Vol.2015-HPC-152 No.13
2015/12/17

複素対称行列のハウスホルダ型三重対角化法について

村上 弘[1,a)]

概要：複素対称行列の標準固有値問題をその対称性を利用して解く既知の方法である 1) 複素ヤコビ回転による対角化，2) 複素三重対角行列に対する QL 反復法，3) 複素鏡映変換を用いるハウスホルダ型の三重対角化，4) 実鏡映変換を主に用いて三重対角化を行う Splitting 法，5) 複素ランチョス法，6) 複素対称行列に対するヤコビ・デビッドソン法，について紹介する．

キーワード：固有値，対角化，複素対称，三重対角，ハウスホルダ，Splitting 法，QL 反復，ヤコビ回転

On Householder-type Method to Tridiagonalize Complex Symmetric Matrices

Hiroshi Murakami[1,a)]

Abstract: We introduce several known methods which solve a complex standard symmetric matrix eigenproblem by using the symmetry: 1) Diagonalization by complex Jacobi rotations, 2) Diagonalization of a complex symmetric tridiagonal matrix by QL-iterations, 3) Householder-type tridiagonalization using complex reflectors, 4) Splitting method for tridiagonalization using mostly real reflectors, 5) Lanczos method, 6) Jacobi-Davidson method.

Keywords: Eigenvalue, diagonalization, complex symmetric, tridiagonal, Householder, splitting method, QL iteration, Jacobi rotation

1. はじめに

今回は，複素対称行列の標準固有値問題の数値解法について，既に知られている方法の中から幾つかを紹介する．

減衰を伴う振動，波動の伝搬（媒質中の波動，電磁波の伝搬）の問題やある種の量子力学の問題（散乱や反応など）では複素対称な行列 K，M を係数とする複素一般対称固有値問題 $K\mathbf{u} = \lambda M\mathbf{u}$ が現れる．行列 A が複素対称であるとは $A^T = A$ のことでエルミート対称 $A^H = A$ とは異なる．実対称の場合と同様に行列 M の複素コレスキ分解 $M = LL^T$ ができると，一般固有値問題は変換 $\mathbf{u} = L^{-T}\mathbf{v}$ で複素対称な行列 $A = L^{-1}KL^{-T}$ を係数とする標準固有値問題 $A\mathbf{v} = \lambda\mathbf{v}$ に帰着できる．以下では，複素対称行列の標準固有値問題 $A\mathbf{v} = \lambda\mathbf{v}$ だけを考察対象とする．

標準固有値問題の行列がエルミート対称の場合には固有値はすべて実数で，ユニタリ行列（常に条件が良い）による相似変換で固有値が並んだ対角行列にできる．しかし行列が複素対称の場合には固有値は一般には複素数で，しかもジョルダン標準形は必ずしも対角にはならない，つまり相似変換による対角化が不可能な場合もある．

> 複素対称行列に対する一般論が [6] の第 4 章'Hermitian and symmetric matrices' に出ている．たとえば任意の複素行列はある複素対称行列に相似であり，つまり任意のジョルダン標準形と相似な複素対称行列が存在する．
>
> 一般の複素行列を複素正則行列により相似変換する場合の標準形がジョルダン標準形であるのと同様に，複素対称行列にも複素直交行列により相似変換する場合の標準形がある [1]，[3]，[9]．

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

複素対称行列の固有値問題は，その対称性を利用せずに一般の複素行列の固有値問題として扱うことが可能で，その場合にはまずユニタリ行列による両側変換を繰り返して中間形であるヘッセンベルグ形（Hessenberg form）にまで相似変換する．行列の対称性を利用する場合は複素直交行列による両側変換を繰り返して中間形として複素対称三重対角形にまで相似変換をすることを試みる．

中間形として三重対角形に変換する処理はヘッセンベルグ形に相似変換する処理と比べて必要な記憶量は半分で，演算量も少ない．また中間形を三重対角形とする方がヘッセンベルグ形とするよりもその後で固有値を反復法で収束させて求める処理に必要な記憶参照量や演算量が少なくて済む．固有ベクトルを求める場合も三重対角形を経由した方が計算が容易になる．しかしユニタリ行列による変換の条件が常に良好であるのに対して複素直交変換の条件は幾らでも悪くなり得る．そのため複素対称性を用いる処理は計算の量的な面では有利であるが，計算結果の精度の保証の面では不利になる．

応用上の必要から，大規模な複素対称行列の固有値解析が行なわれ，その場合に対称性を有効利用することが実際に計算を遂行する上で重要となる場合があり，そのような解法が工夫されて使われて来た．

教科書や標準ライブラリは複素対称固有値問題には冷淡

Wilkinson,J.H. の古典的な著書 Algebraic Eigenvalue Problems[2] では，対称行列は複素行列の場合には実の場合のような良い性質を持たないとだけ簡潔に述べられており，複素対称行列の対称性を利用する固有値の計算法についての記述は無い．標準的な行列計算の教科書 Matrix Computations(第 3 版)[11] にも複素対称行列の固有値問題に対する記述は特に無い．

標準的な線形計算ライブラリである LAPACK にも複素対称行列専用に作られた固有値解法ルーチンは含まれていない．LAPACK FAQ[4] の中の想定質問 “Are there routines in LAPACK for the complex symmetric eigenproblem?” に対する回答の記述が参考になる．

2. 複素対称行列の標準固有値問題

N 次の複素対称行列 A の標準固有値問題 $A\mathbf{v} = \lambda\mathbf{v}$ を考察対象とする．

複素の問題を扱う場合，2 つの複素ベクトル $\mathbf{x}$，$\mathbf{y}$ の内積として，エルミート対称行列を扱う場合には「エルミート内積」$(\mathbf{x},\mathbf{y})_H = \mathbf{x}^H\mathbf{y}$ を用いるが，複素対称行列を扱う場合には「複素内積」$(\mathbf{x},\mathbf{y}) = (\mathbf{y},\mathbf{x}) = \mathbf{x}^T\mathbf{y}$ を用いる（以下単に内積と呼ぶが，文献によっては dot 積と呼んで内積と呼ばない流儀もある）．内積が零の 2 つの複素ベクトルは互いに直交であるとする．実の場合と同様に，行列が対称であることから固有値の異なる 2 つの固有ベクトルは互いに直交する．自己と直交しない複素ベクトルは定数を乗じることで $\mathbf{v}^T\mathbf{v} = 1$ と正規化できる．N 次の複素対称行列 A は固有値に重複がなければ，その N 個の固有ベクトルはすべて正規化できる．そこでいま A の固有値に重複がなければ，正規化した N 個の固有ベクトルを並べた行列 V は複素直交行列（$V^TV = I$）になる．固有値を V の固有ベクトルに対応して並べた対角行列を Λ とすれば，$AV = V\Lambda$ である．結局，複素対称行列 A は固有値に重複がなければ，正規化された固有ベクトルを並べた複素直交行列 V による相似変換で $V^TAV = \Lambda$ と対角化される．

対角化不能な行列の例

実対称の場合とは異なり，複素対称な行列は対角化ができない場合がある．例：2 次の対称行列 $\begin{pmatrix} 1 & i \\ i & -1 \end{pmatrix}$ は重複する固有値 0 を持ち，そのジョルダン標準形は $J = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ なので対角化はできない．固有ベクトルは $\mathbf{z} = (1, i)^T$ の 1 個だけで，自己と直交しているから正規化はできない．

正規化可能な固有ベクトル $\mathbf{v}$ に対する固有値 λ の条件数は，$\kappa(\lambda) = ||\mathbf{v}||_2^2/|\mathbf{v}^T\mathbf{v}|$ である．条件数の大きい固有値は摂動に敏感で（精度を固定した通常の）数値計算では精度良く求めることはできない．複素ベクトルは自己との内積が 1 に正規化されていても，そのノルム（たとえば 2 乗ノルム $||\cdot||_2$ や最大ノルム $||\cdot||_\infty$ など）の値は非常に大きい可能性がある．

2.1 複素対称固有値問題の複素直交行列による変換

複素対称固有値問題の複素直交行列による変換の議論は，実対称固有値問題の実直交行列による変換の議論と同様になる．

いま U を任意の複素直交行列（$U^TU = I$）とすると，複素対称行列 A を U で両側変換した $B = U^TAU$ は A と相似で，B も複素対称行列である．A の固有ベクトル $\mathbf{x}$ は $A\mathbf{x} = \lambda\mathbf{x}$ で，$U^TAUU^T\mathbf{x} = \lambda U^T\mathbf{x}$ より $BU^T\mathbf{x} = \lambda U^T\mathbf{x}$ となるから $\mathbf{y} = U^T\mathbf{x}$ は B の固有値 λ の固有ベクトルになる．B の固有ベクトル $\mathbf{y}$ に対応する A の固有ベクトル $\mathbf{x}$ は逆変換 $\mathbf{x} = U\mathbf{y}$ で求めることができる．あるいは A の固有ベクトルを並べた行列 X とそれに対応して固有値を並べた対角行列を Λ とすれば，$AX = X\Lambda$ であることから $BU^TX = U^TX\Lambda$ となり，$Y = U^TX$ が B の固有ベクトルを並べた行列になる．

複素直交行列の積が複素直交行列になることを用いると，いま $A = A^{(0)}$ から出発して複素直交行列 $U^{(j)}$，j=1, 2, ..., k による両側変換を繰り返し適用して得られた行列 $A^{(j)} = U^{(j)\,T}A^{(j-1)}U^{(j)}$，$j$=1, 2, ..., k もまた複素対称になる．いま $W^{(j)} = U^{(1)}U^{(2)}\cdots U^{(j)}$ とおけば $W^{(j)}$ は複素直交行列で，$A^{(j)} = W^{(j)\,T}AW^{(j)}$ である．複素直交行列による両側変換は相似変換でもあるので，固有値の全体は不変に保たれる．両側変換で得られた行列 $B = A^{(k)}$ の固有

ベクトル $\mathbf{y}$ からそれに対応する A の固有ベクトル $\mathbf{x}$ を，変換 $U^{(j)}$ を逆順に適用して $\mathbf{x} = W^{(k)}\mathbf{y} = U^{(1)}U^{(2)}\cdots U^{(k)}\mathbf{y}$ と求めることができる．

3. 複素対称密行列に対するヤコビ法

複素対称密行列に対するヤコビ法の文献には [5] がある．通常の実対称密行列に対するヤコビ法と類似の算法であり，複素ヤコビ回転を繰り返すことで対角化を行う．ただし数値と演算を実数から複素数に置き換えるだけの単純な算術的拡張ではない（毎回の複素ヤコビ回転は行列の非対角部分のノルムを減少させるように決める．実の場合は直交回転で非対角成分の 2 乗和の値がピボット要素の値の 2 乗（の 2 倍）だけ減少するが，複素の回転の場合には実の回転以外に伸縮の効果も入るため，実の場合のようなピボット要素を消去するだけの方針ではピボットの行あるいは列のノルムに対する寄与が増大する可能性があるので，単純な拡張では収束性が保証されない）．固有値に重複がなければこの算法は（最終的には）2 次収束する．

3.1 複素対称密行列に対するヤコビ法の実験例

複素対称密行列に対するヤコビ法を，CPU が intel Core i7-2600K（3.4GHz，4 コア，HT=OFF，TB=OFF）の計算機システム上で IEEE 64bit 倍精度の数値と演算で 1 コアだけを用いて 1 スレッドで計算して実験した．使用したコンパイラは intel fortran version 15.0.0（intel x86_64 Linux 用）である．測定した経過時間を**表 1** に，その経過時間をグラフにプロットしたものを**図 1** にそれぞれ示す．

テストに用いた複素対称行列は，独立な要素の実部と虚部をそれぞれ区間 $[-1, 1]$ 上の一様分布乱数で生成したものである（乱数行列は固有値が近接しない傾向を持つので，計算を行う上で都合が良い状況になっているであろう）．

実験に用いたプログラムは，文献 [5] に掲載されている ALGOL60 のソースコードを Fortran90 用に書き換えたものであり，まだ改良できる余地があるから，ここで掲げている経過時間はあくまでも一応の参考程度である．

表 1 複素対称ヤコビ法の実験例（intel Core i7-2600K（1 スレッド））

N	経過時間（秒）	N	経過時間（秒）
100	6.65E-2	900	7.54E+1
200	5.70E-1	1,000	1.10E+2
300	2.07E+0	1,500	4.15E+2
400	5.17E+0	2,000	1.02E+3
500	1.03E+1	3,000	3.53E+3
600	2.22E+1	5,000	1.81E+4
700	3.21E+1	6,000	3.16E+4
800	7.04E+1		

実対称の場合と同様に，ヤコビ法は記憶参照量が多く記憶参照の局所性もなく，またさらに（行列の行と列の両方向への参照が最内側で発生するので）アクセスが不連続に行われるので記憶転送の効率も悪いため，行列の次数が少し大きくなると計算の処理速度が遅くなるのが欠点である．

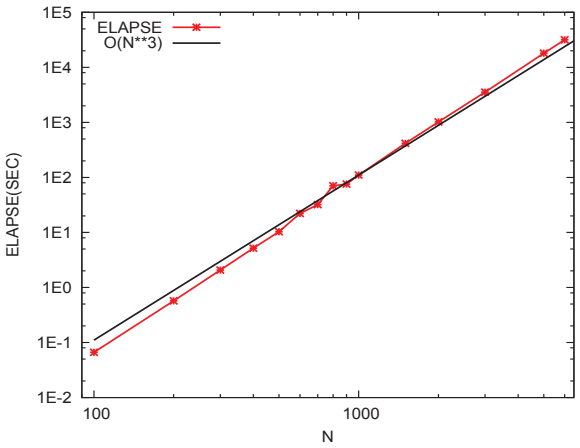

図 1 複素対称ヤコビ法の経過時間（秒）（intel Core i7-2600K（1 スレッド））

4. 複素対称三重対角行列用の QL 反復法

既約な複素対称三重対角行列の対角化には，実対称三重対角行列用の QL 反復法（QR 反復法の変種）[20] と類似の算法が適用できる（[7] の第 6 章 4 節'QL Algorithm, complex symmetric tridiagonal matrices' を参照）．また QL 法の計算コードの例 `CMTQL1` は，文献 [8] の第 7 章で頁 504–507（NETLIB 版）にある（これは Fortran で書かれており，注釈を含めて全部で 146 行である）．複素対称な場合の QL 法の計算は実対称やエルミート対称の場合とほぼ同様になるが，それでもやはり複素対称の場合は不都合な状況が生じる可能性があり，文献 [10] にはそのような状況に対する記述がある．

行列の次数が N の場合 QL 法の演算量はすべての固有値を求めるだけなら $O(N^2)$ で高速である．すべての固有ベクトルもこの QL 反復法の中で回転の蓄積により求めると計算量は $O(N^3)$ になる．すべての固有値だけを先に求めておいてそれから逆反復法により各固有値の固有ベクトルを求めると，固有対あたりの計算量は $O(N)$ で全体では $O(N^2)$ となるので高速に計算できる．ただし実対称の場合などと同様に，固有値の分布状況によっては近接する固有値を持つ固有ベクトルの直交性のずれを補正する（部分）再直交化が必要になると計算量が $O(N^3)$ 程度にまで大きくなり記憶参照量も多いので，むしろ再直交化が経過時間を支配するようになる（もしも対称三重対角行列の固有値問題に対する文献 [12] に記述されているものと同様の方法が複素対称の場合にも困難なく適用できるのであれば，すべての固有ベクトルを数値的にほぼ完全な直交性を維持して求める計算量は固有値の分布状況に依らずに $O(N^2)$ になる）．

4.1 複素対称三重対角行列用の QL 反復法の実験例

数値実験例として，複素対称な三重対角行列に対する QL 反復法を，CPU が intel Core i7-2600K（3.4GHz，4 コア，HT=OFF，TB=OFF）の計算機システム上で IEEE 64bit 倍精度の数値と演算で 1 コアだけを用いて 1 スレッドで計算した．使用したコンパイラは intel fortran version 15.0.0（intel x86_64 Linux 用）である．測定した経過時間の数値を**表 2** に，その経過時間をグラフにプロットしたものを**図 2** に示す（この実験例では QL 法の回転の蓄積により固有ベクトルを求めている）．テストに用いた対称三重対角行列の要素は主対角と下側副対角の値の実部と虚部をそれぞれ区間 $[-1, 1]$ 上の一様乱数で与えた（乱数で生成された行列は固有値が近接しない傾向を持つので，計算を行う上で極めて都合の良い状況となっているであろう）．

表 2 QL 反復法の実験例（intel Core i7-2600K（1 スレッド））

	固有値だけ	固有値と固有ベクトル			
N	経過時間（秒）	経過時間（秒）	条件数の最大	正規直交性の誤差	残差のノルムの最大
10	2.80E-5	2.90E-5	2.4E+0	3.6E-15	3.0E-14
30	1.79E-4	2.34E-4	8.5E+0	1.0E-14	1.1E-13
100	1.93E-3	4.42E-3	3.6E+0	5.6E-14	1.1E-12
300	1.67E-2	8.34E-2	6.4E+0	6.9E-14	6.4E-12
1,000	1.80E-1	3.03E+0	1.4E+1	1.8E-13	4.1E-11
3,000	1.58E+0	7.90E+1	8.1E+0	2.2E-12	1.3E-10
10,000	1.72E+1	2.83E+3	1.6E+1	1.9E-12	2.6E-08
30,000	1.48E+2	7.33E+4	2.0E+1	4.2E-12	2.7E-08

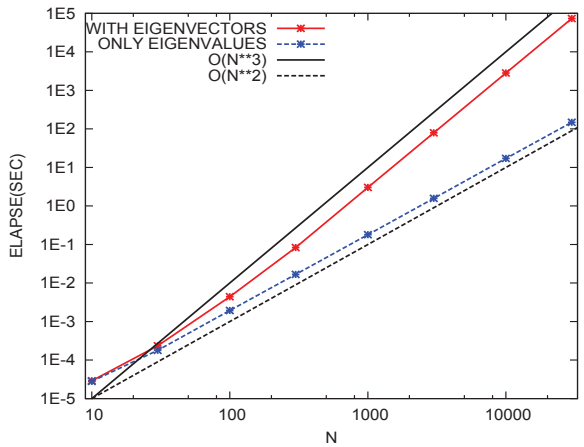

図 2 QL 反復法の経過時間（秒）（intel Core i7-2600K（1 スレッド））

5. 複素対称密行列用のハウスホルダ型三重対角化

複素対称密行列用のハウスホルダ型三重対角化は実の鏡映変換を複素の場合にそのまま拡張した鏡映変換を用いる．

複素ベクトル $\mathbf{u}$ が自己と直交しない（$\mathbf{u}^T\mathbf{u} \neq 0$）ときには，$H = I - (\frac{2}{\mathbf{u}^T\mathbf{u}})\,\mathbf{u}\mathbf{u}^T$ とおくと，$H^T = H$ と $H^TH = I$ が成り立つので，H は複素対称な複素直交変換で，H の逆行列は H それ自身である（複素鏡映変換）．

複素対称行列 A に対して $B = HAH = H^TAH$ を作れば B も複素対称で，$B = H^{-1}AH$ でもあるから B は A と相似であるので固有値の全体が一致して，B の固有ベクトルと A の固有ベクトルは変換 H で相互に入れ替わる．

このことから実対称行列に対するハウスホルダ法の場合と同様に，複素対称の行列に対して両側からの複素鏡映変換の適用を繰り返すと，途中で破綻が生じなければ複素対称の三重対角行列が得られる．計算が破綻するのは，鏡映の軸になるべき非零ベクトル $\mathbf{u}$ が自己と直交している $\mathbf{u}^T\mathbf{u} = 0$ 状況に遭遇したときである．完全な破綻ではなくてもベクトル $\mathbf{u}$ の条件 $\kappa(\mathbf{u}) = ||\mathbf{u}||_2^2/|\mathbf{u}^T\mathbf{u}|$ の値が大きいと，複素直交変換は悪条件になり数値誤差が拡大されて結果の精度が失なわれる．

5.1 複素対称密行列用のハウスホルダ型三重対角化の実験例

複素対称密行列に対して複素直交行列の鏡映を用いるハウスホルダ型の三重対角化の実験例を示す．使用した計算機システムの CPU は intel Core i7-2600K（3.4GHz，4 コア，HT=OFF，TB=OFF）である．IEEE 64bit 倍精度の数値と演算で 1 コアだけを用いて 1 スレッドで計算した．使用したコンパイラは intel fortran version 15.0.0（intel x86_64 Linux 用）である．各処理のステップの経過時間を**表 3** に，計算結果の品質に関する量を**表 4** に，三重対角化のステップだけの経過時間をグラフにプロットしたものを**図 3** に示す．実験に用いた複素対称行列はその対称性から対角を含めた下半分の要素の実部と虚部をそれぞれ，区間 $[-1, 1]$ 上の一様乱数で与えた（通常，乱数で生成された例題は数値的な条件は悪くない）．

注：固有ベクトルは $\mathbf{v}^T\mathbf{v} = 1$ となるように正規化した．その場合には固有値の条件数は $||\mathbf{v}||_2{}^2$ で，固有対に対する残差は $\mathbf{r} = A\mathbf{v} - \lambda\mathbf{v}$ である．残差のノルムには最大ノルム $\max_i |r_i|$ を用いた．

表 3 ハウスホルダ型三重対角化の経過時間（秒）（intel Core i7-2600K（1 スレッド））

N	三重対角化	QL-反復	逆変換
10	7.15E-6	2.69E-5	2.86E-6
30	3.39E-5	2.76E-4	3.70E-5
100	5.28E-4	4.65E-3	8.42E-4
300	1.23E-2	8.45E-2	1.98E-2
1,000	4.80E-1	3.05E+0	7.01E-1
3,000	1.67E+1	8.06E+1	2.29E+1
10,000	6.51E+2	3.02E+3	1.01E+3

表 4　ハウスホルダ型三重対角化の結果の品質（intel Core i7-2600K（1 スレッド））

N	条件数の最大	正規直交性の誤差	残差のノルムの最大
10	3.33E+0	3.45E-15	1.69E-14
30	4.94E+0	2.76E-13	3.22E-12
100	1.27E+1	9.67E-13	5.61E-11
300	2.23E+1	3.07E-12	5.48E-10
1,000	9.22E+1	1.20E-10	4.85E-07
3,000	1.43E+2	5.98E-10	9.13E-06
10,000	3.39E+2	1.38E-08	7.43E-03

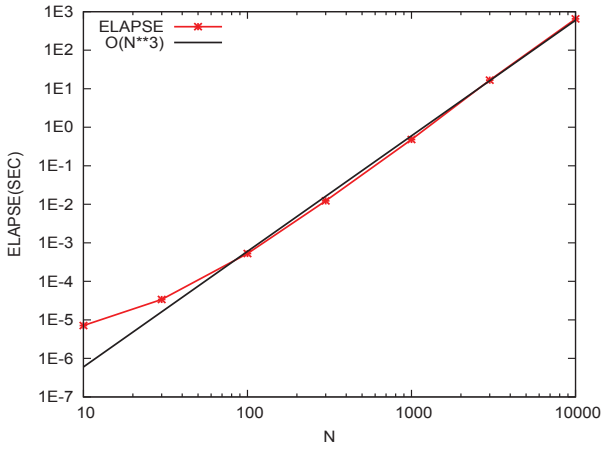

図 3　ハウスホルダ型三重対角化の経過時間（秒）（intel Core i7-2600K（1 スレッド））

6. ハウスホルダ型三重対角化の数値的性質を改善する Splitting 法

Splitting 法の参考文献には [13]，[14]，[15]，[16]，[21] がある．

実対称行列の場合の実鏡映変換の繰り返しで三重対角化を行うハウスホルダ法と同様に，複素対称行列の場合のハウスホルダ型の三重対角化法は複素鏡映を用いた両側変換で三重対角の外側の行列要素を順次消去していく．ただし複素鏡映変換は数値的に悪条件となる場合がある．そこで Splitting 法では処理を以下のように変更している

(1) まず複素行列 A のピボット列の実部に着目して，数値安定な実の鏡映変換 P を用いて $A' = PAP$ のピボット列の実部を三重対角形にする．

(2) 次に A' のピボット列の実部の非零構造を保って，数値安定な実の鏡映変換 Q を用いて $A'' = QA'Q$ のピボット列の虚部を 5 重対角形にする．（あるいは先に虚部を三重対角形にして，次に実部を 5 重対角形にすることもできる）．

(3) さらに A'' のピボット列の対角要素の直下にある 2 個の要素に対して，下側のものを消去して三重対角形にするように 2 次の複素直交回転 G を決めて，$B = GA''G^T$ とする．

処理の概略を**図 4** に示す．この三重対角化のための各段では，ほとんどの演算量を上記のステップ 1)，2) の実の鏡映変換による操作が占めており，誤差を拡大させる可能性があるステップ 3) の演算量は少ない．実際には，2 次の複素直交回転の段階で破綻が生じる可能性がある．文献 [16] では，Splitting 法の中の 2 次の複素直交回転行列のノルムの大きさを三重対角化の最中に観察することを提案している．文献 [13] では破綻が起きるあるいは破綻に近くなる場合に前に戻ってなるべく回避するような操作を加える方法が述べられている．今回の以下の実験例では，そのような観察や回復による対策を取り入れずにただ単純に計算を行なっている．

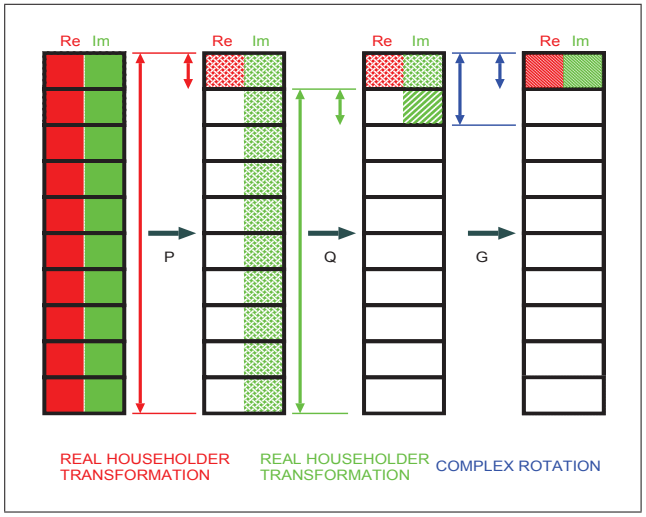

図 4　Splitting 法の概要

6.1 Splitting 法の実験例

複素対称密行列の三重対角化を行うのに際して，複素直交変換（複素鏡映）を直接適用して Splitting 法を使用しなかった場合と Splitting 法を適用して大部分の計算を実直交変換（実鏡映）で組み立てた場合の両方について，三重対角行列を得るまでの経過時間を**表 5** に，その経過時間をグラフにプロットしたものを**図 5** に示す．使用した計算機システムの CPU は intel Core i7-2600K（3.4GHz，4 コア，HT=OFF，TB=OFF）であり，IEEE 64bit 倍精度の数値と演算で 1 コアだけを用いて 1 スレッドで計算した．使用したコンパイラは intel fortran version 15.0.0 （intel x86_64 Linux 用）である．テストに用いた複素対称行列はその対称性から下半分の各行列要素の実部と虚部をそれぞれ区間 $[-1, 1]$ の一様乱数で生成したものである．実験に用いたプログラムは Splitting 法を適用する場合もしない場合もどちらも同程度に簡単な実装であるので，実測された経過時間の値はまだ短縮ができるようにも思われるが，複素あるいは実の鏡映による両側変換の処理は level-2 BLAS

であるから主に記憶転送のバンド幅が処理性能を支配していると思われる．

実験に用いた Splitting 法の現在の実装では，複素密行列 A を複素数を要素とする配列で保持しており実部と虚部を別々の配列に分けて保持してはいない（実部と虚部を別々の配列にして保持すると，実の鏡映による両側変換は実部と虚部の配列に対してそれぞれ独立に並行処理ができる）．複素鏡映による両側変換を Splitting 法による変換に置き換えると，内部では実の鏡映変換による両側変換を 2 回行う処理がほとんどを占める．今回の実装ではその実の鏡映変換による両側変換 2 回を重ねずに完全に逐次に適用して処理しているので，行列 A を格納している配列への走査の回数が 2 倍に増えている．そのため，表 5 からもわかるように，Splitting 法を適用した場合は適用しない場合と比べて経過時間が 1.6 倍から 1.7 倍程度に増加している．

表 5 Splitting 法の適用の有無と経過時間（秒）
（intel Core i7-2600K（1 スレッド））

N	適用なし	適用あり
10	7.15E-6	1.10E-5
30	3.39E-5	5.91E-5
100	5.28E-4	8.50E-4
150	1.63E-3	2.61E-3
300	1.23E-2	2.01E-2
500	5.47E-2	9.12E-2
700	1.46E-1	2.37E-1
1,000	4.80E-1	7.67E-1
1,500	1.97E+0	3.29E+0
3,000	1.67E+1	2.82E+1
5,000	7.76E+1	1.31E+2
10,000	6.51E+2	1.06E+3

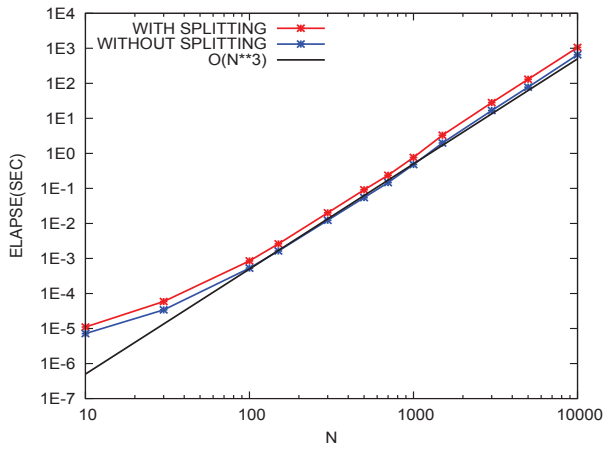

図 5 Splitting 法の適用の有無と経過時間（秒）
（intel Core i7-2600K（1 スレッド））

Splitting 法の採用の有無による結果の精度への影響の違いを比較するために以下の方法を採った．まず元の複素対称行列 A について「標準解法」を用いて固有値をすべて求めてそれを A の真の固有値の代わりの「基準値」とした．（ここでいう「標準解法」とは複素一般行列を安定なユニタリ変換の複素鏡映変換を用いてヘッセンベルグ形にし，次に QR 反復法により上三角形にすることで固有値をすべて求める方法である．しかも今回この「標準解法」の中ではすべての計算を四倍精度で行った．）つぎに Splitting 法を用いる場合と用いない場合のそれぞれについて A の三重対角化を倍精度計算で行って複素対称三重対角行列 T を作る．そうして「標準解法」を用いて T の固有値をすべて求めた．両方それぞれの場合について，得られた T の固有値を「基準値」と対応させ比較することで変動の大きさを求めて，それを三重対角化の計算法による固有値の変動の大きさであるとして，変動の大きさの最大値を求めた．得られた変動の大きさの最大値を**表 6** に，それをグラフにプロットしたものを表 6 に示す．これらの表とグラフから，固有値の変動の大きさは行列の次数 N の増加に伴って増大する傾向を示すが，Splitting 法を適用した場合の変動の大きさは Splitting 法を適用しない場合に比べて増大傾向が緩やかであり，同じ次数で比較すると固有値の変動が小さいことが分かる．

表 6 Splitting 法の適用の有無と固有値の変化の最大値
（intel Core i7-2600K（1 スレッド））

N	適用なし	適用あり
10	5.60E-14	5.52E-14
30	1.63E-13	1.72E-13
100	9.48E-11	5.68E-12
150	2.75E-10	2.85E-11
300	9.58E-10	1.16E-11
500	6.27E-09	2.05E-10
700	1.52E-08	1.89E-11
1,000	5.91E-07	5.28E-11
1,500	3.56E-06	7.07E-11
3,000	2.76E-05	1.47E-09
5,000	8.91E+01	7.44E-09
10,000	1.18E+02	1.51E-08

7. 複素対称行列に対するランチョス法

複素対称行列に対するランチョス法（Lanczos 法）は，実対称の場合のランチョス法の拡張であり，行列 A の固有値問題を（クリロフ空間の中の複素直交変換により相似変換して）三重対角行列 T の固有値問題に帰着させる [7]．

通常の実対称の場合のランチョス法と同様に，算法は行列ベクトル積を用いて組み立てられており，行列 A 自身を変形しないのでフィルイン（fill-in）は発生しない．また収束性を持つ反復法として使用することもできる．そのため A が特に疎行列である場合には計算量や記憶量の面で有利になる（しかし数値的安定性が適用上の問題点になることも実対称の場合と同様である）．多くの重要な応用に於い

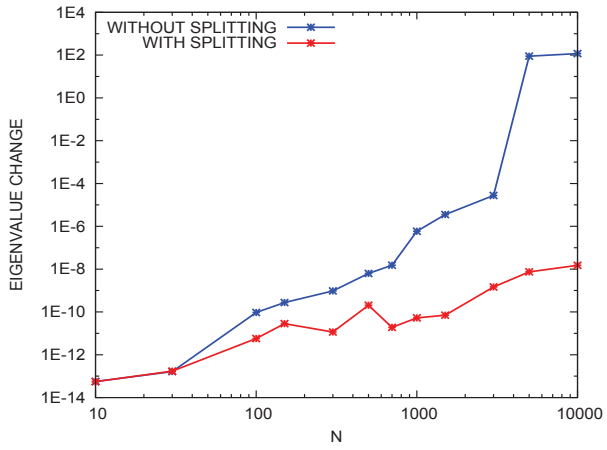

図 6 Splitting 法適用の有無と固有値の変化の最大値
（intel Core i7-2600K（1 スレッド））

て扱う行列は次数は大きいが非零の割合の少ない疎行列であり，帯行列として扱うには帯幅が広すぎたり，あるいはランダムに近い非零パターンを持つなどの場合には，密行列（あるいは帯行列で帯内が密）の場合のような行列変形を行なう方法は必要な記憶量や演算量が過大となるので適用ができない．そのような場合にはランチョス法のような反復法を選択せざるを得ない．

実の場合と同様に，ランチョス法の算法はブロック化することで，行列 A の走査回数が減らせて収束性も強まる [18]，[19]．

8. 複素対称行列に対するヤコビ・デビッドソン法

疎行列に対する反復解法であるヤコビ・デビッドソン法（Jacobi-Davidson 法）も複素対称行列に対して拡張がなされている．参考文献には [17]，[22] などがある．

9. まとめ

今回は，複素対称行列に対する標準固有値問題の解法として，行列の対称性を利用して解く既存の方法のいくつかを調べて紹介し，簡単な計算実験を実際に行なってみた．

実対称の場合とは異なり，複素対称の場合にはもともと対角化のできない問題である場合がある．複素対称行列の固有値問題では固有ベクトルが自己との内積が零あるいは零に近い場合にはその固有値は悪条件であり，摂動の影響により大きく変化するので数値計算では良い精度で求めることはできない．

複素対称密行列に対しては行列の対称性を利用せずに一般の複素密行列用の標準的な解法を適用することができる．標準的な解法では複素鏡映を用いた変換を繰り返すことでまず中間形であるヘッセンベルグ形にして，そこからさらに複素ギブンス回転を用いた QL 反復法により上三角形（シューア形）にすることで固有値を求める．この場合に用いる複素鏡映も複素ギブンス回転もユニタリ変換であり条件数は 1 なので数値的に安定な計算ができる．

複素対称な密行列は実対称の場合と同様に行列の対称性を利用して両側からの複素直交変換を用いて三重対角形に変換することが（途中で破綻が起きなければ）可能である．対称性を利用する計算法には記憶容量や記憶参照量，演算回数が少ない利点がある．しかしそれと引き換えで結果の精度が落ちる可能性がある．実直交変換やユニタリ変換は常に数値的な条件が良いことが保証されているが，複素直交変換はそうではないので計算で得られた解の精度が悪い場合がある．

複素対称行列を三重対角化するハウスホルダ型の算法の各段に於ける複素鏡映変換を置き換えて Splitting 法と呼ばれる方法による実の鏡映変換 2 回と計算量の少ない 2 次の複素回転の合成にすることで，計算結果の精度低下の傾向が改善できて，合成された変換の条件の良否は 2 次の複素回転の条件だけに帰着される．

参考文献

[1] Gantmacher,F.R.: *The Theory of Matrices*, vol.2, Chelsea, 1959/1964. Chap.1,Sec.3:'Normal form of a complex symmetric matrix'.
[2] Wilkinson,J.H.: *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, 1965.
[3] Craven, B.D.: "Complex Symmetric Matrices", *J. Austral. Math. Soc.*, vol.10 (1969), pp.341–354.
[4] LAPACK FAQ
(URL = `http://www.netlib.org/lapack/faq.html`).
[5] Anderson,P.J. and Loizou,G.: "A Jacobi Type Method for Complex Symmetric Matrices", *Numer. Math.*, vol.25 (1976), pp.347–363.
[6] Horn,R.A. and Johnson,C.R.: *Matrix Analysis*, Cambridge Univ. Press, 1985.
[7] Cullum,J.K. and Willoughby,R.A.: *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, Vol.I: Theory, Chap.6:'Non Defective Complex Symmetric Matrices', Birkhäuser, Boston, 1985. および上記の SIAM からの 2002 年の復刻版.
[8] Cullum,J.K. and Willoughby,R.A.: *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, Vol.II: Documentation and Listings Original Lanczos Codes, Chap.7:'Nondefective Complex Symmetric Matrices', Birkhäuser, Boston, 1985. この第 2 巻の内容は NETLIB から url="http://www.netlib.org/lanczos/vol2.ps.gz" として配布されている.
[9] Scott,N.H.: "A New Canonical Form for Complex Symmetric Matrices", *Proc. R. Soc. Lond.* **A**, vol.441, no.1913 (1993), pp.625–640.
[10] Cullum,J.K. and Willoughby,R.A.: "A QL Procedure for Computing the Eigenvalues of Complex Symmetric Tridiagonal Matrices", *SIAM J. Matrix Anal. Appl.*, vol.17, no.1 (1996), pp.83–109.
[11] Golub,G.H. and Van Loan,C.F.: *Matrix Computations*, 3rd Ed., The Johns Hopkins University Press, Baltimore and London,1996.
[12] Dhillon,I.,S.:"A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenavlue/Eigenvector Problem", Dissertation in Computer Science in Graduate Division of Univ.

California, Berkeley, 1997.

[13] Bar-On,I. and Ryaboy,V.: "Fast Diagonalization of Large and Dense Complex Symmetric Matrices, with Applications to Quantum Reaction Dynamics", *SIAM J. Sci. Comput.*, vol.18, no.5 (1997), pp.1412–1435.

[14] Bar-On,I. and Paprzycki,M.: "High Performance Solution of the Complex Symmetric Eigenproblem", *Numerical Algorithms*, vol.18, no.2 (June,1998), pp.195–208.

[15] Bar-On,I. and Paprzycki,M.: "A Fast Solver for the Complex Symmetric Eigenproblem", *Computer Assisted Mechanics and Engineering Science*, vol.5, no.1 (1998), pp.85–92.

[16] Gansterer,W., Haunschmid,E.J. and Ueberhuber,C.W.: "Complex Symmetric Eigenproblems", *Report AURORA TR2000-07*, Vienna Univ. of Tech (April,2000).

[17] Arbenz, P. and Hochstenbach, M.E.: "A Jacobi-Davidson Method for Solving Complex Symmetric Eigenvalue Problems", *SIAM J. Sci. Comput.*, vol.25, no.5 (2004), pp.1655–1673.

[18] Qiao,S., Liu,G. and Xu,W.: "Block Lanczos Tridiagonalization of Complex Symmetric Matrices", Advanced Signal Processing Algorithms, Architectures and Implementations XV, Proc. of SPIE (2005).

[19] Thies,J.: "Parallel Iterative Solution of Complex Symmetric Eigenvalue Problems", *Master of Science Thesis*, Royal Institute of Technology, School of Computer Science and Communication, Stockholm, Sweden (2006).

[20] Press,W.H., Teukolsky,S.A., Vetterling,W.T. and Flannery,B.P.: *Numerical Recipes in C*, 3rd Ed., (§11.4, Eigenvalues and Eigenvectors of a Tridiagonal Matrix), Cambridge Univ. Press, 2007.

[21] Gansterer,W.N., Schabauer,H., Pacher,C. and Finger,N.: "Tridiagonalizing Complex Symmetric Matrices in Waveguide Simulations", Bubak,M.*et al.*(Eds.): *ICCS 2008, Part I*, LNCS, vol.5101, Springer-Verlag (2008), pp.945–954.

[22] Arbenz, P. and Chinellato O.: "On Solving Complex-Symmetric Eigenvalue Problems Arising in the Design of Axisymmetric VCSEL Devices", *Applied Numerical Mathematics*, vol.58 (2008), pp.381–394.

[23] Gansterer,W.N., Gruber,A.R. and Pacher,C.: "Non-splitting Tridiagonalization of Complex Symmetric Matrices", Allen,G.*et al.*(Eds.): *ICCS 2009, Part I*, LNCS, vol.5544, Springer-Verlag (2009), pp.481-490.