数式処理による行列の標準形の厳密計算法

1996年3月

栗山和子

①

数式処理による行列の標準形の厳密計算法

1996 年 3 月

栗山和子

目次

1 はじめに 1

2 基本概念 8

3 Jordan 標準形の計算法 14

3.1 定義 14

3.2 アルゴリズム 18

3.3 計算例 20

3.4 計算結果と考察 23

3.5 Jordan 標準形の応用 37

3.5.1 線形差分方程式の解法 37

3.5.2 定係数連立線形微分方程式の解法 42

3.5.3 計算結果と考察 47

4 第 2 の標準形と Jacobson 標準形の計算法 52

4.1 定義 52

4.2 アルゴリズム 54

4.3 計算例 55

4.4 計算結果と考察 56

5 有理標準形の計算法 59

5.1 従来のアルゴリズム 59

5.1.1 定義 59

5.1.2 従来のアルゴリズムの問題点 60

5.2 アルゴリズムの効率化 61

5.2.1 基本概念 61

5.2.2 分数なしアルゴリズム 63

5.3 計算例 65

5.3.1 従来のアルゴリズムによる計算例 65

5.3.2 分数なしアルゴリズムによる計算例 67
5.4 計算結果と考察 72

6 おわりに 89

謝辞 91

参考文献 92

付録 102

A 有理標準形の構成法 102

B プログラムリスト 107
B.1 標準形の計算プログラム 107
B.2 Jordan 標準形の応用のプログラム 119

C データと計算結果 123
C.1 テストデータ一覧 123
C.2 REDUCE3.5 による計算結果 124
C.2.1 標準形の計算結果 124
C.2.2 応用の計算結果 133

表一覧

3.1 Jordan 標準形の計算時間 (データ 3.1) 27
3.2 Jordan 標準形の計算時間 (データ 3.2) 29
3.3 Jordan 標準形の計算時間 (データ 3.3) 31
3.4 Jordan 標準形の計算時間 (データ 3.4) 33
3.5 応用の CPU 時間 (データ 3.1) 47
3.6 応用の CPU 時間 (データ 3.3) 47
3.7 小行列の個数と定義多項式の次数 (データ 3.1) 50
3.8 小行列の個数と定義多項式の次数 (データ 3.3) 50

4.1 2 つの標準形の CPU 時間 (データ 3.3) 56
4.2 2 つの標準形の CPU 時間 (データ 3.4) 56

5.1 有理標準形の CPU 時間とその比（データ 5.1 : コンパニオン行列 1 個） 74
5.2 有理標準形の CPU 時間とその比（データ 5.2 : コンパニオン行列 1 個） 76
5.3 有理標準形の CPU 時間とその比（データ 5.3 : コンパニオン行列 3 個） 78
5.4 有理標準形の CPU 時間とその比（データ 5.4 : コンパニオン行列 3 個） 80
5.5 有理標準形の CPU 時間とその比（データ 3.1 : コンパニオン行列 1 個） 82
5.6 有理標準形の CPU 時間とその比（データ 3.4 : コンパニオン行列 3 個） 84

図一覧

1.1 Jordan 標準形の計算法 3

3.1 Jordan 標準形の計算時間のグラフ (データ 3.1) 28
3.2 Jordan 標準形の計算時間のグラフ (データ 3.2) 30
3.3 Jordan 標準形の計算時間のグラフ (データ 3.3) 32
3.4 Jordan 標準形の計算時間のグラフ (データ 3.4) 34
3.5 応用の CPU 時間のグラフ (データ 3.1) 48
3.6 応用の CPU 時間のグラフ (データ 3.3) 49

4.1 2 つの標準形の CPU 時間のグラフ (データ 3.3) 57
4.2 2 つの標準形の CPU 時間のグラフ (データ 3.4) 57

5.1 有理標準形の CPU 時間のグラフ（データ 5.1 : コンパニオン行列 1 個） 75
5.2 有理標準形の CPU 時間のグラフ（データ 5.2 : コンパニオン行列 1 個） 77
5.3 有理標準形の CPU 時間のグラフ（データ 5.3 : コンパニオン行列 3 個） 79
5.4 有理標準形の CPU 時間のグラフ（データ 5.4 : コンパニオン行列 3 個） 81
5.5 有理標準形の CPU 時間のグラフ（データ 3.1 : コンパニオン行列 1 個） 83
5.6 有理標準形の CPU 時間のグラフ（データ 3.4 : コンパニオン行列 3 個） 85
5.7 有理標準形の CPU 時間の比のグラフ（データ 5.1,5.2 : コンパニオン行列 1 個） . . . 86
5.8 有理標準形の CPU 時間の比のグラフ（データ 5.3,5.4 : コンパニオン行列 3 個） . . . 86
5.9 有理標準形の CPU 時間の比のグラフ（データ 3.1,3.4） 87

第 1 章

はじめに

本研究の目的は、1 変数整数係数多項式を要素とする任意の n 次正方行列を、数式処理を用いた厳密計算によって Jordan 標準形をはじめとするいくつかの標準形に変換するアルゴリズムを示し、その改良について考察することである。

本研究では、次の標準形と変換行列を計算するアルゴリズムを作成し、有理標準形と Jordan 標準形の計算アルゴリズムについて改良・効率化を行なった。

・有理標準形（Frobenius 標準形、第 1 の標準形（First Natural Normal Form））。
・有理標準形への変換行列とその逆行列。
・第 2 の標準形（Second Natural Normal Form）。
・ Jacobson 標準形。
・ Jordan 標準形。
・ Jordan 標準形への変換行列とその逆行列。

本研究のアルゴリズムでは、まず、与えられた行列の有理標準形を求め、その有理標準形から他の標準形と変換行列を計算する。

有理標準形から Jordan 標準形を計算するアルゴリズムは既に知られている。しかし、そのアルゴリズムを含めて、すべての既知の Jordan 標準形の計算法には共通の限界がある。

Jordan 標準形の計算法としては、次の 3 つの方法がよく知られている [57]。

固有値に基づく方法：　この方法では、まず、行列 A の固有値 λ を求め、次に、各固有値 λ_i に対する特性行列のべき $(\lambda_i E - A)^l$ の階数を用いて A の Jordan 標準形を求める。

単因子に基づく方法：　この方法では、まず、行列 A の特性行列 $\lambda E - A$ を単因子標準形（Smith 標準形）とよばれる行列に変形して、$\lambda E - A$ の単因子を求め、次に、その単因子から A の Jordan 標準形を求める。

有理標準形に基づく方法：　この方法では、まず、もとの行列を有理標準形に変形し、次に、そ

の有理標準形から Jordan 標準形への変換行列を有理標準形自体から作り、その変換行列によって有理標準形を Jordan 標準形に変換する。

これらの計算法には、中間のステップとして、必ず、行列の固有値の計算（代数方程式の求解）が含まれる。もとの行列の要素が有理数（あるいは 1 変数整数係数多項式。以下、括弧内は要素が 1 変数整数係数多項式の場合に対応する）であっても、その行列の固有値は必ずしも有理数（1 変数整数係数多項式）であるとは限らない。つまり、固有値を根としてもつ多項式、すなわち、特性多項式を有理数体（1 変数有理関数体）上で因数分解すると、その既約因子の中に複素数（代数関数）を根にもつ 2 次以上の多項式が含まれる場合がある。ところが、5 次以上の代数方程式を代数的に解くアルゴリズムは存在しない。したがって、「固有値が 5 次以上の方程式の根」である場合には、固有値を代数的に計算することはできず、厳密に Jordan 標準形を求めることはできない。また、固有値が 4 次以下の方程式の根である場合には、代数的に固有値を計算することはできるが、固有値が 3 次、4 次の方程式の根であるときには、数式処理を用いて代数的にその方程式を解くと、その根は 3 乗根、4 乗根を含む複雑な表現になり、数式処理システム上での扱いは非常に困難になる。数式処理システム上では、代数的数については、その表現の一意性は必ずしも保証されない。例えば、「固有値に基づく方法」では、階数の計算（Gauss の消去法）の過程において代数的数を含む行列要素が 0 に等しいかどうかというような判定が困難になる [70]。

本研究では「有理標準形に基づく方法」を改良し、固有値を代数的に計算できない場合にも、Jordan 標準形の構造を厳密に求められるアルゴリズムを作成した [82]。すなわち、まず、有理標準形と単因子との関係と、有理数体（1 変数有理関数体）上の単純単因子（単因子を有理数体（1 変数有理関数体）上で因数分解したとき含まれる既約因子のべき）と Jordan 細胞との関係を明らかにした。次に、それらの関係に基づき、既知の計算法を改良し、有理数体（1 変数有理関数体）に属さない固有値を記号を用いて表すことによって、Jordan 標準形の構造を厳密に計算できるアルゴリズムを作成した。記号として表された固有値は、必要な精度まで数値解析的に計算すればよい。その場合にも、既に求められた Jordan 標準形の構造は確定しており、固有値の数値的精度の影響は受けない。次頁に、行列の要素が有理数であるときの、従来の 3 つの Jordan 標準形の計算法と本研究の計算法との関係を図として示す。図中の矢印の上の×印は、『矢印で示されている計算のステップには、「行列の固有値の計算（代数方程式の求解）」が含まれている』という意味である。

図 1.1: Jordan 標準形の計算法

改良する方法として「有理標準形に基づく方法」を選んだ理由としては、次のようなことが挙げられる。

(1) 「固有値に基づく方法」は、固有値が具体的に求まることを前提としている。

(2) 「単因子に基づく方法」では、計算時間に対して支配的である単因子標準形の計算は、多項式行列に対する有理演算であり、行列の次数が大きくなると、要素の多項式の次数も増大し係数成長も起こるため、計算時間の増大が著しくなる。

(3) 「有理標準形に基づく方法」では、有理標準形の計算は、単因子標準形の計算と同様に有理演算であるが、もとの行列が数値行列である場合には、数値行列に対する演算であるため、次数の増大や多項式間の Euclid の互除法に現れるような係数成長も起こらないので、単因子標準形の計算よりも容易である。もとの行列の要素が 1 変数多項式の場合には、単因子標準形の計算が 2 変数多項式行列に対する演算になるのに対し、有理標準形の計算は 1 変数多項式行列に対する演算のままである。

(4) [70] より、上記の 3 方法の中では、「有理標準形に基づく方法」が計算時間の点で最も有利である。

(5) Jordan 標準形の応用を考えたとき、Jordan 標準形への変換行列を同時に計算できることが望ましいが、3 方法の中で同時に変換行列を計算できるのは、「有理標準形に基づく方法」だけである。

また、本研究では、Jordan 標準形のほかに、第 2 の標準形、Jacobson 標準形、記号を要素とする Jordan 標準形への変換行列も計算できるようにアルゴリズムを拡張した。

Jordan 標準形への変換行列は、固有値と各単因子中に含まれる固有値の重複度から求めるため、固有値が記号で表されている場合には、その要素も記号の式となる。そのとき、同じ既約多項式の根である各固有値を表す各記号間については、その共役関係を表現し、式を簡単化しなければならない。研究当初は、行列の要素が有理数である場合には、既約因子の分解体（有理数体 $\mathbf{Q}$ に既約因子のすべての根 $\lambda_1, \lambda_2, \cdots$ を添加した代数的拡大体 $\mathbf{Q}(\lambda_1, \lambda_2, \cdots)$）を構成し、その原始元を用いて固有値間の関係を表すことを考えていた。

拡大体の構成法としては、終結式による方法 [76],[111],[123]、線形方程式を用いた方法 [121]、Gröbner 基底を用いる方法 [123] があるが、現在のところ、終結式による方法を文献 [80] の方法を用いて改良した [123] のアルゴリズムが最も効率的である。終結式による方法には、拡大体上での因数分解が含まれる。拡大体上での因数分解は、実際には、その拡大体の原始元の最小多項式を法とする剰余類体上での因数分解として行われる。したがって、拡大が進むにつれて因数分解は困難になり、既約多項式の次数が大きくなると、（アルゴリズム上は計算可能であるのに）数式処理システム上では計算ができなくなってしまう。本研究で使用している数式処理システム REDUCE (version 3.5) [41] 上には、[111] のアルゴリズムがインプリメントされているが、そのプログラムでは、4 次程度

の既約多項式の分解体までしか計算することができない。よって、数式処理システム上の分解体の計算は非常に困難であり、現時点では、実用的ではないということが分かった。

次に、方程式の「解と係数の関係」の式から生成されるイデアルの Gröbner 基底を計算し、その基底を用いて記号間の関係を表現し、式の簡単化（M- 簡約）を行なった。この方法は根の共役関係を表現するためには有効である [69],[71] が、共役関係にある根を 1 つの根（記号）の多項式として表すことになるため、表現が煩雑になることがある。そこで、現在では、固有値を根とする既約多項式、すなわち固有値の定義多項式を用いて式の簡単化（M- 簡約）を行ない、同じ既約多項式の根である複数の固有値を組にして扱っている。

本研究では、次のような簡単な Jordan 標準形の応用 [57] について考察し、アルゴリズムを試作した。

- 1 階連立線形差分方程式の解法。
- 1 階定係数連立線形微分方程式の解法。

1 階連立線形差分方程式と 1 階定係数連立線形微分方程式の解法には、1 つの共通点がある。それは、その解の係数行列を Jordan 標準形とその変換行列を利用して計算できるということである。本研究では、前述の計算法を利用して、それぞれの解の係数行列を計算するアルゴリズムを作成した。

さらに、本研究では、改良した「有理標準形に基づく計算法」において重要である有理標準形自体の計算法の効率化した。「有理標準形に基づく方法」では、有理標準形の計算にかかる時間が全体の計算時間に対して支配的であるので、計算の対象が数値行列よりも多くの計算時間を要する 1 変数多項式行列である場合には、有理標準形の計算法の効率化が必要となる。

有理標準形の計算法では、Gauss の消去法に類似した階段化消去を行なう。階段化消去を行なうとき、入力行列と出力行列の要素が多項式であったとしても、変形操作には除算が含まれているため、一般に、中間結果の行列の要素は有理式となる。有理式同士の演算では、多項式同士の計算に比べて、通分・約分に余分な時間がかかり、また、共通分母は計算の段階が進むにつれて増大するので、途中の計算ではより多くのメモリを必要とする。

本研究では、有理標準形の計算の中間結果に対する階段化消去を効率化するため、Bareiss の分数なし Gauss の消去法 (Single–step fraction–free elimination) [7], [100] に類似した、途中の計算における有理式の出現を回避するアルゴリズムを開発した。

最後に、作成したすべてのアルゴリズムを数式処理システム REDUCE (version 3.5) 上に実験的にインプリメントし、その計算時間について比較・考察を行なった。

行列の標準形に関する研究は、既に多数発表されているが、本研究で計算する 4 つの標準形の構成的算法に関しては、Jordan 標準形と有理標準形に関する文献が多く、第 2 の標準形と Jacobson 標準形について構成的算法を述べている文献はほとんど見あたらない。行列の標準形としてこの 4 つの標準形すべてを挙げているのは [5] のみであり、[12] に有理標準形・Jordan 標準形・Jacobson 標

準形が、[27] に有理標準形・第 2 の標準形・ Jordan 標準形が定義的に述べられている。

Jordan 標準形の計算については、広い意味での「固有値に基づく方法」に分類できる方法が多い。「広い意味での」というのは、まず、固有値を計算し、次に、固有値を用いて Jordan 標準形への変換行列を構成するベクトル列の計算に類することを行なう方法はすべて含むということである。広い意味での「固有値に基づく方法」は、次のように、大きく 3 つに分けられる。

(1) 行列の基本変形の合成を用いて変換していく方法。

(2) 逆反復法などによって $(\lambda_i E - A)\mathbf{x}_l = \mathbf{x}_{l-1}$ を順次解く方法。

(3) 前述のように、特性行列のべき $(\lambda_i E - A)^l$ の階数を用いる方法。

数値計算に方法 (1),(2) を適用したものとして、それぞれ、(1)[52], (2)[96] が挙げられる。実際にアルゴリズムをインプリメントした結果、[52] では最大 12 次、[96] では既に固有値が分かっている場合で最大 18 次の行列を処理できたと述べられている。ただし、どちらのアルゴリズムも数値計算に固有の近似計算によって引き起こされる問題（近似計算での多重固有値と近接固有値の判別など）の取り扱いに重点を置いているので、そのまま数式処理に適用することは適当でない。(3) を数式処理に適用したものとしては、[104] のアルゴリズムが、数式処理システム MACSYMA にインプリメントされている。その他に、(1) ～ (3) とは異なった新しいアプローチとして、近似代数の概念を導入し、近似重複固有値と Schur 分解を用いて Jordan 標準形を計算する方法が発表されている [64]。

「固有値に基づく方法」に対して、「単因子に基づく方法」、「有理標準形に基づく方法」は、単因子標準形・有理標準形の計算自体がまた 1 つの問題となるためか、これらについての文献は少なかった。数式処理への適用としては、「単因子に基づく方法」による、数式処理システム REDUCE を用いたプログラム [87] が公表されている。

単因子標準形（Smith 標準形）[102] 自体の計算法については、基本算法の組合せとしては最良の計算量をもつアルゴリズムが、[59] によって与えられているほか、並列化・確率的算法による改良が [53],[54],[55] に示されている。

有理標準形 [27] については、Danilevskii 法 [17] を、数値計算での特性多項式、固有値、固有ベクトルの計算への応用として述べたもの [22],[26],[37] がある。Danilevskii 法は、行列の特性多項式を求めるアルゴリズムとしては最良の計算量をもつが、数値的精度の問題から、数値計算ではほとんど実用にならない。しかし、数式処理の無限精度演算では、この方法は非常に有用 [109],[110] であり、モジュラ算法による改良 [29],[30],[44],[79],[83] も提案されている。入力行列が数値行列である場合には、行列の次数 n に対して、Danilevskii 法の計算量は、$O(n^4)$ とされている [79]。文献 [29], [30] では、確率的算法 [56] とモジュラ算法を組み合わせて、その計算量を改善している。入力行列の要素が多項式である場合には、要素が数値である場合に比べて、モジュラ計算（多項式を法とする有理標準形の計算と、それを用いた入力行列の有理標準形の復元）に必要とされる計算量が大幅に増加し、また、数値行列に対して用いていたアルゴリズムをそのまま適用することはできない。その他の方法でも、多項式行列の有理標準形の計算アルゴリズムの改良についての研究は行なわれていない。

数式処理システムの現状としては、本研究で使用したREDUCE (version 3.5) 自体には行列の標準形に関する組込の機能はないが、ユーザが作成したパッケージの中に数式処理システムMapleの関数をREDUCE用にLISPで書き直したプログラムが存在する。このプログラムでは、本研究で用いたデータのすべての標準形を本研究のプログラムと同程度の時間内で計算することはできず、メモリもそれ以上に必要とした。また、前述のように「単因子に基づく方法」によるプログラム [87] も公表されている。ただし、このプログラムでは、「固有値が高々2次方程式の根」として与えられる数値行列しか計算しないようになっている。その他の代表的な数式処理システムであるMACSYMAには「固有値に基づく方法」でJordan標準形を計算する関数jordanform [104] を含むパッケージがインプリメントされている。この関数では、数だけでなく多変数多項式を要素とする行列も処理可能であるが、代数方程式を直接代数的に解いて固有値を求めているため、「固有値が5次以上の方程式の根」である場合には処理できない。Mathematicaには、JordanForm（あるいはJordanDecomposition）というJordan標準形を求める関数が存在するが、実際に使用してみたところ、上記の2つのプログラム同様、代数方程式を直接解いて固有値を計算してJordan標準形を求めていることが分かった。MACSYMA, Mathematicaにはその他の3つの標準形に関する組込の機能はない。
Mapleには、Jordan標準形、有理標準形、その変換行列を計算する組込関数が存在するが、ソースプログラムが非公開であるため、どのようなアルゴリズムを用いているかは不明である。その他のシステムとしては、Axiomを用いて、Jordan標準形について本研究と同様のアプローチをしている研究 [31] が発表されている。

したがって、4つの標準形は数学的定理としては既知であるが、数式処理システム上でさらに実用的な機能を実現するためには、数式処理の特性を生かした、アルゴリズムの改良・インプリメント・結果の解析が必要であると考え、本研究のテーマとした。

第 2 章

基本概念

本章では、3 章以下で用いる基本的な用語の定義を与える。数学的によく知られた定理は、証明なしで用いるので、文献 [5],[45],[57],[106],[107] を参照されたい（ただし、用語・記法は必ずしも共通ではないので、本稿では以下のように統一する）。

定義 1 (行列の直和) $A_1, A_2, \cdots, A_r$ をそれぞれ $n_1, n_2, \cdots, n_r$ 次の正方行列とする。これらの行列を $(1,1)$ 要素が A_1 の $(1,1)$ 要素に、(n_1+1, n_1+1) 要素が A_2 の $(1,1)$ 要素に、・・・、$(n_1+\cdots+n_{r-1}+1, n_1+\cdots+n_{r-1}+1)$ 要素が A_r の $(1,1)$ 要素にそれぞれ一致するように対角形に並べ、さらに $A_1, \cdots, A_r$ が並べられたところ以外の要素はすべて 0 としてできる $n_1+\cdots+n_r$ 次の正方行列

$$A = \begin{bmatrix} A_1 & & & O \\ & A_2 & & \\ & & \ddots & \\ O & & & A_r \end{bmatrix} \tag{2.1}$$

を $A_1, A_2, \cdots, A_r$ の直和といい、$A = A_1 \oplus A_2 \oplus \cdots \oplus A_r$ と書く。この形の行列 A をブロック対角行列といい、各 $A_i (i = 1, \cdots, r)$ を A のブロックとよぶ。

定義 2 (特性行列と特性多項式) 対角要素がすべて 1 であり、他の要素がすべて 0 であるような n 次正方行列を単位行列とよび、E と書く。任意の n 次正方行列 A に対して行列 $\lambda E - A$ を A の特性行列とよぶ。また、A の特性行列の行列式 $\det(\lambda E - A)$ を A の特性多項式とよび $\varphi_A(\lambda)$ と書く。$\varphi_A(\lambda) = 0$ とおいたものを A の特性方程式とよび、それを満足する λ を A の固有値という。

定義 3 (最小多項式) n 次元ベクトル空間を V_n とする。一次変換 $\mathbf{A} : \mathrm{V}_n \to \mathrm{V}_n$ に対して、$f(\mathbf{A}) = \mathbf{O}$（零変換）となるような多項式 $f(\lambda)$ は 1 つ以上ある。その中で、最小次数の多項式を $\mathbf{A}$ の最小多項式とよび $\phi_{\mathbf{A}}(\lambda)$ と書く。正方行列 A に対しても同じように、最小多項式 $\phi_A(\lambda)$ が $f(A) = O$（零行列）となる最小次数の多項式として定義される。A の最小多項式 $\phi_A(\lambda)$ は A の特性多項式 $\varphi_A(\lambda)$ を割り切り、$\phi_A(\lambda)$ の根には、相異なるすべての固有値が含まれる。また、A の固有値がすべて相異なるならば、$\varphi_A(\lambda)$ と $\phi_A(\lambda)$ は一致する。

定義 4 (Jordan 細胞) n 次の正方行列

$$J(\lambda, n) = \begin{array}{c} \begin{bmatrix} \lambda & 1 & & & O \\ & \lambda & 1 & & \\ & & \ddots & \ddots & \\ & & & \lambda & 1 \\ O & & & & \lambda \end{bmatrix} \end{array} \begin{array}{c} \uparrow \\ \\ n \\ \\ \downarrow \end{array} \tag{2.2}$$

を、λ を固有値とする n 次の Jordan 細胞という。特に、$n = 1$ のときは、$J(\lambda, 1) = [\lambda]$ とする。

定理 1 (Jordan 標準形) 任意の n 次正方行列を A とする。A は適当な正則行列 $S(\det S \neq 0)$ を用いると、相似変換 $S^{-1}AS$ によって

$$J = S^{-1}AS = \begin{bmatrix} \Lambda_1 & & & O \\ & \Lambda_2 & & \\ & & \ddots & \\ O & & & \Lambda_p \end{bmatrix}, \quad \Lambda_i = \begin{bmatrix} J_{i1} & & & O \\ & J_{i2} & & \\ & & \ddots & \\ O & & & J_{iq_i} \end{bmatrix} \tag{2.3}$$

という形のブロック対角行列に変換することができる。J を A の Jordan 標準形という。ここで、各ブロック $J_{ij}(i = 1, \cdots, p; j = 1, \cdots, q_i)$ は、A の固有値 λ_i に対する n_{ij} 次の Jordan 細胞 $J(\lambda_i, n_{ij})$ の k_{ij} 個の直和

$$J_{ij} = J(\lambda_i, n_{ij}) \oplus \cdots \oplus J(\lambda_i, n_{ij}) \tag{2.4}$$

である。ただし、$k_{ij} = 0$ のとき、J_{ij} はないものとする。ここで、$\sum_{j=1}^{q_i}(n_{ij} \times k_{ij}) = m_i$ が成り立つ。m_i は特性方程式における λ_i の重複度であり、$m_1 + \cdots + m_p = n$ である。A の Jordan 標準形 J は、Jordan 細胞の並べ方の順序を除き、A によって一意的に定まる。Jordan 標準形は任意の体上の元を要素とする行列について存在する。

定義 5 (多項式行列) 環 $\mathbf{D}$ の元を係数に持つ x の多項式 $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ $(a_i \in \mathbf{D})$ の全体を $\mathbf{D}[x]$ で表す。$\mathbf{D}[x]$ の元について和と積を定義すれば、$\mathbf{D}[x]$ は $\mathbf{D}$ を含む可換環となる。$\mathbf{D}[x]$ を多項式環とよぶ。また、$\mathbf{D}[x]$ の元を要素とするような行列を多項式行列とよび、$A(x), B(x), \cdots$ と書く。

定義 6 (基本変形と対等) 多項式環 $\mathbf{D}[x]$ の元を要素とする多項式行列 $A(x)$ に、基本変形とよばれる次の 6 つの操作を有限回施して、新しい多項式行列 $B(x)$ が得られるとき、$A(x)$ と $B(x)$ は対等であるといい、$A(x) \cong B(x)$ と表す。ただし、有限回の中には 0 回も含める。

(行 1) $A(x)$ の第 k 行と第 l 行を入替える。

(行 2) $A(x)$ の第 k 行を c 倍する $(c \in \mathbf{D})$。

(行 3) $A(x)$ の第 k 行に多項式 $f(x)$ $(\in \mathbf{D}[x])$ を掛けて、第 l 行に加える。

(列 1) $A(x)$ の第 k 列と第 l 列を入替える。

(列 2) $A(x)$ の第 k 列を c 倍する $(c \in \mathbf{D})$。

(列 3) $A(x)$ の第 k 列に多項式 $f(x)$ $(\in \mathbf{D}[x])$ を掛けて、第 l 列に加える。

定理 2 (単因子標準形) 任意の $m \times n$ の多項式行列を $A(x)$ とすると、$A(x)$ は有限回の基本変形によって

$$A(x) \simeq \begin{bmatrix} e_1(x) & & & & & & \\ & e_2(x) & & & & O & \\ & & \ddots & & & & \\ & & & e_r(x) & & & \\ & & & & 0 & & \\ & O & & & & \ddots & \\ & & & & & & 0 \quad \cdots \quad 0 \end{bmatrix} \tag{2.5}$$

という形の対角行列に変換することができる。この対角行列を $A(x)$ の単因子標準形 (Smith 標準形) という。ただし、$0 \leq r \leq p = \min(m, n)$ であり、多項式 $e_i(x)$ は次の (i),(ii) を満たす。

(i) $e_{i-1}(x)$ は $e_i(x)$ を割り切る $(2 \leq i \leq r)$。

(ii) $e_i(x) \neq 0$ の最高次の係数は 1 である。

この $e_1(x), \cdots, e_r(x)$ を $A(x)$ の単因子という。単因子は $A(x)$ によって一意的に定まる。

定理 3 (特性行列の単因子) 任意の n 次正方行列 A に対し、特性行列 $\lambda E - A$ の単因子を $e_1(\lambda), \cdots, e_n(\lambda)$ (A の特性多項式 $\varphi_A(\lambda)$ は λ の n 次多項式であり、$\det(\lambda E - A) = \varphi_A(\lambda) \neq 0$ であるから、$\lambda E - A$ の階数は n であり、その (0 でない) 単因子は n 個ある) とすれば、A の最小多項式 $\phi_A(\lambda)$ は、$\lambda E - A$ の最後の単因子 $e_n(\lambda)$ に等しい。

定義 7 (単純単因子) 多項式行列 $A(x)$ の 1 でない単因子 $e_1(x), \cdots, e_t(x)$ を既約多項式 $p_{ij}(x)$ (最高次係数は 1) のべきの積として

$$e_i(x) = p_{i1}(x)^{k_{i1}} \cdots p_{ir_i}(x)^{k_{ir_i}} \quad (i = 1, \cdots, t) \tag{2.6}$$

(p_{ij} と p_{il} は $j \neq l$ のとき互いに素で $k_{ij} \geq 1$ とする) と因数分解するとき、$p_{ij}(x)^{k_{ij}}$ $(i = 1, \cdots, t;$ $j = 1, \cdots, r_i)$ を $A(x)$ の単純単因子という。因数分解を行なう基礎体を決めれば、これは、$A(x)$ によって一意的に定まる。

定理 4 (Jordan 細胞の次数と個数) 任意の正方行列 A の特性行列 $\lambda E - A$ の単純単因子全体の中に $(\lambda-\lambda_i)^{k_i}$ が m 個あるとすれば、A の Jordan 標準形 J の中に固有値 λ_i の k_i 次 Jordan 細胞 $J(\lambda_i, k_i)$ は、ちょうど m 個だけ含まれる。

定義 8 (巡回部分空間) ベクトル空間 V_n の上の一次変換を $\mathbf{A}$ とする。あるベクトル $\mathrm{a} \in \mathrm{V}_n$ に対してベクトルの系列 $\{\mathrm{a}, \mathbf{A}\mathrm{a}, \mathbf{A}^2\mathrm{a}\}, \cdots$ を定めると、a に対してある自然数 p ($p \leq n$ であることは明らか) が必ず存在し、$\{\mathrm{a}, \mathbf{A}\mathrm{a}, \cdots, \mathbf{A}^{p-1}\mathrm{a}\}$ は一次独立で、$\{\mathrm{a}, \mathbf{A}\mathrm{a}, \cdots, \mathbf{A}^{p-1}\mathrm{a}, \mathbf{A}^p\mathrm{a}\}$ は一次従属になる。このとき、$\{\mathrm{a}, \mathbf{A}\mathrm{a}, \cdots, \mathbf{A}^{p-1}\mathrm{a}\}$ が張る p 次元の部分空間を a によって生成される巡回部分空間 $\mathrm{C}(\mathrm{a})$ という。

定義 9 (最小消去多項式) V_n の上の一次変換 $\mathbf{A}$ が与えられたとき、ベクトル $\mathrm{a} \in \mathrm{V}_n$ に対して $f(\mathbf{A})\mathrm{a} = \mathbf{0}$ (零ベクトル) を満足する多項式 $f(\lambda)$ は a を消去するという。a を消去する最小次数の多項式 (最高次の係数は 1 とする) を a の最小消去多項式という。 与えられたベクトル $\mathrm{a} \in \mathrm{V}_n$ の最小消去多項式 $f(\lambda)$ は次のようにして定められる。まず、列 $\{\mathrm{a}, \mathbf{A}\mathrm{a}, \mathbf{A}^2\mathrm{a}\}, \cdots$ を作り、$\{\mathrm{a}, \mathbf{A}\mathrm{a}, \cdots, \mathbf{A}^{p-1}\mathrm{a}\}$ は一次独立であるが $\mathbf{A}^p\mathrm{a}$ はそれらに一次従属になるような p を求める。そして

$$\mathbf{A}^p\mathrm{a} = c_0\mathrm{a} + c_1\mathbf{A}\mathrm{a} + \cdots + c_{p-1}\mathbf{A}^{p-1}\mathrm{a} \tag{2.7}$$

となる $c_0, \cdots, c_{p-1}$ を定め、$f(\lambda) = \lambda^p - c_{p-1}\lambda^{p-1} - \cdots - c_1\lambda - c_0$ とおけば、$f(\lambda)$ は a の最小消去多項式である。($f(\mathbf{A})\mathrm{a} = \mathbf{0}$ であることは明らかであり、逆に q 次 $(q < p)$ の多項式 $g(\lambda)$ に対して $g(\mathbf{A})\mathrm{a} = \mathbf{0}$ になったとすると、$\{\mathrm{a}, \mathbf{A}\mathrm{a}, \cdots, \mathbf{A}^q\mathrm{a}\}$ が一次従属になってしまう。) また、この構成法は、「a の最小消去多項式の次数は a が生成する巡回部分空間 $\mathrm{C}(\mathrm{a})$ の次元に等しい」ことを示している。

定義 10 (コンパニオン行列) ベクトル空間 V_n の中の一次変換 $\mathbf{A}(\in \mathrm{V}_n)$ の最小多項式を $f(\lambda) = \lambda^p - a_{p-1}\lambda^{p-1} - \cdots - a_1\lambda - a_0$ とすれば、$\mathrm{B} = \{\mathrm{a}, \mathbf{A}\mathrm{a}, \cdots, \mathbf{A}^{p-1}\mathrm{a}\}$ は a が生成する巡回部分空間 $\mathrm{C}(\mathrm{a})$ の基底となる。このとき、

$$\left.\begin{aligned} \mathrm{u}_1 &= \mathrm{a} \\ \mathrm{u}_{i+1} &= \mathbf{A}\mathrm{u}_i \ \ (i = 1, \cdots, p-1) \end{aligned}\right\} \tag{2.8}$$

とおけば

$$\mathbf{A}\mathrm{u}_p = a_0\mathrm{u}_1 + \cdots + a_1\mathrm{u}_2 + \cdots + a_{p-1}\mathrm{u}_p \tag{2.9}$$

である。(2.8),(2.9) をまとめて、行列の形で書けば

$$[\mathbf{A}\mathrm{u}_1, \cdots, \mathbf{A}\mathrm{u}_p] = [\mathrm{u}_1, \cdots, \mathrm{u}_p]\,A, \tag{2.10}$$

ここで、右辺の行列 A は

$$A = \begin{bmatrix} 0 & 0 & \cdots & 0 & a_0 \\ 1 & 0 & \cdots & 0 & a_1 \\ & 1 & \ddots & \vdots & \vdots \\ & & & 0 & a_{p-2} \\ O & & & 1 & a_{p-1} \end{bmatrix} \tag{2.11}$$

となる。この p 次正方行列 A は、一次変換 $\mathbf{A}$ によって C(a) の上に誘導された一次変換 $\mathbf{A_p}$ の基底 B に関する表現行列であり、多項式 $f(\lambda) = \lambda^p - a_{p-1}\lambda^{p-1} - \cdots - a_1\lambda - a_0$ に随伴するコンパニオン行列という。特に、$f(\lambda) = \lambda - a_0$ のコンパニオン行列は 1 次の行列 $[a_0]$ であるとする。p 次のコンパニオン行列 A の特性多項式 $\varphi_A(\lambda)$ と最小多項式 $\phi_A(\lambda)$ は $f(\lambda)$ に一致する。

定理 5 (有理標準形) 任意の n 次正方行列 A は、適当な正則行列 $S(\det S \neq 0)$ を用いると、相似変換 $S^{-1}AS$ によって

$$S^{-1}AS = \begin{bmatrix} C_1 & & & O \\ & C_2 & & \\ & & \ddots & \\ O & & & C_t \end{bmatrix} \tag{2.12}$$

という形のブロック対角行列に変換することができる。これを A の有理標準形（Frobenius 標準形、第 1 の標準形（First Natural Normal Form））という。ここで、各ブロック $C_i(i = 1, \cdots, t)$ は、p_i 次のコンパニオン行列

$$C_i = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 & a_0^{(i)} \\ 1 & 0 & \cdots & \cdots & 0 & a_1^{(i)} \\ & 1 & \ddots & & \vdots & \vdots \\ & & \ddots & \ddots & \vdots & \vdots \\ & O & & \ddots & 0 & a_{p_i-2}^{(i)} \\ & & & & 1 & a_{p_i-1}^{(i)} \end{bmatrix} \tag{2.13}$$

であり、C_{j+1} の最小多項式 $\phi_{C_{j+1}}(\lambda)$ は C_j の最小多項式 $\phi_{C_j}(\lambda)$ を割り切る $(j = 1, \cdots, t-1)$。(2.12),(2.13) の形の行列は、A によって一意的に定まる。また、相似な行列の有理標準形は一致し、逆に、有理標準形が一致する行列は互いに相似である。有理標準形は任意の体上の元を要素とする行列について存在する。

定理 6 (有理標準形と Jordan 標準形) n 次正方行列 A の有理標準形 $S^{-1}AS = C_1 \oplus \cdots \oplus C_t$ が (2.12), (2.13) のように定められたとする。コンパニオン行列 $C_1, \cdots, C_t$ の最小多項式を $\phi_{C_1}(\lambda), \cdots, \phi_{C_t}(\lambda)$

とし、その次数を $m_1, \cdots, m_t$ とする。まず、最初のコンパニオン行列 C_1 に対して、C_1 のすべての相異なる固有値を $\lambda_1, \cdots, \lambda_l$ として $\phi_{C_1}(\lambda) = \lambda^{m_1} - a_{m_1-1}\lambda^{m_1-1} - \cdots - a_1\lambda - a_0$ を

$$\phi_{C_1}(\lambda) = (\lambda - \lambda_1)^{q_1} \cdots (\lambda - \lambda_l)^{q_l} (q_1 + \cdots + q_l = m_1) \tag{2.14}$$

と $\phi_{C_1}(\lambda)$ の分解体上で 1 次因子の積に分解する。ただし、q_j は λ_j の重複度である。次に、固有値 λ_1 に対し、$m_1 \times q_1$ 行列 $U_1^{(1)}$ を

$$u_{ij} = \begin{pmatrix} i-1 \\ q_1 - j \end{pmatrix} \lambda_1^{i+j-q_1-1} (i = 1, \cdots, m_1; j = 1, \cdots, q_1) \tag{2.15}$$

を (i, j) 要素とする行列として定義する。（$\begin{pmatrix} k \\ l \end{pmatrix}$ は二項係数であり、$\begin{pmatrix} k \\ l \end{pmatrix} = \frac{k!}{(k-l)!l!}$ である。ただし、$\begin{pmatrix} k \\ l \end{pmatrix}$ は $l > k$ または $l < 0$ のとき 0 であると約束する。また、ここでは 0^0（0 の 0 乗）は 1 であると定義する。）このとき、

$${C_1}^T {U_1}^{(1)} = {U_1}^{(1)} J^T(\lambda_1, q_1) \tag{2.16}$$

が成り立つ。ここで、${C_1}^T$ は C_1 の転置、$J^T(\lambda_1, q_1)$ は固有値 λ_1 に対する q_1 次の Jordan 細胞 $J(\lambda_1, q_1)$ の転置である。他の固有値 $\lambda_2, \cdots, \lambda_l$ に対しても同様のことを行なう。その結果、得られる $m_1 \times q_k$ 行列を $U_1^{(k)} (k = 2, \cdots, l)$ とすると、

$${C_1}^T {U_1}^{(k)} = {U_1}^{(k)} J^T(\lambda_k, q_k) \quad (k = 2, \cdots, l) \tag{2.17}$$

が成り立つ。式 (2.16),(2.17) をまとめると、

$$\left.\begin{aligned} {C_1}^T U_1 &= U_1 {J_1}^T \\ U_1 &= \left[U_1^{(1)}, \cdots, U_1^{(l)}\right] \\ {J_1}^T &= J^T(\lambda_1, q_1) \oplus \cdots \oplus J^T(\lambda_l, q_l) \end{aligned}\right\} \tag{2.18}$$

U_1 は $U_1^{(1)}, \cdots, U_1^{(l)}$ をこの順に並べて作った m_1 次の正方行列である。以上のような操作を他のコンパニオン行列 $C_i (i = 2, \cdots, t)$ に対しても行い、m_2 次行列 U_2, J_2 ; $\cdots$; m_t 次行列 U_t, J_t を作り、

$$U = U_1 \oplus U_2 \oplus \cdots \oplus U_t \tag{2.19}$$

とおく。すると、U は m_i 次の正方行列であり、

$$U^T \begin{bmatrix} C_1 & & & O \\ & C_2 & & \\ & & \ddots & \\ O & & & C_t \end{bmatrix} (U^T)^{-1} = \begin{bmatrix} J_1 & & & O \\ & J_2 & & \\ & & \ddots & \\ O & & & J_t \end{bmatrix} \tag{2.20}$$

となる。よって、有理標準形は変換行列 $U^T, (U^T)^{-1}$ によって、Jordan 標準形に変換することができる。

第 3 章

Jordan 標準形の計算法

3.1 定義

1 章で述べたように、数式処理を用いた従来の Jordan 標準形の計算法の問題点は、代数方程式の求解（行列の固有値の計算）にあった。本研究では、その問題を解決するために、「有理標準形に基づく方法」を単因子と有理標準形との関係を用いて改良する。

任意の行列の単因子と有理標準形との関係は、次の定理によって与えられる。

定理 7 (単因子と有理標準形) 任意の正方行列 A の特性行列 $\lambda E - A$ の単因子標準形を計算し、1 でない単因子を次数の低い順から $e_1(\lambda), \cdots, e_t(\lambda)$ とおく。このとき、A の有理標準形 R は次のような形の行列になる。

$$R = C_t \oplus C_{t-1} \oplus \cdots \oplus C_1 \tag{3.1}$$

ただし、各ブロック $C_i (i = 1, \cdots, t)$ は、単因子 $e_i(\lambda)$ に随伴するコンパニオン行列である。すなわち、$e_i(\lambda) = \det(\lambda E - C_i)$ が成り立つ。

「単因子に基づく方法」では、**2** 章の定義 7、定理 4より、複素数体（もとの行列の要素が 1 変数整数係数多項式の場合には代数関数体。以下、括弧内は 1 変数整数係数多項式の場合に対応する）上で単因子を単純単因子（具体的には、1 次因子のべき）に因数分解して、Jordan 標準形の中にある Jordan 細胞の次数と個数を求めている。すなわち、定義 7、定理 4は、全固有値が属する体（具体的には、複素数体（代数関数体））上の各単純単因子はそれぞれ 1 個の Jordan 細胞に対応しているということを意味している。以上のことを踏まえて、本研究では、代数方程式の求解を回避するために、有理数体（1 変数有理関数体）上の単純単因子と Jordan 細胞の関係を考える。

定義 7において、基礎体を有理数体 $\mathbf{Q}$ にとり、特性行列の 1 でない単因子の 1 つを因数分解すると、次のようになる。

$$e(\lambda) = p_1(\lambda)^{k_1} p_2(\lambda)^{k_2} \cdots p_s(\lambda)^{k_s} \quad p_j(\lambda) \in \mathbf{Q}[\lambda] \quad (\mathbf{R}(x)[\lambda]) \quad k_j \geq 1 \tag{3.2}$$

この中の各単純単因子 $p_j(\lambda)^{k_j}$ について、次の定理が成り立つ。

定理 8 (有理数体上の単純単因子と Jordan 細胞) 有理数体上の単純単因子 $p(\lambda)^k$ について、$p(\lambda) = 0$ の根を $\lambda_1, \lambda_2, \cdots, \lambda_d$ ($d = \deg p,\ \lambda_i \in \mathbf{C}$) とすると、この単純単因子に対応する Jordan 細胞は、次のようになる。($d = 1$ のときは $\lambda_1 \in \mathbf{Q}$ となる。) [82]

$$J(\lambda_1, k) \oplus J(\lambda_2, k) \oplus \cdots \oplus J(\lambda_d, k) \tag{3.3}$$

定理 8は、有理数体上の単純単因子 $p(\lambda)^k$ を複素数体上で因数分解すると、

$$p(\lambda)^k = \{(\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_d)\}^k = (\lambda - \lambda_1)^k (\lambda - \lambda_2)^k \cdots (\lambda - \lambda_d)^k \tag{3.4}$$

となり、定理 4より、複素数体上の各単純単因子 $(\lambda - \lambda_i)^k$ に対応する Jordan 細胞は $J(\lambda_i, k)$ となることから確かめられる。

本研究では、定理 8にしたがい、有理数体上の単純単因子 $p(\lambda)^k$ を定義多項式として、複素数体に属する固有値を記号 λ_i として表すことにより、2 次以上の代数方程式の求解を回避する。これにより、要素が有理数体 $\mathbf{Q}$ の元である任意の行列の Jordan 標準形の構造を代数的に厳密に計算することが可能になる。記号として表された固有値 λ_i は、必要に応じて、定義方程式 $p(\lambda) = 0$ を数値解析的に解いて具体的な値を計算すればよい。その場合にも、既に求められた Jordan 標準形の厳密な構造は固有値の数値的精度の影響を受けない。1 変数有理関数体上の単純単因子についても上記と同様に考える。

本研究では、もとの行列を Jordan 標準形に変換する変換行列とその逆行列も同時に計算する。有理標準形から Jordan 標準形への変換行列は、定理 6に基づいて、固有値と各単因子中の固有値の重複度から求めるため、記号を要素とする Jordan 標準形への変換行列の要素は固有値を表す記号の単項式になり、その逆行列の要素は複数の記号の多項式になる。つまり、変換行列の逆行列の計算は、記号を要素とする行列の逆行列の計算ということになる。

本研究では、文献 [60] のアルゴリズムに基づき、Vandermonde 行列の逆行列の性質を用いて有理標準形から Jordan 標準形への変換行列の逆行列を計算し、計算の途中で記号で表れた固有値の単項式を固有値の定義多項式を用いて M- 簡約することによって、行列の要素である多項式の次数の増大を防ぐ。

本研究のアルゴリズムで用いる簡単化法である M- 簡約は次のように定義される。

定義 11 (M- 簡約 (Reduction of Multiples)) 多項式 $f_2 = c_2 u_2 + r_2$ による f_1 の M- 簡約とは、f_1 の中に u_2 の倍数になっているべき積 u が含まれている (その係数を c とする) とき、

$$\tilde{f}_1 = f_1 - \frac{cu}{c_2 u_2} f_2 \tag{3.5}$$

により項 cu を消去して、より低順位の項で置き換えることをいう。

本研究のプログラムでは、M- 簡約を行なうために、REDUCE のユーザが開発した Gröbner 基底の計算に関するパッケージ中の関数 PREDUCE を用いている。PREDUCE では、ある多項式 f_1

を複数の多項式 $\{f_2, f_3, \cdots\}$ で簡約するとき、多項式のリスト $\{f_2, f_3, \cdots\}$ が Gröbner 基底でない場合には、その一意性は保証されず、結果 $\tilde{f}_1$ はリスト中の多項式の順序によって異なることがある。本研究のプログラムでは、行列の要素中の記号で表された固有値 λ_i のべきを定義多項式 $p(\lambda)$ に λ_i を代入した多項式よって M- 簡約するだけであり、共役関係にある他の固有値との関係は導入していないので、その順序は問題になることはない。

さらに、同じ定義多項式の根である固有値を表す記号に共役関係を導入し、固有値と定義多項式との間の「解と係数の関係」の関係式から生成されるイデアルの Gröbner 基底を用いて M- 簡約を行なえば、変換行列の要素中の記号の個数とべきを減らすことができる [69], [71]。すなわち、変換行列の要素中の、同じ定義多項式を持つ複数の固有値をそれぞれを表す記号 $\lambda_1, \lambda_2, \cdots$ を、1 つの記号 λ_1 の多項式として表すことができる。本研究の以前のアルゴリズム・プログラム [69] では、その方法を用いて簡単化を行なっていたが、却って要素の表現が複雑になることがあるので、本稿で計算に使用するアルゴリズム・プログラムからは除いてある。

以下に、記号で表現された根に共役関係を導入するために必要な定義・定理を簡単に述べておく。代数方程式の「解と係数の関係」は、次のように定義される。

定理 9 (解と係数の関係) 最高次の係数が 1 である任意の n 次方程式を $f(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0 = 0$ とし、その解を $\alpha_1, \alpha_2, \cdots, \alpha_n$ とすると、$f(x)$ の係数 $c_0, c_1, \cdots, c_{n-1}$ と解 $\alpha_1, \alpha_2, \cdots, \alpha_n$ の間には、次のような関係がある。

$$\left.\begin{array}{l} c_0 = (-1)^n \alpha_1 \alpha_2 \cdots \alpha_n \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ c_{n-1} = \alpha_1\alpha_2 + \alpha_1\alpha_3 + \cdots + \alpha_{n-1}\alpha_n \\ c_n = -(\alpha_1 + \alpha_2 + \cdots + \alpha_n) \end{array}\right\} \tag{3.6}$$

この関係を「解と係数の関係」という。

Gröbner 基底を計算するためには、記号のべき積間に順序を導入しなければならない。ここでは、固有値を表す記号のべき積間に、次のように定義される辞書式順序を導入する。

定義 12 (辞書式順序) 2 つのべき積の間の大小関係 $\succ$ を以下のように定義する。

$$x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n} \succ x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n} \Leftrightarrow a_i > b_i \ (1 \le \exists i \le n) \ \text{かつ} \ a_j = b_j \ (1 \le \exists j \le i) \tag{3.7}$$

また、Gröbner 基底は次のように定義される。

定義 13 (Gröbner 基底) イデアル $I = (f_1, f_2, \cdots, f_r) \subset \mathbf{Q}[x_1, x_2, \cdots, x_n]$ に対して $G = \{g_1, g_2, \cdots, g_s\}$ が以下の性質を満たすとき、G は I の Gröbner 基底 [13] であるという。

$$(g_1, g_2, \cdots, g_s) = I \tag{3.8}$$

$\exists f \in I$ について、f を G の要素で可能な限り M- 簡約した結果が $\tilde{f}$ であるとき、$\tilde{f} = 0$ である。

定理 10 (Gröbner 基底による M- 簡約の一意性) 任意の多項式 $f \in \mathbf{Q}[x_1, x_2, \cdots, x_n]$ について、f を Gröbner 基底 G の要素で可能な限り M- 簡約した結果を $\tilde{f}$ とすると、$\tilde{f}$ は G の要素の適用順序によらず一意的である。

定理 10から、M- 簡約後の逆行列の要素の一意性は保証される。

3.2 アルゴリズム

1変数整数係数多項式を要素とする任意の n 次正方行列を A とおき、2つの n 次単位行列を E_1, E_2 とおく。前述の定理に基づいて、有理標準形 R、A を有理標準形に変換する変換行列 T ($T^{-1}AT = R$) とその逆行列 T^{-1}、Jordan 標準形、及び、A を Jordan 標準形 J に変換する変換行列 S ($S^{-1}AS = J$) とその逆行列 S^{-1} を計算するアルゴリズムは次の手順となる。(ただし、手順は、数式処理言語 REDUCE で表現することを前提として記述する。)

入力：有理数（あるいは1変数整数係数多項式）を要素とする任意の行列 A

出力：有理標準形，有理標準形への変換行列と逆行列，単因子，有理数体（1変数有理関数体）上での単純単因子，Jordan 標準形，Jordan 標準形への変換行列と逆行列

(1) A を文献 [57] に示された方法（後述 付録 **A** 有理標準形の構成法）で有理標準形 R に変換する。同時に、A に施した変換を、行については E_1 に対して、列については E_2 に対して施し、A を R に変換する変換行列 T とその逆行列 T^{-1} ($T^{-1}AT = R$) を求める ($E_1 \to \cdots \to T$, $E_2 \to \cdots \to T^{-1}$)。$R$ はコンパニオン行列 $C_1, \cdots, C_t$ を要素とするブロック対角行列であり、各 C_i の次数を m_i とおく。

(2) 各コンパニオン行列の最小多項式を求めることにより、(1を除いた) 単因子のリストを得る。

$$\{e_1(\lambda), e_2(\lambda), \cdots, e_t(\lambda)\} \quad e_i(\lambda) = \phi_{C_{t-i+1}}(\lambda)$$

(3) REDUCE の組込関数 FACTORIZE を用いて、各単因子を因数分解して、有理数体（1変数有理関数体）上での単純単因子のリストを求める。

$$\{p_{ij}(\lambda)^{k_{ij}}\} \ (i = 1, \cdots, t; j = 1, \cdots, s_i) \quad p_{ij} \in \mathbf{Q}[\lambda] \ k_{ij} \geq 1$$

(4) 式 (3.3) により、各単純単因子に対応する Jordan 細胞を構成する。すなわち、$d = \deg p_{ij}$ とするとき、REDUCE の組込関数 MKID を用いて d 個の変数を生成（同一の p_{ij} に対しては同一の変数の組を使用）し、それらの変数によって Jordan 細胞を表現して、Jordan 標準形を構成する。生成した変数は定義多項式 p_{ij} とともに保存する。

(5) R の各コンパニオン行列 $C_i (i = 1, \cdots, t)$ について、次の操作を行なう。
たとえば、C_1 に対して

(i) (4) で生成した変数で表された固有値とその定義多項式を用いて、$\phi_{C_1}(\lambda)(= e_t(\lambda))$ に含まれるすべての相異なる固有値 $\lambda_h (h = 1, \cdots, l)$ とその重複度 q_h を求める。

(ii) 各固有値 $\lambda_h (h = 1, \cdots, l)$ に対して、次の操作を行なう。
まず、λ_1 に対して、$m_1 \times q_1$ 行列 $U_1^{(1)}$ を、λ_1 を代入した定義多項式 $p(\lambda_i)$ で M- 簡約

しながら、式 (2.15) に基づいて求める。他の固有値に対しても同様のことを行ない、その結果、得られる $m_1 \times q_k$ 行列を $U_1^{(k)}(k=2,\cdots,l)$ とする。そして $U_1^{(1)},\cdots,U_1^{(l)}$ をこの順に並べて作った m_1 次の正方行列を U_1 とする。
同様にして、各コンパニオン行列 $C_i(i=2,\cdots,t)$ に対しても m_i 次行列 U_i を作り、

$$U = U_1 \oplus U_2 \oplus \cdots \oplus U_t$$

とおく。REDUCE の組込み関数 TP を用いて U を転置し、U^T とする。

(6) (5) で求めた変換行列 U^T の逆行列 $(U^T)^{-}1$ を各固有値 λ_i を代入した定義多項式 $p(\lambda_i)$ で M-簡約しながら文献 [60] に基づいて計算する。

(7) $T, T^{-1}, U^T, (U^T)^{-1}$ から、$S(=T(U^T)^{-1})$ と $S^{-1}(=U^T T^{-1})$ を求める。

(8) 必要に応じて、$p_{ij}(\lambda)=0$ を数値解析的に解いて、各固有値を求める。

3.3 計算例

3.2 節のアルゴリズムにしたがって、次の行列 A の有理標準形 R、A から有理標準形 R への変換行列 T とその逆行列 T^{-1}、Jordan 標準形 J、A から Jordan 標準形 J への変換行列 S とその逆行列 S^{-1} を計算する。

$$A = \begin{bmatrix} 243 & -323 & 121 & -81 \\ 219 & -291 & 109 & -73 \\ 122 & -162 & 61 & -41 \\ 45 & -60 & 23 & -15 \end{bmatrix}$$

(1) A の有理標準形 R は次のようになる。

$$R = \begin{bmatrix} 0 & 0 & 0 & -4 \\ 1 & 0 & 0 & -4 \\ 0 & 1 & 0 & -5 \\ 0 & 0 & 1 & -2 \end{bmatrix}$$

この例では、有理数標準形 R はコンパニオン行列 C_1 1 個からなっている $(R = C_1)$ ので、$m_1 = 4$ である。また、A を R に変換する変換行列 T とその逆行列 $T^{-1}(T^{-1}AT = R)$ は次のようになる。

$$T = \begin{bmatrix} 1 & -\frac{77494}{57121} & \frac{28239}{57121} & \frac{7875}{57121} \\ 0 & \frac{625}{57121} & -\frac{5257}{57121} & \frac{12480}{57121} \\ 0 & \frac{475}{228484} & -\frac{8565}{228484} & \frac{20909}{228484} \\ 0 & \frac{1547}{228484} & -\frac{6249}{228484} & \frac{9413}{228484} \end{bmatrix},$$

$$T^{-1} = \begin{bmatrix} 1 & 243 & -571 & -17 \\ 0 & 219 & -499 & -53 \\ 0 & 122 & -235 & -125 \\ 0 & 45 & -74 & -50 \end{bmatrix}$$

(2) (1) から、単因子は次の 1 つだけである。

$$e_1(\lambda) = \phi_{C_1}(\lambda) = \det(\lambda E - C_1) = \lambda^4 + 2\lambda^3 + 5\lambda^2 + 4\lambda + 4$$

(3) $e_1(\lambda)$ を有理数体上で因数分解すると、次のようになる。

$$e_1(\lambda) = (\lambda^2 + \lambda + 2)^2$$

したがって、単純単因子は $(\lambda^2 + \lambda + 2)^2$ 1 つだけである。ここで、$p(\lambda) = \lambda^2 + \lambda + 2$ とおく。

(4) $d = \deg p(\lambda) = 2$ であるので、2 個の変数 z_{11}, z_{12} を生成する。

$$p(\lambda) = \lambda^2 + \lambda + 2 = (\lambda - z_{11})(\lambda - z_{12})$$

よって、Jordan 標準形 J は次のようになる。

$$J = J(z_{11}, 2) \oplus J(z_{12}, 2) = \left[\begin{array}{cc|cc} z_{11} & 1 & 0 & 0 \\ 0 & z_{11} & 0 & 0 \\ \hline 0 & 0 & z_{12} & 1 \\ 0 & 0 & 0 & z_{12} \end{array}\right]$$

(5) C_1 に対して、U_1 を構成する。

(4) より、$\phi_{C_1}(\lambda)$ に含まれるすべての相異なる固有値は $\lambda_1 = z_{11}, \lambda_2 = z_{12}$ であり、重複度はそれぞれ $q_1 = 2, q_2 = 2$ である。$\lambda_1 = z_{11}$ に対して、$m_1 \times q_1 = 4 \times 2$ 行列 ${U_1}^{(1)}$ は

$$U_1^{(1)} = \begin{bmatrix} 0 & 1 \\ 1 & z_{11} \\ 2z_{11} & {z_{11}}^2 \\ 3{z_{11}}^2 & 3{z_{11}}^3 \end{bmatrix}$$

である。$U_1^{(1)}$ を $p(z_{11}) = z_{11}^2 + z_{11} + 2$ を用いて M- 簡約すると、

$$U_1^{(1)} = \begin{bmatrix} 0 & 1 \\ 1 & z_{11} \\ 2z_{11} & -z_{11} - 2 \\ -3z_{11} - 6 & -z_{11} + 2 \end{bmatrix}$$

$\lambda_2 = z_{12}$ に対して、$m_1 \times q_2 = 4 \times 2$ 行列 $U_1^{(2)}$ は

$$U_1^{(2)} = \begin{bmatrix} 0 & 1 \\ 1 & z_{12} \\ 2z_{12} & {z_{12}}^2 \\ 3{z_{12}}^2 & 3{z_{12}}^3 \end{bmatrix}$$

である。$U_1^{(2)}$ を $p(z_{12}) = z_{12}^2 + z_{12} + 2$ を用いて M- 簡約すると、

$$U_1^{(2)} = \begin{bmatrix} 0 & 1 \\ 1 & z_{12} \\ 2z_{12} & -z_{12} - 2 \\ -3z_{12} - 6 & -z_{12} + 2 \end{bmatrix}$$

よって、U_1 は次のようになる。

$$U_1 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & z_{11} & 1 & z_{12} \\ 2z_{11} & -z_{11}-2 & 2z_{12} & -z_{12}-2 \\ -3z_{11}-6 & -z_{11}+2 & -3z_{12}-6 & -z_{12}+2 \end{bmatrix}$$

この例では、$U_1$1 個で、U になっている ($U = U_1$)。したがって、U の転置行列 U^T は次のようになる。

$$U^T = \begin{bmatrix} 0 & 1 & 2_{11} & -3z_{11}-6 \\ 1 & z_{11} & -z_{11}-2 & -z_{11}+2 \\ 0 & 1 & 2_{12} & -3z_{12}-6 \\ 1 & z_{12} & -z_{12}-2 & -z_{12}+2 \end{bmatrix}$$

(6) U^T の逆行列 $(U^T)^{-1}$ を計算し、$\{p(z_{11}), p(z_{12})\} = \{z_{11}^2 + z_{11} + 2, z_{12}^2 + z_{12} + 2\}$ を用いて M-簡約すると、

$$(U^T)^{-1} = \begin{bmatrix} -\frac{2z_{11}+2}{7} & -\frac{11z_{11}-19}{49} & -\frac{2z_{12}+2}{7} & -\frac{11z_{12}-19}{49} \\ -\frac{z_{11}+3}{7} & -\frac{24z_{11}+12}{49} & -\frac{z_{12}+3}{7} & -\frac{24z_{12}+12}{49} \\ -\frac{z_{11}+2}{7} & -\frac{6z_{11}+3}{49} & -\frac{z_{12}+2}{7} & -\frac{6z_{12}+3}{49} \\ -\frac{1}{7} & -\frac{4z_{11}+2}{49} & -\frac{1}{7} & -\frac{4z_{12}+2}{49} \end{bmatrix}$$

となる。

(7) $T, T^{-1}, U^T, (U^T)^{-1}$ から、S と S^{-1} を計算すると、

$$S = T(U^T)^{-1} = \begin{bmatrix} 0 & -\frac{3691z_{11}+6782}{228484} & \frac{1617z_{11}+16466}{228484} & \frac{13579z_{11}-6558}{228484} \\ 1 & \frac{z_{11}-644}{478} & -\frac{13z_{11}-246}{478} & \frac{41z_{11}-114}{478} \\ 0 & -\frac{3691z_{12}+6782}{228484} & \frac{1617z_{12}+16466}{228484} & \frac{13579z_{12}-6558}{228484} \\ 1 & \frac{z_{12}-644}{478} & -\frac{13z_{12}-246}{478} & \frac{41z_{12}-114}{478} \end{bmatrix},$$

$$S^{-1} = U^T T^{-1} = \begin{bmatrix} \frac{326z_{11}+428}{7} & -\frac{2349z_{11}+1150}{49} & \frac{326z_{12}+428}{7} & -\frac{2349z_{12}+1150}{49} \\ \frac{280z_{11}+394}{7} & -\frac{2050z_{11}+1025}{49} & \frac{280z_{12}+394}{7} & -\frac{2050z_{12}+1025}{49} \\ \frac{113z_{11}+229}{7} & -\frac{1018z_{11}+509}{49} & \frac{113z_{12}+229}{7} & -\frac{1018z_{12}+509}{49} \\ \frac{29z_{11}+63}{7} & -\frac{436z_{11}+218}{49} & \frac{29z_{12}+63}{7} & -\frac{436z_{12}+218}{49} \end{bmatrix}$$

(8) $p(\lambda) = \lambda^2 + \lambda + 2$ を解いて固有値 $\lambda_1 = z_{11}, \lambda_2 = z_{12}$ を求めると、次のようになる。

$$z_{11} = \frac{1 \pm \sqrt{7}i}{2}, \quad z_{12} = \frac{1 \mp \sqrt{7}i}{2} \quad \text{(複号同順)}$$

3.4 計算結果と考察

数式処理システム REDUCE (version3.5) 上に **3.2** 節のアルゴリズムをインプリメントし、以下に示す単因子 L_N $(N = 3, \cdots, 20)$ と Jordan 標準形 J_N $(N = 3, \cdots, 20)$ を持つ、4 種類の 3 次から 20 次までの行列について計算を行なった。具体的な行列と計算結果の一部は、付録 **C** に示す。

$p_1 = \lambda + x$, $p_2 = \lambda^2 + x\lambda + 2x$, $p_3 = \lambda^3 + x\lambda^2 + 2x\lambda + 3x$, $p_4 = \lambda^4 + x\lambda^3 + 2x\lambda^2 + 3x\lambda + 4x$, $p_5 = \lambda^5 + x\lambda^4 + 2x\lambda^3 + 3x\lambda^2 + 4x\lambda + 5$ とする。α_{21}, α_{22} は $p_2(\lambda) = 0$ の 2 根、$\alpha_{31}, \alpha_{32}, \alpha_{33}$ は $p_3(\lambda) = 0$ の 3 根、$\alpha_{41}, \alpha_{42}, \alpha_{43}, \alpha_{44}$ は $p_4(\lambda) = 0$ の 4 根、$\alpha_{51}, \alpha_{52}, \alpha_{53}, \alpha_{54}, \alpha_{55}$ は $p_5(\lambda) = 0$ の 5 根を表す。

[データ 3.1] 有理標準形がコンパニオン行列 1 個からなる場合 $(R = C_1)$
($p_1 \sim p_5$ において、$x = 1$ としたときの数値行列)

3 次 $L_3 = \{1, 1, p_2 p_1\}$,
$J_3 = J(\alpha_{21}, 1) \oplus J(\alpha_{22}, 1) \oplus J(-1, 1)$,

4 次 $L_4 = \{1, 1, 1, p_2^2\}$,
$J_4 = J(\alpha_{21}, 2) \oplus J(\alpha_{22}, 2)$,

5 次 $L_5 = \{1, 1, 1, 1, p_2^2 p_1\}$,
$J_5 = J(\alpha_{21}, 2) \oplus J(\alpha_{22}, 2) \oplus J(-1, 1)$,

6 次 $L_6 = \{1, 1, 1, 1, 1, p_3^2\}$,
$J_6 = J(\alpha_{31}, 2) \oplus J(\alpha_{32}, 2) \oplus J(\alpha_{33}, 2)$,

7 次 $L_7 = \{1, 1, 1, 1, 1, 1, p_3^2 p_1\}$,
$J_7 = J(\alpha_{31}, 2) \oplus J(\alpha_{32}, 2) \oplus J(\alpha_{33}, 2) \oplus J(-1, 1)$,

8 次 $L_8 = \{1, 1, 1, 1, 1, 1, p_4^2\}$,
$J_8 = J(\alpha_{41}, 2) \oplus J(\alpha_{42}, 2) \oplus J(\alpha_{43}, 2) \oplus J(\alpha_{44}, 2)$,

9 次 $L_9 = \{1, 1, 1, 1, 1, 1, 1, p_4^2 p_1\}$,
$J_9 = J(\alpha_{41}, 2) \oplus J(\alpha_{42}, 2) \oplus J(\alpha_{43}, 2) \oplus J(\alpha_{44}, 2) \oplus J(-1, 1)$,

10 次 $L_{10} = \{1, 1, 1, 1, 1, 1, 1, 1, 1, p_5^2\}$,
$J_{10} = J(\alpha_{51}, 2) \oplus J(\alpha_{52}, 2) \oplus J(\alpha_{53}, 2) \oplus J(\alpha_{54}, 2) \oplus J(\alpha_{55}, 2)$,

11 次 $L_{11} = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, p_5^2 p_1\}$,
$J_{11} = J(\alpha_{51}, 2) \oplus J(\alpha_{52}, 2) \oplus J(\alpha_{53}, 2) \oplus J(\alpha_{54}, 2) \oplus J(\alpha_{55}, 2) \oplus J(-1, 1)$,

12 次 $L_{12} = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, p_5^2 p_2\}$,
$J_{12} = J(\alpha_{51}, 2) \oplus J(\alpha_{52}, 2) \oplus J(\alpha_{53}, 2) \oplus J(\alpha_{54}, 2) \oplus J(\alpha_{55}, 2) \oplus J(\alpha_{21}, 1) \oplus J(\alpha_{22}, 1)$,

13 次 $L_{13} = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, p_5^2 p_3\}$,
$J_{13} = J(\alpha_{51}, 2) \oplus J(\alpha_{52}, 2) \oplus J(\alpha_{53}, 2) \oplus J(\alpha_{54}, 2) \oplus J(\alpha_{55}, 2)$
$\oplus J(\alpha_{31}, 1) \oplus J(\alpha_{32}, 1) \oplus J(\alpha_{33}, 1)$,

14 次 $L_{14} = \{1,1,1,1,1,1,1,1,1,1,1,1,1,p_5^2p_4\}$,

$J_{14} = J(\alpha_{51},2) \oplus J(\alpha_{52},2) \oplus J(\alpha_{53},2) \oplus J(\alpha_{54},2) \oplus J(\alpha_{55},2)$
$\oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1)$,

15 次 $L_{15} = \{1,1,1,1,1,1,1,1,1,1,1,1,1,1,p_5^3\}$,

$J_{15} = J(\alpha_{51},3) \oplus J(\alpha_{52},3) \oplus J(\alpha_{53},3) \oplus J(\alpha_{54},3) \oplus J(\alpha_{55},3)$,

16 次 $L_{16} = \{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,p_5^3p_1\}$,

$J_{16} = J(\alpha_{51},3) \oplus J(\alpha_{52},3) \oplus J(\alpha_{53},3) \oplus J(\alpha_{54},3) \oplus J(\alpha_{55},3) \oplus J(-1,1)$,

17 次 $L_{17} = \{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,p_5^3p_2\}$,

$J_{17} = J(\alpha_{51},3) \oplus J(\alpha_{52},3) \oplus J(\alpha_{53},3) \oplus J(\alpha_{54},3) \oplus J(\alpha_{55},3) \oplus J(\alpha_{21},1) \oplus J(\alpha_{22},1)$,

18 次 $L_{18} = \{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,p_5^3p_3\}$,

$J_{18} = J(\alpha_{51},3) \oplus J(\alpha_{52},3) \oplus J(\alpha_{53},3) \oplus J(\alpha_{54},3) \oplus J(\alpha_{55},3)$
$\oplus J(\alpha_{31},1) \oplus J(\alpha_{32},1) \oplus J(\alpha_{33},1)$,

19 次 $L_{19} = \{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,p_5^3p_4\}$,

$J_{19} = J(\alpha_{51},3) \oplus J(\alpha_{52},3) \oplus J(\alpha_{53},3) \oplus J(\alpha_{54},3) \oplus J(\alpha_{55},3)$
$\oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1)$,

20 次 $L_{20} = \{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,p_5^4\}$,

$J_{20} = J(\alpha_{51},4) \oplus J(\alpha_{52},4) \oplus J(\alpha_{53},4) \oplus J(\alpha_{54},4) \oplus J(\alpha_{55},4)$

[データ 3.2] 有理標準形がコンパニオン行列 1 個からなる場合 ($R = C_1$)
（データ 3.1 において、$x = x$、つまり x を変数としたときの 1 変数多項式行列）

[データ 3.3] 有理標準形がコンパニオン行列 3 個からなる場合 ($R = C_1 \oplus C_2 \oplus C_3$)
（$p_1 \sim p_5$ において、$x = 1$ としたときの数値行列）

3 次 $L_3 = \{p_1, p_1, p_1\}$,

$J_3 = J(-1,1) \oplus J(-1,1) \oplus J(-1,1)$,

4 次 $L_4 = \{p_1^2, p_1, p_1\}$,

$J_4 = J(-1,2) \oplus J(-1,1) \oplus J(-1,1)$,

5 次 $L_5 = \{p_1^2, p_1^2, p_1\}$,

$J_5 = J(-1,2) \oplus J(-1,2) \oplus J(-1,1)$,

6 次 $L_6 = \{p_2, p_2, p_2\}$,

$J_6 = J(\alpha_{21},1) \oplus J(\alpha_{22},1) \oplus J(\alpha_{21},1) \oplus J(\alpha_{22},1) \oplus J(\alpha_{21},1) \oplus J(\alpha_{22},1)$,

7 次 $L_7 = \{p_2p_1, p_2, p_2\}$,

$J_7 = J(\alpha_{21},1) \oplus J(\alpha_{22},1) \oplus J(\alpha_{21},1) \oplus J(\alpha_{22},1) \oplus J(\alpha_{21},1) \oplus J(\alpha_{22},1) \oplus J(-1,1)$,

8次 $L_8 = \{p_2p_1, p_2p_1, p_2\}$,

$$J_8 = J(\alpha_{21},1) \oplus J(\alpha_{22},1) \oplus J(\alpha_{21},1) \oplus J(\alpha_{22},1) \oplus J(\alpha_{21},1) \oplus J(\alpha_{22},1) \oplus J(-1,1) \oplus J(-1,1),$$

9次 $L_9 = \{p_3, p_3, p_3\}$,

$$J_9 = J(\alpha_{31},1) \oplus J(\alpha_{32},1) \oplus J(\alpha_{33},1) \oplus J(\alpha_{31},1) \oplus J(\alpha_{32},1) \oplus J(\alpha_{33},1) \oplus J(\alpha_{31},1) \oplus J(\alpha_{32},1) \oplus J(\alpha_{33},1),$$

10次 $L_{10} = \{p_3p_1, p_3, p_3\}$,

$$J_{10} = J(\alpha_{31},1) \oplus J(\alpha_{32},1) \oplus J(\alpha_{33},1) \oplus J(\alpha_{31},1) \oplus J(\alpha_{32},1) \oplus J(\alpha_{33},1) \oplus J(\alpha_{31},1) \oplus J(\alpha_{32},1) \oplus J(\alpha_{33},1) \oplus J(-1,1),$$

11次 $L_{11} = \{p_3p_1, p_3p_1, p_3\}$,

$$J_{11} = J(\alpha_{31},1) \oplus J(\alpha_{32},1) \oplus J(\alpha_{33},1) \oplus J(\alpha_{31},1) \oplus J(\alpha_{32},1) \oplus J(\alpha_{33},1) \oplus J(\alpha_{31},1) \oplus J(\alpha_{32},1) \oplus J(\alpha_{33},1) \oplus J(-1,1) \oplus J(-1,1),$$

12次 $L_{12} = \{p_4, p_4, p_4\}$,

$$J_{12} = J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1),$$

13次 $L_{13} = \{p_4p_1, p_4, p_4\}$,

$$J_{13} = J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(-1,1),$$

14次 $L_{14} = \{p_4p_1, p_4p_1, p_4\}$,

$$J_{14} = J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(-1,1) \oplus J(-1,1),$$

15次 $L_{15} = \{p_4p_1, p_4p_1, p_4p_1\}$,

$$J_{15} = J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(-1,1) \oplus J(-1,1) \oplus J(-1,1),$$

16次 $L_{16} = \{p_4p_1^2, p_4p_1, p_4p_1\}$,

$$J_{16} = J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(-1,2) \oplus J(-1,1) \oplus J(-1,1),$$

17次 $L_{17} = \{p_4p_1^2, p_4p_1^2, p_4p_1\}$,

$$J_{17} = J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(-1,2) \oplus J(-1,2) \oplus J(-1,1),$$

18 次 $L_{18} = \{p_4p_2, p_4p_2, p_4p_2\}$,

$$J_{18} = J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1)$$
$$\oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{21},1)$$
$$\oplus J(\alpha_{22},1) \oplus J(\alpha_{21},1) \oplus J(\alpha_{22},1),$$

19 次 $L_{19} = \{p_4p_2p_1, p_4p_2, p_4p_2\}$,

$$J_{19} = J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1)$$
$$\oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{21},1)$$
$$\oplus J(\alpha_{22},1) \oplus J(\alpha_{21},1) \oplus J(\alpha_{22},1) \oplus J(-1,1),$$

20 次 $L_{20} = \{p_4p_2p_1, p_4p_2p_1, p_4p_2\}$,

$$J_{20} = J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1)$$
$$\oplus J(\alpha_{44},1) \oplus J(\alpha_{41},1) \oplus J(\alpha_{42},1) \oplus J(\alpha_{43},1) \oplus J(\alpha_{44},1) \oplus J(\alpha_{21},1)$$
$$\oplus J(\alpha_{22},1) \oplus J(\alpha_{21},1) \oplus J(\alpha_{22},1) \oplus J(-1,1) \oplus J(-1,1)$$

[データ 3.4] 有理標準形がコンパニオン行列 3 個からなる場合 ($R = C_1 \oplus C_2 \oplus C_3$)
（データ 3.3 において、$x = x$、つまり x を変数としたときの 1 変数多項式行列）

実際の入力データ $A_N (N = 3, \cdots, 20)$ は、以下の手順で作成した。

(1) 結果として想定した単因子から、定理 7により、対応する有理標準形 C_N を作る。

(2) 相似変換用行列 $S_N = (s_{ij})$, $s_{ij} = \begin{cases} N+1-\max(i,j) & j \geq i-1 \\ 0 & j < i-1 \end{cases}$ を作る。

(3) C_N に相似変換を施して、$A_N = S_N C_N S_N^{-1}$ とする。（$\det S_N = 1$ なので、各 A_N の要素はデータ 3.1 と 3.3 ではすべて整数、データ 3.2 と 3.4 ではすべて 1 変数整数係数多項式である。）

SUN SPARC server10 (40SPECmark 相当以上、主記憶容量 128MB) 上の数式処理システム REDUCE (version3.5) 上で、32MB のメモリを使用してデータ 3.1 ～ 3.4 についてそれぞれ計算したときの CPU 時間（単位：ms（ミリ秒））を表 3.1～ 3.4に示す。数字は、**3.2** 節のアルゴリズムを関連のあるいくつかの部分に分けたとき、その部分を計算するのに要した CPU 時間である。各部分の番号と **3.2** 節のアルゴリズムの各ステップの番号との対応は、(i)-(1)、(ii)-(2)(3)、(iii)-(4)、(iv)-(5)(6)、(v)-(7)、(全)-(1) ～ (7)（すべての計算に要した CPU 時間）である。ただし、アルゴリズムのステップ (8)（有理数でない固有値の具体的な値の計算）は行なっていないので、その計算時間は含まない。表 3.1～ 3.4の CPU 時間をそれぞれ縦棒積上グラフとして、図 3.1～ 3.4に示す。グラフの縦軸は CPU 時間（単位：ms）であり、横軸は入力データの行列の次数である。ただし、次数による CPU 時間の差が大きく、同じスケールでは表示しにくいので、13 次以降は省略する。なお、データ 3.2 の 16 次以上の行列については、メモリ不足のため計算が行なえなかった。

表 3.1: Jordan 標準形の計算時間 (データ 3.1)

次数	(i)	(ii)	(iii)	(iv)	(v)	(全)
3	370	300	170	1190	110	2140
4	890	320	170	1700	120	3200
5	1890	500	220	3210	1441	7261
6	3750	520	290	6310	5321	16191
7	6940	750	370	8850	8100	25010
8	12320	730	460	14069	16580	44159
9	21150	1070	540	17840	23201	63801
10	34329	1030	590	26660	42131	104740
11	54660	1320	700	32850	55640	145170
12	87770	1540	789	41531	77150	208780
13	137020	1750	940	52950	106199	298859
14	210990	1980	1110	67190	147270	428540
15	314070	1940	1040	74520	260050	651620
16	464070	2480	1180	84550	332370	884650
17	685930	2770	1370	100790	417450	1208310
18	999580	3010	1489	120210	549741	1674030
19	1427790	3210	1600	143800	699040	2275440
20	2004630	3150	1670	168320	1168510	3346280

(単位： ms)

(i) (1) 有理標準形と有理標準形への変換行列を計算するのに要した CPU 時間

(ii) (2)(3) 有理標準形から単純単因子を計算するのに要した CPU 時間

(iii) (4) 単純単因子から Jordan 標準形を計算するのに要した CPU 時間

(iv) (5)(6) 有理標準形から Jordan 標準形への変換行列とその逆行列を計算するのに要した CPU 時間

(v) (7) 入力行列から Jordan 標準形への変換行列とその逆行列を計算するのに要した CPU 時間

(全) (1) ～ (7) すべての計算に要した CPU 時間

図 3.1: Jordan 標準形の計算時間のグラフ (データ 3.1)

表 3.2: Jordan 標準形の計算時間 (データ 3.2)

次数	(i)	(ii)	(iii)	(iv)	(v)	(全)
3	1550	430	180	1440	980	4580
4	9619	400	200	2421	3890	16530
5	36649	681	280	5760	19990	63360
6	66730	660	300	14440	67190	149320
7	217010	1020	390	22450	128159	369029
8	275870	951	460	47590	320070	644941
9	868309	1360	591	71809	622370	1564439
10	901000	1260	620	154890	1395550	2453320
11	2949660	1800	750	192620	2290700	5435530
12	5092981	2000	870	123930	2300190	7519971
13	8718990	2260	1000	250960	5489110	14462320
14	14079250	2420	1140	267279	7205460	21555549
15	22981490	2380	1100	3038040	39960970	65983980

(単位： ms)

16 次以上の行列については、メモリ不足のため計算が行なえなかった。

(i) (1) 有理標準形と有理標準形への変換行列を計算するのに要した CPU 時間

(ii) (2)(3) 有理標準形から単純単因子を計算するのに要した CPU 時間

(iii) (4) 単純単因子から Jordan 標準形を計算するのに要した CPU 時間

(iv) (5)(6) 有理標準形から Jordan 標準形への変換行列とその逆行列を計算するのに要した CPU 時間

(v) (7) 入力行列から Jordan 標準形への変換行列とその逆行列を計算するのに要した CPU 時間

(全) (1) ～ (7) すべての計算に要した CPU 時間

図 3.2: Jordan 標準形の計算時間のグラフ (データ 3.2)

表 3.3: Jordan 標準形の計算時間 (データ 3.3)

次数	(i)	(ii)	(iii)	(iv)	(v)	(全)
3	231	250	129	531	9	1150
4	600	340	161	750	20	1871
5	1460	390	180	939	40	3009
6	2230	450	370	1781	149	4980
7	4260	630	420	2390	340	8040
8	7130	800	500	2940	530	11900
9	9970	720	570	4210	1320	16790
10	16490	931	680	5460	2510	26071
11	26090	1100	790	6400	3970	38350
12	34960	1010	890	8680	6550	52090
13	50970	1260	1000	10420	9560	73210
14	73750	1460	1130	12420	12610	101370
15	94350	1660	1230	13800	15550	126590
16	131020	1780	1320	15730	20440	170290
17	182550	2040	1500	18310	25830	230230
18	234220	2140	1650	22050	34680	294740
19	315950	2400	1810	25030	42720	387910
20	422751	2720	1870	27720	54420	509481

(単位： ms)

(i) (1) 有理標準形と有理標準形への変換行列を計算するのに要した CPU 時間

(ii) (2)(3) 有理標準形から単純単因子を計算するのに要した CPU 時間

(iii) (4) 単純単因子から Jordan 標準形を計算するのに要した CPU 時間

(iv) (5)(6) 有理標準形から Jordan 標準形への変換行列とその逆行列を計算するのに要した CPU 時間

(v) (7) 入力行列から Jordan 標準形への変換行列とその逆行列を計算するのに要した CPU 時間

(全) (1) ～ (7) すべての計算に要した CPU 時間

図 3.3: Jordan 標準形の計算時間のグラフ (データ 3.3)

表 3.4: Jordan 標準形の計算時間 (データ 3.4)

次数	(i)	(ii)	(iii)	(iv)	(v)	(全)
3	271	280	140	640	10	1341
4	980	360	180	900	71	2491
5	7921	450	220	1180	410	10181
6	10590	540	369	2070	800	14369
7	41671	769	480	2790	3519	49229
8	102650	979	551	3630	7849	115659
9	108030	750	650	4610	7890	121930
10	251930	1110	760	6060	20160	280020
11	559440	1370	920	8000	41030	610760
12	466220	1080	950	9530	33119	510899
13	939351	1440	1090	12140	82890	1036911
14	2090289	1850	1330	15050	160650	2269169
15	3193710	2200	1440	17820	257400	3472570
16	5849300	2430	1400	23040	447350	6323520
17	12120990	2800	1600	28940	803850	12958180
18	9512710	2770	1800	28310	692650	10238240
19	17027991	3220	2020	35750	1213940	18282921
20	34397550	3520	2080	41050	2114680	36563300

(単位： ms)

(i) (1) 有理標準形と有理標準形への変換行列を計算するのに要した CPU 時間

(ii) (2)(3) 有理標準形から単純単因子を計算するのに要した CPU 時間

(iii) (4) 単純単因子から Jordan 標準形を計算するのに要した CPU 時間

(iv) (5)(6) 有理標準形から Jordan 標準形への変換行列とその逆行列を計算するのに要した CPU 時間

(v) (7) 入力行列から Jordan 標準形への変換行列とその逆行列を計算するのに要した CPU 時間

(全) (1) ～ (7) すべての計算に要した CPU 時間

図 3.4: Jordan 標準形の計算時間のグラフ (データ 3.4)

Jordan 標準形については、比較のため、同じデータを用いて、従来の 3 方法に基づくアルゴリズムを REDUCE 上にインプリメントしたプログラム [70] を実行してみた。その結果、本研究のプログラムでは、すべてのデータについて、実用的な時間で 4 つの標準形と Jordan 標準形への変換行列を計算することができたが、従来の Jordan 標準形の計算プログラムでは、いずれも、方程式を解いて固有値を具体的に代数的に計算しているため、3 次以上の既約多項式の根を固有値とするデータの行列に関しては、その固有値を用いたその後の処理が極めて非効率的になり、実際には計算ができなくなることが分かった。

また、文献 [70] のテストデータを用いて、本研究のプログラムを実行してみた。同文献で使用されているデータは、従来の 3 方法のプログラムで計算できるように、行列の固有値が整数か 2 次以下の方程式の根であるもののみとされ、問題となる 3 次以上の方程式の根である固有値をもつ行列は含まれていない。本研究のプログラムの Jordan 標準形についての CPU 時間は、「有理標準形に基づく方法」の CPU 時間と同程度であった。

図 3.1,3.2,3.3,3.4より、もとの行列から Jordan 標準形の計算に要する CPU 時間においては、単純単因子から Jordan 標準形を求めるのにかかる CPU 時間よりも、有理標準形の計算に要する CPU 時間の方が支配的であることが分かる。有理標準形の計算については、CPU 時間の増加は入力行列の次数の増加に必ずしも比例しない。表 3.4, 図 3.4 の 11 次と 12 次、17 次と 18 次から分かるように、次数の小さい行列の方が次数の大きい行列よりも CPU 時間が多くかかることもある。入力行列の次数 n に対して有理標準形の計算量は $O(n^4)$ とされている [79] が、もとの行列の要素の配列と有理標準形の中のコンパニオン行列との関係によって、その計算時間には不安定性がある [69]。本研究で用いた有理標準形の計算アルゴリズムでは、有理標準形の中のコンパニオン行列を 1 つ 1 つ計算していくので、もとの行列の要素の配置にもよるが、有理標準形の中に含まれるコンパニオン行列の個数が多い方が効率的に計算することができる。データ 3.1,3.2 ではコンパニオン行列の個数は 1 個であり、データ 3.3,3.4 では 3 個である。行列の要素が異なっているので一概には言えないが、図 3.1と図 3.3, 図 3.2と図 3.4から分かるように、同じ次数の行列でも、コンパニオン行列が 3 個である方が CPU 時間が短くなっている。このアルゴリズムは、文献 [57] のアルゴリズムにほぼしたがっている（記述上の不備を訂正した箇所がある。後述 付録 **A** 有理標準形の構成法）が、これを改良・効率化することができれば、さらに、各標準形の計算時間の短縮が可能である。そこで、本研究では、分数なし計算法による有理標準形の計算法の効率化を行なった（後述 **5** 章）。

Jordan 標準形への変換行列とその逆行列の計算は、従来の方法でも、本研究の方法でも、固有値の計算が必須である。したがって、従来の方法では、Jordan 標準形を計算するときと同じ問題（代数方程式の求解）があった。本研究では、固有値を記号として扱うことにより、その問題を解決した。しかし、そのため、**3.1** 節で述べたように、記号を要素とする行列の逆行列の計算という新たな問題が発生した。本研究では、逆行列の要素である多項式の次数と変数の個数を抑えるために、Gröbner 基底 [69] や固有値の定義多項式による M- 簡約を導入したが、数式処理システムでは記号処理のために大量のメモリを必要とするので、現在の計算機上で計算できる行列の次数は 20 次程度である（た

だし、もとの行列の性質にもよる）。しかし、理論上は次数による限界はない。また、アルゴリズム上では、逆行列の計算を速く行なうため、有理標準形の各コンパニオン行列に対する変換行列の各ブロックごとに逆行列を計算しているので、有理標準形が1個のコンパニオン行列から成っている場合よりも、複数のコンパニオン行列に分かれている場合の方が計算時間が短くなることがある。ここでのデータで、各表の (iv),(v) の列から分かるように、コンパニオン行列の個数が3個の方が変換行列の計算に要する CPU 時間は短くなっている。各図から分かるように、計算全体にかかった CPU 時間においては、入力行列から Jordan 標準形への変換行列の計算に要する CPU 時間も比較的大きな部分を占めているが、このステップ (7) で行なっているのは、実際には、2回の2つの行列の乗算だけであり、REDUCE の組込み機能に任せている。

3.5 Jordan 標準形の応用

本研究で扱った行列の標準形には、いくつかの応用がある。本章では、標準形の応用の中でも一般的な Jordan 標準形の応用 [57] への本研究のアルゴリズムの適用について述べる。

3.5.1 線形差分方程式の解法

3.5.1.1 定義

定義 14 (1 階の連立線形差分方程式) $x_1(m), x_2(m), \cdots, x_n(m)$ を n 個の数列とするとき、

$$\begin{cases} x_1(m+1) &= a_{11}x_1(m) + a_{12}x_2(m) + \cdots + a_{1n}x_n(m) \\ x_2(m+1) &= a_{21}x_1(m) + a_{22}x_2(m) + \cdots + a_{2n}x_n(m) \quad (m = 0, 1, 2, \cdots) \\ & \cdots\cdots\cdots\cdots \\ x_n(m+1) &= a_{n1}x_1(m) + a_{n2}x_2(m) + \cdots + a_{nn}x_n(m) \end{cases} \tag{3.9}$$

という形の方程式を 1 階の連立線形差分方程式という。方程式 (3.9) は、右辺の係数 a_{ij} を (i, j) 要素とする n 次正方行列を A とし、$\mathbf{x}(m) = [x_1(m), \cdots, x_n(m)]^T$ とおくと

$$\mathbf{x}(m+1) = A\mathbf{x}(m) \ \ (m = 0, 1, 2, \cdots) \tag{3.10}$$

と表せる。このとき、初期値もベクトル $\mathbf{x}(0)$ の形にまとめられる。

定理 11 (1 階の連立線形差分方程式の解) 1 階の連立線形差分方程式 (3.10) の解は、初期値を $\mathbf{x}(0)$ とするとき

$$\mathbf{x}(m) = A^m\mathbf{x}(0) \ \ (m = 0, 1, 2, \cdots) \tag{3.11}$$

で与えられる。

したがって、式 (3.10) の形の線形差分方程式を解くためには、係数行列のべき A^m を計算すればよい。任意の行列のべきと Jordan 標準形との関係は、次の 2 つの定理によって与えられる。

定理 12 (Jordan 細胞のべき) p 次正方行列 N を

$$N = \begin{bmatrix} 0 & 1 & & & \\ & 0 & \ddots & & O \\ & & \ddots & \ddots & \\ O & & & \ddots & 1 \\ & & & & 0 \end{bmatrix} \tag{3.12}$$

とおくと、対角要素が α の p 次の Jordan 細胞 $J = J(\alpha, p)$ のべきは、

$$J^m = \sum_{i=0}^{m} \begin{pmatrix} m \\ i \end{pmatrix} \alpha^{m-i} N^i \tag{3.13}$$

と表せる（$m < p-1$ のとき $\begin{pmatrix} m \\ p-1 \end{pmatrix} = 0$ とみなす）。

定理 13 (行列のべき) 任意の n 次正方行列 A の Jordan 標準形を $S^{-1}AS = J_1 \oplus J_2 \oplus \cdots \oplus J_t$ とすると、$(S^{-1}AS)^m = S^{-1}A^mS$ であるので、

$$A^m = S \begin{bmatrix} J_1^m & & & O \\ & J_2^m & & \\ & & \ddots & \\ O & & & J_t^m \end{bmatrix} S^{-1} \tag{3.14}$$

である。

3.5.1.2 アルゴリズム

式 (3.10) の形の連立線形差分方程式の係数行列を A とおく。A の要素は、アルゴリズムの制約上、有理数であるとする。前述の定理に基づいて、係数行列のべき A^m を計算するアルゴリズムは次の手順となる。（ただし、手順は、数式処理言語 REDUCE で表現することを前提として記述する。）

入力： 有理数体 $\mathbf{Q}$ の元を要素とする任意の行列 A、指数 m $(m > 0)$

出力： A のべき A^m

[1] **3.2**節の Jordan 標準形の計算アルゴリズムのステップ (1)–(8) のうち、(1)–(7) を行ない、A のすべての相異なる固有値 λ_i $(i = 1, \cdots, p)$ とその定義多項式 $p_i(x)$、A の Jordan 標準形 J の中のすべての Jordan 細胞 $J_{ij} = J(\lambda_i, n_{ij})(i = 1, \cdots, p; j = 1, \cdots, q)$、$A$ から J への変換行列 S, S^{-1} を求める。

[2] 各 Jordan 細胞 $J_{ij} = J(\lambda_i, n_{ij})(i = 1, \cdots, p; j = 1, \cdots, q)$ について、べき ${J_{ij}}^m$ を求める。たとえば、$J_{11} = J(\lambda_1, n_{11})$ に対して、$B = {J_{11}}^m$ とすると、$n_{11} \times n_{11}$ 行列 B の (k, l) 要素 b_{kl} は次のように定められる。

$$b_{kl} = \begin{cases} \begin{pmatrix} m \\ l-k \end{pmatrix} {\lambda_1}^{m-(l-k)} & (k \leq l \text{ かつ } m-k+1 \geq l) \\ 0 & (\text{上記以外のとき}) \end{cases}$$

同様にして、各 Jordan 細胞のべき ${J_{ij}}^m$ を作り、それらを対角線上に並べて Jordan 標準形 J のべき J^m を構成する。

$$J^m = {J_{11}}^m \oplus {J_{12}}^m \oplus \cdots \oplus {J_{pq}}^m$$

[3] J^m に変換行列 S, S^{-1} をかけ、各固有値 λ_i を代入した定義多項式 $p_{ij}(\lambda_i)$ で M– 簡約して、A のべき A^m を求める。

$$A^m = SJ^mS^{-1}$$

[4] 必要に応じて、定義多項式 $p_i(x)=0$ を数値解析的に解いて各固有値 λ_i を求める。

3.5.1.3 計算例

3.5.1.2 項のアルゴリズムにしたがって、次の連立線形差分方程式の解を計算する。

$$\begin{bmatrix} x_1(m+1) \\ x_2(m+1) \\ x_3(m+1) \\ x_4(m+1) \end{bmatrix} = \begin{bmatrix} 243 & -323 & 121 & -81 \\ 219 & -291 & 109 & -73 \\ 122 & -162 & 61 & -41 \\ 45 & -60 & 23 & -15 \end{bmatrix} \begin{bmatrix} x_1(m) \\ x_2(m) \\ x_3(m) \\ x_4(m) \end{bmatrix}$$

右辺の係数行列を A とし、指数を m とする。

[1] A の Jordan 標準形 J は次のようになる。

$$J = J(z_{11},2) \oplus J(z_{12},2) = \left[\begin{array}{cc|cc} z_{11} & 1 & 0 & 0 \\ 0 & z_{11} & 0 & 0 \\ \hline 0 & 0 & z_{12} & 1 \\ 0 & 0 & 0 & z_{12} \end{array}\right]$$

$J_1 = J(z_{11},2), J_2 = J(z_{12},2)$ とおく。固有値 z_{11}, z_{12} は定義多項式 $p(\lambda) = \lambda^2+\lambda+2$ の 2 根である。A から Jordan 標準形 J への変換行列 S, S^{-1} は次のようになる。

$$S = T(U^T)^{-1} = \begin{bmatrix} 0 & -\frac{3691z_{11}+6782}{228484} & \frac{1617z_{11}+16466}{228484} & \frac{13579z_{11}-6558}{228484} \\ 1 & \frac{z_{11}-644}{478} & -\frac{13z_{11}-246}{478} & \frac{41z_{11}-114}{478} \\ 0 & -\frac{3691z_{12}+6782}{228484} & \frac{1617z_{12}+16466}{228484} & \frac{13579z_{12}-6558}{228484} \\ 1 & \frac{z_{12}-644}{478} & -\frac{13z_{12}-246}{478} & \frac{41z_{12}-114}{478} \end{bmatrix},$$

$$S^{-1} = U^T T^{-1} = \begin{bmatrix} \frac{326z_{11}+428}{7} & -\frac{2349z_{11}+1150}{49} & \frac{326z_{12}+428}{7} & -\frac{2349z_{12}+1150}{49} \\ \frac{280z_{11}+394}{7} & -\frac{2050z_{11}+1025}{49} & \frac{280z_{12}+394}{7} & -\frac{2050z_{12}+1025}{49} \\ \frac{113z_{11}+229}{7} & -\frac{1018z_{11}+509}{49} & \frac{113z_{12}+229}{7} & -\frac{1018z_{12}+509}{49} \\ \frac{29z_{11}+63}{7} & -\frac{436z_{11}+218}{49} & \frac{29z_{12}+63}{7} & -\frac{436z_{12}+218}{49} \end{bmatrix}$$

[2] [1] から、Jordan 細胞 J_1, J_2 のべき ${J_1}^m, {J_2}^m$ はそれぞれ次のようになる。

$${J_1}^m = \begin{bmatrix} {z_{11}}^m & m{z_{11}}^{m-1} \\ 0 & {z_{11}}^m \end{bmatrix}, \quad {J_2}^m = \begin{bmatrix} {z_{12}}^m & m{z_{12}}^{m-1} \\ 0 & {z_{12}}^m \end{bmatrix}$$

よって、Jordan 標準形のべき J^m は次のようになる。

$$J^m = \left[\begin{array}{cc|cc} {z_{11}}^m & m{z_{11}}^{m-1} & 0 & 0 \\ 0 & {z_{11}}^m & 0 & 0 \\ \hline 0 & 0 & {z_{12}}^m & m{z_{12}}^{m-1} \\ 0 & 0 & 0 & {z_{12}}^m \end{array}\right]$$

[3] A のべき A^m は

$$A^m = SJ^mS^{-1}$$

となる。ここで、定義多項式のリスト $\{z_{11}^2 + z_{11} + 2,\ z_{12}^2 + z_{12} + 2\}$ を用いて M- 簡約した A^m の (i,j) 要素を $\tilde{a}_{ij}$ で表すとすると、各要素 $\tilde{a}_{ij}(i,j = 1,2,3,4)$ は次のようになる。

$$\begin{aligned}
\tilde{a}_{11} &= (2282z_{11}^m mz_{11}z_{12}+2996z_{11}^m mz_{12}+1199z_{11}^m z_{11}z_{12}+4698z_{11}^m z_{12} \\
&\quad +2282z_{12}^m mz_{11}z_{12}+2996z_{12}^m mz_{11}+1199z_{12}^m z_{11}z_{12}+4698z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{12} &= (-3073z_{11}^m mz_{11}z_{12}-4046z_{11}^m mz_{12}-1544z_{11}^m z_{11}z_{12}-6176z_{11}^m z_{12} \\
&\quad -3073z_{12}^m mz_{11}z_{12}-4046z_{12}^m mz_{11}-1544z_{12}^m z_{11}z_{12}-6176z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{13} &= (1155z_{11}^m mz_{11}z_{12}+1666z_{11}^m mz_{12}+536z_{11}^m z_{11}z_{12}+2144z_{11}^m z_{12} \\
&\quad +1155z_{12}^m mz_{11}z_{12}+1666z_{12}^m mz_{11}+536z_{12}mz_{11}z_{12}+2144z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{14} &= (-483z_{11}^m mz_{11}z_{12}-1106z_{11}^m mz_{12}-320z_{11}^m z_{11}z_{12}-1280z_{11}^m z_{12} \\
&\quad -483z_{12}^m mz_{11}z_{12}-1106z_{12}^m mz_{11}-320z_{12}^m z_{11}z_{12}-1280z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{21} &= (1960z_{11}^m mz_{11}z_{12}+2758z_{11}^m mz_{12}+1025z_{11}^m z_{11}z_{12}+4100z_{11}^m z_{12} \\
&\quad +1960z_{12}^m mz_{11}z_{12}+2758z_{12}^m mz_{11}+1025z_{12}^m z_{11}z_{12}+4100z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{22} &= (-2639z_{11}^m mz_{11}z_{12}-3724z_{11}^m mz_{12}-1322z_{11}^m z_{11}z_{12}-5386z_{11}^m z_{12} \\
&\quad -2639z_{12}^m mz_{11}z_{12}-3724z_{12}^m mz_{11}-1322z_{12}^m z_{11}z_{12}-5386z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{23} &= (987z_{11}^m mz_{11}z_{12}+1526z_{11}^m mz_{12}+468z_{11}^m z_{11}z_{12}+1872z_{11}^m z_{12} \\
&\quad +987z_{12}^m mz_{11}z_{12}+1526z_{12}^m mz_{11}+468z_{12}^m z_{11}z_{12}+1872z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{24} &= (-399z_{11}^m mz_{11}z_{12}-994z_{11}^m mz_{12}-284z_{11}^m z_{11}z_{12}-1136z_{11}^m z_{12} \\
&\quad -399z_{12}^m mz_{11}z_{12}-994z_{12}^m mz_{11}-284z_{12}^m z_{11}z_{12}-1136z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{31} &= (791z_{11}^m mz_{11}z_{12}+1603z_{11}^m mz_{12}+509z_{11}^m z_{11}z_{12}+2036z_{11}^m z_{12} \\
&\quad +791z_{12}^m mz_{11}z_{12}+1603z_{12}^m mz_{11}+509z_{12}^m z_{11}z_{12}+2036z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{32} &= (-1064z_{11}^m mz_{11}z_{12}-2163z_{11}^m mz_{12}-668z_{11}^m z_{11}z_{12}-2672z_{11}^m z_{12} \\
&\quad -1064z_{12}^m mz_{11}z_{12}-2163z_{12}^m mz_{11}-668z_{12}^m z_{11}z_{12}-2672z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{33} &= (385z_{11}^m mz_{11}z_{12}+868z_{11}^m mz_{12}+262z_{11}^m z_{11}z_{12}+950z_{11}^m z_{12} \\
&\quad +385z_{12}^m mz_{11}z_{12}+868z_{12}^m mz_{11}+262z_{12}^m z_{11}z_{12}+950z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{34} &= (-119z_{11}^m mz_{11}z_{12}-518z_{11}^m mz_{12}-156z_{11}^m z_{11}z_{12}-624z_{11}^m z_{12} \\
&\quad -119z_{12}^m mz_{11}z_{12}-518z_{12}^m mz_{11}-156z_{12}^m z_{11}z_{12}-624z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{41} &= (203z_{11}^m mz_{11}z_{12}+441z_{11}^m mz_{12}+218z_{11}^m z_{11}z_{12}+872z_{11}^m z_{12} \\
&\quad +203z_{12}^m mz_{11}z_{12}+441z_{12}^m mz_{11}+218z_{12}^m z_{11}z_{12}+872z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{42} &= (-273z_{11}^m mz_{11}z_{12}-595z_{11}^m mz_{12}-289z_{11}^m z_{11}z_{12}-1156z_{11}^m z_{12} \\
&\quad -273z_{12}^m mz_{11}z_{12}-595z_{12}^m mz_{11}-289z_{12}^m z_{11}z_{12}-1156z_{12}^m z_{11})/49z_{11}z_{12}, \\
\tilde{a}_{43} &= (98z_{11}^m mz_{11}z_{12}+238z_{11}^m mz_{12}+107z_{11}^m z_{11}z_{12}+428z_{11}^m z_{12} \\
&\quad +98z_{12}^m mz_{11}z_{12}+238z_{12}^m mz_{11}+107z_{12}^m z_{11}z_{12}+428z_{12}^m z_{11})/49z_{11}z_{12},
\end{aligned}$$

$$\tilde{a}_{44} = (-28z_{11}^m m z_{11} z_{12} - 140 z_{11}^m m z_{12} - 41 z_{11}^m z_{11} z_{12} - 262 z_{11}^m z_{12}$$
$$-28 z_{12}^m m z_{11} z_{12} - 140 z_{12}^m m z_{11} - 41 z_{12}^m z_{11} z_{12} - 262 z_{12}^m z_{11})/49 z_{11} z_{12}$$

初期条件を $\mathbf{x}(0) = [x_1(0), x_2(0), x_3(0), x_4(0)]^T$ とすれば、上記の連立線形差分方程式の解 $\mathbf{x}(m)$ は

$$\mathbf{x}(m) = A^m \mathbf{x}(0)$$

で与えられる。

[4] $p(\lambda) = \lambda^2 + \lambda + 2$ を解いて固有値 $\lambda_1 = z_{11}, \lambda_2 = z_{12}$ を求めると、次のようになる。

$$z_{11} = \frac{1 \pm \sqrt{7}i}{2}, \quad z_{12} = \frac{1 \mp \sqrt{7}i}{2} \quad \text{（複号同順）}$$

3.5.2 定係数連立線形微分方程式の解法

3.5.2.1 定義

定義 15 (1 階の定係数連立線形微分方程式) 実変数 t の n 個の（複素数値）関数 $x_1(t), x_2(t), \cdots, x_n(t)$ に対する

$$\begin{cases} \frac{d}{dt}x_1(t) &= a_{11}x_1(t) + a_{12}x_2(t) + \cdots + a_{1n}x_n(t) \\ \frac{d}{dt}x_2(t) &= a_{21}x_1(t) + a_{22}x_2(t) + \cdots + a_{2n}x_n(t) \\ & \cdots\cdots\cdots\cdots \\ \frac{d}{dt}x_n(t) &= a_{n1}x_1(t) + a_{n2}x_2(t) + \cdots + a_{nn}x_n(t) \end{cases} \tag{3.15}$$

という形の方程式を 1 階の定係数連立線形微分方程式という。方程式 (3.15) は、右辺の係数 a_{ij} を (i,j) 要素とする n 次正方行列を A とし、$\mathbf{x}(t) = [x_1(t), \cdots, x_n(t)]^T$ とおくと

$$\frac{d}{dt}\mathbf{x}(t) = A\mathbf{x}(t) \tag{3.16}$$

と表せる。このとき、初期値もベクトル $\mathbf{x}(0)$ の形にまとめられる。

定義 16 (行列の指数関数) 任意の正方行列 A に対して、A の指数関数は次のように定義される。

$$e^A = \sum_{p=0}^{\infty} \frac{1}{p!} A^p \tag{3.17}$$

定理 14 (1 階の定係数連立線形微分方程式の解) 1 階の定係数連立線形微分方程式 (3.16) の解は、初期値を $\mathbf{x}(0)$ とするとき

$$\mathbf{x}(t) = e^{tA}\mathbf{x}(0) \tag{3.18}$$

で与えられる。

e^{tA} は原理的には定義 16の式 (3.17) より、

$$e^{tA} = \sum_{p=0}^{\infty} \frac{1}{p!} t^p A^p \tag{3.19}$$

と定められるが、無限和 $\sum_{i=0}^{\infty}$ が含まれているため、この定義は構成的ではない。そこで、行列の指数関数と Jordan 標準形との関係を利用して、e^{tA} を具体的に定義する。行列の指数関数と Jordan 標準形との関係は、次の 2 つの定理によって与えられる。

定理 15 (Jordan 細胞の指数関数) 対角要素が α の p 次の Jordan 細胞 $J(\alpha,p)$ と任意の実数 t について、$e^{tJ(\alpha,p)}$ は

$$e^{tJ(\alpha,p)} = e^{\alpha t}\sum_{i=0}^{p-1}\frac{1}{i!}t^iN^i = \begin{bmatrix} e^{\alpha t} & \frac{te^{\alpha t}}{1!} & \frac{t^2e^{\alpha t}}{2!} & \cdots & \cdots & \frac{t^{p-1}e^{\alpha t}}{(p-1)!} \\ & e^{\alpha t} & \frac{te^{\alpha t}}{1!} & \ddots & & \vdots \\ & & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & \ddots & \frac{t^2e^{\alpha t}}{2!} \\ & O & & & \ddots & \frac{te^{\alpha t}}{1!} \\ & & & & & e^{\alpha t} \end{bmatrix} \tag{3.20}$$

と表せる。ただし、N は、(3.12) の形の p 次正方行列である。

定理 16 (行列の指数関数と Jordan 細胞) 任意の n 次正方行列 A の Jordan 標準形を $S^{-1}AS = J_1 \oplus J_2 \oplus \cdots \oplus J_r$ $(J_i = J(\lambda_i, p_i))$ とすると、定義 16より

$$\begin{aligned} e^{S^{-1}tAS} &= \sum_{p=0}^{\infty}\frac{1}{p!}(S^{-1}tAS)^p \\ &= \sum_{p=0}^{\infty}\frac{1}{p!}S^{-1}(tA)^pS = S^{-1}\left(\sum_{p=0}^{\infty}\frac{1}{p!}(tA)^p\right)S \\ &= S^{-1}e^{tA}S \end{aligned} \tag{3.21}$$

である。さらに、任意の正方行列 B, C に対して

$$\begin{aligned} e^{B\oplus C} &= \sum_{p=0}^{\infty}\frac{1}{p!}(B\oplus C)^p = \sum_{p=0}^{\infty}\frac{1}{p!}(B^p \oplus C^p) \\ &= \left(\sum_{p=0}^{\infty}\frac{1}{p!}B^p\right) \oplus \left(\sum_{p=0}^{\infty}\frac{1}{p!}C^p\right) = e^B \oplus e^C \end{aligned} \tag{3.22}$$

であるから

$$\begin{aligned} e^{S^{-1}tAS} &= e^{(tJ_1\oplus\cdots\oplus tJ_r)} \\ &= e^{tJ_1}\oplus\cdots\oplus e^{tJ_r} \end{aligned} \tag{3.23}$$

である。

したがって、式 (3.16) の形の定係数連立線形微分方程式を解くためには、Jordan 標準形についての指数関数 $e^{S^{-1}tAS} = e^{tJ_1}\oplus\cdots\oplus e^{tJ_r}$ と変換行列 S, S^{-1} を用いて、係数行列の t についての指数関数 $e^{tA} = S(e^{tJ_1}\oplus\cdots\oplus e^{tJ_r})S^{-1}$ を計算すればよい。

3.5.2.2 アルゴリズム

式 (3.16) の形の定係数連立線形微分方程式の係数行列を A とおく。A の要素は、アルゴリズムの制約上、有理数であるとする。前述の定理に基づいて、指数関数 e^{tA} を計算するアルゴリズムは次の手順となる。（ただし、手順は、数式処理言語 REDUCE で表現することを前提として記述する。）

入力： 有理数体 $\mathbf{Q}$ の元を要素とする任意の行列 A

出力： A についての指数関数 e^{tA}

[1] **3.2**節の Jordan 標準形の計算アルゴリズムのステップ (1)–(8) のうち、(1)–(7) を行ない、A のすべての相異なる固有値 $\lambda_i (i = 1, \cdots, p)$ とその定義多項式 $p_i(x)$、A の Jordan 標準形 J の中のすべての Jordan 細胞 $J_{ij} = J(\lambda_i, n_{ij}) (i = 1, \cdots, p; j = 1, \cdots, q)$、$A$ から J への変換行列 S, S^{-1} を求める。

[2] 各 Jordan 細胞 $J_{ij} = J(\lambda_i, n_{ij}) (i = 1, \cdots, p; j = 1, \cdots, q)$ について、指数関数 $e^{tJ_{ij}}$ を求める。

たとえば、$J_{11} = J(\lambda_1, n_{11})$ に対して、$B = e^{tJ_{11}}$ とすると、$n_{11} \times n_{11}$ 行列 B の (k, l) 要素 b_{kl} は次のように定められる。

$$b_{ij} = \begin{cases} \frac{e^{\lambda_1} t^{l-k}}{(l-k)!} & (k \leq l) \\ 0 & (k > l) \end{cases}$$

同様にして、各 Jordan 細胞に対する指数関数 $e^{tJ_{ij}}$ を作り、それらを対角線上に並べて Jordan 標準形 $S^{-1}AS$ についての指数関数 $e^{S^{-1}tAS}$ を構成する。

$$e^{S^{-1}tAS} = e^{tJ_{11}} \oplus e^{tJ_{12}} \oplus \cdots \oplus e^{tJ_{pq}}$$

[3] $e^{S^{-1}tAS}$ に変換行列 S, S^{-1} をかけて、各固有値 λ_i を代入した定義多項式 $p_{ij}(\lambda_i)$ で M– 簡約して、A についての指数関数 e^{tA} を求める。

$$e^{tA} = Se^{S^{-1}tAS}S^{-1}$$

[4] 必要に応じて、定義多項式 $p_i(x) = 0$ を数値解析的に解いて各固有値 λ_i を求める。

3.5.2.3 計算例

3.5.2.2 項のアルゴリズムにしたがって、次の連立線形微分方程式の解を計算する。

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} = \begin{bmatrix} 243 & -323 & 121 & -81 \\ 219 & -291 & 109 & -73 \\ 122 & -162 & 61 & -41 \\ 45 & -60 & 23 & -15 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix}$$

右辺の係数行列を A とする。

[1] A の Jordan 標準形 J は次のようになる。

$$J = J(z_{11}, 2) \oplus J(z_{12}, 2) = \left[\begin{array}{cc|cc} z_{11} & 1 & 0 & 0 \\ 0 & z_{11} & 0 & 0 \\ \hline 0 & 0 & z_{12} & 1 \\ 0 & 0 & 0 & z_{12} \end{array}\right]$$

$J_1 = J(z_{11}, 2), J_2 = J(z_{12}, 2)$ とおく。固有値 z_{11}, z_{12} は定義多項式 $p(\lambda) = \lambda^2 + \lambda + 2$ の 2 根である。A から Jordan 標準形 J への変換行列 S, S^{-1} は次のようになる。

$$S = T(U^T)^{-1} = \begin{bmatrix} 0 & -\frac{3691z_{11}+6782}{228484} & \frac{1617z_{11}+16466}{228484} & \frac{13579z_{11}-6558}{228484} \\ 1 & \frac{z_{11}-644}{478} & -\frac{13z_{11}-246}{478} & \frac{41z_{11}-114}{478} \\ 0 & -\frac{3691z_{12}+6782}{228484} & \frac{1617z_{12}+16466}{228484} & \frac{13579z_{12}-6558}{228484} \\ 1 & \frac{z_{12}-644}{478} & -\frac{13z_{12}-246}{478} & \frac{41z_{12}-114}{478} \end{bmatrix},$$

$$S^{-1} = U^T T^{-1} = \begin{bmatrix} \frac{326z_{11}+428}{7} & -\frac{2349z_{11}+1150}{49} & \frac{326z_{12}+428}{7} & -\frac{2349z_{12}+1150}{49} \\ \frac{280z_{11}+394}{7} & -\frac{2050z_{11}+1025}{49} & \frac{280z_{12}+394}{7} & -\frac{2050z_{12}+1025}{49} \\ \frac{113z_{11}+229}{7} & -\frac{1018z_{11}+509}{49} & \frac{113z_{12}+229}{7} & -\frac{1018z_{12}+509}{49} \\ \frac{29z_{11}+63}{7} & -\frac{436z_{11}+218}{49} & \frac{29z_{12}+63}{7} & -\frac{436z_{12}+218}{49} \end{bmatrix}$$

[2] [1] から、Jordan 細胞 J_1, J_2 についての指数関数 e^{tJ_1}, e^{tJ_2} はそれぞれ次のようになる。

$$e^{tJ_1} = \begin{bmatrix} e^{tz_{11}} & te^{tz_{11}} \\ 0 & e^{tz_{11}} \end{bmatrix}, \quad e^{tJ_2} = \begin{bmatrix} e^{tz_{12}} & te^{tz_{12}} \\ 0 & e^{tz_{12}} \end{bmatrix}$$

よって、Jordan 標準形 $J = S^{-1}AS$ についての指数関数 $e^{S^{-1}tAS}$ は次のようになる。

$$e^{S^{-1}tAS} = \left[\begin{array}{cc|cc} e^{tz_{11}} & te^{tz_{11}} & 0 & 0 \\ 0 & e^{tz_{11}} & 0 & 0 \\ \hline 0 & 0 & e^{tz_{12}} & te^{tz_{12}} \\ 0 & 0 & 0 & e^{tz_{12}} \end{array}\right]$$

[3] A についての指数関数 e^{tA} は

$$e^{tA} = Se^{S^{-1}AS}S^{-1}$$

となる。ここで、定義多項式のリスト $\{z_{11}^2+z_{11}+2,\ z_{12}^2+z_{12}+2\}$ を用いて M- 簡約した A^m の (i,j) 要素を $\tilde{a}_{ij}$ で表すとすると、各要素 $\tilde{a}_{ij}(i,j=1,2,3,4)$ は次のようになる。

$$
\begin{aligned}
\tilde{a}_{11} &= (2282e^{tz_{11}}tz_{11}+2996e^{tz_{11}}t-2349e^{tz_{11}}z_{11}-1150e^{tz_{11}}+2282e^{tz_{12}}tz_{12}+2996e^{tz_{12}}t-2349e^{tz_{12}}z_{12} \\
&\quad -1150e^{tz_{12}})/49, \\
\tilde{a}_{12} &= (-3073e^{tz_{11}}tz_{11}-4046e^{tz_{11}}t+3088e^{tz_{11}}z_{11}+1544e^{tz_{11}}-3073e^{tz_{12}}tz_{12}-4046e^{tz_{12}}t+3088e^{tz_{12}}z_{12} \\
&\quad +1544e^{tz_{12}})/49, \\
\tilde{a}_{13} &= (1155e^{tz_{11}}tz_{11}+1666e^{tz_{11}}t-1072e^{tz_{11}}z_{11}-536e^{tz_{11}}+1155e^{tz_{12}}tz_{12}+1666e^{tz_{12}}t-1072e^{tz_{12}}z_{12} \\
&\quad -536e^{tz_{12}})/49, \\
\tilde{a}_{14} &= (-483e^{tz_{11}}tz_{11}-1106e^{tz_{11}}t+640e^{tz_{11}}z_{11}+320e^{tz_{11}}-483e^{tz_{12}}tz_{12}-1106e^{tz_{12}}t+640e^{tz_{12}}z_{12}+320e^{tz_{12}})/49, \\
n\tilde{a}_{21} &= (1960e^{tz_{11}}tz_{11}+2758e^{tz_{11}}t-2050e^{tz_{11}}z_{11}-1025e^{tz_{11}}+1960e^{tz_{12}}tz_{12}+2758e^{tz_{12}}t-2050e^{tz_{12}}z_{12} \\
&\quad -1025e^{tz_{12}})/49, \\
\tilde{a}_{22} &= (-2639e^{tz_{11}}tz_{11}-3724e^{tz_{11}}t+2693e^{tz_{11}}z_{11}+1371e^{tz_{11}}-2639e^{tz_{12}}tz_{12}-3724e^{tz_{12}}t+2693e^{tz_{12}}z_{12} \\
&\quad +1371e^{tz_{12}})/49, \\
\tilde{a}_{23} &= (987e^{tz_{11}}tz_{11}+1526e^{tz_{11}}t-936e^{tz_{11}}z_{11}-468e^{tz_{11}}+987e^{tz_{12}}tz_{12}+1526e^{tz_{12}}t-936e^{tz_{12}}z_{12}-468e^{tz_{12}})/49, \\
\tilde{a}_{24} &= (-399e^{tz_{11}}tz_{11}-994e^{tz_{11}}t+568e^{tz_{11}}z_{11}+284e^{tz_{11}}-399e^{tz_{12}}tz_{12}-994e^{tz_{12}}t+568e^{tz_{12}}z_{12}+284e^{tz_{12}})/49, \\
\tilde{a}_{31} &= (791e^{tz_{11}}tz_{11}+1603e^{tz_{11}}t-1018e^{tz_{11}}z_{11}-509e^{tz_{11}}+791e^{tz_{12}}tz_{12}+1603e^{tz_{12}}t-1018e^{tz_{12}}z_{12}-509e^{tz_{12}})/49, \\
\tilde{a}_{32} &= (-1064e^{tz_{11}}tz_{11}-2163e^{tz_{11}}t+1336e^{tz_{11}}z_{11}+668e^{tz_{11}}-1064e^{tz_{12}}tz_{12}-2163e^{tz_{12}}t+1336e^{tz_{12}}z_{12} \\
&\quad +668e^{tz_{12}})/49, \\
\tilde{a}_{33} &= (385e^{tz_{11}}tz_{11}+868e^{tz_{11}}t-475e^{tz_{11}}z_{11}-213e^{tz_{11}}+385e^{tz_{12}}tz_{12}+868e^{tz_{12}}t-475e^{tz_{12}}z_{12}-213e^{tz_{12}})/49, \\
\tilde{a}_{34} &= (-119e^{tz_{11}}tz_{11}-518e^{tz_{11}}t+312e^{tz_{11}}z_{11}+156e^{tz_{11}}-119e^{tz_{12}}tz_{12}-518e^{tz_{12}}t+312e^{tz_{12}}z_{12}+156e^{tz_{12}})/49, \\
\tilde{a}_{41} &= (203e^{tz_{11}}tz_{11}+441e^{tz_{11}}t-436e^{tz_{11}}z_{11}-218e^{tz_{11}}+203e^{tz_{12}}tz_{12}+441e^{tz_{12}}t-436e^{tz_{12}}z_{12}-218e^{tz_{12}})/49, \\
\tilde{a}_{42} &= (-273e^{tz_{11}}tz_{11}-595e^{tz_{11}}t+578e^{tz_{11}}z_{11}+289e^{tz_{11}}-273e^{tz_{12}}tz_{12}-595e^{tz_{12}}t+578e^{tz_{12}}z_{12}+289e^{tz_{12}})/49, \\
\tilde{a}_{43} &= (98e^{tz_{11}}tz_{11}+238e^{tz_{11}}t-214e^{tz_{11}}z_{11}-107e^{tz_{11}}+98e^{tz_{12}}tz_{12}+238e^{tz_{12}}t-214e^{tz_{12}}z_{12}-107e^{tz_{12}})/49, \\
\tilde{a}_{44} &= (-28e^{tz_{11}}tz_{11}-140e^{tz_{11}}t+131e^{tz_{11}}z_{11}+90e^{tz_{11}}-28e^{tz_{12}}tz_{12}-140e^{tz_{12}}t+131e^{tz_{12}}z_{12}+90e^{tz_{12}})/49
\end{aligned}
$$

したがって、初期条件を $\mathbf{x}(0)=[x_1(0),x_2(0),x_3(0),x_4(0)]^T$ とすれば、上記の連立線形微分方程式の解 $\mathbf{x}(t)$ は

$$\mathbf{x}(t)=e^{tA}\mathbf{x}(0)$$

で与えられる。

[4] $p(\lambda)=\lambda^2+\lambda+2$ を解いて固有値 $\lambda_1=z_{11},\lambda_2=z_{12}$ を求めると、次のようになる。

$$z_{11}=\frac{1\pm\sqrt{7}i}{2},\ z_{12}=\frac{1\mp\sqrt{7}i}{2}\quad(\text{複号同順})$$

3.5.3 計算結果と考察

数式処理システム REDUCE (version3.5) 上に、**3.5.1.2** 項、**3.5.2.2** 項のアルゴリズムをそれぞれインプリメントし、**3.4**節に示したデータ 3.1,3.3 の一部（2 種類の 3 次から 10 次までの行列）について計算を行なった。ただし、各アルゴリズムに対して、データの行列は連立方程式の係数行列であるとし、差分方程式の解の係数行列である行列のべきの指数は、係数行列の一般形を求めるという意味から、m とする。

SUN SPARC server10 (40SPECmark 相当以上、主記憶容量 128MB) 上の数式処理システム REDUCE (version3.5) 上で、32MB のメモリを使用してデータ 3.1,3.3 についてそれぞれ計算したときの CPU 時間（単位： ms（ミリ秒））を表 3.5,3.6 に示す。数字は各アルゴリズムのステップ [1] にあたる処理、すなわち、データの行列の Jordan 標準形、固有値、固有値の定義多項式、Jordan 標準形への変換行列とその逆行列の計算が終了したところから、それぞれ、差分方程式の解の係数行列（以下、「差分方程式」と約す）、微分方程式の解の係数行列（以下、「微分方程式」と約す）を計算するまでに要した CPU 時間である。つまり、データの Jordan 標準形が前もって与えられているときの CPU 時間である。なお、データ 3.1 の 8 次以上の行列については、メモリ不足のため計算が行なえなかった。表 3.5,3.6 の CPU 時間をグラフとして、図 3.5,3.6に示す。グラフの縦軸は CPU 時間（単位： ms）であり、横軸は入力データの行列の次数である。

表 3.5: 応用の CPU 時間 (データ 3.1)

次数	差分方程式	微分方程式
3	1920	2669
4	3481	3990
5	25500	23441
6	2498540	2309160
7	2100859	1736840

（単位： ms）

8 次以上の行列については、メモリ不足のため計算が行なえなかった。

表 3.6: 応用の CPU 時間 (データ 3.3)

次数	差分方程式	微分方程式
3	410	1141
4	721	2151
5	1121	3781
6	31019	49460
7	54730	83110
8	40740	55920
9	1506601	2169439
10	1169059	1211050

（単位： ms）

図 3.5: 応用の CPU 時間のグラフ (データ 3.1)

図 3.6: 応用の CPU 時間のグラフ (データ 3.3)

図 3.5,3.6から分かるように、差分方程式の CPU 時間のグラフと微分方程式の CPU 時間のグラフは、ほぼ同じ傾向を示している。これは、各 Jordan 細胞に対応する小行列を構成し、その直和を解の係数行列とするというアルゴリズム上の類似によるものである。全体の CPU 時間には、この係数行列に含まれる小行列の個数（すなわち、対応する Jordan 細胞の個数）と記号で表された固有値の定義多項式の次数が関係している。解の係数行列に含まれる小行列の個数と固有値の定義多項式の最高次数を表 3.7,3.8に示す。

表 3.7: 小行列の個数と定義多項式の次数 (データ 3.1)

行列の次数	3	4	5	6	7	8	9	10
小行列の個数	3	2	3	3	4	4	5	5
定義多項式の次数	2	2	2	3	3	4	4	5

表 3.8: 小行列の個数と定義多項式の次数 (データ 3.3)

行列の次数	3	4	5	6	7	8	9	10
小行列の個数	3	3	3	6	7	8	9	10
定義多項式の次数	1	1	1	2	2	2	3	3

データ 3.1 では、表 3.5と図 3.5から分かるように、行列の次数が 6 次のときに CPU 時間が急激に増加し、7 次で少し減少している。これは、表 3.7に示したように、6 次のときには、M- 簡約に用いられる定義多項式の次数が 1 つ増えていることに加えて、（ここで具体的に要素を示すことはしないが）6 次の Jordan 標準形への変換行列が 7 次の変換行列よりも要素の係数の桁数が多く、また、複雑であるからである。8 次の行列に対してメモリ不足で計算が行なえなくなるのも、定義多項式の次数の増加の影響であると考えられる。また、データ 3.3 でも、表 3.6 と図 3.6から分かるように、行列の次数が 9 次のときに CPU 時間が急激に増加し、10 次で少し減少している。このことから、定義多項式の次数が同じである場合、行列の次数が増加しても同時に解の係数行列中の小行列の個数が増加すると、CPU 時間は減少する傾向があることがわかる。しかし、今回のテストデータの計算結果からは、小行列の個数と定義多項式の次数のどちらが CPU 時間に対してより支配的であるかは明らかではなかった。

ここでの差分方程式についての計算では、解の係数行列である行列のべきの指数を m として計算しているが、アルゴリズム・プログラム上は、具体的な整数であっても構わない。ただし、m が整数である場合は、差分方程式の係数行列 A のべき A^m を直接計算した方が効率的であろう。指数を m としたのは、行列のべきの一般形を求めるためであったが、固有値が記号で表されているときには、本研究のプログラムで計算された一般形 A^m の m に直接整数を代入しても、A^m の要素が有

理数にならないことがある。これは、A^m が完全に簡単化されていないからである。完全に簡単化するためには、m に整数を代入した後、固有値を表す記号に対するすべての定義多項式で、再度、M-簡約すればよい。

この2つのアルゴリズムは、Jordan 標準形を厳密に計算できれば、このような問題に適用できるという例として示した。複雑な差分方程式・微分方程式の解を求めるという場面では、このアルゴリズムで計算した解の係数行列をそのまま提示するよりも、その係数行列に、数値計算によって求めた固有値を代入して、近似解を計算する方が実用的であろう。

第 4 章

第 2 の標準形と Jacobson 標準形の計算法

4.1 定義

本研究では、Jordan 標準形のほかに、有理標準形から第 2 の標準形と Jacobson 標準形を計算できるように、Jordan 標準形の計算アルゴリズムを拡張した。

定理 17 (第 2 の標準形) 任意の n 次正方行列 A は、適当な正則行列 $S(\det S \neq 0)$ を用いて

$$S^{-1}AS = D_{11} \oplus D_{12} \oplus \cdots \oplus D_{tr_t} \tag{4.1}$$

という形のブロック対角行列に変換することができる。これを A の第 2 の標準形 (Second Natural Normal Form) という。ここで、各ブロック $D_{ij}(i = 1, \cdots, t; j = 1, \cdots, r_i)$ は、A の特性行列 $\lambda E - A$ の単純単因子 $p_{ij}(\lambda)^{k_{ij}}$ に随伴するコンパニオン行列である。だたし、単純単因子は基礎体のとり方によって異なるので、第 2 の標準形は基礎体のとり方に依存する。(4.1) の形の行列は、コンパニオン行列の並べ方の順序を除き、A によって一意的に定まる。第 2 の標準形は任意の体上の元を要素とする行列について存在する。

定義 17 (超同伴行列) 多項式 $f(\lambda)$ が p 次の既約多項式 $g(\lambda)$ のべき $g(\lambda)^q$ に等しいとき、次の形の r 次正方行列

$$L = \begin{array}{c} \left[\begin{array}{ccccc} C & M & & & O \\ & C & \ddots & & \\ & & \ddots & \ddots & \\ & & & \ddots & M \\ O & & & & C \end{array}\right] \end{array} \begin{array}{l} \uparrow \\ \\ r \\ \\ \downarrow \end{array} \tag{4.2}$$

を、多項式 $f(\lambda) = g(\lambda)^q (r = p \times q)$ の超同伴行列という。ただし、C は多項式 $g(\lambda)$ に随伴する p 次のコンパニオン行列であり、M は右上隅が 1 で他がすべて 0 からなる p 次正方行列

$$M = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ & & & 0 \\ & O & & \vdots \\ & & & 0 \end{bmatrix} \tag{4.3}$$

である。特に、$q = 1$、すなわち $f(\lambda) = g(\lambda)$ のときは、$L = C$ とする。

定理 18 (Jacobson 標準形) 任意の n 次正方行列 A は、適当な正則行列 $S(\det S \neq 0)$ を用いて

$$S^{-1}AS = L_{11} \oplus L_{12} \oplus \cdots \oplus L_{tr_t} \tag{4.4}$$

という形のブロック対角行列に変換することができる。これを A の Jacobson 標準形という。ここで、各ブロック $L_{ij}(i = 1, \cdots, t; j = 1, \cdots, r_i)$ は、A の特性行列 $\lambda E - A$ の単純単因子 $p_{ij}(\lambda)^{k_{ij}}$ の超同伴行列である。だたし、単純単因子は基礎体のとり方によって異なるので、Jacobson 標準形は基礎体のとり方に依存する。全固有値が属する体を基礎体にとれば、Jacobson 標準形は Jordan 標準形に一致する。(4.4) の形の行列は、超同伴行列の並べ方の順序を除き、A によって一意的に定まる。Jacobson 標準形は任意の体上の元を要素とする行列について存在する。

有理標準形は、体の拡大なしに有理演算のみで求まるが、Jordan 標準形は、全固有値が属するように体の拡大を必要とする。これらに対し、第 2 の標準形と Jacobson 標準形は、単純単因子の属する基礎体のとり方に依存するので、ちょうど中間に位置する標準形といえる。

4.2 アルゴリズム

1変数整数係数多項式を要素とする任意の n 次正方行列を A とおく。前述の定理に基づいて、第2の標準形、Jacobson 標準形を計算するアルゴリズムは次の手順となる。（ただし、手順は、数式処理言語 REDUCE で表現することを前提として記述する。）

入力： 有理数（あるいは1変数整数係数多項式）を要素とする妊ホ　鱸銓纈湯鳫鮑　　鞜鞳　櫂　意の行列

出力： 有理標準形，単因子，有理数体（1変数有理関数体）上での単純単因子，第2の標準形，Jacobson 標準形

(1) A を文献 [57] に示された方法（後述 付録 **A** 有理標準形の計算法）で有理標準形 R に変換する。R はコンパニオン行列 $C_1, \cdots, C_t$ を要素とするブロック対角行列であり、各 C_i の次数を m_i とおく。

(2) 各コンパニオン行列の最小多項式を求めることにより、（1を除いた）単因子のリストを得る。

$$\{e_1(\lambda), e_2(\lambda), \cdots, e_t(\lambda)\} \quad e_i(\lambda) = \phi_{C_{t-i+1}}(\lambda)$$

(3) REDUCE の組込関数 FACTORIZE を用いて、各単因子を因数分解して、有理数体（1変数有理関数体）上での単純単因子のリストを求める。

$$\{p_{ij}(\lambda)^{k_{ij}}\} \ (i = 1, \cdots, t; j = 1, \cdots, s_i) \quad p_{ij} \in \mathbf{Q}[\lambda] \ k_{ij} \geq 1$$

(4) 各単純単因子に対応するコンパニオン行列を構成し、（有理数体（1変数有理関数体）上の）第2の標準形を構成する。

(5) 各単純単因子に対応する超同伴行列を構成し、（有理数体（1変数有理関数体）上の）Jacobson 標準形を構成する。

4.3 計算例

4.2 節のアルゴリズムにしたがって、次の行列 A の第 2 の標準形、Jacobson 標準形を計算する。

$$A = \begin{bmatrix} 243 & -323 & 121 & -81 \\ 219 & -291 & 109 & -73 \\ 122 & -162 & 61 & -41 \\ 45 & -60 & 23 & -15 \end{bmatrix}$$

(1) A の有理標準形 R は次のようになる。

$$R = \begin{bmatrix} 0 & 0 & 0 & -4 \\ 1 & 0 & 0 & -4 \\ 0 & 1 & 0 & -5 \\ 0 & 0 & 1 & -2 \end{bmatrix}$$

この例では、有理数標準形 R はコンパニオン行列 C_1 1 個からなっている $(R = C_1)$ ので、$m_1 = 4$ である。

(2) (1) から、単因子は次の 1 つだけである。

$$e_1(\lambda) = \phi_{C_1}(\lambda) = \det(\lambda E - C_1) = \lambda^4 + 2\lambda^3 + 5\lambda^2 + 4\lambda + 4$$

(3) $e_1(\lambda)$ を有理数体上で因数分解すると、次のようになる。

$$e_1(\lambda) = (\lambda^2 + \lambda + 2)^2$$

したがって、単純単因子は $(\lambda^2 + \lambda + 2)^2$ 1 つだけである。ここで、$p(\lambda) = \lambda^2 + \lambda + 2$ とおく。

(4) 単純単因子 $p(\lambda)^2 = (\lambda^2 + \lambda + 2)^2$ に随伴する有理数体上の第 2 の標準形は次のようになる。

$$\begin{bmatrix} 0 & 0 & 0 & -4 \\ 1 & 0 & 0 & -4 \\ 0 & 1 & 0 & -5 \\ 0 & 0 & 1 & -2 \end{bmatrix}$$

(5) 単純単因子 $p(\lambda)^2 = (\lambda^2 + \lambda + 2)^2$ に随伴する有理数体上の Jacobson 標準形は次のようになる。

$$\left[\begin{array}{cc|cc} 0 & -2 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ \hline 0 & 0 & 0 & -2 \\ 0 & 0 & 1 & -1 \end{array}\right]$$

4.4 計算結果と考察

数式処理システム REDUCE (version3.5) 上に、**4.2**節のアルゴリズムをインプリメントし、**3.4**節に示したデータ 3.3,3.4（2 種類の 3 次から 20 次までの行列）について計算を行なった。具体的な行列と計算結果の一部は、付録 **C** に示す。

SUN SPARC server10 (40SPECmark 相当以上、主記憶容量 128MB) 上の数式処理システム REDUCE (version3.5) 上で、32MB のメモリを使用してデータ 3.3,3.4 についてそれぞれ第 2 の標準形と Jacobson 標準形を計算したときの CPU 時間（単位：ms（ミリ秒））を表 4.1,4.2 に示す。数字は、**4.2** 節のアルゴリズムのステップ (4),(5) にあたる処理、すなわち、データの行列の単純単因子の計算が終了したところから、それぞれ、第 2 の標準形、Jacobson 標準形を計算するまでに要した CPU 時間である。つまり、データの単純単因子が前もって与えられているときの CPU 時間である。表 4.1,4.2 の CPU 時間をグラフとして、図 4.1,4.2に示す。グラフの縦軸は CPU 時間（単位：ms）であり、横軸は入力データの行列の次数である。

表 4.1: 2 つの標準形の CPU 時間 (データ 3.3)

次数	第 2	Jacob	次数	第 2	Jacob
3	139	161	12	960	1030
4	190	190	13	1070	1080
5	250	250	14	1190	1210
6	370	370	15	1300	1360
7	410	430	16	1380	1460
8	490	510	17	1650	1600
9	590	640	18	1710	1770
10	670	730	19	1850	1860
11	780	830	20	1970	2030

（単位：ms）

表 4.2: 2 つの標準形の CPU 時間 (データ 3.4)

次数	第 2	Jacob	次数	第 2	Jacob
3	160	190	12	1130	1120
4	241	239	13	1190	1270
5	300	300	14	1400	1410
6	400	400	15	1540	1580
7	481	510	16	1640	1610
8	590	590	17	1740	1810
9	700	720	18	2110	2190
10	850	830	19	2170	2110
11	880	910	20	2150	2270

（単位：ms）

図 4.1: 2 つの標準形の CPU 時間のグラフ (データ 3.3)

図 4.2: 2 つの標準形の CPU 時間のグラフ (データ 3.4)

Jordan 標準形以外の標準形の計算については、現在広く使用されている数式処理システム上には、同様の機能をもつ関数が存在しないので、CPU 時間による比較はできないが、単純単因子から各標準形を計算するアルゴリズムを明らかにしたことは有意義であると思われる。

表 4.1,4.2 から分かるように、もとの行列から第 2 の標準形、Jacobson 標準形をそれぞれの計算に要する CPU 時間全体においては、単純単因子からその標準形を求めるのにかかる CPU 時間は、有理標準形の計算に要する CPU 時間（**3.4** 節 表 3.3(i), 3.4(i) 参照）に比べれば問題にならないくらい短く、また、入力行列の有理標準形の種類にかかわらず、Jordan 標準形とこの 2 つの標準形の構成のみに要する時間は同程度である（**3.4** 節 表 3.3(iii), 3.4(iii) 参照）。特に、第 2 の標準形と Jacobson 標準形の CPU 時間はほとんど同くらいであるので、図 4.1,4.2では、グラフが重なってしまっている。

図 4.1,4.2から分かるように、有理標準形のタイプが同じである場合には、要素が数値であるか多項式であるかにかかわらず、CPU 時間はほぼ同じ傾向（グラフの形）を持つしたがって、Jordan 標準形の場合と同様に、第 2 の標準形、Jacobson 標準形に要する計算時間を短縮するためには、有理標準形の計算法の改良・効率化が必要である。

第 5 章

有理標準形の計算法

5.1 従来のアルゴリズム

5.1.1 定義

本章では、有理標準形の計算法の効率化について述べる。3 章、4 章で述べたように、本研究の標準形の計算アルゴリズムでは、有理標準形の計算にかかる時間が全体の計算時間に対して支配的であるので、有理標準形の計算法の効率化が必要である。

前述のアルゴリズムの中の有理標準形の計算法は、Danilevskii の方法 [17] に基づいており、それを行列の基本変形の合成で表現したものである [57]。まず、従来のアルゴリズムの主要ステップとその問題点を示す。

記法の便宜上、基本変形を以下のように定義しなおす。

定義 18 (基本変形) 任意の n 次正方行列 A に対する次の 3 つの操作を基本変形と呼ぶ。

$op1(k,l)$: A の第 k 行と第 l 行を入れ換え、同時に第 k 列と第 l 列を入れ換える。

$op2(k,c)$: A の第 k 行を c^{-1} 倍し、同時に第 k 列を c 倍する。

$op3(k,l,c)$: A の第 k 行に第 l 行の c 倍を加え、同時に第 l 列から第 k 列の c 倍を引く。

基本変形は相似変換 ($A \mapsto S^{-1}AS$) であるので、A に任意の基本変形を施した行列 $\tilde{A}$ は A と相似であり、同じ有理標準形を持つ。

5.1.2 従来のアルゴリズムの問題点

任意の n 次正方行列を $A = [a_{i,j}]\,(i,j = 1,\cdots,n)$ とする。従来のアルゴリズムでは、A の有理標準形を計算する主要なステップとして以下の操作を繰り返し行なう。基本変形を施された A は操作の回数にかかわらず $\tilde{A} = [\tilde{a}_{i,j}]$ と表すことにする。

$p \leftarrow 0$;　% p はコンパニオン行列の個数
$k \leftarrow 1$;
$p \leftarrow p + 1$;
$kk \leftarrow k$;
while $k < n$ do
　{ $l \leftarrow k + 1$;
　　while $(l \leq n)$ & $(\tilde{a}_{l,k} = 0)$ do $l \leftarrow l + 1$;
　　if $l > n$ then break ;
　　else { $op1(k+1, l)$;
　　$op2(k+1, \tilde{a}_{k+1,k})$;
　　for $i \leftarrow 1$ to k do $op3(i, k+1, -\tilde{a}_{i,k})$;
　　for $i \leftarrow k+2$ to n do $op3(i, k+1, -\tilde{a}_{i,k})$;
　　$k \leftarrow k + 1$ }} ;
$m_p \leftarrow kk - k + 1$;　% m_p は p 番目のコンパニオン行列の次数

入力行列 A が多項式（あるいは整数。以下、括弧内は整数の場合に対応する）を要素とするとき、A の有理標準形の要素は必ず多項式（整数）になる。A の特性多項式 $\varphi_A(\lambda) = \det(\lambda E - A)$ の係数 c_i $(i = 0,\cdots,n-1)$ は、行列式の計算規則により、明らかに多項式（整数）であり、有理標準形がコンパニオン行列 1 個からなる場合には、定義 10, 定理 5 より $-c_i$ は A の有理標準形の最右列の要素に等しいからである。しかし、中間結果の行列の要素は多項式（整数）であるとは限らない。基本変形は、A に相似変換行列 S を $S^{-1}AS$ として、両側から乗じることによって行なわれ、l 回の基本変形の合成は、相似変換行列 $S_1,\cdots,S_l$ によって ${S_l}^{-1}\cdots{S_2}^{-1}{S_1}^{-1}AS_1S_2\cdots S_l = \tilde{A}$ として表せる。基本変形は除算を含むので、一般に、中間結果の行列 $\tilde{A}$ の要素は有理式（有理数）になる。有理式（有理数）同士の演算では、多項式（整数）同士の計算に比べて、通分・約分、すなわち、分母の多項式の最小公倍数・分母と分子の最大公約数の計算に余分な時間がかかり、また、共通分母は計算の段階が進むにつれて増大するので、途中の計算ではより多くのメモリを必要とする。

5.2 アルゴリズムの効率化

5.2.1 基本概念

5.1.2 項で述べたアルゴリズムの主要ステップにおける消去操作 $op2(k,c)$, $op3(k,l,c)$ は、Gauss の消去法における上三角化に類似している。Bareiss の Single-step fraction-free elimination アルゴリズム [7], [100] では、「第 k 回目の消去で生じる行列の要素中の有理式の分母は第 $k+1$ 回目の消去後の行列の各要素の共通因子と打ち消し合う ($k = 1, \cdots, n-1$)」という性質を用いて、分数なし計算を行なう。Gauss の消去法による上三角化では、行に対する消去操作（$op3(k,l,c)$ の前半部分）のみを行なえばよいのに対し、有理標準形の計算では、入力行列 A とその有理標準形の間の相似関係を保つために、施される変換は相似変換（行と列の両方に対する操作）でなければならないが、その点を考慮に入れれば、上の性質はそのまま有理標準形の計算法に適用することができる。以下で、同じ性質が有理標準形の計算アルゴリズムにも存在することを示し、次項で効率化した主要ステップのアルゴリズムを与える。

任意の n 次正方行列 $A = [a_{i,j}]\,(i,j = 1,\cdots,n)$ について考える。基本変形を施された A は、$A^{(0)} = \tilde{A}^{(0)} = A$ として、k 回目の階段化消去が施された行列を $A^{(k)}$、$A^{(k)}$ に対して $op2(k+1, 1/\tilde{a}_{k,k-1}^{(k-2)})$ を施した行列を $\tilde{A}^{(k)}$ とする。$\tilde{a}_{i,j}^{(k)}$ は有理式である $\tilde{A}^{(k)}$ の (i,j) 要素の分子であるとする。（ただし、$k=1$ のとき、$\tilde{a}_{k,k-1}^{(k-2)} = 1$ とする。）

$a_{2,1} \neq 0$ として（$a_{2,1} = 0$ のときは $a_{l,1} \neq 0$ なる l $(l > 2)$ をみつけ、$op1(2,l)$ を行なう）、A に対して $op2(2, a_{2,1})$ を行ない、次に、各行・列について $op3(i, 2, -a_{i,1})$ $(i = 1, 3, \cdots, n)$ を施すと、

$$\tilde{A}^{(1)} = \frac{1}{a_{2,1}} \begin{bmatrix} 0 & \tilde{a}_{1,2}^{(1)} & \tilde{a}_{1,3}^{(1)} & \cdots & \tilde{a}_{1,n}^{(1)} \\ a_{2,1} & \tilde{a}_{2,2}^{(1)} & a_{2,3} & \cdots & a_{2,n} \\ 0 & \tilde{a}_{3,2}^{(1)} & \tilde{a}_{3,3}^{(1)} & \cdots & \tilde{a}_{3,n}^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \tilde{a}_{n,2}^{(1)} & \tilde{a}_{n,3}^{(1)} & \cdots & \tilde{a}_{n,n}^{(1)} \end{bmatrix}$$

となり、第 2 列目以降は $a_{2,1}$ を共通分母として持つので、これをくくり出しておける。同様にして、$\tilde{a}_{3,2}^{(1)} \neq 0$ として、$\tilde{A}^{(1)}$ について $op2(3, \tilde{a}_{3,2}^{(1)})$、$op3(i, 3, -\tilde{a}_{i,2}^{(1)})$ $(i = 1, 2, 4, \cdots, n)$ を施せば、第 3 列以降の共通分母 $\tilde{a}_{3,2}^{(1)}$ をくくり出して

$$A^{(2)} = \frac{1}{a_{2,1}\tilde{a}_{3,2}^{(1)}} \begin{bmatrix} 0 & 0 & a_{2,1}^2\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\ a_{2,1}\tilde{a}_{3,2}^{(1)} & 0 & a_{2,1}^2\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\ 0 & \tilde{a}_{3,2}^{(1)} & a_{2,1}\triangle & \triangle & \cdots & \triangle \\ 0 & 0 & a_{2,1}^2\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & a_{2,1}^2\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \end{bmatrix}$$

と表せる。ここで、$\triangle$ は整数係数多項式である。上の行列では、第 3 行以外の行の要素がすべて $a_{2,1}$ を因子として含んでいることがわかる。$(3,3)$ 要素を除いて、第 3 列は $a_{2,1}^2$ を因子として含むので、$op2(3, 1/a_{2,1})$ を施すと、

$$
\begin{aligned}
\tilde{A}^{(2)} &= \frac{1}{a_{2,1}\tilde{a}_{3,2}^{(1)}}\begin{bmatrix}
0 & 0 & a_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\
a_{2,1}\tilde{a}_{3,2}^{(1)} & 0 & a_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\
0 & a_{2,1}\tilde{a}_{3,2}^{(1)} & a_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\
0 & 0 & a_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & a_{2,1}\triangle & a_{2,1}\triangle & \cdots & a_{2,1}\triangle
\end{bmatrix} \\
&= \frac{1}{\tilde{a}_{3,2}^{(1)}}\begin{bmatrix}
0 & 0 & \triangle & \triangle & \cdots & \triangle \\
\tilde{a}_{3,2}^{(1)} & 0 & \triangle & \triangle & \cdots & \triangle \\
0 & \tilde{a}_{3,2}^{(1)} & \triangle & \triangle & \cdots & \triangle \\
0 & 0 & \triangle & \triangle & \cdots & \triangle \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \triangle & \triangle & \cdots & \triangle
\end{bmatrix}
\end{aligned}
$$

となり、分母 $a_{2,1}$ がキャンセルされて、$\tilde{A}^{(2)}$ の各要素の共通分母は $\tilde{a}_{3,2}^{(1)}$ のみとなる。有理標準形を計算するための階段化消去は、同じ手順を各列に対して施していくので、3 回目以降の消去においても、k 回目の消去で生じる分母が $k+1$ 回目の消去後の因子と打ち消し合うのは明らかである。この性質から、k 回目の階段化消去で $op2(k+1, \tilde{a}_{k+1,k}^{(k-1)})$ を施す直前に、$\tilde{A}^{(k-1)}$ 全体に予め $\tilde{a}_{k+1,k}^{(k-1)}$ を乗じておくことによって、分数なし計算が実現でき、その因子は $k+1$ 回目の消去の後で $A^{(k+1)}$ に $op2(k+2, 1/\tilde{a}_{k+1,k}^{(k-1)})$ を施すことにより除去されるので、行列の要素が増大していくことはない。

5.2.2 分数なしアルゴリズム

前項の考え方に基づき、**5.1.2** 項の主要ステップのアルゴリズムを改良すると、以下のようになる。任意の n 次正方行列を $A = [a_{i,j}]\,(i, j = 1, \cdots, n)$ とする。操作を施された A は操作の回数にかかわらず $\tilde{A} = [\tilde{a}_{i,j}]$ と表すことにする。

$p \leftarrow 0$;　% p はコンパニオン行列の個数
$k \leftarrow 1$;
$qindx \leftarrow 0$;　% $qindx$ は分母を保存する配列 q の index
$rfl \leftarrow 0$;　% rfl は分数なし計算をするかどうかのフラグ
$p \leftarrow p + 1$;
$kk \leftarrow k$;
while $k < n$ do
　{ $l \leftarrow k + 1$;
　　while $(l \leq n)$ & $(\tilde{a}_{l,k} = 0)$ do $l \leftarrow l + 1$;
　　if $l > n$ then break ;
　　else { $op1(k + 1, l)$;
　　if $rfl = 0$ then { $qindx \leftarrow qindx + 1$;
　　$q(qindx) \leftarrow \tilde{a}_{k+1,k}$;
　　if $\tilde{a}_{k+1,k} \neq 1$
　　then { if $k + 1 \neq n$ then $\tilde{A} \leftarrow \tilde{A} * \tilde{a}_{k+1,k}$;
　　$op2(k + 1, \tilde{a}_{k+1,k})$ } } ;
　　else　$op2(k + 1, \tilde{a}_{k+1,k})$;
　　$\alpha \leftarrow \tilde{a}_{k+1,k}$;
　　for $i \leftarrow 1$ to k do $op3(i, k + 1, -\tilde{a}_{i,k}/\alpha)$;
　　for $i \leftarrow k + 2$ to n do $op3(i, k + 1, -\tilde{a}_{i,k}/\alpha)$;
　　if $qindx \geq 2$ then { $qq \leftarrow q(qindx - 1)$;
　　$op2(k + 1, qq^{-1})$;
　　$\tilde{A} \leftarrow \tilde{A}/qq$ } ;
　　$k \leftarrow k + 1$ }} ;
$m_p \leftarrow kk - k + 1$;　% m_p は p 番目のコンパニオン行列の次数
if $(m_p = 1)$ & $(rfl = 0)$ then if $qindx \neq 0$ then { $\tilde{A} \leftarrow \tilde{A}/q(qindx)$;
　　$qindx \leftarrow 0$ } ;
if $k = n$ then $qindx \leftarrow 0$;

5.1.2 項に述べた従来のアルゴリズム中の主要ステップでも、上記の分数なしアルゴリズムの主要ステップでも、有理標準形の中のコンパニオン行列を 1 つ 1 つ構成していくが、構成されたコンパ

ニオン行列が定理5の条件を満たさない場合がある。具体的には、第k行・第k列まで計算が終った中間結果の行列$\tilde{A}$の第1行から第k行・第1列から第k列までを部分的な行列$\tilde{A}_{(1\leftrightarrow k)}$として考えたとき、$\tilde{A}_{(1\leftrightarrow k)}$が以下のようになる場合である。ここで、$c_{i,j}$は$i$番目のコンパニオン行列の最右列の上から$j$番目の要素であるとする。

$$\tilde{A}_{(1\leftrightarrow k)} = \left[\begin{array}{cccc|ccc} 0 & \cdots & 0 & c_{1,1} & & & \star \\ 1 & \ddots & \vdots & c_{1,2} & & & \vdots \\ & \ddots & 0 & \vdots & & O & \vdots \\ O & & 1 & c_{1,m_1} & & & \vdots \\ \hline & & & & & \ddots & \star \\ & O & & & 0 & \cdots\ 0 & c_{p,1} \\ & & & & 1 & \ddots\ \vdots & c_{p,2} \\ & & & & & \ddots\ 0 & \vdots \\ & & & & O & 1 & c_{p,m_p} \end{array}\right] \tag{5.1}$$

このような場合には、まず、基本変形を用いて、0でない$\star$を消去する。そのような操作を施しても、0でない$\star$が1つ以上存在する場合、ある行・列の交換を行なった後、$p \leftarrow j-1$として、$\star\ (\neq 0)$が右にあるコンパニオン行列の中で一番上の行列C_jの前まで遡って計算をしなおさなければいけない（後述 付録 **A** 有理標準形の構成法）。その場合、上記のステップを含む本研究の分数なしアルゴリズムでは、$rlf \neq 0$として、以降は従来のアルゴリズムを用いている。なぜならば、そのときの$\star\ (\neq 0)$は必ずしも多項式ではないからである。$p \leftarrow 0$として最初のコンパニオン行列から計算をしなおす場合、有理式である$\star$の共通分母をくくり出さなければ、以降の計算は分数なし計算ではなくなる。上記のアルゴリズムでは、共通分母をくくり出すことによって行列全体の各要素が成長するのをさけるために、$p=0$となるときは、無条件で分数なし計算を中止することにしている。

また、コンパニオン行列が定理5の条件に合わない別の場合として、あるコンパニオン行列C_{j+1}の最小多項式$\phi_{C_{j+1}}(\lambda)$が1つ前のコンパニオン行列C_jの最小多項式$\phi_{C_j}(\lambda)$を割り切らないということがある。その場合にも、ある行・列の交換を行ない、$p \leftarrow j-1$として、条件に合わないコンパニオン行列C_jの前まで遡って、計算をやりなおす。

著者は、計算をやりなおす場合にも、有理式である$\star$の共通分母をくくり出し、分数なし計算を引き続き行なうアルゴリズムも作成したが、その優劣の比較は今後の検討課題としたい。

5.3 計算例

比較のため、**5.1.2** 項の主要ステップを含む従来のアルゴリズムと **5.2.2** 項で示した分数なしアルゴリズムによる計算例をそれぞれ示す。入力行列は以下の 4 次正方行列 A である。

$$A = \begin{bmatrix} -1080y+3 & 1440y-3 & -540y+1 & 360y-1 \\ -978y+3 & 1304y-3 & -489y+1 & 326y-1 \\ -564y+2 & 752y-2 & -282y+1 & 188y-1 \\ -234y+3 & 312y-4 & -117y+2 & 78y-1 \end{bmatrix}$$

5.3.1 従来のアルゴリズムによる計算例

$$\begin{bmatrix} -1080y+3 & 1440y-3 & -540y+1 & 360y-1 \\ -978y+3 & 1304y-3 & -489y+1 & 326y-1 \\ -564y+2 & 752y-2 & -282y+1 & 188y-1 \\ -234y+3 & 312y-4 & -117y+2 & 78y-1 \end{bmatrix}$$

$$\downarrow \ op2(2, -978y+3)$$

$$\begin{bmatrix} -1080y+3 & -1408320y^2+7254y-9 & -540y+1 & 360y-1 \\ 1 & 1304y-3 & \frac{489y-1}{978y-3} & -\frac{1}{3} \\ -564y+2 & -735456y^2+4212y-1 & -282y+1 & 188y-1 \\ -234y+3 & -305136y^2+4848y-12 & -117y+2 & 78y-1 \end{bmatrix}$$

$$\downarrow \ op3(1, 2, 1080y-3)$$

$$\begin{bmatrix} 0 & 102y & \frac{17y}{326y-1} & 0 \\ 1 & 224y & \frac{489y-1}{978y-3} & -\frac{1}{3} \\ -564y+2 & -126336y^2+360y & -282y+1 & 188y-1 \\ -234y+3 & -52416y^2+906y-3 & -117y+2 & 78y-1 \end{bmatrix}$$

$$\downarrow \ op3(3, 2, 564y-2)$$

$$\begin{bmatrix} 0 & \frac{23664y^2-68y}{326y-1} & \frac{17y}{326y-1} & 0 \\ 1 & -\frac{56724y^2-870y+2}{978y-3} & \frac{489y-1}{978y-3} & -\frac{1}{3} \\ 0 & -\frac{245112y^2-1392y+2}{978y-3} & \frac{282y-1}{978y-3} & -\frac{1}{3} \\ -234y+3 & 13572y^2-456y+1 & -117y+2 & 78y-1 \end{bmatrix}$$

$$\downarrow \ op3(4, 2, 234y-3)$$

$$\begin{bmatrix} 0 & \frac{23664y^2-68y}{326y-1} & \frac{17y}{326y-1} & 0 \\ 1 & \frac{19560y^2-342y+1}{978y-3} & \frac{489y-1}{978y-3} & -\frac{1}{3} \\ 0 & -\frac{168828y^2-180y-1}{978y-3} & \frac{282y-1}{978y-3} & -\frac{1}{3} \\ 0 & -\frac{37644^2+244y-1}{326y-1} & \frac{202y-1}{326y-1} & 0 \end{bmatrix}$$

$$\downarrow \ op2(3, -\frac{168828y^2-180y-1}{978y-3})$$

$$\begin{bmatrix} 0 & \frac{23664y^2-68y}{326y-1} & -\frac{2870076y^3-3060y^2-17y}{318828y^2-1956y+3} & 0 \\ 1 & \frac{19560y^2-342y+1}{978y-3} & -\frac{82556892y^3-256848y^2-309y+1}{956484y^2-5868y+9} & -\frac{1}{3} \\ 0 & 1 & \frac{282y-1}{978y-3} & \frac{326y-1}{168828y^2-180y-1} \\ 0 & -\frac{37644y^2+244y-1}{326y-1} & -\frac{34103256y^3-205188y^2-22y+1}{318828y^2-1956y+3} & 0 \end{bmatrix}$$

$$\downarrow \ op3(1, 3, -\frac{23664y^2-68y}{326y-1})$$

$$\begin{bmatrix} 0 & 0 & -\frac{9758y^2-17y}{326y-1} & -\frac{23664y^2-68y}{168828y^2-180y-1} \\ 1 & \frac{19560y^2-342y+1}{978y-3} & -\frac{13126716y^3+155640y^2-921y+1}{956484y^2-5868y+9} & -\frac{1}{3} \\ 0 & 1 & \frac{282y-1}{978y-3} & \frac{326y-1}{168828y^2-180y-1} \\ 0 & -\frac{37644y^2+244y-1}{326y-1} & -\frac{34103256y^3-205188y^2-22y+1}{318828y^2-1956y+3} & 0 \end{bmatrix}$$

$$\downarrow \ op3(2, 3, -\frac{19560y^2-342y+1}{978y-3})$$

$$\begin{bmatrix} 0 & 0 & -\frac{9758y^2-17y}{326y-1} & -\frac{23664y^2-68y}{168828y^2-180y-1} \\ 1 & 0 & -\frac{6354y^2+33y}{326y-1} & -\frac{62796y^2-174y}{168828y^2-180y-1} \\ 0 & 1 & 20y & \frac{326y-1}{168828y^2-180y-1} \\ 0 & -\frac{37644y^2-244y-1}{326-1} & -\frac{752880y^3+28896y^2-188y}{326y-1} & 0 \end{bmatrix}$$

$$\downarrow \ op3(4, 3, \frac{37644y^2+244y-1}{326y-1})$$

$$\begin{bmatrix} 0 & 0 & -\frac{2320908y^3-16932y^2+51y}{168828y^2-180y-1} & -\frac{23644y^2-68y}{168828y^2-180y-1} \\ 1 & 0 & \frac{3960612y^3+25476y^2-207y}{168828y^2-180y-1} & -\frac{62796y^2-174y}{168828y^2-180y-1} \\ 0 & 1 & \frac{3376560y^3-41244y^2-264y+1}{168828y^2-180y-1} & \frac{326y-1}{168828y^2-180y-1} \\ 0 & 0 & -\frac{16784184y^3+7572y^2-6y-1}{168828y^2-180y-1} & \frac{37644y^2+244y-1}{168828y^2-180y-1} \end{bmatrix}$$

$$\downarrow \ op2(4, -\frac{16784184y^3+7572y^2-6y-1}{168828y^2-180y-1})$$

$$\begin{bmatrix} 0 & 0 & -\frac{2320908y^3-33762y^2+51y}{168828y^2-180y-1} & \frac{397180930176y^5-962140704y^4-656880y^3-23256y^2+68y}{28502893584y^4-60778080y^3-305256y^2+360y+1} \\ 1 & 0 & \frac{3960612y^3+25476y^2-207y}{168828y^2-180y-1} & \frac{10539796184 64y^5-2444956704y^4-1694304y^3-61752y^2+174y}{28502893584y^4-60778080y^3-305256y^2+360y+1} \\ 0 & 1 & \frac{3376560y^3-41244y^2-264y+1}{168828y^2-180y-1} & -\frac{5471643984y^4-14315712y^3-9528y^2-320y+1}{28502893584y^4-60778080y^3-305256y^2+360y+1} \\ 0 & 0 & 1 & \frac{37644y^2+244y-1}{168828y^2-180y-1} \end{bmatrix}$$

$$\downarrow\ op3(1,4,\frac{2320908y^3-16932y^2+51y}{168828y^2-180y-1})$$

$$\begin{bmatrix} 0 & 0 & 0 & 17y \\ 1 & 0 & \frac{3960612y^3+25476y^2-207y}{168828y^2-180y-1} & \frac{662145362640y^5+831402432y^4-11031384y^3-69504y^2+225y}{28502893584y^4-60778080y^3-305256y^2+360y+1} \\ 0 & 1 & \frac{3376560y^3-41244y^2-264y+1}{168828y^2-180y-1} & -\frac{5471643984y^4-14315712y^3-9528y^2-320y+1}{28502893584y^4-60778080y^3-305256y^2+360y+1} \\ 0 & 0 & 1 & \frac{37644y^2+244y-1}{168828y^2-180y-1} \end{bmatrix}$$

$$\downarrow\ op3(2,4,\frac{3960612y^3+25476y^2-207y}{168828y^2-180y-1})$$

$$\begin{bmatrix} 0 & 0 & 0 & 17y \\ 1 & 0 & 0 & 18y \\ 0 & 1 & \frac{3376560y^3-41244y^2-264y+1}{168828y^2-180y-1} & \frac{668662202736y^5-1883492016y^4-29177976y^3+21312y^2+527y-1}{28502893584y^4-60778080y^3-305256y^2+360y+1} \\ 0 & 0 & 1 & \frac{37644y^2+244y-1}{168828y^2-180y-1} \end{bmatrix}$$

$$\downarrow op3(3,4,\frac{3376560y^3-41244y^2-264y+1}{168828y^2-180y-1})$$

$$\begin{bmatrix} 0 & 0 & 0 & 17y \\ 1 & 0 & 0 & 18y \\ 0 & 1 & 0 & 19y \\ 0 & 0 & 1 & 20y \end{bmatrix}$$

5.3.2 分数なしアルゴリズムによる計算例

ここでは、$op2(k+1,\tilde{a}_{k+1,k}^{(k-1)})$ を施す前に、予め $\tilde{a}_{k+1,k}^{(k-1)}$ を乗じた後の行列を、行列の前に分母 $\tilde{a}_{k+1,k}^{(k-1)}$ がくくり出された形 $(\times 1/\tilde{a}_{k+1,k}^{(k-1)})$ で表すが、実際の計算では、$op2(k+1,1/\tilde{a}_{k+1,k}^{(k-1)})$ を施す前までは、その分母を意識する必要はない。

$$\begin{bmatrix} -1080y+3 & 1440y-3 & -540y+1 & 360y-1 \\ -978y+3 & 1304y-3 & -489y+1 & 326y-1 \\ -564y+2 & 752y-2 & -282y+1 & 188y-1 \\ -234y+3 & 312y-4 & -117y+2 & 78y-1 \end{bmatrix}$$

$$\downarrow\ (\times\ -978y+3)$$

$$\left(\times\ \frac{1}{-978y+3}\right)\begin{bmatrix} 1056240y^2-6174y+9 & -1408320y^2+7254y-9 & 528120y^2-2598y+3 & -352080y^2+2058y-3 \\ 956484y^2-5868y+9 & -1275312y^2+6846y-9 & 478242y^2-2445y+3 & -318828y^2+1956y-3 \\ 551592y^2-3648y+6 & -735456y^2+4212y-6 & 275796y^2-1824y+3 & -183864y^2+1542y-3 \\ 228852y^2-3636y+9 & -305136y^2+4848y-12 & 114426y^2-2307y+6 & -76284y^2+1212y-3 \end{bmatrix}$$

$$\downarrow\ op2(2,-978y+3)$$

$$\left(\times\ \frac{1}{-978y+3}\right)$$

$$\begin{bmatrix} 1056240y^2-6174y+9 & \begin{matrix}1377336960y^3\\-11319372y^2+30564y-27\end{matrix} & 528120y^2-2598y+3 & -352080y^2+2058y-3 \\ -978y+3 & -1275312y^2+6846y-9 & -489y+1 & 326y-1 \\ 551592y^2-3648y+6 & \begin{matrix}719275968y^3\\-6325704y^2+18504y-18\end{matrix} & 275796y^2-1824y+3 & -183864y^2+1542y-3 \\ 228852y^2-3636y+9 & \begin{matrix}298423008y^3\\-5656752y^2+26280y-36\end{matrix} & 114426y^2-2307y+6 & -76284y^2+1212y-3 \end{bmatrix}$$

$$\downarrow\ op3(1,2,1080y-3)$$

$$\left(\times\ \frac{1}{-978y+3}\right)$$

$$\begin{bmatrix} 0 & -99756y^2+306y & -51y & 0 \\ -978y+3 & -219072y^2+672y & -489y+1 & 326y-1 \\ 551592y^2-3648y+6 & \begin{matrix}123556608y^3\\-731088y^2+1080y\end{matrix} & 275796y^2-1824y+3 & -183864y^2+1542y-3 \\ 228852y^2-3636y+9 & \begin{matrix}51262848y^3\\-1043316y^2+5652y-9\end{matrix} & 114426y^2-2307y+6 & -76284y^2+1212y-3 \end{bmatrix}$$

$$\downarrow\ op3(3,2,564y-2)$$

$$\left(\times\ \frac{1}{-978y+3}\right)$$

$$\begin{bmatrix} 0 & -70992y^2+204y & -51y & 0 \\ -978y+3 & 56724y^2-870y+2 & -489y+1 & 326y-1 \\ 0 & 245112y^2-1392y+2 & -282y+1 & 326y-1 \\ 228852y^2-3636y+9 & \begin{matrix}-13273416y^3\\+486684y^2-2346y+3\end{matrix} & 114426y^2-2307y+6 & -76284y^2+1212y-3 \end{bmatrix}$$

$$\downarrow\ op3(4,2,234y-3)$$

$$\left(\times\ \frac{1}{-978y+3}\right)\begin{bmatrix} 0 & -70992y^2+204y & -51y & 0 \\ -978y+3 & -19560y^2+342y-1 & -489y+1 & 326y-1 \\ 0 & 168828y^2-180y-1 & -282y+1 & 326y-1 \\ 0 & 112932y^2+732y-3 & -606y+3 & 0 \end{bmatrix}$$

$$\downarrow\ \left(\times\ 168828y^2-180y-1\right)$$

$$\left(\times\ \frac{1}{-978y+3}\right)\left(\times\ \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & \begin{matrix}-11985437376y^4+47219472y^3\\+34272y^2-204y\end{matrix} & \begin{matrix}-8610228y^3\\+9180y^2+51y\end{matrix} & 0 \\ \begin{matrix}-165113784y^3\\+682524y^2+438y-3\end{matrix} & \begin{matrix}-3302275680y^4+61259976y^3\\-210828y^2-162y+1\end{matrix} & \begin{matrix}-82556892y^3\\+256848y^2+309y-1\end{matrix} & \begin{matrix}55037928y^3\\-227508y^2-146y+1\end{matrix} \\ 0 & \begin{matrix}28502893584y^4-60778080y^3\\-305256y^2+360y+1\end{matrix} & \begin{matrix}-47609496y^3\\+219588y^2+102y-1\end{matrix} & \begin{matrix}55037928y^3\\-227508y^2-146y+1\end{matrix} \\ 0 & \begin{matrix}19066083696y^4+103254336y^3\\-751176y^2-192y+3\end{matrix} & \begin{matrix}-102309768y^3\\+615564y^2+66y-3\end{matrix} & 0 \end{bmatrix}$$

$$\downarrow \; op2(3, 168828y^2 - 180y - 1)$$

$$\left(\times \frac{1}{-978y+3}\right)\left(\times \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & \begin{matrix}-11985437376y^4+47219472y^3\\+34272y^2-204y\end{matrix} & \begin{matrix}-1453647572784y^5+3099682080y^4\\+15568056y^3-18360y^2-51y\end{matrix} & 0 \\ \begin{matrix}-165113784y^3\\+682524y^2+438y-3\end{matrix} & \begin{matrix}-3302275680y^4+61259976y^3\\-210828y^2-162y+1\end{matrix} & \begin{matrix}-13937914962576y^5+58223374704y^4\\+88492104y^3-481296y^2-129y+1\end{matrix} & \begin{matrix}55037928y^3\\-227508y^2-146y+1\end{matrix} \\ 0 & 168828y^2-180y-1 & -47609496y^3+219588y^2+102y-1 & 326y-1 \\ 0 & \begin{matrix}19066083696y^4+103254336y^3\\-751176y^2-192y+3\end{matrix} & \begin{matrix}-17272753511904y^5+122340197232y^4\\+2650896y^3-1133928y^2+474y+3\end{matrix} & 0 \end{bmatrix}$$

$$\downarrow \; op3(1, 3, 70992y^2 - 204y)$$

$$\left(\times \frac{1}{-978y+3}\right)\left(\times \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & 0 & \begin{matrix}-4833540912816y^5+28401010560y^4\\-21986712y^3-110160y^2+153y\end{matrix} & \begin{matrix}23143392y^3\\-137496y^2+204y\end{matrix} \\ \begin{matrix}-165113784y^3\\+682524y^2+438y-3\end{matrix} & \begin{matrix}-3302275680y^4+61259976y^3\\-210828y^2-162y+1\end{matrix} & \begin{matrix}-2216157208848y^5-23913581040y^4\\+196632504y^3-178968y^2-741y+1\end{matrix} & \begin{matrix}55037928y^3\\-227508y^2-146y+1\end{matrix} \\ 0 & 168828y^2-180y-1 & -47609496y^3+219588y^2+102y-1 & 326y-1 \\ 0 & \begin{matrix}19066083696y^4+103254336y^3\\-751176y^2-192y+3\end{matrix} & \begin{matrix}-17272753511904y^5+122340197232y^4\\+2650896y^3-1133928y^2+474y+3\end{matrix} & 0 \end{bmatrix}$$

$$\downarrow \; op3(2, 3, 19560y^2 - 342y + 1)$$

$$\left(\times \frac{1}{-978y+3}\right)\left(\times \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & 0 & \begin{matrix}-4833540912816y^5+28401010560y^4\\-21986712y^3-110160y^2+153y\end{matrix} & \begin{matrix}23143392y^3\\-137496y^2+204y\end{matrix} \\ \begin{matrix}-165113784y^3\\+682524y^2+438y-3\end{matrix} & 0 & \begin{matrix}-3147398950608y^5-3335992128y^4\\+75919032y^3-13824y^2-297y\end{matrix} & \begin{matrix}61414488y^3\\-358560y^2+522y\end{matrix} \\ 0 & 168828y^2-180y-1 & \begin{matrix}-3302275680y^4+13650480y^3\\+8760y^2-60y\end{matrix} & 326y-1 \\ 0 & \begin{matrix}19066083696y^4+103254336y^3\\-751176y^2-192y+3\end{matrix} & \begin{matrix}-372932597093760y^6\\-12771807700032y^5+1532800990008y^4\\-353750112y^3-507096y^2+1692y\end{matrix} & 0 \end{bmatrix}$$

$$\downarrow \; op3(4, 3, -112932y^2 - 732y + 3)$$

$$\left(\times \frac{1}{-978y+3}\right)\left(\times \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & 0 & \begin{matrix}-2219911367472y^5+29814275232y^4\\-169025832y^3+451656y^2-459y\end{matrix} & \begin{matrix}23143392y^3\\-137496y^2+204y\end{matrix} \\ \begin{matrix}-165113784y^3\\+682524y^2+438y-3\end{matrix} & 0 & \begin{matrix}3788262008208y^5+1126515168y^4\\-311839848y^3+1443960y^2-1863y\end{matrix} & \begin{matrix}61414488y^3\\-358560y^2+522y\end{matrix} \\ 0 & 168828y^2-180y-1 & \begin{matrix}-3302275680y^4+50466312y^3\\+134460y^2-1770y+3\end{matrix} & 326y-1 \\ 0 & 0 & \begin{matrix}-16053803449056y^5+91247094864y^4\\-100886256y^3+853128y^2-5814y+9\end{matrix} & \begin{matrix}-36815832y^3\\-125700y^2+1710y-3\end{matrix} \end{bmatrix}$$

$$\downarrow \; op2(3, \frac{1}{-978y+3})$$

$$\left(\times\ \frac{1}{-978y+3}\right)\left(\times\ \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & 0 & \begin{matrix}2269848024y^4-23522220y^3\\+100674y^2-153y\end{matrix} & \begin{matrix}23143392y^3\\-137496y^2+204y\end{matrix} \\ \begin{matrix}-165113784y^3\\+682524y^2+438y-3\end{matrix} & 0 & \begin{matrix}-3873478536y^4-13033692y^3\\+278874y^2-621y\end{matrix} & \begin{matrix}61414488y^3\\-358560y^2+522y\end{matrix} \\ 0 & \begin{matrix}-165113784y^3\\+682524y^2+438y-3\end{matrix} & \begin{matrix}-3302275680y^4+50466312y^3\\+134460y^2-1770y+3\end{matrix} & -318828y^2+1956y-3 \\ 0 & 0 & \begin{matrix}16414931952y^4-42947136y^3\\-28584y^2-960y+3\end{matrix} & \begin{matrix}-36815832y^3\\-125700y^2+1710y-3\end{matrix} \end{bmatrix}$$

$$\downarrow\ \left(\times\ \frac{1}{-978y+3}\right)$$

$$\left(\times\ \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & 0 & -2320908y^3+16932y^2-51y & -23664y^2+68y \\ 168828y^2-180y-1 & 0 & 3960612y^3+25476y^2-207y & -62796y^2+174y \\ 0 & 168828y^2-180y-1 & 3376560y^3-41244y^2-264y+1 & 326y-1 \\ 0 & 0 & -16784184y^3-7572y^2+6y+1 & 37644y^2+244y-1 \end{bmatrix}$$

$$\downarrow\ op2(4, -16784184y^3 - 7572y^2 + 6y + 1)$$

$$\left(\times\ \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & 0 & \begin{matrix}-2320908y^3\\+16932y^2-51y\end{matrix} & \begin{matrix}397180930176y^5-962140704y^4\\-656880y^3-23256y^2+68y\end{matrix} \\ 168828y^2-180y-1 & 0 & \begin{matrix}3960612y^3\\+25476y^2-207y\end{matrix} & \begin{matrix}1053979618464y^5-2444956704y^4\\-1694304y^3-61752y^2+174y\end{matrix} \\ 0 & 168828y^2-180y-1 & \begin{matrix}3376560y^3\\-41244y^2-264y+1\end{matrix} & \begin{matrix}-5471643984y^4+14315712y^3\\+9528y^2+320y-1\end{matrix} \\ 0 & 0 & 1 & 37644y^2+244y-1 \end{bmatrix}$$

$$\downarrow\ op3(1, 4, 2320908y^3 - 16932y^2 + 51y)$$

$$\left(\times\ \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & 0 & 0 & \begin{matrix}484549190928y^5-1033227360y^4\\-5189352y^3+6120y^2+17y\end{matrix} \\ 168828y^2-180y-1 & 0 & \begin{matrix}3960612y^3\\+25476y^2-207y\end{matrix} & \begin{matrix}662145362640y^5+831402432y^4\\-11031384y^3-69504y^2+225y\end{matrix} \\ 0 & 168828y^2-180y-1 & \begin{matrix}3376560y^3\\-41244y^2-264y+1\end{matrix} & \begin{matrix}-5471643984y^4+14315712y^3\\+9528y^2+320y-1\end{matrix} \\ 0 & 0 & 1 & 37644y^2+244y-1 \end{bmatrix}$$

$$\downarrow\ op3(2, 4, -3960612y^3 - 25476y^2 + 207y)$$

$$\left(\times \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & 0 & 0 & \begin{array}{c}484549190928y^5-1033227360y^4\\-5189352y^3+6120y^2+17y\end{array} \\ 168828y^2-180y-1 & 0 & 0 & \begin{array}{c}513052084512y^5-1094005440y^4\\-5494608y^3+6480y^2+18y\end{array} \\ 0 & 168828y^2-180y-1 & \begin{array}{c}3376560y^3\\-41244y^2-264y+1\end{array} & \begin{array}{c}668662202736y^5-1883492016y^4\\-29177976y^3+21312y^2+527y-1\end{array} \\ 0 & 0 & 1 & 37644y^2+244y-1 \end{bmatrix}$$

$$\downarrow \ op3(3,4,-3376560y^3+41244y^2+264y-1)$$

$$\left(\times \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & 0 & 0 & 484549190928y^5-1033227360y^4-5189352y^3+6120y^2+17y \\ 168828y^2-180y-1 & 0 & 0 & 513052084512y^5-1094005440y^4-5494608y^3+6480y^2+18y \\ 0 & 168828y^2-180y-1 & 0 & 541554978096y^5-1154783520y^4-5799864y^3+6840y^2+19y \\ 0 & 0 & 1 & 3376560y^3-3600y^2-20y \end{bmatrix}$$

$$\downarrow \ op2(4,\frac{1}{168828y^2-180y-1})$$

$$\left(\times \frac{1}{168828y^2-180y-1}\right)\begin{bmatrix} 0 & 0 & 0 & 870076y^3-3060y^2-17y \\ 168828y^2-180y-1 & 0 & 0 & 3038904y^3-3240y^2-18y \\ 0 & 168828y^2-180y-1 & 0 & 3207732y^3-3420y^2-19y \\ 0 & 0 & 168828y^2-180y-1 & 3376560y^3-3600y^2-20y \end{bmatrix}$$

$$\downarrow \ \left(\times \frac{1}{168828y^2-180y-1}\right)$$

$$\begin{bmatrix} 0 & 0 & 0 & 17y \\ 1 & 0 & 0 & 18y \\ 0 & 1 & 0 & 19y \\ 0 & 0 & 1 & 20y \end{bmatrix}$$

5.4 計算結果と考察

数式処理システム REDUCE (version3.5) 上に **5.1.2** 項,**5.2.2** 項のアルゴリズムをインプリメントし、以下に示す単因子 L_N $(N = 3, \cdots, 20)$ を持つ、6 種類の 3 次から 20 次までの行列について計算を行なった。

$f_1 = \lambda - 3x,\ f_2 = \lambda - 5x,\ f_3 = \lambda - 7x,\ f_4 = \lambda - 11x$ とする。

[データ 5.1] 有理標準形がコンパニオン行列 1 個からなる場合 $(R = C_1)$ (数値行列)

i 次 $L_i = \{\underbrace{1, \cdots, 1}_{i-1\text{個}}, \lambda^i - 20\lambda^{i-1} - 19\lambda^{i-2} - \cdots - (20 - i + 2)\lambda - (20 - i + 1)\}$ $(i = 1, \cdots, 20)$

[データ 5.2] 有理標準形がコンパニオン行列 1 個からなる場合 $(R = C_1)$ (1 変数多項式行列)

i 次 $L_i = \{\underbrace{1, \cdots, 1}_{i-1\text{個}}, \lambda^i - 20x\lambda^{i-1} - 19x\lambda^{i-2} - \cdots - (20 - i + 2)x\lambda - (20 - i + 1)x\}$ $(i = 1, \cdots, 20)$

[データ 5.3] 有理標準形がコンパニオン行列 3 個からなる場合 $(R = C_1 \oplus C_2 \oplus C_3)$
($f_1 \sim f_4$ において、$x = 1$ としたときの数値行列)

3 次 $L_3 = \{f_1, f_1, f_1\}$,

4 次 $L_4 = \{{f_1}^2, f_1, f_1\}$,

5 次 $L_5 = \{{f_1}^2, {f_1}^2, f_1\}$,

6 次 $L_6 = \{{f_1}^2, {f_1}^2, {f_1}^2\}$,

7 次 $L_7 = \{{f_1}^3, {f_1}^2, {f_1}^2\}$,

8 次 $L_8 = \{{f_1}^3, {f_1}^3, {f_1}^2\}$,

9 次 $L_9 = \{{f_1}^3, {f_1}^3, {f_1}^3\}$,

10 次 $L_{10} = \{{f_1}^3 f_2, {f_1}^3, {f_1}^3\}$,

11 次 $L_{11} = \{{f_1}^3 f_2, {f_1}^3 f_2, {f_1}^3\}$,

12 次 $L_{12} = \{{f_1}^3 f_2, {f_1}^3 f_2, {f_1}^3 f_2\}$,

13 次 $L_{13} = \{{f_1}^3 {f_2}^2, {f_1}^3 f_2, {f_1}^3 f_2\}$,

14 次 $L_{14} = \{{f_1}^3 {f_2}^2, {f_1}^3 {f_2}^2, {f_1}^3 f_2\}$,

15 次 $L_{15} = \{{f_1}^3 {f_2}^2, {f_1}^3 {f_2}^2, {f_1}^3 {f_2}^2\}$,

16 次 $L_{16} = \{{f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2, {f_1}^3 {f_2}^2\}$,

17 次 $L_{17} = \{{f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2\}$,

18 次 $L_{18} = \{{f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2 f_3\}$,

19 次 $L_{19} = \{{f_1}^3 {f_2}^2 f_3 f_4, {f_1}^3 {f_2}^2 f_3, {f_1}^3 {f_2}^2 f_3\}$,

20 次　$L_{20} = \{{f_1}^3{f_2}^2f_3f_4, {f_1}^3{f_2}^2f_3f_4, {f_1}^3{f_2}^2f_3\}$

[データ 5.4]　有理標準形がコンパニオン行列 3 個からなる場合 ($R = C_1 \oplus C_2 \oplus C_3$)
（データ 5.3 において、$x = x$、つまり x を変数としたときの 1 変数多項式行列）

[データ 3.1]　有理標準形がコンパニオン行列 1 個からなる場合 ($R = C_1$)
（数値行列）（**3.4** 節参照）

[データ 3.4]　有理標準形がコンパニオン行列 3 個からなる場合 ($R = C_1 \oplus C_2 \oplus C_3$)
（1 変数多項式行列）（**3.4** 節参照）

実際の入力データ A_N ($N = 3, \cdots, 20$) は、**3.4** 節と同様にして作成した。要素はデータ 5.1 と 5.3 ではすべて整数、データ 5.2 と 5.4 ではすべて 1 変数整数係数多項式である。

SUN SPARC server10 (40SPECmark 相当以上、主記憶容量 128MB) 上の数式処理システム REDUCE (version3.5) 上で、32MB のメモリを使用して計算を行なった。従来のアルゴリズム [57],[71] を用いたプログラムと本研究の分数なしアルゴリズムを用いたプログラムによって、データ 5.1 ～ 5.4, 3.1,3.4 の行列の有理標準形を計算するのに要した CPU 時間（単位： ms（ミリ秒））とその比を表 5.1～ 5.6に示す（データ 3.1,3.4 については、**3.4** 節の計算結果と合わせるために、同時に変換行列も計算した場合の CPU 時間を示す）。CPU 時間の比は、従来のアルゴリズムによる CPU 時間 t_1 に対する分数なしアルゴリズムによる CPU 時間 t_2 の割合 t_2/t_1 である（小数点以下第 3 位四捨五入）。表 5.1～ 5.6の CPU 時間をそれぞれ図 5.1～ 5.6に示す。グラフの縦軸は CPU 時間（単位： ms）であり、横軸は入力データの行列の次数である。ただし、次数による CPU 時間の差が大きく、同じスケールでは表示しにくいので、13 次以降は省略する。また、表 5.1,5.2の CPU 時間の比 t_2/t_1 を図 5.7に、表 5.3,5.4の CPU 時間の比 t_2/t_1 を図 5.8に、表 5.5,5.6の CPU 時間の比 t_2/t_1 を図 5.9にグラフとして示す。グラフの縦軸は CPU 時間の比 t_2/t_1 であり、横軸は入力データの行列の次数である。

表 5.1: 有理標準形の CPU 時間とその比（データ 5.1：コンパニオン行列 1 個）

次数	従来 (t_1)	分数なし (t_2)	t_2/t_1
3	220	260	1.18
4	520	540	1.04
5	1070	1000	0.93
6	2070	1750	0.85
7	3780	2740	0.72
8	6329	4220	0.67
9	10540	6100	0.58
10	16879	8780	0.52
11	26280	12050	0.46
12	40540	16220	0.40
13	61560	21630	0.35
14	91409	28210	0.31
15	135290	37270	0.26
16	194420	48810	0.25
17	277330	62360	0.22
18	389800	79520	0.20
19	545710	101880	0.19
20	745900	127700	0.17

（CPU 時間の単位： ms）

図 5.1: 有理標準形の CPU 時間のグラフ（データ 5.1：コンパニオン行列 1 個）

表 5.2: 有理標準形の CPU 時間とその比（データ 5.2：コンパニオン行列 1 個）

次数	従来 (t_1)	分数なし (t_2)	t_2/t_1
3	360	420	1.17
4	2769	1380	0.50
5	8770	4370	0.50
6	20420	7340	0.36
7	41510	13950	0.34
8	76421	24321	0.32
9	134650	41161	0.31
10	224519	66050	0.29
11	369960	105390	0.28
12	587289	162199	0.28
13	942780	256290	0.27
14	1439500	389361	0.27
15	2245600	604870	0.27
16	3386840	917320	0.27
17	5063080	1381760	0.27
18	7496030	2054370	0.27
19	10938220	3035070	0.28
20	15615590	4372800	0.28

（CPU 時間の単位： ms）

図 5.2: 有理標準形の CPU 時間のグラフ（データ 5.2：コンパニオン行列 1 個）

表 5.3: 有理標準形の CPU 時間とその比（データ 5.3：コンパニオン行列 3 個）

次数	従来 (t_1)	分数なし (t_2)	t_2/t_1	最大ブロックの次数
3	230	260	1.13	1
4	470	850	1.81	2
5	970	970	1.00	
6	1410	1310	0.93	
7	2510	2320	0.92	3
8	4379	3730	0.85	
9	6050	4830	0.72	
10	9890	7200	0.73	4
11	15790	10280	0.65	
12	20510	12770	0.62	
13	31641	17740	0.56	5
14	47940	23500	0.49	
15	62310	28300	0.45	
16	88880	37470	0.42	6
17	129370	48170	0.37	
18	162001	56530	0.35	
19	226950	74060	0.33	7
20	320890	93149	0.29	

（CPU 時間の単位： ms）

図 5.3: 有理標準形の CPU 時間のグラフ（データ 5.3：コンパニオン行列 3 個）

表 5.4: 有理標準形の CPU 時間とその比（データ 5.4：コンパニオン行列 3 個）

次数	従来 (t_1)	分数なし (t_2)	t_2/t_1	最大ブロックの次数
3	270	270	1.00	1
4	690	840	1.22	2
5	5010	2910	0.58	
6	11470	4900	0.43	
7	35050	15420	0.44	3
8	89690	37090	0.41	
9	165530	64210	0.39	
10	366569	157540	0.43	4
11	848141	311560	0.37	
12	1335800	506470	0.38	
13	2581629	1111040	0.43	5
14	5574190	2062670	0.37	
15	8319400	3209460	0.39	
16	14648250	6734980	0.46	6
17	29191180	11656879	0.40	
18	41939831	18000590	0.43	
19	65370919	35359530	0.54	7
20	128409130	74293507	0.58	

（CPU 時間の単位： ms）

図 5.4: 有理標準形の CPU 時間のグラフ（データ 5.4：コンパニオン行列 3 個）

表 5.5: 有理標準形の CPU 時間とその比（データ 3.1：コンパニオン行列 1 個）

次数	従来 (t_1)	分数なし (t_2)	t_2/t_1
3	370	420	1.14
4	890	900	1.01
5	1890	1830	0.97
6	3750	3260	0.87
7	6940	5360	0.77
8	12320	8160	0.66
9	21150	12330	0.58
10	34329	17560	0.51
11	54660	24780	0.45
12	87770	34130	0.39
13	137020	47380	0.35
14	210990	64220	0.30
15	314070	85180	0.27
16	464070	114740	0.25
17	685930	155430	0.23
18	999580	206210	0.21
19	1427790	271770	0.19
20	2004630	357141	0.18

（CPU 時間の単位： ms）

図 5.5: 有理標準形の CPU 時間のグラフ（データ 3.1：コンパニオン行列 1 個）

表 5.6: 有理標準形の CPU 時間とその比（データ 3.4：コンパニオン行列 3 個）

次数	従来 (t_1)	分数なし (t_2)	t_2/t_1	最大ブロックの次数
3	271	290	1.07	1
4	980	1130	1.15	2
5	7921	4761	0.60	
6	10590	5220	0.49	
7	41671	18930	0.45	3
8	102650	47210	0.46	
9	108030	41400	0.38	
10	251930	117120	0.46	4
11	559440	245261	0.44	
12	466220	174820	0.37	
13	939351	466841	0.50	5
14	2090289	929120	0.44	
15	3193710	1410230	0.44	
16	5849300	3233730	0.55	6
17	12120990	6001420	0.50	
18	9512710	4544340	0.48	
19	17027991	10434530	0.61	7
20	34397550	19621999	0.57	

(CPU 時間の単位： ms)

図 5.6: 有理標準形の CPU 時間のグラフ（データ 3.4：コンパニオン行列 3 個）

図 5.7: 有理標準形の CPU 時間の比のグラフ（データ 5.1,5.2：コンパニオン行列 1 個）

図 5.8: 有理標準形の CPU 時間の比のグラフ（データ 5.3,5.4：コンパニオン行列 3 個）

図 5.9: 有理標準形の CPU 時間の比のグラフ（データ 3.1,3.4）

3.2 節で述べたように、有理標準形と同時に変換行列も計算することができる。上のデータについて有理標準形と変換行列を同時に計算した場合にかかる CPU 時間は、上に示した表の約 2 倍である。

表 5.1～5.6からわかるように、分数なしアルゴリズムの CPU 時間は、従来のアルゴリズムの CPU 時間を 1 としたとき、3 ～ 5 次の行列を除けば、[データ 5.1] 0.17 ～ 0.93、[データ 5.2] 0.27 ～ 0.50、[データ 5.3] 0.29 ～ 0.93、[データ 5.4] 0.37 ～ 0.58、[データ 3.1] 0.18 ～ 0.97、[データ 3.4] 0.37 ～ 0.61 になり、分数なしアルゴリズムの方が効率的であることがわかる。3 ～ 5 次の行列に対して効果が現れないのは、有理式が生じるのを回避する操作のオーバーヘッドのためである。

図 5.1,5.2,5.7から分かるように、表 5.1,5.2の比は、次数が大きくなるのにしたがって、次第に小さくなり、コンパニオン行列の次数が増加するのにつれて効率化の効果も増大していることがわかる。数値行列であるデータ 5.1 が、多項式行列であるデータ 5.2 よりも効率化の効果が現れはじめるのが遅いのは、有理数の通分・約分は有理式の通分・約分ほど時間がかからず、次数が小さいときには、従来の CPU 時間との差があまり出ないからである。また、データ 5.2 では 19,20 次程度で効果が減り始める。このことから、効率化の効果に限界があることが分かるが、効果の限界が何次であるかは、入力行列の要素やその配置、有理標準形のコンパニオン行列の分かれ方やその次数などに影響を受けるため、どんな行列に対しても一定であるわけではない。例えば、コンパニオン行列のタイプが同じデータ 5.1 と 5.2, データ 5.3 と 5.4 では、どちらとも多項式行列の方が効果が現れはじめるのが早く、ある程度一定になった後、効果が減りはじめるのも早い。さらに他のタイプのデータについても計算を行ない、検討する必要がある。

図 5.8から分かるように、表 5.3,5.4,5.6の比が表 5.1,5.2,5.5の比のように段階的に変化しないのは、有理標準形の中に含まれるコンパニオン行列の次数の増え方の影響を受けているためである。データ 5.3,5.4,3.4 では、最大ブロック C_1（1 番上の最も次数の大きいコンパニオン行列）の次数は、4,7, 10,13,16,19 次のときに 1 次ずつ増える。表 5.3,5.4の比は、次数を 4–6,7–9,10–12,13–15,16–18, 19–20 と 3 つずつのグループに分けたとき、その中では比の増減に一定の傾向がみられ、特に、行列の要素が多項式である表 5.4,5.6では、$3 \to 4, 6 \to 7, 9 \to 10, 12 \to 13, 15 \to 16, 18 \to 19$ で大幅に増加している。これは、最大ブロックの次数の増加に対応しており、この次数が 2 番目以降のブロックの次数よりも計算時間に影響を与えていることが分かる。

表 5.6, 図 5.6から分かるように、データ 3.4 では、8,11,17 次の CPU 時間がそれぞれ 9,12,18 次の CPU 時間よりも長くなっている。これは、8,11,17 次では、計算の途中で、中間結果の行列が、**5.2.2** 項で示した式 (5.1) のような形になり、0 でない $\star$ を消去する操作を行なっているため、その分の時間が余計にかかっているからである。この場合には、$\star$ はその操作によってすべて消去されるので、1 つ以上前のコンパニオン行列まで遡って計算をしなおすことはしていない。なお、データ 5.1 ～ 5.4,3.1,3.4 には、1 つ以上前のコンパニオン行列まで遡って計算をしなおすタイプの行列は含まれていないが、そのようなデータについても実験を行なう必要がある。

第 6 章

おわりに

本研究の目的は、1変数整数係数多項式を要素とする任意の n 次正方行列を、数式処理を用いた厳密計算によって Jordan 標準形をはじめとする4つの標準形に変換するアルゴリズムを示し、その改良について考察することであった。研究の成果として、4つの標準形と Jordan 標準形への変換行列の計算法をアルゴリズムとして記述し、数式処理システム REDUCE (version 3.5) 上にインプリメントを行なった。特に、Jordan 標準形の計算アルゴリズム・プログラムとしては、既知のアルゴリズム・数式処理システム上の既存のプログラムの共通の問題点を解決し、これまでのアルゴリズム・プログラムでは計算できなかった行列についても計算できるようにした。また、標準形の計算アルゴリズムの応用として連立線形差分方程式・定係数連立線形微分方程式の解法を考察し、アルゴリズムとして記述し、実験的に REDUCE 上にインプリメントを行なった。

さらに、本研究では、行列の基本変形において要素に有理式が現れるのを回避するため、Bareiss の分数なし Gauss の消去法 [7] から類推し、従来の有理標準形の計算アルゴリズムを効率化した、分数なしアルゴリズムを開発した。本稿の分数なしアルゴリズムを REDUCE 上に実験的にインプリメントし、いくつかの計算を行なった結果、特に、入力行列の要素が1変数多項式である場合には、計算時間の面において、従来のアルゴリズムよりも効率的であることが確かめられた。

今後の課題としては、次のようなことが考えられる。

1つめは、アルゴリズム全体の計算量の厳密な解析である。本稿では、REDUCE の組込関数（FACTORIZE、PREDUCE など）を用いている部分の解析が困難であることから CPU 時間による考察に留まっている。しかし、本稿の考察から分かるように、行列の次数だけでなく、有理標準形に含まれるコンパニオン行列の個数や固有値の定義多項式の次数によっても、CPU 時間は影響を受ける。したがって、CPU 時間についても、コンパニオン行列の個数や定義多項式の次数について細かく場合分けしたデータによる実験とその解析が必要であると思われる。

2つめは、有理標準形の分数なし計算法のさらなる効率化についてである。分数なし Gauss の消去法では、Single-step fraction-free elimination をそれ自体に適用し、冗長な部分を省いて複数のステップを1つにまとめることによって、Multistep integer-preserving Gaussian elimination[7] にすることができる。ただし、Single-step でも Multistep でも Bareiss の fraction-free elimi-

nation は行列の要素が整数または1変数多項式の場合には非常に有効であるが、多変数多項式の場合には、共通分母をくくり出したまま、次の消去過程で行なう加減乗算のときに、行列の各要素の成長が著しくなり、効率が悪くなる。その要素の成長の問題は、分数なし Gauss の消去法を効率化した Sasaki & Murao の Efficient Gaussian elimination[100] によって軽減することができる。この方法では、行列の対角要素を別の新しい変数に置き換え、その変数に比例する部分を選択的に計算する特別な乗算を定義することにより、分数なし計算のすべての段階ですべての要素の結果を計算しなくても、各段階の計算に必要な要素を構成する1段階前の要素だけを計算できるようにし、効率化を行なっている。分数なし計算法の次の段階として、Multistep integer-preserving Gaussian elimination と Efficient Gaussian elimination を有理標準形の分数なしアルゴリズムに適用することを検討したい。

謝辞

本論文を作成する機会を下さり、貴重なご指導をして下さった名取亮教授に深く感謝致します。常に、丁寧なご指導・ご助言をして下さり、公私ともに支援して下さった森継修一図書館情報大学助教授に心から感謝致します。

適切なご助言・ご討論をして下さった稲垣敏之教授、北川高嗣助教授、久野誉人助教授、佐々木建昭筑波大学数学系教授に深く感謝致します。

研究にご協力下さった櫻井鉄也講師、周偉東講師、筑波大学数値解析研究室の皆さんに感謝致します。また、山本毅雄図書館情報大学副学長、長谷川秀彦図書館情報大学講師、図書館情報大学総合情報処理センターの皆さんにも感謝致します。

参考文献

[1] 赤坂秀子; 桜井裕幸. 固有値計算の精度について：ダニレフスキー法とハウスホルダー法. 第23回数値解析シンポジウム. 仙台, 1994-06.

[2] Alagar, V. S.; Roy, A. K. A Comparative Study of Algorithms for Computing the Smith Normal Form of an Integer Matrix. International Journal of System Science. Vol.15, No.7, p.727-744 (1984)

[3] Anai, H. et al. : Computation of the Splitting Fields and the Galois Groups of Polynomials. Preprint (1993)

[4] Arcese, A. The Jordan Canonical Form of a Particular Matrix. American Mathematical Monthly. Vol.75, No.7, p.752 (1968)

[5] Ayres, F. Jr. 行列. 今井正隆訳. 東京, マグロウヒル出版, 1989, p.253-267. (マグロウヒル大学演習シリーズ) (ISBN 4-89501-120-8)

[6] Bancroft, S. A Note on the Jordan Canonical Form. Notices of the American Mathematical Society. Vol.22, No.1, p.A-83 (1975)

[7] Bareiss, E. H. Sylvester's Identity and Multistep Integer Preserving Gaussian Elimination. Mathematics of Computaton. Vol.22, p.565-578 (1968)

[8] Bareiss, E. H. Computational Solutions of Matrix Problems over an Integral Domain. Journal of the Institute of Mathematics and Its Applications. Vol.10, p.68-104 (1972)

[9] Bavely, C. A.; Stewart, G. W. An Algorithm for Computing Reducing Subspaces by Block Diagonalization. SIAM Journal on Numerical Analysis. Vol.16, No.2, p.359-367 (1979)

[10] Bekishev, G. A. A Method of Reducing Matrices to the Jordan Normal Form. U.S.S.R. Computational Mathematics and Mathematical Physics. Vol.5, No.4, p.189-195 (1965)

[11] Birkhoff, G.; MacLane, S. A Survey of Modern Algebra. 3rd ed. New York, The Macmillan Company, 1965, p.296-312.

[12] Browne, E. T. On the Reduction of a Matrix to a Canonical Form. American Mathematical Monthly. Vol.47, p.437–450 (1940)

[13] Buchberger, B. "6. Groebner Bases : An Algorithmic Method in Polynomial Ideal Theory". Multidimensional Systems Theory. Bose, N.K., ed. Dordrecht, D. Reidel Publishing Company, 1985, p.184–232.

[14] Cai, J. : Computing Jordan Normal Forms Exactly for Computing Matrices in Polynomial Time. Prepint (1995)

[15] Carstensen, C. Linear Construction of Companion Matrices. Linear Algebra and Its Applications. Vol.149, p.191–214 (1991)

[16] Char, B. W. et al. Maple V : Library Reference Manual, New York, Springer–Verlag, 1991, vi, 698p.

[17] Danilevskii, A. M. On the numerical solution of the secular equation. Matematicheskii sbornik. Vol.2(44), No.1, p.169–172 (1937)

[18] Demmel, J. The Condition Number of Equivalence Transformations That Block Diagonalize Matrix Pencils. SIAM Journal on Numerical Analysis. Vol.20, No.2, p.599–610 (1983)

[19] Dixion, J. D. et al. The Shuffle Algorithm and Jordan Blocks. Linear Algebra and Its Applications. Vol.142, p.154–165 (1990)

[20] Edelman, A.; Murakami, H. Polynomial Roots from Companion Matrix Eigenvalues. Mathematics of Computation. Vol.64, No.210, p.763–776 (1995)

[21] Egervary, E. Uber eine Konstruktive Methode zur Reduktion einer Matrix auf die Jordansche Normalform. Acta Mathematica Academiae Scientiarum Hungaricae. Vol.10, p.31–54 (1959)

[22] Faddeev, D. K.; Fadeeva, V. N. "4.6 アー・エム・ダニレフスキー法". 線型代数の計算法（上）. 小国力訳. 東京, 産業図書, 1970, p.281–290.

[23] Farahat, H. K. A Note on the Classical Canonical Form. The Jounal of The London Mathematical Society. Vol.32, p.178–181 (1957)

[24] Filippov, A. F. A Short Proof of the Theorem on the Reduction of a Matrix to Jordan Form. Moscow University Mathematics Bulletin. Vol.26, p.70–71 (1971)

[25] Fletcher, R.; Sorensen, D. C. An Algorithmic Derivation of the Jordan Canonical Form. American Mathematical Monthly. Vol.90, p.12–16 (1983)

[26] Frank, W. L. Computing Eigenvalues of Complex Matrices by Determinant Evaluation and by Methods of Danilewski and Wielandt. Journal. Society of Industrial and Applied Mathematics. Vol.6, No.4, p.378–392 (1958)

[27] Gantmacher, F. R. The Theory of Matrices Vol.1. 2nd ed. New York, Chelsea Publishing Company, 1959,374p. (ISBN 0–8284–1131–X)

[28] Geddes, K. O. et al. "Chapter 9. Solving Systems of Equations". Algorithms for Computer Algebra. Boston, Kluwer Academic Pub., 1992, p.389–427.

[29] Giesbrecht, M. Fast Algorithms for Rational Forms of Integer Matrices. Proceedings of ISSAC'94. Oxford, 1994-06, New York, ACM, 1994, p.305–311.

[30] Giesbrecht, M. Fast Algorithms for Matrix Normal Forms. Proceedings of 33rd Annual Symposium on Foundatons of COmputer Science. Pittsburgh, 1992-10, Washington, IEEE Computer Society Press, 1992, p.121–130.

[31] Gil, I. Computation of the Jordan Canonical Form of a Square Matrix (Using the Axiom Programming Language). Proceedings of ISSAC'92. Wang, P. S., ed. California, 1992-04, ACM. New York, 1992, p.138–145.

[32] Glüsing–Lüerssen, H.; Hinrichsen, D. A Jordan Control Canonical Form for Singular System. International Journal of Control. Vol.48, No.5, p.1769-1785 (1988)

[33] Golub, G. H.; Wilkinson, J. H. Ill-conditioned Eigensystems and the Computaton of the Jordan Canonical Form. SIAM Review. Vol.18, No.4, p.578–619 (1976)

[34] Gow, R. Generation of SL(n,**Z**) by a Jordan Unipotent Matrix and Its Transpose. Linear Algebra and Its Applications. Vol.181, p.63–71 (1993)

[35] Gregory, R. T. Defective and Derogatory Matrices. SIAM Review. Vol.2, No.2, p.134–139 (1960)

[36] Gregory, R. T.; Krishnamurthy, E. V. Methods and Applications of Error–Free Computation. New York, Springer–Verlag, 1984, p.180-185. (ISBN 0–387–90967–2)

[37] Hansen, E. R. On the Danilewski Method. Journal. ACM. Vol.10, No.1, p.102–109 (1963)

[38] Hansen, T.; Mullen, G. L. Primitive Polynomial over Finite Fields. Mathematics of Computations. Vol.59, No.200, p.639–643 (1992)

[39] Hartl, J. Ein einfacher Induktionsbeweis fur die Existenz der Jordan–Normalform. Archiv det Mathematik. Vol.50, p.323–327 (1988)

[40] Hayton, G. E. at al. Infinite Elementary Divisors of a Matrix Polynomial and Implications. International Journal of Control. Vol.47, No.1, p.53–64 (1988)

[41] Hearn, A. C. REDUCE User's Manual Version 3.4. Rand Pub., 1991, 206p.

[42] Hinrichsen, D. A Jordan Canonical Form for Reachable Linear Systems. Linear Algebra and Its Applications. Vol.122/123/124, p.489–524 (1989)

[43] Hong, Y. The Jordan Canonical Form of a Product pf a Hermitian and a Positive Semidefine Matrix. Linear Algebra and Its Applications. Vol.147, p.373–386 (1991)

[44] Howell, J. A. An Algorithm for Exact Reduction of a Matrix to Frobenius Form Using Modular Arithmetic. I. II. Mathematics of Computation. Vol.27, No.124, p.887–920 (1973)

[45] 伊理正夫，韓太舜．線形代数：行列とその標準形．東京，新曜社， 1977, vii,265p. （シリーズ新しい応用の数学， 16）(ISBN 4-316-37670-5)

[46] Jacobson, N. Lectures in Abstract Algebra Vol.1 Princeton, New Jarsey, D. Van Nostrand Company, 1951.

[47] Jacobson, N. Lectures in Abstract Algebra Vol.2. Princeton, New Jarsey, D. Van Nostrand Company, 1953, p.63–114.

[48] Jacobson, N. Lectures in Abstract Algebra Vol.3. Princeton, New Jarsey, D. Van Nostrand Company, 1964.

[49] Jiang, E. An Algorithm for Computing Transformation Matrix of the Jordan Canonical Form of any Matrix. Numerical Mathematics: A Jornal of Chinese Universities. Vol.1, No.1, p.1–10 (1979)

[50] Johnson, C. R. Explicit Jordan Form for Certain Block Triangular Matrices. Linear Algebra and Its Applications. Vol.150, p.297–314 (1991)

[51] Kågstrom, B.; Ruhe, A. Algorithm 560 JNF, An Algorithm for Numerical Computation of the Jordan Normal Form of a Complex Matrix [F2]. ACM Transactions on Mathematical Software, Vol.6, No.3, p.437–443 (1980)

[52] Kågstrom, B.; Ruhe, A. An Algorithm for Numerical Computation of the Jordan Noraml Form of a Complex Matrix. ACM Transactions on Mathematical Software, Vol.6, No.3, p.398–419 (1980)

[53] Kaltofen, E. et al. Fast Parallel Algorithms for Similarity of Matrices. Proceedings of the 1986 Symposium on Symbolic and Algebraic Computation. Char, B.W., ed. Waterloo, 1986-06, ACM. New York, 1986, p.65-70.

[54] Kaltofen, E. et al. Fast Parallel Computation of Hermite and Smith Forms of Polynomial Matrices. SIAM Journal on Algebraic and Discrete Methods. Vol.8, No.4, p.683-690 (1987)

[55] Kaltofen, E. et al. Mr. Smith Goes to Las Vegas : Randomaized Parallel Computation of the Smith Noraml Form of Polynomial Matrices. Proceedings of EUROCAL'87. Davenport, J. H; Goos, G; Hartmanis, J., eds. Leipzig, 1987-06, New York, Springer-Verlag, 1989, p.317-322. (Lecture Notes in Computer Science, 378) (ISBN 0-387-51517-8)

[56] Kaltofen, E.; Krishnamoorthy, M. S. Parallel Algorithms for Matrix Normal Forms. Linear Algebra and Its Applications. Vol.136, p.189-208 (1990)

[57] 韓太舜，伊理正夫．ジョルダン標準形．東京，東京大学出版会， 1982, iv,206p. （UP 応用数学選書， 8） (ISBN 4-13-064068-2)

[58] Kannan, R.; Bachem, A. Polynomial Algorithms for Computing The Smith and Hermite Normal Forms of An Integer Matrix. SIAM Journal on Computing. Vol.8, No.4, p.499-507 (1979)

[59] Kannan, R. Solving Systems of Linear Equations over Polynomials. Theoretical Computer Science, Vol.39, p.69-88 (1985)

[60] Kaufman, I. The Inversion of the Vandermonde Matrix and the Transformation to the Jordan Canonical Form. IEEE Transactions on Automatic Control. Vol.AC14, No.16, p.774-777 (1969)

[61] Kaufman, I.; Roe, P. H. On Systems Decsribed by the Companion Matrix. IEEE Transactions on Automatic Control. Vol.AC15, No.6, p.692-693 (1970)

[62] Keller-Gehrig, W. Fast Algorithms for the Characteristic Polynomials. Theoretical Computer Science. Vol.36m p.309-317 (1985)

[63] Kitamoto, T. : Computation of Jordan Decomposition in the Form of Power Series. Prepint (December, 1994)

[64] 北本卓也．擬 Jordan 分解とその応用．日本応用数理学会平成 7 年度年会講演予稿集．応用数理学会編．岡山, 1995-09, p.202-203.

[65] Košir, T. Common Jordan Chains of Matrices. Linear Algebra and Its Applications. Vol.99, p.253–258 (1988)

[66] Kreisselmeier, G. On the Determination of the Jordan Form of Matrix. IEEE Transactions on Automatic Control. Vol.18, p.686–687 (1973)

[67] Krishnamurthy, E. V. Error–Free Polynomial Matrix Computations. New York, Springer–Verlag, 1985, 154p. (Texts and Monographs in Computer Science)

[68] 栗山和子. 数式処理システム REDUCE による行列の Jordan 標準形の構成法. 茨城, 図書館情報大学, 1991, 143p. 卒業研究論文.

[69] 栗山和子. 数式処理による行列の標準形の計算法とその応用. 茨城, 図書館情報大学, 1993, 115p. 修士論文.

[70] 栗山和子, 森継修一. 数式処理による行列の Jordan 標準形の構成法. 図書館情報大学研究報告. Vol.10, No.1, p.61–85 (1991)

[71] 栗山和子, 森継修一. 数式処理による行列の標準形の計算法. 記号・数式処理と先端的科学技術計算. 理化学研究情報科学研究室編. 埼玉, 1992–03, 理化学研究所. 埼玉, 1992, p.35–41. (理研シンポジウム要旨集)

[72] 栗山和子; 森継修一. 数式処理による行列の標準形の厳密計算法. 日本応用数理学会論文誌. Vol.4, No.2, p.183–194 (1994)

[73] Landau, S. Factoring Polynomials over Algebraic Number Fields. SIAM Journal on Computing. Vol.14, No.1, p.184–195 (1985)

[74] Lenstra, A. K. Factoring polynomials over algebraic number fields. Computer Algebra:Proceedings of EUROCAL'83. Van Hulzen, J. A.; Goos, G; Hartmanis, J., eds. London, 1983–03, New York, Springer–Verlag, 1983, p.245–254. (Lecture Notes in Computer Science, 162) (ISBN 0–387–12868–9)

[75] Lesky, P. Uber die Konstruktion der Jordanschen Normalform. Elemente der Mathematik. Vol.40, No.5, p.105–109 (1985)

[76] Loos, R. Computing in Algebraic Extension. Computer Algebra : Symbolic and Algebraic Computation. 2nd ed. New York, Springer–Verlag, 1982, p.173–187.

[77] Loos, R. Generalized Polynomial Remainder Sequences. Computer Algebra : Symbolic and Algebraic Computation. 2nd ed. New York, Springer–Verlag, 1982, p.115–137.

[78] Luneburg, H. On the Rational Normal Form of Endomorphisms : A Primer to Constructive Algebra. Mannheim, BI-Wissenschaftsverlag, 1987, 397p.

[79] Mathieu, M. H.; Ford, D. On p-adic Computation of the Rational Form of a Matrix. Journal of Symbolic Computation. Vol.10, p.453–464 (1990)

[80] Mignotte, M. Some Useful Bounds. Computer Algebra : Symbolic and Algebraic Computation. 2nd ed. New York, Springer–Verlag, 1982, p.259–263.

[81] 森継修一　ほか．単因子の計算法．数式処理と数学研究への応用．京都大学数理解析研究所編．京都，1984–12，京都大学数理解析研究所．京都，1985，p.1–14．（数理解析研究所講究録，551）（REDUCE プログラミング資料第二集．東京大学大型計算機センター編）

[82] 森継修一，栗山和子．行列の Jordan 標準形の数式処理による厳密計算法．日本応用数理学会論文誌．Vol.2, No.1, p.91–103 (1992)

[83] Mukhopadhyay, A.; Alagar, V. S. Exact Computation of the Characteristic Polynomial of an Integer Matrix. Proceedingsf AAECC-3. Goos, G; Hartmanis, J., eds. Grenoble, 1985–06, AAECC, New York, Springer–Verlag, 1986, p.316–324. (Lecture Notes in Computer Science, 229) (ISBN 0-387-16776-5)

[84] Murota, K. On the Smith Normal Form of Structured Polynomial Matrices. SIAM Journal on Matrix Analysis and Applications. Vol.12, No.4, p.747–765 (1991)

[85] Newman, M. Invariant Factors of Combinatorial Matrices. Israel Journal of Mathematics. Vol.10, No.1, p.126–130 (1971)

[86] Nijenhuis, A. Smith Canonical Forms of Integer Matrices. Notices of the American Mathematical Society. Vol.21, No.1, p.A-389 (1974)

[87] 落合豊行，永友清和．REDUCE による線形代数．東京，近代科学社，1990，iv,161p．（コンピュータと数学，2）(ISBN 4–7649–0164–1)

[88] Pontryagin, L. S. 常微分方程式．新版．木村俊房，千葉克裕訳．東京，共立出版，1968，p.301–322. (ISBN 4–320–01038–8)

[89] Putzer, E. J. Avoiding the Jordan Canonical Form in the Discussion of Linear System with Constant Coefficients. American Mathematical Monthly. Vol.73, No.1, p.2–7 (1966)

[90] Roberts, L. G. Higher Derivations and the Jordan Canonical Forms of the Companion Matrix. Canadian Mathematical Bulletin. Vol.15, No.1, p.143–144 (1972)

[91] Robinson, D. W. The Generalized Jordan Canonical Form. American Mathematical Monthly. Vol.77, No.4, p.393–395 (1970)

[92] Rubin, W. B. A Simple Method for Finding the Jordan Form of a Matrix. IEEE Transactions on Automatic Control. Vol.17, p.145–146 (1972)

[93] Sarma, I. G. et al. A Note on the Transformation from Schwarz to Jordan Canonical Form. IEEE Transactions on Automatic Control. Vol.AC13, p.310 (1968)

[94] Sarma, I. G. et al. On the Transformation to Schwarz Canonical Form. IEEE Transactions on Automatic Control. Vol.AC13, p.311–312 (1968)

[95] 佐々木宏. ジョルダン標準形と可換なすべての行列の行列式. 電子通信学会論文誌. Vol.J59–A, No.8, p.679–681 (1976)

[96] 佐々木宏. 行列のジョルダン標準形への新しい変換法. 電子通信学会論文誌. Vol.J60–A, No.6, p.579–586 (1977)

[97] 佐々木建昭. 数式処理. 東京, 情報処理学会, 1981, v,111p. (情報処理叢書, 7)

[98] 佐々木建昭 ほか. 数式処理システム. 東京, 昭晃堂, 1986, 211p. (ソフトウェア講座, 36) (ISBN 4–7856–3536–3)

[99] 佐々木建昭 ほか. 計算機代数と計算幾何. 東京, 岩波書店, 1993, 166p. (岩波講座 応用数学, 5 [方法 9]) (ISBN 4–00–010515–9)

[100] Sasaki, T.; Murao, H. Efficient Gaussian Elimination Method for Symbolic Determinants and Linear Systems. ACM Transactions on Mathematical Software. Vol.8, No.3, p.277–289 (1982)

[101] Smiley, M. F. The Rational Canonical Form of a Matrix. American Mathematical. Vol.49, p.451–454 (1942)

[102] Smith, H. J. S. On System of Linear Indeterminate Equation and Congruences. Philosophical transactions of the Royal Society of London. Vol.151, p.293–326 (1861)

[103] Strauss, N. Jordan Forms of Binomial Coefficient Matrix over $\mathbf{Z}_p$. Linear Algebra and Its Applications. Vol.90, p.65–72 (1987)

[104] Strauss, N. "Algorithm and Implementation for Computaion of Jordan Form over $A[x_1,...,x_m]$". Computers and Mathematics. Kaltofen, E.; Watt, S. M., eds. New York, Springer–Verlag, 1989, p.21–26.

[105] Su, C. An Algorithm for Putting a Matrix into a Jordan Form. Notices of the American Mathematical Society. Vol.18, No.5, p.807 (1971)

[106] 杉浦光雄. Jordan 標準形と単因子論 I. 東京, 岩波書店, 1987, v,160p. (岩波講座 基礎数学, 2) (ISBN 4-00-010002-5)

[107] 杉浦光雄. Jordan 標準形と単因子論 II. 東京, 岩波書店, 1987, v,161-320p. (岩波講座 基礎数学, 2) (ISBN 4-00-010002-5)

[108] Symbolics, ed. MACSYMA : Reference Manual Version 13. U.S.A., 1988.

[109] 竹島卓, 横山和弘. 連立代数方程式の一解法：剰余環上の線形写像の固有ベクトルの利用. 数式処理通信. Vol.6, No.4. p.27-36 (1990)

[110] 竹島卓 ほか. 行列の有理標準形の一計算法. 数式処理と数学研究への応用. 京都大学数理解析研究所編. 京都, 1990-11, 京都大学数理解析研究所. 京都, 1991, p.18-21. (数理解析研究所講究録, 753)

[111] Trager, B. M. Algebraic Factoring and Rational Function Integration. Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation. Jenks, R. D., ed. New York, 1976-08, ACM. New York, 1976, p.216-226.

[112] Unger, H. Zur Praxis der Biorthonormierung von Eigen und Hauptvektoren. Zeitschrift fur angewandte Mathematik und Mechanik. Vol.33, No.10/11, p.319-331 (1953)

[113] Väliaho, H. An Elementary Approach to the Jordan Form of a Matrix. American Mathematical Monthly. Vol.93, No.9, p.711-714 (1986)

[114] Varga, G. Y. Nonzero Eigenvalues of Singular Matrices, Jordan Normal Form of Matrices. Numerical Methods. Rozsa, P. ed. Keszthely, 1977, Bolyai Janos Mathematikai Tarsulat, Amsterdam, 1980, p.619-625. (Colloquia Mathematica Societatis Janos Bolyai, 22)

[115] Von zur Gathen, J. A Polynomial Factorization Challenge. ACM SIGSAM Bulletin. Vol.26, No.2, p.22-24 (1993)

[116] Wall, B. Computing in Algebraic Extensions Represented by Groebner Bases. Linz, Austria, Research Institute for Symbolic Computation, Johannes Kepler University, 1992, RISC-Linz Report Series No.92-49, 19p.

[117] Wallach, S. A Simple Derivation of Jordan Canonical Matrices over Arbitary Fields. American Mathematical Monthly. Vol.72, No.6, p.614-618 (1965)

[118] Werner, W. A Generalized Companion Matrix of a Polynomial and Some Applications. Linear Algebra and Its Applications. Vol.55, p.19–36 (1983)

[119] Wolfram, S. Mathematica : A System for Doing Mathematics by computer. 2nd ed. Redwood City, Calif., Addison-Wesley Publishing Company, 1991, xxii, 961p. (ISBN 0–201–51502–4; 0–201–51507–5(pbk))

[120] ウルフラム, スティーブン. Mathematica : A System for Doing Mathematics by Computer（日本語版）. 2nd ed. 東京, アジソン・ウェスレイ・パブリッシャーズ・ジャパン（アジソン・ウェスレイ）, 1992, xxii, 937p.

[121] 横山和弘　ほか. Qの代数的拡大体の原始元の算出法について. 数理の世界における計算：数式処理と数値計算との接続. 富士通国際情報社会科学研究所編. 1987-09, 富士通国際情報社会科学研究所. 静岡, 1987, p.34–50.（夏期シンポジウム報告書）

[122] Yokoyama, K. et al. : Computing Primitive Elements of Extension Fields. Preprint (March, 1988)

[123] Yokoyama, K. et al. Computing Primitive Elements of Extension Fields. Journal of Symbolic Computation. Vol.8, p.553–580 (1989)

[124] Yokoyama, K.; Takeshima, T. : On Hensel Construction of Eigenvalues and Eigenvectors of Matrices with Polynomial Entries. Preprint (1993)

付録 A

有理標準形の構成法

文献 [57] の有理標準形を求めるアルゴリズムの不十分な点を補ったアルゴリズムを示す。同文献のアルゴリズムを補った部分については、注意）として文中に示す。

5 章で定義した基本変形を用いて、操作を施された A を操作の回数に関係なく $\tilde{A}$、その要素を $\tilde{a}_{ij}$ と表すことにする。

[1] ここでは、A に基本変形を施し、左上に可能な限り大きな 1 番目のコンパニオン行列 C_1 が現れている行列 $\tilde{A}$ を作る。

もし、$a_{21} = a_{31} = \cdots = a_{n1} = 0$ ならば、$C_1 = [a_{11}], p = 1, m_1 = 1$ とおいて [2] へ行く。そうでなければ、$a_{q1} \neq 0$ $(q \neq 1)$ なる要素を 1 つ選び、A の第 2 行と第 q 行の入替えおよび 第 2 列と第 q 行の入替え $(op1(2,q))$ を行ない、a_{q1} を A の $(2,1)$ 要素にもってくる。（$q = 2$ のときはこの操作はとばす。）この行列 $\tilde{A}$ の第 2 行を $\tilde{a}_{21}$ で割り第 2 列に $\tilde{a}_{21}$ を掛ければ $(op2(2,\tilde{a}_{21}))$、$(2,1)$ 要素が 1 である行列が得られる。この行列に対して、$i \neq 2$ なるすべての i に対して、第 i 行から第 2 行の $\tilde{a}_{i1}$ 倍を引き同時に第 2 列に第 i 列の $\tilde{a}_{i1}$ 倍を加えると $(op3(i,2,-\tilde{a}_{i1}))$、第 1 列は $(2,1)$ 要素だけが 1 で他の要素がすべて 0 の次のような行列 $\tilde{A}$ が得られる。

$$\tilde{A} = \begin{bmatrix} 0 & \star & \star & \cdots & \star \\ 1 & \star & \star & \cdots & \star \\ 0 & \star & \star & \cdots & \star \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \star & \star & \cdots & \star \end{bmatrix}$$

ただし、$\star$ は 0 とは限らない数を表す。

ここで、$\tilde{a}_{32} = \tilde{a}_{42} = \cdots = \tilde{a}_{n2} = 0$ ならば、

$$C_1 = \begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{12} \\ \tilde{a}_{21} & \tilde{a}_{22} \end{bmatrix} = \begin{bmatrix} 0 & \tilde{a}_{12} \\ 1 & \tilde{a}_{22} \end{bmatrix}$$

とおき、$p=1, m_1=2$ として [2] へ行く（C_1 は 2 次のコンパニオン行列である）。そうでなければ、第 2 列の非零要素 $\tilde{a}_{q2} \neq 0$ $(q \geq 3)$ を行の入替えおよび列の入替え $(op1(3,q))$ によって $(3,2)$ 要素に移す。

この結果の行列において、第 3 行を $\tilde{a}_{32}$ で割り第 3 列に $\tilde{a}_{32}$ を掛ければ $(op2(3,\tilde{a}_{32}))$、$(3,2)$ 要素が 1 である行列が得られる。$i \neq 3$ なるすべての i に対して、第 i 行から第 3 行の $\tilde{a}_{i2}$ 倍を引き同時に第 3 列に第 i 列の $\tilde{a}_{i2}$ 倍を加えれば $(op3(i,3,-\tilde{a}_{i2}))$、次のような行列 $\tilde{A}$ を得る。

$$\tilde{A} = \begin{bmatrix} 0 & 0 & \star & \cdots & \star \\ 1 & 0 & \vdots & & \vdots \\ 0 & 1 & \vdots & & \vdots \\ \vdots & 0 & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \star & \cdots & \star \end{bmatrix}$$

同様な操作を第 3 列以降に対しても可能な限り続けて行けば、一般に、次のような形の行列 $\tilde{A}$ を得る。

$$\tilde{A} = \begin{array}{c} \\ \uparrow \\ m_1 \\ \\ \downarrow \\ \\ \end{array} \begin{array}{c} \leftarrow \quad m_1 \quad \rightarrow \\ \left[\begin{array}{ccccc|ccc} 0 & \cdots & \cdots & 0 & \star & \star & \cdots & \star \\ 1 & & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \ddots & & \vdots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 1 & \star & \star & \cdots & \star \\ \hline 0 & \cdots & \cdots & \cdots & 0 & \star & \cdots & \star \\ \vdots & & & & \vdots & \vdots & & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & \star & \cdots & \star \end{array}\right] \end{array}$$

この行列 $\tilde{A}$ の左上に現れている m_1 次の正方行列（これを C_1 と書く）はコンパニオン行列であり、第 m_1 列の第 m_1+1 行以降の要素がすべて 0 となっているので、上記の操作は第 m_1 列で停止している。

そこで、$p=1$ として [2] へ行く。

[2] ここに入ってくるときの行列 $\tilde{A}$ の形は次のようになっている。

$$\tilde{A} = \left[\begin{array}{ccc|ccc} C_1 & & O & \star & \cdots & \star \\ & C_2 & & \vdots & & \vdots \\ & & \ddots & \vdots & & \vdots \\ O & & C_p & \star & \cdots & \star \\ \hline & & & \star & \cdots & \star \\ & O & & \vdots & & \vdots \\ & & & \star & \cdots & \star \end{array}\right]$$

ただし、$C_1, C_2, \cdots, C_p$ はコンパニオン行列（C_i の最小多項式を $\phi_i(x)$、その次数を m_i と書く）であり、$\phi_i(x)$ は $\phi_{i-1}(x)$ を割り切る $(i = 2, \cdots, p)$。$\tilde{A} = C_1 \oplus C_2 \oplus \cdots \oplus C_p$ となっている場合（$m_1 + \cdots + m_p = n$ の場合）には、これで終了する。そうでない場合には、$\tilde{A}$ の第 $(m_1 + \cdots + m_p + 1)$ 列において第 $(m_1 + \cdots + m_p + 2)$ 行以降の 0 でない要素を探す。0 でない要素があれば、それを第 $(m_1 + \cdots + m_p + 2)$ 番目以後の行の入替えと列の入替えを適当に行なうことにより $(op1(m_1 + \cdots + m_p + 2, q))$、$(m_1 + \cdots + m_p + 2, m_1 + \cdots + m_p + 1)$ 要素にもってくる。

この行列に対し [1] と同様の操作 $(op1, op2, op3)$ を第 $(m_1 + \cdots + m_p + 1)$ 列から始めてそれ以後の列に繰り返し施し、次のような行列 $\tilde{A}$ を得る。（$\tilde{A}$ の第 $(m_1 + \cdots + m_p + 1)$ 列の第 $(m_1 + \cdots + m_p + 2)$ 行以後がみな 0 である場合には、$\tilde{A}$ は、そのまま、次のような行列になっているので、$m_{p+1} = 1$ とする。）

$$\tilde{A} = \left[\begin{array}{ccc|cccc|ccc} C_1 & & O & & & & \triangle & \star & \cdots & \star \\ & \ddots & & & O & & \vdots & \vdots & & \vdots \\ O & & C_p & & & & \triangle & \vdots & & \vdots \\ \hline & & \uparrow & 0 & \cdots & 0 & \star & \vdots & & \vdots \\ & & & 1 & \ddots & \vdots & \vdots & \vdots & & \vdots \\ & & m_{p+1} & & \ddots & 0 & \vdots & \vdots & & \vdots \\ & O & \downarrow & O & & 1 & \star & \vdots & & \vdots \\ \hline & & & \leftarrow & m_{p+1} & & \rightarrow & \vdots & & \vdots \\ & & & & O & & & \star & \cdots & \star \end{array}\right]$$

$\tilde{A}$ の第 $(p+1)$ 番目の対角ブロック（m_{p+1} 次のコンパニオン行列）を C_{p+1} とおく。$\tilde{A}$ で △ 印をつけた個所（列ベクトル）のうちのブロック C_i の m_i 個の行に対応する部分（すなわち、第 $(m_1 + \cdots + m_{p+1})$ 列の第 $(m_1 + \cdots + m_{i-1} + 1)$ 行以後第 $(m_1 + \cdots + m_i)$ 行まで）を $\mathbf{e}_i$ で表す $(i = 1, \cdots, p)$。$\{\mathbf{e}_1, \cdots, \mathbf{e}_p\}$ のすべてが零ベクトル $\mathbf{0}$ であれば [3] へ行く。そうでなければ、$\mathbf{e}_i \neq \mathbf{0}$ なる最小の i を q とおいて、次の操作を行なう。

(a) $m_{p+1} \geq m_q$ ならば、第 $(m_1+\cdots+m_{q-1}+1)$ 行と $(m_1+\cdots+m_p+1)$ 行を入替え、同時に第 $(m_1+\cdots+m_{q-1}+1)$ 列と第 $(m_1+\cdots+m_p+1)$ 列を入替えて $(op1(m_1+\cdots+m_{q-1}+1, m_1+\cdots+m_p+1))$、$p$を$q-1$に等しいとおく。$p=0$ならば[1]の先頭へ戻る注意)。そうでなければ [2] の先頭へ戻る。

(b) $m_{p+1} \leq m_q - 1$ ならば、各 $j = 1, 2, \cdots, r$ に対して順に以下の操作 (i),(ii) を行なう $(r = m_q - m_{p+1})$。

(i) その時点での行列の $(m_1+\cdots+m_q-j+1, m_1+\cdots+m_{p+1})$ 要素を α_j とおく。

(ii) 各 $k = 1, 2, \cdots, m_{p+1}$ に対して順に、第 $(m_1+\cdots+m_{p+1}+1-k)$ 列から第 $(m_1+\cdots+m_q-k-j+1)$ 列の α_j 倍を引き、同時に第 $(m_1+\cdots+m_q-k-j+1)$ 行に第 $(m_1+\cdots+m_{p+1}+1-k)$ 行の α_j 倍を加える $(op3(m_1+\cdots+m_q-k-j+1, m_1+\cdots+m_{p+1}+1-k, \alpha_j))$。

各 j, k に対する操作 (i),(ii) を終了したときに得られる行列 $\tilde{A}$ において、第 $(m_1+\cdots+m_{p+1})$ 列のうちでブロック C_q に対応する部分の中の残りの部分、すなわち、第 $(m_1+\cdots+m_{q-1}+1)$ 行から第 $(m_1+\cdots+m_q-r)$ 行まで、もすべて 0 となっている場合には、$\mathbf{e}_i \neq \mathbf{0}$ $(i > q)$ なる最小の i を探し、このような i が存在するなら、これをあらためて q とおき m_{p+1} と m_q-1 の大小に応じて [2] の (a) あるいは (b) へ戻る。$\{\mathbf{e}_1, \cdots, \mathbf{e}_p\}$ がすべて 0 であるなら [3] へ行く。$\tilde{A}$ の第 $(m_1+\cdots+m_{p+1})$ 列の第 $(m_1+\cdots+m_{q-1}+1)$ 行から第 $(m_1+\cdots+m_q-r)$ 行までに非零要素が少なくとも一つある場合には、第 $(m_1+\cdots+m_{q-1}+1)$ 行と第 $(m_1+\cdots+m_p+1)$ 行とを入替え同時に第 $(m_1+\cdots+m_{q-1}+1)$ 列と第 $(m_1+\cdots+m_p+1)$ 列とを入替えて $(op1(m_1+\cdots+m_{q-1}+1, m_1+\cdots+m_p+1))$、$p$を$q-1$に等しいとおく。$p=0$ならば[1]の先頭へ戻る注意)。そうでなければ [2] の先頭へ戻る。

[3] ここに入ってくるときの行列 $\tilde{A}$ は次のようになっている。

$$\tilde{A} = \left[\begin{array}{cccc|ccc} C_1 & & & O & \star & \cdots & \star \\ & C_2 & & & \vdots & & \vdots \\ & & \ddots & & \vdots & & \vdots \\ & & C_p & & \vdots & & \vdots \\ O & & & C_{p+1} & \star & \cdots & \star \\ \hline & & O & & \star & \cdots & \star \\ & & & & \vdots & & \vdots \\ & & & & \star & \cdots & \star \end{array}\right]$$

ただし、C_i は m_i 次の最小多項式 $\phi_i(x)$ をもつコンパニオン行列 $(i = 1, \cdots, p, p+1)$ で、$\phi_j(x)$ は $\phi_{j-1}(x)$ を割り切る $(j = 2, \cdots, p)$。

$\phi_{p+1}(x)$ が $\phi_p(x)$ を割り切らなければ、第 $(m_1+\cdots+m_p+1)$ 列を第 $(m_1+\cdots+m_{p-1}+1)$ 列に加え同時に第 $(m_1+\cdots+m_p+1)$ 行から第 $(m_1+\cdots+m_{p-1}+1)$ 行を引き $(op3(m_1+\cdots+m_p+1, m_1+\cdots+m_{p-1}+1, -1))$、$p-1$ を p とおいて、[2] へ行く。割り切る場合には、$m_1+\cdots+m_p+m_{p+1}=n$ ならば、これで終了し、そうでなければ $p+1$ を p とおいて [2] へ行く。

注意）文献 [57] には、この 2ヶ所の部分がないので、$p=0$ の場合、計算が終了しない。

付録 B

プログラムリスト

3.2節、3.5.1.2 項、3.5.2.2 項、4.2節、5.2.2節のアルゴリズムに基づき、数式処理言語 REDUCE3.5 によって記述したプログラムを示す。

B.1 標準形の計算プログラム

ここでは、有理標準形を計算する方法として分数なしアルゴリズムを用いた、有理標準形、第 2 の標準形、Jacobson 標準形、Jordan 標準形を計算するプログラムを示す（3.2節、4.2節、5.2.2節のアルゴリズムに基づくプログラム）。

```
on comp;

procedure swapmat(m,k);
   begin scalar i,j,temp;
      % jm,n,sm11,sm12 : global ;
      % n : dimension of matrix jm ;
      % sm11,sm12 : transformation matrices for rational normal form ;
      %             sm11*a*sm12 = (s**(-1)*a*s) = rnf ;
      if m = k then return;
      for j := 1:n do
          <<temp := jm(m,j);
            jm(m,j) := jm(k,j);
            jm(k,j) := temp>>;
      for i := 1:n do
          <<temp := jm(i,m);
            jm(i,m) := jm(i,k);
            jm(i,k) := temp>>;
      for j := 1:n do
          <<temp := sm11(m,j);
            sm11(m,j) := sm11(k,j);
            sm11(k,j) := temp>>;
      for i := 1:n do
          <<temp := sm12(i,m);
            sm12(i,m) := sm12(i,k);
            sm12(i,k) := temp>>;
   end;

procedure rcadd(k1,k2,s);
   begin scalar i;
      % jm,n,sm11,sm12 : global ;
      % n : dimension of matrix jm ;
      % sm11,sm12 : transformation matrices for rational normal form ;
      %             sm11*a*sm12 = (s**(-1)*a*s) = rnf ;
      for i := 1:n do
          jm(k1,i) := jm(k1,i)+s*jm(k2,i);
      for i := 1:n do
          jm(i,k2) := jm(i,k2)-s*jm(i,k1);
```

```
      for i := 1:n do
          sm11(k1,i) := sm11(k1,i)+s*sm11(k2,i);
      for i := 1:n do
          sm12(i,k2) := sm12(i,k2)-s*sm12(i,k1);
   end;

procedure mdivmult(k,p);
   begin;
      % jm,n,sm11,sm12 : global ;
      % n : dimension of matrix jm ;
      % sm11,sm12 : transformation matrices for rational normal form ;
      %             sm11*a*sm12 = (s**(-1)*a*s) = rnf ;
      % mscl,imscl : global(local) ;
      % mscl : number for conversion of rational matrix into integer matrix ;
      % imscl : index of mscl ;
      imscl := imscl+1;
      mscl(imscl) := p;
      if p neq 1 then <<if k neq n then <<jm := jm*p;
                                          sm11 := sm11*p>>;
                        divmult(k,p)>>;
      return jm(k,k-1);
   end;

procedure rdivmult(k2);
   begin scalar i,mms,mms2;
      % jm,m,n,p,sm11,sm12 : global ;
      % n : dimension of matrix jm ;
      % sm11,sm12 : transformation matrices for rational normal form ;
      %             sm11*a*sm12 = (s**(-1)*a*s) = rnf ;
      % mscl,imscl : global(local) ;
      % mscl : number for conversion of rational matrix into integer matrix ;
      if imscl < 2
         then return
         else <<mms := mscl(imscl-1);
                divmult(k2,1/mms);
                jm := jm/mms;
                sm11 := sm11/mms>>;
   end;

procedure rdivmult2();
   begin scalar i,mms,mms2;
      % jm,m,n,p,sm11,sm12 : global ;
      % n : dimension of matrix jm ;
      % sm11,sm12 : transformation matrices for rational normal form ;
      %             sm11*a*sm12 = (s**(-1)*a*s) = rnf ;
      % mscl,imscl : global(local) ;
      % mscl : number for conversion of rational matrix into integer matrix ;
      if imscl < 1
         then retun
         else <<jm := jm/mscl(imscl);
                sm11 := sm11/mscl(imscl);
                clear mscl,imscl;
                array mscl(100);
                imscl := 0>>;
   end;

procedure divmult(k,p);
   begin scalar i;
      % jm,n,sm11,sm12 : global ;
      % n : dimension of matrix jm ;
      % sm11,sm12 : transformation matrices for rational normal form ;
      %             sm11*a*sm12 = (s**(-1)*a*s) = rnf ;
      for i := 1:n do
          jm(k,i) := jm(k,i)/p;
      for i := 1:n do
          jm(i,k) := jm(i,k)*p;
      for i := 1:n do
          sm11(k,i) := sm11(k,i)/p;
      for i := 1:n do
          sm12(i,k) := sm12(i,k)*p;
   end;

procedure divmult2(ma,k,p,l);
   begin scalar i;
      clear ddmm;
      matrix ddmm;
      ddmm := ma;
      for i := 1:l do
          ddmm(k,i) := ddmm(k,i)/p;
```

```
      for i := 1:l do
          ddmm(i,k) := ddmm(i,k)*p;
      return ddmm;
   end;

procedure cycle(k1);
   begin scalar k,k2,f,q,a1,a2,mk;
      % jm,n : global ;
      % n : dimension of matrix jm ;
      % mscl,imscl : global(local) ;
      % mscl : number for conversion of rational matrix into integer matrix ;
      f := 0;
      k := k1;
      while (f = 0) and (k < n) do
         <<q := k+1;
          while (q <= n) and (jm(q,k) = 0) do
             q := q+1;
          if q > n
             then f := 1
             else <<k2 := k+1;
                    swapmat(k2,q);
                    a1 := jm(k2,k);
                    if ratflg = 0
                       then a1 := mdivmult(k2,a1)
                       else <<divmult(k2,a1);
                              a1 := 1>>;
                    for i := 1:n do
                        if i neq k2 then
                           <<a2 := jm(i,k);
                             rcadd(i,k2,-a2/a1)>>;
                    rdivmult(k2);
                    k := k+1>>>>;
      mk := k-k1+1;
      if (mk = 1) and (ratflg = 0) then rdivmult2();
      if k = n then<<clear mscl,imscl;
                    array mscl(100);
                    imscl := 0>>;
      return mk;
   end;

procedure chkzero2(s,sp,ep);
   begin scalar f;                          % jm : global ;
      f := 0;
      while (sp <= ep) and (f = 0) do
         <<if jm(sp,s) neq 0 then f := 1;
           sp := sp+1>>;
      return f;
   end;

procedure chkzero(p,s);
   begin scalar i,s1,s2,q;                  % m : global ;
      i := 1;                               %   : array of dimensions of ;
      q := 0;                               %     companion matrices of ;
      s1 := 0;                              %     rational normal form ;
      while (i <= p) and (q = 0) do
         <<s2 := s1+1;
           s1 := s1+m(i);
           if chkzero2(s,s2,s1) neq 0 then q := i;
           i := i+1>>;
      return q;
   end;

procedure mksmat(p1,p2,ma);
   begin scalar i,j,k;                      % jm : global ;
      % mscl,imscl : global(local) ;
      % mscl : number for conversion of rational matrix into integer matrix ;
      clear pm1,pm2;
      pm1 := ma;
      k := p2-p1;
      if k <= 0 then rederr "matrix's degree is zero or negative";
      matrix pm2(k,k);
      for i := 1:k do
          for j := 1:k do
              pm2(i,j) := pm1(p1+i,p1+j);
      clear pm1;
      if imscl >= 2 then pm2 := pm2/mscl(imscl);
      return pm2;
```

```
   end;

procedure compmp(ma);
   begin scalar i,f,nn;                          % var0 : global ;
      clear fm;                                  %      : variable of ;
      nn := first length ma;                     %        polynomials ;
      fm := ma;
      f := var0**nn;
      for i := 1:nn do f := f-fm(i,nn)*var0**(i-1);
      clear fm;
      return f;
   end;

procedure remainmp(ma1,ma2,p);
   begin scalar mp1,mp2;
      mp1 := num(compmp(ma1));      % companion matrix's ;
      mp2 := num(compmp(ma2));      % minimal polynomial = ;
      if remainder(mp1,mp2) neq 0   % its characteristic polynomial ;
         then return 1
         else return 0;
   end;

procedure ratnf();
   begin scalar i,j,k,sp,sq,spp,sqm,r,a,q,sqj,sqjk;
      % jm,m,n,p : global ;
      % jm : rational normal form ;
      % m : array of dimensions of companion matrices of ;
      %     rational normal form ;
      % n : dimension of matrix jm ;
      % p : number of companion matrices ;
      % mscl,imscl,ratflg : global(local) ;
      % mscl : number for conversion of rational matrix into integer matrix ;
      % imscl : index of mscl ;
      % ratflg : flag for fraction free method(0) or normal method(1) ;
      clear mscl,imscl,ratflg;
      array mscl(100);
      imscl := 0;
      ratflg := 0;
      on ezgcd;
st1:  k := 1;
      p := 1;
      m(p) := cycle(k);
st2:  sp := (for i := 1:p sum m(i));
      if sp = n
         then return;
      m(p+1) := cycle(sp+1);
      q := chkzero(p,sp+m(p+1));
      if q = 0 then goto st4;
st3:  sqm := (for i := 1:q-1 sum m(i));          % sqm = m(1)+...+m(q-1)
      if m(p+1) >= m(q)
         then <<if ratflg = 0 then rdivmult2();
                swapmat(sqm+1,sp+1);
                p := q-1;
                if p = 0
                   then <<ratflg := 1;
                          goto st1>>
                   else goto st2>>
         else <<r := m(q)-m(p+1);
                sq := (for i := 1:q sum m(i)); % sq = m(1)+...+m(q)
                spp := sp+m(p+1);              % spp = m(1)+...+m(p+1)
                for j := 1:r do
                    <<sqj := sq-j+1;
                      for k := 1:m(p+1) do
                          <<sqjk := sqj-k;
                            rcadd(sqjk,spp+1-k,
                                  jm(sqjk+1,spp+1-k)/jm(sqjk+1,sqjk))>>>>>>;
      if chkzero2(spp,sqm+1,sq-r) = 0
         then <<i := chkzero(p,spp);
                if i > q then <<q := i;
                                goto st3>>
                          else goto st4>>
         else <<if ratflg = 0 then rdivmult2();
                swapmat(sqm+1,sp+1);
                p := q-1;
                if p = 0
```

```
                    then <<ratflg := 1;
                           goto st1>>
                    else goto st2>>;
 st4:  wm1 := mksmat(sp-m(p),sp,jm);
       wm2 := mksmat(sp,sp+m(p+1),jm);
       if remainmp(wm1,wm2,p) neq 0
          then <<rcadd(sp+1,sp-m(p)+1,-1);
                 p := p-1;
                 if p = 0
                    then goto st1>>
          else p := p+1;
       goto st2;
    end;

 procedure mkeldiv();
    begin scalar i,j,k,l;
       % elst,jm,m,p : global ;
       % elst : list of elementary divisors ;
       % jm : rational normal form ;
       % m : array of dimensions of companion matrices ;
       %     of rational normal form ;
       % p : number of companion matrices ;
       elst := {};
       l := 0;
       for k := 1:p do
           <<clear uc;
             matrix uc(m(k),m(k));
             for i := 1:m(k) do
                 for j := 1:m(k) do
                     uc(i,j) := jm(i+l,j+l);
             elst := append(elst,{compmp(uc)});
             l := l+m(k)>>;
       clear uc;
    end;

 procedure mkpos(lst1);
    % var0 : global ;
    %      : variable of polynomials ;
    for each p in lst1 collect
        if numberp(lcof(p,var0))
           then if lcof(p,var0) < 0 then p*(-1) else p
           else p;

 procedure lterm2(p,v);
    if deg(p,v) = 0 then p
                    else lterm(p,v);

 procedure lcof2(l,v);
    if deg(l,v) = 0 then l else lcof(l,v);

 procedure pord2(l1,l2);
    if numberp(l1)
       then if numberp(l2)
               then if l1 > l2 then 1
                                else 0
               else 0
       else if numberp(l2)
               then 1
               else if ordp(l1,l2) then 1 else 0;

 procedure pord(l1,l2);
    % var0 : global ;
    %        variable of polynomial ;
    if l1 = l2 then 1
       else if lterm2(l1,var0) = lterm2(l2,var0)
            then pord(reduct(l1,var0),reduct(l2,var0))
            else if deg(l1,var0) > deg(l2,var0)
                    then 1
                    else if deg(l1,var0) = deg(l2,var0)
                            then if pord2(lcof2(l1,var0),lcof2(l2,var0)) = 1
                                    then 1
                                    else 0
                            else 0;

 procedure psort(plst);
    begin scalar a,l1,l2,ll;
       if plst = {} then return {};
```

```
      l1 := {};
      l2 := {};
      a := first plst;
      ll := rest plst;
      while (ll neq {}) do
         <<if pord(a,first ll) = 1
              then l2 := first ll . l2
              else l1 := first ll . l1;
           ll := rest ll>>;
      return append(psort(l1),append({a},psort(l2)));
   end;

procedure chkmltp(plst);
   begin scalar l1,l2,k,nlst;
      if plst = {}
         then return {}
         else if rest plst = {} then return {{first plst,1}};
      nlst := {};
      l1 := first plst;
      l2 := rest plst;
      while (l2 neq {}) do
         <<k := 1;
           while (l2 neq {}) and (l1 = first l2) do
              <<k := k+1;
                l2 := rest l2>>;
           nlst := append(nlst,{{l1,k}});
           if l2 neq {}
              then <<if rest l2 neq {}
                        then l1 := first l2
                        else nlst := append(nlst,{{first l2,1}});
                     l2 := rest l2>>>>;
      return nlst;
   end;

procedure msort(lst);
   begin scalar i,n,l;
      clear s;
      n := length lst;
      array s(n);
      for i := 1:n do
          s(i) := part(lst,i);
      lst := {};
      while (s(1) neq {}) do
         <<l := first s(1);
           s(1) := rest s(1);
           lst := append(lst,{l});
           i := 2;
           while (s(n) = {}) do n := n-1;
           while (i <= n) and (first l = first first s(i)) do
              <<lst := append(lst,{first s(i)});
                s(i) := rest s(i);
                i := i+1>>>>;
      clear s;
      return lst;
   end;

procedure mkjclst();
   begin scalar d,i,jlst1,jlst2,lst,nn,ms,r,vlst,pl,opl;
      % p,elst,jclst,var0,z,pvlst,fclst : global ;
      % p : number of companion matrices ;
      % elst : list of elementary divisors ;
      % jclst : list of jordan cells ;
      % var0 : variable of polynomials ;
      % z : variable used to make names of eigen-values ;
      % pvlst : eigen-values and their definition polynomials ;
      % fclst : list of factorized elementary divisors ;
      jclst := {};
      fclst := {};
      for i := 1:p do
         <<pl := num(part(elst,i));
           fclst := append(fclst,
                    {chkmltp(psort(mkpos(factorize(pl))))});
           jclst := append(jclst,{part(fclst,i)})>>;
      jclst := msort(jclst);
   end;

procedure mkcomp(lst,s);
```

```
    begin scalar i,j,d,f,l,clst;                    % jm,var0 : global ;
       f := (first lst)**(second lst);              % jm : second natural ;
       d := deg(f,var0);                            %      normal form ;
       l := lcof(f,var0);                           % var0 : variable of ;
       clear cc;                                    %        polynomials ;
       matrix cc(d,d);
       clst := coeff(f,var0);
       for i := 1:d do
           for j := 1:d do
               cc(i,j) := 0;
       for i := 1:d-1 do
           cc(i+1,i) := 1;
       for i := 1:d do
           cc(i,d) := -part(clst,i)/l;
       for i := 1:d do
           for j := 1:d do
               jm(i+s,j+s) := cc(i,j);
       clear cc;
    end;

 procedure mk2nnf(jlst);
    begin scalar s;                                 % jm,n,var0 : global ;
       for i := 1:n do                              % jm : second natural ;
           for j := 1:n do                          %      normal form ;
               jm(i,j) := 0;                        % n : dimension of ;
       s := 0;                                      %     matrix jm;
       while (jlst neq {}) do                       % var0 : variable of ;
          <<lst := first jlst;                      %        polynomials ;
            jlst := rest jlst;
            mkcomp(lst,s);
            s := s+deg(first lst,var0)*second(lst)>>;
    end;

 procedure mkscomp(lst,s);
    begin scalar i,j,d,l,f,ms,ad,ss,clst;           % jm,var0 : global ;
       f := first lst;                              % jm : jacobson normal ;
       ms := second lst;                            %      form ;
       d := deg(f,var0);                            % var0 : variable of ;
       l := lcof(f,var0);                           %        polynomials ;
       ad := d*ms;
       clear sc;
       matrix sc(ad,ad);
       clst := coeff(f,var0);
       for i := 1:ad do
           for j := 1:ad do
               sc(i,j) := 0;
       ss := 0;
       for j := 1:ms do
           <<for i := 1:d-1 do
                 sc(i+ss+1,i+ss) := 1;
             ss := ss+d>>;
       ss := 0;
       for j := 1:ms do
           <<for i := 1:d do
                 sc(i+ss,d+ss) := -part(clst,i)/l;
             ss := ss+d>>;
       for i := 1:ms-1 do
           sc(1+d*(i-1),d*(i+1)) := 1;
       for i := 1:ad do
           for j := 1:ad do
               jm(i+s,j+s) := sc(i,j);
       clear sc;
    end;

procedure mkjacobson(jlst);
    begin scalar s;
       for i := 1:n do                              % jm,n,var0 : global
           for j := 1:n do                          % jm : jacobson normal ;
               jm(i,j) := 0;                        %      form ;
       s := 0;                                      % n : dimension of ;
       while (jlst neq {}) do                       %     matrix jm ;
          <<lst := first jlst;                      % var0 : variable of ;
            jlst := rest jlst;                      %        polynomials ;
            mkscomp(lst,s);
            s := s+deg(first lst,var0)*second(lst)>>;
    end;

procedure mkjc(r,ms,s);
```

```
    begin scalar i,j;
       clear jc;                                  % jm : global ;
       matrix jc(ms,ms);                          %    : jordan normal ;
       jc := r*jc**0;                             %      form ;
       for i := 1:ms-1 do
           jc(i,i+1) := 1;
       for i := 1:ms do
           for j := 1:ms do
               jm(i+s,j+s) := jc(i,j);
       clear jc;
    end;

 procedure mkjordan(jlst);
    begin scalar d,i,j,lst,nn,ms,s,opl,r,vlst,pl;
       % jm,n,var0,z,pvlst : global ;
       % jm : jordan normal form ;
       % n : dimension of matrix jm;
       % var0 : variable of polynomials ;
       % z : variable used to make names of eigen-values ;
       % pvlst : eigen-values and their definition polynomials ;
       for i := 1:n do
           for j := 1:n do
               jm(i,j) := 0;
       s := 0;
       nn := 1;
       opl := 0;
       vlst := {};
       pvlst  := {};
       while (jlst neq {}) do
        <<lst := first jlst;
          jlst := rest jlst;
          pl := first lst;
          ms := second lst;
          d := deg(pl,var0);
          if d = 1
             then <<r := var0-pl/lcof(pl,var0);
                    mkjc(r,ms,s);
                    s := s+ms;
                    if opl neq pl then
                       pvlst  := append(pvlst,{{pl,{r}}})>>
             else <<if opl = pl
                       then nn := nn-1;
                    for i := 1:d do
                        <<r := mkid(z,nn*10+i);
                          vlst := append(vlst,{r});
                          mkjc(r,ms,s);
                          s := s+ms>>;
                    if opl neq pl
                       then
                       <<write vlst," are the roots of ",pl;
                         pvlst := append(pvlst,{{pl,vlst}})>>;
                    vlst := {};
                    nn := nn+1>>;
         opl := pl>>;
    end;

 procedure evalue(plst);
    begin scalar gl,k,p,v,vlst,lst1,lst2;
       % pvlst : global ;
       %       : eigen-values and their definition polynomials ;
       lst1 := {};
       lst2 := {};
       for each p in plst do
           <<k := 1;
             while (first(p) neq first(part(pvlst,k))) do
                k := k+1;
             vlst := part(pvlst,k);
             lst2 := append(lst2,{first vlst});
             lst1 := append(lst1,{{second vlst,second p}})>>;
       return {lst2,lst1};
    end;

 procedure fac(k);
    if k < 0 then rederr "given number is negative"
       else if k = 0 then 1
                     else (for i := 1:k product i);

 procedure makeumat(pl,lk,plst,elst);
```

```
    begin scalar i,j,k,l,ii,kk,ll,p,r,ev,plv,evl;
      % var0 : variable of polynomials ;
      % uc1 : global(local) ;
      %     : sub-matrix of transformation matrix sm21 ;
      clear uc1;
      l := deg(pl,var0);
      matrix uc1(l,l);
      k := 0;
      for kk := 1:lk do
          <<r := part(elst,kk);
            evl := first r;
            p := second r;
            ll := length evl;
            ev := first evl;
            plv := sub(var0=ev,part(plst,kk));
            for i := 1:l do
                for j := 1:p do
                    if ((p-j) > (i-1)) or ((p-j) < 0)
                       then uc1(j+k,i) := 0
                       else if (ev = 0) and ((i+j-p-1) = 0)
                            then uc1(j+k,i) :=
                                 (fac(i-1)/(fac((i-1)-(p-j))*fac(p-j)))
                            else
                               <<uc1(j+k,i) :=
                                 (fac(i-1)/(fac((i-1)-(p-j))*fac(p-j)))
                                 *ev**(i+j-p-1);
                                 if not numberp(plv) then
                            uc1(j+k,i) := preduce(uc1(j+k,i),{plv},{ev})>>;
            k := k+p;
            for ii := 2:ll do
                <<for i := 1:l do
                      for j := 1:p do
                         uc1(j+k,i) := sub(ev=part(evl,ii),
                                       uc1(j+(k-p*(ii-1)),i));
                 k := k+p>>>>;
    end;

procedure mkqmat(l,l2,pl,lk,lst1,lst2);
    begin scalar ll,i,j,k,kk,r,clst,evl,p,ev,plv;
      % var0 : global ;
      %      : variable of polynomials ;
      % qm : global(local) ;
      %    : factor of sub-matrix uc2 ;
      clear qar;
      clst := rest reverse coeff(pl,var0);
      matrix qar(l,l2);
      for i := 1:l do
          for j := 1:l2 do
              if i = j
                 then qar(i,j) := 1
                 else qar(i,j) := 0;
      for i := 2:l do
          qar(i,1) := qar(i-1,1)*var0+part(clst,i-1);
      for j :=2:l2 do
          for i := j+1:l do
              qar(i,j) := qar(i-1,j)*var0+qar(i-1,j-1);
      k := 0;
      for kk := 1:lk do
          <<r := part(lst2,kk);
            evl := first r;
            p := second r;
            ll := length evl;
            ev := first evl;
            plv := sub(var0=ev,part(lst1,kk));
            for i := 1:l do
                for j := 1:p do
                    <<qm(i,j+k) := sub(var0=ev,qar(l-i+1,p-j+1));
                      if not numberp(plv) then
                         qm(i,j+k) := preduce(qm(i,j+k),{plv},{ev})>>;

            k := k+p;
            for ii := 2:ll do
                <<for i := 1:l do
                      for j := 1:p do
                         qm(i,j+k) := sub(ev=part(evl,ii),
                                      qm(i,j+(k-p*(ii-1))));
                 k := k+p>>>>;
```

```
    %   clear qar;
    end;

procedure mkdmmat(l,l2,pl,lk,lst1,lst2);
   begin scalar ll,l22,i,j,k,kk,pl2,r,clst,evl,p,k2,ev;
      % var0 : global ;
      %      : variable of polynomials ;
      % dmm : global(local) ;
      %     : factor of sub-matrix uc2 ;
      clear dar;
      l22 := l2*2-1;
      array dar(l22);
      pl2 := pl;
      for i := 1:l22 do dar(i) := 0;
      for i := 1:l22 do
          <<pl2 := df(pl2,var0);
            dar(i) := pl2/fac(i)>>;
      for i := 1:l do
          for j := 1:l do
             dmm(i,j) := 0;
      k := 0;
      for kk := 1:lk do
          <<r := part(lst2,kk);
            evl := first r;
            p := second r;
            ll := length evl;
            ev := first evl;
            plv := sub(var0=ev,part(lst1,kk));
            clear dmd1,dmd2;
            array dmd1(p),dmd2(p);
            for i := 1:p do
                dmd1(i) := sub(var0=ev,dar(p-1+i));
            dmd2(1) := 1/dmd1(1);
            for i := 2:p do
                dmd2(i) := -dmd2(1)*(for j := 1:i-1 sum dmd1(i+1-j)*dmd2(j));
            for i := 1:p do
                <<k2 := 1;
                  for j := p-i+1:p do
                      <<dmm(i+k,j+k) := dmd2(k2);
                        if not numberp(plv) then
                           <<dmm(i+k,j+k) :=
                               preduce(num(dmm(i+k,j+k)),{plv},{ev})/
                               preduce(den(dmm(i+k,j+k)),{plv},{ev})>>;
                        k2 := k2+1>>>>;
            k := k+p;
            for ii := 2:ll do
                <<for i := 1:p do
                      for j := 1:p do
                         dmm(i+k,j+k) := sub(ev=part(evl,ii),
                                         dmm(i+(k-p*(ii-1)),j+(k-p*(ii-1))));
                  k := k+p>>>>;
      clear dar,dmd1,dmd2;
   end;

procedure makeuinv(pl,ll,lst1,lst2);
   begin scalar i,l2,ppl,ppvl,ppvll,k,kk,j,v,v2;
      % var0 : global ;
      %      : variable of polynomials ;
      % uc2,qm,dmm : global(local) ;
      % uc2 : sub-matrix of transformation matrix sm22 ;
      clear uc2,qm,dmm;
      l := deg(pl,var0);
      l2 := 1;
      for i := 1:ll do
          if second(part(lst2,i)) > l2 then l2 := second(part(lst2,i));
      matrix uc2(l,l),qm(l,l),dmm(l,l);
      mkqmat(l,l2,pl,ll,lst1,lst2);
      mkdmmat(l,l2,pl,ll,lst1,lst2);
      uc2 := qm*dmm;
      for k := 1:length lst1 do
          <<ppl := part(lst1,k);
            ppvl := first(part(lst2,k));
            ppvll := length ppvl;
            for i := 1:l do
                for j := 1:l do
                    <<v := uc2(i,j);
                      for kk := 1:ppvll do
```

```
                                <<v2 := sub(var0=part(ppvl,kk),ppl);
                                  if (v neq 0) and (not numberp(v2)) then
                                      v := preduce(num(v),{v2})
                                           /preduce(den(v),{v2})>>;
                          uc2(i,j) := v>>>>;
        clear qm,dmm;
    end;

procedure mktrnsmat();
    begin scalar lst,l,pl,lst1,lst2,ll;
        % m,n,p,var0,elst,z,pvlst,sm21,sm22 : global ;
        % m  : array of dimensions of companion matrices of ;
        %      rational normal form ;
        % n  : dimension of original matrix ;
        % p  : number of companion matrices ;
        % x  : variable of polynomials ;
        % fclst : list of factorized elementary divisors ;
        % elst : list of elementary divisors ;
        % z  : variable used to make names of eigen-values ;
        % pvlst : eigen-values and their definition polynomials ;
        % sm21,sm22 : transformation matrices for jordan normal form ;
        %             sm21*rnf*sm22 = (u**(-1)*r*u) = jrd ;
        % uc1 : global(local) ;
        %     : sub-matrix of transformation matrix sm21 ;
        % uc2 : global(local) ;
        %     : sub-matrix of transformation matrix sm22 ;
        matrix sm21(n,n),sm22(n,n);
        l := 0;
        for k := 1:p do
            <<pl := num(part(elst,k));
              lst := evalue(part(fclst,k));
              lst1 := first lst;
              lst2 := second lst;
              ll := length lst2;
              makeumat(pl,ll,lst1,lst2);
              makeuinv(pl,ll,lst1,lst2);
              for i := 1:m(k) do
                  for j := 1:m(k) do
                      <<sm21(i+l,j+l) := uc1(i,j);
                        sm22(i+l,j+l) := uc2(i,j)>>;
              l := l+m(k)>>;
        clear uc1,uc2;
    end;

procedure jordan(ma);
    begin scalar t0,t1,t2;
        t0 := lisp time();
        clear jm,m,n,p,var0,elst,jclst,fclst,z;
        clear rnf,snf,jcb,jrd,pvlst,sm11,sm12,sm21,sm22;
        % jm,m,n,p,var0,elst,jclst,fclst,z : global variables ;
        % rnf,snf,jcb,jrd,pvlst,sm11,sm12,sm21,sm22 : global variables ;
        % jm : original matrix -> rational normal form ;
        %      -> second natural normal form ;
        %      -> jacobson normal form -> jordan normal form ;
        % m  : array of dimensions of companion matrices of ;
        %      rational normal form ;
        % n  : dimension of original matrix ;
        % p  : number of companion matrices ;
        % var0 : variable of polynomials ;
        % elst : list of elementary divisors ;
        % jclst : list of jordan cells (simple elementary divisors and ;
        %         thier multiplicities) ;
        % fclst : list of factorized elementary divisors ;
        % z  : variable used to make names of eigen-values ;
        % rnf : rational normal form ;
        % snf : second natural normal form ;
        % jcb : jacobson normal form ;
        % jrd : jordan normal form ;
        % pvlst : eigen-values and their definition polynomials ;
        % sm11,sm12 : transformation matrices for rational normal form ;
        %             sm11*a*sm12 = (s**(-1)*a*s) = rnf ;
        % sm21,sm22 : transformation matrices for rational normal form ;
        %             to jordan normal form ;
        %             sm21*rnf*sm22 = (u**(-1)*r*u) = jrd ;
```

```
      matrix jm;
      array m(100);
      n := first length ma;
      if n neq second length ma
         then return <<write "not square matrix">>;
      jm := ma;
      m(0) := 0;
      sm11 := ma**0;
      sm12 := sm11;
      write "===== rational normal form =====";
      ratnf();
      write jm;
      rnf := jm;                    % rnf = rational normal form;
      write " ===== transformation matrices",
            " from the input matrix";
      write "       to its rational normal form ===== ";
      write sm11;
      write " ";
      write sm12;
      t1 := lisp time();
      write t1-t0," ms";
      write "===== elementary divisors =====";
      mkeldiv();
      write elst;
      write "===== simple elementary divisors =====";
      mkjclst();
      write jclst;
      t2 := lisp time();
      write t2-t1," ms";
      write "===== second natural normal form =====";
      mk2nnf(jclst);
      write jm;
      t1 := lisp time();
      write t1-t2," ms";
      snf := jm;                    % snf = 2nd natural normal form;
      write "===== jacobson normal form =====";
      mkjacobson(jclst);
      write jm;
      t2 := lisp time();
      write t2-t1," ms";
      jcb := jm;                    % jcb = jacobson normal form;
      write "===== jordan normal form =====";
      mkjordan(jclst);
      write jm;
      t1 := lisp time();
      write t1-t2," ms";
      jrd := jm;                    % jrd = jordan normal form;
      mktrnsmat();
      write " ===== transformation matrices",
            " from the rational normal form";
      write "       to jordan normal form ===== ";
      write sm21;
      write " ";
      write sm22;
      t2 := lisp time();
      write t2-t1," ms";
      write " ===== transformation matrices to jordan normal form ==== ";
      write sm21*sm11;
      write " ";
      write sm12*sm22;
      t1 := lisp time();
      write t1-t2," ms";
      write t1-t0," ms";
      clear jm,m,n,p,var0,elst,jclst,fclst,z;
%     clear rnf,snf,jcb,jrd,pvlst,sm11,sm12,sm21,sm22 ;
   end;

;end;
```

B.2 Jordan 標準形の応用のプログラム

ここでは、1 階の線形差分方程式と 1 階の定係数連立線形微分方程式の解の係数行列を計算するプログラムを示す（3.2節、3.5.1.2 項、3.5.2.2 項のアルゴリズムに基づくプログラム）。ただし、B.1 節と同じプロシージャは省略する。

```
procedure mktrnsmat()$
   begin scalar lst,l,pl,lst1,lst2,ll;
      % m,n,p,var0,elst,z,pvlst,evlst,sm21,sm22 : global ;
      % m  : array of dimensions of companion matrices of ;
      %      rational normal form ;
      % n  : dimension of original matrix ;
      % p  : number of companion matrices ;
      % x  : variable of polynomials ;
      % fclst : list of factorized elementary divisors ;
      % elst : list of elementary divisors ;
      % z  : variable used to make names of eigen-values ;
      % pvlst : eigen-values and their definition polynomials ;
      % evlst : list of eigen-values and multiplicities ;
      % sm21,sm22 : transformation matrices for jordan normal form ;
      %             sm21*rnf*sm22 = (u**(-1)*r*u) = jrd ;
      % uc1 : global(local) ;
      %     : sub-matrix of transformation matrix sm21 ;
      % uc2 : global(local) ;
      %     : sub-matrix of transformation matrix sm22 ;
      matrix sm21(n,n),sm22(n,n);
      l := 0;
      evlst := {};
      for k := 1:p do
          <<pl := num(part(elst,k));
            lst := evalue(part(fclst,k));
            evlst := append(evlst,{lst});
            lst1 := first lst;
            lst2 := second lst;
            ll := length lst2;
            makeumat(pl,ll,lst1,lst2);
            makeuinv(pl,ll,lst1,lst2);
            for i := 1:m(k) do
                for j := 1:m(k) do
                    <<sm21(i+l,j+l) := uc1(i,j);
                      sm22(i+l,j+l) := uc2(i,j)>>;
            l := l+m(k)>>;
      clear uc1,uc2;
   end$

procedure jordan(ma)$
   begin scalar t0,t1,t2;
      t0 := lisp time();
      clear jm,m,n,p,var0,elst,jclst,fclst,z,sm1,sm2;
      clear rnf,snf,jcb,jrd,pvlst,evlst,sm11,sm12,sm21,sm22;
      matrix jm;
      array m(100);
      n := first length ma;
      if n neq second length ma
         then return <<write "not square matrix">>;
      jm := ma;
      m(0) := 0;
      sm11 := ma**0;
      sm12 := sm11;
      write "===== rational normal form =====";
      ratnf();
      write jm;
      write " ===== transformation matrices",
            " from the input matrix";
      write "       to its rational normal form ===== ";
      t1 := lisp time();
      write t1-t0," ms";
      write "===== elementary divisors =====";
      mkeldiv();
      write elst;
      write "===== simple elementary divisors =====";
      mkjclst();
```

```
      write jclst;
      t2 := lisp time();
      write t2-t1," ms";
      write "===== jordan normal form =====";
      mkjordan(jclst);
      write jm;
      t1 := lisp time();
      write t1-t2," ms";
      mktrnsmat();
      write " ===== transformation matrices",
            " from the rational normal form";
      write "       to jordan normal form ===== ";
      t2 := lisp time();
      write t2-t1," ms";
      write " ===== transformation matrices to jordan normal form ==== ";
      sm1 := sm21*sm11;
      sm2 := sm12*sm22;
      t1 := lisp time();
      write t1-t2," ms";
      write t1-t0," ms";
     clear jm,n,var0,elst,jclst,fclst,z;
     clear pvlst,sm11,sm12,sm21,sm22 ;
   end$

%------------------------- application programs -------------------------;

procedure matrixpred(ma)$
   begin scalar i,j,k,l,ii,lst,pl,pll,evl,ll,v,v1,v2;
      % n,evlst : global variables ;
      % n  : dimension of original matrix ;
      % evlst : list of eigen-values and multiplicities ;
      clear mmmm;
      mmmm := ma;
      l := length evlst;
      for k := 1:l do
          <<pl := first(part(evlst,k));
            lst := first(second(part(evlst,k)));
            evl := first lst;
            ll := length evl;
            for i := 1:n do
                for j := 1:n do
                    <<v := mmmm(i,j);
                      for ii := 1:ll do
                          <<pll := sub(var0=part(evl,ii),pl);
                            if (v neq 0) and (not numberp(first pll)) then
                               <<v1 := preduce(num(v),pll);
                                 v2 := preduce(num(v),pll);
                                 if v2 neq 0 then v := v1/v2>>;
                            mmmm(i,j) := v>>>>>>;
      return mmmm;
   end$

%---- coefficient matrix of solution of system of                          ;
%                 differential equations of first order --------------------;

procedure mkpred(vlst,vlst2,v)$
   begin scalar i,l;
      l := length vlst;
      for i := 1:l do
          v := sub(part(vlst,i)=part(vlst2,i),v);
      return v;
   end$

procedure slvdif()$
   begin scalar l,i,j,k,ml,ev1,ev2,sz,lst,vlst,vlst2,ll,evl,
                ii,v1,v2,p,t0,t1,pl;
      % n,evlst,sm1,sm2 : global ;
      % n : dimension of original matrix ;
      % evlst : list of eigen-values and multiplicities ;
      % sm1,sm2 : transformation matrices for jordan normal form ;
      %           sm1 = sm21*sm11,sm2 = sm12*sm22 ;
      %           sm1*a*sm2 = (t**(-1)*a*t) = jrd ;
      t0 := lisp time();
      clear cdm,o;
      matrix cdm(n,n);
      l := length evlst;
      for i := 1:n do
          for j := 1:n do
```

```
            cdm(i,j) := 0;
      sz := 0;
      vlst := {};
      vlst2 := {};
      for k := 1:l do
          <<lst := first(second(part(evlst,k)));
            evl := first lst;
            ml := second lst;
            ll := length evl;
            ev1 := first evl;
            clear km;
            matrix km(ml,ml);
            for i := 2:ml do
                for j := 1:i-1 do
                    km(i,j) := 0;
            v1 := mkid(o,k*10+1);
            vlst := append(vlst,{v1});
            vlst2 := append(vlst2,{exp(ev1*tt)});
            for i := 1:ml do
                for j := i:ml do
                    <<p := j-i;
                      km(i,j) := (tt**p*v1)/fac(p)>>;
            for i := 1:ml do
                for j := 1:ml do
                    cdm(i+sz,j+sz) := km(i,j);
            sz := sz+ml;
            for ii := 2:ll do
                <<ev2 := part(evl,ii);
                  v2 := mkid(o,k*10+ii);
                  vlst := append(vlst,{v2});
                  vlst2 := append(vlst2,{exp(ev2*tt)});
                  for i := 1:ml do
                      for j := 1:ml do
                          cdm(i+sz,j+sz) := sub(ev1=ev2,v1=v2,km(i,j));
                  sz := sz+ml>>>>;
      cdm := matrixpred(sm2*cdm*sm1);
      for i := 1:n do
          for j := 1:n do
              cdm(i,j) := mkpred(vlst,vlst2,cdm(i,j));
      clear km;
      t1 := lisp time();
      write "===== coefficient matrix of solution of system of ";
      write "      differential equations of first order =====";
      write t1-t0," ms";
      write "'tt' is the variable of the equations ";
      return cdm;
   end$

%--------coefficient matrix of solution of system of                    ;
%                   difference equations of first order -----------------;

procedure matpowtm(t0,t1)$
   begin;
      write "===== coefficient matrix of solution of system of ";
      write "      difference equations of first order =====";
      write t1-t0," ms";
   end$

procedure matpow(m1)$
   begin scalar i,j,el,sz,k1,k2,l,ll,vlst,ij,p,v1,v2,q,pll,lst,
               vv1,vv2,vlst2,vlst3,t0,t1;
      % n,evlst,sm1,sm2 : global ;
      % n : dimension of original matrix ;
      % evlst : list of eigen-values and multiplicities ;
      % sm1,sm2 : transformation matrices for jordan normal form ;
      %           sm1 = sm21*sm11,sm2 = sm12*sm22 ;
      %           sm1*a*sm2 = (t**(-1)*a*t) = jrd ;
      t0 := lisp time();
      if numberp(m1) and (m1 < 0)
         then rederr "power is negative";
      clear pm,w,mm;
      mm := m1;
      matrix pm(n,n);
      for i := 1:n do
          for j := 1:n do
              pm(i,j) := 0;
      el := length evlst;
      sz := 0;
```

```
    vlst2 := {};
    vlst3 := {};
    for k1 := 1:el do
        <<lst := first(second(part(evlst,k1)));
          vlst := first lst;
          ll := length vlst;
          p := second lst;
          v1 := part(vlst,1);
          if numberp(mm)
             then <<if mm < p
                      then q := mm
                      else q := p-1;
                    vv1 := v1**mm>>
             else <<q := p-1;
                    vv1 := mkid(w,k1*10+1);
                    vlst2 := append(vlst2,{v1**mm});
                    vlst3 := append(vlst3,{vv1})>>;
          for i := 1:p do
              for j := 1:p do
                  <<ij := j-i;
                    if (ij >= 0) and (ij <= q)
                       then pm(i+sz,j+sz) :=
                            ((for l := 0:ij-1 product mm-l)/fac(ij))*
                            v1**(-ij)*vv1>>;
          sz := sz+p;
          for k2 := 2:ll do
              <<v2 := part(vlst,k2);
                if numberp(mm)
                   then vv1 := w;
                vv2 := mkid(w,k1*10+k2);
                vlst2 := append(vlst2,{v2**mm});
                vlst3 := append(vlst3,{vv2});
                for i := 1:p do
                    for j := 1:p do
                        pm(i+sz,j+sz) := sub(v1=v2,vv1=vv2,
                                             pm(i+sz-(p*(k2-1)),
                                             j+sz-(p*(k2-1))));
                sz := sz+p>>>>;
    pm := matrixpred(sm2*pm*sm1);
    t1 := lisp time();
    if not numberp(mm) then
       <<ll := length vlst2;
         for k1 := 1:ll do
             <<v1 := part(vlst3,k1);
               v2 := part(vlst2,k1);
               for i := 1:n do
                   for j := 1:n do
                       pm(i,j) := sub(v1=v2,pm(i,j))>>>>;
    matpowtm(t0,t1);
    return pm;
end$
```

付録 C

データと計算結果

参考のため、計算に用いたデータと計算結果の一部を示す。

C.1 テストデータ一覧

3.4 節で用いたデータのうち、データ 3.3,3.4 の 3 次から 5 次までの 3 つの入力行列を示す。

[データ 3.3] 有理標準形がコンパニオン行列 3 個からなる整数行列 ($R = C_1 \oplus C_2 \oplus C_3$)

3 次

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

4 次

$$\begin{bmatrix} 27 & -35 & 7 & 0 \\ 24 & -31 & 6 & -0 \\ 8 & -10 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

5 次

$$\begin{bmatrix} 224 & -279 & 69 & -20 & 5 \\ 220 & -274 & 68 & -20 & 5 \\ 195 & -243 & 62 & -20 & 5 \\ 144 & -180 & 48 & -17 & 4 \\ 36 & -45 & 12 & -4 & 0 \end{bmatrix}$$

[データ 3.4] 有理標準形がコンパニオン行列 3 個からなる 1 変数整数係数多項式行列
($R = C_1 \oplus C_2 \oplus C_3$)

3 次

$$\begin{bmatrix} -x & 0 & 0 \\ 0 & -x & 0 \\ 0 & 0 & -x \end{bmatrix}$$

4 次

$$\begin{bmatrix} 12x^2+12x+3 & -16x^2-16x-3 & 4x^2+3x & 0 \\ 9x^2+12x+3 & -12x^2-16x-3 & 3x^2+3x & 0 \\ 6x+2 & -8x-2 & x & 0 \\ 0 & 0 & 0 & -x \end{bmatrix}$$

5 次

$$\begin{bmatrix} 92x^2+104x+28 & -115x^2-130x-34 & 29x^2+32x+8 & -9x^2-9x-2 & 3x^2+2x \\ 88x^2+104x+28 & -110x^2-130x-34 & 28x^2+32x+8 & -9x^2-9x-2 & 3x^2+2x \\ 72x^2+96x+27 & -90x^2-120x-33 & 24x^2+30x+8 & -9x^2-9x-2 & 3x^2+2x \\ 48x^2+72x+24 & -60x^2-90x-30 & 16x^2+24x+8 & -6x^2-9x-2 & 2x^2+2x \\ 24x+12 & -30x-15 & 8x+4 & -3x-1 & 0 \end{bmatrix}$$

C.2 REDUCE3.5 による計算結果

SUN SPARC server10 上の REDUCE3.5 を用いて、**C.1** に示したデータ 3.3,3.4 の 3 次から 5 次までの 3 つの行列について計算を行なった結果を示す。有理標準形の計算する方法としては分数なしアルゴリズムを用い、プログラムは **B.1, B.2** に示したものを使用した。

C.2.1 標準形の計算結果

REDUCE3.5 による出力結果をそのまま示す。

データ 3.4 の変換行列の計算結果については、長くなるので省略した。以下の計算例では、便宜上、固有値を表す変数 λ を var0 、記号で表された固有値 λ_{ij} ををzij、変数 x を y として表している。

[データ 3.3]

3 次
```
clear a3$
matrix a3$
%-----------------------------------------------------------------$
a3:=
mat((-1,0,0),(0,-1,0),(0,0,-1))$
jordan(a3)$
===== rational normal form =====
[-1  0   0 ]
[          ]
[0   -1  0 ]
```

```
[
[0   0   -1]
 ===== transformation matrices from the input matrix
       to its rational normal form =====
[1  0  0]
[
[0  1  0]
[
[0  0  1]
[1  0  0]
[
[0  1  0]
[
[0  0  1]
280 ms
===== elementary divisors =====
{var0 + 1,var0 + 1,var0 + 1}
===== simple elementary divisors =====
{{var0 + 1,1},{var0 + 1,1},{var0 + 1,1}}
239 ms
===== second natural normal form =====
[-1  0   0 ]
[
[0   -1  0 ]
[
[0   0   -1]
150 ms
===== jacobson normal form =====
[-1  0   0 ]
[
[0   -1  0 ]
[
[0   0   -1]
151 ms
===== jordan normal form =====
[-1  0   0 ]
[
[0   -1  0 ]
[
[0   0   -1]
130 ms
 ===== transformation matrices from the rational normal form
       to jordan normal form =====
[1  0  0]
[
[0  1  0]
[
[0  0  1]
[1  0  0]
[
[0  1  0]
[
[0  0  1]
540 ms
 ===== transformation matrices to jordan normal form ====
[1  0  0]
[
[0  1  0]
[
[0  0  1]
[1  0  0]
```

```
[0  1  0]
[0  0  1]
10 ms
1500 ms
%
```

4次

```
clear a4$
matrix a4$
%-----------------------------------------------------------------$
a4:=
mat((27,-35,7,0),(24,-31,6,0),(8,-10,1,0),(0,0,0,-1))$
jordan(a4)$
===== rational normal form =====
[0  -1  0   0 ]
[1  -2  0   0 ]
[0  0   -1  0 ]
[0  0   0   -1]
 ===== transformation matrices from the input matrix
       to its rational normal form =====
[      - 29    1     ]
[1  -------   ---  0]
[      24      4     ]
[                    ]
[      1             ]
[0    ----     0   0]
[      24            ]
[                    ]
[      - 1           ]
[0   ------     1   0]
[      3             ]
[                    ]
[0      0       0   1]
[            - 1     ]
[1   27   ------  0]
[            4       ]
[                    ]
[0   24     0      0]
[0   8      1      0]
[0   0      0      1]
689 ms
===== elementary divisors =====
     2
{var0  + 2*var0 + 1,var0 + 1,var0 + 1}
===== simple elementary divisors =====
{{var0 + 1,2},{var0 + 1,1},{var0 + 1,1}}
310 ms
===== second natural normal form =====
[0  -1  0   0 ]
[1  -2  0   0 ]
[0  0   -1  0 ]
[0  0   0   -1]
190 ms
===== jacobson normal form =====
[-1  1   0   0 ]
```

```
[0   -1  0   0 ]
[0   0   -1  0 ]
[0   0   0   -1]
200 ms
===== jordan normal form =====
[-1  1   0   0 ]
[0   -1  0   0 ]
[0   0   -1  0 ]
[0   0   0   -1]
170 ms
 ===== transformation matrices from the rational normal form
       to jordan normal form =====
[0  1   0  0]
[1  -1  0  0]
[0  0   1  0]
[0  0   0  1]
[1  1  0  0]
[1  0  0  0]
[0  0  1  0]
[0  0  0  1]
730 ms
 ===== transformation matrices to jordan normal form ====
[      1                 ]
[0   ----     0    0]
[      24                ]
[                        ]
[      - 5     1         ]
[1   ------   ---   0]
[      4       4         ]
[                        ]
[      - 1               ]
[0   ------    1    0]
[      3                 ]
[0     0      0    1]
[                 - 1    ]
[28  1   ------   0]
[                 4      ]
[24  0     0      0]
[8   0     1      0]
[0   0     0      1]
61 ms
2350 ms
%
```

5次

```
clear a5$
matrix a5$
%-----------------------------------------------------------------$
a5:=
mat((224,-279,69,-20,5),(220,-274,68,-20,5),(195,-243,62,-20,5),
(144,-180,48,-17,4),(36,-45,12,-4,0))$
jordan(a5)$
===== rational normal form =====
```

```
[0  -1  0  0   0 ]
[1  -2  0  0   0 ]
[0  0   0  -1  0 ]
[0  0   1  -2  0 ]
[0  0   0  0   -1]
```

```
===== transformation matrices from the input matrix
      to its rational normal form =====
```

```
[      - 17          5        - 1 ]
[1  -------   0   ----    ------]
[      16           64       24  ]
[                                ]
[      1            - 17         ]
[0    ----     0  -------    0   ]
[      16           192          ]
[                                ]
[      - 9          - 35     5   ]
[0   ------   1  -------   ----  ]
[      16           64       24  ]
[                                ]
[      - 3          55           ]
[0   ------   0   -----    0     ]
[      16           192          ]
[                                ]
[                   - 1          ]
[0     0      0  ------    1     ]
[                   4            ]
```

```
[                  1    ]
[1  224  0  69    ----  ]
[                  24   ]
[                       ]
[0  220  0  68    0     ]
[                       ]
[                  - 5  ]
[0  195  1  62   ------]
[                  24   ]
[                       ]
[0  144  0  48    0     ]
[0  36   0  12    1     ]
```

1541 ms

```
===== elementary divisors =====
```

$\{var0^2 + 2*var0 + 1, var0^2 + 2*var0 + 1, var0 + 1\}$

```
===== simple elementary divisors =====
```

{{var0 + 1,2},{var0 + 1,2},{var0 + 1,1}}

390 ms

```
===== second natural normal form =====
```

```
[0  -1  0  0   0 ]
[1  -2  0  0   0 ]
[0  0   0  -1  0 ]
[0  0   1  -2  0 ]
[0  0   0  0   -1]
```

250 ms

```
===== jacobson normal form =====
```

```
[-1  1   0   0   0 ]
[0   -1  0   0   0 ]
[0   0   -1  1   0 ]
[0   0   0   -1  0 ]
[0   0   0   0   -1]
```

```
260 ms
===== jordan normal form =====
[-1  1   0   0   0 ]
[0  -1   0   0   0 ]
[0   0  -1   1   0 ]
[0   0   0  -1   0 ]
[0   0   0   0  -1]
180 ms
 ===== transformation matrices from the rational normal form
       to jordan normal form =====
[0  1   0  0   0]
[1  -1  0  0   0]
[0  0   0  1   0]
[0  0   1  -1  0]
[0  0   0  0   1]
[1  1  0  0  0]
[1  0  0  0  0]
[0  0  1  1  0]
[0  0  1  0  0]
[0  0  0  0  1]
920 ms
 ===== transformation matrices to jordan normal form ====
[      1            - 17          ]
[0   ----    0   -------    0    ]
[     16            192           ]
[                                 ]
[     - 9            1       - 1 ]
[1  ------   0    ---     ------ ]
[      8             6       24  ]
[                                 ]
[     - 3           55           ]
[0  ------   0   -----     0     ]
[      16           192           ]
[                                 ]
[     - 3          - 5      5    ]
[0  ------   1   ------    ----   ]
[      8            6       24    ]
[                                 ]
[                  - 1            ]
[0     0     0   ------     1     ]
[                   4             ]
[                                 ]
[                    1            ]
[225  1  69  0    ----            ]
[                   24            ]
[                                 ]
[220  0  68  0     0              ]
[                                 ]
[                   - 5           ]
[195  0  63  1   ------           ]
[                   24            ]
[                                 ]
[144  0  48  0     0              ]
[                                 ]
[36   0  12  0     1              ]
140 ms
3681 ms
%
```

[データ 3.4]

3次 `clear a3$`

```
matrix a3$
%-----------------------------------------------------------------$
a3:=
mat(( - y,0,0),(0, - y,0),(0,0, - y))$
jordan(a3)$
===== rational normal form =====
[ - y   0     0  ]
[                ]
[ 0     - y   0  ]
[                ]
[ 0     0     - y]
 ===== transformation matrices from the input matrix
       to its rational normal form =====
290 ms
===== elementary divisors =====
{var0 + y,var0 + y,var0 + y}
===== simple elementary divisors =====
{{var0 + y,1},
 {var0 + y,1},
 {var0 + y,1}}
270 ms
===== second natural normal form =====
[ - y   0     0  ]
[                ]
[ 0     - y   0  ]
[                ]
[ 0     0     - y]
160 ms
===== jacobson normal form =====
[ - y   0     0  ]
[                ]
[ 0     - y   0  ]
[                ]
[ 0     0     - y]
180 ms
===== jordan normal form =====
[ - y   0     0  ]
[                ]
[ 0     - y   0  ]
[                ]
[ 0     0     - y]
140 ms
 ===== transformation matrices from the rational normal form
       to jordan normal form =====
630 ms
 ===== transformation matrices to jordan normal form ====
10 ms
1680 ms
%
```

4次

```
clear a4$
matrix a4$
```

```
%-----------------------------------------------------------------$
a4:=
mat((3*(4*y**2 + 4*y + 1), - 16*y**2 - 16*y - 3,y*(4*y + 3),0),
(3*(3*y**2 + 4*y + 1), - 12*y**2 - 16*y - 3,3*y*(y + 1),0),
(2*(3*y + 1),2*( - 4*y - 1),y,0),(0,0,0, - y))$
jordan(a4)$
===== rational normal form =====
[        2                ]
[0   - y     0      0    ]
[                         ]
[1   - 2*y   0      0    ]
[                         ]
[0    0      - y    0    ]
[                         ]
[0    0      0      - y  ]
 ===== transformation matrices from the input matrix
       to its rational normal form =====
1130 ms
===== elementary divisors =====
{var0^2 + 2*var0*y + y^2,
 var0 + y,
 var0 + y}
===== simple elementary divisors =====
{{var0 + y,2},
 {var0 + y,1},
 {var0 + y,1}}
380 ms
===== second natural normal form =====
[        2                ]
[0   - y     0      0    ]
[                         ]
[1   - 2*y   0      0    ]
[                         ]
[0    0      - y    0    ]
[                         ]
[0    0      0      - y  ]
220 ms
===== jacobson normal form =====
[ - y   1      0      0  ]
[                         ]
[ 0     - y    0      0  ]
[                         ]
[ 0     0      - y    0  ]
[                         ]
[ 0     0      0      - y]
240 ms
===== jordan normal form =====
[ - y   1      0      0  ]
[                         ]
[ 0     - y    0      0  ]
[                         ]
[ 0     0      - y    0  ]
[                         ]
[ 0     0      0      - y]
180 ms
 ===== transformation matrices from the rational normal form
```

```
        to jordan normal form =====

920 ms

 ===== transformation matrices to jordan normal form ====

110 ms

3180 ms
%
```

5次

```
clear a5$

matrix a5$

%-----------------------------------------------------------------$

a5:=
mat((4*(23*y**2 + 26*y + 7), - 115*y**2 - 130*y - 34,29*y**2 + 32*y + 8,
 - 9*y**2 - 9*y - 2,y*(3*y + 2)),(4*(22*y**2 + 26*y + 7),
2*( - 55*y**2 - 65*y - 17),4*(7*y**2 + 8*y + 2), - 9*y**2 - 9*y - 2,
y*(3*y + 2)),(3*(24*y**2 + 32*y + 9),3*( - 30*y**2 - 40*y - 11),
2*(12*y**2 + 15*y + 4), - 9*y**2 - 9*y - 2,y*(3*y + 2)),(24*(2*y**2 + 3*y + 1),
30*( - 2*y**2 - 3*y - 1),8*(2*y**2 + 3*y + 1), - 6*y**2 - 9*y - 2,2*y*(y + 1)),
(12*(2*y + 1),15*( - 2*y - 1),4*(2*y + 1), - 3*y - 1,0))$

jordan(a5)$

===== rational normal form =====

[        2                  ]
[0   - y    0    0      0   ]
[                            ]
[1   - 2*y  0    0      0   ]
[                            ]
[                     2      ]
[0    0     0   - y     0   ]
[                            ]
[0    0     1   - 2*y   0   ]
[                            ]
[0    0     0    0      - y]

 ===== transformation matrices from the input matrix
        to its rational normal form =====

4761 ms

===== elementary divisors =====

     2                    2
{var0  + 2*var0*y + y  ,

     2                    2
 var0  + 2*var0*y + y  ,

 var0 + y}

===== simple elementary divisors =====

{{var0 + y,2},

 {var0 + y,2},

 {var0 + y,1}}

450 ms

===== second natural normal form =====

[        2                  ]
[0   - y    0    0      0   ]
[                            ]
[1   - 2*y  0    0      0   ]
[                            ]
[                     2      ]
[0    0     0   - y     0   ]
[                            ]
[0    0     1   - 2*y   0   ]
[                            ]
[0    0     0    0      - y]

280 ms
```

```
===== jacobson normal form =====
[ - y   1     0     0     0  ]
[                            ]
[ 0     - y   0     0     0  ]
[                            ]
[ 0     0     - y   1     0  ]
[                            ]
[ 0     0     0     - y   0  ]
[                            ]
[ 0     0     0     0     - y]
270 ms
===== jordan normal form =====
[ - y   1     0     0     0  ]
[                            ]
[ 0     - y   0     0     0  ]
[                            ]
[ 0     0     - y   1     0  ]
[                            ]
[ 0     0     0     - y   0  ]
[                            ]
[ 0     0     0     0     - y]
250 ms
 ===== transformation matrices from the rational normal form
       to jordan normal form =====
1160 ms
 ===== transformation matrices to jordan normal form ====
449 ms
7620 ms
%
```

C.2.2 応用の計算結果

REDUCE3.5 による出力結果をそのまま示す。

C.2.2.1 線形差分方程式

以下の計算例では、便宜上、差分方程式の係数行列のべき指数 m は pp として表している。

[データ 3.3]

3 次

```
matpow(pp);
===== coefficient matrix of solution of system of
      difference equations of first order =====
410 ms
[       pp                    ]
[( - 1)        0          0    ]
[                              ]
[                   pp         ]
[   0       ( - 1)         0   ]
[                              ]
[                          pp  ]
[   0          0       ( - 1)  ]
```

4 次

```
matpow(pp);
===== coefficient matrix of solution of system of
      difference equations of first order =====
```

```
721 ms
mat(( - 28*( - 1)^pp *pp + ( - 1)^pp ,35*( - 1)^pp *pp, - 7*( - 1)^pp *pp,0),
    ( - 24*( - 1)^pp *pp,30*( - 1)^pp *pp + ( - 1)^pp , - 6*( - 1)^pp *pp,0),
    ( - 8*( - 1)^pp *pp,10*( - 1)^pp *pp, - 2*( - 1)^pp *pp + ( - 1)^pp ,0),
    (0,0,0,( - 1)^pp ))
```

5次

```
matpow(pp);
===== coefficient matrix of solution of system of
      difference equations of first order =====
1121 ms
mat(( - 225*( - 1)^pp *pp + ( - 1)^pp ,279*( - 1)^pp *pp, - 69*( - 1)^pp *pp,
     20*( - 1)^pp *pp, - 5*( - 1)^pp *pp),
    ( - 220*( - 1)^pp *pp,273*( - 1)^pp *pp + ( - 1)^pp , - 68*( - 1)^pp *pp,
     20*( - 1)^pp *pp, - 5*( - 1)^pp *pp),
    ( - 195*( - 1)^pp *pp,243*( - 1)^pp *pp, - 63*( - 1)^pp *pp + ( - 1)^pp ,
     20*( - 1)^pp *pp, - 5*( - 1)^pp *pp),
    ( - 144*( - 1)^pp *pp,180*( - 1)^pp *pp, - 48*( - 1)^pp *pp,
     16*( - 1)^pp *pp + ( - 1)^pp , - 4*( - 1)^pp *pp),
    ( - 36*( - 1)^pp *pp,45*( - 1)^pp *pp, - 12*( - 1)^pp *pp,4*( - 1)^pp *pp
     , - ( - 1)^pp *pp + ( - 1)^pp ))
```

C.2.2.2 定係数線形微分方程式

以下の計算例では、便宜上、微分方程式の変数 t は tt として表している。

[データ 3.3]

3次

```
slvdif();
===== coefficient matrix of solution of system of
      differential equations of first order =====
1141 ms
'tt' is the variable of the equations
[  1                  ]
[-----     0       0  ]
[  tt                 ]
[ e                   ]
[                     ]
[           1         ]
[  0     -----     0  ]
[           tt        ]
[          e          ]
[                     ]
[                  1  ]
[  0       0     -----]
```

```
[                 tt ]
[                e   ]
```

4次 slvdif();

```
===== coefficient matrix of solution of system of
      differential equations of first order =====
2151 ms
'tt' is the variable of the equations
[ 28*tt + 1        - 35*tt        7*tt          ]
[-----------     ----------      ------      0   ]
[     tt            tt            tt             ]
[    e             e             e               ]
[                                                ]
[   24*tt        - 30*tt + 1      6*tt          ]
[  -------     --------------    ------      0   ]
[     tt            tt            tt             ]
[    e             e             e               ]
[                                                ]
[   8*tt           - 10*tt      2*tt + 1          ]
[  ------        ----------    ----------    0   ]
[   tt             tt            tt             ]
[  e              e             e               ]
[                                                ]
[                                            1   ]
[    0             0             0         -----]
[                                            tt  ]
[                                           e    ]
```

5次 slvdif();

```
===== coefficient matrix of solution of system of
      differential equations of first order =====
3781 ms
'tt' is the variable of the equations
      225*tt + 1    - 279*tt   69*tt    - 20*tt   5*tt
mat((------------,-----------,-------,----------,------),
          tt            tt        tt        tt       tt
         e             e         e         e        e

     220*tt    - 273*tt + 1   68*tt    - 20*tt   5*tt
    (--------,--------------,-------,----------,------),
       tt           tt          tt        tt       tt
      e            e           e         e        e

     195*tt    - 243*tt   63*tt + 1    - 20*tt   5*tt
    (--------,-----------,-----------,----------,------),
       tt          tt          tt         tt       tt
      e           e           e          e        e

     144*tt    - 180*tt   48*tt    - 16*tt + 1   4*tt
    (--------,-----------,-------,--------------,------),
       tt          tt        tt         tt         tt
      e           e         e          e          e

     36*tt    - 45*tt   12*tt    - 4*tt   tt + 1
    (-------,----------,-------,---------,--------))
       tt        tt        tt        tt       tt
      e         e         e         e        e
```

Inches
1 2 3 4 5 6 7 8
cm
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Kodak Color Control Patches
© Kodak, 2007 TM: Kodak
Blue
Cyan
Green
Yellow
Red
Magenta
White
3/Color
Black
Kodak Gray Scale
C
Y
M
© Kodak, 2007 TM: Kodak
A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19