数理解析研究所講究録
1514巻 2006年 94-100

有理式を要素とする行列式の計算法

梅田 恭 (Yasushi Umeda)*
筑波大学大学院数理物質科学研究科
GRADUATE SCHOOL OF PURE AND APPLIED SCIENCES, UNIVERSITY OF TSUKUBA
佐々木建昭 (Tateaki Sasaki)†
筑波大学 数学系
INSTITUTE OF MATHEMATICS, UNIVERSITY OF TSUKUBA

Abstract

有理式を要素とする行列式で、特に高次のものを取り扱う。有理式要素の行列式計算は従来の研究の盲点で、著名システムでも想定外であることを指摘する。次に、分数なしガウス消去法と小行列式展開法について、分母を独立変数で置き換える手法により、真正直に計算する方法より3〜6倍の効率化が達成できることを示す。

1 はじめに

多項式を要素とする行列式の計算法については、Smit[Smi81] による小行列式展開法の効率化など、1970〜1980年代に多くの研究がなされてきた。しかし、有理式を要素とする行列式の計算法についての研究はほとんど行われていない。実際に市販の著名な数式処理システムを使って有理式の行列式計算を行うと、世界的に流布している数式処理システムMapleでさえ非常に計算時間がかかり、有理式の行列式は想定外のようであった。最も著名なMathematicaは、計算制御をうまく行えば常識的な答えを出すが、制御を行わなければ不十分な答えを出力する。一方、国産の数式処理システムGAL(General Algebraic Language/Laboratory)は計算制御を行わないでも常識的に計算する。これはGALが計算機内部で有理式を巧みに表現し処理しているからである。しかし、GALを用いても次数が高くなれば計算時間は指数関数的に増えてしまうので、高次の行列式で効率的に計算する方法を研究する必要がある。

本稿では、多項式要素の行列式の代表的な算法であるBareiss[Bar68]による「分数なしガウス消去法」と「小行列式展開法」を取り上げ、有理式要素の行列式に適用することを考える。多項式要素用の算法としては他に補間法等があるが、多変数多項式に対しては効率的ではないので本稿では取り上げない。

有理式を要素とする行列をランダムに生成して行列式を計算すると、テストした範囲ではMapleはGALより100〜1000倍遅い。Mapleが入力された分数式を通分して計算するのに対し、GALは分数式を通分することなく計算するからである。我々はGALのこの性質に着目して有理式を分数式の和で表現し、有理式要素の行列式の計算に多項式用算法を使用できるように、有理式の分母多項式を全て独立変数の逆数で置き換える (ただし、同じ分母多項式には同じ独立変数を用いる)。これにより有理式は多項式に変換されるので、有理式要素の行列式計算で問題が生じる分数なしガウス消去法が無修正で適用できる。また、小行列式展開法では、独立変数への置き換えにより一番時間のかかる部分が多項式演算で実行できるので、行列式の計算の効率化が期待できる。

*umeda@math.tsukuba.ac.jp
†sasaki@math.tsukuba.ac.jp

この方法の有効性をテストするため、ランダムに生成した有理式の行列式について条件を変えて実験を行った。その結果、この工夫により平均で3〜6倍速くなることを確認できた。

2 各システムの計算比較とその考察

各システムで計算に使用した算法は以下の通りである。( ) のキーワードは算法を表し、表中で用いる。

- GAL : 小行列式展開法 (Minor)
- Maple : 小行列式展開法を指定して計算する場合 (Minor) と分数なしガウス消去法を指定して計算する場合 (Gauss) の二つについて計測した。
- Mathematica : Mathematica は行列式計算に Det コマンド (Det) を装備している。計算結果を見ると不十分な場合があったので、Det コマンドの出力に Expand コマンド (Expand) を作用させた場合と Simplify コマンド (Simplify) を作用させた場合についても計測した。

実験1 : 計算結果がシンプルな有理式行列式

まず、行列式のゼロ判定を考慮して、計算結果が数項になるような有理式行列式について実験を行った。実験に使用する行列として、4変数で次数4および項数4の有理式を要素とする5次〜7次の行列を次のように生成した。[1] 第1行として適当な有理式要素の行を生成する。[2] 第1行目にランダムに生成した数値を乗じた行を残りの各行とする。[3] 各対角要素にランダムに生成した数値を加える。

各システムにおける計算時間を表1に示す (縦軸は行列の次数を表す)。表の数値は10個の異なる行列式の計算に要した時間の平均 (単位 : 秒) を表す。

	GAL	Maple		Mathematica		
#N	Minor	Minor	Gauss	Det	Expand	Simplify
5	0.02	9.03	1000 over	0.03	0.03	0.33
6	0.12	4.54	1000 over	0.04	0.05	0.17
7	0.91	2.76	1000 over	0.07	0.08	0.25

表 1: 計算結果がシンプルな有理式行列式の計算時間 (秒)

(1) GAL と Maple の比較

計算時間を比較すると GAL は Maple より数倍以上速い。GAL が分数式の和の形で結果を出力するのに対し、Maple は通分して結果を出力する。Maple の計算途中の数式を printlevel コマンドで出力すると、Maple は入力された有理式を通分してから指定された算法で計算していることがわかる。そのため分子と分母の多項式が非常に大きくなり、さらに分数なしガウス消去法による計算では中間式膨張を引き起こしていたため、計算時間が膨大になってしまったと考えられる。

(2) GAL と Mathematica の比較

Det コマンドによる計算結果は分子が単項の分数式の和の形で表現されており、Expand コマンドを作用させても Det の計算結果と変わらない。GAL では行列の次数が上がるにつれて計算時間が増加するが、Mathematica ではほとんど変わらない。Mathematica はシンプルな有理式行列式を巧みに計算していると言える。一方、Det コマンドの出力に Simplify コマンドを作用させると、Det コマンドで計算された分数式の和の有理式を通分したり、共通因子を取り出している。そのため Det コマンドによる計算時間よりも時間を要したが、計算結果がシンプルなため行列の次数を上げても計算時間はほとんど同じだった。

実験 2 : ランダムに生成した有理式行列式

次に、ランダムに生成した有理式行列式で実験した。4 次と 5 次の行列では 3 変数で次数 4 および項数 2 の有理式を生成し、6 次の行列では 3 変数で次数 2 および項数 2 の有理式を生成して実験を行った。

	GAL	Maple		Mathematica		
#N	Minor	Minor	Gauss	Det	Expand	Simplify
4	0.03	118.62	1000 over	0.07	0.34	85.63
5	0.09	192.33	1000 over	0.05	1.02	276.03
6	0.32	40.86	1000 over	0.15	1.86	145.51

表 2: ランダムに生成した有理式行列式の計算時間 (秒)

(1) GAL と Maple の比較

計算時間を比較すると GAL は Maple より 100 倍以上速い。実験 1 と比べてこれほどの大きな差が出た原因は Maple が有理式を通分してから計算することによると考えられる。実験 2 では計算結果の有理式の項数が非常に大きいので、通分するための時間も大きくなる。

(2) GAL と Mathematica の比較

Mathematica の Det コマンドによる計算時間はやはり GAL より速いが、その計算結果を見ると不十分な答えになっていることがわかった。その答えのうち一つの項を取り出して以下に示す。

$$\frac{1}{x^3}\left(\left(\frac{35xy^2}{41}+\frac{43yz^2}{82}\right)\left((5x^3+78y^4+61yz^3)(\frac{4x^3+\frac{51y^2}{4}+\frac{33xy^2}{4}}{3xy^2+19y^3}+\frac{\frac{145x^2y}{2}+46xz}{23x+17y^3})\right)\Big/\right.$$

$$\left.\left(21x^3+29xy^3\right)-\frac{1}{x^2}\left((\frac{83x^3y}{17}+\frac{67y^3z}{17})(\frac{56x^3}{31y+18x^3y+26x^3z}+\frac{33x^2z^2}{4y^4+24xyz^2+73y^2z^2})\right)\right)$$

上式は分子分母ともに有理式になっていて、特に分子は有理式によって有理式を表現している。すなわち、行列式を展開すると項数が大きくなりそうな場合には Mathematica は展開せずに形式的にまとめて計算するので、ユーザの欲する結果にならない。そこで計算結果にコマンド Expand と Simplify を作用させて、さらに計算を行わせてみた。すると、Expand を作用させた場合では分子部分のみが単項の分数式の和の形で表されたが、計算時間は GAL より数倍以上遅くなった。一方、Simplify を作用させた場合には、非常に計算時間がかかったにもかかわらず、計算結果は Det の場合とほとんど変わらなかった。

3 新しい工夫

以下、有理式行列式の計算の研究を行うに際して GAL を用いるため、有理式は分数式の和の形で表現されることを前提とする。

3.1 有理式行列式計算における多項式用算法の問題点

小行列式展開法は GAL の行列式計算ルーチンで用いられており、2 章の実験でも使用したが、そのまま有理式行列式に適用できる。しかし、分数なしガウス消去法を有理式行列式に適用するには注意が必要である。なぜなら分数なしガウス消去法をそのまま用いると、そのアルゴリズムから有理式による有理式の除算が必要だが、有理式は分数式の和の形で表現されていると仮定したので、この除算をアルゴリズム化するためには少なくとも次の二つを考慮しなければならない。

(i) 分数式の項順序を除算に矛盾しないように定義する必要がある。

(ii) 分数式の分子と分母が簡約された場合、除算では簡約された因子を再生する必要がある。

上記の (i) と (ii) について例を用いて説明する。

(i) 分数式の項順序の定義が簡単でないこと

例 3.1 GAL では有理式の項順序は分母多項式の辞書式順序 $\prec_{\mathrm{lex}}$ によって定義されている。多項式に対しては、$\prec_{\mathrm{lex}}$ は先頭の項から順に、まず次数を比較して大きい方を高順位と定め、次数が同じ場合は数係数で大小を定める。たとえば多項式 x^2 と x^2-x+1 では、$x^2 \prec_{\mathrm{lex}} x^2-x+1$ と順序付けられる。しかし、両辺に (x^2+x) をかけると項順序は逆転する。

$$x^4+x=(x^2+x)(x^2-x+1) \prec_{\mathrm{lex}} (x^2+x)(x^2)=x^4+x^3 \tag{1}$$

したがって、次の例が示すように、被除有理式と除有理式の間で最大順位の分数式が対応しなくなる場合がある (次式では波線部を付した分数式どうしが対応する)。

$$\left(\frac{1}{x^2+x}+\cdots\right)\left(\frac{1}{\underset{\sim}{x^2-x+1}}+\frac{1}{x^2}+\cdots\right)=\frac{1}{x^4+x^3}+\frac{1}{\underset{\sim}{x^4+x}}+\cdots \tag{2}$$

(ii) 分子と分母の簡約が行われる場合

例 3.2 次の行列 M の行列式 $|M|$ を分数なしガウス消去法で計算する。

$$M=\begin{pmatrix} x^2 & \frac{1}{x^4-x^3} & \frac{x+3}{x-2} \\ 1 & \frac{1}{x^2} & x-\frac{x^2}{x+1} \\ x & \frac{x}{x-1} & x+1 \end{pmatrix} \Rightarrow |M| = \begin{vmatrix} 1-\frac{1}{\underset{\sim}{x^4-x^3}} & \underline{\underline{x^3}}-\frac{x^4}{x+1}-\frac{x+3}{x-2} \\ \frac{x^3}{x-1}-\underline{\underline{\frac{1}{x^3-x^2}}} & \underset{\sim}{x^3+x^2}-\frac{x^2+3x}{x-2} \end{vmatrix} \Big/ x^2 \tag{3}$$

波線部どうしの積と二重下線部どうしの積に注意して 2 回目の分数なしガウス消去を行うと次式になる。

$$\begin{aligned} |M| &= \left((1-\tfrac{1}{\underset{\sim}{x^4-x^3}})(\underset{\sim}{x^3+x^2}-\tfrac{x^2+3x}{x-2})-(\tfrac{x^3}{x-1}-\underline{\underline{\tfrac{1}{x^3-x^2}}})(\underline{\underline{x^3}}-\tfrac{x^4}{x+1}-\tfrac{x+3}{x-2})\right)/x^2 \\ &= \left(x^3+x^2+\tfrac{x^3+3x}{x-2}-\underset{\sim}{\tfrac{x+1}{x^2-x}}-\tfrac{x^6-x}{x-1}+\underline{\underline{\tfrac{x}{x-1}}}+\tfrac{x^7}{x^3-1}-\tfrac{x^2}{(x-1)(x+1)}+\tfrac{x^4+3x^3}{(x-2)(x-1)}\right)/x^2 \end{aligned}$$

アルゴリズムによれば、上式右辺の分子は x^2 で割り切れるので、分子の各項は x^2 を因子として取り出すことができるはずである。しかし、波線部と二重下線部の項は x^2 を因子として持たない。

上例では単項式が約分されたのだが、有理式の計算では分子分母の間で簡約を行うのが普通である。この簡約が行われると、簡約された因子を再生しない限り、有理式による有理式の除算がうまくいかない。

3.2 新しい工夫について

有理式を要素とする行列式の計算に分数なしガウス消去法を適用するには、有理式による有理式の除算が必要だが、例 3.1 と例 3.2 に示した問題点から正しく動作する除算のプログラムは複雑になってしまう。

そこで、入力された有理式行列を多項式行列に変換して多項式用算法を適用する。具体的には、入力行列の有理式全体が m 個の異なる分母多項式を持つとして、それらの分母多項式を内部表現の順序則に従って並べたものを $D_1, D_2, \cdots, D_m$ とするとき、これらをそれぞれ $1/V_1, 1/V_2, \cdots, 1/V_m$ で置き換える。ここで、$V_1, V_2, \cdots, V_m$ は $V_1 \succ V_2 \succ \cdots \succ V_m \succ$ [他の変数] なるよう順序付けられた独立変数である。

この工夫は非常に簡単だが、この置き換えにより有理式を多変数多項式に変換できるので、多項式用算法を無修正で適用できる。すなわち、分数なしガウス消去法が有理式による有理式の除算を導入せず有理式行列式に適用できる。小行列式展開法についても、有理式演算は多項式演算に比べて非常に重いので、この工夫による計算の方が効率的と考えられる。

4 実験

実験に使用した算法は以下の4つである。

- 有理式要素のままで計算する小行列式展開法 (R-Minor)。
- 分母置き換え小行列式展開法。独立変数を分母に戻すときに分母因子の積を展開する (S-Minor)。
- 分母置き換え小行列式展開法。独立変数を分母に戻す時に分母因子の積を展開しない (U-Minor)。
- 分母置き換え効率的ガウス消去法 (E-Gauss)。

行列要素の生成用パラメータ

行列要素の有理式は次の4つのパラメータを指定してランダムに生成する。

- #N : 行列の次数
- #D : 行列に現れる異なる分母多項式の個数
- #V : 行列に現れる変数の個数
- #T : 一つの有理式に含まれる分数式の個数

上記の4つの算法とパラメータのもと、下記の2つの実験を行った。

実験 3 各行列要素として2変数と4変数の有理式をランダムに生成する。各行列式の計算時間を数秒～百秒以内におさめるため、$\#N \leq 6$, $\#T \leq 4$ に制限して、#D=5,10 の場合について実験を行った。この場合、分母を独立変数で置き換えた多項式行列式の項数は数万～数十万 (#N=6 の場合) となり、計算結果の有理式の分数式の個数は数百～数千になる。

実験 4 有名な行列式である Symmetric Toeplitz 行列式について実験を行う。実験3と同様に、行列要素として2変数と4変数の有理式を生成して$\#N \leq 6$, $\#T \leq 4$ に制限し、#D=5,10,15 で実験を行った。なお、分母を独立変数で置き換えた多項式行列式の項数と計算結果の有理式の分数式の個数は、パラメータ値が同じ場合は実験3とほぼ同じになる。

表3～12に各実験における行列式の計算時間を示す (縦軸は#N、横軸は#T を表す)。表中の数値は10個の異なる行列式の計算に要した時間の平均 (単位：秒) を表す。また、表中の mem-ov はメモリオーバーにより計算できなかったことを示し、no-mat は指定されたパラメータ値では行列が構成できないことを示す。

4.1 実験3：有理式をランダムに生成する場合 (単位：秒)

	#T = 2				#T = 4			
#N	R-Minor	S-Minor	U-Minor	E-Gauss	R-Minor	S-Minor	U-Minor	E-Gauss
4	0.065	0.022	0.021	0.224	0.327	0.078	0.073	1.314
5	0.581	0.178	0.174	9.686	2.195	0.749	0.778	62.069
6	11.034	1.436	1.546	365.492	16.414	3.227	3.384	591.706

表 3: 2変数で、異なる分母の個数=5 (#V=2 , #D=5)

	#T = 2				#T = 4			
#N	R-Minor	S-Minor	U-Minor	E-Gauss	R-Minor	S-Minor	U-Minor	E-Gauss
4	0.103	0.056	0.057	0.867	0.631	0.333	0.353	7.093
5	1.872	0.820	0.911	105.073	8.772	4.672	4.902	mem-ov
6	24.750	10.597	11.236	mem-ov	90.237	40.764	41.570	mem-ov

表 4: 4変数で、異なる分母の個数=5 (#V=4 , #D=5)

#N	#T = 2				#T = 4			
	R-Minor	S-Minor	U-Minor	E-Gauss	R-Minor	S-Minor	U-Minor	E-Gauss
4	0.133	0.039	0.038	0.492	1.282	0.221	0.213	4.325
5	1.920	0.456	0.443	42.284	13.606	3.325	3.462	mem-ov
6	23.383	5.550	5.569	mem-ov	225.269	41.679	43.188	mem-ov

表 5: 2 変数で、異なる分母の個数=10 (#V=2 , #D=10)

#N	#T = 2				#T = 4			
	R-Minor	S-Minor	U-Minor	E-Gauss	R-Minor	S-Minor	U-Minor	E-Gauss
4	0.214	0.073	0.064	0.710	1.426	0.730	0.782	23.787
5	3.510	1.418	1.508	mem-ov	37.196	20.795	21.605	mem-ov
6	84.352	27.522	30.308	mem-ov	mem-ov	mem-ov	mem-ov	mem-ov

表 6: 4 変数で、異なる分母の個数=10 (#V=4 , #D=10)

4 変数までは S-Minor,U-Minor ともに R-Minor と比べて平均で 3〜4 倍効率化できた。なお、S-Minor は分母多項式を展開する分、U-Minor より遅いと考えられるが、そうなっていない。この理由は今の例では分子も分母も数係数が大きな桁となるので (10〜20 桁)、U-Minor の分子多項式の正規化に時間がかかったと考えられる。

4.2 実験 4 : Symmetric Toeplitz 行列式 (単位 : 秒)

#N	#T = 2				#T = 4			
	R-Minor	S-Minor	U-Minor	E-Gauss	R-Minor	S-Minor	U-Minor	E-Gauss
4	0.134	0.028	0.028	0.159	0.962	0.102	0.101	1.164
5	2.483	0.282	0.285	4.883	4.770	0.898	0.888	29.566
6	18.596	1.827	1.832	116.024	42.918	5.328	5.080	426.807

表 7: 2 変数で、異なる分母の個数=5 (#V=2 , #D=5)

#N	#T = 2				#T = 4			
	R-Minor	S-Minor	U-Minor	E-Gauss	R-Minor	S-Minor	U-Minor	E-Gauss
4	0.285	0.098	0.093	0.621	1.410	0.534	0.554	7.481
5	4.231	1.420	1.447	48.027	17.393	7.008	7.009	mem-ov
6	48.165	13.531	13.400	mem-ov	201.473	66.413	65.211	mem-ov

表 8: 4 変数で、異なる分母の個数=5 (#V=4 , #D=5)

#N	#T = 2				#T = 4			
	R-Minor	S-Minor	U-Minor	E-Gauss	R-Minor	S-Minor	U-Minor	E-Gauss
4	no-mat	no-mat	no-mat	no-mat	2.006	0.296	0.282	2.346
5	6.646	0.573	0.566	9.294	37.120	4.642	4.589	mem-ov
6	40.948	5.356	4.910	mem-ov	351.865	41.708	41.206	mem-ov

表 9: 2 変数で、異なる分母の個数=10 (#V=2 , #D=10)

	#T = 2				#T = 4			
#N	R-Minor	S-Minor	U-Minor	E-Gauss	R-Minor	S-Minor	U-Minor	E-Gauss
4	no-mat	no-mat	no-mat	no-mat	3.274	1.137	1.151	12.558
5	8.633	2.093	2.040	60.455	93.324	24.350	26.198	mem-ov
6	247.826	32.491	33.031	mem-ov	mem-ov	mem-ov	mem-ov	mem-ov

表 10: 4 変数で、異なる分母の個数=10 (#V=4 , #D=10)

	#T = 4				#T = 6			
#N	R-Minor	S-Minor	U-Minor	E-Gauss	R-Minor	S-Minor	U-Minor	E-Gauss
4	3.021	0.473	0.413	3.227	8.892	1.526	1.306	17.141
5	56.148	8.518	7.768	mem-ov	222.863	29.939	27.691	mem-ov
6	mem-ov	105.325	100.535	mem-ov	mem-ov	mem-ov	mem-ov	mem-ov

表 11: 2 変数で、異なる分母の個数=15 (#V=2 , #D=15)

	#T = 4				#T = 6			
#N	R-Minor	S-Minor	U-Minor	E-Gauss	R-Minor	S-Minor	U-Minor	E-Gauss
4	4.415	1.400	1.285	13.253	16.098	5.560	5.637	108.452
5	160.603	45.503	41.977	mem-ov	mem-ov	mem-ov	mem-ov	mem-ov
6	mem-ov	mem-ov	mem-ov	mem-ov	mem-ov	mem-ov	mem-ov	mem-ov

表 12: 4 変数で、異なる分母の個数=15 (#V=4 , #D=15)

こちらでは平均で 5〜6 倍速かった。実験 3 と比べると、S-Minor と U-Minor に比べて相対的に R-Minor の計算時間が増えている。行列式は下の行から順に小行列式を計算しており、Symmetric Toeplitz 行列式では、計算の初期の段階から多数個の分数式を扱うことになる。そのため、ランダム行列に比べて R-Minor が余計に時間を喰ったと思われる。

5 まとめ

有理式を分数式の和の形で表現する方法は有理式行列式の計算に非常に有効である。次に、分母多項式を独立変数に置き換える手法により、分数なしガウス消去法が無修正で有理式行列式に適用可能となるが、分数なしガウス消去法は小行列式展開法に比べて圧倒的に非効率である。小行列式展開法については、有理式行列式に小行列式展開法を適用する場合よりも平均で 3〜6 倍効率化できた。

参 考 文 献

[Bar68] E.H. Bareiss: Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination. Mathematics of Computation, Vol. 22, No. 103, 1968, pp. 565-578.

[Smi81] J. Smit: A Cancellation Free Algorithm, with Factoring Capabilities, for the Efficient Solution of Large Sparse Sets of Equations. Proc. SYMSAC 1981, ACM, New York, pp. 146-154.

[SM82] T. Sasaki and H. Murao: Efficient Gaussian Elimination Method for Symbolic Determinants and Linear Systems. ACM Transaction on Mathematical Software, Vol. 8, No. 3, 1982, pp. 277-284.

[Sas81] 佐々木建昭: 数式処理 情報処理学会, オーム社発行, 東京, 1981, pp. 21-32.