静的に割り付ける場合はチャンクサイズ分の演算を各スレッドへ順番に割り振る。動的に割り付ける場合は順次，演算の終わったスレッドにまだ終わっていない演算を割り振る。

動的に割り付けを指示する際は “static” の代わりに “dynamic” と宣言する。静的に割り付けを行った際は，割り振り方が決定的で時間がかからない。しかし，1 回の演算の計算時間が各々異なる場合，計算時間が偏り並列効果が下がることがある。その場合には，動的な割り付けを選択することにより，並列効果の向上が期待できる。しかし，動的な割り付けは静的な割り付けより割り付け時間を要する。

詳しい OpenMP の利用法についてはマニュアル「最適化 C 使用の手引」を参照されたい。また以後の小節では，本研究で施した並列化および並列化したアルゴリズムの項目の主要部分について説明を行う。

5.3 並列化項目

4 節で示した固有値問題に対する多分割の分割統治法のアルゴリズム中で，主に，以下に述べる 4 つの項目に対して並列化を行った。デフォルトのチャンクサイズで静的な割り付けを行った場合に並列効果が低い場合は，動的な割り付けを試し，並列効果の高い方法を選択した。

5.3.1 T_i の固有値問題計算

T_i の固有値問題計算とは，実対称三重対角行列 T を分割して得られた行列 T_i の固有値問題計算のことである (4.1 小節参照)。数値実験では，並列版 LAPACK ライブラリにある，2 分割の分割統治法を用いた実対称三重対角行列の固有値問題解法関数を用いて並列計算を行った。

明示的に再帰呼び出しをせず，ライブラリ関数を利用した理由は，1. 分割するにつれ，最適分割数が小さくなる傾向があること，2. 現在，与えられた問題に対し動的に最適分割数を選ぶ手法が明らかにされていないこと，3. 従来の分割統治法との差異を少なくし比較しやすくすることの 3 点である。最適分割数については以降で説明する。

5.3.2 対角行列と低階数の摂動の和の固有値計算

対角行列と低階数の摂動の和の固有値計算の主要部分は「対角行列と階数 1 の摂動の和の固有値計算」である (4.2 小節参照)。対角行列と階数 1 の摂動の和の固有値計算では，各固有値計算が独立した計算であるため，for 指示文を用いて並列化を行う。反復解法を用いているが，割り付けの方法による計算時間の差異が見られなかったので，本プログラムではデフォルトのチャンクサイズを用い，静的な割り付けを行った。

5.3.3 対角行列と低階数の摂動の和の固有ベクトル計算

各固有ベクトルは，内部的に反復解法を用いて求める (4.2 小節参照)。各固有ベクトルの計算は独立した計算であるため，for 指示文により効率的な並列化を行うことができる。本研究では各々の反復法による計算時間が一定でないため，チャンクサイズを 1 とし動的に割り付けを行った。

5.3.4 行列積計算

「行列積計算」は実対称三重対角行列の固有ベクトルを求めるための行列積計算で，式 (22) の Q と P の行列積を指す。この計算には並列版 BLAS ライブラリを用いた。このライブラリは SR11000 用にチューニングが施されていて，2.3.2 小々節で述べたように，自作するよりも高速に計算できる。

6. 数値実験

6.1 実験環境

本研究では SR11000 の 1 ノードを用いて実験を行った。並列計算に使用した CPU の台数は 16 台である。実行時間の計測には FORTRAN の関数 `xclock` を用いて，実時間の計測を行った。以降では，使用したライブラリおよびそのライブラリを利用したプログラムのコンパイル方法について説明する。

6.1.1 ライブラリ

数値計算ライブラリとして SR11000 で提供されている LAPACK(BLAS) ライブラリを用いた。比較対象として用いた関数は表 2 の通りである。

表 2 LAPACK 関数名

	関数名
QR 法	`dsteqr`
二分法・逆反復法	`dstebz`, `dstein`
MRRR 法	`dstegr`
分割統治法	`dstevd`

表 3 コンパイラ

言語	コンパイラ名	コマンド名
C 言語	最適化 C	cc
FORTRAN90	最適化 FORTRAN90	f90

6.1.2 コンパイル方法

コンパイラとして最適化 C コンパイラと最適化 FORTRAN90 コンパイラを使用した (表 3)。ソースコードは C 言語で作成したため，コンパイルには最適化 C コンパイラを用いた。オブジェクトのリンクには，FORTRAN 用に作成された LAPACK ライブラリをリンクするので，最適化 FORTRAN90 コンパイラを用いた。

最適化オプションは，-Os にすると本プログラムでは正常に計算できなかったので，正常に計算できる最大レベルである-O4 を選択した。並列化レベルは，最高レベルの 4 とした。並列プログラムでは OpenMP を使用しているので-omp を指定する。逐次プログラムでは逐次ライブラリをリンクするため-L/usr/local/lib -llapack_sc -lblas_sc を指定する。並列プログラムでは並列ライブラリをリンクするため-L/usr/local/lib -llapack -lblas を指定する。LAPACK ライブラリ

は FORTRAN 用に作成されているため，C 言語から呼び出せるように-lf90s を指定する。実際に指定したオプションは表 4 の通りである。

表 4　コンパイル命令

逐次	コンパイル	cc -64 -O4 +Op -noparallel -c
	リンク	f90 -64 -O4 -i,P -noparallel -L/usr/local/lib -llapack_sc -lblas_sc -L/opt/ofort90/lib/ -lf90s -lm
並列	コンパイル	cc -64 -O4 +Op -parallel=4 -pardiag=2 -omp -c
	リンク	f90 -64 -O4 -i,P -parallel=4 -pardiag=2 -omp -L /usr/local/lib -llapack -lblas -L/opt/ofort90/lib/ -lf90s -lm

6.2　実験方法

本実験では，大きく性質の異なる 2 種類の行列を入力とし，いくつかの分割数を用いて数値実験を行った。この時，計算に必要とした実行時間を測定し，計算結果の精度を評価するため残差と直交誤差を求めた。以降では，入力として用いた行列，最適分割数および精度の評価法について説明する。

6.2.1　対象行列

計算対象の行列として，主対角要素に $(2, 4]$，副対角要素に $(1, 2]$ の区間の一様乱数を持つ行列と，主対角の第 j 要素に $j \times 10^{-6}$，副対角要素に 1 を持つ行列の 2 種類を用いた。前者はデフレーションの発生が多い行列で行列 A と呼ぶ。後者はデフレーションの発生が非常に少ない行列で行列 B と呼ぶ。

6.2.2　時間計測

時間計測には前述の通り，`xclock` を用いて実時間を計測した。行列 A に対しては 50 回の計測を行って平均を取った（行列要素は計測ごとに異なる）。現状のプログラムでは正常に計算することができないことがあり，その場合は計算できた中で平均を取った。

6.2.3　精度の評価法

計算結果の精度については，相対残差 ϵ_R と直交誤差 ϵ_O，

$$\epsilon_R = \max_{1 \le i \le n} \frac{||T\boldsymbol{q}_i - \lambda_i \boldsymbol{q}_i||_2}{||T||_2}, \quad \epsilon_O = \max_{1 \le i \le j \le n} |\boldsymbol{q}_i^T \boldsymbol{q}_j - \delta_{ij}| \tag{23}$$

を用いて評価する。ただし，T は入力した実対称三重対角行列，λ_i は T の固有値，$\boldsymbol{q}_i$ は λ_i に対応する T の固有ベクトル，δ_{ij} はクロネッカのデルタとする。

6.2.4　最適分割数

対象行列の性質やサイズによって，提案手法を用いて最も高速に計算できる行列の分割数は変化する。今回の数値実験では，一つの対象行列に対していくつかの分割数で実行時間を計測し，その中で最も速く計算できた時の分割数を，その対象行列に対する最適分割数と呼ぶこととする。

表 5　各アルゴリズムの実行時間：$n = 10000$

アルゴリズム		行列 A			行列 B		
		逐次 (秒)	並列 (秒)	台数効果	逐次 (秒)	並列 (秒)	台数効果
LA PA CK	QR 法	1671.504	450.395	3.711	1489.022	424.209	3.510
	二分法	1074.868	483.799	2.222	2910.293	1221.946	2.382
	MRRR 法	24.191	23.689	1.021	19.137	18.614	1.028
	分割統治法	6.914	3.297	2.097	209.829	23.258	9.022
提案手法		8.137	3.256	2.499	120.540	14.060	8.573

表 6　各アルゴリズムの実行時間：$n = 5000$

アルゴリズム		行列 A			行列 B		
		逐次 (秒)	並列 (秒)	台数効果	逐次 (秒)	並列 (秒)	台数効果
LA PA CK	QR	228.364	90.585	2.521	199.937	80.023	2.498
	二分法	97.980	77.412	1.266	383.135	259.976	1.474
	MRRR 法	6.226	6.144	1.013	4.713	4.593	1.026
	分割統治法	2.480	1.525	1.626	28.334	4.687	6.045
提案手法		4.726	1.947	2.427	12.240	2.509	4.878

6.3　実験結果

6.3.1　各アルゴリズムの比較

表 5, 6 はアルゴリズム別の実行時間についてまとめた表で，縦軸はアルゴリズム，横軸は各対象行列の計算に要した時間と台数効果である。行列 A, B のサイズ 10000 (表 5) と 5000 (表 6) について各々逐次プログラムと並列プログラムの計測を行った。提案手法の分割数は，並列計算時の最適分割数を選択した。

表 5, 6 より，行列 A の場合は，LAPACK の MRRR 法と比較すると逐次プログラムの段階で計算速度が速く，並列化することによりさらに高速化できている。LAPACK の分割統治法と比較すると $n = 10000$ の時は同等の速度だが $n = 5000$ の時は遅い。行列 B の場合は LAPACK の MRRR 法と比較すると，逐次プログラムでは遅いものの，並列プログラムでは台数効果が高く高速に計算できている。LAPACK の分割統治法と比較しても高速に計算できている。

表 5, 6 より台数効果の比較を行うと，多分割の分割統治法の台数効果は LAPACK の分割統治法と同等である。また，MRRR 法と比較すると，多分割の分割統治法の台数効果が高い。

6.3.2　最適分割数

表 7, 8 は並列プログラムの計算時間を分割数別にまとめた表で，縦軸は対象行列のサイズ，横軸は各分割数である。行列 A (表 7), B (表 8) のサイズ 10000 と 5000 について計測を行った。

表 7 より行列 A の最適分割数は $n = 10000$ では 3 分割であった。$n = 5000$ では 16 分割であったが 2 番目は 3 分割であった。このことから，行列 A の最適分割数は小さい傾向があるように見て取れる。表 8 より行列 B の最適分割数は $n = 10000$ でも $n = 5000$ でも 16 分割であった。このことから，行列 B の最適分割数は大きい傾向があるように見て取れる。よって，最適分割数は対象行列の性質に依存して大きく変化することが窺える。

表 7　分割数による実行時間の変化：行列 A

分割数	3	4	5	6	10	16	20
$n = 10000$ (秒)	3.256	3.346	3.504	3.583	3.917	3.897	3.955
$n = 5000$ (秒)	1.953	2.003	2.002	2.019	1.968	1.947	1.955

表 8　分割数による実行時間の変化：行列 B

分割数	3	4	5	6	10	16	20
$n = 10000$ (秒)	19.451	17.245	16.327	15.458	14.944	14.060	15.357
$n = 5000$ (秒)	3.405	2.995	3.002	2.716	2.682	2.509	2.848

表 9　並列プログラムの計算結果の精度：行列 A, 問題サイズ $n = 10000$

アルゴリズム		相対残差 ϵ_R	直交誤差 ϵ_O
LAPACK	MRRR 法	1.99×10^{-14}	5.36×10^{-12}
	分割統治法	6.84×10^{-15}	5.77×10^{-15}
研究手法	3 分割	5.49×10^{-15}	8.49×10^{-15}
	16 分割	5.84×10^{-15}	1.07×10^{-14}

6.3.3　計算結果の精度

表 9 は並列プログラムの計算結果の精度についてまとめた表で縦軸は各アルゴリズム，横軸は各項目である。各アルゴリズムを用いて，行列 A のサイズ 10000 について計測を行った。

表 9 より計算結果の精度について比較を行うと，残差について比較した場合，どのアルゴリズムも同等の精度で計算が行われている。直交誤差について LAPACK の分割統治法と比較した場合，同等の精度で計算が行えている。LAPACK の MRRR 法と比較した場合，研究手法の方がより小さな誤差で精度の高い計算が行えている。

6.3.4　多分割の分割統治法の実行時間の内訳

表 10 は，研究手法の並列プログラムにおける，各並列化項目別の計測結果の表である。縦軸は各分割数および各対象行列，横軸は各並列化項目である。計算時間の長い行列 B のサイズ 10000 について計測を行った。各並列化項目は以下の通りである。

小問題　「T_i の固有値問題計算」
固有値　「対角行列と低階数の摂動の和の固有値計算」
固有ベクトル　「対角行列と低階数の摂動の和の固有ベクトル計算」
行列積　「行列積計算」

表 10 より各並列化項目別に実行時間を計測した場合，行列積部分の台数効果が最も大きい。これは，ベンダによってチューニングを施されたライブラリを利用していることの恩恵である。固有値，固有ベクトル計算に関しては OpenMP による並列化を行ったが，16 台の台数効果は得られないまでも並列化効果が出ている。よって，行列積や「対角行列と低階数の摂動の和の固有値，固有ベクトル計算」は並列性が高く，多分割の分割統治法は並列化に向いているといえる。

表 10　多分割の分割統治法の実行時間の内訳：対象行列 B , $n = 10000$

分割数	プログラム	全体	小問題	固有値	固有ベクトル	行列積	その他
4 分割	逐次 (秒)	172.427	15.846	7.19	73.021	75.873	0.497
	並列 (秒)	17.245	4.936	0.756	6.908	4.592	0.053
	台数効果	9.999	3.210	9.511	10.570	16.523	9.377
16 分割	逐次 (秒)	120.54	1.679	9.766	86.924	21.693	0.478
	並列 (秒)	14.06	2.297	1.173	9.203	1.335	0.052
	台数効果	8.573	0.731	8.326	9.445	16.249	9.192

7.　まとめ

実対称固有値問題に対する多分割の分割統治法の並列化の第一段階として，OpenMP を用いた共有メモリ型の並列化を施した。次の 4 項目，分割した実対称三重対角行列の固有値問題計算，対角行列と低階数の摂動の和の固有値計算，対角行列と低階数の摂動の和の固有ベクトル計算，および実対称三重対角行列の固有ベクトルを求めるための行列積計算が全体の計算の大部分を占めているため，我々はこの 4 項目を中心に並列化を施した。

研究手法および LAPACK の MRRR 法や分割統治法の数値実験を行った結果，サイズ 10000 の行列では，研究手法が LAPACK の MRRR 法や分割統治法よりも高速に計算でき，研究手法は従来の分割統治法と並んで， MRRR 法よりも大きな台数効果が得られるという結果を得た。最適分割数は，並列計算においても逐次計算同様，行列の性質に依存して変化する結果が得られた。誤差評価の面では，残差は他のアルゴリズムと同等の精度を保持することができ，直交誤差は従来の分割統治法と並んで MRRR 法よりも高精度を実現できた。並列化を施したことにより，対角行列と低階数の摂動の和の固有値計算とその固有ベクトル計算，および，実対称三重対角行列の固有ベクトルを求めるための行列積計算の 3 項目の速度が向上された。以上のことから，サイズが 10000 程度の行列の固有値問題解法として，多分割の分割統治法は 2 分割の分割統治法や MRRR 法よりも高速であることが示された。

8.　今後の課題

8.1　デフレーション発生率の研究

多分割の分割統治法では逐次実行，並列実行ともに分割数によって実行時間は大きく異なる。このため，自動的に最適分割数を決定する方法が必要とされる。最適分割数はいくつかの要因によって決まっていると思われるが，その要因の一つにデフレーションの発生率があげられる。デフレーションの発生率が非常に低い場合，行列積の計算に時間を要すために最適分割数は大きくなり，逆に，デフレーションの発生率が高い場合,「対角行列と低階数の摂動の和の固有値，固有ベクトル計算」にかかる時間が重要となり，最適分割数は小さくなる傾向がある。このため，行列の分割を行う前に少ない計算量でデフレーションの発生率について知ることができたならば，最適分割数を検討することができる。このため，デフレーションの発生率について研究する必要がある。

8.2 プログラムの安定性

現在のプログラムでは，逐次版および並列版のプログラムで一部の問題に対して，正常に計算できないことがある。これは 0 に近い値の除算が数多く存在し，精度が確保できないことによる。また，逐次版プログラムで正常に計算できる問題でも並列版プログラムで正常に計算できないことがある。このため，アルゴリズムを改善し数値計算に適した方法を考案する必要や，安定に計算を進めるプログラムの作成が必要である。

8.3 並列化に伴う速度低下への対応

サイズの十分小さい問題に対して並列版 LAPACK ライブラリを用いると，逐次版よりも計算時間が掛かることがあった。この現象はライブラリ内部で並列化された処理が高速に計算できず，加えて，並列化に必要な処理時間が加算されたことによると思われる。表 10 の小問題の項目で台数効果が得られていないのはこのためである。本来この計算 (T_i の固有値問題計算) は並列性があり，並列効果が期待できる部分である。このため，この部分に関して改善できたならば今よりも高速に計算できるはずである。

8.4 分散メモリ型並列計算プログラムの開発

現在のプログラムは共有メモリ型並列計算プログラムであり，分散メモリ型並列計算プログラムへの実装は行っていない。SR11000 ではノードを複数利用した分散メモリ型並列計算も行うことができ，多分割の分割統治法を分散メモリ型に実装することで更なる高速化，および，対応できる問題の大規模化が期待できる。マシン性能を引き出すには分散メモリ型並列計算プログラムを開発する必要がある。

9. 付録

本節ではプログラムの実装を行う中で，高速化に成功したことやうまく高速化できなかったことについて記載する。

9.1 行列積の高速化

本研究では行列積の計算に BLAS ライブラリを用いた。BLAS を使う場合，必ずしも行列のデータの各列ベクトルデータがメモリ上で連続である必要はない。しかし，連続であるほうがキャッシュのアクセス性能があがる可能性があり，実際に「本問題の固有ベクトル計算」でメモリ上のデータを連続に並べた場合，並べない場合よりも行列積の計算が高速に行えた。

9.2 C 言語を用いた LAPACK の使用方法

C 言語から LAPACK を利用する際は注意が必要である。一つは関数名。ライブラリの作成に用いた言語と違う言語からそのライブラリを使用する時，環境によって関数名が異なる。SR11000 の環境で C 言語から LAPACK を呼ぶ場合は，関数名は全て小文字となる。二つ目はリンク方法。LAPACK が FORTRAN で書かれているため，コンパイルは C コンパイラで，リ

ンクは FORTRAN コンパイラで行わなくてはならない。リンク時には -lf90s オプションが必要である。三つ目はプロファイラが使えないこと。コンパイラにはプロファイル使用のためのオプションがついているが，コンパイル時に C 言語版のライブラリ呼び出しが書き込まれ，リンク時に FORTRAN 版をリンクしようとするためか，コンパイル出来なかった。

9.3 LAPACK を利用した並列プログラムの作成上のトラブル

本小節では LAPACK を利用した並列プログラムの作成に関し，問題のあった項目について説明し，本研究でとった対処法について述べる。

9.3.1 トラブル

プログラムの作成に当たり 3 つの問題点があった。以下，その問題点について説明する。

■並列版 LAPACK をサイズの小さい問題に使用した時の速度低下 前述の通り，サイズの十分小さい問題に対して並列版のライブラリを用いると，逐次版よりも計算時間が掛かる。このため，同一プログラム内で大規模な問題と小規模な問題を解く場合，並列版のリンクに注意する必要がある。

■LAPACK の逐次版, 並列版ライブラリの同時利用 前項で述べたように，サイズの小さい問題を解く際に並列版ライブラリを用いた場合，逐次版を用いた場合よりも遅いため，サイズの小さい問題を解く際は逐次版を用い，サイズの大きい問題を解く際は並列版を用いることが相応しい。しかし，SR11000 に備え付けの LAPACK ライブラリは逐次版でも並列版でも関数名は同一である。このため使い分けは出来ない。

■LAPACK のスレッドセーフ性 並列版 LAPACK を使用した場合には一部で速度低下が発生するため，OpenMP で並列化を行い，各スレッドで逐次版のライブラリを呼び出すことで高速化が期待できる。しかし，SR11000 上に備え付けの LAPACK ライブラリはスレッドセーフではない。このため，LAPACK 関数を並列に呼び出すコードを書くと正しい計算が行われない可能性がある。

9.3.2 対処法

以上 3 点の理由から本研究では並列版プログラムに並列版ライブラリのみを用いる。ただし，OpenMP を用いて並列化を行いたい部分に LAPACK 関数が含まれている場合，代用の関数を自作して置き換えた後並列化を行った。対象行列によっては全体の実行時間で逐次版よりも並列版の方が遅くなってしまうこともあるがこの問題は今後の課題とした。

9.4 LAPACK(BLAS) サービスへの要望

SR11000 に備え付けの LAPACK(BLAS) ライブラリはマシンに対して最低限のチューニングしか施されていないそうである。このため，十分にマシン性能を引き出せていなかったり並列化効果が上がらないことがある。また，SR11000 に備え付けの LAPACK(BLAS) ライブラリはスレッドセーフでないため，逐次ライブラリを並列に実行することができない。ライブラリのチューニングが強化されることを希望する。

参 考 文 献

桑島 豊, 重原 孝臣, 実対称三重対角固有値問題の分割統治法の拡張, 日本応用数理学会論文誌, 15 (2005), 89–115.

桑島 豊, 重原 孝臣, 実対称三重対角固有値問題に対する多分割の分割統治法の改良, 日本応用数理学会論文誌, 16 (2006), 453–480.

山本有作, 密行列固有値解法の最近の発展 (I) – Multiple Relatively Robust Representations アルゴリズム –, 日本応用数理学会論文誌, 15 (2005), 181–208.

William H. Press, 丹慶勝市 訳,『NUMERICAL RECIPES in C [日本語版]』,685 p, 技術評論社, 1999

OpenMP, http://www.openmp.org/

LAPACK (Linear Algebra PACKage),http://www.netlib.org/lapack/index.html

BLAS (Basic Linear Algebra Subprograms),http://www.netlib.org/blas/index.html

ATLAS(Automatically Tuned Linear Algebra Software),http://www.netlib.org/atlas/index.html