(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5017666号
(P5017666)

(45)発行日　平成24年9月5日(2012.9.5)　(24)登録日　平成24年6月22日(2012.6.22)

(51)Int.Cl.		F I		
G06F 17/16	(2006.01)	G06F	17/16	K
G06T 1/00	(2006.01)	G06T	1/00	A

請求項の数 16　(全 63 頁)

(21)出願番号	特願2008-528725 (P2008-528725)
(86)(22)出願日	平成19年1月31日 (2007.1.31)
(86)国際出願番号	PCT/JP2007/051575
(87)国際公開番号	WO2008/018188
(87)国際公開日	平成20年2月14日 (2008.2.14)
審査請求日	平成21年10月30日 (2009.10.30)
(31)優先権主張番号	特願2006-215660 (P2006-215660)
(32)優先日	平成18年8月8日 (2006.8.8)
(33)優先権主張国	日本国(JP)

特許法第３０条第１項適用　平成１８年６月２６日発行の情報処理学会研究報告等で発表

(73)特許権者　504132272
国立大学法人京都大学
京都府京都市左京区吉田本町３６番地１
(74)代理人　100115749
弁理士　谷川　英和
(72)発明者　中村　佳正
京都府京都市左京区吉田本町　国立大学法人京都大学大学院情報学研究科内
(72)発明者　坪井　洋明
京都府京都市左京区吉田本町　国立大学法人京都大学大学院情報学研究科内
(72)発明者　譽田　太朗
京都府京都市左京区吉田本町　国立大学法人京都大学大学院情報学研究科内

最終頁に続く

(54)【発明の名称】固有値分解装置、及び固有値分解方法

(57)【特許請求の範囲】

【請求項１】

対称３重対角行列Ｔが記憶される対角行列記憶部と、

前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、当該対称３重対角行列Ｔを２個の対称３重対角行列に分割して前記対角行列記憶部に蓄積し、その対称３重対角行列を２個の対称３重対角行列に分割して前記対角行列記憶部に蓄積する処理を、分割後の各対称３重対角行列があらかじめ決められた大きさ以下となるまで繰り返す行列分割部と、

前記あらかじめ決められた大きさ以下の各対称３重対角行列の固有値と、当該各対称３重対角行列の固有ベクトルからなる直交行列の一部の要素である行列要素とが記憶される固有値分解記憶部と、

前記あらかじめ決められた大きさ以下の各対称３重対角行列を前記対角行列記憶部から読み出し、前記各対称３重対角行列に対して固有値分解を行い、前記各対称３重対角行列の固有値と、前記各対称３重対角行列の固有ベクトルからなる直交行列の行列要素とを少なくとも算出し、当該固有値と当該行列要素とを前記固有値分解記憶部に蓄積する固有値分解部と、

前記対称３重対角行列Ｔの固有値が記憶される固有値記憶部と、

各対称３重対角行列の固有値と、行列要素とを前記固有値分解記憶部から読み出し、前記固有値と、前記行列要素とから、分割元の対称３重対角行列の固有値と、分割元の対称３重対角行列の行列要素とを算出して前記固有値分解記憶部に蓄積し、その分割元の対称３重対角行列の固有値と、行列要素とを算出する処理を、対称３重対角行列Ｔの少なくとも

１個の固有値を算出するまで繰り返し、前記対称３重対角行列Ｔの少なくとも１個の固有値を前記固有値記憶部に蓄積する固有値算出部と、
前記対称３重対角行列Ｔの固有ベクトルが記憶される固有ベクトル記憶部と、
前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、前記固有値記憶部から前記対称３重対角行列Ｔの固有値を読み出し、前記対称３重対角行列Ｔとその固有値とから、ツイスト分解法を用いて前記対称３重対角行列Ｔの少なくとも１個の固有ベクトルを算出して前記固有ベクトル記憶部に蓄積する固有ベクトル算出部と、を備えた固有値分解装置。
【請求項２】
前記固有ベクトル算出部は、ｑｄ型ツイスト分解法により固有ベクトルを算出する、請求項１記載の固有値分解装置。
【請求項３】
前記固有ベクトル算出部は、
前記固有値記憶部から前記対称３重対角行列Ｔの固有値を読み出し、当該固有値のいずれかが負の値である場合に、前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、当該対称３重対角行列Ｔの固有ベクトルを変化させることなく、当該対称３重対角行列Ｔのすべての固有値が正の値となるように、当該対称３重対角行列Ｔを正定値化し、正定値化した後の固有値を前記固有値記憶部に蓄積し、正定値化した後の対称３重対角行列Ｔを前記対角行列記憶部に蓄積する正定値化部と、
前記対角行列記憶部からすべての固有値が正の値である対称３重対角行列Ｔを読み出し、前記固有値記憶部から前記対称３重対角行列Ｔの正の値の固有値を読み出し、前記対称３重対角行列Ｔの各要素に関してｑｄ型変換を行うことによって、前記対称３重対角行列Ｔを上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）にコレスキー分解するコレスキー分解部と、
前記上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）の各要素と、前記対称３重対角行列Ｔの正の値の固有値とを用いて固有ベクトルを算出して前記固有ベクトル記憶部に蓄積するベクトル算出部と、を備えた、請求項２記載の固有値分解装置。
【請求項４】
前記固有ベクトル算出部は、ＬＶ型ツイスト分解法により固有ベクトルを算出する、請求項１記載の固有値分解装置。
【請求項５】
前記固有ベクトル算出部は、
前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、前記固有値記憶部から前記対称３重対角行列Ｔの固有値を読み出し、前記対称３重対角行列Ｔの各要素に関してミウラ変換、ｄＬＶｖ型変換、逆ミウラ変換を行うことによって、前記対称３重対角行列Ｔを上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）にコレスキー分解するコレスキー分解部と、
前記上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）の各要素と、前記対称３重対角行列Ｔの固有値とを用いて固有ベクトルを算出して前記固有ベクトル記憶部に蓄積するベクトル算出部と、を備えた、請求項４記載の固有値分解装置。
【請求項６】
前記コレスキー分解部は、複数のコレスキー分解手段を備え、
前記複数のコレスキー分解手段が、前記対称３重対角行列Ｔをコレスキー分解する処理を並列実行する、請求項３または請求項５記載の固有値分解装置。
【請求項７】
前記ベクトル算出部は、複数のベクトル算出手段を備え、
前記複数のベクトル算出手段が、固有ベクトルを算出する処理を並列実行する、請求項３、請求項５または請求項６記載の固有値分解装置。
【請求項８】
前記固有値算出部は、複数の固有値算出手段を備え、

前記複数の固有値算出手段が、分割元の対称３重対角行列の固有値と、行列要素とを算出する処理を並列実行する、請求項１から請求項７のいずれか記載の固有値分解装置。
【請求項９】
前記固有値分解部は、複数の固有値分解手段を備え、
前記複数の固有値分解手段が、対称３重対角行列に対して固有値分解を行う処理を並列実行する、請求項１から請求項８のいずれか記載の固有値分解装置。
【請求項１０】
対称行列Ａが記憶される行列記憶部と、
前記対称行列Ａを前記行列記憶部から読み出し、前記対称行列Ａを３重対角化した前記対称３重対角行列Ｔを算出して前記対角行列記憶部に蓄積する対角化部と、をさらに備えた請求項１から請求項９のいずれか記載の固有値分解装置。
【請求項１１】
前記行列分割部は、対称３重対角行列を略半分の２個の対称３重対角行列に分割する、請求項１から請求項１０のいずれか記載の固有値分解装置。
【請求項１２】
前記固有値算出部は、対称３重対角行列Ｔの全ての固有値を算出する、請求項１から請求項１１のいずれか記載の固有値分解装置。
【請求項１３】
前記固有ベクトル算出部は、対称３重対角行列Ｔの全ての固有ベクトルを算出する、請求項１２記載の固有値分解装置。
【請求項１４】
前記対称行列Ａは、顔画像データを示すベクトルの平均から算出された共分散行列である、請求項１０記載の固有値分解装置。
【請求項１５】
対称３重対角行列Ｔが記憶される対角行列記憶部と、行列分割部と、固有値分解部と、前記固有値分解部によって固有値分解された固有値と固有ベクトルからなる直交行列の一部の要素である行列要素とが記憶される固有値分解記憶部と、前記対称３重対角行列Ｔの固有値が記憶される固有値記憶部と、固有値算出部と、前記対称３重対角行列Ｔの固有ベクトルが記憶される固有ベクトル記憶部と、固有ベクトル算出部とを備えた固有値分解装置で用いられる固有値分解方法であって、
前記行列分割部が、前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、当該対称３重対角行列Ｔを２個の対称３重対角行列に分割して前記対角行列記憶部に蓄積し、その対称３重対角行列を２個の対称３重対角行列に分割して前記対角行列記憶部に蓄積する処理を、分割後の各対称３重対角行列があらかじめ決められた大きさ以下となるまで繰り返す行列分割ステップと、
前記固有値分解部が、前記あらかじめ決められた大きさ以下の各対称３重対角行列を前記対角行列記憶部から読み出し、前記各対称３重対角行列に対して固有値分解を行い、前記各対称３重対角行列の固有値と、前記各対称３重対角行列の固有ベクトルからなる直交行列の行列要素とを少なくとも算出し、当該固有値と当該行列要素とを前記固有値分解記憶部に蓄積する固有値分解ステップと、
前記固有値算出部が、各対称３重対角行列の固有値と、行列要素とを前記固有値分解記憶部から読み出し、前記固有値と、前記行列要素とから、分割元の対称３重対角行列の固有値と、分割元の対称３重対角行列の行列要素とを算出して前記固有値分解記憶部に蓄積し、その分割元の対称３重対角行列の固有値と、行列要素とを算出する処理を、対称３重対角行列Ｔの少なくとも１個の固有値を算出するまで繰り返し、前記対称３重対角行列Ｔの少なくとも１個の固有値を前記固有値記憶部に蓄積する固有値算出ステップと、
前記固有ベクトル算出部が、前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、前記固有値記憶部から前記対称３重対角行列Ｔの固有値を読み出し、前記対称３重対角行列Ｔとその固有値とから、ツイスト分解法を用いて前記対称３重対角行列Ｔの少なくとも１個の固有ベクトルを算出して前記固有ベクトル記憶部に蓄積する固有ベクトル算出ス

テップと、を備えた固有値分解方法。
【請求項１６】
コンピュータに、
対称３重対角行列Ｔが記憶される対角行列記憶部から前記対称３重対角行列Ｔを読み出し、当該対称３重対角行列Ｔを２個の対称３重対角行列に分割して前記対角行列記憶部に蓄積し、その対称３重対角行列を２個の対称３重対角行列に分割して前記対角行列記憶部に蓄積する処理を、分割後の各対称３重対角行列があらかじめ決められた大きさ以下となるまで繰り返す行列分割ステップと、
前記あらかじめ決められた大きさ以下の各対称３重対角行列を前記対角行列記憶部から読み出し、前記各対称３重対角行列に対して固有値分解を行い、前記各対称３重対角行列の固有値と、前記各対称３重対角行列の固有ベクトルからなる直交行列の行列要素とを少なくとも算出し、当該固有値と当該行列要素とを固有値分解記憶部に蓄積する固有値分解ステップと、
各対称３重対角行列の固有値と、行列要素とを前記固有値分解記憶部から読み出し、前記固有値と、前記行列要素とから、分割元の対称３重対角行列の固有値と、分割元の対称３重対角行列の行列要素とを算出して前記固有値分解記憶部に蓄積し、その分割元の対称３重対角行列の固有値と、行列要素とを算出する処理を、対称３重対角行列Ｔの少なくとも１個の固有値を算出するまで繰り返し、前記対称３重対角行列Ｔの少なくとも１個の固有値を固有値記憶部に蓄積する固有値算出ステップと、
前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、前記固有値記憶部から前記対称３重対角行列Ｔの固有値を読み出し、前記対称３重対角行列Ｔとその固有値とから、ツイスト分解法を用いて前記対称３重対角行列Ｔの少なくとも１個の固有ベクトルを算出して固有ベクトル記憶部に蓄積する固有ベクトル算出ステップと、を実行させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、固有値分解を行う固有値分解装置等に関する。
【背景技術】
【０００２】
固有値分解は、例えば、分子軌道法による化学計算、統計計算、情報検索等の非常に広い分野で利用されている。また、固有値分解の方法も知られている（例えば、非特許文献１参照）。
【非特許文献１】Ｊ．Ｊ．Ｍ．Ｃｕｐｐｅｎ，「Ａ ｄｉｖｉｄｅ ａｎｄ ｃｏｎｑｕｅｒ ｍｅｔｈｏｄ ｆｏｒ ｔｈｅ ｓｙｍｍｅｔｒｉｃ ｔｒｉｄｉａｇｏｎａｌ ｅｉｇｅｎｐｒｏｂｌｅｍ」、Ｎｕｍｅｒｉｓｃｈｅ Ｍａｔｈｅｍａｔｉｋ，Ｖｏｌ．３６、ｐ．１７７－１９５、１９８１年
【発明の開示】
【発明が解決しようとする課題】
【０００３】
近年、これらの応用分野におけるデータ量の増大などに伴って、高速・高精度な固有値分解が求められている。また、固有値分解を並列処理することができる固有値分解装置等の開発が望まれていた。
【０００４】
本発明は、上記状況の下になされたものであり、並列性に優れた高速、高精度な固有値分解を実行可能な固有値分解装置等を提供することを目的とする。
【課題を解決するための手段】
【０００５】
上記目的を達成するため、本発明による固有値分解装置は、対称３重対角行列Ｔが記憶される対角行列記憶部と、前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、

当該対称３重対角行列Ｔを２個の対称３重対角行列に分割して前記対角行列記憶部に蓄積し、その対称３重対角行列を２個の対称３重対角行列に分割して前記対角行列記憶部に蓄積する処理を、分割後の各対称３重対角行列があらかじめ決められた大きさ以下となるまで繰り返す行列分割部と、前記あらかじめ決められた大きさ以下の各対称３重対角行列の固有値と、当該各対称３重対角行列の固有ベクトルからなる直交行列の一部の要素である行列要素とが記憶される固有値分解記憶部と、前記あらかじめ決められた大きさ以下の各対称３重対角行列を前記対角行列記憶部から読み出し、前記各対称３重対角行列に対して固有値分解を行い、前記各対称３重対角行列の固有値と、前記各対称３重対角行列の固有ベクトルからなる直交行列の行列要素とを少なくとも算出し、当該固有値と当該行列要素とを前記固有値分解記憶部に蓄積する固有値分解部と、前記対称３重対角行列Ｔの固有値が記憶される固有値記憶部と、各対称３重対角行列の固有値と、行列要素とを前記固有値分解記憶部から読み出し、前記固有値と、前記行列要素とから、分割元の対称３重対角行列の固有値と、分割元の対称３重対角行列の行列要素とを算出して前記固有値分解記憶部に蓄積し、その分割元の対称３重対角行列の固有値と、行列要素とを算出する処理を、対称３重対角行列Ｔの少なくとも１個の固有値を算出するまで繰り返し、前記対称３重対角行列Ｔの少なくとも１個の固有値を前記固有値記憶部に蓄積する固有値算出部と、前記対称３重対角行列Ｔの固有ベクトルが記憶される固有ベクトル記憶部と、前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、前記固有値記憶部から前記対称３重対角行列Ｔの固有値を読み出し、前記対称３重対角行列Ｔとその固有値とから、ツイスト分解法を用いて前記対称３重対角行列Ｔの少なくとも１個の固有ベクトルを算出して前記固有ベクトル記憶部に蓄積する固有ベクトル算出部と、を備えたものである。

【０００６】

このような構成により、まず固有値を算出し、その固有値に基づいてツイスト分解法を用いて固有ベクトルを算出することによって、高速で高精度な固有値分解を実現することができうる。また、並列性にも優れている。さらに、全ての固有値や固有ベクトルを算出する必要がない場合には、必要な範囲で固有値や固有ベクトルを算出することができ、処理負荷を軽減することができる。

【０００７】

また、本発明による固有値分解装置では、前記固有ベクトル算出部は、ｑｄ型ツイスト分解法により固有ベクトルを算出してもよい。

このような構成により、高速で高精度な固有値分解を実現することができうる。また、並列性にも優れている。

【０００８】

また、本発明による固有値分解装置では、前記固有ベクトル算出部は、前記固有値記憶部から前記対称３重対角行列Ｔの固有値を読み出し、当該固有値のいずれかが負の値である場合に、前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、当該対称３重対角行列Ｔの固有ベクトルを変化させることなく、当該対称３重対角行列Ｔのすべての固有値が正の値となるように、当該対称３重対角行列Ｔを正定値化し、正定値化した後の固有値を前記固有値記憶部に蓄積し、正定値化した後の対称３重対角行列Ｔを前記対角行列記憶部に蓄積する正定値化部と、前記対角行列記憶部からすべての固有値が正の値である対称３重対角行列Ｔを読み出し、前記固有値記憶部から前記対称３重対角行列Ｔの正の値の固有値を読み出し、前記対称３重対角行列Ｔの各要素に関してｑｄ型変換を行うことによって、前記対称３重対角行列Ｔを上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）にコレスキー分解するコレスキー分解部と、前記上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）の各要素と、前記対称３重対角行列Ｔの正の値の固有値とを用いて固有ベクトルを算出して前記固有ベクトル記憶部に蓄積するベクトル算出部と、を備えてもよい。

【０００９】

このような構成により、負の値の固有値が含まれる場合には、対称３重対角行列Ｔを正定値化してからｑｄ型ツイスト分解法を実行することができる。また、ＬＶ型ツイスト分

解法よりも高速に計算することができうる。

【００１０】

また、本発明による固有値分解装置では、前記固有ベクトル算出部は、ＬＶ型ツイスト分解法により固有ベクトルを算出してもよい。

このような構成により、数値安定的に固有ベクトルの算出を行うことができうる。

【００１１】

また、本発明による固有値分解装置では、前記固有ベクトル算出部は、前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、前記固有値記憶部から前記対称３重対角行列Ｔの固有値を読み出し、前記対称３重対角行列Ｔの各要素に関してミウラ変換、ｄＬＶｖ型変換、逆ミウラ変換を行うことによって、前記対称３重対角行列Ｔを上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）にコレスキー分解するコレスキー分解部と、前記上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）の各要素と、前記対称３重対角行列Ｔの固有値とを用いて固有ベクトルを算出して前記固有ベクトル記憶部に蓄積するベクトル算出部と、を備えてもよい。

このような構成により、コレスキー分解の処理において、複数の補助変数を用いることによって、数値安定的に固有ベクトルの算出を行うことができうる。

【００１２】

また、本発明による固有値分解装置では、前記コレスキー分解部は、複数のコレスキー分解手段を備え、前記複数のコレスキー分解手段が、前記対称３重対角行列Ｔをコレスキー分解する処理を並列実行してもよい。

このような構成により、コレスキー分解の処理を短時間で行うことができうる。

【００１３】

また、本発明による固有値分解装置では、前記ベクトル算出部は、複数のベクトル算出手段を備え、前記複数のベクトル算出手段が、固有ベクトルを算出する処理を並列実行してもよい。

このような構成により、固有ベクトルを算出する処理を短時間で行うことができうる。

【００１４】

また、本発明による固有値分解装置では、前記固有値算出部は、複数の固有値算出手段を備え、前記複数の固有値算出手段が、分割元の対称３重対角行列の固有値と、行列要素とを算出する処理を並列実行してもよい。

このような構成により、固有値を算出する処理を短時間で行うことができうる。

【００１５】

また、本発明による固有値分解装置では、前記固有値分解部は、複数の固有値分解手段を備え、前記複数の固有値分解手段が、対称３重対角行列に対して固有値分解を行う処理を並列実行してもよい。

このような構成により、固有値分解する処理を短時間で行うことができうる。

【００１６】

また、本発明による固有値分解装置では、対称行列Ａが記憶される行列記憶部と、前記対称行列Ａを前記行列記憶部から読み出し、前記対称行列Ａを３重対角化した前記対称３重対角行列Ｔを算出して前記対角行列記憶部に蓄積する対角化部と、をさらに備えてもよい。

【００１７】

このような構成により、任意の対称行列Ａについて固有値を算出することができる。また、対称３重対角行列Ｔの固有ベクトルを用いることにより、対称行列Ａの固有ベクトルを算出することもできうる。

【００１８】

また、本発明による固有値分解装置では、前記行列分割部は、対称３重対角行列を略半分の２個の対称３重対角行列に分割してもよい。

このような構成により、並列処理を適切に行うことができうる。

【００１９】

また、本発明による固有値分解装置では、前記固有値算出部は、対称３重対角行列Ｔの全ての固有値を算出してもよい。

【００２０】

また、本発明による固有値分解装置では、前記固有ベクトル算出部は、対称３重対角行列Ｔの全ての固有ベクトルを算出してもよい。

【００２１】

また、本発明による固有値分解装置では、前記対称行列Ａは、顔画像データを示すベクトルの平均から算出された共分散行列であってもよい。

このような構成により、固有値分解装置を顔画像データの認識の処理に用いることができうる。

【発明の効果】

【００２２】

本発明による固有値分解装置等によれば、高速で高精度な固有値分解を行うことができる。また、並列性にも優れている。

【発明を実施するための最良の形態】

【００２３】

以下、本発明による固有値分解装置について、実施の形態を用いて説明する。なお、以下の実施の形態において、同じ符号を付した構成要素及びステップは同一または相当するものであり、再度の説明を省略することがある。

【００２４】

（実施の形態１）

本発明の実施の形態１による固有値分解装置について、図面を参照しながら説明する。

図１は、本実施の形態による固有値分解装置１の構成を示すブロック図である。図１において、本実施の形態による固有値分解装置１は、行列記憶部１１と、対角化部１２と、対角行列記憶部１３と、行列分割部１４と、固有値分解部１５と、固有値分解記憶部１６と、固有値算出部１７と、固有値記憶部１８と、固有ベクトル算出部１９と、固有ベクトル記憶部２０とを備える。

【００２５】

行列記憶部１１では、任意の対称行列Ａが記憶される。この対称行列Ａは、各要素が実数である実行列である。なお、対称行列Ａが記憶されているとは、対称行列Ａを示すデータが記憶されている、という意味である。後述する記憶部においても同様である。行列記憶部１１は、所定の記録媒体（例えば、半導体メモリや磁気ディスク、光ディスクなど）によって実現されうる。行列記憶部１１での記憶は、ＲＡＭ等における一時的な記憶でもよく、あるいは、長期的な記憶でもよい。行列記憶部１１に対称行列Ａが記憶される過程は問わない。例えば、記録媒体を介して対称行列Ａが行列記憶部１１で記憶されるようになってもよく、通信回線等を介して送信された対称行列Ａが行列記憶部１１で記憶されるようになってもよく、あるいは、キーボードやマウス等の入力デバイスを介して入力された対称行列Ａが行列記憶部１１で記憶されるようになってもよい。

【００２６】

対角化部１２は、対称行列Ａを行列記憶部１１から読み出し、その読み出した対称行列Ａを３重対角化した対称３重対角行列Ｔを算出する。そして、対角化部１２は、その算出した対称３重対角行列Ｔを対角行列記憶部１３に蓄積する。対角化部１２は、例えば、ハウスホルダー（Ｈｏｕｓｅｈｏｌｄｅｒ）変換を必要なだけ繰り返し行う方法や、ランチョス（Ｌａｎｃｚｏｓ）法を用いた方法、その他の３重対角化法を用いて、対称行列Ａを３重対角化する。

【００２７】

対角行列記憶部１３では、対称３重対角行列Ｔが記憶される。対角行列記憶部１３は、所定の記録媒体（例えば、半導体メモリや磁気ディスク、光ディスクなど）によって実現されうる。対角行列記憶部１３での記憶は、ＲＡＭ等における一時的な記憶でもよく、あるいは、長期的な記憶でもよい。

【００２８】
行列分割部１４は、対角行列記憶部１３から対称３重対角行列Ｔを読み出し、その対称３重対角行列Ｔを２個の対称３重対角行列に分割して対角行列記憶部１３に蓄積する。行列分割部１４は、その対称３重対角行列を２個の対称３重対角行列に分割して対角行列記憶部１３に蓄積する処理を、分割後の各対称３重対角行列があらかじめ決められた大きさ以下となるまで再帰的に繰り返す。
【００２９】
固有値分解部１５は、あらかじめ決められた大きさ以下の各対称３重対角行列を対角行列記憶部１３から読み出し、その各対称３重対角行列に対して固有値分解を行い、各対称３重対角行列の固有値と、その各対称３重対角行列の固有ベクトルを算出する。固有値分解部１５は、例えば、２分法と逆反復法を組み合わせた方法、ＭＲ＾３法、ＱＲ法等を用いて固有値分解を行ってもよい。ここで、ＭＲ＾３法による固有値分解を行う場合には、ＬＡＰＡＣＫ３．０（ｈｔｔｐ：／／ｗｗｗ．ｎｅｔｌｉｂ．ｏｒｇ／ｌａｐａｃｋ／）で提供されているＤＳＴＥＧＲを用いてもよい。また、ＱＲ法による固有値分解を行う場合には、ＬＡＰＡＣＫ３．０で提供されているＤＳＴＥＱＲを用いてもよい。これらの固有値分解の方法については、すでに公知であり、その詳細な説明を省略する。固有値分解部１５は、算出した固有値を固有値分解記憶部１６に蓄積する。また、固有値分解部１５は、算出した固有ベクトルからなる直交行列の一部の要素である行列要素も固有値分解記憶部１６に蓄積する。固有ベクトルからなる直交行列とは、固有ベクトルを各列とする直交行列のことである。行列要素の詳細については後述する。
【００３０】
固有値分解記憶部１６では、固有値分解部１５によって固有値分解された固有値と、前述の行列要素とが記憶される。また、固有値分解記憶部１６では、固有値算出部１７によって算出された固有値や行列要素も記憶される。固有値分解記憶部１６は、所定の記録媒体（例えば、半導体メモリや磁気ディスク、光ディスクなど）によって実現されうる。固有値分解記憶部１６での記憶は、ＲＡＭ等における一時的な記憶でもよく、あるいは、長期的な記憶でもよい。
【００３１】
固有値算出部１７は、固有値分解部１５によって算出された固有値と、行列要素とを固有値分解記憶部１６から読み出し、その固有値と行列要素とから、分割元の対称３重対角行列の固有値と、分割元の対称３重対角行列の行列要素とを算出して固有値分解記憶部１６に蓄積する。固有値算出部１７は、その分割元の対称３重対角行列の固有値と、行列要素とを算出する処理を、対称３重対角行列Ｔの固有値を算出するまで再帰的に繰り返す。そして、固有値算出部１７は、算出した対称３重対角行列Ｔの固有値を固有値記憶部１８に蓄積する。
【００３２】
固有値記憶部１８では、対称３重対角行列Ｔの固有値が記憶される。固有値記憶部１８は、所定の記録媒体（例えば、半導体メモリや磁気ディスク、光ディスクなど）によって実現されうる。固有値記憶部１８での記憶は、ＲＡＭ等における一時的な記憶でもよく、あるいは、長期的な記憶でもよい。
【００３３】
固有ベクトル算出部１９は、対角行列記憶部１３から対称３重対角行列Ｔを読み出し、固有値記憶部１８から対称３重対角行列Ｔの固有値を読み出す。そして、固有ベクトル算出部１９は、対称３重対角行列Ｔとその固有値とから、ツイスト分解法を用いて対称３重対角行列Ｔの固有ベクトルを算出して固有ベクトル記憶部２０に蓄積する。固有ベクトル算出部１９は、ＬＶ型ツイスト分解法により固有ベクトルを算出してもよく、あるいは、ｑｄ型ツイスト分解法により固有ベクトルを算出してもよい。なお、具体的なツイスト分解の手法については後述する。
【００３４】
固有ベクトル記憶部２０では、対称３重対角行列Ｔの固有ベクトルが記憶される。固有

ベクトル記憶部２０は、所定の記録媒体（例えば、半導体メモリや磁気ディスク、光ディスクなど）によって実現されうる。固有ベクトル記憶部２０での記憶は、ＲＡＭ等における一時的な記憶でもよく、あるいは、長期的な記憶でもよい。

【００３５】

なお、行列記憶部１１、対角行列記憶部１３、固有値分解記憶部１６、固有値記憶部１８、固有ベクトル記憶部２０の任意の２以上の記憶部は、同一の記録媒体によって実現されてもよく、あるいは、別々の記録媒体によって実現されてもよい。前者の場合には、例えば、対称行列Ａの記憶されている領域が行列記憶部１１となり、対称３重対角行列Ｔ等の記憶されている領域が対角行列記憶部１３となる。

【００３６】

また、行列記憶部１１、対角行列記憶部１３、固有値分解記憶部１６、固有値記憶部１８、固有ベクトル記憶部２０の各記憶部は、２以上の記録媒体から構成されてもよい。

【００３７】

次に、本実施の形態による固有値分解装置１の動作について、図２のフローチャートを用いて説明する。

（ステップＳ１０１）対角化部１２は、行列記憶部１１で記憶されている対称行列Ａを読み出し、その行列Ａを３重対角化して対称３重対角行列Ｔを算出して対角行列記憶部１３に蓄積する。

【００３８】

（ステップＳ１０２）行列分割部１４、固有値分解部１５、固有値算出部１７によって対称３重対角行列Ｔの固有値が算出され、固有値記憶部１８に蓄積される。この処理の詳細については後述する。

【００３９】

（ステップＳ１０３）固有ベクトル算出部１９は、対角行列記憶部１３から対称３重対角行列Ｔを読み出し、固有値記憶部１８から対称３重対角行列Ｔの固有値を読み出し、対称３重対角行列Ｔの固有ベクトルを算出して固有ベクトル記憶部２０に蓄積する。この処理の詳細については後述する。

【００４０】

このようにして、対称３重対角行列Ｔの固有値分解が終了する。ここで、対称行列Ａの固有値は対称３重対角行列Ｔの固有値に等しいため、対称行列Ａの固有値も算出されたことになる。また、後述するように、所定の変換を行うことによって、対称行列Ａの固有ベクトルも対称３重対角行列Ｔの固有ベクトルから容易に算出することができる。

【００４１】

次に、図２のフローチャートのステップＳ１０１，Ｓ１０２の処理、すなわち、対称３重対角行列Ｔの固有値を算出するまでの処理について、より詳細に説明する。まず、図２のフローチャートのステップＳ１０２の処理について、図３のフローチャートを用いて説明する。

【００４２】

（ステップＳ２０１）行列分割部１４は、対角行列記憶部１３から対称３重対角行列Ｔを読み出し、その対称３重対角行列Ｔを２個の対称３重対角行列に分割して対角行列記憶部１３に蓄積する。行列分割部１４は、その対称３重対角行列を２個の対称３重対角行列に分割する処理を、分割後の各対称３重対角行列があらかじめ決められた大きさ以下となるまで繰り返す。この処理の詳細については後述する。

【００４３】

（ステップＳ２０２）固有値分解部１５は、対角行列記憶部１３で記憶されているあらかじめ決められた大きさ以下の各対称３重対角行列について、固有値分解を行う。固有値分解部１５は、固有値分解の結果である固有値と、固有ベクトルからなる直交行列の一部の要素である行列要素とを固有値分解記憶部１６に蓄積する。

【００４４】

（ステップＳ２０３）固有値算出部１７は、対称３重対角行列の固有値と、行列要素と

を固有値分解記憶部１６から読み出し、分割元の対称３重対角行列の固有値と、分割元の対称３重対角行列の行列要素とを算出して固有値分解記憶部１６に蓄積する。固有値算出部１７は、その分割元の対称３重対角行列の固有値と、行列要素とを算出する処理を、対称３重対角行列Ｔの固有値を算出するまで繰り返し、対称３重対角行列Ｔの固有値を固有値記憶部１８に蓄積する。このようにして、固有値を算出する処理が終了する。

【００４５】

［対称行列Ａの対角化］

対称行列Ａは、例えば、ハウスホルダー変換等を用いて、以下に示されるように３重対角化することができることが知られている。ここで、対称３重対角行列Ｔは、行の数と列の数とが一致する正方行列である。

【００４６】

【数１】

$$P^{\mathrm{T}}AP=T,\qquad T=\begin{pmatrix} t_1 & t_2 & & 0 \\ t_2 & t_3 & \ddots & \\ & \ddots & \ddots & t_{2n-2} \\ 0 & & t_{2n-2} & t_{2n-1} \end{pmatrix}$$

ただし、A は$n\times n$ 実対称行列、P は直交行列である。

【００４７】

したがって、対角化部１２は、上記のようにして、行列記憶部１１から対称行列Ａを読み出し、対称３重対角行列Ｔを算出することができる。その算出された対称３重対角行列Ｔは、対角行列記憶部１３で記憶される（ステップＳ１０１）。

【００４８】

［対称３重対角行列Ｔの分割］

対称３重対角行列Ｔを分割する処理について説明する。次式で示されるように、ｎ×ｎの対称３重対角行列Ｔが与えられた場合に、それらを２個の対称３重対角行列Ｔ$_1$，Ｔ$_2$と、その他の要素（下記のθとβ）とに分けることができる。

【００４９】

【数２】

$$T=\begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix}+\theta\beta\begin{pmatrix} \mathbf{e}_l \\ \theta^{-1}\mathbf{e}_f \end{pmatrix}\begin{pmatrix} \mathbf{e}_l^{\mathrm{T}} & \theta^{-1}\mathbf{e}_f^{\mathrm{T}} \end{pmatrix}$$

ただし、T は$n\times n$（nは２以上の整数）の実対称３重対角行列、T_1 は$n_1\times n_1$の実対称３重対角行列、T_2は$(n-n_1)\times(n-n_1)$の実対称３重対角行列、$\mathbf{e}_l\equiv(0,\cdots,0,1)^{\mathrm{T}}\in\mathbf{R}^{n_1}$、$\mathbf{e}_f\equiv(1,0,\cdots,0)^{\mathrm{T}}\in\mathbf{R}^{n-n_1}$、β は$T$ の(n_1+1,n_1)の要素、θ は非ゼロ定数である。

ここで、ｎ$_1$は、１<ｎ$_1$<ｎとなる整数である。例えば、８×８の対称３重対角行列Ｔを半分の大きさに分割する具体的な処理は、次のようになる。

【００５０】

【数３】

$$
T=\left(\begin{array}{ccccc|ccccc}
t_1 & t_2 & & & & & & & & \\
t_2 & t_3 & t_4 & & & & & & & \\
 & t_4 & t_5 & t_6 & & & & & & \\
 & & t_6 & t_7 & t_8 & & & & & \\
\hline
 & & & t_8 & t_9 & t_{10} & & & & \\
 & & & & t_{10} & t_{11} & t_{12} & & & \\
 & & & & & t_{12} & t_{13} & t_{14} & & \\
 & & & & & & t_{14} & t_{15} & &
\end{array}\right)
$$

$$
=\left(\begin{array}{ccccc|ccccc}
t_1 & t_2 & & & & & & & & \\
t_2 & t_3 & t_4 & & & & & & & \\
 & t_4 & t_5 & & t_6 & & & & & \\
 & & t_6 & t_7-\theta & t_8 & & & & & \\
\hline
 & & & & & t_9-t_8/\theta & t_{10} & & & \\
 & & & & & t_{10} & t_{11} & t_{12} & & \\
 & & & & & & t_{12} & t_{13} & t_{14} & \\
 & & & & & & & t_{14} & t_{15} &
\end{array}\right)
$$

$$
+\left(\begin{array}{ccccc|ccccc}
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & & & \theta\ t_8 & t_8 & & & \\
\hline
 & & & & & t_8 & t_8/\theta & & & \\
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & & & & & & &
\end{array}\right)
$$

$$
=\begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix}+\theta\beta\begin{pmatrix} \mathbf{e}_l \\ \theta^{-1}\mathbf{e}_f \end{pmatrix}\left(\mathbf{e}_l^{\mathrm{T}} \quad \theta^{-1}\mathbf{e}_f^{\mathrm{T}}\right)
$$

【００５１】

ただし、β＝ｔ$_8$であり、θは計算誤差が少なくなるように任意に設定することができる。また、上記各行列において、明記されている以外の行列の要素は０である。なお、上記各行列の明記されている各要素は０以外であってもよく、０であってもよい。ここで、並列処理を適切に実行するためには、ｎ$_1$を、ｎ／２を超えない最大の整数にとるか、あるいは、ｎ／２を下まわらない最小の整数にとることが好適である（このｎ$_1$の値を用いて行列を２個の行列に分割する場合を、「行列を略半分の２個の行列に分割する」と呼ぶことにする）。

【００５２】

【数４】

$$
n_1=\lfloor n/2 \rfloor
$$

本実施の形態では、ｎ$_1$を上式のようにとるものとする。なお、ｎ$_1$の値は、前述のように、１<ｎ$_1$<ｎの範囲内で任意であることは言うまでもない。

【００５３】

したがって、行列分割部１４は、上述のようにして、対角行列記憶部１３が記憶している対称３重対角行列Ｔを読み出し、その対称３重対角行列Ｔを２個の対称３重対角行列Ｔ

$_1$，Ｔ$_2$と、２個の値β、θに分割することができ、その分割の処理を繰り返すことができる。行列分割部１４は、分割後の２個の対称３重対角行列Ｔ$_1$，Ｔ$_2$と、２個の値β、θとを、対角行列記憶部１３に蓄積する。なお、２個の値β、θは、どの分割で発生した値であるのかがわかるように対角行列記憶部１３で記憶されることが好適である。図４は、図３のフローチャートのステップＳ２０１における行列分割部１４による行列を分割する処理を示すフローチャートである。

【００５４】

（ステップＳ３０１）行列分割部１４は、カウンタＩを「１」に設定する。

（ステップＳ３０２）行列分割部１４は、対角行列記憶部１３からＩ番目の分割をしていない対称３重対角行列を読み出し、その対称３重対角行列を２個の対称３重対角行列と、その他の要素とに分割する。そして、行列分割部１４は、分割した２個の対称３重対角行列と、その他の要素とを対角行列記憶部１３に蓄積する。

【００５５】

（ステップＳ３０３）行列分割部１４は、Ｉ番目の分割を行っていない対称３重対角行列が、対角行列記憶部１３で記憶されているかどうか判断する。そして、Ｉ番目の分割を行っていない対称３重対角行列が、対角行列記憶部１３で記憶されている場合には、ステップＳ３０２に戻り、そうでない場合には、ステップＳ３０４に進む。

【００５６】

（ステップＳ３０４）行列分割部１４は、Ｉ番目の分割を行った対称３重対角行列の大きさがあらかじめ決められている大きさ以下かどうか判断する。行列分割部１４は、例えば、目的とする行列の大きさ（例えば、２５×２５など）の記憶されている図示しない記録媒体から、その目的とする行列の大きさを読み出し、対角行列記憶部１３で記憶されている、Ｉ番目の分割後の対称３重対角行列がその大きさ以下であるかどうかを判断してもよい。そして、Ｉ番目の分割を行った対称３重対角行列の大きさがあらかじめ決められている大きさ以下である場合には、対称３重対角行列を分割する処理は終了となり、そうでない場合には、ステップＳ３０５に進む。

（ステップＳ３０５）行列分割部１４は、カウンタＩを１だけインクリメントする。そして、ステップＳ３０２に戻る。

【００５７】

なお、この図４のフローチャートでは、上述のように、行列分割部１４が各行列を、略半分の２個の行列に分割するため、Ｉ番目の分割後の各対称３重対角行列の大きさがほとんど同じである場合について説明したが、行列分割部１４が各行列を、略半分の２個の行列に分割しない場合には、行列分割部１４は、分割後の各行列があらかじめ決められた大きさ以下となるように、分割を繰り返すものとする。

【００５８】

また、図４のフローチャートでは、ステップＳ３０４において、行列分割部１４が、対角行列記憶部１３で記憶されている分割後の行列の大きさをあらかじめ決められた大きさと比較する処理を実行する場合について説明したが、これは一例であって、行列分割部１４は、ステップＳ３０４において、それ以外の処理を行ってもよい。例えば、上述のように、行列分割部１４が各行列を、略半分の２個の行列に分割する場合には、元の対称３重対角行列Ｔの大きさを知ることができれば、何番目の分割で目的とする行列の大きさとなるのかを知ることができる。したがって、Ｎ番目（Ｎは１以上の整数）の分割で目的とする行列の大きさとなる場合には、ステップＳ３０４において、ＩがＮであるかどうかを比較し、ＮでなければステップＳ３０５に進み、Ｎであれば一連の処理を終了するようにしてもよい。

【００５９】

図５は、行列の分割について説明するための図である。まず、行列分割部１４は、１番目の分割として、対称３重対角行列Ｔを、対称３重対角行列Ｔ$_1$と、対称３重対角行列Ｔ$_2$とに分割する（ステップＳ３０１，Ｓ３０２）。対称３重対角行列Ｔは、１個しかないため、行列分割部１４は、１番目の分割をしていない行列がないと判断する（ステップＳ

３０３）。また、対称３重対角行列T_1等はあらかじめ決められた大きさ以下の行列でないとすると（ステップＳ３０４）、行列分割部１４は、２番目の分割として、対称３重対角行列T_1を、対称３重対角行列T_{11}と、対称３重対角行列T_{12}とに分割する（ステップＳ３０５，Ｓ３０２）。この場合には、２番目の分割を行っていない対称３重対角行列T_2が存在するため、行列分割部１４は、対称３重対角行列T_2も、対称３重対角行列T_{21}と、対称３重対角行列T_{22}とに分割する（ステップＳ３０３，Ｓ３０２）。このようにして、分割後の各対称３重対角行列が目的とする大きさ以下になるまで、行列を分割する処理が繰り返される。なお、図５では、対称３重対角行列以外の要素（上述のβ、θ）については省略している。また、図５において、各対称３重対角行列は、正方行列である。

【００６０】

［分割された行列の固有値分解］

行列T_iが対称行列であるとすると、行列T_iの固有値分解は、次のようになる。

【数５】

$$T_i = Q_i D_i Q_i^{\mathrm{T}}$$

【００６１】

ここで、D_iは、行列T_iと同じ大きさの対角行列である。D_iの各対角成分はT_iの固有値である。また、Q_iは、行列T_iの固有ベクトルからなる直交行列である。

【００６２】

したがって、固有値分解部１５は、対角行列記憶部１３から分割後の各対称３重対角行列を読み出し、上記のようにして、固有値分解を行う（ステップＳ２０２）。固有値分解の方法として、例えば、２分法と逆反復法を組み合わせた方法、ＭＲ＾３法、ＱＲ法等を用いてもよいことは前述の通りである。図５で示されるように対称３重対角行列の分割が行われた場合には、固有値分解部１５は、各対称３重対角行列T_{111}，T_{112}，T_{121}，T_{122}，…，T_{222}に対して固有値分解を行うことになる。なお、固有値分解部１５は、固有値分解の結果得られた各固有値と、固有値分解の結果得られた直交行列の一部の要素である行列要素とを固有値分解記憶部１６に蓄積する。ここで、行列要素が、直交行列のどの要素を含むのかについては後述する。なお、固有値分解部１５が固有値と行列要素とを固有値分解記憶部１６に蓄積するのではなく、固有値と固有ベクトルからなる直交行列とを固有値分解記憶部１６に蓄積してもよい。その場合でも、固有ベクトルからなる直交行列を蓄積することによって、少なくとも、行列要素が蓄積されたことになる。

【００６３】

なお、後述するように、固有値を算出するときに用いるために必要なのは、直交行列Q_iの全体ではなく、その直交行列Q_iの第１行と最終行である。したがって、この固有値分解において、直交行列Q_iの全体を求めるのではなく、直交行列Q_iの第１行と最終行を求めるようにしてもよい。そのためには、例えば、ＬＡＰＡＣＫ３．０で提供されているＱＲ法の固有値分解ルーチンであるＤＳＴＥＱＲにおいて、直交行列Q_iの算出における初期値として、単位行列Ｉを用いる代わりに、次式で示される行列Ｙを用いてもよい。

【数６】

$$Y = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}$$

ただし、Q_iはｎ×ｎの直交行列であり、Yは２×ｎの行列である。

【００６４】

このようにすることで、ＤＳＴＥＱＲで直交行列Q_iの第１行と最終行を求めるための計算領域を削減することができる。直交行列Q_iそのものを算出する場合には、Ｏ（n^2）の作業領域が必要であるが、上記の行列Ｙを用いて直交行列Q_iの第１行と最終行を算出する場合には、Ｏ（ｎ）の作業領域でよいことになる。その結果、計算量も少なくなり

、固有値分解の処理速度も向上する。このように、固有値分解部１５は、分割後の対称３重対角行列に対して、直交行列Q_iそのものを算出する固有値分解を行ってもよく、あるいは、直交行列の一部の成分（ここでは、第１行と最終行）を算出する処理を行ってもよい。この後者の処理も固有値分解と呼ぶことにする。このように、固有値分解部１５は、あらかじめ決められた大きさ以下の各対称３重対角行列に対して固有値分解を行い、各対称３重対角行列の固有値と、各対称３重対角行列の固有ベクトルからなる直交行列の行列要素（ここでは、第１行と最終行）とを少なくとも算出し、その固有値と、その行列要素とを固有値分解記憶部１６に蓄積してもよい。

【００６５】

［固有値の算出］

まず、図６で示されるように、分割された２個の対称３重対角行列T_1，T_2から、分割元の対称３重対角行列T_0の固有値等を算出する処理について説明する。対称３重対角行列T_1，T_2は、次のように固有値分解されていたとする。ここで、対称３重対角行列T_1，T_2は、両者共に正方行列であるとする。

【数７】

$$T_1 = Q_1 D_1 Q_1^{\mathrm{T}}$$

$$T_2 = Q_2 D_2 Q_2^{\mathrm{T}}$$

【００６６】

この場合に、分割元の対称３重対角行列T_0は、次のようになる。

【数８】

$$T_0 = Q_{12}(D_{12} + \rho \mathbf{z}\mathbf{z}^{\mathrm{T}})Q_{12}^{\mathrm{T}}$$

$$Q_{12} \equiv \begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix},\ D_{12} \equiv \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix},$$

$$\mathbf{z} \equiv \frac{1}{\sqrt{1+\theta^{-2}}} \begin{pmatrix} \mathbf{l}_1^{\mathrm{T}} \\ \theta^{-1}\mathbf{f}_2^{\mathrm{T}} \end{pmatrix},\ \rho \equiv \beta(\theta + \theta^{-1})$$

ただし、$\mathbf{l}_1$はQ_1の最終行であり、$\mathbf{f}_2$はQ_2の第１行である。

【００６７】

上記の式において、β、θは、対称３重対角行列の分割の処理において説明した値である。ここで、行列（$D_{12}+\rho zz^{T}$）の固有値分解が得られれば、行列T_0の固有値分解が得られたことになる。なぜなら、行列（$D_{12}+\rho zz^{T}$）の固有値分解$D_{12}+\rho zz^{T}=QDQ^{T}$が求まると、次式のようになり、行列$T_0$の固有値分解が完了するからである。

【数９】

$$T_0 = Q_{12}(QDQ^{\mathrm{T}})Q_{12}^{\mathrm{T}} = (Q_{12}Q)D(Q_{12}Q)^{\mathrm{T}} = Q_0 D Q_0^{\mathrm{T}}$$

【００６８】

次に、行列（$D_{12}+\rho zz^{T}$）の固有値を算出する方法について説明する。行列（$D_{12}+\rho zz^{T}$）の固有値をλ_i、固有ベクトルをq_iとすると、

【数１０】

$$(D_{12} + \rho \mathbf{z}\mathbf{z}^{\mathrm{T}})\mathbf{q}_i = \lambda_i \mathbf{q}_i$$

となる。

【００６９】

（１）ｚの成分に０が含まれず、D_{12}の対角成分に同じ値が含まれない場合

まず、ｚの成分に０が含まれず、D_{12}の対角成分に同じ値が含まれない場合について説明する。その場合には、上記の式を、次に示すように順次、変形することができる。

【数１１】

$$(D_{12}-\lambda_i I)\mathbf{q}_i=-\rho(\mathbf{z}^{\mathrm{T}}\mathbf{q}_i)\mathbf{z}$$

$$\mathbf{z}^{\mathrm{T}}\mathbf{q}_i=-\rho(\mathbf{z}^{\mathrm{T}}\mathbf{q}_i)\mathbf{z}^{\mathrm{T}}(D_{12}-\lambda_i I)^{-1}\mathbf{z}$$

$$1+\rho\ \mathbf{z}^{\mathrm{T}}(D_{12}-\lambda_i I)^{-1}\mathbf{z}=0$$

【００７０】

ここで、

【数１２】

$$D_{12}=\mathrm{diag}(d_1,d_2,\cdots,d_n)$$

$$\mathbf{z}=(z_1,z_2,\cdots,z_n)^{\mathrm{T}}$$

とすると、次の固有方程式を得ることができる。固有値λ$_i$（ｉ＝１，２，…，ｎ）は、その固有方程式ｎ個の解である。下記のｆ（λ）は、単調増加であり、かつｆ（λ）＝０の解の範囲が既知であるため、ニュートン法を用いて計算することができる。

【数１３】

$$f(\lambda)=1+\rho\sum_{j=1}^{n}\frac{z_j^2}{d_j-\lambda}=0 \qquad (式１)$$

【００７１】

なお、Ｄ＝ｄｉａｇ（λ$_1$，λ$_2$，…，λ$_n$）となる。行列Ｑは、行列（Ｄ$_{12}$＋ρｚｚ^{T}）の固有ベクトルからなる直交行列であるため、Ｑ＝（ｑ$_1$，ｑ$_2$，…，ｑ$_n$）となる。したがって、固有ベクトルｑ$_i$を求めると、行列Ｑを求めることができる。なお、固有ベクトルｑ$_i$は、次式で与えられる。

【数１４】

$$\mathbf{q}_i=\frac{\left(\frac{z_1}{d_1-\lambda_i},\frac{z_2}{d_2-\lambda_i},\cdots,\frac{z_n}{d_n-\lambda_i}\right)^{\mathrm{T}}}{\sqrt{\sum_{k=1}^{n}\frac{z_k^2}{(d_k-\lambda_i)^2}}} \qquad (式２)$$

【００７２】

（２）ｚの成分に０が含まれる場合

ｚの成分に０が含まれる場合には、行列の次数を下げることができ、その次数を下げた行列を用いて固有値、及び固有ベクトルを求めることができる。この行列の次数を下げる操作をデフレーションと呼ぶ。

【００７３】

ｚ$_j$＝０の場合には、次式で示すように、（ｄ$_j$，ｅ$_j$）が一組の固有値と固有ベクトルとなる。ただし、ｅ$_j$は、ｊ番目の成分が１であり、その他の成分が０である単位ベクトルである。

【数１５】

$$((D_{12}+\rho\mathbf{z}\mathbf{z}^{\mathrm{T}})-d_j I)\mathbf{e}_j=(D_{12}-d_j I)\mathbf{e}_j+\rho\mathbf{z}(\mathbf{z}^{\mathrm{T}}\mathbf{e}_j)$$

$$=0$$

【００７４】

また、行列（Ｄ$_{12}$＋ρｚｚ^{T}）の固有方程式は、次式のようになる。

【数１６】

$$\left|D_{12}+\rho\mathbf{z}\mathbf{z}^{\mathrm{T}}-\lambda_i I_n\right|=(d_j-\lambda_i)\left|\hat{D}_{12}+\rho\hat{\mathbf{z}}\hat{\mathbf{z}}^{\mathrm{T}}-\lambda_i I_{n-1}\right|$$

ただし、$\hat{D}_{12}+\rho\hat{\mathbf{z}}\hat{\mathbf{z}}^{\mathrm{T}}$は、$D_{12}+\rho\mathbf{z}\mathbf{z}^{\mathrm{T}}$の第ｊ行、第ｊ列を除いた小行列である。

また、I_nは、n次の単位行列である。

【００７５】

したがって、ｚ$_{j}$＝０の場合の固有値は、ｄ$_{j}$と、行列（Ｄ$_{12}$＋ρｚｚ^{T}）の第ｊ行、第ｊ列を除いた小行列について、上記式１の固有方程式を解いた値である。すなわち、固有値は、ｄ$_{j}$と、次の固有方程式を解いた値となる。なお、次の固有方程式を解いた固有値を、固有値λ$_{i}$（ｉ＝１，２，…，ｊ－１，ｊ＋１，…，ｎ）とする。

【数１７】

$$f(\lambda)=1+\rho\sum_{\substack{k=1\\k\neq j}}^{n}\frac{z_k^2}{d_k-\lambda}=0$$

【００７６】

固有ベクトルは、行列（Ｄ$_{12}$＋ρｚｚ^{T}）の第ｊ行、第ｊ列を除いた小行列について上記式２で算出したｎ－１次元の固有ベクトルの第（ｊ－１）行と、第ｊ行との間に０を挿入した次式のｎ次元のベクトルと、上述のｅ$_{j}$となる。

【数１８】

$$\mathbf{q}_i=\frac{\left(\frac{z_1}{d_1-\lambda_i},\frac{z_2}{d_2-\lambda_i},\cdots,\frac{z_{j-1}}{d_{j-1}-\lambda_i},0,\frac{z_j}{d_j-\lambda_i},\cdots,\frac{z_n}{d_n-\lambda_i}\right)^{\mathrm{T}}}{\sqrt{\sum_{\substack{k=1\\k\neq j}}^{n}\frac{z_k^2}{(d_k-\lambda_i)^2}}}\quad(i\neq j)$$

$$\mathbf{q}_j=\mathbf{e}_j$$

【００７７】

このように、ｚの成分に０が含まれる場合には、行列（Ｄ$_{12}$＋ρｚｚ^{T}）よりも次元の小さい行列について固有値分解を行うことになるため、計算を高速化することが可能であり、結果として、計算時間が短縮されうる。

【００７８】

（３）Ｄ$_{12}$の対角成分に同じ値が含まれる場合

ｄ$_{j+1}$＝ｄ$_{j+2}$＝…＝ｄ$_{k}$の場合にも、上記説明と同様のデフレーションを行うことによって、固有値と固有ベクトルとを算出することができる。

【００７９】

【数１９】

$$\hat{\mathbf{z}}=(z_{j+1},z_{j+2},\cdots,z_k)^{\mathrm{T}}$$

に対して、次の２個の直交行列を考える。

【数２０】

$$\hat{Q}\ \hat{\mathbf{z}}=(*,0,0,\cdots,0)^{\mathrm{T}}$$

$$\tilde{Q}=\begin{pmatrix}I_j & & \\ & \hat{Q} & \\ & & I_{n-k}\end{pmatrix}$$

ただし、$\hat{Q}\in\mathbf{R}^{(k-j)\times(k-j)}$である。

また、*は所定の実数である。

【００８０】
すると、$d_{j+1}=d_{j+2}=\cdots=d_k$より、
【数２１】

$$\tilde{Q}D_{12}\tilde{Q}^{T}=D_{12}$$

$$\tilde{Q}\,\mathbf{z}=(z_1,z_2,\cdots,z_j,*,0,\cdots,0,z_{k+1},\cdots,z_n)^{T}=\tilde{\mathbf{z}}$$

となる。したがって、次式が得られる。
【数２２】

$$D_{12}+\rho\mathbf{z}\mathbf{z}^{T}=\tilde{Q}^{T}(D_{12}+\rho\tilde{\mathbf{z}}\tilde{\mathbf{z}}^{T})\tilde{Q}$$

【００８１】
ここで、
【数２３】

$$D_{12}+\rho\tilde{\mathbf{z}}\tilde{\mathbf{z}}^{T}$$

の固有値と固有ベクトルとは、上記のｚの成分に０が含まれるデフレーションの場合と同様にして求めることができる。また、
【数２４】

$$\hat{Q}$$

は、符号の任意性を除いて一意に定まる。ここで、その算出方法について説明する。まず、次式を満たす直交行列について考える。
【００８２】
【数２５】

$$\hat{Q}_{j+1,i}\cdot(z_{j+1},\cdots,z_i,\cdots,z_k)^{T}=(*,z_{j+2},\cdots,z_{i-1},0,z_{i+1},\cdots,z_k)^{T}$$

この直交行列を次式のようにとると、上記式を満たすことになる。
【数２６】

$$\hat{Q}_{j+1,i}=\begin{pmatrix}\dfrac{z_{j+1}}{\sqrt{z_{j+1}^2+z_i^2}} & & \dfrac{z_i}{\sqrt{z_{j+1}^2+z_i^2}} \\ & I_{i-j-2} & \\ \dfrac{-z_i}{\sqrt{z_{j+1}^2+z_i^2}} & & \dfrac{z_{j+1}}{\sqrt{z_{j+1}^2+z_i^2}} \\ & & & I_{k-i}\end{pmatrix}$$

【００８３】
ここで、上記の式の＊の値は、次式から求まる。
【数２７】

$$\hat{Q}_{j+1,i}\cdot(z_{j+1},\cdots,z_i,\cdots,z_k)^{T}=(\sqrt{z_{j+1}^2+z_i^2},z_{j+2},\cdots,z_{i-1},0,z_{i+1},\cdots,z_k)^{T}$$

【００８４】
したがって、
【数２８】

$$\hat{Q}=\hat{Q}_{j+1,j+2}\cdot\hat{Q}_{j+1,j+3}\cdots\hat{Q}_{j+1,k}$$

となる。その結果、

【数２９】

$\widetilde{Q}$

も、符号の任意性を除いて求めることができる。このようにして、ｄ$_{ｊ＋１}$＝ｄ$_{ｊ＋２}$＝…＝ｄ$_{ｋ}$の場合にも行列（Ｄ$_{１２}$＋ρｚｚ^{Ｔ}）の固有値と固有ベクトルとを求めることができる。すなわち、固有値は、ｄ$_{ｊ＋２}$，ｄ$_{ｊ＋３}$，…，ｄ$_{ｋ}$と、次の固有方程式を解いた値となる。なお、次の固有方程式を解いた固有値を、固有値λ$_{ｉ}$（ｉ＝１，２，…，ｊ＋１，ｋ＋１，…，ｎ）とする。

【数３０】

$$f(\lambda)=1+\rho\sum_{\substack{m=1\\ m\neq j+2,\cdots,k}}^{n}\frac{z_m^2}{d_m-\lambda}=0$$

【００８５】

固有ベクトルは、次のようになる。

【数３１】

$$\mathbf{q}_i=\begin{cases}\widetilde{Q}^{\mathrm{T}}\dfrac{\left(\dfrac{\widetilde{z}_1}{d_1-\lambda_i},\dfrac{\widetilde{z}_2}{d_2-\lambda_i},\cdots,\dfrac{\widetilde{z}_n}{d_n-\lambda_i}\right)^{\mathrm{T}}}{\sqrt{\displaystyle\sum_{k=1}^{n}\frac{\widetilde{z}_k^2}{(d_k-\lambda_i)^2}}} & (1\le i\le j+1,k+1\le i\le n)\\ \widetilde{Q}^{\mathrm{T}}\mathbf{e}_i & (j+2\le i\le k)\end{cases}$$

【００８６】

なお、上記（２）ｚの成分に０が含まれる場合、（３）Ｄ$_{１２}$の対角成分に同じ値が含まれる場合において、ｚの成分が０であるかどうかの判断や、Ｄ$_{１２}$の対角成分に同じ値が含まれているかどうかの判断は、微少な誤差を考慮して行ってもよい。例えば、次式が満たされる場合には、ｚの成分が０であると判断してもよく、また、Ｄ$_{１２}$の対角成分が同じ値であると判断してもよい。

【数３２】

$$|z_i|<TOL_1$$

$$|d_{j+1}-d_k|<TOL_2$$

ただし、$TOL_1\equiv\max(z_1,z_2,\cdots,z_n)\times\varepsilon$、$TOL_2\equiv\max(d_1,d_2,\cdots,d_n)\times\varepsilon$ である。

【００８７】

なお、εは、マシン・イプシロンに定数（例えば、２，４，８などの実数であり、計算誤差を少なくするための値を選択することが好適である）を掛けた値を用いてもよく、他の適切な値であってもよい。

【００８８】

このようにして、（１）ｚの成分に０が含まれず、Ｄ$_{１２}$の対角成分に同じ値が含まれない場合、（２）ｚの成分に０が含まれる場合、（３）Ｄ$_{１２}$の対角成分に同じ値が含まれる場合のそれぞれについて、固有値分解Ｄ$_{１２}$＋ρｚｚ^{Ｔ}＝ＱＤＱ^{Ｔ}が求まることになる。したがって、行列Ｑと行列Ｑ$_{１２}$から行列Ｑ$_{０}$を求めることができ、行列Ｄも求まっているため、分割元の対称３重対角行列Ｔ$_{０}$の固有値分解を行うことができる。このように、対象となる行列を２個の副行列に分割し、その分割した後の各副行列について固有値分解を行い、その後に各副行列の固有値分解を用いて分割元の行列の固有値分解を行う方法は、分割統治法（Ｄｉｖｉｄｅ　ａｎｄ　Ｃｏｎｑｕｅｒ：D&C）と呼ばれている。

【００８９】

分割統治法では、固有ベクトルまで算出することになり、その固有ベクトルを算出する処理において行列の計算をしなければならないため、非常に負荷の大きい処理となる。分

割統治法において固有値分解をする場合には、例えば、計算時間の９５％程度が固有ベクトルを算出するベクトル更新の処理（例えば、上述のように、行列Ｑと行列Q_{12}から行列Q_0を算出するために行列を掛け合わせる処理）に費やされることもありうる。

【００９０】

一方、分割元の行列T_0の固有値のみを算出する場合には、上記処理を簡略化することができる。次に、その方法について説明する。上記の結果より、次のようになる。

【００９１】

【数３３】

$$\mathbf{f} = \mathbf{f}_{12} Q \qquad \text{（式３）}$$

$$\mathbf{l} = \mathbf{l}_{12} Q \qquad \text{（式４）}$$

ただし、$\mathbf{f}$はQ_0の第１行であり、$\mathbf{l}$はQ_0の最終行である。

また、$\mathbf{f}_{12}$はQ_{12}の第１行であり、$\mathbf{f}_{12} = (\mathbf{f}_1, 0, \cdots, 0)$となる。

また、$\mathbf{l}_{12}$はQ_{12}の最終行であり、$\mathbf{l}_{12} = (0, \cdots, 0, \mathbf{l}_2)$となる。

また、$\mathbf{f}_1$はQ_1の第１行であり、$\mathbf{l}_2$はQ_2の最終行である。

【００９２】

ここで、上記のように、ｆは行列T_0を固有値分解した結果の直交行列の最初の行であり、ｌは行列T_0を固有値分解した結果の直交行列の最後の行である。このｆとｌ、すなわち、直交行列の最初の行と、最後の行とが行列要素となる。なお、直交行列Q_{12}について、後述する列の入れ替えを行った場合には、f_{12}、l_{12}はそれぞれ、その列を入れ替えた後の直交行列Q_{12}の第１行、最終行となる。

【００９３】

上記の結果から、図６で示されるように、行列T_1，T_2から分割元の行列T_0を構成する場合に、行列T_0について、

【数３４】

$$\lambda_i, \mathbf{f}, \mathbf{l}$$

をそれぞれ算出することができる。このような処理を繰り返すことにより、最終的に、対称行列Ａを３重対角化した対称３重対角行列Ｔの固有値を算出することができる。

【００９４】

なお、上記の（３）D_{12}の対角成分に同じ値が含まれる場合には、式３，式４が次のようになる。

【数３５】

$$\mathbf{f} = \mathbf{f}_{12} Q = \mathbf{f}_{12} \tilde{Q}^{\mathrm{T}} \overline{Q}$$

$$\mathbf{l} = \mathbf{l}_{12} Q = \mathbf{l}_{12} \tilde{Q}^{\mathrm{T}} \overline{Q}$$

ただし、$\overline{Q} = \tilde{Q} Q$ である。

【００９５】

ここで、上記式を計算する場合に、

【数３６】

$$\mathbf{f}_{12} \tilde{Q}^{\mathrm{T}}, \ \mathbf{l}_{12} \tilde{Q}^{\mathrm{T}}$$

を先に計算して、その後に

【数３７】

$$\overline{Q}$$

を掛けた方が、計算量が少なくなるため、その順番で計算するようにしてもよい。

【００９６】

したがって、固有値算出部１７は、上述のようにして、固有値分解記憶部１６から固有値と行列要素とを読み出し、対角行列記憶部１３から行列の分割時に発生したその他の要素に関するβ、θを読み出し、それらを用いることによって、分割元の対称３重対角行列の固有値と行列要素とを算出する。そして、それらを算出する処理を繰り返すことによって、最終的に対称３重対角行列Ｔの固有値を算出する。図７は、図３のフローチャートのステップＳ２０３における固有値算出部１７が固有値を算出する処理を示すフローチャートである。

【００９７】

（ステップＳ４０１）固有値算出部１７は、カウンタＪを「１」に設定する。

（ステップＳ４０２）固有値算出部１７は、Ｊ番目の固有値の算出は最後の固有値の算出であるかどうか判断する。ここで、最後の固有値の算出とは、対称３重対角行列Ｔの固有値を算出することである。そして、最後の固有値の算出である場合には、ステップＳ４０６に進み、そうでない場合には、ステップＳ４０３に進む。

【００９８】

（ステップＳ４０３）固有値算出部１７は、分割元の対称３重対角行列の固有値等を算出する。この処理の詳細については後述する。

（ステップＳ４０４）固有値算出部１７は、Ｊ番目の固有値の算出において、分割元の全ての対称３重対角行列の固有値等を算出したかどうか判断する。そして、Ｊ番目の固有値の算出において、分割元の全ての対称３重対角行列の固有値等を算出した場合には、ステップＳ４０５に進み、そうでない場合には、ステップＳ４０３に戻る。

【００９９】

（ステップＳ４０５）固有値算出部１７は、カウンタＪを１だけインクリメントする。そして、ステップＳ４０２に戻る。

（ステップＳ４０６）固有値算出部１７は、対称３重対角行列Ｔの固有値を算出し、その算出した固有値を固有値記憶部１８に蓄積する。このようにして、対称３重対角行列Ｔの固有値を算出する一連の処理は終了となる。

【０１００】

図８は、図７のフローチャートにおけるステップＳ４０３の詳細な処理を示すフローチャートである。なお、この図８のフローチャートの処理を実行する前に、対角行列$D_{1\ 2}$の対角成分が昇順、あるいは降順となるように並べ替えておくことが好適である。なお、その場合には、直交行列$Q_{1\ 2}$の列、及びｚの成分も、対角行列$D_{1\ 2}$の並び替えと同様に並び替えるものとする。具体的には、対角行列$D_{1\ 2}$の第ｉ成分と、第ｊ成分とを入れ替えた場合には、直交行列$Q_{1\ 2}$の第ｉ列と、第ｊ列とを入れ替え、ｚの第ｉ行と、第ｊ行とを入れ替えればよい。

【０１０１】

（ステップＳ５０１）固有値算出部１７は、分割元の行列の固有値を、式１等を用いて算出する。そして、固有値算出部１７は、算出した固有値を固有値分解記憶部１６に蓄積する。

【０１０２】

（ステップＳ５０２）固有値算出部１７は、ステップＳ５０１で算出した固有値を用いて、式２等からq_iを算出する。このq_iを算出することにより、q_iを列ベクトルに有する行列Ｑを算出したことになる。固有値算出部１７は、算出した行列Ｑを図示しない記録媒体において一次記憶してもよい。

【０１０３】

（ステップＳ５０３）固有値算出部１７は、ステップＳ５０２で算出した行列Ｑを用いて、分割元の行列に関する行列要素ｆ、ｌを算出する。この行列要素は、式３，式４で示されるものである。そして、固有値算出部１７は、算出した分割元の行列要素を固有値分解記憶部１６に蓄積する。このようにしてステップＳ４０３の処理は終了となる。

【０１０４】

なお、図８のフローチャートにおいて、ｚの成分に０が含まれず、$D_{1\ 2}$の対角成分に

同じ値が含まれない場合には、上記（１）で説明したようにして、固有値、固有ベクトルからなる行列Ｑを算出することができる。また、ｚの成分に０が含まれ、D_{12}の対角成分に同じ値が含まれない場合には、上記（２）で説明したようにして、固有値、固有ベクトルからなる行列Ｑを算出することができる。また、D_{12}の対角成分に同じ値が含まれる場合には、ｚの成分に０が含まれるかどうかに関わらず、上記（３）で説明したようにして、固有値、固有ベクトルからなる行列Ｑを算出することができる。上記（３）で説明したようにして固有値等を算出する場合には、前述のように、行列Ｑを算出するのではなく、行列Ｑを構成する行列を直接f_{12}等に作用させることによって、ｆ、ｌを算出するようにしてもよい。

【０１０５】

また、図８のフローチャートでは、行列Ｑから、行列要素ｆ、ｌを一度に算出する場合について示したが、行列要素ｆ、ｌをその成分ごとに計算してもよい。上記の式３，式４から明らかなように、f_{12}、l_{12}に固有ベクトルq_iを掛けたものが、行列要素ｆ、ｌのｉ番目の成分となる。このような方法で行列要素ｆ、ｌを計算する場合には、固有ベクトル固有ベクトルq_iのみをｉの値を順次、変えながら記憶することによってその計算を行うことができる。したがって、行列要素ｆ、ｌの計算において行列Ｑ全体を記憶する必要がなくなり、その計算で用いるメモリ上の作業領域を削減することが可能である。

【０１０６】

図９は、固有値を算出する処理について説明するための図である。固有値分解記憶部１６では、固有値分解部１５によって行列T_{111}，T_{112}，…，T_{222}が固有値分解された結果、すなわち、それらの固有値と、行列要素とが記憶されているものとする。まず、固有値算出部１７は、固有値等の１番目の算出を開始する（ステップＳ４０１，Ｓ４０２）。ここで、固有値等の１番目の算出とは、図９の一番下の行の行列の固有値と、行列要素とから、下から２番目の行の行列の固有値と、行列要素とを算出することである。固有値算出部１７は、行列T_{111}と、行列T_{112}との各固有値、及び行列T_{111}と、行列T_{112}との各行列要素を固有値分解記憶部１６から読み出す。また、行列T_{11}を行列T_{111}と、行列T_{112}とに分解したときに発生した２個の値β、θを対角行列記憶部１３から読み出す。そして、固有値算出部１７は、それらの値を用いて、行列T_1の固有値を算出して固有値分解記憶部１６に蓄積する（ステップＳ５０１）。次に、固有値算出部１７は、行列T_{11}の固有値を用いて直交行列Ｑを算出し（ステップＳ５０２）、その直交行列Ｑを用いて行列T_{11}の行列要素を算出して固有値分解記憶部１６に蓄積する（ステップＳ５０３）。

【０１０７】

固有値等の１番目の算出はまだ終わっていないため（ステップＳ４０４）、固有値算出部１７は、上記説明と同様にして、行列T_{121}、行列T_{122}の固有値、行列要素等に基づいて、行列T_{12}の固有値、行列要素を算出する（ステップＳ４０２，Ｓ４０３）。このようにして、固有値等の１番目の算出が終了すると、固有値算出部１７は、固有値等の２番目の算出、すなわち、行列T_{11}、行列T_{12}の固有値、行列要素等に基づいて行列T_1の固有値、行列要素の算出を行う（ステップＳ４０４，Ｓ４０５，Ｓ４０２，Ｓ４０３）。その後、固有値算出部１７は、同様にして、行列T_{21}、行列T_{22}の固有値、行列要素等に基づいて行列T_2の固有値、行列要素の算出を行う（ステップＳ４０４，Ｓ４０３）。

【０１０８】

固有値算出部１７が固有値等の２番目の算出を終了すると（ステップＳ４０４）、次の固有値等の算出は、対称３重対角行列Ｔの固有値を求める最後の処理となるため（ステップＳ４０５，Ｓ４０２）、固有値算出部１７は、固有値の算出のみを行い、その算出した固有値を固有値記憶部１８に蓄積する（ステップＳ４０６）。このようにして、固有値の算出が終了する。

【０１０９】

なお、図９の行列T_1、行列T_2の固有値、行列要素等に基づいて行列Ｔの固有値を算

出する場合には、行列要素f_1、l_2、あるいは行列要素f_2、l_1を用いるだけでよい。したがって、行列T_1、行列T_2の行列要素を算出する場合に、行列要素f_1、l_2、あるいは行列要素f_2、l_1を計算しなくてもよい。このようにすることで、計算量を削減することができ、処理スピードを向上させることが可能である。

【０１１０】

次に、固有ベクトル算出部１９について説明する。本実施の形態では、固有ベクトル算出部１９が（１）ｑｄ型ツイスト分解法により固有ベクトルを算出する場合と、（２）ＬＶ型ツイスト分解法により固有ベクトルを算出する場合とについて説明する。

【０１１１】

（１）ｑｄ型ツイスト分解法により固有ベクトルを算出する場合

図１０は、ｑｄ型ツイスト分解法により固有ベクトルを算出する場合の固有ベクトル算出部１９の構成を示す図である。図１０において、固有ベクトル算出部１９は、正定値化部２１と、コレスキー分解部２２と、ベクトル算出部２３とを備える。

【０１１２】

正定値化部２１は、固有値記憶部１８から対称３重対角行列Ｔの固有値を読み出し、その固有値のいずれかが負の値であるかどうか判断する。そして、その固有値のいずれかが負の値である場合に、正定値化部２１は、対角行列記憶部１３から対称３重対角行列Ｔを読み出し、その対称３重対角行列Ｔの固有ベクトルを変化させることなく、対称３重対角行列Ｔのすべての固有値が正の値となるように、その対称３重対角行列Ｔを正定値化し、正定値化した後の固有値を固有値記憶部１８に蓄積し、正定値化した後の対称３重対角行列Ｔを対角行列記憶部１３に蓄積する。この正定値化の処理を行うのは、ｑｄ型ツイスト分解法により固有ベクトルを算出する場合には、対称３重対角行列Ｔの固有値がすべて正の値でなければならないからである。なお、対称３重対角行列Ｔを正定値化した行列も、対称３重対角行列Ｔと呼ぶことにする。また、この正定値化の処理が行われた場合には、コレスキー分解や固有ベクトルを算出する処理は、正定値化後の対称３重対角行列Ｔ、固有値に対して行われることになる。この処理の詳細については後述する。ここで、正定値化の行われた場合であっても、固有値記憶部１８には、正定値化の前の固有値が記憶されているものとする。その正定値化の前の固有値が、固有値分解装置１が最終的に求める固有値だからである。また、正定値化の行われた場合であって、ｑｄ型ツイスト分解法による固有ベクトルの算出のみを行う場合には、対角行列記憶部１３で記憶されている対称３重対角行列Ｔを、正定値化の後の対称３重対角行列Ｔで上書きしてもよい。一方、ＬＶ型ツイスト分解法による固有ベクトルの算出も行う場合には、対角行列記憶部１３には、正定値化の前の対称３重対角行列Ｔと、正定値化の後の対称３重対角行列Ｔとが記憶されることになる。

【０１１３】

コレスキー分解部２２は、対角行列記憶部１３から、すべての固有値が正の値である対称３重対角行列Ｔを読み出し、固有値記憶部１８から、対称３重対角行列Ｔの正の値の固有値を読み出す。そして、コレスキー分解部２２は、対称３重対角行列Ｔの各要素に関してｑｄ型変換を行うことによって、対称３重対角行列Ｔを上２重対角行列$B^{(+1)}$及び下２重対角行列$B^{(-1)}$にコレスキー分解する。この処理の詳細については後述する。なお、「対称３重対角行列Ｔを上２重対角行列$B^{(+1)}$及び下２重対角行列$B^{(-1)}$にコレスキー分解する」ことには、対称３重対角行列Ｔの固有値に単位ベクトルを掛けたものを対称３重対角行列Ｔから引いた行列を、上２重対角行列$B^{(+1)}$及び下２重対角行列$B^{(-1)}$にコレスキー分解することが含まれてもよい。

【０１１４】

ベクトル算出部２３は、コレスキー分解部２２が算出した上２重対角行列$B^{(+1)}$及び下２重対角行列$B^{(-1)}$の各要素と、対称３重対角行列Ｔの正の値の固有値とを用いて固有ベクトルを算出して固有ベクトル記憶部２０に蓄積する。この固有ベクトルを各列に有する行列が、固有値分解の直交行列となる。

【０１１５】

次に、図２のフローチャートのステップＳ１０３の処理、すなわち、対称３重対角行列Ｔの固有ベクトルを算出する処理について、より詳細に説明する。まず、図２のフローチャートのステップＳ１０３の処理について、図１１のフローチャートを用いて説明する。

【０１１６】

（ステップＳ６０１）正定値化部２１は、固有値記憶部１８から対称３重対角行列Ｔの固有値を読み出し、その固有値のいずれかが負の値であるかどうか判断する。そして、負の値がある場合には、ステップＳ６０２に進み、そうでない場合には、ステップＳ６０３に進む。

【０１１７】

（ステップＳ６０２）正定値化部２１は、対角行列記憶部１３から対称３重対角行列Ｔを読み出し、その対称３重対角行列Ｔの固有ベクトルを変化させることなく、対称３重対角行列Ｔのすべての固有値が正の値となるように、その対称３重対角行列Ｔを正定値化し、正定値化した後の対称３重対角行列Ｔを対角行列記憶部１３に蓄積し、正定値化した後の固有値を固有値記憶部１８に蓄積する。

【０１１８】

（ステップＳ６０３）コレスキー分解部２２は、対角行列記憶部１３から、すべての固有値が正の値である対称３重対角行列Ｔを読み出し、固有値記憶部１８から、対称３重対角行列Ｔの正の値の固有値を読み出す。なお、正定値化が行われた場合には、正定値化の後の対称３重対角行列Ｔ、及び固有値が読み出されることになる。そして、コレスキー分解部２２は、対称３重対角行列Ｔの各要素に関してｑｄ型変換を行うことによって、対称３重対角行列Ｔを上２重対角行列Ｂ$^{(+1)}$及び下２重対角行列Ｂ$^{(-1)}$にコレスキー分解する。

【０１１９】

（ステップＳ６０４）ベクトル算出部２３は、コレスキー分解部２２が算出した上２重対角行列Ｂ$^{(+1)}$及び下２重対角行列Ｂ$^{(-1)}$の各要素と、対称３重対角行列Ｔの正の値の固有値とを用いて固有ベクトルを算出する。

【０１２０】

（ステップＳ６０５）ベクトル算出部２３は、算出した固有ベクトルを正規化する。すなわち、ベクトル算出部２３は、算出した固有ベクトルのノルムを算出し、固有ベクトルを算出したノルムで割ったものを最終的な固有ベクトルとして固有ベクトル記憶部２０に蓄積する。このようにして、対称３重対角行列Ｔを固有値分解する処理は終了となる。

なお、この後に、行列記憶部１１が記憶している対称行列Ａの固有ベクトルを算出する処理が行われてもよいが、ここでは省略している。

【０１２１】

また、固有ベクトル算出部１９が算出した固有ベクトルの精度を上げるために、図１２で示されるように、固有ベクトルに関する処理を実行してもよく、あるいは、実行しなくてもよい。すなわち、図示しない逆反復法処理部は、固有ベクトル記憶部２０から固有ベクトル算出部１９が算出した固有ベクトルを読み出し、その固有ベクトルに対して逆反復法の処理を実行し、その結果の固有ベクトルを固有ベクトル記憶部２０に蓄積する（ステップＳ７０１）。このように、逆反復法の処理を実行することによって、固有ベクトルの精度を向上させることができうる。次に、図示しないグラムシュミット処理部は、高精度が必要かどうか判断する（ステップＳ７０２）。この判断は、あらかじめ高精度が必要かどうか設定されている記録媒体等から、その設定を読み出すことによって判断してもよい。そして、図示しないグラムシュミット処理部は、高精度が必要な場合には、固有ベクトル記憶部２０から逆反復法の処理の実行された固有ベクトルを読み出し、その固有ベクトルに対してグラムシュミット法の処理を実行し、その結果の固有ベクトルを固有ベクトル記憶部２０に蓄積する（ステップＳ７０３）。このように、グラムシュミット法の処理を実行することによって、固有ベクトルの直交性を高めることができうる。なお、逆反復法や、グラムシュミット法については、すでに公知であり、その詳細な説明を省略する。

【０１２２】

［対称３重対角行列Ｔの正定値化］
行列Ｔを次のように固有値分解できたとする。
【数３８】
$$T = QDQ^{\mathrm{T}}$$

【０１２３】
その行列Ｔを次のようにａ倍してｓＩだけシフトしたとする。ここで、Ｉは行列Ｔと同じ大きさの単位行列であり、ａ，ｓは実数である。
【数３９】
$$T \to \hat{T} = aT + sI$$

【０１２４】
すると、シフト後の固有値分解は、次のようになる。
【数４０】
$$\hat{T} = Q\hat{D}Q^{\mathrm{T}}$$
ただし、$\hat{D} = aD + sI$ である。
【０１２５】
このように、行列Ｔをａ倍してｓＩだけシフトした場合には、各固有値がａ倍されてｓだけシフトされるが、固有ベクトルは変化しない。したがって、この実数倍とシフトとによって、すべての固有値を正の値にしてから、ツイスト分解法を用いた固有ベクトルの算出を行えばよいことになる。
【０１２６】
なお、算出された固有値に負の値が含まれる場合には、例えば、（Ａ）最小の固有値によりシフトを行ってもよく、（Ｂ）最大の固有値によりシフトを行ってもよい。ここで、固有値には、負の値と正の値とが含まれているものとする。固有値がすべて正の値であれば、シフトを行う必要はなく、固有値がすべて負の値であれば、単に行列Ｔを－Ｔとすれば、正定値化することができる。
【０１２７】
（Ａ）最小の固有値によるシフト
最小の固有値を$\lambda_{min}<0$とすると、ａ＝１、ｓ＝$-\lambda_{min}+\lambda_{max}\times\varepsilon$とする。ここで、εとしては、マシン・イプシロンに定数を掛けた値を用いてもよく、他の適切な値であってもよい。このように、行列Ｔをシフトすることによって、行列Ｔのすべての固有値を正の値にすることができる。
【０１２８】
（Ｂ）最大の固有値によるシフト
最大の固有値を$\lambda_{max}>0$とすると、ａ＝－１、ｓ＝$\lambda_{max}-\lambda_{min}\times\varepsilon$とする。この場合には、次のように行列Ｔが変換されることになる。
【数４１】
$$T \to \hat{T} = -T + \lambda_{\max} - \lambda_{\min} \times \varepsilon I = -(T - (\lambda_{\max} - \lambda_{\min} \times \varepsilon)I)$$

【０１２９】
上式の右辺の括弧内では、シフトによってすべての固有値が負の値になるが、全体にマイナスを掛けることによって、正定値化することができている。ここで、εは、（Ａ）の場合と同様にマシン・イプシロンに定数（例えば、２，３，４などの実数であり、計算誤差を少なくするための値を選択することが好適である）を掛けた値を用いてもよく、他の適切な値であってもよい。
【０１３０】
固有値の最小値の絶対値が、固有値の最大値の絶対値よりも小さい場合には、上記（Ａ）のシフトを行い、固有値の最小値の絶対値が、固有値の最大値の絶対値よりも大きい場

合には、上記（Ｂ）のシフトを行うようにしてもよい。このようにすることで、シフトの量を少なくすることができ、丸め誤差を削減することが可能だからである。

なお、正定値化の後に固有ベクトルを算出する処理の詳細については、次のＬＶ型ツイスト分解法の場合と一緒に説明する。

【０１３１】

（２）ＬＶ型ツイスト分解法により固有ベクトルを算出する場合

図１３は、ＬＶ型ツイスト分解法により固有ベクトルを算出する場合の固有ベクトル算出部１９の構成を示す図である。図１３において、固有ベクトル算出部１９は、コレスキー分解部３１と、ベクトル算出部３２とを備える。

【０１３２】

コレスキー分解部３１は、対角行列記憶部１３から、対称３重対角行列Ｔを読み出し、固有値記憶部１８から、対称３重対角行列Ｔの固有値を読み出す。そして、コレスキー分解部３１は、対称３重対角行列Ｔの各要素に関してミウラ変換、ｄＬＶｖ型変換、逆ミウラ変換を行うことによって、対称３重対角行列Ｔを上２重対角行列Ｂ$^{(+1)}$及び下２重対角行列Ｂ$^{(-1)}$にコレスキー分解する。この処理の詳細については後述する。なお、「対称３重対角行列Ｔを上２重対角行列Ｂ$^{(+1)}$及び下２重対角行列Ｂ$^{(-1)}$にコレスキー分解する」ことには、対称３重対角行列Ｔの固有値に単位ベクトルを掛けたものを対称３重対角行列Ｔから引いた行列を、上２重対角行列Ｂ$^{(+1)}$及び下２重対角行列Ｂ$^{(-1)}$にコレスキー分解することが含まれてもよい。

【０１３３】

ベクトル算出部３２は、コレスキー分解部３１が算出した上２重対角行列Ｂ$^{(+1)}$及び下２重対角行列Ｂ$^{(-1)}$の各要素と、対称３重対角行列Ｔの固有値とを用いて固有ベクトルを算出して固有ベクトル記憶部２０に蓄積する。この固有ベクトルを各列に有する行列が、固有値分解の直交行列となる。

【０１３４】

次に、図２のフローチャートのステップＳ１０３の処理、すなわち、対称３重対角行列Ｔの固有ベクトルを算出する処理について、より詳細に説明する。まず、図２のフローチャートのステップＳ１０３の処理について、図１４のフローチャートを用いて説明する。

【０１３５】

（ステップＳ８０１）コレスキー分解部３１は、対角行列記憶部１３から対称３重対角行列Ｔを読み出し、固有値記憶部１８から対称３重対角行列Ｔの固有値を読み出す。そして、コレスキー分解部３１は、対称３重対角行列Ｔの各要素に関してミウラ変換、ｄＬＶｖ型変換、逆ミウラ変換を行うことによって、対称３重対角行列Ｔを上２重対角行列Ｂ$^{(+1)}$及び下２重対角行列Ｂ$^{(-1)}$にコレスキー分解する。

【０１３６】

（ステップＳ８０２）ベクトル算出部３２は、上２重対角行列Ｂ$^{(+1)}$及び下２重対角行列Ｂ$^{(-1)}$の各要素と、対称３重対角行列Ｔの固有値とを用いて固有ベクトルを算出する。

【０１３７】

（ステップＳ８０３）ベクトル算出部３２は、算出した固有ベクトルを正規化する。すなわち、ベクトル算出部３２は、算出した固有ベクトルのノルムを算出し、固有ベクトルを算出したノルムで割ったものを最終的な固有ベクトルとして固有ベクトル記憶部２０に蓄積する。このようにして、対称３重対角行列Ｔを固有値分解する処理は終了となる。

【０１３８】

なお、この後に、行列記憶部１１が記憶している対称行列Ａの固有ベクトルを算出する処理が行われてもよいが、ここでは省略している。また、固有ベクトル算出部１９が算出した固有ベクトルの精度を上げるために、図１２で示されるように、固有ベクトルに関する処理を実行してもよく、あるいは、実行しなくてもよいことは、ｑｄ型ツイスト分解法の場合と同様である。

【０１３９】

［固有値からの固有ベクトルの算出］

固有値から固有ベクトルを算出する処理については、ｑｄ型ツイスト分解法を用いる方法と、ＬＶ型ツイスト分解法を用いる方法とについて説明する。図１１のフローチャートで示されるように、ｑｄ型ツイスト分解法を用いて固有ベクトルを算出する場合に、負の固有値があれば、対称３重対角行列Ｔを正定値化する処理をあらかじめ実行しておく（ステップＳ６０１，Ｓ６０２）。ｑｄ型ツイスト分解法を用いる場合には、この後の処理において、その正定値化の後の対称３重対角行列Ｔ、及び固有値を用いるものとする。

【０１４０】

対角行列記憶部１３では、次のようなｍ×ｍの対称３重対角行列Ｔが記憶されているものとする。

【数４２】

$$T=\begin{pmatrix} t_1 & t_2 & & 0 \\ t_2 & t_3 & \ddots & \\ & \ddots & \ddots & t_{2m-2} \\ 0 & & t_{2m-2} & t_{2m-1} \end{pmatrix}$$

【０１４１】

また、固有値記憶部１８では、固有値λ_i（$1 \leqq i \leqq m$）が記憶されているものとする。このような場合に、対称３重対角行列Ｔの固有ベクトルを求めることは、次の方程式の固有ベクトルx_jを求めることになる。

【数４３】

$$(T-\lambda_j I)\mathbf{x}_j=\gamma_k \mathbf{e}_k \qquad (j=1,2,...,m)$$

ただし、行列Tは$m \times m$行列である。

$\mathbf{e}_k$は、k番目の要素が1で他の要素が0のベクトルである。

（$\mathbf{e}_k$は、単位行列Iのk番目の列である）

【０１４２】

本来であれば、上記式の右辺は０になるはずであるが、行列Ｔの固有値を算出するときに、いくらかの誤差を含むため、固有ベクトルx_jが真値であれば、上記のように右辺に残差項が存在する。

【０１４３】

ここで、以下のようにコレスキー分解できたとする。

【数４４】

$$T-\lambda_j I=\begin{cases}\left(B^{(+1)}\right)^{\mathrm{T}}B^{(+1)}\\ \left(B^{(-1)}\right)^{\mathrm{T}}B^{(-1)}\end{cases}$$

$$B^{(n)}\equiv\begin{pmatrix} b_1^{(n)} & b_2^{(n)} & & \\ & b_3^{(n)} & \ddots & \\ & & \ddots & b_{2m-2}^{(n)} \\ 0 & & & b_{2m-1}^{(n)}\end{pmatrix},\quad (n=0,+1),$$

$$T\equiv B^{(0)\mathrm{T}}B^{(0)}$$

$$B^{(-1)}\equiv\begin{pmatrix} b_1^{(-1)} & & & 0 \\ b_2^{(-1)} & b_3^{(-1)} & & \\ & \ddots & \ddots & \\ & & b_{2m-2}^{(-1)} & b_{2m-1}^{(-1)}\end{pmatrix}$$

$$b_{2k-1}^{(n)}\equiv\xi_k^{(n)}\sqrt{q_k^{(n)}},\quad b_{2k}^{(n)}\equiv\eta_k^{(n)}\sqrt{e_k^{(n)}},$$

$$\left(\xi_k^{(n)}\right)^2=1,\quad \left(\eta_k^{(n)}\right)^2=1,\quad (n=0,+1,-1),$$

$$\xi_k^{(1)}\eta_k^{(1)}=\xi_k^{(0)}\eta_k^{(0)},\quad \xi_{k+1}^{(-1)}\eta_k^{(-1)}=\xi_k^{(0)}\eta_k^{(0)}$$

【０１４４】

ここで、行列Ｔの各要素と、行列Ｂ（０）の各要素との対応は、次のようになる。

【数４５】

$$t_{2k-1}=(b_{2k-2}^{(0)})^2+(b_{2k-1}^{(0)})^2\qquad k=1,2,\cdots,m$$

$$t_{2k}=b_{2k-1}^{(0)}b_{2k}^{(0)}\qquad k=1,2,\cdots,m-1$$

ただし、$b_0^{(0)}=0$ である。

【０１４５】

なお、行列Ｔの各要素と、行列Ｂ（０）の各要素との対応を次のように書くこともできる。下記の式から明らかなように、行列Ｂ（０）の各要素を、行列Ｔの各要素を用いて順次、算出することができる。

【数４６】

$$b_{2k-1}^{(0)}=\sqrt{t_{2k-1}-b_{2k-2}^{(0)}}\qquad k=1,2,\cdots,m$$

$$b_{2k}^{(0)}=\frac{t_{2k}}{b_{2k-1}^{(0)}}\qquad k=1,2,\cdots,m-1$$

ただし、$b_0^{(0)}=0$ である。

【０１４６】

上記のコレスキー分解した結果を、次式のように書くことができる。

【数４７】

$$T-\lambda_j I=\begin{cases} LD^{+}L^{\mathrm{T}} \\ UD^{-}U^{\mathrm{T}} \end{cases}$$

ただし、L,U,Dは以下の通りである。

$$L\equiv\begin{pmatrix} 1 & & & 0 \\ L_1 & 1 & & \\ & \ddots & \ddots & \\ & & L_{m-1} & 1 \end{pmatrix},\qquad L_k\equiv\frac{b_{2k}^{(+1)}}{b_{2k-1}^{(+1)}}$$

$$U\equiv\begin{pmatrix} 1 & U_1 & & \\ & 1 & \ddots & \\ & & \ddots & U_{m-1} \\ 0 & & & 1 \end{pmatrix},\qquad U_k\equiv\frac{b_{2k}^{(-1)}}{b_{2k+1}^{(-1)}}$$

$$D^{\pm}\equiv\mathrm{diag}(D_1^{\pm},D_2^{\pm},...,D_m^{\pm}),\qquad D_k^{\pm}\equiv(b_{2k-1}^{(\pm1)})^2$$

【０１４７】

そして、次のようになる。

【数４８】

$$T-\lambda_j I=N_k D_k\left(N_k\right)^{\mathrm{T}}$$

$$N_k\equiv\begin{pmatrix} 1 & & & & & & \\ L_1 & 1 & & & & & \\ & \ddots & \ddots & & & & \\ & & L_{k-1} & 1 & U_k & & \\ & & & & 1 & \ddots & \\ & & & & & \ddots & U_m \\ & & & & & & 1 \end{pmatrix},$$

$$D_k\equiv\mathrm{diag}(D_1^{+},D_2^{+},...,D_{k-1}^{+},\gamma_k,D_{k+1}^{-},...,D_m^{-})$$

【０１４８】

ここで、行列Ｎ$_k$をツイスト行列と呼ぶ。また、

【数４９】

$$D_k\mathbf{e}_k=\gamma_k\mathbf{e}_k,\quad N_k\mathbf{e}_k=\mathbf{e}_k$$

であるので、（Ｔ－λ$_j$Ｉ）ｘ$_j$＝γ$_k$ｅ$_k$は、

【数５０】

$$N_k^{\mathrm{T}}\mathbf{x}_j=\mathbf{e}_k$$

となる。この簡単な式を解くことによって固有ベクトルを算出することができる。具体的には、

【数５１】

$$x_j(\rho)=\begin{cases}1, & \rho=k,\\ -L_\rho x_j(\rho+1), & \rho=k-1,k-2,...,1,\\ -U_{\rho-1}x_j(\rho-1), & \rho=k+1,k+2,...,m\end{cases}$$

ただし、$x_j(\rho)$は$\mathbf{x}_j$のρ番目の要素である。

のようにわずかな演算で固有ベクトルを算出することができる。なお、あるρ０について、$D_{\rho 0}^{+}=0$あるいは$D_{\rho 0}^{-}=0$であったとしても、行列Ｂ（０）の（ρ，ρ）成分である$b_{2\rho-1}^{(0)}$と、行列Ｂの（ρ，ρ＋１）成分である$b_{2\rho}^{(0)}$を用いて、あるいは、行列Ｔの成分を用いて、

【数５２】

$$x_j(\rho 0)=\begin{cases}-\dfrac{b_{2\rho 0+1}^{(0)}b_{2\rho 0+2}^{(0)}}{b_{2\rho 0-1}^{(0)}b_{2\rho 0}^{(0)}}x_j(\rho 0+2), & \rho 0<k,\\ -\dfrac{b_{2\rho 0-5}^{(0)}b_{2\rho 0-4}^{(0)}}{b_{2\rho 0-3}^{(0)}b_{2\rho 0-2}^{(0)}}x_j(\rho 0-2), & \rho 0>k\end{cases}$$

$$=\begin{cases}-\dfrac{t_{2\rho 0+2}}{t_{2\rho 0}}x_j(\rho 0+2), & \rho 0<k,\\ -\dfrac{t_{2\rho 0-4}}{t_{2\rho 0-2}}x_j(\rho 0-2), & \rho 0>k\end{cases}$$

と固有ベクトルを算出することができる（このように固有ベクトルを算出する処理を例外処理と呼ぶことにする）。なお、残差項のパラメータγ_k及びｋの値は、

【数５３】

$$\gamma_k\equiv q_k^{(+1)}+q_k^{(-1)}-(e_{k-1}^{(0)}+q_k^{(0)}-\lambda_j) \qquad (式5)$$

の絶対値が最小となるように決定する。したがって、上述のコレスキー分解を求め、ツイスト行列N_kを求めることができれば、固有ベクトルを求めることができることになる。そこで、次にコレスキー分解について説明する。

【０１４９】

図１５で示すように、$T-\lambda_j I$をコレスキー分解することは、Ｂ（０）に対応する{$q_k^{(0)}$，$e_k^{(0)}$}から、Ｂ（±１）に対応する{$q_k^{(\pm 1)}$，$e_k^{(\pm 1)}$}を求めることである。

【０１５０】

（ｑｄ型ツイスト分解法）

まず、ｑｄ型ツイスト分解法で用いるｑｄ型変換について説明する（図１５参照）。

【数５４】

$$\text{ｑｄ型変換}：\{q_k^{(0)},e_k^{(0)}\}\mapsto\{q_k^{(\pm 1)},e_k^{(\pm 1)}\}$$

この変換は、さらにｓｔａｔｉｏｎａｒｙ　ｑｄ　ｗｉｔｈ　ｓｈｉｆｔ（ｓｔｑｄｓ）変換

【数５５】

$$\text{ｓｔｑｄｓ変換}：\{q_k^{(0)},e_k^{(0)}\}\mapsto\{q_k^{(+1)},e_k^{(+1)}\}$$

$$q_k^{(+1)}+e_{k-1}^{(+1)}=q_k^{(0)}+e_{k-1}^{(0)}-\lambda_j, \qquad k=1,2,...,m,$$

$$q_k^{(+1)}e_k^{(+1)}=q_k^{(0)}e_k^{(0)}, \qquad k=1,2,...,m-1,$$

$$e_0^{(0)}\equiv 0, \quad e_0^{(+1)}\equiv 0$$

及びｒｅｖｅｒｓｅ－ｔｉｍｅ ｐｒｏｇｒｅｓｓｉｖｅ ｑｄ ｗｉｔｈ ｓｈｉｆｔ（ｒｐｑｄｓ）変換

【数５６】

$$\text{ｒｐｑｄｓ変換}：\{q_k^{(0)},e_k^{(0)}\}\mapsto\{q_k^{(-1)},e_k^{(-1)}\}$$

$$q_k^{(-1)}+e_k^{(-1)}=q_k^{(0)}+e_{k-1}^{(0)}-\lambda_j,\qquad k=1,2,\ldots,m,$$

$$q_{k+1}^{(-1)}e_k^{(-1)}=q_k^{(0)}e_k^{(0)},\qquad k=1,2,\ldots,m-1,$$

$$e_0^{(0)}\equiv 0,\quad e_m^{(-1)}\equiv 0$$

に分けられる。固有値λ$_j$が既知であれば反復的な計算が不要なため、計算量を少なく抑えることができるが、常に数値安定性と精度が高い方法ではない。それは、ｓｔｑｄｓ変換、ｒｐｑｄｓ変換は、共に減算による桁落ちが発生する可能性があるからである。例えば、ｓｔｑｄｓ変換において、ｑ$_k^{(0)}$＋ｅ$_{k-1}^{(0)}$－λ$_j$～ｅ$_{k-1}^{(+1)}$であれば、ｑ$_k^{(+1)}$を求める際に、倍精度計算でもの有効数字がわずか１桁になることもある。その場合には、ｑ$_k^{(0)}$ｅ$_k^{(0)}$／ｑ$_k^{(+1)}$を計算すると誤差が生じる。つまり、ｅ$_k^{(+1)}$が精度よく計算できないことになる。また、ｑ$_{k+1}^{(+1)}$を求めるのにｅ$_k^{(+1)}$が必要であり、ｅ$_k^{(+1)}$を求めるのにｑ$_k^{(+1)}$が必要であるといったように逐次的な計算が要求されるので、１箇所で発生した桁落ちによる誤差が波及し、さらなる誤差増大の可能性も秘めている。その結果、理論上はｑ$_k^{(+1)}$≠０であるが、誤差蓄積によりｑ$_k^{(+1)}$＝０となり、ｑ$_k^{(0)}$ｅ$_k^{(0)}$／ｑ$_k^{(+1)}$の計算においてオーバーフローが起こるといった数値不安定な状況も想定される。Ｂ$^{(0)}$の成分｛ｂ$_{2k-1}^{(0)}$，ｂ$_{2k}^{(0)}$｝が与えられる、すなわち｛ｑ$_k^{(0)}$，ｅ$_k^{(0)}$｝が与えられると、λ$_j$及びｅ$_{k-1}^{(+1)}$が一意的に決まるので、この状況は避けることはできない。ｒｐｑｄｓ変換も同様の性質を持つため、実用的なレベルにまで達したとはいいがたい。ＬＡＰＡＣＫにおいてＦＯＲＴＲＡＮルーチンＤＳＴＥＧＲとして改良版が公開されているものの欠点は完全に解決されてはいない。

【０１５１】

なお、このように対角行列Ｔの各要素を、行列Ｂ$^{(0)}$の各要素に変換して、その行列Ｂ$^{(0)}$の各要素についてｑｄ型変換を行う場合であっても、間接的に対角行列Ｔの各要素に関してｑｄ型変換を行うことになるため、「対角行列Ｔの各要素に関してｑｄ型変換を行う」と言うことにする。

また、ここではスタンダードなｑｄ型変換について説明したが、ｑｄ型変換はｄｉｆｆｅｒｅｎｔｉａｌ ｑｄ型変換であってもよい。

【０１５２】

（ＬＶ型ツイスト分解法）

次に、ＬＶ型ツイスト分解法で用いるミウラ変換、ｄＬＶｖ型変換、逆ミウラ変換について説明する（図１５参照）。

【数５７】

$$\text{ミウラ変換}：\{q_k^{(0)},e_k^{(0)}\}\mapsto\{u_{2k-1}^{(0)},u_{2k}^{(0)}\}$$

$$\text{ｓｔｄＬＶｖ変換}：u_k^{(0)}\mapsto u_k^{(+1)}$$

$$\text{ｒｄＬＶｖ変換}：u_k^{(0)}\mapsto u_k^{(-1)}$$

$$\text{逆ミウラ変換}：\{u_{2k-1}^{(\pm1)},u_{2k}^{(\pm1)}\}\mapsto\{q_k^{(\pm1)},e_k^{(\pm1)}\}$$

【０１５３】

まず、ミウラ変換について説明する。この変換は、次のように示される。

【数５８】

$$u_{2k-1}^{(0)} = \frac{t_k^{(0)}}{\delta^{(0)}}, \qquad k = 1,2,...,m,$$

$$u_{2k}^{(0)} = \frac{e_k^{(0)}}{t_k^{(0)}}, \qquad k = 1,2,...,m-1,$$

$$t_k^{(0)} \equiv \frac{q_k^{(0)}}{\frac{1}{\delta^{(0)}} + u_{2k-2}^{(0)}} - 1,$$

$$u_0^{(0)} \equiv 0,$$

ただし、$\delta^{(0)}$は任意である。

【０１５４】

次に、ｄＬＶｖ型変換について説明する。この変換は、さらにｓｔａｔｉｏｎａｒｙ ｄｉｓｃｒｅｔｅ Ｌｏｔｋａ－Ｖｏｌｔｅｒｒａ ｗｉｔｈ ｖａｒｉａｂｌｅ ｓｔｅｐ－ｓｉｚｅ（ｓｔｄＬＶｖ）変換

【数５９】

$$u_k^{(+1)} = u_k^{(0)} \times \frac{1+\delta^{(0)}u_{k-1}^{(0)}}{1+\delta^{(+1)}u_{k-1}^{(+1)}}, \qquad k = 1,2,...,2m-1,$$

$$u_0^{(0)} \equiv 0, \quad u_0^{(+1)} \equiv 0$$

及びｒｅｖｅｒｓｅ－ｔｉｍｅ ｄｉｓｃｒｅｔｅ Ｌｏｔｋａ－Ｖｏｌｔｅｒｒａ ｗｉｔｈ ｖａｒｉａｂｌｅ ｓｔｅｐ－ｓｉｚｅ（ｒｄＬＶｖ）変換

【数６０】

$$u_k^{(-1)} = u_k^{(0)} \times \frac{1+\delta^{(0)}u_{k-1}^{(0)}}{1+\delta^{(-1)}u_{k+1}^{(-1)}}, \qquad k = 1,2,...,2m-1,$$

$$u_0^{(0)} \equiv 0, \quad u_{2m}^{(-1)} \equiv 0$$

に分けられる。ただし、

【数６１】

$$\frac{1}{\delta^{(0)}} - \frac{1}{\delta^{(n)}} = \lambda_j, \quad (n = +1,-1)$$

を満たす範囲内で$\delta^{(\pm 1)}$は任意である。

【０１５５】

最後に、逆ミウラ変換について説明する。この変換は、次のように示される。

【数６２】

$$q_k^{(\pm 1)} = \frac{(1+\delta^{(\pm 1)}u_{2k-2}^{(\pm 1)}) \times (1+\delta^{(\pm 1)}u_{2k-1}^{(\pm 1)})}{\delta^{(\pm 1)}}, \quad k = 1,2,...,m,$$

$$e_k^{(\pm 1)} = \delta^{(\pm 1)}u_{2k-1}^{(\pm 1)}u_{2k}^{(\pm 1)}, \quad k = 1,2,...,m-1$$

$$u_0^{(\pm 1)} \equiv 0$$

【０１５６】

このようにして、ｑｄ型変換と同様に、コレスキー分解を行うことができる。ｑｄ型変換では見られない離散Ｌｏｔｋａ－Ｖｏｌｔｅｒｒａ型変換の大きな特徴は、任意パラメータを持つことである。すなわち、$\lambda_j = 1/\delta^{(0)} - 1/\delta^{(\pm 1)}$を満たす範囲で$\delta^{(n)}$の値を任意に設定できる。$\delta^{(n)}$を変動させると補助変数$u_k^{(n)}$の値も変化するが、桁落ちによる誤差や数値不安定が発生するかどうかは事前に判定できる。この判定は、ｉｆ文によって実装されてもよい。この場合は、$\delta^{(n)}$を再設定後に再度計算すればよい。また、$u_k^{(\pm 1)}$が求まれば逆ミウラ変換によって$q_k^{(\pm 1)}$及びe_k

$^{(\pm 1)}$が独立に計算されるので、誤差が伝播しないという性質を持つ。なお、逆ミウラ変換をミウラ変換、ミウラ変換を逆ミウラ変換と呼んでもよく、ｓｔｄＬＶｖ変換をｓｔＬＶｖ変換と呼んでもよく、ｒｄＬＶｖ変換をｒＬＶｖ変換と呼んでもよい。

【０１５７】

なお、このように対角行列Ｔの各要素を、行列Ｂ$^{(0)}$の各要素に変換して、その行列Ｂ$^{(0)}$の各要素についてＬＶ型変換を行う場合であっても、間接的に対角行列Ｔの各要素に関してＬＶ型変換を行うことになるため、「対角行列Ｔの各要素に関してＬＶ型変換を行う」と言うことにする。

【０１５８】

ここで、ＬＶ型ツイスト分解法による処理のより詳細な処理の一例について説明する。図１６～図２１は、ＬＶ型ツイスト分解法による処理の一例を示すフローチャートである。

【０１５９】

図１６は、コレスキー分解の全体の処理の一例を示す図である。

（ステップＳ９０１）コレスキー分解部３１は、ミウラ変換を行う。この処理の詳細については後述する。

【０１６０】

（ステップＳ９０２）コレスキー分解部３１は、$1/\delta^{(\pm 1)}$を$1/\delta^{(0)}-\lambda_j$とする。

（ステップＳ９０３）コレスキー分解部３１は、後述するＰｒｏｃｅｄｕｒｅ１の処理を実行する。

【０１６１】

（ステップＳ９０４）コレスキー分解部３１は、後述するＰｒｏｃｅｄｕｒｅ２の処理を実行する。

（ステップＳ９０５）コレスキー分解部３１は、$q_k^{(+1)}$，$e_k^{(+1)}$がすでに算出されているかどうか判断する。そして、すでに算出されている場合には、コレスキー分解の一連の処理は終了となり、一方、算出されていない場合には、ステップＳ９０１に戻る。

【０１６２】

図１７は、図１６のフローチャートにおけるステップＳ９０３の処理の詳細を示すフローチャートである。

（ステップＳ１００１）コレスキー分解部３１は、$q_k^{(+1)}$，$e_k^{(+1)}$がすでに算出されているかどうか判断する。そして、すでに算出されている場合には、終了となり、算出されていない場合には、ステップＳ１００２に進む。

（ステップＳ１００２）コレスキー分解部３１は、ｓｔｄＬＶｖ変換等の処理を行う。この処理の詳細については後述する。

【０１６３】

図１８は、図１６のフローチャートにおけるステップＳ９０４の処理の詳細を示すフローチャートである。

（ステップＳ１１０１）コレスキー分解部３１は、$q_k^{(-1)}$，$e_k^{(-1)}$がすでに算出されているかどうか判断する。そして、すでに算出されている場合には、終了となり、算出されていない場合には、ステップＳ１１０２に進む。

（ステップＳ１１０２）コレスキー分解部３１は、ｒｄＬＶｖ変換等の処理を行う。この処理の詳細については後述する。

【０１６４】

図１９は、図１６のフローチャートのステップＳ９０１の処理の詳細を示すフローチャートである。

（ステップＳ１２０１）コレスキー分解部３１は、$1/\delta^{(0)}$の値を決定する。この値は、前述のように任意に決定することができるため、行列を正定値化する値に決定する。例えば、コレスキー分解部３１は、$1/\delta^{(0)}$の値を次式のように決定する。

【数６３】

$$-\frac{1}{\delta^{(0)}} = |\lambda_{\min}| + \varepsilon^{(0)} \times |\lambda_{\max}|$$

【０１６５】

なお、なお、ε（０）は、マシン・イプシロンに定数（例えば、２，３，４などの実数であり、計算誤差を少なくするための値を選択することが好適である）を掛けた値を用いてもよく、他の適切な値であってもよい。その後、例えば、ステップＳ１２０３等で桁落ちの発生する可能性があると判断された場合に、コレスキー分解部３１は、１／δ（０）の値を決めるε（０）の値を、例えば、マシン・イプシロンに掛ける定数を１ずつ大きくしながら設定していってもよい。なお、１／δ（０）の値を決定する方法は、これに限定されないことは言うまでもない。このように、ＬＶ型ツイスト分解法では、この１／δ（０）の値を自由に設定することができるため、ｑｄ型ツイスト分解法の場合のように、行列Ｔの正定値化を行わなくてもよいことになる。なお、ｑｄ型ツイスト分解法の場合と同様に、行列Ｔの正定値化の処理を行ってから、ＬＶ型ツイスト分解法を実行してもよいことは言うまでもない。

（ステップＳ１２０２）コレスキー分解部３１は、β１を$q_1^{(0)}$－１／δ（０）に設定する。

【０１６６】

（ステップＳ１２０３）コレスキー分解部３１は、β１の絶対値がεよりも大きいかどうか判断する。そして、大きい場合には、ステップＳ１２０４に進み、そうでない場合には、ステップＳ１２０１に戻って、１／δ（０）の値を決定する処理を再度実行する。なお、このεも任意に定めることができる。例えば、εとしてマシン・イプシロンを用いてもよい。εの値を大きくすると精度が向上し、εの値を小さくすると精度が低下する。なお、このステップＳ１２０３で行っている処理は、桁落ちが発生する可能性について判断する処理である。β１の絶対値がεよりも大きくない場合には、桁落ちの発生する可能性があると判断されることになる。

【０１６７】

（ステップＳ１２０４）コレスキー分解部３１は、β２を$q_1^{(0)}$／（１＋δ（０）$u_0^{(0)}$）に設定する。なお、前述のように$u_0^{(0)}$＝０であるため、β２は$q_1^{(0)}$に設定されたことになる。

【０１６８】

（ステップＳ１２０５）コレスキー分解部３１は、$u_1^{(0)}$（１＋δ（０）$u_0^{(0)}$）をβ１に設定する。ここで、β１は、ステップＳ１２０２において$q_1^{(0)}$－１／δ（０）に設定されているが、前述のように$u_0^{(0)}$＝０であるため、β１は、$q_1^{(0)}$／（１＋δ（０）$u_0^{(0)}$）－１／δ（０）と等しく、これにミウラ変換を行うと$u_1^{(0)} = u_{2k-1}^{(0)}|_{k=1}$となるからである。なお、$u_0^{(0)}$＝０から（１＋δ（０）$u_0^{(0)}$）＝１である。

【０１６９】

（ステップＳ１２０６）コレスキー分解部３１は、カウンタｋを「１」に設定する。

（ステップＳ１２０７）コレスキー分解部３１は、α１を$e_k^{(0)}$／β１に設定する。前述のように、β１は$u_{2k-1}^{(0)}$となるから、$e_k^{(0)}$／β１にミウラ変換を行うと、α１はδ（０）$u_{2k}^{(0)}$と等しいことになる。

【０１７０】

（ステップＳ１２０８）コレスキー分解部３１は、α２をα１＋１に設定する。ステップＳ１２０７の説明からわかるように、α２は１＋δ（０）$u_{2k}^{(0)}$と等しいことになる。

【０１７１】

（ステップＳ１２０９）コレスキー分解部３１は、α２の絶対値がεよりも大きいかどうか判断する。そして、大きい場合には、ステップＳ１２１０に進み、そうでない場合に

は、ステップＳ１２０１に戻って、１／$\delta^{(0)}$の値を決定する処理を再度実行する。なお、このステップＳ１２０９で行っている処理は、ステップＳ１２０３の処理と同様に、桁落ちが発生する可能性について判断する処理である。α２の絶対値がεよりも大きくない場合には、桁落ちの発生する可能性があると判断されることになる。

【０１７２】

（ステップＳ１２１０）コレスキー分解部３１は、α１×β２を算出して$u_{2k}^{(0)}$（１＋$\delta^{(0)}u_{2k-1}^{(0)}$）に設定する。前述のように、α１は$\delta^{(0)}u_{2k}^{(0)}$と等しく、β２は$q_k^{(0)}$／（１＋$\delta^{(0)}u_{2k-2}^{(0)}$）であるため、α１×β２にミウラ変換を実行すると、$u_{2k}^{(0)}$（１＋$\delta^{(0)}u_{2k-1}^{(0)}$）に等しくなるからである。

【０１７３】

（ステップＳ１２１１）コレスキー分解部３１は、β２を$q_{k+1}^{(0)}$／α２に設定する。前述のように、α２は１＋$\delta^{(0)}u_{2k}^{(0)}$と等しいため、β２は、$q_{k+1}^{(0)}$／（１＋$\delta^{(0)}u_{2k}^{(0)}$）となり、β２におけるｋの値を１だけインクリメントしたことになる。

【０１７４】

（ステップＳ１２１２）コレスキー分解部３１は、β１をβ２－１／$\delta^{(0)}$に設定する。前述のように、β２は$q_{k+1}^{(0)}$／（１＋$\delta^{(0)}u_{2k}^{(0)}$）と等しいため、β１は、$q_{k+1}^{(0)}$／（１＋$\delta^{(0)}u_{2k}^{(0)}$）－１／$\delta^{(0)}$となり、ミウラ変換を実行すると、β１は、$u_{2k+1}^{(0)}$となる。したがって、β１におけるｋの値を１だけインクリメントしたことになる。

【０１７５】

（ステップＳ１２１３）コレスキー分解部３１は、β１の絶対値がεよりも大きいかどうか判断する。そして、大きい場合には、ステップＳ１２１４に進み、そうでない場合には、ステップＳ１２０１に戻って、１／$\delta^{(0)}$の値を決定する処理を再度実行する。なお、このステップＳ１２１３で行っている処理は、ステップＳ１２０３の処理と同様に、桁落ちが発生する可能性について判断する処理である。β１の絶対値がεよりも大きくない場合には、桁落ちの発生する可能性があると判断されることになる。

【０１７６】

（ステップＳ１２１４）コレスキー分解部３１は、α２×β１を算出して$u_{2k+1}^{(0)}$（１＋$\delta^{(0)}u_{2k}^{(0)}$）に設定する。前述のように、α２は１＋$\delta^{(0)}u_{2k}^{(0)}$と等しく、β１は、$u_{2k+1}^{(0)}$に等しいからである。

（ステップＳ１２１５）コレスキー分解部３１は、カウンタｋを１だけインクリメントする。

【０１７７】

（ステップＳ１２１６）コレスキー分解部３１は、カウンタｋがｍに等しいかどうか判断する。そして、ｍに等しい場合には、一連の処理は終了となり、そうでない場合には、ステップＳ１２０７に戻る。

【０１７８】

このようにして、$u_{2k+1}^{(0)}$（１＋$\delta^{(0)}u_{2k}^{(0)}$）と、$u_{2k}^{(0)}$（１＋$\delta^{(0)}u_{2k-1}^{(0)}$）とが算出されることになる。これらの値は、コレスキー分解部３１が有する図示しないメモリ等において一時的に記憶されてもよい。

【０１７９】

図２０は、図１７のフローチャートのステップＳ１００２の処理の詳細を示すフローチャートである。

（ステップＳ１３０１）コレスキー分解部３１は、α２を１＋$\delta^{(+1)}u_0^{(+1)}$に設定する。なお、前述のように$u_0^{(+1)}$＝０であるため、α２は１に設定されたことになる。

【０１８０】

（ステップＳ１３０２）コレスキー分解部３１は、β１を$u_1^{(0)}$（１＋$\delta^{(0)}u$

$_0{}^{(0)}$）／（１＋δ$^{(+1)}$ｕ$_0{}^{(+1)}$）に設定する。ここで、ｕ$_1{}^{(0)}$（１＋δ$^{(0)}$ｕ$_0{}^{(0)}$）の値としては、ステップＳ１２０５で算出したものを用いる。なお、前述のようにｕ$_0{}^{(+1)}$＝０である。また、このβ１の式にｓｔｄＬＶｖ変換を実行すると、β１は、ｕ$_1{}^{(+1)}$に設定されたことになる。

【０１８１】

（ステップＳ１３０３）コレスキー分解部３１は、カウンタｋを「１」に設定する。

（ステップＳ１３０４）コレスキー分解部３１は、α１をβ１＋１／δ$^{(+1)}$に設定する。

【０１８２】

（ステップＳ１３０５）コレスキー分解部３１は、α１の絶対値がεよりも大きいかどうか判断する。そして、大きい場合には、ステップＳ１３０６に進み、そうでない場合には、図１６のフローチャートのＰｒｏｃｅｄｕｒｅ２（ステップＳ９０４）に進む。なお、このステップＳ１３０５で行っている処理は、ステップＳ１２０３の処理と同様に、桁落ちが発生する可能性について判断する処理である。α１の絶対値がεよりも大きくない場合には、桁落ちの発生する可能性があると判断されることになる。

【０１８３】

（ステップＳ１３０６）コレスキー分解部３１は、β２をｕ$_{2k}{}^{(0)}$（１＋δ$^{(0)}$ｕ$_{2k-1}{}^{(0)}$）／α１に設定する。ここで、ｕ$_{2k}{}^{(0)}$（１＋δ$^{(0)}$ｕ$_{2k-1}{}^{(0)}$）の値としては、ステップＳ１２１０で算出したものを用いる。また、このβ２の式にｓｔｄＬＶｖ変換を実行すると、β２は、δ$^{(+1)}$ｕ$_{2k}{}^{(+1)}$に設定されたことになる。

【０１８４】

（ステップＳ１３０７）コレスキー分解部３１は、α１×α２を算出する。前述のように、α１はβ１＋１／δ$^{(+1)}$＝ｕ$_{2k-1}{}^{(+1)}$＋１／δ$^{(+1)}$に等しく、α２は１＋δ$^{(+1)}$ｕ$_{2k-2}{}^{(+1)}$に等しいため、α１×α２に逆ミウラ変換を実行するとｑ$_k{}^{(+1)}$に等しくなる。したがって、コレスキー分解部３１は、α１×α２を算出してｑ$_k{}^{(+1)}$に設定する。

【０１８５】

（ステップＳ１３０８）コレスキー分解部３１は、α２を１＋β２に設定する。前述のように、β２はδ$^{(+1)}$ｕ$_{2k}{}^{(+1)}$と等しいため、α２は１＋δ$^{(+1)}$ｕ$_{2k}{}^{(+1)}$となる。したがって、α２におけるｋの値を１だけインクリメントしたことになる。

【０１８６】

（ステップＳ１３０９）コレスキー分解部３１は、α２の絶対値がεよりも大きいかどうか判断する。そして、大きい場合には、ステップＳ１３１０に進み、そうでない場合には、図１６のフローチャートのＰｒｏｃｅｄｕｒｅ２（ステップＳ９０４）に進む。なお、このステップＳ１３０９で行っている処理は、ステップＳ１２０３の処理と同様に、桁落ちが発生する可能性について判断する処理である。α２の絶対値がεよりも大きくない場合には、桁落ちの発生する可能性があると判断されることになる。

【０１８７】

（ステップＳ１３１０）コレスキー分解部３１は、β１×β２を算出する。前述のように、β１はｕ$_{2k-1}{}^{(+1)}$に等しく、β２はδ$^{(+1)}$ｕ$_{2k}{}^{(+1)}$と等しいため、β１×β２に逆ミウラ変換を実行すると、ｅ$_k{}^{(+1)}$に等しくなる。したがって、コレスキー分解部３１は、β１×β２を算出してｅ$_k{}^{(+1)}$に設定する。

【０１８８】

（ステップＳ１３１１）コレスキー分解部３１は、β１をｕ$_{2k+1}{}^{(0)}$（１＋δ$^{(0)}$ｕ$_{2k}{}^{(0)}$）／α２に設定する。ここで、ｕ$_{2k+1}{}^{(0)}$（１＋δ$^{(0)}$ｕ$_{2k}{}^{(0)}$）の値としては、ステップＳ１２１４で算出したものを用いる。また、このβ１の式にｓｔｄＬＶｖ変換を実行すると、β１は、ｕ$_{2k+1}{}^{(+1)}$に設定されたことになる。したがって、β１におけるｋの値を１だけインクリメントしたことになる。

【０１８９】
（ステップＳ１３１２）コレスキー分解部３１は、カウンタｋを１だけインクリメントする。
（ステップＳ１３１３）コレスキー分解部３１は、カウンタｋがｍに等しいかどうか判断する。そして、ｍに等しい場合には、ステップＳ１３１４に進み、そうでない場合には、ステップＳ１３０４に戻る。
【０１９０】
（ステップＳ１３１４）コレスキー分解部３１は、$\alpha 2\times(\beta 1+1/\delta^{(+1)})$を算出する。これは、ステップＳ１３０４で$\alpha 1$を更新した後に、ステップＳ１３０７で$\alpha 1\times\alpha 2$を計算することと等しい。したがって、コレスキー分解部３１は、$\alpha 2\times(\beta 1+1/\delta^{(+1)})$を算出して$q_1^{(+1)}$に設定する。このようにして、ミウラ変換された結果にｓｔｄＬＶｖ変換と、逆ミウラ変換とを実行して、$q_k^{(+1)}$、$e_k^{(+1)}$を算出する処理が終了する。これらの値は、コレスキー分解部３１が有する図示しないメモリ等において一時的に記憶されてもよい。
【０１９１】
図２１は、図１８のフローチャートのステップＳ１１０２の処理の詳細を示すフローチャートである。
（ステップＳ１４０１）コレスキー分解部３１は、$\beta 1$を$u_{2m-1}^{(0)}(1+\delta^{(0)}u_{2m-2}^{(0)})/(1+\delta^{(-1)}u_{2m}^{(-1)})$に設定する。ここで、$u_{2m-1}^{(0)}(1+\delta^{(0)}u_{2m-2}^{(0)})$の値としては、ステップＳ１２１４で算出したものを用いる。なお、前述のように$u_{2m}^{(-1)}=0$である。また、この$\beta 1$の式にｒｄＬＶｖ変換を実行すると、$\beta 1$は、$u_{2m-1}^{(-1)}$に設定されたことになる。
【０１９２】
（ステップＳ１４０２）コレスキー分解部３１は、カウンタｋを「ｍ－１」に設定する。
（ステップＳ１４０３）コレスキー分解部３１は、$\alpha 1$を$\beta 1+1/\delta^{(-1)}$に設定する。
【０１９３】
（ステップＳ１４０４）コレスキー分解部３１は、$\alpha 1$の絶対値がεよりも大きいかどうか判断する。そして、大きい場合には、ステップＳ１４０５に進み、そうでない場合には、図１６のフローチャートのミウラ変換（ステップＳ９０１）に戻る。なお、このステップＳ１４０４で行っている処理は、ステップＳ１２０３の処理と同様に、桁落ちが発生する可能性について判断する処理である。$\alpha 1$の絶対値がεよりも大きくない場合には、桁落ちの発生する可能性があると判断されることになる。
【０１９４】
（ステップＳ１４０５）コレスキー分解部３１は、$\beta 2$を$u_{2k}^{(0)}(1+\delta^{(0)}u_{2k-1}^{(0)})/\alpha 1$に設定する。ここで、$u_{2k}^{(0)}(1+\delta^{(0)}u_{2k-1}^{(0)})$の値としては、ステップＳ１２１０で算出したものを用いる。また、この$\beta 2$の式にｒｄＬＶｖ変換を実行すると、$\beta 2$は、$\delta^{(-1)}u_{2k}^{(-1)}$に設定されたことになる。
【０１９５】
（ステップＳ１４０６）コレスキー分解部３１は、$\alpha 2$を$1+\beta 2$に設定する。前述のように、$\beta 2$は$\delta^{(-1)}u_{2k}^{(-1)}$と等しいため、$\alpha 2$は$1+\delta^{(-1)}u_{2k}^{(-1)}$となる。
【０１９６】
（ステップＳ１４０７）コレスキー分解部３１は、$\alpha 2$の絶対値がεよりも大きいかどうか判断する。そして、大きい場合には、ステップＳ１４０８に進み、そうでない場合には、図１６のフローチャートのミウラ変換（ステップＳ９０１）に戻る。なお、このステップＳ１４０７で行っている処理は、ステップＳ１２０３の処理と同様に、桁落ちが発生

する可能性について判断する処理である。α２の絶対値がεよりも大きくない場合には、桁落ちの発生する可能性があると判断されることになる。

【０１９７】

（ステップＳ１４０８）コレスキー分解部３１は、α１×α２を算出する。前述のように、α１はβ１＋１／δ$^{(-1)}$＝$u_{2k+1}^{(-1)}$＋１／δ$^{(-1)}$に等しく、α２は１＋δ$^{(-1)}$$u_{2k}^{(-1)}$に等しいため、α１×α２に逆ミウラ変換を実行すると$q_{k+1}^{(-1)}$に等しくなる。したがって、コレスキー分解部３１は、α１×α２を算出して$q_{k+1}^{(-1)}$に設定する。

【０１９８】

（ステップＳ１４０９）コレスキー分解部３１は、β１を$u_{2k-1}^{(0)}$（１＋δ$^{(0)}$$u_{2k-2}^{(0)}$）／α２に設定する。ここで、$u_{2k-1}^{(0)}$（１＋δ$^{(0)}$$u_{2k-2}^{(0)}$）の値としては、ステップＳ１２０５，Ｓ１２１４で算出したものを用いる。また、このβ１の式にｓｔｄＬＶｖ変換を実行すると、β１は、$u_{2k-1}^{(-1)}$に設定されたことになる。したがって、β１におけるｋの値を１だけデクリメントしたことになる。

【０１９９】

（ステップＳ１４１０）コレスキー分解部３１は、β１×β２を算出する。前述のように、β１は$u_{2k-1}^{(-1)}$に等しく、β２はδ$^{(-1)}$$u_{2k}^{(-1)}$と等しいため、β１×β２に逆ミウラ変換を実行すると、$e_k^{(-1)}$に等しくなる。したがって、コレスキー分解部３１は、β１×β２を算出して$e_k^{(-1)}$に設定する。

【０２００】

（ステップＳ１４１１）コレスキー分解部３１は、カウンタｋを１だけデクリメントする。

（ステップＳ１４１２）コレスキー分解部３１は、カウンタｋが０に等しいかどうか判断する。そして、０に等しい場合には、ステップＳ１４１３に進み、そうでない場合には、ステップＳ１４０３に戻る。

【０２０１】

（ステップＳ１４１３）コレスキー分解部３１は、β１＋１／δ$^{(-1)}$を算出する。これは、ステップＳ１４０３でα１を更新した後に、ステップＳ１４０８でα１×α２を計算することと等しい。この場合、α２は１だからである。したがって、コレスキー分解部３１は、β１＋１／δ$^{(-1)}$を算出して$q_1^{(-1)}$に設定する。このようにして、ミウラ変換された結果にｒｄＬＶｖ変換と、逆ミウラ変換とを実行して、$q_k^{(-1)}$、$e_k^{(-1)}$を算出する処理が終了する。これらの値は、コレスキー分解部３１が有する図示しないメモリ等において一時的に記憶されてもよい。

【０２０２】

なお、図１６のフローチャートの処理によって算出することができるのは１個の固有値に対応する固有ベクトルであるため、全ての固有値に対応する固有ベクトルを算出する場合には、コレスキー分解部３１は、図１６のフローチャートの処理を固有値の数だけ繰り返すことになる。

【０２０３】

メモリ消費を抑えるために、補助変数$u_k^{(n)}$のための配列は必ずしも用意する必要はない。一方、１＋δ$^{(0)}$$u^{(0)}$のためのメモリ領域を確保し、ミウラ変換、ｄＬＶｖ型変換、逆ミウラ変換のステップにまたがってこの値を利用することで、メモリ消費を抑え、計算量を低減することができる。これにより、誤差も低減される。

【０２０４】

ここで、誤差について説明する。人間が理想的な状況、すなわち、無限桁の計算をいくらでもできるとすると、ｑｄ型ツイスト分解法であっても、ＬＶ型ツイスト分解法であっても問題ない。しかしながら、コンピュータで計算を行う場合は注意が必要である。有限桁の計算しか行えないコンピュータ上では、数学的に正しい計算法を使ったとしても必ずしも正しい結果が得られる訳ではない。そればかりか、いつまでたっても計算が終了しな

いといった思わぬ数値的な問題が発生する場合もある。

【０２０５】

コンピュータ計算による誤差には、丸め誤差及び桁落ちによる誤差などが知られている。丸め誤差単独では、高々有効桁の最後の桁が真値と比べて異なる程度で大きな誤差にはならない。また、指数部が異なる２つの実数の加算、乗算、除算の少なくとも１つの演算を行えばやはり丸め誤差が生じるが、それ以上の誤差は発生しない。さらに、このような丸め誤差を発生するような操作が繰り返されても、丸めモードがｎｅａｒ（四捨五入）ならば、一方的に切り上げられたり、あるいは切り捨てられたりして誤差が極端に蓄積することは少ない。よって、多くの数値計算法は加算、乗算、除算の少なくとも１つの演算によって発生する丸め誤差を特別注意することは少なく、適切な固有値計算でも結果的に丸め誤差は一様に増大しない。

【０２０６】

問題となるのは、同符号の実数の減算及び異符号の実数の加算により生じる、桁落ちである。桁落ちによる誤差で値が０となった後、その値による除算を行うと、０が分母にくるような不定形となり計算不可能となる。こうなるといつまでたっても計算が終了しない。減算→除算と計算する部分がｑｄ型ツイスト分解法、ＬＶ型ツイスト分解法の両方に存在するので、桁落ち誤差には十分に注意する必要がある。

【０２０７】

ＬＶ型ツイスト分解法では、上述の桁落ちによる誤差を含んでいるかどうかは減算によって得られた値が小さいかどうかで判断できる。ｑｄ型ツイスト分解法の場合、桁落ち誤差を含むことが分かったとしても、それを回避することはできない。なぜならば、初期値として$\{q_k^{(0)}, e_k^{(0)}\}$が与えられると、λ_jは一意的に決定され、$\{q_k^{(\pm 1)}, e_k^{(\pm 1)}\}$も一意的に導出されるためである。すなわち、任意パラメータを持たない自由度のない計算法であるためである。

【０２０８】

それに対して、ＬＶ型ツイスト分解法は、自由に設定できるパラメータ$\delta^{(0)}$を持つため、補助変数$u_k^{(n)}$の値を様々に変化させることができる（図２２参照）。すなわち、様々な経路で$\{q_k^{(\pm 1)}, e_k^{(\pm 1)}\}$を計算することができる。よって、桁落ちが発生する場合も回避できる。図１９～図２１の条件判定によって桁落ちの影響をチェックし、減算によって得られた値の絶対値が小さな数εより大きいという条件が満たされなければ、パラメータ$\delta^{(0)}$の設定に戻るというものである。この処理は、条件が満たされるまで繰り返される。なお、精度よりも高速性を重視する場合は、数回条件が満たさなければ（$q_k^{(+1)}=0$あるいは$q_k^{(-1)}=0$ならば）、例外処理を行ってもよい。

【０２０９】

なお、前述の行列Ｔの各要素と、行列$B^{(0)}$の各要素との対応を示す式を用いると、ｑｄ型変換を次のように書き換えることもできる。

【数６４】

ｑｄ型変換：$\{t_{2k-1}, t_{2k}\} \mapsto \{q_k^{(\pm 1)}, e_k^{(\pm 1)}\}$

【０２１０】

この変換は、さらにｓｔｑｄｓ変換

【数６５】

ｓｔｑｄｓ変換：$\{t_{2k-1}, t_{2k}\} \mapsto \{q_k^{(+1)}, e_k^{(+1)}\}$

$$q_k^{(+1)} + e_{k-1}^{(+1)} = t_{2k-1} - \lambda_j, \qquad k = 1,2,\ldots,m,$$

$$q_k^{(+1)} e_k^{(+1)} = (t_{2k})^2, \qquad k = 1,2,\ldots,m-1,$$

$$e_0^{(+1)} \equiv 0$$

及びｒｐｑｄｓ変換

【数６６】

ｒｐｑｄｓ変換：$\{t_{2k-1},t_{2k}\}\mapsto\{q_k^{(-1)},e_k^{(-1)}\}$

$$q_k^{(-1)}+e_k^{(-1)}=t_{2k-1}-\lambda_j,\qquad k=1,2,...,m,$$

$$q_{k+1}^{(-1)}e_k^{(-1)}=(t_{2k})^2,\qquad k=1,2,...,m-1,$$

$$e_m^{(-1)}\equiv 0$$

に分けられる。このように、｛ｔ$_{2k-1}$，ｔ$_{2k}$｝から｛ｑ$_k^{(\pm1)}$，ｅ$_k^{(\pm1)}$｝を求める場合には、｛ｔ$_{2k-1}$，ｔ$_{2k}$｝から｛ｑ$_k^{(0)}$，ｅ$_k^{(0)}$｝を求め、さらにｑ$_k^{(0)}$，ｅ$_k^{(0)}$｝から｛ｑ$_k^{(\pm1)}$，ｅ$_k^{(\pm1)}$｝を求める場合に比べて、計算量が少なくなることとなり、計算速度を向上させることも可能である。

【０２１１】

なお、この場合には、対角行列Ｔの各要素について直接ｑｄ型変換が行われることになる。したがって、この変換が、「対角行列Ｔの各要素に関してｑｄ型変換を行う」場合に含まれることは言うまでもない。

【０２１２】

また、ｑｄ型変換と同様に、前述の行列Ｔの各要素と、行列Ｂ$^{(0)}$の各要素との対応を示す式を用いると、ＬＶ型変換におけるミウラ変換を次のように書き換えることもできる。

【数６７】

ミウラ変換：$\{t_{2k-1},t_{2k}\}\mapsto\{u_{2k-1}^{(0)},u_{2k}^{(0)}\}$

ｓｔｄＬＶｖ変換：$u_k^{(0)}\mapsto u_k^{(+1)}$

ｒｄＬＶｖ変換：$u_k^{(0)}\mapsto u_k^{(-1)}$

逆ミウラ変換：$\{u_{2k-1}^{(\pm1)},u_{2k}^{(\pm1)}\}\mapsto\{q_k^{(\pm1)},e_k^{(\pm1)}\}$

【０２１３】

この場合のミウラ変換は、次のように示される。

【数６８】

$$t_{2k-1}=\frac{1}{\delta^{(0)}}(1+\delta^{(0)}u_{2k-2}^{(0)})(1+\delta^{(0)}u_{2k-1}^{(0)})+\delta^{(0)}u_{2k-3}^{(0)}u_{2k-2}^{(0)}\qquad k=1,2,...,m$$

$$(t_{2k})^2=(1+\delta^{(0)}u_{2k-2}^{(0)})(1+\delta^{(0)}u_{2k-1}^{(0)})u_{2k-1}^{(0)}u_{2k}^{(0)}\qquad k=1,2,...,m-1$$

$$u_0^{(0)}\equiv 0,$$

ただし、$\delta^{(0)}$は任意である。

このように、｛ｔ$_{2k-1}$，ｔ$_{2k}$｝から｛ｕ$_{2k-1}^{(0)}$，ｕ$_{2k}^{(0)}$｝を求める場合には、｛ｔ$_{2k-1}$，ｔ$_{2k}$｝から｛ｑ$_k^{(0)}$，ｅ$_k^{(0)}$｝を求め、さらにｑ$_k^{(0)}$，ｅ$_k^{(0)}$｝から｛ｕ$_{2k-1}^{(0)}$，ｕ$_{2k}^{(0)}$｝を求める場合に比べて、計算量が少なくなることとなり、計算速度を向上させることも可能である。

【０２１４】

なお、この場合には、対角行列Ｔの各要素について直接ＬＶ型変換が行われることになる。したがって、この変換が、「対角行列Ｔの各要素に関してＬＶ型変換を行う」場合に含まれることは言うまでもない。

【０２１５】

ｑｄ型ツイスト分解法あるいはＬＶ型ツイスト分解法によってコレスキー分解をすることができると、上述のようにツイストされた行列Ｎ$_k$を算出することができ、その行列Ｎ$_k$のＮ$_k^{T}$ｘ$_j$＝ｅ$_k$を解くことによって、ｘ$_j$を算出することができる。ここで、

【数６９】

$$\mathbf{x}_j \leftarrow \frac{\mathbf{x}_j}{\left\|\mathbf{x}_j\right\|}$$

とｘ$_j$を置き換えることにより、このｘ$_j$を正規化する。このようにして、固有ベクトルｘ$_j$を求めることができる。この固有ベクトルｘ$_j$を用いて、Ｑ$_T$＝（ｘ$_1$，ｘ$_2$，...，ｘ$_m$）とすることにより、直交行列Ｑ$_T$を算出することができる。

【０２１６】

したがって、次式のように対称３重対角行列Ｔについて固有値分解が行われたことになる。

【数７０】

$$T = Q_T D_T Q_T^{\mathrm{T}}$$

【０２１７】

また、対称３重対角行列Ｔの固有値分解を行うことができれば、次のように、対称行列Ａの固有値分解も行われる。

【数７１】

$$\begin{aligned} A &= PTP^{\mathrm{T}} \\ &= P(Q_T D_T Q_T^{\mathrm{T}})P^{\mathrm{T}} \\ &= (PQ_T)D_T(PQ_T)^{\mathrm{T}} \\ &= Q_A D_T Q_A^{\mathrm{T}} \end{aligned}$$

【０２１８】

固有ベクトル算出部１９は、上述のようにして、固有値記憶部１８が記憶している対称３重対角行列Ｔの固有値と、対角行列記憶部１３が記憶している対称３重対角行列Ｔとを用いて、対称３重対角行列Ｔの固有ベクトルを算出する。まず、コレスキー分解部３１は、対角行列記憶部１３から対称３重対角行列Ｔを読み出し、固有値記憶部１８から対称３重対角行列Ｔの固有値を読み出す。そして、コレスキー分解部３１は、図２３のフローチャートで示されるように各変換を行い、コレスキー分解の処理を行う。ここでは、ＬＶ型ツイスト分解法を用いる場合について説明する。

【０２１９】

コレスキー分解部３１は、まず、対称３重対角行列Ｔの各要素の値から、ｑ$_k^{(0)}$，ｅ$_k^{(0)}$を求める。そして、コレスキー分解部３１は、ミウラ変換を実行することにより、ｕ$_1^{(0)}$，ｕ$_2^{(0)}$等を順次求めていく（ステップＳ１５０１）。

【０２２０】

次に、コレスキー分解部３１は、ミウラ変換で得られたｕ$_k^{(0)}$を用いて、ｓｔｄＬＶｖ変換を実行することにより、ｕ$_1^{(+1)}$等を順次求めていく（ステップＳ１５０２）。また、コレスキー分解部３１は、ミウラ変換で得られたｕ$_k^{(0)}$を用いて、ｒｄＬＶｖ変換を実行することにより、ｕ$_{2m-1}^{(-1)}$等を順次求めていく（ステップＳ１５０３）。

【０２２１】

最後に、コレスキー分解部３１は、ｓｔｄＬＶｖ変換で得られたｕ$_k^{(+1)}$及びｒｄＬＶｖ変換で得られたｕ$_k^{(-1)}$を用いて、逆ミウラ変換を実行することにより、ｑ$_1^{(\pm 1)}$、ｅ$_1^{(\pm 1)}$等を順次求めていく（ステップＳ１５０４）。ｑ$_k^{(\pm 1)}$及びｅ$_k^{(\pm 1)}$が求まると、すなわち、上２重対角行列Ｂ$^{(+1)}$と、下２重対角行列Ｂ$^{(-1)}$が求まると、コレスキー分解部３１は、それらをベクトル算出部３２に渡す。なお、コレスキー分解部３１は、各固有値について、それぞれコレスキー分解を行うものとする。

【０２２２】

なお、ここでは、説明の便宜上、図２３のフローチャートを用いてコレスキー分解の処理を説明したが、図１６～図２１のフローチャートで示されるように処理を行ってもよいことは言うまでもない。

【０２２３】

ベクトル算出部３２は、固有値記憶部１８から固有値を読み出し、式５を用いてｋの値を決定する。ベクトル算出部３２は、そのｋの値を用いて、$q_k^{(\pm 1)}$及び$e_k^{(\pm 1)}$からツイスト行列N_kを求め、$N_k^T x_j = e_k$を解くことによって、x_jを算出する（ステップＳ８０２）。

【０２２４】

次に、ベクトル算出部３２は、算出したx_jを正規化して、固有ベクトルx_jを求めて固有ベクトル記憶部２０に蓄積する（ステップＳ８０３）。なお、ベクトル算出部３２は、各固有値について、x_jを算出する処理と、正規化する処理とを行うことによって、全ての固有ベクトルを算出することができる。このようにして、固有ベクトルの算出が終了する。

この後、行列記憶部１１が記憶している対称行列Ａの固有ベクトルを算出してもよい。その算出の方法は、前述の通りである。

【０２２５】

また、固有値算出部１７が算出した固有値、固有ベクトル算出部１９が算出した固有ベクトルを出力する図示しない出力部を固有値分解装置１が備えてもよい。ここで、図示しない出力部による出力は、例えば、固有値等の表示デバイス（例えば、ＣＲＴや液晶ディスプレイなど）への表示でもよく、固有値等の所定の機器への通信回線を介した送信でもよく、固有値等のプリンタによる印刷でもよく、固有値等の記録媒体への蓄積でもよい。なお、その出力部は、出力を行うデバイス（例えば、表示デバイスやプリンタなど）を含んでもよく、あるいは含まなくてもよい。また、その出力部は、ハードウェアによって実現されてもよく、あるいは、それらのデバイスを駆動するドライバ等のソフトウェアによって実現されてもよい。

【０２２６】

なお、ここでは、コレスキー分解部３１がＬＶ型ツイスト分解法によってコレスキー分解を行う場合について説明したが、コレスキー分解部３１は、ｑｄ型ツイスト分解法によってコレスキー分解を行ってもよい。例えば、コレスキー分解部３１は、算出された固有値の分布を見て、分布が密でない場合には、ｑｄ型ツイスト分解法を用いるようにしてもよい。

【０２２７】

また、上記説明では、行列$B^{(0)}$が上２重対角行列である場合について説明したが、行列$B^{(0)}$は下２重対角行列であっても固有値分解を同様に実行することができうる。ただし、行列Ｔの各要素と、行列$B^{(0)}$の各要素との対応を示す式が上述のものと異なることになる。

【０２２８】

また、上記説明では、固有値算出部１７が全ての固有値を算出する場合について説明したが、一部の固有値のみを算出するようにしてもよい。例えば、図９において、固有値算出部１７は、行列T_1，T_2までは、全ての固有値を算出する必要があるが、行列T_1，T_2から対称３重対角行列Ｔの固有値を算出する際に、必要な範囲で固有値を算出してもよい。したがって、固有値算出部１７は、対称３重対角行列Ｔの少なくとも１個の固有値を算出するものであってもよい。その場合には、不必要な固有値を算出することに伴う余分な処理を実行しなくてよいため、処理負荷を軽減することができうる。このように、一部の固有値のみを算出する場合には、例えば、対称３重対角行列Ｔの固有値を算出する処理に費やす計算時間を約（１＋ｋ／ｍ）／２倍にすることができうる。ここで、ｍは行列サイズであり、ｋは求める固有値の個数である。

【０２２９】

また、上記説明では、固有ベクトル算出部１９が算出された固有値に対応する全ての固

有ベクトルを算出する場合について説明したが、上記説明から明らかなように、固有ベクトルを算出する処理は、固有値ごとに行うことができる。したがって、固有ベクトル算出部１９は、対称３重対角行列Ｔの少なくとも１個の固有ベクトルを算出するものであってもよい。このように、固有ベクトル算出部１９は、必要な範囲で固有ベクトルを算出することができ、不必要な固有ベクトルを算出することに伴う余分な処理を実行しなくてよいため、処理負荷を軽減することができうる。

【０２３０】

次に、本実施の形態による固有値分解装置１における並列処理について説明する。

本実施の形態による固有値分解装置１では、固有値の計算及び固有ベクトルの計算において、並列的に処理を実行することができる。例えば、図２４～図２６で示されるように、固有値分解装置１において、固有値分解部１５、固有値算出部１７、コレスキー分解部２２、ベクトル算出部２３，コレスキー分解部３１、ベクトル算出部３２において、それぞれ並列処理を行ってもよい。

【０２３１】

固有値分解部１５は、複数の固有値分解手段１５ａ、１５ｂを備え、その複数の固有値分解手段１５ａ、１５ｂが、分割後の対称３重対角行列の固有値と、行列要素とを算出する処理を並列実行してもよい。例えば、図９において、固有値分解手段１５ａが行列T_{111}，T_{112}，T_{121}，T_{122}の固有値分解を実行し、固有値分解手段１５ｂが行列T_{211}，T_{212}，T_{221}，T_{222}の固有値分解を実行してもよい。

【０２３２】

固有値算出部１７は、複数の固有値算出手段１７ａ、１７ｂを備え、その複数の固有値算出手段１７ａ、１７ｂが、分割元の対称３重対角行列の固有値と、行列要素とを算出する処理を並列実行してもよい。例えば、図９において、固有値算出手段１７ａが行列T_{11}，T_{12}，T_{1}，Ｔの固有値等を算出する処理を実行し、固有値算出手段１７ｂが行列T_{21}，T_{22}，T_{2}の固有値等を算出する処理を実行してもよい。

【０２３３】

コレスキー分解部２２は、複数のコレスキー分解手段２２ａ、２２ｂを備え、その複数のコレスキー分解手段２２ａ、２２ｂが、対称３重対角行列Ｔに関してコレスキー分解する処理を並列実行してもよい。例えば、コレスキー分解手段２２ａが半分の固有値についてコレスキー分解する処理を実行し、コレスキー分解手段２２ｂが残りの半分の固有値についてコレスキー分解する処理を実行してもよい。

【０２３４】

ベクトル算出部２３は、複数のベクトル算出手段２３ａ、２３ｂを備え、その複数のベクトル算出手段２３ａ、２３ｂが、固有ベクトルを算出する処理を並列実行してもよい。例えば、ベクトル算出手段２３ａがコレスキー分解手段２２ａによってコレスキー分解された固有値に関して固有ベクトルを算出する処理を実行し、ベクトル算出手段２３ｂがコレスキー分解手段２２ｂによってコレスキー分解された固有値に関して固有ベクトルを算出する処理を実行してもよい。

【０２３５】

コレスキー分解部３１が複数のコレスキー分解手段３１ａ、３１ｂを備えて、対称３重対角行列Ｔに関してコレスキー分解する処理を並列実行してもよいことは、コレスキー分解部２２と同様である。

【０２３６】

ベクトル算出部３２が複数のベクトル算出手段３２ａ、３２ｂを備えて、固有ベクトルを算出する処理を並列実行してもよいことは、ベクトル算出部２３と同様である。

【０２３７】

なお、固有値算出部１７の複数の固有値算出手段１７ａ、１７ｂは、図６で示される分割された２個の行列T_{1}，T_{2}の固有値と行列要素とから、分割元の行列T_{0}の固有値と行列要素とを算出する処理を、並列実行してもよい。以下、その処理について説明する。

【０２３８】

まず、式１から、各固有値を求める処理については、並列実行可能なことがわかる。例えば、固有値算出手段１７ａが半分の固有値を算出し、固有値算出手段１７ｂが残りの半分の固有値を算出してもよい。なお、固有値算出手段１７ａ、１７ｂのそれぞれは、行列D_{12}と、ベクトルｚとの各成分の値を有するものとする。

【０２３９】

また、式２から、各固有ベクトルq_iを求める処理についても、並列実行可能なことがわかる。例えば、固有値算出手段１７ａは、算出した固有値に対応する半分の固有ベクトルq_iを算出し、固有値算出手段１７ｂは、算出した固有値に対応する残りの半分の固有ベクトルq_iを算出してもよい。

【０２４０】

また、式３，式４から、ｆ、ｌを求める処理についても、並列実行可能なことがわかる。例えば、固有値算出手段１７ａは、算出した固有ベクトルq_iを用いて、ｆ、ｌの一部の要素を算出し、固有値算出手段１７ｂは、算出した固有ベクトルq_iを用いて、ｆ、ｌの残りの要素を算出してもよい。その後、固有値算出手段１７ａ、あるいは固有値算出手段１７ｂが、固有値算出手段１７ａ、１７ｂの算出したｆ、ｌの各要素を統合することによって、式３，式４で示されるｆ、ｌを算出することができる。

【０２４１】

ここでは、ｚの成分に０が含まれず、D_{12}の対角成分に同じ値が含まれない場合における固有値算出手段１７ａ、１７ｂの処理について説明したが、ｚの成分に０が含まれる場合、D_{12}の対角成分に同じ値が含まれる場合であっても、固有方程式を解く処理、固有ベクトルq_iを算出する処理、固有ベクトルから分割元の行列の行列要素を算出する処理を同様に並列実行することができうる。

【０２４２】

上記のような各構成要素における並列処理において、各手段が固有値等の情報を格納するメモリを用いる場合に、複数の手段が同一のメモリ、すなわち共有メモリを使用してもよく、あるいは、各手段がそれぞれ別のメモリを使用してもよい。

【０２４３】

なお、図２４～図２６を用いて、固有値分解部１５や固有値算出部１７等が２個の手段によって並列処理を実行する場合について説明したが、固有値分解部１５や固有値算出部１７等が３以上の手段を備え、並列処理を実行してもよい。また、固有値分解部１５、固有値算出部１７、コレスキー分解部２２、ベクトル算出部２３，コレスキー分解部３１、ベクトル算出部３２のそれぞれにおいて並列処理が実行される場合について説明したが、いずれかの任意の１以上の部において、並列処理が実行されなくてもよい。例えば、固有値分解部１５において並列処理が行われなくてもよい。

【０２４４】

また、その並列処理は、１の装置において、２以上のＣＰＵ等を用いて実行されてもよく、２以上の装置において実行されてもよい。例えば、図２７で示されるように、装置Ａと、装置Ｂとがそれぞれ固有値分解手段１５ａ、１５ｂを備え、各装置において、固有値分解の処理が並列実行されてもよい。この場合には、装置Ａの固有値分解手段１５ａを備える固有値分解部１５－１と、装置Ｂの固有値分解手段１５ｂを備える固有値分解部１５－２とによって固有値分解部１５が構成されることになる。したがって、固有値分解装置１は、装置Ａと、装置Ｂとからなるシステムを構成することになる。ここでは、固有値分解部１５の並列処理について説明したが、その他の固有値算出部１７やコレスキー分解部２２等についても、２以上の装置による並列処理を行ってもよい。

【０２４５】

次に、数値実験による性能の評価について説明する。ここでは、本実施の形態による固有値分解装置１の方法（ＤＤＣ）と、標準的な分割統治法（ＤＣ）と、ＱＲ法（ＱＲ）と、二分法と逆反復法とを組み合わせた方法（Ｂ&Ｉ）とを比較した。なお、本実施の形態による固有値分解装置１の方法では、コレスキー分解をｑｄ型ツイスト分解法で行い、逆反復（図１２のステップＳ７０１参照）を１回用いた。分割統治法としては、ＬＡＰＡＣ

Ｋで提供されているＤＳＴＥＤＣを用いた。ＱＲ法としては、ＬＡＰＡＣＫで提供されているＤＳＴＥＱＲを用いた。二分法と逆反復法とを組み合わせた方法としては、ＬＡＰＡＣＫで提供されているＤＳＴＥＶＲのソースコードを一部変更したものを用いた。実験で用いるＬＡＰＡＣＫはＢＬＡＳ（Ｂａｓｉｃ　Ｌｉｎｅａｒ　Ａｌｇｅｂｒａ　Ｓｕｂｐｒｏｇｒａｍｓ）を呼び出す。また、本実施の形態による固有値分解装置１の固有値分解部１５では、ＱＲ法を用いて固有値分解を行った。また、この数値実験では、ランダムに与えた対角行列を、ランチョス法を用いて対称３重対角化した行列を用いた。数値実験結果のグラフにおいて、横軸は、数値実験で用いた行列の大きさである。

【０２４６】

図２８は、計算の実行時間を示す図である。実行時間が短いほど、高速に計算できることになる。図２８の数値実験結果のグラフからわかるように、本実施の形態による固有値分解装置１の方法では、他の方法に比べて高速に固有値分解を行うことができることがわかる。

【０２４７】

図２９は、算出された固有ベクトルの直交性を示す図である。値の小さい方が、直交性が高いことになる。図２９の数値実験結果のグラフからわかるように、本実施の形態による固有値分解装置１の方法では、他の方法に比べて、固有ベクトルの直交性は高くない。本実施の形態による固有値分解装置１の方法では、二分法と逆反復法とを組み合わせた方法と同程度の直交性を有することがわかる。これは、固有値をまず算出し、その後に、固有ベクトルを算出していることに起因していると考えられる。

【０２４８】

図３０は、全固有値の相対誤差の総和を示す図である。値の小さい方が、誤差が小さく、精度が高いことになる。図３０の数値実験結果のグラフからわかるように、本実施の形態による固有値分解装置１の方法では、他の方法に比べて固有値の精度が高いことがわかる。分割統治法により算出した固有値と同程度の固有値の精度を有する。本実施の形態による固有値分解装置１の方法では、分割統治法によって固有値を算出しているため、当然の結果であると考えられる。

【０２４９】

図３１は、前固有ベクトルの誤差の総和を示す図である。値の小さい方が、誤差が小さく、精度が高いことになる。図３１の数値実験結果のグラフからわかるように、本実施の形態による固有値分解装置１の方法では、他の方法に比べて固有ベクトルの精度が高いことがわかる。

【０２５０】

このように、本実施の形態による固有値分解装置１の方法では、高速に固有値分解をすることができ、さらに、固有値、固有ベクトルの精度の高いことがわかる。また、前述のように、並列性にも優れているため、並列処理を行うことによってさらに高速化を図ることも可能である。

【０２５１】

［主成分分析による顔画像認識］

次に、主成分分析による顔画像認識に固有値分解を応用する場合について説明する。

【０２５２】

主成分分析による顔画像認識の処理は、以下の各ステップによって行われる。

（１）射影行列の作成

（２）個人の顔画像に関する情報の登録

（３）入力された顔画像と、登録されている個人の顔画像に関する情報とを比較することによる認識

【０２５３】

まず、上記（１）の射影行列の作成について説明する。

種々の人物の顔画像データを用意する。その顔画像データは、例えば、スキャナやデジタルスチルカメラ、デジタルビデオカメラ等の光学的読み取り機器によって読み取られた

２次元画像データであってもよい。各顔画像データは、各画素値を成分とするベクトルで表示することができる。そのベクトルを、次のようにする。ここでは、Ｍ個の顔画像データを扱うものとする。

【数７２】

$$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M$$

【０２５４】

次に、その顔画像データのベクトルの平均値μを算出する。

【数７３】

$$\mu = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i$$

【０２５５】

その算出したμを用いて、共分散行列Ｃを次のように算出する。

【数７４】

$$C = \frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{M}(\mu - \mathbf{x}_i)(\mu - \mathbf{x}_j)^{\mathrm{T}}$$

【０２５６】

その後、上述の固有値分解装置１によって、その行列Ｃに対して、固有値分解を行う。すなわち、行列記憶部１１に行列Ｃを蓄積し、その行列Ｃに対して対角化や、行列の分割等の処理を行うことによって、固有値分解を行う。その固有値分解された結果である固有値λ_jは、固有値記憶部１８に蓄積されることになる。また、固有値分解された結果である固有ベクトルv_jは、固有ベクトル記憶部２０で記憶されている固有ベクトルに対して、対角化で用いた直交行列を作用させることによって、前述のようにして求めることができる。なお、添字ｊは、固有値の降順になるように設定されているものとする。すなわち、固有値λ_1が最大の固有値となる。

【０２５７】

その求めた固有ベクトルのうち、対応する固有値が上位からｄ個の固有ベクトルを用いて、射影行列ηを次のように算出する。

【数７５】

$$\eta = (\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_d)$$

【０２５８】

次に、上記（２）の個人の顔画像に関する情報の登録について説明する。

各個人について、ｍ個の顔画像データ

【数７６】

$$\mathbf{x}_1^k, \mathbf{x}_2^k, \cdots, \mathbf{x}_m^k$$

を用意する。ここで、添字ｋは、個人を識別するための添字である。そのベクトルに射影行列ηをそれぞれ掛けることにより、次式のようにしてｍ個のｄ次元ベクトルξ_iを算出し、その平均値μ^kを算出する。そして、その平均値μ^kを登録しておく。

【数７７】

$$\xi_i^k = \eta^{\mathrm{T}}\mathbf{x}_i^k \qquad i = 1,2,\cdots,m$$

$$\mu^k = \frac{1}{m}\sum_{i=1}^{m}\xi_i^k$$

【０２５９】

次に、上記（３）の入力された顔画像と、登録されている個人の顔画像に関する情報とを比較することによる認識について説明する。

ある人物の顔画像データのベクトルｘが入力されたとする。すると、その顔画像データのベクトルｘに射影行列ηを掛けることにより、ｄ次元ベクトルξを算出する。

【数７８】

$$\xi = \eta^{\mathrm{T}}\mathbf{x}$$

【０２６０】

次に、そのｄ次元ベクトルξと、各個人のｄ次元ベクトルξ_iの平均値μ^kとの差のノルムを計算する。

【数７９】

$$\left\| \mu^k - \xi \right\|$$

【０２６１】

そのノルムがあらかじめ設定されているしきい値より小さい場合に、入力された顔画像が、添字ｋで識別される個人の顔画像であると判断してもよい。あるいは、すべての添字ｋについて上記のノルムを計算し、入力された顔画像が、最小のノルムに対応する添字ｋで識別される個人の顔画像であると判断してもよい。このようにして、固有値分解装置１を用いて、主成分分析による顔画像認識を行うことができうる。

【０２６２】

［２次元画像から３次元へ復元する画像処理への応用］

次に、物体の２次元画像から３次元へ復元する画像処理に固有値分解を応用する場合について説明する。

【０２６３】

複数の２次元画像から３次元復元を行う処理は、以下の各ステップによって行われる。

（１）２次元画像から特徴点を抽出するステップ。

（２）特徴点データより形状（元の物体の特徴点の３次元座標データ）及び回転（３次元データから特徴点データへの変換）に関するデータを計算するステップ。

（３）形状及び回転のデータより可視化を行うステップ。

【０２６４】

以下、上記（１）、（２）の各ステップについて、図３２のフローチャートを用いて説明する。ここで、２次元画像は、例えば、スキャナやデジタルスチルカメラ、デジタルビデオカメラ等の光学的読み取り機器によって読み取られた２次元画像であってもよい。

【０２６５】

ステップＳ５００１において、２次元画像ｊ（ｊ＝１，・・・，ｍ、ｍは３以上の整数）から特徴点ｉ（ｉ＝１，・・・，ｎ、ｎは２以上の整数）の座標（x_i^j，y_i^j）を抽出する。取り扱う２次元画像は、弱中心射影画像であることが好ましい。このとき、次の式が成り立つ。

【数８０】

$$\begin{pmatrix} x_i^j \\ y_i^j \end{pmatrix} = s_j \begin{pmatrix} \mathbf{r}_{1,j}^{\mathrm{T}} \\ \mathbf{r}_{2,j}^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix}$$

【０２６６】

ここで、s_jは物体のスケールに相対するｊ番目の画像のスケール、$r_{1,j}$，$r_{2,j}$はそれぞれ物体座標系に相対するｊ番目のカメラ座標系の回転行列の１番目と２番目の行ベクトル、$(Xi, Yi, Zi)^{\mathrm{T}}$はｉ番目の点の３次元座標である。物体のスケールは１番目の画像のスケールと同じにし（$s_1 = 1$）、物体の座標系の姿勢は１番目の画像のカメラ座標系と同じにする（$r_{1,1} = (1, 0, 0)^{\mathrm{T}}$，$r_{2,1} = (0, 1, 0)^{\mathrm{T}}$）。ｍ枚全ての画像におけるｎ個全ての点の座標を行列Ｄの要素として並べると、次式のようになる。

【０２６７】

【数８１】

$$D = MS$$

ただし、$D = \begin{pmatrix} x_1^1 & \cdots & x_i^1 & \cdots & x_n^1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_1^j & \cdots & x_i^j & \cdots & x_n^j \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_1^m & \cdots & x_i^m & \cdots & x_n^m \\ y_1^1 & \cdots & y_i^1 & \cdots & y_n^1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ y_1^j & \cdots & y_i^j & \cdots & y_n^j \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ y_1^m & \cdots & y_i^m & \cdots & y_n^m \end{pmatrix}$, $M = \begin{pmatrix} s_1 \mathbf{r}_{1,1}^{\mathrm{T}} \\ \cdots \\ s_j \mathbf{r}_{1,j}^{\mathrm{T}} \\ \cdots \\ s_m \mathbf{r}_{1,m}^{\mathrm{T}} \\ s_1 \mathbf{r}_{2,1}^{\mathrm{T}} \\ \cdots \\ s_j \mathbf{r}_{2,j}^{\mathrm{T}} \\ \cdots \\ s_m \mathbf{r}_{2,m}^{\mathrm{T}} \end{pmatrix}$,

$S = \begin{pmatrix} X_1 & \cdots & X_i & \cdots & X_n \\ Y_1 & \cdots & Y_i & \cdots & Y_n \\ Z_1 & \cdots & Z_i & \cdots & Z_n \end{pmatrix}$

【０２６８】

ＭとＳの形から分かるように、Ｄのランクは３である。ここで、ステップＳ５００１において、行列Ｄが与えられている。以下、回転に関するデータＭ及び形状Ｓを求める。

【０２６９】

そこで、行列Ｄの特異値分解

Ｄ＝ＵΣＶ^{T}

を考える。ここで、Σは特異値を大小順に対角線上に並べたもので、Ｕ及びＶはそれぞれ左直交行列、及び右直交行列である。この特異値分解において、前述の固有値分解を用いることができる。すなわち、固有値分解装置１において、行列記憶部１１が記憶している対称行列Ａを、Ｄ^{T}Ｄとすることにより、前述の説明のようにして、対称行列Ａの固有値分解がなされることになる。なお、固有値分解装置１において、固有ベクトル記憶部２０に蓄積されるのは、対称行列Ｄ^{T}Ｄを変換した対称３重対角行列Ｔの固有ベクトルであるので、その固有ベクトルを、前述の説明のようにして、行列Ｄ^{T}Ｄの固有ベクトルに変換する必要がある。また、その行列Ｄ^{T}Ｄの各固有値の１／２乗（平方根）が各特異値となる。すなわち、特異値σ$_{j}$＝（λ$_{j}$）$^{1/2}$となる。また、固有ベクトルと、右直交行列Ｖの各列を構成する右特異ベクトルｘ$_{j}$とは等しいため、固有ベクトルを各列に有する行列が、右直交行列Ｖとなる。なお、特異値σ$_{j}$が０でない場合には、左直交行列Ｕの各列を構成する左特異ベクトルｙ$_{j}$を次のようにして求めることができる。

【０２７０】

【数８２】

$$\mathbf{y}_j = \frac{D\ \mathbf{x}_j}{\sigma_j}$$

となる。一方、特異値σ$_{j}$が０である場合には、

【数８３】

$$D^{\mathrm{T}}\mathbf{y}_j = 0$$

を解くことにより、ｙ$_{j}$を算出することができる。ここで、次の置き換えを行うことによって、左特異ベクトルｙ$_{j}$を正規化する。

【数８４】

$$\mathbf{y}_j \leftarrow \frac{\mathbf{y}_j}{\|\mathbf{y}_j\|}$$

【０２７１】

このようにして、左特異ベクトルｙ$_j$を求めることができ、この左特異ベクトルｙ$_j$を用いて、Ｕ＝（ｙ$_1$，ｙ$_2$，…，ｙ$_m$）とすることにより、左直交行列Ｕを算出することができる。

【０２７２】

ここで、画像のデジタル誤差のため、ゼロでない特異値は３つ以上出てくる。しかし、４番目以降の特異値はノイズによるもので、最初の３つの特異値と比べて格段に小さい。そこで、ステップＳ５００２では、最初の３つの特異値に対して特異ベクトルを計算する。すなわち、本実施の形態による固有値分解装置１では、最初の３個の固有値に対して固有ベクトルを計算する。採用する３個のベクトルをまとめると、次式となる。

【０２７３】

Ｄ'＝Ｌ'Σ'Ｒ'^{T}＝Ｍ'Ｓ'

ただし、Ｍ'＝Ｌ'（Σ'）$^{1/2}$、Ｓ'＝（Σ'）$^{1/2}$Ｒ'^{T}、Ｄ'は‖Ｄ－Ｄ'‖を最小にするランク３の行列である。

【０２７４】

次に、Ｄ'からＭ及びＳを求めたいが、その組合せは唯一ではない。なぜなら、任意の正則行列Ｃが

Ｄ'＝（Ｍ'Ｃ）（Ｃ$^{-1}$Ｓ'）

を満たすからである。そこで、上式におけるＣをＭ＝Ｍ'Ｃを満たすように決める。Ｃは下記の式を満たす。

【数８５】

$$MM^{T} = M'CC^{T}M'^{T} = \begin{pmatrix} 1 & \cdots & \cdots & 0 & \cdots & \cdots \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & s_m^2 & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & 1 & \cdots & \cdots \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & 0 & \cdots & \cdots & s_m^2 \end{pmatrix}$$

【０２７５】

Ｅ＝ＣＣ^{T}とすると、上式からＥの６つの要素に関する２ｍ＋１個の線形方程式が得られる。ｍ≧３であるので、Ｅの要素を一意に決めることができる。ステップＳ５００３において、行列Ｅを求める。

【０２７６】

次に、ステップＳ５００４において、ＥからＣを求める。Ｃの自由度（９）はＥの自由度（６）より多い。そこで、条件ｒ$_{1j}$＝（１，０，０）^{T}，ｒ$_{2j}$＝（０，１，０）^{T}を加えれば、Ｃを決めることができる。このとき２つの解（Ｎｅｃｋｅｒ　Ｒｅｖｅｒｓａｌ）が出る。

次に、ステップＳ５００５において、Ｍ＝Ｍ'Ｃ及びＳ＝Ｃ$^{-1}$Ｓ'より、回転に関するデータＭ及び形状Ｓが決まる。

【０２７７】

［文書検索への応用］

次に、文書検索処理に固有値分解を応用する場合について説明する。文書中からその文書の内容に関連する索引語を抽出し、索引語の重みを計算する処理の後、ベクトル空間モデルでは、この索引語の重みを要素とするベクトルで文書を表現する。ここで、検索対象となる文書をｄ$_1$，ｄ$_2$，・・・，ｄ$_n$とし、これら文書集合全体を通して全部でｍ個の

索引語w_1，w_2，・・・，w_mがあるとする。このとき、文書d_jは、次のようなベクトルで表現されることになる。これを文書ベクトルと呼ぶ。

【数８６】

$$\mathbf{d}_j = \begin{pmatrix} d_{1j} \\ d_{2j} \\ \vdots \\ d_{mj} \end{pmatrix}$$

【０２７８】

ここで、d_{ij}は索引語w_iの文書d_jにおける重みである。また、文書集合全体は、次のようなm×n行列Dによって表現することができる。

【数８７】

$$D = \begin{pmatrix} \mathbf{d}_1 & \mathbf{d}_2 & \cdots & \mathbf{d}_n \end{pmatrix} = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{pmatrix}$$

【０２７９】

行列Dを索引語文書行列と呼ぶ。索引語文書行列の各列は文書に関する情報を表している文書ベクトルであるが、同様に、索引語文書行列の各行は索引語に関する情報を表しているベクトルであり、これを索引語ベクトルと呼ぶ。検索質問も、文書と同様に、索引語の重みを要素とするベクトルで表現することができる。検索質問文に含まれる索引語w_iの重みをq_iとすると、検索質問ベクトルqは次のように表されることになる。

【数８８】

$$\mathbf{q} = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{pmatrix}$$

【０２８０】

実際の文書検索においては、与えられた検索質問文と類似した文書を見つけ出す必要があるが、検索質問ベクトルqと各文書ベクトルd_jの間の類似度を計算することにより行う。ベクトル間の類似度の定義としてはさまざまなものが考えられるが、文書検索においてよく用いられているものはコサイン尺度（２つのベクトルのなす角度）または内積である。

【０２８１】

【数８９】

$$\cos(\mathbf{d}_j, \mathbf{q}) = \frac{\mathbf{d}_j \cdot \mathbf{q}}{\|\mathbf{d}_j\| \|\mathbf{q}\|} = \frac{\sum_{i=1}^{m} d_{ij} q_i}{\sqrt{\sum_{i=1}^{m} d_{ij}^2} \sqrt{\sum_{i=1}^{m} q_i^2}} \qquad \text{（コサイン尺度）}$$

$$\mathbf{d}_j \cdot \mathbf{q} = \sum_{i=1}^{m} d_{ij} q_i \qquad \text{（内積）}$$

なお、ベクトルの長さが１に正規化（コサイン正規化）されている場合には、コサイン尺度と内積とは一致する。

【０２８２】

図３３は、本実施の形態による固有値分解装置１を利用した文書検索方法の一例を示すフローチャートである。

ステップＳ６００１において、質問ベクトルｑを受け取る。

【０２８３】

ここでは、Ｄの近似行列D_kを使った検索を考える。ベクトル空間モデルでは、検索質問ベクトルｑと索引語文書行列Ｄ中の各文書ベクトルd_jの間の類似度を計算することにより検索を行うが、ここではＤの代わりにD_kを使う。ベクトル空間モデルでは、文書ベクトルの次元数は索引語の総数と等しい。したがって、検索対象となる文書の数が増えるに従い、文書ベクトルの次元数も増加する傾向にある。しかし、次元数が増加してくると、コンピュータのメモリによる制限や検索時間の増大などの問題が生じてくるばかりでなく、文書中に含まれる不必要な索引語がノイズ的な影響を及ぼし、検索精度を低下させてしまうという現象も起こってくる。潜在的意味インデキシング（ｌａｔｅｎｔ　ｓｅｍａｎｔｉｃ　ｉｎｄｅｘｉｎｇ；ＬＳＩ）は、高次元の空間にある文書ベクトルを低次元の空間へと射影することにより、検索精度の改善を図る技術である。高次元の空間では別々に扱われていた索引語が、低次元の空間では相互に関連を持ったものとして扱われる可能性もあるため、索引語の持つ意味や概念に基づく検索を行うことができる。たとえば、通常のベクトル空間モデルでは"ｃａｒ"という索引語と"ａｕｔｏｍｏｂｉｌｅ"という索引語はまったく別物であり、一方の索引語による質問ではもう片方の索引語を含んだ文書を検索することができない。しかし、低次元の空間ではこれらの意味的に関連した索引語は１つの次元に縮退することが期待できるため、"ｃａｒ"という検索質問によって"ｃａｒ"を含む文書ばかりでなく"ａｕｔｏｍｏｂｉｌｅ"を含む文書をも検索することが可能となる。潜在的意味インデキシングでは、特異値分解により高次元ベクトルの次元削減を行うが、これは基本的に多変量解析における主成分分析と等価である。

【０２８４】

ステップＳ６００２において、ｋを選択する。ここでは、$k<r$なるｋの値を選択する。$r=min(n, m)$である。ｋの値は、予め与えられていてもよいし、計算ごとに選択可能であってもよい。

【０２８５】

次に、ステップＳ６００３では、行列Ｄの特異値分解を行う。この特異値分解として、前述の２次元画像から３次元へ復元する画像処理への応用で説明したように、固有値分解による方法を用いることができる。すなわち、固有値分解装置１において、行列記憶部１１が記憶している対称行列Ａを、このたびの行列$D^T D$とすることにより、前述の説明のようにして、対称行列Ａの固有値分解がなされ、その結果として、行列Ｄの特異値分解を行うことができうる。なお、この特異値分解では、計算された特異値のうち、大きい順に１番目からｋ番目までのｋ個の特異値に対してＤの特異ベクトルを算出する。すなわち、固有値分解において、計算された固有値のうち、大きい順に１番目からｋ番目までのｋ個の固有値に対して固有ベクトルを算出し、この固有ベクトルを用いて特異ベクトルを算出すればよい。ｋは、ステップＳ６００２で選択された値である。

【０２８６】

すなわち、

$$D_k = U_k \Sigma V_k^T$$

なるU_k及びV_kを計算する。ここで、U_kは、最初のｋ個の左特異ベクトルのみから構成されるｍ×ｋ行列であり、V_kは、最初のｋ個の右特異ベクトルのみから構成されるｎ×ｋ行列であり、Σは、最初のｋ個の特異値のみから構成されるｋ×ｋ対角行列である。

【０２８７】

次に、ステップ６００４において、行列D_kと質問ベクトルｑとの類似度を計算する。いま、ベクトルe_jをｎ次元の単位ベクトルとすると、D_kのｊ番目の文書ベクトルは$D_k e_j$で表すことができる。文書ベクトル$D_k e_j$と検索質問ベクトルｑとの間の類似度計算は、

【数９０】

$$\cos(D_k\mathbf{e}_j,\mathbf{q})=\frac{(D_k\mathbf{e}_j)\cdot\mathbf{q}}{\|D_k\mathbf{e}_j\|\|\mathbf{q}\|}=\frac{(D_k\mathbf{e}_j)^{\mathrm{T}}\mathbf{q}}{\|D_k\mathbf{e}_j\|\|\mathbf{q}\|}$$
$$=\frac{(U_k\Sigma_kV_k^{\mathrm{T}}\mathbf{e}_j)^{\mathrm{T}}\mathbf{q}}{\|U_k\Sigma_kV_k^{\mathrm{T}}\mathbf{e}_j\|\|\mathbf{q}\|}=\frac{\mathbf{e}_j^{\mathrm{T}}V_k\Sigma_kU_k^{\mathrm{T}}\mathbf{q}}{\|\Sigma_kV_k^{\mathrm{T}}\mathbf{e}_j\|\|\mathbf{q}\|}$$
$$=\frac{(\Sigma_kV_k^{\mathrm{T}}\mathbf{e}_j)^{\mathrm{T}}(U_k^{\mathrm{T}}\mathbf{q})}{\|\Sigma_kV_k^{\mathrm{T}}\mathbf{e}_j\|\|\mathbf{q}\|}$$

としてもよいが、別の定義を用いてもよい。上式では、Ｄ$_k$をＵ$_k$，Σ$_k$，Ｖ$_k$から再構成する必要はなく特異値分解の結果から、直接、類似度を計算できることを示している。上式の中に現われるΣ$_k$Ｖ$_k$^{T}ｅ$_j$は、

Σ$_k$Ｖ$_k$^{T}ｅ$_j$＝Ｕ$_k$^{T}Ｄ$_k$ｅ$_j$

と書き直すことができる。この式の右辺は、近似行列Ｄ$_k$におけるｊ番目の文書ベクトルの基底Ｕ$_k$のもとでの座標（文書のｋ次元表現）を表している。同様に、上式の中のＵ$_k$^{T}ｑは、検索質問ベクトルｑの基底Ｕ$_k$のもとでの座標（検索質問のｋ次元表現）である。

【０２８８】

ステップＳ６００５において、ステップＳ６００４において計算された類似度を基準に、検索結果を出力する。

このように、本実施の形態による固有値分解装置１は、画像処理や検索処理等の種々の処理に対して用いることができうる。なお、上記以外の処理に用いてもよいことは言うまでもない。

【０２８９】

以上のように、本実施の形態による固有値分解装置１では、分割統治法を用いて固有値のみを算出するため、標準的な分割統治法で実行されるベクトル更新が必要なくなり、標準的な分割統治法よりも非常に高速である。また、固有値から固有ベクトルを算出する処理も、高速に処理することができる。さらに、ＱＲ法では固有値を算出する段階の並列化が困難であるが、本実施の形態による固有値分解装置１では、固有値を算出する段階、及び固有値から固有ベクトルを算出する段階のそれぞれにおいて、本質的に高い並列性を持つ。また、本実施の形態による固有値分解装置１では、他の方法と比べて高い精度の得られることがわかる。

【０２９０】

なお、本実施の形態による固有値分解装置１は、スタンドアロンの装置であってもよく、サーバクライアントシステムにおけるサーバ装置であってもよく、前述のように、複数の装置から構成されるシステムであってもよい。固有値分解装置１がサーバクライアントシステムにおけるサーバ装置である場合には、行列記憶部１１で記憶される対称行列Ａや、固有値記憶部１８で記憶される固有値や固有ベクトル記憶部２０で記憶される固有ベクトル等は、インターネットやイントラネット等の通信回線を介して送受信されてもよい。

【０２９１】

また、本実施の形態による固有値分解装置１における固有ベクトルの算出において、ある固有値を用いた固有ベクトルの算出では、ｑｄ型ツイスト分解法を用い、他の固有値を用いた固有ベクトルの算出では、ＬＶ型ツイスト分解法を用いるようにしてもよい。例えば、値の近接している固有値については、ＬＶ型ツイスト分解法を用いて固有ベクトルの算出を行い、値が近接していない固有値については、ｑｄ型ツイスト分解法を用いて固有ベクトルの算出を行ってもよい。固有値の値が近接しているかどうかは、所定の記録媒体等においてあらかじめ設定されているしきい値と比較することによって判断してもよい。

【０２９２】

また、上記実施の形態において、各処理または各機能は、単一の装置または単一のシス

テムによって集中処理されることによって実現されてもよく、あるいは、複数の装置または複数のシステムによって分散処理されることによって実現されてもよい。

【０２９３】

また、上記実施の形態において、各構成要素は専用のハードウェアにより構成されてもよく、あるいは、ソフトウェアにより実現可能な構成要素については、プログラムを実行することによって実現されてもよい。例えば、ハードディスクや半導体メモリ等の記録媒体に記録されたソフトウェア・プログラムをＣＰＵ等のプログラム実行部が読み出して実行することによって、各構成要素が実現され得る。なお、上記実施の形態における固有値分解装置を実現するソフトウェアは、以下のようなプログラムである。つまり、このプログラムは、コンピュータに、対称３重対角行列Ｔが記憶される対角行列記憶部から前記対称３重対角行列Ｔを読み出し、当該対称３重対角行列Ｔを２個の対称３重対角行列に分割して前記対角行列記憶部に蓄積し、その対称３重対角行列を２個の対称３重対角行列に分割して前記対角行列記憶部に蓄積する処理を、分割後の各対称３重対角行列があらかじめ決められた大きさ以下となるまで繰り返す行列分割ステップと、前記あらかじめ決められた大きさ以下の各対称３重対角行列を前記対角行列記憶部から読み出し、前記各対称３重対角行列に対して固有値分解を行い、前記各対称３重対角行列の固有値と、前記各対称３重対角行列の固有ベクトルからなる直交行列の行列要素とを少なくとも算出し、当該固有値と当該行列要素とを固有値分解記憶部に蓄積する固有値分解ステップと、各対称３重対角行列の固有値と、行列要素とを前記固有値分解記憶部から読み出し、前記固有値と、前記行列要素とから、分割元の対称３重対角行列の固有値と、分割元の対称３重対角行列の行列要素とを算出して前記固有値分解記憶部に蓄積し、その分割元の対称３重対角行列の固有値と、行列要素とを算出する処理を、対称３重対角行列Ｔの少なくとも１個の固有値を算出するまで繰り返し、前記対称３重対角行列Ｔの少なくとも１個の固有値を固有値記憶部に蓄積する固有値算出ステップと、前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、前記固有値記憶部から前記対称３重対角行列Ｔの固有値を読み出し、前記対称３重対角行列Ｔとその固有値とから、ツイスト分解法を用いて前記対称３重対角行列Ｔの少なくとも１個の固有ベクトルを算出して固有ベクトル記憶部に蓄積する固有ベクトル算出ステップと、を実行させるためのものである。

また、このプログラムでは、前記固有ベクトル算出ステップにおいて、ｑｄ型ツイスト分解法により固有ベクトルを算出してもよい。

【０２９４】

また、このプログラムでは、前記固有ベクトル算出ステップが、前記固有値記憶部から前記対称３重対角行列Ｔの固有値を読み出し、当該固有値のいずれかが負の値である場合に、前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、当該対称３重対角行列Ｔの固有ベクトルを変化させることなく、当該対称３重対角行列Ｔのすべての固有値が正の値となるように、当該対称３重対角行列Ｔを正定値化し、正定値化した後の固有値を前記固有値記憶部に蓄積し、正定値化した後の対称３重対角行列Ｔを前記対角行列記憶部に蓄積する正定値化ステップと、前記対角行列記憶部からすべての固有値が正の値である対称３重対角行列Ｔを読み出し、前記固有値記憶部から前記対称３重対角行列Ｔの正の値の固有値を読み出し、前記対称３重対角行列Ｔの各要素に関してｑｄ型変換を行うことによって、前記対称３重対角行列Ｔを上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）にコレスキー分解するコレスキー分解ステップと、前記上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）の各要素と、前記対称３重対角行列Ｔの正の値の固有値とを用いて固有ベクトルを算出して前記固有ベクトル記憶部に蓄積するベクトル算出ステップと、を備えてもよい。

また、このプログラムでは、前記固有ベクトル算出ステップにおいて、ＬＶ型ツイスト分解法により固有ベクトルを算出してもよい。

【０２９５】

また、このプログラムでは、前記固有ベクトル算出ステップが、前記対角行列記憶部から前記対称３重対角行列Ｔを読み出し、前記固有値記憶部から前記対称３重対角行列Ｔの

固有値を読み出し、前記対称３重対角行列Ｔの各要素に関してミウラ変換、ｄＬＶｖ型変換、逆ミウラ変換を行うことによって、前記対称３重対角行列Ｔを上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）にコレスキー分解するコレスキー分解ステップと、前記上２重対角行列Ｂ（＋１）及び下２重対角行列Ｂ（－１）の各要素と、前記対称３重対角行列Ｔの固有値とを用いて固有ベクトルを算出して前記固有ベクトル記憶部に蓄積するベクトル算出ステップと、を備えてもよい。

【０２９６】

また、このプログラムでは、コンピュータに、対称行列Ａが記憶される行列記憶部から前記対称行列Ａを読み出し、前記対称行列Ａを３重対角化した前記対称３重対角行列Ｔを算出して前記対角行列記憶部に蓄積する対角化ステップをさらに実行させてもよい。

なお、上記プログラムにおいて、情報を蓄積するステップなどでは、ハードウェアでしか行われない処理は少なくとも含まれない。

【０２９７】

また、このプログラムは、サーバなどからダウンロードされることによって実行されてもよく、所定の記録媒体（例えば、ＣＤ－ＲＯＭなどの光ディスクや磁気ディスク、半導体メモリなど）に記録されたプログラムが読み出されることによって実行されてもよい。

【０２９８】

また、このプログラムを実行するコンピュータは、単数であってもよく、複数であってもよい。すなわち、集中処理を行ってもよく、あるいは分散処理を行ってもよい。

【０２９９】

図３４は、上記プログラムを実行して、上記実施の形態による固有値分解装置１を実現するコンピュータの外観の一例を示す模式図である。上記実施の形態は、コンピュータハードウェア及びその上で実行されるコンピュータプログラムによって実現されうる。

【０３００】

図３４において、コンピュータシステム１００は、ＣＤ－ＲＯＭ（Ｃｏｍｐａｃｔ Ｄｉｓｋ Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）ドライブ１０５、ＦＤ（Ｆｌｅｘｉｂｌｅ Ｄｉｓｋ）ドライブ１０６を含むコンピュータ１０１と、キーボード１０２と、マウス１０３と、モニタ１０４とを備える。

【０３０１】

図３５は、コンピュータシステムを示す図である。図３５において、コンピュータ１０１は、ＣＤ－ＲＯＭドライブ１０５、ＦＤドライブ１０６に加えて、ＣＰＵ（Ｃｅｎｔｒａｌ Ｐｒｏｃｅｓｓｉｎｇ Ｕｎｉｔ）１１１と、ブートアッププログラム等のプログラムを記憶するためのＲＯＭ（Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）１１２と、ＣＰＵ１１１に接続され、アプリケーションプログラムの命令を一時的に記憶すると共に、一時記憶空間を提供するＲＡＭ（Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）１１３と、アプリケーションプログラム、システムプログラム、及びデータを記憶するハードディスク１１４と、ＣＰＵ１１１、ＲＯＭ１１２等を相互に接続するバス１１５とを備える。なお、コンピュータ１０１は、ＬＡＮへの接続を提供する図示しないネットワークカードを含んでもよい。

【０３０２】

コンピュータシステム１００に、上記実施の形態による固有値分解装置１の機能を実行させるプログラムは、ＣＤ－ＲＯＭ１２１、またはＦＤ１２２に記憶されて、ＣＤ－ＲＯＭドライブ１０５、またはＦＤドライブ１０６に挿入され、ハードディスク１１４に転送されてもよい。これに代えて、そのプログラムは、図示しないネットワークを介してコンピュータ１０１に送信され、ハードディスク１１４に記憶されてもよい。プログラムは実行の際にＲＡＭ１１３にロードされる。なお、プログラムは、ＣＤ－ＲＯＭ１２１やＦＤ１２２、またはネットワークから直接、ロードされてもよい。

【０３０３】

また、行列記憶部１１、対角行列記憶部１３、固有値分解記憶部１６、固有値記憶部１８，固有ベクトル記憶部２０は、ＲＡＭ１１３やハードディスク１１４によって実現され

てもよい。

【０３０４】

プログラムは、コンピュータ１０１に、上記実施の形態による固有値分解装置１の機能を実行させるオペレーティングシステム（ＯＳ）、またはサードパーティプログラム等を必ずしも含まなくてもよい。プログラムは、制御された態様で適切な機能（モジュール）を呼び出し、所望の結果が得られるようにする命令の部分のみを含んでいてもよい。コンピュータシステム１００がどのように動作するのかについては周知であり、詳細な説明を省略する。

また、本発明は、以上の実施の形態に限定されることなく、種々の変更が可能であり、それらも本発明の範囲内に包含されるものであることは言うまでもない。

また、本発明のほんのいくつかの典型的な実施例について上で詳細に説明したが、その典型的な実施例において、発明の利益と新規な技術から実質的にはずれることなく多くの変更が可能であることを当業者は容易に認識することができるであろう。したがって、そのようなすべての変更は、本発明の範囲に含まれるものである。

【産業上の利用可能性】

【０３０５】

以上のように、本発明による固有値分解装置等によれば、固有値分解を高速に処理することができ、分子軌道法による化学計算処理や統計計算処理、情報検索処理、その他の固有値分解を用いる処理を実行する装置等において有用である。

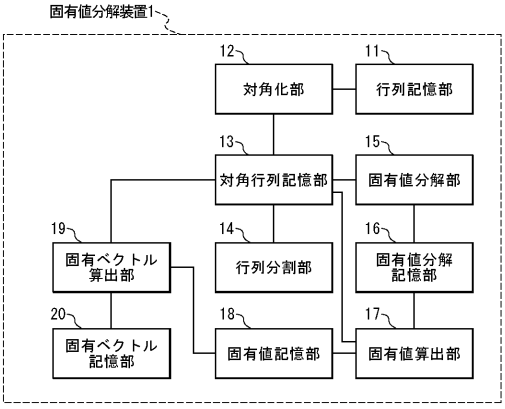
【図面の簡単な説明】

【０３０６】

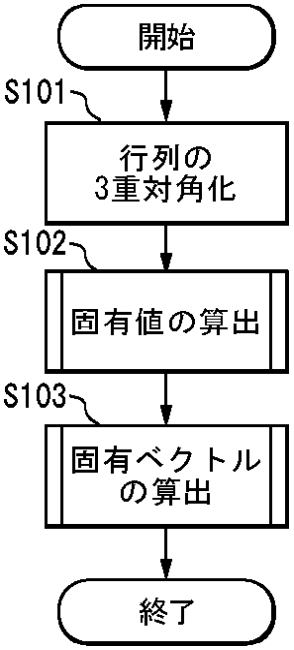
【図１】本発明の実施の形態１による固有値分解装置の構成を示すブロック図
【図２】同実施の形態による固有値分解装置の動作を示すフローチャート
【図３】同実施の形態による固有値分解装置の動作を示すフローチャート
【図４】同実施の形態による固有値分解装置の動作を示すフローチャート
【図５】同実施の形態における対称３重対角行列Ｔの分割について説明するための図
【図６】同実施の形態における対称３重対角行列Ｔの固有値の算出について説明するための図
【図７】同実施の形態による固有値分解装置の動作を示すフローチャート
【図８】同実施の形態による固有値分解装置の動作を示すフローチャート
【図９】同実施の形態における対称３重対角行列Ｔの固有値の算出について説明するための図
【図１０】同実施の形態における固有ベクトル算出部の構成を示すブロック図
【図１１】同実施の形態による固有値分解装置の動作を示すフローチャート
【図１２】同実施の形態による固有値分解装置の動作を示すフローチャート
【図１３】同実施の形態における固有ベクトル算出部の構成を示すブロック図
【図１４】同実施の形態による固有値分解装置の動作を示すフローチャート
【図１５】同実施の形態におけるコレスキー分解について説明するための図
【図１６】同実施の形態による固有値分解装置の動作を示すフローチャート
【図１７】同実施の形態による固有値分解装置の動作を示すフローチャート
【図１８】同実施の形態による固有値分解装置の動作を示すフローチャート
【図１９】同実施の形態による固有値分解装置の動作を示すフローチャート
【図２０】同実施の形態による固有値分解装置の動作を示すフローチャート
【図２１】同実施の形態による固有値分解装置の動作を示すフローチャート
【図２２】同実施の形態におけるｑｄ型ツイスト分解法とＬＶ型ツイスト分解法とについて説明するための図
【図２３】同実施の形態による固有値分解装置の動作を示すフローチャート
【図２４】同実施の形態による固有値分解装置の構成の他の一例を示すブロック図
【図２５】同実施の形態における固有ベクトル算出部の構成の他の一例を示すブロック図
【図２６】同実施の形態における固有ベクトル算出部の構成の他の一例を示すブロック図

【図２７】同実施の形態による固有値分解装置の構成の他の一例を示すブロック図
【図２８】同実施の形態による固有値分解装置の数値実験の結果を示す図
【図２９】同実施の形態による固有値分解装置の数値実験の結果を示す図
【図３０】同実施の形態による固有値分解装置の数値実験の結果を示す図
【図３１】同実施の形態による固有値分解装置の数値実験の結果を示す図
【図３２】同実施の形態における画像処理の一例を示すフローチャート
【図３３】同実施の形態における文書検索処理の一例を示すフローチャート
【図３４】同実施の形態におけるコンピュータシステムの外観一例を示す模式図
【図３５】同実施の形態におけるコンピュータシステムの構成の一例を示す図

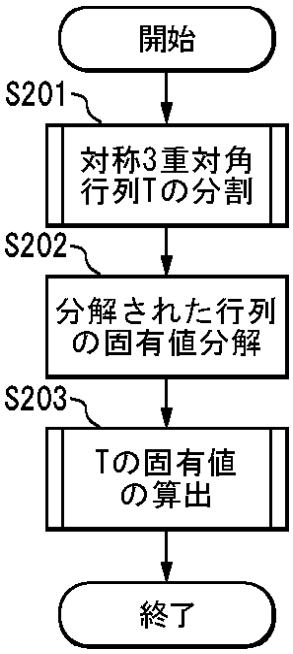
【図１】



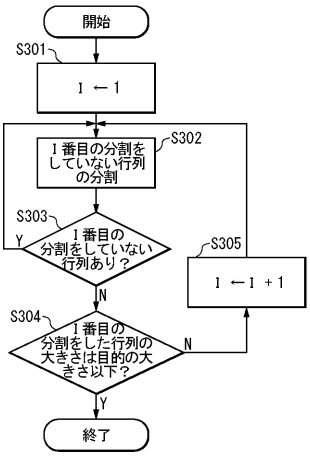
【図２】



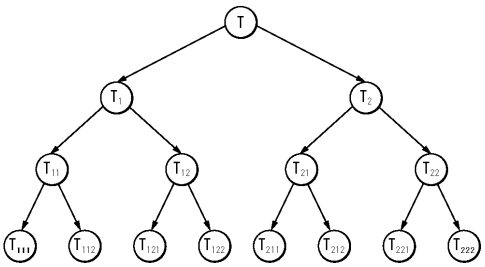
【図３】



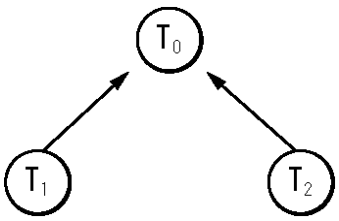
【図４】



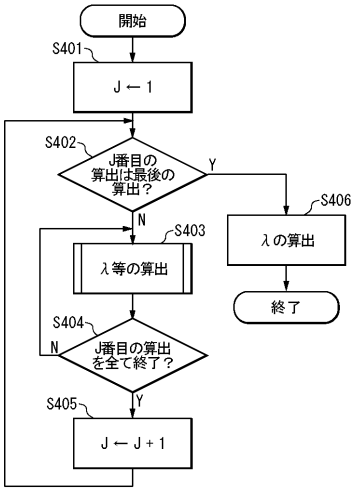
【図５】



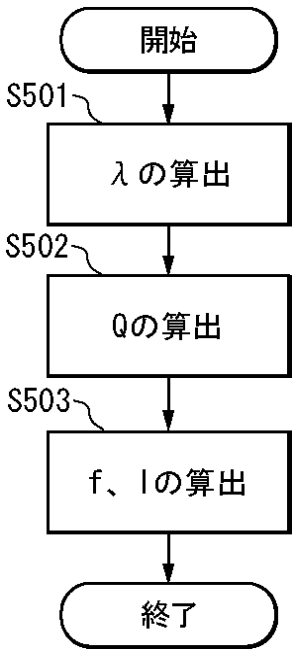
【図６】



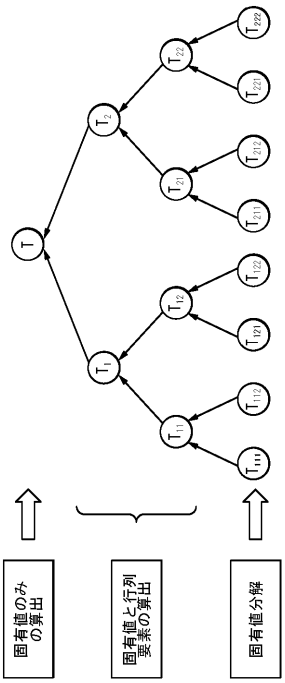
【図７】



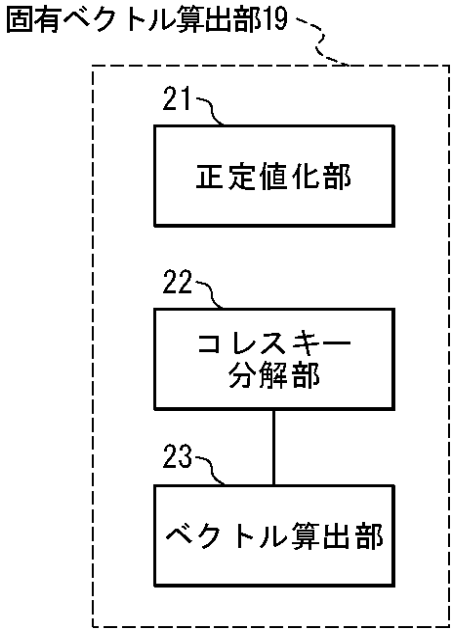
【図８】



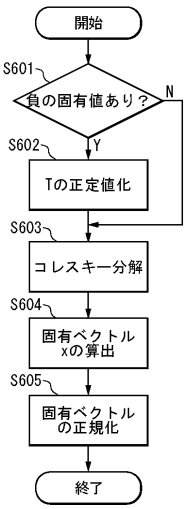
【図９】



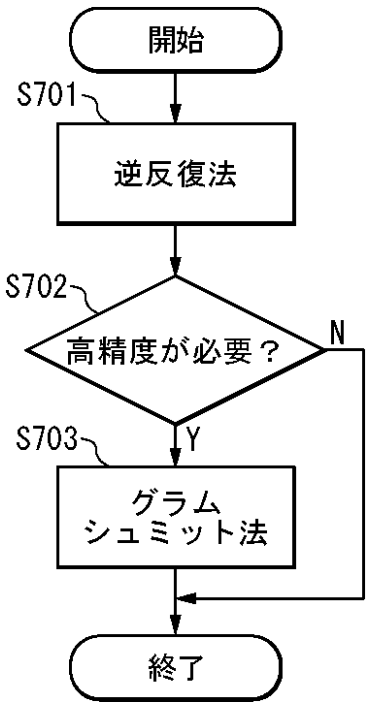
【図１０】



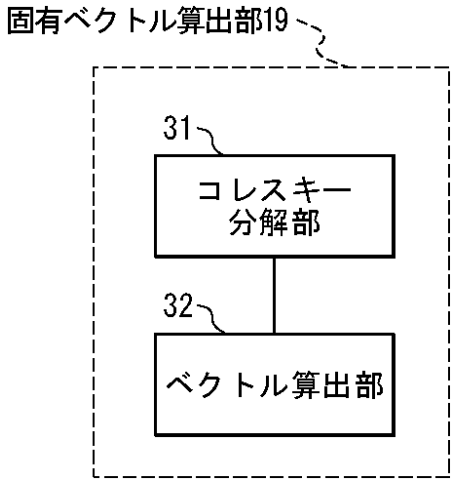
【図１１】



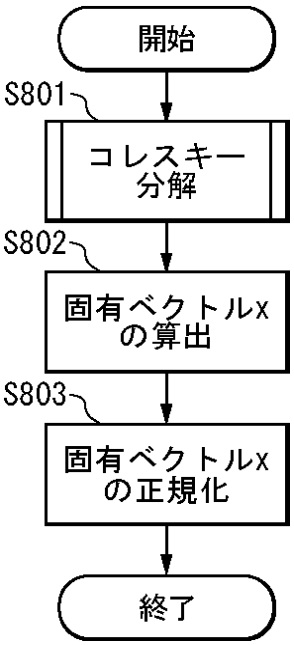
【図１２】



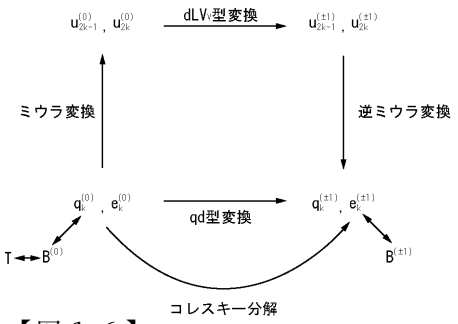
【図１３】



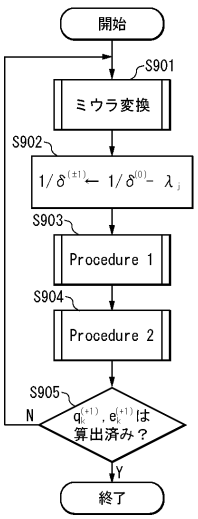
【図１４】



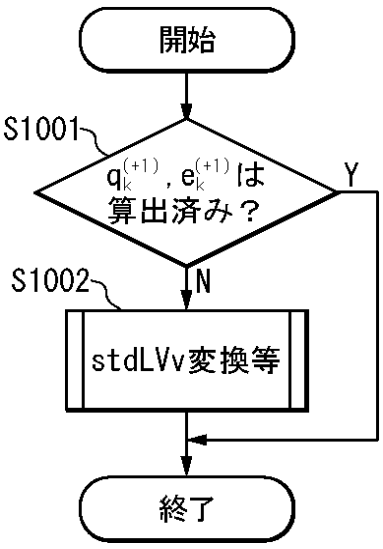
【図１５】



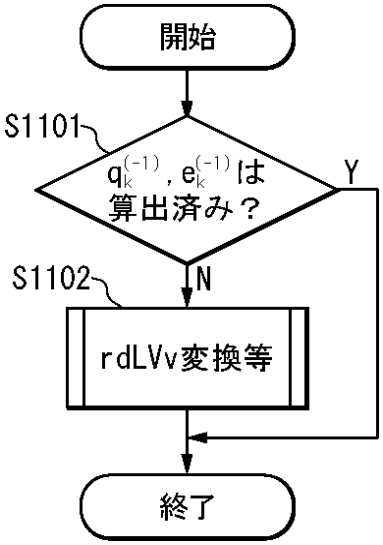
【図１６】



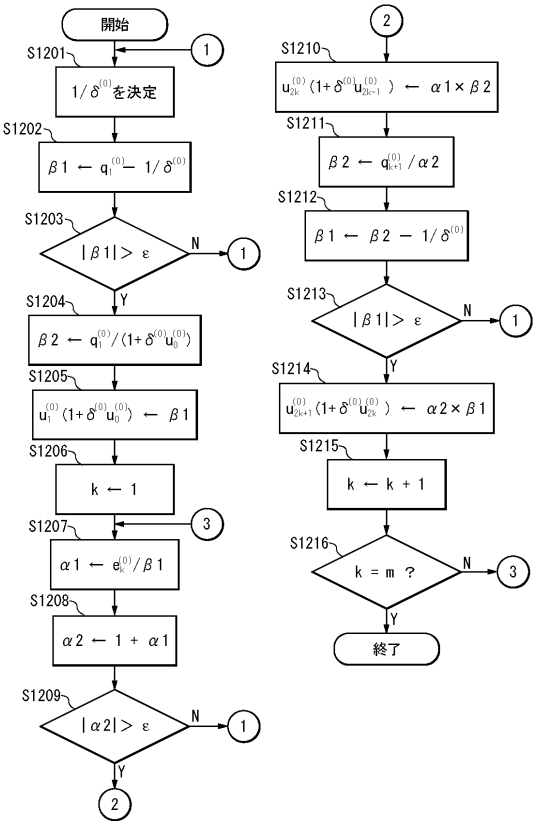
【図１７】



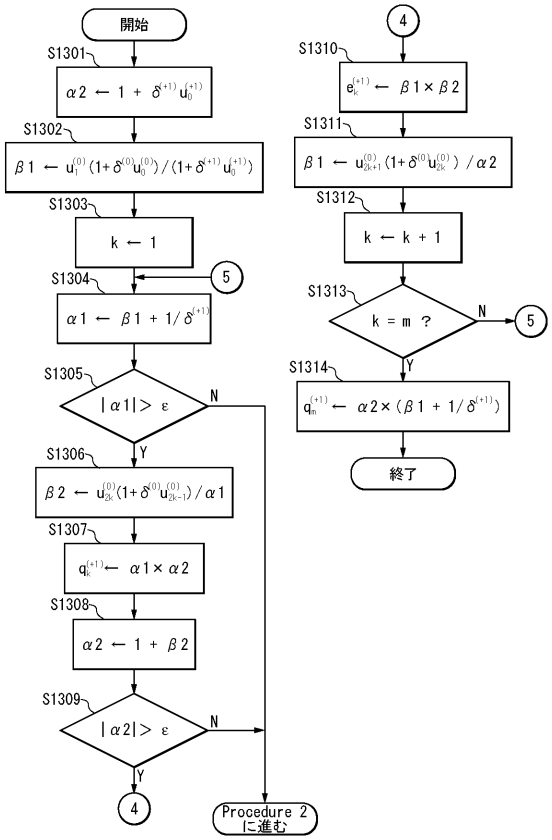
【図１８】



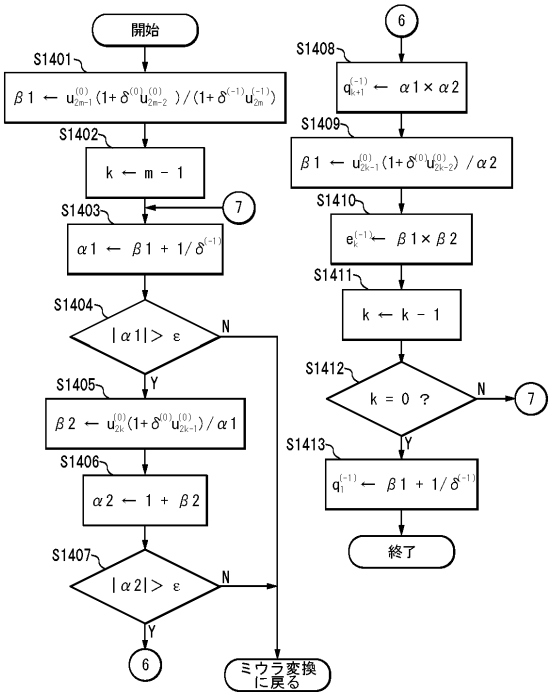
【図１９】



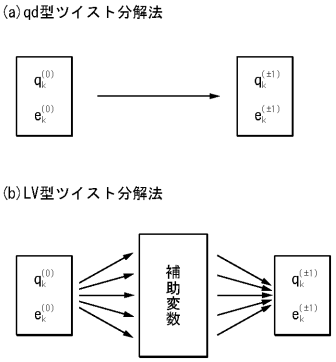
【図２０】



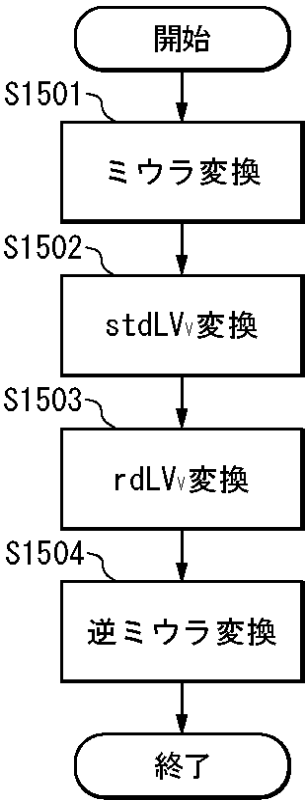
【図２１】



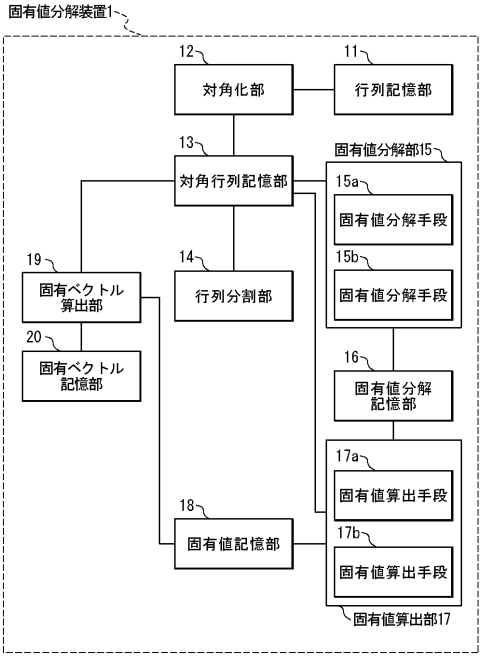
【図２２】



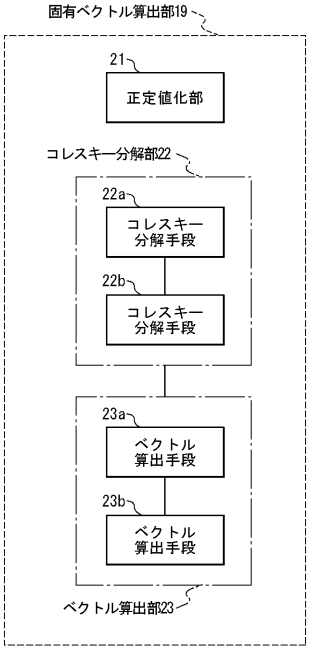
【図２３】



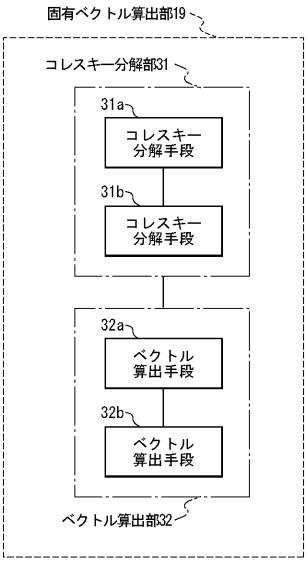
【図２４】



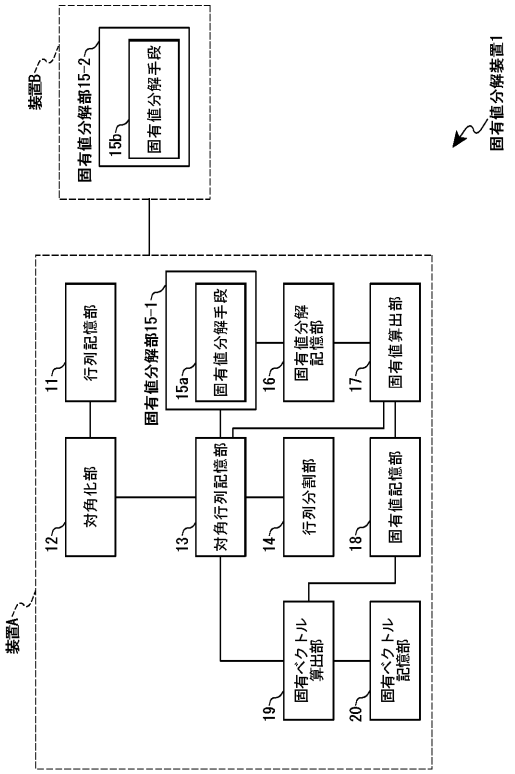
【図２５】



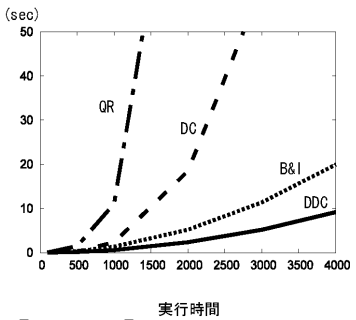
【図２６】



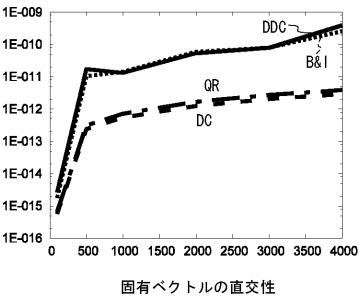
【図２７】



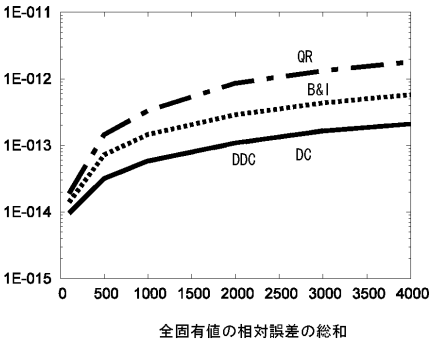
【図２８】



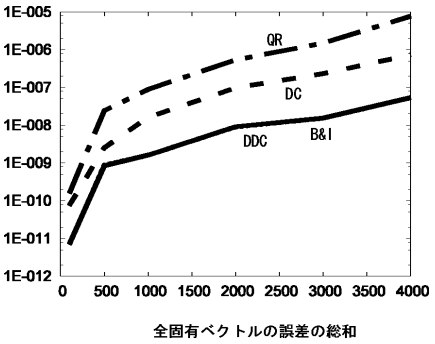
【図２９】



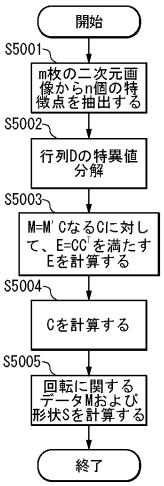
【図３０】



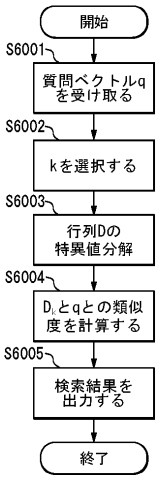
【図３１】



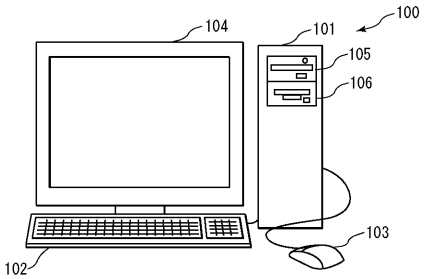
【図３２】



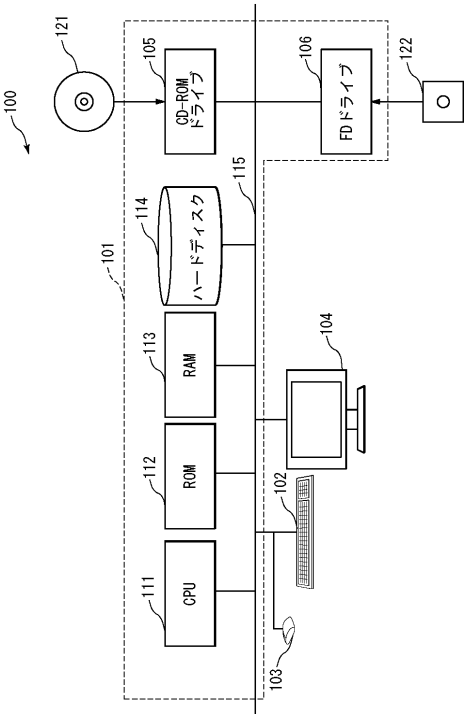
【図３３】



【図３４】



【図３５】



フロントページの続き

(72)発明者　岩▲崎▼　雅史
京都府京都市左京区吉田本町　国立大学法人京都大学大学院情報学研究科内
(72)発明者　高田　雅美
京都府京都市左京区吉田本町　国立大学法人京都大学大学院情報学研究科内

審査官　田中　幸雄

(56)参考文献　国際公開第２００５／１１９５０７（WO，Ａ１）
桑島豊ほか，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，日本，
日本応用数理学会，２００５年　6月２５日，vol．15，no．2，８９－１１５頁

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F　17/16
G06T　1/00